GILBERT: A Reimagined Biomedical Data Storage and Analysis Pipeline

A Major Qualifying Project submitted to the faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
degree of Bachelor of Science

Submitted by:

Mark Bernardo
Jonathan Dang
Andrew Markoski
Bridget McLean
Samantha Peznola
Fay Whittall

Submitted to:

Professor Mark Claypool
Worcester Polytechnic Institute

Professor Adam Lammert
Worcester Polytechnic Institute

Dr. Kajal Claypool
MIT Lincoln Laboratory

Submitted on: 03/18/2021

Abstract

Due to the lack of a pipeline that can store and analyze a wide variety of biomedical data, our team built GILBERT[1]. GILBERT is a web application that uses Django and stores its data in PostgreSQL. It contains four main components considered essential elements of a data analysis pipeline: a system that controls which users have access to which studies, an uploader which allows users to add studies to GILBERT's database, an interface that displays study information and metadata, and a feature that analyzes studies in the database through statistical and graphical means.

To evaluate GILBERT, twenty-seven participants responded to two surveys and attended a virtual evaluation session. In the evaluation sessions, a team member read tasks to the participant while another team member scored the participant on their behavior, their ability to complete the tasks, and understanding of the system. Overall, tasks had a high completion rate. Participants rated GILBERT as easy to use and easy to understand. Differences in ratings were insignificant between different majors, amounts of experience with large amounts of data, and experience with other data handling tools such as MATLAB, Python, and SQL.

These results show that GILBERT is an effective tool for biomedical data handling. Future work could make GILBERT's uploader more flexible, allow for easier cross-study analysis, and add more options to create charts.

[1] GILBERT - Greatest Instrument for Leveraging Biomedical Electronic Resources Together. GILBERT is identified by the green frog logo, as seen throughout the paper and the web application.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

In the past twenty years, the growth of data has been widespread across a multitude of industries. Computing power has paralleled this growth, becoming stronger and more accessible, allowing industries to leverage data-driven approaches to problem-solving. In addition to bringing about new discoveries and research endeavors, the data boom has also brought with it new challenges around ways to organize, store, manage, and process such large amounts of data (Chen, 2014).

The healthcare and medical device industries have seen great advances in this new data-rich era. Large datasets and increased data sharing capabilities allow for improved diagnostic predictions, disease tracking, health record management, and clinical trials (Baljak, 2018). Furthermore, the richness and diversity of physiological data collected by different devices is beneficial to many areas of research. Data sharing is a rising phenomenon in the healthcare industry, but it is a delicate process under patient privacy regulations and standards.

Physiological data is not exempt from the challenges of big data. Databases are implemented to manage and store large data sets, usually electronically in a computer system (Bella & Nagle, 2014). Database management systems (DBMS) are often responsible for controlling these databases and acting as a liaison between the user and the data (Esakkirajan & Sumathi, 2007). Scripting languages offer a level of data organization that allows for storage, extraction, and manipulation. Methods of data processing and analysis close the gap between data inputs and meaningful outputs.

To allow for fluidity from data storage through to analysis, researchers have shown interest in comprehensive data analysis pipelines. These pipelines allow permitted users to upload data into databases, generate data queries for filtering, and select from various options of data analysis, such as summary statistics, tables, and data visualizations. Comprehensive pipelines are preferable to data management, but they have some drawbacks. Often, they work with one type of data collected from one study, limiting cross-study research and data sharing due to inefficient upload methods and limited database schemas. Analysis options are also often narrow and lack diagnostics, predictions, and the ability to generate patient statuses (Shin, 2013; Dremio, 2020). Data uploading could be improved in many pipelines with a more automated method and simpler user interface (UI).

In understanding the hindrances of current tools and the motivation to find new ways to handle big data, our team created a data analysis pipeline fit for a wider variety of biomedical signals, named GILBERT (Greatest Instrument for Leveraging Biomedical Electronic Resources Together). As seen in *Figure 1*, our pipeline ingests datasets from different studies to promote data sharing while maintaining patient privacy standards. The various study data is populated into a centralized database that works in conjunction with different physiological data signals including discrete, analog, and digital. Our front end gives researchers and clinicians the ability to query data across one or many studies. From here, the user has additional control over the analytical method and desired output. These outputs include descriptive metrics about the data, including descriptive metrics, analysis-ready datasets, and a variety of data visualizations.

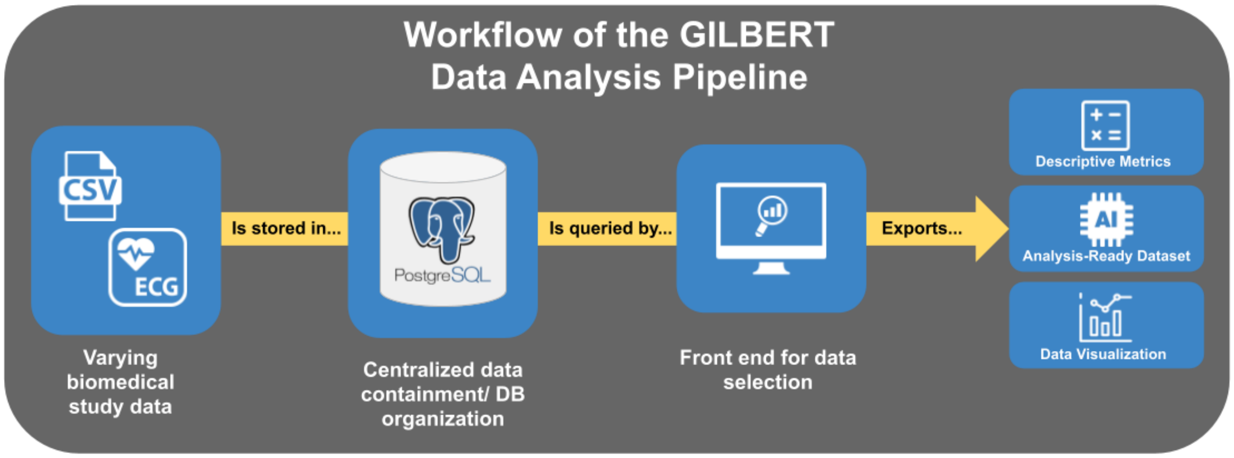


Figure 1. Data analysis pipeline for biomedical signals.

Through a user study in which twenty-seven users completed tasks related to uploading, querying, and analyzing data, we show that GILBERT is easy to use and easy to understand. Also, differences are insignificant between majors, amounts of experience with large amounts of data, and experience with other data handling tools such as MATLAB, Python, and SQL. We believe GILBERT is a strong foundation for a data pipeline and provide recommendations for improving its UI and flexibility.

Chapter 2 discusses our relevant background research on industry standards, data handling and analysis tools, and existing data pipelines and shows how this research influenced our development of GILBERT. Chapter 3 outlines our methodology, including design decisions, product development schedules, and our implementation of the Agile Development methodology. Chapter 4 details GILBERT’s internal design and features. Chapter 5 discusses our evaluation process and displays the results of our evaluation. Finally, Chapter 6 demonstrates what the results in Chapter 5 mean in relation to GILBERT, both in how it can be improved and how it can benefit the world of biomedical data analysis.

Chapter 2. Related Work

This section discusses the background information we collected before and during the completion of our project, including standards for data collection, biomedical data storage, technologies we utilized or considered, and examples of data pipelines in the real world.

2.1. Standards for Research Data

When using data with a medical device, especially data pertaining to patient information, it is crucial to adhere to regulations and standards to ensure patient safety and confidentiality as well as device success.

2.1.1 FDA Guidance for Software as a Medical Device (SaMD)

The Food and Drug Administration (FDA) is responsible for ensuring the safety and effectiveness of products relating to public health such as medical products and foods. Their goal is to provide users with "accurate, science-based information they need to use medical products and food to maintain and improve their health" (U.S. Food and Drug Administration, 2020).

Included in their definition of medical products is Software as a Medical Device (SaMD). SaMD is defined as "software intended to be used for one or more medical purposes that perform these purposes without being part of a hardware medical device" (Software as a Medical Device, 2020). The stages of SaMD functionality begin with the ingestion and storage of patient data, followed by processing and manipulation by programmed algorithms, and the final retrieval of defined outputs that fulfill the device's purpose (IMDRF, 2014). As seen in *Figure 2,* inputs into a SaMD specifically include results from labs, medical image signals, physiological statuses, and symptoms. These inputs are fed into algorithms specific to the SaMD along with reference information such as other data, knowledge bases, rules, and criteria. The computing algorithm may include equations, inference engines, and model-based logic. The output of the SaMD specific algorithm are defined outputs that assist the device in accomplishing its intended use.

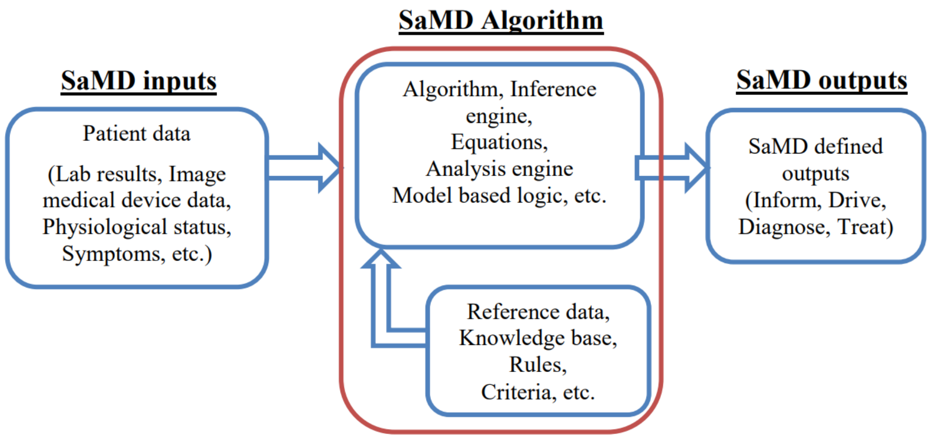

SaMD Algorithm
SaMD inputs
Patient data
(Lab results, Image medical device data, Physiological status, Symptoms, etc.)
Algorithm, Inference engine, Equations, Analysis engine Model based logic, etc.
Reference data, Knowledge base, Rules, Criteria, etc.
SaMD outputs
SaMD defined outputs (Inform, Drive, Diagnose, Treat)

Figure 2. SaMD flowchart (IMDRF, 2017).

There are three intended use categories of SaMDs: to treat or to diagnose, to drive clinical management, and to inform clinical management. Driving clinical management indicates that the information will be used immediately to diagnose a disease, treat a patient, or identify early signs of a developing disease or condition. Informing clinical management infers that "information provided by the SaMD will not trigger an immediate or near term action" (IMDRF, 2014). SaMDs are further categorized as either critical, serious, or non-serious depending on the seriousness of the healthcare situation or condition it would be used for (IMDRF, 2014).

FDA regulations to introduce a SaMD into practice are primarily concerned with risk assessment and mitigation, accurate documentation for the benefit of the user, and transparency of device quality (IMDRF, 2015). Manufacturers of SaMD should outline a methodical development process and the lifecycle stages of the software while taking into consideration the device's intended socio-technical environment. Socio-technical environments are differentiated by their location, workflow, responsibility, the technology they use, and their ambient conditions (IMDRF, 2015).

Risk management and mitigation aim to address all hazardous situations that may arise during the use of the device. Hazards are categorized by the origin of their risk: user-based, application-based, device-based, environment-based, and security-based. Estimation and evaluation of hazards must be followed by action plans to control them and monitoring methods. One of the primary risks associated with data-based devices is information security. In order to protect patient information, the SaMD needs to account for a variety of users with different access needs through authentication with user information. Deeper levels of user accessibility may be required if the device is connected to the internet or less-secure servers (IMDRF, 2015).

In the final stages of FDA approval, the quality of the device is accessed through *SaMD N41 Clinical Evaluation.* Two of the critical stages of this include clinical validation and analytical validation. Clinical validation verifies the clinical safety and performance of the device when used according to the outlined use and includes the computation of sensitivity, specificity, and odds ratios. Analytical validation verifies the device's ability to reliably generate accurate and precise output data according to specification outlined by the manufacturer and includes the calculations of accuracy, reproducibility, and repeatability. The clinical evaluation stage is crucial to patient safety and ensures that its purposes are fulfilled in accordance with the specifications (IMDRF, 2017).

2.1.2 NIH and Data Sharing Standards

The National Institutes of Health (NIH) is a division of the United States Department of Health and Human Services and is the largest biomedical research agency in the world (NIH, 2019). In order to advance new tools, treatments, and solutions, the NIH houses research institutes and provides funding to private sector medical ventures. To maintain the integrity of the institution and its research, there are numerous data standards that these ventures must uphold in order to receive funding. For example, the National Institute on Drug Abuse (NIDA) Sharing Policy outlines a timeframe in which supported studies must release their final research data so that other researchers can translate findings into knowledge, products, and procedures (NIH, 2019). Research funding organizations such as the National Institute of Technology (NIT) and the National Science Foundation (NSF) have policies that require data sharing (Hrynaszkiewicz, 2010). Data sharing is critical for cross-disciplinary scientific research and is crucial for future improvements in patient outcomes (Freymann, 2011).

2.1.3 HIPAA and Subject De-Identification

Even with encouragement from large funders and overall benefit to future research projects, little data sharing has occurred outside of prearranged research groups. This is partially attributable to fear of inadvertently violating the Health Insurance Portability and Accountability Act (HIPAA) and the lack of a commonly adopted method to de-identify subject data, where de-identification is defined as the assurance that no data object can be connected to a specific human subject (Hrynaszkiewicz 2010). In order to be in compliance with HIPAA, the standard for patient health information (PHI) safety in the United States, there are categories of identifiers that need to be removed or concealed with pseudonymous values.

These crucial categories for PHI include:

(1) Names

(2) Geographic subdivisions

(3) All elements of dates
(4) Telephone and fax numbers
(5) Web and Internet addresses
(6) Social security and health plan beneficiary numbers
(7) Medical record numbers
(8) Vehicle and device identifiers
(9) Full-face photographs
(10) Any other unique identification characteristics

The last of this list, "any other unique identification characteristics," adds a heightened level of difficulty in ensuring full compliance with the standards (Freymann, 2011).

As described previously, the most effective method of de-identification is the use of pseudonymous values. This, for example, could be accomplished by assigning each subject a unique numeric ID and by creating a code for demographic values such as race and gender. This is often a step taken by researchers before data sharing or the publication of datasets. Publications and databases contain fewer specific data objects than data in a self-contained research group (Hrynaszkiewicz, 2010).

Our team's product classifies as Software as a Medical Device according to FDA standards, with the intention of informing clinical management based on defined outputs. Referencing the intended uses and functionalities of SaMDs was paramount to our design decisions. For example, GILBERT stores subjects anonymously, using a subject number rather than a subject name. These subject numbers are either produced automatically or are taken from the uploaded file, depending on the file format. However, GILBERT assumes that the data fed into it is already anonymized. So, if the numbers are to be taken from the uploaded files, it is crucial for the user to replace subject names with ID numbers. The progression from patient data to mathematical operations/algorithms to final defined outputs as shown in *Figure 2* was also a process kept at the forefront of our decision making. Although the current functionalities of GILBERT limit the amount of defined outputs, the product serves as a foundation to be built upon in future iterations, as explained in Chapter 6. Furthermore, we understand the severity of the healthcare situation that GILBERT may be used in and its potential classification as a medical device. By referencing these specific SaMD categories, our team hopes for GILBERT to be able comply with quality and security regulations in the future.

2.2 Big Biomedical Data

The growth of data in the last few decades has fueled industries with new computing and analysis capabilities. Biomedical data is rich and versatile, allowing for strong analytic capabilities, which is crucial in a field where diagnosis, prognoses, and patient statuses depend on data analytics. In this section, we will discuss the analysis of biomedical data in addition to methods of storage and organization.

2.2.1 Biomedical Data Analysis

As computing power becomes stronger and more accessible, industries are taking advantage of data-driven approaches of research and development. This is especially true in the healthcare industry, where procedures and decisions can be a matter of life or death.

Management and analysis of big biomedical data are prevalent in all areas of the industry including clinical trials, health records, disease tracking, and diagnostic prediction. These processes rely on a high volume of physiological data from various input devices, or “Internet of Healthcare Things” (Baljak, 2018). These devices include, but are not limited to wearable health trackers, smartphones, heart rate monitors, and fitness equipment. This data is not only abundant but also diverse; collecting inputs from the body’s various processes and mechanisms can reveal novel insights and correlations.

Biomedical data can be broken into three broad categories of signals: analog, discrete, and digital. Analog signals can be represented with continuous time and amplitude, where every time instance has a corresponding data point. An example of this is an analog mercury thermometer. The height of mercury in the thermometer column is continuously recorded to display changes in subject temperature over time (Najarian, 2012). Discrete signals have a continuous amplitude axis but at discrete time values. Values are only present at certain time stamps but can take on any value. Often, discrete signals are more common because they require less space to be stored and are the more practical choice for biomedical variables that do not change continuously. Digital signals are defined by having both discrete time values and discrete amplitude values. The limitations in digital signals is due to the device used to collect them. Digital devices can only report a value to a certain number of significant figures with confidence (Najarian, 2012). These three categories of biomedical signals are all crucial and necessary inputs for analysis methods to be capable of handling.

There are commonly implemented methods for computational analyses of these signal types, outlined across many studies. For example, a 2013 publication by Schuller et al. reviews paralinguistic analysis as a rising method of speech processing. Following the collection of linguistic data, methods for computational analysis are implemented. The first of these methods is pre-processing, a technique frequently used for many biomedical applications. Pre-processing cleans the data before it is used for further analyses and most commonly includes removing artifacts or unnecessary segments of the signal. Interfering frequencies are a common artifact to many signals and can be attributed to a variety of things such as electrical activity from other nearby sources (Najarian, 2012). Schuller et al. describes a pre-processing technique that is used to enhance a speech signal, remove artifacts such as noise, or separate multiple speakers (Schuller, 2013). Often, pre-processing and the removal of artifacts is accomplished through filtering techniques. The creation of bandpass filters, which combine high and low pass filters allows the user to define an acceptable range of frequencies, attenuating all other frequency levels. This would allow for a level of selectivity over desired frequencies. More complex enhancement algorithms, as cited in Schuller et al., can also achieve desired filtering. The Independent Component Analysis algorithm, specifically described in the paper, isolates single microphone inputs from a summation of many signals (Schuller, 2013).

Another crucial step in biomedical signal processing is data segmentation, most often into a partitioning of the time series. Often, the segments are made through a sliding window, a technique used in the study on paralinguistics (Schuller, 2013). Feature vectors were extracted from acoustic data with a window size of 10-30 milliseconds. This allows for the extraction of time and frequency domains and frame-by-frame estimations, all of which allow for the extraction of Low-Level Descriptors. In this example, Low Level Descriptors may include pitch, sound intensity, amplitudes, and harmonicity. A similar method of segmentation is known as chunking. Chunking is used to define the temporal unit of analysis. This can include determining frames for analysis such as words, segments of voiced parts, or syllables. Extraction of key features from these frames such as extremes, means, percentiles, and variation provide the user with further information about the data (Schuller, 2013). In the example of the electrocardiogram, this can be essential to determining key features about the patient including the amplitude of the R-wave, and the time between subsequent R-peaks to determine heart rate. Further, one can narrow into other peaks of the waveform to possibly identify signs of problems such as tachycardia and arrhythmias.

Feature selection can also be accomplished through more complex methods such as statistical analyses, correlations, model learning, and other mathematical algorithms. Statistical analyses provide information on the correlation between variables and the predictive strength of parts of the biomedical signal. For example, from acoustic data in Schuller et al. (2013), they found it possible to determine the speaker’s weight through correlations in speech pattern. Previous data in the system may show that subjects within a specified pitch range are also part of a specified weight range, which can be confidently determined with statistical significance.

Model learning is an example of an algorithm that makes use of data classification and categorization. Classification aims to relate different test instances with different labels and determine correlations. In Schuller et al., researchers determined different correlations between speaker traits and states. Classifications and correlations, as seen outlined in the acoustics study, can then be used to train a model to be able to predict traits and states from inputted acoustic signal data. In this application specifically, acoustic models consist of learnt dependencies between different variables such as emotions and personality and the acoustic signals and language models store learnt dependencies between the similar categories and linguistics such as words and syllables (Schuller, 2013).

A summary of the processing and analysis techniques used in Schuller et al. (2013) is seen below in *Figure 3*.

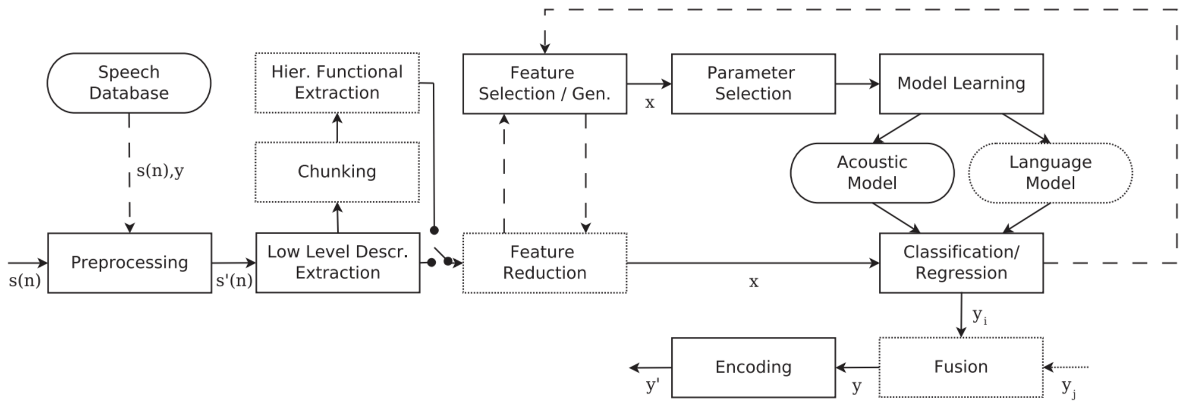


Figure 3. An overview of the analysis methods implemented in speech analysis systems (Schuller, 2013).

Further implementation of this analysis process can be seen in research like the eye-tracking study conducted at Bellevue Hospital Center in New York City (Samadani, 2015). In this study, researchers used eye tracking sensors in an effort to quantify the association between disconjugate eye movements with concussions and other traumatic brain injury (TBI). Though abnormal eye movement patterns have long been identified as a sign of TBI, current technology has allowed researchers and clinicians to quantitatively identify and measure such injuries. The sensor used in this study was an Eyelink 1000 eye tracker that followed the pupil position of the subjects - categorized as positive or negative for head trauma - as they watched videos on a monitor placed 55 centimeters away. From this tracker data, researchers created scatter plots of time series data and used statistical software to quantify aberrations in eye tracking. Using this analysis, researchers drew correlations between eye tracking disconjugacies and the extent of a subject having a positive scan for TBI.

Researchers and clinicians are also using data to make informed predictions when making diagnoses for infectious diseases. With data from computed tomography (CT) scans, the National Key R&D Program of China created a COVID-19 identifier using machine learning methods. To do this, researchers labeled each scan with its patient's COVID-19 test result, and then fed this into a program to find patterns in the scans that indicate the presence of COVID-19 (Wang, 2020). Although a model like this is not yet validated for clinical use, machine learning offers researchers a fast and convenient tool to identify high risk subjects and understand the effects of diseases on the entire body.

Because of the specificity of some of the analysis methods, we have not considered them as viable methods for GILBERT's analysis features. However, these crucial steps such as preprocessing, filtering, feature extraction, correlations and statistical analyses, and deeper algorithms like machine learning are important to understand for future design considerations and iterations of our product, as described in Chapter 6.

2.2.2 Standard Analysis Tools

Performing computational analyses on biomedical data relies on three main components: management, storage, and analysis methods. Successful data analysis is accomplished with fluid transition to and from data management and storage (Tchagna Kouanou et al., 2018). Many standard tools of practice have these capabilities that allow researchers to efficiently analyze large quantities of biomedical data. Various tools that derive meaningful analytical outcomes from data include Microsoft Excel, scripting languages such as Python, R, and MATLAB, and visualization software like Tableau. Even though these programs and languages are all utilized for the purpose of analysis, each has its own defining characteristics which make it applicable for certain types of analysis.

Though at first glance it may seem like a modest spreadsheet editor, Microsoft Excel is a powerful tool for the analysis and transformation of tabular data. The Excel interface makes smaller datasets easy to manipulate and edit, and users can create basic charts and formulas to delve further into data analysis. Though these processes are basic, they provide an important statistical foundation for researchers. As reinforced by Divisi et al., a lack of understanding in basic statistics can lead to major discrepancies in further data analysis. In many cases, analysis on Excel is an effective and straightforward first step in further understanding a dataset.

Scripting languages can support automated functions that create a more repetitive and repeatable environment for data analysis. MATLAB is a scripting language whose foundational data storage is in matrices. It supports a large range of computational capabilities including statistical analyses, machine learning, signal and image processing, graphical capabilities, and much more. MathWorks, the creator of MATLAB, provides many add-on products to users that increase functionality and include many additional built-in functions. Some of these add-ons include curve fitting, data acquisition, image processing, signal processing, statistics and machine learning, and symbolic math toolboxes. In conjunction with MATLAB's wide range of initial capabilities, these add-ons create a high-level environment that is widely used in problem solving with data driven solutions. Its applications have been used in industries beyond the field of biomedical engineering, including aerospace and defense, communications, geologics, and energy production (MathWorks, 2021). Another commonly used scripting language is the open-source software package, R. R is known for its user-friendly interface compared to other programs such as Python and C. It also hosts nearly 6,000 preset software packages that allow for easy use of general capabilities like statistical analysis and data visualizations, as well as more complex, specialized packages. Many notorious analytical products have been created with R at their core including Bioconductor, a genome sequence analyzer, IPMpack, a population growth modeler, and ggplot2, a graphical analyzer (Tippmann, 2014).

Interactive data dashboards like Tableau are emerging in popularity for their ability to display fluid, immersive visualizations. Data visualization is an effective means of providing easily consumable insight to the data analysis, utilizing visual rhetoric fundamentals to create an implicit understanding of the results. These dashboards offer more impactful analysis to the user than traditional visualization methods, as their interactivity allows for the selection of qualities and attributes that the user deems most effective (Janvrin, 2014).

As we built our application, one of our objectives was to make data handling easier in GILBERT than it would be through these tools. For example, we aimed to provide the statistics and charts that can be made using Excel and MATLAB without any confusing chart making or code writing. When completing analysis across studies, we hope that GILBERT will have the same or better impact as an interactive data dashboard. We also used these tools to help build our project. For example, GILBERT automatically computes statistics using Python and makes visualizations using Matplotlib, a Python library that mimics the MATLAB graphing capabilities. We discuss these technologies more in Section 2.3.1.

2.2.3 Big Data Storage

When discussing how to store large amounts of data of different types, there are two main options: the data warehouse and the data lake. While both consume data easily and have quick response times, the two have many differences in structure, flexibility, how data is put into it, and more (Sharma, 2018).

The data warehouse is the more structured of the two storage systems. It uses the schema-on-write technique, meaning that the database schema had to be created before data was entered, and all data had to fit this template at the time of storage (Inmon, 2005; “Schema on Write”, 2020). They are typically designed and populated in a step-by-step process, rather than all at once (Inmon, 2005). The data in the warehouse is also integrated, meaning that it is reformatted to communicate easily across tables (Inmon, 2005). For example, there are multiple ways to format a gender variable in a database, such as ‘m’ and ‘f’, 0 and 1, and ‘male’ and ‘female’. The data warehouse will take this data and convert all of these columns to have the same values. Given the previous example, the data warehouse would have one specific gender variable name, such as ‘m’ and ‘f’, and convert all gender variable columns with different labels to ‘m’ and ‘f’ (Inmon, 2005). Data warehouses are also known to store data at different levels of granularity (Inmon, 2005). For example, it can hold each record from each patient using a Fitbit, along with a table of summary statistics of all users using all Fitbits (Inmon, 2005). This allows easy querying both for specific questions, as in asking how many steps an individual made on a certain day, and broad questions, as in which time of day exhibits the highest heart rate among all Fitbit users (Inmon, 2005). Overall, while data warehouses can take more time and effort to plan and build in the beginning, it allows for easy access to multiple sets of data (Sharma, 2018).

Data lakes, however, are more flexible. They are fed data in its raw form, and little or no processing is involved in storing and structuring it (Sharma, 2018). The schema is not known until the data is fed; this is called the schema-on-read approach (Sharma, 2018). This means that the data lake can store data in virtually any format, from spreadsheets to blog posts to sound files (Sharma, 2018). They also do not need to perform as much integration since there is no rigid schema (Sharma, 2018). However, this also means that building a data lake is far more complex (Sharma, 2018). Without a definitive structure, many different technologies are needed to be able to put data into the data lake and retrieve it for analysis (Sharma, 2018). It is up to the creators to determine how the data can be integrated upon retrieval if necessary and to ensure that the transferring up data remains private and secure (Sharma, 2018).

In the span of our project, we did not create either a data warehouse or a data lake. However, assuming the scope of our pipeline grows after we complete our portion of the project, connecting a data warehouse or data lake would both have benefits. A data warehouse could be a simpler approach if the user knows how to integrate any entered data properly. On the other hand, a data lake could allow for speed and flexibility, and these are used by other biomedical researchers (Poudel).

2.2.4 Databases and Database Management Systems

A database is an organized collection of structured information, or data, typically stored in a computer system (Bella & Nagle, 2014). They are usually controlled by a Database Management System, or DBMS. A DBMS is a software tool that acts as the interface between the end user and the database (Esakkirajan & Sumathi, 2007). While the functions of a DBMS can vary greatly, some of the general functionalities are data management, a database schema that defines the database’s logical structure, and a database engine that allows data to be accessed, locked, and modified.

In a DBMS, data models are used to describe the underlying structure of a database. The model defines how data is connected and how it is processed and stored inside the system (Navathe, 1992). In the early stages of DBMS, the popular data models were the hierarchical and network models. In the hierarchy model, each node/component has a child-parent relationship with one other node/component. In the network model, a node/component can have multiple relationships (Navathe, 1992). While these were the primarily used models used by DBMS, they had the drawbacks of minimal data independence, minimal theoretical foundation, and complex data access. These drawbacks were overcome with the introduction of the relational database model, which introduced the concept of storing data rows and columns in tables rather than as a hierarchy in a file (Esakkirajan & Sumathi, 2007). This new data model led to the creation of a more advanced DBMS system, known as the relational database management system (RDBMS).

2.2.4.1 Relational Database Management System

The relational database model was a landmark development because it provided a mathematical basis to the discipline of data-modeling based on the notions of sets and relations (Navathe, 1992). In an RDBMS, the data in the database is organized in the form of tables (relations), and each table consists of rows (tuples) of data that are defined over a set of attributes. With this unique organization, RDBMS are able to provide benefits such as:

- Relational algebra
- Simple and precise semantics that is able to provide a good amount of query optimization within the system
- A clean separation of the logical from the physical parameters, which makes it easier for a wider group of users to understand and work with the database
- An easily maintainable system

With these benefits, this management system gained wide popularity in the business and technology industry. Some of the popularly used RDBMS are MySQL, PostgreSQL, Oracle DB, and SQL Server. Almost all RDBMS use SQL (Structured Query Language) as the standard programming language used to access and manage the relational databases, but most of them have their own extensions of the language. We considered the different properties of RDBMS when choosing to use PostgreSQL for this project.

2.2.4.2 Non-Relational Databases

While relational databases are popular in the business and technology industry, non-relational databases are on the rise. Non-relational (NoSQL) databases were developed in the late 2000s with a focus on scaling, fast queries, frequent application changes, and making programming simpler for developers (Schaefer, 2020). While relational databases tend to have rigid, complex, and tabular schemas, NoSQL offers more flexibility. It encompasses a wide range of database technologies that can store structured, semi-structured, unstructured, and polymorphic data (Navathe, 1992). In other words, it does not require a predefined schema. While the flexibility that NoSQL databases offer is an advantage, it unfortunately does not handle data reliability and consistency well. Unlike relational databases, it does not natively support ACID (Atomicity, Consistency, Isolation, Durability), a standard set of properties that guarantees that database transactions are processed reliably (Navathe, 1992). By lacking in this, NoSQL databases tend to have a higher chance of compromising data consistency. With these drawbacks, NoSQL databases cannot replace relational databases, but they can be more effective to use for certain types of projects. We considered using NoSQL databases when implementing this project.

2.2.5 Data Pipelines

In an age where vast amounts of data are being stored and analyzed, the need for a fast, reliable data management system is growing. Oftentimes, companies that collect large amounts of data from many different sources need to store that data in several different places. This can make data difficult to access for later analysis. Data pipelines offer a valuable solution to this problem. At its core, a data pipeline is responsible for moving data from a source to a destination (Dearmer, 2020). This could include moving data from where it is being generated or collected and moving it to a data warehouse or an analysis tool.

Data pipelines get their name due their flow of data, as the data must be processed in one step before it can move to the next. Along the way from source to destination, the data in the pipeline is manipulated. It is a data pipeline's job to combine new data with old data, perform feature extraction, filter data, and error check along the way (Dremio, 2020). This automates much of the manual data cleaning and processing that would be required without a pipeline.

Data pipelines offer a wide variety of uses in many important fields. One example of this is an automated analysis pipeline that transforms microscopic images of cells into tabular data (Evans & González, 2019). This data pipeline, seen in *Figure 4*, uses several stages of neural networking to process a raw image of many cells, identify each of the cells, specify areas of interest on each cell, and extract data about those areas into a table. This table can later be used for visual analysis in languages like R or Python. The pipeline automates a large amount of data collection; about 13 gigabytes for each experiment. González says, "This push towards full automation removes the burden of step-by-step manual data analysis, freeing the scientist to focus on experiment design, data acquisition, and interpretation."

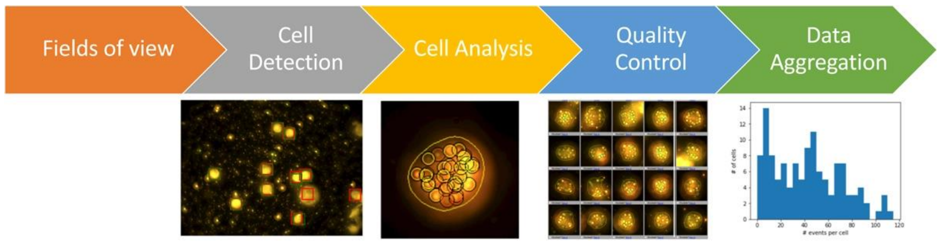


Figure 4. Overview of automated analysis pipeline described above.

Data pipelines also have uses in the healthcare field. One such pipeline, proposed by a group of researchers, would be responsible for collecting medical data from sensors on patients, running analysis on the data to determine indicators of health, and finally predicting the disease or injury in the patient (Baljak et al. 2018). The pipeline would handle multiple types of data for each patient, such as outputs from electrograms to monitor heart function and vital signs such as blood pressure, body temperature, and heart rate. In the current healthcare system, patient data is collected and monitored but rarely recorded. Implementing a pipeline that stores and analyzes that data could change the way healthcare professionals care for their patients. Not only could the pipeline run real-time analysis on the patient that it is collecting data from, but it would also store that data to help with diagnosis predictions for future patients.

One example of a pipeline that is progressing towards accomplishing both data storage and analysis is the Cerebral Blood Flow Biomedical Informatics Research Network. The Cerebral Blood Flow Biomedical Informatics Research Network, more simply known as CBFBIRN, is a combination pipeline that includes a database and analysis system. Developed by a team from the University of California at San Diego, this pipeline works with a magnetic resonance imaging (MRI) technique known as arterial spin labeling, or ASL. ASL is an arising method for quantifying a subject’s cerebral blood flow (CBF) and is known for being non-invasive. The analysis of CBF is crucial for studying and better understanding a multitude of conditions and disorders including Alzheimer’s disease, schizophrenia, bipolar disorder, depression, traumatic brain injuries, human immunodeficiency virus, and methamphetamine abuse (Shin, 2013).

Because of its wide range of applications and relatively new arrival in the medical field, there is a need for data sharing and storage. CBFBIRN provides a central data repository for researchers to upload arterial spin labeling images and other clinical data so that the data may be processed and analyzed. Benefits of data sharing in this application include greater effectiveness of statistical analyses, better testing of new algorithms, and a better understanding of the systems accuracy and reliability (Shin, 2013). The CBFBIRN database is an entity-attribute-value (EAV) database that improves usability and efficiency by storing data and metadata in the same centralized schema. This allows for easy accessibility to different studies and their characteristics, demographic data, clinical assessments, and raw data. There are small extensions to the primary schema for uses like security and raw data uploading. A user can upload raw data from their machine through two paths: (1) a Java Web Start application and (2) a user interface that connects a web browser to CBFBIRN. When attempting to upload raw data, the user is guided through steps of browsing, selecting, and uploading. The system performs a check to analyze the quality of the uploaded image, classify the uploaded image, and anonymize the information relating to the image. Anonymization is vital to patient security and the removal of possible identifying features. Security is also ensured when it comes to accessing and uploading data through a privilege-based system that is facilitated by the administrator (Shin, 2013).

Following the upload of a raw ASL image, the user can follow one of two processing modules. Processing module one consists of cerebral blood flow quantification. CBF quantification allows the user to select different processing methods from a subset after being presented the different options. These include field map correction, partial volume correction, spatial smoothing, and skull stripping (Shin, 2013). Processing module two focuses on filtering and group analysis. The CBFBIRN filters data with a query that allows the user to select based on group type, demographics, clinical assessment, along with previously performed processing techniques. There are three specific types of group analysis: (1) whole-brain mean grey matter analysis, (2) regional analysis for each of the user defined regions of interest, and (3) voxel wise standard space analysis. The sequential steps of filtration and group analysis allows for the comparison of different variables across different groups. Statistical methods are invoked to generate results and quantify group comparisons. These results include summary tables, tables displaying statistical results, derived data, and importantly, data visualizations. *Figure 5* details the functional flow of the CBFBIRN system. It highlights key features that our team added to GILBERT such as the database, the ability to build a query, options for analysis, and displayed results in the form of summary tables or visuals.

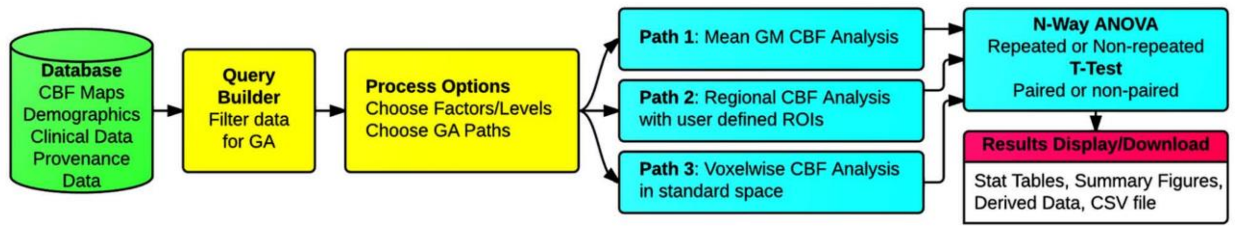


Figure 5. CBFBIRN flows from the central data repository to displayed results (Shin, 2013).

While this pipeline allows for users to upload and share data as well as select different analytical methods, it has some limitations. CBFBIRN only works with one type of data collected through magnetic resonance imaging. The analysis methods are very specific to the type of data this pipeline works with and has limited selection options for the user. These analysis methods also do not include diagnostics, predictions, or patient status. There is a need for another analysis step outside of the pipeline. In short, this pipeline is too specific, and our team aimed to build one that is less so.

By analyzing these other data pipelines, our team had a better understanding of the advantages and shortcomings that data pipelines have. For example, many of these pipelines had a very specific use. They were only able to process one type of data and produce results that are specific to certain conditions. For this reason, our pipeline was built to have a larger scope. While GILBERT currently does not have as many analysis capabilities, we hope that its ability to analyze existing data will be expanded upon and applicable to more than one kind of biomedical data.

2.3 Technologies

We considered and chose to use various programming languages, frameworks, and libraries to accomplish the goals of the project.

2.3.1 Python

Python is a high-level object-oriented scripting language. It is also considered an interpreted language, meaning that the code is run by an interpreter rather than being compiled into machine language instructions, allowing it to be platform independent. Some key features of Python include its relatively simple syntax and dynamic typing and binding (Python Software Foundation, n.d.). The current version of Python is Python 3, which has significantly redesigned syntax, text formatting, and other various changes from Python 2, which stopped receiving support in January 2020 (Coghlan, 2020). Python has a wide variety of libraries for different purposes and interfacing with other software. Because of this, the team used Python for the back end of this project, alongside various open-source Python libraries and frameworks such as Pandas, SQLAlchemy, Matplotlib, and Django.

Pandas is a widely used open-source Python library that is built on the NumPy Python library and is used for data analysis and manipulation. It provides easy functionality for reading and writing to SQL databases and different file formats such as .csv, .txt, and .xlsx. Furthermore, it has multiple operations for reshaping datasets and handling missing data. This is accomplished through the use of DataFrame objects, which Pandas uses for storing and modifying datasets (NumFOCUS, 2020). GILBERT uses Pandas to manage and analyze data on its back end.

SQLAlchemy is a Python library for working with relational databases and ORM, or object-relational mapping. It consists of two main components: its core package, which features abstraction functions for generating SQL statements, and its optional ORM package, which allows for control over how data is translated between Python objects and SQL data types. SQLAlchemy supports most forms of SQL, including SQLite, PostgreSQL, MySQL, Oracle, MS-SQL, Firebird, and Sybase (Bayer, n.d.). GILBERT uses SQLAlchemy to convert data between its database and back end.

Matplotlib is a Python library for creating different types of data visualizations. It was originally developed by John D. Hunter around 2003 to emulate the graphing capabilities of MATLAB within Python (Solomon, 2021). It is a widely used open-source library with an abundance of documentation and support. Multiple add-on toolkits and third-party packages are available to provide additional features. The library offers a multitude of different configuration options for visualizations, as well as the ability to export and embed them (Hunter, 2021). GILBERT uses Matplotlib to generate graphs within its analysis feature.

Django is an open-source framework designed for creating web applications using Python. It was initially developed between 2003 and 2005 before being released as an open-source project. It is scalable and secure, with built in features for adding user authentication, site maps, and RSS feeds. For hosting purposes, it is compatible with several different web platforms and services. Django follows the Model-View-Template design architecture, where a view contains handlers for accessing model data, while delegating front-end formatting to a HTML template (MDN Contributors, 2019). The web application portion of GILBERT uses Django.

Similar to Django, Flask is another open-source framework used for making web applications in Python. It depends on the Jinja template engineer and the Werkzeug WSGI toolkit and was originally released in 2010. Flask allows for support of extensions and does not specify which database can be used, allowing for flexibility, and broadening its applications. These extensions can include upload handling, database integration, authentication, and more ("Flask Documentation", 2020). We considered Flask when designing the web application portion of GILBERT.

2.3.2 MySQL and PostgreSQL

MySQL is a popular open-source relational database used widely in web applications. The tool was created by MySQL AB, a Swedish software company which has been acquired by Oracle (MySQL, 2020). There are multiple libraries available for most common web and object oriented languages to connect with MySQL databases, such as the mysql.connector library for Python. The team used MySQL for this project in its initial stages.

PostgreSQL is another popular open-source relational database which has millions of users. The tool has over 30 years of active development, originating from research done at University of California at Berkeley in 1986. It extends the standards of SQL with added features for improved transaction safety and scaling. It also runs on all major operating systems and has numerous add-ons and APIs for interfacing with different languages and frameworks (The PostgreSQL Global Development Group, 2021). The team used PostgreSQL for this project in its later stages.

2.3.3 Heroku

Heroku is a cloud platform for hosting and maintaining web applications. It is treated as a platform as a service, which internally uses Amazon Web Services, which is an infrastructure as a service. Developers can deploy applications and services on dynos, which are lightweight virtualized Unix containers. Heroku has various add-ons for supporting different data sources and supports applications in languages such as Node, Ruby, Java, Scala, and PHP. Applications do not need to be modified extensively to work with Heroku (Heroku, 2020). We hosted GILBERT on Heroku.

2.3.4 Amazon S3

Amazon Simple Storage Service (Amazon S3) is an object storage service that proves a platform for storage of data in the form of objects (“Amazon S3”, n.d). The storing of objects allows users to store, analyze, and retrieve large data from a variety of platforms such as websites, mobile apps, etc. The team used this service for temporarily storing the uploaded .csv files and also retrieving them for the upload process.

2.3.5 Celery

Celery is a Python-based task queuing software package that executes asynchronous computational workloads (McQuistan, n.d). The technology is commonly used in conjunction with a storage solution, also known as a message broker, which allows for communication to happen between the application code and Celery. The message broker stores messages produced by the application code which describes the work to be done in the Celery task queue. Celery transfers this to a Celery worker which spawns threads and executes the actual task. Once the task is completed, the results are stored in the message broker for the application code to retrieve. GILBERT uses Celery to handle tasks related to its uploader feature.

2.3.6 Redis

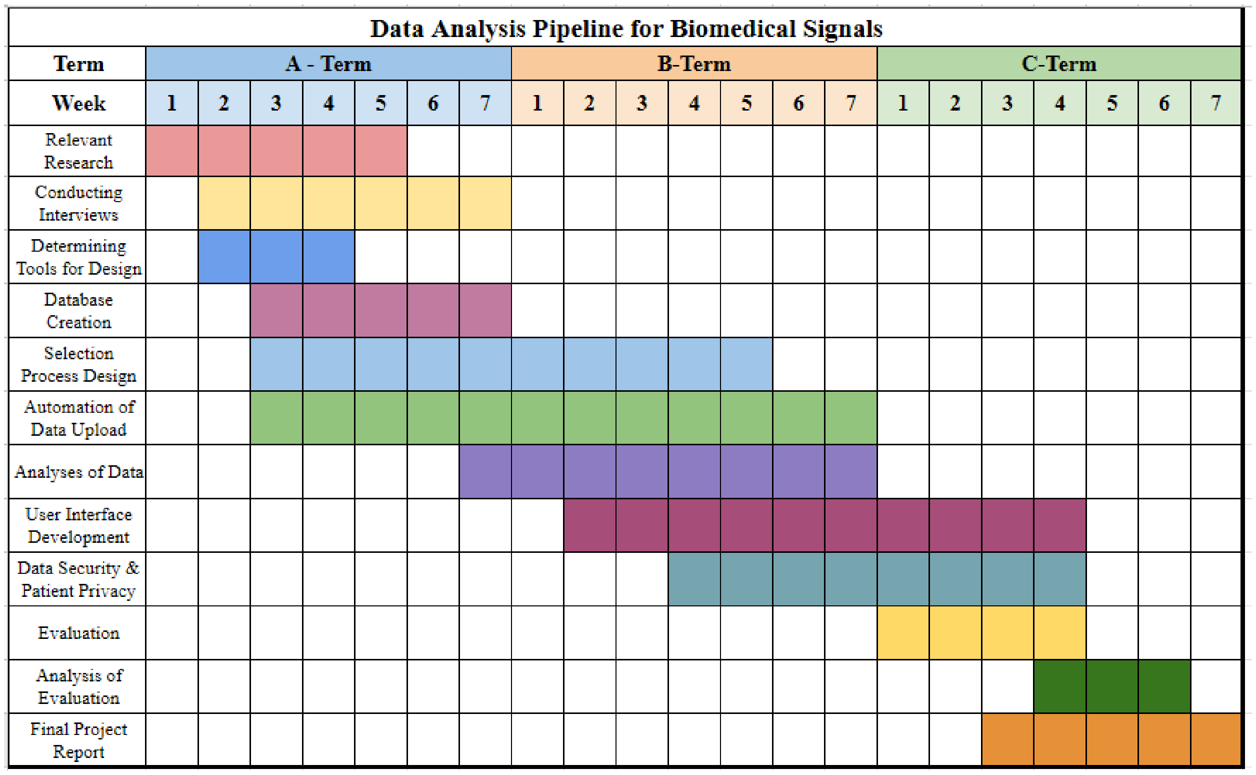
Redis, which stands for Remote Dictionary Server, is a fast, open-source, in-memory data structure store that is used as a database, cache, message broker, and queue (“Redis”, n.d). It is a key-value data store, which is a data storage paradigm that is designed to use a simple key-value method to store data. The database stores data as a collection of key-value pairs in which the key acts as a unique identifier. Because Redis stores the data in-memory rather than on disk, it eliminates the need to access disks and avoids seek time delays. GILBERT uses Redis as a message broker to store tasks related to its uploader feature.

