

Automated Vehicle Entry and Exit Management for Residency

A PROJECT REPORT

Submitted by,

Mr. Manoj Yadav N	20201CSE0679
Ms. Nandini Desai	20201CSE0690
Ms. Greeshma S Devadiga	20201CSE0712

Under the guidance of,

Ms. Rakheeba Taseen

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 “Automated Vehicle Entry and Exit Management for Residency” being submitted by Manoj Yadav N, Nandini Desai, Greeshma S Devadiga bearing roll number(s) 20201CSE0679, 20201CSE0690, 20201CSE0712 in partial fulfilment 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. Rakhee Taseen
Assistant Professor
School of CSE
Presidency University


Dr. Pallavi R
Associate Professor & HoD
School of CSE
Presidency University


Dr. C. KALAI ARASAN
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


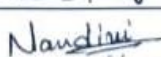
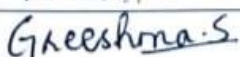
PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **Automated Vehicle Entry and Exit Management for Residency** 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. **Rakheeba Taseen, 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.

NAME	ROLL NO	SIGNATURE
Manoj Yadav N	20201CSE0679	
Nandini Desai	20201CSE0690	
Greeshma S Devadiga	20201CSE0712	

ABSTRACT

In the rapidly advancing landscape of urban living, our project takes center stage, presenting an innovative solution to transform vehicular management within residential areas. At its core, the system functions as a smart traffic controller, harnessing advanced license plate detection and robust database management to ensure secure, authorized, and well-documented access. Beyond access control, the project aligns with the vision of smart city living, where technology converges with urban infrastructure for heightened security and the mitigation of congestion. This venture aspires to redefine residential spaces, envisioning intelligent living environments that seamlessly integrate technology into daily life. By epitomizing the synergy between safety, convenience, and technological innovation, the project strives to illustrate how simple yet powerful applications can revolutionize traffic and security management, establishing new standards for urban living. In essence, it paints a vivid picture of a future where technology becomes a catalyst for elevated living standards, offering a transformative urban experience synonymous with safety, convenience, and innovation. This project encapsulates the essence of a smarter, more secure urban lifestyle, emphasizing the potential of intelligent technology to shape the cities of tomorrow.

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 **Dr. Md. Sameeruddin Khan**, Dean, School of Computer Science and Engineering, 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, 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. Rakheeba Taseen, Assistant Professor**, School of Computer Science and Engineering, Presidency University for her 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 the department Project Coordinators **Mr. Zia Ur Rahaman, Mr. Penial John Whistely**.

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

Mr. Manoj Yadav N	20201CSE0679
Ms. Nandini Desai	20201CSE0690
Ms. Greeshma S Devadiga	20201CSE0712

LIST OF TABLES

Sl. No.	Table Name	Table Caption	Page No.
1	Table 6.1	Registered Users Table	18
2	Table 6.2	Logs Table	18
3	Table 6.3	Testing Table	19

LIST OF FIGURES

Sl. No.	Figure Name	Caption	Page No.
1	Figure 6.1	Android App Architecture	13
2	Figure 6.2	Android App Flowchart	15
3	Figure 7.1	Gantt Chart	21
4	Figure B.1	Home Page	15
5	Figure B.2	Admin Login Page	13
6	Figure B.3	Vehicle Registration Page	15
7	Figure B.4	Scanning of Number Plate	21
8	Figure B.5	Fetching Data from Database	15
9	Figure B.6	Entry and Exit Log	15
10	Figure B.7	SMS Notification	21
11	Figure B.8	Alert Message for Unregistered Vehicle	15

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iv
	ACKNOWLEDGMENT	v
	LIST OF TABLES	vi
	LIST OF FIGURES	vii
1.	INTRODUCTION	2
	1.1 Resident Data Management	2
	1.2 Comprehensive Entry and Exit Logging	2
	1.3 Advanced Number Plate Detection	3
	1.4 Integration of Notification Services	3
	1.5 Database Integration for Holistic Community Management	3
2.	LITERATURE REVIEW	4
	2.1 Plate Detection	4
	2.2 Character Recognition	4
	2.3 Database Management	5
	2.4 The Design of Notification	5
	2.5 Technological Advancements	5
	2.6 OCR and Tesseract	6
	2.7 Use of CNN, RCNN in ALPR	6
	2.8 Database connection using Firebase	7
	2.9 ResNet	7
3.	RESEARCH GAPS OF EXISTING METHODS	9
4.	PROPOSED METHODOLOGY	10
	4.1 Number Plate Recognition Methodology	10
	4.2 Database Management Methodology	10
	4.3 Notification Services Methodology	11
5.	OBJECTIVES	12
6.	SYSTEM DESIGN AND IMPLEMENTATION	13

	6.1 System Architecture	13
	6.2 System Flow Chart	15
	6.3 Database Design	17
	6.4 Testing	19
7.	TIMELINE FOR EXECUTION OF PROJECT	20
8.	OUTCOMES	22
9.	RESULTS AND DISCUSSIONS	24
	9.1 Results	24
	9.2 Discussions	24
10.	CONCLUSION	27
	REFERENCES	28
	APPENDIX-A PSEUDOCODE	30
	A.1 Fragments	30
	A.2 Services	36
	APPENDIX-B SCREENSHOTS	57
	APPENDIX-C ENCLOUSRES	59

CHAPTER-1

INTRODUCTION

The Android application stands as a pioneering solution designed to revolutionize the management of resident data, streamline entry and exit logging processes, and integrate cutting-edge number plate detection for enhanced security within community living environments. This comprehensive introduction aims to provide an in-depth exploration of the project's key features, objectives, and the significant value it brings to modern community living.

1.1 Resident Data Management

In the dynamic landscape of community living, effective resident data management is crucial for seamless operations. The project addresses this need by providing a user-friendly interface for residents to input and update their information. This includes essential details such as names, vehicle numbers, and contact information. To ensure the reliability of the resident database, the system incorporates robust validation mechanisms, guaranteeing the accuracy and completeness of the stored data.

1.1.1 Streamlined Data Entry and Validation

User experience takes center stage, with a streamlined data entry process designed to simplify the task for residents. The system's validation processes are meticulous, acting as a safeguard against erroneous or incomplete entries. By maintaining data integrity, the project establishes a foundation for a dependable and trustworthy resident database. The intuitive design ensures that residents can effortlessly navigate through the data entry process, fostering user adoption and engagement.

1.2 Comprehensive Entry and Exit Logging

A critical aspect of community security and management is the systematic logging of resident entries and exits. The application excels in this area by providing a comprehensive and organized record-keeping system. This feature allows for the efficient tracking and management of resident movements within the community, offering valuable insights for security and administrative purposes. The system timestamps each entry and exit, creating a

chronological log that can be accessed and analyzed for various purposes, including security audits and community planning.

1.3 Advanced Number Plate Detection

Security measures are significantly bolstered through the integration of advanced number plate detection algorithms. This technology empowers the system to recognize and capture vehicle number plates with a high degree of accuracy. Leveraging machine learning and image processing techniques, the application can swiftly and precisely identify vehicles entering or exiting the community premises. This advanced security feature addresses the evolving security needs of modern living, ensuring a vigilant and controlled community environment.

1.4 Integration of Notification Services

In tandem with the robust resident data management and entry/exit logging capabilities, the application integrates notification services. Residents receive timely and relevant notifications, including updates on their data submissions, entry/exit confirmations, and community announcements. This real-time communication feature enhances resident engagement and ensures that community members are well-informed about pertinent events and developments.

1.5 Database Integration for Holistic Community Management

The project envisions a holistic approach to community management by integrating a robust database that consolidates resident data, entry and exit logs, and other relevant information. A single database serves as the repository for all community-related data, facilitating seamless information retrieval and analysis. This centralized database architecture supports scalability, ensuring that the system can accommodate the evolving needs of a growing community.

In conclusion, application is poised to redefine community living through its innovative approach to resident data management, entry and exit logging, and advanced security measures. Subsequent sections will delve deeper into the intricate implementation strategies, user interactions, and the holistic functionality that makes this application a cornerstone for modern, secure, and efficient community living.

CHAPTER-2

LITERATURE SURVEY

1 Plate Detection

Paper name: “Vehicle license plate recognition using super-resolution technique.”

Authors: C. -H. Chuang, L. -W. Tsai, M. -S. Deng, J. -W. Hsieh and K. -C. Fan

Published year: 2014.

Once a vehicle is spotted, the system must determine the area of interest (ROI) where the license plate is likely to be found on the vehicle. The literature paper " Vehicle license plate recognition using super-resolution technique" explains about the stages involved in this. The stage includes localizing the region of the picture or frame holding the license plate.[2] License plate detection may be classified into four categories: the first is utilizing the color as the key attribute. The second is employing edge characteristics to search for licensing block. Within the ROI, the algorithm further localizes the precise place and limits of the license plate. The third is the searching function based on characters in the picture directly. The fourth is the others.

2 Character Recognition

Paper name: “Vehicle license plate recognition using super-resolution technique.”

Authors: C. -H. Chuang, L. -W. Tsai, M. -S. Deng, J. -W. Hsieh and K. -C. Fan

Published year: 2014.

Character recognition is a crucial aspect of license plate detection systems, tackled in two ways: using raw data or extracting features. This is discussed in "Vehicle license plate recognition using super-resolution technique". Raw data involves template matching on binary images, suitable for fixed-size, non-skewed plates. [2] Feature-based recognition enhances accuracy, employing techniques like average deviation calculation, edge detection, Gabor Filters, and sub-block gray value analysis. This dual approach ensures accurate character recognition on license plates, vital for applications such as vehicle tracking and access control.

3 Database Management

Paper name: “An Empirical Study of Local Database Usage in Android Applications.”

Authors: Y. Lyu, J. Gui, M. Wan and W. G. J. Halfond

Published year: 2017.

Many mobile applications display database use difficulties in security and speed. Developers typically adopt inadequate APIs and techniques for SQL command execution, resulting in preventable SQL injection vulnerabilities. The literature paper "An Empirical Study of Local Database Usage in Android Applications," [5] discusses about the usage of untrusted inputs heightens the SQL injection risk. Database startup and write operations are extremely resource-intensive, and their improper organization inside loops without sufficient batching leads to significant resource consumption and inefficiencies. These results give significant insights, underscoring the necessity for developers to address security and performance optimization in mobile app design and execution.

