



Interactive Map of SETU Carlow Design Document

14th February 2025

Supervisor: Dr Oisín Cawley
Student: Gábor Major – C00271548

Table of Contents

Introduction.....	3
System Architecture.....	3
Technology Stack.....	4
Godot.....	4
Godot Geolocation Library.....	4
Godot Firebase Library.....	4
Google Firebase.....	4
Structure of stored data.....	4
Algorithms.....	6
A* Pathfinding.....	6
Search Algorithm.....	7
Class Diagram.....	9
Design Plan.....	10
Iteration 0 & 1.....	10
Iteration 2.....	10
Iteration 3.....	11
System Sequence Diagrams.....	12
Navigate To Room.....	12
Use Case.....	12
View Campus Map.....	12
Search Room.....	13
View Room Information.....	13
Modify Room Information.....	14
Modify Map Waypoint.....	14
UI Design.....	15
Map Overview Screen.....	15
Search Screen.....	15
Information Viewing Screen.....	15
Navigation Screen.....	15
Admin Room Editing Screen.....	16
Admin Map Editing Screen.....	16
References.....	17

Table of Figures

Figure 1: System Architecture Diagram.....	3
Figure 2: Entity Relation Diagram.....	5
Figure 3: Class Diagram.....	9
Figure 4: Gantt Chart for Iteration 0 & 1.....	10
Figure 5: Gantt Chart for Iteration 2.....	10
Figure 6: Gantt Chart for Iteration 3.....	11
Figure 7: Navigate To Room Sequence Diagram.....	12
Figure 8: View Campus Map Sequence Diagram.....	12
Figure 9: Search Room Sequence Diagram.....	13
Figure 10: View Room Information Sequence Diagram.....	13
Figure 11: Modify Room Information Sequence Diagram.....	14

Figure 12: Modify Map Waypoint Sequence Diagram.....14

Figure 13: Map Overview Screen.....15

Figure 14: Search Screen.....15

Figure 15: Information View Screen.....15

Figure 16: Navigation Screen.....15

Figure 17: Admin Room Editing Screen.....16

Figure 18: Admin Map Editing Screen.....16

Introduction

The purpose of this document is to outline a design for the Interactive Map of SETU (South East Technological University) Carlow application. It includes the technologies and tools required to create this application and also includes diagrams to show the structure.

A high level overview of the whole system is given, followed by the technologies and libraries used to create the application. A description of the storage structure of data and algorithms used throughout the application are laid out as well.

A class diagram and system sequence diagrams are given according to the specification of the application.

Finally a section of the UI (User Interface) designs are shown as a series of screens that are open on the device during the usage of the application.

System Architecture

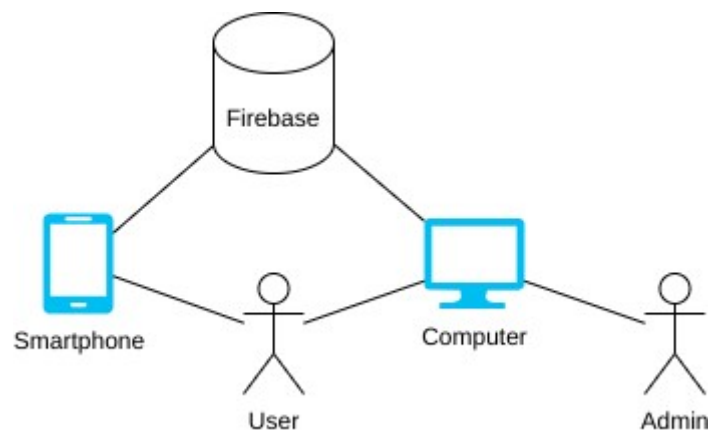


Figure 1: System Architecture Diagram

Technology Stack

Godot

The application is written using the Godot game engine. All logic is written so that it can be contained in the application for both the Android operating system and for desktops on the web.

The Android version of the application has only read only access to the database map data, whereas the desktop version will have a login page for the admin user so it would also be able to modify the data.

Godot Geolocation Library

The Android version of the Godot application will get the location of the phone of the user through the Android application programming interface. To do this a plugin called GodotGPSPlugin will be used to connect the map application to the proper geolocation services. The plugin allows for querying for the phone's latitude, longitude, accuracy, altitude, vertical accuracy, speed, time, and bearing. (PraxisMapper, 2024)

Godot Firebase Library

The Godot application needs a connection to the Firebase database which is facilitated by the GodotFirebase software development kit plugin. It allows for storage of the map data required and for authenticating admin users. (GodotNuts, 2024)

Google Firebase

Google Firebase is used to store all of the map data for the application, including the buildings, rooms and various waypoints around the college campus.

