
FIT3047 - Industry Experience (S2 2021)

Schoolbox Technical Documentation



Prepared by:

Dane Rainbird

Henry Barnett

Hao (Henry) Zhang

Pinhui (Phyllis) Ji

Prepared for:

Andy Ryan (Schoolbox)

Schoolbox Technical Staff

FIT3048 Mentors

Version: 1.9, 03/10/2021

i - Table Of Contents

1 - Introduction	4
2 - The Website	5
2.1 - Website Framework and Architecture	5
2.1.1 - CakePHP	5
2.1.2 - Other Technical Architecture Requirements	5
2.2 - Plugins / Libraries / APIs	7
2.2.1 - Back-end Plugins / Libraries	7
2.2.1.1 - CakePHP-Social-Auth	7
2.2.2 - Front-end Plugins / Libraries	8
2.2.2.1 - SoftUI Dashboard	8
2.2.2.2 - Font-Awesome	8
2.2.2.3 - Highlight.js	9
2.2.2.4 - DataTables	10
2.3 - Security	11
2.3.1 - Authentication and Authorization	11
2.3.2 - SSL Certificates	11
3 - The Database	13
3.1 - Database Setup	13
3.1.1 - Setup using Provided SQL Files	14
3.1.2 - Manual Setup	14
3.2 - Database Structure	18
3.2.1 - Entity Relationship Diagram	18
3.3 - Database Backup and Restoration	19
3.3.1 - Database Backup	19
3.3.2 - Database Restoration	19
4 - Recurring Tasks	20
4.1 - Querying Data from the Server	20
4.1.1 - The Job	20
4.1.2 - Frequency of Execution	21
4.2 - Cleanup of Data from the Database	22
4.2.1 - The Job	22
4.2.2 - Frequency of Execution	22
5 - Website Hosting and Deployment	23
5.1 - Web Server Requirements	23
5.2 - Initial Setup	24
5.2.1 - Prerequisites	24

5.2.2 - Creating the Database	24
5.2.3 - Obtaining Google OAuth Keys	24
5.3 - Setting up the Website	27
5.3.1 - Obtaining the Code	27
5.3.2 - Updating / Installing Dependencies	27
5.3.3 - Configuration	28
5.3.3.1 - Database Connection	28
5.3.3.2 - Google Login	28
5.3.3.3 - Obtaining First Dataset	29
5.3.4 - Scheduling Tasks	29
5.3.5 - Creating an Admin Account	30
5 - Assorted Design / Implementation Documentation	32

1 - Introduction

This document is intended to act as a point of reference for the technical components, platforms, individual components, and deployment of the new statistical dashboard for Schoolbox. It will provide a comprehensive, detailed view of the system's components, deployment details, database structure, and other necessary technical information required to understand, maintain, and upgrade the system following the formal handover of the system from Team 26 ("Team Marble") to Schoolbox.

As such, this report will be broken into five main sections, covering the website, the database, recurring tasks, hosting requirements, and assorted design and implementation diagrams. Where relevant, pictograms and videos will be provided to clearly illustrate required actions.

This document presumes the reader has technical knowledge and experience relating to website development, web hosting / deployment, and general computer literacy.

2 - The Website

The Statistical Dashboard Website is the main component of the new system. This section will thus cover three main sections regarding the website: the website framework and architecture, plugins / libraries / APIs, and security.

2.1 - Website Framework and Architecture

The Schoolbox Statistical Dashboard website was developed using CakePHP 4.2, a Model-View-Controller (MVC) architecture for web applications, as a backend, with traditional web development technologies (such as HTML and JavaScript) being used for the front-end.

2.1.1 - CakePHP

The Schoolbox Statistical System makes use of CakePHP 4.2, and utilises an MVC architecture for handling requests to the server.

As such, requests to the server are handled by CakePHP as follows:

- A visitor to the website will request for a specific page
- The “Controller” for this page will request with the associated “Model”(s) for the requested page, as well as perform any other back-end operations necessary
- The “Controller” will then use any details it has generated or received from the “Model” in order to generate a “View”
- The “View” will then be sent and displayed to the website visitor.

CakePHP’s documentation (known as the “Cookbook”) can be found at the CakePHP site:

<https://book.cakephp.org/4/>

The Cookbook also includes a more detailed description at how CakePHP’s “request cycle” works:

<https://book.cakephp.org/4/en/intro.html#cakephp-request-cycle>

2.1.2 - Other Technical Architecture Requirements

The Schoolbox Statistical Dashboard website also requires various other software modules to be installed in order to run. The name, minimum version number, and purpose of these can be found below:

Software	Minimum Version	Purpose
MySQL / MariaDB	15.1	The database for the website.
PHP	7.2	The programming language and interface used by CakePHP
Composer	1.10.6	The dependency management tool used by CakePHP

Table 1 - *Software Requirements for the Schoolbox Statistical Dashboard Website*

In addition, some various configuration changes must be made to PHP in order for it to work with CakePHP. Please refer to the instructions at the below link:

<https://book.cakephp.org/4/en/installation.html>

2.2 - Plugins / Libraries / APIs

The Schoolbox Statistical Dashboard website makes use of a few assorted plugins, both on the front-end and back-end, which assist in both general website functionality, as well as the appearance of the website.

