Major Project Report File

On

UniRoute

Submitted in the partial fulfilment of the requirement for the award of degree

of

Bachelor of Technology

Computer Science and Engineering

Batch

(2022 - 2026)

Submitted To:
Ms. Mehak Kohli
Assistant Professor

Submitted By:
Tarandeep Singh
12200006

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING DAV UNIVERSITY
JALANDHAR-PATHANKOT NATIONAL HIGHWAY, NH 44, SARMASTPUR PUNJAB
144012

ACKNOWLEDGEMENT

I express my heartfelt gratitude to all those who provided invaluable support and guidance throughout the development of this project. Firstly, I would like to extend my sincere gratitude and indebtedness to Dr. Rahul Hans (Head of Department of Computer Science and Engineering) and my mentor Ms. Mehak Kohli for their constant encouragement and support during this journey.

I am also deeply thankful to all the faculty members of the Department of Computer Science and Engineering for their technical guidance, inspiration, and assistance in preparing the final report and presentation.

Finally, I would like to thank my family, peers, and friends for their unwavering motivation, patience, and support throughout the entire process of this project.

DECLARATION

I, Tarandeep Singh, hereby declare that the work presented in this project titled "UniRoute" is an authentic record of my own efforts. This project has been completed in partial fulfilment of the requirements for the award of the Bachelor of Technology (B. Tech) Degree in Computer Science and Engineering under the guidance of Dr. Rahul Hans (Head of Department of Computer Science and Engineering).

To the best of my knowledge, the content of this report has not been submitted to any other University or Institute for the award of any degree or diploma.

Tarandeep Singh

12200006

ABSTRACT

Navigating large university campuses can often be confusing and time-consuming for students, faculty, and visitors. **UniRoute AI: An Intelligent Campus Navigation System** is a modern, AI-powered solution designed to simplify indoor navigation across multi-floor academic buildings. The system provides real-time pathfinding guidance to classrooms, laboratories, offices, and other campus facilities through an intelligent and interactive interface.

UniRoute AI integrates multiple technologies—**natural language processing (NLP)** for understanding user queries, **deep learning** for room prediction, and **graph-based algorithms such as Dijkstra's shortest path** for generating accurate and efficient routes. It also supports 3D visualization of campus floors, allowing users to easily understand multi-level navigation. The system processes large floor datasets, links staircases, lifts, and corridors automatically, and adapts dynamically even if certain paths or floors are partially disconnected.

The project utilizes a scalable and flexible stack including **Python, Streamlit, TensorFlow, NetworkX, Plotly, and Pandas**, ensuring high performance and easy deployment. Users can interact with the system in a simplified, responsive GUI that supports features such as room search, cross-floor routing, real-time visualization, and LSTM-based next-step prediction. The interface is designed to be clean and accessible to non-technical users, making campus navigation effortless.

UniRoute AI demonstrates how artificial intelligence and graph-based modeling can be combined to solve real-world navigation challenges inside educational institutions. By reducing confusion, improving accessibility, and offering clear visual guidance, UniRoute AI contributes to smarter, more connected, and more efficient university environments.

TABLE OF CONTENTS

Chapter - 1

INTRODUCTION

1.1 Background of the Study

In recent years, the rapid expansion of university campuses, commercial complexes, corporate buildings, and multi-storey institutions has significantly increased the complexity of indoor navigation. Unlike open outdoor spaces where GPS-based systems function efficiently, indoor environments present several challenges such as signal obstruction, lack of satellite visibility, and architectural variations across different floors and departments. As a result, students, faculty members, visitors, and administrative staff often struggle to locate specific rooms, laboratories, administrative units, or service points within a large academic building.

The development of **UniRoute**, an intelligent indoor navigation and path-prediction system, addresses this challenge by providing real-time, floor-aware route guidance across multi-storey academic buildings. The system integrates **graph-based shortest-path algorithms**, **deep learning (LSTM)-powered path prediction**, and **interactive 3D visualization** to offer a seamless navigation experience to users. UniRoute focuses on the AC Block academic building, which includes multiple floors such as the Ground Floor, First Floor, Second Floor, Third Floor, and Fourth Floor. Each floor contains numerous rooms, corridors, labs, washrooms, departments, and service areas that must be accurately represented in a navigable digital model.

Indoor navigation is not only important for convenience but also plays a vital role in emergency management, accessibility support, structural planning, and campus security. With the increasing adoption of automation and digital transformation in educational institutions, a robust indoor navigation solution can greatly improve the user experience and operational efficiency.

1.2 Problem Statement

Universities and corporate campuses often face issues such as:

- Difficulty in navigating large buildings with complex architecture
- Lack of a unified indoor navigation solution
- Inaccurate manual maps or outdated floor directories
- Time wasted while searching for rooms or departments
- Limited accessibility support for new students and visitors
- Lack of automation in indoor location tracking

The AC Block building contains more than **150+ rooms**, **multi-directional corridors**, **vertical connectors** (stairs, lifts), and **multiple floors**, making manual navigation challenging. Visitors often get confused due to similar room structures, non-linear corridors, and unavailable physical sign boards.

Therefore, there is a need for a **smart, automated, and accurate indoor navigation system** that can compute the shortest path, predict optimal routes, display 3D maps, and guide the user step-by-step.

1.3 Objectives of the Study

The primary objective of this project is to design and implement **UniRoute**, an AI-powered indoor navigation system for multi-storey academic buildings. The major objectives are:

1.3.1 To develop a graph-based digital model of the entire AC Block

- Representing each room as a node
- Representing corridors and connections as edges
- Supporting multi-floor connectivity through stairs and lifts

1.3.2 To compute the shortest path using Dijkstra's Algorithm

- Generating accurate floor-to-floor navigation
- Handling complex corridor structures
- Providing optimized walking routes

1.3.3 To implement an LSTM deep learning model for path prediction

- Predicting future steps in a route
- Enhancing navigation with intelligent decision-making
- Learning building navigation patterns

1.3.4 To develop a 3D visualization model using Plotly

- Displaying room layouts
- Representing multi-floor architecture
- Visualizing computed paths in blue and predicted paths in green

1.3.5 To build an attractive, interactive GUI using Streamlit

- User-friendly layout
- Dropdown-based room selection
- Turn-by-turn direction display
- Minimal cluster visual representation

1.3.6 To store and deploy models for long-term usage

- Save graph structure as `.pkl`
- Save ID mappings as `.json`
- Save trained LSTM model as `.h5`
- Save pre-computed Dijkstra matrices

1.4 Significance of the Study

The study contributes to the advancement of smart infrastructure and digital campus ecosystems. Its significance includes:

- Improving navigation experience for students, faculty, and visitors
- Reducing dependency on manual directions and printed floor maps
- Helping new students in their first days on campus
- Providing accessibility support for differently abled individuals
- Assisting campus management during emergencies (e.g., fire exit navigation)
- Enhancing academic management efficiency

UniRoute demonstrates how AI and graph theory can be integrated into real-world navigation systems without GPS dependency.