2.4 Summary

A data pipeline provides many ways to automate the activities of researchers, healthcare professionals, and engineers. When large quantities of data are collected, a system needs to be in place to run automatic analysis. By leveraging a system for this purpose, people in these professions can focus their efforts on other aspects of their job. For this reason, we created a data pipeline whose primary functionality is to manage, store, and analyze various types of biomedical data. Research of industry standards and investigations into available tools supported design decisions and the selection of tools such as Django, Heroku, and PostgreSQL.

Chapter 3. Methodology

This chapter discusses the steps taken to develop GILBERT. We began with research of relevant literature and interviewing stakeholders. The information collected from this research influenced our design decisions. The Agile methodology was implemented to promote teamwork, retrospective evaluations, and process improvement. Through our sprints, GILBERT was improved by implementing new functionalities, beginning with the creation of the database, and progressing through the creation of an easy-to-use user interface (UI). The timeline of our methodology and design process is shown in the Gantt chart below (*Figure 6*).



Data Analysis Pipeline for Biomedical Signals

Term	A - Term							B-Term							C-Term						
Week	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7
Relevant Research	■	■	■	■	■																
Conducting Interviews		■	■	■	■	■	■														
Determining Tools for Design		■	■	■																	
Database Creation			■	■	■	■	■														
Selection Process Design			■	■	■	■	■	■	■	■	■	■									
Automation of Data Upload			■	■	■	■	■	■	■	■	■	■	■	■							
Analyses of Data						■	■	■	■	■	■	■	■	■							
User Interface Development									■	■	■	■	■	■	■	■	■	■			
Data Security & Patient Privacy											■	■	■	■	■	■	■	■			
Evaluation															■	■	■	■			
Analysis of Evaluation																		■	■	■	
Final Project Report																	■	■	■	■	■

Figure 6. Gantt chart outlining GILBERT development timeline.

3.1 Requirements Gathering

In the initial phase of our project, we researched tools, industry standards, practiced methodologies, and existing pipelines to guide our decision making and aid us in determining the scope of our project. We accomplished this through reviewing relevant literature and conducting multiple interviews.

We began with a semi-structured interview with Boston Scientific data analytics analyst Umanga Poudel. In this interview, we aimed to understand current practices and tools that are used in the medical device industry. When asked to briefly explain his role at Boston Scientific, Mr. Poudel explained that he works with feedback data on company devices stored in a tabular format. He detailed the method for data upload as follows:

1. Users fill out a form on the device
2. Form is sent to a back-end Oracle database
3. Data from the form goes into a data lake

From there, the team writes queries to analyze the data. Mr. Poudel outlined many different tools for data analysis including Statistical Analysis Software (SAS), Tableau, Risk Based Inspection (RBI), Python, and Microsoft Excel. The performed analyses allow their team to create real-time automatic reports and data visualizations. We then asked Mr. Poudel to explain the rationale for selecting these tools along with advice for our selection process. He stressed the importance of speed in diagnostic capabilities which can be improved by having compatibility between the data storage and analysis tools. Further, he explained a system must be easy to validate into use. For this reason, his team heavily relies on SAS as it is already validated, helping his team avoid the troubles of validating the constantly updating libraries of Python. Finally, we asked Mr. Poudel to detail the troubles that his team faces to help us identify possible areas our project could improve upon. He explained that data that he and his team collect contain a lot of free text fields, which can lead to incorrect or poorly entered data. He also discussed that having many different methods for analysis decreases the efficiency of the analysis process and makes it difficult to centralize the data storage and analysis methods. The list of questions that we asked Poudel can be found in Appendix A (Poudel).

Next, we interviewed two professors at Worcester Polytechnic Institute: Adam Lammert, Ph.D., a biomedical engineering professor and our project advisor, and Emmanuel Agu, Ph.D., a professor of computer science. To better understand the scope and need for our project, we asked Professor Lammert to explain the inspiration for the biomedical data pipeline project. Professor Lammert explained that in his time as an employee at Massachusetts Institute of Technology's Lincoln Labs, he saw a need for combining data storage and a centralized method of flow between storage and analysis. He cited frequent use of ad hoc methods, in which the data was collected for a single use and then not referenced again, and the possible benefit of being able to incorporate new data as collected and perform cross-study analyses. Professor Lammert also described some important functionalities and design considerations for our team's pipeline in response to further questions. This included ensuring patient data privacy, creating an easy-to-use interface for busy clinicians, and having quantifiable diagnostic abilities with specified sensitivity and specificity. The full list of questions we asked Professor Lammert can be found in

Appendix B (Lammert). During our team's interview with Professor Agu, he first explained that he is currently working on a drunk driving project that uses machine learning and sensor data from phones to make sure someone is safe to drive home. Currently, data for his projects is stored locally on mobile devices or sent to a server, with analysis often being done with Python scripts. Based on the description of our pipeline, Professor Agu believed it would make working with data simpler and provided our team with advice on pipeline construction based on shortcomings of systems he has worked with. He instructed us to code using industry standards and to be careful with data transformations as they can have bugs that are difficult to detect. The full list of questions can be found in Appendix C (Agu).

At the start of our project, our team was provided two data sets. Both were collected from previous WPI Interdisciplinary Qualifying Projects (IQPs). One project aimed to understand changes in cognition in relation to different exercise regimens. Their dataset included accelerometer and heart rate data collected from Fitbits and two cognitive assessment tests, the Corsi block-tapping test and the Eriksen Flanker Task (Finn). The other project's goal was to identify differences in speech and gait data between those who had experienced a traumatic brain injury and those who had not. In addition to interviewing the two teams to better under the organization of their datasets, both teams were asked to explain the attributes of a system that would have assisted them in organizing, processing, and analyzing their data. Both teams mentioned they had a significant amount of manual work and that a system with automation would have been ideal. Further, they explained that for the different types of data in each of their datasets, they had different, discrete steps to extract and analyze the data. A centralized, universal method of pulling and analyzing data would have been beneficial to both projects. The full list of questions asked to both IQP teams can be found in Appendix D and Appendix E (Finn; Pandey).

As we designed the schema for our database, we met with one of our project advisors, Dr. Kajal Claypool, to get feedback and suggestions regarding the best method for storing the data necessary for our pipeline. This helped us improve our data storage model over time and address issues we had not previously considered. In our first meeting, we presented our preliminary schema to her and received various points of feedback. Most of these included renaming tables, adding attributes, and addressing privacy concerns regarding keeping subjects anonymized while still storing demographic data. Additionally, she recommended that we investigate non-relational databases as a potentially more effective method of data storage. After improving our model and conducting research on non-relational databases, we met with Dr. Claypool a second time to clarify the need for a non-relational database and explore how it could be configured to support the pipeline. During the meeting, we discussed potential approaches and limitations for using a relational database for our project, and concluded that it would be just as effective, if not more so, than using a non-relational database alone, which matched the conclusions we reached from our research. The meeting agenda used to lead the discussion can be found in Appendix F (Claypool).

Following the interviews, we investigated literature to better understand current practices including relevant pipeline examples, to understand industry standards, and to find possible tools for implementation. We researched Food and Drug Administration (FDA), National Institutes of Health (NIH), and Health Insurance Portability and Accountability Act (HIPAA) standards of data use in biomedical applications. This gave us a better understanding of patient privacy, data sharing, and design requirements for medical devices. Next, to better understand data type and storage methods, we investigated management methods for big biomedical data including both Relational and Non-Relational Database Management Systems. Our team developed a toolbox to aid with the pipeline design. This included narrowing in on a scripting language, Python, and complementary packages, databases such as MySQL and PostgreSQL, and possible platforms to develop a user interface. To finally understand the combination of all these components, we analyzed current data pipelines for biomedical applications and identified their successes and shortcomings.

3.2 Applied Agile Methodology

When working in a large group on a project with changing objectives, the need for excellent group communication and collaboration is evident. To maintain a standard of communication, our team used the Agile Development Methodology for meeting schedules and code development. As defined by the software company Atlassian, the Agile approach is built around small iterations that allow for flexibility and variability as the nature of a project evolves (Atlassian, n.d.). These iterations - known as sprints - focus on completing small tasks within a short period of time. This approach allows for immediate deliverable feedback to expand on for the next sprint. When beginning the project, our team met to define the terms of our agile sprints, and how we would maintain and uphold them.

3.2.1 Length of Sprint

In a project that relies on regular input from advisors and stakeholders, it was imperative that the team regularly provided deliverables for feedback and direction. To accomplish this, we decided to set each sprint length to two weeks. This was agreed upon by the team as an ideal time frame to frequently deliver parts of the project while still making significant developments.

3.2.2 User Stories as Deliverables

To create tangible and understandable deliverables, we created user stories to outline the features we would implement in our pipeline. A user story is a high-level description of a scenario that a user of GILBERT would encounter, including any features or functionality that would be used. For example, a user story on a data privacy feature could be written as "As a researcher, I want my data to be stored privately so that others will not have access to the data." These user stories were added to our backlog, or list of user stories that have not been started. Then, before each sprint, stories from the backlog were selected to be completed during the sprint. The size of our team allowed multiple user stories to be completed in parallel by assigning different stories to individuals, pairs, or groups within the team, maximizing our productivity.

A comprehensive list of our user stories can be found in Appendix G. This list shows all of the user stories that were completed in each sprint, along with those remaining on the backlog.

3.2.3 Meeting Structure

When working on various tasks in separate groups, important information is at risk of going unshared or unrecorded. To ensure excellent communication, the team devised a series of meetings that would happen at different parts of the sprint. These meetings included scrum meetings (also known as standup meetings), demonstration meetings, retrospective meetings, and sprint planning meetings.

We held brief scrum meetings with our advisors on every Monday or Tuesday during the sprint weeks. These meetings lasted between 15 and 30 minutes and had the purpose of getting each team member acclimated with the current state of the project. Each team member would outline what progress they made since the last meeting and what future work they hope to accomplish. If applicable, the team member also outlined any obstacles they foresee in their future work so advisors or team members can assist them.

We also held demonstration meetings with our advisors every Thursday or Friday during the sprint weeks. At the midway point of a two-week sprint, this meeting was a smaller, informal demonstration of progress with the purpose of providing deliverables for feedback and direction. At the end of the two-week sprint, we provided a comprehensive demonstration of the working product to showcase new features and functionalities. These meetings usually lasted an hour, which allowed for dialogue between the team and advisors.

In between sprints, we held retrospective meetings. Only team members attended retrospective meetings. These meetings lasted 30 minutes and focused on the team dynamic rather than project goals. We discussed what went well during the previous sprint, what could have been improved on in the previous sprint, and what will be done differently in the next sprint. The takeaways from this meeting were applied to each following sprint to ensure constant improvement.

Before the start of each sprint, the team held sprint planning meetings. These lasted 30 minutes to an hour with the goal of planning the main objectives for the upcoming sprint. This included selecting user stories from the backlog to focus on for the next sprint and assigning team members to work on them. If a user story was too vague or large, the team either split it into two user stories or created subtasks within the user story, which were then assigned to different team members. In the event that a subtask became too large, the team created a new user story dedicated to the separate subtask. The team also added user stories to the backlog to address new requirements, feedback, or ideas that had come up during the previous sprint. These meetings helped bring clarity to our future plans and allowed us to immediately start working as soon as the next sprint started.

3.3 Data Storage Design

Because GILBERT needs to store and analyze any kind of biomedical data, we spent a considerable amount of time planning what type of database and schema to use. Our first decision was choosing between using a relational or non-relational database. We made this decision by evaluating the benefits and limitations of the two options and estimating how effectively we would be able to use them to support GILBERT.

Relational databases support structured, tabular data with consistent schemas. They have simple syntax and are designed to support linking data in different tables through the use of joins and foreign keys. This suits most of the needs for the pipeline, but without preset data schemas, it is not obvious how different data should be stored since different kinds of data can have different attributes and formats. With a relational database, this problem can be solved by creating new tables for each dataset. However, this would require considerable back-end logic for comparing data across different studies.

Non-relational databases offer flexibility and varying levels of structure for storing data without needing a predefined schema. These often take the form of key-value pairs in JSON, where you can freely add object fields and nested objects. A non-relational database would handle the unpredictable format of incoming data better than a relational database, but it otherwise did not seem appropriate for this project. The data model required to effectively support essential queries and analysis would be more complicated than what would be needed for a relational database and would likely require a combination of nested and linked objects. Using a non-relational database would also not make it significantly easier to compare data across multiple studies.

Based on this evaluation and our prior experience with these technologies, we chose to use a relational database because of the previously mentioned strengths. Initially, we selected MySQL since it is widely used and well supported for Python applications and used it to run databases locally. When setting up our application to use Heroku, we found that PostgreSQL had significantly more support on Heroku so chose to use it instead. Similarly to MySQL, PostgreSQL is widely used and well supported for Python applications. So, it still supported the goals of the project effectively.

3.4 Feature Selection

With a database client selected, we identified the desired features and characteristics of our future pipeline application. The possibilities for biomedical data management and analysis functions are plenty, but in order to create a meaningful working product in our allotted time, we agreed to focus on only the fundamental pipeline features. After much deliberation, we decided to design our application with the following functionality in mind:

1. The user should be able to easily upload and store data into a central, organized database.
2. The user should be able to easily search and browse all stored data and metadata in a consumable, at-a-glance format.
3. The user should be able to easily perform simple filtered queries and analysis with an option to export data for higher level manipulation.

In addition to these features, the team also placed high importance on certain design decisions of the application. Mainly, having a web-hosted application that is easily accessible and structured for research and institutional use and a simple interface that will not get in the way of the user's workflow. When combining complex features like we sought out to do in our application, it was crucial for our interface to be as straightforward as possible with plenty of embedded documentation and assistance.

3.5 Development

GILBERT is an application that users can interact with in many ways, making it imperative to include some form of a user interface. It has the crucial role of collecting the necessary information for our GILBERT's back end to use, process, and clearly present the information back to the user. If the user interface is not well made, it may be difficult for the user to understand the information the pipeline is displaying or to provide the information that our pipeline needs. So, our team focused on structuring the front end in a manner that would make it easy for a user to interact with.

3.5.1 Version Control System

In order to create an environment to develop the pipeline application as a team, our team used Git, a commonly used version control system. A version control system is a system that records changes to a set of files over time, which allows developers to be able to recall specific versions later on (Git, n.d). Git made collaboration easier, allowing changes by multiple people to all be merged into one source. It also offers the ability to branch out from the main code base, which our team used to develop new features in parallel. The concept of developing new features on a separate branch allows the team to encapsulate their edits, making it harder for unstable code to get merged into the main code base. Once the edits were completed and stable, they were then merged into the main branch.

3.5.2 Command Line Prototype

At first, our team wanted to make sure that GILBERT could filter the correct results out of the database. Due to our lack of knowledge of front-end development with Python, we decided to make an application that ran out of the command line. This allowed us to understand how we can use our database to display the desired results to the user and how we can successfully add data into our database. By doing so, our team could focus on storing and retrieving data from the database before building a more user-friendly front end.

When performing a selection on the desired data, our command line application asked for these things, in order:

1. Which study(s) to analyze
2. Which data in the study(s) will be analyzed
3. Which groups (control, experimental) are wanted
4. Which columns in the table are wanted
5. Any filtering conditions (for example, a heart rate greater than 80 bpm, or a subject number of 15)

Figure 7 shows an example of our command line prototype and how a user would go through the steps for data selection listed above. In this query, the user is asking for all heart rate data over 150bpm.

```
Welcome to the biomedical data pipeline!
Type in the study ID for the study you wish to analyze.
        1       Exercise IQP
Study ID:
1

What data from this study would you like to view? If you list multiple items, please leave only a space
between each one, or type 'all' if you would like to look at all the data.
        HeartRate_1
        Corsi_1
        Flanker_1
Data Tables: HeartRate_1

Which groups would you like to analyze? Select one, or type "all" if you would like to look at all data.
        Control
        Experimental
Enter the group you wish to analyze, or type "all" if you wish to use all groups.
all

What attributes of this data would you like to analyze? If you list multiple items, please leave only a
space between each one, or type 'all' if you would like to look at all the data.
SELECT * FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = N'HeartRate_1'
1       HeartRate_1.date_time
2       HeartRate_1.heart_rate
3       HeartRate_1.doc_id
4       StudyGroup.study_group_name
5       Subject.subject_number
Attributes: 1 2
SELECT DATA_TYPE, COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = 'HeartRate_1'
Any conditions you would like to filter with? Type y or n
y

What attributes of this data would you like to filter by? If you list multiple items, please leave only
a space between each one.
SELECT * FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = N'HeartRate_1'
1       HeartRate_1.date_time
2       HeartRate_1.heart_rate
3       HeartRate_1.doc_id
4       StudyGroup.study_group_name
5       Subject.subject_number
Attributes: 2
How would you like to filter the attribute HeartRate_1.heart_rate? Please separate each condition by a
comma.
If you input a date, please use this format: YYYY-MM-DD HH:MM:SS. Use 24hr time.
Example input: subject_id = 1,  subject_id = 2
SELECT DATA_TYPE FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = 'HeartRate_1' AND COLUMN_NAME =
'heart_rate'
HeartRate_1.heart_rate = x, HeartRate_1.heart_rate != x, HeartRate_1.heart_rate >= x,
HeartRate_1.heart_rate > x, HeartRate_1.heart_rate <= x, HeartRate_1.heart_rate < x
Conditions: HeartRate_1.heart_rate > 150
HeartRate_1.heart_rate > 150
Here is your data!

HeartRate_1.date_time           HeartRate_1.heart_rate          Subject.subject_number
2019-12-11 13:10:00             154                             C1
2019-12-11 13:10:05             157                             C1
2019-12-11 13:10:10             156                             C1
```

Figure 7. An example of a query on our command line application. Highlighted text shows user input.

With this information, GILBERT queried its database and printed the data. If multiple data categories are selected, GILBERT performed a join on all the selected tables on the subject ID. This resulted in duplicate data being printed out; however, this proved that GILBERT could perform the query across multiple tables in a study.

To perform uploads, our team created a separate front end connected to the same database. At first, this front end only took in data specific to a user and a data category within a study. While this upload is tedious for the user, it helped our team determine how to add data into our database. In a later sprint, we expanded our upload process to be more automated and generalized, allowing the user to specify some conditions about the dataset and format of the files up front and then reading multiple folders and files to upload their data all at once. When asking to upload the data, the uploader asks the following series of questions:

1. Which study the data belongs to
2. If the study uses separate study groups (control, experimental, etc.), and if so, to enter them in
3. Which data category the data belongs to
4. If the files are sorted by one subject per file or one subject per row of a file
5. If the data is time series or not
6. If there are column labels present in the files

Using this information, our uploader adds the data into the correct place in the database, given these constraints:

- The files are in .csv format
- Each study group's files are placed in separate directories
- If one subject per file:
    - File name is <subject_id>_<data_category_name>
- If one subject per row:
    - File name is <data_category_name>
    - If subject IDs are present, it is the first column in the table
    - If the data is time series, there are no column labels

3.5.3 Developing a Django Web Application

The initial command line application acted as a proof of concept for what our application would consist of. The final version GILBERT is a web application that combines all the features in the command-line version of the application. Because GILBERT is a web application, it has a wider reach and is easily accessible to the users. Users do not have to install any external software to use our application.

Our team focused on using Python as the main programming language to develop the web application, which left us with two options of web frameworks to choose from: Django and Flask. While the two frameworks are both Python-based web frameworks that enable developers to build scalable applications, there are key differences between the two. Django is a heavy-weight framework that takes care of many of the standard functionalities to build secure and maintainable websites. On the other hand, Flask is a light-weight framework that provides some standard functionalities, but not as much when compared to Django. Because of that, Flask offers more flexibility (Protasiewicz, 2018). Despite this advantage, our team chose to use Django as the web framework to build our application. Django offers a better way to structure the code base as the project grows and becomes more complicated, which is better for us and for future teams. In addition, Django allows us to focus more on building the business logic of the application without having to focus on the other details involved when developing a secure web application.

After deciding to use Django to build our web application, we came across another key decision that we needed to make. Django offers the ability to map a model class to a single database table, which is a solution to abstract out the write-up of queries for creating, deleting, updating, or any other database related capabilities. The other option was for GILBERT to use raw SQL queries that were used in our initial command-line application. The option to make use of the Django models seemed favorable, but due to the team’s inexperience in working in a Django environment, we decided to use raw SQL queries to make the transition from the command line application to a web application go smoothly. Additionally, because our application dealt with dynamically created tables, we felt that using Django models would not work well for our situation. Our research did not suggest that Django had a straightforward solution to handle it. However, we would later integrate Django models into our application to help clean up the code and to reduce the overall complexity of it. We use Django models to the tables that are not dynamic, and we still use raw SQL to make the queries for the tables that are dynamic.

Chapter 4. GILBERT

This chapter outlines GILBERT, our biomedical data pipeline. The infrastructure of GILBERT is shown in *Figure 8*, with each box indicating a different component. Users interact with GILBERT using a web browser. Most of the application is hosted across multiple machines in the cloud via Heroku. Gunicorn, a web server gateway interface (WSGI), is used to host the Django portion of the application and handles web requests from users. The Django framework contains code related to the front-end and back-end logic of the application. PostgreSQL is the primary database of the application, storing uploaded data and user data. GILBERT uses Celery to run worker threads for the progress bar in the uploader feature (see Section 4.4 for details). Redis allows GILBERT to communicate with the Celery workers. GILBERT uses Amazon S3 storage to temporarily store uploaded files. In regard to deployment, Heroku designates separate machines for Gunicorn, PostgreSQL, and Redis. Celery is hosted as a task queue on the same machine as Gunicorn. Amazon S3 is not managed by Heroku and is on its own machine.

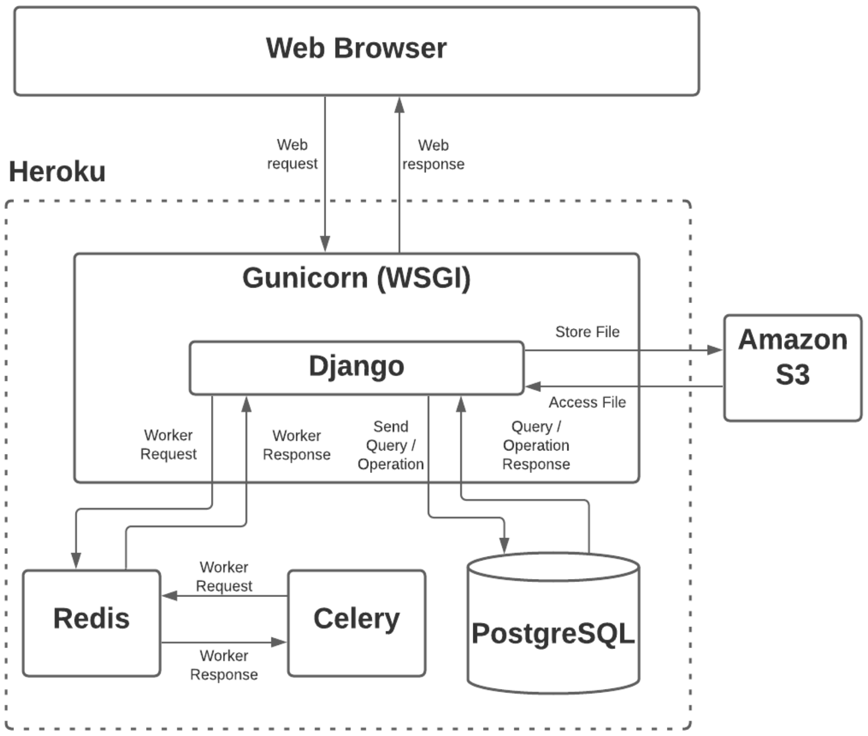


Figure 8. GILBERT infrastructure diagram.

The directory structure of GILBERT's Django component follows the standards of the Django framework. The "mqp_project" directory is the central directory for application-wide settings, URL routes, and static files such as .css files and images. GILBERT separates the different features of the application into Django "apps", which are modular collections of code. As shown in *Figure 9,* the apps in GILBERT are "Analysis", which handles summary statistics and graph plotting in the study analysis feature, "DataPipeline", which contains the dashboard page and handles filtered queries in the study analysis feature, "Inventory", which handles the inventory feature, "Uploader", which handles the uploader, and "Users", which handles the user system. Each app contains similar files. HTML template files structure the front end of each page of the app. A Python forms file contains form classes that inherit from built-in Django classes and define forms on the front end. A Python models file contains model classes that inherit from built-in Django classes and define the database schema of the application. A Python views file contains the back-end logic for each web page. A Python URLs file contains URL routes to link URLs to different back-end functions in the views file. The link to the GitHub repository containing GILBERT's code is included in Appendix H.

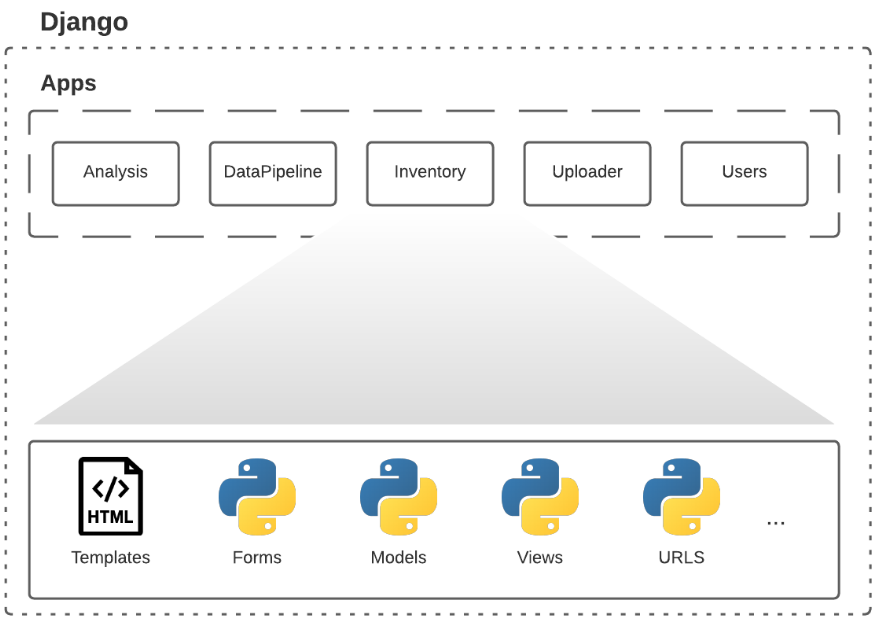


Figure 9. Django directory structure in GILBERT.

4.1 Database Schema

When designing the schema, we aimed to logically store different types of biomedical data, information about participants in studies, and study metadata. *Figure 10* is an entity-relationship diagram which describes our schema but is simplified to exclude table attributes. A full entity-relationship diagram with table attributes can be found in Appendix I. Each table has a unique identifier that serves as a primary key. Foreign keys link records across multiple tables. We used Django's built-in "Auth_user" table to store user account information including usernames, email addresses, and passwords. The

"Study" table contains information related to a study, such as its name, and additional metadata, including the study objective, institutions and organizations involved, start and end dates, and contact information. Each study is linked to the user that uploaded it. A table called "Study Group" represents groups of participants, such as control or experimental groups. Each group is linked to a study. A table called "Subject" represents participants in a study. Each subject is linked to a study group.

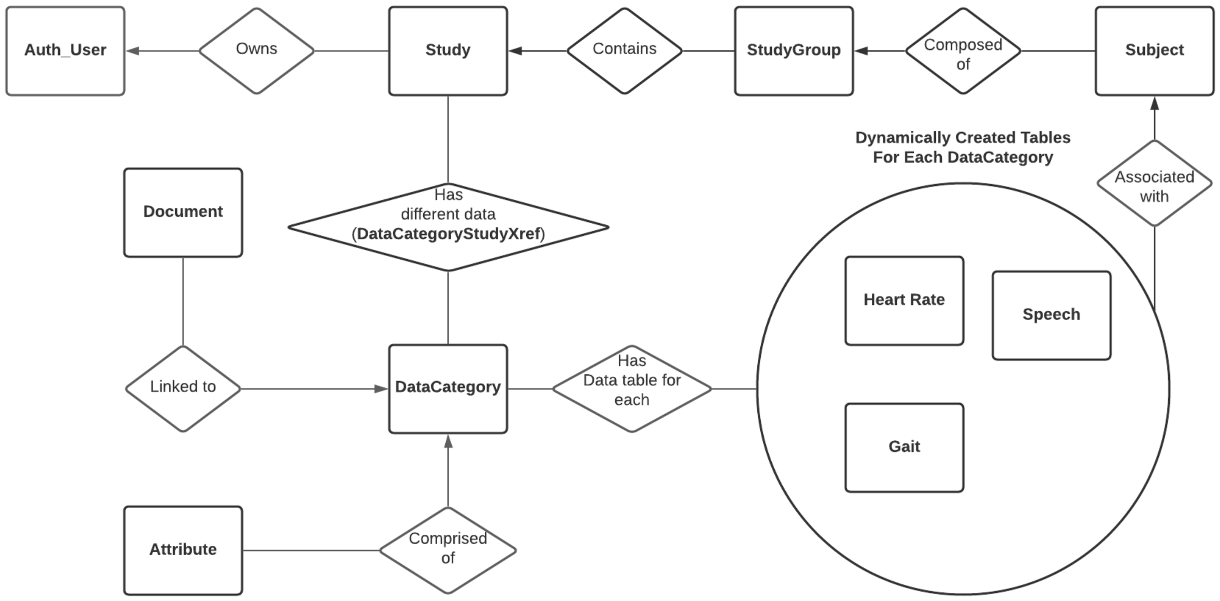
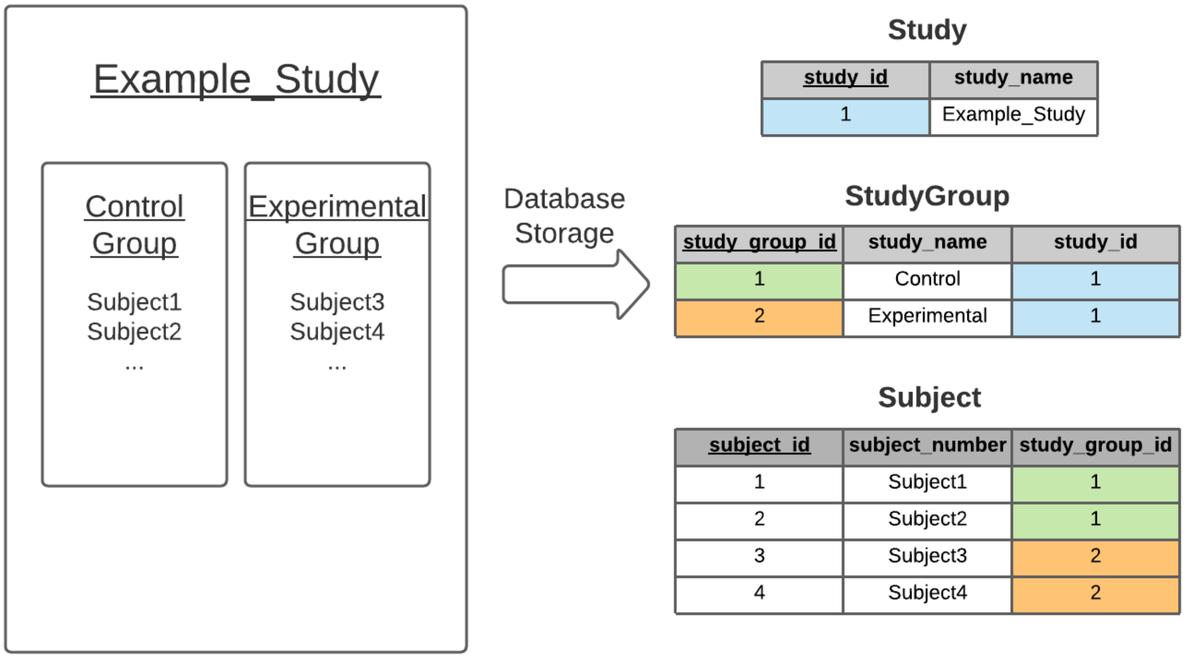


Figure 10. Simplified Entity-Relationship diagram for database schema.

Figure 11 displays a visual representation of how the components of a study are stored within GILBERT's database schema. The color coding indicates matching IDs between different tables, showing how the study, study groups, and subjects are linked together.



Study

study_id	study_name
1	Example_Study

StudyGroup

study_group_id	study_name	study_id
1	Control	1
2	Experimental	1

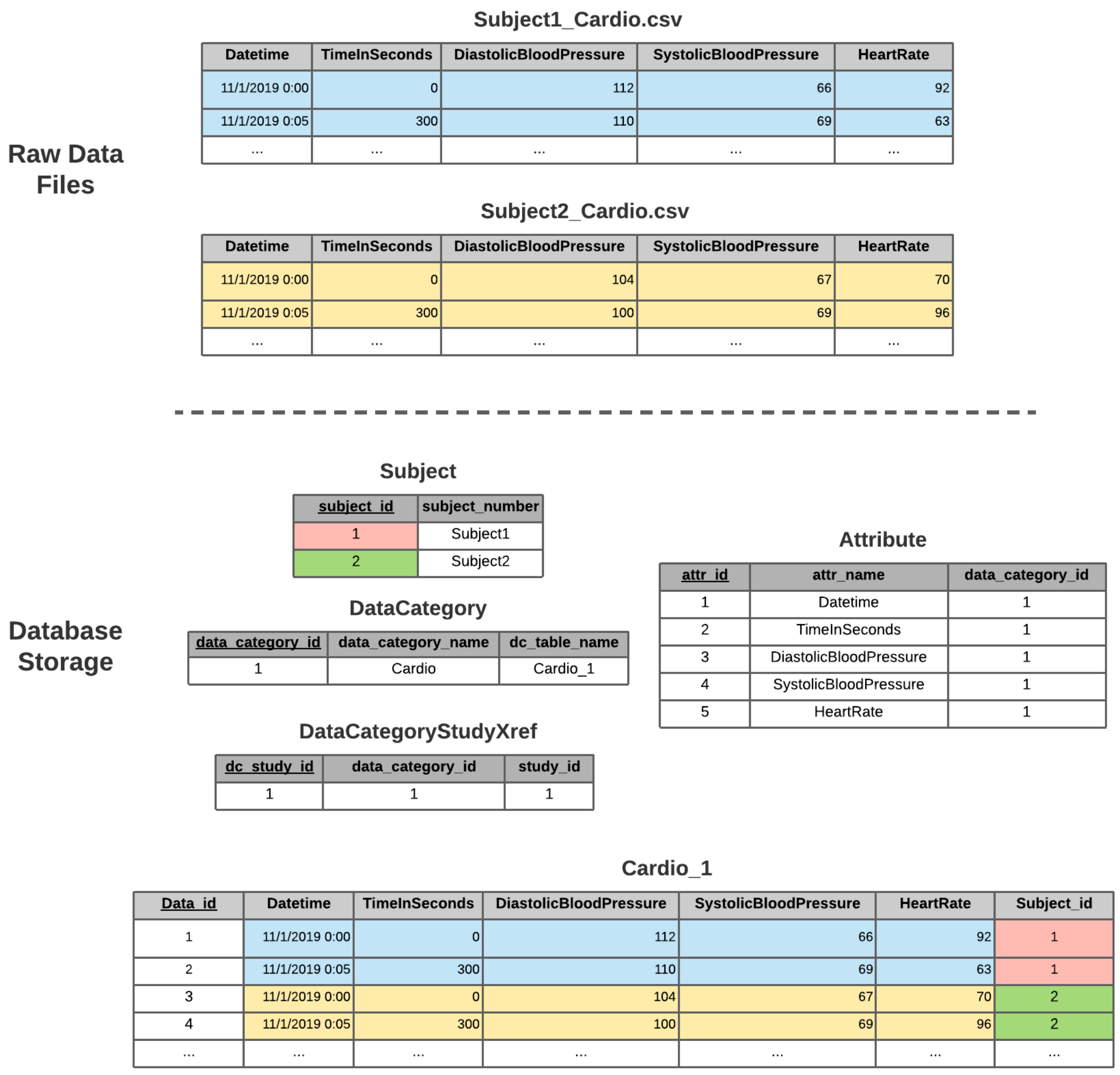
Subject

subject_id	subject_number	study_group_id
1	Subject1	1
2	Subject2	1
3	Subject3	2
4	Subject4	2

Figure 11. Visual representation of study storage in GILBERT's database.

Allowing our pipeline to handle a wide variety of data types and formats proved to be difficult since relational databases work best when the data they are handling follows a predefined schema. To store an arbitrary amount of different data types in a way that researchers could easily view and analyze them, GILBERT designs tables that match the format of the spreadsheets the data originated from. Based on user input, GILBERT dynamically creates tables with schema that match the files that the user uploaded. It stores data with the same columns from each subject within the same table and links each data point to its corresponding subject in the "Subject" table. In *Figure 10*, the "Heart Rate", "Gait", and "Speech" tables within the circle are examples of dynamically created tables. The naming scheme of the dynamically created tables are generated as "<data_category_name>_<study_id of the first study to use this data category>". This allows multiple data categories with the same name to exist as long as they belong to different studies. The "Data Category" table contains the names of the dynamically created tables and other metadata, such as whether the data category is a time series. For this project, we defined "Data Category" as a collection of attributes that would often be columns within the same spreadsheet and generally thematically related. We wanted to allow studies to have many data categories while also allowing multiple studies to have the same data category, which would occur if their data columns matched and allow similar data to be compared across multiple studies. "DataCategoryStudyXref" models this many-to-many relationship and represents the cross reference between which data categories correspond with which studies. To provide additional metadata about the attributes of data categories, GILBERT uses an “Attribute” table, which is linked to the “Data Category” table and includes information about the data type (integer, string, etc.), unit of measurement, and measuring device for each attribute. For uploading purposes, GILBERT uses a table called “Document” to link file names of uploaded files to the appropriate data category.

Figure 12 displays a visual representation of how raw data for a study is stored within GILBERT’s database. In this example, the Data Category format is Subject per File. Data Categories are discussed in Section 4.4. The color coding indicates matching subject IDs between tables and shows where and how the raw data is formatted within the database. In this case, two files with the same columns contain cardiovascular data about two subjects. When uploaded, GILBERT puts this data together into a data category with a user selected name (in this case, “Cardio”). The data is stored in a table called “Cardio_1” since the data category name is “Cardio” and the study ID is 1. This table has all the same columns as the raw .csv files, with additional columns for a primary key and foreign key to link each data point to its corresponding subject.



Subject1_Cardio.csv

Datetime	TimeInSeconds	DiastolicBloodPressure	SystolicBloodPressure	HeartRate
11/1/2019 0:00	0	112	66	92
11/1/2019 0:05	300	110	69	63
...	...	...	...	...

Subject2_Cardio.csv

Datetime	TimeInSeconds	DiastolicBloodPressure	SystolicBloodPressure	HeartRate
11/1/2019 0:00	0	104	67	70
11/1/2019 0:05	300	100	69	96
...	...	...	...	...

Subject

subject_id	subject_number
1	Subject1
2	Subject2

DataCategory

data_category_id	data_category_name	dc_table_name
1	Cardio	Cardio_1

DataCategoryStudyXref

dc_study_id	data_category_id	study_id
1	1	1

Attribute

attr_id	attr_name	data_category_id
1	Datetime	1
2	TimeInSeconds	1
3	DiastolicBloodPressure	1
4	SystolicBloodPressure	1
5	HeartRate	1

Cardio_1

Data_id	Datetime	TimeInSeconds	DiastolicBloodPressure	SystolicBloodPressure	HeartRate	Subject_id
1	11/1/2019 0:00	0	112	66	92	1
2	11/1/2019 0:05	300	110	69	63	1
3	11/1/2019 0:00	0	104	67	70	2
4	11/1/2019 0:05	300	100	69	96	2
...	...	...	...	...	...	...

Figure 12. Visual representation of raw data storage example in GILBERT's database.

4.2 Dashboard and Navigation Bar

When designing our dashboard, we kept two factors in mind: simplicity and ease of access. The dashboard is the first page of GILBERT that users will see, so it is important that the user interface is not overwhelming. Also, we found it necessary to have each of the core features of GILBERT immediately accessible to users on the dashboard. This includes our study analysis, upload, and inventory features, and are shown on the dashboard in *Figure 13*.

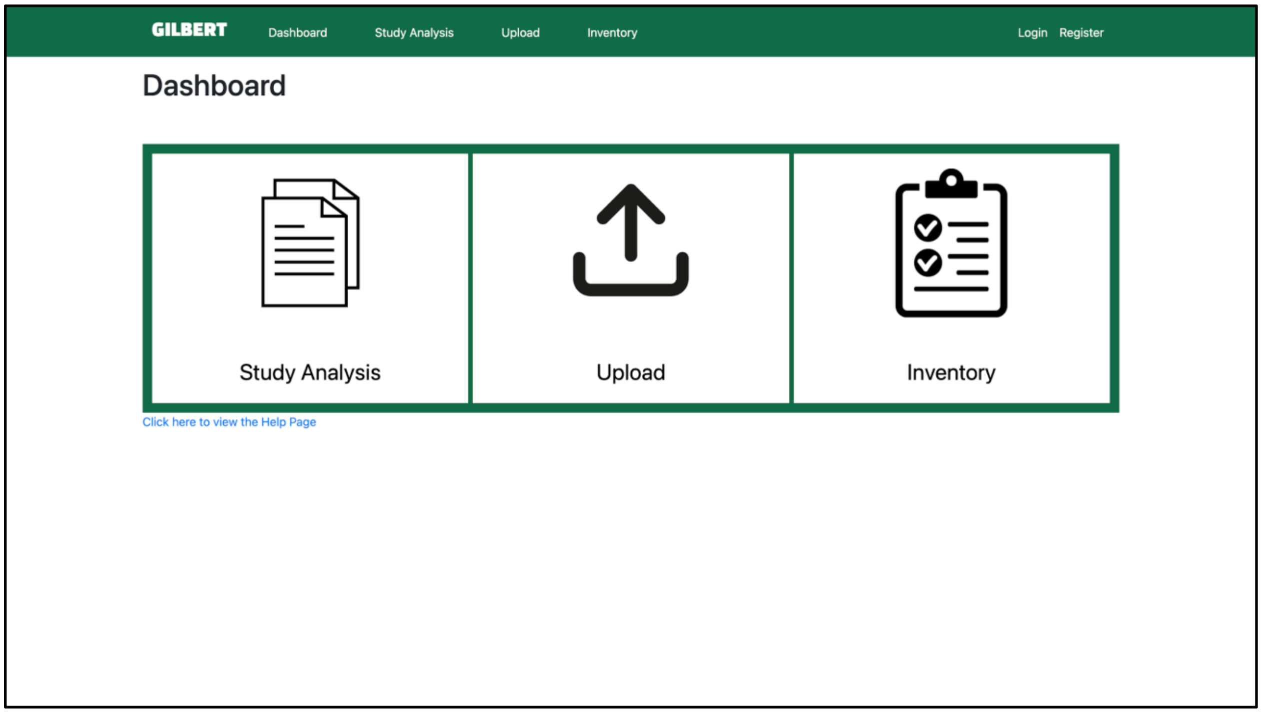


Figure 13. Dashboard, with study analysis, upload, and inventory features.

Having these features readily available helps users to quickly and easily find the tools they would like to use as soon as they open the application. To extend this ease of access throughout the application, we added a navigation bar to the top of every page. This allows the user to switch to GILBERT's features from any point in the pipeline. The navigation bar also includes a login and a register button to access our user system if the user is not logged in. If the user is logged in, the login and register buttons will be replaced with a profile button for viewing and modifying the user's account information and a logout button. The two variants of the navigation bar are shown in *Figure 14* and *Figure 15.*

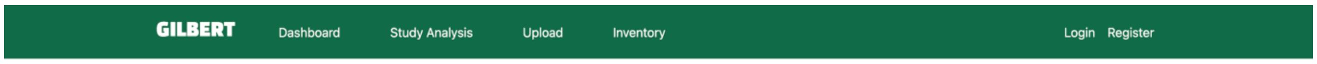


Figure 14. Navigation bar when the user is not logged in.

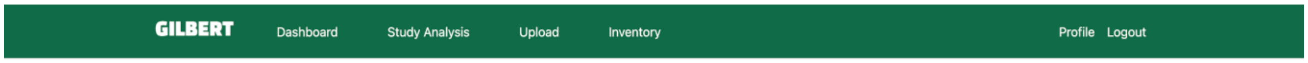


Figure 15. Navigation bar when the user is logged in.

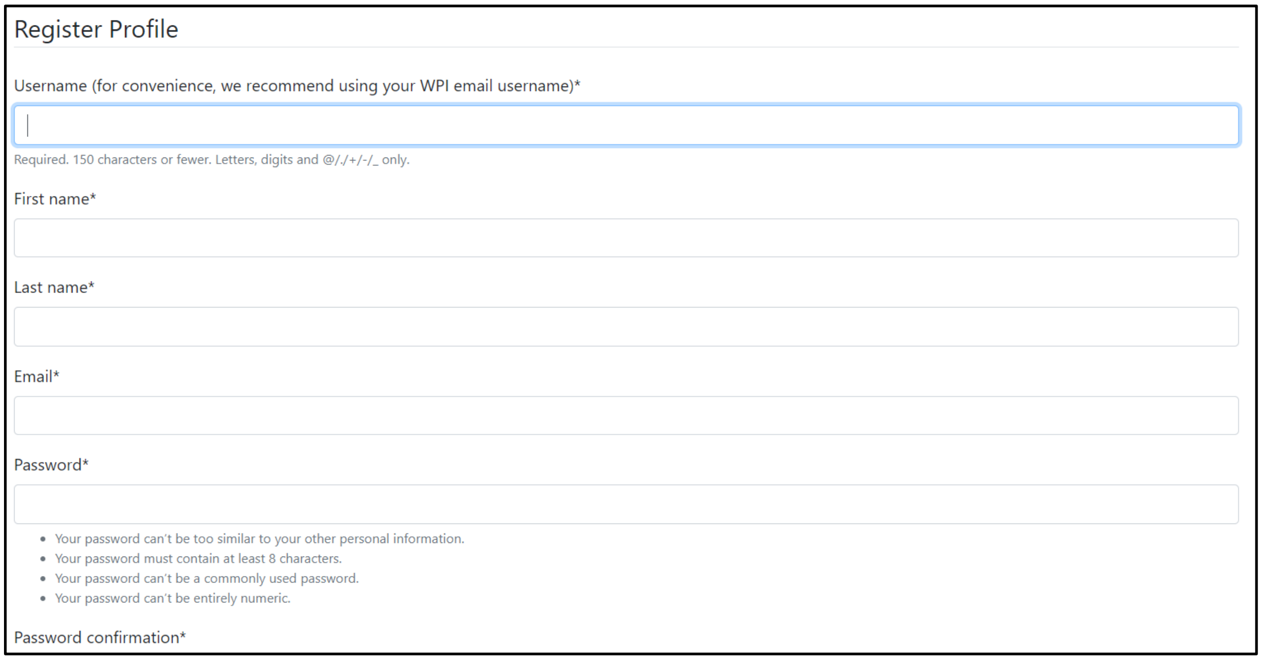
4.3 User System

To provide security and privacy measures, GILBERT has a system that restricts the access of certain data to certain users. Rather than requiring a custom user system, GILBERT leverages the

"Auth_User" table and various authentication measures built into Django. When storing user information into the "Auth_User" table, GILBERT utilizes Django functions to perform a SHA256 hash on the password and then stores the hash. Since the hash is a one-way process, it protects the user's information. Once a user is logged in, Django authenticates the user using sessions. Sessions are made of temporary data that is maintained across different pages. They are stored in a database table, with a corresponding ID stored in the browser's cookies. GILBERT uses sessions to determine which user is logged in.

