

**System Design Specification  
of  
Enhancing Railway Safety using Computer Vision and Cloud Based  
Technologies**

**Tech Wizards**

<b>Name</b>	<b>Registration Number</b>	<b>Index Number</b>
M.K.M Halis	ASP/19/20/022	4856
G.E. Vinonsan	ASP/19/20/131	4793
M.M Mohamed	ASP/19/20/042	4858
Mathurika. J	ASP/19/20/097	4838
M.S.F Sumaiya	ICT/19/20/112	5048

Supervised by: N.M.A.P.B. Nilwakke

**Faculty of Applied Sciences  
Rajarata University of Sri Lanka  
2019/2022**

# Table of Contents

<b>1. Introduction</b>	<b>5</b>
1.1. Problems To Be Addressed	5
1.2. Objectives of the Project	5
1.3. Project Deliverables	5
1.3.1 Web Application	5
1.3.2 Object Detection Model exe file	5
1.3.3 Thesis	5
1.3.4 Research Paper	5
1.4. System Design Approach	6
1.5. Standards to be Followed	8
1.6. Organization of the SDS	8
<b>2. Architectural Design</b>	<b>10</b>
2.1. System Architecture	10
2.2. Objects and Communication	11
2.2.1. Dynamic Behaviors	13
2.4. State Machines	23
2.5. Tools, techniques, libraries, and Implementation Environment	31
<b>3. UI Design</b>	<b>31</b>
3.1. Overview of UI Design	31
3.2.1 People	32
3.2.2 Activity	33
3.2.2.1 Temporal Aspects	33
3.2.2.2 Cooperation & Complexity	33
3.2.2.3 Nature of the Content	33
3.2.2.4 Security critical	33
3.2.3 Contexts	34
3.5. Design Aspects	50
3.5.1. Input Design Aspects	50
3.5.2. Output Design Aspects	51
3.5.3. Input and Output Design Aspects	51
3.5.4. Dialogue Design Aspects	52
3.6. Hosting environment	52
<b>4. Data management</b>	<b>52</b>
4.1. Data Requirement	53
4.2. Design Tools and Techniques	53
4.4. Conceptual Database Design	54
4.5. Logical Database Design	54
4.5.1 Map Regular Events	55
4.5.2. Map Binary Relationship	57
4.6. Schema refinement	60

4.7. Physical Database Design .....	61
4.8. Security Design.....	61
5. Research Design .....	62
5.1. Objective Based Literature Review .....	62
5.1.1 To Develop a Model for Object Detection .....	62
5.1.2 Development of a Model for Real-Time Location and Weather Prediction.....	63
5.1.3. Developing of web application .....	63
5.2. Formalizing high-level implementation components .....	65
5.3. Data extractions, sample design, test data sets, training data sets.....	66
5.4. Non-functional aspects.....	67
5.4.1. Product requirement.....	67
5.4.2. Organizational Requirement.....	67
5.4.3. External Requirements .....	67
6.References .....	67
7. Approval.....	69

## Table of Figures

Figure 1 - Sprint Diagram .....	7
Figure 2- Architecture design.....	10
Figure 3 - Registration .....	13
Figure 4 - Login .....	14
Figure 5 – Reset Password .....	15
Figure 6 – Object Detection .....	16
Figure 7 - weather forecasting .....	17
Figure 8 - Weather Forecasting .....	18
Figure 9 – Update Hazard .....	19
Figure 10 – Approve Hazard.....	20
Figure 11 - Component Diagram .....	21
Figure 12- State machine 1 .....	23
Figure 13 - State machine 2 .....	24
Figure 14 - State machine 3 .....	25
Figure 15 - State machine 4 .....	26
Figure 16 - State machine 5 .....	27
Figure 17 - State machine 6 .....	28
Figure 18 - State machine 7 .....	29
Figure 19 - State machine 8 .....	30
Figure 20 - PACT Table .....	32
Figure 21 - Register Page Layout .....	36
Figure 22 - Login Page Layout .....	37
Figure 23 - Reset Password Layout .....	37
Figure 24 - Update Hazard Layout .....	38
Figure 25 - Admin Home Page Layout .....	38
Figure 26 - Home Page Layout.....	39
Figure 27 - Select Route Layout .....	39
Figure 28 - Hazard Location Layout.....	40
Figure 29 - Admin Approve Hazard Layout 1 .....	40
Figure 30 - Admin Hazard Location Update Layout .....	41
Figure 31 - Admin Approve Hazard Layout 2 .....	41
Figure 32 - Admin locomotive pilot Details Layout .....	42
Figure 33 - Register Page.....	42

Figure 34 - Login Page.....	43
Figure 35 - Reset Password.....	43
Figure 36 - Select Route.....	44
Figure 37 - Home Page 1 .....	44
Figure 38 - Home page 2.....	45
Figure 39 - Home page 3.....	46
Figure 40 - Home page 4.....	46
Figure 41 - Update Hazard.....	47
Figure 42 - Hazard Location .....	47
Figure 43 - Admin Home Page .....	47
Figure 44 - Admin Approve Hazard.....	48
Figure 45 - Update Hazard Location.....	49
Figure 46 - Admin Hazard Location Update.....	49
Figure 47 - Update Pilot Details .....	50
Figure 48 - EER Diagram .....	54
Figure 49 - Schema refinement .....	60
Figure 501- MVC pattern.....	64
Figure 51 - High-level implementation component.....	65
Figure 52- Performance analyzer neural network.....	67

# 1. Introduction

## 1.1. Problems To Be Addressed

The railway industry faces numerous challenges related to safety, operational efficiency, and hazard mitigation. One of the critical issues is the timely detection of obstacles, adverse weather conditions, and potential hazards along railway tracks. Traditional methods of manual surveillance and reactive response mechanisms often fall short in addressing these challenges effectively. The lack of automated systems for object detection, real-time weather updates, predictive forecasting, and hazard identification contributes to increased risks, operational disruptions, and safety concerns within railway operations.

## 1.2. Objectives of the Project

- Develop precise algorithms for identifying objects on and near the railway tracks, ensuring accuracy in diverse weather conditions.
- Integrate cloud technology for real time updates on train locations and weather information.
- Integrate cloud-based services to provide real-time updates on the geographic coordinates of bull and elephant presence. Implement rainfall level measurements and trigger alerts for landslide detection.
- Develop a system for quick processing of live data streams, minimizing delays for immediate feedback to drivers.
- Design an intuitive interface displaying real-time information clearly, with voice interaction for seamless use by locomotive pilots.

## 1.3. Project Deliverables

Railway safety is a critical concern that demands innovative solutions. Our proposed system is a web application specifically designed to address these safety challenges. By leveraging advanced object detection techniques and hazard identification algorithms, our system integrates an alerting system and an intuitive user interface. This integration creates a real-time responsive system that significantly enhances railway safety.

### 1.3.1 Web Application

A web application that is dedicated to railway safety enhancements, utilizing object detection techniques and hazard identification algorithms. It provides real-time monitoring, detecting obstacles like animals, vehicles, and humans, and identifying potential hazards such as landslide conditions and adverse weather conditions. With an intuitive interface, it enables locomotive pilots to access crucial safety information and alerts promptly, ensuring swift action to mitigate risks and improve safety protocols.

### 1.3.2 Object Detection Model exe file

Deliver an advanced object detection algorithm for railway tracks that accurately identify stationary and moving objects, including wild animals like bulls and elephants in the path of the train. Additionally, these algorithms will detect humans and large vehicles within the railway track, alerting pilots to potential obstacles.

### 1.3.3 Thesis

This thesis explores the development and implementation of a web application aimed at improving railway safety. It investigates advanced object detection techniques and hazard identification algorithms to create a real-time responsive system. The thesis discusses the integration of an alerting system and a user-friendly interface, highlighting the system's potential to significantly enhance railway safety.

### 1.3.4 Research Paper

This research paper presents a novel approach to addressing railway safety concerns through the development of a web application. It delves into the utilization of advanced object detection techniques and

hazard identification algorithms. The paper discusses the integration of an alerting system and an intuitive user interface, providing insights into how this system can enhance railway safety in real-time.

#### 1.4. System Design Approach

The selection of Scrum as the system design approach for our proposed application is driven by the objective of engaging and motivating users to actively participate in every process of our proposed web application. This choice is supported by several technical factors that contribute to the development of the application. The reasons why we chose Scrum:

- i. The agile philosophy of Scrum promotes collaboration, iterative development, and delivering value at each project stage, aligning with the railway safety system web application's needs.
- ii. Scrum provides transparent methods like product backlogs, sprint backlogs, and burn-down charts to track progress, monitor project status, and gain valuable insights for effective development.
- iii. Scrum's focus on continuous improvement supports the long-term success of the web application.
- iv. Scrum's focus on continuous improvement supports the long-term success of the web application.

Daily stand-up meetings in Scrum facilitate coordination among team members.

So, by adopting Scrum, we can enhance the development process of our proposed application.




The below figure 1 illustrates the breakdown of our project into three sprints, each outlining the specific activities to be implemented within each sprint. By breaking the project into these sprints, we ensure a systematic and incremental development approach. This allows for efficient collaboration among team members, early identification and resolution of issues, and the timely delivery of our machine learning model web application.

Let's elaborate on the technical details of each sprint:

**Sprint 1:** This sprint concentrates on machine learning model development, client-side web development, and database management. Within this sprint, the team will engage in activities such as researching and building machine learning models, creating user interfaces for the client side of the web application, and setting up and managing the database for storing and retrieving data. This includes designing and implementing the user interface (UI) components, creating interactive features, and ensuring a seamless user experience. Additionally, database management tasks will be undertaken, such as designing the database schema, establishing database connectivity, and implementing data storage and retrieval mechanisms.

**Sprint 2:** In the second sprint, the focus shifts to server-side web development. The team will develop the server-side logic and functionalities required to handle client requests, process data, and interact with the database. This includes implementing APIs, server routes, and business logic. Furthermore, unit testing will be conducted to verify the correctness of the server-side code and to identify and fix any defects or errors.

**Sprint 3:** The third sprint is dedicated to integration testing, where different components and modules developed in the previous sprints are combined and tested as a whole system. Following successful integration, acceptance testing will be performed to validate that the system meets the specified requirements and satisfies the needs of stakeholders. This involves evaluating the system's functionality, usability, performance, and overall quality. Additionally, the sprint includes evaluation activities to assess the effectiveness and efficiency of the implemented machine learning model and its integration into the web application.

Implementation 01			Implementation 02			Implementation 03		
								
<b>Sprint</b>	<b>30%</b>	<b>Duration</b>	<b>Sprint</b>	<b>70%</b>	<b>Duration</b>	<b>Sprint</b>	<b>100%</b>	<b>Duration</b>
<b>Set up and configure database</b>		<b>01 Week</b>	<b>Server-side Logic Development</b>		<b>01 Week</b>	<b>Integration testing</b>		<b>01 Week</b>
<b>Creating user interfaces</b>		<b>01 Week</b>	<b>API Implementation</b>		<b>01 Week</b>	<b>Acceptance Testing</b>		<b>01 Week</b>
<b>Complete User interface</b>		<b>01 Week</b>	<b>Server Route Creation</b>		<b>01 Week</b>	<b>Machine Learning Model Evaluation</b>		<b>01 Week</b>
<b>Machine learning model development</b>		<b>01 Week</b>	<b>Business Logic Implementation</b>		<b>01 Week</b>	<b>Deployment Preparation</b>		<b>01 Week</b>

*Figure 1 - Sprint Diagram*

For each sprint mentioned earlier, we have defined specific sprint backlogs that outline the tasks and deliverables for each sprint. These sprint backlogs are presented in tabular form to provide a clear overview of the work to be accomplished.

As we already mentioned, our project has three distinct phases: first implementation, second implementation, and third implementation. Each implementation phase has a duration of one month allocated for completion.

During the initial phase of the project, our primary goal is to complete 70% of the functional requirements for our software application. To achieve this objective, we have created a sprint backlog, represented in Table 1, which outlines the specific tasks and deliverables that need to be completed. The sprint backlog comprises the prioritized list of activities.

In the second phase of the project, our main objective is to finalize the remaining 30% of the functional requirements for our software application. This phase will also involve conducting comprehensive unit testing for the entire application to ensure its quality and reliability.

To accomplish these objectives, we will create another sprint backlog that outlines the specific tasks and deliverables to be completed during this phase. The backlog will include activities such as implementing the remaining functional requirements, designing, and executing unit tests to validate the application's behavior and functionality, and conducting experiments and analysis for the research components.

By following the sprint backlog, we ensure a systematic approach to completing the remaining functional requirements, performing thorough unit testing, and executing the research components. This approach allows us to meet the project's objectives, deliver a high-quality software application, and incorporate any valuable findings from the research components into the final product.

In the final phase of the project, our focus will be on conducting the integration testing process. This entails combining and testing various modules of the application to ensure their proper functionality and seamless integration. To facilitate this, we will create an integration testing sprint backlog that outlines the specific tasks and deliverables for this phase.

Once the integration testing is successfully completed, we will proceed to provide the application to our stakeholders for acceptance testing. During this phase, the stakeholders will actively use the application, provide feedback, and validate its suitability for their requirements. Their input will be invaluable in identifying any potential issues or areas for improvement.

## **1.5. Standards to be Followed**

Ensuring safety and efficiency in railway operations is paramount, requiring innovative solutions that leverage technology and data-driven insights. Our project focuses on developing a web application tailored specifically for railway safety, object detection, real-time weather updates, and hazard identification. By integrating advanced algorithms, real-time data streams, and user-friendly interfaces, our goal is to enhance safety protocols, improve operational decision-making, and mitigate risks within the railway industry.

### **Coding standards**

We adhere to IEEE coding standards to ensure clarity and consistency in our code. This includes using clear variable and function names, consistent formatting, and commenting to explain complex parts. We manage errors calmly, break down code into reusable parts, and use version control tools like Git. We also employ techniques such as Test-Driven Development (TDD) and Continuous Integration (CI) to ensure code quality.

### **Data standards**

We follow IEEE data standards to maintain integrity and security in our railway safety application. This involves using standardized data models, implementing data quality checks, and employing encryption and access controls. Techniques such as data anonymization and pseudonymization are also used to protect sensitive information.

### **Document Standards**

Our documentation adheres to IEEE standards, providing clear instructions, code explanations, and API documentation. We maintain version history, ensure accessibility, and collaborate to keep documentation up to date. We also follow techniques like Single Sourcing and Documentation as Code to streamline the documentation process.

### **System security standards**

We implement IEEE system security standards to protect sensitive data in our railway safety application. This includes encryption, access controls, and robust authentication mechanisms. We also employ techniques such as threat modeling and security testing to identify and mitigate potential vulnerabilities.

## **1.6. Organization of the SDS**

### **1. Introduction**

Our System Design Specification details the architecture and design of our railway safety web application, emphasizing object detection, real-time weather updates, and hazard identification. The aim is to improve railway safety through accurate information and proactive risk management tools. This document outlines the specifics of each component, ensuring a user-friendly and robust system.



## **2. Architectural Design**

The architectural design of our railway safety web application integrates advanced algorithms tailored for object detection, leveraging computer vision techniques for identifying obstacles like elephants, bulls, vehicles, and humans near railway tracks, as well as algorithms for real-time weather analysis, hazard prediction, and data management, ensuring precise and timely information to enhance railway safety protocols and operational efficiency.

## **3. UI Design**

The UI (User Interface) design of our railway safety web application focuses on creating intuitive and user-friendly interfaces for railway personnel to interact with, incorporating features such as interactive maps for object detection alerts, real-time weather updates displayed in a visually clear format, hazard identification notifications with actionable steps, and secure login screens with access controls tailored to different user roles, ensuring a seamless and efficient user experience that enhances railway safety protocols and operational efficiency.

## **4. Data Management**

Data management in our railway safety web application involves designing robust systems for storing, organizing, and accessing data related to object detection, real-time weather updates, hazard identifications, locomotive pilot details, administrative data, and user interactions. This includes implementing database schemas, data models, and storage mechanisms optimized for performance, scalability, and data integrity, ensuring that critical information is securely managed, efficiently retrieved, and utilized to enhance railway safety protocols and operational decision-making.

## **5. Research Design**

The research design for our railway safety web application involves defining the methodology, data collection methods, analysis techniques, and evaluation criteria used to investigate and validate the effectiveness of our object detection algorithms, real-time weather forecasting models, hazard identification systems, user interface designs, and security measures. This includes conducting experiments, gathering data from railway scenarios, analyzing results, and iteratively refining our system based on research findings to ensure robustness, accuracy, and usability in enhancing railway safety protocols and operational efficiency.

## 2. Architectural Design

Develop advanced object detection algorithms for the railway track to identify animals, vehicles, and hazards accurately. Ensure reliability in different environmental conditions. Integrate cloud-based technologies and real-time data sources to provide up-to-date information on geographical position and weather conditions to locomotive pilots. Deliver timely updates on factors like precipitation and wind speed. Create an efficient system for processing live camera feeds promptly, providing feedback on detected objects, location, and weather to pilots. Design an intuitive web interface displaying real-time information in a user-friendly manner, including voice-enabled functionality for seamless interaction.

### 2.1. System Architecture

Our railway safety web application is designed to enhance safety protocols and operational efficiency in railway systems. Key components of our application include object detection for identifying obstacles, real-time weather updates for informed decision-making, hazard identification for proactive risk management, user interface components for intuitive interaction, data storage solutions for efficient data management, and security modules for protecting sensitive information. This session focuses on the architectural design of our application, outlining how these components interact and contribute to the overall functionality and performance of the system.

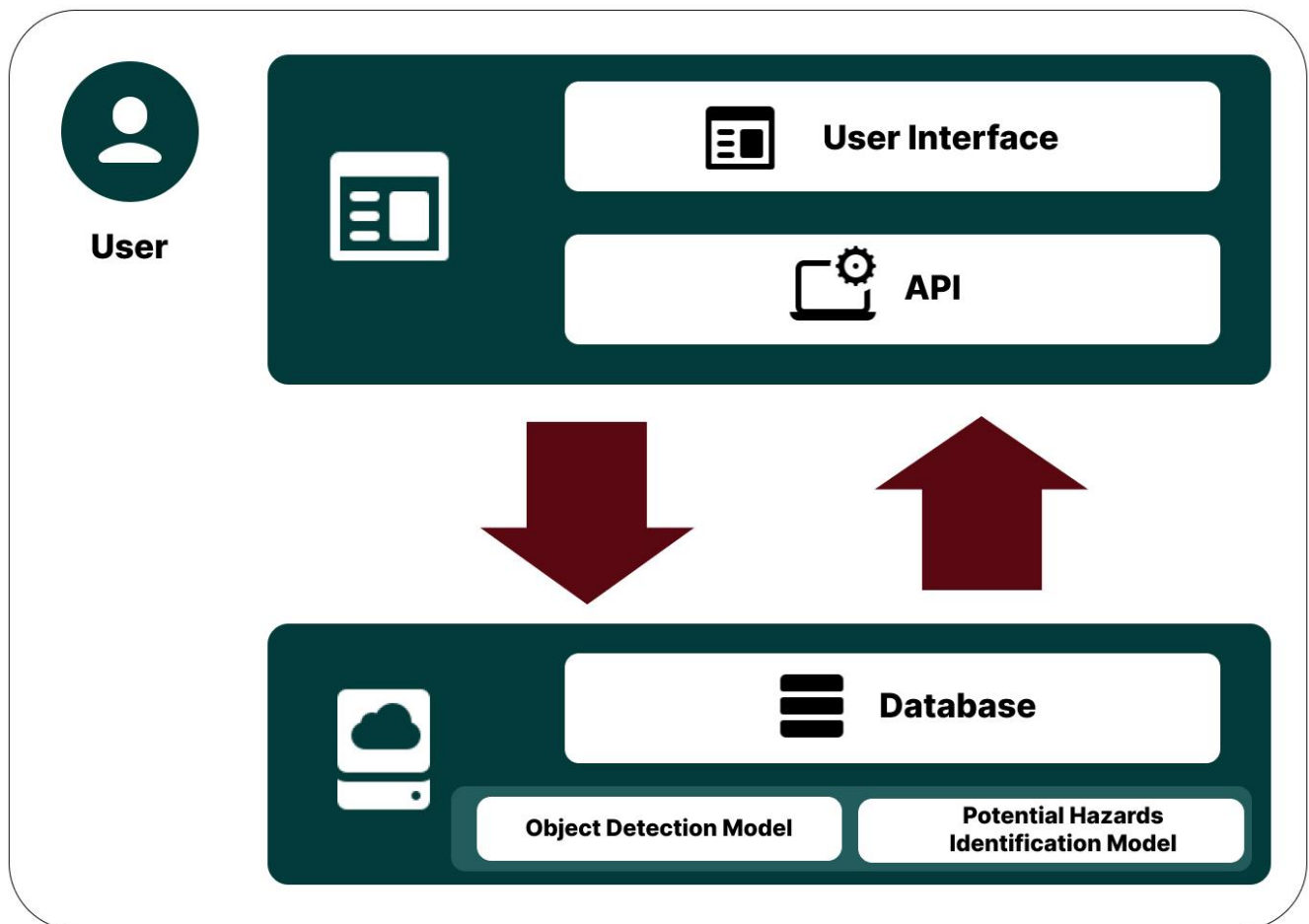


Figure 2- Architecture design

- Frontend: In your system, the frontend is customized for train pilots who exclusively access the system while on trains. Admins, on the other hand, have the flexibility to access the system from any location to manage and monitor the system's overall functionality and data. This involves creating an easy-to-use interface for pilots to access essential information during their journeys, with real-time maps and alerts providing updates on train locations, weather conditions, and potential hazards along the railway tracks. Additionally, the system employs robust security measures including data encryption and

authentication protocols to safeguard sensitive details such as locomotive pilot information and admin credentials, ensuring a secure experience for all users accessing the system, whether onboard trains or remotely as administrators.

- **Backend:** In the backend process, we establish a secure database to store critical information about locomotive pilot and system administrators. We integrate real-time data sources and alert systems for immediate hazard notifications. Additionally, we manage sensor data, such as train locations and weather conditions, ensuring accuracy and reliability. To safeguard this data, we implement encryption and access control measures, which are advanced security codes. Lastly, we leverage cloud services to optimize system performance and scalability.

The railway safety web application architecture follows a three-layered structure, comprising the presentation layer, the application layer, and the data layer.

- **Presentation layer:** The Presentation Layer in your system is how train pilots interact with the system while on trains. It includes screens, displays, and controls for pilots to access important information during their trips. This layer focuses on showing data clearly, making it easy for pilots to use the system. Pilots can check train locations, weather conditions, and receive alerts about hazards. The Presentation Layer is designed to be user-friendly, with features like real-time updates, interactive maps, and voice commands. It acts as a bridge between the backend (handling data processing and storage) and the pilots, making communication smooth. Additionally, it's designed to not disrupt pilots during train movement and avoids the need for scrolling by fitting everything on one screen that adjusts to the screen size. This ensures a seamless experience for pilots throughout their journey.
- **Application layer:** Your system is designed to handle various tasks smoothly, issuing immediate alerts for potential issues to ensure timely responses to hazards. It gathers real-time weather data, which helps train operators make better decisions. User access is carefully managed, controlling who can access sensitive information and perform specific actions within the system. Additionally, the system maintains detailed logs of admin interactions, providing a comprehensive record of actions taken for security, auditing, and operational analysis. These functionalities work together to enhance system efficiency, security, and accountability, ultimately ensuring smooth and reliable train operations.
- **Data layer:** The Data Layer in our system is crucial for managing key information related to train operations and system functionality. It stores essential train data like routes, current locations, and operational status, helping monitor train movements and manage logistics efficiently. Additionally, it manages user details such as credentials, permissions, roles, and activity logs for secure access and accountability. System configurations are stored here too, allowing customization of settings like notifications and alerts for smooth system functionality. Real-time weather data stored in this layer assists in decision-making for train operations based on current weather conditions. Moreover, detailed activity logs track user actions and system events, enhancing accountability and facilitating system performance monitoring.

## 2.2. Objects and Communication

The class diagram is essential for our railway safety web application as it provides a visual representation of the system's structure and helps us organize and understand the relationships between different classes. It serves as a blueprint for the application's design, facilitating the implementation and maintenance of the software.

In our proposed application, we have identified several key classes that play crucial roles.

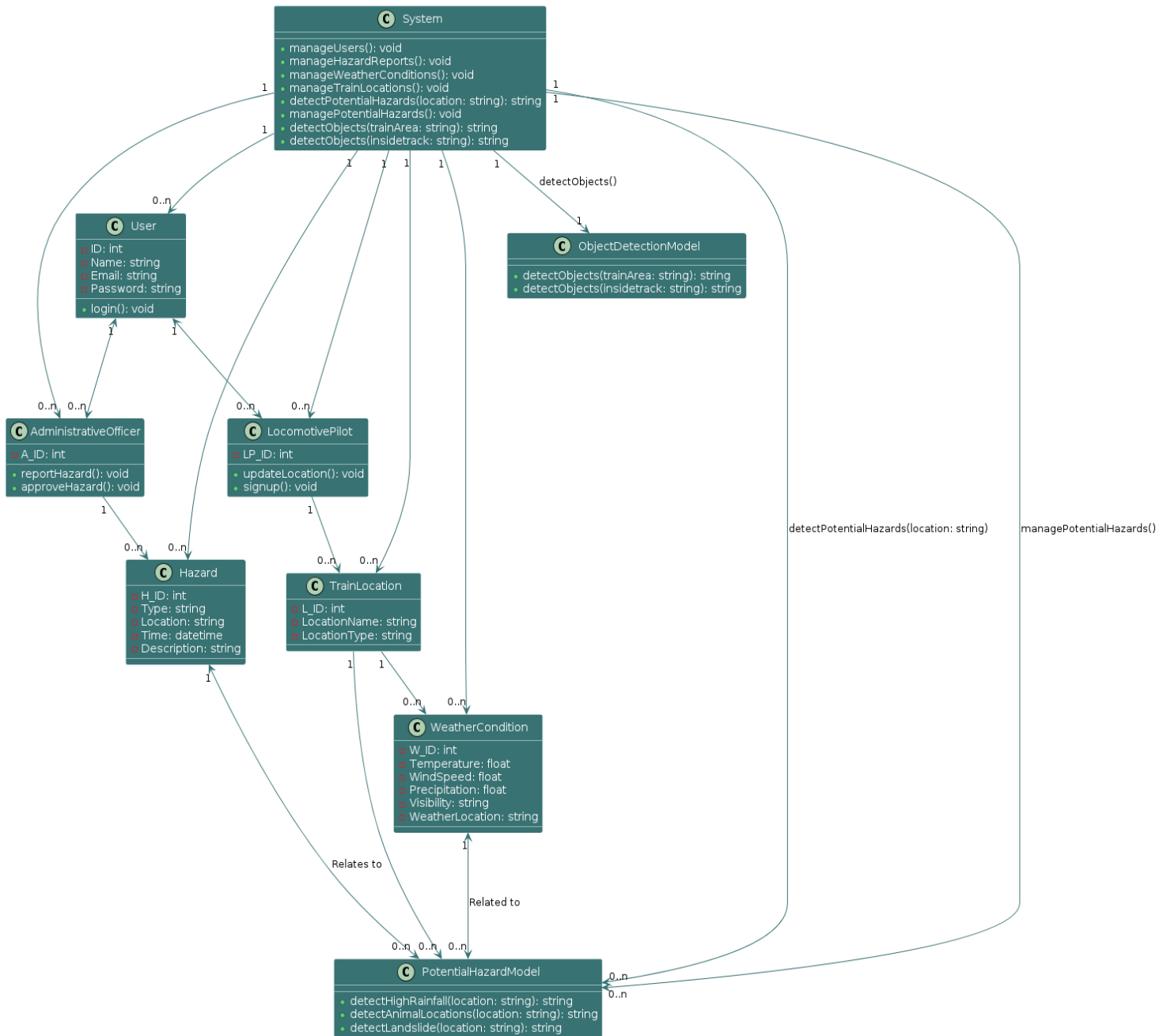


Figure 3 - Class Diagram

The class diagram serves as a crucial tool for visualizing the system's architecture, illustrating the relationships between various classes, and facilitating the design, implementation, and maintenance of the Railway Safety Enhancement System.