4 The Design of Notification

Paper name: “Unified platform for the delivery of notifications to smartphones notification.”

Authors: A. Mojžišová and M. Mojžiš

Published year: 2012.

Research made in paper "Unified platform for the delivery of notifications to smartphones notification", [6] proves the existence of a main architecture of the system, that emphasizes the usage of stateless interactions and the idea of explicit state transfer. It indicates that numerous data formats are conceivable, although the preference is given to JSON. The paper also briefly touches on the security element, emphasizing the usage of HTTP Digest authentication with HTTPS/TLS.

5 Technological Advancements

Paper name: “Computer Vision based License Plate Detection for Automated Vehicle Parking Management System.”

Authors: N. Darapaneni et al.

Published year: 2020.

The integration of technologies mentioned in "Computer Vision based License Plate Detection for Automated Vehicle Parking Management System", such as Optical Character Recognition (OCR), and YOLO (You Only Look Once) has played a significant role in automating vehicle across control systems.

6 OCR and Tesseract

Paper name: “Automatic License Plate Recognition Using Mobile Device.”

Authors: Hung Ngoc Do, Minh-Thanh Vo, Bao Quoc Vuong

Published year: 2016.

In the above paper the process of optimizing text extraction from images, covering techniques for handling noisy images captured from mobile devices, converting color images to grayscale, and using thresholds for text block identification. It explains procedures such as erosion and dilation operations for noise reduction and character enhancement, as well as the application of local methods for binarization. The overall focus is on the technical steps involved in extracting text from images, including the use of algorithms and operations to improve performance.

OpenCV and the Tesseract engine are fundamental to the development of the model. OpenCV libraries play a crucial role in efficient image processing, saving programming time, and facilitating steps such as license plate localization, character segmentation, and OCR recognition. Additionally, the Tesseract engine, in conjunction with a neural network, is utilized for optical character recognition (OCR) to convert captured license plate images into machine-encoded text. These technologies are essential in achieving the successful processing and extraction of text from images, making the Android program highly effective in meeting its objectives.

7 Use of CNN, RCNN in ALPR

Paper name: “Automatic Number Plate Detection in Vehicles using Faster R-CNN.”

Authors: N. Palanivel Ap, T. Vigneshwaran, M. Sriv Arappadhan, R. Madhanraj

Published year: 2021.

The document primarily focuses on automatic number plate detection using Faster R-CNN, detailing the challenges such as varying lighting conditions and the importance of feature extraction. It discusses object detection techniques, including region proposal networks (RPN)

and the application of convolutional neural networks (CNN) for feature mapping. The paper highlights the localization of number plates on vehicles and the use of image processing, filters, and image preprocessing for plate detection.

Regarding CNN and R-CNN, CNN processes the input image to generate a convolutional feature map, serving as the basis for identifying region proposals, while R-CNN uses selective search for region proposals, slowing down progress. However, Faster R-CNN, by eliminating selective search, significantly improves speed and efficiency. Moreover, R-CNN's region proposal network (RPN) predicts anchors, leading to object identification. In summary, the combination of CNN and R-CNN, particularly in the form of Faster R-CNN, has enhanced feature mapping, region proposal generation, and object identification, thereby improving the performance and efficiency of the automatic number plate detection model.

8 Database connection using Firebase.

Paper name: “Application of Firebase in Android App Development-A Study.”

Authors: Chunnu Khawas, Pritam Shah

Published year: 2018.

The document discusses the use of Firebase in Android application development, highlighting its ability to provide a secure channel for direct communication with the database using JAVA. Firebase features such as Firebase Analytics, Firebase Cloud Messaging (FCM), and Firebase Auth are detailed, emphasizing their roles in app usage insights, cross-platform messaging and notifications, and social login support. Additionally, the document mentions Firebase's storage of data in JSON format, its real-time database, and the specific use of child nodes for storing user email IDs and passwords. Finally, it includes screenshots of an app developed using Firebase and concludes with an emphasis on making Android apps faster and more efficient by eliminating the need for third-party languages like PHP.

9 ResNet

Paper name: “Text Recognition in Moving Vehicles using Deep learning Neural Networks.”

Authors: K.T. Ilayarajaa, V. Vijayakumar, M. Sugadev, T. Ravi

Published year: 2021.

The document discusses a model for vehicle number plate identification using ResNet CNN network for feature extraction and character recognition. ResNet 34, a deep residual network with 34 layers, is employed to extract features and identify characters in the vehicle number plate images. The ResNet architecture, with its residual connections, helps in reducing the resources required for image processing without losing relevant information. This is particularly helpful for character recognition from number plates in various environmental conditions. Additionally, the system successfully recognizes vehicle numbers and ownership details with fair accuracy, even for vehicles moving at speeds up to 10 km/h. The ResNet CNN architecture, with its ability to handle images of different forms, dimensions, and lighting conditions, plays a crucial role in the accurate detection and recognition of vehicle number plates.

CHAPTER-3

RESEARCH GAPS OF EXISTING METHODS

The existing methods in the literature have emphasized the development of an efficient Automatic Number Plate Recognition (ANPR) system, the integration of advanced neural network architectures for accurate identification, and the focus on security measures and robust database management for community living environments. However, specific research gaps may include:

1. **Comprehensive Integration with Community Management Framework:** The existing literature may not extensively address the seamless integration of number plate recognition systems with the broader framework of community management, including factors such as access control, data management, and its impact on overall security within residential areas.
2. **User Experience and Customization:** There might be a research gap in exploring the user experience aspects and the level of customization that residents can have with the notification system in terms of their preferences and security needs within residential societies.
3. **Real-time Data Synchronization and Security:** Further research may be needed to explore the real-time data synchronization capabilities and the robustness of security protocols within the context of residential area management to ensure secure and efficient operation of the number plate detection and recognition systems.

CHAPTER-4

PROPOSED METHODOLOGY

4.1 Number Plate Recognition Methodology:

In the realm of Number Plate Recognition (NPR), the methodology revolves around integrating advanced neural network architectures for accurate identification. The process unfolds in distinct stages:

4.1.1 Data Preparation:

Curate a diverse dataset containing images of varied number plates pertinent to the project. Apply preprocessing techniques such as scaling, noise reduction, and region of interest identification to enhance image quality.

4.1.2 Model Integration and Training:

Configure and integrate Convolutional Neural Networks (CNN), Region-based Convolutional Neural Networks (RCNN), Resolution Network, and YOLO for real-time object detection. Train the integrated model using the prepared dataset to achieve precise character identification.

4.2 Database Management Methodology:

Efficient database management, a cornerstone of the project, is achieved through Firebase integration and SQLite for structured data activities:

4.2.1 Firebase Database Setup:

Design a robust data schema capturing information related to vehicles, license plates, and entry/exit events.

Leverage Firebase's NoSQL capabilities for flexible data storage.

4.2.2 Real-time Data Management:

Integrate Firebase Realtime Database to store and manage dynamic data associated with inhabitants, vehicles, and access activities.

Ensure real-time data synchronization for seamless updates across connected devices.

4.2.3 Offline Capability and Structured Data Management:

Implement offline data management using Firebase to address potential connectivity issues. Complement Firebase's capabilities with SQLite for structured data activities like registration, ensuring a harmonious balance between real-time and structured data integrity.

4.3 Notification Services Methodology:

Effective user communication through notifications is facilitated by Firebase Cloud Messaging (FCM) and Android's Notification Manager:

4.3.1 Notification Implementation:

Utilize Android's NotificationManager and NotificationCompat.Builder for crafting notifications with essential properties such as title, text, and icons.

4.3.2 User Interaction Handling:

Define Intents and PendingIntent to specify user interactions and actions when engaging with notifications.

Implement optional features like notification channels for user preference management.

4.3.3 Message Delivery:

Establish a reliable message delivery mechanism to push notifications to Android devices.

Ensure that notifications are delivered promptly and consistently to enhance user responsiveness.

4.3.4 Notification Channels for User Choice:

Implement notification channels to categorize messages and allow users to manage their preferences. Enhance user experience by providing customization options for different types of notifications.

By adhering to this methodology, the project will implement a robust and user-friendly notification system, focusing on traditional messaging methods without relying on Firebase Cloud Messaging. This approach maintains simplicity while ensuring effective communication with users through timely and well-crafted notifications.

CHAPTER-5

OBJECTIVES

1. Develop and implement an efficient Automatic Number Plate Recognition (ANPR) system using the Open ALPR library in an intuitive Android application for swift and precise vehicle identification during entry and exit from the residential community, while providing real-time SMS notifications to registered residents about their vehicles' movements for enhanced security and convenience.
2. Incorporate comprehensive security measures, including the identification of unregistered or unauthorized vehicles, robust database management, and customizable notification preferences for residents, ensuring a secure and convenient vehicle management system within the residential community.
3. Streamline residential vehicle access and security measures while delivering a hassle-free vehicle management solution for residents, highlighting the superior license plate recognition capabilities of Open ALPR over OpenCV.

