iTrucking: Intelligent Trucking Management System for Efficient Logistics

UNDERGRADUATE THESIS
Submitted as one of the requirements to obtain
Sarjana Komputer (S.Kom.)

By:

GERALD SUTARSO 001202100117

FACULTY OF COMPUTER SCIENCE
INFORMATICS STUDY PROGRAM
CIKARANG
MAY, 2024

Copyright by

GERALD SUTARSO 001202100117

ITRUCKING: INTELLIGENT TRUCKING MANAGEMENT SYSTEM FOR EFFICIENT LOGISTICS

By

GERALD SUTARSO 001202100117

Approved:

Rosalina, S.Kom., M. Kom. Thesis Advisor	Cutifa Safitri, B. CS., M.IT., Ph.D. Program Head of Informatics Study Program

Ir. Rila Mandala, M.Eng., Ph.D.
Dean of Faculty of Computer Science

STATEMENT OF ORIGINALITY

In my capacity as an active student of President University and as the author of the thesis/final project/business plan (underline that applies) stated below:

Name : Gerald Sutarso – 001202100117
Study Program : Informatics
Faculty : Computer Science

I hereby declare that my thesis/final project/business plan entitled **"ITRUCKING: INTELLIGENT TRUCKING MANAGEMENT SYSTEM FOR EFFICIENT LOGISTICS"** is to the best of my knowledge and belief, an original piece of work based on sound academic principles. If there is any plagiarism detected in this thesis/final project/business plan, I am willing to be personally responsible for the consequences of these acts of plagiarism, and will accept the sanctions against these acts in accordance with the rules and policies of President University.

I also declare that this work, either in whole or in part, has not been submitted to another university to obtain a degree.

Cikarang, 2024

GERALD SUTARSO

SCIENTIFIC PUBLICATION APPROVAL FOR ACADEMIC INTEREST

As an academic community member of the President's University, I, the undersigned:

Name : Gerald Sutarso – 001202100117
Study Program : Informatics

for the purpose of development of science and technology, certify, and approve to give President University a non-exclusive royalty-free right upon my final report with the title

"iTrucking: Intelligent Trucking Management System for Efficient Logistics"

With this non-exclusive royalty-free right, President University is entitled to converse, to convert, to manage in a database, to maintain, and to publish my final report. There are to be done with the obligation from President University to mention my name as the copyright owner of my final report.

This statement I made in truth.

Cikarang, 17 May 2024

GERALD SUTARSO

ADVISOR APPROVAL FOR JOURNAL/INSTITUTION'S REPOSITORY

As an academic community member of the President's University, I, the undersigned:

Name : Rosalina
Study Program : Informatics
Faculty : Computer Science

declare that following thesis :

Title of thesis : **iTrucking: Intelligent Trucking Management System for Efficient Logistics.**

Thesis author :Gerald Sutarso – 001202100117

will be published in the **Faculty of Computer Science.**

Cikarang, 2024

Rosalina, S.Kom., M. Kom.

SIMILARITY INDEX REPORT

TURNITIN:

Paper Title	Uploaded	Grade	Similarity
iTrucking: Intelligent Trucking Management System for Efficient Logistics	17 May 2024 23:56	--	10%

ABSTRACT

The trucking and logistics sectors are always under pressure to improve user experience, efficiency, and transparency. In order to help customers, drivers, and administrators, this web-based application integrates additional functionalities to address these difficulties. Users may manage their accounts, receive customized vehicle suggestions, and use powerful filtering tools to make well-informed decisions thanks to the user-friendly Customer Management module. Transparent communication channels, real-time delivery tracking, and safe and practical payment solutions all contribute to higher levels of customer satisfaction.

In order to promote driver well-being, the Driver Management module focuses on driver profile, customer request management, and speedy medical exams. Vehicle management ensures optimal vehicle performance by using sensor data and machine learning to deliver preemptive maintenance notices and effective tracking mechanisms. In addition to producing insightful reports and streamlining data administration, the Administrator module also makes maintenance scheduling easier and keeps an eye on deliveries and potential delays.