- **User:** Represents users of the system with attributes like ID, Name, Email, and Password, along with methods like login().
- **AdministrativeOfficer:** Handles hazard reporting and approval with a unique identifier (A\_ID) and methods reportHazard() and approveHazard().
- **LocomotivePilot:** View train locations and done sign-ups, identified by LP\_ID, and methods updateLocation() and signup().
- **Hazard:** Represents hazards with attributes like H\_ID, Type, Location, Time, and Description.
- **WeatherCondition:** Stores weather-related data such as temperature, wind speed, precipitation, visibility, and WeatherLocation.

- **TrainLocation:** Manages train location details with attributes L\_ID, LocationName, and LocationType.
- **ObjectDetectionModel:** Detects objects in train areas with methods detectObjects(trainArea) and detectObjects(insideTrack).
- **PotentialHazardModel:** Detects potential hazards like high rainfall, animal locations, and landslides, with methods for detection and management.
- **System:** Represents the core system functionality with methods for managing users, hazard reports, weather conditions, train locations, potential hazards, and object detection.

### 2.2.1. Dynamic Behaviors

Dynamic behaviors are pivotal in the functionality of our railway safety web application, ensuring real-time responsiveness and accurate decision-making capabilities. These behaviors encompass actions such as real-time data updates, object detection alerts, hazard identifications, user interactions, and system responses. By dynamically analyzing incoming data, generating alerts, and facilitating user interactions, our system empowers railway personnel with timely information and actionable insights to enhance safety protocols and mitigate risks effectively.

#### i. Sequence Diagram for Registration

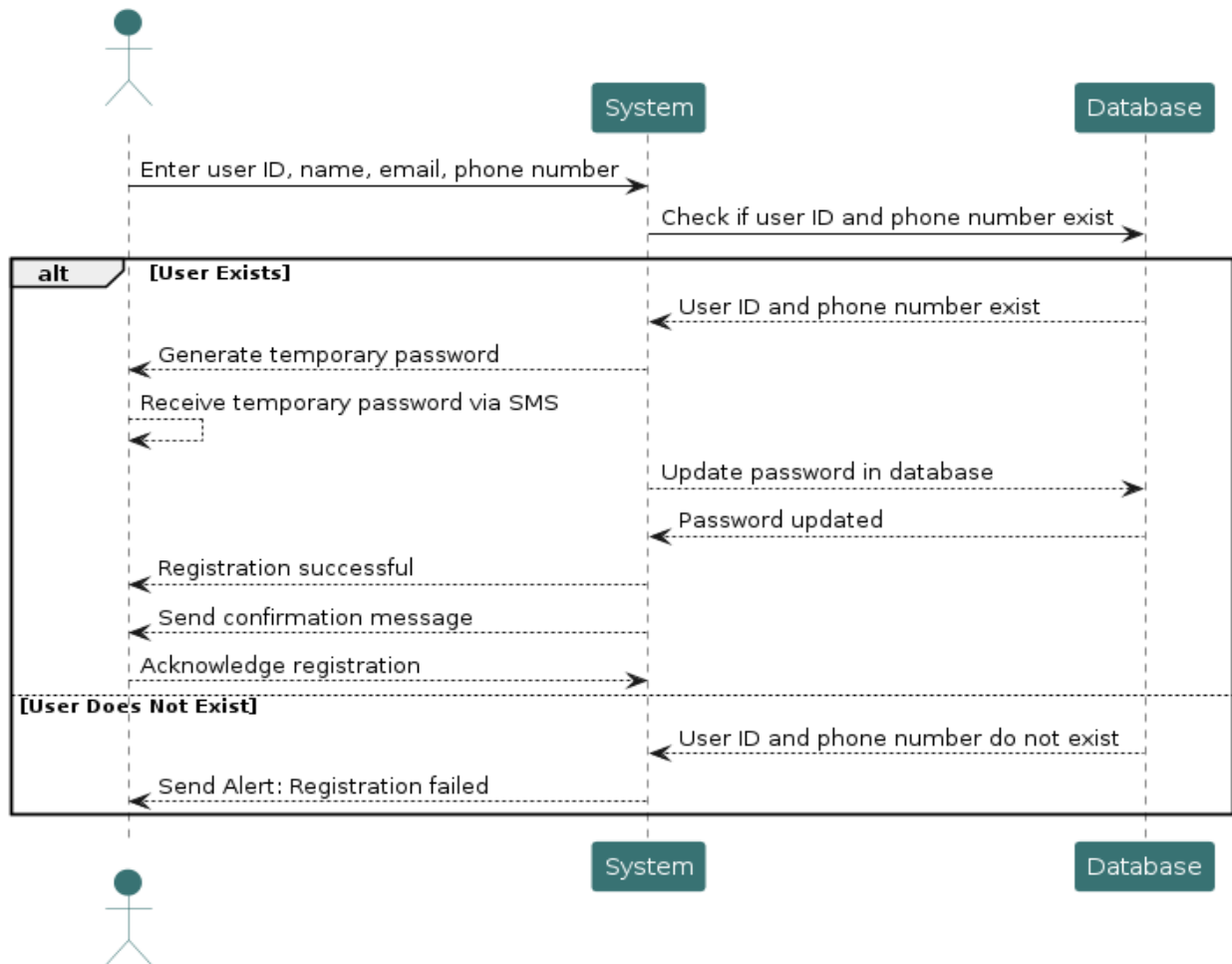


Figure 3 - Registration

The registration sequence diagram for our railway safety web application illustrates the process initiated when a locomotive pilot enters the registration form. The pilot provides essential details such as locomotive ID, name, email, and phone number.

The system then checks if the provided locomotive ID and phone number already exist in the database. If they do, and if the details match the records related to the railway department, the system allows the pilot to proceed with registration. However, if the locomotive ID or phone number is already associated with an existing account or if there is a mismatch with the railway department records, the system notifies the pilot that registration cannot proceed.

Upon successful registration, the pilot receives a temporary password through their phone number. This temporary password allows the pilot to log in to the system for the first time.

These checks ensure data accuracy and maintain the security of the registration process, preventing unauthorized persons who are not related to the railway department from registering.

## ii. Sequence Diagram for Login

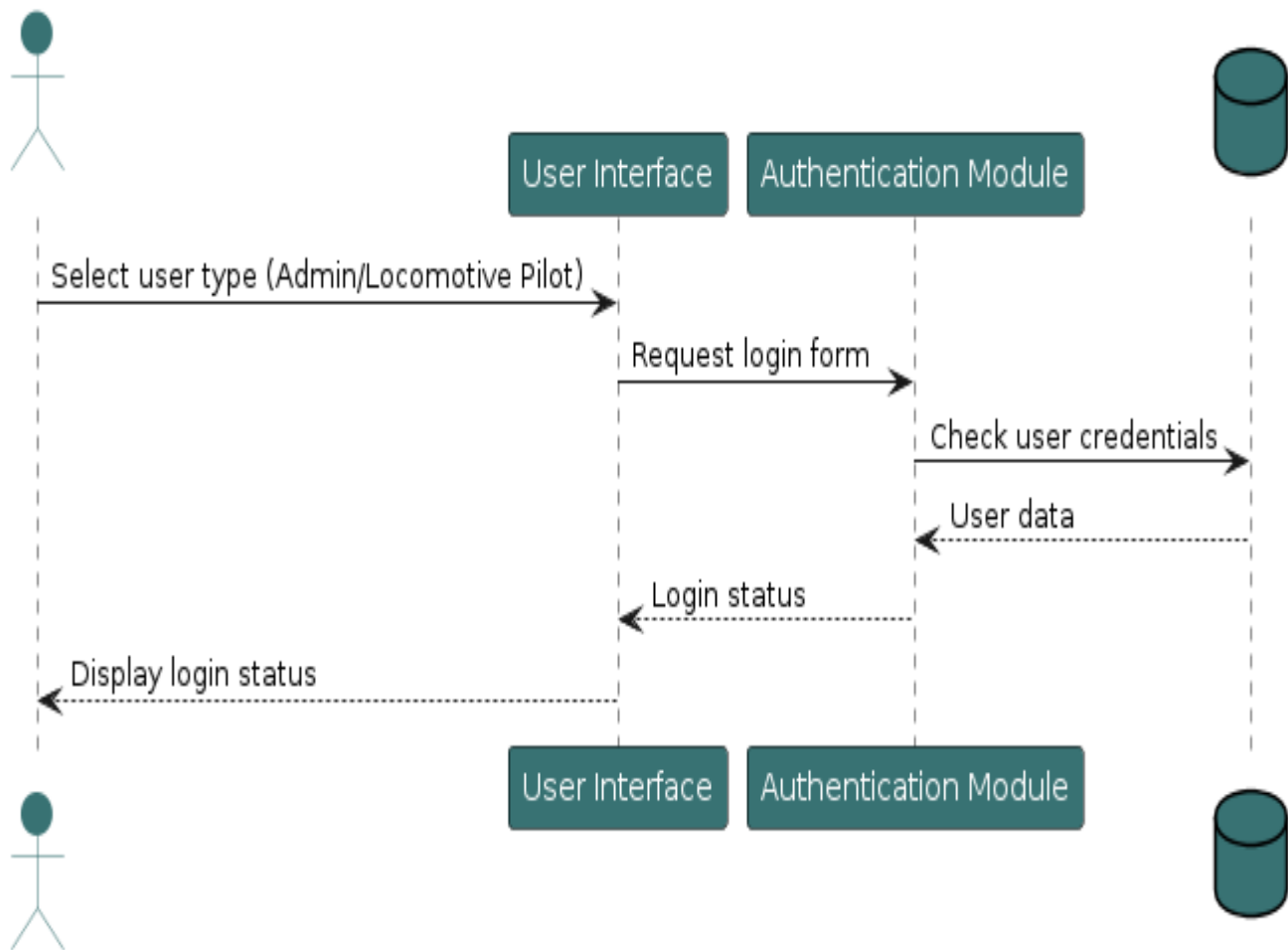


Figure 4 - Login

The login process for our railway safety web application involves the user selecting either 'Admin' or 'Locomotive Pilot' on the login page. The user then provides their user ID and password.

If the user's credentials are correct, the system allows them to proceed to the next page. However, if the username or password is incorrect, an error message is displayed.

In cases where the user forgets their password, they can navigate to the 'Forgot Password' page. Here, the system automatically sends an OTP (One-Time Password) to the user's registered phone number without requiring the user to enter their phone number again. Upon entering the correct OTP, the user is directed to a page where they can reset their password.

This login process ensures secure access for both admins and locomotive pilots, with seamless OTP delivery for password recovery.

### iii. Sequence Diagram for Reset Password

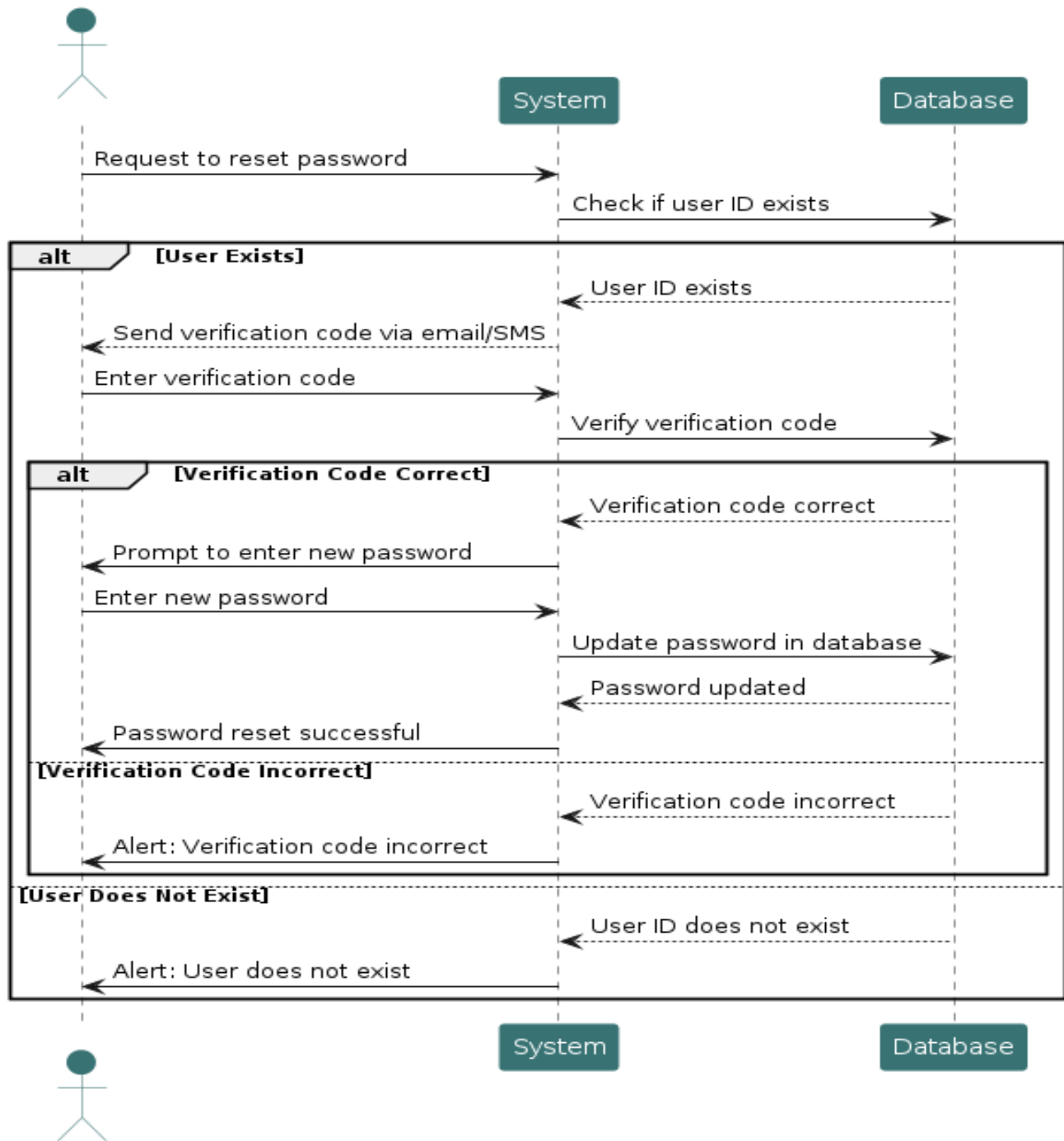


Figure 5 – Reset Password

The process begins when a user requests a password reset by entering their user ID on the reset password page. The system checks if the user ID exists in the database and, if so, generates a one-time password (OTP) sent to the user's registered phone number via SMS. Upon receiving the OTP, the user enters it on the password reset page to verify their identity. The system then validates the OTP and prompts the user to enter a new password, which is validated for length, complexity, and uniqueness. If the new password meets the criteria and the OTP verification is successful, the system updates the user's password in the database, confirming the password reset's success. This process ensures a secure and user-friendly way for users to reset their passwords using an OTP sent to their phone number for identity verification.

#### iv. Sequence Diagram for Object Detection

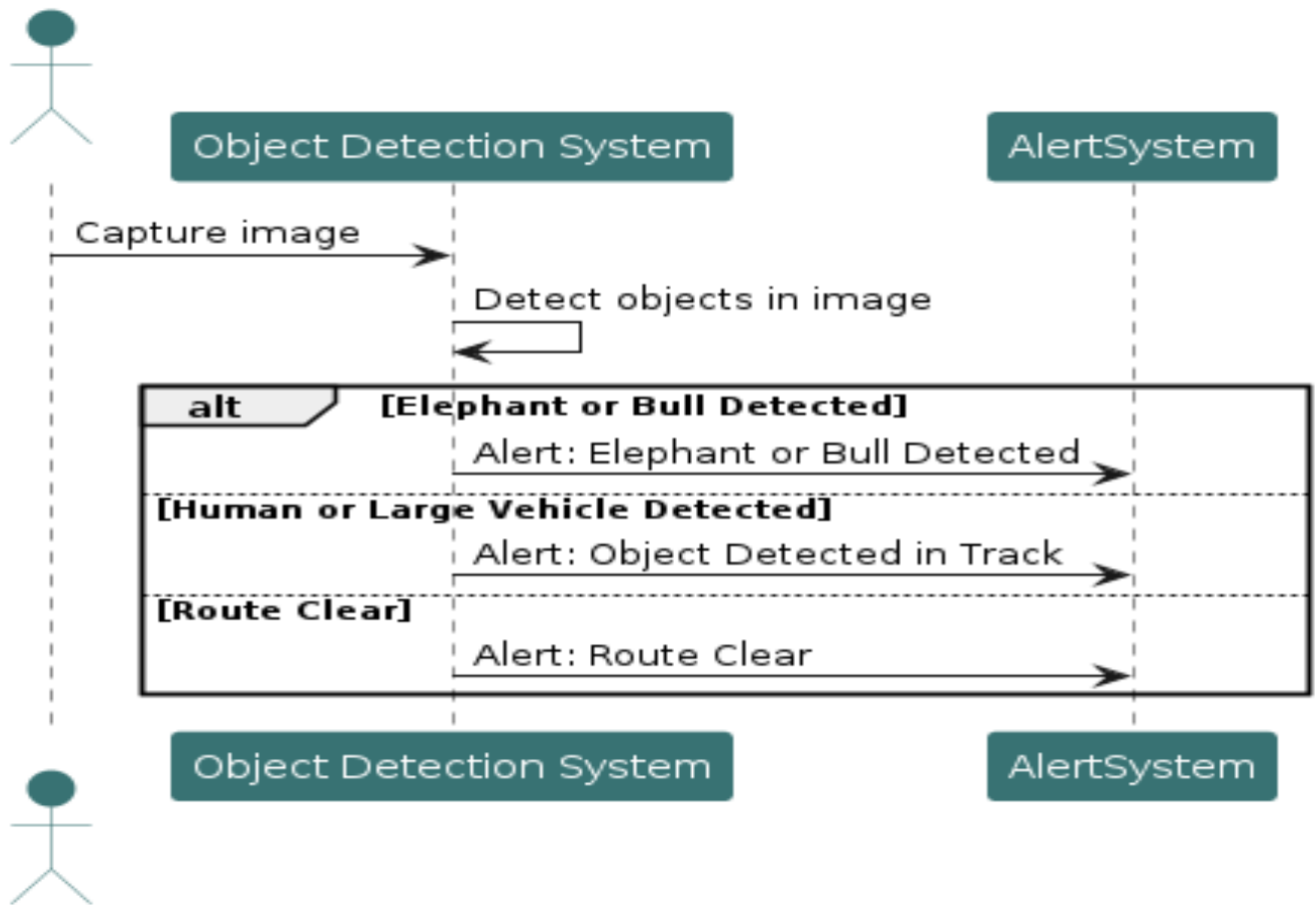


Figure 6 – Object Detection

The sequence diagram for Object Detection in our railway safety web application starts with the Object Detection Module initiating the detection process. It retrieves image frames from the Image Frame Dataset and sends them for processing through image processing algorithms. Once objects are detected in the processed frames, the Object Detection Module communicates the results to both the User Interface for visual alert display and the Voice Alert system for auditory alerts.



v. **Sequence Diagram for Weather Forecasting**

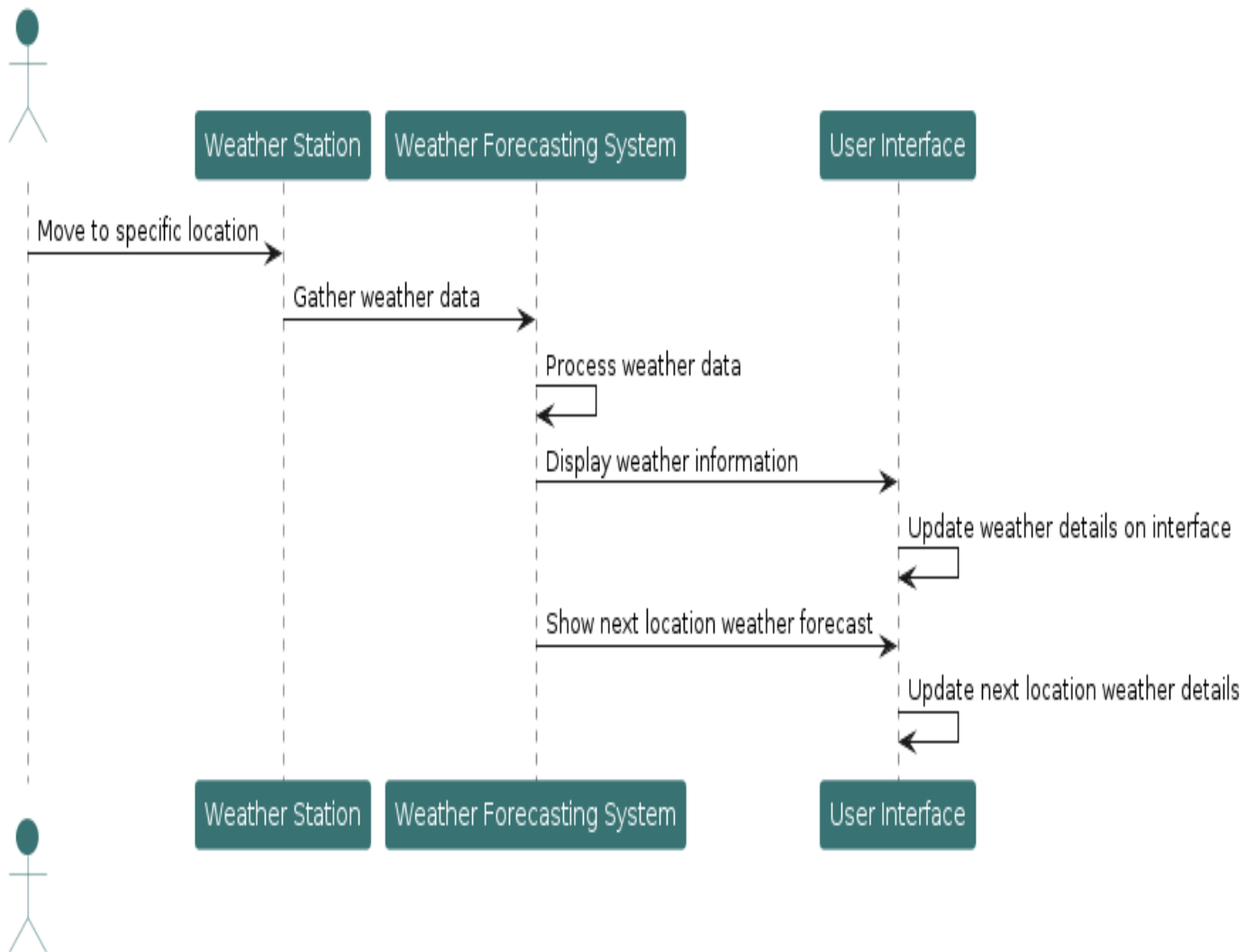


Figure 7 - weather forecasting

The sequence diagram for weather prediction in our railway safety web application begins with the Actor initiating the process by requesting weather data from the Real-time Weather Update Service. The service fetches current weather data from the Weather API, which then provides the data back to the service. The Real-time Weather Update Service utilizes a Weather Prediction Algorithm to forecast the next weather conditions, sending the predicted data back to the service. Finally, the service communicates the predicted weather information to the User Interface, where it is displayed for users. This sequence illustrates the flow of events from data retrieval to prediction and visualization, enhancing real-time decision-making for railway safety.

## vi. Sequence Diagram for Potential Hazard

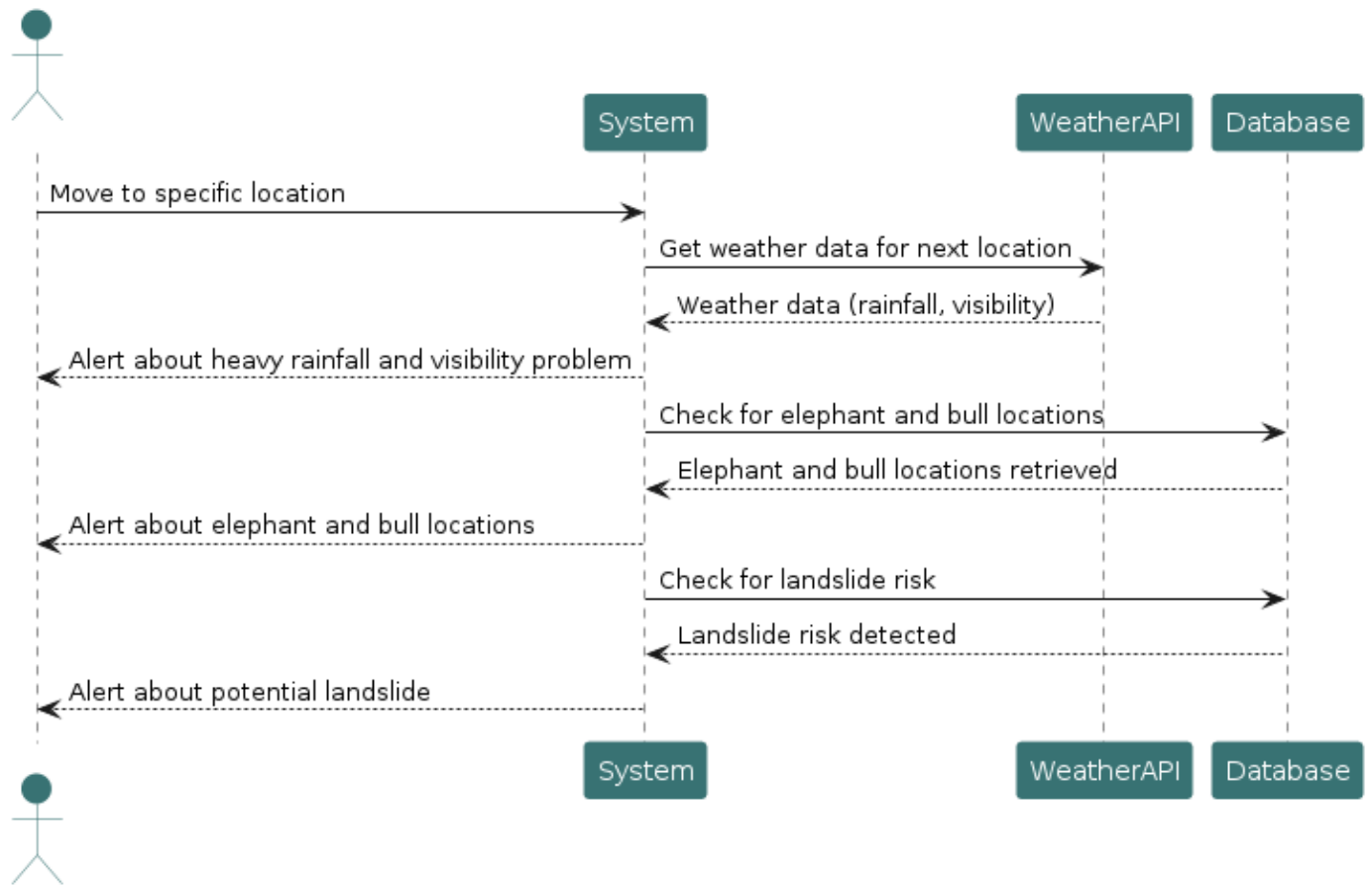
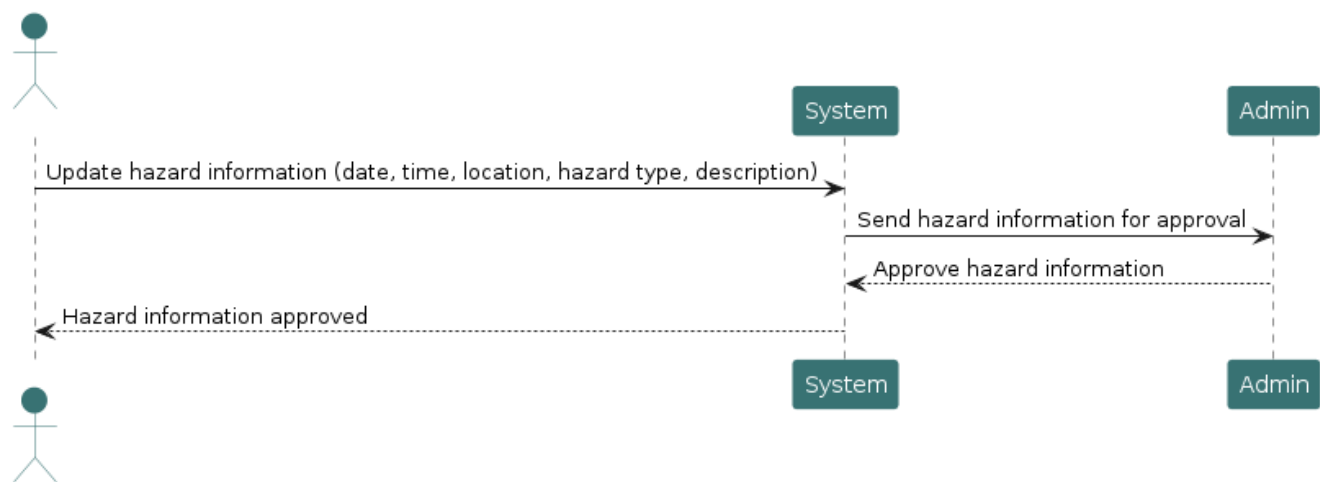


Figure 8 - Weather Forecasting

In our updated potential hazard identification process for the railway safety web application, the Hazard Identification System can detect various weather-related hazards. If the rainfall exceeds 120mm in the selected location, as indicated by data from the landslide database, signaling a potential landslide risk, or if heavy mist or fog occurs, indicating low visibility conditions, the system promptly sends alerts to the Real-time Weather Update Service. Upon receiving this information, the service relays it to the User Interface Components. Subsequently, the components promptly issue appropriate alerts through voice and interface, such as a landslide alert for high rainfall situations or a low visibility alert for mist or fog, ensuring users receive timely information to take necessary precautions.

