Mini-Project Report On

Automated Railway Crossing()

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science & Engineering

By

Kevin Suresh (U2003118)
George K George (U2004035)
Jovan C John (U2003111)
Kamil Hadi (U2003115)

Under the guidance of
Mr. Sandy Joseph

RSET
RAJAGIRI SCHOOL OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS)

Department of Computer Science & Engineering
Rajagiri School of Engineering and Technology (Autonomous)
Rajagiri Valley, Kakkanad, Kochi, 682039

July 2023

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
RAJAGIRI SCHOOL OF ENGINEERING AND TECHNOLOGY (AUTONOMOUS)
RAJAGIRI VALLEY, KAKKANAD, KOCHI, 682039

CERTIFICATE

This is to certify that the mini-project report entitled ***"Automated Railway Crossing ()"*** *is a bonafide work done by* ***Mr. Kevin Suresh (U2003118), Mr. George K George (U2004035), Mr. Jovan C John (U2003111) and Mr. Kamil Hadi (U2003115)****, submitted to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B. Tech.) in Computer Science and Engineering during the academic year 2022-2023.*

Dr. Preetha K. G.	**Mr. Uday Babu P.**	**Mr. Sandy Joseph**
Head of Department	Mini-Project Coordinator	Mini-Project Guide
Dept. of CSE	Asst. Professor	Asst. Professor
RSET	Dept. of CSE	Dept. of CSE
	RSET	RSET

ACKNOWLEDGEMENTS

We wish to express our sincere gratitude towards **Dr. P. S. Sreejith**, Principal of RSET, and **Dr. Preetha K. G.**, Head of Department of Computer Science and Engineering for providing us with the opportunity to undertake our mini-project, "Automated Railway Crossing".

We are highly indebted to our mini-project coordinators, **Mr. Uday Babu**, Assistant Professor, Department of Computer Science and Engineering and **Ms. Tripti C**, Assistant Professor, Department of Computer Science and Engineering for their valuable support.

It is indeed our pleasure and a moment of satisfaction for us to express our sincere gratitude to our mini-project guide **Mr. Sandy Joseph**, for his patience and all the priceless advice and wisdom he has shared with us.

Last but not the least, we would like to express our sincere gratitude towards all other teachers and friends for their continuous support and constructive ideas.

Kevin Suresh

George K George

Jovan C John

Kamil Hadi

ABSTRACT

Manual crossings are prone to inefficient traffic management and long waiting times. This could cause major traffic blocks which causes emergency response vehicles to not be able to reach the destination in time. In contrast, automated systems enhance safety by detecting approaching trains and ensure timely closure of gates. They facilitate efficient traffic flow, minimize waiting times, and can be installed in remote areas for better accessibility. In this project we have developed an automated railway crossing system that maximizes efficiency at railroad crossings while also maintaining safety. The program runs constantly, updating a trains arrival time to a station from the RailYatri website and sending signals to a processing board responsible for the closing of the gate. The processing board also checks for any obstruction under the gate before closing it and alerts nearby vehicles the closing time of the gate with an audible tone few moments before the gate closes. The prototype of this board consists of an Arduino to receive and process the signal, a servo to simulate the opening and closing of the gate, an ultrasonic sensor to detect obstructions and an LCD screen to display the timer. The program uses web scraping to get all information of the trains that travel between the two stations throughout the day and adds the expected time of arrival of these trains to the stations into a database. This database will be constantly checked for the train expected to arrive next and will signal the processing board to begin the gate closing procedure a few minutes before the actual arrival of the gate.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

The Automated Railway Crossing project is a valuable tool that has the potential to improve railway safety, efficiency, and reliability in a number of ways.

The project automates railway crossings, ensuring enhanced safety for trains and pedestrians. By providing real-time train information, train operators and railway staff can efficiently manage the crossing gates and avoid potential accidents. This can significantly reduce the risk of accidents at railway crossings, which are a leading cause of death and injury around the world.

The project reduces the need for manual intervention, leading to more efficient and smoother operations. This can result in cost savings and increased reliability in railway services. For example, the project could be used to automatically close crossing gates when a train is approaching, eliminating the need for a human operator to do so. This would free up the operator to focus on other tasks, such as monitoring the train's progress or responding to other emergencies.

The project provides a practical learning experience for developers and students interested in web scraping, GUI development, and database management. It allows them to apply their skills in a real-world scenario and gain insights into railway systems. This can be a valuable experience for students who are interested in pursuing a career in transportation engineering or automation.

The project serves as a showcase of technology, demonstrating the use of web scraping, database management, and GUI development in creating a functional and practical application. This can be helpful for railway authorities and operators who are considering implementing similar systems. The project's findings can also be used to improve the design and implementation of future automated railway crossing systems.

Overall, the project is a valuable tool that has the potential to improve railway safety, efficiency, and reliability in a number of ways. It is a valuable resource for railway authorities, operators, and students interested in transportation systems and automation technologies.

1.2 Existing System

The most traditional system is a system with a track circuit, where the rails are used as a kind of electrical circuit and the voltage in these sections is measured and when any heavy rolling stock enters that section, an identification is made and this section of track next to the level crossing, the process of warning cars that a train is approaching begins and the entire process of closing the barrier begins, luminous signals and bells . Then, when the train passes through this circuit and enters the next track circuit, after the level crossing, the barrier is opened again. The other technology uses what we call axis sensors. An axle sensor (fail safe) identifies the axle of the locomotive on approach to the level crossing.

1.3 Problem Statement

The challenge is to create an application which will completely automate the operation of railway crossing gates.

Manual crossings result in inefficient traffic management, longer waiting times, and limited availability of gate crossings. They are prone to safety risks due to human error and can lead to accidents and collisions with trains.

The application addresses the issues discussed above by eliminating the human factor from the process by making the whole process unmanned and automatic.

1.4 Objectives

- **Obtain Train Details**- Obtain the train codes & subsequent details of the trains passing through the level crossing.

- **Automated Gate Closing** - Use the time known in the database to close the gate appropriately.
- **Emergency Open/Close Feature** - Manually obtain control of the level crossing when unexpected situation arises.

1.5 Scope

Web scraping is an automatic method to obtain large amounts of data from websites. Most of this data is unstructured data in an HTML format which is then converted into structured data in a spreadsheet or a database so that it can be used in various applications. There are many different ways to perform web scraping to obtain data from websites. These include using online services, particular API's or even creating your code for web scraping from scratch. Many large websites, like Google, Twitter, Facebook, etc. have API's that allow you to access their data in a structured format. This is the best option, but there are other sites that don't allow users to access large amounts of data in a structured form or they are simply not that technologically advanced. In that situation, it's best to use Web Scraping to scrape the website for data.

With many websites showing additional details of train such as estimated arrival, expected delay and current position, we can obtain these details using web scraping and predict with more accuracy and efficiency the estimated time of arrival of the train at the railway crossing.

Chapter 2

Literature Review

Explain the existing methods and their drawbacks. Draw a comparison table to compare the existing methods.

2.1 Sensor Based Automated Railway Crossing

A sensor-based automated railway crossing, also known as an "automatic level crossing," is a type of railway crossing that employs various sensors and technologies to detect approaching trains and vehicles. The system automatically controls the closing and opening of the crossing gates to ensure the safety of both road users and train passengers.

How it works: 1. Sensors: Automated railway crossings typically use various sensors such as radar, infrared detectors, or cameras to detect approaching trains. These sensors can detect the presence, speed, and distance of an oncoming train.

2. Traffic Detection: The system also incorporates sensors to detect road traffic, including vehicles and pedestrians, on the approaching roads. This helps in determining the timing of the gate closure and ensures that vehicles and pedestrians have sufficient time to clear the crossing before the train arrives.

3. Control System: The data from the sensors is fed into a control system that processes the information and triggers the automated crossing mechanism. If a train is detected within a certain distance from the crossing, the system activates the warning signals, closes the crossing gates, and activates flashing lights and bells to alert road users.

4. Train Clearance: Once the train has passed the crossing and is at a safe distance, the system opens the crossing gates again, allowing road traffic to pass.

Drawbacks: 1. Initial Cost: Implementing an automated railway crossing system involves significant upfront costs for installing sensors, control systems, and related infrastructure. These costs can be a barrier for many railways, especially in regions with limited resources.

2. Maintenance: Sensor-based systems require regular maintenance to ensure sensors are functioning correctly. Dust, debris, and environmental factors can impact sensor performance and may lead to false readings or malfunctions.

3. Vulnerability to Technical Failures: Like any automated system, there is a risk of technical failures, such as sensor malfunctions or communication issues, which could lead to unsafe conditions at the crossing.

4. Power Dependency: Automated crossings rely on a stable power supply. Power outages or electrical failures could lead to the crossing gates not operating correctly, posing safety risks.

5. Compatibility with Existing Infrastructure: Retrofitting existing manual railway crossings with automated systems can be challenging and may require significant modifications to the infrastructure.

6. Limited Adaptability: Automated crossings operate based on predefined algorithms and sensor data. They may lack the ability to respond quickly to dynamic situations or unexpected events, which human operators can handle more flexibly.

2.2 Computer Vision Based Railway Crossing

A computer vision-based railway crossing is an advanced system that utilizes computer vision technology to enhance safety and efficiency at railway crossings. Computer vision involves using cameras and image processing algorithms to analyze visual data and make decisions based on the detected objects and their movements. In the context of railway crossings, computer vision can be used to detect trains, vehicles, pedestrians, and other relevant objects to automate the crossing's operation.

How it works: 1. Cameras: Computer vision-based railway crossings are equipped with cameras strategically positioned to capture real-time video footage of the crossing area.

2. Object Detection: The computer vision algorithms process the video data to detect various objects, such as trains, vehicles, pedestrians, and obstacles, approaching or within the crossing zone.

3. Decision Making: Based on the analysis of the video feed, the system can make

real-time decisions on whether to activate the warning signals, close the crossing gates, and activate the flashing lights and bells.

4. Train Clearance: Once the train has passed the crossing and is at a safe distance, the system opens the crossing gates again, allowing road traffic to pass.

Potential Drawbacks: 1. High Initial Cost: Implementing a computer vision-based railway crossing involves significant upfront investments in high-quality cameras, computing hardware, and sophisticated image processing software. This can make the system costly to deploy and maintain.

2. Image Processing Challenges: Computer vision algorithms may face challenges in adverse weather conditions, poor lighting, or cluttered scenes, affecting the accuracy of object detection and decision-making.

3. Maintenance: Cameras and related equipment require regular maintenance to ensure they are clean, functional, and calibrated correctly. Dust, debris, and environmental factors can impact the performance of the system.

4. Dependence on Network Connectivity: Computer vision-based systems often require a reliable network connection to transmit video data to the processing unit. Network outages or disruptions can affect the system's real-time response.

7. Privacy Concerns: The use of cameras for real-time monitoring raises privacy concerns, especially if the system records or stores sensitive data.

8. Compatibility with Existing Infrastructure: Retrofitting existing manual railway crossings with computer vision-based systems can be complex and may require extensive modifications.

2.3 RFID Based Railway Crossings

Radio Frequency Identification (RFID) railway crossing systems are a type of technology used to enhance safety and efficiency at railway crossings. RFID is a wireless technology that uses radio waves to communicate between RFID tags and readers. In the context of railway crossings, RFID technology is employed to identify and track trains, vehicles, and other objects in the crossing area.

How it works: 1. RFID Tags: RFID tags are attached to trains, vehicles, and other objects of interest. These tags contain unique identification information that can be read by RFID readers.

2. RFID Readers: RFID readers are installed at the railway crossing area. As the tagged objects, such as trains or vehicles, approach the crossing, the RFID readers scan the RFID tags to identify and track them.

3. Data Processing: The RFID reader captures the unique identification information from the tags and sends it to a central control system for processing.