This web-based program, which makes use of state-of-the-art technologies, streamlines logistics processes and provides a comprehensive solution for a modern and effective trucking ecosystem.

Keywords : Applications for logistics, the trucking industry, machine learning, customer and driver management, real-time tracking, proactive maintenance, secure payments, and data management.

DEDICATION

I dedicate this final project to almighty god, my family, and for my own self.

ACKNOWLEDGMENTS

God is to be praised. Through His kindness and sincerity, I was still given the chance to be able to do and finish this assignment. I would also like to extend my sincere gratitude to everyone who has helped and supported me, directly or indirectly. I'm hoping that all of the hard work and generosity that I've received will be returned in kind. I sincerely thank each and every one of you.

On this occasion, the author would like to thank the parties who have helped a lot in the implementation and preparation of the Final Project, including:

1. My family, for the love and never-ending support from the very beginning until now.
2. Maam Rosalina, S.Kom., M. Kom. as my Final Project Advisor, for all advice and guidance throughout this Final Project journey.
3. All lecturers of the computing faculty who have accompanied my progress from the beginning of the lecture to the end of the lecture.
4. All my friends that I could not mention one by one, for the support so we can together get the Final Project journey done.
5. Myself, for the effort, for the hard work, and for not giving up in the middle of completing this Final Project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I
F100
PROPOSAL

I.1 Background of the Problem

Indonesia's logistics management system encounters many problems, including constant delivery delays and an increase in truck theft cases. Geographical issues such as the archipelagic character of the country, add to logistical difficulties, resulting in delays in the final delivery process. For example, during the Ramadan season in 2019, delivery delays were significant as logistics failed to keep up with higher demands, traffic jams, and inadequate last-minute operation, affecting business enterprises and troubling consumers who were hoping for immediate delivery. Simultaneously, a rise of truck thefts threaten the distribution chain's reliability. In 2020, several popular truck thefts were reported, with a focus on delivery trucks carrying expensive goods. These illegal acts not only cause immediate financial damages, but they also hinder the logistics chain's reliability, generating additional delays in delivery.

Moreover, internal issues occur frequently in Indonesia's logistics management system, due particularly to poor management practices that limit efficiency in operation. Inefficient routing is an important issue, with poor planning and coordination of delivery routes leading to disruptions and inefficiency. For example, a lack of real-time data and advanced route optimization technologies impedes the system's capacity to adapt flexibly to shifts in conditions, leading to slow performance. The poor integration of technology worsens these issues. The slow adoption of new logistics technology, such as real-time tracking and data analytics, restricts the system's ability to optimize operations and respond proactively to disturbances. This technological gap not only reduces delivery speed and accuracy, but it also prevents the application of current methods that could improve overall efficiency.

I.2 Problem Statement

- Existing systems lacks features, offering minimal options for users to manage profiles and receive personalized truck recommendations. This results in a suboptimal customer experience and inefficient decision-making processes.
- Communication between customer and the driver may have some gaps which will impact transparency and timely information sharing.
- The absence of advanced vehicle management systems, incorporating sensor data and machine learning for maintenance notifications, results in unplanned vehicle breakdowns that can cause disruptions in the logistics operations.
- The shortage of combinations of tracking system to the management system specified for trucking processes.
- The lack of systems to monitor and evaluate driver health which may lead to reduced performance or even more fatal risks.

I.3 Problem Functional Analysis

I.3.1 *Functionalities 1: Customer Management*

The Customer Management functionality aims to provide customers with a set of tools to navigate the application's logistics and trucking system. By implementing features such as personal recommendations, a transparent booking processes and a secure and efficient payment, while also providing real-time tracking for their delivery, this functionality will aim to contribute to an improved overall customer experience within the logistics and trucking industry

I.3.2 Functionalities 2: Driver Management

The Driver Management functionality is designed to streamline and enhance the role of drivers within the logistics and trucking system. By prioritizing driver profiling, efficient request handling, transparent notifications, and quick medical checkups, this functionality aims to create a system that ensures the well-being of drivers, enhances communication, and contributes to the overall efficiency and reliability of the logistics operations.

