User Control System

For Esteem's House Design

Stage 2 – Progress Report

By Hometrics

Team Lakicon
Ram Attra
Vincent Chung
Lee Donovan
Mohammed Yaseen Jamal

Project Manager
Prof. Mike Chantler

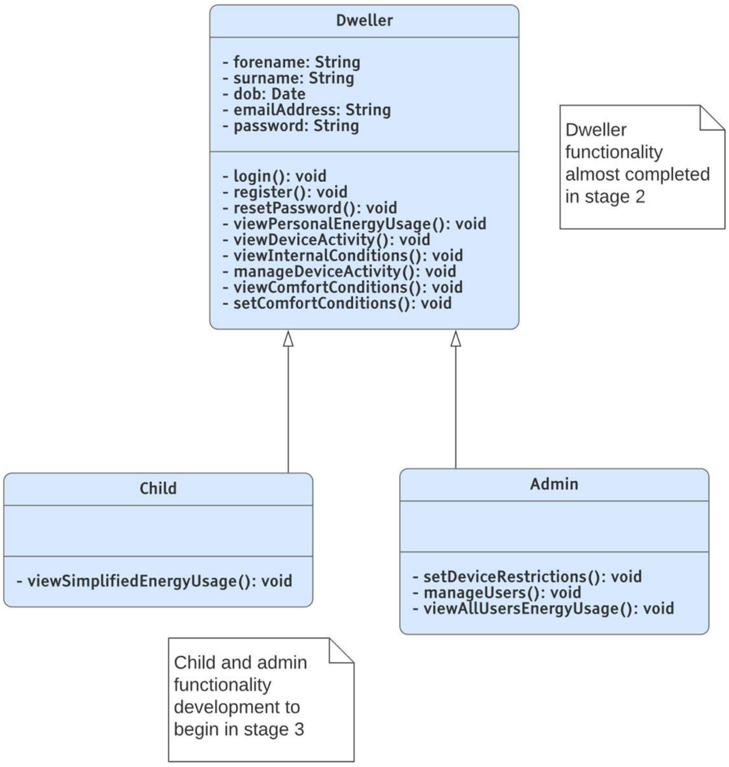
NB: Our group does not contain a member enrolled in BSc Information Systems.

Table of Contents

Progress

Achieved Functionality

The focus of stage 2 was implementing general "home-dweller" functionality, the biggest class among the 3 user types defined in our requirements specification:


Dweller
- forename: String
- surname: String
- dob: Date
- emailAddress: String
- password: String
- login(): void
- register(): void
- resetPassword(): void
- viewPersonalEnergyUsage(): void
- viewDeviceActivity(): void
- viewInternalConditions(): void
- manageDeviceActivity(): void
- viewComfortConditions(): void
- setComfortConditions(): void
Dweller functionality almost completed in stage 2
Child
- viewSimplifiedEnergyUsage(): void
Admin
- setDeviceRestrictions(): void
- manageUsers(): void
- viewAllUsersEnergyUsage(): void
Child and admin functionality development to begin in stage 3

Now at the end of this stage, fulfilment of functional requirements specified in stage 1 is as follows:

Fully complete – 51% *(18 of 35) [GREEN]*
In Progress – 11% *(4 of 35) [YELLOW]*
Untouched - 37% *(13 of 35) [RED]*

A summarised version of planned functionality:

ID	Description
F-SR 1.1	The system must provide a graphical representation of Esteem's current proposed house design.
F-SR 1.2	The simulation shall indicate a change of state to registered devices that can be managed by the user control system.
F-SR 1.3	The simulation shall adapt to the addition/removal of registered devices in real time.
F-SR 1.4	The system shall represent faults in devices that contain them.
F-SR 1.5	The simulation shall represent in-house sensors monitoring internal environmental factors.
F-SR 1.6	The simulation shall represent in-house devices regulating internal environmental factors.
F-SR 1.7	The system shall represent external weather conditions.

