

Mobile Application Location Based Tracker

A dissertation submitted in partial fulfilment of  
the requirements for the degree of

BACHELOR OF SCIENCE  
in Computer Science

in  
The Queen's University of Belfast

by  
Dean Meehan

9<sup>th</sup> May 2017

**SCHOOL OF ELECTRONICS, ELECTRICAL ENGINEERING and COMPUTER  
SCIENCE**

**CSC3002 – COMPUTER SCIENCE PROJECT**

**Dissertation Cover Sheet**

A signed and completed cover sheet must accompany the submission of the Software Engineering dissertation submitted for assessment.

Work submitted without a cover sheet will **NOT** be marked.

Student Name: Mr Dean Meehan

Student Number: 40041181

Project Title: Mobile Application Location Based Tracker

Supervisor: Dr Paul McMullan

**Declaration of Academic Integrity**

Before signing the declaration below please check that the submission:

1. Has a full bibliography attached laid out according to the guidelines specified in the Student Project Handbook
2. Contains full acknowledgement of all secondary sources used (paper-based and electronic)
3. Does not exceed the specified page limit
4. Is clearly presented and proof-read
5. Is submitted on, or before, the specified or agreed due date. Late submissions will only be accepted in exceptional circumstances or where a deferment has been granted in advance.

**I declare that I have read both the University and the School of Electronics, Electrical Engineering and Computer Science guidelines on plagiarism - <http://www.qub.ac.uk/schools/eeecs/Education/StudentStudyInformation/Plagiarism/> - and that the attached submission is my own original work. No part of it has been submitted for any other assignment and I have acknowledged in my notes and bibliography all written and electronic sources used.**

*Student's signature*

*Date of submission*

## **Acknowledgements**

The journey for completing this dissertation had been circuitous. I would like to thank Queens University Belfast for the amazing opportunity to complete my study here and the staff that I have had the pleasure of working with throughout the past few years. I would like to give a special thanks to Dr Paul McMullan for the support in completing this project and the many fellow students I've met at my time here who are going through the same journey in their help, support and enthusiasm.

I wouldn't be here if it wasn't for the first computer I got to call my own. I would like to thank the generosity of Mrs Collette Somerville in helping me to stay curious and begin my journey into the field of Computer Science.

I also wish to thank everyone within the stack overflow community for their sharing of ideas, solutions and problems that will help pave the way for a better future.

## **Abstract**

Having a strategic approach to space utilisation is important to organisations, from managing building occupancy to individual meeting and classrooms; whether they have too much space or are a growing workforce running out of space. It can be difficult to tackle this problem without the use of technology.

This project aims to solve the problem by evaluating the utilisation of existing spaces using a cloud based solution in conjunction with a mobile location based tracker that is able to automatically record attendance within events, providing valuable feedback on all aspects of a buildings usage.

# Table of Contents

<b>1</b>	<b>INTRODUCTION AND PROBLEM SPECIFICATION .....</b>	<b>7</b>
1.1	INTRODUCTION .....	7
1.2	GOALS OF THIS ASSIGNMENT .....	7
1.3	SOLUTION APPROACH .....	8
1.4	SOFTWARE AND HARDWARE ENVIRONMENT.....	8
1.4.1	Server.....	8
1.4.2	Dashboard.....	8
1.4.3	Mobile Client.....	9
1.5	SUCCESS CRITERIA .....	9
1.5.1	Acceptance Tests .....	9
1.6	DEVELOPMENT PLAN.....	10
1.6.1	Development Software.....	11
<b>2</b>	<b>SYSTEM REQUIREMENTS SPECIFICATION .....</b>	<b>11</b>
2.1	SYSTEM ARCHITECTURE.....	11
2.2	SERVER OVERVIEW .....	12
2.2.1	Server Database .....	12
2.3	MOBILE APPLICATION OVERVIEW.....	12
2.3.1	Tracking Technologies.....	13
2.4	DASHBOARD OVERVIEW .....	14
2.5	LOCATION DASHBOARD OVERVIEW .....	14
2.6	FUNCTIONAL REQUIREMENTS.....	14
2.6.1	Server .....	14
2.6.2	Mobile Application.....	15
2.6.3	Dashboard.....	16
2.6.4	Location Dashboard.....	17
2.7	NON-FUNCTIONAL REQUIREMENTS.....	17
2.7.1	Server.....	17
2.7.2	Mobile Application.....	17
2.7.3	Dashboard.....	18
2.7.4	Location Dashboard.....	18
2.8	HARDWARE AND SOFTWARE REQUIREMENTS .....	18
2.9	DASHBOARD APPLICATION PROGRAMMING INTERFACE (API).....	19
2.9.1	Authentication .....	19
2.9.2	Generic Routes .....	20
2.9.3	User Routes .....	20
2.9.4	Event Routes.....	20
2.9.5	Location Routes.....	21
2.9.6	Place Routes.....	22

2.9.7	<i>Statistics Route</i> .....	22
2.9.8	<i>Polling Route</i> .....	22
<b>3</b>	<b>DESIGN</b> .....	<b>23</b>
3.1	SYSTEM ARCHITECTURE .....	23
3.1.1	<i>Cloud Server</i> .....	24
3.1.2	<i>Mobile Application</i> .....	24
3.1.3	<i>Dashboard</i> .....	25
3.2	DATABASE DESIGN .....	25
3.2.1	<i>Database Security</i> .....	26
3.3	SCHEMA DESIGN .....	26
3.3.1	<i>Event Schema</i> .....	26
3.3.2	<i>Location Schema</i> .....	27
3.3.3	<i>Place Schema</i> .....	28
3.3.4	<i>User Schema</i> .....	28
3.4	USER INTERFACE DESIGN .....	28
3.5	CODE ARCHITECTURE .....	46
<b>4</b>	<b>IMPLEMENTATION AND TESTING</b> .....	<b>46</b>
4.1	CHOICE OF LANGUAGES .....	46
4.1.1	<i>Server</i> .....	46
4.1.2	<i>Mobile Application</i> .....	47
4.1.3	<i>Dashboard</i> .....	47
4.2	SERVER ENVIRONMENT .....	47
4.3	DASHBOARD ENVIRONMENT .....	48
4.4	MOBILE APPLICATION REQUIREMENTS .....	49
4.5	TESTING .....	50
4.5.1	<i>Unit Tests</i> .....	50
4.5.2	<i>Regression Tests</i> .....	50
<b>5</b>	<b>SYSTEM EVALUATION</b> .....	<b>51</b>
5.1	FUNCTIONALITY .....	51
5.2	SYSTEM RELIABILITY AND SPEED .....	51
5.3	PERFORMANCE .....	51
5.4	INSTALLATION AND OPERATION .....	52
5.5	COST .....	52
5.6	USER FEEDBACK .....	52
<b>6</b>	<b>CONCLUSION</b> .....	<b>53</b>
<b>7</b>	<b>REFERENCES</b> .....	<b>54</b>
<b>8</b>	<b>APPENDICES</b> .....	<b>55</b>

# 1 Introduction and Problem Specification

## 1.1 Introduction

Many companies, universities, and organisations would benefit from having a strategic approach to space utilisation from managing building occupancy to individual meeting and classrooms. Having a centralised space management application would maximise the potential of existing spaces and allow for strategic analysis of building management.

Building are an inherently expensive asset to any business with the cost of a city-centre heated office in the UK and Ireland averages around £1,750/m<sup>2</sup> and an out-of-town business park location costing on average £1,400/m<sup>2</sup> [1]

Large organisations with many buildings, a growing workforce, or growing customer base such as universities, training academy's and corporations may find that space is becoming more precious as they grow or they have a lot of space that isn't being utilised. Providing a system that allows facilities managers have an overview of a locations usage can elevate the difficulties in decision making regarding the management, maintenance, and operating costs allowing these organisations to save money.

## 1.2 Goals of this assignment

This project aims to solve the problem by developing a cloud-based system that can organise the spaces within a building and provide features for organising and managing meetings, events or other activities involving rooms, services and people. The system will be able to use existing wireless technologies such as mobile tracking through GPS, Bluetooth beacons and using existing infrastructure already in buildings such as wireless internet switches to identify when people are using the space, recording the buildings utilisation in real-time allowing building administration to make more informed decisions.

End devices such as web browsers, mobile computer tablets and mobile phones can be used to provide a flexible, user-friendly interface to a building, providing a better scope of room utilisation so they can quickly and easily find available spaces, book rooms, and check attendance at meetings. Devices such as tablets can be mounted beside meeting rooms to allow the checking of availability of a particular room, providing live information about meetings taking place such as the meeting name, people attending and times.

### 1.3 Solution Approach

The project will be a cloud based application that will do most of the processing remotely with the end user device such as web browsers, mobiles and tablets able to use their internet connection to connect to a central server. The cloud service should store information about buildings, rooms, companies and users allowing end devices to login and request rooms with mobile devices will report on their location using assigned beacons placed in rooms.

The first step of this project would be to create a cloud-based application with API endpoints that will store all information regarding spaces/rooms and their locations, users and attendance. The cloud-based application will also be responsible for processing metrics of utilisation and assigning meeting rooms based on numbers, location and other requirements. The API then can be easily consumed using a range of different devices and technologies without exposing the internal architecture.

Once the cloud-based system is up and running; development can begin on creating a browser based user-interface to allow users to login, view and book rooms and then finally showing reports to admin users and displaying room utilisation. The UI could use Google Maps to do route planning to the next event, and also provide push notifications to notify users if a meeting is coming up they are attending. Finally, a mobile app can be developed providing only essential information and user tracking using GPS and/or Bluetooth beacons.

When requesting a room, device information such as location can be used to allocate a room that is best suited using GPS and beacons to find the closest location, number of people, free rooms or resources needed.

### 1.4 Software and Hardware Environment

#### 1.4.1 Server

During initial development, the cloud-based application will be developed on a virtual machine. Once completed, organisations will be able to deploy the project to different cloud services such as Microsoft Azure, Amazon Web Services, and Digital Ocean or with their own infrastructure.

The application will be written in NodeJS and exposed to the internet via a RESTful API behind an authentication layer, while data can be stored using a database.

#### 1.4.2 Dashboard

A dashboard will be created to help users to manage their events, and to provide administrable control over all aspects of the application. Functionality of the dashboard will include reporting



of utilisation, management of assets and timetabling. The dashboard will be accessible through a web browser taking full utilisation of the server API

An additional interface will be created for purpose of use by a tablet device connected outside a meeting room. This interface will be designed with touch in mind and will allow anyone to see upcoming events booked for that room, current events taking place and allow the user to easily book an event in that room for any period in that day.

#### 1.4.3 Mobile Client

A mobile client will be created using Apache Cordova, which allows Mobile apps to be developed with HTML, CSS and JavaScript while targeting multiple platforms. The app will provide push notifications, access point, GPS, and Bluetooth Beacon support on top of the cloud-based RESTful API. For this project, the mobile app will only be targeted for the Android platform due to time constraints and due to limitations within the Apache Cordova framework, a custom background service will be created that runs in the background sending the current logged in user's location data securely to the server. The mobile client will also provide functionality to see upcoming events, create a new event and see event information. Finally, the app will be designed to alert users of meetings they may have.

### 1.5 Success Criteria

The project can be considered a success when the system created is able to manage space within a building for meetings and classes, the ability to track people's attendance of the meetings and generate live reports based on the findings to allow the end user to view building and room utilisation.

#### 1.5.1 Acceptance Tests

User Story	Acceptance Test
Use tracking technology to identify when people attend meetings/events/classrooms.	When a user attends a meeting, the solution will check the user in as an attendee and change their status to attended.
Recording tracking data against rooms, meetings and users in a central server.	All tracking data is stored on a central server that is able to be queried and viewed as a report.
A web interface used by meeting rooms that will display specific information about the meeting room/meetings being held, who booked the room.	An interface is available to be displayed outside a room that displays that room data gained from the server.

The software will run on a centralised server which will allow access from the internet anywhere.	The server will be accessible from any internet connected device.
Tracking performed by a mobile application using GPS, BSSID's and/or eBeacon's.	A mobile app that is able to detect eBeacons/ibeacons, GPS and BSSID's related to meetings/rooms and relay this data to the server.
Allow users to search for, find and book rooms based on their location providing directions where appropriate.	The ability for the solution to recommend the best room possible based on a data provided by the user and the users location. Providing detailed information about the location of the room.
Provide push notifications and reminders to meeting attendees.	The mobile app will notify users via a push notification of any events they are invited to. A notification will also be sent when an event is about to start to remind them of their attendance.

## 1.6 Development Plan

The development plan is seen in the following Gantt chart. The development of the project can be seen to follow three distinct sections; cloud-based development, UI/Browser based development and Mobile based development. A detailed Gantt chart is available at the URL below.

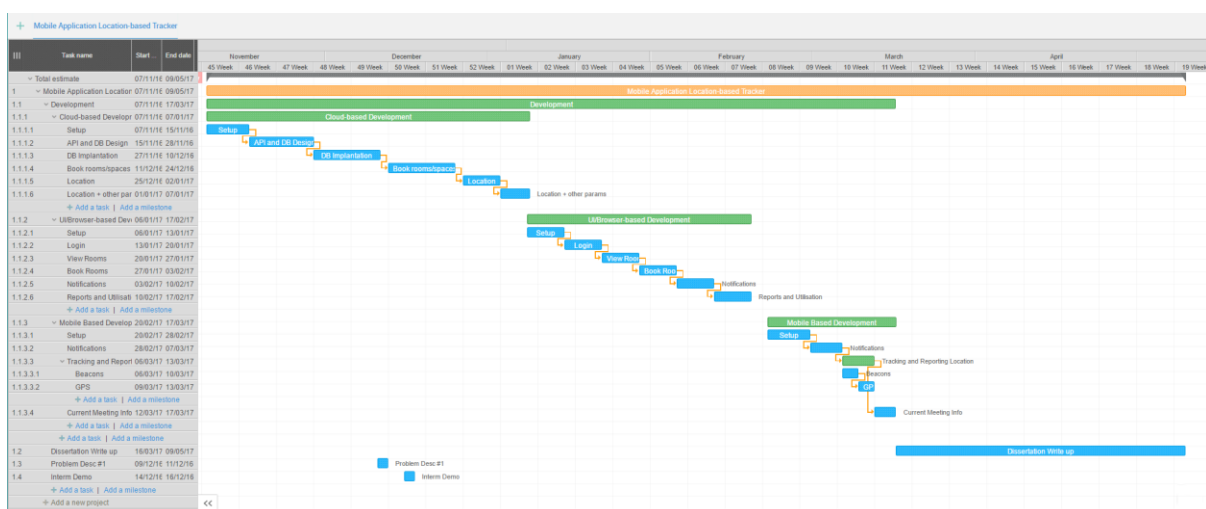


Figure 1: Gantt Chart of Development Lifecycle Available at the URL

<https://app.ganttpro.com/shared/token/a5cdbc49a57d21ab4ab98e412036e1cf03053fc7a8e834844e9f6b925316d65d>

### 1.6.1 Development Software

The project will be developed with the help of the following software packages

Software Package Used	Software Type	Info
Android Studio	IDE	Debugging the Android App as an interface for adb and IDE used for the location awareness background service
Atom	IDE	Text editor with syntax highlighting, ftp and git support
PuTTY	SSH Client	SSH client for accessing the server
Postman	REST Client	REST client for testing the API
Microsoft PowerShell	Terminal	Command line tool used for compiling the mobile application
Google Chrome	Browser	Browser used to access the dashboard web application.