2.2.1 - Back-end Plugins / Libraries

External back-end plugins for CakePHP are contained within the vendor / directory, and are obtained via the Composer Dependency manager mentioned in [2.1.2 - Other Technical Architecture Requirements](#). Although developers generally provide specific installation instructions for their plugins, general instructions for installing plugins can be found at the link below:

<https://getcomposer.org/doc/03-cli.md#require>

Generally, back-end plugins are loaded by adding a reference to them within the bootstrap() function within src/Application.php. Please see the below link for more details:

<https://book.cakephp.org/4/en/plugins.html>

2.2.1.1 - CakePHP-Social-Auth

The Schoolbox Statistical Dashboard makes extensive use of one major CakePHP plugin - ADmad's cakephp-social-auth.

cakephp-social-auth is a "middleware" that is responsible for the Google authentication system that has been implemented into the system, and is therefore directly responsible for allowing users to log in.

cakephp-social-auth can be installed by executing the following commands from the root folder of the website:

```
composer require admad/cakephp-social-auth
bin/cake plugin load ADmad/SocialAuth
```

Detailed setup and configuration instructions can be found at the plugin's GitHub page:

<https://github.com/ADmad/cakephp-social-auth>

2.2.2 - Front-end Plugins / Libraries

Front-end libraries for CakePHP are generally contained within the webroot/ directory, and are obtained either through the Composer Dependency manager mentioned in [2.1.2 - Other Technical Architecture Requirements](#), or are simply downloaded from the Internet.

In most cases, if files are stored within the webroot/ directory, these plugins / libraries are loaded through the use of the HTML Helper, which is a CakePHP “Helper”, designed to assist with inserting HTML elements into a View.

More details for HTML Helper, and CakePHP Helpers in general, can be found at the below links:

<https://book.cakephp.org/4/en/views/helpers/html.html>

<https://book.cakephp.org/4/en/views/helpers.html>

2.2.2.1 - SoftUI Dashboard

The Schoolbox Statistical Dashboard makes use of the free HTML template SoftUI Dashboard Version 1.0.3 by Creative Tim for its overall appearance. This template was chosen as it provides a full suite of HTML design elements that make it easy to add and modify items on the page.

All files relating to the template exist within the webroot/ directory, under the CSS/, JS/, webfonts/, and img/ directories, and are loaded into the template files used by CakePHP, within the templates/layouts/ directory with HTML Helper.

SoftUI Dashboard is powered by Bootstrap 5, a design framework for websites, whose documentation can be found at the below link:

<https://getbootstrap.com/docs/>

Furthermore, Creative Tim maintains its own documentation for this template’s specific features and modifications from the base Bootstrap 5 styles, which can be found below:

<https://www.creative-tim.com/learning-lab/bootstrap/overview/soft-ui-dashboard>

2.2.2.2 - Font-Awesome

The Schoolbox Statistical Dashboard makes use of the free version of “Font-Awesome” Version 5.15.4, a CSS-based icon pack, in order to display pictograms on various pages for various actions (such as a silhouette of a user next to the “Sign out” button).

All files relating to Font-Awesome exist within the `webroot/webfonts/` directory, and are loaded into the template files used by CakePHP, within the `templates/layouts/` directory with HTML Helper.

Font-Awesome provides its own documentation (including a list of icons), which can be found at the below link:

<https://fontawesome.com/v5.15/how-to-use/on-the-web/>

2.2.2.3 - Highlight.js

The Schoolbox Statistical Dashboard makes use of “Highlight.js” Version 11.2.0, a JavaScript library that performs “syntax highlighting” for code chunks displayed to end-users.