F-SR 2.1	The system shall record external weather conditions, using a predetermined dataset sourced from the Solar Decathlon’s website.
F-SR 2.2	When external weather conditions are cool enough, the system shall decrease AC.
F-SR 2.3	When external weather conditions are warm enough, the system shall decrease the thermostat setting.
F-SR 2.4	When external weather conditions are bright enough, the system shall dim light power.
F-SR 2.5	The system shall adjust the humidifier according to external weather conditions.
F-SR 2.6	The system shall adjust the air conditioner according to external weather conditions.
F-UR 1.1	The system must allow home dwellers to sign up to the system.
F-UR 1.2	Once a user has attempted to sign up, the system must send a confirmation email to their associated email before finalising their sign up.
F-UR 1.3	If the user signing up does not have their home already setup on the system, then the system must declare them a home manager.
F-UR 1.4	If the user signing up does have their home already set up on the system, then an authentication request must be sent to existing home managers, and they must accept it for the user to join the home setup.
F-UR 1.5	Once signed up onto the system, users should be able to login to the system at any time with their email address and password.
F-UR 1.6	The system must provide users with the option to reset their password, by sending an invitation link to their email that will allow them to change their password.
F-UR 1.7	The system must allow home managers to set preferred home conditions.
F-UR 2.1	The system shall allow users to switch lights on/off.
F-UR 2.2	The system shall allow users to setup generic devices via smart plugs and turn them on/off.
F-UR 2.3	The system shall allow users to access smart security cameras and view footage.
F-UR 2.4	The system shall allow users to access the thermostat.
F-UR 2.5	The system shall prevent users from accessing devices, by following restrictions set by home managers in the home setup.
F-UR 3.1	All users’ device activity shall be recorded, detailing the device used, time of activity and duration.
F-UR 3.2	The system shall display device activity for all users in the form of a (scrollable) timeline, indicating the time and duration of each device’s on-time.
F-UR 3.3	All users’ energy usage shall be recorded, by totalling the units of energy used across all on-time of devices they have used.
F-UR 3.4	Home managers and home dwellers shall receive energy consumption reports in a statistical manner (line graph), with time on the x-axis and total energy used in the y-axis.
F-UR 3.5	Children shall receive energy usage reports in the form of simple, textual statements.

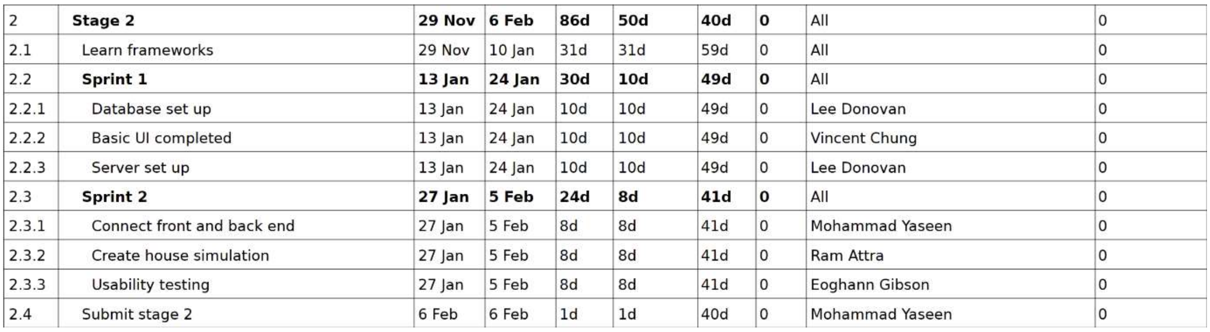
F-UR 3.6	Home managers shall be able to view other users' device activity and energy consumption reports.
F-UR 3.7	All users shall receive a comparison report of their energy usage against the home's total energy generation.
F-UR 3.8	Users can filter their energy usage reports by durations of time (*i.e.* week, month, year).
F-UR 3.9	*Gamification of energy consumptions:* A scoreboard report shall be sent to all users, ranking users by the most energy-conscious home dweller.
F-UR 3.10	Users should be able to delete their own recorded data.