## 2 System Requirements Specification

### 2.1 System Architecture

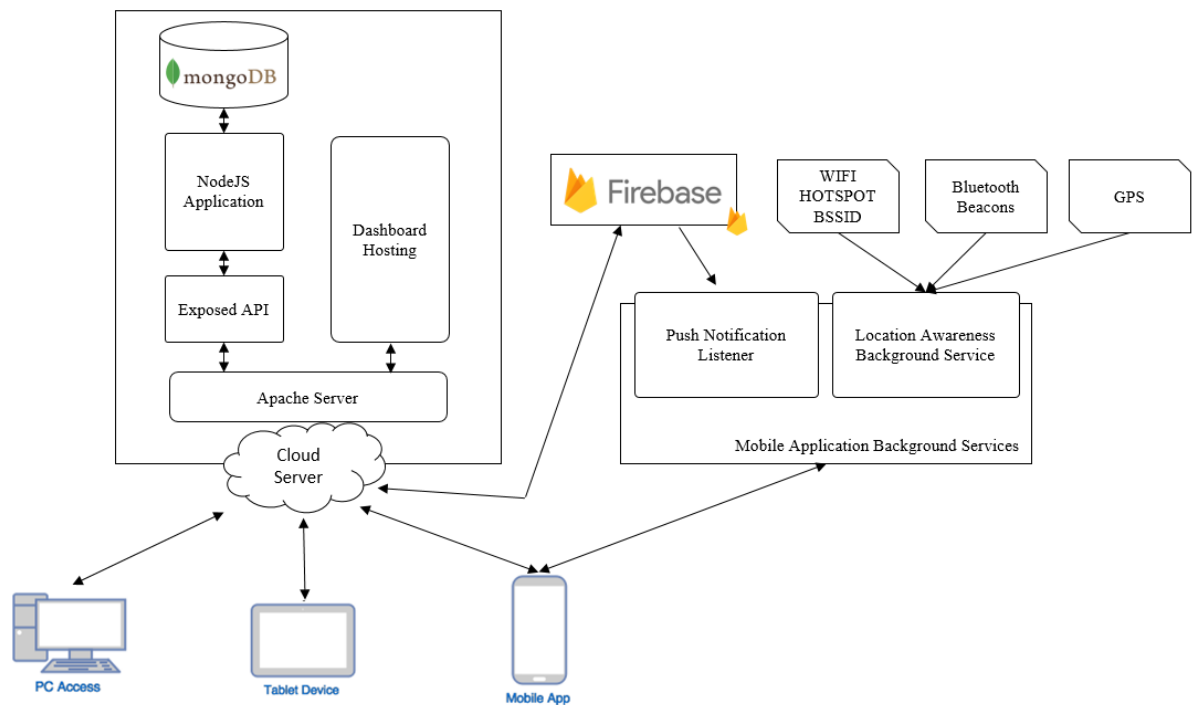


Figure 2: The system architecture detailing a description how different components of the system communicate.

The solution is centralised around a REST web service, exposing access to different objects contained within a MongoDB. The server will also act as a hosting platform for the dashboard web application with all connections to the cloud server protected by encryption using HTTPS. Additionally, there will be a mobile application created that will regularly poll the server with a user's location information, updating the events attendance levels.

## 2.2 Server Overview

The server will provide the main logic for application, with all clients interacting with the server directly using a REST web service. All communication between clients and the server will be presented in the header and body of a HTTP request using the HTTP/1.1 specification [2].

All connections to the web service are protected by requiring a valid username and password that must be passed in the header the request. All requests between the clients and server are encrypted using HTTPS and a basic access authentication header. Using both these methods enforces encryption and verification of every request. Once a user is authenticated, their user data is passed to the requested route and a response is generated.

The server processes each request depending on the route and HTTP verb. For example, sending the request ``GET /users`` will return all users and ``DELETE /users/0001`` would delete the user with ID 0001. Additional routes will process data inputted, query and update the database such as ``GET /poll`` is used by the mobile application awareness background service to poll the user's location. These routes will be detailed further in this document.

### 2.2.1 Server Database

The server will store all data about users, locations, places and events in a database. All access to the database will be encapsulated by the REST calls made by clients keeping access to the database secure.

## 2.3 Mobile Application Overview

The mobile application is a key requirement for the project as it keeps track of users attending scheduled meetings. The mobile application is required to:

1. Poll the server with location data from a custom background service
2. Listen to push notifications from the server to alert them of any new events they are invited to or events that about to start.
3. Give the users a quick and convenient way to book new events, view upcoming events and view details of these upcoming events.

Additionally, the mobile application will allow administrative users access to the tracking data sent to the server such as BSSID's, GPS and Bluetooth beacon's. This data will help build the database at the planning stage of implementation without the use of additional apps.

The mobile application will only start tracking once a user is logged into the application. Their user credentials are stored within the app and cannot be accessed by any other application. Once a user logs out of an app, the user's credentials will be deleted and the background services stopped.

### 2.3.1 Tracking Technologies

A wide range of different technologies have been chosen in this project to match existing infrastructure in buildings and to keep costs down.

GPS (Global Positional Service) is a space-based radio navigation system that works with all mobile phones providing a maximum location accuracy of 10m. This accuracy is very often not the case and objects such as buildings further make GPS more and more inaccurate. GPS is not suitable for indoor tracking.

Bluetooth beacons are low powered devices that advertise data packets within the Bluetooth spectrum such as a URL<sup>1</sup> or UUID<sup>2</sup>. Bluetooth Beacons are becoming more and more popular way of tracking, and many phones now contain Bluetooth modules that can listen to Bluetooth beacons using less power, allowing them to be run in the background of mobile phones without using lots of battery. Assigning a beacon UUID to a location will allow us to identify where the user is and track if they have accessed a meeting.

BSSID is an access point's unique MAC address. Many locations where this project will be implemented should have already available infrastructure such as Wi-Fi hotspots. Tracking a user's location by the BSSID's they interact with is a very cost effective way of achieving the goals of this project without any additional hardware.

The project will enable the availability of choosing one or many of the different tracking technologies depending on the requirements of a location.

---

<sup>1</sup> URL - The address of a World Wide Web page.

<sup>2</sup> UUID – Universally unique identifier used to identify information in computer systems.

## 2.4 Dashboard Overview

The dashboard will allow users to login from any PC with internet access and view their details, view upcoming events and all past events. It will also allow users to create new events and manage any events they've already created. All information about an event is available through the dashboard including a live graphical overview of the attendance at an event.

Admin users will have additional features available to them, giving them full access to add, edit or delete users, places, locations and events. They will also have access to view a locations dashboard and statistics about a particular location.

## 2.5 Location Dashboard Overview

The location dashboard is designed to be placed outside a location, providing stateless access to that location's current events for the day, showing the time and date of each event currently happening. The location dashboard is designed to be touchscreen friendly and to suit a range of different resolutions to cater for a wide range of different tablet screens.

The location dashboard will also provide an easy interface for booking the location for a time that day, without requesting a user's password.

## 2.6 Functional Requirements

### 2.6.1 Server

- Must be accessible on the web
- Verify all user connections match submitted the correct username and password
- Deny access to any connections without the current username or password
- Allow image of users, locations or events to be uploaded.
- Push notifications to a user's mobile device when a new event is added where they are invited to, or when a new event is about to start.
- Return a JSON list of all the events a user has created or been invited to, sorted by most recent to oldest.
- Return a JSON list of all events created on the system sorted by most recent to oldest.
- Return a JSON list of all upcoming events for the current logged in user sorted by next event to the event furthest in the future.
- Return a JSON list of all past events for the current user logged in, sorted by the most recent event to oldest.
- Return a single JSON object of an event by a specified ID displaying detailed information about its location, the place that location resides in and owner.
- Allow a user to create a new event.
- Allow an admin user to delete an event by its ID.

- Allow the currently logged in user to update their status of an event manually to accepted or declined
- Allow an admin user or the owner of an event to directly edit any details of an event saved on the system.
- Return a JSON list of events that the current user is invited to, but hasn't accepted or denied they are going to attend the event.
- Return a JSON list of all locations created on the system with full data about the locations place populated.
- Return a JSON object of a location with full data about the locations place populated.
- Return a JSON list of events that start today at a location by its ID, sorted by events earliest in the day to the evening, showing additional information about an events owner.
- Return a JSON list of locations sorted by how close they are to a set of GPS coordinates and a maximum distance with full data about the locations place populated.
- Allow an admin user to create a new location.
- Allow an admin user to edit a previously added location.
- Return a JSON object of the current authenticated user document except their hashed password.
- Return a JSON object of the API version and author.
- Return an empty JSON object when a favicon is requested.
- Return a JSON list of places, populating details of its parent place.
- Return a JSON object of a place by its ID.
- Allow an admin user to add new place.
- Allow an admin user to delete a previously added place by ID.
- Allow an admin user to edit the details of a previously added place by ID.
- Return a JSON object of the statistics stored about a location by ID including the location document and all events that have ever taken place at that location.
- Allow any user to send their location information to the server where the server will query the database for events at that location where the user is invited, accepted or declined and the event is currently active; set the status of that user to attended.
- Return a JSON list of all users.
- Return a JSON object of a user by their ID.
- Returns a JSON object containing the name and ID of a user when an email is queried.
- Allows an admin user to add a new user.
- Allows an admin user to edit a previously added user by ID.
- Allows an admin user to delete a previously added user by ID.
- Allows a user to update the push notification token stored on their user document.

### 2.6.2 Mobile Application

- Allow a user to login with their credentials
- Display a list of upcoming events.
- Allow a user to add a new event, selecting an event photo, location and the ability to invite users
- Display information on an upcoming event in detail

- Logout, deleting their data from the phone
- Display data sent by the background service to admin users
- View a detailed list of users invited to an event, showing their status
- Allow the logged in user to change their status to an event to accepted or declined.
- Display directions to an events location using Google Maps, or other direction software on the phone (Uber, Lyft etc.)
- Even when the app is not visible, it should keep polling the server background location information.
- The background location service should start-up on boot and stay active until the user logs out or removes the application.

### 2.6.3 Dashboard

- Allow a user to login with their credentials
- Takes a user to the login page if they access any part of the dashboard without valid credentials
- Displays events the user is attending that are currently happening on the dashboard welcome/home screen.
- Displays a list of upcoming events on the welcome/home screen
- Displays a list of past events on the welcome/home screen
- Shows notifications of the amount of events the current logged in user is invited to but has not accepted or denied attendance
- Allows a user to click on a notification and the dashboard will display that event information.
- Show profile information about the current logged in user.
- Allow all users to add a new event
- Allow all users to select a location for a new event which is filtered by their location based on if the user accepts the browser can access their location, otherwise show an unordered list of locations.
- Allow users to search for a location based on any search term (such as services, name, type etc.)
- When adding an event, allow the selection of users from a dropdown menu.
- Allow admin users to view all events as well as their own events.
- Allow users to delete events they created.
- Allow admin users to delete any events.
- Allow users to edit events they created.
- Allow admin users to edit any events.
- Allow admin users to view all users.
- Allow admin users to search for users.
- Allow admin users to add a new user.
- Allow admin users to edit an existing user.
- Allow admin users to delete an existing user.
- Allow admin users to add a new location.
- Allow admin users to search for existing locations.



- When adding a new location, the user should be able to select a place already stored on the system.
- When adding a new location, the user will be able to view a map of the entered GPS coordinates or search for GPS coordinates via entering in an address.
- Allow admin users to view detailed statistics about a location.
- Allow admin users to edit the details of an existing location.
- Allow admin users to delete an existing location.
- Allow admin users to access the dashboard screen about a location.
- Allow admin users to add a new place.
- When adding a new place, allow a user to add a parent place, or no parent place if applicable.
- Allow admin users to delete an existing place.
- Allow admin users to edit an existing place.
- Allow admin users to search for places.

#### 2.6.4 Location Dashboard

- Display current events happening at a location in a list.
- Allow a user to add a new event by using just their email address.
- Display the current location on screen.
- Display visibly if there is a current event at the location.
- Update the screen live as events change throughout the day.

### 2.7 Non-Functional Requirements

#### 2.7.1 Server

- The application must be accessible from the web.
- Data between clients and the server must be encrypted at all times as passwords could be vulnerable.
- User passwords must be hashed at all times.
- The server should be easily scalable and easily customisable.
- Any requests must always return data when accessed.
- Error messages should be displayed when a user is not authenticated properly.
- Error messages should be displayed if there was an error with a request with the appropriate HTTP status code.
- There must be little to no downtime to the application, even when an internal error has occurred. The application needs to restart itself to go back online.

#### 2.7.2 Mobile Application

- User credentials should be stored securely on the device.
- The information must be displayed to the user in a user friendly, accessible way.
- Buttons must be large and easy to tap with their finger.
- Error messages should clearly be displayed from the if appropriate.
- All pages should carry the same design aspects.
- Validation should be done before submitting.

### 2.7.3 Dashboard

- The information must be displayed to the user in a user friendly, accessible way.
- Error messages should clearly be displayed from the if appropriate.
- All pages should carry the same design aspects.
- Validation should be done before submitting forms.
- Only options available to a user group (such as admin) should be visible. i.e. If a user cannot edit an event, then the manage button should not appear.
- The event page should update momentarily when open to show any changes to the event such as attendance.
- Event stats should be clearly visible in charts when possible.
- Location statistics must be easily understandable with a wide range of use of graphs and charts.
- The dashboard must be located behind a HTTPS server, ideally the same server as the API.
- When a user logs out, all data should be deleted from the browsers cache.

### 2.7.4 Location Dashboard

- The information must be displayed to the user in a user friendly, accessible way.
- Buttons must be large and easy to tap with their finger.
- Error messages should clearly be displayed from the if appropriate.
- All pages should carry the same design aspects.
- Validation should be done before submitting.
- Time, Date, application name and location should be displayed at the top of the screen.
- Prompts should follow an easy flow as access will be with a tablet touchscreen device.
- The display should look similar of a range of different display sizes.
- Any controls must be touch screen friendly and accessible using an onscreen keyboard.

## 2.8 Hardware and Software Requirements

The project is built around a cloud based design with a central RESTful API server running on a remote Ubuntu machine also providing the hosting for the dashboard although the project could be run locally with the correct setup.

The mobile application will be compiled into an Android app, this android app will display information requested from the server, while also allowing the user to create new events and update the server the with the user's location. Any users on the system currently will need to have an android application but with more time and testing, the application could be suited to compile on other mobile operating systems supported by Apache Cordova.



Figure 3: Mobile Operating Systems currently supported by Apache Cordova

Due to the custom service needed to track users' location in the background, custom services will need to be wrote for each operating system in their respected languages, and due to the time constraints, this would not be possible for the project.

The dashboard and location dashboard are web applications and in theory can be run using any web browser with access to port 445 to the server that supports ECMAScript 6, HTML and CSS. Desktop browsers that should be able to support the application include Google Chrome, Safari, Firefox, Opera and Microsoft Edge. [3]

## 2.9 Dashboard Application Programming Interface (API)

All API requests require authentication by a user. Users with admin privileges will have access to greater information and additional calls that will be detailed in the documentation.

### 2.9.1 Authentication

Authentication is handled using "Basic Authentication" header type. All requests shall be sent with this header or the request will be denied.

The basic authentication header is structures with the word "Basic " followed by a base64 encoding of "username:password". For example, the header would be formatted as follows.

```
Authorization: Basic ZDNhbi4353FdsfdFAaG90bWFpbC5jDf43SDWFAadvcmQ=
```

This is **NOT** encryption or a hashing function and it is expected that this string is stored on a secure device. All access the API is provided through SSL/TLS certificate that ensures all packets sent between the server and client are secured and encrypted. The only point of failure in a security standpoint is the token being stolen from the user's machine itself.

As there is no explicit login or logout function so users with the Authorization token have access until the password is changed. Logout should be implemented by providing a function that simply deletes the saved user Authorization token.

### 2.9.2 Generic Routes

Verb	Route	Returns
<b>GET</b>	/	{ "version":1,"author":"Dean Meehan" }
<b>GET</b>	/whoami	A user object containing basic user information
		As there is no login function, there is a route available that will return basic information about a user as long as Authentication is correct.
<b>GET</b>	/favicon.ico	Returns a favicon for the project

### 2.9.3 User Routes

Verb	Route	Returns
<b>GET</b>	/user	Returns the current user's object
<b>GET</b>	/user/all	Returns all users. (Restricted to admin users)
<b>GET</b>	/user/:id	Get's a users detailed from a user id. (Restricted to admin users)
<b>POST</b>	/user/email	Returns the ID and Name of a user by email. (Used in searches)
<b>POST</b>	/user	Creates a new user (Restricted to admin users)
<b>DELETE</b>	/user/:id	Deletes a user by ID (Restricted to admin users)
<b>PUT</b>	/user	Passing in a User object will edit the user details that have changed (Restricted to admin users)
<b>PUT</b>	/user/token/:token	Passing in a token updates the saved Firebase token for the user for push notifications

### 2.9.4 Event Routes

Verb	Route	Returns
<b>GET</b>	/event	Returns all the current user's events sorted with newest first
<b>GET</b>	/event/all	Returns all events sorted with newest first (Restricted to admin users)
<b>GET</b>	/event/upcoming	Returns all the current user's events from today on, sorted with next first
<b>GET</b>	/event/previous	Returns all the current user's past events, sorted with most recent first

<b>GET</b>	/event/:id	Returns all data about an event with the id :id (Restricted to admin users)
<b>POST</b>	/event	Creates a new event
<b>DELETE</b>	/event/:id	Deletes an event with the id :id (Users can delete their own events, while admins can delete all events)
<b>PUT</b>	/event/:id/:status	Changes the status for the current logged in user for the event by id to the status status
<b>PUT</b>	/event/:id	Edits an event with the id :id with the object provided.
<b>GET</b>	/notifications	Returns an object detailing events the current logged in user needs to accept or decline (Events stuck in the invited status)

### 2.9.5 Location Routes

The location object holds more specific information about a lecture hall, classroom, or room within a place. They should be specified with basic details plus location specific information such as floor, max people, services available, beaconid, gps coordinates and access point BSSID. These are used for tracking attendance and utilization.

