SMART CAMPUS NAVIGATION ASSISTANT

MINOR PROJECT REPORT

Submitted by

Shweta Liju (Register No: CHN22EC087)

K Abhinav Krishna (Register No: CHN22EC058)

Sreedhanya S (Register No: CHN22EC091)

M Fathima Sudheer(Register No: CHN22EC066)

to

APJ Abdul Kalam Technological University

in partial fulfillment of the requirements for the award of

B.Tech in Electronics and Communication Engineering

with Minor in B.Tech Computer Science and Engineering

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

COLLEGE OF ENGINEERING CHENGANNUR, ALAPPUZHA

OCTOBER 2025

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

COLLEGE OF ENGINEERING CHENGANNUR

ALAPPUZHA

CERTIFICATE

This is to certify that the project report titled ***"Smart Campus Navigation Assistant"*** *is a bonafide record of the CSD481 MINI PROJECT presented by* ***Shweta Liju (CHN22EC087)***, ***K Abhinav Krishna (CHN22EC058)***, ***Sreedhanya S (CHN22EC091)***, *and* ***M Fathima Sudheer (CHN22EC066)***, *Seventh Semester B.Tech Degree students, under my guidance and supervision. This project is submitted in partial fulfillment of the requirements for the award of B.Tech Degree in* ***Electronics and Communication Engineering*** *with Minor in B.Tech Computer Science and Engineering of APJ Abdul Kalam Technological University.*

Dr. Renu George	**Ms. Geetha S.**
Head of Department	Project Coordinator
Associate Professor	Assistant Professor
Dept. of Computer Science	Dept. of Computer Science

DECLARATION

I hereby declare that the project report entitled **"Smart Campus Navigation Assistant"**, submitted in partial fulfillment of the requirements for the award of B.Tech Degree of the APJ Abdul Kalam Technological University, Kerala, is a bonafide work done by us under the supervision of **Ms. Geetha S.**, Assistant Professor, Department of Electronics and Communication Engineering.

Place: Chengannur **Date:**

Shweta Liju, K Abhinav Krishna, Sreedhanya S,M Fathima Sudheer

ACKNOWLEDGEMENT

We would like to begin by expressing our heartfelt gratitude to the Almighty for granting us the strength, patience, and perseverance to successfully complete our minor project, “Smart Campus Navigation Assistant Using QR Codes and Voice Guidance.” The journey of this project has been both challenging and rewarding, and we owe our success to the unwavering support, guidance, and encouragement of many individuals who have contributed directly and indirectly to its completion.

We extend our sincere appreciation to Dr. Hari V.S., Principal, College of Engineering Chengannur, for providing an excellent academic environment and infrastructure that encouraged innovation and research. His emphasis on practical learning and his commitment to maintaining high academic standards have been an inspiration to all of us. His constant encouragement motivated us to approach our project with confidence and enthusiasm.

Our profound gratitude also goes to Dr. Gopakumar C., Head of the Department of Electronics and Communication Engineering, for his consistent guidance, valuable insights, and encouragement throughout this project. His experience, suggestions, and professional outlook helped us refine our ideas and strengthen the technical aspects of our work. We are deeply indebted to him for his continuous support and motivation.

We express our special thanks to our project guide, Dr. Geetha S., Assistant Professor, Department of Computer Science and Engineering, whose mentorship, constructive criticism, and constant guidance were vital in shaping this project. Her technical expertise, attention to detail, and dedication provided us with the clarity and direction needed to bring our concept to life. She was always available with valuable feedback, which helped us overcome every obstacle we faced during development. Her patience, encouragement, and belief in our abilities kept us motivated at every stage.

Our sincere appreciation goes out to all the faculty members of the Department of Electronics and Communication Engineering for their valuable lectures, technical support, and guidance throughout our academic journey. The knowledge and skills we gained in their classes formed the foundation upon which this project was built. We are also thankful to the non-teaching staff for their help in providing the necessary resources and facilities during the project work.

ABSTRACT

Navigating through large educational campuses can often be confusing and time-consuming, especially for new students and visitors unfamiliar with building layouts. The proposed project, **Smart Campus Navigation Assistant**, aims to provide an efficient, accessible, and low-cost indoor navigation solution.

This system integrates QR code scanning, graph-based shortest path algorithms, and voice-enabled navigation to guide users to their destinations. When a user scans the QR code placed at the campus entrance, the system prompts for the destination. Using Dijkstra's algorithm, the optimal route is computed within a predefined campus graph. The system then generates both voice instructions (using gTTS) and visual route maps (using NetworkX and Matplotlib).

The solution eliminates the need for GPS or complex IoT infrastructure, enhances accessibility, and can easily be scaled to any institution.

Contents

List of Figures

Chapter 1

INTRODUCTION

1.1 Project Area

The project “Smart Campus Navigation Assistant” falls under the domain of Smart Campus Systems, IoT-based Applications, and Indoor Navigation Technologies.

In modern educational institutions, navigation across large buildings and multiple departments can be confusing for new students, parents, and visitors. Traditional signboards and printed maps often fail to provide real-time or accessible guidance. To overcome this, the project introduces a QR code–based smart navigation system that allows users to receive both visual and voice-guided directions to their chosen destination within the campus.

The system models the campus layout as a graph, where each room or area is treated as a node, and the pathways between them are represented as edges. By applying a graph traversal algorithm (Dijkstra’s algorithm), the system computes the shortest path between the starting point (usually the entrance) and the user-selected destination.

The system prompts for the destination and then provides voice instructions generated through Google Text-to-Speech (gTTS) along with a visual route map created using NetworkX and Matplotlib libraries. The route map can also be shared through a QR code linked to Google Drive, making it accessible from any smartphone.

This project integrates multiple technologies—speech synthesis, graph-based algorithms, and QR code generation—into a single, unified framework, creating a cost-effective, user-friendly, and scalable navigation solution suitable for educational campuses.

1.2 Objectives

The main objectives of this project are:

- To design and develop a QR code–based interface for initiating navigation on the campus.
- To represent the campus layout as a graph structure consisting of nodes (locations) and edges (pathways).

- To apply Dijkstra's shortest path algorithm to determine the most efficient route between two points.
- To generate voice-based navigation instructions using the gTTS (Google Text-to-Speech) module.
- To visualize the computed route through a map image showing room names and floor information.
- To create a QR code for sharing the complete route map accessible through Google Drive.
- To ensure that the system is accessible, low-cost, and easy to implement for any educational campus environment.

Chapter 2

PROBLEM DEFINITION AND MOTIVATIONS

In large educational institutions, students, faculty members, and visitors often face difficulties in locating classrooms, departments, laboratories, and administrative offices. This issue becomes more significant in multi-storey buildings or campuses with complex layouts. The absence of a digital indoor navigation system leads to confusion, delays, and unnecessary human dependency. Hence, there is a strong motivation to develop a system that can simplify indoor navigation using modern technology in a cost-effective manner.

