Smart Navigation System for Emergency Exits

A Project Report
Presented to
The Faculty of the College of
Engineering

San Jose State University
In Partial Fulfillment
Of the Requirements for the Degree
Master of Science in Software Engineering

By

Han-Wei Lin
Swati Ganesh Narkhede
Sheetal Narvekar
Roger Navarro

05/2021

APPROVED

____
Ishie Eswar, Project Advisor

____
Dan Harkey, Director, MS Software Engineering

____
Rod Fatoohi, Department Chair

ABSTRACT

Smart Navigation System for Emergency Exits

By

Han-Wei Lin, Swati Ganesh Narkhede, Sheetal Narvekar, Roger Navarro

Safe evacuation from indoor areas in case of life-threatening emergencies is an essential aspect of the people's safety present at site and for the first responders who oversee assisting the evacuation. As stated by the Department of Occupational Safety and Health Administration, the most frequent causes of evacuations in the U.S. each year are fires and floods. In such emergency cases, the local officials are the first to initiate the emergency alert, followed by the designated onsite trained coordinators who give instructions & assist in evacuation.

Whether to shelter in place or to evacuate to safety is among the most critical decisions to make in an emergency. The local authorities may not be available to assess the situation and provide the right decisions and instructions in time. Planning building evacuation in advance may be difficult as situations change rapidly, making it hard to consider unpredictable factors during evacuation. Accordingly, there is a need for improved communication & execution of an evacuation of a site.

In this project, we propose a smart indoor navigation system to guide a building's evacuation in an emergency. Our approach uses a visual tool-based application to help users to escape safely. The system considers the current safe routes available and navigates the user to the nearest available safe exit avoiding possible blockages and actively responding to new possible

threats. The outcome will be an optimized evacuation plan enhancing the safety of people and security personnel.

Acknowledgments

The authors are extremely grateful to Professor Ishie Eswar for his support and guidance in the preparation of this work. We are fortunate to have an advisor who always finds time for listening to the problems and roadblocks that occur during the course of the project. His technical and editorial advice has given us valuable insights on the working of the project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. Literature Search, State of the Art

Navigation services like Google Maps and Apple maps have made the outdoor navigation effectively simple. Along with outdoor navigation, indoor navigation is also required in case of emergency evacuation, finding a specific shop in a mall, etc. Global Positioning Systems (GPS) which is used in outdoor navigation is not useful for indoor navigation. Considering this problem, we propose a smart indoor navigation system for emergency exits.

The system shall provide an application for seamless audio/visual guide to exit a site during an emergency. Currently, a group of selected individuals (volunteers) working in the building are trained and assigned the responsibility of guiding people out towards the safe site during an emergency. These individuals are responsible to ensure every individual follows their guided path and nobody is left behind. These volunteers can exit the building only after they can confirm everyone has left the building. This responsibility puts these individuals at the risk of facing the disaster for longest period of time compared to others. If there is an automated system that guides people towards the safe exit of the building/premise, then it would drastically reduce the risk for certain individuals who volunteer to supervise the exit process of the building occupants. If everyone has this app, then the administrator (at remote location) can confirm the status of each individual and alert them if there is deviation or failure to follow the directions by any individual.

1.1. Proposed Areas of Study and Academic Contribution

The primary area of study is focused on indoor positioning and navigation. For indoor localization low bandwidth IoT devices will be used with Bluetooth technology to connect the remote devices for location purpose. Indoor navigation will be achieved by training a machine learning model to determine the shortest available path for navigation between two nodes. Outdoor positioning is based on global positioning system and there are different tools available for localization. The area of indoor positioning and navigation is relatively new and not many tools are available. In an emergency the first responders perform the difficult task of evacuation and navigation, and majorly rely on manual updates of the current scenario. The department of National Institute of Standards and Technology (NIST) is undergoing research to equip first responders with modern technical tools and applications for efficient evacuation plan. Through the project the users will be provided an overlay of the navigation path using AR technology for efficient navigation.

1.2. Current State of the Art

1.2.1. Emergency Casualties

Natural disasters and emergencies are common phenomenon that affect hundreds of people each year. These events may prompt evacuations from certain areas. Among the different types of natural disasters floods and storms are more common in the United States. In the state of California fires, floods and earthquakes are the most common emergencies [1]. California has the greatest number of emergencies compared to other states.

Santa Clara county in California is the sixth most populous county in the state. The county’s Public Health Department is responsible for registering all births and deaths in the county [2]. Santa Clara office of emergency Management has an entire network assigned to assist in an emergency. The department aids in any natural or other emergencies pertaining to a group or an individual. The range of emergencies managed vary from hospital visits due to medical emergency or any other accidents or public emergencies. Residents receive alerts from the California state alert system in case of emergency. The instruction may come from the fire department, sheriff’s or the police department. The residents follow the guidelines laid out by the local authorities. The Santa Clara county fire department provides services to all the cities in Santa Clara county in an emergency [3].

1.2.2. State of the Art for Public Safety

Public Safety using Location-Based Services in The United States

The Communications Technology Laboratory (CTL) at the National Institute of Standards and Technology (NIST) of the United States, have a division called Public Safety Communication Research (PSCR) where different research takes in place to identify the different problems that concern the public safety and propose different solutions using state of the art technology [4].

Among the different Research and Development (R&D) portfolios held by the PSCR division, there is one that focus specifically on public safety using indoor mapping, tracking and navigation technologies named Location Based Services (LBS) [5]. As such, they have the mission to accelerate the development of solutions for public safety by removing the impediments of adopting new technologies and the vision to position their portfolio of research, technology and approaches to products, standards, and systems [6].

In 2013, the PSCR division conducted a research with the goal to propose a roadmap that will instruct the government, industry, and academia to deliver enhanced LBS to public safety [7].

The PSCR reported that LBS are divided in three main sections [7]:

1. Software Applications
2. Devices
3. Networks

The scope of this project explores the first two sections only. Software applications and Devices.

Software and Applications

The LBS working group, had identified three components that need to be considered in order to solve public safety operations [7]:

- Front End (User experience and User interface)
- Back End (Data access and Data Management)
- Integrated Analytics.

Today, there are technologies that fully or partially operate over the previous components and enable partial solutions to the LBS problems. These technologies are:

- Digital 3-D mapping – Software and hardware that enable the mapping of three dimensional (3-D) environments. Currently there are companies that offer 3D mapping solutions. For example, Google released in 2018 ARCore, which is a software development kit that enables phone’s cameras to detect the size and location of flat horizontal surfaces [8].

- Augmented Reality (AR) – AR enable users to empower their visualization by adding visual objects in real time to improve their awareness of the physical environment optionally enhanced by sound [7].

Both 3-D mapping and AR present some gaps before they become fully operational for public safety. The following gaps were identified to be considered [7]:

- Not every public building is digitally mapped.
- There is not a centralized database to provide 3-D Maps.
- Location accuracy is not precise for indoor environments as much as for outdoor environments.
- It is not clear if wearables that provide AR capabilities will become widely successful.
- It is not tested if AR will not distract or interrupt users' primary functions in case of an emergency.

Devices

The PSCR division discussed multiple operational objectives to spot the devices that will enable to solve LBS challenges. These objectives are [7]:

- Device convergence supports greater mobility.
- Diversified approaches improve indoor accuracy positioning and situational awareness.

As a result, PSCR evaluated four components that would impact public safety operations. They include [7]:

- Global Positioning System (GPS)/Global Navigation Satellite System (GNSS).
- Cellular.
- Terrestrial Beacon.
- Short Range (e.g., Wi-Fi, BTLE, RFID, NFC, IoT).

1.2.3. State of The Art for Indoor Navigation Systems

In recent times, Global Positioning System (GPS) has helped us solve outdoor positioning problems. It proves to be a reliable system only when it comes to an outdoor environment. To navigate the user correctly in an indoor environment, first we must know the indoor positioning of the user. Indoor positioning enables to trace location of human beings, animals, and objects within the interior space [9]. An indoor environment requires a positioning accuracy in the scale of centimeters. However, the GPS provides accuracy on a centimeter scale. Some of its reasons are signal loss and the scattering caused by building walls. This issue is solved using Indoor Positioning System (IPS), made based on several technologies and techniques. Since it provides unlimited implementations in various applications, IPS has attracted many researchers. Existing infrastructures like Bluetooth beacons, Wi-Fi, internal sensors of smartphones and radio frequency identification (RFID) tags are some of the positioning methods that can be used within an indoor environment [9].

“The paper [10] represents the development of a low-cost optical indoor positioning system solution where a static commercial camera is the only sensor used.” [10] The system has tested classification, detection, and tracking techniques of an object on mobile robots and achieved high localization accuracy within the range of 1 centimeter. The resultant position details are compared against the original measurements followed by sending them to the rover using Message Queuing Telemetry Transport (MQTT), a lightweight broker-based publish/subscribe messaging protocol.

The above proposed indoor positioning techniques have limitations related to high energy consumption, consistency, and accuracy [11] . The thesis [11] by Viney Ugave proposes innovative techniques based on Machine Learning algorithm and smart sensor management. Many research works have been performed on several classifier and regression-based machine learning algorithms such as Naïve Bayes Classifier, Decision Trees, Linear Regression, Neural Networks and Reinforcement learning. The study performed in this thesis work shows that K-Nearest Neighbor (KNN) algorithm based LearnLoc approach works well in presence of noise and magnetic inference. KNN based approach outperformed other machine learning based approaches. This approach has low energy consumption.

Indoor Navigation [9] [12] [13] based on mobile device is common research area. Almost every mobile device such as smartphone, tablets have gyroscope, magnetometer, accelerometer that enables indoor navigation. An accuracy of indoor navigation using smartphone can be improved by incorporating mobile device sensors with Wi-Fi or BLE beacons. [9] [14]

1.2.4. Generating The Shortest Path

To navigate users to safety efficiently, finding the shortest path for the users is a must. The shortest path has been a topic almost already solved and not having any novel algorithm other than Dijkstra. Although, there is one interesting paper that can improve the calculation speed by applying extra preprocessing to the map, which is usually used on a large-scale map [15]. This might not be useful as indoor maps may not reach such a scale, but it is worth a try.

CHAPTER 2. Project Architecture

2.1. Clients and Cloud Infrastructure

To fulfill the established requirements and to serve the identified uses cases, a cloud solution is proposed as illustrated in Figure 1. Client and Cloud Infrastructure. This cloud solution serves three clients:

- Users (People inside the public space)
- Emergency triggers (Administrator or IoT devices)
- Monitoring tool (Used by first responders)

The following cloud services work as microservices.

- Device gateway - enables IoT devices to send and receive instructions with various cloud services securely.
- Authorization and Authentication service - Enable user registration, sign-in and access control credentials to interact with the different cloud services.
- API gateway - Once the user has the proper credentials to operate the cloud services, the API gateway describes the operations the user can do according to its credentials and establishes the communication protocols.
- Compute Service - To process the business logic. It directly interacts with the database and Machine learning services.

- Database Service - To store structured data related to the user's location, building, IoT devices, and monitoring data.
- Load Balancers – To distribute a set of tasks to a set of resources with the purpose of improving the overall performance of the system.
- Event Listeners – To respond to events from other services and call other services that are interested in such events.
- Machine Learning Service - To enable smart and accurate decision-making capabilities.
- Content Delivery Network - To provide non-structured data such as maps and ML models.
- Storage and Backup services - To store non-structured data such as maps and ML model data.

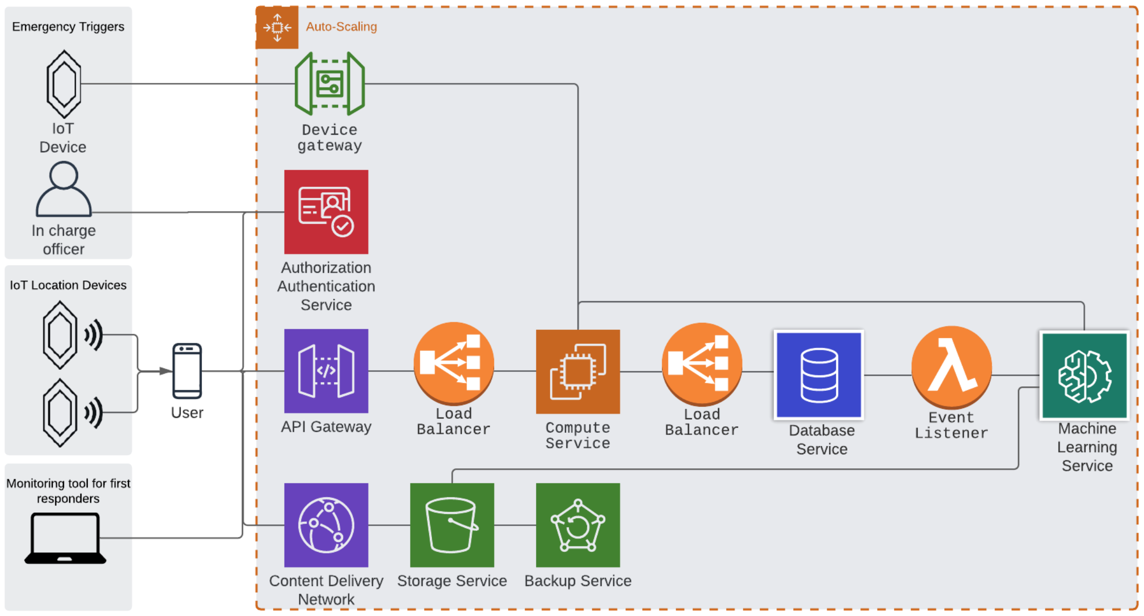


Figure 1. Client and Cloud Infrastructure
Source: Own work

2.2. Deployment Pipeline

The following deployment pipeline improves the development and deployment workflow, containing the following components as illustrated in Figure 2:

- Code repository: The code repository provides versioning and storage for the infrastructure's code.
- Continuous integration and continuous deployment tools to automate the testing, building, and deployment tasks.
- The infrastructure builder follows a set of instructions to build the cloud infrastructure defined as code (infrastructure as a code) to deploy or deploy services in the cloud.

