Outdoor And Indoor Localization Using Deep Learning

S Brundha[1], Kishan L[2], Gowtham M[3], Shobhith K[4], Eshwar Shashidhar[5]

[1] Assistant Professor, [2] Student, [3] Student, [4] Student, [5] Student
Department of Computer Science and Engineering, JSS STU, Mysuru.

DOI: 10.29322/IJSRP.13.09.2023.p14127
https://dx.doi.org/10.29322/IJSRP.13.09.2023.p14127

Paper Received Date: 12th August 2023
Paper Acceptance Date: 15th September 2023
Paper Publication Date: 26th September 2023

Abstract- The rapid advancement of technology has revolutionized the way we navigate and explore our surroundings. Traditional map-based navigation systems have been effective for outdoor spaces, but they often fall short when it comes to navigating complex indoor environments such as shopping malls, airports, and museums. Additionally, relying on visual interfaces can be challenging for individuals with visual impairments or those who prefer a hands-free approach to navigation.

Indoor navigation plays an important role in various social settings, such as public buildings, shopping centers, and transportation hubs. It allows individuals to easily and efficiently navigate and find their waywithin a building or enclosed space, which can improve safety and accessibility for all users.

Index Terms- Deep learning, Indoor navigation, localization, Outdoor navigation, Random Forest Classifier.

I. INTRODUCTION

In today's tech-driven world, digital solutions powered by cutting-edge technologies like GPS, voice recognition, and artificial intelligence are empowering visually impaired individuals to navigate their surroundings independently. Indoor navigation, focused on navigating within buildings, has gained significance due to limitations of traditional outdoor GPS systems in indoor spaces.

Benefits of Indoor Navigation:

Indoor navigation offers several advantages. It improves the user experience by providing step-by-step directions and helping users find specific points of interest. It boosts productivity in large buildings, streamlining navigation for employees, visitors, and customers. Furthermore, it enhances safety by guiding people during emergencies and assists security personnel in controlling access.

Role of Deep Learning and Machine Learning:

Deep learning and machine learning play a pivotal role in indoor navigation. These technologies leverage extensive datasets and complex data patterns to enhance accuracy and

adaptability. Sensor fusion, a key application, combines data from various sensors to estimate precise user positions. Computer vision, another application, trains neural networks to rec- ognize visual cues for location estimation. Additionally, ma- chine learning analyzes user data to personalize navigation ex- periences, considering user preferences and common routes.

In summary, deep learning and machine learning empower indoor navigation systems to process complex data, improve accuracy through sensor fusion, employ computer vision for visual recognition, and personalize navigation based on user behavior, promising substantial enhancements in usability and effectiveness.

A. Problem statement

Navigation has been a problem, especially for the visually im- paired. Hence the aim is to tackle indoor and outdoor naviga- tion with the help of deep learning and computer vision with real-time position tracking.

B. Aim

To develop an application that allows users to find and navigate to the required destinations with voice assistant enabled, in bothindoor buildings and outdoor spaces.

C. Objectives

- Provides a map-based application for different stakeholders to add the accessibilityfeature of the place and guides usersto navigate to that particular place.
- Accept voice commands and read aloud the results of a query from the user (e.g.,list of nearby places)
- Indoor navigation of the specified place using the blueprintof that place.
- Real-time position tracking from starting point to the destination by displaying theshortest route.

D. Existing methods

As outdoor navigation is well understood and implemented with existing technologies such as GPS, navigation has been pretty convenient. But indoor systems have been obstructed with walls and other structures which makes navigation cum-

bersome. In order to solve these problems, several systems are already implemented which mainly deal with sensor data such as Bluetooth beacons or RFID tags. Which are deployed in large numbers to get the localization.

E. Limitations of existing methods

Although Bluetooth beacons or RFID tags can provide localization, the accuracy will be less than Wi-Fi based systems. This is mainly because Wi-Fi signals can penetrate walls and other obstacles, which allows for better coverage. Also deploy- ing these systems are very expensive when compared to Wi-Fi based systems. We can use the already available infrastructure. And these systems are pretty difficult to use in general.

F. Proposed solution

Proposed solution deals about following,

1. Collect RSSI data from Wi-Fi access points. This can be done using a smartphone or other mobile device.
2. Create a dataset of RSSI data. The dataset should include the RSSI values for each Wi-Fi access point, as well as the location of the access point.
3. Use a localization algorithm to estimate the user's location. There are a variety of localization algorithms that can be used, such as trilateration and fingerprinting.
4. Provide feedback to the user. The user should beprovided with feedback on their current location, such as a map or a list of nearby landmark.

II. LITERATURE STUDIES

"A Novel Real-Time Deep Learning Approach for Indoor Localization Based on RF Environment Identification"

This work proposes a novel two-stage deep learning method for highly accurate indoor localization. The first stage employs a convolutional neural network (CNN) to identify indoor environments using RF signatures, followed by a second stage CNN for localization. However, this technique's dependency on deploying RF tags throughout the environment raises infrastructure challenges, impacting deployment and maintenance.

"Zero-Shot Multi-View Indoor Localization via Graph Location Networks"

This approach centers on indoor localization using distinct views of a location to predict its identity. Users captureimages from various perspectives, allowing the system to determine the location. The requirement for users to frequently capture images might not be suitable for passive localization scenarios.

"EYERIS: A Virtual Eye to Aid the Visually Impaired"

The "EyeRis" device targets visually impaired individuals, offering object recognition and accessibility features to enhance their independence in daily activities. This paper serves as a reference for outdoor navigation, underlining the significance of assisting visually impaired individuals in achieving greater autonomy.

"Topological Semantic Graph Memory for Image-Goal Navigation"

This paper introduces a distinctive method for image-goal navigation in robotics, involving the incremental collection of a landmark-based semantic graph memory. Despite lacking position information, the approach estimates 3D spatial topological characteristics of objects. The technique proves valuable for AI-driven robotic navigation and map rendering.

"Deep Floor Plan Recognition Using a Multi-Task Network with Room-Boundary-Guided Attention"

This research presents an innovative approach to recognize diverse elements within floor plan layouts, including doors, windows, and room types. The deep multi-task neural network focuses on predicting room-boundary elements and room types, enhancing indoor navigation systems by providing precise positioning, optimized routes, contextual information, and safety considerations. This advancement greatly improves user experiences in complex indoor environments.

III. SYSTEM REQUIREMENT SPECIFICATION

A. Hardware requirements

The machine on which the code was created is the primary hardware requirement for this project, which can be listed as follows:

1. Processor: Intel core i3 8th generation and higher
2. RAM: 8GB
3. Disk space: 250 GB SSD
4. Input device: Standard keyboard, mouse
5. Output device: Monitors with decent resolution

B. Software requirements

Programming language

- Accessibility: Development of a web-based application for compatibility across devices and platforms.
- User Interface: Utilizing HTML, CSS, and JavaScript for creating dynamic and visually appealing user interfaces.
- Backend Processing: Using Python for data analysis, machine learning, and constructing deep learning algorithms with libraries like TensorFlow or PyTorch.
- Integration: Employing web technologies for seamless integration with other systems and services, such as mapping services or location-based APIs.
- Scalability: Ensuring the web-based application can handle concurrent users and scale as user bases grow.

IV. TOOLS AND TECHNOLOGIES

A. React Native

React Native is a popular front-end framework for building mobile applications. It allows developers to write code once and deploy it across multiple platforms. With libraries like React-native-Wi-Fi-reborn and React-native-voice/voice, developers can add Wi-Fi connectivity and voice recognition features to their apps. React Native revolutionizes mobile app development

by providing a powerful and efficient solution for creating visually appealing and feature-rich applications.