The user must be logged in to use the uploader feature, but it is not necessary for the Study Analysis and Inventory features. This is because GILBERT links the user's account to the studies they upload. GILBERT uses a permission system to control which studies users are able to see or interact with. Each study has a visibility field that is set to Public or Private. Public studies can be viewed in the Study Analysis and Inventory features by anyone. Private studies can only be seen by their owner. The only user that can upload to a study is the owner of the study. GILBERT uses this permission system to protect the privacy of the study data stored within it.

GILBERT has a registration page for creating an account and a login page for logging into it. Usernames and email addresses must be unique for each user. *Figure 16* displays the registration page and *Figure 17* displays the login page. Both pages use built-in Django forms.


Register Profile

Username (for convenience, we recommend using your WPI email username)*

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

First name*

Last name*

Email*

Password*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation*

Figure 16. Registration page of user system.

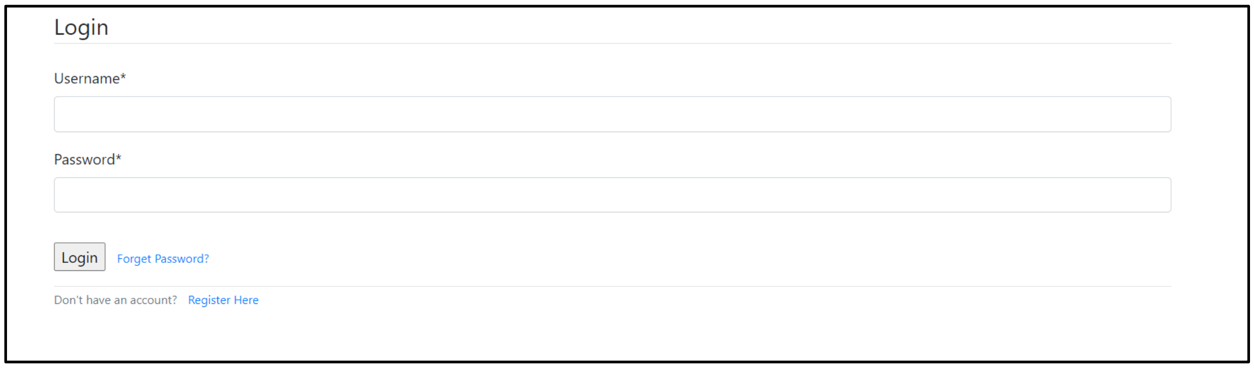


Figure 17. Login page of user system.

When logged in, the user can update their profile information on the Profile page, shown in *Figure 18*. This page also allows users to delete their account, which deletes their data from the "Auth_User" table.

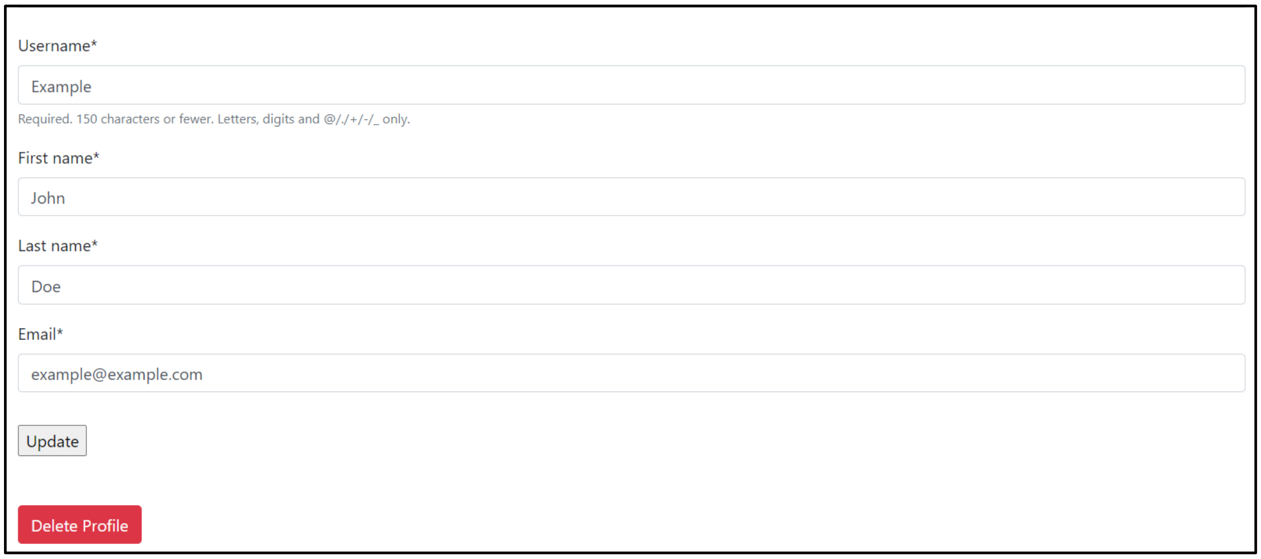


Figure 18. Profile page of user system.

If a user forgets their password, GILBERT allows them to reset it by clicking the "Forgot Password?" link from the login page shown in *Figure 16* and then providing the email address of the account. If an account exists with this email address, the address receives an email with a link to provide a new password. This link would have a unique and randomized string in the URL. Currently, GILBERT uses a Gmail account to send these emails. The pages for resetting a password make use of built-in Django forms and logic.

4.4 Uploader

One of GILBERT's key responsibilities is to allow users to upload their own sets of data. By uploading data to the application, users of the system can perform different types of analysis on it. *Figure 19* displays the flow between each page within the uploader section of GILBERT.

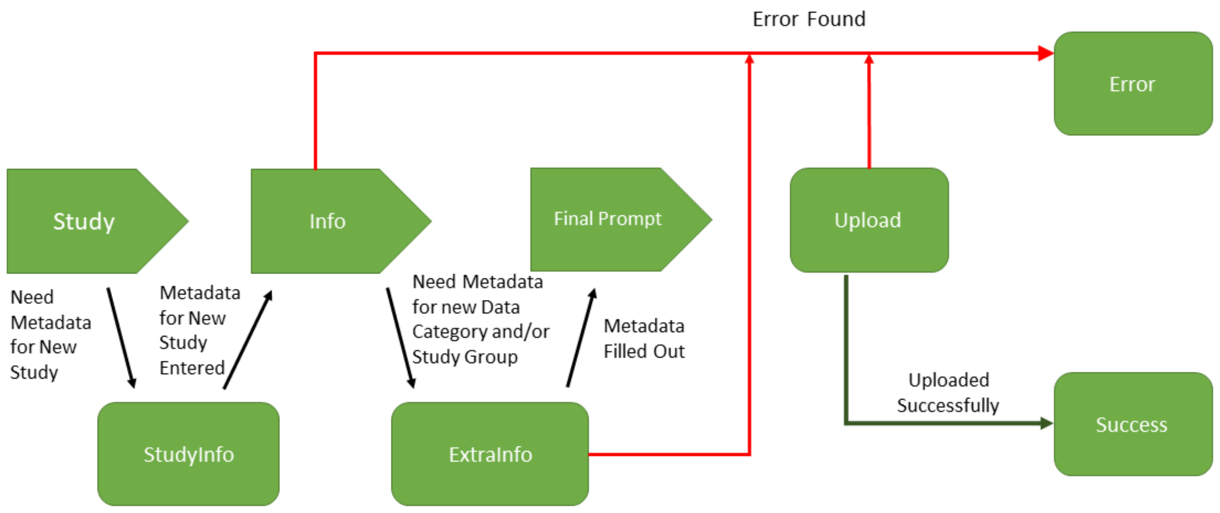
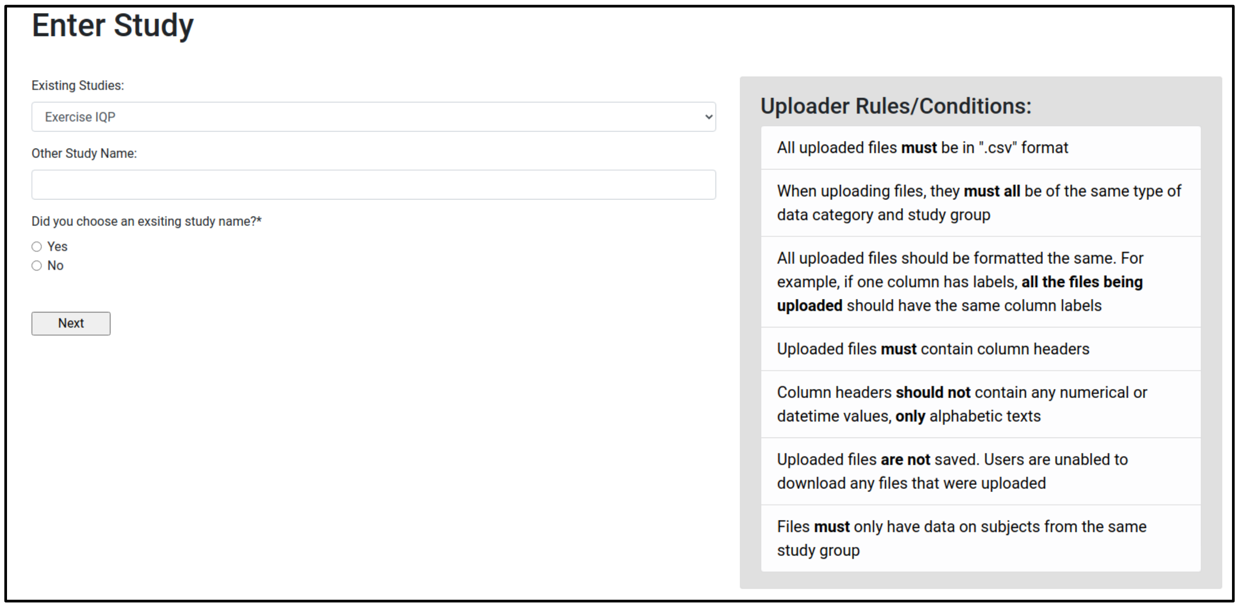

Error Found
Error
Study
Info
Final Prompt
Upload
Need Metadata for New Study
Metadata for New Study Entered
Need Metadata for new Data Category and/or Study Group
Metadata Filled Out
Uploaded Successfully
StudyInfo
ExtraInfo
Success

Figure 19. Flow chart of uploader.

The first page of this feature is the Study page, shown in *Figure 20*. At this page, the user is prompted to either choose an existing study name or enter in a new one. The user's action determines the next page that they are redirected to. If the user enters a new study name, the system redirects the user to the study info page where the system asks for the metadata of the new study, shown in *Figure 21*. If the user chooses an existing study name, then the user is redirected to the info page, as shown in *Figure 22*, which is where the user uploads their files. This page is the next step in the flow of the uploader regardless of the route the user takes from the first page of the uploader feature.


Enter Study

Existing Studies:

Exercise IQP

Other Study Name:

Did you choose an exsiting study name?*

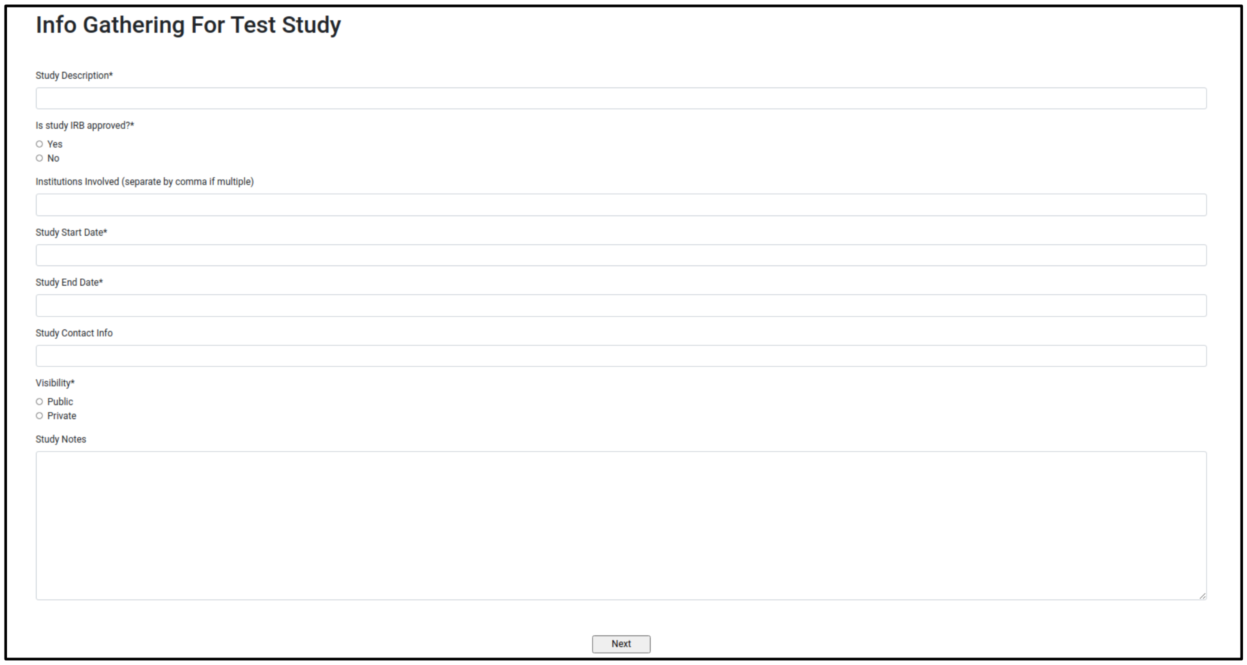
- Yes
- No

Next

Uploader Rules/Conditions:

- All uploaded files **must** be in ".csv" format
- When uploading files, they **must all** be of the same type of data category and study group
- All uploaded files should be formatted the same. For example, if one column has labels, **all the files being uploaded** should have the same column labels
- Uploaded files **must** contain column headers
- Column headers **should not** contain any numerical or datetime values, **only** alphabetic texts
- Uploaded files **are not** saved. Users are unabled to download any files that were uploaded
- Files **must** only have data on subjects from the same study group

Figure 20. Study page of uploader.


Info Gathering For Test Study

Study Description*

Is study IRB approved?*

- Yes
- No

Institutions Involved (separate by comma if multiple)

Study Start Date*

Study End Date*

Study Contact Info

Visibility*

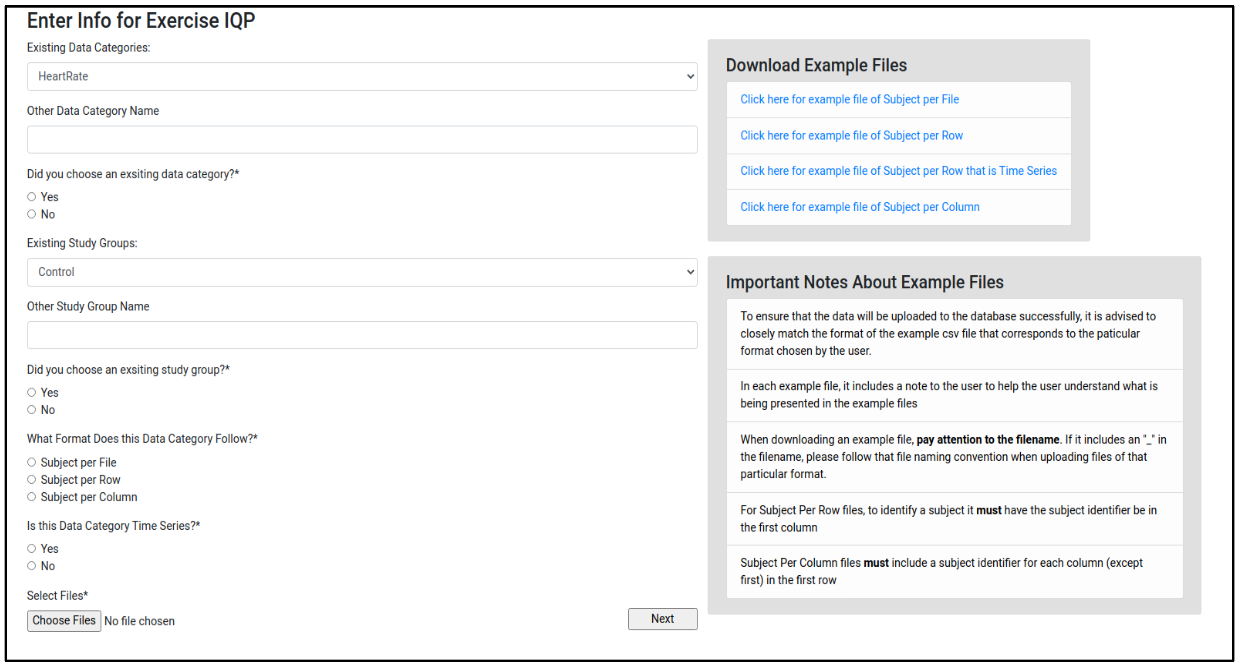
- Public
- Private

Study Notes

Next

Figure 21. Study info page of uploader.

The responsibility of the info page is to learn about the format and organization of the file(s) that GILBERT will be uploading. Additionally, the user is prompted to provide information about the data category(s) and study group(s) that the uploaded file(s) correspond to. These prompts are found in the top portion of the info page. If a data category and/or study group exists for the study, the user is able to select them through a drop-down menu. The user will always have the option to add new Data categories and/or study groups by entering in the names into the text field seen in *Figure 22*. Data categories can have three different file formats: Subject per File, Subject per Row, or Subject per Column. If a data category is Subject per File, each file being uploaded contains data about a different subject, with each column being a different attribute. If a data category is Subject Per Row, each row of a file being uploaded would correspond with a different subject, with each column being a different attribute or interval in a time series. If a data category is Subject Per Column, each column of a file being uploaded would correspond with a different subject, with each row being a different attribute or interval in a time series. The info page provides example files of each format type for reference.


Enter Info for Exercise IQP

Existing Data Categories: HeartRate

Other Data Category Name:

Did you choose an exsiting data category?*
- Yes
- No

Existing Study Groups: Control

Other Study Group Name:

Did you choose an exsiting study group?*
- Yes
- No

What Format Does this Data Category Follow?*
- Subject per File
- Subject per Row
- Subject per Column

Is this Data Category Time Series?*
- Yes
- No

Select Files*

Choose Files No file chosen

Next

Download Example Files
- Click here for example file of Subject per File
- Click here for example file of Subject per Row
- Click here for example file of Subject per Row that is Time Series
- Click here for example file of Subject per Column

Important Notes About Example Files
- To ensure that the data will be uploaded to the database successfully, it is advised to closely match the format of the example csv file that corresponds to the paticular format chosen by the user.
- In each example file, it includes a note to the user to help the user understand what is being presented in the example files
- When downloading an example file, **pay attention to the filename**. If it includes an "_" in the filename, please follow that file naming convention when uploading files of that particular format.
- For Subject Per Row files, to identify a subject it **must** have the subject identifier be in the first column
- Subject Per Column files **must** include a subject identifier for each column (except first) in the first row

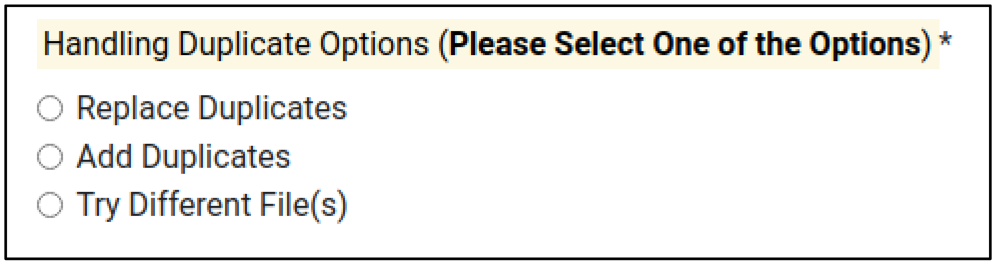
Figure 22. Info page of uploader.

Due to the possible variability of data, the team defined standards for the user to follow. The standards are the following:

- Files must be in .csv format
- All files that are uploaded together must have the same type of data category and study group
- Files must be formatted as Subject per File, Subject per Row, or Subject per Column
- All files that are uploaded together must be formatted the same and have the same column headers but can have their column headers in a different order from the data category as long as all the files being uploaded together at once have the same order

- Files must contain column headers with the exception of the file being formatted as Subject per Row and is time series
- Column headers must only contain alphabetical characters and not include any numerical or datetime values
- Files must only have data on subjects from the same study group

In an effort to verify that the standards are followed, GILBERT performs some error checking. Additionally, the application also checks for attempts to upload duplicate data into the database. GILBERT stores all the names of the files that were uploaded and to check for duplicate files, it verifies if any of the names of the files currently being uploaded already exist. In an event that the system found a matching file name within the database, a warning message informs the user that an attempt to upload a duplicate file was found. A new prompt appears on the page asking the user how they would like to handle the duplicate file(s), as shown in *Figure 23*.

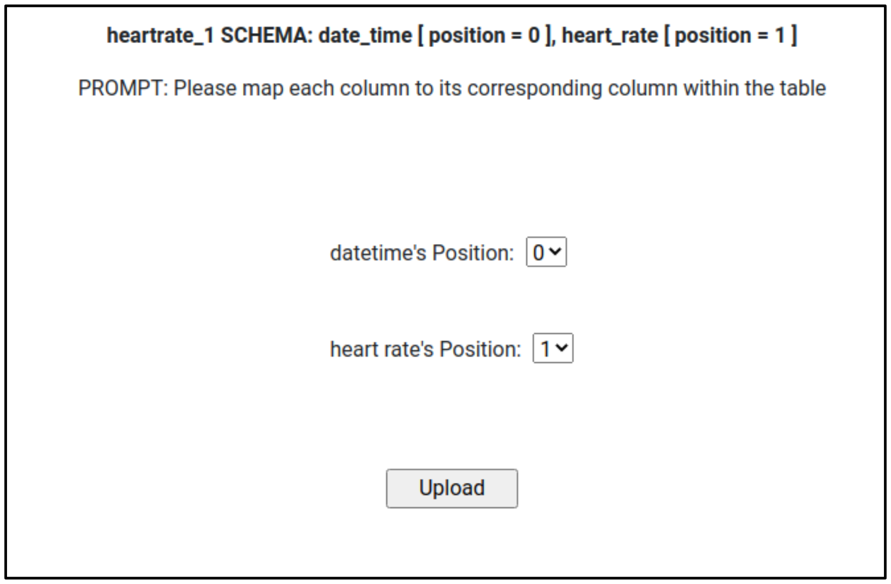

Handling Duplicate Options (**Please Select One of the Options**) *

- ○ Replace Duplicates
- ○ Add Duplicates
- ○ Try Different File(s)

Figure 23. Available options for handling uploading duplicate files.

If the user selects a new data category or study group, the user is redirected to a page where they are asked to provide relevant metadata. Then, the user is redirected to the next stage in the uploader feature, the final prompt page. If the user selects existing data category(s) and study group(s), the user is taken directly to the final prompt page.

The purpose of the final prompt page, shown in *Figure 24,* is to provide the user the option to remap the columns within their uploaded files to match the corresponding column defined within the database. Because there is no guarantee that the order of the columns and their names will match the database table that the data will be uploaded to, the user can remap the columns without having to make the edits in their own files.


heartrate_1 SCHEMA: date_time [position = 0], heart_rate [position = 1]

PROMPT: Please map each column to its corresponding column within the table

datetime's Position: 0

heart rate's Position: 1

Figure 24. Final prompt page of uploader.

At the page shown in *Figure 25*, all the needed information to upload data to the database has been provided, and the user can start the actual process of uploading the data. A progress bar is displayed on the page to keep the user updated on the uploader process. GILBERT handles the upload to the database asynchronously to prevent the main thread of the website from being blocked. This requires Celery and Redis. In Django, the logic to upload the data is implemented in a task, which is a Python function that is specifically used by Celery. Then, Celery uses workers to run the function asynchronously. Django communicates with the Celery workers via a message broker, which is responsible for accepting tasks from Django and delivering them to the worker(s). GILBERT uses Redis, a commonly used message broker.

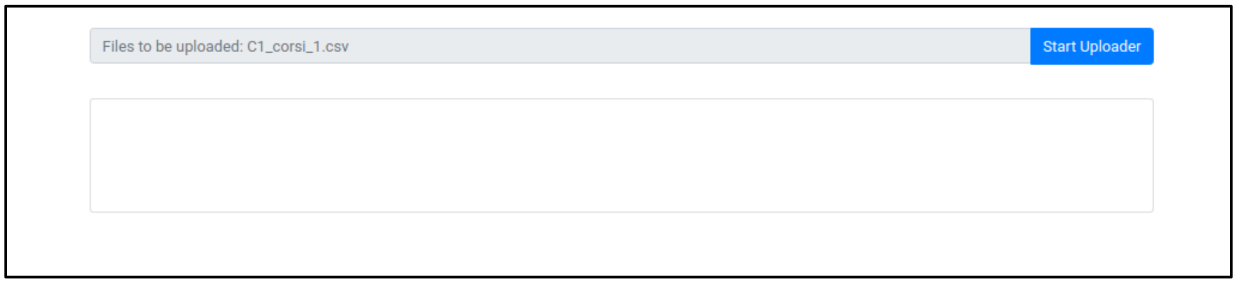
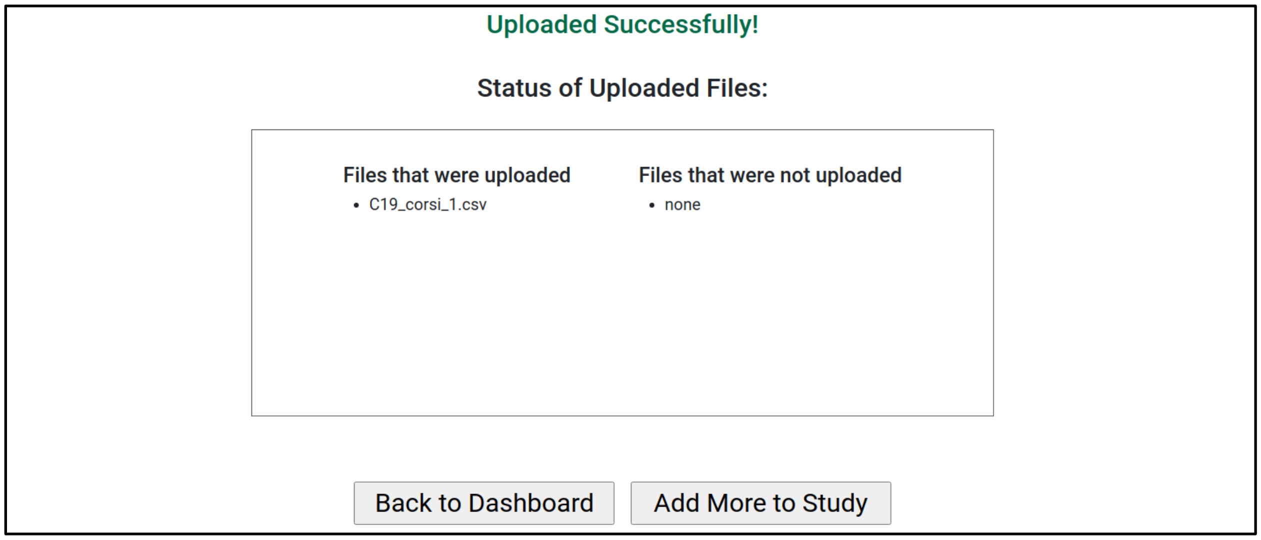

Files to be uploaded: C1_corsi_1.csv

Figure 25. Upload page of uploader.

If the user reaches the page shown in *Figure 26*, the process of uploading the data to the database was successful. The user can see the names of the files that they just uploaded to GILBERT under the column that indicates which files were uploaded successfully. From this page, the user can either continue to upload more data to the same study or go back to the dashboard.

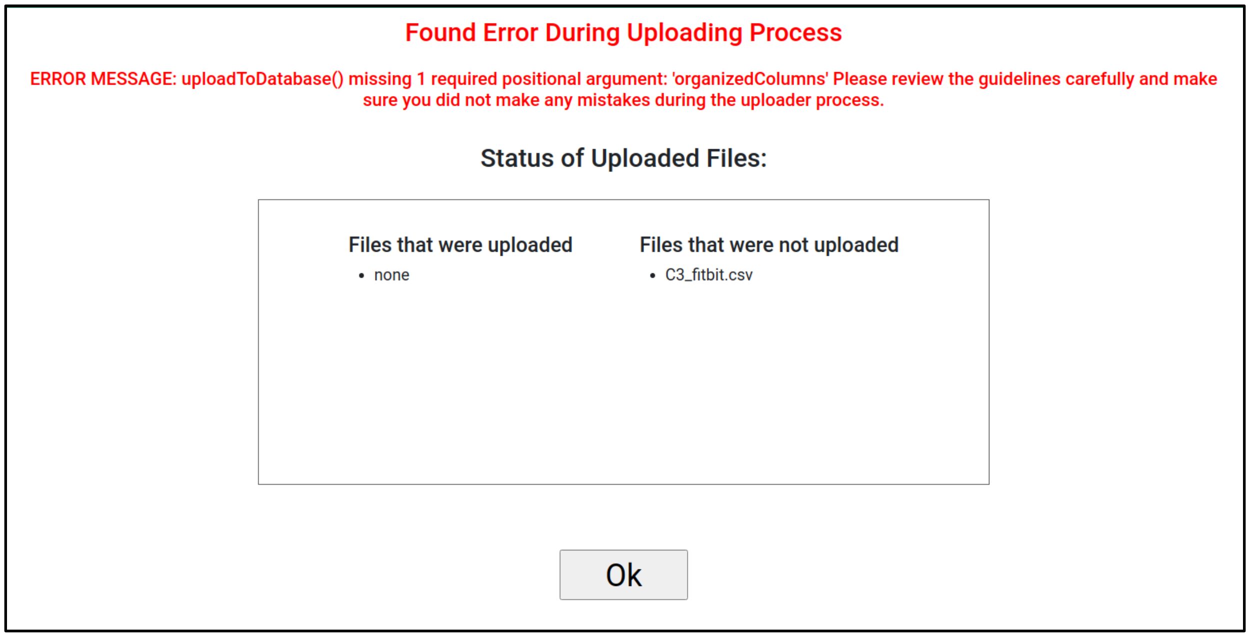

Uploaded Successfully!

Status of Uploaded Files:

Files that were uploaded	Files that were not uploaded
• C19_corsi_1.csv	• none

Figure 26. Success page of uploader.

If GILBERT detected any errors at any point of the upload process, the user is directed to the error page as shown in *Figure 27*. An error message appears at the top of the page which describes the error that was detected. Similarly to the success page, the user sees a list of the names of the files that were successfully uploaded to GILBERT and those that were not. This tells the user which files they need to upload again. If the cause of error occurred during the process of inserting new entries into the database, GILBERT performs a rollback to put the database at the state before the recent insert statements had occurred. By performing a rollback, it maintains that the database remains in a clean state despite having an error that occurred.


Found Error During Uploading Process

ERROR MESSAGE: uploadToDatabase() missing 1 required positional argument: 'organizedColumns' Please review the guidelines carefully and make sure you did not make any mistakes during the uploader process.

Status of Uploaded Files:

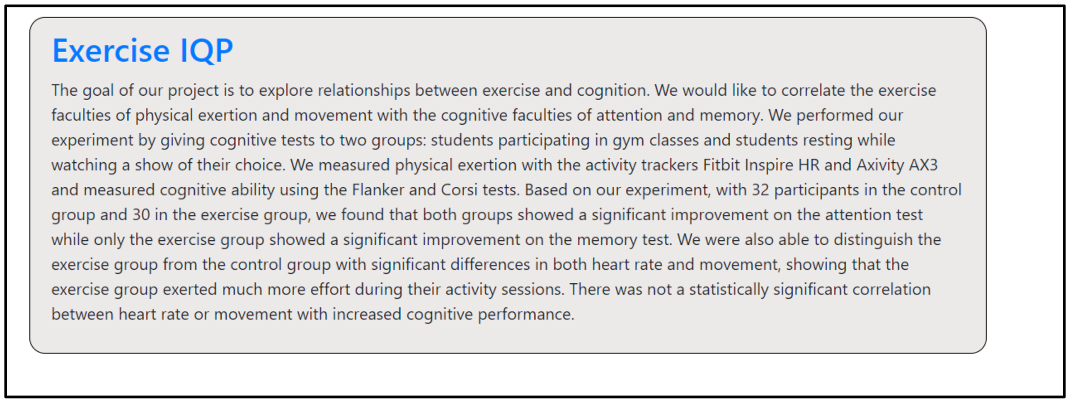
Files that were uploaded	Files that were not uploaded
• none	• C3_fitbit.csv

Figure 27. Error page of uploader.

4.5 Inventory

GILBERT's inventory feature allows users to search for studies that they have access to and view their metadata. This is useful for searching for studies based on their metadata and quickly viewing different statistics about a study.

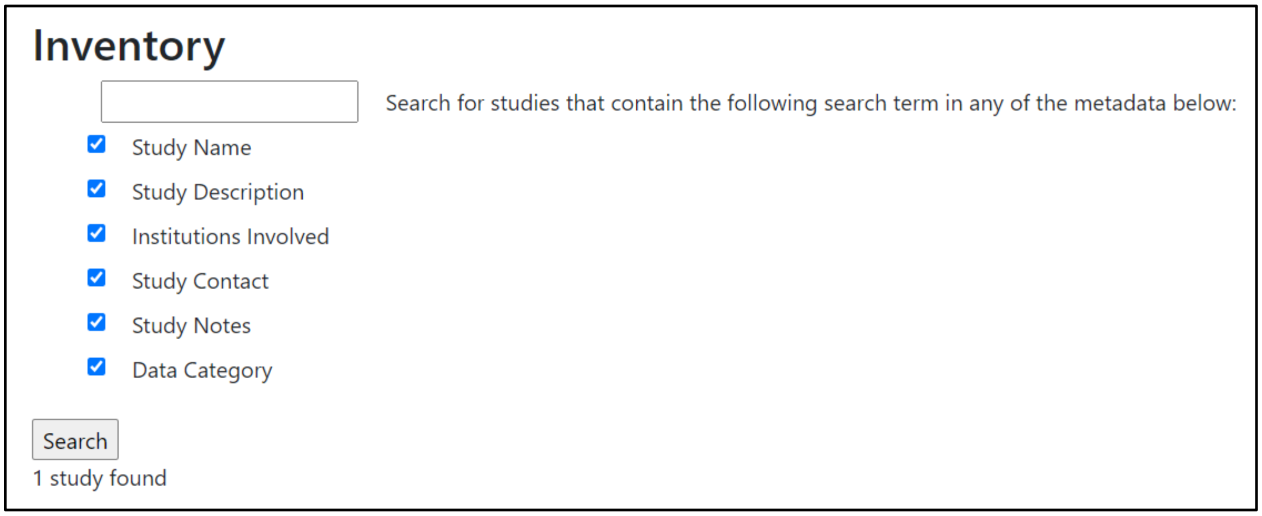
The first page in this feature displays the name and description of all studies that the user has access to. *Figure 28* shows an example of how a study is shown.


Exercise IQP

The goal of our project is to explore relationships between exercise and cognition. We would like to correlate the exercise faculties of physical exertion and movement with the cognitive faculties of attention and memory. We performed our experiment by giving cognitive tests to two groups: students participating in gym classes and students resting while watching a show of their choice. We measured physical exertion with the activity trackers Fitbit Inspire HR and Axivity AX3 and measured cognitive ability using the Flanker and Corsi tests. Based on our experiment, with 32 participants in the control group and 30 in the exercise group, we found that both groups showed a significant improvement on the attention test while only the exercise group showed a significant improvement on the memory test. We were also able to distinguish the exercise group from the control group with significant differences in both heart rate and movement, showing that the exercise group exerted much more effort during their activity sessions. There was not a statistically significant correlation between heart rate or movement with increased cognitive performance.

Figure 28. Study name and description on inventory search page.

At the top of the page, there is a search bar, as shown in *Figure 29*, with different checkboxes that allow the user to search for studies that have metadata that contain a certain search term. Searchable metadata for this includes study name, study description, institutions involved in study, contact information, notes, and names of data categories that studies use. Using this search filters the studies that appear on the page. Currently, GILBERT checks if the search term is a substring of the selected metadata for each study.


Inventory

Search for studies that contain the following search term in any of the metadata below:

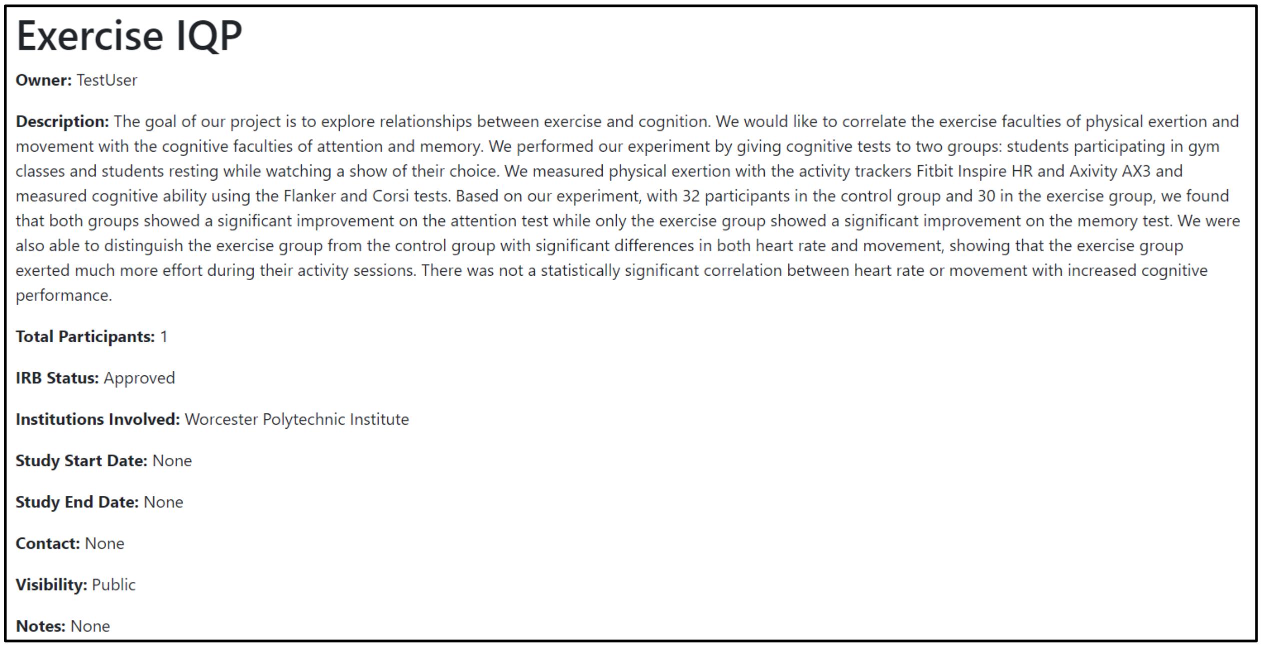
- [✓] Study Name
- [✓] Study Description
- [✓] Institutions Involved
- [✓] Study Contact
- [✓] Study Notes
- [✓] Data Category

Search

1 study found

Figure 29. Search functionality on inventory search page.

The name of each study is a hyperlink to another page that displays a variety of metadata about the study. This includes the study's description, total participants, IRB status, institutions involved, start date, end date, contact information, visibility, and notes. The study owner's username is displayed beneath the study name. Most of this data is readily available in the "Study" table of the database except for total participants, which requires a joined query across the "Study Group" and "Subject" tables. *Figure 30* displays a partial screenshot of how this data is shown within the application. This screenshot uses a test study called "Exercise IQP".



Exercise IQP

Owner: TestUser

Description: The goal of our project is to explore relationships between exercise and cognition. We would like to correlate the exercise faculties of physical exertion and movement with the cognitive faculties of attention and memory. We performed our experiment by giving cognitive tests to two groups: students participating in gym classes and students resting while watching a show of their choice. We measured physical exertion with the activity trackers Fitbit Inspire HR and Axivity AX3 and measured cognitive ability using the Flanker and Corsi tests. Based on our experiment, with 32 participants in the control group and 30 in the exercise group, we found that both groups showed a significant improvement on the attention test while only the exercise group showed a significant improvement on the memory test. We were also able to distinguish the exercise group from the control group with significant differences in both heart rate and movement, showing that the exercise group exerted much more effort during their activity sessions. There was not a statistically significant correlation between heart rate or movement with increased cognitive performance.

Total Participants: 1

IRB Status: Approved

Institutions Involved: Worcester Polytechnic Institute

Study Start Date: None

Study End Date: None

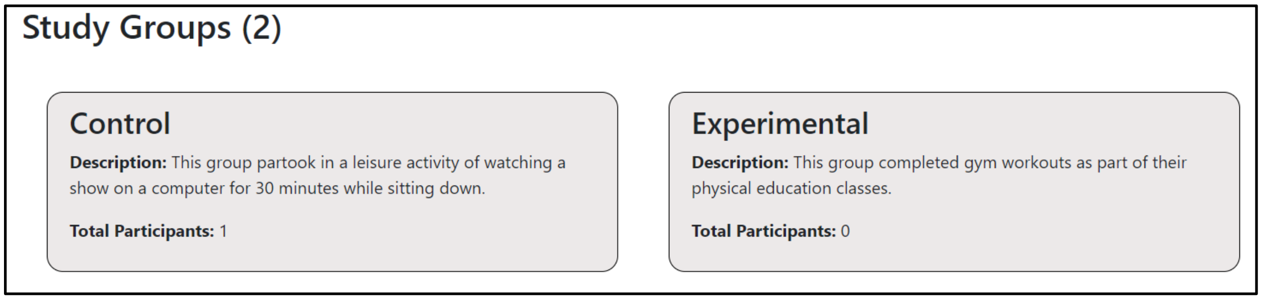
Contact: None

Visibility: Public

Notes: None

Figure 30. Study metadata on the inventory study page.

The study groups of the study are displayed on the page as well, as shown with example data in *Figure 31*. This shows the description and total participants for each of the groups. The total participants require a joined query across the “Study Group” and “Subject” tables.



Study Groups (2)

Control

Description: This group partook in a leisure activity of watching a show on a computer for 30 minutes while sitting down.

Total Participants: 1

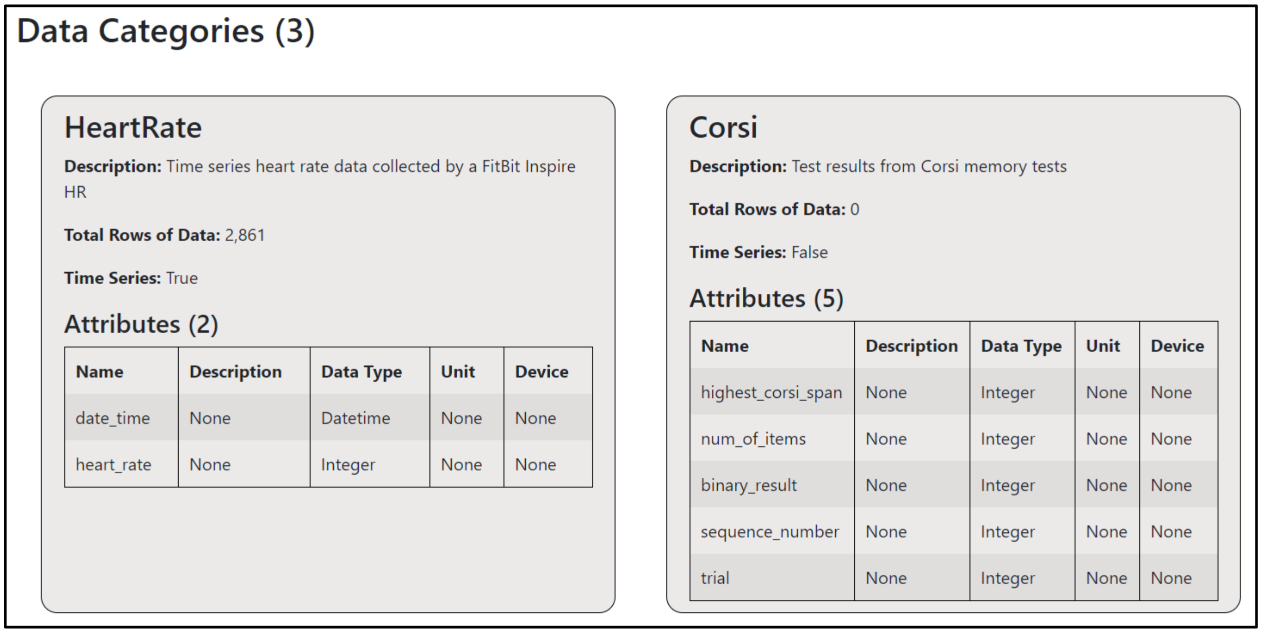
Experimental

Description: This group completed gym workouts as part of their physical education classes.

Total Participants: 0

Figure 31. Study group metadata on the inventory study page.

The data categories of the study are displayed on the page as well, as shown with example data in *Figure 32*. This shows the description, total rows of data, whether it is a time series, and a table of the column attributes for each data category. The total rows of data require a joined query that sums the number of rows in each data category table that is linked to the subject within the study.



