Mobile Reporting Application

A Major Qualifying Project Report

Submitted to the faculty of the
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements of the
Degree of Bachelor of Science

By:

Jian Mao, Computer Science and Mathematical Science
Caitlin McMahon, Actuarial Mathematics
Patchara Santawisook, Mathematical Science

Project Sponsor: Barclays

Submitted to:
On-site Liaisons: Gregory Friel
Terrance Snyder

Project Advisors: Professor Jon Abraham, Department of Mathematics
Professor Micha Hofri, Department of Computer Science

Submitted on
January 3, 2013

Abstract

Barclays uses an external service platform, ServiceNow, to report and track application issues internally. The platform is available on work computers and BlackBerries. With the increase in personal smartphones, specifically iOS and Android devices, it is imperative to provide this platform on these devices as well. The team was tasked with creating a live prototype application to fill this void. The outcome of the project was a prototype application that works on iOS devices and reports real-time incident data.

Acknowledgements

Our group would like to acknowledge all of the people that helped to provide an opportunity to take part in this project and the great success that resulted from it.

Foremost we would like to thank Mr. Gregory Friel, our sponsor, who made this project possible and oversaw the project from the Barclays side. His support provided us with a great opportunity to work for Barclays and apply the knowledge we have gained through our studies at Worcester Polytechnic Institute.

We also would like to thank Mr. Terrance Snyder, the project manager, who managed the day to day operations of the project. We truly appreciate the many hours of guidance and support he provided us during our time at Barclays. His assistance was especially vital to the completion of this project.

Moreover, we would like to thank Miles Dolphin for his guidance on ServiceNow and David Cheung for his assistance with integrating our application with the Barclays Live framework.

Additionally, we want to extend our gratitude towards Professor Arthur Gerstenfeld, Director of the Wall Street Project Center, for creating such an enriching experience. Along with Professor Gerstenfeld, we want to thank Professor Jon Abraham and Professor Micha Hofri, our faculty advisors, for their continued support, advice, and assistance throughout the project.

Lastly, we would like to thank Worcester Polytechnic Institute and Barclays for providing us with the opportunity to come to New York City and complete this project.

Without the assistance of these two organizations and all of these individuals, this project would not have been the success that it was. Thank you!

Authorship

This report was developed through a collaborative effort by all members of the project team: Jian Mao, Caitlin McMahon, and Patchara Santawisook. All sections were created and edited as a team, with equal contributions made by each member.

Executive Summary

Barclays strives for pioneering solutions for clients and employees and has become a leader in innovation. ServiceNow is a third party platform that Barclays currently uses to report the health of various systems. It not only provides employees with a convenient channel to view and report issues, but also automates and enhances the workflow in tracking issues by sending out daily status summary to management.

The ServiceNow platform provides a BlackBerry application that employees can use on the go. However, this application is not very user friendly, and Android and iOS mobile devices now hold a greater market share. Access to the application on an employee's personal device can greatly improve issue management.

The WPI project team was invited to Barclays to participate in the development of the mobile application of ServiceNow. The goal of the project was to develop a live Major Incident Handling prototype application capable running on various devices to improve the mobility of ServiceNow. The team divided the application structure into three stages: the data stage, the RESTful API stage, and the client stage. The data stage is to automatically extract data in XML format from the ServiceNow server and convert it to an interim data format (Java Object). The RESTful API stage is used to convert the data from the interim format to a light-weight format, (JSON), and then render it to display on the interface. The data stage and RESTful API stage form the backend server. The client stage is the user interface, which is used to send requests to the server and to display the data.

For the prototype application, the team created a design template which emulated the iOS mail application. The prototype application was designed to have two screens – list screen and detail screen. The list screen displays all incidents with partial details, including application name, service owner, status, and short description. The detail screen includes more information of each incident.

This project utilized various technologies. The backend program mainly used Java for development and Maven repository for version control. An XStream library was used for parsing data. A Dropwizard platform was set up for hosting server and rendering data from scratch. The creation of the interface heavily relied on HTML5, CSS3 and JavaScript.

At the end of the first phase, the prototype application worked independently on computers but was not yet connected to the external environment or server. A demo was given to the management and suggestions were collected. Based on the feedback from management, the team revised the prototype application accordingly. The project then moved to the second phase, which was a formal development and enhancement phase. The end goal was for the application to be well-integrated with Barclays' existing environment, including the Barclays Live framework.

In the second phase, the program was further developed in order to retrieve real-time data from the external server every five minutes. To better integrate with the Barclays Live framework, which was supported by the PhoneGap package, the team modified the client side to follow Model-View-Controller architecture. JavaScript libraries such as Backbone.js, RequireJS, and jQuery were applied. In addition, the team used "user agent" (a unique way of identifying the mobile device being used) to apply different templates and style sheets so that the application would match with different iOS mobile devices. Finally, the application was integrated with Barclays Live framework. The prototype application worked well with iPhone and iPad.

This project was in part of a research and development effort. Therefore, new technologies that were not used by Barclays were employed, which may be of interest for future Barclays' projects. Furthermore, many features could be implemented to support this application.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

Barclays is a major global banking and financial services company headquartered in the United Kingdom. By following their guiding principles, Barclays has become a leader in innovation, improving processes for clients and employees.

Currently, Barclays uses a third party software platform, ServiceNow, to report on the health of various systems. Employees can report errors and issues through this platform. These requests are then sent to the "right" people automatically. In addition, the system can send a weekly report regarding system status to management. ServiceNow not only provides employees with a convenient channel to report issues, but also automates and enhances the workflow in tracking issues. The ServiceNow platform currently provides a BlackBerry application that employees can use on the go. However, this application is not particularly user friendly, and Android and iOS mobile devices now hold a greater market share. The accessibility to the application on an employee's personal device can greatly improve issue management.

In order to improve the mobility of ServiceNow, the goal of this project was to develop a mobile reporting application that can display Major Incident Handlings (MIH) on various mobile devices. To attain this goal, the team followed a modified scrum approach and completed the following five objectives:

1. collect requirements for the mobile application;
2. conduct research on application development, such as helpful tools and software;
3. design and implement the application;
4. evaluate and enhance the application; and
5. recommend future improvements.

To fulfill these objectives, the team conducted research and held meetings. The information collected helped to develop the guidelines to create a user-friendly mobile application. Based on the design guidelines, and using the iOS mail application as inspiration, the team designed two screens for the application. The first screen is the list screen displaying all incidents with partial details, including application name, service owner, status, and short description. The second screen, the detail screen, includes more information about each incident.

The application had three stages of implementation, including the data stage, the RESTful API stage, and the client stage. The data stage is to automatically extract data from the

ServiceNow server and convert it to an interim data format. The RESTful API stage is to convert the data from the interim format to a light-weight format and render it to display on the interface. The data stage and RESTful API stage form the backend server. The client stage is the user interface, which is used to send requests to the server and to display the data.

To build the prototype application, the project consisted of two phases – prototyping and enhancement. At the prototyping phase, the team designed, set up and implemented the prototype application. Major components of implementation were explored separately. At the end of this phase, the prototype worked independently on computers, but was not yet connected to the ServiceNow server. A demo was given to management and suggestions were collected. Based on this feedback, the team revised the application. The project then moved to the second phase, which was a formal development and enhancement phase. The application could communicate with external servers and retrieve real-time ServiceNow data. It was successfully integrated with the Barclays Live framework.

The team also provided recommendations for future improvements for both the application and the programming process.

Chapter 2. Background

This chapter presents the background information regarding the company, Barclays, and its target application, ServiceNow. The key data attributes, candidate tools, and technologies that were used to build the application are reviewed.

2.1. Barclays

Barclays is a major global banking and financial services company headquartered in London. With operations in over 50 countries, the asset size of Barclays ranks it the 7th largest in the world and 2nd largest in the United Kingdom(Global Finance Magazine, 2012). Around the world Barclays focuses on five guiding principles -- winning together, retaining the best people, creating trust, focusing on the client, and pioneering solutions for clients. Following these principles, Barclays has demonstrated a history of innovation and leadership though notable achievements including: introducing the world to the automated teller machine (ATM), becoming the first foreign bank to file with the Securities Exchange Commission (SEC) in the United States, and recently, partnering with five international banks to launch the first global ATM alliance (Barclays, 2012). In addition to the innovations Barclays has introduced to their clients, Barclays creates and implements processes to improve functionality for their employees.

2.2. ServiceNow

Barclays currently uses ServiceNow, a leading Software-as-a-Service (SaaS) provider of enterprise Information Technology (IT) operations, to report the 'health' of various IT systems. ServiceNow provides uniquely designed solutions for each client company based on its needs. Service options include IT Asset Management, Change Management, and Incident and Problem Management (ServiceNow, 2012).

Barclays utilizes ServiceNow's Incident and Problem Management system. The purpose of this system is to enhance communication by providing one location to report IT-related systems issues and route notification of issues to the correct employees. Employees can access this system using their work computer. A sample screen shot of this web-based system can be seen in Figure 1.