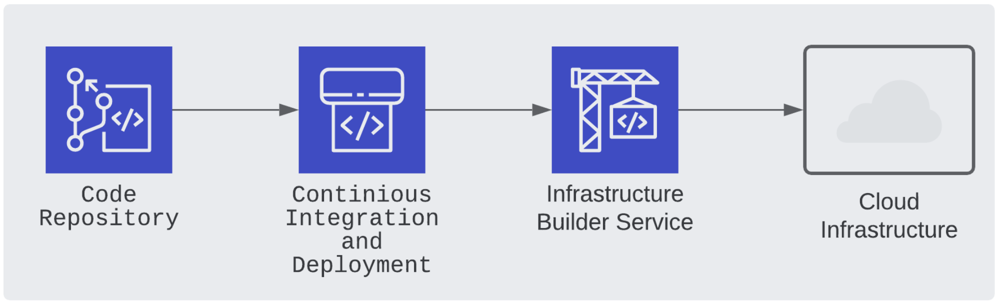

Code Repository
Continious Integration and Deployment
Infrastructure Builder Service
Cloud Infrastructure

Figure 2. Deployment Pipeline
Source: Own work

CHAPTER 3. Technology Descriptions

Following are the different software technologies and framework we used for our application.

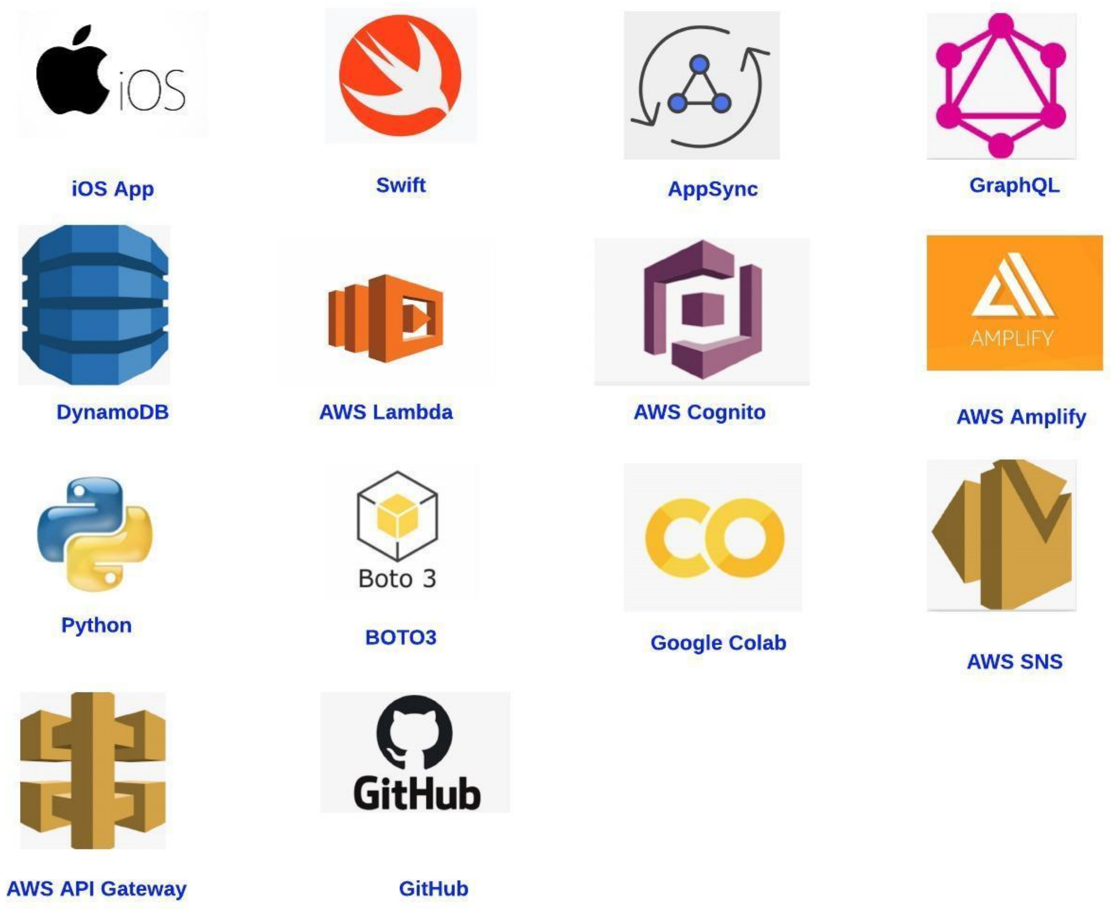


Figure 3. Technology Stack
Source: Own work [16]

3.1. Client Technologies

The project uses Xcode which is an iOS Integrated Development Environment (IDE) used for iOS mobile application development. Programming languages such as Swift and Python are used for iOS app development and AWS SNS functionality implementation, respectively. Software frameworks such as Apple MapKit, Core Locations, AWS Amplify, AWS AppSync, Apple ARKit, Apple Storyboards and Apple Push Notifications have been used for mobile application development.

3.1.1. Mobile App and Monitoring Tool Components

The mobile app and the monitoring tool use the following Software Development Kits (SDK's) as illustrated in Figure 4. Mobile App and Monitoring Tool Components:

- Geo Location SDK
    - Provides different tools for developers to use geolocation capabilities in smartphone devices.
    - Internally uses a data structure for indoor maps called GeoJSON [17].
- Network SDK
    - Allow developers to use networking functions.
    - To interact with the cloud infrastructure.
- Augmented Reality SDK

  - Augmented Reality works in conjunction with other SDKs such as the camera controllers, environment sensor controllers, and machine learning controllers.

- User Interface SDK
  - The user interface SDK provides deferent tools for visualization and User Experience (UX) enrichment.

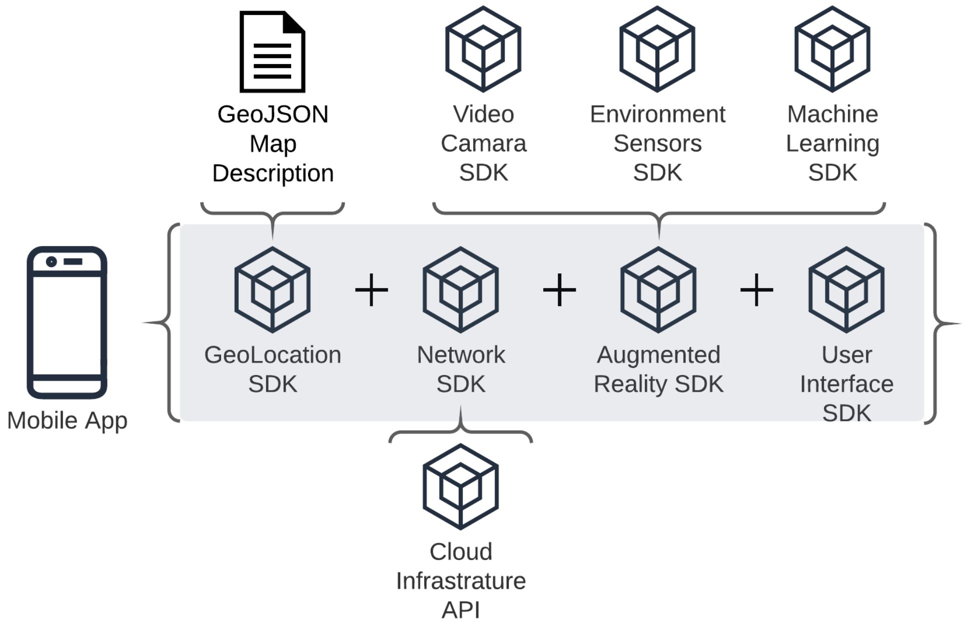


Figure 4. Mobile App and Monitoring Tool Components
Source: Own work

3.2. Middle-Tier Technologies

3.2.1. AWS Cloud Services Setup

AWS account is created and each member in the team is provided with separate login using IAM user to setup or access the services required for the application development.

AWS serverless compute Lambda functions is configured with Python language to trigger notification in an emergency and generate a path to the safe area.

3.3. Data-Tier Technologies

The database is implemented using AWS DynamoDB non-relational database. GraphQL is used for seamless access of database from the iOS application using AWS Amplify and AppSync services.

CHAPTER 4. Project Design

4.1. UI Mockups

The system provides two applications, one for regular users and another for the first responders.

4.1.1. User Application

The user's application will receive a notification in case of an emergency. Once the user opens the application, the user will receive the instructions to escape, as illustrated in Figure 5. User's application's Mockup – 2D and 3D maps. The top section uses AR (Augmented reality) to show 3D signs, so the user has a better perspective in tridimensional space. In contrast, the bottom section shows a map in 2D to give the user a better sense of perspective. Both sections can be open in full-screen mode, as illustrated in Figure 6.

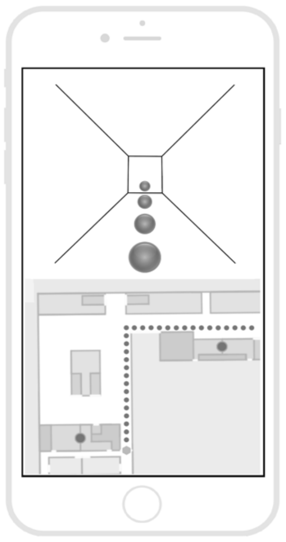


Figure 5. User's application's Mockup – 2D and 3D maps
Source: Own work

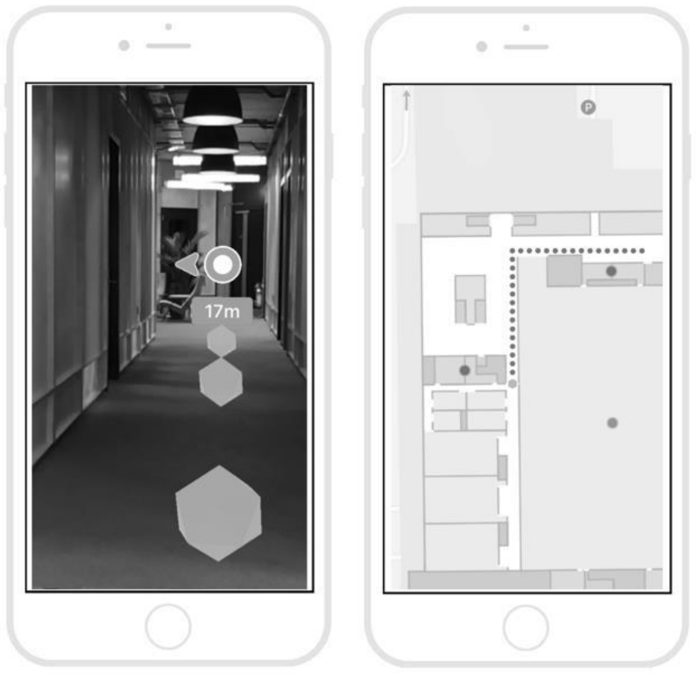


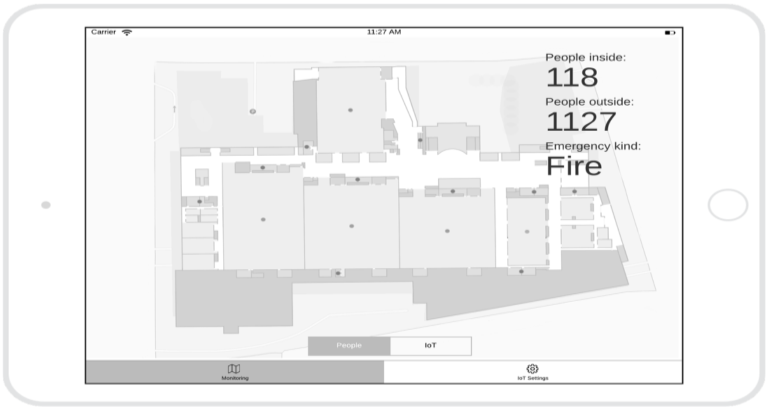
Figure 6. User's application's Mockup – Full screen maps

4.1.2. First Responder's Application

The First Responder's application provides two sections, the map section, and the IoT settings section. The map section shows all the users and their current location inside the building. Inside the map, first responders can see the user's location, all the IoT devices inside the building, the number of people inside and outside the building, and the emergency type that triggered the alarm.