4. Decision Making: The central control system uses the data from the RFID readers to make real-time decisions on when to activate the warning signals, close the crossing gates, and activate the flashing lights and bells.

5. Train Clearance: Once the train has passed the crossing and is at a safe distance, the system opens the crossing gates again, allowing road traffic to pass.

Potential Drawbacks: 1. Cost: Implementing RFID railway crossing systems involves costs associated with RFID tags, readers, and the central control system. While RFID technology has become more affordable over time, it can still be a significant investment, especially for large railway networks.

2. Infrastructure Requirements: The successful deployment of RFID systems requires the installation of RFID readers and associated infrastructure at the railway crossing area. Retrofitting existing crossings with RFID technology may require modifications and adjustments.

3. Maintenance: RFID readers and tags require regular maintenance to ensure they function correctly. Physical damage, environmental factors, or technical issues could impact the system's reliability.

4. Reading Range: The reading range of RFID systems is limited. In some cases, the RFID tags may need to be in close proximity to the readers for reliable communication, which could potentially cause delays at the crossing if not optimized properly.

5. Integration with Other Systems: Integrating RFID systems with existing railway management and signaling systems can be complex, and interoperability challenges may arise.

6. Electromagnetic Interference: RFID systems can be sensitive to electromagnetic interference, which could affect their performance in certain environments.

Despite these challenges, RFID railway crossing systems offer the potential for improved safety and automated control, reducing the likelihood of accidents at level crossings. Addressing the drawbacks requires careful planning, regular maintenance, and adherence to safety and privacy regulations to ensure the effective and secure implementation of RFID technology in railway crossing operations.

Chapter 3

System Analysis

3.1 Expected System Requirements

The system is expected to have the following features:

- Windows platform with a version above Windows 10.
- Requirement of Internet connection for web scraping.
- A storage space of approximate 100 MB for the app.
- A minimum Ram size of 4GB is required in the device.
- Arduino board with an ultrasonic sensor and a servo motor.

3.2 Feasibility Analysis

3.2.1 Technical Feasibility

The project is technically feasible as merging this method with existing railway crossings is not very difficult.

3.2.2 Operational Feasibility

Once installed the entire process runs automatically with the exception of an emergency stop. Installation of the app is the only prerequisite operation to be done.

3.2.3 Economic Feasibility

The app can reduce the overhead of expense incurred by hiring people to keep watch at the railway crossing by automating the entire process.

3.3 Hardware Requirements

The following are the system requirements to develop the Automated Railway Crossing.

- Processor: Intel Core i5
- Hard Disk: Minimum 100GB
- RAM: Minimum 8GB
- Arduino UNO v3
- LCD Display,Buzzer,Servo,LEDs

3.4 Software Requirements

The following are the softwares used in the development of the app.

Operating System: Windows 11

3.4.1 Python 3.11

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation via the off-side rule.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

3.4.2 Arduino IDE 2

Arduino IDE is an open-source software, designed by Arduino.cc and mainly used for writing, compiling and uploading code to almost all Arduino Modules.It is an official Arduino software, making code compilation too easy that even a common person with no prior technical knowledge can get their feet wet with the learning process.

It is available for all operating systems i.e. MAC, Windows, Linux and runs on the Java Platform that comes with inbuilt functions and commands that play a vital role in debugging, editing and compiling the code.A range of Arduino modules available including Arduino Uno, Arduino Mega, Arduino Leonardo, Arduino Micro and many more.

3.4.3 Selenium

Selenium is a Python library and tool used for automating web browsers to do a number of tasks. One of such is web-scraping to extract useful data and information that may be otherwise unavailable. Selenium provides a wide range of ways to interact with sites, such as clicking buttons, populating forms with data, scrolling the page, taking screenshots, executing your own, custom JavaScript code etc.

But the strongest argument in its favor is the ability to handle sites in a natural way, just as any browser will. This particularly comes to shine with JavaScript-heavy Single-Page Application sites. If you scraped such a site with the traditional combination of HTTP client and HTML parser, you'd mostly have lots of JavaScript files, but not so much data to scrape.

Chapter 4

Methodology

Block diagram + Explain each block.

4.1 Proposed Method

- Develop a python application that can scrape details of trains from the internet.
- User gives two station codes as input and application gives the details of trains passing through the railway crossing as output.
- The application calculates time in seconds for the next train to arrives and sends it to the arduino.
- Application also contains features like force opening and force closing of the railway gate and automatic updation of details.

4.1.1 Web Scraping

Web scraping is the process of extracting data from websites automatically. It involves using software tools or scripts to access web pages, fetch the data, and then parse and extract the relevant information. Web scraping is commonly used for various purposes, such as data analysis, market research, content aggregation, monitoring website changes, and much more.

In our proposed architecture, web scraping is done using the Selenium library, which is primarily used for automating web browsers. The script accesses the website ”railyatri.in” to retrieve live train status information. It extracts details such as station names, arrival times, train statuses, halt times, platforms, and locomotive reverse information. These details are then stored in a local SQLite database named ”Train.db.”

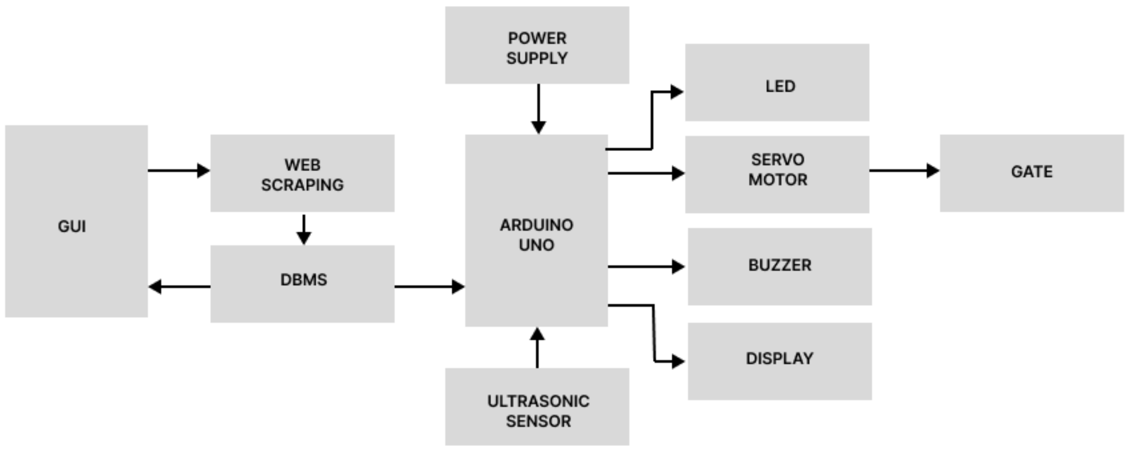


Figure 4.1: model structure

The script goes through a list of train codes, likely obtained from a custom module named "traincodeextract",which defines a function named "traincode" that aims to extract and return train codes along with their corresponding station names for trains traveling between two specified source and destination stations.We utilize the "datetime" library to get the current date and formats it accordingly. The script then sends HTTP requests to these URLs using "requests" and parses the HTML content of the web pages using "BeautifulSoup." It extracts the train codes and station names from the parsed data, storing them in a dictionary named "deets." The function ultimately returns this dictionary containing the train codes and their respective station names for trains traveling between the specified source and destination stations.

Our python script then uses each train code to fetch the corresponding train's status from the website. It does this by navigating to the URL specific to each train using Selenium's browser automation capabilities. After successfully scraping the required train details, the information is inserted into the SQLite database for analysis.By combining Selenium with SQLite, the script effectively automates the process of collecting live train status data and organizing it in a structured manner within a local database.

4.1.2 Arduino

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It consists of both physical boards with microcontrollers (e.g., Arduino Uno, Arduino Mega) and an integrated development environment (IDE) that simplifies programming. Arduino boards are widely used in the maker community, educational settings, and various projects to control and interact with a wide range of electronic components.

In our sketch, Arduino is used to control a gate system. The key components used in this project are:

1. Arduino Board:We use the Arduino Uno.The board acts as the main controller for the gate system.

2. Servo Motor: Arduino can interface with various components, including servo motors. A servo motor is used in this project to physically control the gate's movement, allowing it to open and close smoothly.

3. LiquidCrystal Display: Arduino can drive a LiquidCrystal display to provide visual feedback and status information. In this project, a 16x2 LiquidCrystal display is used to show messages and the countdown timer for the gate closing.

4. LEDs: Arduino can control LEDs to indicate different states or events. In this sketch, two LEDs (led1 and led2) are used to indicate the gate's status or some specific events during the gate's operation.

5. Buzzer: Arduino can generate tones using a buzzer. Here, a buzzer is used to produce a beep sound, alerting users during certain gate operations.

6. Serial Communication: Arduino can communicate with other devices, like a computer, using the Serial interface. In this sketch, the gate system can receive commands through Serial communication, allowing users to remotely control the gate actions by sending specific commands from a computer or other devices.

Overall, Arduino simplifies the process of integrating hardware components and programming them to build interactive projects like this gate control system. It offers a user-friendly platform that enables both beginners and experienced users to create various electronic projects with ease.

Chapter 5

System Design

Draw usecase diagrams, activity diagrams, sequence diagrams etc. Add a brief explanation for each UML diagram.

