

Marathwada Shikshan Prasarak Mandal's  
**Deogiri Institute of Engineering and Management Studies,  
Chatrapati Sambhajinagar**

**Project Report**

**on**

# **Accident Detection & Reporting System**

Submitted By

**Sarode Siddhant Suraj (CS4261)**

**Patil Sahil Sanjay (CS4262)**

**Dr. Babasaheb Ambedkar Technological University  
Lonere (M.S.)**



Department of Computer Science and Engineering  
**Deogiri Institute of Engineering and Management Studies,  
Chatrapati Sambhajinagar**  
(2024- 2025)

**Project Report**  
**on**  
**Accident Detection & Reporting System**

Submitted By

**Sarode Siddhant Suraj (CS4261)**

**Patil Sahil Sanjay (CS4262)**

**In partial fulfillment of**  
**Bachelor of Technology**  
**(Computer Science & Engineering)**

Guided By

**Prof. P. N. Borase**

Department of Computer Science & Engineering  
**Deogiri Institute of Engineering and Management Studies,**  
**Chatrapati Sambhajinagar**  
(2024- 2025)

# **CERTIFICATE**

This is to certify that, the Project entitled “**Accident Detection & Reporting System**” submitted by **Sarode Siddhant Suraj (CS4261)**, **Patil Sahil Sanjay (CS4262)** is a bonafide work completed under my supervision and guidance in partial fulfillment for award of Bachelor of Technology (Computer Science and Engineering) Degree of Dr. Babasaheb Ambedkar Technological University, Lonere.

Date:

Place: Chatrapati Sambhajnagar

**Prof. P. N. Borase**

Project Guide

**Dr. S. C. Nandedkar**

Head of Department

**Dr. S. V. Lahane**

Dean Academics

**Dr. U. D. Shiurkar**

Director

## DECLARATION

This is to certify that, the Project report entitled, “**Accident Detection & Reporting System**” Submitted by **Sarode Siddhant Suraj (CS4261)**, **Patil Sahil Sanjay (CS4262)** is a bonafide work completed under my supervision and guidance in partial fulfillment for award of bachelor’s degree in computer science and engineering of Deogiri Institute of Engineering and Management Studies, Chatrapati Sambhajanagar under Dr. Babasaheb Ambedkar Technological University, Lonere.

Place: Chatrapati Sambhajanagar

Date:

External Examiner

**Prof. P. N. Borase**

Guide

## **Abstract**

In this world of transportation road accidents are the main cause of death & injury. Traditional accident reporting causes delay in emergency response and due to that people not getting help in an emergency and due to these delays, sometimes people die. This project aims to address these delays by creating an automated Smart Accident Detection & Reporting System using IoT, Machine Learning, Web Technology & Cloud Technology. This system uses street cameras to detect accidents in real-time, check the severity and report them to emergency services using Web Portal.

# Contents

List of Screens	i
List of Tables	ii
List of Figures	iii
<b>1. INTRODUCTION</b>	<b>1</b>
1.1 Introduction	
1.2 Necessity	
1.3 Objectives	
1.4 Theme of the Project	
<b>2. LITERATURE SURVEY</b>	<b>4</b>
2.1 Existing Solutions for Accident Detection	
2.2 Gaps in Existing Solutions	
2.3 Research on Accident Detection Using AI	
2.4 Evolution of Accident Detection Systems	
2.5 Role of Machine Learning	
2.6 Comparative Study of Existing Accident Detection Systems	
<b>3. SYSTEM DEVELOPMENT</b>	<b>12</b>
3.1 Requirement Specification	
3.1.1 Functional Requirements	
3.1.2 Non-Functional Requirements	
3.1.3 System Architecture	
3.1.4 DFD	
3.1.5 UML Diagrams	
3.2 User Interface Design	
3.3 Database Design	
3.5 Technology Used	
3.6 Security Measures	
<b>4. PERFORMANCE EVALUATION</b>	<b>40</b>
<b>5. CONCLUSION</b>	<b>48</b>

## **REFERENCES**

**50**

## **ACKNOWLEDGEMENT**

## **List of Screens**

<b>Figure</b>	<b>Illustration</b>	<b>Page</b>
3.1	Home Page	21
3.2	Login Page	22
3.3	Registration Page	23
3.4	Reports Page	24
3.5	Dashboard	25
3.6	Live Page	26
3.7	Users Page	26
3.8	Jobs Page	27
3.9	Emergency Services	28
3.10	Accident Reporting	29



## **List of Tables**

<b>Figure</b>	<b>Illustration</b>	<b>Page</b>
2.1	Key Milestones in Accident Detection Evolution	8
2.2	Advantages of Using Machine Learning in Accident Detection	10
2.3	Existing Accident Detection Systems	11
4.1	Test Cases for System Performance Metrics	42
4.2	Test Cases for UI and Portal Performance	46

## **List of Figures**

<b>Figure</b>	<b>Illustration</b>	<b>Page</b>
3.1	System Architecture	15
3.2	DFD Level 0	16
3.3	DFD Level 1	17
3.4	DFD Level 2	19
3.5	Use Case Diagram	20

# INTRODUCTION

## 1.1 Introduction

Accidents are a significant cause of death and injury across the world. Globally, road accidents claim more than 1.35 million lives each year and leave millions injured. India, in particular, has one of the highest accident rates, with thousands of lives lost on roads annually. Delays in accident detection and response times often worsen the severity of injuries, and in many cases, the fatality rate could be reduced with faster medical attention.

To address this issue, our project introduces a Smart Accident Detection & Reporting System, designed to improve the response to road accidents. By leveraging modern technologies such as Internet of Things (IoT), Machine Learning (ML), Cloud Services, and Web Technologies, the system aims to automatically detect accidents, identify the severity, and send real-time alerts to emergency services.

The system will utilize existing infrastructure, such as street cameras installed in urban areas, to continuously monitor road traffic. By analyzing the camera feeds and detecting accidents, the system will generate real-time reports, including the accident's location, vehicle details, and a severity assessment. This information is then relayed to the appropriate authorities, ensuring that emergency services can be dispatched in the shortest possible time.

## 1.2 Necessity

There is a growing need for automated accident detection systems due to the following reasons:

- **Delays in Reporting:** In many cases, accidents, especially in remote areas, go unnoticed for a long time, leading to delayed medical attention.
- **Human Error:** Relying on human witnesses to report accidents is unreliable and often results in incomplete or incorrect information.
- **Scalability Issues:** With increasing urbanization and traffic density, it becomes difficult for manual monitoring systems to track and respond to accidents quickly and accurately.

- **Need for Real-Time Reporting:** Quick response times are critical in saving lives. Traditional methods lack the ability to provide real-time data, which this system offers.

Our system addresses these needs by providing a scalable and reliable solution that works in real-time, automating accident detection and ensuring faster response times from the relevant authorities.

### **1.3 Objectives**

The primary objectives of the Smart Accident Detection & Reporting System are as follows:

- **Automated Accident Detection:** To automatically detect accidents using street camera feeds without requiring human intervention.
- **Accurate Location Tracking:** To pinpoint the exact location of the accident using GPS and alert the nearest emergency services, including police, ambulance, and fire brigade.
- **Severity Assessment:** To assess the severity of the accident by analyzing video footage and the type of vehicles involved (car, bus, truck, etc.).
- **Emergency Alerts:** To send real-time notifications to emergency services, reducing the time taken to respond.
- **Dashboard for Monitoring:** To provide a web-based dashboard for the admin, police, and emergency services where they can monitor accident data, view live footage, and manage accident reports.
- **Data Analysis:** To store and analyze historical data to identify accident-prone areas and develop strategies to prevent future accidents.
- **User-Friendly Interface:** To ensure that the system is easy to use for authorities, with a clear and intuitive interface.

### **1.4 Theme of the Project**

The project's theme revolves around the integration of modern technology to enhance road safety and accident management. The system is designed to function within the smart city framework, utilizing IoT and AI-based analytics to provide actionable insights for real-time accident management.

This system will not only benefit emergency responders but will also serve as a valuable tool for city planners and traffic management authorities by providing data-driven insights into road safety. It aligns with the broader theme of smart cities, where automation and intelligent systems are used to improve the quality of life for citizens.

# LITERATURE SURVEY

## 2.1 Existing Solutions for Accident Detection

Many accident detection systems have been proposed, and some are currently in use. These systems generally fall into two categories: in-vehicle systems and external surveillance systems.

### 1. In-Vehicle Accident Detection Systems:

These systems rely on sensors placed within vehicles to detect sudden decelerations or collisions. Examples include:

- **eCall System (Europe):** The European eCall initiative is a regulation that mandates vehicles to be equipped with in-vehicle sensors that automatically notify emergency services in the event of a severe crash. The system uses GPS and GSM technologies to report the location of the accident.
- **OnStar System (General Motors):** OnStar is a subscription-based service available in General Motors vehicles that uses in-car sensors to detect accidents. It automatically connects to emergency services upon detecting a crash.
- **Smartphone-Based Apps:** Mobile applications such as DriveAware and Accident Detector Pro use smartphone sensors (accelerometers, gyroscopes) to detect accidents, especially when a vehicle's airbags are deployed. These apps notify emergency contacts or services if a crash is detected.

### 2. External Surveillance-Based Accident Detection Systems:

These systems use external devices like cameras and roadside sensors to detect accidents. Examples include:

- **CCTV-Based Monitoring:** Many smart cities use CCTV cameras installed along highways and busy intersections to manually monitor traffic. Some cities use AI algorithms to detect accidents by analyzing camera feeds, but such systems are still in development.

- **Smart Traffic Signals:** Some advanced systems integrate sensors at intersections with traffic signals to detect accidents, measure traffic flow, and control traffic lights accordingly.
- **AI-Based Monitoring (Deep Learning):** AI systems using deep learning and image processing techniques are being developed to automatically identify traffic accidents by analyzing video footage from cameras. These systems are in the prototype stage and face challenges such as accuracy in varying weather conditions, night-time monitoring, and camera angles.

## **2.2 Gaps in Existing Solutions**

Although current solutions have made significant advancements, they have certain limitations:

### **1. In-Vehicle Solutions:**

- **Cost-Prohibitive:** Installing in-vehicle sensors is expensive and is only available in modern or high-end vehicles, limiting access for the broader population.
- **Limited Coverage:** In-vehicle systems do not address accidents involving pedestrians, cyclists, or older vehicles without advanced sensors.

### **2. External Surveillance Solutions:**

- **Manual Monitoring:** Most cities still rely on manual monitoring of CCTV footage, which is labor-intensive and inefficient. Relying on human operators can result in delays or missed accidents.
- **AI Challenges:** AI-based accident detection using video feeds has limitations related to lighting conditions, weather (rain, fog), and obstructions (trees, poles).
- **Lack of Real-Time Integration:** Current systems do not fully integrate with emergency services to provide real-time alerts and coordinated responses.

