Technical Documentation

STC Web Analytics Dashboard

Project Name	STC Web Analytics Dashboard
Author	Ansel Chang, Daniel Stoiber, Lorenzo Manfredi Segato, Pedro Leão
Date	28 February 2024
Version	v1.0

Version History

Version	Author	Updated Date
v1.0	Ansel Chang, Daniel Stoiber, Lorenzo Manfredi Segato, Pedro Leão	1 March 2024

Table of Contents

Introduction

The goal of this project is twofold: To create a dashboard displaying aggregated web analytics from social media clearly and succinctly; develop an automatic report based upon these analytics containing derived metrics and statistics. Given these requirements, the team decided to develop an authenticated web app encompassing these features hosted on STC's domain.

Scope-wise, the project aims to fully-integrate as many social media sources as possible. However, due to the volume of accounts under STC's purview, concessions must be made due to time and feasibility constraints (a well-documented API, for example).

The purpose of this project is to support STC's new marketing enterprise in the upcoming months by automating the existing manual process of obtaining web-analytics, reorganising the report structure and providing many quality-of-life features streamlining gathering and analysis of marketing data.

Document Outline

This document will include information on what the Web Analytics Dashboard does, all layers of functionality, what tools were used to develop it, and how to get the dashboard up and running from start to finish.

System Overview

Features and Functionalities

Assumptions and Dependencies

Describe here the assumptions and dependencies about your software and its use.

Assumptions

- Users have an understanding of how to navigate browsers and websites.
- Users are familiar with web analytic terminology.

Dependencies

- Social media have usable and well-documented APIs.
- Existing frameworks and libraries for a timely development process.
- An internet connection is required (even if hosted locally, as data must be fetched from sources).

General Constraints

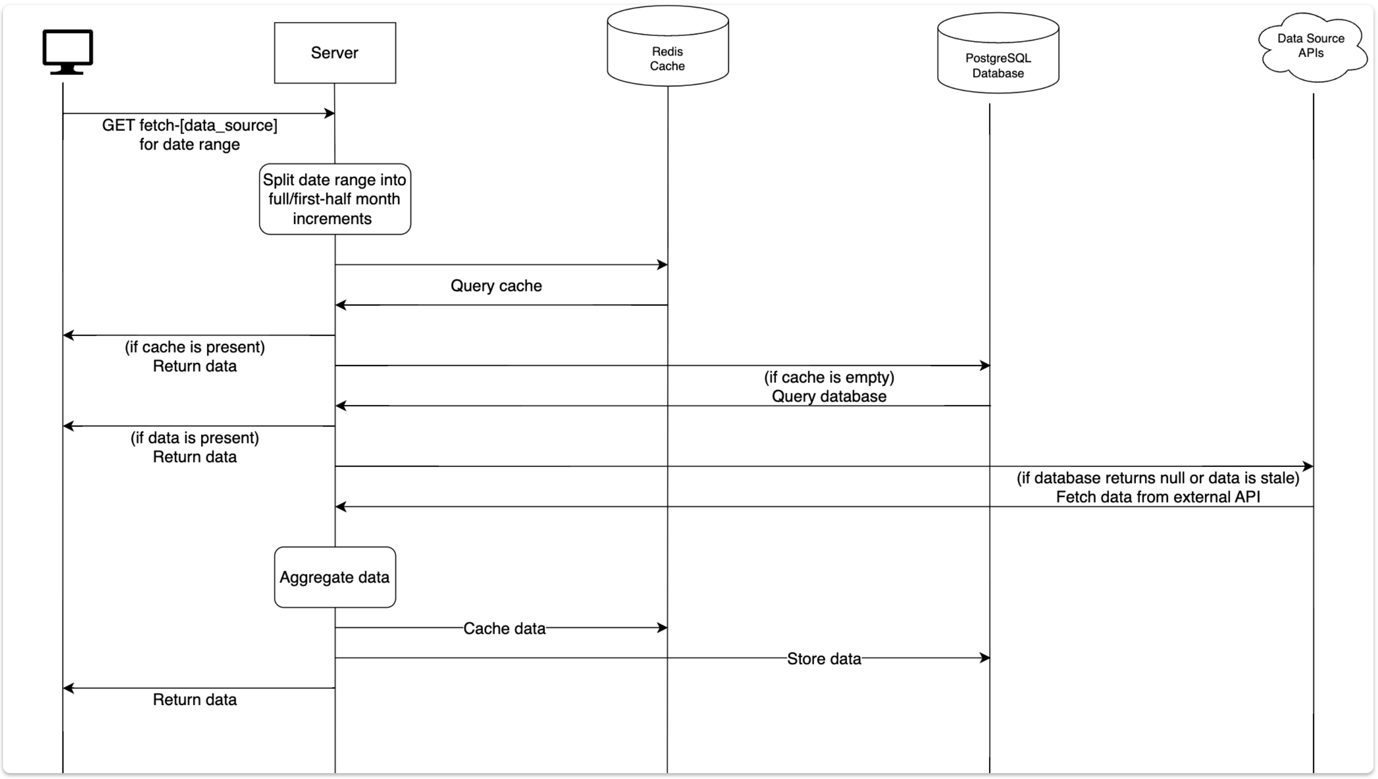
The constraint that affected this enterprise mostly falls under implementation feasibility. This project is to be completed in the short span of 7 weeks. With such a limited time frame the development structure has been organised into a Minimal Viable Product (MVP), with additional features time permitting. The MVP consists of successfully achieving the primary goals: the web analytics dashboard and the generated report. Additional features may include a persistent database (back-up of fetched data) containing the aggregated data, caching, dynamic chart generation and more quality-of-life-related features.

