Majlis Arts and Science College, Puramannur

(Affiliated to the University of Calicut)

PROJECT REPORT

On

DRIVER VIGILANCE SYSTEM

Submitted in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF COMPUTER APPLICATIONS

UNIVERSITY OF CALICUT

Submitted By:

MOHAMMED RASEEN M

(MSAVBCA049)

Guided By:

Mrs. MANJULA H

Assistant Professor

Department of Computer Science

2023-2024

MAJLIS ARTS AND SCIENCE COLLEGE PURAMANNUR

(Affiliated to Calicut University)

DEPARTMENT OF COMPUTER SCIENCE

Certificate

This is to certify that the project entitled **DRIVER VIGILANCE SYSTEM** submitted in the partial fulfillment of the requirement for the award of the Degree of Bachelor of Computer Applications, University of Calicut, is a Bonafide record of the project work done by **MOHAMMED RASEEN M (MSAVBCA049),** during the academic year 2023-2024 in the Department of Computer Science, Majlis Arts and Science College, Puramannur under my supervision and guidance.

Mrs. MANJULA H	**Dr. MOHAMED KUTTY K K**	**Mr.AJAYA KUMAR. U**
Assistant professor	Principal	Head of department
Majlis arts and Science College		Department of Computer Science

Submitted for the University exam held on …………

Internal Examiner	External Examiner

DECLARATION

I hereby declare that this project work entitled **DRIVER VIGILANCE SYSTEM** submitted to the University of Calicut in partial fulfillment of requirements for the award of Bachelor of Computer Application is a record of original work done by **MOHAMMED RASEEN M (MSAVBCA049)** under the guidance of **Mrs.MANJULA H**, Assistant Professor, Department of Computer Science, Majlis Arts and Science College Puramannur.

Signature of students

MOHAMMED RASEEN M
(MSAVBCA049)

ACKNOWLEDGEMENT

With profound sense of gratitude, we wish to express my sincere thanks to **Dr. MOHAMED KUTTY K K**, Principal, Majlis Arts and Science College for giving an opportunity to undertake this project.

I have great pleasure in expressing my profound gratitude to **Mr. Ajaya Kumar U**, Head of the Department of Computer Science for giving the valuable help during the project work.

I gratefully acknowledge my internal guide **Mrs.Manjula H**, Asst professor, Department of Computer Science, Majlis Arts and Science College for her encouragement till the end of the project.

Last but not least, I would like to express my sincere thanks to the God almighty for his constant love and grace that he has bestowed upon us. Finally, I thank my parents, family members and beloved friends for their moral support and encouragement without which I have not been able to follow my dreams.

CONTENTS

1. INTRODUCTION

In recent years, the increasing number of vehicles on roads leads to an increase in traffic accidents. It is found that distracted driving was related to one-tenth of fatal crashes. Distracted driving fatalities have increased more rapidly than those caused by drunk driving, speeding and failing to wear a seat belt. A driver is considered to be distracted when there is an activity that attracts his/her attention away from the task of driving. Distracted driving is an established cause of motor vehicle crashes for all ages. With the rapidly growing elderly population and more adults embracing technology, distracted driving is also increasing in prevalence within that population—particularly
cell phone usage behind the wheel. Another cause of accidents is drowsiness during driving. Studies show that around one quarter of all serious motorway accidents is attributable to sleepy drivers in need of a rest, meaning that drowsiness causes more road accidents than drink-driving. **—DRIVER VIGILANCE SYSTEM‖** can identify the drowsiness of the person driving and alert them. The system will be a blessing to the people who are driving and also people who are walking on the roads and also, we can find locations, block zones and Signals in our areas.

The problem addressed by the Driver Vigilance System project is the alarming rate of road accidents caused by drowsy and inattentive driving. Drowsiness and distraction significantly impair a driver's ability to respond effectively to changing road Conditions, posing a grave threat to road safety. The project aims to develop a robust computer vision-based solution using Python that can accurately detect signs of driver
drowsiness, such as eyelid drooping or frequent blinking, in real time. By doing so, the
system will provide timely alerts to the driver, thereby preventing potential accidents and fostering safer driving behavior. The project seeks to tackle this critical issue by harnessing the power of technology to enhance driver vigilance, reduce accidents, and ultimately save lives on the road.

The Driver Vigilance System project addresses the pressing concern of road safety by targeting the risks associated with drowsy and distracted driving. With an increasing number of accidents attributed to drivers' reduced alertness and attention on
the road, there is a critical need for an intelligent system that can detect these behaviors
in real time. The project seeks to develop an innovative solution using Python that employs computer vision techniques to monitor and analyze facial cues, such as eye movements and expressions, to determine instances of drowsiness or distraction. By accurately identifying these hazardous states, the system aims to provide timely warnings and alerts to the driver, thereby significantly reducing the likelihood of accidents caused by impaired focus or fatigue. The project's ultimate goal is to contribute to road safety by leveraging Python's capabilities to create a proactive and reliable Driver Vigilance System that enhances driver awareness and prevents potential
collisions.

1.1 ABSTRACT

Drowsiness and Fatigue of drivers are amongst the significant causes of road accidents. Every year, they increase the amounts of deaths and fatalities injuries globally. Drivers who do not take regular breaks when driving long distances run a high risk of becoming drowsy a state which they often fail to recognize early enough according to the experts. In addition, activities such as usage of phone, turning around, picking something from rear seat during driving can lead to accidents. Detection of such activities on time can help in reducing accidents to an extent. To overcome this issue, I propose a DRIVER VIGILANCE SYSTEM that has the potential to be implemented in real vehicles. The proposed work focuses on driver distraction activities detection via images using different kinds of machine learning techniques. The input of my model is videos of driver taken in the car.

2. SYSTEM STUDY

2.1 PROBLEM DEFINITION AND INITIAL REQUIREMENTS

Car accident is the major cause of death in which around 1.3 million people die every year. Majority of these accidents are caused because of the drowsiness of driver. The countless number of people drives for long distance every day and night on the highway may lead to an accident. To prevent such accidents we propose a system which alerts the driver if the driver gets drowsy. Facial detection is used with help of image processing of images of the face captured using the camera.Even though security systems for passenger safety are provided by major motor companies, there exists no system that can detect the driver distraction completely. These security systems are partial, such as, speed sensing, distraction alert, call blocking, screen light intensity settings etc.

2.2 BASIC FUNCTIONALITIES OF THE PROJECT

The basic functionalities of the project are as follows:

- Driver is notified when he receives new messages from his partners.
- Messages are read out using Google’s Text to Speech engine which is built-in feature in smart phones.

  The android application captures images from video captured using phone camera. These images are processed to find eye regions, and to detect whether eyes are open or not. If the eyes are closed for more than a set threshold, a beep sound is played to alert driver and wake him up.

  Similarly, the camera can identify the drivers pose changes to detect whether he / she is distracted from driving, by some activities. If distractions are found, the application alerts the driver to focus on driving.
- Drive now text later implements safety feature while taking phone when driving

2.3 PROPOSED SYSTEM

The main objective of the proposed system is to detect drowsiness in drivers and alert them on time to avoid occurrence of accidents. In addition, activities such as usage of phone, turning around, picking something from rear seat during driving can lead to accidents. Detection of such activities on time can help in reducing accidents to an extent. To overcome this issue, we propose a distraction detection system that has the potential to be implemented in real vehicles. The proposed work focuses on driver distraction activities detection via images using different kinds of machine learning techniques. The input of our model is videos of driver taken in the car.

3. System Analysis

System Analysis is concerned with analyzing, designing, implementing and evaluating information system. It is carried out to make the system more effective either by modification or by substantial redesign. In system analysis we identify the problem, study the alternative solution and select the most suitable solution, which meet the technical economic and social demands for analysis, various tools such as dataflow diagram (DFD), interviews on site observation, questionnaires etc., are used.

System analysis process is also called a life cycle methodology since it relates to four significant phases in life cycle of all information system. They are

1. System Analysis / Study Phase.
2. System Design / Design phase.
3. System Development / Development Phase.
4. Testing and implementation / Operation Phase.

All activities associated with each life cycle phase must be performed managed and documented. So, system analysis is the performance, management and documentation of the activities related to the four life cycle phases of a computer-based system.

3.1 Feasibility Study

Feasibility is a test of system according to workability, impact on organization ability to meet user needs, and effective use of resources. Following are the feasibility study employed,

- Technical Feasibility
- Economic Feasibility
- Operational Feasibility

3.1.1 Technical Feasibility

A study of function, performance and constraints may improve the ability to create an acceptable system. Technical Feasibility is frequently the most difficult area to achieve at the stage of product engineering process. Considering that are normally associated with the technical feasibility include,

- Development risk
- Resource availability
- Technology

Technical Feasibility study deals with the hardware as well as software requirements. This project Driver Drowsiness Detection is platform independent since it is being coded in JAVA language (using JSP). MY SQL database will be used for storing data. Hardware requirements used are compatible with all OS. Only authorized person would be able to use website so it would be secure. The scope was whether the work for the project is done with the current requirements and existing software technology has to be examined in the feasibility study. The outcome was found to be positive. In the proposed system, data can be easily stored and managed using database management system software. The reports and results for various queries can be generated easily. Therefore, the system is technically feasible.

3.1.2 Economic Feasibility

Nowadays more accident occurs in trucks and cars than vehicles due to drowsiness. Nearly 97% of crashes of vehicles happen due to drowsiness of driver. It results into loss for e.g., human loss, money loss, medical loss. The accident or crashes not only affect the internal system but also to outside world. 70% injury occurs in internal system and 30% injury happen to the external system. Environmental loss is one of disadvantage of accident. Accidents results in human as well as non-human loss. The main objective of this paper is the design and implementation of a hardware system that is able to detect drowsiness of drivers, especially those diagnosed at the right time to alert. This will prevent many accidents and save countless lives, and reduces the high cost of damages caused by accidents.

Operational Feasibility

Proposed projects are beneficial only if they can be turned into information system that will meet the organization’s operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are these major barriers to implementation?

The purpose of the operational feasibility study is to determine whether the new system will be used if it is developed and implemented from users that will undermine the possible application benefits. Since the proposed system is an android application, the operations are mainly concerned and are completely depended on the users. The application is developed by giving prime importance to the ease with which the end users can operate on the system. If the user of the system is fully aware of the internal working of the system, then the users will not be facing any problem in running the system.