**vii. Sequence Diagram for Update Hazard**



*Figure 9 – Update Hazard*

The sequence diagram for hazard entry and approval in our railway safety web application begins with the locomotive pilot entering hazard data through the User Interface. The User Interface saves this data temporarily in the Hazard Database and notifies the Admin Interface for approval. If approved by the admin, the hazard data is saved permanently in the database. However, if not approved within the specified time limit, an expiry timer triggers the automatic deletion of the unapproved hazard data from the Hazard Database. This sequence ensures that hazards are managed effectively, with approved entries being retained and unapproved entries being automatically removed to maintain data integrity and safety standards within the application.

### viii. Sequence Diagram for Approve Hazard

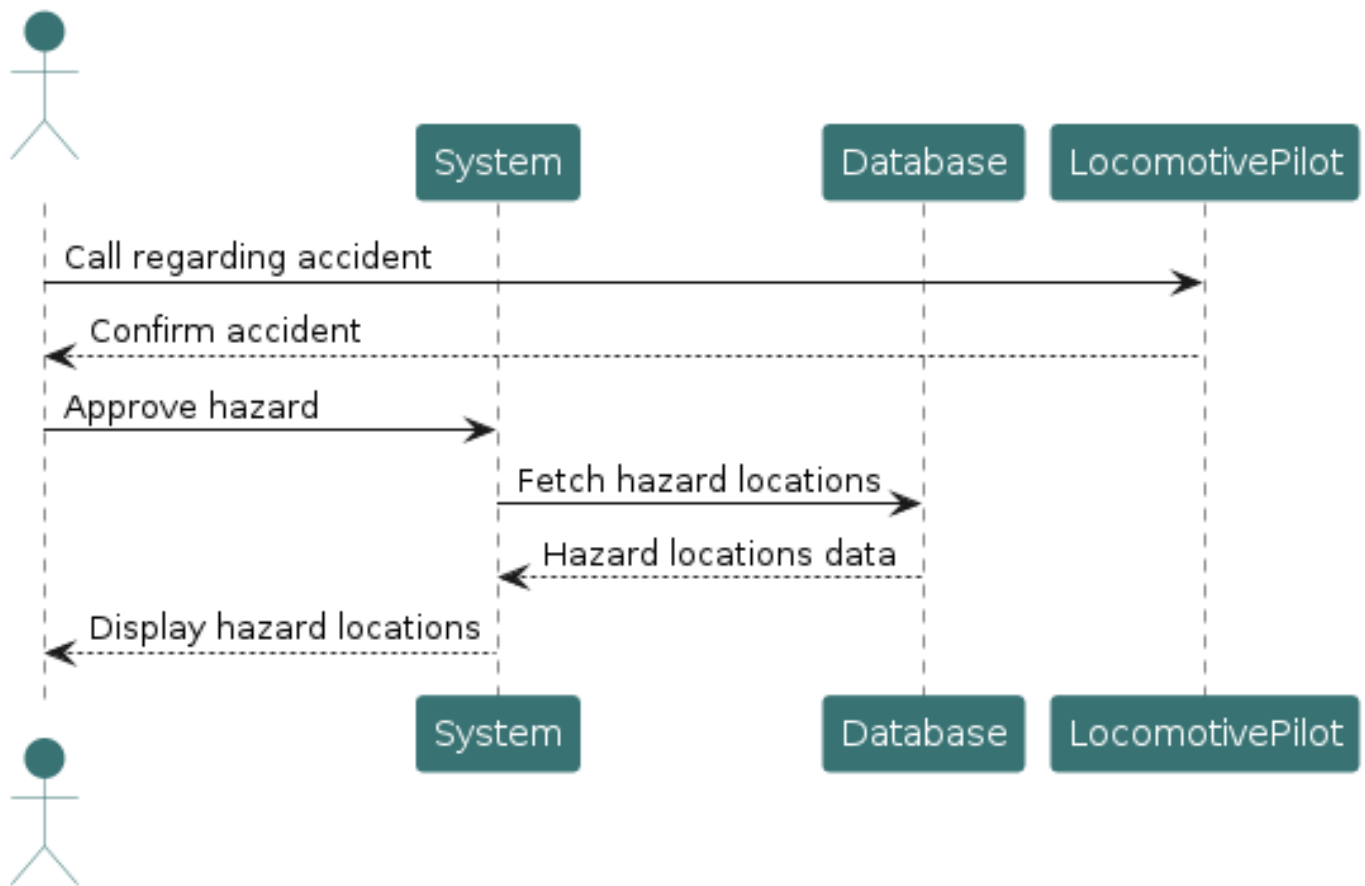


Figure 10 – Approve Hazard

The sequence diagram for approving hazards in our railway safety web application incorporates an additional step for the admin to confirm whether the hazard occurred or not, using phone or email communication. After approving a hazard through the Admin Interface and updating its status in the Database, the Admin Interface requests confirmation via the Hazard Monitoring System, which offers options for confirmation such as phone or email. The admin then confirms the hazard status through the chosen method, leading to an update in the hazard confirmation status in the Database. Finally, the Admin Interface displays the confirmed hazard status on the User Interface, ensuring a thorough process for hazard approval and confirmation within the application.

### 2.3. Component Diagram

The component diagram for our railway safety web application represents the interconnected components that collaborate to ensure the efficiency and reliability of our system. These components encompass key functionalities such as object detection, hazard identification, real-time weather updates, user interface elements, backend logic, data management, and external API integrations. The diagram illustrates how these components interact with each other to deliver a comprehensive railway safety solution, empowering railway personnel with real-time data insights, hazard alerts, and intuitive user interfaces for effective decision-making and risk management.

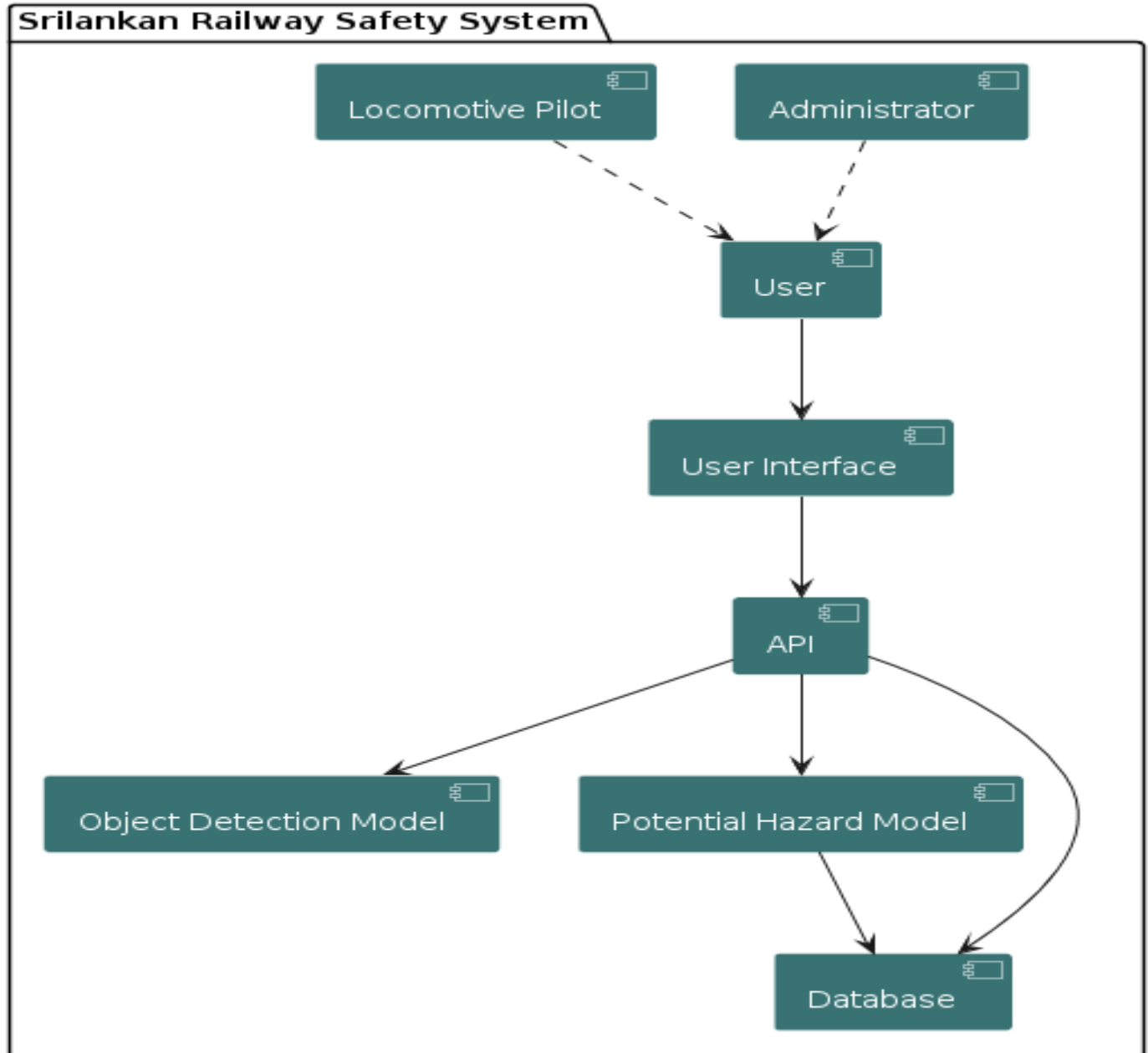


Figure 11 - Component Diagram

- **User Interface:** The User Interface component represents the graphical interface through which users interact with the railway safety system. It provides functionalities such as user login, access to system features, and displaying alerts and notifications. Allows locomotive pilots and administrators to log in and access their respective functionalities. Provides a user-friendly interface for viewing object detection results, potential hazard alerts, and system status. Displays alerts and notifications for important events such as hazard detection, system updates, or critical warnings.
- **Object Detection Model:** The Object Detection Model component is responsible for detecting and identifying objects on the railway track using cameras. It plays a crucial role in ensuring track safety by detecting animals like bull and elephant, large vehicles and human.

- **Potential Hazard Model:** The Potential Hazard Model component utilizes data from the Database, weather sensors, and track monitoring systems to analyze railway track conditions. Using algorithms and logic, it identifies potential hazards like animal presence and adverse weather conditions. By generating real-time hazard alerts specifically for locomotive pilots, it empowers proactive measures, ensuring railway safety and accident prevention.
- **Database:** The Database component is the central repository in the railway safety system, managing diverse data related to users, hazards, system configurations, and historical records. It stores user profiles and authentication details, along with access permissions for locomotive pilots and administrators. Additionally, the Database maintains records of potential hazard alerts historical data, facilitating analysis, reporting, and informed decision-making. Its functionality extends to supporting data retrieval and querying operations, ensuring efficient access to critical information and enabling the generation of insightful reports within the system.
- **User:** The User component represents the user roles within the railway safety system, including locomotive pilots and administrators. It manages user authentication, access permissions, and user interactions with the system. Provides user authentication and authorization mechanisms for logging into the system and accessing functionalities based on user roles. Facilitates user interactions with the User Interface for performing tasks, viewing information, and receiving alerts.<sup>3</sup>
- **API:** The API (Application Programming Interface) component defines the methods, protocols, and data formats for communication and integration between system components and external systems. It enables external applications or services to access functionalities and data within the railway safety system. Defines APIs for accessing object detection results, hazard alerts, user data, and system configurations.

## 2.4. State Machines

### 1. State Register

In our system, the registration process starts when a user provides their personal information, such as user Id, password, email, and contact details. This information is then validated to ensure it meets security requirements, including password complexity and uniqueness of usernames. Once validated, the system creates the user account and sends a confirmation email to the registered email address, completing the registration process.

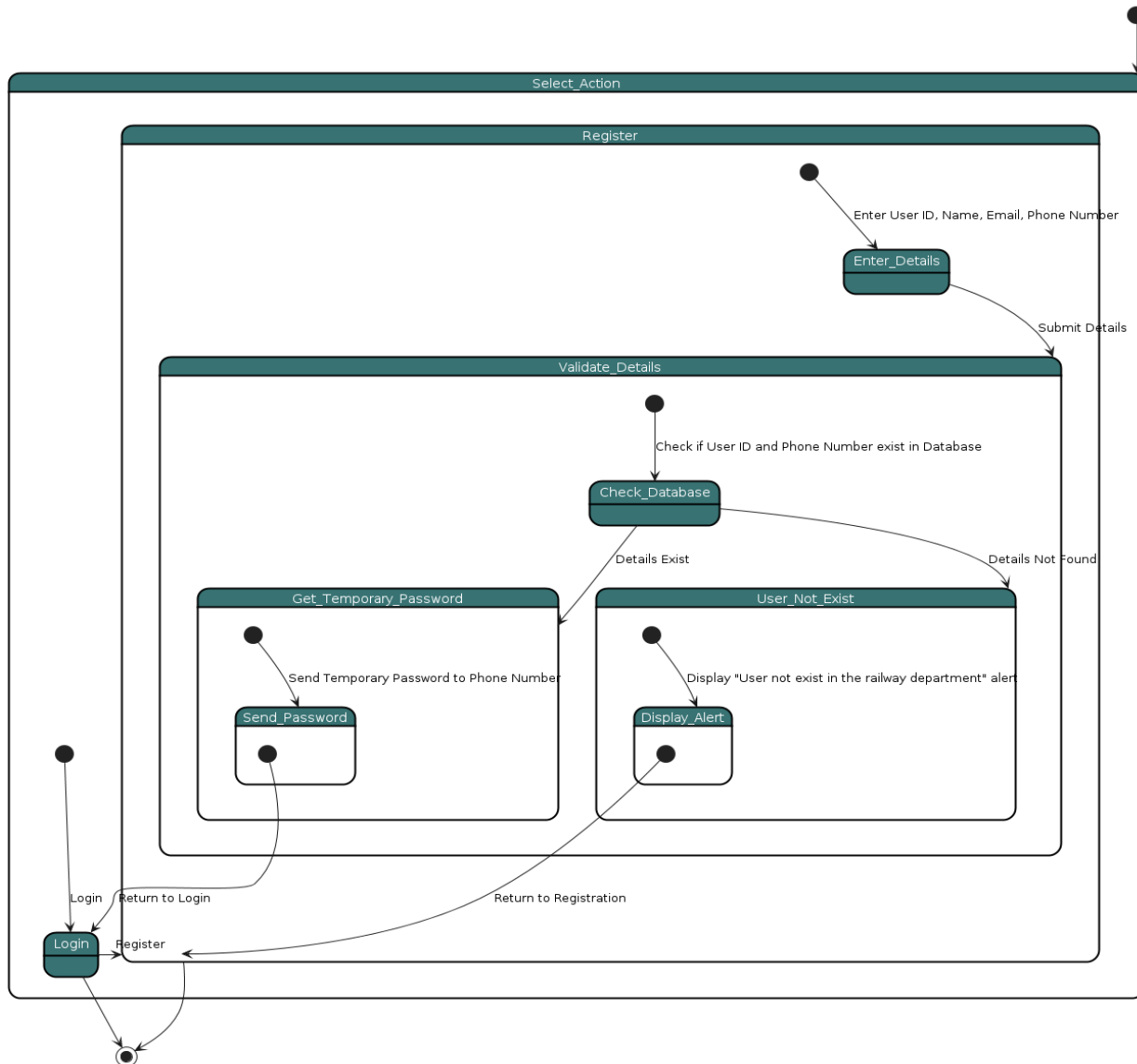


Figure 12- State machine 1

## 2. State2 login

In our railway safety system, the login process involves authorized personnel entering their unique userId and password into the system's secure interface. The system then conducts thorough validation checks to ensure the accuracy and security of the credentials, verifying them against the centralized database.

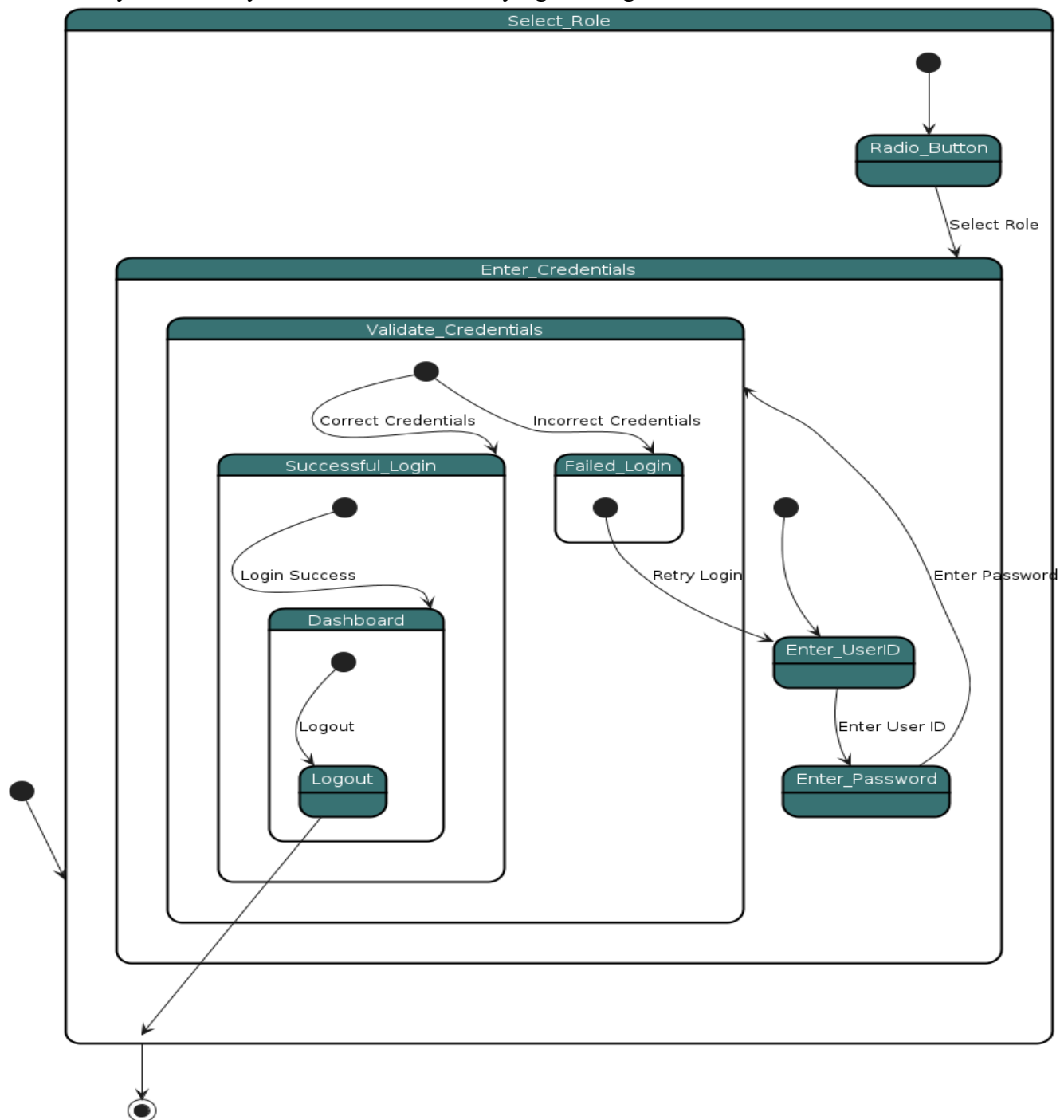


Figure 13 - State machine 2



### 3. State3 Reset Password

In our railway safety system, users can reset their password by following a secure process. When a user initiates a password reset, they receive a one-time password (OTP) on their registered phone number for verification. After entering the OTP into the designated field on the password reset page, the system validates the OTP to ensure its accuracy and authenticity. Upon successful verification, the user gains access to a page where they can securely create a new password. The user is prompted to enter their desired new password and confirm it to ensure accuracy. Once the new password is submitted and accepted by the system, the user can then log in using their updated credentials, providing enhanced security for accessing critical railway safety information and tools.

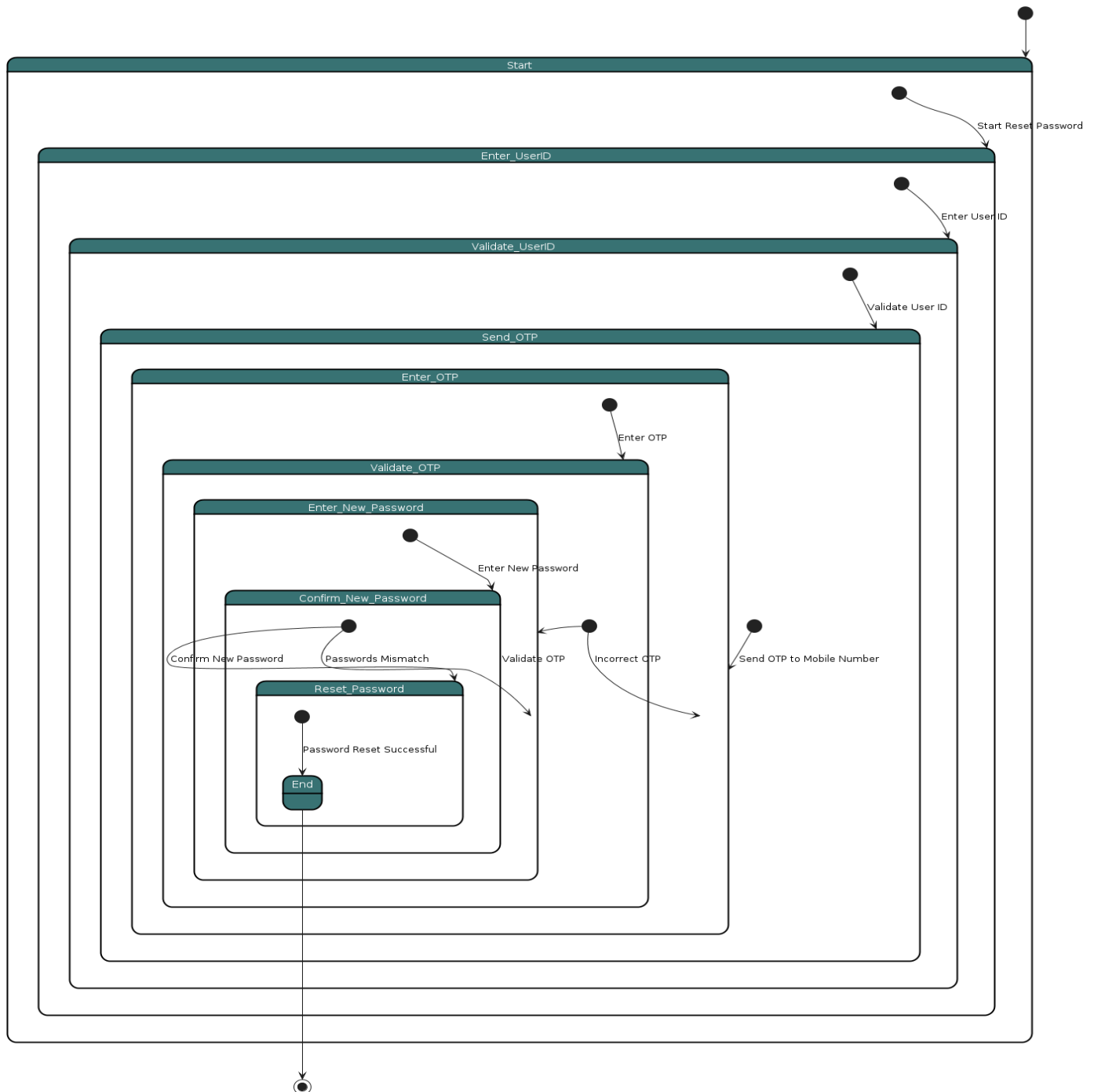


Figure 14 - State machine 3

#### 4. State 4 Object Detection

The object detection process in our railway safety system follows a systematic approach, starting with the acquisition of image frames from cameras placed strategically along the railway tracks. These images undergo preprocessing techniques such as resizing, normalization, and noise reduction to improve quality. Subsequently, computer vision algorithms extract relevant features from the images, allowing for the identification and classification of objects. This classification includes categorizing objects into groups such as bull, elephant, large vehicle, and human. When critical objects or hazards are detected, the system generates alerts in real-time, prompting swift action from railway personnel to ensure safety and prevent potential accidents.