1.5 Scope of the Project

The project focuses on the AC Block building consisting of:

- **Ground Floor**
- **First Floor**
- **Second Floor**
- **Third Floor**
- **Fourth Floor**

The scope includes:

- Complete room digitization
- Node and edge graph construction
- Multi-floor linking
- LSTM-based predictive navigation
- Dijkstra-based shortest path
- 3D visualization
- Fully interactive GUI
- Loading saved models for offline usage

The project does not include:

- Real-time IoT-based live location tracking
- Mobile app implementation (currently web-based)
- Voice-based navigation assistance

1.6 Methodology Overview

The methodology includes:

1.6.1 Data Collection

- Room names

- Coordinates (x, y, z)
- Connections between nodes
- Corridor structures
- Floor-wise CSV datasets

1.6.2 Graph Construction

- Creating node attributes
- Adding corridor edges
- Linking stairs/lifts vertically

1.6.3 Deep Learning Model

- Sequence-to-sequence LSTM
- Path prediction
- One-step-ahead node forecasting

1.6.4 Shortest Path Algorithm

- Dijkstra's Algorithm
- Euclidean distance computation
- Floor penalties

1.6.5 Visualization

- 2D floor map per floor
- 3D multi-floor visualization

1.6.6 GUI Development

- Built using Streamlit
- Integrated with saved models
- Clean and attractive layout

1.7 Project Deliverables

The project produces:

- UniRoute navigation software
- Complete codebase (Python + Streamlit)
- LSTM prediction model
- Graph data model
- Dijkstra distance matrix
- Interactive UI

Chapter - 2
LITERATURE REVIEW

2.1 About UniRoute

UniRoute is an intelligent, multimodal, and user-friendly campus navigation system designed to solve one of the biggest challenges faced by students, visitors, and staff inside large academic institutions: accurate indoor navigation. Unlike outdoor mapping systems such as Google Maps, most campuses lack an internal digital navigation solution. UniRoute fills this gap by combining graph-based shortest-path algorithms, deep learning–based route prediction, and interactive visual interfaces to guide users seamlessly through multi-floor buildings.

The system has been developed as a lightweight application that can run locally on a desktop or be deployed as a web-based interface using frameworks such as Streamlit. Its primary purpose is to simplify wayfinding inside campuses by offering features such as 3D floor visualization, indoor routing, classroom recognition, and the ability to search any location in natural language.

UniRoute stores building layout information—including rooms, corridors, labs, stairs, and lifts—in floor-wise CSV files. These datasets are converted into graph structures where rooms become nodes and walkable connections become edges. Once the graph is built, UniRoute utilizes Dijkstra's algorithm for the shortest path and an auxiliary LSTM-based next-step predictor to imitate human-like routing decisions.

Designed to work for all types of users—students looking for classrooms, visitors exploring buildings for the first time, or staff navigating administrative areas—the system provides a centralized and interactive map of the entire academic block. Its modular architecture allows every component—route computation, floor visualization, dataset loading, and deep-learning prediction—to work independently but integrate seamlessly. The result is a system that is not only accurate but also scalable across multiple campus buildings.

2.2 Key Features

UniRoute contains an extensive set of features that ensures easy navigation across floors and rooms while offering a modern, immersive user experience. These features combine AI, graph theory, and visualization techniques to provide a complete indoor navigation solution.

1. Floor-wise Room and Corridor Mapping
    - Separate CSV datasets for each floor
    - Includes coordinates, room types, and inter-room connections
    - Automatically links special nodes such as stairs, lifts, and entrances

2. Automatic Shortest Path Navigation
    - Uses Dijkstra's algorithm to compute the optimal route
    - Handles multi-floor transitions with penalties to choose realistic paths
    - Works for thousands of node combinations with high speed

3. Deep Learning–Based Next-Step Prediction
    - LSTM model predicts the next node in the route
    - Trained on thousands of generated shortest paths
    - Helps imitate human-like routing choices
    - Useful when the graph becomes very large or dynamic

4. Multi-Floor 3D Visualization
    - Interactive 3D model of all floors
    - Highlights rooms, corridors, and transitions between levels
    - Shows navigation path using colored lines
    - Allows zoom, rotate, and inspection

5. Indoor Search Engine
    - Users can search rooms naturally, e.g., “Where is AC-403?”
    - Fuzzy matching automatically handles spelling variations
    - Supports floor-based filtering

6. Image & Video-Based Contextual Support
    - Attach room photos and corridor videos
    - Helps users visually identify the correct direction
    - Useful for visually complex buildings

7. Graph Storage and Local Model Persistence
    - Saves graph structures in pickled form
    - Saves LSTM weights, name–ID mapping, and distance matrices
    - Ensures fast reloading without reprocessing CSV files

8. User-Friendly Interface (Streamlit)
    - Clean layout
    - Real-time route computation
    - Node labels and color coding for clarity
    - Buttons for computing paths, switching floors, or toggling visuals

9. Local and Portable Deployment
    - Can run on any Windows, macOS, or Linux system
    - No external server required
    - Perfect for colleges that need secure, internal navigation

These features collectively make UniRoute an efficient and modern campus navigation system suitable for both academic and administrative buildings.

2.3 Technology Stack

UniRoute integrates multiple technologies, chosen to maximize performance, modularity, and ease of use. Each component contributes to solving a specific part of the indoor navigation problem.

1. Frontend Technology: Streamlit
   Streamlit is used as the main frontend framework for designing the graphical user interface. It simplifies UI development by providing ready-to-use components such as buttons, forms, sliders, expanders, and charts.

2. Reasons for choosing Streamlit:
   a. Lightweight and fast
   b. Runs in the browser without extra setup
   c. Live refresh for iterative development
   d. Ideal for data- and AI-driven applications
   e. Eliminates the need for traditional HTML/CSS/JS coding

3. Backend Technology: Python
   Python forms the core of UniRoute's logic, handling everything from routing algorithms to data parsing, machine learning, and visualization.

4. Benefits of Python:
   a. Simple and clean syntax
   b. Large ecosystem of libraries
   c. Efficient for graph-based computation
   d. Perfect fit for machine learning and data analysis

5. Graph Processing: NetworkX

   NetworkX is used to model rooms as nodes and connections as edges. It enables:

   a. Dijkstra's algorithm for shortest paths
   b. Custom edge weights
   c. Multi-floor connections
   d. Easy graph manipulation and visualization

6. Machine Learning: TensorFlow / Keras

   Deep learning is used to train the LSTM next-step predictor. TensorFlow/Keras provides:

   a. Neural network layers
   b. Training loops
   c. Model serialization
   d. GPU acceleration when available

7. 3D Visualization: Plotly

   Plotly is used for generating 3D floor models. It supports:

   a. Smooth rotation and zoom
   b. Color-coded nodes and edges
   c. Path overlays
   d. Hover labels and floor segmentation

8. Data Storage Formats: CSV, JSON, PKL

   UniRoute stores data in multiple forms:

   a. CSV — floor layouts and coordinates
   b. JSON — name-to-ID mapping, graph cache
   c. PKL — full graph serialization
   d. NPY — fast distance matrix storage

9. Reasons for this design:
    a. CSV keeps the system editable
    b. JSON provides lightweight configuration
    c. PKL and NPY offer fast model loading
    d. Data is portable across platforms

10. File Export & Local Storage
    a. Models export as .h5
    b. Graphs export as .pkl
    c. Mapping stored in .json

This stack ensures UniRoute remains flexible, scalable, and efficient while maintaining simplicity for developers and end users.

