THE UNIVERSITY OF WESTERN AUSTRALIA

CITS3200 Project Team 18

UWA Academic Skills Drop-in - Client queue flow and usage reporting software

Alex Mai, Jordan Hartley, Frinze Lapuz, Jake Yendell, Alex Hoffman, Liam Hovell

Table of contents

1. What is this project?

1.1 Authors

- Jordan Hartley
- Liam Hovell
- Frinze Lapuz
- Alex Hoffman
- Alex Mai
- Jake Yendell

1.2 References

Albert Einstein photo ref: MPI/Getty Images Diagrams by Jasper Paterson and Simon Dransfield Background art by Jordan Hartley

2. User

2.1 User Documentation

2.1.1 How to access the website

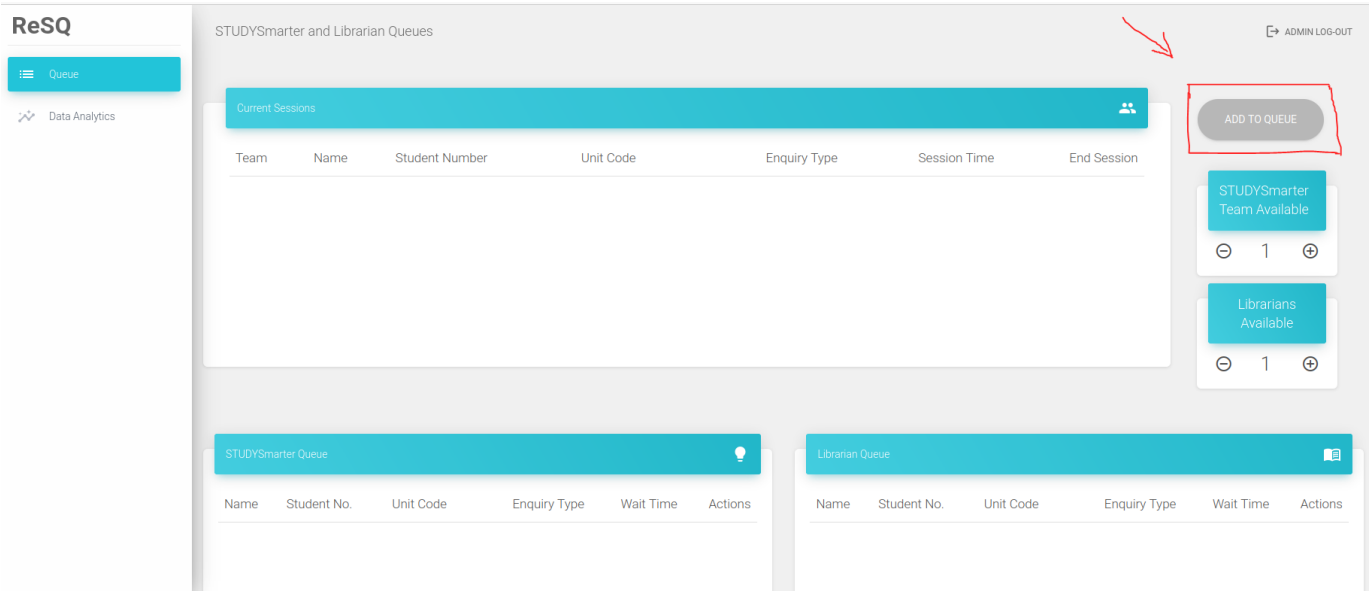
Click here to redirect to the website. User will need username and password to login.
There will be 3 main tabs in the website, which are Queue, Data Analytics and Export Data