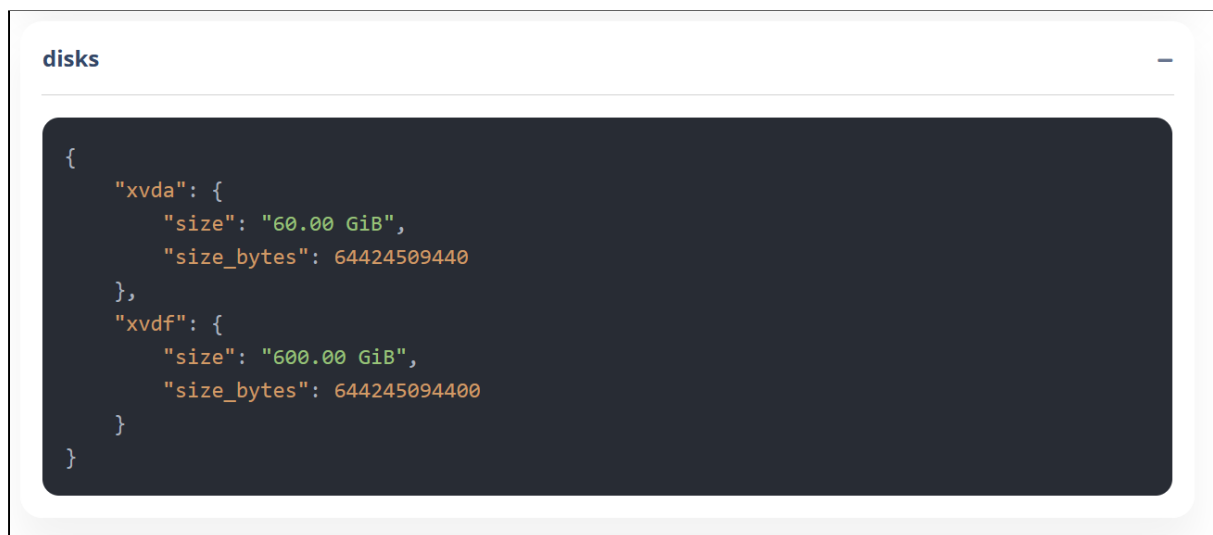


Figure 1 - An example of Highlight.js syntax highlighting for JSON data from a Schoolbox Server.

All files relating to Highlight.js exist within the `webroot/css/` and `webroot/js/` directories, and are loaded into the template files used by CakePHP, within the `templates/layouts/` directory with HTML Helper.

Highlight.js provides its own documentation, including a list of supported languages and themes, which can be found at the below link:

<https://highlightjs.org/usage/>

Please note that the currently installed files only support syntax highlighting for JSON data, and only provide the “atom-dark-one” theme.

2.2.2.4 - DataTables

The Schoolbox Statistical Dashboard makes use of “DataTables” Version 1.11.3, a JavaScript library that converts standard HTML tables into interactive tables, with support for sorting, searching, pagination, and more.

Presently, the version of DataTables used on the dashboard is hosted on the DataTables CDN, with the “Buttons”, “Responsive”, and “Scroller” plugins, as well as making use of the Bootstrap 5 theme set.

Each table is individually initialized based on the data contained within it (e.g. some tables have a default sort on the first column, whilst others may sort on their second column by default)

DataTables provides its own documentation, including a startup guide and all API functions, which can be found at the below link:

<https://datatables.net/manual/index>

2.3 - Security

The Schoolbox Statistical Dashboard website has been designed with data security and sovereignty in mind, and to this end has a number of security features.

2.3.1 - Authentication and Authorization

As outlined in [2.2.1.1 - CakePHP-Social-Auth](#), the Schoolbox Statistical Dashboard makes use of Google authentication to ensure that only Schoolbox users may sign into the site. As such, any attempts to sign into the site (or to visit any page on the site), will redirect users to a Google sign-in page.

More information regarding the Google “OAuth” protocol and it’s authentication process can be found at the below link:

<https://developers.google.com/identity/protocols/oauth2>

Furthermore, the use of Google Authentication requires a Google OAuth private and client keys, which are installed into `src/Application.php`. More detailed instructions will be provided in [5 - Website Hosting and Deployment](#).

Once authenticated, users are broken into two authorization groups - administrators, and regular users:

- **Administrators** - Administrators have the ability to delete existing data sets, as well as to promote other users to the role of administrator (or to set them back to a regular user if necessary)
- **Regular Users** - Regular users are simply able to sign into the site and view the data within it, but are unable to delete data (nor promote / demote other users)

User types are defined by a BOOLEAN value (“isAdmin”) within the database’s USERS table.

While administrators are able to promote regular users to administrators within the website itself, it is also possible to promote users from CakePHP’s command line interface. This will be covered later in [5 - Website Hosting and Deployment](#).

2.3.2 - SSL Certificates

Similarly to existing Schoolbox websites, an SSL Certificate is installed onto the system, which has been provided by Let’s Encrypt. An SSL Certificate ensures that communication to and from the Schoolbox Statistical Dashboard is encrypted.

The Let's Encrypt certificate has been provided and installed by "Certbot", a command-line tool for automating the deployment of certificates, and as such, the certificate is set to automatically renew when it expires.

More detailed information regarding SSL and Let's Encrypt can be found below:

<https://letsencrypt.org/docs/>

Furthemore, detailed instructions and setup details for "Certbot" can be found at the below link:

<https://certbot.eff.org/instructions>

3 - The Database

The Schoolbox Statistical Dashboard utilises a MySQL / MariaDB (version 15.1) installation in order to store user details and historical data sets, and is connected to directly via CakePHP's Object-relational Mapping (ORM) functionality.

By default, the website looks for a database named `u21s1026_schoolbox` on the same server as the webserver (i.e. localhost), however this can be changed by modifying the `app.php` configuration file. Details regarding this process (and other general details related to databases in CakePHP) can be found below:

<https://book.cakephp.org/4/en/orm/database-basics.html#database-configuration>

<https://book.cakephp.org/4/en/orm.html>

This section will thus cover setting up the database, the database structure, and backing up / restoring the database.

