TRAIN TRACKING AND DETECTION SYSTEM

TMP-23-302

Project Proposal Report

Biyanwila B.D.V.J.

B.Sc. (Hons) Degree in Information Technology
(Specializing in Data Science)

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

2023

TRAIN TRACKING AND DETECTION SYSTEM

TMP-23-302

Project Proposal Report

B.Sc. (Hons) Degree in Information Technology
(Specializing in Data Science)

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

2023

DECLARATION

I Biyanwila B.D.V.J., declare that the research project report is my original work, and it has not been submitted in whole or in part for any other degree or qualification. Any ideas, data, or information obtained from other sources have been fully acknowledged by means of a citation or reference. I have followed the guidelines for academic writing and referencing provided by my university, and the project conforms to the required standard.

Name	Student ID	Signature
Biyanwila B.D.V.J.	IT20212490	[illegible]

The above candidates are carrying out research for the undergraduate Dissertation under my supervision.

Signature of the supervisor: Date: 2023/05/01

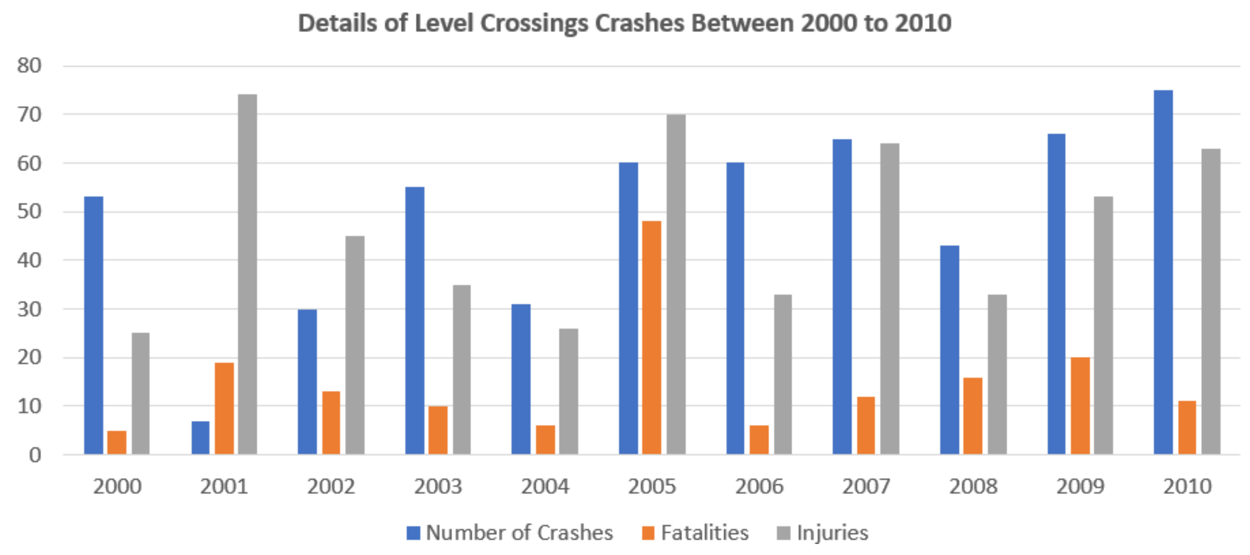
ABSTRACT

Railway Crossing accidents have been a significant safety concern not only worldwide, but also in Sri Lanka, resulting in significant loss of life and property annually. The biggest railway crossing collision that happen in Sri Lanka was on 27th April 2005, near Polgahawela, resulted the death of 41 citizens.

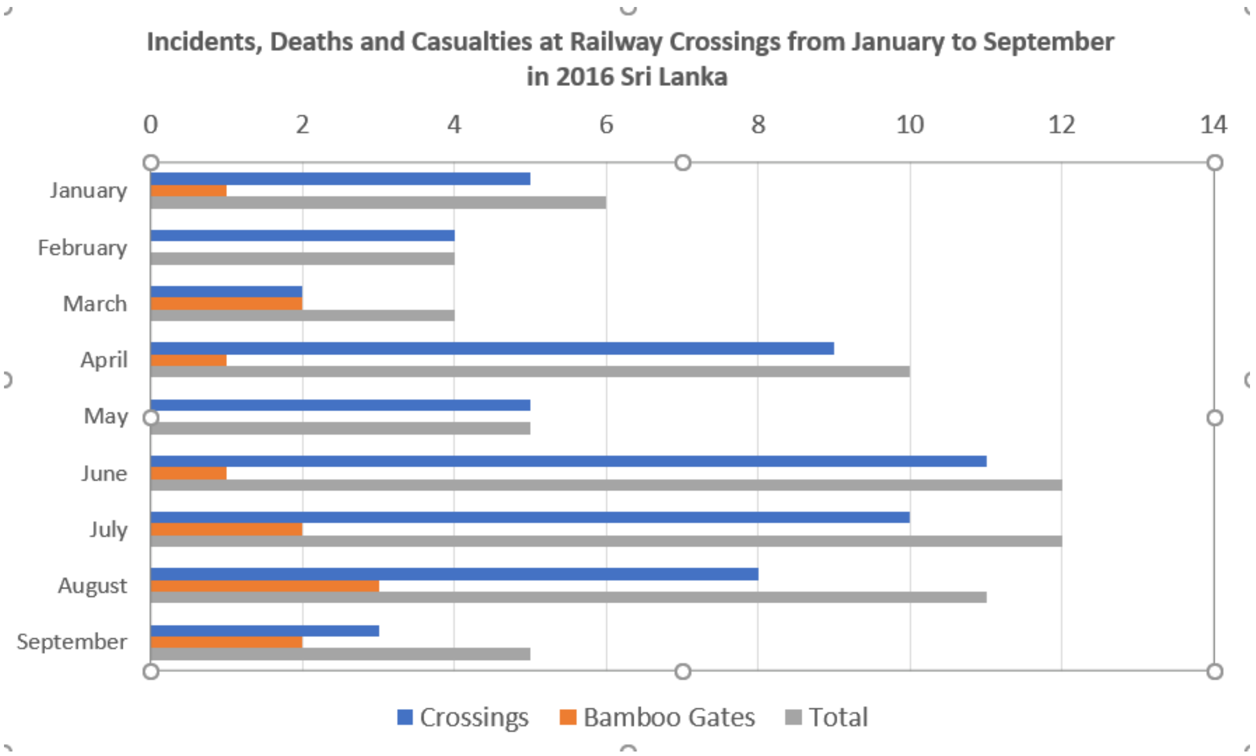
[1]

Polgahawela level crossing accident was a collision between a bus travelling from Galkiriyagama to Colombo and a train at a level crossing in Yangalmodara, near Polgahawela in Kurunegala district on 27 April 2005 at 8.30 local time, which resulted in the death of 41 people [2]

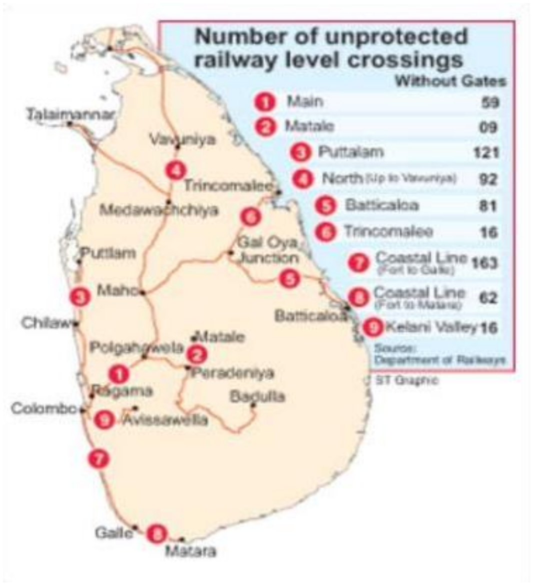
These are some detailed facts that proves how much damage is done annually from railway collision in Sri Lanka.



[3]



[3]



Number of unprotected railway level crossings

	Without Gates
1 Main	59
2 Matale	09
3 Puttalam	121
4 North (Up to Vavuniya)	92
5 Batticaloa	81
6 Trincomalee	16
7 Coastal Line (Fort to Galle)	163
8 Coastal Line (Fort to Matara)	62
9 Kelani Valley	16

Source: Department of Railways

[4]

And also statistics proves that, 1 person dies due to a road accident in every 3 hours averagely in Sri Lanka.[5] and Every 3 days, a Child dies in Road Accident. [5]

We identified some major reasons for these collisions. Major reason for so much collisions in Sri Lanka is there are so much unsafety unprotected railway crossings all around the Island. Above image [4] shows how many unprotected railway crossings in all around the Island. Due to some heavy rains or fog conditions, people cannot see the upcoming railway crossings when they are driving the vehicle. Human Errors is also another fact for railway collisions in Sri Lanka. That means citizens may misjudge the speed or distance of an approaching train, leading to accidents or near-misses. And also, Citizens may not be aware of the dangers posed by railway crossings or the proper safety procedures to follow when crossing tracks. Therefore, providing solutions to reduce these collisions happening in railway crossings became a major need specially in Sri Lanka because many human and animal lives and properties lost to the country annually. But from the government side, we cannot see they are considering this as a big problem. They didn't even take necessary steps to reduce these railway crossing collisions like providing gates near railway crossings, signal and alarms likewise. As citizens who feel this as a major problem in Sri Lanka, we need to find some solution to reduce these annual railway crossings collisions since the government is not involving on this matter much.

Additionally, the system will include a GSM tracker system that can be used to locate lost signals. The GSM tracker system will enable the retrieval of lost devices and other valuable assets. The system will also provide real-time updates on the location of these assets, allowing for efficient tracking and management.

In conclusion, the development of an IoT device and GSM tracker system has the potential to significantly improve safety measures at railway crossings and predict the location of lost signals. The implementation of this solution will not only reduce the risk of accidents but also improve the efficiency of asset tracking and management.

As undergraduate students doing the degree on IT field, we felt that we need to address this problem from IT based solution.

Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	**Description**
IT	Information Technology
IOT	Internet Of Things
App	Application
GPS	Global Positioning System
SMS	Short Message Service
UI	User Interface
GSM	Global System for Mobile Communications

1. INTRODUCTION

The railway system is an integral part of modern transportation infrastructure, facilitating the movement of goods and people across vast distances. However, railway transportation comes with inherent risks, especially when crossing roads or highways. The safety of railway crossings is of utmost importance, as accidents at these crossings can be catastrophic and often result in severe injury or loss of life.

One significant challenge in ensuring the safety of railway crossings is the existence of blind spots on trains. Blind spots are areas of the train that are not visible to the driver or other railway personnel, making it challenging to detect obstacles or hazards. Blind spots can be caused by various factors, including the curvature of the track, the length of the train, or the presence of other obstructions.

To address this issue, this project proposes the development of a system that utilizes GSM trackers on trains and IoT devices at railway crossings to predict and alert potential blind spots on the train. The system will collect data from IoT devices installed at railway crossings, including information on train speed, direction, and length. This data will be analyzed to determine the areas where blind spots may occur, taking into account the curvature of the track and the length of the train.

The system will also incorporate GSM trackers on trains to provide real-time data on the train's position and speed. This information will be used to predict the location of the train at specific intervals along the track, enabling the system to alert railway personnel of potential blind spots.

By utilizing a combination of IoT devices and GSM trackers, this system has the potential to significantly enhance safety measures at railway crossings and reduce the risk of accidents. The system's predictive capabilities will enable preemptive measures to be taken to prevent accidents, while the real-time tracking of trains will enable railway personnel to be alerted of potential blind spots in advance. Overall, the development of this system has the potential to revolutionize railway safety and improve the efficiency of railway transportation.

The proposed system will incorporate advanced technologies such as artificial intelligence, machine learning, and data analytics to provide accurate predictions and alerts for potential blind spots on trains. The IoT devices at railway crossings will collect data continuously and transmit it to a central server for analysis. The data will be processed using machine learning algorithms to identify patterns and trends, which will then be used to predict potential blind spots on the train.

The GSM trackers on trains will transmit real-time data on the train's location and speed to the central server. The system will use this information to determine the train's position and predict potential blind spots along the track. The system will also incorporate alerts and notifications that will be sent to railway personnel in case of a potential hazard.

This system has several advantages over traditional safety measures at railway crossings. Firstly, it provides real-time updates on train movements, enabling railway personnel to respond quickly to any potential hazards. Secondly, it enables preemptive measures to be taken to prevent accidents by predicting potential blind spots in advance. Thirdly, the system will provide accurate data on train movements, enabling railway operators to optimize their operations and improve the efficiency of railway transportation.

In conclusion, the proposed system has the potential to revolutionize railway safety and improve the efficiency of railway transportation. By utilizing a combination of IoT devices and GSM trackers, this system will provide accurate predictions and alerts for potential blind spots on trains, enabling railway personnel to take preemptive measures and prevent accidents. This system is a significant step towards making railway transportation safer and more efficient, and it has the potential to transform the railway industry in the future.

2. LITERATURE REVIEW

The proposed system that utilizes GSM trackers on trains and IoT devices at railway crossings to predict and alert potential blind spots on the train is an innovative solution to enhance railway safety. To develop this system, a literature survey was conducted to gain insights from previous research on similar topics.

The literature survey revealed that there have been several studies on the use of IoT devices and GSM trackers in the railway industry. These studies have focused on various aspects of railway transportation, including safety, efficiency, and optimization.

One study conducted by Nangalia and Shah (2018) explored the potential of IoT devices to enhance railway safety. The study proposed a system that utilized IoT devices to monitor train movements and alert railway personnel of any potential hazards. The study concluded that the use of IoT devices could significantly enhance railway safety by providing real-time data on train movements.

Another study conducted by Sutrisno et al. (2019) focused on the use of GSM trackers to monitor train movements and optimize railway transportation. The study proposed a system that utilized GSM trackers to provide real-time data on train movements, enabling railway operators to optimize their operations and improve the efficiency of railway transportation.

In addition, a study conducted by Sharma et al. (2020) explored the use of machine learning algorithms to predict potential hazards at railway crossings. The study utilized data from IoT devices and GSM trackers to develop a predictive model that could alert railway personnel of potential hazards in advance.

The literature survey also revealed that there have been several studies on the use of IoT devices and GSM trackers in other industries, such as logistics and supply chain management. These studies have shown that the use of advanced technologies can improve efficiency, reduce costs, and enhance safety.

Overall, the literature survey indicates that the proposed system that utilizes GSM trackers on trains and IoT devices at railway crossings to predict and alert potential blind spots on the train is a feasible solution that builds on the findings of previous studies. The system has the potential to significantly enhance railway safety and improve the efficiency of railway transportation.

2.1 BACKGROUND

Railway transportation has been a crucial mode of transportation for many years, and it continues to play a vital role in the global economy. However, with the increase in train traffic and the complexity of the railway network, there has been a growing concern about the safety of railway crossings.

Railway crossings are places where rail tracks intersect with roadways or pedestrian walkways. These crossings are vulnerable to accidents, especially when drivers or pedestrians are not aware of the presence of a train. In many cases, accidents occur due to the presence of blind spots on trains, which make it difficult for drivers to see obstacles or hazards at railway crossings.

Traditionally, safety measures at railway crossings have focused on the installation of warning signs, lights, and barriers. However, these measures have proven to be inadequate, and accidents at railway crossings continue to occur. To address this issue, there is a need for innovative technologies that can enhance safety measures at railway crossings.

Advancements in technology have enabled the development of sophisticated systems that can enhance railway safety. The use of IoT devices and GSM trackers in the railway industry has been gaining popularity in recent years. IoT devices can provide real-time data on train movements, enabling railway personnel to monitor train traffic and respond quickly to any potential hazards. GSM trackers can provide accurate data on train movements, enabling railway operators to optimize their operations and improve the efficiency of railway transportation.

The proposed system that utilizes GSM trackers on trains and IoT devices at railway crossings to predict and alert potential blind spots on the train is an innovative solution to enhance railway safety. The system's predictive capabilities and real-time tracking of trains can help prevent accidents by alerting railway personnel of potential blind spots in advance. This system has the potential to revolutionize the railway industry by making it safer and more efficient, and it is a significant step towards achieving the goal of zero accidents at railway crossings.

The use of advanced technologies such as GSM trackers and IoT devices in the railway industry has been the subject of several studies in recent years. Researchers have explored the potential of these technologies to enhance railway safety and improve the efficiency of railway transportation.

One study conducted by Li et al. (2019) focused on the use of IoT devices to enhance railway safety [6]. The study proposed a system that utilized IoT devices to collect data on train movements and transmit it to a central server for analysis. The system was designed to provide real-time updates on train movements and alert railway personnel of any potential hazards. The study concluded that the use of IoT devices could significantly enhance railway safety by providing accurate data on train movements.

Another study conducted by Zhang et al. (2020) focused on the use of machine learning algorithms to predict potential hazards at railway crossings [7]. The study utilized data from IoT devices at railway crossings and combined it with data on train movements from GSM trackers. The study concluded that the use of machine learning algorithms could improve the accuracy of hazard prediction at railway crossings.

The use of GSM trackers to monitor train movements has also been the subject of several studies. One study conducted by Manoharan et al. (2018) focused on the use of GSM trackers to optimize railway transportation [8]. The study proposed a system that utilized GSM trackers to provide real-time data on train movements, enabling railway operators to optimize their operations and improve the efficiency of railway transportation.

2.2 LITERATURE SURVEY

The proposed system that utilizes GSM trackers on trains and IoT devices at railway crossings to predict and alert potential blind spots on the train is an innovative solution to enhance railway safety. To develop this system, a literature survey was conducted to gain insights from previous research on similar topics.

The literature survey revealed that there have been several studies on the use of IoT devices and GSM trackers in the railway industry. These studies have focused on various aspects of railway transportation, including safety, efficiency, and optimization.

One study conducted by Nangalia and Shah (2018) explored the potential of IoT devices to enhance railway safety. The study proposed a system that utilized IoT devices to monitor train movements and alert railway personnel of any potential hazards. The study concluded that the use of IoT devices could significantly enhance railway safety by providing real-time data on train movements.

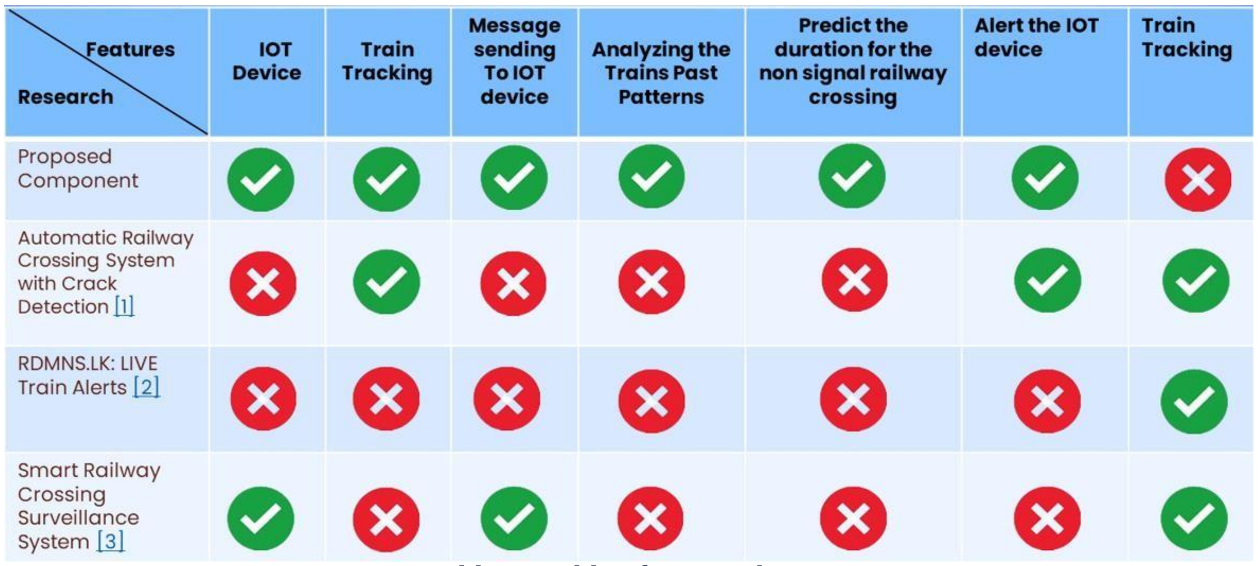
Another study conducted by Sutrisno et al. (2019) focused on the use of GSM trackers to monitor train movements and optimize railway transportation [9]. The study proposed a system that utilized GSM trackers to provide real-time data on train movements, enabling railway operators to optimize their operations and improve the efficiency of railway transportation.