## **2.3 Research on Accident Detection Using AI**

Several research studies have explored AI-based systems for accident detection:

- **Real-Time Traffic Accident Detection Using AI (2021):** This study proposed a real-time accident detection system using deep learning and video analysis. The system was trained on large datasets of traffic accidents and demonstrated a high detection accuracy rate. However, it was noted that the system performed poorly under low-light conditions.
- **Image-Based Accident Detection for Smart Cities (2020):** This research focused on detecting accidents using images captured by CCTV cameras in smart cities. The study used image classification algorithms to identify accident scenarios but faced challenges with data processing speeds and false positives.

## 2.4 Evolution of Accident Detection Systems

The evolution of accident detection systems has been marked by significant technological advancements over the decades, revolutionizing the way accidents are reported, detected, and responded to. Historically, accident detection relied solely on human intervention, which often led to delays and inefficiencies in emergency response. Over time, this process evolved with the introduction of technology, paving the way for automated and intelligent systems.

### 2.4.1 Early Stages: Manual Reporting

In the early days, accident reporting was entirely manual, requiring witnesses or victims to contact emergency services. These methods were:

- **Time-Consuming:** Often delaying the emergency response, especially in isolated areas.
- **Prone to Errors:** Witnesses might provide inaccurate or incomplete information.
- **Lacking Standardization:** No uniform system for collecting and reporting accident data existed.

### 2.4.2 Emergence of GPS and GSM Technologies (1990s)

The 1990s saw the introduction of GPS (Global Positioning System) and GSM (Global System for Mobile Communications) technologies, which enabled semi-automated accident reporting systems:

- **Location Tracking:** Vehicles could transmit location data to emergency centers.
- **Manual Alerts:** Systems required users to trigger alerts manually in case of accidents.



While this was a step forward, it still lacked full automation and real-time capabilities.

#### 2.4.3 Integration of Telematics (2000s)

The 2000s brought telematics to the forefront, allowing vehicles to transmit data such as speed, acceleration, and impact:

- **Vehicle Diagnostics:** Onboard diagnostics (OBD) systems could detect malfunctions or collisions.
- **Automated Alerts:** Some systems automatically contacted emergency services upon detecting an accident.

#### 2.4.4 IoT and AI Revolution (2010s)

The Internet of Things (IoT) and Artificial Intelligence (AI) marked a transformative phase in accident detection:

- **IoT Devices:** Sensors in vehicles and roadside units collected real-time data on speed, impact, and environmental conditions.
- **AI Models:** Machine learning algorithms analyzed data from cameras, sensors, and other sources to detect accidents automatically.

#### 2.4.5 Current Trends (2020s)

The 2020s have witnessed the integration of advanced AI with IoT, cloud computing, and autonomous vehicles:

- **Autonomous Vehicles:** Accident detection systems now include preventive measures, using AI to predict and avoid potential accidents.
- **Cloud-Based Systems:** Centralized data processing and storage enable seamless sharing of accident data with relevant authorities.
- **Edge Computing:** Reduces latency in data processing for real-time decision-making.

#### 2.4.6 Key Milestones in Accident Detection Evolution

<b>Decade</b>	<b>Technological Advancement</b>	<b>Impact</b>
1990s	GPS and GSM	Improved location tracking
2000s	Telematics	Enhanced vehicle diagnostics and reporting
2010s	IoT and AI	Real-time accident detection
2020s	Integration with autonomous vehicles, 5G networks	Preventive accident measures and rapid alerts

Table 2.1 Key Milestones in Accident Detection Evolution

## 2.5 Role of Machine Learning in Modern Accident Detection Systems

Machine Learning (ML) plays a transformative role in modern accident detection systems, enabling real-time and accurate identification of accidents with minimal human intervention. By leveraging large datasets and advanced algorithms, ML has significantly enhanced the efficiency and effectiveness of accident detection mechanisms.

### 2.5.1 Key Applications of Machine Learning in Accident Detection

#### 1. Image and Video Analysis

- **Techniques Used:** Convolutional Neural Networks (CNNs) and Deep Learning models are employed to analyze video feeds from street cameras, dashcams, and drones.
- **Functionality:** ML models detect patterns such as vehicle collisions, pedestrian falls, or unusual traffic flow, triggering alerts in real-time.
- **Example:** An ML model trained on annotated accident datasets can detect a collision with over 90% accuracy.

## 2. Sensor Data Interpretation

- **Techniques Used:** Supervised learning algorithms analyze data from accelerometers, gyroscopes, and GPS sensors installed in vehicles.
- **Functionality:** ML algorithms recognize abnormal readings, such as sudden deceleration or high-impact forces, to identify potential accidents.
- **Example:** Predictive models can flag anomalies like a rapid drop in vehicle speed, indicative of a crash.

## 3. Traffic Pattern Analysis

- **Techniques Used:** Clustering and anomaly detection algorithms examine real-time traffic flow.
- **Functionality:** By comparing current patterns with historical data, ML identifies irregularities like roadblocks or pileups.
- **Example:** Traffic congestion near an accident site can be detected and reported to authorities.

## 4. Natural Language Processing (NLP)

- **Techniques Used:** NLP models process user-reported accident data submitted through mobile applications or social media.
- **Functionality:** Extracts key information such as location, severity, and type of accident for further processing.
- **Example:** An NLP-based chatbot allows users to report accidents with voice commands or text input.

### 2.5.2 Advantages of Using Machine Learning in Accident Detection

Advantage	Description
<b>Real-Time Processing</b>	ML enables instant analysis of data, reducing response time to accidents.
<b>High Accuracy</b>	Continuous training with new data improves the precision of accident detection.
<b>Scalability</b>	ML models can process data from multiple sources simultaneously, such as cameras and sensors.
<b>Cost-Efficiency</b>	Automated detection reduces dependency on human monitoring and interventions.
<b>Predictive Analysis</b>	Identifies potential accident-prone zones or situations, enabling preventive measures.

Table 2.2 Advantages of Using Machine Learning in Accident Detection

### 2.5.3 Challenges in Implementing ML for Accident Detection

#### 1. Data Quality and Quantity:

- High-quality, labelled datasets are required to train ML models effectively.
- Ensuring data privacy and compliance with regulations such as GDPR.

#### 2. Model Interpretability:

- Explaining ML decisions to non-technical stakeholders is often challenging.

#### 3. Edge Cases:

- Models may fail to detect rare or unusual accident scenarios due to limited training data.

#### 4. Hardware and Infrastructure Requirements:

- Real-time ML applications require robust hardware and cloud/edge computing capabilities.

## 2.6 Comparative Study of Existing Accident Detection Systems

A comparative study of various accident detection systems is provided below:

System Type	Technology	Strengths	Weaknesses
<b>In-Vehicle Systems</b>	Sensors	Quick accident detection, automated response	Expensive, limited to modern vehicles
<b>CCTV-Based Systems</b>	Manual monitoring of cameras	Widely available in cities	Human error, slow response time
<b>AI-Based Systems</b>	Deep learning & image analysis	Automated detection, scalable	Accuracy affected by lighting/weather
<b>Smartphone Apps</b>	Phone sensors (gyroscope, etc.)	Inexpensive, widely available	Dependent on smartphone sensors

Table 2.3 Study of Existing Accident Detection Systems

# SYSTEM DEVELOPMENT

System development is the critical phase where we turned our plans into a working application. This involved frontend development using React Native, backend development using Flask, integrating the machine learning model, and ensuring smooth communication between the frontend and backend. Here's a detailed look into each aspect of our development process.

## 3.1 Requirement Specification

### 3.1.1 Functional Requirements

1. Accident Detection:
  - The system must detect accidents in real-time using street camera feeds.
  - It should identify the type of accident (e.g., vehicle collision).
2. Severity Assessment:
  - The system should analyze footage and sensor data to assess the severity of the accident.
  - It must differentiate between minor, moderate, and severe accidents based on collision impact and surrounding conditions.
3. Location Tracking:
  - Use GPS or geolocation data from camera feeds to provide precise accident locations.
  - Map the accident scene on a dashboard for better visualization and navigation.
4. Reporting and Notification:
  - The system should automatically notify emergency services (police, ambulance, fire brigade) based on the severity of the accident.

- It must allow real-time alerts for citizens nearby or those subscribed to the public dashboard.

5. Footage and Evidence Storage:

- Store accident footage and reports securely in the cloud for future analysis or legal purposes.
- Maintain a timeline of events, including accident occurrence, response, and resolution status.

6. Dashboard Access for Admin and Emergency Services:

- Provide an admin panel to manage accidents, monitor statistics, and generate reports.
- Display live feeds from street cameras for emergency services, including the ability to switch between different camera locations.

### **3.1.2 Non-Functional Requirements**

1. Scalability:

- The system should be scalable to handle multiple cameras across different cities.
- It must support large-scale data storage and processing in the cloud.

2. Reliability:

- Ensure 99% uptime to maintain uninterrupted accident detection and reporting services.
- Implement fallback mechanisms for situations like network failures.

3. Security:

- Secure all data, especially personal information, footage, and location details, using encryption.
- Implement strict access control for dashboard users (admins, police, emergency services).

#### 4. Performance:

- The system should process video feeds and detect accidents in under 5 seconds.
- Ensure quick response and low latency in communication between the system and emergency services.

#### 5. Usability:

- The dashboard should have a user-friendly interface for quick navigation.
- Ensure mobile compatibility for emergency responders using tablets or smartphones.

### 3.1.3 System Architecture

- **High-Level System Architecture**

The system architecture consists of the following core components:

#### 1. Street Cameras:

- Capture real-time traffic footage.
- The cameras are integrated with motion sensors and provide continuous video streams.

#### 2. AI-Based Processing Unit:

- This unit processes the video feeds using machine learning algorithms to detect accidents.



- The AI model uses computer vision techniques to identify collision patterns and other accident indicators (e.g., skidding vehicles).