Additionally, a significant portion of this project rests upon the APIs of the different social media - something outside of the team's control. A well-documented and flexible API allows for a shorter familiarisation period, leading to an easier implementation process and the development of incremental features.

Architectural Strategies

System Architecture

Features including the social dashboard, year comparison, and report generation all use a function that implement the following flow:


Server
Redis Cache
PostgreSQL Database
Data Source APIs
GET fetch-[data_source] for date range
Split date range into full/first-half month increments
Query cache
(if cache is present) Return data
(if cache is empty) Query database
(if data is present) Return data
(if database returns null or data is stale) Fetch data from external API
Aggregate data
Cache data
Store data
Return data

1. The user selects a date range to visualize or export.
2. A request is made to the server where the date range is split into full month increments or first half of the month increments, depending on the resolution specified.
3. Both full and first half of the month are requested for reports.
4. The server checks, for each increment of time, if the data is already present in the Redis cache.
5. If the data is present, it will return the data.
6. If the data is not present, it will continue to the next step.
7. The server checks, for each increment of time, if the data is present in the PostgreSQL database.

8. If the data is present and it is not stale, meaning the data was originally fetched at a date that is deemed to be too long ago, it will return the data.
9. If the data is not present or it is stale, it will continue to the next step.
10. The server will make a request to the data source, for each increment of time, and structure the data returned in the specified format. It will then store that data in both the Redis cache, as well as the PostgreSQL database.
11. For data sources that no longer track the month being requested, the server will return the stale data from the database if it is available. Otherwise, it will return null.

Policies and Tactics

- Coding Standards:
  - Camel case for variable, function, and class names (uniformity and clarity)
  - Kebab case for file names due to Next.js using the application file structure as the website path routes.
- Design Choices:
  - Next.js App Router: The React framework introduced an App Router which supports the use of shared layouts and nested routing, loading states, error handling, and more. The App Router works similar to a file-based directory system. Folders are used to define routes (a single path of nested folders) following the file-system hierarchy from the root folder downwards. Files are used to create a UI that is shown for that specific route segment. Please see documentation for full details on the App Router.
  - Next.js/Shadcn Accessibility: Next.js and Shadcn provide multiple automatic checks to make sure the application is compatible and up to ADA accessibility standards (keyboard navigation, screen reader compatibility, etc.).

Detailed System Design

Resources

- Next.js
  - Classification: Website Framework (React-based)
  - Definition: A React Framework for building full-stack applications.
  - Responsibilities: React components are used to build user interfaces while Next.js adds optimizations and features. Next.js is responsible for the automatic configuring and abstraction of tooling needed for React. See documentation in bibliography for full details.
  - Constraints: Limits the application to a web app.
  - Composition: Using the ‘App Router’ model for application building (as opposed to ‘Page Router’ functionality other React frameworks may use).
  - Resources: See bibliography for full documentation.
- Shadcn
  - Classification: UI Library, not a dependency (see documentation for full details)