2.1.2 Queue

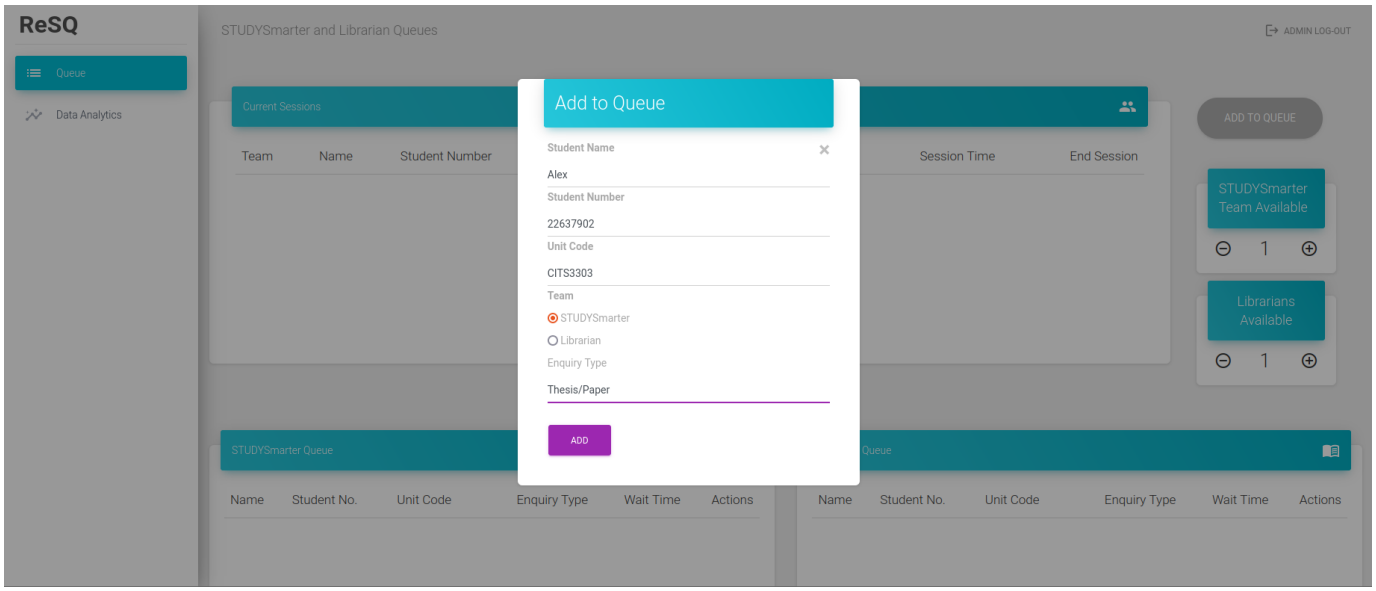
User can manage student sessions in the Queue tab. There are 3 tables, 2 of which are STUDYSmarter queue and Librarian queue. Students will be put into either queue depending on user's choice.

Add to queue

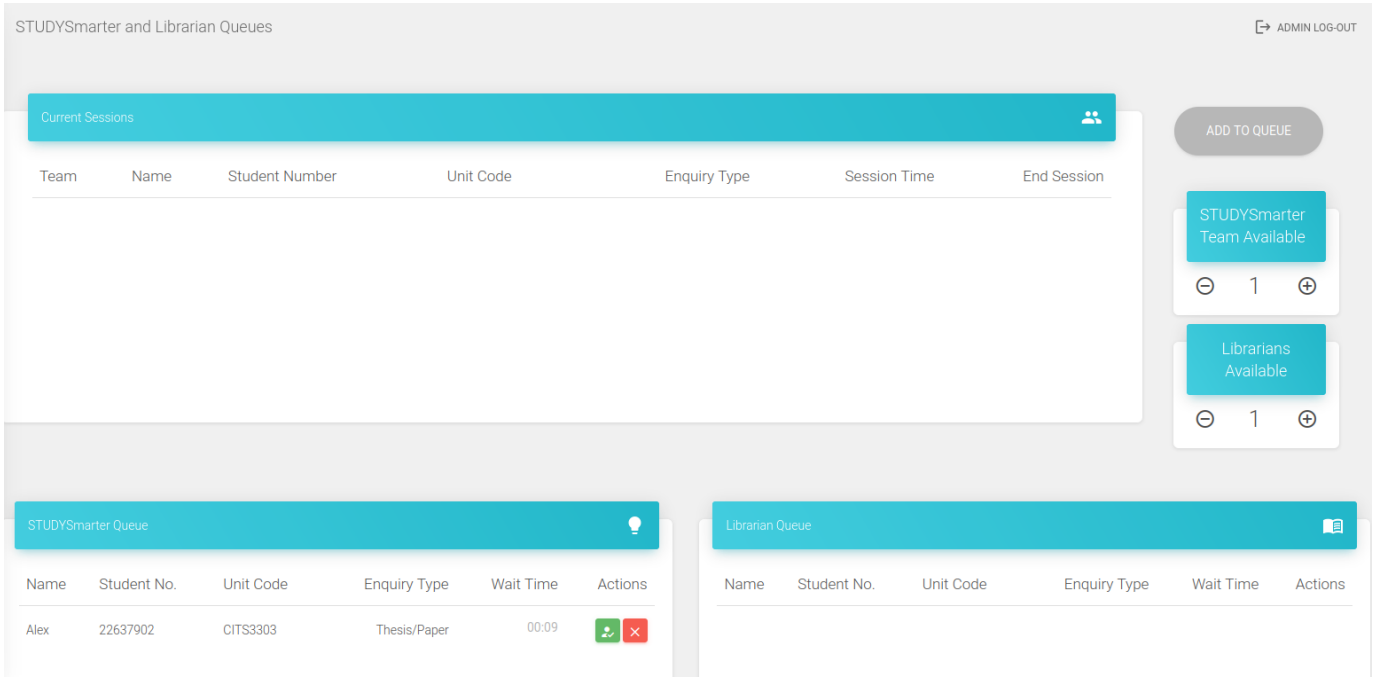
In order to add a student to a waiting queue, look for the "ADD TO QUEUE" button in the top right corner