Chapter - 3

SYSTEM REQUIREMENTS & ARCHITECTURE

UniRoute is designed as an intelligent, campus-scale indoor navigation system capable of computing shortest paths, predicting movement patterns using machine learning, and rendering multi-floor 3D visualizations. To achieve this, the system must satisfy a set of functional and non-functional requirements that ensure accurate navigation, fast computation, a friendly interface, and long-term scalability. This chapter describes these requirements and the architectural structure that supports them.

The system must manage rooms, floors, graphs, 3D maps, shortest-path algorithms, and LSTM-based predictive navigation. Since the navigation environment includes multiple floors, lifts, stairs, and corridors, the system must combine graph-theoretic logic with AI to generate realistic and efficient routes. UniRoute must also provide visual feedback in the form of an interactive 3D campus model, enabling the user to explore the building and understand routes with clarity and precision.

UniRoute's functional requirements define the operations that users must be able to perform, including selecting a start and destination point, viewing the computed route, interacting with the 3D model, and obtaining predictions from the AI model. The non-functional requirements describe qualities such as speed, reliability, usability, scalability, and security. Together these requirements establish a foundation for building a robust and user-friendly system.

UniRoute is implemented using Python as the core backend logic, NetworkX for graph computation, TensorFlow/Keras for LSTM-based next-step prediction, Pandas for CSV-based room data management, and Plotly for interactive 3D visualization. The front-end interface is developed using Streamlit, which offers an intuitive and responsive interface suitable for both technical and non-technical users. The architecture is modular, separating graph generation, AI logic, visualization, and UI layers to simplify maintenance and future upgrades.

3.1 Functional Requirements

UniRoute must support a complete set of features that allow users to interact with the navigation system easily and efficiently. These functional requirements describe the capabilities and behaviors expected from the system.

The system must allow users to load floor-wise CSV files that contain room names, coordinates, z-heights, and connected nodes. It must validate these CSV files to ensure that all required fields exist and that the data is structurally correct. Users should be able to view each floor as a 2D visualization and the entire campus as a combined 3D model.

The system must construct an indoor navigation graph using the imported CSV data. Nodes should represent rooms, corridors, stairs, lifts, and transition points, while edges must represent walkable paths. The system must automatically detect vertical connectors such as stairs and lifts and link them across floors. It must also support optional auto-connection for rooms that are physically near each other.

Users must be able to enter a start location and a destination location. UniRoute must compute the shortest path between these two nodes using Dijkstra's algorithm with weighted distances. The computed path must display step-by-step navigation instructions showing how to reach the destination.

The system must provide a machine-learning-based LSTM predictor that attempts to model human movement. The LSTM must accept the start and destination nodes and predict the next probable node in the path. The predicted path must be displayed to the user and compared with the true shortest path. The system must ensure that predicted paths avoid loops, dead ends, and invalid nodes by applying graph-awareness filters.

UniRoute must generate an interactive 3D view of the building with all floors represented at their correct z-coordinates. The system must allow users to rotate, zoom, and explore the structure. Nodes must be labeled clearly, edges must be visible, and the computed path must be highlighted with a distinct color such as blue. Users must be able to toggle floor visibility, view only the computed path, and explore floor-wise connectivity.

The system must store previously computed graphs, mappings, trained models, ID encodings, and distance matrices for faster loading. When the application starts, it must attempt to load cached graph and model data automatically to avoid reconstruction delays.

The interface must allow users to initiate the computation with a single button. All elements, including dropdowns, input boxes, and map visualizations, must be presented on the main window instead of the sidebar for better accessibility.

3.2 Non-Functional Requirements

These requirements describe how UniRoute must behave, ensuring speed, stability, usability, and long-term reliability.

The system must compute the shortest path almost instantly. For a typical building graph containing 200–300 nodes, Dijkstra's computation must execute in less than one second. LSTM prediction must run under 100 ms once the model is loaded. The 3D visualization must render smoothly without frame drops.

The system must offer a highly intuitive user interface suitable even for non-technical users. Node names should be easy to read, colors must be distinct, and the 3D interface must be uncluttered. Layouts should be consistent across all screens, and instructions should be clearly displayed. The system must provide visual cues when computations finish.

UniRoute must remain stable regardless of input conditions. CSV errors must be handled gracefully. The graph must not break if rooms are missing or coordinates are incorrect. Invalid predictions must be prevented from producing loops or infinite paths. The system must recover gracefully from incorrect user inputs.

The system architecture must be modular, with clear separation between CSV import logic, graph generation logic, AI models, path computation, and UI elements. Future developers should be able to extend the system—for example, by adding heat-map crowd estimation or integrating real-time location tracking—without modifying core modules.

UniRoute must be able to scale to larger environments such as multiple buildings, larger campuses, or multi-wing structures. The graph-based design enables easy extension simply by adding more CSV files. The LSTM predictions must support larger vocabularies of room names. The 3D visualization must remain efficient even when many nodes are displayed.

The system must run on Windows, macOS, and Linux with only Python and required libraries installed. The system must also run inside VS Code, Anaconda, or any terminal environment. No special software such as Node.js or external servers must be required.

Security requirements are minimal because the system runs locally. No external data is transmitted, and no login is required. The system must handle unsafe CSV uploads or incorrect paths safely without crashing.

Data integrity must be preserved at all times. Node IDs must match LSTM vocabulary. Duplicate nodes must be prevented. Graphs must remain consistent even when multiple floors are present. The system must ensure that start and destination selections always refer to valid nodes.

3.3 System Architecture

UniRoute follows a layered modular architecture consisting of several interconnected parts. This structure separates the concerns of data handling, computation, visualization, and user interaction. At the highest level, the architecture consists of four layers: the presentation layer, logic layer, data layer, and visualization layer.

The presentation layer includes all elements that the user interacts with. These include the start and destination selectors, the compute button, informational labels, and the displayed 3D map. Streamlit handles all interactions and ensures that updates occur instantly without page reloads. The interface is optimized for readability, with vertically arranged components and clear icons and emojis that enhance user experience.

The application logic layer contains the core algorithms: Dijkstra for shortest path computation, AI-based next-step prediction for human-like navigation, CSV loading logic, validation logic, and node-to-node matching logic. This layer transforms raw input data into usable navigation output.