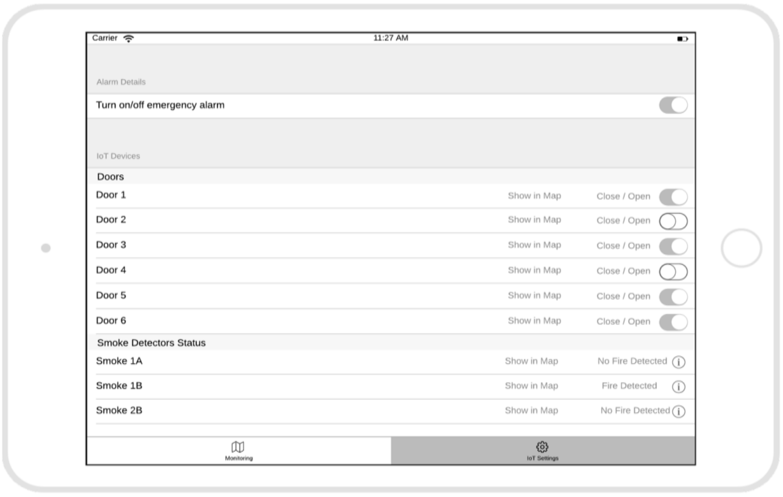
In a separate section, as illustrated in Figure 8. First Responders' Mockup – IoT Settings, the first responders can turn the alarm on and off and control other IoT devices inside the building, such as door locks and fire detectors.

Below, Figure 7. First Responder’s Mockup – Map illustrates the look and feel for the map section.



People inside:
118
People outside:
1127
Emergency kind:
Fire

Figure 7. First Responder’s Mockup – Map
Source: Own work



Alarm Details

Turn on/off emergency alarm

IoT Devices

Doors		
Door 1	Show in Map	Close / Open
Door 2	Show in Map	Close / Open
Door 3	Show in Map	Close / Open
Door 4	Show in Map	Close / Open
Door 5	Show in Map	Close / Open
Door 6	Show in Map	Close / Open
Smoke Detectors Status		
Smoke 1A	Show in Map	No Fire Detected
Smoke 1B	Show in Map	Fire Detected
Smoke 2B	Show in Map	No Fire Detected

Figure 8. First Responders’ Mockup – IoT Settings
Source: Own work

4.2. Package Diagram

The users and first responder software application follow the design patterns proposed by the Elements Architecture Pattern. This architecture is divided into two main categories as illustrated in Figure 9, Core Logic and User Interface Logic.

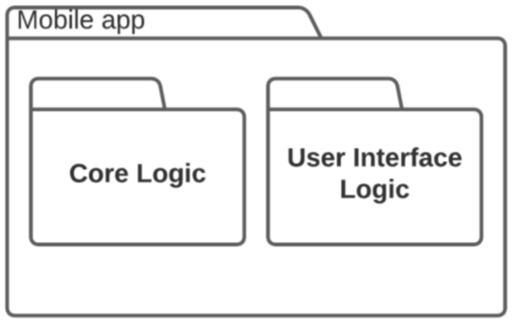


Figure 9. Mobile App Architecture Pattern – Package Diagram
Source: Own work

4.2.1. Core Logic

The Core Logic package (as illustrated in Figure 10. Core Logic – Package Diagram) is composed of Entity package, Data Store package, Remote API package, Use Case package, and Broadcaster package. This package interacts with the User Interface Logic package through the Use Case Package.

Entity package

Here are placed all data-objects. Their only responsibility is to stores values while the application is running.

Data Store package

In the Data Store package is the logic to CRUD the different entities from and to the data store.

Remote API package

This package contains different network calls to external APIs.

Use Case package

The Use Case package separates the app's core logic from the user interface logic. The purpose of this package is to isolate the logic of the different use cases of a feature. Also, the test framework can consume the resources from this package.

Broadcaster package

Inside this package are all broadcasters. If something changes in the app, Broadcasters notify their subscribers. Multiple subscribers can subscribe to a broadcaster.

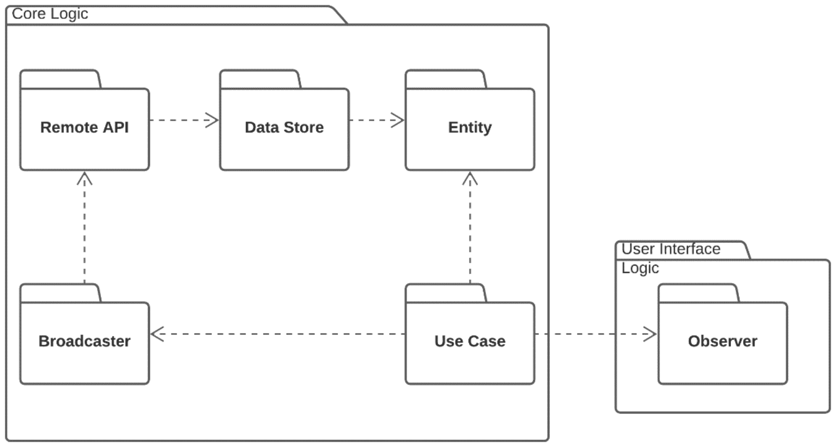


Figure 10. Core Logic – Package Diagram
Source: Own source

4.2.2. User Interface Logic

The User Interface Logic package (as illustrated in Figure 11. User Interface Logic – Package Diagram) separates all the user interface logic from the business logic. This package contains packages such as User Interface Package, Observer Package, Interaction Responder package, and Displayable entity package. The logic to interact with the Core Logic package is through the Observer package.

Displayable entity package

The Displayable entity package handles the logic to display the different entities from the Core Logic Package and customizations to the presented data.

Observer package

The Observer package stores the objects that receive input signals from the Use Case Package. Observers know how to subscribe to the different publishers from the Core logic.

User Interface package

In the User Interface package are the different UI components and the rendering logic and animations.

Interaction Responder package

The Interaction Responder package holds the objects responsible for responding to the user interface events and the logic to display the data through the User Interface.

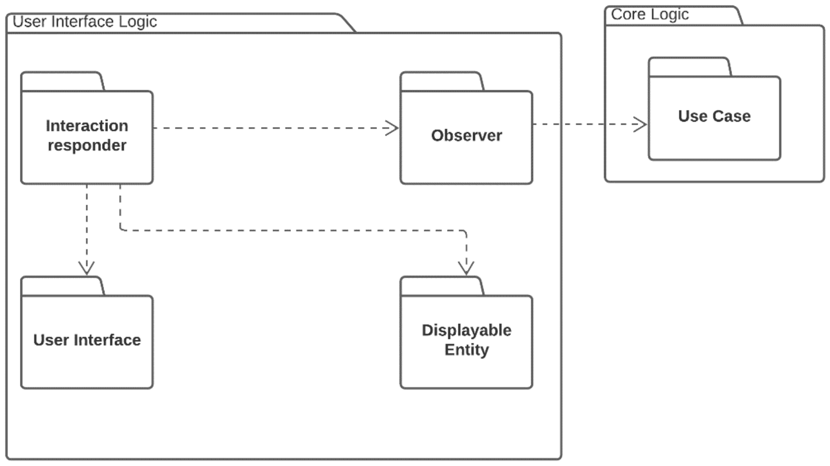


Figure 11. User Interface Logic – Package Diagram
Source: Own work

4.3. Deployment Diagram

As illustrated in Figure 12. Deployment Diagram describes the system's layout by specifying the different operative systems, physical devices, and web services.

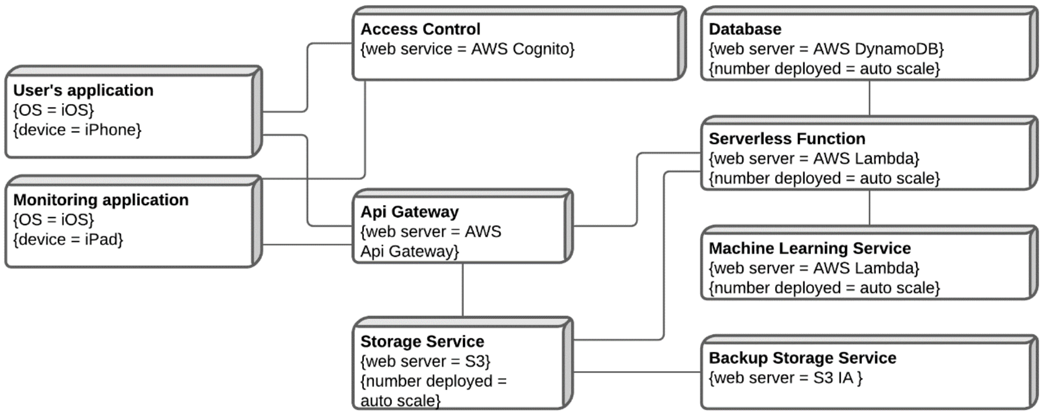

Access Control
{web service = AWS Cognito}
Database
{web server = AWS DynamoDB}
{number deployed = auto scale}
User's application
{OS = iOS}
{device = iPhone}
Serverless Function
{web server = AWS Lambda}
{number deployed = auto scale}
Monitoring application
{OS = iOS}
{device = iPad}
Api Gateway
{web server = AWS Api Gateway}
Machine Learning Service
{web server = AWS Lambda}
{number deployed = auto scale}
Storage Service
{web server = S3}
{number deployed = auto scale}
Backup Storage Service
{web server = S3 IA }

Figure 12. Deployment Diagram
Source: Own work

4.4. Class Diagram

The Figure 13 describes the main building blocks of the model. The diagram describes the various objects in the system, their attributes, their operations, and the relationships among them.

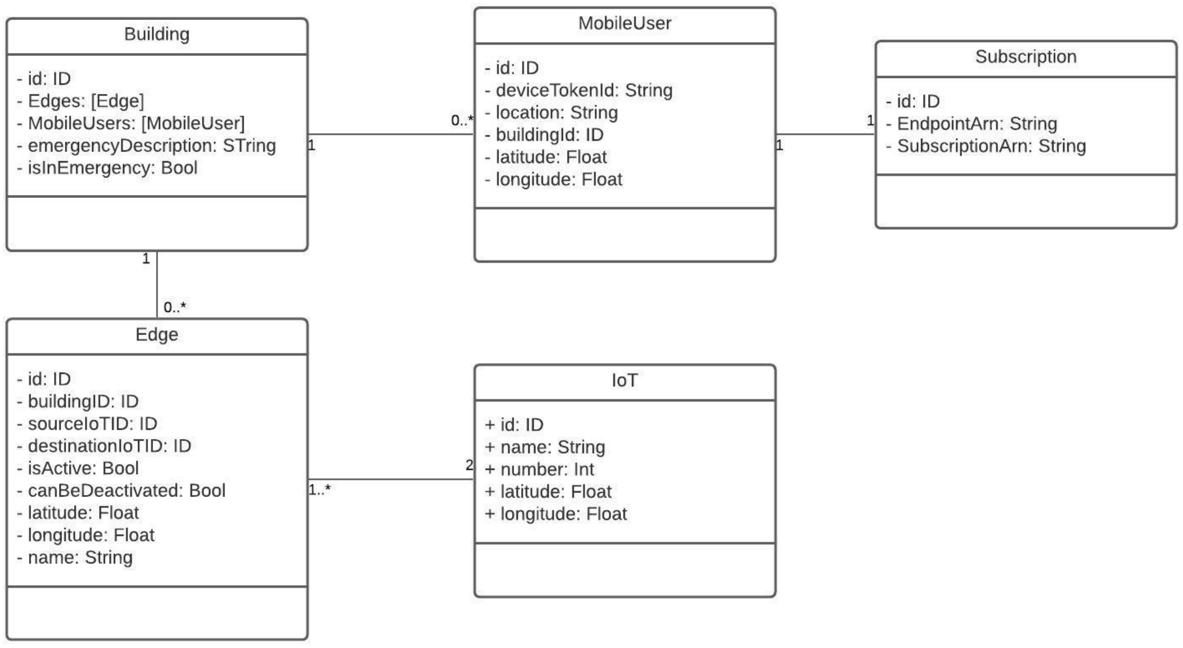


Figure 13. Class Diagram
Source: Own work

4.5. Database Entity Diagram

The Figure 14 describes how all the data will be stored in our NoSQL database.