I.3.3 Functionalities 3 : Vehicle Management

The Vehicle/Truck Management functionality aims to ensure the efficient and reliable operation of the logistics and trucking fleet. By implementing real-time monitoring, predictive maintenance notifications, and vehicle tracking, this functionality contributes to the overall optimization of logistics operations, minimizes disruptions, and enhances the transparency and communication channels between administrators and customers.

I.3.4 Functionalities 4: Administration

The Administration functionality helps to ensure that every aspect from bookings and payments, maintenance, deliveries, down until the customers, vehicles and their drivers data are properly managed. The connector between the interactions of customers and drivers, as well as drivers and their vehicles. This functionality aims to create a system that enables administrators to make informed decisions, improve customer satisfaction, and optimize the logistics and trucking operations.

I.4 Solution Selection

I.4.1 *Alternative Solution 1: A Web based Vehicle Management*

A web-based application that improves the logistics and trucking industry by seamlessly integrating an advanced set of functionalities designed to enhance customer, driver, and administrator experiences. With an intuitive Customer Management module, users can effortlessly manage their profiles, receive personalized truck recommendations, and make informed decisions through advanced filtering options. The system ensures secure and convenient payments while providing real-time delivery tracking and transparent communication channels between customers and drivers. Driver Management focuses on profiling drivers, managing customer requests, and implementing quick medical checkups, ensuring a holistic approach to driver well-being. Vehicle Management employs sensor data and machine learning for proactive maintenance notifications, alongside efficient tracking mechanisms. The Administrator module streamlines data management, facilitates maintenance scheduling, generates insightful reports, and monitors deliveries and potential delays. This web-based application leverages cutting-edge technologies to optimize logistics operations, delivering a comprehensive solution for a modernized and efficient trucking ecosystem.

I.4.2 *Alternative Solution 2: A blockchain-based platform*

A blockchain-based platform that enables secure and transparent transactions between customers, drivers, and administrators in the logistics and trucking industry. With a decentralized ledger system, customers can easily verify the identity and reputation of drivers, as well as track the delivery status and location of their goods. Drivers can access smart contracts that specify the terms and conditions of each delivery, as well as receive instant payments upon completion. Administrators can monitor the performance and compliance of drivers and vehicles, as well as generate analytics and

reports based on the data stored on the blockchain. This blockchain-based platform leverages innovative technologies to address the challenges of customer management, driver management, and vehicle management, offering a reliable and efficient solution for a modernized and sustainable trucking ecosystem.

I.4.3 Alternative Solution 3

A digital network that connects haulage, logistics, and freight forwarding companies through a mobile app, allowing them to utilize the unused capacity of the logistic transport system. This solution can reduce the number of empty miles, optimize the route planning, and increase the efficiency and transparency of the delivery process. This solution can also leverage machine learning and data analytics to provide insights and recommendations for the customers, drivers, and administrators.

I.4.4 Alternative Solution 4

An aerospace logistics system that can send more goods via drones or airships, bypassing the congested roads and ports. This solution can offer faster and more reliable delivery options, as well as reduce the carbon footprint and fuel consumption of the logistics and trucking industry. This solution can also employ IoT devices and sensors to monitor the condition and location of the goods, as well as provide real-time communication and feedback for the customers, drivers, and administrators.

