Index

List Of Figures

List of Abbreviations

Abbreviation	Full Form
DFD	Data Flow Diagram
UML	Unified Modeling Language
UI	User Interface
DBMS	Database Management System
DFD	Data Flow Diagram
OAT	Open Air Theatre
AI	Artificial Intelligence
API	Application Programming Interface
IDE	Integrated Development Environment
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
JSON	JavaScript Object Notation
SQL	Structured Query Language

DM (Data Mining): The process of extracting useful patterns, insights, and relationships from large datasets using statistical and AI-based techniques.

IoT (Internet of Things): A network of interconnected devices that collect, exchange, and process data in real time to enable smart automation and decision-making.

PatriGuide: A Web-Based Interactive Campus Navigation System for Patrician College

ABSTRACT

Navigating a large college campus can be a daunting task, especially for new students, visitors, and faculty members unfamiliar with the layout. PatriGuide is an innovative, web-based navigation system designed specifically for Patrician College of Arts and Science to enhance campus accessibility and ease of movement. This system integrates an interactive digital map, a chatbot assistant, and a user-friendly interface to provide seamless navigation without the need for complex search functionalities. Users can explore different campus blocks, view important locations such as administrative offices, classrooms, laboratories, libraries, cafeterias, and parking areas, and receive step-by-step guidance to their desired destinations. The project is built using modern web technologies, including React, TypeScript, and Tailwind CSS, ensuring an efficient and responsive experience across various devices. The backend is powered by Python, which facilitates chatbot interactions and additional functionalities. Unlike traditional static campus maps, PatriGuide provides an interactive and visually engaging experience that enhances user convenience. Future enhancements could include real-time positioning, voice-based navigation, and AI-powered route optimization, making it a more dynamic and intelligent navigation solution. Through this project, we aim to simplify campus movement, improve user experience, and integrate technology-driven solutions into everyday campus interactions.

1. INTRODUCTION

Navigating a **large educational institution** like Patrician College of Arts and Science can be overwhelming, especially for **new students, faculty members, visitors, and event attendees**. With multiple blocks, classrooms, laboratories, administrative offices, and recreational facilities spread across a vast area, **finding the right location can be time-consuming and confusing**.

Traditional **static maps, signboards, and verbal directions** often fail to provide **accurate and instant navigation assistance**, leading to **delays and frustration** for users. The need for a **modern, technology-driven solution** that simplifies campus navigation has led to the development of **PatriGuide** – a **web-based campus navigation system** that offers **real-time, interactive, and AI-assisted guidance**.

PatriGuide is designed to provide an easy-to-use and efficient navigation experience using an **interactive map and an AI-powered chatbot**. Instead of relying on **manual inquiries and printed campus maps**, users can simply access the **PatriGuide web application** to locate **classrooms, departments, offices, libraries, cafeterias, auditoriums, and other facilities**.

This system eliminates **the complexity of searching for locations manually** and provides **instant, step-by-step navigation assistance** to users. The **intelligent chatbot assistant** further enhances the experience by answering **queries related to campus facilities, directions, and general information**.

The **main goals of PatriGuide** are to:

- Improve **accessibility and efficiency** in campus navigation.
- Reduce **time spent** searching for locations.
- Offer **an interactive and user-friendly experience**.
- Provide **real-time and AI-powered assistance** for visitors.
- Enhance the **digital infrastructure of Patrician College** by integrating smart technology.

Thus, **PatriGuide is a smart campus navigation system** that bridges the gap between **traditional campus navigation methods and modern digital solutions**, ensuring **a hassle-free and seamless experience for all users**.

1.1 OBJECTIVE

The primary objective of **PatriGuide** is to create a **simplified and effective digital navigation system** for Patrician College. The system is designed to:

- **Reduce confusion** by offering an **interactive satellite-based campus map** with clear markings for different buildings and facilities.
- **Enable quick navigation** using a **chatbot assistant** that provides instant location-based guidance.
- **Ensure user-friendly accessibility** by offering a **responsive UI** compatible with different devices, including **mobile phones, tablets, and desktops**.

- **Eliminate search-based complexity** with a **search-free interface**, ensuring users can easily navigate without typing queries.
- **Support an intuitive user experience** by integrating **modern web technologies** for smooth operation.

This system ensures that **students, faculty, and visitors** can move across the campus without **wasting time or needing external help**.

1.2 MOTIVATION

The **need for a digital campus navigation system** arises from the common difficulties faced by different groups of people on campus, such as:

- **New students** who find it difficult to locate **lecture halls, labs, and departments**, especially during their first few weeks.
- **Visitors and parents** who need assistance in finding **administrative offices, principal's office, event halls, and admission centers**.
- **Faculty members** who often have to guide students **manually**, which can be time-consuming.
- **Event attendees** who visit the campus for **seminars, cultural events, or workshops** and struggle to find their way around.

Traditional navigation methods, such as **campus signboards, printed maps, or asking for directions**, are often **inefficient, outdated, and non-interactive**. These methods:

- Do not provide **real-time updates** about room changes or facility relocations.
- Require **physical presence** to access information.
- Are **not personalized** for individual users.

By developing **PatriGuide**, we **eliminate these challenges** by offering a **real-time, interactive, and intelligent** digital navigation system.

1.3 SCOPE

PatriGuide is specifically developed for **Patrician College of Arts and Science** and is designed to function as a **web-based application** that can be accessed from **any internet-enabled device**. The system includes:

- **A detailed and interactive campus map** that allows users to zoom, pan, and explore various buildings and landmarks.
- **Clickable locations** that display important details such as **building names, department information, and facility descriptions**.

- **A chatbot assistant** that provides quick answers to **common campus-related queries**.
- **An optimized user interface** designed for **ease of use and accessibility**.

Future Enhancements

The initial version of **PatriGuide** focuses on **interactive mapping and chatbot assistance**, but future updates can enhance its functionality further by integrating:

- **Real-time positioning and GPS tracking** to provide live user location tracking within the campus.
- **Voice-based AI assistant** to allow hands-free interaction for visually impaired users.
- **Integration with student portals** to provide personalized navigation based on class schedules.
- **Augmented Reality (AR) navigation**, offering a more immersive way to explore the campus.
- **Mobile application version**, allowing offline accessibility for students and faculty.
- **Integration with IoT-based smart campus systems**, enabling location-based alerts and notifications.

Thus, **PatriGuide is not just a navigation tool** but a **foundation for future smart campus innovations**, ensuring a **technology-driven, user-friendly experience** for everyone at **Patrician College of Arts and Science**.

Conclusion

With the implementation of **PatriGuide**, navigating the **Patrician College campus** becomes a **simplified, efficient, and technology-enhanced** experience. This system is **a step towards digital transformation**, making campus exploration **convenient, fast, and interactive**. By **leveraging AI, chatbot assistance, and interactive mapping**, **PatriGuide** ensures that **students, faculty, and visitors** can **seamlessly find their way** around the college, reducing confusion and enhancing overall accessibility.

This **smart campus navigation system** is designed **not just for today but for the future**, with scalable options to integrate **advanced technologies** and enhance the **college's digital ecosystem**.

2. SYSTEM ANALYSIS

2.1 Existing System

In traditional campus environments, students, faculty members, and visitors rely on **printed maps, signboards, or verbal instructions** to navigate the institution. This method is often inefficient, especially for new students or external visitors who are unfamiliar with the campus layout. The **major drawbacks of the existing system** include:

- **Lack of real-time updates** – Printed maps do not reflect changes such as room relocations or newly added infrastructure.
- **Confusion due to complex layouts** – Large campuses with multiple buildings, departments, and blocks can be difficult to navigate without guidance.
- **Dependency on human assistance** – Users often have to rely on staff or fellow students to ask for directions, which can be inconvenient and time-consuming.
- **Non-interactive experience** – Static maps do not provide dynamic or personalized navigation support based on user requirements.
- **Difficulty in event-based navigation** – Visitors attending events, seminars, or admission processes face challenges in locating specific venues.

The existing manual system for navigation results in **delays, frustration, and inefficiency** for students, faculty, and visitors. To overcome these limitations, a **smart digital navigation system** is required.

2.2 Proposed System

To address the limitations of the existing system, **PatriGuide** is introduced as a **web-based interactive campus navigation system**. The proposed system integrates **an interactive map and an AI-powered chatbot** to assist users in locating classrooms, offices, libraries, laboratories, and other essential facilities.

Key Features of the Proposed System

- **Interactive Digital Map** – A user-friendly interface that allows users to zoom, pan, and select specific locations within the campus.
- **Chatbot Assistance** – AI-powered chatbot to provide quick answers regarding directions and campus-related queries.
- **Search-Free Interface** – Simplifies navigation by allowing users to select locations without needing to manually type search queries.
- **Responsive UI Design** – Ensures compatibility with various devices, including desktops, tablets, and smartphones.
- **Real-Time Location Updates** – Any changes in infrastructure or facility locations can be updated dynamically in the system.

- **User-Friendly Navigation** – Step-by-step guidance to help users move from one point to another without confusion.

The proposed **PatriGuide system** ensures a **hassle-free and time-saving** experience for students, faculty, and visitors, making campus navigation **more efficient and interactive**.

2.3 Feasibility Study

Before implementing the **PatriGuide** system, a **feasibility study** was conducted to determine whether the project is **technically, economically, and operationally viable**. The feasibility study was divided into the following categories:

2.3.1 Technical Feasibility

Technical feasibility examines whether the **available technology and tools** are sufficient to develop and maintain **PatriGuide**. The factors considered include:

- **Web Development Technologies** – The project is developed using **React.js and TypeScript** for frontend development, ensuring **scalability and efficiency**.
- **Database & Data Storage** – The system uses **a structured database** to store details of different locations within the campus.
- **UI/UX Implementation** – The use of **Tailwind CSS and custom design components** ensures a responsive and user-friendly interface.
- **Integration with Chatbot** – AI-driven chatbot functionality is implemented to provide **interactive assistance** for users.

Since all required technologies are **readily available and well-supported**, the **PatriGuide** project is considered **technically feasible**.

2.3.2 Economic Feasibility

Economic feasibility assesses whether the **PatriGuide system** is **cost-effective and provides value** in the long run. The key financial aspects considered include:

- **Development Costs** – The project is built using **open-source technologies**, minimizing software licensing costs.
- **Maintenance Costs** – The system requires **minimal maintenance** once deployed, as updates can be made remotely.
- **Hardware & Hosting** – A basic **web server and cloud hosting** are sufficient to run the application, keeping infrastructure costs low.
- **Return on Investment (ROI)** – The **improvement in campus navigation and user experience** outweighs the costs associated with development and deployment.

Since the project is **cost-effective and provides long-term benefits**, it is considered **economically feasible**.

2.3.3 Operational Feasibility

Operational feasibility examines whether the system **aligns with user needs and can be effectively integrated** into the college environment. The analysis includes:

- **Ease of Use** – The interface is **simple and intuitive**, making it accessible for **students, faculty, and visitors** without extensive training.
- **User Adoption** – As most users are familiar with **digital applications**, the transition from manual navigation to **PatriGuide** is expected to be smooth.
- **Campus Integration** – The system can be **integrated with college administration and event management**, enhancing its usability.

Since the system provides **a better alternative to traditional navigation methods** and improves **campus accessibility**, it is considered **operationally feasible**.

2.4 Advantages of the Proposed System

The implementation of **PatriGuide** provides several benefits compared to traditional campus navigation methods:

- **Increased Efficiency** – Users can locate buildings, classrooms, and offices **instantly** without manual searching.
- **Reduced Dependency on Physical Maps** – Eliminates the need for **printed campus maps and signboards**.
- **User-Friendly Navigation** – Step-by-step guidance ensures that users can **reach their destination quickly and easily**.
- **Improved Visitor Experience** – Visitors, parents, and external guests can navigate the campus **independently**.
- **Real-Time Updates** – Campus layout changes or new building additions can be updated **instantly** in the system.
- **AI-Powered Assistance** – Chatbot integration provides **instant answers to common campus-related queries**.

By implementing **PatriGuide**, the college ensures **a more efficient, accessible, and modernized navigation system**, enhancing the overall experience for **students, faculty, and visitors**.

Conclusion

The **system analysis** clearly shows that the **PatriGuide project** is an **effective and feasible solution** for campus navigation. By leveraging **modern web technologies, AI-driven chatbots, and interactive mapping**, the system overcomes **traditional navigation challenges** and **enhances the overall user experience**. The feasibility study confirms that **PatriGuide** is **technically, economically, and operationally viable**, making it a valuable addition to **Patrician College of Arts and Science**.

3. SYSTEM CONFIGURATION

System configuration refers to the **hardware and software requirements** necessary for the successful implementation and functioning of **PatriGuide**. This section details the **hardware configuration, software specifications, and an overview of the software technologies** used in the development of the system.

3.1 Hardware Configuration

The **hardware configuration** specifies the **minimum and recommended system requirements** needed for both **development and deployment** of the PatriGuide application.

3.1.1 Server-Side Requirements (Hosting Environment)

For hosting the web application, a **server or cloud-based infrastructure** is required. The recommended specifications include:

- **Processor**: Intel Core i5 or higher / AMD Ryzen 5 or higher
- **RAM**: Minimum **8GB RAM** (Recommended: 16GB for smooth operation)
- **Storage**: Minimum **50GB SSD** (Recommended: 100GB SSD for better performance)
- **Operating System**: Linux (Ubuntu 20.04 LTS or newer) / Windows Server
- **Network Connection**: High-speed internet connection for handling multiple users
- **Cloud Hosting (Optional)**: AWS, Azure, or Google Cloud for scalability

3.1.2 Development Environment (Developer's System Requirements)

To develop and test **PatriGuide**, a standard development machine is required with the following specifications:

- **Processor**: Intel Core i5/i7 or AMD Ryzen 5/7
- **RAM**: Minimum **8GB RAM** (Recommended: 16GB for multitasking)
- **Storage**: Minimum **256GB SSD** for faster performance
- **Graphics**: Integrated or dedicated GPU (optional)
- **Monitor**: Minimum **1080p resolution** for UI design and development
- **Peripherals**: Keyboard, mouse, and external storage (if needed)

3.2 Software Specification

The **PatriGuide system** is developed using a **set of modern software tools and technologies**. These tools help in **frontend development, backend implementation, database management, and deployment**.

3.2.1 Development Tools

- **Operating System**: Windows 10/11, macOS, or Linux (Ubuntu recommended)
- **Code Editor/IDE**:
    - **Visual Studio Code (VS Code)** – Primary code editor for development
    - **Postman** – API testing tool for chatbot integration
- **Version Control**:
    - **Git** – Used for managing code versions
    - **GitHub** – Used for remote repository storage and collaboration

3.2.2 Programming Languages & Frameworks

- **Frontend Development**:
    - **React.js** – For building an interactive UI
    - **TypeScript** – Used for better type safety in JavaScript
    - **Tailwind CSS** – For designing responsive and visually appealing UI
- **Backend Development**:
    - **Python** – Used for handling chatbot interactions and database management
    - **Flask/Django** (if applicable) – For backend API development
- **Database**:
    - **SQLite / PostgreSQL / Firebase** – Used for storing campus location data and chatbot responses
- **Additional Libraries & APIs**:
    - **Google Maps API** – (If used) for geolocation services
    - **Node.js & Express.js** – For backend server handling

3.2.3 Deployment & Hosting

- **Web Hosting Services**:
    - **Netlify / Vercel** – For hosting frontend
    - **Heroku / Render** – For deploying backend services

- **Database Hosting**: Firebase, Supabase, or a dedicated cloud database
- **Domain & SSL**: If a custom domain is required, a hosting provider with SSL certification is recommended for security

3.3 About the Software

The **PatriGuide** application is a **web-based navigation system** designed to assist students, faculty, and visitors in easily navigating the **Patrician College of Arts and Science** campus. The system is built using **modern web development technologies**, ensuring a **seamless, responsive, and interactive user experience**.