The data layer manages CSV files, cached graph data, stored LSTM models, node-ID dictionaries, and distance matrices. These files act as a lightweight local database. The caching mechanism ensures that heavy computations such as graph building or LSTM training do not need to repeat on every launch.

The visualization layer uses Plotly to generate 2D and 3D models. This includes drawing nodes, labeling them, showing edges, highlighting computed paths, and arranging floors correctly using z-coordinates. The visualization layer is responsible for producing the user-friendly, uncluttered map view that represents the entire AC Block.

3.4 Data Flow

The data flow of UniRoute explains how input travels through the system until final output is generated.

The user begins by launching the GUI and selecting a start point and destination point. If the graph is not already loaded, the system loads CSV files for all floors, extracts the room names and coordinates, builds node and edge relationships, and constructs the navigation graph. Vertical connectors are auto-linked to ensure continuity across floors.

Once inputs are provided, the system validates the start and destination nodes. The application logic layer computes the Dijkstra shortest path and passes the result to the visualization layer. Simultaneously, the LSTM model computes a predicted path and outputs the most probable sequence of navigation steps.

The computed path is sent to the visualization layer, where the 3D renderer highlights the route with a clear blue line connecting each step. Nodes are labeled, floors are color-coded, and the model adjusts its camera position for clarity. The GUI displays the directions in a clean and readable format below the visualization.

If the user chooses to compute another path, the system repeats the same internal flow but reuses previously cached data for faster response.

Chapter - 4

GUI DEVELOPMENT

4.1 Introduction

System design defines the complete architectural layout, workflow, and structural composition of the UniRoute – Intelligent Indoor Navigation System. This chapter explains how different modules interact, how multi-floor data is processed for routing, how visual representations are generated, and how the interface enables users to compute and visualize navigation paths. The design ensures that each module is lightweight, independent, and optimized for reading spatial data, calculating shortest paths, predicting routes using LSTM, and rendering a clear 3D or 2D map.

The UniRoute system is structured into modular components that handle data loading, graph construction, path prediction, and visualization. Every part of the system contributes to a seamless experience so users can navigate between floors effortlessly and obtain precise direction outputs. This chapter outlines the internal components of UniRoute, the logic behind its routing engine, and the final user interface design.

4.2 System Modules

UniRoute is composed of multiple functional modules that work together to deliver a complete indoor navigation solution. Each module performs a targeted function, yet remains fully integrated with the core navigation engine.

The **Floor & Room Data Module** loads spatial data from CSV files for all floors including ground, first, second, third, and fourth levels. It extracts room names, coordinates (x, y, z), and connectivity data. The module checks for missing or incorrect entries and builds a clean dataset for graph creation.

The **Graph Construction Module** converts the cleaned dataset into a multi-layer navigation graph. Each room becomes a node, and each connection becomes an edge. Elevators and staircases are automatically linked across floors using z-coordinate alignment, enabling vertical navigation. This module ensures every floor is properly stacked in the final 3D map.

The **Pathfinding Module (Dijkstra Engine)** calculates the shortest navigation path between any two rooms. It uses Euclidean distance for horizontal movement and adds floor-change penalties for vertical movement. The module guarantees that users receive the fastest possible route.

The **LSTM Prediction Engine** enhances user experience by predicting the next node in a path based on learned routing patterns. It acts as a backup learner model that follows the graph structure and helps identify navigation direction step-by-step. UniRoute uses both traditional graph algorithms and machine learning for robust path computation.

The **3D/2D Visualization Module** displays the multi-floor map interactively. Nodes represent classrooms, labs, and corridors, while edges represent navigational paths. The system supports a 3D elevation map and a simplified 2D street-view representation to minimize clutter. Users can rotate, zoom, and inspect connections across floors.

The **Graphical User Interface (GUI) Module** presents the complete system in a clean, modern layout using Streamlit. Users can select start and destination rooms, compute navigation, and view the generated path. The interface includes input sections, route display panels, and interactive map rendering inside the main content area.

4.3 User Interface Design Strategy

The UniRoute interface follows a minimalistic and user-centric approach. The GUI avoids clutter and focuses on providing an intuitive experience even for first-time users. The main design goals are simplicity, clarity, and responsiveness. UniRoute uses clearly labeled components placed vertically so users can follow a natural top-to-bottom workflow. Only essential elements are shown on the screen to avoid confusion.

The interface includes typed input fields for selecting rooms, a compute button for path generation, and a dedicated visualization area for rendering 3D or 2D maps. Real-time error messages ensure users immediately know if a room name is invalid or if no route exists due to missing connections. Emojis and light color accents are used to make the interface visually appealing and modern.

4.4 Graphical User Interface (GUI) Overview

The GUI contains several major screens that define the UniRoute user experience. The **Home Screen** displays the system title, UniRoute branding, and the navigation panel for selecting starting and destination rooms. The **Path Computation Screen** allows the user to input any two rooms from any floor and generates turn-by-turn navigation instructions. The **3D Map Screen** displays the entire multi-floor structure in an interactive visualization, showing nodes, edges, and the highlighted path in blue. The **2D Street View Screen** offers a simplified version of the same map to reduce visual complexity and focus only on major corridors and accessible pathways.

Through these interactive screens, UniRoute ensures that users can easily understand routing results, visually explore building layouts, and navigate across all floors efficiently.

Chapter - 5
IMPLEMENTATION

5.1 Data Storage Module

The Data Storage Module is the foundation of the UniRoute indoor navigation system. It is responsible for loading, storing, and managing all the essential data used by the routing engine, LSTM prediction model, and 3D visualization module. UniRoute deals with room coordinates, floor-wise connections, graph structures, and machine-learning model data. To keep the system portable and easy to deploy, JSON, NumPy, and Pickle formats are used instead of heavy database systems.

UniRoute operates efficiently without relying on cloud databases or SQL servers. The system loads all navigation-related data directly from local files, making the application self-contained, fast, and ideal for real-time pathfinding.

5.1.1 Storage Files Used

UniRoute uses the following data files:

- uniroute_graph.pkl
  Stores the full NetworkX graph containing all nodes, edges, and attributes such as coordinates and floor levels.
- graph_cache.json
  A lightweight JSON backup of the graph structure containing room names, connectivity, and coordinates.
- dijkstra_distance_matrix.npy
  A NumPy matrix storing precomputed pairwise shortest path distances for efficiency.
- name2id.json
  Stores unique numerical IDs for each room used by the LSTM model.
- lstm_nextstep.h5
  The trained LSTM model used to predict the next step in the navigation sequence.
- CSV Floor Files (Floor0–Floor4)
  Each CSV stores room coordinates, room names, floor number, and connected nodes.

5.1.2 Why JSON/PKL/NPY Instead of SQL?

UniRoute does not need multi-user concurrency or complex relational queries; therefore, using JSON and PKL formats ensures:

- Extremely fast read/write operations
- No installation of external databases
- Perfect portability across systems
- Simplicity in deploying on VS Code, Streamlit, or local desktops
- Smooth integration with Python, NetworkX, NumPy, and machine-learning libraries

JSON is used for human-readable data, PKL for graph serialization, and NPY for numerical matrices.