Structure of stored data

The Firebase Firestore is a NoSQL database and as such it is not a traditional relational database with tables, it instead has Collections, which store Documents, which store Data, which can be primitive data types such as numbers or words, or links to other Collections of Documents. (Google, 2024)

The structure of the Documents stored in the Firestore is as follows:

1. The Base Map Document stores the longitude and latitude, a link to the Collection of Waypoints that are directly on the Base Map, a link to the Collection of Buildings on the Base Map, and two values for when the Waypoints and Buildings in the Collections last got updated, stored as a Unix timestamp.
2. The Building Document stores the longitude and latitude of the Building's main entrance, the name, description and letter of the Building, a link to the Collection of Waypoints directly in the Building, a link to the Collection of Rooms in the Building,

and two values for when the Waypoints and Rooms in the Collections last got updated, stored as a Unix timestamp.

3. The Room Document stores the longitude and latitude of the Room in the Building, the floor number the Room is on, the name, description and lecturer using the Room if there are any, a link to the Collection of Waypoints directly in that Room, and a value for when the Waypoints in the Collection last got updated, stored as a Unix timestamp.
4. The Waypoint Document stores the longitude and latitude, the floor number the Waypoint is on, and an associative array which contains the identification numbers of any of the Waypoints it has a connection with as the keys, and what feature that connection is, for example a stair, or an elevator as the values.

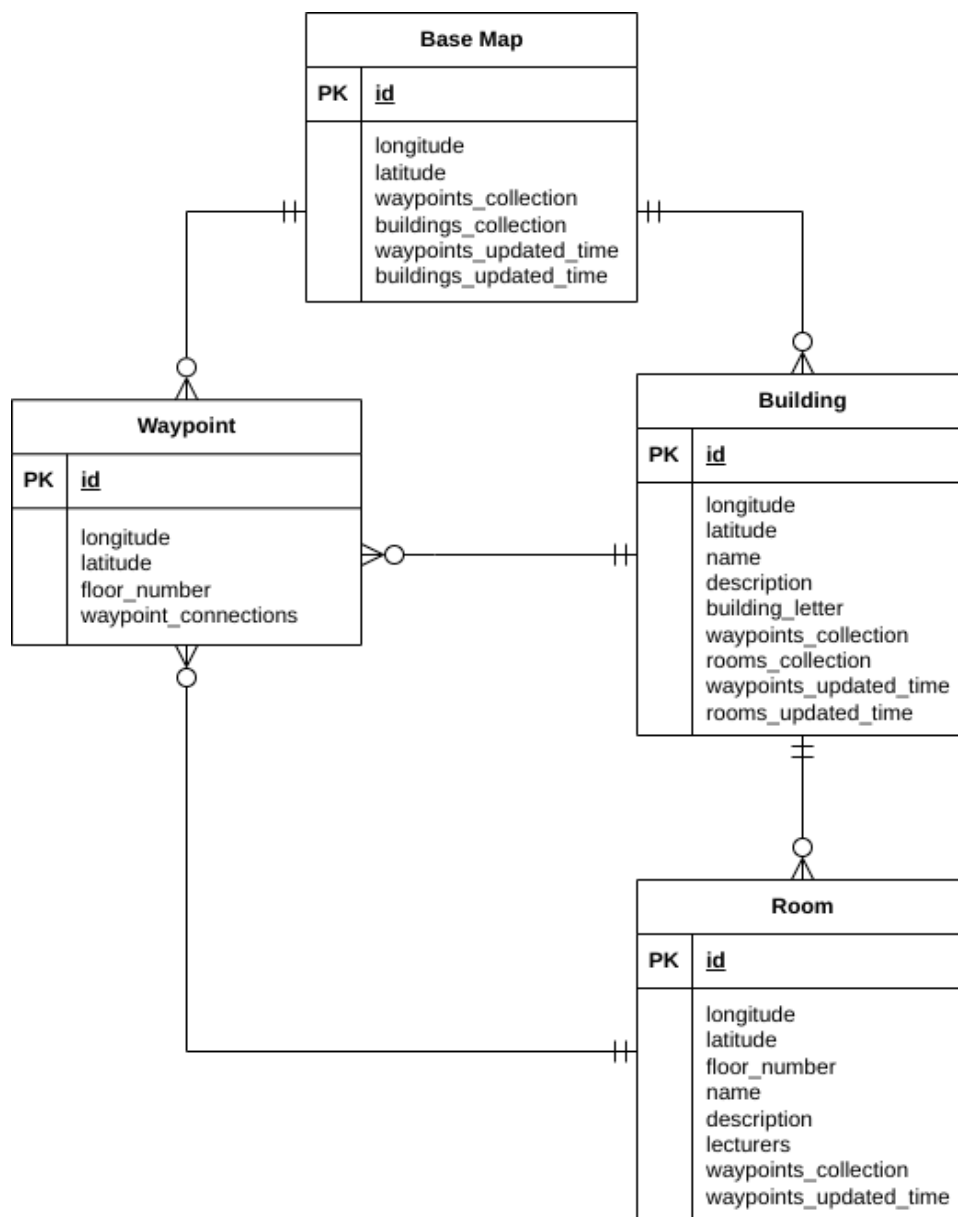


Figure 2: Entity Relation Diagram

Algorithms

A* Pathfinding

The main algorithm that is going to be used extensively throughout the whole application is the A* Pathfinding algorithm. The costs required for the algorithm are explained as follows:

- **F** cost is the total estimated cost of the path through this node. This is $f = g + h$.
- **G** cost is the cost needed to reach this node.
- **H** cost is the estimated cost from the node to the end node. This cost is calculated using the Euclidean Distance.

Short pseudocode of the algorithm is given: (Brilliant.org, 2024), (Sebastian Lague, 2014)

- Create a list of unchecked nodes and add the starting node.
- Create an empty list for checked nodes.
- **While** the unchecked node list is not empty run the following:
 - Set current node to node with lowest **f** cost from the unchecked list.
 - **If** current node is destination node then:
 - Signal that the destination has been reached.
 - **Else:**
 - Remove current node from unchecked list.
 - Add current node to checked list.
 - **For** each neighbour node of current node run the following:
 - **If** (neighbour is not in the checked list) and ((neighbour not in unchecked list) or (current node **g** cost plus cost to neighbour is smaller than neighbour **g** cost)) then:
 - Set the neighbour's **g** cost with newer lower value.
 - Set the neighbour's from node to the current node.
 - **If** neighbour not in unchecked list then:
 - Set neighbour **h** cost.
 - Add neighbour to unchecked list.
- Signal that destination has not been reached.

Search Algorithm

The search algorithm is a secondary algorithm in the map application however it is still an important one as it allows the user to use the application with more ease and it allows them to find Rooms and Buildings that they may not even know the location of.

The search algorithm is split into two different branches, the first branch runs when the search text is greater than 2 characters, and the second is when only a single character is entered. The splitting of the search is so that if only a single character is used only search through the Building letters and not anything else.

For the longer text search exacts matches of Structure names are prioritised, and for both branches of the search algorithm Buildings are prioritised and ranked higher than Rooms.

- Get array of all Buildings.
- Get array of all Rooms.
- **If** searching text longer than 2 characters then:
 - Create name match array.
 - Create lecturers match array.
 - Create description match associative array, with the number of occurrences of the searching text as the keys, and the Structures array as the values.
 - **For** each Room in Room array do:
 - **If** Room name equals searching text then:
 - Put Room to the front of name match array.
 - **Else if** Room name contains searching text then:
 - Append Room to name match array.
 - **Else if** Room description contains searching text then:
 - Append Room to description match associative array, with the searching text count as the key.
 - **Else if** Room lecturers contains searching text then:
 - Append Room to lecturers match array.
 - **Else if** Room parent name equals searching text then:
 - Push Room to the front of name match array.
 - **Else if** Room parent name contains searching text then:
 - Append Room to name match array.
 - **For** each Building in Building array do:

- **If** Building name equals searching text then:
 - Put Building to the front of name match array.
- **Else if** Building name contains searching text then:
 - Append Building to name match array.
- **Else if Building** description contains searching text then:
 - Append Building to description match associative array, with the searching text count as the key.
- Create a result array.
- Append name match array to result array.
- Append lecturers match array to result array.
- Append each array stored in the description match array in order of keys starting from the biggest to the result array.
- **Return** result array.
- **Else if** searching text is 1 character:
 - Create location match array.
 - **For** each Building in Building array:
 - **If** Building letter equals searching text then:
 - Append Building to location match array.
 - **Break**
 - **For** each Room in Room array:
 - **If** Room parent letter equals searching text then:
 - Append Room to location match array.
 - **Return** location match array.
- **Return** empty array

