Thermostadt App

Building Useable Software: Group Project

Group 49

Joshua Banga
jlb997@student.bham.ac.uk
2050997

Damion Brecknell
dtb824@student.bham.ac.uk
2123224

Matthew Cummerson
mxc555@student.bham.ac.uk
2330297

Nathaniel Toma Drăgușin
tnd096@student.bham.ac.uk
2150296

Subhaan Hussain
sxh1311@student.bham.ac.uk
2185865

Ho Jim
hlj097@student.bham.ac.uk
2179097

William Kerry
wxk100@student.bham.ac.uk
1972700

Anisa Khan
axk880@student.bham.ac.uk
1821854

William Pearson
wjp997@student.bham.ac.uk
2052597

December 2021

Contents

A Requirements Engineering

A1 Proposed system

i Specification

The app (Thermostadt) is an energy-management mobile application developed for the University of Birmingham that allows users to input heating and lighting preferences while allowing administrators to receive and utilise user data. Thermostadt employs data from digital sensors, to monitor heating/cooling, lighting, ventilation, noise levels, and occupancy. It will give students restricted control over heating and lighting in parts of the campus by aggregating user preferences in campus spaces, modifying the conditions in that room or building to match the average preference. This presents students the opportunity to have an interactive role in the university's energy management. Thermostadt's aim is to increase its users' energy consumption awareness to promote more energy-efficient behaviours.

Thermostadt will display campus sensory data, allowing users to directly affect the energy consumption of campus spaces and award points for environmentally-friendly preferences. Points will be awarded on the basis of how a user's preferences have contributed to decreases in energy usage. Thermostadt's users are students, visitors, and facilities staff. Users will log in using their university accounts or a one-time QR code for non-university members.

The energy consumption of each individual student is estimated based on their preferences, and students are ranked in leaderboards to see how they compare with others. These competitions can run both weekly and termly. This provides students with an incentive to be more conscious of their energy consumption.

Students will receive termly reports of their energy consumption behaviour. The report will include a breakdown of the points, their estimated energy usage, recommendations on how to improve their score, and a comparison of their results with the mean average user. Students can use this information to adapt their behaviour to any energy lifestyle goals they may have.

For normal users, Thermostadt provides a map of the campus that displays: occupancy, noise levels, ambient temperatures, lighting levels, and typical energy efficiency, based on sensor data. This allows students to choose a study space based on their preferences. The map will be updated hourly so users have access to accurate information.

The primary feature for administrative users (such as building managers) is the ability to see the energy usage of buildings and rooms, and students' preferences. Thermostadt will also provide graphs showing predictive, previous, and current energy usage. This will help the university to determine their students' energy demands. This data could be significant when the university is deciding how to allocate resources to reduce its energy wastage as it will illuminate the buildings with high energy usage and demand. However, to prevent abuse, administrative users will have override permissions, and student preferences will be limited to a practical range between 10 to 30°C.

The scope of the design makes certain assumptions. Firstly, it assumes the university possesses a unified HVAC control system that can be controlled with external inputs and can independently control the heating and cooling in different rooms and buildings. This will allow user preferences to affect the heating levels. We assume Thermostadt will have access to the device operating system's location services (henceforth, 'GPS'). This will allow Thermostadt to effectively estimate building/room capacity and its users average energy usage. Also, the system assumes administrative users will be authorised and validated by the university through the university email.

Thermostadt is suitable only to users that are on the university campus given that user preferences are only taken into account when in close proximity to the buildings and rooms. Also, it is assumed that users must have an account to set preferences and obtain rewards. Furthermore, it's assumed accounts will be split into two types, a normal user and an administrative user.

ii Assumptions

- The university possesses a unified HVAC control system that may be controlled with external inputs, and that can independently control the heating and cooling in different rooms.
- The service will have access to UoB AD-FS SSO, course lists and timetabling information.
- The app will have access to GPS.
- The university will be able to validate administrative users.
- The system can use the university email to verify normal users.
- The system is only functional if the users are on campus as preferences only have an influence if the user is in close proximity (measured by GPS) or their location was recorded through their university keycard usage to a building for 10 minutes.
- Users can change their preferences at any time.
- The use of smart light bulbs that are able to be remotely controlled using the aggregated preferences of users.
- The app is suitable only for mobile devices.
- The app will make use of the existing UoB digital map while adding key features.

A2 Requirements

i Functional requirements

User accounts

a. Must require the user to log in

b. Must allow the user to sign using their university account

c. Must allow users to sign out if they are already signed in

d. Should allow users to sign in with a QR code
e. Should sign out users who signed in with a QR code when the app closes
f. Should classify user account type when signing in

Energy preferences

a. Must allow user to input their preferred temperature
b. Must allow user to input their preferred lighting intensity
c. Must allow user able to input their preferred lighting warmth
d. Must control rooms HVAC system based upon preferences of users in the room
e. Must estimate each user's energy consumption
f. Should allow staff users to control preference boundaries

Leaderboards

a. Must calculate a score based upon user's energy consumption
b. Must rank students in order of their score
c. Should have leader boards that run termly
d. Could have a weekly leaderboard
e. Could rewards users for studying in groups by increasing their score

Campus maps

a. Must show the map of the campus
b. Must show the availability of each room.
c. Must show the HVAC setting of the room that the user is in.
d. Should colour coordinate building for staff based upon energy consumption
e. Should show a floorplan of each building.
f. Should show the HVAC setting of rooms the user is not in.
g. Could colour coordinate study spaces based upon how much of a match it is for the users' preferences

Energy reports

a. Must produce a termly report for each user
b. Must include an estimate of the amount of energy consumed
c. Should include an estimate of the amount of carbon emissions caused
d. Should calculate a report based upon the average users
e. Should allow the user to compare their report with the average report
f. Should include a heat map of where they have spent most time on campus
g. Could keep termly for users to compare with their history

Model of usage

a. Must estimate how much energy a building uses
b. Must monitor how much energy is being used
c. Must provide a graph to compare the estimated usage with the actual usage
d. Could predict students that are going to rank top of the leaderboard
e. Override settings
f. Sometimes students are devious and may want to exploit the application with malicious intent. Or maybe a conference is taking place in one of these rooms. It is important that our application is flexible and allows the energy preferences to be overridden.
g. Must allow staff to control room states
h. Must permit existing conventional HVAC to continue operation in the event of a catastrophic app failure

ii Non-Functional Requirements

Security

a. Must ensure that all data gathered from sensors, and other means are stored securely to prevent access from unauthorised parties.
b. Should host servers locally
c. Could host servers on campus

d. Should protect transfer of data between the user app, servers, and third parties (i.e the university login SSO). Encryption should be used when transferring sensitive information such as emails and passwords.

e. Must only receive username, account type, and sign in status when signing in using the university login system. Unnecessary data such as addresses, courses should not be accessed.

f. Must anonymise all data stored

Reliability

a. Must carry out scheduled updates every hour

b. Must update user maps

c. Must update room availability

d. Must update the Leaderboards

e. Should only permit temperature inputs in the range 10-30°C unless otherwise specified by staff.

f. Should re-authenticate users at the start of each academic semester

g. Should be able to tell what room the user is in 80% of the time

Scalability

a. The system must be able to handle 20,000 concurrent users

b. Must show feedback within a second when a user changes their preferences

c. Must update HVAC system within 5 minutes when the user changes their preferences

d. Must not update the server if there are no changes in preferences

e. Must allow user to update their preferences within 10 seconds

f. Must allow user to find their rank in the leaderboards within 10 seconds

Accessibility

a. Should have a colourblind modes suitable for users with protanomaly, deutranomaly, or tritanomaly

b. Could have a dark mode

c. Should allow users to zoom into the map at least 2×

d. Should have push notifications

B Software design with UML

B1 Use case diagram

See figure B1a.

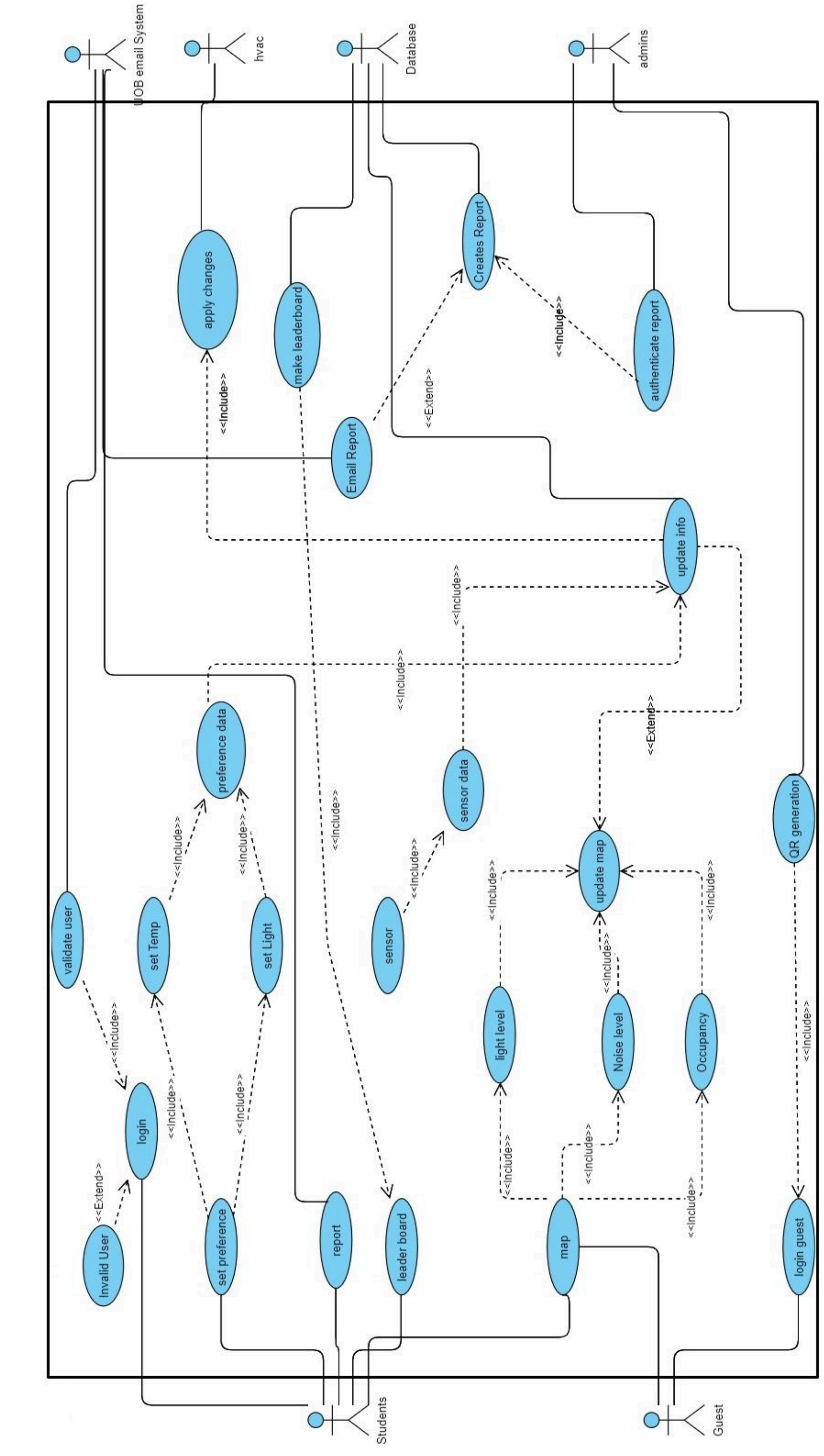


Figure B1a: Use-case diagram

B2 Use cases

Use cases are supplied for the setting of temperature preferences and delivery of reports.

	Set Student Temperature
Description	This use case describes how Students will set their temperature preference on the app.
Actors	1. Students 2. UOB Email System
Preconditions	Students already have the app installed on their phone
Flow of events	1. The use case begins when students launch the app and enter main menu 2. The user uses their UoB username and password to log into system 3. Use case: Validate user is performed 4. The app displays the different tasks users can do 5. User selects on set preferences and then set temperature 6. Student enters temperature 7. user and their preference sent to database 8. use case ends successfully
Alternative flows	1. Invalid email or password 2. In step 3 of the use case specification: 3. Student Set temperature. 4. App does not allow user to proceed 5. Display invalid username or password message
Post-conditions	1. A message appears on the app confirming preference change 2. Within the message box a link should be provided for steps on how to find study spaces that match Student's preference. See use case: Find preference Temperature

Table B2a: Use-case: setting a temperature preference

	Email Student Report
Description	The student should receive an email of his of their contribution of reducing the energy consumption of the university
Actors	1. Student 2. Admin 3. UoB Email System
Preconditions	1. Students already have the app installed on their phone. 2. The user has already set up their preferences. 3. The user has GPS on their phone so the app can determine contribution.
Flow of events	1. The use case begins when the term comes to an end 2. The database uses the create report case, to create individual personalised reports. See use case: create report for more details 3. Use case: validate reports is performed 4. The UoB email system then emails each report by using the username and sensory data provided by create report 5. The students receive an email showing their contribution to reducing emissions on campus 6. The use case ends successfully
Alternative flows	1. Validate report fails 2. In step 3 of use case specification: Email Student Reports 3. Reports are not emailed to students
Post-conditions	The students receive an email of their termly report in PDF form.

Table B2b: Use-case: emailing a student report

B3 User stories

Scenario one

A first-year student, James, is going to attend his first lecture today at 14:00. To get a better understanding of where he can study in his breaks between lessons, James decides to go to the campus a few hours yearly. He intends to walk around the campus for a while and then find a quiet study space so he can do some basic revision of the material content before his lecture.

While walking around one of the study spaces, he sees a leaflet on a door which claims that the Thermostadt app will find a study space at the university which is best suited to a student's preference. As it is a cold winter's day, James decides he wants to find a warm place to study and opens his smartphone's app store and downloads the app.

