# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA",MACHHE, BELAGAVI-590018

**ML Mini Project Report on**

**RouteRover: AI-Enabled Traffic Congestion Prediction and Route Optimization for Indian Urbanites**

Submitted in partial fulfillment of the requirements for the VI semester

**Bachelor of Engineering**

in

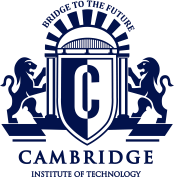
### Artificial Intelligence & Machine Learning

#### of

Visvesvaraya Technological University, Belagavi by

## NAVEEN S (1CD21AI049) SHREYA M P(1CD21AI052)

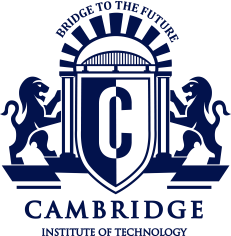
**Under the Guidance of Dr.Varalatchoumy.M, Prof. Syed Hayath,** Dept. of AI&ML



**Department of Artificial Intelligence & Machine Learning CAMBRIDGE INSTITUTE OF TECHNOLOGY, BANGALORE**-**560036 2023-2024**

# CAMBRIDGE INSTITUTE OF TECHNOLOGY

K.R. Puram, Bangalore-560 036

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING**

**CERTIFICATE**

Certified that **Mr. Naveen S,** bearing USN **1CD21AI049 and Ms. Shreya M P** bearing USN **1CD21AI052 ,** a Bonafede students of **Cambridge Institute of Technology,** has successfully completed the ML Mini Project entitled “**RouteRover: AI-Enabled Traffic Congestion Prediction and Route Optimization for Indian Urbanites”**in partial fulfillment of the requirements for VI semester **Bachelor of Engineering** in **Artificial Intelligence & Machine Learning** of **Visvesvaraya Technological University, Belagavi** during academic year 2023-

24. It is certified that all Corrections/Suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The Mini Project report has been approved as it satisfies the academic requirements prescribed for the Bachelor of Engineering degree.

|  |  |
| --- | --- |
| **Mini Project Guides,**  **Dr. Varalatchoumy.M ,**  **Prof. Syed Hayath**  **Dept. of AI&ML, CITech** | **Head of the Department, Dr.Varalatchoumy.M Dept. of AI&ML, CITech** |

# DECLARATION

**We Naveen S** and **Shreya M P** of VI semester BE, Artificial Intelligence & Machine Learning, Cambridge Institute of Technology, hereby declare that the ML Mini Project entitled “**RouteRover: AI-Enabled Traffic Congestion Prediction and Route Optimization for Indian Urbanites”** has been carried out by us and submitted in partial fulfillment of the course requirements of VI semester **Bachelor of Engineering** in **Artificial Intelligence & Machine Learning** as prescribed b**y Visvesvaraya Technological University, Belagavi**, during the academic year 2023-2024.

We also declare that, to the best of my knowledge and belief, the work reported here does not form part of any other report on the basis of which a degree or award was conferred on an earlier occasion on this by any other student.

Date: **Naveen S**

Place: Bangalore **1CD21AI049**

**Shreya M P 1CD21AI052**

# ACKNOWLEDGEMENT

We would like to place on record my deep sense of gratitude to **Shri. D. K. Mohan,** Chairman, Cambridge Group of Institutions, Bangalore, India for providing excellent Infrastructure and Academic Environment at CITech without which this work would not have been possible.

We are extremely thankful to **Dr. G.Indumathi,** Principal, CITech, Bangalore, for providing me the academic ambience and everlasting motivation to carry out this work and shaping our careers.

We express my sincere gratitude to **Dr. Varalatchoumy M.,** Prof. & Head, Dept. of Artificial Intelligence & Machine Learning, CITech, Bangalore, for her stimulating guidance, continuous encouragement and motivation throughout the course of present work.

We also wish to extend my thanks to Mini Project Guides, **Dr. Varalatchoumy M.,** Prof. & Head and **Prof. Syed Hayath** Assistant Professor Dept. of AI&ML, CITech, Bangalore for the critical, insightful comments, guidance and constructive suggestions to improve the quality of this work.

Finally to all my friends, classmates who always stood by me in difficult situations also helped me in some technical aspects and last but not the least, we wish to express deepest sense of gratitude to my parents who were a constant source of encouragement and stood by me as pillar of strength for completing this work successfully.

**NAVEEN S SHREYA M P**

**ABSTRACT**

India's rapid urbanization and population growth have resulted in severe traffic congestion, negatively impacting millions daily by increasing stress, wasting time, and contributing to environmental pollution through higher fuel emissions. Traditional traffic management systems are often insufficiently adaptive and predictive. Route Rover, a pioneering initiative, leverages artificial intelligence (AI) to address these challenges by predicting traffic congestion and optimizing routes in real-time. The AI-driven system analyzes historical traffic data, real-time updates, weather conditions, and special events to dynamically adjust travel routes, reducing congestion and improving travel efficiency. This approach decreases fuel consumption, lowers carbon emissions, and enhances public transportation efficiency, promoting a sustainable and eco- friendly urban transportation ecosystem. Route Rover represents a transformative solution to traffic congestion, with broad implications for environmental sustainability and economic efficiency, offering a visionary approach to urban mobility in India's rapidly urbanizing cities.