<b>Verb</b>	<b>Route</b>	<b>Returns</b>
<b>GET</b>	/location	Returns all locations
<b>GET</b>	/location/:id	Returns all data stored about a location with id :id
<b>GET</b>	/location/:id/events	Returns all events for the location of :id for today. (Used by Location Dashboard)
<b>GET</b>	/location/near/:x/:y/:dist	Returns all locations in order of distance to a geolocation :x, :y limited to a distance in meters specified in :distance
<b>POST</b>	/location	Adds a new location (Restricted to admin users)
<b>DELETE</b>	/location/:id	Deletes the location with :id (Restricted to admin users)
<b>PUT</b>	/location	Edits a location, the location is specified from the location object PUT's id field (Restricted to admin users)

### 2.9.6 Place Routes

The place object is used by locations to specify where they are. They contain basic location information such as address and also can contain a parent place. The idea behind this will allow buildings and groups of buildings to be linked together such as "EEECS Building, Ashby Building, Students Union can all have the parent place Queens University Belfast. The EEECS Building can then be the master place of 'EEECS Stranmillis Rd' and 'EEECS Elmwood'".

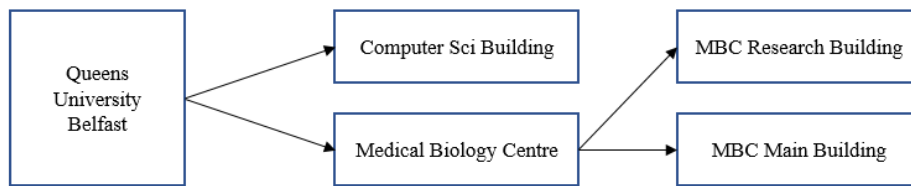


Figure 4: Relationship between places and their parent places.

Verb	Route	Returns
GET	/place	Returns all places
GET	/place/:id	Returns all data stored about a place with ID :id
POST	/place	Adds a new place (Restricted to admin users)
DELETE	/place/:id	Deletes the place with an ID :id (Restricted to admin users)
PUT	/place	Edits a place, the place is specified from the place object PUT's id field (Restricted to admin users)

### 2.9.7 Statistics Route

The statistics route is used to provide statistics about the usage of locations.

Verb	Route	Returns
GET	/stats/location/:id	Returns statistical data about a locations usage over time including all events held at that location, the average usage etc. This data is used within the dashboard. (Restricted to admin users)

### 2.9.8 Polling Route

The polling route checks location data sent from the mobile application. If the location data matches location data from an event location the user is invited to it checks them attended. The location data is not kept ensuring the privacy of the projects users are kept.

Verb	Route	Returns
POST	/poll	The interface where app can poll a uses location data such as GPS, Beacons and BSSID's from access points. Polling will update a users attendance at any events currently happening at any matching locations.

## 3 Design

### 3.1 System Architecture

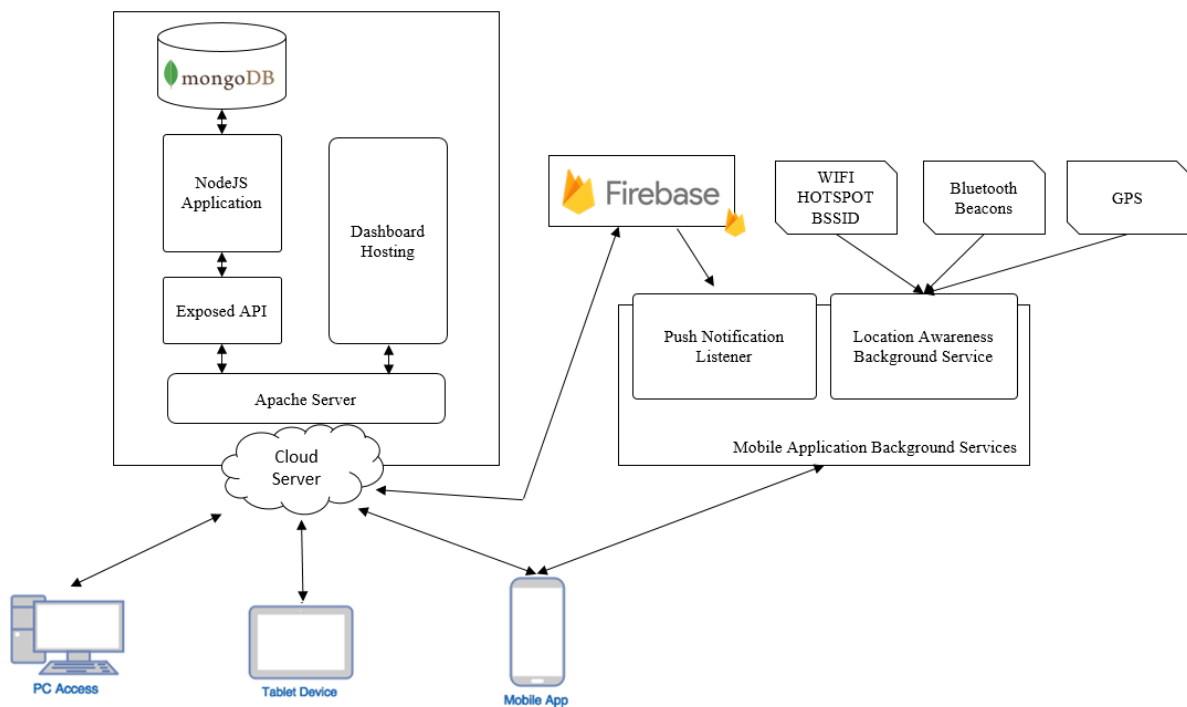


Figure 5 The system architecture detailing a description how different components of the system communicate.

The system architecture can be broken into four parts, the centralised server, PC Dashboard, Location (Tablet) Dashboard and Mobile Application.

### 3.1.1 Cloud Server

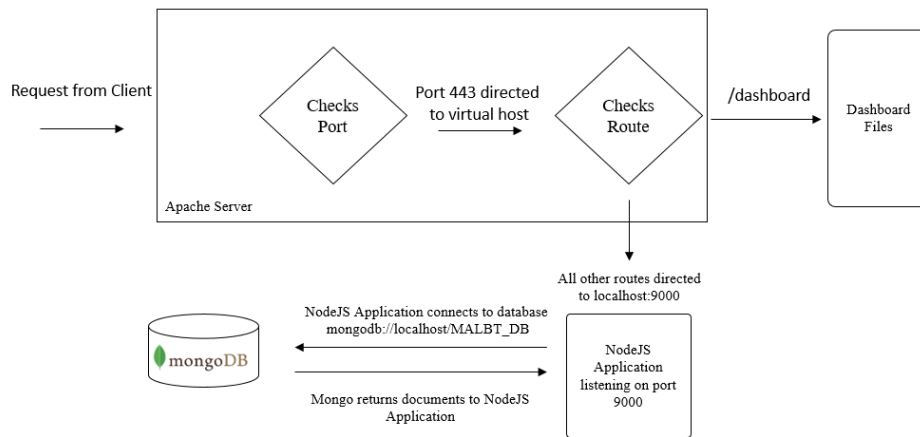


Figure 6: Detailed System Architecture detailing how the internals of the Cloud Server work with incoming connections.

The cloud server is designed to run on a Linux machine, supporting applications such as Apache, NodeJS and Mongo. It uses Apache to handle virtual hosts (allowing many applications to run on one server) and providing the SSL/TLS server security. The server then listens to all requests and depending on the port and route, the request is sent to the internal application (<https://cloud.dean.technology:443/whoami> is directed to localhost:9000). Node then will connect to the mongo database to access or update data and respond with a JSON object. If a connection is directed to <https://cloud.dean.technology:443/dashboard>, apache directs the connection to its own web service and responds with the web application assets.

### 3.1.2 Mobile Application

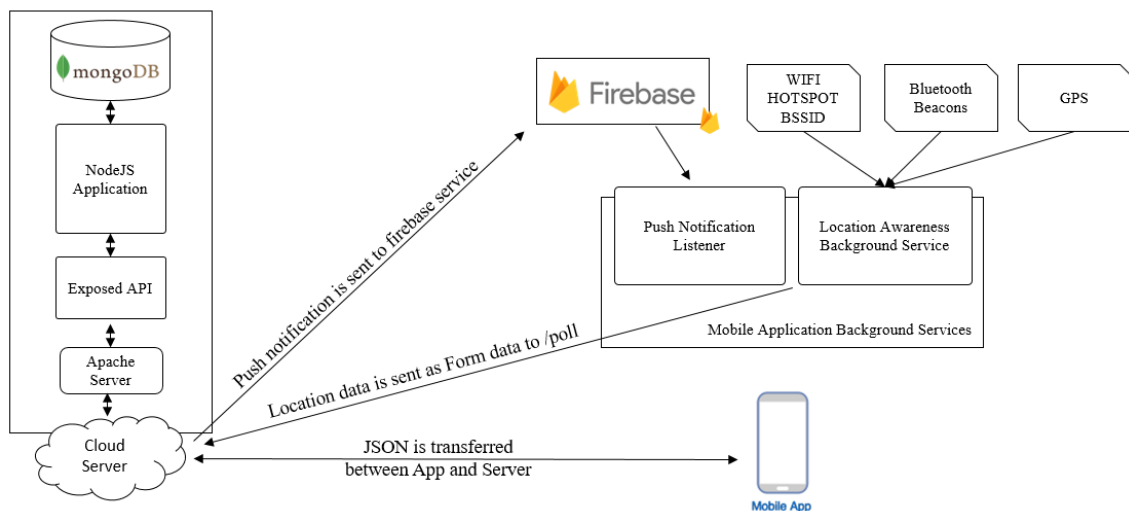


Figure 7: Detailed Architecture how the mobile app and services interact with the server

The mobile application connects to the cloud server to access event information as well as poll location data to the NodeJS application. The Location Awareness Background service requests



access point, GPS, and Bluetooth beacon data from the Android operating system and periodically every 5 minutes sends this data to the cloud server which is able to then process if a user should be in an event and set their status to attended.

The mobile app will also have a push notification listener that listens in the background for firebase notifications. Firebase is a service run by Google for providing push notifications. If the cloud server wants to send a cloud notification immediately it can call the firebase service with a firebase UUID sent to the server on start-up from the application that attaches the user to their mobile phone. The service is then able to prompt the user with a notification. The server should also be able to schedule this notification to happen at a later time using cron services on the server.

### 3.1.3 Dashboard

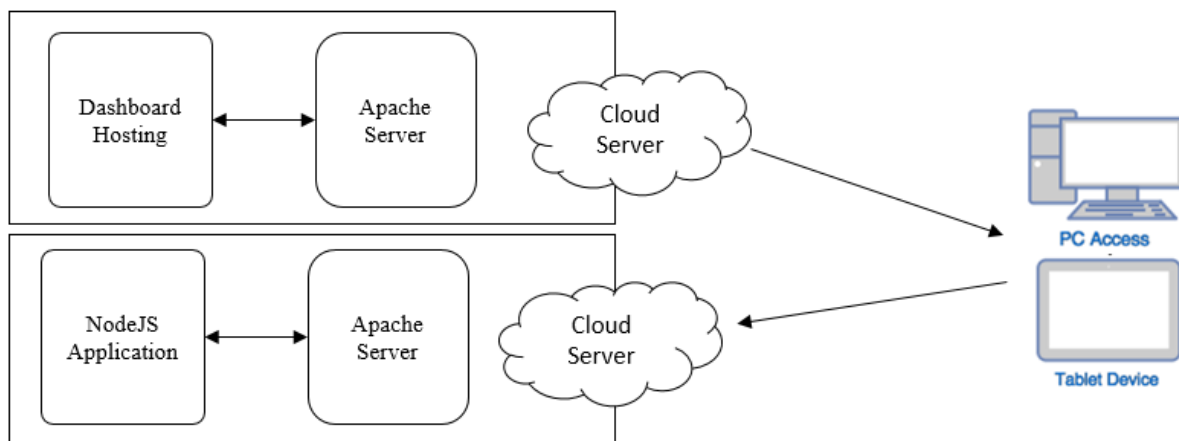


Figure 8: Detailed Architecture of the interaction between Hosting, Client and NodeJS Application

The dashboards are hosted HTML files served by an Apache server. When the user accesses the dashboard, the browser downloads the web application onto their computer and then the JavaScript within the web app requests additional data from the NodeJS service which is received as a JSON object that can be displayed within the browser. This allows data within the application to be updated without reloading the whole page while keeping data and presentation layers separate, modularising the design.

## 3.2 Database Design

The database engine selected for the project was MongoDB, an open-sourced database that stores data in a JSON-like document that can vary in structure. This was chosen due to the many advantages over a traditional relational database design that suit this project such as:

- **Document-based data modal** – As the data modal closely matches the JSON format that is output from the API. Data is already in the correct format for transmission and in the correct format to be processed by JavaScript clients.
- **Deep query ability** - With the deep-query ability being nearly as powerful as SQL when the application needs to query larger sets of data such as building reports on building utilisation can be done within one request.
- **No concrete schema** - Due to not having a fixed schema, the application is able to store large lists such as location services or event attendees within the location or event document without the need for complicated joins. This also allows for additional features such as an event message board to be developed when needed or required with minimum overhead.

MongoDB also supports horizontal scaling by default with sharding, allowing documents to be stored across multiple machines as the project grows. This is not done for this project, but could be considered once in production.

### 3.2.1 Database Security

To ensure the systems users are save, any passwords saved in the database will be hashed and salted, protecting passwords if there is a system breach. All communication is encrypted with HTTPS so the password will not be in plaintext from the user's device to the server database.

## 3.3 Schema Design

MongoDB can embed multiple documents, or lists of other documents as well as a range of different properties called schemas.

### 3.3.1 Event Schema

Name	Type	Properties
<b>name</b>	String	required
<b>description</b>	String	
<b>owner</b>	ObjectID of type User	required
<b>image</b>	String	
<b>type</b>	ENUM "meeting", "event", "party", "conference", "lecture", "other"	required
<b>location</b>	ObjectID of type Location	required
<b>attendees</b>	List of Objects containing	

	User: ObjectID of type User Status: ENUM "invited", "accepted", "declined", "attended"	
<b>starts_at</b>	Date	required
<b>ends_at</b>	Date	required
<b>addedby</b>	ObjectID of type User	required, auto populated
<b>created_at</b>	Date	required, auto populated
<b>updated_at</b>	Date	required, auto populated

### 3.3.2 Location Schema

Name	Type	Properties
<b>name</b>	String	required
<b>description</b>	String	
<b>type</b>	String	required
<b>floor</b>	Number	
<b>image</b>	String	
<b>max_people</b>	Number	
<b>services</b>	List of Object containing Name: String Description: String	
<b>place</b>	ObjectID of type Place	required
<b>beacon</b>	String	
<b>access_point</b>	String	
<b>gps</b>	Array	
<b>addedby</b>	Object of type User	required, auto populated
<b>created_at</b>	Date	required, auto populated
<b>updated_at</b>	Date	required, auto populated

### 3.3.3 Place Schema

Name	Type	Properties
<b>name</b>	String	required
<b>description</b>	String	
<b>address</b>	required	
<b>Address.street</b>	String	required
<b>Address.city</b>	String	required
<b>Address.postcode</b>	String	required
<b>Address.country</b>	String	required
<b>parentPlace</b>	ObjectID of type Place	
<b>addedby</b>	ObjectID of type User	required, auto populated
<b>created_at</b>	Date	required, auto populated
<b>updated_at</b>	Date	required, auto populated

### 3.3.4 User Schema

Name	Type	Properties
<b>name</b>	String	required
<b>email</b>	String	required
<b>password</b>	String	required
<b>admin</b>	Boolean	
<b>location</b>	String	
<b>image</b>	String	
<b>token</b>	String	
<b>addedby</b>	ObjectID of type User	required, auto populated
<b>created_at</b>	Date	required, auto populated
<b>updated_at</b>	Date	required, auto populated

## 3.4 User Interface Design

The project's user interface tries to follow some of the core design principles of UI design such as simplicity, consistency and feedback. Making the experience of using the mobile application and dashboard easy and enjoyable ensure that users want to use the system, in turn allowing the system to keep an up-to-date modal of building utilisation and usage.

