

# **Real-time Road Accident reporting and Complaint registering website**

**A PROJECT REPORT**

*Submitted by,*

<b>Mr. Deepak V</b>	<b>-20201CSE0673</b>
<b>Mr. Jayanth B S</b>	<b>-20201CSE0680</b>
<b>Mr. Mohamed Shaiz</b>	<b>-20201CSE0684</b>
<b>Ms. Aishwarya Oji</b>	<b>-20201CSE0701</b>

*Under the guidance of,*

**Ms. V. Kayalvizhi**

*in partial fulfillment for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING.**

**At**



**PRESIDENCY UNIVERSITY**

**BENGALURU**

**JANUARY 2024**

# **PRESIDENCY UNIVERSITY**

## **SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

### **CERTIFICATE**

This is to certify that the Project report “**Real-time Road Accident reporting and Complaint registering website**” being submitted by “**Deepak V**”, “**Jayanth B S**”, “**Mohammed Shaiz**”, “**Aishwarya Oji**” bearing roll number(s) “**20201CSE0673**”, “**20201CSE0680**”, “**20201CSE0684**”, “**20201CSE0701**” in partial fulfillment of requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering is a bonafide work carried out under my supervision.

**Ms. V. Kayalvizhi**

Assistant Prof.  
School of CSE  
Presidency University

**Dr. Pallavi .R**

Associate Professor & HoD  
School of CSE  
Presidency University

**Dr. C. KALAIARASAN**

Associate Dean  
School of CSE&IS  
Presidency University

**Dr. L. SHAKKEERA**

Associate Dean  
School of CSE&IS  
Presidency University

**Dr. SAMEERUDDIN KHAN**

Dean  
School of CSE&IS  
Presidency University

# **PRESIDENCY UNIVERSITY**

## **SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

### **DECLARATION**

We hereby declare that the work, which is being presented in the project report entitled *“Real-time Road Accident reporting and Complaint registering website”* in partial fulfilment for the award of Degree of **“Bachelor of Technology in Computer Science and Engineering”**, is a record of our own investigations carried under the guidance of **Ms.V.Kayalvizhi, Assistant Professor, School of Computer Science and Engineering, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

<b>NAME</b>	<b>ROLL NO</b>	<b>SIGNATURE</b>
Deepak V	20201CSE0673	
Jayanth B S	20201CSE0680	
Mohammed Shaiz	20201CSE0684	
Aishwarya Oji	20201CSE0701	

## ABSTRACT

This innovative project represents a paradigm shift in road safety, introducing a pioneering web application for intelligent **Real-time Road Accident reporting and Complaint registering website**. With a focus on user-centric design and advanced technologies, this initiative aims to re-define the landscape of road safety in an ever-evolving transportation environment. The application serves as a beacon of innovation, seamlessly integrating advanced functionalities to empower individuals, law enforcement.

Beyond traditional reporting methods, the application envisions a future where swift and intelligent responses to road incidents become the norm. The user-friendly interface allows law enforcement personnel to effortlessly report accidents, while the intelligent License Plate Recognition (LPR) system ensures precise information extraction, triggering real-time alerts to the nearest police and ambulance services.

This project goes beyond mere reporting by introducing a collaborative paradigm where law enforcement actively participates in accident confirmation and reporting. A dedicated police portal facilitates comprehensive reporting, contributing valuable insights to a centralized incident database. Automated notifications, driven by real-time updates and communication channels, bridge the gap between civilians, law enforcement, insurance companies, and nominated contacts.

In a world constantly in motion, this application strives to be a catalyst for change—a change that prioritizes safety, efficiency, and collaboration on our roads. It is not just an application but a commitment to fostering a culture of safety and collaboration, ultimately shaping a future where road incidents are met with intelligent and coordinated responses.

## ACKNOWLEDGEMENT

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Dean, School of Computer Science and Engineering & Information Science, Presidency University for getting us permission to undergo the project.

We record our heartfelt gratitude to our beloved Associate Deans **Dr. Kalaiarasan C and Dr. Shakkeera L**, School of Computer Science and Engineering & Information Science, Presidency University and **Dr. Pallavi R**, Head of the Department, School of Computer Science and Engineering, Presidency University for rendering timely help for the successful completion of this project.

We are greatly indebted to our guide **Ms. V. Kayalvizhi, Assistant Professor**, School of Computer Science and Engineering, Presidency University for his inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the University Project-II Coordinators **Dr. Sanjeev P Kaulgud, Dr. Mrutyunjaya MS** and also the department Project Coordinators **Mr. Zia Ur Rahman, Mr. Peniel John Whistely**.

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

Mr. Deepak V	20201CSE0673
Mr. Jayanth B S	20201CSE0680
Mr. Mohammed Shaiz	20201CSE0684
Ms. Aishwarya Oji	20201CSE0701

## LIST OF TABLES

<b>Sl. No</b>	<b>Table Name</b>	<b>Table Caption</b>	<b>Page No.</b>
01	Table 6.3	Admin Database	21
02	Table 6.4	Accident Reports	22

# LIST OF FIGURES

<b>Sl. No.</b>	<b>Figure Name</b>	<b>Caption</b>	<b>Page No.</b>
01	Fig.6.1	System Architecture of The website	15
02	Fig.6.2	Workflow Of Website	19
03	Fig.6.3	Use Case Diagram	23
04	Fig.6.4	Sequence Diagram	25
05	Fig.6.5	Class Diagram	26
06	Fig.7.1	Gantt Chart	28

# **TABLE OF CONTENTS**

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
	<b>ABSTRACT</b>	<b>iv</b>
	<b>ACKNOWLEDGMENT</b>	<b>v</b>
	<b>LIST OF TABLES</b>	<b>vi</b>
	<b>LIST OF FIGURES</b>	<b>vii</b>
<b>1.</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Road Safety as a Top Priority	
	1.1.1 The Need for a Responsive Accident Reporting System	
	1.2 Redefining the Landscape of Road Safety	
	1.3 Empowering Individuals and Law Enforcement	
	1.1 Road Safety as a Top Priority	
	1.1.1 The Need for a Responsive Accident Reporting System	
	1.2 Redefining the Landscape of Road Safety	
	1.3 Empowering Individuals and Law Enforcement	
<b>2.</b>	<b>LITERATURE REVIEW</b>	<b>3</b>
<b>3.</b>	<b>RESEARCH GAPS OF EXISTING METHOD</b>	<b>7</b>
	3.1 Limited Accuracy in Accident Detection	
	3.2 Insufficient Detail in Reporting Systems	
	3.3 Lack of Immediate Assistance in Accident Reporting	
	3.4 Inefficiency in Accident Reporting Processes	



<b>4.</b>	<b>PROPOSED METHODOLOGY</b>	<b>8</b>
	4.1 Overview:	
	4.2 Requirement Analysis	
	4.3 Technology Stack Selection	
	4.4 Tools and Technologies Used	
	4.1 Tesseract OCR:	
	4.2 NumPy	
	4.3 COCO API	
	4.4 TensorFlow	
	4.4.1 MediaPipe	
	4.5 System Architecture	
	4.5.1 Frontend Development using React.js	
	4.5.2 Backend Development with Python	
	4.6. Accident Reporting Workflow	
	4.6.1 OCR Integration for Image Processing	
	4.7 Firebase Database Interaction	
	4.8. Potential Integration of LPR	
	4.8.1 License Plate Recognition (LPR)	
<b>5.</b>	<b>OBJECTIVES</b>	<b>14</b>
	5.1 Facilitate Efficient and Swift Accident Reporting	
	5.2 Implement Robust License Plate Recognition (LPR)	
	5.3 Enable Collaboration with Law Enforcement	
	5.4 Ensure Secure User Authentication and Data Protection	
	5.5 Design a User-friendly Interface	
<b>6.</b>	<b>SYSTEM DESIGN &amp; IMPLEMENTATION</b>	<b>15</b>
	6.1 System Architecture	
	6.2 Optical Character Recognition (OCR)	
	6.3 Firebase Database Structure:	
	6.4 Use Case Diagram	
	6.5 Sequence Diagram	
	6.6 Class Diagram	
<b>7.</b>	<b>TIMELINE FOR EXECUTION OF PROJECT</b>	<b>28</b>
<b>8.</b>	<b>OUTCOMES</b>	<b>30</b>

<b>9.</b>	<b>RESULTS AND DICUSSIONS</b>	<b>32</b>
	9.1 Advantages	
	9.2 Disadvantages	
	9.3 Future Scope	
<b>10.</b>	<b>CONCLUSION</b>	<b>35</b>
	<b>REFERENCES</b>	<b>37</b>
	<b>APPENDIX-A PSUEDOCODE</b>	<b>38</b>
	<b>APPENDIX-B SCREENSHOTS</b>	<b>49</b>
	<b>APPENDIX-C ENCLOSURES</b>	<b>53</b>

---

# **CHAPTER-1**

## **INTRODUCTION**

### **1.1 Road Safety as a Top Priority**

Road safety has become our paramount concern amid the continuous advancements in technology and the automotive industry. In this context, our project serves as an initiative poised to revolutionize the road safety landscape. In a world where transportation dynamics are ever-evolving, the demand for a responsive and efficient accident reporting system has never been more critical.

#### **1.1.1 The Need for a Responsive Accident Reporting System**

This project aims to address the pressing need for a comprehensive and efficient accident reporting system. With advancements in technology and the evolution of transportation, a system that transcends traditional reporting methods is imperative. Our application stands as a beacon of innovation, seamlessly integrating advanced technologies and adopting a user-centric approach to establish a pioneering web application for intelligent accident reporting and emergency response.

### **1.2 Redefining the Landscape of Road Safety**

Navigating the intricacies of road safety requires a solution that goes beyond traditional reporting methods. This application positions itself at the forefront of this transformation, envisioning a future where road incidents are met with swift and intelligent responses. It is not merely an application but a commitment to fostering a culture of safety and collaboration on our roads.

---

### **1.3 Empowering Individuals and Law Enforcement**

At its core, this application is designed to empower individuals, law enforcement, and emergency services through a seamless integration of advanced functionalities. Users, whether civilians or law enforcement personnel, can effortlessly report accidents through an intuitive interface. The application's intelligent Optical Character Recognition (OCR) system ensures the precise extraction of crucial information, initiating real-time alerts to the nearest police and ambulance services.

The project goes beyond mere reporting; it introduces a collaborative paradigm where law enforcement plays an active role in accident confirmation and reporting. A dedicated police portal allows for comprehensive reporting, contributing valuable insights to the centralized incident database. Automated notifications, fueled by real-time updates and communication channels, bridge the gap between civilians, law enforcement, insurance companies, and nominated contacts.

In a world constantly on the move, this application strives to be the catalyst for change—a change that prioritizes safety, efficiency, and collaboration.

---

## **CHAPTER-2**

### **LITERATURE SURVEY**

#### **Summary of papers:**

1. A Survey on Smartphone Based Accident Reporting and Guidance Systems
    - a. Authors: Alexandra Fanca<sup>1</sup>, Adela Puscasiu, Honoriu Valean, Silviu, Folea
    - b. Year Published: 2018
    - c. Proposed Idea: As shown in this paper, the smartphones can detect the occurrence of an accident, but the main problems remain the high rate of false detection. Different from other systems, our system will learn the user's daily behavior. For this we can use different machine learning algorithms such as neural networks, support vector machine (SVM) or decision tree. In the beginning, the system will work in the learning phase and after that it can be used being able to minimize false detection.
  
  2. A Modeling of Traffic Accident Reporting System through UML Using GIS
    - a. Authors: Dr. Gufran Ahmad Ansari, Dr. M. Alshabi
    - b. Year Published: 2012
    - c. Proposed Idea: From the above study of work, it is accomplished that the UML is a powerful modeling language to solve scientific and research problems. In this paper a complete modeling of Traffic Accident Reporting System has been done through the UML and its results are shown in the form of bar chart graph. This model is a simple and has a reusability property also model can easily enhance, modify and updated according to the need of data. This basic work can be expended in the field of data mining using UML and expert system.
  
  3. Accident Reporting System using an iOS Application
    - a. Authors: Mohamad Fahmi Bin Hussin, Mohamad Huzaimy Jusoh
    - b. Year Published: 2014
    - c. Proposed Idea: In this paper, the design and the simulation of the application have been successfully presented. This application does not cover all the detail of accident reporting as in standard paper form. However, this application consists of important detail, which is sufficient for summary of
-

accident reporting. Besides, this application offers a lot of benefits compared to the manual reporting by using paper form. The benefits include fewer delays, report submission to various departments simultaneously, easier means of reporting and can easily be emailed. Therefore, with the development of the application, it significantly improves the timeliness of accident reporting as it encourages prompt reporting and investigation for quick action. This application is important and useful for occupational safety and health sector.

**4. Accident Reporting and Guidance System with automatic detection of the accident**

a. Authors: Alexandra Fanca, Adela Pușcașiu, Honoriu Vălean

b. Year Published: 2016

c. Proposed Idea: The main purpose of the system is to find the nearest medical unit from the accident location. This was achieved by using some features that the GPS receiver and Google Maps are providing. The server application together with the mobile one and the database compose the strongly connected client server applications.