3.1 - Database Setup

In order to set up the database, MySQL must be installed onto the server, be exposed to a specific port number (by default MySQL uses 3306), and the default setup be completed. This process changes depending on the type of server in use, however generic instructions for this process can be found below:

<https://dev.mysql.com/doc/mysql-installation-excerpt/5.7/en/>

Furthermore, in order to ensure the overall security of both the website's database, and the web hosting server, we strongly recommend creating a new user account with read and write access to the "u21s1026_schoolbox" database. Instructions for creating database users and applying permissions can be found below:

<https://dev.mysql.com/doc/refman/8.0/en/create-user.html>

<https://dev.mysql.com/doc/refman/8.0/en/grant.html>

Once MySQL is successfully installed, the database and it's associated tables can be created either by manually creating the tables, or by using the provided SQL structure file within [5 - Assorted Design / Implementation Documentation](#).

3.1.1 - Setup using Provided SQL Files

Once the database structure file has been downloaded, upload the file to your server, and navigate to the directory containing the file. From here, execute the following command in order to load the database file:

```
$ mysql -u {account_username} -p {account_password} <
u21s1026_schoolbox_iter3complete.sql
```

Please note that the script provided will automatically generate the database and its tables, and overwrite any existing databases that may already exist which use the name "u21s1026_schoolbox".

Upon completion, you can check if the database exists by opening MySQL and seeing if the table structure has been created successfully. Instructions for this process can be found below:

<https://dev.mysql.com/doc/refman/5.7/en/sys-table-exists.html>

More details on importing files into MySQL can be found below:

<https://www.digitalocean.com/community/tutorials/how-to-import-and-export-databases-in-mysql-or-mariadb>

3.1.2 - Manual Setup

If there are any errors with using the provided script files, or if you are using a non-standard MySQL install (or any other database type, such as SQLite), you can manually create all four tables as required.

Below is a table of the tables contained within the database "u21s1026_schoolbox", along with their associated data types. Please refer to the documentation for the database type you are using to confirm the data types and commands required to create the tables:

historical_facts Table

Column Name	Data Type	Extra Details
id	int(11)	AUTO_INCREMENT Primary Key

timestamp	datetime	N/A
schoolbox_totalusers	longtext	utf8mb4_bin
schoolbox_config_site_type	longtext	utf8mb4_bin
schoolbox_users_student	longtext	utf8mb4_bin
schoolbox_users_staff	longtext	utf8mb4_bin
schoolbox_users_parent	longtext	utf8mb4_bin
schoolbox_totalcampus	longtext	utf8mb4_bin
schoolbox_package_version	longtext	utf8mb4_bin
schoolboxdev_package_version	longtext	utf8mb4_bin
schoolbox_config_site_version	longtext	utf8mb4_bin
virtual	longtext	utf8mb4_bin
lsbdistdescription	longtext	utf8mb4_bin
kernelmajversion	longtext	utf8mb4_bin
kernelrelease	longtext	utf8mb4_bin
php_cli_version	longtext	utf8mb4_bin
mysql_extra_version	longtext	utf8mb4_bin
processorcount	longtext	utf8mb4_bin
memorysize	longtext	utf8mb4_bin
schoolbox_config_date_timezone	longtext	utf8mb4_bin
schoolbox_config_external_type	longtext	utf8mb4_bin
schoolbox_first_file_upload_year	longtext	utf8mb4_bin

social_profiles Table

Column Name	Data Type	Extra Details
id	int(11)	AUTO_INCREMENT Primary Key
user_id	int(11)	Index Key

provider	varchar(255)	utf8_general_ci
access_token	blob	N/A
identifier	varchar(255)	utf8_general_ci
username	varchar(255)	utf8_general_ci
first_name	varchar(255)	utf8_general_ci
last_name	varchar(255)	utf8_general_ci
full_name	varchar(255)	utf8_general_ci
email	varchar(255)	utf8_general_ci
birth_date	date	N/A
gender	varchar(255)	utf8_general_ci
picture_url	varchar(255)	utf8_general_ci
email_verified	tinyint(1)	N/A
created	datetime	N/A
modified	datetime	N/A

users Table

Column Name	Data Type	Extra Details
id	int(11)	AUTO_INCREMENT Primary Key
email	varchar(320)	utf8mb4_general_ci
isAdmin	tinyint(1)	Defaults of 0

a_dmad_social_auth_phinxlog Table

Column Name	Data Type	Extra Details
version	bigint(20)	Primary Key
migration_name	varchar(100)	utf8_general_ci Defaults to NULL

start_time	timestamp	Defaults to NULL
end_time	timestamp	Defaults to NULL
breakpoint	tinyint(1)	N/A

3.2 - Database Structure

In order to better understand the database, please find within this section an Entity-Relationship Diagram indicating the layout of the database used by the Schoolbox Statistical Dashboard system.