A form will show up as below. In this form, you can enter student information and choose the queue type between "STUDYSmarter" and "Librarian"

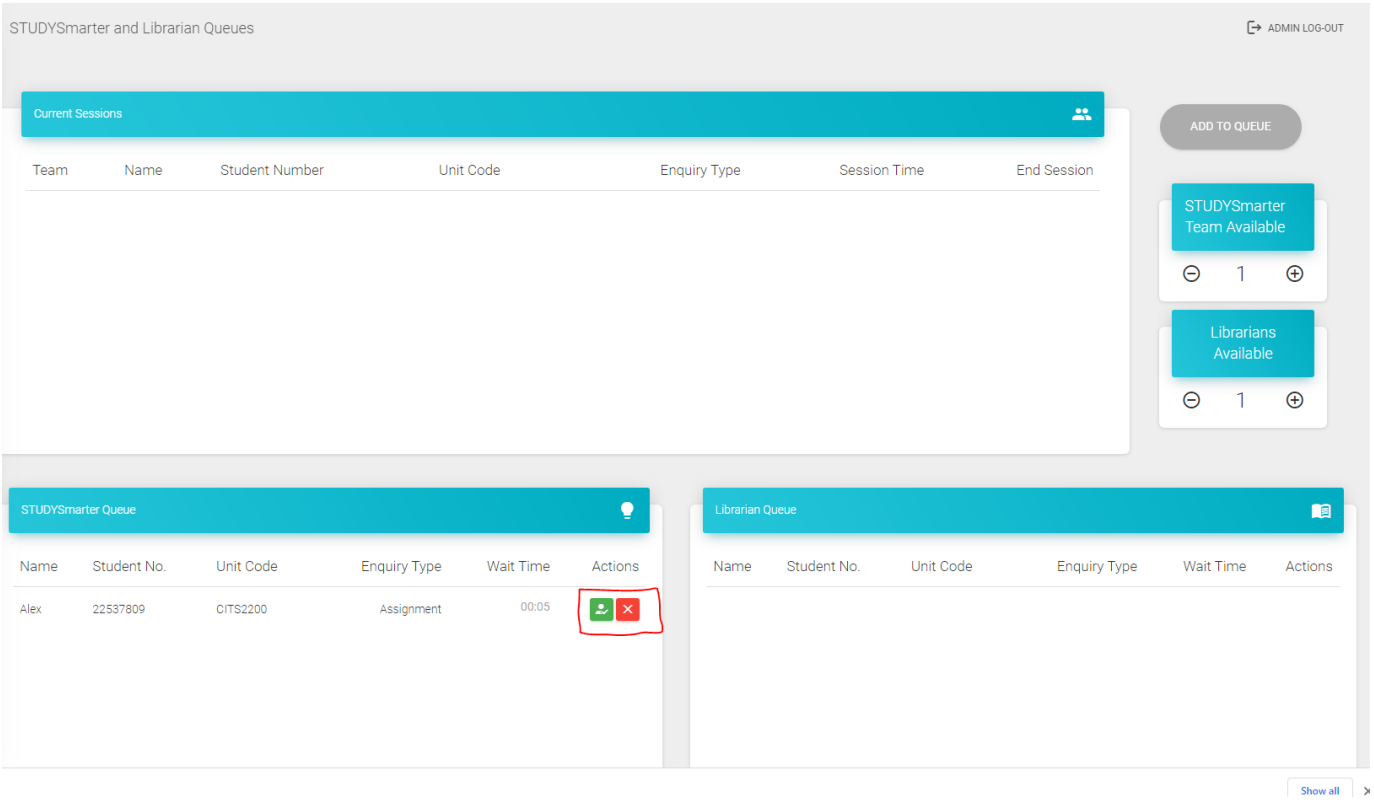


When successfully added, the student information will show up in the corresponding queue. Below is an example