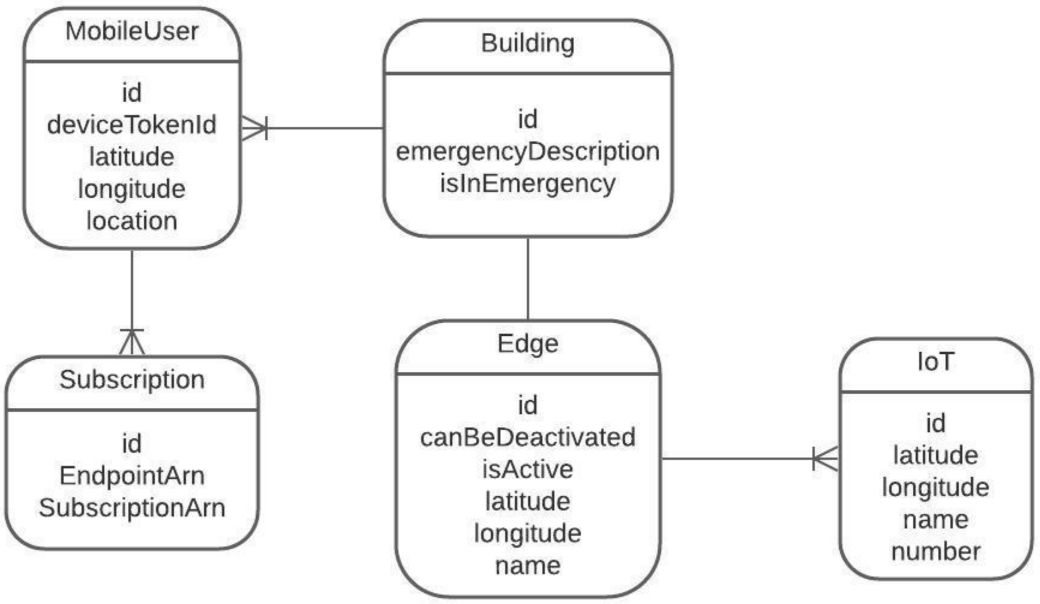


Figure 14. Database Entity Diagram
Source: Own work

4.6. Use Case Diagram

The Use case diagram shows the relationship between the user and the different use cases in which the user is involved.

4.6.1. Mobile App Use Case Diagram

The Figure 15. Mobile App Use Case Diagram depicts the use cases available to the smart device user. Since the user will access the application through their smart device, following use cases shall allow the end-user to access the different features of the mobile app. The user can receive notifications and navigation directions and access information through a secure cloud-based login.

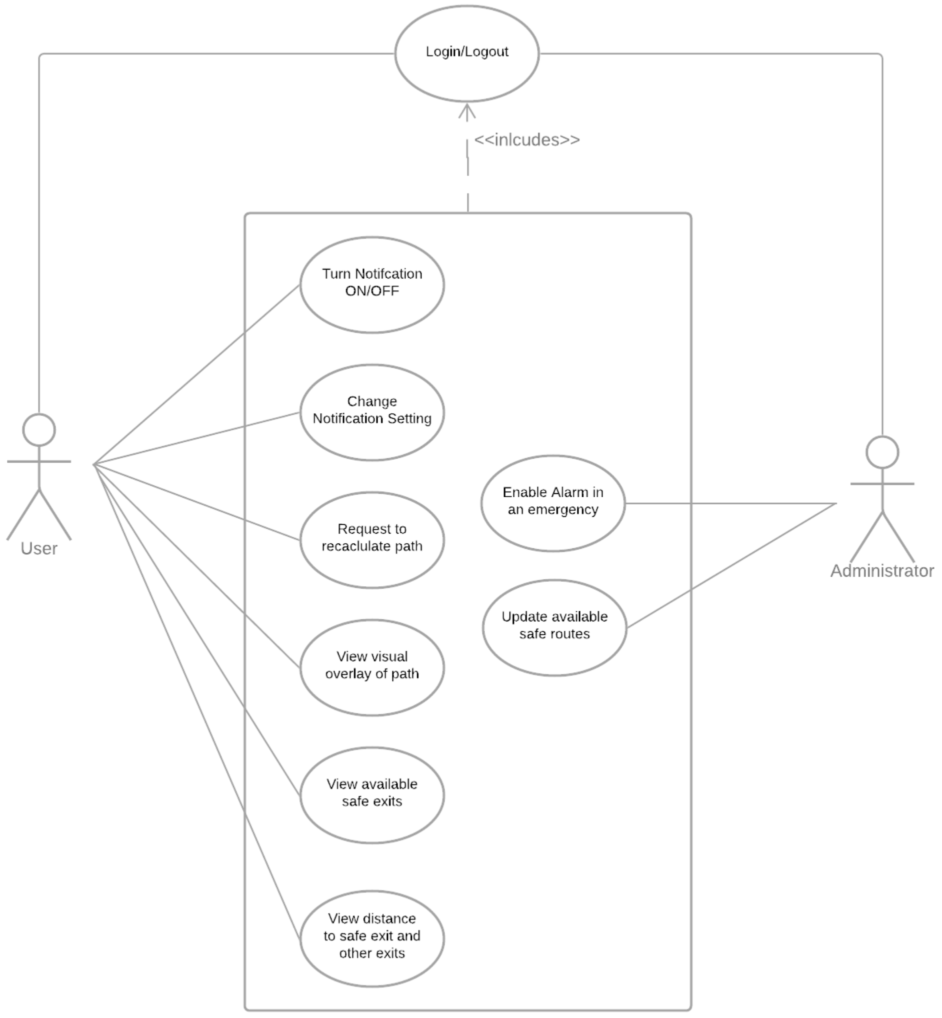

Login/Logout
<<inlcudes>>
Turn Notifcation ON/OFF
Change Notification Setting
Request to recaclulate path
View visual overlay of path
View available safe exits
View distance to safe exit and other exits
Enable Alarm in an emergency
Update available safe routes
User
Administrator

Figure 15. Mobile App Use Case Diagram
Source: Own work

4.6.2. Monitoring App Use Case Diagram

The Figure 16. Monitoring App Use Case Diagram shows the use case for first responders or anyone who will be monitoring the building. Users can register and retrieve the building's information.

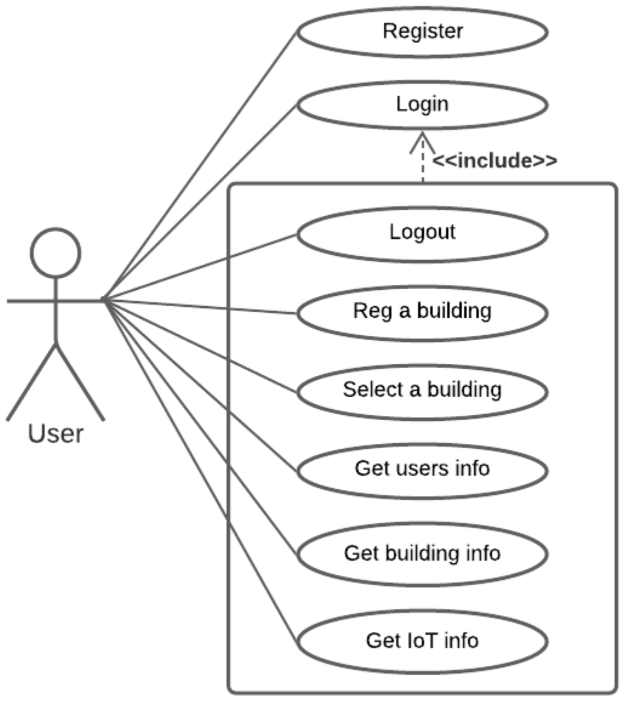


Figure 16. Monitoring App Use Case Diagram
Source: Own work

4.7. Sequence Diagram

Figure 17. New User Sequence Diagram shows the workflow of creation of a new user.

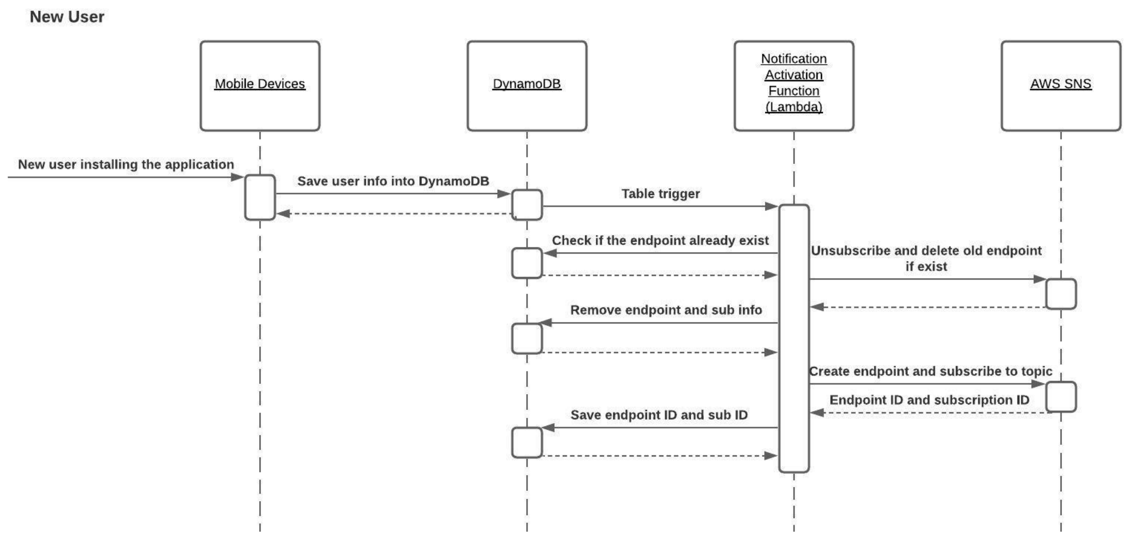

New User
Mobile Devices
DynamoDB
Notification Activation Function (Lambda)
AWS SNS
New user installing the application
Save user info into DynamoDB
Table trigger
Check if the endpoint already exist
Unsubscribe and delete old endpoint if exist
Remove endpoint and sub info
Create endpoint and subscribe to topic
Endpoint ID and subscription ID
Save endpoint ID and sub ID

Figure 17. New User Sequence Diagram
Source: Own work

Figure 18. Update User Token Sequence Diagram shows the workflow of when a user's token gets updated (token is for registering for notification system).

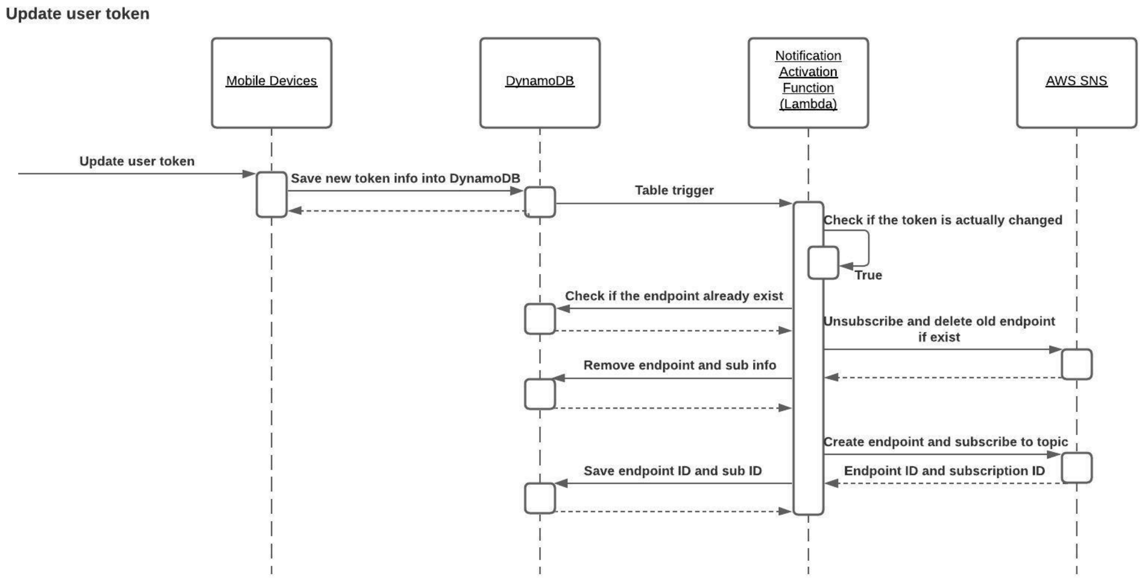

Update user token
Mobile Devices
DynamoDB
Notification Activation Function (Lambda)
AWS SNS
Update user token
Save new token info into DynamoDB
Table trigger
Check if the token is actually changed
True
Check if the endpoint already exist
Unsubscribe and delete old endpoint if exist
Remove endpoint and sub info
Create endpoint and subscribe to topic
Save endpoint ID and sub ID
Endpoint ID and subscription ID

Figure 18. Update User Token Sequence Diagram
Source: Own work

Figure 19. Remove User Sequence Diagram shows the workflow of how to cancel the notification registration when a user uninstalls the application.

