Software Engineering Group Project

COMP2002 / G52GRP: Interim Report

Project information:

Project title: Novel ways of using NCC Open Data Resources

Project sponsor: Peer-Olaf Siebers

Academic supervisor: Peer-Olaf Siebers

Team information:

Team number: 16

(Alice, Thornton, 20287635, psyat9)

(David, Laming, 20357978, psydl4)

(Kuang, Li, 20377034, psykl7)

(Maria, Constantin, 20370023, psymc8)

(Rahul, Ravi, 20389281, psyrr5)

(Roan, Hutcheson, 20361598, psyrh11)

(Saamiya, Aqeel, 20378737, psysa11)

Documentation (links):

Trello board: https://trello.com/b/cGpDwP3m/interim-report

Code repository: https://projects.cs.nott.ac.uk/comp2002/2022-2023/team16_project/-/tree/main/src

Document repository: https://projects.cs.nott.ac.uk/comp2002/2022-2023/team16_project/-/tree/main/Documentation/Project%20Documentation

Meeting minutes: https://drive.google.com/drive/folders/1wr94Ty5T_BqGOCOZvpBDu4MakcmnA-T-

CONTENTS

PROJECT BACKGROUND

MOTIVATIONS FOR THE PROJECT

We have been made aware the platform "NCC Open Data" is largely unused, with most frequent tags being "transport" or "services", we have established that an application could be made to provide the public of Nottingham with fruitful advice and guidance about the city from the information derived from the datasets. By the means of creating an interactive educational app namely called "Adventurer's Guide to Nottingham" we have utilised the data to create something innovative and unique. Our team expanded upon the requirements to design an inclusive app that appeals to all, including all ages, those new to the city, and those familiar. We offer the user the option to traverse through the app by choosing a "mode", such as kids or normal, with each mode being heavily discerned from the other.

One way the NCC Open Data platform was incorporated was using the latitudes and longitudes of locations we believe are of interest to the public to populate an interactive map. We strive to provide a practical app capable of navigating Nottingham whilst promoting new locations, as well as integrating educational tools such as games and quizzes to facilitate the sharing of knowledge of this city.

HOW DO WE COMPARE TO OTHER PRODUCTS IN THE MARKET?

With the nature of the app being so flexible, it has allowed us to make almost the entirety of it our additions. By offering two versions of the app, we had made it to appeal to all the archetypes of Nottingham. The usage of NCC Open Data plays a large role in supporting the data displayed by our app. While some data may already be available in our competitors' applications, we plan on using data that is currently underutilised in the current market. For example, the car park occupancy data is used to display the current capacity of various car parks around Nottingham. This niche data is hard to find in other applications as discovered.

Another key aim is to present the data in a fresh and engaging way that tells the user more about Nottingham in depth. Each point of interest (leisure centres, landmarks etc.) would contain more detail about them upon click, removing the need for external searches outside the app for information.

Additionally, with one of our main goals showcasing Nottingham as a popular destination and encouraging people to visit, we decided to implement games and quizzes to provide a fun way to share knowledge about the city. This sets us apart from other travel apps whose main purpose is to provide navigation and booking.

REQUIREMENTS AND CRITICAL ANALYSIS

UML DIAGRAMS

Through the use of UML diagrams, we have outlined the design of our app with the final version in mind.

As seen in the activity diagram, users can choose between a normal mode and a kid's mode, in both modes the user can explore Nottingham using a map displaying key locations, including landmarks, Museums & Galleries, leisure centres, markets, and more! An activity diagram for the general use of the map is shown in **Figure 1**.

Normal mode enables the user to navigate between a map, an eating out page, and a quiz menu which includes a selection of quizzes relating to Nottingham. An activity diagram for the general flow of a quiz is shown in **Figure 2**. On the other hand, kids' mode allows the user to navigate between a map and a games menu which includes educational games based around Nottingham. A simple activity diagram for the GUI's navigation is shown in **Figure 3**.

In order to fully grasp and validate the user requirements, several sequence diagrams have been designed to help envision the app's users in various runtime scenarios. The **figures 4**, **5**, **6** distinctly demonstrate the inter-system events, process concurrency, and the message exchange between the system and the user. In each case, the system consists of three lifelines: the GUI, server, and database. These subsystems form the backbone of the application, which further reinforces our dedication in implementing our system design principles, which are maintainability and extensibility **(req 1.7 – spec 1.7).**

The use case in **Figure 7** is a diagrammatic view on the specifications we have formed based on the requirements. The diagram displays the interaction between the different users and the system. The "new to Nottingham" and "local to Nottingham" users inherit tasks from the general user but have different motives. For example, someone new may use the quiz **(spec 2.3)** option to learn more about the city whilst someone who is native to the city may use it to test existing knowledge. The use case is a combination of the KISS, alpha, beta versions of the app (will be spoken in depth later), in short it is what the final version of the app will look like.

Additionally, throughout the whole project we have maintained a backlog of user stories **Figure 8** to which we have added story points by labelling relevant ones . This method allows us to categorise which off the bat would take more time to implement based on difficulty. This to our benefit, developed more confidence in designing the prototype later on.

Finally, we have designed a low-fidelity prototype for the app interface **(Figure 9)** which corresponds to our KISS requirements. This lo-fi prototype enabled us to develop our design and iterate through different group suggestions in a short space of time which prompted innovation and improvement of our initial idea.

OUR REQUIREMENTS

When creating the app we had a timeline in mind, firstly we must implement the KISS (keep it super simple) version of the app, following with the alpha and beta versions. The KISS requirements are vital in order to start the app. By arranging the barebones of the app by the end of the semester, we are able to carry on with the creative features on a solid base for a well-rounded experience for the user. Our main priority was to use data from the NCC platform as effectively as possible **(req 1.3),** this was also further stressed upon by our sponsor.

Additionally, the application we plan on making is a hybrid of a smart travel app and interactive gaming app, so we find it absolutely crucial to implement an information-dense map that includes popups in our stages of developing the program. By the means of API and using java libraries, we plan on offering the users an information-packed map. Inside this we include the popular landmarks and many other sub-categories such as places of worship and markets. This will be done by extracting and refining the data from the NCC Open Data platform **(req 1.4 and 1.5)**.

After successfully implementing our KISS version of the app, our next goal is to proceed with the beta version. The beta version grazes upon all aspects of the previous version but extends them to provide a whole new playground. For example, this version adds transport schedules **(req 2.2)** and games specific to mode clicked **(req 2.4),** which heightens the app experience.

Finally, for implementing our alpha version, we plan on polishing the app and adding more challenging interesting features, such as dynamically updating the data on the app as made available on the platform using a pipeline **(req 3.1).**

WHAT CHANGED?

During development, we realised we were not able to add the feature of saving the user mode **(req 1.6)**. We chose to postpone this feature for alpha release and prioritise the features which gives the KISS version more functionality in order to enforce the proof of our concept. The scope reduction has been necessary as currently there are not enough features to make the distinction between the normal and kid's mode. Instead, we added a beta feature of allowing the user to navigate through the map via a legend **(req 2.6)**. Furthermore, by the end of the semester, we realised we have not executed **req 1.4** to its full potential. Even though the map is present for navigation for the user to look through, we had implied that the user (via userStory, Actor "New to Nottingham") should be able to click on a location and get a route to this location. This was supposed to be implemented via Directions API, but due to time constraints it had not come to be. Nevertheless, the requirements missed will be our top priority when we come to our beta phase of the project.