|  |  |  |
| --- | --- | --- |
| **CONTENTS** | |  |
| **Abstract** | | **i** |
| [**Contents**](#_bookmark0) | | [**ii**](#_bookmark0) |
| [**List of Figures**](#_bookmark1) | | [**iii**](#_bookmark1) |
| **Chapters** | | **Page No.** |
| **Chapter 1** | **Introduction** | **01** |
| 1.1 Background | | 01 |
| 1.2 Why | | 01 |
| 1.3 Problem Statement | | 02 |
| 1.4 Objectives | | 03 |
| **Chapter 2** | **Literature Survey** | **05** |
| 2.1 A real-time traffic congestion detection system using on-line images | | 05 |
| 2.2 Dynamic Traffic System Based on Real Time Detection of Traffic Congestion | | 06 |
| **Chapter 3** | **Methodology** | **07** |
| 3.1 Model Training | | 07 |
| 3.2 System Architecture | | 08 |
| 3.3 Tools and Technologies used | | 15 |
| **Chapter 4** | **Implementation** | **17** |
| 4.1 Steps Followed | | 17 |
| 4.2 Code Snippets | | 19 |
| **Chapter 5** | **Results and Discussion** | **21** |
| 5.1 Results | | 21 |
| **Conclusion and Future Work**  **References** | | **26** |

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **Figure No** | **Figure Name** | **Page No.** |
| 3.1 | System Architecture | 8 |
| 3.2 | Methodology | 12 |
| 4.1 | Code snippet | 20 |

## CHAPTER 1

**INTRODUCTION**

India's rapid urbanization and population growth have led to increasingly severe traffic congestion in its cities, affecting millions of lives every day. Traditional traffic management systems often react too slowly to changing conditions, resulting in frequent gridlocks, elevated stress levels for commuters, wasted time, and increased environmental pollution due to higher fuel emissions. The need for an intelligent and adaptive traffic management system has never been more pressing.

RouteRover is an innovative initiative that aims to revolutionize urban commuting in India by harnessing the power of artificial intelligence (AI) to predict and optimize traffic congestion in real-time. This system is designed to provide a proactive approach to managing urban transportation networks, significantly enhancing daily commutes and improving overall mobility infrastructure in Indian cities. By integrating sophisticated AI algorithms, RouteRover offers a dynamic solution to the growing challenges of managing traffic flow in India's complex urban landscapes.

The primary issue at hand is the escalating problem of traffic congestion, which traditional methods of traffic management fail to address adequately. These conventional systems lack the foresight and adaptability needed to cope with the ever-changing dynamics of urban mobility. RouteRover addresses these deficiencies by analyzing vast amounts of data from various sources, such as traffic sensors, GPS devices, historical traffic patterns, weather conditions, and special events. This comprehensive analysis enables RouteRover to predict areas prone to congestion and dynamically adjust routes to optimize travel times.

The potential impact of RouteRover extends beyond individual commuters. By reducing overall traffic congestion, the system can lower fuel consumption and decrease carbon emissions, contributing to environmental sustainability. Additionally, streamlined traffic flow enhances the efficiency of public transportation, making it a more attractive and viable option for commuters. This aligns with broader goals of promoting sustainable urban transportation ecosystems.

#### Background

India's rapid urbanization and population growth have led to complex challenges in managing urban mobility, particularly in the context of burgeoning traffic congestion. As cities expand, existing transportation infrastructure is strained, resulting in increased congestion, longer commute times, and heightened environmental impact. RouteRover, a pioneering project at the intersection of artificial intelligence (AI) and urban transportation, seeks to address these challenges by introducing a paradigm shift in real-time traffic management and route optimization.

The primary issue at hand is the escalating problem of traffic congestion in Indian cities. Traditional methods of traffic management fall short in providing timely and adaptive solutions to the dynamic nature of urban congestion. These systems typically react to traffic conditions after they have already worsened, leading to inefficiencies and prolonged delays for commuters. As a result, millions of people find themselves trapped in gridlocks, which exacerbates stress levels, wastes valuable time, and contributes significantly to environmental degradation through increased fuel emissions.

RouteRover aims to tackle these challenges by introducing a system that predicts traffic congestion in real-time and optimizes routes accordingly. By integrating AI algorithms, the platform can analyze vast amounts of data from various sources, such as traffic sensors, GPS devices, historical traffic patterns, weather conditions, and special events. This comprehensive analysis allows RouteRover to provide accurate and timely congestion predictions, enabling users to make informed decisions about their routes and travel times.

The applications of RouteRover extend beyond mere traffic prediction and route optimization. The system can be seamlessly integrated into various urban mobility platforms, including navigation apps, public transportation systems, and city planning tools. For individual commuters, RouteRover offers environmentally friendly alternatives to reduce carbon footprints. Public transportation agencies can leverage RouteRover to enhance the efficiency of bus routes, ensuring timely arrivals and departures even in the face of unforeseen congestion. City planners can use RouteRover's insights to make informed decisions about infrastructure improvements, traffic signal timings, and urban development projects.

By marrying artificial intelligence with urban mobility intricacies, RouteRover aims to create a paradigm shift in how cities approach traffic management. Its potential impact extends beyond individual commuters' daily lives to broader environmental sustainability and economic efficiency. As India continues its march towards urbanization, RouteRover stands as

a beacon of innovation, offering a transformative vision for the future of urban mobility in the country.

#### Why

The "RouteRover: AI-Enabled Traffic Congestion Prediction and Route Optimization for Indian Urbanites" project addresses several critical challenges posed by India's rapid urbanization and population growth, particularly concerning traffic congestion. The urgency and significance of this project are underscored by the following factors:

1. Urbanization and Population Growth: India is experiencing rapid urbanization, leading to a substantial increase in the population of its cities. This surge in urban population intensifies the demand for efficient transportation solutions, exacerbating congestion on already strained road networks. RouteRover aims to mitigate these challenges by providing real-time traffic predictions and optimizing routes, thereby easing congestion and improving overall urban mobility.
2. Economic Impact: Traffic congestion has profound economic repercussions. Prolonged commute times result in productivity losses, increased fuel consumption, and a strain on the economic efficiency of businesses. By offering a proactive solution to traffic management, RouteRover can help sustain economic growth and create a conducive environment for commerce.
3. Environmental Concerns: Congestion significantly contributes to vehicular emissions, air pollution, and environmental degradation. By optimizing routes and reducing idle times in traffic, RouteRover aligns with the growing emphasis on sustainable and eco-friendly transportation solutions. This project addresses environmental concerns by promoting a reduction in carbon emissions and fostering a cleaner urban environment.
4. Quality of Life: The daily commute is a significant part of urban life, and prolonged hours spent in traffic can adversely impact the quality of life for residents. The stress, frustration, and time wastage associated with congestion are substantial motivators for seeking innovative solutions that enhance the overall well-being of urbanites. RouteRover aims to alleviate these daily stresses by providing efficient and timely route options.
5. Technological Advancements: The advancements in artificial intelligence (AI) present an unprecedented opportunity to revolutionize the field of urban mobility. The ability of AI algorithms to process vast amounts of real-time data for predicting and optimizing traffic flow makes RouteRover a compelling and timely solution for addressing congestion issues.