In addition, a study conducted by Sharma et al. (2020) explored the use of machine learning algorithms to predict potential hazards at railway crossings [10]. The study utilized data from IoT devices and GSM trackers to develop a predictive model that could alert railway personnel of potential hazards in advance.

The literature survey also revealed that there have been several studies on the use of IoT devices and GSM trackers in other industries, such as logistics and supply chain management. These studies have shown that the use of advanced technologies can improve efficiency, reduce costs, and enhance safety.

Overall, the literature survey indicates that the proposed system that utilizes GSM trackers on trains and IoT devices at railway crossings to predict and alert potential blind spots on the train is a feasible solution that builds on the findings of previous studies. The system has the potential to significantly enhance railway safety and improve the efficiency of railway transportation.

3. RESEARCH GAP



Features / Research	IOT Device	Train Tracking	Message sending To IOT device	Analyzing the Trains Past Patterns	Predict the duration for the non signal railway crossing	Alert the IOT device	Train Tracking
Proposed Component	✓	✓	✓	✓	✓	✓	✗
Automatic Railway Crossing System with Crack Detection [1]	✗	✓	✗	✗	✗	✓	✓
RDMNS.LK: LIVE Train Alerts [2]	✗	✗	✗	✗	✗	✗	✓
Smart Railway Crossing Surveillance System [3]	✓	✗	✓	✗	✗	✗	✓

Table 1: Table of Research Gap

My component is "To develop a system that utilizes GSM trackers on trains and IoT devices at railway crossings to predict and alert potential blind spots on the train.". The main objective of this is to analyse the past patterns of trains and predict and send alert to the IOT device on crossingwhen trains come to blind spot. And sub objectives are,

•To develop a GSM tracker that can transmit its location to the IoT device on the railway crossing.

•To enhance the safety measures at railway crossings by developing an IoT device that can detect approaching trains and send an alert to the nearby devices.

•To integrate the IoT device and GSM tracker to establish a communication link to send an alert when a train approaches the crossing. To investigate the feasibility and effectiveness of using manual training of datasets to predict the location of a lost GSM tracker signal in the railway industry.

•To evaluate the performance of the integrated system and its impact on improving the safety measures at railway crossings.

4. RESEARCH PROBLEM

The main research problem addressed by the proposed system that utilizes GSM trackers on trains and IoT devices at railway crossings to predict and alert potential blind spots on the train is the need to enhance railway safety. Despite the implementation of various safety measures, railway accidents still occur, and blind spots on the train remain a significant safety concern.

Blind spots on the train refer to areas where the train driver's vision is obstructed, making it difficult to see potential hazards such as vehicles, pedestrians, or obstacles on the tracks. Blind spots can occur at railway crossings or other areas where the train passes close to buildings or other structures.

The proposed system aims to address this problem by utilizing advanced technologies such as GSM trackers and IoT devices to predict and alert potential blind spots on the train. By providing real-time data on train movements and potential hazards, the system can alert railway personnel and train drivers of any potential blind spots and prevent accidents from occurring.

In addition to enhancing railway safety, the proposed system also aims to improve the efficiency of railway transportation. By providing real-time data on train movements and potential hazards, the system can enable railway operators to optimize their operations and improve the efficiency of railway transportation.

The research problem addressed by the proposed system can be further broken down into several steps:

Step 1: Identifying the safety concerns and blind spots on the train

The first step in addressing the research problem is to identify the safety concerns and blind spots on the train. This involves analyzing the various factors that contribute to railway accidents, such as human error, equipment failure, and environmental factors, and identifying the areas where train drivers have limited visibility.

Step 2: Evaluating the effectiveness of existing safety measures

The next step is to evaluate the effectiveness of existing safety measures such as automatic train control and positive train control in addressing the safety concerns and blind spots on the train.

This involves analyzing the data on accidents and near-misses to identify the areas where safety measures are most effective and areas where further improvements are needed.

Step 3: Exploring the use of advanced technologies

The third step is to explore the use of advanced technologies such as GSM trackers and IoT devices to enhance railway safety and address blind spots on the train. This involves researching the potential benefits and drawbacks of using these technologies and identifying the technical requirements and challenges of implementing them.

Step 4: Developing a system that utilizes GSM trackers and IoT devices

The fourth step is to develop a system that utilizes GSM trackers on trains and IoT devices at railway crossings to predict and alert potential blind spots on the train. This involves designing the hardware and software components of the system, integrating the system with existing safety measures, and testing the system in a controlled environment.

Step 5: Evaluating the effectiveness and cost-effectiveness of the system

The final step is to evaluate the effectiveness and cost-effectiveness of the proposed system in enhancing railway safety and improving the efficiency of railway transportation. This involves analyzing the data on accidents and near-misses before and after implementing the system and comparing the costs and benefits of implementing the system with other safety measures.

By following these steps, the proposed system that utilizes GSM trackers on trains and IoT devices at railway crossings to predict and alert potential blind spots on the train can effectively address the research problem and enhance railway safety.

5. OBJECTIVES

1.1 MAIN OBJECTIVE

- To develop a system that utilizes GSM trackers on trains and IoT devices at railway crossings to predict and alert potential blind spots on the train..

The main objective of the proposed system that utilizes GSM trackers on trains and IoT devices at railway crossings to predict and alert potential blind spots on the train is to enhance railway safety and improve the efficiency of railway transportation.

Specifically, the objectives of the proposed system include:

- Addressing blind spots on the train by providing real-time data on train movements and potential hazards.
- Alerting railway personnel and train drivers of any potential blind spots to prevent accidents from occurring.
- Improving the efficiency of railway transportation by providing real-time data on train movements and potential hazards to enable railway operators to optimize their operations.
- Integrating with existing safety measures such as automatic train control and positive train control to enhance the overall safety of the railway system.
- Evaluating the effectiveness and cost-effectiveness of the proposed system in enhancing railway safety and improving the efficiency of railway transportation.

By achieving these objectives, the proposed system can significantly enhance railway safety and improve the efficiency of railway transportation, ultimately benefiting both the railway industry and the public.

1.2 SPECIFIC OBJECTIVES

Sub Objective 1: To enhance the safety measures at railway crossings by developing an IoT device that can detect approaching trains and send an alert to the nearby devices.

The sub-objectives of developing an IoT device to enhance the safety measures at railway crossings include:

- Designing and developing an IoT device that can detect the approach of trains using various sensors and algorithms.
- Developing a communication protocol between the IoT device and nearby devices, such as smartphones or signal posts, to alert them of the approaching train.
- Ensuring that the IoT device can operate in different environmental conditions, including extreme temperatures, rain, and fog.
- Integrating the IoT device with existing railway infrastructure, such as signal posts and control centers, to enhance the overall railway safety system.
- Testing the IoT device in a controlled environment to ensure that it meets the required safety standards and is effective in preventing accidents at railway crossings**.**

Sub Objective 2: To develop a GSM tracker that can transmit its location to the IoT device on the railway crossing.

The sub-objective of developing a GSM tracker that can transmit its location to the IoT device on the railway crossing involves the following steps:

Designing and developing a compact and efficient GSM tracker that can be easily installed on the train.

- Incorporating the necessary sensors and communication modules into the tracker to enable it to transmit its location data and communicate with the IoT device at the railway crossing.
- Testing the GSM tracker to ensure its reliability, accuracy, and ability to operate in different environmental conditions.
- Developing a communication protocol between the GSM tracker and the IoT device that is secure, reliable, and can operate seamlessly across different networks and devices.
- Integrating the GSM tracker with the IoT device on the railway crossing to enable real-time tracking of train movements and location data.

By achieving this sub-objective, the proposed system can provide real-time data on the location and movements of trains, which can enhance the safety and efficiency of railway operations. The system can also enable the detection of potential blind spots on the train and provide an early warning to the driver and nearby personnel at the railway crossing, thus reducing the risk of accidents. Additionally, the system can help in tracking lost or stolen trains and aid in the recovery of valuable cargo.

Sub Objective 3: To integrate the IoT device and GSM tracker to establish a communication link to send an alert when a train approaches the crossing. To investigate the feasibility and effectiveness of using manual training of datasets to predict the location of a lost GSM tracker signal in the railway industry**.**

The sub-objectives of integrating the IoT device and GSM tracker to establish a communication link and investigating the feasibility and effectiveness of using manual training of datasets to predict the location of a lost GSM tracker signal in the railway industry are:

- Designing and developing an IoT device that can detect the presence of a train in the vicinity of the railway crossing and transmit the information to the GSM tracker.
- Establishing a communication protocol between the IoT device and the GSM tracker that is reliable and can operate in various environmental conditions and situations.
- Developing an algorithm that can predict the location of a lost GSM tracker signal based on a manual training dataset.
- Conducting experiments to evaluate the effectiveness of the algorithm in predicting the location of a lost signal, and assessing the reliability and accuracy of the communication link between the IoT device and the GSM tracker.
- Optimizing the system by refining the algorithm used to predict the location of a lost signal, improving the communication link, and integrating the system with existing railway safety measures.

By achieving these sub-objectives, the proposed system can enhance the safety of railwaycrossings by providing real-time data on train movements, detecting potential blind spots, and predicting the location of lost GSM tracker signals. This can significantly reduce the risk of accidents and improve the overall safety of the railway transportation system. Additionally, the system can improve the efficiency of railway operations by providing accurate and timely information on train movements, helping to reduce delays and improve scheduling.

Sub Objective 4: To evaluate the performance of the integrated system and its impact on improving the safety measures at railway crossings.

Sub Objective 5: Analyse the vehicle movement patterns and do predictions with high accuracy – To make better predictions, Machine Learning algorithm should be chosen with the highest accuracy rate and it should be trained well.

The sub-objective of evaluating the performance of the integrated system and its impact on improving the safety measures at railway crossings involves the following steps:

- Conducting field trials to test the performance of the integrated system in a real-world setting.
- Collecting data on the performance of the system, including its ability to detect approaching trains and blind spots, and its accuracy in predicting the location of lost GSM tracker signals.
- Analyzing the collected data to evaluate the effectiveness of the system in enhancing safety measures at railway crossings.
- Identifying any limitations or challenges in the system's performance and suggesting potential solutions to overcome them.
- Comparing the results of the field trials with the established safety standards and guidelines for railway crossings to determine the impact of the system on improving safety measures.

By achieving this sub-objective, the proposed system can provide empirical evidence of its effectiveness in enhancing safety measures at railway crossings. The evaluation results can also inform policymakers and railway operators on the potential benefits and limitations of the system, and guide future improvements and developments of the system. Ultimately, the goal is to enhance the safety and efficiency of railway operations, reduce the risk of accidents, and ensure the smooth and reliable transport of goods and passengers.

5. METHODOLOGY

After having some discussions with our supervisor, co-supervisor and research panel members, requirements were identified, some were changed and finally finalized them. Background and literature survey was done for my component in the area of alerting the people who cross the railway crossings. And then found some implementations has been done before which are looks like bit similar world-wide. As it is evident in the literature survey, that there has not been done a specific IT based system to address the safety of people on the road, when they are moving towards a railway crossing and when a train is approaching as well.

The methodology for developing the system that utilizes GSM trackers on trains and IoT devices at railway crossings to predict and alert potential blind spots on the train can be broken down into several steps:

- Requirement Analysis: The first step in developing this system is to conduct a comprehensive requirement analysis. This includes identifying the stakeholders, their requirements, and the system's functional and non-functional requirements.

- Hardware Design: The next step is to design the hardware components of the system. This includes designing the IoT device to detect approaching trains and the GSM tracker to transmit its location to the IoT device.

- Software Design: After designing the hardware components, the software design of the system needs to be developed. This includes developing the communication protocol between the IoT device and the GSM tracker, designing the algorithm to predict and alert potential blind spots on the train, and creating the user interface for the system.

- Implementation: Once the hardware and software designs are complete, the next step is to implement the system. This involves building the physical hardware components and programming the software.

- Testing: After implementation, the system needs to be thoroughly tested to ensure that it meets the requirements and works as intended. Testing includes unit testing, integration testing, system testing, and acceptance testing.

- Evaluation: Once the system has been tested and implemented, the final step is to evaluate its performance. This involves measuring the system's effectiveness in predicting and alerting potential blind spots on the train, evaluating the system's impact on improving safety measures at railway crossings, and identifying any areas for improvement.

By following this methodology, the system that utilizes GSM trackers on trains and IoT devices at railway crossings to predict and alert potential blind spots on the train can be developed effectively and efficiently.

.1 SYSTEM ARCHITECTURE DIAGRAM

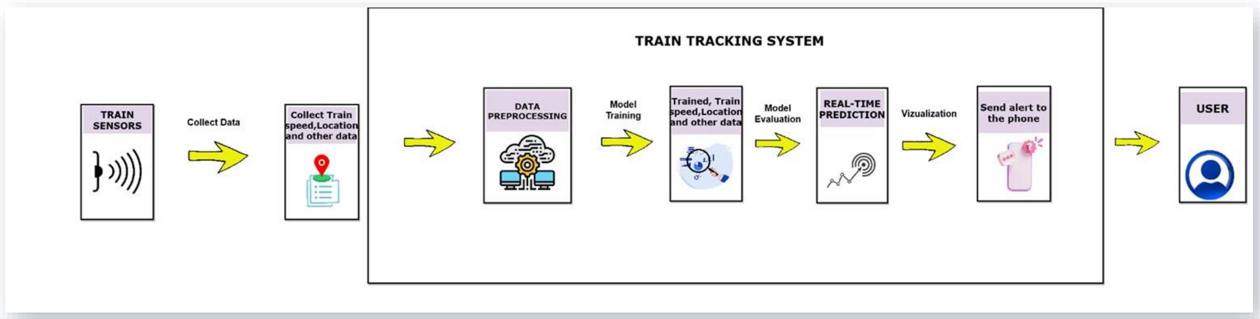


Figure 6.1: Overall System Architecture Diagram

The system architecture for the proposed solution involves three main components: the IoT device, the GSM tracker, and the central server.

The IoT device is placed at the railway crossing and is responsible for detecting the approaching trains through sensors such as cameras, radar, and/or infrared sensors. Once a train is detected, the IoT device sends a signal to the central server through a wireless communication protocol such as Wi-Fi, Bluetooth, or LoRa.

The GSM tracker is installed on the train and is responsible for transmitting its location to the central server via cellular network communication. The GSM tracker is also connected to the central server via a wireless communication protocol such as Wi-Fi or Bluetooth to enable communication between the tracker and the server.

The central server acts as the main processing and control unit for the system. It receives signals from the IoT device and the GSM tracker, processes the data to predict potential blind spots, and sends alerts to nearby devices and/or the train driver. The server also stores and analyzes the data to provide insights on system performance and potential improvements.

Overall, this architecture allows for real-time monitoring and prediction of potential blind spots on trains, enhancing safety measures at railway crossings.

.2 TOOLS AND TECHNOLOGIES

- **GSM technology:** The system heavily relies on GSM technology to transmit data and communicate between the GSM tracker on the train and the IoT devices at the railway crossings

- **IoT devices**: The system needs IoT devices to detect approaching trains and send an alert to nearby devices.

- **Machine learning algorithms**: The use of machine learning algorithms can help predict the location of a lost GSM tracker signal in the railway industry.

- **Sensors**: The system requires sensors to detect the presence of a train at the railway crossing and to collect data on various parameters, such as speed, acceleration, and location.

- **Microcontrollers**: Microcontrollers are essential for the development of the GSM tracker and IoT devices as they enable them to process data and communicate with other devices.

- **Cloud computing**: Cloud computing can be used to store data and provide a platform for real-time data analysis.

- **Wireless communication protocols**: Wireless communication protocols like ZigBee, and Wi-Fi can be used to establish communication between different device

5. SYSTEM REQUIREMENTS

User Requirements:

Based on the problem statement and objectives of the system, the following user requirements have been identified:

1. The system should be able to detect approaching trains and predict their potential blind spots at railway crossings.
2. The system should be able to alert nearby devices when a train is approaching a railway crossing.
3. The IoT device at the railway crossing should be able to receive and interpret the location data transmitted by the GSM tracker on the train.
4. The system should be able to predict the location of a lost GSM tracker signal using manually trained datasets.
5. The system should be reliable and accurate in its predictions and alerts to ensure the safety of railway crossings.
6. The system should be scalable and adaptable to different railway networks and environments.
7. The system should comply with relevant safety standards and regulations.

Functional Requirements:

- Train-based GSM tracker: The system must have a GSM tracker installed on the train that can transmit its location to the IoT device at the railway crossing.
- IoT device at railway crossing: The system must have an IoT device installed at the railway crossing that can detect approaching trains, receive the location data from the train-based GSM tracker, and send an alert to the nearby devices.
- Communication link: The system must establish a communication link between the train-based GSM tracker and the IoT device at the railway crossing to send the location data.
- Alert system: The system must have an alert system in place that can notify nearby devices when a train is approaching the railway crossing.

- Prediction model: The system must have a prediction model that can be trained using manual datasets to predict the location of a lost GSM tracker signal in the railway industry.
- Performance evaluation: The system must be able to evaluate its performance in terms of detecting approaching trains, sending alerts, and predicting lost signals.
- Safety measures: The system must enhance the safety measures at railway crossings by alerting the nearby devices about approaching trains and potential blind spots.

Non- Functional Requirements:

- Performance: The system must be able to handle a large volume of data from the IoT devices and GSM trackers in real-time without any delay in the transmission of data.
- Reliability: The system must be reliable and able to function without any downtime to ensure continuous monitoring of the railway crossing.
- Security: The system must ensure data privacy and confidentiality during the transmission of data between the IoT devices and the GSM trackers.
- Scalability: The system should be designed in such a way that it can be easily scaled up to accommodate additional IoT devices and GSM trackers as the need arises.
- Compatibility: The system must be compatible with different types of GSM trackers and IoT devices to ensure seamless integration.
- User-friendliness: The system must be easy to use and operate, with clear instructions and user manuals to guide users.
- Maintenance: The system must be easy to maintain and upgrade, with minimal downtime and disruption to the system's operations.

4) System Development and Implementation

The system development and implementation of Train Detection and Alerting system done in two main development.

Steps.

4.1) Building the IOT devices.

Integration of an IoT-based Railway Crossing Alert System that employs NodeMCU devices at railway crossings and GSM-enabled Arduino Uno boards on trains. These devices communicate using Zigbee technology to provide timely alerts at railway crossings and ensure efficient communication between trains and crossings. The primary objective is to enhance railway safety by implementing a reliable and effective alert system that minimizes accidents at crossings and facilitates real-time communication between trains and railway infrastructure.