9



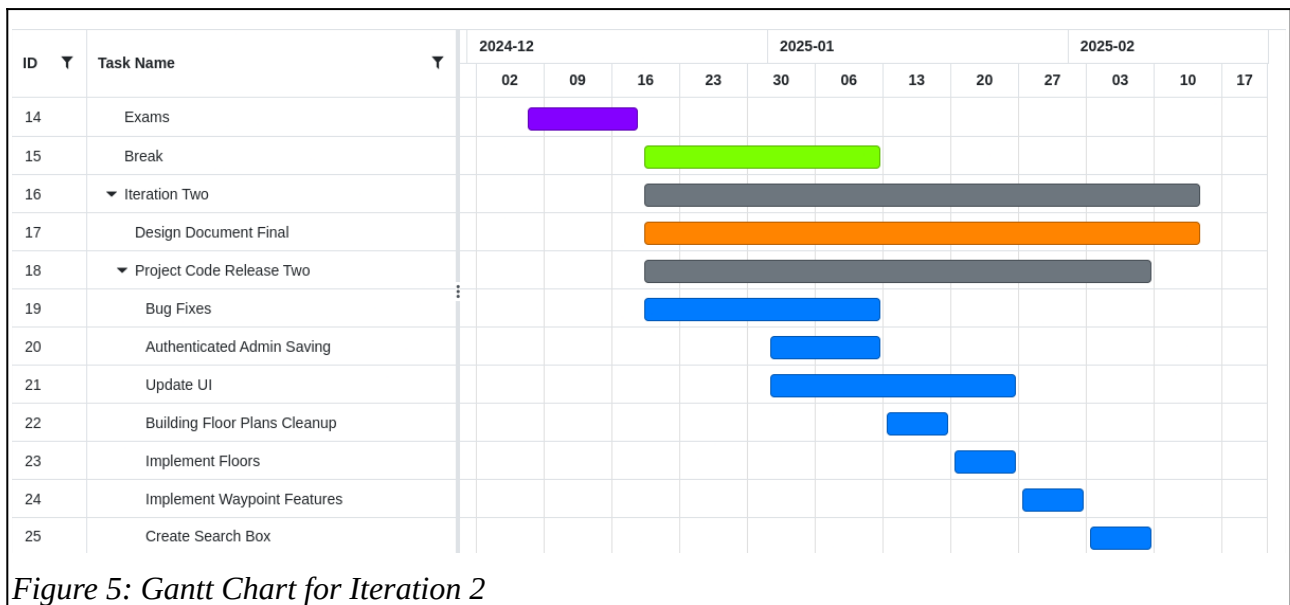
Design Plan

The design and development plan for the project is shown below as Gantt charts (Gantt.com, 2024). Iteration 0 and 1 are together in the first chart followed by the Iteration 2 chart and finally the Iteration 3 chart.