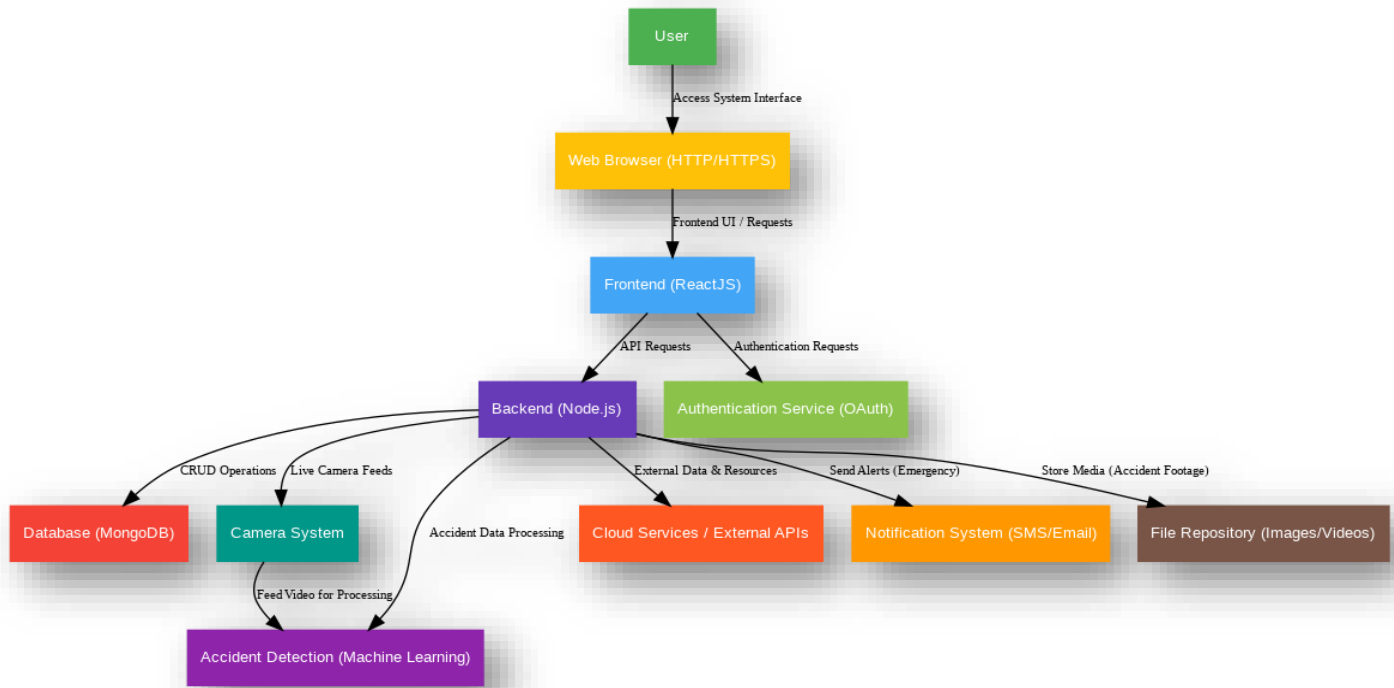


Fig 3.1 System Architecture

### 3. Cloud Storage and Computing:

- All accident-related data (footage, location, severity reports) is stored in the cloud.
- The cloud infrastructure also provides computing power to process large amounts of video data efficiently.

### 4. Dashboard and User Interfaces:

- Admin and emergency services access the dashboard to view live footage, track accidents, and generate reports.

- The public interface provides basic accident information and notifications without exposing sensitive data.

### 3.1.4 DFD (Data Flow Diagrams)

- **DFD Level 0 – Context Diagram**

This is the high-level DFD that represents the entire system as a single process with its interactions.

- Input: Street camera footage, GPS data, user login requests (admin, emergency services).
- Process: Accident detection, severity assessment, reporting.
- Output: Real-time alerts, notifications, accident reports, and access to live footage.
- Diagram

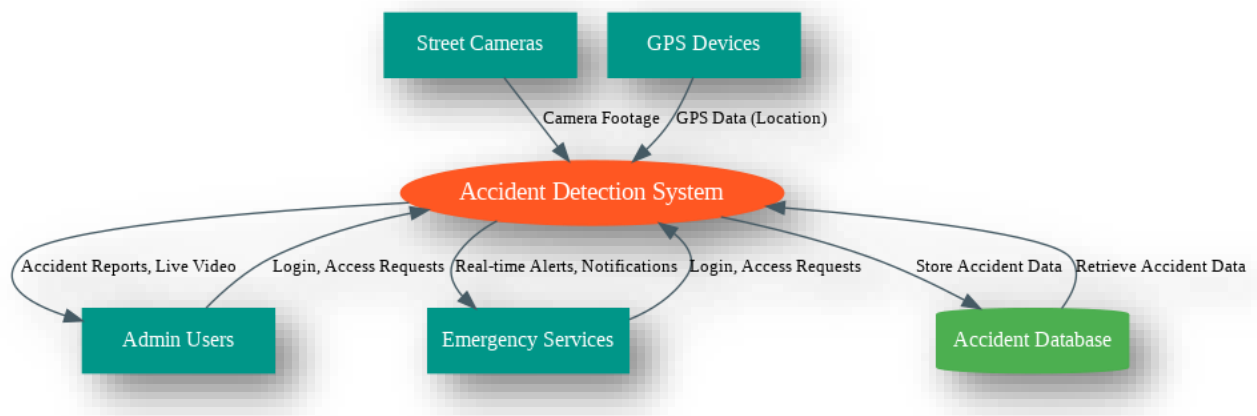


Fig 3.2 DFD Level 0

- **DFD Level 1 – Detailed Processes**

This level breaks down the system into individual processes:

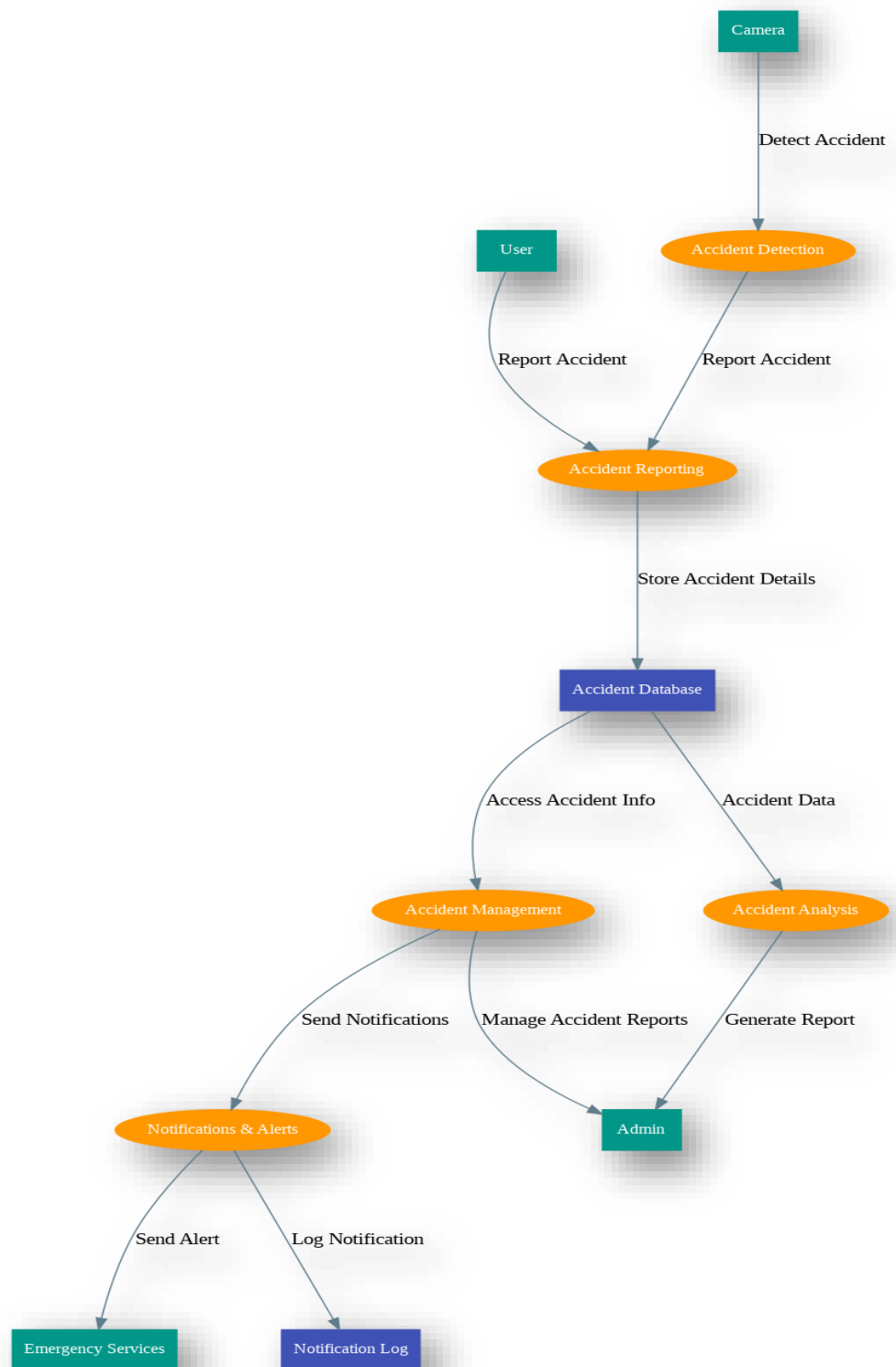


Fig 3.3 DFD Level 1

1. Accident Detection Process:

- Input: Camera footage.
- Process: AI model processes video for accident detection.
- Output: Accident confirmation and severity level.

2. Accident Reporting Process:

- Input: Accident details (location, severity, type).
- Process: Generates reports, alerts relevant authorities.
- Output: Reports sent to police, ambulance, fire services.

- **DFD Level 2 – Sub-Processes**

This level further details each component:

1. AI Processing Sub-Processes:

- Motion analysis, object detection, crash severity calculation.

2. Notification Sub-Processes:

- Choose emergency services based on accident severity, push notifications to dashboards.