INITIAL SOFTWARE IMPLEMENTATION AND TESTING

OUR AIM

Our main objective remains to be using data from the NCC Open Data site to prove that publicly available data sources could be used in an advantageous manner. As later demonstrated, our app broadcasts the city of Nottingham in a knowledgeable manner appealing to a wide audience by providing navigation information easily, and niche information sub-categorized for both modes. We plan in the future to be expanding this app to update in real-time via a pipeline to the NCC Open Data site, ensuring our viewers are receiving up-to-date information.

HOW DO WE USE VERSION CONTROL TO OUR BENEFIT?

Designing our branching model strategy for Git was the next step towards the start of development. After conducting research, two models of interest were identified, these being GitLab flow and GitHub flow. **Figure 10** presents a combination of these branching models, which has been tailored specifically for our project. This model suggests a "main" branch, a separate "develop" branch with supporting "feature" branches, and "maintenance".

"Main" is reserved for code that is ready for release, while "develop" is used for storing the work in progress. The feature branches are being used by each sub-team in order to support concurrent, parallel development. As "maintenance" suggests, this branch keeps track of fixes and changes to the code that is ready for release.

Our merging approach consists of assigning other team members to review and approve the changes to be made before pushing. We have utilised GitLab to support the project development since it allows file transfer back and forth and allows us to manage version history.

The upcoming development plan is ready on git in the form of milestones. We plan on implementing the main features of the application that include transport information, landmarks, quizzes, and games.

BACK-END IMPLEMENTATION

For the KISS version of the application, we thought it was best to implement the backend as a server utilising a REST API. This would allow the front end to retrieve the NCC Open Data without having to worry about the implementation of the database or to develop the app with a serverless architecture which no members of the team had experience with. Additionally, we decided to implement a REST API as opposed to another API such as GraphQL since it is simpler to get functioning for the high-fidelity prototype.

To create the server, we have used Java as well as the Spring Boot framework - this decision was reached due to all members having a great deal of Java experience and Spring Boot being one of the most popular frameworks for building REST APIs in Java.

Furthermore, we chose to utilise the University's MYSQL servers to host our relational database since all team members had experience using these servers and SQL. On top of that, this approach is a quick and inexpensive way to prototype the app. Currently, the data stored in our database is from a collection of many of the different datasets available in the NCC Open Data, such as tram stops, landmarks, and leisure centres.

FRONT-END IMPLEMENTATION

For the front-end implementation of our app, it was preferable to use web languages such as HTML, CSS, and JavaScript since our front-end group has plentiful experience. Our team also decided to utilise the Bootstrap CSS framework as it provides responsive, mobile friendly web development. We have also modified our used Bootstrap templates (i.e., navigation bar and buttons) with our own CSS stylesheets in order to personalise the design to fit our preferences.

The map for the normal and kids' mode was implemented using the Google Maps JavaScript API as it provides a way to display interactive customisable maps which we believe will be more engaging to the user. This API also allows us to maintain more control over the mapping experience. For instance, this allowed us to centre the map on Nottingham with a custom zoom value and add our own pins using the NCC Open Data with unique icons for different locations corresponding to the SQL tables in the database, i.e., Landmarks, Museums/Galleries, etc. Additionally, we set up a legend for the map to help users understand what different icons represent.

HOW HAVE WE INTEGRATED THE FRONT AND BACK END?

For every MySQL database table containing NCC data, there is a folder in the back end which contains a set of files responsible for converting SQL data into a JSON file. The four files which execute this process are a base class, a repository, a service layer, and a controller. The base class contains the variables of the corresponding data source, a constructor to initialise those variables as well as a method to prepare these variables to be output in the format of a JSON file. Secondly, the repository interface is initialised to operate as a temporary storage in order to hold data before it's mapped into a JSON file. Subsequently, there is a service layer which uses the repository to hold all the data extracted from a particular MySQL table. Finally, the controller, responsible for output operations, utilises the repository and service layer alongside the base class to map the MySQL table data into the form of a JSON file

The front end communicates with the back-end server through the imported JavaScript jQuery library which loads the JSON-encoded data from the server using a GET HTTP request, e.g., when a GET request is sent to /API/landmark, a list of JSON landmarks is returned with each object having the same elements specified by the NCC Open Data files. Consequently, we were able to populate our maps with pins using the latitude and longitude elements from the database tables. Furthermore, we utilised the name element in order to display the location's name when a pin is selected on the map.

To improve maintainability **(req 1.7)**, we have utilised Javadocs for the backend Java code. This includes describing database entities, as well as the API endpoints and server-side logic where required. It also includes details to build, run, and test the server for developers. Additionally, adding coding conventions to the back end **(Figure 11)** allowed our code to have a uniform style and made it easier to understand code written by other team members.

The project instructions of how the application should be run are written in detail in the README.md file located in the git repository.

IMPLEMENTATION TESTING

For the moment, we have only been conducting defect testing for the back end in order to ensure the software produces the correct results. It is currently confirmed that all tests are passing with the expected output. After conducting unit testing, we plan on implementing integration testing to guarantee the functionality of the front and back end as a group. We will follow with unit and integration testing after every version to verify the software's performance. In the future, after we have implemented the high-fidelity prototype, we will also conduct validation testing to ensure that the software fully meets the initial requirements with our sponsor.

Furthermore, at the end of development, we will be conducting release testing - a full system test to meet the full specifications and check that the software is fit for external use. During this period, having a few candidates try our application might be beneficial as the feedback would help us find areas requiring improvement and allow us to identify flaws in our design.

Lastly, having test plans established also serves as evidence to our sponsor and the client of proper Software Engineering. The overall test plan document includes the tested items, specifications of the inputs to the test, and what the expected output from the system is. These test plans are available on git, where all developers can access it.

PROJECT MANAGEMENT

TOOLS USED

Our team utilises a wide variety of project management tools in order to ensure we are following the timeline outlined for the project, and to guarantee all team members are on the same page in terms of the work to be done. However, our main way of communicating tasks is through Trello, which is managed by our team leader**.**

This is updated throughout the week as a task is completed. Alongside that, we have a Trello for user stories and the project roadmap which includes brainstorming points for the timeline of the project. These are later organised into a Gantt chart format for a strict period of time allocated to each task **(Figure 12)**. Furthermore, these are sorted into git milestones and issues by our git lead.

For daily check-ups, regular stand-up meetings, and sprints, we as a group use a discord server that is categorised into channels which are sorted by relevance. All group resources are uploaded there prior to approval by sponsor, after which they are uploaded to the team's google drive.

HOW DOES OUR TEAM WORK TOGETHER?

During the conclusion of the KISS version development, our team split into the front and back end sub-groups, formed by aligning members with their strengths. After tasks were delegated to sub-groups, we introduced a flexible approach to our meetings in which each sub-group had the freedom of scheduling their own informal meetings, as well as setting their own objectives. But as follows with any team, there is a wide range of skills and expertise. Some members were not as well versed in the designated coding language(s) as others, so precautionary methods were taken before carrying out any tasks. Measures such as in-depth discussions and rough diagrams helped in ensuring a smooth process and mitigated the risk of an unexpected incident occurring during programming.