Comparison of Original and Adapted Plan

Following our Gantt chart from the project planning document, we did not specify detailed tasks to be completed, rather we set bigger picture goals for us to achieve instead. Later, the team agreed this was not a long-term viable approach, so we began breaking down the big chunks into smaller objectives complete each week. This ensured our deliverables were easily tracked and assessed in our meetings and the work could be more easily assigned to different members.

At the start of stage 2, group members had begun learning the necessary technologies and development tools to a novice degree. To accelerate this process, regular group coding sessions were organised to reduce downtime spent debugging problems individually - this allowed us to begin development right away as planned.

(Snapshot of Gantt Chart from stage 1)



2	**Stage 2**	**29 Nov**	**6 Feb**	**86d**	**50d**	**40d**	**0**	All	0
2.1	Learn frameworks	29 Nov	10 Jan	31d	31d	59d	0	All	0
2.2	**Sprint 1**	**13 Jan**	**24 Jan**	**30d**	**10d**	**49d**	**0**	All	0
2.2.1	Database set up	13 Jan	24 Jan	10d	10d	49d	0	Lee Donovan	0
2.2.2	Basic UI completed	13 Jan	24 Jan	10d	10d	49d	0	Vincent Chung	0
2.2.3	Server set up	13 Jan	24 Jan	10d	10d	49d	0	Lee Donovan	0
2.3	**Sprint 2**	**27 Jan**	**5 Feb**	**24d**	**8d**	**41d**	**0**	All	0
2.3.1	Connect front and back end	27 Jan	5 Feb	8d	8d	41d	0	Mohammad Yaseen	0
2.3.2	Create house simulation	27 Jan	5 Feb	8d	8d	41d	0	Ram Attra	0
2.3.3	Usability testing	27 Jan	5 Feb	8d	8d	41d	0	Eoghann Gibson	0
2.4	Submit stage 2	6 Feb	6 Feb	1d	1d	40d	0	Mohammad Yaseen	0

We set the objectives of creating and connecting the overall architecture of the system in the first sprint including the database, UI and server - due to assigning time prior to learning frameworks, this was achieved well before the set date. Additionally, we found that we did not need to waste excess resources on this as individual members of the group were able to complete the tasks of the first sprint quickly with minimal issues - thus we were able to assign these resources to begin working on the simulation before the scheduled start date for that task.

Once the objectives of sprint 1 were complete, we were able to connect the frontend and backend of the system without much resistance; this was achieved well before the previously set deadline which gave us more time to refine the basic features of the system and focus our efforts assisting in the development of the house simulation. Due to unexpected changes in the schedule relating to the availability of our project manager; we were required to meet the requirements of Stage 2 by Monday the 3rd of February.

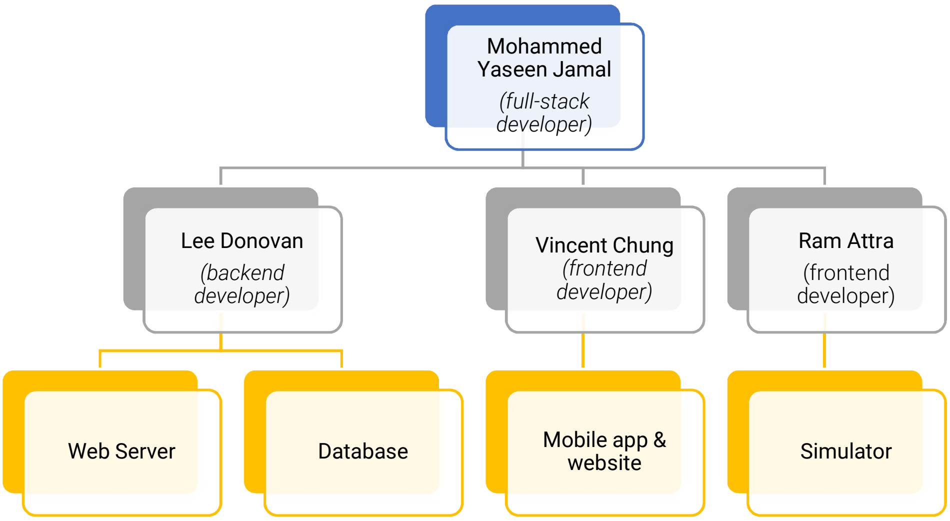
Leading up to this new deadline, we sped up our development to begin introducing more functionality into the system. The basic UI that had been created was consistently being worked on; additional resources were assigned to this task in order to include as much functionality as possible before the app demonstration. Though we had planned on conducting usability testing in this stage; we forfeit this so that we could assign more resources to creating new functionality as we saw this as a bigger objective to meet.