### 3. Digram

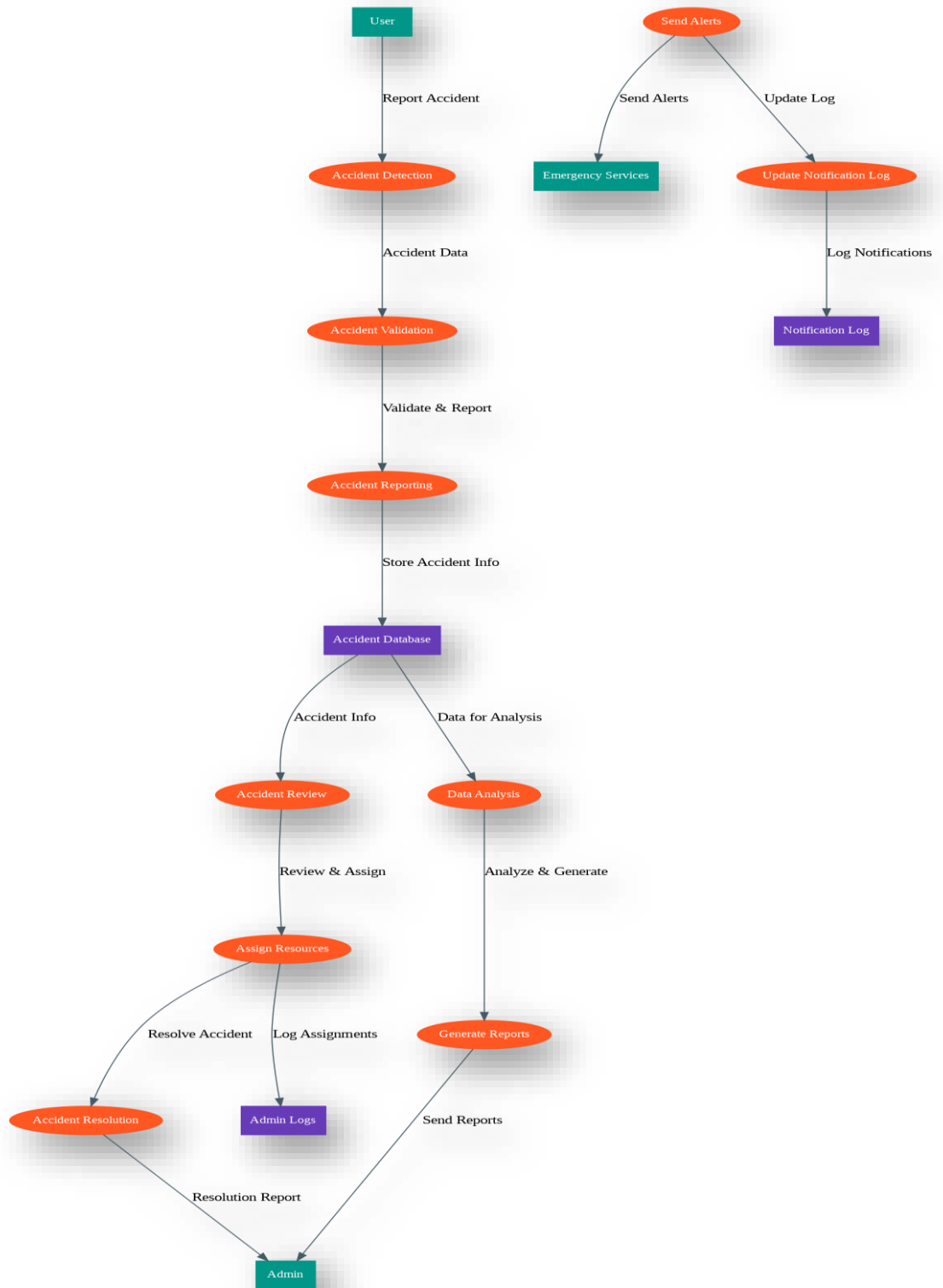


Fig 3.4 DFD Level 2

### 3.1.5 UML Diagrams

- Use Case Diagram

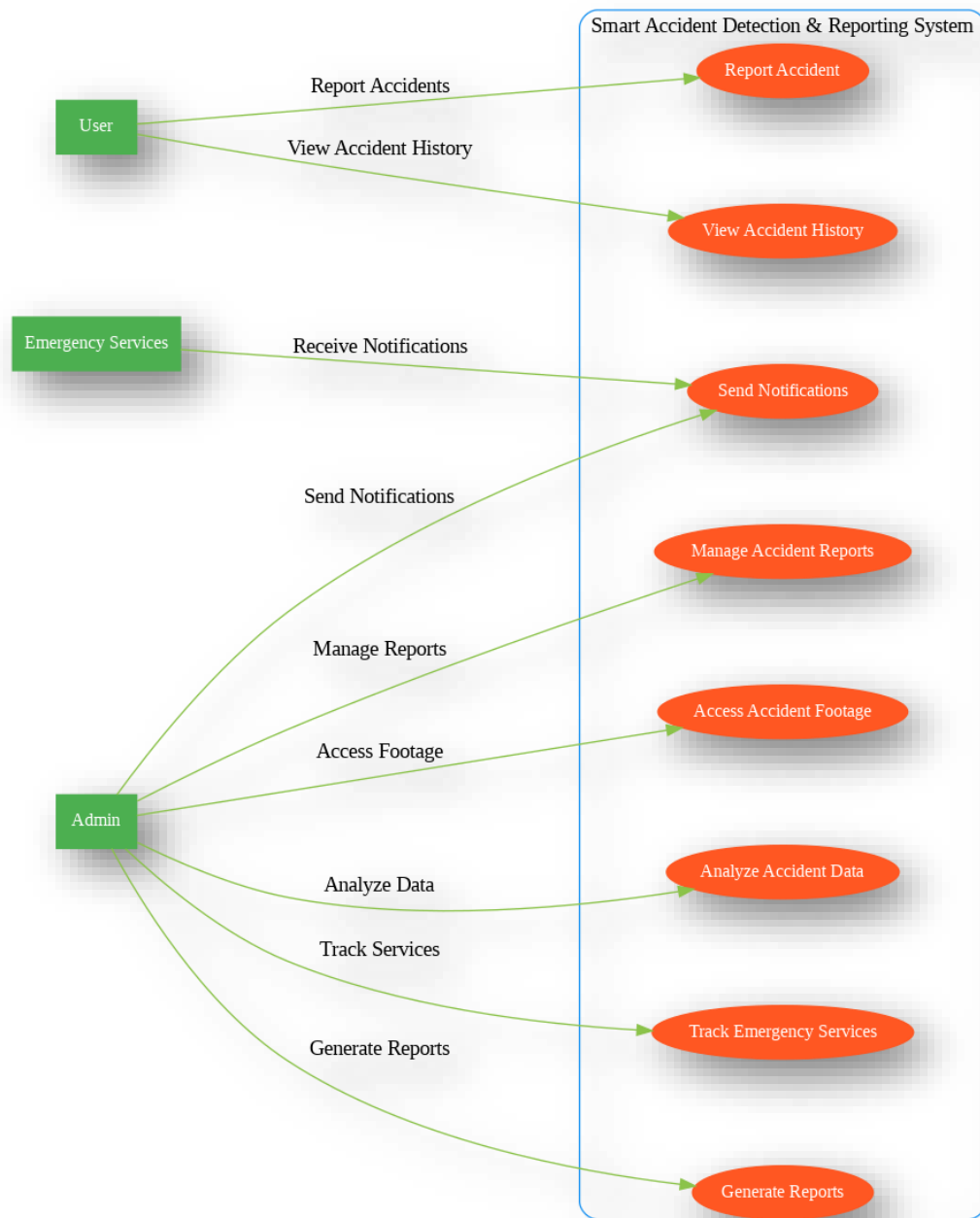
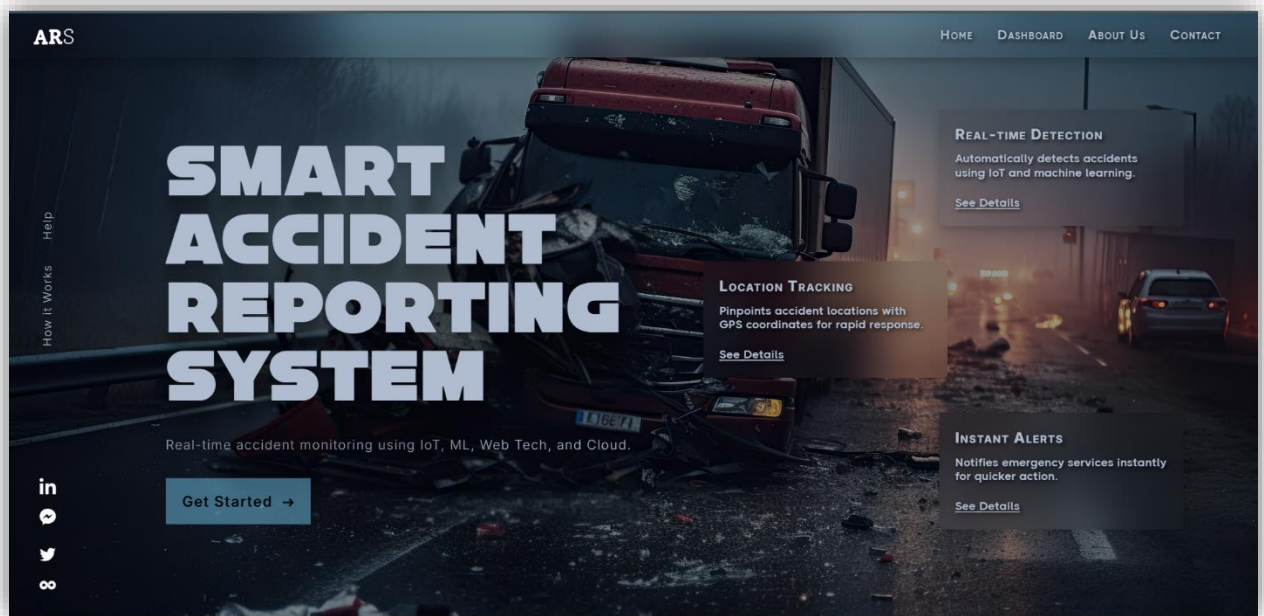


Fig 3.5 Use Case Diagram

- Actors: Admin, Police, Ambulance, Fire Brigade, Public Users.
- Use Cases: View live feeds, receive accident notifications, generate reports, track emergency responses.

## 3.2 User Interface Design

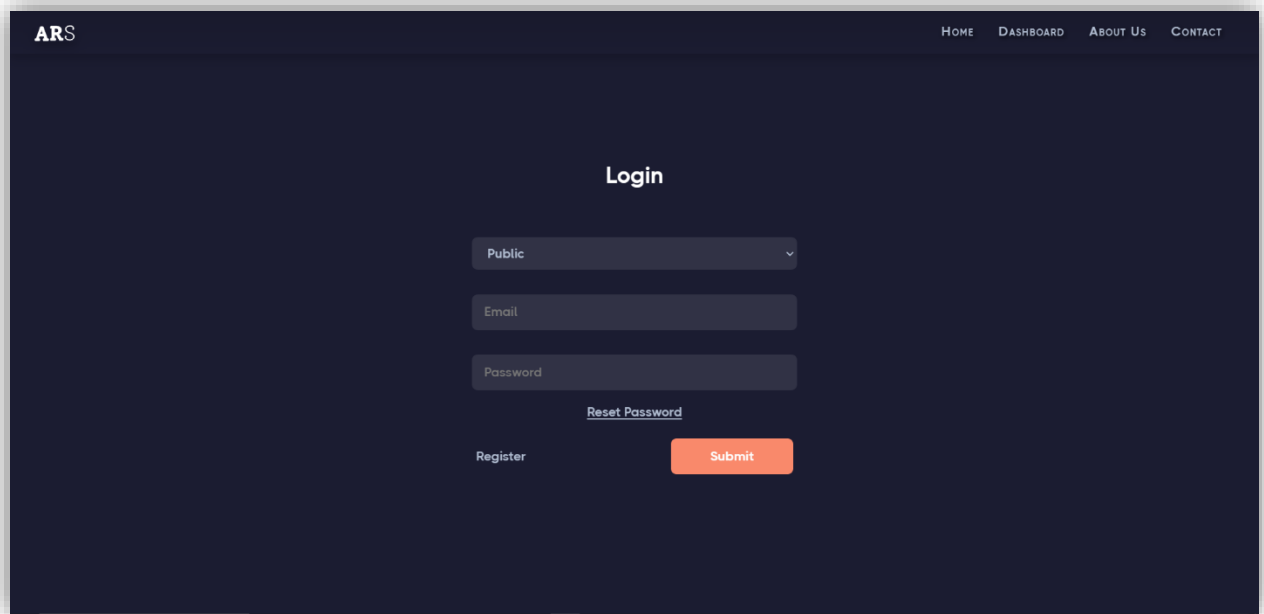
### 3.2.1 Home Page



Screen 3.1 Home Page

- Description: The Home Page serves as the landing page for the portal, presenting general information about the system, its purpose, and any recent announcements or updates.
- Key Elements:
  - Welcome message and system introduction.
  - Navigation links to login and register pages.
  - Contact information for assistance.

