An integrated solution for planning campus paths from student's timetables

Yi Liu

Supervisor. Dr. Sandy Gould

BSc. Computer Science with an Industrial Year

School of Computer Science
University of Birmingham

GIT REPOSITORIES

University GitLab:
https://git-teaching.cs.bham.ac.uk/mod-ug-proj-2018/yxl605.git

GitHub (permanent):
https://github.com/YiLiuNat/yiliunat.github.io/tree/master/fyp

Project website (permanent):
https://yiliunat.github.io/fyp

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and appreciation to my supervisor, Dr. Sandy Gould, for his time and assistance to support this project, and by extending to me his interest and feedback without hesitation at all times.

ABSTRACT

This project was developed using user-centred design to provide an integrated solution for student timetables and campus path planning system. This solution connects with student timetables and plans a path automatically for students to reach their lecture buildings. The selected implementation method is an online mobile web-app, and it has been designed to behave like a native mobile app (compatible with both Android and iOS system). This solution applies to university visitors and students, allowing visitors to navigate university's landmarks/buildings, and enabling students to upload their timetables for an automatically updated map that helps them plan paths for their next lectures. The aim is to help students and visitors plan more efficient paths. This project included research on the following online-map interaction topics: (1) the ways to solve the problem of multiple building entrances in the navigation system; (2) a complete set of methods to combine student timetables and path planning; (3) the ways to improve student work efficiency as much as possible. Moreover, most of functions in this solution are based on JavaScript. Therefore, it includes research on the following front-end topics: (1) the ways to recognise a student's timetable without using server computations; (2) the ways to store a student's timetable with web local storage (once timetables are uploaded, results should be kept and loaded automatically every time users start the system).

Keywords: Front-end Development, Online Mapping System, JavaScript, Google Maps API, Human-Computer Interaction, User-centred Design

Table of Contents

Table of Tables

Table of Codes

Table of Figures

1 Introduction

Buildings with multiple entrances may be considered an obstacle for people using a navigation system, as the system may not take users to the closest entrance. Since the system will not tell users if a building has multiple entrances, it can be a hassle for those who have never been to the building or who are unfamiliar with its surroundings. This forces users to leave plenty of time to go to unfamiliar places.

For students of University of Birmingham (UoB), another possible problem is that students having difficulty locating the lecture room only using their timetable, especially for year 1 students. In addition to the unfamiliar environment, one of the reasons is that some of the building names given in student timetables are not shown correctly on the mobile map (Neither Google Maps nor Apple Maps).

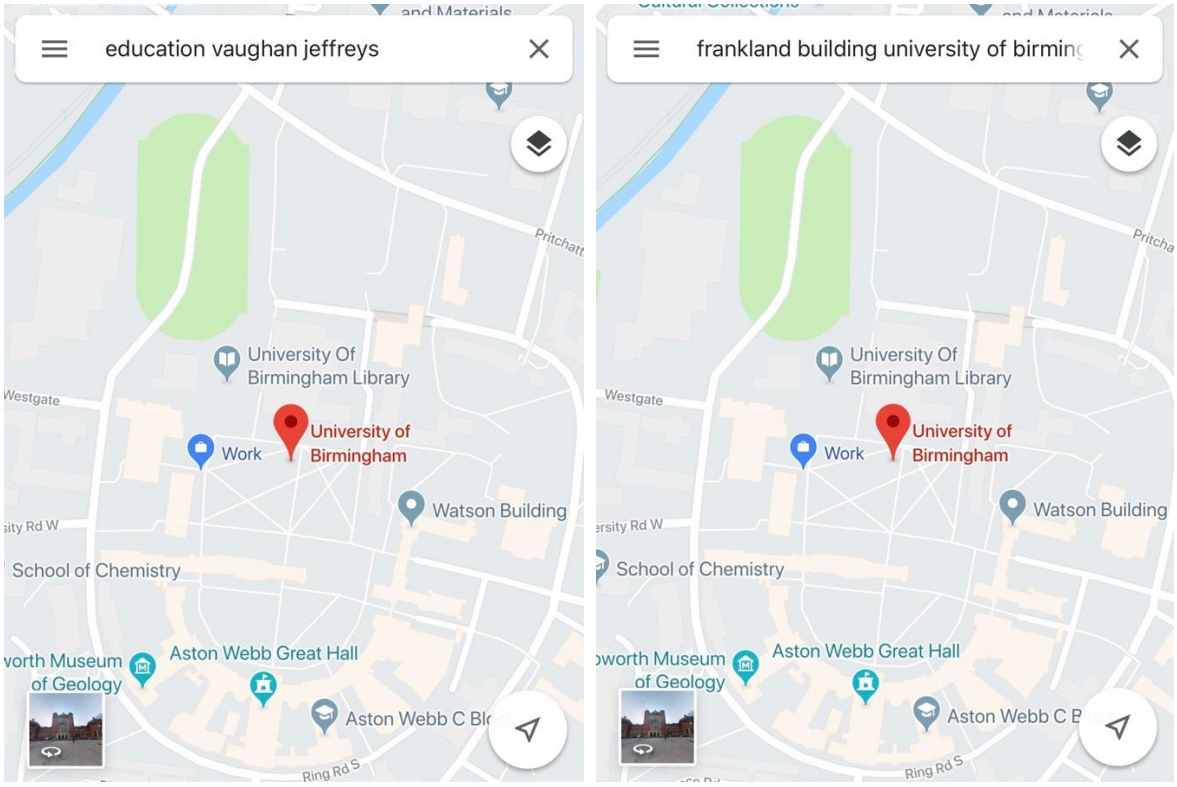


Figure 1. Wrong Search Results of University Buildings

Due to this issue, students typically use the university-offered Campus Maps, but the university-offered map is not functional when used for navigation. Most importantly, the existing timetable system (my.bham) is not mobile-friendly, students usually save the screenshot of their timetable, but the building name on a screenshot cannot be copied and pasted to the navigation app. Students continue to alternate between timetable and map repeatedly on their phone, which is a terrible experience.

In addition to above problems, since this project is targeted at mobile devices, the solutions of mobile web layout was also studied. The main problems and their solutions of this project were summarised in Table 1.

Problems	Descriptions	Proposed Solutions
Multiple entrances during navigation	When navigating a building with multiple entrances, the result would return a house rather than the entrance.	Use a Pop-up tab to ask users to select their preferred entrance and navigate precisely to that selected entrance.
Wrong search results of UoB buildings	When searching for UoB buildings in the navigation application, the result may be incorrect.	Record the correct coordinates of UoB buildings and replace the wrong results.
Interaction problem of campus navigation	Users have to switch between timetable and navigation applications to obtain results.	Combine the timetable and navigation applications, obtain the next/ongoing lecture as a prompt and plan paths automatically for students.
Mobile web-app layout	Traditional web layouts show different output at different resolutions, which is bad for mobile experience.	Use flexible layouts to display output equally on different screen sizes.

Table 1. Mobile Navigation Problems And Proposed Solutions

Using the problems in the table above I surveyed students from the University of Birmingham (UoB) School of Computer Science and used a user-centred design method to implement an integrated solution for solving the existing problems. This system is a mobile web-app that connects with student timetables, it automatically reminds student's ongoing lectures and plans a most efficient path to their next lecture building. Students will no longer need to constantly switch between their timetables and maps, using this new system to replace their timetable while accessing the system on a mobile browser as they walk around campus. To provide precise navigation, this system provides entrance pictures and allows users to choose which entrance to navigate towards. This solution is especially useful when students are unfamiliar with their lecture buildings at the beginning of each semester. Apart from that, it is also very friendly for year 1 students and visitors.

2 Literature Review

I mainly studied the solutions and reviewed literature for the following questions:

1. What research have been done on the navigation system for mobile devices? What are some excellent existent interaction designs?
2. What are good examples of existent campus navigations (e.g., 3D Navigation, AR Navigation or Traditional 2D Navigation)?
3. What are the pros and cons of existing map APIs (e.g., Google Maps, Apple Maps & Bing Maps, etc.)? Which map API is best for this project?

2.1 Mobile Navigation

Mobile navigation is a widely studied topic. This part of literature review focuses on excellent interactive projects of mobile navigation. Instead of only looking at campus navigation, extensive research was made into different fields of navigation software to provide a better approach for this project.

D. Suleiman created a mobile shopping mall navigator [1], an excellent example of indoor navigation. Such indoor systems usually employ Bluetooth [1], Wi-Fi or NFC [2] to approach the navigation function, requiring additional equipment installations in the scene. Therefore, the system itself will not be considered in this project, but in the interaction field, the mobile shopping mall navigator is designed to help customers unfamiliar with their surroundings to find the destinations in the most convenient way, which is very similar to this project. There are also useful interaction ideas in the indoor navigation field: (1) indoor navigation usually leads users directly to shop entrances, instead of just providing the location of a building like standard outdoor navigation; (2) In order to improve the efficiency, indoor navigation systems classifies commonly used places and facilities, creating categories for the most accessible places to help users to find what they want. For example, the indoor navigator of the Singapore Jurong Point shopping centre has designed a well-categorised places/facilities menu to help users navigate efficiently.

Figure 2. Indoor Navigator of Singapore Jurong Point Shopping Mall

Apart from traditional maps, many other interesting approaches of indoor navigation exist, for example, A. Möller et al. [3] researched Augmented Reality (AR) indoor navigation, it provides navigation by accessing the camera and adding an overlay (i.e. arrows and road signs) to the real world. Users usually have more interest on 3D navigation than 2D navigation according to the user study [4], and the main advantage of AR visualized navigation is user-friendly, because it does not need users to switch between virtual representation and real world [5]. The overlay information on the AR map system are helpful to improve the interaction experience [6].

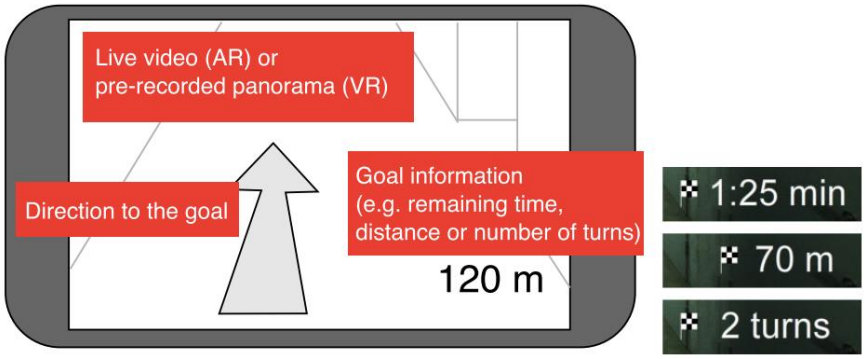


Figure 3. Overlay of AR Navigation [3]

However. there are two major issues with mobile AR navigation. The first issue is that system requires constant camera access and real-time graphics calculations will overload a phone's

battery dramatically [9]. The research by C. H. Lin et al. shown the results of their test Android phone with a Qualcomm Snapdragon 625 processor, which is not a very old processor, but showed a significant performance impact when running graphics calculations [8]. The second issue is that AR navigation requires the user to keep the camera pointed at the surrounding environment [3], which is uncomfortable and difficult to maintain.

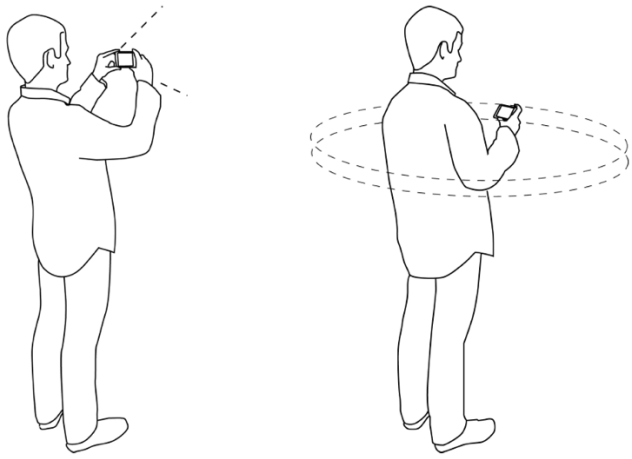


Figure 4. Posture of AR Navigation [3]

One of the major differences between shopping mall navigation and campus navigation is that campus navigation is a daily software. This project requires software that will be used frequently after combining with student's timetable and the defects of excessive battery consumption and short usage time must be avoided. Nevertheless, the interactive idea of a superimposed layer is valuable because it prevents the need to constantly switch between pages.

2.2 Campus Navigation

The existing interaction of the UoB campus (using a my.bham timetable and the mobile maps app) is a poor one. Students must alternate between their timetables and the map repeatedly on their phones. The following situation may occur when a long and complicated building name appears in front of a new student: the student tries to remember the complicated name shown in his timetable screenshot before switching to the maps app to type into the search bar, but cannot remember it clearly. Switching back to the screenshot, the student cannot immediately find where the lecture is located. Once found, there is alternating back to the maps app, but still no results. The student starts to think there is a typo on the screenshot, and tries opening a background apps page to preview the small preview window of the screenshot before switching back to the maps quickly to text it in, and so on. The experience is painful.

To solve this problem, I propose combining the student timetable with the navigation system. Such a system would plan routes automatically based on a student's lecture schedule.

According to my literature review, no one has tried to do this before. Therefore, in this part of the literature review, I shall focus on existing campus navigation projects to find useful samples and interaction designs.

AR navigation is also a popular topic in the campus navigation field [6], [8]–[10]. As the studies in the previous section, AR navigation currently has certain limitations and it is not the best solution for this project. However, the interaction logic in some AR navigation projects are valuable as the reference. One useful idea to improve efficiency is adding an overlay layer to the map [3], [6], reducing the number of times users switch between pages. A campus application implementation by H. Alqahtani [6] has combined the Google Maps and AR navigation, allowing users to select which map system to pop-up. This idea partially addresses the disadvantages of AR navigation, using an selected map system for different scenarios.

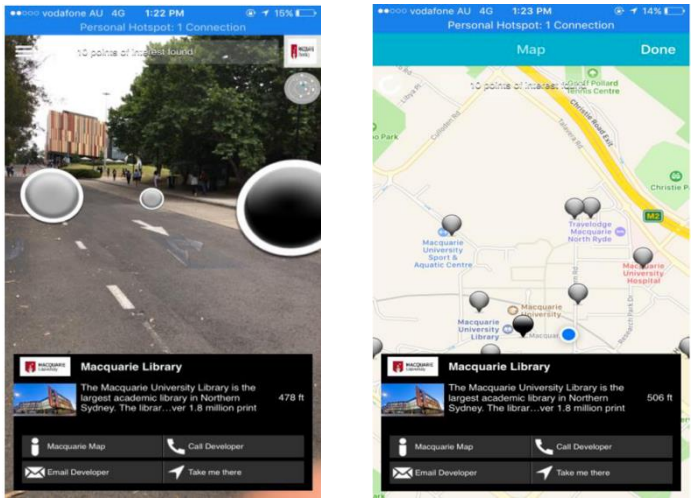


Figure 5. Overlay Page of Navigation Systems [6]

Apart from the AR navigation, A. S. Pagare et al. [11] completed a project on an event driven campus navigation system. This system is relatively similar to my project, but while my target is navigation based on student lectures, their target was navigation based on university events. This project proposed an enhanced solution to solve the problem of event navigation based on Google Maps. The proposed system included a registration system, a notification reminder for related campus events, and a Google navigator to lead users to the destination through Google Maps API. According to their studies, these sets of combinations could solve the problems effectively for both existing students and newcomers [11]. However, since the my.bham timetable has an inconvenient login function for mobile users, I would like to help users to skip the login steps, which will be explained later in the implementation of the system.

In addition to the app-based navigator, some projects are embedded in web pages (web-app). There are two main advantages for web-based applications on mobile platform. Firstly, users are not required to download the application, as they could visit the webpage and use the application directly. Secondly, web-based applications are compatible with both iOS and Android system, making them available to the wide range of users with a reasonable programming workload. M. Bopp et al. [12] completed a project about campus travelling to help users plan their trips, using a web-based design to solve the multiple platform problem. One useful design aspect from their project is the parking lot display function. Based on their research, the function was an effective way to summarise locations and provide suggestions for travellers. This function could be enhanced to display different facilities (e.g., study spaces, cafés and restaurants).

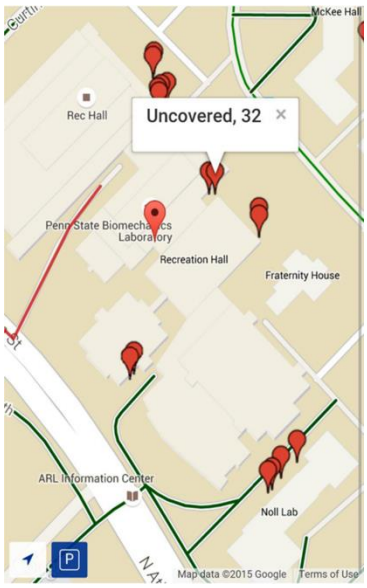


Figure 6. Parking Lots Display [12]

However, the disadvantage of mobile web-app is the layout problem. The general problem of the layout is that different resolutions (and the ratio between width and height) will cause different results of rendering, especially for various mobile screens. When we talk about mobile web adaption, a commonly question is, what do we really adapt? The answer refers to fitting webpages with multiple screen sizes to make them appear as the designer intended. In other words, when the same code runs on mobile phones with different resolutions, the space between elements, the size of elements, and the size of pictures will change, with the same proportion as the design draft. For better understanding, here is a sample page in two different resolutions, with a default fixed layout:

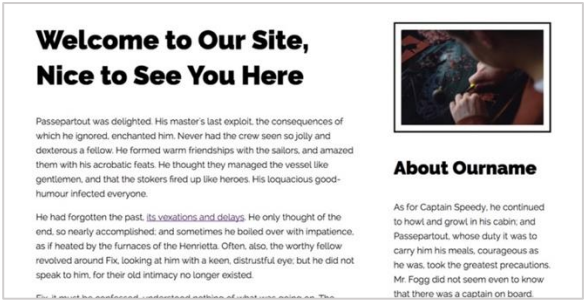


Figure 7. A webpage in 1920px width [36]

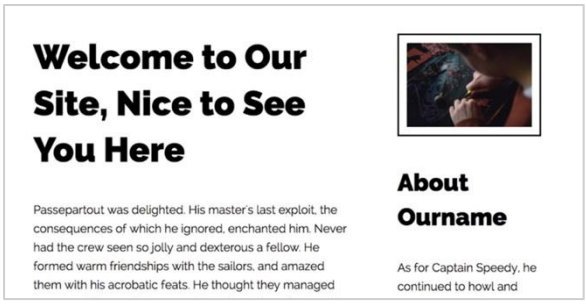


Figure 8. A webpage in 720px width [36]

The problem of the website in Figure 7 and 8 is that the proportion of font size and picture will be changed in different web browser resolutions. The key to solving this problem is to find a

unit of length so that the same proportion is scaled on different resolutions. According to Koch Peter-Paul's view-port idea [13], the most basic approach is using a rem layout as follows:

1. Set the viewport as the device width (there are other ways to implement this change, but I will focus on just this one for now).
2. Divide the viewport into 10rem (for easier calculations) and determine the pixel of 1rem in the current browser, giving it the font-size of the HTML tag.
3. Use 'rem' instead of 'px' when coding in CSS.

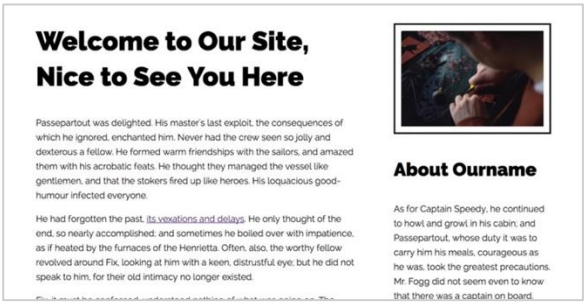


Figure 9. A webpage in 1920px width [36]

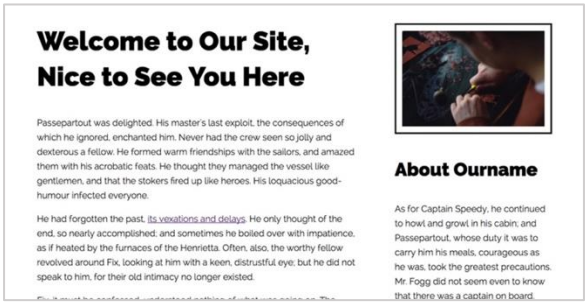


Figure 10. A webpage in 720px width [36]

By using this method, web pages will be scaled to the same proportion for different resolutions, which is exactly what this mobile-focused project needed.

Most of the campus navigation projects mentioned used open map Application Programming Interfaces (APIs), like Google Maps, as their navigation tool (e.g., projects in [6], [11], [12]), while shopping mall navigation and indoor navigation projects used their own drawn maps (e.g., projects in [1], [2]). Map drawing is hugely complex work aimed at areas where open map APIs are not drawn e.g., indoor areas. Open map APIs will be sufficient to solve this project's campus navigation problems once campus paths are drawn completely. Therefore, the most suitable direction for this project to progress is to find appropriate map APIs, and implement the corresponding functions using existing map APIs with an overlay layer.

2.3 Maps API Selection

There are many options available for map API selection, such as the popular Google Maps API [14], the recently released Apple Maps API (MapKit JS) [15], and the Microsoft Bing Maps API [16]. To determine the most appropriate map API for this project, several dimensions must be considered, such as the number of users, developers and supporting documents (as well as the accuracy of navigation on the UoB campus), all of which will be explained in this section.

There is a statistics about top 10 maps APIs in 2018 from RapidAPI [17], the best API was the Google Maps API; the second one was the Microsoft Bing Maps API. The Apple MapKit JS is newly released, therefore it was not included in the list. The number of API developers is

often important because it would have a better chance of finding answers to questions through the existing developer community if problems arise. However, there are significant differences in the number of discussions and developers between Google Maps API, Apple MapKit JS and Bing Maps API, as can be seen by the number of results on Stack Overflow. There are 2,450,000 results for the Google Maps API, 125,000 results for Bing Maps API and 16,500 results for Apple MapKit JS on Stack Overflow (obtained in March 2019). These results indirectly indicate that Google Maps API has the most developers and discussions.

The Google Maps API has a well-built developer platform and appreciable documentation. It has been updated with two major releases and is now in its third, named Google Maps V3. It has a very frequent update rate and supports more and more functions [18]. Not only the maps API, but the Google Maps application itself is also the best maps & navigation mobile application both in App Store and Google Play Market in 2018 [19], it has a very large user base and more rapid renewal in response to road changes [18].