Incidents New Go to Number

All > Active = true

Number	Category	Priority	Incident state	Short description
INC0000002	Software	4 - Low	Awaiting Problem	Can't get to network file shares
INC0000003	Network	4 - Low	New	Wireless access not available on floor 3
INC0000005	Software	2 - High	Active	CPU load high for over 10 minutes
INC0000007	Database	4 - Low	Active	Need access to sales db for the west
INC0000014	Hardware	4 - Low	New	missing my home directory
INC0000015	Software	4 - Low	New	I can't launch my game anymore
INC0000016	Software	4 - Low	New	Rain is leaking on main DNS Server
INC0000017	Hardware	4 - Low	New	How do I create a sub-folder
INC0000018	Hardware	1 - Critical	New	Sales forecast spreadsheet is READ O
INC0000019	Software	2 - High	New	Can't launch X-Win32
INC0000020	Inquiry / Help	4 - Low	New	Request for a Blackberry
INC0000025	Software	4 - Low	New	I need more memory
INC0000027	Software	2 - High	New	please remove this hotfix
INC0000029	Inquiry / Help	1 - Critical	New	I cant get my weather report
INC0000031	Hardware	1 - Critical	New	EMAIL Server Down
INC0000037	Hardware	3 - Moderate	New	Request for a new service
INC0000039	Network	4 - Low	New	Routing to oregon mail server
INC0000040	Inquiry / Help	3 - Moderate	Active	Javascript error

Figure 1: Sample ServiceNow incident screen

In addition to the web-based tool, Barclays employees can also utilize ServiceNow's Blackberry application to review and address IT systems issues. Barclays' goal was to create an application enabling employees to access incident information on other personal devices, particularly iPhones and iPads.

In Barclays' system, each incident has up to 200 pieces of information such as the incident number, application affected, and urgency. The pieces of information that are relevant for the team's application are:

- incident number, unique identification number for each incident for client use;
- system id, unique identification number for each incident for system use;
- initial escalation, time and date when the incident was opened/escalated;
- next update due, time and date when the next update is due;
- last commented, time and date when the incident was last updated/commented;
- last comment statement, most recent statement about the incident status;
- incident owner, individual who reported the incident;
- business units impacted, division(s) of the business impacted by the incident;
- application, the application affected by the incident;
- operational status, indicator showing the status in solving the incident; and
- short description, brief description of what the incident is.

In addition to the website, emails are sent daily to senior management to summarize the incidents that occurred or were fixed within the past 24 hours.

2.3. Technology Review

This section reviews technologies that the team used throughout the project to build the application.

2.3.1. Programming Languages

A variety of programming languages can be used in mobile application development. The most significant languages the team used are Java, HTML5, CSS3, and JavaScript.

2.3.1a. Java

Java is a general-purpose, object-oriented language. It is designed to provide a simple and efficient way to develop programs by having as few existing dependencies as possible. The Java Archive (JAR) format can be generated during the program compilation, which makes the program more portable. Currently, this language is ranked second among programming languages in popularity (Finley, 2012).

2.3.1b. HTML5

HTML5 is a markup language that was created by the World Wide Web Consortium and the Web Hypertext Application Technology Working Group (w3schools, 2012). This language has already become one of the web standards. Its combination with other standards such as Cascading Style Sheets (CSS) and JavaScript can deliver abundant functions, including Web Typography, Web Gallery, Photo Transitions, Audio, VR, Pixel Manipulation (Apple Inc., 2012). This language is necessary in developing hybrid applications, which leverage the device's browser engine to deliver the functions through processing HTML and JavaScript locally (Seven, 2012).

2.3.1c. CSS3

Cascading Style Sheets (CSS) is a simple style sheet language that defines how to display HTML elements such as background color, spacing, and fonts. Prior to the invention of CSS, styling for a webpage was added to each element within the page. CSS allows one document to be created and called upon to apply one style to all elements within the page. This allows for a quicker process to both load and make changes to the webpage (W3Schools, 2012).

2.3.1d. JavaScript

JavaScript is a prototype-based scripting language, which is primarily used for the client side in order to create enhanced user interfaces and dynamic behavior on an application. In mobile application development, JavaScript allows the user to display data, show and hide the navigation menus, and apply style sheets to the existing HTML (Stark, 2010).

2.3.2. Data Formats

The application involved using multiple data formats to receive, parse, and transmit the data. The data formats the team used were Extensible Markup Language (XML), Java Object, and JavaScript Object Notation (JSON).

2.3.2a. XML

XML is a markup definition language that was defined by a World Wide Web Consortium recommendation. XML is designed to be self-descriptive by using tags. These tags are not predefined, allowing users to name tags based on the data being stored. Additionally, XML can be used for data exchange. Typically exchanging data between incompatible systems over the Internet is a time-consuming challenge for developers. However, XML improves this process because it is stored in text format, which can be read by different incompatible systems(Liu, 2009).

2.3.2b. Java Object

In Java, an object is the instance of a class and can be used to store data. The class is a template that defines the structure of the object (member elements, methods). A class can contain more than one object (Samanta, 2004).

2.3.2c. JSON

JSON is a lightweight text-data interchange format used for storing and exchanging text information. It is readable and writeable by both humans and machines. JSON is built on two structures – a collection (object) and a list (array). JSON is advantageous because JSON documents are much smaller in terms of data size than the equivalent XML documents(Allan, 2010).

2.3.3. Libraries and Software

Programmers have created multiple tools and libraries to make the programming process more efficient. These libraries can be used by other programmers to simplify development processes.

2.3.3a. Dropwizard

Dropwizard is a Java framework that incorporates multiple Java libraries and provides RESTful web services. Dropwizard Core is the main package that offers various services such as server hosting, JSON processing, HTTP handling (Hale, 2012).

2.3.3b. jQuery

jQuery is an open-source JavaScript library for building web applications. Each web browser handles JavaScript in different ways, making it difficult for developers to create cross-browser compatible websites. jQuery simplifies this process by creating a common set of functions across all browsers(Holzner, 2009).

2.3.3c. jQuery Mobile

jQuery Mobile is a unified, HTML5 based user interface system for all mobile device platforms. Similar to how jQuery created a single set of functions for all browsers, jQuery Mobile has created a simplified process to make mobile applications work on multiple platforms(Reid, 2011). It is built on a jQuery and jQuery UI foundation and allows the developer to create a website or application that is able to work on all popular smartphones and tablets. It has a simple to use design and is used by hundreds of popular organizations' websites and applications, including OpenTable, Ikea, Disney World and Chase (The jQuery Foundation, 2012).

2.3.3d. Backbone.js & Underscore.js

Backbone.js is a client-side framework written in JavaScript that helps users to write highly responsive client-side applications(Bates, 2012). It has three important components – model, collection, and view. The model class provides a basic set of functionalities for managing changes. The collection class contains functions for handling and working with the data. The view class listens to changes in the associated model, and can be updated when the model changes. These three classes are connected over a RESTful and JSON interface(Ashkenas, 2012).

Underscore.js is an open-source JavaScript utility library that contains plenty of functional programming support. It is usually used to support *Backbone.js*, especially in the collection class.

2.3.3e. RequireJS

When an application runs, the load of the program each time can hamper the application's usability. RequireJS is a JavaScript file and a module loader that lessens this problem. RequireJS is able to manage script modules, load them in the right order and can spread out the download size over time(Burke, 2012). It can be used in conjunction with jQuery, thus making the process even simpler.

2.3.3f. XStream

XStream is an open-source Java library for converting Java Objects to and from XML (Fitzgerald, 2004). This library has a relatively high speed and uses little memory in the process, which fits large data source conversion (XStream, 2012). For this project, ServiceNow data was exported in XML format from external server, deserialized to Java Objects using XStream, and then converted to JSON.

2.3.3g. Apache HttpClient

Apache HttpClient is an open-source library to handle HTTP requests. With Apache HttpClient, users can access online resources and retrieve or send data via HTTP (Kalnichevski, 2008).

2.3.3h. Eclipse

The team used Eclipse, a multi-language software development environment, to develop the application. It is both a workspace and an extensible plug-in system. It is a typical platform for Java development (Eclipse Foundation, 2012).

2.3.3i. Apache Maven

Apache Maven is a software project management. This tool can help users to build and manage Java-based project. Maven project uses project object model (POM) to define its configuration, based on which Apache Maven can build the project (Apache Software Foundation, 2012). This tool not only provides a uniform build system, but also provides

convenience to developers in building large projects, especially those depending on many other existing projects or packages.

2.3.3j. iOS Simulator

The iOS simulator is a tool produced by Apple and included in the Xcode development tool. The simulator allows a developer to run an iOS application virtually on a Mac. By running the application on the simulator, major problems can be identified and fixed during design and early testing. It also allows developers to test the application's user interface and compare it with different iOS devices (iPhone, iPhone with Retina display, and iPad) (Apple Inc., 2012).

Chapter 3. Methodology

The main goal of this project was to develop a live Major Incident Handling prototype application capable of running on various devices that improves the mobility of ServiceNow. In order to build this hybrid application, the team conducted research, developed the program using Eclipse, tested the application using iOS Simulator, and evaluated as well as enhanced the application. In addition to developing this application, the project team provided recommendations for further improvements.