When James launches the app he is presented with a sign-in page. The sign-in page tells James to use his University Email and password (which he has already setup) to gain access to the functionality of the app. Once signed in he selects the "set preference" option and sets his temperature preference to 25°C. The app then confirms his preference by showing a message box confirming his preference has been changed.

Scenario two

The first term at the University of Birmingham has come to an end and the Thermostadt system administrator, named Charles, needs to validate the output from the reporting system. In order for Charles to validate the reporting system he needs to select a minimum of five students at random and create their reports manually.

To select the five random students Charles decides to use a random number generator and selects the corresponding students with the same index number as the generated number on the database. Once he has his sample group of students, Charles begins creating their corresponding reports on a spreadsheet software (like Microsoft Excel).

The formulas and student's preferences needed for this report can be found on the Thermostadt database. Charles uses the preference data and formulas to calculate each student's contribution in the reduction of CO_2 emission. He then checks his calculation with the automated system's calculation to see if they are the same.

The calculations in the reports Charles has created matches the automated report system calculations, so he now has to confirm this to the system. Charles can confirm the reports by selecting the 'confirm reports valid' button on the app (while signed into the admin account). Once he does this the system then uses the preexisting university emailing system to send the reports to the students.

B4 Activity diagram

See figure B4a.

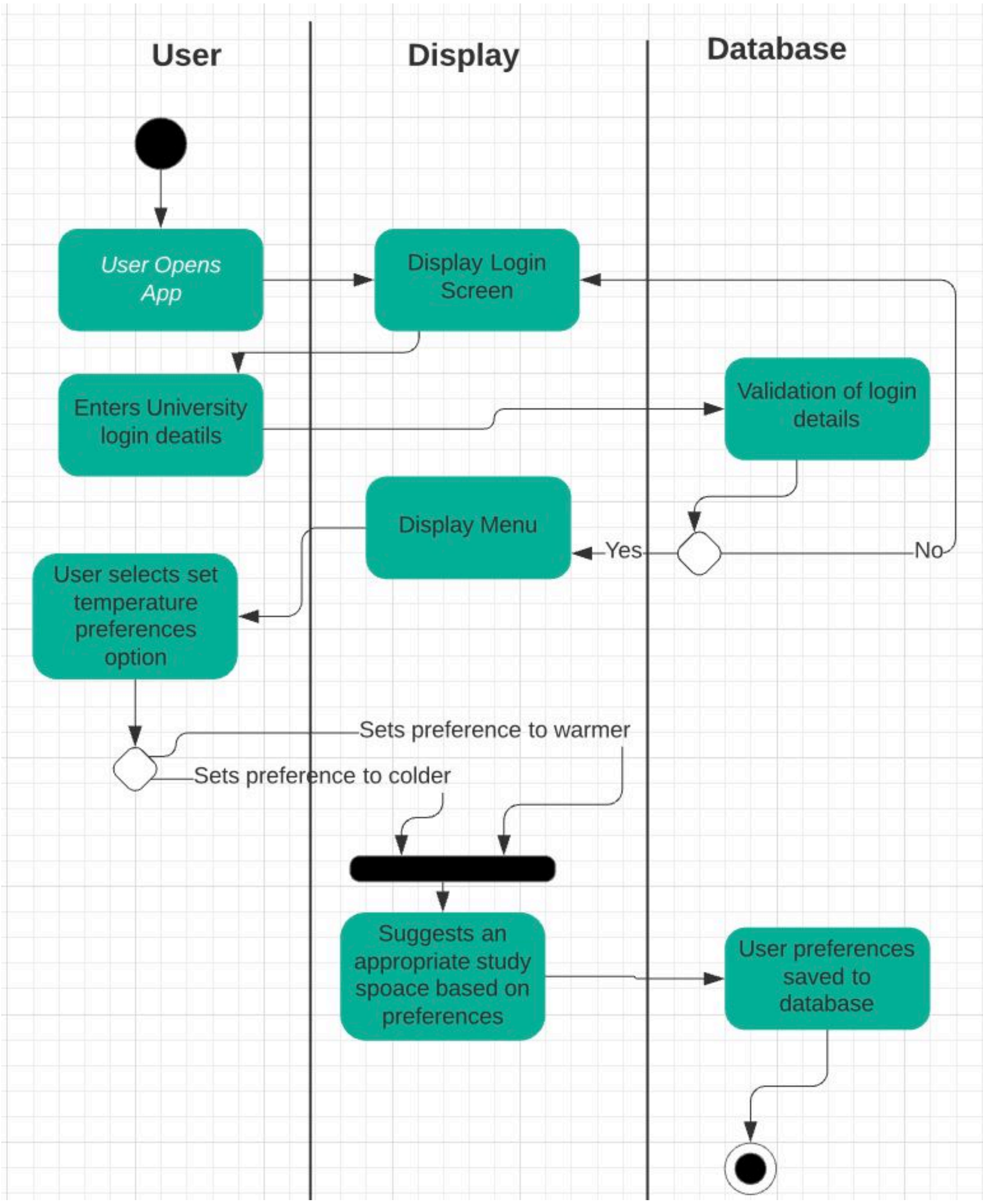


Figure B4a: Activity diagram

B5 Class analysis

i Noun-verb analysis

Term		Accepted	Reason
application		No	'Application' here refers to the entire product. It is not useful to map the term onto a component.
developed		No	
UoB		No	Refers to context for application usage.
users	User	Yes	The application will have users, which will be a rational class.
administrators	Admins	Yes	Administrators will be a subclass of the User class.
digital sensors	Sensor	Yes	Application will handle inputs from sensors, whose locations on campus must be individually recorded for mapping/modelling purposes.
employs		No	Term is employed in description of project's intended effects. Not relevant to class analysis.
heating/cooling		No	A parameter that will be defined for various objects in the system.
lighting		No	A parameter that will be defined for various objects in the system.
ventilation		No	A parameter that will be defined for various objects in the system.
occupancy		No	A parameter for Room objects.
students	Students	Yes	A subclass of the User class.
control	tbc	Yes	A method that outputs processed user inputs to third-party systems.
campus		No	Analogous to 'all rooms'.
aggregating	tbc	Yes	A method that processes user-inputted preferences and sensor inputs, for example: performs checks to prevent abuse that aggregates multiple numerical values into a single numerical value according to a specified weighting.
user preferences	Preference	Yes	Class for storing user preference settings.
room or building	Room	Yes	Class for rooms, the spatial unit used in the application's map and condition-control functionality.
average preference		No	Duplicated functionality contained in previous classes and methods.
energy-management		No	Term is used to elaborate on implications of application's use, but does not refer to its explicit functionality.
display	displayMap	Yes	A method for displaying the data maps detailed in the application description.

Term		Acc.	Reason
energy consumption	estimateUsage	Yes	Method to estimate a user's energy consumption for a given time period, based on that user's preferences.
award		No	Term is used to describe business practice that would necessarily occur outside of software.
points	generatePoints	Yes	Function that calculates a points value based on some of a user object's parameters.
visitors	Guest	Yes	Another subclass of User. These users will not have access to UoB SSO authentication, so will temporarily authenticate using QR codes in rooms.
facilities staff		No	Duplicate of administrative staff.
university accounts		No	Duplicate of existing authentication classes.
one-time QR code	QRCode	Yes	Temporary-use authentication codes, encoded as QR Codes according to ISO/IEC 18004:2015 and generated only by administrative users.
estimated		No	Duplicates estimate energy usage method.
student groups	Group	Yes	Arbitrary groupings of normal users, used for competitive/gamification purposes when compiling leaderboards.
courses		No	In its implementation, duplicates Group functionality.
departments		No	Again, in its implementation, duplicates Group functionality.
ranked	sortEntities	Yes	A method that sorts a supplied array of users, or of groups of users, for use in the leaderboard method.
leaderboards	displayLeadberboard	Yes	A method that generates and displays a leaderboard of users or groups of users, according to supplied parameters.
incentive		No	Refers to contextual impact and non-software organisational activities of its implementing organisation.
consumption		No	Duplicates energy usage estimate functionality.
energy		No	Duplicates energy usage estimate functionality.
reports	generateReport	Yes	A method that generates reports for system administrators. Such reports might contain temperature, lighting, occupancy, and user preference data for a given space over a given time.
behaviour		No	Refers to the way in which human users might interpret the application's output data.
energy usage		No	Duplicates energy usage estimate functionality.

Term		Acc.	Reason
recommendations		No	Insufficiently defined for ready implementation.
comparison		No	Term describes the
occupancy		No	A parameter of room objects.
noise levels		No	A parameter of room objects.
ambient temperatures		No	A parameter of room objects.
lighting levels		No	A parameter of room objects.
energy efficiency		No	Duplicates energy usage estimate functionality.
study space		No	Duplicates Room class.
map	generateMap	Yes	A method that will generate an interactive map of the campus, including building floorplans for multiple storeys. The map will be overlaid with a visual representation of the stored sensor-reading variables for those spaces, in which the magnitude of values is represented with a colour scale.
administrative users		No	Duplicates administrative user type.
graphs	generateGraph	Yes	A method that will generate a diagrammatic representation of the relationship between two variables, each measured along one of a pair of axes at right angles to each other.
override permissions		No	Describes the permissions of the administrator user type.
energy wastage		No	Wastage (in this context) is extremely difficult to calculate or define. It is more practicable to use the energy consumption metric.
abuse		No	An undesirable external human behaviour that the application will be designed to mitigate against.
proximity	checkUserProximity	Yes	Secondary authentication method that verifies a user's device is in or near a room, before that user's defined preference is acted upon in provision of building services.

ii Class-Responsibility-Collaboration Cards

Seven CRC cards are provided (find them at reference tables: B5d, B5e, B5f, B5g, B5h, B5h, B5i and B5j).

iii Class diagrams

A first cut class diagram (figure B5a) and a more-detailed class diagram (figure B5b) are provided.

User	
Responsibilities	Collaborators
Uses an authentication method to access the app	Students, Guest
Checks proximity to see if a user is in a certain room or not	Sensor, Room

Table B5d: CRC card: user.

Admins (extends User)	
Responsibilities	Collaborators
Generate graphs to compare expected and actual energy consumption of the students	Students
Generate one-time QR codes for visitor authentication	QRCode, Guest
Overrides student preferences to prevent mischievous behaviour	Students, Preference
Generate campus heatmap based on user activity	User, Room

Table B5e: CRC card: administrator.

Students (extends User)	
Responsibilities	Collaborators
Joins student groups for shared preferences	Group
Views leaderboards based on points	
Sets preferences for rooms	Preference
Views map of campus and rooms	Room
Views energy consumption for each room	Room
Views availability and capacity for each room	Room

Table B5f: CRC card: student.

Guest (extends User)	
Responsibilities	Collaborators
Uses a QR code to authenticate	QRCode
Use sensors to generate points based on behaviour	Sensor

Table B5g: CRC card: guest.

Sensor	
Responsibilities	Collaborators
Provides latest value for a given sensor	Database

Table B5h: CRC card: sensor.

Group	
Responsibilities	Collaborators
Contains a collection of students	Students
Stores as average of those students' energy consumption performance	Leaderboard

Table B5i: CRC card: group.

Preference	
Responsibilities	Collaborators
Provides functions to set and get preference values for a given student	Students , Database

Table B5j: CRC card: preference.

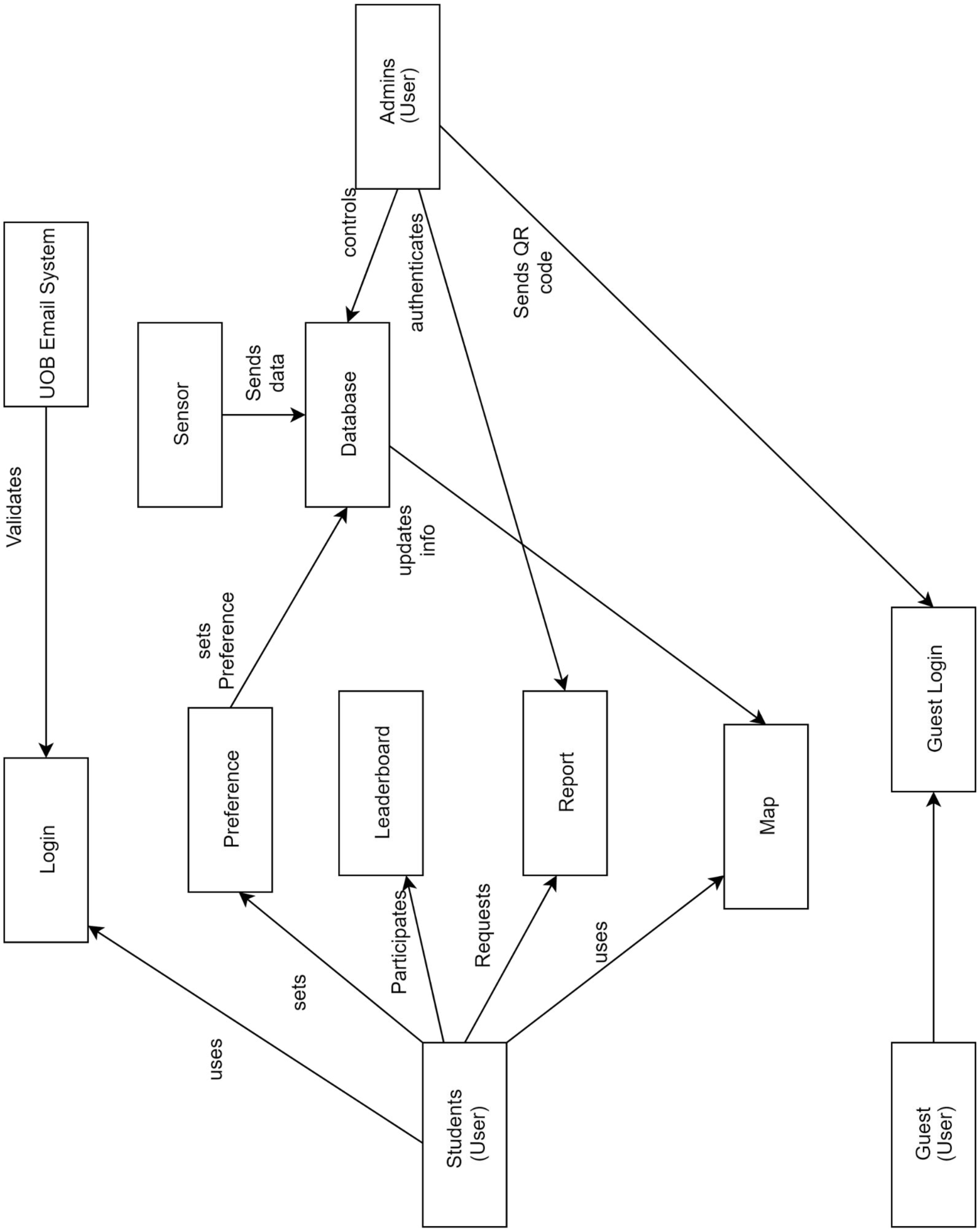


Figure B5a: First-cut class diagram.