3.2.1 - Entity Relationship Diagram

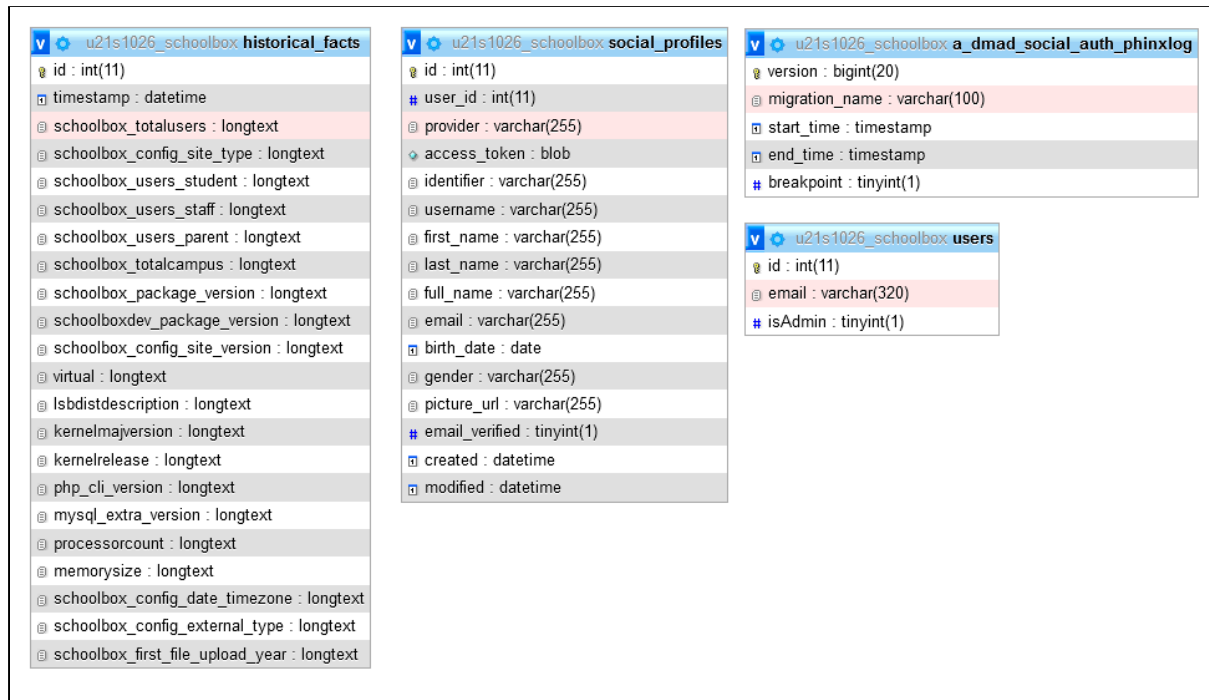


Figure 2 - An "ERD" of the Schoolbox Statistical Dashboard System's Database

3.3 - Database Backup and Restoration

In order to ensure that the Schoolbox Statistical Dashboard's database does not need to be recreated in case of a server failure or other unforeseen situation, we highly recommend taking regular backups of the database and its contents.

3.3.1 - Database Backup

Backing up the database (known as “dumping” the SQL files) can be performed through the use of the following command:

```
$ mysqldump -u {account_username} -p {account_password}  
u21s1026_schoolbox > backup.sql
```

This will create a file named `backup.sql` within the current working directory. Ensure that this file is stored in a safe location, as it can be used to restore the database.

More details related to `mysqldump` can be found below:

<https://dev.mysql.com/doc/refman/8.0/en/mysqldump.html>

3.3.2 - Database Restoration

Restoring the previously generated backup file can be achieved in the same way as in [3.1.1 - Setup using Provided SQL Files](#). Please execute the following command:

```
$ mysql -u {account_username} -p {account_password} < backup.sql
```

More details on importing files into MySQL can be found below:

<https://www.digitalocean.com/community/tutorials/how-to-import-and-export-databases-in-mysql-or-mariadb>

4 - Recurring Tasks

In order to allow the Statistical Dashboard to constantly have up-to-date data regarding the Schoolbox fleet, two main commands have been created, and are scheduled to run on a recurring basis (known as “cronjobs” on Unix systems, or “scheduled tasks” on Windows systems).

More information relating to cronjobs and their functionality can be found in the documentation for whichever Linux / Windows distribution you are using. However, the documentation for Ubuntu (the current server type at time of writing) can be found below:

<https://help.ubuntu.com/community/CronHowto>

The cronjobs used by the system rely on executing a CakePHP command, located within the src/Command directory. More information on CakePHP commands can be found below:

<https://book.cakephp.org/4/en/console-commands.html>

This section will therefore outline what each cronjob does, and the scheduled frequency of each.

4.1 - Querying Data from the Server

In order to have up-to-date data on the dashboard, data is periodically queried from the Schoolbox PuppetDB servers, has data analytics performed upon it, and then stored as JSON dumps within the dashboard’s database.

