Software Design Specification

Skill Development Project III – ICT 3206

Bachelor of Information and Communication Technology (Honors)

Department of Information and Communication Technology
Faculty of Technology
Rajarata University of Sri Lanka

Details of the Project

Project Title : Train Sync

Group Number : Group 14

Group Name : Team ECODUINO

Submission Date : 06/21/2025

Details of the Group Members

Name	Registration ID	Index No.	Signature
S.A.S.N Samarakkodi	ITT/2021/090	1711	
B.R.C.M Bulugalla	ITT/2021/017	1639	
C.I.D Fernando	ITT/2021/032	1654	
H.M.D.M Hearth	ITT/2021/039	1661	
A.M.G.N.G Kalpadeepa	ITT/2021/054	1675	
H.M.S.N Herath	ITT/2021/040	1662	

Details of Supervisor(s)

Name : Mr. Husni Mohamed

Designation : Lecturer(T)

Department/ Unit/ Institute : Department of Information and Communication Technology

Contact Details : (+94)76 638 3023

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Background of the project

Although there are still issues with safety, efficiency, and the passenger experience, railway travel is still an essential form of transportation for millions of people. Delays and safety issues get worse by problems including frequent platform or crossing accidents, illegal access, and a lack of real-time train tracking. For these problems we founded a solution.

By combining automation, secure access management, and real-time monitoring, the Automated Train Platform Safety and Tracking System tackles these problems. The system offers real-time GPS tracking and precise ETA estimations by utilizing technologies like the ESP32, GSM modules, ultrasonic sensors, and QR code-based authentication. In the end, this makes railway travel more intelligent, secure, and effective by improving passenger convenience, preventing unwanted access, enhancing safety at railway crossings, and lowering the possibility of platform accidents.

1.2 Purpose and significance of the project

Railway transportation is a vital part of public travel, but problems like delays, accidents, and safety concerns often affect passengers' experience. The Automated Train Platform Safety and Tracking System addresses these challenges by using modern technology such as ESP32 microcontrollers, GSM modules, ultrasonic sensors, and QR code scanning.

The system improves safety by installing automatic gates at railway crossings that close when a train is approaching, preventing vehicle-train collisions. On platforms, automated gates only open once the train has completely stopped, stopping passengers from boarding too early and reducing accidents. Additionally, the system provides real-time train location tracking and accurate Estimated Time of Arrival (ETA), so passengers can plan their trips better and avoid unexpected delays. To enhance security and control access, QR code authentication ensures that only passengers with valid tickets for 1st and 2nd class compartments can board those sections, preventing overcrowding and unauthorized entry. Altogether, this system improves passenger safety, increases efficiency, and makes railway travel more reliable and secure for everyone.

1.3 Scope of the project

The Automated Train Platform Safety and Tracking System integrates automation, real-time tracking, and access control to increase railway efficiency and safety. The scope of the project outlines the system's features and limitations.

The Automated Train Platform Safety and Tracking System, which aims to increase railroad efficiency and safety, is made up of several essential parts. The automated platform gate system ensures that platform gates remain closed until a train has fully entered and halted at the station, preventing mistakes from being made by hurrying passengers. The system also features an automated railway crossing barrier system that shuts down railway crossing gates when a train approaches, thereby reducing auto accidents. The technology improves the traveler experience by using real-time GPS tracking and ETA computation. Passengers can obtain accurate train location and expected arrival times via web interface. Additionally, a QR code-based boarding system restricts access to first- and second-class compartments, allowing only authorized passengers to enter reserved cabins. To facilitate effective data interchange and control, the system makes use of flask, GSM modules, and ESP32 microcontrollers. This makes it possible for all components to be automated and updated in real time.

While significantly enhancing railway efficiency and safety, our system has limitations. Due to the system's dependence on mobile networks (Dialog, Mobitel) for GPS and control data transmission, tracking updates may be delayed in remote areas with insufficient network coverage. Due to budgetary constraints, the project uses low-cost GPS and QR scanning modules, which may result in worse performance and less accuracy than more costly alternatives. Furthermore, automatic platform gates will only be implemented in large train stations because smaller rural stations might not have the capacity to handle them. The QR code-based boarding system is only applicable to first- and second-class passengers; public areas will not be subject to restricted access control.

1.4 Objectives of the Project

The main objective of this project is to use automation and technology to improve the safety, efficiency, and dependability of railway transportation systems. The following are the project's additional objectives:

1. To develop a platform gate that will open when the train arrives at the station and either goes very slowly or stops entirely.

2. To build an automated railway crossing gate control system that will guarantee efficient train railway cross road management and prevent collisions with automobiles.

3. A GPS tracking system that uses a website to provide passengers with the location of trains in real time.

4. To develop an ETA system, or estimated time of arrival, that will display the passengers' ETA on a webpage.

5. To develop a door entry system that uses QR codes for quick and safe train boarding.

6. To develop a website to booking train seats and give an e – ticket through E-mail.

7. To use Flask and Firebase for backend and tracking.

1.5 System design approach

The Waterfall Model, a phased, sequential software development strategy, was chosen because of its systematic, strict approach, which is ideal for our well-defined project requirements. Its systematic methodology ensures that each step is completed in a systematic manner, minimizing complexity and making sure the system satisfies all safety and functional standards. The waterfall module will be added to our system in the way described below.

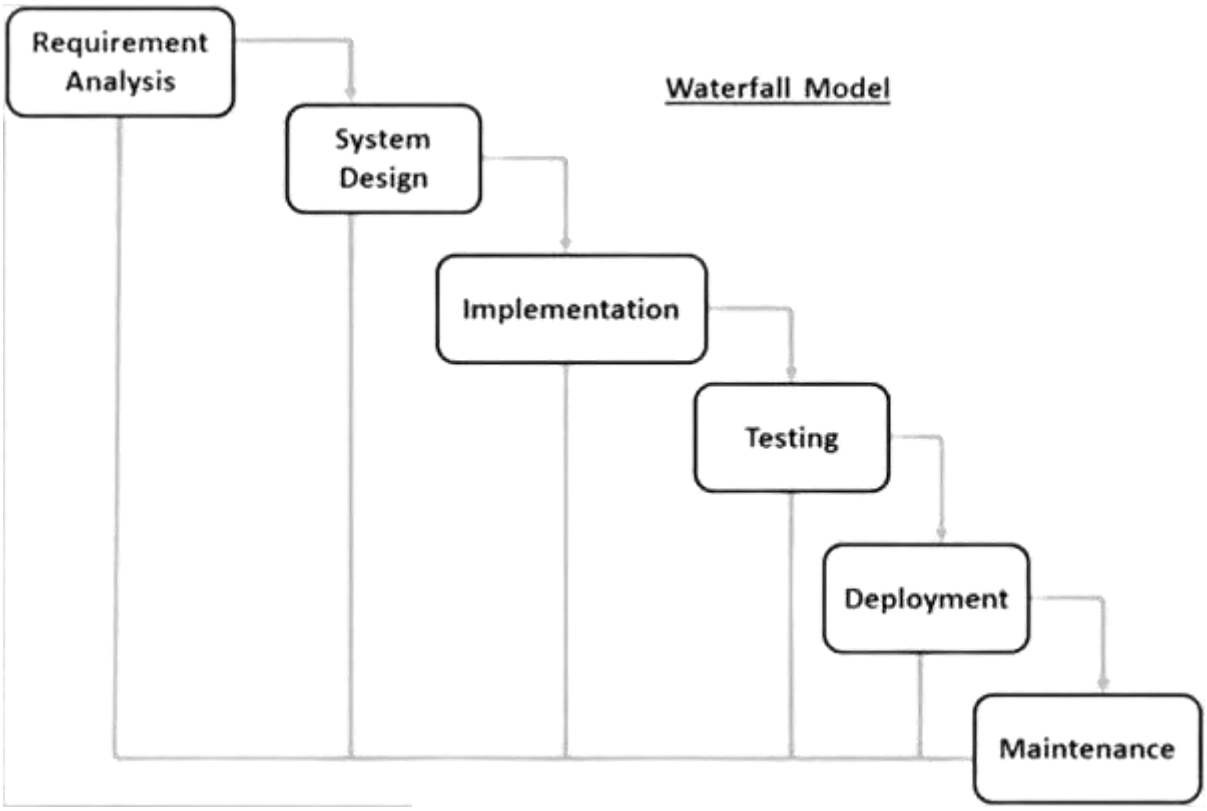


Figure 1 Waterfall model

1. Requirement Analysis

Our TrainSync System contains a number of features. Prior to conducting the requirement analysis, we determined and defined the primary features of our system as below,

- Platform gates are implemented and have automatic opening mechanisms.
- An automated system for opening and closing railway crossing gates.
- ETA and GPS monitoring system.
- Integrated boarding system with QR codes.

The next stage is gathering the circuit and sensor requirements for our system. The ESP32 Dev Kit is our system's central brain. This inexpensive microcontroller, which has Bluetooth and Wi-Fi built in, is frequently utilized in smart systems, automation, and Internet of Things projects. It is utilized in our project to transmit data to the website and Firebase for the tracking procedure. Next the NEO-6M is a widely used GPS module in IoT applications, providing accurate location and timing data. In our project, it is used for the GPS tracking system. Servo motors are efficient rotary actuators commonly used in robotics and automation for precise control of angular position. In our project, they are used in both the platform gate opening system and the railway crossroad gate mechanism. Next one is the ESP32-CAM. This is a low-cost microcontroller with built-in Wi-Fi, Bluetooth, and a camera module, making it ideal for image processing, surveillance, and smart security systems. In our project, it is used to detect vehicles at railway crossings through real-time image capture.

Figure 2 Esp 32 Module

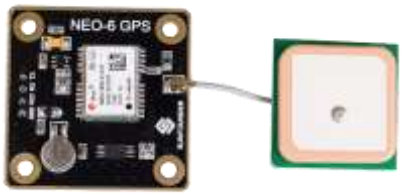


Figure 3 Neo - 6M Module

Figure 5 Esp 32 CAM Module

Figure 4 Ultrasonic Sensor

2. System Design

System design is the process of organizing the architecture, parts, and connections of a system. It involves creating complex designs that act as a plan of action for the high-level system structure as well as particular modules.

Design the platform gate mechanism with sensors and control logic. Servo motors in this part will pull the gates to the ground in order to open them. When the train is identified by ultrasonic sensors and the GPS indicates that it has stopped traveling on the website, the servo motors will start up.

Plan the closing of the railway crossroad gates in reaction to an approaching train. Here, we have set up the system such that when the ultrasonic sensor detects the train, the gates will close till the other ultrasonic sensor confirms that the train is leaved. Since a gate can open and close even if a bird flies past it if there is only ultrasonic sensor detection, we go with time-based detection.

Using GPS tracking modules and a 4G router dongle to establish an internet connection, the GPS tracking system will be constructed in order to send the data to the firebase via the ESP 32 module.

As part of the QR-based boarding system, which will only be implemented in the cabin doors, the first-class and second-class reservation doors will open automatically when the QR code scanning module in our case an ESP 32 CAM module reads them.

3. Implementation and Testing

In the implementation phase of the Waterfall model, each module's code is written, and the system is developed to comply with the design. The following step is testing, which finds and fixes any bugs or problems to ensure the system operates as planned.

Use IOT technology to accurately control GPS data transmission, open and close platform and railroad cross road gates autonomously, and implement a QR-based boarding system using an ESP32 CAM Module. and create the web interface and flask backend for managing GPS tracking and QR codes. In this case, Firebase also will serve as our database, HTML, CSS, and JavaScript will be utilized to create the web interface, and QR code scanning will be used to open train doors. Here, the IOT part will be coded using the Arduino IDE, the QR generating portion will be coded using Python.

Finish the unit testing by install and test individual hardware components, including motors, sensors, and communication modules, and confirm that platform gates are functioning properly in response to train detection. Verify that railroad crossing gates are promptly actuated and deactivated. Verify the GPS tracking and ETA predictions' accuracy. Make sure the QR code system is secure and that only the right codes can open doors.

4. Deployment of the system

We are currently working on a project prototype. Taking into consideration the deployment for a real rail stops, at rail stations, install the platform gate system. Install railroad crossing barriers and incorporate sensors and train detection systems into them. Start the passenger GPS tracking web application. On specified train routes, introduce the QR code boarding system. Train railroad employees and inform travelers about the new systems. **Phased deployment**, which entails installing the system gradually over time, is the deployment type used for this technology. By allowing for the independent testing and improvement of every function, including platform gates, railway crossroad gates, GPS tracking, and QR code boarding, this approach lowers risks and guarantees seamless operation. It is ideal because it minimizes the impact of potential issues and makes gradual adaption easier for both travelers and train workers.

Deployment phases according to phase deployment,

- Phase 1: Involves installing platform gates at specific stations.
- Phase 2: Railway barriers in locations with heavy traffic.
- Phase 3: GPS tracking along a designated path.
- Phase 4: Some trains will allow QR code boarding.