An example of where we utilised skill transfer lies in the front-end development where when developing the screens for the project the more experienced members started earlier, this action allowed the experienced members to enforce a standard of code and synchrony between the screens. It also allowed them to teach the others more about the development of the code. The back-end development, however, contained a variety of tasks to be carried out so doing the head start approach was not efficient time wise. Nevertheless, collaboration was done in a different way, via pairs. Each pair carried out their assigned tasks, such as retrieving coordinates from the database using a Spring Boot framework or adding to the database. After each sub-group achieved their goals, the entire team was reunited with frequent sprint meetings. All meetings are recorded by our group admin in the form of meeting minutes.

Our team has quickly adapted to the SCRUM sprints throughout the timeline. Our first sprint consisted of requirements and specification elicitation alongside designing UML diagrams for our app. We thought that it was important to have this as our first sprint to ensure that we were on the same page with the project supervisor about our design.

Our second sprint consisted of creating an implementation of our app that met the KISS requirements. For the back end, this required constructing a REST API that could retrieve the data for the application and send it to the front end. During this sprint, we also jumped ahead on connecting the backend to the database as it would allow us to test our code as we were writing it. We also added the NCC Open Data tables we planned to use during this sprint as well. Some of the data sources needed cleaning before uploading to the database.

Our third sprint was mainly focused on connecting the front end and the back end of the app which had been developed separately. This involved the writing of jQuery code for the front end to send GET requests to the REST API to retrieve the relevant JSON data to create map pins.

All team members maintained a professional attitude when approaching the project. Communication for the project is done through a discord server as well as consistent group meetings to help plan the next steps. Interactions with the sponsor are also planned prior in order to decide what to discuss, to ensure the meeting is as productive as possible. All members show commitment to the project by being punctual to group meetings and meeting deadlines for tasks set within the group. This shows we are an autonomous group that can plan and carry out tasks independently. Discipline is also important and is maintained by planning out the problem well and creating small sub-tasks to prevent multi-tasking and work overloading.

Progress made so far

According to our Gantt chart, by the end of the semester, we are in line with what it portrays. We did stumble across a few hiccups (discussed in the next section) and not all our KISS requirements were executed to the level we had originally anticipated. However, our team expected some ideas to be axed due to time constraints. Nevertheless, by planning to use what we have learned thus far we hope to fast-track our project for the next phase of our plan.

PRELIMINARY REFLECTION

Challenges and lessons learnt

No project comes without its individual challenges and due to the volume of work alongside other courseworks, it consequently led to our development taking longer than anticipated, according to the Gantt chart made, especially towards the end as we fumbled with front and back-end connection due to a communication error. However, this situation was easily remedied by listing down tasks to be completed in priority order, completing them with high-level collaboration. This made an immediate impact as the progress continued to be on track.

A concern identified in the development process was the lack of communication/understanding between the front and back-end groups. Moving forwards, we plan to put in place more meetings between the sub-groups and utilise skill transfer so there is more understanding from both ends of how the other works in order to combine future implementations more effectively.

CONCLUSION

RETROSPECTIVE

By the end of this semester, we've successfully designed a high-fidelity prototype integrating HTML, CSS, and JavaScript as our front-end languages, with Java as our back-end language. As a group we were able to utilise the NCC Open Data datasets latitude and longitude elements to produce map markers.

The data, however, needed adapting as it contained multiple columns of redundant information such as "body name" and "create_dat", which had to be removed entirely before proceeding with the conversion to MySQL. Furthermore, some of the files are available in a xml format rather than csv, and some files only contain northing and easting while others have longitude and latitude. Neither is a negative, but rather acts as a setback when there is lack of consistency between the files.

FUTURE PLAN

As a team, we have thoroughly enjoyed undertaking this task. It has allowed us to explore the potential of data and how it can be used as a powerful tool to change people's lives. Nevertheless, with this project we always strive to do more, and when it comes to our upcoming beta release, we plan on implementing the remainder of the KISS requirements which are not up to standard before moving on to transforming the app into a hub of entertainment and functionality. Equally important is the planned alpha release, which is where we plan on adding improvements throughout the app to provide an enjoyable user experience.

APPENDIX

REQUIREMENTS

Req No	KISS VERSION 1	ALPHA VERSION 2	BETA VERSION 3
1	1.1) Different versions of the app - Kid's mode and Normal mode	2.1) Landmarks are issued with information for users to read	3.1) Pipeline made to update data as made available
2	1.2) Ability to choose on screen which version to proceed with	2.2) Transport information is easily available for users to access	3.2) Filters to see certain types of landmarks or places to eat.
3	1.3) Optimum use of data available on the hub	2.3) Discerning the kids' mode and normal mode. Kids mode being more colourful and simpler while the normal mode being more information dense	3.3) Use an external data source to develop a "places to eat" option
4	1.4) Map for the users to follow for navigation	2.4) Quizzes available to play with major landmarks Note: simpler quiz for the kids' mode and a more complex version for the normal mode	3.4) Educational games available to play for the kids' version and normal mode, kids' version being more interactive and normal mode being more educational - Both being easy to play
5	1.5) Setup database with all the data we would use e.g., landmarks	2.5) Emoticon icons available for different purposes of the app e.g., heart for favouriting, lightbulb for random facts or pop ups	3.5) Ability to share on social media to increase awareness of the app
6	1.6) Data is preserved, mode preferences etc	2.6) Create a legend for the map in order to quickly identify the types of landmarks available to visit.	3.6) Include opening hours for landmarks which require tickets to enter
7	1.7) Easy to maintain and extend		3.7) Develop an adaptable interface for different devices
8			3.8) Recommendations available for users based on previous data collected

FUNCTIONAL REQUIREMENTS 🔵

NON-FUNCTIONAL REQUIREMENTS 🟢

SPECIFICATIONS

SPEC NO.	KISS VERSION 1	ALPHA VERSION 2	BETA VERSION 3
1	1.1) The system should be able to provide the option of different modes available to users(kids and normal). **Req 1.1**	2.1) The system should display some information about the landmark when users click on it. **Req 2.1**	3.1) Dynamic updating of data used by the app **Req 3.1**
2	1.2) The system should optimally use the data available on the hub **Req 1.3**	2.2) The system will display bus and tram timings and stops according to users' will. **Req 2.2**	3.2) The system provides a filter option for the map; users can filter according to their preference. **Req 3.2**
3	1.3) The system should display a map. **Req 1.4**	2.3) The system provides the option of playing a quiz **Req 2.4**	3.3) The system provides option of playing a game(in both modes) **Req 3.4**
4	1.4) Users should be able to select pins on the map **Req 1.4**	2.4) The system will provide the score after quiz is played **Req 2.4**	3.4) The system provides the score of the game after being played by the user. **Req 3.4**
5	1.5) Users should be able to zoom in and zoom out on the map **Req 1.4**	2.5) The system shows which question is answered correctly/ incorrectly. **Req 2.4**	3.5) The system displays a list of places to eat. **Req 3.3**
6	1.6) Users should be able to navigate between options. (Home page, Map, Fun Facts, Games) **Req 1.2**	2.6) The system presents emoticon icons in order for users to "rate" landmarks. **Req 2.5**	3.6) System offers user an option to share landmark/app on social media **Req 3.5**
7	1.7) System should be easily maintained and extendable by programmers **Req 1.7**	2.7) The system displays a legend for users on the map page for easy identification. **Req 2.6**	3.7) System displays opening hours and timing for landmarks that require tickets **Req 3.6**
8	1.8)System preserves the data of the user, the mode pressed or frequently chosen **Req 1.6**		3.8) The system is able to adapt onto different devices **Req 3.7**
9			3.9) The system provides the user recommendations based on previous data. (e.g., pins selected on map) **Req 3.8**