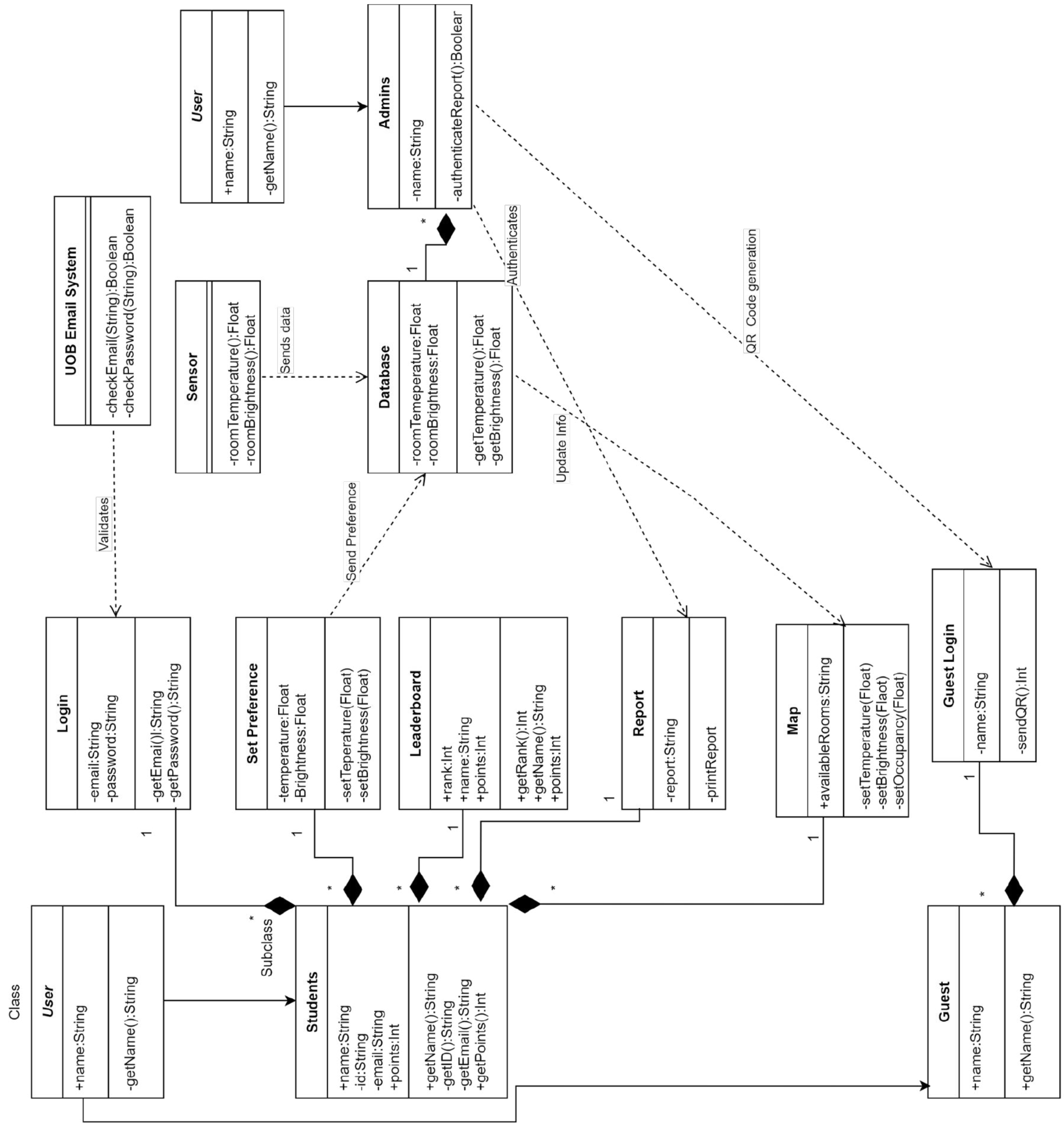


Figure B5b: Detailed class diagram.

B6 Object diagram

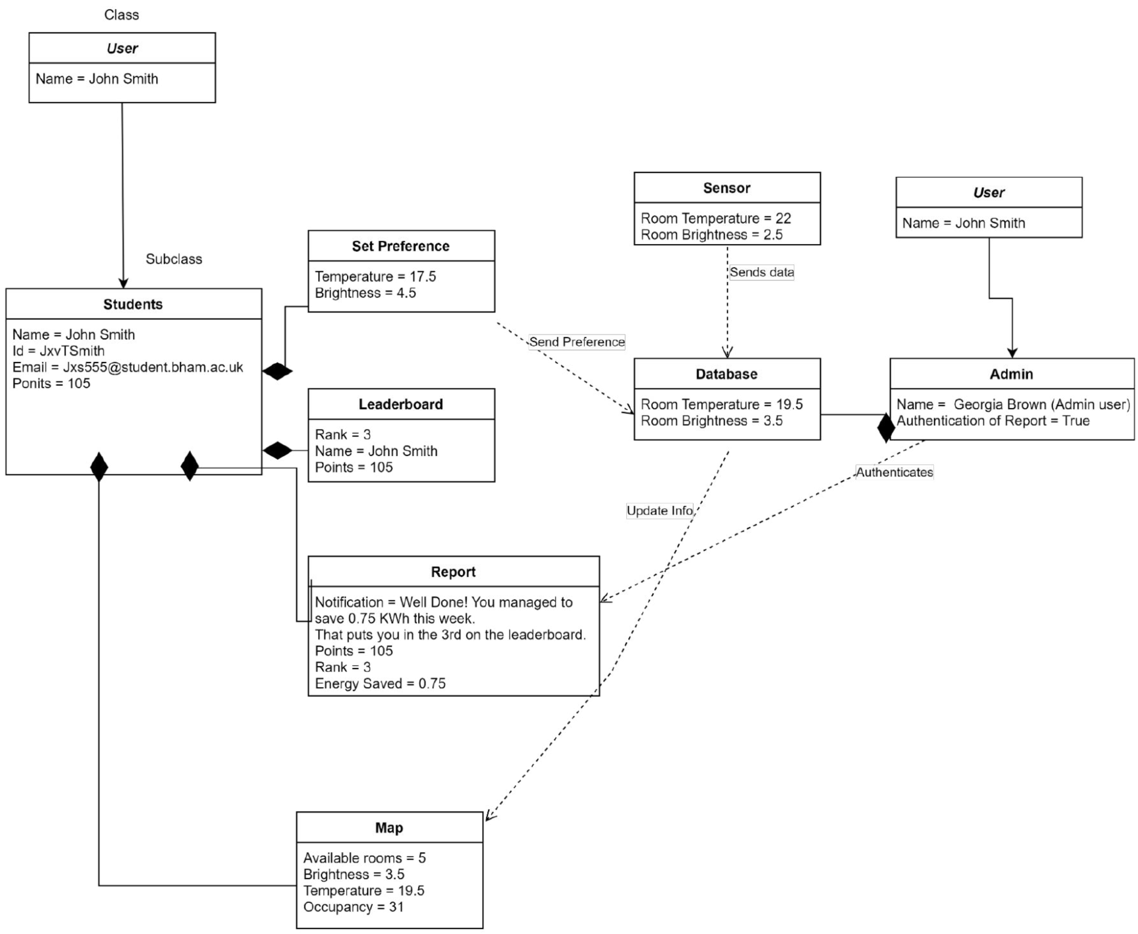


Figure B6a: Object diagram.

B7 Sequence diagrams

Two sequence diagrams are supplied: figure B7a addresses the updating of preferences by students, and figure B7b the creation of email reports.

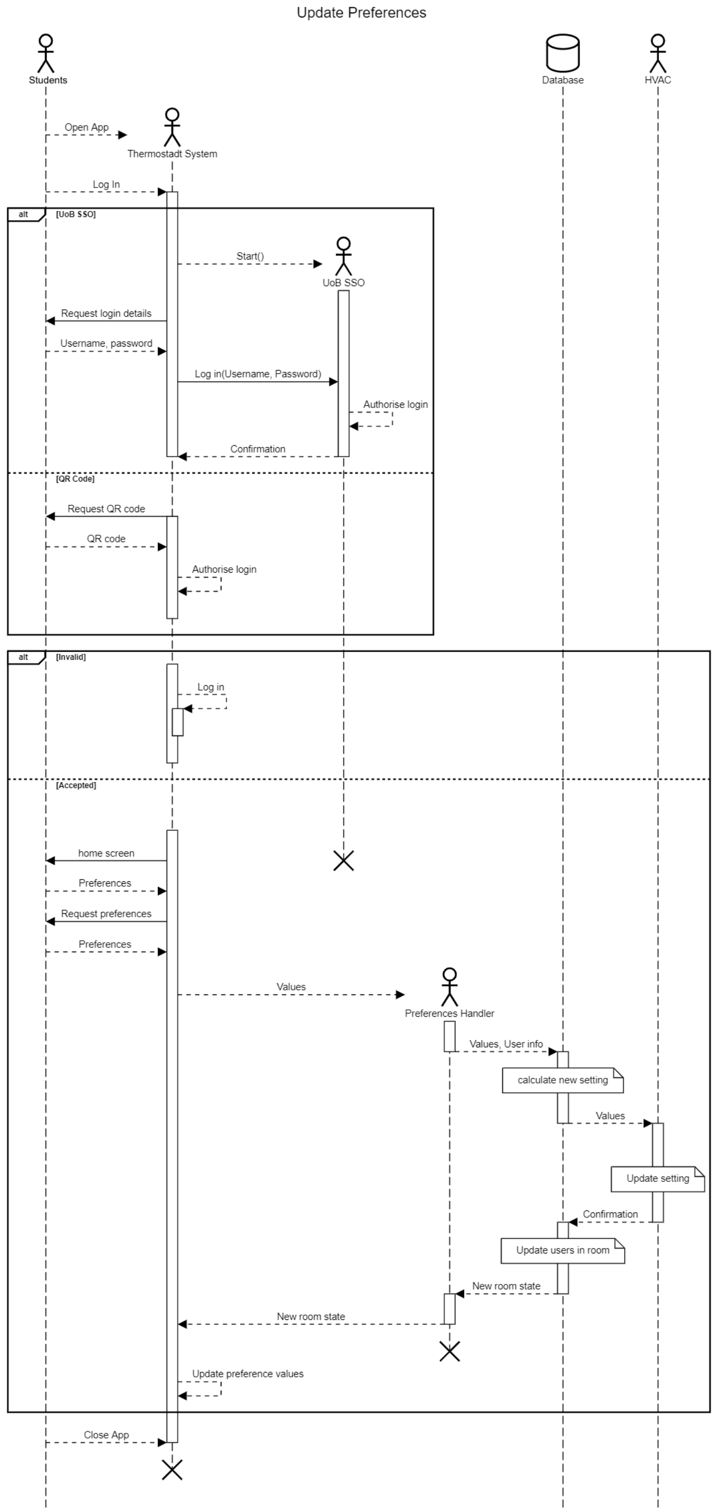


Figure B7a: Sequence diagram for the updating of preferences.