Post-demonstration, we have already begun Stage 3 before the planned start date, as this will allow us more time to deal with any unexpected changes in the scheduling. Despite "*Mobile version testing*" being an objective in stage 3, this was done concurrently with the web version in stage 2 and will continue to be done this way throughout the final phase as the front end developers are comfortable and competent in doing so, further increasing our available slippage time.

Group Organisation

Group meetings were held at least once per week, involving members to showcase their finished objectives, though not switching roles as a completely true Scrum fashion would entail, because the project is not deemed big enough to reap the benefits of doing so. When chosen tasks were not delivered on time by a member, others took it upon themselves to complete the unsatisfied objectives.

As we have progressed into the project thus far; clear roles have been established and members are aware of their own and each other's capabilities, strengths and weaknesses.



NB: Although 6 members were assigned to the team in stage 1, only 4 have been actively contributing.

Design

Brief Product Scope

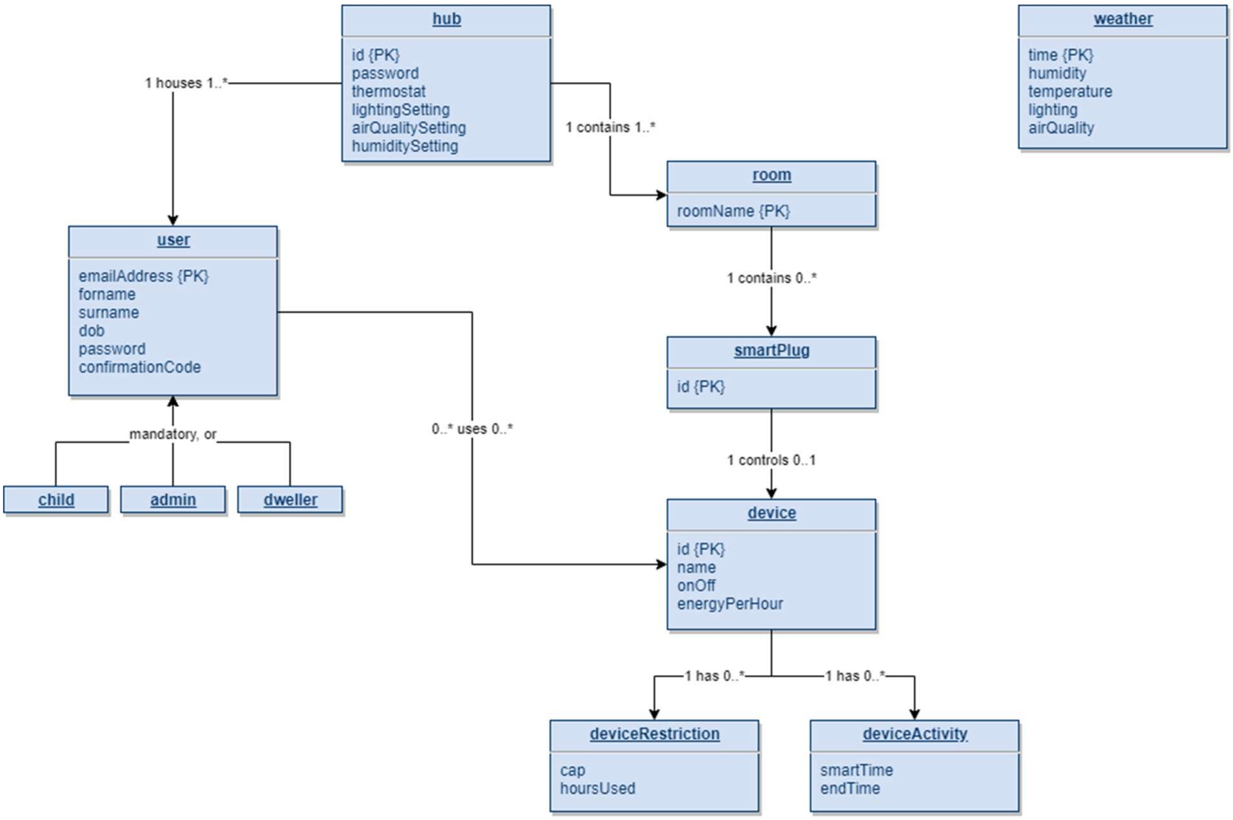
Hometrics is developing a RESTful, cross-platform user control system for Esteem's smart home design, with backend technologies consisting of an Express web server, running on a Node.js environment and managing data with a MySQL database.