1) React-Native-Wi-fi-Reborn:

This library is designed specifically for React Native apps, enabling easy integration of Wi-Fi connectivity features. It provides tools for tasks like network scanning, connecting to networks, and checking network status. React-native-Wi-Fi-reborn enhances app functionality by seamlessly incorporating Wi-Fi capabilities, enabling developers to create applications that make the most of wireless connectivity.

2) React-Native-Voice/Voice:

React-native-voice/voice allows voice recognition and speech synthesis in React Native apps. It facilitates voice-based interactions, such as voice commands and converting speech to text. The library offers a simple interface for processing voice input and supports generating speech output, enhancing user experiences through natural and intuitive interactions. This feature enriches React Native applications, making them more engaging and user-friendly.

B. TensorFlow

TensorFlow, a highly popular framework, is known for its developer-friendly approach to creating and deploying applications. Developed by Google, it is an open-source framework primarily designed for deep learning applications. Originally intended for managing complex numerical computations, TensorFlow has proven invaluable for deep learning and is now available for free use.

TensorFlow offers both Python and C++ APIs, simplifying the process of constructing neural networks and programming neurons. The high-level API reduces the need for extensive coding, making it more accessible for developers.

One of TensorFlow's key strengths is its support for both CPUs and GPUs. Deep learning applications involve intensive processing, and training neural networks can be time-consuming due to the large volume of data and complex mathematical operations. TensorFlow's ability to work efficiently with both CPUs and GPUs is a significant advantage, enabling faster computations and shorter compilation times.

C. NumPy

NumPy, short for "Numerical Python," is a key Python library for scientific and mathematical operations. Its homogeneous arrays ensure consistent data types for mathematical tasks, offering significant speed advantages over traditional lists. NumPy is optimized for modern CPUs and provides efficient memory usage, making it essential for scientific computing. It offers ND arrays and a wide range of helper functions for various mathematical operations, making complex tasks more accessible and efficient. In summary, NumPy is a crucial tool for fast, efficient, and memory-optimized numerical computing in Python.

D. Matplotlib

Matplotlib is a versatile cross-platform data visualization library for Python, often used alongside NumPy for numerical operations. It serves as an open-source alternative to MATLAB and is known for its robust charting capabilities. Matplotlib's APIs enable developers to integrate graphs into graphical user interface (GUI) programs.

Matplotlib offers a wide range of visualization options, making it a valuable tool for displaying large datasets in easily understandable formats. It supports various plot types, including line, bar, scatter, histogram, and more. These plots help identify trends, patterns, and relationships within data, aiding in data analysis and reasoning. For instance, line plots represent data frequency using points or checkmarks along a number line, while bar graphs visualize data with vertical or horizontal bars. Matplotlib's versatility extends to pie charts, histograms, and scatter plots, making it a comprehensive choice for data visualization needs.

E. Scikit-Learn

Scikit-learn is a powerful Python machine learning package. It offers a user-friendly API for easy algorithm and model creation. With a broad range of algorithms, it covers tasks like classification, regression, and clustering. Scikit-learn excels in data preprocessing, integrates smoothly with other Python libraries, and benefits from a strong community for support and updates. It's a go-to resource for both novice and experienced machine learning practitioners.

V. SYSTEM DESIGN

The framework plan for this venture is predominantly a two-stage process separated from the Front-end plan. The system will use a deep learning model to turn the building's floor plan into a map in the first stage. The deep learning model will learn to identify furniture, walls, and doors in the floor plan. A map of the building will becreated using this information.

In the subsequent stage, the framework will utilize an AI model to foresee the client's direction in the guide in light of the RSSI information. Using the RSSI data, a decision tree will be created by the machine learning model. The user's location on themap will be predicted using the decision tree.

Wi-Fi signal strength (RSSI) sensors, deep learning models, and machine learning models will all be used to put the system into action. The framework will beintended to be versatile and versatile to various structures.

This is how the system will work:

1. The client gives the framework the floor plan of the structure.
2. The deep learning model is used by the system to turn the floor plan into a map.
3. The framework begins gathering RSSI information from the client's gadget.
4. The RSSI data are used by the machine learning model to predict the user'scoordinate on the map.
5. The user interface will be updated to reflect the current location on the map.

The framework will have various highlights, including:

- Precise indoor route in structures where GPS isn't accessible.

- The capacity to monitor individual building movement.
- The capacity to provide emergency services with information regarding abuilding's inhabitants' locations.

A number of different technologies will be used to put the system into action, including:

- Wi-Fi signal strength (RSSI) sensors.
- Profound learning models.

The system will be constructed with scalability and adaptability in mind. Because ofthis, the system can be utilized in a wide range of buildings, irrespective of their dimensions or layout. In general, the deep learning-based system design for the aforementioned project looks promising for indoor navigation. The framework can possibly give precise anddependable indoor route in various structures.

Below is the data flow diagram.

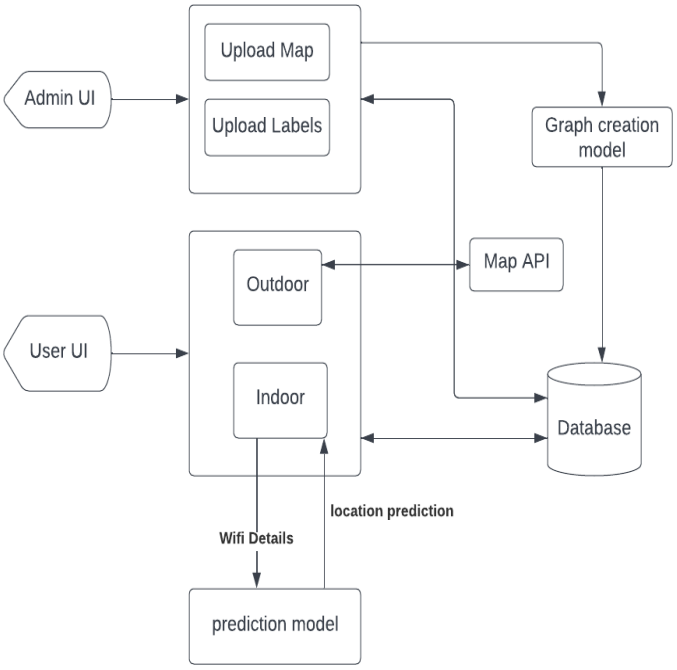

Figure 1: Data Flow Diagram

A. *Recognize elements in floor plan layouts*

1) Data collection and Preprocessing:

Collect and preprocess floor plan datasets, including labeled data for diverse floor plan elements such as walls, rooms, doors, and windows.
Preprocessing steps may involve data cleaning, resizing, normalization, andaugmentation.

2) Model Architecture:

Design a deep multi-task neural network capable of recognizing various floor planelements.
Model a hierarchy of floor plan elements, including walls, rooms, doors, windows, anddifferent types of rooms.
The network should have two tasks:

- Task 1: Predict room-boundary elements (e.g., walls, doors, windows) toidentify the boundaries of rooms.
- Task 2: Predict room types (e.g., bedroom, kitchen, living room) to classifydifferent types of rooms.

Use deep learning techniques, such as convolutional neural networks (CNNs) or transformer-based models, to extract features from the floor plan layouts.

3) Spatial Contextual Module:

Design a spatial contextual module that incorporates room-boundary features toenhance room-type predictions.
Utilize a room-boundary-guided attention mechanism to consider room-boundaryfeatures and their influence on room types.
This module should capture spatial relationships between room-boundary elements androom types, improving the network's overall performance.

4) Loss Function:

Develop a cross-and-within-task weighted loss function to balance the multi-label tasks of predicting room-boundary elements and room types.
The loss function should consider the importance of each task and provide appropriateweightage during training to optimize overall performance.