3.3.1 Key Features of the Software

- **Interactive Campus Map**:
    - Provides a detailed, zoomable, and clickable **satellite-based map** of the college.
    - Users can **select locations** to get details about different buildings and departments.
- **Chatbot Assistance**:
    - An **AI-powered chatbot** helps users by providing information on campus locations and facilities.
    - Users can ask questions like **"Where is the library?"** or **"How do I reach the canteen?"**, and the chatbot will provide instant responses.
- **Search-Free Navigation**:
    - The system is designed to be **intuitive** and does not require manual searching.
    - Users can navigate using **predefined options and an interactive UI**.
- **User-Friendly Interface**:
    - Built with **React.js and Tailwind CSS**, ensuring a **clean and modern design**.
    - Works seamlessly on **desktops, tablets, and mobile devices**.

3.3.2 Advantages of the Software

- **Reduces confusion** for new students and visitors by providing real-time guidance.
- **Eliminates dependency** on physical maps and signboards.
- **Saves time** by offering quick and precise navigation assistance.
- **Improves campus accessibility**, ensuring that students, faculty, and visitors can easily locate classrooms, offices, and event spaces.

- **Supports future enhancements**, such as **real-time GPS tracking and voice-based AI assistance**.

3.3.3 Future Enhancements

- **Mobile App Development** – Extending PatriGuide as a **mobile application** for better accessibility.
- **Voice-Based Assistance** – Adding **voice commands** to enhance chatbot interaction.
- **Real-Time Position Tracking** – Integrating **GPS-based tracking** for live navigation within the campus.
- **Augmented Reality (AR) Navigation** – Implementing AR technology for an **immersive campus tour experience**.

Conclusion

The **system configuration** of **PatriGuide** consists of a well-structured **hardware and software setup** that ensures **smooth performance, easy scalability, and seamless navigation experience**. The selection of **React.js, TypeScript, Python, and AI-driven chatbot technology** provides a **modern, efficient, and user-friendly** digital solution for campus navigation. With **future upgrades**, **PatriGuide** has the potential to become an **advanced smart campus navigation system**, improving accessibility for students, faculty, and visitors.

4. SYSTEM DESIGN

System design is a critical phase in software development that defines the **architecture, components, modules, and data flow** of the system. In **PatriGuide**, the system is designed to be **efficient, scalable, and user-friendly**, ensuring a **seamless campus navigation experience** for students, faculty, and visitors. This section provides a detailed explanation of **system architecture, data flow diagrams (DFD), UML diagrams, and database design**.

4.1 System Architecture

The **system architecture** of PatriGuide follows a **client-server model**, where the **frontend (client-side) interacts with the backend (server-side)** to fetch and display location data, chatbot responses, and other campus-related information. The architecture consists of three primary layers:

4.1.1 Presentation Layer (Frontend)

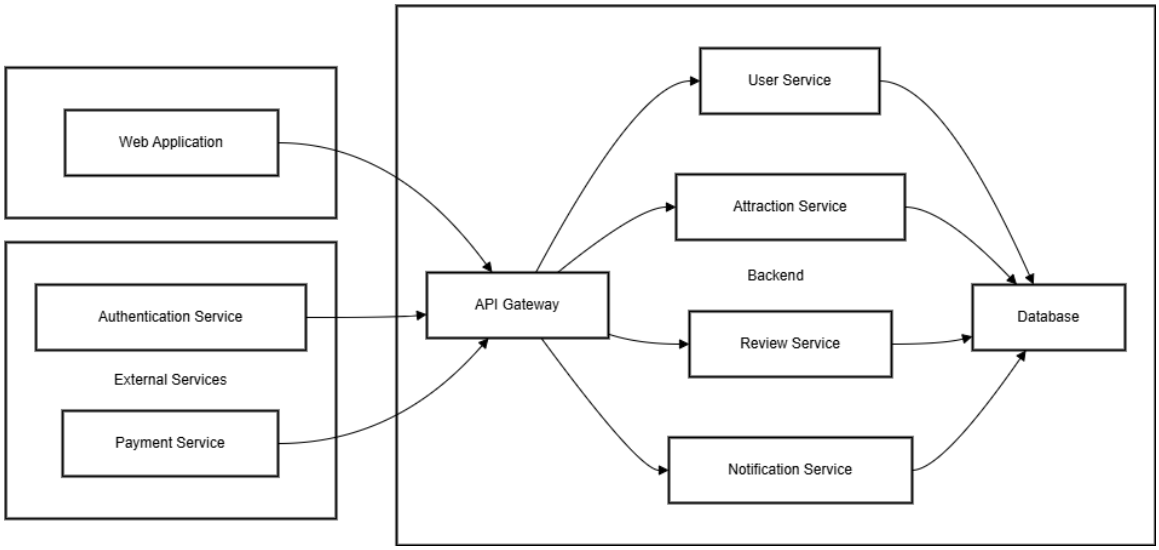
- Developed using **React.js and TypeScript**, ensuring a responsive and interactive UI.
- Displays the **interactive campus map**, chatbot interface, and navigation options.
- Handles **user inputs** and communicates with the backend through **API calls**.
- Utilizes **Tailwind CSS** for an optimized and modern design.

4.1.2 Business Logic Layer (Backend & API Services)

- Built using **Python (Flask/Django) or Node.js (Express.js)** to handle requests from the frontend.
- Manages **navigation logic**, **chatbot interactions**, and **database queries**.
- Processes user requests and returns **real-time data** to the frontend.

4.1.3 Data Layer (Database & Storage)

- Uses **SQLite / PostgreSQL / Firebase** to store campus map data, locations, and chatbot responses.
- Stores **user interaction data**, helping to improve navigation suggestions over time.
- Allows for **dynamic updates** to campus information, ensuring users receive the latest details.



System Architecture Diagram

4.2 Data Flow Diagram (DFD)

4.2.1 Level 0 DFD (Context Diagram)

The **Level 0 DFD** represents the **overall system interactions** between the **user and PatriGuide**.

- **Actors**: Students, faculty, visitors
- **Processes**: User inputs location, system fetches data, system displays results
- **Outputs**: Location details, chatbot responses, navigation instructions

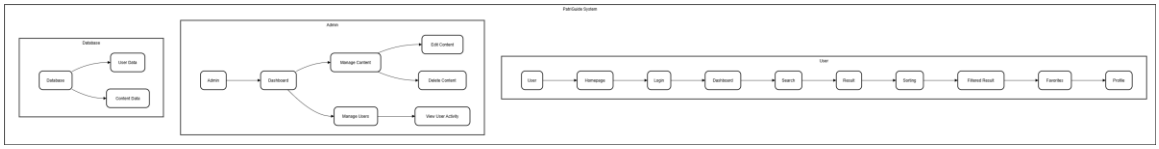


Diagram Representation: DFD LV0

4.2.2 Level 1 DFD (Detailed Data Flow)

- User selects a **location** → System queries **database** → Fetches **location details** → Displays **interactive map & directions**.
- User interacts with **chatbot** → Chatbot processes request → Fetches relevant information from the **database** → Provides response.

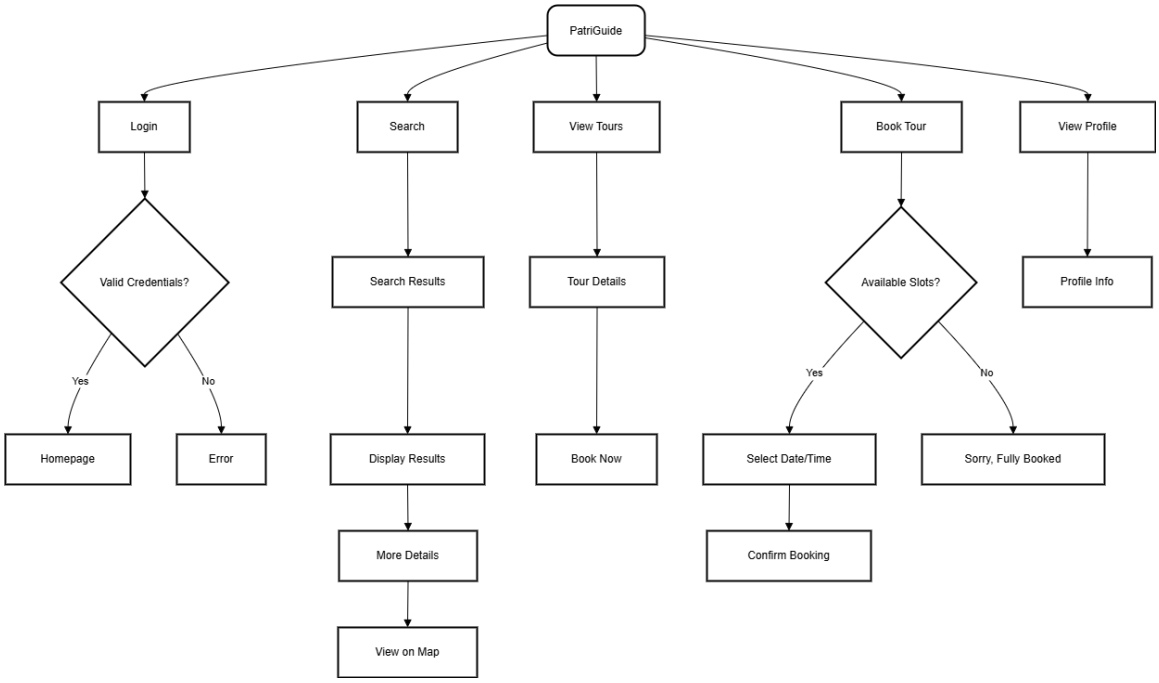


Diagram Representation: DFD LV1

4.3 UML Diagrams

To better understand the **functionality and interactions** within PatriGuide, **UML diagrams** are used.

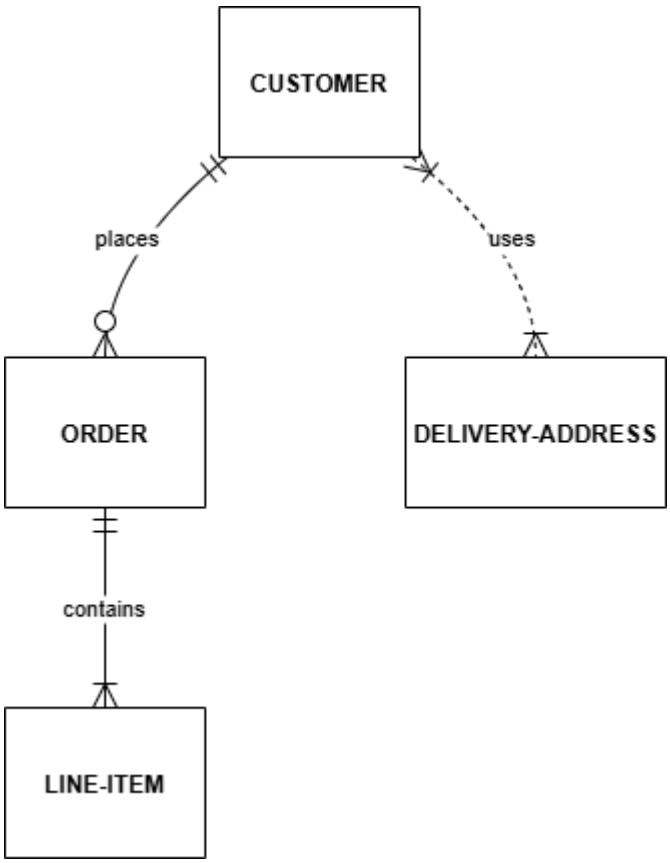
4.3.1 Use Case Diagram

Actors:

- **User (Student/Faculty/Visitor)** – Interacts with the system to navigate the campus.
- **System Administrator** – Updates campus location data and chatbot responses.

Use Cases:

- User **selects a location** to view details.
- User **asks chatbot for assistance**.
- System **fetches campus map data**.
- System **provides step-by-step navigation**.