Apple MapKit JS [15] is a new released API, but path planning returns results identical to the Apple Maps App. It has a very large number of users and excellent navigation results in the public field navigation on iOS platform. It was the third popular maps & navigation application in the U.S. 2018 [20]. Apple MapKit JS provides free 250,000 maps requests and 25,000 service requests per day [15], which is more than Google Maps' free limitation of 100,000 per month [21].

Since this project is aimed at campus navigation, the map with most accurate path planning function in the UoB campus should be chosen. As none of the three previously mentioned maps correctly included the names of all UoB buildings, the coordinates of campus buildings will be manually recorded into my programme. An important quality metric of map API is the accuracy of path planning between two coordinates. The accuracy of campus map path planning depends on the completeness of recorded paths. In general, the more complete the paths are (as provided through more frequent updating), the better the accuracy of the path planning. However, in terms of the completeness of UoB campus paths, Google Maps and Bing Maps have similar completeness, but the path planning results of Google Maps are significantly better than Bing Maps.

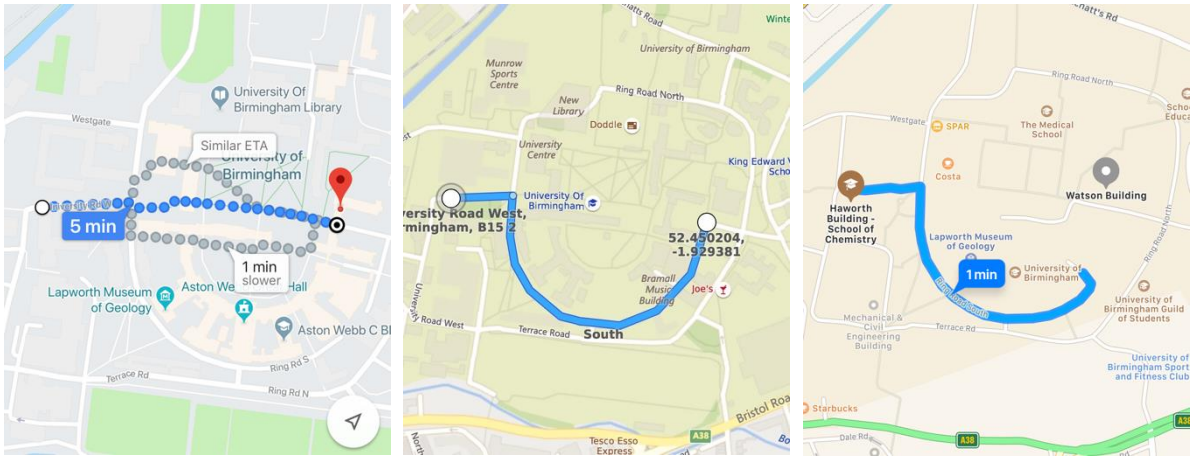


Figure 11. Path Planning from Haworth to Watson

(Google maps, Bing maps and Apple maps)

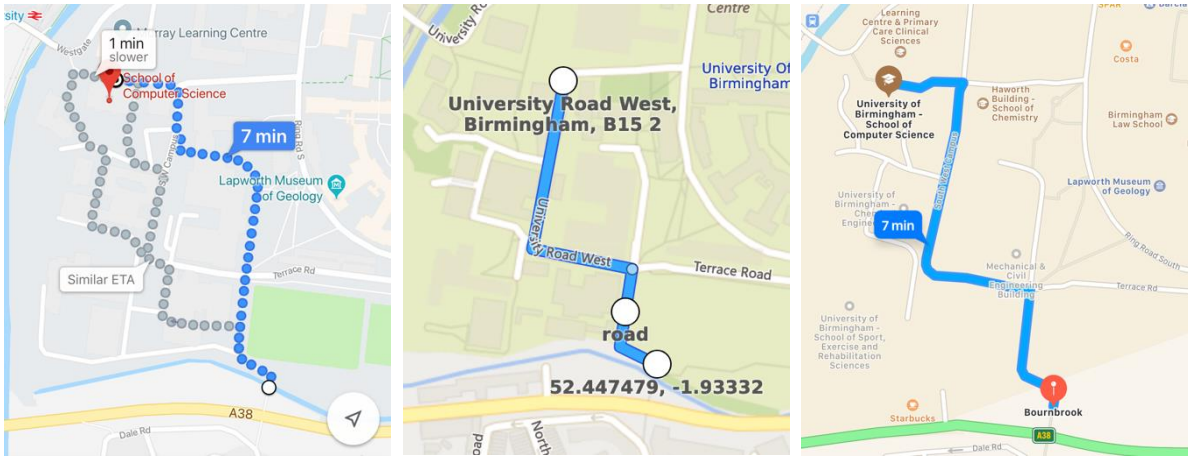


Figure 12. Path Planning from Computer Science to South Gate

(Google Maps, Bing Maps and Apple Maps)

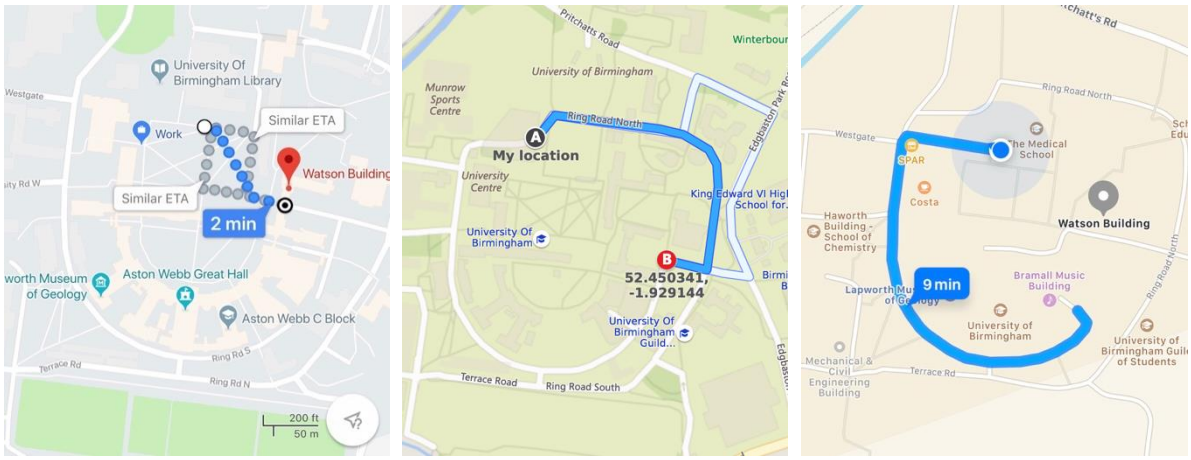


Figure 13. Path Planning from Library to Watson

(Google Maps, Bing Maps and Apple Maps)

Several path planning tests were made to obtain results from each map API, tracing the designated path from the Haworth building to the Watson building, Computer Science to the South gate and from the Library to the Watson building, respectively. In most cases, Google maps gave one or more similar solutions, but Bing maps and Apple maps could only give one solution. Importantly, Google maps was the only map that could plan the routes inside the university centre, e.g. in the first and third test. According to these tests, the accuracy of Google Maps was the best, allowing passage through the centre of the campus, while the results of Apple Maps and Bing Maps were relatively poor, providing paths that could only pass through the outer roads of the campus.

Apart from the poor performance on UoB campus path planning, Apple has also stated the requirement that application should not send more than one geocoding request per user per minute [22], which is not enough to refresh user's path timely. Microsoft Bing Maps has the same path planning issue as the Apple Maps, therefore it is not suitable for this project. With the consideration of the accuracy in the UoB campus and the larger developer community, the correct selection of maps API for this project should be Google Maps API.

2.4 Decisions

I will summarise the several decisions made regarding determining the technology chosen for the project here, and provide the reasons for those choices.

2.4.1 Type of Application

Decision	Technologies	Reasons
✓	Web-app	1. It does not require users to download an application. 2. It supports multiple operating systems (Android, iOS and Windows). 3. There are multiple map APIs to support a web-app.
✗	Android / iOS app	1. It requires users to download applications from App Store or Google Play Market, which is not convenient enough for efficient software. 2. The workload of this project is only permits the developmental support of a single-platform system.

Table 2. Decisions of Application Type

2.4.2 Type of Navigation

Decision	Technologies	Reasons
✓	2D navigation	1. The operations like most of navigation applications and is convenient for users to get started quickly. 2. It has low power consumption and high efficiency for mobile

		operation, which is sufficient for a daily used campus navigation app. 3. It has multiple map APIs support web-embed 2D navigation function.
✗	3D navigation	1. It has high power consumption, and not all mobile phones perform well using it.
✗	AR navigation	1. It requires constantly access to a phone's camera and real-time graphics calculation would overload the battery. 2. It requires the user to keep the camera pointed at the surrounding environment, causing discomfort and difficulty navigating.

Table 3. Decisions of Navigation Type

2.4.3 Type of Interaction

Decision	Technologies	Reasons
✓	Overlay layers	1. It reduces the times that users switch between pages. 2. It increases the efficiency and convenience after being combined with the map interface. 3. Switching operations, like my.bham timetable should be avoided.
✗	Multiple pages	Contrary to the above

Table 4. Decisions of Interaction Type

2.4.4 Maps API

Decision	Technologies	Reasons
✓	Google Maps API	1. Excellent accuracy of the UoB campus path planning. 2. It has a well-built developer platform and appreciable documentation. 3. It has a large developer base and community.
✗	Apple Maps API	1. Poor accuracy of the UoB campus path planning. 2. New released API may have uncertainty. 3. Lack of developer base.
✗	Microsoft Bing Maps API	1. Poor accuracy of the UoB campus path planning. 2. Lack of developer base.

Table 5. Decisions of Maps API

2.4.5 Layout Scheme

Selection	Technologies	Reasons
✓	Rem layout	1. Every element of web page will be scaled to the same size for different resolutions.
✗	Fixed layout	1. Different resolutions (and the different ratio between width and height) will cause the different results of rendering, especially bad for various mobile screens. 2. May cause problems for low-resolution devices.

Table 6. Decisions of Layout Scheme

3 Product Specification

In this project, I divided the product specification into three parts. I will firstly explain the tasks that should be carried out for users in this part, then explain the functional requirements for designed solutions in the next part and finally explain the non-functional requirements in the last part. See section 4 (Product Description) for function analysis and implementation details.

3.1 Tasks for Users

For visitors, the following requests and tasks should be fully accepted by the system.

1. Use the system without timetable uploading.
2. Be able to select the building and receive the corrected path planning results.
3. Move a significant distance and receive the updated paths.
4. Click any facility buttons to obtain the suggested locations.
5. Obtain a larger interface of the map.

For students, the following requests and tasks should be fully accepted by the system.

1. Upload the my.bham timetable to receive automatically updated lecture notifications and path plans.
2. Receive the notifications and path plans automatically upon application start up (timetable loaded automatically every time).
3. Open the sidebar to check their enhanced timetable.
4. Be able to select another building to receive a new path plan.
5. Move a significant distance and receive the updated paths.
6. Click any facility buttons to obtain the suggested locations.

3.2 Functional Requirements

1. Coordinates data

   1.1 The system should have a dataset of the university buildings' coordinates (latitudes and

longitudes), so that all possible destinations can be navigated correctly.

1.2 The system should have a series coordinates for campus parking lots, markets, cafés and restaurants.

2. Path planning

2.1 The system should have the path planning function to plan the path between the user's geolocation and a requested target location.

2.2 The planned path should be update regularly as the user moves.

2.3 The system should provide a drop-down menu to allow the user to select a campus building, showing a planned path from the user's geolocation to the selected building.

3. Visitor mode

3.1 The system should have a visitor mode, allowing campus visitors to use a basic navigation function without timetable uploading.

3.2 The system should allow visitors to select a destination (buildings & landmarks) and navigate the destination for them.

3.3 The system should hide all the unnecessary layouts (timetables, lecture reminder, etc.) and provide a larger navigation area for visitors.

4. Timetable storing

4.1 The system should allow students to upload their timetable and save the results.

4.2 The system should load the timetables automatically upon start up.

4.3 The system should allow students to delete and change saved timetable.

5. Timetable reading

5.1 The system should be able to read student timetables and parse out all the lectures students have.

5.2 The system should be able to understand the relationship between the time and date of different lectures in the context of timetable.

6. Lecture reminder

6.1 The system should provide a full list of lectures with dates and times based on student's timetable, which will help students check past, ongoing and future lectures.

6.2 The system should display the building and time for a student's next lecture based on the stored timetable data. The path to the next lecture should be also displayed automatically upon application start up.

7. Others

7.1 The system should have the function of displaying campus parking lots, markets, cafés and restaurants.

3.3 Non-functional Requirements

1. Robustness

1.1 The system should handle errors at all times instead of throwing them.

1.2 The system should be able to parse different type of student's timetables.

1.3 The system should alert the user and return to the visitor mode if user upload a wrong timetable or non-timetable files.

2. Compatibility

2.1 The system should compatible with both Android and iOS system.

2.2 The system should compatible with all modern browsers.

2.3 The system should display the content correctly in different sizes and resolutions of screen.

3. Usability

3.1 Users should not have any difficulties to use the system.

3.2 The designed solution should be able to use without guidance or only little guidance.

3.3 All designed content should be proven the usability.

4 Product Description

4.1 Rem Layout

This system is focused on mobile devices, making it particularly important to support mobile phones with different resolutions. In terms of layout, the disadvantage of a traditional fixed layout is that differing resolutions (with differing ratios between width and height) will cause results to vary when rendering pages. This kind of problem should be avoided in this project, as it may cause conflict between the map and content frames. For example, here are some Instagram webpages with different resolutions and the conflicts between elements:

Figure 14. 240px Width of Instagram Webpage

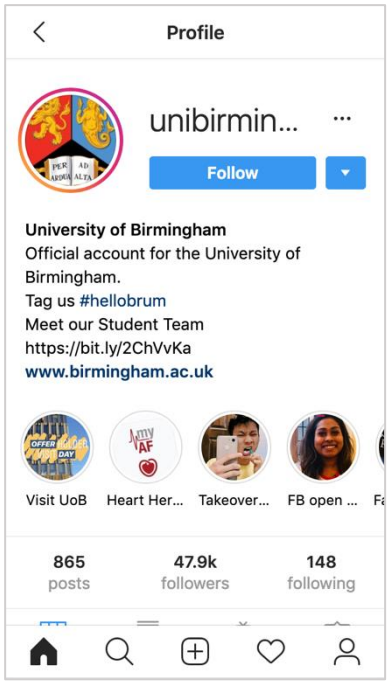


Figure 15. 320px Width of Instagram Webpage

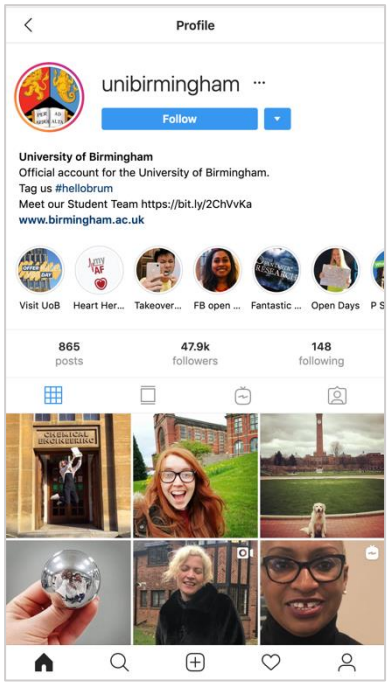


Figure 16. 480px Width of Instagram Webpage

There are two major problems here:

1. The first screen size is smaller, so the profile photo and buttons should also be smaller.
2. The first and second screens cannot show the full username, and the text layout takes up more space on the screen, preventing photos from being displayed.

The key to these problems is to determine a unit of length, so that the element sizes-values scale across different screen sizes. In other words, when the same code runs on mobile phones with different resolutions, the space between elements, the size of elements and the size of pictures should change while keeping the same proportion. According to the literature review [13], a rem layout is an appropriate way to solve this problem, because it is a unit that makes all elements scale in the same proportion on different screen sizes. The rem unit is calculated relative to the font size of the <HTML> tag. For example, if the font size of the <HTML> tag is set at 50 px and the width of tag is set at 1.2 rem, the actual width of tag should be 50 px*1.2 rem = 60 px. A necessary point to achieve this function is that the font size must be changed according to the size of the screen. When the font size changes, the actual value of the tag, “1.2 rem”, also changes. This dynamic font size function can be implemented using JavaScript, and the following pictures are system screenshots after applying a rem layout.

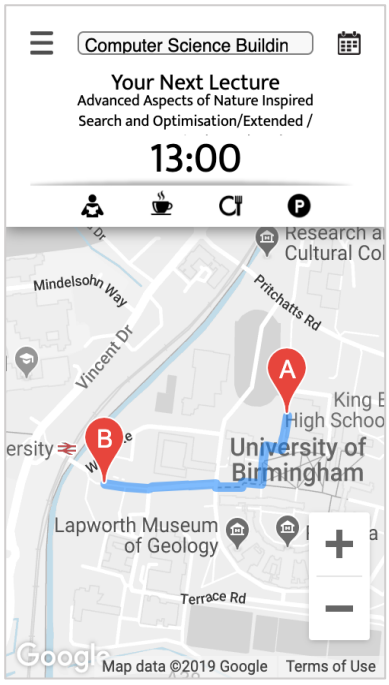


Figure 17. 320px-width with rem-layout

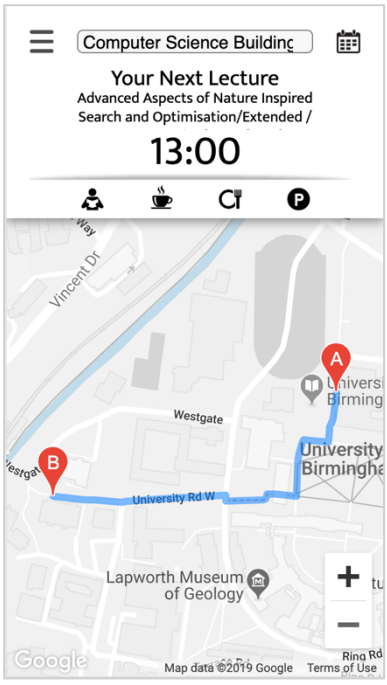


Figure 18. 240px-width with rem-layout

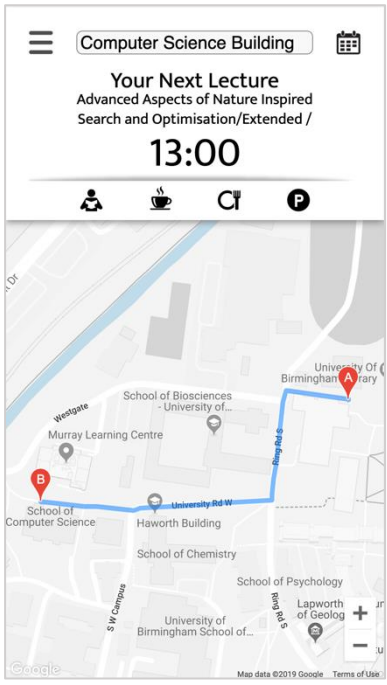


Figure 19. 480px-width with rem-layout

The proportion of content and map frames have been maintained, and the elements inside the content frame are also consistent. However, the zoom buttons inside the map frame have not applied the rem layout, since Google Maps API frame uses px as its size unit rather than rem.

The aim of this layout scheme is to resolve the frame conflict problem, making the size of zoom buttons irrelevant.

4.2 Incorrect Building Location Fixing

To ensure the building search results in the navigation app were correct, I created an array list to store the corrected results of the UoB buildings. The method of modifying search results in Google Maps involves providing the correct results in the form of latitude and longitude coordinates, and then using the path planning API to obtain the corrected path. The latitudes and longitudes can be collected manually using geolocation applications. (e.g., Latlng Finder [23]) The array list of coordinates should be look like this:

```
var latlngData = {
    "AstonWebbGreatHallG1": "52.449093,-1.930821",
    "AstonWebbGreatHallG2": "52.448617,-1.930905",
    "StaffHouseG1": "52.450400,-1.932921",
    "StaffHouseG2": "52.450541,-1.931957",
    "LibraryG1": "52.4513352,-1.9312653",
                    ⋮
}
```

Code 1. Sample Coordinates List

The coordinate data has two different formats in Google Maps API: “LatLng” and “LatLngLiteral”. Coordinates information in those two formats needs to be switched between different APIs. For example, Directions API uses both “LatLng” and “LatLngLiteral” formats but Markers API uses “LatLngLiteral” only [24]. Therefore, converting the data format based on different APIs is necessary. I used the following function to achieve:

```
new google.maps.LatLng(lat,lng)
```

Code 2. Function of LatLng Data Converting

It should be mentioned that Google Maps once drew a part of a UoB campus path incorrectly, and that the path was one of the main paths from the university station to university centre. As a result, feedback was sent to Google three times and each time, support responded to the feedback quickly. This exchange shows that Google Maps focuses on user experiences and feedback.

4.3 Visitor Mode

The system will not require users to upload their timetable in visitor mode, because visitors do not have them. However, visitors are still potential users of this system, requiring the corrected navigation function for buildings and landmarks. The interface of visitor mode will only include a drop-down menu, facility buttons and the map frame. The map will show a

corresponding path when a user selects a destination, and will update the results while the user is walking around campus.