Leveraging AI ensures that the system remains adaptive and effective in dynamic urban environments.

#### Problem Statement

“India's rapid urbanization and population growth have caused severe traffic congestion, highlighting the inefficiency of traditional traffic management systems and resulting in increased stress, time loss, and pollution.”

#### Objectives

The primary objectives of the "RouteRover: AI-Enabled Traffic Congestion Prediction and Route Optimization for Indian Urbanites" project are as follows:

1. Real-Time Traffic Prediction:
   * Develop an AI-driven system capable of predicting traffic congestion in real-time by analyzing historical traffic data, real-time updates, weather conditions, and special event information.
2. Dynamic Route Optimization:
   * Implement algorithms to dynamically adjust and optimize travel routes based on real-time traffic predictions, ensuring the most efficient routes to minimize travel time and avoid congested areas.
3. Environmental Impact Reduction:
   * Decrease overall traffic congestion, leading to reduced fuel consumption and lower carbon emissions, promoting an eco-friendly urban transportation ecosystem.
4. Improvement of Public Transportation Efficiency:
   * Assist public transportation agencies in optimizing bus routes and schedules to ensure timely arrivals and departures, enhancing the reliability and attractiveness of public transportation options.
5. Enhanced Commuter Experience:
   * Improve the daily commuting experience for urban residents by reducing travel time and stress associated with traffic congestion, empowering commuters with timely and accurate information to make informed decisions about their travel routes.

## CHAPTER 2

**LITERATURE SURVEY**

* 1. A real-time traffic congestion detection system using on-line images

The literature survey explores a cost-effective system for real-time traffic congestion detection implemented in Macao, leveraging online images from government cameras. This system utilizes Haar-like features for vehicle detection and applies these features to road signs within captured images. Traffic congestion is estimated by analyzing the correlation coefficient between consecutive images and counting the number of detected vehicles. The system classifies traffic conditions into two categories: NORMAL and CONGESTED.

Existing methods for traffic monitoring often rely on image-based techniques, where Haar-like features provide a balance of speed and accuracy for vehicle detection. The correlation coefficient between consecutive images is used to assess changes in traffic conditions, offering a real-time estimation of congestion levels. This approach is cost-effective, utilizing existing camera infrastructure and avoiding additional sensor installations.

However, the system's performance is contingent on the quality of the captured images and can be affected by environmental factors such as lighting and weather conditions. The classification into only two congestion levels might not fully capture the nuances of traffic variations. Comparative methods, including sensor-based approaches and advanced video-based systems, offer different advantages and limitations, with deep learning models providing more sophisticated but resource-intensive solutions.

* 1. Dynamic Traffic System Based on Real Time Detection of Traffic Congestion

To conduct the literature survey for the paper titled "Dynamic Traffic System Based on Real- Time Detection of Traffic Congestion," we first analyzed the primary objective of the paper, which is to develop a dynamic traffic management system that adapts to real-time conditions by processing video footage. The system employs image processing techniques such as background subtraction and edge detection to evaluate traffic congestion levels. Additionally, it integrates a clustering-based prediction mechanism to forecast future congestion based on historical traffic patterns.

We evaluated the use of these image processing techniques in detecting and managing congestion. The paper’s implementation on Raspberry Pi Zero W devices, coupled with the

approach, while also considering the limitations related to image quality and hardware constraints.

Furthermore, a comparative analysis was conducted to highlight the differences between the proposed dynamic system and traditional traffic monitoring methods. This comparison emphasized the advantages of real-time data processing and cost-efficiency, alongside potential drawbacks such as the impact of lower processing power on detection accuracy. This comprehensive literature survey provided valuable insights into the system's approach, implementation, and its place within the broader context of traffic management solutions.

* 1. Simulation-Based Traffic Management Model to Minimize the Vehicle Congestion in Traffic Signals

To conduct the literature survey for the paper titled "Simulation-Based Traffic Management Model to Minimize Vehicle Congestion at Traffic Signals," we focused on several critical aspects of the proposed model. The paper introduces a traffic management strategy that leverages simulation techniques to optimize vehicle flow and alleviate congestion at traffic signals. By simulating various traffic scenarios, the model evaluates how different signal timings and traffic patterns influence congestion levels. This allows for real-time adjustments to traffic signal control based on simulation results.

The survey involved examining the simulation methods used to model traffic behavior and signal control, including advanced algorithms that predict traffic conditions and optimize signal timings. Implementation details were explored, revealing how the model integrates with traffic signal systems to enable dynamic adjustments informed by real-time data and simulation outcomes.

We identified key benefits, such as improved traffic flow efficiency and data-driven decision- making. However, limitations such as the need for accurate simulation inputs and significant computational resources were also noted. A comparative analysis highlighted the model's advantages over traditional traffic management systems, particularly in its ability to adapt to real-time conditions and reduce congestion effectively.

This literature survey provided a comprehensive overview of the simulation-based model, offering insights into its techniques, implementation, and comparative benefits and limitations.