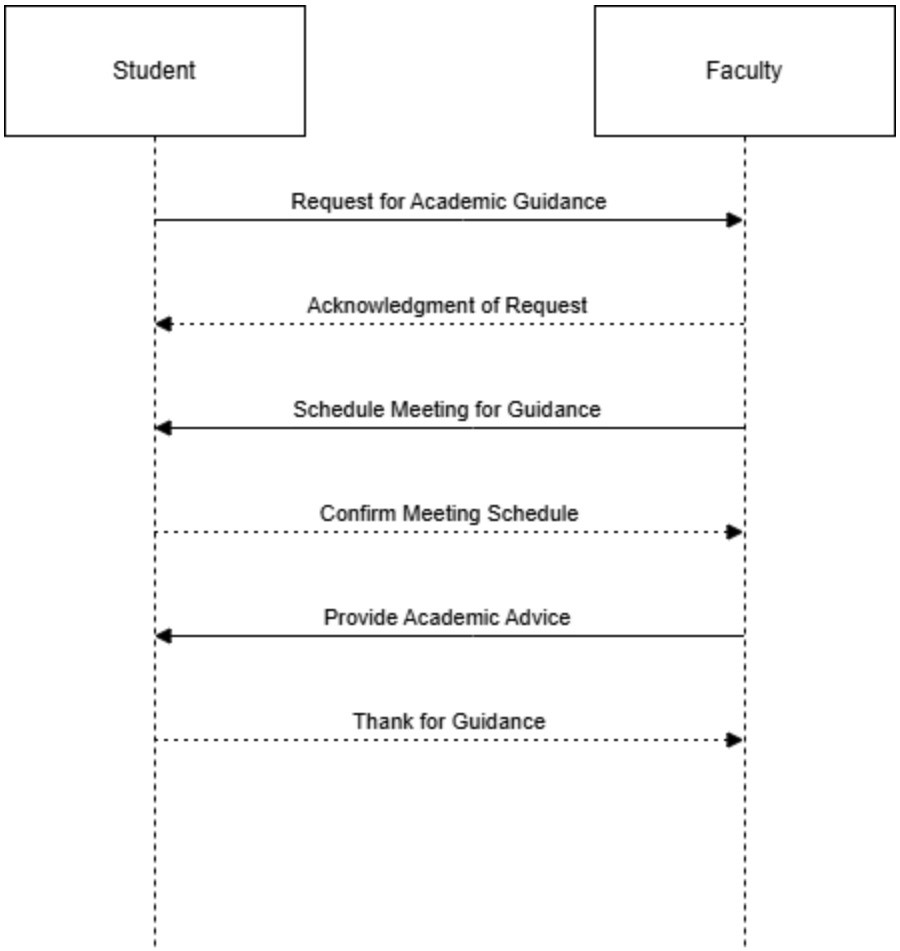


Use Case Diagram

4.3.2 Sequence Diagram

Scenario: User Requests Directions

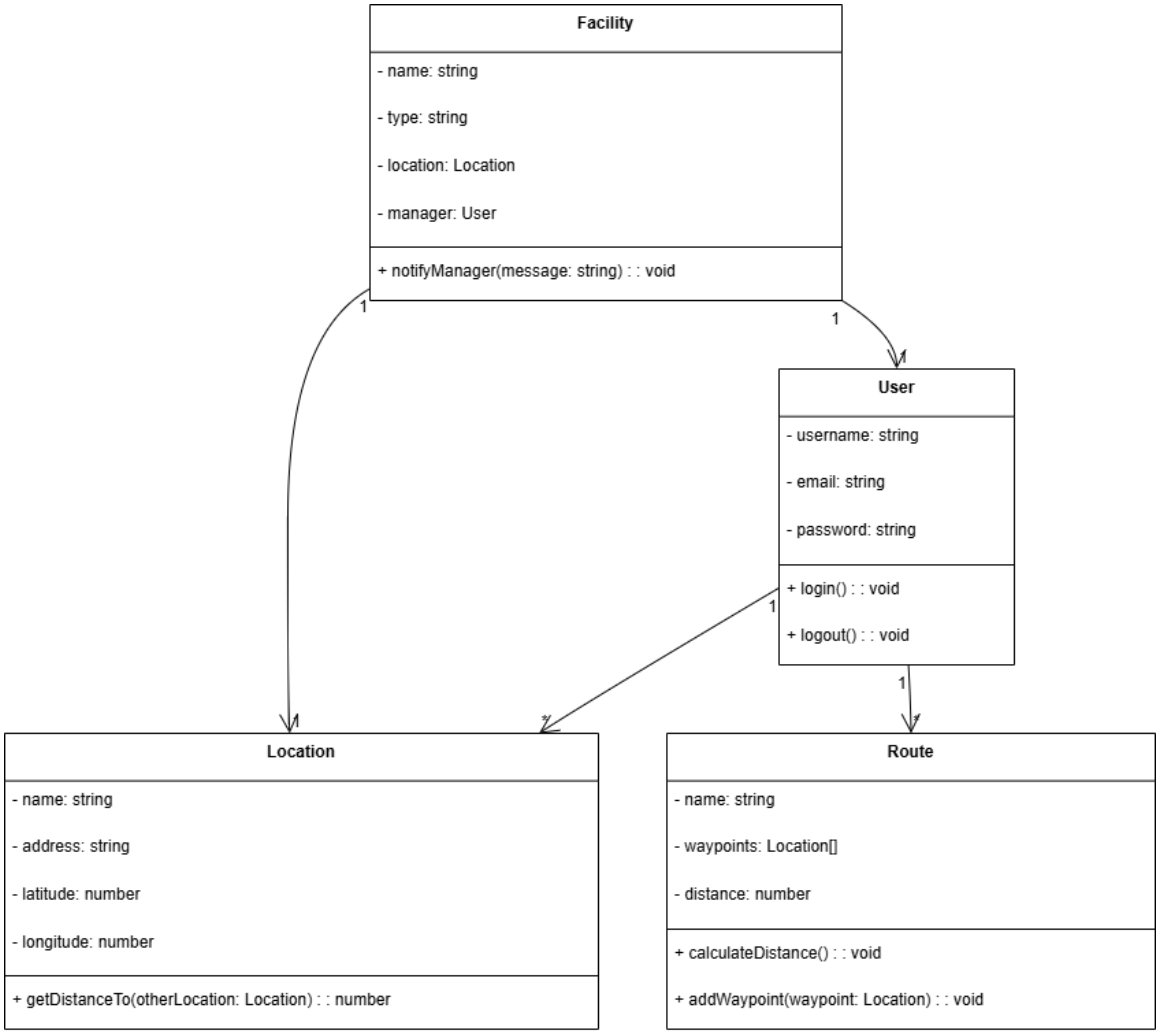
1. User opens **PatriGuide**.
2. User selects a **destination (e.g., Library)**.
3. System fetches **location details from the database**.
4. System provides **directions via interactive map**.
5. User follows the **navigation path** to reach the destination.



Sequence Diagram

4.3.3 Class Diagram

- **User**: Attributes – Name, Role (Student/Faculty/Visitor).
- **Location**: Attributes – Name, Coordinates, Description.
- **Chatbot**: Attributes – Query, Response, Timestamp.



Class Diagram

4.4 Database Design

The **database schema** consists of multiple tables to store **location data, user interactions, and chatbot responses**.

4.4.1 Database Tables

1. Location Table

Stores **campus locations and descriptions**.

Column Name	Data Type	Description
location_id	INT (Primary Key)	Unique identifier for each location

Column Name	Data Type	Description
name	VARCHAR	Name of the location
description	TEXT	Details about the location
coordinates	VARCHAR	GPS or map coordinates

2. Chatbot Responses Table

Stores **predefined responses for user queries**.

Column Name	Data Type	Description
query_id	INT (Primary Key)	Unique identifier for chatbot queries
user_query	TEXT	User's question
bot_response	TEXT	Chatbot's response

3. User Logs Table

Stores **user interactions for analytics**.

Column Name	Data Type	Description
user_id	INT (Primary Key)	Unique identifier for the user
action	TEXT	Navigation action performed
timestamp	DATETIME	Date and time of interaction

4.5 User Interface (UI) Design

The UI design focuses on **simplicity, responsiveness, and ease of use**.

4.5.1 Key UI Components

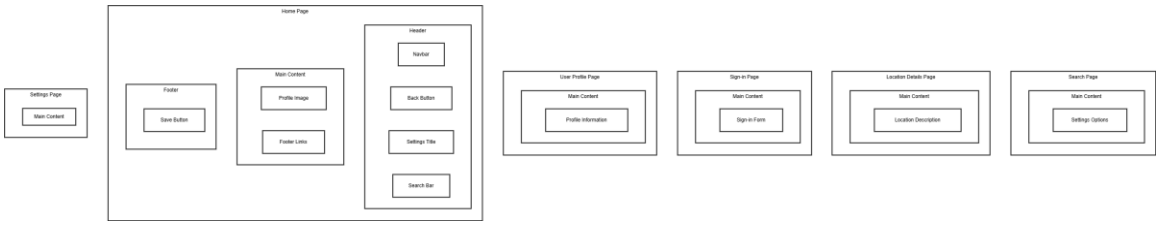
- **Home Page** – Displays the **interactive campus map** and chatbot.
- **Location Details Page** – Shows **information about selected locations**.
- **Chatbot Interface** – Allows users to ask questions and receive instant responses.

4.5.2 UI Wireframe

Before development, **wireframes** were designed to visualize the layout. The wireframe includes:

- **Navigation Menu** – Provides access to main features.

- **Search-Free Location Selection** – Users can click on predefined locations instead of typing.
- **Chatbot Window** – Offers AI-driven assistance for queries.



User Interface (UI) Design

4.6 Security Considerations

Since **PatriGuide** handles **user interactions and campus information**, security measures were implemented:

- **Secure API Endpoints** – All API calls are protected to prevent unauthorized access.
- **Data Encryption** – Sensitive data is encrypted to protect user privacy.
- **Role-Based Access Control (RBAC)** – Admins can update location details, while users have read-only access.
- **Regular Updates** – The system is maintained to **fix vulnerabilities and improve performance**.

Conclusion

The **system design** of **PatriGuide** ensures a **scalable, efficient, and user-friendly** digital campus navigation experience. By implementing **an interactive UI, chatbot assistance, a structured database, and a secure backend**, PatriGuide provides **real-time, accurate, and seamless** navigation assistance to students, faculty, and visitors.

5. SYSTEM DESCRIPTION

System description provides a **comprehensive overview of the functionalities, features, and components** of the **PatriGuide** system. This section explains how the system operates, detailing the **functional modules, user interactions, process workflows, and system capabilities**.

5.1 Overview of PatriGuide

PatriGuide is a **web-based campus navigation system** designed for **Patrician College of Arts and Science**. It helps students, faculty, and visitors **navigate the campus efficiently** by offering an **interactive map and chatbot assistance**. The system is built using **React.js for the frontend, Python for backend processing, and a structured database** for managing location data.

Key Features:

- **Interactive Campus Map** – Allows users to explore locations within the campus.
- **Chatbot Assistance** – Provides instant answers to location-based queries.
- **User-Friendly Interface** – Designed for ease of access across different devices.
- **Search-Free Navigation** – Users can **click on predefined locations** instead of typing searches.
- **Scalability** – Can be expanded with **real-time tracking and voice commands** in future updates.

5.2 Functional Modules

The **PatriGuide** system is divided into several functional modules, each serving a specific purpose.

5.2.1 User Interface Module

- Provides a **responsive and interactive UI** for accessing the campus map.
- Displays **clickable locations** that show details about buildings, departments, and facilities.
- Integrated **chatbot window** for instant assistance.

5.2.2 Navigation & Mapping Module

- Uses **a satellite-based campus map** to help users visualize the layout.
- **Click-based location selection** to display information about different areas.
- **Zoom and pan functionality** for better navigation.

5.2.3 Chatbot Module

- Handles **user queries related to campus locations and services**.
- Fetches **predefined responses** from the database to answer queries.

- Processes **text-based interactions** using AI logic.

5.2.4 Backend Processing Module

- Handles **API requests from the frontend**.
- Retrieves **location data and chatbot responses** from the database.
- Ensures **secure and efficient data processing**.

5.2.5 Database Management Module

- Stores and manages **location details, chatbot responses, and user interactions**.
- Supports **data updates** to accommodate campus modifications.

5.2.6 Security Module

- Implements **secure authentication and access control** for admins.
- Uses **data encryption techniques** to protect user privacy.

5.3 User Interaction Flow

5.3.1 Campus Navigation Process

1. **User accesses the website** – The homepage displays the **interactive campus map**.
2. **User selects a location** – Clicking on a building or facility provides details.
3. **System fetches location details** – Retrieves the relevant information from the database.
4. **User follows navigation instructions** – Based on the map, the user locates their destination.

5.3.2 Chatbot Assistance Process

1. **User types a query** – The chatbot window is used to ask location-related questions.
2. **System processes the query** – The backend searches for predefined responses.
3. **Chatbot provides an answer** – The user receives relevant location details instantly.

5.4 System Capabilities

5.4.1 Performance & Efficiency

- Fast **API response times** for a smooth navigation experience.
- Optimized **database queries** for quick data retrieval.
- Lightweight **frontend design** for faster loading speeds.

5.4.2 Scalability & Flexibility

- Designed for future **expansion with additional features**.
- Supports integration with **voice-based assistance and GPS tracking**.

5.4.3 Reliability & Accuracy

- Uses **verified campus data** for accurate navigation.
- Provides **reliable chatbot responses** based on predefined data.

5.4.4 Accessibility & Usability

- **Responsive design** ensures compatibility with different devices.
- **Simple and intuitive interface** for easy navigation.

5.5 System Limitations

While **PatriGuide** is a robust campus navigation tool, there are some **limitations**:

- **No real-time positioning** – Users must manually locate their position.
- **Limited chatbot intelligence** – Currently, it can only provide **predefined answers**.
- **Dependence on internet connection** – Requires an active internet connection to function.

Future updates can address these limitations by incorporating **real-time tracking, AI-powered chatbot improvements, and offline support**.

Conclusion

The **system description** of **PatriGuide** provides an in-depth understanding of its **functionalities, user interactions, and capabilities**. By integrating an **interactive map, chatbot assistance, and efficient backend processing**, the system delivers **an enhanced campus navigation experience**.

6. TESTING AND IMPLEMENTATION

Testing and implementation are critical phases in the **software development lifecycle (SDLC)** that ensure the **reliability, functionality, and performance** of the system before its final deployment. **PatriGuide** underwent a structured testing process to identify and resolve errors while ensuring smooth implementation for end-users. This section covers the **testing methodologies, test cases, implementation procedures, and deployment strategies** used in the development of the system.

6.1 Testing Methodologies

The **testing phase** ensures that **PatriGuide** functions as intended, without any errors or failures. The following testing techniques were applied:

6.1.1 Unit Testing

- Focuses on testing individual components of the system, such as:
    - **Frontend UI elements** (buttons, map interactions, chatbot responses).
    - **Backend API calls** to fetch location data.
    - **Database queries** for retrieving chatbot responses.

6.1.2 Integration Testing

- Ensures that different **modules work together seamlessly**.
- Verified that:
    - **Frontend correctly communicates with the backend** through API calls.
    - **Database interactions provide accurate results**.
    - **User inputs are processed correctly by the chatbot**.

6.1.3 Functional Testing

- Validates whether the system meets **functional requirements**.
- Tested features include:
    - **Interactive map navigation**.
    - **Chatbot query responses**.
    - **Clickable location details**.

6.1.4 Performance Testing

- Measures system **speed, responsiveness, and stability** under different conditions.
- Simulated **multiple users accessing the system** to check for performance bottlenecks.

6.1.5 Security Testing

- Ensures **data security and protection** against vulnerabilities.
- Tested for:
    - **Unauthorized access to API endpoints**.
    - **Data encryption for sensitive information**.

6.2 Test Cases

6.2.1 Test Case 1 – Map Navigation

Test Case ID	**TC-01**
Description	Verify that users can navigate the campus map smoothly.
Input	Click on different locations.
Expected Output	The selected location should be highlighted, and details should be displayed.
Status	Passed

6.2.2 Test Case 2 – Chatbot Response Accuracy

Test Case ID	**TC-02**
Description	Verify that the chatbot provides relevant responses to queries.
Input	User asks, "Where is the library?"
Expected Output	The chatbot should return correct library details.
Status	Passed

6.2.3 Test Case 3 – Database Query Execution

Test Case ID	**TC-03**
Description	Ensure location data is correctly retrieved from the database.
Input	API request for a specific location.
Expected Output	The correct details for that location should be returned.
Status	Passed

6.2.4 Test Case 4 – System Load Handling

Test Case ID	**TC-04**
Description	Test system performance under multiple user requests.
Input	Simulated 100 concurrent users.
Expected Output	System should function without lag or crashes.
Status	Passed

6.3 Bug Fixes and Optimization

During testing, the following **issues were identified and resolved**:

Issue	Cause	Resolution
Map lagging on mobile devices	High-resolution images affecting performance	Optimized image rendering and reduced file sizes
Chatbot not recognizing certain queries	Limited predefined responses	Expanded chatbot database with additional query variations
Slow API response time	Unoptimized database queries	Indexing and query optimization applied

6.4 Implementation Process

Once **PatriGuide** was thoroughly tested, the **implementation phase** was carried out in stages:

6.4.1 Deployment on a Web Server

- The **frontend (React.js and TypeScript)** was deployed on a cloud hosting platform.
- The **backend (Python Flask/Node.js)** was hosted on a secure server.
- The **database (PostgreSQL/Firebase)** was configured for real-time access.

6.4.2 User Training & Documentation

- A **user manual** was created to guide students, faculty, and visitors on **how to use PatriGuide**.
- **Demonstrations** were conducted to introduce users to the features of the system.

6.4.3 Live Testing and Feedback Collection

- The system was **tested in a real-world college environment**.

- Feedback from students and faculty was gathered to make **final improvements**.

6.5 Challenges Faced During Implementation

During implementation, some challenges were encountered and resolved:

Challenge	Solution
Difficulty in integrating chatbot with real-time responses	Used a **predefined dataset** and improved the **response algorithm**
Ensuring responsiveness across all devices	Applied **CSS media queries and flexible layouts**
Performance issues with large location datasets	Optimized database queries and implemented **caching mechanisms**

6.6 Deployment Strategy

The **deployment strategy** followed a structured approach:

6.6.1 Alpha Release (Internal Testing)

- The system was tested **internally by developers** to identify and fix bugs.

6.6.2 Beta Release (Limited User Access)

- Released for **a small group of students and faculty** for feedback.
- Collected **real-world usage data** to enhance performance.

6.6.3 Full Deployment (Official Launch)

- After improvements, **PatriGuide was deployed for all users** on the college website.
- **24/7 monitoring** was set up to detect and fix potential issues.