4. SYSTEM REQUIREMENTS

4.1 SYSTEM REQUIREMENT

The system study involves the identification of objectives and requirements, evaluation of the alternative solutions and recommendation for more feasible solution. Requirement analysis is the process of identifying the system requirements through the observation of existing system discussion with potential users and procedures, task analysis etc. A perfect requirement analysis removes all ambiguities and inconsistencies from the initial customer perception of the problem. The software should meet the functional requirement and perform the functionality effectively and efficiently.

4.2 SOFTWARE REQUIREMENT

For the proposed system to work properly, it is necessary that following software are installed and running on the server / client.

- FRONT END : PYTHON(FLASK), ANDROID
- BACK END : My SQL
- SOFTWARE USED : PYCHARM, ANDRIODSTUDIO, SQLYOG, WAMP/XAMP SERVER
- OPERATING SYSTEM : WINDOWS 8/10 for better performance
- WEB BROWSER : Internet Explorer/Google Chrome/Firefox

4.3 HARDWARE REQUIREMENT

It is recommended that for optimal performance, the following minim minimum hardware is installed on the server on which the portal is hosted, as well as on clients that access the portal.

- PROCESSOR : Intel core i3 and above
- MONITOR : LCD Display
- MEMORY : Min 4 GB RAM
- MOBILE PHONE : Android(Version above 5.1)

5. SYSTEM DESIGN

5.1 MODULE DESCRIPTION

The proposed system is divided into three parts,

1. Driver App
2. Partner App
3. Website for synchronising both app
4. Camera based driver vigilance control

Functions of Driver App

1. Registration
2. Login
3. Partner Setting
4. Auto messaging to own contacts
5. Location Sharing
6. Driving pattern detection using mobile sensors
7. Voice assistance support
8. Camera assistance while closing eye lids(sleeping)
9. Get road condition notifications using GPS, accelerometer

Functions Of partner app

10. Login
11. View drivers (partners)
12. Driver phone mode settings
    a. Speed based
    b. Vehicle angle based

Main settings are,

1. Call blocking
2. Screen light intensity setting
3. Screen lock while speed increases
4. Auto smsing service
5. Ring mode settings

13. View Driving logs
14. Messaging to partner
15. View Locations

Functions of Website:

1. Controls and manage the data flow between partner and driver

Function of driver viglilance camera app

The proposed work focuses on driver distraction activities detection via images using different kinds of machine learning techniques. The input of our model is videos of driver taken in the car.

1. Training part
2. Testing part
3. Prediction
4. Evaluation of our model

5.2 ABOUT THE LANGUAGE

Android:

Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on game consoles, digital cameras, PCs and other electronics. Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, along with the founding of the Open Handset Alliance – a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. Beginning with the first commercial Android device in September 2008, the operating system has gone through multiple major releases, with the current version being 8.0 "Oreo", released in August 2017. Android applications ("apps") can be downloaded from the Google Play store, which features over 2.7 million apps as of February 2017. Android has been the best-selling OS on tablets since 2013, and runs on the vast majority of smartphones. As of May 2017, Android has two billion monthly active users, and it has the largest installed base of any operating system. Android's source code is released by Google under an open-source license, although most Android devices ultimately ship with a combination of free and open source and proprietary software, including proprietary software required for accessing Google services. Android is popular with technology companies that require a ready-made, low-cost and customizable operating system for high-tech devices. Its open nature has encouraged a large community of developers and enthusiasts to use the opensource code as a foundation for community-driven projects, which deliver updates to older devices, add new features for advanced users or bring Android to devices originally shipped with other operating systems. The extensive variation of hardware in Android devices causes significant delays for software upgrades, with new versions of the operating system and security patches typically taking months before reaching consumers, or sometimes not at all. The success of Android has made it a target for patent and copyright litigation between technology companies.

PYTHON:

Python is a high-level, versatile, dynamically-typed and widely-used programming language known for its readability, simplicity, and extensive libraries. Created by Guido van Rossum in the late 1980s, Python has gained immense popularity due to its focus on code readability and clean syntax, making it an excellent choice for both beginners and experienced developers. Python is widely used for various purposes, including web development, data analysis, scientific computing, automation, and more.

Here's an overview of some key aspects of Python programming:

Key Features of Python:

1. Readability and Syntax: Python is known for its clean and readable syntax. It uses indentation to define code blocks, which makes the code visually appealing and easier to understand. Python emphasizes clear and concise code, using indentation for block structure instead of traditional braces.

2. Versatility: Python is a versatile language that can be used for a wide range of applications, including web development, desktop applications, data analysis, machine learning, artificial intelligence, scientific computing, and more.

3. Extensive Standard Library: Python comes with a large standard library that provides modules for various tasks such as web development, data manipulation, and scientific computing; and functions for various tasks, such as file manipulation, networking, regular expressions, and more. This eliminates the need to reinvent the wheel and makes development faster and more efficient.

4. Third-Party Libraries and Packages: Python has a rich ecosystem of third-party libraries and packages available through the Python Package Index (PyPI). These libraries extend Python's capabilities and enable developers to work on specific domains or tasks.

5. Object-Oriented Programming (OOP): Python supports object-oriented programming (OOP) principles, allowing you to create and work with classes, objects, and inheritance.

6. Interpreted Language: Python is an interpreted language, which means that code is executed line by line by an interpreter. This makes development and testing quicker as changes can be tested immediately.

7. Dynamic Typing: Python is dynamically typed, meaning that variable types are determined at runtime. You don't need to declare the type of a variable explicitly, making the code more flexible.

8. Community and Resources: Python has a large and active community of developers, which means there are plenty of resources, tutorials, forums, and opensource projects available. This community support makes it easier to learn and troubleshoot Python-related issues.

9. Cross-Platform Compatibility: Python is available for various platforms, including Windows, macOS, and various Linux distributions. This allows you to write code once and run it on different operating systems.

10. Web Development Frameworks: Python has popular web development frameworks like Flask and Django, which simplify the process of building web applications by providing tools and structures for handling routing, templates, databases, and more.

11. Data Science and Machine Learning: Python is widely used in data science and machine learning due to libraries like NumPy, pandas, scikit-learn, and TensorFlow. These libraries provide tools for data manipulation, analysis, visualization, and building machine learning models. Overall, Python's simplicity, versatility, and strong community support make it an excellent choice for programmers of all levels and for a wide range of applications. Whether you're a beginner or an experienced developer, Python offers a comfortable environment to write efficient, clean, and readable code.

JAVA

Java is a high-level, object-oriented programming language developed by Sun Microsystems (now owned by Oracle) in the mid-1990s. It is designed to be platform-independent, meaning that Java programs can run on any device or operating system that has a Java Virtual Machine (JVM). Here are some key points about Java:

1. Platform Independence: Java achieves platform independence through its "write once, run anywhere" philosophy. This means that once a Java program is compiled into bytecode, it can be executed on any device or platform that supports the Java Virtual Machine (JVM).
2. Object-Oriented: Java is fundamentally based on the principles of object-oriented programming (OOP). It supports features such as classes, objects, inheritance, polymorphism, and encapsulation, making it well-suited for building modular and scalable applications.
3. Rich Standard Library: Java comes with a vast standard library (Java API) that provides ready-to-use classes and methods for various tasks such as input/output operations, networking, database connectivity, GUI development, and more. This helps developers save time and effort in coding common functionalities.
4. Memory Management: Java manages memory allocation and deallocation automatically through its garbage collection mechanism. This relieves developers from manual memory management tasks, reducing the risk of memory leaks and improving application stability.
5. Platform for Enterprise Applications: Java is widely used in enterprise environments for developing robust, scalable, and secure applications. Technologies like Java EE (Enterprise Edition) provide frameworks and APIs for building large-scale enterprise applications, web services, and distributed systems.
6. Community and Ecosystem: Java has a thriving developer community and a vast ecosystem of tools, frameworks, libraries, and resources. This ecosystem continues to evolve, ensuring that Java remains relevant and adaptable to modern software development trends.

Overall, Java's versatility, portability, strong OOP support, rich standard library, and widespread adoption make it a popular choice for a wide range of software development projects, from desktop and web applications to mobile apps and enterprise solutions.

MySQL: -

MySQL, the most popular Open-Source SQL database management system, is developed, distributed, and supported by Oracle Corporation.

MySQL is a database management system.

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

MySQL databases are relational.

A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one-to-one, one-tomany, unique, required or optional, and ―pointers‖ between different tables. The database enforces these rules, so that with a well-designed database, your application never sees inconsistent, duplicate, orphan, out-of-date, or missing data.

The SQL part of ―MySQL‖ stands for ―Structured Query Language‖. SQL is the most common standardized language used to access databases. Depending on your programming environment, you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language, or use a language specifi

API that hides the SQL syntax. SQL is defined by the ANSI/ISO SQL Standard.
The SQL standard has been evolving since 1986 and several versions exist

MySQL software is Open Source.
Open Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything. If you wish, you may study the source code and change it to suit your needs.

The MySQL software uses the GPL (GNU General Public License), to define what you
may and may not do with the software in different situations.

The MySQL Database Server is very fast, reliable, scalable, and easy to use.
MySQL Server can run comfortably on a desktop or laptop, alongside your other applications, web servers, and so on, requiring little or no attention. If you dedicate an entire machine to MySQL, you can adjust the settings to take advantage of all the memory, CPU power, and I/O capacity available. MySQL can also scale up to clusters of machines, network together.

SQLyog:

- SQLyog provides you with powerful means to manage your MySQL databases.
- Runs on all Windows version from Win XP to Win 8.x (desktop systems) as well as "Windows Server" systems of same generations (Windows Server 2003 and higher)
- MySQL 5.x compatible
- Create/Drop/Alter Tables, Stored Procedures, Functions, Views, Triggers and Events

- HTTP and SSH Tunneling - smartly manage your MySQL server even if the MySQL port is blocked or remote access to MySQL is disallowed!
- Protect your data with SSL encryption.
- Smart AutoComplete.
- Formats SQL statements.
- Proactive Query Profiler.
- Favorite Manager to neatly organize your favorite SQL statements.
- Very fast retrieval of data.
- Advanced GUI Query Builder. Supports JOINs, aggregate as well as 'common' functions, sorting (ORDER BY) and filtering (WHERE and HAVING) and ALIAS.
- SQLyog Import External Data wizard - use the GUI or specify a query.
- Schema and Data synchronization tools.
- Powerful compressed Scheduled Backup with email notification.
- Schedule various jobs.