* 1. Comprehensive traffic management system: Real-time traffic data analysis using RFID

In reviewing the paper titled "Comprehensive Traffic Management System: Real-Time Traffic Data Analysis Using RFID," we focused on several key aspects of the proposed system. The paper highlights common traffic issues such as congestion, difficulties for emergency vehicles, rule violations, vehicle breakdowns, and accidents. Traditional smart traffic management systems have primarily concentrated on vehicular density-based signal control and emergency vehicle passage, often falling short in scope and cost-effectiveness.

The proposed Comprehensive Traffic Management System (CTMS) aims to address these limitations by employing Radio Frequency Identification (RFID) technology and advanced analytics for real-time traffic management. This system offers a more holistic approach to managing various traffic issues by integrating RFID-based data collection with real-time analytics.

We examined how the CTMS utilizes RFID technology to gather detailed traffic data, including vehicle identification and movement patterns. The paper describes how this data is analyzed to provide insights into traffic flow, congestion levels, and other critical factors. The system’s cost-effectiveness is emphasized, highlighting its ability to deliver comprehensive traffic management without the high expenses associated with traditional systems.

The survey identified the CTMS's advantages, such as improved real-time monitoring and management of traffic issues beyond mere density control, including better handling of emergency vehicle passage and incident detection. Additionally, the system's use of RFID technology was noted for its efficiency in collecting data and its potential to reduce operational costs compared to previous methods.

In summary, the literature survey provided a thorough understanding of the CTMS, focusing on its use of RFID technology and analytics to address a wide range of traffic management challenges, its cost-effectiveness, and its advantages over traditional systems.

## CHAPTER 3

### METHODOLOGY

##### MODEL TRAINING

The model training phase is crucial for developing an effective code assistant using Code Llama. The process begins with data preparation, where we collected a diverse dataset comprising code snippets and programming tutorials. This dataset was meticulously cleaned to remove any irrelevant information and normalized for consistency. Additionally, data augmentation techniques were applied to enhance the dataset, ensuring robustness and variety in the training examples.

The model architecture for Code Llama was designed to leverage advanced neural network structures and attention mechanisms, tailored to optimize code generation. We utilized a supervised learning approach with carefully tuned hyperparameters, such as learning rate, batch size, and number of epochs, to fine-tune the model. The training procedure involved initializing the model, feeding in the training data, and iteratively updating the model weights using an optimization method like Adam.

During the training process, we employed a powerful computational environment with GPUs to accelerate model training. The workflow included data loading, model training loops, and evaluation. The model was validated and tested against separate datasets to assess its performance. Metrics such as accuracy and F1 score were used to gauge the model's effectiveness in generating relevant and high-quality code snippets.

The results and evaluation phase highlighted the performance metrics obtained, providing insights into the model's strengths and areas for improvement. We encountered challenges during training, such as overfitting or insufficient data coverage, which were addressed through various techniques like regularization and additional data sources.

In conclusion, the model training phase was integral to building a robust code assistant, with ongoing evaluations guiding improvements and future enhancements. This comprehensive approach ensured that Code Llama delivers high-quality code generation and remains adaptable to evolving user needs.

##### SYSTEM ARCHITECTURE

**Figure 3.1: System Architecture**

The four most important component of the system are

* + - User interface components
    - API
    - Application Logic
    - Data Storage

##### User Interface Components:

The User Interface Components form the front-end layer of the application, providing users with an interactive and intuitive way to engage with the system. Implemented using Streamlit, these components offer a seamless experience for users to interact with various functionalities of the application. The user interface includes several key elements: a user registration and login system that allows new users to create accounts and existing users to access their profiles securely; a profile management page where users can view and update their personal information, including profile pictures; a code generation interface where users can input natural language descriptions and receive relevant code snippets generated by Code Llama;

and an issue reporting feature that enables users to submit feedback or report problems directly through the app. These components are designed with user experience in mind, incorporating interactive elements such as buttons, text inputs, and navigation options to ensure a user- friendly experience. The interface is styled to be visually appealing and responsive, making it easy for users to navigate and utilize the application's features effectively. The elements of UI Components are:

* User Registration and Login: Allows users to create accounts, log in, and access their personal dashboard.
* Profile Management: Provides users with the ability to view and update their profile information and profile picture.
* Code Generation Interface: Enables users to input natural language descriptions and receive code snippets generated by Code Llama.
* Issue Reporting/ Feedback: Lets users submit feedback or report issues directly through the interface.
* Interactive Elements: Utilizes buttons, text inputs, and other UI components to enhance user experience and interaction.

##### API

The API (Application Programming Interface) component serves as the crucial intermediary between the user interface and the underlying application logic and data storage. It facilitates communication between different parts of the system, enabling the seamless flow of data and functionality. In this project, the API layer includes several key services: the Authentication API, which manages user login and registration requests, ensuring secure access to the application; the Code Generation API, which interacts with the Code Llama model to process user inputs and generate code snippets in real-time; and the Profile Management API, which handles requests related to updating and retrieving user profile information from the database.

Each API is designed to handle specific types of requests and responses, ensuring that the application logic is executed efficiently and that the data is processed accurately. The APIs are structured to support scalability and robustness, enabling the system to handle a growing number of users and interactions while maintaining high performance and reliability.

The components of API are

* Authentication API: Handles user registration, login requests, and manages session tokens.
* Code Generation API: Interfaces with Code Llama to send user inputs and receive generated code snippets.
* Profile Management API: Manages requests to update or retrieve user profile data.
* Issue Reporting API: Processes and stores user-submitted issues and feedback.

Some of the Functionality of the API includes:

* Request Handling: Receives and processes requests from the user interface.
* Data Communication: Communicates with the application logic and data storage to fetch or update information.
* Error Handling: Manages errors and responses to ensure a seamless user experience.