CHAPTER-6

SYSTEM DESIGN & IMPLEMENTATION

6.1 System Architecture

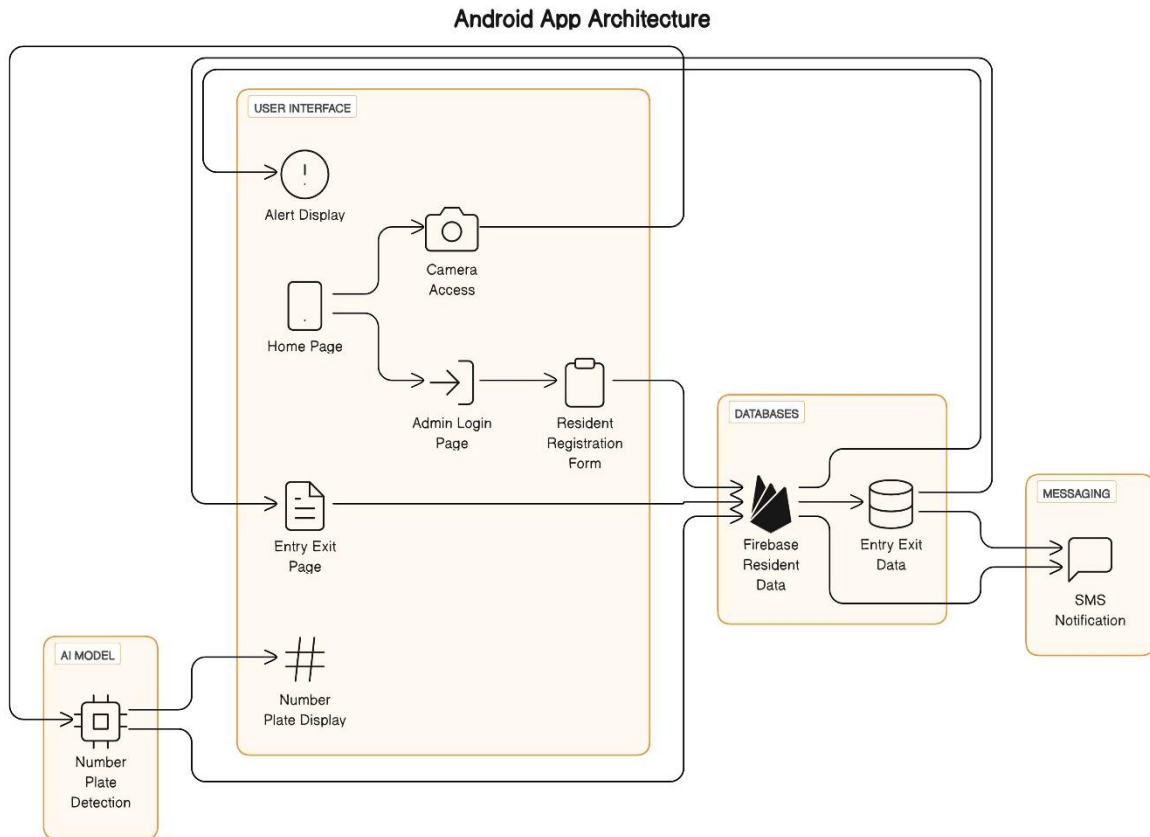


Figure 6.1 Android App Architecture

6.1.1 User Interface and Functions:

- **Alert Display:** This component is responsible for showing alerts or notifications to users if the number plate is not registered in the system. It ensures that users are informed about any discrepancies with the registered vehicles.

- **Home Page:** The landing page of the application provides access to key functions, including scanning number plates and admin login. It serves as the central hub for user interaction.

- **Admin Login Page:** Authorized personnel use this page to access admin-specific functionalities. Upon successful login, access is granted to the Resident Registration Form, where new residents can be registered within the system.

- Resident Registration Form: This interface facilitates the registration of new residents and their pertinent information within the database. It enables the expansion of the system's resident database and is linked to the Firebase Resident Data for storage and retrieval of resident information.

- Camera Access: This allows access to the device's camera for number plate detection. It is linked to the AI model for number plate recognition and also to the Firebase Resident Data for relevant resident information retrieval.

- Entry Exit Page: This page maintains and displays entry and exit records of vehicles. It's connected to the Entry Exit Data database, ensuring accurate tracking of vehicle movements within the residency.

6.1.2 AI Model:

- Number Plate: Utilizes an AI model to detect and recognize number plates from the camera feed. It enables the system to identify and process number plate information for further action, such as resident verification or entry/exit recording.

- Detection: Once the number plate is detected, this component is responsible for displaying the recognized number plate. It bridges the gap between the AI model's recognition and the display interface for user and system reference.

6.1.3 Databases:

- Firebase Resident Data: This database stores and manages information about the registered residents of the system. It serves as a central repository for resident details, enabling efficient resident information retrieval and storage.

- Entry Exit Data: This database maintains records of entry and exit events of the registered vehicle. It ensures accurate tracking and logging of vehicle movements within the residency, enabling efficient management and monitoring.

6.1.4 Messaging:

- SMS Notification: This component is responsible for sending SMS notifications to registered residents about the movement of their vehicle within the residency. It enhances communication and keeps residents informed about their vehicle activities within the premises.

By outlining and detailing the interactions and functions of these components, the system architecture is designed to efficiently manage and monitor the residents, their vehicles, and their activities within the residency, thereby enhancing overall security and management.

6.2 System Flow Chart

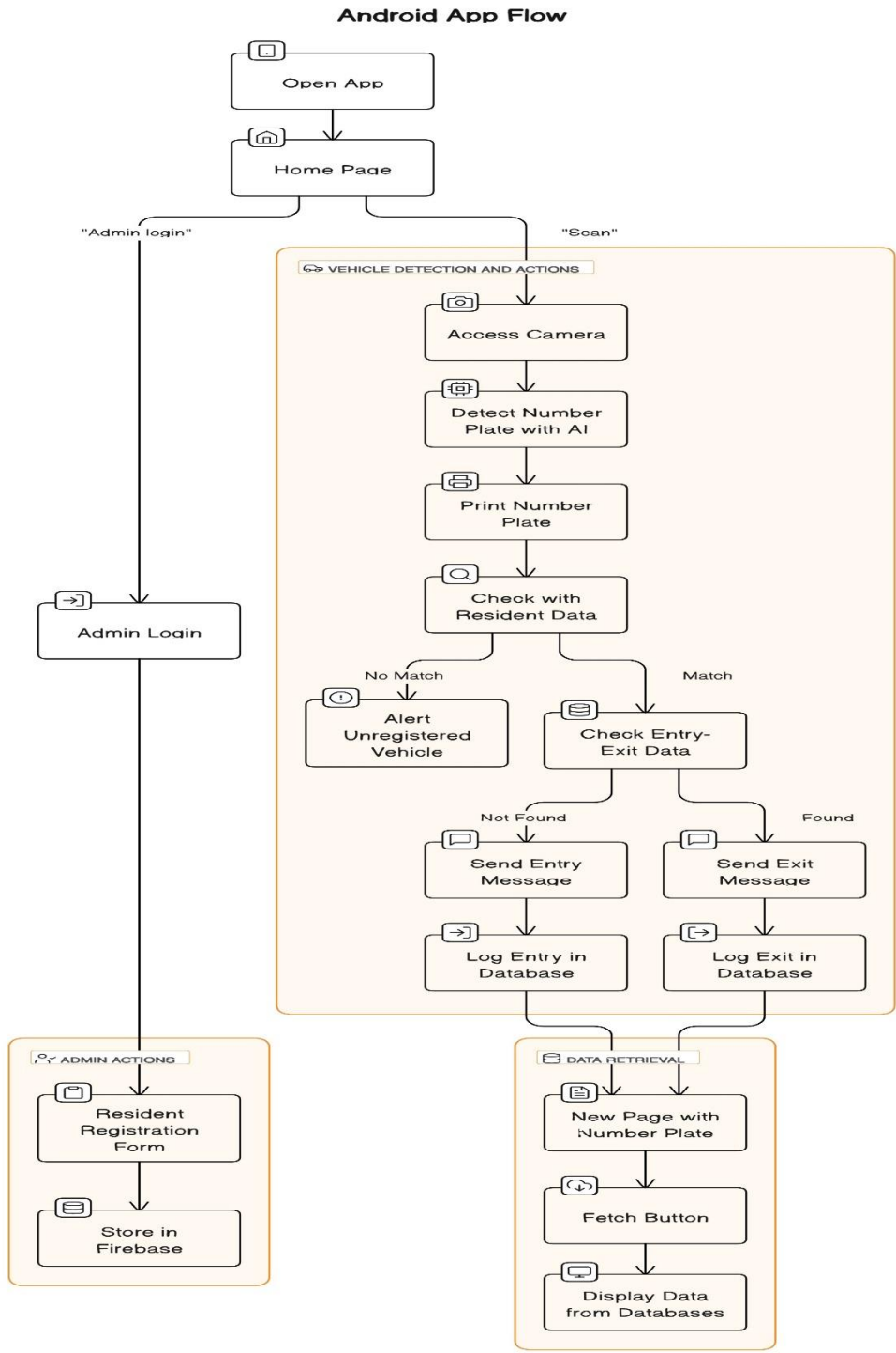


Figure 6.2 Android App Flowchart

6.2.1 Home Page:

- The app's home page contains two buttons: "Scan" and "Admin Login."

6.2.2 Admin Login:

- When the user taps "Admin Login," the app navigates to the Admin Login Page.

6.2.3 Admin Login Page:

- After successful admin login, the user gains access to the Resident Registration Form.

6.2.4 Resident Registration Form:

- This form collects data such as Resident Name, Number Plate, and Phone Number. Upon submission, the data is stored in the Firebase database for later retrieval and verification.

6.2.5 Scan Button:

- When the user taps "Scan," the app activates the device camera to scan for number plates.
- Utilizes an AI model to detect the number plate from the camera feed.
- Once detected, the number plate is displayed in real-time on the screen.

6.2.6 Backend Functionality:

- The detected number plate undergoes verification against the Resident data stored in the Firebase database.
- If the detected number plate does not find a match in the database, an alert "An unregistered vehicle has entered" is displayed on the screen, notifying the user about the unregistered vehicle.
- If the number plate matches an entry in the Resident data, the app checks the Entry-Exit Data database to determine the vehicle's status.
- If the number plate is not present in the Entry-Exit Data database, the corresponding phone number is fetched from the Resident data and a message is sent to notify the resident that their

vehicle has entered. Simultaneously, the number plate, along with the timestamp and entry data, is stored in the Entry-Exit Data database for future reference and tracking.

- If the number plate is already found in the Entry-Exit Data database, indicating that the vehicle is already present in the premises, a message is sent to the corresponding phone number to notify the resident that their vehicle has exited. Subsequently, the number plate, timestamp, and exit data are updated in the Entry-Exit Data database.