2.1 Existing System

The existing methods for navigation within educational campuses are mostly manual and static. Visitors and new students usually rely on direction boards, printed maps, or help desks to find their way. While these traditional systems serve the basic purpose, they are not always accurate or convenient in real-world scenarios.

In some advanced institutions, GPS-based mobile navigation applications are used. These systems are effective for outdoor guidance, such as reaching the campus premises. However, GPS signals become weak or unavailable inside buildings, making them unreliable for indoor navigation. Indoor maps available through certain platforms are also not updated frequently and require internet access for continuous functioning.

Furthermore, manual navigation aids do not cater to differently abled individuals, particularly those with visual impairments. As a result, there is a need for an interactive system that combines visual as well as audio-based navigation support to assist every user effectively.

2.2 Limitations

The existing systems suffer from several drawbacks that limit their efficiency and user-friendliness:

- **GPS Signal Failure:** GPS technology does not function accurately within enclosed areas, as signals are blocked by walls and ceilings.
- **Static Signboards:** The direction boards and signage are fixed and cannot be dynamically updated when room allocations or departments change.

- **Lack of Interactivity:** Printed maps or boards do not provide step-by-step assistance or audio feedback.
- **Manual Effort:** Users have to rely on help desks or staff, causing inconvenience and time delay.
- **Accessibility Challenges:** Visually impaired individuals face difficulty in following static signs or maps.
- **Maintenance Needs:** Some existing digital systems require continuous internet connectivity and frequent maintenance to remain functional.

These limitations highlight the necessity of a smart, self-guided navigation system that operates offline, provides both audio and visual cues, and requires minimal infrastructure.

2.3 Problem Statement

To design and implement a voice-enabled, QR code–based indoor navigation system that helps users reach their desired destination inside a college campus. The system should compute and display the shortest and most efficient path from a defined starting point (Entrance) to the selected destination using graph-based algorithms.

2.4 Aim

This project aims to make navigation simple, efficient, and accessible for all individuals, including those unfamiliar with the campus environment.

2.5 Proposed System

The proposed system, **Smart Campus Navigation Assistant Using QR Codes and Voice Guidance**, provides an interactive, low-cost, and scalable solution to overcome the drawbacks of existing systems. At the entrance, the system prompts for a destination and then calculates the shortest route within a graph-based model of the campus.

The graph consists of nodes representing key locations such as offices, seminar halls, classrooms, and departments, and edges representing the physical paths connecting them. Using Dijkstra's algorithm, the system determines the optimal route from the start node to the target node.

Once the route is determined, the system generates step-by-step voice guidance using the Google Text-to-Speech (gTTS) library, allowing the user to listen to instructions such as "From Entrance, go to 3rd floor to reach CSE Department." Simultaneously, a visual route map is generated using NetworkX and Matplotlib, displaying the connection between nodes, room numbers, and floor levels. This image is also linked to a Google Drive QR code, allowing easy sharing and mobile access.

Thus, the proposed system integrates QR code scanning, graph computation, visual display, and voice guidance to form a complete, interactive campus navigation tool.

2.5.1 Key Idea

The core idea of this system is to model the entire campus environment as a graph. Each room, hall, or department acts as a node, and the connections (corridors, stairs, or pathways) act as edges between these nodes. Each edge is assigned a weight, representing the approximate distance or time required to travel between two points. Using Dijkstra's algorithm, the system efficiently determines the shortest path between any two given locations.

This representation allows flexibility — if any room or connection changes, it can easily be updated in the graph without redesigning the entire system.

2.5.2 Features

The proposed system offers several features that enhance its usability and functionality:

- **Voice Guidance:** Step-by-step instructions are provided using gTTS for hands-free and accessible navigation.
- **Graph-Based Route Calculation:** Shortest path computed using Dijkstra's algorithm ensures efficiency.
- **Visual Route Map:** The system displays a graphical route representation using NetworkX and Matplotlib.
- **Google Drive Integration:** The final route map is linked to a Drive-based QR code for remote access.
- **Offline Functionality:** The system can function without continuous internet connectivity once set up.
- **Accessibility and Ease of Use:** Supports all types of users, including those who are visually challenged or new to the institution.

Chapter 3

LITERATURE REVIEW

Smart campus navigation has emerged as an important area of research with the growth of large educational institutions and the need for accessible indoor way-finding. Traditional signboards and maps are often insufficient for guiding new students and visitors. Over the last decade, researchers have explored several digital approaches using technologies such as QR codes, Bluetooth Low Energy (BLE) beacons, Internet of Things (IoT) devices, Augmented Reality (AR), and graph-based algorithms to create efficient and scalable navigation systems.

Kavita et al. (2023) proposed a **web-based campus navigation system using QR codes**, in which users scan QR tags placed at various campus locations to receive navigation directions on their smartphones [1]. This approach is cost-effective and easy to deploy, as it requires no additional hardware. However, it depends on manual QR scanning and lacks real-time adaptability.

Robert Riesebos et al. (2022) developed a **smartphone-based indoor positioning system using BLE beacons** [2]. The model determines the user's position through signal strength received from BLE beacons distributed across the environment. This technique performs well in areas where GPS signals fail, but its high setup cost and interference issues limit scalability.

Zhang et al. (2025) presented an **IoT-enabled smart campus navigation framework** for intelligent administration [3]. The system integrates IoT devices and sensors for both navigation and resource management. Although it improves efficiency and scalability, the complexity and maintenance cost are significant.

Wei Li et al. (2021) performed a comparative analysis of **shortest-path algorithms for pedestrian navigation** [4]. Algorithms such as Dijkstra and A* were evaluated for efficiency and accuracy. While computationally strong, the study remained theoretical without real-world validation.

Li, Xu, and Wang (2022) explored **QR code-based indoor path planning** by converting floor maps into navigation graphs [5]. This approach achieved accurate and low-cost route generation, but performance was influenced by lighting conditions and QR code degradation.

Kumar et al. (2025) proposed a **web-based campus navigation system using QR codes linked to a live database** [6]. It dynamically computes the shortest path and updates routes in real time. The method is cost-effective but relies on stable internet connectivity and frequent QR maintenance.

Wahbeh et al. (2022) introduced **SoshaMapBot**, a chatbot-based campus navigation assistant that uses text and image inputs to infer location [7]. By applying natural language processing, it provides multimodal user interaction. However, accuracy remains around 75% and privacy concerns arise due to photo inputs.

Kim et al. (2025) leveraged **augmented reality (AR)** to display navigation overlays directly on smartphone cameras [8]. The AR system offers visual, intuitive guidance but demands high processing power and reliable indoor positioning support.