Managing a session

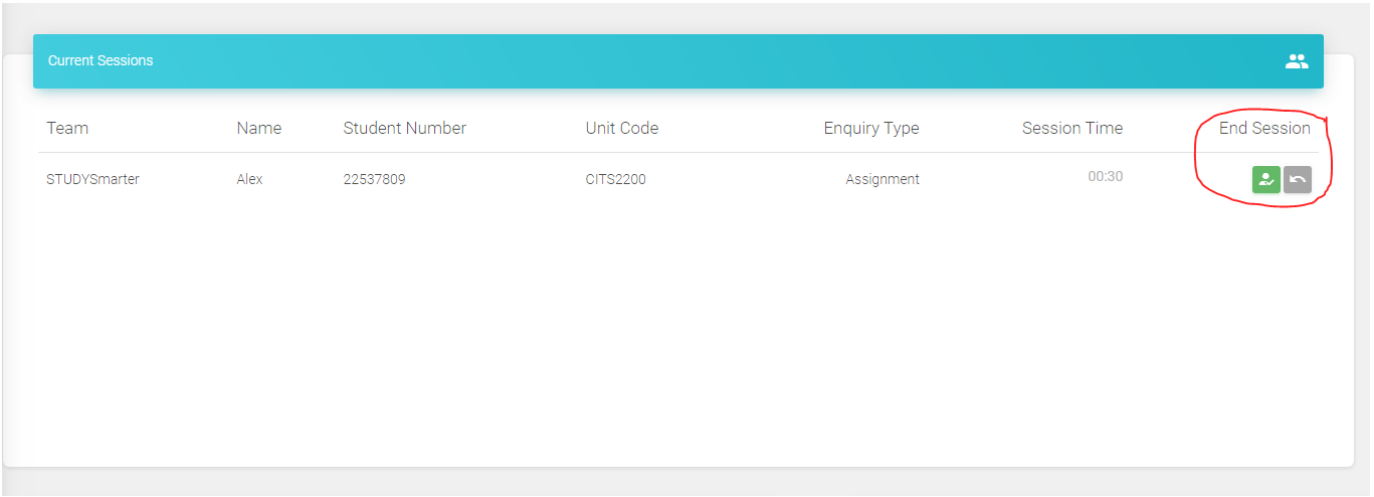
If you want to move students from either waiting queue to "Current Session" table, click the green button in the "Actions" column



If a student cancel the appointment, you can remove her/him from the waiting queue by clicking the red button in the "Actions" column
A message box will show up and ask to confirm your action, click "Yes".

Finish/Undo a session

When a session is done, user can click the green button to finish the session.
When user accidentally added a student to this table from waiting queue, undo it by choosing the undo button