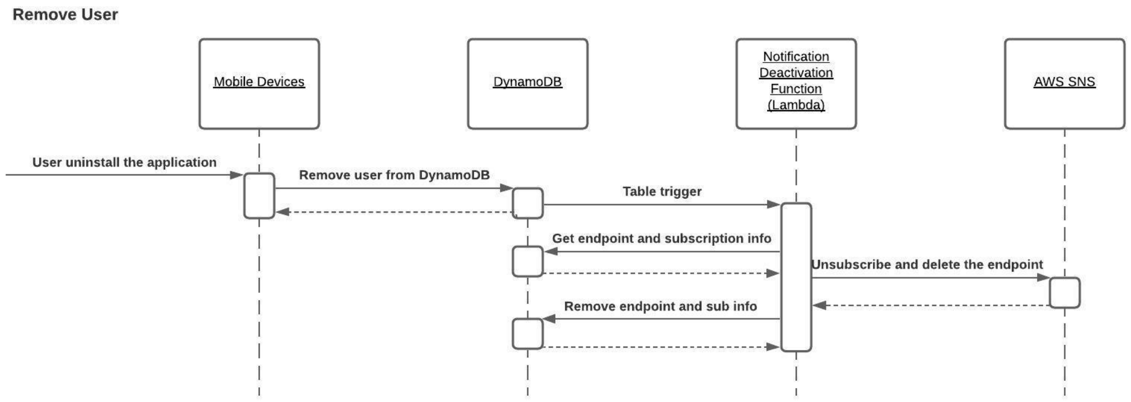

Remove User
Mobile Devices
DynamoDB
Notification Deactivation Function (Lambda)
AWS SNS
User uninstall the application
Remove user from DynamoDB
Table trigger
Get endpoint and subscription info
Unsubscribe and delete the endpoint
Remove endpoint and sub info

Figure 19. Remove User Sequence Diagram
Source: Own work

Figure 20. Trigger Emergency Sequence Diagram shows the workflow of how an emergency is triggered.

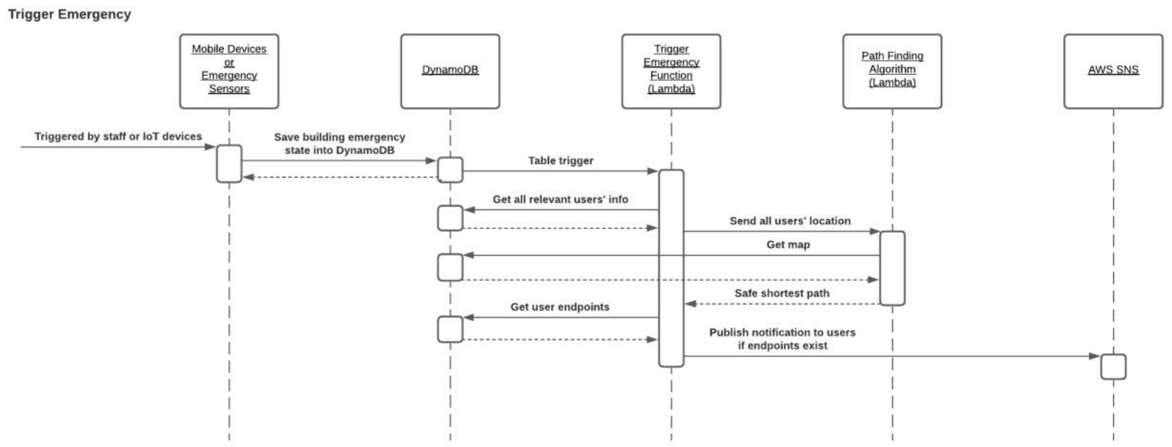

Trigger Emergency
Mobile Devices or Emergency Sensors
DynamoDB
Trigger Emergency Function (Lambda)
Path Finding Algorithm (Lambda)
AWS SNS
Triggered by staff or IoT devices
Save building emergency state into DynamoDB
Table trigger
Get all relevant users' info
Send all users' location
Get map
Safe shortest path
Get user endpoints
Publish notification to users if endpoints exist

Figure 20. Trigger Emergency Sequence Diagram
Source: Own work

Figure 21. Map Update Sequence Diagram shows the workflow of how the notification system resends the notification when a map gets updated.

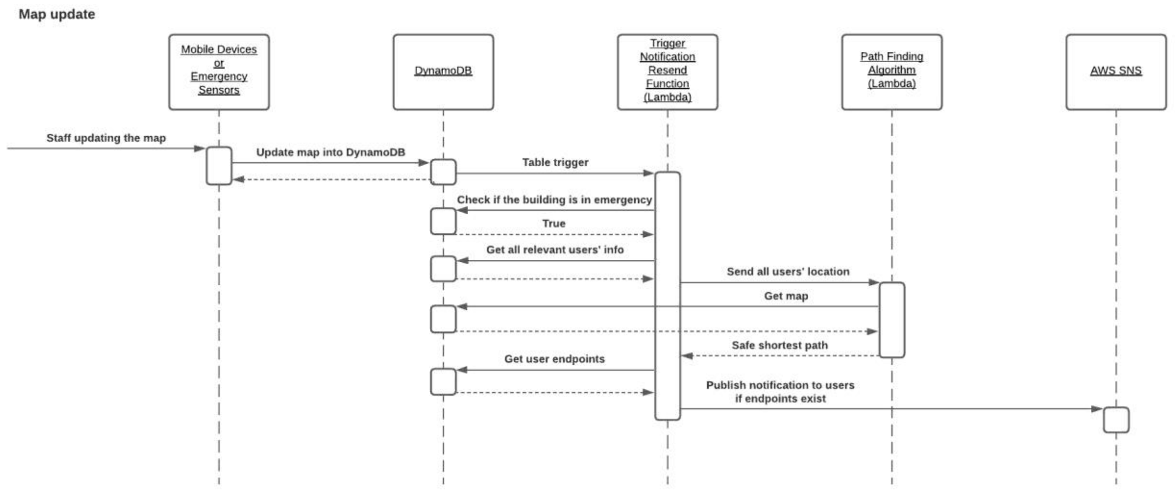
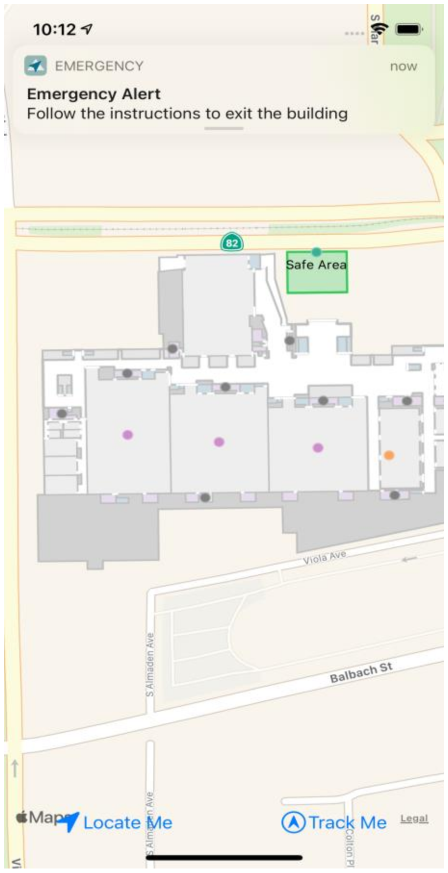

Map update
Mobile Devices or Emergency Sensors
DynamoDB
Trigger Notification Resend Function (Lambda)
Path Finding Algorithm (Lambda)
AWS SNS
Staff updating the map
Update map into DynamoDB
Table trigger
Check if the building is in emergency
True
Get all relevant users' info
Send all users' location
Get map
Safe shortest path
Get user endpoints
Publish notification to users if endpoints exist

Figure 21. Map Update Sequence Diagram
Source: Own work

CHAPTER 5. Project Implementation

5.1. Client Implementation

1. **Emergency Alert:** User receives emergency alert notification on his/her phone screen in case of emergency. below figure shows the mobile application screen with the emergency alert notification.



Emergency Alert
Follow the instructions to exit the building

Figure 22. Emergency Alert Screen
Source: Own work

2. **Navigation path to the Safe Area:** Once user opens application, the application shows the screen with a path to the safe area from his/her current user location. If the premise is still under emergency, it shows red background to show the status of the situation. below figure is a screen capture of our mobile application in case of emergency. The application shows path from user's current location to the safe area.

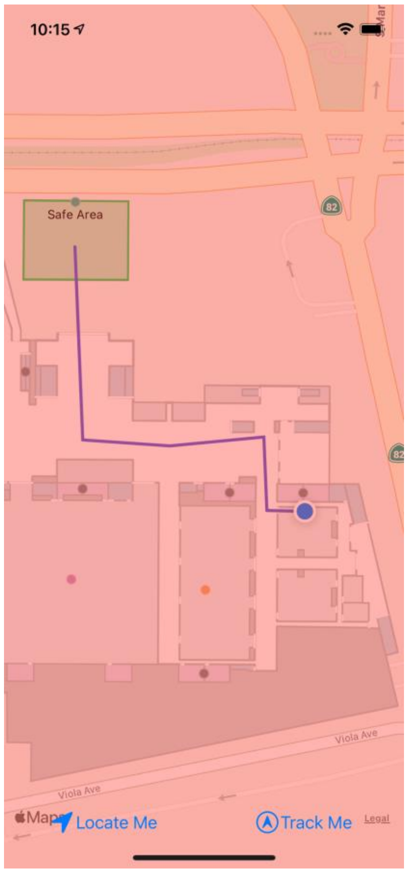
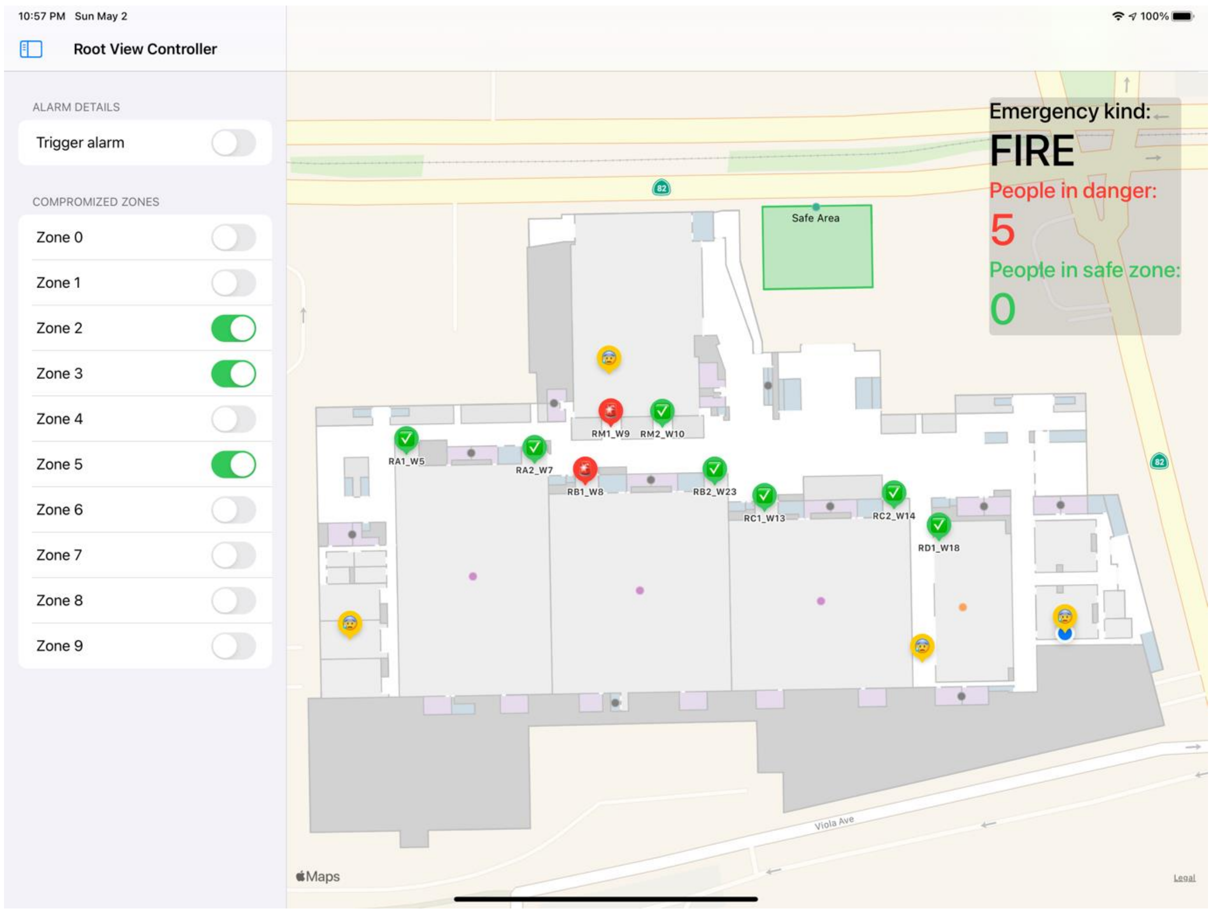


Figure 23. Application screen showing path to the safe area
Source: Own work

3. **First Responder’s Application:** below figure shows the First Responder’s Application.