Cross-platform usage is an important idea Hometrics wishes to value, thus our frontend is built on the React Native framework; allowing us to develop native platform UIs for Android, iOS, and Web.

High-Level System Overview

Database

The system's backend involves a relational, SQL (MySQL) database to manage data:



Predetermined test datasets have been prepared for demonstration of the system, loaded into the database via a Python script (`load.py`):

Dataset	Description
`comfort.csv`	Pre-set internal house conditions.
`device.csv`	Pre-registered devices.
`deviceActivity.csv`	Pre-determined user activity.
`deviceRestriction.csv`	Pre-set device restrictions.

`hub.csv`	All house hubs registered onto the system.
`room.csv`	Names of all rooms in the house.
`smartPlug.csv`	Smart plugs installed throughout the home.
`user.csv`	Pre-registered users on the system.
`weather.csv`	External weather conditions, declaring the humidity, temperature, solar radiation and air quality every minute throughout a 24-hour period.

Server

An Express web server, running on a Node.js environment, acts as our RESTful API. Routes (sections of Express code) have been categorised into separate JavaScript files:

main.js

Route	Type	Description
`/ping`	`GET`	Returns the string "Ping!" to denote the server is reachable.

device.js

Route	Type	Description
`/activity`	`POST`	Returns a list of JSON objects detailing the recent device activity of a user.
`/totalUserEnergy`	`POST`	Returns the total energy consumed by a user within a week, month and year.
`/userEnergyBreakdown`	`POST`	Returns the energy consumed by a user, spread out across a week, month and year.
`/scoreboard`	`GET`	Returns a list of JSON objects detailing the energy consumption of each user in a household, sorted by most conservative in consumption.
`/totalHomeEnergy`	`GET`	Returns the total energy consumption of the entire household.

deviceManagement.js

Route	Type	Description
`/room`	`GET`	Returns a list of JSON objects detailing the number of devices associated with each room in the household.
`/roomDevices`	`POST`	Returns a list of JSON objects detailing the devices in a selected room.
`/toggle`	`POST`	Toggles a selected device on/off.

homeSetup.js

Route	Type	Description
`/comfort/thermostat`	`POST`	Change the thermostat setting to any value between 0 and 30 degrees celcius.

user.js

Route	Type	Description
`/signup`	`POST`	Allows a user to register onto the system. Also includes form validation using regular expressions and sends confirmation code to user's email address.
`/login`	`POST`	Allows registered users to login.
`/authenticate`	`POST`	Checks whether a user's credentials are valid.
`/confirmationCode`	`POST`	Checks a confirmation code. If valid, then the user's type is calculated via their age and can now login to the system.

weather.js

Route	Type	Description
`/`	`GET`	Returns current external weather conditions according to the server's current time.

database.js

Export	Description
`connection`	A synchronised MySQL connection object used by routes in the server to query the database.