Data Categories (3)

HeartRate

Description: Time series heart rate data collected by a FitBit Inspire HR

Total Rows of Data: 2,861

Time Series: True

Attributes (2)

Name	Description	Data Type	Unit	Device
date_time	None	Datetime	None	None
heart_rate	None	Integer	None	None

Corsi

Description: Test results from Corsi memory tests

Total Rows of Data: 0

Time Series: False

Attributes (5)

Name	Description	Data Type	Unit	Device
highest_corsi_span	None	Integer	None	None
num_of_items	None	Integer	None	None
binary_result	None	Integer	None	None
sequence_number	None	Integer	None	None
trial	None	Integer	None	None

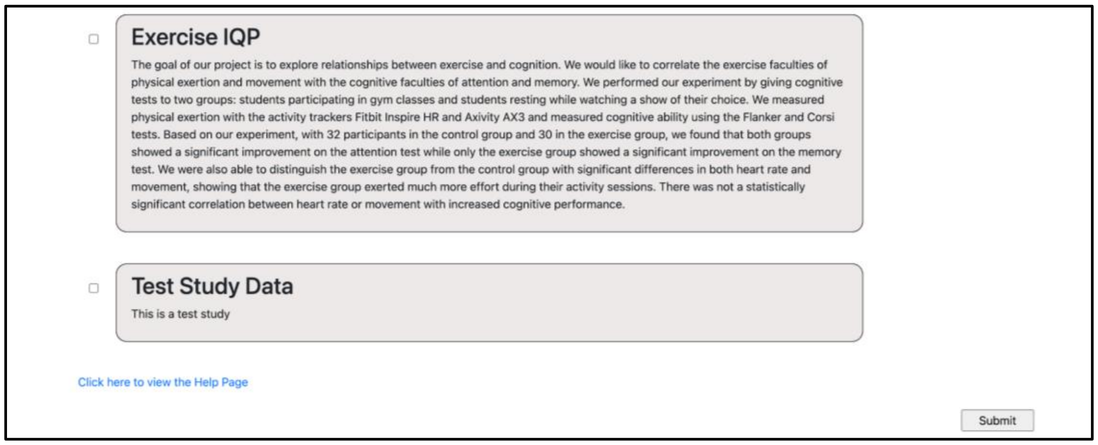
Figure 32. Data category metadata on the inventory study page.

4.6 Data Selection and Analysis

The data selection and analysis portion of GILBERT allows users to select and filter existing data from the database, export this data for use in other systems, and create visualizations in the application. Once the data is selected, GILBERT offers several forms of analysis to perform on the data.

4.6.1 Data Selection

Going through GILBERT's selection process allows the system to construct a SQL query from user input. This query is used to select the data that the user wants to see from the database. First, the user selects the study they want to view, as shown in *Figure 33*.



Exercise IQP

The goal of our project is to explore relationships between exercise and cognition. We would like to correlate the exercise faculties of physical exertion and movement with the cognitive faculties of attention and memory. We performed our experiment by giving cognitive tests to two groups: students participating in gym classes and students resting while watching a show of their choice. We measured physical exertion with the activity trackers Fitbit Inspire HR and Axivity AX3 and measured cognitive ability using the Flanker and Corsi tests. Based on our experiment, with 32 participants in the control group and 30 in the exercise group, we found that both groups showed a significant improvement on the attention test while only the exercise group showed a significant improvement on the memory test. We were also able to distinguish the exercise group from the control group with significant differences in both heart rate and movement, showing that the exercise group exerted much more effort during their activity sessions. There was not a statistically significant correlation between heart rate or movement with increased cognitive performance.

Test Study Data

This is a test study

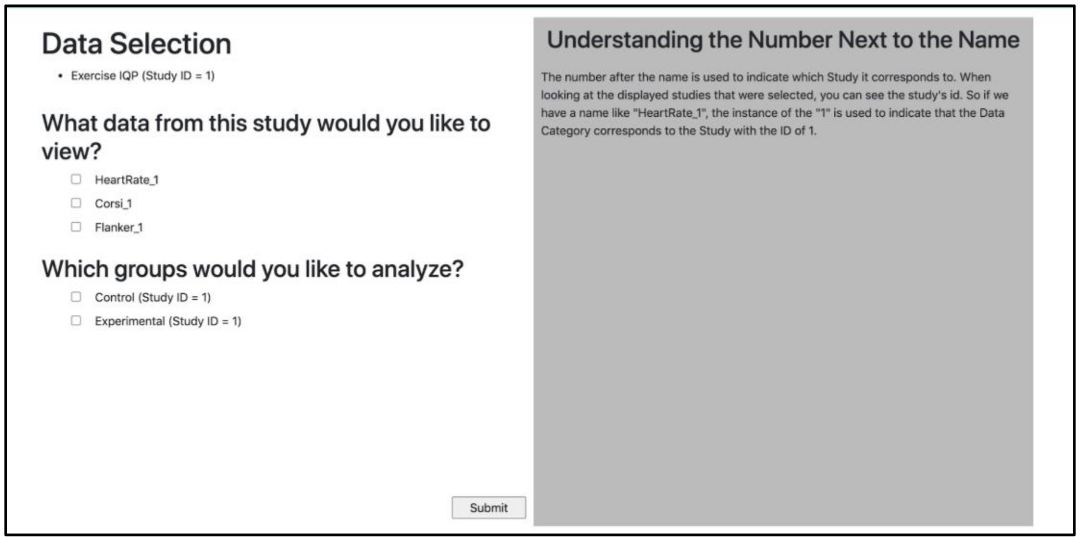
Click here to view the Help Page

Submit

Figure 33. Selection process Step 1: Select a study.

Once the user selects a study, GILBERT performs a query to get the data categories and study groups for the selected study. The available data categories and study groups appear on the next page of the selection process, as shown in *Figure 34*. Selecting a data category or categories allows the user to choose what type of data they would like to view from the study. Selecting a study group or groups lets the user control what group's data the user would like to see.

Because these data categories correspond to tables in our database, the selected data categories are saved to be used in the FROM clause in the final query. When multiple data categories are selected, GILBERT constructs a JOIN statement that joins the data from all the data category tables on subject ID.



Data Selection

- Exercise IQP (Study ID = 1)

What data from this study would you like to view?

- HeartRate_1
- Corsi_1
- Flanker_1

Which groups would you like to analyze?

- Control (Study ID = 1)
- Experimental (Study ID = 1)

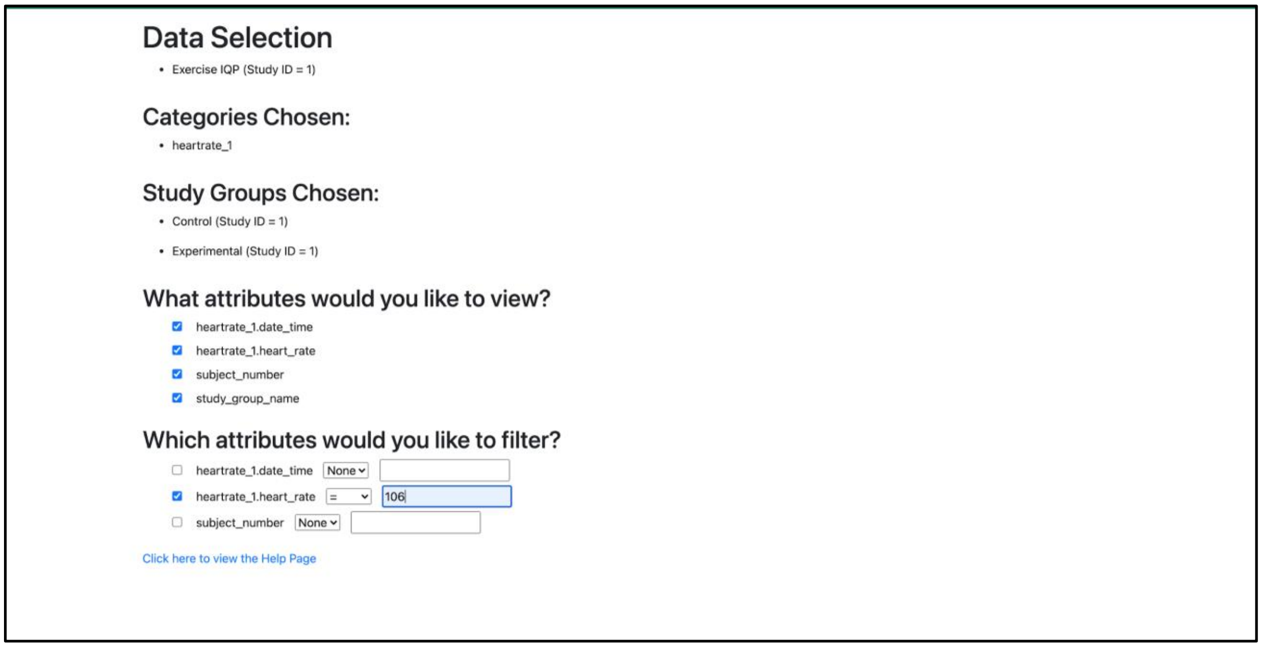
Submit

Understanding the Number Next to the Name

The number after the name is used to indicate which Study it corresponds to. When looking at the displayed studies that were selected, you can see the study's id. So if we have a name like "HeartRate_1", the instance of the "1" is used to indicate that the Data Category corresponds to the Study with the ID of 1.

Figure 34. Selection process Step 2: Selecting data categories and study groups.

After the data categories and study groups have been chosen, the user can select the specific attributes they would like to view, as well as how they would like to filter those attributes. For example, a data category “Heart Rate” may have attributes such as “datetime” and “heart rate” (in beats per minute). A user who wants to see all subjects’ heart rate data above 106 bpm and the time at which the measurement was taken would select the attributes “datetime” and “heart rate”, and then filter the heart rate to above 106. An example of this is shown in *Figure 35*.


Data Selection
- Exercise IQP (Study ID = 1)

Categories Chosen:
- heartrate_1

Study Groups Chosen:
- Control (Study ID = 1)
- Experimental (Study ID = 1)

What attributes would you like to view?
- [✓] heartrate_1.date_time
- [✓] heartrate_1.heart_rate
- [✓] subject_number
- [✓] study_group_name

Which attributes would you like to filter?
- [ ] heartrate_1.date_time None
- [✓] heartrate_1.heart_rate = 106
- [ ] subject_number None

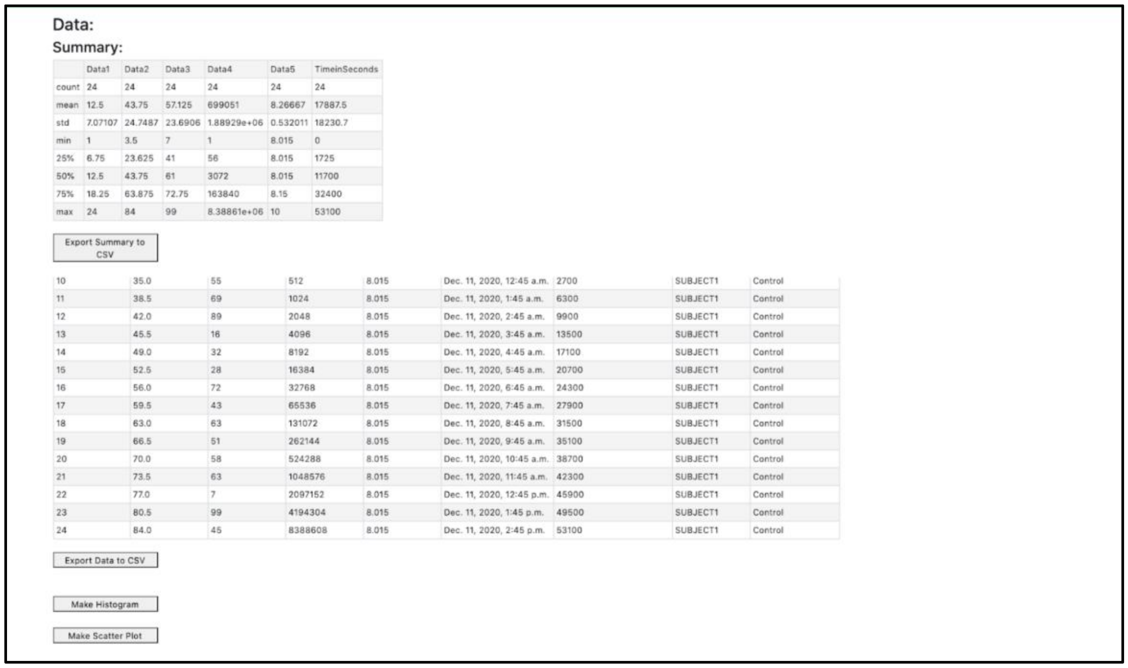
Figure 35. Selection process Step 3: Selecting and filtering attributes.

The attributes that are selected for viewing are used in the SELECT clause of the final query. Any filtering specified by the user is added to the WHERE clause along with the selected study groups. After this step, all the information for the final query has been collected. The query is then built and executed.

The requested data is displayed to the user in a table, shown in *Figure 36*, and can be exported as a .csv file. GILBERT also gives a statistical summary of the selected data, which can also be exported as a .csv file separately from the data it is contrived from. This summary is calculated using a built-in function in Pandas that reads selected data into a DataFrame and calculates the mean, quartiles, and other statistically significant information for each variable in the selected data.

4.6.2 Methods of Analysis

The selection output screen contains two buttons labelled “Make Histogram” and “Make Scatter Plot,” located below the displayed data summary and requested data, as shown in *Figure 36*. Selecting either button will lead to a prompt to create visualizations based on the current data.



Data:

Summary:

	Data1	Data2	Data3	Data4	Data5	TimeinSeconds
count	24	24	24	24	24	24
mean	12.5	43.75	57.125	699051	8.26667	17887.5
std	7.07107	24.7487	23.6906	1.88929e+06	0.532011	18230.7
min	1	3.5	7	1	8.015	0
25%	6.75	23.625	41	56	8.015	1725
50%	12.5	43.75	61	3072	8.015	11700
75%	18.25	63.875	72.75	163840	8.15	32400
max	24	84	99	8.38861e+06	10	53100

10	35.0	55	512	8.015	Dec. 11, 2020, 12:45 a.m.	2700	SUBJECT1	Control
11	38.5	69	1024	8.015	Dec. 11, 2020, 1:45 a.m.	6300	SUBJECT1	Control
12	42.0	89	2048	8.015	Dec. 11, 2020, 2:45 a.m.	9900	SUBJECT1	Control
13	45.5	16	4096	8.015	Dec. 11, 2020, 3:45 a.m.	13500	SUBJECT1	Control
14	49.0	32	8192	8.015	Dec. 11, 2020, 4:45 a.m.	17100	SUBJECT1	Control
15	52.5	28	16384	8.015	Dec. 11, 2020, 5:45 a.m.	20700	SUBJECT1	Control
16	56.0	72	32768	8.015	Dec. 11, 2020, 6:45 a.m.	24300	SUBJECT1	Control
17	59.5	43	65536	8.015	Dec. 11, 2020, 7:45 a.m.	27900	SUBJECT1	Control
18	63.0	63	131072	8.015	Dec. 11, 2020, 8:45 a.m.	31500	SUBJECT1	Control
19	66.5	51	262144	8.015	Dec. 11, 2020, 9:45 a.m.	35100	SUBJECT1	Control
20	70.0	58	524288	8.015	Dec. 11, 2020, 10:45 a.m.	38700	SUBJECT1	Control
21	73.5	63	1048576	8.015	Dec. 11, 2020, 11:45 a.m.	42300	SUBJECT1	Control
22	77.0	7	2097152	8.015	Dec. 11, 2020, 12:45 p.m.	45900	SUBJECT1	Control
23	80.5	99	4194304	8.015	Dec. 11, 2020, 1:45 p.m.	49500	SUBJECT1	Control
24	84.0	45	8388608	8.015	Dec. 11, 2020, 2:45 p.m.	53100	SUBJECT1	Control

Figure 36. Buttons for creating data visualizations.

Selecting the “Make Histogram” button prompts the user to make two choices. The first, an integer input, allows the user to choose the number of bins the data is partitioned by. The next prompt is a radio button prompt of all selected attributes. This determines which attribute data is used to make the histogram. When these questions are completed and the user hits “Submit”, GILBERT uses the Matplotlib Python package to create the histogram. This figure is displayed in the output page as shown in *Figure 37*. The title is automatically filled with “Distribution of [Selected Variable]”. The y-axis is labeled as “Frequency” and the x-axis is labeled with the selected variable.

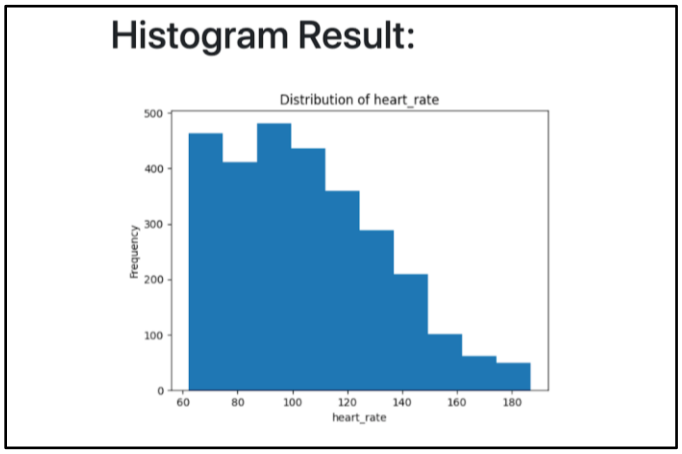
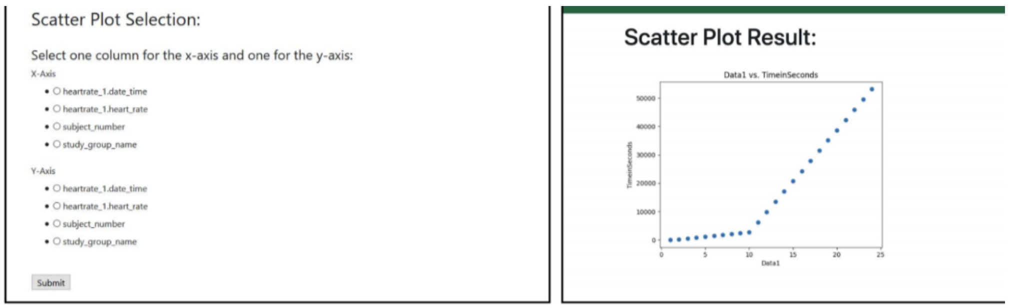


Figure 37. Histogram selection prompt and figure output.

The process for creating a scatter plot is similar to the process of creating a histogram. The scatter plot prompts require the user to choose two attributes from two radio button lists. These selected attributes become the respective x and y axes of the resulting figure. The title of the figure is automatically set to be “[X-Axis Variable] v. [Y-Axis Variable]”. Upon submission, the resulting Matplotlib figure is displayed as a PNG file. The prompt and the generated figure for the scatter plot feature are seen below in *Figure 38*. These features offer a streamlined, straightforward approach to plotting data comparable to programs like Excel and MATLAB.


Scatter Plot Selection:

Select one column for the x-axis and one for the y-axis:

X-Axis

- ○ heartrate_1.date_time
- ○ heartrate_1.heart_rate
- ○ subject_number
- ○ study_group_name

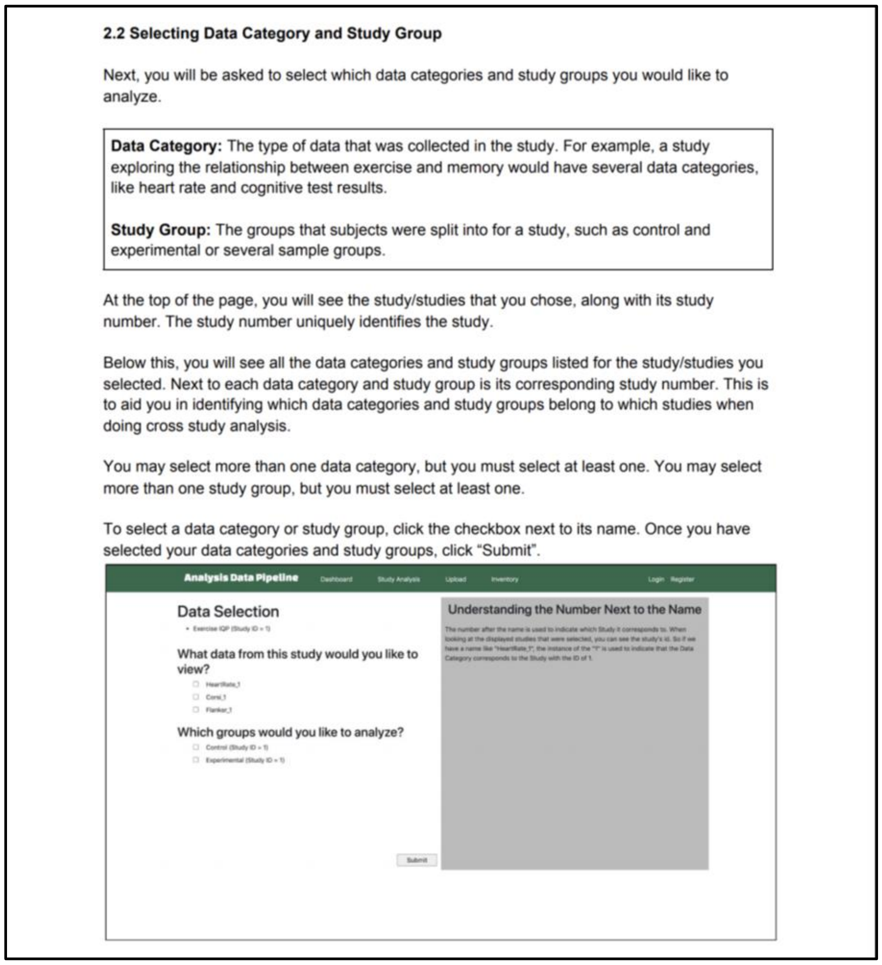
Y-Axis

- ○ heartrate_1.date_time
- ○ heartrate_1.heart_rate
- ○ subject_number
- ○ study_group_name

Figure 38. Scatter plot selection prompt (left) and figure output (right).

4.7 Help Manual

Although we designed GILBERT to be easy to use, there will be instances in which a user does not know how to use the application. For these situations, we created a help manual - a document embedded into our application that includes helpful information through every step of the pipeline. This includes information on data input standards for the uploader, a walkthrough of how to use each of GILBERT’s features, and definitions for terminology used throughout the application. An example of the information in our help manual can be seen in *Figure 39*. The entire help manual can be found in Appendix J.



2.2 Selecting Data Category and Study Group

Next, you will be asked to select which data categories and study groups you would like to analyze.

Data Category: The type of data that was collected in the study. For example, a study exploring the relationship between exercise and memory would have several data categories, like heart rate and cognitive test results.

Study Group: The groups that subjects were split into for a study, such as control and experimental or several sample groups.

At the top of the page, you will see the study/studies that you chose, along with its study number. The study number uniquely identifies the study.

Below this, you will see all the data categories and study groups listed for the study/studies you selected. Next to each data category and study group is its corresponding study number. This is to aid you in identifying which data categories and study groups belong to which studies when doing cross study analysis.

You may select more than one data category, but you must select at least one. You may select more than one study group, but you must select at least one.

To select a data category or study group, click the checkbox next to its name. Once you have selected your data categories and study groups, click "Submit".

Data Selection

- Exercise IQP (Study ID = 1)

What data from this study would you like to view?

- ☐ HeartRate_1
- ☐ Corsi_1
- ☐ Flanker_1

Which groups would you like to analyze?

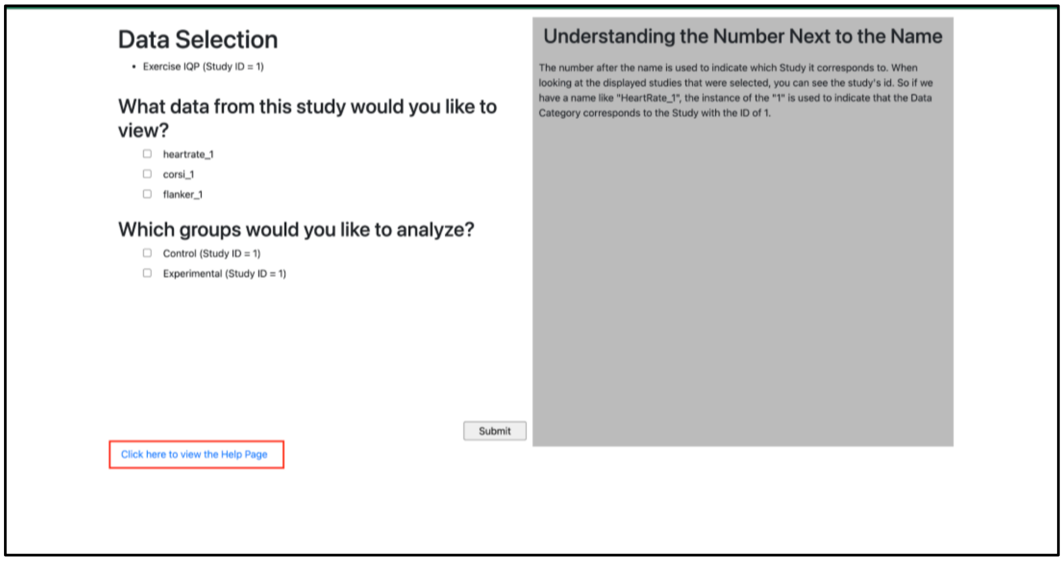
- ☐ Control (Study ID = 1)
- ☐ Experimental (Study ID = 1)

Understanding the Number Next to the Name

The number after the name is used to indicate which Study it corresponds to. When looking at the displayed studies that were selected, you can see the study's id. So if we have a name like "HeartRate_1", the instance of the "1" is used to indicate that the Data Category corresponds to the Study with the ID of 1.

Figure 39. A page in our help manual.

The help manual is accessible through a link located on every page of our application, as shown in *Figure 40*. Depending on what page of the application the user is on, the link will take them to the corresponding section of the help manual. This allows the user to quickly find information about the stage in the application they are on rather than sifting through the help manual. However, if a user would like to view a different section of the help manual, they have the ability to scroll through the entire document.


Data Selection

- Exercise IQP (Study ID = 1)

What data from this study would you like to view?

- ☐ heartrate_1
- ☐ corsi_1
- ☐ flanker_1

Which groups would you like to analyze?

- ☐ Control (Study ID = 1)
- ☐ Experimental (Study ID = 1)

Understanding the Number Next to the Name

The number after the name is used to indicate which Study it corresponds to. When looking at the displayed studies that were selected, you can see the study's id. So if we have a name like "HeartRate_1", the instance of the "1" is used to indicate that the Data Category corresponds to the Study with the ID of 1.

Figure 40. A link to our help manual on the data category/study group Selection page.

Alternatively, there is a table of contents which contains hyperlinks to every section of the help manual. The help manual also provides definitions to terminology specific to GILBERT such as “Data Category” or “Study Group” when they are relevant to a given feature.

Chapter 5. Evaluation

This chapter details the evaluation of GILBERT as a data analysis pipeline and presents results collected throughout the process, from both surveys and user studies. Overall, we aimed to assess GILBERT's effectiveness and ease of use, as well as how it compares to existing systems for data storage and analysis. We hoped to also establish a need for a tool like GILBERT. Data collected from the evaluation was used to outline changes that would be helpful to make in future iterations of this project.

5.1 Procedure

To assess GILBERT, our team decided to conduct a user study that simulates ways in which a user would use GILBERT. To minimize variation across evaluations, our team wrote scripts to guide participants through each of GILBERT's features. The script testing the user system walked through creating an account and logging in to GILBERT. The script testing the uploader walked through uploading a new study to GILBERT's database. The script testing the inventory walked through finding a study and identifying some of its metadata. The script testing the analysis component walked through creating a scatter plot out of data previously uploaded to the system. Throughout, we encouraged participants to view the embedded help manual if they have trouble completing their tasks rather than ask our team questions. We also wrote an introduction script to explain our project to participants and a closing script to thank them for their participation and encourage verbal feedback. These scripts can all be found in Appendix K. From this point on, we will refer to each script as shown in *Table 1.*

Script Name	**Feature Evaluated**
Script A	User System
Script B	Uploader
Script C	Inventory
Script D	Analysis

Table 1. Script names used in this paper and what they evaluate.

To encourage participation, we decided to keep our evaluation only 30 minutes in length. Because of this, we split our participants into two groups. Group 1 used Scripts A, C, and D while Group 2 used Scripts A and B. Both groups tested the user system. Due to the COVID-19 pandemic, our team conducted our evaluations virtually. We chose to use Zoom as our platform to hold our evaluation because it is a software familiar to many students. Also, Zoom allows its users to share their desktop screen and give another person on the call control of their screen. Our team used this remote-control functionality to ensure that participants do not have access to GILBERT after the evaluation and to speed up the set-up process for each evaluation.

Before contacting potential participants, our team made an interest survey in Qualtrics. While this survey collected basic information such as their major, education level, and email address, we also asked the participant for information regarding their background in handling and analyzing data. This allowed us to understand both the variety of backgrounds of our participants and how it may affect their performance on the evaluation. These survey questions can be found in Appendix L. To incentivize participation, our team gave each person who completed the study an entry into a raffle for a $60.00 Amazon gift card. We advertised this raffle both in the interest survey and email. Survey respondents needed to be over the age of 18 and specify that they were interested in being a user study participant in order to participate in the evaluation. All potential participants signed an IRB approved consent form prior to any evaluation involvement to fully understand the study.

Once our interest survey was ready to distribute, we chose groups to target. As a team of biomedical engineering and computer science majors, we believed that biomedical engineering and computer science students would be the target audience for our application. Also, since GILBERT relies on the consumption and analysis of data, our team chose to target data science majors as well. We sent our interest email to the following aliases on the WPI campus:

- Biomedical Engineering Major Qualifying Projects
- Biomedical Engineering Undergraduate Students
- Computer Science Undergraduate Students
- Data Science Undergraduate Students
- Data Science Graduate Students

As we received results, we randomly assigned respondents to a pair of members on our team. We sent each respondent the schedule corresponding to their pair and asked them to select a time slot. Then, on the day of their scheduled evaluation, a team member sent the participant a link to the Zoom meeting in which their pair of team members would be hosting the evaluation. Each pair alternated between hosting a Group 1 evaluation and a Group 2 evaluation, allowing a relatively even number of participants to assess the different parts of the system.

In each pair, one team member acted as the facilitator and the other as the note taker. The facilitator ran GILBERT locally on their machine and gave Zoom remote control to the participant. The facilitator also read each script to the participant as they completed the evaluation. The scripts that outlined the necessary tasks were kept open on a tab separate from GILBERT so the participant could reference the instructions while the evaluation was in progress. The note taker scored the evaluation, timed each task that the participant completed, and took other notes on the participant's behavior and feedback.

Figure 42 shows part of the spreadsheet that the note taker used during the evaluation. The note taker wrote down each participant's group number, the pair that did the evaluation, the participant's name, and the tasks they completed. They also wrote down the time taken to complete each task, the number of times the help manual was referenced, the number of times the participant verbally asked for help, and if they completed the task. The final score was calculated using the equation shown in *Figure 41*. Our team used these scores to summarize a participant's behavior and understanding of GILBERT. We weighted verbal questions more heavily than help manual references because we encouraged participants to seek answers on their own rather than asking a team member for help. Also, we included the amount of time to complete a task to gauge how quickly the participant understands GILBERT. Failing to complete a task was weighted the most heavily since this demonstrates the least amount of understanding. A lower score indicates that the participant had less difficulty with the system.

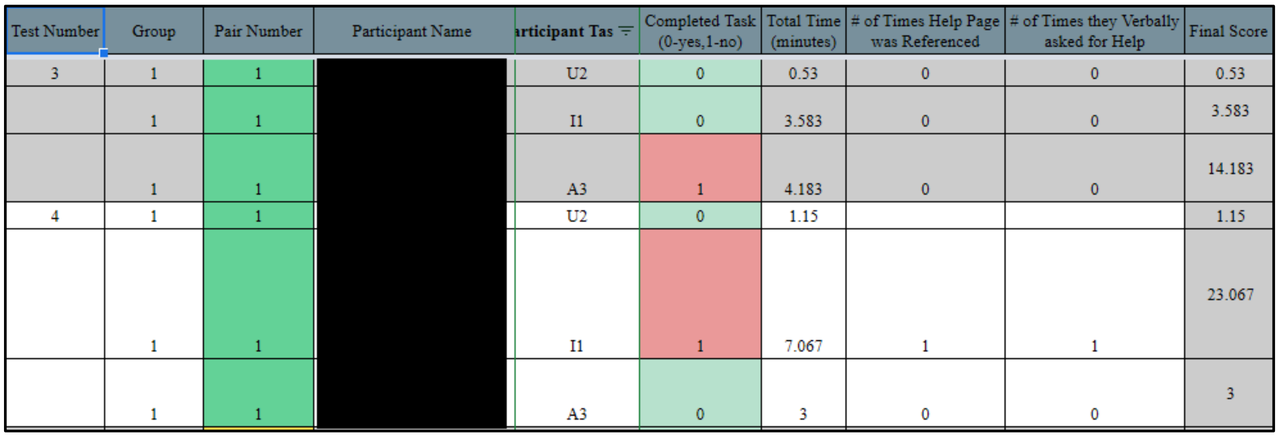
$$\mathit{Final\ Score} = t + 2h + 4q + 10x$$

t = time taken to complete task (minutes) h = number of help manual references

q = number of verbal questions x = 1 if task was incomplete, 0 if task was complete

Figure 41. Equation used to calculate final score in evaluation.

The note taker also wrote notes about the participant's behavior and verbal feedback in the spreadsheet in a column not shown in *Figure 42*.



Test Number	Group	Pair Number	Participant Name	articipant Tas	Completed Task (0-yes,1-no)	Total Time (minutes)	# of Times Help Page was Referenced	# of Times they Verbally asked for Help	Final Score
3	1	1	[redacted]	U2	0	0.53	0	0	0.53
	1	1	[redacted]	I1	0	3.583	0	0	3.583
	1	1	[redacted]	A3	1	4.183	0	0	14.183
4	1	1	[redacted]	U2	0	1.15			1.15
	1	1	[redacted]	I1	1	7.067	1	1	23.067
	1	1	[redacted]	A3	0	3	0	0	3

Figure 42. Screenshot of the sheet used to score participants. Names removed for privacy.

At the end of each evaluation, we thanked the user for participating and sent them a follow-up survey to complete. This survey assesses the effectiveness of GILBERT and compares it to other systems. These questions are available in Appendix M. The user left the Zoom meeting prior to completing the survey to avoid any influence from our team.

5.2 Results

This section presents the results collected through the interest survey, the evaluation of GILBERT, and the follow-up survey administered to any evaluation participants.

5.2.1 Characterizing Study Participants

In total, 113 students from Worcester Polytechnic Institute responded to our User Study Interest Survey (Appendix K). Of the respondents who specified their academic level, 64 were undergraduate students, 4 were graduate students, and the remainder did not respond to the question. The distribution of respondents who provided their major can be seen below in *Figure 43(a).* Of the 113 responses, 27 students participated in an evaluation of GILBERT. Of the participants in our evaluation, 25 were undergraduate students and 2 were graduate students. The distribution of their majors is displayed below in *Figure 43(b).*

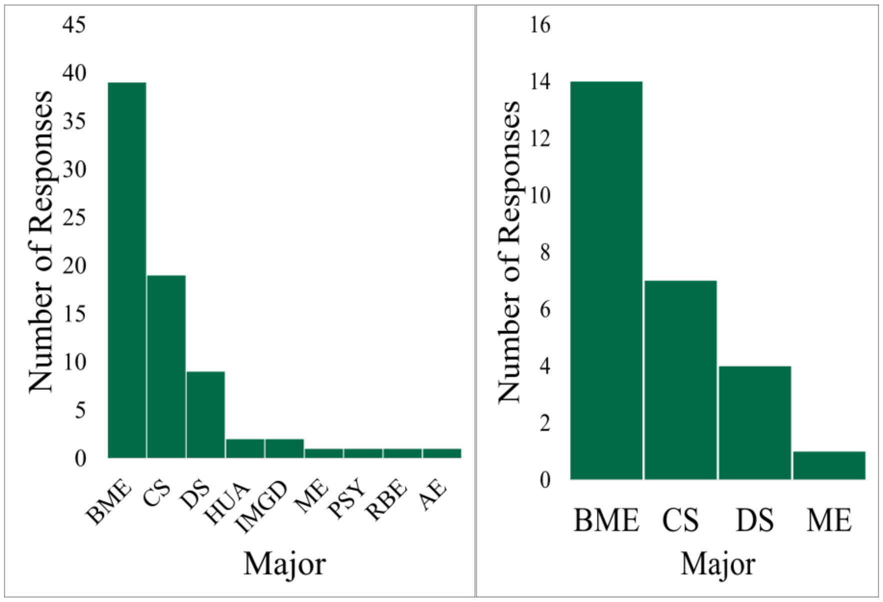


Figure 43 (a) - Majors of respondents *Figure 43 (b) - Majors of participants*

Figure 43. Majors of interest survey respondents: (a) All respondents in interest survey (n=113), (b): Evaluation participants (n=27).

From the survey, we found that most students have worked with large amounts of raw data (*Figure 44a*), relying mostly on graphical methods for analysis (*Figure 44b*) and spreadsheets for data storage and organization (*Figure 44c*).

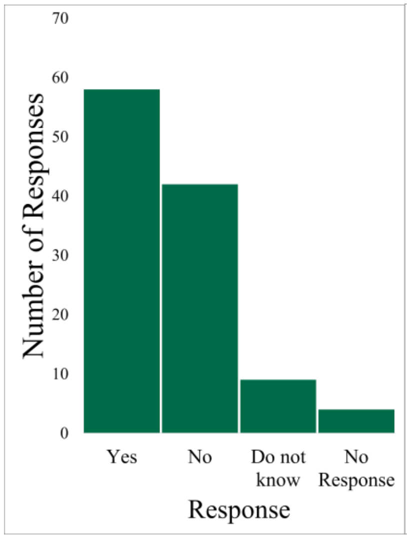


Figure 44 (a)

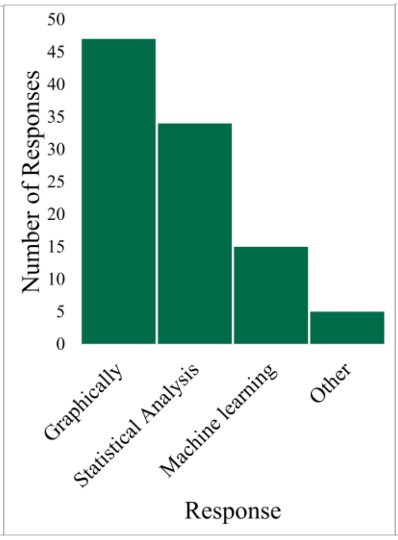


Figure 44 (b)

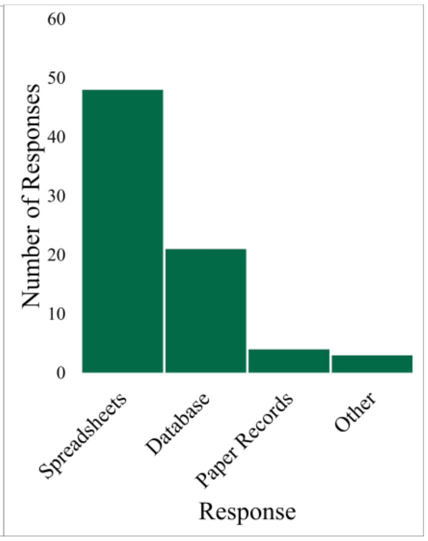


Figure 44 (c)

Figure 44. Interest survey responses for all respondents: (a) - Have you worked with large amounts of raw data? (b) - How do you analyze large amounts of data (c) - How do you store/organize large amounts of data? (n=113).

Much like the whole group of respondents, most of our evaluation participants have worked with large amounts of raw data (*Figure 45a*) which they have analyzed primarily through graphical and statistical analyses (*Figure 45b*) and have stored mostly through spreadsheets (*Figure 45c*).

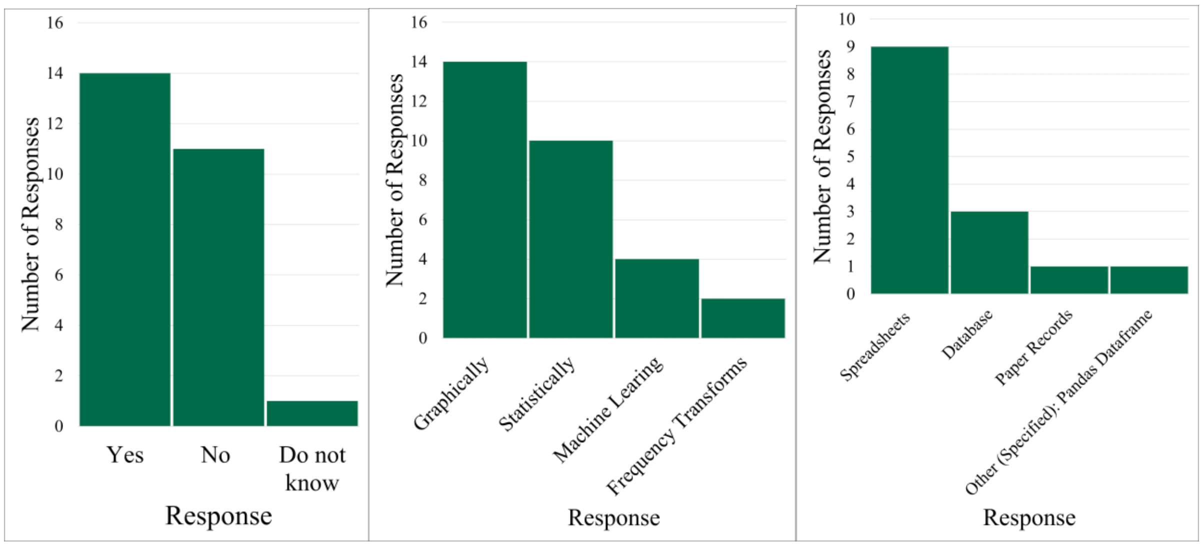


Figure 45(a) *Figure 45 (b)* *Figure 45 (c)*

Figure 45. Interest survey responses for evaluation participants: (a) - Have you worked with large amounts of raw data? (b) - How do you analyze large amounts of data (c) - How do you store/organize large amounts of data? (n=27).

Also, we found that most survey respondents have experience with specific tools whose key features include data storage, various analytical methods, and data visualizations controlled through programming or spreadsheet operations. Specifically, nearly all participants cited the use of either Microsoft Excel, Python, or MATLAB as frequented tools (*Figure 46a*). Lastly, when asked where they had used large amounts of raw data, courses at WPI were the most common setting, as seen in *Figure 46b*. Respondents were also given the option to elaborate on specific courses, naming DS501, BME 1004, BME 2211, and BME 3505 most often.

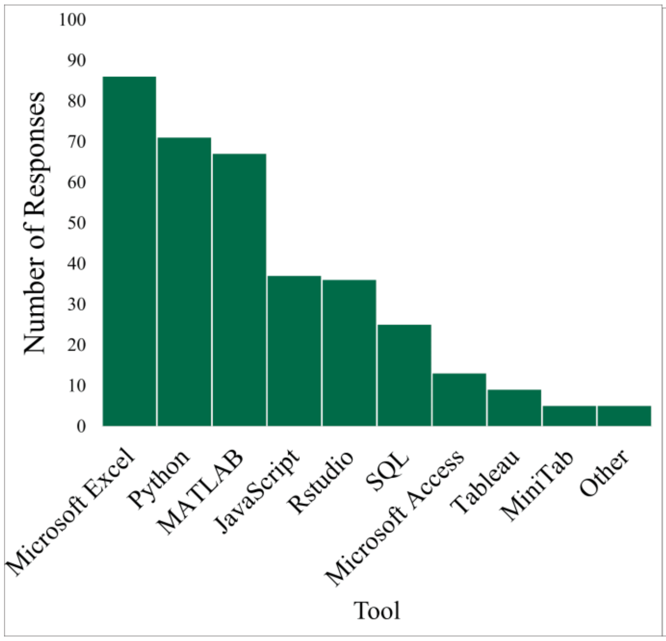


Figure 46(a)

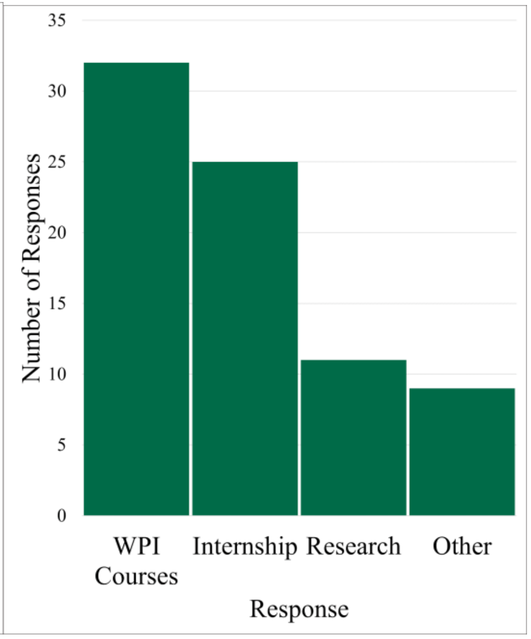
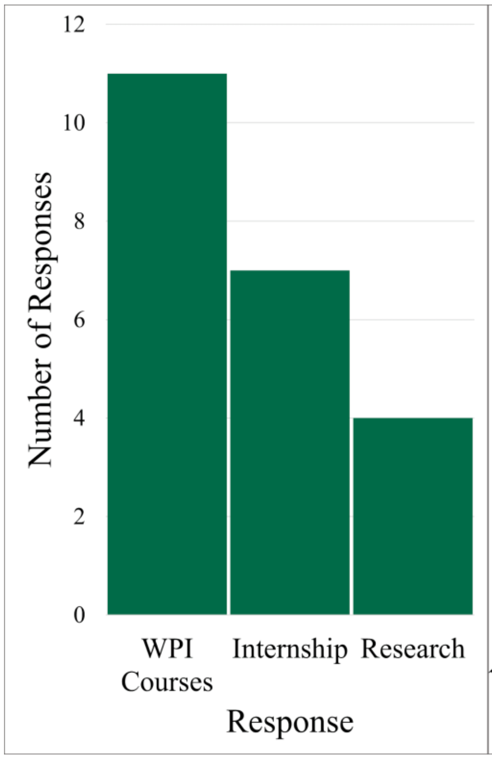


Figure 46(b)

Figure 46. Interest Survey Responses: (a) What tools do you have experience with? (b) In what setting have you used large amounts of data? (n=113).

Much like the large group of respondents, most of our evaluation participants have worked with large amounts of raw data which they have analyzed primarily through graphical and statistical analyses and have stored majorly through spreadsheets. Further, when asked which tools they had previous experience with, Microsoft Excel, MATLAB, and Python were again the most frequented tools (*Figure 47a*). Finally, when asked the setting in which they used large amounts of data, most evaluation participants attributed their experience to WPI Courses, again specifying DS501, BME 2211, and BME 1004 most often (*Figure 47b*).



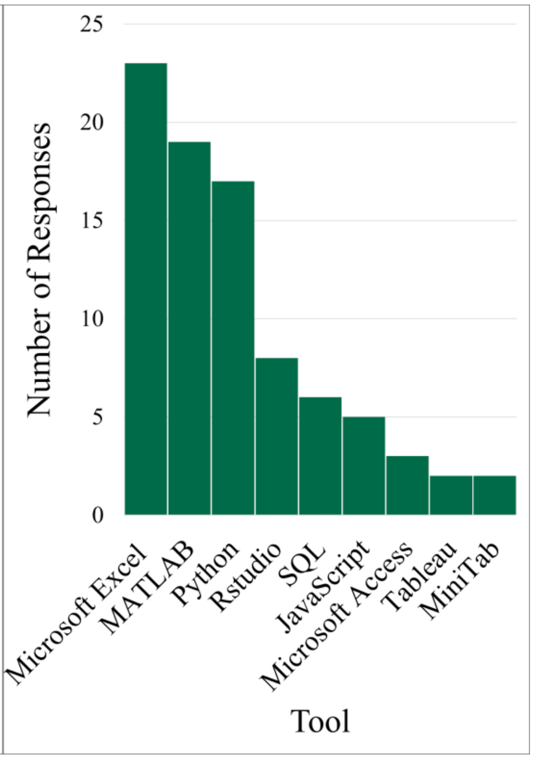
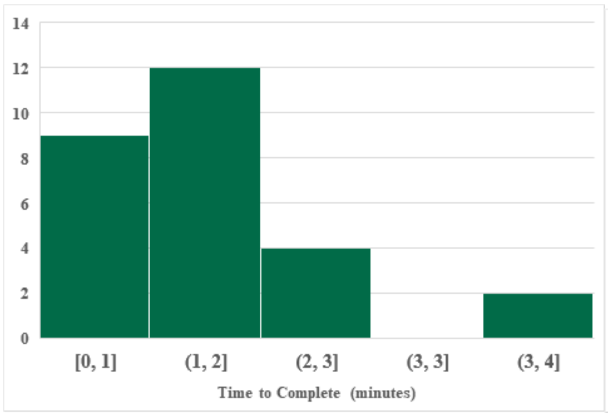


Figure 47(a) *Figure 47(b)*

Figure 47. Interest Survey Responses (Evaluation Participants): (a) In what setting have you used large amounts of data? (b) What tools do you have experience with? (n=27).