3.1. Research Goals and Objectives

For this project, the team delivered a mobile reporting application for Barclays during their eight weeks in New York (See Appendix A for timeline). Users can view a list of Major Incident Handlings (MIHs), their corresponding status and important information from this application. The team completed the following objectives to fulfill this goal:

1. collect requirements for the mobile application;
2. conduct research on application development, such as helpful tools and software;
3. design and implement the application;
4. evaluate and enhance the application; and
5. recommend future improvements.

3.2. Data Collection

In order to understand and develop the mobile reporting application, the team used two essential methods: research and meetings. The team researched the software available to develop an application in addition to the web-based application ServiceNow, which the mobile application was based on. Additionally, the team set up relevant software and created sample projects on their personal devices to better understand the backend logic. Furthermore, the team met with several Barclays employees who had mobile application development experience.

3.2.1. Research

The team began the research by reviewing the software and languages that would be useful to the project. The computer languages that the team focused on include JavaScript, HTML5, Java, CSS3, XML and JSON. The team also reviewed useful libraries such as backbone.js, jQuery, RequireJS, XStream, and additional software such as Maven, and

Dropwizard. While conducting the research, the team set up most of the tools on their own devices and created sample projects to better understand how they work separately.

In addition, the team examined the ServiceNow platform online to see how people use the Incident Management software and decided on the key information that should be displayed in the mobile application.

3.2.2. Meetings

The team continued data collection by meeting with several Barclays employees. The purpose of these meetings was to obtain the necessary information for the ServiceNow mobile application and discuss the integration process. During the meetings, all team members took notes and asked any applicable questions.

3.3. Development Approach

The team followed a modified scrum approach to develop the application. The scrum approach is a "framework for organizing and managing work" (Rubin, 2012). It splits the development process into 'sprints' of a predetermined length. At the beginning of each sprint, goals are set and prioritized. It has three core roles - product owner, scrum master, and development team. The product owner is the central point of product leadership and has two main tasks. The first is to understand the needs of the end user and act as their voice. The second is to communicate with the development team about product requirements and the priority of each requirement. The scrum master on the other hand, is the development team's leader. This role has a number of responsibilities, including ensuring the sprint is on schedule and discussing and adjusting the sprint deliverables with the product owner when a task is unattainable or needs more time. The development team is the group that designs, develops, integrates and tests the product. The team members meet each morning to plan each day's tasks based on how to accomplish the sprint's goals. Additionally, the development team meets with the product owner at the end of the sprint to discuss the previous sprint's achievements and the next sprint's tasks (Rubin, 2012).

In the team's version of this approach, each sprint was one week long. The product owner for the team was the sponsor, Gregory Friel, however at times Terrance Snyder also held this role. The scrum master was one of the team members and the position rotated each sprint. The team as

a whole constituted the development team. During each sprint, the development team and scrum master met each morning to assess the progress and plan that day's work. At the end of each sprint the development team, scrum master and product owner all met to discuss that sprint's accomplishments and future objectives.

Using this approach the team was constantly motivated and had attainable weekly tasks. The timeline in Appendix A displays the team's progress in creating and implementing the application during the eight weeks in New York.

Chapter 4. Application Design

This chapter describes the principles that the team considered in designing the application, the general framework of the application, the structure of the application for implementation, and the requirements to the environment.

4.1. Design Considerations

The team first created Design Considerations based on the research on what makes a user-friendly application. These considerations are displayed in Table 1 below.

Table 1: Design Considerations

Quality	Method
Friendly User Interface	• Acceptable font size • Responsive – change on screen when link is clicked • Easy-to-use and intuitive ○ Appropriately designed links • Attention-to-detail
Continuity	• Uses familiar elements • Follows conventions and patterns
Portability	• Can run on multiple mobile devices
Integrated	• Incorporates device technologies (location services, accelerometer, etc.)
Fast	• Only includes necessary information

4.2. ServiceNow Application Design

After creating the design considerations, the team examined both the iOS mail application and an available ServiceNow application. The iOS mail application is seen in Figure 2.

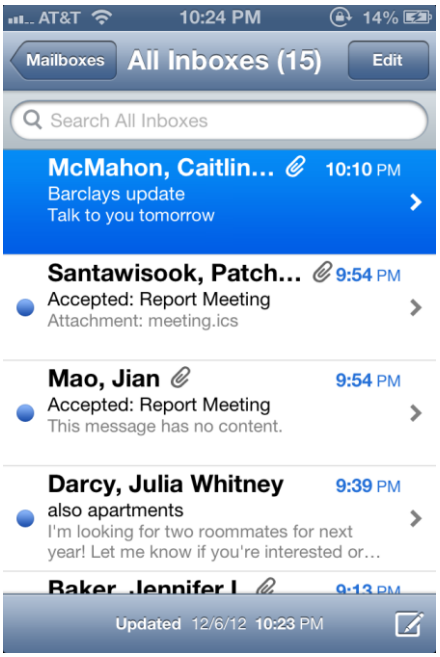


Figure 2: iOS Mail Application List Screen

The mail application has a number of great features that the team emulated. The first is having a list screen and a detail screen. In the mail application, the list screen displays all e-mails with a little bit of detail (sender, time received, and a few lines of the e-mail). The detail screen on the other hand, shows one e-mail and includes much more information from the message. Additionally, the text on both screens is formatted to display each field differently. Moreover, on the list screen, a blue circle denotes an unopened message. This makes the structure easy to understand. Lastly, when a message on the list screen is touched (so as to see the details of that message) the background of that message changes from white to blue.

Next, the team assessed the available ServiceNow mobile application and found a number of areas that could be improved. The screens can be seen in Figure 3 and Figure 4 below.

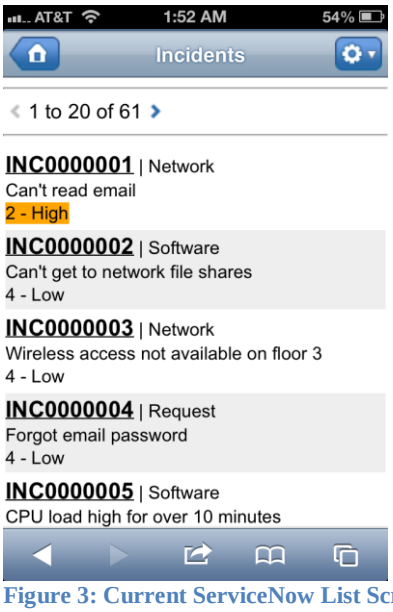


Figure 3: Current ServiceNow List Screen

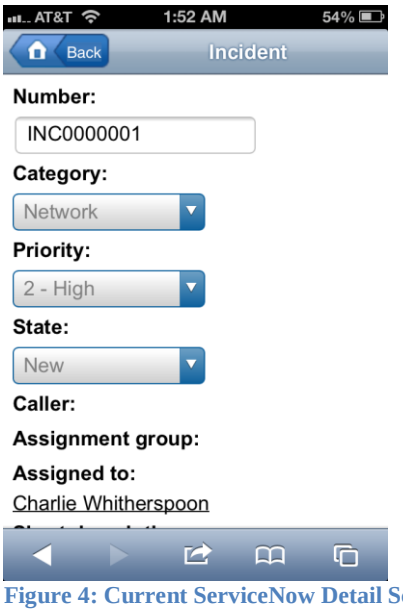


Figure 4: Current ServiceNow Detail Screen

The first area that could be improved was enlarging the clickable area on the list screen for each incident. Only the incident number could be clicked to view a specific incident's details. Moreover, upon first glance at the list screen, it is not obvious the severity of each incident aside from a highlight on some of the incidents. Also, it is not apparent if the information shown for each incident is customizable. Based on the team's discussions with Greg, the team's sponsor, the information displayed in this case is not what is important to see. Both the list screen and detail screen are not very user-friendly, and instead are very bare with little color. The detail screen provides too much unnecessary information and is deceiving in that it displays the information as though it could be edited. Finally and most importantly, this application is a web application, which means that it is a webpage that is custom made for a mobile device. It can not be downloaded onto a phone and can not as easily access the phone's features.

After inspecting the mail application and the ServiceNow web application, the team worked to design a sample of the new application. The sample screenshot of the list screen is seen in Figure 5. In the application, each incident is displayed with a color-coded icon to inform the user the status of the incident (red being degraded and green being operational). Furthermore, the information is kept to a minimum and is formatted to show the different fields. Similar to the mail application, when an incident is clicked, the background color of that incident would change as the application transitioned to the next screen.

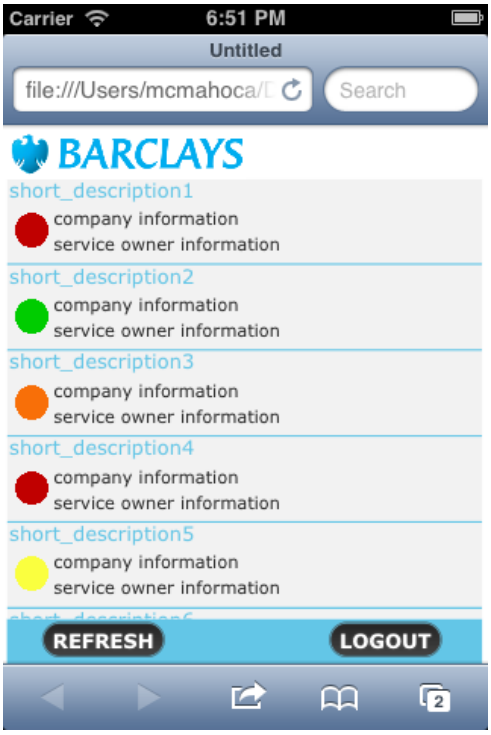


Figure 5: ServiceNow List Screen Sample

4.3. Application Framework

Instead of creating a web application (a webpage designed specifically for a mobile device) or a native application (an application that is built for a specific device and can be downloaded onto the device), the team created a hybrid application. A hybrid application is an application that is downloaded onto a mobile device and can utilize the device's browser engine. Hybrid applications are also able to work on cross-platforms.