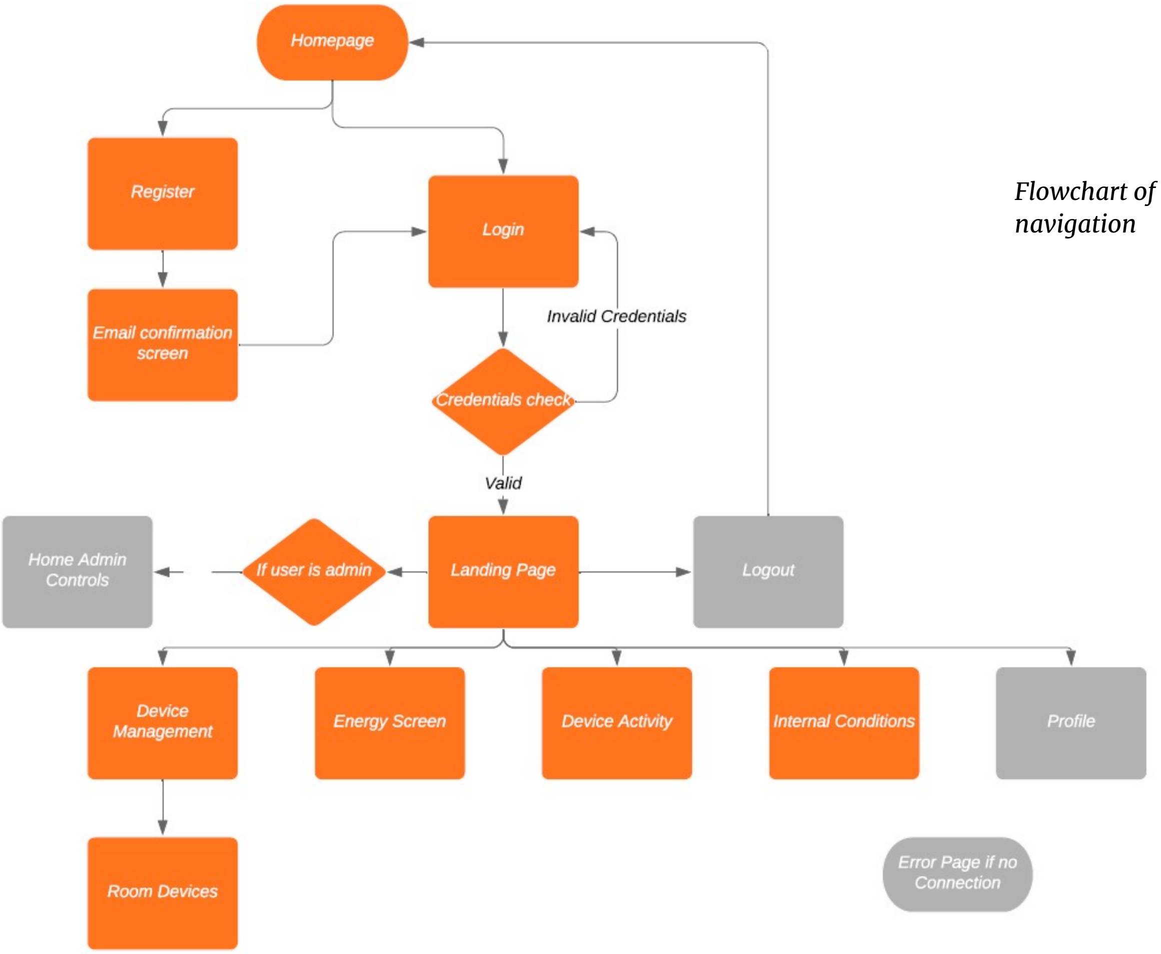
Server Dependencies

Package	Justification for Use
`body-parser`	Extract JSON body from incoming frontend requests.
`date-and-time`	Eases handling date objects in JavaScript.
`mysql`	Node.js connector for MySQL database.
`nodemailer`	Allows the system to send emails to users. *(hometrics.donotreply@gmail.com)*
`sync-mysql`	Offers synchronised functionality of the `mysql` connector – handling Promise objects in JavaScript can get very messy, and since querying with MySQL in Node.js is an asynchronous function, using a synchronized version is much simpler for current needs.