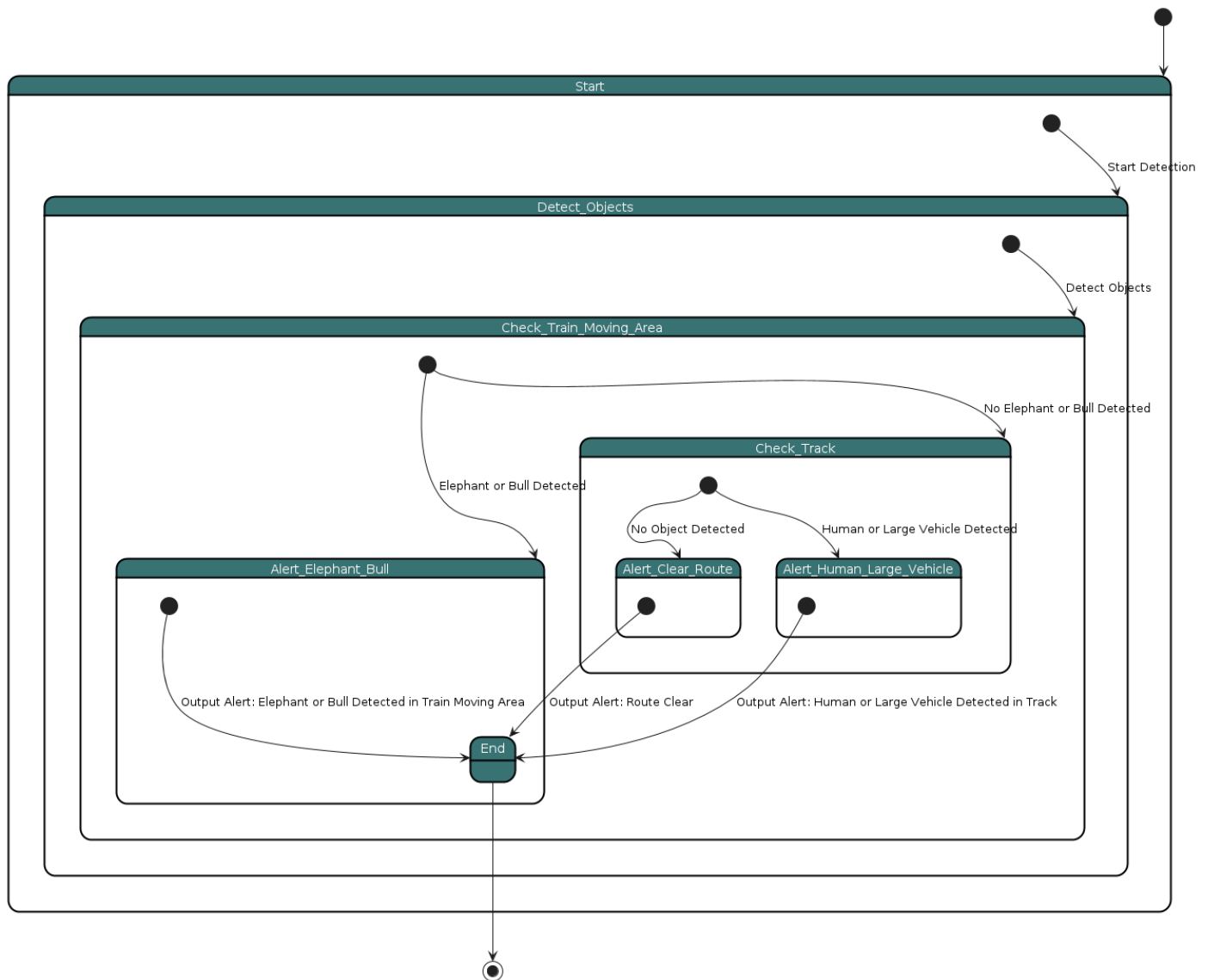


Figure 15 - State machine 4

## 5. State 5 Weather update

The weather update feature continuously gathers real-time meteorological data relevant to railway operations, including information such as temperature, precipitation, wind speed, visibility, and weather conditions. This data is analyzed and validated to ensure accuracy and reliability. Railway personnel can access this information to make informed decisions about train schedules, track maintenance, and safety protocols, particularly in response to adverse weather events like heavy rainfall, fog, or storms. Overall, the weather update feature enhances railway safety by enabling proactive risk management and efficient operational planning based on current weather conditions.

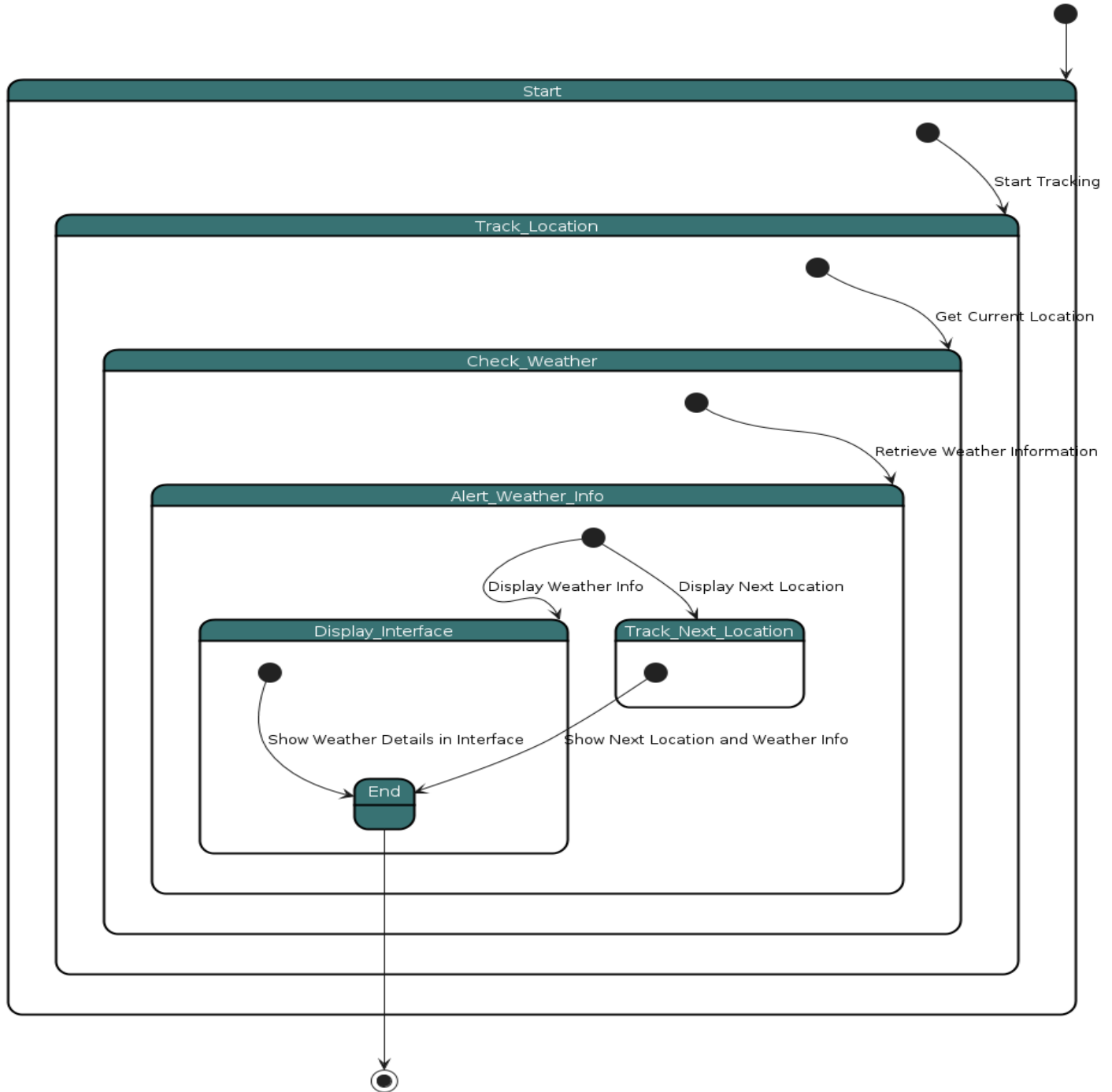


Figure 16 - State machine 5

## 6. State 6 Potantial Hazard

The hazard alert system in our railway safety framework operates through a series of states to monitor and detect potential risks along railway tracks. It begins by checking visibility conditions, identifying any issues during fog or mist conditions that could affect train operations. Next, it monitors rainfall levels, issuing an alert if rainfall exceeds 120mm, indicating a potential landslide risk. Additionally, the system detects the presence of wildlife such as elephants and bulls near the tracks, alerting railway personnel when trains approach these areas.

These checks and alerts are crucial for ensuring the safety and efficiency of railway operations. They prompt immediate actions, such as adjusting train schedules, implementing safety protocols, or diverting trains to alternative routes, to mitigate risks and prevent accidents. Overall, the hazard alert system plays a vital role in enhancing railway safety by providing real-time notifications of potential hazards.

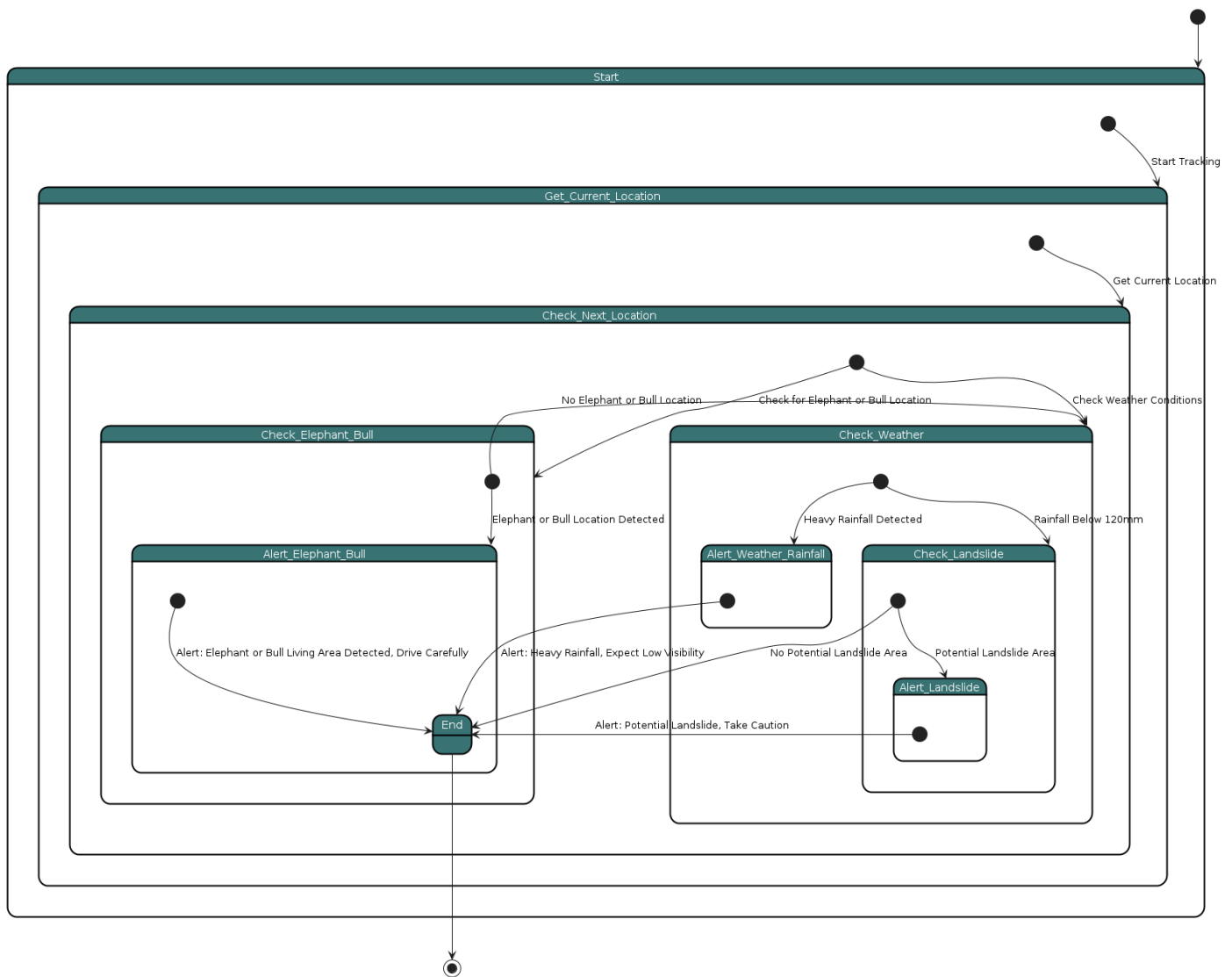


Figure 17 - State machine 6

## 7. State 7 Update Hazard

When a train driver has an accident, they quickly enter details like the date, time, where it happened, what kind of problem it was, and a short description. When they click "submit," this info is saved briefly. The admin is then notified. If the admin doesn't approve it within a set time, the data is deleted automatically. This ensures that only approved reports remain in the system for further action.

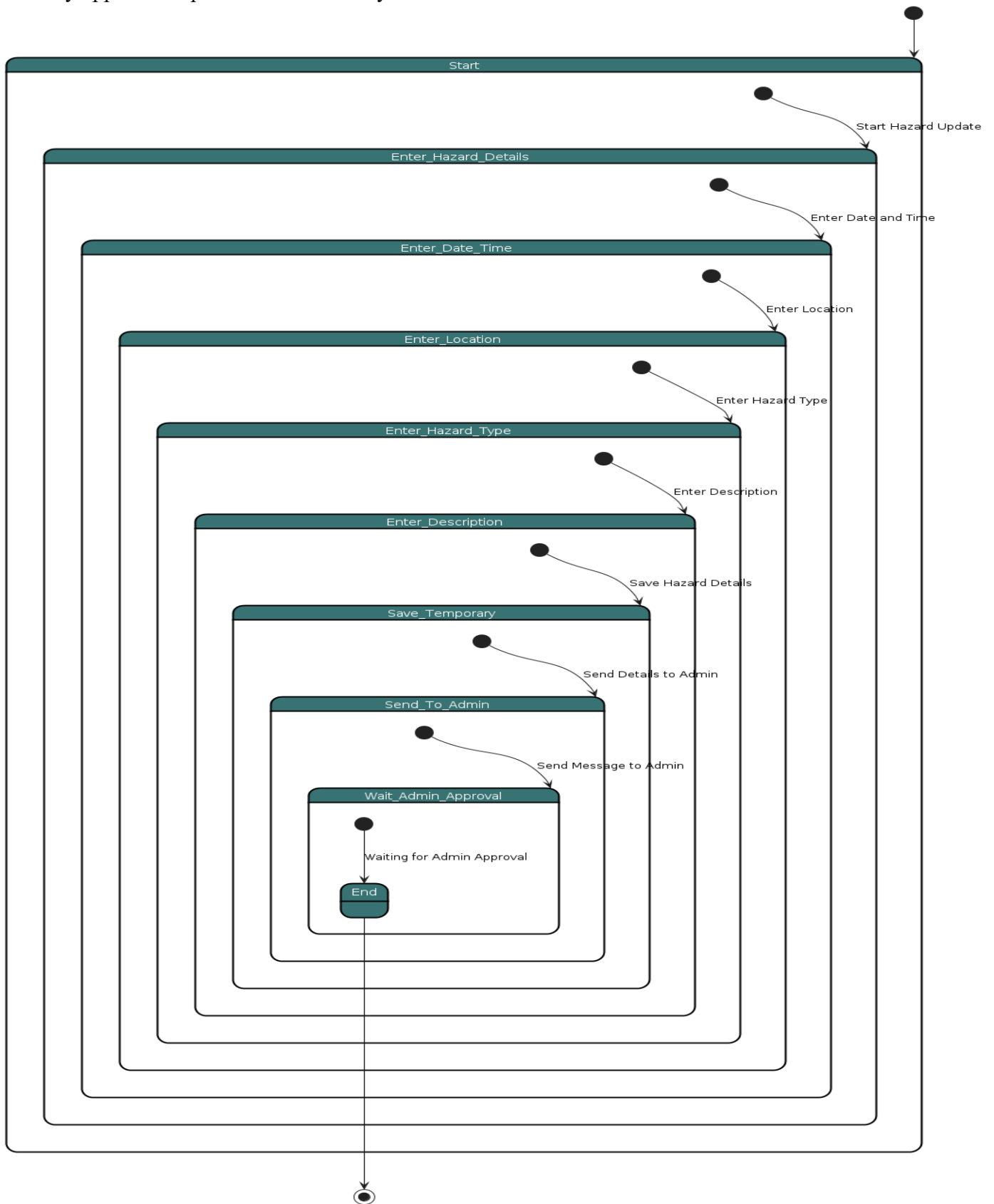


Figure 18 - State machine 7

## 8. State 8 Approve Hazard

After confirming the hazard details with the train driver via phone call, the admin approves the hazard report. They then navigate to the hazard location page, where they can search for specific locations to see what type of hazards have been reported there. This feature allows the admin to track and manage hazards efficiently based on their location.

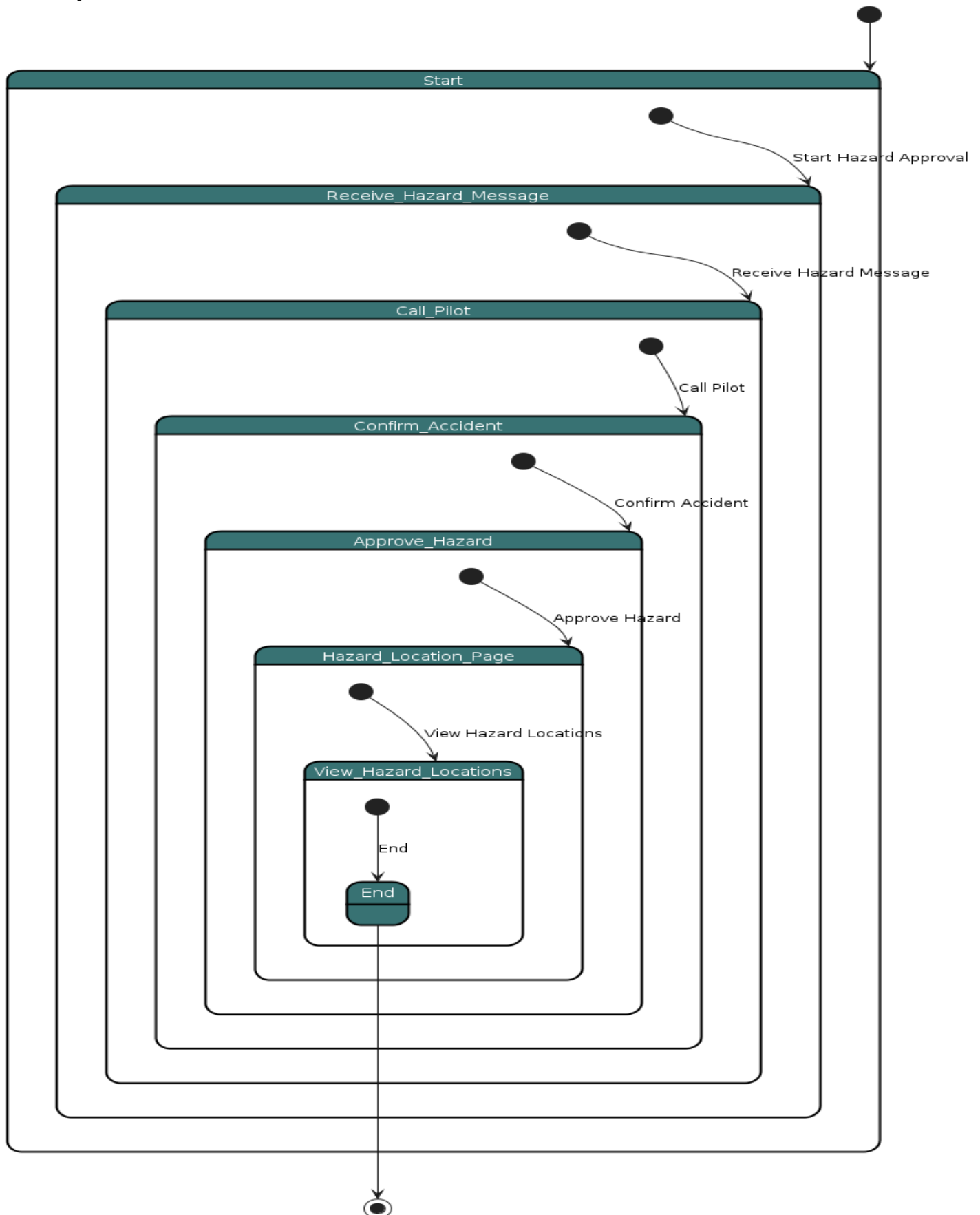


Figure 19 - State machine 8

## 2.5. Tools, techniques, libraries, and Implementation Environment

In railway safety, advanced tools and technologies play a crucial role. This section explores key solutions for object detection, weather updates, and hazard identification, vital for enhancing safety measures.

- **Computer Vision Libraries:** Use libraries such as OpenCV, TensorFlow Object Detection API, or YOLO (You Only Look Once) for object detection tasks. These libraries provide pre-trained models and tools for detecting objects like humans, vehicles, animals, and obstacles.
- **Deep Learning Frameworks:** Utilize deep learning frameworks like TensorFlow for developing custom object detection models tailored to railway safety scenarios.
- **Camera Systems:** Deploy high-resolution cameras with infrared capabilities along railway tracks to capture images for object detection purposes.
- **Weather APIs:** Integrate weather APIs such as OpenWeatherMap API to fetch real-time weather data including temperature, precipitation, wind speed, visibility, and weather forecasts for specific locations along the railway network.
- **Cloud-Based Weather Services:** Leverage cloud-based weather services that provide weather data in XML format, which can be easily processed and integrated into your system.
- **Geospatial Data Analysis:** Use geospatial data analysis tools like Geographic Information Systems (GIS) to analyze spatial data such as weather conditions, and environmental factors to identify potential hazards like elephant location, bull location, landslides, and other weather-related risks.
- **Cloud Computing Platforms:** Deploy your system on cloud platforms like Google Cloud Platform for scalability, reliability, and easy access to cloud-based services.
- **Web Development Frameworks:** Use web development frameworks such as Flask for building the front-end user interface and backend services of your railway safety system.
- **Database Management Systems:** Choose appropriate database systems like MongoDB or PostgreSQL for storing and managing data related to object detection results, weather updates, and hazard identification.
- **External APIs:** Integrates with third-party APIs for additional functionalities such as geolocation services, location services, meteorological information and authentication mechanisms.
- **Backend Services:** Node.js, a server-side JS runtime, can power backend services for processing data, running machine learning algorithms for object detection, analyzing weather data, and identifying potential hazards.

## 3. UI Design

This chapter aims to elaborate on how users use this web application.

### 3.1. Overview of UI Design

The user interface (UI) design for the railway object detection system focuses on delivering an intuitive and efficient platform for users to interact with the system seamlessly. It encompasses a centralized dashboard providing real-time insights into the system's status, including potential hazard locations and detected objects. Utilizing a map-based interface, users can select the route and identify potential hazards on that route. Real-time alerts and notifications promptly notify users of any anomalies or safety hazards. Additionally, the UI offers tools for historical data analysis, enabling locomotive pilots to enter hazard details and admin updates new hazard details for continuous improvement. Overall, the UI design prioritizes accessibility, responsiveness, and user feedback to enhance usability and optimize railway operations.

### 3.2. PACT (People, Activities, Contexts, Technologies) analysis

Creating user-centered applications to provide solutions is a challenging task. It requires understanding the problems faced by the users who will be using our web application. It is important to know about the activities they perform and the context in which these activities take place. By utilizing the PACT framework, we can create designs that are meaningful and effective. The PACT framework, which focuses on People, Activities,

Contexts, and Technologies, provides a valuable framework for understanding various aspects before diving into a specific design. It enables us to carefully consider the needs and preferences of the users, ensuring that the design approach we choose is well-suited to their requirements. By employing the PACT framework, we can develop a successful web application that aligns with user expectations and delivers a satisfying user experience.

User interface	People				Activity				Context				Technologies					Remark
	physical		Psychological	Usage	Overside Direction	Hazard Update And Management	Navigation and Menu Options	Authentication and User Management	Physical context		Social context	Organizational Context	Input	Output	Communication	Content	Data Store	
	Typography	Color Theme	Language	Experience					Administration Officers	Locomotive Pilot								
Register	Clearly Visible typography	Analogous Colors	English	No Need To consider	No	No	Yes	Yes	Department premises	Only Train Engines	Yes	User Support	Touch Screen Or Mouse	Screen	API Needed	UI Tool	Mongo DB	The interface is simple and easy to use. The fields are clearly labeled. The required fields are indicated. The interface should only be accessible to authorized users. The system should validate the user's input before creating a new account. Emphasize typography and use readable fonts. Visually appealing design.
Login	Clearly Visible typography	Analogous Colors	English	No Need To consider	No	No	Yes	Yes	Department premises	Only Train Engines	Yes	User Support	Touch Screen Or Mouse	Screen	API Needed	UI Tool	Mongo DB	Login form should be so user friendly, simple and easy to use. The form fields together and use clear labels for each field. Visually appealing design.
Reset password	Clearly Visible typography	Analogous Colors	English	No Need To consider	No	No	Yes	Yes	Department premises	Only Train Engines	Yes	User Support	Touch Screen Or Mouse	Screen	API Needed	UI Tool	Mongo DB	The interface is clear and easy to understand. It uses labels that are easy for users to understand "Username" and "Reset Password". Once the user reset the password, he will receive an OTP to his provided phone number which should be entered in the provided field and press the "submit" button. If the locomotive pilot didn't receive the OTP to his phone number, he needs to click the "Resend code" link. Visually appealing design.
Select route	Clearly Visible typography	Analogous Colors	English	No Need To consider	No	Yes	Yes	Yes	Department premises	Only Train Engines	Yes	User Support	Touch Screen Or Mouse	Screen	API Needed	UI Tool	Mongo DB	The interface is showing a railway map to easily to select the route. The labels for the start and end points are clear. There is a large, prominent button labeled "Start" to initiate the journey. All the fields are created in dropdown system. Visually appealing design.
Home Page	Clearly Visible typography	Analogous Colors	English	No Need To consider	Yes	Yes	Yes	Yes	Department premises	Only Train Engines	Yes	User Support	Touch Screen Or Mouse	Screen	API Needed	UI Tool	Mongo DB	The interface is clear and easy to read, with large text and bright colors. It displays important information for users including the date, time, next location, route status (clear), and any objects/elephant or bull detected on the track. The visibility level is categorized as normal or low. The interface is clear and concise, with large buttons for easy selection. It displays a clear image of the animal the user is reporting (in this case, an elephant). Visually appealing design.
Update Hazard	Clearly Visible typography	Analogous Colors	English	No Need To consider	No	Yes	Yes	Yes	Department premises	Only Train Engines	Yes	User Support		Screen	API Needed	UI Tool	Mongo DB	The interface is simple and easy to use. The fields are clear labeled and easy to fill. The submit button is there to submit the typed details and clear button is to clear the typed details. Visually appealing design.
Hazard Locations	Clearly Visible typography	Analogous Colors	English	No Need To consider	No	Yes	Yes	Yes	Department premises	Only Train Engines	Yes	User Support	Touch Screen Or Mouse	Screen	API Needed	UI Tool	Mongo DB	The interface is simple and easy to use. The interface clearly displays the hazard location and what type of hazard are possible in that area. It has a search bar that allows users to search for specific hazards. There is a back button to navigate back to the previous screen. Visually appealing design.
Admin Home	Clearly Visible typography	Analogous Colors	English	No Need To consider	No	Yes	Yes	Yes	Department premises	Only Train Engines	Yes	User Support	Touch Screen Or Mouse	Screen	API Needed	UI Tool	Mongo DB	The interface is simple and easy to use. Admin has 3 options to approve hazard, check the hazard locations, and update the locomotive pilot details. They appear as buttons. Visually appealing design.
Admin Approve Hazard	Clearly Visible typography	Analogous Colors	English	No Need To consider	No	Yes	Yes	Yes	Department premises	Only Train Engines	Yes	User Support	Touch Screen Or Mouse	Screen	API Needed	UI Tool	Mongo DB	The interface is simple and easy to use and explore. The interface clearly displays the details of the hazard report, including location, type of hazard. It has clear buttons for approving or declining the hazard report. Visually appealing design.
Admin Approve Hazard Locations	Clearly Visible typography	Analogous Colors	English	No Need To consider	No	Yes	Yes	Yes	Department premises	Only Train Engines	Yes	User Support	Touch Screen Or Mouse	Screen	API Needed	UI Tool	Mongo DB	The interface is simple and easy to use. The interface is clear and easy to understand, with large fonts and icons. This interface will be shown the newly updated hazards as notification. The admin can approve it or decline it through clicking the appropriate button. Visually appealing design.
Admin Hazard Locations	Clearly Visible typography	Analogous Colors	English	No Need To consider	No	Yes	Yes	Yes	Department premises	Only Train Engines	Yes	User Support	Touch Screen Or Mouse	Screen	API Needed	UI Tool	Mongo DB	The interface is simple and easy to use. The interface allows users to easily report a new hazard by selecting a location from a dropdown menu and choosing the hazard type from a list. It uses clear and concise labels for the hazard location and type. The system should consider allowing admin to provide a text description of the hazard. Visually appealing design.
Admin Locomotive Pilot Details	Clearly Visible typography	Analogous Colors	English	No Need To consider	No	No	Yes	Yes	Department premises	Only Train Engines	Yes	User Support	Touch Screen Or Mouse	Screen	API Needed	UI Tool	Mongo DB	The interface is simple and easy to use. this interface provides some fields to update the locomotive pilot details except his ID. This can be updated by only administrative officer. Visually appealing design.