2 Architectural Design

2.1 System Architecture

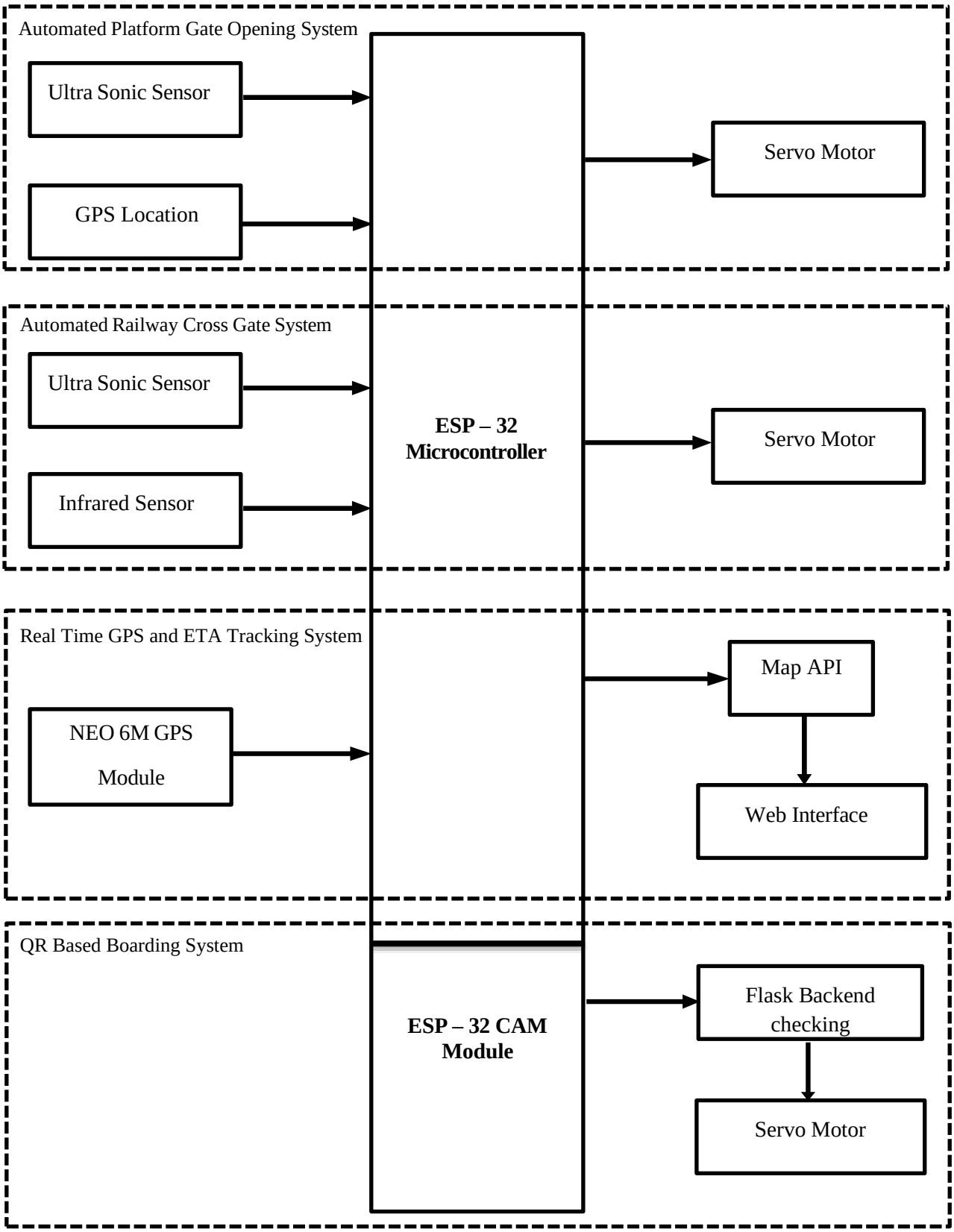


Figure 6 Hardware Architectural Design

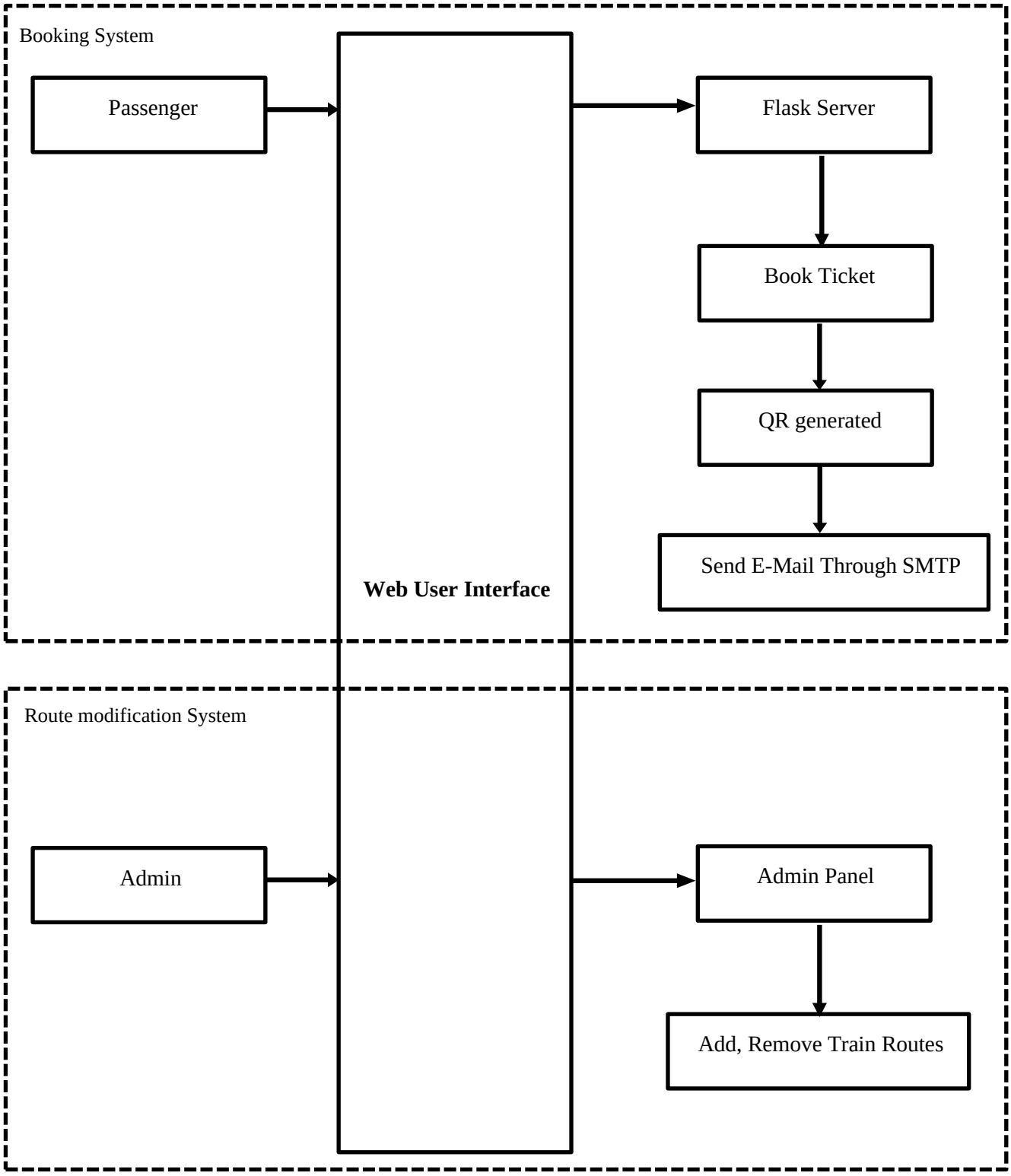


Figure 7 Software Architectural Design

1. Automated Platform Gate Opening System

   The system uses servo motors to open and close platform gates. These gates remain shut until the train has stopped entirely, entered the station, or is moving very slowly. While GPS data from the train helps in verifying its condition, ultrasonic sensors are used to identify the train. To enable safe boarding, the servo motors activate and open the gates.

2. Automated Railway Crossroad System

   The system uses servo motors that are managed to automate railway crossing gates. When an approaching train is detected by ultrasonic sensors near the crossing, the system will automatically close the gates if the sensor detects an object for five seconds. When the train exits the railway Cross road and the gate opens when the ultrasonic sensor detects that the train has left.

3. Real Time GPS and ETA Tracking System

   The location data that the NEO-6M GPS Module continuously collects from the moving train is sent to a central database through the4G router dongle. When this data is analyzed and displayed on a web interface, passengers are provided with precise information regarding the train's location and anticipated arrival time.

4. QR Based Boarding System

   After reading the QR code using the ESP-32 CAM Module, sends the code to confirm its accuracy and determine whether it is present in the database. Access to the reserved seating zones is made possible by the system sending a signal to servo motors to unlock the inside doors of the first- or second-class trains if the QR code is valid.

5. Booking System

   After interacting with the user interface (UI) in the booking system, the passenger selects a date and destination. The system then uses a GET method to retrieve the trains from the flask server and an SMTP server to deliver the passenger their booking e-ticket via email after successfully done the payment.

6. Route modification System

   Since the administrator has their own admin panel, they can add or delete trains from routes after logging in with the username and password that the Sri Lankan railway department has provided.

2.2 Component design

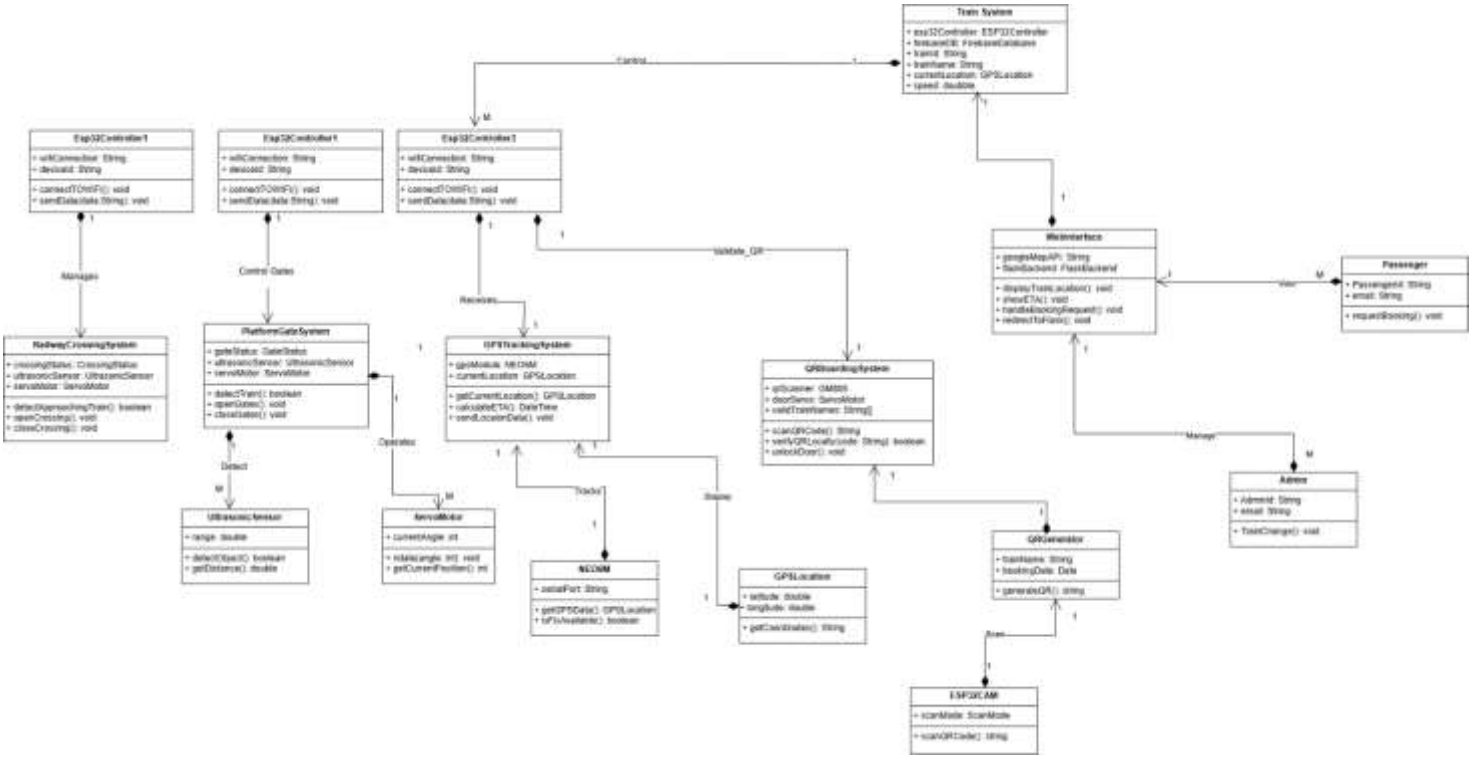


Figure 8 Class Diagram of the System

Drive Link of the Class Diagram

The diagram shows a fully integrated smart train automation system that includes IoT devices, real-time tracking, automated gate control, and QR-based passenger boarding. The TrainSystem is at the center of the system, controlled by an ESP32Controller and linked to a FirebaseDatabase for storing and retrieving real-time train data. Each Train includes attributes such as ID, current location, and speed, and it can update its location and calculate ETA using the GPSTrackingSystem, which is based on a NEO6M GPS module and returns data via the GPSLocation class. The RailwayCrossingSystem and PlatformGateSystem manage safety systems, with UltrasonicSensors detecting incoming trains and ServoMotors opening or closing gates as needed. The system also includes a smart passenger involvement layer, which allows passengers to order tickets using a WebInterface that communicates with a FlaskBackend to manage bookings, produce QR codes with the QRGenerator, and deliver them via an EmailService. At the station, the QRBoardingSystem, which includes an ESP32CAM, scans and validates QR codes to securely grant or restrict entrance by controlling door servos. Overall, the system provides a safe, automated, and real-time solution for train operations by

integrating embedded devices, cloud databases, web technologies, and sensor-actuator logic to improve efficiency and passenger experience.