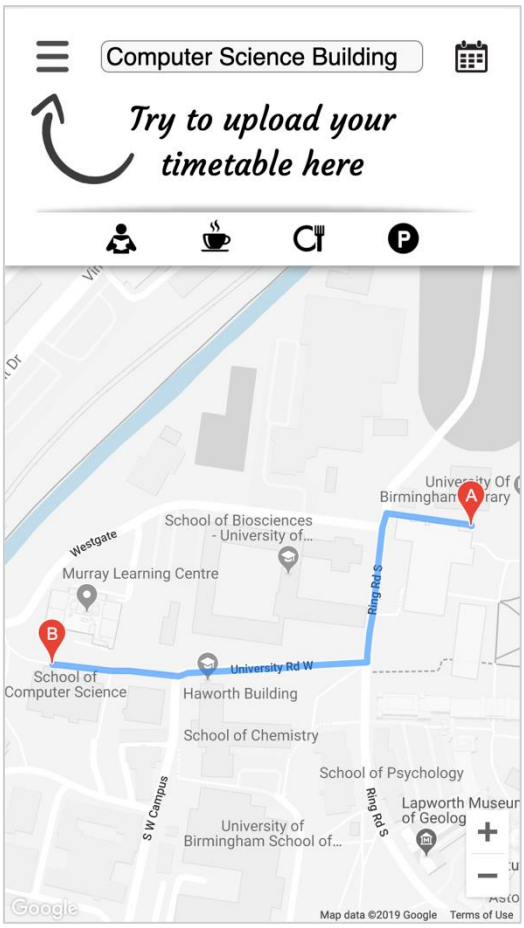


Figure 21. Visitor Mode

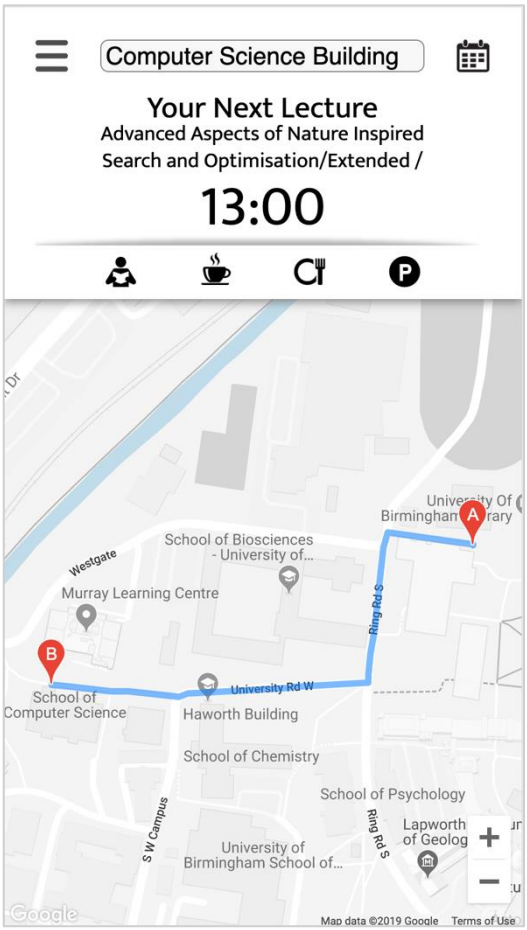


Figure 20. Student Mode

The system will change the user interface after recognizing an uploaded timetable, changing from the content of visitor mode to a new interface that includes a timetable display, a display of the next lecture, and a planned path to the user's next lecture.

4.4 Multiple Entrances Option

If the chosen building has multiple entrances, the system will pop up a tab and ask users to select their preferred entrance. The selected option will extract the corresponding location coordinates from coordinates list, and push them to the direction service to calculate the shortest path from user's geolocation to their selected entrance.

4.5 Maps Rendering

Since the Google Maps platform has an excellent direction API with accurate path planning function in UoB campus, but it does not provide a navigation function, the solution is to use the direction API to simulate a navigation function for the user. I used the direction API provided by Google Maps to implement a function that calculates the path between the user's geolocation and the target location. This function refreshes every 15 seconds, maintaining a freshly rendered map as users walk around the campus. However, every time the map is refreshed, the display range of the map will rescale to the full range of the path. This can affect the user experience if the map was zoomed or dragged on the screen. To address this problem, I added two functions: Firstly, a function to obtain the current centre position of the map and preserve the current viewport if that position changes, e.g. if the user has dragged the map, the display range will not be changed by refreshing. Secondly, as the viewport must change when a user changes the destination, another function will determine whether the system moved the map or the user dragged the map.

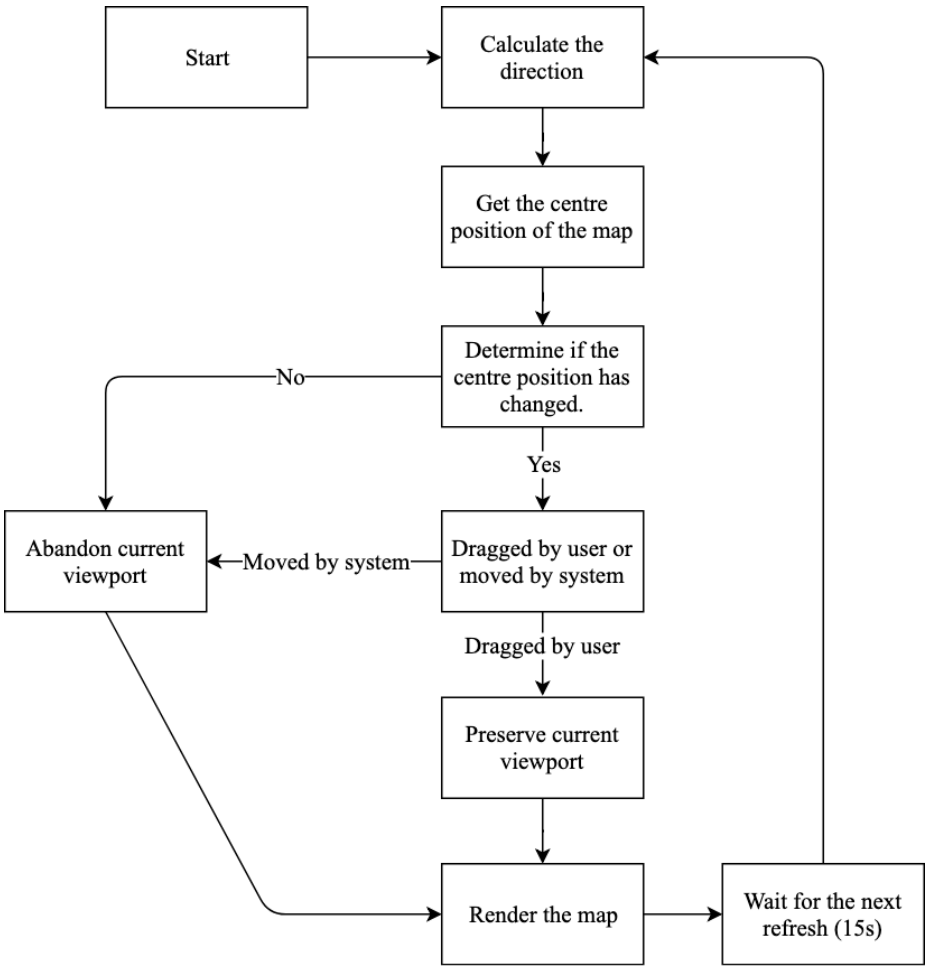


Figure 22. Flowchart of Maps Rendering

4.6 Methods of Timetable Access

The university usually does not provide the access to the student-timetable database, therefore, I will have to request that students upload their timetables to this system. In the early generations of the system, I manually wrote timetable data into a JavaScript list and used another function to read the timetable list. As a result, for users to use this software, they would also have to manually write the timetable into the JavaScript list, which would affect the user experience negatively. To solve the problem, I considered the following schemes to access student timetables: (1) Image recognition, where a screenshot of a timetable would be saved and the user asked to upload the screenshot for character recognition; (2) PDF recognition, where the timetable page would be printed and saved as a PDF file, and the user asked to upload the PDF for character recognition; and (3) HTML recognition, where the HTML of the timetable webpage would be saved, and the user asked to upload the HTML file for character recognition.

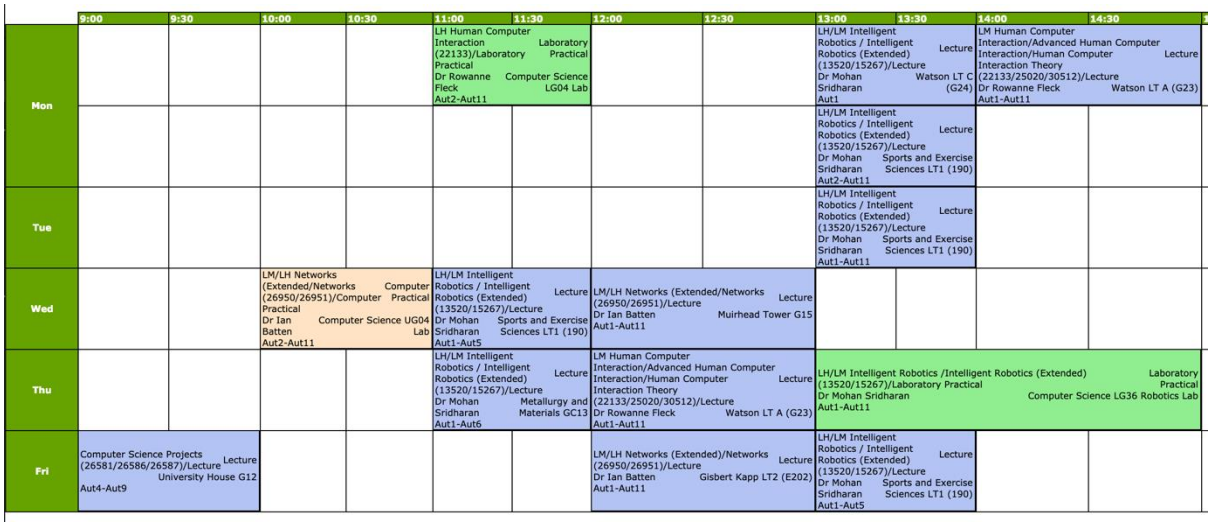


Figure 23. Timetable Sample From My.bham

The advantage of image recognition is the user operation convenience, especially for mobile phones, which easily allow screenshot and uploading operations. However, popular OCR engines (e.g. Ocrad.js [25] and Tesseract.js[26]) normally recognise image-text as plain-text [27], which means all the formatting information would be lost. For example, using image recognition to recognise a timetable (Figure 23) would cause the recognised content to be a contiguous set of text with indistinguishable dates and times. Therefore, image recognition is not the best solution for timetable reading.

The PDF recognition was originally considered as a backup solution, as HTML recognition was not successful initially. PDF recognition is not the easiest option, but it is achievable. There are JavaScript libraries [28] for PDF recognition suitable for this function. The result after PDF recognition will still be plain text [29], but every empty slot in a timetable contains a blank

space, making it possible to parse out the lectures and recognise corresponding times and dates by reading the number of blank spaces.

Ultimately, HTML recognition was selected as the recognition method for this project. The my.bham webpage uses relatively old technology, making capturing timetable data difficult. During the early stages of the study, I found that the timetable data was not kept inside the main HTML file; only by choosing "webpage (complete)" when downloading the timetable webpage allowed the downloading of an HTML file containing usable timetable code. The my.bham timetable system most likely uses JavaScript to retrieve timetable data, instead of loading the HTML file directly with timetable content. Given such a situation, obtaining the timetable data would be extremely difficult. The link for retrieving data is this, but the university does not allow users to log in that way for security reasons. If "all files" are selected when downloading a timetable, the main HTML files remain almost full of JavaScript code. However, after disconnecting from the Internet, I found the downloaded webpage could still be opened, indicating that the timetable data had been downloaded. After examining each one of downloaded files, I found the timetable data stored in a file named "timehout.html".

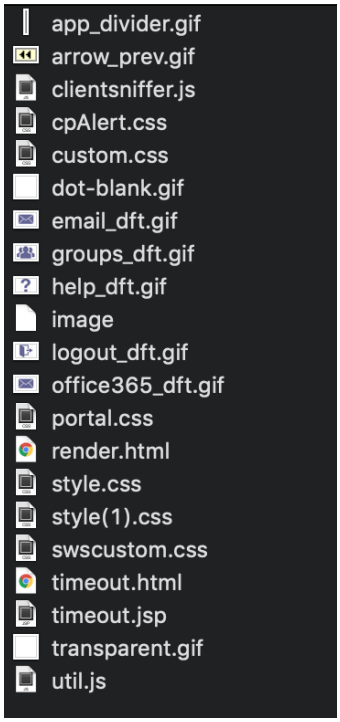


Figure 24. Downloaded Timetable Files

Since the information format of the data can be obtained from the HTML file, which is more robust than PDF or image recognition, the method of HTML recognition was selected. To employ this method, users must download their timetable webpage and upload the applicable timeout.html file. This would be more complex than image recognition, would also not be a frequent operation. Once uploaded, users will not need to upload their timetable information again unless their timetables change.

4.7 Local Storage

When users upload their timetable file to the system, the file is stored locally instead of being uploaded directly to the host server. The system will then read the stored data every time a user opens the system. Local storage [30] is provided by every HTML-5 supported browser with the traditional web cookie usually providing 4 KB to store data, but local storage providing at least 5 MB of space. As a typical timetable file is larger than 25 KB, local storage is the only viable option for this project, since local storage provides larger and more secure storage than a cookie. This technology supports the idea of "permanent storing of timetable"; once a

timetable is stored, users will not need to upload it again unless their timetable changes. Local storage would save the timetable file as string type, allowing the system to read the string of timetable data and parse out every lecture with its corresponding time and location.

4.8 Timetable Reading

Once a timetable file is uploaded, the system starts to read its contents. To make the system understand the relationship between the time and dates of different lectures in the context of a timetable, the system requires multiple calculations to identify the corresponding time of each lecture. Attempting to open the timetable HTML file in a code editor will reveal groups of code blocks, each representing a lecture. Each code block contains a lecture's corresponding name, lecturer and building name, but not a corresponding time, as shown below:

```
1 <td style=";" class="Laboratory_Practical" colspan="2" rowspan="1">
2 <table class="object-cell-args" cellspacing="0" border="0" width="100%">
3   <colgroup><col class="object-cell-0-0">
4   <col class="object-cell-0-2">
5 </colgroup><tbody><tr>
6   <td align="left">LH Human Computer Interaction (22133)/Laboratory
    Practical</td>
7   <td align="right">Laboratory Practical</td>
8 </tr>
9 </tbody></table>
10 <table class="object-cell-args" cellspacing="0" border="0" width="100%">
11   <colgroup><col class="object-cell-1-0">
12   <col class="object-cell-1-2">
13 </colgroup><tbody><tr>
14   <td align="left">Dr Rowanne Fleck</td>
15   <td align="right">Computer Science LG04 Lab</td>
16 </tr>
17 </tbody></table>
18 <table class="object-cell-args" cellspacing="0" border="0" width="100%">
19   <colgroup><col class="object-cell-2-0">
20   <col class="object-cell-2-2">
21 </colgroup><tbody><tr>
22   <td align="left">Aut2-Aut11</td>
23   <td align="right"></td>
24 </tr>
25 </tbody></table></td>
```

Code 3. Code Block of Lecture in My.bham Timetable Page.

As shown in Code 3, the lecture name is shown on line 6, the lecturer on line 14 and the building name on line 15, but there is no corresponding time. The my.bham system renders timetables row-by-row, and the first row of the timetable is the time slot, containing a total of 20 time-slots from 09:00 to 18:30.

```
1 <tbody><tr>
2     <td></td>
```

```
 3     <td class="col-label-one" colspan="1">9:00</td>
 4     <td class="col-label-one" colspan="1">9:30</td>
 5     <td class="col-label-one" colspan="1">10:00</td>
 6     <td class="col-label-one" colspan="1">10:30</td>
                              ⋮
19     <td class="col-label-one" colspan="1">17:00</td>
20     <td class="col-label-one" colspan="1">17:30</td>
21     <td class="col-label-one" colspan="1">18:00</td>
22     <td class="col-label-one" colspan="1">18:30</td>
23 </tr>
```

Code 4. Time Slots of My.bham Timetable Page.

The system then renders the second row (all the lectures on Monday), the third row (all the lectures on Tuesday), and so on. Therefore, the corresponding time of each lecture is not directly available for reading, but still needs to be calculated by traversing the whole timetable.

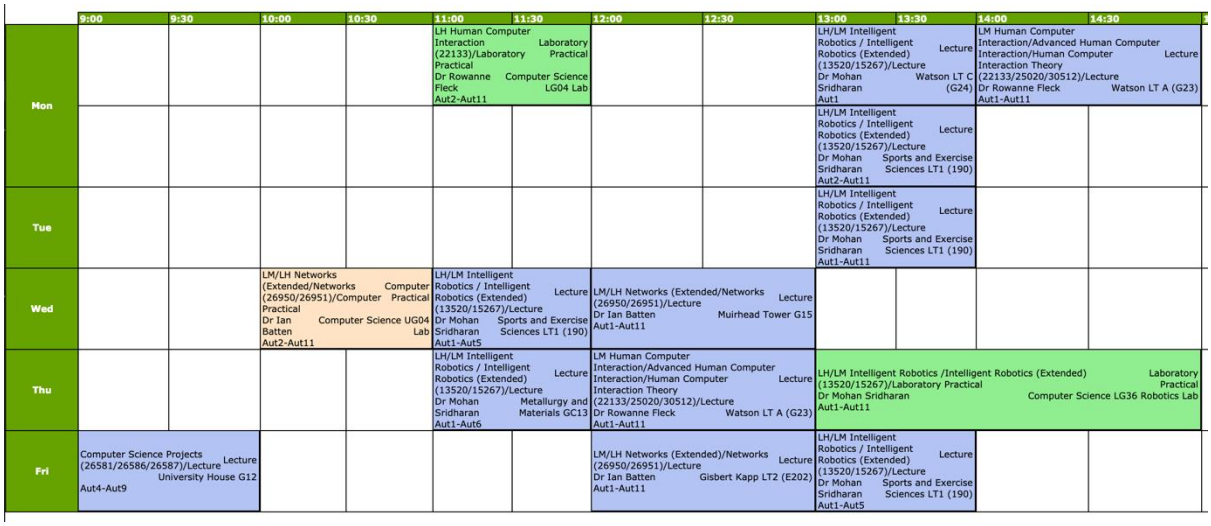


Figure 25. Rows of My.bham Timetable

As shown in Code 4, each time slot has a “colspan = 1” attribute. The length of each colspan represents 30 minutes on a timetable, giving a one-hour lecture (line 1 in Code 3) a “colspan = 2”, which represents 60 minutes. Therefore, 20 “colspans” equals 20*30/60, or 10 hours per day, which is the exact duration of the total time slots from 9:00 to 19:00 (slot 18:30 represents 18:30–19:00). However, if there is no lecture in a specific time slot, there will be no colspan attribute in the code. Instead, there will be a blank space placeholder (“ ” in HTML) in the table representing 30 minutes (one colspan), which is the content of following code:

```
1     <td style=";" class="cell-border"> </td>
2     <td style=";" class="cell-border"> </td>
3     <td style=";" class="cell-border"> </td>
```

Code 5. Placeholders of "Null" Lecture in My.bham Timetable.

The start time of each lecture can be calculated using the sum of all colspans before that lecture. The name and location of the lecture can be parsed out from the corresponding code block in the timetable. The logic of this function should resemble the following flowchart:

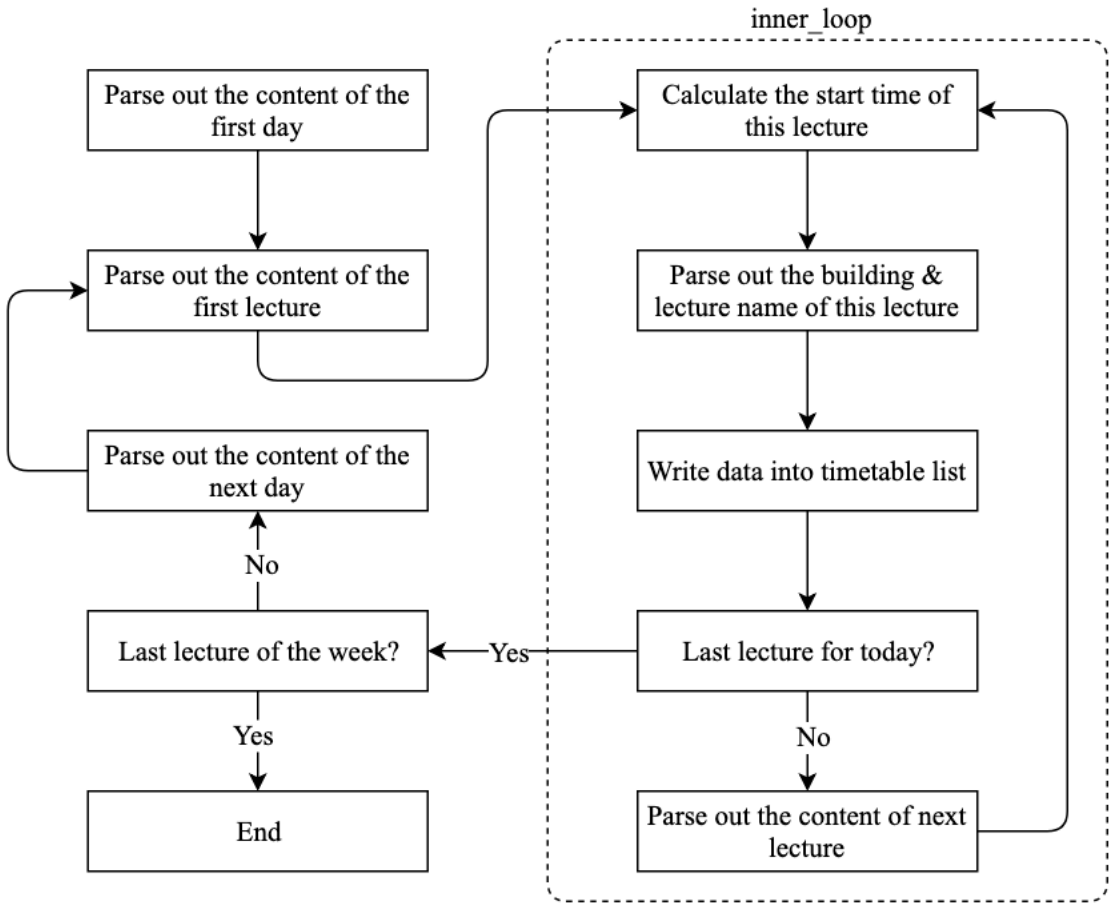


Figure 26. Flowchart of Timetable Reading

All the HTML codes of timetable were stored as the string type in the local storage, therefore I used the split function in JavaScript to parse out the information that I need. However, there are two data have to be converted before the timetable data reading: firstly, convert the integers 1 – 5 to the string from Mon - Fri, and then convert the number of "colspans" to the corresponding time, like the following:

```
function num2day(number){
    var strDay = "";
    switch(number){
        case (1): strDay = "Mon"; break;
        case (2): strDay = "Tue"; break;
        case (3): strDay = "Wed"; break;
        case (4): strDay = "Thu"; break;
        case (5): strDay = "Fri"; break;
    }
    return strDay;
}
```

Code 6. Convert String-day to Number-day.

```
1       function slot2time(slotNum){
2           var curTime = 0000;
3           switch(slotNum){
4                case (1) : curTime = 900;  break;
5                case (2) : curTime = 930;  break;
6                case (3) : curTime = 1000; break;
7                case (4) : curTime = 1030; break;
                                  ⋮
20               case (17): curTime = 1700; break;
21               case (18): curTime = 1730; break;
22               case (19): curTime = 1800; break;
23               case (20): curTime = 1830; break;
24          }
25          return curTime;
26      }
```

Code 7. Convert "Colspan" Number to Corresponding Time.