Solution	**Advantages**	**Disadvantages**
Solution 1	Intuitive and user-friendly interface Advanced filtering options for customers	Potential cybersecurity risks High initial development and maintenance costs

	Real-time tracking and communication	Dependence on internet connectivity
Solution 2	High security and transparency Decentralized ledger for identity and reputation verification	Complexity of blockchain integration Potential scalability issues
Solution 3	Utilizes unused logistic capacity Reduces empty miles and optimizes routes	Data privacy and security concerns Reliance on accurate data and algorithms
Solution 4	Faster and more reliable delivery options Utilizes IoT and sensors for real-time monitoring	High cost of implementation and operation Limited payload capacity compared to traditional trucking

1.1 Alternative Solution 4 Table

Solution Selection

After careful consideration, Solution 1, the web-based application designed to revolutionize the logistics and trucking industry, is selected as the preferred alternative solution. This choice is based on its comprehensive approach, seamlessly integrating a robust set of functionalities to enhance the experiences of customers, drivers, and administrators. The solution addresses key challenges such as customer management, driver well-being, vehicle management, and administrator oversight. It leverages cutting-edge technologies to optimize logistics operations, providing a modernized and efficient trucking ecosystem.

Solution 1 not only aligns with the identified problems in the logistics and trucking industry but also offers a holistic and user-centric approach to address these issues. The integration of features such as real-time tracking, personalized recommendations, and transparent communication channels demonstrates a commitment to improving efficiency, transparency, and overall user satisfaction within the logistics and trucking ecosystem.

I.5 Solution Usage Scenarios

Scenario:
- Customer Booking and Delivery

1. Customer Booking:

- Action: John Smith uses the web application and then selects the "Truck Recommendation" feature in the home page and may explore available trucks based on his preferences, John can also use additional filters to further filter items such as: the truck and its driver options so they perfectly matches his preferences.

- Result: This results in John being able to choose a truck and reading the driver's profile that matches his selections, John can then proceed to book the truck for his future delivery.

2. Payment and Confirmation:

- Action: John is able to enter the delivery details by using the application, which includes the content, date, and destination of the order. John can then select among the various payment methods, and choosing to pay via credit card.

- Result: After John’s payment is validated by the system, the system will then proceed to confirm the booking, notify the driver, and then notify John about the successful payment.

3. Real-time Tracking and Notifications:

- Action: After a successful payment confirmation, John may use the application to track the truck in real-time using the map on the program. John will receive notifications from the driver during real-time at key delivery milestones.

- Result: This transparent communication channel ensures that John is well-informed throughout the entire delivery process.

4. Delivery Completion:

- Action: At the end of the order, Once the truck successfully arrives at the order destination, John will be able to receive pictures from the driver as proof of successful delivery.

- Result: John will then be able to select the application and has up to two days to confirm the successful delivery and provide feedback.

5. Driver Availability and Notification:

- Action: On the driver's side, Sarah can set her availability status. Using the application, Sarah receives notifications of customer booking requests.

- Result: Sarah can review in the application about pending requests, and may accept a suitable booking, and the system will confirm her availability for the delivery.

6. Real-time Tracking and Event Notifications (Driver):

- Action: The driver Sarah can initiate the delivery, and the system can provide her with a route. She updates the system with event notifications during the delivery.

- Result: The customer and administrator can track the truck's real-time location and receive notifications of key events during the delivery.

7. Delivery Completion and Feedback (Driver):

- Action: After reaching the destination, the driver Sarah can notify the customer through the application and can submit proof of the successful delivery through the system.

- Result: The customer will receive confirmation from Sarah, and any feedback is recorded by the system for future improvements.

8. Administrator Oversight:

- Action: The administrator can use the application to monitor ongoing deliveries, and can access real-time information, and receives notifications of possible delays.

- Result: Using administrator privileges, The Admin can easily take proactive measures to address these delays, ensuring smooth and efficient logistics operations.

- **Truck Maintenance:**

1. **Sensor Detection:**

   - Action: The sensor detects that maintenance is needed and the system gives notification to the administrator.

   - Result: The administrator verifies the notification and decides that maintenance is indeed necessary.

2. **Maintenance Scheduling:**

   -Action: The administrator schedules the maintenance and sends the date request to the driver.

   - Result: The driver gets the requested maintenance date, agrees to it, and maintenance is carried out on the agreed date.