Figure 20 - PACT Table

PACT Table: [https://drive.google.com/file/d/19nBOMSSCMVoDccXHbc64r9mljou93v\\_I/view?usp=sharing](https://drive.google.com/file/d/19nBOMSSCMVoDccXHbc64r9mljou93v_I/view?usp=sharing)

## 3.2.1 People

The people who would be the users of our railway safety web application can be categorized into two types:

- **Locomotive pilot:** People who use the web application to detect obstacles and other potential hazards on railway tracks to ensure safe navigation during train operation.
- **Administration Officers:** Individuals who utilize the web application for monitoring, approving, and updating hazards.



**Physical Aspects:** When creating a theme and choosing typography for the web application, we make sure it fits well with the purpose and the people who will be using it. We consider things like visual impairments, such as poor eyesight. This helps us select colors, icons, and typography that are easy to see and understand. This way, the design looks good and is accessible to our users, even those with visual challenges.

**Psychological Aspects:** Once they see the English language, our users shouldn't become upset. However, since this language is used worldwide, we cannot ignore it. Thus, throughout our interfaces, we have used very basic English with a few short phrases in order to make our system user friendly.

**Usage Aspects:** Our system has developed to be easily understandable and usable without requiring training. Users with basic computer knowledge like inputting the details and clicking the buttons is enough in order to navigate and handle the system effectively.

## **3.2.2 Activity**

The Activities section focuses on the actions that users have to perform throughout our system in order to ensure railway safety throughout Sri Lanka.

### **3.2.2.1 Temporal Aspects**

The proposed web application for train drivers is crucial for ensuring timely detection and interpretation of objects on the railway track. Real-time processing of data is essential to provide immediate alerts and warnings to train drivers, enabling them to take preventive actions swiftly. Furthermore, historical data analysis can offer insights into past incidents, facilitating proactive measures to enhance railway safety. Continuous updates and maintenance of the system will be necessary to ensure its effectiveness and reliability over time.

### **3.2.2.2 Cooperation & Complexity**

Cooperation and complexity are inherent in the development and deployment of such a system. Collaboration with railway authorities, technology providers, and other stakeholders is essential for integrating the application with existing infrastructure and ensuring seamless operation. The system's complexity arises from the integration of various technologies, including computer vision, data processing, and communication systems. A modular design approach will be beneficial, allowing for scalability and flexibility as the system evolves to meet changing requirements.

### **3.2.2.3 Nature of the Content**

Regarding the nature of the content, the web application should prioritize visual presentation to provide train drivers with real-time photo feeds from cameras mounted on the train. A clear and intuitive representation of object detection results, weather information, and other relevant data will facilitate quick comprehension by users. Interactive features, such as alerts and notifications, will enhance user engagement and effectiveness, further contributing to railway safety.

### **3.2.2.4 Security critical**

Security considerations are paramount given the critical nature of railway safety. The system must adhere to stringent security standards to prevent unauthorized access, tampering, or manipulation of data. Encryption should be employed to secure communication channels, while access control mechanisms should restrict user permissions to authorized personnel only. Regular security audits and vulnerability assessments will help identify and address potential weaknesses, ensuring the system's robustness against emerging threats.

## 3.2.3 Contexts

### 3.2.3.1 Physical Context

The web application can be accessed from various physical environments, including train engines, offices, or even remotely from different locations. The design should consider flexibility to accommodate different physical contexts.

### 3.2.3.2 Social Context

The web application should be accessible from any location with internet access. It should consider privacy concerns and ensure that sensitive information is not exposed to unauthorized individuals.

## 3.2.4 Technology

- **Input:** When users need to enter information, they can use touch screens and mouse. These input devices enable users to interact with the application and input values effectively and conveniently.
- **Output:** The primary outputs of the application are displayed on the screen. The outputs can be in the form of text or images, and their resolution is adjusted based on the device's interface. This ensures that the displayed content is optimized for the specific screen size and resolution, providing a clear and visually appealing output for users.
- **Communication:** The web application needs to interact with external systems or services and consider integrating relevant APIs (Application Programming Interface).
- **Content:** The JS Framework is used as a core technology for building the web application, and the UI Tool is used to design the web application.
- **Data Store:** My SQL server is used to store and manage user data.

## 3.3. *UI Design Considerations and Approaches*

Based on our UI design principles and strategies, these encompass the guiding principles and methodologies employed during the creation of the user interface. These principles aim to ensure user-friendliness and comprehension of the interface. Among the core tenets of our UI design are user-centered design, consistency maintenance across the web application, deliberate page layout, strategic utilization of color and texture, typography for hierarchical structure and clarity, effective communication of system status, and consideration of default settings. Let's delve deeper into the analysis of these foundational principles.

We place the needs and preferences of the target users at the center of our design process. Understand their goals, behaviors, and expectations. It conducts user research and usability testing to gather insights and validate design decisions.

Ensuring consistency is key to creating a cohesive user experience within our application. It is important to employ common UI patterns and conventions that users are familiar with, as this helps them navigate effortlessly. By maintaining consistency in layout, typography, icons, and interactions, we make it easier for users to navigate through our application.

To provide a seamless user experience across different devices and screen sizes, we prioritize responsiveness in our web application. This involves designing and testing the interface to gracefully adapt to various resolutions, from desktop to mobile devices.

We have developed a clear and intuitive navigation system that enables users to navigate through the application effortlessly. To assist users in understanding the application's content and easily moving between sections, we utilize descriptive labels, logical groupings, and hierarchical structures.

Attention to typography is crucial for ensuring content readability and legibility. We carefully select fonts, font sizes, and line spacing that facilitate easy reading. Additionally, we maintain a good contrast between text and background to enhance readability. Employing headings, subheadings, and bullet points helps break up content and make it scannable.

To guide users, we have used clear visual infographics in our web application. This approach is effective because it combines words and visuals to present information clearly, quickly, and appealingly and emphasize key statistics, breaking up large blocks of text. It also captures the attention of the user and communicates visually, so that users can use our application seamlessly and in a more engaging way.

Accessibility is a priority in our design approach. We consider factors such as color contrast, alt text for images, keyboard navigation, and compatibility with assistive technologies.

To provide a fast and seamless user experience, we optimize the performance of our web application. This includes minimizing file sizes, utilizing caching techniques, and optimizing images to reduce load times. Our goal is to ensure quick response times for user interactions and transitions.

## **3.4. UI Design Assets**

### **3.4.1. Design Tools**

We used FIGMA as our design tool. It is a cloud-based design tool that allows designers to collaborate in real-time and versatile. It helps us to create visually appealing and interactive interfaces. For additional information regarding the UI design wireframes, color interfaces, and Style guide, refer to the following Figma link. -

### **3.4.2. Techniques**

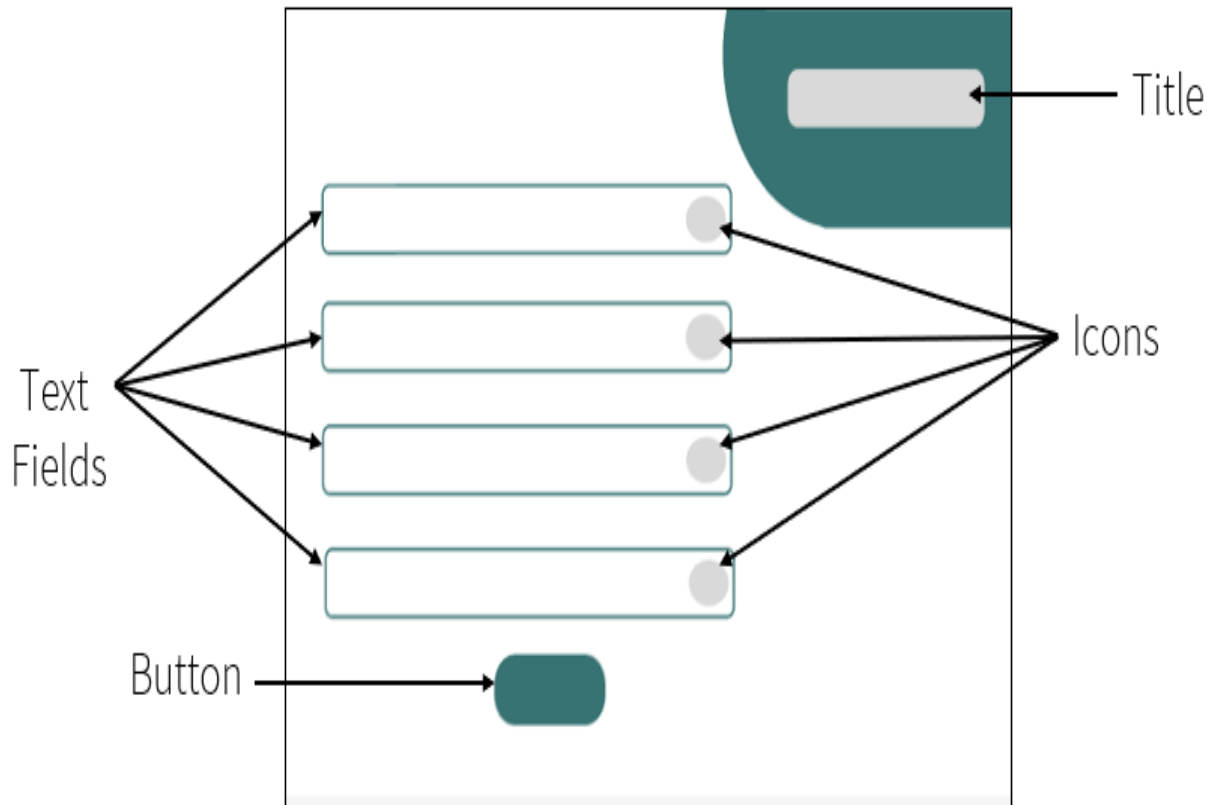
UI design techniques are essential methods and strategies employed by users to create user interfaces that are intuitive, user-friendly, and visually appealing. In our web application, we utilize a variety of these techniques to ensure an optimal user experience.

- We use visually appealing colors and animations that can help make our interface more engaging and dynamic.
- By using large-size fonts users can easily see what is in the display.
- Use a clear visual hierarchy to prioritize important information such as object detection alerts and weather updates. Ensure that essential elements are easily distinguishable and prominently displayed to catch the attention of train drivers.
- Utilize intuitive icons to represent different objects on the railway track, weather conditions, and other relevant data. Icons can help convey information quickly and efficiently, especially in situations where train drivers need to make rapid decisions.
- Incorporate maps or schematic representations of the railway network to provide train drivers with spatial context and situational awareness. Highlight the current location of the train, upcoming stops, and areas with detected objects or potential hazards on the track.

### 3.4.3. Layout Of the User Interfaces

#### Input Design Layouts

#### Register Layout



*Figure 21 - Register Page Layout*

## Login Layout

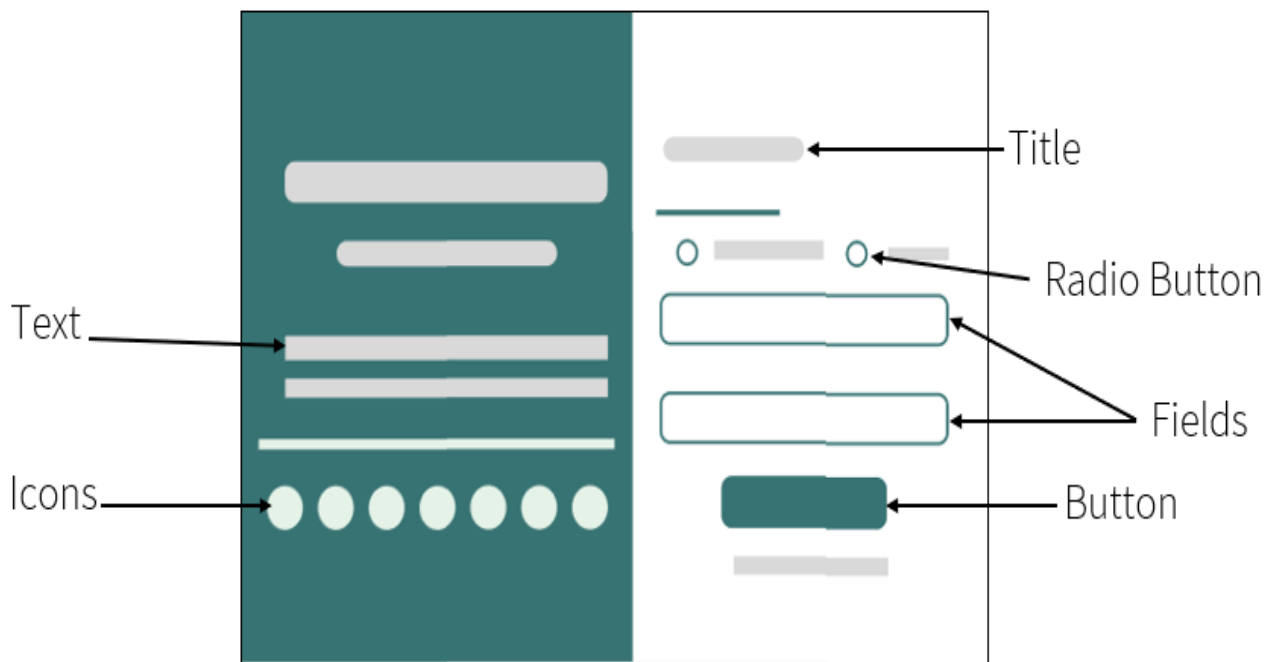


Figure 22 - Login Page Layout

## Reset password Layout

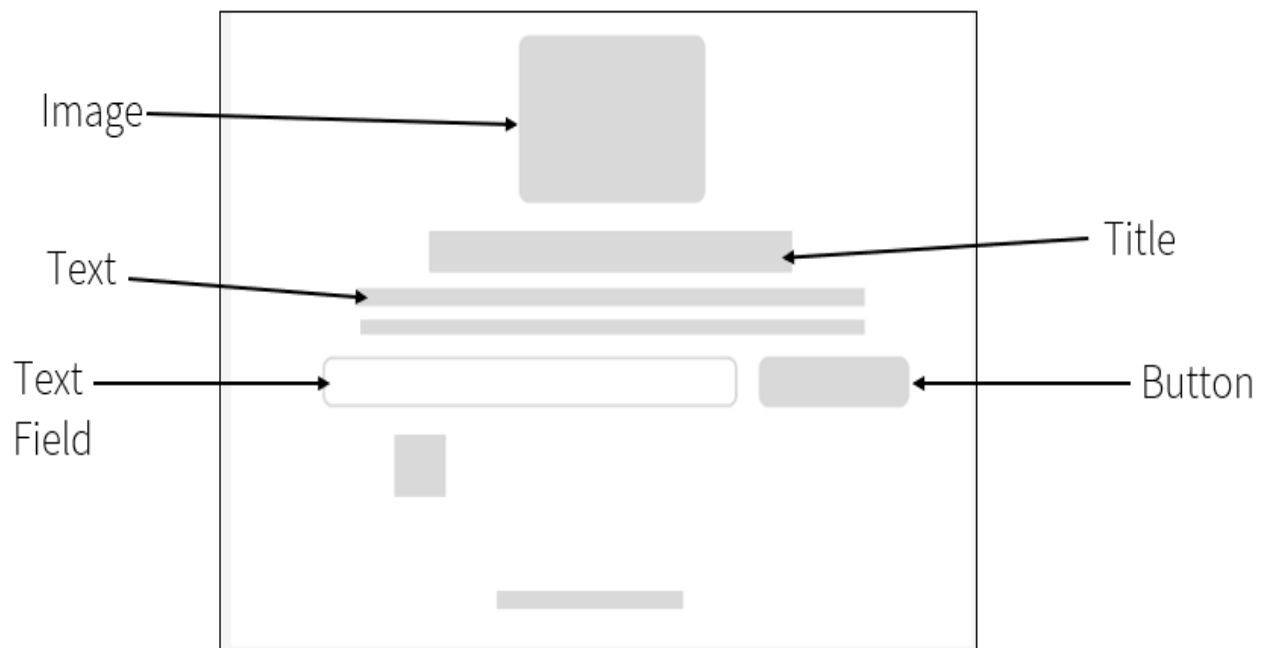


Figure 23 - Reset Password Layout

## Update Hazard Layout

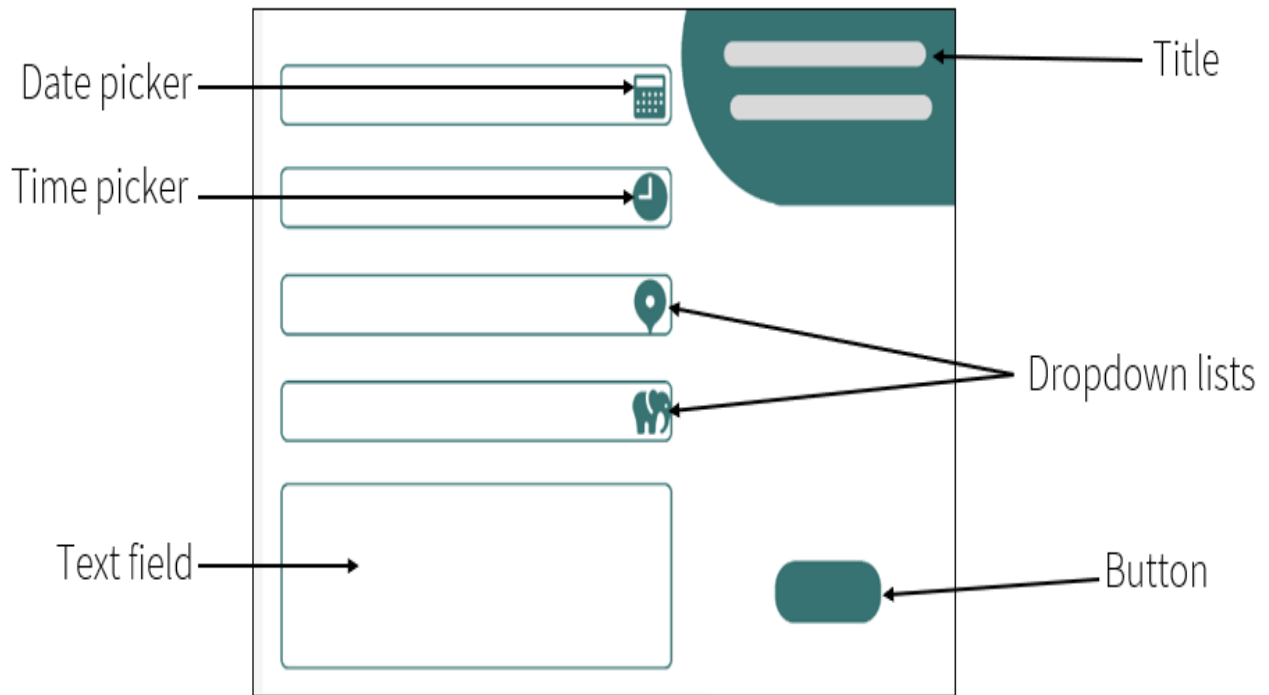


Figure 24 - Update Hazard Layout

## Admin Home Page

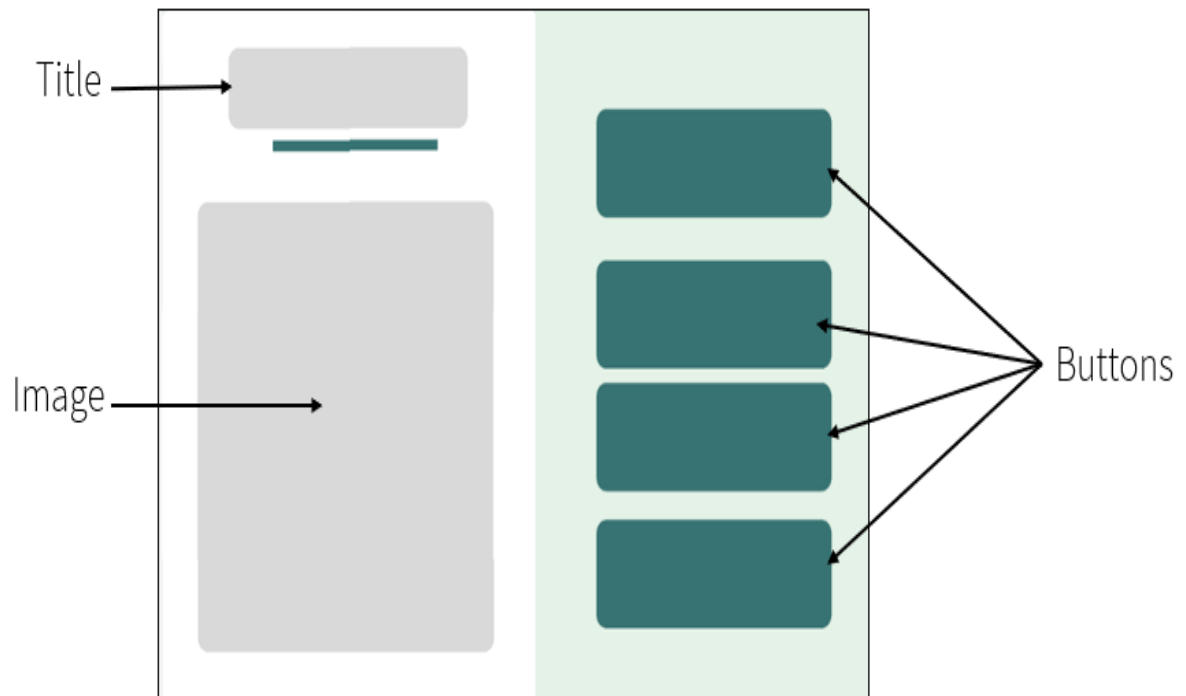
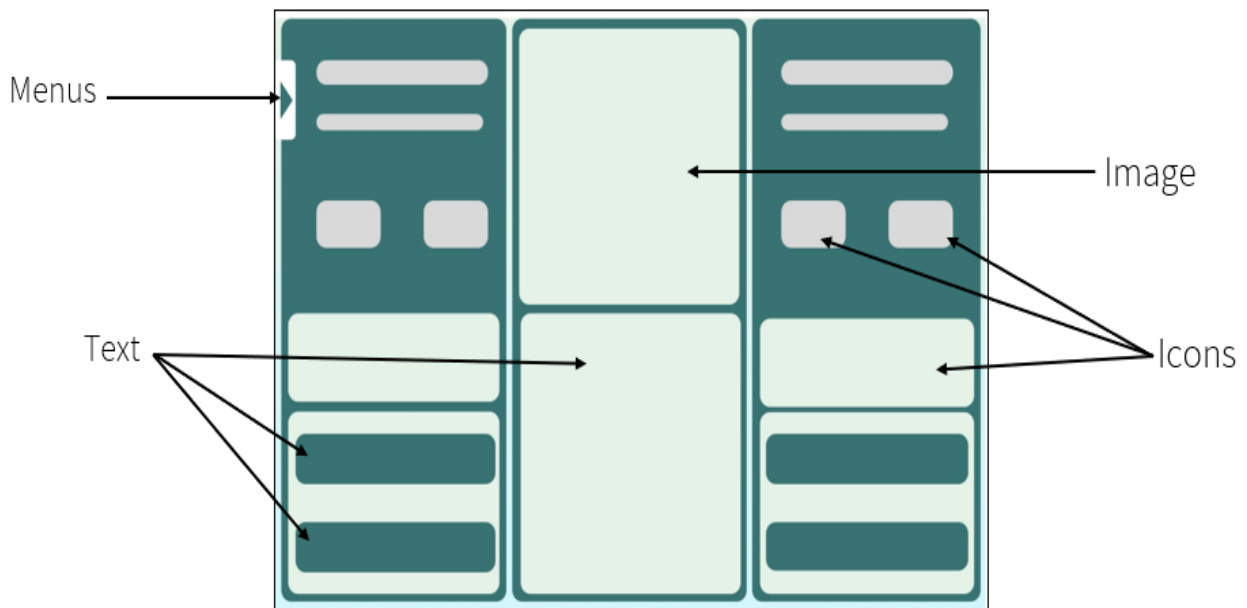