5.1 Architecture Diagram

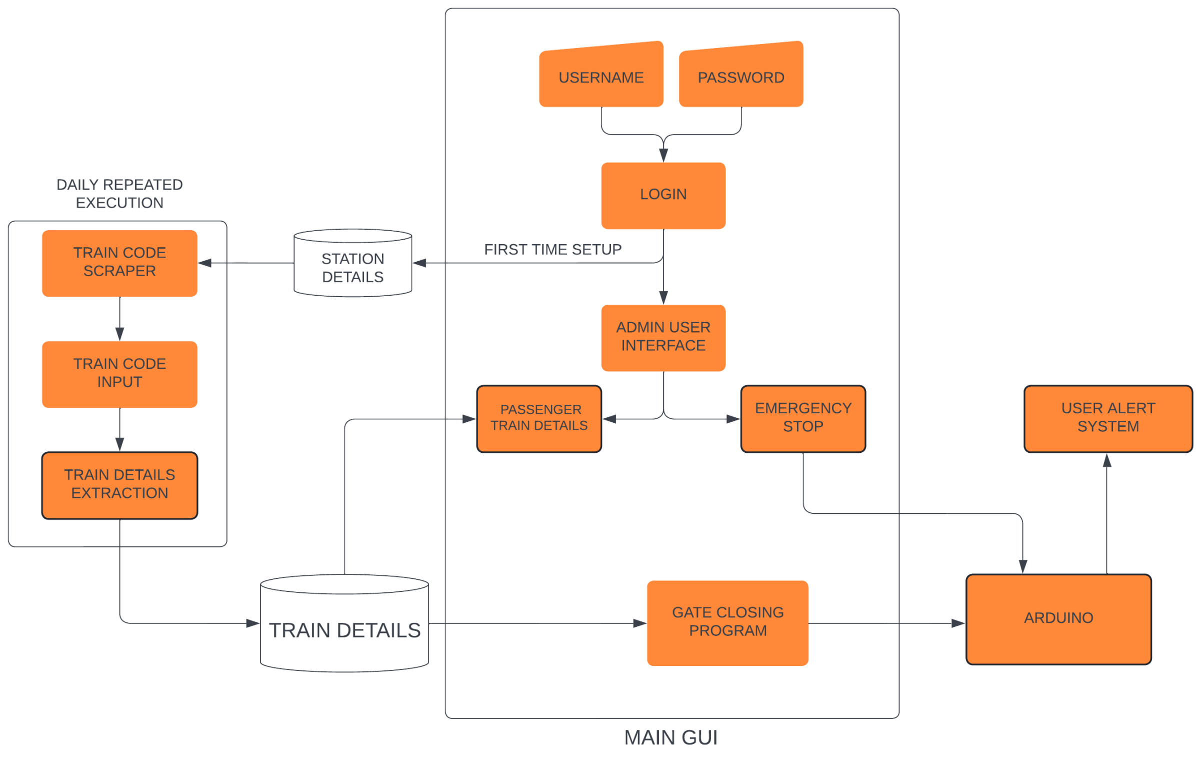


Figure 5.1: Architecture diagram

5.2 Sequence diagram

5.2.1 First Time Setup

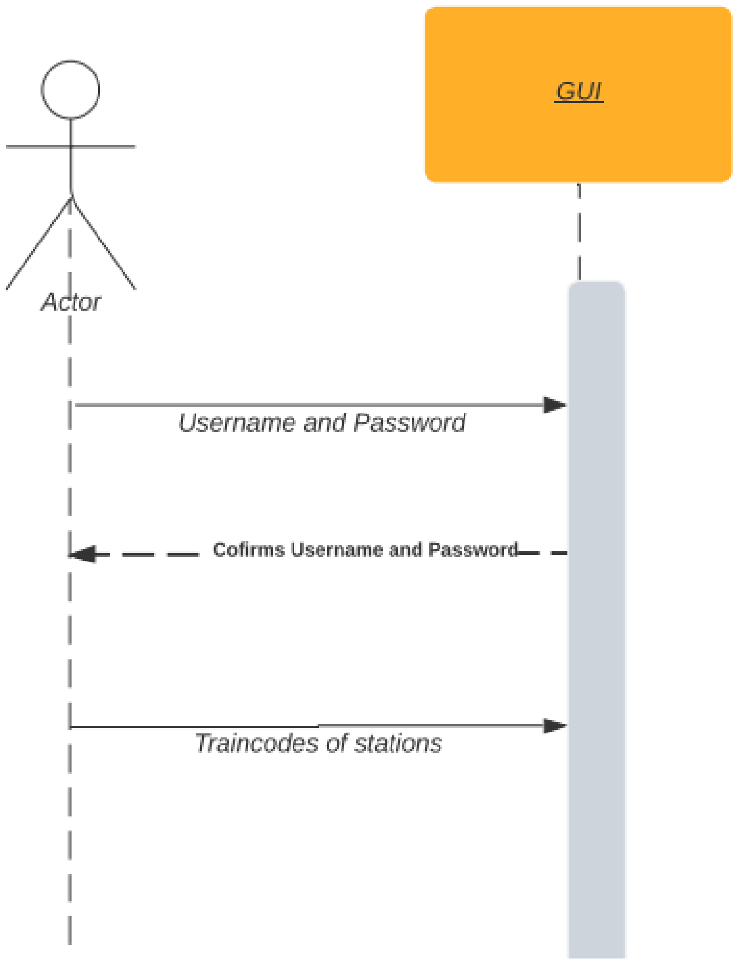


Figure 5.2: First Time Setup

5.2.2 Train Details Scraping

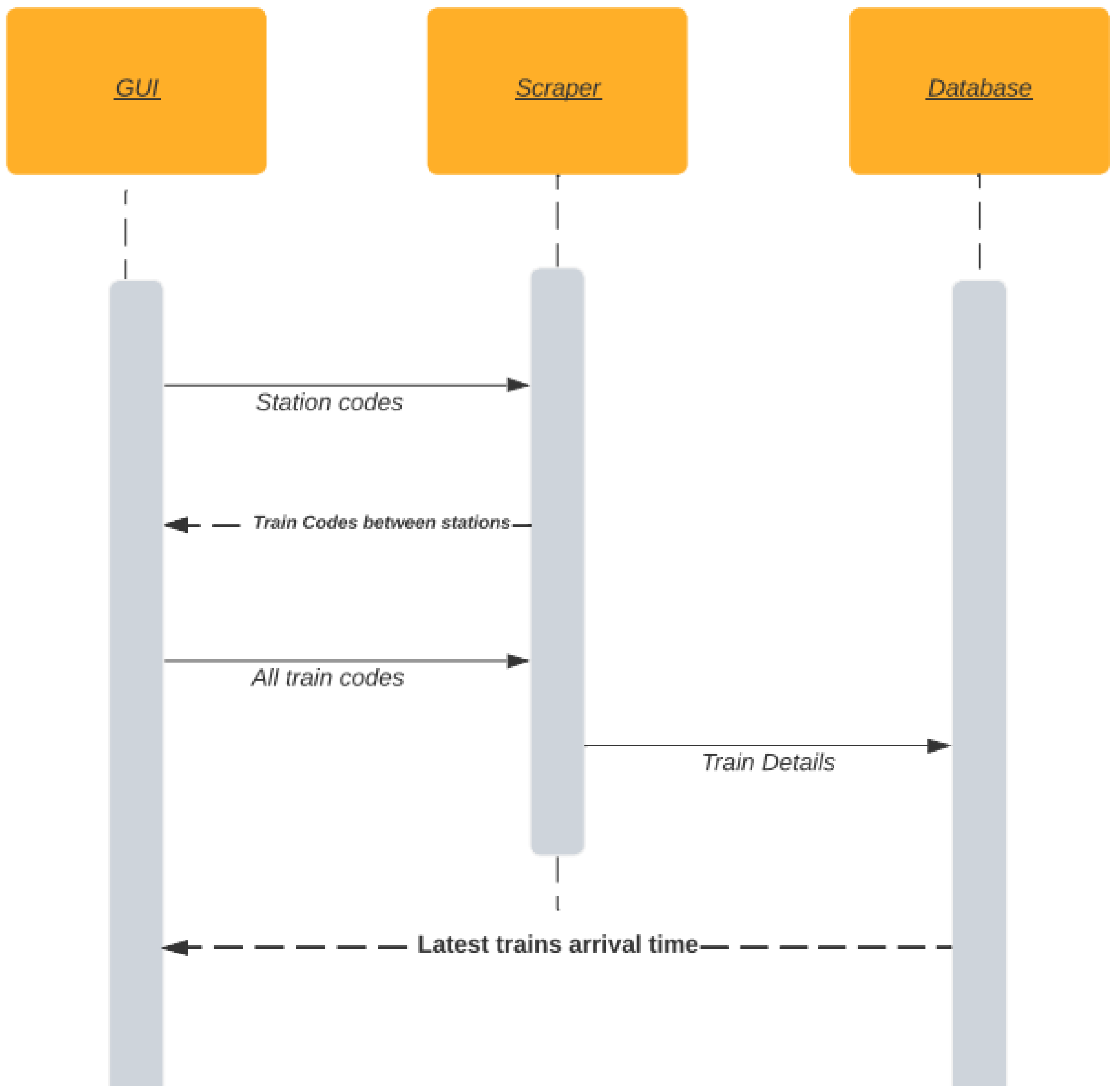


Figure 5.3: Train Details Scraping

5.2.3 Gate Closing Mechanism

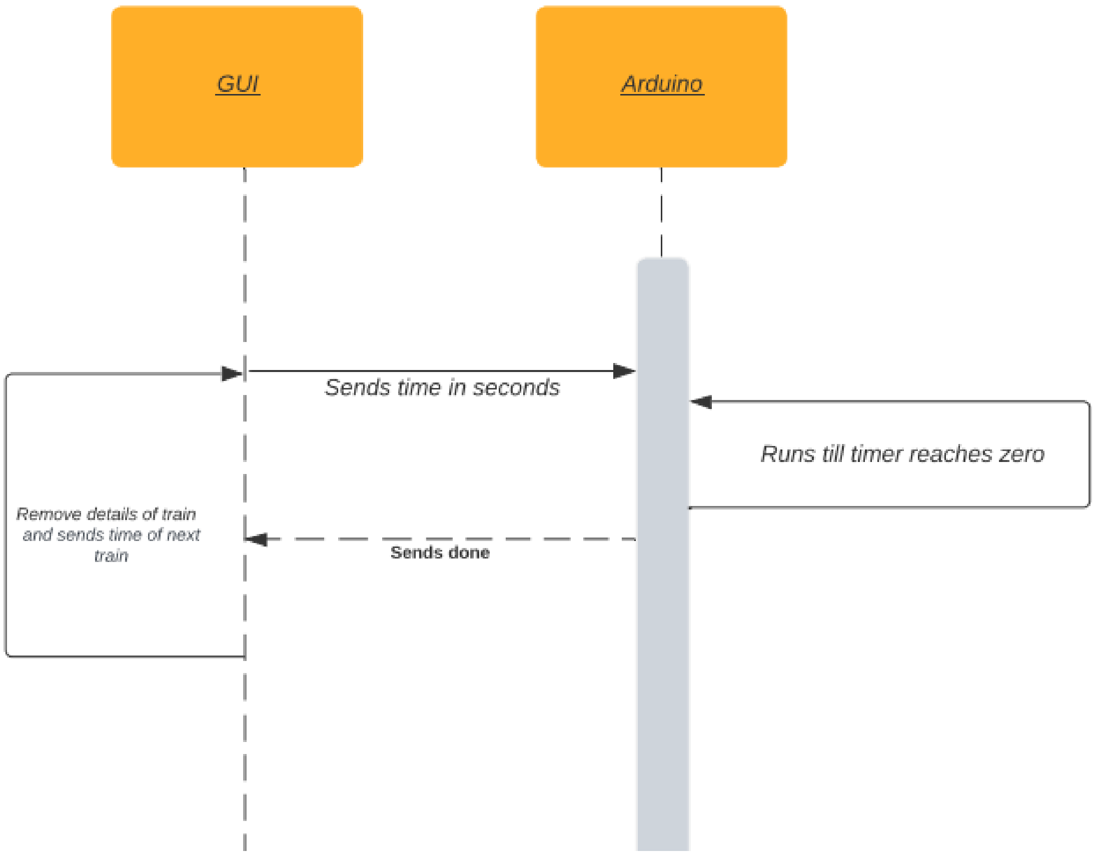

GUI
Arduino
Sends time in seconds
Runs till timer reaches zero
Remove details of train and sends time of next train
Sends done

Figure 5.4: Gate Closing Mechanism

Chapter 6

System Implementation

Explain the various modules in the system in detail + Code.

6.1 Admin Terminal GUI

GUI

Frame 1: Login Frame ('loginframe')

This frame represents the login screen where users are required to enter their ID and password to access the application. The frame provides entry fields for ID and password, along with a "Show Password" checkbox to toggle password visibility. When the user clicks the "Login" button, the application validates the entered ID and password.Upon successful login, the frame switches to 'frame2'.

Frame 2: Train Details Input Frame ('frame2')

After successful login, this frame is displayed, allowing users to input train details. The frame contains entry fields for the two station codes where the crossing is located in between. When the user clicks the "Continue" button, the application uses the entered station codes to fetch real-time train details using web scraping. Once the details are fetched, the frame switches to 'frame3'.

Frame 3: Train Details Display Frame ('frame3')

This frame displays the real-time train information fetched using web scraping from the train website. The frame shows the train code, station name, arrival time for the requested stations. Additionally, control buttons ("Open Gate," "Close Gate," and "Exit") are provided to manually control the railway crossing in case of emergency. The application continuously updates and displays the latest train details.

In summary, the GUI application provides an interactive interface for an "Automated Railway Crossing" system, where users can log in, enter train details, fetch real-time train information, and visualize the train details on the screen. The application allows users to monitor train movements and control the railway crossing gates as if it were a real automated system.

6.2 Web Scraping

The Python function ‘traincode‘ employs web scraping techniques to obtain and process train details between two designated stations. Using the BeautifulSoup and requests libraries, the function parses the content of a web page related to the specified stations, extracting relevant information such as train codes and station names, for all the trains going between the two stations on the current day. It then organizes this data into a dictionary, where the train codes serve as keys and their respective station names act as values. The function repeats the process for the opposite direction as well, providing a comprehensive dictionary encompassing train details for both directions of the specified route.