Iteration 0 & 1



Iteration 2



Iteration 3

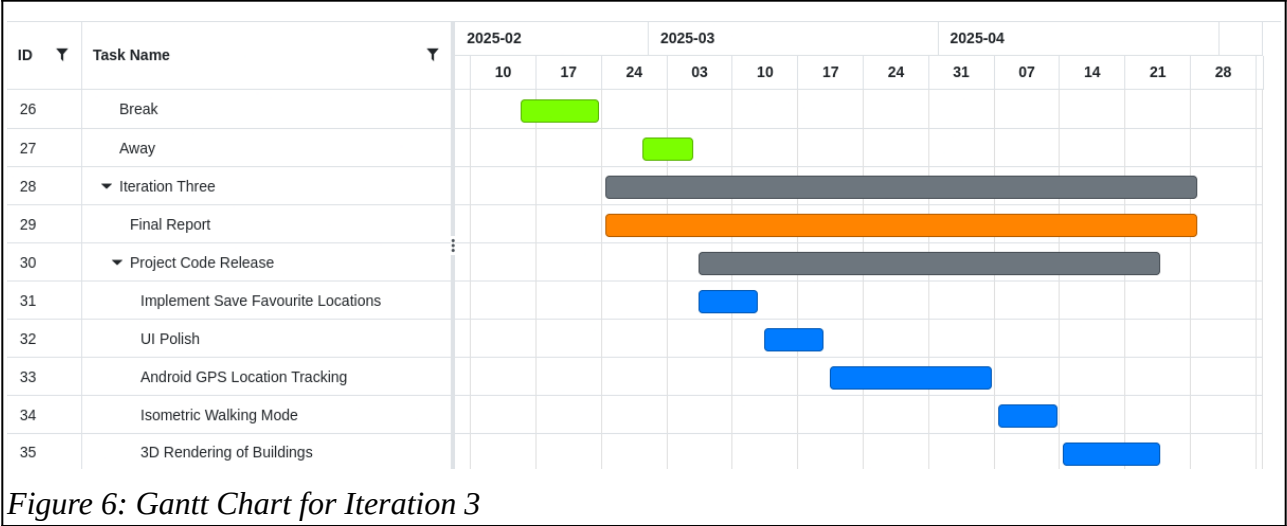


Figure 6: Gantt Chart for Iteration 3

System Sequence Diagrams

Navigate To Room

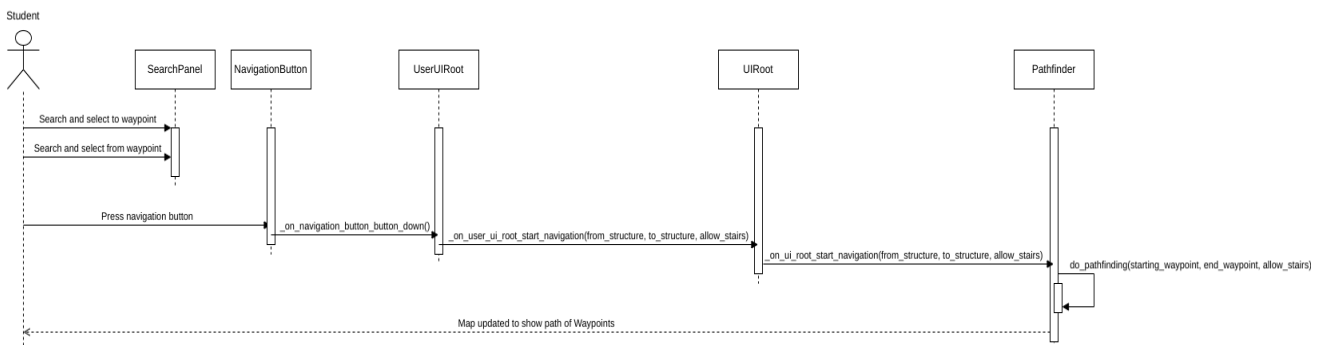


Figure 7: Navigate To Room Sequence Diagram

Use Case

The main use case for this map is the navigation aspect and the above sequence diagrams shows this in action.

1. The student uses the search bar to search and select a place to go to.
2. The student then selects a place to go from.
3. The student presses the navigation button to initiate the process.
4. The message propagates to the Pathfinder.
5. The Pathfinder then uses the A* algorithm to create a valid path between the two locations.