5.2.2 Evaluation Scoring and Notes

We collected several quantitative measures to determine each user's ease of interacting with GILBERT. These included if the task was completed, total time to complete the task, number of references to the help manual, and number of questions asked to the proctors. For each of the scripts listed in *Table 1* in Section 5.1, population distributions and measurements for completion, time consumption, and questions asked for each task can be seen in *Figures 48* through *50* below.



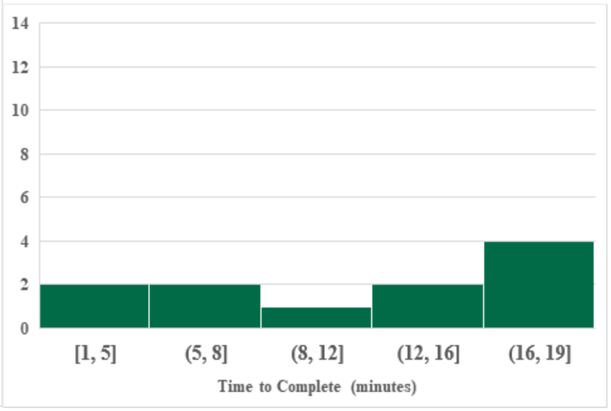
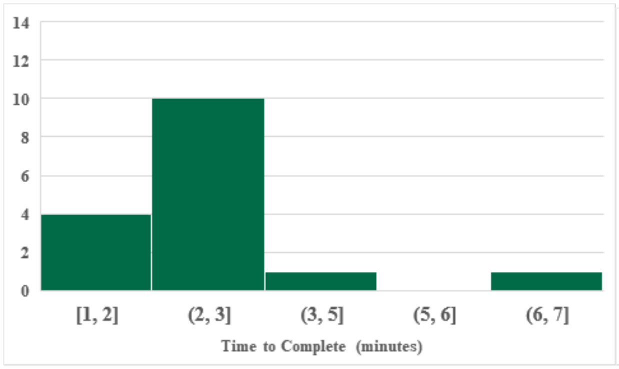


Figure 48 (a) - Time taken to register account

Figure 48 (b) - Time taken to upload study



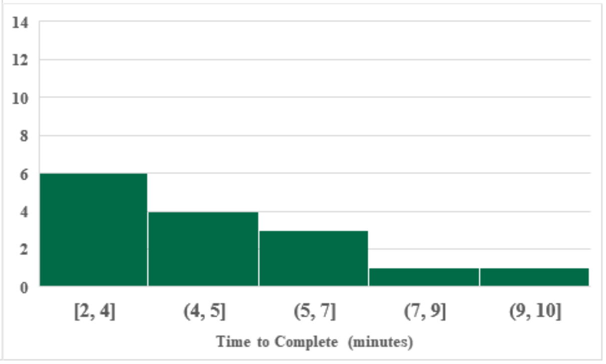
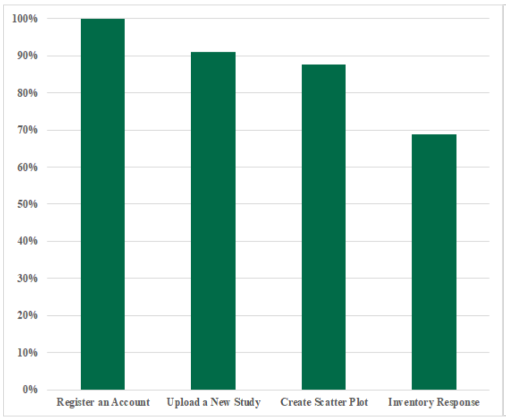


Figure 48 (c) - Time taken to answer inventory question

Figure 48 (d) - Time taken to create scatter plot

Figure 48. Recorded times for users to complete tasks. (n=27).



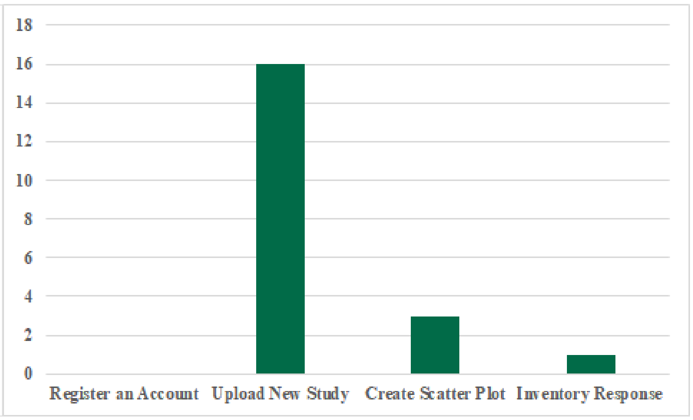


Figure 49 (a) - Completion percentage per task

Figure 49 (b) - Number of questions asked per task

Figure 49. Completion percentages and question counts for each task. (n = 27).

Script	A (Register an account)	B (Upload a new study)	C (Inventory questions)	D (Create scatter plot)
Time to Complete (minutes)	1:22	10:57	2:48	4:38

Table 2. Average time to complete evaluation tasks.

Using the scoring equation seen in *Figure 41* with the data shown in *Figures 48, 49,* and *Table 2* allowed us to determine the distributions of the final scores for each task. This measurement, plotted in *Figure 50*, represents how easily and self-sufficiently a user completed a task with GILBERT. The following box-and-whisker plot denotes the first through third quartiles of each scoring distribution in green, the maximum and minimum scores as whiskers, the horizontal line as the median score, the “x” as the mean score, and green dots as any outliers.

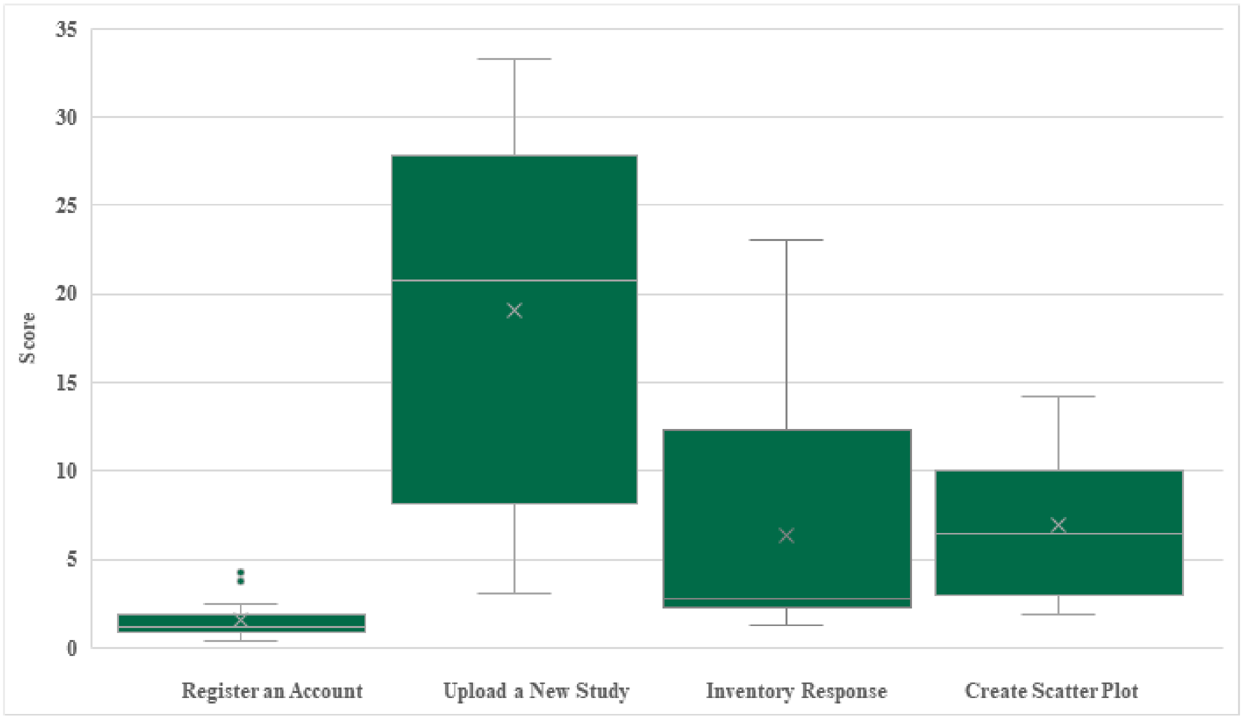
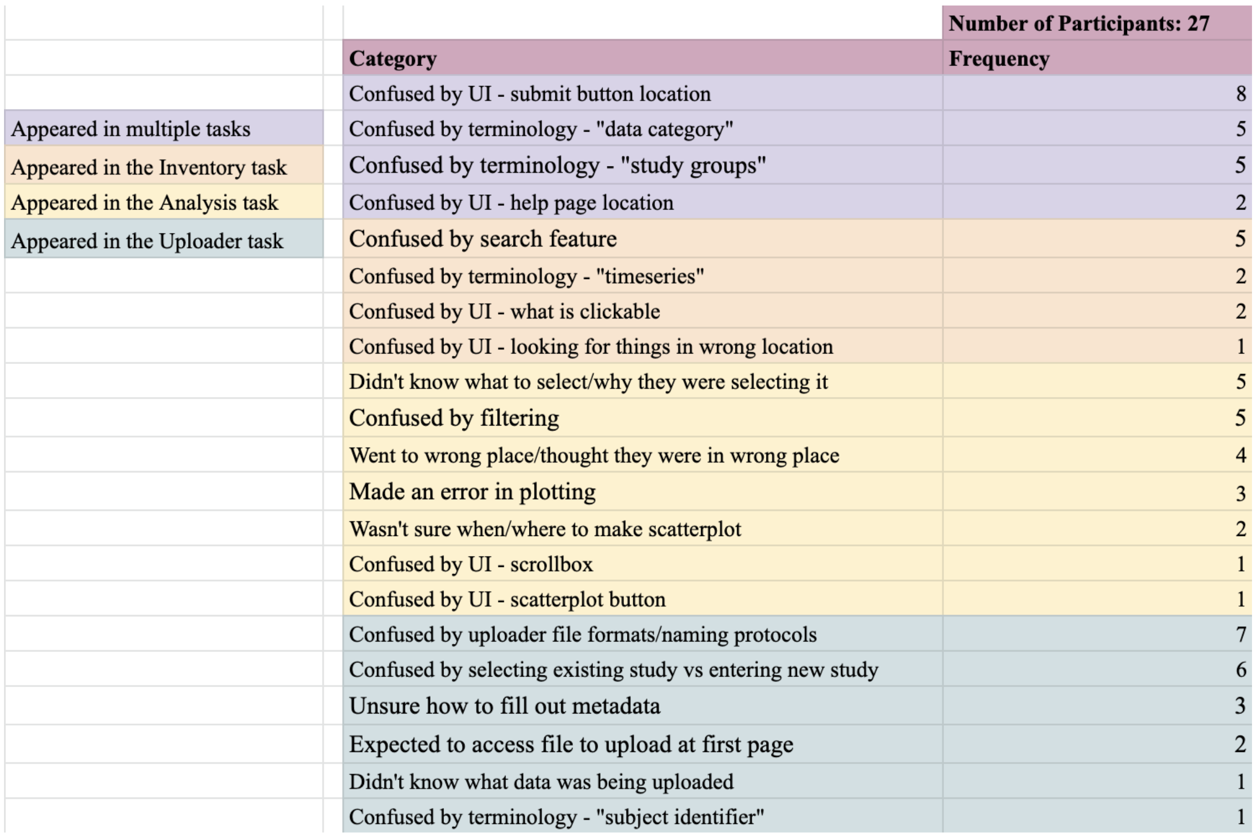


Figure 50. Score distributions by task. A lower score indicates better performance. (n=27).

Based on *Figure 50*, we concluded that users had the easiest time registering an account, as every recorded score for this task was below 5 points. The data also suggests users had a similar level of ease in completing the inventory response and creating the scatter plot, as the mean score and quartile spread of both tasks were similar. Lastly, it appears that users found uploading a new study to be the most difficult task, as this is where the most variance in score occurred.

Throughout each evaluation, the note taker recorded what the user did incorrectly, what they were confused by, what questions they asked, and other behaviors and interactions. These notes were analyzed and split into categories in order to see what stages and features users struggled with the most. These categories, along with the frequency of users who experienced each issue, are shown in *Figure 51*. The categories listed below were agreed upon by both of our team members who scored the evaluation notes.



Appeared in multiple tasks
Appeared in the Inventory task
Appeared in the Analysis task
Appeared in the Uploader task

	Number of Participants: 27
Category	**Frequency**
Confused by UI - submit button location	8
Confused by terminology - "data category"	5
Confused by terminology - "study groups"	5
Confused by UI - help page location	2
Confused by search feature	5
Confused by terminology - "timeseries"	2
Confused by UI - what is clickable	2
Confused by UI - looking for things in wrong location	1
Didn't know what to select/why they were selecting it	5
Confused by filtering	5
Went to wrong place/thought they were in wrong place	4
Made an error in plotting	3
Wasn't sure when/where to make scatterplot	2
Confused by UI - scrollbox	1
Confused by UI - scatterplot button	1
Confused by uploader file formats/naming protocols	7
Confused by selecting existing study vs entering new study	6
Unsure how to fill out metadata	3
Expected to access file to upload at first page	2
Didn't know what data was being uploaded	1
Confused by terminology - "subject identifier"	1

Figure 51. Issue categorizations and frequency during the evaluations.

This data highlights issues that users may have with GILBERT that our team may not have realized as developers of the system. We were particularly interested in seeing what specific issues users encountered the most, as well as which features users had the most trouble with.

Figure 52 shows the four issues users experienced the most during their evaluation. The type of issue experienced most frequently was related to GILBERT’s user interface. This includes issues such as not being able to find buttons, not knowing what can be clicked, looking for things in the wrong location, and confusion with scroll bars. The next biggest issue users experienced was not knowing certain terminology throughout the application. This included terms like data category, study groups, time series, and subject identifier. A significant number of users were also confused by the file formats and naming protocols required by the uploader. Finally, many users could not immediately see the difference between adding new data to an existing study and adding an entirely new study to GILBERT.

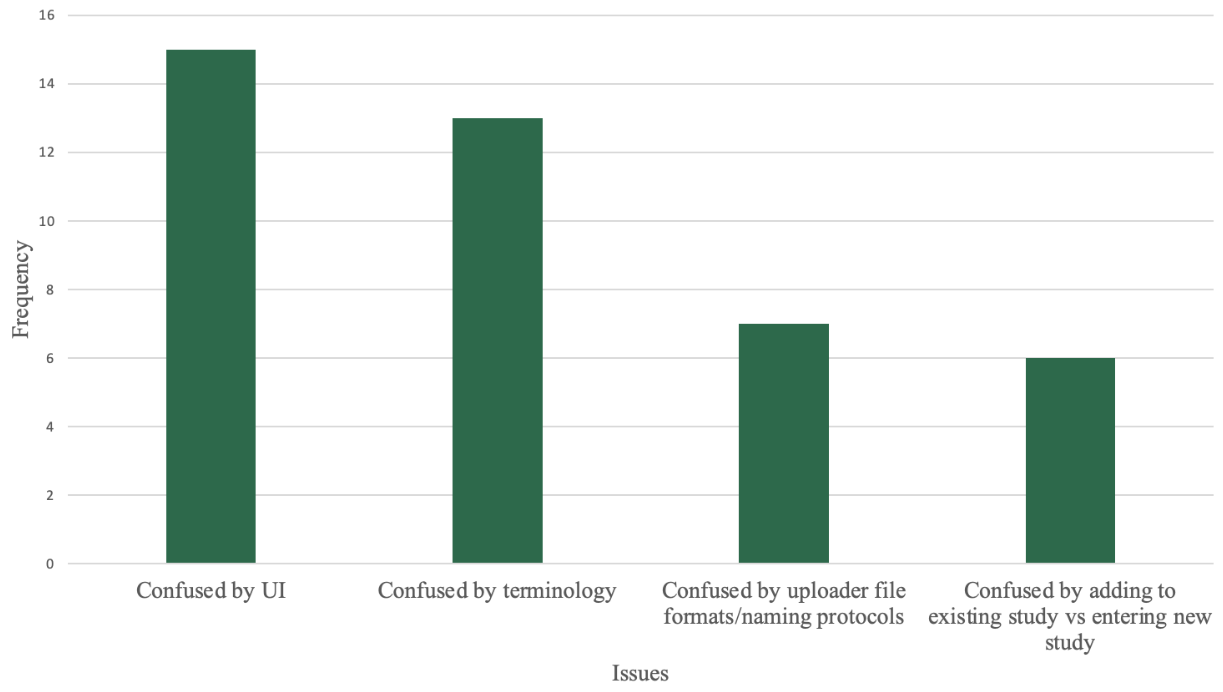


Figure 52. Frequency of most common issues experienced during evaluations (n=27).

Figure 53 shows the number of issues experienced in each feature of GILBERT. This shows which features users struggled with the most and identifies which areas of GILBERT require the most improvement. The analysis task had the most issues, followed by the uploader, then inventory, then the user system.

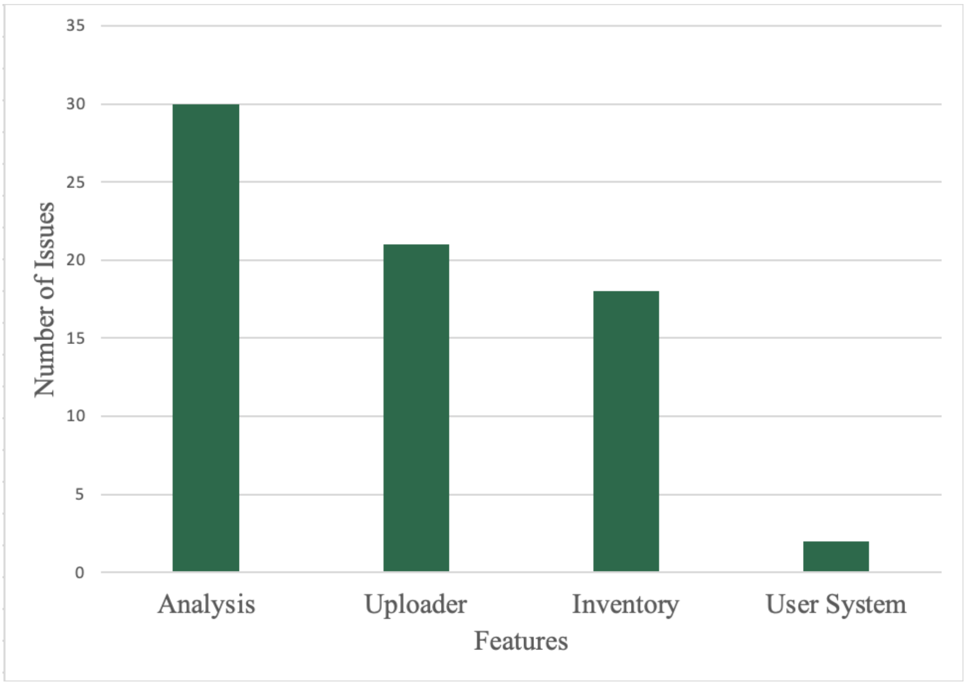


Figure 53. Number of issues experienced in each of the core features of GILBERT during the evaluation (n=27).

5.2.3 Post-Evaluation Survey

After each evaluation, we sent the participants a survey to assess their experience with GILBERT and how it compares to other applications. The survey questions can be found in Appendix M. Of the 27 participants, 26 of the participants responded.

Participants were asked to provide a ranking for five statements with the value of their ranking based on how strongly they agreed or disagreed with the statement. A "1" indicates "Strongly Disagree" and "5" indicates "Strongly Agree." The statements are displayed below in *Table 3*.

Statement Letter	***Statement***
A	The application was easy to use.
B	The application was easy to navigate.
C	The application would be useful in a research setting.
D	The application had a logical flow.
E	The application had an attractive design.

Table 3. Post-evaluation statements for rating.

The ratings on each of the statements were evaluated across different skills levels based on major and experience with different tools. Specifically, we were interested in whether different in-school experiences through courses or project work would directly relate to a person's ability to use GILBERT. For example, we hypothesized that Computer Science majors, who likely have the largest amount of exposure to different user interfaces, would find working with our system easiest. Similarly, evaluating whether exposure to large amounts of data and specific tools that related to various functionalities within our system would allow us to determine how universally user-friendly GILBERT is. *Figure 54* begins by comparing the five statements according to major.

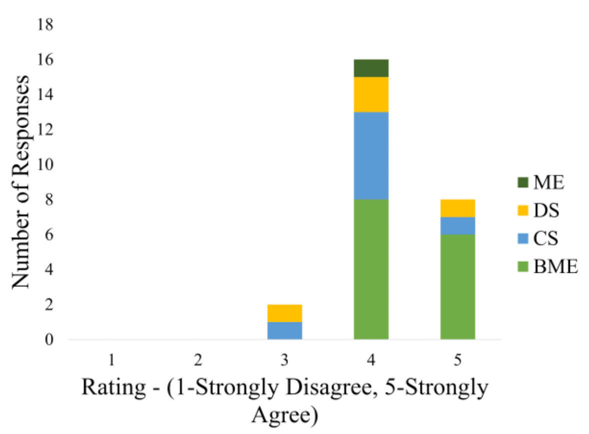


Figure 54 (a) - "Easy-to-Use" according to major

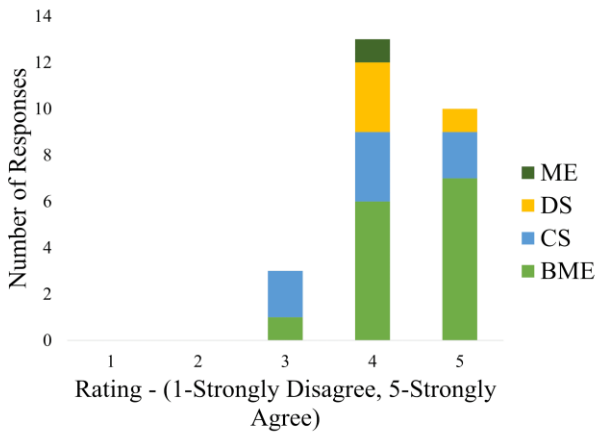


Figure 54 (b) - "Easy-to-Navigate" according to major

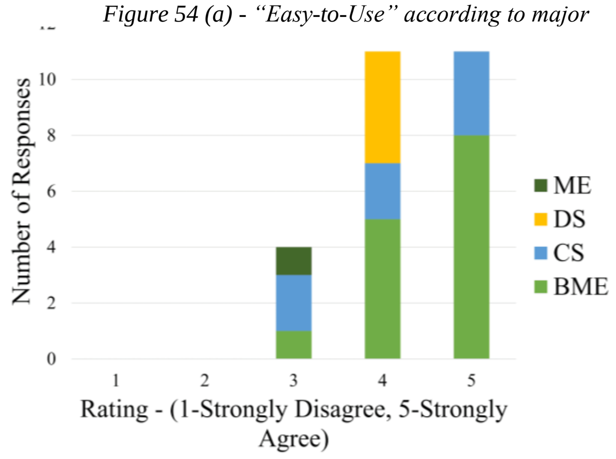


Figure 54 (c) - "Useful in Research" according to major

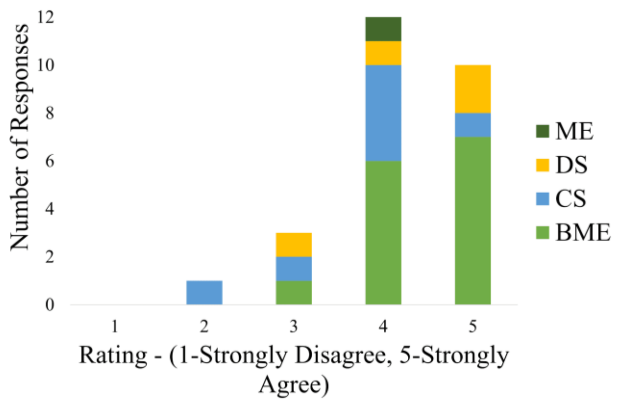


Figure 54 (d) - "Logical Flow" according to major

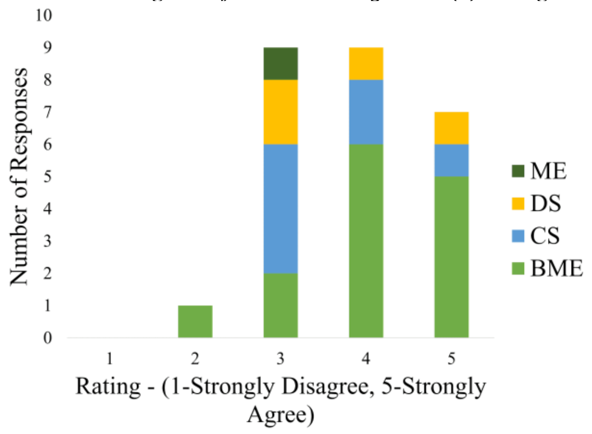


Figure 54 (e) - "Attractive Design" according to major

Figure 54. Ratings of post-evaluation statements according to major (n=26).

The data from each of the plots was analyzed to identify if there was a difference in ratings per statement that was dependent on each major. Due to the size of our evaluation sample (n < 30), a non-parametric test of significance was used to evaluate if there is a difference in median rating according to major for each statement. Specifically, a Kruskal - Wallis test of significance was employed and evaluated at a significance level of 0.05. We used MATLAB's built-in function to perform the computation. The degrees of freedom are equal to the number of groups, *k*, minus one and the number of groups is dependent on the collected responses (e.g., the different majors of respondents). The results of the tests of significance are displayed below in *Table 4(a)*. *Table 4(b)* displays the size of each group, *n*, the number of groups, *k*, and the degrees of freedom, *df*.

	Kruskal - Wallis (p-value)
Major and Statement A Rating	0.37
Major and Statement B Rating	0.57
Major and Statement C Rating	0.37
Major and Statement D Rating	0.31
Major and Statement E Rating	0.17

Table 4(a). P-values from Kruskal-Wallis test of significance according to major.

Group	***Group size (n)***	***Number of groups (k)***	***df***
BME	14	4	3
CS	7		
DS	4		
ME	1		

Table 4(b). Group sizes, number of groups, and degrees of freedom for Kruskal-Wallis test according to major.

Table 4. Kruskal-Wallis test of significance for statement ratings according to major.

Because the results of *Table 4* were evaluated at a significance level of 0.05, we cannot reject the null hypothesis and can therefore assume that there is no difference between the ratings according to different majors.

Responses to post-evaluation statements were then analyzed according to whether a respondent had previous experience with large data. *Figure 55* displays the ratings below according to previous data management experience.

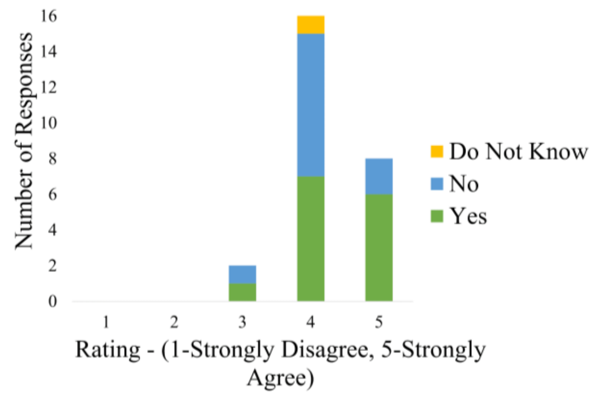


Figure 55 (a) - "Easy-to-Use" according to large data experience

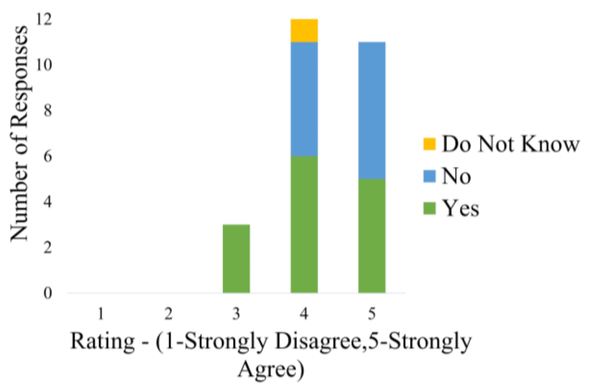


Figure 55 (b) - "Easy-to-Navigate" according to large data experience

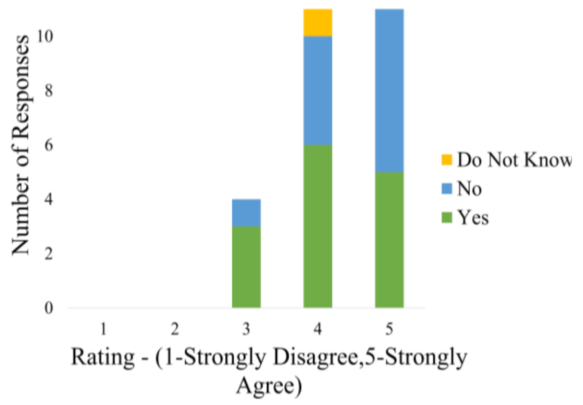


Figure 55 (c) - "Useful in Research" according to large data experience

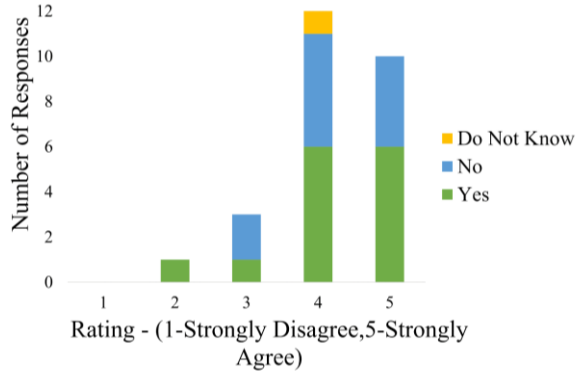


Figure 55 (d) - "Logical Flow" according to large data experience

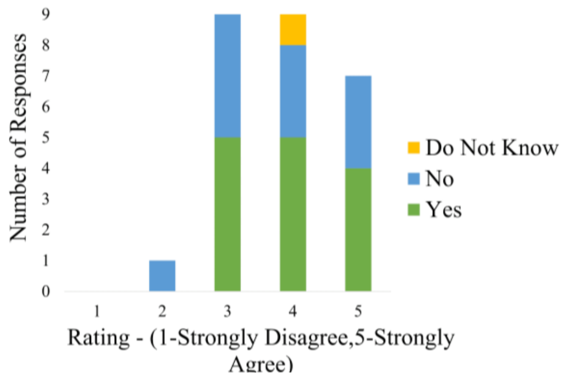


Figure 55 (e) - "Attractive Design" according to large data experience

Figure 55. Ratings of post-evaluation statements according to experience with large data. (n=26).

Again, a Kruskal - Wallis test of significance was used to evaluate whether previous experience with large amounts of data affected respondents' experience with GILBERT. The results of the Kruskal-Wallis test, grouped by experience with large amounts of data, is displayed in *Table 5*. *Table 5(a)* presents the p-values of the test and *Table 5(b)* presents the groups, their size, and the degrees of freedom.

	Kruskal-Wallis (p-value)
Large Data Experience and Statement A Rating	0.45
Large Data Experience and Statement B Rating	0.34
Large Data Experience and Statement C Rating	0.87
Large Data Experience and Statement D Rating	0.86
Large Data Experience and Statement E Rating	0.5

Table 5(a). P-values from Kruskal-Wallis test of significance according to experience with large data.

Group	***Group size (n)***	***Number of groups (k)***	***df***
Yes	14	3	2
No	11		
Do Not Know	1		

Table 5(b). Group sizes, number of groups, and degrees of freedom for Kruskal-Wallis test according to large data experience.

Table 5. Kruskal-Wallis test of significance for statement ratings according to experience with large data.

The results in *Table 5* show no significant p-values were determined, indicating that previous experience with large amounts of data did not affect the rating of the five statements presented in the post-evaluation survey.

To further analyze the effect of previous experience on interactions with GILBERT, tests of significance were again performed to evaluate whether previous experience with three key tools implemented in our system changed how a participant rated each of the five statements. The three key tools of interest were MATLAB, Python, and SQL. In each case, experience with each specific system was compared to a lack of experience with the same system. The test of significance evaluated whether each experience level influenced the rating of each of the five statements. *Table 6* presents the results of the tests of significance. *Table 6(a)* presents the p-values and *Table 6(b)* presents the groups and the size of each group for each of the different applications. For each of these Kruskal-Wallis tests the groups were “Yes” or “No,” therefore for each test the number of groups was 2 and the degrees of freedom were 1.

	MATLAB	Python	SQL
Ratings for Statement A	0.69	0.57	0.09
Ratings for Statement B	0.68	0.88	0.18
Ratings for Statement C	0.98	0.29	0.52
Ratings for Statement D	0.75	0.58	0.47
Ratings for Statement E	0.41	0.62	0.25

Table 6(a). P-values from Kruskal-Wallis test of significance according to experience with different tools.

Group	***Group size (n) (SQL)***	***Group size (n) (Python)***	***Group size (n) (MATLAB)***
Yes	6	17	19
No	20	9	7

Table 6(b). Groups and group sizes for Kruskal-Wallis tests according to experience with different tools.

Table 6. Kruskal-Wallis test of significance for statement ratings according to experience with various tools.

As with the previous tests of significance, there was no difference in rating GILBERT between those who were experienced with MATLAB, Python, or SQL and those who were not.

Then, we asked participants what they like and dislike about GILBERT in free-response questions. These questions were presented in addition to rating Statements A through E and were added to allow study participants to elaborate more specifically on their experience with GILBERT. A summary of the answers is listed below in *Figure 56*. The categories used were agreed upon between the two team members who scored the responses. Answers shown were stated by three or more participants, with the exception of *Figure 56b*, whose statements were said by two or more participants. Each question can be found under each image in *Figure 56*.

Comment	*Freq.*
Easy to use	9
Simple style/design	8
Easy to follow	7
Looks appealing	4
Study tasks were easy	3
Lots of helpful text (help page and other)	3
Scatter plot easy to make	3

(a) - What did you like about the application?

Comment	*Freq.*
Poor button placement/too small	7
Confusing process	6
Confused by attributes	3
Too many options/prompts	2
Filename restriction is a bad idea	2
More limited than other tools	2
Unclear names in navigation bar	2

(b) - What didn't you like about the application?

Comment	*Freq.*
Customizable	5
Lots of available help (online, peers)	5
Data organization is easy	5
Lots of features	4
Able to make graphs/easier	3
Gives you more control	3

(c) - What do you like about other data handling tools you have experience with (for example: Excel, Python, SQL, MATLAB)?

Comment	*Freq.*
Unintuitive	5
Too many features/overwhelming	5
Not knowing how to use	5
Unclear errors	3
Hard to get help	3
Hard to make/edit graph	3

(d) - What do you not like about other data handling tools that you have experience with, if you have any?

Figure 56. Most common phrases/ideas in post-evaluation survey responses.

Chapter 6. Discussion

In this chapter, we present the capabilities of GILBERT as a data management tool and how its usability varies across different backgrounds and experience levels. From evaluation results and feedback, we explain how GILBERT compares to other industry tools and outline recommendations for future additions and modifications to improve and expand on existing functionality.

6.1 Relevance of GILBERT

Although most respondents to our interest survey did not participate in our evaluation, all responses established a need for our application. Most respondents have previous experience with large amounts of raw data across a wide array of backgrounds. In total, we collected responses from nine majors at WPI and had students with four different majors participate in our evaluation of GILBERT. Similar to the wide range of collected majors, we found that those who had previous experience with large amounts of raw data reported over four primary methods of analysis and four methods of data storage. They also employed over ten different tools to help store, manage, and analyze data and found themselves in a multitude of settings where they were exposed to handling large amounts of data, with WPI Courses being the most frequent setting. From here, respondents provided courses from four different departments that exposed them to large amounts of data. Overall, both the wide range of settings that might expose a student to large amounts of data and the many different methods of storage, organization, and analysis utilized to manage it indicates the need for a tool that can accomplish storage, organization, and analysis in a single application.

In addition to the supporting evidence collected from our survey responses, we found through our background research that there is a strong motivation to build a data pipeline like GILBERT. The growth of data in recent decades has increased computational requirements and has prompted a rise in new methods of analysis but has also increased the need for a tool to handle such large amounts of data. Further, there has been an increasing desire to share data safely and perform cross-study analyses (Hrynaszkiewicz, 2010). GILBERT's ability to consolidate data storage, organization, and analysis into a single application makes it a potential system for accomplishing many data handling tasks in an efficient, organized manner. Future iterations of GILBERT, as described in Section 6.4, can expand upon current functionalities and improve analytical methods and cross-study analysis capabilities.

6.2 Effectiveness of GILBERT as an Application

GILBERT was created to streamline the biomedical data analysis process and provide a simple, intuitive workflow. As seen in the evaluation results, the completion rate for all of the tested features were above 68.8% and as high as 100%. In addition, the average time of completion between all tasks was 4 minutes and 56 seconds. A further breakdown of these average completion times can be seen in *Table 2* in Section 5.2.2. GILBERT also benefits from housing all its functionality within a single web application. This eliminates the need to switch between software that traditional workflows may require.

Another metric of importance was the number of questions asked by participants during each task. Of the 70 task assignments to participants, only 20 required assistance beyond the help page. It is important to note that the discrepancies that prompted these questions may have not all been a result of faults in the application; throughout the evaluation process, some confusion arose with the wording of scripts and the way in which the evaluation was proctored. Based on the resulting data, it can be inferred that most evaluated users were able to efficiently navigate and interact with the application.

In addition to the data collected during the evaluation regarding the completion of tasks, the responses to our post-evaluation survey provided us with insight into how user-friendly GILBERT is and how experiences using the system varied according to skill level. Specifically, we used the ratings of Statements A through E (*Table 3*) as quantifications for overall experience with GILBERT. When analyzing how the ratings on each of these statements varied according to different experience levels and backgrounds, we found no significant difference in experience with GILBERT.

The different experience levels and backgrounds that we looked into were major, experience with large amounts of raw data, and experience with three tools that were incorporated into our system. We hypothesized that students in the Computer Science department, participants with previous experience working with large amounts of raw data, and those who had used MATLAB, Python, and SQL would have an improved experience, seen through higher ratings, when using GILBERT. Although we hypothesized these different ratings according to different demographics, there was no difference in ratings on the statements presented in the post-evaluation survey according to any of the factors. Thus, the system was rated equally easy-to-use, easy-to-navigate, attractive in design, logical in flow, and useful in a research setting regardless of previous experience.

In the post-evaluation survey comments, nine participants stated that they found GILBERT easy to use, eight enjoyed its simple design, and seven stated that the application was easy to follow. Three participants specifically noted that the scatter plot was easy to make, which further suggests that our analysis tool is particularly easy to use. While not many participants who viewed the help manual, those who used it found it helpful. Also, four participants stated that GILBERT has an attractive design. These results are shown in *Figure 56a* in Section 5.2.3.

The most common dislike about GILBERT is the poor button placement. Seven participants stated that the buttons were too hard to find, too small, or both. Also, although seven participants said the application was easy to follow, six found the process confusing. Therefore, we may want to make our application easier to navigate. Some participants suggested adding a progress bar to show how long the process will take or showing the list of steps as the user is uploading data or doing analysis. The results to this question are shown in *Figure 56b* in Section 5.2.3.

6.3 Comparisons to other applications

According to our analysis of post-evaluation survey comments, participants found other data handling tools to be more customizable. They also liked how data organization and graph-making is easier in other tools. For example, Microsoft Excel allows for direct viewing and editing of the data due to all the data being stored in the spreadsheet. Some users commented that this is easier to handle than looking at the GILBERT's output and then selecting the columns used for a histogram or a scatter plot. However, some stated that making graphs is more difficult in other tools than in GILBERT. For example, making a graph in MATLAB requires knowledge of programming, while making a graph in GILBERT only requires clicking a few buttons. While this is less customizable than MATLAB, it quickly generates a graph that may have taken longer to create in MATLAB.

Another comment that came up frequently in the likes and dislikes of other data analysis applications is that other applications have more features than GILBERT does. Those who viewed this as a positive noted that this creates more flexibility in making charts and manipulating the data. For example, GILBERT can only make histograms and scatter plots, while other applications can make other charts such as box-and-whisker plots, line graphs, and more. Those who viewed the greater number of features as a negative stated that having more features made other applications more intimidating to use. Usually, each feature needs to be learned separately, so having many features can make a tool overwhelming to learn. On the other hand, GILBERT uses similar processes for uploading, selecting, and analyzing different sets of data, so users may feel less intimidated by it.

Also, some participants believe getting help while using other tools is easier. Many other data analysis tools have online resources that provide help to users. This can be through the application's help manual, online forums, or resources found via a search engine. On top of this, the other tools are more widely used among the participants' peers. Participants mention having the ability to ask a colleague questions about Excel or MATLAB as an advantage to using those systems rather than a system like GILBERT. However, some participants noted that it can be hard to get help on how to use other systems.

Specifically, it is difficult to troubleshoot coding errors in SQL, Python, or MATLAB. Three participants stated that they like the amount of helpful text that GILBERT provides, both in the help manual and directly on the web pages. While a user likely cannot seek help on using GILBERT through the internet or through their peers, GILBERT provides information to help users to offset this concern. Also, since users do not have to code to use GILBERT, users should have less difficulty troubleshooting GILBERT than SQL, Python, or MATLAB.

6.4 Future work

While our team built the foundation of the GILBERT data pipeline, there are still many ways that GILBERT could be improved or expanded upon in the future. The current state of the application is functional, but it has areas for improvement. Additionally, our team also determined new features that would improve the functionality of GILBERT but were unable to implement within our time of working on the project.

6.4.1 Database Design

The design of the database was a critical step in the development of GILBERT, with many features relying on the relationships between different tables. The design process was influenced by our goals for the system, the data we were testing with at the time, our technical experience, and short term features that GILBERT needed to support. Because of this, we recognize the potential for the database design to be improved or changed in the future due to various limitations.

One limitation is related to the scalability of the database. By using a relational database and relying on creating new tables for every data category, GILBERT may create an abundance of tables as different types of data are uploaded. To improve its efficiency, we recommend researching various solutions. This may include designing a more optimal relational database schema, incorporating a non-relational database, which would be less restricted by predefined schema, and experimenting with larger scale methods such as data lakes or data warehouses.

Another limitation is the concept of a data category itself in regard to cross-study analysis. The goal of this component was to group related data measurements together so that similarly formatted data from different studies could be compared and analyzed together. This would be facilitated through the use of the DataCategoryStudyXref table, which represents the many-to-many relationship between data categories and studies. For this to work, different studies have to contain data with the exact same column headers and type of data. This rigidity, along with implementation difficulties, restricted GILBERT's functionality to perform cross-study analyses. We recommend that this feature is expanded on and revised to be more flexible. This way, GILBERT could perform cross study analyses on studies with similar data that may have slightly different column headers and types of data.

As we implemented GILBERT, we relied on a few similar datasets for testing. As a result, the database design was built to support these specific datasets but may not allow for the variety of metadata, data formatting, and study structure seen in other datasets. To improve this, we recommend testing GILBERT with various datasets to find ways to configure the database design, and consequently much of the application, to better suit more types of data and metadata. One possible addition is including support for demographic information about each subject, such as their age, sex, and race. Another addition may include storing information related to each device used to measure each attribute, such as model number and precision.

6.4.2 Data Analysis

GILBERT's data analysis feature is currently limited in respect to formatting and flexibility. To combat this, we recommend adding other features, such as filtering flexibility and additional chart options.

GILBERT's current filtering implementation is able to apply basic filtering but is limited in its variability. Users should have the ability to use multiple filters on a single attribute rather than being restricted to just one. We recommend expanding the current implementation to support performing multiple logical operators on the same attribute. Also, in regard to multiple overlapping filters, users should have the option to specify if filters should be applied to include data points that apply to either one filter or another (OR in SQL) or if filters should be applied to only include data points that apply to all filters (AND in SQL). Currently, multiple filters automatically default to only include data points that apply to all of the filters.

Another recommendation is to add other types of analysis to GILBERT such as linear regression and box and whisker plots. We also recommend allowing multiple attributes in a scatter plot or a histogram to allow for more flexible graph making. Customizing size, color, font, and other visual aspects of visualization may also be useful. By offering a wider variety of analyses, users can get more out of GILBERT.

Beyond this, we recommend expanding GILBERT's options even more in the future to allow for specialized and advanced levels of analysis. This may include the use of machine learning or neural networks to perform predictive analysis. This has practical implications, such as diagnosing subjects for a certain condition based on automated analysis of given data.

Other minor improvements to GILBERT's analysis feature include the ability to sort the study analysis query results by a certain column.

6.4.3 User Interface

While we received some positive feedback for GILBERT's overall design, we also received comments that suggest there is work needed to be done to improve the UI. One common feedback point that we received both during the evaluation and from the post-evaluation survey was button placement being inconsistent across the pages. For example, many users struggled to locate the 'Submit' button on the data selection filtering page, which was placed far below the main content section of the page. We suspected that a potential reason for why users could not find the button was because they got used to a button being placed near the content section of the page. Based on this observation and feedback, we recommend there being more consistency for button placement among the pages of the application.

Also, participants found it challenging to locate the link to the help manual due to its position on the page. Based on the feedback received, we recommend adding a help icon near the top of each page that will redirect the user to the help manual when clicked on. Some participants suggested that this would make it easier to access the help page.

Confusion related to terminology was another common observation from participants. Evidence of this can be seen in results from during and after the evaluations. Given that GILBERT uses unique terminology to structure its data and metadata, it is imperative that users are made aware of what components, such as study groups and data categories, are in the context of this application and how they relate to one another. We recommend providing clearer explanations of this terminology and making them readily available. This may take the form of revised instructions on the web pages, revisions to the help manual, and mouse-over tooltips for different terms.

We also noticed that users were more lost when interacting with the uploader when compared to other features. While this was not unexpected, we believe the information presented on the info page was not as effective as we hoped. Multiple users were confused that they could not upload files on the first page of the uploader section. An improvement we thought that could help handle this issue is to add a progress bar at the top section for each page within the uploader section to help the user be aware of where they are in the uploading process.