Train Details

The provided Python script utilizes the Selenium web automation library to scrape live train status information from the website ”railyatri.in.” It connects to an SQLite database named ”Train.db” and defines a function, ”scrape_details,” responsible for extracting train details and inserting them into the database. The script loops through a set of train codes obtained from ”traincodeextract,”which was previously scrapped and scrapes the details for a specified number of trains. Each train’s details, such as station, arrival time, train status, halt time, platform, and locomotive reverse, are stored in the database. The script handles potential timeout exceptions and prints an error message if needed. After successfully scraping the desired train details, the database connection is closed, and the script terminates the Selenium-controlled browser.

6.3 Arduino

The Arduino sketch presented is a comprehensive program that controls a gate system using a servo motor. It incorporates additional functionalities such as a LiquidCrystal display to show status information, LEDs for visual indicators,sensor for object detection and a buzzer for alert sounds. The sketch employs a countdown mechanism to facilitate the gradual closing of the gate.The countdown is displayed on the screen and the buzzer starts beeping and the LED starts blinking when it's nearing the end of the countdown. It also listens for commands via Serial communication, enabling users to interact with the gate system remotely. Users can issue commands like "OPEN," "FCLOSE" (forceful close), or "FOPEN" (forceful open) to trigger specific gate actions.

Overall, the Arduino sketch provides a versatile and interactive solution for gate control, catering to various scenarios through user-defined commands and utilizing sensor input for obstacle detection. The combination of servo motor control, LiquidCrystal display, LEDs, and buzzer enhances the gate system's user experience, making it a flexible and practical solution for managing gate access and security.

Chapter 7

Testing

Write about the different types of testing you have done (Unit Testing, Integration Testing, System Testing, Acceptance Testing, Cross Device Testing, Security Testing, User Interface Testing). Summarize the results.

The testing of an automated railway crossing is an important part of ensuring the safety and reliability of the system. The tests should cover a wide range of scenarios, including the correct detection of trains approaching the crossing, the timely closure of the crossing gates, the safe operation of the crossing gates in the event of a train malfunction and the ability of the system to handle multiple trains simultaneously.

The results of the tests should be carefully analyzed to identify any potential problems with the system. Any problems that are identified should be addressed before the system is put into operation.

7.1 Unit Testing

For this automated railway crossing project, unit testing was carried out to verify the accuracy and functionality of various functions and methods within the code-base.

7.1.1 Scraping Unit

The scraping unit was tested by running the program multiple times on the system and finding ways to reduce the loading times and optimize retrieval of data from the web page. Rather than waiting for the whole page to load, we load the page till the required data has been loaded. Once the required data is visible, the loading of the web page stops and the data is extracted. If the data fails to load after the wait time, an error will be displayed and the page will be refreshed a fix amount of times to check if the data is available. If the data is still not available a fatal error will be raised and the details of the next train will be extracted.

7.1.2 Gate Operation Unit

The arduino gate closing unit was tested by passing various types of input values and monitoring the results. The passing of data to the arduino was made more fluid and efficient by introducing a delay both in the python application and the arduino program to successfully send and receive data. The components were also tested through various conditions and the values returned by the components were also checked to see whether it fell into the expected ranges.

7.2 Integration Testing

Integration testing in the automated railway gate project involved assessing the interactions and compatibility between different modules and components. Each module, including GUI, web scraping and the arduino gate operation, was systematically integrated to verify seamless communication and functionality. For instance, integration tests were designed to ensure that the GUI accurately displayed information transmitted from the web scraping module, such as the details of the train arriving next. We also tested the communications between the GUI and the arduino gate operation unit to ensure that the commands for forcefully opening and closing the gate worked as expected.

Chapter 8

Results

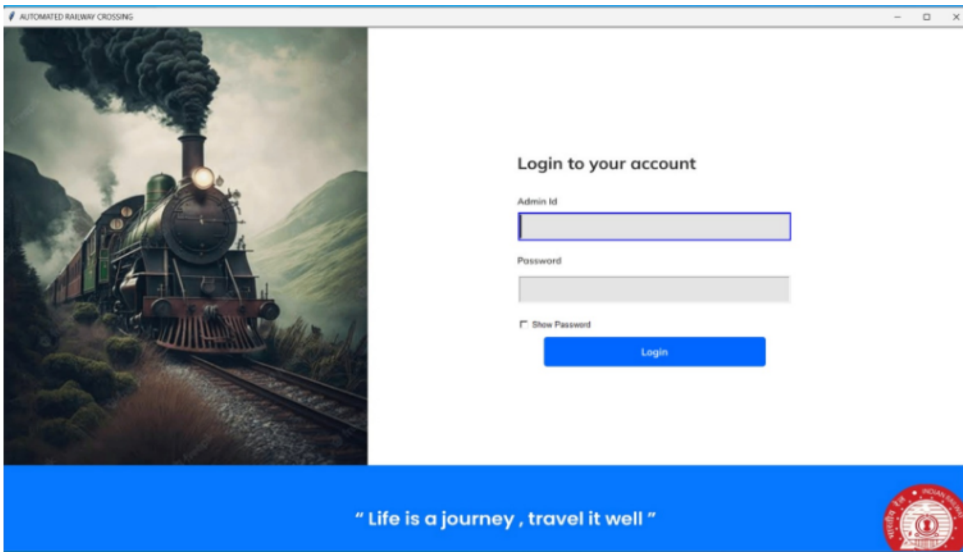

AUTOMATED RAILWAY CROSSING
Login to your account
Admin Id
Password
Show Password
Login
" Life is a journey , travel it well "

Figure 8.1: Admin Login Frame

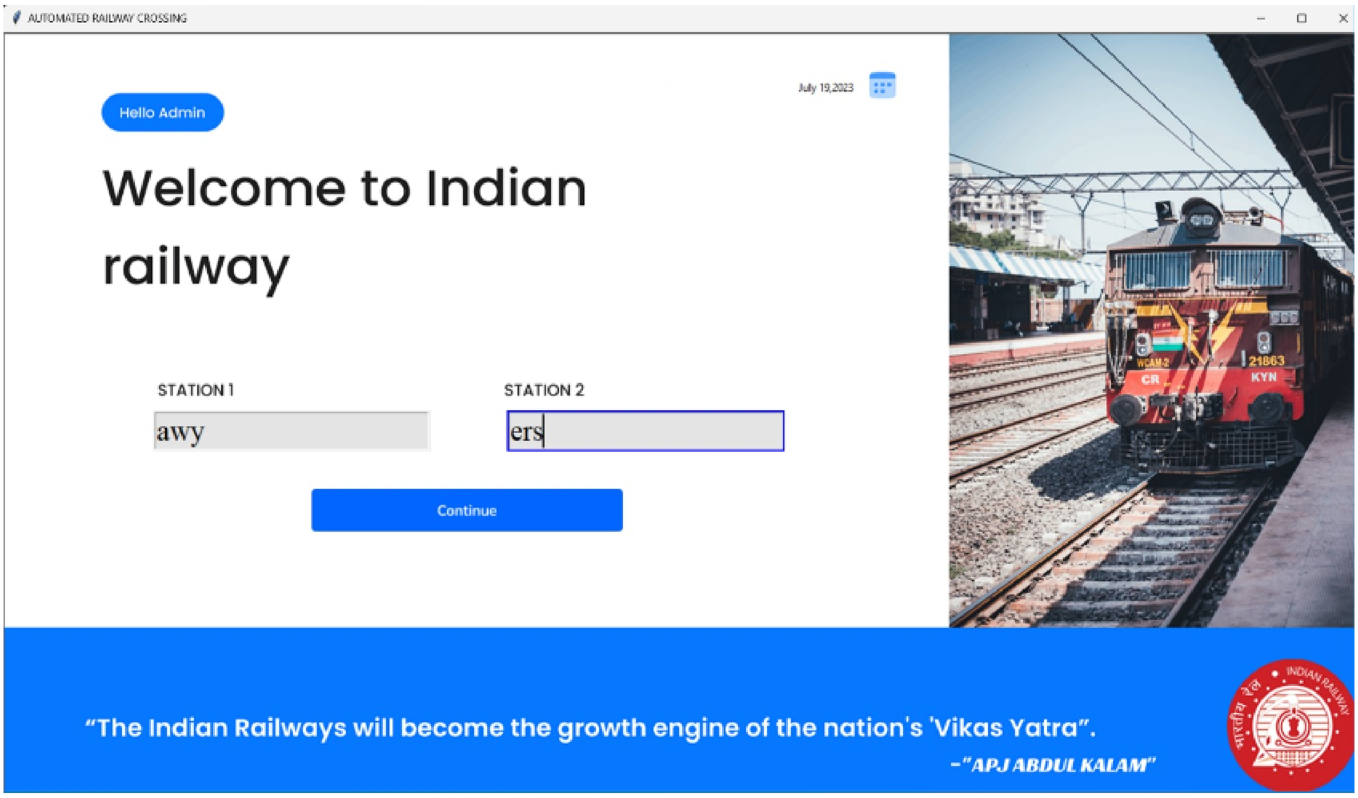
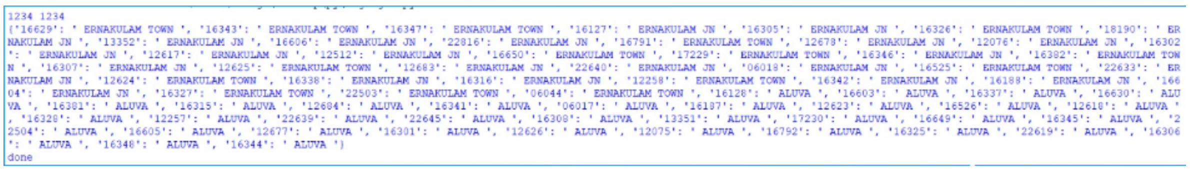
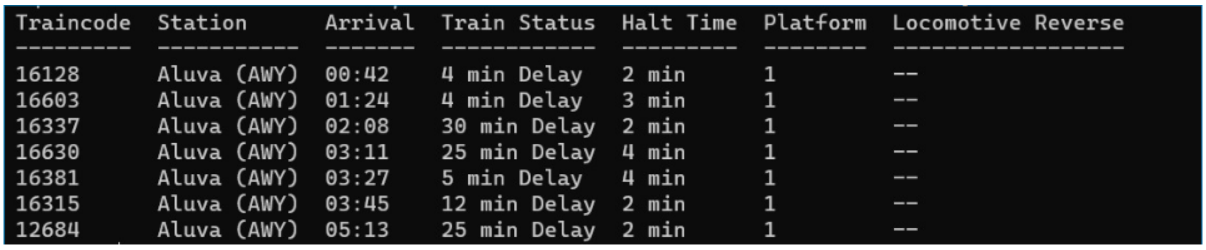

AUTOMATED RAILWAY CROSSING
July 19,2023
Hello Admin
Welcome to Indian railway
STATION 1
awy
STATION 2
ers
Continue
“The Indian Railways will become the growth engine of the nation's 'Vikas Yatra”.
-"APJ ABDUL KALAM"