I.6 Development Effort

I.6.1 Man-months

Name	Tasks	Individual	Time	Total
Arzidan	Handles the booking and payment process from the customer booking the truck, paying for the booking, until the driver accepts the booking. Also integrating the payment methods for the customer to pay	1 people	4 month	4 month
Gerald	Designing and creating the website that handles all of the modules. Also in charge of creating the management system for the Customer, driver, trucks. From the Customer, Driver, and Admin side.	1 person	4 month	4 month
Tristan	Creating a maintenance system using and implementing Machine Learning based on online databases obtained from OBD2 Sensors.	1 person	4 month	4 month

Jim	Creating a truck tracking system using the Internet of Things (IoT) involves integrating various technologies to monitor, collect, and transmit real-time data about the location and status of trucks.	1 person	4 month	4 month
Adisura	Creating application to monitor and evaluate driver performance by using Image Recognition, checking their faces. Also handling the driver Booking and delivery process	1 people	4 month	4 months

1.2 Man-months Table

I.6.2 Machine-months

The machines we need from start to finish for the project:

1. Laptop
2. Hosting Server (4 months of development and testing)

I.6.3 Development Tools

The project will need some tools to develop, which are:

- VSCODE
- Django
- APIs
- Mysql

- Internet

I.6.4 Test Equipment

For testing and debugging our app, here are the tools we might use:

- Laptop/PC
- Personal Car
- Human test subject
- Smartphone

I.6.5 Expert Needs

- This project will need some consultation from Logistic Professional(s) as expert(s) that know better in what a truck may need to maintain, and the currently existing system that most of the Logistic (specifically truckings) are using.

I.6.6 Cost Estimation

● Features	● Tools and Materials	● Cost
Payment Method	Midtrans	Various methods have various pricing/transactions. With average Rp 3.000/transaction
Vehicle Tracking and Condition Sensor	UbloxNeo-6m	Rp 39.000,00 - Rp 500.000,00
	SIM800L	Rp

		50.000,00-100.000,00
	Arduino Uno	
Connection of Sensor to Computer	Google Drive API	Free
Vehicle Tracking	Mapbox API	Free <50.000 load
	Google Geolocation API	Rp 77.46/request

1.3 Cost Estimation Table

I.6.7 Timelines

Phase	M1	M2	M3	M4	Deliverable
Plan					Full Project
Design					Mock-up & Timeline
Develop					Front-end, Back-end, DB
Test					Overview & Timeline
Deploy					Report & Timeline
Review					Demo & Timeline
Launch					Live Website

1.4 Timelines Table

I.7 Conclusion

The implementation of a Smart Logistics Management System offers a transformative solution to mitigate delivery delays in a rapidly evolving business landscape. By harnessing cutting-edge technologies such as IoT, AI, and data analytics, the system optimizes the entire supply chain process. Real-time tracking of vehicles, intelligent route planning, and predictive analytics enable companies to proactively identify bottlenecks and dynamically adjust delivery routes. This not only enhances operational efficiency but also minimizes delays caused by unforeseen circumstances. Moreover, the system facilitates seamless communication between stakeholders, fostering collaboration and swift problem resolution. By providing accurate and timely information on inventory levels, transportation statuses, and potential disruptions, companies can make informed decisions to prevent delays. Automation of routine tasks and predictive maintenance further contribute to streamlined operations.

CHAPTER II
F200

SPECIFICATION

II.1 Existing Business Process

At the moment, the truck management business process at RaHarTo_Trans is dealing with some very significant difficulties. It is believed that truck management is insufficient and less effective in guaranteeing seamless operations. The absence of a system that can deliver precise information about the location, state, and usage of trucks is one of the primary issues.

The manual procedures still used in truck management frequently cause delays in vehicle scheduling and monitoring. Low operational efficiency is the result of truck management systems' lack of technological integration. Consequently, this leads to a less than ideal and slowed down vehicle rental procedure.