Further, the team did not build a stand-alone application, instead embedding the application inside of the Barclays Live framework that Barclays has built to allow for a multi-purpose application.

4.4. Architecture

This project sat on the Barclays Live framework, which is supported by PhoneGap. PhoneGap, an Adobe product, allows for cross-platform development using standard web languages by utilizing the device's browser engine. It acts as a wrapper to package the mobile site and embed it to the native mobile application (See Figure 6).

Figure 6. PhoneGap workflow

To create the mobile application, the team divided the creation process into three components – Data, RESTful API, and Client. RESTful API and Data formed the server. When the user clicks the button and launches the application, the program creates a request, passes it through the firewall into the Barclays intranet, and sends the request to the server. Then the server passes the request to the Network File System and the sends the retrieved data back to the web server, which in turn renders the data to the user interface (See Figure 7).

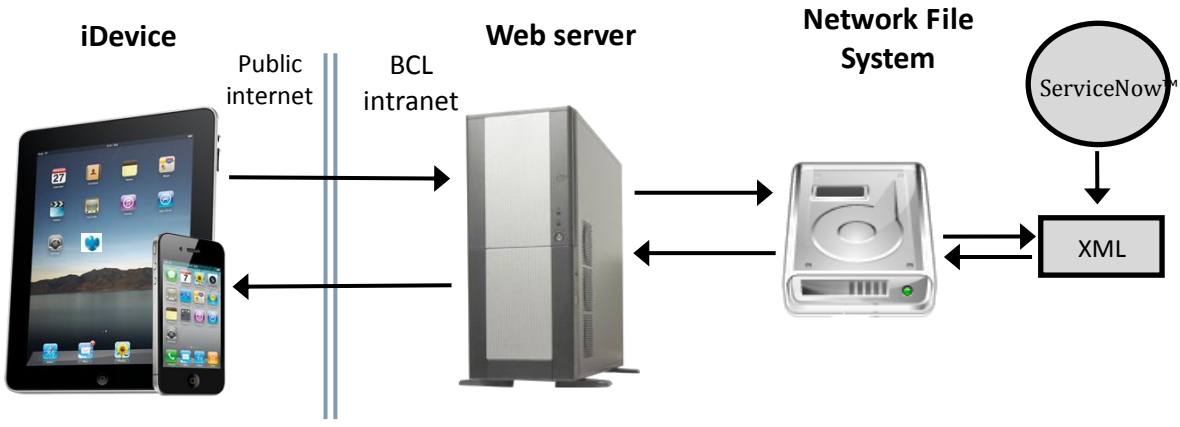


Figure 7: Application Structure

The Network File System retrieves data from a URL, where XML format data is provided by a Barclays employee. This file is updated every five minutes. The team's program was set to automatically retrieve the data from the URL once the file is updated to get the real-time data.

Once the data is retrieved, the team's program uses an XStream library to convert the data from XML format to Java Objects. During the conversion, Java Objects only saves necessary data fields, as determined by the team's meetings with Barclays employees. These information fields can be updated easily in this program, if Barclays wishes to make changes in the future.

The Java Objects are then passed into the RESTful API, which is the web server in this figure. The Jersey package in the RESTful API will convert the Java Objects to JSON, a light-weight data format for data transmission, and render the data onto the user interface, which uses the HTML5, CSS3 and JavaScript to display data and improve the look. The flow of data format conversion can be found in Figure 8.

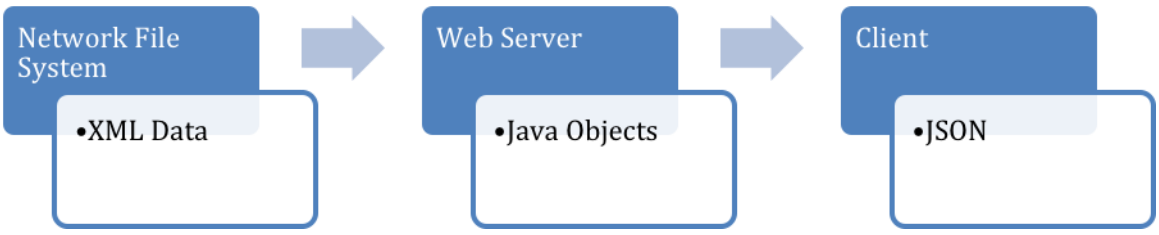


Figure 8: Data Format Conversion Flowchart

4.5. Environment

This project was developed on computers provided by Barclays. The operating systems were Windows XP and Mac OS X Version 10.7.5. The IDE for implementing the data retrieval and RESTful API was Eclipse. The iOS Simulator in Mac OS X was mainly used for testing the user interface. Browsers like Google Chrome, Firefox and Internet Explorer8 were used for testing. Apache Maven software was used for building projects and committing them to the repository. A detailed summary of technology applied in this program is displayed in Table 2 below.

Table 2: Server and Client Side Technology Comparison

	Server	Client
Languages	Java	HTML5, CSS3, JavaScript
Software	Eclipse Apache Maven	
		iOS Simulator Web Browsers
Framework and Libraries	Apache HttpClient XStream Dropwizard	Backbone.js Underscore.js RequireJS jQuery

Chapter 5. Development Phase I: Prototyping

This chapter describes the process of setting up and implementing the team's application to work independently on computers, without being connected to external environment or servers.

5.1. Server

This section illustrates how the backend program works, which includes the deserialization of data and server hosting.

5.1.1. Data

In this phase, the MIH data that were going to be displayed on the user interface were retrieved from the external server and saved in a static file. These MIH data were stored in the format of XML, and then converted to Java Objects.

To convert the data from XML format to Java Objects, XStream library was used. Based on the XML file that contained MIH data, four model classes were created in Java to represent the structure of how the data should be stored.

The first class represented the Simple Object Access Protocol (SOAP) envelope. Inside the SOAP envelope was the SOAP body, which was represented by the second class. The third was the Payload class, which was the list of MIH records. The fourth class was the Record class, which represented the structure of each MIH record. The relationship of the data classes can be seen in Figure 9.

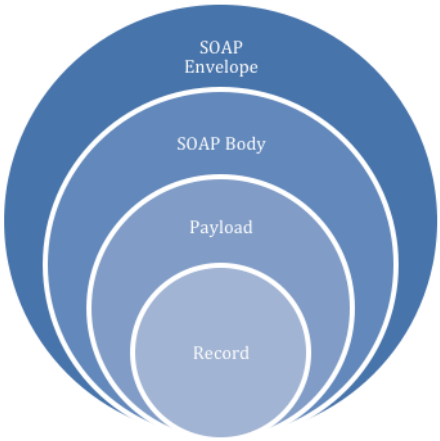


Figure 9: XML data structure

After all classes were created, the *fromXML()* function defined by XStream library was used to deserialize data from XML to Java Objects.

5.1.2. RESTful API

The program that can provide web service upon requests is the key to this project. The team chose to use the RESTful API to support the web service for this project. A RESTful API follows four basic design rules as below:

- use HTTP methods explicitly;
- be stateless;
- expose directory structure-like URIs; and
- transfer XML, JavaScript Object Notation (JSON), or both (Rodriguez, 2008).

To launch the server, the team utilized Dropwizard, which pulls multiple external libraries together, as the framework. Due to the light-weight and simple properties, Dropwizard library helped to initialize the server in a short time (Hale, 2012). Once a user request is received, the program gets the Java objects, and then saves them into a map so that further data retrieval from the client is easier.

The team organized these files into several directories including: service, configuration, core, db, and resource. Table 3 describes each directory and the main uses of each.

Table 3: Dropwizard directories and their uses

Directories	Uses
service	• Main class used for setting up the environment and compiling this program • The *assetBundle* function to set the path for the client files • The *addResource* function to call the data and render them to the client
configuration	• Defined the environment-specify parameters • Used a class and subclass to achieve the definition. The subclass is used to specify these necessary fields and the class acts like a wrapper that contains the subclass as a member field. The grouping makes the configuration file and class manageable, especially when there are too many parameters.
core	• Included compile files for data deserialization from XML to Java objects
db	• Retrieved data from the output from the core deserialization
resource	• Included three resource classes, each associated with a URI template • Included two important annotations, @Path and @Produces - @Path defines the path through which the user can access. For example, @Path("/service") tells Jersey that this resource can be accessed at the URI /service. - @Produces regulates the representation out from the resource and send to clients. This program has used two types, including @Produces("MediaType.APPLICATION_JSON") for rendering data in JSON format and @Produces("MediaType.TEXT_HTML") for displaying the HTML pages.