4.1.1 - The Job

The data querying job is handled by the queryPuppetDb command. This command opens a connection to the staging and production PuppetDB fact servers (<https://puppetdb.stg.1.schoolbox.com.au> and <https://puppetdb.prn.1.schoolbox.com.au>, respectively) and pulls data for a known list of facts, before performing data manipulation, and storing the data as a JSON dump within the `historical_facts` table within the database..

schoolbox_totalusers

```
{"totalUsersFleetCount":678812,"totalUsers":  
{"0":26,"3":35,"5":17,"2":53,"4":25,"10":4,"1":27,"6":5,"9":4,"7":3,"8":2,"13":1}}
```

Figure 3 - *An example JSON dump for the "schoolbox_totalusers" fact, as generated by the queryPuppetDB command.*

Pulling data from the PuppetDB is handled through the use of the PuppetDB API. Documentation for this API can be found at the below link:

<https://puppet.com/docs/puppetdb/5.2/api/query/v4/query.html>

4.1.2 - Frequency of Execution

Presently, the data querying cronjob is set to execute every half-an-hour (for a total of 48 executions per day), with each execution taking roughly 12 seconds (depending on internet speeds, and other operations taking place on the server).

This is to ensure relatively up-to-date data, with enough granularity that historical data sets might be compared.

4.2 - Cleanup of Data from the Database

In order to ensure fast load times on the dashboard, as well as to prevent unnecessary file storage, historical data sets are deleted from the system automatically, according to the following rules:

- After a month, reduce to one historical data set per day (reduction from ~744 entries to ~30)
- After 3 months, reduce to one historical data set per week (reduction from ~30 to 4)
- After 6 months, reduce to one historical data set per month (reduction from 4 to 1)

4.2.1 - The Job

The data deletion job is handled by the `cleanupCachedData` command. This command pulls all the stored data within the `historical_facts` table within the database, before deleting entries based on their timestamps and the rules covered above.

4.2.2 - Frequency of Execution

Presently, the data cleanup cronjob is set to execute every day at 00:00 (12 midnight), with each execution taking roughly 5 seconds (depending on other operations taking place on the server).

This is to ensure that enough data granularity within the historical data sets are preserved for both short and long term comparison, without inflating data storage sizes within the database (as well as increasing server load time)

5 - Website Hosting and Deployment

In order to allow the Statistical Dashboard to be usable on more than just a local machine, the dashboard should be hosted on either a dedicated web hosting platform or server. Thus, this section will provide a non-specific overview of the steps required to deploy the dashboard, as well as any technical requirements that must be taken into account.

5.1 - Web Server Requirements

At minimum, the web server / hosting provider must be capable of running the below technologies simultaneously:

- A web server, such as Apache or Nginx
- A MySQL database, such as MariaDB
- “Cronjobs” or scheduled tasks

Furthermore, the chosen platform must be capable of running the CakePHP-specific requirements outlined in [2.1.2 - Other Technical Architecture Requirements](#).

It is also recommended that the server be capable of handling SSL communication (i.e. port 443 can be opened), and that an SSL certificate be installed, in order to ensure encrypted server-to-client communication.

5.2 - Initial Setup

Before beginning to deploy the Schoolbox Statistical Dashboard, it is important to ensure that the server is configured properly, and that all prerequisites have been met. This section will thus cover the steps required to prepare before initial deployment.

5.2.1 - Prerequisites

As mentioned previously, the dashboard is powered by CakePHP, a MySQL database, and a web server and as such, these technologies must be installed (or confirmed to be running in the case of using a web hosting provider). Installation instructions will vary depending on the software chosen, and the operating system on which the software will be installed.

Please refer to [2.1 - Website Framework and Architecture](#) to ensure that you install the correct software versions required for the dashboard.

Furthermore, a full domain name is required for both allowing Google login (see [5.2.3 - Obtaining Google OAuth Keys](#)), as well as for obtaining an SSL certificate.

5.2.2 - Creating the Database

In order to store the data that will be required by the dashboard, the database structure will need to be created, alongside a MySQL user that has read and write permissions to the tables used by the dashboard.

Steps to create the database, it's tables, and, and a user with appropriate permissions can be found in [3.1 - Database Setup](#).

If the website is being migrated from an existing server, with an existing set of data, it is recommended that the data within the database be backed up and restored as well - please refer to [3.3 - Database Backup and Restoration](#).

Please note: Although creating a user is not strictly necessary, using the default / root user for MySQL is a security vulnerability, and could be used to perform CRUD operations on any other database or table stored on the server / hosting provider.

5.2.3 - Obtaining Google OAuth Keys

In order to handle Google authentication to the website, a Google OAuth Client Key (and it's associated private key) is required to be generated. In order to do this, please follow the instructions provided by Google below:

<https://support.google.com/cloud/answer/6158849?hl=en>

Please ensure that once you complete the steps outlined above, you make a note of the client ID and client secret, both of which are generally in below format:

```
<client_id / client_secret>.apps.googleusercontent.com
```

As noted by Google, it is required to generate an “OAuth Consent Screen”, which is the screen displayed to users during the sign in process:

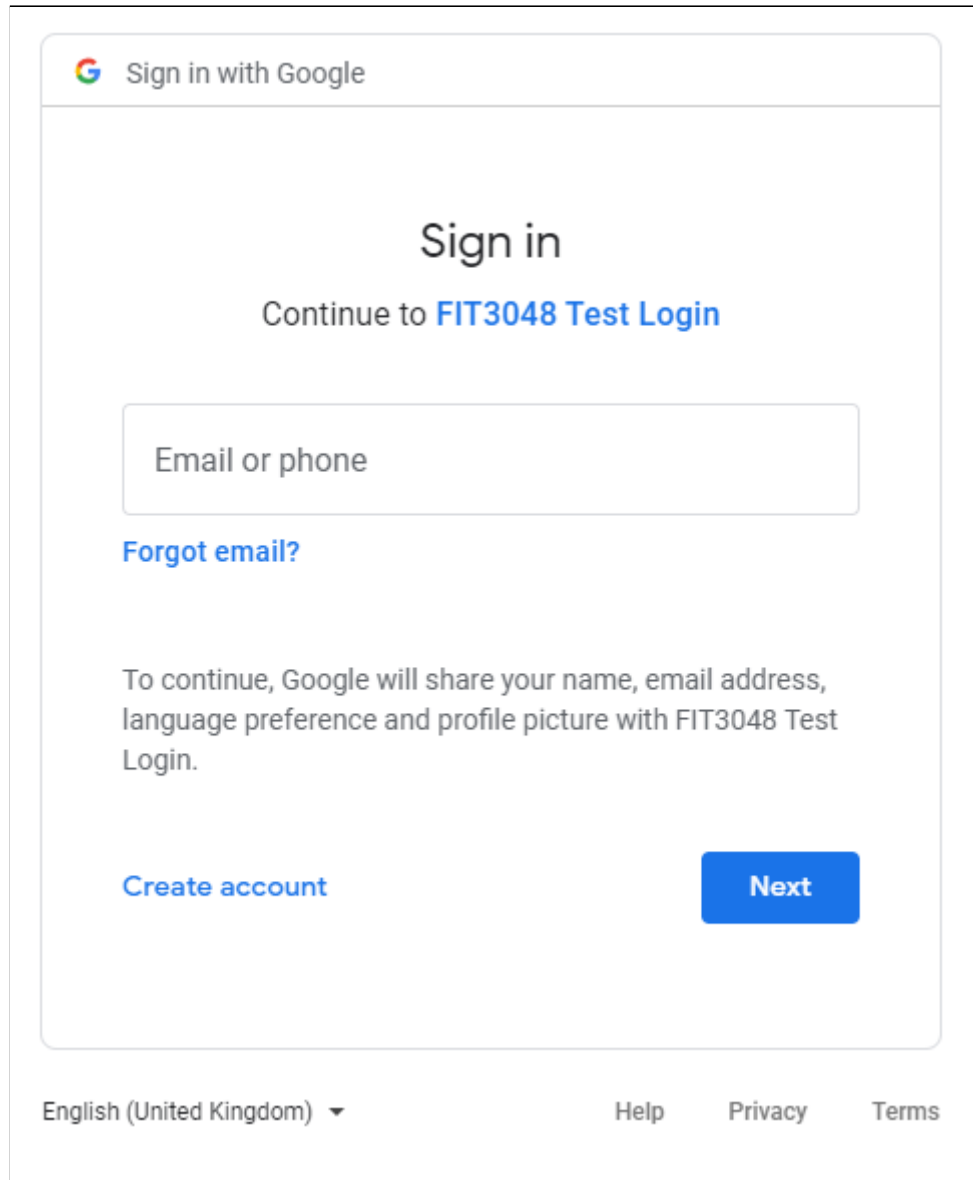


Figure 4 - The Google “OAuth Consent Screen”

During the setup of this OAuth Consent Screen, you must supply Google with an “Authorised redirect URI”, following the form below:

`https://<domain_name>.<TLD>/social-auth/callback/google`

This will ensure that your domain is able to be returned to after a successful authentication with Google.

5.3 - Setting up the Website

Once all prerequisites are met, the Schoolbox Statistical Dashboard can be deployed, and configured to work with the specifics of whichever server it is currently running on. This section will thus cover the steps required to deploy the Schoolbox Statistical Dashboard.

Before beginning to deploy the Schoolbox Statistical Dashboard, it is important to ensure that the server is configured properly, and that all prerequisites have been met. This section will thus cover the steps required to prepare before initial deployment.

5.3.1 - Obtaining the Code

Presently, the code for the website is stored both on a USB flash drive, and on a repository handled by the git Version Control System (VCS). It is recommended that the most up to date version of the dashboard be installed, and thus we recommend the use of “pulling” from the Git repository.

This process, known as “cloning”, will make a copy of all the source code files to a location on the server / web hosting provider. It is recommended that this code be placed within the directory utilized by the web server you set up in [5.2.1 - Prerequisites](#), such as /var/www/html/ for Apache on Linux.

In order to clone the repository, please use the following command, and modify it to use the correct VCS URL:

```
$ git clone https://<git_url>/<repo_name>.git .
```

Please note: the trailing full stop (.) at the end of the above command is very important, as it will clone the files into the current directory, without creating a new directory.