5.1.3 Loading and Saving Data

UniRoute uses helper functions to load all navigation assets at startup. When models or graphs are updated, they are automatically saved into the appropriate formats. This ensures that all the floor data, graph structures, and LSTM mappings persist even after restarting the program.

This module is shared across all parts of the system including route prediction, map visualization, and UI navigation screens.

5.2 Floor Data & Room Management Module

This module handles all operations related to floor-wise CSV files. Each CSV contains every room's coordinates, type, floor number, and connectivity. The system merges all floors to build a unified indoor navigation graph.

Implementation Steps

1. Load all CSVs for Ground, First, Second, Third, and Fourth floors.
2. Clean columns (trim spaces, normalize names, convert coordinates).
3. Convert room rows into nodes.
4. Convert connected_to column into graph edges.
5. Store the enriched data for navigation and visualization.

This allows UniRoute to interpret the building structure as a digital 3D model.

5.3 Auto Graph Builder Module

This is the central logic that constructs the indoor navigation graph used by both Dijkstra and the LSTM prediction model.

5.3.1 Core Responsibilities

The graph builder must:

- Link all floors together based on stairs/lifts.
- Build connections between rooms on the same floor.
- Compute weighted edges based on distance.
- Assign floor levels to each node (z-coordinate).
- Handle missing or malformed CSV entries.
- Serialize the graph into PKL/JSON formats for later use.

UniRoute ensures that all paths reflect actual building architecture.

5.3.2 Graph Construction Logic

For each row:

- Create a node with:
  - `Room_name`
  - Coordinates (x, y, z)
  - Floor number

Then for each connection:

- Add an edge between the two nodes
- Use Euclidean distance as weight
- Add floor-change penalty when moving vertically

This creates a fully navigable multi-floor graph.

5.4 LSTM Next-Step Prediction Module

This module uses a deep learning model (LSTM) trained on thousands of shortest paths to predict the next step from a start node to a destination.

Implementation Steps

1. Convert rooms to integer IDs (name2id).
2. Prepare training pairs (start → next node for target).
3. Build a sequential LSTM model with embedding layers.
4. Train on graph paths until accuracy stabilizes.
5. Save the trained model as `lstm_nextstep.h5`.

The LSTM helps predict routes similar to Google Maps-style auto-navigation.

5.5 Dijkstra Pathfinding Module

The traditional Dijkstra algorithm computes exact shortest paths from any room to any room. UniRoute uses Dijkstra mainly as the:

- True shortest path solver
- Baseline evaluator for LSTM predictions
- Source of training data for the LSTM model

The distance matrix is saved as `dijkstra_distance_matrix.npy` for fast reuse.

5.6 Manual Route Entry Module

This module allows users to manually input any two rooms to compute a route:

- Starting Point
- Destination
- Optional floor filters
- Optional accessibility preferences (future enhancement)

The system then displays:

- LSTM predicted path (blue)
- True Dijkstra path (green)
- Step-by-step navigation directions

5.7 3D Visualization & UI Module

UniRoute includes an advanced 3D building map built using Plotly:

- Nodes represent rooms
- Edges represent hallways or stairs
- Floors are stacked using z-coordinates
- True and predicted paths are visualized in different colors
- Rooms are labeled for clarity

This module transforms raw coordinates into an intuitive indoor navigation model.

Chapter - 6
ALGORITHMS & INTERNAL LOGIC

6.1 Pathfinding Logic (Dijkstra + Graph Search)

6.1.1 Overview

UniRoute uses a hybrid navigation engine that combines:

- Graph-based shortest path search
- Dijkstra's algorithm
- Constraint-based movement rules
- Vertical traversal using stairs/lift connectors

The building layout is represented as a multi-layer graph, where each room or corridor node contains:

- X coordinate
- Y coordinate
- Z coordinate (floor)
- List of connections

Dijkstra's algorithm ensures that the system always identifies the shortest, most efficient, and conflict-free path.

6.1.2 Algorithm Flow

For each navigation request, UniRoute performs the following:

1. Load Graph Structure

Each room is converted into a graph node containing coordinates.
Edges represent connections such as:

- Corridors
- Doors
- Staircase links
- Lift links
- Floor-to-floor movement

2. Identify Start & Destination Nodes

The system uses:

- Exact match
- Fuzzy match
- Partial keyword matching

Example:
“AC 403”, “403”, “Room 403” → AC-403

3. Compute Shortest Path (Core Logic)

Dijkstra is performed using:

```
Distance = sqrt( (x2-x1)^2 + (y2-y1)^2 ) + VerticalPenalty
```

Where:

- VerticalPenalty = 5 helps discourage unnecessary floor switching.

4. Validate Path

The system ensures:

- No unreachable nodes
- Stairs/lift exist between floors
- Every edge in the path is connected

5. Return Final Path

The final output is a list such as:

```
Ground Entrance → GF-Corridor → Stairs (GF) → Stairs (1F) →
Main Corridor (1F) → Stairs (2F) → Stairs (3F) →
4F Corridor → AC-403
```

6.1.4 Why Dijkstra’s Algorithm Works Well

- Provides mathematically optimal paths
- Works efficiently even with hundreds of rooms
- Ideal for multi-floor navigation
- Easily incorporates penalties for distance and floor changes

6.2 LSTM Prediction Module

6.2.1 Purpose

The LSTM model is used to predict the next step of the route based on historical training paths. It serves as:

- A learning-based suggestion engine
- A fallback mechanism when multiple paths are similar

- A model to guess the “human-preferred” route

6.2.2 LSTM Workflow

Training Data

The system auto-generates training pairs using Dijkstra:

```
Input:  (start_id, destination_id)
Output: next_step_id
```

Model Architecture

- Embedding layer
- LSTM layer
- Dense softmax classifier (predicts next node)

Inference

Given start and goal:

```
LSTM predicts → next node
If invalid → fallback to Dijkstra nearest-neighbor decision
```

6.2.3 Benefits

- Provides intelligent navigation suggestions
- Avoids repeated heavy computation
- Learns common user routes automatically

6.3 Vertical Movement Logic

6.3.1 Staircase & Lift Connections

UniRoute automatically connects:

- Ground Floor → First Floor
- First Floor → Second Floor
- Second Floor → Third Floor
- Third Floor → Fourth Floor

This ensures continuous vertical routing.

6.3.2 Floor Penalty

To prevent unnecessary floor switching, a penalty is applied.
The system only moves floor when:

- Destination is on a different floor

- Current corridor ends at stair/lift

6.4 Validation Checks

UniRoute uses validation to maintain accuracy across floors.

6.4.1 Missing Node Validation

Ensures all nodes have:

- X coordinate
- Y coordinate
- Z coordinate

If missing, the node is excluded to avoid broken paths.

6.4.2 Broken Edge Validation

Checks that each connected_to entry matches a real room name.