6. The map is updated for the student to see the path.

View Campus Map

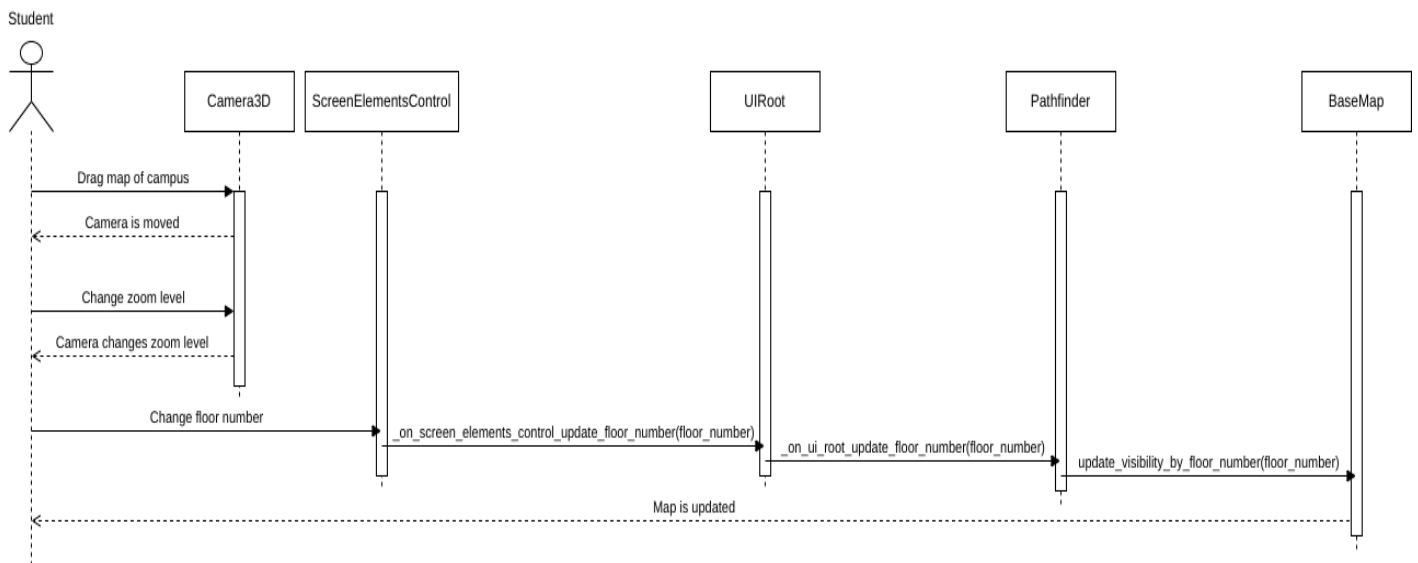


Figure 8: View Campus Map Sequence Diagram

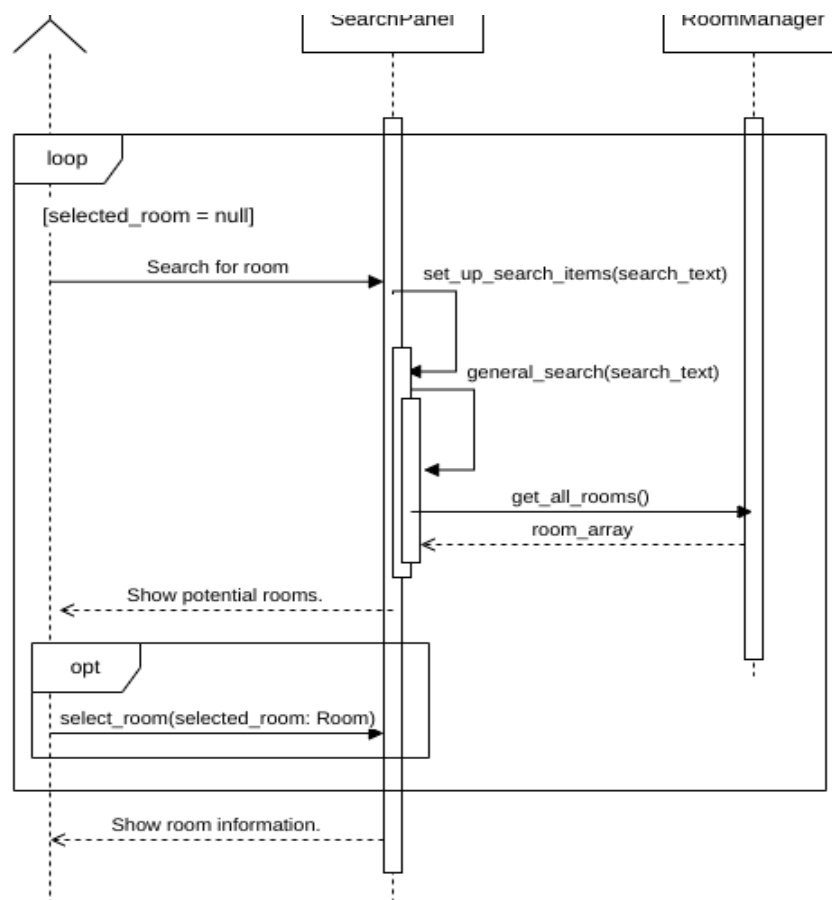


Figure 9: Search Room Sequence Diagram

View Room Information

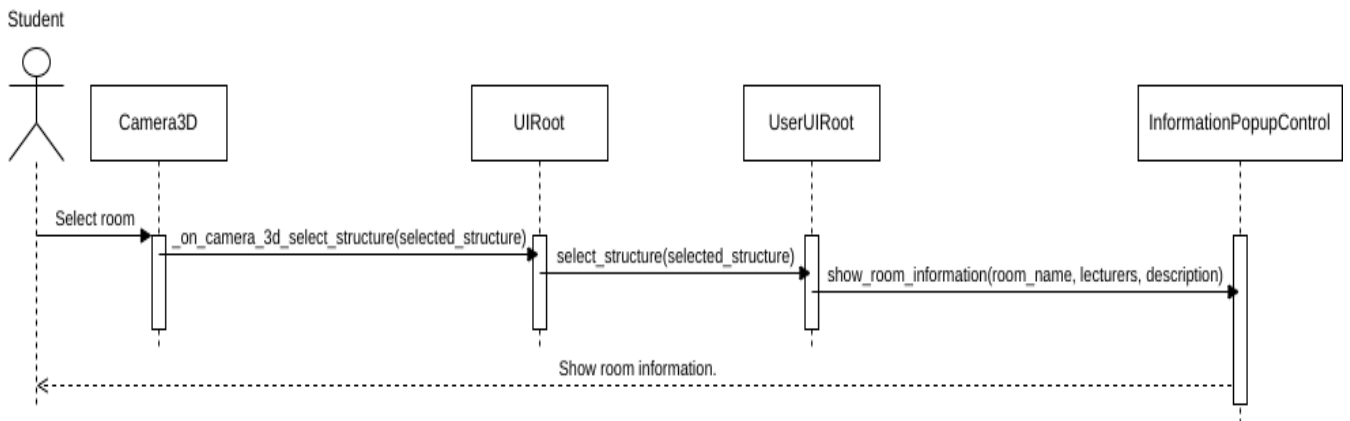


Figure 10: View Room Information Sequence Diagram

Modify Room Information

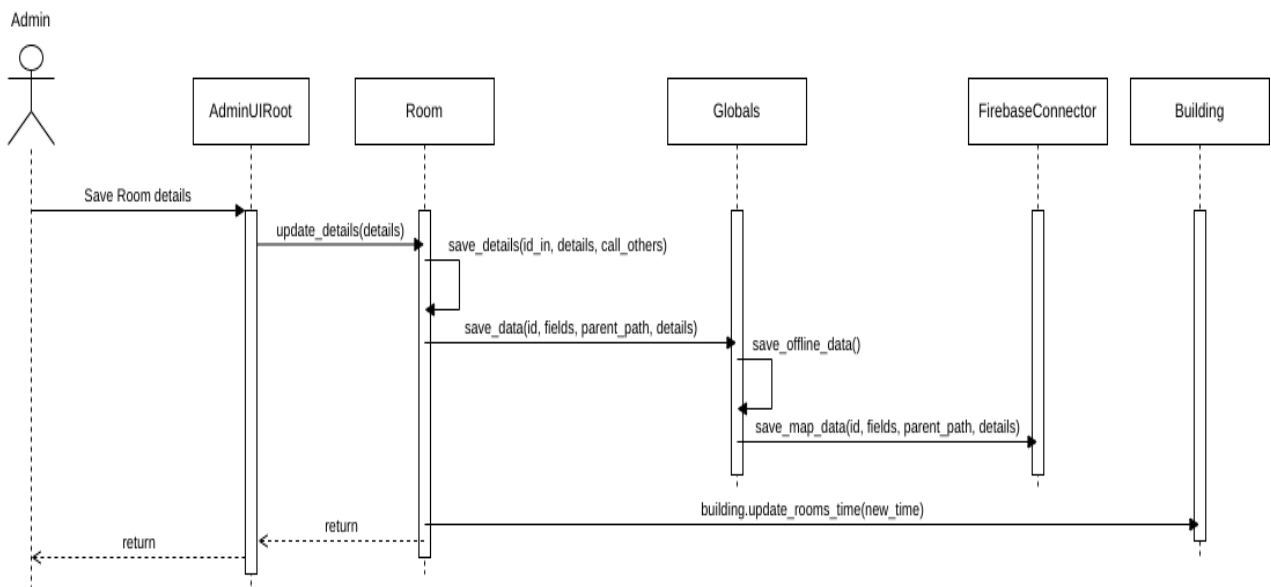


Figure 11: Modify Room Information Sequence Diagram

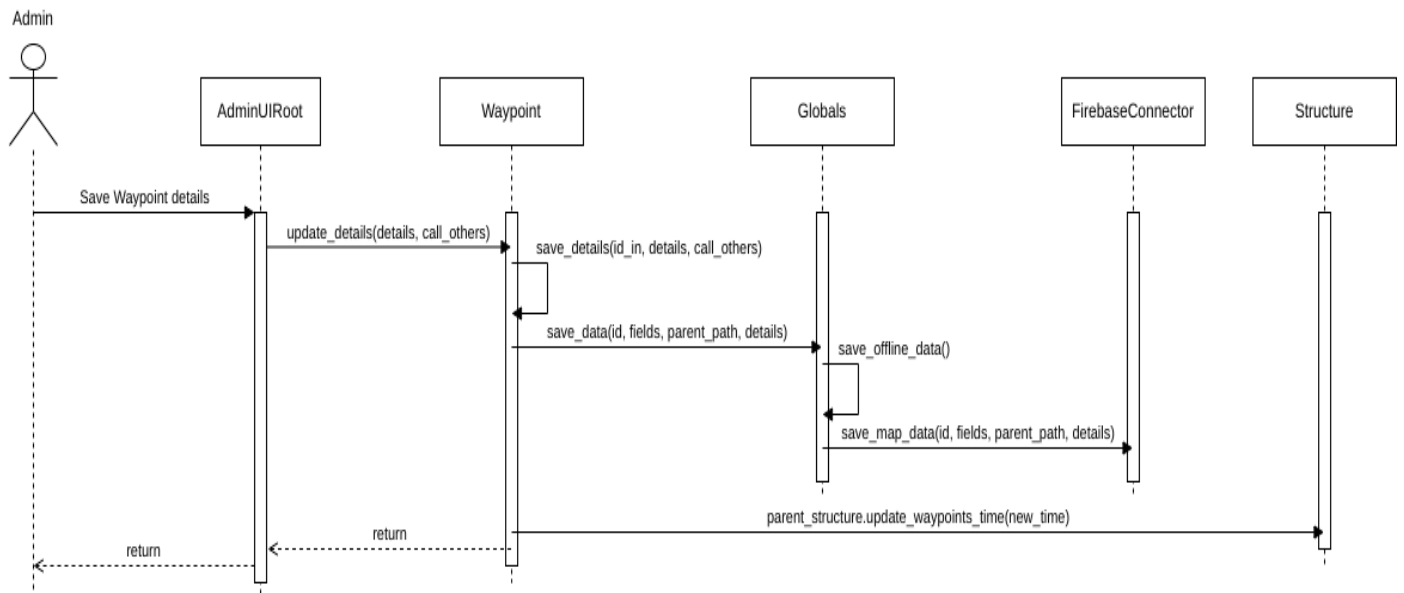


Figure 12: Modify Map Waypoint Sequence Diagram

UI Design

Map Overview Screen



Figure 13: Map Overview Screen

Search Screen

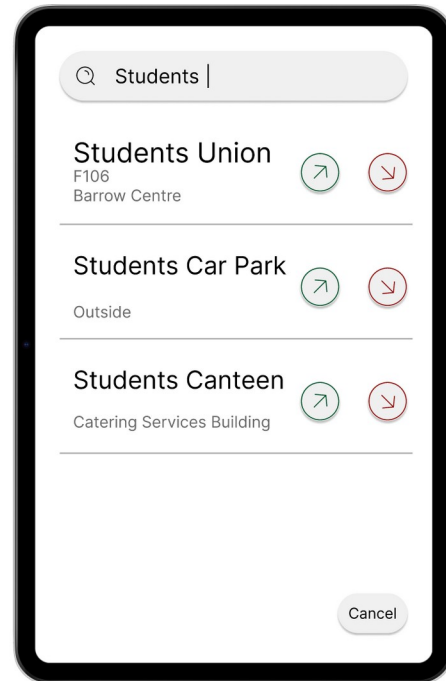


Figure 14: Search Screen

Information Viewing Screen