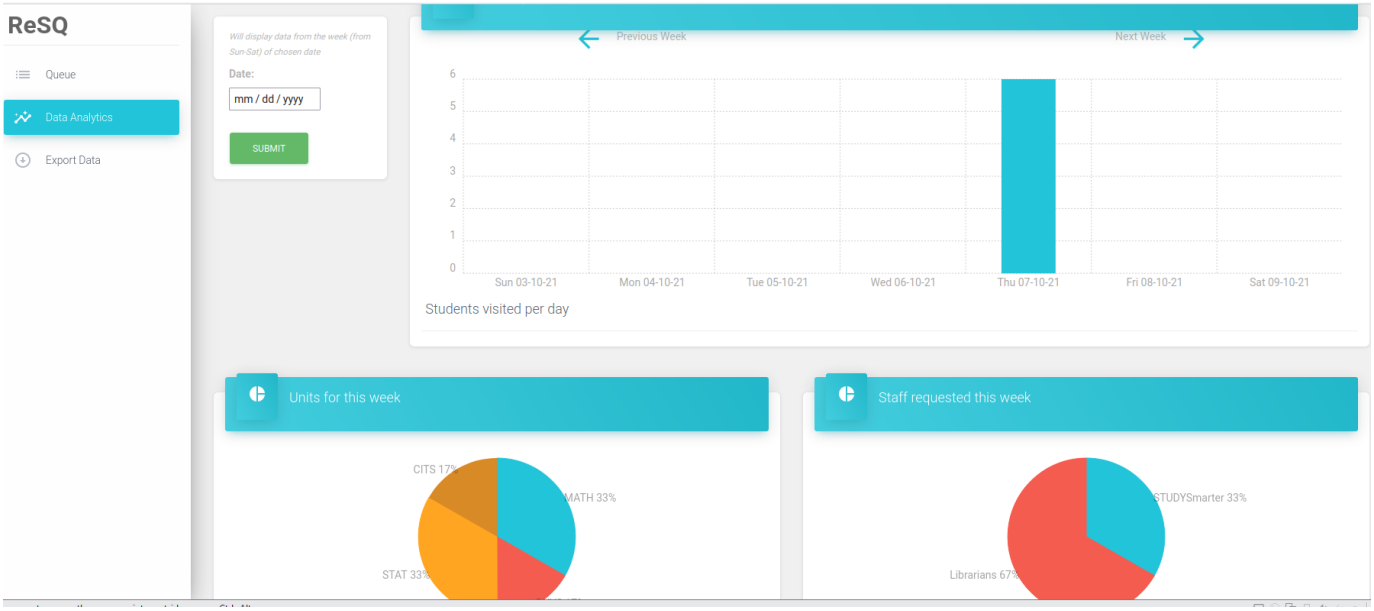


2.1.3 Data analytics

This website provide some basic data analytics for "at a glance" view.

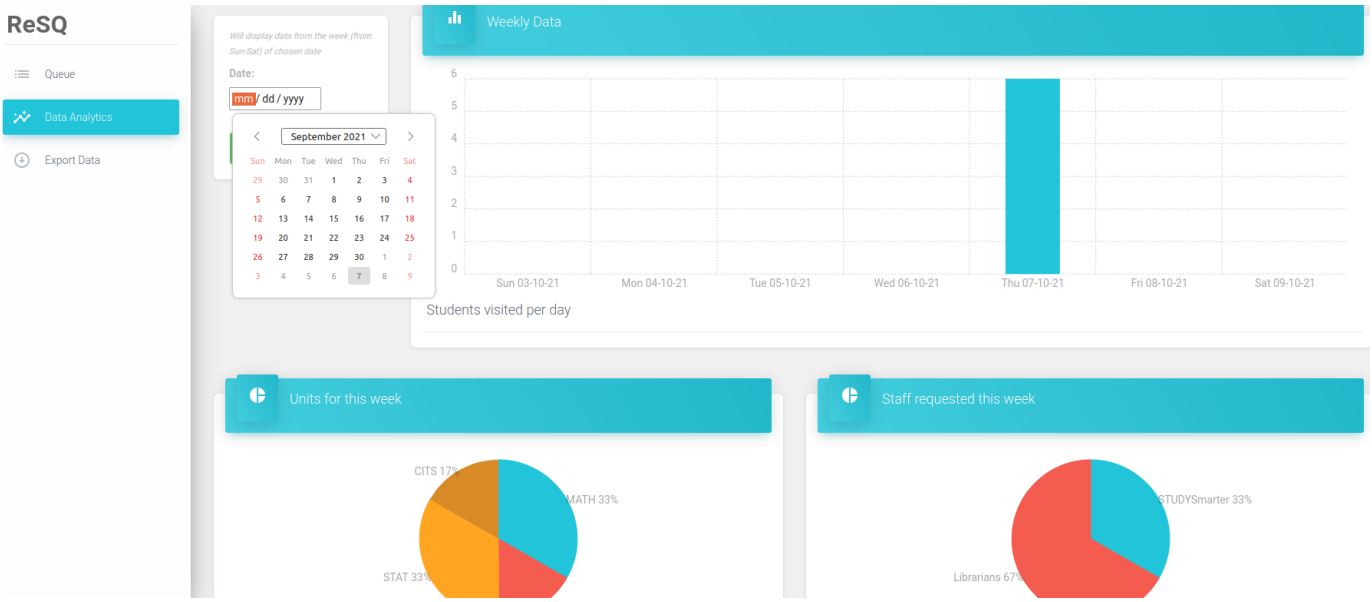
There will be 3 charts generated in the Data analytics tab:

- Students visited per day in the chosen week
- The percentages of popular units in that week
- Comparison between students enter STUDYSmarter and Librarian queue in that week



In order to generate data analytics for a chosen week:

- Choose an arbitrary day in the week you want to generate data

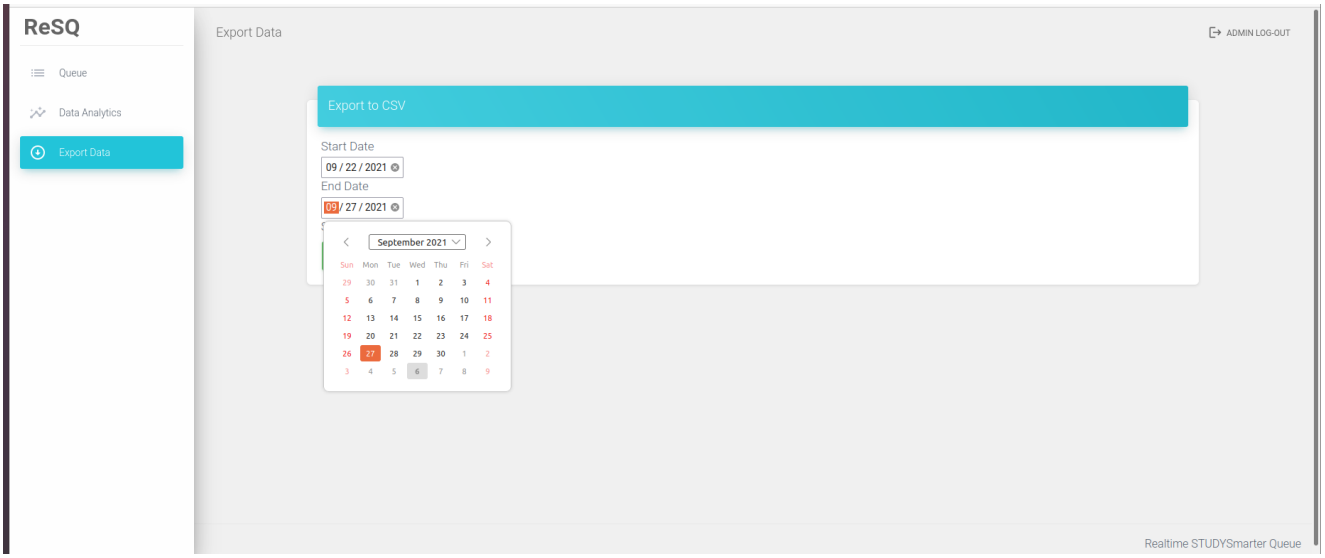


- Click submit

2.1.4 Export data

If user wants to export data to a csv file for further data analysis:

- Step 1: Navigate to the 'Export Data' tab
- Step 2: Choose the 'start' and 'end' date to get the data between those days



- Step 3: Click 'Submit', then download the file

3. Administrator

3.1 Administrator Documentation

4. Developer

4.1 Technical Documentation for Developer

4.1.1 Application

The website is run using a flask server. Flask is a micro framework for the backend of the website. Jinja is used inside the HTML so the display can adapt to server data as well as for running loops. Users and test attempts are saved inside a SQLite database. The username, password, and scores of the user are saved so progress can be encouraged.

4.1.2 Development Workflow

This project uses docker to orchestrate multiple services. Make sure to install docker, see here for documentation.

Once you have it installed do the following:

Environment Variables: Create the `.env` file

There is a file called `template.env`. This contains all the configurations for the entire application for database, flask app, and pgadmin. Copy this to `.env` (you have to create this file).

Run the Docker-Compose

Run the following command

```
docker-compose up
```

This one command will build all the containers. Most notably this will create the Flask Application with pip installation, and database migration to PostgreSQL.

> **Rebuilding containers**
>
> If you do need to build containers, run the following:
>
> ```
> docker-compose up --build
> ```

Services that are running

There are a couple of services that are running

Services	URL
PostgreSQL Database	http://localhost:5432
PgAdmin (PostgreSQL GUI)	http://localhost:8002
ReSQ Flask App	http://localhost:5000
MkDocs Documentation	http://localhost:8001

Going inside the container / Remote Code Execution

Most likely you will be developing inside this container such as doing pip installation and other commands. You can do remote code execution to the container using the following command

```
docker exec -it resq_app bash
```

This will allow you to connect to the container and do whatever command that pleases you.