SQL Commands: -

SQL commands are instructions, coded into SQL statements, which are used communicate with the database to perform specific tasks, work, functions and queries with data. SQL commands can be used not only for searching the database but also to perform various other functions like, for example, you can *create tables*, add data to tables, or modify data, drop the table, set permissions for users. SQL commands are grouped into four major categories depending on their functionality:

Data Definition Language (DDL)

These SQL commands are used for creating, modifying, and dropping the structure of database objects. The commands are CREATE, ALTER, DROP, RENAME, and TRUNCATE.

Data Manipulation Language

These SQL commands are used for storing, retrieving, modifying, and deleting data. These Data Manipulation Language commands are: SELECT, INSERT, UPDATE, and DELETE.

Transaction Control Language (TCL)

These SQL commands are used for managing changes affecting the data. These commands are COMMIT, ROLLBACK, and SAVEPOINT.

Data Control Language (DCL)

These SQL commands are used for providing security to database objects. These commands are GRANT and REVOKE

5.3 Data Flow Diagram (DFD)

Data Flow Diagram (DFD) is a graphical representation of detain an information system, illustrating how inputs are transformed into outputs through processes and data stores. It is a visual tool used in system analysis and design to depict the movement and processing of data within an information system. It is capable of depicting incoming data flow, outgoing data flow, and stored data. The DFD does not mention anything about how data flows through the system.

Data Flow Diagram Components:

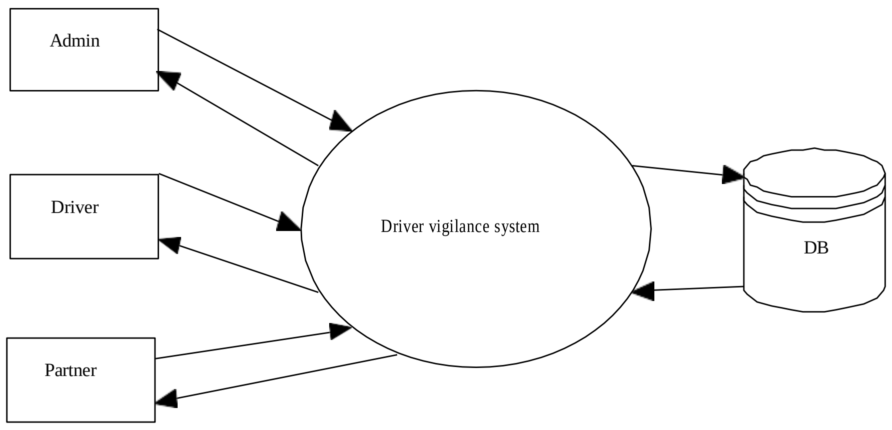
DFD Components DFD can represent source, destination, storage, and flow of data using the following set of components.

- **Entities** - it Represented by squares; external entities denote sources or destinations of data that are external to the system being modelled. These can include users, other systems, or external data sources.
- **Processes** - Represented by circles or ovals, processes in a DFD denote activities or transformations that take place within the system. These processes are responsible for manipulating input data to produce output.
- **Data Stores** - Represented by rectangles, data stores symbolize where data is persisted or stored within the system. They can represent databases, files, or any other storage medium.
- **Data Flow** - Represented by arrows, data flows indicate the movement of data between processes, data stores, and external entities. Arrows show the direction of data movement, from a source to a destination.