### 3.2.2 Login Page



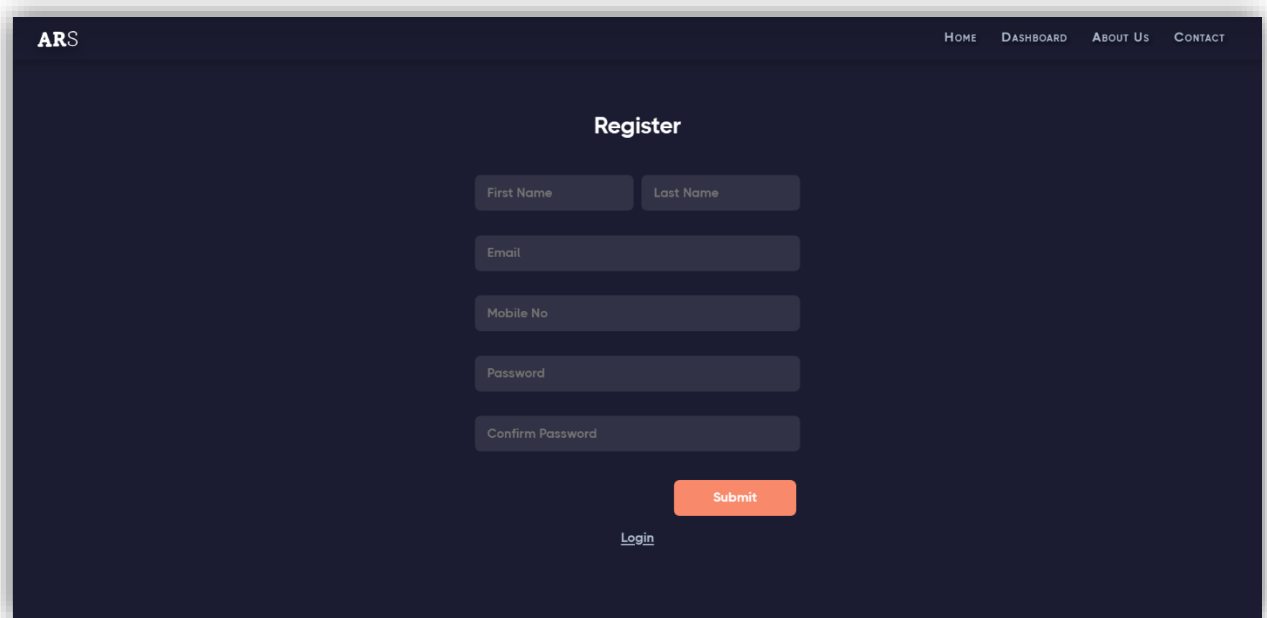
The screenshot displays the ARS Login Page. At the top left is the 'ARS' logo. The top right features a navigation menu with links for 'HOME', 'DASHBOARD', 'ABOUT US', and 'CONTACT'. The main heading 'Login' is centered. Below it is a role selection dropdown menu currently set to 'Public'. Underneath are input fields for 'Email' and 'Password'. A link for 'Reset Password' is positioned below the password field. At the bottom, there are two buttons: 'Register' and 'Submit'.

Screen 3.2 Login Page

- Description: Provides users with a secure interface to log in based on their role (e.g., admin, police, public).
- Key Elements:
  - Input fields for username and password.
  - Role-based login options, with protected access for different user types.
  - Password reset and “remember me” options for convenience.



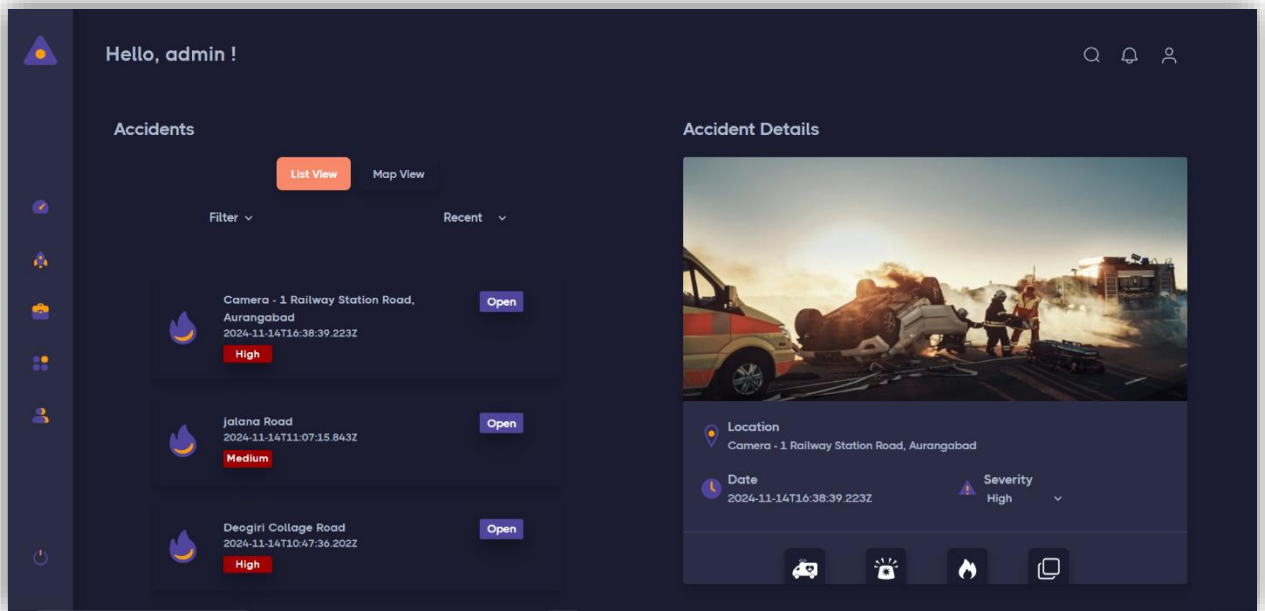
### 3.2.3 Register Page

The screenshot shows a dark-themed web page for registration. At the top left is the 'ARS' logo, and at the top right are navigation links: 'HOME', 'DASHBOARD', 'ABOUT US', and 'CONTACT'. The main heading 'Register' is centered. Below it are input fields for 'First Name', 'Last Name', 'Email', 'Mobile No', 'Password', and 'Confirm Password'. A red 'Submit' button is positioned below the 'Confirm Password' field. At the bottom center, there is a 'Login' link.

Screen 3.3 Register Page

- **Description:** New users can create an account to access the system with role-specific permissions.
- **Key Elements:**
  - Fields for personal details, contact information, and account credentials.
  - Role selection for appropriate access level.
  - Verification mechanisms for secure registration.

### 3.2.4 Reports Page

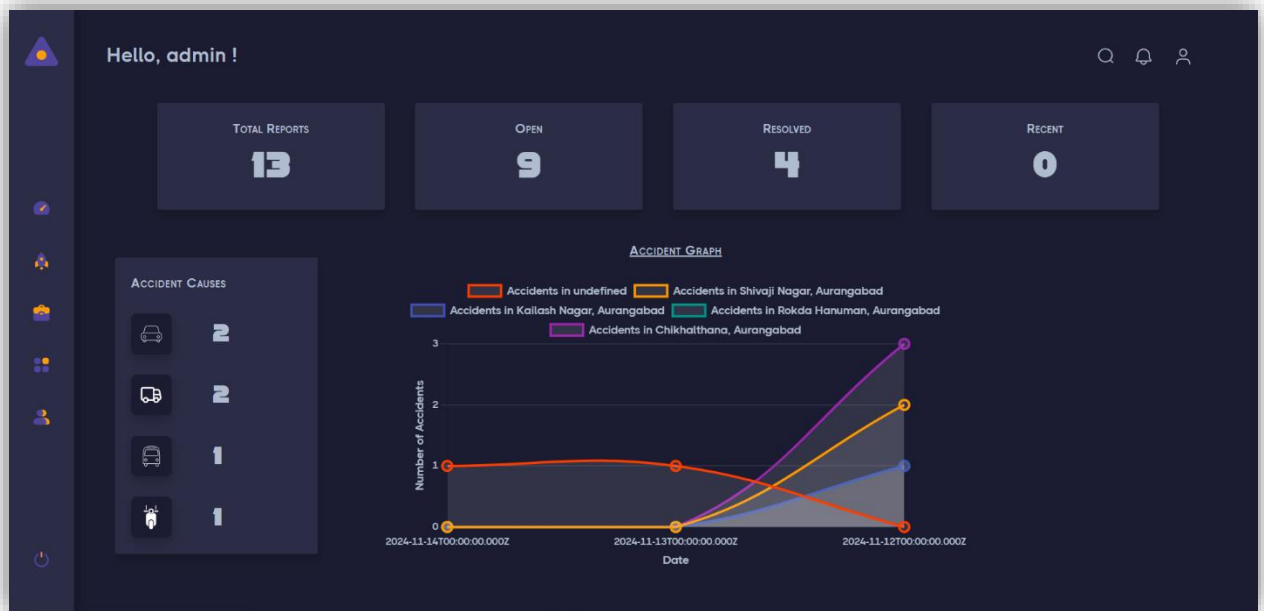


Screen 3.4 Reports Page

- **Description:** Displays a list of all accident reports that users can access based on their permissions.
- **Key Elements:**
  - Search and filter options to narrow down reports.
  - Display of accident summaries, location details, status (open/resolved), and date/time.
  - Navigation to detailed accident information upon selection of a report.

### 3.2.5 Dashboard

- **Description:** An overview page displaying key statistics, recent activity, and various metrics relevant to admins and emergency services.

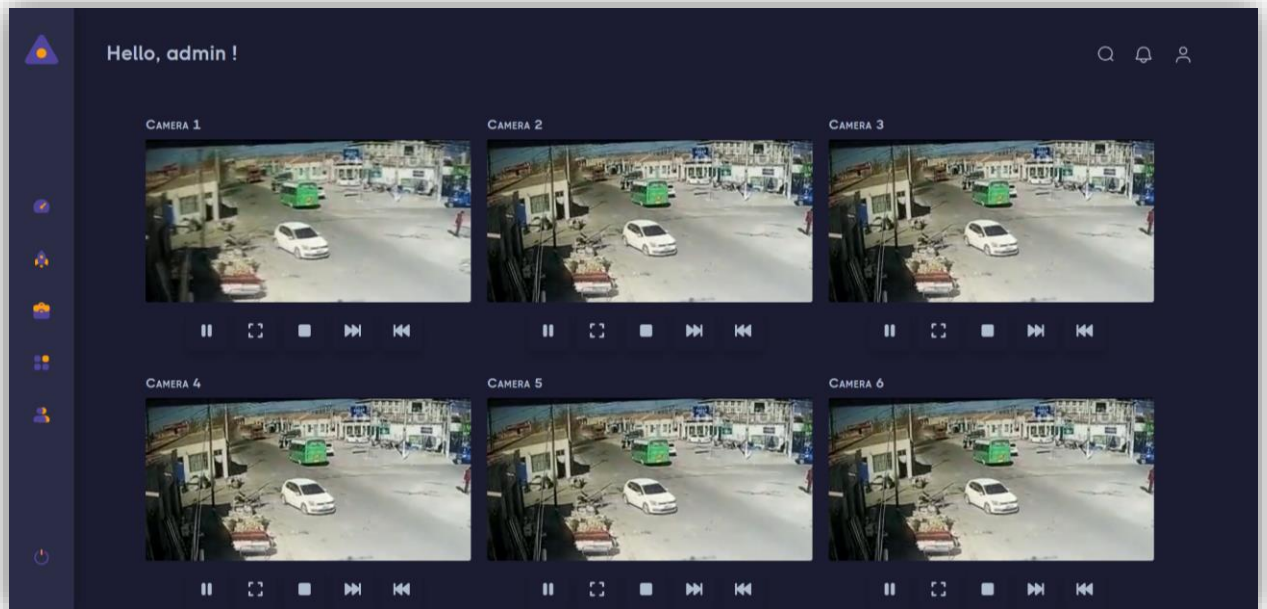


Screen 3.5 Dashboard

- **Key Elements:**
  - Summary boxes displaying accident counts (open, resolved, etc.).
  - Graphs and charts to visualize accident trends by location, type, and frequency.
  - Recent activity feed showing updates or newly reported incidents.