4.2 Requirements

4.2.1 Acronyms, Abbreviations and Definitions

- UWA: The University of Western Australia

4.2.2 Aim and Scope

4.2.3 Requirements

Stage 1 Functional Requirements

The following are the core functional requirements for the first stage application

USERS

Stage 2 Functional Requirements

"Nice to have"

Some of the "nice to haves" of this project will be covered in this requirements documents. Howevever, "nice to haves" usually will come along as the users of the system see fit. This will be documented in the Issue Tracking Management sytem of the code repository.

Non-functional Requirements

Identifier	Name	Description
NFR1	Security	Only authenticated and authorised users should be able to perform actions such as adding equipment, updating equipment location and information, or searching for specific equipment.
NFR2	Performance	The loading time should not hinder the user experience and productivity of the user in the website. The **page/actions** should have a loading time < 5 seconds on most computing environments on standard internet connections**
NFR3	Maintainable and extensible	The website should be relatively easy to update and extended to accomodate for new contexts.
NFR4	Recoverable	In the event of the web server or database server crashing, all stored data should be fully recoverable.
NFR5	Intuitive user interface	The website should have an intuitive / easy-to-use user interface, so that users will be able to easily use the website and update the equipment database
NFR6	Compatibility	The application should be compatible with recent versions of the major browsers (Safari, Chrome, Firefox and Edge) on laptop and desktop computers
NFR7	Deployability	The application should be compatible with deployment in the SHL VPS

4.2.4 Proposed Solution

The proposed solution is to build a custom web application that will encompass and satisfy the requirements (by completing the suggested "ideal solution" as per the Aim and Scope).

Some research for existing design solution has been done for this project see Appendix: Existing Design Solution. The beauty of custom web application for the team is that it upskills the current software engineers as aligned in the purpose of this unit, and the high possibility of extending application depending on the requirements without being constrained with the bulk of codebase

of other unmaintained opensource projects. Comparatively to enterprise systems, most enterprise systems will charge per users that use the system, this easily becomes expensive because the amount of users that will use the system should be able to accomodate the number of users that are interested in looking for the assets.

Core Technologies

The custom web application will aim to satisfy all the requirements in here along with the "Nice to haves" as they come up. The application will be built using the ...

The authentication system will be outsourced to the UWA Pheme Authentication API to allow any users in with a UWA Pheme account to login.

DOCKER

Docker is a deployment technology that allows virtualization in a server to allow the packaging of software into containers for deployment. To satisfy NFR7 - Deployability, the web platform will use docker to allow the deployment through the SHL VPS Server.

Furthermore, Docker will be used for orchestration of different services in development to increase speed of development, and reduce inconsistency between developers devices (NFR3 - Maintainable and extensible).

CODE QUALITY

The code quality will be ensured by peer reviews between the developers in the team.

CODE STORAGE AND DEVELOPMENT CONTROL

Git source control will be used, using the remote UWA System Health Lab organisational GitHub (NFR3 - Maintenace and Extensibility).

Prototype

See the Prototype mentioned in Figma Interface Prototype

Execution Team

The development of the web platform will be performed by

4.2.5 Development and Methodology

As per the staged requirement, majority of the development will take place on the stage 1 whereas stage 2 are feature-based requirements for the syste,.

4.2.6 Appendix

Existing Design Solutions

Some of the design solutions that have been considered with great detail and justification are here.

4.3 Coding Patterns

4.3.1 Casing

This codebase will be using camel casing.

4.3.2 Linters / Formatters

This will automatically format your code if you install ESLint in VS Code or type `yarn lint` in the specific folders.

Make sure you have installed the devDependencies so additional linters can be used.

4.3.3 Github Issues and Pull Requests

Most changes in the codebase can be matched to a github issue that contains description of the work that needs to be done. Each of the pull request are matched to this github issue with the branch name that has a standard `c{Issue Number}-{branch name}`. The issue number allows referencing especially when resolving reason for change.

4.3.4 Development with Docker

The development is done with Docker to orchestrate multiple services as defined in the `docker-compose.yml` file:

- Documentation at localhost:8001

4.3.5 Inconsistencies

During the project, different developers have different terminology. Some of the inconsistencies are documented below

4.3.6 `queue` means Team Name as well

`queue` refers to where the student belongs to in the queue. This can either be `StudySmarter` or `Librarian` team.