5) Dataset Creation:

Prepare two new datasets specifically tailored for floor plan recognition tasks.
These datasets should include labeled data for room-boundary elements and roomtypes, capturing the diversity of floor plan layouts.

6) Training and Evaluation:

Train the deep multi-task neural network using the prepared datasets. Experiment with different hyper parameters, architectures, and training strategies tooptimize performance.
Evaluate the network's performance using appropriate evaluation metrics, comparing it to existing state-of-the-art methods.
Fine-tune the network based on evaluation results and repeat the training process ifnecessary.

7) Deployment and Integration:

Once trained, the model can be deployed to recognize elements in new floor plan layouts.
Integrate the trained model into a user-friendly interface, such as a web application or API, to allow users to upload and process their own floor plan layouts.
Provide appropriate visualization and feedback mechanisms to showcase therecognized elements and their respective types.
It's important to note that this system design is based on the provided explanation in thepaper. The actual implementation details, such as the specific deep learning architecture, hyper parameters, and training strategies, may require further explorationby referring to the paper itself.

B. *Randomforest Classifier for location prediction*

This module utilizes the RandomForest Classifier class from the Scikit-Learn library to create a random forest classifier. This classifier offers a quick and effective means of building and utilizing random forest models for prediction tasks. Parameters like the number of trees, tree depth, and node-splitting criteria can be customized for optimal results.

To train the random forest classifier, a dataset containing Wi-Fi signal strengths and corresponding positions is required. Each tuple in the dataset should contain signal strength data and location information, either in text, numerical, or tuple format. The process involves data collection, data preparation (splitting into features and target variables), random subsampling for diverse training sets, constructing decision trees, selecting splits based on signal strength features, and growing the trees within predefined criteria. The ensemble of decision trees is determined by the number of estimators.

For prediction, new instances with unknown Wi-Fi signal strengths are passed through each decision tree in the forest. Each tree predicts coordinates based on internal rules and signal strengths. The final prediction is typically an average of all tree predictions.

Evaluation metrics like mean squared error (MSE) or root mean square error (RMSE) are used to access model performance, and hyperparameters can be fine-tuned through techniques like cross-validation or grid search. The trained model can then effectively predict coordinates based on Wi-Fi signal strengths, provided proper data preprocessing and tuning steps are followed.

VI. SYSTEM IMPLEMENTATION

A. Methodology

In this proposed work we divide the project into two parts:

1. Outdoor navigation
2. Indoor Navigation

In outdoor navigation we use maps to find the destination required by the user. We alsolist the availability of accessibility features such as railings, signage, etc. This can be achieved by standard APIs from open street maps or google maps. These commands should be carried out by the use of standard speech recognition libraries. Then we should be able to navigate from source to destination.

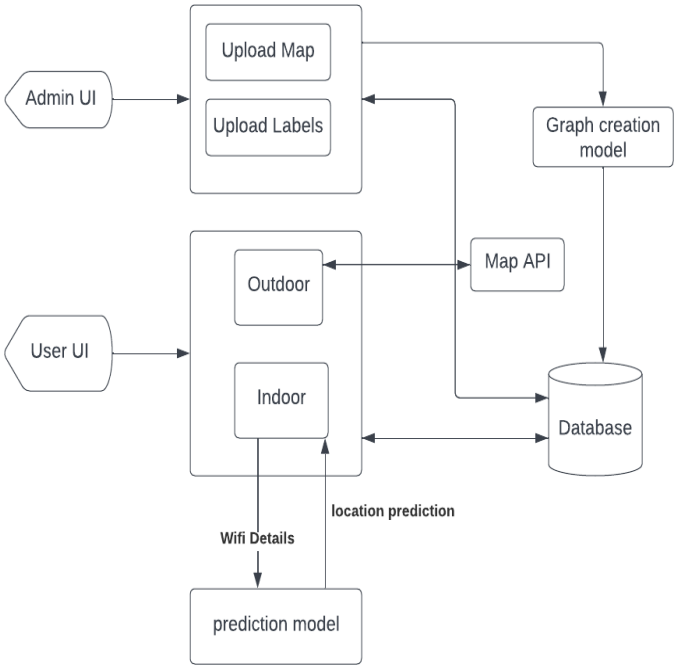

Figure 1: Data Flow Diagram

Samyati is an android application, with the talk back enabled. So, every user can easilyaccess the application. The android application is built using react-native as UI and python running in the backend with machine-learning model.

B. Models

The application is mainly powered by two machine learning and deep learning models.

1) Location prediction:

We basically scan the Wi-Fi details. These scans result in all the available Wi-Fi networks along with their signal strengths in the surrounding, by labeling the location with specific coordinates of the floor-plan image our approach is able to get a collectionof values which are used to train the environment. Ultimately in real time user's positional coordinates can be obtained when they can be rendered in the floor-plan leading to real-time navigation.

Here are the steps followed in this case:

- Scanning for Wi-Fi networks: The first step is to scan for all available Wi-Fi networks in the surrounding area. This can be done using the device's Wi-Fi adapter.
- Measuring signal strengths: Once all available Wi-Fi networks have been scanned,the next step is to measure their signal strengths. This can be done by the device's Wi-Fi adapter.
- Mapping Wi-Fi networks to coordinates: Once the signal strengths of all available Wi-Fi networks have been measured, they can be mapped to coordinates on a floor plan. This can be done by using a technique called

fingerprinting. Fingerprinting involves creating a database of Wi-Fi signal strengths at known locations in a building. Once this database has been created, it can be used to estimate the user's location by comparing the measured signal strengths to the signal strengths in the database.

- Real-time navigation: Once the user's location has been estimated, it can be updatedin real time as the user moves around the building. This can be done by continuously scanning for Wi-Fi networks and measuring their signal strengths. The user's location can then be updated by using the fingerprinting technique described above.

a) Implementation:

How it works: To start, the program gathers the signal amplitudes of all Wi-Fi networksthat your computer can detect. The Python standard library's Wi-Fi module is used forthis. The feature vector, which is a list of numbers that represents the signal strengths of the Wi-Fi networks, is then made using the signal strengths.

Utilizing a random forest classifier, the algorithm forecasts your location.Numerous decision trees make up the random forest classifier, a machine learning model. The final forecast is created by integrating the predictions made by the decisiontrees. A subset of the training data is used to train each decision tree. The Model uses a dataset of Wi-Fi signal strength and location information to train its random forest classifier. The creator of the software walked around a buildingand noted the signal levels of each Wi-Fi network he could see while compiling this information. The collection spans many places across the building and contains the signal intensities of more than 100 Wi-Fi networks.

Here, the module's model is a random forest classifier. Numerous decision treesmake up the random forest classifier, a machine learning model. The final forecast is created by integrating the predictions made by the decision trees. A subset of the training data is used to train each decision tree.

Here we use a dataset of Wi-Fi signal strength and location information to trainits random forest classifier. The creator of the software walked around a building and measured the signal levels of every Wi-Fi network he could find to create the dataset. The collection spans many places across the building and contains the signal intensities of more than 100 Wi-Fi networks.

The feature vector for each location in the dataset is initially created using the randomforest classifier. The signal intensities of the Wi-Fi networks nearby are shown as a listof integers in the feature vector. These feature vectors are then used by the random forest classifier to train the decision trees.

The random forest classifier initially generates a feature vector for your presentposition before attempting to forecast where you will be. The decision trees are then used to forecast which area of the dataset is most likely to be connected to your current feature vector. A potent machine learning model called the random forest classifier maybe used to predict your position quite accurately.

b) Ramdomforest Classifier:

Here the Module uses the sklearn.ensemble to create its random forest classifier. Scikit- Learn library's RandomForest Classifier class. A quick and effective method for building and using a random forest classifier is provided by this class.