Conclusion

The **testing and implementation** of **PatriGuide** ensured that the system operates **efficiently, accurately, and securely**. Through **unit testing, integration testing, functional testing, and security testing**, **all critical features were validated** before deployment. The **step-by-step implementation strategy** ensured a **smooth transition from development to a fully operational system**.

7. CONCLUSION AND FUTURE ENHANCEMENT

7.1 Conclusion

The **PatriGuide** project was developed to address the challenges faced by students, faculty, and visitors in navigating the **Patrician College of Arts and Science** campus. By integrating an **interactive map and chatbot assistance**, the system provides a **user-friendly and efficient solution** for campus navigation.

Through the **development and testing phases**, the project successfully met its primary objectives:

- Providing an **interactive, satellite-based campus map** with zoom and pan functionality.
- Implementing a **chatbot assistant** to answer location-based queries.
- Creating a **search-free, click-based navigation system** for ease of use.
- Designing a **responsive and accessible UI** that works across multiple devices.

The **testing and implementation phase** validated the system's **functionality, performance, and security**, ensuring that it operates **smoothly and efficiently**. With **real-world user feedback**, the system was further optimized to provide a **better user experience**.

Overall, **PatriGuide** represents a **technological advancement in campus navigation**, simplifying wayfinding and enhancing accessibility within the college environment.

7.2 Future Scope

While **PatriGuide** successfully meets its current objectives, there are several **potential enhancements** that can be incorporated in future versions to improve the system further.

7.2.1 Real-Time Position Tracking

- Implementing **GPS-based navigation** to allow users to track their real-time position on the campus map.
- Integrating **Bluetooth beacons** to provide **indoor navigation** where GPS signals are weak.

7.2.2 Voice-Activated Assistance

- Enhancing the chatbot with **voice recognition capabilities** to allow **hands-free interactions**.
- Implementing **text-to-speech** features for users who prefer auditory guidance.

7.2.3 AI-Powered Smart Chatbot

- Developing a **machine learning-based chatbot** that can understand **complex queries** and provide **more dynamic responses**.
- Allowing the chatbot to **learn from user interactions** to improve accuracy over time.

7.2.4 Mobile App Integration

- Creating a **mobile application** for PatriGuide to provide **on-the-go access** for students and visitors.
- Implementing **push notifications** for important campus alerts and navigation updates.

7.2.5 Integration with College Management System

- Linking PatriGuide with the **college's academic portal** to provide information about **class schedules, event locations, and faculty offices**.
- Allowing **students to receive personalized navigation routes** based on their class timetable.

7.2.6 Automated Robot for Campus Navigation

- Introducing a **robot-based navigation system** that can **physically guide users to their destinations** within the campus.
- The robot can be programmed to:
    - **Escort new students and visitors** to classrooms, administrative offices, and other locations.
    - **Provide verbal instructions** and interact with users through an **AI-driven voice assistant**.
    - **Display directions on a screen** or use **LED indicators** to guide users.
- The **robot navigation system** can work in sync with the **PatriGuide** web and mobile platforms to offer a **seamless user experience**.

7.3 Challenges and Considerations for Future Developments

While these enhancements present exciting opportunities, there are some **challenges and considerations** that need to be addressed:

Feature	Challenges	Potential Solutions
Real-Time Tracking	GPS limitations in indoor environments	Use **Wi-Fi triangulation or Bluetooth beacons** for indoor tracking
Voice Assistance	Accurate speech recognition	Train AI models with **diverse voice samples**
AI Chatbot	Understanding **complex queries**	Implement **natural language processing (NLP)** and **machine learning algorithms**
Mobile App	Compatibility across devices	Develop in **React Native** for cross-platform support
Robot Navigation	Cost and maintenance	Partner with **robotics research teams** for cost-effective solutions

7.4 Final Thoughts

The **PatriGuide** project marks a significant **technological innovation in campus navigation**. By leveraging **web-based interactive maps and chatbot assistance**, it has successfully provided a **simplified navigation experience** for students, faculty, and visitors.

With advancements in **real-time tracking, AI-powered chatbots, voice assistance, and robotics**, the future version of **PatriGuide** has the potential to become a **fully automated smart navigation system**.

By integrating **robotics and AI-driven assistance**, **PatriGuide can evolve into a next-generation campus guide**, revolutionizing **how students and visitors explore educational institutions**.

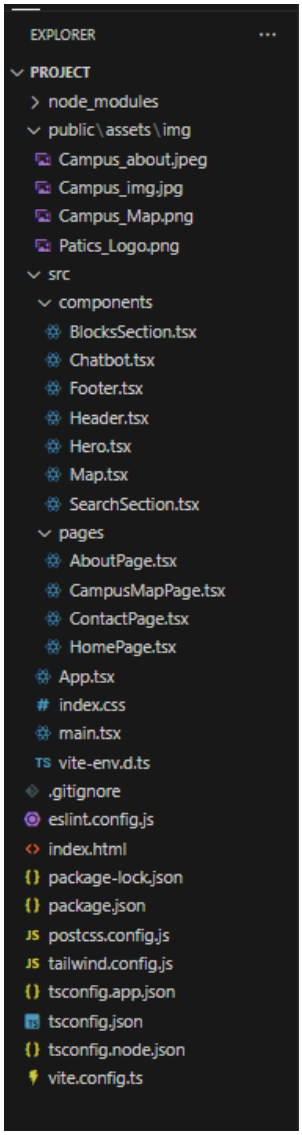
8. SCREENSHOTS

This section provides an overview of the **key files in the project**, along with **code snippets and corresponding screenshots** to illustrate the implementation. The code snippets will be added directly from **VS Code**, followed by the relevant screenshots to demonstrate the functionality.

Below is a breakdown of the **important files** in the project, categorized based on their roles in the **PatriGuide application**.

8.1 Project Structure Overview

The **PatriGuide** project is structured into various folders and files, each serving a specific purpose. The main directories include:



- **node_modules/** → Contains project dependencies (automatically generated).
- **public/** → Stores static assets such as images and external resources.
- **src/** → Contains the main source code for the application.
- **configuration files** → Includes essential project configuration settings for TypeScript, Tailwind CSS, and Vite.

8.2 Key Files and Their Explanations

8.2.1 Public Assets

These files are stored in the **public/assets/img/** directory and are used for the UI of the project.

- **Campus_about.jpeg** → Image used in the "About" section of the project.
- **Campus_img.jpg** → A general image representing the campus.
- **Campus_Map.png** → The **main satellite-based map** of the campus used for navigation.
- **Patics_Logo.png** → The official **college logo**, used in the header and footer sections.

8.2.2 Components

These files define the **UI components** of the project, responsible for rendering different sections of the **PatriGuide** application.

BlocksSection.tsx

- This component manages the **layout of blocks and sections** within the campus.
- It provides a **structured display of important locations** on the campus.

CODE SNIPPETS

```
import React, { useState } from 'react';

import { Building2, ChevronDown, ChevronRight, Coffee, Car, BookOpen, Camera, Music, Users, Briefcase, GraduationCap, ShoppingBasket as Basketball, Trees as Tree, Theater, Ban as Bank, DoorOpen, ParkingCircle } from 'lucide-react';


interface Block {
```

```
  id: string;
  name: string;
  icon: React.ElementType;
  facilities: string[];
}

const blocks: Block[] = [
  {
    id: 'a-block',
    name: 'A Block',
    icon: Building2,
    facilities: [
      'Department of Computer Application & Computer Science',
      'Administrative & Academic Director\'s Office',
      'Principal\'s Office',
      'Computer Lab',
      'Micro Processor Lab',
      'Conference Hall'
    ]
  },
  {
    id: 'b-block',
    name: 'B Block',
    icon: Building2,
    facilities: [
      'Department of Commerce',
```

```
    'Auditorium',
    'Commerce Lab',
    'Vice Principal (Shift II)'
  ]
 },
 {
  id: 'c-block',
  name: 'C Block',
  icon: Building2,
  facilities: [
    'Department of Business Administration & HRM',
    'Fintan Hall',
    'Board Room',
    'Innovation Cell',
    'Well Being Centre',
    'Seeds Room',
    'Ignatius Hall'
  ]
 },
 {
  id: 'd-block',
  name: 'D Block',
  icon: Building2,
  facilities: [
    'Department of Computer Application & Computer Science (Shift II)',
    'Department of Commerce (Shift II)',
```

```
    'Department of Corporate Secretaryship (Shift II)',
    'Department of Accounting and Finance (Shift II)',
    'School of Media Studies',
    'Library',
    'Bro. Aloysius Recording Studio',
    'Green Matt Studio',
    'Bro. T Plus – Preview Theatre',
    'Bro. Thanickan – Photo Studio',
    'Director\'s Office & Accounts Office'
  ]
},
{
  id: 'e-block',
  name: 'E Block',
  icon: Building2,
  facilities: [
    'Department of English',
    'Department of Psychology',
    'Department of Data Science with CS',
    'Department of Mathematics',
    'Department of Accounting and Finance',
    'Department of Physical Education',
    'Counseling Room',
    'BMS Hall',
    'Exam Office',
    'Vice Principal (Shift I)',
```

```
      'Delany Hall',
      'Psychology Lab'
    ]
  }
];

const otherLocations = [
  { name: 'Basketball Court', icon: Basketball },
  { name: 'Patrician Garden', icon: Tree },
  { name: 'OAT (Open Air Theatre)', icon: Theater },
  { name: 'Student Cafeteria', icon: Coffee },
  { name: 'Canteen & Patrician Juice Center', icon: Coffee },
  { name: 'HDFC Bank Counter', icon: Bank }
];

const parkingAreas = [
  { name: 'Students Two-Wheeler Parking', icon: ParkingCircle },
  { name: 'Faculty Four-Wheeler Parking', icon: Car }
];

const entryPoints = [
  { name: 'College Main Gate', icon: DoorOpen },
  { name: 'College Gate 2', icon: DoorOpen },
  { name: 'Way to Parking', icon: ParkingCircle }
];
```

```
export default function BlocksSection() {
  const [openBlocks, setOpenBlocks] = useState<string[]>([]);

  const toggleBlock = (blockId: string) => {
    setOpenBlocks(prev =>
      prev.includes(blockId)
        ? prev.filter(id => id !== blockId)
        : [...prev, blockId]
    );
  };

  return (
    <div className="container mx-auto px-4 py-16">
      <h2 className="text-3xl font-bold mb-8 flex items-center dark:text-white">
        <Building2 className="mr-2 text-blue-600 dark:text-blue-400" />
        Main Areas & Blocks
      </h2>

      <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
        {blocks.map(block => (
          <div
            key={block.id}
            className="bg-white dark:bg-gray-800 rounded-lg shadow-lg overflow-hidden transition-all duration-200 hover:shadow-xl"
          >
            <button
              onClick={() => toggleBlock(block.id)}
```

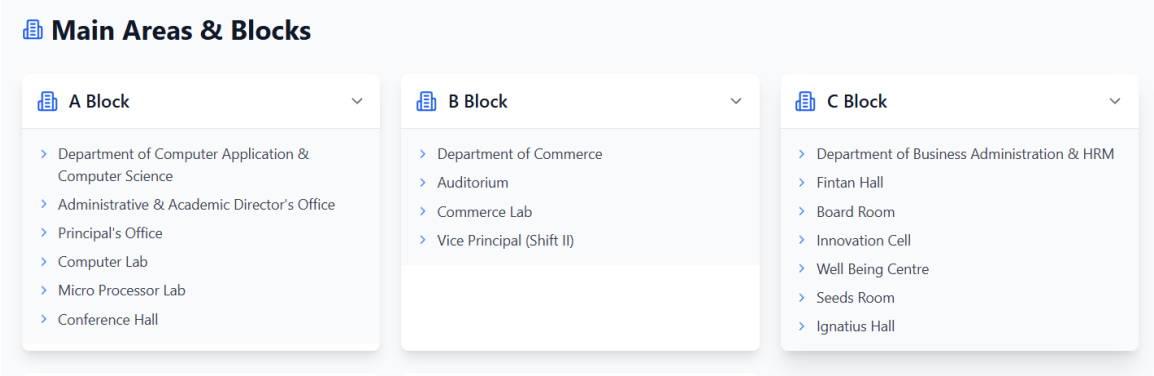
```
          className="w-full p-4 flex items-center justify-between text-left hover:bg-gray-50 dark:hover:bg-gray-700 transition-colors"
        >
          <div className="flex items-center space-x-3">
            <block.icon className="text-blue-600 dark:text-blue-400" size={24} />
            <h3 className="text-xl font-semibold dark:text-white">{block.name}</h3>
          </div>
          {openBlocks.includes(block.id) ? (
            <ChevronDown className="text-gray-500 dark:text-gray-400" size={20} />
          ) : (
            <ChevronRight className="text-gray-500 dark:text-gray-400" size={20} />
          )}
        </button>

        {openBlocks.includes(block.id) && (
          <div className="p-4 bg-gray-50 dark:bg-gray-900 border-t dark:border-gray-700">
            <ul className="space-y-2">
              {block.facilities.map((facility, index) => (
                <li
                  key={index}
                  className="flex items-start space-x-2 text-gray-700 dark:text-gray-300"
                >
                  <ChevronRight size={16} className="mt-1 flex-shrink-0 text-blue-500 dark:text-blue-400" />
                  <span>{facility}</span>
                </li>
              ))}
```

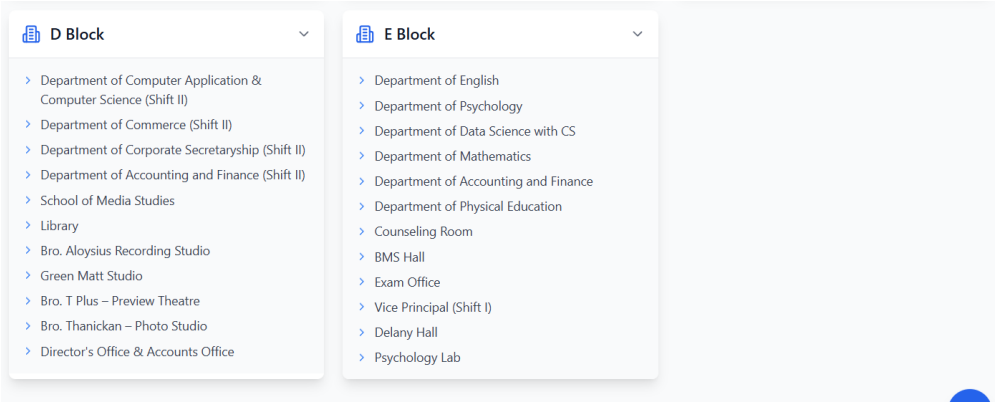
```
            </ul>
          </div>
        )}
      </div>
    ))}
  </div>

  <div className="mt-12 grid grid-cols-1 md:grid-cols-3 gap-8">
    {/* Other Important Locations */}
    <div className="bg-white dark:bg-gray-800 rounded-lg shadow-lg p-6">
      <h3 className="text-xl font-semibold mb-4 flex items-center dark:text-white">
        <Building2 className="mr-2 text-blue-600 dark:text-blue-400" />
        Other Important Locations
      </h3>
      <ul className="space-y-3">
        {otherLocations.map((location, index) => (
          <li key={index} className="flex items-center space-x-3 text-gray-700 dark:text-gray-
300">
            <location.icon size={20} className="text-blue-600 dark:text-blue-400" />
            <span>{location.name}</span>
          </li>
        ))}
      </ul>
    </div>

    {/* Parking Areas */}
    <div className="bg-white dark:bg-gray-800 rounded-lg shadow-lg p-6">
```

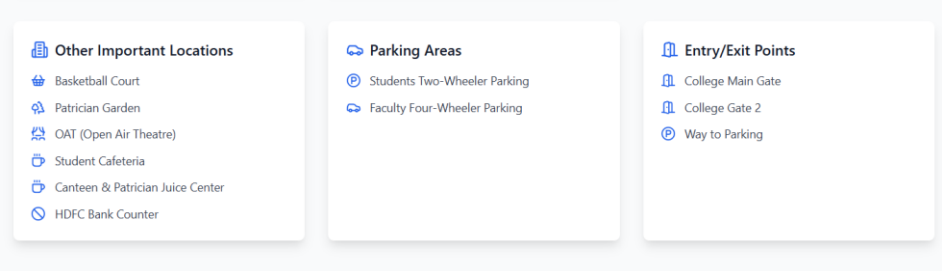
```
        <h3 className="text-xl font-semibold mb-4 flex items-center dark:text-white">
          <Car className="mr-2 text-blue-600 dark:text-blue-400" />
          Parking Areas
        </h3>
        <ul className="space-y-3">
          {parkingAreas.map((area, index) => (
            <li key={index} className="flex items-center space-x-3 text-gray-700 dark:text-gray-
300">
              <area.icon size={20} className="text-blue-600 dark:text-blue-400" />
              <span>{area.name}</span>
            </li>
          ))}
        </ul>
      </div>

      {/* Entry/Exit Points */}
      <div className="bg-white dark:bg-gray-800 rounded-lg shadow-lg p-6">
        <h3 className="text-xl font-semibold mb-4 flex items-center dark:text-white">
          <DoorOpen className="mr-2 text-blue-600 dark:text-blue-400" />
          Entry/Exit Points
        </h3>
        <ul className="space-y-3">
          {entryPoints.map((point, index) => (
            <li key={index} className="flex items-center space-x-3 text-gray-700 dark:text-gray-
300">
              <point.icon size={20} className="text-blue-600 dark:text-blue-400" />
              <span>{point.name}</span>
```

```
            </li>
          ))}
        </ul>
      </div>
    </div>
  </div>
 );
}
```

SCREENSHOTS







Chatbot.tsx

- Implements the **chatbot functionality** for assisting users.
- Handles **user queries** regarding campus navigation.
- Uses **predefined responses** to guide students and visitors.