10:57 PM Sun May 2

Root View Controller

ALARM DETAILS

Trigger alarm

COMPROMIZED ZONES

Zone 0

Zone 1

[✓ Zone 2]

[✓ Zone 3]

Zone 4

[✓ Zone 5]

Zone 6

Zone 7

Zone 8

Zone 9

Emergency kind:
FIRE

People in danger:
5

People in safe zone:
0

Figure 24. First Responder's Application
Source: Own work

5.1.1. Deployment

This project uses the following technologies for the first version of the application:

- Apple iOS is the supported mobile operative system for the first version of the application.
- AWS Cognito is used for authorization and authentication.
- AWS API gateway is used as the API Gateway for cloud services.
- AWS Lambda Functions are used as serverless functions to request CRUD operations from the database.
- AWS DynamoDB is used as a database for the non-sensitive users' data. Other web services may consume the data from this database.
- AWS SNS is used to prepare custom notification for users. Each user may have different instructions.
- Apple Push Notifications is used to send custom notifications to iOS users.

5.1.2. Software Frameworks

The following table shows the frameworks used to develop the mobile application.

Apple MapKit	Software Development Kit (SDK) to visually show a map. Also, it provides development tools for indoor maps, annotations, and custom drawings.
Core Locations	SDK to read the GPS, WIFI, and Bluetooth data as input data for location services.

	General location operations.
Apple ARKit	SDK to add AR capabilities such as reality composer and reality converter.
Apple Storyboards	SDK to implement custom user interfaces.
Apple Combine	SDK that provides a modern declarative approach for app development, rather than the usual design patterns used by iOS Development such as delegate callbacks and completion handlers.
Apple Push Notifications	SDK to receive notifications from external notification services.
AWS Amplify	SDK to communicate with Amazon Web Services.
AWS AppSync	SDK to sync data between the mobile app and the AWS DynamoDB.
network	Python library used to define San Jose Conventional Center in graphical format.
Matplotlib	Python library used to plot a graph.
Numpy	Python library used to perform mathematical operations on multi-dimensional array.

Pandas	Python library used to save and load the updated Q matrix after training in the form of CSV file.
Random	Python library used to generate a random number in Machine Learning Algorithm implementation.
Boto3	Python library used to access all AWS resources.
json	Python library used to read/write json format files.

Table 1. Software Frameworks
Source: Own work

5.1.3. Software Development Tools

The table below shows the development tools to implement the users' mobile application and the back-end services.

Xcode iOS	Integrated Development Environment (IDE). It also provides a testing environment for unit tests and UI automation.
Swift	Programming language to develop the iOS App, unit tests, and UI test automation.

Python	Programming language to implement the functionality of AWS SNS and Machine Learning algorithm.
Google Colaboratory	An interactive notebook environment to implement the functionality of finding the shortest path from user location towards the safe exit.
VS Code	Visual Studio Code is a code editor redefined and optimized for building and debugging modern web and cloud applications.

Table 2. Software Development Tools
Source: Own work

5.2. Middle-Tier Implementation

5.2.1. Serverless Functions

All back-end computation related operations are executed using AWS Lambda serverless functions and the code is implemented using Python programming language. Refer to Table 1. Software Frameworks to see all the library used. Refer to Chapter 4.7 to see how each intermediate function works. All the intermediate functions are called by using AWS DynamoDB table trigger, which means whenever there is any change in the table, certain function(s) will be executed.

5.2.2. Machine Learning Model

To generate a shortest path, the Q-learning algorithm is used. Q-learning is a model free Reinforcement Learning algorithm. As shown in the Figure 25. Reinforcement Learning, it involves an agent who observes the environment, decides the action to be taken based on some strategy and takes an action according to the optimal strategy. Based on the action taken, it receives a reward or a penalty. The reinforcement learning agent learns from its experiences and refines the strategies to reach towards the final goal. It keeps on iterating until an optimal strategy is identified.

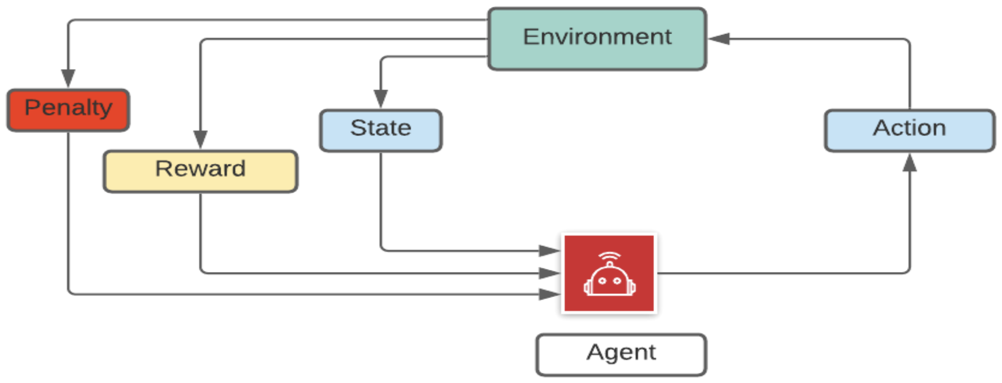


Figure 25. Reinforcement Learning
Source: Own work

In above figure, we have environment state (S), action (A), reward (R) and penalty.

Q-Learning is a value-based algorithm which uses a Q function to get an optimal action selection policy. It uses a reward matrix and Q matrix. The Q matrix is representation of agent's memory of learnings from its experiences. The reward matrix consists rewards associated with a combination of different states and actions taken by agent.

The Q-function takes environment state and action taken by agent as input. It uses the Bellman equation. Below is the Q-function using which the Q-matrix is populated with the Q-values.

*Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]*

5.2.2.1. Q-Learning algorithm implementation for Shortest Path generation

For our problem statement, the environment used is San Jose Convention Center. Figure 26 shows the map of San Jose Convention Center.

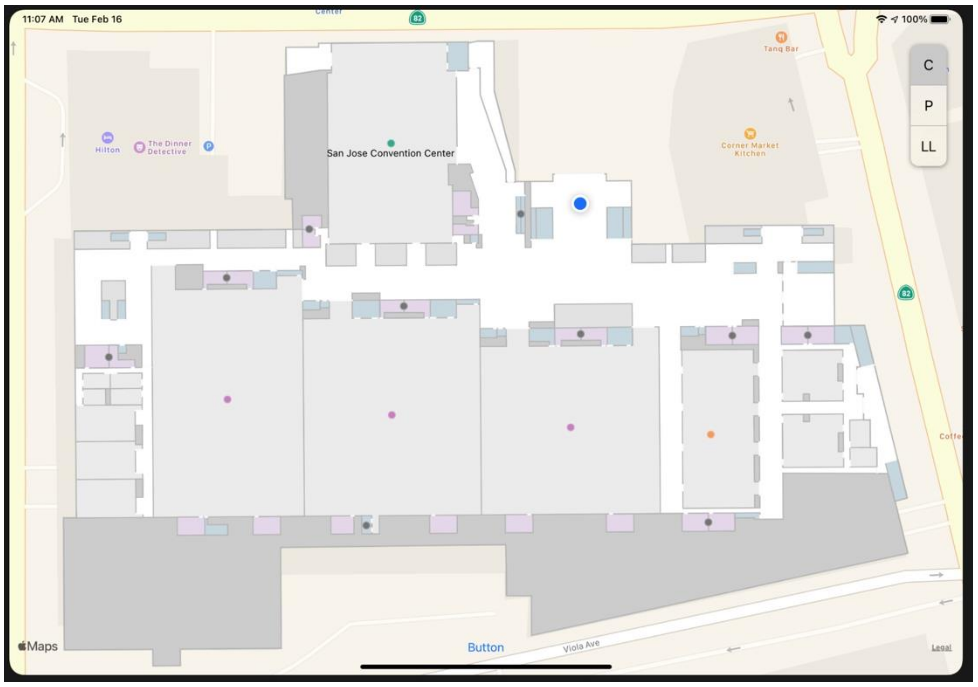


Figure 26. Map of San Jose Conventional Center
Source: WWDC 2019 – Apple Inc
https://developer.apple.com/documentation/mapkit/displaying_an_indoor_map

For indoor navigation in San Jose Convention Center, we have implemented the Q-Learning Algorithm in the form of a graphical environment. As shown in Figure 27, this graph has IoT devices present in the indoor environment of San Jose Convention Center represented as nodes. It also has edges connecting to the IoT devices present inside rooms, doors, and walkways.

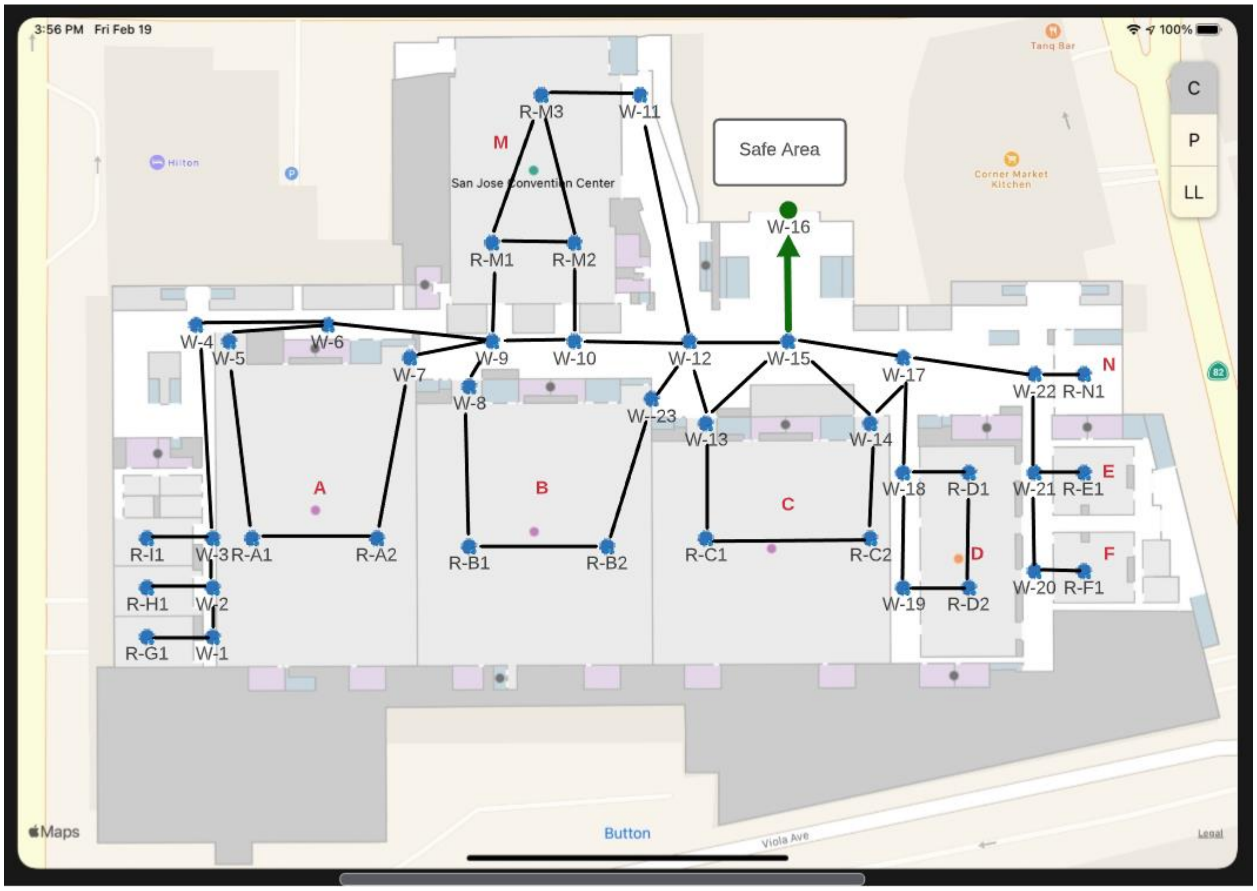


Figure 27. Updated map in graphical format
Source: Own work

The Q matrix and Reward(R) matrix are generated based on the map shown in Figure 27. Further we used Q function to update the Q matrix as we train the agent. Once the training was completed, we generated shortest path for the given origin and destination in San Jose Conventional Center.

5.3. Data-Tier Implementation

All data is stored using AWS DynamoDB tables. Data format is designed using AWS Amplify which gives the ability of querying the database using AWS AppSync with GraphQL provided proper API keys. Each table can also be accessed directly using Boto3 (Python library used to access any AWS resources) given proper IAM policies. There are several REST APIs developed to access those tables at the early designs but are not used in production. The REST APIs are kept for potential future use.