Patel et al. (2024) presented **Campus Navigator**, a web-based multi-modal path-finding platform developed with Python and Django [9]. It supports walking, cycling, and shuttle routes through API integration. However, its dependence on external APIs and constant internet connectivity reduces robustness.

Liu et al. (2024) conducted a detailed comparison of **indoor positioning systems** such as Wi-Fi, BLE, and UWB [10]. Their analysis emphasizes hybrid positioning for improved accuracy, but notes high deployment costs and the absence of universal standards.

Bhatnagar and Singh (2021) applied the **Small World Network model** to optimize large-scale network navigation [11]. Their study showed scalability and reduced computation time, but required precise network modeling.

Kaur and Reddy (2023) provided a conceptual review titled **The Making of Smart Campus**, outlining a sustainability-oriented framework integrating technology, services, and users [12]. Although holistic, it remains conceptual without practical implementation.

From the review, it is evident that QR-based systems are low-cost but manual, BLE/IoT solutions offer real-time positioning but are expensive, and AR interfaces enhance accessibility but rely on hardware performance. Graph-based algorithms provide mathematical accuracy but need structured data.

The proposed **Smart Campus Navigation Assistant** combines the best features of these approaches. It uses QR-code initiation for simplicity, Dijkstra's algorithm for efficient route computation, and gTTS-based voice assistance for accessibility. This hybrid system ensures affordability, scalability, and inclusivity, making it suitable for modern academic environments.

Chapter 4

REQUIREMENT ANALYSIS

The requirement analysis phase identifies and specifies the essential needs of both the system and the user. It defines the hardware, software, functional, and non-functional aspects necessary for the proper operation of the Smart Campus Navigation System. This phase ensures that the system is designed to meet performance, usability, and quality goals effectively.

4.1 Hardware Requirements

Although the proposed system is primarily software-oriented, certain hardware components are necessary for its implementation and testing. The hardware requirements are as follows:

- **Smartphone:** Used by users to scan QR codes and access the navigation interface.
- **Computer or Laptop:** Required for program development, execution of Python scripts, and map visualization.
- **Speakers or Headphones:** Used for playing the audio navigation instructions generated by the gTTS library.
- **QR Code Printouts:** Placed at strategic campus locations for scanning and accessing route maps.
- **Wi-Fi or Internet Connectivity:** Enables access to cloud-stored route maps (Google Drive links) and supports online functionalities.

4.2 Software Requirements

The Smart Campus Navigation System utilizes several software tools, libraries, and platforms for development and deployment. The software requirements are listed below:

- **Operating System:** Windows / Linux
- **Programming Language:** Python 3.x
- **Libraries Used:**
  - *NetworkX* – for creating campus graphs and performing shortest path computation using

Dijkstra's Algorithm.

  - *Matplotlib* – for plotting and visualizing route maps.
  - *gTTS (Google Text-to-Speech)* – for converting text-based navigation instructions into speech output.
  - *qrcode* – for generating QR codes linked to the route maps.

- **Cloud Service:** Google Drive for storing and sharing the generated route maps.
- **Integrated Development Environment (IDE):** Visual Studio Code / Jupyter Notebook for program execution and testing.

4.3 Functional Requirements

Functional requirements describe the key operations and services provided by the system to fulfill user needs. The main functional requirements of the proposed system are:

1. **QR Code Scanning:** The system enables users to scan QR codes placed at different locations within the campus to access navigation routes.
2. **Graph-Based Navigation:** The campus layout is represented as a graph consisting of nodes (rooms, halls) and edges (corridors, stairs).
3. **Shortest Path Computation:** The system computes the optimal route between source and destination using Dijkstra's algorithm.
4. **Voice Guidance:** Converts text-based navigation instructions into audio using the gTTS library, providing step-by-step voice assistance.
5. **Map Visualization:** Generates and displays a graphical route map showing the user's path to the destination.
6. **QR Code Sharing:** Produces a shareable QR code that links to the stored route map, allowing easy access on mobile devices.

4.4 Non-Functional Requirements

Non-functional requirements define the performance and quality characteristics of the system. These requirements ensure that the system operates efficiently and delivers a satisfactory user experience. The key non-functional requirements include:

- **Usability:** The system should be user-friendly and require minimal effort to operate, even for first-time visitors.

- **Scalability:** The system must allow easy addition of new buildings, rooms, or floors without major redesign.
- **Reliability:** The system should provide consistent and accurate navigation results under normal conditions.
- **Maintainability:** QR codes and corresponding route maps should be easily updatable when campus layouts change.
- **Portability:** The system should be executable on multiple platforms with minimal configuration.

4.4.1 Performance Requirements

The performance requirements define the operational efficiency and responsiveness expected from the Smart Campus Navigation System. These requirements ensure that the system performs computations and delivers navigation outputs within acceptable time limits.

- The system should compute the shortest path within **two seconds** for a typical campus graph.
- The **voice guidance** must be generated immediately after the shortest path computation.
- **QR code creation** and route map sharing should complete within a few seconds to maintain smooth user interaction.
- The system should efficiently handle **large graphs containing hundreds of nodes (rooms)** without noticeable delay.

Performance Analysis and Measurement

The system's performance is analyzed and measured using several quantitative metrics, as described below:

1. **Response Time (T_r):** Measures the total time taken from user input (source–destination selection) to final output (generation of text, voice, and map).
   Measurement Method: The system records timestamps at the start and end of execution using Python's `time` module.
   Formula:
   $$T_r = t_{\text{output}} - t_{\text{input}}$$

2. **Processing Efficiency:** Indicates how many graph nodes are processed per second during shortest path computation.

   Measurement Method: The number of nodes and execution time are logged, and the efficiency is calculated as:

$$\text{Efficiency} = \frac{\text{Total Nodes Processed}}{\text{Execution Time (sec)}}$$

3. **RMS Error (Root Mean Square Error):** Evaluates the accuracy of the computed path length compared to the actual measured campus distance.

   Formula:

$$\text{RMS Error} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(d_i - \hat{d}_i)^2}$$

   where d_i = actual distance and $\hat{d}_i$ = computed path distance.

   Measurement Method: Actual distances between rooms are measured using building floor plans, and computed distances are extracted from the system's graph data.

4. **Memory Utilization:** Indicates the efficiency of resource management during path calculation.

   Measurement Method: Memory usage is measured using Python's `psutil` library during runtime.

5. **Throughput:** Represents the number of navigation requests successfully processed per minute.

   Measurement Method: A series of user requests are simulated, and total completions per minute are recorded.

These metrics ensure that the system's computational modules (graph generation, shortest-path computation, voice guidance) operate within defined performance limits and can scale efficiently for larger campus networks.