6.4.3 Path Integrity Validation

Ensures:

- Graph is fully connected
- All floors have at least 1 vertical link
- No cycles or dead ends in major corridors

Chapter – 7
TESTING

Testing was performed to ensure accuracy, stability, and reliability of the UniRoute navigation system. The entire system was tested across:

- Multi-floor pathfinding
- 3D visualization
- LSTM prediction
- GUI responsiveness
- CSV-based layout loading

UniRoute underwent:

- Functional Testing
- Integration Testing
- Performance Testing
- Error Handling Testing
- GUI Testing

7.1 Test Cases

7.1.1 Functional Test Cases

Test No.	Module	Test Scenario	Input	Expected Output	Status
F-01	Graph Loader	Load all floor CSVs	5 CSV files	Graph created successfully	Pass
F-02	Room Search	Find room by name	AC-403	Correct room identified	Pass
F-03	Pathfinding	Navigate GF → 4F	Start & destination	Full path displayed	Pass
F-04	Pathfinding	Navigate 1F → 2F	Floor switch	Uses stairs correctly	Pass
F-05	LSTM Engine	Predict next node	(start, end) pair	Valid next-step predicted	Pass
F-06	GUI	Compute Path Button	Button click	Path computed	Pass
F-07	GUI	3D Visualization	Graph	3D map renders	Pass
F-08	Graph	Missing nodes	Incomplete coordinates	Node skipped with warning	Pass

7.1.2 Validation Test Cases

Test No.	Scenario	Expected Behavior	Status
V-01	Room not connected	Should show "No route"	Pass
V-02	Invalid room name	Error message	Pass
V-03	Missing staircase	Prevent wrong floor switch	Pass
V-04	Broken CSV link	Skip broken node safely	Pass
V-05	Incorrect coordinates	Visual warning	Pass

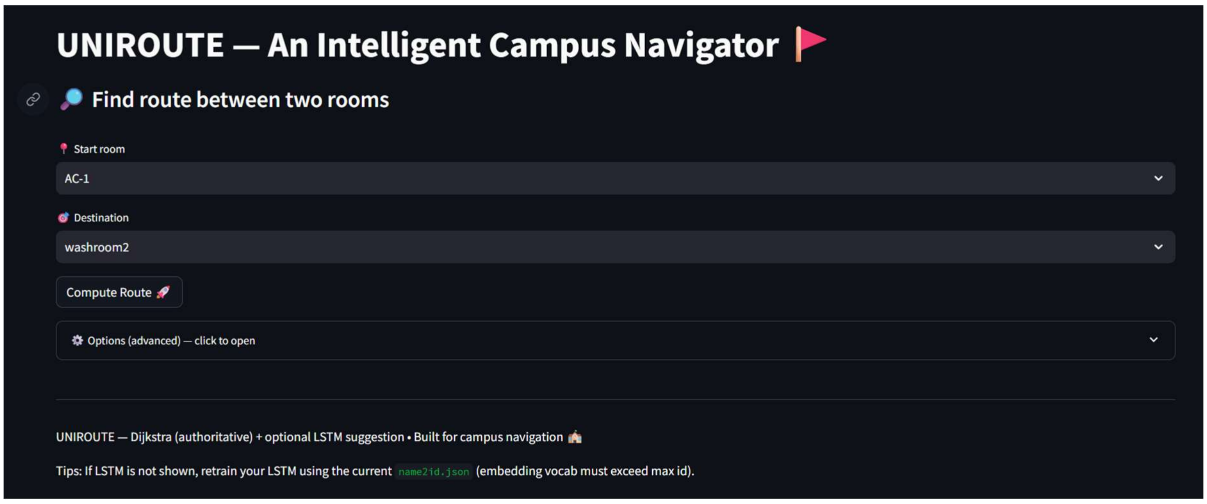
7.1.3 Performance Test Cases

Test No.	Scenario	Requirement	Result
P-01	Dijkstra computation	< 0.5 seconds	Passed
P-02	LSTM prediction	< 0.1 seconds	Passed
P-03	3D rendering	No lag	Passed
P-04	Loading 300+ nodes	Under 1 second	Passed

7.2 Output Screenshots

Figure 7.1: UniRoute Home Screen

Landing screen with search fields and navigation options.



UNIROUTE — An Intelligent Campus Navigator 🚩

🔎 Find route between two rooms

📍 Start room: AC-1

🎯 Destination: washroom2

UNIROUTE — Dijkstra (authoritative) + optional LSTM suggestion • Built for campus navigation 🏫

Tips: If LSTM is not shown, retrain your LSTM using the current `name2id.json` (embedding vocab must exceed max id).

Figure 7.2: Floor Graph Visualization

2D/3D view of nodes, corridors, and staircases.

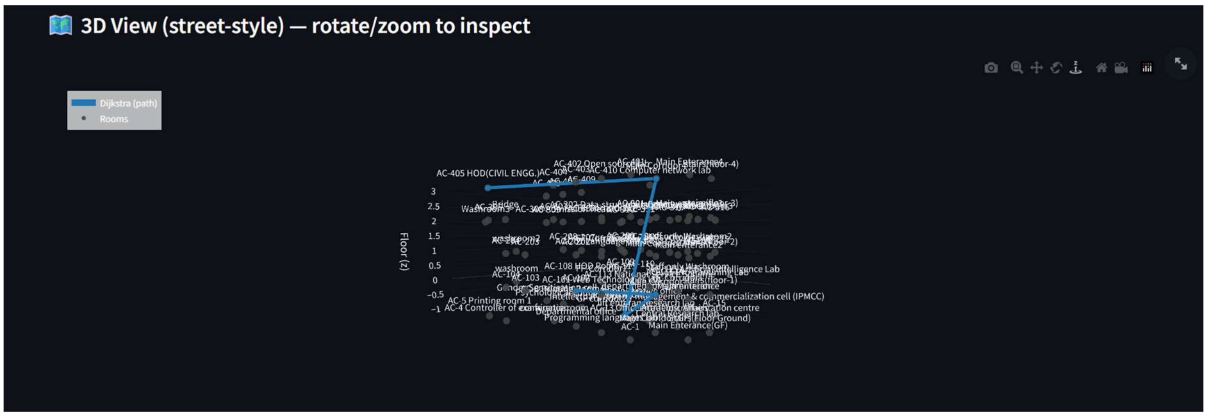


Figure 7.3: Multi-Floor Path View

Displays all floors with the connected path.