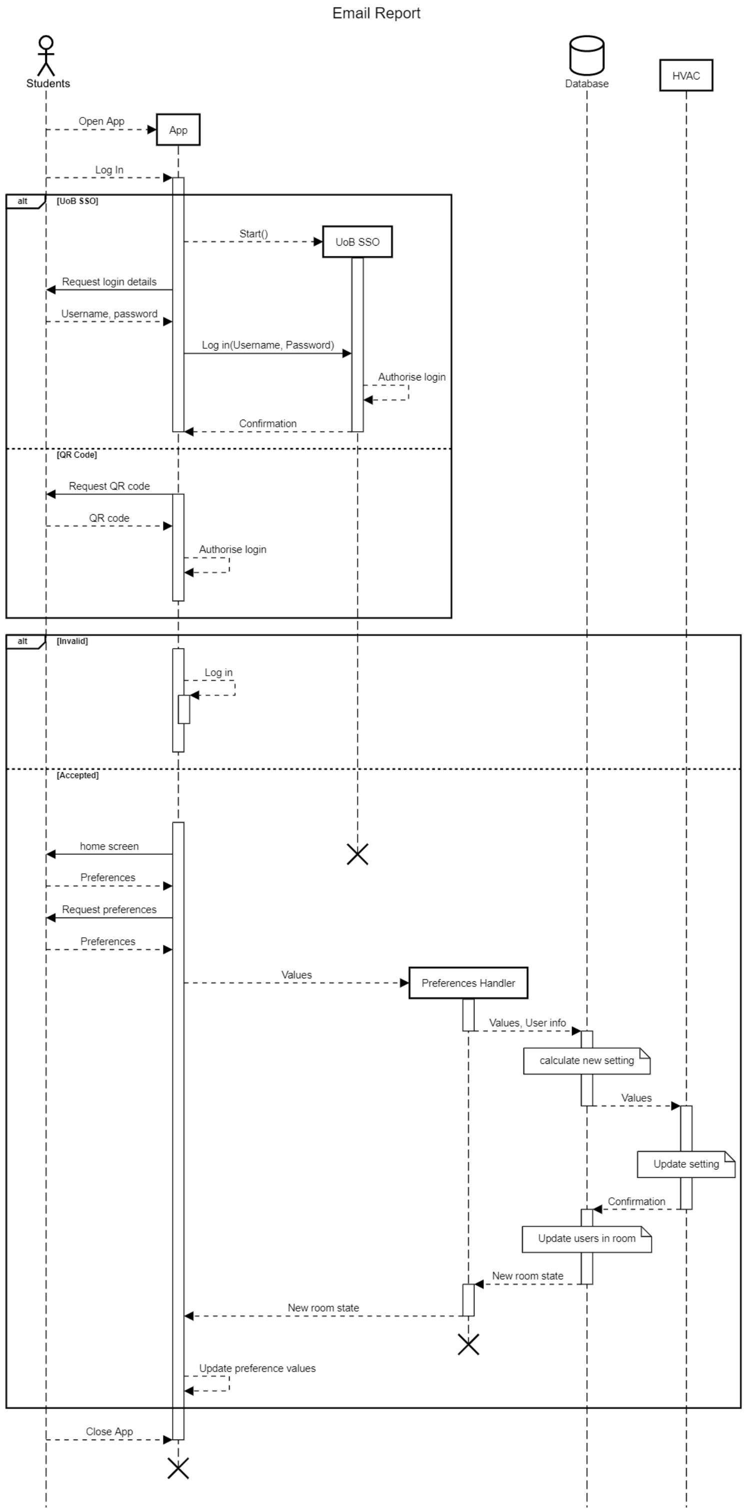


Figure B7b: Sequence diagram for the generation of an email report.

B8 State machine diagrams

The supplied state machine diagrams are figure B8a and figure B8b.

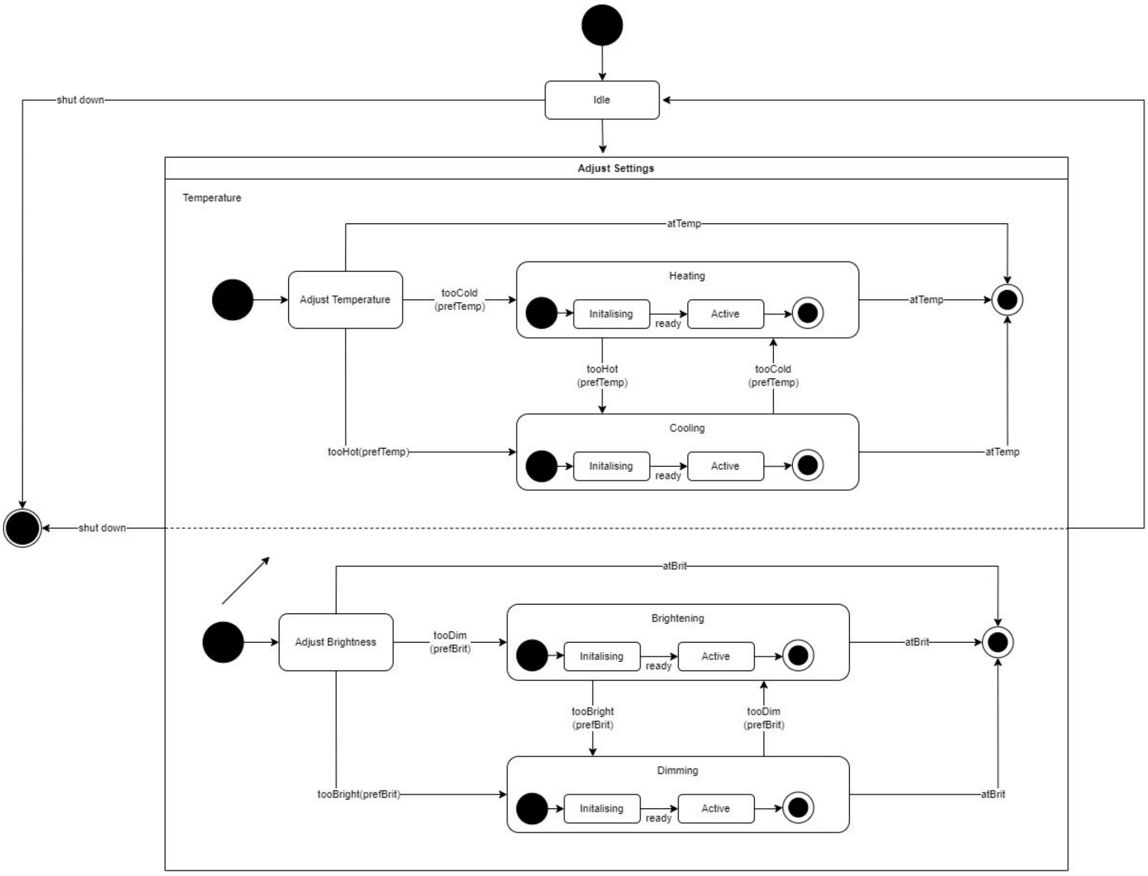


Figure B8a: State machine diagram.

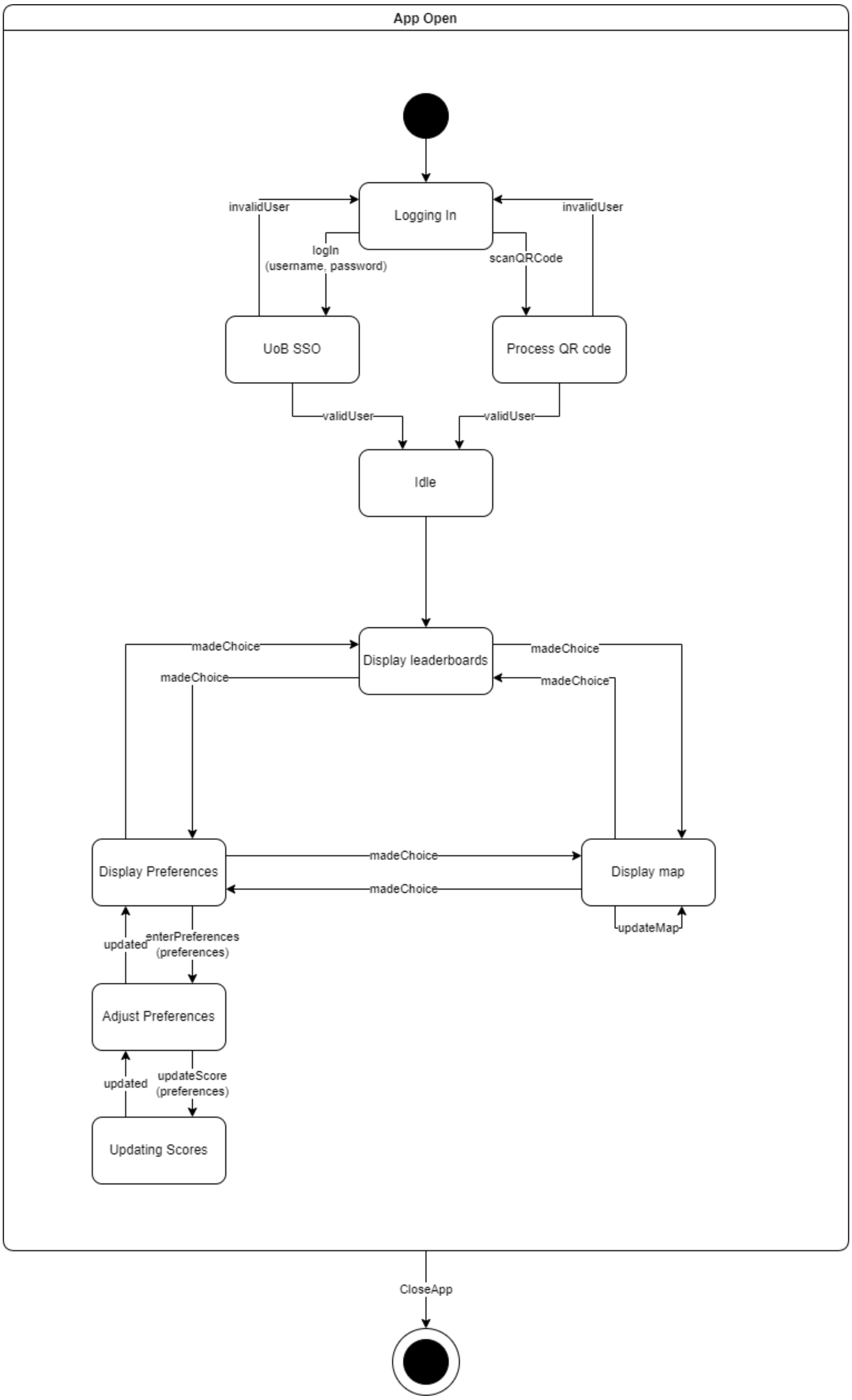


Figure B8b: State machine diagram.

C Software architecture style, modelling and evaluation

C1 Candidate architectures

We have considered 2 architectures for our software requirements and mobile application deployment:

MVC Model, View, Controller

BLoC Flutter's Business Logic Components

...with features and modules separation.

C2 Deployment diagrams

Deployment diagrams are available in figure C2a.

C3 Evaluation of candidate architectures

First, a comparison of the candidate architectures (table C3).

Figure C1a: Components diagram for BloC architecture.

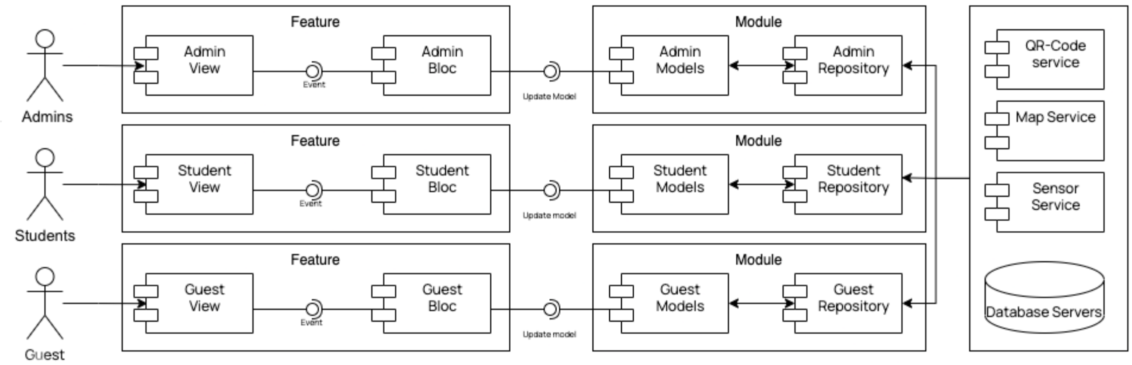


Figure C1b: Components diagram for MVC architecture.

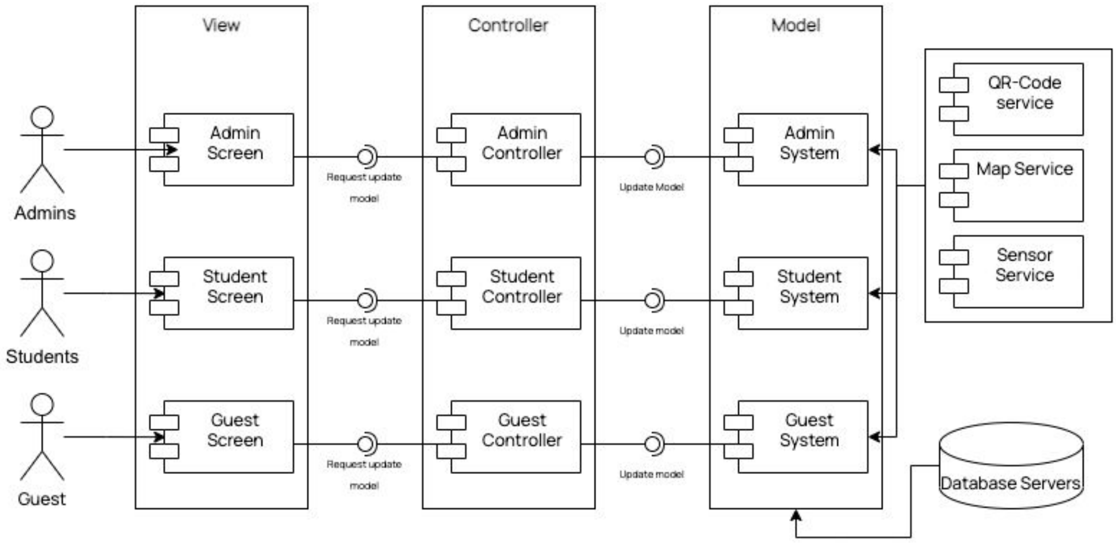


Figure C2a: Deployment diagram for the MVC architecture.