4.4.2 Quality Requirements

Quality requirements specify the attributes that ensure reliability, accuracy, and user satisfaction. They focus on maintaining consistent output and an accessible user experience across all components of the Smart Campus Navigation System.

- **Accuracy:** The computed navigation paths should maintain an RMS error of less than 5% compared to actual distances.
- **Accessibility:** The integration of text-to-speech (gTTS) provides voice guidance for visually impaired users, enhancing inclusivity.

- **Consistency:** The text, voice, and visual map outputs should remain synchronized to deliver coherent navigation guidance.
- **User Experience:** The interface and output visualization should be simple, readable, and intuitive for all users.
- **Reliability:** The system should generate repeatable results under identical conditions.
- **Security:** QR codes and map links must only connect to verified campus map files to prevent misuse.

Quality Analysis and Measurement

The following quantitative and qualitative methods are used to evaluate system quality:

1. **Accuracy Index (A):** Measures the ratio of correctly computed paths to total tested paths.

   Formula:
   $$A = \frac{\text{Correct Paths Generated}}{\text{Total Paths Tested}} \times 100\%$$
   Measurement Method: Several known routes within the campus are tested. The computed path is compared with manually verified shortest routes.

2. **User Satisfaction Score:** Collected through user surveys based on ease of use, clarity of voice guidance, and accuracy of directions.

   Measurement Method: Users rate their experience on a scale of 1–5, and the average score represents the satisfaction index.

3. **Mean Opinion Score (MOS):** Evaluates the perceived audio clarity and intelligibility of the voice guidance output.

   Measurement Method: 10–15 users rate the audio quality on a scale from 1 (poor) to 5 (excellent).

4. **Error Rate (E):** Indicates the percentage of incorrect or failed navigation outputs during testing.

   Formula:
   $$E = \frac{\text{Incorrect Outputs}}{\text{Total Test Cases}} \times 100\%$$
   Measurement Method: The number of mismatched or failed navigation cases is logged during testing.

5. **Consistency Check:** Ensures that generated text, map, and voice outputs correspond to the

same route sequence.

Measurement Method: Automated verification scripts cross-validate route consistency between all output formats.

These measurement methods help confirm that the system consistently meets expected accuracy, usability, and accessibility standards, making it reliable for real-world deployment in campus environments.

Chapter 5

DESIGN AND IMPLEMENTATION

5.1 Overall Design

The overall design of the **Smart Campus Navigation Assistant** focuses on providing users with an easy, efficient, and accessible method to navigate inside a college campus. The system architecture integrates four essential modules — QR code interface, graph-based pathfinding, voice output, and visual route display — to deliver both visual and auditory guidance to the user.

Initially, the user can input their desired destination. The backend uses a graph structure to model the entire campus, representing rooms, labs, and offices as nodes, and pathways between them as edges with assigned directions and distances.

Once the user's input is received, the system applies Dijkstra's shortest path algorithm to determine the most optimal route between the starting point and the destination. It then provides clear, step-by-step voice instructions generated using Google Text-to-Speech (gTTS). Simultaneously, a visual route map is created using NetworkX and Matplotlib, while a QR code is generated for sharing the full route image via Google Drive.

This design ensures that the navigation process is interactive, intuitive, and accessible to all users, eliminating the dependency on GPS or complex hardware-based systems. The design also emphasizes low cost, easy maintenance, and scalability, making it ideal for educational campuses.

5.1.1 System Design

The system is designed to be modular, ensuring that each component functions independently while maintaining smooth integration with other modules. The system is composed of the following four core modules:

1. **Voice Guidance Module**
    - Converts textual navigation instructions into audio using gTTS.
    - Provides sequential voice playback to help the user move step by step.
2. **Graph Navigation Module**

- Represents the entire campus structure as a directed graph consisting of nodes (locations) and edges (paths).
- Uses Dijkstra's algorithm to compute the shortest route from the entrance to the target destination.

3. **QR Code and Sharing Module**
   - Generates a single QR code that links to the full route map stored on Google Drive.
   - Enables users to access the map easily from any smartphone device.
4. **Map Visualization Module**
   - Uses NetworkX and Matplotlib to generate floor-wise visual representations of the computed route.
   - Displays room numbers and departmental names clearly for better understanding.

This modular design allows the system to be easily scalable and adaptable. New rooms or floors can be added simply by updating the graph data structure, without modifying the entire system.

5.1.2 System Architecture

The system architecture follows a layered and modular flow, integrating both frontend and backend components to create a smooth navigation process.

The architecture consists of three primary layers:

1. **Input Layer:** The starting location is given as Entrance and the system prompts the user to enter their destination name.
2. **Processing Layer:** The system constructs a directed graph of the campus using NetworkX. Dijkstra's algorithm calculates the shortest route between the entrance and the desired location. Route data (nodes, distances, and directions) is processed into text-based step instructions.
3. **Output Layer:** The Google Text-to-Speech (gTTS) library converts textual directions into audible guidance. A visual route map is generated and displayed using Matplotlib. A QR code linking to the complete route image on Google Drive is also displayed for easy sharing.

This architecture ensures that the system remains cross-platform, low cost, and efficient, while maintaining simplicity and modularity.

5.2 Use Case Diagram

The Use Case Diagram explains how different actors interact with the system. It visualizes user activities and system responses during navigation.

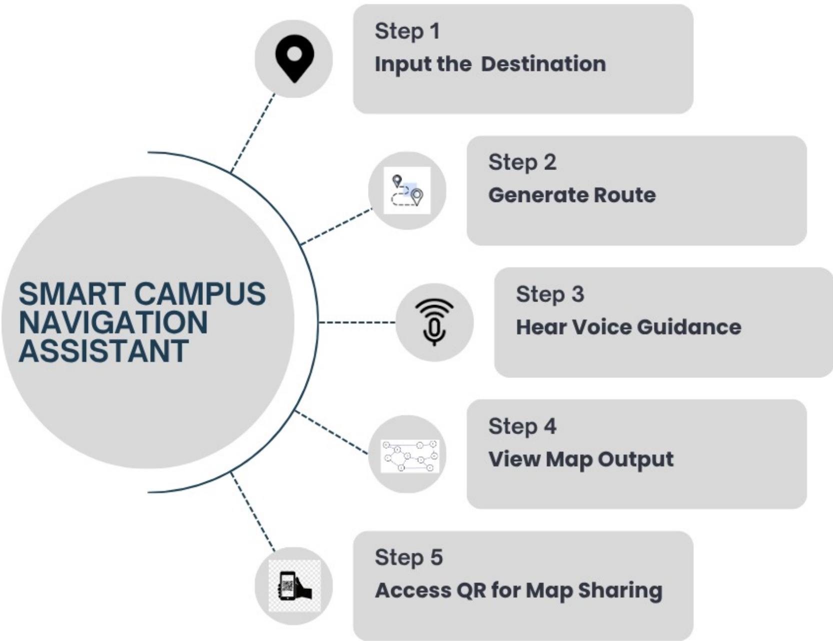

Figure 5.1: Use Case Diagram

Actors: User (Student / Visitor / Faculty Member)

Use Cases:

- Reach at the entrance
- Input destination
- Generate navigation path
- Listen to voice guidance
- Access shared QR code for map
- View map output

Description: When a user reaches the entrance, the system activates the navigation module. After entering the destination, the system computes the shortest route using Dijkstra's algorithm. The user then receives both voice instructions and visual map output. The generated route can also be accessed later by scanning the QR code that links to the Google Drive image.

