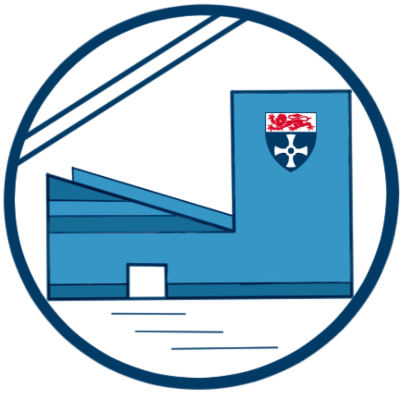
Newcastle University

Specification Document
Project Name: Urban Sciences Building Tour & Navigation

Team 22



Document Information

Project Name: Urban Sciences Building Tour & Navigation

Prepared by:	Email:
● Alex Beeching	--------------------
● Alexander MacLeod	--------------------
● Daniel Vincent	--------------------
● Shorif Akhonda	--------------------
● Chester Lloyd	--------------------
● Patrick Lindley	--------------------
● Moeez Shahid	--------------------
● Daniel Walker	--------------------

Document Version No: 1.7
Preparation Date: 12/12/18
Document Version Date: 07/03/19

Version History

Ver. No.	Ver. Date	Revised By	Description
1.0	08/12/18	Daniel Walker Daniel Vincent	Roles Other considerations Deliverables
1.1	09/12/18	Alex MacLeod Chester Lloyd Daniel Vincent Patrick Lindley	Functional requirements Non-functional requirements Software design Definition of terms System Architecture
1.2	10/12/18	Alex Beeching	Analysis
1.3	24/02/19	Alex Beeching	Updated analysis
1.4	28/02/19	Chester Lloyd Alex Beeching	Improved layout and consistency of the document

		Patrick Lindley	Added table of contents Updated software platforms Added descriptions to Software Design
1.5	05/03/19	Daniel Vincent	Added class diagrams and descriptions to Software Design Updated Software Design overview
1.6	06/03/19	Chester Lloyd	Added contribution matrix Added website designs Added references page
1.7	07/03/19	Moeez Shahid	Updated Task Plan
		Chester Lloyd	Updated Software/Hardware Platforms Updated Definition of Terms
		Daniel Vincent	Updated class diagrams and descriptions in Software Design Added references Added to component diagram to High Level Overview
		Daniel Walker	Added activity diagrams & descriptions to the High Level Overview
1.8	08/03/19	Alex Beeching	Final review with minor edits to all sections

Table of Contents

Executive Summary

The purpose of this document is to provide detailed information regarding Team 22's Urban Sciences Building Tour & Navigation project to the Newcastle University School of Computing and Nicola Dorman, the School Manager.

Analysis

This section details all research conducted on similar pre-existing systems, along with research regarding stakeholders and other details of a similar variety. Also included in this section is the aim of our project.

Roles and Deliverables

This section will detail in depth descriptions of the roles undertaken by all members in our team along with details regarding individual and team deliverables.

Project Plan

Our project's development timeline mapped out through a GANTT chart.

Hardware & Software

Details of what hardware and software will be employed in the development of our project with reasoning behind the choices we made in terms of what software we decided to work with.

Solution Requirements

All the functional and non-functional requirements for our project listed out including priorities and brief descriptions for each requirement.

Software Design

This section is split into a number of subsections and also details our plans for interactivity between our app project and our website. Subsections in this section include a general overview of the software we aim to produce, decisions made regarding the architecture of our system, a high level overview, package and class diagrams, dynamic behaviours across the system and details regarding plans for the GUI.

1. Background & Analysis

1.1 Stakeholders

The stakeholders of our app include all university staff, all students at Newcastle University ranging from undergraduates to PhD (though primarily computer science students) and visitors. Visitors include prospective students, parents and other guests.

1.2 Analysis Process

When performing an analysis on existing systems, there are certain aspects of existing systems that require more attention than others depending on the function the project is being made to fulfil. For the Urban Sciences Building tour & navigation app, the main aspects that should be analysed in existing software, in accordance with Nielsen's ten heuristic principles, are :

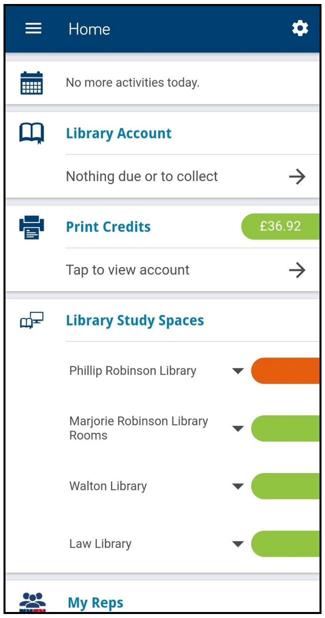
- Recognition rather than recall
- Match between system and real world
- Consistency and standards
- Flexibility and efficiency of use
- Aesthetic and minimalist design

In existing software the features that will come under analysis are those most closely related to self-guided tour or navigation features as these will be the primary features delivered by the Urban Sciences Building app. Design for existing software will also be analysed.

Analysis of existing systems

Newcastle University App

(Newcastle University, 2019)



Home

No more activities today.

Library Account

Nothing due or to collect

Print Credits £36.92

Tap to view account

Library Study Spaces

Phillip Robinson Library

Marjorie Robinson Library Rooms

Walton Library

Law Library

My Reps

This app is the main app for details regarding Newcastle University. It has features including details on how busy the University owned libraries are at any given time, students' timetables, personal tutor information and exam dates.

The Newcastle University app has been very useful for the design team as a basis for the design of the Urban Sciences Building app. WIth one of the stakeholders of the app being identified as students attending the University it is logical to maintain a similar UI to add simplicity for users. This matches the heuristic guideline for consistency and standards. Specifically the design features the Urban Sciences Building will take inspiration from will be the blue and white colour scheme, the sidebar design and the segmented design for displaying different information e.g Library Account, Print Credits, Library Study Spaces.

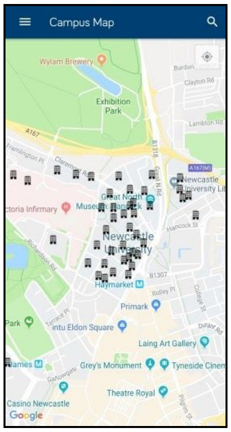
Desirable Qualities

The most desirable quality here is the minimalist design giving multiple pieces of information clearly without much difficulty for the user. This software is a good example of recognition rather than recall where the user can find relevant information without having to learn how to navigate the app prior to its use. A good example of this heuristic principle in practise is the Library Study Spaces segment using a traffic lights system to display simplified details on how many people are occupying certain University facilities. This would be a good system to take inspiration from for displaying occupancy in the Urban Sciences Building computer clusters.

Undesirable Qualities

This app works on a much larger scope than what is intended for the Urban Sciences Building app as it focuses on providing information for the entire University rather than just one building. This is the software's approach to navigation to all the different buildings on campus.

When a building is selected the only information given is the name - something more useful would be to at least give the purpose(s) of the building. This is a very basic approach to navigation and for the Urban Sciences Building app a more detailed approach will be taken.



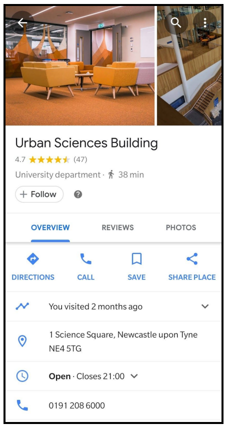
Google Maps

(Maps - Navigate & Explore, 2019)

Google Maps is Google's solution to worldwide navigation and is very widely used with around 10 million downloads from the Google Play store and is generally found to be pre-installed on factory new Android phones.

It provides many features to its wide user base including worldwide navigation with live feedback, route planning, information on locations (example of this shown on right) and personalised information on local areas catered to the user from where they have visited previously.

The team researched this app for inspiration on tackling navigation and presenting information for a self-guided tour around the Urban Sciences



Urban Sciences Building
4.7 ★★★★½ (47)
University department · 38 min

You visited 2 months ago

1 Science Square, Newcastle upon Tyne NE4 5TG

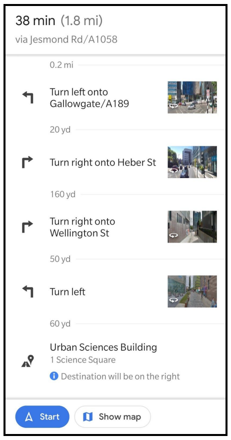
Open · Closes 21:00

0191 208 6000

Building. When tackling navigation the heuristic aspects we deemed most important are match between the system and real world, flexibility and efficiency of use & consistency and standards. Our aim for the navigation feature is to provide a consistent format for giving directions that accurately reflect the real world and satisfy the heuristic principles mentioned.

Desirable Qualities

Google Maps' approach to navigation satisfies the heuristic principles under analysis and is a source of inspiration for the design of the Urban Sciences Building navigation feature focusing on textual turn-by-turn navigation rather than a visual representation. Without the ability to implement live tracking within the Urban Sciences Building this is a more suitable approach to navigation as a visual representation only works well when you have the ability to update the user's location along their desired route because without this the user is unable to track their progress and would have to keep a mental note on their current location on the map. The textual approach used by Google Maps eliminates any ambiguity for the user as to where they currently are along their route as the user will simply be following a chronological list of directions that can be followed one by one. Google Maps' implementation of turn-by-turn navigation can be seen on the right.



38 min (1.8 mi)
via Jesmond Rd/A1058

0.2 mi

Turn left onto Gallowgate/A189

20 yd

Turn right onto Heber St

160 yd

Turn right onto Wellington St

50 yd

Turn left

60 yd

Urban Sciences Building
1 Science Square
Destination will be on the right

Undesirable Qualities

In order to use Google Maps' navigation features a connection to the internet is required unless the user specifically downloads a map for a certain area. For the scope in which the software operates this is suitable as it avoids the user having to download very large amounts of data on maps. For the Urban Sciences app, however, this is not a suitable approach as the data set is much smaller so a more suitable app approach is to cache all the map data upon initial download of the app.

With the turn-by-turn navigation Google Maps gives the user an estimated time for the completion of their journey. For navigating around just one building this is a level of information not necessary.

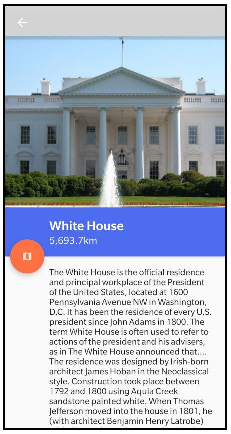
Washington DC Tourist

(Washington DC Tourist, 2018)

The Washington DC Tourist app is used primarily by tourists for finding out basic information about certain landmarks found around the Washington DC area. The app provides a picture and long description of multiple landmarks whilst also providing their current distance from the user. The app also works off the back of Google Maps by allowing the user to, at the press of a button, be taken to Google Maps with their landmark of choice being searched.

Desirable Qualities

One thing the Washington DC Tourist app does well is fulfill the heuristic principle of having an aesthetic and minimal design. All the information on the app is available to the user at the touch of, at most, two buttons. The layout used by this app for presenting the information on each landmark is something that will be used as inspiration for the design of the tour aspect of the Urban Sciences Building tour and navigation app in conjunction with Google Maps' solution to presenting information seen above.



White House
5,693.7km

The White House is the official residence and principal workplace of the President of the United States, located at 1600 Pennsylvania Avenue NW in Washington, D.C. It has been the residence of every U.S. president since John Adams in 1800. The term White House is often used to refer to actions of the president and his advisers, as in The White House announced that.... The residence was designed by Irish-born architect James Hoban in the Neoclassical style. Construction took place between 1792 and 1800 using Aquia Creek sandstone painted white. When Thomas Jefferson moved into the house in 1801, he (with architect Benjamin Henry Latrobe)

Undesirable Qualities

On the page seen on the right the amount of detail given for just one landmark is staggeringly large with no formatting. All the information is presented as one large paragraph which is not digestible for a user. To improve upon this our app will break down areas of the Urban Sciences Building into several pages similar to the design of the Washington DC Tourist app with a shorter description provided on each page.

Another undesirable quality of the Washington DC Tourist app is the lack of navigation features on the app itself. For our app navigation around the Urban Sciences Building is unable to rely on Google Maps as it operates on a larger scope than what is needed so our app cannot adopt a similar strategy.

Urban Sciences Building Tour

As part of the initial research on existing systems to aid the design choices made on the Urban Sciences Building app three members of the team attended a tour around the Urban Sciences Building with the client, Nicola Dorman, to gain insight on how the tour is currently presented. The purpose of the tour is primarily to showcase the features provided by the building to visitors or prospective students. In order to most effectively carry out this purpose a tour was taken to gain insight on where the client placed most emphasis on the building. Having taken the time to understand how the client would ideally want the building showcased it gave the team a better idea of how to tailor the tour content provided when presenting it to the user.

Domain Analysis Summary

This section will briefly outline the decisions made on what features are to be included in the app based on the analysis of existing systems above. This summary will not include a list of all features to be included in the project but only those influenced by existing systems.

Desirable Features	● Turn-by-turn text navigation ● Tour content with emphasis on content reflective of how the client wants the building represented ● Feature showing if computer clusters are busy using traffic light design (red, amber, green)
Undesirable Features	● Live navigation with map representation ● Timings shown for time to reach destination ● Features operating on a larger scope than is desirable (e.g. location postcodes, students' timetables/ exam dates)

1.3 Project Purpose

Through the analysis of the specification we were given along with correspondance with Nicola Dorman, our aim on what it is we want to deliver to our stakeholders is as follows. We aim to produce an app that helps users navigate around the Urban Sciences Building in two respects.

The first is through a self guided tour that follows the pre-existing tour path used by the School of Computing for showing people round the facilities available. The aim here is that the app will subsidise the need to have a member of staff or ambassador conducting the tour as the app will be sufficient in providing the user directions along with useful information along the way.

The second fulfillment of the specification in aiding users in navigating the Urban Sciences Building is through the implementation of a turn-by-turn navigation system giving the user clear directions to get from A to B from anywhere to anywhere in the building.

Through correspondence with the client we were posed the question of what would be most important to us as students of computing to have as features for an app to help students in the Urban Sciences Building. The posing of this question is what fueled all the subsequent (lower priority) features that make up the remainder of the functions provided by our app.

The website also has purpose within our project as a platform through which admin in the School of Computing can quickly and easily change information presented on the app such as the location of lectures' offices or the day's/ week's opening and closing hours for the Urban Sciences Building.

Overall our aim is to provide an app that helps users explore the Urban Sciences Building, provide useful information and helps administrators alter information and get said information out to those affected as soon as possible.

2. Roles and Deliverables

2.1 Roles

Team Leader - Alex Beeching

This role comes with the most responsibility as they overlook the project as a whole, ensuring everyone is collaborating effectively and know what they are doing. They also need to encourage fairness and agreeability within the team, taking on board all team members inputs to decide upon a final decision. They represent the team in leader meetings, as well as being the first point of call should there be any problems with the team as a whole. In his absence, we have appointed a deputy leader (D.Vincent) to pick up the responsibilities of this important role, who will be constantly up to date with team developments and issues.

Researchers - All Members

All members will take on this role at some stage. This is a greater priority earlier on when planning the project, for example researching the building, accessibility options and other similar applications. However, will be constantly ongoing with programmers researching new libraries to utilize, designers learning new software etc.

Lead Programmer - Alexander MacLeod

This is the team's most experienced programmer, responsible for providing guidance for the programming team during implementation and making sure the programmers are all working towards the same goal to minimize confusion. It is this person's responsibility to ensure junior programmers know what tasks to complete as well as advise them and review their code when necessary. In his absence, one of our junior programmers would step up into this position, as through regular communication all programmers are equally aware of current progress. In order to meet deadlines, either D.Walker or M.Shahid would join the programming team to assist, if needed.

Lead Web Developer - Chester Lloyd

This person is responsible for website used to support the application. This includes front end development with the website layout, content and interactivity; and back end server implementation for updating information within the app and login functionality. Since there are no junior web developers (as this is not the focus of the project), general junior programmers can be utilized to help if the workload becomes too great (who would also take on this role in C.Lloyd’s absence).