2.3 Processes and interaction design

1. Sequence Diagram of Platform Gate System

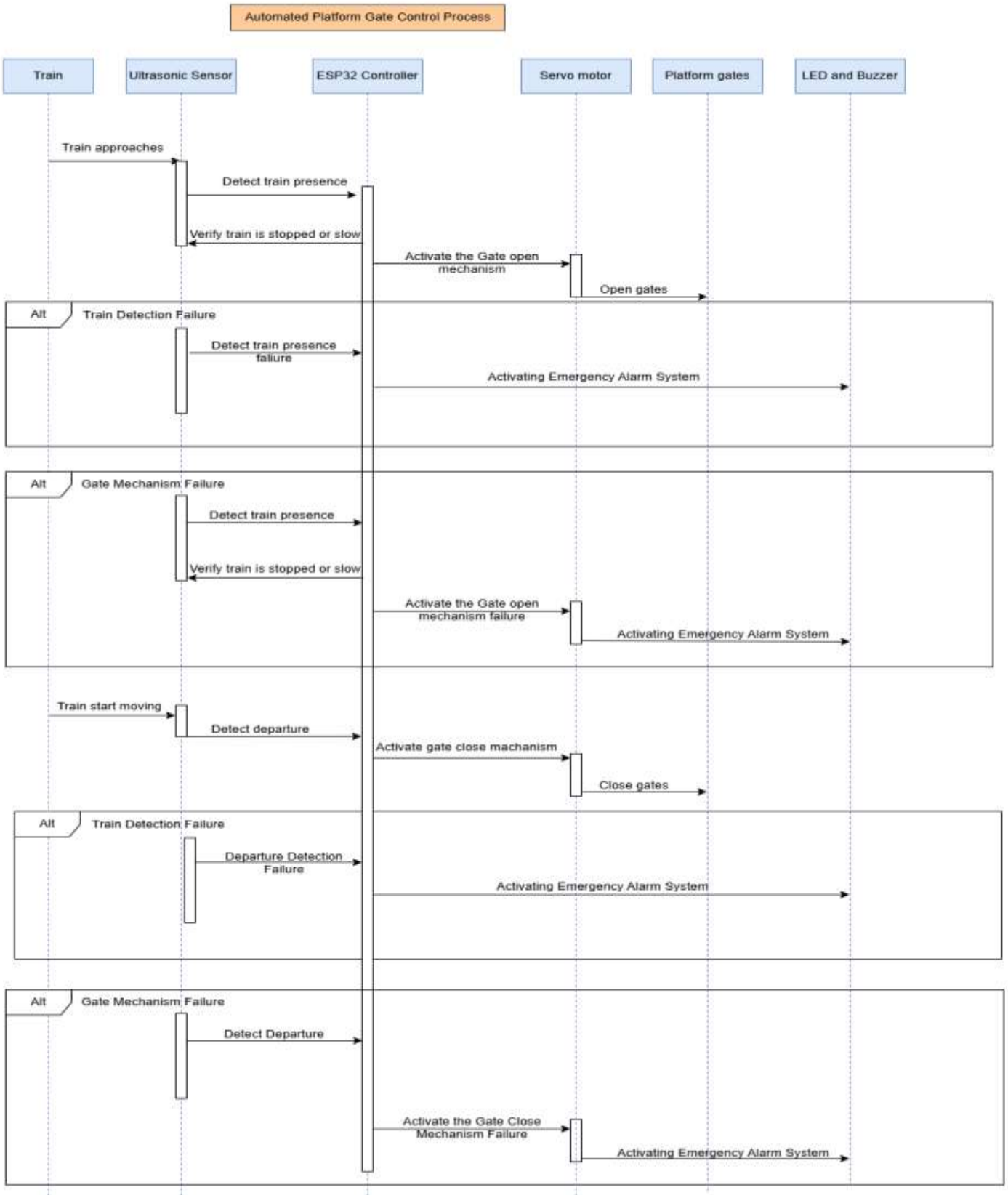


Figure 9 Sequence Diagram of Platform Gate System

2. Sequence Diagram for Railway Crossroad System

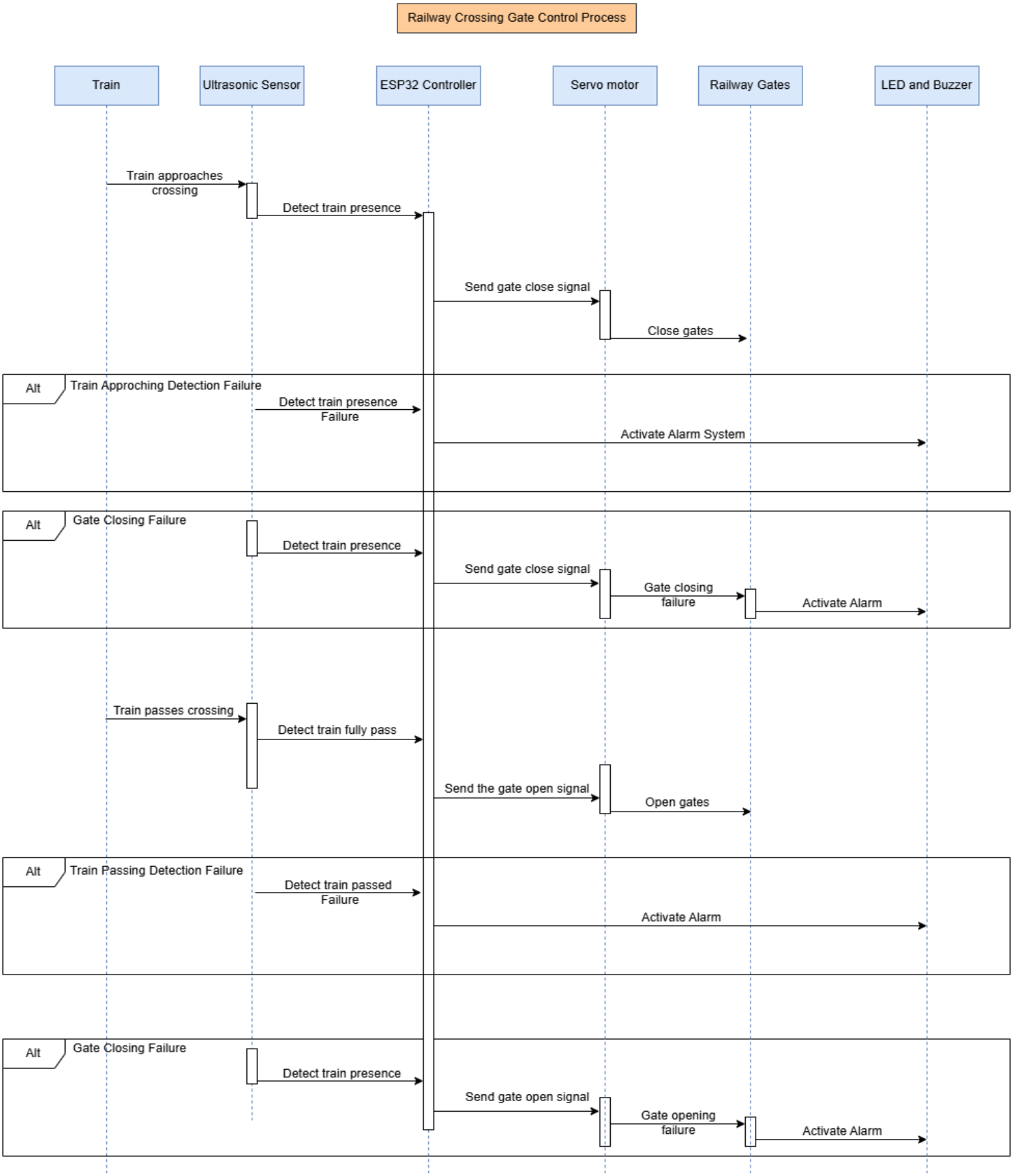


Figure 10 Sequence Diagram of Railway Crossing Gate System

3. GPS and ETA System

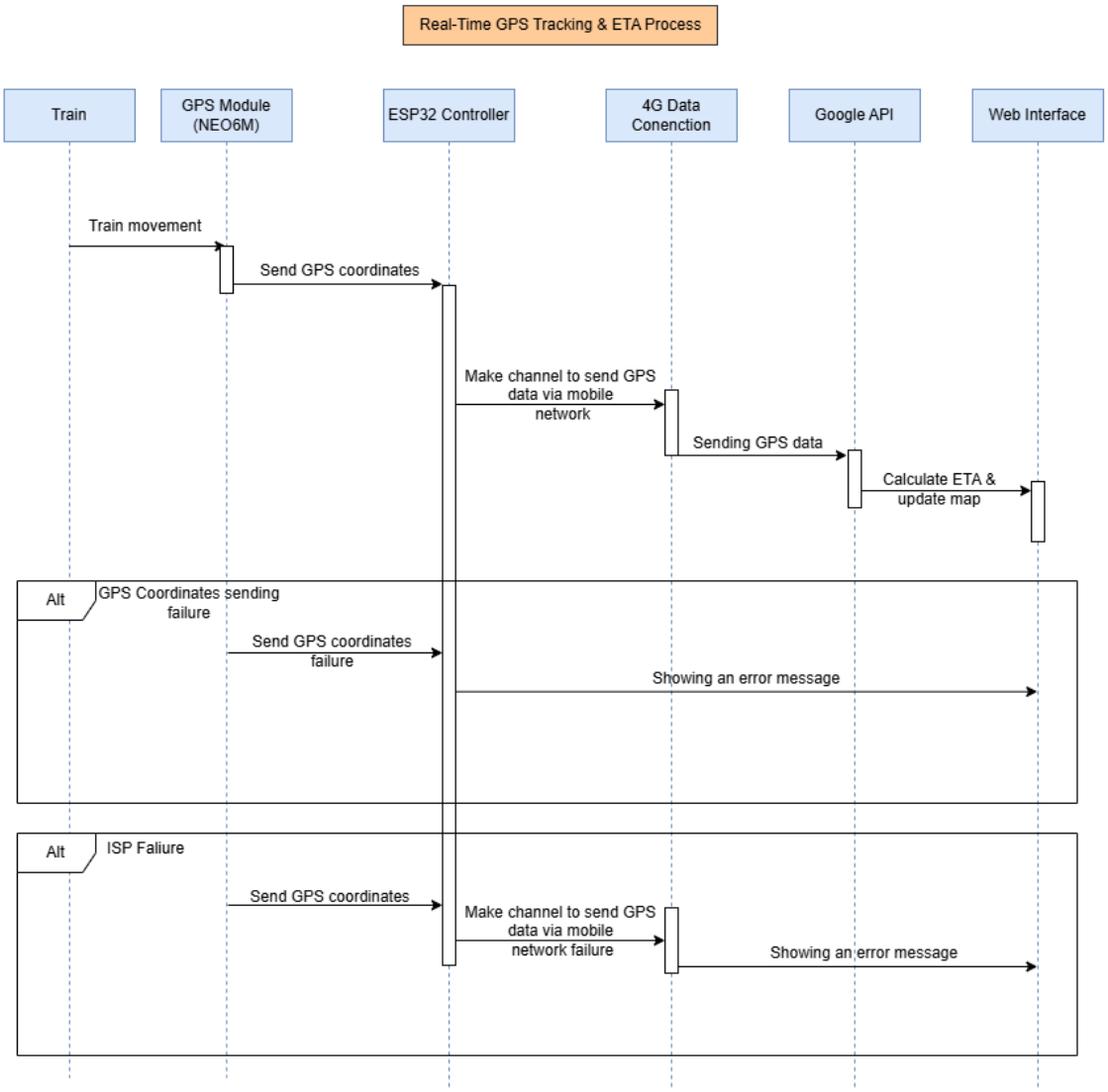


Figure 11 Sequence Diagram of GPS and ETA System

4. QR Based Boarding System

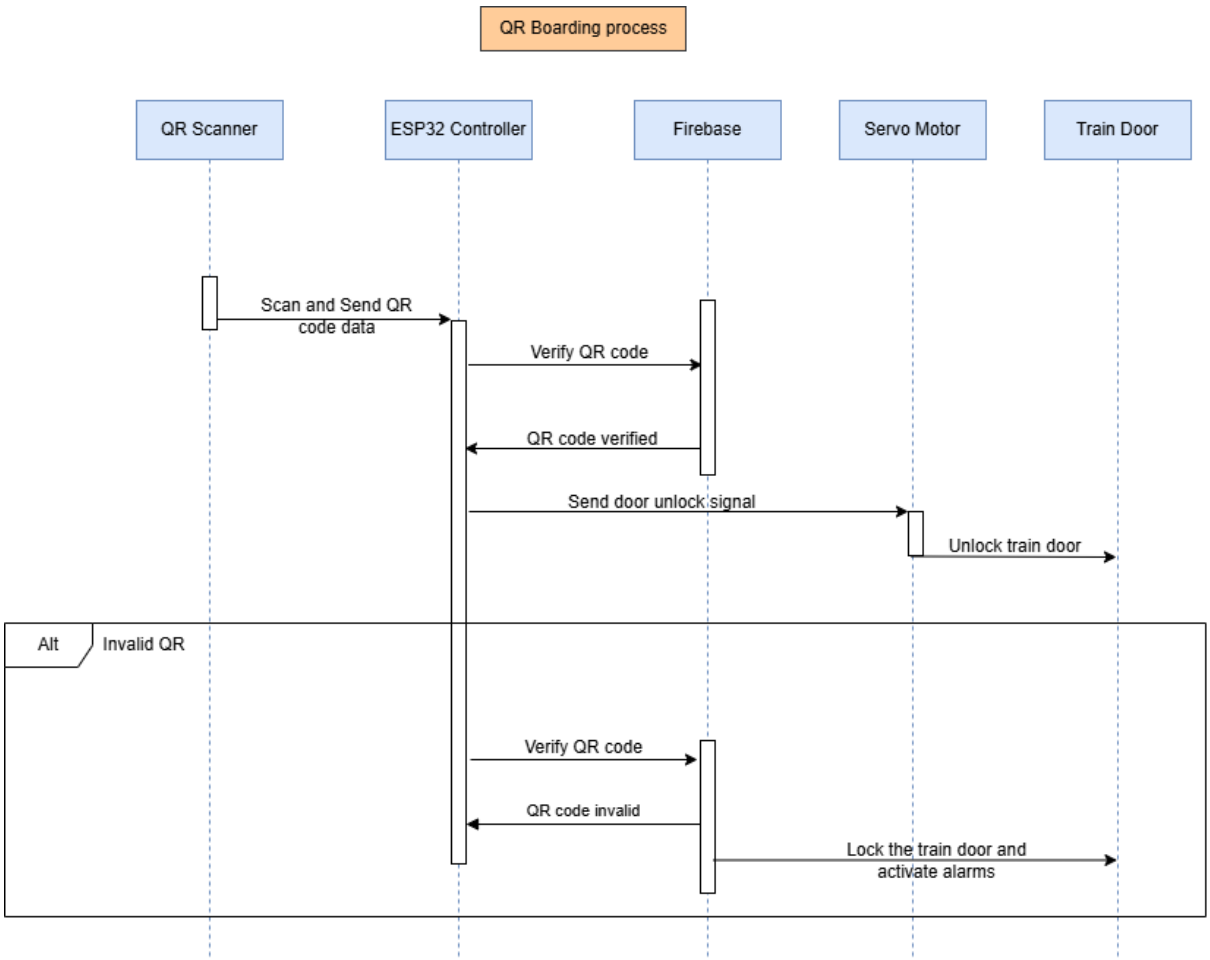


Figure 12 Sequence Diagram of QR Boarding System

5. Online Ticket Booking and QR Generation Process

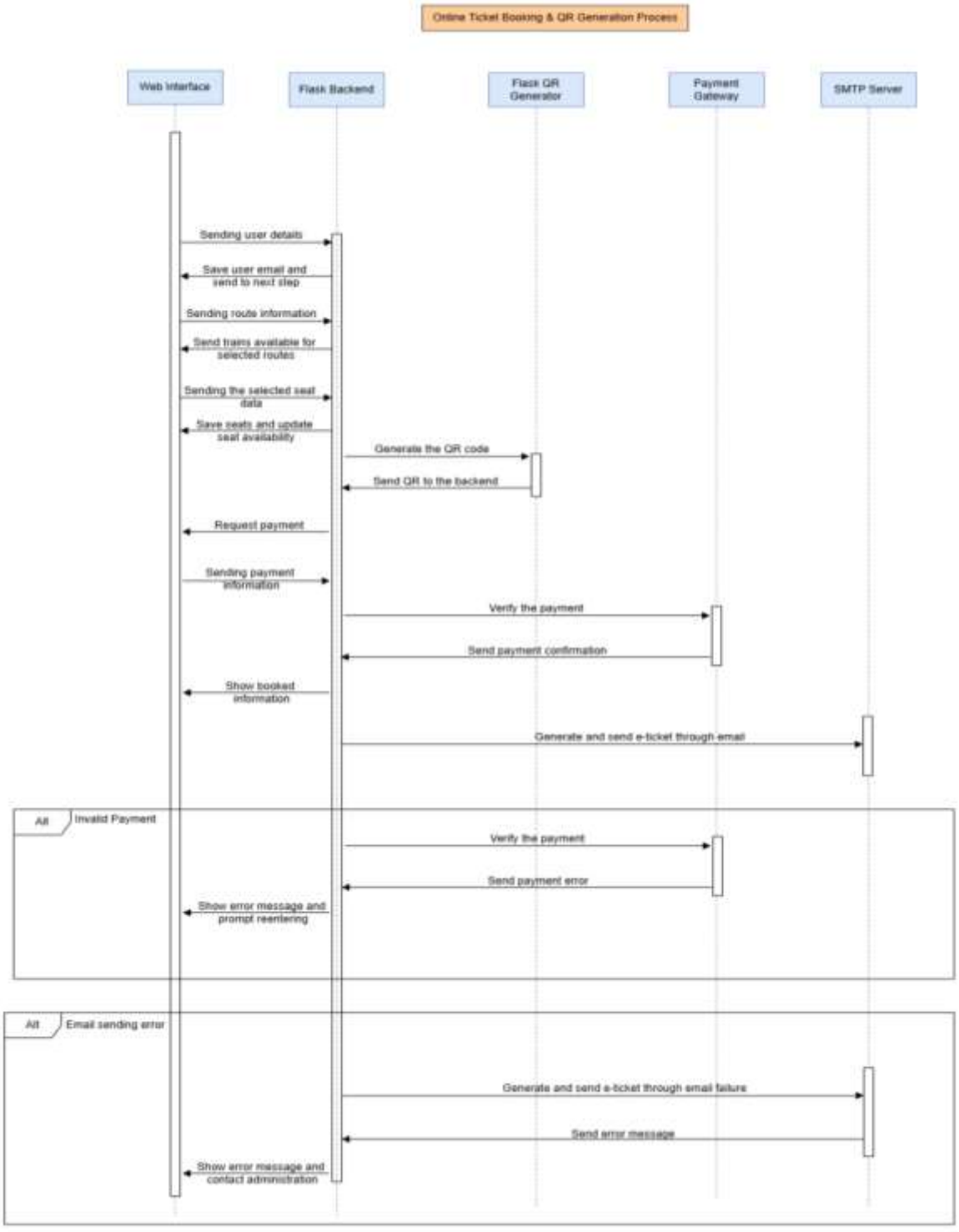


Figure 13 Sequence Diagram of Online Booking System

Drive Link of the above Diagram

2.4 Tools, libraries, special algorithms and implementation environment

A variety of hardware and software tools and technologies are integrated into the Automated Train Platform Safety and Tracking System's implementation that ensure automation, secure access management, and real-time tracking. Cost-effectiveness, reliability, and simplicity of integration were the main factors in the tool selection.

1. Tools and Frameworks

- Flask

  The main backend web framework for creating RESTful APIs and controlling backend functions including database administration, email notifications, booking management, and QR code generation is Flask. It provides a lightweight, flexible method that works well for modular design and quick prototyping.

- Firebase Realtime Database

  Train GPS tracking in real time is the only use case for Firebase. The web frontend uses the Firebase JavaScript SDK to get the location data that the ESP32 microcontroller transmits to Firebase. This makes it possible to visualize locations in real time and integrate with Google Maps.

- HTML, CSS, JavaScript

  These are used in the development of the admin and passenger user interface. Booking, train location, and ETA pages are all available on the frontend.

2. Libraries

- Libraries that use in flask

  Os – Interact with the operation system.

  Json – Handles JSON data.

  Copy – Allows object copying.

  Flask – Web framework to build APIs/web apps.

  Flask_mail – Sends mail through flask.

  Qrcode – Generates QR codes.

  Io.BytesIO – Memory byte stream

  Dotenv – Loads environment variables from .env.

  Flask_cors – Handles Cross-Origin Resource Sharing.

  Reportlab – creates PDFs Programmically.

- Libraries that use in circuit or Arduino part

  Arduino.h – Core functions for Arduino (ex: digitalwrite, delay).

  WiFi.h – Connect ESP32 to Wi-Fi networks.

  ESP32QRCodeReader.h – Scans and reads QR codes using a camera.

  ESP32Servo.h – Controls servo motors with ESP32.

  ArduinoJson.h – Parses and creates JSON data.

  Esp_camera.h – controls theESP32-CAM module’s camera.

3. Special Algorithms

- Qr code Generation in the flask - After a reservation, Flask uses qrcode to create a unique QR code.make() using the booking information.

- Sending a Email to the User – Through SMTP server the Flask backend will send email to the user using the email that we supply in the .env witch is the environment variable file.

- Real time train location - The NEO-6M module provides GPS data to the ESP32. It provides Firebase with the current latitude and longitude every few seconds. The Firebase SDK is used by the web application to retrieve this data.

- Train Detections – Use ultrasonic sensors to detect the train and avoid false detection by make the sensor detect an object for 5 seconds to confirm It is a train

4. Development Tools

    - Vs Code – Visual Studio Code is use for develop and coding the website of our project it includes a main.html, admin.html, booking.html and tracking.html. In here main.html is the home page and other three files will connect by buttons.

    - Arduino IDE – ESP 32 and ESP 32 CAM boards will be programmed using this Arduino IDE.

    - Python IDLE – Use to code the backend. So, in here all the programming for the flask backend that the functions like QR generation and Sending email will be done using this software.

3 Interface Design

3.1 PACT (People, Activities, Contexts, Technologies) analysis of the system

The PACT (People, Activities, Contexts, Technologies) framework is a human-centered design method that evaluates how interactive systems can be developed to effectively support its users, the jobs they achieve, and the contexts in which they operate. In this part, we use the PACT framework to evaluate our project Trainsync, ensuring that both software and hardware elements are in compliance with user requirements and practical situations. This ensures that our system is accessible, functional, dependable, and efficient for all types of users and operating settings.