**5. Android Based Real-Time Road Accident Reporting Application**

a. Authors: Carlos A. Villanueva, Thelma D. Palaoag

b. Year Published: 2022

c. Proposed Idea: This is beneficial to both car/motorcycle owners, civilians and police personnel. This suggests that the application might be able to save lives and lessen serious and lifetime injuries in the event of an accident, even if it occurred in a distant location. The mobile application was created using Java programming language and Android Studio.

**6. Accident Detection and Reporting System using Internet of Things**

a. Authors: Marimuthu Muthuvel

b. Year Published: 2018

c. Proposed Idea: This system aims at providing early detection of accidents and communicating the information immediately to the emergency responses on time to provide quick assistance for the injured person. When the rider met with an accident and the helmet hits the ground, the vibration sensor which is embedded in the helmet senses the vibration frequency and transfers the

value to the raspberry pi module that is interfaced to it. When the vibration threshold frequency exceeds the programmed maximum limit, the raspberry pi board extracts GPS data from the GPS module and the message with all the necessary information is sent quickly to the registered emergency contacts of the rider. This system assures to provide immediate assistance to the victim of the accident. The results give exact locations of the accident.

## **7. Accident Detection and Alert System**

a. Authors: Dr. C. K. Gomathy, K Rohan, Bandi Mani Kiran Reddy, Dr. V Geetha

b. Year Published: 2022

c. Proposed Idea: The main idea of this paper is to build an application that makes use of the sensors present in mobile phones like GPS and Accelerometer and detect any collision if there is a sudden external disturbance in the speed with the help of the Sensor Fusion Based Algorithm. With the help of the data obtained from the Accelerometer sensor, when there is a sudden disturbance to the mobile phone, the user is notified with an alert message before sending the request help signal. If no emergency is required, they can cancel it within 10 seconds. But, if they press the "Call Help" button or if the alert message is unattended for more than 10 seconds, the "request for help" message will be sent to the emergency services as well as the family members, the users provided.

## **8. Accident Detection and Smart Rescue System**

a. Authors: Swapnil A. Patil, Kritika V. Honde, Sonali D. Leakage, Prof. Pravin M. Tambe

b. Year Published: 2022

c. Proposed Idea: The main idea of this paper features an appealed notification system. Users register to access a private dashboard for inputting and viewing car details, streamlining emergency response. In case of a crash, impact sensors in vehicles trigger signals to a central IoT device at a police station or hospital. Via GSM, GPS coordinates swiftly dispatch the nearest ambulance, aided by Arduino controllers and GSM modems. Additional features include vibration triggered collision sensors and user activated switches for prioritized responses.

### **Advantages:**

- Learning phase helps improve accuracy over time.
- Can be personalized to individual users.
- Simple model with reusability properties.
- Easily modifiable and updatable according to data needs
- Design and simulation of the application are successfully presented.
- Offers benefits like fewer delays, simultaneous submission to various departments, and easy emailing.
- Improves timeliness of accident reporting.
- Locates the nearest medical unit using GPS and Google Maps.
- Strongly connected client-server applications.

### **Disadvantages:**

- Initially, high rate of false detection.
- Requires continuous learning phase for optimal performance.
- Specific focus on modelling, may lack real-time reporting capabilities.
- Does not cover all details of accident reporting as in standard paper forms
- Specific focus on guiding to medical units, may lack some reporting features.
- Relies on embedded sensors in helmets, which may not cover all accident scenarios.
- Relies on user response for cancellation, may lead to false alerts.

In summary, the ideas and methodologies presented in these papers, ranging from machine learning algorithms to sensor-based collision detection, provide a rich source of inspiration and validation for the development of our accident reporting system. The ideas we incorporated in our project is to focus on user behavior learning, efficient reporting mechanisms, simultaneous submission to various departments, alerting emergency services and navigating them to accident spot. To make the reporting efficient License Plate recognition is used to fetch civilians' details from database based on the number plate determined. All of collectively contribute to shaping innovative features of our project.



---

## CHAPTER-3

### RESEARCH GAPS OF EXISTING METHODS

In the existing methods, emphasizes IoT based to detect accident but there is lack of alert system and data of the accident by extensive literature survey License Plate Recognition (LPR) and Optical Character Recognition (OCR) is not incorporated, which helps reduce manual work of entering data of the vehicle involved in accident a gap our project fills by implementing:

#### **1. Limited Accuracy in Accident Detection:**

Existing systems, especially smartphone-based ones, face challenges in accurately detecting accidents, leading to a high rate of false positives. Our project aims to enhance accuracy by implementing OCR to extract information from accident scenes, such as license plates or text on road signs. Additionally, LPR can be employed to recognize license plates for better accident documentation.

#### **2. Insufficient Detail in Reporting Systems:**

Some systems may lack comprehensive detail in accident reporting. By incorporating OCR, our project addresses this gap by automatically extracting relevant details from accident scenes, reducing the reliance on manual input and ensuring more comprehensive reporting.

#### **3. Lack of Immediate Assistance in Accident Reporting:**

While several systems focus on accident reporting, there is a gap in providing immediate assistance to the victims. Our project, by integrating OCR and LPR, aims to not only detect accidents in real-time but also automatically extract crucial information for quicker and more effective emergency responses.

#### **5. Inefficiency in Accident Reporting Processes:**

Current reporting systems, especially manual ones, may suffer from inefficiencies. Our project, with OCR and potentially LPR, streamlines the reporting process by automating data extraction, reducing delays, and improving overall efficiency in accident reporting within occupational safety and health sectors. By considering OCR and LPR features, our project extends its capabilities to automatically process visual information, enhancing the overall efficiency and effectiveness of accident reporting and guidance systems.

---

## **CHAPTER-4**

### **PROPOSED METHODOLOGY**

#### **4.1 Overview:**

The proposed methodology is meticulously designed to orchestrate the development of an advanced accident reporting and guidance system. A synergy of React.js for frontend, Python for backend, and a sophisticated OCR model This methodology aims to deliver not only a user-friendly interface but also robust backend functionalities, enabling seamless accident reporting and efficient data management.

#### **4.2 Requirement Analysis:**

Before delving into implementation, a detailed requirement analysis was conducted to understand the needs of the system. Key stakeholders, including administrators and civilians, are identified, and their specific requirements are documented. Functionalities such as user authentication, dynamic dashboard creation, and efficient accident reporting are outlined, providing a solid foundation for the implementation process.

#### **4.3 Technology Stack Selection**

The technology stack was carefully chosen to align with the project's goals. React.js was selected for frontend development due to its modular architecture and efficient rendering capabilities. Python was chosen for backend development, leveraging its readability and extensive library support. The integration of a trained OCR model, based on CNN and R-CNN, was essential for accurate character extraction from license plates

## **4.4 Tools and Technologies Used:**

### **1. Hardware:**

- Operating System: Windows 11
- RAM: 16GB

### **2. Software:**

- Text Editors or IDEs for Coding: Visual Studio Code
- Database Management Tools: Firebase Console
- Version Control Systems: Git, GitHub
- OCR Libraries or Tools for Text Extraction: Tesseract

### **3. Technological Stack:**

- Frontend Development: React.js
- Backend Development: Python
- Database Management and Authentication: Firebase

### **4. Libraries Used:**

- **React.js:**

React.js is a JavaScript library for building user interfaces. It facilitates the creation of interactive and dynamic frontend components, providing a robust foundation for building a responsive user interface.

- **Firebase SDK:**

Firebase SDK is a comprehensive set of tools and libraries provided by Firebase for building scalable and feature-rich applications. It includes authentication services, real-time database management, and cloud-based storage, contributing to the seamless integration of backend functionalities.

- **OCR Libraries and Tools:**

#### **4.4.1 Tesseract OCR:**

- Tesseract is an open-source OCR engine that enables accurate character recognition from images.

- It plays a pivotal role in the extraction of alphanumeric details from license plates, contributing to the efficiency of the accident reporting system.

#### **4.4.2 NumPy:**

- NumPy is a library for numerical operations in Python. It is employed in the OCR model's training process, providing efficient data manipulation capabilities.

#### **4.4.3 COCO API:**

- The COCO (Common Objects in Context) API is utilized for working with image datasets. It assists in preparing and organizing data for training the OCR model.

#### **4.4.4 TensorFlow:**

- TensorFlow is an open-source machine learning framework. It is employed in the training of the OCR model, contributing to the model's ability to recognize patterns in license plate images.

#### **4.4.5 MediaPipe:**

MediaPipe offers a library for building perception pipelines. It is used for image processing and handling within the OCR model, enhancing the accuracy of character extraction.

#### **4.4.6- pip:**

- Pip is the package installer for Python, essential for managing and installing various libraries and dependencies used in the project.

### **4.5 System Architecture:**

#### **4.5.1 Frontend Development using React.js:**

The frontend development with React.js is a strategic choice to ensure an engaging and responsive user interface. Leveraging React.js, the frontend is architected to include modular components for user authentication, dynamic dashboard creation, and efficient complaint register viewing. The virtual DOM of React.js optimizes rendering, providing users with a smooth, immersive experience.

## **Component Design**

Detailed component design is undertaken to create modular and reusable elements for the frontend. Each component is strategically designed to encapsulate specific functionalities, such as user authentication, dashboard creation, and complaint register viewing. React.js's component-based architecture facilitates code modularity, making the frontend scalable and maintainable.

## **Virtual DOM Optimization**

React.js's virtual DOM is optimized to enhance rendering efficiency. The virtual DOM allows for efficient updates and rendering of UI components, ensuring a smooth and responsive user experience. State management techniques are implemented to handle dynamic data updates, ensuring that the frontend reflects real-time changes seamlessly.

### **4.5.2 Backend Development with Python:**

#### **Backend Logic Implementation**

Python is employed for backend logic implementation, covering critical functionalities such as user authentication, accident report processing, and interaction with external services. Asynchronous features of Python are utilized to handle real-time data processing, ensuring quick response times for dynamic user interactions. The backend logic is designed to act as a bridge between the frontend and the OCR model.

#### **Firebase Integration**

Firebase, a cloud-based database, is integrated into the backend for efficient data storage and retrieval. Firebase offers real-time synchronization, making it an ideal choice for handling accident reports and user data. Secure authentication mechanisms are implemented to ensure the integrity and confidentiality of user data. The data structure within Firebase is meticulously designed to accommodate accident reports and associated details.

## **4.6. Accident Reporting Workflow:**

### **4.6.1 OCR Integration for Image Processing:**

#### **OCR Model Training**

The OCR model, a pivotal component for license plate recognition, undergoes comprehensive training using a dataset of license plate images. Convolutional Neural Networks (CNN) and Region-based CNN (R-CNN) architectures are employed to train the model. The training process involves optimizing parameters to enhance accuracy in character extraction. Extensive testing is conducted to validate the model's proficiency in recognizing characters from license plates.

#### **Image Preprocessing**

Image preprocessing is implemented to enhance the OCR model's ability to extract alphanumeric details accurately. Techniques such as noise reduction, contrast enhancement, and resizing are applied to image links submitted during the accident reporting process. The goal is to isolate relevant portions containing license plates, optimizing the subsequent OCR processing

## **4.7 Firebase Database Interaction:**

#### **Alphanumeric Data Cross-Referencing**

Following character extraction by the OCR model, alphanumeric details (number plates) are extracted. These details are cross-referenced with Firebase to retrieve comprehensive vehicle information. The integration ensures that administrators have access to a rich dataset, allowing them to efficiently manage accident reports and associated details.

#### **User-Friendly Form Implementation**

A user-friendly form is designed and implemented for administrators to input accident details. The form serves as a crucial interface for collecting information related to accidents, ensuring a standardized and comprehensive approach to data entry. Accident details entered through the form are securely stored in Firebase, creating a reliable repository for historical data.

## **4.8. Potential Integration of LPR:**

### **4.8.1 License Plate Recognition (LPR):**

In a continuous pursuit of innovation, the methodology explores the potential integration of License Plate Recognition (LPR). LPR could serve as a complementary technology to further automate the extraction of vehicle details during accident reporting.

The investigation involves a thorough exploration of various LPR libraries and APIs to assess their compatibility with the existing system. The goal is to elevate the system's accuracy and efficiency in recognizing license plates, offering a comprehensive solution for streamlined accident documentation.

---

## **CHAPTER-5**

### **OBJECTIVES**

The problem statement of our project arises from the observation that existing methods for reporting accidents often lack efficiency, coordination, and real-time communication. This leads to delays in emergency response and a deficiency in comprehensive data for analysis. The identified problem revolves around the need for a modernized solution that leverages advanced technologies to streamline the reporting process, enhance collaboration between civilians and law enforcement, and ensure prompt and informed emergency responses. Accordingly, the objectives set for our project are as follows:

#### **5.1 Facilitate Efficient and Swift Accident Reporting:**

- Develop a system that allows users to report accidents quickly and efficiently through a user-friendly interface.

#### **5.2 Implement Robust Optical Character Recognition (OCR):**

- Integrate a robust OCR system to accurately capture license plate and extract number from the plate information for precise identification.

#### **5.3 Enable Collaboration with Law Enforcement:**

- Establish a collaborative platform for law enforcement to actively

#### **5.4 Ensure Secure User Authentication and Data Protection:**

- Prioritize the security of user authentication and implement robust data protection measures to safeguard sensitive information by allowing only law enforcement personnel to access.