UML Diagrams

Activity Diagrams

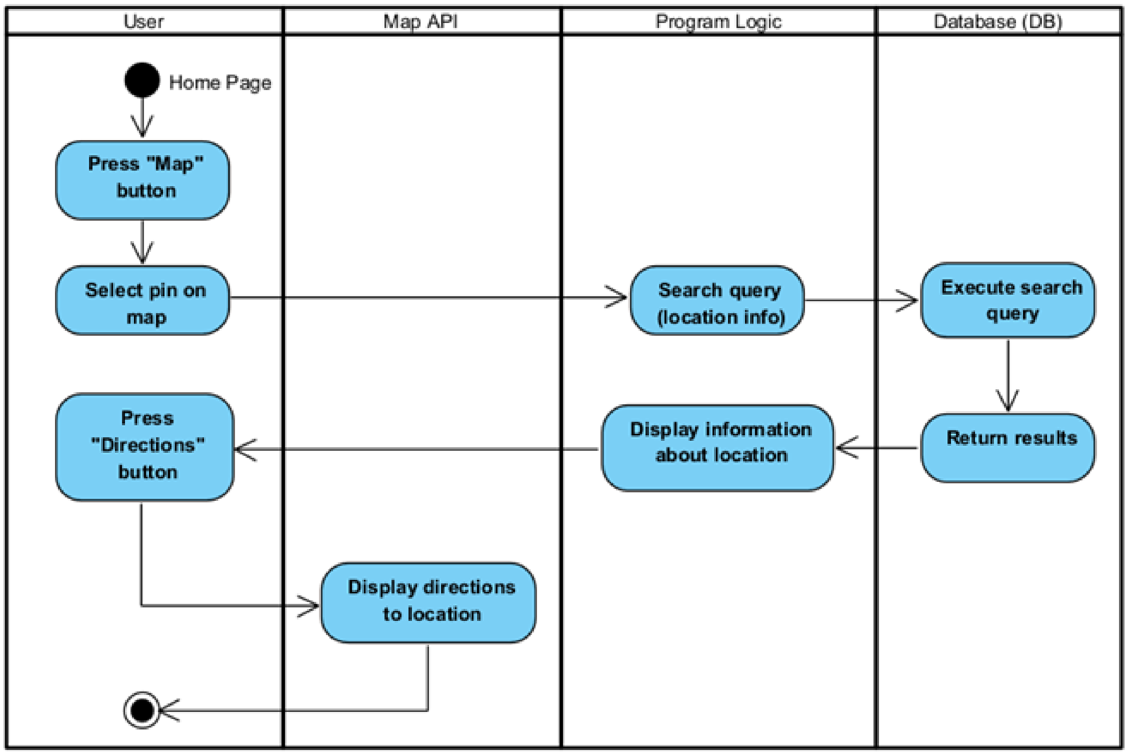

User
Map API
Program Logic
Database (DB)
Home Page
Press "Map" button
Select pin on map
Search query (location info)
Execute search query
Return results
Display information about location
Press "Directions" button
Display directions to location

Figure (1)

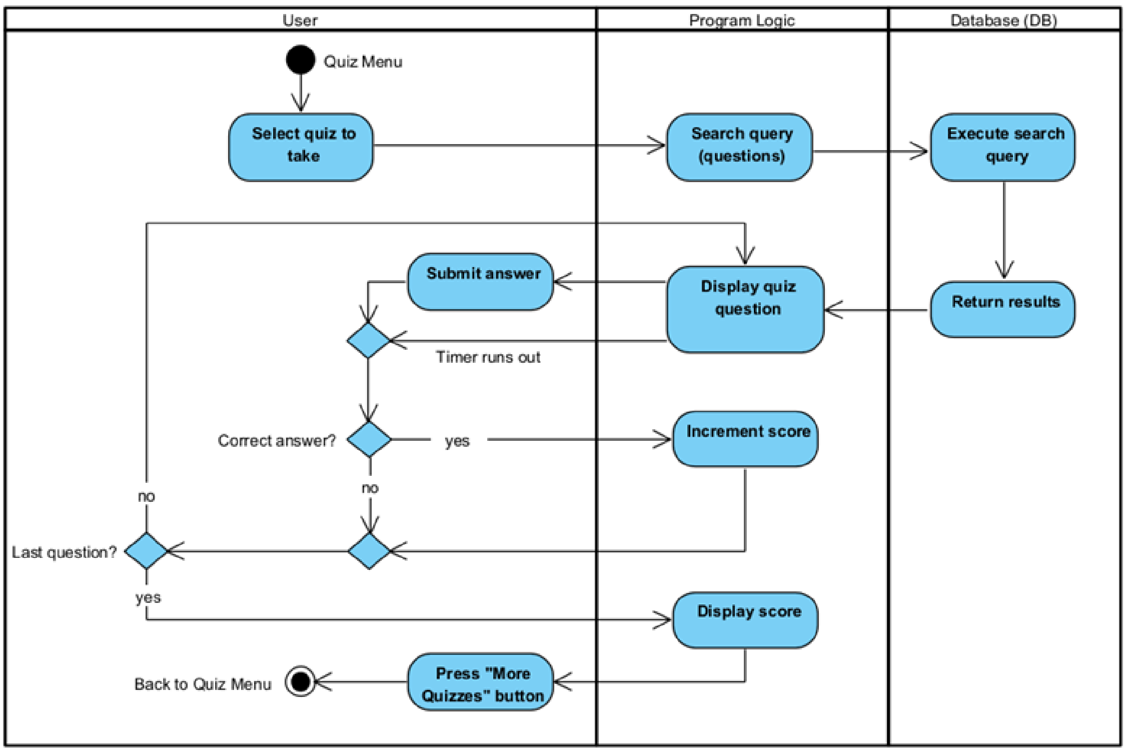


Figure (2)