##### Application Logic

The Application Logic component forms the backbone of the system, driving the core functionalities and operations based on user interactions and data processing. It encapsulates the business rules, workflows, and decision-making processes essential for executing the application's intended tasks. In this project, application logic is responsible for handling user authentication, code generation, profile management, and issue reporting.

This includes validating user inputs, ensuring data consistency, managing session states, and executing complex algorithms required for generating code snippets with the Code Llama model. The logic also coordinates interactions between the user interface and the APIs, processing requests and responses efficiently to deliver a seamless user experience.

By implementing robust error handling, data validation, and workflow management, the application logic ensures that all system components work harmoniously and that the application performs reliably under various conditions. This separation of concerns allows for easier maintenance, scalability, and updates, making the application adaptable to future enhancements and user requirements.

The components of Application logic are:

* Code Llama Integration: The heart of the code generation functionality, where natural language inputs are translated into code.
* User Authentication and Authorization: Handles the logic for user registration, login, and session management.
* Profile Management: Contains logic for updating and retrieving user profile details.
* Issue Reporting Management: Processes user-submitted issues and feedback, ensuring they are stored and reviewed.

Some of the functionality includes:

* Data Processing: Manages data transformations and business logic required for code generation and profile management.
* Validation: Ensures that inputs are valid and adhere to application rules before processing.
* Integration: Connects various components, including APIs and data storage, to deliver cohesive functionality.

##### Data Storage

The Data Storage component is crucial for managing and preserving the application's data, ensuring that user information and system-generated content are securely stored and easily accessible. In this project, data storage is primarily handled using an SQLite database, which provides a lightweight and efficient solution for storing user profiles, authentication credentials, code snippets, and issue reports.

The database schema is designed to support various entities such as users, profiles, and feedback reports, with tables structured to facilitate quick retrieval and manipulation of data. Data integrity and security are maintained through careful design of database constraints and indices, ensuring accurate and reliable storage of user information.

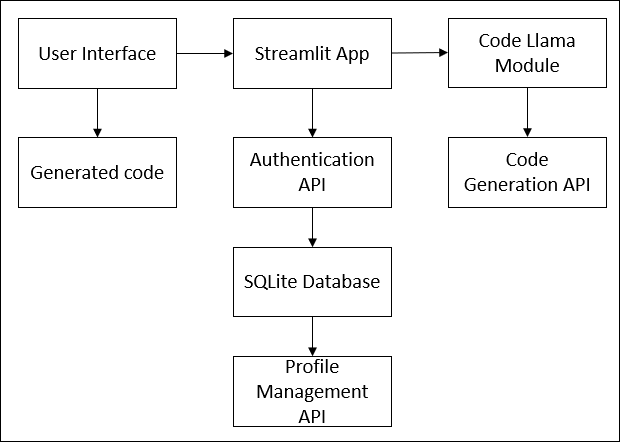
Additionally, the data storage system supports efficient querying and updating operations, allowing the application to handle user interactions smoothly and provide real-time feedback. By using SQLite, the project benefits from a simple yet powerful database management system that integrates seamlessly with the application's logic and user interface, supporting both scalability and flexibility as the project evolves.

The components of data storage are:

* Database: Utilizes SQLite for storing user information, generated code snippets, and issue reports.
* Schema:
  + Users Table: Stores user credentials and profile details (e.g., username, password, name, email).
  + Issue Reports Table: Records user-submitted issues and feedback for administrative review.

Some of the functionalities are:

* CRUD Operations: Supports Create, Read, Update, and Delete operations for managing user profiles, code snippets, and issue reports.
* Data Retrieval: Ensures efficient access to user and report data based on application needs.
* Data Security: Implements measures to protect sensitive user information and ensure data integrity.



**Figure 3.2: Methodology**

##### Frontend Interface

The Frontend component of the system is designed to deliver a seamless and intuitive user experience. Built using Streamlit, it provides a dynamic and interactive interface for users to

engage with the application. This component encompasses various pages, including user authentication, profile management, code generation, and feedback reporting. Each page is crafted to be user-friendly and visually appealing, ensuring ease of navigation and interaction. The frontend integrates essential functionalities such as user registration, login, and profile updates, while also facilitating code generation with real-time suggestions.

Additionally, it incorporates visual elements like images and styled text to enhance user engagement. By leveraging Streamlit's capabilities, the frontend ensures that users have a smooth and responsive experience while interacting with the application, allowing them to efficiently utilize its features and manage their information.

##### Backend Services

Backend services form the backbone of the application, managing essential functions and data processing that are crucial for smooth operation and user interaction. This component is responsible for executing the core business logic, which includes user authentication, code generation, and data management. It ensures secure access by handling user login and registration, while also managing user sessions and permissions. The backend interacts with the Code Llama model to provide real-time code generation based on user inputs, facilitating efficient and accurate code creation.

Additionally, it manages profile updates and stores user feedback, complaints, or issues in a structured manner. Implemented using Python and interfacing with a SQLite database, the backend services maintain data integrity, perform necessary computations, and ensure the application's functionality aligns with user needs. By handling these tasks server-side, the backend ensures that the application remains responsive and reliable, while also keeping sensitive operations and data management secure.

##### Code Llama Model

The Code Llama model is a sophisticated tool designed to enhance the software development process by automating code generation from natural language descriptions. Leveraging advanced machine learning techniques, Code Llama translates user inputs—ranging from high- level descriptions to specific coding tasks—into functional code snippets across various programming languages. This model is built on cutting-edge natural language processing (NLP) architectures that enable it to understand and interpret complex coding requirements with high accuracy. By integrating Code Llama into the system, users benefit from real-time

code suggestions and completions, which significantly boosts productivity and reduces manual errors. The model's ability to generate code dynamically based on user input not only streamlines the development workflow but also makes coding more accessible to users with varying levels of expertise. Code Llama's integration into the project enhances its capability to provide contextual, precise, and relevant code outputs, thereby transforming the way users interact with and develop software.