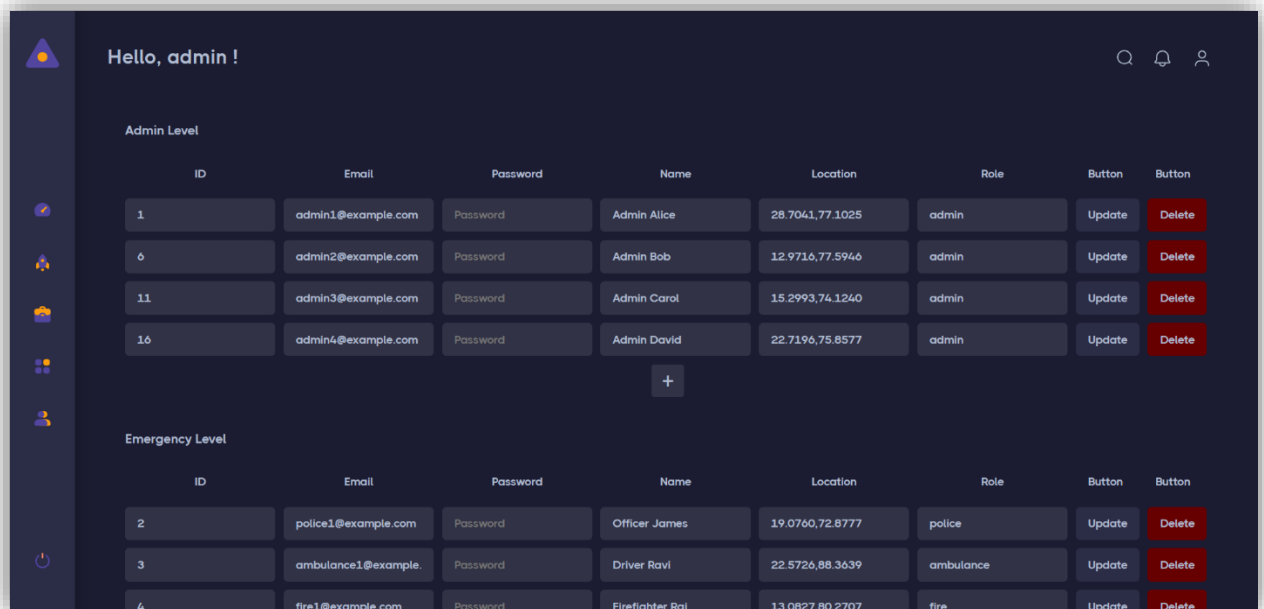
### 3.2.6 Live Page

- **Description:** Displays live footage from various street cameras for emergency service personnel.
- **Key Elements:**
  - Multiple video streams with real-time views of locations.
  - Labels indicating the location of each feed.
  - Controls for each video stream to adjust quality, pause/play, and enlarge to full screen within the window.



Screen 3.6 Live Cameras Page

### 3.2.7 User Management Page

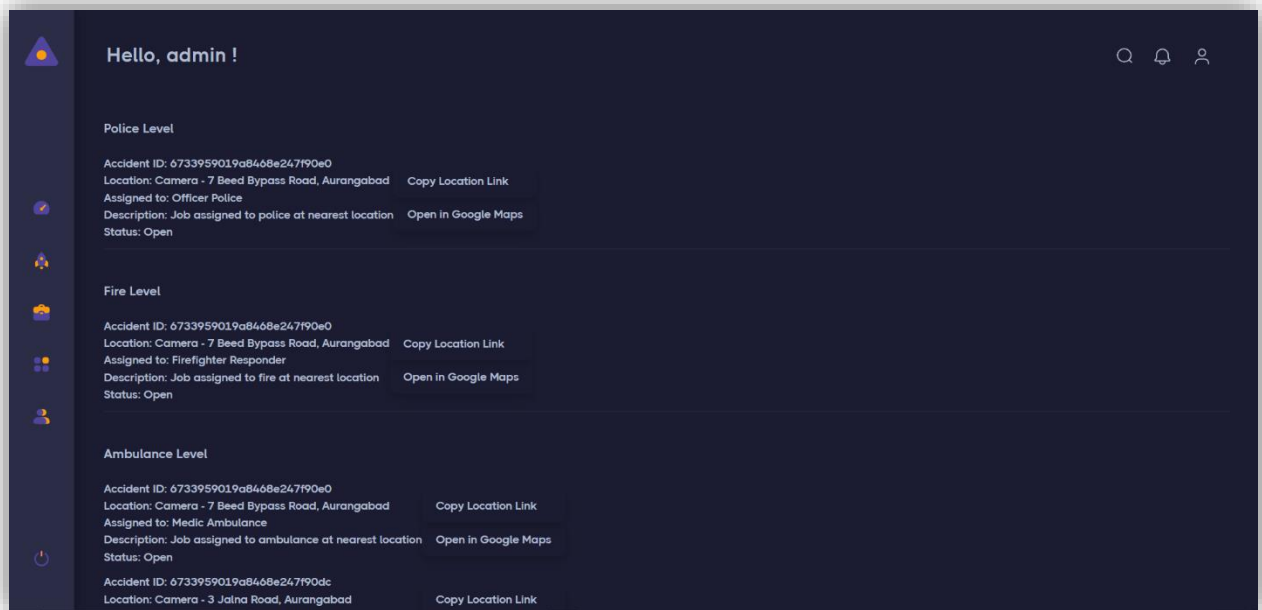


Screen 3.7 User Management Page

- **Description:** Allows admins to manage user accounts, roles, and access permissions.
- **Key Elements:**
  - List of registered users with roles, status, and last login information.
  - Options to edit roles, deactivate accounts, and reset passwords.
  - Filters and search functionality to locate specific users quickly.

### 3.2.8 Jobs Page

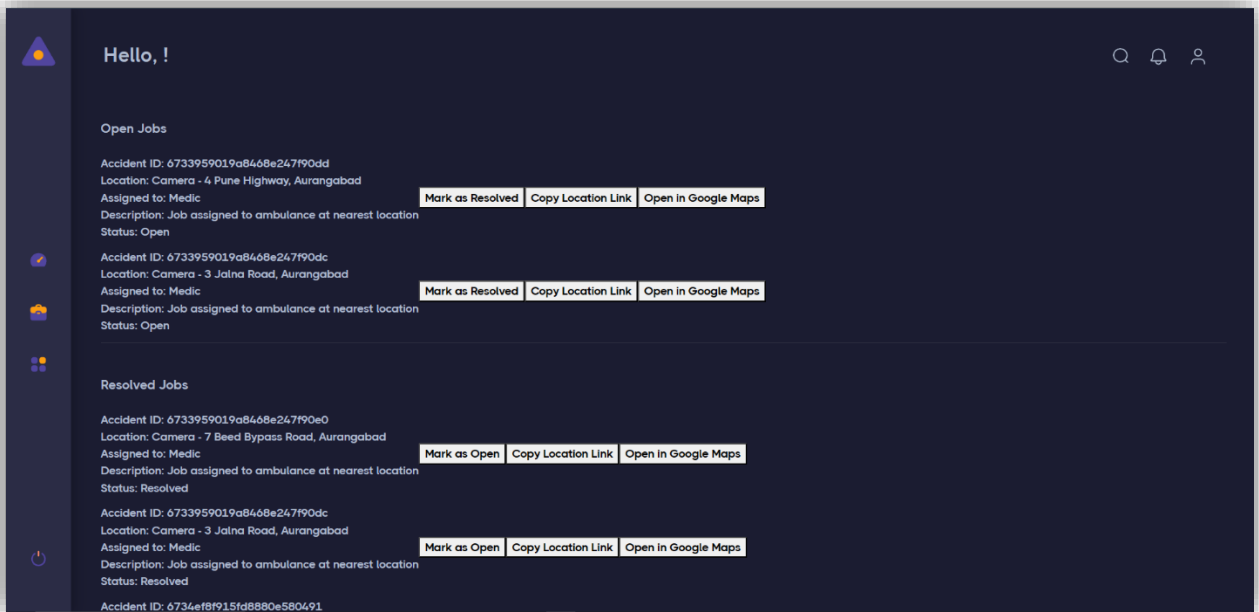
- **Description:** Displays job-related opportunities, primarily for admin-level access to post or manage.
- **Key Elements:**
  - Listing of job openings with position descriptions and requirements.



Screen 3.8 Jobs Page

- Admin options to post new jobs, edit existing postings, and review applications.
- Option for users to apply or express interest if roles permit.

### 3.2.9 Emergency Services Page



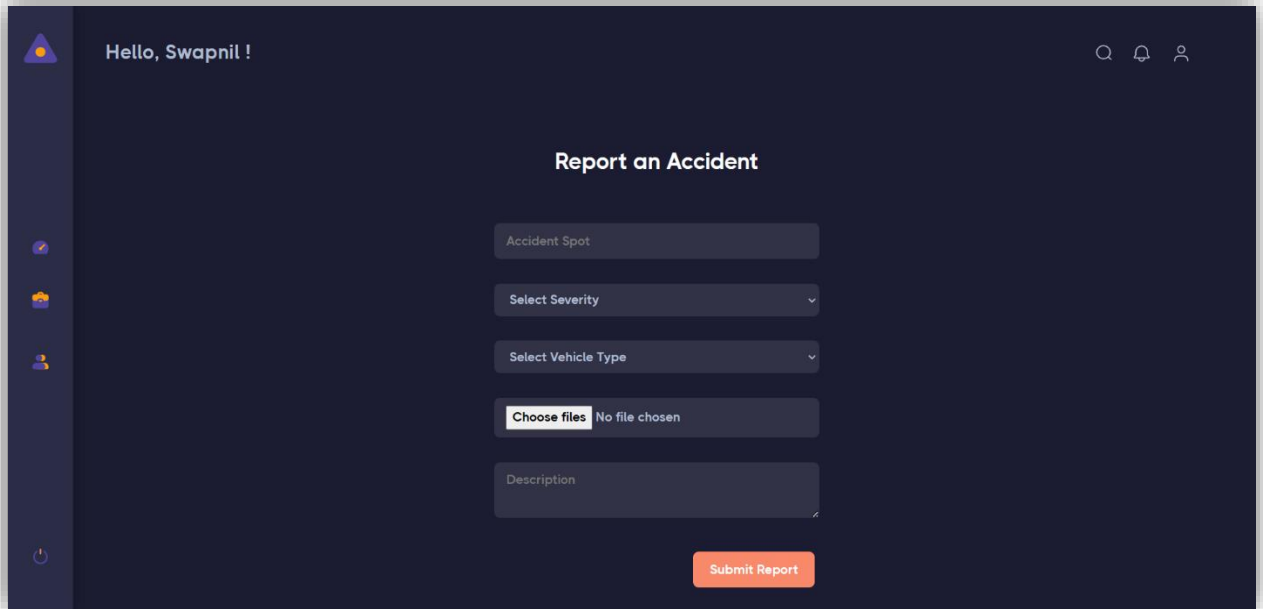
Screen 3.9 Emergency Services Page

- **Description:** Displays contact details for emergency services and relevant resources.
- **Key Elements:**
  - Emergency contact numbers (e.g., police, ambulance, fire brigade).
  - Quick-call buttons for direct communication.
  - Additional resources like response times and availability.