Furthermore, the truck's technical trouble avoidance mechanism is frequently flawed. When truck conditions are not tracked in real time, there is a greater chance of damage or malfunction, which can compromise driver productivity and public safety.

The lack of openness in truck repair and upkeep is another issue with truck management. Truck performance may suffer from an inconsistent maintenance schedule, which would also be bad for the user experience.

It is imperative to tackle the issue of undermining efficient reporting mechanisms. It is challenging for pertinent parties to recognize issues and take prompt action to address them in the absence of precise and thorough reports.

Aside from that, another significant concern is trash management in the vicinity of the truck area. garbage can build up around truck storage facilities as a result of poor garbage management, which can be bad for the area's appearance and cleanliness.

For this reason, The developed truck and rental management system needs to be improved in order to boost productivity, security, and operational downtime. The introduction of technology, planned maintenance, and improved waste management must be the major priorities in order to boost productivity and give truck users an improved experience.

II.2 Global Description of The Product

II.2.1 Main Functionality

The web-based transportation program offers a complete modernization solution for logistical processes. Customers may book trucks with a variety of payment choices, maintain their profiles effectively, and receive personalized truck recommendations. Real-time tracking, event notifications during deliveries, and efficient customer request handling are all advantageous to drivers. Through sensors and predictive maintenance, the vehicle management module guarantees ideal truck conditions. Meanwhile, administrators have centralized control over user and vehicle data, maintenance scheduling, and intelligent report generating. The system establishes a technologically advanced, transparent, and efficient

trucking environment by incorporating state-of-the-art technologies like image recognition and machine learning.

II.2.2 User Characteristics

Users	Responsibility	Access rights	Education Levels	Skill Levels	Experience	Type of Training
Customer	Manage profile, book trucks, track deliveries	Access to own profile, trucks available, and own booking, payment, delivery information	Varied	Varied	Varied	Online tutorials, help guides
Driver	Handle customer requests, deliver goods	Access to customer requests, delivery details, maintenance notice,	High school diploma	Commercial driving	Truck driving experience	On-the-job training, safety guidelines

		truck details				
Administrator	Manage user and vehicle data, oversee deliveries	Full access to user, vehicle, and delivery information	Bachelor's degree	Technical proficiency	Logistics management	System walkthrough, admin training modules

2.1 User Characteristics Table

II.2.3 Constraints

The constraints from the user's point of view

- **Technology Access:**

  Constraint: Users may face limitations if they do not have reliable internet access or access to devices like smartphones or computers.

- **Digital Literacy:**

  Constraint: Users with limited digital literacy may find it challenging to navigate the application's features.

- **Payment Method Limitations:**

  Constraint: Customers might face constraints if the application's payment methods are not convenient for them.

- **Device Compatibility:**

  Constraint: Users may encounter issues if the application is not compatible with their specific devices or operating systems.

- **Data Security Concerns:**

  Constraint: Users may have concerns about the security of their personal and financial information stored within the application.

- **Geographic Limitations:**

  Constraint: Users in certain geographic locations may experience limitations due to the application's availability or functionality.

II.2.4 Product Development Environment

- **Hardware :** High-performance development machines with modern processors (e.g., Intel Core i7 or equivalent), sufficient RAM (16GB or more), and SSD storage for faster development processes.
- **Software :**
  - Integrated Development Environment (IDE) Visual Studio Code for coding and debugging.
  - Version control system such as Git for collaborative development.
  - Databases such as MySQL for data storage and retrieval.
- **Connection :** High-speed internet connectivity for seamless collaboration, code sharing, and access to development tools and repositories

II.2.5 Product Operational Environment

Hardware :

- UbloxNeo-6m for tracking and getting the geolocation
- Smartphones to connect it to Computer device. Also smartphones are for taking pictures for the driver rapid checkup.