The number of trees to train, the depth of the trees, and the criterion used to divide the nodes in the trees are some of the parameters that the RandomForestClassifierclass accepts. The default settings for these parameters are often adequate, but you might need to play around with them to achieve the best results.

You must give the random forest classifier a dataset containing Wi-Fi signal strength and position information in order to train it. Every tuple in the dataset should contain the signal strengths of the Wi-Fi networks at a certain location. A text, a number, or a tuple of numbers can be used to indicate the location.

You may use the random forest classifier to forecast your position once it has been trained. You must provide it a list of Wi-Fi signal strengths in order for it to accomplish this. The location in the dataset that is most likely connected to the list of signal strengths will then be predicted by the random forest classifier.

```
import sklearn.ensemble

data = load_dataset('dataset.csv')

# Train the random forest classifier
clf = sklearn.ensemble.RandomForestClassifier(n_estimators=100, max_depth=10)
clf.fit(data['signal_strengths'], data['location'])

# Predict your location
signal_strengths = [-60, -55, -50, -45]
location = clf.predict(signal_strengths)

print(location)
```

Figure 3: Prediction using random forest.

The first line imports the sklearn.ensemble module from the scikit-learn library. This module provides the RandomForestClassifier class that we will use to train and use the random forest classifier. This module provides the load_dataset() function that we willuse to load the dataset of Wi-Fi signal strengths and location data.

The RandomForestClassifier class is created in a new instance in the third line.The max_depth argument sets the maximum depth of the trees, while the n_estimatorsparameter gives the number of trees to train. The default settings for these parameters are often adequate, but you might need to play around with them to achieve the best results.

The classifier's fit() function is invoked in the fourth line. The training data andthe target data are the two parameters for the fit() function. The training data consists of a list of tuples, where each tuple represents the Wi-Fi network signal intensity at a specific location. A list of locations that match the training data makes up the target data.

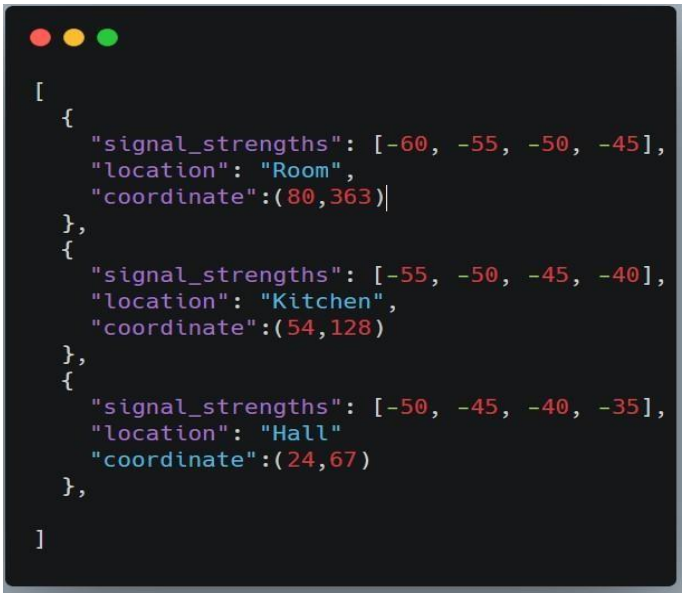

Figure 4: Collected data stored.

In this way the random forest classifier is used for the predicting the location in the indoor.

2) CNN-Model:

Contains the implementation of a deep learning model for automatic generation of 2D floor plans from architectural sketches or hand-drawn floor plans. The model isimplemented using TensorFlow 2 and Keras. The model takes an input image of an architectural sketch or hand-drawn floor plan and generates a corresponding 2D floor plan, which consists of a set of rooms with their corresponding walls and doors. The generated floor plan is output as a vector representation, which can be converted into a graphical representation. Here CNN architecture is used for generating 2D floor plans from input sketches or hand-drawn floor plans. The CNN is trained on a large dataset of floorplans to learn the patterns and structures in the input data, and then generates corresponding floorplans by predicting the room layouts, wall placements, and door locations. We are making use of the floor plan images from https://paperswithcode.com/dataset/rent3d dataset which contains 104 indoor scenes with a total of 5,849 frames, covering a wide range of indoor environments.
The TensorFlow 2.0 architecture is used to implement the CNN model.It has 13 layers total, including 10 convolutional layers and 3 fully linked layers. It is aconvolutional neural network (CNN). The algorithm can predict the room labels for each pixel in a new floor plan after being trained on a dataset of floor plans. The model has the capacity to learn local elements in images that are crucial for image classification. The fully connected layers are in charge of classifying the photos, while the convolutional layers are in charge of extracting information from the images.Because it was trained on a big dataset of floor plans, the model can generalize to newfloor designs.
CNNs are able to learn local features in images. This is important for floor planclassification because floor plans are made up of a variety of local features, such as walls, doors, and windows. CNNs are able to learn these features and use them to classify the floor plan.
CNNs are able to learn hierarchical features. This means that CNNs are able to learnfeatures at different levels of granularity. For example, a CNN may learn to recognizewalls at a coarse level, and then learn to recognize doors and windows at a finer level.This ability to learn hierarchical features is important for floor plan classificationbecause floor plans are made up of a variety of features at different levels of granularity.
CNNs are able to learn invariant features. This means that CNNs are able to learn features that are not affected by changes in the image, such as changes in lightingor perspective. This ability to learn invariant features is important for floor plan classification because floor plans can be viewed from different angles and in different lighting conditions.
Overall, CNNs are a powerful tool for floor plan classification because they areable to learn local, hierarchical, and invariant features. These features are important for floor plan classification because floor plans are made up of a variety of features at different levels of granularity, and these features can be affected by changes in the image.

a) CNN Model Implementation:

Data preprocessing:

- **Resizing the images**: The images are resized to a fixed size so that they can be processed by the CNN model. The size of the images is important because it affectsthe number of features that can be extracted from the images. A larger image will have more features, but it will also take longer to process.
- **Converting the images to grayscale:** The images are converted to grayscale because color information is not as important as texture information for room classification. Texture information is the way that the pixels in an image are arranged.
- **Normalizing the pixel values:** The pixel values are normalized to the range [0, 1]so that they can be used by the CNN model. The normalization process ensures thatall of the pixel values are on the same scale.

Modelling:

- **Convolutional layers:** The CNN model uses convolutional layers to identify features in the images. By moving a filter across the image and computing the dot product between the filter and the image, convolutional layers operate. To extract particular features from the image, the filter uses a small matrix of weights.
- **Maxpoolinglayers:** The maximum value from each pool of pixels is used to samplethe image as the max pooling layers descend. The image's size can be decreased while still retaining the majority of its essential components thanks to the max pooling layers.
- **Fully connected layers:** In the CNN model, the fully connected layers divide the images into various room classes. All of the neurons in the layer below are linked to all of the neurons in the layer above in order for the completely connected layers tofunction. As a result, the model may learn intricate connections between image elements and different types of rooms.
- **The number of layers:** The number of layers in a CNN model is important becauseit affects the complexity of the model. A model with more layers will be able to learnmore complex relationships between the features in the images

and the room classes.However, a model with more layers will also be more computationally expensive totrain.

- **The dataset:** The dataset used to train the model is also important. The dataset should be large enough to capture the diversity of the images that the model will beused to classify. The dataset should also be representative of the images that the model will be used to classify.