Lead Designer - Shorif Akhonda

This person is responsible for producing designs for the UI of all areas of the application, so programmers can use this as a basis for implementation. This individual has a broad understanding of the system requirements, to be as accurate as possible when producing designs. It is this person's responsibility to lead junior designers, allocating tasks, providing assistance where necessary and ensuring everyone is ‘on the same page’. In his absence, this responsibility would be gathered by either D.Walker or M.Shahid, depending on involvement in design tasks.

Junior Programmers - Daniel Vincent, Patrick Lindley

These individuals complete tasks allocated by the lead programmer during development to spread the workload more evenly, whilst constantly communicating with the lead. Since the application is the main priority of this project, junior programmers will prioritize this over the website.

Junior Programmer / Designers - Daniel Walker, Moeez Shahid

These members take a more general role within the team, helping with whichever aspect of the project is under the most strain at any time. This makes the team less rigid and more agile, as through the year some areas will become overloaded and need additional assistance.

Testing - All Members

It is the responsibility of these people to expose flaws in the system and advice the programmers where they were found so they can be fixed as soon as possible. This will be ongoing throughout the whole implementation process, so everyone will be doing this at some stage to a different extent. Towards the end of the project this will be a greater priority, with team members being specifically tasked with this role.

Component testing as programming tasks are undertaken will be an ongoing process for the programmers. For example, Junit testing to ensure cafe items are properly read in from the database, complete with a name, price, etc. Full system testing will be undertaken in the majority by team members less involved with the programming, as this will ensure tests are non-directed and results are not influenced by their expectations. It will also allow them to actively fix any errors simultaneously to other members carrying out tests, hence following our agile planning strategy.

Documentation - All Members

This role will be up taken at some point by every team member, however as with testing, will vary in intensity. Members with junior roles, notably the 2 general team members, will adopt most of this work. It is the responsibility of someone with this role to produce large formal reports, such as: specifications, final production and testing documentation and design plans. Communication with members in other areas will be essential in order to make documents as accurate as possible. UML diagrams will be completed by A.Beeching and D.Walker, working alongside the programming team to ensure accurate representations.

2.2 Deliverables

Android Application Prototypes

Once the specification has been finalised and delivered to the client, development of the first Application Prototypes will being. These prototypes will be regularly tested and improved upon, then submitted to the client for feedback. Using this development loop, production of the application will continue until the client is satisfied with the product.

Final Android Application

Once the client has received a prototype which fulfills all aspects of the specification and has returned positive feedback, final testing will take place on that build of the application, after which the finished product will be delivered to the client.

Website

Like the Application Prototypes, the website will be continually developed until client satisfaction and positive QA testing has been attained. At this point, the website’s interoperability with the application will be tested and once all tests are complete, the website will be ready for client use. Unlike the application prototypes, development changes to the website will be live, rather than build by build.

Product Demonstration

Alongside Application Prototypes, if significant changes or feature additions have been made, live demonstrations will be made to the client to open these changes up for discussion and critique by the client.

Product Documentation

The final build of the Android application and website will be accompanied by full documentation of the features, usage and implementation in both in text form.

2.3. Modifications

No modifications have been made to the program itself during the construction of the app.

3. Project Plan

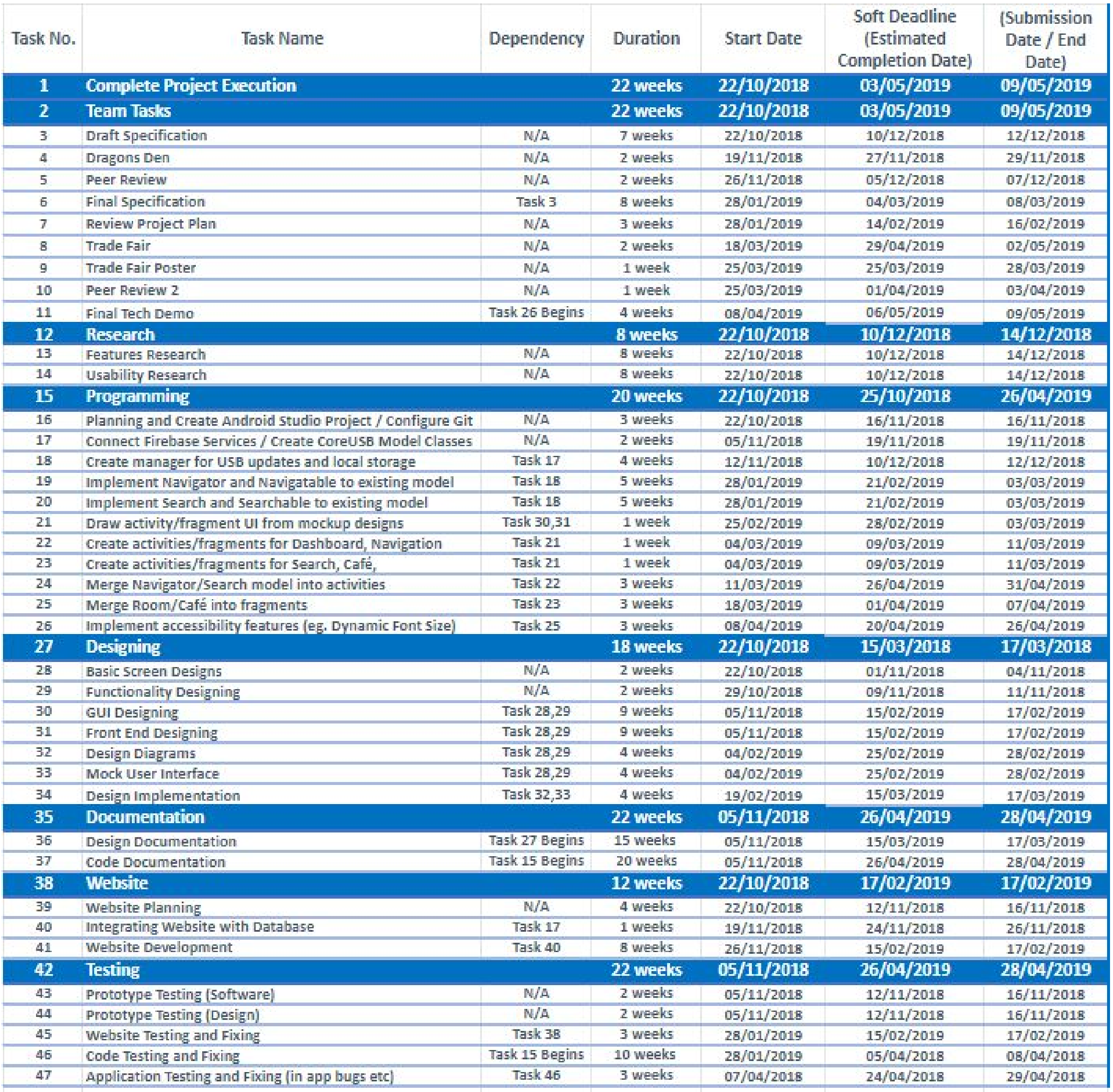
The Project Plan is a very useful tool as the team can get an overview of the entire project and see what tasks lie ahead. We have decided to utilise a Gantt Chart for the project plan as it allows us to see which tasks can be done in parallel and which tasks need to be done sequentially. This is essential for a long term project as it allows team members to get to work on tasks in the most efficient way possible to ensure that we use the time frame to the best of our ability.

The Project Plan has been separated into seven main sections where each sub group of the team can see what they need to do. There are also hard and soft deadline which are very useful as it gives us some leniency on completion dates and allows other members to look over tasks and polish them if need be to ensure all work is done to a high standard.

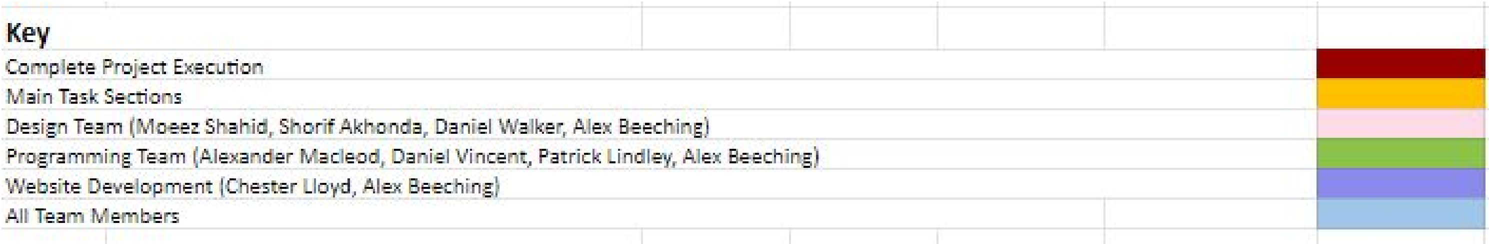
Dependencies have also been listed to show which tasks must take place before the current task can go ahead. For certain tasks the dependency is that another needs to have begun, meaning that they can be worked on simultaneously. Other tasks require various tasks to be fully complete before they can be started.

The Gantt Chart has also been colour coded with different colours for each sub section of the team. This is very useful firstly due to the fact that team members can look at the diagram and identify what work they need to be doing straight away. It can also be shown to stakeholders to give them a good idea on the course of the project throughout the year and where we expect to be at certain times.

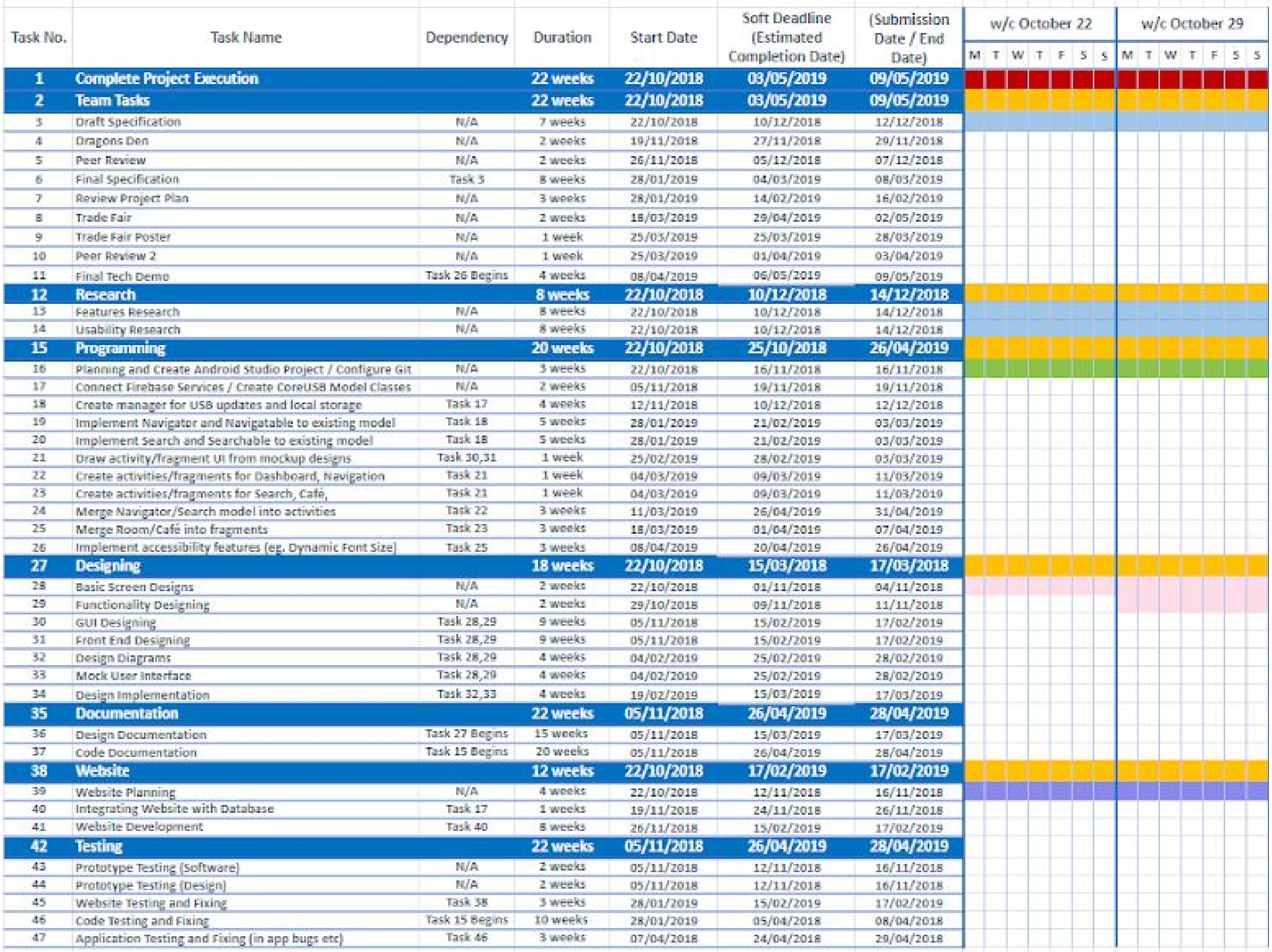
Please zoom in to get a detailed view of the Project Plan (a zoomed in view of the headings has been given below so that they are easier to read):