Although the mobile application and dashboard require very different approaches when it comes to design, the project aims to keep the look and feel as similar as possible incorporating similar design aspects such as colours and logos to both.

### 3.4.1 Mobile Application

The mobile applications design is aimed to make the experience of adding new events and viewing upcoming events as easy as possible.

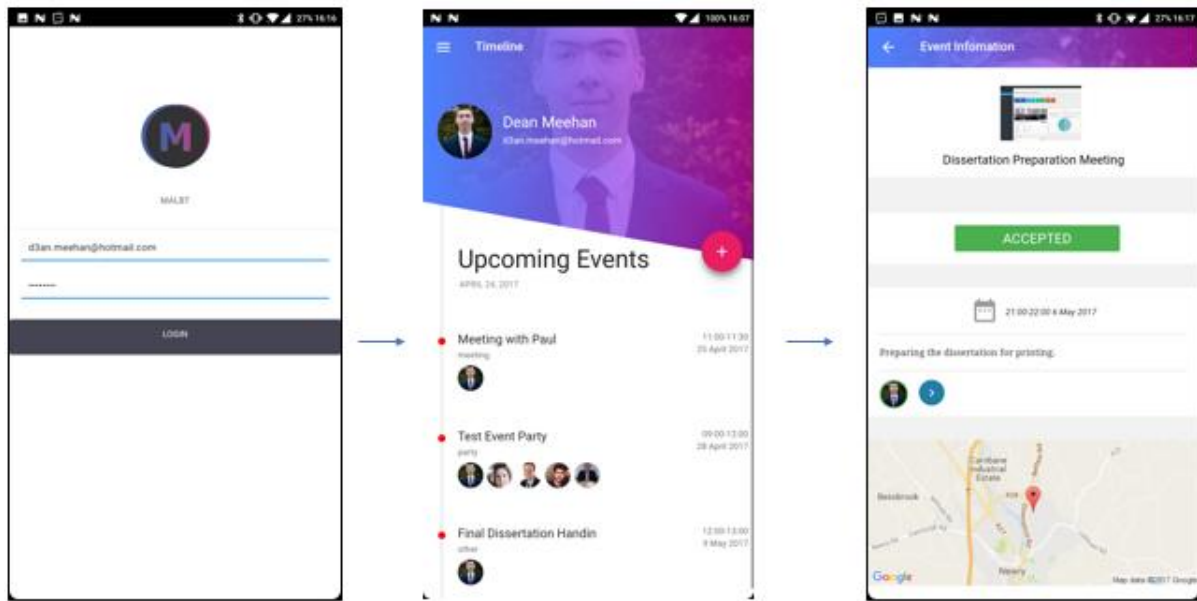


Figure 9: Design of Mobile App showing interaction between logging in and viewing information about an event

When logging in the user will be asked for their username and password which will be verified for syntax before sending to the server. The request gets sent to ``/whoami`` and if the server returns with the HTTP status ``401: Unauthorised`` an error message appears as seen in Figure 11: Error message showing the username and password were incorrect., otherwise the user will be taken to the Upcoming Events page.

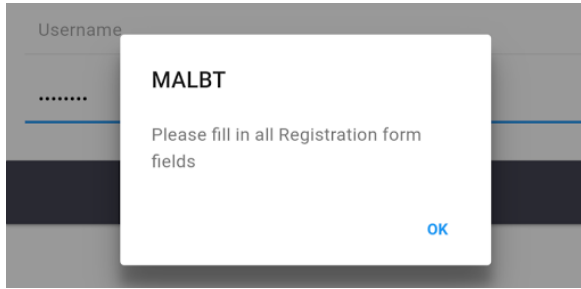


Figure 10: Error Message showing all fields where not filled out.

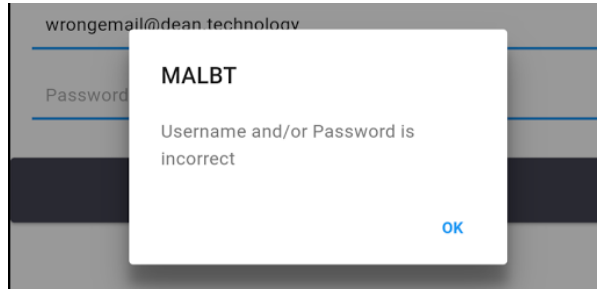


Figure 11: Error message showing the username and password were incorrect.

### 3.4.2 Event Information

The event information page displays information about an event and to do various actions regarding the event such as allowing a user to accept or decline the invitation.

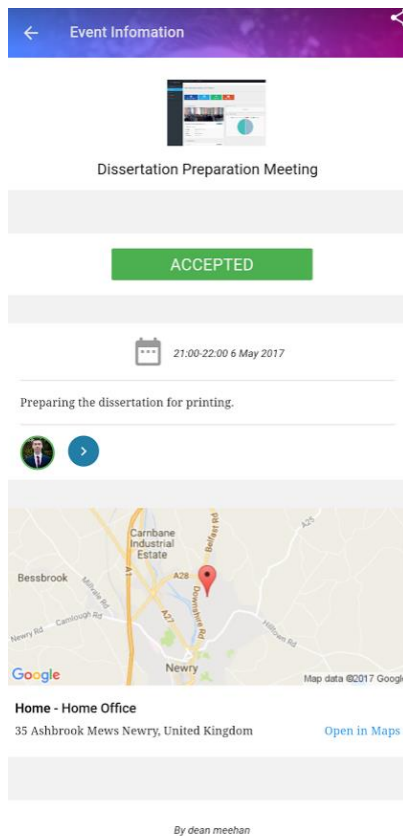


Figure 13: Design of the event information page

The button accepted will display “Please select an option” when the user has just been invited to an event and is updated when the user either selects an option, or attends an event.

Clicking on the blue arrow will display a list of users and status for the event (Appendix 3:List of Event Attendees).

When a user has accepted, or declined they are going to attend a meeting, a coloured ring around their profile photo. coloured green or red.




Figure 12: User List

Clicking on either the map or the text “Open in Maps” will take the user to a dialog screen where they can select an application to open directions that will bring them to the event. This has been tested with Uber and Google Maps, where directions or a taxi can be ordered respectively.

### 3.4.3 Timeline

The timeline page displays the currently logged in user, their photo, an add event button, the date and a list of upcoming events. Clicking on an upcoming event will open the event information page while clicking on the red “+” will bring the user to the event add page.

The design of the page follows a modern unique design keeping with the blue and pink colour scheme of the app. The page was designed to be simple and functional without including too much information so users can quickly see an overview of their next events.

Clicking on the menu icon  will slide over a menu (Appendix 5: Sliding Menu) allowing the user to return to this menu, create a new event, and logout. If the user is an admin user and additional option is shown that will allow the user to enter the track information screen.

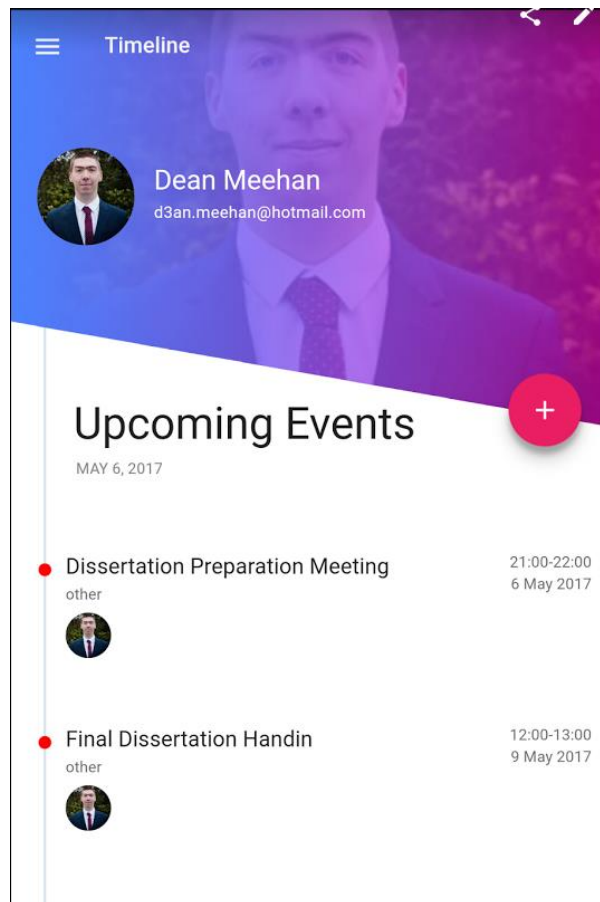


Figure 14: Screenshot of Android Timeline

### 3.4.4 Tracking Information

This screen was created for a purely technical standpoint of allowing admin users quick access to GPS, Beacon and BSSID information. This allows them to poll for information about the location they are in without having to collect the data using additional applications or tools.

As seen in Appendix 4: Tracking Information Screen, the design is simple and shows the current GPS location, WIFI BSSID access points nearby, Bluetooth Beacons nearby and whether the background service is active or not. There is a button to reset the view which will update the information on screen without having to exit the screen and re-enter.

### 3.4.5 New Event

The new event screen allows a user to add a new event. It's simply a form that will ask the user details about the event such as:

- Name
- Description
- Type
- Start Date/Time
- End Date/Time
- Event Photo
- Event Location
- Attendees

The form is designed to be intuitive and easy to use, with “Start Time/Date” and “Event Time/Date” displaying a visual date selector and “Select Event Photo” displaying the native photo viewer.

When selecting a location, there is a location selection screen that displays additional information about the location such as its photo, max amount of people, address and image. See Appendix 6:Select Location Screen

When submitting the form by clicking “Add Event” the form is checked for any missing data and then sent to the server. If successful, a message will appear informing the user of the addition and bring the user back to the Timeline, otherwise an error message will appear allowing the user to edit their selection.

New Event

Name  
New event

Description  
This is the event description

Event Type  
lecture

Start Time/Date  
05/06/2017, 8:59 PM

End Time/Date  
05/06/2017, 9:59 PM

SELECT EVENT PHOTO

CHOOSE EVENT LOCATION

Event Location Selected: Home Office

INVITE USERS

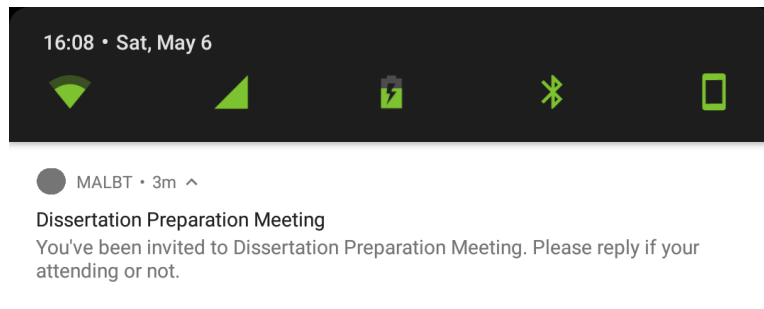
Dean Meelhan  
Carl Meehan  
Rachel Shields  
Joe Patterson  
Paul McCabe

ADD EVENT

Figure 15: New Event Screen



### 3.4.6 Mobile Application Push notifications

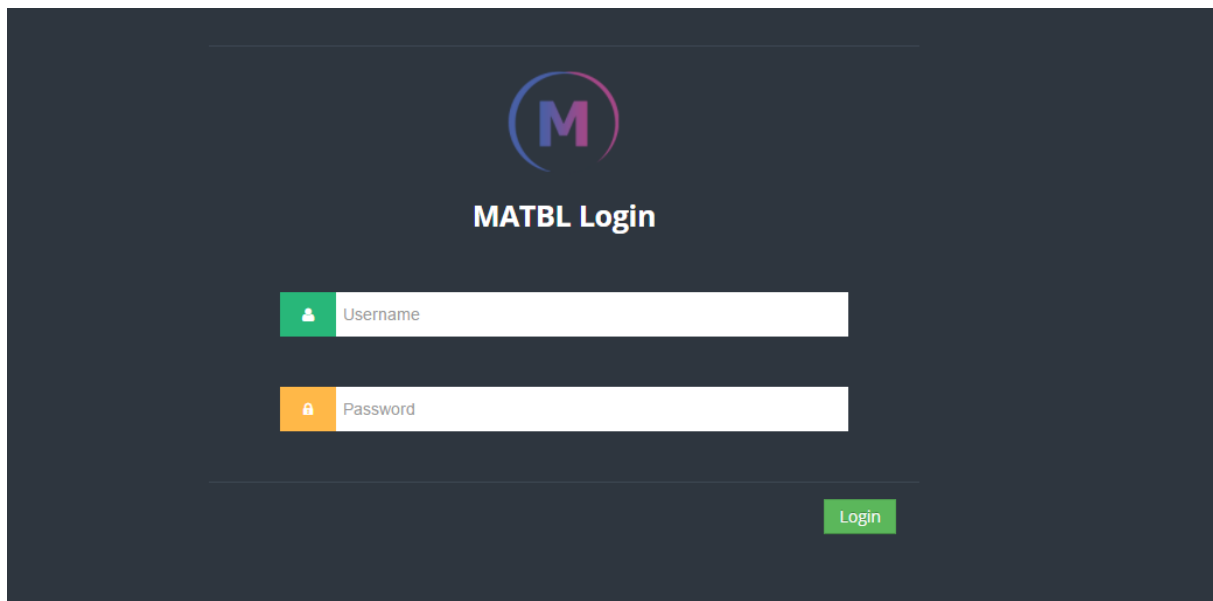


*Figure 16: Mobile Application Push Notification*

When a user is invited to a new event or an event is about to start, a push notification will be sent to the users' phone. The notification displays further instructions such as bringing the user to the event page on their app, or opening up navigation to an event location.

### 3.4.7 Dashboard Login

When accessing any page of the dashboard, the user will be presented with a login screen if they have not already logged in. The login screen is designed to be simple with a username field, password field and a login button.



*Figure 17: Dashboard Login Screen*

When submitting the username and password, the details are sent to the cloud server to verify the user. The request gets sent to `/whoami` and if the server returns with the HTTP status `401: Unauthorised` an error message appears otherwise the user is taken to the dashboard homepage.

Username and/or Password is incorrect

Figure 18: Error message detailing a username and password is incorrect.

Validation is also carried out on the username and password fields before sending to check for any formatting errors, or if the user didn't enter anything into the fields. This is to save wait time before an inevitable error and to save bandwidth.

Username and/or Password is empty

Figure 19: Error message detailing when username or password not imputed

### 3.4.8 Dashboard Home

The dashboard home is designed to give the user a quick overview of current events taking place, upcoming events and recent events that have just passed. The design of the page changes depends on whether the user is an admin user or a normal user; giving admin users additional options within the navigation pane to view users, locations and places.