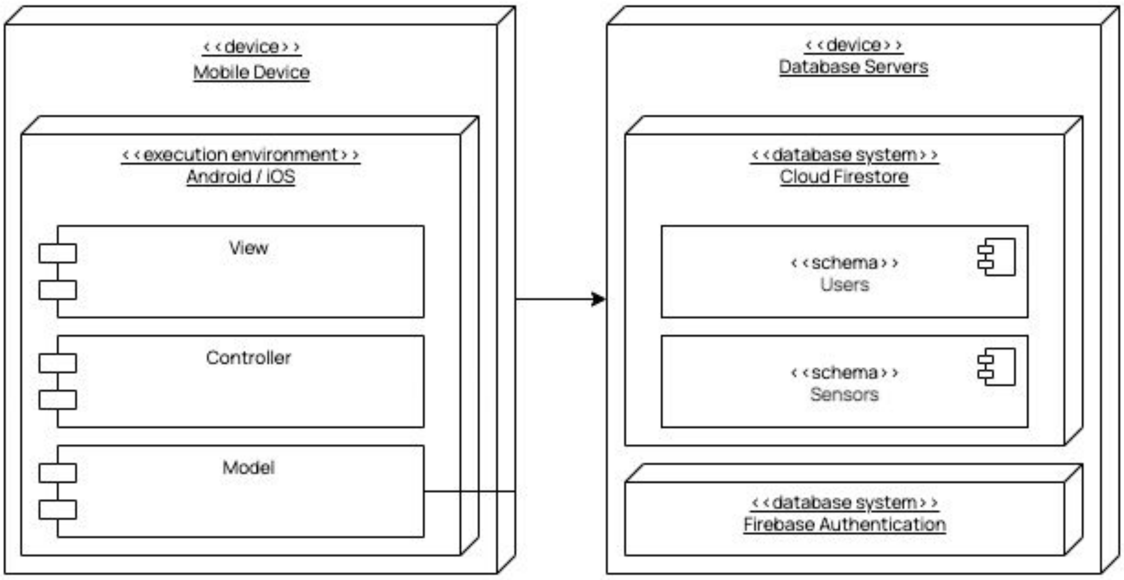
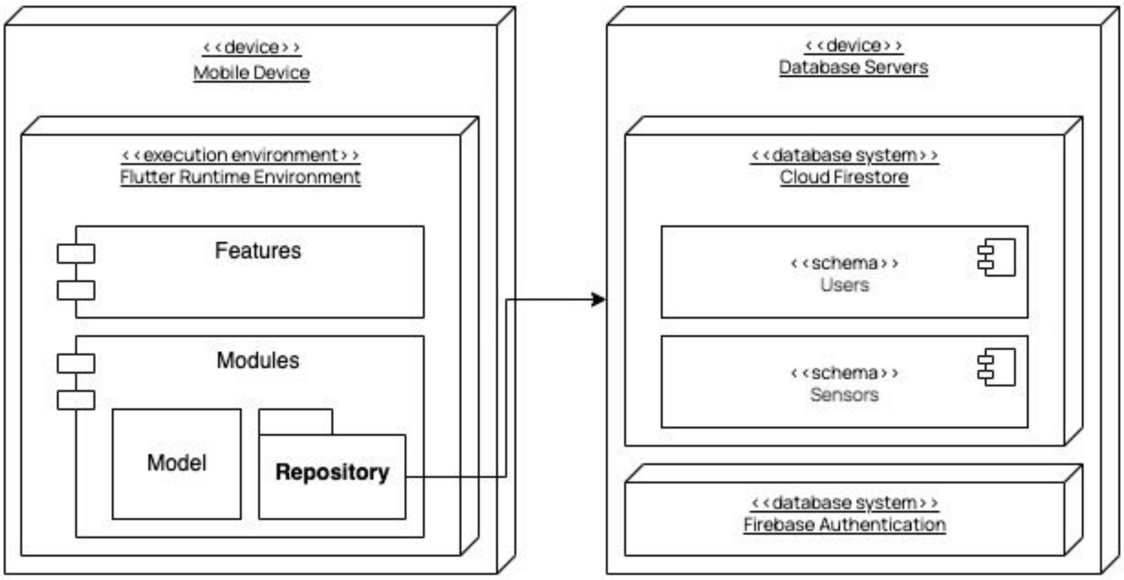


Figure C2b: Deployment diagram for the BLoC architecture.



MVC (Model, View, Controller)	BLoC with Features & Modules
More abstract, can be implemented in many ways depending on our software development stack	More strict, can be implemented only for mobile apps made in the Flutter framework
Requires more implementations for multiple platforms, e.g. Android & iOS	Due to the cross-platform nature of Flutter, requires only one implementation
Separation of models and user interface through a controller	Separation of models and user interface through a logic component
Might not have libraries available for reducing boilerplate code	Does have libraries available for reducing boilerplate code
Well known security standards and throughout documentation	Less known, not the most comprehensive documentation
Hard to scale and add functionality on top	Easy to scale due to its classification of models, logic and view in features and microservices
There is not a specific way to implement models	Models are implemented as abstract data sources, making modules easy to test through mocked or local databases

As long as we will choose Flutter as our tech stack, BLoC (Business logic components) architecture makes the most sense to be used for our project, due to its scalability but also ease of use.

As we can see, BLoC at its base is quite similar to MVC, though more refined and strict, which is a good option especially by saving costs and time implementing multiple software for different platforms.

D Software Testing

Our project should provide the user an app, which allows them to access the essential functionalities of the system in an interactive and easy-to-understand way. This application should also provide a simple way for users to sign-in and set their preferences for the HVAC system. To ensure the best possible user experience, there must be a test plan.

1. In order to adequately test our project, we aim to meet the following objectives during the testing phase.
2. The application matches the functionality defined by the non-functional and functional requirements
3. Any bugs detected by the system should be reported to the development team so they can fixed before the deadline
4. The application does not allow external users (guests) to affect the HVAC system.
5. Users can only interact with functionalities that have been given access to be the admin

D1 Test items

The main items to be tested include the application GUI, to ensure the app is easy to use, and the consistency between data on the database and user preferences on their mobile device. As this is a mobile application the items to be selected for testing must be functional on Android and iOS devices.

i Features to be tested

1. Functional requirements
   a) Must only be usable by authenticated students and staff of the university
   b) Must display a heatmap that represents the temperature of bookable rooms using colour scales
   c) Must allow users to input a temperature preference
   d) Must have a sign in mechanism
   e) Must be capable of issuing push notifications
   f) Should allow users to input a lighting preference
   g) Must output temperature preference data to a centralised HVAC system
2. Non-functional requirements

a) Should only permit temperature inputs in the range 10 to 30°C.

b) Must only count a user's energy consumption and only output their preference to the external HVAC system if their timetabled location and GPS location are within 50m of each other.

c) Colour scales in the heatmaps should be accessible to people with colourblindness.

ii Features not to be tested

All remaining requirements will not be tested due to the constraints on the testing phase. One of these constraints is ensuring we finish the project within the deadline specified, without compromising on the security of the system. Another constraint is the fact our team is limited by budget and availability of testing tools to test some of the requirement features, and so these requirements were not included into our testing phase.

D2 Approach

The testing will consist of both black box and white box testing to ensure both the implementation in principle and code is valid, and matches the requirements specified in this document. In order to ensure the testing process is of a high quality, the testing team will adopt various software quality assurance practices that will be maintained throughout the testing stage. One of the main quality factors that will be tested is consistency of user preference on the database and on their device

The testing team will use various black box testing methods, like robust BVT testing, to test the various features of the system. One example where the testing team is planning on using robust BVT is to check the requirement, 'should only permit temperature inputs in the range 10 to 30°C' as it is imperative the system does not allow temperatures outside of this range.

When a tester identifies a fault with the system, they should immediately report their findings to the software development team along with the corresponding actions taken to produce the fault. By adapting the strategy, the development team can fix the bugs in the code at an earlier stage. Once the development team has fixed the problem, they should notify the testers that it has been fixed, so the testers can assess if the changes made have caused potential faults in other parts of the system. When a test has been completed the testers should mark it with a pass, fail or incomplete mark.

D3 Pass-Fail criteria

ID	Test description	Test steps	Test Data	Expected result
TD-01	Verify the login with valid studentID and password	Launch app Enter Student-email Enter Password Click Submit	Email: subhaan@bham.ac.uk Password: pass1	Student should be able to login and use the system

ID	Test description	Test steps	Test Data	Expected result
TD-02	Verify the login with invalid student email and valid password	Launch app Enter Student-email Enter Password Click Submit	Email: subhaan@bham.ac.uk Password: pass1	User should not Login into the app and should return invalid email or password
TD-03	Verify the Set temperature preference works for students and database has correct value	Go to set preference page on app Set temp preference Save preference	Input: 10	App should confirm preference has been saved. The data base should have the correct corresponding temperature preference saved for user logged in.
TD-04	Verify app does not mix user preferences when saving to database	2 concurrent users Both input similar temperatures Click save preference at the same time	User a: 21 User b: 20	The centralised HVAC database should have instances of user preference saved. There should be a field for user a with temperature set to 21, and another for user b with temperature set to 20.
TD-05	Verify system only calculator user's consumption when in the specified range	Launch app with GPS on Stand within 10 meters of the specified location on the timetable for 10 minutes. Walk to 51 meters from the specified range and wait 10 more minutes.	email: subhaan@bham.ac.uk password: pass1 GPS: on	The system should compute the correct user consumption for the user "subhan" when within the specified range.
TD-06	Verify the system stops calculating user's consumption once out of range	Launch app with GPS on Stand within 10 meters of the location specified on timetable for 10 minutes Walk 51 meters from the specified range and wait 10 mins. Repeat process twice.	email: subhaan@bham.ac.uk password: pass1 GPS: on	The system should correctly compute a users consumption only when they are within the range. For this case it should only compute users' consumption for a total of 20 minutes.
TD-07	Verify the colour scale heatmap is useable for users with color blindness	Launch app Click on the settings icon Select the accessibility field Input deuteranomaly colour blindness	Colour Blindness: true Type: deuteranomaly	The app should change the colour scheme of the map so it can be accessible for users with colour blindness. Testers can test colour scheme with a pre printed palette for deuteranomaly compatible colour scheme

ID	Test description	Test steps	Test Data	Expected result
TD-08	Verify temperatures out of range can not be inputted by user	Launch app Go to set preference page on app Input temperatures Click save preference	Input 1: 9 Input 2: 10 Input 3: 11 Input 4: 20 Input 5: 29 Input 6: 30 Input 7: 31	Inputs 1 and 9 should not be able to be saved. The app should tell user that they are trying to input data outside range. Inputs 2 to 6 should proceed as normal.
TD-09	Verify accuracy of heatmap	Launch app Select heatmap option	Set the temperature of Café Go (at University Centre) to 15°C Set the temperature for the food court at University Centre to 23°C	The app should have the Café Go shaded with a light orange colour and the food court should be shaded with a darker orange colour

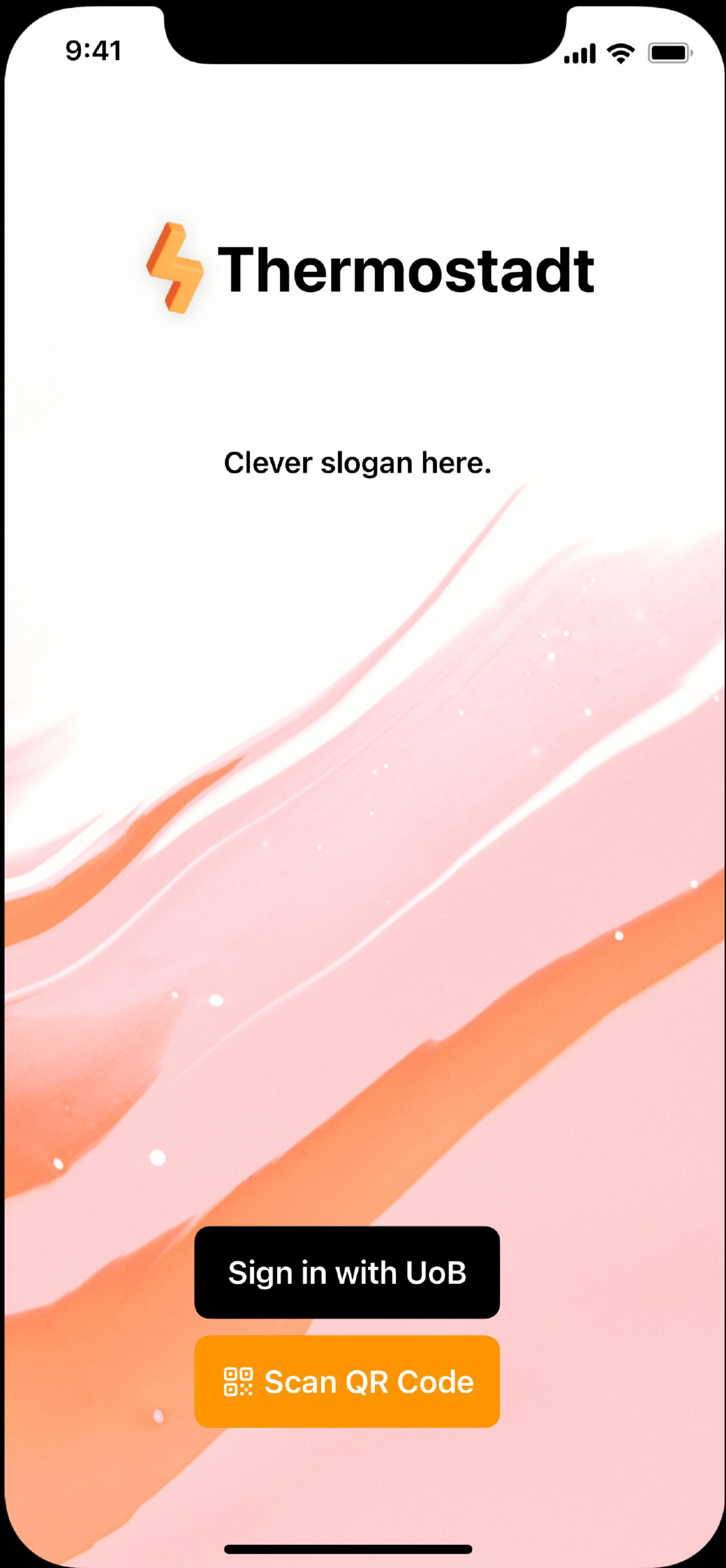
Exit criteria

When 98% of items pass the testing criteria, testing will conclude.