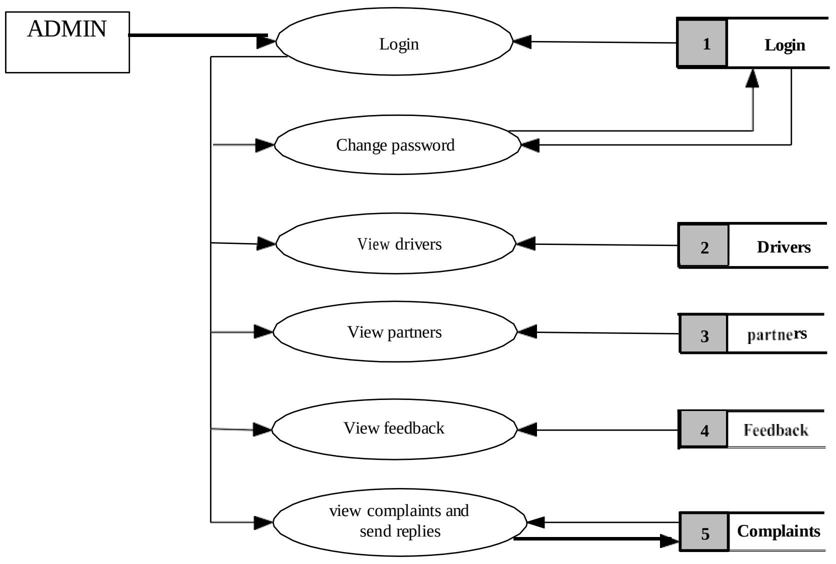
<u>DFD SYMBOLS</u>

Entity	▭
Process	⬭
Data Storage	⊏
Data Flow	→ ←

<u>LEVEL 0</u>

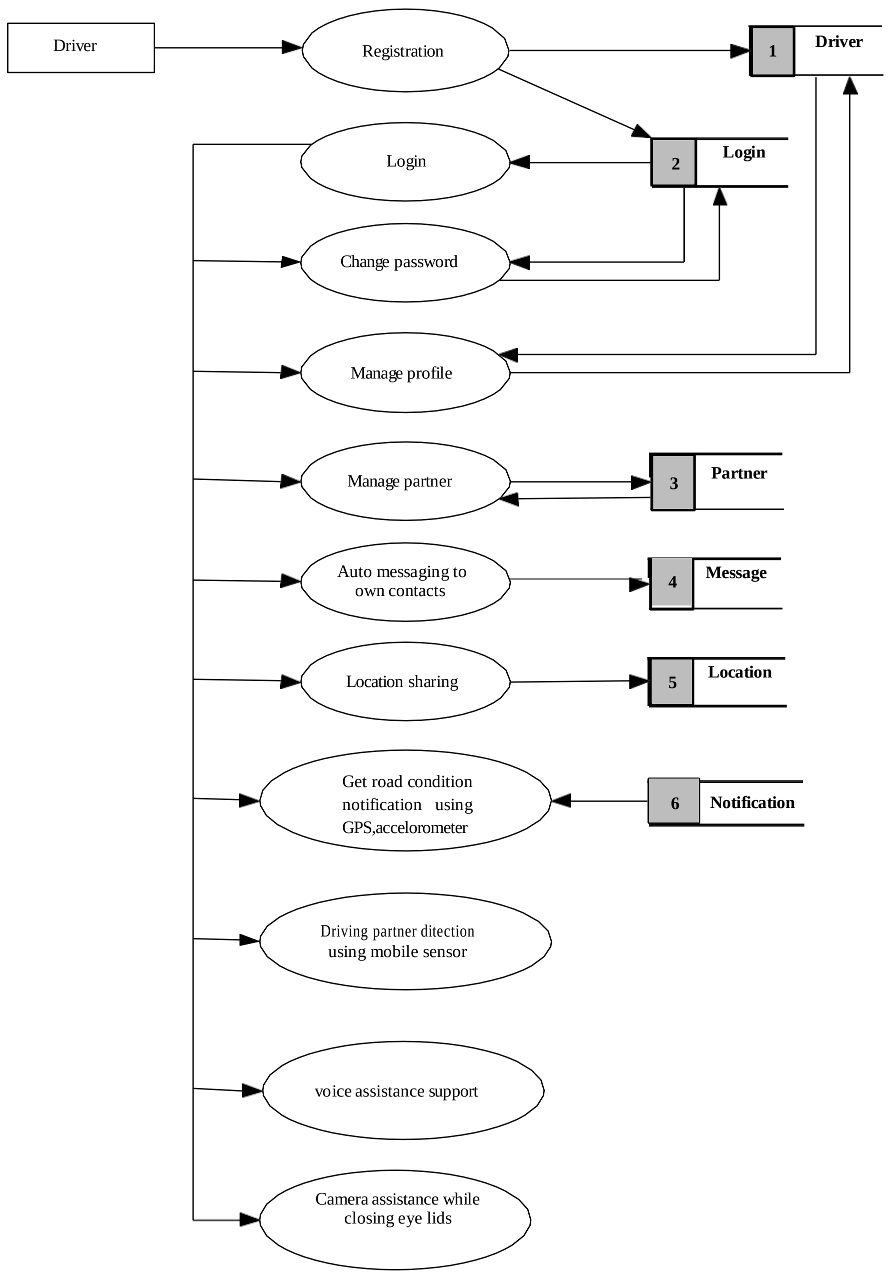


LEVEL 1

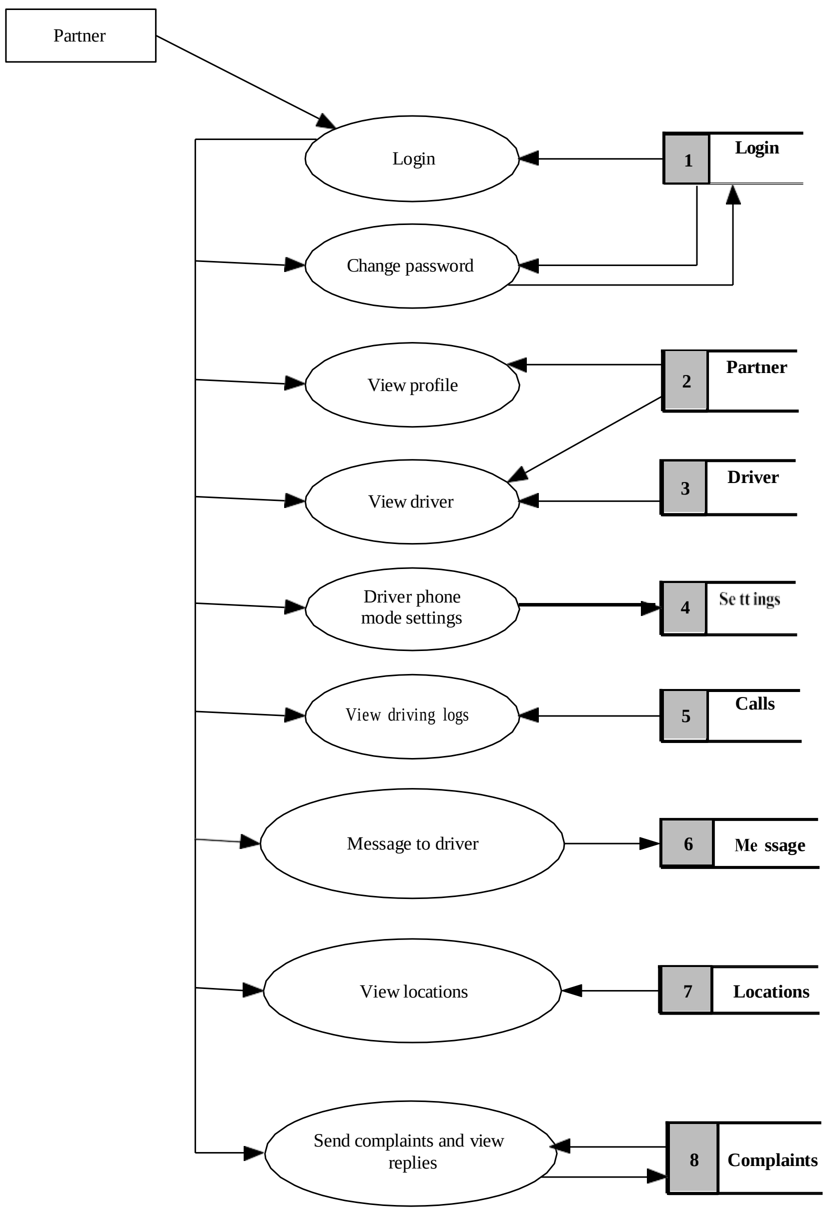


ADMIN
Login
Change password
View drivers
View partners
View feedback
view complaints and send replies
1 Login
2 Drivers
3 partners
4 Feedback
5 Complaints

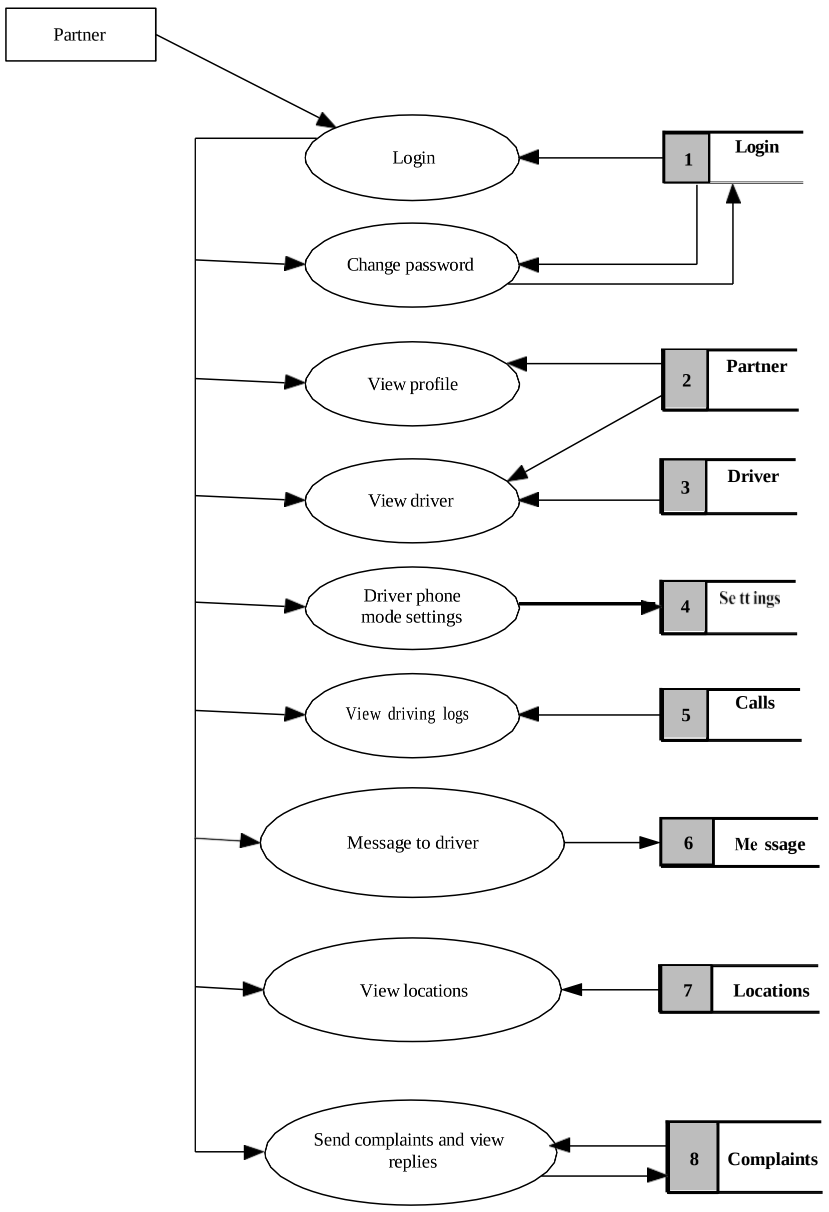
LEVEL 2


Driver
Registration
1
Driver
Login
2
Login
Change password
Manage profile
Manage partner
3
Partner
Auto messaging to own contacts
4
Message
Location sharing
5
Location
Get road condition notification using GPS,accelorometer
6
Notification
Driving partner ditection using mobile sensor
voice assistance support
Camera assistance while closing eye lids

LEVEL 3


Partner
Login
Change password
View profile
View driver
Driver phone mode settings
View driving logs
Message to driver
View locations
Send complaints and view replies
1 Login
2 Partner
3 Driver
4 Se tt ings
5 Calls
6 Me ssage
7 Locations
8 Complaints

LEVEL 4



Partner
Login
Change password
View profile
View driver
Driver phone mode settings
View driving logs
Message to driver
View locations
Send complaints and view replies
1 Login
2 Partner
3 Driver
4 Settings
5 Calls
6 Message
7 Locations
8 Complaints

5.4 DATABASE TABLES

LOGIN

Column name	Data type	Size	Constraints
ID	Int	11	Primary key,Not null
User name	Varchar	100	Not null
Password	Varchar	100	Not null
Type	Varchar	20	Not null

DRIVERS

Column name	Data type	Size	Constraints
ID	Int	11	Primary key,Not null
Name	Varchar	100	Not null
Photo	Varchar	300	Not null
Email	Varchar	100	Not null
Phone	Big int	20	Not null
Gender	Varchar	100	Not null
House name	Varchar	100	Not null
Place	Varchar	100	Not null
PIN	Int	11	Not null
District	Varchar	100	Not null
LOGIN	Int	11	Foreign Key,Not null

PARTNERS

Column name	Data type	Size	Constraints
ID	Int	11	Primary key,Not null
Partner name	Varchar	100	Not null
Photo	Varchar	300	Not null
Email	Varchar	100	Not null
Phone	Big int	20	Not null
Gender	Varchar	100	Not null
House name	Varchar	100	Not null
Place	Varchar	100	Not null
PIN	Int	11	Not null
District	Varchar	100	Not null
LOGIN	Int	11	Foreign key,Not null
DRIVER	Int	11	Foreign key,Not null

FEEDBACK

Column name	Data type	Size	Constraints
ID	Int	11	Primary key,Not null
Date	Date		Not null
Feedback	Varchar	300	Not null
DRIVER	Int	11	Foreign key, Not null

COMPLAINTS

Column name	Data type	Size	Constraints
ID	Int	11	Primary key,Not null
Date	Date		Not null
Complaints	Varchar	200	Not null
Reply	Varchar	200	Not null
Status	Varchar	100	Not null
Type	Varchar	100	Not null
LOGIN	Int	11	Foreign key,Not null

MESSAGE

Column name	Data type	Size	Constraints
ID	Int	11	Primary key,Not null
Date	Date		Not null
Message	Varchar	150	Not null
DRIVER	Int	11	Foreign key,Not null
PARTNER	Int	11	Foreign key,Not null

LOCATION

Column name	Data type	Size	Constraints
ID	Int	11	Primary key,Not null
Date	Date		Not null
Latitude	Varchar	15	Not null
Longitude	Varchar	15	Not null
DRIVER	Int	11	Foreign key,Not null

NOTIFICATION

Column name	Data type	Size	Constraints
ID	Int	11	Primary key,Not null
Date	Date		Not null
Notification	Varchar	200	Foreign key,Not null

SETTINGS

Column name	Data type	Size	Constraints
ID	Int	11	Primary key,Not null
DRIVER	Int	11	Foreign key,Not null
Brightness	Int	3	Not null
Touch block	Varchar	5	Not null
Auto SMS	Varchar	5	Not null
Ring mode	Varchar	10	Not null
Cell block	Varchar	5	Not null

CALLS

Column name	Data type	Size	Constraints
ID	Int	11	Primary key,Not null
Logs	Varchar	100	Not null
DRIVER	Int	11	Foreign key,Not null
Date	Date		Not null
Time	Time		Not null

6. IMPLMENTATION AND TESTING

6.1 IMPLEMENTATION

Here is a high-level outline for implementing a career recommendation system based on content-based filtering:

1. **Data Collection:** - Gather data on various careers, including job titles, job descriptions, required skills, educational qualifications, experience levels, and any other relevant factors. This data can be obtained from job websites, career portals, or through APIs provided by job aggregators.

2. **Data Cleaning and Preprocessing:** - Clean the collected data by removing duplicates, irrelevant information, and inconsistencies. Preprocess the data to ensure consistency and standardization, such as converting text to lowercase, removing stop words, and tokenizing the text. Here is a high-level outline for implementing a career recommendation system based on content-based filtering:

3. **Feature Extraction:** - Extract relevant features from the job descriptions, such as keywords, skills, and qualifications. This can be done using natural language processing (NLP) techniques and tools such as TF-IDF (Term Frequency-Inverse Document Frequency) or word embedding models like Word2Vec or GloVe.