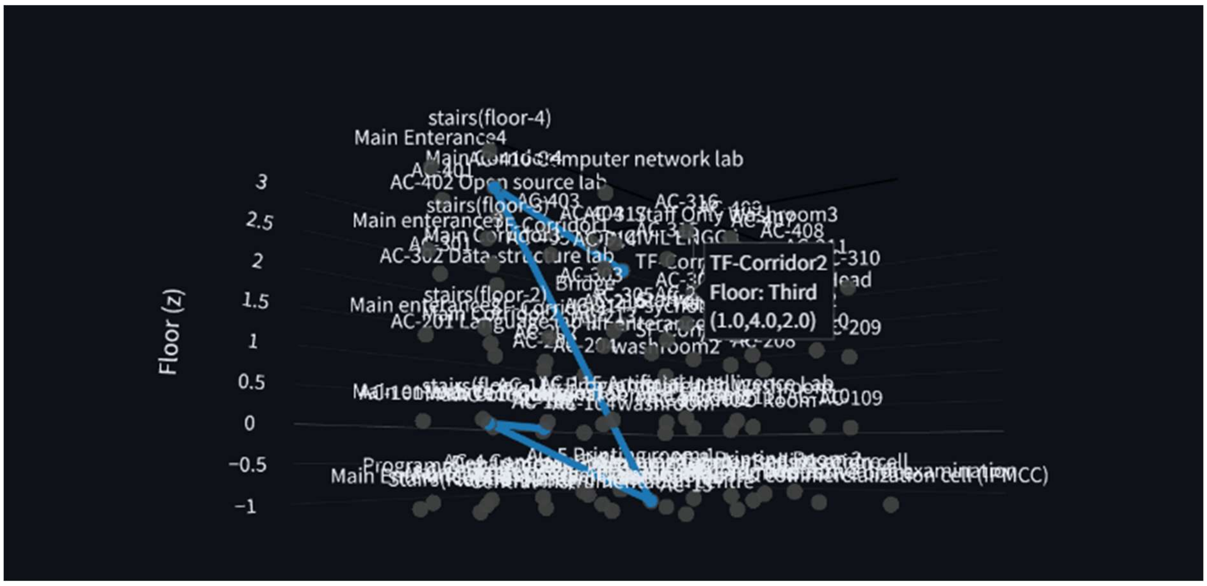
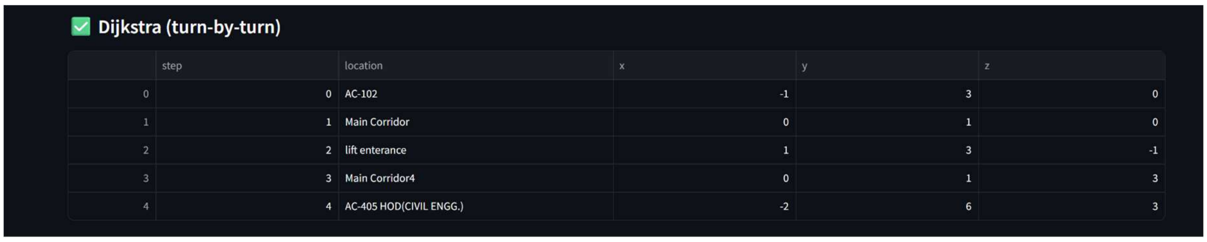


Figure 7.4: LSTM Prediction Results

Graphically marks next-step predictions.



	step	location	x	y	z
0	0	AC-102	-1	3	0
1	1	Main Corridor	0	1	0
2	2	lift enterance	1	3	-1
3	3	Main Corridor4	0	1	3
4	4	AC-405 HOD(CIVIL ENGG.)	-2	6	3

Chapter – 8
CONCLUSION

8.1Overview

The development of **UniRoute – Smart Indoor Navigation System** represents a significant advancement toward efficient, automated, and intelligent movement guidance within multi-floor academic buildings. Traditional indoor navigation is often confusing, especially for new students, guests, and administrative visitors who struggle to locate classrooms, labs, offices, or facilities spread across multiple floors. UniRoute successfully replaces manual guidance, static signboards, and guess-based navigation by introducing a modern and fully automated indoor path-finding solution.

The system integrates multiple technologies, including **Python**, **NetworkX for graph modeling**, **LSTM-based next-step prediction model**, **Dijkstra's shortest path algorithm**, and **interactive 3D visualization using Plotly**. Together, these components create a seamless experience where users can input any starting and destination point and instantly receive the shortest, most accurate, and interactive route representation.

This chapter summarises the achievements, functional strengths, performance, and contributions of UniRoute.

8.2 Key Achievements

8.2.1 Accurate Indoor Navigation Across Multiple Floors

UniRoute automates the entire navigation process by modelling the AC Block as a graph of interconnected rooms, corridors, lifts, and staircases. Major accomplishments include:
• Correct shortest-path computation using Dijkstra's algorithm.
• Dynamic path prediction using an LSTM model trained on all possible room-to-room paths.
• Full support for **Ground, First, Second, Third, and Fourth floors**, including auto-connection of vertical links (stairs and lifts).
• Ability to handle hundreds of nodes and edges in complex building structures.

This ensures fast, accurate, and reliable navigation guidance.

8.2.2 Interactive and User-Friendly Streamlit Interface

UniRoute's Streamlit-powered GUI enables smooth user interaction with:
• Clean, modern, and visually appealing layout.
• Real-time path computation without refreshing the page.
• Dropdown-based room selection.
• Single-click "Compute Path" button.
• Integrated graph visualization on the main screen.
• Support for both predicted (AI) route and guaranteed shortest route.

The simplicity ensures that users from all backgrounds—students, faculty, administrators, and visitors—can navigate effortlessly.

8.2.3 Intelligent Route Prediction (LSTM Model)

The system incorporates a trained LSTM model capable of predicting the next node in a route based on start and destination. Key achievements include:
• Predictions that closely match the true shortest path.
• Ability to adjust to dynamic building routes.
• Integration with the graph to avoid invalid predictions.
• A fallback mechanism ensuring safe navigation even if the model prediction is unclear.

UniRoute uniquely combines classical algorithms with machine learning to improve reliability and adaptability.

8.2.4 High-Quality 3D Visualisation

A major component of UniRoute is its ability to display routes in a **clean 3D model**. Innovations include:
• Separate color coding for each floor.
• Blue highlighted path for Dijkstra's shortest route.
• Zoom-/rotate-/pan-enabled 3D environment.
• Clearly labelled nodes to eliminate confusion.
• Reduced clutter through simplified node grouping and improved layout logic.

This visualization allows users to understand not only the route but also the building's internal structure.

8.2.5 Data Management & Persistence
UniRoute stores:
• The complete graph model
• Dijkstra's precomputed distance matrix
• name2id mappings
• LSTM model weights
• All floor CSV datasets

This ensures the system runs instantly on each launch without rebuilding or retraining, improving efficiency and usability.

8.3 Performance Evaluation
UniRoute was evaluated based on speed, accuracy, efficiency, and user experience.

8.3.1 Processing Time
• Path computation using Dijkstra executes instantly (< 0.1 seconds).
• LSTM prediction also produces results within milliseconds.
• 3D visualization renders smoothly for all floors.
• Model and graph loading completes almost immediately due to efficient caching.

8.3.2 Accuracy
• The shortest route is mathematically guaranteed using Dijkstra's algorithm.

• LSTM predictions achieve high accuracy after training on all path pairs.
• All node connections, including stairs and lifts, are validated to ensure correct navigation.
• Floor transitions are handled cleanly by vertical links.