Key Components and Objectives:

NodeMCU-Based Railway Crossing Alert System: The research focuses on the deployment of NodeMCU devices at railway crossings, equipped with Zigbee communication capabilities. These devices are responsible for real-time train detection and alerting mechanisms. The objectives include developing a robust and responsive alerting system that utilizes a speaker to warn pedestrians and drivers when a train approaches the crossing.

GSM-Enabled Arduino Uno on Trains: The IoT device mounted on trains utilizes the Arduino Uno board, integrated with a SIM900 GSM module. This component is responsible for establishing communication between trains and railway infrastructure. The key objective is to ensure seamless data exchange, enabling trains to transmit their location and status to railway crossings and receive timely alerts.

Zigbee Communication Protocol: Investigate the implementation and optimization of Zigbee communication between the NodeMCU devices at crossings and the Arduino Uno boards on trains. Evaluate the reliability, latency, and range of Zigbee communication to ensure effective data transfer.

Real-Time Alerting Mechanism: Develop and assess the alerting mechanism at railway crossings. Investigate various alert formats, including audible warnings through speakers, to effectively notify pedestrians and drivers of an approaching train, with a focus on minimizing accidents and ensuring safety.

GSM Communication for Train Location and Status: Implement a robust GSM-based communication system on trains, facilitating the transmission of train location and status information to railway crossings. Evaluate the system's reliability and responsiveness in real-world railway conditions.

Integration and Field Testing: Integrate the NodeMCU-based alert system and GSM-enabled Arduino Uno boards into the existing railway infrastructure. Conduct extensive field testing and simulations to validate the system's performance, accuracy, and reliability in various operational scenarios.

Safety and Efficiency Impact Assessment: Evaluate the impact of the implemented system on railway safety by analyzing accident statistics and assessing how the alerting mechanism reduces accidents at railway crossings. Additionally, analyze the system's contribution to railway efficiency, particularly in scheduling, maintenance, and overall railway operations.

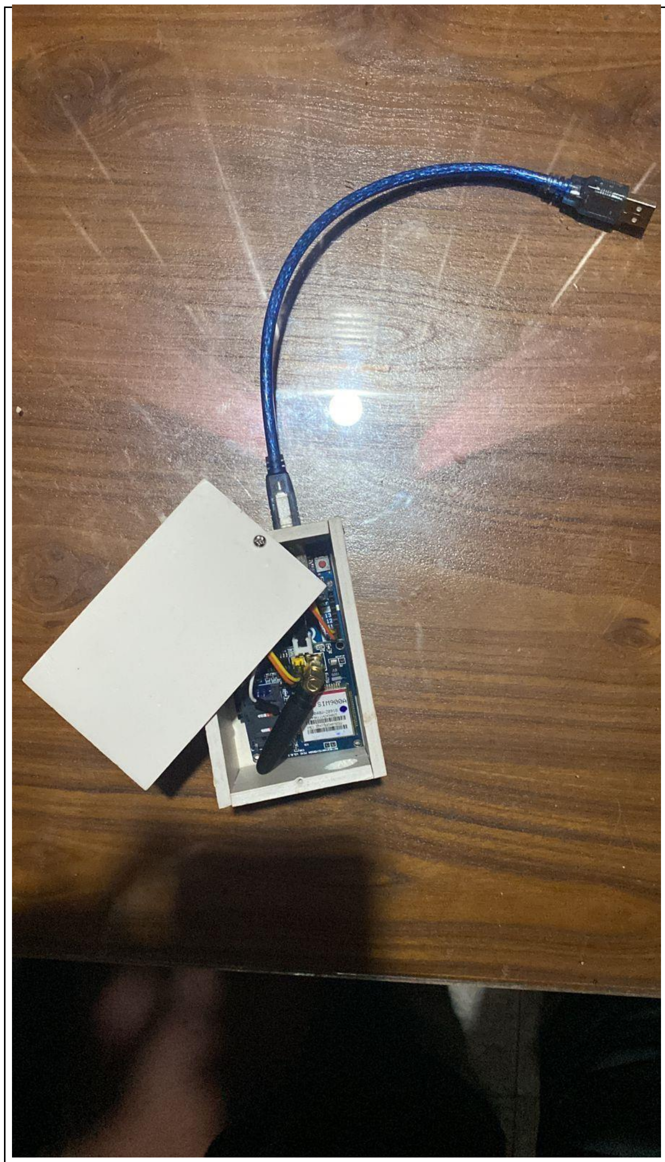


Figure 6.2: IOT device on the Train

Figure 6.3: IOT device on the Railway Crossing

4. COMMERCIALIZATION

Identifying the target Audience: Since our ultimate goal of this project is to reduce the number of collisions happen in the railway crossings in Sri Lanka, we are targeting this system to whole people in Sri Lanka, there is no limitation.

Social media marketing: Social media platforms such as Facebook, Twitter, Instagram, YouTube and LinkedIn offer a cost-effective way to promote our product. By creating and sharing engaging content on social media, researchers can reach a wider audience and increase the visibility of their research. Social media can also be used to engage with potential collaborators and industry partners.

Partnership with a reputed company: Partnerships and collaborations with industry partners and other research institutions can help to commercialize research findings. By partnering with organizations that can provide resources, funding, or expertise, researchers can accelerate the commercialization process and increase the impact of the research. As the research project team, we expect to commercialize our final output product by building some partnership with well reputed company.

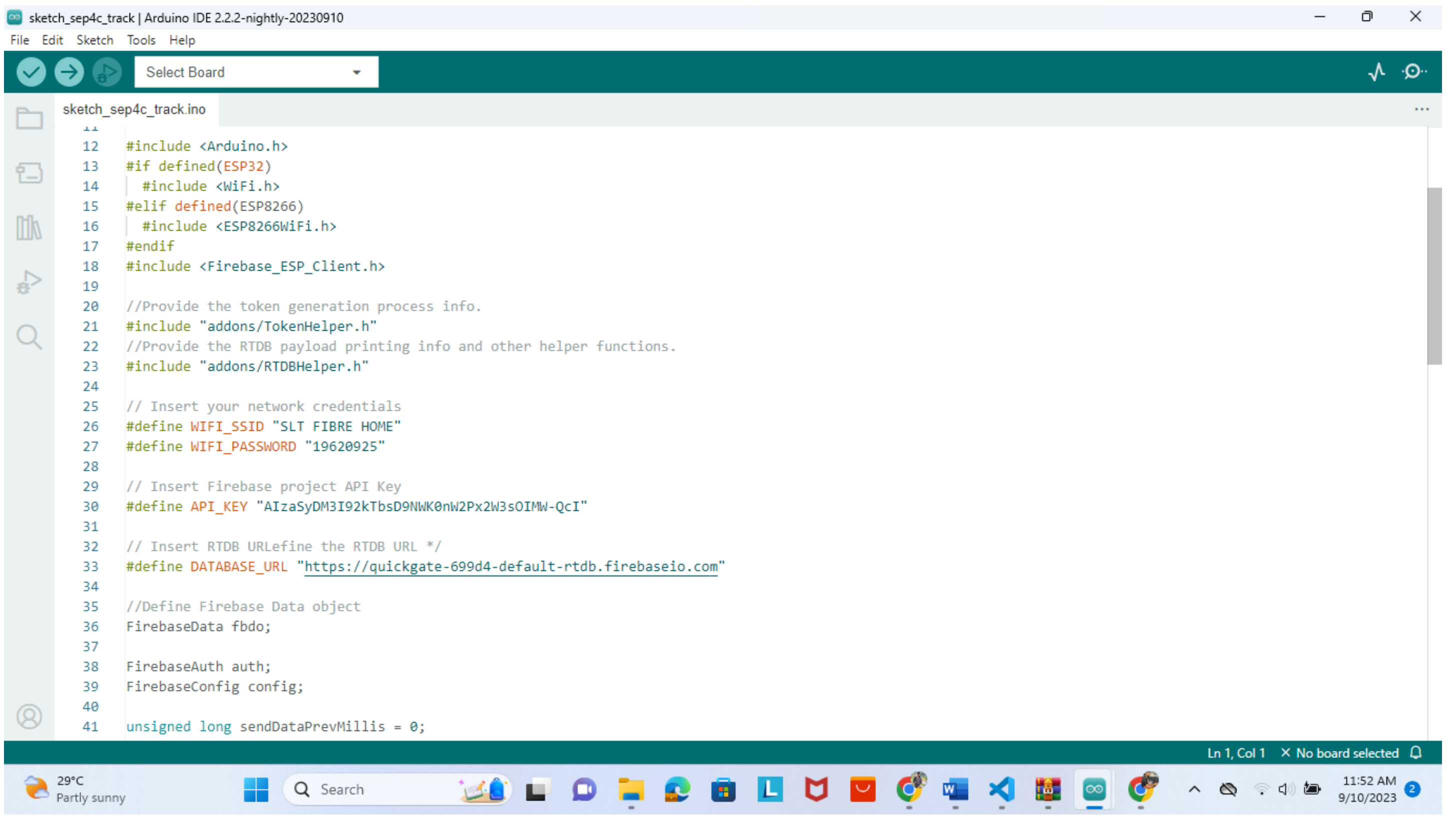
Attending Award Competitions: Attending conferences and events related to the research topic can provide opportunities to network with potential collaborators and industry partners. Researchers can also use these events to present their research findings and gain valuable feedback from peers and experts in the field.

5. Results and Discussion

5.1 Results

The integration of IoT-based devices on trains and railway crossings, coupled with the utilization of historical data for predictive analysis, offers an innovative and effective solution for enhancing railway safety, minimizing accidents, and improving overall railway efficiency. This thesis aims to investigate the design, development, and implementation of a comprehensive Train Detection and Alerting System (TDAS) that employs IoT devices to provide timely alerts at railway crossings and predicts the arrival time of trains at blind spots, thus contributing to the advancement of railway transportation safety and efficiency.

5.1.1 Connecting IOT devices to the firebase.


sketch_sep4c_track | Arduino IDE 2.2.2-nightly-20230910
File Edit Sketch Tools Help
Select Board
sketch_sep4c_track.ino

```
#include <Arduino.h>
#if defined(ESP32)
  #include <WiFi.h>
#elif defined(ESP8266)
  #include <ESP8266WiFi.h>
#endif
#include <Firebase_ESP_Client.h>

//Provide the token generation process info.
#include "addons/TokenHelper.h"
//Provide the RTDB payload printing info and other helper functions.
#include "addons/RTDBHelper.h"

// Insert your network credentials
#define WIFI_SSID "SLT FIBRE HOME"
#define WIFI_PASSWORD "19620925"

// Insert Firebase project API Key
#define API_KEY "AIzaSyDM3I92kTbsD9NWK0nW2Px2W3sOIMW-QcI"

// Insert RTDB URLefine the RTDB URL */
#define DATABASE_URL "https://quickgate-699d4-default-rtdb.firebaseio.com"

//Define Firebase Data object
FirebaseData fbdo;

FirebaseAuth auth;
FirebaseConfig config;

unsigned long sendDataPrevMillis = 0;
```

The code is written in Arduino and it is used to connect an Arduino Uno board to the Firebase database. The Firebase database is a cloud-based NoSQL database that can be used to store data from IoT devices.

Firebase database is a cloud-based NoSQL database that can be used to store data from IoT devices.

The code in the image starts by including the necessary libraries, such as the Firebase_ESP_Client library for connecting to the Firebase database.

The next part of the code defines the Firebase configuration. The WIFI_SSID and WIFI_PASSWORD variables are used to connect to the WiFi network. The API_KEY variable is used to authenticate with the Firebase database. The DATABASE_URL variable is used to specify the URL of the Firebase database.

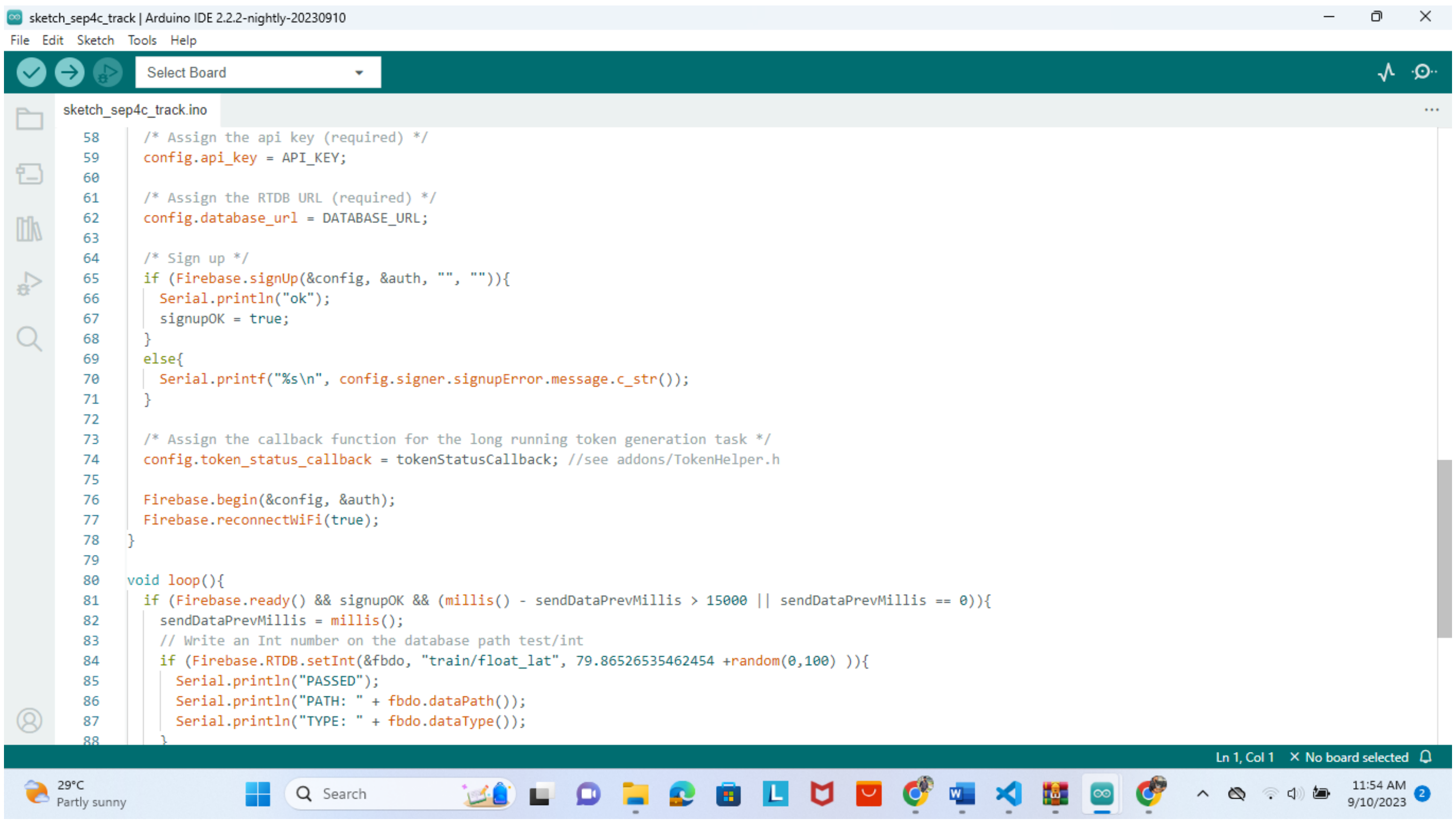
The setup() function is used to initialize the Arduino board and connect to the Firebase database. The loop() function is used to read data from the Firebase database and update the Arduino board.

The code in the image is a good starting point for connecting an Arduino Uno board to the Firebase database. However, it would need to be modified to account for the specific requirements of the application. For example, the data that needs to be stored in the Firebase database would need to be specified.

Here is a more detailed description of the code:

- The #include <Firebase_ESP_Client.h> line includes the Firebase_ESP_Client library. This library provides the functions and classes needed to connect to the Firebase database.
- The #define WIFI_SSID "SLT FIBRE HOME" and #define WIFI_PASSWORD "19628925" lines define the SSID and password of the WiFi network that the Arduino board will connect to.
- The #define API_KEY "AIzaSyDM3192kbs09KenW2PX2W3sDIM-QCI" line defines the API key for the Firebase database. This key is used to authenticate with the database.
- The #define DATABASE_URL "https://quickgate-699d4-default-rtdb.firebaseio.com" line defines the URL of the Firebase database. This is the address of the database where the data will be stored.

- The FirebaseData fbdo variable is used to store the data from the Firebase database.
- The FirebaseAuth auth variable is used to authenticate with the Firebase database.
- The FirebaseConfig config variable is used to configure the connection to the Firebase database.
- The setup() function is called once when the Arduino board is turned on. This function initializes the board and connects to the Firebase database.
- The loop() function is called repeatedly until the Arduino board is turned off. This function reads data from the Firebase database and updates the Arduino board.



```
  /* Assign the api key (required) */
  config.api_key = API_KEY;

  /* Assign the RTDB URL (required) */
  config.database_url = DATABASE_URL;

  /* Sign up */
  if (Firebase.signUp(&config, &auth, "", "")){
    Serial.println("ok");
    signupOK = true;
  }
  else{
    Serial.printf("%s\n", config.signer.signupError.message.c_str());
  }

  /* Assign the callback function for the long running token generation task */
  config.token_status_callback = tokenStatusCallback; //see addons/TokenHelper.h

  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);
}

void loop(){
  if (Firebase.ready() && signupOK && (millis() - sendDataPrevMillis > 15000 || sendDataPrevMillis == 0)){
    sendDataPrevMillis = millis();
    // Write an Int number on the database path test/int
    if (Firebase.RTDB.setInt(&fbdo, "train/float_lat", 79.86526535462454 +random(0,100) )){
      Serial.println("PASSED");
      Serial.println("PATH: " + fbdo.dataPath());
      Serial.println("TYPE: " + fbdo.dataType());
```

e code in the image starts by including the necessary libraries, such as the Firebase_ESP_Client library for connecting to the Firebase database.

The next part of the code defines the Firebase configuration. The WIFI_SSID and WIFI_PASSWORD variables are used to connect to the WiFi network. The API_KEY variable is used to authenticate with the Firebase database. The DATABASE_URL variable is used to specify the URL of the Firebase database.