5.2. Client Side

The next step for this project was to make the interface to display rendered data in a better way. This program has used HTML5, CSS3, and JavaScript (jQuery) to achieve this goal.

HTML5 is mainly used to define the structure of the page, such as where the header and content are located. HTML also refers to the CSS file and JavaScript to make connections. This project contained two HTML web pages, one for the MIH list page and the other one for the MIH detail page. These two pages mainly used id and class to identify different tags so as to hold the place for the real data.

The CSS file here is used for describing the format and the look of the HTML file, such as the scroll bar, the background color, the fonts and size of words. It used the class name or id to match up with the tags in the HTML. The CSS file is the external style sheet, and the internal CSS code has the higher priority to apply in web pages.

JavaScript played an important role in rendering the data to the HTML, especially jQuery library in this project. The jQuery library has easier syntax and can achieve the same features with less code compared with the standard JavaScript library (jQuery: Advantages and Disadvantages). The program mainly used *getJSON* to retrieve data from the server program by using the URI defined in the resource file. *Each* and *append* are the other two key functions, which can collaborate together to deal with each record in a whole list of records and show it on the HTML.

5.3. Result

In this phase, the team was able to display the MIH data retrieved from the static file through the RESTful API. The team also used CSS3, HTML5, and jQuery to make the style and layout of the screen display in a nicer way. The result of Phase I for the List screen and Detail screen can be seen in Figure 10 and Figure 11.

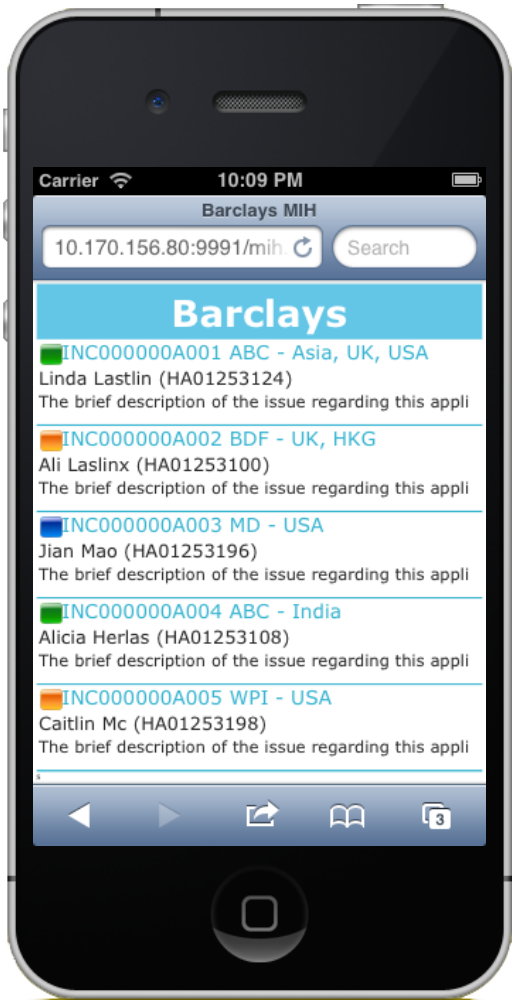


Figure 10: List Screen

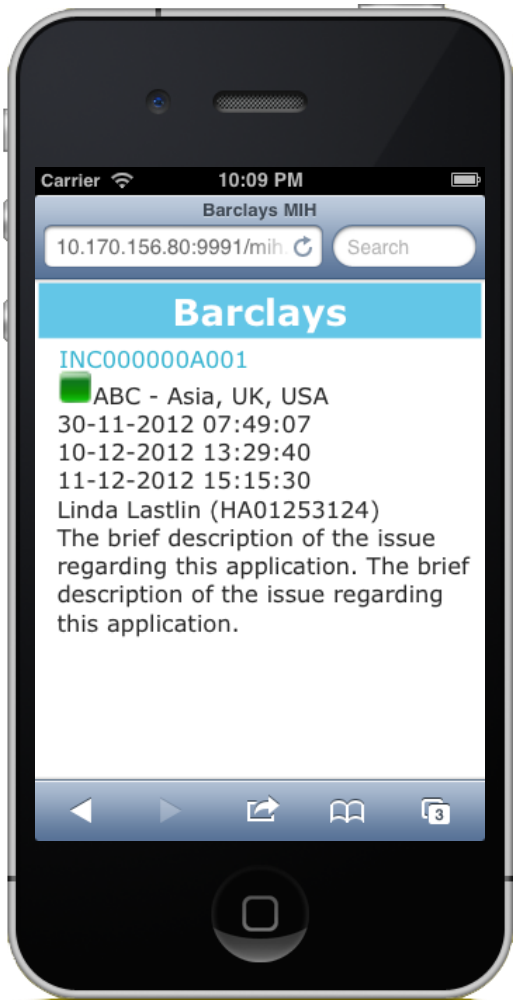


Figure 11: Detail Screen

5.4. Feedback

After the team got the Phase I result, two screens displayed on iPhone Simulator were presented to the team's sponsor, Greg, and discussed among the team members. Feedback focused on the structure of the screens and the functionality of the application. Figure 12 shows the screenshot of the list screen before the feedback and the screenshot of it after the team made the changes based on the feedback.

Figure 12: List Screen before and after feedback

1. The incident number was removed from the list screen since only the most important piece of information should be shown on the list screen.
2. Based on Greg's suggestion, the MIH application's name was the most important information that should be on the first screen.
3. The incident owner was one of the important documents that should be shown on the list screen, but his/her username or id was removed to make the user interface looked better.
4. Based on the mail application on the iPhone, two lines of short descriptions would be better than one so that users could have better picture of what the issue of that incident was.

Figure 13 shows the screenshot of the detail screen before the feedback and the screenshot of the detail screen after the team made the changes based on the feedback.

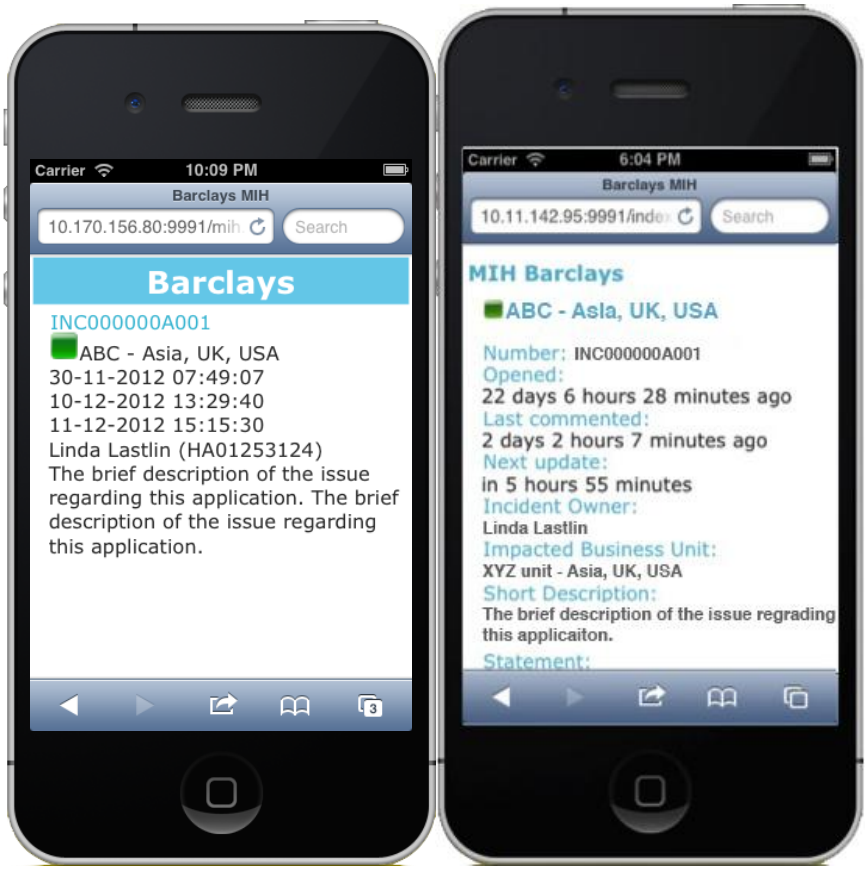


Figure 13: Detail screen before and after feedback

1. The incident number was moved down to the second line. For the second screen, it is necessary to include the incident number so that users could refer to this number as an identifier when it happened to have duplicate names of the applications
2. The MIH application's name has been set to the top. As it was mentioned before, it was primary information that should be shown.
3. The name's tag of each piece of information was added so users could have better idea what displayed information was.
4. The information about "impacted business unit" was added.
5. The format of the time was changed based on the professor's suggestion that the later format is more user friendly than the former.

Chapter 6. Development Phase II: Enhancement

Development Phase I dealt with the setup and implementation of the application on its own. The results did not rely on anything located outside of the running computer. In Phase II, improvements were made to the application components, and the team's application prototype was ready to be moved to Barclays' application environment.

6.1. Server

The improvements in the backend server include real-time data retrieval and the implementation of database.