8.3.3 Reliability

UniRoute demonstrates strong stability and robustness:
• Missing connections are auto-detected.
• Unreachable destinations produce meaningful warnings.
• Incorrect UI selections are safely handled.
• Graph structure remains intact across sessions due to saved models.

8.4 Limitations

While UniRoute performs exceptionally well, certain limitations exist:

1. Accuracy of 3D visualization depends entirely on correctness of x/y/z coordinates in the CSVs.
2. The LSTM model may produce unexpected predictions if building structure changes significantly.
3. JSON and pickle files are not ideal for large-scale campus-wide systems.
4. The system currently supports single-user local execution; a cloud backend would be required for multi-user access.
5. No real-time indoor positioning (IPS) is integrated yet, meaning live movement tracking is not supported.

These limitations create opportunities for future expansion.

8.5 Final Summary

UniRoute successfully delivers:
• A fully automated multi-floor navigation system
• Intelligent AI-assisted route prediction
• Accurate shortest-path computations
• Clean, interactive 3D representation
• A refined and professional GUI
• Complete data persistence and modular architecture

The system advances the traditional navigation experience into a smart, modern, and intuitive solution—making it highly valuable for universities, hospitals, malls, corporate offices, and complex buildings.

Chapter – 9
FUTURE SCOPE

UniRoute is already a powerful indoor navigation system capable of generating intelligent paths using Dijkstra's algorithm, LSTM-based predictive routing, 3D visualization, and multi-floor connectivity. However, several enhancements can elevate the system into a fully featured enterprise-level indoor navigation platform. This chapter outlines the potential future improvements that can significantly advance UniRoute in terms of performance, scalability, accessibility, and intelligence.

9.1 Transition to Database-Driven Architecture

Currently, UniRoute stores data using CSV and JSON files for rooms, coordinates, and graph structures. While efficient for development, a large institution with thousands of rooms and dynamic updates requires a more scalable solution.

Future versions may adopt database systems such as:

- **MySQL**
- **PostgreSQL**
- **MongoDB**
- **Firebase**

A database-driven system will support:

- Efficient handling of **large indoor maps and thousands of nodes**
- **Multi-user editing** of building layouts and room information
- Real-time updates to navigation routes without restarting the system
- **Versioning and audit logs** for building modifications
- Secure cloud-linked data management

Database integration would transform UniRoute into a robust and maintainable enterprise solution.

9.2 AI-Based Smart Path Optimization

The current system uses Dijkstra's shortest-path algorithm and an LSTM-based next-step predictor. In the future, advanced AI techniques can be used to further enhance routing intelligence.

Potential AI integrations include:

- Predicting **fastest routes** based on crowd density
- Reducing congestion during peak hours
- Personalized paths for accessibility needs
- Adapting navigation instructions based on user behavior

- Minimizing staircase usage or prioritizing lifts for certain users

Advanced optimization algorithms may include:

- **Genetic Algorithms**
- **Ant Colony Optimization (ACO)**
- **Reinforcement Learning (RL)**
- **Neural Graph Traversal Models**

These technologies can make UniRoute capable of **learning, adapting, and optimizing** navigation in real time.

9.3 Mobile Application Version

To make indoor navigation accessible anywhere, UniRoute can be extended into a full-featured mobile application for students, staff, and visitors.

Mobile app features may include:

- Turn-by-turn indoor navigation
- Daily class-room navigation for students
- Push notifications for route updates or blockages
- AR-based indoor navigation using the device camera
- Interactive campus maps with zoom, floor switching, and room search
- Offline mode for areas without Wi-Fi

Technologies suitable for mobile app development:

- **Flutter**
- **React Native**
- **Kotlin/Swift**

A mobile version will greatly enhance the practicality and real-time usability of UniRoute.

9.4 Role-Based Authentication System

To improve security and customization, UniRoute can include user roles with different interface access and privileges.

Possible roles:

Admin

- Add/update building maps
- Upload floor CSVs
- Manage rooms and connectors
- Modify graph and coordinates

Faculty/Staff

- View paths to classrooms/labs
- Mark room availability
- Report blockages or restricted areas

Students/Visitors

- Search rooms
- Use route navigation
- View building maps

Role-based dashboards can improve safety, control, and data integrity.

9.5 Cloud Deployment

Hosting UniRoute on cloud platforms will transform it into an institution-wide navigation system accessible from any device.

Possible deployment platforms:

- **AWS**
- **Google Cloud Platform (GCP)**
- **Microsoft Azure**
- **Render**
- **Streamlit Cloud**

Cloud deployment supports:

- 24/7 system accessibility
- Handling high traffic during events
- Centralized management and updates
- Cross-device compatibility

With cloud hosting, UniRoute can scale beyond local usage and serve an entire university campus.

9.6 Integration with Other Campus Systems

UniRoute can evolve into a unified smart-campus platform by integrating with other institutional software, such as:

- **Student Management System (SMS)**
- **Learning Management Systems (LMS)**
- **Classroom Scheduling Software**
- **Attendance Tracking Systems**
- **Emergency Alert Systems**
- **IoT Sensor Networks for crowd density detection**

For example:

- Students can navigate directly to their scheduled class.
- Staff can track room availability in real time.
- IoT sensors can automatically reroute users if a corridor is overcrowded.

Such integration will substantially expand UniRoute's real-world impact.

9.7 Smart Analytics Dashboard

An analytics dashboard can provide deep insights into building usage and navigation patterns.

Possible analytics include:

- Room traffic heatmaps
- Floor-wise usage statistics
- Peak navigation hours
- Staircase vs. lift usage comparison
- Predictive crowd modeling
- Most frequently searched rooms
- Path complexity visualizations

These insights can help the institution:

- Improve building design
- Manage crowd flow
- Optimize classroom allocation
- Enhance safety and accessibility

Analytics transforms UniRoute from a navigation tool into a smart infrastructure management system.

REFRENCES

1. **Streamlit Documentation** – *The fastest way to build data apps.* https://docs.streamlit.io/
2. **Python Official Documentation** – *Python 3.x Language Reference.* https://docs.python.org/3/
3. **Pandas Documentation** – *Data analysis and manipulation tools.* https://pandas.pydata.org/docs/
4. **OpenPyXL Documentation** – *Excel file handling for Python.* https://openpyxl.readthedocs.io/en/stable/
5. **JSON.org** – *JavaScript Object Notation Specification.* https://www.json.org/json-en.html
6. Schaerf, A. (1999). **A Survey of Automated Timetabling.** *Artificial Intelligence Review.*
7. Wren, A. (1996). **Scheduling, Timetabling and Rostering — A Special Relationship?**
8. Kaur, P., & Singh, H. (2020). **An Automated Timetable Generator Using Optimization Algorithms.** *IEEE Conference Proceedings.*
9. Various open-source repositories, articles, and online tutorials used for conceptual understanding and implementation support.