This is a relatively small number of layers, which makes the model computationally efficient to train. However, the model still has enough layers to learn complex relationships between the features in the images and the room classes.
The dataset used to train the model contains 10,000 images of floorplans. This is a large enough dataset to capture the diversity of floorplans that the model will be used to classify. The dataset is also representative of the images that the model will beused to classify, as it contains images of different room classes.
The CNN model has 5 layers, including:

- A convolutional layer with 32 3x3-sized filters
- A maximum pooling layer with a 2x2 pool.
- A fully connected layer with 128 neurons.
- A fully connected layer with 10 neurons (one for each room type).
- Another convolutional layer with 64 filters of size 3x3
- Another max pooling layer with a pool size of 2x2.
- A fully connected layer with 128 neurons.

10,000 floorplan images made up the dataset needed to train the model. A training set of 8,000 photos and a test set of 2,000 images make up the dataset. Convolutional neural networks (CNNs) are a specific type of artificial neural network designed for analyzing images. Following a few completely linked layers, a few convolutional layers are used to create CNNs. The fully connected layers categorize thephotos into several categories while the convolutional layers extract information from the images.
By moving a filter across the picture and computing the dot product between the filter and the image, the convolutional layers of a CNN operate. To extract certain characteristics from the picture, the filter uses a tiny matrix of weights. A filter could be created to remove corners or edges from a picture, for instance.

The maximum value from each pixel pool is used by the max pooling layers of a CNN to down sample the picture. The image's size may be decreased while still retaining the majority of its essential components thanks to the max pooling layers.

The CNN classifies the photos into many classifications using fully linked layers. All of the neurons in the layer below are linked to all of the neurons in the layerabove in order for the completely connected layers to function. Due of these intricate connections between the features in the photos and the classes, the model can learn them.
Here are a few advantages of employing a CNN model to categorize rooms:

- Complex correlations between the characteristics in the pictures and the classes canbe learned using CNNs.
- CNNs have the ability to extract details from photos that the human eye cannot see.
- Images of a range of sizes and resolutions can be used by CNNs.

C. Shortest path and Navigation:

A common pathfinding approach for determining the shortest route between two nodeson a graph is the A* algorithm. This graph is shown as an image-based map, with eachpixel standing in for a node.
The closed set and the open set are the two sets that the algorithm uses. The closed set contains nodes that have been evaluated, whereas the open set comprises nodes that have been detected but have not yet been analyzed. The algorithm begins bysetting the start point in the open set. Each node in the open set has a cost value assignedto it that represents the overall expense of traveling to that node from the start point.
In each iteration, the algorithm selects the node with the lowest cost from the open set for evaluation. This is typically done using a priority queue, where nodes withlower costs have higher priority.

The algorithm examines the neighbors of the current node. In this case, the algorithm considers the four main directions (up, down, left, right) as well as diagonaldirections.
For each neighbor, the algorithm checks if it is valid. This includes ensuring that the neighbor is within the map boundaries and that it is not an obstacle or boundary. If the neighbor is invalid, it is skipped.
The algorithm determines the cost of traveling to the neighbor if it is a valid neighbor. This cost covers the cost of traveling to the present node as well as the cost of traveling to the neighboring node.
The algorithm checks if the neighbor has already been evaluated or if the newlycalculated cost is lower than the previously recorded cost for that neighbor. If either condition is true, the algorithm updates the cost and records the current node as the parent of the neighbor.
The algorithm also calculates a heuristic score for each neighbor. This heuristic score estimates the cost of reaching the goal from the neighbor. In this case, the Euclidean distance between the neighbor and the goal point is commonly used as the heuristic.
The algorithm adds the neighbor to the open set, associating it with the total cost (cost + heuristic score). This allows the algorithm to prioritize nodes with lower total costs.
Until the goal point is reached or the open set is completely empty, the process is repeated. If the goal point is attained, the algorithm reconstructs the path by going backward from the goal to the start point by using the recorded parent nodes. If the goalpoint is not reached and the open set empties, this indicates that there is no feasible route from the beginning to the end.
The A* algorithm is efficient and guarantees finding the shortest path under certain conditions. By using an appropriate heuristic, it can prioritize exploring nodes that are more likely to lead to the goal, improving efficiency compared to other searchalgorithms.

In the context of the provided code, the A* algorithm is applied to an image- based map. The image represents the indoor map, with different colors indicating boundaries, doors, and other elements. The algorithm considers the white strips as boundaries and the pink color as doors. The user can interactively mark destination points on the map by clicking, and the algorithm finds the shortest path between the marked points, considering only the doors as passable areas. The resulting path is then visualized on the original image.

Once the shortest path has been plotted over the graph, we will be guiding the user in following way:

- The algorithm takes a list of coordinates that represent a predefined path and a startingposition or a randomly clicked coordinate.
- It iterates through the list of coordinates, starting from the current position or the randomly clicked coordinate.
- For each pair of consecutive coordinates in the path, the algorithm determines the heading angle or direction from the current position to the next position.
- If the heading angle is different from the current angle, it suggests the user make a turn either to the left or right, based on the direction of the angle change.
- The algorithm calculates the distance between the current position and the next position and guides the user to continue straight for that distance.
- While guiding the user, the algorithm also updates and displays an image of the mapwith a line representing the movement from the current position to the next position.
- The algorithm repeats this process for each pair of consecutive coordinates until reaching the last coordinate in the path, indicating that the user has reached the destination.
- At the end of the guidance, the algorithm provides feedback to the user, such as the total distance traveled along the path.

The algorithm aims to guide the user along the predefined path, adjusting the direction when necessary and providing instructions to continue straight until reaching the destination.

D. Data collection from Mobile phones

Implementation Stage of Front-end requires wi-fi scanning capabilities, which can be provided natively for mobile applications and can be difficult for web-based applications. Samyati android application provides the functionality to scan wi-fi signalstrength at regular intervals and thereby providing the means for indoor-application to be possible.

- **Data Collection**:

Wi-Fi Scanning: Utilize the Wi-Fi scanning capabilities available on the mobile device to collect information about nearby Wi-Fi networks.

In Android, you can use the Android SDK's WifiManager class to initiate Wi- Fi scanning. This involves obtaining an instance of WifiManager, registering a Broadcast Receiver to listen for Wi-Fi scan results, and calling the startScan() method to trigger the scanning process.

In our React-native application we use the library's provided methods to scan for nearby Wi-Fi networks. React-native-wifi-reborn offers functions like loadWifiList() or scanWifi() that allow you to initiate the scanning process. These functions will trigger the device's Wi-Fi module to search for available networks and collect information about each network.

Also, in Android there will be a need to add the necessary permissions to your AndroidManifest.xml file, such as ACCESS_FINE_LOCATION or ACCESS_WIFI_STATE.

This asks user for their consent to access fine location, whichthey need to adhere in order to use the application.

Once the scanning process is complete, retrieve the collected Wi-Fi network information, which includes the RSSI (Received Signal Strength Indicator) values. TheRSSI value indicates the strength of the Wi-Fi signal received from each network.

- **Mapping RSSI to Image Co-ordinates:**

Determination of the mapping between RSSI values and image co-ordinates in the application. This mapping defines how the RSSI values will be translated into specificpoints or regions on the image.

Establishing a corresponding range of image co-ordinates that align with the desired locations on the image in our case the image width and height are respectfully 400 and 600. Thus, co-ordinates range from (0,0) to (400,600). We convert these image co-ordinates to pixel co-ordinates based on mathematical calculations for matrix operations.

In order to map the image co-ordinates to RSSI value we take on manual approach, process is to go on to physical location matching the location in the map and then submitting the corresponding co-ordinates to the RSSI value. This process can beimproved in future studies.