After those two conversions, the data can be extracted by using the number of time and day. I used a double for-loop to implement this function. The inner loop is used to traverse every lecture of the day, and the outer loop is used traverse every day of the week, like the following pseudo-code:

```
1  for (i = 1; i < 6; i++)
2      thisDay = Parse out the string between num2day(i) and num2day(i+1)
3      timeSlot = 0 , colspan = 0
4
5      for (j = 1; thisDay != null; j++)
6          thisLecture = Parse out the string between lecture j and lecture j+1
7          beforeLecture = Find out how many " " before the lecture.
8          timeSlot = timeSlot + beforeLecture + colspan
9          //Calculate the start time of this lecture.
10         lectureTime = slot2time(timeSlot)
11         lectureName = Parse out the lecture name from thisLecture.
12         lectureBuilding = Parse out the building name from thisLecture.
13
14         Write the lecture time, name and building into the timetable list.
15
16         colspan = Find out the colspan of this lecture.
17     end
18 end
```

Code 8. Simple Form of Timetable Reading Pseudo-code.

It should be noticed that data writing procedure (line 14 in Code 8) should be inserted into the corresponding position of i (day) and j (lecture) in the timetable list. If there is a “null” lecture (empty space in timetable), then just skip it. According to this method, all the lectures in the timetable could be completely inserted into an array list.

4.9 Timetable Display

This function is designed from suggestions of user evaluation. One of designed tasks was requiring test users to check the correctness of their recognised timetable, and some users complained about the difficulty of checking. Therefore, I designed this function to help.

After the timetable reading, this function will change the layout of the side bar and display all lectures in a categorized format from Monday to Friday. This feature is designed to help users check the correctness of timetable reading, and improve the user, allowing users to replace their my.bham timetable by using this combined system directly.

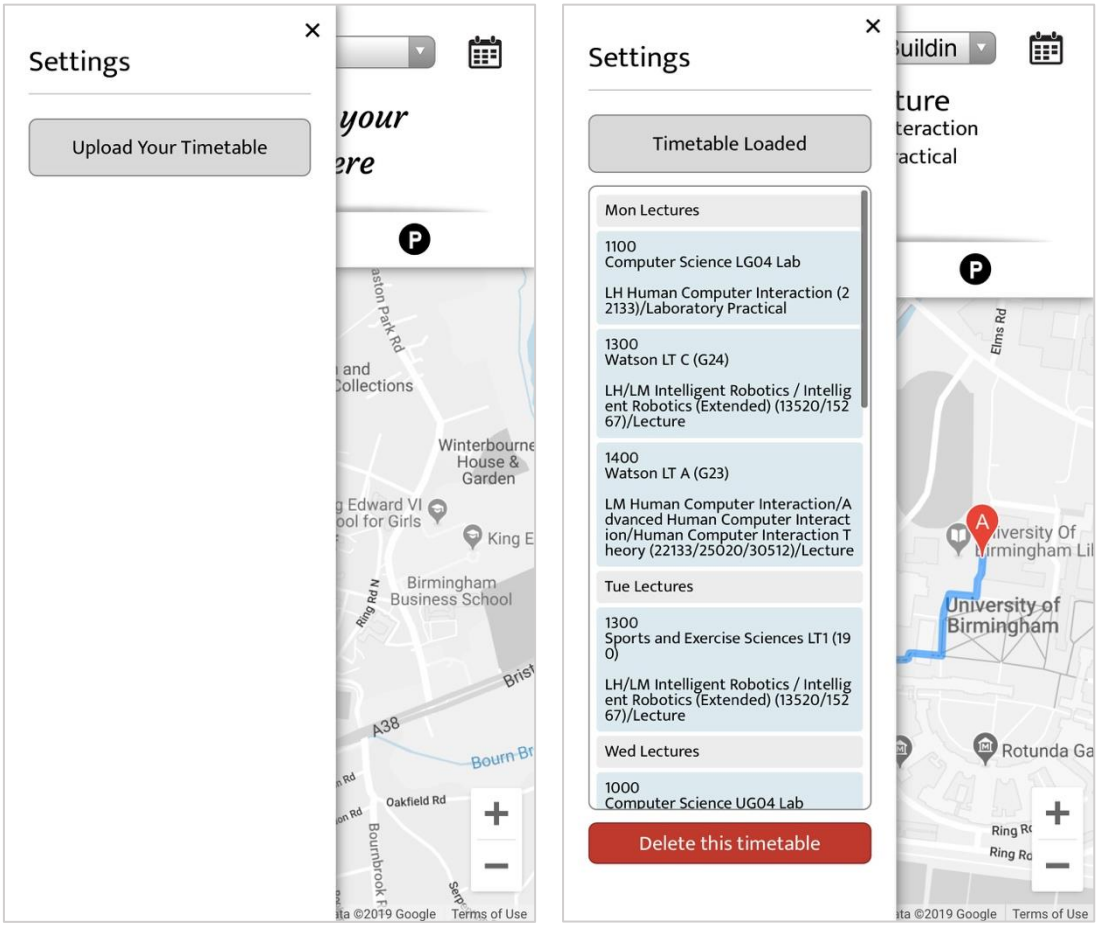


Figure 27. Timetable Display

This function is implemented by inserting the corresponding lecture block (blue blocks) when traversing every lecture of the day, and inserting the corresponding day block (grey blocks) when traversing every day of the week. It should be noted that this system is mainly designed for mobile browsers. The moving operation of a mobile screen is dragging rather than using a mouse wheel. To support the draggable function smoothly on mobile devices, I used the iScroll.js library to support a draggable object in a web-embedded frame, for more information please see [31].

4.10 Lecture Reminder

This function shows users when their next lecture is and automatically plans the shortest path for them by reading the lectures in the timetable list. The system needs to compare the current time with the lecture time to determine which lecture to prompt. However, the following points need to be highlighted:

1. After a certain lecture has past, the information of the next lecture should be pushed automatically, instead of requiring the user to refresh the page manually.
2. If all lectures for a day are finished, the reminder should display the next day's first lecture.
3. At some timetables may have a full day without any lectures, the system should prompt the next available lecture.
4. If accessed on a weekend, the system should display the first lecture on the following week.
5. To prevent an endless loop, an exit and throw "empty timetable" alert after traversed the timetable twice.

The logic should be like the following flowchart:

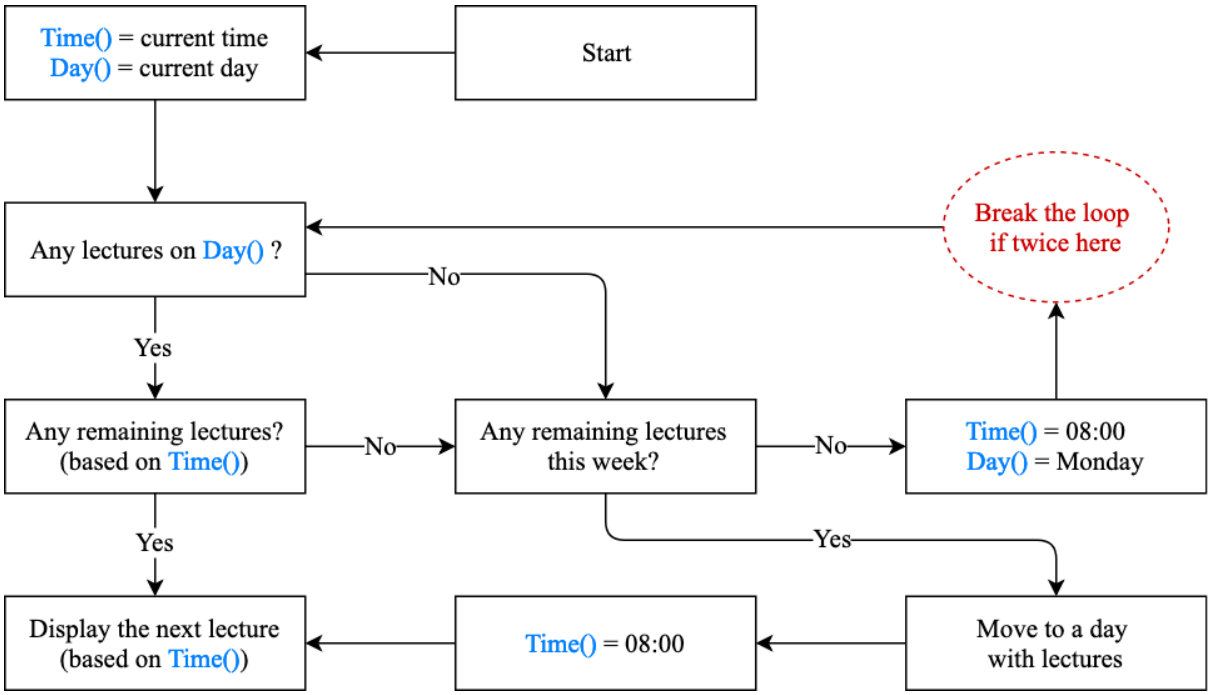


Figure 28. Flowchart of Lecture Reminder

To implement this function, the Day() and Time() function should be defined first. The first element in JavaScript list is 0, therefore the Day() function should return the value that minus one.

```
function day(){
    var date = new Date()
      , week = date.getDay()
      , _week = week-1;
```

```
5     return _week;
6 }
```

Code 9. Definition of Day()

JavaScript has separated the function of getMinutes() and getHours(), therefore the Time() function should return a combination of those two results. The result of getMinutes() would only contain a single number if the result is smaller than 10, therefore an if-condition is necessary to be held.

```
1 function time(){
2     var date = new Date()
3       , minutes = date.getMinutes();
4     if(minutes < 10){minutes = '0'+minutes;} //e.g. 20:03 is 3 rather than 03
5     var combinedTime = parseInt(date.getHours()+''+minutes);
6     return combinedTime;
7 }
```

Code 10. Definition of Time()

4.11 User Interface

I will explain the user interface by using a set of annotated wireframes in this part. All of figures was captured from a real-time working mobile phone, you can visit this link to get the same experience.

4.11.1 Visitor mode

This page is for users without a timetable, e.g. visitors. Its main functions contain a drop-down menu of destinations (buildings and landmarks) and several facility buttons. Users can obtain path planning by selecting a destination.

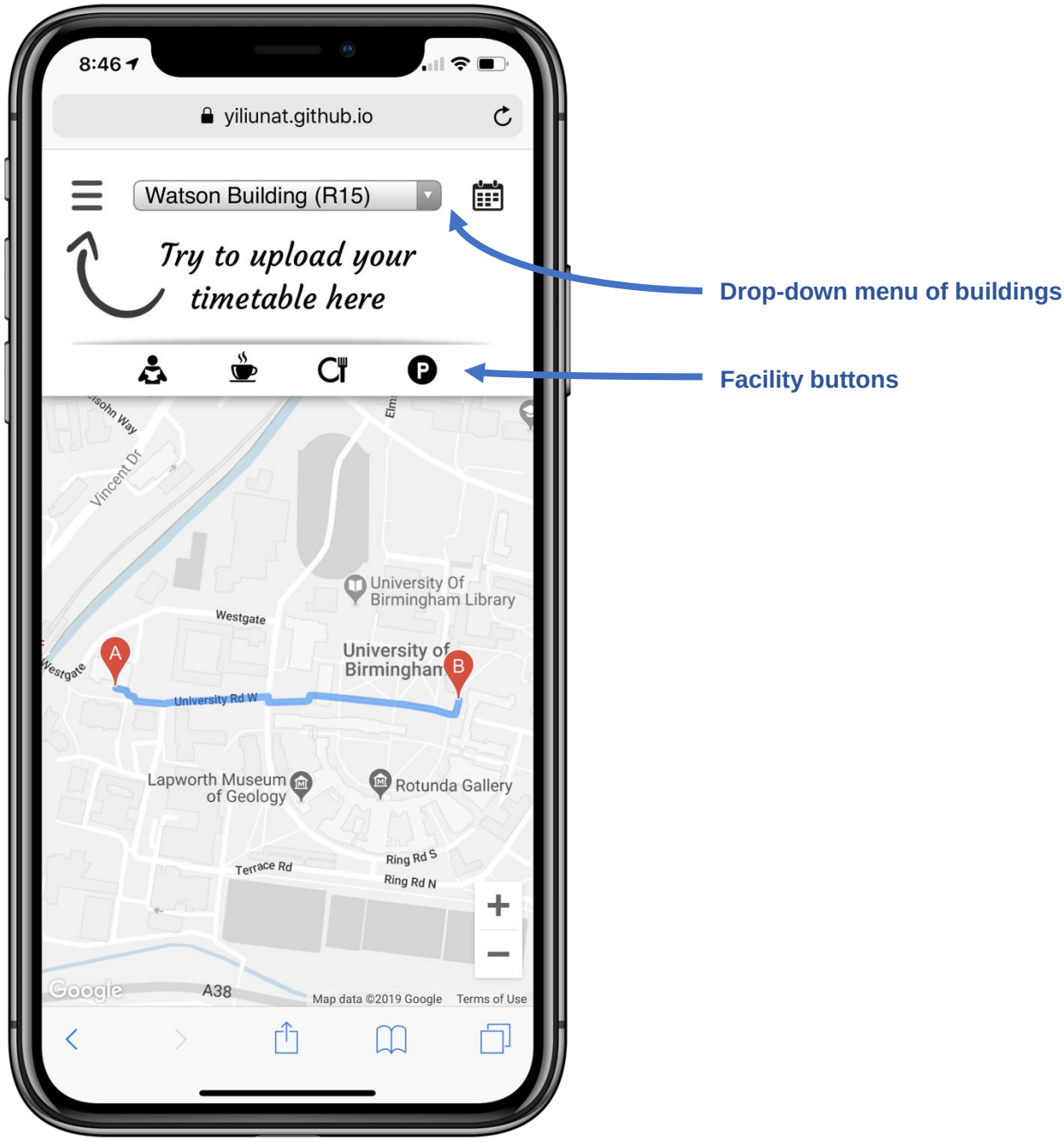


Figure 29. Interface of Visitor Mode

4.11.2 Facility Buttons

After users tap one of the facility buttons, the system will show the corresponding locations on the maps. All the facility buttons were designed as the toggle button, which means you can turn it on or off. The number of locations could be added on the latlngData.js file.

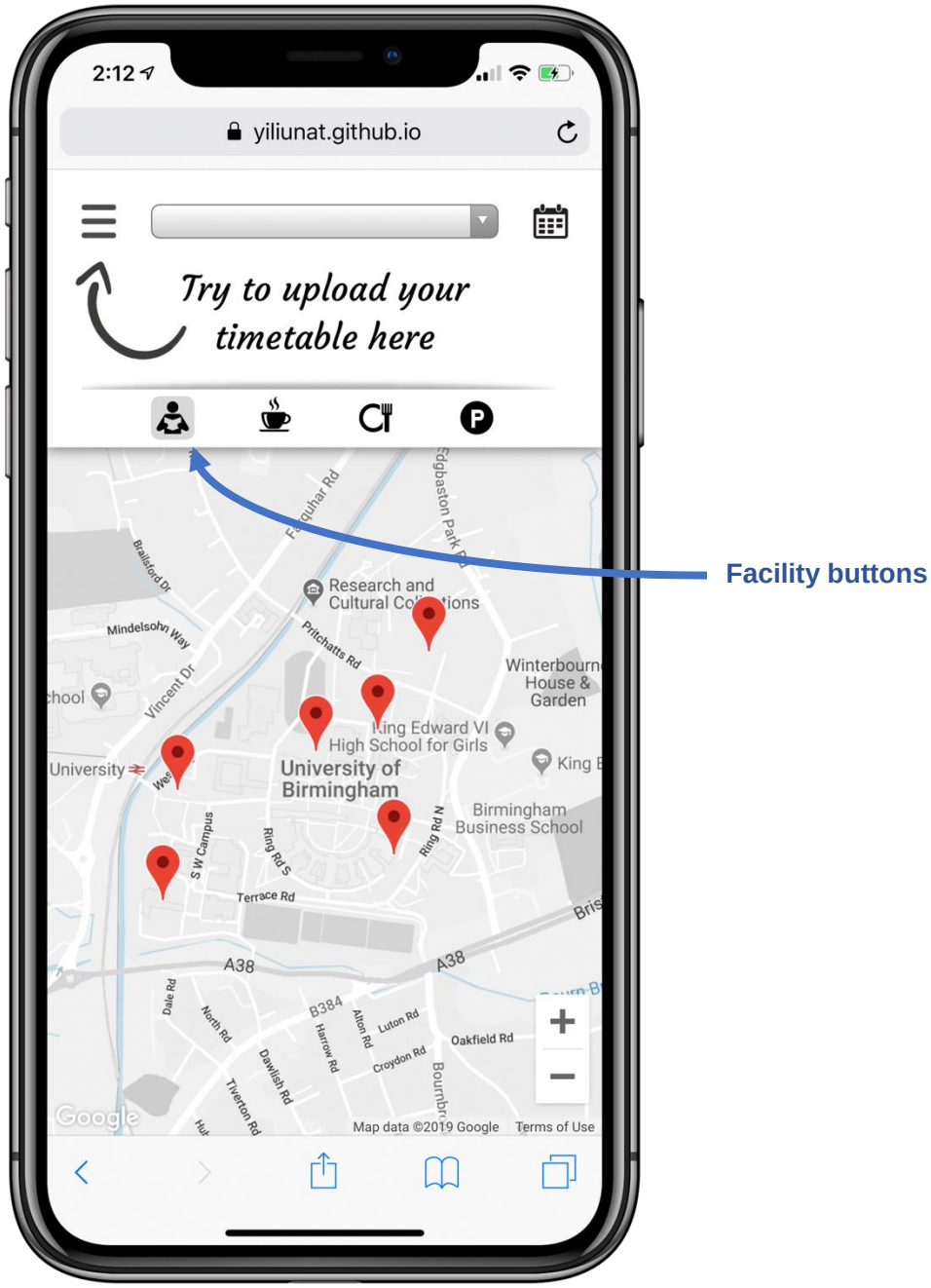


Figure 30. Interface of Facility Buttons

4.11.3 Multiple Entrance Option

If the building has multiple entrances, the system will pop-up a window and ask you to select your preferred entrance with their picture. Once you selected your entrance, the system would provide the path precisely to that selected entrance.

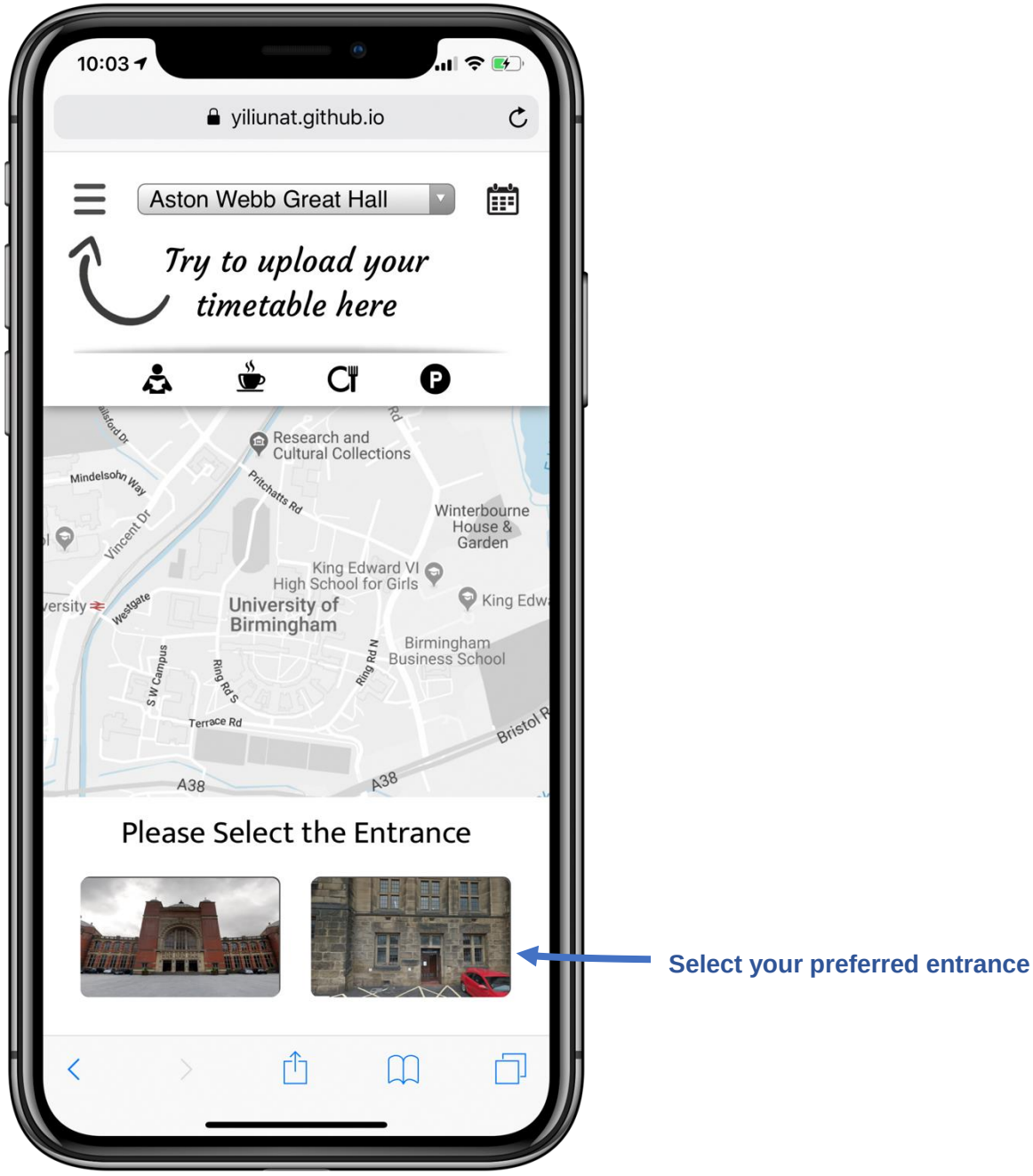


Figure 31. Interface of Multiple Entrance Tab

4.11.4 Timetable Uploading

This sidebar allows users to upload their timetables. The timetable should download from my.bham timetable page and save to iCloud files (iPhone) or system memory (Android).