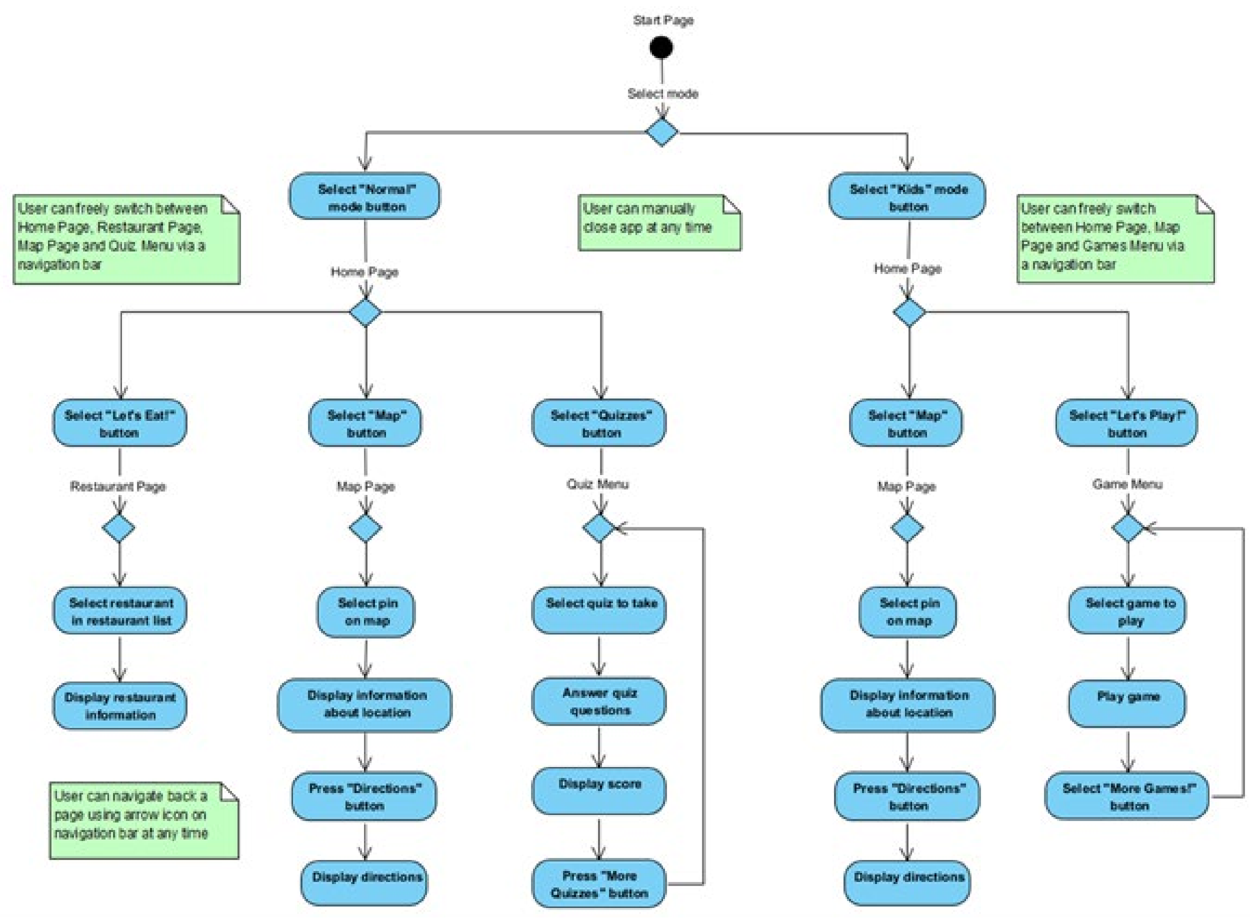

Start Page
Select mode
Select "Normal" mode button
Select "Kids" mode button
User can freely switch between Home Page, Restaurant Page, Map Page and Quiz Menu via a navigation bar
User can manually close app at any time
User can freely switch between Home Page, Map Page and Games Menu via a navigation bar
Home Page
Home Page
Select "Let's Eat!" button
Select "Map" button
Select "Quizzes" button
Select "Map" button
Select "Let's Play!" button
Restaurant Page
Map Page
Quiz Menu
Map Page
Game Menu
Select restaurant in restaurant list
Select pin on map
Select quiz to take
Select pin on map
Select game to play
Display restaurant information
Display information about location
Answer quiz questions
Display information about location
Play game
User can navigate back a page using arrow icon on navigation bar at any time
Press "Directions" button
Display score
Press "Directions" button
Select "More Games!" button
Display directions
Press "More Quizzes" button
Display directions

Figure (3)