Frontend

Our system provides a cross-platform UI, compiling to native functionality for Android, iOS and Web through React Native *(including a subset of React.js components)*. Screens/pages of the frontend are built as JavaScript subclasses of React Native's `Component` Class:

Screens	Description
`ConfirmationScreen.js`	Prompts user to enter confirmation code emailed to them after signing up.
`DeviceActivityScreen.js`	Displays recent device activity as a timeline.
`DeviceManagementScreen.js`	UI for toggling devices on/off, adding/removing devices.
`EnergyScreen.js`	Displays charts for energy usage.
`HomeScreen.js`	Home screen of the app.
`InternalScreen.js`	Displays set internal house conditions, shall also include options to change it.
`LandingScreen.js`	User arrives here after a successful login. Currently displays cards leading to view energy usage, internal house conditions, device management and activity.
`LoginScreen.js`	Allows existing users to login.
`RegisterScreen.js`	Allows users to sign up to the app.
`RoomDevicesScreen.js`	Displays cards detailing the rooms of the house and the number of devices plugged in each room. Clicking on a room leads to the `DeviceManagementScreen` for that room.
`SimulationScreen.js`	Shows the simulation – for now, this is a part of the user's app, though we plan to separate it as its own application after implementing its full functionality.



Flowchart of UI screen navigation

Dependencies

Package	Justification for Use
`expo`	Provides interface and services that do not require us to manually link additional libraries to Android Studio *(Android)* and XCode *(iOS)*.
`react`	Web UI requires React.js components.
`react-native-elements`	Provides more customisable React Native components for `Card` and `Button`. *E.g. React Native's default Button component does not allow us to change its dimensions.*
`react-native-gesture-handler`	Makes it easier to deal with touchable gestures in React Native; a must have for the Android and iOS platforms.
`react-native-modal`	Required for showing content in the simulator when clicking on a room to present.
`react-native-pure-chart`	Required for displaying energy usage charts *(line, bar and pie)*.
`react-native-reanimated`	Provides animation features used in displaying energy usage charts.
`react-navigation`	Routing and navigations for changing screens/pages in frontend.
`react-native-timeline-flatlist`	Used to display recent device activity in a timeline format, using the new `flatlist` component.

Issues

Resolved

Due to the new firewall put in place for machines accessing the MACS development server outside of the university's network, Hometrics is unable to fulfil NF-SR 1:

ID	Description	MoSCoW
NF-SR 1	The system's backend shall be hosted on the MACS server.	Must

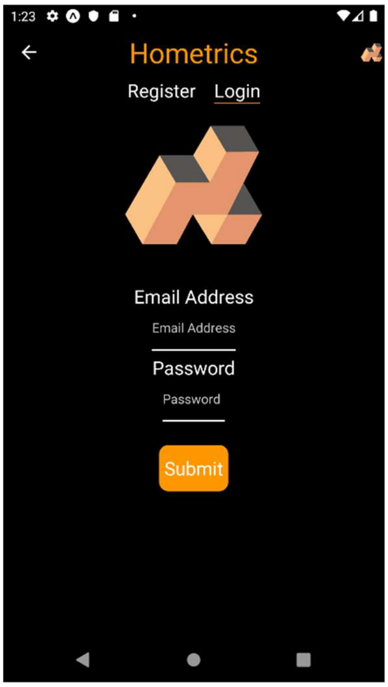
As a resolution for stage 2, we used the localhost on our own development machines to demonstrate web server and database functionality. `Ngrok`, a reverse proxy software, was used to connect mobile devices demonstrating frontend capability to the backend hosted on our localhosts. However, this is only a temporary fix, as `Ngrok` limits us to 20 HTTP requests per minute. This would seriously limit later development in stage 3, and so we plan to migrate our backend to Heroku; a cloud application platform.

Persisting

Due to React Native being a rapidly evolving framework, many of its components used in our frontend have been deprecated *(which was not the case during the time we started development)*. The obvious solution is to replace used deprecated functionality with its new successors, though we are expecting further deprecation of more components later in stage 3, as new releases of React Native are being published every month.