Lastly, we recommend improving GILBERT's UI by considering general best practice related to UI design. While our team tried to create a clean UI, it was not our main priority compared to implementing GILBERT's core components. Our evaluations helped us realize the importance of UI design as it was a major factor in causing our users to struggle. With a cleaner UI, we believe users will more easily interact with our system. Therefore, we recommend researching and applying more professional and effective UI strategies to the application as a whole.

6.4.4 Study Management

Though GILBERT allows users to upload, view, analyze, and export study data, it does not feature many options for managing and editing studies. When testing GILBERT, Python scripts or direct database access were used to delete or modify data in the database. To give users more control over this, we recommend implementing various study management options on the front end to allow users to edit or delete data. This includes allowing users to modify metadata for studies, study groups, subjects, data categories, and attributes. Users could also delete entire studies or study groups, which may result in multiple deletions within the database.

With its permission system, GILBERT allows study information to either be visible to only the owner of the study or to all users. This is restrictive since it does not allow for multiple users to access a private study. We recommend adding a collaborator permission system to GILBERT so specific users can be assigned to have viewing or editing permissions on a study. This would allow a study to be publicly invisible or visible while still restricting the ability to modify the study to only a select few. Additionally, we recommend implementing a feature that tracks changes within a study. For each action, such as uploading data or modifying metadata, GILBERT would track when this change occurred and which user was involved. A log of this information would be available to study collaborators to help them stay aware of changes to their study.

6.4.5 Uploader Feature

While the uploader is in a functional state, there are limitations that should be addressed in the future version of GILBERT. Currently, the uploader does not handle duplicate files effectively. With the current implementation for detecting duplicate files, GILBERT checks if the name of the file to be uploaded has been used already for its specific data category. Ideally, the users should not have to worry about other file names, so we recommend figuring out a more effective way to handle duplicate files. Another limitation in the current uploader state is that it only supports uploading .csv files. As the project grows, we imagine that GILBERT should be able to handle other file types, both other spreadsheets (e.g., Excel files) and other types of data (e.g., images, sound).

Chapter 7. Conclusion

In healthcare and other biomedical research fields, there are few centralized options for storing the vast amounts of data collected by researchers. To fulfill this need, we created GILBERT, a hub for researchers to upload, access, filter, and analyze biomedical data across multiple studies. We did this using the Agile Development Methodology for twenty-one weeks with a team of six biomedical engineering and computer science students. GILBERT uses the Django framework for web development and a PostgreSQL database for storage. Our team implemented four main features in GILBERT which we categorized as key functionalities at the beginning of our project: the uploader, study analysis, inventory, and the user system.

We conducted a user study to assess the effectiveness of GILBERT and to understand how different users interacted with the system. Twenty-seven participants in the study were randomly assigned to two groups. The groups were assigned a different set of tasks to complete, each task testing one of the four main features of the application. Each evaluation was completed individually by the participant as two members of our team observed to take notes and facilitated the study. Overall, we collected and analyzed data from our user study interest survey, scoring and notes taken during each evaluation, and a post-evaluation survey.

From our evaluation, we concluded that users from many different backgrounds could understand and navigate through GILBERT and that it may be useful in modern research settings. While participants in our evaluation said that GILBERT is straightforward and easy to use, they noted that it lacks some features, customization (specifically in creating graphs), and online help that other data management systems have. Despite these shortcomings, we concluded that GILBERT has the potential to be an effective tool for researchers who need a centralized data pipeline to store and assess a variety of biomedical data.

References

Atlassian. (n.d.). What is Agile? Retrieved October 17, 2020, from https://www.atlassian.com/agile

AWS. (2002). Amazon S3. Retrieved March 02, 2021, from https://aws.amazon.com/s3/

AWS. (2012). Redis. Retrieved March 16, 2021, from https://aws.amazon.com/redis/

Baljak, V., Ljubovic, A., Michel, J., Montgomery, M., & Salaway, R.N. (2018). A Scalable Realtime Analytics Pipeline and Storage Architecture for Physiological Monitoring Big Data. *Smart Health*, 275-286.

Bayer, M. (n.d.). Key Features of SQLAlchemy. Retrieved October 06, 2020, from https://www.sqlalchemy.org/features.html

Claypool, K. (2020, September 24). Personal communication [Interview].

Claypool, K. (2020, October 13). Personal communication [Interview].

Coghlan, N. (2020, February 11). Python 3 Q & A. Retrieved October 06, 2020, from http://python-notes.curiousefficiency.org/en/latest/python3/questions_and_answers.html

Dearmer, A. (2020, September 22). The Importance and Benefits of a Data Pipeline. Retrieved October 06, 2020, from https://www.xplenty.com/blog/what-is-a-data-pipeline/

Divisi, D., Di Leonardo, G., Zaccagna, G., & Crisci, R. (2017). Basic Statistics With Microsoft Excel: a Review. *Journal of Thoracic Disease, 9*(6), 1734-1740. doi:10.21037/jtd.2017.05.81

Dremio. (2020). What is a Data Pipeline? Retrieved October 06, 2020, from https://www.dremio.com/what-is-a-data-pipeline/

Evans, C., González, G. (2019, May 16). Biomedical Image Processing with Containers and Deep Learning: An Automated Analysis Pipeline. *BioEssays, 41*(6). https://doi.org/10.1002/bies.201900004

Flask Documentation. (2020, September 25). Retrieved October 08, 2020, from https://flask.palletsprojects.com/en/1.1.x/#additional-notes

Finn, M., Smith, K. (2020, September 23). Personal communication [Interview].

Freymann, J.B., Kirby, J.S., Perry, J.H., Clunie, D.A., & Jaffe, C.C. (2012). Image Data Sharing for Biomedical Research—Meeting HIPAA Requirements for De-identification. *Journal of Digit Imaging, 25*, 14–24. https://doi.org/10.1007/s10278-011-9422-x

Git. (n.d.). 1.1 Getting Started - About Version Control. Retrieved March 03, 2021, from https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

Heroku. (2020). What is Heroku? Retrieved October 06, 2020, from https://www.heroku.com/what

Hunter, J. D. (2021, January 28). Matplotlib: Visualization with Python. Retrieved February 28, 2021, from https://matplotlib.org/stable/index.html

Hrynaszkiewicz, I., Norton, M. L., Vickers, A. J., Altman, D.G. (2010). Preparing Raw Clinical data for Publication: Guidance for Journal Editors, Authors, and Peer Reviewers. *BMJ*, *340*. https://doi.org/10.1136/bmj.c181

IMDRF Software as a Medical Device (SaMD) Working Group. (2014, September 18). "Software as a Medical Device": Possible Framework for Risk Categorization and Corresponding Considerations. *International Medical Device Regulators Forum*. Retrieved October 03, 2020, from http://www.imdrf.org/docs/imdrf/final/technical/imdrf-tech-140918-samd-framework-risk-categorization-141013.pdf

IMDRF SaMD Working Group. (2015, October 02). Software as a Medical Device (SaMD): Application Of Quality Management System. *International Medical Device Regulators Forum*. Retrieved

October 03, 2020, from http://www.imdrf.org/docs/imdrf/final/technical/imdrf-tech-151002-samd-qms.pdf

IMDRF SaMD Working Group. (2017, December 8). Software as a Medical Device (SaMD): Clinical Evaluation: Guidance for Industry and Food and Drug Administration Staff. *International Medical Device Regulators Forum*. Retrieved October 03, 2020, from https://www.fda.gov/media/100714/download

Janvrin, D. J., Raschke, R. L., & Dilla, W. N. (2014, October 22). Making Sense of Complex Data Using Interactive Data Visualization. *Journal of Accounting Education, 32*(4), 31-48. https://doi.org/10.1016/j.jaccedu.2014.09.003

Inmon, B. (2005). Building the Data Warehouse (4th ed.). Indianapolis, IN: Wiley Publishing Inc.

Lammert, A. (2020, September 30). Personal communication [Interview].

La, B. L. (2014). How Do I Use a Database?. Retrieved from https://ebookcentral-proquest-com.ezpxy-web-p-u01.wpi.edu

MathWorks - Solutions. (2021). www.mathworks.com. Retrieved March 03, 2021, from https://www.mathworks.com/solutions.html#capabilities

McQuistan, A. (n.d.). Asynchronous Tasks in Django with Redis and Celery. Retrieved March 16, 2021, from https://stackabuse.com/asynchronous-tasks-in-django-with-redis-and-celery/

MDN Contributors. (2019, November 30). Django Introduction. Retrieved October 06, 2020, from https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction

MySQL. (2020). About MySQL. Retrieved October 06, 2020, from https://www.mysql.com/about/

Navathe, S. B. (1992). Evolution of data modeling for databases. *Communications of the ACM 35*(9), 112–123. https://doi.org/10.1145/130994.131001

NIH Data Sharing Policies. (2019, August 9). Retrieved October 09, 2020, from https://www.nlm.nih.gov/NIHbmic/nih_data_sharing_policies.html

NumFOCUS. (2020). About Pandas. Retrieved October 06, 2020, from https://pandas.pydata.org/about/

Pandey, S., Sultan, M. (2020, September 21). Personal communication [Interview].

Poudel, U. (2020, September 14). Personal communication [Interview].

Protasiewicz, J. (2018, June 26). Flask vs. Django Comparison. https://www.netguru.com/blog/flask-vs-django.

Python Software Foundation. (n.d.). What is Python? Executive Summary. Retrieved October 06, 2020, from https://www.python.org/doc/essays/blurb/

Samadani, U., Ritlop, R., Reyes, M., Nehrbass, E., Li, M., Lamm, E., Schneider, J., Shimunov, D., Sava, M., Kolecki, R., Burris, P., Altomare, L., Mehmood, T., Smith, T., Huang, J. H., McStay, C., Todd, S. R., Qian, M., Kondziolka, D., Wall, S., … Huang, P. (2015). Eye Tracking Detects Disconjugate Eye Movements Associated with Structural Traumatic Brain Injury and Concussion. *Journal of Neurotrauma*, *32*(8), 548–556. https://doi.org/10.1089/neu.2014.3687

Schaefer, L. (2020). What is NoSQL? NoSQL Databases Explained. Retrieved October 07, 2020, from https://www.mongodb.com/nosql-explained

Schema on Write. (2020). Retrieved from https://www.techopedia.com/definition/30899/schema-on-write.

Schuller, B., et al. (2013) Paralinguistics in Speech and Language—State-of-the-art and the Challenge. *Computer Speech & Language*, *27*, 4-39.

Sharma, B. (April 2018). Architecting Data Lakes (2nd ed.). Sebastopol, CA: O'Reilly Media Inc.

Shin, D. D., Ozyurt, I. B., & Liu, T. T. (2013). The Cerebral Blood Flow Biomedical Informatics Research Network (CBFBIRN) Database and Analysis Pipeline for Arterial Spin Labeling MRI Data. *Frontiers in Neuroinformatics*, *7*(21). https://doi.org/10.3389/fninf.2013.00021

Solomon, B. (2021, February 06). Python Plotting with Matplotlib (Guide). Retrieved February 28, 2021, from https://realpython.com/python-matplotlib-guide/

Sumathi, S., & Esakkirajan, S. (2007). Fundamentals of Relational Database Management Systems. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-48399-1

Tchagna Kouanou, A., Tchiotsop, D., Kengne, R., Zephirin, D., Adele Armele, N. & Tchinda, R. (2018). An Optimal Big Data Workflow for Biomedical Image Analysis. *Informatics in Medicine Unlocked*, *11*, 68-74.

The PostgreSQL Global Development Group. (2021). About. Retrieved February 28, 2021, from https://www.postgresql.org/about/

Tippmann, S., 2014. Programming tools: Adventures with R. *Nature*, 517(7532), pp.109-110.

U.S. Food and Drug Administration. (2020). Software As A Medical Device (SaMD). Retrieved October 4, 2020, from https://www.fda.gov/medical-devices/digital-health-center-excellence/software-medical-device-samd

U.S. Food and Drug Administration. (2020). What We Do. Retrieved October 4, 2020, from https://www.fda.gov/about-fda/what-we-do

Wang, S., et. al. (2020). A Fully Automatic Deep Learning System for COVID-19 Diagnostic and Prognostic Analysis. *European Respiratory Journal*, *56*(2). doi:10.1101/2020.03.24.20042317

Appendix A: Umanga Poudel Interview Questions

1. Tell us about your research and what you do
2. What kind of data do you work with?
    a. What format is this data stored in?
        i. If possible, ask what kind of database(s) they use and why (looking to see if there may be advantages between the dbs)
3. What tools do you use for data management and analysis?
    a. Who are the target users of the system?
    b. What are the inputs and results of the analysis tools?
        i. What kinds of outputs does your system provide?
        ii. What kinds of outcomes are you interested in - clinical, technological, scientific?
    c. What kind of queries does it support?
    d. What kind of analysis or prediction does it support?
    e. Do you use these tools for generating reports, data models, or data visualizations? If so, how and for what purposes?
    f. What do you find particularly useful about these tools that make it easier for you to conduct your research?
    g. What shortcomings does your current system have that we could potentially include or make easier in our pipeline?
        i. In addition, are there any recurring obstacles that teams face when working with biomedical data?
        ii. What kinds of constraints do users have with this system- time, accuracy, richness of information?
4. What would you like to see in a pipeline like the one we aim to make?
    a. What are the must-have characteristics?
5. What practices do you follow for data privacy and confidentiality?
    a. Do you anonymize the data? If so, how?

Appendix B. Adam Lammert Interview Questions

1. What inspired you and Professor Claypool to propose this MQP?
2. How do you see a Biomedical Data Pipeline being used in your work with speech data?
    a. Have you used any data management pipelines in previous work?
3. What do you see being the strongest contributions to this project from a biomedical standpoint?
4. What are some of the most common challenges faced when dealing with large sets of raw physiological signals?
    a. If you have experienced these challenges in the past, what were methods to overcome them?
5. We are interested in incorporating some extent of diagnostics and predictability. Have you worked with any research before where the end goal was to produce a diagnosis?
6. When considering the creation of a user interface, what are some of the most important aspects that would make it easy to use for a clinician who may not be familiar with the code?

Appendix C. Emmanuel Agu Interview Questions

1. Tell us about what kind of research/projects you've worked on
    a. Can you explain your views on mobile devices being a platform for delivering healthcare?
2. For these projects and research, what kind of data did you work with?
    a. What format is this data stored in?
        i. For large files, how did you go about in storing them?
    b. How were these data used in his project?
        i. More specifically how did you utilize the collected gait data?
3. What tools do you use for data management and analysis?
    a. Who are the target users of the system?
    b. What are the inputs and results of the analysis tools?
        i. What kinds of outputs does your system provide?
        ii. What kinds of outcomes are you interested in - clinical, technological, scientific?
    c. What are the selection capabilities of the system?
        i. For data management tools, how is it queried?
        ii. For analysis, can subsets of the data be selected?
    d. What kind of analysis or prediction does it support?
    e. Do you use these tools for generating reports, data models, or data visualizations? If so, how and for what purposes?
    f. What do you find particularly useful about these tools that make it easier for you to conduct your research?
    g. What shortcomings does your current system have that we could potentially include or make easier in our pipeline?
        i. What kinds of constraints do users have with this system- time, accuracy, richness of information?
4. Given the description of our pipeline, do you think you would use it for your research and how would you use it?
    a. What are the must-have characteristics?
5. What practices do you follow for data privacy and confidentiality?
    a. Do you anonymize the data? If so, how?

Appendix D. Exercise IQP Interview Questions

1. Can you briefly overview the data collection and data storage process?
    a. What tools did you use to store the data?
2. Did you have any difficulties storing and organizing your data?
    a. If so, what could have been helpful?
3. Can you differentiate some of the files that were saved?
    a. For example, subject 1 has multiple .csv files for corsi, flanker, and Fitbit data.
4. Why do some subjects have trimmer heart rate data while others don't? Was this only performed for certain subjects?
5. Can you explain the different columns in the .csv files for the corsi results?
6. Is the data stored just the raw data? Does it also include calculated/processed data?
    a. How did you differentiate the stored data?
7. Would a system that allowed you to select data according to variables or constraints (for example, extracting just heart rate data) have been useful in doing analysis?
    a. If so, what attributes would you have looked for in a selection process?

Appendix E. TBI IQP Interview Questions

a. What was your data collection and organization process?
    i. How did you manage your raw data from the apps you used?
    ii. Did you explore any database structures for storage?
b. How did you organize your analysis process?
    i. Report talks about individually processing each patients data through Python → excel
    ii. Was there any automation involved in moving the raw data to python and excel?
c. What are characteristics of data pipelines that you found important as you were creating one?
    i. What are some challenges you saw in building your pipeline that our team could address/look into?

Appendix F. Kajal Claypool Meeting Agenda

- Clarify the potential issues that may exist in our current database schema (the one we discussed with you last time)
- Present our preliminary approach to using NoSQL
- Talk about possibility of using both MySQL and a NoSQL database
- Ask about existing upload systems that may be helpful for us to check out

Appendix G. User stories

User Stories from Sprint 1:

- As a researcher, I want a place to store all my data for analysis so that I can easily keep track of and access it.
- As a researcher, I want to select a subset of the data to analyze data with common features.
- As a researcher, I want my data to be stored in a convenient place so that it is quick to access for my analysis.

User Stories from Sprint 2:

- As a consumer, I want to be able to view the results of the query, so I can see the analysis results of the data.
- As a researcher, I want to be able to parse my data into the database so that I can take the data and use it for my analysis.
- As a researcher, I want to be able to get simple descriptive metrics (mean, sd, range, etc.) from a subset of data I pull from the pipeline.
- As a consumer, I want to filter the data in multiple different ways so that I can compute analysis on a more streamlined version of the data.
- As a researcher, I want to gather information about the database such as row counts, column names, and the number of subjects, to learn more about the data that was added.

User Stories from Sprint 3:

- As a researcher, I want to see how much progress there has been on my data upload so I can know how much time it will take for the upload to complete.
- As a researcher, I want to perform analysis such as T-tests or regression on data.
- As a clinician, I want to use the web application to select and filter data so that I can view information that is important to me.
- As a user of the system, I would like the information that I entered into the page to stay on the page upon a redirect back to the page so that I do not have to re-enter most of the information back again

User Stories from Sprint 4:

- As a researcher, I want to add information about the study such as a study objective, if it is IRB approved, involved institutions and contact information, start and end dates, and a description of the study so other I and other researchers can use this information to compare studies in the database.
- As a developer, I would like to implement a login system for all users so the system only allows certain users to have access to sensitive information.
- As a researcher, I would like the web browser application to upload my data so I can analyze it.
- As a researcher, I would like to know the units of measurements to numerical attributes so my analysis is accurate.
- As a researcher, I would like to perform feature extraction on the data I have analyzed so I can obtain different perspectives on the raw data.

User Stories from Sprint 5:

- As a researcher, I want to select data from multiple different sources so I can easily analyze many types of data.
- As a researcher, I want to generate data visualizations from the stored data and analysis.
- As a user of the system, I would like a help page on the browser so I can better understand how to use the system.
- As a developer of the system, I want to have my database on a server instead of a local machine because it's more versatile.
- As a PI, I want to see which files I have uploaded to the system after I have uploaded them so I can remember what data I have added to the database.
- As a researcher, I would like to export my results as a .csv so I can use it for other applications.
- As a user of the system, I would like to gather information about the inventory so that I am aware of the data that is currently stored within the database.
- As a researcher, I want an easy-to-follow platform to perform analysis on data so I can easily see what I am doing, which data I am using, and the results of the analysis.

User Stories from Sprint 6:

- As a researcher, I want an easy-to-follow platform to perform analysis on data so I can easily see what I am doing, which data I am using, and the results of the analysis.
- As a developer, I would like to refactor the code base so that the code is cleaner and easier to maintain.
- As a developer, I want to grant permission to certain users so they have access to certain studies so that sensitive information is only accessed by those who are allowed to use it and need to.
- As a researcher, I would like to perform feature extraction on the data I have analyzed so I can obtain different perspectives on the raw data.

User Stories on Backlog:

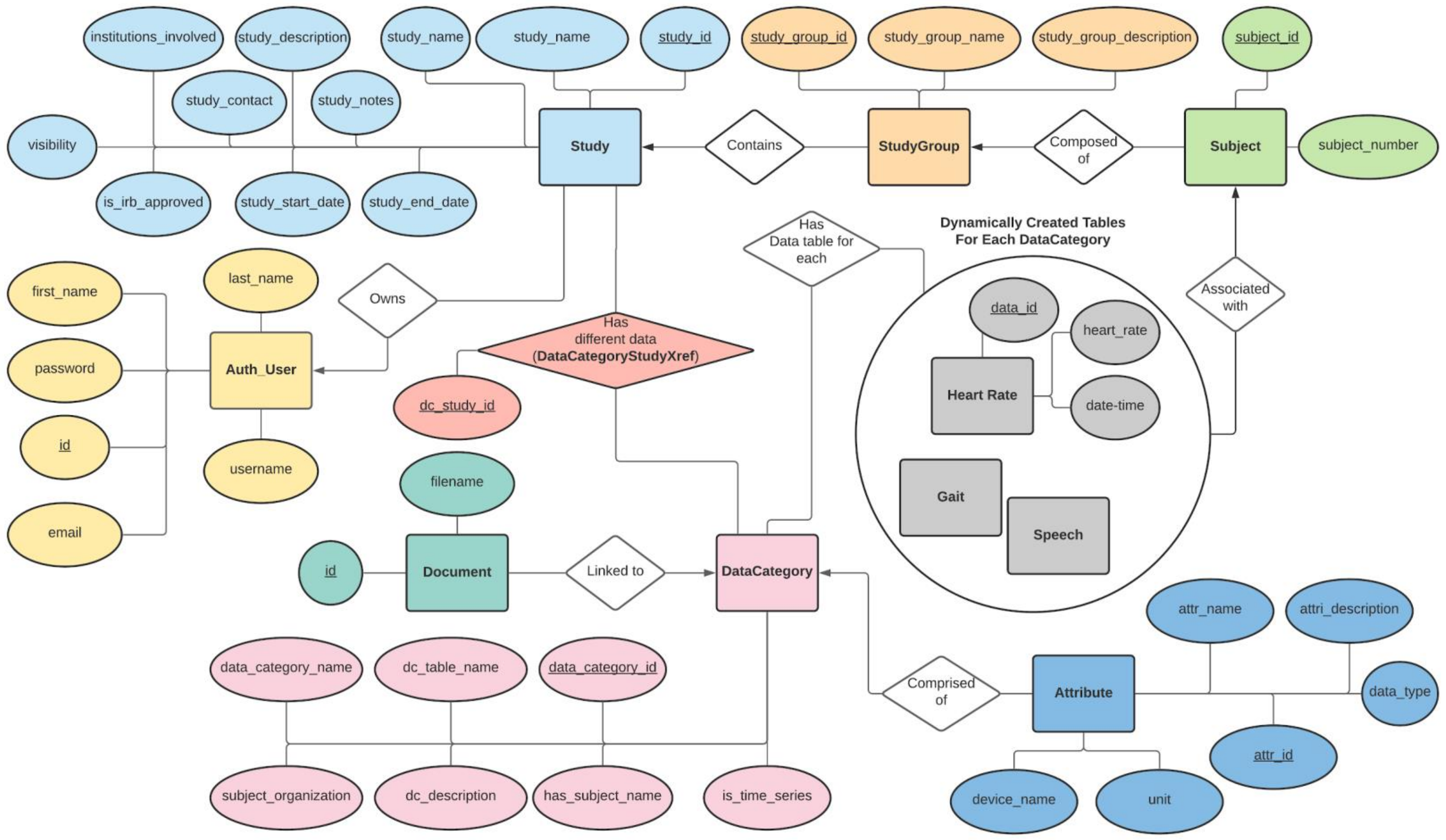
- As a researcher, I want to see the last time a certain data set has been updated so I can know if the data still needs to be updated.
- As a researcher, I want to update and delete data in the database so that the data in the database will be accurate.
- As a researcher, I would like to use tags in studies so I can search for studies that contain a certain tag.
- As a researcher, I want the database to be flexible enough for me to add different survey data, so that I can store and be able to reference the data.
- As a PI, I want to distinguish between no data, missing data, and N/A data so that users will better understand what an empty value represents.
- As a consumer, I want to see a diagnosis so I can know if my outcome is positive or negative.
- As a researcher, I want to run a correlation analysis so I can determine if certain data relates to certain diagnoses/conditions.
- As a researcher, I would like to add data processed through feature extraction into the database so that other researchers will have access to the analysis I had performed.

- As a researcher, I would like to filter study data based on demographic information so I can understand how different demographic groups were affected by different studies.
- As a developer, I want to grant permission to certain users so they have access to certain studies so that sensitive information is only accessed by those who are allowed to use it and need to.
- As a researcher, I would like to specify if my filters will be combined using AND or OR so that I can obtain a more accurate subset of data.

Appendix H. Link to GitHub Repository

https://github.com/WPI-DataPipelineMQP/Biomedical-Analysis-DataPipeline

Appendix I. Entity-Relationship Diagram of Database Schema



Appendix J. Help Manual

Table of Contents

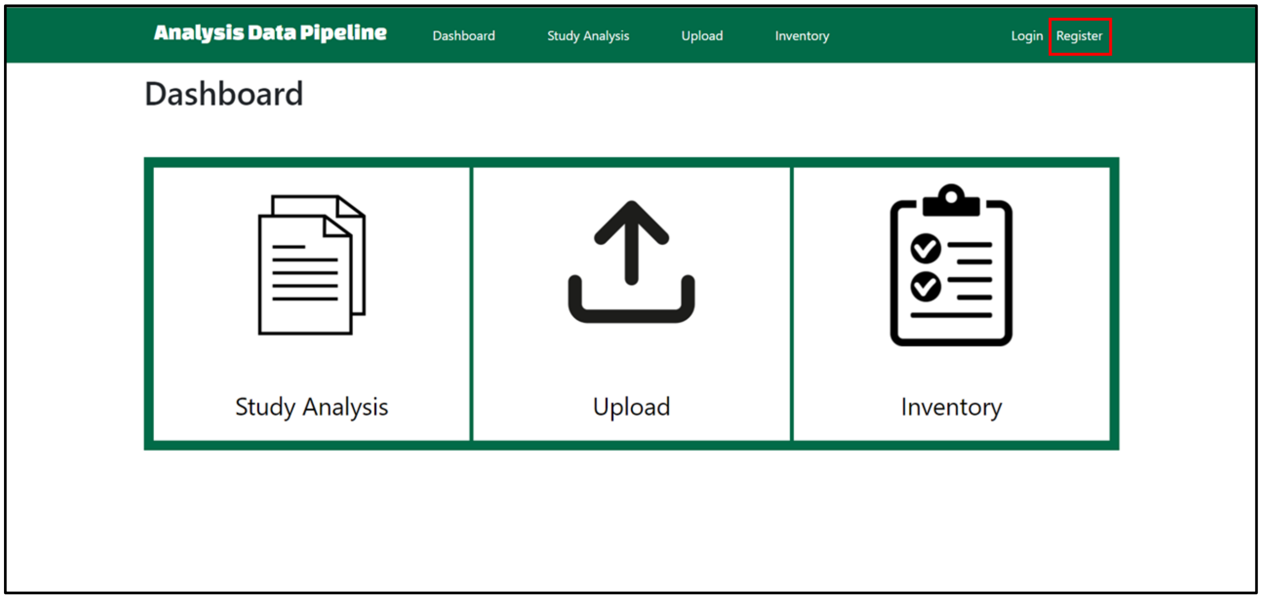
1. User System

This application allows users to create their own accounts. This is required for uploading study data, as the account is linked to the study so only verified users can upload to it. The study analysis and inventory analysis features do not require you to be logged in. However, you can only view studies that have a visibility of “Public”. Studies with the visibility of “Private” are only visible to their owners.

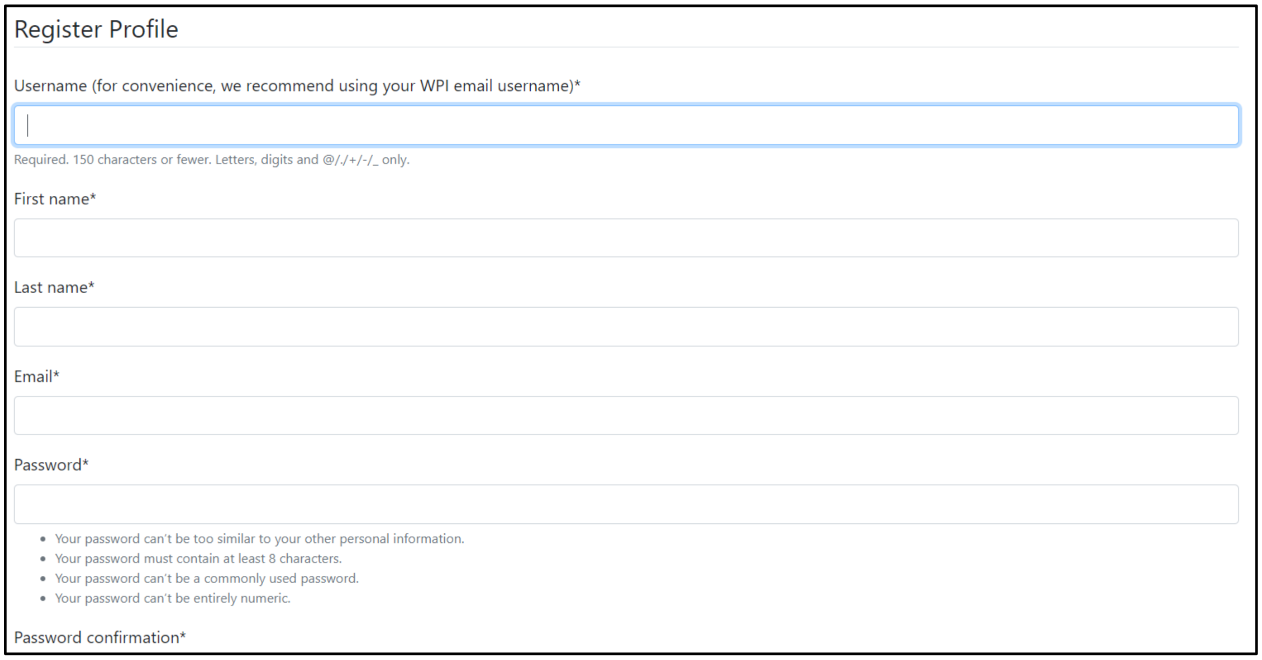
1.1 Register Profile and Login

1.1.1 Register

To register a profile click the “Register” text on the top right of the nav bar. Note this will only show on the nav bar if you are not logged in.



This will bring you to a new page with input fields for your account. Please enter a username, first name, last name, email address, and password. Then, click register to create your profile. Note that the password has specific requirements to ensure that it is secure. Profiles must have unique usernames and email addresses.



Register Profile

Username (for convenience, we recommend using your WPI email username)*

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

First name*

Last name*

Email*

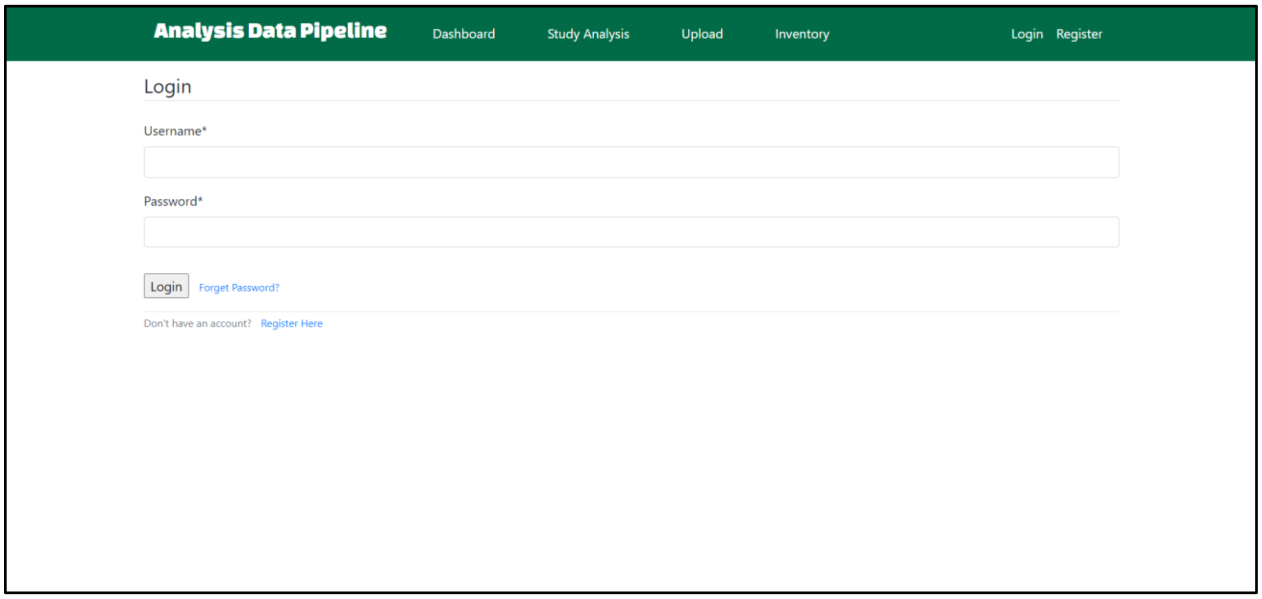
Password*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation*

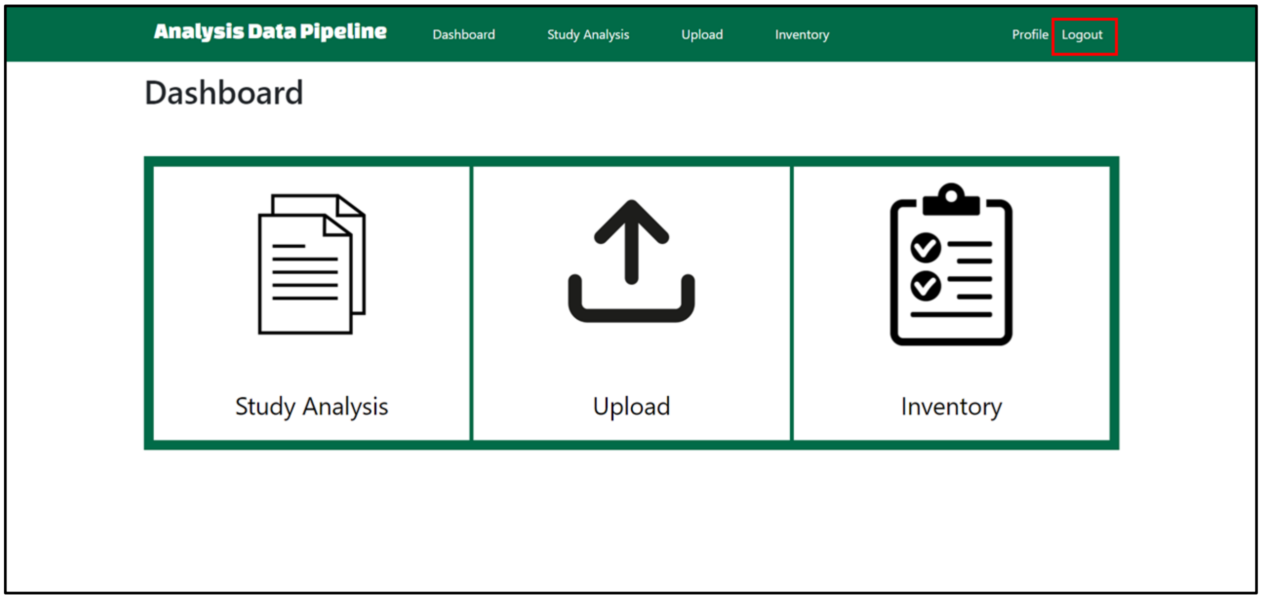
1.1.2 Login

Once the profile has been created, you will be prompted to log in. You can also reach the login screen by clicking the "Login" text on the nav bar.



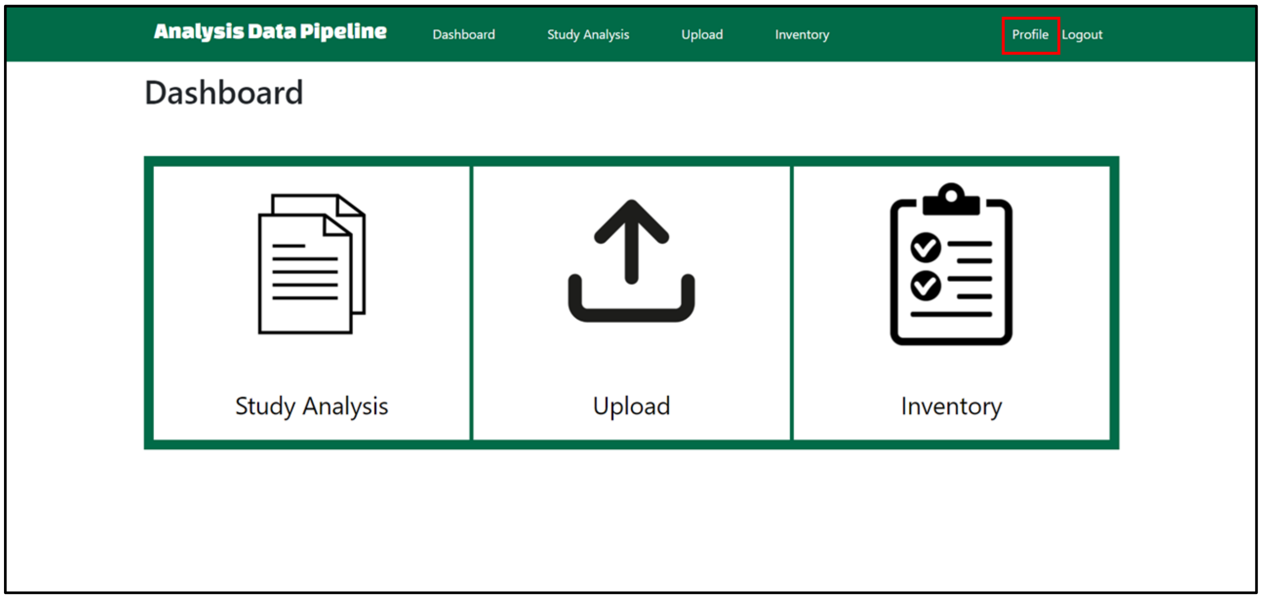
1.2 Logout

To log out, click the "Logout" text in the top right of the nav bar. Note that you need to be logged in to see this option.

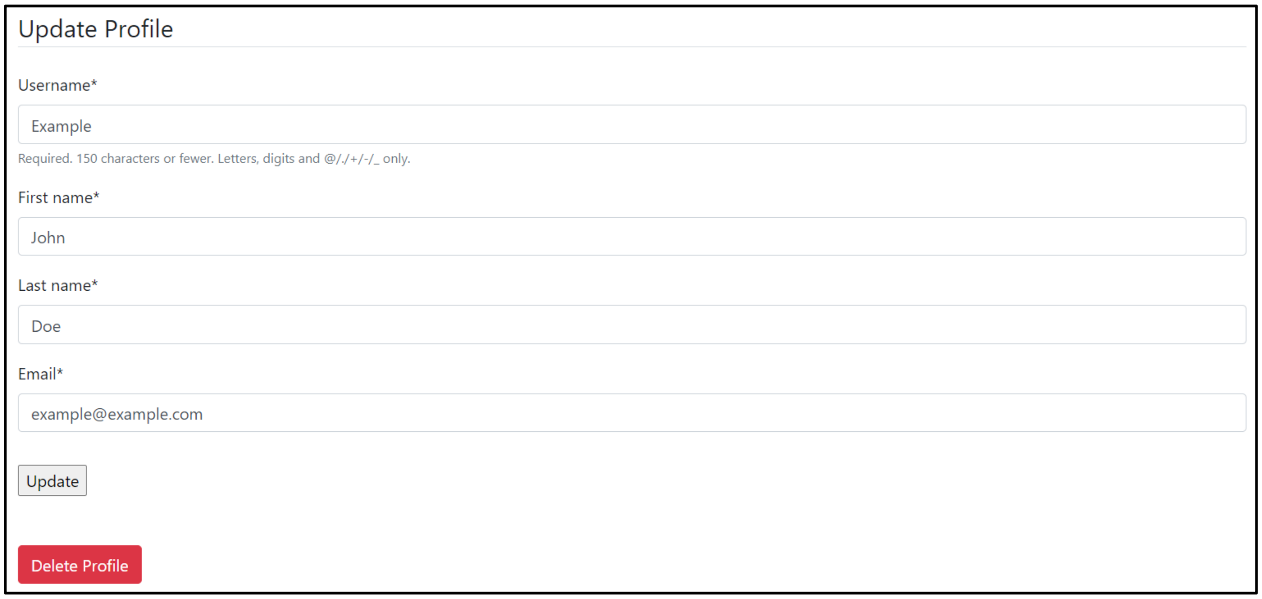


1.3 Update Profile

To view or update your profile information, click the “Profile” text at the top right of the nav bar.



This next page will show your profile information and allow you to update the information by clicking the “Update” button. If successful, a popup at the top of screen will confirm the update was successful.



Update Profile

Username*

Example

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

First name*

John

Last name*

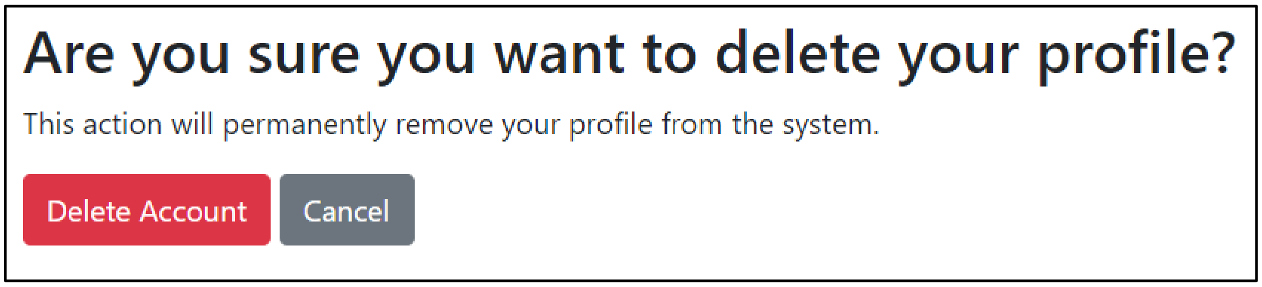
Doe

Email*

example@example.com

1.3.1 Delete Profile

Users can also delete their profile by clicking the “Delete Profile” button. This will take them to another screen to ensure they want to actually delete their profile. Clicking “Delete Account” will delete the account from the system and return the user to the login page.

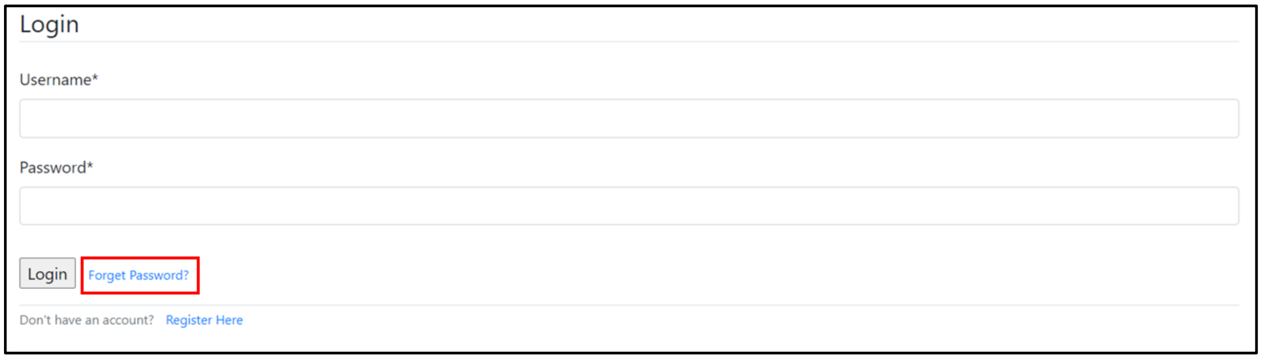


Are you sure you want to delete your profile?

This action will permanently remove your profile from the system.

1.4 Reset Password

If you forgot your password or want to change it, go to the Login page and click "Forgot Password". Note that you must be logged out to get to the login page.

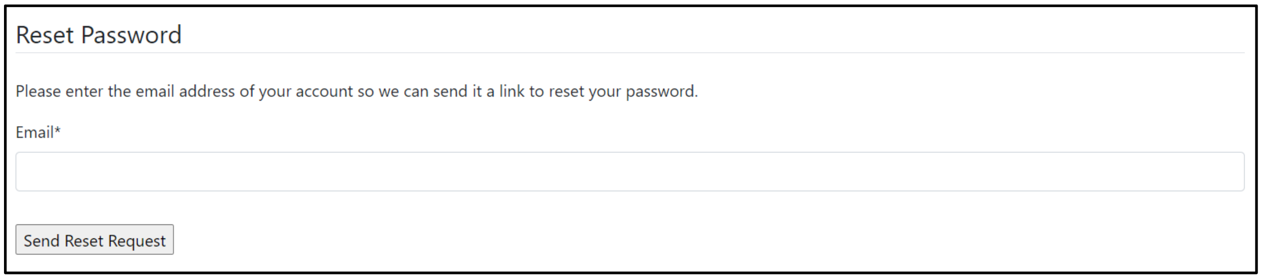

Login

Username*

Password*

Don't have an account? Register Here

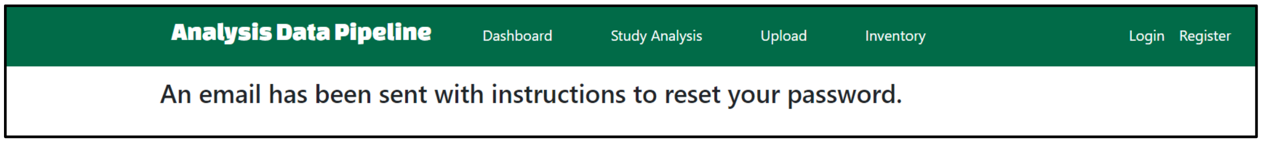
On the next page, enter the email address of your account and click "Send Reset Request".


Reset Password

Please enter the email address of your account so we can send it a link to reset your password.

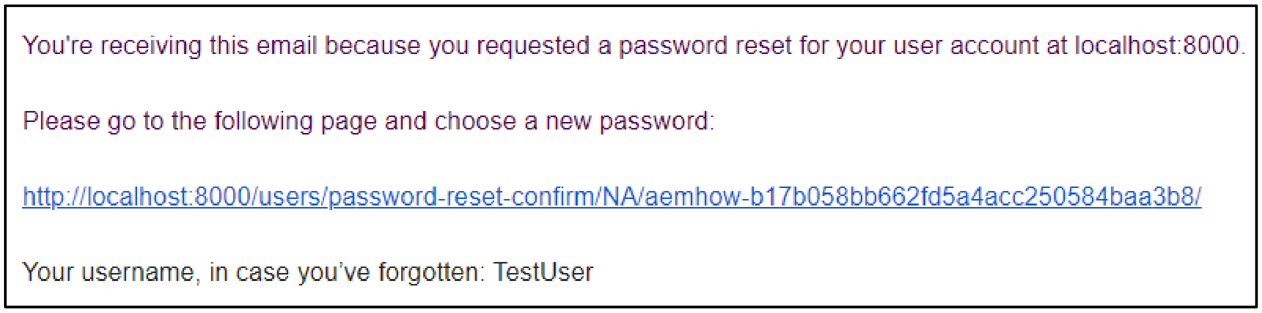
Email*

You will then be taken to the following screen. For security reasons, this will not confirm if there is an existing profile with that email.