1. People

   Understanding the end-users and stakeholders is essential to developing useable solutions. Our system serves a wide range of users, from regular travelers to administrative workers and automated hardware agents. These groups have a variety of needs and technological knowledge.

   - Passenger

     Passengers are the main users of the system. They use the user interface to do important operations like checking train schedules, reserving tickets, and scanning QR codes for boarding. The user group includes everyday workers who expect rapid, intuitive, and dependable services, as well as tourists and periodic visitors who may need assistance and clarification in their interactions.

     To accommodate this diversity, the system must deliver a responsive, mobile-first design with clear instructions, minimal user effort, and automation wherever possible (for example, email-based QR delivery rather than requiring account logins), and also clear visual cues and confirmation messages after key actions such as booking, QR generation, or tracking.

- Administrative

  Administrators, such as railway control officers and station officials, help users navigate the system. Their responsibilities include managing train timetables and route configurations, monitoring bookings and authenticating passenger QR codes when scanners fail, overriding the system during emergencies or sensor failures, and manually opening or closing gates as an alternative.

  Admin users are often better skilled and knowledgeable about technology than passengers. However, they require a secure and user-friendly admin panel, dashboards for easily monitoring train routes, error logs and notification alerts via LED systems and buzzers in the case of a system failure, and tools for manual ticket validation during system downtime.

- Automated System Components

  The system incorporates hardware components (e.g., ESP32, GPS, GSM, ultrasonic sensors) that respond to programmed logic and environmental inputs. These non-human agents are included in the PACT framework because they interact with the system environment and influence user experiences indirectly. For example, ultrasonic sensors detect the presence of a train and activate gates, GPS modules communicate location data to Firebase via ESP 32 for real-time tracking, and QR scanning modules (ESP 32 CAM) for boarding access.

  These devices must be reliable under different environmental circumstances (rain, light, vibrations), low-latency, and high-accuracy, because they directly affect safety-critical functions such as gate opening or train detection. Easily maintained and tested during hardware inspections or upgrades.

2. Activities

The system supports a wide range of actions performed by humans and machines. Each action is assigned to a user role and is intended to be efficient and error-free.

- Passenger Activities

Passengers participate in a multi-step journey that includes both digital and physical responsibilities, such as searching for trains, users browse available trains by route, class, and timing, and then booking tickets, this technology enables seat selection, class selection, fare calculation, and payment confirmation. Receiving QR Code, which A QR code is generated and emailed after a successful booking. It is then included in a nicely designed PDF ticket. Tracking trains, which Passengers can check the train's real-time location and ETA using Google Maps integration, which allows for improved scheduling. Scanning QR Code for Boarding Witch Users present the QR code to the station's scanner. If correct, the servo motor opens the train door. Finally, in the event of a failure (for example, an invalid QR code), users must be given clear error feedback and routed to station staff.

- Admin Activities

Admins are responsible for more backend responsibilities, such as maintaining train data, which allows them to add and delete train schedules, prices, and seat availability. If sensor data is absent or corrupted, perform an emergency override by manually operating platform gates with control switches. When a sensor or actuator is damaged, replacement is also a responsibility.

- Automation Activities

Autonomous activities occur in hardware components, such as ESP32, which analyzes ultrasonic sensor readings to detect trains and when the detection time exceeds 5 seconds, it considers the detected object to be a train and GPS data is communicated to Firebase every few seconds over 4G connection through ESP 32. QR scanners read the input and verify that the ticket is correct. When access is granted, servo motors are triggered.

3. Context

The physical context includes the real-world surroundings in which the system's hardware components and users are located. These include railway platforms, train compartments, railroad crossings, and station waiting areas.

- Physical Context

  Railway platforms are frequently semi-covered outdoor environments subjected to dust, rain, heavy foot traffic, and changing lighting conditions. To avoid false triggering, ultrasonic sensors and servo motors used for platform gate automation must be weather-resistant, durable, and positioned at the appropriate height. Power backup systems may be required to ensure that gate automation continues to work effectively even during outages.

  QR code scanning systems are installed at the entry doors to first- and second-class compartments. QR scanners and ESP32 modules must be securely attached and verified to ensure stability during train travel. Servos that unlock compartment doors must be silent, rapid, and in rhythm with the Flask server's response.

  Crossings are outdoor road-rail crossings that are subject to heavy traffic, pedestrians, and adverse weather conditions. Ultrasonic sensors at crossings must detect oncoming trains without being mistakenly activated by passing vehicles or animals. Because of the restricted amount of human intervention, servo motors should run railway crossing gates efficiently and with little maintenance.

- Technological Context

  The technological context describes the technical environment in which the system's software and hardware components operate. This includes internet access, device compatibility, backend infrastructure, and data synchronization.

  The system's essential functionality is dependent on reliable mobile network access. Specifically, GPS data from the ESP32 is transmitted to Firebase via the Dialog 4G connection. Mobile service is limited in locations ranging from Gampola to Hatton, requiring the use of alternative communication protocols such as LoRa. The web interface should gracefully accommodate latency or data delays, and administrators should be notified if data stops updating. For booking, the frontend relies on internet

availability from user devices, which can vary depending on whether passengers are at home, on the go, or at a station.

The booking platform should work with all major browsers and mobile operating systems (Chrome, Firefox, Safari, Android, and iOS). The system's frontend must be responsive (mobile-first design) and load quickly even on low bandwidth connections. QR code production and scanning should function on a variety of screen sizes and camera types, with high contrast for easy detection.

Real-time data updates (such as GPS to Firebase or QR scan validation) must be asynchronous and fault-tolerant. adjustments to the admin panel and passenger interfaces (for example, route adjustments) must be reflected within a few seconds. If synchronization fails (for example, due to a Firebase outage), essential components such as gates and boarding should have manual override options.

- Social Context

  The social context includes human behavior, interaction patterns, public expectations, and culture, all of which influence how the technology is adopted and used. The system serves a large public audience, which includes non-technical users, children, and the elderly. Users may be unfamiliar with QR code-based boarding, therefore the interface should offer visual guidance, tooltips, and error management. During rush hours, users may disregard instructions, therefore feedback (such as gate lights or confirmation tones) is critical.

  Railway stations are frequently noisy, busy, and fast-paced, influencing how people engage with technology. To avoid boarding delays, scanning a QR code should take less than 2 seconds and include clear visual/audio confirmation. If a scanner fails, travelers should be able to contact station workers right away for manual verification.

  Admins and station staff must be trained to efficiently use the admin panel, execute manual overrides, and assist passengers who are having trouble scanning or booking. The system must also provide clear user documentation or an onboard help screen for kiosks.

- Environment Context

  This consists of both natural and artificial environmental conditions that affect device performance and user comfort. Rain, humidity, and dust are prevalent in many parts of Sri Lanka. Sensors and servo motors must be closed in weatherproof enclosures made of corrosion-resistant materials and engineered to endure temperature differences between stations such as Badulla (cooler) and Colombo (warmer).

  QR scanners should work in daylight, low light, and artificial lighting, particularly on early morning or evening trains. Display screens must be reflective and bright so that passengers can see ETA information even in direct sunshine. Audible feedback (beeps, chimes) should be loud enough to be heard but not disruptive. Visual cues (LEDs, screen prompts) must augment sound to ensure accessibility for the hearing impaired.

4. Technologies

   The TrainSync System's technical stack combines several hardware and software components to ensure the system's security, real-time functionality, scalability, and reliability. These technologies enable interaction between users, administrators, and physical devices while also providing data synchronization, validation, and reliable communication across the platform.

   - Frontend Technologies

     The frontend is built with HTML, CSS, and JavaScript to provide a clean, responsive, and mobile-friendly interface for both passengers and administrators. It enables users to book tickets, provide feedback, monitor train status, and track train locations on a real-time map. All displays are created with accessibility in mind and suited for smartphones and desktop computers. The use of current UI libraries increases uniformity and usability.

   - Backend Technologies

     The backend is built on the Flask framework (Python), which handles all server-side logic such as routing, QR code generation, email sending, data validation, and booking processing. Flask was chosen because it is simple, modular, and compatible with other Python libraries. It interfaces with hardware (ESP32) via APIs and returns dynamic

data to the frontend. The backend also provides admin authentication and secure RESTful API endpoints.

- Realtime tracking and Mapping

  The tracking system uses NEO-6M GPS modules on the train to determine latitude and longitude. These coordinates are relayed to Firebase via GSM via a 4G connection. The Google Maps API is incorporated into the website, allowing you to see the train's real-time location and dynamically calculate the ETA. This provides passengers with a convenient tool to track their train's progress and make timely boarding selections.

- QR Code and Email Handling

  After a successful booking, the backend generates a unique QR code using the Python qrcode package. This QR code is integrated in a PDF ticket using the ReportLab library and emailed to the traveler via Flask-Mail and smtplib. The QR code is then scanned at the train entrance and verified using code. This method provides frictionless, quick, and verifiable boarding via digital authentication.

- IoT and Embedded Technologies

  The ESP32 microcontroller serves as the primary control unit for all embedded operations, such as communicating with sensors, scanning QR codes, and activating servo motors. It is designed with the Arduino framework and linked to peripherals such an ultrasonic sensor (for train detection), an ESP 32 CAM, and servo motors (for gate/door control). The ESP32 also handles the logic that determines when a train is approaching or has stopped using sensor feedback or GPS data.

- Sensor and Automation Components

  Ultrasonic sensors detect the movement of trains on platforms or at railway crossings. If the sensor detects an object continually for 5 seconds, the ESP32 recognizes it as a genuine train detection and activates the relevant servo motors. Servo motors are used to open and close station gates, railway barriers, and train doors during boarding. These automated components increase safety while reducing the need for manual intervention by station personnel.

- Communication Technologies

  The 4G connection dongle transmits GPS coordinates to Firebase over mobile networks (Dialog). This provides access even when Wi-Fi is absent, as is frequently the case in moving trains. The ESP32 and the Flask server (for QR verification) communicate RESTfully over HTTP and JSON. This hybrid communication approach ensures asynchronous, dependable, and cost-effective data transfer between hardware and software.

- Security and Authenticity

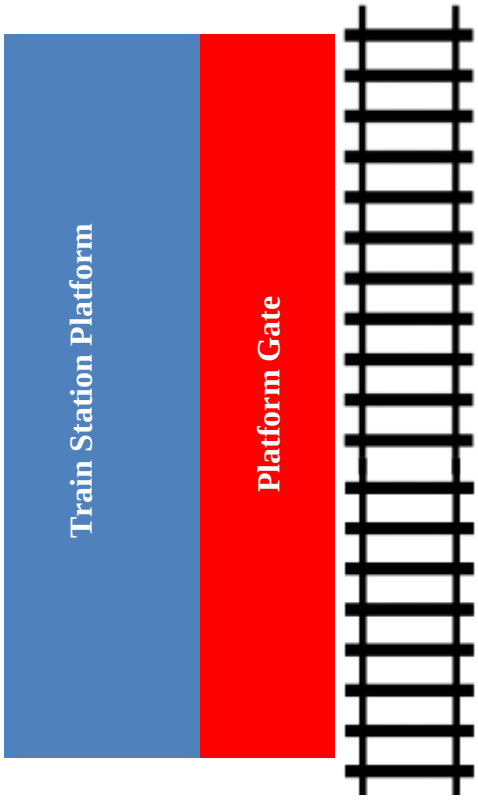
  The system provides security through various layers. Admin login credentials are safely kept in environment variables (.env files), with authentication handled on the backend via Flask routes. QR codes are unique and time-sensitive, and they are linked to specific bookings to avoid duplication or misuse. HTTPS is recommended for deployment since it encrypts API traffic between client devices, ESP32 modules, and the server.

3.2 Interfaces (software/hardware) of the system

1. Hardware Interface

- Platform Gate System

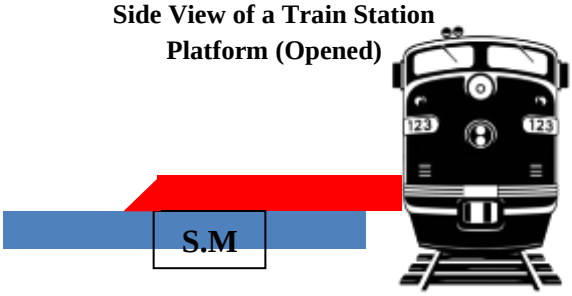
Plan View of a Train Station Platform



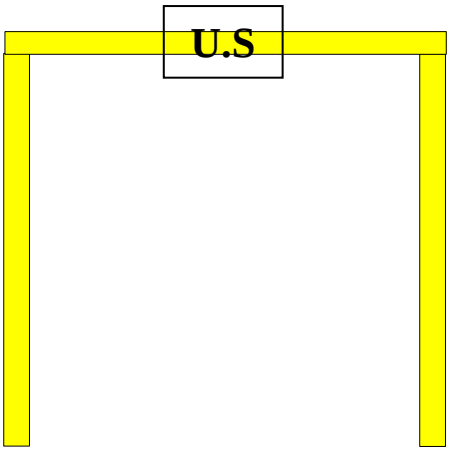
Side View of a Train Station Platform (Closed)

Platform Gate is closed (Train is Approaching)

Side View of a Train Station Platform (Opened)



Platform Gate is Open (Train is stopped in the station)

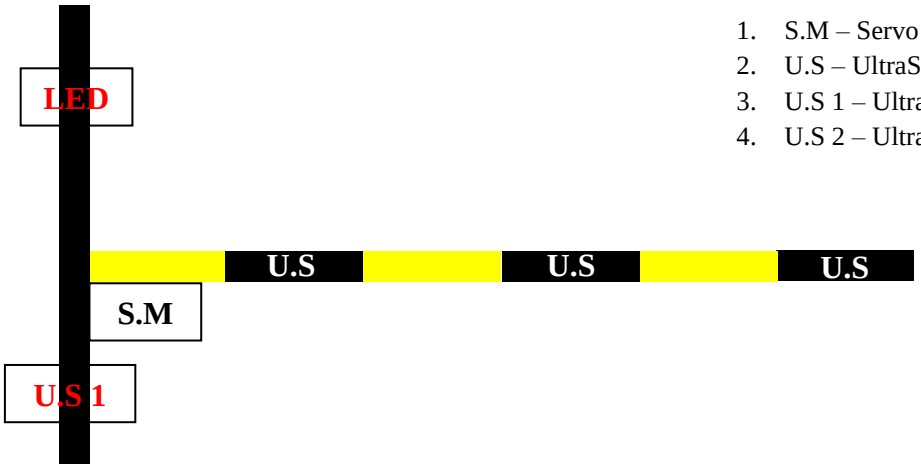


Placement of the UltraSonic sensor to detect the train

1. S.M – Servo Motor
2. U.S – UltraSonic Sensor

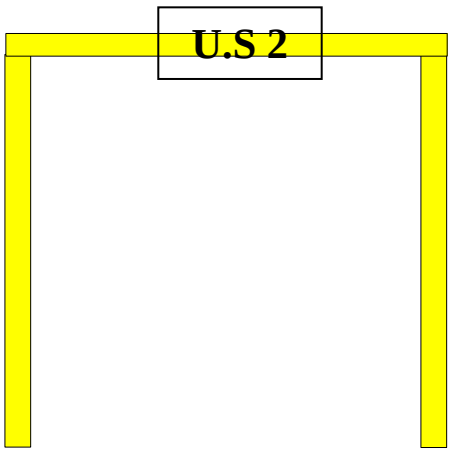
In here Side view of the train station platform the servo motors are placed in the bottom of the moving part of the platform gate so it will push the moving part upwards so the gates will be closed make avoidance of accidents during boarding. For make this servo motor works there is ultrasonic sensor situated far away from the train station that will detect trains.