Moving the device to different positions within the mapping area while scanningthe Wi-Fi networks. This allows you to capture a range of RSSI values corresponding to different locations on the image.

Storing the labeled data, which consists of RSSI values and their correspondingimage co-ordinates, for further processing and training. This labeled dataset will serveas the training data for your machine learning model.

E. Handling Shortest path Images

Facilitation of the transfer of images from a mobile device to a backend server in the context of indoor navigation. The backend server can process the image, generate navigation data, store the necessary information, and provide the relevant data back tothe mobile app for indoor navigation purposes.

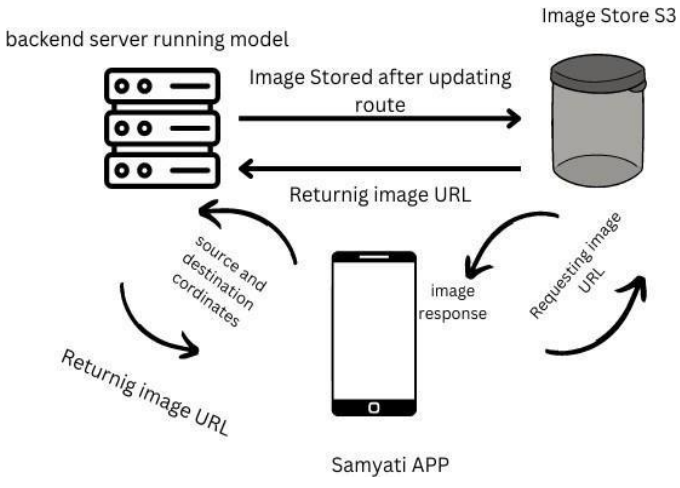

Figure 5: Server Architecture.

- **Samyati Mobile App:**

Retrieve the source and destination coordinates from the user within your mobile app. These coordinates will represent the starting point and desired destination within the indoor

environment.

- **API Endpoint Creation**:

Creating an API endpoint on your backend server that will receive the image, source coordinates, and destination coordinates from the mobile app. This endpoint will handlethe image transfer and processing for indoor navigation.

- **Image Transfer to Backend:**

API Request: From the mobile app, make an HTTP request to the API endpoint on thebackend server, sending the image file, source coordinates, and destination coordinatesas parameters in the request payload.

- **Handle Image Processing:**

On the backend, receive the API request and extract the image, source coordinates, anddestination coordinates from the request payload.

Implement the necessary image processing operations on the backend server using computer vision techniques or specialized libraries. This includes applying A* algorithm as described above.

- **Store Navigation Data:**

Store the generated navigation data, including floor plans, waypoints, paths, or any other relevant data, in your backend server's database or storage. Particularly S3 is usedas object storage image store. S3 object URL is returned.

- **Return Navigation Data and Image URL:**

Send the generatednavigation data and image URL as a response from the backend API to the mobile app.

F. Outdoor navigation

A robust toolkit for building interactive, dynamic maps that let users explore outside isprovided by the MapGL API for outdoor navigation. we may take advantage of a varietyof capabilities using the MapGL API to create effective outdoor navigation applications. Map rendering, geolocation, route planning, and user interaction are someof these capabilities.

Developers can start by incorporating the MapGL API into their application to implement outdoor navigation. Map tiles can be displayed using the API, and different geographic information like highways, landmarks, and sites of interest can be overlaid.Users can interact with the map by using the built-in zoom and pan controls of the API. The MapGL API offers routines to retrieve the user's current location using the device's GPS or other geolocation technologies. There exists access to wealth of resources withthe MapGL API.

The MapGL API is a strong resource that gives programmers the capacity to produce dynamic and resizable maps for outdoor navigation apps. It is based on cutting-edge web technologies like WebGL, which enables fast rendering of maps with fluid animations.

The extensibility and adaptability of the MapGL API are two of its primary characteristics. Map styles, markers, overlays, and layers are only a few of the many options available for altering the map's design. To produce the correct visual depictionof the outer environment, you can select from a variety of map types, including satellite imagery, street maps, or bespoke layouts.

G. Voice input and read-aloud navigation

An indoor navigation app tailored for visually impaired individuals plays a vital role in enhancing mobility and independence within indoor environments. By leveraging technology and providing specialized features, these apps offer a range of benefits to empower visually impaired users in their navigation endeavors.

Indoor navigation apps serve as reliable companions by providing step-by-stepdirections and guidance within complex indoor spaces. Through the use of audio cues and spoken instructions, visually impaired individuals can receive real-time guidance, enabling them to navigate from one location to another with confidence and ease.

There are two libraries, that actually make this possible,

- React-native-voice
- React-native-tts

React Native Voice is a library that enables speech recognition capabilities in ReactNative applications. It allows the app to capture spoken words or phrases and convert them into text format. This functionality can be used to interpret voice commands and adjust in our current location room. Making it possible for hands-on navigation.

In the context of indoor navigation, app can leverage React Native Voice to provide voice input for navigating through different locations or performing actions based on voice commands. By capturing spoken instructions from the user, application can thenprocess the text and interpret it to trigger relevant navigation actions within the indoornavigation system.

React Native TTS, on the other hand, stands for Text-to-Speech. This library provides capabilities for converting text into audible speech. It allows the app to generate voice output from text, providing audio feedback to users. React Native TTS supports various voices and customization options, such as pitch, volume, and speed control.

For indoor navigation, React Native TTS can be used to provide audio instructionsor guidance to users. Instead of relying solely on visual cues, app can convert text-based navigation instructions into spoken words, allowing users to receive real-time guidancethrough auditory means. This can be particularly useful for users with visual impairments or in situations where visual cues may not be easily visible or accessible. There is an endpoint running in the background that formulates the next step of directiona user needs to follow, based on the generated path and his current location. This algorithm running in the background calculates the angle of inclination and provide his directional guidance. Which the tts picks up and guides the user by reading aloud.

By combining React Native Voice and React Native TTS, app can create a comprehensive indoor navigation system that supports voice input for user instructionsand provides audio feedback with text-to-speech capabilities for navigation guidance.

Users can speak their commands, and the system can respond with spoken instructions,enhancing the overall user experience and accessibility of your application.

VII. RESULTS

A. Prediction

Figure 6: Samyati

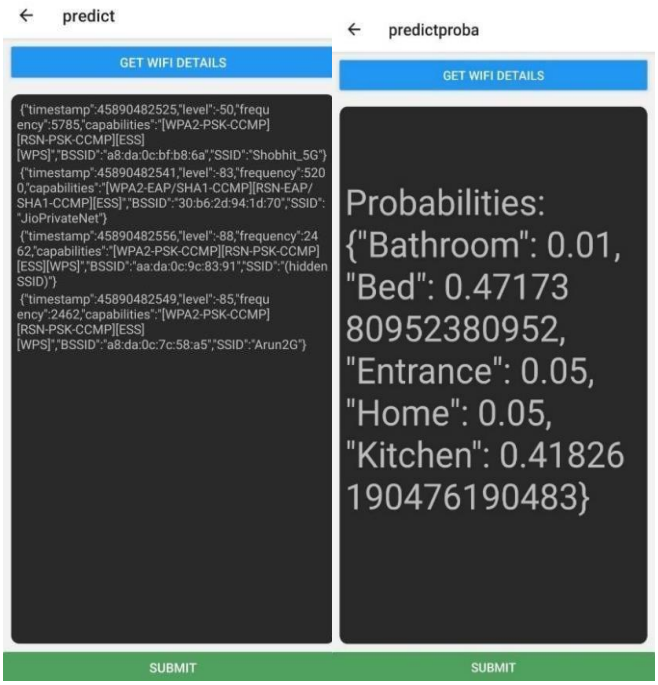

Figure 7: Samyati Prediction screens

Above results in figure 6 and figure 7 indicate Samyati prediction pages in order for testing the wi-fi detailsand also the probability of each location. As the results above indicate there is 0.47percent probability that “Bed” is the current location when compared to the rest. Ultimately indicating your position as Bed.