An email has been sent with instructions to reset your password.

If there is a profile with the entered email address, it will receive an email from wpidatapipeline@gmail.com with the following format:



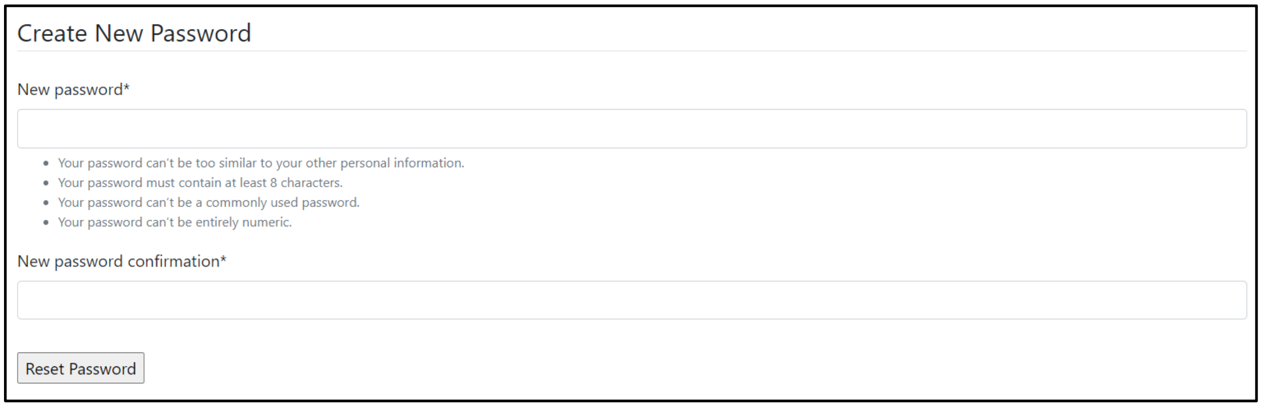
You're receiving this email because you requested a password reset for your user account at localhost:8000.

Please go to the following page and choose a new password:

http://localhost:8000/users/password-reset-confirm/NA/aemhow-b17b058bb662fd5a4acc250584baa3b8/

Your username, in case you've forgotten: TestUser

Clicking the link from the email will lead to the following page where the user can reset their password and then be redirected to the login page.



Create New Password

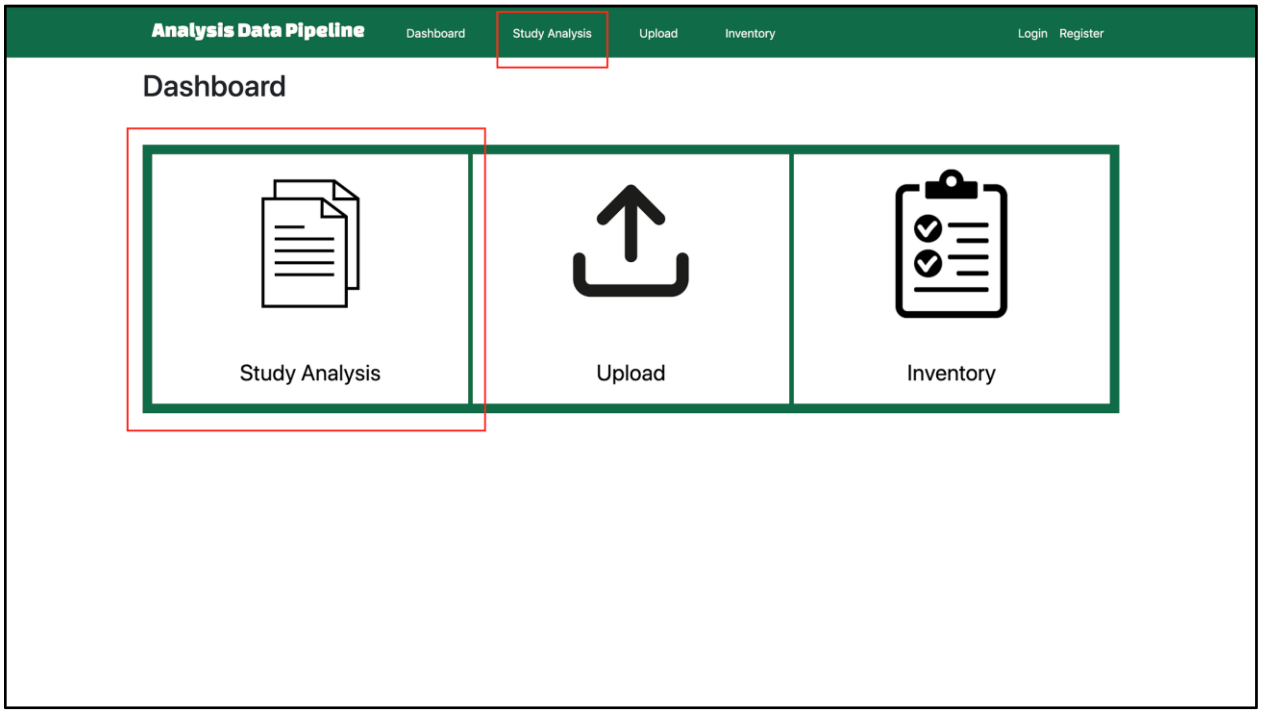
New password*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

New password confirmation*

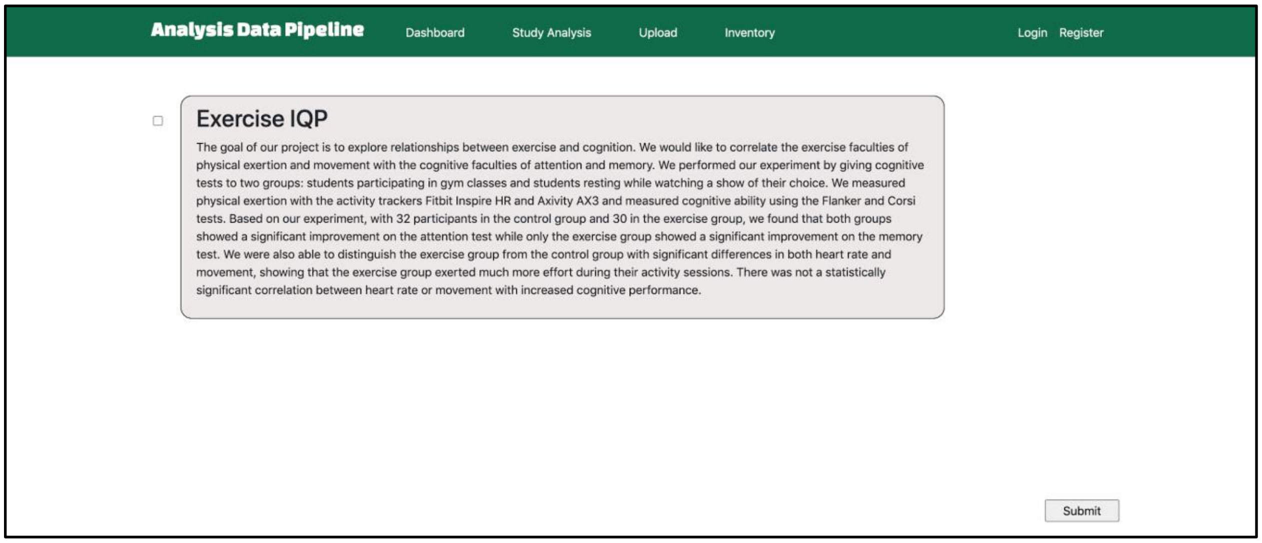
2. Study Analysis

The Study Analysis feature of the pipeline allows you to view and filter uploaded data to analyze and export for future use. You can access this feature by clicking “Study Analysis” on the Dashboard or the nav bar.



2.1. Selecting A Study

This page lists all of the studies with data in the pipeline, as well as the description of the study. Here, you may select one or more studies to analyze by clicking the checkboxes next to the study name. Once you have selected your study/studies, click “Submit”.



Exercise IQP

The goal of our project is to explore relationships between exercise and cognition. We would like to correlate the exercise faculties of physical exertion and movement with the cognitive faculties of attention and memory. We performed our experiment by giving cognitive tests to two groups: students participating in gym classes and students resting while watching a show of their choice. We measured physical exertion with the activity trackers Fitbit Inspire HR and Axivity AX3 and measured cognitive ability using the Flanker and Corsi tests. Based on our experiment, with 32 participants in the control group and 30 in the exercise group, we found that both groups showed a significant improvement on the attention test while only the exercise group showed a significant improvement on the memory test. We were also able to distinguish the exercise group from the control group with significant differences in both heart rate and movement, showing that the exercise group exerted much more effort during their activity sessions. There was not a statistically significant correlation between heart rate or movement with increased cognitive performance.

2.2 Selecting Data Category and Study Group

Next, you will be asked to select which data categories and study groups you would like to analyze.

Data Category: The type of data that was collected in the study. For example, a study exploring the relationship between exercise and memory would have several data categories, like heart rate and cognitive test results.

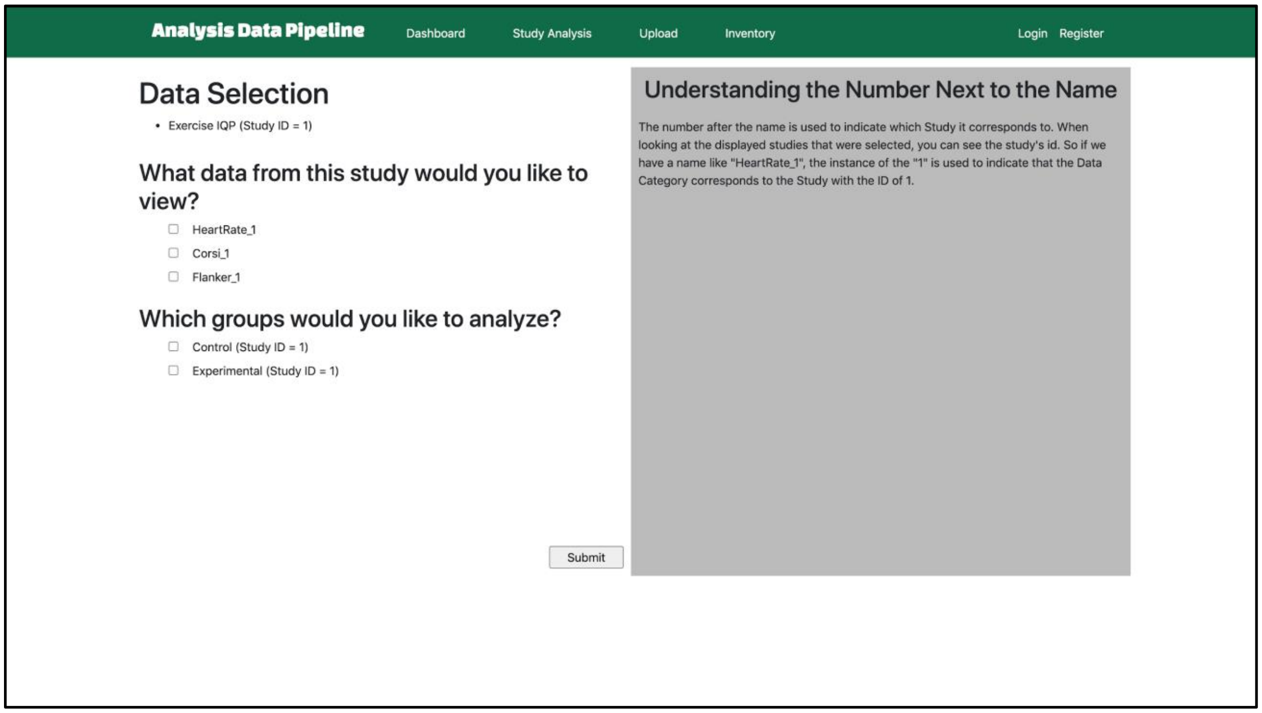
Study Group: The groups that subjects were split into for a study, such as control and experimental or several sample groups.

At the top of the page, you will see the study/studies that you chose, along with its study number. The study number uniquely identifies the study.

Below this, you will see all the data categories and study groups listed for the study/studies you selected. Next to each data category and study group is its corresponding study number. This is to aid you in identifying which data categories and study groups belong to which studies when doing cross study analysis.

You may select more than one data category, but you must select at least one. You may select more than one study group, but you must select at least one.

To select a data category or study group, click the checkbox next to its name. Once you have selected your data categories and study groups, click "Submit".



Data Selection

- Exercise IQP (Study ID = 1)

What data from this study would you like to view?

- ☐ HeartRate_1
- ☐ Corsi_1
- ☐ Flanker_1

Which groups would you like to analyze?

- ☐ Control (Study ID = 1)
- ☐ Experimental (Study ID = 1)

Understanding the Number Next to the Name

The number after the name is used to indicate which Study it corresponds to. When looking at the displayed studies that were selected, you can see the study's id. So if we have a name like "HeartRate_1", the instance of the "1" is used to indicate that the Data Category corresponds to the Study with the ID of 1.

2.3 Selecting Data Attributes and Filtering

On this page, you can select the specific attributes you would like to view and filter how you would like to view them.

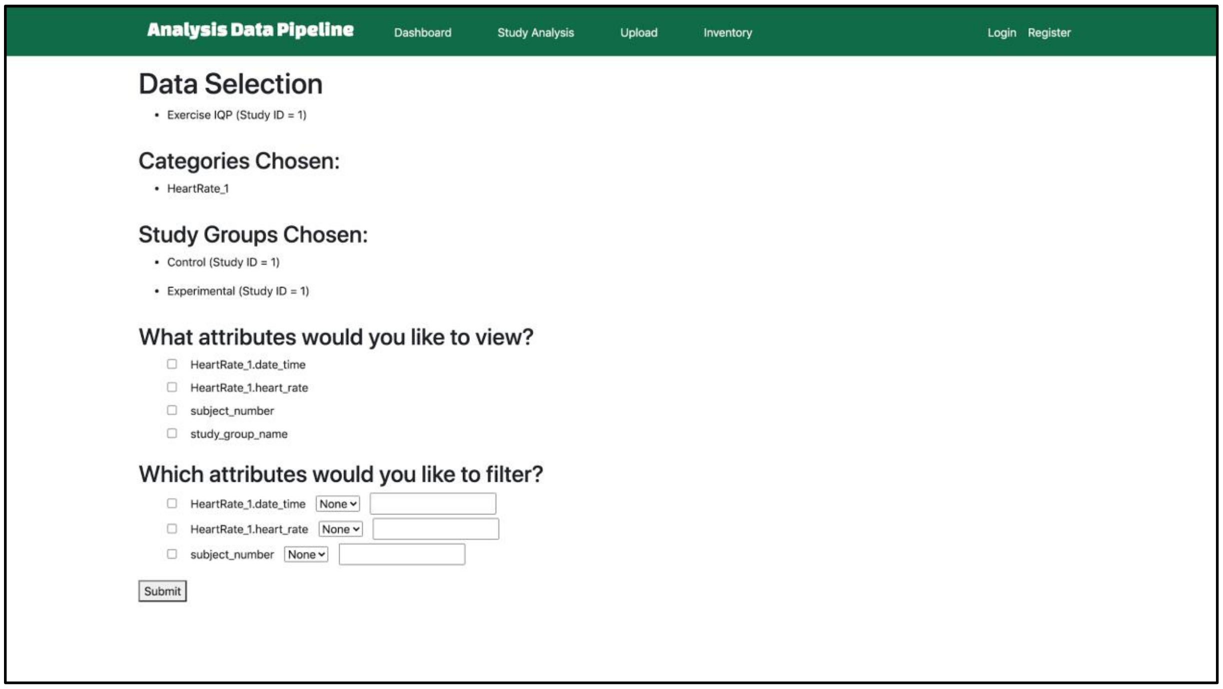
Attribute: The specific fields collected for each data category. For example, a study collecting heart rate data (data category) would collect data for attributes like the heart rate in bpm, as well as what time that reading was taken. Often, these are the columns of the data table.

At the top of the page, you will see the study/studies, data categories, and study groups that you selected.

Below this, you will see a list of all the attributes for the data category/categories you selected, titled "What attributes would you like to view?" The attributes you select here will be shown as columns in the output table. You may select more than one data attribute, but you must select at least one. To select the attribute, click the checkbox next to its name.

In the section titles "Which attributes would you like to filter?", each attribute is listed with a checkbox, a dropdown menu, and a text box. Here, you may filter the attributes based on their values. To filter the results based on the attribute, click the checkbox next to the attribute name, select a symbol from the dropdown, and enter the value that you would like to filter by into the text box. For example, setting heart rate to "=" in the dropdown box, and typing "106" into the text box will yield all the heart rate values that are equal to 106.

Note: You may filter by attributes that you did not select for viewing. Attributes selected for viewing and attributes selected for filtering are not dependent on each other.



Data Selection

- Exercise IQP (Study ID = 1)

Categories Chosen:

- HeartRate_1

Study Groups Chosen:

- Control (Study ID = 1)
- Experimental (Study ID = 1)

What attributes would you like to view?

- [ ] HeartRate_1.date_time
- [ ] HeartRate_1.heart_rate
- [ ] subject_number
- [ ] study_group_name

Which attributes would you like to filter?

- [ ] HeartRate_1.date_time None
- [ ] HeartRate_1.heart_rate None
- [ ] subject_number None

Submit

2.4 Output Screen

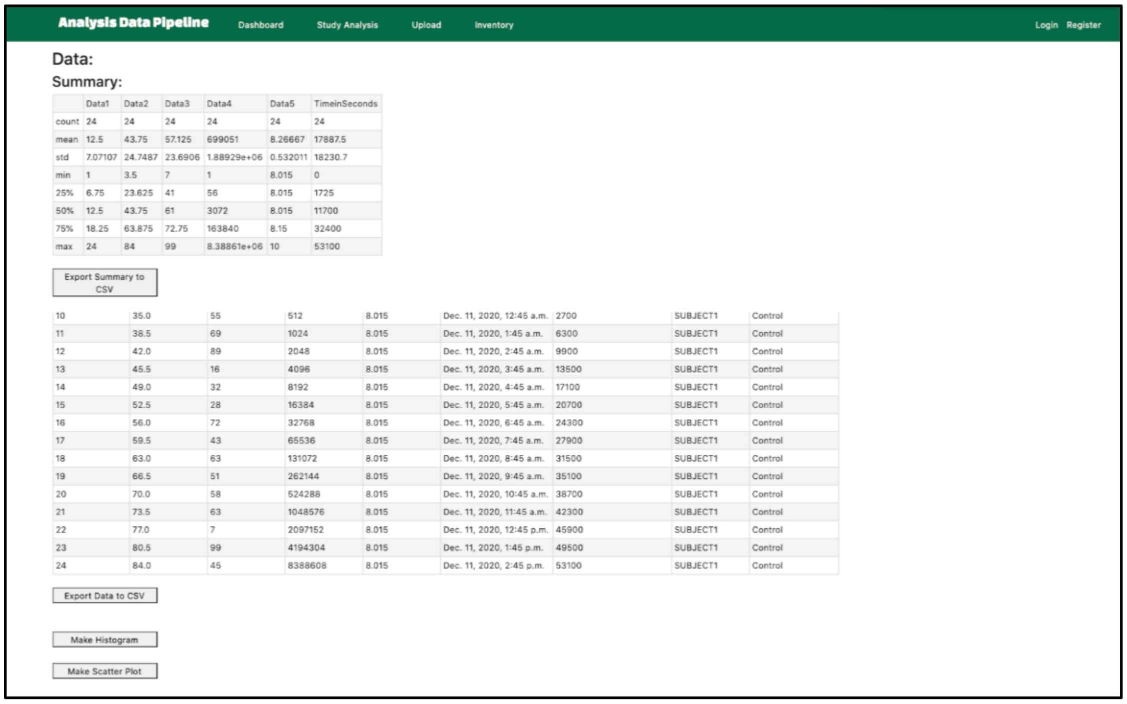
After performing a selection, two tables are shown. One is a statistical summary of the data and the other shows the selected data.

The data summary at the top of the page provides a list of all data columns with numerical values. The rows consist of eight descriptive metrics for each of these data columns.

The table below the summary contains all data that was selected from the previous process. It is formatted as a smaller box for formatting, but the user can scroll through the data to see it in its entirety.

Both tables can be exported to ".csv" files. The "Export data to CSV" buttons for each will be found under their respective tables, and clicking the button will start the ".csv" download.

To further analyze the data, select either option below to generate a histogram or a scatter plot.



Data:

Summary:

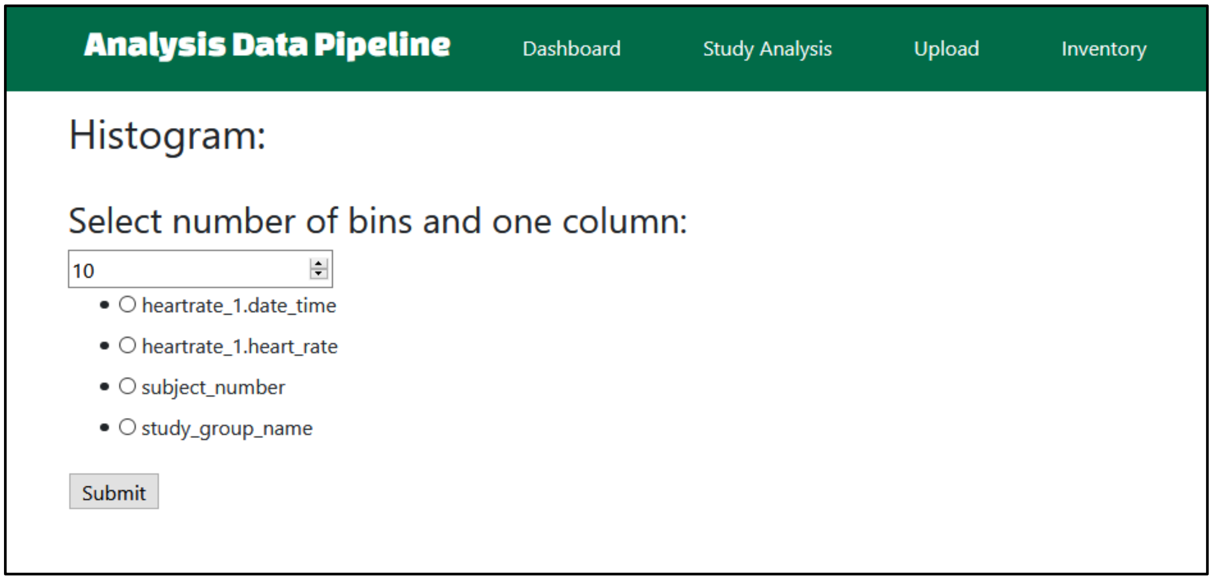
	Data1	Data2	Data3	Data4	Data5	TimeinSeconds
count	24	24	24	24	24	24
mean	12.5	43.75	57.125	699051	8.26667	17887.5
std	7.07107	24.7487	23.6906	1.88929e+06	0.532011	18230.7
min	1	3.5	7	1	8.015	0
25%	6.75	23.625	41	56	8.015	1725
50%	12.5	43.75	61	3072	8.015	11700
75%	18.25	63.875	72.75	163840	8.15	32400
max	24	84	99	8.38861e+06	10	53100

10	35.0	55	512	8.015	Dec. 11, 2020, 12:45 a.m.	2700	SUBJECT1	Control
11	38.5	69	1024	8.015	Dec. 11, 2020, 1:45 a.m.	6300	SUBJECT1	Control
12	42.0	89	2048	8.015	Dec. 11, 2020, 2:45 a.m.	9900	SUBJECT1	Control
13	45.5	16	4096	8.015	Dec. 11, 2020, 3:45 a.m.	13500	SUBJECT1	Control
14	49.0	32	8192	8.015	Dec. 11, 2020, 4:45 a.m.	17100	SUBJECT1	Control
15	52.5	28	16384	8.015	Dec. 11, 2020, 5:45 a.m.	20700	SUBJECT1	Control
16	56.0	72	32768	8.015	Dec. 11, 2020, 6:45 a.m.	24300	SUBJECT1	Control
17	59.5	43	65536	8.015	Dec. 11, 2020, 7:45 a.m.	27900	SUBJECT1	Control
18	63.0	63	131072	8.015	Dec. 11, 2020, 8:45 a.m.	31500	SUBJECT1	Control
19	66.5	51	262144	8.015	Dec. 11, 2020, 9:45 a.m.	35100	SUBJECT1	Control
20	70.0	58	524288	8.015	Dec. 11, 2020, 10:45 a.m.	38700	SUBJECT1	Control
21	73.5	63	1048576	8.015	Dec. 11, 2020, 11:45 a.m.	42300	SUBJECT1	Control
22	77.0	7	2097152	8.015	Dec. 11, 2020, 12:45 p.m.	45900	SUBJECT1	Control
23	80.5	99	4194304	8.015	Dec. 11, 2020, 1:45 p.m.	49500	SUBJECT1	Control
24	84.0	45	8388608	8.015	Dec. 11, 2020, 2:45 p.m.	53100	SUBJECT1	Control

2.4.1: Histogram

Clicking the “Make Histogram” button will bring the user to a new page with a selection prompt. From here, the user can select the amount of bins (equally spaced intervals for data points to fall into) as well as a single attribute from the selected data to be used in the histogram. The default number of bins is 10, but can go as low as 1 and infinitely high.

Clicking the submit button will redirect to a new page that displays the figure created from the selection prompt. For most Web browsers, right click and select 'Save as' to save the image to a local file.

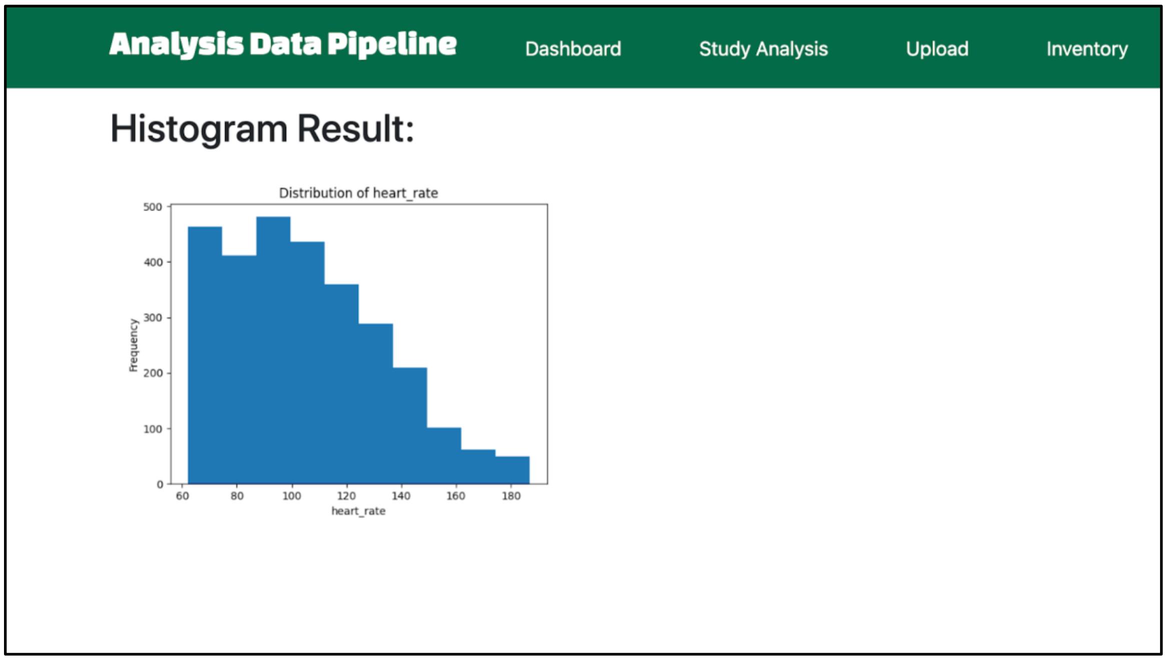


Histogram:

Select number of bins and one column:

10

- ○ heartrate_1.date_time
- ○ heartrate_1.heart_rate
- ○ subject_number
- ○ study_group_name

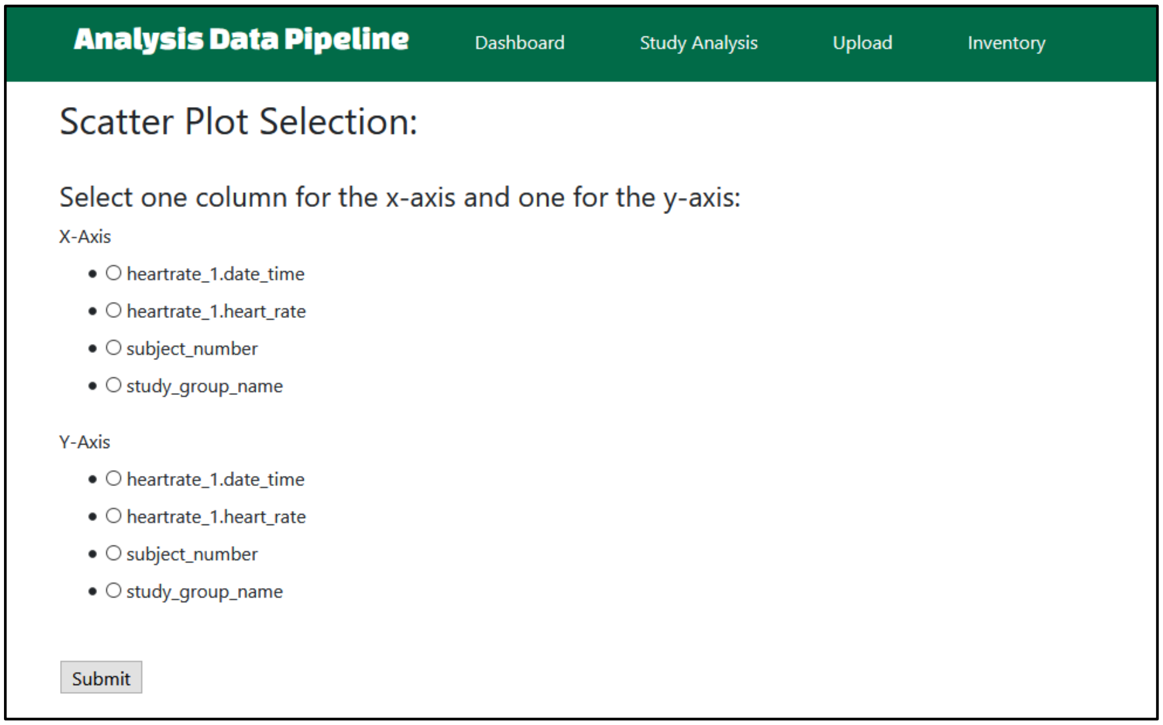


Histogram Result:

2.4.2: Scatter Plot

Clicking the “Make Scatter Plot” button will bring the user to a new page with a selection prompt. From here, the user will select one attribute from the “Select X-Column” boxes and one category from the “Select Y-Column” boxes.

Clicking the submit button will redirect to a new page that displays the plot created with the previous selections. Selecting more than one category from either list will result in the figure output being the category selected highest in the list.



Scatter Plot Selection:

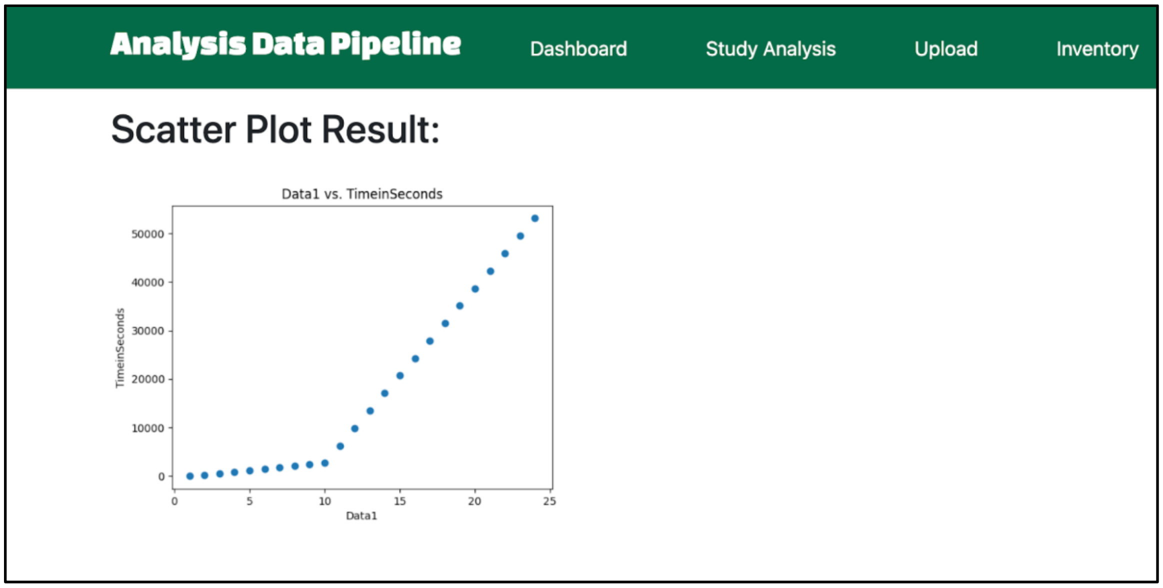
Select one column for the x-axis and one for the y-axis:

X-Axis

- ○ heartrate_1.date_time
- ○ heartrate_1.heart_rate
- ○ subject_number
- ○ study_group_name

Y-Axis

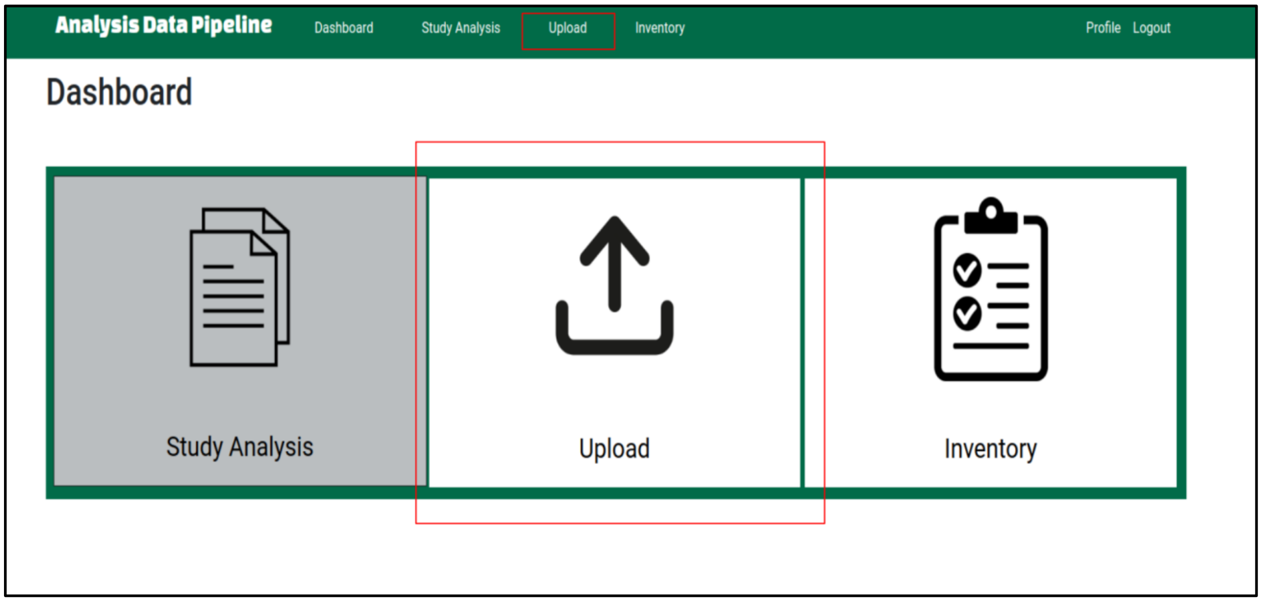
- ○ heartrate_1.date_time
- ○ heartrate_1.heart_rate
- ○ subject_number
- ○ study_group_name



Scatter Plot Result:

3. Upload

The Upload feature of the pipeline allows you to upload data to the application. You can access this feature by clicking “Upload” on the Dashboard or the navigation bar.



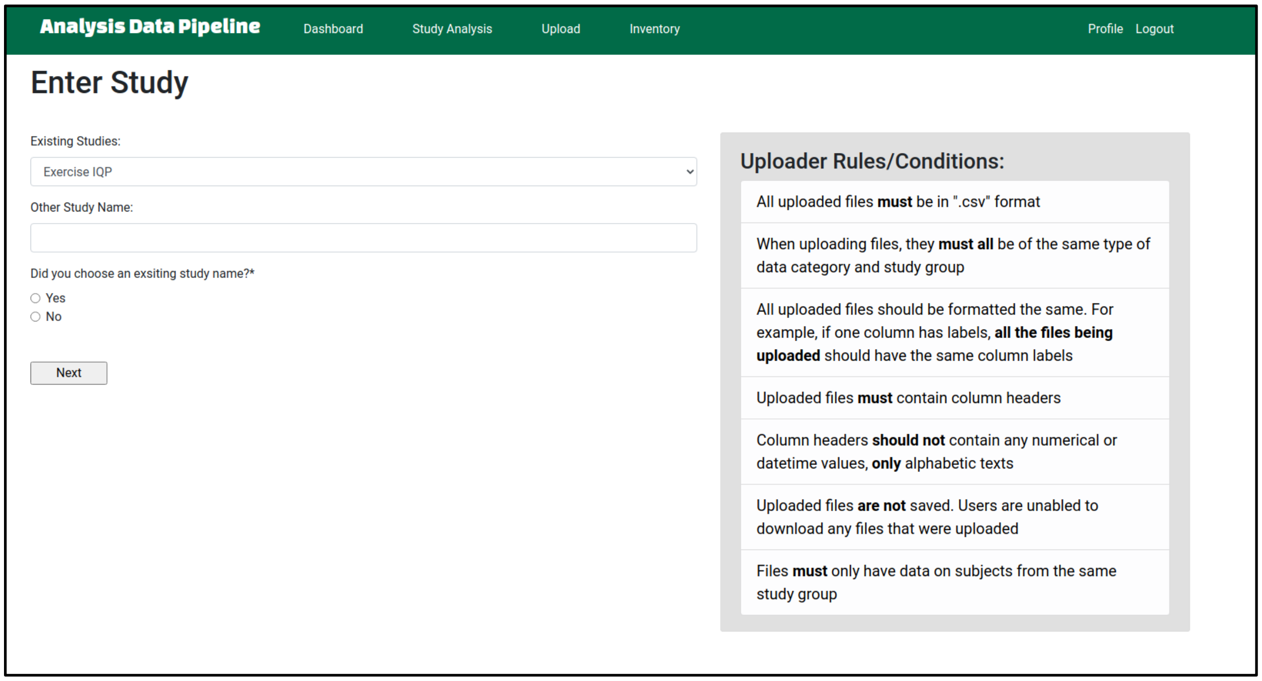
3.1: Study Name

On this page, the user is prompted to provide the name of the study that they want to upload data to. If there are currently existing studies within the database, the user will have the option to select one of those studies using the drop down menu. It is important that the user selects the 'Yes' option when asked if they chose an existing study name.

If there are no existing studies in the database, 'Existing Studies' drop down will not exist, and the user will be asked to enter in the name of the study.

The page includes a small menu that displays the important uploader rules/conditions to follow in order to use this pipeline's uploader feature. At the bottom of the menu, the user can download a pdf version of the rules/conditions.

By clicking on the 'Next' button, the user will be able to proceed forward in the uploader process.



Enter Study

Existing Studies: Exercise IQP

Other Study Name:

Did you choose an exsiting study name?*

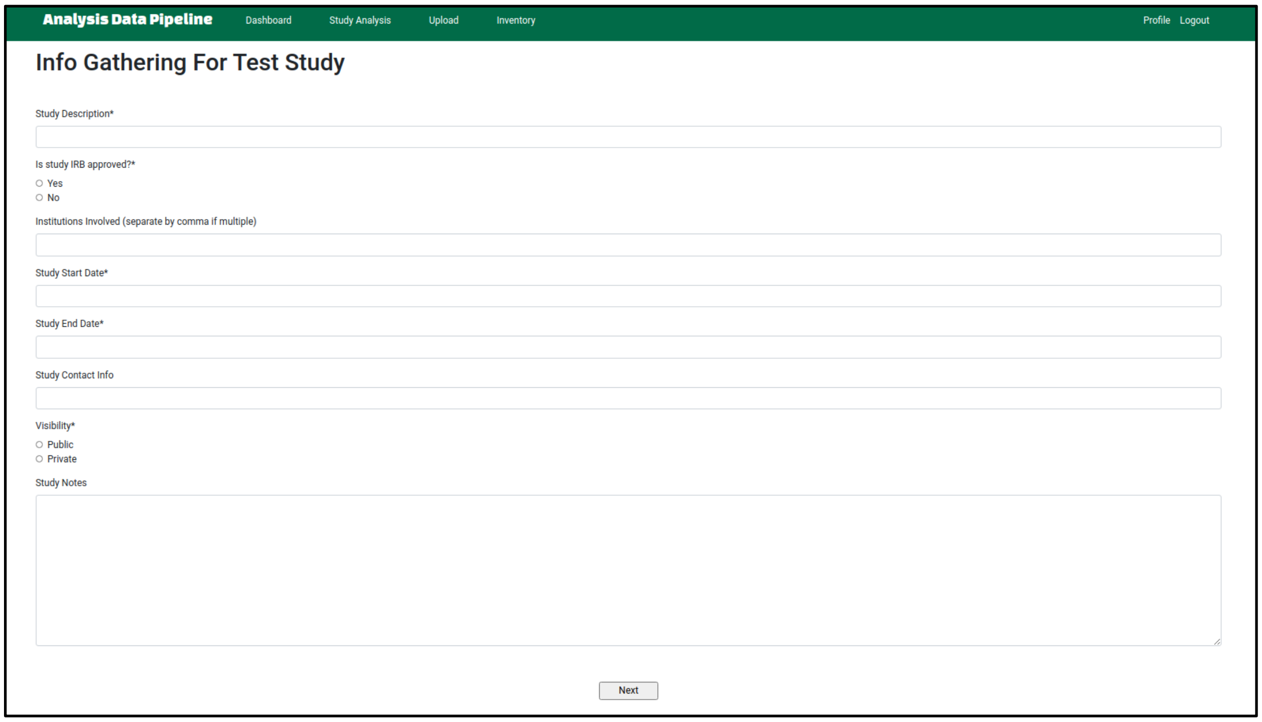
- Yes
- No

Uploader Rules/Conditions:

- All uploaded files **must** be in ".csv" format
- When uploading files, they **must all** be of the same type of data category and study group
- All uploaded files should be formatted the same. For example, if one column has labels, **all the files being uploaded** should have the same column labels
- Uploaded files **must** contain column headers
- Column headers **should not** contain any numerical or datetime values, **only** alphabetic texts
- Uploaded files **are not** saved. Users are unabled to download any files that were uploaded
- Files **must** only have data on subjects from the same study group

3.1.1: Study Info

If the user reached this page, the system has detected that the user is adding a new study to the database. Because a new study is being added to the database, the user is required to provide metadata for that study so that other users of the system will be able to understand what this new study is.



Info Gathering For Test Study

Study Description*

Is study IRB approved?*
○ Yes
○ No

Institutions Involved (separate by comma if multiple)

Study Start Date*

Study End Date*

Study Contact Info

Visibility*
○ Public
○ Private

Study Notes

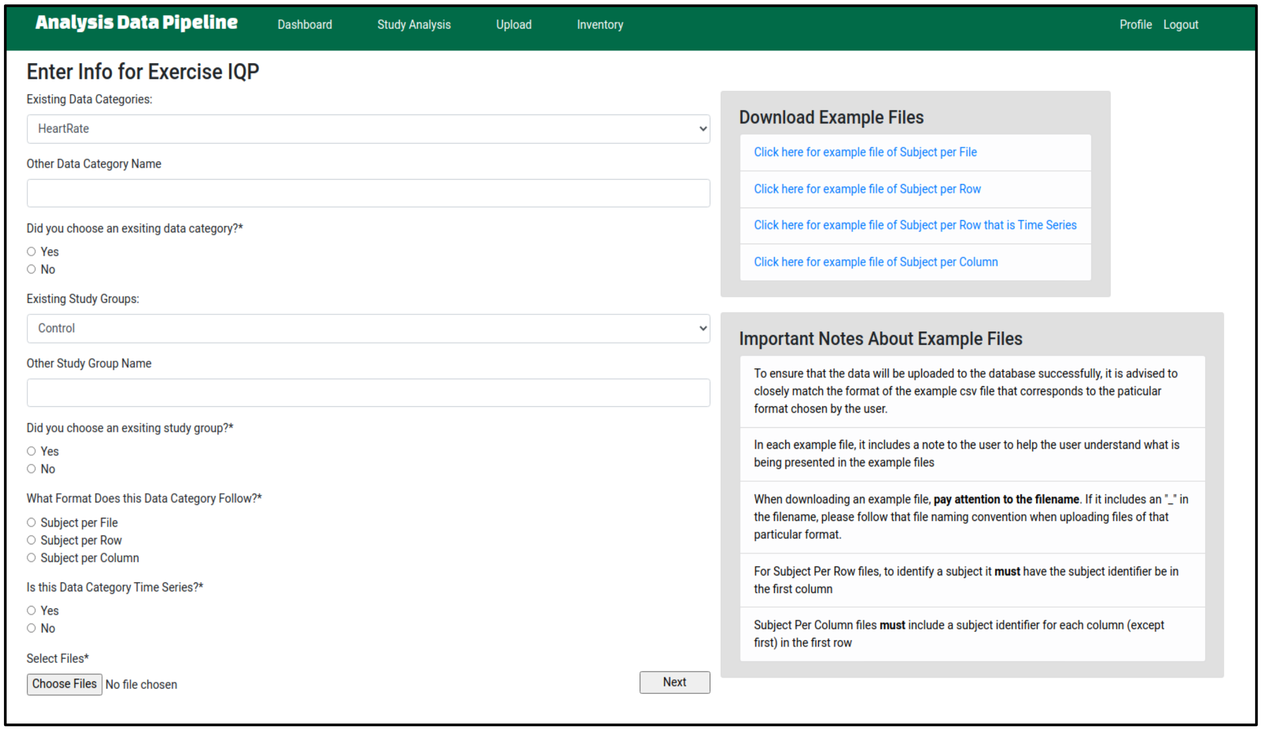
3.2: Info

Data Category: The type of data that was collected in the study. For example, a study exploring the relationship between exercise and memory would have several data categories, like heart rate and cognitive test results.

Study Group: The groups that subjects were split into for a study, such as control and experimental or several sample groups.

On this page, the user will provide information about which data category (if it exists) and study group (if it exists) they want to upload data to. If the option exists, the user should select 'Yes' if they chose an existing data category and/or an existing study group.

If there are existing data categories and/or study groups for the specified study, there will be a drop down box that will allow the user to select any of the existing data categories/study groups. In the image below, it indicates that HeartRate, Corsi, Flanker are an existing data category for the 'Exercise IQP' study.