6.2.7 Fetch Page:

- The app presents a new page that automatically fetches the number plate and displays all associated data from both the Resident data and Entry-Exit data databases. It includes a "Fetch" button to initiate the retrieval and display of detailed data related to the specific number plate.

6.3 Database Design

Designing a database for your project involves defining tables, relationships, and attributes to store information effectively. Below is a simplified database design based on the functionalities of your OCR reader project. Note that this design assumes certain details about your requirements; you may need to adjust it based on your specific needs.

6.3.1. Registered Users Table:

- Table Name: RegisteredUsers
- Attributes:
 - VehicleNumber (Primary Key)
 - Name
 - PhoneNumber

- Sample Data:

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

Table 6.1 Registered Users Table

6.3.2 Logs Table:

- Table Name: Logs
- Attributes:
 - LogID (Primary Key)
 - VehicleNumber (Foreign Key referencing RegisteredUsers)
 - LogType (Entry/Exit)
 - Timestamp
- Sample Data:

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

Table 6.2 Logs Table

6.3.3 Database Relationships:

- One-to-Many Relationship:
 - RegisteredUsers (One) to Logs (Many):
 - One registered user can have multiple log entries.
 - Connect VehicleNumber in RegisteredUsers to VehicleNumber in Logs.

6.4 Testing

Test Case Number	Testing Scenario	Expected result	Result
Admin Login Testing:			
TC-01	Clicking submit without entering login details	Alert "Please enter the username and password"	Pass
TC-02	Clicking submit without entering user name	Alert "Please fill Username"	Pass
TC-03	Clicking submit without entering password	Alert "Please enter the password"	Pass
TC-04	Clicking submit entering wrong email	Alert "Invalid Username or Password"	Pass
TC-05	Clicking submit entering wrong password	Alert "Invalid Username or Password"	Pass
Resident Registration testing:			
TC-06	Clicking submit without entering Vehicle number	Alert "Vehicle number cannot be empty"	Pass
TC-07	Clicking submit without entering Owner name	Alert "Please enter your name"	Pass
TC-08	Clicking submit without entering Phone number	Alert "Please enter your Phone number"	Pass
Unregistered vehicle entry testing:			
TC-09	When an unregistered vehicle enters	Alert "Unregistered vehicle entered"	Pass

Table 6.3 Testing Table

CHAPTER-7

TIMELINE FOR EXECUTION OF PROJECT

Initial Phase

- Title Finalization
- Literature Survey
- Finalizing Objectives
- Deciding the Methodology
- Existing Methods and Drawbacks
- Proposed Methods
- Hardware and Software Details
- References

Design Phase

- Architecture Diagram
- Modules
- Algorithm Details
- Source Code Details
- 50% implementation

Development Phase

- Algorithm Details
- Source Code Details
- 100% implementation details
- Live Demonstration of the project

Final Phase

- Finalizing the application design
 - Submit 100% of the completed report as a hard copy and soft copy.
 - Live Demonstration of the project
 - Submit a plagiarism report of the project report.
-

GANTT CHART

Gantt Chart

Automated Vehicle Entry and Exit Management for Residency

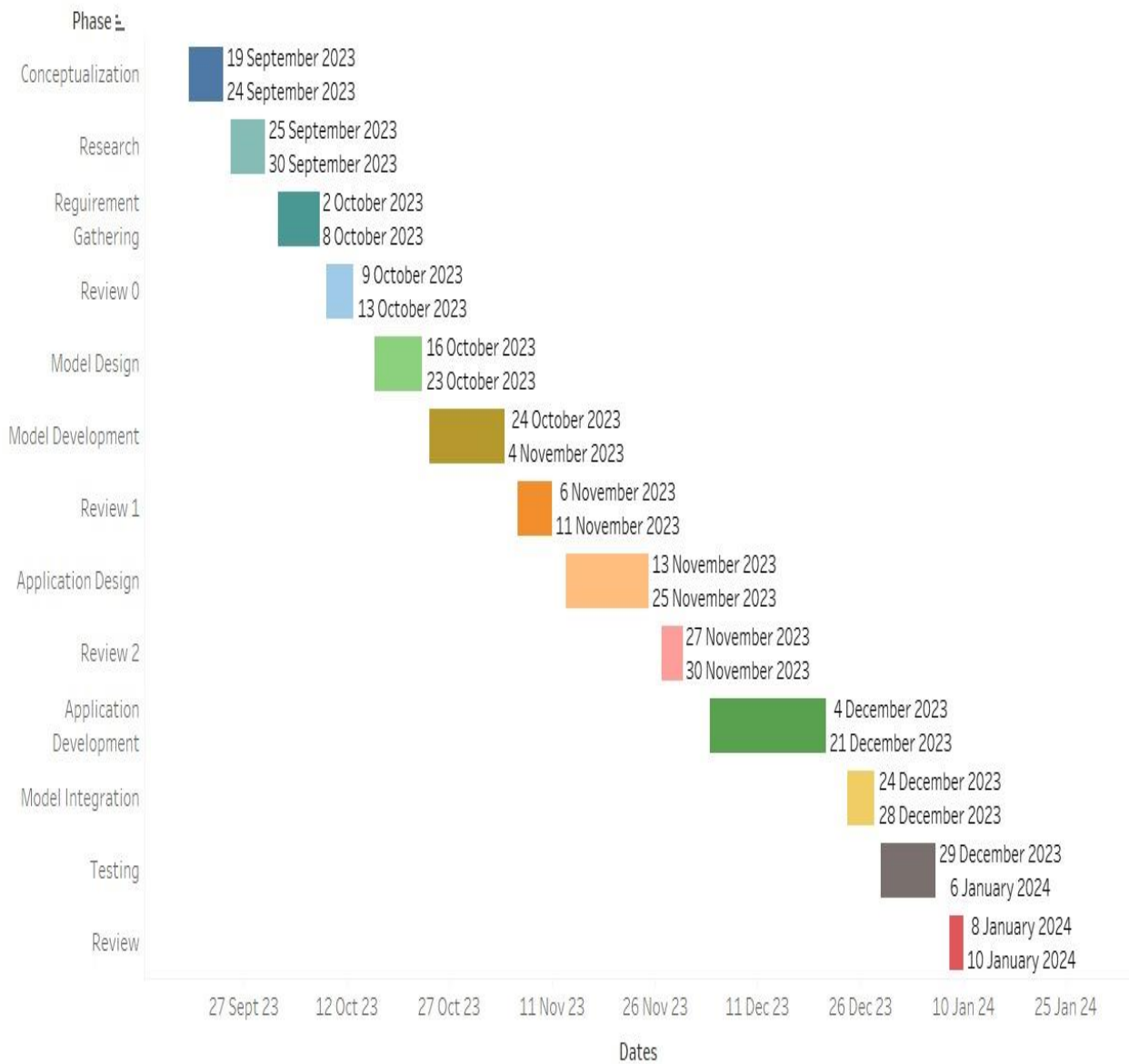


Figure 7.1 Gantt Chart

CHAPTER-8

OUTCOMES

The implementation of an Automated Vehicle Entry and Exit Management system in a residential community can yield several expected outcomes, which are beneficial for both residents and property management. These outcomes include:

8.1 Enhanced Security:

- Reduced unauthorized access: The system helps prevent unauthorized vehicles from entering the community, thereby improving overall security.
- Access logs and records: Access control systems maintain detailed logs of vehicle entries and exits, which can be valuable for security and investigations.

8.2 Improved Safety:

- Controlled traffic flow: The system can manage the flow of vehicles, reducing the risk of accidents and enhancing safety within the community.
- Monitoring of speed limits: Some systems can monitor and enforce speed limits, promoting safe driving within the residential area.

8.3 Visitor Management:

- Streamlined guest access: Residents can pre-register visitors, ensuring a smooth entry process for guests and service providers.
- Reduced inconvenience: Visitors no longer need to wait for manual approvals at entry gates, making their experience more convenient.

8.4 Efficient Service Delivery:

- Improved access for service providers: Delivery services, contractors, and maintenance personnel can be granted access as needed, reducing delays in service delivery.
- Time-stamped records: The system records service provider entries and exits, which can be useful for tracking service appointments.

8.5 Cost Savings:

- Reduced manpower: Automated systems can decrease the need for security personnel or manual gate attendants, leading to potential cost savings.
- Lower maintenance costs: Modern access control systems are typically reliable and require less maintenance compared to manual gates and barriers.

8.6 Privacy and Accountability:

- Secure data handling: Properly configured systems ensure the privacy and security of access data, complying with data protection regulations.
- Accountability: The system holds residents and service providers accountable for their actions, discouraging misconduct.

8.7 Resident Satisfaction:

- Overall improved living experience: Residents typically appreciate the added security and convenience, leading to increased satisfaction with their living environment.

CHAPTER-9

RESULTS AND DISCUSSIONS

9.1 Results

9.1.1 OCR Functionality:

The OCR functionality successfully detects and extracts text from images, particularly license plates.

Detected text is displayed in real-time on the application interface.

9.1.2 Database Integration:

User registration and log tracking are integrated with Firebase Realtime Database.

User information, including name and phone number, is stored securely.

9.1.3 SMS Notifications:

SMS notifications are sent to registered users when their vehicle enters or exits a specific location.

Twilio API (or any other SMS gateway) integration ensures reliable message delivery.

9.1.4 UI/UX Design:

The application features an intuitive user interface for easy navigation.

Visual feedback, such as Snackbar messages, enhances user experience.

9.1.5 Security Measures:

Appropriate security measures, such as Firebase authentication, are implemented to protect user data.

User data and logs are stored securely in the database.

9.2 Discussions

9.2.1 OCR Design, Development, and Integration:

Text Recognition Algorithm:

Delve into the OCR (Optical Character Recognition) algorithm implemented for text recognition in license plates.

Discuss the design considerations, such as the utilization of the Google Mobile Vision API's TextRecognizer, and its role in extracting text information from images.

Integration Challenges and Solutions:

Address challenges encountered during the integration of OCR into the application, such as ensuring accurate recognition under varying lighting conditions.

Discuss solutions implemented, like image preprocessing techniques, to enhance OCR accuracy and overall system performance.

9.2.2 User Interaction with OCR Results:

Real-time Feedback to Users:

Highlight the importance of providing real-time feedback to users based on OCR results.