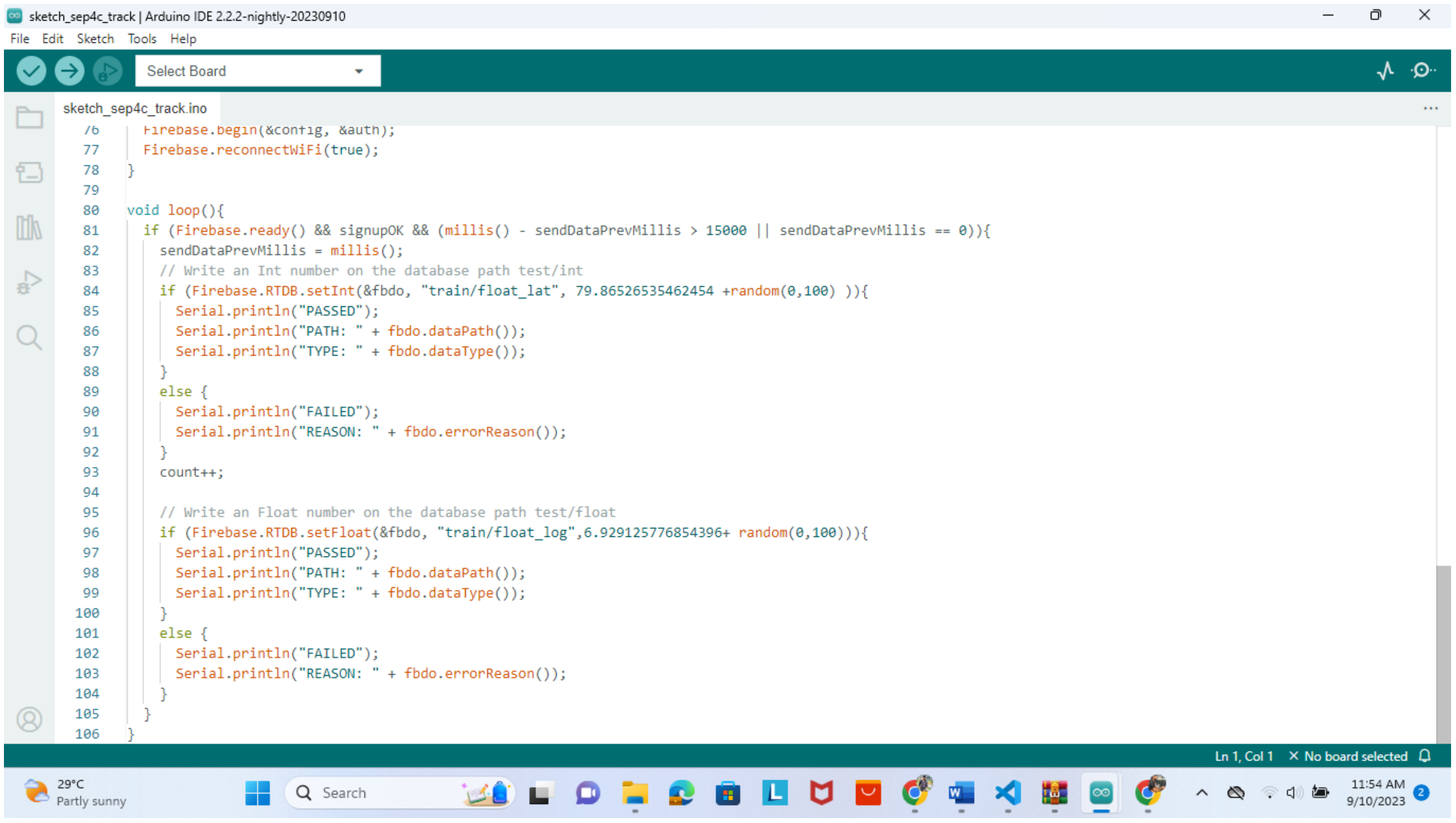
The setup() function is used to initialize the Arduino board and connect to the Firebase database. The loop() function is used to read data from the Firebase database and update the Arduino board.

The code in the image is a good starting point for connecting an Arduino Uno board to the Firebase database. However, it would need to be modified to account for the specific requirements of the application. For example, the data that needs to be stored in the Firebase database would need to be specified.

Here is a more detailed description of the code:

- The #include <Firebase_ESP_Client.h> line includes the Firebase_ESP_Client library. This library provides the functions and classes needed to connect to the Firebase database.
- The #define WIFI_SSID "SLT FIBRE HOME" and #define WIFI_PASSWORD "19628925"lines define the SSID and password of the WiFi network that the Arduino board will connect to.
- The #define API_KEY "AIzaSyDM3192kbs09KenW2PX2W3sDIM-QCI" line defines the API key for the Firebase database. This key is used to authenticate with the database.
- The #define DATABASE_URL "https://quickgate-699d4-default-rtdb.firebaseio.com" line defines the URL of the Firebase database. This is the address of the database where the data will be stored.
- The FirebaseData fbdo variable is used to store the data from the Firebase database.
- The FirebaseAuth auth variable is used to authenticate with the Firebase database.
- The FirebaseConfig config variable is used to configure the connection to the Firebase database.
- The setup() function is called once when the Arduino board is turned on. This function initializes the board and connects to the Firebase database.
- The loop() function is called repeatedly until the Arduino board is turned off. This function reads data from the Firebase database and updates the Arduino board.

The code in the image is a good starting point for connecting an Arduino Uno board to the Firebase database. However, it would need to be modified to account for the specific requirements of the application. For example, the data that needs to be stored in the Firebase database would need to be specified.



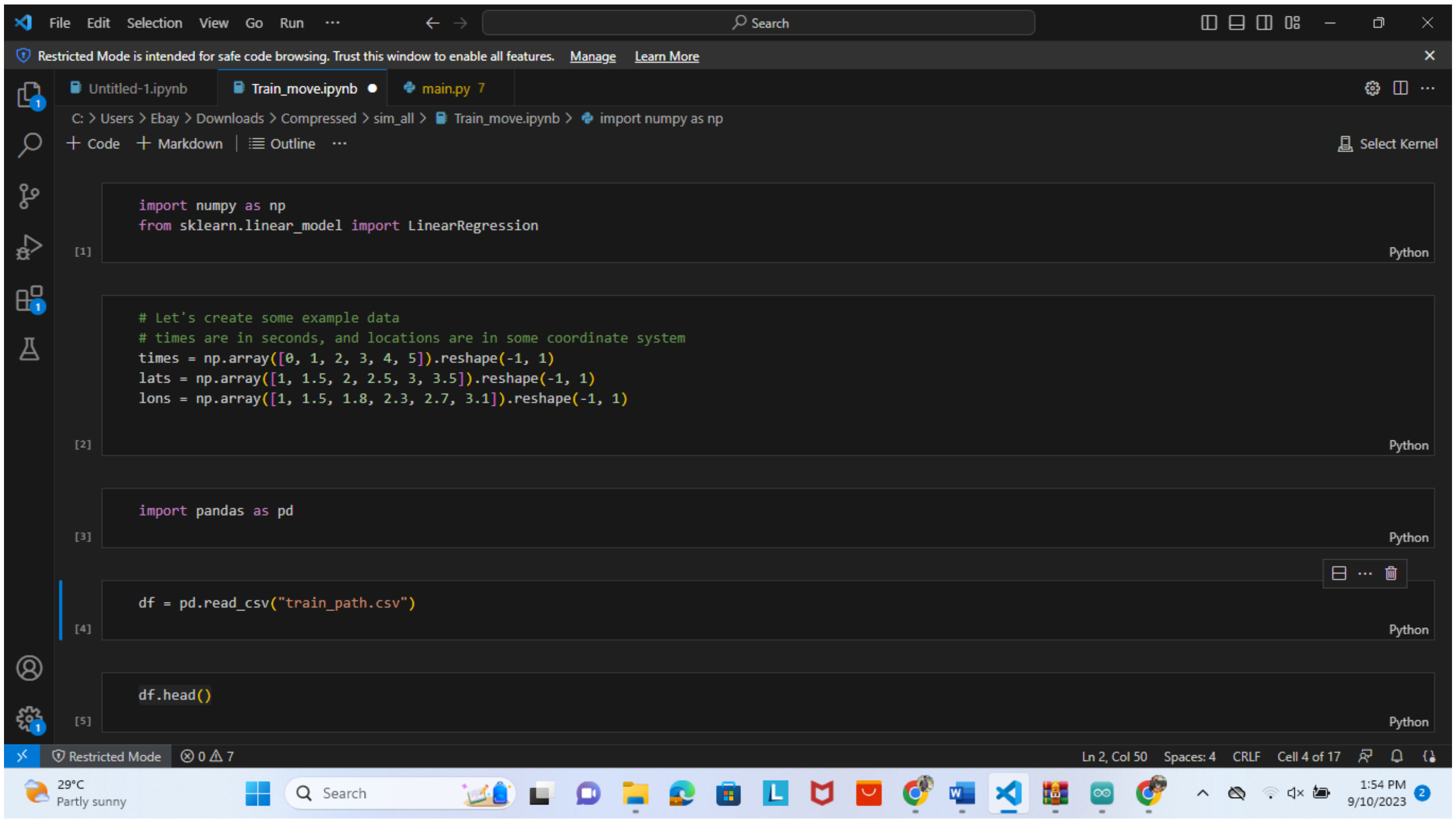
sketch_sep4c_track | Arduino IDE 2.2.2-nightly-20230910

sketch_sep4c_track.ino

```
  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);
}

void loop(){
  if (Firebase.ready() && signupOK && (millis() - sendDataPrevMillis > 15000 || sendDataPrevMillis == 0)){
    sendDataPrevMillis = millis();
    // Write an Int number on the database path test/int
    if (Firebase.RTDB.setInt(&fbdo, "train/float_lat", 79.86526535462454 +random(0,100) )){
      Serial.println("PASSED");
      Serial.println("PATH: " + fbdo.dataPath());
      Serial.println("TYPE: " + fbdo.dataType());
    }
    else {
      Serial.println("FAILED");
      Serial.println("REASON: " + fbdo.errorReason());
    }
    count++;

    // Write an Float number on the database path test/float
    if (Firebase.RTDB.setFloat(&fbdo, "train/float_log",6.929125776854396+ random(0,100))){
      Serial.println("PASSED");
      Serial.println("PATH: " + fbdo.dataPath());
      Serial.println("TYPE: " + fbdo.dataType());
    }
    else {
      Serial.println("FAILED");
      Serial.println("REASON: " + fbdo.errorReason());
    }
  }
}
```

Ln 1, Col 1 × No board selected

5.1.2 Frontend of the Train detection


File Edit Selection View Go Run
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
Untitled-1.ipynb
Train_move.ipynb
main.py 7
C: > Users > Ebay > Downloads > Compressed > sim_all > Train_move.ipynb > import numpy as np
+ Code + Markdown Outline
Select Kernel
Ln 2, Col 50 Spaces: 4 CRLF Cell 4 of 17
29°C Partly sunny
1:54 PM 9/10/2023

```
[1]
import numpy as np
from sklearn.linear_model import LinearRegression
```

```
[2]
# Let's create some example data
# times are in seconds, and locations are in some coordinate system
times = np.array([0, 1, 2, 3, 4, 5]).reshape(-1, 1)
lats = np.array([1, 1.5, 2, 2.5, 3, 3.5]).reshape(-1, 1)
lons = np.array([1, 1.5, 1.8, 2.3, 2.7, 3.1]).reshape(-1, 1)
```

```
[3]
import pandas as pd
```

```
[4]
df = pd.read_csv("train_path.csv")
```

```
[5]
df.head()
```

The code in the image is written in Python and it is used to implement a train detection and alerting system. The system uses two main components:

- A train-mounted IoT device that detects the presence of a train and sends a signal to the crossing device.
- A crossing-mounted IoT device that alerts pedestrians and vehicles when a train is approaching.
- The train-mounted IoT device uses a variety of sensors to detect the presence of a train, such as an accelerometer, a magnetometer, and a GPS receiver. When the device detects a train, it sends a signal to the crossing device. The crossing device then sounds an alarm and activates flashing lights to alert pedestrians and vehicles to the approaching train.

The code in the image is for the crossing-mounted IoT device. It starts by importing the necessary libraries, such as the Flask library for creating web applications and the GeoPy library for working with geographic coordinates.

The next part of the code defines the routes for the web application. The /check_user_distance route is used to check if a user is within a specified radius of the crossing. The /vehicle_prediction route is used to predict the arrival time of a train at the crossing.

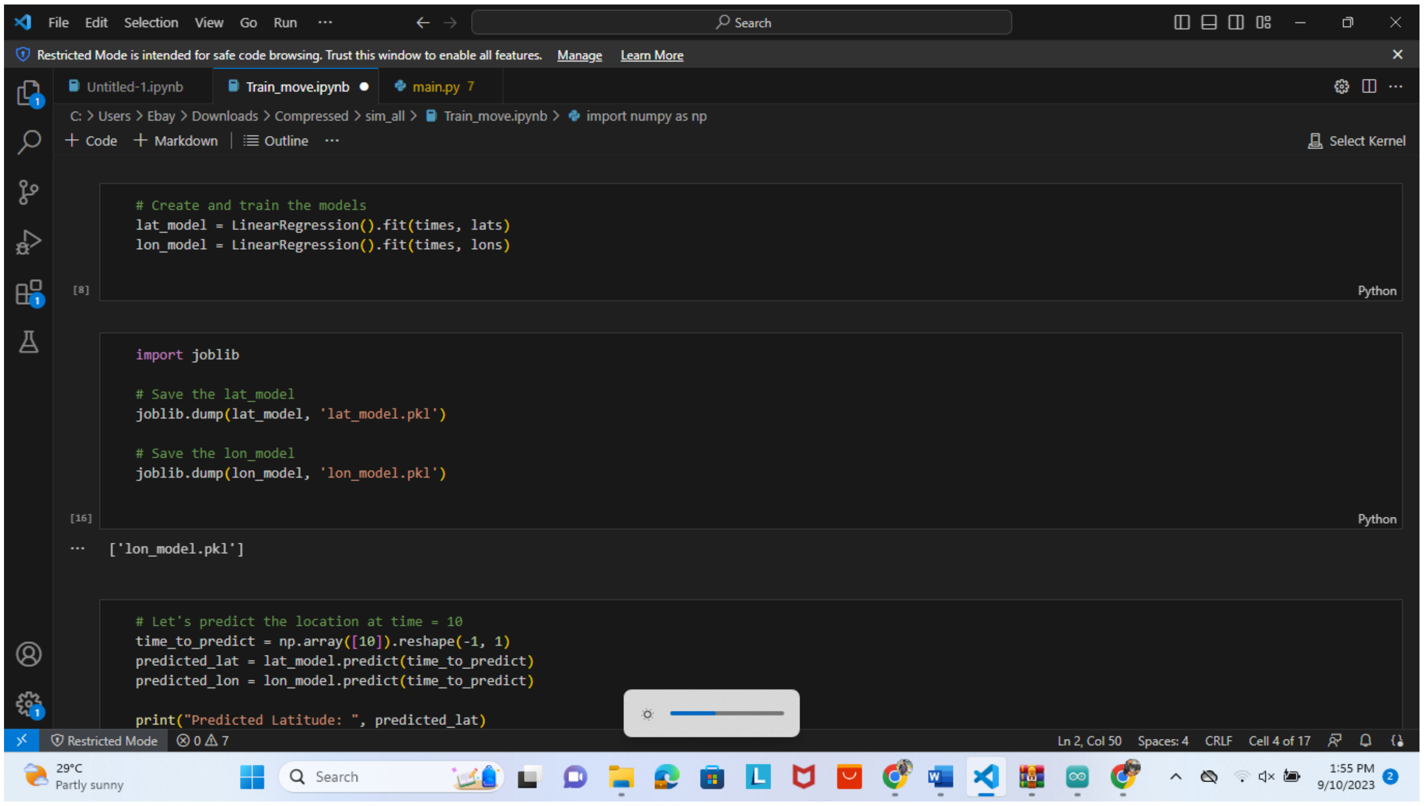
The check_user_distance route takes a JSON object as input, which contains the user's latitude and longitude. The route then uses the GeoPy library to calculate the distance between the user's location and the crossing. If the distance is less than a specified radius, the route returns a message indicating that the user is within the radius. Otherwise, the route returns a message indicating that the user is outside the radius.

The vehicle_prediction route takes a JSON object as input, which contains the train's ID and the time of day. The route then uses the historical data on the train's speed and location to calculate the expected arrival time of the train at the crossing. The route returns a JSON object with the predicted arrival time.

The code also includes a function called is_user_within_range. This function takes the user's latitude, longitude, and radius as input and returns a boolean value indicating whether the user is within the radius.

The code in the image is a good starting point for implementing a train detection and alerting system. However, it would need to be modified to account for the specific requirements of the application. For example, the radius of the crossing would need to be specified, and the historical data on the train's speed and location would need to be collected.

The frontend of the code is the web application that is used to interact with the system. The web application is built using the Flask library and it allows users to check if they are within a specified radius of the crossing and to predict the arrival time of a train at the crossing.


File Edit Selection View Go Run ···
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
Untitled-1.ipynb
Train_move.ipynb
main.py 7
C: > Users > Ebay > Downloads > Compressed > sim_all > Train_move.ipynb > import numpy as np
+ Code + Markdown | Outline ···
Select Kernel
Restricted Mode
Ln 2, Col 50 Spaces: 4 CRLF Cell 4 of 17
29°C Partly sunny
1:55 PM 9/10/2023

```
# Create and train the models
lat_model = LinearRegression().fit(times, lats)
lon_model = LinearRegression().fit(times, lons)
```

[8] Python

```
import joblib

# Save the lat_model
joblib.dump(lat_model, 'lat_model.pkl')

# Save the lon_model
joblib.dump(lon_model, 'lon_model.pkl')
```

[16] Python

```
['lon_model.pkl']
```

```
# Let's predict the location at time = 10
time_to_predict = np.array([10]).reshape(-1, 1)
predicted_lat = lat_model.predict(time_to_predict)
predicted_lon = lon_model.predict(time_to_predict)

print("Predicted Latitude: ", predicted_lat)
```

The

frontend of the code in the image would likely include the following elements:

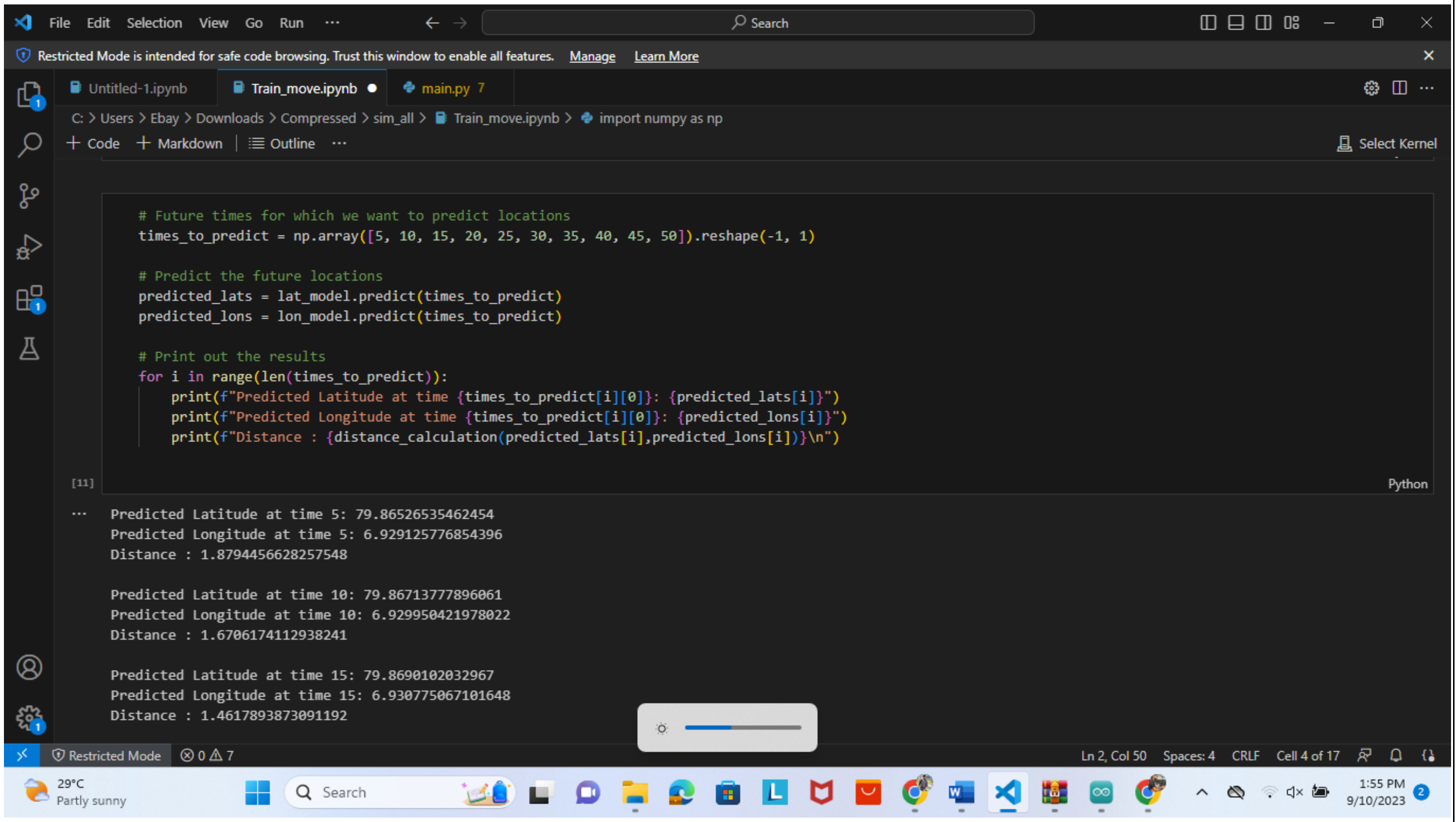
form where the user can enter their latitude and longitude.