Backend API Design

S. No	Endpoint	Description
1.	POST /SendNotification	Send a notification to all users.
2.	GET /MobileUser	Get all users information
3.	POST /MobileUser	Create a new user
4.	PUT /MobileUser	Update a user’s information
5.	GET /MobileUser/{id}	Get a user by ID
6.	DELETE /MobileUser/{id}	Delete a user by ID
7.	GET /ActivateNotification/{id}	Activate the user’s notification function by ID

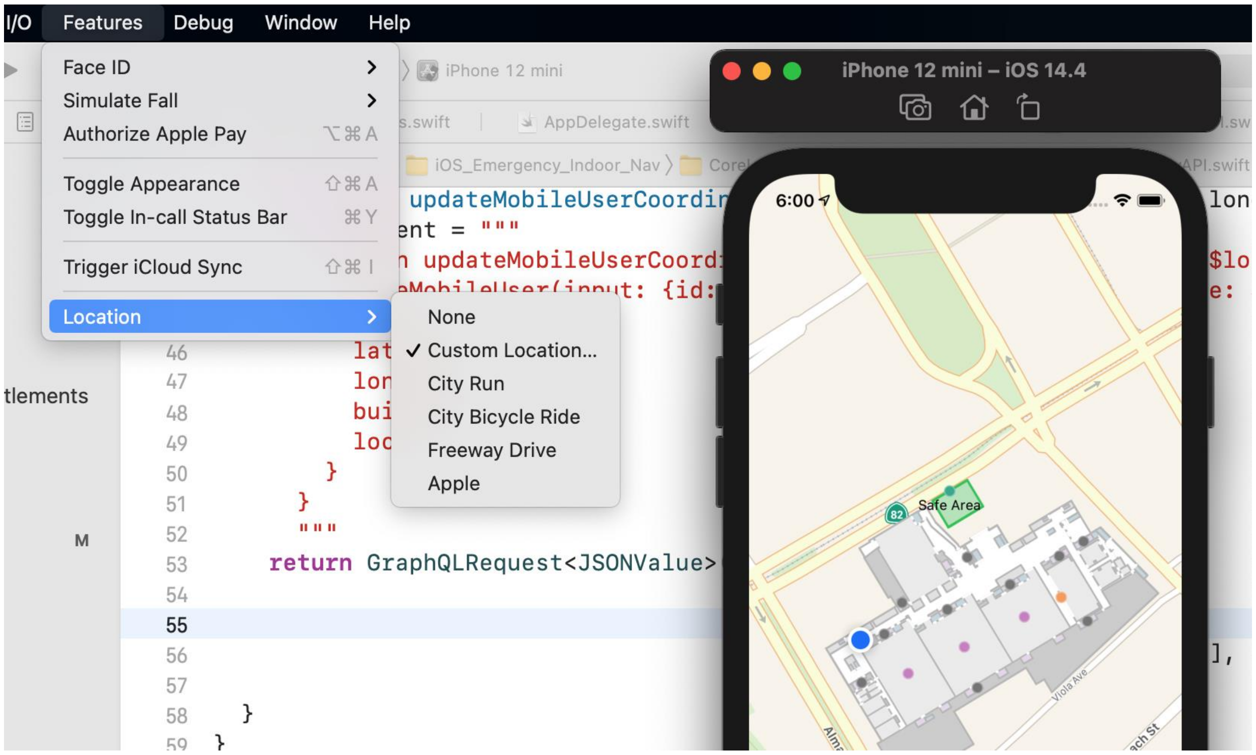
8.	GET /DeactivateNotification/{id}	Deactivate the user's notification function by ID
9.	GET /TriggerEmergency/{id}	Trigger emergency by building ID
10.	GET /TriggerNotificationResend/{id}	Resend emergency notification by building ID

Table 3. Backend APIs
Source: Own work

CHAPTER 6. Testing and Verification

6.1. Location Simulation and Location Testing

Due to the development of this project involves the use of localization services, it was essential to simulate the user's localizations rather than testing in real places. To overcome this issue, Apple's Xcode IDE can simulate different iOS devices (iOS and iPad). In addition, Xcode also facilitates the capability to simulate custom locations by only providing the location's latitude and longitude, and also is possible to simulate movement in a specific area by providing a path of coordinates and the time lapse between coordinates. Figure 28 shows the integrated feature to simulate locations in Xcode.



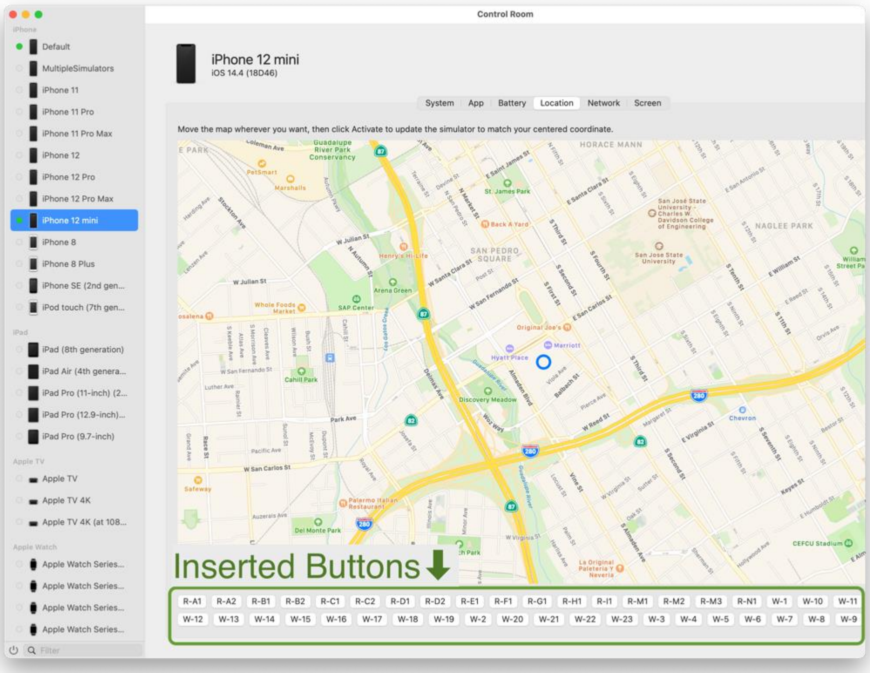
```
 updateMobileUserCoordin                          lon
ent = """
n updateMobileUserCoord                           $lo
eMobileUser(input: {id:                           e:
          lat
          lon
          bui
          loc
        }
      }
      """
    return GraphQLRequest<JSONValue>

                                                  ],

  }
}
```

Figure 28: Xcode localization feature
Source: screenshot

During the testing process, it was necessary to simulate real-life scenarios where multiple devices are inside a building. Thus, to facilitate this process, an open-source project was used to increase the capabilities of the native Xcode simulator [18]. The open-source project was modified by adding buttons to efficiently simulate the device's location to locations of interest. The added buttons position the user in a specific location. (See Figure 29).



Move the map wherever you want, then click Activate to update the simulator to match your centered coordinate.

Figure 29. Control Room – Open- source project to simulate custom locations in multiple devices
Source: screenshot

6.2. Unit Testing

Unit Testing is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.

S. No.	**Description**
1.	Test to validate if the request to the REST APIs have all the required parameters.
2.	Test to verify exception handling if incorrect data is provided as input to API
3.	Test to verify only authorized and authenticated users can access the First Responder's application
4.	Test to verify the accuracy of the safe navigation path provided by the API
5.	Test to verify the backend API provides alternate path when the original path is blocked and there is a secondary safe path available
6.	Test to verify backend API provides most accurate location of the simulated IoT devices

7.	Test to verify the UI displays correct path on the mobile app to the safe area
8.	Test to verify the first responder’s app displays the correct numbers of users inside the building

Table 4. Unit Testing
Source: Own work

6.3. System Integrity Testing

Flow	**Scenario**	**Expected Result**
User Subscription	User subscribes to the Smart Navigation App	User receives a notification regarding subscription and will be notified in case of emergency
User Authentication	User logins to the first responder’s app using valid credentials	App successfully navigates user to Dashboard
	User logins with incorrect credentials	Error message is displayed to verify credentials

Emergency Trigger	Emergency trigger is enabled	All users present inside the building are notified
Safe Path Navigation	Path to safe area is determined for each user inside the building	The designated safe path is displayed on the user’s screen
	User follows the safe path suggested	Safety message is displayed once user successfully navigates to safe area
	User deviates from suggested safe path	The current location of the user will be tracked and safe path will be displayed from the new location
Update area	If any area is not safe first responders can block it	The blocked areas are highlighted in the first responder’s app
Blocked Areas	If there is a blockade in the path to the safe area	User receives an alternate safe path if available

	If there is a blockade and no safe path is available	User receives a message and needs to wait for first responders. First responders are notified

Table 5. System Integrity Testing
Source: Own work

6.4. Function Testing

Functional Testing is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

Functional testing mainly involves black box testing, and it is not concerned about the source code of the application. This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application Under Test. The testing can be done either manually or using automation.

In functional testing all the primary functions of the application were tested to ensure the end-to-end functionality of the application works.

6.5. Machine Learning Model Testing

To evaluate the performance of the trained Machine Learning model the rewards per episode are considered. The reinforcement learning agent is trained for 50000 episodes. Each episode is a sequence of environmental states, actions taken, rewards gained by agent and a terminal state. Figure 30 shows the total number of rewards gained by Agent per 1000 episodes. For first 1000 episodes the total mean rewards were 152.6. Later with further training, the model performance improved along with efficiency of the agent. For the final set of 1000 episodes, the agent gained 62,112.4 mean rewards.

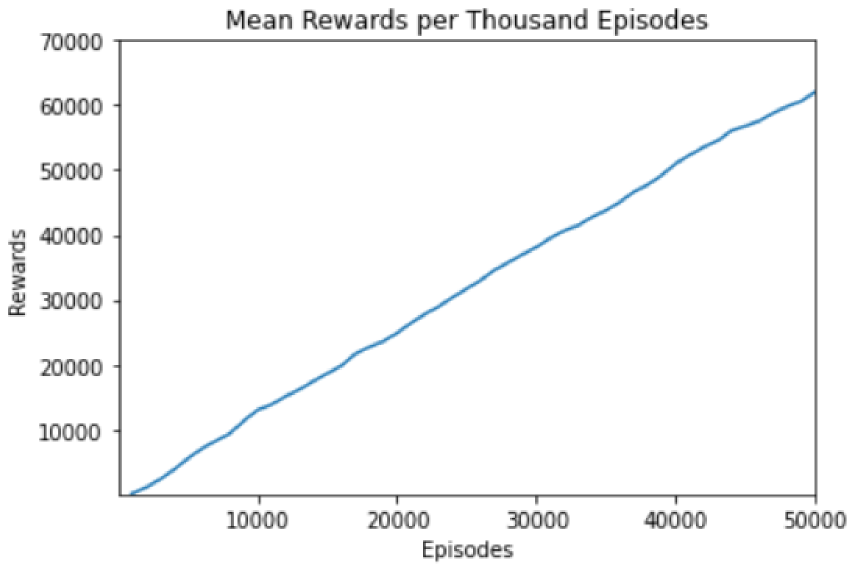


Figure 30. Machine Learning Algorithm Evaluation
Source: Own work

To test correctness of path generated by the Machine Learning model, Black Box testing is performed on set of 50 different test cases. Table 6 consists of some manual test cases used to verify the paths generated by model in different scenarios. These test cases also help to verify the path generated by Machine Learning model when one or more zones in San Jose Conventional Center are blocked due to security reasons in case of fire.

No.	**Scenario**	**Test Case**	**Expected Output**	**Actual output**
1.	No zones are compromised.	It generates shortest path from room H to the safe exit. (Node R-H1 to W-16).	['R-H1', 'W-2', 'W-3', 'W-4', 'W-6', 'W-9', 'W-10', 'W-12', 'W-15', 'W-16']	['R-H1', 'W-2', 'W-3', 'W-4', 'W-6', 'W-9', 'W-10', 'W-12', 'W-15', 'W-16']
2.	No zones are compromised.	It generates shortest path from room D to the safe exit. (Node R-D1 to W-16).	['R-D1', 'W-18', 'W-17', 'W-15', 'W-16']	['R-D1', 'W-18', 'W-17', 'W-15', 'W-16']
3.	No zones are compromised.	It generates shortest path from room M2 to the safe exit. (Node R-M2 to W-16).	['R-M2', 'W-10', 'W-12', 'W-15', 'W-16']	['R-M2', 'W-10', 'W-12', 'W-15', 'W-16']
4.	Zone H is compromised.	It generates shortest path from room H to the safe exit. (Node R-H1 to W-16).	It should return Code-01 for rooms having windows. Code-01 is generated when there is no path available, but the room has a window. A user can be rescued through the window.	Returns Code-01
5.	Zone B1 is compromised.	It generates shortest path from location B1 to the safe exit. (Node R-B1 to W-16).	It should generate an alternate short path from B2 to safe exit as below: ['R-B1', 'R-B2', 'W-23', 'W-12', 'W-15', 'W-16']	['R-B1', 'R-B2', 'W-23', 'W-12', 'W-15', 'W-16']
6.	Zone B1 and B2 are compromised.	It generates shortest path from location B1 to the safe exit. (Node R-B1 to W-16).	It should return Code-02 for rooms not having windows. Code-02 is generated when there is no path available, and the room does not have a window. A user cannot be rescued through the window.	Returns Code-02

7.	Zone C1 is compromised.	It generates shortest path from location C1 to the safe exit. (Node R-C1 to W-16).	It should generate alternate path from node C2 to the safe exit as below: ['R-C1', 'R-C2', 'W-14', 'W-15', 'W-16']	['R-C1', 'R-C2', 'W-14', 'W-15', 'W-16']
8.	All zones are compromised.	It generates shortest path from location C1 to the safe exit. (Node R-C1 to W-16).	It should return Code-02.	Returns Code-02.