Figure 15: Information View Screen

Navigation Screen

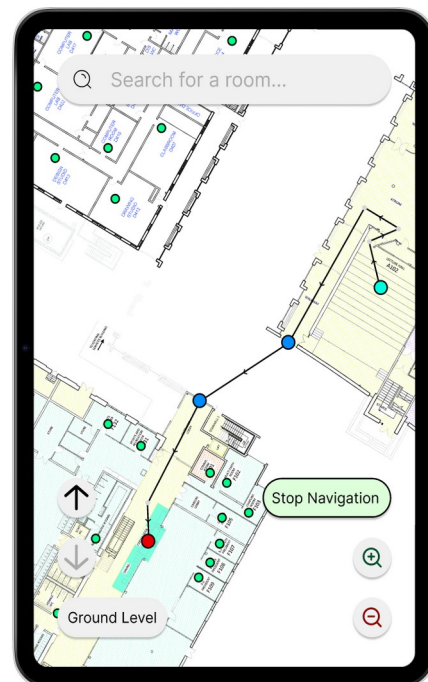


Figure 16: Navigation Screen

Admin Room Editing Screen

Editing Room

Name:
A102 - Lecture Hall

Lecturers:
None

Floor Number:
1

Longitude:
-6.9357201628823

Latitude:
52.8268276272825

Description:
This is a large lecture hall with a presentation deck.
This room is located just outside the entrance of the library.

Save Cancel

Figure 17: Admin Room Editing Screen

Admin Map Editing Screen

Editing Waypoint

Floor Number:
1

Feature:
None

Longitude:
-6.935668834824693

Latitude:
52.826913024733756

Parent:
A102 - Lecture Hall

Connections:
Connection 1

Save Cancel

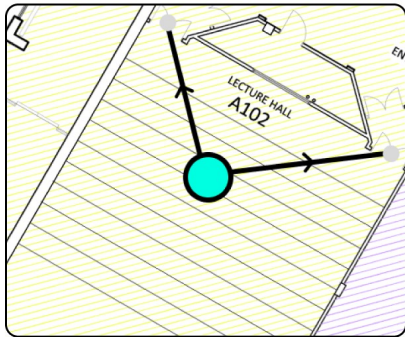


Figure 18: Admin Map Editing Screen

References

PraxisMapper (2024) GodotGPSPlugin. Available at:

<https://github.com/PraxisMapper/GodotGPSPlugin> (Accessed: 19 November 2024)

GodotNuts (2024) GodotFirebase. Available at:

<https://github.com/GodotNuts/GodotFirebase> (Accessed: 19 November 2024)

Google (2024) Cloud Firestore. Available at: <https://firebase.google.com/docs/firestore>

(Accessed: 19 November 2024)

Brilliant.org (2024) A* Search. Available at: <https://brilliant.org/wiki/a-star-search/>

(Accessed: 24 November 2024)

Sebastian Lague (2014) A* Pathfinding (E01: algorithm explanation). Available at:

<https://www.youtube.com/watch?v=-L-WgKMFuhE> (Accessed: 24 November 2024)

Gantt.com (2024) What is a Gantt Chart?. Available at: <https://www.gantt.com/> (Accessed:

6 December 2024)