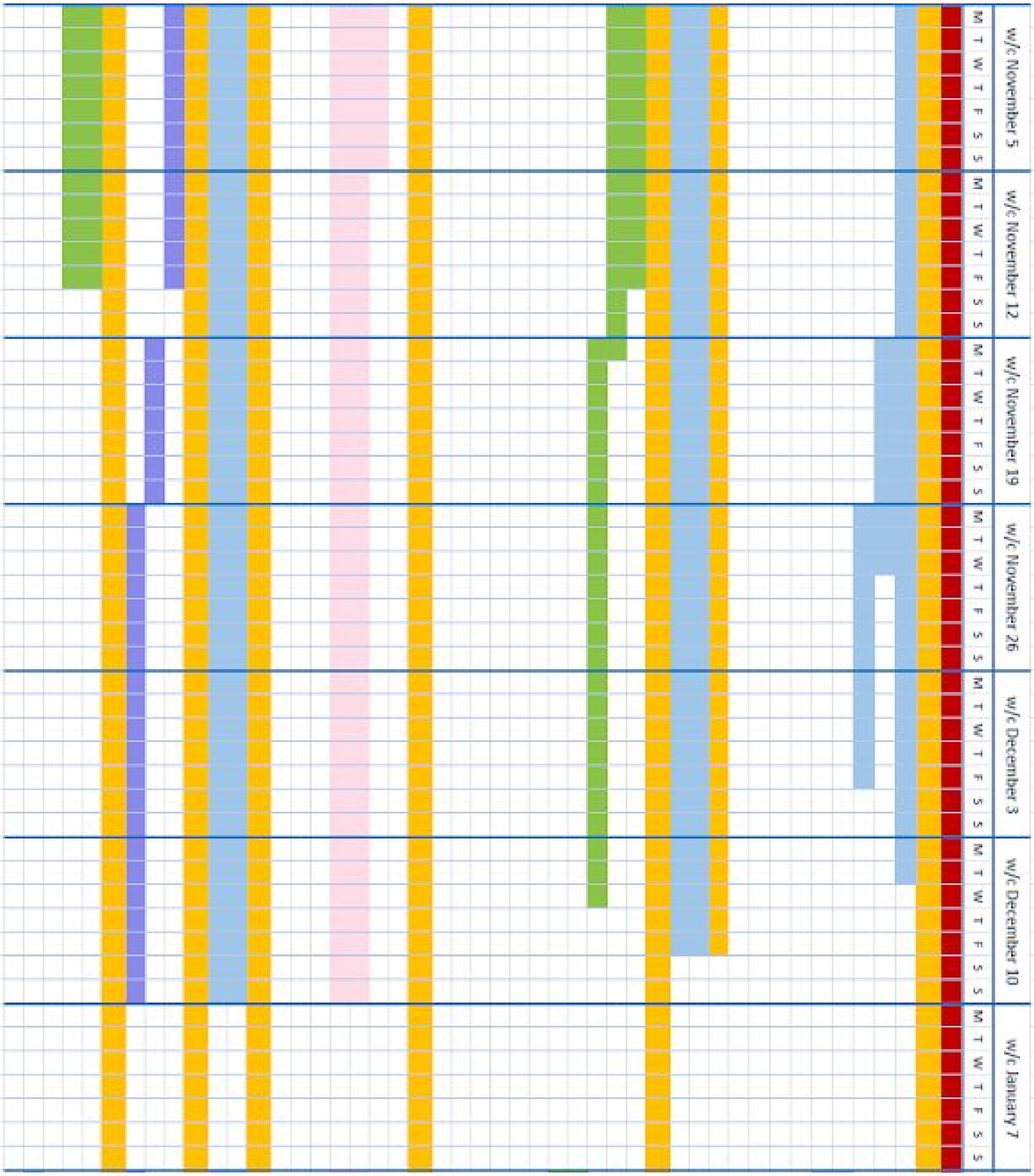
Task No.	Task Name	Dependency	Duration	Start Date	Soft Deadline (Estimated Completion Date)	(Submission Date / End Date)
1	**Complete Project Execution**		**22 weeks**	**22/10/2018**	**03/05/2019**	**09/05/2019**
2	**Team Tasks**		**22 weeks**	**22/10/2018**	**03/05/2019**	**09/05/2019**
3	Draft Specification	N/A	7 weeks	22/10/2018	10/12/2018	12/12/2018
4	Dragons Den	N/A	2 weeks	19/11/2018	27/11/2018	29/11/2018
5	Peer Review	N/A	2 weeks	26/11/2018	05/12/2018	07/12/2018
6	Final Specification	Task 3	8 weeks	28/01/2019	04/03/2019	08/03/2019
7	Review Project Plan	N/A	3 weeks	28/01/2019	14/02/2019	16/02/2019
8	Trade Fair	N/A	2 weeks	18/03/2019	29/04/2019	02/05/2019
9	Trade Fair Poster	N/A	1 week	25/03/2019	25/03/2019	28/03/2019
10	Peer Review 2	N/A	1 week	25/03/2019	01/04/2019	03/04/2019
11	Final Tech Demo	Task 26 Begins	4 weeks	08/04/2019	06/05/2019	09/05/2019
12	**Research**		**8 weeks**	**22/10/2018**	**10/12/2018**	**14/12/2018**
13	Features Research	N/A	8 weeks	22/10/2018	10/12/2018	14/12/2018
14	Usability Research	N/A	8 weeks	22/10/2018	10/12/2018	14/12/2018
15	**Programming**		**20 weeks**	**22/10/2018**	**25/10/2018**	**26/04/2019**
16	Planning and Create Android Studio Project / Configure Git	N/A	3 weeks	22/10/2018	16/11/2018	16/11/2018
17	Connect Firebase Services / Create CoreUSB Model Classes	N/A	2 weeks	05/11/2018	19/11/2018	19/11/2018
18	Create manager for USB updates and local storage	Task 17	4 weeks	12/11/2018	10/12/2018	12/12/2018
19	Implement Navigator and Navigatable to existing model	Task 18	5 weeks	28/01/2019	21/02/2019	03/03/2019
20	Implement Search and Searchable to existing model	Task 18	5 weeks	28/01/2019	21/02/2019	03/03/2019
21	Draw activity/fragment UI from mockup designs	Task 30,31	1 week	25/02/2019	28/02/2019	03/03/2019
22	Create activities/fragments for Dashboard, Navigation	Task 21	1 week	04/03/2019	09/03/2019	11/03/2019
23	Create activities/fragments for Search, Café,	Task 21	1 week	04/03/2019	09/03/2019	11/03/2019
24	Merge Navigator/Search model into activities	Task 22	3 weeks	11/03/2019	26/04/2019	31/04/2019
25	Merge Room/Café into fragments	Task 23	3 weeks	18/03/2019	01/04/2019	07/04/2019
26	Implement accessibility features (eg. Dynamic Font Size)	Task 25	3 weeks	08/04/2019	20/04/2019	26/04/2019
27	**Designing**		**18 weeks**	**22/10/2018**	**15/03/2018**	**17/03/2018**
28	Basic Screen Designs	N/A	2 weeks	22/10/2018	01/11/2018	04/11/2018
29	Functionality Designing	N/A	2 weeks	29/10/2018	09/11/2018	11/11/2018
30	GUI Designing	Task 28,29	9 weeks	05/11/2018	15/02/2019	17/02/2019
31	Front End Designing	Task 28,29	9 weeks	05/11/2018	15/02/2019	17/02/2019
32	Design Diagrams	Task 28,29	4 weeks	04/02/2019	25/02/2019	28/02/2019
33	Mock User Interface	Task 28,29	4 weeks	04/02/2019	25/02/2019	28/02/2019
34	Design Implementation	Task 32,33	4 weeks	19/02/2019	15/03/2019	17/03/2019
35	**Documentation**		**22 weeks**	**05/11/2018**	**26/04/2019**	**28/04/2019**
36	Design Documentation	Task 27 Begins	15 weeks	05/11/2018	15/03/2019	17/03/2019
37	Code Documentation	Task 15 Begins	20 weeks	05/11/2018	26/04/2019	28/04/2019
38	**Website**		**12 weeks**	**22/10/2018**	**17/02/2019**	**17/02/2019**
39	Website Planning	N/A	4 weeks	22/10/2018	12/11/2018	16/11/2018
40	Integrating Website with Database	Task 17	1 weeks	19/11/2018	24/11/2018	26/11/2018
41	Website Development	Task 40	8 weeks	26/11/2018	15/02/2019	17/02/2019
42	**Testing**		**22 weeks**	**05/11/2018**	**26/04/2019**	**28/04/2019**
43	Prototype Testing (Software)	N/A	2 weeks	05/11/2018	12/11/2018	16/11/2018
44	Prototype Testing (Design)	N/A	2 weeks	05/11/2018	12/11/2018	16/11/2018
45	Website Testing and Fixing	Task 38	3 weeks	28/01/2019	15/02/2019	17/02/2019
46	Code Testing and Fixing	Task 15 Begins	10 weeks	28/01/2019	05/04/2018	08/04/2018
47	Application Testing and Fixing (in app bugs etc)	Task 46	3 weeks	07/04/2018	24/04/2018	29/04/2018

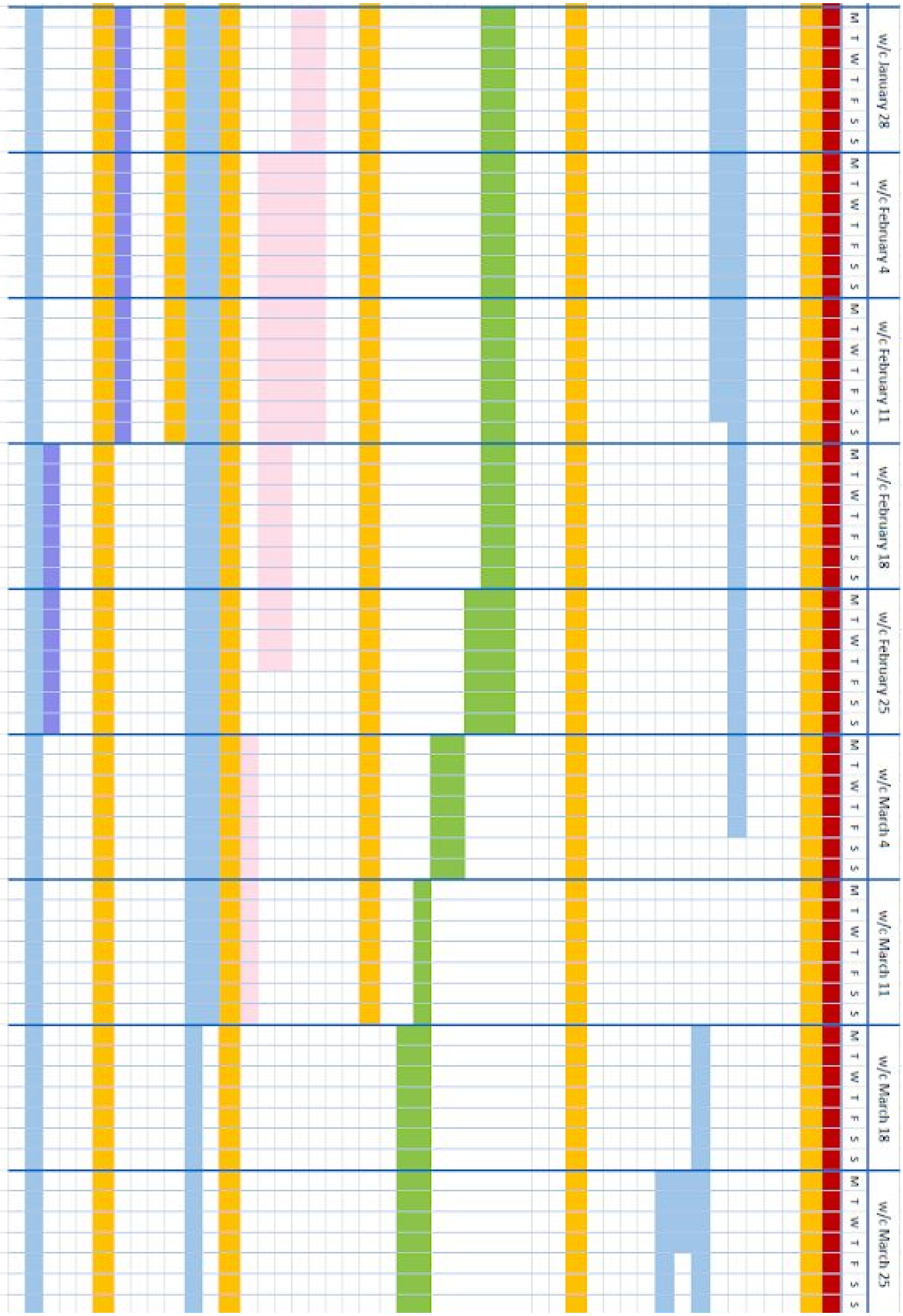


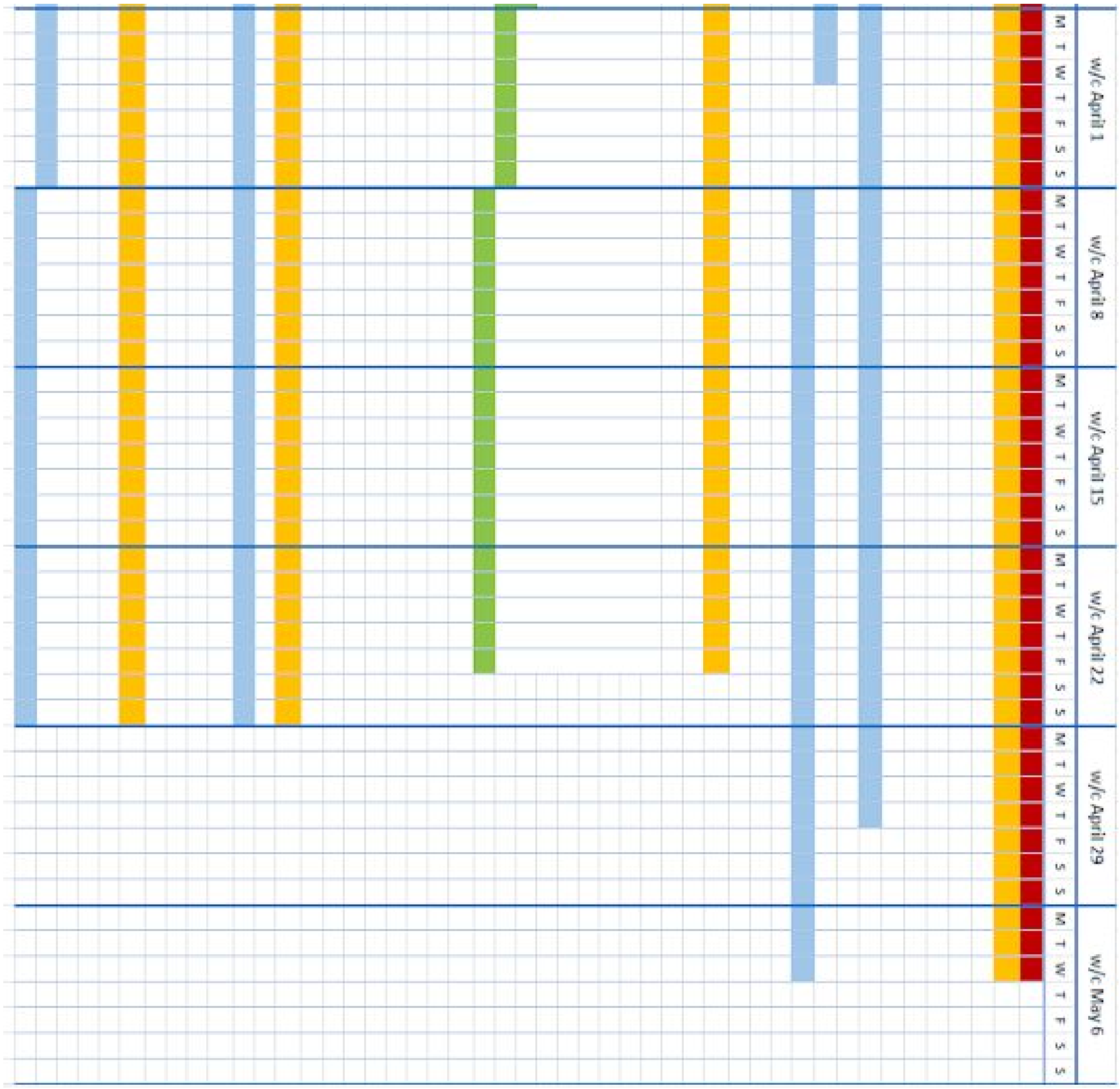
Key	
Complete Project Execution	🟥
Main Task Sections	🟧
Design Team (Moeez Shahid, Shorif Akhonda, Daniel Walker, Alex Beeching)	▨
Programming Team (Alexander Macleod, Daniel Vincent, Patrick Lindley, Alex Beeching)	🟩
Website Development (Chester Lloyd, Alex Beeching)	🟪
All Team Members	🟦