Figure 8.2: First Time Setup



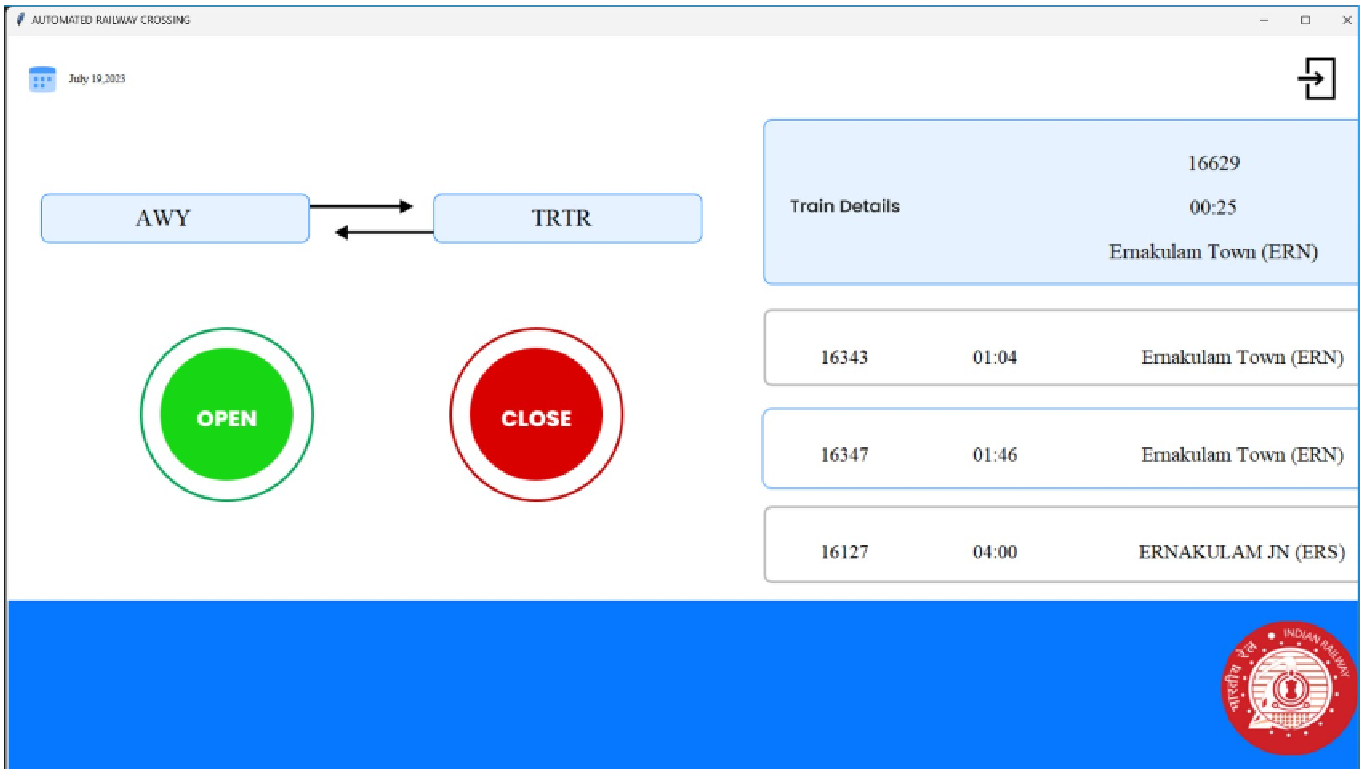
```
1234 1234
{'16629': ' ERNAKULAM TOWN ', '16343': ' ERNAKULAM TOWN ', '16347': ' ERNAKULAM TOWN ', '16127': ' ERNAKULAM JN ', '16305': ' ERNAKULAM JN ', '16326': ' ERNAKULAM TOWN ', '18190': ' ER
NAKULAM JN ', '13352': ' ERNAKULAM JN ', '16606': ' ERNAKULAM JN ', '22816': ' ERNAKULAM JN ', '16791': ' ERNAKULAM TOWN ', '12678': ' ERNAKULAM JN ', '12076': ' ERNAKULAM JN ', '16302
': ' ERNAKULAM JN ', '12617': ' ERNAKULAM JN ', '12512': ' ERNAKULAM JN ', '16650': ' ERNAKULAM TOWN ', '17229': ' ERNAKULAM TOWN ', '16346': ' ERNAKULAM JN ', '16382': ' ERNAKULAM TOW
N ', '16307': ' ERNAKULAM JN ', '12625': ' ERNAKULAM TOWN ', '12683': ' ERNAKULAM JN ', '22640': ' ERNAKULAM JN ', '06018': ' ERNAKULAM JN ', '16525': ' ERNAKULAM TOWN ', '22633': ' ER
NAKULAM JN ', '12624': ' ERNAKULAM TOWN ', '16338': ' ERNAKULAM JN ', '16316': ' ERNAKULAM JN ', '12258': ' ERNAKULAM TOWN ', '16342': ' ERNAKULAM JN ', '16188': ' ERNAKULAM JN ', '166
04': ' ERNAKULAM JN ', '16327': ' ERNAKULAM TOWN ', '22503': ' ERNAKULAM TOWN ', '06044': ' ERNAKULAM TOWN ', '16128': ' ALUVA ', '16603': ' ALUVA ', '16337': ' ALUVA ', '16630': ' ALU
VA ', '16381': ' ALUVA ', '16315': ' ALUVA ', '12684': ' ALUVA ', '16341': ' ALUVA ', '06017': ' ALUVA ', '16187': ' ALUVA ', '12623': ' ALUVA ', '16526': ' ALUVA ', '12618': ' ALUVA '
, '16328': ' ALUVA ', '12257': ' ALUVA ', '22639': ' ALUVA ', '22645': ' ALUVA ', '16308': ' ALUVA ', '13351': ' ALUVA ', '17230': ' ALUVA ', '16649': ' ALUVA ', '16345': ' ALUVA ', '2
2504': ' ALUVA ', '16605': ' ALUVA ', '12677': ' ALUVA ', '16301': ' ALUVA ', '12626': ' ALUVA ', '12075': ' ALUVA ', '16792': ' ALUVA ', '16325': ' ALUVA ', '22619': ' ALUVA ', '16306
': ' ALUVA ', '16348': ' ALUVA ', '16344': ' ALUVA '}
done
```

Figure 8.3: Intermediate Train Codes



```
Traincode  Station       Arrival  Train Status   Halt Time  Platform  Locomotive Reverse
---------  ------------  -------  -------------  ---------  --------  ------------------
16128      Aluva (AWY)   00:42    4 min Delay    2 min      1         --
16603      Aluva (AWY)   01:24    4 min Delay    3 min      1         --
16337      Aluva (AWY)   02:08    30 min Delay   2 min      1         --
16630      Aluva (AWY)   03:11    25 min Delay   4 min      1         --
16381      Aluva (AWY)   03:27    5 min Delay    4 min      1         --
16315      Aluva (AWY)   03:45    12 min Delay   2 min      1         --
12684      Aluva (AWY)   05:13    25 min Delay   2 min      1         --
```

Figure 8.4: Intermediate Train Details


AUTOMATED RAILWAY CROSSING

July 19,2023

Train Details

16629

00:25

Ernakulam Town (ERN)

16343	01:04	Ernakulam Town (ERN)
16347	01:46	Ernakulam Town (ERN)
16127	04:00	ERNAKULAM JN (ERS)

Figure 8.5: Main Frame

Chapter 9

Risks and Challenges

1. Internet Dependence : Failure to be able to access internet could cause the program to miss recording details of a train to the database.
2. Website Updation : Failure of data not being updated in the site could cause incorrect data to be stored in the database.
3. Hardware Reliability : Hardware components should be properly maintained and serviced for optimal functioning.

Chapter 10

Conclusion

Conclusion + Future scope

We have developed an python application that completely automates the gates at railway crossings to optimize efficiency and improve safety. The app consists of the following features: Web scraping train details, Timed closing of railway gate, forced opening and closing of gates and fully automated program execution. All of these features have been verified to be working as intended.

We have used selenium to extract all train codes using web scraping and store details of each train codes in a database. This database will be continuously checked for the most recent train to arrive at the railway crossing.

The program then sends the seconds remaining for the train to arrive at the railway crossing to the Arduino via serial ports. After the train passes the Arduino sends a message back to the program indicating that the train has successfully passed.

References

[1] Saifuddin Mahmud, 2015, International Journal of Computer Trends and Technology (IJCTT)

[2] M.I.M.Amjath and T.Kartheeswaran 2020, March. An Automated Level Crossing system. 2020 International Conference on Image Processing and Robotics.

[3] P Ilampiray , K Deepak , M G Deepak Santhosh , S kishore 2021, February. Automated Railway gate control system using Arduino and Ultrasonic sensors. ICCCEBS 2021: International Conference on Computing, Communication, Electrical and Biomedical Systems

Appendix A: Base Paper

Automated Railway gate control system using Arduino and Ultrasonic sensors

Automated Railway gate control system using Arduino and Ultrasonic sensors

P Ilampiray [1], K Deepak [1], M G Deepak Santhosh [1], S kishore [1]
[1] Department of Information Technology, Sri Krishna College of Technology, Coimbatore, India
ilampiray.p@skct.edu.in

Abstract. In a country like India, which has a population of about 1.39 Billion depends on its Transportation for its daily living. Transportation plays a main role in India's Economic Development and more than 35% of the lives of people in India depends on its transportation. Railways is the most popular and highly used transportation in India. It is the most effective mode of transportation not only in India, but also all over the World. As like its usage it also leads to a high number of accidents. Like road accidents, there are more accidents happening at railway crossings due to the unmanned level crossings and the carelessness of the road users. According to the National Crime Records Bureau [NCRB], the rate of level crossing accidents is 20% up in 2019 than 2018. However there will be an increase in the percentage of accidents every year but the percentage in the last two years has increased drastically. There were 1788 level crossing accidents in 2019, which include 1762 deaths. "Necessity is the mother of Invention" as like this saying the is a simple system, which automatically closes the level crossing gates during the train's arrival and then the gates are opened automatically when the train passes by. In our system we have used ultrasonic sensors for detecting the departure and arrival of the trains. As soon as the sensor senses the train's arrival it sends a message to the Arduino which will switch on the buzzer, so that the road users will know that the train is nearer, and after that the servo motor which is attached to the gates will close them and it will be opened after the train passes by. And then the buzzer will also be turned off. This is automated, highly effective and cheap. Our system will eliminate the manpower used at the level crossings.

Keywords: Arduino, Ultrasonic sensor, servo motor, buzzer.

1. Introduction
One of the most commonly used transportation nowadays is railways. Which is also a cost-effective transportation mode. Indian Railways is the largest rail network in Asia and second largest in the world. The trains are being continuously operated every single day. So, its nearly impossible to prevent some of the accidents during the train passing [1-3]. As per the survey there are thousands and thousands of people losing their life in train accidents. Most of these accidents are caused due to carelessness of people and unmanned level crossing. In our country it happens often due to the country's vast population, mainly it is happening in rural areas [4-7]. At least 1/3rd of the railway crossings goes unnoticed due to remote placement and less traffic, which results in accidents. Currently present railway crossings are not advanced and safe [8]. Therefore, these accidents cause serious damage to human life. The most common

rail accidents occur due to collision of trains and human errors [9,10]. The level crossing depends on humans for operating the opening and closing of railway gates. Hence for accessing the gates without man power, a new system is developed using Arduino. So, we are implementing this wherever it is possible to make sure all the people are safe during level crossings with less manpower, which is automated.

2. Related Work

In recent times many automatic railway gate control systems are invented for the railway crossings to prevent the accidents. [11], reduced the accident by concentrating on the unguarded railway crossing. The system is done using microcontroller and IR sensor. In this proposed system the arrival of the train is detected by the IR sensor and sends the signal to the microcontroller. On this basis the gate crossing is controlled by the microcontroller. The drawback of the proposed system is its less consistent. The use of IR sensor is not visible in daylight and open field. [12], this system is proposed by a cost efficient method for enhancing the quality of the railway gate level crossing. The system is proposed of ATmega 16A microcontroller and IR sensor which is placed to notice the coming and leaving of the train through which the opening and closing of the gate is operated. [13], The system is proposed using detectors, GPS and GSM. The GSM and GPS are combined together, the GPS is the tracking system and the GSM is a modem to close the gate at the railway level crossing. The detectors used to detect the coming and leaving of the train and transmit the signal to the upcoming railway level crossings. This system concludes that it is flexible and accurate.[14], this system tries to operate the opening and closing of railway gate at railway level crossing. The system uses IR sensors to notice the coming and leaving of the train and uses a raspberry pi board to control the opening and closing of railway gates. [15], The proposed system presents an automated railway gate controller at railway level crossing to prevent the accidents. The system uses vibration sensors to operate the opening and closing of railway gates. An IR sensor is placed to detect the moving objects on the rail tracks and the status of the rail track will be sent to the control room using wireless communication. The main drawback of the system is it is costly and difficult to implement because of more requirements. [16], This system is proposed for the multiple track railway crossing which has a controller which receives the signals by the train by the sensor. The signals contain the information of the train including its identity. Through the information the railway gates will be opened and closed. The drawback of this system is it is costly.