#### **5.5 Design a User-friendly Interface:**

- Focus on designing an intuitive and user-friendly interface to enhance accessibility and usability for all users.

These objectives collectively aim to address the inefficiencies in the current accident reporting system and contribute to the development of a more streamlined, collaborative, and responsive solution.

---



---

## CHAPTER-6

### SYSTEM DESIGN & IMPLEMENTATION

#### 6.1 System Architecture

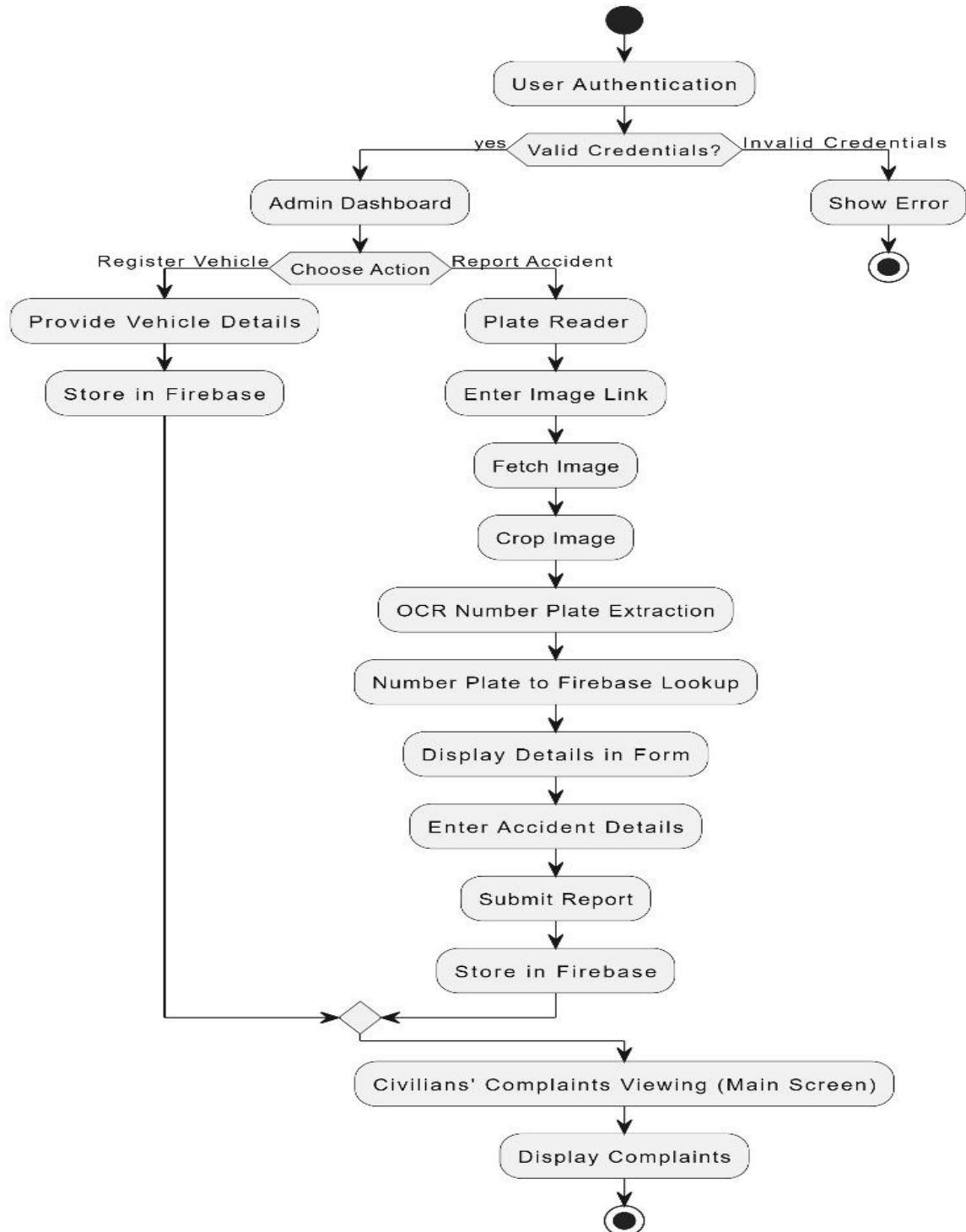


Fig.6.1 System Architecture of the website

---

The intricate system architecture of the website orchestrates a seamless workflow, incorporating user authentication, admin dashboard functionalities, OCR model processing, and civilian complaint viewing. A detailed exploration of each component, highlighting the distinctive features of your project, is presented below.

### **1. Start:**

The user journey initiates with users accessing the website, triggering a series of events that ensure a secure and efficient accident reporting system.

### **2. User Authentication:**

#### **2.1. Validation of Credentials:**

User authentication is a critical entry point. Administrators, responsible for overseeing accident reports, input their credentials, initiating a robust validation process to guarantee secure access to the system.

#### **2.2. Admin Dashboard:**

Upon successful validation, administrators gain access to the Admin Dashboard – a centralized hub that houses essential functionalities crucial for managing the accident reporting system.

### **3. Admin Dashboard:**

#### **3.1. Register New Vehicle.**

##### **3.1.1. Provide Details:**

Administrators can register a new vehicle by inputting key details such as vehicle type, registration number, and owner information. This information is integral to maintaining an up-to-date vehicle registry.

##### **3.1.2. Store in Firebase:**

The entered vehicle details are securely stored in the Firebase database. This database serves as a comprehensive repository, facilitating efficient retrieval of information when needed.

## **3.2. Report Accident:**

### **3.2.1. Plate Reader Integration:**

A distinctive feature of your system is the Plate Reader module, integrating Optical Character Recognition (OCR) capabilities. This module plays a pivotal role in extracting license plate details efficiently.

### **3.2.2. Enter Image Link:**

Administrators input the image link containing accident details, initiating the process of fetching and processing the accident image.

### **3.2.3. Fetch Image:**

The system fetches the specified image, preparing it for further OCR processing. This step is crucial for obtaining accurate and relevant information.

### **3.2.4. Crop Image:**

For precision, administrators have the option to crop the fetched image, focusing specifically on the section containing the license plate. This step streamlines the OCR extraction process.

### **3.2.5. OCR Number Plate Extraction:**

The OCR model, utilizing Convolutional Neural Networks (CNN) and Region-based Convolutional Neural Networks (R-CNN), processes the cropped image. Grayscale conversion, noise reduction, and character recognition contribute to the accurate extraction of license plate details.

### **3.2.6. Number Plate to Firebase Lookup:**

The extracted license plate details are looked up in the Firebase database, retrieving associated vehicle information. This seamless integration ensures that accident reports are linked to the correct vehicle.

### **3.2.7. Display Details in Form:**

The retrieved vehicle details are dynamically displayed in a user-friendly form within the Admin Dashboard. Administrators can efficiently review and process the information.

### **3.2.8. Enter Accident Details:**

Administrators input specific accident details into the form, including the nature of the accident, location, and any additional relevant information.

### **3.2.9. Submit Report:**

The completed accident report, along with the associated details, is securely submitted and stored in the Firebase database for future reference and analysis.

## **4. Invalid Credentials:**

### **4.1. Error Handling:**

In the event of invalid credentials during user authentication, the system employs robust error-handling mechanisms, displaying an error message to prevent unauthorized access.

## **5. Civilians' Complaints Viewing (Main Screen):**

### **5.1. Display Complaints:**

Civilians accessing the main screen can transparently view registered complaints without requiring authentication. This inclusive feature fosters community engagement and awareness.

## **6. End:**

The comprehensive workflow concludes, delivering a user-friendly and efficient system for accident reporting and complaint viewing.

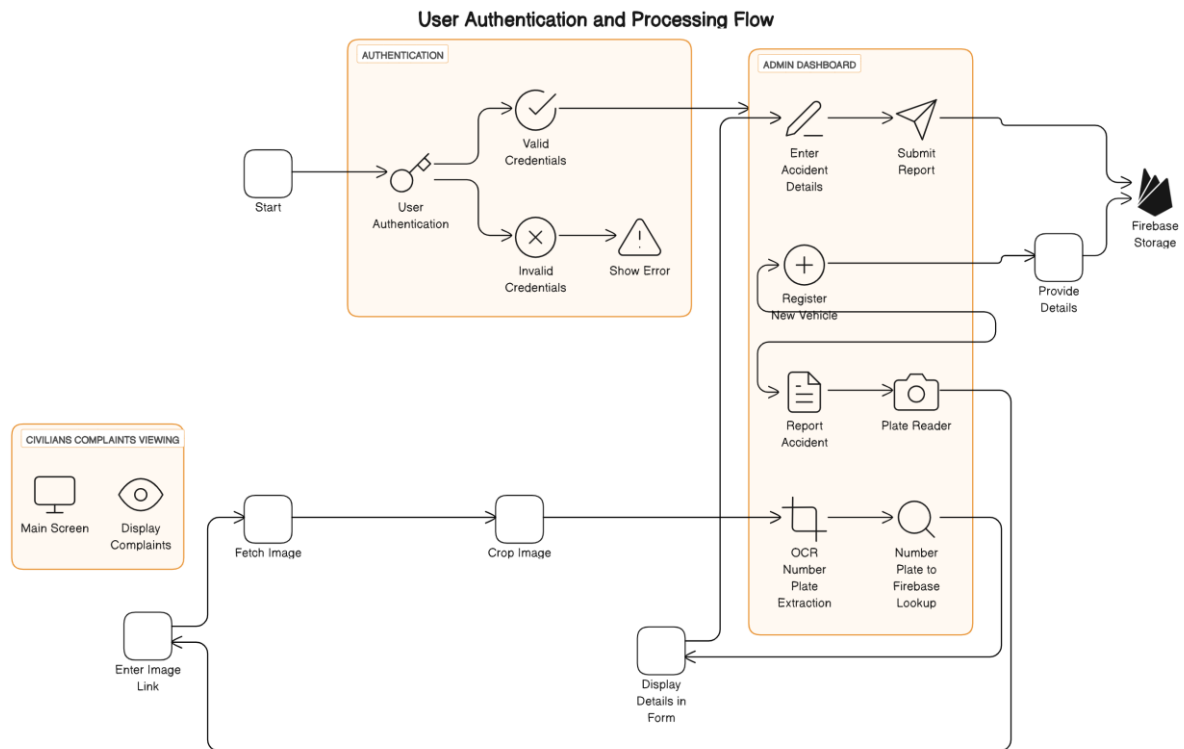


Fig.6.2 Workflow of website

## 6.2 Optical Character Recognition (OCR)

The Optical Character Recognition (OCR) model is a pivotal component of the system, tasked with extracting alphanumeric details, particularly license plate information, from images. Below is an in-depth exploration of the OCR model's workings, encompassing various modules and intricacies.

### 1. Image Input:

At the onset, the OCR model receives an image containing relevant license plate details captured from an accident scene.

### 2. Preprocessing:

#### 2.1. Image Fetching:

Administrators initiate the OCR process by providing an image, which is then fetched by the OCR model.

#### 2.2. Image Cropping:

To enhance the accuracy of OCR, administrators have the flexibility to crop the image, focusing on the specific region containing the license plate. This step ensures that the OCR algorithm processes only the relevant information.

## **2.3. Image Processing:**

### **2.3.1. Grayscale Conversion:**

The input image undergoes grayscale conversion using NumPy arrays. Grayscale simplifies the image, reducing complexity and aiding the OCR model in recognizing characters effectively.

### **2.3.2. Additional Preprocessing:**

Beyond grayscale conversion, other preprocessing steps include noise reduction, contrast enhancement, and resizing. These steps collectively contribute to optimizing the image for accurate character recognition.

## **3. OCR Extraction:**

### **3.1. OCR Engine Initialization:**

The OCR engine is initialized, employing advanced algorithms designed to recognize and extract characters from the preprocessed image.

### **3.2. Number Plate Recognition:**

With the OCR engine active, the model focuses on the region of the image containing the license plate. By leveraging its trained capabilities, the OCR model identifies individual characters, reconstructing the entire number plate.

## **4. Number Plate Lookup:**

### **4.1. Firebase Integration:**

The extracted number plate is seamlessly integrated into the Firebase database, ensuring a centralized repository for vehicle-related information.

### **4.2. Data Retrieval:**

The OCR module performs a lookup operation within the Firebase database, retrieving detailed information associated with the recognized number plate. This information typically includes vehicle owner details and registration information.

## **5. Data Presentation:**

### **5.1. Form Display:**

The OCR model plays a crucial role in presenting the retrieved vehicle details in a user-friendly form. This form is then displayed to administrators, providing them with a structured interface to process and input additional accident-related information.

## 6. Interaction with Backend:

Facilitating a smooth integration into the broader system, the OCR module interacts seamlessly with the backend. This interaction ensures that the OCR-extracted data is harmoniously incorporated into the accident reporting workflow.

## 7. Handling OCR Errors:

Recognizing the possibility of errors during OCR extraction, the system is designed with error-handling mechanisms. Administrators are promptly notified of any discrepancies, allowing for swift corrections and ensuring the accuracy and reliability of the processed data.