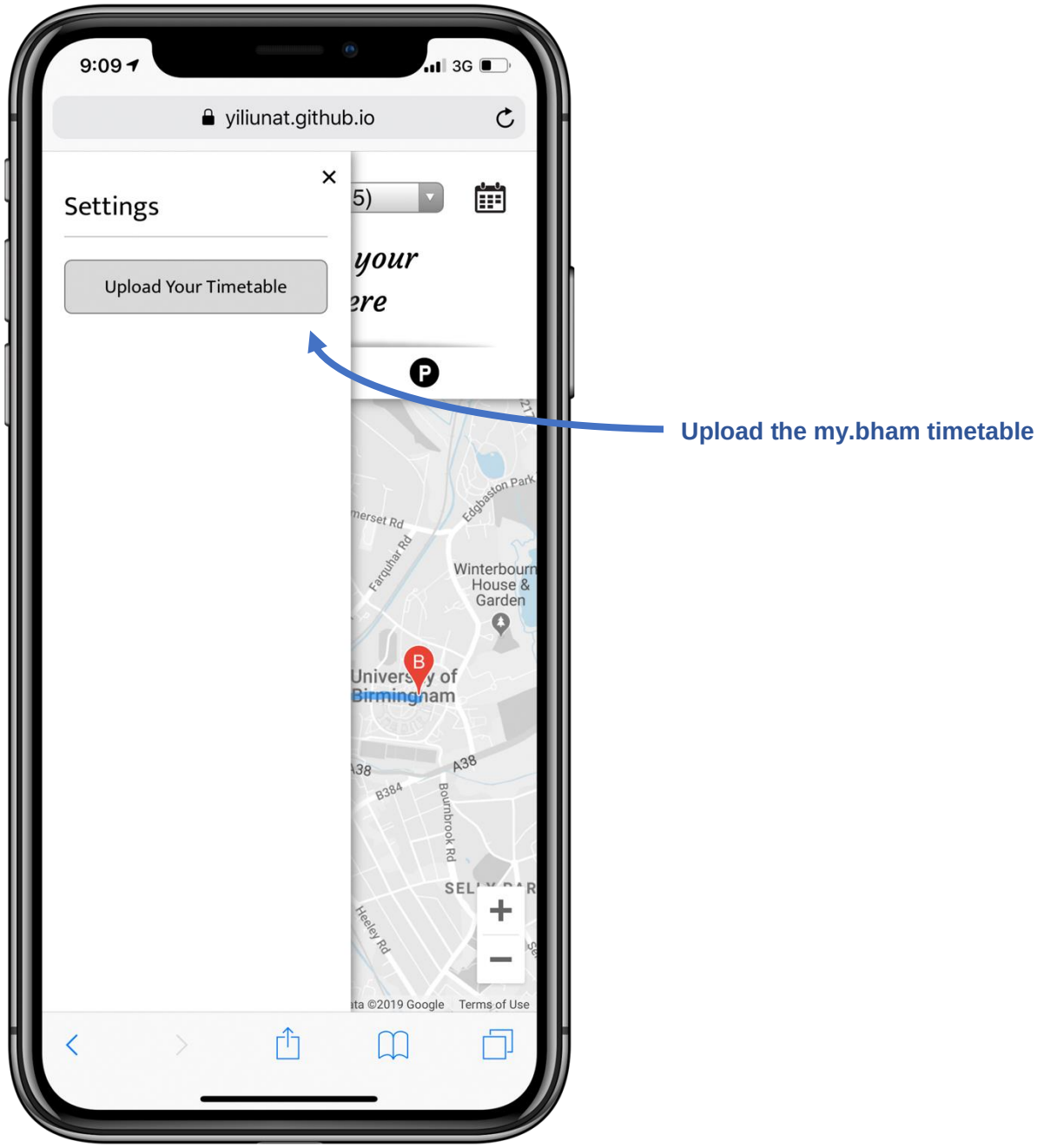


Figure 32. Interface of Timetable Uploading

4.11.5 Timetable Display

Once a timetable is uploaded, the system will store the timetable data in Local Storage, and load it automatically every time the user opens this system. A list of all lectures will be parsed out from the uploaded timetable and shown in a sidebar.

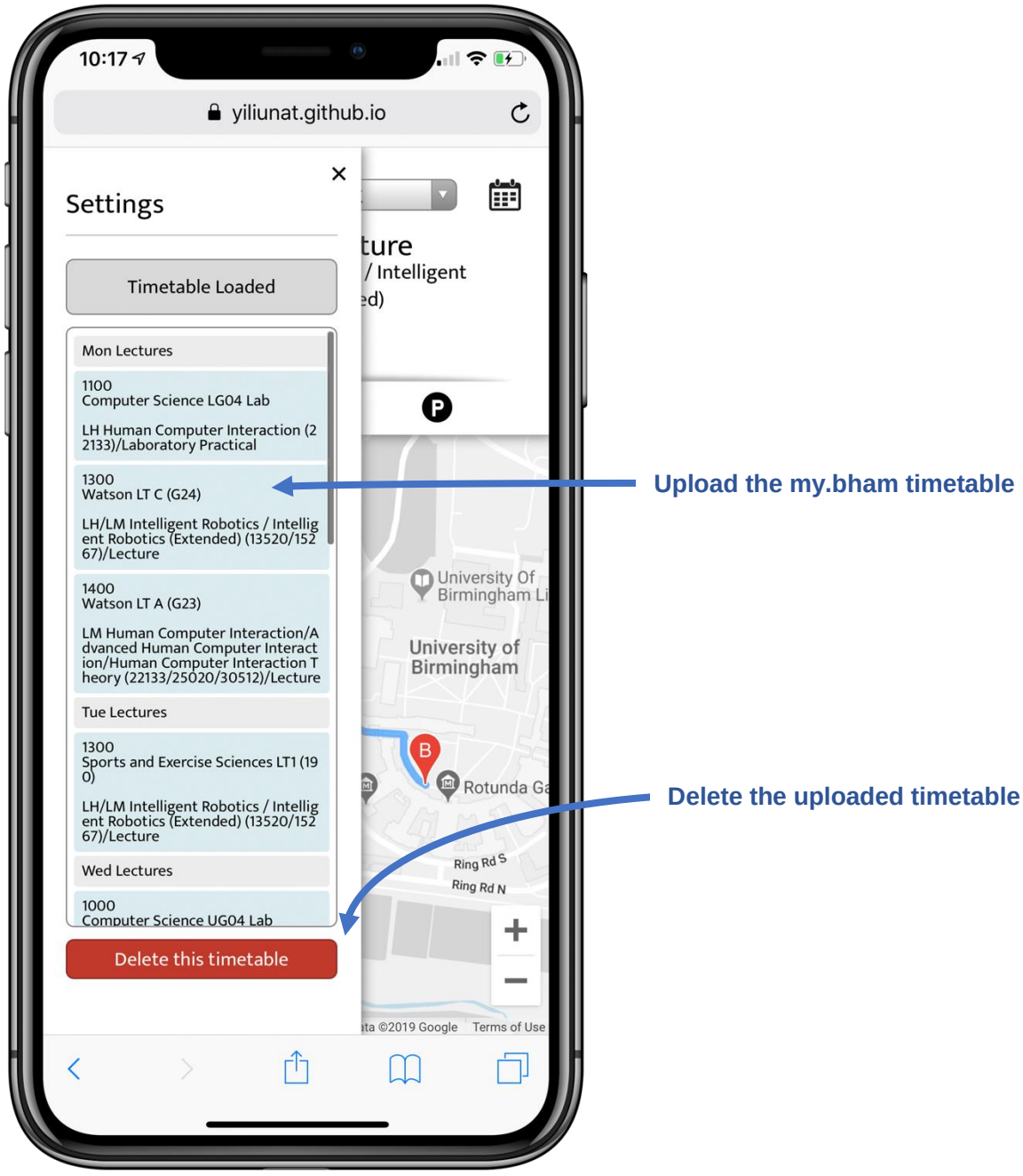


Figure 33. Interface of Timetable Sidebar

4.11.6 Lecture Reminder

Lecture reminder will remind the user of the next available lecture with the lecture's name, time and location. The system will automatically update to the newest lecture according to the time, without requiring users to manually refresh the page.

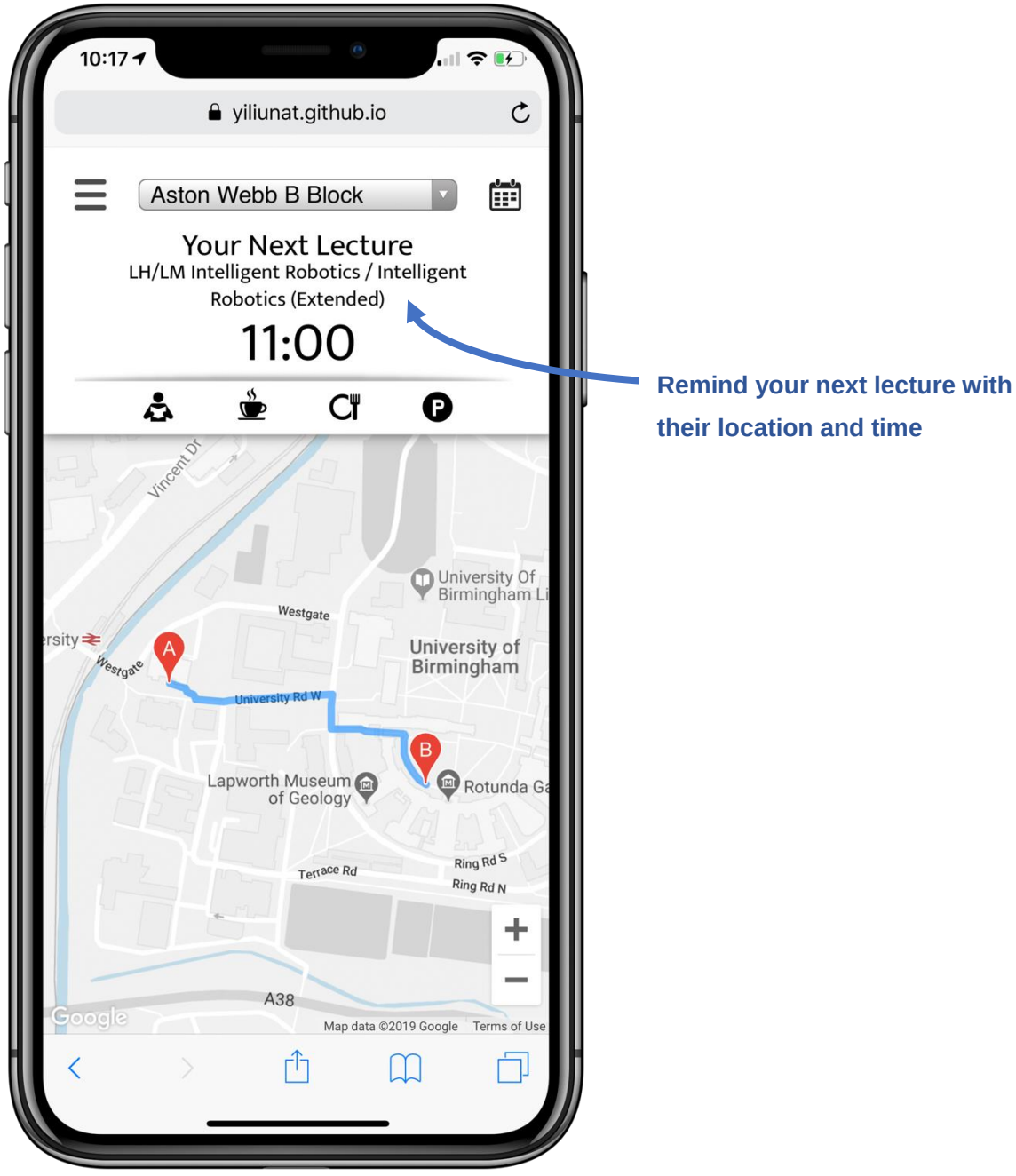


Figure 34. Interface of Lecture Reminder

5 Methodology

This project is based on a user-centred design. The reason for choosing this approach is because user-centred design is focused on usability and desirability [32]. Since no one has tried to do this project before, more user involvements are required to prove the usability for the project. The main difference of user-centred design is it focused on the iteration of prototyping and evaluation. There are three main steps in the user-centred design [33]:

1. Establish requirements: identify the target users of the product, understand the reasons and conditions for use.
2. Design solutions: Propose solutions based on Step 1; the design process may contain multiple stages, building from a rough concept to a complete design.
3. Prototyping and Evaluating: demonstrate the designs to users and evaluate the usability of current design.

After each evaluation, the new user requirements or improvements should be added until it reaches the final product. The existing designs will be considered for modification based on the users' feedback, and the correctness and efficiency of every design of this project should be proven.

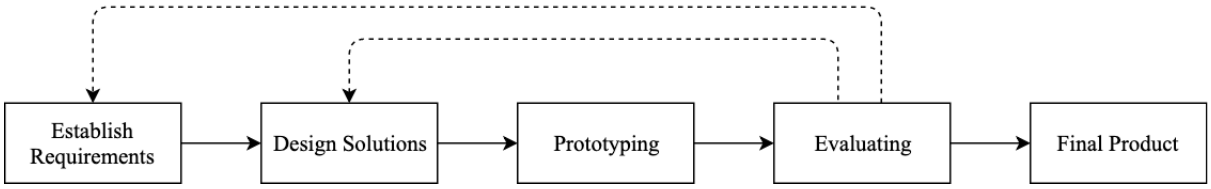


Figure 35. User-Centred Design

During the whole implementation process, several cycles of evaluation were carried out, among which the following improved functions were obtained in the final version of the system: (1) the system contains a complete copy of users' timetable information, and this mobile-friendly system can replace the existing timetable system (my.bham); (2) the system provides multiple enhanced functions of campus navigation; (3) the updated user interface improves efficiency.

See the following sections: 'User Evaluation (Formative)', 'User Evaluation (Summative)' and 'System Evaluation' for details on implementation of this design method.

5.1 Project Plan

The following is the weekly plan and progress of the project. The project plan is divided into three parts that correspond with the first semester, Christmas holiday and the second semester. The three formative user evaluation was conducted after 12th November 2018.

Semester 1	
Week	**Task**
1st October	Determine the idea and content of project; conduct brief background research and feasibility study.
8th October	Identify the development methodology (User Centred Design) and develop the user requirements.
15th October	Propose solutions and prepare the proposal of project.
22nd October	Begin literature reviewing, starting with interaction logic and map framework considerations.
29th October	Begin programming, starting with the construction of Google Maps API v3 service framework.
5th November	Correct the campus building coordinates and build a path planning system that will use an automatic refresh function.
12th November	First release of system: Complete the implementation of the home page and multiple entrance selection. The home page contains the maps framework and destination selection.
19th November	First Evaluation: Conduct the evaluation of first release of system, evaluate the initial interaction design and improve the design based on user's feedback.
26th November	Finalise the literature review and prepare the demonstration.
3rd December	Inspection week: demonstrate the proposed solution and programming progress.

Table 7. Project Plan of Semester 1

Christmas Holiday	
Week	**Task**
31st December	Produce the structure of time comparison algorithm, and design the logic of "lecture reminder".
7th January	Second release of system: Complete the implementation of "lecture reminder" and the structure of timetable datasets.

Table 8. Project Plan of Christmas Holiday

Semester 2	
Week	**Task**
14st January	Second Evaluation: Conduct the evaluation of the second release of system; evaluate the lecture reminder function and improve the design based on users' feedback.
21th January	Study and establish the method for accessing timetables.
28th January	Complete the implementation of the local storage function.
4nd February	Third release of system: Complete the design and implementation of my.bham timetable reading function (lecture parser).
11th February	Third Evaluation: Conduct the evaluation of third release of the system; evaluate the timetable reading function and improve the design based on user's feedback.
18th February	Complete the structure and basic conception of dissertation.
4th March	Forth release of system: Improve the interaction design based on user's feedback; create a side bar setting panel to display the timetable; create several facility interaction buttons.
11th March	Summative Evaluation: Conduct the summative evaluation of forth release of system; evaluate the timetable reading function and finalise the design based on user's feedback.
18th March	Demonstration week: present the project.
25rd March	Finalise the program to ensure all requirements have been achieved; finalise the dissertation to ensure all content has been included.

Table 9. Project Plan of Semester2

6 User Evaluation (Formative)

The formative evaluation were implemented through the whole design process, and different methods of evaluation were used at different stages.

The formative evaluation mainly focused on the design period of sketches and prototypes. There are four reasons for formative user evaluation in this project:

1. Check that the designs meet users' needs.
2. Compare different designs and find out which one works best.

3. Observe the effects of a specific interface on users. How does the user choose various designs?
4. Determine the usability of the current designed interactions and make sure the chosen designs are the best.

In this project, there have been four major releases, which with three formative evaluations and one summative evaluation. I will firstly explain the formative evaluations in this part, then explain the summative evaluation in the next part.

6.1 First Evaluation

The first formative user evaluation follows the development of sketches. It contains the original ideas of the campus navigation system, which includes the solutions for corrected navigation and the multiple entrance function, but without combining the timetable.

6.1.1 Sketches (First Evaluation)

The 'Idea 1.0' is the most basic idea for this project: a dataset of corrected building results with the navigation function. The navigation function will plan the route from the user's geolocation to the requested destination and refresh the planned route as the user moves.

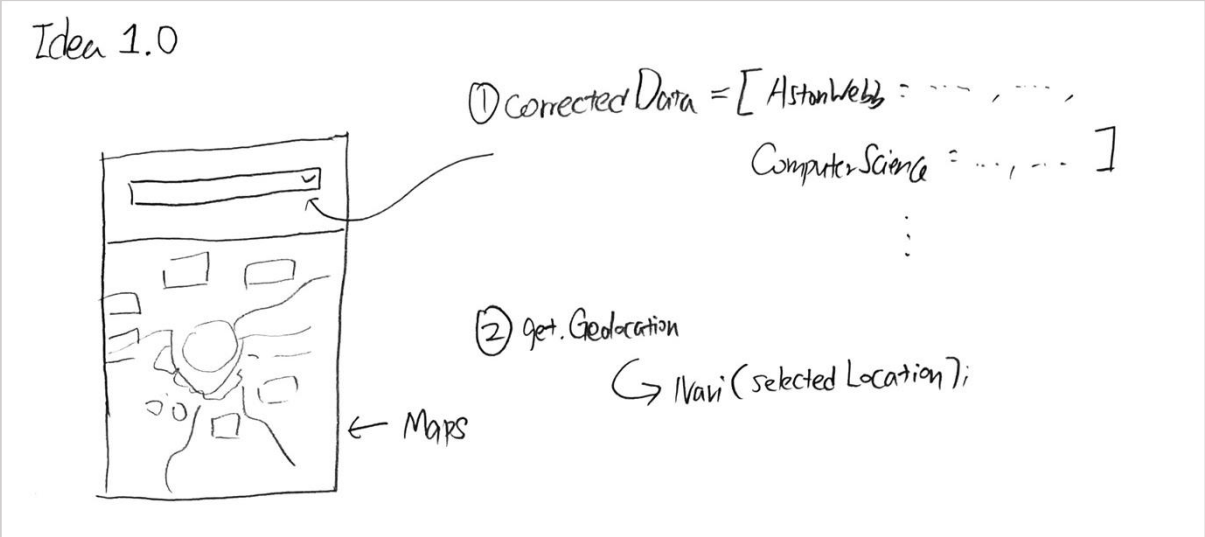


Figure 36. Basic Idea of Navigation

I had implemented this version at first, but it was too simple to conduct a user evaluation; I only described the idea to some users and demonstrated it to my supervisor. I received an improved idea, requesting that I solve the multiple entrance problem. I proposed two different solutions for the multiple entrance problem. The first idea was the implementation of a settings panel with entrance options, where users could select their preferred entrance and get the results based on their selected entrance. The second idea was the inclusion of a pop-up window which

asks students to select their entrance every time a building with multiple entrances has been selected.

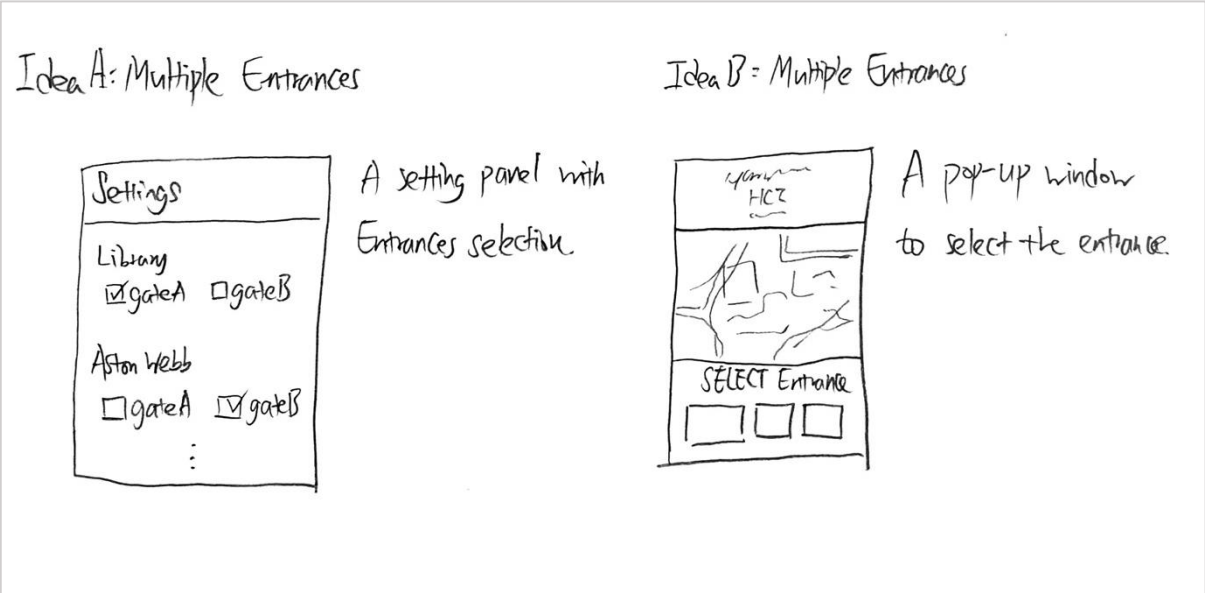


Figure 37. Idea of Multiple-entrances

After I surveyed some students from school of computer science, I got some ideas of pros and cons for both designs. I decided to select "Idea B" as the final design, because (1) it does not require users switch between multiple pages, and (2) users' selections may be different each time. I summarised the pros and cons in the following table:

Ideas	Survey	Advantages	Disadvantages
Idea A	2/10	1. Once users select their preferred entrance, it will save the results and plan the route automatically based on saved entrance. 2. Users are not required to select entrances frequently.	1. It is inconvenient if users want to temporarily change the entrance of current navigation. 2. Users need to switch between pages. 3. User's selection may be different each time.
Idea B	8/10	1. It does not require users to switch between pages frequently. 2. Users could select different entrance based on their route. 3. It only takes a very short time to make the selection.	1. Users need to select the entrance every time a building with multiple entrances has been selected.