6.1.1. Data

In Phase II, MIH data were retrieved as real-time data , and converted into Java Objects before insertion into the database.

The recent MIH data from the browser were scheduled to be retrieved every five minutes so that the displayed MIHs on the user interface were always up to date. HttpClient library was used to download the data from the external browser. Below is the HttpClient process:

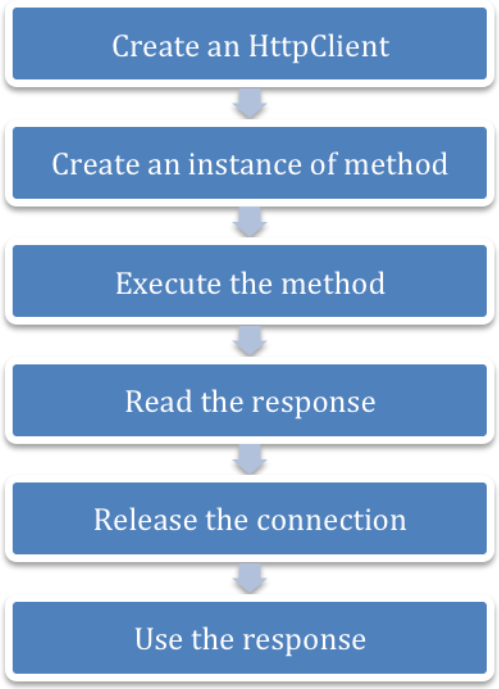


Figure 14: HttpClient process

The steps above were followed by first creating the instance of HttpClient, and the instance of one of the methods called *getMethod()*. This method took the URL as the input and retrieved the data that the URL points to. The method was executed using HttpClient by calling the *executeMethod()* function. After the execution process, the response was read by using the *getResponseBodyAsString()* function. In this way, the response downloaded from the external browser was a string containing the response body (MIH data in XML format). After the data was downloaded, the connection was released indicating that the downloading process had been finished. Then the downloaded data was converted into Java Objects using XStream library, which was also used in the first phase. To periodically retrieve data from the browser, the *ExecutorService* method was used to schedule the command to execute every five minutes.

6.1.2. RESTful API

Considering the limitation of available data source, the team decided to begin implementing a database in the server to store historical data and provide it for users in this application. The database lays the foundation for a Monthly Reporting function for future use by senior management.

6.2. Client Side

To better fit with the Barclays' framework, the team modified the structure of the client into Model – View – Controller (MVC) architecture. This organization of files separates the representation of information into three types of components:

- Model
    - Manages the application data
    - Responds to requests of data information
    - Responds to instructions to change data information
- View
    - Manages the display of information
    - Requests the necessary data to generate the presentation
- Controller
    - Interprets the mouse and keyboard inputs from the user
    - Notifies associated models or views to change

Figure 15 shows the interaction between each component.

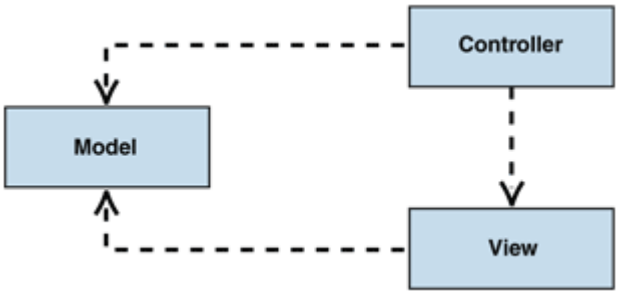


Figure 15. Model-View-Controller structure interaction Invalid source specified.

In this phase, Backbone.js, Underscore.js, Require.js, and jQuery libraries were used. Backbone.js and Underscore.js provided the structure through models, collections, and views. Require.js was a module loader and connects each component, which could improve the speed of the application. jQuery was used in the HTML templates to display data. See Figure 16 for the file structure.

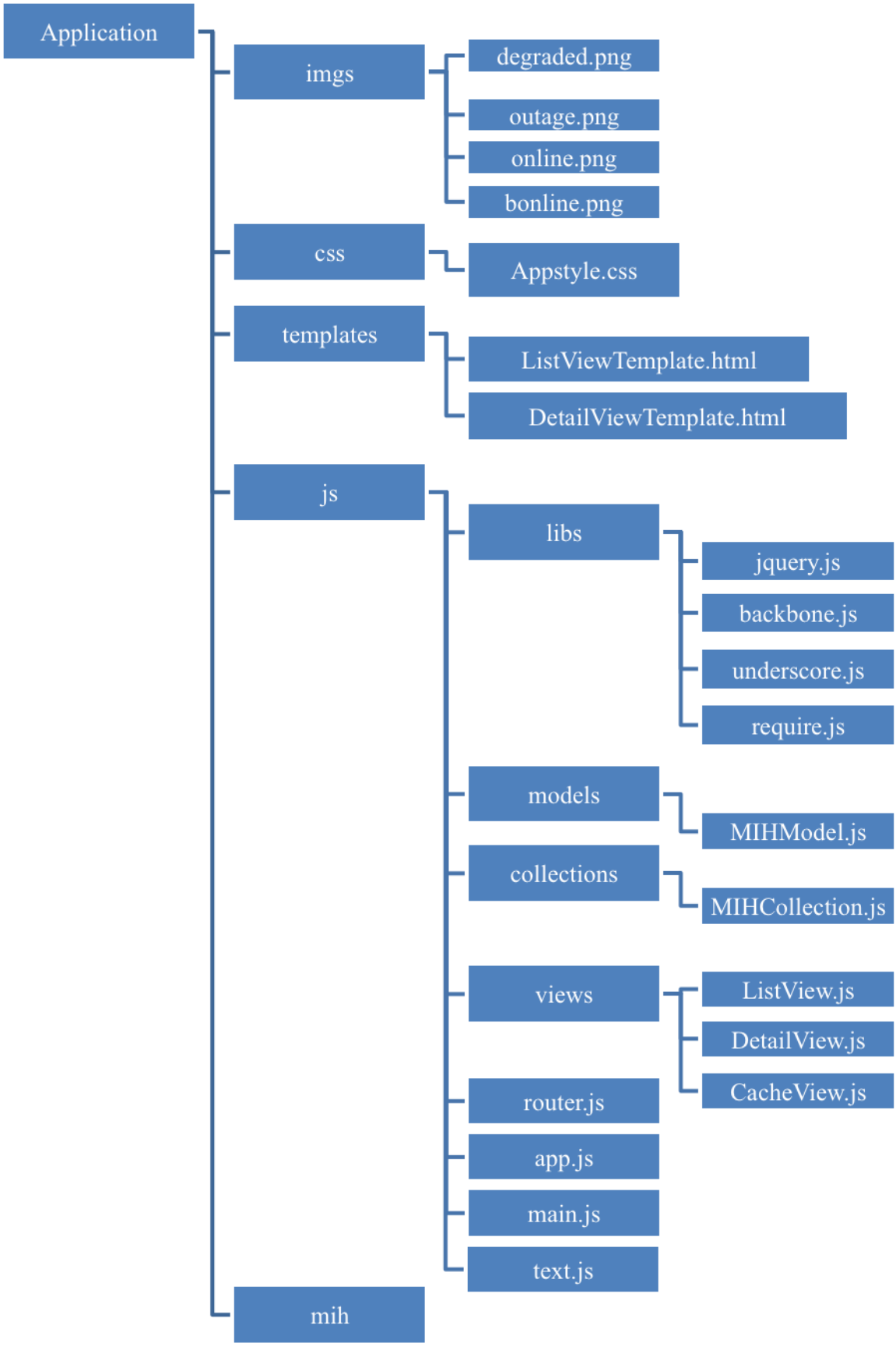


Figure 16: File structure

In this structure, MIH was the index page, which only defined the header and the content. The header was the Barclays logo on the page and content was an empty container to be filled by templates. To make the screen more user friendly, the team designed a screen to display when loading the data, which was defined in the index page. The Templates folder contained two templates – ListViewTemplate for the MIH list page and the DetailViewTemplate for the MIH detail page. JavaScript was used to define the mechanism of assigning, retrieving, and filling data into these two templates. This application would load the data only when opening the list screen. All data was cached temporarily and cleared once the user quitted the application. No more data would be loaded when going to detail screen or going back to the list screen from the detail screen.

Besides the above changes in the file structure, the team designed an iPad version for this application. Because the principles are similar to the iPhone version, the things that needed to be changed were only the style sheets and a minor change in the ListView. User agent was used in the application for conditioning statement. The program would determine the device first and then apply different code.

6.3. Integration

This application is an embedded application in the Barclays Live framework. Since this application was a prototype for further development, the integration process was not formalized. To create a working prototype, the team used one employee's proxy and a cURL command, a command line tool for transferring data with URL syntax, to post requests to the team's running server (Haxx, 2012). The server responded to the request, and then sent data back.

6.4. Result

In this phase, the team was able to retrieve the MIH data from external server every five minutes and display them to the user interface. To better fit into the Barclays Live framework, the team made modifications to the structure of the client side and finally used the Model-View-Controller architecture to make the web pages work. The team also implemented different views for iPhone and iPad. Finally, the web pages were integrated with the Barclays Live framework and the application could be viewed using both iPhone and iPad.