## 8. Continuous Learning:

The OCR model is not static; it evolves over time through continuous learning. Regular updates and retraining sessions with new datasets contribute to the refinement and improvement of the OCR model's accuracy.

### 6.3 Firebase Database Structure:

#### 6.3.1. Admins:

Firestore collection: admins

Fields:

- Admin\_Id: Unique identifier for each admin.
- Username: Admin's username.
- Password: Admin's password (consider using secure methods for password storage).

VehicleNumber	Name	PhoneNumber
ABC123	John Doe	+1234567890
XYZ789	Jane Smith	+9876543210

Table 6.3 (Admin Database)

**6.3.2. Vehicles:**

Firestore collection: vehicles

Fields:

- vehicleId: Unique identifier for each registered vehicle.
- numberPlate: Vehicle's number plate.
- ownerName: Owner's name.
- ownerContact: Owner's contact information.
- Driver license: owners drivers license
- model: Vehicle model details.

Other relevant vehicle details.

**6.3.3. AccidentReports:**

Firestore collection: accidentReports

Fields:

- vehicleId: Reference to the registered vehicle involved in the accident.
- dateTime: Date and time of the accident.
- location: Accident location details.
- details: Description of the accident.

LogID	VehicleNumber	LogType	Timestamp
1	ABC123	Accident	2024-01-01 10:00
2	XYZ789	Accident	2024-01-02 5:30

Table 6.4 (Accident Reports)

**6.3.4. Complaints:**

Firestore collection: complaints

Fields:

- vehicleId: Reference to the registered vehicle associated with the complaint.
- dateTime: Date and time of the complaint.
- details: Description of the complaint.



### 6.3.5 Relationships:

#### AdminVehicles Relationship:

Each admin has a collection of vehicles they manage. The adminId in the vehicles collection links to the respective admin.

#### VehicleAccidentReports Relationship:

Each registered vehicle can be associated with multiple accident reports. The vehicleId in the accidentReports collection links to the respective vehicle.

#### VehicleComplaints Relationship:

Each registered vehicle can have multiple complaints associated with it. The vehicleId in the complaints collection links to the respective vehicle.

### Security Rules:

Implement Firebase security rules to restrict access and ensure data integrity. For example, only authenticated admins should be able to write to the accidentReports and complaints collections.

## 6.4 Use Case Diagram

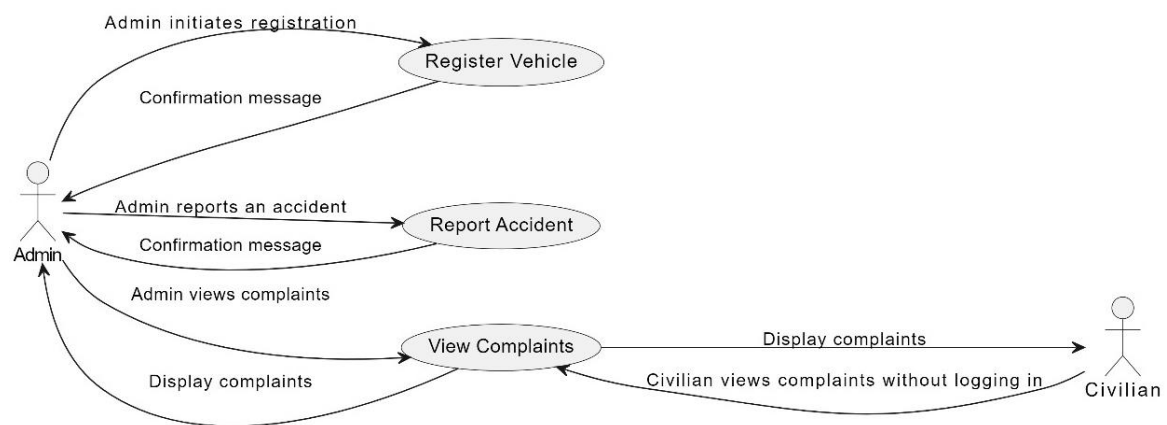


Fig.6.3 Use Case Diagram

### 1. Actors:

- **Admin (A):** Represents an administrator with privileged access to the accident reporting system. The Admin is responsible for managing vehicle registrations, accident reporting, and viewing complaints.
- **Civilian (C):** Represents a civilian user who interacts with the system, primarily for viewing registered complaints without the need to log in.

### 2. Use Cases:

- **Register Vehicle (UC1):**
  - Description: The admin initiates this use case to register a new vehicle in the system. Details such as plate number, owner information, and vehicle details are provided.
  - Actor Interaction: Admin (A) interacts with the system to maintain an updated record of registered vehicles.
- **Report Accident (UC2):**
  - Description: The admin utilizes this use case to report details of a vehicular accident. Information such as accident description, location, and timestamp are recorded.
  - Actor Interaction: Admin (A) actively contributes to the system's accident reporting functionality.
- **View Complaints (UC3):**
  - Description: Both Admin and Civilian users can initiate this use case to view registered complaints. Admins can access a comprehensive list, while Civilians have limited access for viewing purposes.
  - Actor Interaction: Admin (A) uses this use case to manage and address reported issues, while Civilian (C) gains insights into the complaints without logging in.

### 3. Actor-Use Case Relationships:

- **Admin (A) to UC1:** The Admin, acting in the system's managerial capacity, engages in registering new vehicles to ensure an accurate and updated database.
- **Admin (A) to UC2:** Admins contribute to the accident reporting process, providing crucial information to the system for proper documentation and analysis.
- **Admin (A) to UC3:** Admins utilize this use case to stay informed about registered complaints, enabling effective management and resolution.
- **Civilian (C) to UC3:** Civilians, without logging in, can use this use case to view complaints, fostering transparency in the reporting system.

## 6.5 Sequence Diagram

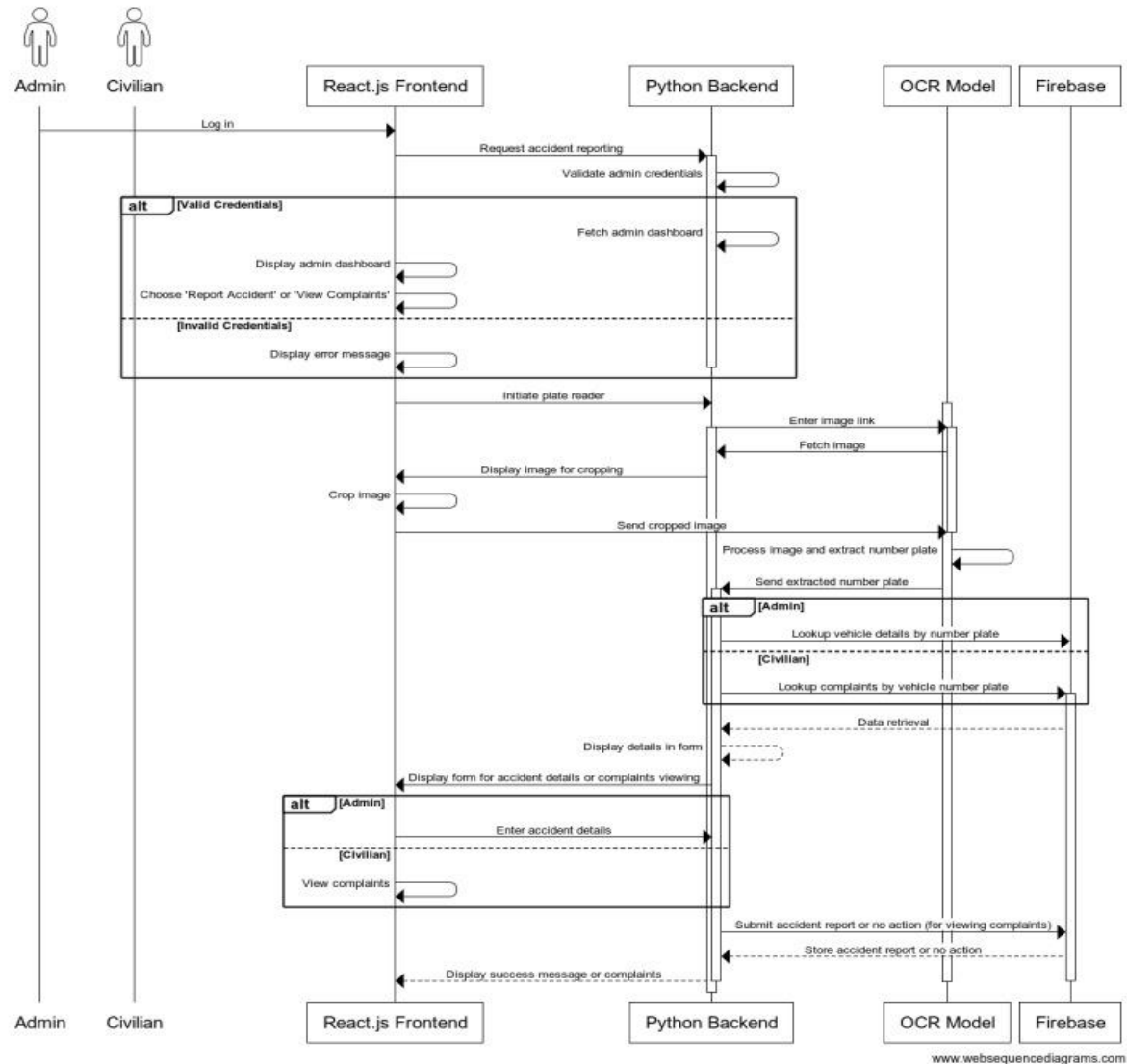


Fig.6.4 Sequence Diagram

The process begins with Admin and Civilian actors interacting with a React.js Frontend. Upon logging in, Admins access the Admin Dashboard to either report accidents or view complaints. The system validates Admin credentials and displays relevant options.

The Plate Reader, initiated by the Backend, uses OCR to extract number plates from images provided by Admins. The extracted plate is then looked up in Firebase for vehicle details. Admins can enter accident details, which are stored in Firebase, or view complaints. For Civilians, the process involves viewing complaints without authentication. The Backend retrieves complaint details from Firebase based on the vehicle's number plate. The complaints or viewing action is then displayed on the Frontend.

## 6.6 Class Diagram

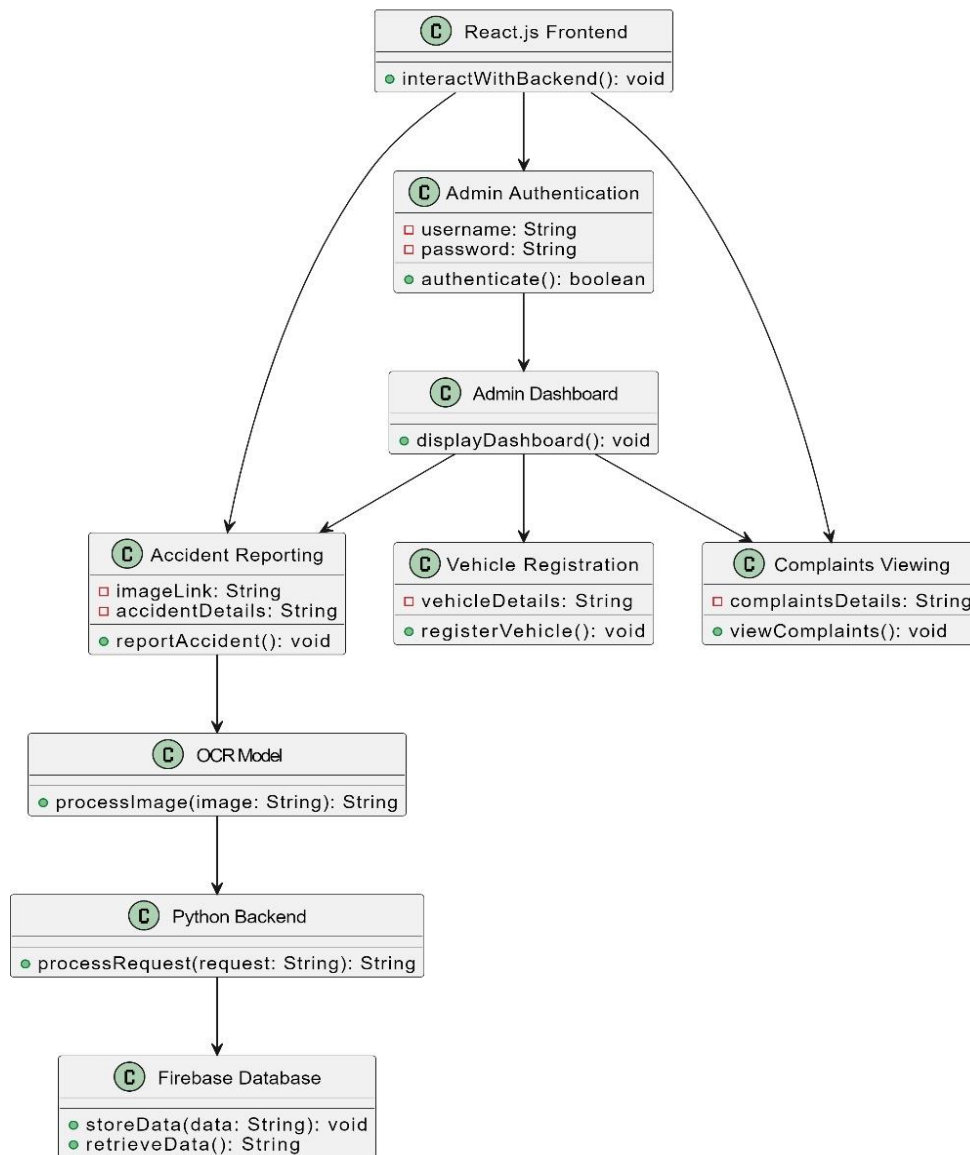


Fig 6.5 Class Diagram

The diagram represents the class structure of an accident reporting system. It includes key components such as Admin Authentication, Admin Dashboard, Vehicle Registration, Accident Reporting, Complaints Viewing, Firebase Database, OCR Model, React.js Frontend, and Python Backend.