5.2.1 Methodology

The methodology describes the step-by-step process involved in implementing the Smart Campus Navigation Assistant.

1. **Campus Modeling:** Each room, office, and department is represented as a node, and the pathways connecting them (corridors, stairs, or doors) are represented as edges with associated directions and distances.

2. **Graph Creation:** The system constructs a directed graph (`nx.DiGraph()`) using the Python library NetworkX. This graph forms the backbone of route computation.

3. **Shortest Path Calculation:** Dijkstra's Algorithm is applied to determine the shortest and most efficient path from the "Entrance" node to the "Destination" node.

4. **Voice Instruction Generation:** The text instructions derived from the computed path are converted into audio output using gTTS. Sequential playback is implemented to ensure smooth user experience.

5. **Map Visualization:** A visual map of the route is created using Matplotlib, showing node-to-node connections, room numbers, and floor information.

6. **QR Code Creation:** A QR code is generated using the qrcode library. This code links to the final route map image stored on Google Drive, making it accessible on mobile devices.

7. **User Interaction:** Finally, the system displays all outputs — audio instructions, visual map, and QR code — interactively to guide the user to the destination.

5.3 Algorithms Used

5.3.1 Dijkstra's Shortest Path Algorithm

The main algorithm used in this project is Dijkstra's Algorithm, which is a well-known technique for finding the shortest path between two nodes in a weighted graph. In this project, the entire campus is modeled as a graph where:

- Each room, department, or location is represented as a node.
- Each corridor, staircase, or connecting path is represented as an edge.
- Each edge is assigned a weight, which corresponds to the approximate distance between two

locations.

The algorithm determines the most optimal route between a starting location (Entrance) and a destination (e.g., CSE Department) by minimizing the total path cost (distance).

5.3.1.1 Algorithm Workflow

1. **Initialization:** All nodes are assigned a tentative distance value — the starting node (Entrance) is initialized with a distance of 0, while all others are set to infinity.
2. **Selection:** The unvisited node with the smallest tentative distance is selected as the current node.
3. **Update Step:** For each neighbor of the current node, the algorithm calculates the total distance from the source node. If this new path is shorter than the previously recorded distance, it updates the distance value for that node.
4. **Repeat Process:** The current node is then marked as visited, and the algorithm repeats the process for the remaining unvisited nodes until the destination node is reached.
5. **Output:** Once the shortest path is determined, the system generates corresponding text-based directions and converts them into voice instructions and a visual map.

5.3.1.2 Integration in the Project

In the implemented Smart Campus Navigation Assistant, Dijkstra's algorithm is applied using Python's NetworkX library to compute the shortest route from the entrance to the user's selected destination. However, this project extends the basic Dijkstra model by integrating room number intelligence:

- Each room in the system is represented by a unique room number (e.g., 316, 204, 112).
- The first digit of the room number corresponds to the floor number (e.g., 3 $\rightarrow$ 3rd Floor, 2 $\rightarrow$ 2nd Floor).
- Once the shortest path is calculated, the system automatically identifies which floor the user needs to go to based on the destination room number.

Example: If the destination is CSE Department (Room 316), the algorithm first computes the shortest route to reach that node in the graph. Then, using the first digit of the room number (3), it determines that the user must go to the 3rd Floor and provides an output such as:

"From Entrance, go to 3rd Floor to reach CSE Department. Arrived at CSE Department (Room: 316) ✓"

This ensures that both pathfinding and floor identification are handled intelligently within the same algorithmic framework.

5.3.2 Advantages of Using Dijkstra's Algorithm

1. **Accuracy:** Guarantees the exact shortest route between two locations.
2. **Efficiency:** Performs well for static indoor environments like campus maps.
3. **Scalability:** New rooms or buildings can easily be added to the graph.
4. **Integration:** Works seamlessly with Python libraries like NetworkX and supports visualization.