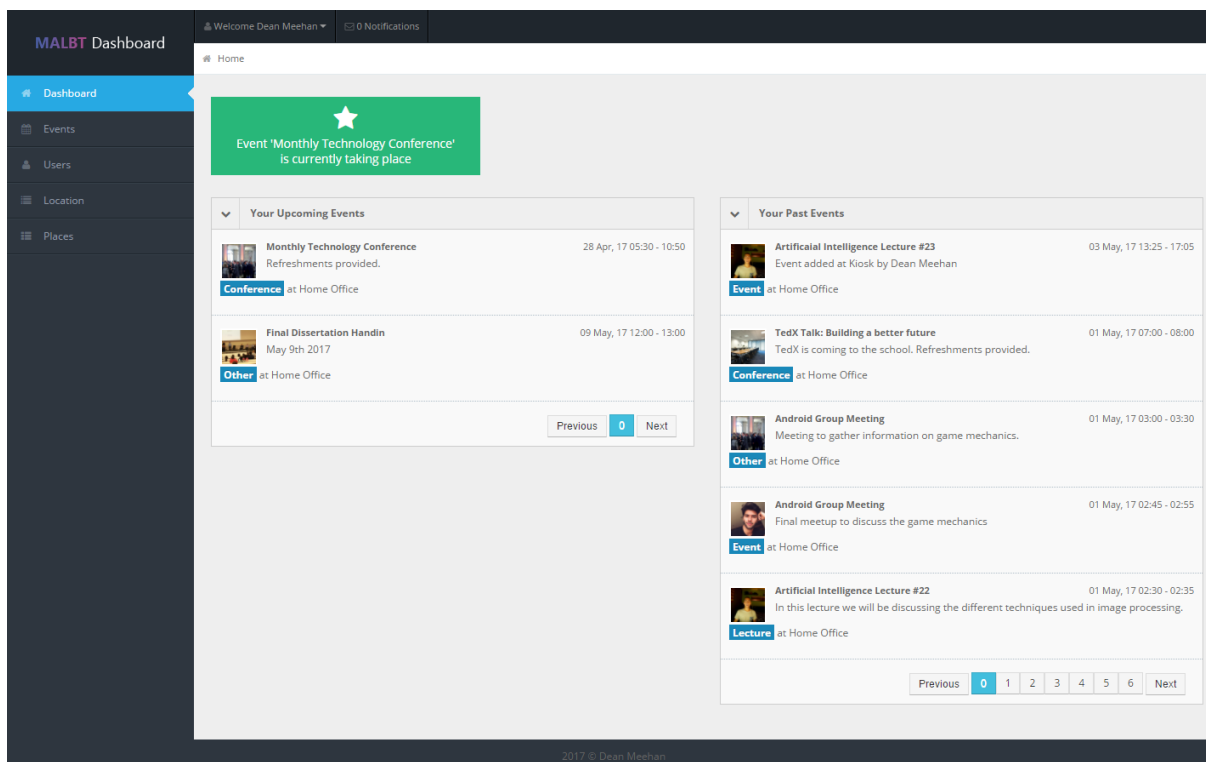


Figure 20: Dashboard Homepage Design of an admin user showing different elements and an ongoing event

The dashboard contains the same design of sidebar and heading through the application. This consistency can be seen though the colour scheme used through the interface and MABL T logo.

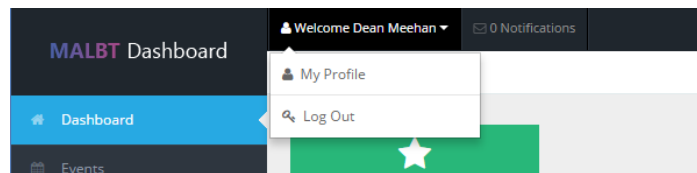


Figure 21: Screenshot of application showing dropdown and sidebar

The top bar clearly indicates the logged in user's name and contains a caret to indicate that a user clicking on their username will open an options menu. This menu allows the logged in user to view more information stored about them via "My Profile" or logout of the system by clicking "Log Out".

On the left sidebar, the current page (or parent page) is highlighted in a light blue colour to indicate clearly the users' position within the application. This sidebar also acts as a navigation bar to take the user to different sections within the application.

When a users' attention is needed, within the navigation bar there is a notifications dropdown that displays in a bright red coloured box, the amount of notifications for the user to gain their attention. Clicking on a notification takes the user to the page of the notification and clears the notification from the system.

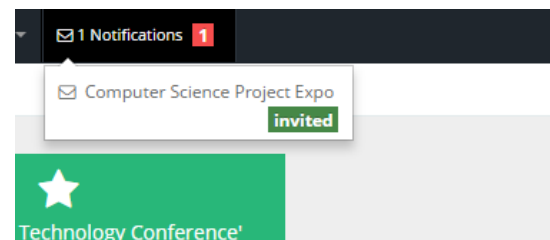


Figure 22: Screenshot indicating notifications

The large green box in Figure 20: Dashboard Homepage Design of an admin user showing different elements and an ongoing event) indicates there is an event currently active. The application allows for a user to be invited to multiple events at the same time. Clicking on this item takes the user to the event details screen detailing more information about the event. When there are multiple events active for the same user, they will appear horizontally beside each other.

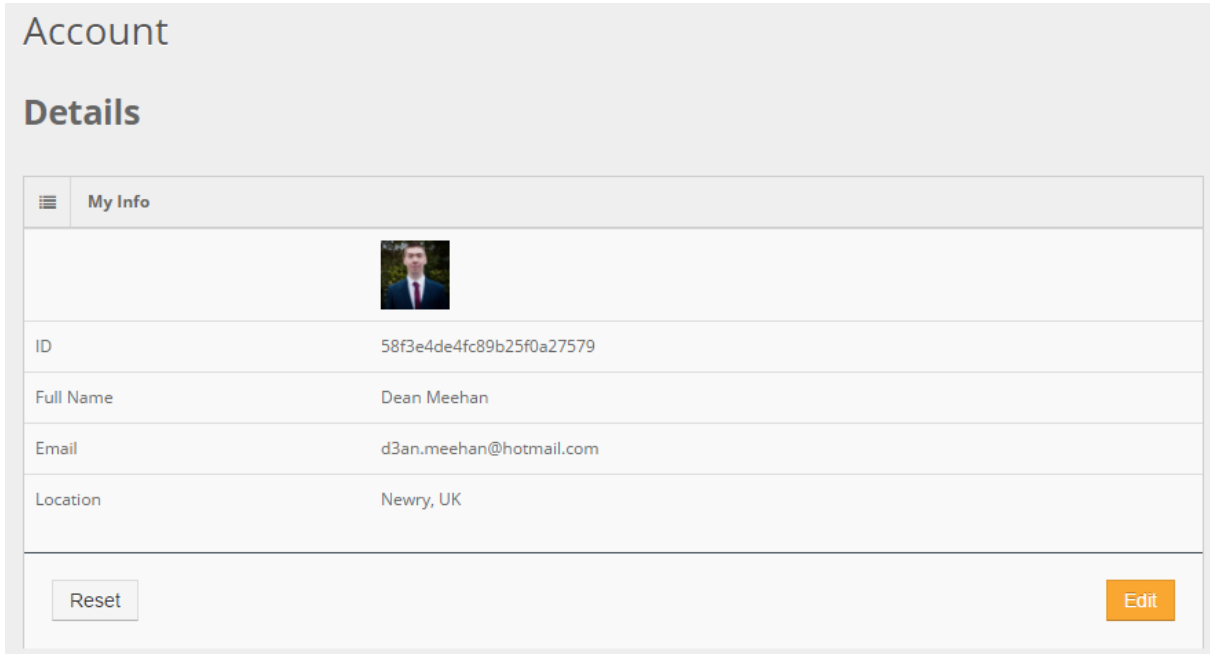
A design aspect carried throughout the application is the usage of breadcrumbs that display an indication of the route the user took to get them to the page they are currently on. All pages within the application can be accessed through 3 or less clicks helping with simplicity. Clicking on any of the breadcrumbs will take the use to that page.



Figure 23: Breadcrumbs of Event Information of Event with ID

### 3.4.9 User Profile

The user profile displays basic information about a user displaying the data stored within the system. This page allows all users to view their information and edit them appropriately if their email or location change and/or they want to change their password.



The screenshot shows a web interface for 'Account Details'. At the top, there's a header 'Account' and a sub-header 'Details'. Below this is a tabbed interface with 'My Info' selected. The main content area displays a profile picture of a man in a suit, followed by a table of personal information:

ID	58f3e4de4fc89b25f0a27579
Full Name	Dean Meehan
Email	d3an.meehan@hotmail.com
Location	Newry, UK

At the bottom of the form, there are two buttons: 'Reset' (light grey) and 'Edit' (orange).

Figure 24: Account Details page of user Dean Meehan

### 3.4.10 Places

The places screen (See Figure 27: Screenshot of places screen) is available to administrative users to manage the places used within the application. It follows the same design pattern used for locations, users and events by displaying a simple table displaying relevant information such as name, description and options. Within the options section there is a dropdown that gives further options to delete or edit the place and a button that takes the user to the add place screen.

The add place button is coloured green to indicate that something is going to be added, using the same shade of green keeping consistency throughout the application.

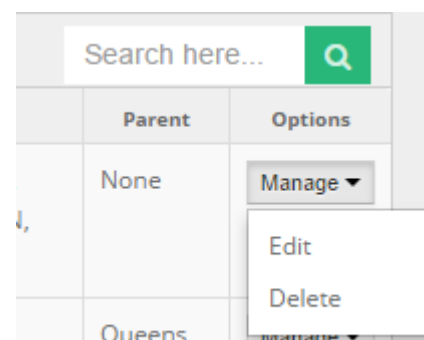


Figure 25: Edit and delete buttons

Additionally, on all management screens such as users, places, events and location, the application includes a search bar that searches through all fields that enables the list of items to be filtered making finding the correct object simpler.

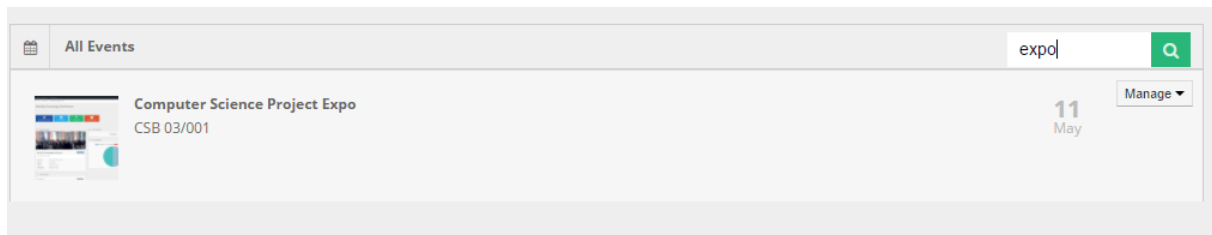


Figure 26: Event Filtering by Search Bar

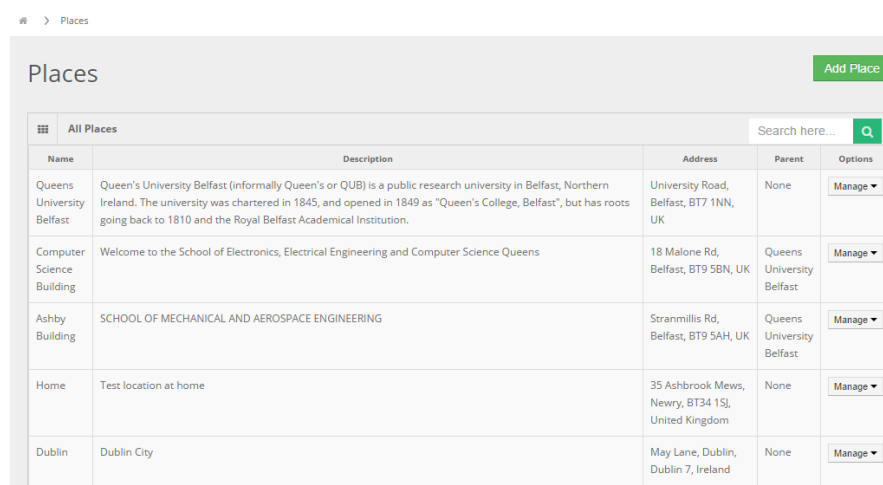


Figure 27: Screenshot of places screen

### 3.4.11 Add Place

The design of add place is a simple form, asking the user for details about the place they are adding such as Name, Description, Street, City, Postcode and Country. The place schema also supports adding a parent place.

When clicking parent place, a modal will appear with a sortable list of existing places on the system.

Figure 28: Screenshot of add place form

#### 3.4.11.1 Select Place Modal

The add place modal is used when selecting parent places for locations and places through the add location and place form. The design incorporates a search bar, a table of basic information with a button to select this place clearly visible on each row.

There is also an option to select no parent place. When clicking any of these buttons the modal will close. This option is not available when adding locations as a location is required to have a parent place.

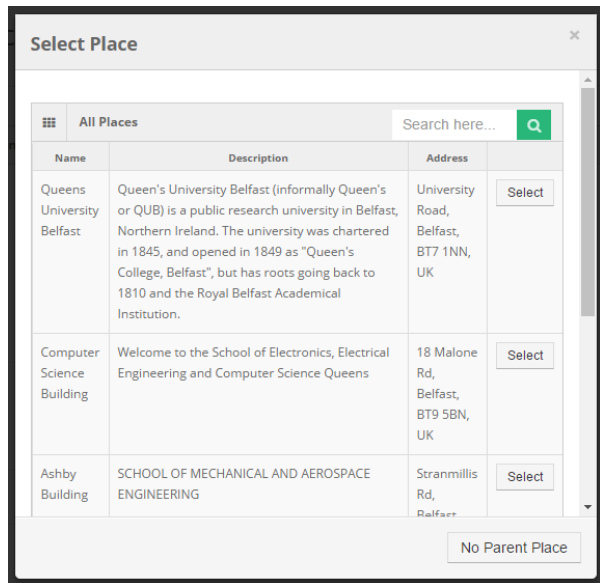


Figure 29: Select Place Modal

#### 3.4.12 Edit Place

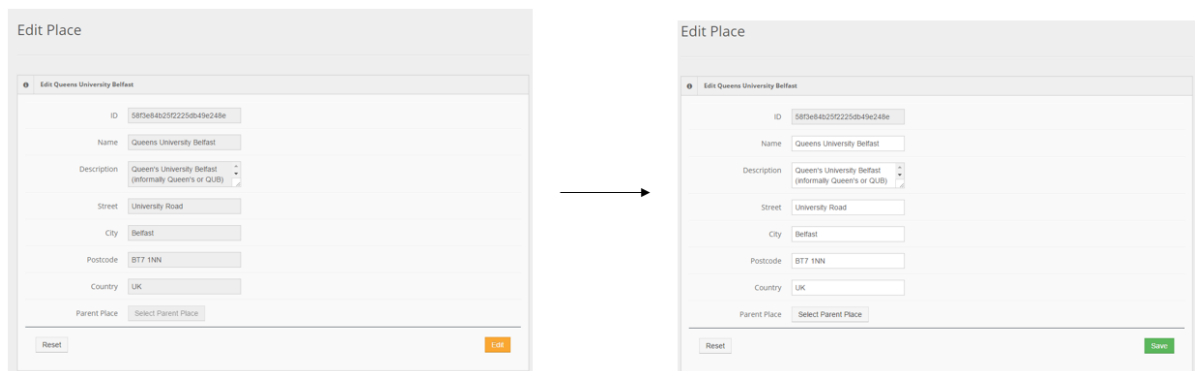


Figure 30: Edit Place screen showing transition between disabled controls and editing mode

When editing a place, initially the user has access to view all the information stored about that place with the options reset and edit available at the bottom of the page. The colour of orange was used to aid as a warning to indicate clicking on the button will take the screen to edit mode as seen on the right of Figure 30: Edit Place screen showing transition between disabled controls and editing mode.

There is also an option of reset that resets the form back to the initial state allowing the user to reset any changes they made. When clicking Edit the button is replaced by a save button that is coloured green. When saving, an indication is shown below the buttons that the form is saving, when the save was successful or there was an error with the save.

### 3.4.13 Delete Place Modal

From the Places screen, within options there is an option to delete the place from the system. To avoid accidental deletion of items, a delete dialogue appears with the question to delete and is coloured red indicating that the deletion of an object is dangerous.

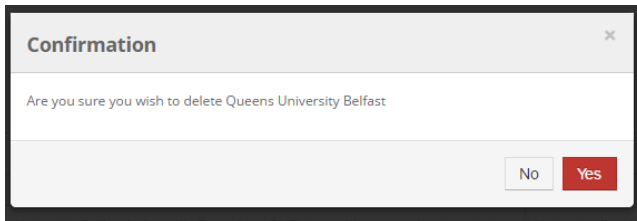


Figure 31: A delete dialogue

The confirmation screen appears above all other content and can be excited by pressing No, outside of the box or the `x` at the top right of the dialogue, taking the user out of the danger.