Discuss how the system interprets license plate information, checks against the database, and triggers appropriate actions, such as sending SMS alerts and updating logs.

9.2.3 User Authentication and Access Control:

Admin Login:

Discuss the implementation of admin login using Firebase authentication, ensuring secure access to administrative functionalities.

Highlight the importance of secure authentication for controlling system access and preventing unauthorized usage.

9.2.4 Database Design and Storage:

Firebase Realtime Database:

Elaborate on the use of Firebase Realtime Database for storing user information and vehicle logs.

Discuss the advantages of a real-time database in maintaining up-to-date information for efficient tracking.

9.2.5 Driver Interaction and Communication:

SMS Alerts:

Explore the integration of SMS alerts to notify vehicle owners upon entry or exit, enhancing real-time communication.

Discuss the benefits of SMS alerts, such as providing immediate information to users without requiring app access.

9.2.6. Mobile App Navigation:

HomeActivity:

Detail the functionalities of the HomeActivity, including scanning, admin login, contact features, and user feedback.

Emphasize the user-friendly design and the seamless navigation experience within the mobile application.

CHAPTER-10

CONCLUSION

In conclusion, the implementation of an Automated Vehicle Entry and Exit Management system in a residential community is a significant step toward enhancing security, convenience, and efficiency for both residents and property management. This project offers a range of valuable outcomes, including improved security, streamlined access, enhanced safety, and cost savings. By embracing technology-driven solutions, residential communities can create a more welcoming and secure living environment while also optimizing operational processes. The success of such a project relies on thorough planning, proper technology selection, and ongoing maintenance to ensure the system's reliability. Additionally, clear communication with residents, training, and the establishment of access policies are crucial for a smooth transition to the new system. As technology continues to evolve, residential communities should also stay up to date with advancements in access management systems to adapt and further improve their living environments. Ultimately, the Automated Vehicle Entry and Exit Management system project not only contributes to the security and convenience of the community but also enhances the overall quality of life for residents.

REFERENCES

1. N. Darapaneni et al., "Computer Vision based License Plate Detection for Automated Vehicle Parking Management System," 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 2020, pp. 0800-0805, doi: 10.1109/UEMCON51285.2020.9298091.
2. C. -H. Chuang, L. -W. Tsai, M. -S. Deng, J. -W. Hsieh and K. -C. Fan, "Vehicle licence plate recognition using super-resolution technique," 2014 11th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Seoul, Korea (South), 2014, pp. 411-416, doi: 10.1109/AVSS.2014.6918703.
3. P. Kulkarni, A. Khatri, P. Banga and K. Shah, "Automatic Number Plate Recognition (ANPR) system for Indian conditions," 2009 19th International Conference Radioelektronika, Bratislava, Slovakia, 2009, pp. 111-114, doi: 10.1109/RADIOELEK.2009.5158763.
4. K. Yogheedha, A. S. A. Nasir, H. Jaafar and S. M. Mamduh, "Automatic Vehicle License Plate Recognition System Based on Image Processing and Template Matching Approach," 2018 International Conference on Computational Approach in Smart Systems Design and Applications (ICASSDA), Kuching, Malaysia, 2018, pp. 1-8, doi: 10.1109/ICASSDA.2018.8477639.
5. Y. Lyu, J. Gui, M. Wan and W. G. J. Halfond, "An Empirical Study of Local Database Usage in Android Applications," 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME), Shanghai, China, 2017, pp. 444-455, doi: 10.1109/ICSME.2017.75.
6. A. Mojžišová and M. Mojžiš, "Unified platform for the delivery of notifications to smartphones notification," Proceedings of the 13th International Carpathian Control Conference (ICCC), High Tatras, Slovakia, 2012, pp. 490-494, doi: 10.1109/CarpathianCC.2012.6228693.

7. G. G. Desai and P. P. Bartakke, "Real-Time Implementation of Indian License Plate Recognition System," 2018 IEEE Punecon, Pune, India, 2018, pp. 1-5, doi: 10.1109/PUNECON.2018.8745419.
8. Khawas, Chunnu & Shah, Pritam. (2018). Application of Firebase in Android App Development-A Study. International Journal of Computer Applications. 179. 49-53. 10.5120/ijca2018917200.
9. N. P. Ap, T. Vigneshwaran, M. S. Arappadhan and R. Madhanraj, "Automatic Number Plate Detection in Vehicles using Faster R-CNN," 2020 International Conference on System, Computation, Automation and Networking (ICSCAN), Pondicherry, India, 2020, pp. 1-6, doi: 10.1109/ICSCAN49426.2020.9262400.

APPENDIX-A

PSUEDOCODE

A.1 Fragments

A.1.1 AdminLogin.java

```
package com.google.android.gms.samples.vision.ocreader;

import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

public class AdminLogin extends AppCompatActivity {
    EditText etun,etpw;
    ProgressDialog progressDialog;
    FirebaseAuth firebaseAuth;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_admin_login);
```

```
etun = (EditText) findViewById(R.id.editTextUN);
etpw = (EditText) findViewById(R.id.editTextPW);
progressDialog = new ProgressDialog(this);
firebaseAuth = FirebaseAuth.getInstance();

}

public void btn_login_click(View v)
{
    String username = etun.getText().toString().trim();
    String password = etpw.getText().toString().trim();

    if(username.isEmpty())
    {
        etun.setError("Username can't be empty");
        etun.requestFocus();
        return;
    }
    if(password.isEmpty())
    {
        etpw.setError("Please enter your password");
        etpw.requestFocus();
        return;
    }
    progressDialog.setMessage("Logging in");
    progressDialog.show();

    firebaseAuth.signInWithEmailAndPassword(username,password).addOnCompleteListener(
        new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task)
            {
                if(task.isSuccessful())
```

```
{
    progressDialog.dismiss();
    Intent myintent = new Intent(AdminLogin.this,RegisterActivity.class);
    startActivity(myintent);
}
else
{
    progressDialog.dismiss();
    Toast.makeText(getApplicationContext(),"Invalid Username or
    Password",Toast.LENGTH_SHORT).show();
}
}
});
}
```

A.1.2 HomeActivity.java

```
package com.google.android.gms.samples.vision.ocrreader;
```

```
import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
```



```
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import com.google.android.material.bottomnavigation.BottomNavigationView;

public class HomeActivity extends AppCompatActivity {

    private TextView mTextMessage, tvloginfo, tvinfo, tvfeedback;
    private ImageView iv2, iv3;
    private Button b5, adlogin, buttonFeedback;

    private BottomNavigationView.OnNavigationItemSelectedListener
mOnNavigationItemSelectedListener
    = new BottomNavigationView.OnNavigationItemSelectedListener() {

        @Override
        public boolean onNavigationItemSelected(@NonNull MenuItem item) {
            return false;
        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);

        mTextMessage = findViewById(R.id.textView5);
        iv2 = findViewById(R.id.imageView2);
        iv3 = findViewById(R.id.imageView3);
        b5 = findViewById(R.id.button5);
        adlogin = findViewById(R.id.adlogin);
        tvloginfo = findViewById(R.id.tvloginfo);
        buttonFeedback = findViewById(R.id.buttonFeedback);
```

```
        tvinfo = findViewById(R.id.tvinfo);
        tvinfo.setVisibility(View.INVISIBLE);
        tvlogininfo.setVisibility(View.INVISIBLE);
        buttonFeedback.setVisibility(View.INVISIBLE);

        BottomNavigationView navigation = findViewById(R.id.navigation);

        navigation.setOnNavigationItemSelectedListener(mOnNavigationItemSelectedListener);
    }

    public void btn_scan(View v) {
        Intent i = new Intent(HomeActivity.this, OcrCaptureActivity.class);
        startActivity(i);
    }
    public void adlogincall(View v)
    {
        Intent j = new Intent(HomeActivity.this, AdminLogin.class);
        startActivity(j);
    }

    public void btn_contact(View v) {
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.CALL_PHONE) !=
        PackageManager.PERMISSION_GRANTED) {
            if (ActivityCompat.shouldShowRequestPermissionRationale(this,
            Manifest.permission.CALL_PHONE)) {
                } else {
                    ActivityCompat.requestPermissions(this, new
                    String[]{Manifest.permission.CALL_PHONE}, 0);
                }
            } else {
                Intent callIntent = new Intent(Intent.ACTION_CALL);
                callIntent.setData(Uri.parse("tel:8763964692"));
                startActivity(callIntent);
            }
        }
```

```
}

    public void onRequestPermissionsResult(int requestCode, String permissions[], int[]
grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        switch (requestCode) {
            case 0: {
                if (grantResults.length > 0
                    && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                    Intent callIntent = new Intent(Intent.ACTION_CALL);
                    callIntent.setData(Uri.parse("tel:8763964692"));
                    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.CALL_PHONE) != PackageManager.PERMISSION_GRANTED) {
                        return;
                    }
                    startActivity(callIntent);
                } else {
                    Toast.makeText(getApplicationContext(),
                        "Call failed, please try again.", Toast.LENGTH_LONG).show();
                    return;
                }
            }
        }
    }

    public void btn_feedback(View v)
    {
        Intent email = new Intent(Intent.ACTION_SEND);
        email.putExtra(Intent.EXTRA_EMAIL, new String[]{ "sambitgulu@gmail.com"});
        email.putExtra(Intent.EXTRA_SUBJECT, "Feedback");
        email.putExtra(Intent.EXTRA_TEXT, "write your valuable feedback / suggestions
here...");

        email.setType("message/rfc822");
    }
}
```

```
        startActivity(Intent.createChooser(email, "Choose an Email client :"));
    }

    @Override
    public void onBackPressed() {
        super.onBackPressed();
        System.exit(1);
        moveTaskToBack(true);
        android.os.Process.killProcess(android.os.Process.myPid());

    }
}
```