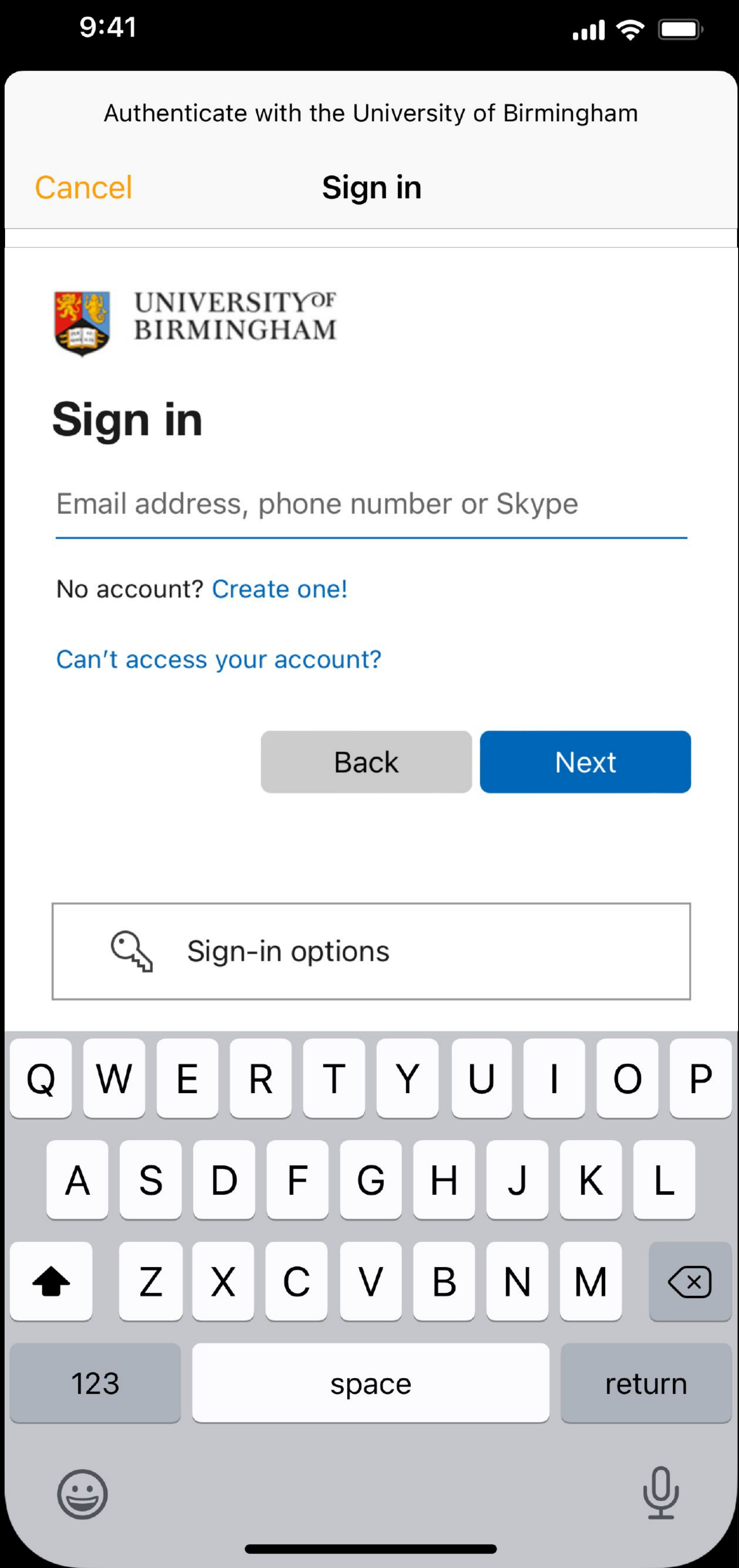
E Usability and Prototyping

E1 Prototyping

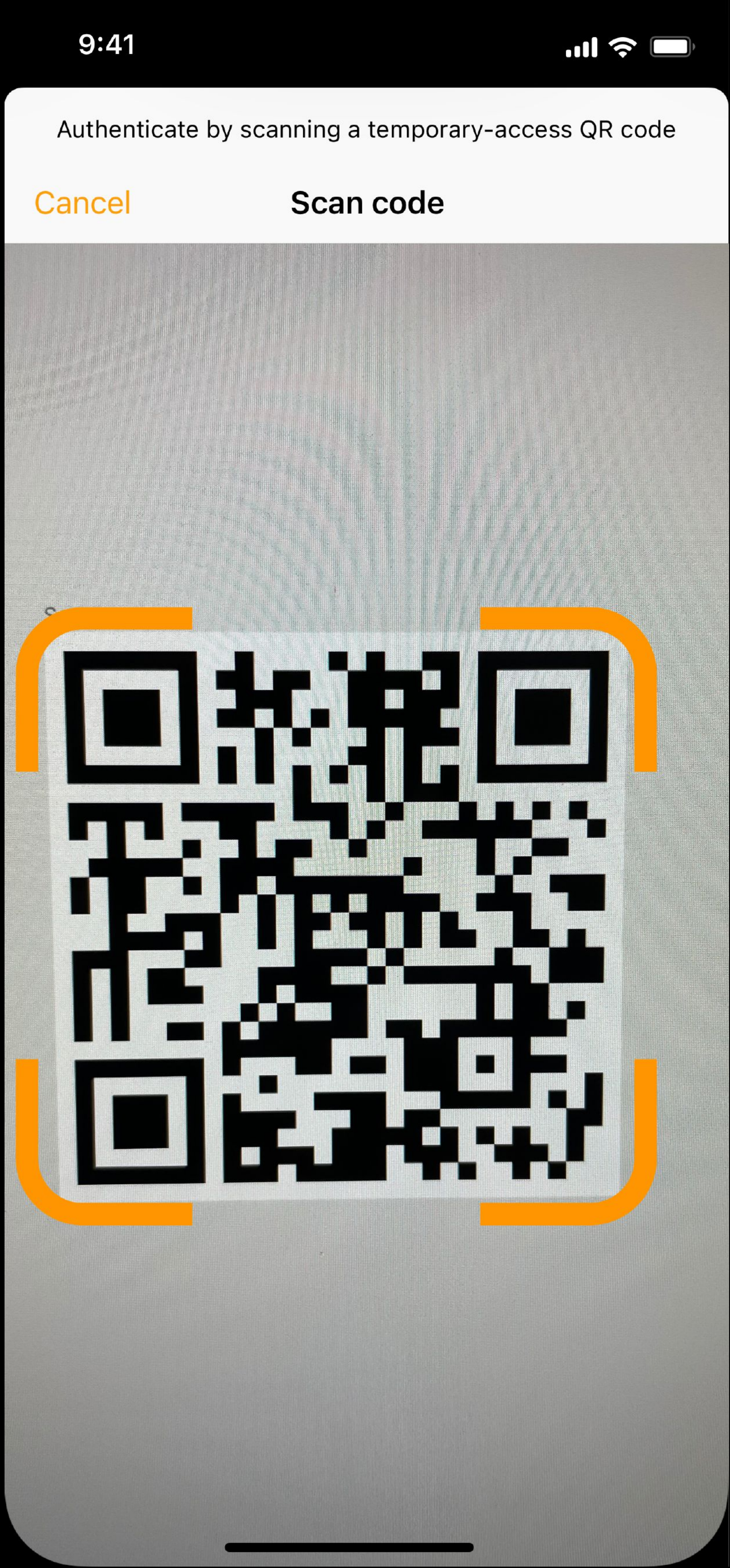
Our primary prototype is embedded in the following pages. Additionally, an initial working protoype is included in the appendix.



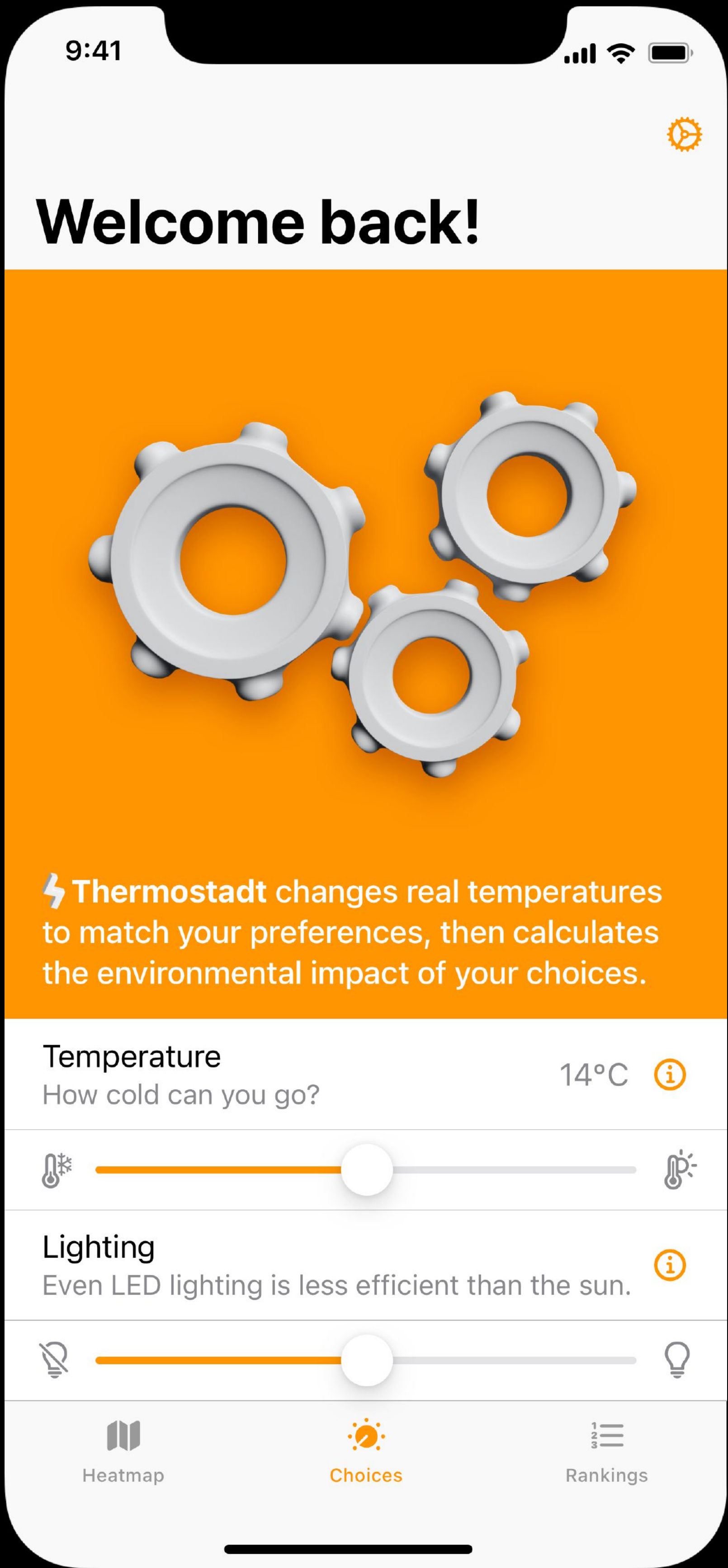
For new or unauthenticated users, the app launches to a splash screen. Users have two options: sign in with a UoB account, or scan a QR access code. There is no account registration functionality.



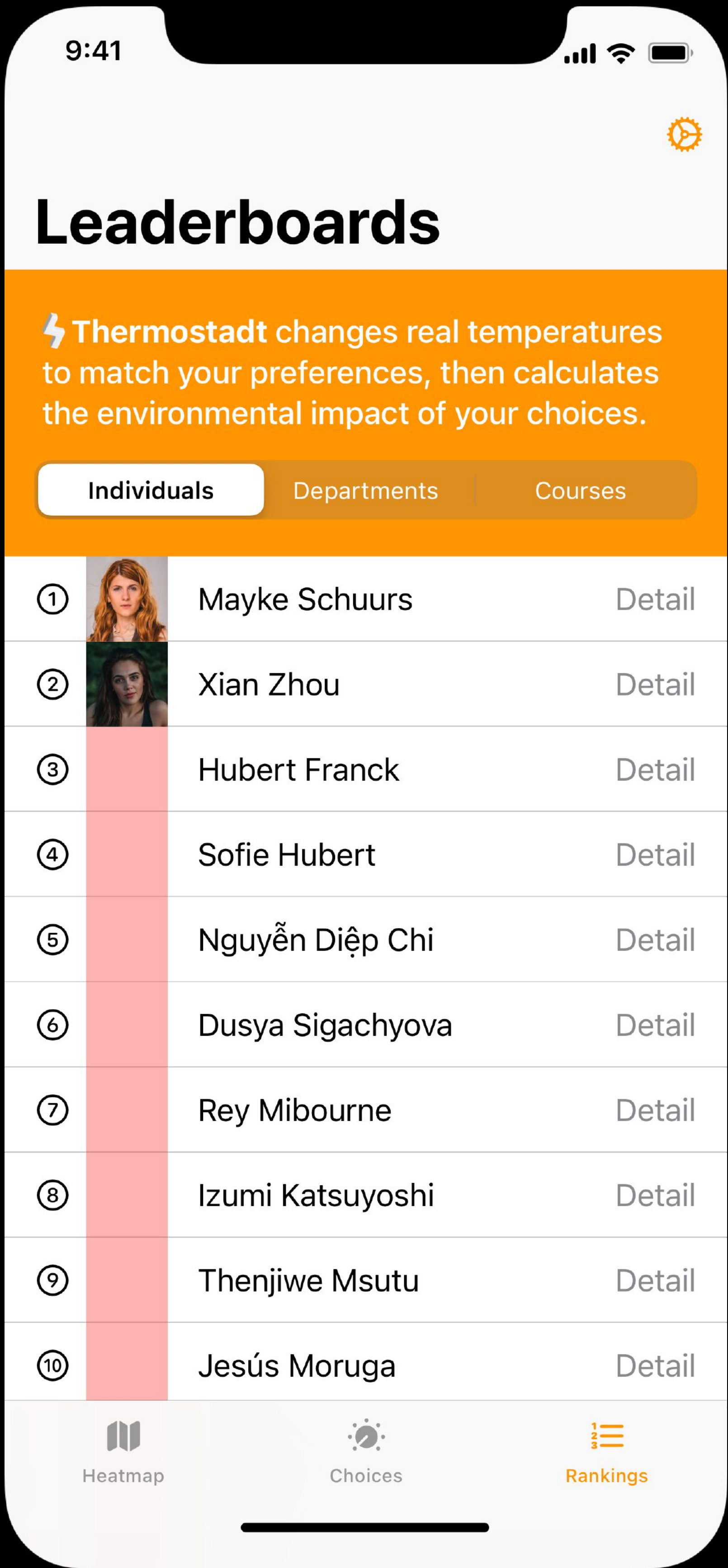
We implement SSO using the UoB's AD FS. Microsoft's sign-in page is framed on a modal sheet.