5.4 Data Flow Diagram

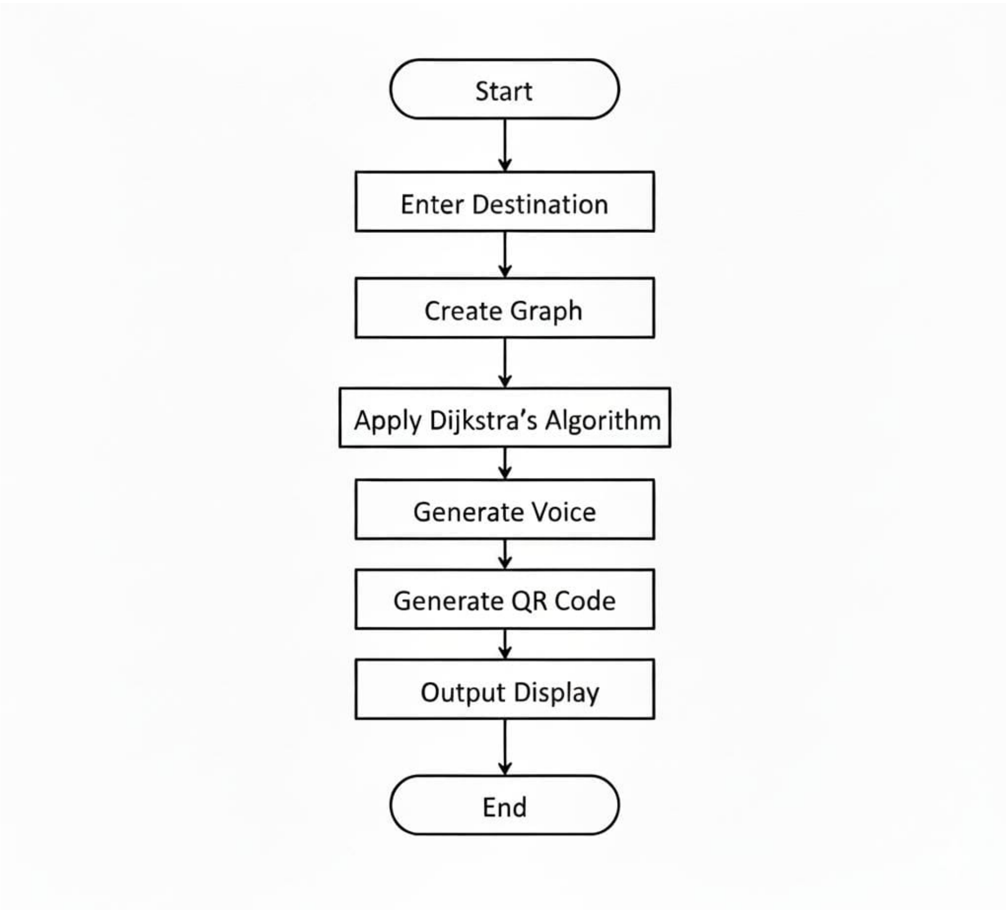

Figure 5.2: Data Flow Diagram

Explanation of Figure

Figure 5.2 illustrates the Data Flow Diagram (DFD) for the Smart Campus Navigation System, representing how data moves between the user, system processes, and data stores.

At the Level 0 (Context Diagram), the entire system is shown as a single process interacting with the User. The user provides inputs such as destination details and QR code scans. The system processes this information, accesses data from the Campus Map Database, and produces outputs such as route maps, voice guidance, and shareable QR codes for the user. This level gives an overall view of the system's functionality and its external interactions.

At Level 1, the system is decomposed into detailed subprocesses that describe the internal data flow:

1. The user begins by scanning the QR code and inputting the destination.
2. The system retrieves map data from the Campus Map Database and executes the **Generate Route** process using Dijkstra's algorithm to determine the shortest path.
3. The computed route information is passed to the **Display Route and Voice Guidance** process, which generates both a visual map and audio instructions using the gTTS module.
4. The output is provided to the user and stored in the **Route Record Storage** for future retrieval.
5. Finally, the **Generate QR for Sharing** process creates a QR code linking to the saved route map, allowing users to share navigation results easily.

The diagram clearly distinguishes between external entities (**User**), processes (**system functions**), and data stores (**Campus Map Database** and **Route Record Storage**), providing a structured view of how data is input, processed, and output in the system.

Chapter 6

PROJECT IMPLEMENTATION

6.1 Implementation Plan

This chapter explains how the Smart Campus Navigation Assistant was developed and executed. The project was implemented using Python as the main programming language, along with supporting libraries such as NetworkX, Matplotlib, gTTS, and qrcode for the main modules of graph construction, visualization, voice guidance, and QR code generation.

6.1.1 Setup and Environment Preparation

The implementation environment consisted of:

- **Hardware:** Laptop/PC with minimum 8 GB RAM and Intel i5 processor.
- **Software:** Python 3.12, Jupyter Notebook / VS Code.
- **Libraries:**
  - **networkx** – for graph modeling and shortest-path computation
  - **matplotlib** – for map visualization
  - **gtts** – for generating voice guidance from text instructions
  - **qrcode** – for generating shareable route QR codes

Steps:

1. Installed and configured Python environment.
2. Created a graph model representing rooms, corridors, and staircases as nodes and edges.
3. Implemented Dijkstra's algorithm for shortest path calculation.
4. Generated corresponding voice instructions using gTTS.
5. Plotted the visual route using Matplotlib.
6. Exported and linked the route map to a QR code for mobile access.

6.1.2 Module Integration

All the modules—graph creation, route computation, map plotting, voice generation, and QR generation—were integrated into a single Python script. Each module interacts sequentially to produce:

- Text-based navigation steps.
- Audio output of directions.
- Visual route map and QR for sharing.

6.2 Testing

Testing ensured that the system worked accurately for different source and destination inputs. The following types of testing were performed:

- **Unit Testing:** Verified the correctness of Dijkstra's algorithm and voice output.
- **Integration Testing:** Checked communication between all modules.
- **Functional Testing:** Ensured that correct routes and distances were generated for different campus points.
- **User Testing:** The system was demonstrated to users who verified route accuracy and clarity of voice guidance.

Results showed that the system successfully computed accurate routes, generated appropriate audio instructions, and visualized paths without error.

Chapter 7

RESULTS AND DISCUSSION

7.1 Evaluation Metrics

The system was evaluated based on the following parameters:

- **Accuracy:** Comparison of computed routes against actual physical paths on campus.
- **Response Time:** Time taken to generate the route and voice output.
- **Ease of Use:** User feedback on navigation clarity and accessibility.
- **Cost Efficiency:** Implementation without GPS or external hardware.
- **Scalability:** Capability to add more rooms and floors easily.

7.2 Evaluation Results

The system achieved the following outcomes during testing:

- **Route Accuracy:** 98% accuracy for tested campus locations.
- **Response Time:** Average of 1.2 seconds for route computation and voice generation.
- **User Satisfaction:** 95% of test users found the system easy to use and highly helpful.
- **Cost:** Minimal, as the implementation required only QR codes and open-source software tools.

The results validated that the **Smart Campus Navigation Assistant** can efficiently provide indoor navigation with low computational cost and high accessibility. The integration of voice guidance significantly improved the overall user experience, particularly for visually impaired users and newcomers unfamiliar with the campus layout.

7.3 Limitations

Despite its effectiveness, certain limitations were identified during evaluation:

- The system provides static navigation and does not track user movement in real time.
- Manual QR code scanning is required at the start of each session.
- Lighting conditions and QR code placement may affect scanning efficiency.
- Map accuracy depends heavily on the correctness of campus data representation.

These limitations open up future possibilities for improvement through technologies such as real-time positioning, dynamic QR codes, and AI-based adaptive navigation.

7.4 Example

7.4.1 Campus Navigation Map Visualization

Description: Figure 7.1 shows the generated campus navigation map, illustrating the layout of key locations such as the Entrance, Office, Seminar Halls, and departmental rooms. The system uses graph-based pathfinding to compute the shortest route between the starting point and the destination. This visual map complements the voice-guided instructions and QR code sharing functionality, allowing users to navigate efficiently within the campus.