SEQUENCE DIAGRAMS

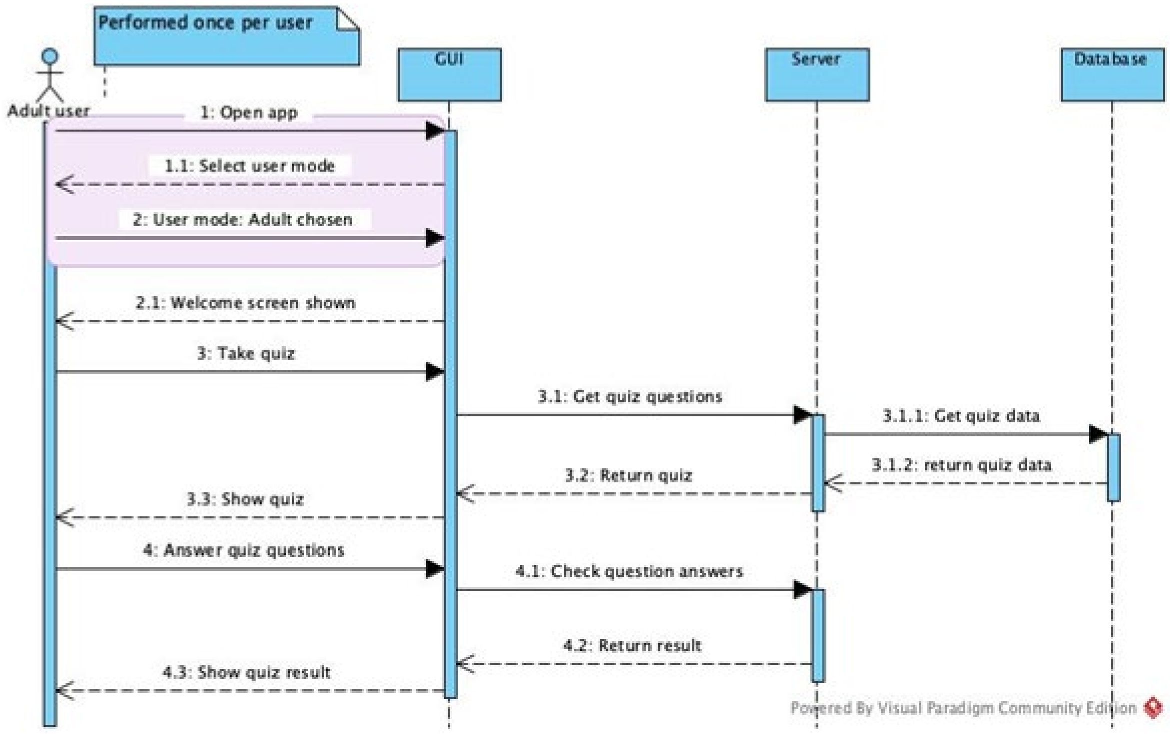


Figure (4) – Adult taking a quiz

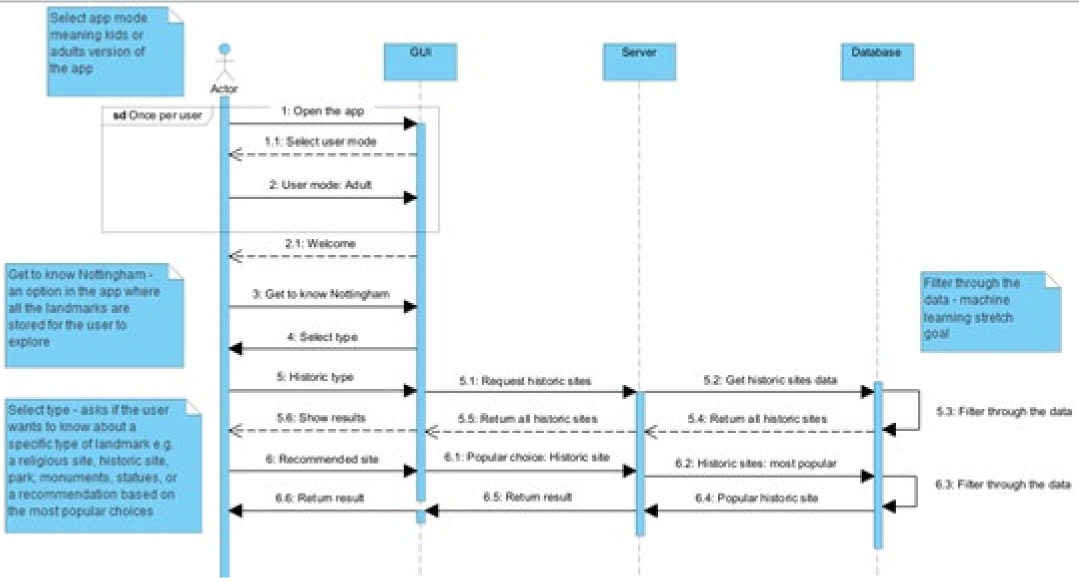

Select app mode meaning kids or adults version of the app
Actor
GUI
Server
Database
sd Once per user
1: Open the app
1.1: Select user mode
2: User mode: Adult
2.1: Welcome
Get to know Nottingham - an option in the app where all the landmarks are stored for the user to explore
3: Get to know Nottingham
4: Select type
Filter through the data - machine learning stretch goal
5: Historic type
5.1: Request historic sites
5.2: Get historic sites data
5.3: Filter through the data
Select type - asks if the user wants to know about a specific type of landmark e.g. a religious site, historic site, park, monuments, statues, or a recommendation based on the most popular choices
5.6: Show results
5.5: Return all historic sites
5.4: Return all historic sites
6: Recommended site
6.1: Popular choice: Historic site
6.2: Historic sites: most popular
6.3: Filter through the data
6.6: Return result
6.5: Return result
6.4: Popular historic site

Figure (5) – Adult using the filter option

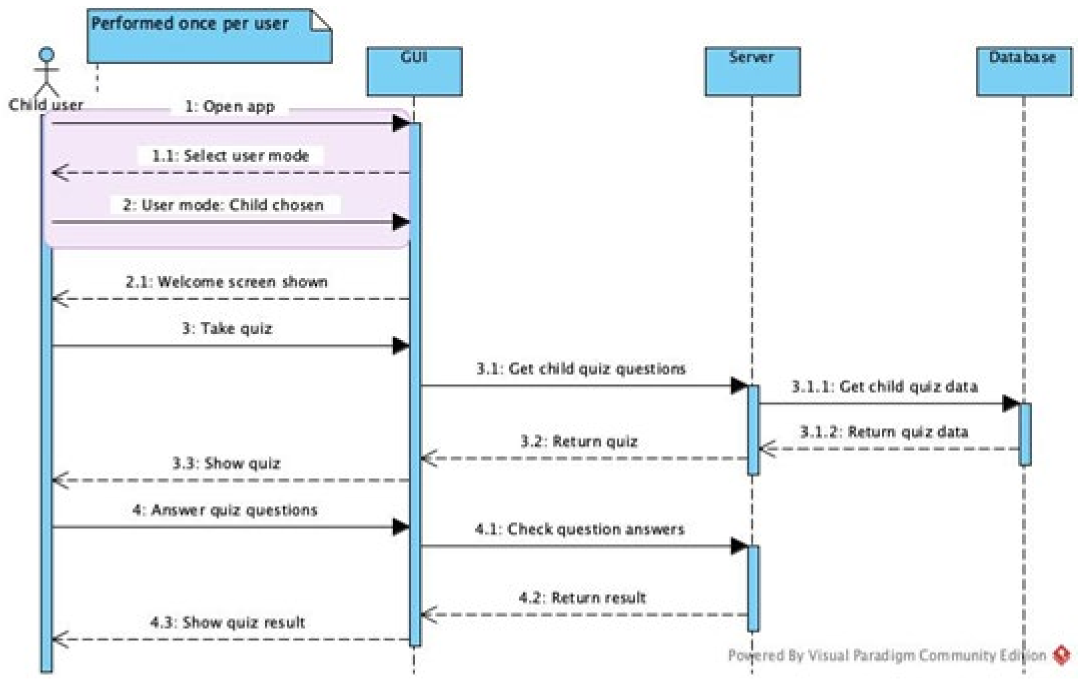


Figure (6) – Child taking a quiz

USE CASE DIAGRAM

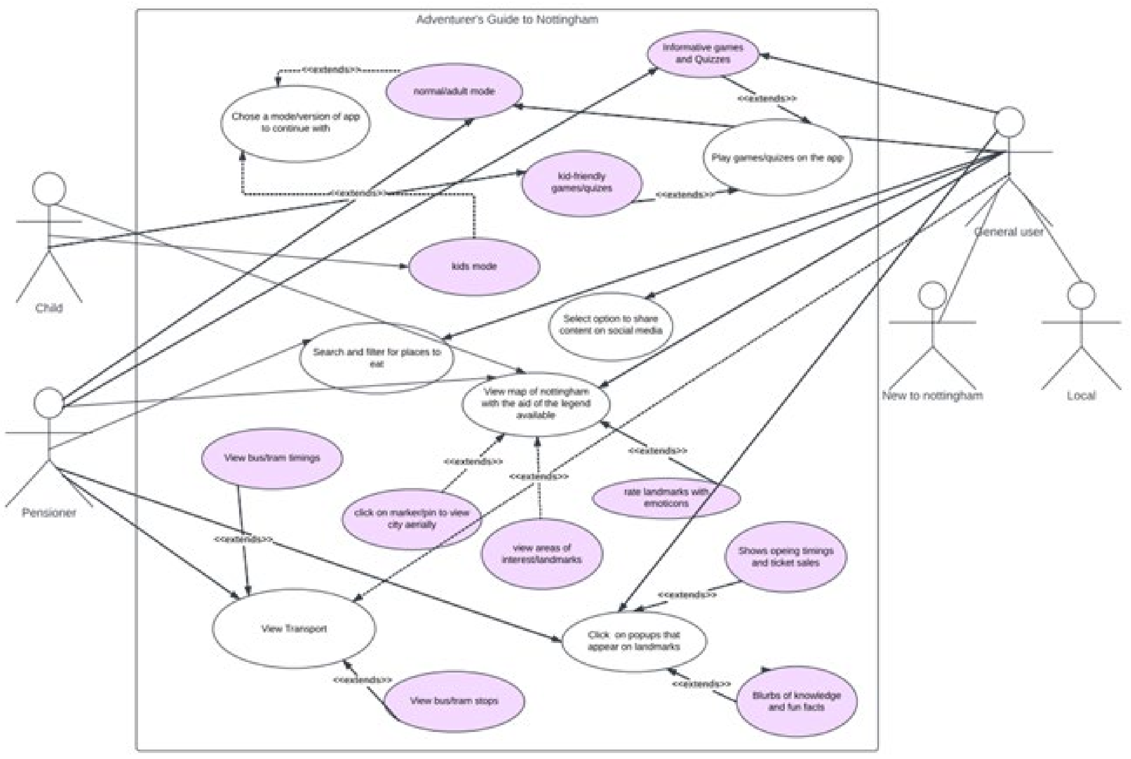
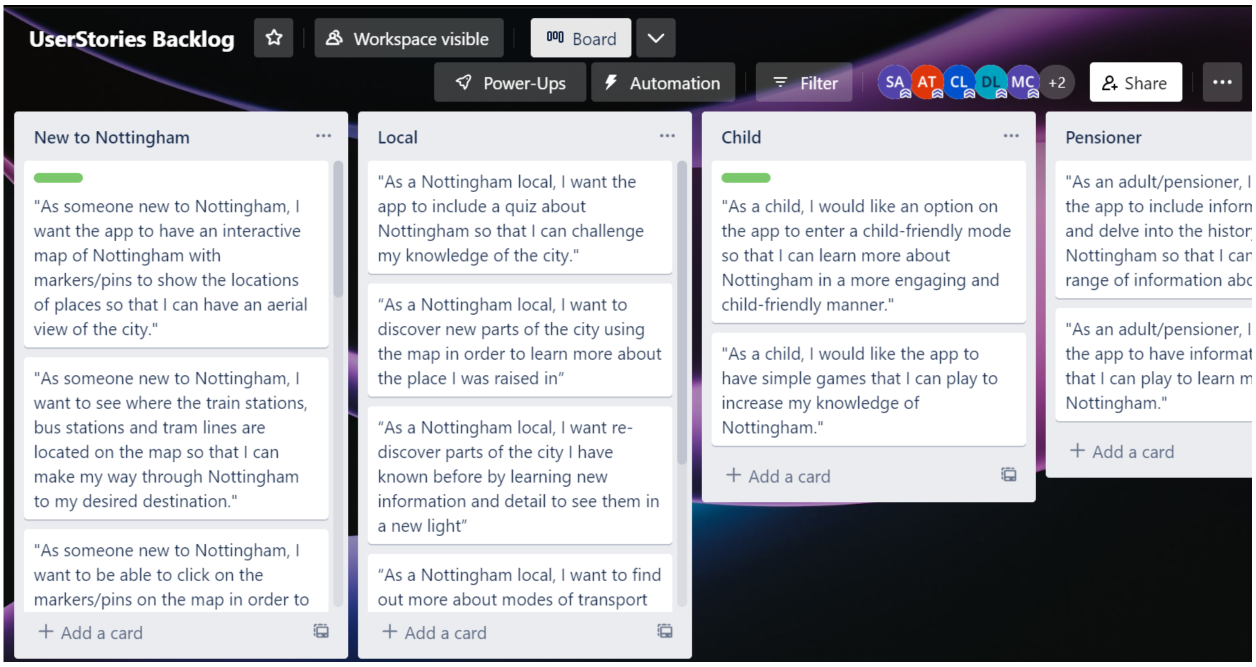