CODE SNIPPETS

```
import React, { useState, useRef, useEffect } from 'react';

import { MessageCircle, X, Mic, Send } from 'lucide-react';

interface Message {

text: string;

isBot: boolean;

timestamp: Date;

}

// Campus knowledge base

const campusKnowledge = {

blocks: {

'a block': 'A Block houses the Department of Computer Application & Computer Science, Administrative & Academic Director\'s Office, Principal\'s Office, Computer Lab, Micro Processor Lab, and Conference Hall.',

'b block': 'B Block contains the Department of Commerce, Auditorium, Commerce Lab, and the office of Vice Principal (Shift II).',

'c block': 'C Block houses the Department of Business Administration & HRM, Fintan Hall, Board Room, Innovation Cell, Well Being Centre, Seeds Room, and Ignatius Hall.',
```

'd block': 'D Block contains multiple departments including Computer Application & Computer Science (Shift II), Commerce (Shift II), Corporate Secretaryship (Shift II), Accounting and Finance (Shift II), School of Media Studies, Library, Bro. Aloysius Recording Studio, Green Matt Studio, Bro. T Plus – Preview Theatre, Bro. Thanickan – Photo Studio, and Director\'s Office & Accounts Office.',

'e block': 'E Block houses the Department of English, Psychology, Data Science with CS, Mathematics, Accounting and Finance, Physical Education, Counseling Room, BMS Hall, Exam Office, Vice Principal (Shift I), Delany Hall, and Psychology Lab.'

},

facilities: {

'library': 'The library is located in D Block. It has a vast collection of books, journals, and digital resources.',

'cafeteria': 'The Student Cafeteria is located near the OAT (Open Air Theatre). There\'s also a Canteen & Patrician Juice Center near the College Gate 2.',

'auditorium': 'The Auditorium is located in B Block and is used for college events and functions.',

'basketball court': 'The Basketball Court is located between C Block and D Block.',

'garden': 'The Patrician Garden is located in the center of the campus, providing a green space for students.',

'oat': 'The Open Air Theatre (OAT) is located near the Student Cafeteria and is used for outdoor events.',

'parking': 'There are designated parking areas for students (two-wheeler) and faculty (four-wheeler) near the entrance.',

'bank': 'There is an HDFC Bank Counter on campus for financial services.'

},

departments: {

'computer': 'The Department of Computer Application & Computer Science is located in A Block. For Shift II, it\'s in D Block.',

'commerce': 'The Department of Commerce is located in B Block. For Shift II, it\'s in D Block.',

'business': 'The Department of Business Administration & HRM is located in C Block.',

'english': 'The Department of English is located in E Block.',

'psychology': 'The Department of Psychology is located in E Block. There\'s also a Psychology Lab in the same block.',

'mathematics': 'The Department of Mathematics is located in E Block.',

'accounting': 'The Department of Accounting and Finance is located in E Block. For Shift II, it\'s in D Block.',

'media': 'The School of Media Studies is located in D Block along with recording and photo studios.'

},

navigation: {

'entrance': 'The main entrance to the college is through the College Main Gate. There\'s also a College Gate 2 and a separate way to the parking area.',

'map': 'You can view the interactive campus map on the Campus Map page of this website.',

'directions': 'To get directions to a specific location, please use the interactive map and click on the location you want to navigate to.'

},

general: {

'about': 'Patrician College of Arts and Science is located in Chennai, Tamil Nadu. The campus has multiple blocks (A to E) housing various departments and facilities.',

'contact': 'For general inquiries, you can contact the college at +91 1234567890 or email at info@patriciancollege.edu.',

'emergency': 'For emergencies, contact Security at +91 9876543210 or Medical Emergency at +91 9876543211.'

}

};

// AI response generator

const generateAIResponse = (query: string): string => {

const lowerQuery = query.toLowerCase();

// Check for greetings

if (/^(hi|hello|hey|greetings)/.test(lowerQuery)) {

return "Hello! I'm PatriGuide Assistant. How can I help you navigate Patrician College campus today?";

}

```
// Check for thanks
if (/thank|thanks/.test(lowerQuery)) {
return "You're welcome! Feel free to ask if you need any more information about our campus.";
}
// Check for blocks
for (const [key, value] of Object.entries(campusKnowledge.blocks)) {
if (lowerQuery.includes(key)) {
return value;
}
}
// Check for facilities
for (const [key, value] of Object.entries(campusKnowledge.facilities)) {
if (lowerQuery.includes(key)) {
return value;
}
}
// Check for departments
for (const [key, value] of Object.entries(campusKnowledge.departments)) {
if (lowerQuery.includes(key)) {
return value;
}
}
// Check for navigation
for (const [key, value] of Object.entries(campusKnowledge.navigation)) {
if (lowerQuery.includes(key)) {
return value;
```

```
}

}

// Check for specific questions

if (lowerQuery.includes('where is') || lowerQuery.includes('how to find') || lowerQuery.includes('locate')) {

if (lowerQuery.includes('library')) return campusKnowledge.facilities.library;

if (lowerQuery.includes('cafeteria') || lowerQuery.includes('canteen') || lowerQuery.includes('food'))

return campusKnowledge.facilities.cafeteria;

if (lowerQuery.includes('auditorium')) return campusKnowledge.facilities.auditorium;

if (lowerQuery.includes('basketball') || lowerQuery.includes('court'))

return campusKnowledge.facilities.basketball;

if (lowerQuery.includes('garden')) return campusKnowledge.facilities.garden;

if (lowerQuery.includes('oat') || lowerQuery.includes('theatre') || lowerQuery.includes('theater'))

return campusKnowledge.facilities.oat;

if (lowerQuery.includes('parking')) return campusKnowledge.facilities.parking;

if (lowerQuery.includes('bank') || lowerQuery.includes('hdfc'))

return campusKnowledge.facilities.bank;

// Generic location response

return "I can help you locate that. Please check the interactive campus map on the Campus Map page, or be more specific about what you're looking for.";

}

// Check for general information

for (const [key, value] of Object.entries(campusKnowledge.general)) {

if (lowerQuery.includes(key)) {

return value;

}

}
```

```
// Default response

return "I'm not sure about that specific information. I can help you with locations of blocks, departments, facilities, and navigation around Patrician College campus. Could you please rephrase your question?";

};

export default function Chatbot() {

const [isOpen, setIsOpen] = useState(false);

const [messages, setMessages] = useState<Message[]>([

{ text: "Hello! How can I help you navigate the Patrician College campus today?", isBot: true, timestamp: new Date() }

]);

const [inputText, setInputText] = useState("");

const [isTyping, setIsTyping] = useState(false);

const messagesEndRef = useRef<HTMLDivElement>(null);

const scrollToBottom = () => {

messagesEndRef.current?.scrollIntoView({ behavior: "smooth" });

};

useEffect(() => {

scrollToBottom();

}, [messages]);

const handleSend = () => {

if (!inputText.trim()) return;

// Add user message

setMessages(prev => [...prev, { text: inputText, isBot: false, timestamp: new Date() }]);

setInputText("");

setIsTyping(true);

// Generate AI response with a realistic delay

setTimeout(() => {
```

```
const aiResponse = generateAIResponse(inputText);

setMessages(prev => [...prev, {

text: aiResponse,

isBot: true,

timestamp: new Date()

}]);

setIsTyping(false);

}, 1500);

};

const handleVoiceInput = () => {

// This is a placeholder for voice recognition functionality

// In a real implementation, you would use the Web Speech API

alert("Voice recognition would be implemented here. This feature is coming soon!");

};

return (

<div className="fixed bottom-6 right-6 z-50">

{isOpen ? (

<div className="bg-white dark:bg-gray-800 rounded-lg shadow-xl w-80 h-96 flex flex-col animate-slide-up">

<div className="p-4 bg-blue-600 text-white rounded-t-lg flex justify-between items-center">

<div className="flex items-center space-x-2">

<div className="w-2 h-2 rounded-full bg-green-400 animate-pulse"></div>

<h3 className="font-semibold">PatriGuide Assistant</h3>

</div>

<button

onClick={() => setIsOpen(false)}

className="hover:bg-blue-700 p-1 rounded-full transition-colors"
```

```
>
<X size={20} />
</button>
</div>
<div className="flex-1 p-4 overflow-y-auto space-y-4">
{messages.map((msg, index) => (
<div
key={index}
className={`flex ${msg.isBot ? 'justify-start' : 'justify-end'}`}
>
<div
className={`rounded-lg p-3 max-w-[80%] ${
msg.isBot
? 'bg-blue-100 text-blue-900 dark:bg-blue-900 dark:text-blue-100'
: 'bg-blue-600 text-white'
}`}
>
{msg.text}
</div>
</div>
))}
{isTyping && (
<div className="flex justify-start">
<div className="bg-gray-100 dark:bg-gray-700 rounded-lg p-3 flex space-x-1">
<div className="w-2 h-2 bg-gray-400 rounded-full animate-bounce"></div>
<div className="w-2 h-2 bg-gray-400 rounded-full animate-bounce" style={{ animationDelay:
'0.2s' }}></div>
```

<div className="w-2 h-2 bg-gray-400 rounded-full animate-bounce" style={{ animationDelay: '0.4s' }}>

)}

<input

type="text"

value={inputText}

onChange={(e) => setInputText(e.target.value)}

onKeyPress={(e) => e.key === 'Enter' && handleSend()}

placeholder="Ask about campus locations..."

className="flex-1 px-3 py-2 border rounded-lg focus:outline-none focus:border-blue-500 dark:bg-gray-700 dark:border-gray-600 dark:text-white"

/>

<button

onClick={handleSend}

className="p-2 text-blue-600 hover:bg-blue-50 rounded-lg transition-colors dark:text-blue-400 dark:hover:bg-gray-700"

aria-label="Send message"

>

<Send size={20} />

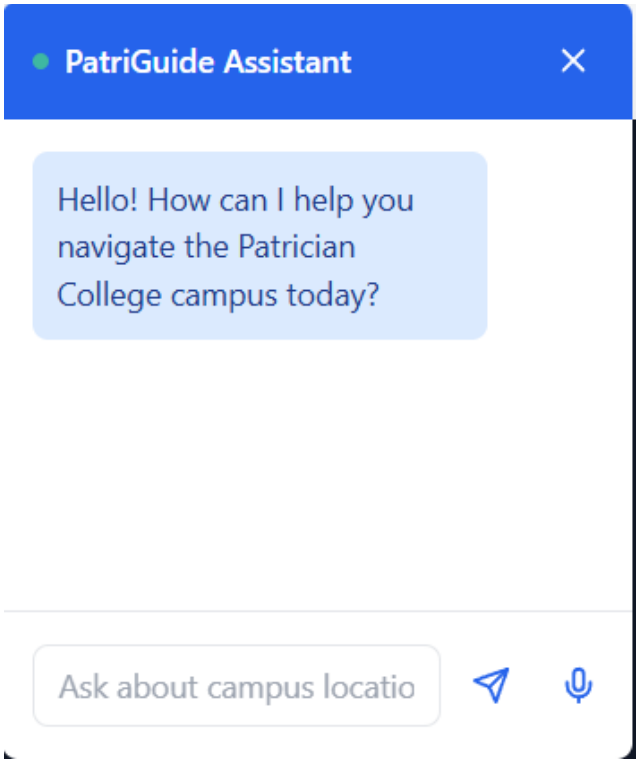
</button>

<button

onClick={handleVoiceInput}

```
className="p-2 text-blue-600 hover:bg-blue-50 rounded-lg transition-colors dark:text-blue-400 dark:hover:bg-gray-700"

aria-label="Voice input"

>

<Mic size={20} />

</button>

</div>

</div>

</div>

) : (

<button

onClick={() => setIsOpen(true)}

className="bg-blue-600 text-white p-4 rounded-full shadow-lg hover:bg-blue-700 transition-all hover:scale-110 animate-bounce-slow"

aria-label="Open chat assistant"

>

<MessageCircle size={24} />

</button>

)}

</div>

);

}
```

SCREENSHOTS



Footer.tsx

- Defines the **footer section** of the website, containing **copyright information, contact details, and quick links**.