Task No.	Task Name	Dependency	Duration	Start Date	Soft Deadline (Estimated Completion Date)	(Submission Date / End Date)	w/c October 22							w/c October 29						
							M	T	W	T	F	S	S	M	T	W	T	F	S	S
1	**Complete Project Execution**		**22 weeks**	**22/10/2018**	**03/05/2019**	**09/05/2019**	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
2	**Team Tasks**		**22 weeks**	**22/10/2018**	**03/05/2019**	**09/05/2019**	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧
3	Draft Specification	N/A	7 weeks	22/10/2018	10/12/2018	12/12/2018	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
4	Dragons Den	N/A	2 weeks	19/11/2018	27/11/2018	29/11/2018														
5	Peer Review	N/A	2 weeks	26/11/2018	05/12/2018	07/12/2018														
6	Final Specification	Task 3	8 weeks	28/01/2019	04/03/2019	08/03/2019														
7	Review Project Plan	N/A	3 weeks	28/01/2019	14/02/2019	16/02/2019														
8	Trade Fair	N/A	2 weeks	18/03/2019	29/04/2019	02/05/2019														
9	Trade Fair Poster	N/A	1 week	25/03/2019	25/03/2019	28/03/2019														
10	Peer Review 2	N/A	1 week	25/03/2019	01/04/2019	03/04/2019														
11	Final Tech Demo	Task 26 Begins	4 weeks	08/04/2019	06/05/2019	09/05/2019														
12	**Research**		**8 weeks**	**22/10/2018**	**10/12/2018**	**14/12/2018**	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧
13	Features Research	N/A	8 weeks	22/10/2018	10/12/2018	14/12/2018	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
14	Usability Research	N/A	8 weeks	22/10/2018	10/12/2018	14/12/2018	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
15	**Programming**		**20 weeks**	**22/10/2018**	**25/10/2018**	**26/04/2019**	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧
16	Planning and Create Android Studio Project / Configure Git	N/A	3 weeks	22/10/2018	16/11/2018	16/11/2018	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
17	Connect Firebase Services / Create CoreUSB Model Classes	N/A	2 weeks	05/11/2018	19/11/2018	19/11/2018														
18	Create manager for USB updates and local storage	Task 17	4 weeks	12/11/2018	10/12/2018	12/12/2018														
19	Implement Navigator and Navigatable to existing model	Task 18	5 weeks	28/01/2019	21/02/2019	03/03/2019														
20	Implement Search and Searchable to existing model	Task 18	5 weeks	28/01/2019	21/02/2019	03/03/2019														
21	Draw activity/fragment UI from mockup designs	Task 30,31	1 week	25/02/2019	28/02/2019	03/03/2019														
22	Create activities/fragments for Dashboard, Navigation	Task 21	1 week	04/03/2019	09/03/2019	11/03/2019														
23	Create activities/fragments for Search, Café,	Task 21	1 week	04/03/2019	09/03/2019	11/03/2019														
24	Merge Navigator/Search model into activities	Task 22	3 weeks	11/03/2019	26/04/2019	31/04/2019														
25	Merge Room/Café into fragments	Task 23	3 weeks	18/03/2019	01/04/2019	07/04/2019														
26	Implement accessibility features (eg. Dynamic Font Size)	Task 25	3 weeks	08/04/2019	20/04/2019	26/04/2019														
27	**Designing**		**18 weeks**	**22/10/2018**	**15/03/2018**	**17/03/2018**	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧
28	Basic Screen Designs	N/A	2 weeks	22/10/2018	01/11/2018	04/11/2018	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
29	Functionality Designing	N/A	2 weeks	29/10/2018	09/11/2018	11/11/2018								▨	▨	▨	▨	▨	▨	▨
30	GUI Designing	Task 28,29	9 weeks	05/11/2018	15/02/2019	17/02/2019														
31	Front End Designing	Task 28,29	9 weeks	05/11/2018	15/02/2019	17/02/2019														
32	Design Diagrams	Task 28,29	4 weeks	04/02/2019	25/02/2019	28/02/2019														
33	Mock User Interface	Task 28,29	4 weeks	04/02/2019	25/02/2019	28/02/2019														
34	Design Implementation	Task 32,33	4 weeks	19/02/2019	15/03/2019	17/03/2019														
35	**Documentation**		**22 weeks**	**05/11/2018**	**26/04/2019**	**28/04/2019**														
36	Design Documentation	Task 27 Begins	15 weeks	05/11/2018	15/03/2019	17/03/2019														
37	Code Documentation	Task 15 Begins	20 weeks	05/11/2018	26/04/2019	28/04/2019														
38	**Website**		**12 weeks**	**22/10/2018**	**17/02/2019**	**17/02/2019**	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧	🟧
39	Website Planning	N/A	4 weeks	22/10/2018	12/11/2018	16/11/2018	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪
40	Integrating Website with Database	Task 17	1 weeks	19/11/2018	24/11/2018	26/11/2018														
41	Website Development	Task 40	8 weeks	26/11/2018	15/02/2019	17/02/2019														
42	**Testing**		**22 weeks**	**05/11/2018**	**26/04/2019**	**28/04/2019**														
43	Prototype Testing (Software)	N/A	2 weeks	05/11/2018	12/11/2018	16/11/2018														
44	Prototype Testing (Design)	N/A	2 weeks	05/11/2018	12/11/2018	16/11/2018														
45	Website Testing and Fixing	Task 38	3 weeks	28/01/2019	15/02/2019	17/02/2019														
46	Code Testing and Fixing	Task 15 Begins	10 weeks	28/01/2019	05/04/2018	08/04/2018														
47	Application Testing and Fixing (in app bugs etc)	Task 46	3 weeks	07/04/2018	24/04/2018	29/04/2018														


w/c November 5
M T W T F S S
w/c November 12
M T W T F S S
w/c November 19
M T W T F S S
w/c November 26
M T W T F S S
w/c December 3
M T W T F S S
w/c December 10
M T W T F S S
w/c January 7
M T W T F S S


w/c January 28
w/c February 4
w/c February 11
w/c February 18
w/c February 25
w/c March 4
w/c March 11
w/c March 18
w/c March 25
M T W T F S S


w/c April 1
M T W T F S S
w/c April 8
M T W T F S S
w/c April 15
M T W T F S S
w/c April 22
M T W T F S S
w/c April 29
M T W T F S S
w/c May 6
M T W T F S S

4. Hardware and Software Platforms

Software

Programming / Designing

- Android Studio - IDE for the programming and testing of our application
- Adobe XD - Program that will be used to design the GUI of our application
- Adobe Photoshop - Program that will be used to create graphics for the application
- GitHub - Will be used as version control for any app development source files
- Firebase - Used as the database for the project
- StarUML - Program that will be used for designing all UML diagrams

Web Development

- Atom - An open source text editor with git built in
- Affinity Designer - Graphic design software used to create website mockups
- GitLab - Will be used as version control for any website development source files

Work Storage

- Google Drive - Will be a storage point for some of our work

Communications

- Facebook Messenger - Used for communication between team and team leader announcements.
- Slack - For more specialised communication - channels dedicated to programming, design etc.
- Trello - Centralised location to show progress of tasks and display who is working on which task
- Outlook/ Gmail - Used for communications via email between team or with the client

Miscellaneous

- Google Chrome - Used to access the internet for resources to assist programming / designing
- TeamViewer 14 - Allows team members to assist each other via desktop sharing

Hardware

Programming / Designing

- Computers (at Urban Sciences Building) - Used for programming and designing our application
- Personal Laptops - Used for programming and designing our application when not at the USB

Application Usage

- Mobile Devices Running Android 7 and above - Will be used to test our applications functionality. This is the only minimum requirement of a device to run this app due to the app being very small in size and not demanding a large amount of resources.

Backing Up Data

- GitHub - Will be used to keep an up to date, complete history of software development progress

5. Definition of Terms

Admin Panel - This is an area in a website that allows select users to log in to view and make amendments to data

CSS - Cascading Style Sheets - This is a style sheet language used to describe the presentation of web document written in a markup language such as HTML

GUI - Graphical User Interface - A form of user interface that uses graphics and icons

HTML - Hypertext Markup Language - This is a markup language used to create web pages and web applications

Java - An object-oriented programming language

JavaScript - This is a high-level language used as a scripting language for web pages

JSON - JavaScript Object Notation - This is an open-standard file format used to transmit data between a web app and a server

mySQL - This is an open source relational database management system

PHP - Hypertext Preprocessor - This is a server-side scripting language for web development

SQL - Structured Query Language - This is the language used for relational database management

UML - Unified Modelling Language - A general purpose modelling language used to represent the design of a system

USB - Newcastle University Urban Sciences Building

VCS - Version Control System - This is a system that stores version history of files and the changes made as they are uploaded.

Gantt Chart - A chart in which a series of horizontal lines shows the amount of work done or production completed in certain periods of time and whether tasks can be done simultaneously or not.

6. Requirements

6.1. Functional Requirements

Code	Requirement	Priority (H, M, L)	Supplier Compliance (Full, partial or will not be delivered)	Supplier Comment
	Android Application			
FA.01	Ship with all required data to function with no further network connection	H		Building layout, lecturer rooms, Tour Guide functionality, Building navigation: accurate at time of application release
FA.02	Navigate between any two points within the USB	H		Turn-by-turn navigation, overview of trip display
FA.03	Provide a guided tour throughout the USB	H		Lead user around landmarks following guided tour offered at reception
FA.04	Search for lecturers, rooms, facilities	H		Unified search - display most relevant results

FA.05	List available computers/printers in each room	H		Update live over network if available
FA.06	Alert users of out of hours access - Opening/Closing times	M		Set through website
FA.07	Display and navigate to Fire Exits	H		Hard coded to ensure functionality, but notes can be added through website if an exit is under maintenance for example
FA.08	Request any modified data from the server upon launch	M		The mobile app must download an updated copy of data stored on a remote server when it launches
FA.09	Offer Cafe menu (updated daily)	M		Downloaded from server at launch
FA.11	Calculate ETA for a journey from room to room	M		Offer an estimated time of arrival when navigation from one room in the Urban Sciences Building to another

FA.12	Offer various accessibility features	H		Features Include but are not limited to, text size adjustment, colourblind mode, offering alternative routes to points in the Urban Sciences Building when stairs are not accessible to the user
	Website			
FW.01	Website will update data on server	H		This data is what the client can modify via the admin panel.
FW.02	Allow Cafe staff to update daily menu	M		
FW.03	Provide links to download / open application	M		Use custom URI

6.2. Non-Functional Requirements

Code	Requirement	Priority (H, M, L)	Supplier Compliance (Full, partial or will not be delivered)	Supplier Comment
NF.01	Documentation for Application	H		Full documentation of features, usage and implementation
NF.02	Documentation for Website	H		Full documentation of features, usage and implementation
NF.03	Use Urban Sciences Building colour scheme	H		Requested by client

7. Other considerations

7.1 Assumptions

Some aspects of the specification aren't explicitly clear, and therefore had to be assumed when designing our solution. These can be categorized into application, user and research specific. As of now, our project assumptions are:

Application Specific

- Directions around the building are fixed, i.e. building architecture or publically accessible areas will not change. We will assume other locatable objects such as printers and vending machines will also never change location.
- The tour only needs to cover publically accessible areas of the building, i.e. not cover staff offices.
- Live computer usage information received from the university is accurate.

Research Specific

- With 70% of the market dominated by android, this will reflect onto our target audience and support our decision to make an android app over iOS or any other platform.

User Specific

- People with accessibility needs will actually use their respective features when using the app.
- All data amended through the admin page of our website will be correct and up to date.

7.2 Constraints and dependencies

Some factors that our application depends on included the user having a stable internet connection whilst our app is in use. This is required for in-app features such as computer availability, which will be updated in real time, so without a stable connection the latest information will not be received. This principle applies to loading our website itself and will be needed to update information through the admin page. In addition, internet connectivity will be fundamental in keeping the servers required for this online.

8. Software Design

8.1 Overview of what the software will do and not do.

The website will serve two purposes in this project; the first is as a marketing tool to show what the mobile app will be able to do and the second will allow amendments to be made to some of the app's data via an admin panel.

Some data displayed on the app may change from time to time, so there will be a way for the client, or another designated user, to adjust this data when required. There will be an admin area on the website that will require an account to gain access. These credentials are independent to those stored by the university and will be handled by Firebase Authentication. This will allow authorised users to sign in with an email address and password. Multiple email addresses can be added to Firebase if multiple admins are required. Data that can be changed via this method include: the building opening times (including out of hours access) (FA.06), the café menu (FA.09) and the staff offices.

A Google Firestore database will be used to store the amendable data on the web server. When a user accesses a page of data to amend, the opening times for example, the web interface will load a page with the values currently stored in the database. Client-side scripting will be used to prevent invalid inputs and alert the user of any incorrect values. When the mobile application is launched, a check will be made, comparing the local database version to the version stored on Firestore. Next, the user will be prompted to update the local database to the most up to date version (FA.08). In this way, users will receive any changes made through the website by administrators when the application is launched.

The Android application will provide users with live information such as opening times and available computers throughout the Urban Sciences Building (FA.05), the latter being downloaded from the Newcastle University IT services' online dashboard for viewing in the application, polled and updated at regular intervals.

A unified search (FA.04) will enable users to search for lecturers, rooms and facilities such as toilets, from a single search field, with filters for these different categories. Any searched for item will have the option to be navigated to using the navigation feature (FA.02) using turn-by-turn directions.

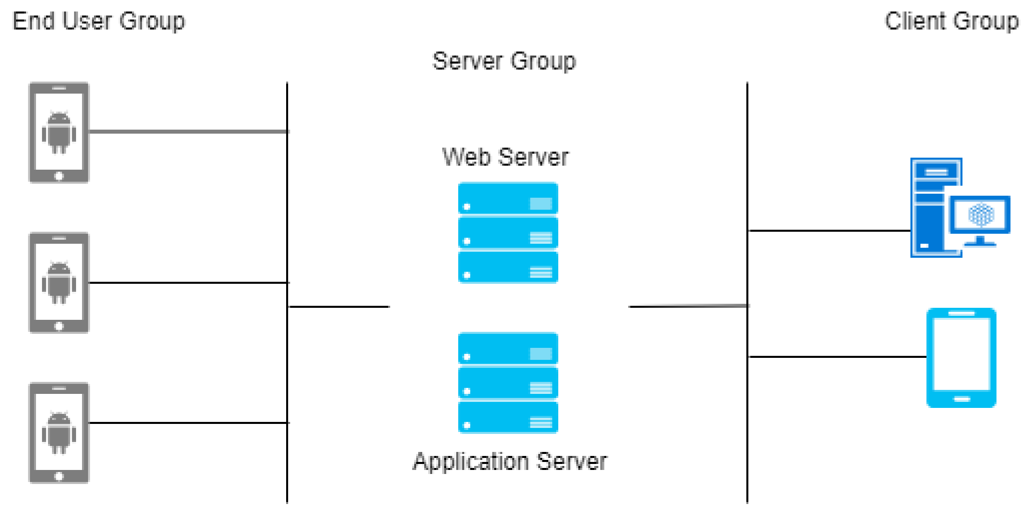
There are a couple of features that the application will not be capable of doing. For example, the navigation will not be capable of real-time navigation using the users' physical location. This would be achieved using GPS technologies, but due to the small scale of the system it is not feasible to implement currently.

The unified search will not be capable of searching online for any extra information that the user may want. This is due to the search being implemented only on the data cached by the application upon start-up. To be capable of an online search the application would require a constant internet connection, which is something that the client was insistent on the application not needing (FA.01).

8.2 System Architecture

There will be three main groups within the system being developed:

- End user group
- Server group
- Client group



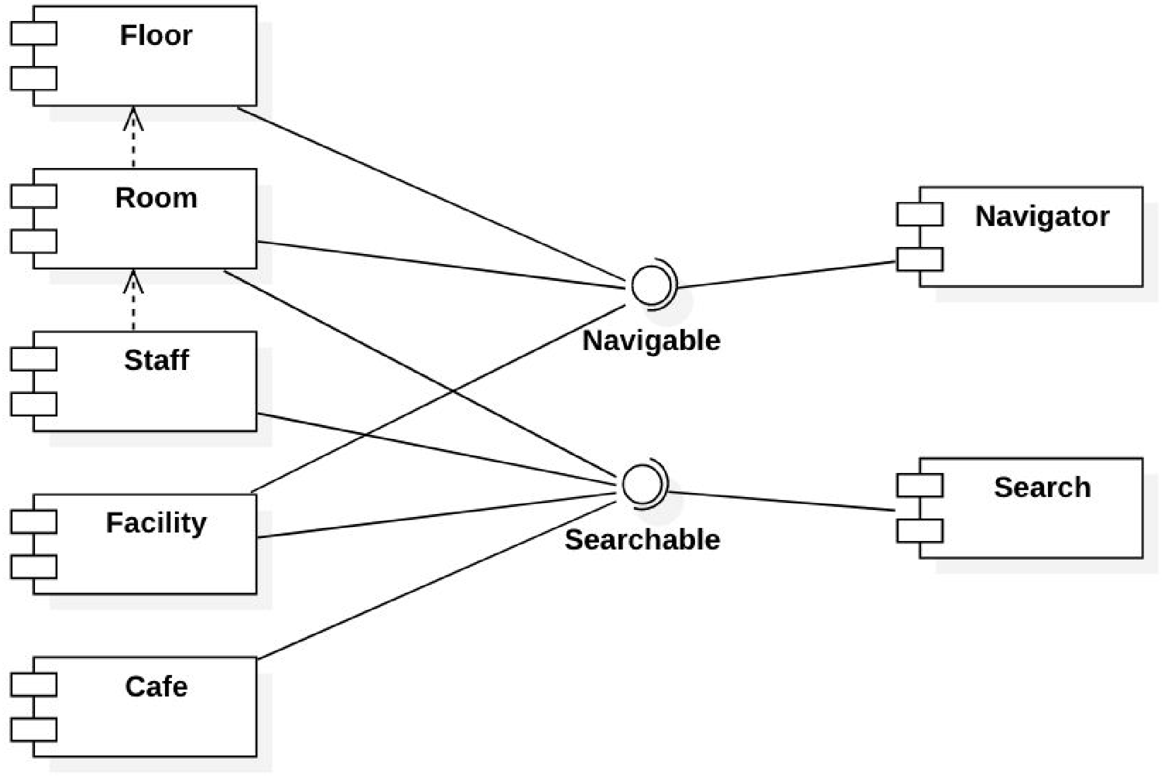
End user group: This is the application on android phones. This is the main focus of the project, with all major functionality being built into the app to reduce the need for cross-platform communication. The app will store all required data so it does not need to communicate with the server group in real time (FA.01). It will however contact the server group on startup, which will update the cached data if required (FA.08). This information will be received in the form of a read-only JSON file, which will then be cached internally by the application. Each device contacts the application server using communication protocols (WiFi). It was decided that data will be stored locally on the device as it allows users to make use of the application without having an internet connection.