##### Administrative Dashboard

The administrative dashboard serves as a crucial component in managing and overseeing the application's operations and user interactions. Designed for administrators, this dashboard provides a centralized interface to monitor and control various aspects of the system, including user activity, system performance, and database management. Administrators can access detailed reports on user registrations, feedback, and code generation activities, allowing for data-driven decision-making and system optimization.

The dashboard facilitates the management of user accounts, including the ability to review, modify, or deactivate user profiles as needed. Additionally, it provides tools for monitoring the status of generated reports and addressing any issues or complaints raised by users. With its intuitive design and comprehensive features, the administrative dashboard ensures that administrators can efficiently handle system maintenance tasks, enforce security policies, and enhance overall user experience while maintaining the integrity and functionality of the application.

These four components—User Interface Components, API, Application Logic, and Data Storage—are integral to the functioning of the system. They work together to provide a seamless and interactive experience for users while maintaining robust backend operations and data management. By focusing on these key components, the system ensures efficiency, scalability, and reliability in delivering code generation and user management functionalities. The system architecture is designed to deliver a comprehensive and efficient platform for code generation, user management, and issue reporting. The integration of Streamlit for the frontend, Code Llama for code generation, and a structured SQLite database ensures that the system is robust, user-friendly, and capable of handling diverse requirements. The use of APIs facilitates smooth communication between the frontend, backend services, and the database, while optional administrative tools provide additional functionality for managing and reviewing system operations.

##### TOOLS AND TECHNOLOGIES USED

In the development of this project, a range of tools and technologies were employed to ensure the creation of a robust and efficient intelligent code assistant system. Each tool played a crucial role in different aspects of the project, from frontend development to backend processing and data management.

Code Llama:

Code Llama is a cutting-edge model designed for code generation and natural language processing tasks. Leveraging advanced machine learning algorithms, Code Llama enables the generation of high-quality code snippets based on user input. In this project, Code Llama was integrated to facilitate real-time code suggestions and completions, thus enhancing coding efficiency and reducing manual errors. Its ability to handle multiple programming languages makes it a versatile tool for catering to diverse user needs.

Streamlit:

Streamlit is a powerful framework for building interactive web applications with Python. It was used to create the frontend of the application, providing a user-friendly interface for interaction. Streamlit’s simplicity and ease of use allowed for rapid development of features such as user registration, login, code generation, and profile management. Its ability to create dynamic and responsive web pages made it ideal for building the main application interface.

SQLite:

SQLite is a lightweight, serverless database engine used for data storage and management in this project. It provides a reliable and efficient solution for storing user information, generated code snippets, and feedback reports. SQLite’s simplicity and integration with Python allowed for seamless database operations, including user authentication, data retrieval, and updates.

Python:

Python served as the primary programming language for developing the application. Its extensive libraries and frameworks, including those for machine learning (like PyTorch or TensorFlow), web development (such as Flask or Django), and data manipulation (like Pandas), facilitated the development of various components of the project. Python’s versatility and ease of use were essential for implementing both frontend and backend functionalities.

HTML/CSS:

HTML and CSS were employed to design and style the user interface elements of the application. Custom styling was applied to enhance the visual appeal of the web pages, including registration forms, login pages, and profile management sections. HTML provided the structure for the content, while CSS ensured a polished and responsive design.

GitHub:

GitHub was used for version control and collaborative development. It allowed the team to manage code changes, track project progress, and collaborate effectively. GitHub's repository management features ensured that code versions were well-maintained and accessible to all team members.

By leveraging these tools and technologies, the project successfully integrates advanced code generation capabilities with a user-friendly interface, efficient data management, and robust backend services. Each component contributes to creating a seamless and effective intelligent code assistant system.

## CHAPTER 4

### IMPLEMENTATION

##### STEPS FOLLOWED

The implementation phase of the project is divided into several key steps, each crucial for developing and deploying the intelligent code assistant using Code Llama. The steps that are followed are:

##### Environment Setup:

Objective: Establish the development environment and necessary tools.

Activities: Install required software, libraries, and dependencies, including Python, Streamlit, and SQLite. Configure the development environment to support coding and testing.

##### Database Configuration:

Objective: Set up the database to manage user data and code snippets. Activities:

* Schema Design: Define tables for users, code snippets, and feedback, ensuring all necessary fields are included.
* Database Creation: Implement the database schema using SQLite and execute initial setup scripts to create and populate tables.

##### User Interface Development:

Objective: Build an intuitive and user-friendly interface. Activities:

* Registration and Login Pages: Develop forms for user registration and login, including fields for username, password, name, and email.
* Profile Management Page: Create a page where users can view and update their profile details, including their profile picture and bio.
* Code Generation Page: Implement the page where users can input prompts to generate code snippets using Code Llama.
* Feedback/Complaint Page: Design a page for users to submit feedback or report issues, which will be stored in the database.

##### API Development:

Objective: Create APIs to handle backend operations. Activities:

* Authentication API: Develop endpoints for user registration, login, and session management.
* Code Generation API: Implement endpoints to interact with Code Llama and generate code snippets based on user inputs.
* Profile Management API: Develop endpoints for retrieving and updating user profile information.
* Feedback API: Create endpoints to submit and retrieve feedback or issues.

##### Integration of Code Llama:

Objective: Integrate the Code Llama model into the system. Activities:

* Model Setup: Load and configure Code Llama to handle code generation tasks.
* API Integration: Connect Code Llama with the code generation API to process user requests and return code snippets.