In the figure 8 comparison between alternative machine learning models are depicted. Ultimately Random Forest has highest accuracy when compared to the rest. KNN is close next to second but has very high disadvantage that it is a lazy learner.

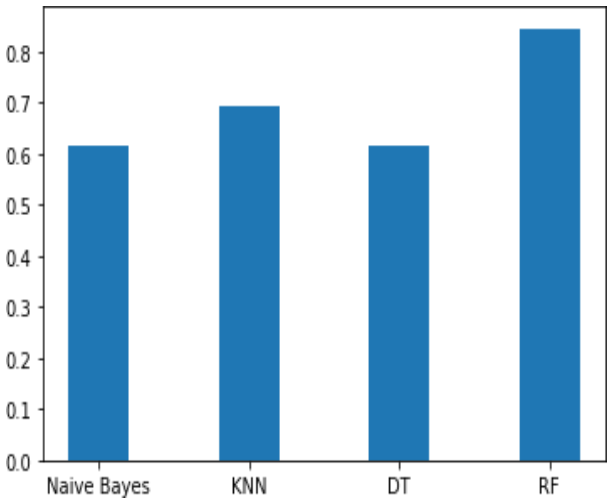

Figure 8: Comparison of accuracy

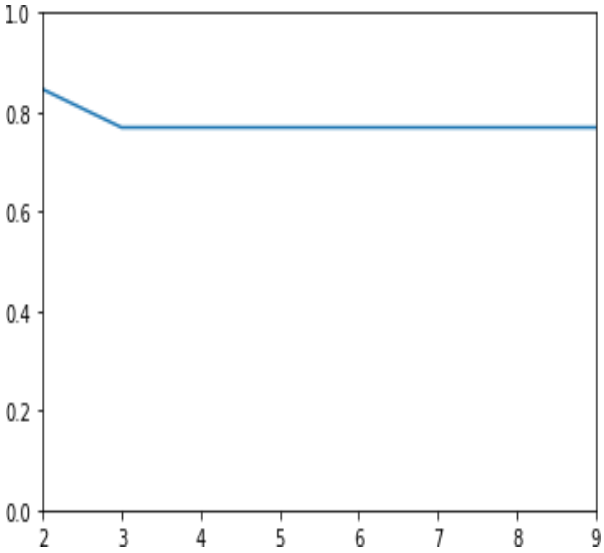

Figure 9: Accuracy of Random Forest

B. Floor Plan

Figure 10: Floor Plan before conversion

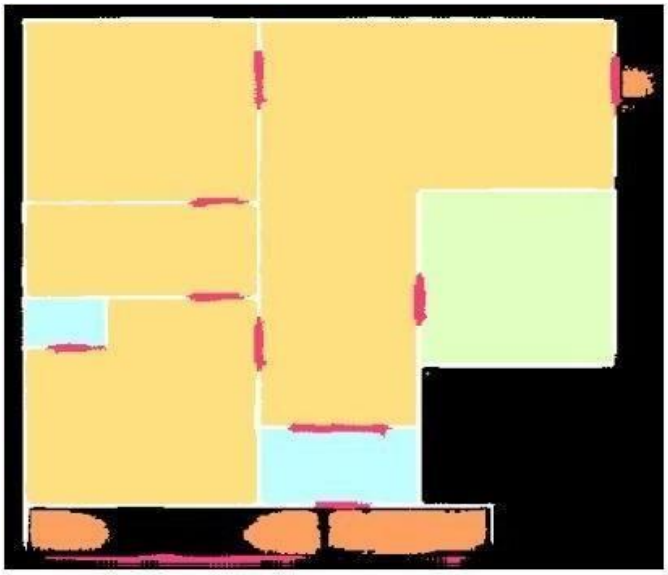

Figure 11: Floor plan after conversion

Figure 12: Loss function Result 1

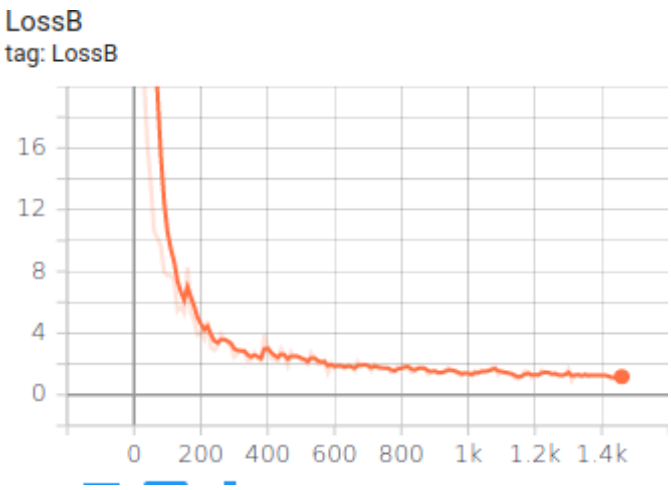

Figure 13: Loss function Result 2

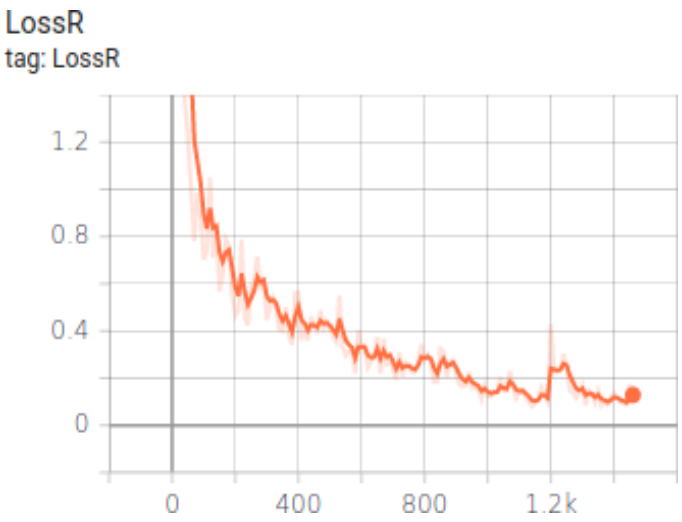

Figure 14: Loss function Result 3

C. Admin UI

Figure 15: Admin authentication

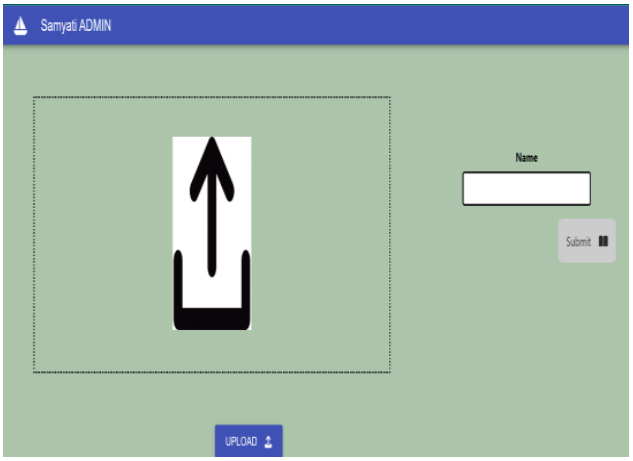

Figure 16: Admin upload floor plans

Samyati also provides web interface for the admin to upload the designated floor plans,that gets stored in s3 buckets with name of the location as image name. This image storeis connected to the backend hence a copy of the same will be updated in the backed server every time.