CODE SNIPPETS

```
import React from 'react';

import { Facebook, Twitter, Instagram, Mail, Phone, MapPin } from 'lucide-react';

export default function Footer() {

return (

<footer className="bg-gray-900 text-gray-300">

<div className="container mx-auto px-4 py-12">

<div className="grid grid-cols-1 md:grid-cols-3 gap-8">

<div>
```

```
<div className="flex items-center space-x-3 mb-4">
<img
src="/assets/img/Patics_Logo.png"
alt="Patrician College Logo"
className="h-12 w-12"
/>
<div className="flex flex-col">
<h3 className="text-xl font-bold text-white">Patrician College</h3>
<span className="text-sm text-gray-400">of Arts and Science</span>
</div>
</div>
<div className="space-y-2">
<p className="flex items-center">
<MapPin size={20} className="mr-2" />
Canal Bank Road, Adyar, Chennai - 600020
</p>
<p className="flex items-center">
<Phone size={20} className="mr-2" />
+91 1234567890
</p>
<p className="flex items-center">
<Mail size={20} className="mr-2" />
info@patriciancollege.edu
</p>
</div>
</div>
```

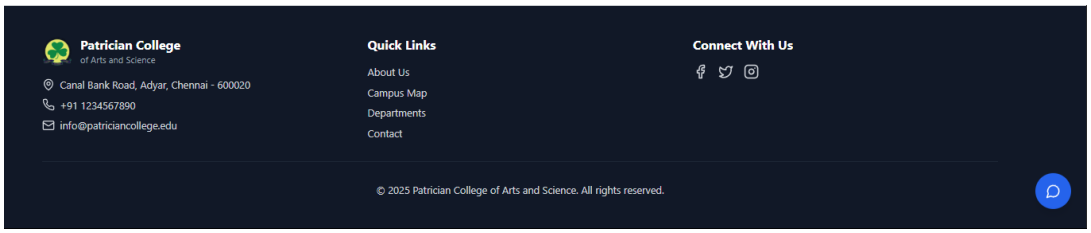
```
<div>
<h3 className="text-xl font-bold text-white mb-4">Quick Links</h3>
<ul className="space-y-2">
<li><a href="#" className="hover:text-white">About Us</a></li>
<li><a href="#" className="hover:text-white">Campus Map</a></li>
<li><a href="#" className="hover:text-white">Departments</a></li>
<li><a href="#" className="hover:text-white">Contact</a></li>
</ul>
</div>
<div>
<h3 className="text-xl font-bold text-white mb-4">Connect With Us</h3>
<div className="flex space-x-4">
<a href="#" className="hover:text-white">
<Facebook size={24} />
</a>
<a href="#" className="hover:text-white">
<Twitter size={24} />
</a>
<a href="#" className="hover:text-white">
<Instagram size={24} />
</a>
</div>
</div>
</div>
<div className="border-t border-gray-800 mt-8 pt-8 text-center">
<p>&copy; {new Date().getFullYear()} Patrician College of Arts and Science. All rights reserved.</p>
```

</footer>

);

}

SCREENSHOTS



Header.tsx

- Contains the **navigation bar** and the **title/logo of the project**.
- Ensures smooth navigation between different pages of the application.

CODE SNIPPETS

import React, { useState } from 'react';

import { Menu, X, Globe, ChevronDown } from 'lucide-react';

import { Link, useLocation } from 'react-router-dom';

const languages = [

{ code: 'en', label: 'English' },

{ code: 'ta', label: 'தமிழ்' },

{ code: 'hi', label: 'हिंदी' }

];

export default function Header() {

const [isOpen, setIsOpen] = useState(false);

```
const [currentLang, setCurrentLang] = useState('en');

const location = useLocation();

const isActive = (path: string) => location.pathname === path;

return (

<header className="bg-white shadow-md fixed w-full top-0 z-50 dark:bg-gray-800 dark:text-white">

<nav className="container mx-auto px-4 py-3">

<div className="flex items-center justify-between">

<Link to="/" className="flex items-center space-x-3">

<img

src="/assets/img/Patics_Logo.png"

alt="Patrician College Logo"

className="h-16 w-16"

/>

<div className="flex flex-col">

<span className="text-blue-600 text-xl font-bold dark:text-blue-400">Patrician College</span>

<span className="text-sm text-gray-600 dark:text-gray-300">of Arts and Science</span>

</div>

</Link>

<div className="hidden md:flex items-center space-x-8">

<Link

to="/"

className={`text-gray-700 hover:text-blue-600 transition-colors dark:text-gray-300 dark:hover:text-blue-400 ${isActive('/') && 'text-blue-600 font-semibold dark:text-blue-400'}`}

>

Home

</Link>
```

<Link

to="/about"

className={`text-gray-700 hover:text-blue-600 transition-colors dark:text-gray-300 dark:hover:text-blue-400 ${isActive('/about') && 'text-blue-600 font-semibold dark:text-blue-400'}`}

>

About

</Link>

<Link

to="/campus-map"

className={`text-gray-700 hover:text-blue-600 transition-colors dark:text-gray-300 dark:hover:text-blue-400 ${isActive('/campus-map') && 'text-blue-600 font-semibold dark:text-blue-400'}`}

>

Campus Map

</Link>

<Link

to="/contact"

className={`text-gray-700 hover:text-blue-600 transition-colors dark:text-gray-300 dark:hover:text-blue-400 ${isActive('/contact') && 'text-blue-600 font-semibold dark:text-blue-400'}`}

>

Contact

</Link>

<button className="flex items-center space-x-1 text-gray-700 hover:text-blue-600 dark:text-gray-300 dark:hover:text-blue-400">

<Globe size={20} />

{languages.find(l => l.code === currentLang)?.label}

<ChevronDown size={16} />

</button>

{languages.map((lang) => (

<button

key={lang.code}

onClick={() => setCurrentLang(lang.code)}

className="block w-full text-left px-4 py-2 text-gray-700 hover:bg-blue-50 dark:text-gray-300 dark:hover:bg-gray-600"

>

{lang.label}

</button>

))}

<button

className="md:hidden"

onClick={() => setIsOpen(!isOpen)}

>

{isOpen ? <X size={24} /> : <Menu size={24} />}

</button>

{/* Mobile menu */}

{isOpen && (

```
<div className="flex flex-col space-y-4">
<Link
to="/"
className={`text-gray-700 hover:text-blue-600 dark:text-gray-300 dark:hover:text-blue-400 ${isActive('/') && 'text-blue-600 font-semibold dark:text-blue-400'}`}
onClick={() => setIsOpen(false)}
>
Home
</Link>
<Link
to="/about"
className={`text-gray-700 hover:text-blue-600 dark:text-gray-300 dark:hover:text-blue-400 ${isActive('/about') && 'text-blue-600 font-semibold dark:text-blue-400'}`}
onClick={() => setIsOpen(false)}
>
About
</Link>
<Link
to="/campus-map"
className={`text-gray-700 hover:text-blue-600 dark:text-gray-300 dark:hover:text-blue-400 ${isActive('/campus-map') && 'text-blue-600 font-semibold dark:text-blue-400'}`}
onClick={() => setIsOpen(false)}
>
Campus Map
</Link>
<Link
to="/contact"
```

className={`text-gray-700 hover:text-blue-600 dark:text-gray-300 dark:hover:text-blue-400 ${isActive('/contact') && 'text-blue-600 font-semibold dark:text-blue-400'}`}

onClick={() => setIsOpen(false)}

>

Contact

</Link>

Select Language

{languages.map((lang) => (

<button

key={lang.code}

onClick={() => {

setCurrentLang(lang.code);

setIsOpen(false);

}}

className="block w-full text-left py-2 text-gray-700 hover:bg-blue-50 dark:text-gray-300 dark:hover:bg-gray-600"

>

{lang.label}

</button>

))}

)}

</nav>

</header>

);

}

SCREENSHOTS

Hero.tsx

- Implements the **landing page section** with an introduction to the PatriGuide project.
- Displays a **welcome message and an overview** of the navigation system.

CODE SNIPPETS

```
import React from "react";

import { Navigation } from "lucide-react";

import { useNavigate } from "react-router-dom";

export default function Hero() {

const navigate = useNavigate();

return (

<div className="relative h-screen mt-16">

{/* Background Image */}

<div className="absolute inset-0">

<img

src="/assets/img/Campus_img.jpg" // ✅ Corrected Image Path

alt="College Campus"

className="w-full h-full object-cover"

/>

<div className="absolute inset-0 bg-blue-900/40"></div>
```

```
</div>

{/* Hero Content */}

<div className="relative container mx-auto px-4 h-full flex items-center">

<div className="max-w-3xl text-white">

<h1 className="text-6xl font-bold mb-8 leading-tight">

Explore Patrician College with Smart Navigation!

</h1>

<p className="text-2xl mb-10 leading-relaxed">

Find your way around campus easily with our intelligent navigation system.

Available in English, தமிழ், and हिंदी.

</p>

<button

onClick={() => navigate("/campus-map")}

className="bg-blue-600 hover:bg-blue-700 text-white px-10 py-4 rounded-lg flex items-center space-x-3 transform transition hover:scale-105 text-xl"

>

<Navigation size={24} />

<span>Start Navigation</span>

</button>

</div>

</div>

</div>

);

}
```

SCREENSHOTS

Map.tsx

- Handles the **interactive campus map** functionality.
- Enables **zooming, panning, and clicking** on locations to display details

CODE SNIPPETS

```
import React, { useState } from 'react';

import { ZoomIn, ZoomOut, Compass, Navigation2 } from 'lucide-react';

interface MapMarker {

id: number;

name: string;

description: string;

x: number;

y: number;

}

const markers: MapMarker[] = [

{ id: 1, name: 'A Block', description: 'Department of Computer Application & Computer Science', x: 10, y: 30 },
```

{ id: 2, name: 'B Block', description: 'Department of Commerce', x: 32, y: 48 },

{ id: 3, name: 'C Block', description: 'Department of Business Administration & HRM', x: 52, y: 55 },

{ id: 4, name: 'D Block', description: 'School of Media Studies & Library', x: 45, y: 75 },

{ id: 5, name: 'E Block', description: 'Department of English & Psychology', x: 78, y: 85 },

{ id: 6, name: 'OAT', description: 'Open Air Theatre', x: 34, y: 25 },

];

export default function Map() {

const [activeMarker, setActiveMarker] = useState<MapMarker | null>(null);

const [showDirections, setShowDirections] = useState(false);

const [scale, setScale] = useState(1);

const handleZoomIn = () => setScale(prev => Math.min(prev + 0.2, 2));

const handleZoomOut = () => setScale(prev => Math.max(prev - 0.2, 0.5));

const handleReset = () => setScale(1);

return (

<h2 className="text-2xl font-semibold dark:text-white">Interactive Campus Map</h2>

<button onClick={handleZoomIn} className="p-3 hover:bg-gray-200 dark:hover:bg-gray-600 rounded-lg">

<ZoomIn size={24} className="dark:text-white" />

</button>

<button onClick={handleZoomOut} className="p-3 hover:bg-gray-200 dark:hover:bg-gray-600 rounded-lg">

<ZoomOut size={24} className="dark:text-white" />

</button>

<button onClick={handleReset} className="p-3 hover:bg-gray-200 dark:hover:bg-gray-600 rounded-lg">

<Compass size={24} className="dark:text-white" />

</button>

{/* Map Container */}

<div className="relative w-full" style={{ transform: `scale(${scale})`, transformOrigin: 'center', transition: 'transform 0.3s ease' }}>

<img

src="/assets/img/Campus_Map.png"

alt="Patrician College Campus Map"

className="w-full h-auto object-cover"

/>

{/* Markers */}

{markers.map((marker) => (

<div

key={marker.id}

className="absolute cursor-pointer transform -translate-x-1/2 -translate-y-1/2"

style={{ left: `${marker.x}%`, top: `${marker.y}%`, transform: `scale(${1/scale})` }}

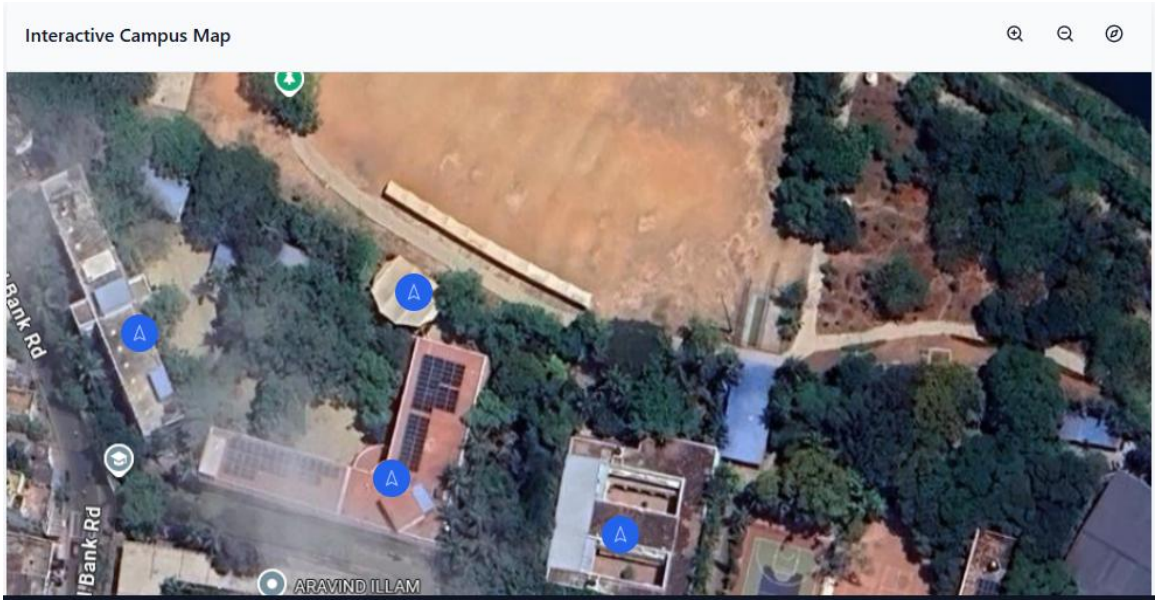
```
onMouseEnter={() => setActiveMarker(marker)}

onMouseLeave={() => setActiveMarker(null)}

>

<div className="p-3 bg-blue-600 rounded-full text-white hover:bg-blue-700">

<Navigation2 size={24} className="animate-pulse" />

</div>

{activeMarker?.id === marker.id && (

<div className="absolute bottom-full left-1/2 transform -translate-x-1/2 mb-3 w-64 bg-white dark:bg-gray-800 rounded-lg shadow-lg p-4 z-10">

<h4 className="text-xl font-semibold text-gray-900 dark:text-white">{marker.name}</h4>

<p className="text-base text-gray-600 dark:text-gray-300 mt-2">{marker.description}</p>

<button

onClick={() => setShowDirections(true)}

className="mt-3 text-blue-600 dark:text-blue-400 hover:text-blue-700 flex items-center"

>

<Navigation2 size={18} />

<span>Get Directions</span>

</button>

</div>

)}

</div>

))}

{/* Directions Modal */}

{showDirections && (
```

```
<div className="absolute inset-0 bg-black/50 flex items-center justify-center">
<div className="bg-white dark:bg-gray-800 p-8 rounded-lg max-w-lg w-full mx-4">
<h3 className="text-2xl font-semibold mb-4 dark:text-white">Getting Directions</h3>
<p className="text-lg text-gray-600 dark:text-gray-300 mb-6">
Use your current location to get directions to {activeMarker?.name}?
</p>
<div className="flex justify-end space-x-4">
<button
onClick={() => setShowDirections(false)}
className="px-6 py-3 text-gray-600 dark:text-gray-300 hover:bg-gray-100 dark:hover:bg-gray-700 rounded-lg"
>
Cancel
</button>
<button
onClick={() => setShowDirections(false)}
className="px-6 py-3 bg-blue-600 text-white rounded-lg hover:bg-blue-700"
>
Navigate
</button>
</div>
</div>
</div>
)}
</div>
```

);

}

SCREENSHOTS.



SearchSection.tsx

- Provides **text-based navigation** from the **entrance (A Block)** to the selected destination.
- Generates **step-by-step directions** without requiring a search bar.