    - Definition: A reusable UI component collection using Tailwind CSS and Radix UI design to support frameworks such as Next.js. This library offers a large suite of accessible and customizable components.
    - Responsibilities: Responsible for many of the front-end components of the application.
    - Constraints: N/A: Flexible and versatile. There are no dependencies needed to use Shadcn, and its functionality, if needed, can be removed at any time.
    - Composition: Using the components is as simple as copying them over from the documentation onto the project.
    - Resources: See bibliography for full documentation.
- PostgreSQL
    - Classification: Object-relational Database Management System.
    - Definition: Please refer to bibliography for full definition and documentation.
    - Responsibilities: Dashboard Authentication, Datasource Authentication, Persistent Data Storage
    - Constraints: Steep learning curve and operational complexity
    - Composition: Please see Entity Relation Diagram for table composition.
    - Users/Interactions: N/A.
    - Resources: Please see bibliography for full documentation.
    - Processing: PostgreSQL supports SQL queries, thereby making it extremely powerful and versatile. Please refer to the SQL documentation in the bibliography for further details.
    - Interface/Exports: Please see bibliography for full documentation.
- Redis
    - Classification: Relational Database Management System
    - Definition: Redis is used as a caching data store to reduce the amount of API calls.
    - Responsibilities: Caching API data.
    - Constraints: Memory size limitations and persistent data storage.
    - Composition: Please Entity Relation Diagram for table composition.
    - Users/Interactions: N/A.
    - Resources: Please see bibliography for full documentation.
    - Processing: Does not support SQL querying natively. However, it does support Key-Value, String, List, Set and Hash operations. Please see documentation in bibliography for full processing documentation.
    - Interface/Exports: Please see documentation in bibliography for full processing documentation.
- Vercel
    - Classification: Cloud platform
    - Definition: All-encompassing cloud platform enabling deployment, hosting and scaling of websites and applications.
    - Responsibilities: Deployment of the web application.

    - **Constraints**: Serverless function execution limits and backend capabilities. Please see documentation for more details.
    - **Composition**: Git-based integration, server-less functions. Vercel is optimized for Next.js so deployment composition is seamless.
    - **Users/Interactions**: N/A
    - **Resources**: Please see documentation for full details.
- **Tailwind**
    - **Classification**: CSS Framework
    - **Definition**: A CSS Framework with a plethora of classes that can be composed to build up designs.
    - **Responsibilities**: User Interface.
    - **Composition**: Tailwind Dependency Installation for development.
    - **Resources**: Please see bibliography for full documentation.