Adventurer's Guide to Nottingham
Informative games and Quizzes
<<extends>>
normal/adult mode
Chose a mode/version of app to continue with
Play games/quizes on the app
kid-friendly games/quizzes
kids mode
Select option to share content on social media
Search and filter for places to eat
View map of nottingham with the aid of the legend available
View bus/tram timings
click on marker/pin to view city aerially
view areas of interest/landmarks
rate landmarks with emoticons
Shows opeing timings and ticket sales
View Transport
Click on popups that appear on landmarks
View bus/tram stops
Blurbs of knowledge and fun facts
Child
Pensioner
General user
New to nottingham
Local

Figure (7)

User Stories


UserStories Backlog
Workspace visible
Board
Power-Ups
Automation
Filter
SA AT CL DL MC +2
Share

New to Nottingham

"As someone new to Nottingham, I want the app to have an interactive map of Nottingham with markers/pins to show the locations of places so that I can have an aerial view of the city."

"As someone new to Nottingham, I want to see where the train stations, bus stations and tram lines are located on the map so that I can make my way through Nottingham to my desired destination."

"As someone new to Nottingham, I want to be able to click on the markers/pins on the map in order to

+ Add a card

Local

"As a Nottingham local, I want the app to include a quiz about Nottingham so that I can challenge my knowledge of the city."

"As a Nottingham local, I want to discover new parts of the city using the map in order to learn more about the place I was raised in"

"As a Nottingham local, I want re-discover parts of the city I have known before by learning new information and detail to see them in a new light"

"As a Nottingham local, I want to find out more about modes of transport

+ Add a card

Child

"As a child, I would like an option on the app to enter a child-friendly mode so that I can learn more about Nottingham in a more engaging and child-friendly manner."

"As a child, I would like the app to have simple games that I can play to increase my knowledge of Nottingham."

+ Add a card

Pensioner

"As an adult/pensioner, I the app to include inforn and delve into the histor Nottingham so that I car range of information abc

"As an adult/pensioner, I the app to have informat that I can play to learn m Nottingham."

+ Add a card

Figure(8)

Low-Fidelity Prototype

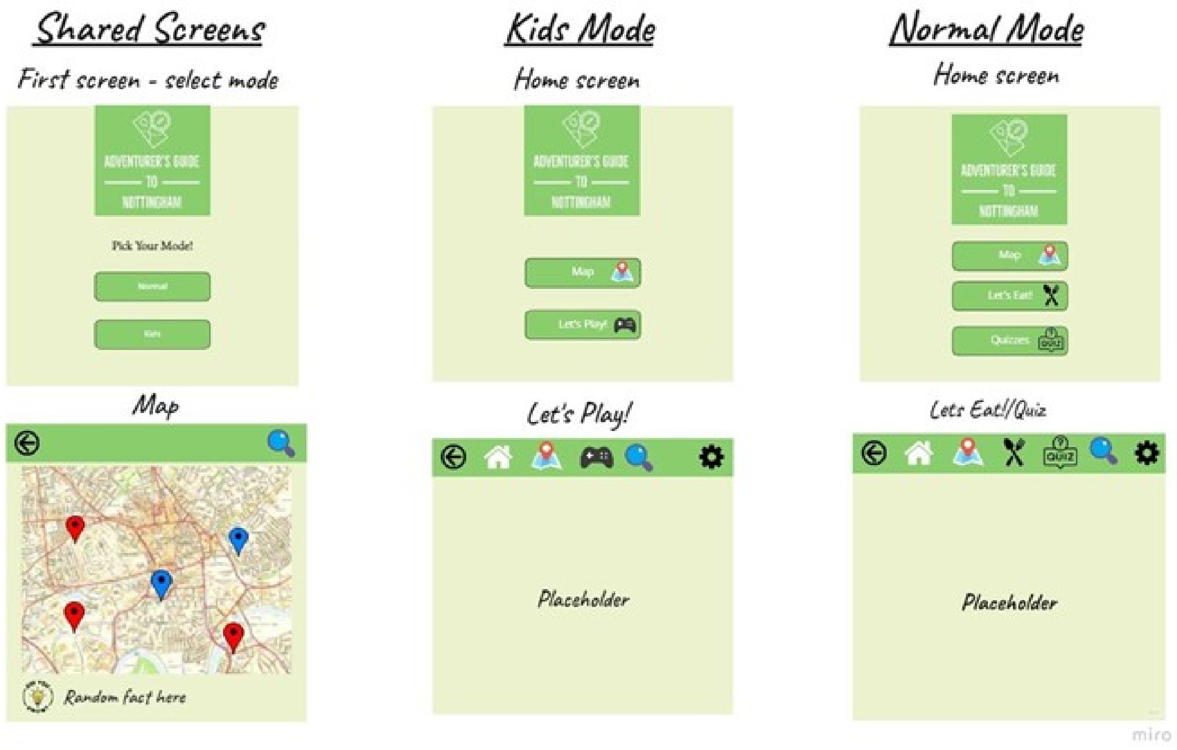


Figure (9)