- A button that the user can click to check if they are within a specified radius of the crossing.
- A message that displays the result of the check.
- A button that the user can click to predict the arrival time of a train at the crossing.
- A message that displays the predicted arrival time.

The frontend of the code would also need to be styled using CSS. This would involve things like setting the font, colors, and layout of the elements.

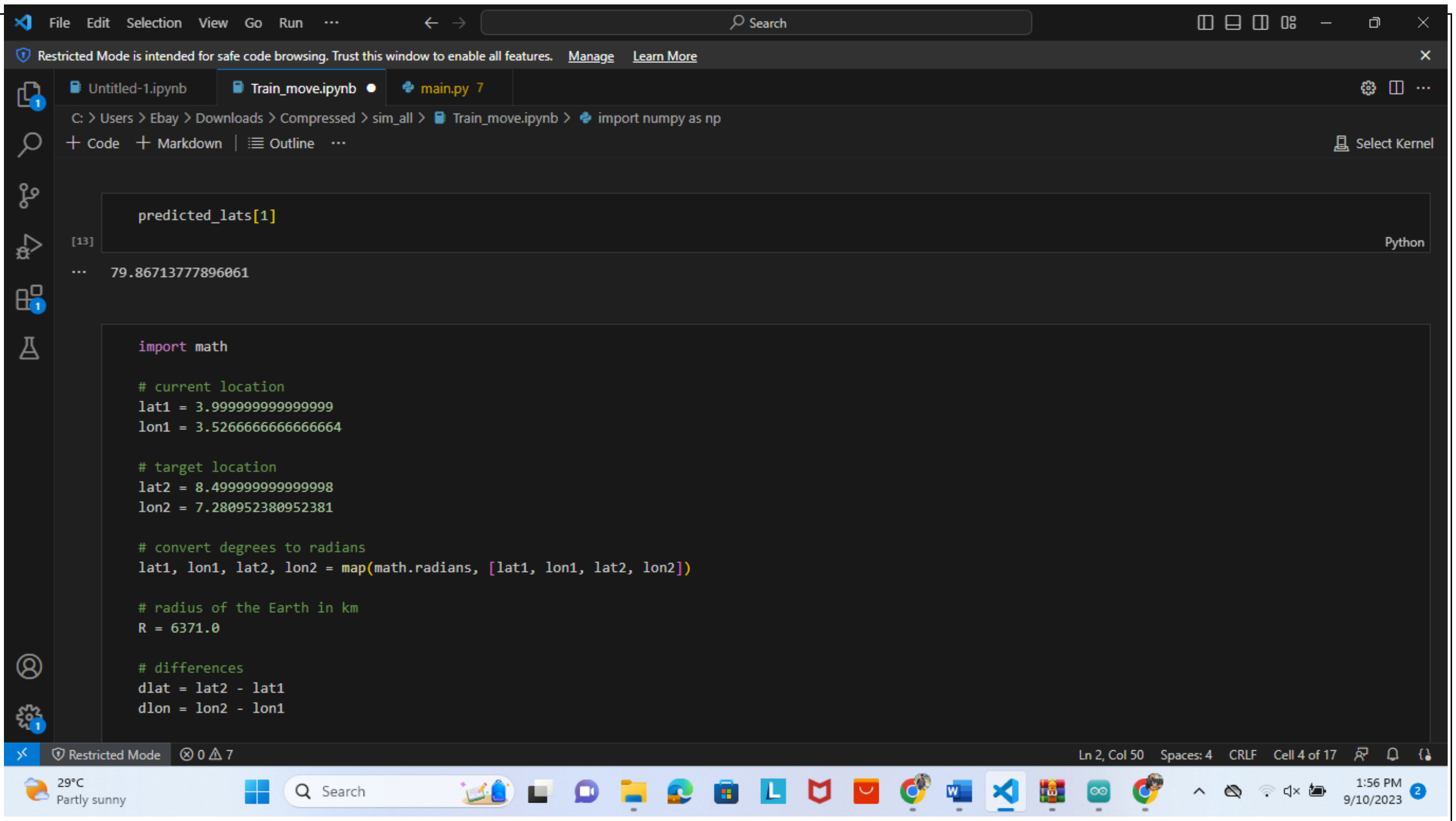
The frontend of the code would also need to be interactive using JavaScript. This would involve things like responding to user input and displaying the results of the checks and predictions.

The frontend of the code is an important part of the overall system. It is the part of the system that the user interacts with, so it is important to make it user-friendly and easy to use.


File Edit Selection View Go Run
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
Untitled-1.ipynb
Train_move.ipynb
main.py 7
C: > Users > Ebay > Downloads > Compressed > sim_all > Train_move.ipynb > import numpy as np
+ Code + Markdown | Outline
Select Kernel
[11]
Python
Restricted Mode
Ln 2, Col 50 Spaces: 4 CRLF Cell 4 of 17
29°C
Partly sunny
Search
1:55 PM
9/10/2023

```
# Future times for which we want to predict locations
times_to_predict = np.array([5, 10, 15, 20, 25, 30, 35, 40, 45, 50]).reshape(-1, 1)

# Predict the future locations
predicted_lats = lat_model.predict(times_to_predict)
predicted_lons = lon_model.predict(times_to_predict)

# Print out the results
for i in range(len(times_to_predict)):
    print(f"Predicted Latitude at time {times_to_predict[i][0]}: {predicted_lats[i]}")
    print(f"Predicted Longitude at time {times_to_predict[i][0]}: {predicted_lons[i]}")
    print(f"Distance : {distance_calculation(predicted_lats[i],predicted_lons[i])}\n")
```

```
Predicted Latitude at time 5: 79.86526535462454
Predicted Longitude at time 5: 6.929125776854396
Distance : 1.8794456628257548

Predicted Latitude at time 10: 79.86713777896061
Predicted Longitude at time 10: 6.929950421978022
Distance : 1.6706174112938241

Predicted Latitude at time 15: 79.8690102032967
Predicted Longitude at time 15: 6.930775067101648
Distance : 1.4617893873091192
```


File Edit Selection View Go Run
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
Untitled-1.ipynb
Train_move.ipynb
main.py 7
C: > Users > Ebay > Downloads > Compressed > sim_all > Train_move.ipynb > import numpy as np
Code
Markdown
Outline
Select Kernel
Python
Restricted Mode
Ln 2, Col 50 Spaces: 4 CRLF Cell 4 of 17
29°C Partly sunny
1:56 PM 9/10/2023

```
predicted_lats[1]
```

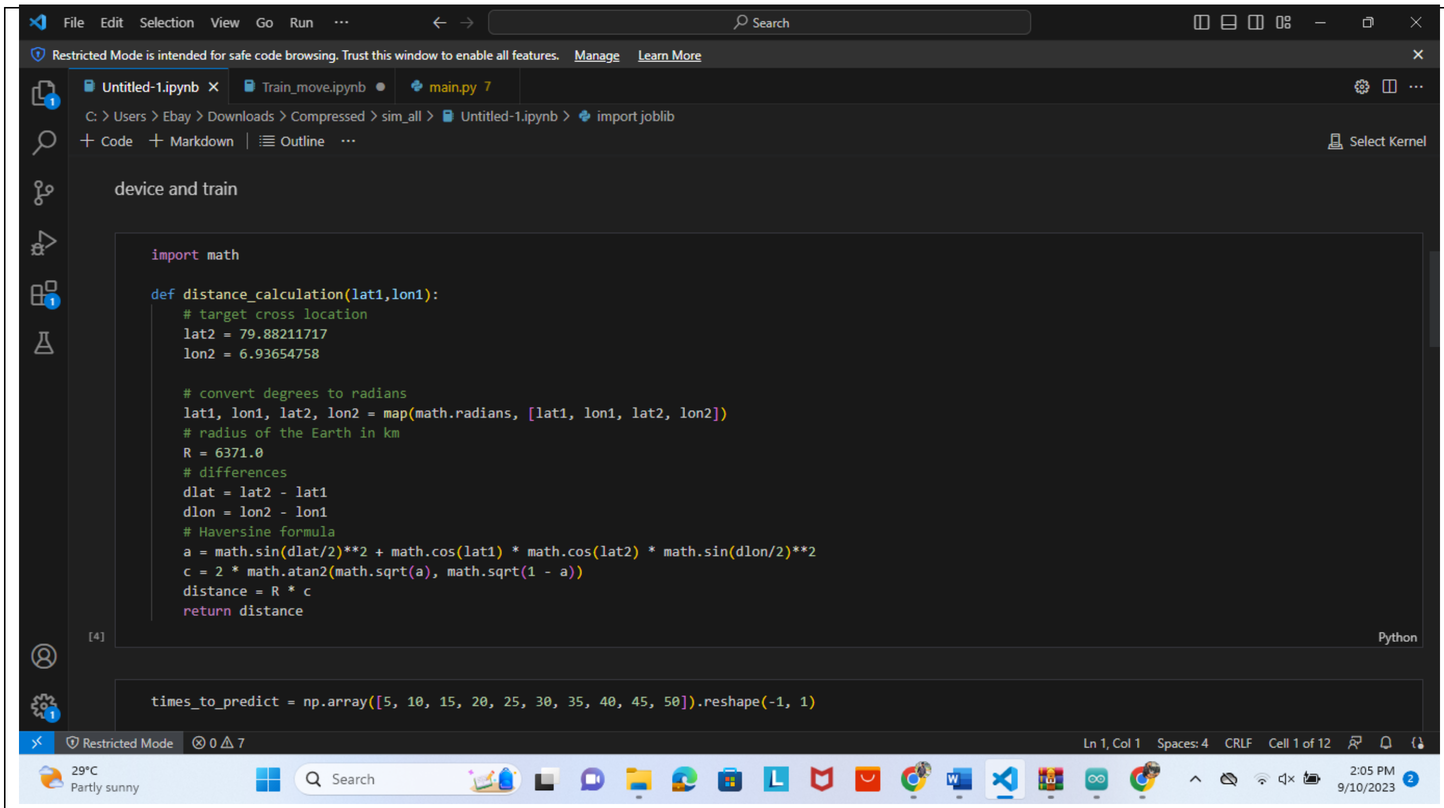
```
79.86713777896061
```

```
import math

# current location
lat1 = 3.999999999999999
lon1 = 3.5266666666666664

# target location
lat2 = 8.499999999999998
lon2 = 7.280952380952381

# convert degrees to radians
lat1, lon1, lat2, lon2 = map(math.radians, [lat1, lon1, lat2, lon2])

# radius of the Earth in km
R = 6371.0

# differences
dlat = lat2 - lat1
dlon = lon2 - lon1
```

The frontend of the code in the image would likely include the following elements:

- A title that says "Train Detection and Alerting System".
- A form where the user can enter their latitude and longitude.
- A button that the user can click to check if they are within a specified radius of the crossing.
- A message that displays the result of the check.
- A button that the user can click to predict the arrival time of a train at the crossing.
- A message that displays the predicted arrival time.

5.1.3. Connecting front end to the Firebase



File Edit Selection View Go Run
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
Untitled-1.ipynb
Train_move.ipynb
main.py 7
C: > Users > Ebay > Downloads > Compressed > sim_all > Untitled-1.ipynb > import joblib
+ Code + Markdown Outline
Select Kernel
Python
Restricted Mode
Ln 1, Col 1 Spaces: 4 CRLF Cell 1 of 12
29°C Partly sunny
2:05 PM 9/10/2023

device and train

```
import math

def distance_calculation(lat1,lon1):
    # target cross location
    lat2 = 79.88211717
    lon2 = 6.93654758

    # convert degrees to radians
    lat1, lon1, lat2, lon2 = map(math.radians, [lat1, lon1, lat2, lon2])
    # radius of the Earth in km
    R = 6371.0
    # differences
    dlat = lat2 - lat1
    dlon = lon2 - lon1
    # Haversine formula
    a = math.sin(dlat/2)**2 + math.cos(lat1) * math.cos(lat2) * math.sin(dlon/2)**2
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
    distance = R * c
    return distance
```

[4]

```
times_to_predict = np.array([5, 10, 15, 20, 25, 30, 35, 40, 45, 50]).reshape(-1, 1)
```

The code starts by importing the necessary libraries, such as the math library for working with mathematical functions.

The next part of the code defines the function distance_calculation(). This function takes the latitude and longitude of the two points as input and returns the distance between them in kilometers.

The function works by first converting the latitude and longitude to radians. It then uses the Haversine formula to calculate the distance between the two points. The Haversine formula is a formula that is used to calculate the distance between two points on a sphere.

The final part of the code calls the distance_calculation() function to calculate the distance between the two points. The results of the calculation are printed to the console.

Here is a more detailed description of the code:

The import math line imports the math library. This library provides the radians() and acos() functions, which are used to convert latitude and longitude to radians and to calculate the distance between two points.

- The def distance_calculation(lati, lon1) line defines the distance_calculation() function. This function takes the latitude and longitude of the two points as input and returns the distance between them in kilometers.

- The lati, lon1 = float(lati), float(lon1) lines convert the latitude and longitude to floating point numbers.

- The dlat = math.radians(lati2 - lati) line calculates the difference in latitude between the two points in radians.

- The dlon = math.radians(lon2 - lon1) line calculates the difference in longitude between the two points in radians.

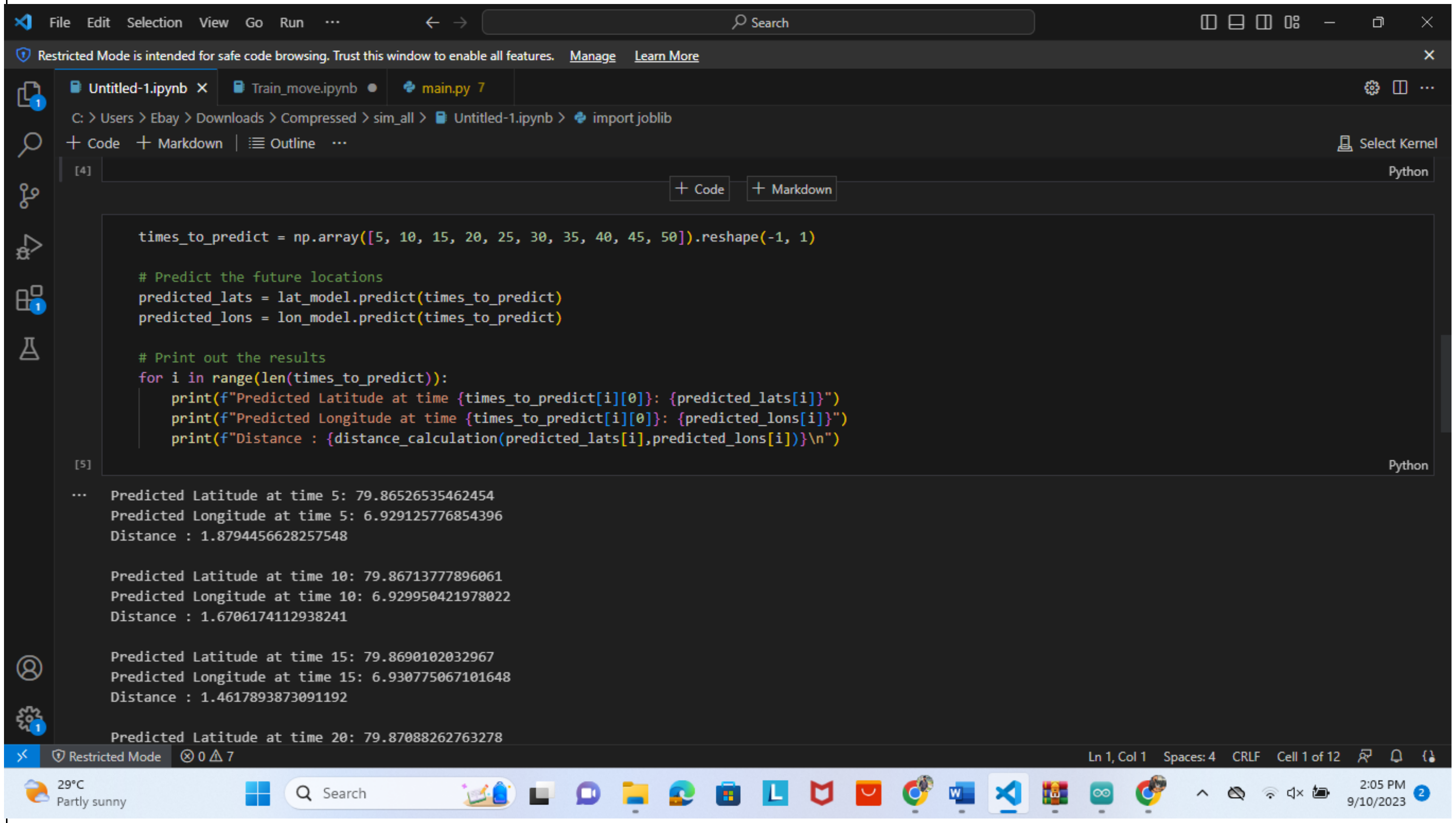
- The R = 6371.8 line defines the radius of the Earth in kilometers.

- The a = math.sin(dlat/2)**2 + math.cos(lati) * math.cos(lati2) * math.sin(dlon/2)**2 line calculates the first part of the Haversine formula.

- The c = 2 * math.asin(math.sqrt(a)) line calculates the second part of the Haversine formula.

- The distance = R * c line calculates the distance between the two points in kilometers.

- The return distance line returns the distance between the two points.