Table 6. Test cases for Shortest Path functionality
Source: Own work

CHAPTER 7. Performance and Benchmarks

Cost-efficient	Usability	Availability	Security	Throughput
All online services are serverless and on on-demand mode, which means there is no server to maintain and pay only when used. It is also possible to provision resources to match the pattern of use of those services, which can even further reduce the cost.	A user should be able to easily use the application with minimum to no prior experience required. To achieve this, we followed Apple’s user interface guideline.	All online services are operated using serverless approaches, which for DynamoDB is 99.99% and for Lambda is 99.95%, thanks to having multiple instances in distinct locations that if one instance fails, others can take the work.	All user data stored in Amazon DynamoDB is fully encrypted at rest using keys managed by AWS.	Current concurrency limit is 1000 concurrent Lambda function calls, if more is needed, will have to be requested from AWS.
	A user should be able to easily view the entire path to the safe exit in an emergency.		All data needs proper credentials to access, such as AWS IAM policies or API Keys.	
	The user should be able to select alternate available path.			

Table 7. Performance and Benchmarks
Source: Own work

CHAPTER 8. Deployment, Operations, Maintenance

8.1. Deployment

8.1.1. CI & CD for the iOS App and Back End Services

For continuous integration and continuous deployment (CI & CD), three different tools work in sync:

- GitHub - Code repository.
- GitHub Actions - Feature of code repository to support actions before and after a code commit.
- Amplify – Set of tools to automatically create infrastructure as a code to later deploy it.

Firstly, GitHub Actions triggers after the developer makes a new commit. If there are any errors, the team and the developer will receive an email, and the process will stop.

Secondly, if the first step succeeds, Amplify generates a file with the infrastructure as code in a format that AWS CloudFormation accepts. This file contains all the details to deploy AWS API, AWS Cognito, and AWS DynamoDB with different security layers in between each service.

Lastly, AWS CloudFormation receives the template generated by Amplify to test the viability of building the described infrastructure. If the tests pass, CloudFormation proceeds the setup of all cloud services.

8.1.2. Mobile Application Deployment

The mobile application will be installed on the user’s smart device. No other additional tools are required for installing the application.

8.1.3. First Responders Application Deployment

The first responder’s application is a mobile application as well intended for use on Tablet devices by the first responders. This application will have additional features specifically designed for first responders. No additional tools are required for installing this on the Tablet device.

8.2. Operations

AWS CloudWatch

Amazon CloudWatch is a monitoring and observability service which is used for tracking application and services deployed on AWS Cloud. It helps to watch out on the current operations being performed on AWS Cloud.

It helps to track and evaluate instance performance on AWS Cloud in which you can get real-time performance metrices like CPU Utilization, Active Connection counts, read throughput, Write throughput, Network I/O etc.

AWS Lambda

For our backend services we have used serverless application using AWS Lambda. Lambda is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code on a high-availability compute infrastructure and performs all the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring and logging. With Lambda, you can run code for virtually any type of application or backend service.

The key features of Lambda key help you develop Lambda applications that are scalable, secure, and easily extensible.

Security

For our application we have utilized AWS cloud services that let you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations. AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be authenticated (signed in) and authorized (have permissions) to use Lambda resources.

High Availability and Scalability

Lambda runs your function in multiple Availability Zones to ensure that it is available to process events in case of a service interruption in a single zone. When your function receives a request while it's processing a previous request, Lambda launches another instance of your function to handle the increased load. Lambda automatically scales to handle 1,000 concurrent executions per Region, a quota that can be increased if needed.

8.3. Maintenance

The following steps will be included as part of maintaining the mobile application for the user’s and the first responders.

- Run set of test scenario which are designed to test the entire application if the application is working as expected after doing any correction or changes.
- AWS CloudWatch service is used for monitoring for monitoring AWS service which are used in the application. CloudWatch is used to monitor DynamoDB and Lambda usage. Alarm is set on health check of services.
- Bug fixes
- Implementing new functionality

CHAPTER 9. Summary, Conclusions, and Recommendations

9.1. Summary

The Smart Navigation System application serves as a helpful tool to assist first responders during emergency evacuation. First responders rely on communication among the team for onsite updates during an emergency. The application allows the first responders to perform site updates dynamically thus helping to manage the foot traffic more efficiently.

9.2. Conclusions

It is important to rescue and protect people in event of fire or other emergencies. The smart navigation application allows first responders to efficiently navigate the people to a safe area and ensures the safety of the first responders as well. The first responders can reduce the time required to vacant the building and thus increase the possibility of their own safety as well. This application provides visual navigation instructions and map for its users to easily reach to safe exit. It also helps first responders to monitor number of people in the property in real-time. It allows first responders to control different zones of the property so that they can safely evacuate people. This application generates shortest path to the safe exit out of all available paths to ensure the safe and fast rescue. If the shortest path is blocked due to safety reasons, it generates alternate safe path for users else notifies users to wait for first responder’s help. This application can be helpful for first responders and users for indoor navigation and emergency rescue operations.

9.3. Limitations

The below Table 8 describes limitations of this project.

No.	Limitation	Description
1.	Limited support to location	The Smart Navigation Application currently supports only one floor of San Jose Convention Center. In future, we are planning to support all floors of the same premises.
2.	Accessing User location	Due to the COVID19 restrictions set by County of Santa Clara, it was not possible to access the selected location in person. Therefore, instead of implementing this application in actual premises of San Jose Convention Center, simulation has been used for implementation. It was possible to access user location through simulation. However, more issues can be encountered while testing this application in real-life scenario.
3.	Limitation due to hardware capacity	Due to the limitation of computer's hardware capacity, currently this application simulates 5 user locations at a time. If hardware with more power is used, this application can simulate more user locations. However, this does not limit the backend functionality.

4.	Can simulate one user movement at a time	Due to some limitations in Xcode IDE, currently application can simulate a single user movement at a time.
5.	Not used real IoT devices	As it is not possible to access the San Jose Convention Center premises physically, real IoT devices have not been used for this implementation. Rather, IoT device simulator has been used.
6.	User location accuracy in real life scenario	The testing assumes 100% user location accuracy. However, we could not test real-life scenario where location accuracy is lost.
7.	Limited Mobile OS support	At present, this mobile application can only be installed in iPhones and it does not support Android OS.

Table 8. Limitations of project
Source: Own work

9.4. Recommendations for Further Research

Due to the COVID19 restrictions simulation was used in place of real IoT devices. This led to approximating the user’s location when displaying the path on the map to the safe area as the simulation has limits on the accuracy of the object’s position. In future, we recommend implementing this project in real-life scenario.

To access this application, users must manually install this application in their mobile devices. It is mandatory to have this application installed in user’s mobile devices for this application to help users through emergency situations. If this application is natively embedded in the operating system, users do not need to install it manually. In addition, it will create a huge impact in emergency rescue operations for people and first responders.

Emerging technologies like Apple U1 chip allows devices to precisely locating and communicating with other devices equipped with U1 chip or the ones supporting ultra-wideband. U1 chips helps more accurate perception of indoor places not only in 2 dimensional but also in 3 dimensional places. It also includes location detection at multiple levels of premises.

Glossary

Name	Description
IoT Devices	Internet of Things (IoT) devices are wireless sensors and software devices that enables automatic transfer of data among objects that operate on internet.
REST API	An application programming interface conforming the constraints of REST architecture style. It allows interaction with RESTful web services.
AWS	Amazon Web Service (AWS) is subsidiary of Amazon which provides on-demand cloud computing platforms and APIs.
DynamoDB	DynamoDB is a AWS NoSQL database service
AWS Lambda functions	A serverless compute service used to execute code in AWS without provisioning a server.
AWS Amplify	A package of tools and services used to create and launch application in AWS cloud.
AWS AppSync	A robust GraphQL interface used in application development to combine data from DynamoDB, AWS Lambda and REST APIs.
GraphQL	A query language used in API to execute queries using a type system defined for data.
CRUD	Create, Read, Update and Delete
NoSQL	It is a 'not only SQL' database which stores data in documents like JSON objects.
Reinforcement Learning	It is a part of Machine Learning which focuses on algorithms that learn to take an optimal action based on its experience of gaining rewards or failure through many attempts.

TABLE OF REFERENCES

[1] L. A. Office, "Main Types of Disasters and Associated Trends," The California Legislature's Nonpartisan Fiscal and Policy Advisor, 10 Jan 2019. [Online]. Available: https://lao.ca.gov/Publications/Report/3918.

[2] S. C. C. F. Dept, "Emergency Preparedness," 2020. [Online]. Available: https://www.sccfd.org/education-and-preparedness-overview/emergency-preparedness.

[3] S. C. C. P. H. Dept, "Healthy and Safe Environment Statistics - Zip Code," Santa Clara County Public Health Dept, 22 May 2019. [Online]. Available: https://data-sccphd.opendata.arcgis.com/datasets/healthy-safe-environment-statistics-zip-code/data?page=2.

[4] "National Institute of Standards and Technology," 14 April 2015. [Online]. Available: https://www.nist.gov/ctl/pscr/about-pscr. [Accessed 25 September 2020].

[5] "National Institute of Standards and Technology," 12 July 2016. [Online]. Available: https://www.nist.gov/ctl/pscr/research-portfolios. [Accessed 25 September 2020].

[6] "National Institute of Standards and Technology," 20 September 2017. [Online]. Available: https://www.nist.gov/ctl/pscr/research-portfolios/location-based-services. [Accessed 25 September 2020].

[7] R. Felts, M. Leh, D. Orr and T. A. McElvaney, *Location-Based Services R&D Roadmap,* Gaithersburg: National Institute of Standards and Technology Technical Note 1883, 2015.

[8] "ARCore overview," Google Inc., 28 Feb 2019. [Online]. Available: https://developers.google.com/ar/discover. [Accessed 25 Sep 2020].

[9] S. P. W. Chathurika S, "State-of-Art-in-Indoor Navigation and Positioning of Visually Impaired and Blind," in *2017 International Conference on Advances in ICT for Emerging Regions (ICTer)*, 2017.

[10] A. H. K. M. S. B. Youssef N. Nagger, "A Low Cost Indoor Positioning System Using Computer Vision," in *I. J. Image Graphics and Signal Processing*, 2019.

[11] V. A. Ugave, "Smart Indoor Navigation Using Machine Learning Techniques," Viney Anand Ugave, 2014.

[12] E. Diaz, "Inertial pocket navigation system: Unaided 3D positioning," in *Sensors 2015*, 2015.

[13] J. W. B. Z. W. Y. Z. Xu, "A robust method to detect zero velocity for improved 3D personal navigation using inertial sensors," in *Sensors 2015*, 2015.

[14] R. P. P. Davidson, "A Survey of Selected Indoor Positioning Methods for Smartphones," in *IEEE Communications Surveys & Tutorials*, 2017.

[15] R. Geisberger, P. Sanders, D. Schultes and C. Vetter, "Exact Routing in Large Road Networks Using Contraction Hierarchies," *Transportation Science,* vol. 46, no. 3, pp. 388-404, 2012.

[16] "Icons," [Online]. Available: https://www.flaticon.com/authors/monkik.

[17] H. Butler, M. Daly, A. Doyle, S. Gillies, S. Hagen and T. Schaub, "The GeoJSON Format," Internet Engineering Task Force (IETF), August 2016. [Online]. Available: https://tools.ietf.org/html/rfc7946. [Accessed 26 September 2020].

[18] P. Hudson, "Control Room," Bath, UK.

[19] "Official U.S. government information about the Global Positioning System (GPS) and related topics," 30 January 2020. [Online]. Available: https://www.gps.gov/systems/gnss/. [Accessed 25 September 2020].

[20] "Official U.S. government information about the Global Positioning System (GPS) and related topics," 22 April 2020. [Online]. Available: https://www.gps.gov/systems/gps/. [Accessed 25 September 2020].

[21] "Official U.S. government information about the Global Positioning System (GPS) and related topics," 12 July 2016. [Online]. Available: https://www.gpsworld.com/terrestrial-beacons-bring-wide-area-location-indoors/. [Accessed 25 September 2020].

[22] A. H. K. M. S. B. Youssef N. Nagger, "A Low Cost Indoor Positioning System Using Computer Vision," in *I. J. Image Graphics and Signal Processing*, 2019.

[23] "Flaticon Icons," [Online]. Available: https://www.flaticon.com/authors/monkik.

Appendices

Appendix A. Description of project implementation repository

All code and documentation related to this project are maintained on below GitHub repositories.

1.First responder’s app: https://github.com/roanacla/iPadOS_Emergency_Indoor_Nav

2. User’s app: https://github.com/roanacla/iOS_Emergency_Indoor_Nav

3. Machine learning model: https://github.com/snarvekark/ML_Emergency_Indoor_Nav

4. Backend APIs: https://github.com/Wayne122/AWS_Emergency_Indoor_Nav