Figure 25 - Admin Home Page Layout

## Output Design Aspects

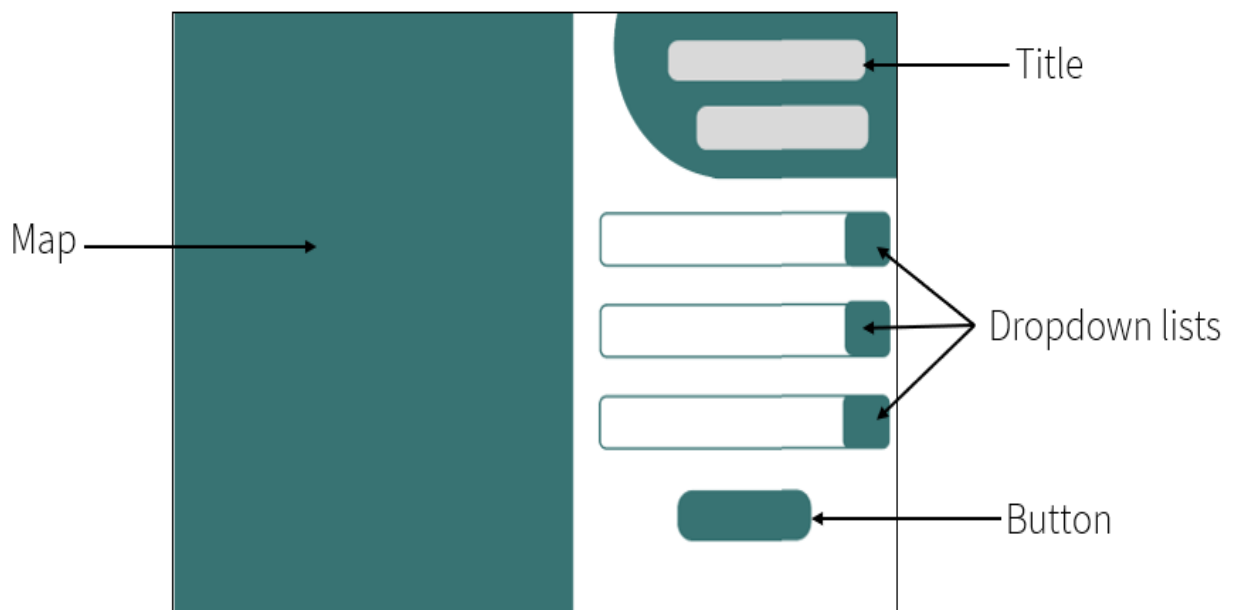
### Home Page Layout



*Figure 26 - Home Page Layout*

## Input and Output Design Aspects

### Select Route Layout



*Figure 27 - Select Route Layout*

## Hazard Locations Layout

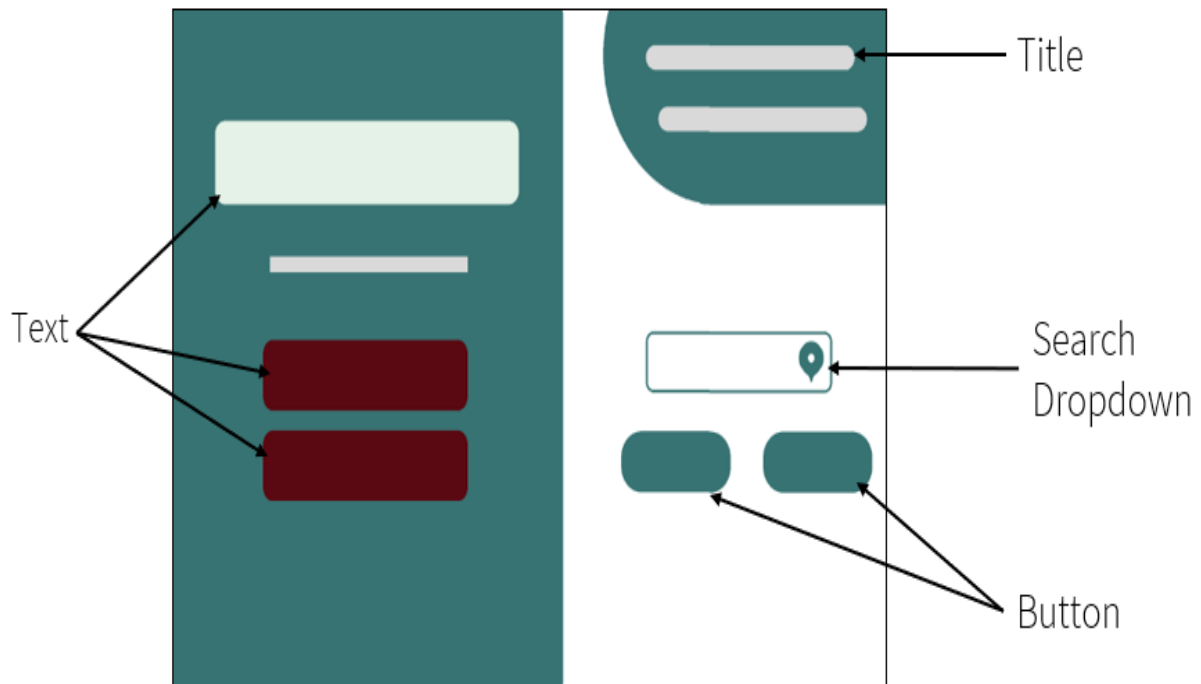


Figure 28 - Hazard Location Layout

## Admin Approve Hazard Layout 1

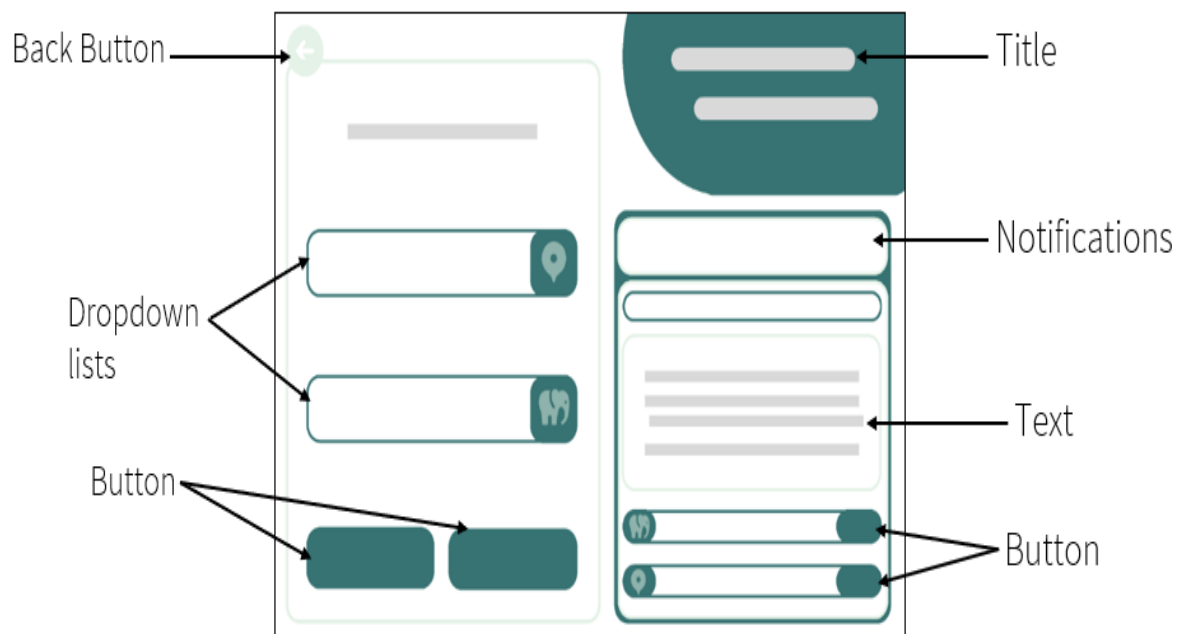


Figure 29 - Admin Approve Hazard Layout 1



## Admin Hazard Location Layout

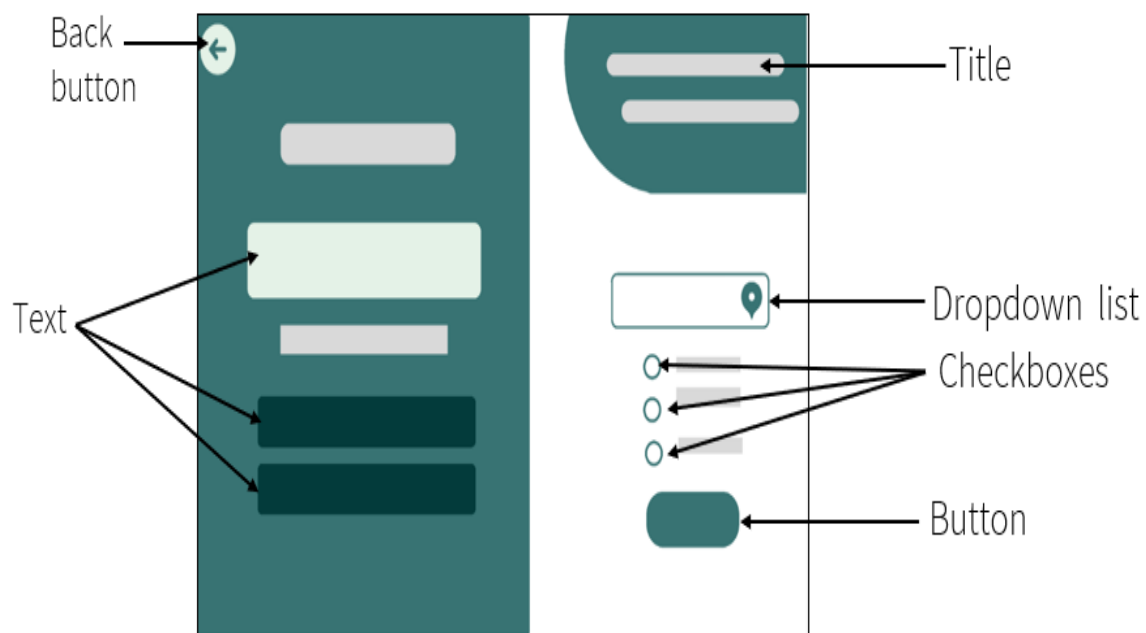


Figure 30 - Admin Hazard Location Update Layout

## Admin Approve Hazard Locations Layout 2

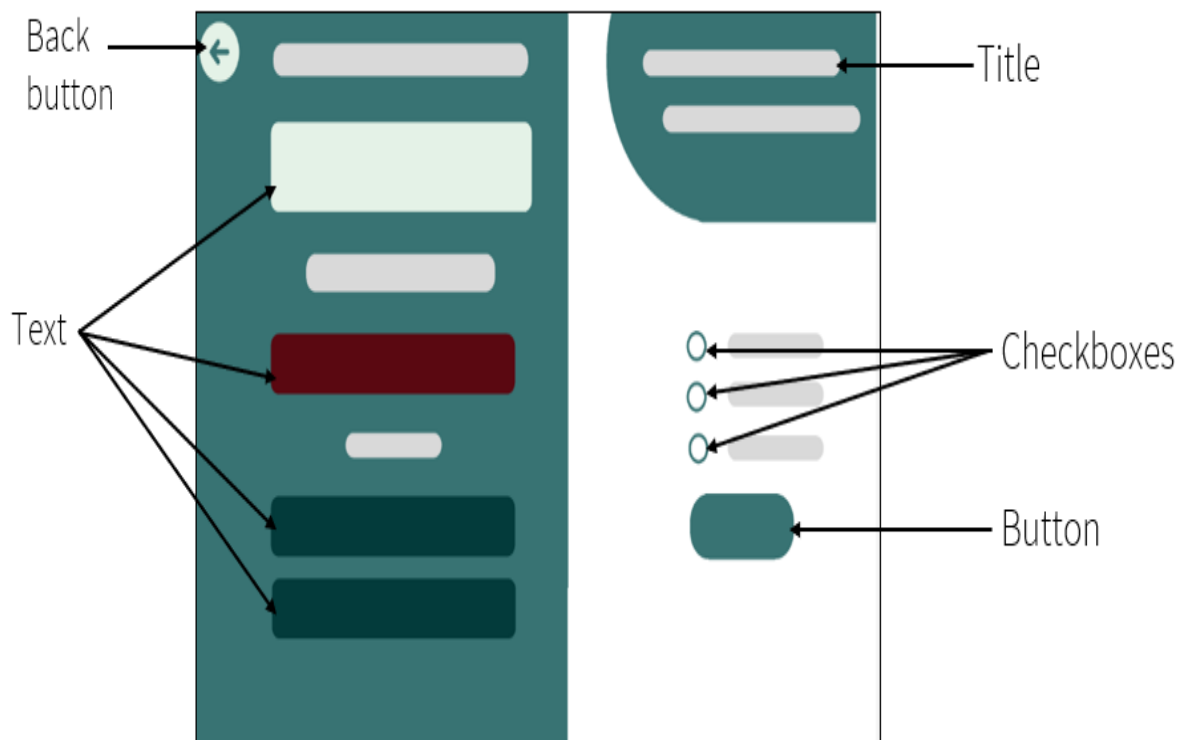


Figure 31 - Admin Approve Hazard Layout 2

## Admin locomotive pilot Details Layout

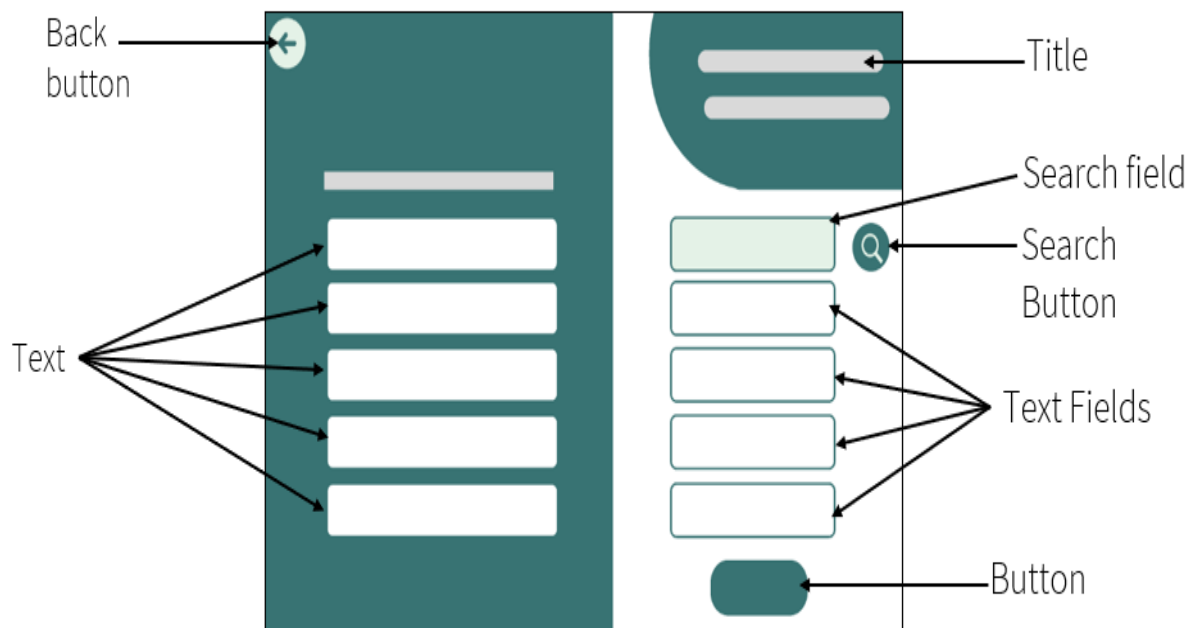


Figure 32 - Admin locomotive pilot Details Layout





### 3.4.4. User Interfaces

A diagram illustrating the Register page UI. The page features a dark teal header with the word 'Register' in white. Below the header, there are four input fields, each with a label and an icon: 'Locomotive id' with a person icon, 'Name' with a person icon, 'Email' with an envelope icon, and 'Phone number' with a phone icon. Below these fields is a dark teal 'Register' button. The background is white.

Figure 33 - Register Page

# Srilankan Railway Department

Welcome to the Sri Lankan Railway Department's Locomotive Pilot and Admin System! Streamlining railway operations for efficiency and safety across Sri Lanka.



## Login

☒ Locomotive Pilot ☐ Admin


Username

Password

Login

[Forgot password ?](#)

Figure 34 - Login Page



## Reset your password

The "Forgot Password" interface initiates account recovery by sending a verification code to the user's phone. Upon entering the correct code, users gain access to change their password securely.

Username

Reset Password

[Return to login ?](#)