Table 10. Pros & Cons of Schemes for Multiple-entrances Problem

6.1.2 Prototypes (First Evaluation)

This is the prototype of the first release, which contains the prototypes of two main functions: navigation and selection from multiple entrances. The main point of the prototype was to show users the design, watch them navigate the prototype, and assess how well users understand how to navigate the system. Since the design was evaluated during the sketch phase, the interaction logic of this version was easily understood by users. Because this was only a prototype, some users did not understand the meaning of those two grey boxes, but they quickly understood with a little explanation.

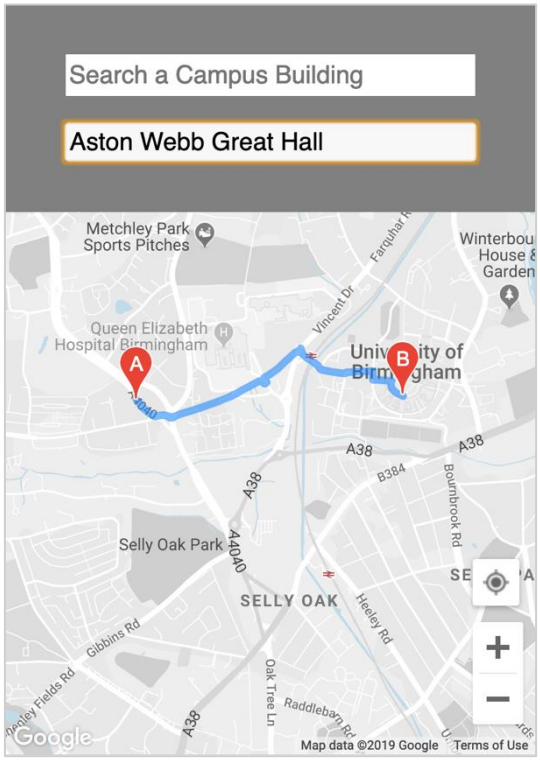


Figure 38. Navigation Function (First Prototype)

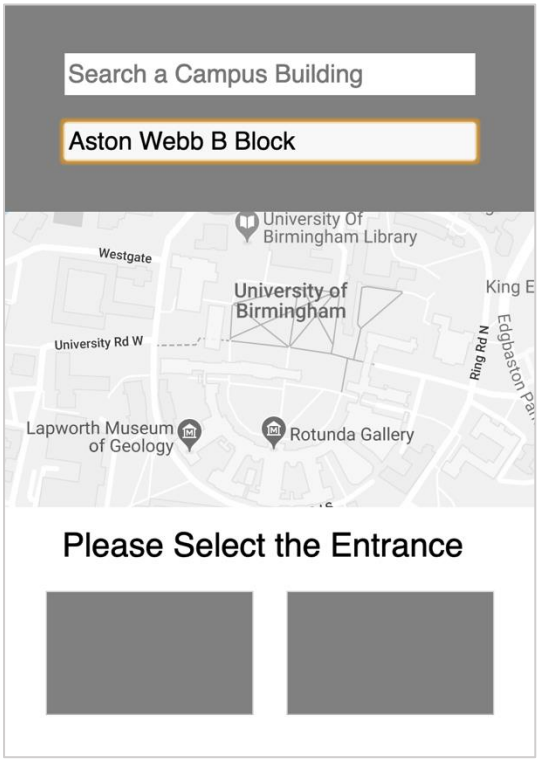


Figure 39. Multiple-entrances Function (First Prototype)

The feedback of this version was the simplicity of the system; therefore, the idea of "lecture reminder" came out, which combined the navigation with students' timetables. The additional new requirements were the following:

1. The system should display the time and building for the student's next lecture based on the stored data.
2. The path of the next lecture should be displayed automatically.

6.2 Second Evaluation

The second formative user evaluation follows the development of the main programme (January). It contains the basic implementation of the lecture notification function, which includes the building navigation function and a lecture reminder function, but without combining timetable recognition.

6.2.1 Sketches (Second Evaluation)

According to the new additional requirements from the previous section, 'Idea 2.0' is an enhanced version of campus navigation. In this version, the 'reminder' function uses a time comparison algorithm to compare the current time with the lecture time, reminding students of their next lecture by citing the lecture name, start time and building name. The navigation function was also changed to provide path planning automatically based on the student's next lecture.

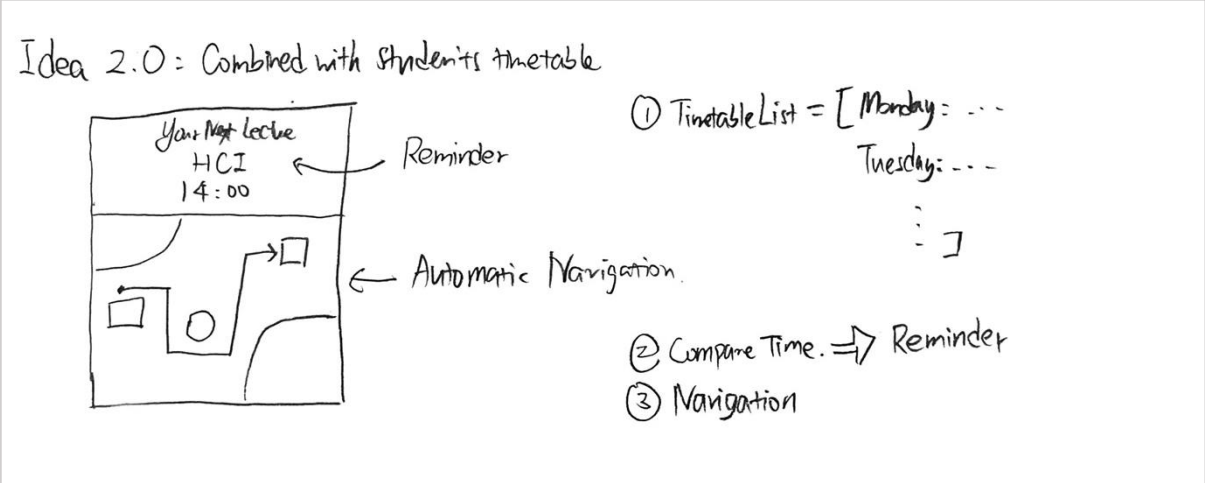


Figure 40. Enhanced Solution of Navigation

6.2.2 Prototypes (Second Evaluation)

This is the prototype of the second release, which does not contain the timetable recognition aspect. I manually wrote a timetable list to store the timetable information; however, users would also need to write their timetable information into the list to use this system, which is not user friendly. Therefore, I considered some alternative methods to access timetables (e.g. image recognition, PDF recognition and HTML recognition).

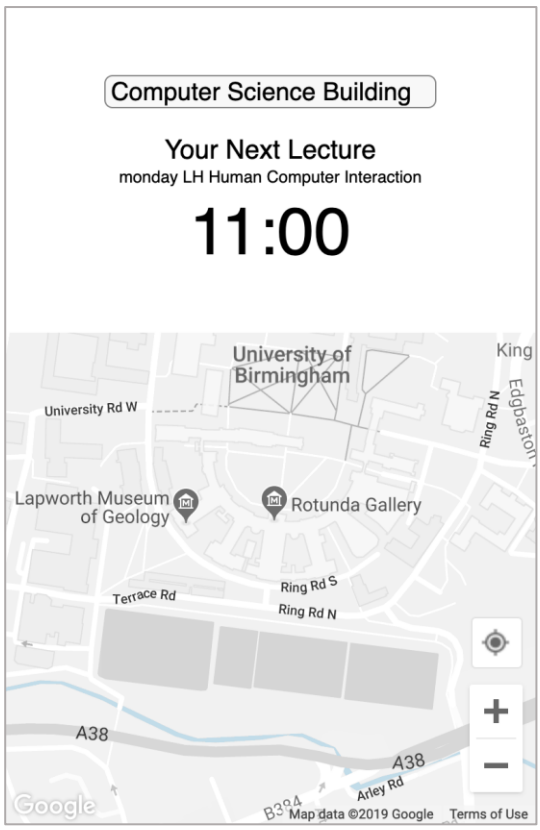


Figure 41. Lecture Reminder (Second Prototype)

```
var timeTable = [{
    day: 1,
    lectures: [{
        "time": 1100,
        "lect": "monday LH Human Computer Interaction",
        "location": "ComputerScience"
    }]
    //lectures: [undefined]
},
{
    day: 2,
    lectures: [{
        "time": 1100,
        "lect": "Intelligent Robotics / Intelligent Robotics(Extend)",
        "location": "SportsandExercise"
    }]
},
{
    day: 3,
    lectures: [{
        "time": 1400,
        "lect": "LM/LH Networks",
        "location": "SportsandExercise"
    }]
},
{
    day: 4,
    lectures: [{
        "time": 1600,
        "lect": "Thur Human Computer Interaction",
        "location": "ComputerScience"
    }]
},
{
    day: 5,
    lectures: [{
        "time": 2000,
        "lect": "20:00LH Human Computer Interaction",
        "location": "ComputerScience"
    },{
        "time": 2100,
        "lect": "21:00 lecture",
        "location": "ComputerScience"
    },{
        "time": 2200,
        "lect": "22:00 lecture",
        "location": "ComputerScience"
    },{
        "time": 2300,
        "lect": "23:00 lecture",
        "location": "ComputerScience"
    }
    ]
    //lectures: [undefined]
}
];
```

Figure 42. Timetable List (Second Prototype)

The feedback for this version concerned experience problems when using the system; therefore, the idea of 'timetable reading' came out. This allows students to upload their timetable and the system will automatically parse all the students' lectures. The new additional requirements were the following:

1. Timetable storing
    a) The system should allow students to upload a timetable and save the results.
    b) The system should automatically load the timetable every time users open it.
    c) The system should allow students to change and delete the timetable.
2. Timetable reading
    a) The system should be able to read students' timetables and parse out all the students' lectures.
    b) The system should understand the relationship between the time and day of different lectures in the context of the timetable.

6.3 Third Evaluation

After several studies (section 4.6), I decided to use HTML recognition, mainly because OCR and PDF recognition have poor format recognition, but also because HTML recognition is more efficient.

6.3.1 Sketches (Third Evaluation)

According to the newly added requirements mentioned in the previous section, the third release of the system contains complete timetable storing and reading functions, which allow students to upload a timetable and save the results to local storage. Additionally, the functionality of timetable reading was enhanced to allow the system to understand the relationship between the times and days of the uploaded my.bham timetable to parse all lectures in the form of a JavaScript list.

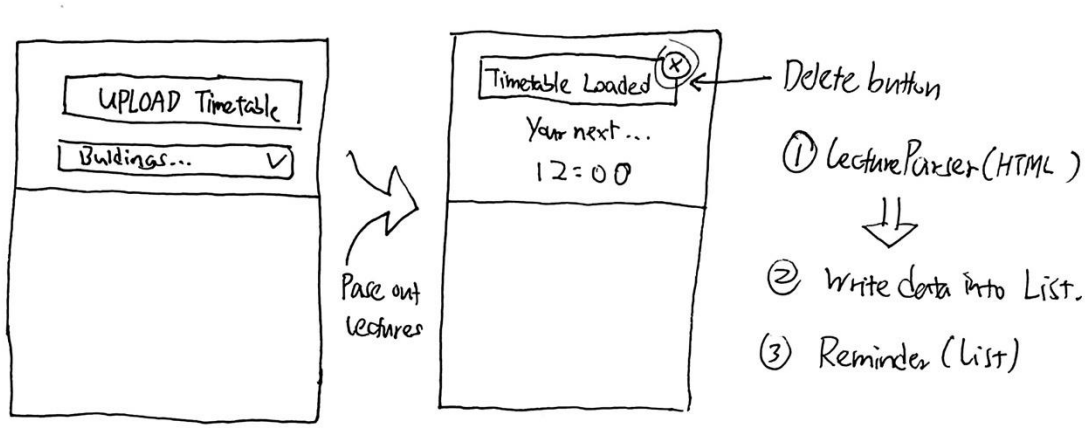


Figure 43. Idea of Timetable Reading

6.3.2 Prototypes (Third Evaluation)

This is the prototype of third release, which is very close to the final version. However, the user interface still needed to be improved. For example, the map frame is too small because the upper lecture panel occupies a large amount of space. The solution could be a settings panel for the uploading button and timetable. Also, some suggestions advised adding a campus facility recommendation function.

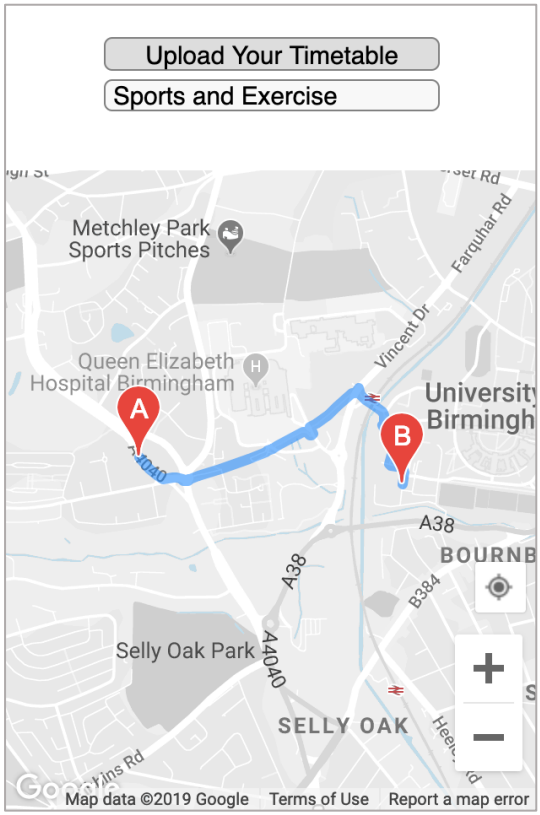


Figure 44. Timetable Uploading (Third Prototype)

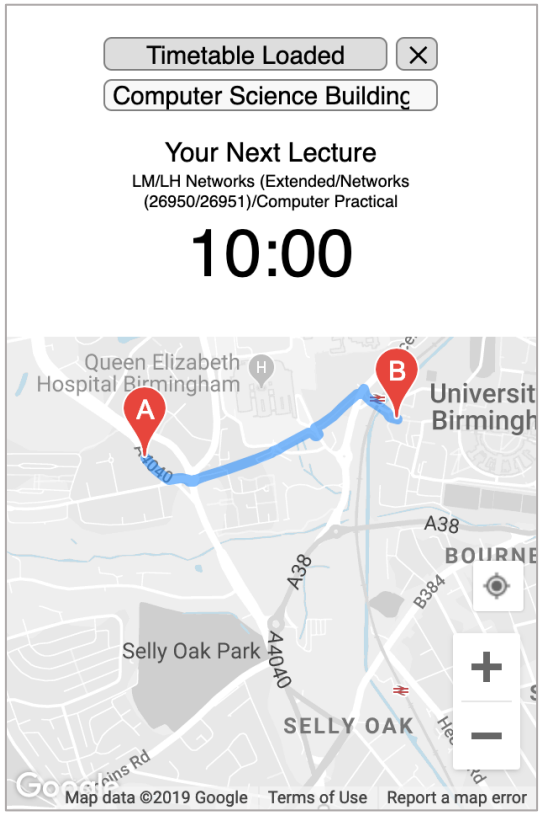


Figure 45. Timetable Reading (Third Prototype)

The new additional requirements were the following:

1. The layout of the home page should be optimised.
2. The system should provide a full list of lectures and time based on students' timetable, to help students check their prior and ongoing lectures.
3. The system should suggest the campus facility.

7 User Evaluation (Summative)

The summative user evaluation is the evaluation after all the designed functions were finished, nearly reaching the end of the development cycle. I divided my summative user evaluation into two parts. First, I wanted to find out which one of two different interaction designs of the home page is better for the user experience; second, there were several general research questions that needed to be solved through the summative user evaluation.

7.1 Interfaces

After several formative evaluations, all functional requirements had almost been achieved. In the last release following the third formative evaluation, we found that there were still

deficiencies in the home page interface design; therefore, a user evaluation was conducted regarding the following two interface designs:

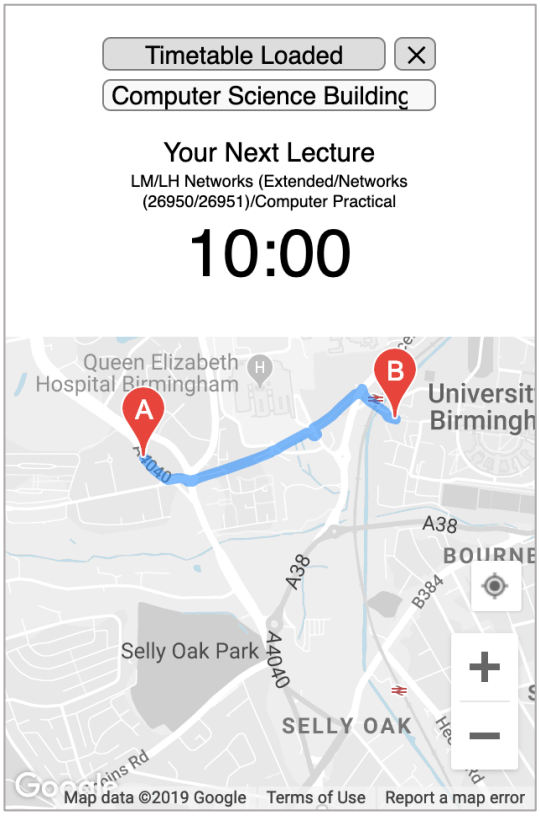


Figure 47. Original Homepage Interface

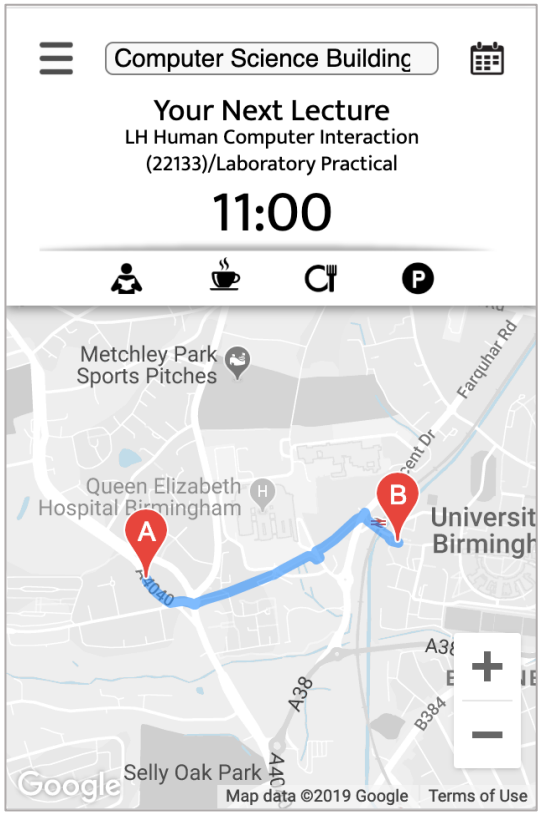


Figure 46. New Homepage Interface

The timetable uploading function was integrated into a sidebar (the top-left button) and the extra space was used to display facility buttons. The new design allows more content to be displayed, but the uploading function requires an extra operation and switching between pages. Therefore, a user evaluation was conducted.

7.1.1 Hypothesis (Interface Problem)

A hypothesis is a prediction about what will happen. Two one-tailed hypotheses were devised based on this interface problem, which were: (1) Users will achieve tasks more smoothly when using the new interface compared to the old one. (2) Users will achieve tasks more smoothly when using the old interface compared to the new one.

7.1.2 Tasks (Interface Problem)

Since this interface mainly affects the uploading behaviour, I designed a simple task to test my hypothesis: upload students' my.bham timetable to this system. This task has two conditions:

1. Use the original interface to upload my.bham timetable.
2. Use the new interface to upload my.bham timetable.

There are two ways to complete these tasks: within-subject design and between-subject design. Within-subject design means that all test users test all conditions (for both the new interface and the old interface). Between-subject design means that one group of users' tests one condition (the new interface) and another group of users tests another condition (the old interface). The drawback of a within-subject design is the order effects, which need to be counterbalanced; the drawback of a between-subject design concerns the differences between participants, which may influence the results [34]. Since it is hard to avoid the bias caused by individuals when using the between-subject design, I used a within-subject design to implement the task.

7.1.3 Findings (Interface Problem)

Designs	Very Dissatisfied	Dissatisfied	Satisfied	Very Satisfied
Old Interface	2	5	3	0
New Interface	0	1	4	5

Table 11. Survey Results of Interface Designs

Accept / Reject	Hypothesis	Reasons
ACCEPT	HYPOTHESIS 1: Users will achieve tasks more smoothly when using the new interface compared to the old one.	1. Timetable uploading is not a frequent operation. Once uploaded, users do not need to upload it again unless their timetable has changed. 2. New interface has a larger maps frame, which is important for a navigation application.
REJECT	HYPOTHESIS 2: Users will achieve tasks more smoothly when using the old interface compared to the new one.	1. Single interface makes it easier to upload, but this is not a routine operation. 2. It takes up more space than the new design.

Table 12. Results of Hypothesis

7.2 Research Questions

Research questions are the reasons for conducting research. Apart from the interface problem, there are several general research questions to be researched. I had three main questions for this research:

1. Do the interactive features in this system improve student's efficiency?
2. Are the download/upload processes of timetable too complicated?
3. Will students replace their timetable by using this system?

7.3 Experiment Design

In order to answer the three questions above, an experiment was designed for this round of user testing:

1. Test visitor mode.
   1.1 Select a destination (University Library), and use the navigation function from a location (University Station).
   1.2 Test the facilities recommendation function.
2. Download timetable from my.bham website.
   2.1 The downloaded timetable should be an HTML file.
3. Test student mode
   3.1 Upload the downloaded timetable and check the correctness of parsed lectures.
   3.2 Change the calendar days and time on the right side to check it they match the uploaded timetable.
   3.3 Try to use this system for 3 days.

Because timetables can vary greatly from student to student, in order to verify the compatibility and robustness of the system for different timetables, I required test users to download their own timetable through my.bham (the university's timetable website) as the independent variable of user evaluation.

7.4 Findings & Improvements

During the experiment, I noticed that some test users tuned on their browser's private mode, which could cause the browser to delete the cache (loaded timetable) automatically. Therefore, I added a new notification in README to ensure that privacy mode would be turned off before testing.

Also, some users ran the tests on an old version of Android phones, where browsers have different compatibilities to some elements of JavaScript statements. This caused the system to be incorrectly performed on their Android phones. Since mobile browsers do not provide the console log, it is difficult to locate which statement went wrong, so to solve this problem, I

downloaded an Android virtual machine on my computer and used the remote control function to connect the console logs to the Google Chrome browser. The debugging process took up a lot of time, but the problem was finally solved.