### 3.4.14 Events

The events screen displays a list of users' events ordered by newest events at the top. The page follows a similar design to the rest of the application and can be viewed by all users. Administrative users have the option to "View All Event" and they will have access to the manage options for all events whether they are the owners of the event or not.

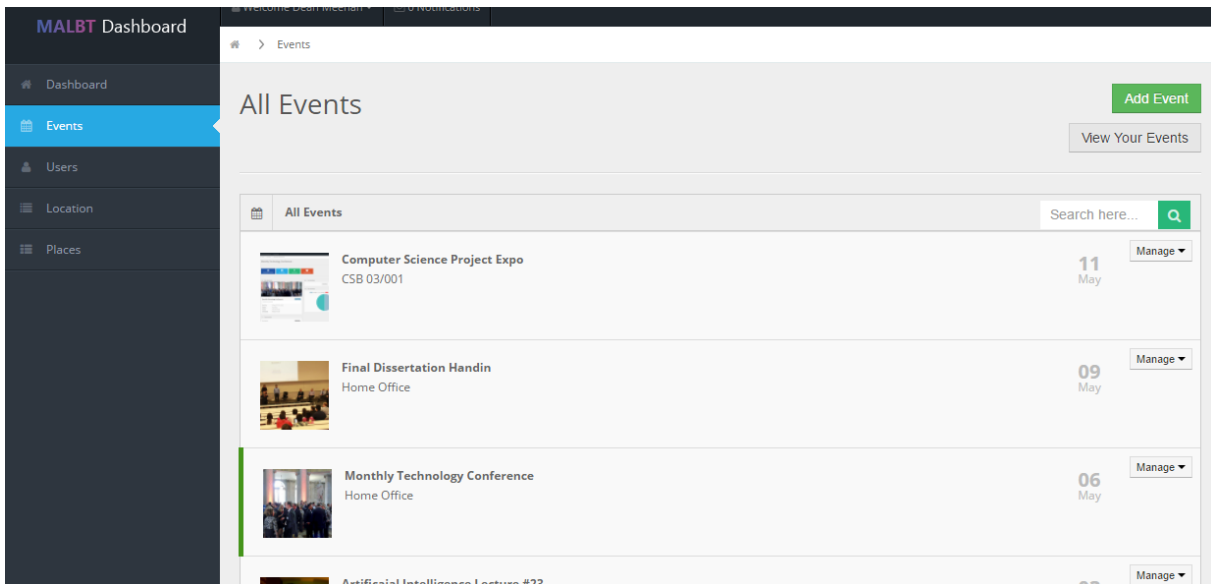


Figure 32: Screenshot of All Event page from the perspective of an admin user, showing an event currently active

When an event is currently active, a green bar will be displayed on the left border. Clicking on any event name will take the user to the event information page. The Manage options follow the same design as places.

### 3.4.15 Event Information

When clicking on an event, the event information page opens displaying detailed information about an event including the event name, description, attendance, start time, duration and other statistics about an event.

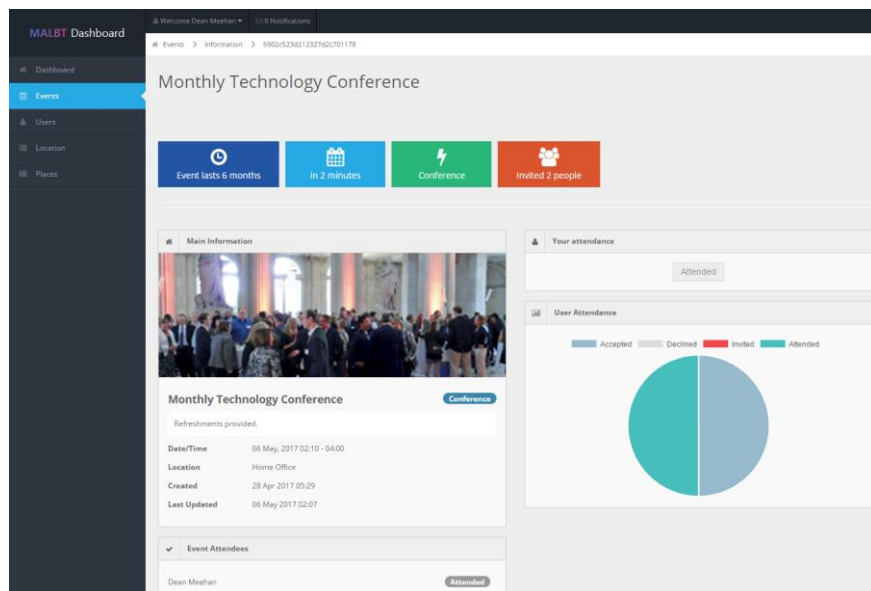


Figure 33: Screenshot of events page

The screen is designed to keep open throughout an event to give the event organiser detailed statistics about attendance. The page is updated asynchronously every minute in the background updating the change of attendance over time.

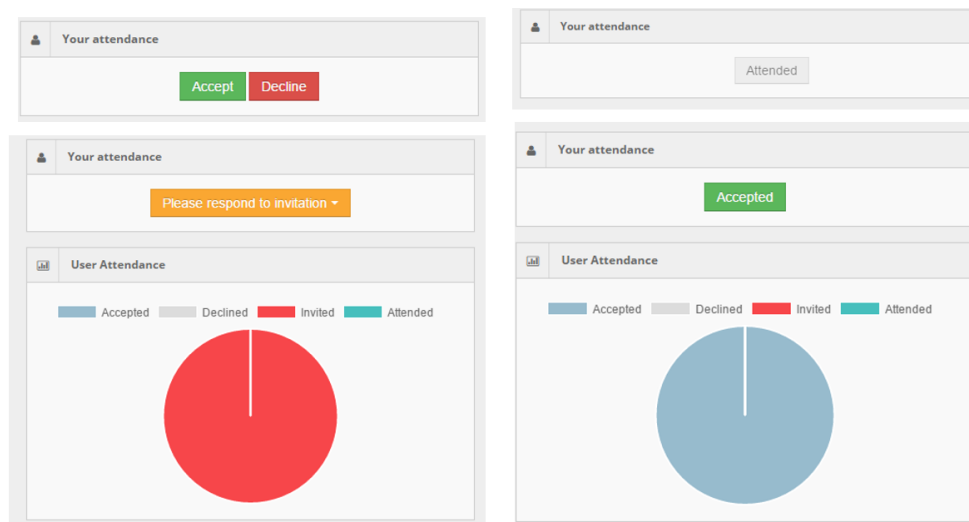


Figure 34: The different attendance states available

A user is able to change their status on the events page. Once the event is over, or they have attended the event, the attendance button changes to a status field of “Attended” or “Absent”.



### 3.4.16 Edit Event

The edit event screen follows a similar design to the editing of other documents within the dashboard. Additionally, the user is able to add or remove invited users to the event.

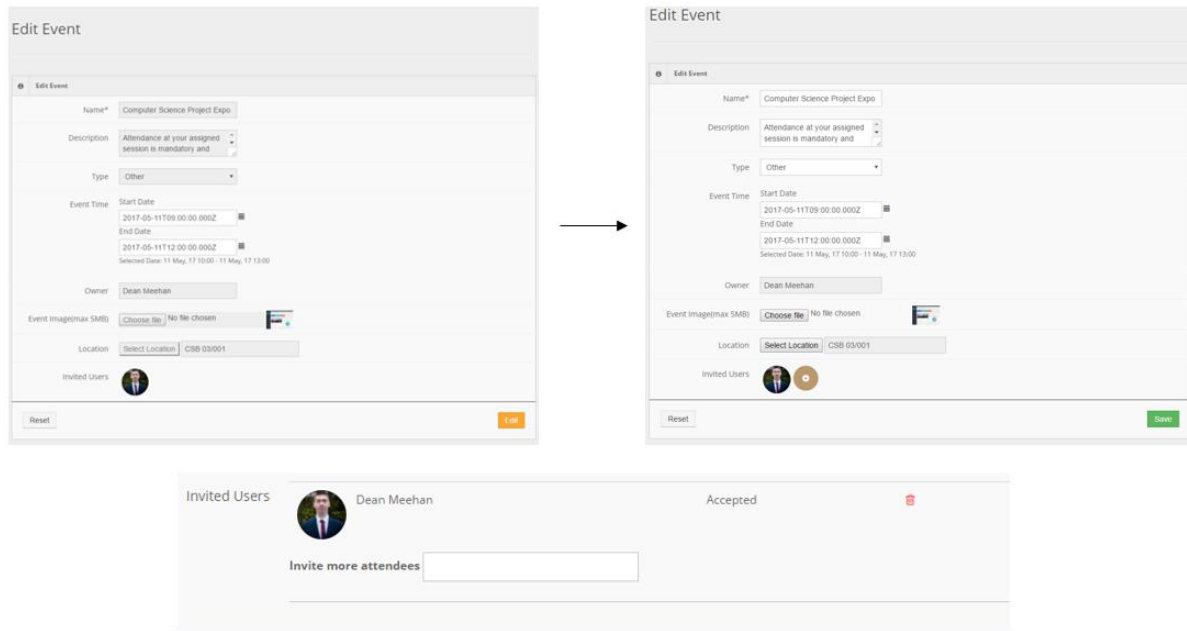


Figure 35: Screenshots of Edit Event page detailing how to invite additional users.

When clicking the tan cog beside the list of users, a dropdown appear showing detailed information about invited users and their status' with the option of deleting them completely from the event or invite new users. When clicking “Save” the form is sent to the API and the document changed within the database.

### 3.4.17 Delete Event

See Figure 31: A delete dialogue

### 3.4.18 Add Event

The add event UI is similar to other add screens. See Appendix 7: Add new event screen showing invite users dropdown.

### 3.4.19 Locations

The locations screen follows the same design as the places screen with added functionality such as list of services, event image and addition options below the manage dropdown.



All Locations						Search here...	Q
Image	Name	Description	Type	Services	Options		
	Home Office	Home office in Newry	Office	Television Games System Disabled Access	Manage		
	Workday Interview Room 001	Workday Interview Room 001	Office	Conference Table	Dashboard		
	Number Seven	Belfast Road	Restaurant	Food Drink	Statistics		
					Edit		
					Delete		

Figure 36: Locations Screenshot

### 3.4.20 Location Statistics

The location statistics page is designed to give an overview of usage at a particular location to admin users such as how many events take place over the past year, which days/hours or weeks are busiest, and a full list of events and their capacity.

The UI was designed to make it easy as possible see a quick overview of a locations statics including providing graphs on events over different timescales.

Boxes at the top of the page display an overview of the number of events that have taken place and an average room occupancy.

The bottom of the design includes a complete list of events at the location including the date/time the event started, invited user capacity and what % of users attended the event who were invited.

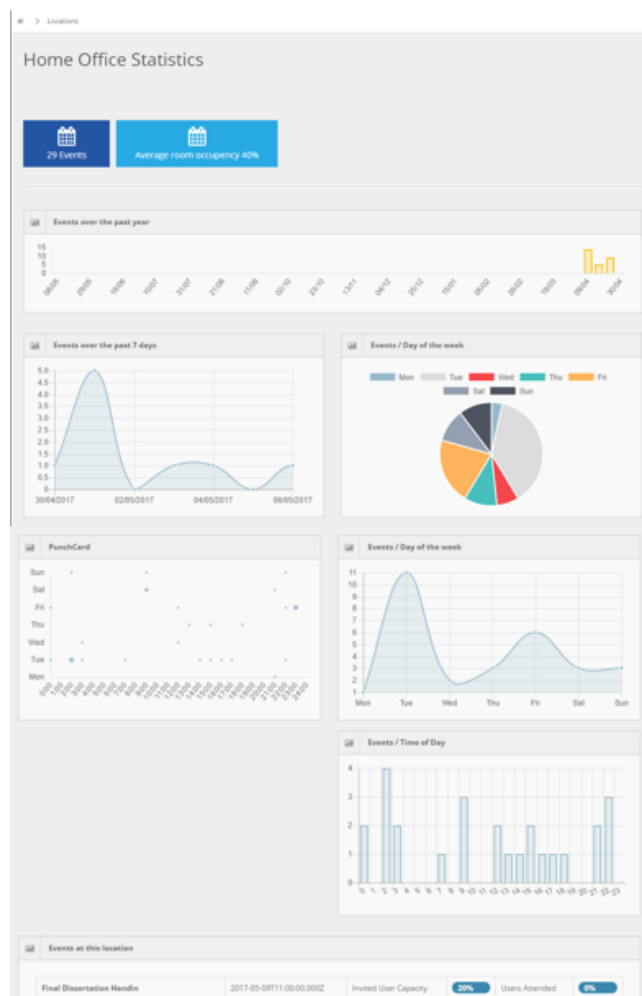


Figure 37: Screenshot of Location Statistics for the home office.

### 3.4.21 Add Location

Adding a location can be a difficult task due to the amount of data needed about its tracking data, location data and other information such as its floor, place and room type. The design of the add location screen has ease of use in mind with creating smart interactive dialogues.

When selecting a Place for a location, the same dialogue appears as when adding a parent place. After chosen, the place address is searched on google maps for its coordinates and a map allows the user to finely choose GPS coordinates.

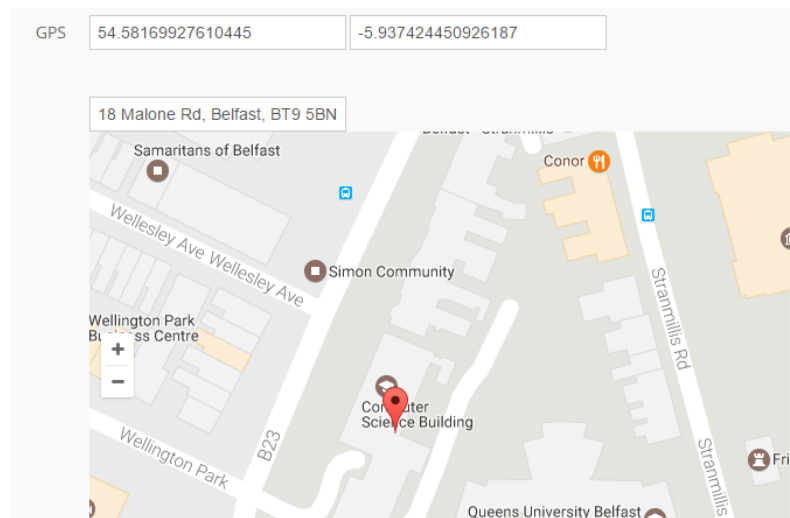


Figure 38: Add Location GPS Selector

As there can be any amount of services available at a location, the Services option has a + and – option to add additional services or remove options. These selections allow the screen not to become cluttered.

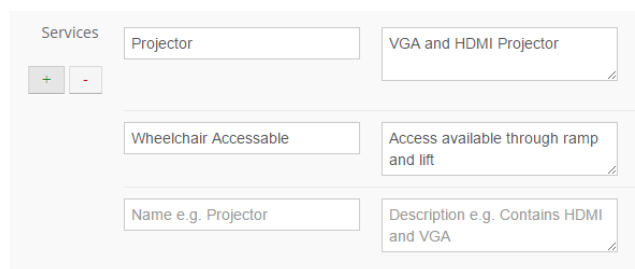


Figure 39: Add Location Services

### 3.4.22 Delete Location

See Figure 31: A delete dialogue

### 3.4.23 Edit Location

The add location UI is similar to other edit screens with the addition of the interactive options seen in Add Location such as Figure 38: Add Location GPS Selector.

### 3.4.24 Users Design

The users design section follows closely the same design used by places for adding, editing, viewing and deleting.

### 3.4.25 Locations Dashboard

The locations dashboard is designed to work on a table touchscreen computer so special thought went into making it quick and easy to use, showing only essential and useful information. The colour scheme keeps with the design of the rest of the project and is designed not to allow the user to return to the dashboard while providing a live view of what's happening. The interface time and date updates every second and the content of the screen updates every 30 seconds.