### 1. Admin Authentication:

- Manages the authentication process with username and password.
- Has a method `authenticate()` to validate admin credentials.

### 2. Admin Dashboard:

- Displays the admin dashboard after successful authentication.
- Interacts with Vehicle Registration, Accident Reporting, and Complaints

Viewing components.

### **3. Vehicle Registration:**

- Handles the registration of new vehicles.
- Includes a method `registerVehicle()` to initiate the registration process.

### **4. Accident Reporting:**

- Manages the reporting of accidents.
- Links to the OCR Model to process images and extract relevant information.
- Has a method `reportAccident()` to initiate the accident reporting process.

### **5. Complaints Viewing:**

- Deals with viewing complaints, either for admins or civilians.
- Has a method `viewComplaints()` to initiate the viewing process.

### **6. Firebase Database:**

- Stores and retrieves data from the Firebase database.
- Utilized by Python Backend for data processing.

### **7. OCR Model:**

- Processes images to extract information using Optical Character Recognition.
- Linked to Accident Reporting for processing accidentrelated images.

### **8. React.js Frontend:**

- Interacts with Admin Authentication, Accident Reporting, and Complaints Viewing components.
- Initiates communication with the backend for various operations.

### **9. Python Backend:**

- Processes requests from the React.js Frontend.
- Utilizes OCR Model and interacts with the Firebase Database.

---

## CHAPTER-7

### TIMELINE FOR EXECUTION OF PROJECT

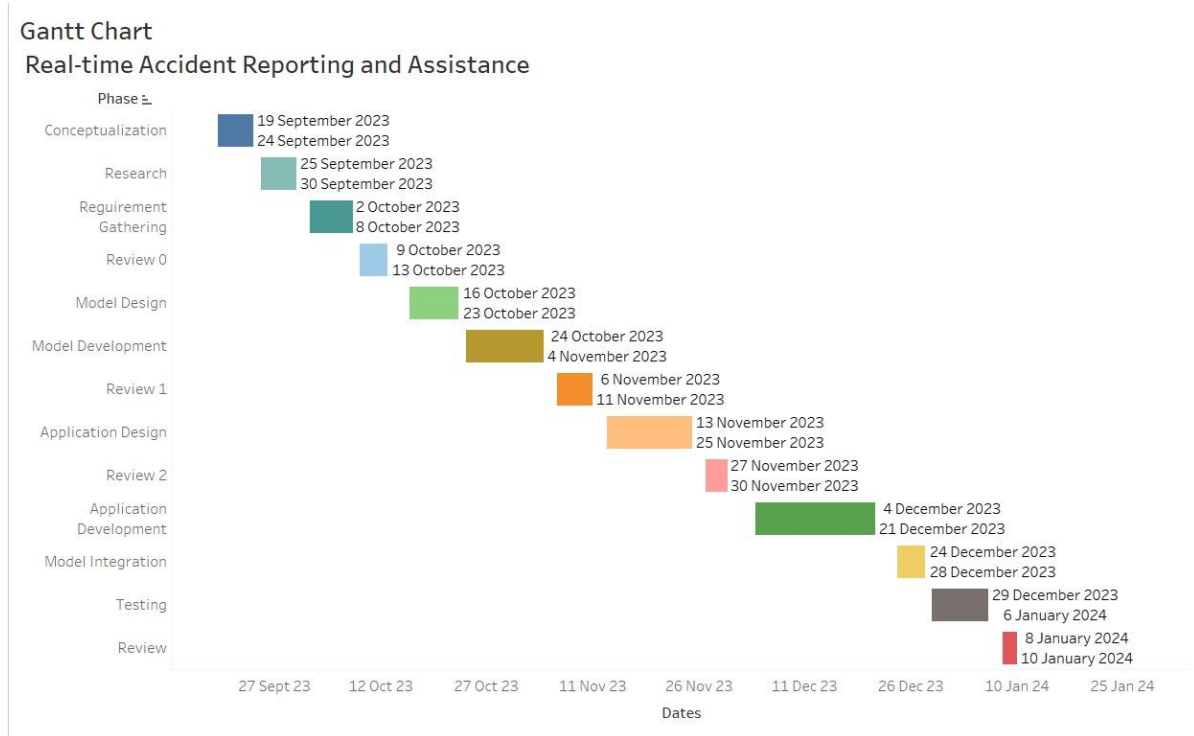


Fig.7.1 Gannt Chart

#### 1. Topic Selection (1 week):

- Define project objectives and scope.
- Identify stakeholders.

#### 2. Research and Planning (2 weeks):

- Explore suitable technologies and frameworks.
- Create a basic project plan.

#### 3. User Stories and Use Case Analysis (2 weeks):

- Develop user stories and use cases.
- Refine requirements based on project guidelines.
- Design Phase:

#### 4. UI/UX Design (3 weeks):

- Develop a user-friendly interface design.
  - Keep the design minimalistic and functional.
-

**5. Database Schema Implementation (2 weeks):**

- Implement a basic database schema.
- Ensure simplicity in data management.

**6. Backend Development (50%) (4 weeks):**

- Implement backend functionalities.
- Develop frontend components.

**7. Integration Testing (Phase 1) (2 weeks):**

- Conduct integration testing.
- Address any early-stage issues.
- Development Phase:

**8. Backend Development (100%) (4 weeks):**

- Complete backend functionalities.
- Finalize frontend development.

**9. Testing (User Acceptance Testing - UAT) (3 weeks):**

- Perform testing, including user acceptance testing.
- Address identified issues.

**10. Documentation (2 weeks):**

- Create comprehensive documentation covering codebase, user manuals, and project report.

**11. Final Phase Demo Preparation (2 weeks):**

- Prepare for project demonstrations.
- Create a presentation for Final Delivery of Project

---

## CHAPTER-8

### OUTCOMES

The anticipated outcomes of the Accident Reporting Web Application project encompass a multitude of positive impacts and benefits, creating a comprehensive and efficient system for handling road incidents. Here is a structured overview of the expected outcomes:

#### **1. Efficient Accident Reporting:**

- Admins can report accidents using the integrated OCR technology.
- OCR facilitates the extraction of number plates from images, streamlining the reporting process.

#### **2. Vehicle Registration:**

- Admins can register new civilian vehicles with their details.
- The registration process is user-friendly and integrated into the admin dashboard.

#### **3. Complaints Viewing:**

- Civilians can view complaints registered against their vehicles without the need for authentication.
- Enhances transparency and allows civilians to stay informed about reported incidents.

#### **4. Optical Character Recognition (OCR) Integration:**

- OCR model processes images and extracts number plates accurately.
- Enhances the accuracy and efficiency of accident reporting by automating number plate extraction.

#### **5. Firebase Database Management:**

- Firebase is utilized for storing vehicle details, accident reports, and complaints.
- Provides a secure and scalable database solution for the application.

#### **6. User-friendly Admin Dashboard:**

- Admins have a dedicated dashboard for managing accident reports, vehicle registrations, and viewing complaints.
  - Simplifies the navigation and control for administrators.
-



### **7. Secure Authentication:**

- Admins need to authenticate themselves to access the system.
- Ensures that only authorized personnel can report accidents and manage vehicle registrations.

### **8. Realtime Reporting:**

- Accidents are reported and processed in near real-time, allowing for quick response and resolution.

### **9. Enhanced Accountability:**

- The system maintains a record of reported accidents, providing accountability and aiding in follow-up actions.

### **10. Technology Stack:**

- Utilizes React.js for the frontend, Python for OCR integration, and Firebase for database management.
- Modern and efficient technology stack ensures reliability and ease of maintenance.

### **11. Scalability and Extensibility:**

- The modular architecture allows for easy scalability and future enhancements.
- Additional features and improvements can be seamlessly integrated into the existing system.

---

## **CHAPTER-9**

### **RESULTS AND DISCUSSIONS**

#### **9.1 Advantages**

##### **9.1.1 Efficient Accident Reporting**

The integration of Optical Character Recognition (OCR) technology significantly enhances the efficiency of accident reporting. Admins can quickly report incidents by extracting information from images, reducing the manual effort involved.

##### **9.1.2 User-Friendly Admin Dashboard**

The user-friendly admin dashboard provides a centralized interface for admins to manage various aspects of the system, including accident reporting, vehicle registration, and complaints viewing. This streamlined approach improves overall usability.

##### **9.1.3 Real-time Reporting**

Accidents are reported and processed in near real-time, allowing for prompt response and faster resolution of reported incidents. This real-time capability is crucial for effective incident management.

##### **9.1.4 Firebase Database Integration**

Utilizing Firebase as the database solution offers advantages such as scalability, real-time data synchronization, and secure data storage. This integration ensures a robust and reliable foundation for the application.

##### **9.1.5 Transparent Complaints Viewing**

Civilians can view complaints against their vehicles without the need for authentication. This transparency fosters accountability and helps civilians stay informed about incidents involving their vehicles.

##### **9.1.6 Scalability and Extensibility**

The modular architecture of the system allows for easy scalability and future enhancements. Additional features and improvements can be seamlessly integrated, ensuring the system's adaptability to changing requirements.

---

## **9.2. Disadvantages**

### **9.2.1 Dependency on Image Quality**

The accuracy of OCR is dependent on the quality of the images provided. Poor image quality may lead to errors in number plate extraction, affecting the overall reliability of the system.

### **9.2.2 Prone to False Positives**

While OCR technology is powerful, it is not immune to false positives. Inaccurate extraction of number plates may occur, leading to potential errors in reporting and database lookups.

### **9.2.3 Limited Authentication for Civilians**

The system currently provides limited functionality for civilians, primarily focusing on complaints viewing. Enhancing authentication mechanisms and functionalities for civilians may be a future consideration.

### **9.3. Future Scope**

#### **9.3.1 Notification System**

Implementing a notification system can further enhance the system's capabilities. Notifications can be sent to relevant parties, such as emergency services, upon the reporting of accidents, improving response times.

#### **9.3.2 Improved OCR Accuracy**

Continued research and development in OCR technology can lead to improved accuracy in number plate extraction. Implementing the latest advancements in OCR models can mitigate errors and enhance system reliability.

#### **9.3.3 Advanced Analytics and Reporting**

Integrating advanced analytics tools can provide valuable insights into accident trends, contributing to data-driven decision-making. Detailed reports and analytics can be beneficial for both administrators and law enforcement agencies.

#### **9.3.4 User Authentication Enhancements**

Expanding authentication mechanisms for civilians and introducing user accounts can provide a more personalized experience. This may include features such as account settings and notifications for civilians.

#### **9.3.5 Mobile Application Development**

Consideration for developing a mobile application can extend the system's accessibility, allowing users to report accidents or view complaints directly from their mobile devices.

---

## **CHAPTER-10**

### **CONCLUSION**

In the culmination of the accident reporting website project, a comprehensive and innovative solution has been developed to revolutionize the process of reporting and managing accidents. The successful integration of Optical Character Recognition (OCR) technology has emerged as a game-changer, offering efficiency, accuracy, and real-time incident processing. This conclusion reflects on the project's achievements, the impact it brings, and areas for potential enhancement.

#### **Achievements and Impact**

The project has achieved its primary objectives, introducing a novel approach to accident reporting through OCR technology. The Admin Dashboard provides a centralized hub for administrators, streamlining the reporting process and ensuring a more user-friendly experience. Real-time incident reporting facilitates quicker responses, ultimately contributing to improved incident management.

The integration of Firebase as the database solution enhances data storage and retrieval, ensuring a robust and secure foundation for the application. Civilians benefit from the transparency of the system, allowing them to access complaints against their vehicles without authentication, fostering a sense of accountability within the community.

The modular architecture of the system sets the stage for scalability and future enhancements. It signifies adaptability to changing requirements and technological advancements, positioning the project as a sustainable and evolving solution.

## **Challenges and Areas for Improvement**

While celebrating successes, it's essential to acknowledge challenges and areas for improvement:

### **1. OCR Accuracy Challenges:**

The dependency of OCR on image quality poses challenges in achieving consistent accuracy. Continuous refinement of the OCR model is crucial to address potential errors and enhance performance.

### **2. Limited Civilian Functionality:**

The system currently offers limited functionalities for civilians. Exploring avenues to enhance civilian participation through additional features and authentication mechanisms could elevate the system's inclusivity.