Server group: This is the web server and app server. Acting as an intermediary between the client and end user group, the server group will hold all information needed to update the application if needed. This information will be stored within a database, which is accessible through the website in order to make changes (FW.01). It will also host the website, which is the method of updating the application. Firebase was decided as the database of choice due to a team member having previous experience with the technology and its extensive documentation for Android integration. The decision to use a website to update the application server data was based on another team member having experience with web technologies, as well as ease of use for the client group.

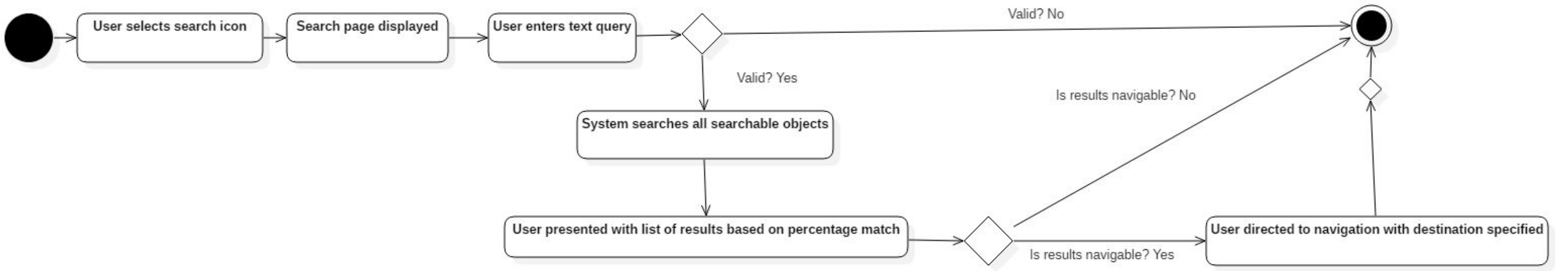
Client group: The client group is any device that the client uses to log into the admin area of the website. This admin page is behind a secure login. Within the admin page, the client can send changes to opening times, staff room lists and cafe menu items (FW.01, FW.02). These changes will be pushed to the application server, which will notify the end user group upon startup if an update is available. The amendable data fields were decided based upon the fact that these are the most likely to change. For example, room numbers cannot be changed, so therefore are not included in the website.

8.3 High Level Overview

Component Diagram

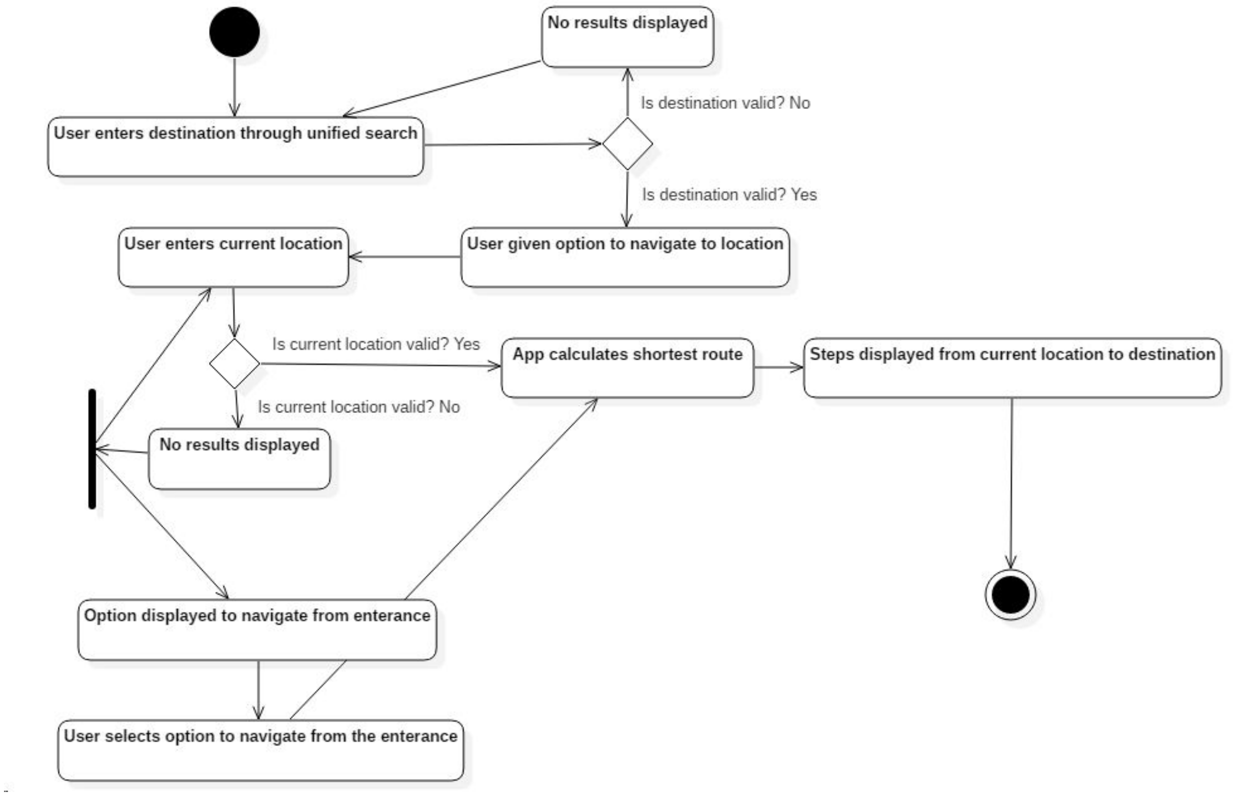


Search Activity Diagram


User selects search icon
Search page displayed
User enters text query
Valid? No
Valid? Yes
System searches all searchable objects
User presented with list of results based on percentage match
Is results navigable? No
Is results navigable? Yes
User directed to navigation with destination specified

This activity allows the user to search for searchable objects within the app. The user must first select the search icon and enter a query into the text field, where (if valid) a list of relevant results will be displayed; prioritised by percentage match to the initial query. If any results are navigable (i.e. a room), when selected the user will be redirected to the navigation feature of the app with their selected room specified.

Activity: Navigation


No results displayed
Is destination valid? No
User enters destination through unified search
Is destination valid? Yes
User enters current location
User given option to navigate to location
Is current location valid? Yes
App calculates shortest route
Steps displayed from current location to destination
Is current location valid? No
No results displayed
Option displayed to navigate from enterance
User selects option to navigate from the enterance

This activity allows the user to navigate around the Urban Sciences Building when a destination is entered through the unified search and (optionally) a current location is entered. In order for directions to be displayed the query must be valid (i.e. the room must exist). If the user does not enter a current location, or what they enter is invalid, they will be presented with an option to start their navigation from a default location.

8.4 Test Plan

The test plan documents our strategy to verify and ensure that the application meets the design specification and other requirements enforced by the client.

Over the course of the product’s development, JUnit test classes will be created in order to monitor the progress of each unit or components as the software is designed. The programming team will ensure that each of their components have corresponding test classes named after their class’s fixture.

It is important for members from outside of the programming team to also test the application. Having had less contact time with the application, these members may interact with the program in a way which the programming team may not have expected. Thus, the application may need to be adjusted in order to improve the workflow for the end user.

In order the test the application, we will use a combination of the Android Emulator, part of the Android Studio IDE, as well as using physical devices. By testing the application on physical devices, we can experience varying network speeds, screen bounds and user defined operating system settings that we may not encounter with the emulator.

Having successfully passed all tests created in the JUnit test classes, the application will be considered ready to be tested by non-programming members. The table below outlines the test plan for these members to be carried out on a mixture of Android emulators and physical devices using a variety of device settings, operating systems and network conditions.

Test	Activity	Test Description	Expected Result
1.1.0	LaunchActivity	Application Launch	Application launches successfully
1.1.1	LaunchActivity	USB First Install	Dialog appears to download USB data and proceeds to do so successfully
1.1.2	LaunchActivity	USB No Internet First Install	Dialog appears to download USB data and proceeds to alert user an active internet connection is required
1.1.3	LaunchActivity	USB Update Install	Dialog appears to update USB data and proceeds to do so successfully

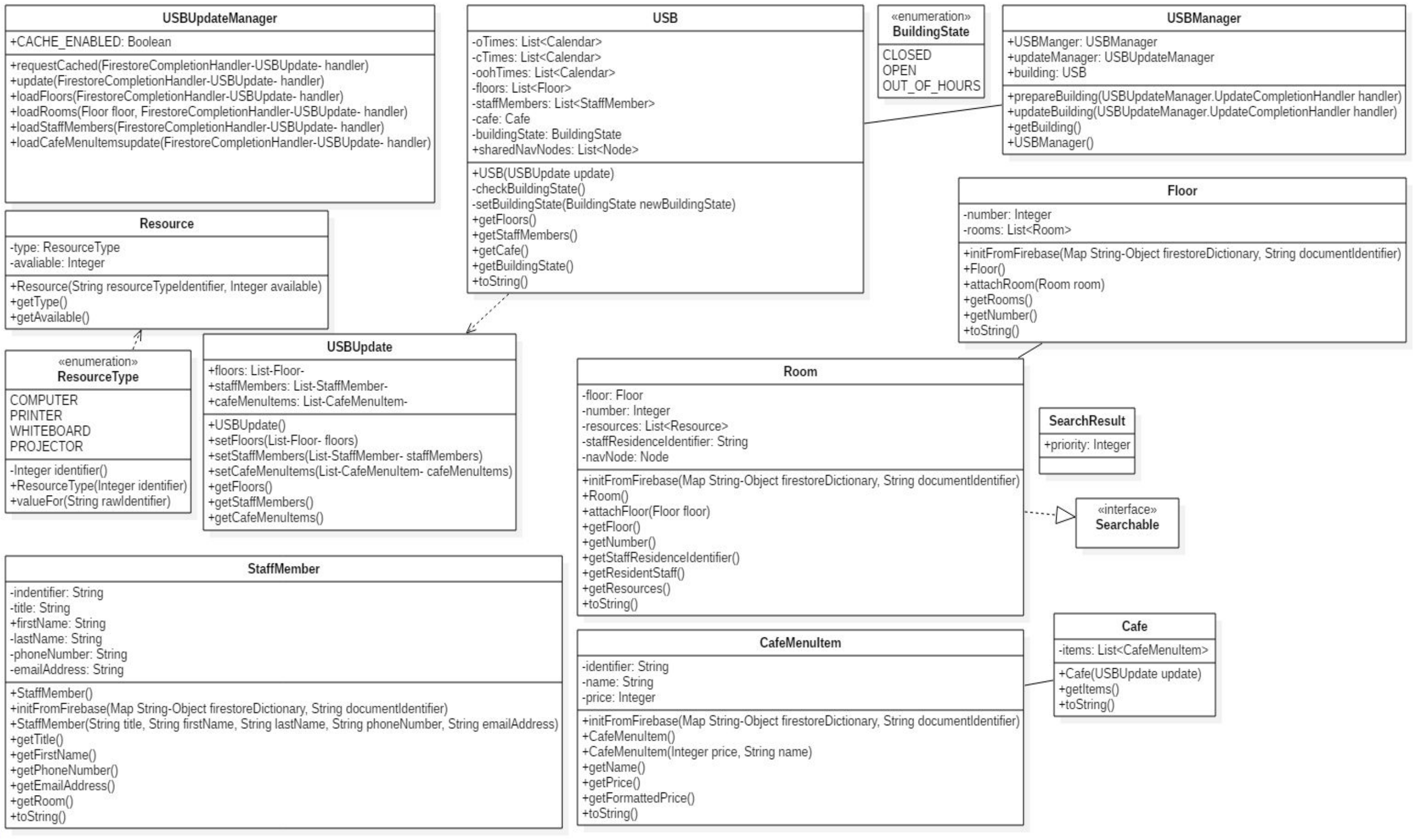
1.1.4	LaunchActivity	USB Update No Internet	USB data is loaded from cache and app successfully launches to dashboard
1.1.5	LaunchActivity	Attempt To Cancel Dialog	USB Update dialogs should not be able to be cancelled
1.2.0	DashboardFragment	View Computer Clusters	Available computers clusters are displayed
1.2.1	DashboardFragment	View Opening Hours	Opening hours are displayed stating whether the Urban Sciences Building is open depending on device time
1.2.2	DashboardFragment	Launch Tour	Tapping the tour button launches the Urban Sciences Building tour
1.3.0	CafeMenuFragment	View Opening Hours	Opening hours are displayed stating whether café is open depending on device time
1.3.1	CafeMenuFragment	View Café Categories	Cafè menu categories are displayed ranging from food to drink
1.3.2	CafeMenuFragment	View Café Item	The name, price, size and subcategory of the menu item is displayed
1.4.0	SearchActivity	Search Room	The results for the room search are displayed
1.4.1	SearchActivity	Search Staff Member	The results for the staff member, including room, is displayed
1.4.2	SearchActivity	Search Café Menu Item	The results for the café menu item are displayed
1.4.3	SearchActivity	Search Resource	The results for the resources are displayed
1.4.4	SearchActivity	Search Unrecognised Item	No results text is displayed when no matching items could be found
1.4.5	SearchActivity	Open Unified Search	The search activity is opened from all fabs in activities conforming to

			USBActivtity
1.5.0	RoomActivity	View Room Details	The details, including its room resources, are displayed
1.5.1	RoomActivity	View Room Staff Residence	The resident staff member is displayed
1.6.0	NavigationActivity	Get Directions Between Two Rooms	Directions between two rooms are displayed
1.6.1	NavigationActivity	Get Reverse Directions Between Two Rooms	Directions between two rooms are displayed
1.6.2	NavigationActivity	Get Accessible Directions	With accessibility mode turned on, directions to a location which require changing floors will use the lift
1.7.0	SettingsActivity	Accessibility Mode	Toggling setting enables accessibility features
1.8.0	NavigationDrawerActivity	Navigate To Fragment	Selecting option navigates to correct fragment
1.8.1	NavigationDrawerActivity	Cancel Navigation	Tapping/Swiping away from navigation drawer cancels navigation
1.9.0	TourActivity	Calculates Navigation For Tour	The tour is accompanied with navigation around the Urban Sciences Building to each point of interest
1.9.1	TourActivity	Detail For Each Point Of Interest	A description is displayed for each point of interest on the Urban Sciences Building Tour
1.9.2	TourActivity	Navigating To Next Point On Tour	The next button will display navigation to the subsequent point of interest on the Urban Sciences Building Tour

8.5 Package & Class Diagrams

CoreUSB Package

This package contains the classes which define the foundations of the Urban Sciences Building.