Developing, Building, and Deploying

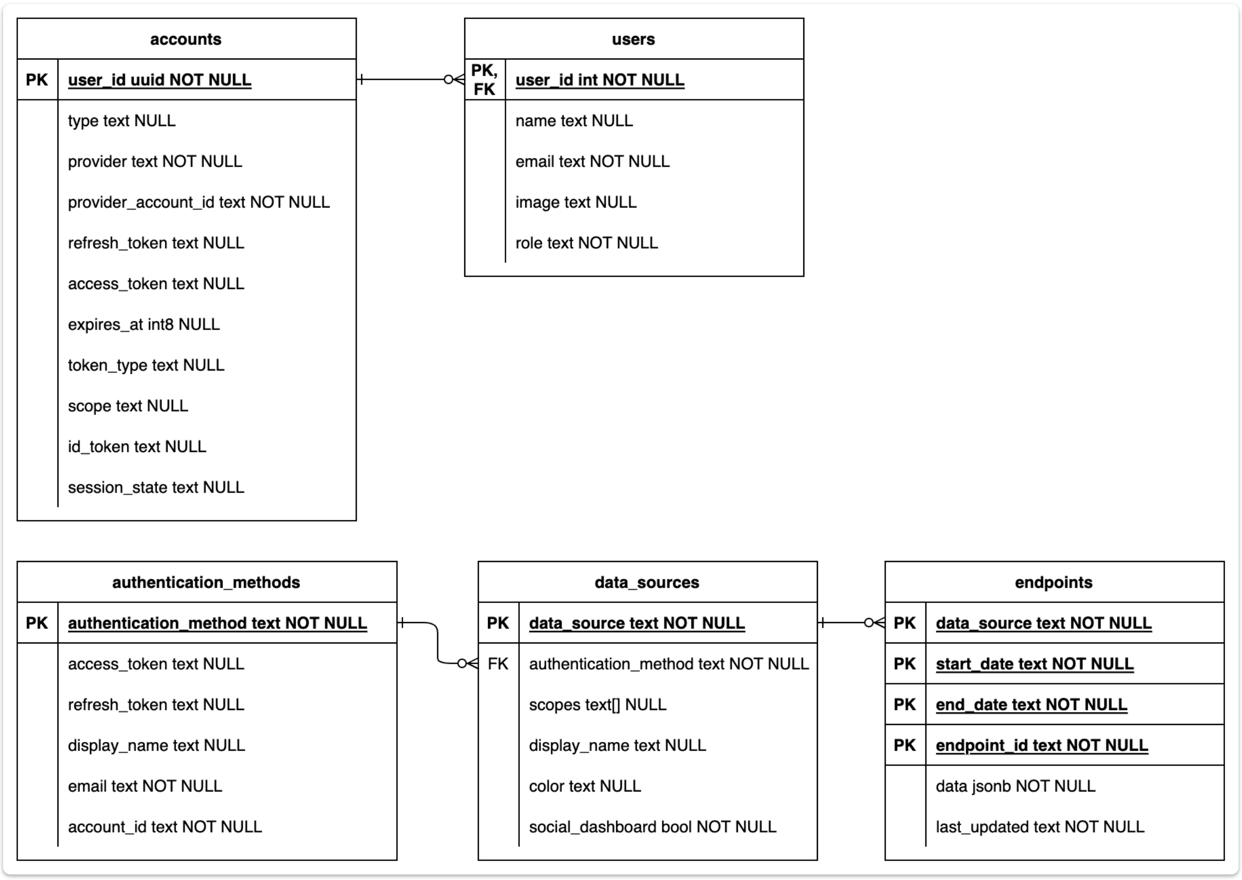
Prerequisites

This application requires the installation of the following tools:

- Node.js (v20.11.1+)
- Yarn (v1.22.19+)
  The application also requires a set of environment variables:
  oAuth
- `NEXTAUTH_URL`
- `GOOGLE_ID`
- `GOOGLE_SECRET`
- `NEXTAUTH_SECRET` (For development this can be any non-empty string, for development a set of 16 randomly generated characters is recommended.)
  Database
- PostgreSQL:
    - `POSTGRES_URL` (Set automatically by Vercel in production)
    - `POSTGRES_PRISMA_URL` (Set automatically by Vercel in production)
    - `POSTGRES_URL_NON_POOLING` (Set automatically by Vercel in production)
    - `POSTGRES_USER` (Set automatically by Vercel in production)
    - `POSTGRES_HOST` (Set automatically by Vercel in production)
    - `POSTGRES_PASSWORD` (Set automatically by Vercel in production)
    - `POSTGRES_DATABASE` (Set automatically by Vercel in production)
    - `DEV` (Only required when hosting a database locally that does not have SSL enabled)
      Redis
- `KV_URL` (Set automatically by Vercel in production)
- `KV_REST_API_URL` (Set automatically by Vercel in production)

- `KV_REST_API_TOKEN` (Set automatically by Vercel in production)
- `KV_REST_API_READ_ONLY_TOKEN` (Set automatically by Vercel in production)
  Data Sources
- Google:
  - `GOOGLE_CLIENT_EMAIL`
  - `GOOGLE_PRIVATE_KEY`
- Meta:
  - `META_APP_ID`
  - `META_CLIENT_ID`
  - `META_SECRET`
  - `INSTAGRAM_USER_ID`