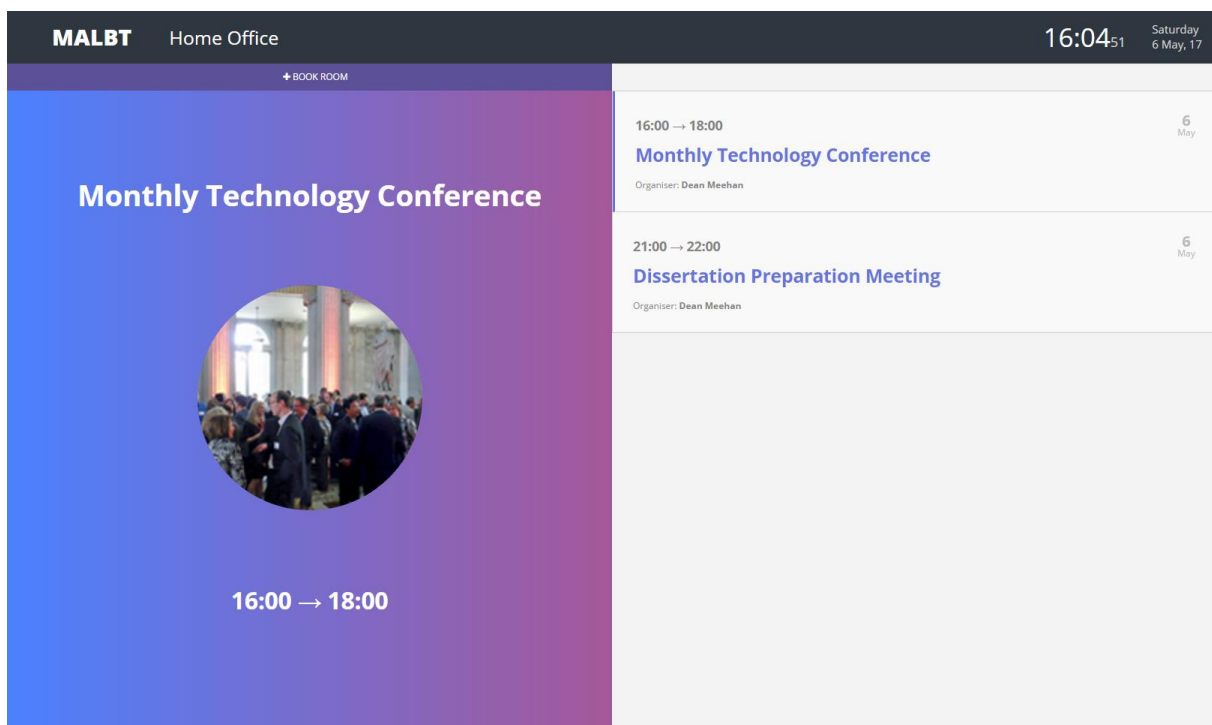


Figure 40: Screenshot of Dashboard Design

The interface contains three distinct sections:

1. The top bar that shows information about the location such as the date and time which is updated live and the location name.
2. The left panel shows currently what is happening at the location, showing the timeframe, event name and event photo. There is also a book now button that appears at the top when there is an event currently happening allowing the user to book an event for later that day. When there is no event currently happening, the event picture is changed to a large plus that the user can tap to book an event.
3. The right panel show a list of events currently happening today. Any event currently happening will had a small border on the left, and any events that have finished will appear semi-transparent.

When booking a room, the interface is designed to avoid the user having to login, just requiring their email address. A back button is available that takes the screen back to the information screen seen in Figure 40: Screenshot of Dashboard Design.

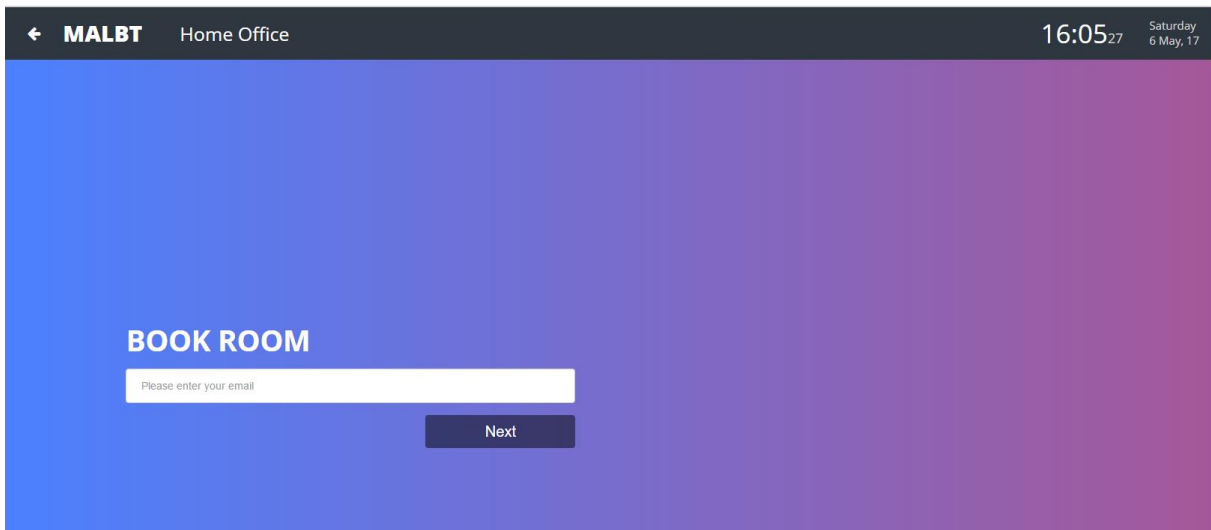
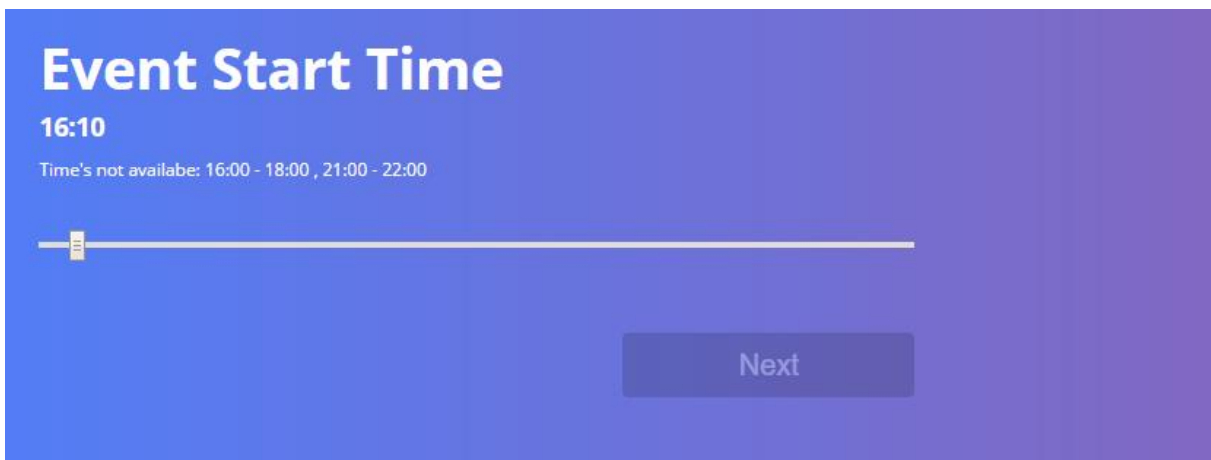


Figure 41: Screenshot of Book Room

Going through each step one by one validates what is inserted building up the form. Users are not able to book an event that overlaps on any other events that day.



Using a slider, the design makes it easy to choose a start and end time. The next button will be disabled when the slider is within a time that overlaps any of the times already making the design as easy to use as possible on touch screen devices.

### 3.5 Code Architecture

The project is designed largely around maintainability and the code architecture closely reflects this trying to follow best practices. The server, dashboard and mobile app all follow a similar pattern of keeping code together, and modulating code as much as possible.

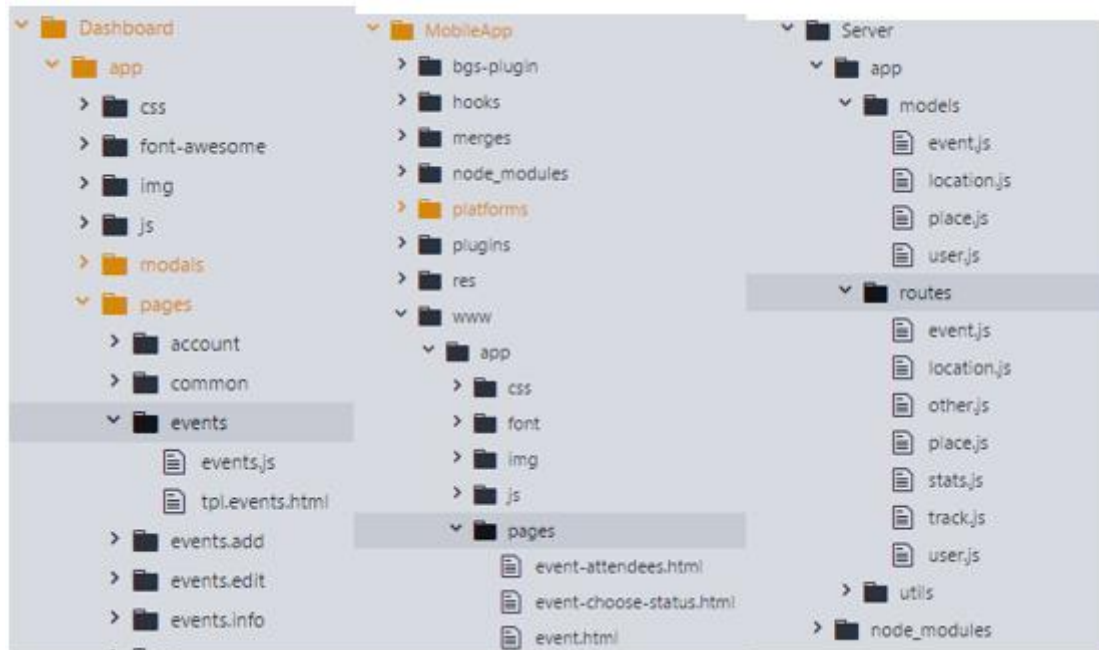


Figure 42: Code Architecture

As well as separating pages as seen in Figure 42, modals, services, images and models are all separated in their own folders creating a hierarchical structure while modularising each section allowing code to be easily found and maintained with a clear design pattern that can be used for future upgrades.

## 4 Implementation and Testing

### 4.1 Choice of Languages

The system language approach was designed to enable code sharing between projects, primarily using ECMAScript (JavaScript) throughout. JavaScript is used within web development, NodeJS and Cordova.

#### 4.1.1 Server

The main requirements for a server language was its suitability for database access and suitability to create a REST interface with the web as easy as possible. Both Python (Using Django or Flask) and NodeJS (Using Express or Restify) were considered. NodeJS using

Restify was selected due to the quick setup time and popularity through the internet for creating a REST server.

NodeJS isn't designed to be publicly facing for web application so Apache was also used to support HTTPS certificates and other features that increased security and maintainability. Apache allows more than one application to be served through the same IP and port allowing the system to host the dashboard and API at the same address. Using Apache's virtual proxy fixes many issues regarding to cross-site scripting and cross origin requests that browsers block.

#### 4.1.2 Mobile Application

The mobile application uses Cordova, a framework that allows the developer to write once and run everywhere supporting a wide amount of different mobile platforms. The main stack of Cordova supports HTML, CSS and JavaScript with plugins available that can create a proxy between native mobile operating system API's and the UI. To aid with design a framework called Framework7 was used that supported advanced features such as handlebars and content routing.

Java was used when creating the Cordova Background service and only will work for Android. Other mobile operating systems require additional services to be wrote to their standards.

#### 4.1.3 Dashboard

The dashboard was designed to work on all browsers and is wrote using HTML, CSS and JavaScript. The application was built on top of AngularJS; a framework that supports additional advanced features such as routing and modularity allowing a modal view architecture to be developed; separating the code used for logic and GUI.

### 4.2 Server Environment

The server is an application that is forward facing to the internet wrote in NodeJS which is compatible with Windows and Unix. There are many different Below are the requirements used for developing and testing:

Type	Name	Version/Information	Usage
Hardware	Digital Ocean Droplet (Virtual Machine)	1GB Memory/ 20 GB Disk / Location London	
Operating System	Ubuntu	16.04.1. LTS	
Software Packages	NodeJS	3.5.2	Language used.
	Mongo	3.2.12	Database Package

	Apache2	2.4.18	Web Server
	Screen	4.03.01	Allows applications to be run in the background in Linux.
	Supervisor	0.12.0	Keeping NodeJS applications alive even when a fatal error occurs.
	certbot	0.4.1-1	Automatically fetches SSL certificates and installs them when they run out.
NPM Modules	bcrypt	1.0.2	Encryption Framework
	bluebird	3.4.7	Promises Framework.
	express	4.15.2	General Purpose Web Framework used to download and server images.
	formidable	1.1.1	Parses form data
	mongodb	2.2.24	MongoDB Driver for interacting between database and Node.
	mongoose	4.8.5	MongoDB Driver Framework
	multipart	0.1.5	Parses multipart requests
	password-hash-and-salt	0.1.4	Hashes and Salts Passwords
	restify	4.3.0	REST framework
NPM Testing Modules	Mocha	3.5.0	Unit-test Framework
	Chai	3.0.0	Assertion Library for Node
	Chai-HTTP	3.3.0	Chai HTTP extension for testing HTTP requests

### 4.3 Dashboard Environment

The dashboard runs on the server alongside the server and is served by Apache. The dashboard requires a TLS/SSL certificate and should not be run on the HTTP protocol. The dashboard is a simple AngularJS client wrote in HTML, CSS, and ES6. Below are the requirements used for developing and testing:

Type	Name	Version/Information	Usage
Hardware	Digital Ocean Droplet (Virtual Machine)	1GB Memory/ 20 GB Disk / Location London	
Operating System	Ubuntu	16.04.1. LTS	
	Apache2	2.4.18	Web Server



Software Packages	certbot	0.4.1-1	Automatically fetches SSL certificates and installs them when they run out.
JavaScript Frameworks	Bootstrap	3.0.0	General Purpose JavaScript and HTML framework
	JQuery	1.7.2	General Purpose JavaScript framework for editing the DOM
	AngularJS	1.6.3	Advanced JavaScript framework with support for MVC and Routing
	Google Maps	~	Framework to allow access to Google Maps
	Moment	2.18.1	Time framework for working with Date and Time in JavaScript
	Chart.js	2.5.0	Chart Framework used for generating charts used through the dashboard.

#### 4.4 Mobile Application Requirements

The mobile app is developed using Apache Cordova and compiled into the Android platform using the Android SDK. The mobile app uses many of the Apache Cordova plugins to provide native functionality and a custom plugin had been developed to provide a background service that sends user location data to the server. Push notifications are provided by Google Firebase service.

Below are the requirements used for developing and testing:

Type	Name	Version/Information	Usage
Framework	Cordova	6.5.0	
Android SDK	Android Debug Bridge	1.0.32	Used for debugging Code
Cordova Plugins	geolocation	2.1.0	Plugin to access device Location Services
	Eddystone	1.0.0	Plugin to access Bluetooth Beacons
	Whitelist	1.3.1	Plugin to allow internet connectivity
	Plugin-ble	2.0.1	Plugin to access Bluetooth Services
	Camera	2.4.0	Plugin to access Camera and File Manager
	Firebase	0.1.20	Plugin to listen for Push Notifications

Background Service	technology.dean.backgr oundservice	Located in the FYP- MALBT/MobileApp/bgs- plugin folder of the github repo	Plugin created to push location data to the server.
--------------------	---------------------------------------	--	--

## 4.5 Testing

### 4.5.1 Unit Tests

Unit tests have been developed for testing all functionality of the server, ensuring any calls to an API endpoint match what is expected. Unit tests are run by calling `npm test` from within the server folder.

```
#Place Route: DELETE /place/:id
✓ Place deleted
✓ Returns correct object
✓ Doesn't allow access to anonymous users
✓ Doesn't allow access to non-admin users

#Place Route: PUT /place
✓ Place edited with new name
✓ Returns correct object
✓ Doesn't allow access to anonymous users
✓ Doesn't allow access to non-admin users

#Statistics Route: /stats/location/:id
✓ Returns correct location object
✓ Returns array of events
✓ Returns correct amount of events

#Polling Route: /poll
✓ Returns String
✓ BSSID: User is marked attended to event 58f8fd6c4cb8b419b35008d3
✓ GPS: User is marked attended to event 58f8fd6c4cb8b419b35008d3
✓ GPS Location out of Range: User is marked accepted to event 58f8fd6c4cb8b419b35008d3
✓ Beacon: User is marked attended to event 58f8fd6c4cb8b419b35008d3
✓ Doesn't allow access to anonymous users

346 passing (239ms)
```

Figure 43: Screenshot of Server Unit Tests with Mocha

The server uses additional development frameworks to perform unit tests such as mocha (for running the test), chai (an assertion library) and chai-http (for making HTTP requests to the server and then resting the responses). The Unit tests have coverage of all API endpoints.