- Railway Platform Gate System



1. S.M – Servo Motor
2. U.S – UltraSonic Sensor
3. U.S 1 – Ultrasonic Sensor for Under gate detection
4. U.S 2 – Ultrasonic Sensor for Train Detection

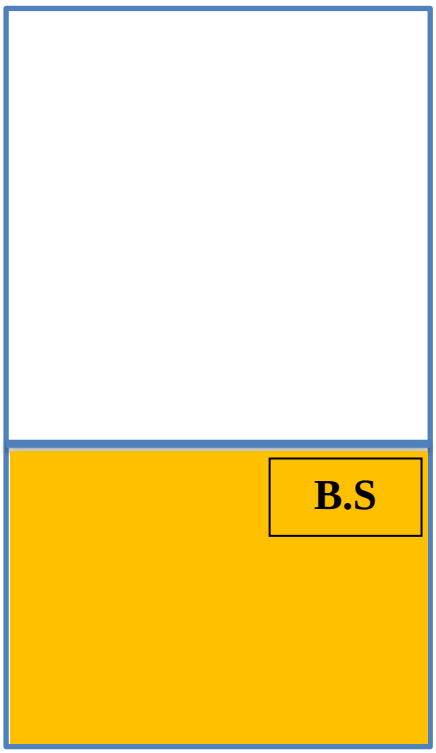
Front View of a Railway Cross Gate



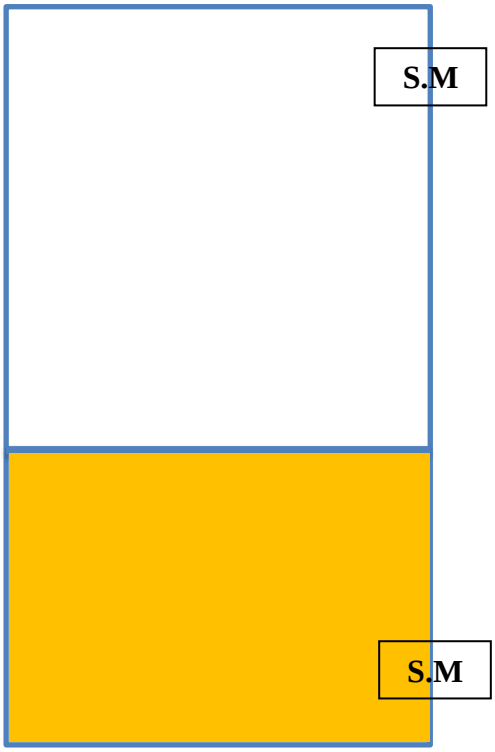
Placement of the Ultrasonic sensor to detect the train

In here the Ultrasonic sensors in the gate itself will detect if any vehicle is stuck in between the railway cross road gates and also U.S 1 will also detect if a vehicle is under the gates. When one of these ultrasonic sensors triggered the gates will not close servo motors will not work even though the train is detected by U.S 2. If U.S and U.S 1 sensors did not detect anything servo motors will work and close the railway gates automatically.

- QR Based Boarding System



Front View of the Boarding Door



Back View of the Boarding Door

1. S.M – Servo Motor
2. B.S – QR Scanning Module

In here the B.S contains ESP 32 CAM that scans QR codes when the QR code is verified the servo motor will start work and it will open the door smoothly.