  A Redis database is also required to be running, along with a PostgreSQL database with the following schema:



```sql
-- Table of oAuth accounts for dashboard authentication
CREATE TABLE "public"."accounts" (
    "user_id" uuid NOT NULL,
    "type" text,
    "provider" text NOT NULL,
    "provider_account_id" text NOT NULL,
    "refresh_token" text,
    "access_token" text,
    "expires_at" text,
    "token_type" text,
    "scope" text,
    "id_token" text,
    "session_state" text,
    PRIMARY KEY ("user_id")
);

-- Table of users for the dashboard
CREATE TABLE "public"."users" (
    "user_id" uuid NOT NULL,
    "name" text,
    "email" text NOT NULL,
    "image" text,
    "role" text NOT NULL DEFAULT 'default'::text,
    PRIMARY KEY ("user_id")
);

-- Table of data source authentication methods
CREATE TABLE "public"."authentication_methods" (
    "authentication_method" text NOT NULL,
    "access_token" text,
    "refresh_token" text,
    "display_name" text,
    "email" text NOT NULL,
    "account_id" text NOT NULL,
    PRIMARY KEY ("authentication_method")
);

-- Table of data sources to display
CREATE TABLE "public"."data_sources" (
    "data_source" text NOT NULL,
    "authentication_method" text,
    "scopes" _text,
    "display_name" text,
    "color" text,
    "social_dashboard" bool NOT NULL,
    PRIMARY KEY ("data_source")
);

-- Table of endpoints for persistent storage
CREATE TABLE "public"."endpoints" (
    "data_source" text NOT NULL,
    "start_date" text NOT NULL,
    "end_date" text NOT NULL,
    "data" jsonb NOT NULL,
    "last_updated" text NOT NULL,
    "endpoint_id" text NOT NULL,
    PRIMARY KEY ("data_source","endpoint_id","start_date","end_date")
);
```

```
-- FK constraint on accounts table to link
ALTER TABLE "public"."accounts" ADD FOREIGN KEY ("user_id") REFERENCES
"public"."users"("user_id") ON DELETE CASCADE;
-- FK constraint on data_sources table to link to authentication methods
ALTER TABLE "public"."data_sources" ADD FOREIGN KEY ("authentication_method")
REFERENCES "public"."authentication_methods"("authentication_method") ON DELETE
CASCADE;
-- FK constraint on endpoints table to link to a data_source
ALTER TABLE "public"."endpoints" ADD FOREIGN KEY ("data_source") REFERENCES
"public"."data_sources"("data_source") ON DELETE CASCADE;
```

Starting development environment

Assuming the project files are downloaded, in terminal:

1. Navigate to the project file directory
2. Install dependencies through `yarn install`
3. Start the development environment through `yarn dev`
   This will start a server on `localhost:3000`

Building the application

Assuming you are in the root project file, build the application through the `yarn build` command. This will take several seconds to complete. If you wish you can then start the application on `localhost:3000` through “yarn start”

Deploying the application

If Vercel has already been configured skip to step 8:

1. On the Vercel dashboard select the "Add New..." dropdown and select project.
2. Import the project’s repository through Vercel’s Git Import tool.
3. Adjust the name to your likeness.
4. Expand the “Build and Output Setting” and select override for "Install Command." Enter `yarn install` into the now enabled text box.
5. Fill in all environment variables from section 3.7.1.
6. Select deploy
7. Configure domains as required.
8. For future deployments, pushing to the main branch of GitHub will automatically deploy that version to production. This is a friendly reminder never to push code directly to the main branch but to submit pull requests instead.

Frontend & API

Front-End Page Example

Path: `[locale]/ga4/page.tsx`

Description:

This represents a front-end page for displaying Google Analytics 4 data, customized for different locales. The `[locale]` segment is a dynamic route parameter indicating the language/locale of the page.

Contents:

- **`page.tsx`**: This file contains the React component responsible for rendering the GA4 page. Pages fetch data from API endpoints and render it using components from the `components` directory.

API Route Example

Path: `api/data-source/ga4/route.ts`

Description:

This represents an API route for fetching data related to Google Analytics 4.

Contents:

- **`route.ts`**: This file defines the API route using Next.js API routes. It handles incoming HTTP requests, interacts with databases or external APIs, and sends back the appropriate responses.

File Structure Overview:

- **`src/`**: The root directory for all source code.
  - **`app/`**: Contains all front-end and back-end logic.
    - **`[locale]/`**: Dynamic route parameter for language/locale customization.
      - **`auth/`**: Authentication-related logic.
        - **`signin/`**: Sign-in related pages and logic.
      - **`ga4/`**, **`reports/`**, **`settings/`**, **`social-dashboard/`**, **`year-comparison/`**: Pages for various dashboard views.
    - **`api/`**: Contains all API routes.
      - **`auth/`**: Authentication-related API routes.
        - **`[...nextauth]/`**: Dynamic API route for handling various authentication actions.
      - **`data-source/`**: API routes related to fetching data from different sources.
        - **`ga4/`**, **`ga4-by-country/`**, **`ga4-by-page/`**: Routes for fetching GA4 data based on different parameters.
      - **`data-source-providers/`**: Routes for managing data source providers like Google and Meta OAuth.
    - **`lib/`**: Contains utility functions and external dependencies.
      - **`data-sources/`**: Classes for different data sources like GA4 and YouTube.