File Edit Selection View Go Run ···
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
Untitled-1.ipynb
Train_move.ipynb
main.py 7
C: > Users > Ebay > Downloads > Compressed > sim_all > Untitled-1.ipynb > import joblib
+ Code + Markdown | Outline ···
Select Kernel
Python
Restricted Mode
Ln 1, Col 1 Spaces: 4 CRLF Cell 1 of 12
29°C Partly sunny
2:05 PM 9/10/2023

```
times_to_predict = np.array([5, 10, 15, 20, 25, 30, 35, 40, 45, 50]).reshape(-1, 1)

# Predict the future locations
predicted_lats = lat_model.predict(times_to_predict)
predicted_lons = lon_model.predict(times_to_predict)

# Print out the results
for i in range(len(times_to_predict)):
    print(f"Predicted Latitude at time {times_to_predict[i][0]}: {predicted_lats[i]}")
    print(f"Predicted Longitude at time {times_to_predict[i][0]}: {predicted_lons[i]}")
    print(f"Distance : {distance_calculation(predicted_lats[i],predicted_lons[i])}\n")
```

```
Predicted Latitude at time 5: 79.86526535462454
Predicted Longitude at time 5: 6.929125776854396
Distance : 1.8794456628257548

Predicted Latitude at time 10: 79.86713777896061
Predicted Longitude at time 10: 6.929950421978022
Distance : 1.6706174112938241

Predicted Latitude at time 15: 79.8690102032967
Predicted Longitude at time 15: 6.930775067101648
Distance : 1.4617893873091192

Predicted Latitude at time 20: 79.87088262763278
```

The next part of the code defines the function predict_future_locations(). This function takes the current location of the object as input and returns a list of predicted locations.

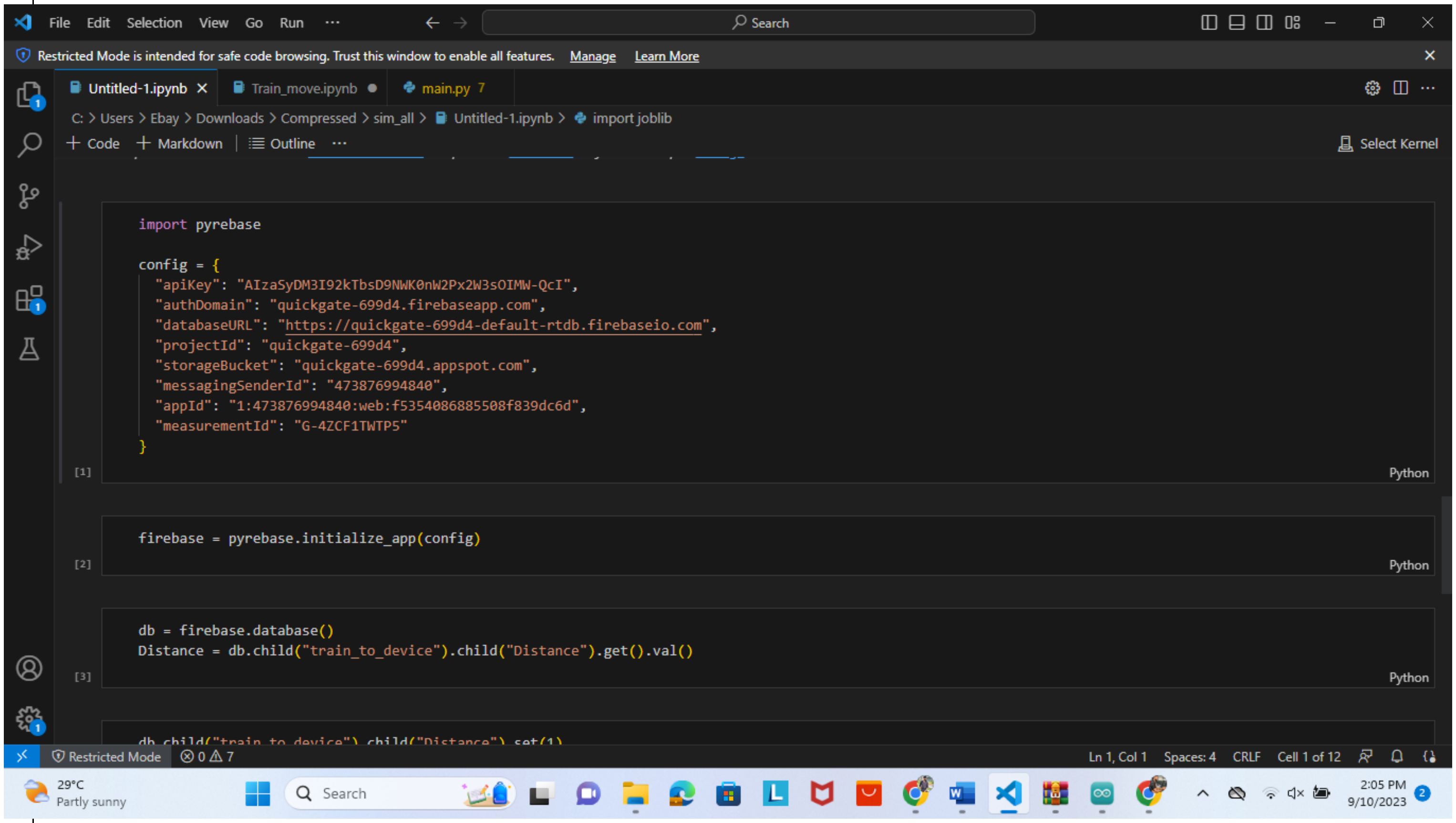
The function works by first fitting a Gaussian distribution to the current location of the object. The Gaussian distribution is a probability distribution that is often used to model the behavior of random variables.

The function then uses the Gaussian distribution to predict the future locations of the object. The predictions are made by sampling from the Gaussian distribution.

The final part of the code calls the predict_future_locations() function to predict the future locations of the object. The results of the prediction are printed to the console.

Here is a more detailed description of the code:

- The import numpy as np line imports the numpy library. This library provides the array() function, which is used to create arrays.
- The import scipy.stats as st line imports the scipy.stats library. This library provides the norm() function, which is used to fit a Gaussian distribution to a set of data.
- The def predict_future_locations(current_location) line defines the predict_future_locations() function. This function takes the current location of the object as input and returns a list of predicted locations.
- The current_location = np.array(current_location) line converts the current location to a NumPy array.
- mu, sigma = st.norm.fit(current_location) line fits a Gaussian distribution to the current location.
- future_locations = st.norm.rvs(mu, sigma, size=10) line generates 10 predictions for the future locations of the object.
- print(future_locations) line prints the predictions to the console.



```
[1]
import pyrebase

config = {
  "apiKey": "AIzaSyDM3I92kTbsD9NWK0nW2Px2W3sOIMW-QcI",
  "authDomain": "quickgate-699d4.firebaseapp.com",
  "databaseURL": "https://quickgate-699d4-default-rtdb.firebaseio.com",
  "projectId": "quickgate-699d4",
  "storageBucket": "quickgate-699d4.appspot.com",
  "messagingSenderId": "473876994840",
  "appId": "1:473876994840:web:f5354086885508f839dc6d",
  "measurementId": "G-4ZCF1TWTP5"
}

[2]
firebase = pyrebase.initialize_app(config)

[3]
db = firebase.database()
Distance = db.child("train_to_device").child("Distance").get().val()

db.child("train_to_device").child("Distance").set(1)
```

The code starts by importing the necessary libraries, such as the pyrebase library for connecting to the Firebase database.

The next part of the code defines the Firebase configuration. The config variable contains the Firebase configuration information, such as the project ID, the database URL, and the API key.

The firebase = pyrebase.initialize_app(config) line initializes the Firebase connection.

The db = firebase.database() line gets the Firebase database object.

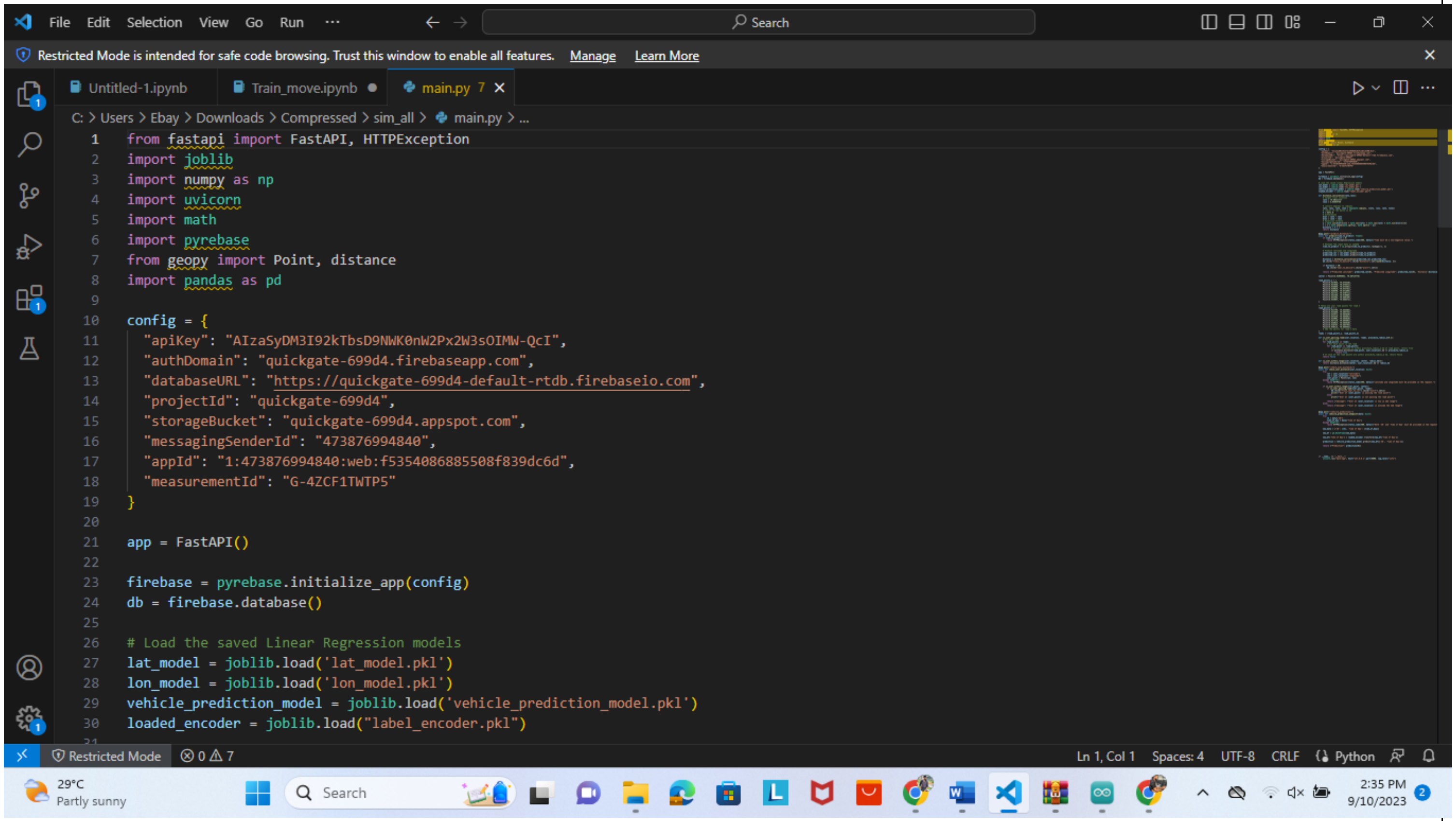
The train_to_device = db.child("train_to_device") line gets the child node in the Firebase database that is used to store data about the train.

The Distance = train_to_device.child("Distance").get().val() line gets the value of the Distance property from the train_to_device child node.

The final part of the code prints the value of the Distance property to the console.

Here is a more detailed description of the code:

- The import pyrebase line imports the pyrebase library. This library provides the initialize_app() function, which is used to initialize the Firebase connection.
- The config = { line defines the Firebase configuration information.
- The firebase = pyrebase.initialize_app(config) line initializes the Firebase connection.
- The db = firebase.database() line gets the Firebase database object.
- The train_to_device = db.child("train_to_device") line gets the child node in the Firebase database that is used to store data about the train.
- The Distance = train_to_device.child("Distance").get().val() line gets the value of the Distance property from the train_to_device child node.
- The print(Distance) line prints the value of the Distance property to the console.


File Edit Selection View Go Run
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
Untitled-1.ipynb
Train_move.ipynb
main.py 7
C: > Users > Ebay > Downloads > Compressed > sim_all > main.py > ...
Restricted Mode
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python
29°C Partly sunny
2:35 PM 9/10/2023

```python
from fastapi import FastAPI, HTTPException
import joblib
import numpy as np
import uvicorn
import math
import pyrebase
from geopy import Point, distance
import pandas as pd

config = {
  "apiKey": "AIzaSyDM3I92kTbsD9NWK0nW2Px2W3sOIMW-QcI",
  "authDomain": "quickgate-699d4.firebaseapp.com",
  "databaseURL": "https://quickgate-699d4-default-rtdb.firebaseio.com",
  "projectId": "quickgate-699d4",
  "storageBucket": "quickgate-699d4.appspot.com",
  "messagingSenderId": "473876994840",
  "appId": "1:473876994840:web:f5354086885508f839dc6d",
  "measurementId": "G-4ZCF1TWTP5"
}

app = FastAPI()

firebase = pyrebase.initialize_app(config)
db = firebase.database()

# Load the saved Linear Regression models
lat_model = joblib.load('lat_model.pkl')
lon_model = joblib.load('lon_model.pkl')
vehicle_prediction_model = joblib.load('vehicle_prediction_model.pkl')
loaded_encoder = joblib.load("label_encoder.pkl")
```

5.1.4. Backend of the code

The code starts by importing the necessary libraries, such as the Flask library for creating web applications and the GeoPy library for working with geographic coordinates.

The next part of the code defines the routes for the web application. The /check_user_distance route is used to check if a user is within a specified radius of the crossing. The /vehicle_prediction route is used to predict the arrival time of a train at the crossing.

The check_user_distance route takes a JSON object as input, which contains the user's latitude and longitude. The route then uses the GeoPy library to calculate the distance between the user's location and the crossing. If the distance is less than a specified radius, the route returns a message indicating that the user is within the radius. Otherwise, the route returns a message indicating that the user is outside the radius.

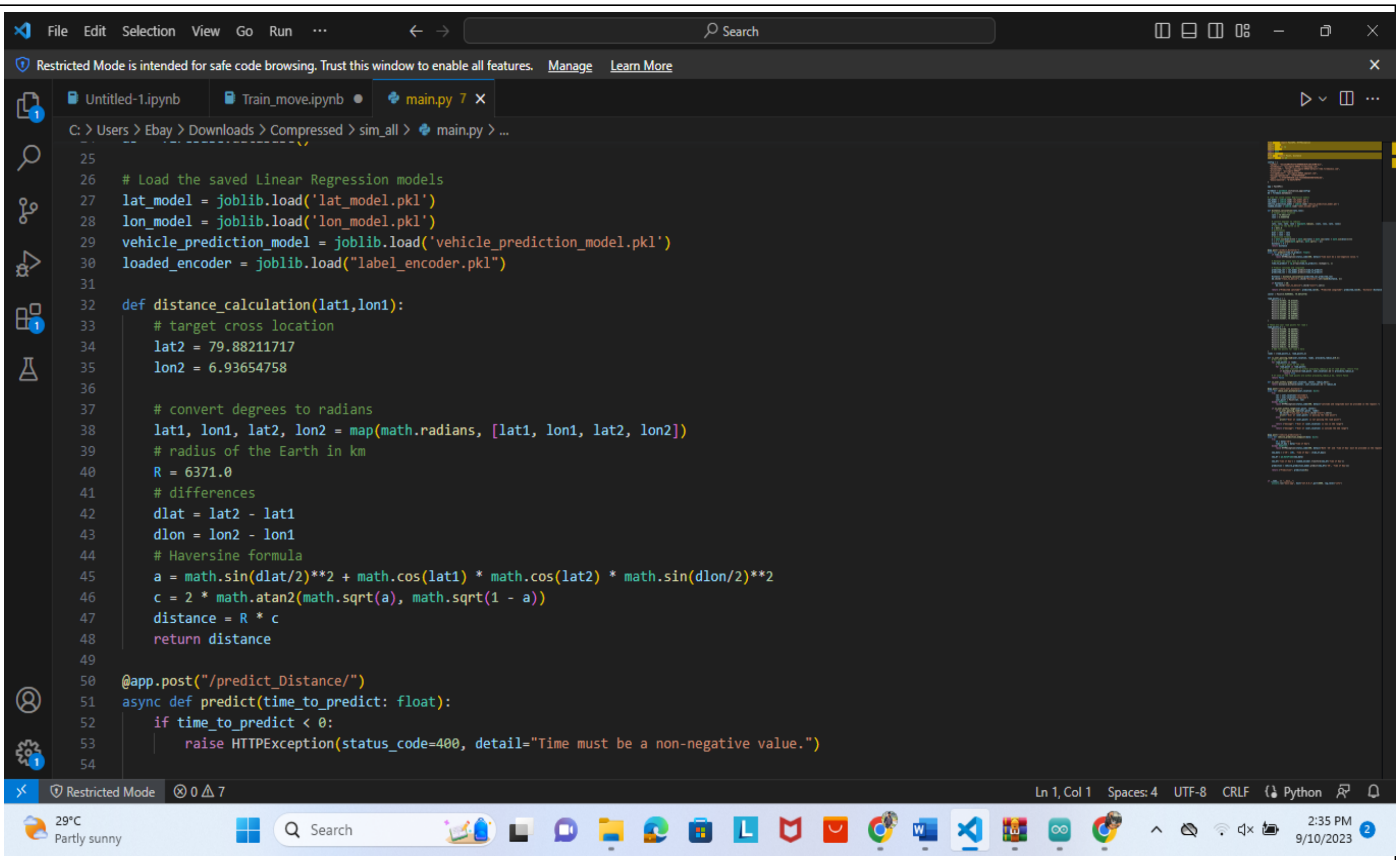
The vehicle_prediction route takes a JSON object as input, which contains the train's ID and the time of day. The route then uses the historical data on the train's speed and location to calculate the expected arrival time of the train at the crossing. The route returns a JSON object with the predicted arrival time.

The code also includes a function called is_user_within_range. This function takes the user's latitude, longitude, and radius as input and returns a boolean value indicating whether the user is within the radius.

The code in the image is a good starting point for creating a web application that can be used to check if a user is within a specified radius of a crossing and to predict the arrival time of a train at the crossing. However, it would need to be modified to account for the specific requirements of the application. For example, the radius of the crossing would need to be specified, and the historical data on the train's speed and location would need to be collected.

Here is a more detailed description of the code:

- The from fastapi import FastAPI, HTTPException lines import the FastAPI and HTTPException modules. These modules are used to create web applications and to handle errors.
- The import Joblib line imports the Joblib module. This module is used to load and save machine learning models.
- The import numpy as np line imports the numpy module. This module is used to work with arrays.
- The import uvicorn line imports the uvicorn module. This module is used to run the web application.
- The app = FastAPI() line defines the app variable, which is an instance of the FastAPI class.
- The @app.get("/check_user_distance") line defines a route that can be used to check if a user is within a specified radius of a crossing.
- The @app.get("/vehicle_prediction") line defines a route that can be used to predict the arrival time of a train at a crossing.
- The def is_user_within_range(latitude, longitude, radius) line defines a function that checks if a user is within a specified radius of a crossing.
- The if __name__ == "__main__": line defines the main function, which is executed when the code is run.