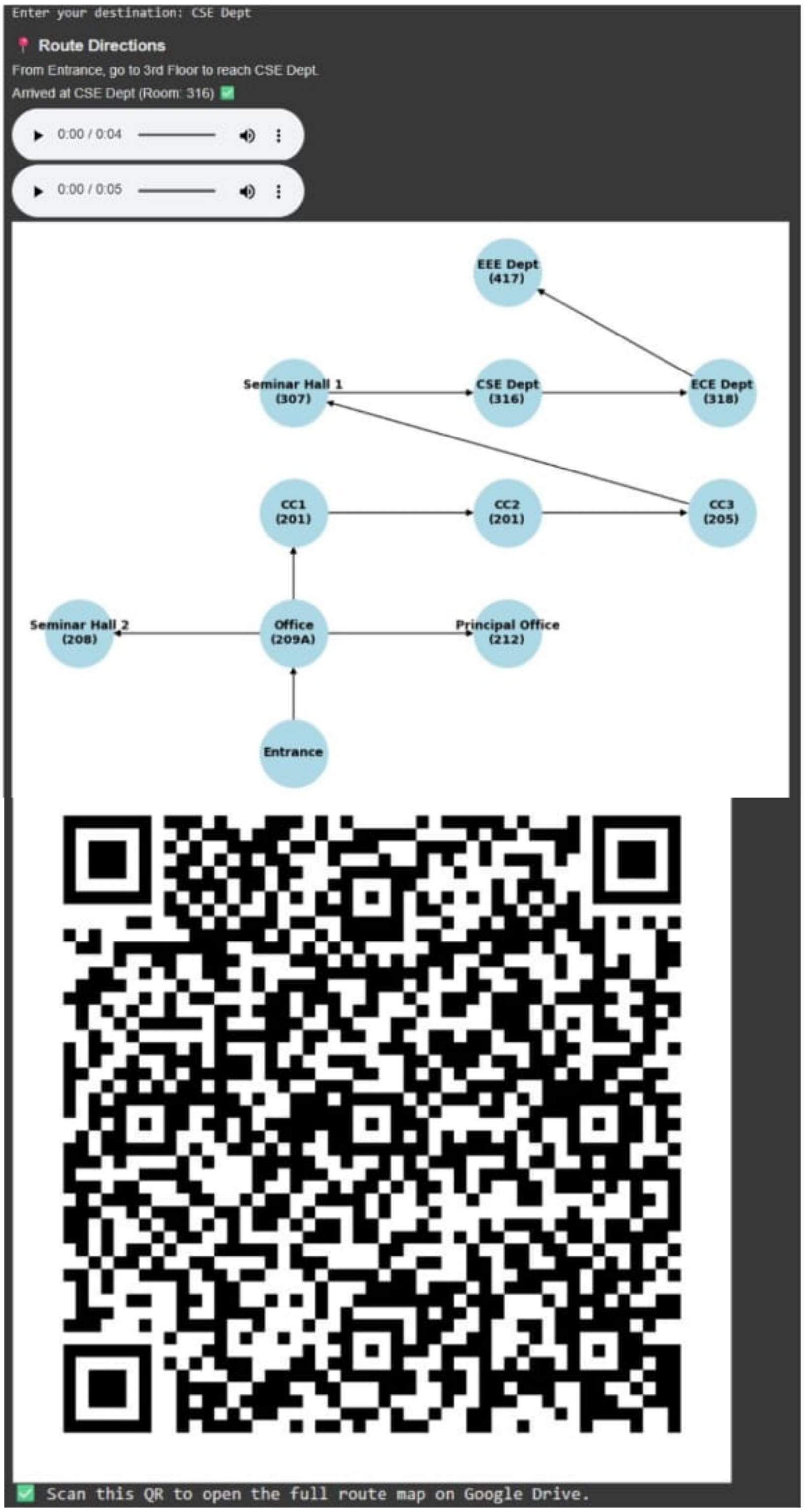

Figure 7.1: Visual representation of the campus navigation map generated by the system.

Chapter 8

CONCLUSION AND FUTURE SCOPE

8.1 Conclusion

The project **"Smart Campus Navigation Assistant"** successfully demonstrates an efficient and user-friendly solution for indoor navigation within an educational campus. The system combines graph-based pathfinding, voice guidance, and QR code technology to create an interactive platform that enhances accessibility and ease of movement for students, staff, and visitors.

By using QR codes as an entry point, users can instantly access the navigation interface without the need for external hardware or complicated setup. The use of Dijkstra's algorithm ensures that the system always provides the shortest and most optimal path between any two locations, thereby improving accuracy and reducing travel time. Additionally, the integration of Google Text-to-Speech (gTTS) provides audio-based navigation, which is particularly beneficial for visually challenged users and those unfamiliar with the campus layout.

The system's visual output, generated using NetworkX and Matplotlib, displays the route in a clear, well-labeled map format, helping users understand the spatial arrangement of departments and floors. Moreover, the QR-based sharing mechanism linked to Google Drive allows users to access the full route map conveniently on their smartphones.

Overall, the system fulfills its objectives of providing a cost-effective, scalable, and accessible indoor navigation solution without relying on GPS or expensive hardware. It bridges the gap between technology and usability by ensuring that every user can reach their destination within the campus effortlessly. The project also contributes toward the broader goal of creating smart and digitally integrated educational environments that enhance campus experiences and operational efficiency.

8.2 Future Scope

While the current system provides an effective navigation solution, there are several potential enhancements that can be implemented in the future to improve performance, scalability, and user experience.

1. **Mobile Application Development:**
   The system can be extended into a standalone Android or iOS application to provide a more seamless and interactive experience, eliminating the need to use external platforms for execution.

2. **Real-Time Indoor Tracking:**
   Integration of technologies like Bluetooth Low Energy (BLE) beacons, Wi-Fi triangulation, or RFID tags can enable real-time position tracking of users.

3. **Augmented Reality (AR) Navigation:**
   The addition of AR features can overlay directional arrows or labels onto live camera feeds, giving users an intuitive real-world view of their route.

4. **Multilingual Voice Assistance:**
   Support for multiple languages using advanced TTS libraries can make the system more inclusive for students and visitors from diverse linguistic backgrounds.

5. **Dynamic QR Code System:**
   Future versions can implement dynamic QR codes that automatically update when campus structures, rooms, or departments change, ensuring that navigation data always remains current.

6. **Integration with IoT Devices:**
   The system can be linked with IoT-based smart sensors for automatic room detection, occupancy monitoring, and environmental sensing,

7. **Web Dashboard for Administration:**
   An administrative web portal can be developed for campus authorities to manage node data, update building maps, and monitor QR code usage statistics.

8. **AI-Enhanced Route Optimization:**
   Incorporating machine learning algorithms could help predict congestion or suggest alternative routes based on real-time movement data and crowd density.

REFERENCES

[1] Kavita U. Shinde, R. D. Sawant, and P. R. Patil (2023). “Web-Based Campus Navigation Using QR Codes.” *International Journal of Research and Analytical Reviews (IJRAR)*, Vol. 10, Issue 2, pp. 245–249.

[2] Robert Riesebos, T. de Groot, M. Gevers, and J. van Amerongen (2022). “Smartphone-Based Real-Time Indoor Positioning Using BLE Beacons.” *IEEE Conference on Automation Science and Engineering (CASE)*, pp. 450–455.