Figure 35 - Reset Password

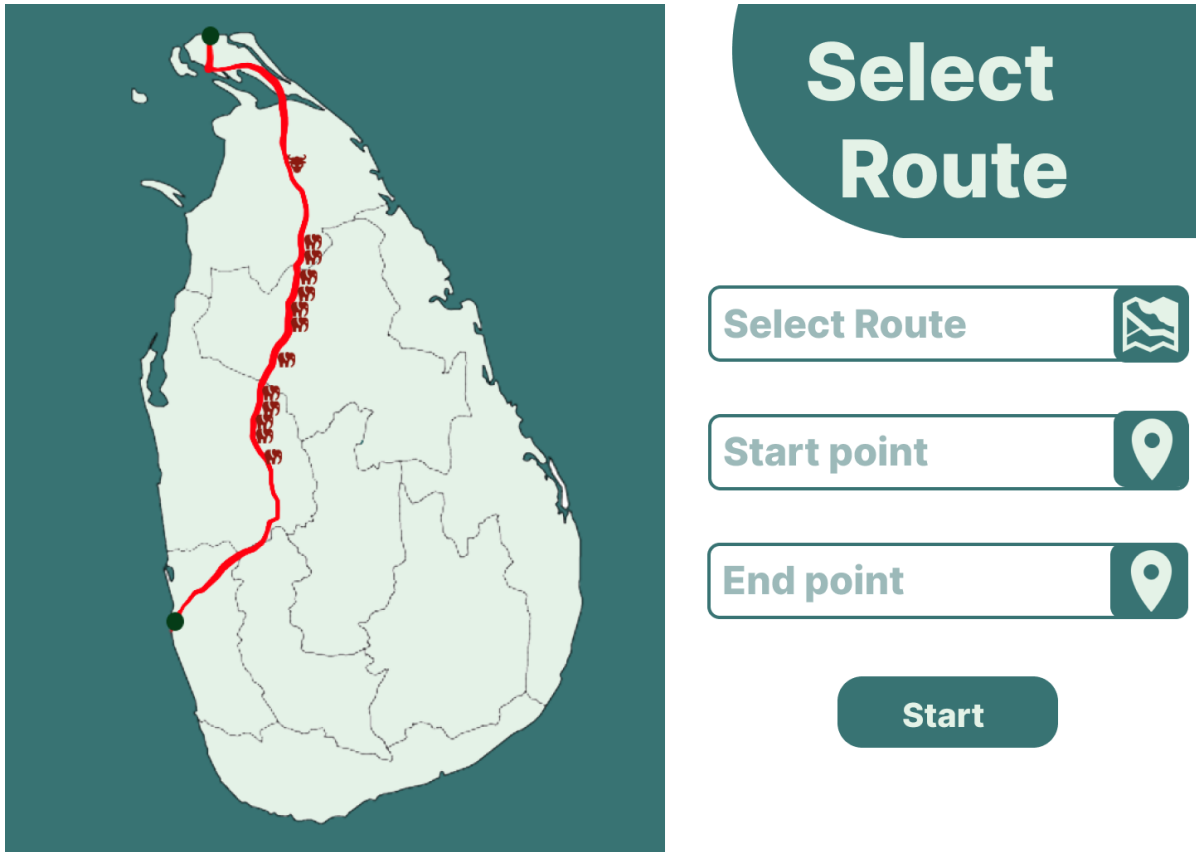


Figure 36 - Select Route

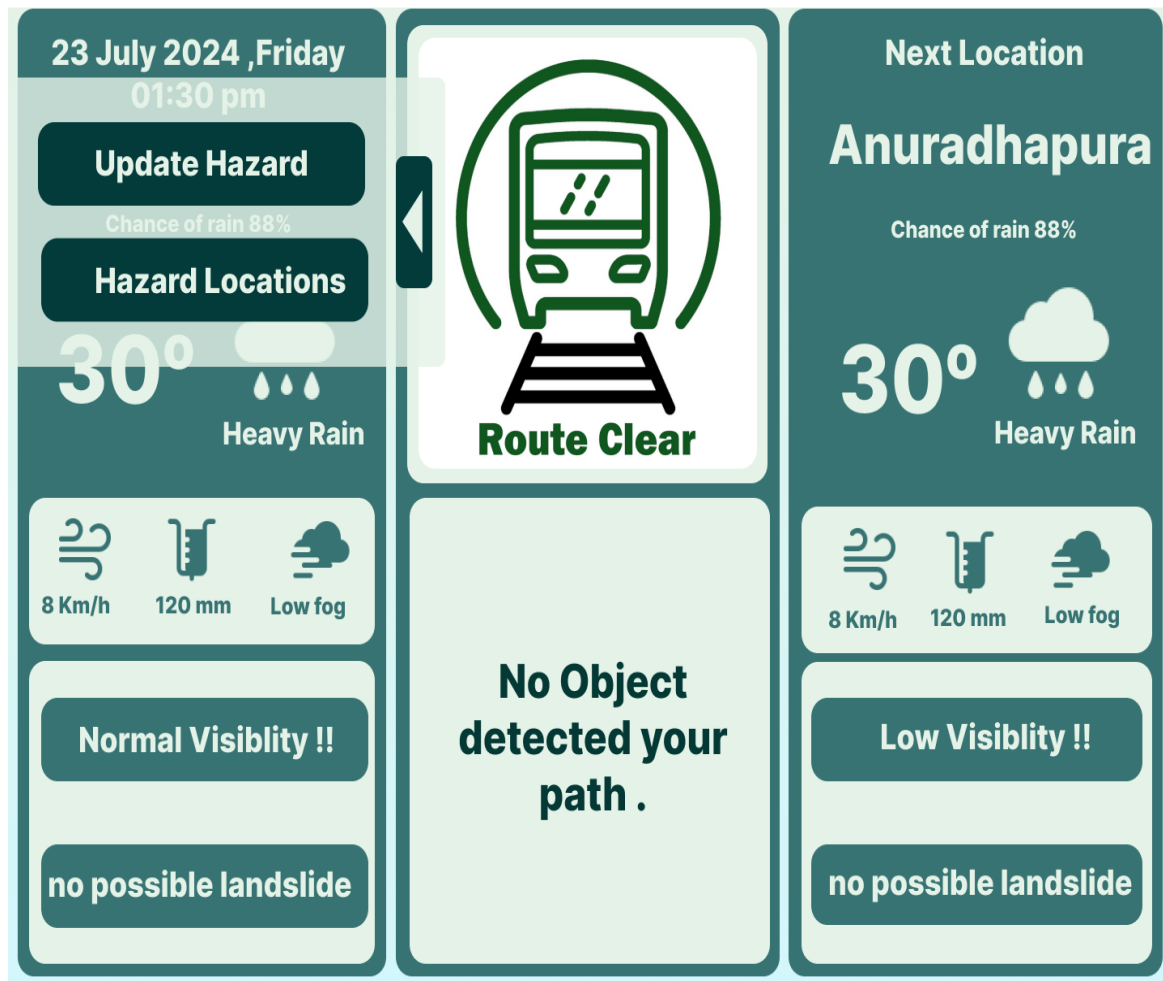


Figure 37 - Home Page 1

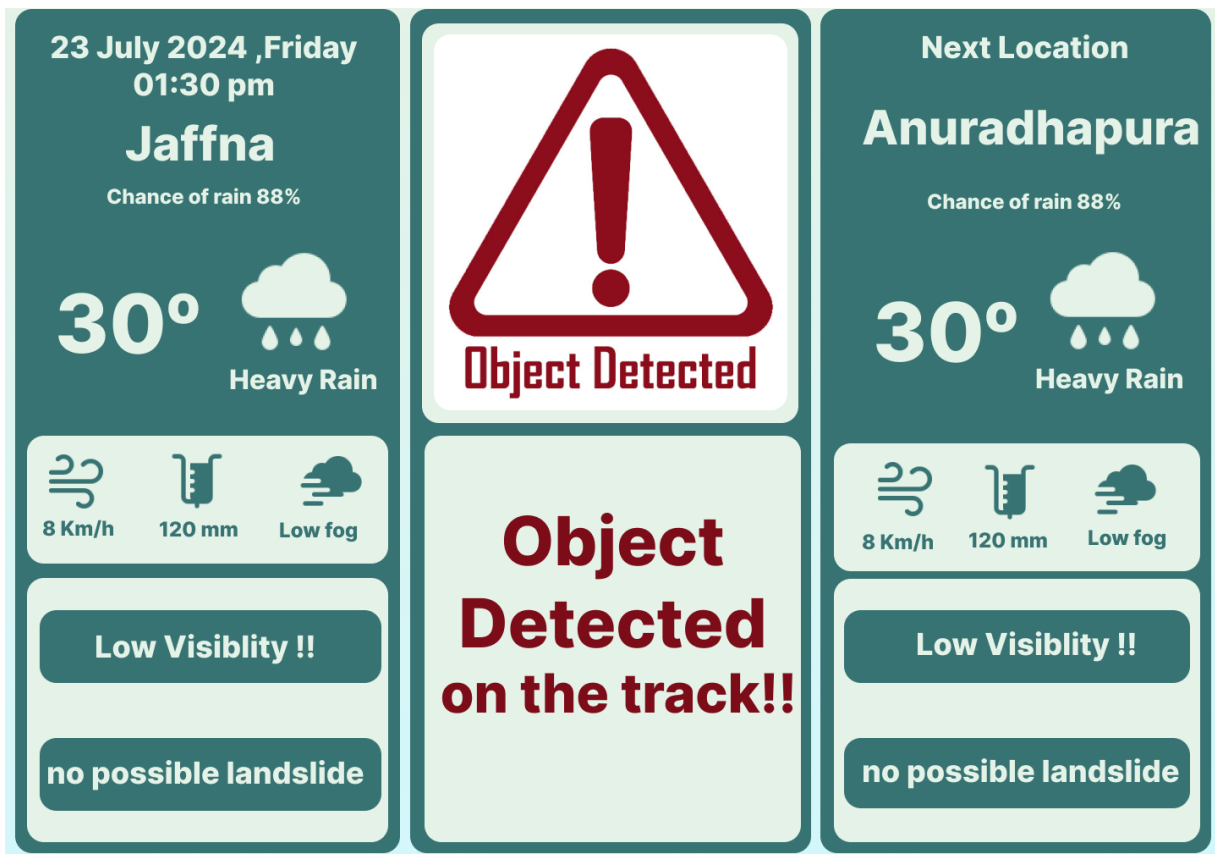


Figure 38 - Home page 2

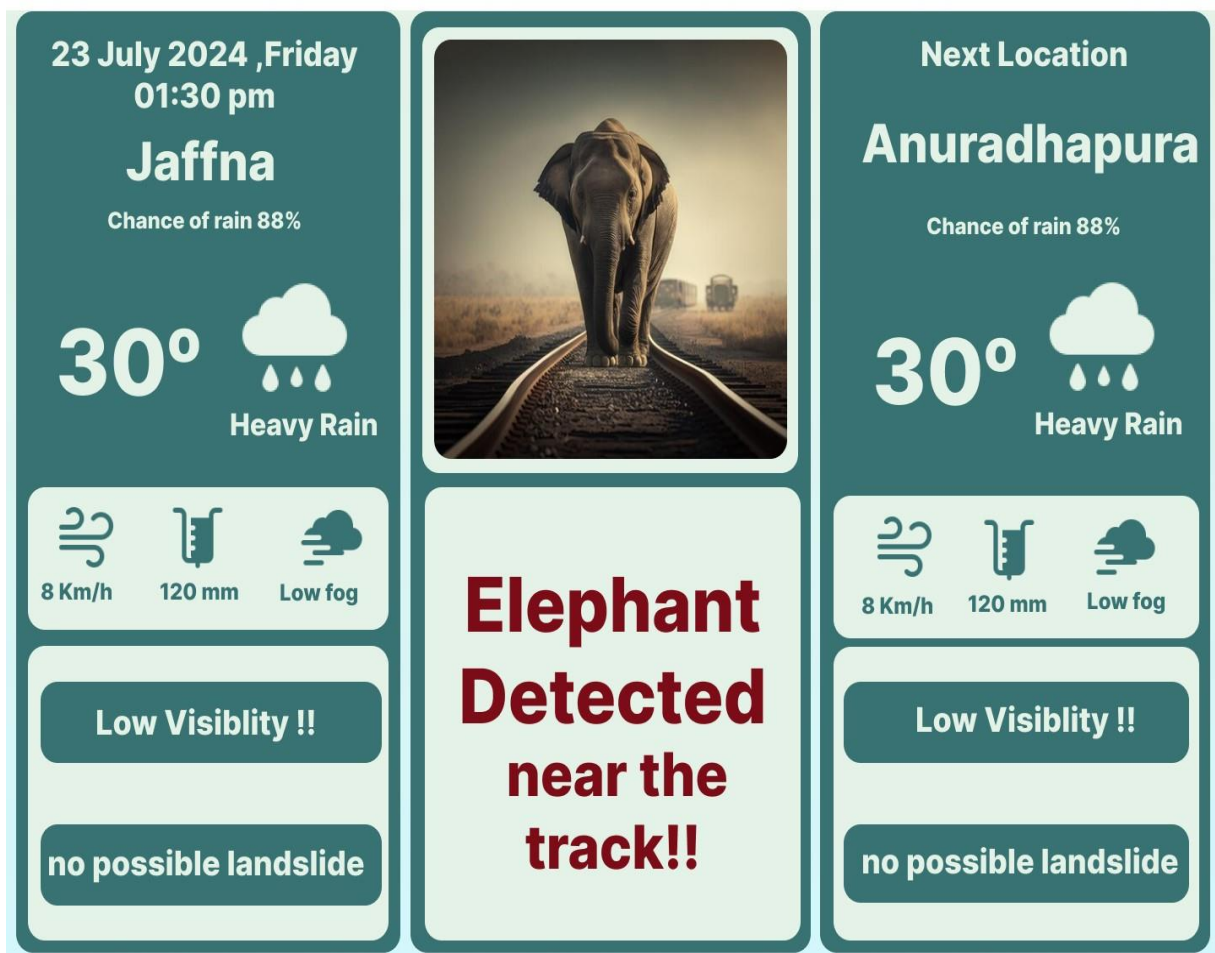


Figure 39 - Home page 3

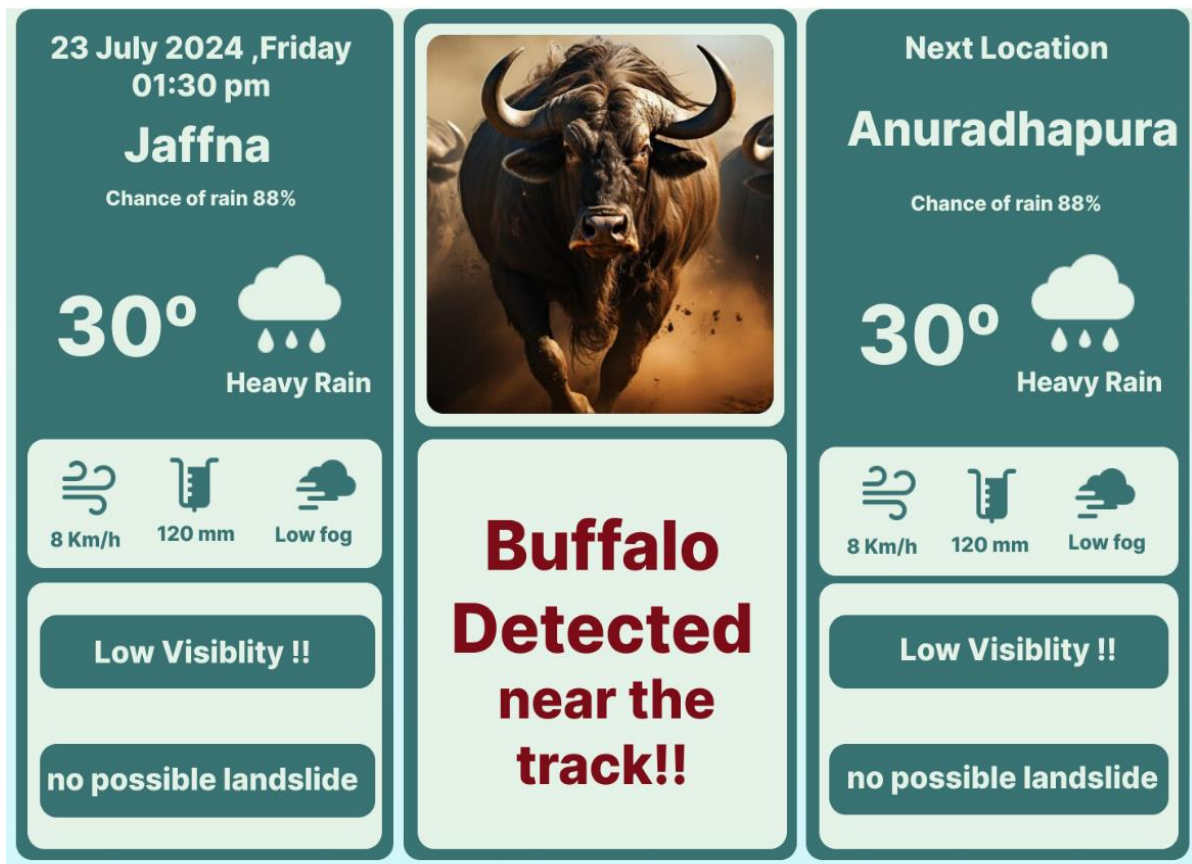


Figure 40 - Home page 4

Date

Time

Location

Hazard type

Description

Update Hazard

Submit

Clear

[illegible]

# Admin



An isometric illustration of a dashboard interface. It features a vertical sidebar on the left with icons for a menu, clock, search, list, and settings. The main area contains several floating panels: a large panel with three donut charts, a smaller panel with a line graph, and another with a sun icon. The design is clean and modern, using a color palette of teal, light green, and white.

- Approve Hazards
- Hazard Locations
- Locomotive Pilots  
Details

47

### Hazard Approval

Location

Hazard type

Approve

Decline

## Approve Hazard

New hazards update

Name : Kasun Perera Id:SLR1505

0774565867

**In Anuradhapura, on Wednesday, 11th February 2024, at 11:00 am, an elephant train collision was reported.**

Elephant Accident

Accept

Anuradhapura

Accept

Figure 44 - Admin Approve Hazard



Hazard Location

Anuradapura

New Hazard

Elephant

Possible Hazards

Elephant

Bull

Figure 45 - Update Hazard Location

# Hazard Locations

- ☒ Elephant
- ☒ Bull
- ☐ Land Slide

Update

Hazard Location

Kekirawa

Possible Hazards

Elephant

Bull

Figure 46 - Admin Hazard Location Update

# Hazard Locations

Location



- ☒ Elephant
- ☒ Bull
- ☐ Land Slide

Update

The image displays two mobile application screens for updating pilot details. The left screen, titled 'Locomotive Pilot', features a back arrow in the top left and five text input fields containing the following data: LP ID (SLR123456), Name (K Perera), NIC (832738907V), Phone (0771234567), and Email (pererak@gmail.com). The right screen, titled 'Locomotive Pilot Details', shows the same fields as input boxes. A magnifying glass icon is positioned to the right of the LP ID field. Below the input fields is a green 'Update' button.

Figure 47 - Update Pilot Details

### 3.5. Design Aspects

#### 3.5.1. Input Design Aspects

Input aspects in a web application are prominent for facilitating data entry. UI design techniques help in designing intuitive and user-friendly input fields. These fields can include text inputs, dropdown menus, radio buttons, buttons, back buttons.

The below user interfaces are used as input interfaces in our web application.

- **Register Interface:** This is the interface that the locomotive pilots use to register themselves. The locomotive pilot will register themselves by filling in the details of the locomotive pilot's ID, name, email address, and phone number. Finally, by clicking the 'register' button, that locomotive pilot will get registered to the system.
- **Login Interface:** we have 2 types of users. They are administrative officers and locomotive pilots. This login interface is available to them if they have already registered with the system. If an administrative officer logged into the system, then he will be redirected to the admin homepage. If he is a locomotive pilot, then will navigate to the UI for selecting a route. Both need to provide their password and ID details to log in.
- **Reset password Interfaces:** Locomotive pilots and administrative officers can reset their password using this interface. There is a link labeled "forgot password" that the locomotive pilot must click to access this interface.

Three login interfaces are available in our system.

- ◆ **Reset password Interface** - This interface is developed to reset the password. In order to do that, the locomotive pilot needs to give his locomotive pilot ID.
- ◆ **Reset password Interface** - In this interface, the locomotive pilot needs to enter his locomotive pilot ID again to receive the OPT to his phone. After that, the received OTP should be entered in the provided field and press the "submit" button. If the locomotive pilot didn't receive the OTP to his phone number, he needs to click the 'Resend code' link.
- ◆ **Reset password Interface** - This interface is developed to create the new password. So the locomotive pilot's ID will be showcased in the interface. All he needs to do is just enter the password and confirm it in the 2nd space and press the 'submit' button.

Each of these three interfaces has a return login link that takes users to the login page.

- **Update Hazard Interface:** This is the interface where the hazard is updated. The locomotive pilot will enter the hazard into the system as soon as it occurs. To update the hazard to the system, locomotive pilots must fill in a few boxes on this interface. They are the date figure, time figure, location, type of hazard, and description of the hazard. There will be dropdown options for the first four entry details. However, the description needs to be typed. If the locomotive pilot needs to clear the description, all he needs to do is just press the back button.
- **Admin homepage Interface:** This is the administrative officer's homepage. There are four buttons to move on. They are hazard notification, approve hazard, hazard location and locomotive pilot details. Once he clicks the one of those buttons, he will be redirected to the corresponding interface.

### 3.5.2. Output Design Aspects

**Home page Interface** - Our system has five different home page interfaces. In order to anticipate landslides, all interfaces include information about the date, time, day, and location—both current and next—as well as weather-related parameters. Other than that,

- **Home page 1 Interface** - This is our system's default home page. This interface will be displayed if no objects are found. There is a button in the upper left corner of it. The above interfaces will appear if any objects, bull, or elephant is discovered.
- **Home page 2 Interface** - This is used to identify objects. This will be shown if an object on the railway track.
- **Home page 3 Interface** - Elephant detection is done via this interface. This interface will be used to notify the locomotive pilot if an elephant is sighted.
- **Home page 4 Interface** - This interface is identical to the second. But the purpose of this interface is to identify bull and alert locomotive pilots.

### 3.5.3. Input and Output Design Aspects

1. **Admin approve Hazard Locations Interface:** This is the interface where administrative officers approve the newly updated hazards by the locomotive pilot. Administrative officers can view the hazard and update it through this interface.
2. **Admin locomotive pilot Details Interface:** this interface is used by administrative officers in order to make changes in the details of the locomotive pilot. Once the administrative office enters the locomotive pilot's ID, He can alter the locomotive pilot's details and view them on the interface's left side.
3. **Select Route Interface:** This is the interface that highlights the whole Sri Lanka railway network. There are three dropdown options visible on this interface. They are the route, starting and ending

points of the journey, respectively. The locomotive pilot can press the "start" button to begin the journey after selecting these options. Then the chosen route will be displayed highlighted throughout.

4. **Admin Approve Hazard Interface:** This interface will be shown the newly updated hazards as notification. Also, it will showcase the details of the locomotive pilot who reported it. And the details of hazard will be displayed as a description. If the Administrative officer needs to approve it and add to the system, all he needs to do is accept it. Once the administrative officer does that, the location and the type of hazard will be shown in the left side of the interface, then he just needs to approve it or decline it through clicking the appropriate button.
5. **Admin Hazard Locations Interface:** The administrative officer will update the hazard using this interface. If the locomotive pilot failed to notify a hazard, this will be applied. However, the administrator is aware of that.

#### 3.5.4. Dialogue Design Aspects

Dialogue design plays a crucial role in ensuring effective communication between the web application and train drivers, facilitating quick decision-making, and enhancing overall safety. One key aspect of dialogue design is the clarity and conciseness of alerts and notifications. Alerts should be formulated in a way that conveys critical information succinctly and unambiguously, using language that is easily understood by train drivers even in high-pressure situations. For example, alerts regarding detected objects on the track should clearly indicate the type of object, its location, and any relevant actions that need to be taken.

Another important aspect of dialogue design is providing contextual information to train drivers to help them understand the significance of alerts and make informed decisions. This includes not only information about detected objects but also relevant environmental factors such as weather conditions, visibility, and track conditions. By providing contextually rich dialogue, the web application can help train drivers assess the situation accurately and take appropriate actions to ensure safety.

In addition to clarity and context, dialogue design should also consider the emotional impact of alerts and notifications on train drivers. Alerts related to potential safety hazards should be formulated in a way that conveys a sense of urgency without causing undue panic or stress. Similarly, positive feedback and reinforcement can be incorporated into the dialogue design to acknowledge safe behavior and encourage continued vigilance. By striking the right balance between urgency and reassurance, the

web application can effectively engage train drivers and promote a proactive approach to safety.

Furthermore, dialogue design should be adaptable to the varying needs and preferences of individual train drivers. Providing customization options for alert settings, language preferences, and notification preferences allows train drivers to tailor the dialogue to their specific requirements and preferences. This flexibility not only enhances user satisfaction but also improves the effectiveness of the dialogue by ensuring that it resonates with the target audience. Through thoughtful dialogue design, the web application can establish a clear and effective channel of communication with train drivers, contributing to safer and more reliable railway transportation systems.

### 3.6. Hosting environment

We host our web application on cloud environment and using MongoDB to create a dynamic and database-driven environment for development and testing.

## 4. Data management

This chapter is about data management of the railway safety system. The data requirements, design tools and techniques, conceptual, logical, and physical database design, as well as schema refinement and security design, are all covered in detail in this chapter.

## 4.1. Data Requirement

Data requirements are the foundation for database design which provides the basis for structuring and organization of the database. By identifying the specific types of data needed, we establish the framework for storing and accessing relevant information efficiently.

Reliable data is crucial for computer vision and cloud-based solutions to reduce railway accidents. By defining data needs, we ensure effective collection, storage, and management.

In essence, identified data requirements pave the way for designing, implementing, and optimizing railway safety systems, thereby reducing accidents, and enhancing overall safety.

## 4.2. Design Tools and Techniques

### Mongo DB

MongoDB is the primary tool for database design and modeling during the project's design phase. It offers a user-friendly interface for visualizing, designing, and modifying our document-oriented schema.

Unlike relational databases, MongoDB utilizes a flexible schema that can evolve as our needs change. This flexibility is perfect for our railway safety system with its ever-growing data type like weather information. Also, it provides visual tools for schema design, allowing us to easily define document structures and relationships.

An additional benefit is the intuitive query builder. It allows building and executing complex queries visually, simplifying data retrieval.

Furthermore, MongoDB serves as a database administration tool, enabling us to manage users, and security configurations. We can leverage this to define access control policies and ensure only authorized users can access specific data within the database.

## 4.4. Conceptual Database Design

A conceptual database is a foundation for organizing data in a database. This describes how data is organized within the database and the structure of the database.

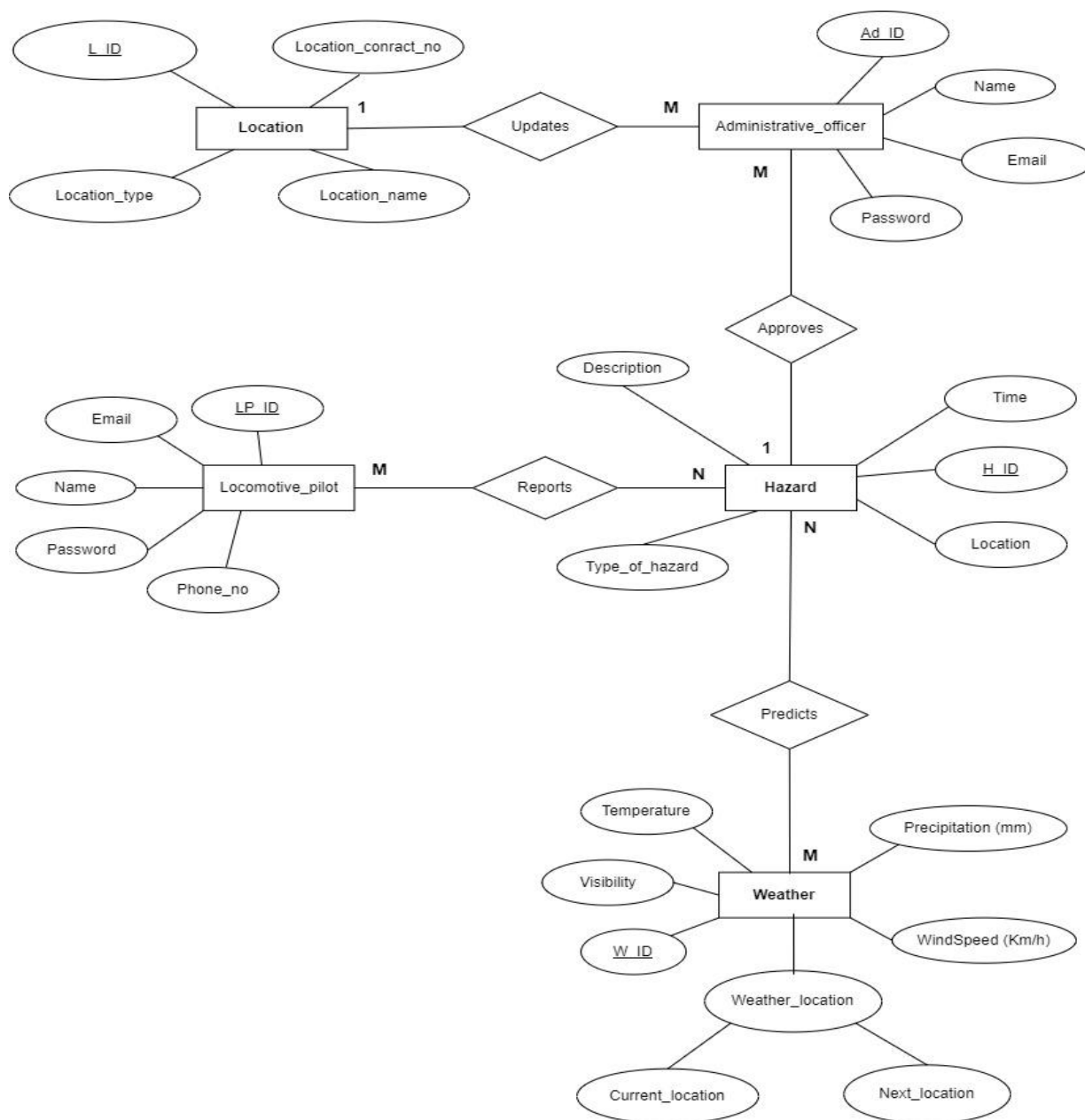


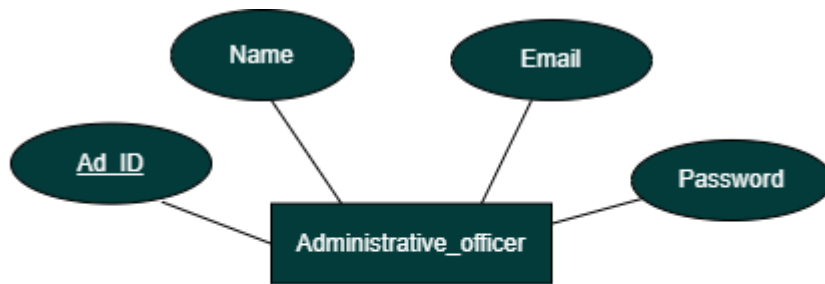
Figure 48 - EER Diagram

## 4.5. Logical Database Design

Logical database translates a conceptual database into a well-organized structure. We will assign a primary key and a foreign key for each table to connect them.

### 4.5.1 Map Regular Events

#### ADMINISTRATIVE\_OFFICER entity

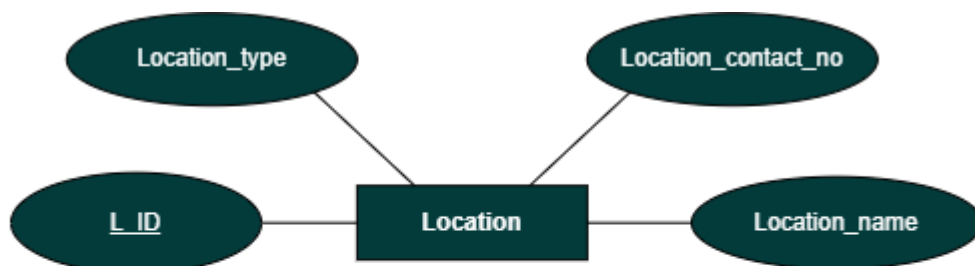


#### ADMINISTRATIVE\_OFFICER

<u>AD_ID</u>	NAME	EMAIL	PASSWORD
--------------	------	-------	----------

The primary key of the ADMINISTRATIVE\_OFFICER is AD\_ID.

#### LOCATION entity

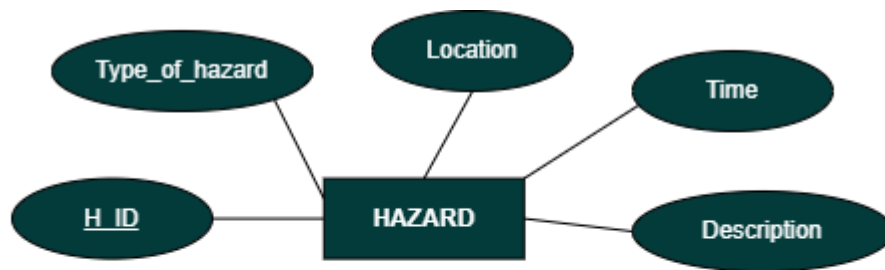


#### LOCATION

<u>L_ID</u>	LOCATION_TYPE	LOCATION_CONTACT_NO	LOCATION_NAME
-------------	---------------	---------------------	---------------

The primary key of the LOCATION is L\_ID.

### HAZARD entity

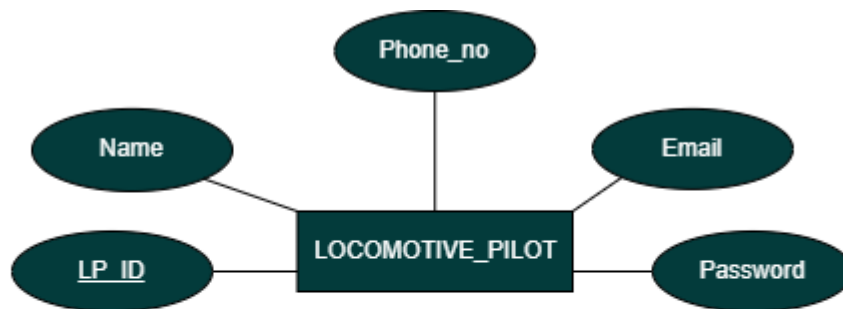


#### HAZARD

<u>H_ID</u>	TYPE_OF_HAZARD	LOCATION	TIME	DESCRIPTION
-------------	----------------	----------	------	-------------

The primary key of the HAZARD is H\_ID.

### LOCOMOTIVE PILOT entity



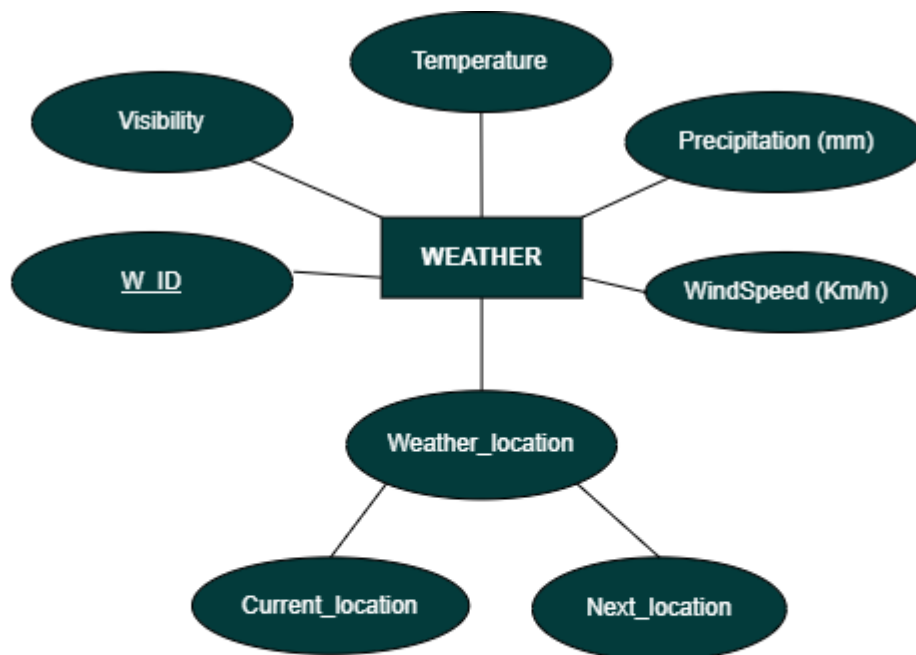
#### LOCOMOTIVE\_PILOT

<u>LP_ID</u>	NAME	PHONE_NO	EMAIL	PASSWORD
--------------	------	----------	-------	----------

The primary key of the LOCOMOTIVE\_PILOT is LP\_ID.



### WEATHER entity



#### WEATHER

<u>W_ID</u>	CURRENT_LOCATION	NEXT_LOCATION	VISIBILITY
TEMPERATURE	PRECIPITATION(mm)	WINDSPEED (km/h)	

The primary key of the WEATHER is W\_ID.  
The WEATHER\_LOCATION is a composite key.

### 4.5.2. Map Binary Relationship

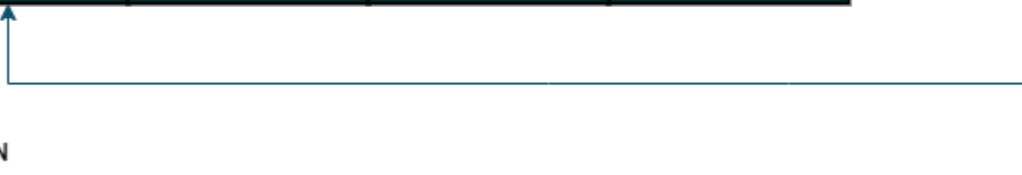
ADMINISTRATIVE\_OFFICER updates LOCATION (1 to M)

#### ADMINISTRATIVE\_OFFICER

<u>AD_ID</u>	NAME	EMAIL	PASSWORD
--------------	------	-------	----------

#### LOCATION

<u>L_ID</u>	LOCATION_NAME	LOCATION_TYPE	LOCATION_CONTACT_NO	AD_ID
-------------	---------------	---------------	---------------------	-------



ADMINISTRATIVE\_OFFICER approves HAZARD (1 to M)

ADMINISTRATIVE\_OFFICER

<u>AD_ID</u>	NAME	EMAIL	PASSWORD
--------------	------	-------	----------

HAZARD

<u>H_ID</u>	TYPE_OF_HAZARD	LOCATION	TIME	AD_ID
DESCRIPTION				

WEATHER predict HAZARD (M to M)

WEATHER

CURRENT_LOCATION	NEXT_LOCATION	TEMPERATURE	VISIBILITY	PRECIPITATION(mm)
WINDSPEED(km/h)	<u>W_ID</u>			

HAZARD

<u>H_ID</u>	TYPE_OF_HAZARD	LOCATION	TIME	W_ID
DESCRIPTION				

**LOCOMOTIVE\_PILOT report HAZARD (M to M)**

LOCOMOTIVE\_PILOT

<u>LP_ID</u>	NAME	PHONE_NO	EMAIL	PASSWORD
--------------	------	----------	-------	----------

HAZARD

<u>H_ID</u>	TYPE_OF_HAZARD	LOCATION	TIME	LP_ID
DESCRIPTION				



## 4.6. Schema refinement

As can be seen that, all the tables are in the 3rd normal form. Hence no need to polish them again. According to our railway safety system, the schema refinement below,



Figure 49 - Schema refinement

## 4.7. Physical Database Design

Physical database design represents the materialization of a database into an actual system. While logical design can be performed independently of the eventual database platform, many physical database attributes depend on the specifics and semantics of the target DBMS.