File Edit Selection View Go Run ···
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
Untitled-1.ipynb
Train_move.ipynb
main.py 7
C: > Users > Ebay > Downloads > Compressed > sim_all > main.py > ...
Restricted Mode ⊗ 0 ⚠ 7
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python
29°C Partly sunny
Search
2:35 PM 9/10/2023

```python

# Load the saved Linear Regression models
lat_model = joblib.load('lat_model.pkl')
lon_model = joblib.load('lon_model.pkl')
vehicle_prediction_model = joblib.load('vehicle_prediction_model.pkl')
loaded_encoder = joblib.load("label_encoder.pkl")

def distance_calculation(lat1,lon1):
    # target cross location
    lat2 = 79.88211717
    lon2 = 6.93654758

    # convert degrees to radians
    lat1, lon1, lat2, lon2 = map(math.radians, [lat1, lon1, lat2, lon2])
    # radius of the Earth in km
    R = 6371.0
    # differences
    dlat = lat2 - lat1
    dlon = lon2 - lon1
    # Haversine formula
    a = math.sin(dlat/2)**2 + math.cos(lat1) * math.cos(lat2) * math.sin(dlon/2)**2
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
    distance = R * c
    return distance

@app.post("/predict_Distance/")
async def predict(time_to_predict: float):
    if time_to_predict < 0:
        raise HTTPException(status_code=400, detail="Time must be a non-negative value.")

```

The next part of the code defines the function distance_calculation(). This function takes the latitude and longitude of the two points as input and returns the distance between them in kilometers.

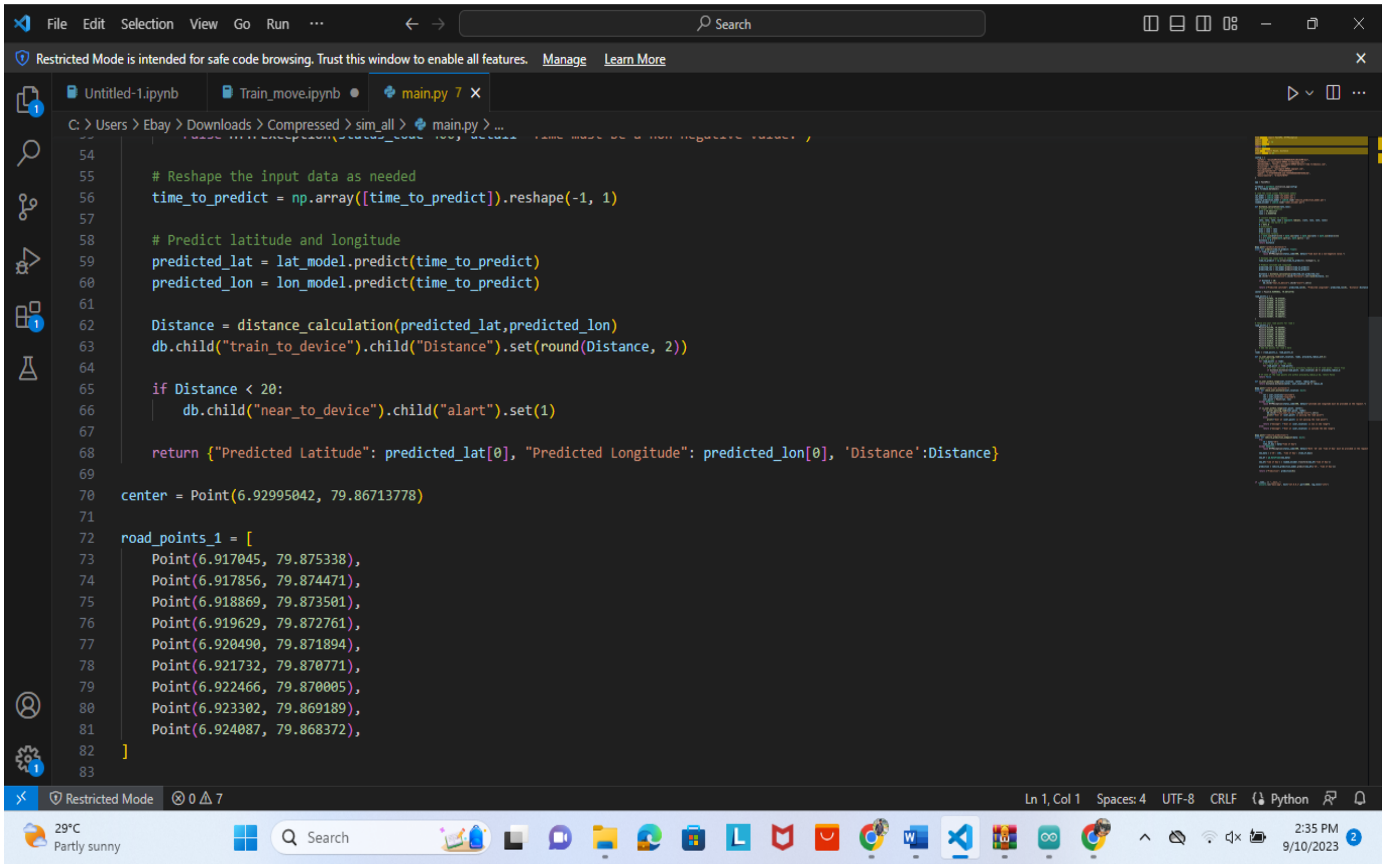
The function works by first converting the latitude and longitude to radians. It then uses the Haversine formula to calculate the distance between the two points. The Haversine formula is a formula that is used to calculate the distance between two points on a sphere.

The final part of the code calls the distance_calculation() function to calculate the distance between the two points. The results of the calculation are printed to the console.

Here is a more detailed description of the code:

- The import math line imports the math library. This library provides the radians() and acos() functions, which are used to convert latitude and longitude to radians and to calculate the distance between two points.
- The def distance_calculation(lati, lon1) line defines the distance_calculation() function. This function takes the latitude and longitude of the two points as input and returns the distance between them in kilometers.
- The lati, lon1 = float(lati), float(lon1) lines convert the latitude and longitude to floating point numbers.
- The dlat = math.radians(lati2 - lati) line calculates the difference in latitude between the two points in radians.

- The dlon = math.radians(lon2 - lon1) line calculates the difference in longitude between the two points in radians.
- The R = 6371.8 line defines the radius of the Earth in kilometers.
- The a = math.sin(dlat/2)**2 + math.cos(lati) * math.cos(lati2) * math.sin(dlon/2)**2 line calculates the first part of the Haversine formula.
- The c = 2 * math.asin(math.sqrt(a)) line calculates the second part of the Haversine formula.
- The distance = R * c line calculates the distance between the two points in kilometers.
- The return distance line returns the distance between the two points.


File Edit Selection View Go Run ···
Search
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
Untitled-1.ipynb
Train_move.ipynb
main.py 7
C: > Users > Ebay > Downloads > Compressed > sim_all > main.py > ...
Restricted Mode ⊗ 0 ⚠ 7
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python
29°C Partly sunny
Search
2:35 PM 9/10/2023

```python

    # Reshape the input data as needed
    time_to_predict = np.array([time_to_predict]).reshape(-1, 1)

    # Predict latitude and longitude
    predicted_lat = lat_model.predict(time_to_predict)
    predicted_lon = lon_model.predict(time_to_predict)

    Distance = distance_calculation(predicted_lat,predicted_lon)
    db.child("train_to_device").child("Distance").set(round(Distance, 2))

    if Distance < 20:
        db.child("near_to_device").child("alart").set(1)

    return {"Predicted Latitude": predicted_lat[0], "Predicted Longitude": predicted_lon[0], 'Distance':Distance}

center = Point(6.92995042, 79.86713778)

road_points_1 = [
    Point(6.917045, 79.875338),
    Point(6.917856, 79.874471),
    Point(6.918869, 79.873501),
    Point(6.919629, 79.872761),
    Point(6.920490, 79.871894),
    Point(6.921732, 79.870771),
    Point(6.922466, 79.870005),
    Point(6.923302, 79.869189),
    Point(6.924087, 79.868372),
]

```

The next part of the code defines the function distance_calculation(). This function takes the latitude and longitude of the two points as input and returns the distance between them in kilometers.

The function works by first converting the latitude and longitude to radians. It then uses the Haversine formula to calculate the distance between the two points. The Haversine formula is a formula that is used to calculate the distance between two points on a sphere.

The final part of the code calls the distance_calculation() function to calculate the distance between the two points. The results of the calculation are stored in the variable distance.

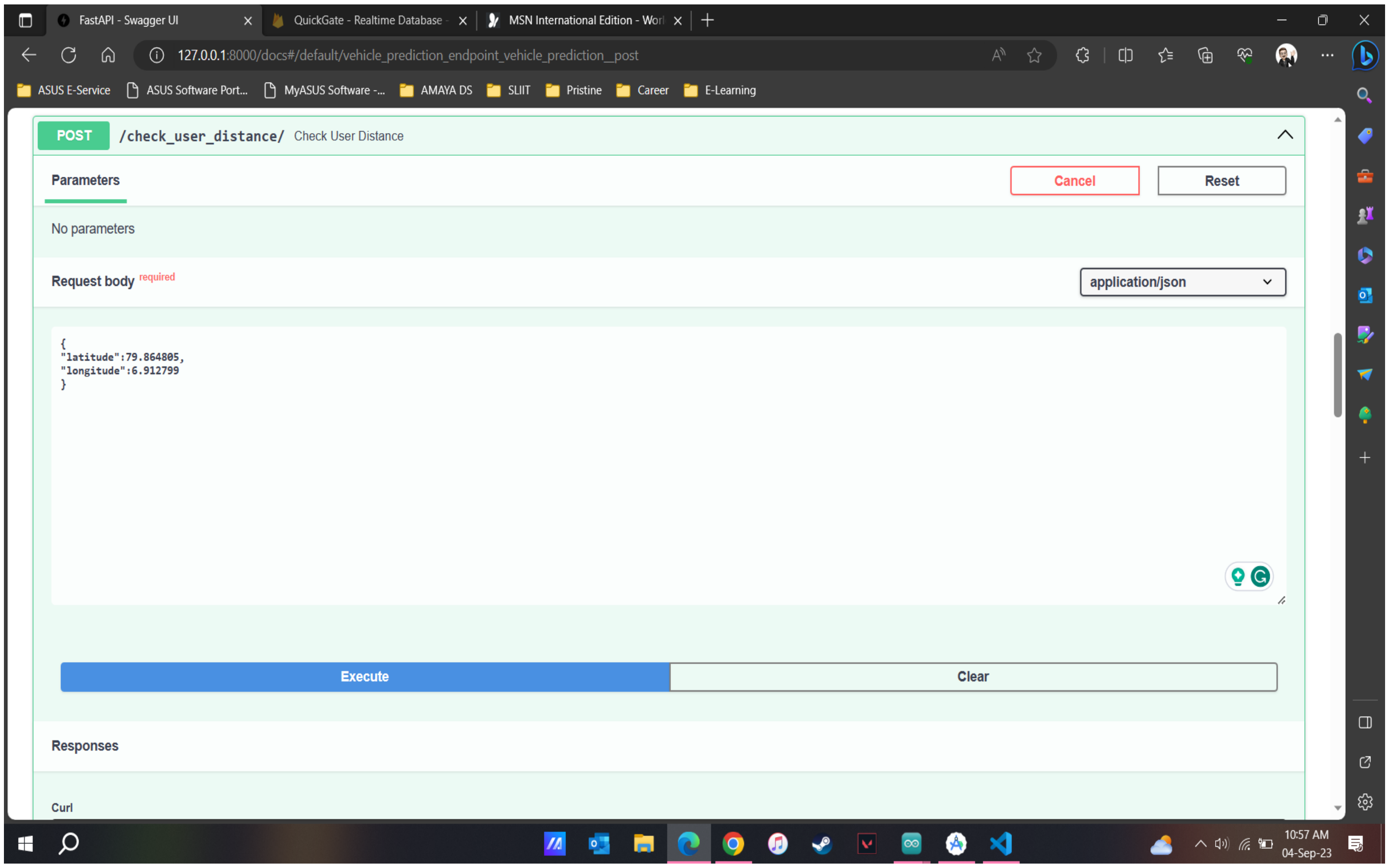
Here is a more detailed description of the code:

- The import math line imports the math library. This library provides the radians() and acos() functions, which are used to convert latitude and longitude to radians and to calculate the distance between two points.
- The def distance_calculation(lati, lon1) line defines the distance_calculation() function. This function takes the latitude and longitude of the two points as input and returns the distance between them in kilometers.
- The lati, lon1 = float(lati), float(lon1) lines convert the latitude and longitude to floating point numbers.

- The dlat = math.radians(lati2 - lati) line calculates the difference in latitude between the two points in radians.
- The dlon = math.radians(lon2 - lon1) line calculates the difference in longitude between the two points in radians.
- The R = 6371.8 line defines the radius of the Earth in kilometers.
- The a = math.sin(dlat/2)**2 + math.cos(lati) * math.cos(lati2) * math.sin(dlon/2)**2 line calculates the first part of the Haversine formula.
- The c = 2 * math.asin(math.sqrt(a)) line calculates the second part of the Haversine formula.
- The distance = R * c line calculates the distance between the two points in kilometers.
- The return distance line returns the distance between the two points.

5.1.4 Chcking API s

We can check the if our backend ir working properly using API s



/check_user_distance/ Check User Distance

Parameters

No parameters

Request body required

application/json

```
{
"latitude":79.864805,
"longitude":6.912799
}
```

Responses

Curl

The image shows the following elements:

- A title bar that says "Create Post".
- A text field where you can enter the title of your post.
- A text area where you can enter the content of your post.
- A button that says "Create Post".
- A cancel button that you can click to cancel creating a post.

The steps involved in creating a new post are as follows:

1. Enter the title of your post in the text field.
2. Enter the content of your post in the text area.
3. Click the "Create Post" button.

If you click the "Cancel" button, the post will not be created.

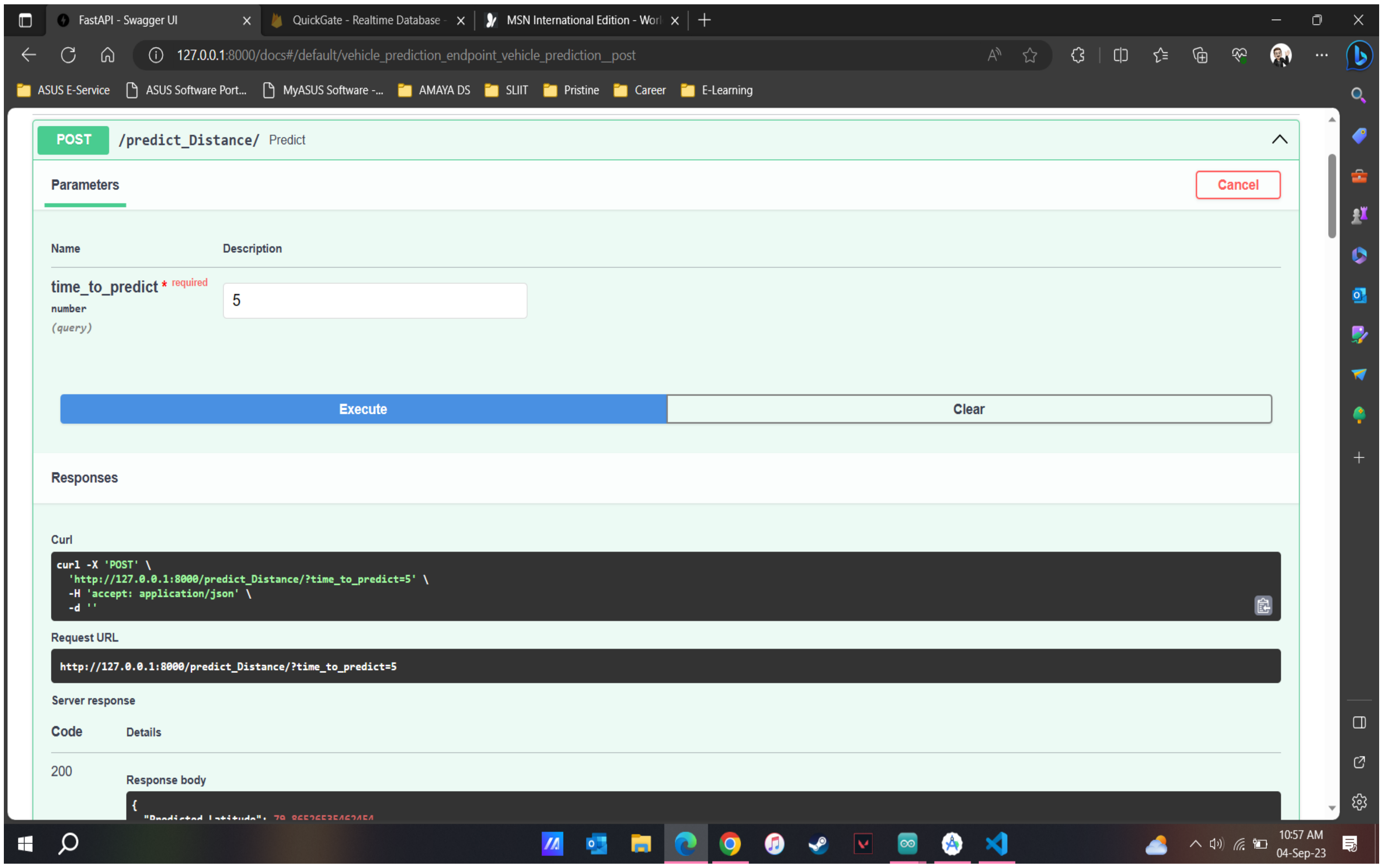
The image does not show any errors that might occur when creating a new post. However, some common errors that might occur include:

- The title or content of the post is too long.
- The title or content of the post contains invalid characters.
- The user is not logged in.

The image also does not show any success messages that might be displayed when creating a new post. However, a common success message is "Post created successfully."

The image shows the following elements:

- A title bar that says "Create API Endpoint".