One of designed tasks was requiring test users to check the correctness of their recognised timetable, and some users complained about the difficulty of checking. Therefore, I designed this function to help. After the timetable reading, this function will change the layout of the sidebar and display all lectures in a categorized format from Monday to Friday. This feature is designed to help users check the correctness of timetable reading, and improve the user, allowing users to replace their my.bham timetable by using this combined system directly.

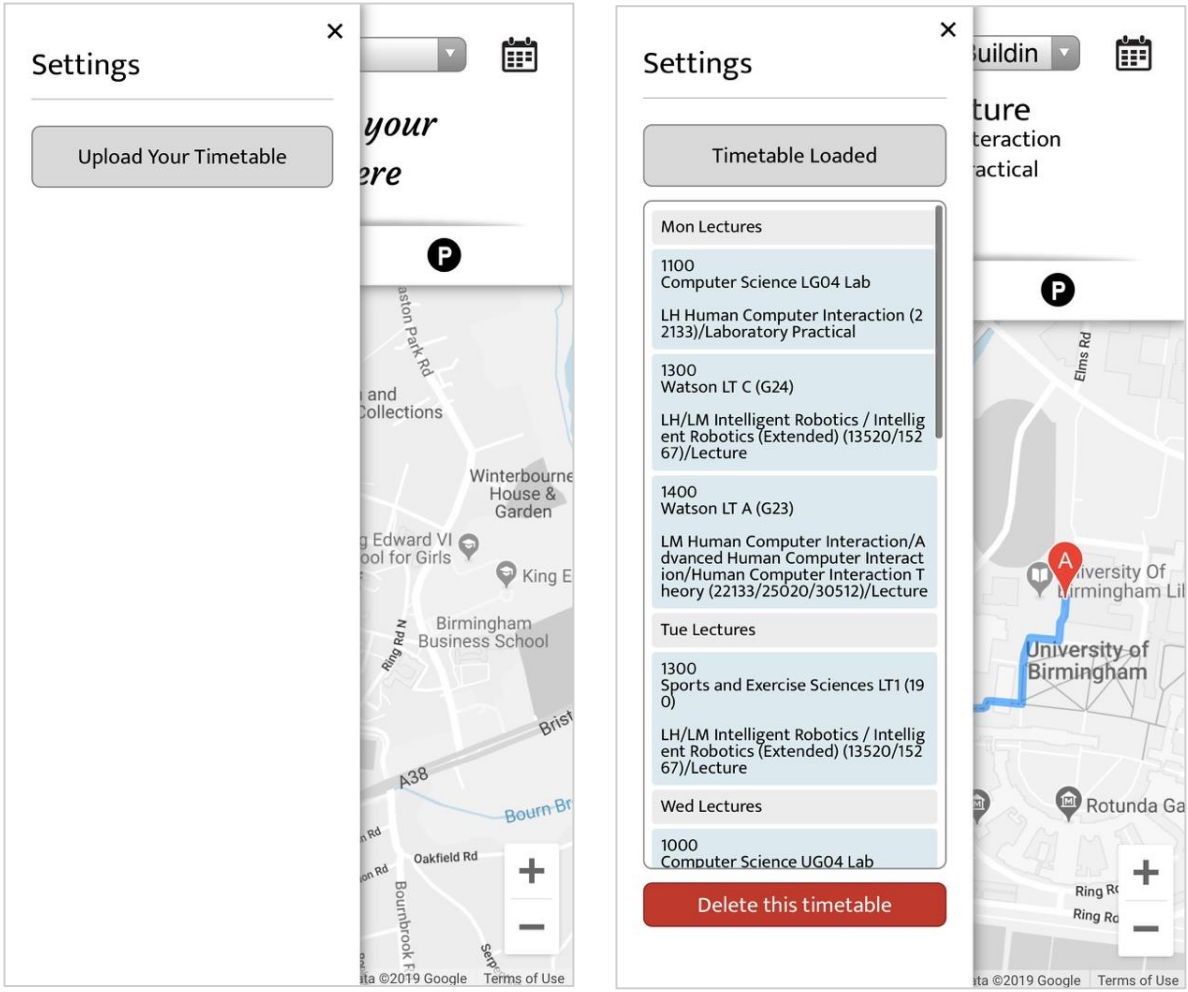


Figure 48. Timetable Display Function

I summarised the main problems and solutions in the following table:

Problems	Descriptions	Solutions
Private mode	The local storage cache (timetable) is wiped if users turned on the private mode of their browser.	Add a notification in readme to ensure that private mode is turned off before use.
Android browser compatibility	The system cannot perform properly on some Android browsers.	Use virtual machine to remote control the console log to debug.
Correctness checking	After a timetable is uploaded, users cannot easily check whether the timetable has been correctly read.	Add a frame in the sidebar to display timetable.

Table 13. Problem & Solution Summary of Summative User Evaluation

7.5 Questionnaires

To ensure that all daily tasks of this system could be easily performed by users, I created a survey to collect the data. A total of 5 users took part in the user test, and each user completed this survey after the test.

Questions	Strongly Disagree	Disagree	Agree	Strongly Agree
The navigation function in visitor mode could be used as an alternative to navigation products.	0	1	2	2
Efficiency of usage is higher than navigation application + my.bham timetable.	0	0	1	4
The difficulty of timetable download is NOT acceptable.	1	3	1	0
The lecture reminder function is clear and intuitive.	0	0	2	3
Would you like to replace your timetable with this system?	0	1	0	4

Table 14. Questionnaire Results of User Test

8 Technical Evaluation

The technical evaluation section was divided into two parts: (1) investigate and determine the completeness of all intended objectives and features and (2) test the impact of different inputs, parameters, and environments on the system. The first part will be presented as black box testing, which tests the relevant events from user's perspective. The second part will be subdivided into several sections that will be introduced later.

8.1 Black Box Testing

No.	Test Description	Expected Outcome	Actual Outcome
1	Visitor mode is enabled before user upload their timetable.	The home page will be changed to visitor mode if no timetable detected.	PASS – The visitor mode was successfully switched.
2	All visitor mode items are correctly loaded in visitor mode.	The drop-down menu, facility buttons, sidebar and upload button will be successfully loaded.	PASS – All items were loaded correctly.
3	All student mode items is hidden correctly in visitor mode.	The lecture reminder, clear button and timetable frame will be successfully hidden.	PASS – All items were hidden correctly.
4	Google Maps API framework is correctly loaded.	The maps framework will be loaded once users open the programme.	PASS – Maps API was loaded correctly.
5	The drop-down menu with navigation system works successfully.	The map function will plan the corresponding path as a user selects a destination.	PASS – Drop-down menu and navigation system successfully work.
6	Planned path was correctly rendered in the map frame.	The planned path will navigate users directly to the building entrance.	PASS – Planned path correctly rendered.
7	Multiple entrance function works properly.	The selection panel will pop up once a user selects a multiple-entrance building.	PASS – Multiple-entrance panels was correctly prompted.
8	The path planning function could automatically refresh.	The system automatically refreshes the path as a user moves a certain distance.	PASS – The refresh function of path planning successfully work.

9	Maps viewport did not change if user drags the map.	The path planning refresh function will not re-scale the frame if a user drags the map.	PASS – Maps viewport was retained.
10	Users can click a facility button to get corresponding locations.	System will correctly push locations to the screen.	PASS – Facility locations were displayed correctly.
11	Timetable upload button worked properly.	Upload button will successfully upload a user's my.bham timetable.	PASS – Timetable upload button works successfully.
12	Timetable reading function works successfully.	All lectures will be parsed out once a my.bham timetable is uploaded.	PASS – All lectures were parsed out.
13	Local storage (timetable saving) function works correctly.	All parsed lectures will be stored in local storage and automatically loaded every time it is opened by a user.	PASS – All lectures were stored in local storage and loaded correctly.
14	Timetable frame correctly display lectures from the uploaded timetable.	Timetable frame display all parsed lectures.	PASS – All lectures were correctly displayed.
15	Lecture reminder correctly prompt the ongoing lecture.	System compare the current time with lecture time and correctly outputs the lecture.	PASS – Lecture reminder correctly displayed.
16	Automatic path planning function works properly.	The path planning function updates with a lecture reminder.	PASS – Path planning function was updated with a lecture reminder.
17	Calendar test tool works correctly.	The lecture notification will change with selected calendar button.	PASS – Calendar test tool was updated correctly.
18	Timetable delete function work properly.	System will refresh the page automatically and return to visitor mode.	PASS – Timetable was deleted and system returned to visitor mode.

Table 15. Results of Black Box Test

8.2 Parameters and Variables

In this system, there are several main parameters and variables that affect the outputs of the system. I will assess the underlying problems of these parameters and variables and determine whether or not I need to correct them. I will introduce them in this section.

8.2.1 Timetable

Student timetables may produce varied results of the system outputs because there are slight differences in the code of each timetable. The parsing function should be able to analyse all types of timetables, and we need to determine whether the system is compatible with different timetables. For this section, I tested several variations of timetables, and I will present my test results as the following:

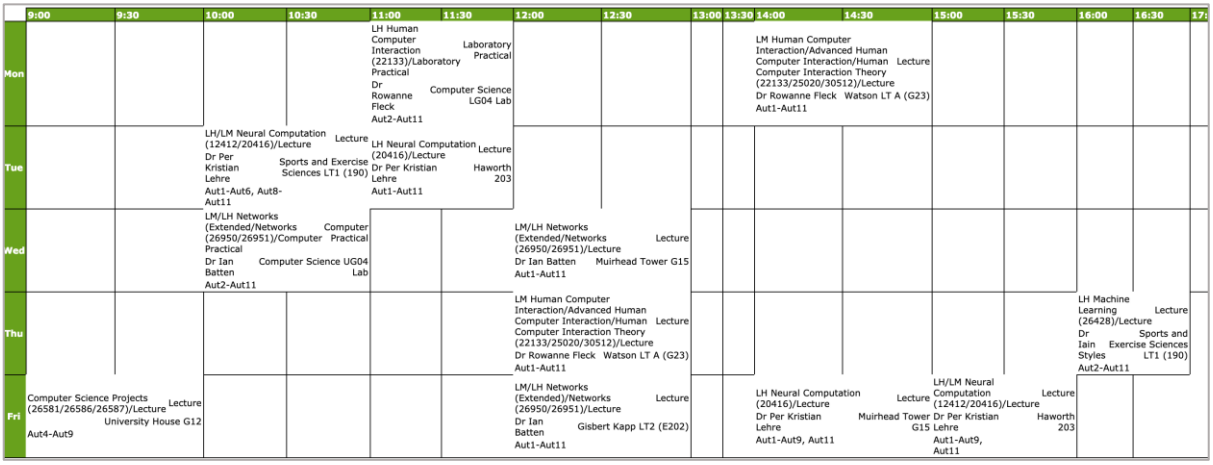


Figure 49. Timetable Sample 1

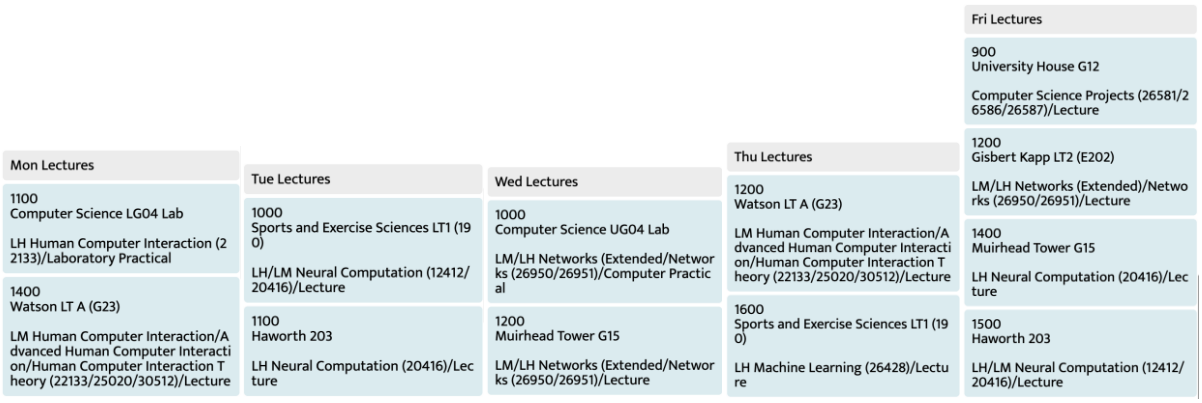


Figure 50. Parsing Results of Timetable Sample 1

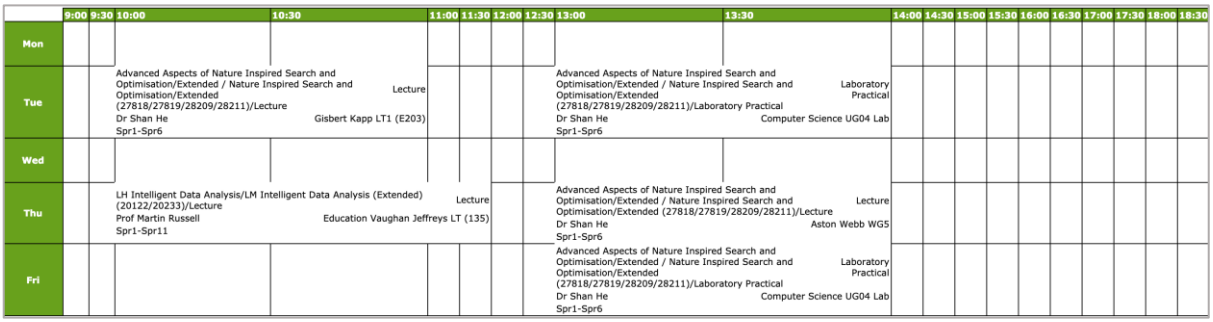


Figure 51. Timetable Sample 2

Mon Lectures

Tue Lectures

1000
Gisbert Kapp LT1 (E203)

Advanced Aspects of Nature Inspired Search and Optimisation/Extended / Nature Inspired Search and Optimisation/Extended (27818/27819/28209/28211)/Lecture

1300
Computer Science UG04 Lab

Advanced Aspects of Nature Inspired Search and Optimisation/Extended / Nature Inspired Search and Optimisation/Extended (27818/27819/28209/28211)/Laboratory Practical

Wed Lectures

Thu Lectures

1000
Education Vaughan Jeffreys LT (135)

LH Intelligent Data Analysis/LM Intelligent Data Analysis (Extended) (20122/20233)/Lecture

1300
Aston Webb WG5

Advanced Aspects of Nature Inspired Search and Optimisation/Extended / Nature Inspired Search and Optimisation/Extended (27818/27819/28209/28211)/Lecture

Fri Lectures

1300
Computer Science UG04 Lab

Advanced Aspects of Nature Inspired Search and Optimisation/Extended / Nature Inspired Search and Optimisation/Extended (27818/27819/28209/28211)/Laboratory Practical

Figure 52. Parsing Results of Timetable Sample 2

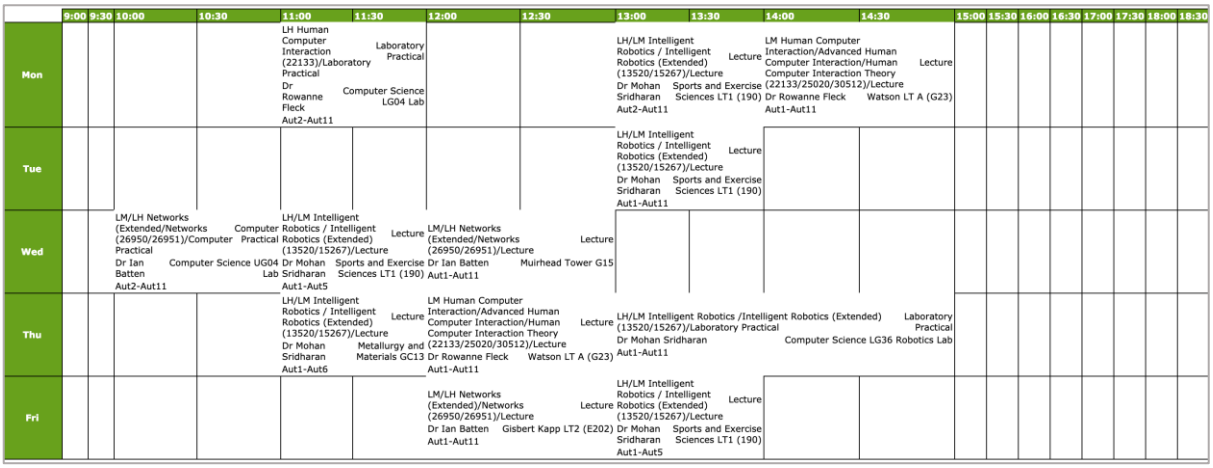


Figure 53. Timetable Sample 3

Mon Lectures

1100
Computer Science LG04 Lab

LH Human Computer Interaction (22133)/Laboratory Practical

1300
Sports and Exercise Sciences LT1 (190)

LH/LM Intelligent Robotics / Intelligent Robotics (Extended) (13520/15267)/Lecture

1400
Watson LT A (G23)

LM Human Computer Interaction/Advanced Human Computer Interaction/Human Computer Interaction Theory (22133/25020/30512)/Lecture

Tue Lectures

1300
Sports and Exercise Sciences LT1 (190)

LH/LM Intelligent Robotics / Intelligent Robotics (Extended) (13520/15267)/Lecture

Wed Lectures

1000
Computer Science UG04 Lab

LM/LH Networks (Extended/Networks (26950/26951)/Computer Practical

1100
Sports and Exercise Sciences LT1 (190)

LH/LM Intelligent Robotics / Intelligent Robotics (Extended) (13520/15267)/Lecture

1200
Muirhead Tower G15

LM/LH Networks (Extended/Networks (26950/26951)/Lecture

Thu Lectures

1100
Metallurgy and Materials GC13

LH/LM Intelligent Robotics / Intelligent Robotics (Extended) (13520/15267)/Lecture

1200
Watson LT A (G23)

LM Human Computer Interaction/Advanced Human Computer Interaction/Human Computer Interaction Theory (22133/25020/30512)/Lecture

1300
Computer Science LG36 Robotics Lab

LH/LM Intelligent Robotics /Intelligent Robotics (Extended) (13520/15267)/Laboratory Practical

Fri Lectures

1200
Gisbert Kapp LT2 (E202)

LM/LH Networks (Extended/Networks (26950/26951)/Lecture

1300
Sports and Exercise Sciences LT1 (190)

LH/LM Intelligent Robotics / Intelligent Robotics (Extended) (13520/15267)/Lecture

Figure 54. Parsing Results of Timetable Sample 3

I summarised the test description and results in the following table:

No.	Description	Results
Timetable Sample 1	This timetable represents a typical week of a current student of UoB. The lectures distribute across the week from Monday to Friday without spare weekdays.	PASS – The parsing function was successfully outputted all lectures.
Timetable Sample 2	This timetable represents a student who has less workload (more free time). The lectures are only distributed on Tuesday, Thursday and Friday.	PASS – The parsing function was successfully outputted all lectures.
Timetable Sample 3	This timetable represents a student who has more workload (less free time). The timetable has 13 lectures per week.	PASS – The parsing function was successfully outputted all lecture.

Table 16. Summary of Timetable Parsing Results

8.2.2 Resolutions

Different resolutions of the screen may produce different results of the layout of elements. The adaptive rem-layout function should be able to ensure that no elements would be obscured under most resolution settings. We need to determine whether the system is compatible with different screen sizes and resolutions. It should be noticed that existing mobile devices usually do not render the webpage content with actual screen resolutions, but based on the DPI of the screens [35]. For this section, I tested different resolutions and summarised the results as the following:

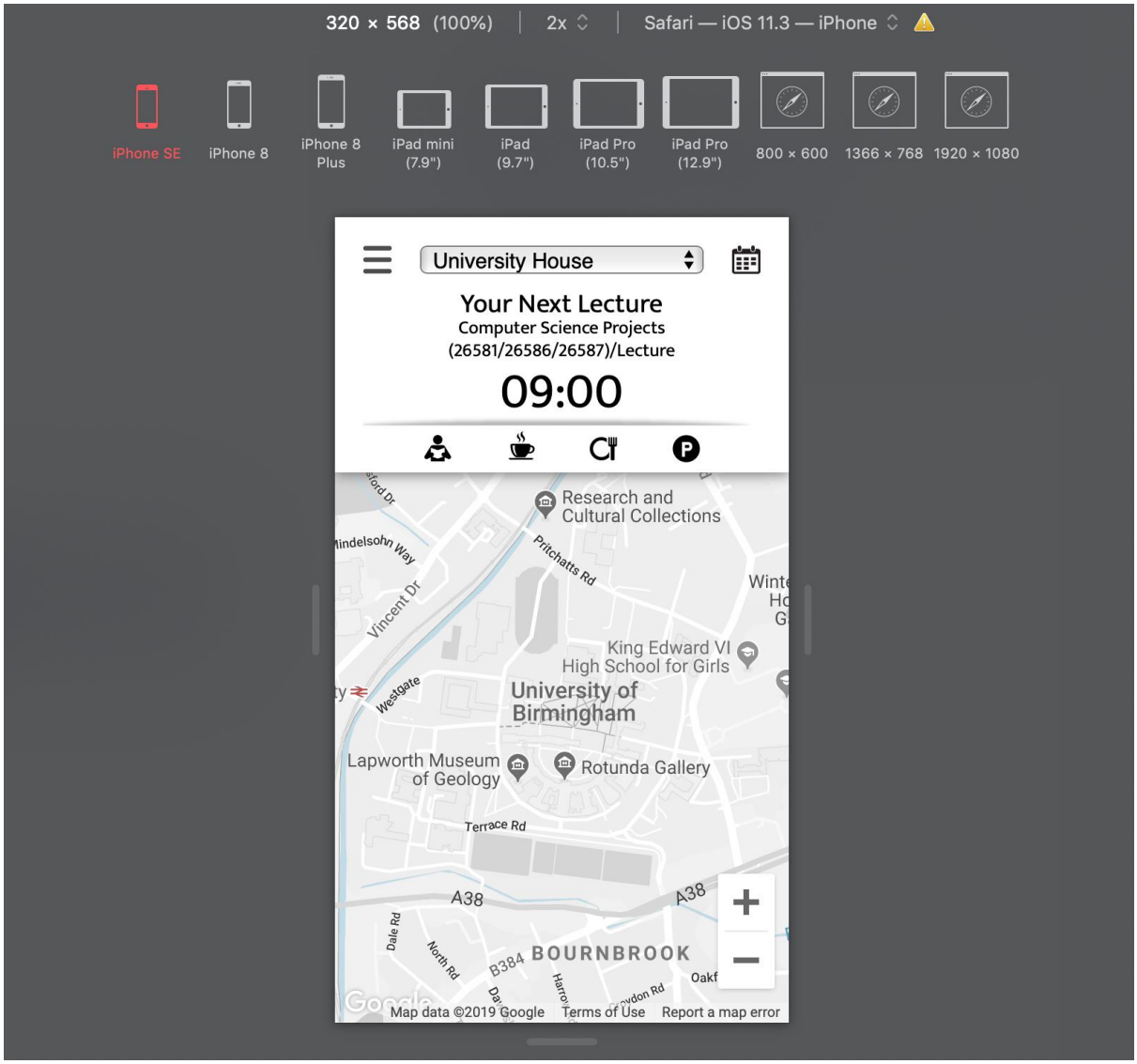


Figure 55. Resolution Tests

Render Resolution	Typical Models	Results
320*568px	iPhone 5, iPhone 5s and iPhone SE	PASS – No elements were obscured.