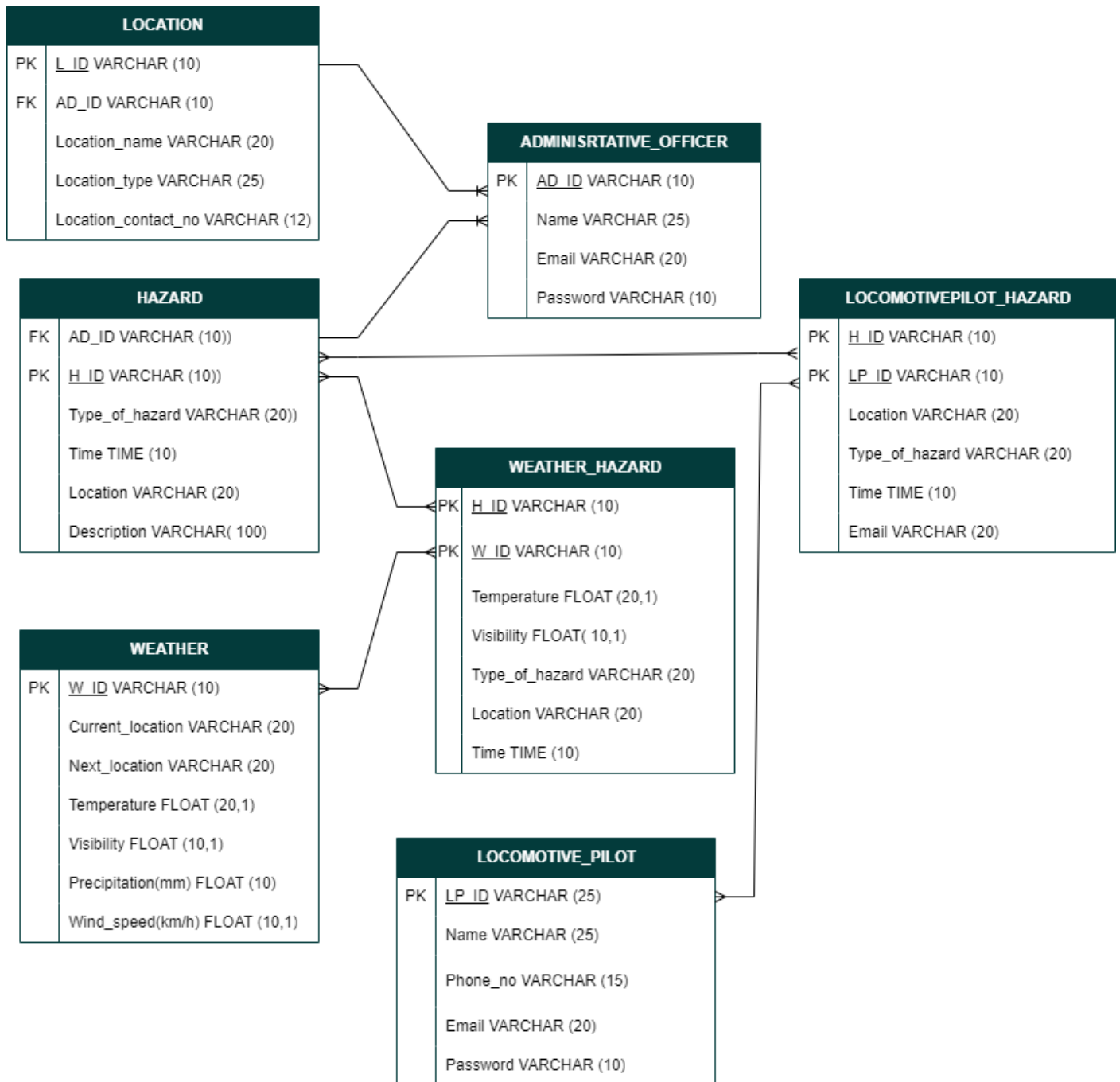


Figure 50 - Physical Database Design

## 4.8. Security Design

Database security includes a range of security controls designed to protect the Database Management System from illegitimate use and malicious threats and attacks. System security design will protect all the data that we store in the database. The security features that are planned to be implemented in our database are:

#### **4.8.1. Encryption**

The system should protect the confidentiality of data related to railway safety and prevent unauthorized parties from accessing our system's information. In our system, the locomotive pilot's registration information will be encrypted due to its importance.

#### **4.8.2. Authenticating Users**

Accept strong passwords. Here, the password must have a minimum length requirement of 8 characters. The password should include a combination of uppercase and lower-case letters, numbers, and special characters.

We planned to design a security part to protect the system from unauthorized access.

Security features that are planned to be implemented:

1. Authentication and authorization
  - Implement secure user authentication mechanisms to verify the identity of users during sign-up and sign-in processes.
  - To keep it safe, our system provides a one-time password (OTP) during the registration process, password resets, and forgotten password services.
2. Secure database management
  - Regularly backup the database to prevent data loss and facilitate system restoration in case of any unforeseen events.
  - Implement database security measures, such as strong passwords.

### **5. Research Design**

This chapter focuses on outlining objectives from the conducted literature review and formalizing high-level implementation components. The literature review serves as a basis for identifying key objectives and requirements for the project, while the formalization process involves defining the major components that will be implemented at a higher level of abstraction. By establishing these objectives and outlining the high-level components, we lay the groundwork for the subsequent phases of the project's implementation.

#### **5.1. Objective Based Literature Review**

The literature review must be done depending on the project objectives. We categorized the literature review according to the objectives of our project.

##### **5.1.1 To Develop a Model for Object Detection**

The specific technology mentioned in the text is Machine Learning, which is described as a subset of artificial intelligence. Machine learning algorithms build mathematical models based on sample data to make predictions or decisions without explicit programming. The specific technology mentioned in the text is Transfer Learning (TL). It is described as a concept in machine learning that involves leveraging knowledge gained from solving one problem and applying it to a different but related problem. In the example given, the knowledge gained while learning to recognize elephants and bulls could be used when trying to differentiate between elephants and other animals.

The specific technology mentioned in the text is Automated System Operations (ASO). ASO refers to the combination of software and hardware that enables computer systems, network devices, hardware devices, or machines to operate without manual intervention. ASOs allow systems to function without the physical presence of a human operator at the installation site. The specific technology mentioned in the text is Convolutional Neural Networks (CNNs), which are a type of neural network known for their effectiveness in image recognition, identification, and classification tasks. CNNs have been particularly successful in identifying faces, objects, human gestures, and animals.[1]

The specific technology mentioned in the text is related to automatically identifying, counting, and describing wild animals in camera trap images using deep learning techniques. Mohomed Sadegh Norouzi highlights the use of motion sensors "Camera Traps" to collect wildlife pictures inexpensively. They also demonstrate that deep learning, a cutting-edge type of artificial intelligence, can automatically extract

information from these images. The system achieved an impressive accuracy of 93.8% in identifying animals clearly, showcasing the effectiveness of deep learning algorithms in wildlife image analysis.[1]

The technology discussed here revolves around rail track segmentation, a critical task in intelligent transportation systems. It's a significant research area, and the approaches for rail track segmentation are categorized into traditional image processing methods and deep convolutional neural networks (CNNs). On the other hand, proposed a rail track extraction method using a Line Segment Detector (LSD) to detect rail segments in images. Additionally, they utilized Histograms of Oriented Gradients (HOG) features based on part models for vehicle detection. The specific technology mentioned in the text is related to real-time human detection and tracking using skin color-based methods for face detection. The authors, Swapnil V. Tathe and Sandipan P. Narote, have proposed a system that utilizes the YCbCr color model to detect skin regions. Additionally, they employ the Mean Shift algorithm and Kalman filter algorithm for tracking human subjects.[2]

### **5.1.2 Development of a Model for Real-Time Location and Weather Prediction**

Remote access has become an integral part of cloud computing, allowing users to access applications and data stored on remote servers via the internet. This section introduces the concept of remote access in the context of cloud computing and its significance in modern IT environments. Platform as a Service (PaaS) is a cloud computing model that provides clients with a platform to develop, deploy, and manage applications without the complexity of managing underlying infrastructure. This paper presents an integrated weather monitoring system leveraging the DL2e Data Logger for remote sites and laboratory applications.

The system utilizes various sensors, including thermometers, barometers, hygrometers, anemometers, pyrometers, and rain gauges, to gather weather-related data. The DL2e Data Logger offers ease of implementation, data logging capabilities, and data analysis features, enhanced by the Ls2Win PC software package. The system provides real-time monitoring, data collection, and graphical analysis for improved weather observation and analysis. This paper provides an overview of Application Programming Interfaces (APIs) and their role in software development. APIs serve as bridges between different software components, providing routes, tools, and protocols for building applications.

The paper discusses the significance of APIs in facilitating interactions between software applications and users, emphasizing their use in creating graphical user interfaces (GUIs). This document explores the utilization of JSON (JavaScript Object Notation) for generating datasets, particularly for the purpose of enabling other developers to utilize these datasets. JSON serves as a versatile format for structuring data, and its integration with JavaScript libraries like AJAX (Asynchronous JavaScript and XML) and jQuery further enhances its functionality.

The document delves into the significance of JSON in dataset formatting and its seamless integration with AJAX and jQuery libraries. This thesis explores the utilization of AJAX (Asynchronous JavaScript and XML) as a tool for dynamically altering and accessing web servers without page reloads. Specifically, AJAX is employed to draw graphs on web pages, enhancing the user experience by providing real-time and interactive content. The document delves into the functionality of AJAX, its implementation for graph rendering, and its impact on web page interactivity.[3]

### **5.1.3. Developing of web application**

Our project scope encompasses the development of a comprehensive web application with a primary focus on four key functionalities: object detection, real-time current weather updates, predictive next weather updates, and potential hazard identification. The object detection module utilizes advanced image processing techniques and deep learning algorithms, implemented using libraries such as OpenCV and TensorFlow. This allows for accurate detection and classification of various objects, including animals, vehicles, and humans, within the railway environment.

The use of convolutional neural networks (CNNs) enhances the system's ability to recognize complex patterns and objects, contributing to improved railway safety by alerting personnel to potential obstacles on the tracks. The real-time current weather update feature integrates with reliable weather APIs or services such as OpenWeatherMap or Weather Underground to fetch up-to-date meteorological information. The backend

of the application is developed using Python and Django framework, providing a robust and scalable architecture for data processing and API integration. The frontend interface is designed using HTML, CSS, and JavaScript, ensuring a responsive and user-friendly experience for railway operators and personnel accessing the application.

Additionally, the predictive next weather update module leverages machine learning algorithms such as random forests or recurrent neural networks (RNNs) to forecast weather conditions based on historical data and current trends. This predictive modeling is implemented using Python's scikit-learn or TensorFlow libraries, allowing for accurate weather forecasting and proactive decision-making for railway management.

Furthermore, our web application includes a potential hazard identification system that utilizes a combination of data analytics tools, geospatial analysis techniques, and real-time sensor inputs. The backend database is powered by PostgreSQL or MongoDB, providing efficient data storage and retrieval capabilities for hazard identification and historical incident analysis.

Overall, our project leverages a stack of technologies including Python, Django, OpenCV, TensorFlow, HTML, CSS, JavaScript, and data analytics tools to deliver a robust, scalable, and feature-rich web application tailored to the safety and operational needs of the railway industry

Our research-focused web application adopts the MVC (Model-View-Controller) pattern, dividing the system into three interconnected components. The Model handles data processing and business logic using advanced algorithms and libraries like OpenCV and TensorFlow, managing object detection, weather updates, predictive modeling, and hazard identification. The View component, designed with HTML, CSS, and JavaScript, delivers a user-friendly interface for accessing real-time information. Meanwhile, the Controller, built with Python and Django, orchestrates data flow, processes user input, and ensures seamless communication between the Model and View. This structured approach enhances modularity, maintainability, and scalability, making our application well-suited for addressing railway safety and operational challenges in a research context.

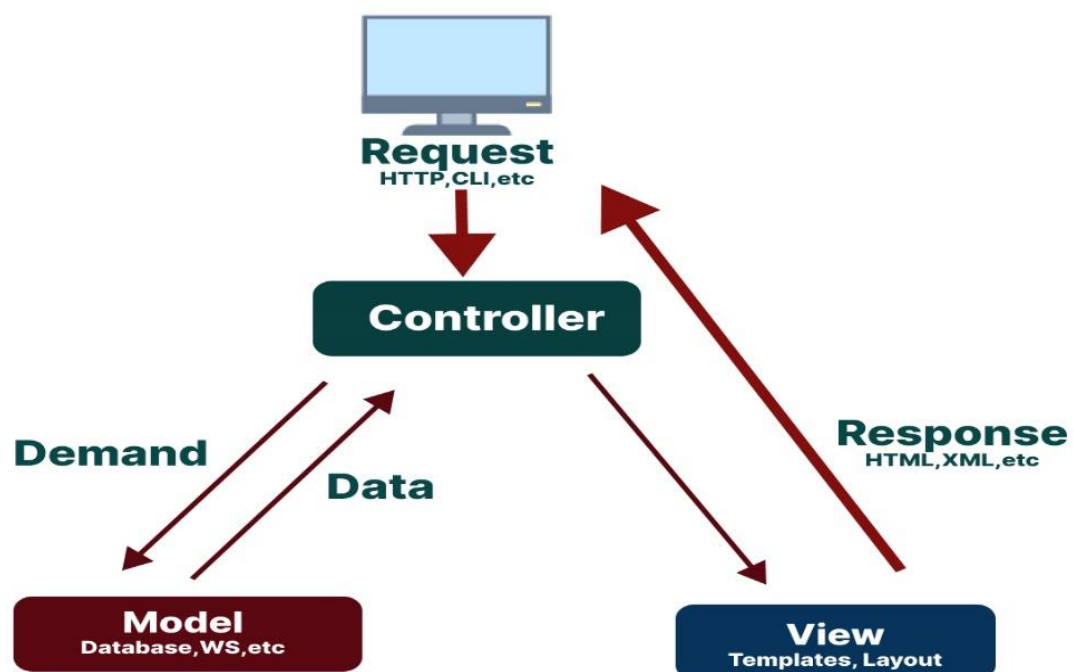


Figure 501- MVC pattern



## 5.2. Formalizing high-level implementation components

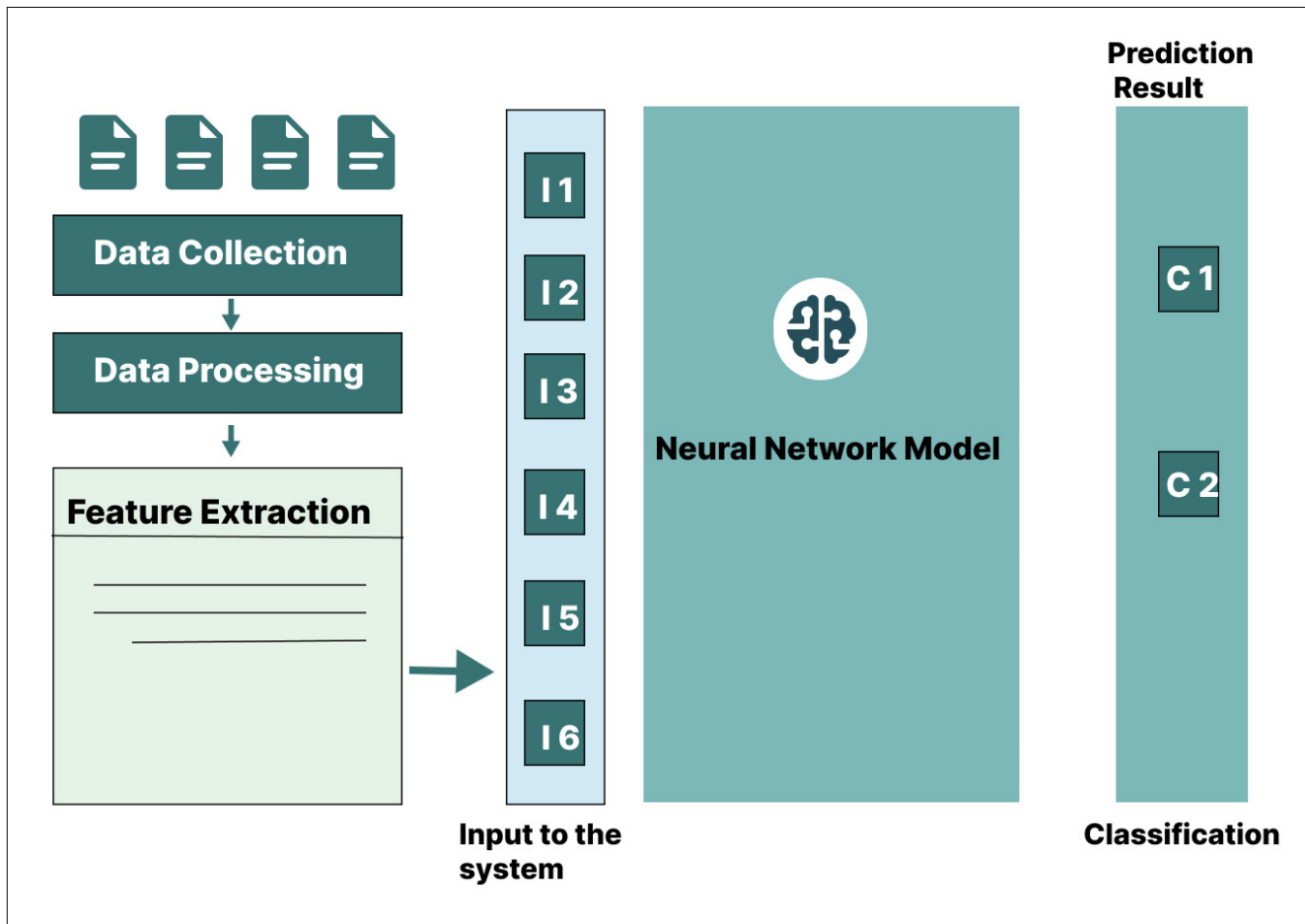


Figure 51 - High-level implementation component

**Object Detection Module:** The scope involves implementing advanced image processing techniques using OpenCV for real-time object detection and developing deep learning models with TensorFlow for accurate classification of obstacles such as animals, vehicles, and humans. This integration ensures seamless data processing and analysis within the web application environment, enhancing its functionality and usability.

**Real-Time Weather Updates:** Integrate reliable weather APIs such as OpenWeatherMap to fetch current meteorological data. Develop backend services in Python to handle real-time data retrieval and processing. Design frontend components using HTML, CSS, and JavaScript to display weather updates in a user-friendly interface.

**Predictive Weather Forecasting:** Implement machine learning algorithms like recurrent neural networks (RNNs) using scikit-learn or TensorFlow for weather forecasting. Train models use historical weather data and current trends to predict future weather conditions. Develop APIs and services to deliver predictive weather updates to the web application.

**Potential Hazard Identification:** Utilize data analytics tools and geospatial analysis techniques to identify potential hazards along railway tracks. Integrate real-time sensor inputs and historical incident data for hazard prediction and prevention. Implement a backend database MongoDB to store hazard-related information and support analytical processes.

**Integration and Scalability:** Ensure seamless integration of all modules and components within the web application architecture. Implement scalable solutions using cloud services like Google Cloud Platform

for efficient data storage, processing, and deployment. Conduct thorough testing and validation of the implemented components to ensure reliability, accuracy, and performance.

**Documentation and Maintenance:** Create comprehensive documentation for each implementation component, including codebase, APIs, database schemas, and system architecture. Establish maintenance protocols and procedures to support ongoing updates, enhancements, and bug fixes for the web application.

### 5.3. Data extractions, sample design, test data sets, training data sets

**Data Extractions:** Identify and extract relevant data sources for each module, including image datasets for object detection, real-time weather data for weather updates, historical weather data for predictive forecasting, and incident reports for hazard identification. Develop scripts or data pipelines to extract, transform, and load (ETL) data into the application's database or storage systems. Ensure data quality and consistency through data cleansing and validation processes.

**Sample Design:** Define sample design methodologies for different data types, such as random sampling for testing datasets and stratified sampling for training datasets. Determine sample sizes based on statistical considerations and the desired level of confidence and accuracy for each module's functionality. Document sample design procedures, including sampling techniques, population characteristics, and sampling error considerations.

**Test Data Sets:** Create test data sets with representative scenarios and edge cases to validate the functionality and performance of each module. Generate synthetic data or use real-world data with known outcomes for testing object detection, weather updates, weather forecasting, and hazard identification. Develop test cases and scenarios based on user stories and use cases to cover a wide range of potential scenarios and inputs.

**Training Data Sets:** Curate training data sets for machine learning models used in object detection, weather forecasting, and hazard identification. Ensure training data sets are diverse, balanced, and representative of the target environment and scenarios. Label training data sets appropriately for supervised learning tasks and annotate data for object detection bounding boxes, weather categories, or hazard classifications.

**Data Management and Versioning:** Establish data management practices to organize, store, and version control test data sets, training data sets, and extracted data sources. Implement data versioning strategies to track changes and updates to data sets, ensuring reproducibility and consistency in testing and training procedures. Document data schemas, metadata, and data lineage to facilitate data governance and traceability throughout the development and testing lifecycle.

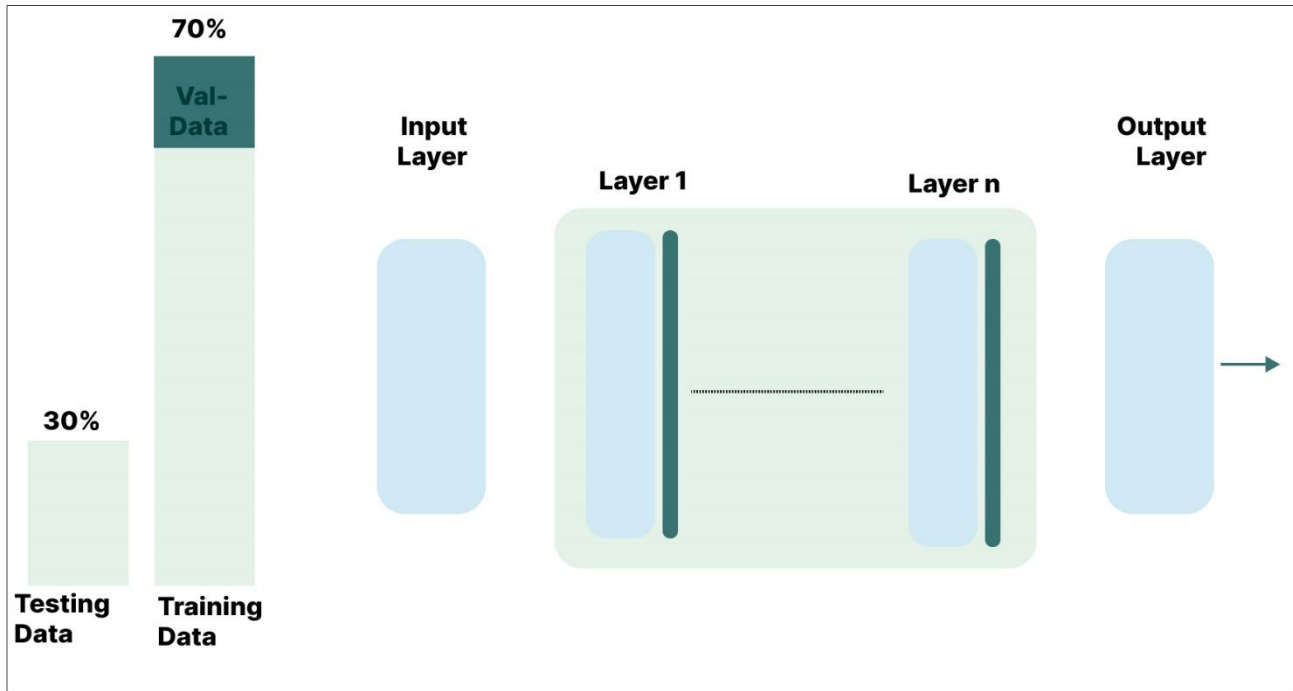


Figure 52- Performance analyzer neural network

## 5.4. Non-functional aspects

### 5.4.1. Product requirement

**Responsive** The application is designed to be responsive to screen resolution on desktop (1440x1024), and tablets (1024 x 768).

To ensure efficient performance, particularly with large datasets, we employ a performance-tuning strategy. When queries access tables with a substantial number of rows, we anticipate potential performance issues. To mitigate this, we've designed the system to store evaluated points in separate tables within the Mongo DB server. This approach optimizes execution time, ensuring that processing times for queries accessing individual tables are shorter compared to those accessing a single table with many rows.

### 5.4.2. Organizational Requirement

We will adopt Python as the primary programming language. Python is chosen for its versatility, extensive library support, and robust ecosystem, which aligns well with our project's objectives.

Leveraging Python's rich set of libraries, including TensorFlow, OpenCV, and Flask, we aim to harness the power of computer vision and cloud-based technologies effectively. Python's readability and ease of use will facilitate collaboration among team members and streamline the development process. Additionally, Python's compatibility with various platforms and operating systems ensures flexibility in deployment, allowing seamless integration with existing railway safety systems.

### 5.4.3. External Requirements

SQL encryption plays a pivotal role in securing sensitive data stored within databases. Converting data into an unreadable format without the decryption key ensures protection against unauthorized access. In our application, SQL encryption will be employed to safeguard stored passwords, fortifying our defenses against potential security breaches.






## 6.References

[1]Reducing Frequent Killing of Elephants from train collisions using Machine Learning CT Wijewantha1 , WPJ Premarathne2 , and WMKS Ilmini3 Faculty of Computing, General Sir John Kotelawala Defence University, Ratmalana, Sri Lanka

- [2] *Railroad Near-Miss Occurrence Detection and Risk Estimation System with Data from Camera Using Deep Learning*, 1st Amartuvshin Dagvasumberel Department of Railway Specialty Railway Institute of Mongolia Ulaanbaatar, Mongolia
- [3] *Mobile Application for Visualizing Weather Data in Oman Based Cloud Computing* Haitham A. Al-Balushi, Jabar H. Yousif, Hussein A Kazem
- [4] S. Harini, V. Abhiram, R. Hegde, B. D. D. Samarth, S. A. Shreyas and K. H. Gowranga, "A smart driver alert system for vehicle traffic using image detection and recognition technique," 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, 2017, pp. 1540-1543, doi: 10.1109/RTEICT.2017.8256856.
- [5] Kapoor, R., Goel, R. & Sharma, A. An intelligent railway surveillance framework based on recognition of object and railway track using deep learning. *Multimed Tools Appl* 81, 21083– 21109 (2022). <https://doi.org/10.1007/s11042-022-12059-z>
- [6] Deepa, M & Raji, C & Ajina, VA & Ashla, & Azra, Afsal & Susanna, George. (2021). Railway track tracer system for creature detection. *IOP Conference Series: Materials Science and Engineering*. 1055. 012041. 10.1088/1757-899X/1055/1/012041.
- [7] Bandara, R. M. S., & Jayasingha, P. (2018). Landslide disaster risk reduction strategies and present achievements in Sri Lanka. *Geosciences Research*, 3(3), 21-27.
- [8] Al-Balushi, H. A., Yousif, J., & Kazem, H. A. (2018). Mobile Application for Visualizing Weather Data in Oman Based Cloud Computing. *International Journal of Computation and Applied Sciences IJOCAAS*, 4(1).
- [9] Kapoor, R., Goel, R. & Sharma, A. An intelligent railway surveillance framework based on recognition of object and railway track using deep learning. *Multimed Tools Appl* 81, 21083– 21109 (2022). <https://doi.org/10.1007/s11042-022-12059-z>

## 7. Approval

### Signature of Team Members

Name	Registration Number	Index Number	Signature
M.K.M Halis	ASP/19/20/022	4856	
G.E. Vinonsan	ASP/19/20/131	4793	
M.M Mohamed	ASP/19/20/042	4858	
Mathurika. J	ASP/19/20/097	4838	
M.S.F Sumaiya	ICT/19/20/112	5048	

**Date:** 2024/05/04

### Recommendation of the supervisor(s)

**Name:**

**Department / Organization:**

**Signature:**