4. **Career Profile Creation:** - Create user profiles based on their skills, qualifications, and job preferences. This will involve collecting user input, such as their educational background, work experience, and career interests.

5. **Content-Based Filtering Algorithm**: - Implement a content-based filtering algorithm to recommend careers to users based on the similarity between their profiles and the features extracted from job descriptions. This can be done using techniques such as cosine similarity or Pearson correlation to calculate the similarity scores.

6. **User Interface:** - Develop a user-friendly interface where users can input their career preferences and receive personalized recommendations based on their profiles.

This interface can be a web application, mobile app, or integrated within an existing career portal.

7. **Testing and Evaluation:** - Test the recommendation system with sample user profiles and verify the accuracy and relevance of the recommendations. Use metrics such as precision, recall, and F1-score to evaluate the performance of the system.

8. **Deployment:** - Deploy the career recommendation system on a suitable platform, ensuring scalability and reliability. Monitor the system performance and gather user feedback for continuous improvement. It's important to note that this is a high-level overview and the actual implementation would involve more detailed steps, such as model training, data storage, and security considerations. Additionally, the success of the system would depend on the quality and relevance of the data collected, as well as the effectiveness of the content-based filtering algorithm and user interface design

6.1 TESTING

Testing in a career recommendation system based on content-based filtering project involves evaluating the system's recommendations to ensure they are accurate and relevant to the user's interests and skills. This may include testing the algorithm's ability to match job opportunities with a user's profile, assessing the system's ability to provide personalized recommendations, and measuring the overall effectiveness of the recommendations in helping users find suitable career opportunities. Testing may involve using sample data sets, conducting user surveys or interviews, and analyzing the system's performance metrics to validate the accuracy and usefulness of the recommendations.

6.2.1 TESTING STRATEGIES

1. **Test for accuracy:** This involves comparing the recommended career paths generated by the system with the actual career paths of individuals. This can be done through a survey of users or by analyzing historical career data.

2. **Test for coverage:** The system should be able to provide recommendations for a wide range of career paths and not be limited to a few popular options. Testing for coverage involves analyzing the diversity of recommendations generated by the system.

3. **Test for relevance:** The recommended careers should be relevant to the user's skills, interests, and qualifications. Testing for relevance involves evaluating the degree to which the recommended careers align with the user's profile.

4. **Test for scalability:** The system should be able to handle a large volume of users and provide accurate and relevant recommendations to each user. Testing for scalability involves evaluating the system's performance under various load conditions.

5. **Test for user satisfaction:** User feedback and satisfaction with the recommended career paths can be measured through surveys and interviews. Testing for user satisfaction helps in understanding the effectiveness of the recommendations in meeting the user's needs and expectations.

6. **Test for diversity:** The system should be able to provide diverse career recommendations that cater to different user profiles and backgrounds. Testing for diversity involves assessing the range of career options recommended by the system.

7. **Test for personalization:** The system should be able to provide personalized career recommendations based on each user's unique profile. Testing for personalization involves evaluating the system's ability to tailor recommendations to individual preferences and characteristics.

8. **Test for robustness:** The system should be able to handle input data with errors, noise, and missing information. Testing for robustness involves analyzing the system's performance in handling imperfect or incomplete user profiles.

7. FUTURE ENHANCEMENTS

Multi-Sensor Integration: Incorporate additional sensors, such as steering wheel angle sensors and vehicle speed sensors, to enhance the accuracy of driver behavior analysis.

Adaptive Alerts: Develop a more dynamic alert system that adjusts its intensity based on the severity of driver impairment, ensuring that alerts are effective yet not overly intrusive.

Driver Profiling: Implement machine learning techniques to create individual driver profiles, allowing the system to adapt its sensitivity and alerts based on each driver's behavior patterns.

Real-Time Data Analysis: Explore the possibility of transmitting real-time data to a central server for further analysis, enabling continuous improvement of the system's algorithms.

External Integration: Consider integration with external devices, such as wearable gadgets or smartphone applications, to provide an additional layer of vigilance and alerts.

Human-Centric Design: Continuously refine the user interface and interaction design to ensure optimal usability and minimize driver distraction during alerts.

Collaborative Partnerships: Collaborate with automotive manufacturers, safety organizations, and transportation authorities to further refine the technology and facilitate wider adoption.

Long-Term Data Collection: Gather and analyze long-term data to measure the system's
impact on reducing accidents and improving overall road safety.

The "Driver Vigilance System" project is not only a technological accomplishment but
also a testament to the potential of technology to create a positive impact on society.
By continuously evolving and incorporating future enhancements, this system has the
potential to save lives, prevent accidents, and ultimately transform the way we approach
road safety.

8. CONCLUSIONS

The project entitled ―DRIVER VIGILANCE SYSTEM‖ is aimed at developing software which is identify the drowsiness of the person driving and alert them. We can
also include view the information about locations, signals and block zones. This project
helps to reducing road accident due to driver drowsiness. The system will be blessing to the people who are driving and also people who are walking on the roads. The performance of the system is proved efficiently and maintenance was achieved in an easy way.

The Driver Vigilance System project developed using Python has successfully achieved its goals of enhancing road safety and preventing accidents by monitoring driver behavior and alerting them when signs of drowsiness or distraction are detected. This project leverages computer vision and machine learning techniques to analyze the driver's facial features, eye movements, and head orientation in real-time. Through careful design, implementation, and testing, the project demonstrates the potential to significantly reduce the risks associated with driver fatigue and inattentiveness.

The package was designed in such a way that future modifications can be done easily. The application simulator is to efficiently evaluate the candidate thoroughly through a fully automated system that not only saves a lot of time but also gives fast results.

9. BIBLIOGRAPHY

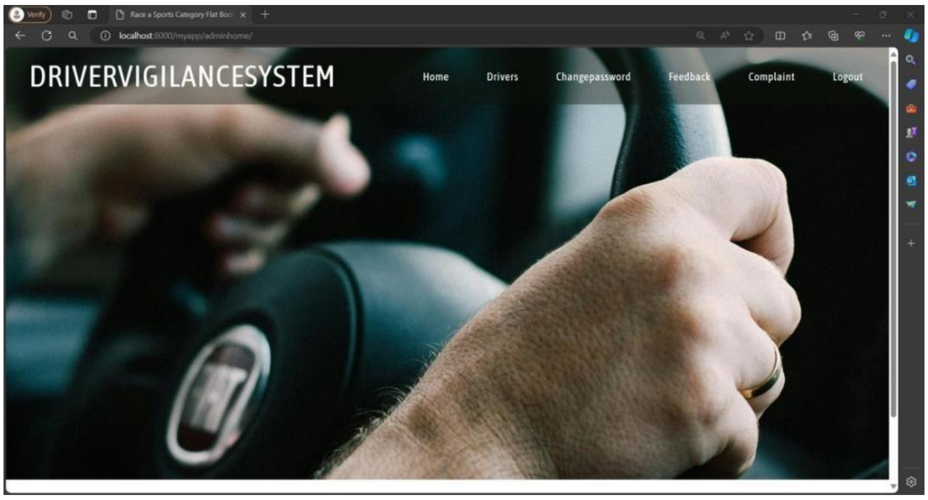
Books

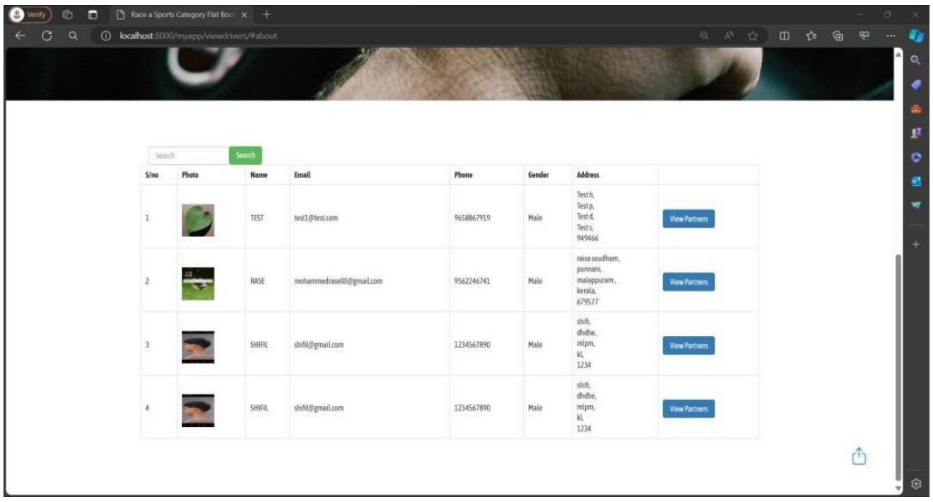
- Database System Concept –Henry. F. Korth (4th Edition)
- Element of System Analysis and Design –Marvin Gore and John W Stubbe
- System Analysis and Design – Robert.E. Leslie