3. Proposed System

Our Proposed System is a practically working system. Our idea is very simple and effective. The idea is to close the railway level crossing gates automatically and to open them automatically, during the time of train's arrival and departure respectively. Automated concept is to reduce the number of accidents with less manpower. In our system, we are placing ultrasonic sensors near the railway tracks. Ultrasonic sensors are used in this system, because it has a very high range of 4 meters (which is better than other sensors). At a certain distance before the level crossing and after the level crossing, these ultrasonic sensors are placed. The reason for sensor placement is to sense, both the train's arrival and departure correctly and effectively. As soon as the train reaches the 1st sensor which is been placed before the level crossing, senses or detects the train, it sends a message to the Arduino connected, and then the buzzer will be turned on automatically so that the road users will be able to know that the train is nearer to the crossing and they can wait till the train passes by. As soon as the buzzer sound starts, the servo motor connected with the level crossing gates will close them automatically. The reason to use servo motor in our system is that it is working is based on Angular Rotation which means that at first the gates will be at 90° which is open, at then during the time of train's arrival the gates will be at 0° (closed), and after the train passes by it will return its original position which is 90°(opened). The proper working of the level crossing gates are because of the attached servo motor (with angular rotation). In case if we have used other motors there would have been a problem in opening and closing of gates because they lack angular rotation. This servo motor helps the gates to come back to normal position(90°) from closed position(0°) instead of going into the ground (270°). As soon as the train passes the second sensor, placed at a certain distance after the level crossing gates, buzzer sound will be turned off itself and the gates will open automatically, and then the road users can use the level crossing road safely. The second Ultrasonic sensor is placed a little far, comparatively higher distance than the distance between 1st Ultrasonic sensor and the level crossing gates(because of the train's length). This whole design is connected with the Arduino Nano ATMEGA 328p, which has a code uploaded in it before the whole process. Code is the main key to work the whole system. Though the idea and working of our system is simple, it's usage will be more effective. It will definitely reduce the railway level crossing accidents and will ensure people's safety figure 1 and figure 2.

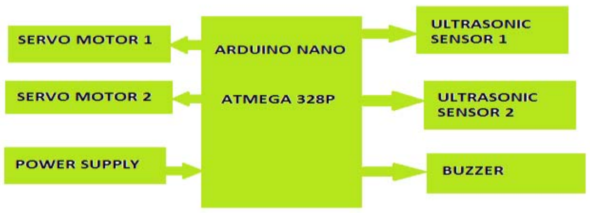


Figure 1. Block Diagram

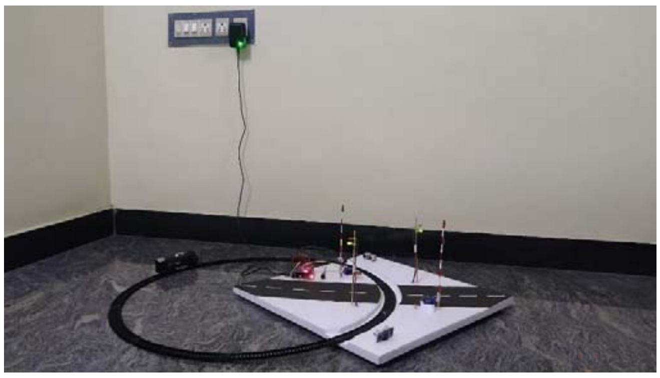


Figure 2. Visual representation of our system

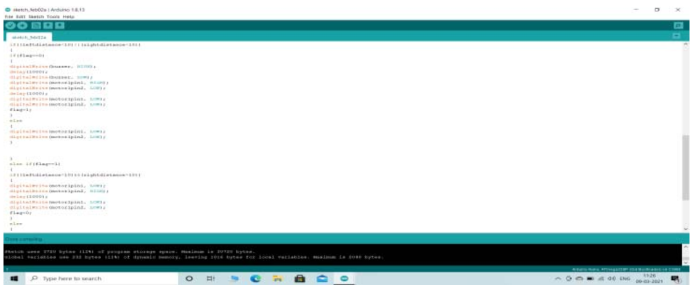
4. Components Required

1. Arduino Nano
2. Ultrasonic sensor
3. Servo motor
4. Buzzer
5. LED green
6. LED red
7. Toy train
8. Joining wires

9. Power supply

5. Implementation and Result

Our idea has been used and developed as a working model. A Railway track of diameter 60cm has been fixed. And the level crossing gate is setup containing two gates facing each other with a gap and the gates are fixed with LED lights and the servo motor has been fixed at the gates, an important point is that the tracks should be in between the level crossing setup. The distance between the level crossing gates is 20cm and the length of the road is 19cm. And the Ultrasonic sensors have been placed before and after the level crossing gates at a distance of about 20cm. And sensors are placed at a distance of about 6cm on each side of the track. The whole setup has been connected with the Arduino nano ATMEGA328p. Buzzer is placed near the Arduino. The Arduino has been connected to the external power supply. After the whole setup is ready, a toy train is fixed on the track. Then the toy-train starts running with the help of batteries. To start the process, power supply is switched on. After that when we turn on the toy train, it starts running, and when the train comes nearer to the 1st Ultrasonic Sensor, the crossing gates will be closed, and if it reaches the 2nd one the gates will open. The servo motor attached with the gates, which has Angular Rotation, helps the system with both the opening and closing of the gates. Like the gates, Buzzer will also be turned on automatically as soon as the train reaches 1st sensor and will be turned off when the train reaches 2nd one. During the whole process, the power supply should be turned on. And before all these operations the code for the whole process should be uploaded in the Arduino. The code can be transferred to Arduino with the help of Transmission cable, with the help of that the Arduino can be attached to the computer or Laptop easily. The code can be uploaded and reset easily. Arduino IDE software is used to upload the code in the computer to the Arduino Figure 3.



```
if((leftdistance<10)||(rightdistance<10))
{
if(flag==0)
{
digitalWrite(buzzer, HIGH);
delay(1000);
digitalWrite(buzzer, LOW);
digitalWrite(motor1pin1, HIGH);
digitalWrite(motor1pin2, LOW);
delay(1000);
digitalWrite(motor1pin1, LOW);
digitalWrite(motor1pin2, LOW);
flag=1;
}
else
{
digitalWrite(motor1pin1, LOW);
digitalWrite(motor1pin2, LOW);
}


}
else if(flag==1)
{
if((leftdistance>10)&&(rightdistance>10))
{
digitalWrite(motor1pin1, LOW);
digitalWrite(motor1pin2, HIGH);
delay(1000);
digitalWrite(motor1pin1, LOW);
digitalWrite(motor1pin2, LOW);
flag=0;
}
else
{
```

```
Done compiling.
Sketch uses 3720 bytes (12%) of program storage space. Maximum is 30720 bytes.
Global variables use 232 bytes (11%) of dynamic memory, leaving 1816 bytes for local variables. Maximum is 2048 bytes.
```

Figure 3. Output (code uploaded in the Arduino)

6. Conclusion

This paper we presented is based on Automated Railway Gate opening system using Arduino. Technologies like these have been already done but still they haven't implemented yet and in process especially in India and in some other countries. We have studied it thoroughly and made a working model for the same, but with some modifications. We have found that as compared to the existing system which have been made already, our system works much efficiently and it is reliable because the whole system is automated. As far as now from the number of accidents occurred and still counting, a proper, safe and durable system is needed. Therefore, to avoid these kind of accidents in the future, we have implemented a new system with ultrasonic sensors. The system has been tested and it is working perfectly in all atmospheric conditions without any flaws. Finally, we conclude by saying that the number of 4 accidents will be reduced and many innocent people's lives could be saved by using this system. Our system will be new as we are using a fully automated system and there are further more ideas on which we can develop this system for the future generation.

References

[1] P.K.Kumar and B.S.ShivaShankara, PLC Based Automatic Fault Detection of Railway Track and Accidence Avoidance System, *International Journal of Engineering Research and General Science*, ISSN 2091-2730, **3**, Issue 2, March-April 2015.

[2] N.Bhargav, A.Gupta, M.Khirwar, S.Yadav and V.Sahu, Automatic Fault Detection of Railway Track System Based on PLC (ADOR TAST), *International Journal of Recent Research Aspects*, ISSN : 2349-7688, **3**, Issue 1, March 2016.

[3] S, D., & H, A. (2019). AODV Route Discovery and Route Maintenance in MANETs. 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS). doi:10.1109/icaccs.2019.8728456

[4] H. Anandakumar and K. Umamaheswari, An Efficient Optimized Handover in Cognitive Radio Networks using Cooperative Spectrum Sensing, Intelligent Automation & Soft Computing, pp. 1–8, Sep. 2017. doi:10.1080/10798587.2017.1364931

[5] M.Prabhakar and A.Ashok Babu, Sensor Based Train Collision Identification and Avoidance Systems, *International Journal & Magazine of Engineering, Technology, Management and Research*, ISSN : 2348-4845 , **3**, Issue 10, October 2016.

[6] T.Dhanabalu, S.Sugumar, S.Surya Prakash and A.VijayAnand, Sensor based identification system for Train Collision Avoidance, *IEEE sponsored 2nd International Conference on Innovations In Information Embedded and Communication Systems (ICIIECS)*, **3**, 2015.

[7] M.Ganapathi and G.Priyanka, Smart System for Train Crash Avoidance, *International Journal of Innovative Technologies*, ISSN : 2321- 8665, **4**, Issue 11, August 2016.

[8] Naga Hema Kumari.V and China Appala Naidu.R, Train Collision Avoidance by Using Sensors,"*International Journal of Advanced Research in Computer Science and Software Engineering*, **6**, Issue 6, June 2016.

[9] R.Gopinathan and B.Sivashankar, PLC based railway level crossing gate control, *International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)*. ISSN: 0976-1353, **8** Issue 1–April 2014.

[10] Pradeep Raj, Increasing accidents in the unmanned level crossing of the railways, 2012.

[11] Xishi Wang, Ning Bin, and Cheng Yinhang, A new micro-processor based approach to an automatic control system. *International Symposium on Industrial Electronics*, pp. 842-843, 1992.

[12] Ahmed Salih Mahdi, Al-Zuhairi, Automatic Railway Gate and Crossing Control based Sensors & Microcontroller *International Journal of Computer Trends and Technology (IJCTT)* –**4** Issue 7–July 2013

[13] Sheikh Shanawaz Mostafa, Md. Mahbub Hossian, Khondker Jahid Reza, Gazi Maniur Rashid, A Radio Based Intelligent Railway Crossing System to Avoid Collision. *IJCSI International Journal in Computer Science Issues*, **Vol 7**, Issue 6, November 2010. ISSN: 1694-0914.

[14] Acy M. Kottalil1, Abhijith S, Ajmal M M, Abhilash L J, Ajith Babu, Automatic Railway Gate Control System, *International Journal in Advanced Research in Electrical, Electronics and Instrumentation Engineering*, **Vol. 3**, Issue 2, February 2014.

[15] K. Vidyasagar, P. Sekhar Babu, R. RamPrasad, Anti Collision and Secured Level Crossing System. *International Journal of Computer Applications (0975 – 8887),* **107** – No 3, December 2014.