Branching strategy

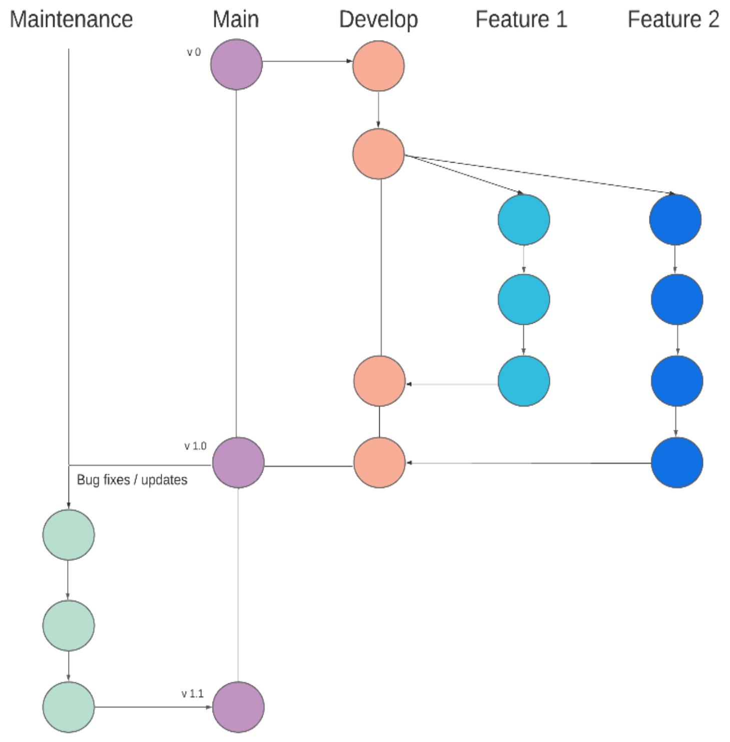
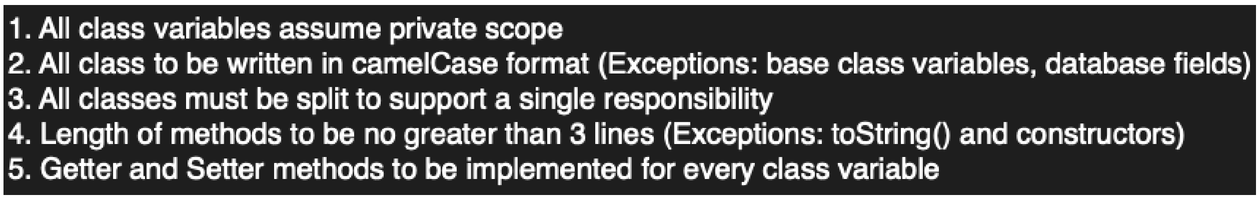


Figure (10)

Coding Conventions



1. All class variables assume private scope
2. All class to be written in camelCase format (Exceptions: base class variables, database fields)
3. All classes must be split to support a single responsibility
4. Length of methods to be no greater than 3 lines (Exceptions: toString() and constructors)
5. Getter and Setter methods to be implemented for every class variable

Figure (11)

GANTT CHART

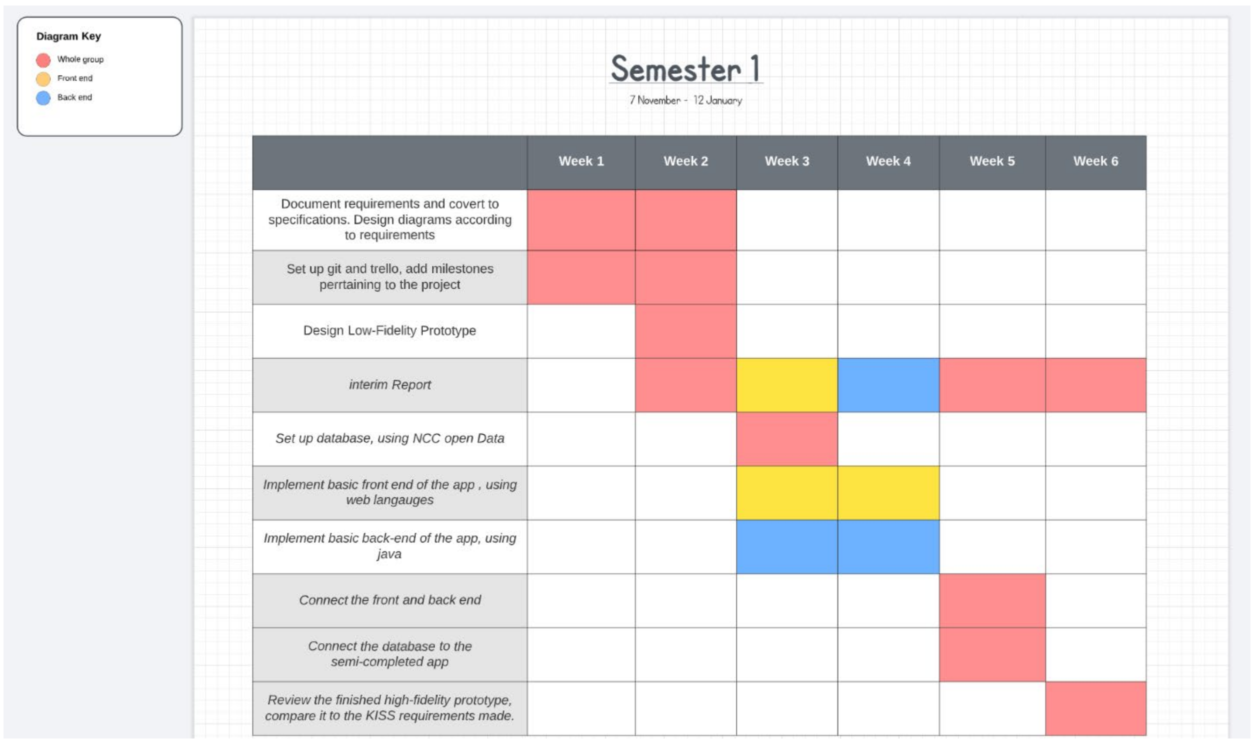


Diagram Key

- 🟥 Whole group
- 🟨 Front end
- 🟦 Back end

Semester 1

7 November - 12 January

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6
Document requirements and covert to specifications. Design diagrams according to requirements	🟥	🟥				
Set up git and trello, add milestones perrtaining to the project	🟥	🟥				
Design Low-Fidelity Prototype		🟥				
interim Report		🟥	🟨	🟦	🟥	🟥
Set up database, using NCC open Data			🟥			
Implement basic front end of the app , using web langauges			🟨	🟨		
Implement basic back-end of the app, using java			🟦	🟦		
Connect the front and back end					🟥	
Connect the database to the semi-completed app					🟥	
Review the finished high-fidelity prototype, compare it to the KISS requirements made.						🟥

Figure (12)