Websites

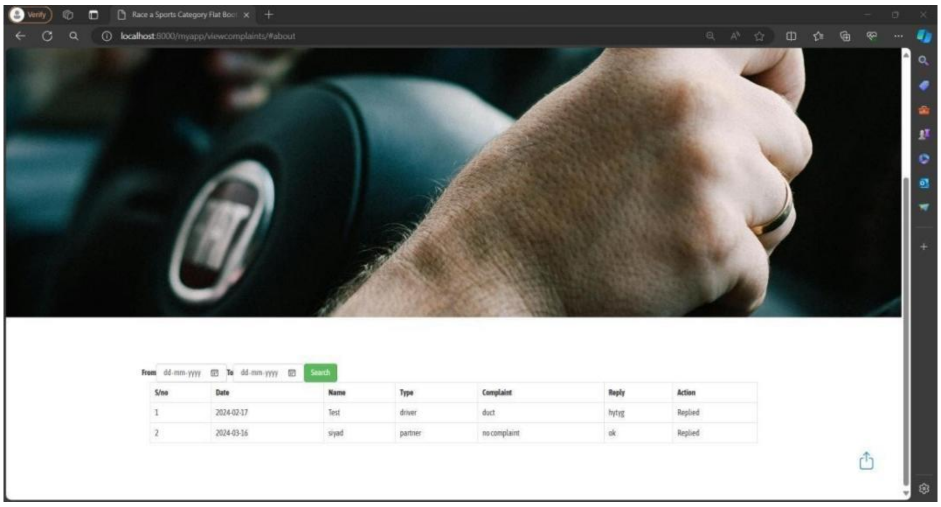
- www.wikipedia.org
- www.guru99.com

10. SCREENSHOTS

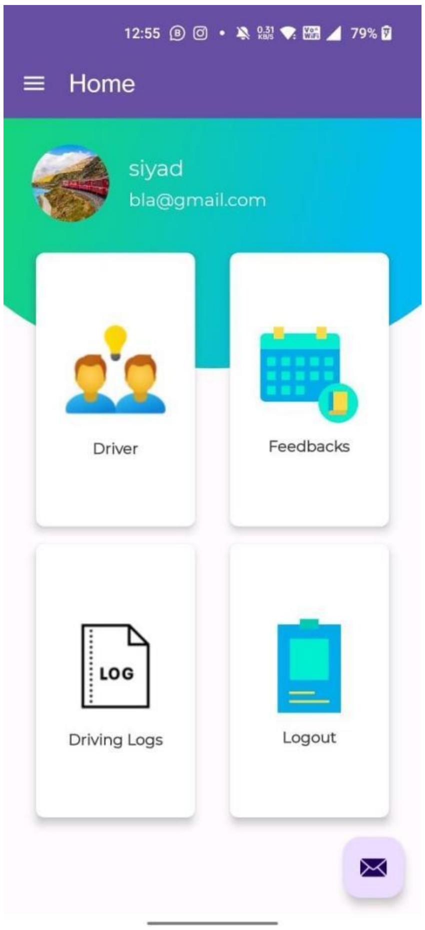




S/no	Photo	Name	Email	Phone	Gender	Address	
1		TEST	test1@test.com	9658867919	Male	Test h, Test p, Test d, Test s, 949466	View Partners
2		RASE	mohammedrase00@gmail.com	9562246741	Male	raisa soudham, ponnani, malappuram, kerala, 679577	View Partners
3		SHIFIL	shifil@gmail.com	1234567890	Male	shifi, dhdhe, mlpm, kl, 1234	View Partners
4		SHIFIL	shifil@gmail.com	1234567890	Male	shifi, dhdhe, mlpm, kl, 1234	View Partners



S/no	Date	Name	Type	Complaint	Reply	Action
1	2024-02-17	Test	driver	duct	hytyg	Replied
2	2024-03-16	siyad	partner	no complaint	ok	Replied

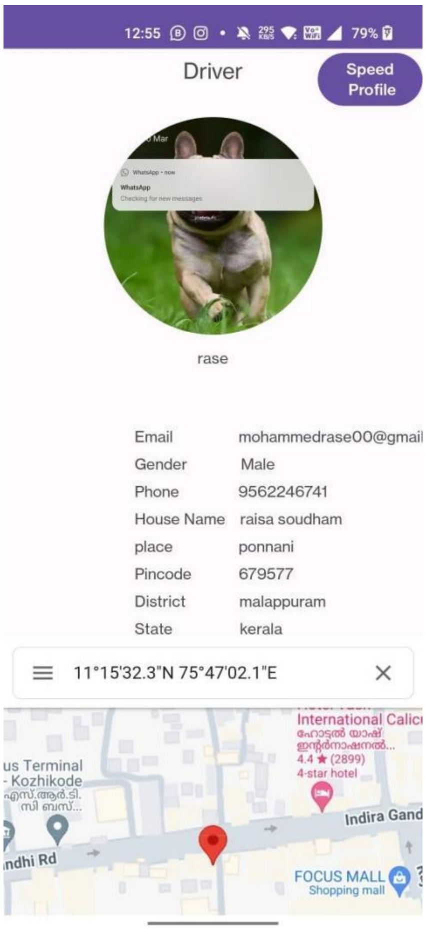


12:55 79%

Home

siyad
bla@gmail.com

Driver
Feedbacks
Driving Logs
Logout



12:55 79%

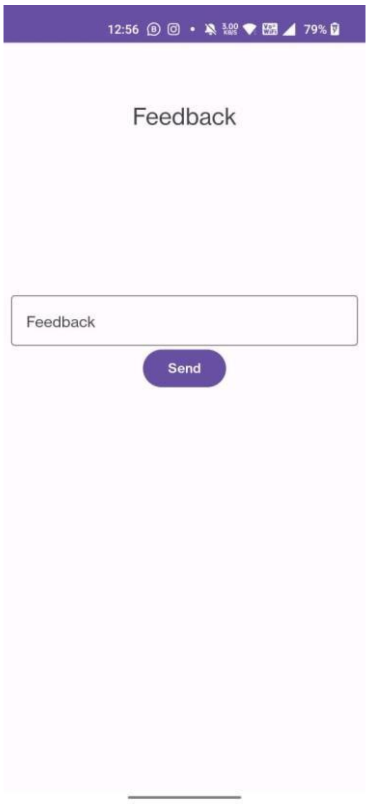
Driver

Speed Profile

rase

Email	mohammedrase00@gmail
Gender	Male
Phone	9562246741
House Name	raisa soudham
place	ponnani
Pincode	679577
District	malappuram
State	kerala

11°15'32.3"N 75°47'02.1"E

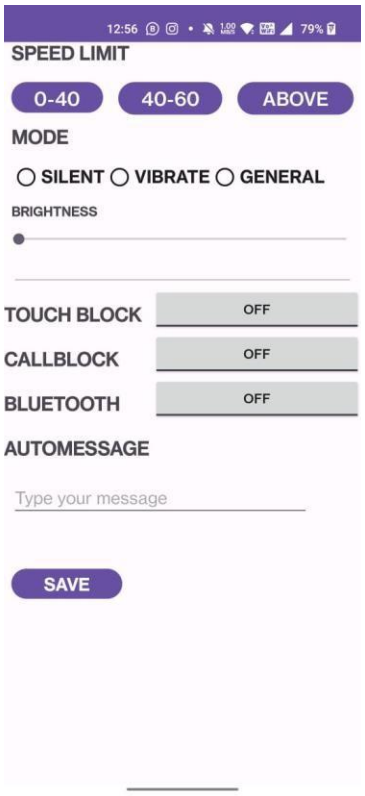


12:56 79%

Feedback

Feedback

Send



12:56 79%

SPEED LIMIT

0-40 40-60 ABOVE

MODE

○ SILENT ○ VIBRATE ○ GENERAL

BRIGHTNESS

TOUCH BLOCK OFF

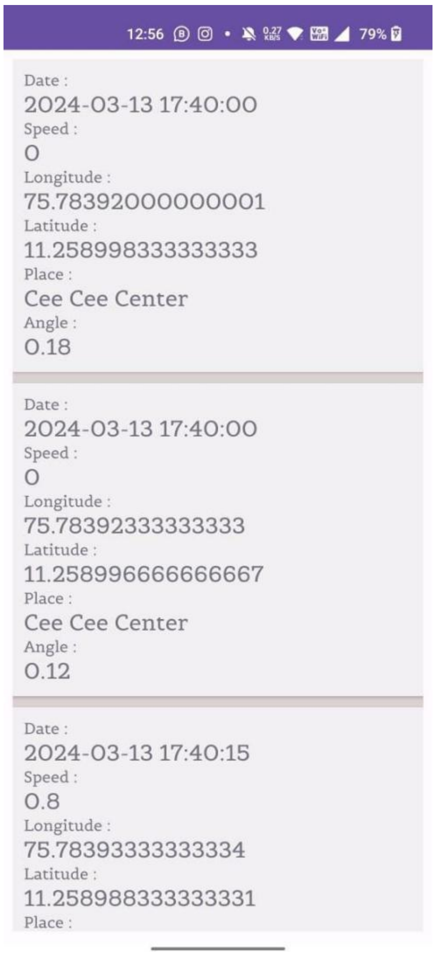
CALLBLOCK OFF

BLUETOOTH OFF

AUTOMESSAGE

Type your message

SAVE

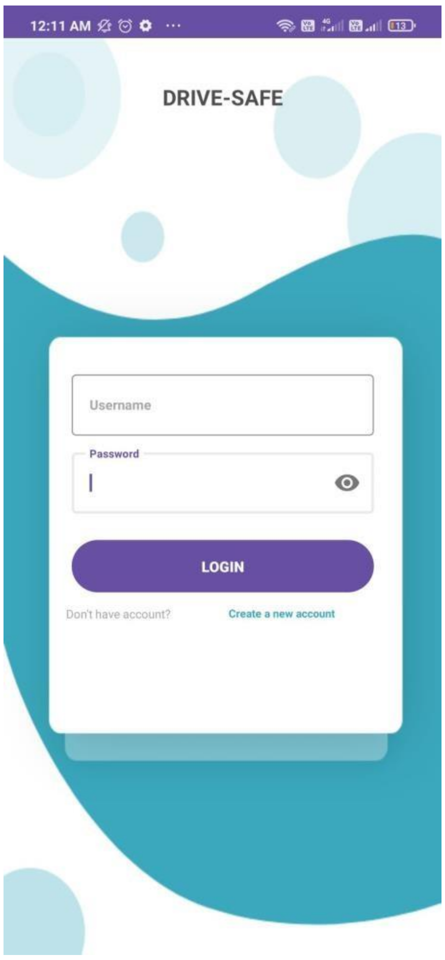


12:56 79%

Date :
2024-03-13 17:40:00
Speed :
0
Longitude :
75.78392000000001
Latitude :
11.258998333333333
Place :
Cee Cee Center
Angle :
0.18

Date :
2024-03-13 17:40:00
Speed :
0
Longitude :
75.78392333333333
Latitude :
11.258996666666667
Place :
Cee Cee Center
Angle :
0.12

Date :
2024-03-13 17:40:15
Speed :
0.8
Longitude :
75.78393333333334
Latitude :
11.258988333333331
Place :