[16] Karthik Krishnamurthi, Monica Bobby, Vidya V, Edwin Baby. Sensor Based Automatic Railway Gates Control, *International Journal in Advanced Research in Computer Science Engineering and Technology (IJARCET),* **4** Issue 2, and February 2015.

Appendix B: Sample Code

Admin Terminal GUI

```
from tkinter import *
from datetime import *
from tkinter import messagebox
from selenium import webdriver
import sqlite3
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities
from selenium.common.exceptions import StaleElementReferenceException, NoSuchElementException,
TimeoutException
import traincodeextract as tc
from selenium.webdriver.common.by import By
import serial
root = Tk()
root.geometry("1920x1080")
root.title("AUTOMATED RAILWAY CROSSING")

global incorrect_id_label,password_error_label
today = date.today()
d2 = today.strftime("%B %d,%Y")

def change():
        b1.config(image=bimg2, state=DISABLED)
        id = e1.get()
        p1 = e2.get()
        print(id, p1)
        if id != 'IR0897':
                root.after(1000,show_id_error)
        elif p1 != '123456':    # Replace 'password' with the correct password
                root.after(1000,show_password_error)
        else:
                incorrect_id_label.config(text="")
                password_error_label.config(text="")
                root.after(2000,show_frame1)
#frame1
global mainpg,bimg,bimg2,bimg3,fr3,op2,cl2,ex
```

```
mainpg = PhotoImage(file = 'main3.png')
bimg = PhotoImage(file = 'CTA.png')
bimg2 = PhotoImage(file = 'load.png')
bimg3 = PhotoImage(file = 'continue.png')
fr3 = PhotoImage(file = 'nframe3.png')
op2 = PhotoImage(file = 'open2.png')
cl2 = PhotoImage(file = 'close2.png')
ex = PhotoImage(file = 'exit 1.png')

login_f=Frame(root)
login_f.pack()
login_c=Canvas(login_f,height=1080,width=1920,bg='black')
login_c.pack(fill = "both",expand=True)
login_c.create_image( 0, 0,image =mainpg,anchor=NW)

e1= Entry(login_c,width=25,bg="#e4e4e4",font=("Times", 25))
e1.place(x=820,y=295)
e1.config(highlightthickness=2, highlightcolor="blue")
'''t1=Text(login_c,width=50,height=2,bg="#e4e4e4")
t1.place(x=820,y=320)'''
e2= Entry(login_c,text='Enter Password',width=25,font=("Times", "25"),bg="#e4e4e4",show="*")
e2.config(highlightthickness=2, highlightcolor="blue")
e2.place(x=820,y=395)
b1=Button(login_c,image=bimg,borderwidth=0,bg='#FFFFFF',cursor='hand2',command=change)
b1.place(x=860,y=492)
c_v1=IntVar(value=0)

def my_show():
	if(c_v1.get()==1):
		e2.config(show='')
	else:
		e2.config(show='*')

c1 = Checkbutton(login_c,text='Show Password',variable=c_v1,
onvalue=1,offvalue=0,command=my_show,borderwidth=0,bg='#FFFFFF',font=("Sans",10),cursor='hand2
')
c1.place(x=820,y=462)

def show_password_error():
	e2.delete(0, END)
	b1.config(image=bimg,state=NORMAL)
	e2.config(highlightthickness=2, highlightbackground="red", highlightcolor="red")
	password_error_label.place(x=820, y=442)
```

```
def show_id_error():
        e1.delete(0, END)
        b1.config(image=bimg,state=NORMAL)
        e1.config(highlightthickness=2, highlightbackground="red", highlightcolor="red")
        incorrect_id_label.place(x=820, y=342)

incorrect_id_label = Label(login_c, text="Incorrect ID", fg="red",bg="white")
password_error_label = Label(login_c, text="Incorrect password", fg="red",bg='white')

def force():
        messagebox.showwarning("Warning","CLOSING")
        arduino = serial.Serial('com3', 115200)##locks the port
        print('Established serial connection to Arduino')
        cmd="FCLOSE"
        time.sleep(2)
        arduino.write(cmd.encode())
        time.sleep(2)
        arduino.close()
def opening():
        messagebox.showwarning("Warning","OPENING")
        arduino = serial.Serial('com3', 115200)##locks the port
        print('Established serial connection to Arduino')
        cmd="FOPEN"
        time.sleep(2)
        arduino.write(cmd.encode())
        time.sleep(2)
        arduino.close()
def des():
        root.after(2000)
        root.destroy()

def show_frame1():
        global frame2,acer
        login_f.pack_forget()    # Hide the login frame
        frame2 = Frame(root)
        frame2.pack()
        acer = Canvas(frame2, height=1080, width=1920, bg='black')
        acer.pack(fill="both", expand=True)
        id_error_img = PhotoImage(file='Home.png')
        acer.image = id_error_img    # Keep a reference to the image
        acer.create_image(0, 0, image=id_error_img, anchor=NW)

        e3= Entry(acer,width=18,bg="#e4e4e4",font=("Times", 25))
        e3.config(highlightthickness=2, highlightcolor="blue")
```

```
    e3.place(x=170,y=415)
    e4= Entry(acer,width=18,font=("Times", "25"),bg="#e4e4e4")
    e4.config(highlightthickness=2, highlightcolor="blue")
    e4.place(x=570,y=415)
    def bclick():
        tc1=e3.get()
        tc2=e4.get()
        mainexec(tc1,tc2)
        frame2.destroy()
        acer.destroy()

        frame3=Frame(root)
        frame3.pack()
        f3=Canvas(frame3,height=1080,width=1920,bg='black')
        f3.pack(fill = "both",expand=True)
        f3.create_image( 0, 0,image =fr3,anchor=NW)

        #dbms
        # Connect to the database
        conn = sqlite3.connect('C:\\Users\\hp\\Train.db')     # Replace with the path to your database
file
        # Create a cursor
        cursor = conn.cursor()

        def db():
            now = datetime.now()
            cur = now.strftime("%H:%M")
            sql="select Traincode,Arrival,Station from details where Arrival>'{}'".format(cur)
            cursor.execute(sql)
            p=cursor.fetchall()
            dl1= Label(f3,width=10,font=("Times",10),bg="#FFFFFF",text=d2)
            dl1.place(x=65,y=40)
            try:
                l1= Label(f3,width=25,font=("Times", 18),bg="#e6f2ff",text=p[0][0])
                l1.place(x=1200,y=130)
                l12= Label(f3,width=25,font=("Times", 18),bg="#e6f2ff",text=p[0][1])
                l12.place(x=1200,y=180)
                l13= Label(f3,width=25,font=("Times", 18),bg="#e6f2ff",text=p[0][2])
                l13.place(x=1200,y=230)
            except:
                l1= Label(f3,width=25,font=("Times", 18),bg="#e6f2ff",text='')
                l1.place(x=1200,y=130)
                l12= Label(f3,width=25,font=("Times", 18),bg="#e6f2ff",text='NO TRAIN')
                l12.place(x=1200,y=180)
                l13= Label(f3,width=25,font=("Times", 18),bg="#e6f2ff",text='')
                l13.place(x=1200,y=230)
            try:
                l2= Label(f3,text=(p[1][0]),width=10,font=("Times", "17"),bg='#FFFFFF')
                l2.place(x=880,y=350)
                l21= Label(f3,text=(p[1][1]),width=10,font=("Times", "17"),bg='#FFFFFF')
                l21.place(x=1050,y=350)
                l22= Label(f3,text=(p[1][2]),width=30,font=("Times", "17"),bg='#FFFFFF')
                l22.place(x=1200,y=350)
            except:
                l2= Label(f3,text='',width=10,font=("Times", "17"),bg='#FFFFFF')
                l2.place(x=880,y=350)
                l21= Label(f3,text='NO TRAIN',width=10,font=("Times", "17"),bg='#FFFFFF')
                l21.place(x=1050,y=350)
                l22= Label(f3,text='',width=30,font=("Times", "17"),bg='#FFFFFF')
                l22.place(x=1200,y=350)
            try:

                l3= Label(f3,text=p[2][0],width=10,font=("Times", "17"),bg='#FFFFFF')
                l3.place(x=880,y=460)
                l31= Label(f3,text=p[2][1],width=10,font=("Times", "17"),bg='#FFFFFF')
                l31.place(x=1050,y=460)
                l32= Label(f3,text=p[2][2],width=30,font=("Times", "17"),bg='#FFFFFF')
                l32.place(x=1200,y=460)
            except:
                l3= Label(f3,text='',width=10,font=("Times", "17"),bg='#FFFFFF')
                l3.place(x=880,y=460)
                l31= Label(f3,text='NO TRAIN',width=10,font=("Times", "17"),bg='#FFFFFF')
                l31.place(x=1050,y=460)
                l32= Label(f3,text='',width=30,font=("Times", "17"),bg='#FFFFFF')
                l32.place(x=1200,y=460)
            try:
                l4= Label(f3,text=p[3][0],width=10,font=("Times", "17"),bg='#FFFFFF')
                l4.place(x=880,y=570)
                l41= Label(f3,text=p[3][1],width=10,font=("Times", "17"),bg='#FFFFFF')
                l41.place(x=1050,y=570)
                l42= Label(f3,text=p[3][2],width=30,font=("Times", "17"),bg='#FFFFFF')
                l42.place(x=1200,y=570)
            except:
                l4= Label(f3,text='',width=10,font=("Times", "17"),bg='#FFFFFF')
                l4.place(x=880,y=570)
                l41= Label(f3,text='NO TRAIN',width=10,font=("Times", "17"),bg='#FFFFFF')
                l41.place(x=1050,y=570)
                l42= Label(f3,text='',width=30,font=("Times", "17"),bg='#FFFFFF')
```

```
                l42.place(x=1200,y=570)
            root.after(5000,db)


        #calling func to show dbms
        db()

        s1=tc1.upper()
        s2=tc2.upper()

        ls1= Label(f3,text=s1,width=10,font=("Times", "20"),bg="#e6f2ff")
        ls1.place(x=100,y=190)
        ls2= Label(f3,text=s2,width=10,font=("Times", "20"),bg="#e6f2ff")
        ls2.place(x=550,y=190)

        b1=Button(f3,image=op2,borderwidth=0,bg='#FFFFFF',cursor='hand2',command=opening)
        b1.place(x=150,y=331)
        b2=Button(f3,image=cl2,borderwidth=0,bg='#FFFFFF',cursor='hand2',command=force)
        b2.place(x=500,y=331)
        b3=Button(f3,image=ex,borderwidth=0,bg='#FFFFFF',cursor='hand2',command=des)
        b3.place(x=1450,y=20)

    b2=Button(acer,image=bimg3,borderwidth=0,bg='#FFFFFF',cursor='hand2',command=bclick)
    b2.place(x=348,y=500)
    l1=Label(acer,text=d2,width=8,bg='#FFFFFF')
    l1.place(x=900,y=50)

"""from_code=input("Enter code of station one:")
to_code=input("Enter code of station two:")"""
def scrape_details(train_number,firststat):
    # Load the webpage
    options = Options()
    capa = DesiredCapabilities.CHROME
    capa["pageLoadStrategy"] = "none"
    options.add_argument('--ignore-certificate-errors')
    options.add_argument('--ignore-ssl-errors')
    options.headless = False    # Run the browser in headless mode,slower
    # Set up the Selenium driver (Replace the path with your own chromedriver path)
    driver_path = 'C:\\Users\\hp\\Downloads\\chromedriver.exe'
    service = Service(driver_path)
    driver = webdriver.Chrome(service=service, options=options,desired_capabilities=capa)
    wait = WebDriverWait(driver,20)
    # Connect to the database
    conn = sqlite3.connect('C:\\Users\\hp\\Train.db')    # Replace with the path to your database file
```

```
# Create a cursor
cursor = conn.cursor()
#cursor.execute('DROP TABLE details')
url = f"https://www.railyatri.in/live-train-status/{train_number}"
driver.get(url)
try:
        wait.until(EC.presence_of_element_located((By.CLASS_NAME, 'table-responsive')))
        driver.execute_script("window.stop();")
except(TimeoutException):
        print("ERROR DETECTED")
        return -1
else:
# Always drop existing tables if any
# Create a table if it doesn't exist
        cursor.execute('''CREATE TABLE IF NOT EXISTS details (
                                        Traincode TEXT,
                                        Station TEXT,
                                        Arrival TEXT,
                                        [Train Status] TEXT,
                                        [Halt Time] TEXT,
                                        Platform TEXT,
                                        [Locomotive Reverse] TEXT
                                )''')
        # Find the parent element containing the table
        parent_element = driver.find_element("class name", "table-responsive")
        # Find all the table rows within the parent element
        rows = parent_element.find_elements("tag name", "tr")
        # Iterate over the rows to extract and insert the details
        for row in rows:
        # Find the columns within the row
                columns = row.find_elements("tag name", "td")
                if len(columns) >= 6:
                # Retrieve the text of each column
                        station = columns[0].text
                        arrival = columns[1].text
                        status = columns[2].text
                        halt_time = columns[3].text
                        platform = columns[4].text
                        loco_reverse = columns[5].text
                        str1=(firststat.upper())[1:]
                        str2=station.upper()
                        if(str1 in str2):
                                # Insert the detail into the database
```

```
                                        cursor.execute("INSERT INTO details (Traincode,Station, Arrival, [Train Status],
[Halt Time], Platform, [Locomotive Reverse]) VALUES (?, ?, ?, ?, ?, ?, ?)", (train_number, station, arrival,
status, halt_time, platform, loco_reverse))
                                        print('done')
                                        conn.commit()
                                        return 1
                # Commit the transaction
                # Close the cursor and connection
        cursor.close()
        conn.close()
def mainexec(tc1,tc2):
        tcodes=tc.traincode(tc1,tc2)
        testc=5
        print(tcodes)
        #modify testcaccording to no of train values needed
        for i in tcodes:
                if(testc==0):
                        break
                else:
                        status=0
                        while(status!=1):
                                chances=5
                                status=scrape_details(i,tcodes[i])
                                if(status!=1):
                                        chances=chances-1
                                if(chances==0):
                                        break
                        testc-=1
# Quit the browser
```