Enter Info for Exercise IQP

Existing Data Categories: HeartRate

Other Data Category Name

Did you choose an exsiting data category?*
○ Yes
○ No

Existing Study Groups: Control

Other Study Group Name

Did you choose an exsiting study group?*
○ Yes
○ No

What Format Does this Data Category Follow?*
○ Subject per File
○ Subject per Row
○ Subject per Column

Is this Data Category Time Series?*
○ Yes
○ No

Select Files*
Choose Files No file chosen
Next

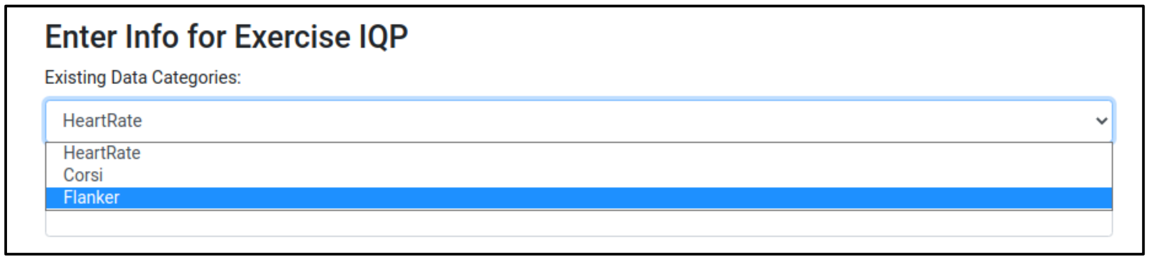
Download Example Files

- Click here for example file of Subject per File
- Click here for example file of Subject per Row
- Click here for example file of Subject per Row that is Time Series
- Click here for example file of Subject per Column

Important Notes About Example Files

- To ensure that the data will be uploaded to the database successfully, it is advised to closely match the format of the example csv file that corresponds to the paticular format chosen by the user.
- In each example file, it includes a note to the user to help the user understand what is being presented in the example files
- When downloading an example file, **pay attention to the filename**. If it includes an "_" in the filename, please follow that file naming convention when uploading files of that particular format.
- For Subject Per Row files, to identify a subject it **must** have the subject identifier be in the first column
- Subject Per Column files **must** include a subject identifier for each column (except first) in the first row

The user is also prompted to provide information about the data they are uploading such as its format and if it is a time series data. Once all the information has been filled out, the user should upload the “.csv” files.


Enter Info for Exercise IQP

Existing Data Categories:

HeartRate

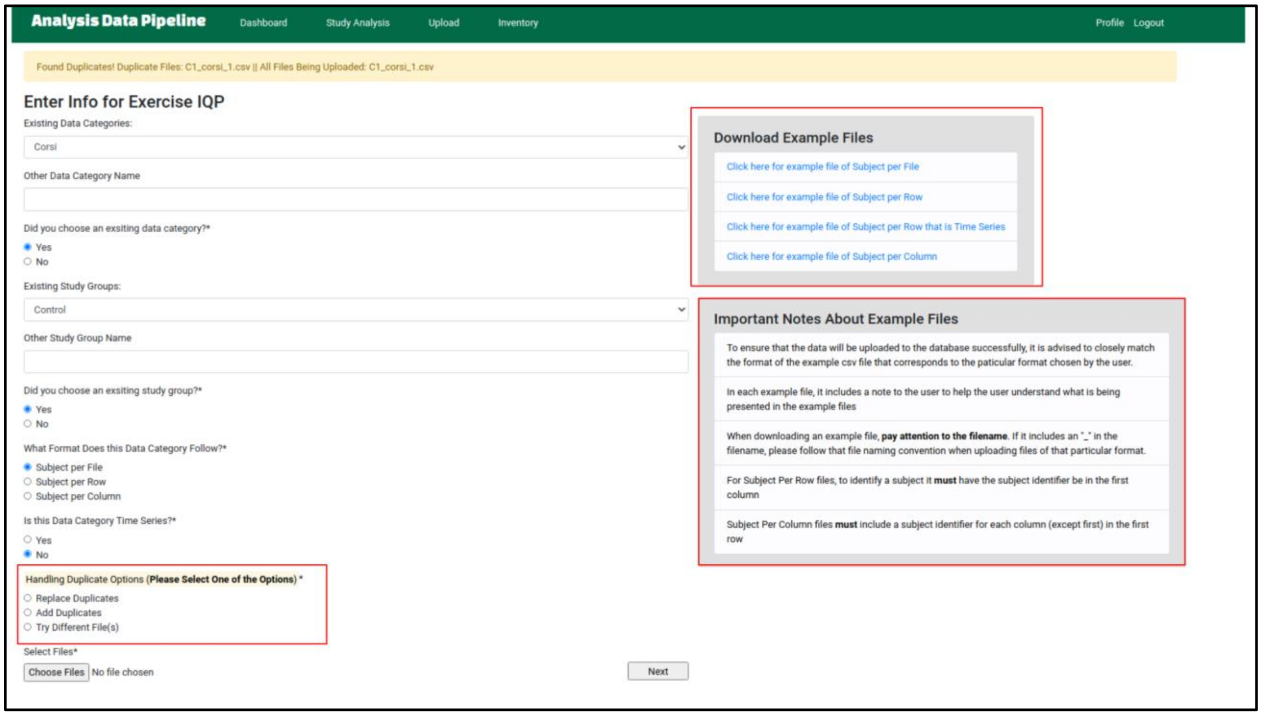
HeartRate
Corsi
Flanker

3.2.1: Info - Important Features to Lookout For

- Download Example Files
    - In case the user is unsure of how to structure their .csv file, they can download an example file of the corresponding format that they are interested in uploading

- Important Notes About Example Files
    - Important notes to help user understand how to format their uploaded files

- Duplicate File Handler
    - Upon detection of duplicate files, the page will include an additional prompt to ask the user on how they want to handle uploading duplicate files
        - ***Replace Duplicates***
            - This would remove all the data that was initially uploaded from the previous version of the duplicate file and replace them with the data from the current uploaded file
        - ***Add Duplicates***
            - This would simply just add the data without doing anything to the database. This potentially adds the risk of having duplicate data within the database table
        - ***Try Different File(s)***
            - This option allows the user to change the name of the detected duplicate file and have the user try uploading a newly named file. If the newly named file is a duplicate of an existing file, the user will simply be prompted again and repeat the process



Found Duplicates! Duplicate Files: C1_corsi_1.csv || All Files Being Uploaded: C1_corsi_1.csv

Enter Info for Exercise IQP

Existing Data Categories: Corsi

Other Data Category Name

Did you choose an exsiting data category?*
[✓ Yes] No

Existing Study Groups: Control

Other Study Group Name

Did you choose an exsiting study group?*
[✓ Yes] No

What Format Does this Data Category Follow?*
[✓ Subject per File] Subject per Row Subject per Column

Is this Data Category Time Series?*
Yes [✓ No]

Handling Duplicate Options (**Please Select One of the Options**) *
Replace Duplicates
Add Duplicates
Try Different File(s)

Select Files*
Choose Files No file chosen

Next

Download Example Files

Click here for example file of Subject per File
Click here for example file of Subject per Row
Click here for example file of Subject per Row that is Time Series
Click here for example file of Subject per Column

Important Notes About Example Files

To ensure that the data will be uploaded to the database successfully, it is advised to closely match the format of the example csv file that corresponds to the paticular format chosen by the user.

In each example file, it includes a note to the user to help the user understand what is being presented in the example files

When downloading an example file, **pay attention to the filename**. If it includes an "_" in the filename, please follow that file naming convention when uploading files of that particular format.

For Subject Per Row files, to identify a subject it **must** have the subject identifier be in the first column

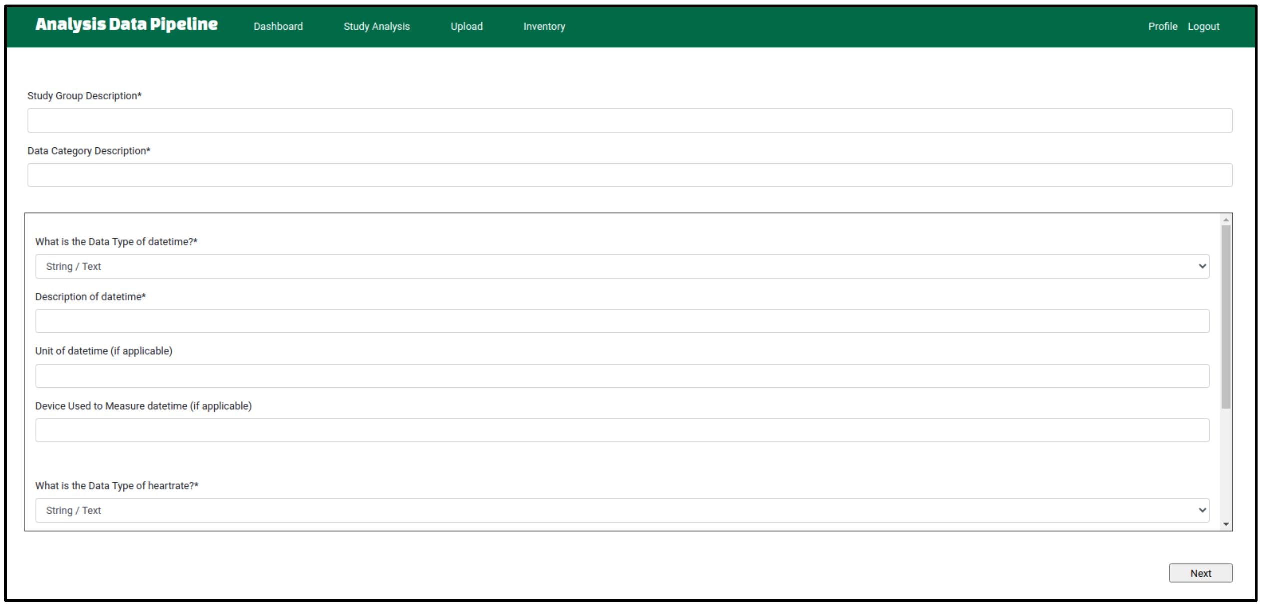
Subject Per Column files **must** include a subject identifier for each column (except first) in the first row

3.3: Extra Info

Attribute: The specific fields collected for each data category. For example, a study collecting heart rate data (data category) would collect data for attributes like the heart rate in bpm, as well as what time that reading was taken. Often, these are the columns of the data table.

If the user reached this page, it means that they are attempting to upload a new set of data to the database. This page asks the user all the necessary information it needs from the user based on the information that was provided from the Info page.

The purpose of this page is to acquire the metadata of the new data category to be added and/or study group name. Additionally, the user will be prompted to fill out the metadata for each of the detected column(s) / attribute(s) from the uploaded ".csv" file(s). By filling out the metadata, it helps other users of the system to be able to understand what other users have contributed to the system.



Study Group Description*

Data Category Description*

What is the Data Type of datetime?*

String / Text

Description of datetime*

Unit of datetime (if applicable)

Device Used to Measure datetime (if applicable)

What is the Data Type of heartrate?*

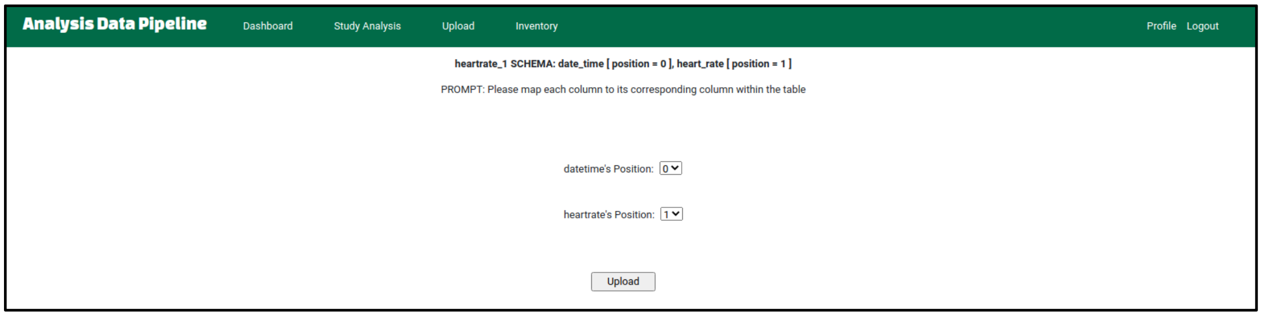
String / Text

3.4: Final Prompt

On this page, the user is prompted to map the columns that were detected from their uploaded files to the corresponding columns of the table that the data will be uploading to. Because the column positioning of the uploaded .csv files will not always match the column order in the database tables, this feature provides the user the ability to simply remap their columns without requiring them to make their own alterations to their files.

The user will use the drop down menu to select the correct positioning number. **Please note that the first column will have positioning of 0**.

Clicking the 'Upload' button, will direct the user to the uploader.



heartrate_1 SCHEMA: date_time [position = 0], heart_rate [position = 1]

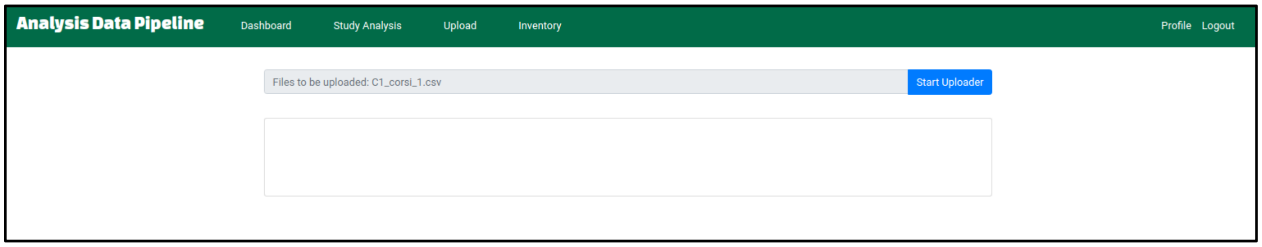
PROMPT: Please map each column to its corresponding column within the table

datetime's Position: 0

heartrate's Position: 1

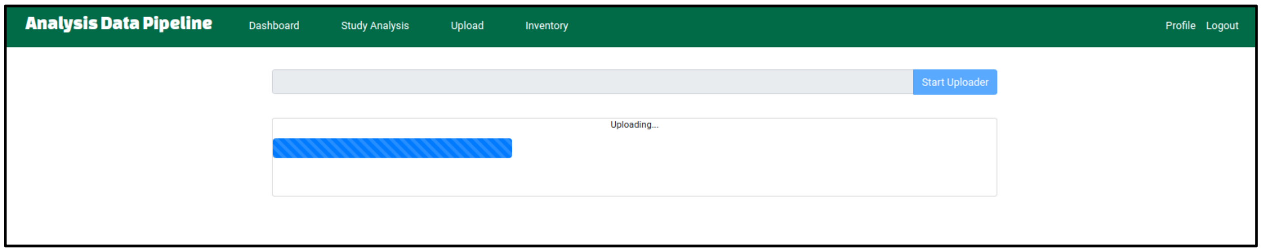
3.5: Uploader Page

At this page, this is where the uploading of the data to the database takes place. In the gray box, the user can view the files to be uploaded, and by pressing 'Start Uploader', it will begin the uploading process.



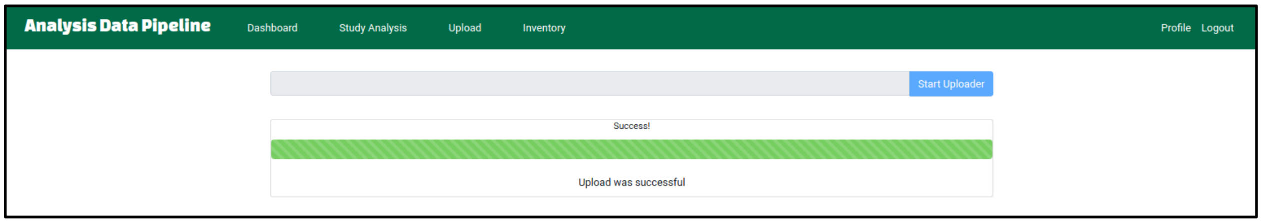
Files to be uploaded: C1_corsi_1.csv

When the bar is blue, it indicates that the system is currently uploading the data. If it hangs, do not worry, it will eventually finish.



3.5.1: Success

When the bar turns green from the uploader page, the green color indicates that the uploading process has completed and was successful. The user will soon be redirected to the successful upload page.



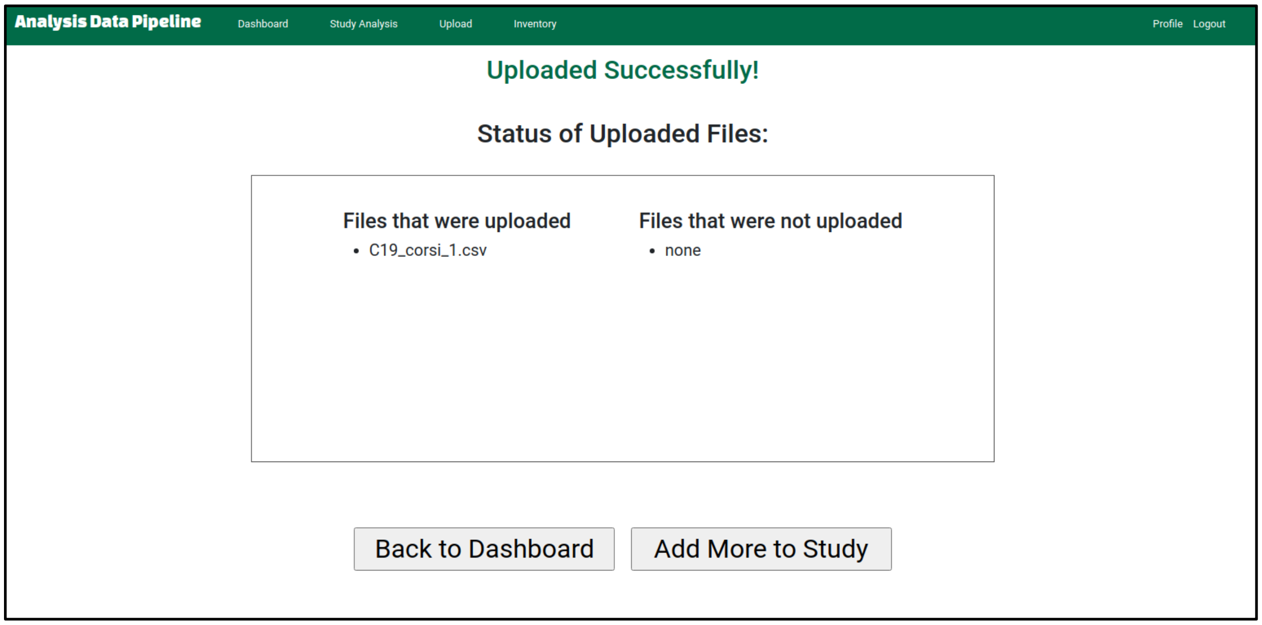
Success!

Upload was successful

The user is brought to this page when all the data was uploaded successfully. All the files that were uploaded from the Info page would fall under the ‘Files that were uploaded’ column.

If the user clicks on the ‘Add More to Study’ button, the user will be redirected to the Info page where they can continue to upload more data to the same study.

If the user clicks on the ‘Back to Dashboard’ button, the user will be directed back to the Dashboard.



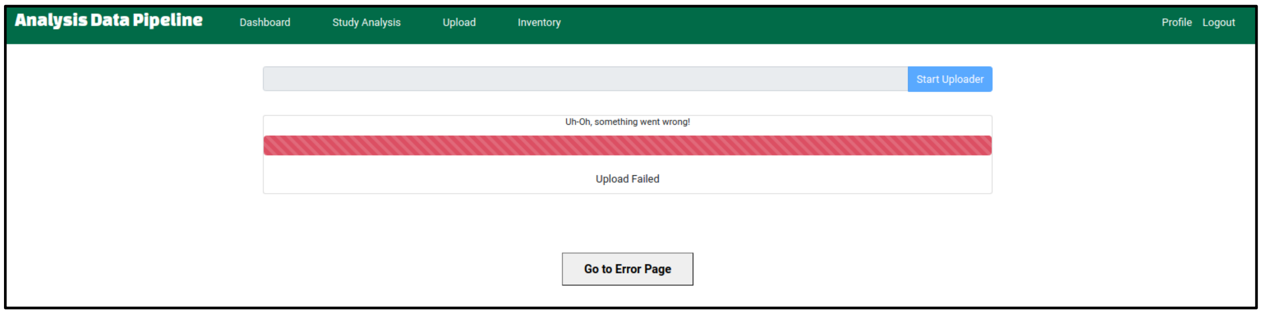
Uploaded Successfully!

Status of Uploaded Files:

Files that were uploaded	Files that were not uploaded
• C19_corsi_1.csv	• none

3.5.2: Error

When the bar turns red from the uploader page, it means an error has occurred during the uploading process. By clicking on the 'Go to Error Page' button, the user will see an error message and a list of all the files that were either uploaded or not uploaded successfully.

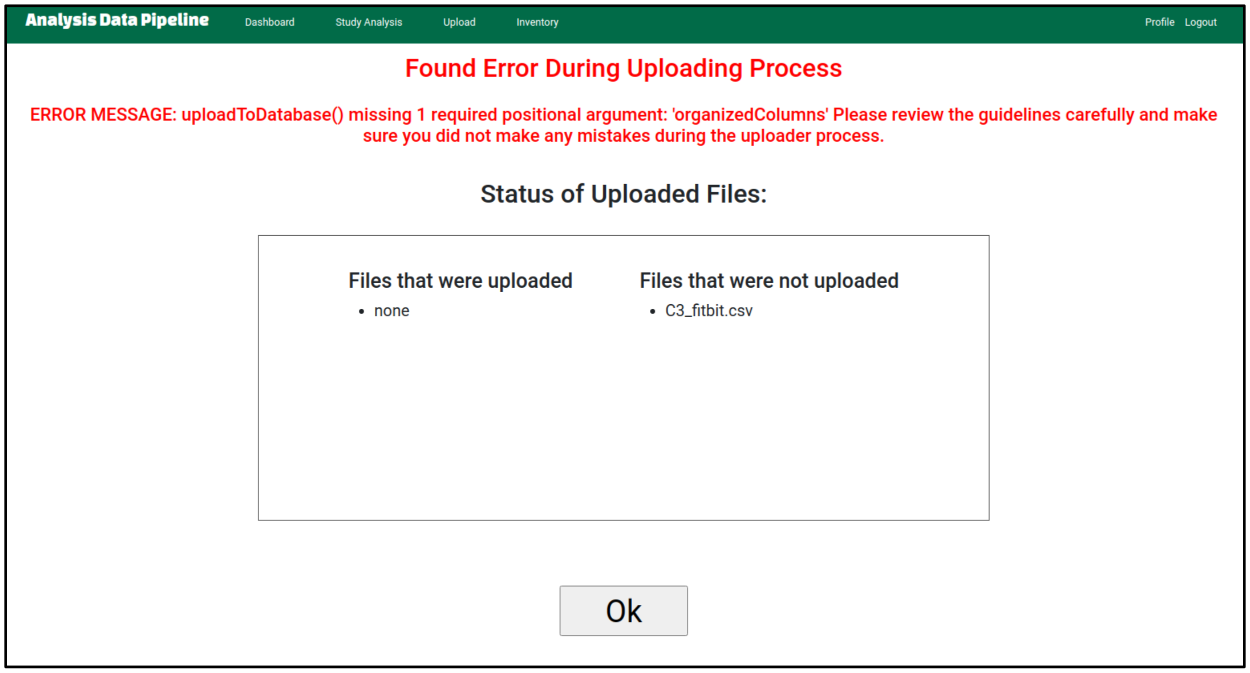


Uh-Oh, something went wrong!

Upload Failed

If there were any errors that occurred in the whole uploading process, the user will be redirected to this page, where they will be informed of the error that was found.

Similar to the Success page, this page also indicates which files were uploaded successfully to the database and which were not. By providing this information, the user will be able to learn which files they will have to try uploading again.



Found Error During Uploading Process

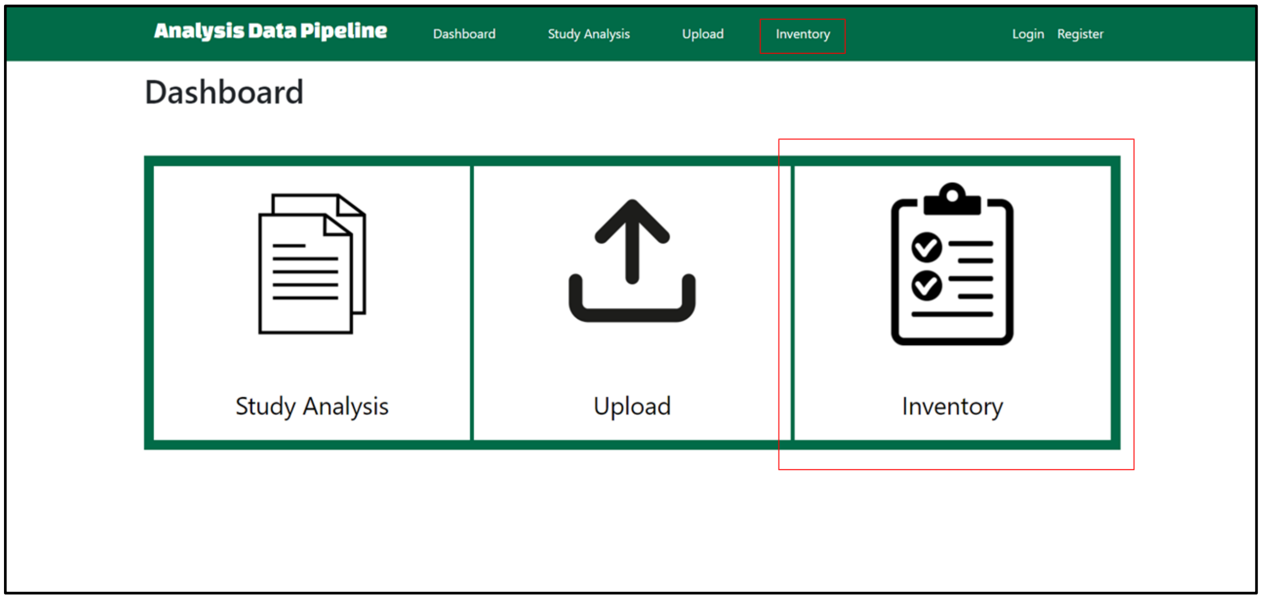
ERROR MESSAGE: uploadToDatabase() missing 1 required positional argument: 'organizedColumns' Please review the guidelines carefully and make sure you did not make any mistakes during the uploader process.

Status of Uploaded Files:

Files that were uploaded	Files that were not uploaded
• none	• C3_fitbit.csv

4. Inventory

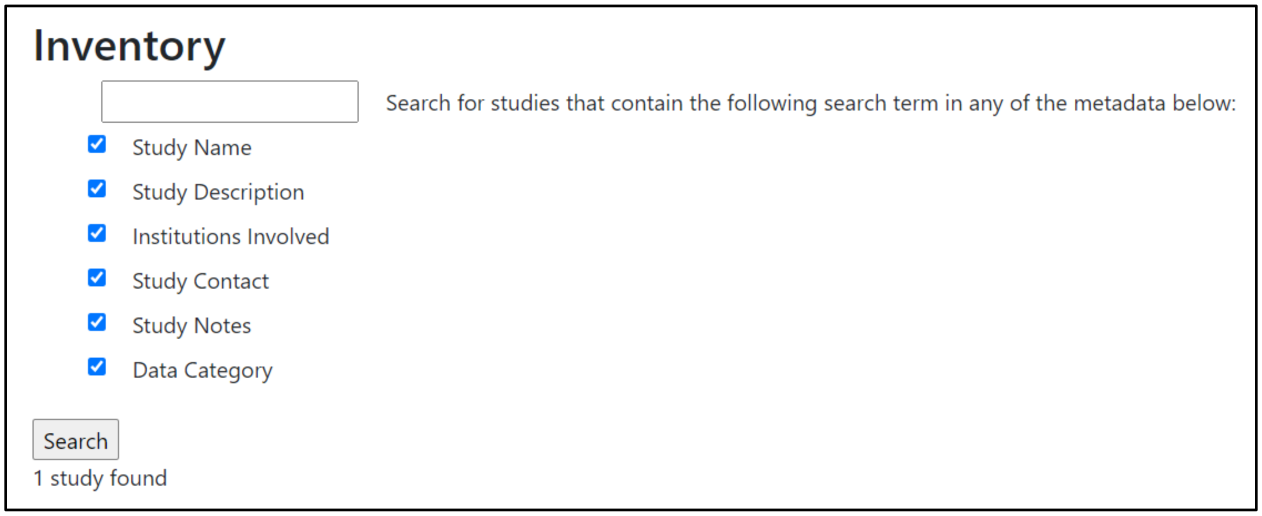
The Inventory feature of the pipeline allows you to search for studies that you have access to and view their metadata. You can access this feature by clicking "Inventory" on the Dashboard or nav bar.



4.1 View Study List

Data Category: The type of data that was collected in the study. For example, a study exploring the relationship between exercise and memory would have several data categories, like heart rate and cognitive test results.

This view is somewhat similar to the study selection page. It shows the studies you have permission to view with links to view more details about them. Additionally, there is a search bar at the top of the page that filters studies based on if their metadata contains a given search term. This metadata includes the study title, study description, institutions involved, study contact, study notes, and data category within the study.

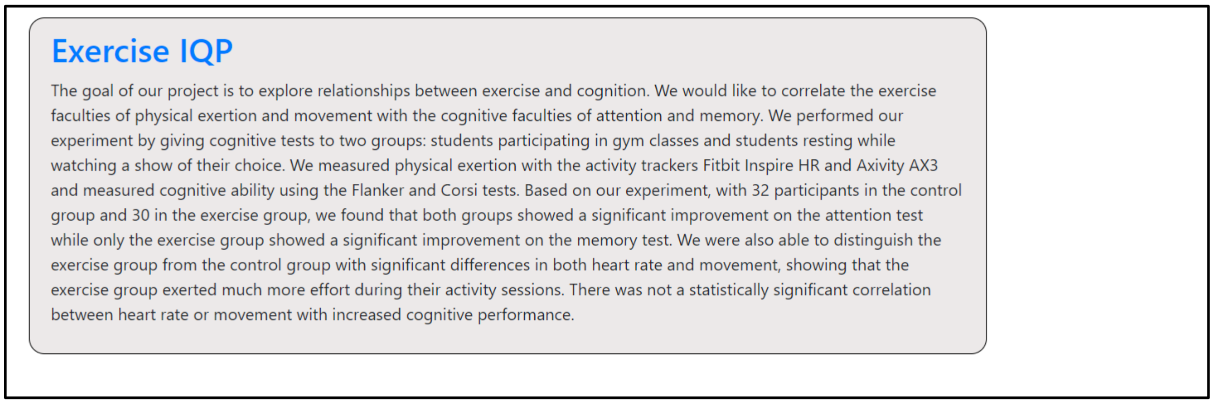


Inventory

Search for studies that contain the following search term in any of the metadata below:

- ✓ Study Name
- ✓ Study Description
- ✓ Institutions Involved
- ✓ Study Contact
- ✓ Study Notes
- ✓ Data Category

1 study found



Exercise IQP

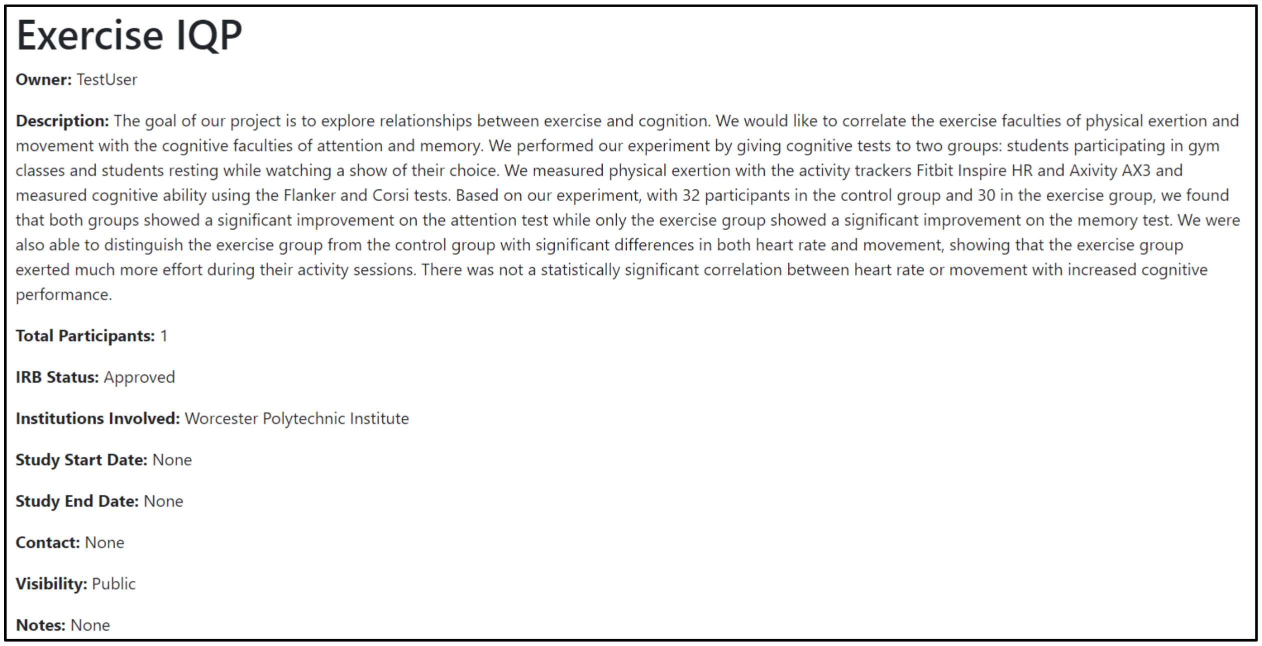
The goal of our project is to explore relationships between exercise and cognition. We would like to correlate the exercise faculties of physical exertion and movement with the cognitive faculties of attention and memory. We performed our experiment by giving cognitive tests to two groups: students participating in gym classes and students resting while watching a show of their choice. We measured physical exertion with the activity trackers Fitbit Inspire HR and Axivity AX3 and measured cognitive ability using the Flanker and Corsi tests. Based on our experiment, with 32 participants in the control group and 30 in the exercise group, we found that both groups showed a significant improvement on the attention test while only the exercise group showed a significant improvement on the memory test. We were also able to distinguish the exercise group from the control group with significant differences in both heart rate and movement, showing that the exercise group exerted much more effort during their activity sessions. There was not a statistically significant correlation between heart rate or movement with increased cognitive performance.

4.2 View Study Metadata

Study Group: The groups that subjects were split into for a study, such as control and experimental or several sample groups.

Attribute: The specific fields collected for each data category. For example, a study collecting heart rate data (data category) would collect data for attributes like the heart rate in bpm, as well as what time that reading was taken. Often, these are the columns of the data table.

This page shows a variety of information for a specific study. This includes the study's description, total participants, IRB status, institutions involved, start date, end date, contact, visibility, and notes. The username of the user that owns that study is displayed beneath the study name.



Exercise IQP

Owner: TestUser

Description: The goal of our project is to explore relationships between exercise and cognition. We would like to correlate the exercise faculties of physical exertion and movement with the cognitive faculties of attention and memory. We performed our experiment by giving cognitive tests to two groups: students participating in gym classes and students resting while watching a show of their choice. We measured physical exertion with the activity trackers Fitbit Inspire HR and Axivity AX3 and measured cognitive ability using the Flanker and Corsi tests. Based on our experiment, with 32 participants in the control group and 30 in the exercise group, we found that both groups showed a significant improvement on the attention test while only the exercise group showed a significant improvement on the memory test. We were also able to distinguish the exercise group from the control group with significant differences in both heart rate and movement, showing that the exercise group exerted much more effort during their activity sessions. There was not a statistically significant correlation between heart rate or movement with increased cognitive performance.

Total Participants: 1

IRB Status: Approved

Institutions Involved: Worcester Polytechnic Institute

Study Start Date: None

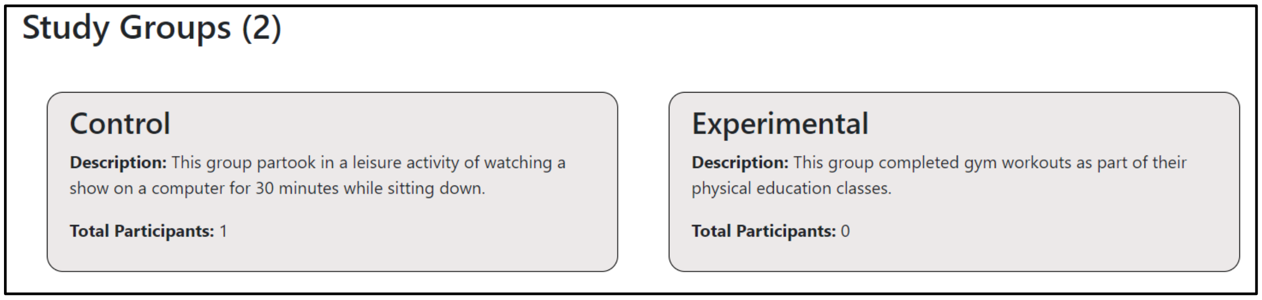
Study End Date: None

Contact: None

Visibility: Public

Notes: None

The study groups of the study are displayed on the page as well. This shows the description and total participants for each of the groups.



Study Groups (2)

Control

Description: This group partook in a leisure activity of watching a show on a computer for 30 minutes while sitting down.

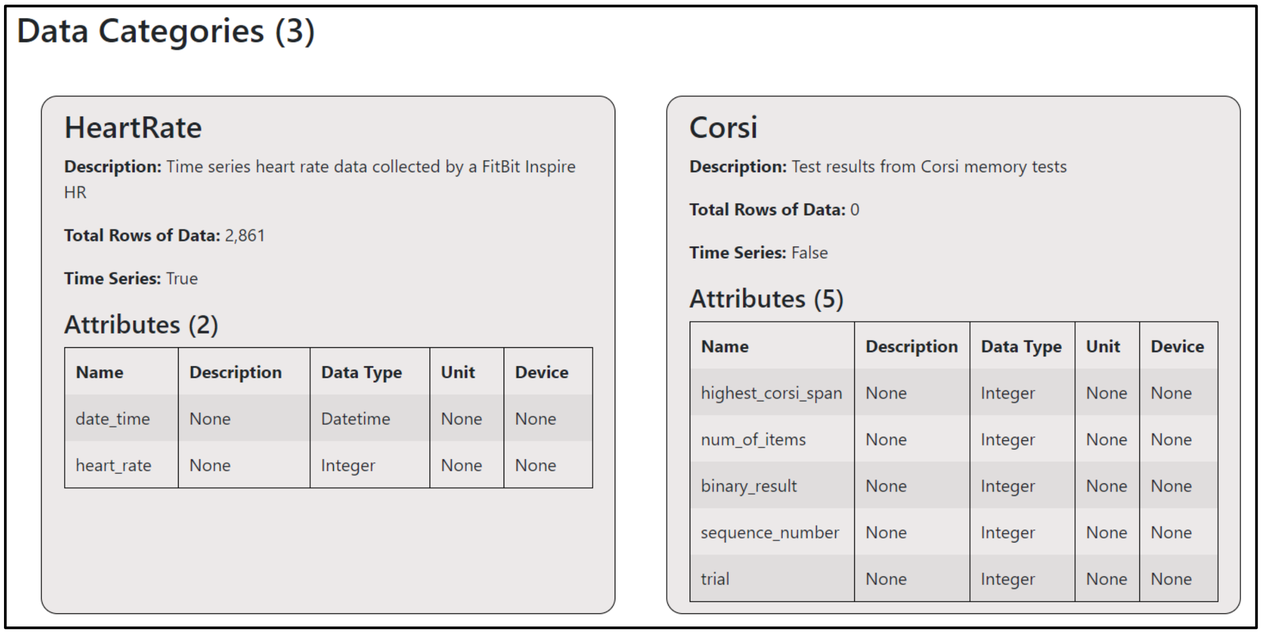
Total Participants: 1

Experimental

Description: This group completed gym workouts as part of their physical education classes.

Total Participants: 0

The data categories of the study are displayed on the page as well. This shows the description, total rows of data, whether it is a time series, and a table of the column attributes for each data category.



Data Categories (3)

HeartRate

Description: Time series heart rate data collected by a FitBit Inspire HR

Total Rows of Data: 2,861

Time Series: True

Attributes (2)

Name	Description	Data Type	Unit	Device
date_time	None	Datetime	None	None
heart_rate	None	Integer	None	None

Corsi

Description: Test results from Corsi memory tests

Total Rows of Data: 0

Time Series: False

Attributes (5)

Name	Description	Data Type	Unit	Device
highest_corsi_span	None	Integer	None	None
num_of_items	None	Integer	None	None
binary_result	None	Integer	None	None
sequence_number	None	Integer	None	None
trial	None	Integer	None	None

Appendix K. Evaluation Scripts

Introduction

Hello! My name is [facilitator's name] and this is [note taker's name], and thank you for participating in this user study for our MQP. As we mentioned in the qualtrics survey, we are a team of 6 CS and BME students working on a data analysis pipeline that is able to consume, store/organize, and analyze biomedical data. This project aims to provide a centralized application for storing and analyzing different types of data and allow for analysis across different studies.

The purpose of this user study is to help us understand how easy to use and efficient our system is, as well as how it compares to alternative systems. This will help inform future MQP groups on how to improve the application. This user study will take about 30 minutes, with some time for sharing feedback at the end. You will only be using a subset of the features available in this application. I will be leading this user study and _ will be taking notes. All data that we record from this will be kept anonymous.

During this evaluation, it would be helpful for us if you could think out loud as you go about completing these tasks so we can understand your thought process but please be clear if you are actually asking us questions directly. Please note, there is a help page attached to the website. Please refer to it if you have any confusion on the terminology and how to proceed forward.

Do you have any questions before we begin?

Group 1

Group 1 Task Overview:

1. **Create profile**
2. **Inventory Analysis**
3. **Select and analyze with scatter plot**

Note: For any troubles throughout the tasks, please reference the *Help Page*.

1. **Create Profile (U2)**
    - Your task is to register a profile.
    - Your username should be your WPI email and the password you create must satisfy the specified conditions.
    - After successfully registering, log in and proceed to the next task.

2. **Inventory (I1)**
    - Please view the inventory of all studies that contain the data category "HeartRate."
    - Select the "Test_Study_Data" study that is displayed to view the metadata of the study.

- Please answer the following questions after navigating to the metadata:
    - Is this study IRB approved?
    - What are the institutions involved?
    - How many study groups participated in this study?
    - How many of the data categories are time series data?

3. **Select and analyze with scatter plot (A3)**
    - Your task is to create a scatter plot of the Systolic Blood Pressure data vs. datetime
        - For all subjects
        - Study: Test_Study_Data
    - After generating the data summary, create the scatter plot.
    - Once the scatter plot has been generated, save the figure under the name "heartrate_datapoints" as a .png file.

Group 2

Group 2 Task Overview:

1. **Create profile**
2. **Upload a new study**
3. **Select and export**

Note: For any troubles during the evaluation, please reference the *Help Page*.

1. **Create Profile**
    - Your task is to register a profile.
    - Your username should be your WPI email and the password you create must satisfy the specified conditions.
    - After successfully registering, log in and proceed to the next task.

2. **Upload a new study:**
    - Use the following metadata to upload a new study called ECG Study that contains heart rate data collected from a control group and experimental group.
        - For this task, you will just upload the data for the control group.
    - **NOTE TO FACILITATOR**
        - Have the participants attempt to upload subject1data.csv
            - Can simply just provide the correct file name "subject1_data.csv" if participant is able to communicate the correct steps to make the fix

Study metadata

- Study name: ECG Study
- Study description: This is a test study.
- IRB status: Approved
- Institutions involved: WPI
- Start Date: November 1, 2019
- End Date: November 1, 2020
- Contact: gr-bmedatapipeline@wpi.edu
- Notes: None
- **Visibility: Private**

Study Group:

- Name: Control
- Description: Control group of the study.

Data Category:

- Name: ECG
- Description: Time series of various cardiovascular data.
- Format: Subject per file
- Is a time series
- Attributes (columns):
    - Datetime
        - Data type: Datetime
        - Description: Timestamp
    - HeartRate
        - Data type: Integer
        - Description: Number of heart beats per minute
    - Age
        - Data type: Integer
        - Description: Participant's age

Files: Subject1Data.csv

Closing

OK, great job on the tasks! With the time we have left, could you share some feedback with us about how easy this system was to use and understand?

OK, that's all for the user study. Thank you again for participating! We have a short post user study questionnaire that we would like you fill out on your own time.

You are in [group number]. We will send a follow up email if you win the raffle.

Appendix L. Pre-Survey Questions

Introduction

Hello! We are an MQP team of CS and BME students at WPI. We have designed and developed a data analysis pipeline that would consume, store/organize, and analyze physiological data. We are looking for participants to evaluate the usability and functionality of our application. This survey should take two minutes or less.

Thank you for your time and we can be contacted with any further questions at gr-bmedatapipeline@wpi.edu!

Questions

1.) Are you 18 years or older? (Yes/No)

If answered “no” to question 1:

Unfortunately, we cannot take participants under the age of 18. Thank you for your interest!

Please click "Next" to finish the survey.

If answered “yes” to question 1:

2.) Have you ever worked with large amounts of raw data? (Yes/No/Do not know)

If answered “yes” to question 2:

3.) In what setting have you worked with large amounts of raw data?
   - a.) WPI Courses (specify)
   - b.) Internship
   - c.) Research
   - d.) Other (specify)

4.) How did you store/organize this data?
   - a.) Spreadsheets
   - b.) Database
   - c.) Paper Records
   - d.) Other (specify)

5.) How did you analyze this data?
   - a.) Graphically (e.g. histograms, scatterplots, regression)

b.) Statistical Analysis (e.g. hypothesis testing, t-tests, ANOVA)
c.) Machine Learning (e.g. cluster analysis, feature extraction)
d.) Frequency Transforms (e.g. Foriour transforms)
e.) Other(specify)

For any answer to question 2:

6.) Which of the following tools do you have experience with?
a.) MATLAB
b.) Python
c.) JavaScript
d.) Microsoft Excel
e.) Tableau
f.) MiniTab
g.) RStudio
h.) SQL
i.) Microsoft Access
j.) Other (specify)

7.) Are you interested in participating in a 30-min virtual user study of our data analysis pipeline between February 8th and February 28th? All participants will have the opportunity to win a $60 Amazon gift card. (Yes/No)

If answered “yes” to question 7:

8.) First Name
9.) Last Name
10.) Email Address
11.) Major(s)
12.) Please select which of the following most closely applies to you:
a.) Undergraduate Student
b.) Graduate Student
c.) Professor
d.) Other (specify)

Appendix M. Post-Survey Questions

Introduction

Please fill out this form following the completion of your user study session. Thank you for your time and participation.

Questions

1.) WPI Email Address
2.) What group were you assigned in this study? Please ask your proctor if you do not know.
    a.) Group 1
    b.) Group 3

The following questions were rated on a scale from 1 to 5, 1 representing "strongly disagree" and 5 representing "strongly agree".

3.) The application was easy to use.
4.) The application was easy to navigate.
5.) The application had an attractive design.
6.) The application had a logical flow.
7.) The application would be useful in a research setting.

The following questions were free response questions.

8.) What did you like about the application?
9.) What didn't you like about the application?
10.) What do you like about other data handling tools you have experience with (for example: Excel, Python, SQL, MATLAB)?
11.) What do you not like about other data handling tools that you have experience with, if you have any?
12.) Do you have any other feedback that you would like to share with us?