12:11 AM

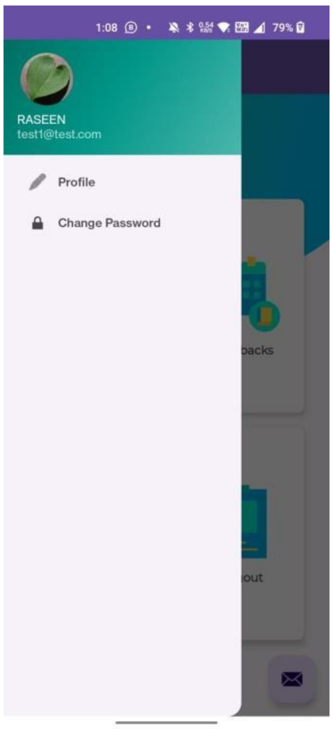
DRIVE-SAFE

Username

Password

LOGIN

Don't have account? Create a new account

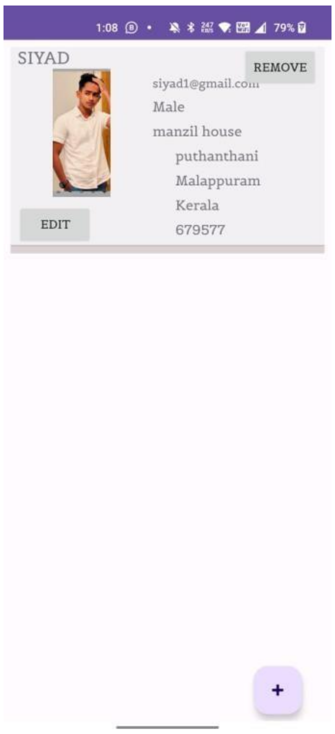


1:08 79%

RASEEN
test1@test.com

Profile

Change Password



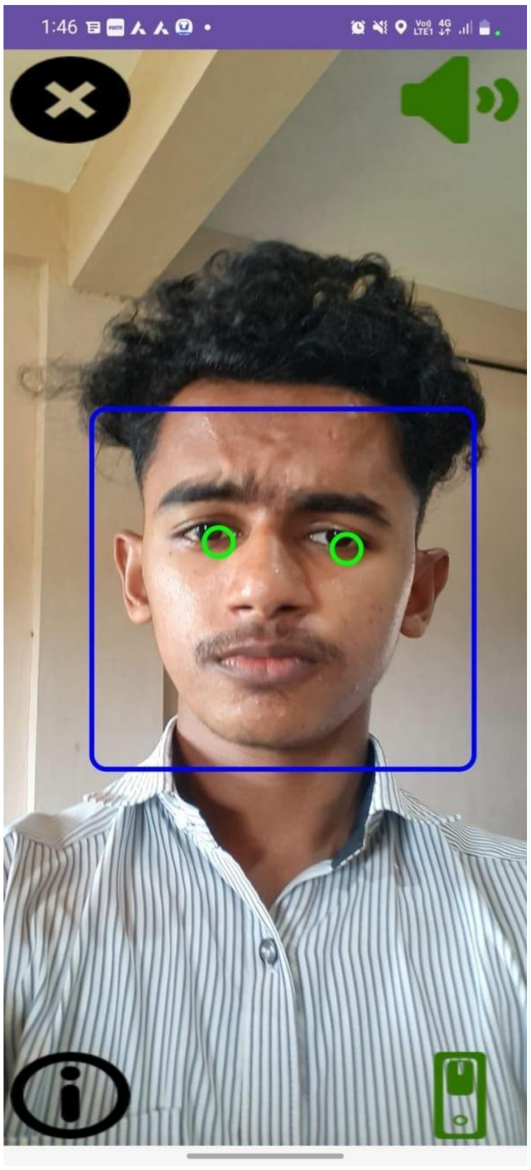
1:08 79%

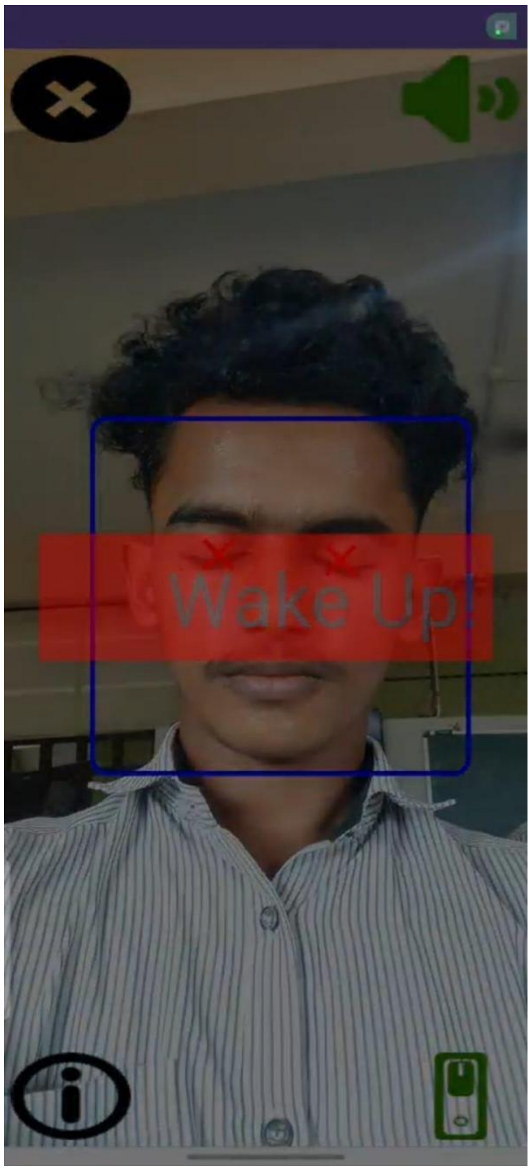
SIYAD REMOVE

siyad1@gmail.com
Male
manzil house
puthanthani
Malappuram
Kerala
679577

EDIT

+


1:46


Wake Up!

10. SAMPLE CODE

```
from django.http import HttpResponse, JsonResponse

from django.shortcuts import render, redirect



from dvigilancesystem import settings

from .models import *



try:

    import socket



    s = socket.gethostbyname(socket.gethostname())

    print('Starting development server at: http://' + s + ':8000')

except:

    pass

# Create your views here.



def login(request):

    return render(request,'admin/loginpage.html')



def login_post(request):

    username = request.POST['user']
    password = request.POST['pass']

    log = Login.objects.filter(username=username,password=password)

    if log.exists():

        logg = Login.objects.get(username=username, password=password)

        request.session['lid'] = logg.id

        if logg.type=='admin':

            return
HttpResponse('''<script>window.location='/myapp/adminhome/'</script>''')

        else:

            return
HttpResponse('''<script>alert('invalid');window.location='/myapp/login/'</script>''')

    else:

        return
HttpResponse('''<script>alert('invalid');window.location='/myapp/login/'</script>''')



def adminhome(request):

    if  request.session['lid']=='':

        return redirect('/myapp/login/')

    return render(request,'admin/new_index.html')



def viewdrivers(request):

    if  request.session['lid']=='':

        return redirect('/myapp/login/')
    driv = Drivers.objects.all()

    return render(request,'admin/viewdrivers.html',{'data':driv})



def dsearch(request):

    if  request.session['lid']=='':

        return redirect('/myapp/login/')

    dsearch = request.POST['dsearch']

    driv = Drivers.objects.filter(name__contains=dsearch)

    return render(request,'admin/viewdrivers.html',{'data':driv})



def viewpartners(request,id):

    if request.session['lid']=='':

        return redirect('/myapp/login/')

    partn = Partners.objects.filter(DRIVER_id=id)

    return render(request,'admin/viewpartners.html',{'data':partn})




# def psearch(request):

#     psearch = request.POST['psearch']

#     partn = Partners.objects.filter(name__contains=psearch)

#     return render(request,'admin/viewpartners.html',{'data':partn})
```

```
def  viewfeedback(request):
    if request.session['lid']=='':
        return redirect('/myapp/login/')
    vfeed = Feedbacks.objects.all()
    l=[]
    for i in vfeed:
        name=''
        if i.type=='driver':
            name=Drivers.objects.get(LOGIN_id=i.LOGIN_id)
        elif i.type=='partner':
            name=Partners.objects.get(LOGIN_id=i.LOGIN_id)

l.append({"name":name.name,"type":i.type,"date":i.date,"feedbacks":i.feedbacks})
    return render(request,'admin/viewfeedback.html',{'data':l})


def feedsearch(request):
    if  request.session['lid']=='':
        return redirect('/myapp/login/')
    frm = request.POST['fdate']
    to = request.POST['tdate']
    sfeed = Feedbacks.objects.filter(date__range=[frm,to])
    return render(request,'admin/viewfeedback.html',{'data':sfeed})


def viewcomplaints(request):
    if request.session['lid']=='':
        return redirect('/myapp/login/')
    comp = Complaints.objects.all()
    l=[]
    for i in comp:
        name=''
        if i.type=='driver':
            name=Drivers.objects.get(LOGIN_id=i.LOGIN_id)
        elif i.type=='partner':
            name=Partners.objects.get(LOGIN_id=i.LOGIN_id)

l.append({"name":name.name,"type":i.type,"date":i.date,"complaints":i.complaints,"r
eplay":i.replay,"status":i.status,"id":i.id})
    return render(request,'admin/viewcomplaints.html',{'data':l})


def compsearch(request):
    if  request.session['lid']=='':
        return redirect('/myapp/login/')
    frm = request.POST['fdate']
    to = request.POST['tdate']

    comp = Complaints.objects.filter(date__range=[frm,to])

    return render(request,'admin/viewcomplaints.html',{'data':comp})


def give_comp_reply(request):

    if request.session['lid']=='':

        return redirect('/myapp/login/')

    return render(request,'admin/givecompreply.html')


def give_comp_reply(request,id):

    if request.session['lid']=='':

        return redirect('/myapp/login/')

    comp = Complaints.objects.get(id=id)

    return render(request,'admin/givecompreply.html',{'data':comp})


def give_comp_reply_post(request):

    if request.session['lid']=='':

        return redirect('/myapp/login/')

    rep = request.POST['rep']

    id = request.POST['id']

    comp = Complaints.objects.get(id=id)

    comp.replay=rep
    comp.status='replied'

    comp.save()

    return
HttpResponse('''<script>alert('replaid');window.location='/myapp/viewcomplaints/'</s
cript>''')



def changepass(request):

    if  request.session['lid']=='':

        return redirect('/myapp/login/')

    return render(request,'admin/changepass.html')


def changepass_post(request):

    if request.session['lid']=='':

        return redirect('/myapp/login/')

    cpass = request.POST['cpass']

    npass = request.POST['npass']

    confpass = request.POST['confpass']

    id = request.session['lid']

    log=Login.objects.get(id=id)

    if log.password==cpass:

        if npass==confpass:
            log.password=npass

            log.save()

            return
HttpResponse('''<script>alert('changed');window.location='/myapp/login/'</script>''')

        else:

            return
HttpResponse('''<script>alert('invalid');window.location='/myapp/changepass/'</script
>''')

    else:

        return
HttpResponse('''<script>alert('invalid');window.location='/myapp/changepass/'</script
>''')



def logout(request):

    request.session['lid'] = ''

    return HttpResponse('''<script>window.location='/myapp/login/'</script>''')
```