D. Navigation

Figure 17: Simulating user navigation with directions

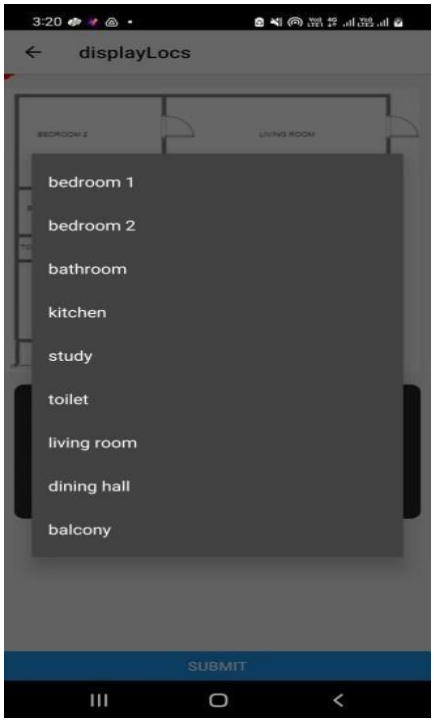

Figure 18: Select Source and destination

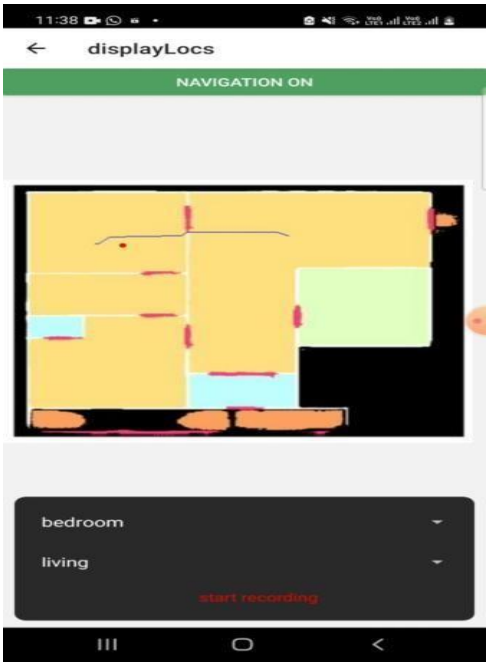

Figure 19: Indoor Navigation

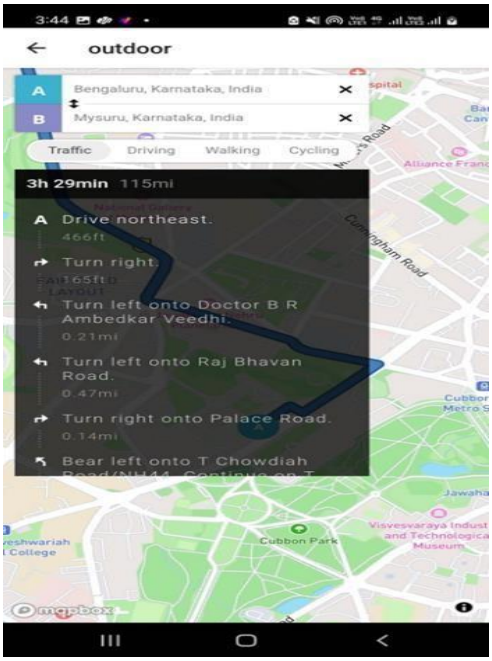

Figure 20: Outdoor navigation

VIII. CONCLUSION AND FUTURE WORK

In conclusion, Indoor navigation technology is a transformative advancement, enhancing accessibility, safety, and user experiences within enclosed spaces. Utilizing advanced positioning systems, smart mapping, and personalized recommendations, it empowers individuals to effortlessly navigate complex indoor environments. This technology is especially valuable for those with disabilities, older adults, and

tourists in unfamiliar buildings, and aids emergency responders during critical situations, improving safety and response times. The interface developed here, combining deep learning and Wi-Fi technology for real-time position tracking, holds immense potential for assisting the visually impaired in indoor navigation. Deep learning algorithms interpret complex environmental data, providing precise guidance, and real-time Wi-Fi tracking ensures accuracy. These advances promise inclusive and accessible environments, enabling visually impaired individuals to navigate independently.

The future involves scaling the project and improving routing and guidance. Currently, we can train the model for up to two places at a time. Further enhancements will focus on dynamic path re-routing and real-time guidance adjustments, offering a more flexible and responsive navigation experience.

REFERENCES

[1] Shah, Jinesh & Raorane, Aashreen & Ramani, Akash & Rami, Hitanshu &Shekokar, Narendra. (2020). EYERIS: A Virtual Eye to Aid the Visually Impaired. 202-207. 10.1109/CSCITA47329.2020.9137777.

[2] Chiou, Meng-Jiun & Liu, Zhenguang & Yin, Yifang & Liu, Anan & Zimmermann, Roger. (2020). Zero-Shot Multi-View Indoor Localization viaGraph Location Networks.

[3] 2001. CMU World Wide Knowledge Base (Web-KB) project. http://www.cs.cmu. edu/afs/cs.cmu.edu/project/theo-11/www/wwkb/. Accessed: 2020-08-04.

[4] Chen, Zhenghua & Alhajri, Mohamed & Wu, Min & Ali, Nazar & Shubair, Raed. (2020). A Novel Real-Time Deep Learning Approach for Indoor Localization Based on RF Environment Identification. PP. 1-1. 10.1109/LSENS.2020.2991145.

[5] Taira, Hajime & Rocco, Ignacio & Sedlar, Jiri & Okutomi, Masatoshi & Sivic, Josef & Pajdla, Tomas & Sattler, Torsten & Torii, Akihiko. (2019). Is This theRight Place? Geometric-Semantic Pose Verification for Indoor Visual Localization. 4372-4382. 10.1109/ICCV.2019.00447.

[6] C. P. Simonsen, F. M. Thiesson, M. P. Philipsen, and T. B. Moeslund, "Generalizing Floor Plans Using Graph Neural Networks," 2021 IEEE International Conference on Image Processing (ICIP), 2021, pp. 654-658, doi: 10.1109/ICIP42928.2021.9506514.

[7] Kim, Nuri & Kwon, Obin & Yoo, Hwiyeon & Choi, Yunho & Park, Jeongho & Oh, Songhwai. (2022). Topological Semantic Graph Memory for Image-Goal Navigation.

[8] I A Koc, T. Serif, S. Gören and G. Ghinea, "Indoor Mapping and Positioning using Augmented Reality," 2019 7th International Conference on Future Internetof Things and Cloud (FiCloud), 2019, pp. 335-342, doi: 10.1109/FiCloud.2019.00056.

[9] Zhiliang Zeng Xianzhi Li Ying Kin Yu Chi-Wing Fu The Chinese University of Hong Kong, Deep Floor Plan Recognition Using a Multi-Task Network with Room-Boundary-GuidedAttention.

AUTHORS

First Author –S Brunda, Assistant Professor, Department of Computer Science and Engineering, JSS STU, Mysuru and email address: sbrunda@sjce.ac.in

Second Author – Kishan L, Department of Computer Science and Engineering, JSS STU, Mysuru and email address: kishanabola@gmail.com

Third Author – Gowtham M, Department of Computer Science and Engineering, JSS STU, Mysuru and email address: gowtham1842001@gmail.com

Fourth Author – Shobhith K, Department of Computer Science and Engineering, JSS STU, Mysuru and email address: shobhithkamata01@gmail.com

Fifth Author – Eshwar Shashidhar, Department of Computer Science and Engineering, JSS STU, Mysuru and email address: eshwarshashidhar@gmail.com