## **Future Directions**

The project's conclusion also paves the way for future directions and opportunities:

### **1. Notification System Integration:**

Implementing a notification system emerges as a valuable future enhancement. Real-time alerts to relevant parties, such as emergency services, can significantly improve response times and overall effectiveness.

### **2. Advanced Analytics and Reporting:**

The integration of advanced analytics tools can unlock valuable insights into accident trends. Robust reporting mechanisms can empower administrators and law enforcement agencies with data-driven decision-making capabilities.

### **3. User-Centric Enhancements:**

Enhancing user authentication mechanisms for civilians and introducing personalized features can contribute to a more engaging and user-centric experience.

## REFERENCES

1. Fanca, Alexandra & Adela, Puscasiu & Valean, Honoriu & Folea, Silviu. (2018). **"A Survey on Smartphone-Based Accident Reporting and Guidance Systems"**. International Journal of Advanced Computer Science and Applications. 9. 10.14569/IJACSA.2018.090952.
2. Gufran, Dr & M., Dr. (2012). **"Modeling of Traffic Accident Reporting System through UML Using GIS"**. International Journal of Advanced Computer Science and Applications. 3. 10.14569/IJACSA.2012.030606.
3. Hussin, Fahmi & Huzaimy, Jusoh & Sulaiman, Ahmad & Aziz, Mohd & Othman, Faiz & Ismail, Muhammad. (2015). **"Accident reporting system using an iOS application"**. Proceedings - 2014 IEEE Conference on System, Process and Control, ICSPC 2014. 13-18. 10.1109/SPC.2014.7086222.
4. Fanca, Alexandra & Adela, Puscasiu & Valean, Honoriu. (2016). **"Accident reporting and guidance system: With automatic detection of the accident"**. 150-155. 10.1109/ICSTCC.2016.7790657.
5. C. A. Villanueva and T. D. Palaoag, **"Android-Based Real-Time Road Accident Reporting Application,"** 2022 6th International Conference on Information Technology (InCIT), Nonthaburi, Thailand, 2022, pp. 285-288, doi: 10.1109/InCIT56086.2022.10067255.
6. Muthuvel, Marimuthu & Nivetha, S & Sirushti, K. (2018). **"Accident Detection and Reporting System using Internet of Things"**.
7. **"ACCIDENT DETECTION AND ALERT SYSTEM."**  
By Gomathy C K.; Rohan K.; Bandi Mani Kiran Reddy; Dr. V Geetha (2022).
8. C. Nandagopal, P. Anisha, K. G. Dharani and N. Kuraloviya, **"Smart Accident Detection and Rescue System using VANET,"** 2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), Erode, India, 2022, pp. 1111-1116, doi: 10.1109/ICSCDS53736.2022.9760714.

---

## APPENDIX-A

### PSUEDOCODE

#### App.js

```
import React, { useState } from 'react';
import Navbar from './component/Navbar';
import { BrowserRouter as Router, Switch, Route } from 'react-router-dom';
import './App.css';
import Home from './component/pages/Home';
import PlateReader from './App/PlateReader';
import Footer from './component/Footer';
import ComplaintHistory from './component/ComplaintHistory';

function App() {
  const [user, setUser] = useState(null);

  const changeUser = (newUser) => {
    setUser(newUser);
  }

  const currentPath = window.location.pathname;

  return (
    <>
      <Router>
        {/* Conditionally render Navbar based on the current path */}
        {currentPath !== '/platereader' && (
          <Navbar user={user} changeUser={({newUser}) => changeUser(newUser)}
        />
        )}

        <Switch>
          <Route path="/" exact><Home user={user} /></Route>
          <Route path="/platereader" exact><PlateReader /></Route>
          <Route path="/complaint" exact><ComplaintHistory /></Route>
          {/* Add other routes as needed */}
        </Switch>

        <Footer />
      </Router>
    </>
  );
}

export default App;
```

---



## RegisterCivilian.js

```
import React, { useState } from 'react';
import { db, storage } from '../firebase';
import firebase from "firebase";
import './RegisterComplain.css';
import Modal from '@material-ui/core/Modal';
import { makeStyles } from '@material-ui/core/styles';
import { Button, Input } from '@material-ui/core';
function getModalStyle() {
  const top = 50;
  const left = 50;

  return {
    top: `${top}%`,
    left: `${left}%`,
    transform: `translate(-${top}%, -${left}%)`,
  };
}

const useStyles = makeStyles((theme) => ({
  paper: {
    position: 'absolute',
    width: 500,
    backgroundColor: theme.palette.background.paper,
    border: '2px solid #000',
    boxShadow: theme.shadows[5],
    padding: theme.spacing(2, 4, 3),
  },
})));

function RegisterComplain({ btnName, user }) {
  const [progress, setProgress] = useState(0);
  const [name, setName] = useState('');
  const [mobno, setMobno] = useState('');
  const [email, setEmail] = useState('');
  const [address, setAddress] = useState('');
  const [city, setCity] = useState('');
  const [pincode, setPincode] = useState('');
  const [dlno, setdl] = useState('');
  const [vehicleNumber, setVehicle] = useState('');
  const [type, setType] = useState('');
  const [model, setmodel] = useState('');
  const [open, setOpen] = useState(false);
  const classes = useStyles();
  const [modalStyle] = useState(getModalStyle);
  const handleUpload = (event) => {
```

```
event.preventDefault();

// Assuming db is your Firestore database reference
const vehiclesCollection = db.collection("vehicles");

// Use the vehicleNumber as the key
const vehicleDocRef = vehiclesCollection.doc(vehicleNumber);

// Set the data for the document
vehicleDocRef.set({
  name: name,
  username: user.displayName,
  mobno: mobno,
  email: email,
  address: address,
  city: city,
  pincode: pincode,
  dlno: dlno,
  vehicleNumber: vehicleNumber,
  type: type,
  model: model,
  timestamp: firebase.firestore.FieldValue.serverTimestamp()
})
.then(() => {
  console.log("Document successfully written!");

  // Reset the form fields
  setName('');
  setMobno('');
  setEmail('');
  setAddress('');
  setCity('');
  setPincode('');
  setDl('');
  setVehicle('');
  setType('');
  setModel('');
})
.catch((error) => {
  console.error("Error writing document: ", error);
});
}
return (
  <div className="RegiserComplain">
    <Modal
      open={open}
      onClose={() => setOpen(false)}
    >
```

```
<div style={modalStyle} className={classes.paper}>
  <form className="nav__signup">
    <center>
      
    </center>
    <Input
      type="text"
      placeholder="Enter your Name "
      value={name}
      onChange={event => setName(event.target.value)}
    />

    <Input
      placeholder="Mobile No. "
      type="text"
      value={mobno}
      onChange={(event) =>
setMobno(event.target.value)}
    />

    <Input
      placeholder="Email "
      type="text"
      value={email}
      onChange={(event) =>
setemail(event.target.value)}
    />

    <Input
      placeholder="Address "
      type="text"
      value={address}
      onChange={(event) =>
setAddress(event.target.value)}
    />

    <Input
      placeholder="City "
      type="text"
      value={city}
      onChange={(event) => setCity(event.target.value)}
    />
    <Input
```

```
        placeholder="Pincode  "
        type="text"
        value={pincode}
        onChange={(event) =>
setPincode(event.target.value)}
    />
    <Input
        placeholder="Vehicle Number "
        type="text"
        value={vehicleNumber}
        onChange={(event) =>
setVehicle(event.target.value)}
    />

    <Input
        placeholder="Driver's License No. "
        type="text"
        value={dlno}
        onChange={(event) => setdl(event.target.value)}
    />

    <Input
        placeholder="Type: "
        type="text"
        value={type}
        onChange={(event) => settype(event.target.value)}
    />

    <Input
        placeholder="Model. "
        type="text"
        value={model}
        onChange={(event) =>
setmodel(event.target.value)}
    />
    <Button type="submit"
onClick={handleUpload}>ADD</Button>
    </form>
  </div>
</Modal>
<button
  className="nav-btn"
  onClick={() => setOpen(true)}
>
  {btnName}
</button>
</div>
)
```

```
}
```

```
export default RegiserComplain
```

### **PlateReader.js**

```
import React, { useState } from 'react'
import './App.css'
import SearchIcon from '@material-ui/icons/Search';
import ClearAllIcon from '@material-ui/icons/ClearAll';
import ClearIcon from '@material-ui/icons/Clear';
import CropperComponent from '../Cropper/Cropper';
import Result from '../Result/Result';
import FileComplaint from '../component/FileComplaint';

const PlateReader = () => {

  const API_KEY = "b51569a3d3cd4966bd9c42ae043763f7"

  const [query, setQuery] = useState('')
  const [img, setImage] = useState('')
  const [loading, setLoading] = useState(false)
  const [error, setError] = useState(false)
  const [sample, setSample] = useState('')
  const [popup, setPopup] = useState(false)
  const [extractedText, setExtractedText] = useState('');

  const URL =
`https://api.apiflash.com/v1/urltoimage?access_key=${API_KEY}&url=${query}&full_page="true"&fresh="true"&quality=100`

  const search = async(e) => {
    e.preventDefault();

    if(query){
      setLoading(true)
      setPopup(true)
      const response = await fetch(URL);
      if (response.ok) {
        setLoading(false)
        setImage(response)
      }
      else {
        setLoading(false)
        setError(true)
      }
    }
  }

  const clear = () =>{
    setSample('');
    setImage('')
    setQuery('')
  }

  const handleExtractedText = (text) => {
    setExtractedText(text);
  }
}
```

```

        toggleComplaintModal(); // Open the modal when text is extracted
    };
    const [complaintModalOpen, setComplaintModalOpen] = useState(false); //
    New state for modal visibility

    const toggleComplaintModal = () => {
        setComplaintModalOpen(!complaintModalOpen);
    };

    return (
        <div className="app-container">
            <div className="header">
                <div className="input-field">
                    <label>Enter site URL: </label>

                    <div className="group">
                        <input value={query} onChange={(e) =>
setQuery(e.target.value)}
                            placeholder="Enter URL">
                        </input>
                        {query && (
                            <button onClick={() => setQuery('')}>
                                <ClearIcon className="icon"/>
                            </button>
                        )}
                        <button onClick={(e) => search(e)}>
                            <SearchIcon className="icon" />
                        </button>
                    </div>

                    <button onClick={clear} className="clearBtn">
                        <ClearAllIcon className='icon'/>
                    </button>
                </div>
            </div>

            <div className="container">
                {popup && (
                    <div className="popupScreen">
                        <CropperComponent pic={img.url} loading={loading}
                            error={error} setSample={setSample}
setPopup={setPopup}/>
                    </div>
                )}
                <Result sample={sample} setSample={setSample} query={query}
onExtractedText={handleExtractedText} />
                <FileComplaint btnName="File Complaint"
extractedText={extractedText} open={complaintModalOpen}
setOpen={setComplaintModalOpen} />
            </div>
        </div>
    )
}