USBUpdateManager
+CACHE_ENABLED: Boolean
+requestCached(FirestoreCompletionHandler-USBUpdate- handler)
+update(FirestoreCompletionHandler-USBUpdate- handler)
+loadFloors(FirestoreCompletionHandler-USBUpdate- handler)
+loadRooms(Floor floor, FirestoreCompletionHandler-USBUpdate- handler)
+loadStaffMembers(FirestoreCompletionHandler-USBUpdate- handler)
+loadCafeMenuItemsupdate(FirestoreCompletionHandler-USBUpdate- handler)
USB
-oTimes: List<Calendar>
-cTimes: List<Calendar>
-oohTimes: List<Calendar>
-floors: List<Floor>
-staffMembers: List<StaffMember>
-cafe: Cafe
-buildingState: BuildingState
+sharedNavNodes: List<Node>
+USB(USBUpdate update)
-checkBuildingState()
-setBuildingState(BuildingState newBuildingState)
+getFloors()
+getStaffMembers()
+getCafe()
+getBuildingState()
+toString()
«enumeration»
BuildingState
CLOSED
OPEN
OUT_OF_HOURS
USBManager
+USBManger: USBManager
+updateManager: USBUpdateManager
+building: USB
+prepareBuilding(USBUpdateManager.UpdateCompletionHandler handler)
+updateBuilding(USBUpdateManager.UpdateCompletionHandler handler)
+getBuilding()
+USBManager()
Resource
-type: ResourceType
-avaliable: Integer
+Resource(String resourceTypeIdentifier, Integer available)
+getType()
+getAvailable()
Floor
-number: Integer
-rooms: List<Room>
+initFromFirebase(Map String-Object firestoreDictionary, String documentIdentifier)
+Floor()
+attachRoom(Room room)
+getRooms()
+getNumber()
+toString()
«enumeration»
ResourceType
COMPUTER
PRINTER
WHITEBOARD
PROJECTOR
-Integer identifier()
+ResourceType(Integer identifier)
+valueFor(String rawIdentifier)
USBUpdate
+floors: List-Floor-
+staffMembers: List-StaffMember-
+cafeMenuItems: List-CafeMenuItem-
+USBUpdate()
+setFloors(List-Floor- floors)
+setStaffMembers(List-StaffMember- staffMembers)
+setCafeMenuItems(List-CafeMenuItem- cafeMenuItems)
+getFloors()
+getStaffMembers()
+getCafeMenuItems()
Room
-floor: Floor
-number: Integer
-resources: List<Resource>
-staffResidenceIdentifier: String
-navNode: Node
+initFromFirebase(Map String-Object firestoreDictionary, String documentIdentifier)
+Room()
+attachFloor(Floor floor)
+getFloor()
+getNumber()
+getStaffResidenceIdentifier()
+getResidentStaff()
+getResources()
+toString()
SearchResult
+priority: Integer
«interface»
Searchable
StaffMember
-indentifier: String
-title: String
+firstName: String
-lastName: String
-phoneNumber: String
-emailAddress: String
+StaffMember()
+initFromFirebase(Map String-Object firestoreDictionary, String documentIdentifier)
+StaffMember(String title, String firstName, String lastName, String phoneNumber, String emailAddress)
+getTitle()
+getFirstName()
+getPhoneNumber()
+getEmailAddress()
+getRoom()
+toString()
CafeMenuItem
-identifier: String
-name: String
-price: Integer
+initFromFirebase(Map String-Object firestoreDictionary, String documentIdentifier)
+CafeMenuItem()
+CafeMenuItem(Integer price, String name)
+getName()
+getPrice()
+getFormattedPrice()
+toString()
Cafe
-items: List<CafeMenuItem>
+Cafe(USBUpdate update)
+getItems()
+toString()

CoreUSB.Room

Represents a room within the Urban Sciences Building. A room can be selected as either the origin or the destination of the Urban Sciences Building navigation route. A room holds various user displayable metadata including but not limited to, the number of computers, whether the room houses printers, or whether the room contains a projector. A user is able to search for a room in order to quickly check whether it contains a facility. A room is constructed using read-only data from Firebase.

CoreUSB.Floor

Represents a floor within the Urban Sciences Building. A floor holds a collection of rooms, facilities, and other notable services found on its level. Although a floor cannot be navigated directly to, it is an important object to help navigate a user from room to room. A floor holds various user displayable metadata including but not limited to, the level number or a client defined floor colour. A user is able to search for a floor in order to determine where the level contains a particular facility such as toilets. A floor is constructed using read-only data from Firebase.

CoreUSB.Resource

A class which represents a resource within the Urban Sciences Building. A resource can be one of a variety of types such as a toilet, vending machine or meeting space. A resource can be navigated to using the Urban Sciences Building journey planner; however, care must be taken when selecting the facility as your origin or destination as multiple floors can contain the same facility type. A resource holds various user displayable metadata which varies depending on the facility type. For example, a toilet facility will include information on its gender where as a meeting space will have details on the number of seats available. A resource is constructed using read-only data from Firebase.

CoreUSB.ResourceType

An enum which defines the type of resource in the Urban Sciences Building. A resource is constructed using read-only data from Firebase.

CoreUSB.CafeMenuItem

A class which represents a purchasable item at the café in the Urban Sciences Building. A café menu item holds various user displayable metadata including its name and price. A café menu item is constructed using data from Firebase which is managed by the application’s website component.

CoreUSB.StaffMember

A class which represents a tutor or lecturer who is active in the Urban Sciences Building. A staff member holds various user displayable metadata including but not limited to, the member’s name, email address or phone number. A staff member can be searched by querying its metadata so that the user can easily match contact details; for example, a phone number can be used to return the name of the individual. A staff member is constructed using data from Firebase which is managed by the application’s website component.

CoreUSB.USB

A class which represents the current instance of the Urban Sciences Building.

CoreUSB.USBManager

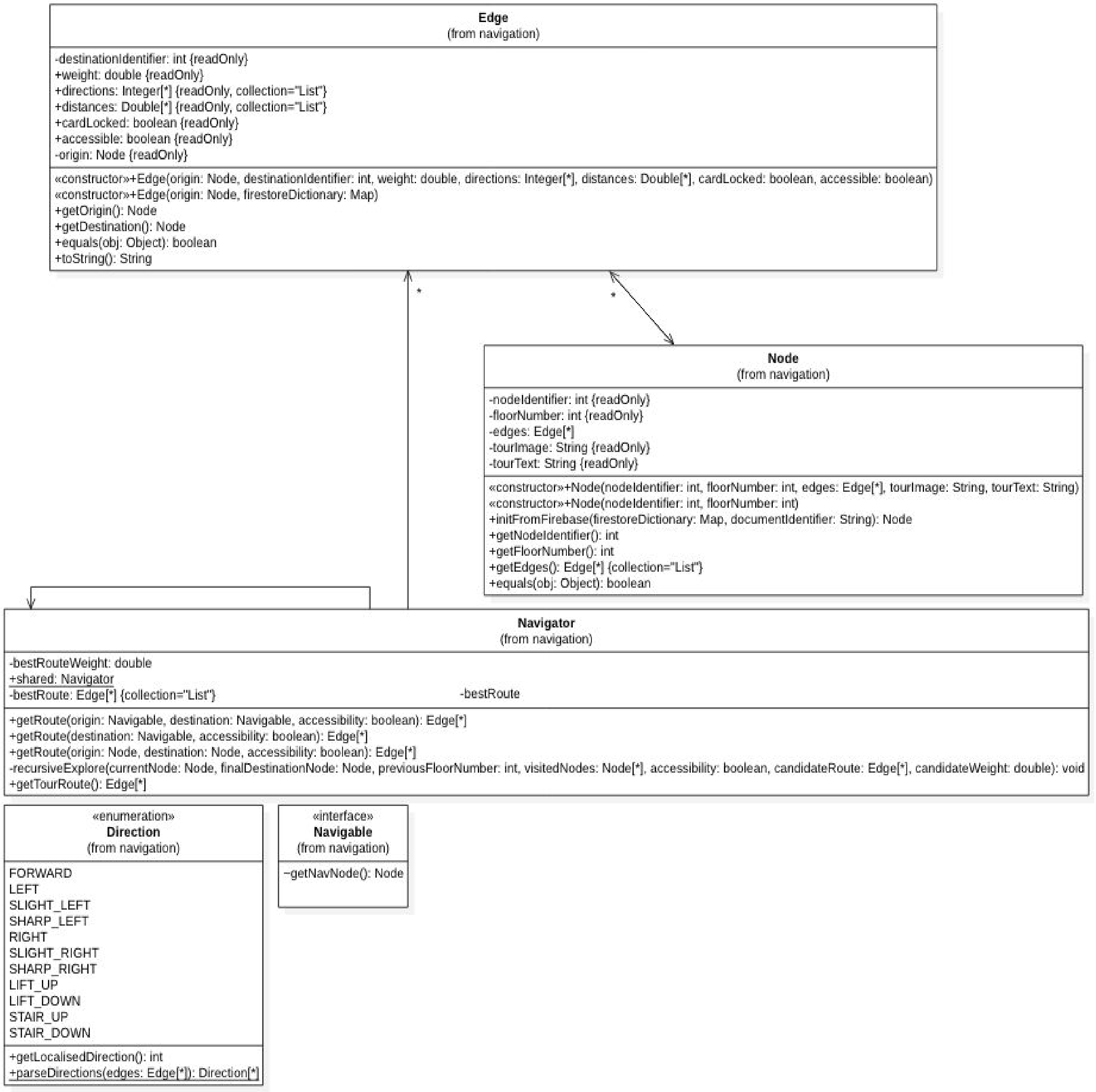
A class which manages the current version, caching and updates of the Urban Sciences Building.

CoreUSB.USBUpdateManager

A class which manages and maintains new and cached versions of the Urban Sciences Building.

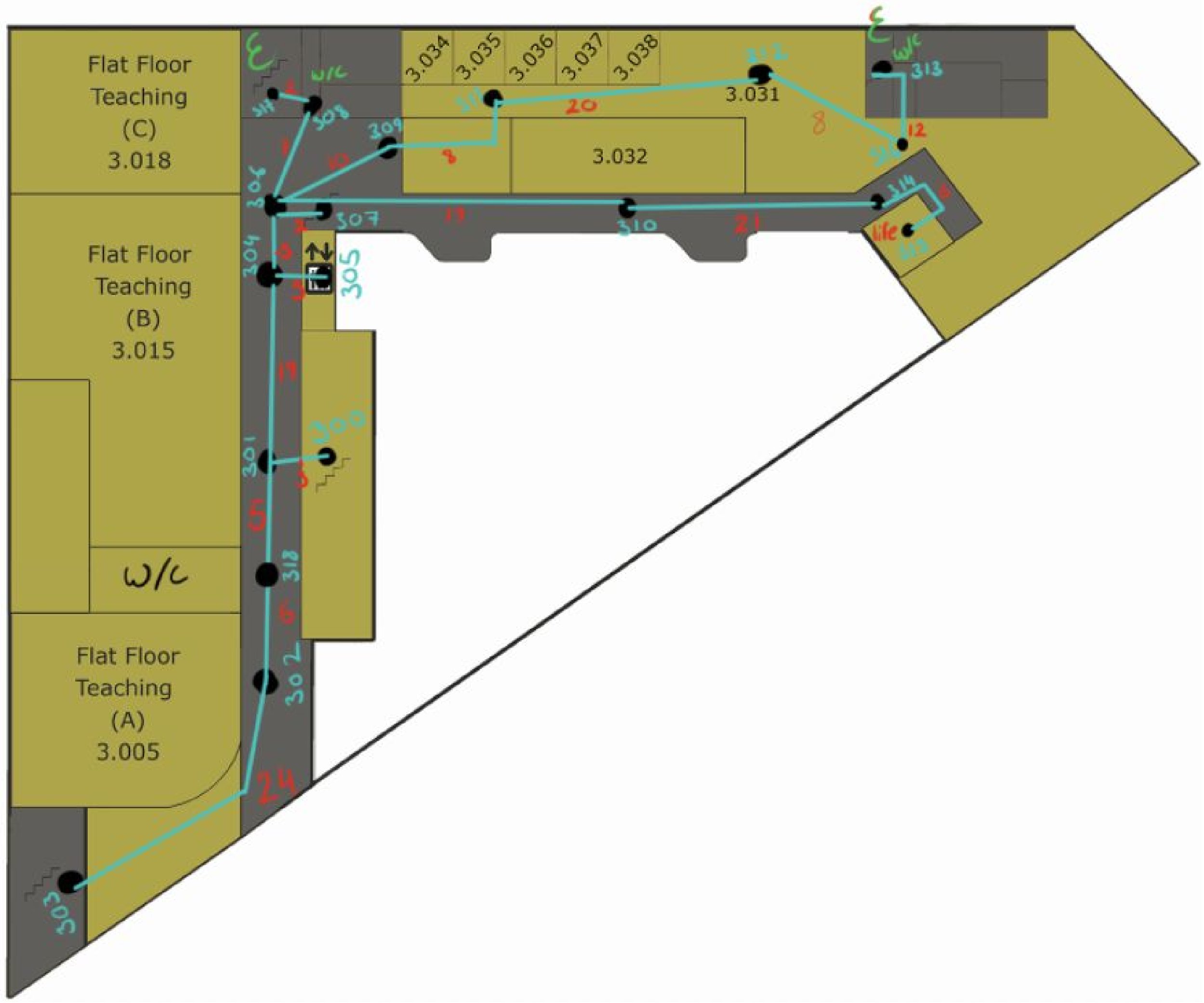
Navigation Package

A package which contains the model to manage navigation throughout the Urban Sciences Building. (Fulfills FA.02, FA.11, FA.12)


Edge
(from navigation)
-destinationIdentifier: int {readOnly}
+weight: double {readOnly}
+directions: Integer[*] {readOnly, collection="List"}
+distances: Double[*] {readOnly, collection="List"}
+cardLocked: boolean {readOnly}
+accessible: boolean {readOnly}
-origin: Node {readOnly}
«constructor»+Edge(origin: Node, destinationIdentifier: int, weight: double, directions: Integer[*], distances: Double[*], cardLocked: boolean, accessible: boolean)
«constructor»+Edge(origin: Node, firestoreDictionary: Map)
+getOrigin(): Node
+getDestination(): Node
+equals(obj: Object): boolean
+toString(): String
*
*
Node
(from navigation)
-nodeIdentifier: int {readOnly}
-floorNumber: int {readOnly}
-edges: Edge[*]
-tourImage: String {readOnly}
-tourText: String {readOnly}
«constructor»+Node(nodeIdentifier: int, floorNumber: int, edges: Edge[*], tourImage: String, tourText: String)
«constructor»+Node(nodeIdentifier: int, floorNumber: int)
+initFromFirebase(firestoreDictionary: Map, documentIdentifier: String): Node
+getNodeIdentifier(): int
+getFloorNumber(): int
+getEdges(): Edge[*] {collection="List"}
+equals(obj: Object): boolean
Navigator
(from navigation)
-bestRouteWeight: double
+shared: Navigator
-bestRoute: Edge[*] {collection="List"}
-bestRoute
+getRoute(origin: Navigable, destination: Navigable, accessibility: boolean): Edge[*]
+getRoute(destination: Navigable, accessibility: boolean): Edge[*]
+getRoute(origin: Node, destination: Node, accessibility: boolean): Edge[*]
-recursiveExplore(currentNode: Node, finalDestinationNode: Node, previousFloorNumber: int, visitedNodes: Node[*], accessibility: boolean, candidateRoute: Edge[*], candidateWeight: double): void
+getTourRoute(): Edge[*]
«enumeration»
Direction
(from navigation)
FORWARD
LEFT
SLIGHT_LEFT
SHARP_LEFT
RIGHT
SLIGHT_RIGHT
SHARP_RIGHT
LIFT_UP
LIFT_DOWN
STAIR_UP
STAIR_DOWN
+getLocalisedDirection(): int
+parseDirections(edges: Edge[*]): Direction[*]
«interface»
Navigable
(from navigation)
~getNavNode(): Node

Navigation.Node

A class which represents a “node” used for planning routes throughout the Urban Sciences Building. Each floor of the Urban Sciences Building is mapped with multiple nodes in key locations and navigation is planned by moving from one node to the next until the destination is reached using the shortest total distances between all nodes in the route. Furthermore, when the virtual tour portion of the application is used (FA.03), nodes’ “tourImage” and “tourText” fields are used to provide information and images for key nodes along the tour route. Here is the map created to use as a reference for where nodes are placed in on floor 3.



Each number in blue next to a black circle represents a node, with blue lines representing edges (discussed next) and red numbers representing the distance the edge the number is next to covers. Each node has a list of edges connecting it to other nodes. In this way, the navigator class (discussed later) can traverse nodes using a recursive backtracking algorithm to find the shortest route between any two nodes. Green “E”s mark emergency exits.

Navigation.Edge

A class which represents a link between two nodes in the Urban Sciences Building. A route built by the navigator class (discussed next) a list of the edges which should be traversed from the start node to reach the destination node in the shortest possible way. Each edge has an origin and destination node, as well as a list of directions, distances and an overall weight value for calculating the shortest route.