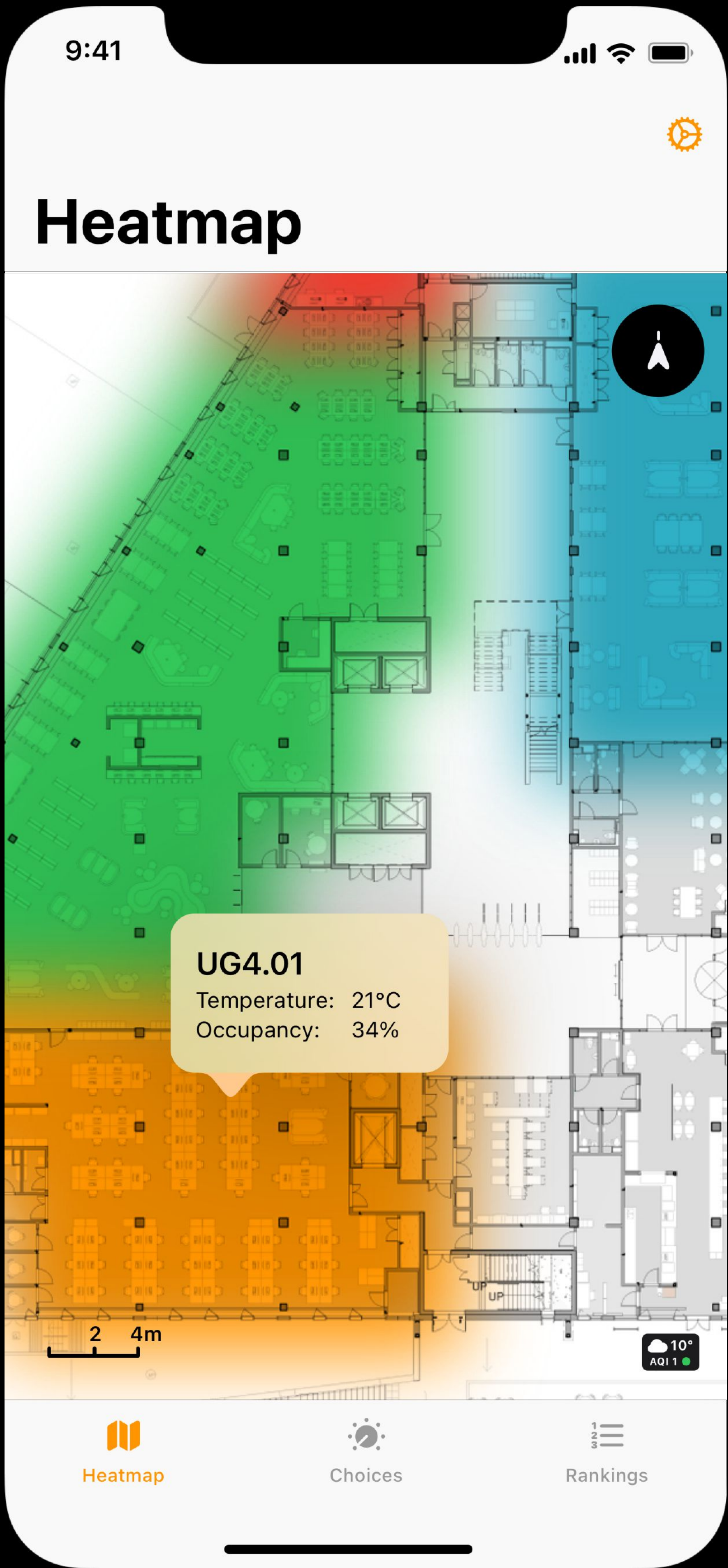
Here, non-UoB users (i.e. visitors) temporarily authenticate their devices by scanning a physical QR code.



For authenticated users, this view is the default. Here, users set their global temperature and lighting preferences.



Leaderboards are available by individuals and for groups.



In the heatmap, users explore collected sensor data from rooms on campus. More specific, room-by-room sensor data is available in popovers.

Periodic reports on user performance are implemented as push notifications.

E2 Video

Included in the Canvas submission.

F Ethics and Professional Practice

This ethical appraisal is organised to follow the seven general principles described in the EEE/ACM code of ethics.

How does the app contribute to society and to human well-being? Thermostadt complies with principle 1.1 of the ACM Code of Ethics as it is designed to encourage behavioural changes that reduce university fossil-fuel usage and emissions. This outcome is a positive contribution to society.

This implementation targets a relatively small number of active users (members of the UoB), but seeks to deliver a small marginal benefit (i.e. mitigation of the climate crisis) to humanity.

How does the app avoid harm? The system complies with principle 1.2 of the ACM Code of Ethics by using manual moderation to prevent abuse of the system. For instance, collusion between users to induce extreme temperatures is impossible. University infrastructure will not be dependent on Thermostadt, so if failures occur, they should not cascade into other systems and cause dangerous real-world conditions (e.g. black-outs).

However, Thermostadt is premised on the personal 'carbon footprint' concept. This concept has been accused of diverting consumers away from the fossil fuel industry, and towards ineffectual climate action and navel-gazing.

How is the app honest and trustworthy? This appraisal has provided an honest view of our ethical successes and failings.

In Thermostadt's implementation, data is displayed accurately through regular updates. Assumptions, weightings and roundings are, wherever possible, disclosed to users.

How is the app fair? Access to Thermostadt may not be universal. Individuals who do not have Thermostadt are 'disenfranchised': their peers have power that they do not.

Some people require particular conditions (e.g. a susceptibility to the cold for a medical reason, or a visual impairment that is exacerbated in dim lighting conditions). These needs have traditionally been accounted for by building managers or a simple light switch. In our system, no equivalent exists.

How does the app respect intellectual property? The sensor infrastructure that supports Thermostadt is supplied by the University of Birmingham's partnership with Siemens. Thermostadt interfaces with Siemens' tech, but does not claim ownership of it.[1] Therefore, the system complies with principle 1.5 of the ACM Code of Ethics.

[1] The core idea for the app (adjusting a room's heating and lighting according to the preferences of the people in it) was described by Bill Gates in his 1995 book, 'The Road Ahead', albeit in a domestic context.

How does the app respect privacy? Thermostadt requires that users volunteer information about themselves: their preferences and their location. The former is provided by users. The latter, with the user's consent, is used for location-verification and checked on an hourly basis. Users will be aware that they have surrendered this information. Thus, complying with principle 1.6 of the ACM Code of Ethics.

The heat map functionality exposes a 'fuzzified' interpretation of sensor data on campus. It's unlikely this data exposure is a breach of privacy, but it might be unintentionally connected to an identifiable individual.

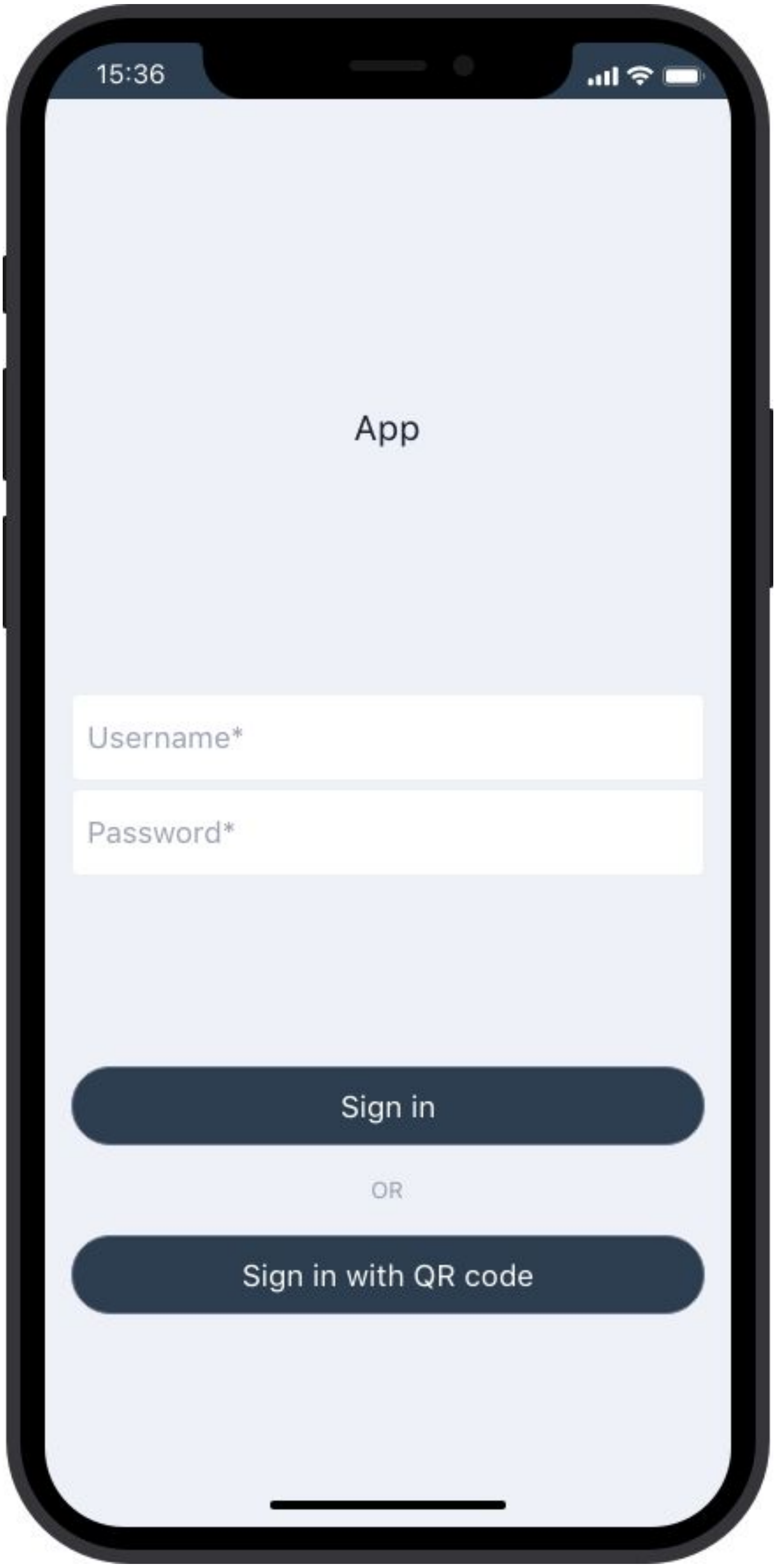
How does the app honour confidentiality? The system complies with principle 1.7 of the ACM Code of Ethics as it offers partial anonymisation of user data is possible without any impact upon functionality. Relatively little user data needs to be stored: there is no function that retains a user's location history.