```

```
export default PlateReader
```

## FileComplaint.js

```
import React, { useState, useEffect } from 'react';
import { db } from '../firebase';
import firebase from "firebase";
import './RegisterComplain.css';
import Modal from '@material-ui/core/Modal';
import { makeStyles } from '@material-ui/core/styles';
import { Button, Input } from '@material-ui/core';

function getModalStyle() {
  const top = 50;
  const left = 50;

  return {
    top: `${top}%`,
    left: `${left}%`,
    transform: `translate(-${top}%, -${left}%)`,
  };
}

const useStyles = makeStyles((theme) => ({
  paper: {
    position: 'absolute',
    width: 500,
    backgroundColor: theme.palette.background.paper,
    border: '2px solid #000',
    boxShadow: theme.shadows[5],
    padding: theme.spacing(2, 4, 3),
  },
})));

function FileComplaint({ btnName, extractedText, open, setOpen }) {
  const [modalStyle] = useState(getModalStyle);
  const [name, setName] = useState('');
  const [address, setAddress] = useState('');
  const [city, setCity] = useState('');
  const [pincode, setPincode] = useState('');
  const [vehicleNumber, setVehicle] = useState(extractedText);
  const [mobno, setMobno] = useState('');
  const [crimeDetails, setCrimeDetails] = useState('');
  const [location, setloc] = useState('');
  const [severity, setsev] = useState('');

  const cleanUpText = (text) => {
    // Remove spaces and special characters
    return text.replace(/[\\s\\W_]+/g, '');
  }
}
```

```
};

const classes = useStyles();
useEffect(() => {
  const fetchData = async () => {
    try {
      const vehicleDoc = await
db.collection('vehicles').doc(cleanUpText(extractedText)).get();
      console.log("fetching data",cleanUpText(extractedText));
      if (vehicleDoc.exists) {
        const vehicleData = vehicleDoc.data();

        // Populate the form fields with data from the "vehicles"
collection
        setName(vehicleData.name || ''); // Replace 'name' with the
actual field in your data
        setAddress(vehicleData.address || ''); // Replace 'address'
with the actual field in your data
        setCity(vehicleData.city || ''); // Replace 'city' with the
actual field in your data
        setPincode(vehicleData.pincode || ''); // Replace 'pincode'
with the actual field in your data
        setVehicle(vehicleData.vehicleNumber || ''); // Replace
'vehicleNumber' with the actual field in your data
        setMobno(vehicleData.mobno || ''); // Replace 'mobno' with the
actual field in your data
      } else {
        console.error("Vehicle not found!");
        // Handle the case where the vehicle is not found
      }
    } catch (error) {
      console.error("Error fetching vehicle data:", error);
    }
  };

  // Fetch data only if the vehicleNumber is not empty
  if (extractedText!=null) {
    fetchData();
  }
}, [extractedText]);
const handleUpload = async (event) => {
  event.preventDefault();

  const complaintsCollection = db.collection("complaints");

  // Set the data for the document in the "complaints" collection
  await complaintsCollection.doc(vehicleNumber).set({
    name: name,
    mobno: mobno,
    address: address,
    city: city,
    pincode: pincode,
    vehicleNumber: vehicleNumber,
    crimeDetails: crimeDetails,
```



```
        location: location,
        severity: severity,
        timestamp: firebase.firestore.FieldValue.serverTimestamp()
    });

    setCrimeDetails('');
    setOpen(false); // Close the modal
};

return (
    <div className="RegisterComplain">
        <Modal
            open={open}
            onClose={() => setOpen(false)}
        >
            <div style={modalStyle} className={classes.paper}>
                <form className="nav__signup">
                    <Input
                        type="text"
                        placeholder="Enter your Name "
                        value={name}
                        onChange={event => setName(event.target.value)}
                    />

                    <Input
                        placeholder="Address "
                        type="text"
                        value={address}
                        onChange={(event) =>
setAddress(event.target.value)}
                    />

                    <Input
                        placeholder="City "
                        type="text"
                        value={city}
                        onChange={(event) => setCity(event.target.value)}
                    />
                    <Input
                        placeholder="Pincode "
                        type="text"
                        value={pincode}
                        onChange={(event) =>
setPincode(event.target.value)}
                    />
                    <Input
                        placeholder="Vehicle Number "
                        type="text"
                        value={vehicleNumber}
                        onChange={(event) =>
setVehicle(event.target.value)}
                    />
                    <Input
                        placeholder="Mobile No. "
```

```
        type="text"
        value={mobno}
        onChange={(event) =>
setMobno(event.target.value)}
      />
      <Input
        placeholder="Accident Details "
        type="text"
        value={crimeDetails}
        onChange={(event) =>
setCrimeDetails(event.target.value)}
      />

      <Input
        placeholder="Loaction"
        type="text"
        value={location}
        onChange={(event) => setloc(event.target.value)}
      />

      <Input
        placeholder="Severity"
        type="text"
        value={severity}
        onChange={(event) => setsev(event.target.value)}
      />
      <Button type="submit"
onClick={handleUpload}>Complain</Button>
    </form>
  </div>
</Modal>
<button
  className="nav-btn"
  onClick={() => setOpen(true)}
>
  {btnName}
</button>
</div>
)
}

export default FileComplaint;
```