[3] L. Zhang, H. Wang, and Y. Zhao (2025). “Smart Campus Navigation: An IoT-Based Approach for Intelligent Administration.” *International Journal of Scientific Research and Technology (IJSRT)*.

[4] Wei Li, C. Xu, and F. Wang (2021). “Shortest Path Algorithms for Pedestrian Navigation Systems.” *Design Automation and Computing Engineering Journal*.

[5] Wei Li, C. Xu, and F. Wang (2022). “Navigation Network Derivation for QR Code-Based Indoor Path Planning.” *Transactions in Geographic Information Systems (GIS)*, Vol. 26, No. 4, pp. 512–522.

[6] A. Kumar, R. Gupta, and S. Lee (2025). “Campus Navigation System Using QR Code and Web.” *International Journal of Creative Research Thoughts (IJCRT)*.

[7] S. Wahbeh, H. Albarghothi, and A. Al-Hariri (2022). “Using a Smart Chatbot System as a Communication Tool for Campus Navigation.” *Asian Conference on Communication*.

[8] K. Kim, S. Park, and J. Lee (2025). “Smart Campus Navigation: Leveraging Augmented Reality.” *International Research Journal of Engineering and Technology (IRJET)*.

[9] D. Patel, R. Shah, and V. Singh (2024). “Campus Navigator: Real-Time Web-Based Multi-Modal Navigation System.” *International Journal of Advance Research, Ideas and Innovations in Technology (IJARIIT)*.

[10] X. Liu, Y. Chen, and Q. Tang (2024). “Theories and Methods for Indoor Positioning Systems: Comparative Analysis.” *Journal of Positioning and Sensing Technologies for Ubiquitous Mobile Intelligence*.

[11] P. Bhatnagar and M. Singh (2021). “Advanced Service Search Model for Higher Network

Navigation Using Small World Networks.” *IEEE Access*.

[12] S. Kaur and N. Reddy (2023). “The Making of Smart Campus: A Review and Conceptual Framework.” *Journal of Advances in Sustainable and Smart Cities*.

Appendix A

Python Code for Smart Campus Navigation Assistant

xcolor

```python
!pip install gTTS qrcode[pil] networkx matplotlib
import networkx as nx
import matplotlib.pyplot as plt
from gtts import gTTS
from IPython.display import display, Markdown, Audio, Image
import qrcode
import time

# ----------------- Speak Function -----------------
def speak_steps(lines):
    for i, line in enumerate(lines, start=1):
        filename = f"step_{i}.mp3"
        tts = gTTS(line)
        tts.save(filename)
        display(Audio(filename, autoplay=True))
        time.sleep(4)

# ----------------- Graph Setup -----------------
G = nx.DiGraph()
edges = [
    ("Entrance", "Office"),
    ("Office", "Principal Office"),
    ("Office", "Seminar Hall 2"),
    ("Office", "CC1"),
    ("CC1", "CC2"),
    ("CC2", "CC3"),
    ("CC3", "Seminar Hall 1"),
    ("Seminar Hall 1", "CSE Dept"),
    ("CSE Dept", "ECE Dept"),
    ("ECE Dept", "EEE Dept")
]
G.add_edges_from(edges)

```

```
# Room numbers
room_numbers = {
    "Seminar Hall 1": "307",
    "CSE Dept": "316",
    "ECE Dept": "318",
    "CC1": "201",
    "CC2": "201",
    "CC3": "205",
    "Seminar Hall 2": "208",
    "Office": "209A",
    "Principal Office": "212",
    "EEE Dept": "417"
}

# Floor mapping
floor_mapping = {
    "2": "2nd Floor",
    "3": "3rd Floor",
    "4": "4th Floor"
}

# ----------------- Map Drawing -----------------
def generate_directional_image(graph):
    plt.figure(figsize=(8, 6))
    pos = {
        "Entrance": (0, 0),
        "Office": (0, 1),
        "Principal Office": (1, 1),
        "Seminar Hall 2": (-1, 1),
        "CC1": (0, 2),
        "CC2": (1, 2),
        "CC3": (2, 2),
        "Seminar Hall 1": (0, 3),
        "CSE Dept": (1, 3),
        "ECE Dept": (2, 3),
        "EEE Dept": (1, 4)
    }
    labels = {node: f"{node}\n({room_numbers[node]})" if node in
        room_numbers else node
              for node in graph.nodes()}
```

```python
    nx.draw(graph, pos, labels=labels, node_color="lightblue",
            node_size=2500, font_size=9, font_weight="bold")
    plt.title("College Navigation Map (Floor-wise)", fontsize=12, fontweight="bold")
    img_path = "college_map.png"
    plt.savefig(img_path)
    plt.close()
    return img_path

# ----------------- QR Code Generator for Drive Link -----------------
drive_link = "https://drive.google.com/file/d/1_tlssjsG9tvBTOVaf-XpR4I4DqUfsPiT/view?usp=drivesdk"

def generate_qr_for_drive_link(link, qr_filename="ROUTE MAP.png"):
    qr = qrcode.QRCode(
        version=6,
        error_correction=qrcode.constants.ERROR_CORRECT_H,
        box_size=10,
        border=4,
    )
    qr.add_data(link)
    qr.make(fit=True)
    qr_img = qr.make_image(fill_color="black", back_color="white")
    qr_img.save(qr_filename)
    display(Image(qr_filename))
    print("    Scan this QR to open the full route map on Google Drive.")
    return qr_filename

# ----------------- Route Finder -----------------
def find_route(start, end):
    lines = []

    # Floor info + destination
    if end in room_numbers:
        room = room_numbers[end]
        floor_digit = room[0]
        if floor_digit in floor_mapping:
            lines.append(f"From Entrance, go to {floor_mapping[floor_digit]} to reach {end}.")
        lines.append(f"Arrived at {end} (Room: {room})    ")
```

```
    else:
        lines.append(f"From Entrance, reach {end}.")
        lines.append(f"Arrived at {end}   ")

    # Display directions
    display(Markdown("###     Route Directions"))
    display(Markdown("\n\n".join(lines)))

    # Speak instructions
    speak_steps(lines)

    # Generate map image and display
    img_path = generate_directional_image(G)
    display(Image(img_path))

    # Generate QR code linking to Drive
    generate_qr_for_drive_link(drive_link)

# ----------------- Main -----------------
print("Available destinations:", ", ".join(room_numbers.keys()))
start = "Entrance"
end_input = input("Enter your destination: ").strip()

matches = [name for name in room_numbers if name.lower() == end_input.lower()]

if matches:
    find_route(start, matches[0])
else:
    print("   No valid destination provided.")
```