            - `database/`: Database-related logic.
            - `google-oauth/`, `meta-oauth/`: Logic for managing OAuth with Google and Meta platforms.
            - `report-generation/`: Logic for generating reports.
            - `util/`: Utility functions for various purposes.
        - `components/`: Reusable React components.
            - `dashboard-stat.tsx`, `data-source-dnd.tsx`, ...: Components for building the user interface.
    - `i18n.ts`: Configuration for internationalization.
    - `middleware.ts`: Middleware logic for handling HTTP requests.
- `public/`: Contains static assets like images and fonts.
- `next.config.mjs`: Configuration file for Next.js.
- `postcss.config.js`, `tailwind.config.ts`: Configuration files for PostCSS and Tailwind CSS.
- `package.json`, `yarn.lock`: Dependency management files.
- `README.md`: Documentation for the project.
- `tsconfig.json`: TypeScript configuration file.
- `components.json`: Configuration file for components.json
- `messages`: Directory containing translations for different languages.
    - `de.json`: German language translations.
    - `en.json`: English language translations.
    - `zh.json`: Chinese language translations.

Bibliography

Database Documentation:

- Redis. (n.d.). Redis Commands
- PG. npm. (n.d.-b). PG npm package
- PostgreSQL 15.6 documentation. PostgreSQL Documentation
- The SQL language. PostgreSQL Documentation
- Vercel KV. Vercel Documentation

General Development Documentation:

- Next.js. Docs | Next.js
- Getting started. NextAuth.js
- Googleapis. npm. (n.d.-a). Googleapis npm package
- Shadcn. (n.d.). Introduction
- Tailwind CSS. Documentation

API Docs

Table of Contents

Table of Contents

API Endpoints

Authentication

`GET` `/auth`

Google OAuth2.0

Sample Request

`https://analytics.stc.group/api/auth`

Data Source Info

`GET` `/data-sources`

Returns the list of data sources and information about them

Sample Request

`https://analytics.stc.group/api/data-sources`

Sample Response

```
[
  {
    "id": "youtube",
    "displayName": "YouTube",
    "color": "#DB4437",
    "authenticationMethod": "youtube",
    "authenticationMethodDisplay": "YouTube",
    "socialDashboard": true
  },
  {
    "id": "google_ads",
    "displayName": "Google Ads",
    "color": "#0F9D58",
    "authenticationMethod": "google",
    "authenticationMethodDisplay": "Google",
    "socialDashboard": false
  },
  ...
]
```

Data Sources

Google Analytics 4

ⓘ `GET` `/data-source/ga4`

Gets metrics for users, viewers, and user engagement duration (seconds)

Query Parameters

Param	Values
start-date	Start of date range to get analytics for. Format YYYY-MM
end-date	End of date range to get analytics for. Format YYYY-MM
resolution	If "full", gets analytics for each calender month within the date range. If "half", gets analytics from the 1st-15th of each month within the date range.
website	`STC`, `HKCC`, `HKIC`
cache?	Defaults to "true". If "true", attempts to use Redis cache. If "false", disables it
db?	Defaults to "true". If "true", attempts to use Postgres database. If "false", disables it

Sample Request

`https://analytics.stc.group/api/data-source/ga4?start-date=2023-12&end-date=2023-12&resolution=full&website=STC`

Sample Response

```
{
  "startDate": { … },
  "endDate": { … },
  "resolution": "full",
  "metrics": [
    "screenPageViews",
    "activeUsers",
    "averageSessionDuration",
    "eventCount"
  ],
  "dateRanges": [ … ],
  "allStats": [
    {
      "month": 12,
      "year": 2023,
      "stats": {
        "dimensions": [

        ],
        "metrics": [
          {
            "screenPageViews": 82648,
            "activeUsers": 38274,
            "averageSessionDuration": 231.83,
            "eventCount": 82017
          }
        ],
        "website": "https://www.stc.group/"
      }
    }
  ]
}
```

Google Analytics 4 by Country

`GET` `/data-source/ga4-by-country`

Gets the country demographics for users, viewers, and user engagement duration (seconds)