DRIVER

```
def d_signup_post(request):

    dname = request.POST['name']

    print(dname)

    demail = request.POST['email']

    dgender = request.POST['gender']

    dphone = request.POST['phone']

    dhouse = request.POST['address']

    dplace = request.POST['place']

    ddistrict = request.POST['district']

    dstate = request.POST['state']

    dpincode = request.POST['pincode']

    image = request.POST['photo']

    dpassword = request.POST['pass']


    import datetime

    import base64

    date = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

    a = base64.b64decode(image)

    fh = open(settings.MEDIA_ROOT+"\\" + date + ".jpg", "wb")

    # fh = open("C:\\Users\\91815\\PycharmProjects\\dvigilancesystem\\media\\" +
date + ".jpg", "wb")
```

```
path = "/media/" + date + ".jpg"

fh.write(a)

fh.close()


log = Login()

log.username=demail

log.password=dpassword

log.type='driver'

log.save()


drv = Drivers()

drv.LOGIN_id=log.id

drv.name=dname

drv.email=demail

drv.gender=dgender

drv.phone=dphone

drv.address=dhouse

drv.place=dplace

drv.district=ddistrict

drv.state=dstate

drv.pincode=dpincode

drv.photo=path
```

```
    drv.save()

    return JsonResponse({"status":"ok"})



def andrologin(request):

    usr = request.POST['user']

    passw = request.POST['pass']

    log = Login.objects.filter(username=usr,password=passw)

    if log.exists():

        logg = Login.objects.get(username=usr,password=passw)

        if logg.type == "driver":

            ress=Drivers.objects.get(LOGIN_id=logg.id)

            return
JsonResponse({"status":"ok",'lid':logg.id,'type':logg.type,'name':ress.name,'email':ress
.email,'photo':ress.photo})

        elif logg.type == "partner":

            res1=Partners.objects.get(LOGIN_id=logg.id)

            return
JsonResponse({"status":"ok",'lid':logg.id,'type':logg.type,'name':res1.name,'email':res
1.email,'photo':res1.photo})

    else:

        return JsonResponse({"status":"not ok"})



def add_partner_post(request):
```

```
pname = request.POST['name']

pemail = request.POST['email']

pgender = request.POST['gender']

pphone = request.POST['phone']

phouse = request.POST['address']

pplace = request.POST['place']

pdistrict = request.POST['district']

pstate = request.POST['state']

ppincode = request.POST['pincode']

image = request.POST['photo']

lid = request.POST['lid']


import random

ppassword=random.randint(0000,9999)


import datetime

import base64


date = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

a = base64.b64decode(image)

fh = open("media/" + date + ".jpg", "wb")

path = "/media/" + date + ".jpg"
```

```
fh.write(a)

fh.close()


log = Login()

log.username = pemail

log.password = pphone

log.type = 'partner'

log.save()


drv = Partners()

drv.LOGIN_id = log.id

drv.DRIVER_id=Drivers.objects.get(LOGIN_id=lid).id

drv.name = pname

drv.email = pemail

drv.gender = pgender

drv.phone = pphone

drv.address = phouse

drv.place = pplace

drv.district = pdistrict

drv.state = pstate

drv.pincode = ppincode

drv.photo = path
```

```
    drv.save()
    return JsonResponse({"status":"ok"})


def view_partner_post(request):
    id = request.POST['lid']
    res = Partners.objects.filter(DRIVER_LOGIN_id=id)
    l=[]
    for i in res:

l.append({'id':i.id,'name':i.name,'email':i.email,'phone':i.phone,'address':i.address,'gen
der':i.gender,'place':i.place,

'district':i.district,'state':i.state,'pincode':i.pincode,'photo':i.photo,'DRIVER_id':i.DRIV
ER_id,'LOGIN_id':i.LOGIN_id})
    return JsonResponse({"status":"ok","data":l})


def  delete_partner(request):
    pid = request.POST['pid']
    lid = request.POST['lid']
    part = Partners.objects.get(id=pid)
    part.delete()
    part2=Login.objects.get(id=lid)
    part2.delete()
    return JsonResponse({"status":"ok"})


def view_edit_part_post(request):

    lid = request.POST['pid']

    part = Partners.objects.get(id = lid)

    return
JsonResponse({"status":"ok","name":part.name,"email":part.email,"gender":part.gend
er,"phone":part.phone,


"address":part.address,"place":part.place,"district":part.district,"state":part.state,"pinc
ode":part.pincode,"photo":part.photo

                    })


def view_driver_profile(request):

    lid=request.POST['lid']

    drv=Drivers.objects.get(LOGIN_id=lid)

    return
JsonResponse({"status":"ok","name":drv.name,"email":drv.email,"gender":drv.gende
r,"phone":drv.phone,"address":drv.address,

                    "place": drv.place, "district": drv.district, "state": drv.state, "pincode":
drv.pincode,

                    "photo": drv.photo

                    })
```

```
def edit_part_post(request):
    pname = request.POST['name']
    pemail = request.POST['email']
    pgender = request.POST['gender']
    pphone = request.POST['phone']
    phouse = request.POST['address']
    pplace = request.POST['place']
    pdistrict = request.POST['district']
    pstate = request.POST['state']
    ppincode = request.POST['pincode']
    pid = request.POST['pid']
    image = request.POST['photo']

    if len(image)>5:

        import datetime
        import base64

        date = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
        a = base64.b64decode(image)
        fh = open("media/" + date + ".jpg", "wb")
        path = "/media/" + date + ".jpg"
        fh.write(a)

        fh.close()


        drv = Partners.objects.get(id=pid)

        drv.photo = path

        drv.save()

    drv = Partners.objects.get(id=pid)

    drv.name = pname

    drv.email = pemail

    drv.gender = pgender

    drv.phone = pphone

    drv.address = phouse

    drv.place = pplace

    drv.district = pdistrict

    drv.state = pstate

    drv.pincode = ppincode

    drv.save()

    return JsonResponse({"status": "ok"})


def drv_send_feed(request):

    lid = request.POST['lid']

    feed = request.POST['feed']
    did=Drivers.objects.get(LOGIN_id=lid).LOGIN_id

    import datetime

    date = datetime.datetime.now().strftime('%Y-%m-%d')

    fe = Feedbacks()

    fe.LOGIN_id=did

    fe.feedbacks=feed

    fe.date=date

    fe.type='driver'

    fe.save()

    return JsonResponse({"status":"ok"})


def view_drv_profile(request):

    lid=request.POST['lid']

    drv = Drivers.objects.get(LOGIN_id=lid)

    return
JsonResponse({"status":"ok","name":drv.name,"email":drv.email,"gender":drv.gende
r,"phone":drv.phone,


"address":drv.address,"place":drv.place,"district":drv.district,"state":drv.state,"pincod
e":drv.pincode,"photo":drv.photo

                        })


def edit_drv_profile(request):
    dname = request.POST['name']

    demail = request.POST['email']

    dgender = request.POST['gender']

    dphone = request.POST['phone']

    dhouse = request.POST['address']

    dplace = request.POST['place']

    ddistrict = request.POST['district']

    dstate = request.POST['state']

    dpincode = request.POST['pincode']

    image = request.POST['photo']

    lid = request.POST['lid']


    if len(image)>5:


        import datetime

        import base64


        date = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

        a = base64.b64decode(image)

        fh = open("C:\\Users\\91815\\PycharmProjects\\dvigilancesystem\\media\\" +
date + ".jpg", "wb")

        path = "/media/" + date + ".jpg"
        fh.write(a)

        fh.close()


        drv = Drivers.objects.get(LOGIN_id=lid)

        drv.photo = path

        drv.save()

    drv = Drivers.objects.get(LOGIN_id=lid)

    drv.name = dname

    drv.email = demail

    drv.gender = dgender

    drv.phone = dphone

    drv.address = dhouse

    drv.place = dplace

    drv.district = ddistrict

    drv.state = dstate

    drv.pincode = dpincode

    drv.save()


    return JsonResponse({"status":"ok"})


def view_drv_complaints(request):

    lid = request.POST['lid']
    comp=Complaints.objects.filter(LOGIN_id=lid)


    l=[]
    for i in comp:

l.append({"id":i.id,"date":i.date,"status":i.status,"complaint":i.complaints,"reply":i.rep
lay})


    return JsonResponse({"status":"ok","data":l})


def add_drv_complaint(request):
    lid=request.POST['lid']
    comp=request.POST['comp']
    import datetime
    date = datetime.datetime.now().strftime('%Y-%m-%d')
    com = Complaints()
    com.LOGIN_id=Drivers.objects.get(LOGIN_id=lid).LOGIN_id
    com.status='pending'
    com.replay='pending'
    com.type='driver'
    com.date=date
    com.complaints=comp
    com.save()

    return JsonResponse({"status":"ok"})


def drv_changepass(request):

    lid = request.POST['lid']

    curpass = request.POST['currentpassword']

    newpass = request.POST['newpassword']

    ps = Login.objects.get(id=lid)

    if ps.password==curpass:

        ps.password=newpass

        ps.save()

        return JsonResponse({"status":"ok"})

    else:

        return JsonResponse({"status":"invalid"})
```