Navigation.Navigator

A class used to find the shortest route between two “navigable” objects (discussed next), a floor number and a “navigable” object, or two nodes (FA.02). Given any two of these parameters, a recursive backtracking algorithm (demonstrated below in pseudo-code) will be used to find the shortest overall route (FA.11 - route length is known, so eta can be calculated), while taking options such as only allowing “accessible” (wheelchair access) (FA.12) routes to be provided to users. When the virtual tour portion of the application is used (FA.03),

```
Algorithm recursiveExplore

Inputs    currentNode, finalDestinationNode : Node; previousFloorNumber : Integer (0 when first instance
          is created); visitedNodes : List<Node> (empty list when first instance is created);
          accessibility: boolean; candidateRoute: List<Edge> (empty list when first instance is
          created); candidateWeight : Double (0 when first instance is created);

Variables bestRoute : List<Edge> (class variable - static across instances of “recursiveExplore”);
          bestRouteWeight : Double (class variable - static across instances of “recursiveExplore”);
          visitedNodes : List<Node>;

Returns   bestRoute;

Begin
          /* Add current node to visited nodes list. */
          visitedNodes.add(currentNode);

          /* Explore each edge of this node. */
          for (Edge currentEdge in currentNode.edges) {
              /* Edge returns to previously visited node, so ignore this edge. */
              if (visitedNodes.contains(currentEdge.destination)) {
                  continue;
              }
              /* Edge leads to a floor further from the destination than previously explored node,
                 so ignore this edge.
              */
              if (currentNode.floorNumber further from finalDestinationNode.floorNumber than
                previousFloorNumber) {
                  continue;
              }
              /* Edge does not meet accessibility requirements, so ignore this edge. */
              if (accessibility and not currentEdge.isAccessible) {
```

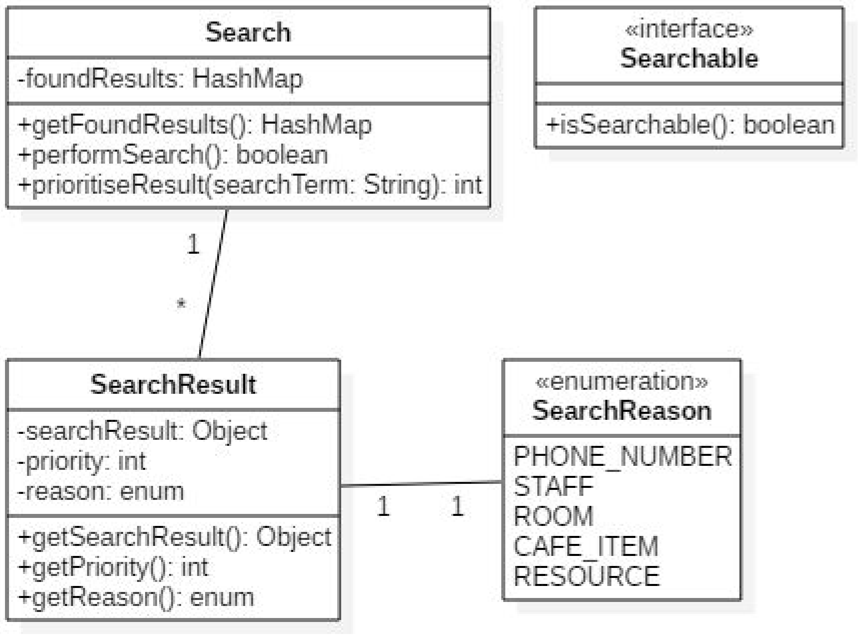
```
            continue;
        }
        /* Continuing route would be longer than current best route, so ignore this edge. */
        if (candidateWeight + currentEdge.weight >= bestRouteWeight) {
            continue;
        }
        /* Current shortest route found. */
        if (currentEdge.destination == finalDestinationNode) {
            candidateRoute.add(currentEdge);

            /* Set static “best” values to their current counterparts. */
            bestRoute = new List<Edge>(candidateRoute);
            bestRouteWeight = candidateWeight + currentEdge.weight;

            /* After best route has been updated, remove current edge from candidate route
              to move back through recursive call chain and explore along the next edge.
             */
            candidateRoute.remove(currentEdge);
            continue;
         }

         /* No backtracking possible, so add edge to candidate route. */
         candidateRoute.add(currentEdge);

         /* Explore along candidate route. */
         recursiveExplore(currentEdge.destination, finalDestinationNode, currentNode.floorNumber,
           visitedNodes, accessibility, candidateRoute, (candidateWeight + currentEdge.weight));

         /* Remove current edge from candidate route to move back through recursive call
            chain and explore along the next edge.
         */
         candidateRoute.remove(currentEdge);
     }

    /* After all edges of this node have been explored, remove current node from list
       of visited nodes.
    */
     visitedNodes.remove(currentNode);

    /* Move back up recursive call chain to explore edges of next node. */
    return;
End
    /* After all layers of recursion have returned, the “bestRoute” variable will contain the list
       of edges to be traversed from origin to destination for the shortest, accessibility
       requirement compliant, route and the “bestRouteWeight” variable will contain the total
       distance for the optimal route. “bestRoute” will be returned by the method calling
       “recursiveExplore” (from within the Navigator class).
    */
```

Navigation.Navigable

An interface which provides a method to return the nearest navigation node to this navigable object. All navigable objects should implement this interface. A navigable object is a room, or facility to which a user should be able to request navigation directions.

Search Package

A package which contains classes to enable the user to search for rooms, staff, facilities and other objects in the Urban Sciences Building (fulfills FA.04).



Search.Search

The class from which the search is run. Contains methods for searching object fields as well as prioritising each search result. Each returned result will be prioritised based on a percentage string match algorithm, which puts the results with a higher match percentage higher in the returned list.

Search.SearchResult

A class which represents a search result from a user search query (FA.04). A search result holds various user displayable metadata including but not limited to, its reason for appearing a result and a priority which is used sort itself against other search results. A search result is generated on device without the need for an internet connection.

Search.SearchReason

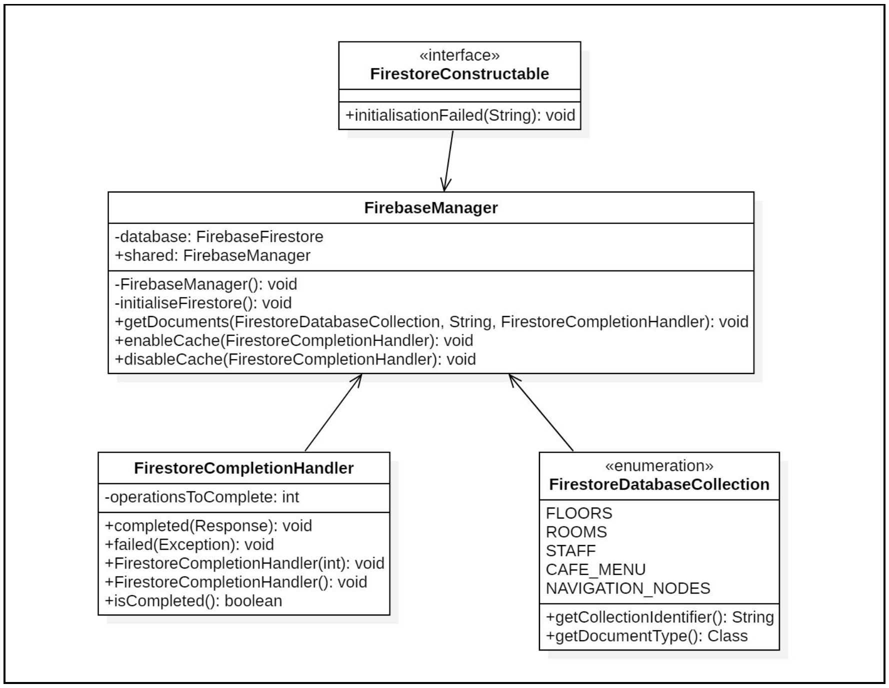
The enum type which is used by SearchResult to denote why that result has been returned. E.g. if Jason Steggles' phone number is searched for, Jason Steggles will be returned with the reason Phone Number. This helps the user to understand why certain results may be returned.

Search.Searchable

An interface which defines the object to be discoverable through the unified search.

Firebase Package

A package which contains classes to enable interaction between the application and Firebase


«interface»
FirestoreConstructable
+initialisationFailed(String): void
FirebaseManager
-database: FirebaseFirestore
+shared: FirebaseManager
-FirebaseManager(): void
-initialiseFirestore(): void
+getDocuments(FirestoreDatabaseCollection, String, FirestoreCompletionHandler): void
+enableCache(FirestoreCompletionHandler): void
+disableCache(FirestoreCompletionHandler): void
FirestoreCompletionHandler
-operationsToComplete: int
+completed(Response): void
+failed(Exception): void
+FirestoreCompletionHandler(int): void
+FirestoreCompletionHandler(): void
+isCompleted(): boolean
«enumeration»
FirestoreDatabaseCollection
FLOORS
ROOMS
STAFF
CAFE_MENU
NAVIGATION_NODES
+getCollectionIdentifier(): String
+getDocumentType(): Class

Firebase.FirebaseManager

A class which manages the interaction and sharing of data between the application and Firebase.

Firebase.FirestoreConstructable

An interface which declares an object as being to be constructed from a Firestore document dictionary. All updatable components of the application should implement this interface.

Firebase.FirestoreCompletionHandler

An abstract class which defines a completion handler for a Firestore operation. This handler manages reporting of whether the update of any particular class from database data was successful or not.

Firebase.FirestoreDatabaseCollection

An enum which defines the Firestore collection identifier and relative path for each data structure in the application's model. The members of this type are used to retrieve the relevant information from the database for each class being updated.

CoreApp Package

A package which contains utility classes which configure various helpers for the Android application.

CoreApp.AppDelegate

A class which acts as the application's delegate.

CoreApp.ColorUtility

A class which parses colors stored in a map by Firebase into colors usable by the Android operating system.

8.7 GUI, Human Interface Views

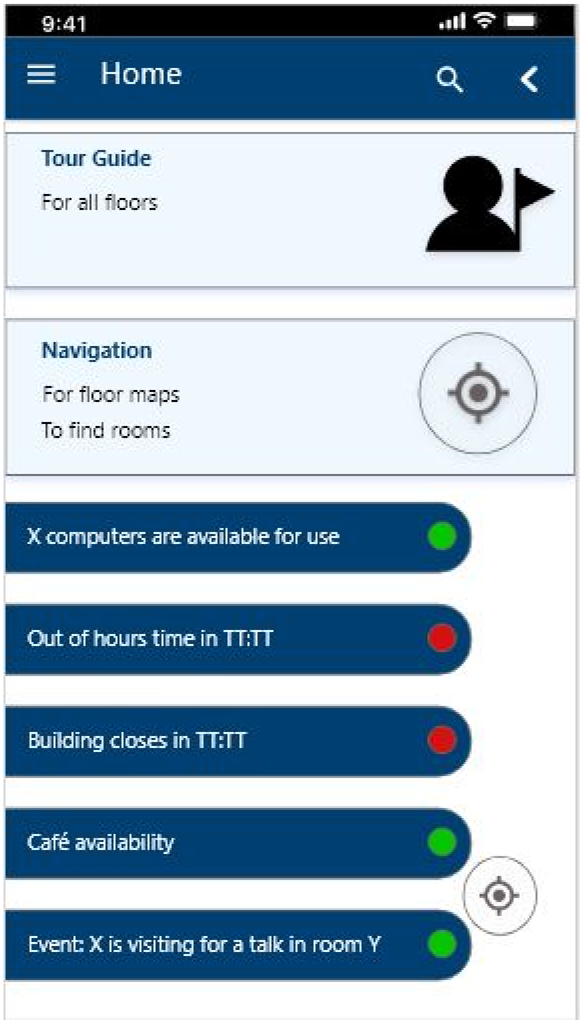
The key strengths of this Graphical User Interface are its simplicity and the fact that it is easily recognizable in relation to the current University app. It is designed to be similar to the existing application that most of the students at Newcastle University have used, by sharing features such as the colour scheme and positioning of certain buttons within the Newcastle University application. The design follows the HCI (human computer interface) guidelines for new users to grow accustomed to, so that they do not struggle with the navigation, or usage of the app (www.tutorialspoint.com. n.d.).

This means the design has to be consistent throughout so the user has an easy time navigating through the app. It follows any general app that a user may have used beforehand when it comes to usability. It allows the user to go back to the homepage easily from the hamburger menu. The application utilises a colour scheme which is based on the floor colours of the actual building so the user can recognise additional information easily. Overall, this results in a clean and user friendly GUI.

The application will have major functions such as the navigation button, the tour guide button, the floating action button (which is located in the bottom right corner), and the hamburger menu. It is important that the user is not overwhelmed with an excess of options and information so that there is a balance between a clean UI and simplified version of each floor plan and the navigation view, which will give clear cut directions to users so that they know where to go and what route to take. The tour guide will provide relevant images of rooms and will give users information about the building and location as a whole.

These decisions were influenced by input from the client, to improve ease of use. The colour scheme was chosen as it is in conjunction with Newcastle University theme. The shades of colours have also been chosen so that they do not strain the eyes as much as a similar app with a brighter colour scheme. We have colour coded the floor buttons, as per the client’s request, so that the floor the user is currently on is easily recognisable without looking for signs. We have made the direction instructions simple and similar to Google Maps so the user can use their previous experience with Google’s navigation products to navigate around the building. All of these choices were made to abide by client requests and to make the product more inclusive to users.