Query Parameters

Param	Values
start-date	Start of date range to get analytics for. Format YYYY-MM
end-date	End of date range to get analytics for. Format YYYY-MM
resolution	If "full", gets analytics for each calender month within the date range. If "half", gets analytics from the 1st-15th of each month within the date range.
website	`STC`, `HKCC`, `HKIC`
cache?	Defaults to "true". If "true", attempts to use Redis cache. If "false", disables it
db?	Defaults to "true". If "true", attempts to use Postgres database. If "false", disables it

Sample Request

`https://analytics.stc.group/api/data-source/ga4-by-country?start-date=2023-12&end-date=2023-12&resolution=full&website=STC`

Sample Response

```json
{
  "startDate": { … },
  "endDate": { … },
  "resolution": "full",
  "metrics": [
    "screenPageViews",
    "activeUsers",
    "averageSessionDuration",
    "eventCount"
  ],
  "dateRanges": [ … ],
  "allStats": [
    {
      "month": 12,
      "year": 2023,
      "stats": {
        "metrics": [
          {
            "country": "Vietnam",
            "screenPageViews": 12191,
            "activeUsers": 3928,
            "averageSessionDuration": 246.41,
            "eventCount": 37308
          },
          {
            "country": "Hong Kong",
            "screenPageViews": 7481,
            "activeUsers": 2353,
            "averageSessionDuration": 208.27,
            "eventCount": 24178
          },
          ...
        ],
        "website": "https://www.stc.group/",
        "dimensions": [
          "country"
        ]
      }
    }
  ]
}
```

Google Analytics by Page

`GET` `/data-source/ga4-by-page`

Gets users, viewers, and user engagement duration (seconds) by webpage

Query Parameters

Param	Values
start-date	Start of date range to get analytics for. Format YYYY-MM
end-date	End of date range to get analytics for. Format YYYY-MM
resolution	If "full", gets analytics for each calender month within the date range. If "half", gets analytics from the 1st-15th of each month within the date range.
website	`STC`, `HKCC`, `HKIC`
cache?	Defaults to "true". If "true", attempts to use Redis cache. If "false", disables it
db?	Defaults to "true". If "true", attempts to use Postgres database. If "false", disables it

Sample Request

`https://analytics.stc.group/api/data-source/ga4-by-country?start-date=2023-12&end-date=2023-12&resolution=full&website=STC`

Sample Response

```
{
  "startDate": { ... },
  "endDate": { ... },
  "resolution": "full",
  "metrics": [
    "screenPageViews",
    "activeUsers",
    "averageSessionDuration",
    "eventCount"
  ],
  "dateRanges": [ ... ],
  "allStats": [
    {
      "month": 12,
      "year": 2023,
      "stats": {
        "metrics": [
          {
            "pagePath": "/en",
            "screenPageViews": 7532,
            "activeUsers": 4606,
            "averageSessionDuration": 178.66,
            "eventCount": 23111
          },
          {
            "pagePath": "/en/contact",
            "screenPageViews": 3609,
            "activeUsers": 2447,
            "averageSessionDuration": 50.69,
            "eventCount": 14761
          },
         ...
        ],
        "website": "https://www.stc.group/",
        "dimensions": [
          "pagePath"
        ]
      }
    }
  ]
}
```

Instagram

ⓘ `GET` `/data-source/instagram`

Gets instagram insights for the STC user account

Query Parameters

Param	Values
start-date	Start of date range to get analytics for. Format YYYY-MM
end-date	End of date range to get analytics for. Format YYYY-MM
resolution	If "full", gets analytics for each calendar month within the date range. If "half", gets analytics from the 1st-15th of each month within the date range.
cache?	Defaults to "true". If "true", attempts to use Redis cache. If "false", disables it
db?	Defaults to "true". If "true", attempts to use Postgres database. If "false", disables it

Sample Request

`https://analytics.stc.group/api/data-source/instagram?start-date=2023-12&end-date=2023-12&resolution=full`

Sample Response

```
{
  "startDate": { ... },
  "endDate": { ... },
  "resolution": "full",
  "metrics": [
    "impressions",
    "reach",
    "profileViews",
    "likes",
    "comments",
    "shares"
  ],
  "dateRanges": [ ... ],
  "allStats": [
    {
      "month": 12,
      "year": 2023,
      "stats": {
        "impressions": 124,
        "reach": 34,
        "profileViews": 9,
        "likes": 9,
        "comments": 0,
        "shares": 0
      }
    }
  ]
}
```