Figure 17 and Figure 18 are the screenshots of List screen and Detail screen respectively for the iPhone version.

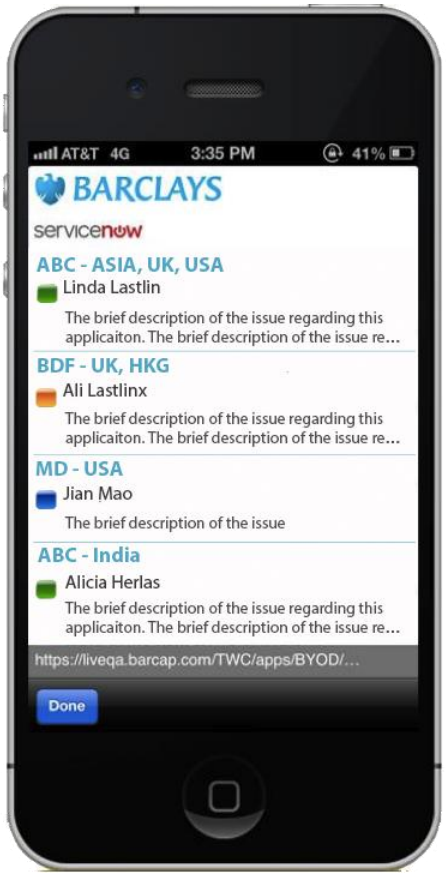


Figure 17: Final iPhone List Screen

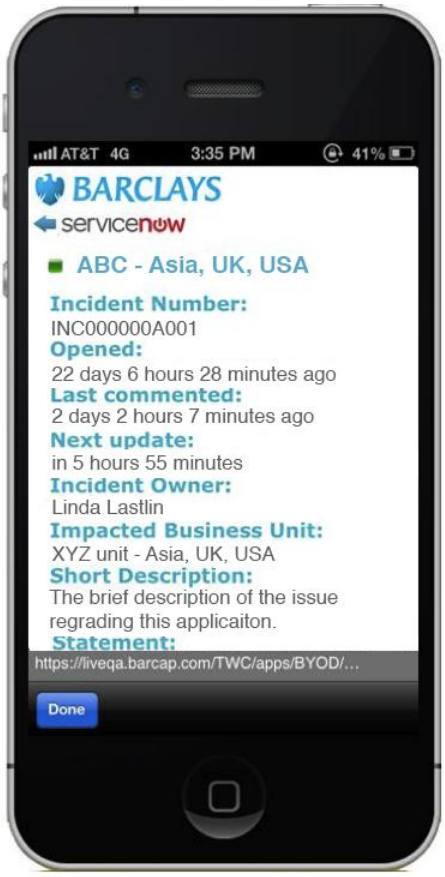


Figure 18: Final iPhone Detail Screen

Figure 19 is the screen for the iPad version.

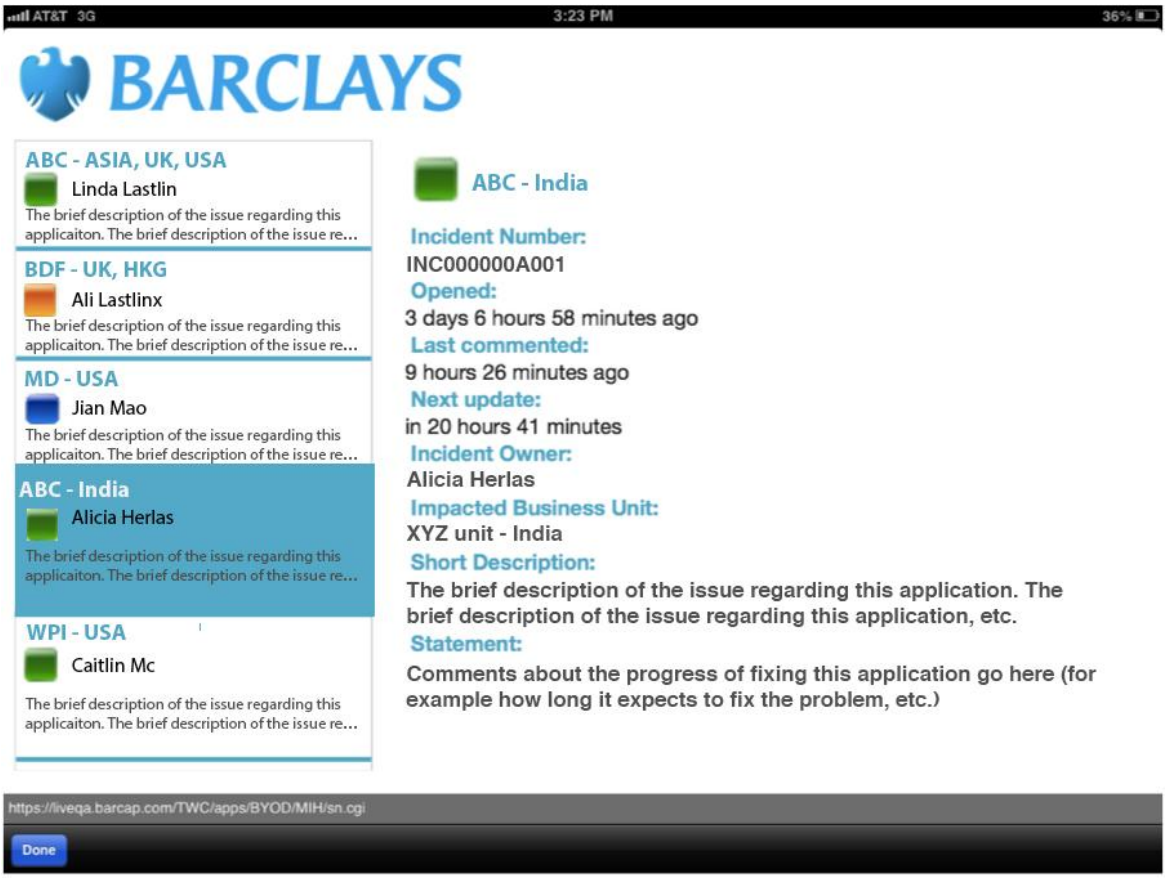


Figure 19: Final iPad screen

Chapter 7. Recommendations

The team compiled a list of enhancements from team discussions, research, and meetings. These recommended enhancements are detailed below.

7.1. Current Subscriptions

Managers at Barclays currently subscribe to specific categories of incidents and receive an email at the beginning of each day, which provides details about the incidents that were open during any part of the past 24 hours. A future enhancement would retrieve employees' subscription preferences and incorporate these into the team's application.

7.2. Monthly Information Summary

A Monthly Information Summary would include a month-to-date number of incidents for each application and department. This would provide a centralized location for incident and metrics trending analysis.

7.3. Recent History

The application only includes incidents that are currently open. It would be beneficial if incidents that were open within the past 24 hours were also included so that if an incident occurred and was fixed over night, the appropriate employees would be aware of it.

7.4. Push Notifications

When an incident occurs and/or is updated, a push notification would be sent to employees with a corresponding subscription election. This would make the notification of incidents happen in real-time.

7.5. Grouping Screen

A grouping screen would improve the functionality of the application by allowing employees to choose which department or application they want to see incidents for. This would also allow employees to see the number of current incidents affecting each Barclays' application and department.

7.6. Data Filter and Sorter

A data filter would allow users to view only the incidents that are of interest to them while a sorter would allow the user to view all of the incidents sorted based on specific characteristics.

7.6.1. Location Based Filter

To further improve the filter, the application could make use of the mobile device's location services and filter or sort the incidents based on the user's location. For instance, if an employee is located in London, England, the application would show incidents in London before incidents in Singapore or the United States.

References

Allan, A. (2010). *Learning iPhone Programming.* Sebastopol: O' Reilly Media Inc.

Apache Software Foundation. (2012). *What is Maven?* Retrieved December 8, 2012, from Apache Maven Project: http://maven.apache.org/what-is-maven.html

Apple Inc. (2012). *Safari Technology Demos*. Retrieved October 7, 2012, from Safari Dev Center: https://developer.apple.com/safaridemos/

Apple Inc. (2012, September 19). *Using iOS Simulator*. Retrieved December 8, 2012, from iOS Developer Library: http://developer.apple.com/library/ios/#DOCUMENTATION/Xcode/Conceptual/ios_development_workflow/25-Using_iOS_Simulator/ios_simulator_application.html

Apple. (2012, September). *iOS Security.* Retrieved October 9, 2012, from iPhone in Business: www.apple.com/iphone/business/integration/

Ascher Consulting & Development. (2007, September 20). *What are the advantages of using JSON?* Retrieved October 8, 2012, from http://ascherconsulting.com/what/are/the/advantages/of/using/json/

Ashkenas, J. (2012, March 21). *Backbone.js*. Retrieved October 9, 2012, from Backbone.js: www.backbonejs.org

Barclays. (2012). *Our history*. Retrieved October 1, 2012, from Barclays: http://group.barclays.com/about-barclays/about-us/our-history

Bates, M. (2012). *Programming in CoffeeScript.* Upper Saddle River, NJ: Addison-Wesley Professional.