More details regarding the Git clone command can be found below:

<https://www.git-scm.com/docs/git-clone>

5.3.2 - Updating / Installing Dependencies

Once the code has been cloned onto the server / web hosting provider, the CakePHP dependencies will need to be updated / installed, if they are not already. In order to do this, please navigate to the directory containing the source code, and execute the below command:

```
$ composer install
```

Please wait until Composer finishes updating / installing the dependencies. This may take some time if this command is being run for the first time.

More details regarding the Composer update process can be found below:

<https://getcomposer.org/doc/03-cli.md#install-i>

5.3.3 - Configuration

After cloning the code to the web server, some changes must be made in order to ensure the dashboard works correctly.

5.3.3.1 - Database Connection

In order to ensure that the database can connect correctly, the Datasources object must be changed within `config/app.php` to the following:

```
'Datasources' => [
    'default' => [
        'host' => 'localhost',
        'username' => '<mysql_username>',
        'password' => '<mysql_password>',
        'database' => 'u21s1026_schoolbox',
        'url' => env('DATABASE_URL', null),
    ],
]
```

Please ensure that the MySQL username and MySQL password fields are replaced with the values created in [3.1 - Database Setup](#). If using a different database name, please also update the database value.

5.3.3.2 - Google Login

As outlined in [5.2.3 - Obtaining Google OAuth Keys](#), a Google OAuth2 client ID and client secret must be generated. Please update the SocialAuthMiddleware object within `src/Application.php` as per below:

```
'serviceConfig' => [
    'provider' => [
        'google' => [
            'applicationId' => '<client_id>',
```

```
        'applicationSecret' => '<client_secret>',
        'scope' => [
            'https://www.googleapis.com/auth/userinfo.email',
            'https://www.googleapis.com/auth/userinfo.profile',
        ],
    ],
]
```

Please ensure that the two `scope` values remain unchanged, as they are required for successful sign in.

5.3.3.3 - Obtaining First Dataset

To ensure that the web server, database, and network connectivity of the site are working correctly, attempting to obtain a full historical data set is recommended.

Please note: at the time of writing, the two PuppetDB endpoints utilised for obtaining data are `puppetdb.stg.1.schoolbox.com.au` and `puppetdb.prd.1.schoolbox.com.au`. If this ever changes, please modify the URLs within the `sendRequest()` function within `src/Command/QueryServerCommand.php` and `src/Command/FactsController.php`

In order to obtain a full historical data set, please run the following command:

```
$ <server_directory>/bin/cake queryPuppetDb
```

This command will pull data from the Schoolbox PuppetDB servers, and place it within the database, ensuring that both the network connectivity and database connectivity of the web server are functioning as intended.

5.3.4 - Scheduling Tasks

Once the dashboard is working correctly, and can read / write to the database, the two automated tasks (“crojobs”) outlined in [4 - Recurring Tasks](#) must be set up to run on the server in order to fill the website with data.

The method by which tasks are scheduled is dependent on the operating system the web server / web hosting provider is using, though in general terms, Windows Servers make use of “Scheduled Tasks”, whilst Linux servers use `crontab`. This section will

discuss the use of crontab, however more details for both crontab and Scheduled Tasks can be found below:

<https://www.man7.org/linux/man-pages/man5/crontab.5.html>

<https://docs.microsoft.com/en-us/windows/win32/taskschd/using-the-task-scheduler>

Before continuing, it is important to ensure that the cake executable is allowed to be run as a script. On Unix systems, please run the following command to allow the file to be executed.

```
chmod +x <server_directory>/bin/cake
```

Cronjobs can be modified by using the `crontab -e` command. This will open a file for modification that can be edited to include a time value and a corresponding command. After saving this file, the next time that the time value is hit, the command will execute.

As outlined in [4 - Recurring Tasks](#), we recommend the following time values / commands be used for setting up the cronjobs:

Task	Cronjob Timing	Meaning
Querying Data	30 * * * * <server_directory>/bin/cake "queryPuppetDb"	Run the "queryPuppetDb" CakePHP Command every half hour
Deleting Data	0 0 * * * <server_directory>/bin/cake "cleanupCachedData"	Run the "cleanupCachedData" CakePHP Command once a day

5.3.5 - Creating an Admin Account

If installing and configuring the system for the first time, or in the case of database corruption, there will be no user currently set as an administrator, meaning that the user management GUI on the dashboard cannot be accessed. As such, the `setUserAsAdmin` command can be used to manually make a user an administrator from the command line.

Please note: the user you wish to make administrator must have already signed into the system once. Please ensure that the user has first successfully logged in before continuing.

In order to set a user as an admin, please use the following command:

```
$ <server_directory>/bin/cake setUserAsAdmin <email_address>
```

This command will take effect instantly, and from this point onwards this user may modify other user permissions from the dashboard directly.

5 - Assorted Design / Implementation Documentation

Please refer to the below links that will take you to various documentation that has been produced throughout the development of the Schoolbox Statistical Dashboard, which may be of use for further development:

Document	Link
Rough Workflow Mockup	<i>redacted</i>
Dashboard Rough Wireframe	<i>redacted</i>
Implementation Plan	<i>redacted</i>
User Stories / Epics	<i>redacted</i>
Database SQL Structure File	<i>redacted</i>