### 3.2.10 Report Accident Page

- **Description:** Public users can report an accident manually, specifying details and location.
- **Key Elements:**
  - Input fields for accident location, type, severity, and additional notes.
  - Option to upload images or videos, if available.

- Submit button that routes the report to the system for admin and emergency service review.



Screen 3.10 Report Accident Page

### 3.3 Database Design

In our Accident Detection and Reporting System, we use MongoDB as the primary database. MongoDB's document-oriented structure provides flexibility in data storage, allowing us to define various schemas for different types of entities in the system. Below is an overview of each database model, followed by an Entity-Relationship (ER) Diagram illustrating the relationships among entities.

#### Database Models

##### 1. Accident Model

- **Attributes:**
  - spot: The name or description of the accident spot.
  - location: Latitude and longitude coordinates of the accident.

- severity: Enum type, representing the accident's severity level (Low, Medium, High).
- type: Type of accident (e.g., vehicle collision, fire).
- status: Enum type to track the current status (Open, In Progress, Resolved).
- time: Date and time of the accident report.
- description: Detailed description of the accident.
- images: Array of image URLs or file paths associated with the accident.
- videos: Array of video URLs or file paths associated with the accident.

## 2. User Model

### ○ Attributes:

- firstName and lastName: User's first and last names.
- email: User's unique email address.
- password: Password for authentication (stored securely).
- mobile: Contact number for user.
- role: User role in the system (default is "public"; others could be "admin", "police", etc.).
- location: User's geographic location coordinates, stored as latitude and longitude.

## 3. Job Model

### ○ Attributes:

- accidentId: Foreign key reference to the Accident model to link the job with a specific accident report.



- assignedUserId: Foreign key reference to the User model, representing the user assigned to the job.
- assignmentTime: Date and time when the job was assigned.
- status: Enum type for job status (Open, In Progress, Completed).
- type: Type of job, representing the department or role required (e.g., police, ambulance, fire).
- description: Description of the job or additional notes.

## 3.4 Technologies Used

This section highlights the various technologies employed in developing the Smart Accident Detection & Reporting System. Each technology was chosen for its ability to address specific challenges and requirements within the project.

### 3.4.1 Front-End Development

The user interface (UI) of the system is designed to provide a seamless and intuitive experience. The following technologies were used:

#### 1. **React.js:**

- **Purpose:** React.js was chosen for its component-based architecture, allowing efficient and reusable UI components.
- **Advantages:**
  - Virtual DOM ensures faster UI updates.
  - React Router supports single-page application behavior for smooth navigation between pages.
  - Rich ecosystem for state management, such as Redux.

## 2. **Bootstrap and Material-UI:**

- **Purpose:** These libraries enhanced the aesthetic appeal and responsiveness of the UI.
- **Advantages:**
  - Predefined UI components reduced development time.
  - Mobile-first design principles ensured compatibility across devices.

### 3.4.2 Back-End Development

The server-side of the system handles data processing, API management, and system logic. The following technologies were used:

#### 1. **Node.js:**

- **Purpose:** Node.js was selected for its non-blocking, event-driven architecture, which ensures scalability and performance.
- **Advantages:**
  - Efficient handling of concurrent requests.
  - Extensive library support via npm.

#### 2. **Express.js:**

- **Purpose:** This web application framework simplified API routing and middleware configuration.
- **Advantages:**
  - Lightweight and flexible framework.
  - Support for robust routing and session management.

### 3.4.3 Database and Data Storage

The system uses **MongoDB** for managing structured and semi-structured data.

#### 1. **MongoDB:**

- **Purpose:** Its schema-less nature allowed flexibility in handling accident data, images, videos, and user details.
- **Advantages:**
  - High scalability for handling a large number of reports.
  - Efficient geospatial queries for location-based services.

### 3.4.4 Machine Learning Framework

To detect accidents in real-time, the system integrates a machine learning (ML) model.

#### 1. **TensorFlow:**

- **Purpose:** TensorFlow powered the accident detection model.
- **Advantages:**
  - Supports deep learning models for video frame analysis.
  - Optimized for both GPU and CPU execution.

#### 2. **OpenCV:**

- **Purpose:** OpenCV was used for image and video processing tasks.
- **Advantages:**
  - Efficient detection of accident features from video feeds.

- Real-time processing capabilities.

### 3.4.5 Middleware

The middleware ensures secure and seamless interaction between the client and server.

#### 1. JSON Web Token (JWT):

- **Purpose:** Used for authentication and authorization.
- **Advantages:**
  - Stateless authentication.
  - Secure token exchange between the client and server.

### 3.4.6 Other Tools and Platforms

#### 1. Postman:

- **Purpose:** Used for testing API endpoints during development.
- **Advantages:**
  - Simplified debugging of API requests and responses.

#### 2. Docker:

- **Purpose:** Used to containerize the application for consistent deployment across environments.
- **Advantages:**
  - Eliminates environment-specific issues.

#### 3. GitHub:

- **Purpose:** Version control and collaborative development.
- **Advantages:**
  - Facilitated code reviews and branch management.

### 3.4.7 Maps and Geolocation Services

#### 1. Google Maps API:

- **Purpose:** Used for visualizing accident locations on interactive maps.
- **Advantages:**
  - Provides geospatial data with high accuracy.
  - Easy integration into web applications.

#### 2. OpenWeatherMap API:

- **Purpose:** Provides real-time weather data for accident environments.
- **Advantages:**
  - Insights into weather conditions contributing to accidents.

## 3.5 Security Measures

In this section, we discuss the security measures implemented in the Smart Accident Detection & Reporting System to ensure data protection, secure communication, and safe usage. Security is a critical aspect of the system, considering the sensitive nature of the data it handles, including user information, location data, and live video feeds.

### 3.5.1 Authentication and Authorization

### 1. **Role-Based Access Control (RBAC):**

- **Purpose:** Different roles, such as admin, police, ambulance, fire brigade, and public users, have varying levels of access to the system.
- **Implementation:**
  - Admins have full control over user management and reports.
  - Police and emergency services have access to live feeds and accident details.
  - Public users can only report accidents and view limited data.
- **Security Advantage:** Prevents unauthorized access to sensitive features.

### 2. **JSON Web Token (JWT):**

- **Purpose:** Ensures secure authentication and session management.
- **Implementation:**
  - Tokens are issued upon login and validated for every request.
  - Tokens are signed using secret keys or public/private key pairs.
- **Security Advantage:** Stateless and tamper-proof authentication mechanism.

## 3.5.2 Data Protection

### 1. **Encryption:**

- **At Rest:**
  - User data and accident reports are encrypted in the database using MongoDB's built-in encryption features.

- **In Transit:**

- Secure communication is ensured using HTTPS with SSL/TLS protocols.

- 2. **Data Anonymization:**

- **Purpose:** Protect user identity in public accident reports.
- **Implementation:** Personal identifiers are removed or obfuscated when data is shared publicly.

- 3. **Regular Backups:**

- **Purpose:** Prevent data loss due to system failure or attacks.
- **Implementation:** Automatic backups of the database are scheduled to secure cloud storage.

### 3.5.3 Network Security

- 1. **Firewall Rules:**

- **Purpose:** Blocks unauthorized traffic to the server.
- **Implementation:** Configured firewall settings to allow only necessary IPs and ports.

- 2. **API Security:**

- **Purpose:** Prevent unauthorized access and data leakage through APIs.
- **Implementation:**
  - Input validation and sanitization.
  - Rate limiting to prevent abuse of endpoints.

- Use of API keys for external integrations like Google Maps.

### 3.5.4 Machine Learning Model Security

#### 1. Adversarial Attack Protection:

- **Purpose:** Prevent manipulation of video or image data to deceive the ML model.
- **Implementation:**
  - Use of robust ML algorithms resistant to adversarial inputs.
  - Regular retraining of models with updated datasets.

#### 2. Model Integrity:

- **Purpose:** Protect against unauthorized modification of ML models.
- **Implementation:**
  - Model files are stored in secure locations with restricted access.
  - Checksums and hash validation are used to ensure integrity.

### 3.5.5 Vulnerability Management

#### 1. Regular Penetration Testing:

- **Purpose:** Identify and fix security loopholes.
- **Implementation:** Conducted testing on both frontend and backend components using tools like OWASP ZAP.

#### 2. Bug Bounty Program:



- **Purpose:** Encourage ethical hackers to identify vulnerabilities.
- **Implementation:** Publicly hosted bug bounty program with rewards for valid findings.

### **3.5.6 User Security Awareness**

#### **1. Secure Passwords:**

- Enforce strong password policies during registration.
- Require periodic password changes for sensitive accounts.

#### **2. Two-Factor Authentication (2FA):**

- Provides an additional layer of security for users.

#### **3. Session Management:**

- Auto-logout after periods of inactivity to reduce unauthorized access risks.

### **3.5.7 Incident Response**

#### **1. Monitoring and Alerts:**

- Real-time monitoring of system activity for suspicious behavior.
- Alerts sent to administrators upon detecting anomalies.

#### **2. Incident Recovery Plan:**

- A documented plan exists to handle security breaches effectively.
- Includes steps for data restoration, system recovery, and post-incident analysis.

# PERFORMANCE EVALUATION

The Smart Accident Detection & Reporting System integrates machine learning, IoT, web technologies, and cloud services, designed to detect, report, and manage road accidents efficiently. This section details the performance evaluation criteria, testing procedures, and results across different aspects of the system, focusing on factors such as detection accuracy, latency, user experience, database performance, and system scalability. Each component's evaluation is divided into specific performance metrics, sub-topics, and test cases to provide a comprehensive assessment.

## 4.1 System Performance Metrics

This section evaluates the core performance metrics critical to the reliability, accuracy, and user satisfaction of the Smart Accident Detection & Reporting System. The metrics are designed to measure system efficiency, accuracy of accident detection, and overall user experience across different operational scenarios.

### 4.1.1 System Performance Metrics Overview

The key metrics assessed in this evaluation include:

**1. Detection Accuracy:**

- Measures how effectively the ML model detects true accident incidents.
- Target accuracy: above 75% with minimal false positives or negatives.