Burke, J. (2012). *RequireJS*. Retrieved October 9, 2012, from RequireJS a JavaScript Module Loader: http://requirejs.org/

Eclipse Foundation. (2012). *Eclipse*. Retrieved December 8, 2012, from Eclipse: http://www.eclipse.org/

Finley, K. (2012, September 12). *JavaScript Tops Latest Programming Language Popularity Ranking From RedMonk*. Retrieved December 6, 2012, from Techcrunch: http://techcrunch.com/2012/09/12/javascript-tops-latest-programming-language-popularity-ranking-from-redmonk/

Fitzgerald, M. (2004, August 18). *Serializing Java Objects with XStream*. Retrieved October 7, 2012, from http://www.xml.com/lpt/a/1462

Global Finance Magazine. (2012, August 27). *World's 50 Biggest Banks 2012*. Retrieved October 8, 2012, from Global Finance: http://www.gfmag.com/tools/best-banks/11986-worlds-50-biggest-banks-2012.html#axzz28uMU2kmJ

Google. (2012, July 27). *Google-gson*. Retrieved October 10, 2012, from Google: https://code.google.com/p/google-gson/

Haxx. (2012, November 20). *cURL*. Retrieved January 2, 2013, from cURL: Curl.haxx.se/

Hale, C. (2012, November 26). *Dropwizard*. Retrieved November 27, 2012, from Dropwizard: http://dropwizard.codahale.com/

Holzner, S. (2009). *jQuery: Visual QuickStart Guide.* Berkeley, CA, USA: Peachpit Press.

Ivan, I., & Zamfiroiu, A. (2011). Quality Analysis of Mobile Applicaions. *Informatica Economică , 15* (3), 136-152.

jQuery: Advantages and Disadvantages. (n.d.). Retrieved December 2, 2012, from JSCRIPTERS.COM: http://www.jscripters.com/jquery-disadvantages-and-advantages/

Kabay, M. E. (2009). iPhone Security: Part 2-iPhone App Security Model. Northfield, VT.

Kalnichevski, O. (2008, February 8). *Apache*. Retrieved December 28, 2012, from HttpComponents: http://hc.apache.org/httpclient-3.x/tutorial.html

Kopchik, J. M. (2011, December 19). *Mobile Banking: Rewards and Risks*. Retrieved October 8, 2012, from Federal Deposit Insurance Corporation: http://www.fdic.gov/regulations/examinations/supervisory/insights/siwin11/mobile.html

Liu, L. (2009). Encyclopedia of Database Systems. New York, New York, United States of America: Springer.

Reid, J. (2011). *jQuery Mobile.* Sebastopol, CA, USA: O'Reilly Media, Inc.

Rodriguez, A. (2008, November 06). *RESTful Web services: The basics*. Retrieved September 17, 2012, from IBM: http://www.ibm.com/developerworks/webservices/library/ws-restful%20/

Rubin, K. S. (2012). *Essential Scrum: A Practical Guide to the Most Popular Agile Process.* Addison-Wesley Professional.

Samanta, D. (2004). *Object-Oriented Programming with C++ and Java.* Prentice-Hall of India Private Limited.

ServiceNow. (2012). *About Us*. Retrieved September 15, 2012, from ServiceNow: http://www.servicenow.com/about-us.do

Seven, D. (2012, June 14). *What is a Hybrid Mobile App?* Retrieved October 6, 2012, from http://icenium.com/community/blog/icenium-team-blog/2012/06/14/what-is-a-hybrid-mobile-app-

Stark, J. (2010). *Building iPhone Apps with HTML, CSS, and JavaScript.* O'Reilly Media.

The jQuery Foundation. (2012). Retrieved October 9, 2012, from jQuery Mobile: http://jquerymobile.com

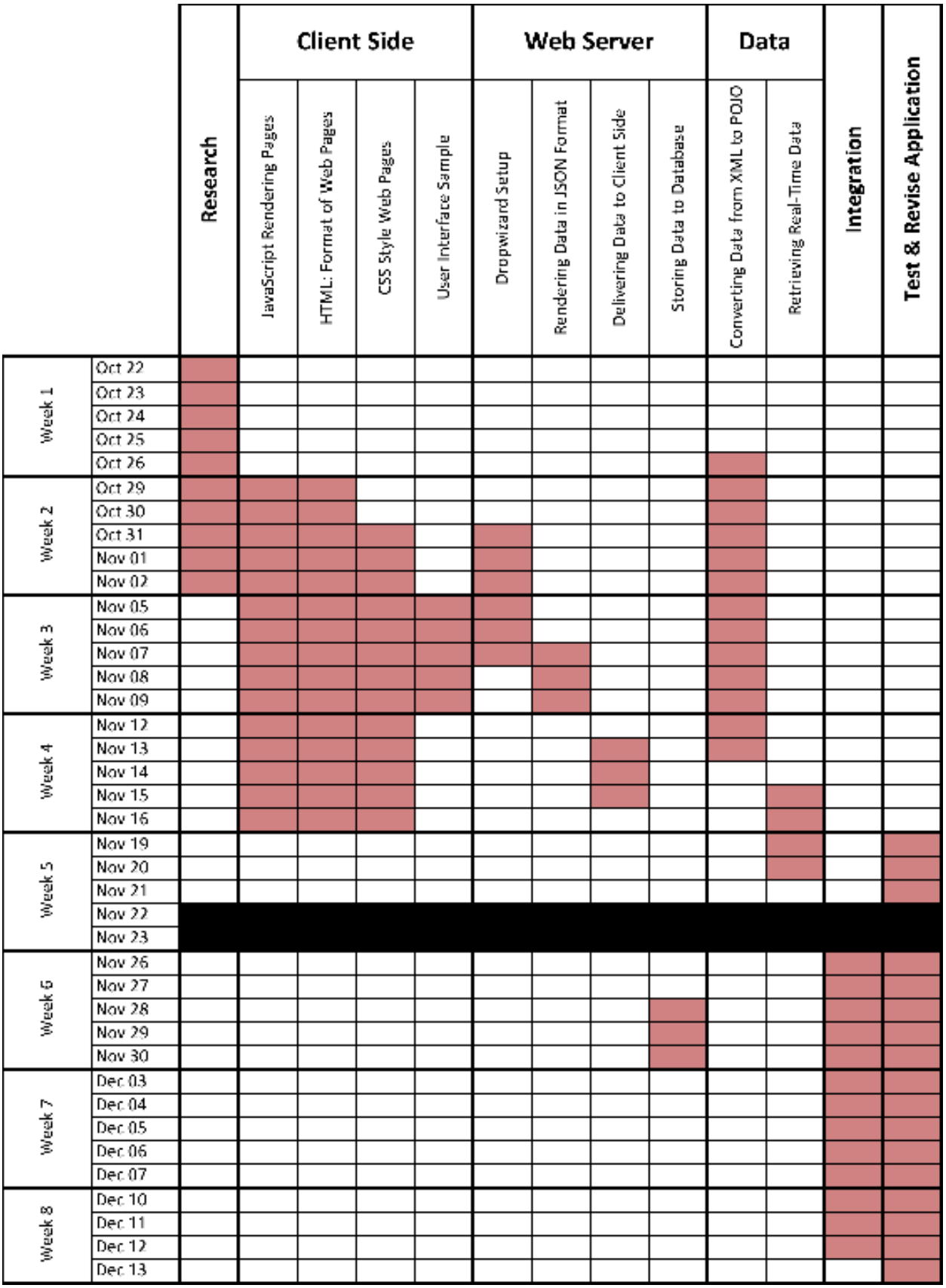
W3Schools. (2012). *CSS Introduction*. Retrieved December 8, 2012, from w3schools.com: http://www.w3schools.com/css/css_intro.asp

w3schools. (2012). *HTML5 Introduction*. Retrieved October 7, 2012, from http://www.w3schools.com/html/html5_intro.asp

XStream. (2012, July 17). *About XStream*. Retrieved October 7, 2012, from XStream: http://xstream.codehaus.org/

Yelton, A. (2012). Mobile Websites. *Library Technology Reports , 48* (1), 9+.

Appendix A


Research
Client Side
JavaScript Rendering Pages
HTML: Format of Web Pages
CSS Style Web Pages
User Interface Sample
Web Server
Dropwizard Setup
Rendering Data in JSON Format
Delivering Data to Client Side
Storing Data to Database
Data
Converting Data from XML to POJO
Retrieving Real-Time Data
Integration
Test & Revise Application
Week 1
Oct 22
Oct 23
Oct 24
Oct 25
Oct 26
Week 2
Oct 29
Oct 30
Oct 31
Nov 01
Nov 02
Week 3
Nov 05
Nov 06
Nov 07
Nov 08
Nov 09
Week 4
Nov 12
Nov 13
Nov 14
Nov 15
Nov 16
Week 5
Nov 19
Nov 20
Nov 21
Nov 22
Nov 23
Week 6
Nov 26
Nov 27
Nov 28
Nov 29
Nov 30
Week 7
Dec 03
Dec 04
Dec 05
Dec 06
Dec 07
Week 8
Dec 10
Dec 11
Dec 12
Dec 13