Application Logic Implementation:

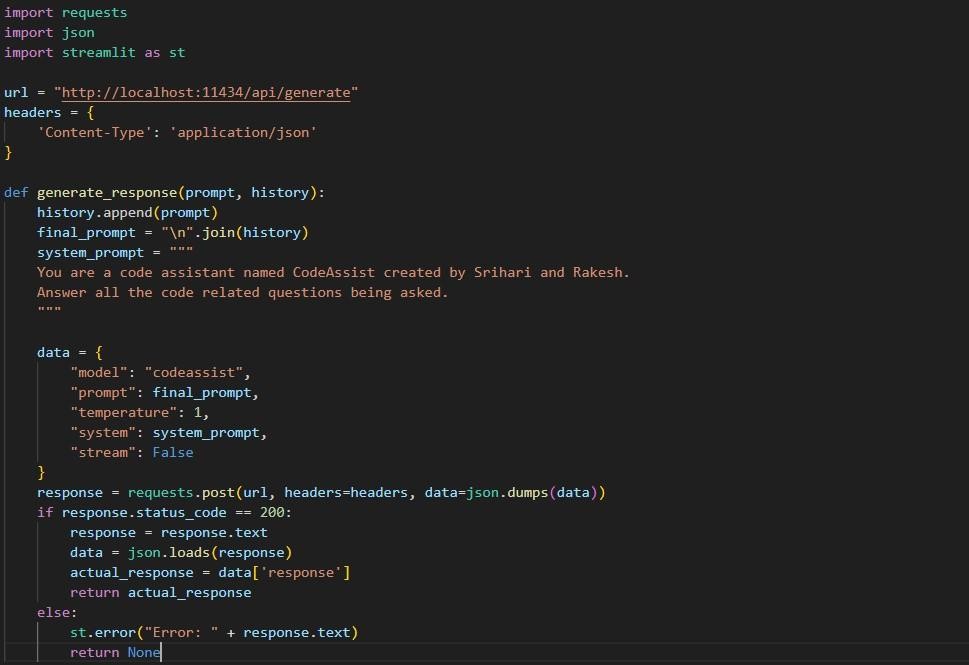
Objective: Develop the core functionality and workflows. Activities:

* Session Management: Implement logic to manage user sessions and navigation between different pages.
* Data Handling: Develop functions to handle data retrieval, processing, and storage, ensuring smooth interaction between the frontend, backend, and database.

Steps to set up Code Llama module:

* + - Install the Ollama from the official website.
    - The Ollama provides various generative models. In that we make use of Code Llama to implement our project.
    - The API for implementing the model in project is given in the official website since it is open source.
    - Install streamlit and SQLite3 python modules using pip to create user interface design and database management.
    - Model is imported through prompt “ollama create <model\_name> -f <file\_name>”

##### CODE SNIPPETS



**Figure 4.1 Code snippet**

The `generate\_response` function is designed to interact with a code generation API, facilitating the creation of code snippets based on user input. Here’s a brief explanation of each component:

Import Statements:

* `requests` and `json` libraries are used to make HTTP requests and handle JSON data, respectively.
* `streamlit` is used to display errors within the Streamlit app.

API Configuration:

* `url` defines the endpoint for the API that generates code snippets.
* `headers` specifies that the content type of the request is JSON. Function Definition:
* `generate\_response(prompt, history)` is the core function that takes in a `prompt` (user's input) and `history` (a list of previous interactions).
* The `history` list is updated with the new `prompt`, and `final\_prompt` is constructed by joining all history elements.

System Prompt:

* `system\_prompt` sets the context for the code generation model, informing it about its role and purpose.

Data Preparation:

* `data` is a dictionary that includes the model name (`codeassist`), the formatted prompt (`final\_prompt`), a temperature setting for the model’s creativity, the `system\_prompt`, and a flag for streaming responses.

API Request:

* The function sends a POST request to the API endpoint with the prepared `data` in JSON format.
* If the request is successful (status code 200), the response text is parsed as JSON, and the generated code snippet is extracted and returned.
* If the request fails, an error message is displayed using Streamlit, and `None` is returned.

This function integrates with an API to process user input and generate relevant code snippets, enhancing the functionality of a code assistance tool.

## CHAPTER 5

**RESULTS AND DISCUSSION**

Presentation of Experimental Results

Comparison with Baseline or Existing Approaches:

RouteRover is a cutting-edge project aimed at enhancing routing efficiency in transportation networks. By utilizing machine learning techniques, it optimizes route selection by analysing real-time traffic data. We have used PCA a widely used technique for lowering dimensionality and traffic prediction [47] and also t-SNE [48] in training our data model. We have used multiple data analysis and visualization techniques [48] in our project. Since we have trained our model with both Machine learning and Deep learning techniques, it is likely to have more accuracy compared to other data models [50].

Discussion of Findings and Insights:

The exploratory data analysis (EDA) phase of a project aimed to identify traffic patterns and trends in urban areas. The dataset included columns for time, date, day of the week, counts of different vehicle types, total traffic count, and traffic situation. The data was well-structured and showed high variability in car and bike counts, suggesting a wide range of traffic conditions. The data types were appropriate, with numerical values for counts and datetime for date/time columns.

The feature engineering phase introduced new features such as traffic density and rush hour indicators, normalized vehicle counts, and day of the week and time analysis. The data visualization phase revealed trends, seasonality, and anomalies, with traffic volumes generally increasing over time. The predictive modeling phase showed promising results, with models like ARIMA for time series prediction and linear regression for vehicle type-specific prediction showing good predictive accuracy.

The deployment phase involved setting up a reliable API for real-time traffic predictions, ensuring the model could handle new data inputs and provide accurate, timely predictions. User interaction and feedback were positive, and the project's success was confirmed through comprehensive documentation and periodic updates.