4.4 Frontend/Client-Side App

4.4.1 Frontend

4.5 Backend/Server-side App

4.5.1 Backend

4.6 Automated Testing

4.6.1 Unit Testing

Unit testing is created using Pytest. Refer documentation closely to Pytest-Flask for this project.

> **conftest**
>
> `conftest.py` are a special file for pytest that is automatically loaded by pytest and typically contans fixtures and other setup code.

> **Docker Container Running**
>
> Make sure that you are running the docker container before doing any testing.

How to run unit tests?

Use the docker remote code execution

```
docker exec -it resq_app pytest
```

or if you want to generate the coverage data

```
docker exec -it resq_app coverage run --source="." --rcfile=.coveragerc -m pytest
```

> **Coverage File**
>
> This will produce a file called `.coverage` that contains the records and can be converted to reports.

> **Allure Results**
>
> Whenever you are running this tests, it will produce a folder called `test_results` that will contain results of the test. Refer below for more information about Allure.

HOW TO GET COVERAGE REPORT?

If you've run the tests using the above command and have `.coverage` file, you can generate the reports in multiple ways. More information in here.

```
coverate report -m
```

will print the coverage report in the terminal.

```
coverage html
```

for the html report.

> **What is `.coveragerc` file?**
>
> This file contains the configurations for the coverage testing.

Unit Testing in Pipelines

This was part of #12, but was cut out of scope. Some artefacts of the code can be seen here.

One particular one is the docker `runtime.sh` that can accomodate a `APP_ENV=UNIT_TESTS` to only run unit tests inside the docker.

4.6.2 Allure Report Generator

Allure Testing report is used as a tool to generate test report. More information here https://docs.qameta.io/allure/

This repo has a file called `send_and_generate.py`. It is a simple script that sends a test report to Allure and generates a report. This is currently integrated with the UWA System Health Lab Allure Setup. Documentation can be seen here.

4.7 Continous Integration Pipeline

These are just scripts that run whenever you do pull requests and successful merges. There are a couple of scripts that are currently configured see `.github/workflows`:

4.7.1 Automated Documentation Deployment `docs.yml`

This automatically deploys this documentation whenever `main` is updated with new changes.

4.8 Deployment

The deployment of ReSQ is in the UWA Infrastructure (to be precise in the UWA System Health Lab). The reason being is that permission is granted to Frinze Erin Lapuz (Software Team Lead of the Redbacks Team at the UWA System Health Lab).

There a couple of steps that were involved in doing this:

4.8.1 DNS Configuration

This is a 1 time configuration

The domain name is set to "resq.systemhealthlab.com" in UWA Cloudflare.

4.8.2 NGINX Configuration

Using Binchicken, I have created the NGINX configuration that will handle all requests going to the application (reverse-proxy).

Nginx Configuration

```
server {
    server_name resq.systemhealthlab.com;
    location / {
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_pass http://localhost:10023;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_read_timeout 3m;
        proxy_send_timeout 3m;
    }

    listen [::]:443 ssl;
    listen 443 ssl;
    ssl_certificate /etc/letsencrypt/live/systemhealthlab.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/systemhealthlab.com/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}

server {
    listen 80;
    listen [::]:80;

    server_name
        resq.systemhealthlab.com
        www.resq.systemhealthlab.com;
    return 301 https://resq.systemhealthlab.com$request_uri;
}
```

Diagrammatic Explanation

graph TD User -->|User goes to resq.systemhealthlab.com|Cloudflare Cloudflare-->|Sees that it is under the DNS registed on VPS|VPS subgraph UWA Infrastructure VPS -->|Configuration on Locations|NGINX NGINX -->|Host Port Location| VPS VPS --> ReSQ subgraph Docker ReSQ Other_UWA_SHL_Services end end

4.8.3 Deployment with Docker Image

This requires access towards the application inside the VPS. The easiest way to do this is to have access with the VPS through SSH (you may need permission for this).

Once you are in there, do

```
git pull
```

to pull in the new changes from the `main` branch.

> `git pull`
>
> This assumes that you already have the repository in the VPS. If it does not exist, just do `git clone`.

Then run

```
sh deploy.sh
```

This will rebuild all the containers (for production) as well as the new code.

4.8.4 Gunicorn Process in Production

The reason as to why we gunicorn process instead of `flask run` in production is for the main following reason:

- gunicorn allows parallelising of HTTP request
- automatic disconnect towards the database after the short-lived process (of responding to HTTP request)

More information about its setup here.