A.2 Services

A.2.1 Database

A.2.1.1 DriverClass.java

```
package com.google.android.gms.samples.vision.ocreader;

import android.Manifest;
import android.app.ProgressDialog;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
```

```
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;

public class DriverClass extends AppCompatActivity {

    EditText et1;
    DatabaseReference myFirebase, myFirebase2;
    TextView tvname, tvnameshow, tvphone, tvphoneShow, logText;
    ProgressDialog progressDialog;
    String myChildphone;
    String myChildname;
    String number;
    int flag = 0;
    private static final int MY_PERMISSIONS_REQUEST_SEND_SMS = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_driver_class);
        et1 = findViewById(R.id.editTextd);
        tvname = findViewById(R.id.textViewname);
        tvnameshow = findViewById(R.id.textViewnameShoe);
        tvphone = findViewById(R.id.textViewphone);
        tvphoneShow = findViewById(R.id.textViewphoneShow);
```

```
logText = findViewById(R.id.logTextView);
progressDialog = new ProgressDialog(this);

Intent intent2 = getIntent();
String receive = intent2.getStringExtra("5555");
et1.setText(receive);

myFirebase = FirebaseDatabase.getInstance().getReference();
}

public void btn_fetch(View v) {
    number = et1.getText().toString().trim();
    if (number.isEmpty()) {
        et1.setError("Vehicle number missing");
        et1.requestFocus();
        return;
    }
    progressDialog.setMessage("Fetching data");
    progressDialog.show();

    String path = "Registered-users/" + number;
    myFirebase.child(path).child("name").addListenerForSingleValueEvent(new
com.google.firebase.database.ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        myChildname = dataSnapshot.getValue(String.class);
        progressDialog.dismiss();
        tvname.setVisibility(View.VISIBLE);
        tvnameshow.setText(myChildname);
        flag = 3;
        if (tvnameshow.getText().length() == 0) {
            flag = 1;
            Toast.makeText(DriverClass.this, "Vehicle is not registered",
Toast.LENGTH_LONG).show();
```

```
    }  
  }  
  
  @Override  
  public void onCancelled(@NonNull DatabaseError databaseError) {  
    progressDialog.dismiss();  
    Toast.makeText(getApplicationContext(), "Error retrieving data",  
Toast.LENGTH_SHORT).show();  
    flag = 2;  
  }  
});  
  
  myFirebase.child(path).child("phoneNo").addListenerForSingleValueEvent(new  
com.google.firebase.database.ValueEventListener() {  
  @Override  
  public void onDataChange(@NonNull DataSnapshot dataSnapshot) {  
    myChildphone = dataSnapshot.getValue(String.class);  
    tvphone.setVisibility(View.VISIBLE);  
    tvphoneShow.setText(myChildphone);  
    progressDialog.dismiss();  
  }  
  
  @Override  
  public void onCancelled(@NonNull DatabaseError databaseError) {  
    progressDialog.dismiss();  
    flag = 2;  
    Toast.makeText(getApplicationContext(), "Error retrieving value",  
Toast.LENGTH_SHORT).show();  
  }  
});  
  
  myFirebase.child(path).child("logs").addValueEventListener(new  
ValueEventListener() {  
    @Override
```

```
public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
    StringBuilder logs = new StringBuilder();

    for (DataSnapshot logSnapshot : dataSnapshot.getChildren()) {
        String logType = logSnapshot.child("logType").getValue(String.class);
        Long timestamp = logSnapshot.child("timestamp").getValue(Long.class);

        if (timestamp != null) {
            String dateTime = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss",
Locale.getDefault())
                .format(new Date(timestamp));

            logs.append(logType).append(" at ").append(dateTime).append("\n");
        }
    }

    logText.setText(logs.toString());
}

@Override
public void onCancelled(@NonNull DatabaseError databaseError) {
}

});
}

public void btn_send(View v) {
    if (flag == 1) {
        et1.setError("No records found for this");
        et1.requestFocus();
        return;
    }
    if (flag == 2) {
        et1.setError("Please try again");
    }
}
```



```
        et1.requestFocus();
        return;
    }
    if (flag != 3) {
        return;
    }

    if (ContextCompat.checkSelfPermission(this, Manifest.permission.SEND_SMS) !=
PackageManager.PERMISSION_GRANTED) {
        if (ActivityCompat.shouldShowRequestPermissionRationale(this,
Manifest.permission.SEND_SMS)) {
            } else {
                ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.SEND_SMS},
MY_PERMISSIONS_REQUEST_SEND_SMS);
            }
        } else {
            SmsManager smsManager = SmsManager.getDefault();
            smsManager.sendTextMessage(myChildphone, null, "Hello" + "\t" + myChildname
+ "\t" + "your vehicle" + "\t" + number + "\t" + "Crossed the main gate right now", null,
null);

            Toast.makeText(getApplicationContext(), "SMS sent.",
Toast.LENGTH_LONG).show();
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, String permissions[], int[]
grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        switch (requestCode) {
            case MY_PERMISSIONS_REQUEST_SEND_SMS: {
                if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
```

```
SmsManager smsManager = SmsManager.getDefault();
smsManager.sendTextMessage(myChildphone, null, "Hello" + "\t" +
myChildname + "\t" + "your vehicle" + "\t" + number + "Crossed the main gate right now",
null, null);

Toast.makeText(getApplicationContext(), "SMS sent.",
Toast.LENGTH_LONG).show();

    } else {
        Toast.makeText(getApplicationContext(), "SMS failed, please try again.",
Toast.LENGTH_LONG).show();
    }
}
}
}

public void btn_home(View v) {
    Intent i = new Intent(DriverClass.this, HomeActivity.class);
    startActivity(i);
    finish();
}
}
```

A.2.1.2 Vehicles.java

```
package com.google.android.gms.samples.vision.ocrreader;

public class Vehicles
{

    public String vno,name,phoneNo;
    public Vehicles(String vno,String name,String phoneNo)
    {
        this.vno = vno;
        this.name = name;
        this.phoneNo = phoneNo;
```

```
}  
public Vehicles()  
{  
  
}  
}
```

A.2.1.3 RegisterActivity.java

```
package com.google.android.gms.samples.vision.ocrreader;  
import android.app.ProgressDialog;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.EditText;  
import android.widget.Toast;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
  
import com.google.android.gms.tasks.OnCompleteListener;  
import com.google.android.gms.tasks.Task;  
import com.google.firebase.auth.UserInfo;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
  
public class RegisterActivity extends AppCompatActivity {  
  
    EditText etph,etname,etvn;  
    String phoneNo,name,vno;  
    DatabaseReference databaseReference;  
    ProgressDialog progressDialog;  
  
    private static final int MY_PERMISSIONS_REQUEST_SEND_SMS =0;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_register);
    etph = (EditText) findViewById(R.id.etphn);
    etname= (EditText) findViewById(R.id.etnm);
    etvn = (EditText) findViewById(R.id.etvn);
    databaseReference = FirebaseDatabase.getInstance().getReference("/Registered-
users");
    progressDialog = new ProgressDialog(this);
}

public void btn_register_click(View v)
{

    phoneNo = etph.getText().toString();
    name = etname.getText().toString();
    vno = etvn.getText().toString();
    if(vno.isEmpty())
    {
        etvn.setError("Vehicle number can not be empty");
        etvn.requestFocus();
        return;
    }

    if(name.isEmpty())
    {
        etname.setError("Please enter your name");
        etname.requestFocus();
        return;
    }
    if(phoneNo.isEmpty())
    {
```

```
        etph.setError("Please enter your phone no");
        etph.requestFocus();
        return;
    }

    progressDialog.setMessage("Registering User");
    progressDialog.show();
    saveUserData(vno,name,phoneNo);

}

private void saveUserData(String vno,String name,String phoneNo)
{
    final String phoneNo1 = phoneNo;
    final String name1 = name;
    final String vno1 = vno;
    Vehicles obj = new Vehicles(vno,name,phoneNo);

    databaseReference.child(vno).setValue(obj).addOnCompleteListener(new
    OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            progressDialog.dismiss();
            Toast.makeText(getApplicationContext(),"Stored
            Successfully",Toast.LENGTH_SHORT).show();
        }
    });

}
}
```

A.2.1.4 SplashActivity.java

```
package com.google.android.gms.samples.vision.ocrreader;
```

```
import android.content.Intent;
import android.os.Handler;
import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;
public class SplashActivity extends AppCompatActivity {
    private static int SPLASH_TIMEOUT = 1000;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                Intent homeIntent = new Intent(SplashActivity.this, HomeActivity.class);
                startActivity(homeIntent);
                finish();
            }
        }, SPLASH_TIMEOUT);
    }
}
```