List of Tables

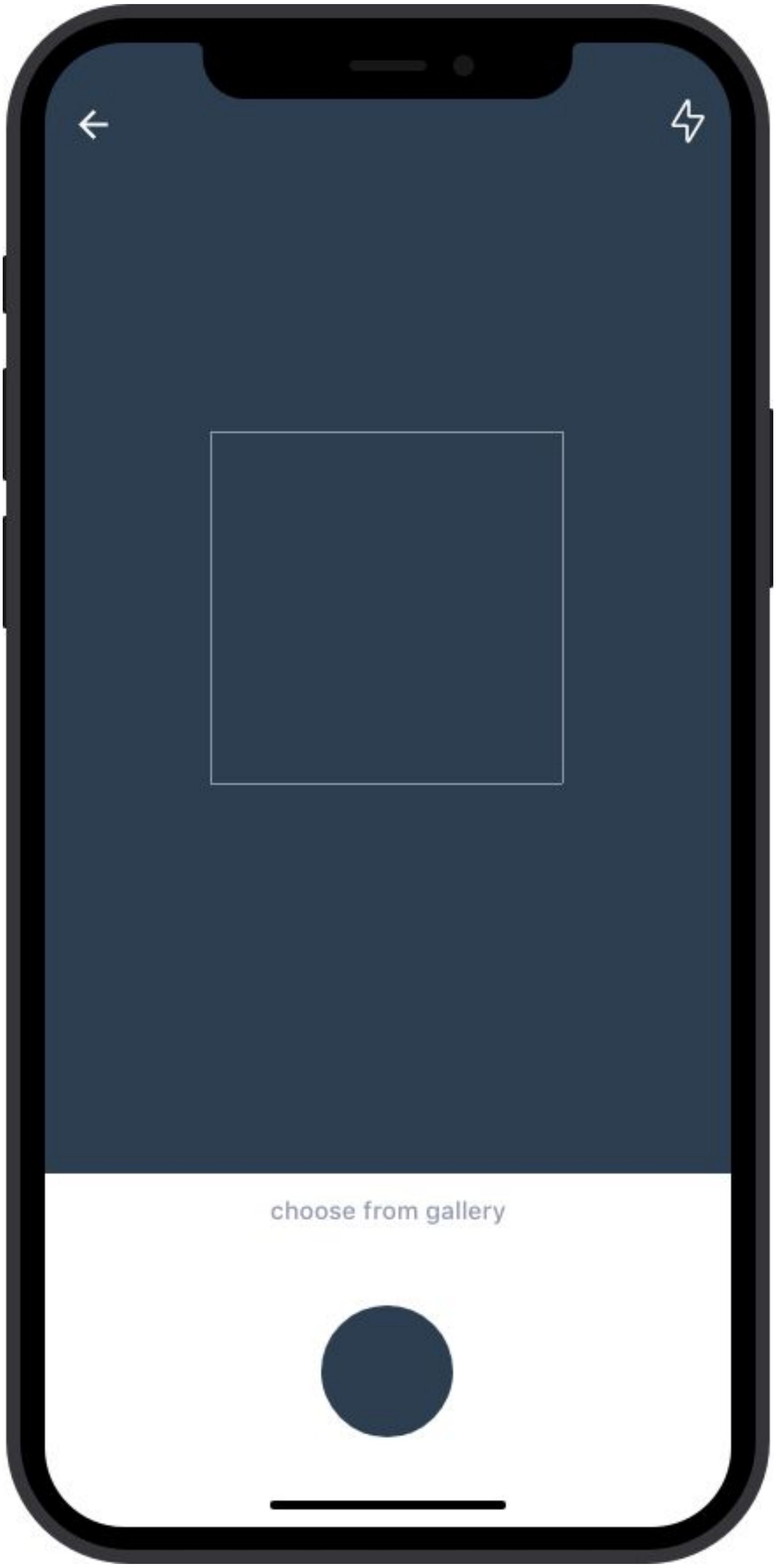
List of Figures

Initial prototype

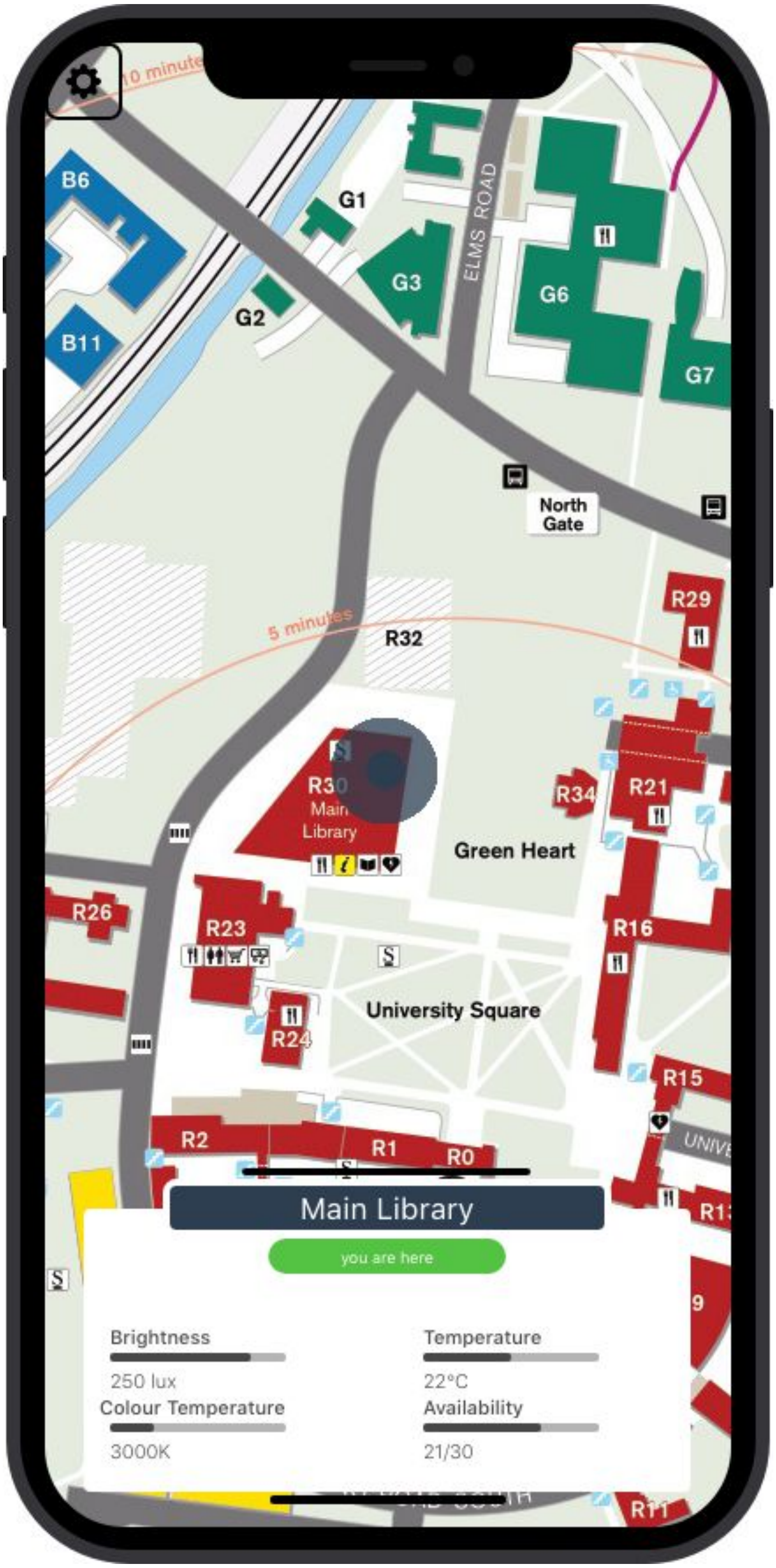
In order to help document our progress, we have included our 'initial' prototype, which was useful in communicating the app to the team. As before, the PDF has been merged with the report, so simply continue to scroll.



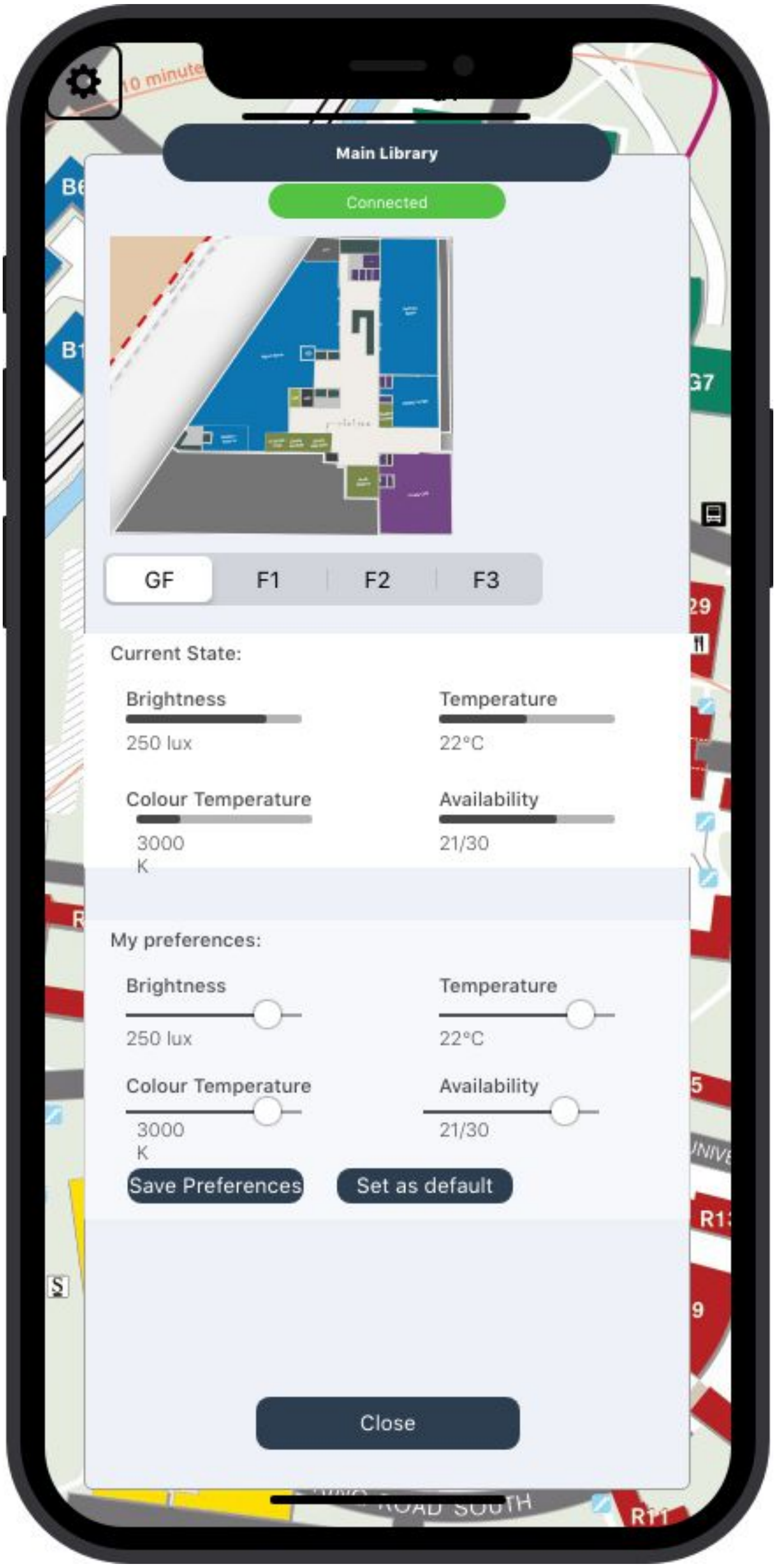
This is the first screen that new users of the app encounter. Users have two options: sign in with a UoB account, or scan a QR access code. There is no account registration functionality.



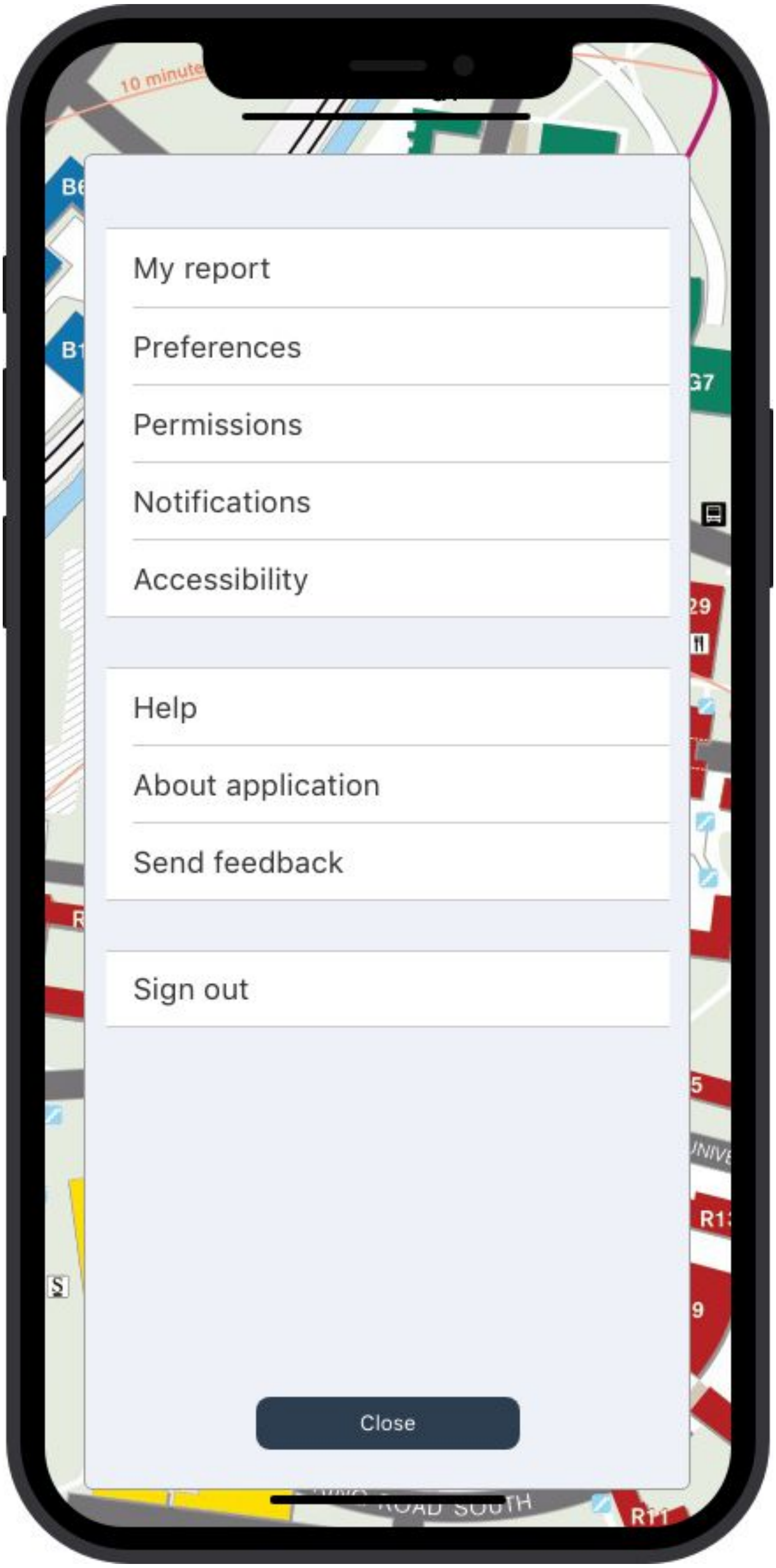
Here, non-UoB users (i.e. visitors) temporarily authenticate their devices by scanning a physical QR code.



In the heatmap, users explore collected sensor data from rooms on campus. A limitation: the university's campus map is a conventional map, not an interior floorplan.



More fine-grained sensor data is available for each building on a modal sheet. In this prototype, the user preference input is located inside these modals. (It didn't stay there.)



App navigation is accessed by tapping a at the top left of the screen.