Facebook

`GET` `/data-source/facebook`

Gets instagram insights for the STC or HKCC user account

Query Parameters

Param	Values
start-date	Start of date range to get analytics for. Format YYYY-MM
end-date	End of date range to get analytics for. Format YYYY-MM
resolution	If "full", gets analytics for each calendar month within the date range. If "half", gets analytics from the 1st-15th of each month within the date range.
account	`STC`, `HKCC`

Param	Values
cache?	Defaults to "true". If "true", attempts to use Redis cache. If "false", disables it
db?	Defaults to "true". If "true", attempts to use Postgres database. If "false", disables it

Sample Request

`https://analytics.stc.group/api/data-source/facebook?start-date=2023-12&end-date=2023-12&resolution=full&account=STC`

Sample Response

```
{
  "startDate": { … },
  "endDate": { … },
  "resolution": "full",
  "metrics": [
    "comments",
    "impressions",
    "likes",
    "shares"
  ],
  "dateRanges": [ … ],
  "allStats": [
    {
      "month": 12,
      "year": 2023,
      "stats": {
        "comments": 0,
        "impressions": 857,
        "likes": 37,
        "shares": 0
      }
    }
  ]
}
```

YouTube

`GET` `/data-source/youtube`

Gets youtube analytics for the STC Youtube account

Query Parameters

Param	Values
start-date	Start of date range to get analytics for. Format YYYY-MM
end-date	End of date range to get analytics for. Format YYYY-MM
resolution	If "full", gets analytics for each calender month within the date range. If "half", gets analytics from the 1st-15th of each month within the date range.
cache?	Defaults to "true". If "true", attempts to use Redis cache. If "false", disables it
db?	Defaults to "true". If "true", attempts to use Postgres database. If "false", disables it

Sample Request

`https://analytics.stc.group/api/data-source/youtube?start-date=2023-12&end-date=2023-12&resolution=full`

Sample Response

```
{
  "startDate": { ... },
  "endDate": { ... },
  "resolution": "full",
  "metrics": [
    "comments",
    "impressions",
    "likes",
    "shares"
  ],
  "dateRanges": [ ... ],
  "allStats": [
    {
      "month": 12,
      "year": 2023,
      "stats": {
        "comments": 0,
        "impressions": 433,
        "likes": 1,
        "shares": 4
      }
    }
  ]
}
```

User Management

ⓘ `GET` `/users`

Gets list of all dashboard users in the the database. (Requires Admin role)

Sample Request

`https://analytics.stc.group/api/users`

Sample Response

```
[
  {
    "user_id": "eb624aa7-a081-4042-9b6f-5284cc2e4905",
    "name": "John Doe",
    "email": "johndoe@example.com",
    "image": "",
    "role": "default"
  },
  {
    "user_id": "e19f707a-ba93-4233-8bb2-ae69966cbed3",
    "name": "Jane Doe",
    "email": "janedoe@example.com",
    "image": "",
    "role": "admin"
  },
   ...
]
```

✓ `POST` `/user`

Inserts a new user into the database. (Requires Admin role)

Request Body

Key	Value
email	The email of the user being added.
role	The role of the user bing added. Either `default` or `admin`

Sample Request

`https://analytics.stc.group/api/user`

Sample Response

```json
{
    "user_id": "ccbefca8-d03e-4baa-918b-556005ebd411",
    "name": null,
    "email": "newemail@example.com",
    "image": null,
    "role": "default"
}
```

❌ `DELETE` `/user`

Removes a user from the database. (Requires Admin role)

Request Body

Key	Value
user_id	The user ID of the user being removed.

Sample Request

`https://analytics.stc.group/api/user`

✏ `PUT` `/user`

Updates the role of a specified user in the database. (Requires Admin role)

Request Body

Key	Value
user_id	The user ID of the user being removed.
role	The role of the user bing added. Either `default` or `admin`

Sample Request

`https://analytics.stc.group/api/user`

Report Generation

✓ `POST` `/generate-report`

Updates the role of a specified user in the database. (Requires Admin role)

Request Body

Key	Value
startDate	`{year: XXXX, month: XX}`
endDate	`{year: XXXX, month: XX}`
metrics	`["STC", "HKCC", ... ]`
resolution	`half`, `full`, `both`
socialMedia	`{dataSources: ... , metrics: ["reach", "likes", ... ]}`

Sample Request

`https://analytics.stc.group/api/generate-report`
API response is an `.xlsx` download