**2. Response Time:**

- Evaluates the speed at which an accident detection is processed, reported, and alerts are dispatched to emergency services.

**3. Latency:**

- Network latency between IoT cameras, the cloud, and the user interface, aimed at less than 1 second for critical real-time operations.

#### 4. **System Uptime and Reliability:**

- Ensures continuous operation without disruptions or downtime, especially under high traffic conditions.

#### 5. **Scalability:**

- Tests the ability to handle increasing numbers of users, reports, and live streams without performance degradation.

#### 6. **User Experience (UX):**

- Measures ease of navigation, response times for user actions, and the clarity of information displayed on the portal.

### **Test Cases for System Performance Metrics**

The following table outlines test cases designed to measure these metrics, with detailed steps, expected outcomes, and results for each test.

<b>Test Case ID</b>	<b>Test Case Description</b>	<b>Metric</b>	<b>Steps</b>	<b>Expected Outcome</b>	<b>Result</b>
<b>TC-01</b>	Test ML Model Detection Accuracy	Detection Accuracy	Input diverse accident video feeds. Record detected true positives and false positives.	Accuracy above 75%; minimal false positives/negatives.	Pass/Fail

<b>TC-02</b>	Measure End-to-End Response Time	Response Time	Simulate accident detection, log detection to alert notification time.	Alerts reach emergency services within 1-2 seconds post-accident detection.	Pass/Fail
<b>TC-03</b>	Network Latency Testing	Latency	Test data transmission from IoT camera to cloud and UI rendering.	Data reaches within 1 second latency to allow real-time responses.	Pass/Fail
<b>TC-06</b>	UI Response Time	UX	Navigate through pages (e.g., dashboard, reports) and measure load times.	Pages load within 3 seconds, smooth and responsive UI interactions.	Pass/Fail
<b>TC-07</b>	Accuracy of Real-time Updates	UX	Update accident status and verify UI reflects changes in under 1 second.	UI updates accurately and quickly without reload; changes are near-instant.	Pass/Fail

Table 4.1 Test Cases for System Performance Metrics

## 4.2 Machine Learning Model Evaluation

This section focuses on evaluating the performance of the machine learning (ML) model used for accident detection. The primary objectives include assessing the model's detection accuracy, precision, recall, and latency, all of which are critical to ensuring prompt and reliable responses to accident incidents. Additionally, this evaluation examines how well the model performs under varying conditions, such as different lighting, weather, and traffic scenarios.

#### 4.2.1 Key Metrics for ML Model Performance Evaluation

The following metrics are considered essential for evaluating the accident detection model:

**1. Detection Accuracy:**

- Measures the proportion of true accident detections relative to total detections.
- Target accuracy:  $\geq 75\%$  to minimize both false positives and false negatives.

**2. Precision:**

- Measures the percentage of correctly identified accidents out of all predicted accidents, minimizing false positives.
- Ensures that false alarms are minimized, keeping emergency services responsive and efficient.

**3. Recall (Sensitivity):**

- Measures the percentage of actual accidents correctly identified by the model, emphasizing sensitivity in accident detection.
- High recall is critical to ensure that no accident is missed by the system.

**4. Model Latency:**

- Evaluates the time taken by the model to process video feeds and detect accidents.
- Target latency:  $< 500$  ms for real-time detection, ensuring prompt emergency response.

**5. Robustness:**

- Assesses the model's performance under diverse environmental conditions, such as lighting, weather, and traffic volume.

### **4.3 User Interface and Portal Performance**

Evaluating the User Interface (UI) and overall portal performance is critical for ensuring a seamless and responsive experience for all users, including administrators, police, emergency services, and the public. This evaluation considers load times, page responsiveness, usability across different devices, and error handling. Key metrics for assessing portal performance include page load time, responsiveness, compatibility, and error rate, which directly influence user satisfaction and efficiency in emergency response.

#### **Key Metrics for UI and Portal Performance Evaluation**

The following metrics are used to evaluate the performance of the portal's user interface:

**1. Page Load Time:**

- Measures the time taken to fully load each page, especially critical pages like the Dashboard, Reports, and Live Video Feed.
- Target load time:  $\leq 3$  seconds for most pages to ensure minimal delay for users accessing vital data.

**2. UI Responsiveness:**

- Ensures that elements (buttons, forms, and tables) respond swiftly to user interactions.
- Responsiveness is essential for actions like updating reports, navigating tabs, and loading new content.

**3. Compatibility and Adaptability:**

- Ensures the portal is compatible across various browsers and devices (desktop, mobile, tablet).
- Responsive design is critical for maintaining functionality on different screen sizes.

#### 4. Error Rate and Reliability:

- Tracks the occurrence of errors or crashes within the portal, aiming for minimal disruption during usage.
- Error rate: Target < 1% to maintain smooth operation and reliable user interactions.

#### 5. Scalability:

- Assesses the portal's capability to handle concurrent users, particularly under peak load conditions.
- Target: 100+ concurrent users without performance degradation.

### Test Cases for UI and Portal Performance

The following table provides test cases for evaluating the user interface and portal performance. These tests are essential for identifying areas where the UI can be optimized for speed, efficiency, and user satisfaction.

Test Case ID	Test Case Description	Metric	Steps	Expected Outcome	Result
TC-UI-01	Page Load Speed Test (Dashboard)	Page Load Time	Access the Dashboard page and measure load time.	Dashboard loads within 3 seconds.	Pass/Fail
TC-UI-02	Page Load Speed Test (Reports)	Page Load Time	Access the Reports page and measure load time.	Reports page loads within 3 seconds.	Pass/Fail

<b>TC-UI-03</b>	Button and Form Responsiveness	UI Responsiveness	Interact with buttons and forms (e.g., report updates, user actions) and measure response time.	Actions respond within 1 second of interaction.	Pass/Fail
<b>TC-UI-04</b>	Device Compatibility Test (Mobile)	Compatibility	Access the portal from mobile devices; check for layout and usability.	Portal adapts to mobile screens without layout issues.	Pass/Fail
<b>TC-UI-05</b>	Browser Compatibility Test	Compatibility	Access the portal on Chrome, Firefox, Edge, Safari, etc., and verify functionality.	Portal operates smoothly on all supported browsers.	Pass/Fail
<b>TC-UI-06</b>	Error Rate Tracking During Navigation	Error Rate	Log any errors encountered during navigation across pages.	Error rate < 1%, ensuring stable and reliable usage.	Pass/Fail
<b>TC-UI-09</b>	Report Submission Test (Manual Reporting)	UI Responsiveness	Submit a manual accident report and measure processing time and error handling.	Report submission is processed instantly without errors.	Pass/Fail
<b>TC-UI-10</b>	Real-time Data Updates on Dashboard	Real-time Responsiveness	Verify real-time updates (e.g., new reports, updates in status) on Dashboard.	Dashboard reflects real-time updates with minimal delay.	Pass/Fail



<b>TC-UI-11</b>	Emergency Services Accessibility Test	Accessibility	Verify ease of access and navigation for emergency services roles within the portal.	Emergency services can access all necessary features promptly and easily.	Pass/Fail
-----------------	---------------------------------------	---------------	--	---	-----------

Table 4.2 Test Cases for UI and Portal Performance

## CONCLUSION

The Smart Accident Detection & Reporting System project represents a transformative approach to enhancing road safety and emergency responsiveness. Through its integration of IoT, machine learning, and a user-friendly web interface, the system addresses the crucial need for real-time accident detection, reporting, and response management. By leveraging street cameras and automated detection models, the platform offers accurate identification of accident types, severity assessment, and vehicle tracking, while allowing manual reporting for public users. This combination ensures a comprehensive solution that can adapt to various accident scenarios.

The project's multi-faceted design allows police, ambulance, and fire brigade services to view live accident footage, assess incident details, and respond effectively. The structured dashboards and detailed reports contribute to better monitoring and analysis of accident trends, while the user-specific role functionality enhances security and data privacy. The portal's performance, scalability, and user interface responsiveness underscore its practicality in handling real-world demands, making it viable for broader implementation.

### Key Takeaways:

- **Enhanced Response:** The system facilitates faster response times by enabling live tracking and immediate alerts to relevant authorities, potentially reducing fatalities and injury severity.
- **Data-Driven Insights:** By capturing and analysing data from accidents, the platform contributes to identifying accident-prone zones and improving preventive measures.
- **User Accessibility:** Role-based access ensures that public users, authorities, and emergency services can interact with the system according to their specific needs, making it a versatile tool for diverse user groups.

This project not only demonstrates the capabilities of modern technology in addressing public safety challenges but also provides a scalable model for accident management in urban and rural settings. Future developments may focus on enhancing the ML model for accident prediction and

extending the system to integrate with additional data sources, furthering its potential as a critical component in smart city initiatives.

## REFERENCES

- [1] OpenWeatherMap API Documentation. (2024). *Weather Data for IoT and ML Integration in Real-Time Monitoring Systems*. Retrieved from <https://openweathermap.org/api>
- [2] MongoDB Documentation. (2024). *Data Modeling and Schema Design for MongoDB Applications*. MongoDB, Inc. <https://www.mongodb.com/docs/manual/>
- [3] React Router Documentation. (2024). *Implementing Protected Routes in Web Applications*. Retrieved from <https://reactrouter.com/>
- [4] Stack Overflow. (2024). *Various threads on Protected Routes and State Management in React*. Retrieved from <https://stackoverflow.com/>
- [5] ChatGPT by OpenAI. (2024). *Guidance on System Architecture, API Integration, and Project Implementation Strategies*. Retrieved from <https://openai.com/chatgpt>
- [6] MDN Web Docs. (2024). *JavaScript and Web Development Techniques*. Retrieved from <https://developer.mozilla.org/>
- [7] W3Schools. (2024). *Tutorials for Web Development, React, and APIs*. Retrieved from <https://www.w3schools.com/>
- [8] GeeksforGeeks. (2024). *Detailed Guides on Accident Detection Systems and MongoDB*. Retrieved from <https://www.geeksforgeeks.org/>
- [9] GitHub. (2024). *Repositories and Code Samples for Accident Detection Systems*. Retrieved from <https://github.com/>
- [10] FreeCodeCamp. (2024). *Web Development and Backend Integration Tutorials*. Retrieved from <https://www.freecodecamp.org/>

## **ACKNOWLEDGEMENT**

We would like to place on record our deep sense of gratitude to Dr. S.C. Nandedkar, HOD-Dept. of Computer Science and Engineering, Deogiri Institute of Engineering and management Studies Chatrapati Sambhajanagar, for her generous guidance, help and useful suggestions.

We express our sincere gratitude to Prof. Punam Borase, Dept. of Computer Science and Engineering, Deogiri Institute of Engineering and management Studies Chatrapati Sambhajanagar, for her stimulating guidance, continuous encouragement and supervision throughout the course of present work.

We are extremely thankful to Dr. Ulhas Shiurkar, Director and Dr. S V. Lahane, Dean Academics Deogiri Institute of Engineering, and management Studies Chatrapati Sambhajanagar, for providing us infrastructural facilities to work in, without which this work would not have been possible.

### **Signatures of Students**

Sarode Siddhant Suraj (CS4261)

Patil Sahil Sanjay (CS4262)