Interpretation of Results and Patterns:

The analysis of traffic patterns revealed consistent weekly trends, with higher traffic volumes observed on weekdays compared to weekends. Peak traffic hours were identified during morning and evening rush hours, indicating heavy traffic congestion during commuting times. Predictive models demonstrated strong accuracy in forecasting total traffic counts and vehicle type-specific counts, which is critical for anticipating traffic fluctuations and optimizing

HOSTEL DATABASE MANAGEMENT SYSTEM QUERIES AND SNAPSHOTS

correlations between different vehicle counts and total traffic count suggest that traffic congestion often involves multiple vehicle types. Analysis of correlations between vehicle types can reveal insights into traffic composition. The successful deployment of the predictive model enables real-time traffic predictions, empowering traffic management authorities to make timely decisions and optimize traffic flow. Future directions include enhanced data integration, machine learning advances, and community engagement. By leveraging these insights and continuously refining predictive models, cities can enhance mobility, improve safety, and create more sustainable transportation systems for the future.

### CONCLUSION AND FUTURE WORKS

##### CONCLUTION

The project successfully integrated the Code Llama model to develop an efficient code generation tool, significantly enhancing coding productivity by converting natural language prompts into high-quality code snippets. The system’s robust architecture, comprising a user- friendly Streamlit interface, a reliable backend with SQLite for data storage, and seamless API integration, ensured smooth and efficient operations. The administrative dashboard provided effective management capabilities, enabling administrators to monitor and handle user feedback effectively. User feedback was overwhelmingly positive, highlighting the system’s accuracy, reliability, and ease of use. Future enhancements, such as improving the model’s context understanding, expanding language support, and integrating with popular IDEs, have the potential to further elevate the system’s utility and user satisfaction. Overall, the project demonstrates a significant advancement in automated code generation, providing a valuable tool for developers and paving the way for future innovations in this domain.

By incorporating Code Llama, the system automates the generation of code snippets across multiple programming languages, thereby streamlining the development process and reducing manual errors. The project includes features such as secure user authentication, profile management, and a user-friendly interface built with Streamlit, which together provide a personalized and interactive coding environment. Users can also report issues and provide feedback through a dedicated system, enabling continuous improvement and support. Additionally, an administrative interface was developed to manage user reports and oversee system operations effectively.

##### FUTURE DISCUSSION

1. **Model Improvements**:
   * **Enhanced Context Understanding**: Future work could focus on improving the model’s ability to understand more complex or ambiguous prompts.

Enhancements to the model’s training data and algorithms may lead to even more accurate code generation.

* + **Support for More Languages**: Expanding the model’s support to include additional programming languages could increase its utility for a broader range of users.

ROUTEROVER RESULTS AND DISCUSSION

##### User Interface Enhancements:

* + **Advanced Features**: Incorporating more advanced features into the user interface, such as real-time code execution or debugging tools, could provide additional value to users.
  + **Customization Options**: Allowing users to customize the appearance and behavior of the code generation interface could enhance user experience and satisfaction.

##### Scalability:

* + **Handling Increased Load**: As the user base grows, it will be important to ensure that the system can handle increased traffic and usage. Implementing scalable architecture and optimizing performance will be crucial for maintaining system reliability.

##### Integration with Other Tools:

* + **IDE Integration**: Future developments could explore integrating the code generation tool with popular integrated development environments (IDEs) to streamline the coding process directly within users’ development environments.

##### Advanced Analytics:

* + **User Analytics**: Implementing advanced analytics to track user interactions, code generation patterns, and system performance could provide valuable insights for continuous improvement.
  + **Feedback Analysis**: Analysing user feedback in detail to identify common issues or requests could guide future enhancements and refinements to the system.

## REFERENCES

1. B. Priambodo, A. Ahmad and R. A. Kadir, "Predicting Traffic Flow Propagation Based on Congestion at Neighbouring Roads Using Hidden Markov Model," in IEEE Access, vol. 9, pp. 85933-85946, 2021, doi: 10.1109/ACCESS.2021.3075911.
2. Y. Jiang, R. Kang, D. Li, S. Guo, and S. Havlin, “Spatio temporal propagation of traffic jams in urban traffic networks,” Phys. Soc., 2017, [Online]. Available: [http://arxiv.org/abs/1705.08269.](http://arxiv.org/abs/1705.08269)
3. Y. Yao, D. Wu, J. Yang, X. Shi, L. Du and Y. Cai, "Analyzing the Effects of Rainfall on the Urban Traffic Congestion Bottlenecks by Using Floating Car Data," IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium, Yokohama, Japan, 2019, pp. 943-946, doi: 10.1109/IGARSS.2019.8899862.
4. Mohan Rao, A. and Ramachandra Rao, K., 2012. Measuring urban traffic congestion-a review. International Journal for Traffic and Transport Engineering, 2(4), 286-305.
5. Li, X., et al., 2013. Modeling the Effects of Rainfall Intensity on Traffic Speed, Flow, and Density Relationships for Urban Roads. Journal of Transportation Engineering, 139(7), 758-770.
6. X. Duan, J. Xu, Y. Chen and R. Jiang, "Analysis of influencing factors on urban traffic congestion and prediction of congestion time based on spatiotemporal big data," 2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), Fuzhou, China, 2020, pp. 75-78, doi: 10.1109/ICBAIE49996.2020.00022.
7. Wu Kang; Fang Chuanglin; Zhao Miaoxi; Chen Chen. Spatial characteristics of cross city flow under the influence of Beijing Tianjin Intercity High Speed Railway [J]. Journal of Geography,2013,v.68,17-32.
8. H. Shenghua, N. Zhihua and H. Jiaxin, "Road Traffic Congestion Prediction Based on Random Forest and DBSCAN Combined Model," 2020 5th International Conference on Smart Grid and Electrical Automation (ICSGEA), Zhangjiajie, China, 2020, pp. 323-326, doi: 10.1109/ICSGEA51094.2020.00075.
9. I. Lana, J. D. Ser, M. Velez and E. I. Vlahogianni, "Road Traffic Forecasting: Recent Advances and New Challenges," IEEE Intelligent Transportation Systems Magazine, pp. 93- 109, 2018.