App Designs



Tour Guide

For all floors

Navigation

For floor maps

To find rooms

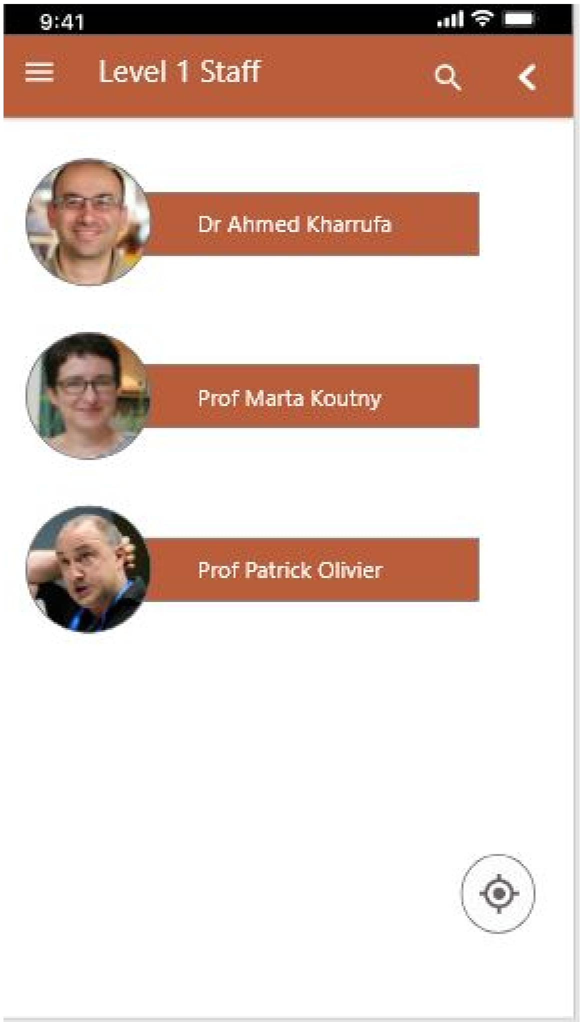
X computers are available for use ●

Out of hours time in TT:TT ●

Building closes in TT:TT ●

Café availability ●

Event: X is visiting for a talk in room Y ●



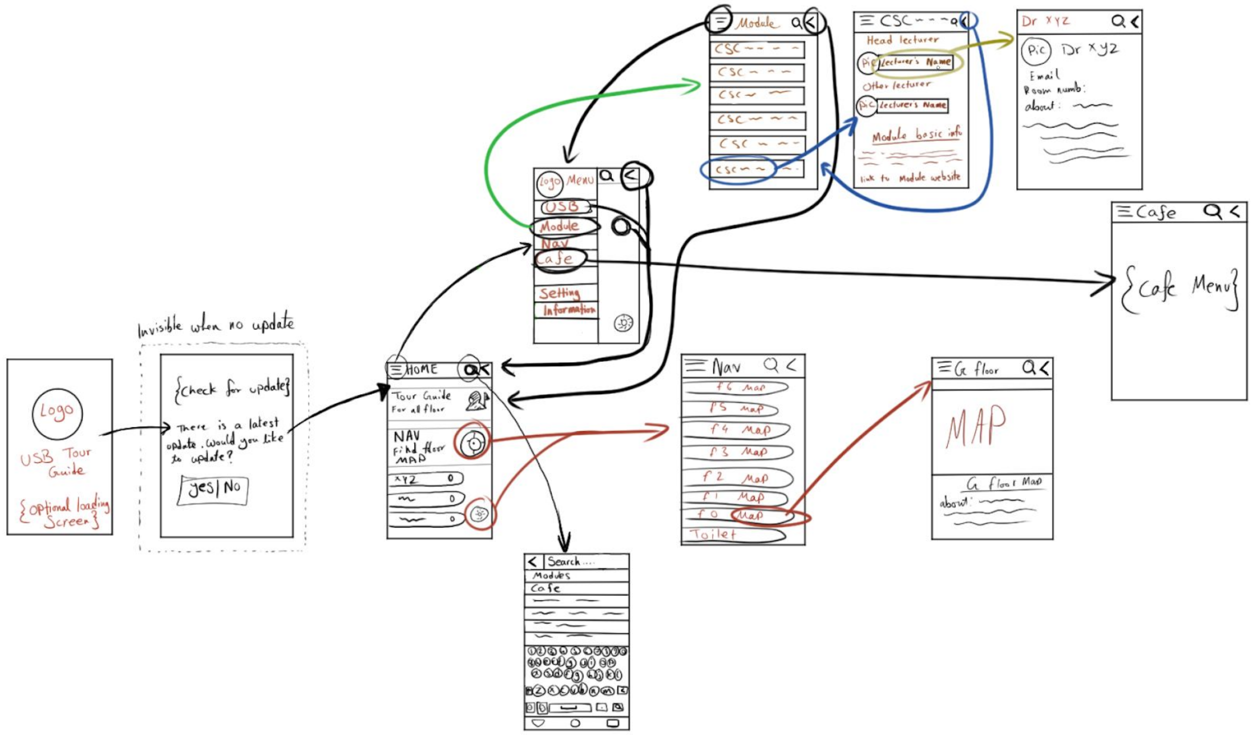
Dr Ahmed Kharrufa

Prof Marta Koutny

Prof Patrick Olivier

These are two concept designs that are subject to change for improvements but these shows the design being something that people are already used to. This includes the inclusion and placements of specific buttons. It is to the point on what it wants the user to see and it follows the standard blue colour of newcastle but as you enter different floors, the colours may change to represent the colour of the floors. These are subtle details that are are naturally integrated as the app is being used.

App Storyboard



The storyboard is included to show the general design of the app on how the user can move from page to page using the buttons given. This illustration, although not a finalized version of what exactly will be in the app with what names, gives a better idea on how the app should look and to make look similar to any normal apps so the users can naturally learn and understand how the app works. The buttons such as the back button and the search button are in places where a user can expect them to be from their previous experience using other apps. The information laid out per pages are not overwhelming and they are just enough for the user to be able to read and understand and not enough to make them confused. This also fits and follows the HCI guidelines for consistency and universal usability (www.tutorialspoint.com. n.d.).

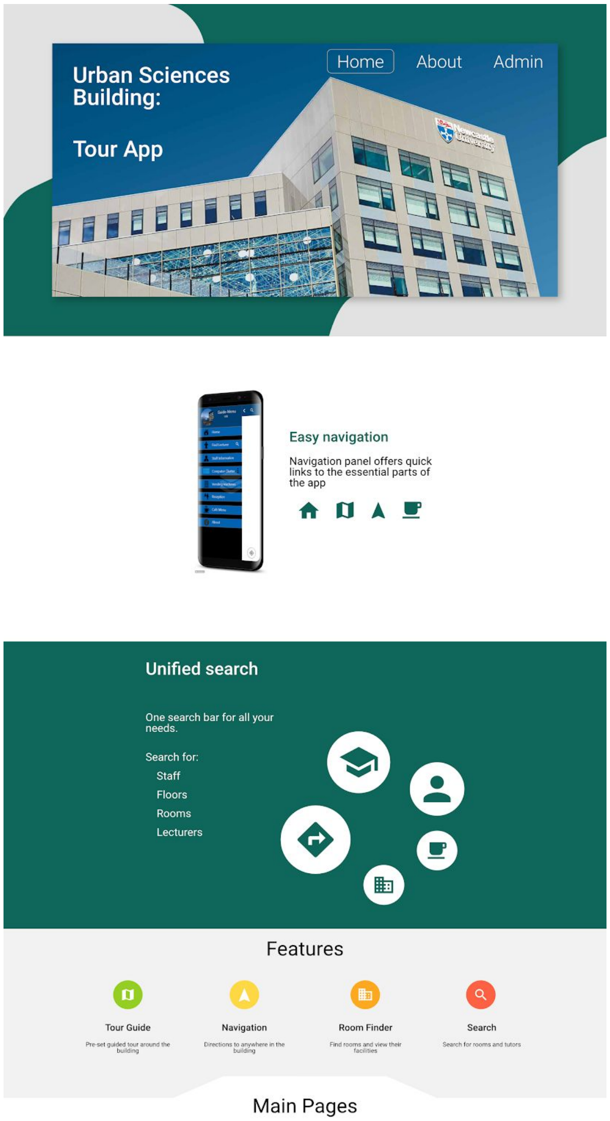
The app starts by displaying the logo and the name of the app which then immediately checks for update which will show up and ask for update if available, if not then it will lead straight to the home (or equivalent) page.

The home page will display tour guide and navigation buttons as a large portion so the user straight away knows what our focus is towards. It will also display some information such as open time for the building, cafe, etc with red and green light to indicate closed and open.

The navigation button leads to a page of buttons to lead to a map of certain floors. For each floors, the buttons will be the colours that represents the respected floor colour, e.g ground floor will have colour grey and floor 6 will have a dark green colour. Clicking on one of the floors will lead to a page of the floor map that is simplified for the user since the user do not need to know every little details of the floor and some about sections on what the floor consists of, for e.g for floor 2, it will include the fact that there is a computing reception.

There is also a button that leads to the cafe menu so the user knows what is available to eat. Search button brings up the keyboard below and relevant searches to what the user may want to look for above so as the user is typing to find something, the search page will keep updating itself to give the user more relevant searches that he or she may want.

Website Homepage Design



Urban Sciences Building:

Tour App

Easy navigation

Navigation panel offers quick links to the essential parts of the app

Unified search

One search bar for all your needs.

Search for:

Staff

Floors

Rooms

Lecturers

Features

Tour Guide

Pre-set guided tour around the building

Navigation

Directions to anywhere in the building

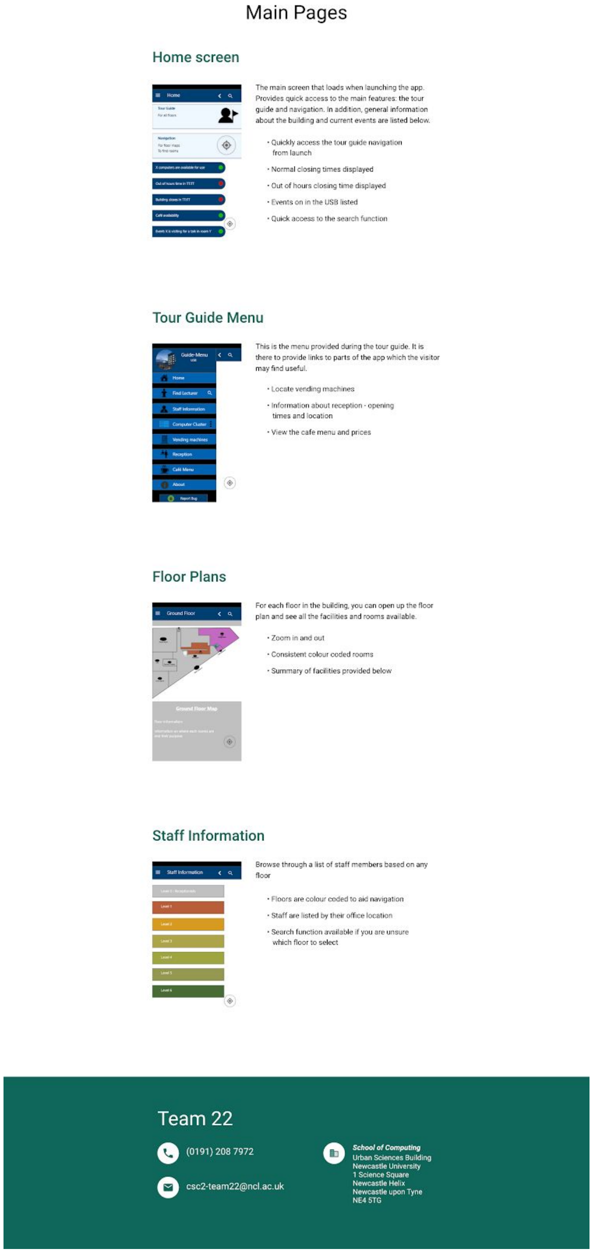
Room Finder

Find rooms and view their facilities

Search

Search for rooms and tutors

Main Pages



Main Pages

Home screen

The main screen that loads when launching the app. Provides quick access to the main features: the tour guide and navigation. In addition, general information about the building and current events are listed below.

- Quickly access the tour guide navigation from launch
- Normal closing times displayed
- Out of hours closing time displayed
- Events on in the USB listed
- Quick access to the search function

Tour Guide Menu

This is the menu provided during the tour guide. It is there to provide links to parts of the app which the visitor may find useful.

- Locate vending machines
- Information about reception - opening times and location
- View the cafe menu and prices

Floor Plans

For each floor in the building, you can open up the floor plan and see all the facilities and rooms available.

- Zoom in and out
- Consistent colour coded rooms
- Summary of facilities provided below

Staff Information

Browse through a list of staff members based on any floor

- Floors are colour coded to aid navigation
- Staff are listed by their office location
- Search function available if you are unsure which floor to select

Team 22

(0191) 208 7972

csc2-team22@ncl.ac.uk

School of Computing
Urban Sciences Building
Newcastle University
1 Science Square
Newcastle Helix
Newcastle upon Tyne
NE4 5TG

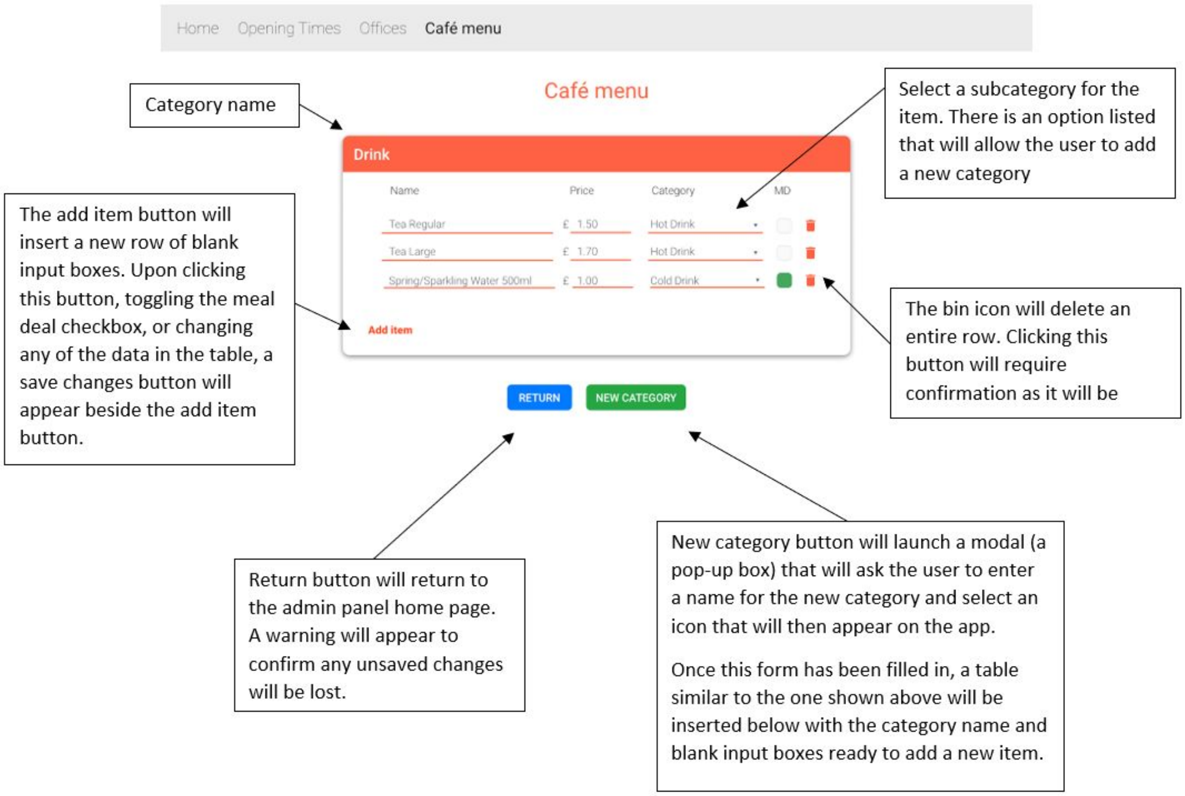
The pictures above show designs for the website's homepage. The left image consists mainly of icons with minimal supporting text to show the main features of this app at a glance. There is an image of the app's main screen running on a model phone to promote the app as a final product.

The image on the right is a continuation of the homepage with focus on the app's main pages. Each page consists of a screenshot of the page with a greater amount of text to explain the features on the given page.

Website Admin Design

Below is an annotated design for part of the admin site. The page shown here can be used to update the café menu. The design follows many android apps as the theme uses elements of Google's material design, their icon set and the Roboto font.

The other pages for the admin area (opening times and staff offices) will follow the same page layout for consistency.



Name	Price	Category	MD
Tea Regular	£ 1.50	Hot Drink	□
Tea Large	£ 1.70	Hot Drink	□
Spring/Sparkling Water 500ml	£ 1.00	Cold Drink	☑

Category name

Select a subcategory for the item. There is an option listed that will allow the user to add a new category

The add item button will insert a new row of blank input boxes. Upon clicking this button, toggling the meal deal checkbox, or changing any of the data in the table, a save changes button will appear beside the add item button.

The bin icon will delete an entire row. Clicking this button will require confirmation as it will be

Return button will return to the admin panel home page. A warning will appear to confirm any unsaved changes will be lost.

New category button will launch a modal (a pop-up box) that will ask the user to enter a name for the new category and select an icon that will then appear on the app.

Once this form has been filled in, a table similar to the one shown above will be inserted below with the category name and blank input boxes ready to add a new item.

References

Maps - Navigate & Explore. (2019). California, United States of America: Google.

Nielsen, J. (1994). *10 Heuristics for User Interface Design*. [online] Nielsen Norman Group. Available at: https://www.nngroup.com/articles/ten-usability-heuristics/ [Accessed 7 Mar. 2019].

Newcastle University. (2019). Newcastle upon Tyne, United Kingdom: Newcastle University.

Washington DC Tourist. (2018). California, United States of America: Stoked Studios.

www.tutorialspoint.com. (n.d.). Guidelines in HCI. [online] Available at: https://www.tutorialspoint.com/human_computer_interface/guidelines_in_hci.htm [Accessed 5 Mar. 2019].

Contribution Matrix

	Alex Beeching	Alexander MacLeod	Daniel Vincent	Shorif Akhonda	Chester Lloyd	Patrick Lindley	Moeez Shahid	Daniel Walker
1 Background & Analysis	C	R	R	R	R	R	R	R
2 Roles and Deliverables	M	M	C	R	R	R	R	C
3 Project Plan	R	C	R	R	R	R	C	R
4 Hardware and Software Platforms	M	R	R	R	M	R	C	R
6 Definition of Terms	R	R	M	R	C	M	R	R
7.1 Functional Requirements	R	R	C	R	R	R	R	R
7.2 Non-Functional Requirements	R	R	C	R	M	R	R	R
8 Other Considerations	R	C	R	R	R	M	R	R
9 Software Design	R	M	M	C	C	C	M	C
References	C	R	R	R	C	R	R	R