CODE SNIPPETS

import React, { useState } from 'react';

import { Search, MapPin, Building2, Phone, Clock, History, Navigation } from 'lucide-react';

const markers = [

{ id: 1, name: 'A Block', description: 'Department of Computer Application & Computer Science', x: 10, y: 30 },

{ id: 2, name: 'B Block', description: 'Department of Commerce', x: 32, y: 48 },

```
{ id: 3, name: 'C Block', description: 'Department of Business Administration & HRM', x: 52, y: 55 },

{ id: 4, name: 'D Block', description: 'School of Media Studies & Library', x: 45, y: 75 },

{ id: 5, name: 'E Block', description: 'Department of English & Psychology', x: 78, y: 85 },

{ id: 6, name: 'OAT', description: 'Open Air Theatre', x: 34, y: 25 },

];

const campusKnowledge = {

entrance: { name: 'Main Entrance (A Block Side)', x: 10, y: 30 },

facilities: {

'Library': { location: 'D Block', directions: 'Walk straight past B Block and reach D Block on your right.' },

'Cafeteria': { location: 'Near OAT', directions: 'Walk towards the Open Air Theatre, the Cafeteria is beside it.' },

'Exam Halls': { location: 'Various Blocks', directions: 'Exam halls are located in multiple blocks. Please check the exam notice for details.' },

'Admin Block': { location: 'A Block', directions: 'The Admin Block is right at the entrance of A Block.' },

}

};

export default function SearchSection() {

const [searchText, setSearchText] = useState('');

const [showSuggestions, setShowSuggestions] = useState(false);

const [recentSearches, setRecentSearches] = useState<string[]>([]);

const [selectedLocation, setSelectedLocation] = useState<string | null>(null);

const [directions, setDirections] = useState<string | null>(null);

const allLocations = [...markers.map(m => m.name), ...Object.keys(campusKnowledge.facilities)];
```

```
const filteredSuggestions = allLocations.filter(location =>
location.toLowerCase().includes(searchText.toLowerCase())
);
const handleSearch = (location: string) => {
setSearchText(location);
setSelectedLocation(location);
setShowSuggestions(false);
if (!recentSearches.includes(location)) {
setRecentSearches(prev => [location, ...prev].slice(0, 5));
}
generateDirections(location);
};
const generateDirections = (destination: string) => {
const entrance = campusKnowledge.entrance;
const destinationInfo = markers.find(m => m.name === destination) || campusKnowledge.facilities[destination];
if (destinationInfo) {
setDirections(`Start at the Main Entrance near A Block. ${destinationInfo.directions}`);
} else {
setDirections('Location not found. Please select a valid destination.');
}
};
return (
<div className="container mx-auto px-4 py-16">
<div className="max-w-3xl mx-auto">
```

<input

type="text"

value={searchText}

onChange={(e) => {

setSearchText(e.target.value);

setShowSuggestions(true);

}}

onFocus={() => setShowSuggestions(true)}

placeholder="Search for locations, departments, or facilities..."

className="w-full px-6 py-4 rounded-lg border-2 border-gray-200 focus:border-blue-500 focus:outline-none pl-14 dark:bg-gray-700 dark:border-gray-600 dark:text-white dark:placeholder-gray-400"

/>

<Search className="absolute left-4 top-1/2 transform -translate-y-1/2 text-gray-400" size={24} />

{showSuggestions && (searchText || recentSearches.length > 0) && (

{searchText && filteredSuggestions.length > 0 && (

<Search size={16} className="mr-2" />

Suggestions

{filteredSuggestions.map((location, index) => (

<button

```
key={index}

onClick={() => handleSearch(location)}

className="w-full text-left px-3 py-2 hover:bg-blue-50 dark:hover:bg-gray-700 rounded-lg transition-colors dark:text-white"

>

{location}

</button>

))}

</div>

)}

{recentSearches.length > 0 && (

<div className="p-2 border-t dark:border-gray-700">

<div className="flex items-center px-3 py-2 text-sm text-gray-500 dark:text-gray-400">

<History size={16} className="mr-2" />

<span>Recent Searches</span>

</div>

{recentSearches.map((search, index) => (

<button

key={index}

onClick={() => handleSearch(search)}

className="w-full text-left px-3 py-2 hover:bg-blue-50 dark:hover:bg-gray-700 rounded-lg transition-colors flex items-center dark:text-white"

>

<Clock size={16} className="mr-2 text-gray-400" />

{search}

</button>
```

```
))}

</div>

)}

</div>

)}

</div>

{selectedLocation && directions && (

<div className="mt-8 bg-blue-50 dark:bg-gray-700 border border-blue-200 dark:border-gray-600 rounded-lg p-4">

<h3 className="text-lg font-semibold text-blue-700 dark:text-blue-400 flex items-center">

<Navigation className="mr-2" size={20} />

Directions to {selectedLocation}

</h3>

<p className="mt-2 text-blue-600 dark:text-blue-400">{directions}</p>

</div>

)}

<div className="mt-8">

<h2 className="text-xl font-semibold mb-4 dark:text-white">Popular Destinations</h2>

<div className="grid grid-cols-2 md:grid-cols-4 gap-4">

{Object.keys(campusKnowledge.facilities).map((dest, index) => (

<button

key={index}

onClick={() => handleSearch(dest)}
```

className="flex flex-col items-center p-4 rounded-lg border border-gray-200 dark:border-gray-700 hover:border-blue-500 hover:bg-blue-50 dark:hover:bg-gray-700 transition-all hover:scale-105 dark:bg-gray-800 dark:text-white"

>

<MapPin size={24} className="text-blue-600 dark:text-blue-400 mb-2" />

{dest}

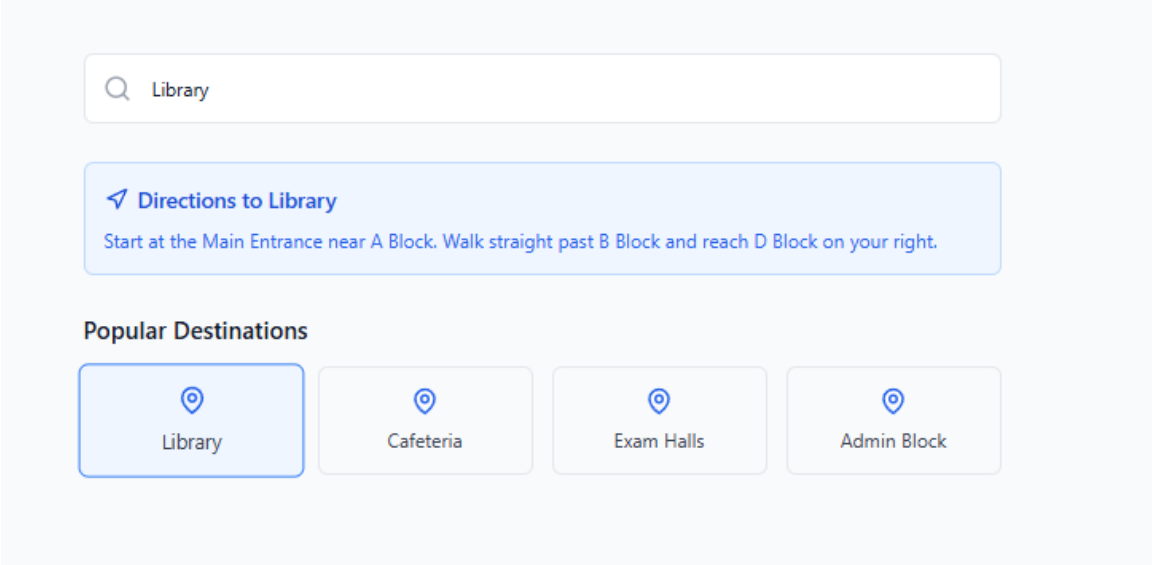
</button>

))}

);

}

SCREENSHOTS



8.2.3 Pages

These files represent different pages within the project.

AboutPage.tsx

- Displays **details about the project, its purpose, and motivation**.

CODE SNIPPETS

```
import React from "react";

import { Building2, Navigation, Users, Clock } from "lucide-react";

export default function AboutPage() {

return (

<main className="pt-24">

<div className="container mx-auto px-4">

<div className="max-w-4xl mx-auto">

<h1 className="text-4xl font-bold mb-8 dark:text-white">About PatriGuide</h1>

<div className="mb-12">

{/* ✅ Updated Image Path */}

<img

src="/assets/img/Campus_about.jpeg"

alt="Patrician College Campus"

className="w-full h-[400px] object-cover rounded-lg shadow-lg mb-8"

/>

<p className="text-lg text-gray-700 dark:text-gray-300 mb-6">

Welcome to PatriGuide, your smart navigation companion at Patrician College. Our system is designed to help students, staff, and visitors navigate the campus efficiently and effectively.
```

<Building2 className="text-blue-600 dark:text-blue-400 mr-3" size={24} />

<h2 className="text-xl font-semibold dark:text-white">Our Campus</h2>

Patrician College boasts modern facilities spread across multiple blocks, each dedicated to different academic departments and activities.

<Navigation className="text-blue-600 dark:text-blue-400 mr-3" size={24} />

<h2 className="text-xl font-semibold dark:text-white">Smart Navigation</h2>

Our interactive map and AI-powered chatbot help you find the shortest route to your destination within the campus.

<Users className="text-blue-600 dark:text-blue-400 mr-3" size={24} />

<h2 className="text-xl font-semibold dark:text-white">Accessibility</h2>

PatriGuide supports multiple languages and provides voice navigation to ensure everyone can use it effectively.

<Clock className="text-blue-600 dark:text-blue-400 mr-3" size={24} />

<h2 className="text-xl font-semibold dark:text-white">24/7 Assistance</h2>

Our virtual assistant is available round the clock to help you with directions and campus information.

<h2 className="text-2xl font-semibold mb-4 dark:text-white">Our Mission</h2>

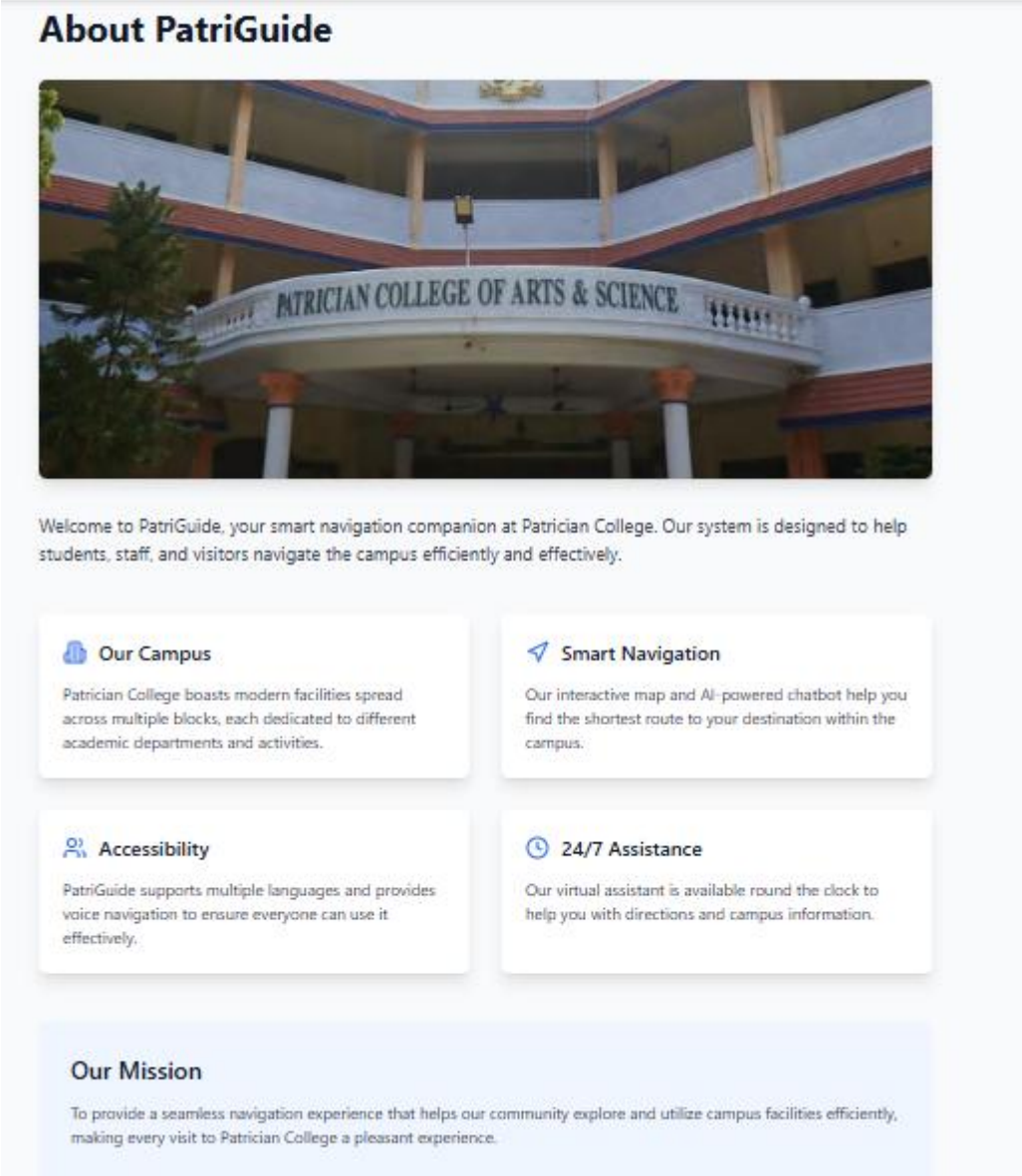
To provide a seamless navigation experience that helps our community explore and utilize campus facilities efficiently, making every visit to Patrician College a pleasant experience.

</main>

);

}

SCREENSHOTS



CampusMapPage.tsx

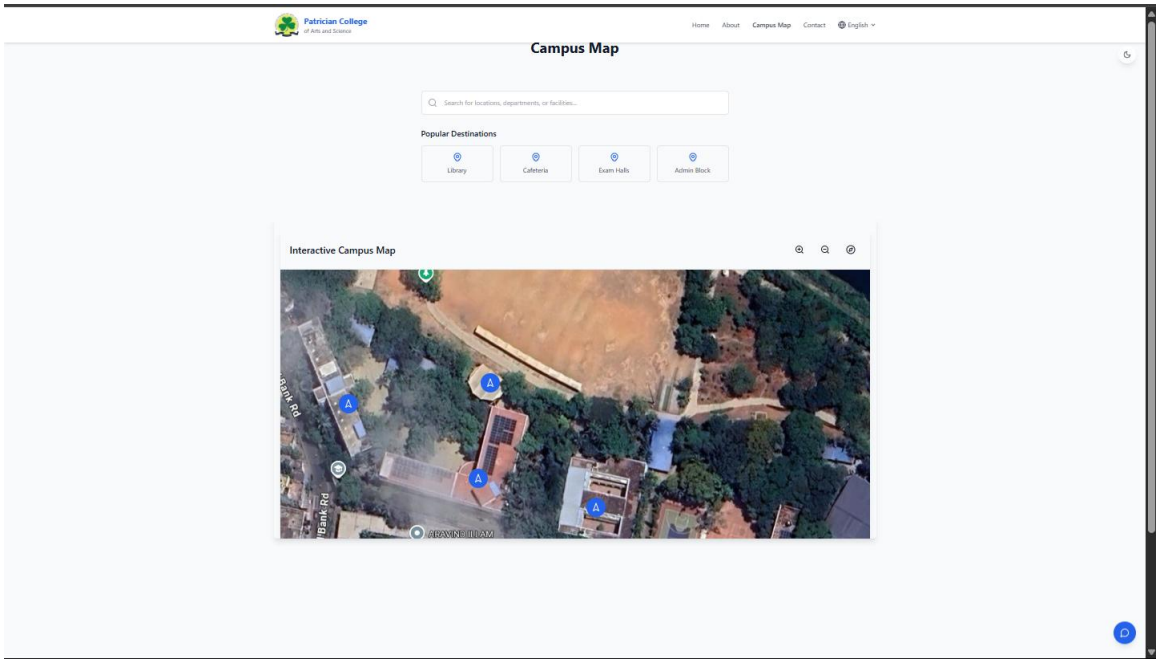
- Hosts the **interactive map** for campus navigation.