2. Software Interface

Figure 14 UI Design of the Home Page

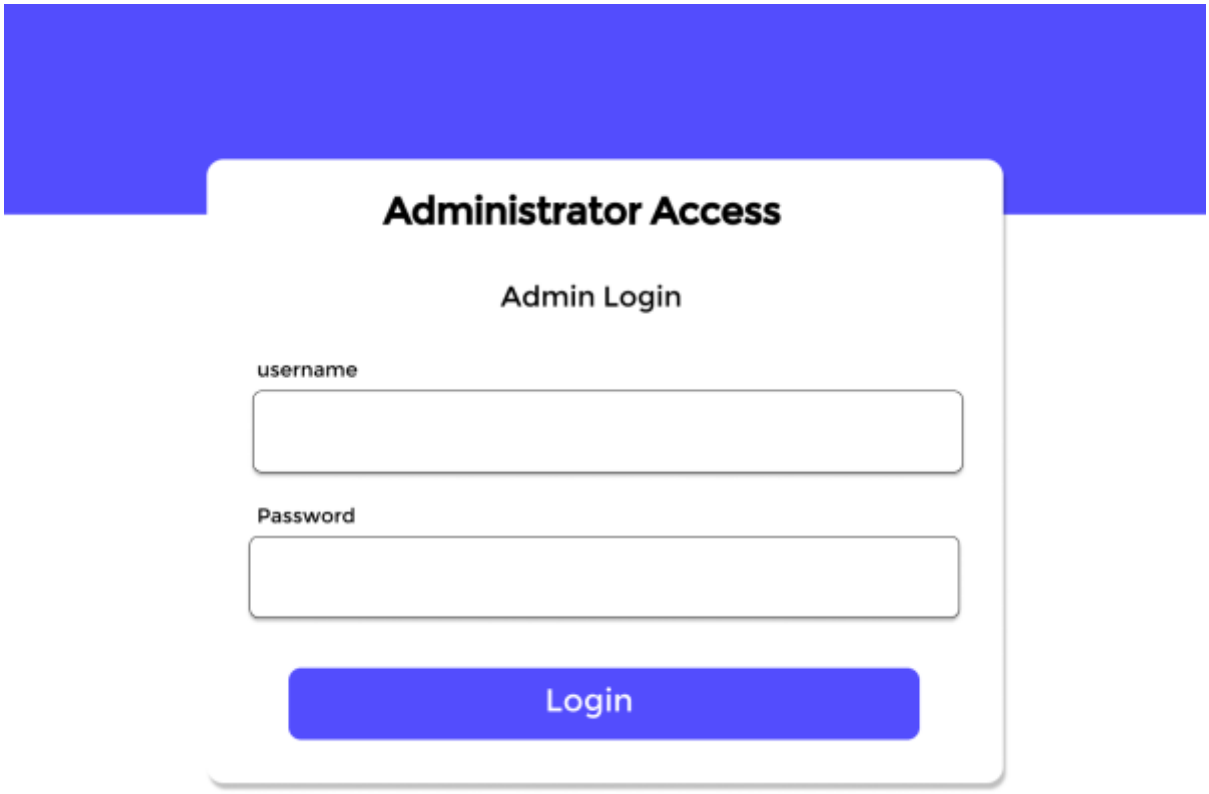


Figure 15 UI Design of the Admin Login page

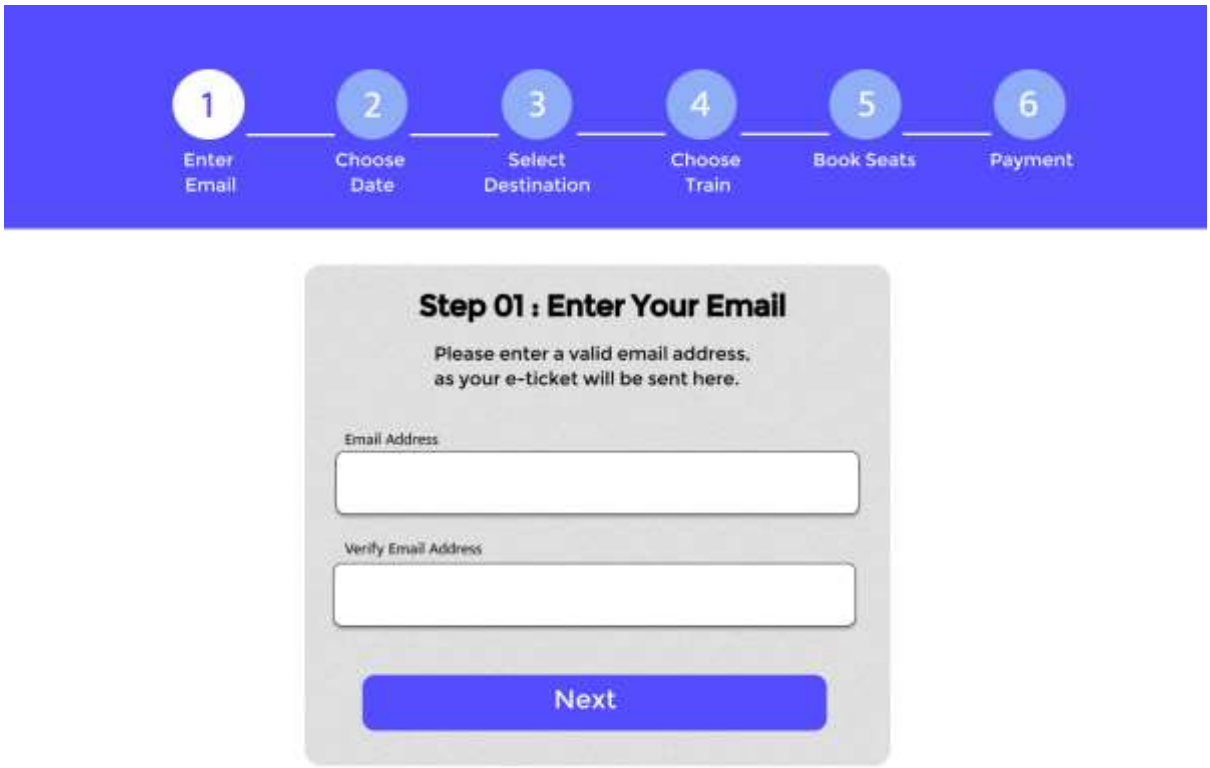


Figure 16 UI Design of the Booking Email Entering Page

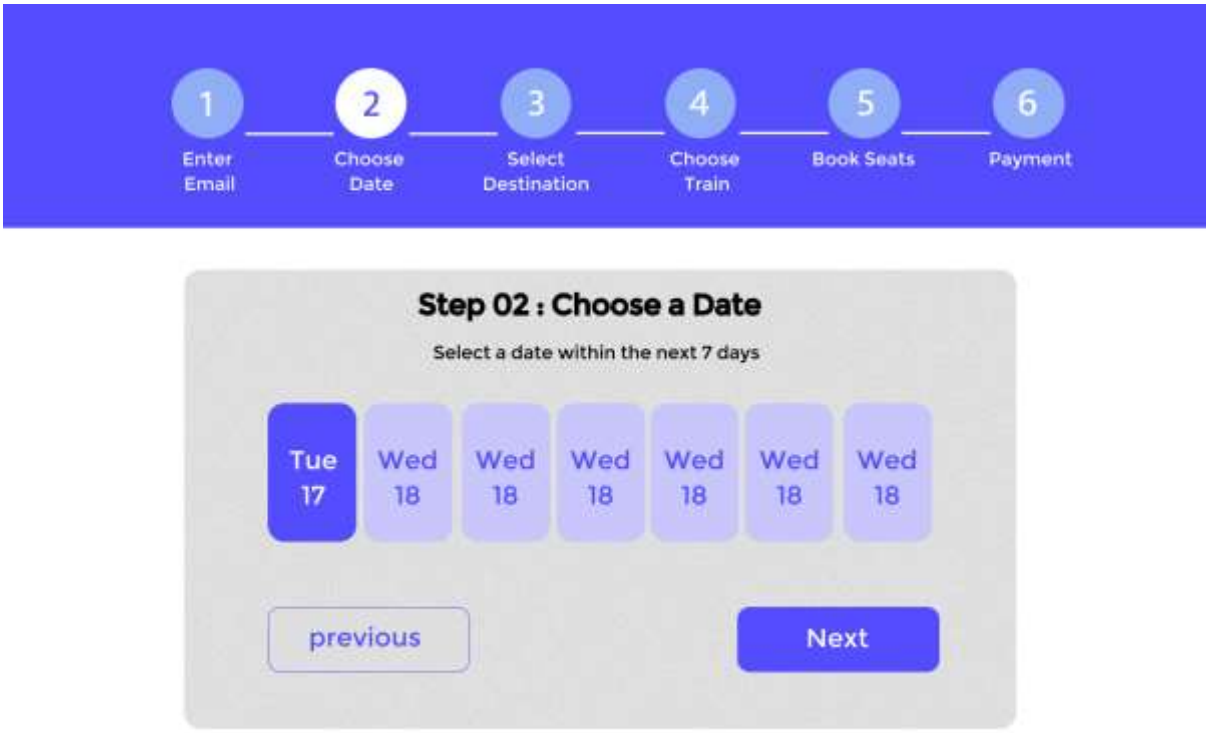


Figure 18 UI Design of the Date Choosing Page

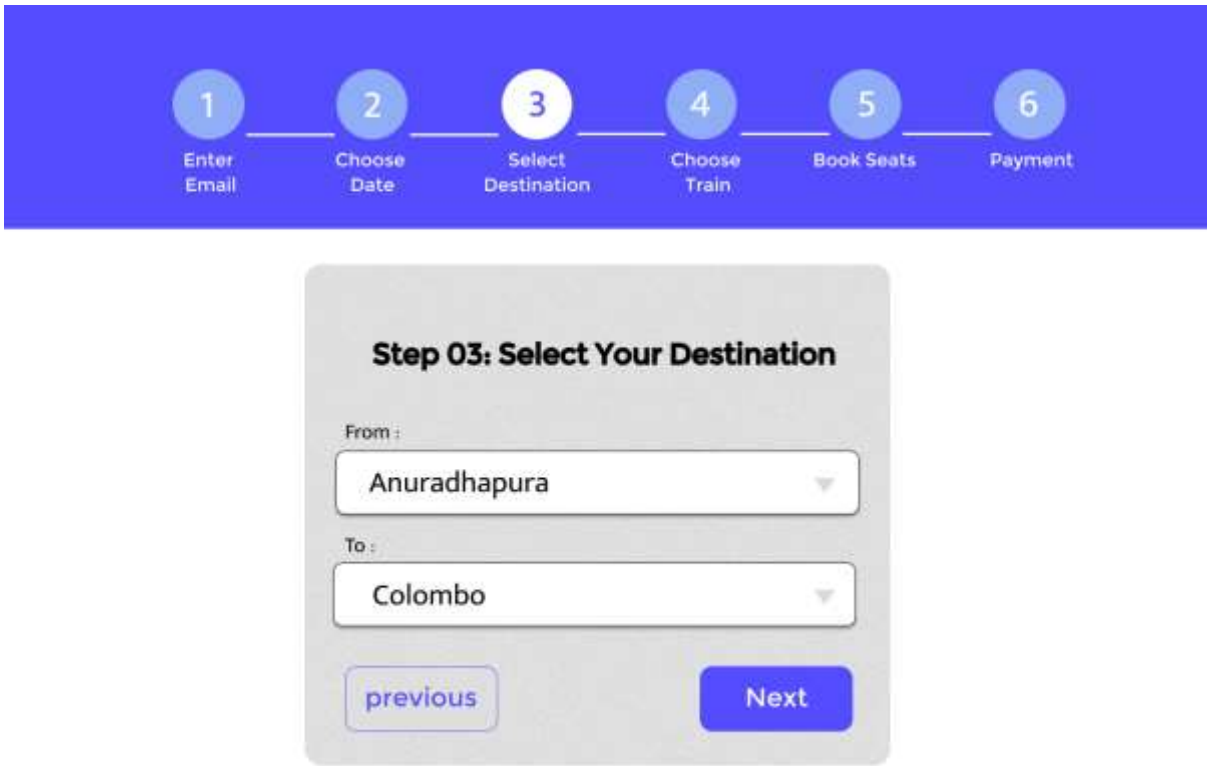


Figure 17 UI Design of the Destination Selection Page

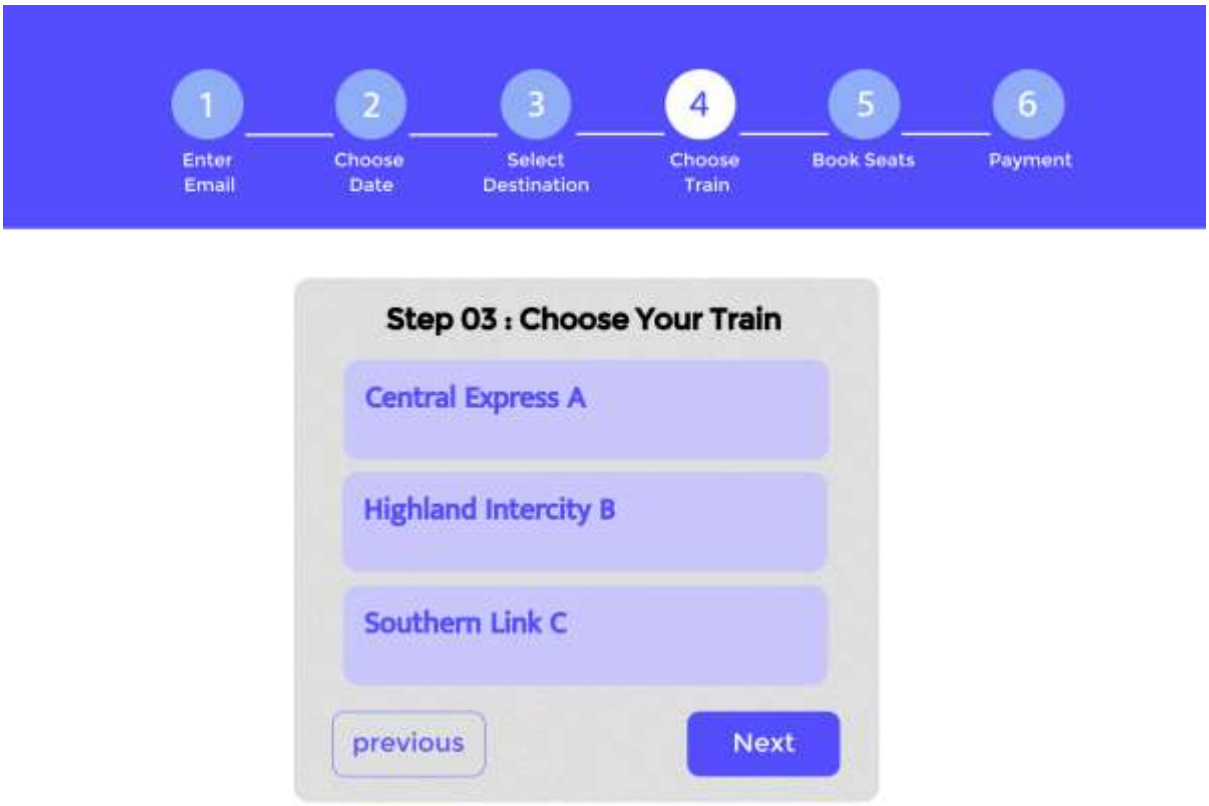


Figure 19 UI Design of the Train Selection Page

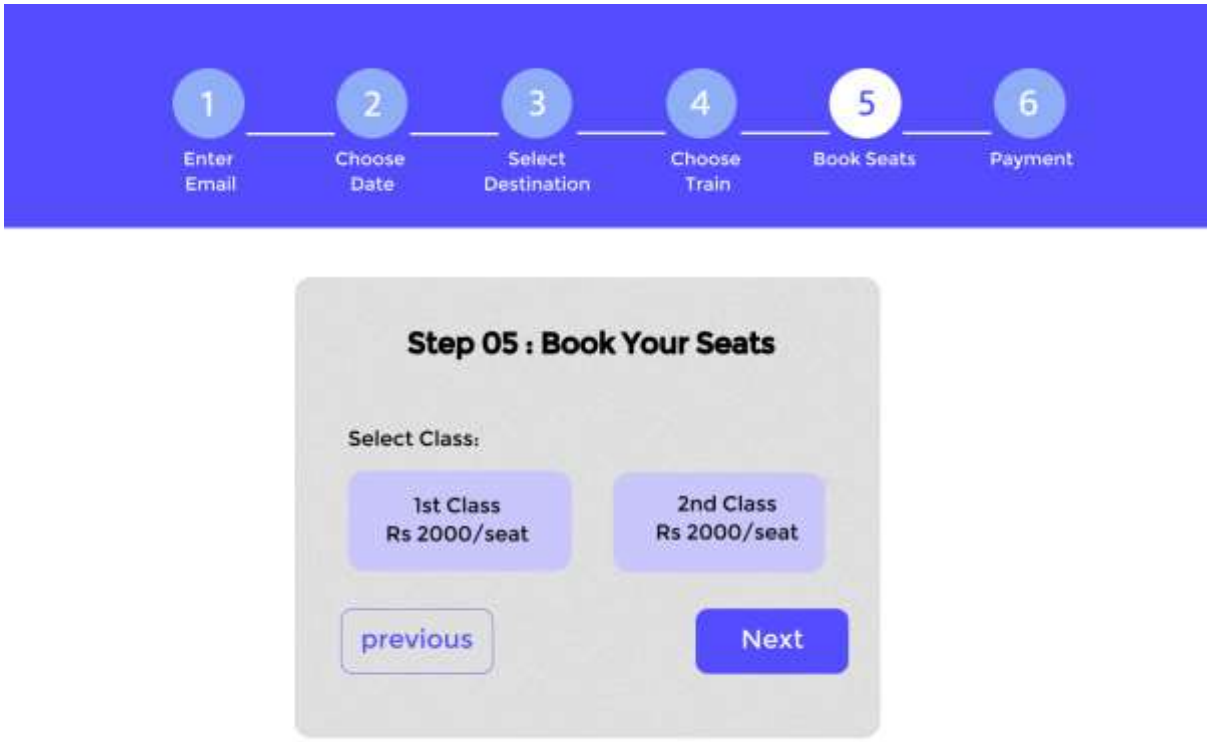


Figure 20 UI Design of the Seat Booking Page

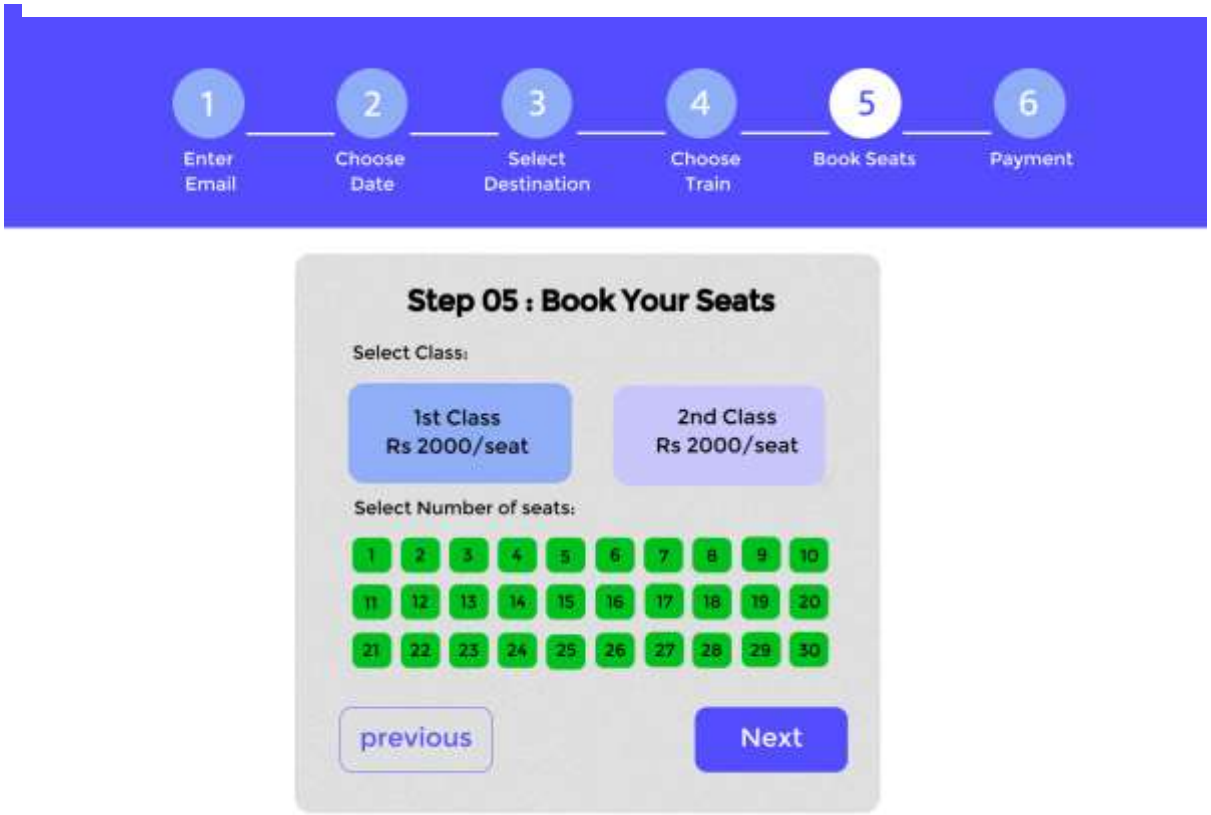


Figure 21 UI Design of the Class Selecting Page

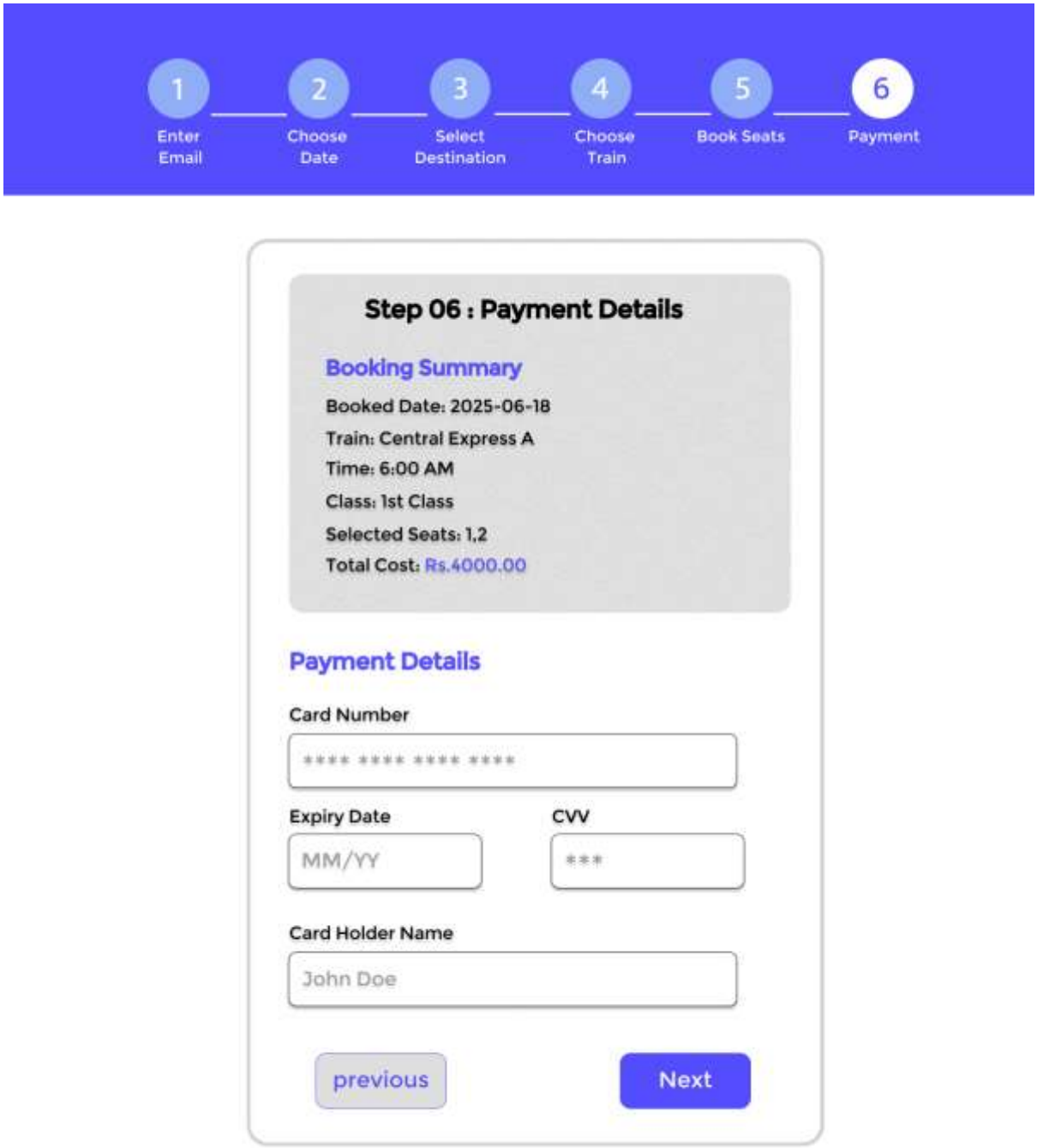


Figure 22 UI Design of the Payment Page

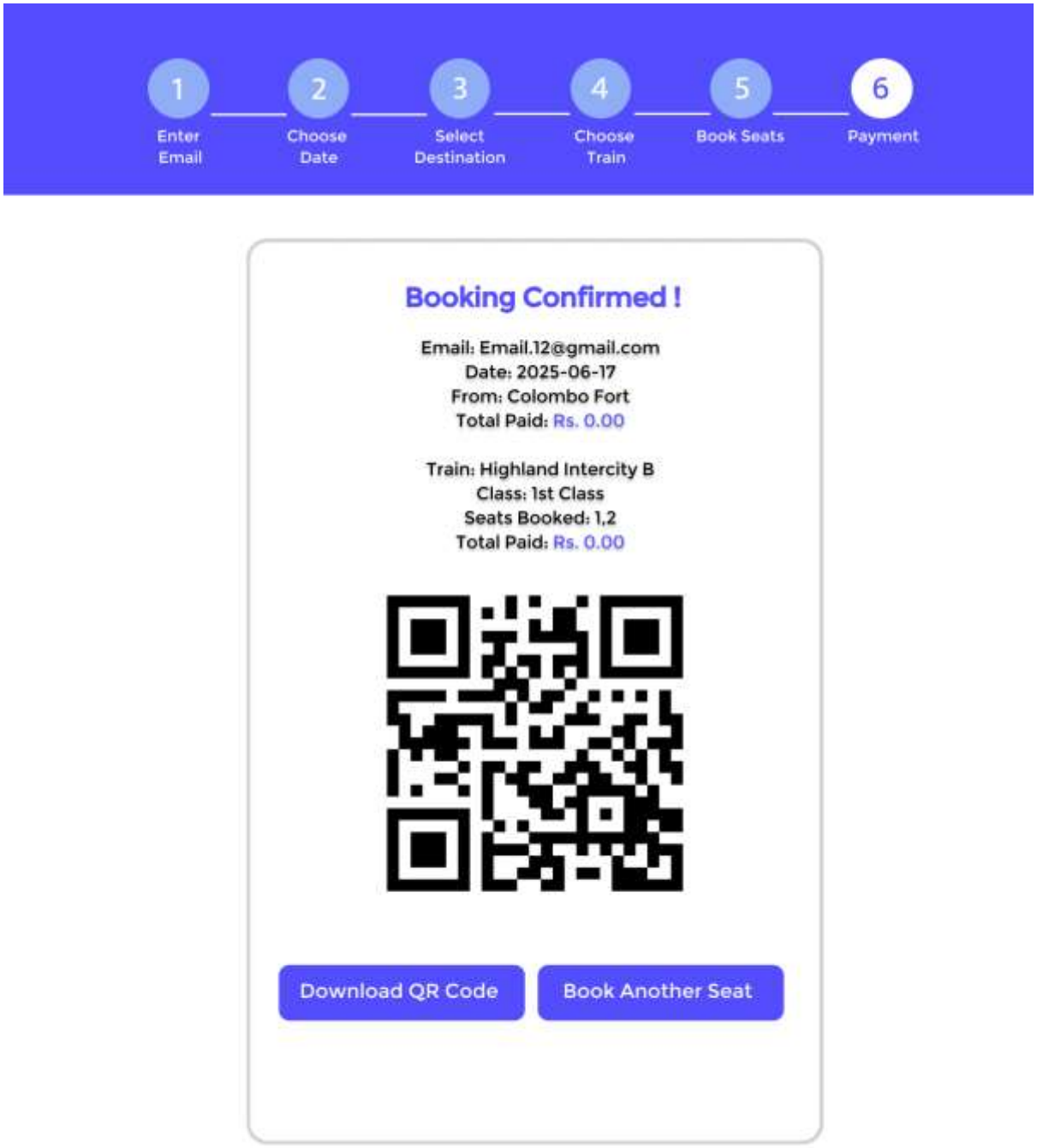


Figure 23 UI Design of the Booking Confirm Page

3.3 Design tools, techniques, templates

The design phase of this system focused on developing easy to use accessible, and responsive interfaces for both passengers and administrative users, as well as designing reliable interaction flows between software and hardware components. The tools and techniques chosen were based on their reliability, flexibility, open-source availability, and suitability for quick prototyping and production deployment. This section describes the design tools, UX/UI approaches, and templates that were used to guide the system's front-end and backend development.

1. UI/UX Design Tools and Techniques

   This step involves the use of tools such as paper sketching and Figma. Low-resolution drafts were first generated to outline the structure of web pages such as the booking page, train timetable viewer, admin panel, and QR scanning screen. Layouts targeted mobile-first design to enable usability on smartphones, tablets, and computers. Throughout the design phase, user personas like "daily commuter," "tourist," and "station admin" were taken into consideration. The UI was created with HTML5 and CSS3, assuring semantic structure and responsive style.

2. Dashboard and Data Display Design

   A customized admin dashboard was created with functionality for viewing current train timetables. Add or remove trains from routes. Validate QR codes manually (if necessary). The dashboard layout was straightforward and informative, with tabular views, buttons, and input forms stylized with minimal CSS. The Google Maps JavaScript API was integrated with the frontend interface to graphically display train whereabouts in real time.

3. QR and PDF Ticket Design

The reportlab library was used in Python to create styled PDF tickets. Ticket covers passenger and journey details, train name and time, QR code for boarding verification. Fonts, headings, and layout space were all changed to create a professional, readable style ideal for printing or mobile use. QR codes were created with the qrcode Python module. Each code encodes a unique string that corresponds to the booking ID. Codes were stored in PNG format and included in both the frontend for download and the PDF file distributed by email.