A.2.2 OCR Model

A.2.2.1 OCRCaptureActivity.java

```
package com.google.android.gms.samples.vision.ocrreader;

import android.Manifest;
import android.annotation.SuppressLint;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.Context;
```

```
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.hardware.Camera;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;

import androidx.annotation.NonNull;
import androidx.core.app.ActivityCompat;

import android.telephony.SmsManager;
import android.util.Log;
import android.view.GestureDetector;
import android.view.MotionEvent;
import android.view.ScaleGestureDetector;
import android.view.View;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GoogleApiAvailability;
import com.google.android.gms.samples.vision.ocrreader.ui.camera.CameraSource;
import com.google.android.gms.samples.vision.ocrreader.ui.camera.CameraSourcePreview;
import com.google.android.gms.samples.vision.ocrreader.ui.camera.GraphicOverlay;
import com.google.android.gms.vision.text.TextRecognizer;
import com.google.android.material.snackbar.Snackbar;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ServerValue;
```

```
import com.google.firebase.database.ValueEventListener;

import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.HashMap;
import java.util.Locale;
import java.util.Map;

public final class OcrCaptureActivity extends AppCompatActivity {

    private static final String TAG = "OcrCaptureActivity";
    private static final int RC_HANDLE_GMS = 9001;
    public static String detected_result = null;
    private static final int RC_HANDLE_CAMERA_PERM = 2;
    public static final String AutoFocus = "AutoFocus";
    public static final String UseFlash = "UseFlash";
    public static final String TextBlockObject = "String";

    private CameraSource cameraSource;
    private CameraSourcePreview preview;
    private GraphicOverlay<OcrGraphic> graphicOverlay;
    private ScaleGestureDetector scaleGestureDetector;
    private GestureDetector gestureDetector;
    private TextToSpeech tts;
    DatabaseReference myFirebase, myFirebase2;
    SharedPreferences preferences;

    @Override
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.ocr_capture);

        preview = findViewById(R.id.preview);
        graphicOverlay = findViewById(R.id.graphicOverlay);
        boolean autoFocus = true;
```



```
boolean useFlash = false;
int rc = ActivityCompat.checkSelfPermission(this, Manifest.permission.CAMERA);
if (rc == PackageManager.PERMISSION_GRANTED) {
    createCameraSource(autoFocus, useFlash);
} else {
    requestCameraPermission();
}

gestureDetector = new GestureDetector(this, new CaptureGestureListener());
scaleGestureDetector = new ScaleGestureDetector(this, new ScaleListener());
Snackbar.make(graphicOverlay, " Point at number plate",
    Snackbar.LENGTH_LONG)
    .show();

preferences=getSharedPreferences("lastLogTypes", Context.MODE_PRIVATE);
FirebaseDatabase.getInstance().setPersistenceEnabled(true);
myFirebase = FirebaseDatabase.getInstance().getReference();
}

private void requestCameraPermission() {
    Log.w(TAG, "Camera permission is not granted. Requesting permission");

    final String[] permissions = new String[]{Manifest.permission.CAMERA};

    if (!ActivityCompat.shouldShowRequestPermissionRationale(this,
        Manifest.permission.CAMERA)) {
        ActivityCompat.requestPermissions(this, permissions,
RC_HANDLE_CAMERA_PERM);
        return;
    }
    View.OnClickListener listener = new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            ActivityCompat.requestPermissions(OcrCaptureActivity.this, permissions,
                RC_HANDLE_CAMERA_PERM);
        }
    };
};
```

```
Snackbar.make(graphicOverlay, R.string.permission_camera_rationale,
              Snackbar.LENGTH_INDEFINITE)
              .setAction(R.string.ok, listener)
              .show();
}

@Override
public boolean onTouchEvent(MotionEvent e) {
    boolean b = scaleGestureDetector.onTouchEvent(e);
    boolean c = gestureDetector.onTouchEvent(e);
    return b || c || super.onTouchEvent(e);
}

@SuppressLint("InlinedApi")
private void createCameraSource(boolean autoFocus, boolean useFlash) {
    Context context = getApplicationContext();
    TextRecognizer textRecognizer = new TextRecognizer.Builder(context).build();
    textRecognizer.setProcessor(new OcrDetectorProcessor(graphicOverlay));
    if (!textRecognizer.isOperational()) {
        Log.w(TAG, "Detector dependencies are not yet available.");
        IntentFilter lowstorageFilter = new
IntentFilter(Intent.ACTION_DEVICE_STORAGE_LOW);
        boolean hasLowStorage = registerReceiver(null, lowstorageFilter) != null;
        if (hasLowStorage) {
            Toast.makeText(this, R.string.low_storage_error,
Toast.LENGTH_LONG).show();
            Log.w(TAG, getString(R.string.low_storage_error));
        }
    }

    cameraSource = new CameraSource.Builder(getApplicationContext(), textRecognizer)
        .setFacing(CameraSource.CAMERA_FACING_BACK)
        .setRequestedPreviewSize(1280, 1024)
        .setRequestedFps(15.0f)
        .setFlashMode(useFlash ? Camera.Parameters.FLASH_MODE_TORCH : null)
        .setFocusMode(autoFocus ?
```

```
Camera.Parameters.FOCUS_MODE_CONTINUOUS_VIDEO : null)
        .build();
    }
    @Override
    protected void onResume() {
        super.onResume();
        startCameraSource();
    }
    @Override
    protected void onPause() {
        super.onPause();
        if (preview != null) {
            preview.stop();
        }
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (preview != null) {
        }
    }
    @Override
    public void onRequestPermissionsResult(int requestCode,
                                           @NonNull String[] permissions,
                                           @NonNull int[] grantResults) {
        if (requestCode != RC_HANDLE_CAMERA_PERM) {
            Log.d(TAG, "Got unexpected permission result: " + requestCode);
            super.onRequestPermissionsResult(requestCode, permissions, grantResults);
            return;
        }
        if (grantResults.length != 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
            Log.d(TAG, "Camera permission granted - initialize the camera source");
            // We have permission, so create the camerasource
```

```
        boolean autoFocus = getIntent().getBooleanExtra(AutoFocus, false);
        boolean useFlash = getIntent().getBooleanExtra(UseFlash, false);
        createCameraSource(autoFocus, useFlash);
        return;
    }
    Log.e(TAG, "Permission not granted: results len = " + grantResults.length +
        " Result code = " + (grantResults.length > 0 ? grantResults[0] : "(empty)"));

    DialogInterface.OnClickListener listener = new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
            finish();
        }
    };

    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("Multitracker sample")
        .setMessage(R.string.no_camera_permission)
        .setPositiveButton(R.string.ok, listener)
        .show();
}

private void startCameraSource() throws SecurityException {
    int code = GoogleApiAvailability.getInstance().isGooglePlayServicesAvailable(
        getApplicationContext());
    if (code != ConnectionResult.SUCCESS) {
        Dialog dlg =
            GoogleApiAvailability.getInstance().getErrorDialog(this, code,
RC_HANDLE_GMS);
        dlg.show();
    }
    if (cameraSource != null) {
        try {
            preview.start(cameraSource, graphicOverlay);
        } catch (IOException e) {
            Log.e(TAG, "Unable to start camera source.", e);
            cameraSource.release();
        }
    }
}
```

```
        cameraSource = null;
    }
}

private boolean onTap(float rawX, float rawY) {
    String cleanedResult = detected_result.replaceAll("[^a-zA-Z0-9]", "");

    String path = "Registered-users/" + cleanedResult;
    myFirebase.child(path).child("phoneNo").addListenerForSingleValueEvent(new
com.google.firebase.database.ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            String myChildname = dataSnapshot.getValue(String.class);
            if (myChildname != null) {
                Log.d("number", myChildname);
                checkAndUpdateLogs(cleanedResult, myChildname);
            } else {
                showAlert("Vehicle not registered");
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
            Toast.makeText(OcrCaptureActivity.this, "Failed to fetch data",
Toast.LENGTH_SHORT).show();
        }
    });
    return false;
}

private class CaptureGestureListener extends GestureDetector.SimpleOnGestureListener
{
    @Override
    public boolean onSingleTapConfirmed(MotionEvent e) {
```

```
        return onTap(e.getRawX(), e.getRawY()) || super.onSingleTapConfirmed(e);
    }
}
```

```
private class ScaleListener implements ScaleGestureDetector.OnScaleGestureListener
{
    @Override
    public boolean onScale(ScaleGestureDetector detector) {
        return false;
    }
    @Override
    public boolean onScaleBegin(ScaleGestureDetector detector) {
        return true;
    }
    @Override
    public void onScaleEnd(ScaleGestureDetector detector) {
        if (cameraSource != null) {
            cameraSource.doZoom(detector.getScaleFactor());
        }
    }
}
```

```
private void checkAndUpdateLogs(String vehicleNumber, String phoneNumber) {
    String path = "Registered-users/" + vehicleNumber;
    DatabaseReference logsRef = myFirebase.child(path).child("logs");
    logsRef.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if (dataSnapshot.exists() && dataSnapshot.getChildrenCount() > 0) {
                DataSnapshot lastLogSnapshot = dataSnapshot.getChildren().iterator().next();
                String lastLogType = lastLogSnapshot.child("logType").getValue(String.class);
                String lastLogTypeFromSharedPreferences =
getLastLogTypeFromSharedPreferences(vehicleNumber);
                Log.d("logType", lastLogTypeFromSharedPreferences);
            }
        }
    });
}
```

```
        if ("entry".equals(lastLogTypeFromSharedPreferences)) {
            createLog(logsRef, "exit");
            sendSMS(phoneNumber, "Vehicle Exited");
            saveLogTypeToSharedPreferences(vehicleNumber, "exit");
            nextPage(vehicleNumber);
        } else {
            createLog(logsRef, "entry");
            sendSMS(phoneNumber, "Vehicle Entered");
            saveLogTypeToSharedPreferences(vehicleNumber, "entry");
            nextPage(vehicleNumber);
        }
    } else {
        createLog(logsRef, "entry");
        sendSMS(phoneNumber, "Vehicle Entered");
        Log.d("logType", "entering");
        saveLogTypeToSharedPreferences(vehicleNumber, "entry");
        nextPage(vehicleNumber);
    }
}

@Override
public void onCancelled(@NonNull DatabaseError databaseError) {
    Toast.makeText(OcrCaptureActivity.this, "Failed to fetch logs",
Toast.LENGTH_SHORT).show();
}

});
}

private void createLog(DatabaseReference logsRef, String logType) {
    String logKey = logsRef.push().getKey();
    Map<String, Object> logData = new HashMap<>();
    logData.put("logType", logType);
    logData.put("timestamp", ServerValue.TIMESTAMP);
    logsRef.child(logKey).setValue(logData);
}

private void sendSMS(String phoneNumber, String message) {
```

```
try {
    SmsManager smsManager = SmsManager.getDefault();
    smsManager.sendTextMessage(phoneNumber, null, message, null, null);
} catch (Exception e) {
    e.printStackTrace();
}
}

private void saveLogTypeToSharedPreferences(String vehicleNumber, String logType) {
    SharedPreferences.Editor editor = preferences.edit();
    editor.putString(vehicleNumber, logType);
    editor.apply();
}

private String getLastLogTypeFromSharedPreferences(String vehicleNumber) {
    return preferences.getString(vehicleNumber, "entry");
}

private void nextPage(String vehicleNumber) {
    Intent intent = new Intent(this, DriverClass.class);
    intent.putExtra("5555", vehicleNumber);
    startActivity(intent);
}

private void showAlert(String message) {
    AlertDialog.Builder builder = new AlertDialog.Builder(OcrCaptureActivity.this);
    builder.setTitle("Alert")
        .setMessage(message)
        .setPositiveButton("OK", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
            }
        })
        .show();
}
}
```