375*667px	iPhone 6, iPhone 6s, iPhone 7 and iPhone 8	PASS – No elements were obscured.
414*736px	iPhone 6 Plus, iPhone 7 Plus, and iPhone 8 Plus	PASS – No elements were obscured.
768*1024px	iPad mini and iPad 9.7''	PASS – No elements were obscured.
834*1112px	iPad Pro 10.5''	PASS – No elements were obscured.
1024*1366px	iPad Pro 12.9''	PASS – No elements were obscured.
240*320	Android Phones	PASS – No elements were obscured.
320*480		PASS – No elements were obscured.
400*800px		PASS – No elements were obscured.
540*960px		PASS – No elements were obscured.
720*1280px		PASS – No elements were obscured.
1080*1920px		PASS – No elements were obscured.
1440*2560px		PASS – No elements were obscured.

Table 17. Results of Resolution Tests

8.2.3 Browser Compatibilities

Different browsers may have different compatibilities of CSS, JavaScript compilation and JavaScript libraries. The system should be able to support all modern browsers, and we need to determine whether the system is fully and correctly compatible with different browsers. For example, some CSS statements in Firefox should have a prefix of "-moz-", but in safari would become "-webkit-". For this section, I tested multiple browsers, and I will present my test results as following:

Browsers	Render Engines	Results
Chrome	Webkit	PASS – All system functions were supported in this browser.
Safari	Webkit	PASS – All system functions were supported in this browser.
Firefox	Gecko	PASS – All system functions were supported in this browser.

Opera	Blink	PASS – All system functions were supported in this browser.
Internet Explorer	Trident & Chakra	PASS – All system functions were supported in this browser.
Microsoft Edge (Windows)	Edge HTML & Chakra	PASS – All system functions were supported in this browser.
Microsoft Edge (iOS)	Webkit	PASS – All system functions were supported in this browser.
Microsoft Edge (Android)	Blink	PASS – All system functions were supported in this browser.

Table 18. Results of Browser Compatibility Tests

9 Discussion & Conclusion

Completion of this project has motivated the execution of a series of studies on campus navigation and produced a robust solution. All initial and additional requirements of the project have been achieved by applying the system to carry out all designed tasks for visitors and students. An extensive reading of the literature and multi-dimensional studies helped to find positive results for different functions in order to design the solution. The combination of functions, evaluations of the solution, and results of the compatibility test were reflected the usability and reliability of the system in response to different situations.

There is seldom a campus map application that can be frequently used by students. The solutions of this project were obtained by addressing users' needs and conducting repeated evaluations, which resulted in providing long-term assistance for students and visitors.

Nevertheless, there are some limitations for this project; the accuracy of navigation depends on the selected map API. Although I selected the map provider with the highest accuracy on the UoB campus, there are still deficiencies in the path planning. Firstly, some of the suggested paths are not optimal. Secondly, all coordinates of buildings were set as the entrance of the building; however, as some buildings are far away from the road, the map API can only pinpoint the nearest road in front of the entrance. Further improvements to the above problems could be obtained by an own drawn map or an extra overlay layer to enhance the paths. Another limitation is the compatibility of old phones; while all modern browsers work with the system, some users may access it on older browsers with outdated phones, which can suffer from the browser incompatilities.

There are several additional features that I would have implemented given more time. The initial proposal **[Appendix 3]** mentioned an additional functionality of providing an updated train schedule for the university station. This function could further drive user engagement with the system. Another feature would be an overlay layer on the map frame to show the orientation of the user. This function could make the system more user friendly.

In summary, although there are still deficiencies that need to be addressed and improvements that could be made, the project itself has achieved positive results in a limited timeframe. If someone wants to implement a similar system, the research results and solutions of this project would greatly benefit them.

For more information about how to run the programme, please read the **[Appendix 1]** and **[Appendix 2]**.

10 Reference

[1] D. Suleiman, “Mobile shopping mall navigator,” no. March 2017, 2012.

[2] B. Ozdenizci, V. Coskun, and K. Ok, “NFC internal: An indoor navigation system,” *Sensors (Switzerland)*, vol. 15, no. 4, pp. 7571–7595, 2015.

[3] A. Möller, L. Roalter, S. Diewald, M. Kranz, and R. Huitl, “A mobile indoor navigation system interface adapted to vision-based localization,” p. 1, 2012.

[4] T. H. Kolbe, “Augmented Videos and Panoramas for Pedestrian Navigation,” *Geowissenschaftliche Mitteilungen*, vol. 66, no. Azuma 1997, pp. 45–52, 2003.

[5] W. Narzt *et al.*, “Augmented reality navigation systems,” *Univers. Access Inf. Soc.*, vol. 4, no. 3, pp. 177–187, 2006.

[6] H. Alqahtani, “iMAP-CampUS : Developing an Intelligent Mobile Augmented Reality Program on Campus as a Ubiquitous System,” pp. 1–5.

[7] P. Dahne and J. N. Karigiannis, “Archeoguide: System architecture of a mobile outdoor augmented reality system,” *Proc. - Int. Symp. Mix. Augment. Reality, ISMAR 2002*, pp. 263–264, 2002.

[8] C. H. Lin, Y. Chung, B. Y. Chou, H. Y. Chen, and C. Y. Tsai, “A novel campus navigation APP with augmented reality and deep learning,” *Proc. 4th IEEE Int. Conf. Appl. Syst. Innov. 2018, ICASI 2018*, pp. 1075–1077, 2018.

[9] K. J. C. Ang, S. L. Carag, D. E. M. Jacinto, J. O. R. Lunasco, and M. K. Cabatuan, “Design and development of Google glass-based campus navigation system,” *J. Telecommun. Electron. Comput. Eng.*, vol. 10, no. 1–5, pp. 75–81, 2018.

[10] C. M. Ş. An, A. Popa, and R. Comes, “AUGMENTED REALITY IN UNIVERSITY CAMPUS INTERACTIVE ORIENTATION,” vol. 61, pp. 153–158, 2018.

[11] A. S. Pagare, H. O. Pal, S. A. Patil, and V. M. More, “An Event Driven Campus Navigation System On Android Platform,” vol. 03, no. 5, pp. 3–6, 2017.

[12] M. Bopp, D. Sims, S. A. Matthews, L. S. Rovniak, E. Poole, and J. Colgan, “There ’ s an app for that : development of a smartphone app to promote active travel to a college campus,” *J. Transp. Heal.*, vol. 3, no. 3, pp. 305–314, 2016.

[13] Koch Peter-Paul, “A tale of two viewports,” *Quirksmode*. [Online]. Available: https://www.quirksmode.org/mobile/viewports.html. [Accessed: 21-Nov-2018].

[14] Google Maps Platform, “Browser Support | Maps JavaScript API | Google Developers,” 2018. [Online]. Available: https://developers.google.com/maps/documentation/javascript/browsersupport. [Accessed: 20-Nov-2018].

[15] Apple, “MapKit JS - Maps - Apple Developer,” 2019. [Online]. Available: https://developer.apple.com/maps/mapkitjs/. [Accessed: 06-Mar-2019].

[16] “Bing Maps API Documentation.” [Online]. Available: https://www.microsoft.com/en-us/maps/documentation. [Accessed: 05-Apr-2019].

[17] RapidAPI, “The Top 10 Mapping & Map APIs (for Developers in 2018) | RapidAPI,” 2018. [Online]. Available: https://blog.rapidapi.com/top-map-apis/. [Accessed: 06-Mar-2019].

[18] “Versioning | Maps JavaScript API | Google Developers.” [Online]. Available: https://developers.google.com/maps/documentation/javascript/versions. [Accessed: 21-Mar-2019].

[19] Apple, “Navigation - App Store Downloads on iTunes,” 2019. [Online]. Available: https://itunes.apple.com/gb/genre/ios-navigation/id6010?mt=8. [Accessed: 06-Mar-2019].

[20] Statista, “• Top U.S. mapping apps by users 2018 | Statistic,” 2018. [Online]. Available: https://www.statista.com/statistics/865413/most-popular-us-mapping-apps-ranked-by-audience/. [Accessed: 06-Mar-2019].

[21] “Understanding Billing for Maps, Routes, and Places | Google Maps Platform | Google Developers.” [Online]. Available: https://developers.google.com/maps/billing/understanding-cost-of-use. [Accessed: 07-Apr-2019].

[22] Apple, “CLGeocoder - Core Location | Apple Developer Documentation,” 2019. [Online]. Available: https://developer.apple.com/documentation/corelocation/clgeocoder. [Accessed: 06-Mar-2019].

[23] “Latitude and Longitude Finder on Map Get Coordinates,” 2019. [Online]. Available: https://www.latlong.net/. [Accessed: 05-Apr-2019].

[24] Google, “LatLngLiteral interface of Google Maps API,” 2019. [Online]. Available: https://developers.google.com/maps/documentation/javascript/reference/coordinates#LatLngLiteral. [Accessed: 11-Mar-2019].

[25] “Ocrad.js - Optical Character Recognition in Javascript.” [Online]. Available: http://antimatter15.com/ocrad.js/demo.html. [Accessed: 05-Apr-2019].

[26] “Tesseract.js | Pure Javascript OCR for 62 Languages!” [Online]. Available: http://tesseract.projectnaptha.com/. [Accessed: 05-Apr-2019].

[27] R. R. Palekar, S. U. Parab, D. P. Parikh, and P. V. N. Kamble, “Real Time License Plate Detection Using OpenCV and Tesseract,” vol. 40055, pp. 2111–2115, 2017.

[28] M. A. Munson and J. S. Cross, “Deep PDF parsing to extract features for detecting embedded malware.,” no. September, 2011.

[29] J. Damerow, B. R. E. Peirson, and M. D. Laubichler, “The Giles Ecosystem – Storage , Text Extraction , and OCR of Documents,” 2017.

[30] “Window.localStorage - Web APIs | MDN.” [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage. [Accessed: 05-Apr-2019].

[31] M. Spinelli, “iScroll, smooth scrolling for the web,” 2019. [Online]. Available: https://github.com/cubiq/iscroll/blob/master/README.md. [Accessed: 15-Mar-2019].
[32] A. M. Ed and D. Hutchison, *Design , User Experience , and Usability*. 2014.
[33] “User-Centered Design Basics | Usability.gov.” [Online]. Available: https://www.usability.gov/what-and-why/user-centered-design.html. [Accessed: 19-Mar-2019].
[34] G. Charness, U. Gneezy, and M. A. Kuhn, “Experimental methods: Between-subject and within-subject design,” *J. Econ. Behav. Organ.*, vol. 81, no. 1, pp. 1–8, 2012.
[35] D. L. Huang, P. L. Patrick Rau, and Y. Liu, “Effects of font size, display resolution and task type on reading Chinese fonts from mobile devices,” *Int. J. Ind. Ergon.*, vol. 39, no. 1, pp. 81–89, 2009.
[36] Kezz Bracey, “Why You Should Be Using Rem-Based Layouts,” *Tutsplus*, 2016. [Online]. Available: https://webdesign.tutsplus.com/tutorials/why-you-should-be-using-rem-based-layouts--cms-27828. [Accessed: 21-Nov-2018].

APPENDIX 1: How to run the programme

Pre-requisites:

1. An Internet connection.
2. A computer or smart phone with any modern browser (Google Chrome is recommended).

* Make sure (1) the private mode of your web browser is turned off and (2) the location service is turned on.

Run:

Open "*/src/RUN_THIS/run.htm*" or visit https://yiliunat.github.io/fyp on your browser.

Test Scheme:

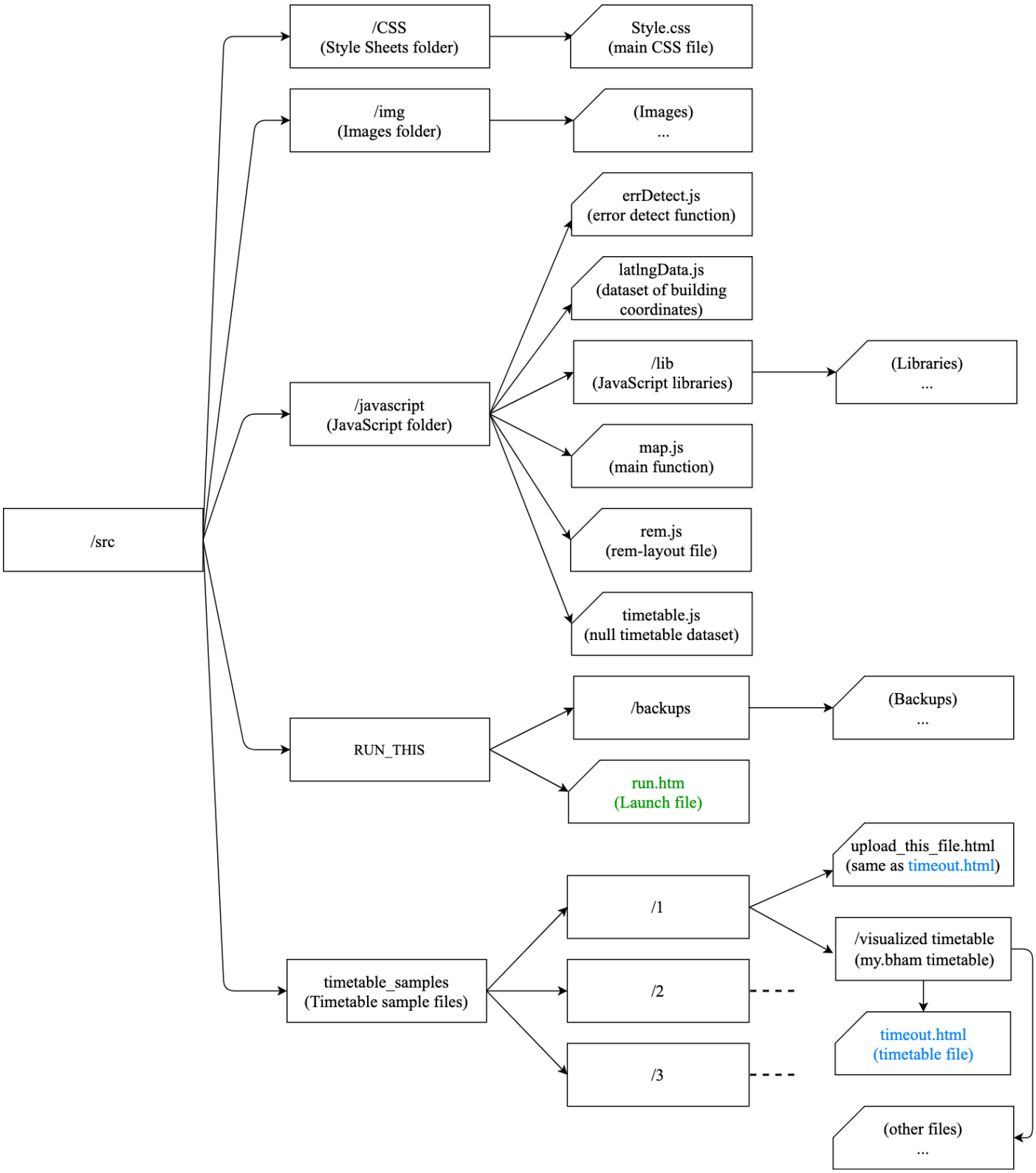
1. Basic functions:
   1.1 Click the drop-down menu and select any building to test the navigation function.
   1.2 To test the multiple entrance function, select any building mentioned in Note (II).
   1.3 Experiment with the different facility buttons by clicking them.
2. Timetable Upload:
   2.1 Open the sidebar (the top-left button), click "*upload your timetable*" button, select one of the my.bham timetable samples in "*/src/timetable_samples/*" and upload the file named "*upload_this_file.html*" (this is the same as the "*timeout.html*" file in the "*visualized timetable*" folder).
   2.2 "*Visualized timetable*" is the my.bham timetable with their style sheets. If you want to check the correctness of the loaded timetable, you can open "*timeout.html*" in any "*visualized timetable*" folder.
   2.3 Once you upload the timetable, you can refresh the page or close the browser, the uploaded timetable will be automatically loaded the next time you use the programme.
3. Check the correctness of the timetable parsing:
   3.1 The uploaded timetable will be loaded in the sidebar (the top-left button).
4. Check the correctness of the lecture reminder:
   4.1 Open the test panel (the top-right calendar button). Change the system time to see how the lecture reminder outputs differs.
5. Delete the uploaded timetable:
   5.1 Open the sidebar (the top-left button) and click the "*delete this timetable*" button.

Note:

(I) Not all the coordinates of UoB buildings were recorded in this system, such an implementation would be very time consuming and irrelevant to the computer science field.

(II) The following buildings are samples of multiple entrance building:
[*'Aston Webb Great Hall', 'Staff House', 'Library'*]

APPENDIX 2: File Tree



The Proposal for an Intelligent Campus Route Planning System

Final Year Project - Yi Liu - 1605486
School of Computer Science

User Research

After I surveyed some students from the school of Computer Science, I noticed students having difficulty locating their lecture room only using their timetable; I noticed this difficulty particularly for year 1 students. I deduced that student confusion is likely because the building name given in student timetables are not usually shown correctly on the mobile map, and sometimes simply cannot be found at all on the mobile map. Due to this issue, students typically use the university-offered online interactive campus map. However, students continue to alternate between timetable and map repeatedly on their phone. Most importantly, the online interactive map is not designed for mobile use and is not functional when used for navigation. Thus, the current campus route method is not sufficient for students who walk around campus. This is why I would like to make a campus route planning system.

Proposed Solution

The solution is a combined system of a timetable and a route planning system. This system will be a mobile web-app that connects with student timetables and plans a route automatically for students to their lecture building. Students will no longer consistently switch between timetable and map, and they can use this system on a mobile browser as they walk around campus. The additional features will include a parking lot display, a market and restaurant display, an updated train schedule to New Street, an overlay of corrected building names, and route planning for bicycles. The surveyed year 1 students suggested many of these solutions. Moreover, this is also a potential campus navigation application for UoB campus visitors.

Functions

1. An automatic route planning system for students.
2. A Search or drop-down menu to select a campus building and show a planned route from the user's geolocation.
3. A function that allows users to view their upcoming and previous lecture.
4. A route planning for bicycles.
5. An updated train schedule to New Street.
6. An overlay of corrected campus building names.
7. A display of parking lots, markets, cafés and restaurants.

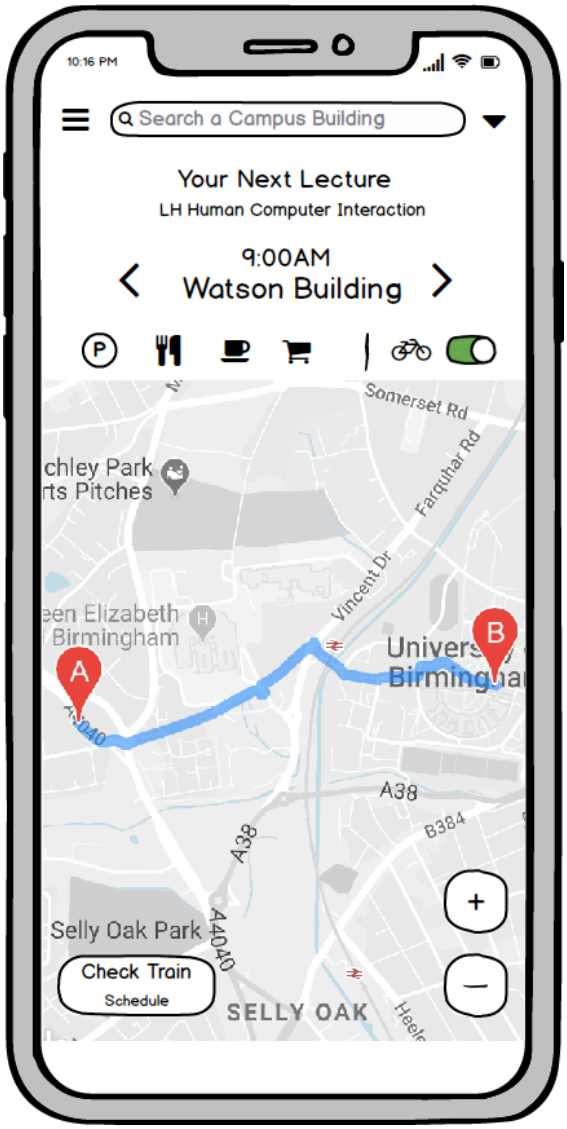
Techniques

In this project, most of the functions will be based on JavaScript, and will have a strong connection with front-end development. The main function will be an automatic route planner from student's geolocation to their lecture buildings. Google Maps API will be a preferred selection for the route planning function, due to the improved accuracy of the planned route and increased usability of API compared to others. I will manually locate every lecture

buildings in my building list to correct the inaccurate search results of mobile maps. From the early stage of researching, I found that Google Maps API does not directly provide the route planning from the user's geolocation to the user's destination. However, it provides the function of route planning from two points. Therefore, I will collect the geolocation of the user and combine the route planning API to achieve this goal. The timetable connection service will require students to upload their timetables, as it is rarely possible to get an API from school's database to reach user's timetable. A preliminary idea is to allow users to save their timetable webpage on the MY.BHAM and upload it to this system. It is also possible to collect the train schedule data from National Rail Darwin API.

Prototype

Here are some of prototypes that I designed for this project.



Summary & Plan

This is a multi-function combined system, and as such I will organize all the user requirements and evaluate the time needed first. Then, I will demonstrate the prototype to users and collect suggestions from the user. Meanwhile, I will finish my literature review in this semester. I will complete the main development portion before the end of Christmas holidays. I will finish the main function first and consider an additional function at a later point, because the additional functions are uncertain and may increase or decrease. I hope that when the system is completed, students and visitors can easily find campus buildings and facilities. I propose that the system will make students' academic lives more convenient through easier interaction.