Furthermore, all contributing members of the team dislike the current UI colour scheme, taken from mock designs in our usability deliverable during stage 1. Thus, stage 3 shall also include a change to this too, introducing a greener, lighter colour scheme.

e.g. Snapshot of current login screen (Android UI)



More Functionality

In addition to the requirements set in stage 1, Hometrics has deemed it fit to fulfil extra functionality, relating to our users' data rights and protection:

ID	Description	MoSCoW
F-UR 3.11	The system shall allow users to download a copy of their saved data, sent to their email addresses.	Must
F-UR 4	The system shall encrypt sensitive user data on the database, using the Node.js `crypto` module. Sensitive data includes: • Date of birth • Hub credentials • Email address • Forename • Surname • Password	Must

Testing

Backend

We were also able to monitor the expected behaviour of the API by calling HTTP requests to the server with `Postman`, a collaboration platform for API development. However, this is not a long-term reliable testing methodology that would ensure a strong, reliable build of the final product. Thus, we plan to incorporate unit & integration testing in stage 3, using the Mocha.js testing framework (as originally planned).

Frontend

Android

The Android version of the mobile app was tested on several Android emulators, acting as different device models *(Google Pixel Phone, Huawei Tablet, …)*, as well as real Android devices belonging to Hometrics' developers.

Web

The web app was demonstrated on several of Hometrics' developer's desktop machines and seems to be working just fine.

iOS

Testing the iOS app was very limited, as none of the Hometrics' developers own an iPhone/iPad and using an iOS emulator requires an installation of the XCode IDE, which is only available to macOS users, which, again none of us possess. Thus, we had to make do with limited testing intervals on apple devices belonging to friends.

Conclusion

Adapted Plan for Stage 3

To conclude, we have devised a new adaptive plan to increase efficiency; we will continue to have each of our contributing members work as leads on their respective part of the system and each week, each member will accept and deliver on tasks pertaining to their expertise so that the overall system functionality will be reliably improved upon in each subsequent week, rather than focusing all of the team's efforts on one section at a time.

Customer's Expectations

Our groups progress has so far met all the customer's expectations, despite the numbers of contributors falling each stage. We started off with 6 members and currently only have 4 contributing members in Stage 2. Regardless, we have managed to complete the overall architecture of the system and have completed over 50% of the app's functionality for both web and for native mobile devices (iOS and Android). We have delivered the expected functionality within the expected timeframe, so overall the group is progressing at an acceptable rate.

The foundations for the system have been paved, we have successfully created the building blocks to be reused later and thus the remaining work left to complete will be developed at an accelerated rate. With the bare minimum of each member completing one requirement per week, we will be on track for a finished system within ~4 weeks (March 5th) - leaving us a large chunk of time to test and refine the product before submission.

References

https://www.postman.com/ – *Postman*

https://mochajs.org/ – *Mocha.js*

https://facebook.github.io/react-native/ – *React Native*

https://developer.android.com/studio – *Android Studio*

https://dev.mysql.com/downloads/mysql/ – *MySQL Community Server*

https://www.npmjs.com/package/sync-mysql – *sync-mysql*

https://www.npmjs.com/package/body-parser – *body-parser*

https://www.npmjs.com/package/date-and-time – *date-and-time*

https://www.npmjs.com/package/mysql – *Node.js driver for MySQL*

https://www.npmjs.com/package/nodemailer – *nodemailer*

https://www.npmjs.com/package/sync-mysql – *sync-mysql*

https://expo.io/ – *expo*

https://www.npmjs.com/package/react – *react.js*

https://www.npmjs.com/package/react-native-elements – *react-native-elements*

https://www.npmjs.com/package/react-native-gesture-handler – *Mobile gestures*

https://www.npmjs.com/package/react-native-modal – *Modal component*

https://www.npmjs.com/package/react-native-pure-chart – *Chart component*

https://www.npmjs.com/package/react-native-reanimated

https://www.npmjs.com/package/react-navigation – *Navigation component*

https://www.npmjs.com/package/react-native-timeline-flatlist – *Timeline component*