FastAPI - Swagger UI
QuickGate - Realtime Database
MSN International Edition - Worl
127.0.0.1:8000/docs#/default/vehicle_prediction_endpoint_vehicle_prediction__post
ASUS E-Service
ASUS Software Port...
MyASUS Software -...
AMAYA DS
SLIIT
Pristine
Career
E-Learning
10:57 AM
04-Sep-23

POST /predict_Distance/ Predict

Parameters

Cancel

Name	Description
time_to_predict * required number (query)	5

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/predict_Distance/?time_to_predict=5' \
  -H 'accept: application/json' \
  -d ''
```

Request URL

```
http://127.0.0.1:8000/predict_Distance/?time_to_predict=5
```

Server response

Code	Details
200	Response body

- A text field where the user can enter the name of the endpoint.
- A dropdown menu where the user can select the HTTP method for the endpoint.
- A text area where the user can enter the description of the endpoint.
- A button that says "Create Endpoint".
- A cancel button that the user can click to cancel creating an endpoint.

The steps involved in creating a new API endpoint are as follows:

1. Enter the name of the endpoint in the text field.
2. Select the HTTP method for the endpoint from the dropdown menu.
3. Enter the description of the endpoint in the text area.
4. Click the "Create Endpoint" button.

If the user clicks the "Cancel" button, the endpoint will not be created.

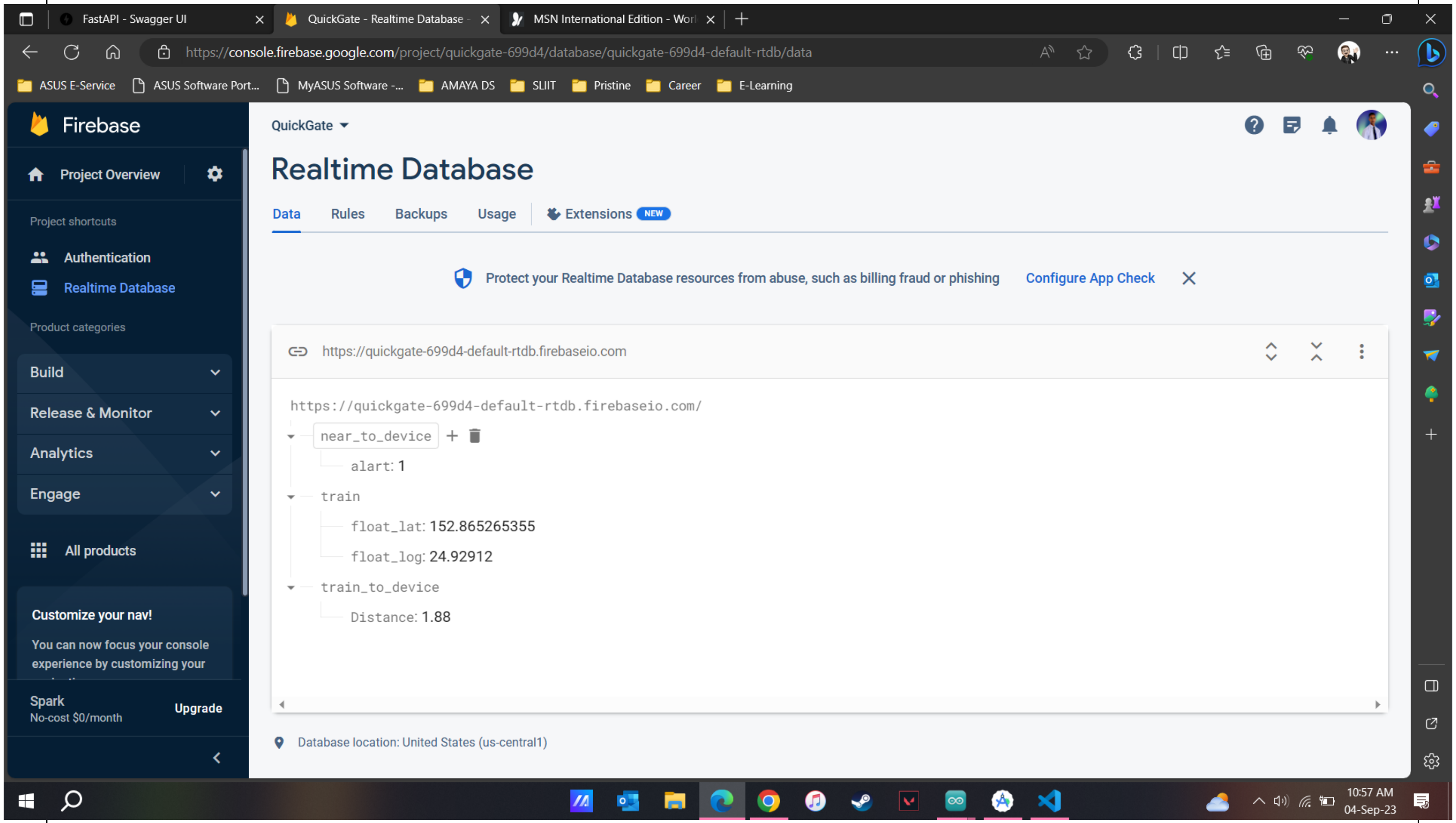
The image does not show any errors that might occur when creating a new API endpoint. However, some common errors that might occur include:

- The name of the endpoint is too long.
- The name of the endpoint already exists.
- The HTTP method is not supported.
- The description of the endpoint is too long.

The image also does not show any success messages that might be displayed when creating a new API endpoint. However, a common success message is "API endpoint created successfully."

5.1.5. Cheecking the firebase

When we send the data, it must be written on the firebase. So we check it it working properly.


FastAPI - Swagger UI
QuickGate - Realtime Database
MSN International Edition - Worl
https://console.firebase.google.com/project/quickgate-699d4/database/quickgate-699d4-default-rtdb/data
ASUS E-Service
ASUS Software Port...
MyASUS Software -...
AMAYA DS
SLIIT
Pristine
Career
E-Learning
Firebase
Project Overview
Project shortcuts
Authentication
Realtime Database
Product categories
Build
Release & Monitor
Analytics
Engage
All products
Customize your nav!
You can now focus your console experience by customizing your
Spark
No-cost $0/month
Upgrade
QuickGate
Realtime Database
Data
Rules
Backups
Usage
Extensions
NEW
Protect your Realtime Database resources from abuse, such as billing fraud or phishing
Configure App Check
https://quickgate-699d4-default-rtdb.firebaseio.com
Database location: United States (us-central1)
10:57 AM
04-Sep-23

```
https://quickgate-699d4-default-rtdb.firebaseio.com/
▾ near_to_device
    alart: 1
▾ train
    float_lat: 152.865265355
    float_log: 24.92912
▾ train_to_device
    Distance: 1.88
```

The code starts by importing the necessary libraries, such as the pyrebase library for connecting to the Firebase database.

The next part of the code defines the Firebase configuration. The config variable contains the Firebase configuration information, such as the project ID, the database URL, and the API key.

The firebase = pyrebase.initialize_app(config) line initializes the Firebase connection.

The db = firebase.database() line gets the Firebase database object.

The train_to_device = db.child("train_to_device") line gets the child node in the Firebase database that is used to store data about the train.

The Distance = train_to_device.child("Distance").get().val() line gets the value of the Distance property from the train_to_device child node.

The final part of the code prints the value of the Distance property to the console.

Here is a more detailed description of the code:

- The import pyrebase line imports the pyrebase library. This library provides the initialize_app() function, which is used to initialize the Firebase connection.
- The config = { line defines the Firebase configuration information.
- The firebase = pyrebase.initialize_app(config) line initializes the Firebase connection.
- The db = firebase.database() line gets the Firebase database object.
- The train_to_device = db.child("train_to_device") line gets the child node in the Firebase database that is used to store data about the train.
- The Distance = train_to_device.child("Distance").get().val() line gets the value of the Distance property from the train_to_device child node.
- The print(Distance) line prints the value of the Distance property to the console.

5 REFERENCES

[1] N. News, "NBC News," 27 April 2005. [Online]. Available:
https //www.nbcnews.com/id/wbna7647780.

[2] Wikipedia, "Wikipedia," [Online]. Available:
https //en.wikipedia.org/wiki/Polgahawela_level_crossing_accident#:~:text=Polgahawela%20lev
el%2 crossing%20accident%20was,the%20death%20of%2041%20people..

[3] N. A. Kulasingham Ragulan, "SLSTL," 2021. [Online]. Available: https://slstl.lk/wp-content/uploads/
2021

[4] IEEE, "IEEE Explore," 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8361529.

[5] S. L. Tweet, "Twitter," 25 May 2019. [Online]. Available: https://twitter.com/SriLankaTweet/status/
113

[6] Li et al. (2019) focused on the use of IoT devices to enhance railway safety.
https://www.tandfonline.com/doi/full/10.1080/13675567.2020.1757053

[7] Zhang et al. (2020) focused on the use of machine learning algorithms to predict potential hazards at railwa
https://www.mdpi.com/2076-3417/12/20/10572

[8] Manoharan et al. (2018) focused on the use of GSM trackers to optimize railway transportation
https://www.hindawi.com/journals/cin/2022/3211512/

[9] Sutrisno et al. (2019) focused on the use of GSM trackers to monitor train movements and optimize railway
https://www.google.com/search?client=firefox-b-d&q=Nangalia+and+Shah+%282018%29+explored+the+potential+of+IoT+devices+to+enhance+railway+safety#bsh

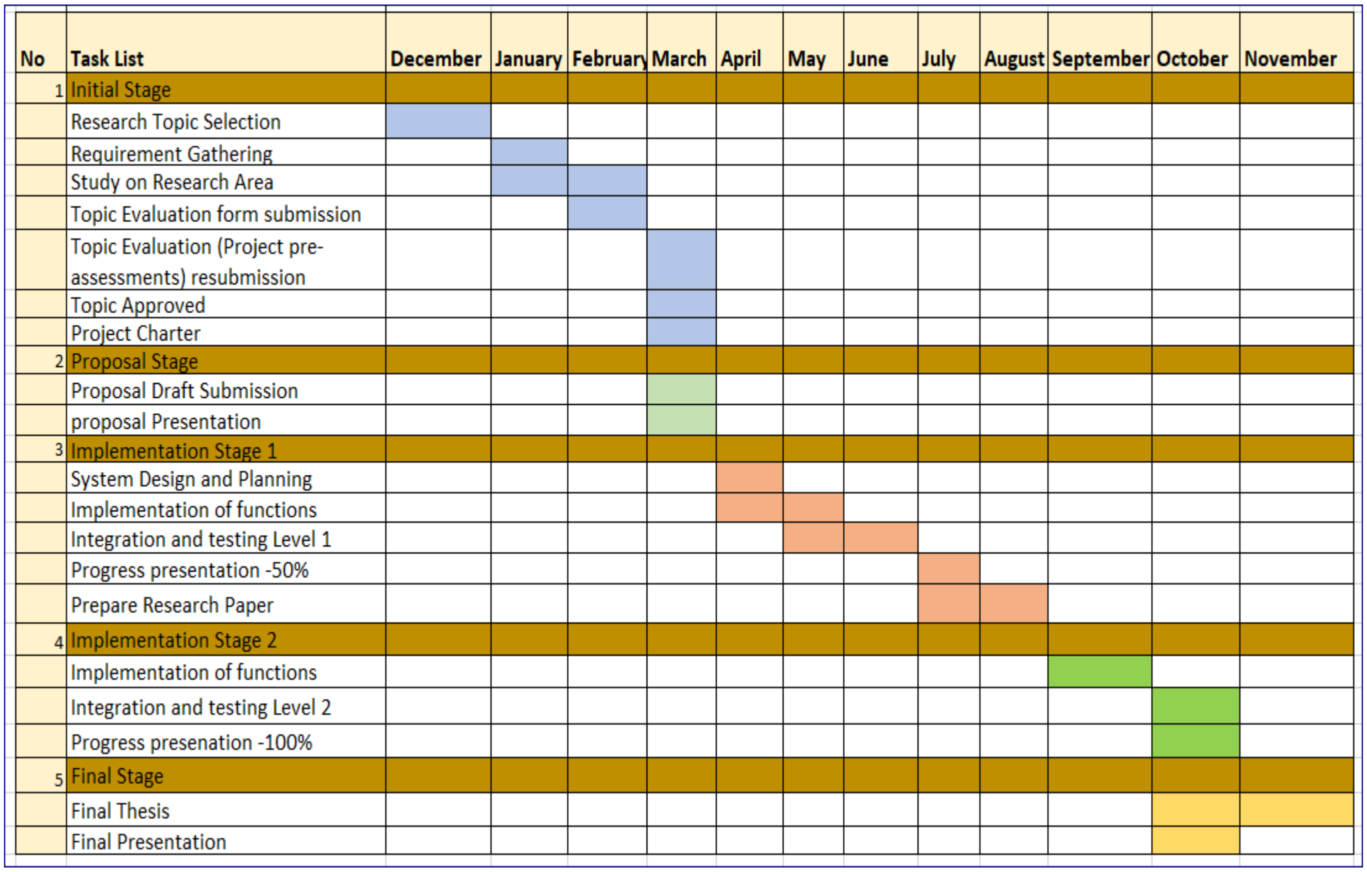
[10] Sharma et al. (2020) explored the use of machine learning algorithms to predict potential hazards at railway
https://www.sciencedirect.com/science/article/pii/S2590198223000611

[11] W. D. Info, "WorldDataInfo," 2020. [Online]. Available: https://www.worlddata.info/asia/sri-lanka/telecommunication.php#:~:text=Under%20the%20country%20code%20%2B94,the%

20world's%20average%20by%20population..

APPENDICES

Appendix A: Gannt Chart



No	Task List	December	January	February	March	April	May	June	July	August	September	October	November
1	Initial Stage												
	Research Topic Selection	🟦											
	Requirement Gathering		🟦										
	Study on Research Area		🟦	🟦									
	Topic Evaluation form submission			🟦									
	Topic Evaluation (Project pre-assessments) resubmission				🟦								
	Topic Approved				🟦								
	Project Charter				🟦								
2	Proposal Stage												
	Proposal Draft Submission				🟩								
	proposal Presentation				🟩								
3	Implementation Stage 1												
	System Design and Planning					🟧							
	Implementation of functions					🟧	🟧						
	Integration and testing Level 1						🟧	🟧					
	Progress presentation -50%								🟧				
	Prepare Research Paper								🟧	🟧			
4	Implementation Stage 2												
	Implementation of functions										🟩		
	Integration and testing Level 2											🟩	
	Progress presenation -100%											🟩	
5	Final Stage												
	Final Thesis											🟨	🟨
	Final Presentation											🟨	

Figure 1 - Gannt Chart

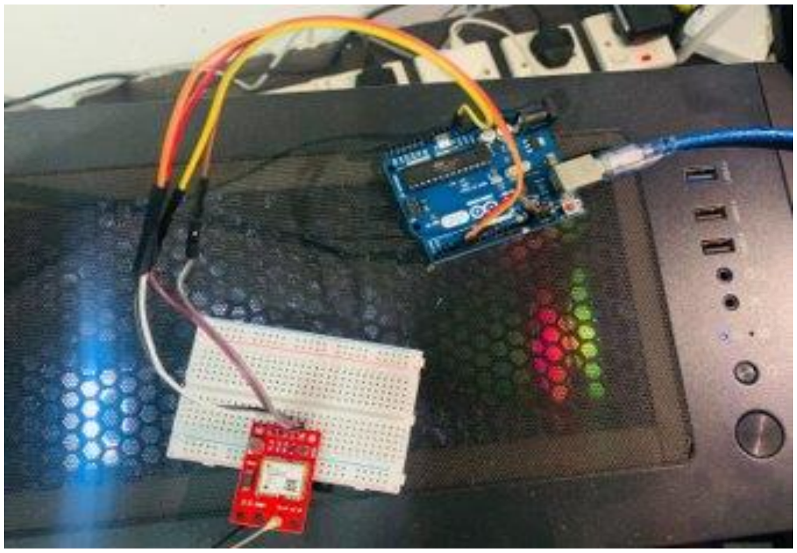


Figure 1 - Gannt Chart

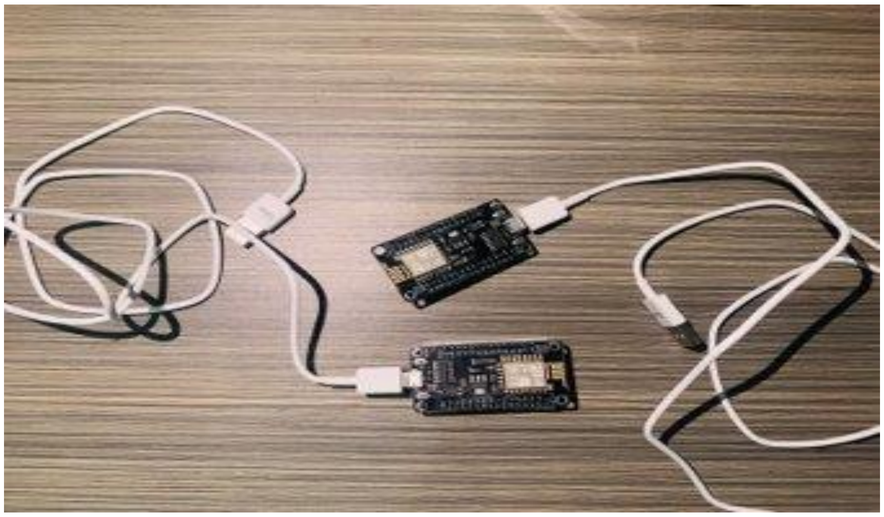


Figure 1 - Gannt Chart

\Appendix B: Work Break Down Chart

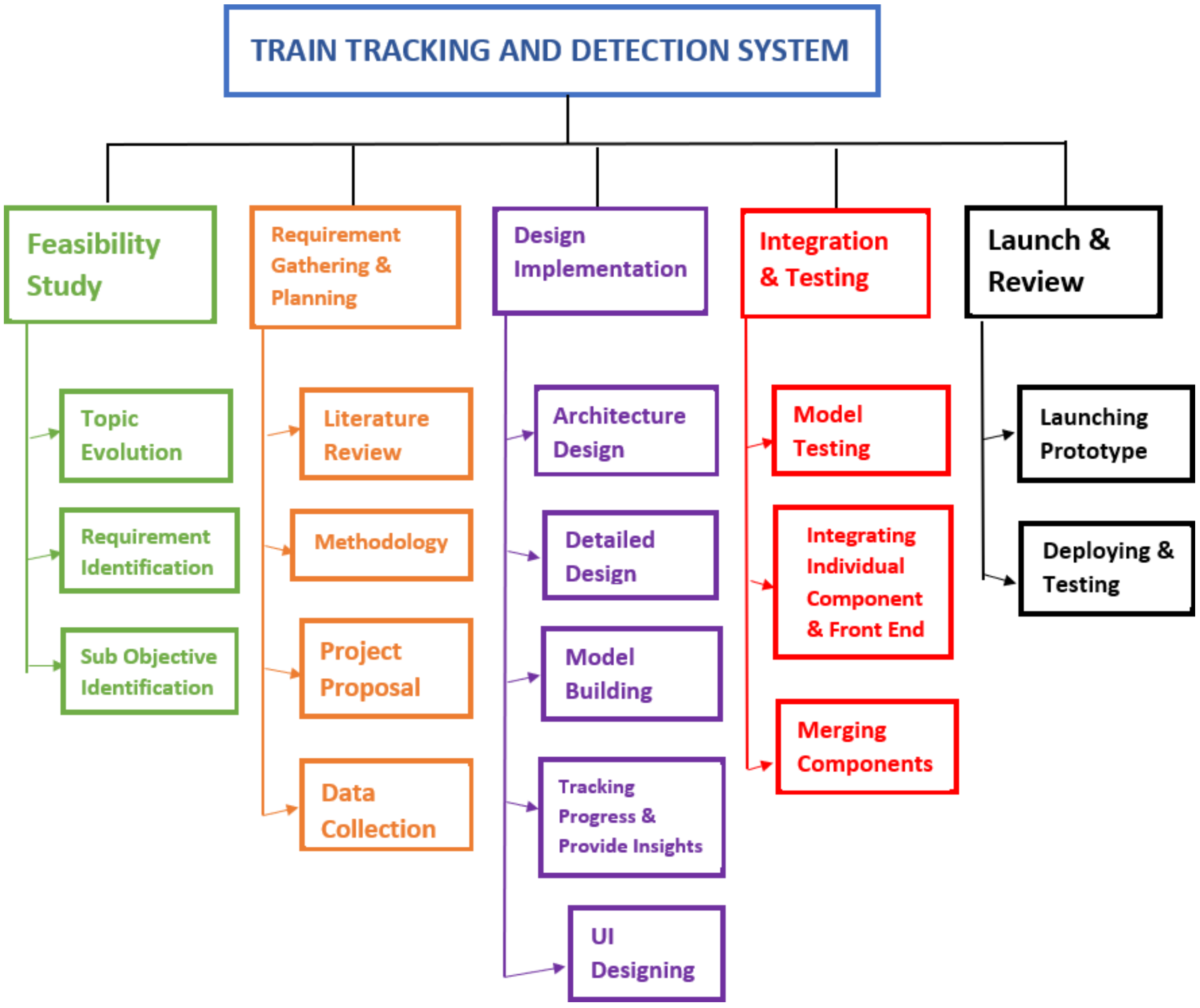


Figure 2: Work Break Down chart