4 Data Management

4.1 Design tools, techniques

The Automated Train Platform Safety and Tracking System's data management design focuses on efficiently managing ticket booking information, train timetables, QR code generation, and email confirmations via a lightweight JSON-based file system and structured REST APIs. The system also works with Firebase for real-time GPS tracking, but all booking-related data is handled on the server side with Python and Flask.

1. Data Storage Mechanism

   The system uses a file-based data storage strategy, with all train data (IDs, names, schedules, and prices) saved in a JSON file named train_data.json. At program initialization, the load_train_data() function loads this file into memory, and save_train_data() saves it. This technique is ideal for small-scale systems since it eliminates the overhead of a standard database while also providing easy readability and flexibility. The loaded train data is saved in a global Python dictionary called train_data, which is changed in memory during execution. New entries (via /api/trains POST request) and updates (via PUT) are first added to this dictionary and then saved to the JSON file.

2. Data Manipulation and Processing Techniques

   All interactions with train schedule data are done using RESTful API endpoints written in Flask. For example, GET /api/trains will return full train data, POST /api/trains will add a new train to a specific route, and DELETE /api/trains// will delete a train from the list. These endpoints enable frontend clients, administrative panels, and other services to access and publish data in a consistent and structured format (JSON). The system uses explicit status codes (200 OK, 400 Bad Request, 409 Conflict, etc.) and JSON messages to indicate success or failure.

3. QR Data Handeling, Email Formatting, Admin Login and Data Security

Booking information (such as passenger email, selected date, train, class, and cost) is handled temporarily during QR code generation and email transmission via the /api/send-email route. This information is utilized to create a visual PDF ticket using the ReportLab library, a QR code with the qrcode Python tool, and an email with both the QR and journey information, which is then sent to the passenger. This data is not permanently stored in the backend; however, it can be upgraded to provide persistent storage in future versions using SQLite or PostgreSQL.

Booking data supplied to the /api/send-email endpoint is checked before processing to ensure that all needed fields (qr_data, recipient_email) are present. It also maps all trip information into a structured manner for PDF and email content. The technology secures SMTP credentials and prevents critical information from being hardcoded.

To secure basic credential information, admin authentication is performed using environment variables (ADMIN_USERNAME, ADMIN_PASSWORD). The /api/admin/login route checks these credentials before giving access to administrative operations. Passwords are saved in plain text in the.env file for simplicity, but in production, they should be hashed and salted using libraries such as bcrypt.

4.2 Conceptual database design

The conceptual database design defines the high-level structure of the system's data entities, including their attributes and relationships. It ensures data consistency, integrity, and logical grouping of information for various modules such as train ticketing, QR code verification, and email notifications.

Although the current system stores data in JSON files, this conceptual model prepares the system for a future move to a relational database system, such as SQLite or PostgreSQL, via an ORM such as SQLAlchemy in Flask.

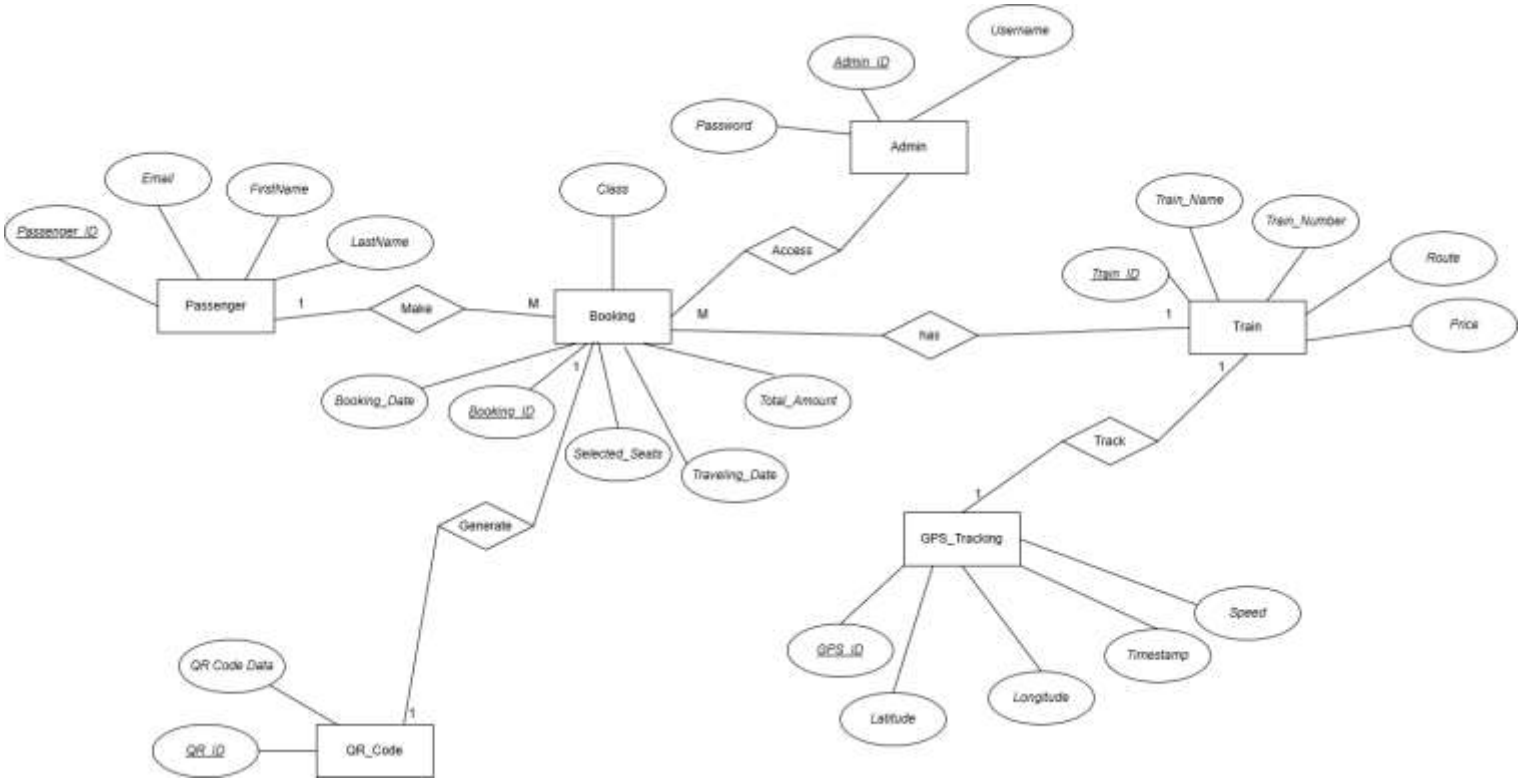


Figure 24 ER Diagram of the System

[Drive Link of the ER Diagram]

The following are the key entities identified in the system, along with their attributes.

1. Passenger

Table 1 Passenger Attribute Table

Attributes	Description
PassengerID	Primary Key
First Name	Passenger's first name.
Last Name	Passenger's last name.
Email	Email address.

2. Train

Table 2 Tain Attributes Table

Attributes	Description
TrainID	Primary Key
TrainName	Train's name.
TrainNumber	Train's number.
Route	Train's route from and to.
Time	Time of departure.
Price	Price of seats.

3. Booking

Table 3 Booking Attributes Table

Attributes	Description
BookingID	Primary Key
PassengerID	Foreign Key
TrainID	Foreign Key
BookingDate	Date of booking was done.
TravelDate	Date of travel.
Class	Booked class.
TotalAmount	Total Amount paid for seats.
SelectedSeats	Selected seats by the passenger.

4. QRCode

Table 4 QR Code Attributes Table

Attributes	Description
QRCodeID	Primary Key
BookingID	Foreign Key
QRCodeData	Encoded booking string.

5. Admin

Table 5 Admin Attributes Table

Attributes	Description
AdminID	Primary Key
Username	Username of the admin.
Password	Password to login.

6. GPS Tracking

Table 6 GPS Tracking Attributes Table

Attributes	Description
TrackingID	Primary Key
TrainID	Foreign Key
Latitude	Latitude code of the train location.
Longitude	Longitude code of the train location.
Speed	Speed of the train.
Timestamp	Timestamp of the train travel happening.

7. ETA Calculation

Table 7 ETA Calculation Attributes Table

Attributes	Description
ETAID	Primary Key
TrainID	Foreign Key
CalculatedETA	The time that calculated and showing.
CalculationTime	The time that always calculating.

Relationships are created as below,

- One Passenger – Many Bookings (One to Many Relationship) which means each passenger can book multiple tickets.
- One Train – Many Bookings (One to Many Relationship) which means a train can have many bookings.
- One Booking – One QRCode (One to One Relationship) which means each booking will generate one QR code.
- One Train – Many GPSTracking (One to Many Relationship) this is because the location must constantly update to show accurate location of the train.

- One Train – Many ETACalculation (One to Many Relationship) this is because like in the location tracking it should also continuously update.

Data constraints and rules are following,

- Primary Keys – Each entity has a unique identifier (PassengerID, TrainID, BookingID, etc.).

- Foreign Keys – Maintain referential integrity between Passenger, Train and Booking.
- Seats should be stored in a format that allows parsing (e.g., comma-separated or JSON list).
- Email addresses must follow proper format.
- Qr data must be unique per booking and the admin username and password were predefined and stored in the database and it will never change unless a security issue occurs.

4.3 Logical database design and schema refinement

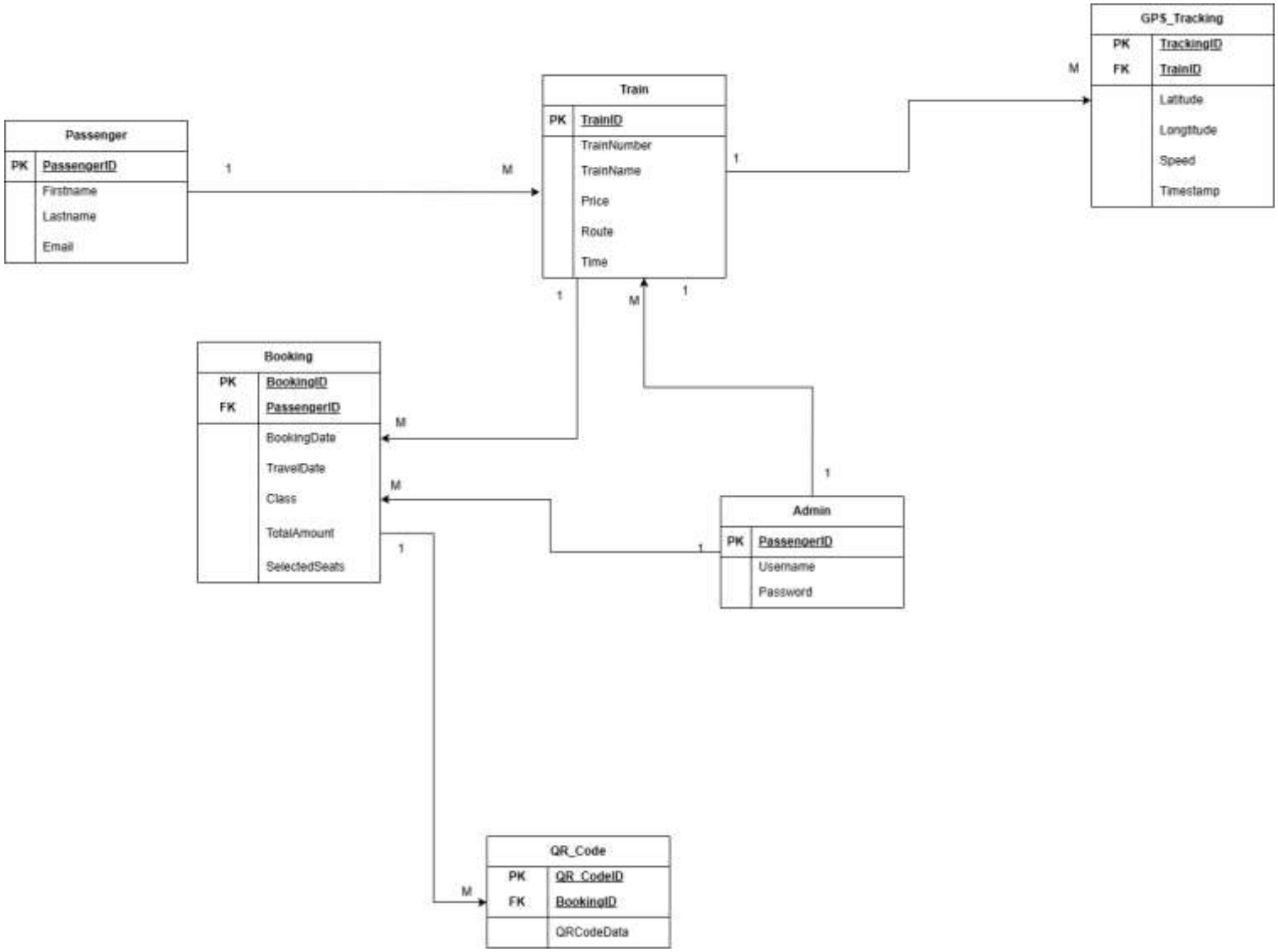
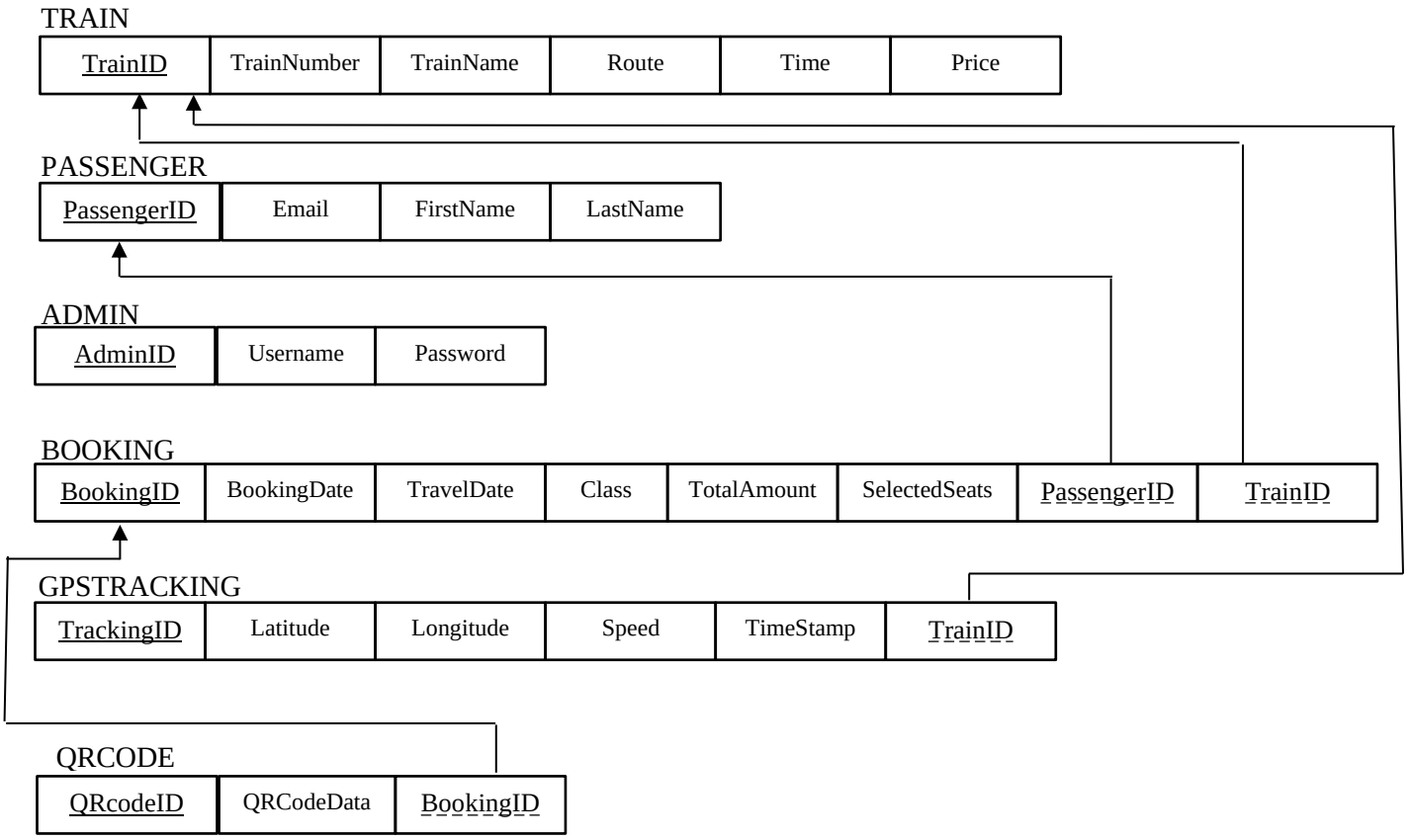