CODE SNIPPETS

```
import React from 'react';

import Map from '../components/Map';

import SearchSection from '../components/SearchSection';

export default function CampusMapPage() {

return (

<main className="pt-20 min-h-screen bg-gray-50 dark:bg-gray-900">

<div className="container mx-auto px-4">

{/* Page Title */}

<h1 className="text-4xl font-bold text-center mb-6 text-gray-900 dark:text-white">

Campus Map

</h1>

{/* Search Section */}

<div className="mb-6">

<SearchSection />

</div>

{/* Map Container */}

<div className="relative w-full h-[700px] md:h-[800px] shadow-lg rounded-lg overflow-hidden">

<Map />

</div>

</div>

</main>

);
```

}

SCREENSHOTS



ContactPage.tsx

- Provides **contact details** and a **form for inquiries**.

CODE SNIPPETS

```
import React, { useState } from 'react';
import { MapPin, Phone, Mail, Facebook, Twitter, Instagram, Send } from 'lucide-react';
export default function ContactPage() {
const [formData, setFormData] = useState({
name: '',
email: '',
message: ''
});
const handleSubmit = (e: React.FormEvent) => {
e.preventDefault();
```

```
// Handle form submission
console.log('Form submitted:', formData);
alert('Thank you for your message! We will get back to you soon.');
setFormData({ name: '', email: '', message: '' });
};
return (
<main className="pt-24">
<div className="container mx-auto px-4">
<div className="max-w-5xl mx-auto">
<h1 className="text-4xl font-bold mb-8 dark:text-white">Contact Us</h1>
<div className="grid grid-cols-1 md:grid-cols-2 gap-12">
<div>
<div className="bg-white dark:bg-gray-800 rounded-lg shadow-lg p-6 mb-8">
<h2 className="text-2xl font-semibold mb-6 dark:text-white">Get in Touch</h2>
<div className="space-y-4">
<div className="flex items-start space-x-4">
<MapPin className="text-blue-600 dark:text-blue-400 mt-1" size={20} />
<div>
<h3 className="font-semibold dark:text-white">Address</h3>
<p className="text-gray-600 dark:text-gray-300">
123 College Road<br />
Chennai, Tamil Nadu<br />
India
</p>
</div>
```

```
</div>
<div className="flex items-start space-x-4">
<Phone className="text-blue-600 dark:text-blue-400 mt-1" size={20} />
<div>
<h3 className="font-semibold dark:text-white">Phone</h3>
<p className="text-gray-600 dark:text-gray-300">+91 1234567890</p>
</div>
</div>
<div className="flex items-start space-x-4">
<Mail className="text-blue-600 dark:text-blue-400 mt-1" size={20} />
<div>
<h3 className="font-semibold dark:text-white">Email</h3>
<p className="text-gray-600 dark:text-gray-300">info@patriciancollege.edu</p>
</div>
</div>
</div>
</div>
<div className="bg-red-50 dark:bg-gray-700 rounded-lg shadow-lg p-6">
<h2 className="text-2xl font-semibold mb-4 text-red-700 dark:text-red-400">Emergency Contacts</h2>
<div className="space-y-3">
<div className="flex items-center space-x-3">
<Phone className="text-red-600" size={20} />
<div>
<p className="font-semibold dark:text-white">Security</p>
```

+91 9876543210

<Phone className="text-red-600" size={20} />

Medical Emergency

+91 9876543211

<h2 className="text-2xl font-semibold mb-6 dark:text-white">Send us a Message</h2>

<form onSubmit={handleSubmit} className="space-y-6">

<label htmlFor="name" className="block text-sm font-medium text-gray-700 dark:text-gray-300 mb-1">

Name

</label>

<input

type="text"

id="name"

value={formData.name}

onChange={(e) => setFormData({ ...formData, name: e.target.value })}

className="w-full px-4 py-2 rounded-lg border border-gray-300 focus:ring-2 focus:ring-blue-500 focus:border-blue-500 dark:bg-gray-700 dark:border-gray-600 dark:text-white"

required

/>

<label htmlFor="email" className="block text-sm font-medium text-gray-700 dark:text-gray-300 mb-1">

Email

</label>

<input

type="email"

id="email"

value={formData.email}

onChange={(e) => setFormData({ ...formData, email: e.target.value })}

className="w-full px-4 py-2 rounded-lg border border-gray-300 focus:ring-2 focus:ring-blue-500 focus:border-blue-500 dark:bg-gray-700 dark:border-gray-600 dark:text-white"

required

/>

<label htmlFor="message" className="block text-sm font-medium text-gray-700 dark:text-gray-300 mb-1">

Message

</label>

<textarea

id="message"

value={formData.message}

onChange={(e) => setFormData({ ...formData, message: e.target.value })}

rows={5}

className="w-full px-4 py-2 rounded-lg border border-gray-300 focus:ring-2 focus:ring-blue-500 focus:border-blue-500 dark:bg-gray-700 dark:border-gray-600 dark:text-white"

required

></textarea>

<button

type="submit"

className="w-full bg-blue-600 text-white py-3 rounded-lg hover:bg-blue-700 transition-colors flex items-center justify-center space-x-2"

>

<Send size={20} />

Send Message

</button>

</form>

<h3 className="text-xl font-semibold mb-4 dark:text-white">Connect With Us</h3>

<a

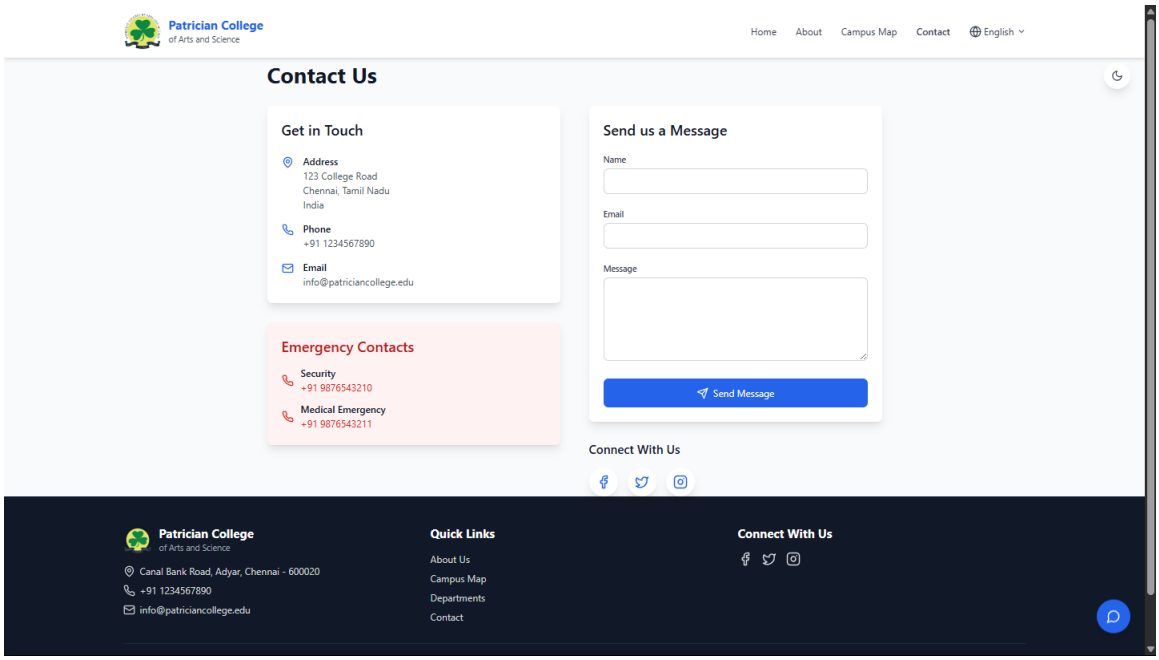
href="#"

```
className="p-3 bg-white dark:bg-gray-800 rounded-full shadow-lg hover:bg-blue-50 dark:hover:bg-gray-700 transition-colors"

>

<Facebook className="text-blue-600 dark:text-blue-400" size={24} />

</a>

<a

href="#"

className="p-3 bg-white dark:bg-gray-800 rounded-full shadow-lg hover:bg-blue-50 dark:hover:bg-gray-700 transition-colors"

>

<Twitter className="text-blue-600 dark:text-blue-400" size={24} />

</a>

<a

href="#"

className="p-3 bg-white dark:bg-gray-800 rounded-full shadow-lg hover:bg-blue-50 dark:hover:bg-gray-700 transition-colors"

>

<Instagram className="text-blue-600 dark:text-blue-400" size={24} />

</a>

</div>

</div>

</div>

</div>

</div>

</div>

</main>
```

);

}

SCREENSHOTS



8.2.4 Configuration and Project Setup

App.tsx

- The **core file** of the application, where all **components and pages are integrated**.

CODE SNIPPETS

```
import React, { useState, useEffect } from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import Header from './components/Header';
import HomePage from './pages/HomePage';
import AboutPage from './pages/AboutPage';
import CampusMapPage from './pages/CampusMapPage';
import ContactPage from './pages/ContactPage';
import Chatbot from './components/Chatbot';
```

```
import Footer from './components/Footer';
import { Moon, Sun } from 'lucide-react';
function App() {
const [darkMode, setDarkMode] = useState(false);
useEffect(() => {
// Apply dark mode class to html element
if (darkMode) {
document.documentElement.classList.add('dark');
} else {
document.documentElement.classList.remove('dark');
}
}, [darkMode]);
return (
<Router>
<div className={`min-h-screen transition-colors duration-200 ${
darkMode ? 'bg-gray-900 text-white' : 'bg-gray-50 text-gray-900'
}`}>
<button
onClick={() => setDarkMode(!darkMode)}
className={`fixed top-24 right-6 z-50 p-3 rounded-full shadow-lg transition-colors ${
darkMode ? 'bg-gray-800 text-yellow-400' : 'bg-white text-gray-700'
}`}
aria-label="Toggle dark mode"
>
{darkMode ? <Sun size={20} /> : <Moon size={20} />}
</button>
```

```
<Header />
<Routes>
<Route path="/" element={<HomePage />} />
<Route path="/about" element={<AboutPage />} />
<Route path="/campus-map" element={<CampusMapPage />} />
<Route path="/contact" element={<ContactPage />} />
</Routes>
<Chatbot />
<Footer />
</div>
</Router>
);
}
export default App;
```

index.css

- Contains the **global styles** used across the project.

main.tsx

- The **entry point** of the application, where the **root component is rendered**.

vite-env.d.ts

- Provides **type definitions** for Vite, ensuring smooth TypeScript integration.

8.2.5 Project Metadata and Build Configuration

.gitignore

- Specifies **files and directories** that should be ignored by Git, such as **node_modules and environment files**.

eslint.config.js

- Defines **code linting rules** to maintain **consistent coding standards**.

index.html

- The **HTML template** used as the main entry point of the application.

LICENSE

- Contains the **legal terms** under which the project is distributed.

package-lock.json & package.json

- Define the **dependencies and project metadata**.

postcss.config.js

- Configures **PostCSS**, which processes CSS styles.

README.md

- Provides an **overview of the project**, including its features and installation steps.

tailwind.config.js

- Configuration file for **Tailwind CSS**, which is used for styling the project.

tsconfig.app.json, tsconfig.json, tsconfig.node.json

- TypeScript configuration files defining **rules and compiler options**.

vite.config.ts

- Configuration for **Vite**, the build tool used in the project.

9. FORMS AND REPORTS

9.1 Introduction

Forms and reports are essential components of the **PatriGuide** project, enabling users to interact with the system efficiently. **Forms** facilitate user input, while **reports** provide structured information about navigation usage, chatbot interactions, and system performance.

The **PatriGuide system** incorporates various forms for user engagement and reports to analyze the effectiveness of campus navigation. This section covers the types of forms and reports used in the project.

9.2 Forms in PatriGuide

Forms in **PatriGuide** serve different purposes, including **user inquiries, feedback collection, and chatbot interactions**. The key forms implemented in the project are:

9.2.1 Contact Form

- Found on the **Contact Page** (ContactPage.tsx).
- Allows users to submit inquiries regarding **campus navigation or general queries**.
- Fields included:
    - **Name** (Text Field)
    - **Email** (Email Field)
    - **Message** (Textarea for user input)
    - **Submit Button** (Sends the message to the admin or support team)
- Uses **form validation** to ensure that fields are properly filled before submission.

9.2.2 Chatbot Input Form

- Integrated into the **Chatbot Component** (Chatbot.tsx).
- Users can **type their queries** related to navigation, departments, or general campus facilities.
- Features:
    - **Text Input Field** (For user questions)
    - **Send Button** (Triggers chatbot response)
    - **Dynamic Responses** (Based on predefined chatbot logic)
- Improves **user experience** by providing **instant answers** without the need for manual navigation.

9.2.3 Feedback Form (Future Scope)

- Can be added to collect **feedback from students and visitors**.
- Fields may include:
    - **Rating System** (1 to 5 stars)
    - **Comments Section**
    - **Suggestions for Improvement**

9.3 Reports in PatriGuide

Reports in the **PatriGuide system** provide insights into how users interact with the platform. These reports help in **analyzing system performance, user engagement, and areas for improvement**.

9.3.1 Navigation Usage Report

- Tracks the most **frequently accessed locations** within the campus.
- Provides insights into:
    - Which buildings students visit the most.
    - Areas that need better accessibility improvements.

9.3.2 Chatbot Interaction Report

- Logs **user queries** and chatbot responses.
- Helps analyze:
    - Common questions asked by students and visitors.
    - Effectiveness of chatbot responses.
    - Areas where the chatbot needs improvement.

9.3.3 System Performance Report

- Analyzes **website loading speed, response time, and errors**.
- Helps ensure **smooth and efficient system operation**.

9.3.4 User Engagement Report

- Tracks **how often the PatriGuide system is used**.
- Provides insights on:
    - Total **number of users accessing the navigation system**.
    - Percentage of users using the **chatbot feature**.

- Devices used for access (**mobile, tablet, or desktop**).

9.4 Future Enhancements for Forms and Reports

To further improve the system, additional **forms and reports** can be implemented:

- **User Login Form** (For personalized navigation recommendations).
- **Automated Weekly Reports** (Sent to the admin for monitoring system usage).
- **Data Visualization Dashboard** (Graphs and charts to analyze system trends).

By continuously monitoring and improving the forms and reports, **PatriGuide** can provide a **better user experience** and ensure its **navigation system remains effective and efficient**.

10. BIBLIOGRAPHY

The bibliography section lists the references, resources, and materials that were used in the development of the **PatriGuide** project. This includes online documentation, research papers, websites, and tools that contributed to the project's implementation.

10.1 Books & Research Papers

- **User Interface Design for Software Applications** – Alan Cooper
- **Principles of Geographic Information Systems (GIS)** – Peter A. Burrough & Rachael A. McDonnell
- **Artificial Intelligence: A Guide for Thinking Humans** – Melanie Mitchell

10.2 Websites & Online Documentation

- **React Documentation**: https://react.dev/
- **TypeScript Documentation**: https://www.typescriptlang.org/docs/
- **Tailwind CSS Documentation**: https://tailwindcss.com/docs/
- **Vite.js Documentation**: https://vitejs.dev/
- **Node.js Documentation**: https://nodejs.org/en/docs
- **OpenAI Chatbot Documentation** (For AI-based chatbot implementation)

10.3 Tools & Technologies Used

- **Visual Studio Code** – Code editor for development.
- **Figma** – Used for UI/UX design prototyping.
- **GitHub** – Version control and project repository management.
- **Postman** – API testing and debugging.
- **Google Maps API** – For integrating satellite-based maps in the project.
- **Firebase (Optional for Future Scope)** – Could be used for user authentication and database management.