---

## APPENDIX-B

### SCREENSHOTS

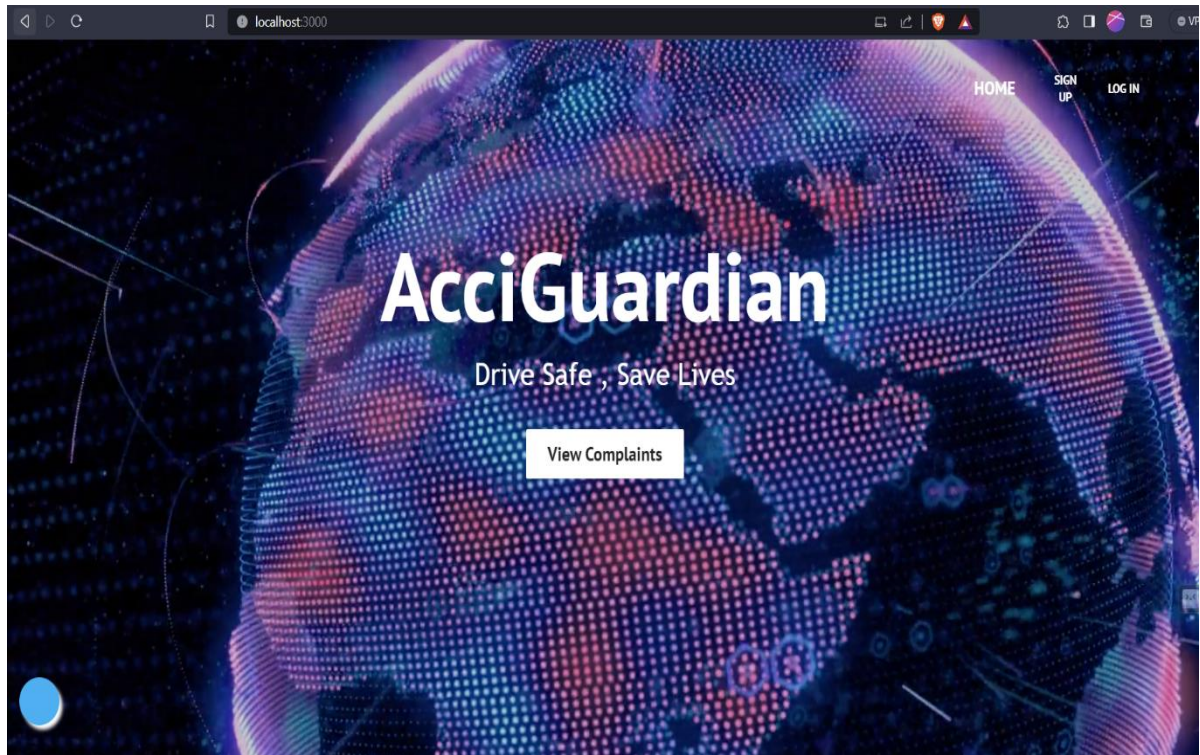


Fig.B.1 Home Page

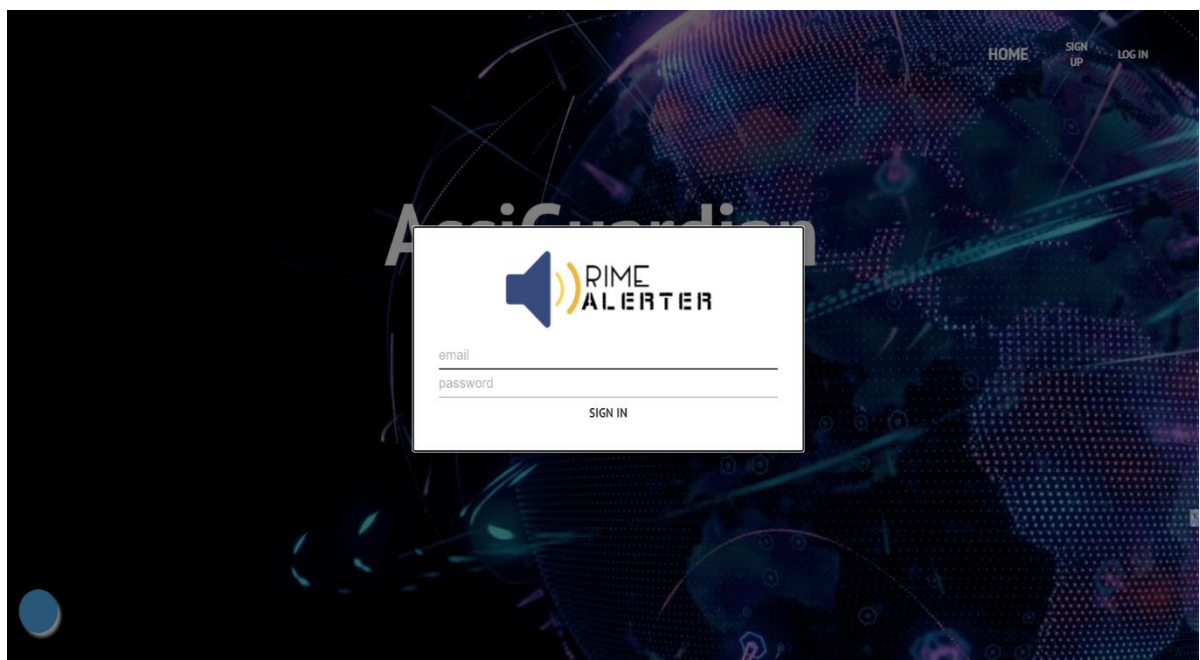


Fig.B.3 Log in

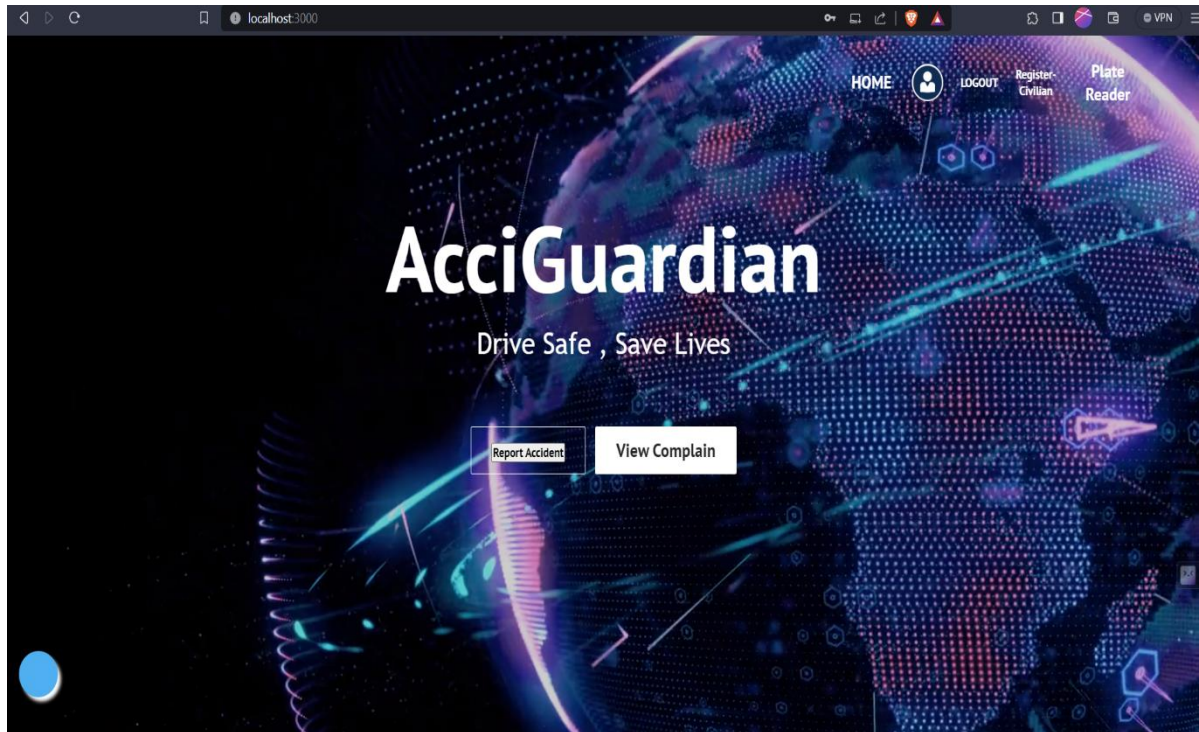


Fig.B.2 Admin Dashboard

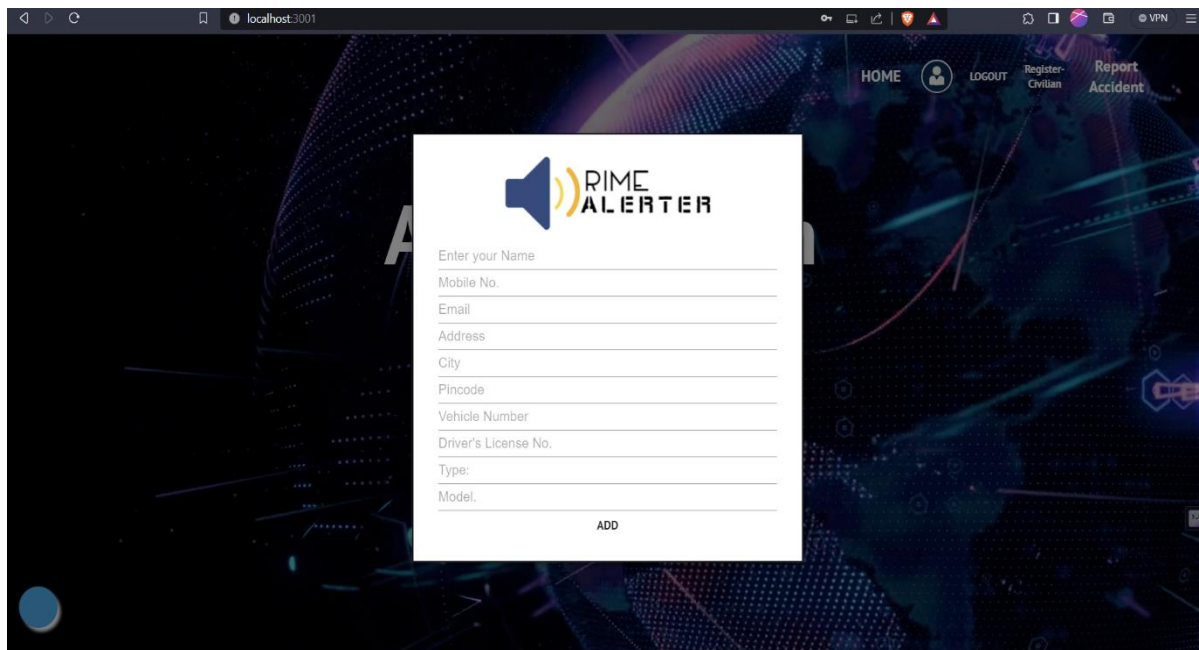


Fig.B.4 Register new civilian



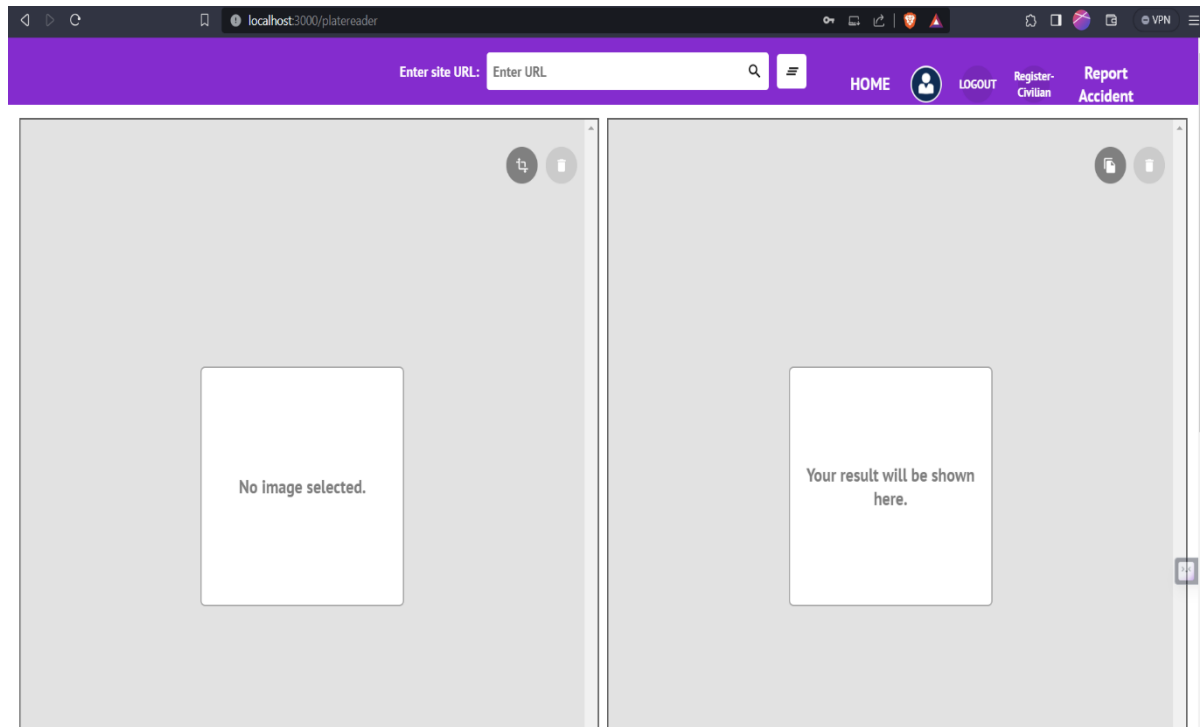


Fig.B.5 Plate Reader

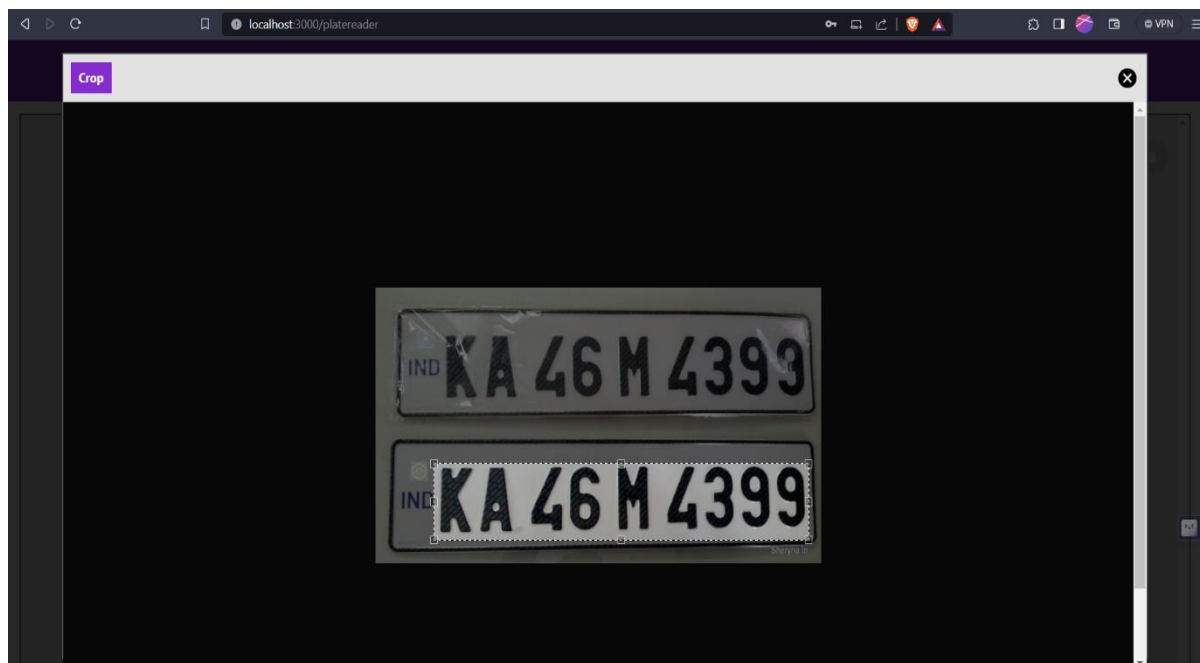
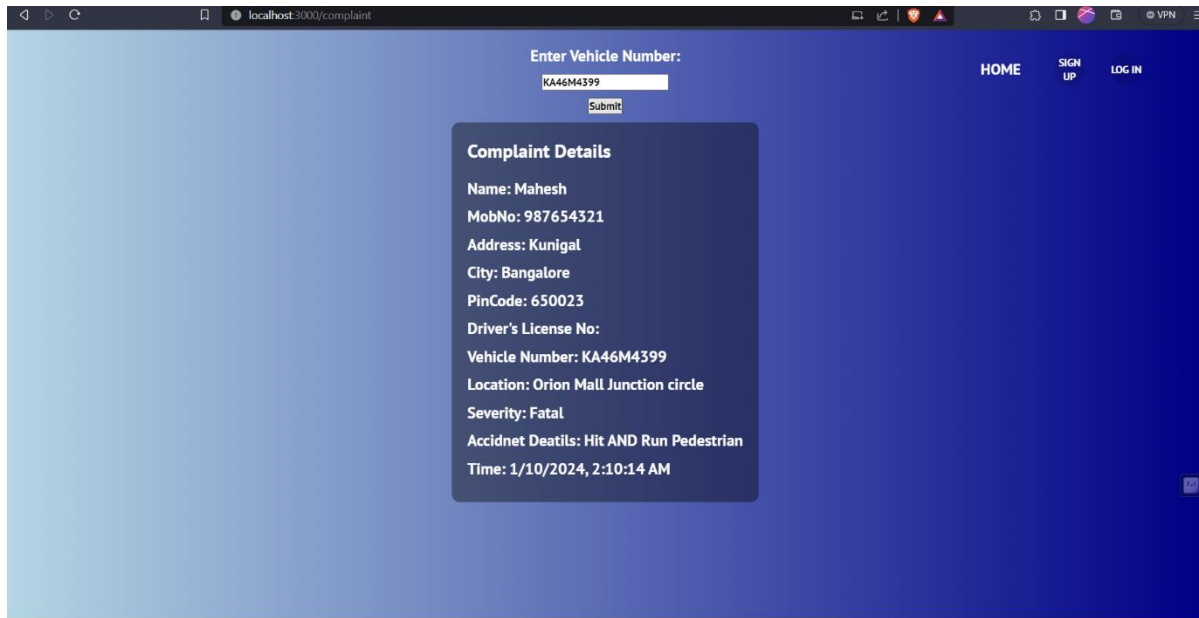


Fig.B.6 Cropper



Enter Vehicle Number:

KA46M4399

Submit

**Complaint Details**

Name: Mahesh  
MobNo: 987654321  
Address: Kunigal  
City: Bangalore  
PinCode: 650023  
Driver's License No:  
Vehicle Number: KA46M4399  
Location: Orion Mall Junction circle  
Severity: Fatal  
Accidnet Deatils: Hit AND Run Pedestrian  
Time: 1/10/2024, 2:10:14 AM

HOME SIGN UP LOG IN

Fig.B.4 Complaint History

## APPENDIX-C

### ENCLOSURES

#### 1. Certificates of journal publishing.



# International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Referred, Scholarly Indexed, Open Access Journal Since 2013)



## CERTIFICATE OF PUBLICATION

The Board of IJIRCCE is hereby awarding this certificate to

**JAYANTH B S**

B.Tech Final Year Student, Computer Science and Engineering, Presidency  
University, Bangalore, India

*Published a paper entitled*

**" Real-Time Road Accident Reporting and Complaint  
Registering Website "**

**in IJIRCCE, Volume 12, Issue 1, January 2024**



e-ISSN: 2320-9801  
p-ISSN: 2320-9798



www.ijircce.com    ijircce@gmail.com



# International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Referred, Scholarly Indexed, Open Access Journal Since 2013)



## CERTIFICATE OF PUBLICATION

The Board of IJIRCCE is hereby awarding this certificate to

**AISHWARYA. OJI**

B.Tech Final Year Student, Computer Science and Engineering, Presidency  
University, Bangalore, India

*Published a paper entitled*

**" Real-Time Road Accident Reporting and Complaint  
Registering Website "**

**in IJIRCCE, Volume 12, Issue 1, January 2024**



Crossref



SJIF Scientific Journal Impact Factor

e-ISSN: 2320-9801  
p-ISSN: 2320-9798



Editor-in-Chief

www.ijircce.com    ijircce@gmail.com

# International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Referred, Scholarly Indexed, Open Access Journal Since 2013)



## CERTIFICATE OF PUBLICATION

The Board of IJIRCCE is hereby awarding this certificate to

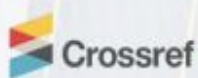
**DEEPAK V**

B.Tech Final Year Student, Computer Science and Engineering, Presidency  
University, Bangalore, India

*Published a paper entitled*

**" Real-Time Road Accident Reporting and Complaint  
Registering Website "**

**in IJIRCCE, Volume 12, Issue 1, January 2024**



e-ISSN: 2320-9801  
p-ISSN: 2320-9798



www.ijircce.com    ijircce@gmail.com



# International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Referred, Scholarly Indexed, Open Access Journal Since 2013)



## CERTIFICATE OF PUBLICATION

The Board of IJIRCCE is hereby awarding this certificate to

**MOHAMMED SHAIZ**

B.Tech Final Year Student, Computer Science and Engineering, Presidency  
University, Bangalore, India

*Published a paper entitled*

**" Real-Time Road Accident Reporting and Complaint  
Registering Website "**

**in IJIRCCE, Volume 12, Issue 1, January 2024**



e-ISSN: 2320-9801  
p-ISSN: 2320-9798



  
Editor-in-Chief

www.ijircce.com    ijircce@gmail.com

## 2. Plagarism Report

ORIGINALITY REPORT			
6%	6%	%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS
PRIMARY SOURCES			
1	pubmed.ncbi.nlm.nih.gov Internet Source	5%	
2	medium.com Internet Source	<1%	
3	fdokumen.id Internet Source	<1%	
4	mountainscholar.org Internet Source	<1%	
5	ece.engin.umich.edu Internet Source	<1%	
6	www.ijraset.com Internet Source	<1%	
7	digitallibrary.usc.edu Internet Source	<1%	
8	docs.wso2.com Internet Source	<1%	
9	roc.ai Internet Source	<1%	

### 3. SDG MAPPING



**The accident reporting web application aligns with several Sustainable Development Goals (SDGs).** Firstly, it contributes to SDG 3 (Good Health and Well-being) by enabling swift accident reporting, potentially reducing injuries and saving lives. SDG 9 (Industry, Innovation, and Infrastructure) is addressed through the application's use of technology, enhancing digital infrastructure for improved emergency response systems. The project aligns with SDG 11 (Sustainable Cities and Communities) by promoting safer urban environments through enhanced accident reporting and response mechanisms. SDG 16 (Peace, Justice, and Strong Institutions) is supported as the project strengthens law enforcement and emergency response capabilities, fostering security and justice. Additionally, the application embraces SDG 17 (Partnerships for the Goals) by collaborating with various stakeholders, including law enforcement and emergency services. This mapping showcases the project's holistic impact on health, innovation, urban sustainability, justice, and collaborative partnerships, aligning with the broader global agenda outlined in the SDGs.