Appendix C: CO-PO And CO-PSO Mapping

COURSE OUTCOMES:

After completion of the course the student will be able to

SL. NO	**DESCRIPTION**	**Blooms' Taxonomy Level**
CO1	Identify technically and economically feasible problems (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO2	Identify and survey the relevant literature for getting exposed to related solutions and get familiarized with software development processes (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO3	Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions of minimal complexity by using modern tools & advanced programming techniques (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO4	Prepare technical report and deliver presentation (Cognitive Knowledge Level: Apply)	Level 3: Apply
CO5	Apply engineering and management principles to achieve the goal of the project (Cognitive Knowledge Level: Apply)	Level 3: Apply

CO-PO AND CO-PSO MAPPING

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO3
CO1	3	3	3	3		2	2	3	2	2	2	3	2	2	2
CO2	3	3	3	3	3	2		3	2	3	2	3	2	2	2
CO3	3	3	3	3	3	2	2	3	2	2	2	3			2
CO4	2	3	2	2	2			3	3	3	2	3	2	2	2
CO5	3	3	3	2	2	2	2	3	2		2	3	2	2	2

3/2/1: high/medium/low

JUSTIFICATIONS FOR CO-PO MAPPING

MAPPING	LOW/ MEDIUM/ HIGH	JUSTIFICATION
100003/CS622T.1-PO1	**HIGH**	Identify technically and economically feasible problems by applying the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS622T.1-PO2	**HIGH**	Identify technically and economically feasible problems by analysing complex engineering problems reaching substantiated conclusions using first principles of mathematics.
100003/CS622T.1-PO3	**HIGH**	Design solutions for complex engineering problems by identifying technically and economically feasible problems.
100003/CS622T.1-PO4	**HIGH**	Identify technically and economically feasible problems by analysis and interpretation of data.
100003/CS622T.1-PO6	**MEDIUM**	Responsibilities relevant to the professional engineering practice by identifying the problem.
100003/CS622T.1-PO7	**MEDIUM**	Identify technically and economically feasible problems by understanding the impact of the professional engineering solutions.
100003/CS622T.1-PO8	**HIGH**	Apply ethical principles and commit to professional ethics to identify technically and economically feasible problems.
100003/CS622T.1-PO9	**MEDIUM**	Identify technically and economically feasible problems by working as a team.
100003/CS622T.1-PO10	**MEDIUM**	Communicate effectively with the engineering community by identifying technically and economically feasible problems.
100003/CS622T.1-P011	**MEDIUM**	Demonstrate knowledge and understanding of engineering and management principles by selecting the technically and economically feasible problems.
100003/CS622T.1-PO12	**HIGH**	Identify technically and economically feasible problems for long term learning.
100003/CS622T.1-PSO1	**MEDIUM**	Ability to identify, analyze and design solutions to identify technically and economically feasible problems.
100003/CS622T.1-PSO2	**MEDIUM**	By designing algorithms and applying standard practices in software project development and Identifying technically and economically feasible problems.
100003/CS622T.1-PSO3	**MEDIUM**	Fundamentals of computer science in competitive research can be applied to Identify technically and economically feasible problems.
100003/CS622T.2-PO1	**HIGH**	Identify and survey the relevant by applying the knowledge of mathematics, science, engineering fundamentals.

100003/CS622T.2-PO2	**HIGH**	Identify, formulate, review research literature, and analyze complex engineering problems get familiarized with software development processes.
100003/CS622T.2-PO3	**HIGH**	Design solutions for complex engineering problems and design based on the relevant literature.
100003/CS622T.2-PO4	**HIGH**	Use research-based knowledge including design of experiments based on relevant literature.
100003/CS622T.2-PO5	**HIGH**	Identify and survey the relevant literature for getting exposed to related solutions and get familiarized with software development processes by using modern tools.
100003/CS622T.2-PO6	**MEDIUM**	Create, select, and apply appropriate techniques, resources, by identifying and surveying the relevant literature.
100003/CS622T.2-PO8	**HIGH**	Apply ethical principles and commit to professional ethics based on the relevant literature.
100003/CS622T.2-PO9	**MEDIUM**	Identify and survey the relevant literature as a team.
100003/CS622T.2-PO10	**HIGH**	Identify and survey the relevant literature for a good communication to the engineering fraternity.
100003/CS622T.2-PO11	**MEDIUM**	Identify and survey the relevant literature to demonstrate knowledge and understanding of engineering and management principles.
100003/CS622T.2-PO12	**HIGH**	Identify and survey the relevant literature for independent and lifelong learning.
100003/CS622T.2-PSO1	**MEDIUM**	Design solutions for complex engineering problems by Identifying and survey the relevant literature.
100003/CS622T.2-PSO2	**MEDIUM**	Identify and survey the relevant literature for acquiring programming efficiency by designing algorithms and applying standard practices.
100003/CS622T.2-PSO3	**MEDIUM**	Identify and survey the relevant literature to apply the fundamentals of computer science in competitive research.
100003/CS622T.3-PO1	**HIGH**	Perform requirement analysis, identify design methodologies by using modern tools & advanced programming techniques and by applying the knowledge of mathematics, science, engineering fundamentals.
100003/CS622T.3-PO2	**HIGH**	Identify, formulate, review research literature for requirement analysis, identify design methodologies and develop adaptable & reusable solutions.

100003/CS622T.3-PO3	**HIGH**	Design solutions for complex engineering problems and perform requirement analysis, identify design methodologies.
100003/CS622T.3-PO4	**HIGH**	Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
100003/CS622T.3-PO5	**HIGH**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools.
100003/CS622T.3-PO6	**MEDIUM**	Perform requirement analysis, identify design methodologies and assess societal, health, safety, legal, and cultural issues.
100003/CS622T.3-PO7	**MEDIUM**	Understand the impact of the professional engineering solutions in societal and environmental contexts and Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions.
100003/CS622T.3-PO8	**HIGH**	Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions by applying ethical principles and commit to professional ethics.
100003/CS622T.3-PO9	**MEDIUM**	Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.
100003/CS622T.3-PO10	**MEDIUM**	Communicate effectively with the engineering community and with society at large to perform requirement analysis, identify design methodologies.
100003/CS622T.3-PO11	**MEDIUM**	Demonstrate knowledge and understanding of engineering requirement analysis by identifying design methodologies.
100003/CS622T.3-PO12	**HIGH**	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change by analysis, identify design methodologies and develop adaptable & reusable solutions.
100003/CS622T.3-PSO3	**MEDIUM**	The ability to apply the fundamentals of computer science in competitive research and prior to that perform requirement analysis, identify design methodologies.
100003/CS622T.4-PO1	**MEDIUM**	Prepare technical report and deliver presentation by applying the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS622T.4-PO2	**HIGH**	Identify, formulate, review research literature, and analyze complex engineering problems by preparing technical report and deliver presentation.

100003/CS622T.4-PO3	**MEDIUM**	Prepare Design solutions for complex engineering problems and create technical report and deliver presentation.
100003/CS622T.4-PO4	**MEDIUM**	Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions and prepare technical report and deliver presentation.
100003/CS622T.4-PO5	**MEDIUM**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools and Prepare technical report and deliver presentation.
100003/CS622T.4-PO8	**HIGH**	Prepare technical report and deliver presentation by applying ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
100003/CS622T.4-PO9	**HIGH**	Prepare technical report and deliver presentation effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.
100003/CS622T.4-PO10	**HIGH**	Communicate effectively with the engineering community and with society at large by prepare technical report and deliver presentation.
100003/CS622T.4-PO11	**MEDIUM**	Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work by prepare technical report and deliver presentation.
100003/CS622T.4-PO12	**HIGH**	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change by prepare technical report and deliver presentation.
100003/CS622T.4-PSO1	**MEDIUM**	Prepare a technical report and deliver presentation to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas.
100003/CS622T.4-PSO2	**MEDIUM**	To acquire programming efficiency by designing algorithms and applying standard practices in software project development and to prepare technical report and deliver presentation.
100003/CS622T.4-PSO3	**MEDIUM**	To apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs by preparing technical report and deliver presentation.
100003/CS622T.5-PO1	**HIGH**	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
100003/CS622T.5-PO2	**HIGH**	Identify, formulate, review research literature, and analyze complex engineering problems by applying engineering and management principles to achieve the goal of the project.

100003/CS622T.5-PO3	**HIGH**	Apply engineering and management principles to achieve the goal of the project and to design solutions for complex engineering problems and design system components or processes that meet the specified needs.
100003/CS622T.5-PO4	**MEDIUM**	Apply engineering and management principles to achieve the goal of the project and use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
100003/CS622T.5-PO5	**MEDIUM**	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO6	**MEDIUM**	Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities by applying engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO7	**MEDIUM**	Understand the impact of the professional engineering solutions in societal and environmental contexts, and apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO8	**HIGH**	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice and to use the engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO9	**MEDIUM**	Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO11	**MEDIUM**	Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team. Manage projects in multidisciplinary environments and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PO12	**HIGH**	Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PSO1	**MEDIUM**	The ability to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas. Apply engineering and management principles to achieve the goal of the project.

100003/CS622T.5-PSO2	**MEDIUM**	The ability to acquire programming efficiency by designing algorithms and applying standard practices in software project development to deliver quality software products meeting the demands of the industry and to apply engineering and management principles to achieve the goal of the project.
100003/CS622T.5-PSO3	**MEDIUM**	The ability to apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs thereby evolving as an eminent researcher and entrepreneur and apply engineering and management principles to achieve the goal of the project.