### 4.5.2 Regression Tests

Regression tests are used for the testing of GUI elements within the dashboard and Mobile Application. They test the usage of the UI against expected and actual results to ensure nothing breaks though development of additional features.

The regression tests are created on an excel document detailing the application name, version and tester. The tester is then required to go through the list of tests, following the steps for each test and marking if the test has passed or not.

1	Date	07/05/2017					
2	Application Title	MALBT - Mobile Application Location Based Tracker					
3	Version	1.0.0					
4	Your Name	Dean Meehan					
5	Your Email	d3an.meehan@hotmail.com					
6							
7	ID	Name	Prerequisites	Expected Actions	Steps	Expected Results	Result
8	1	Test Access		Webpage loads with login screen	1. Access Google Chrome 2. Enter Url https://dean.technology/dashboard	Dashboard Appears requesting login	PASS
9	2	Enter Incorrect Username or password		An error message appears	1. Enter False Email Address 2. Enter False Password 3. Press Login	Error message appears indicating incorrect username/password was incorrect	PASS
10	3	Enter Password with no username		An error message appears	1. Enter Password 2. Press Login	Error message appears: Username and/or Password is empty	PASS
11	4	Enter Username with no password		An error message appears	1. Enter Email Address 2. Press Login	Error message appears: Username and/or Password is empty	PASS
12	5	Correct Login		User is taken to Dashboard Homepage	1. Enter correct username 2. Enter correct password	Homepage is loaded with Welcome "Name" at the top of the page.	PASS
13	6	View Notifications		Message displaying amount of notifications appears	1. Click "Notifications" button on top bar	Either "No Notifications" or a list of notifications appear.	PASS
14	7	View user profile		Users data is displayed	1. Click Welcome "Name" 2. Click on "My Profile"	Profile page is loaded displaying users picture, fullname, ID, Email and Location	PASS

Figure 44: Screenshot of Regression Tests for the Dashboard

## 5 System Evaluation

The project can be evaluated based on 5 types of criteria below.

### 5.1 Functionality

The system functions as expected with both functional and non-functional requirements being met. As per the requirements and specification, the system should be able to record utilisation of a location and track users' attendance to meetings providing a live overview of everything by accessing the system through the cloud. The system is able to identify users who are attending meetings, and notify them before a meeting via push notification when an event is about to start and when they have been invited to a new event.

### 5.2 System Reliability and Speed

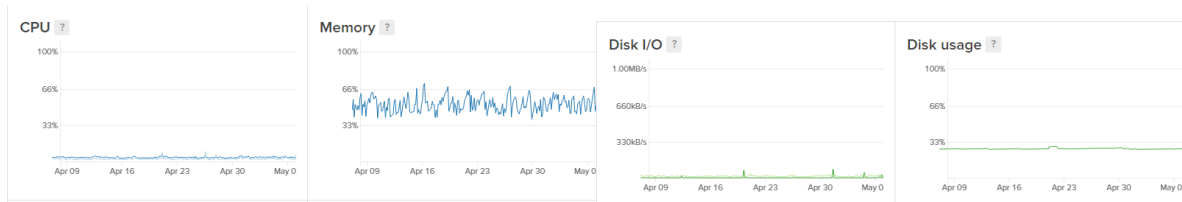
The system's design enforces reliability and speed by being located in a datacentre. The cloud provider supports a high availability, being in a datacentre they monitor the health of their systems and even though they can't guarantee 100% uptime, using a cloud provider can minimise the potential impact.

The server produced in this project is also run behind an application called Supervisor which monitors code changes in NodeJS and the health of an application, being able to automatically handle fatal errors bringing the system back online.

### 5.3 Performance

The server is currently running on a cloud instance within digital ocean on a virtual machine with 512GB of memory and 20GB disk running on Ubuntu which is the lowest plan available.

Over the past 30 days the usage has been minimum between idling and testing. Further research would need to be done to see how the application performed under pressure with many concurrent users.



## 5.4 Installation and Operation

Having all aspects of the application situated within one Ubuntu machine allows easy of installation and setup without human interaction. The system could be installed using a single bash script or the system could be built into a container such as Docker (Linux Container) with very little changes in system setup.

Once the application is installed, the server should stay online. Any fatal errors such as the server going down will cause the server to restart. Additional fail safes should be installed such as live health monitoring of the system to alert administrators of any problems that may appear.

## 5.5 Cost

The cost of the project is relatively low, the server and dashboard can be scaled both vertically and horizontally. As the project is designed to be run in the cloud, it can be installed on corporation's internal infrastructure adding little to no cost, or run in the cloud with relatively low costings. Installing the system into a location can cost relatively nothing if there currently is an infrastructure such as WLAN's to aid with location tracking. Additional Bluetooth beacons can be bought and installed anywhere such as inside lamp bulbs, light switches or standalone battery devices.

## 5.6 User Feedback

In order to get a fair evaluation of the system, a survey was also produced and sent out by email to 10 people. The questionnaires contained some questions based on how users perceived the look and feel of the application. The questionnaire results showed that users where satisfied with the design of the mobile app and security of the application giving a rating of 100%, but some users felt there could be an improvement of the design in the dashboard and ease of use with an average rating of 97%. To view the results and questions in full, see Appendix 8:Survey Results.

## 6 Conclusion

This project attempted to develop a solution to the problem of poor space utilisation within organisations allowing them to maximise the potential of existing spaces and make decisions around building management. A solution was developed that achieved the success criteria providing a cloud-based solution that is able to track the utilisation of a building by tracking room usage and the users using these locations.

The solution developed successfully monitors utilisation within a location and reports can be generated to allow corporations to make decisions of a location. End-users of the system are able to manage their own interactions with a building by booking and attending events. Administrators and event owners can monitor the attendance in their events without any manual input with the use of automated tracking of users through their phones using a range of location technologies.

The solution also contains many weaknesses within its core functionality. Some of these weaknesses could be tackled with more time and resources to complete the project while others would need to be looked at in detail. The project lacks an easy way to view utilisation of a place such as buildings or groups of buildings, currently an administrator will have to view each location individually viewing their usage. The project could be expanded to provide more reports to view utilisation on a range of different levels of locations or utilisation of a particular user.

More focus could also have been placed on the management of events such as reoccurring events, automatic inviting of groups of users and the inclusion of a calendar so events can be seen more clearly.

Within the scope of tracking, the solution will only work if all users within a building use the solution and carry their Android mobile phones with them. Additional requirements such as organisational policy and the usage of fall-back technology would need to be used to stop users just using rooms without booking them or users attending meetings but not carrying an Android phone with them.

To conclude, the project develops a solution that achieves the success criteria and could be successfully used in production to monitor usage of building. More work is needed to expand the range of features within the dashboard and the amount of supported mobile devices the project can use.

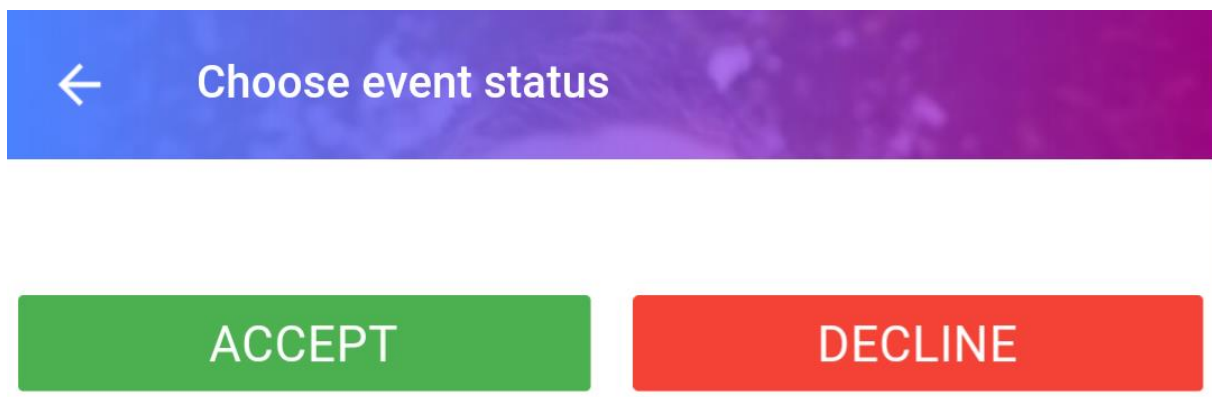
## 7 References

- [1] building.co.uk, “Building costs,” building.co.uk, 2008. [Online]. Available: [http://www.building.co.uk/Journals/Builder\\_Group/Building/2008\\_Issue\\_8/attachments/buildingcosts.pdf](http://www.building.co.uk/Journals/Builder_Group/Building/2008_Issue_8/attachments/buildingcosts.pdf). [Accessed May 2017].
- [2] Network Working Group, “Hypertext Transfer Protocol -- HTTP/1.1,” June 1999. [Online]. Available: <https://tools.ietf.org/html/rfc2616>. [Accessed March 2017].
- [3] Kangax, “ECMA6 compatibility table,” 2 May 2017. [Online]. Available: <https://kangax.github.io/compat-table/es6/>. [Accessed 2 May 2017].

## 8 Appendices

PLEASE RESPOND

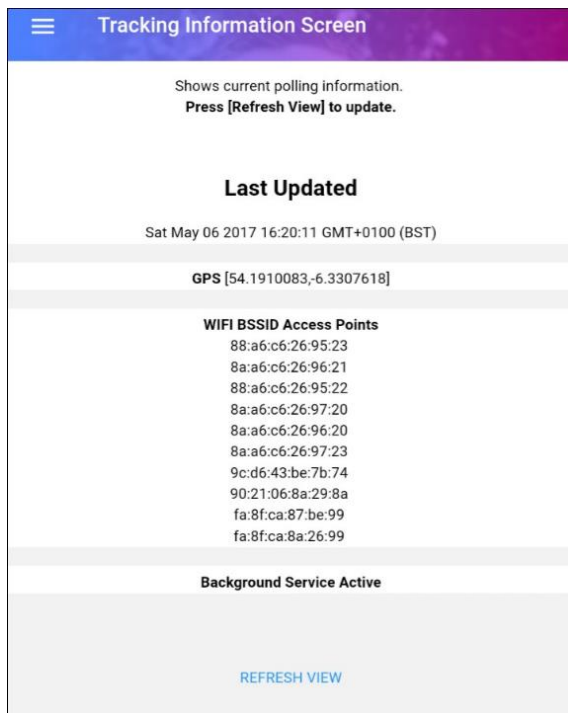
Appendix 1 :Please Respond Button



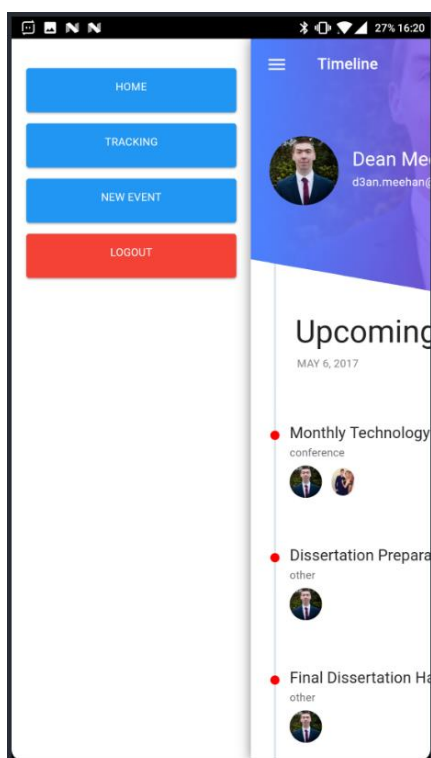
Appendix 2 : ACCEPT / DECLINE

← Event Attendees		
	dean meehan	accepted
	carl meehan	invited
	rachel shields	invited
	joe patterson	invited
	cotton jacks	invited
	paul mccabe	invited
	noadmin meehan	invited

Appendix 3:List of Event Attendees

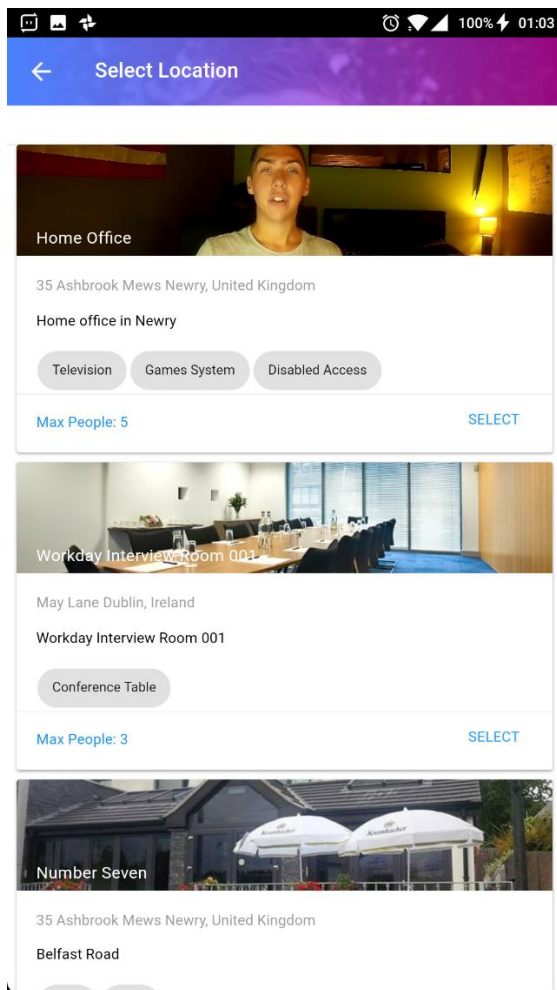


Appendix 4: Tracking Information Screen



Appendix 5: Sliding Menu





Appendix 6: Select Location Screen

Add new Event	
Name*	<input type="text" value="My Event"/>
Description	<input type="text" value="Short description about the event"/>
Type	<input type="text" value="Please select an option"/>
Event Time	<div>Start Date <input type="text"/></div> <div>End Date <input type="text"/></div> <div>Selected Date: -</div>
Owner	<input type="text" value="Dean Meehan"/>
Event Image(max 5MB)	<input type="button" value="Choose file"/> No file chosen
Location	<input type="button" value="Select Location"/>
Invite Users	<input type="text"/> <div> Dean Meehan &lt;d3an.meehan@hotmail.com&gt;  Carl Meehan &lt;carl.meehan509@hotmail.com&gt;  Rachel Shields &lt;rachel.shields@hotmail.com&gt; </div>

Appendix 7: Add new event screen showing invite users dropdown

Survey Results														
<b>Date</b>	2nd May 2017													
<b>Title</b>	MABLT Questionnaire	USER #												
<b>#</b>	<b>Question</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>AVG</b>	<b>%</b>	
<b>1</b>	How Satisfied are you with the software's ease of use? /5	5	5	5	5	4	5	5	4	5	5	<b>4.8</b>	96	%
<b>2</b>	Passwords are encrypted when transmitted, and hashed and salted when stored. How satisfied are you with the software's security? /5	5	5	5	5	5	5	5	5	5	5	<b>5</b>	100	%
<b>3</b>	How satisfied are you with the design of the mobile app? /5	5	5	5	5	5	5	5	5	5	5	<b>5</b>	100	%
<b>4</b>	How satisfied are you with the design of the dashboard? /5	5	5	5	5	5	5	5	4	5	5	<b>4.9</b>	98	%
<b>5</b>	Do you have any thoughts on how to improve this software? /5	n/a			No ne	N o	No pe		N o					

Appendix 8: Survey Results