APPENDIX-B

SCREENSHOTS

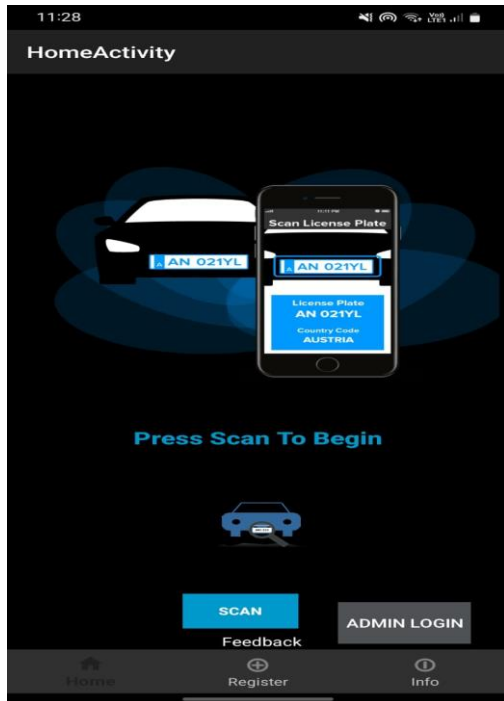


Figure B.1 Home Page

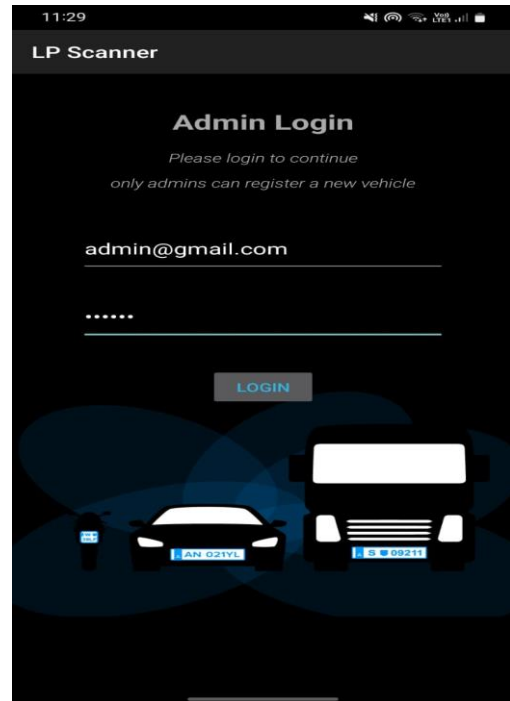


Figure B.2 Admin Login Page

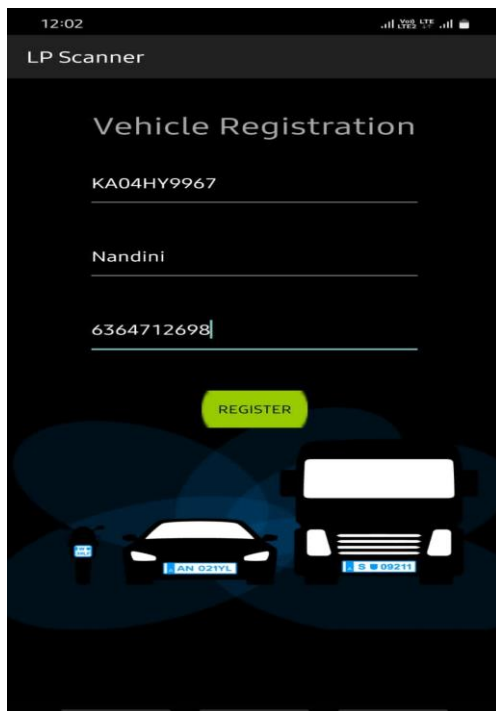


Figure B.3 Vehicle Registration Page



Figure B.4 Scanning of Number Plate

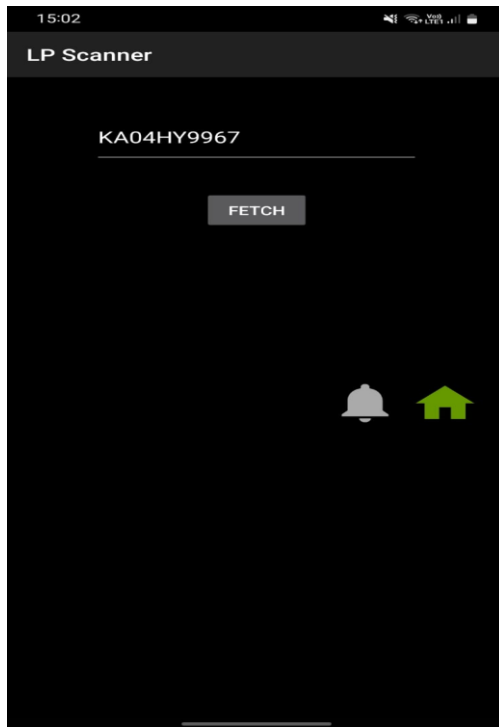


Figure B.5 Fetching data from database

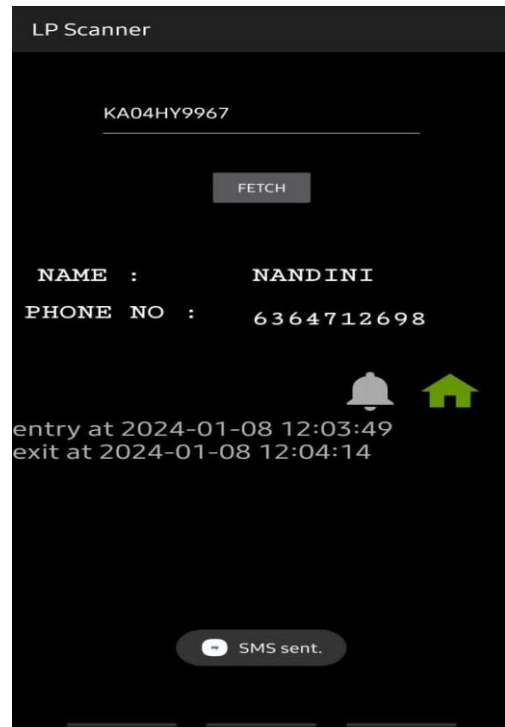


Figure B.6 Entry and Exit Log

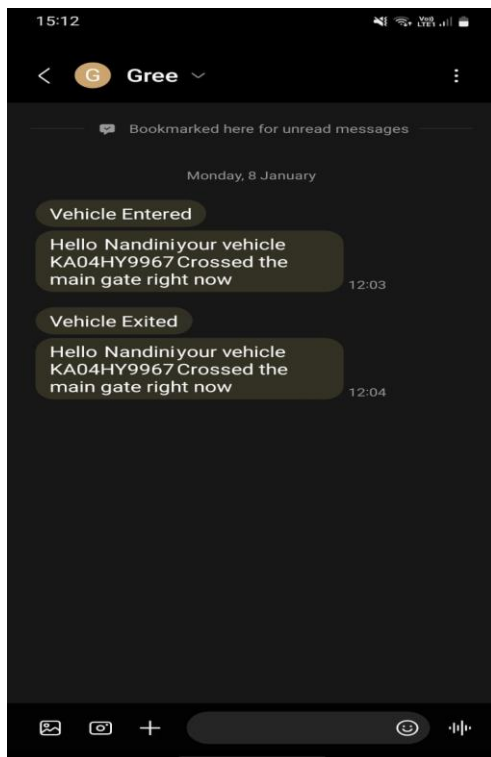


Figure B.7 SMS Notification

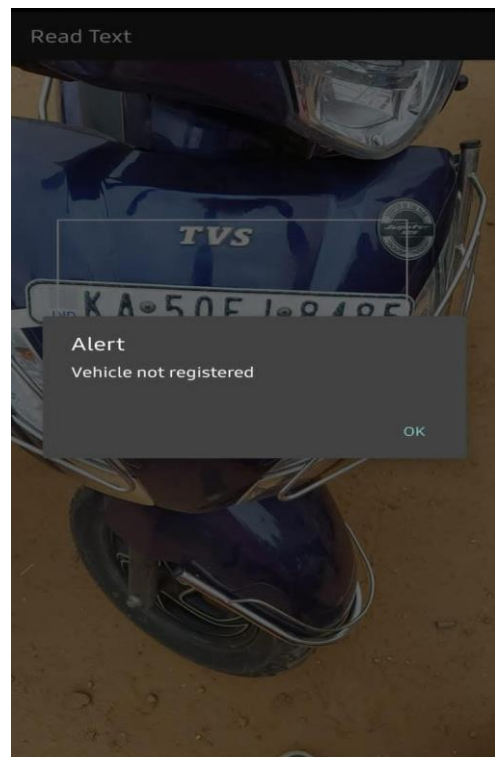


Figure B.8 Alert message for unregistered vehicle

APPENDIX-C

ENCLOSURES

1.Conference Paper Presented Certificates of all students.



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

MANOJ YADAV N

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

Published a paper entitled

**"Automated Vehicle Entry and Exit Management for
Residency"**

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

GREESHMA S DEVADIGA

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

Published a paper entitled

**"Automated Vehicle Entry and Exit Management for
Residency"**

in IJIRCCE, Volume 12, Issue 1, January 2024



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



www.ijircce.com ijircce@gmail.com

2. Similarity Index/Plagiarism Check report clearly showing the Percentage(%). No need of page-wise explanation.

ORIGINALITY REPORT

5%

SIMILARITY INDEX

4%

INTERNET SOURCES

0%

PUBLICATIONS

3%

STUDENT PAPERS

PRIMARY SOURCES

1

Submitted to Presidency University

Student Paper

3%

2

www.ijraset.com

Internet Source

2%

Exclude quotes Off

Exclude matches Off

Exclude bibliography Off

3. SDG Goals Mapping.



This project directly supports SDG 11, "Sustainable Cities and Communities," by employing technology to enhance urban safety and mobility. Through features like vehicle tracking, communication logs, and real-time alerts, it promotes efficient transportation and contributes to the development of safer and smarter urban environments. By addressing key aspects of SDG 11, the project actively works towards creating sustainable and resilient cities for present and future generations.