Figure 25 EER Diagram of the System


TRAIN
TrainID
TrainNumber
TrainName
Route
Time
Price
PASSENGER
PassengerID
Email
FirstName
LastName
ADMIN
AdminID
Username
Password
BOOKING
BookingID
BookingDate
TravelDate
Class
TotalAmount
SelectedSeats
PassengerID
TrainID
GPSTRACKING
TrackingID
Latitude
Longitude
Speed
TimeStamp
TrainID
QRCODE
QRcodeID
QRCodeData
BookingID

4.4 Physical database design

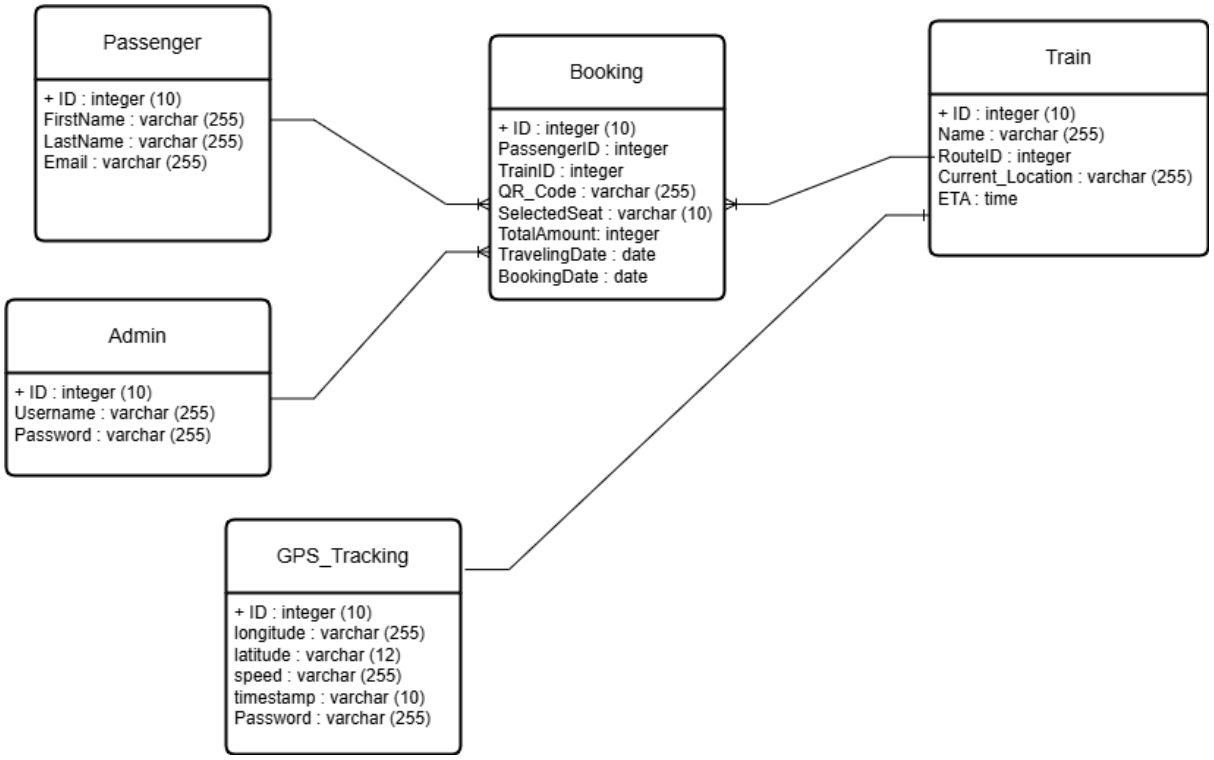


Figure 26 Physical Database Design of the System

5 Hardware Design (if available)

1. Railway Cross Road System

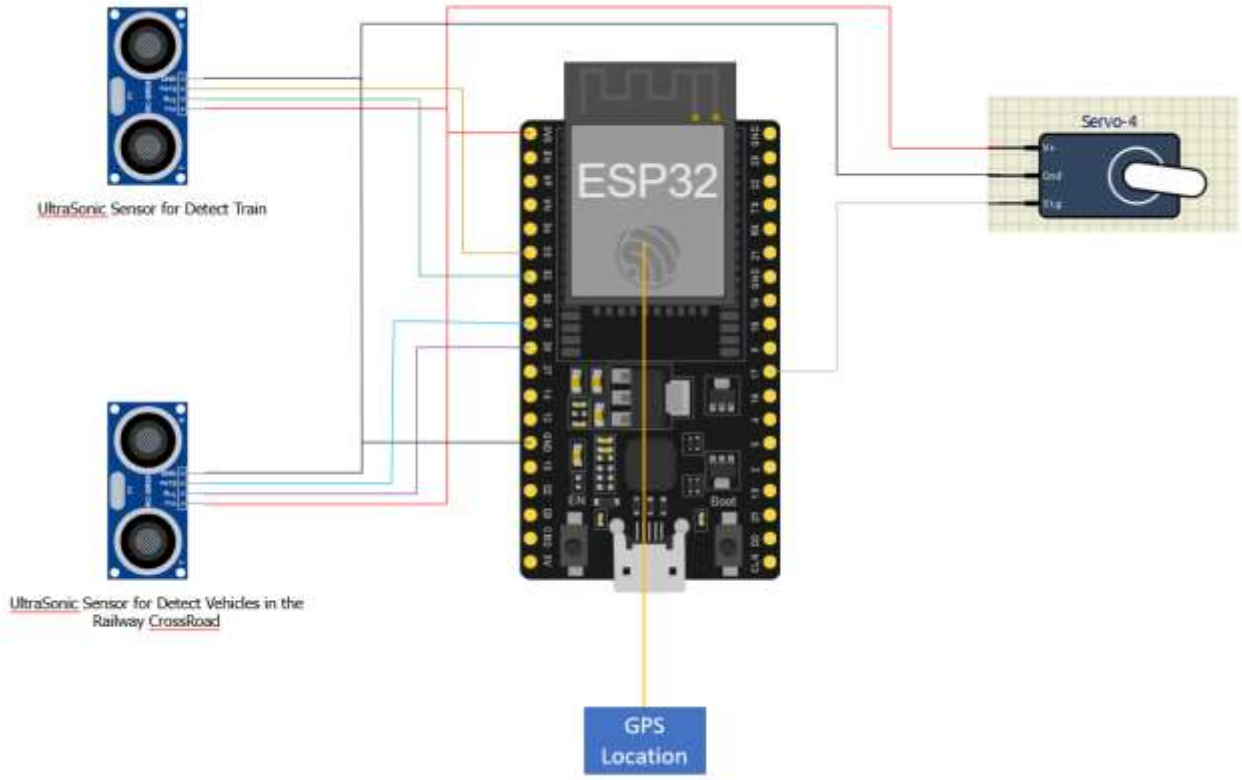


Figure 27 Hardware Design of the Railway Cross Road System

This figure shows the hardware implementation of the Railway Crossroad System, which uses an ESP32 microcontroller to automate the functioning of a railway crossing gate. One HC-SR04 ultrasonic sensors are linked to the ESP32: detect an incoming train, while the other detects vehicles on the railway crossing. These sensors always analyze their surroundings and transmit distance information to the ESP32. When the sensor detects a train approaching within a predefined range and no vehicle is detected on the tracks, the ESP32 activates a servo motor that serves as a crossing gate barrier. When the train has passed and the track is clear, the ESP32 releases the gate by rotating the servo back. This system keeps safety by preventing cars from being on the track when a train is nearby, fully automating crossing gate control with no user intervention, and can be expanded to interface with the main Train System via wireless modules for more intelligent coordination.

2. Platform Gate System

Figure 28 Hardware Design of the Platform Gate System

This figure illustrates the hardware setup for an Automated Platform Gate System, which uses an ESP32 microcontroller, an ultrasonic sensor, and a servo motor to control the opening and closing of gates at a train station platform. In this system, an HC-SR04 ultrasonic sensor is placed along the track to detect the presence of an incoming train. When the sensor detects a train within a specific distance, it transmits a signal to the ESP32. The ESP32 interprets this signal and engages the servo motor, which controls the physical gate mechanism opening or closing the platform gate dependent on train proximity. When the train departs and the sensor no longer detects its presence, the ESP32 instructs the servo to reset the gate to its original position. This basic yet effective solution automates passenger control at platforms, ensuring safety by only allowing access while no trains are in action. The technology can also be extended to interface with central train tracking systems, allowing for more synchronized and intelligent platform operations.

3. QR based Boarding System

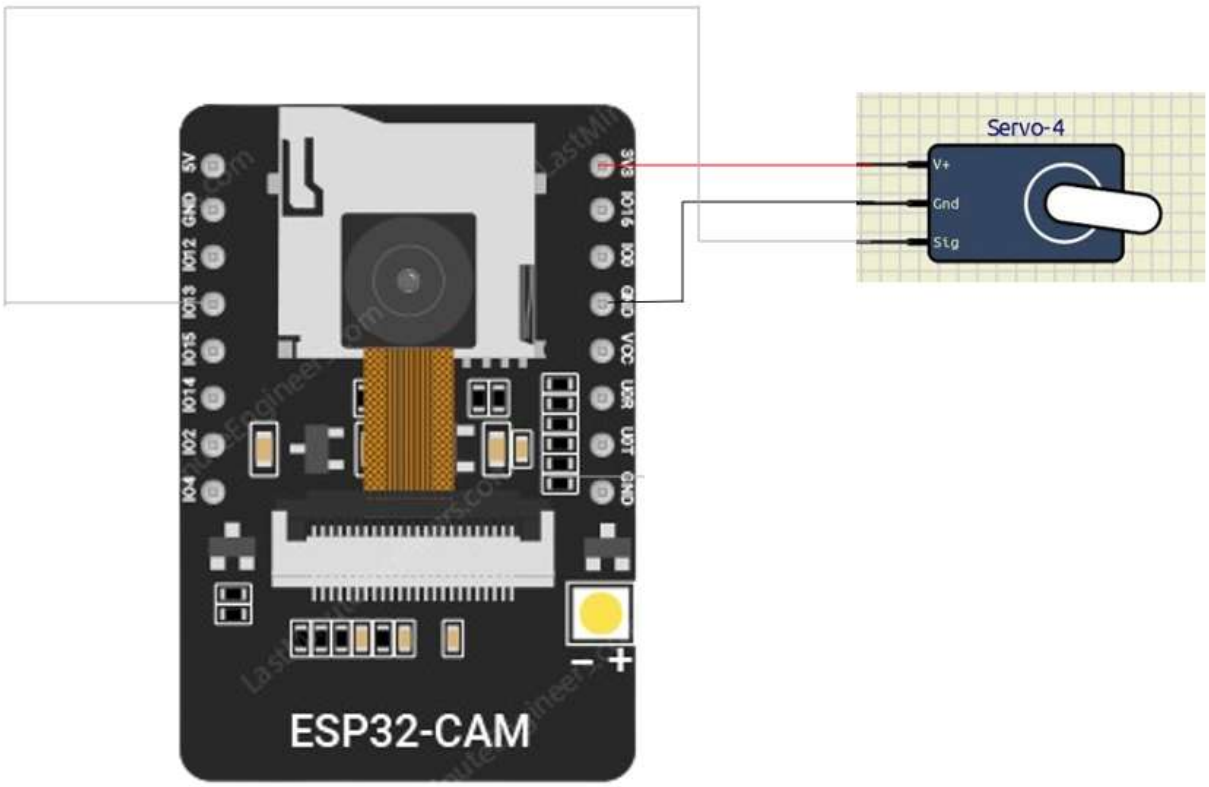


Figure 29 Hardware Design of the QR based Boarding System

This figure shows the hardware setup for a QR-Based Boarding System, which includes an ESP32-CAM module and a servo motor for secure, automated entrance control at train boarding gates. The ESP32-CAM includes an embedded camera that scans passenger QR codes at the gate. After scanning a QR code, the ESP32-CAM compares it to saved or backend booking data (as illustrated in the system architecture using Flask). If the QR code is correct, the ESP32-CAM sends a signal to the attached servo motor, which turns to unlock or open the boarding gate, allowing the passenger to enter. If the QR code is invalid or not recognized, the servo remains inactive, keeping the gate closed. This technology ensures that only authorized people can board, which improves security and streamlines admission without the need for manual verification. It is a critical component of the wider smart railway system, ensuring seamless interface with backend booking and validation services.

4. GPS and ETA System

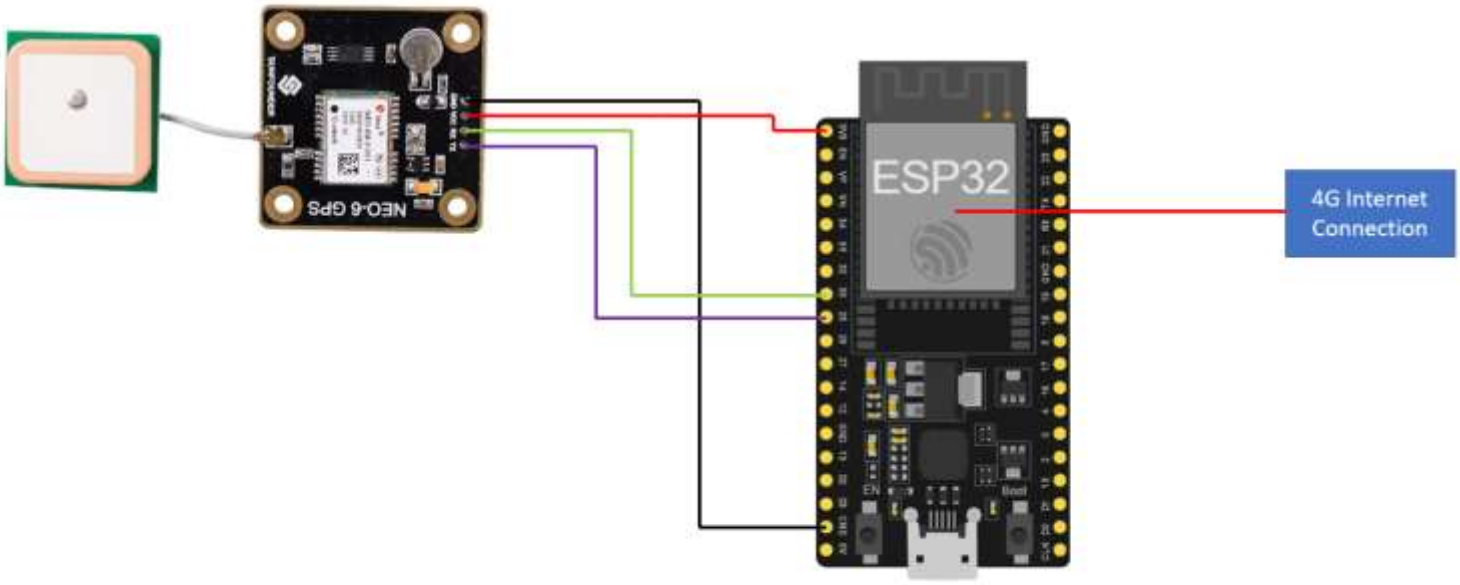


Figure 30 Hardware Design of the GPS and ETA System

This figure shows the hardware setup for a GPS Tracking System in the smart railway infrastructure, which commonly includes an ESP32 microcontroller and a NEO-6M GPS module. In this system, the NEO-6M GPS module connects to the ESP32, which continuously collects real-time GPS coordinates of a moving train. The ESP32 processes these coordinates (latitude and longitude), which can then be used to determine the train's estimated time of arrival (ETA) or track its whereabouts on a live map. The ESP32 may then send this information to a Firebase database, which maintains and updates train location information, allowing the web interface to provide real-time tracking for passengers and control systems.

6 Recommendation of supervisor(s) on the document

(This section should be filled by the supervisor(s)).

Comments (if any):

I/We certify that, the student engaged continuously with me in developing the proposal and, I am confident that they are adequately competent to defend this viva.

Signature(s) of Supervisor(s):

Date:

7 Viva presentation assessment team

(This section should be filled by the department)

Date of viva presentation: _____

Panel members	Name	Department / Institute
Chair		
Member		
Member		
Member		
Member		

8 Comments of the assessment team on viva presentation

(This should be filled by the chair of the assessment panel. In case of revision or fail, needed revision or reasons to fail the viva presentation should be mentioned here)

Result of the viva presentation	Excellent / Good / Pass with revisions / Fail
Score	
Signature of the panel chair	
Date	