Software :

- Web servers like Apache for hosting the application.
- Database management system (DBMS) for storing operational data.
- Geospatial tools for mapping and real-time location tracking.
- Machine learning frameworks (e.g., TensorFlow, scikit-learn) for recommendation systems and rapid check-up, and maintenance notification.

Connection :

- High-speed internet connectivity for real-time data exchange between users, drivers, and administrators.
- APIs for integration with external services such as payment gateways and mapping services.
- Secure connections to protect sensitive user and transaction data.

II.3 Requirement Analysis

II.3.1 External Interface

Hardware Interface:

Processor: - Intel Inside or above

RAM: 4GB or above

SSD : 512GB or above

Software Interface:

The following are needed requirements.

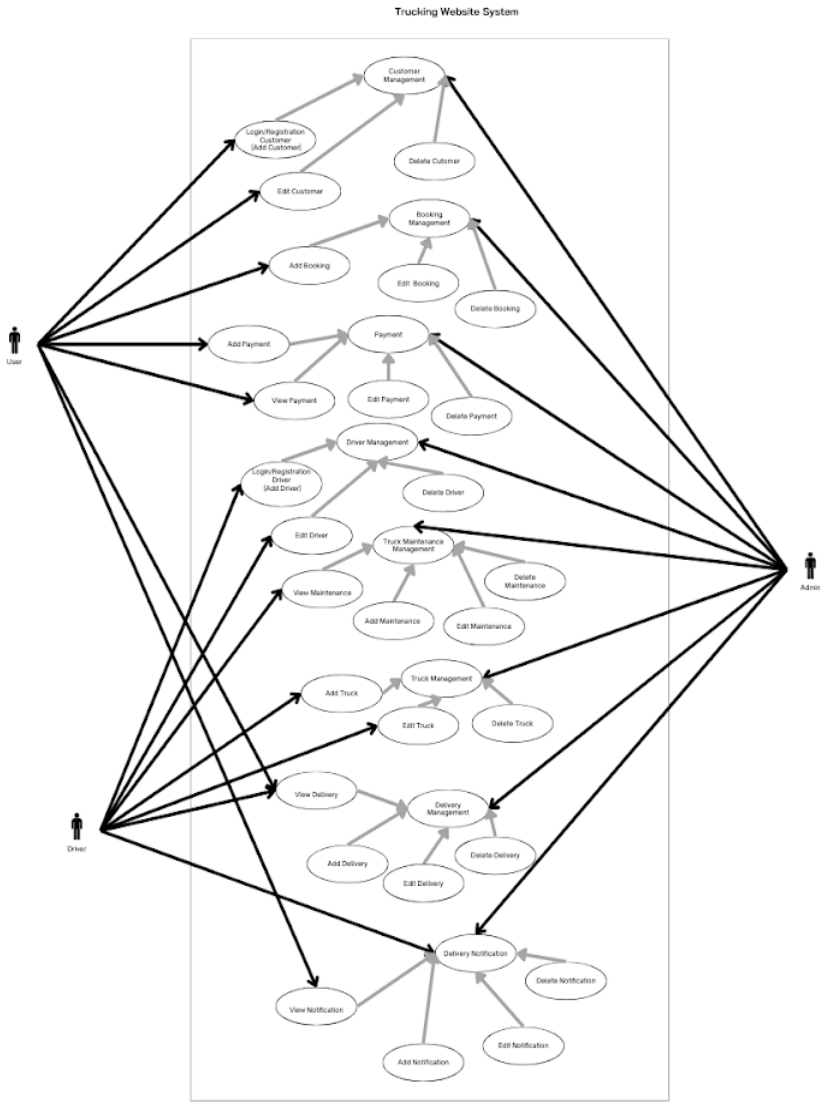
Operating system	: Windows
Development tool	: Django, Python, JavaScript, Bootstrap
Application	: XAMPP, VSCode, Arduino
Data Base	: MySQL Database Server, FireBase

Connection Interface:

Internet connection to connect between the web in one device to another device.

II.3.2 Functional Description

1. A. Use-Case Diagram



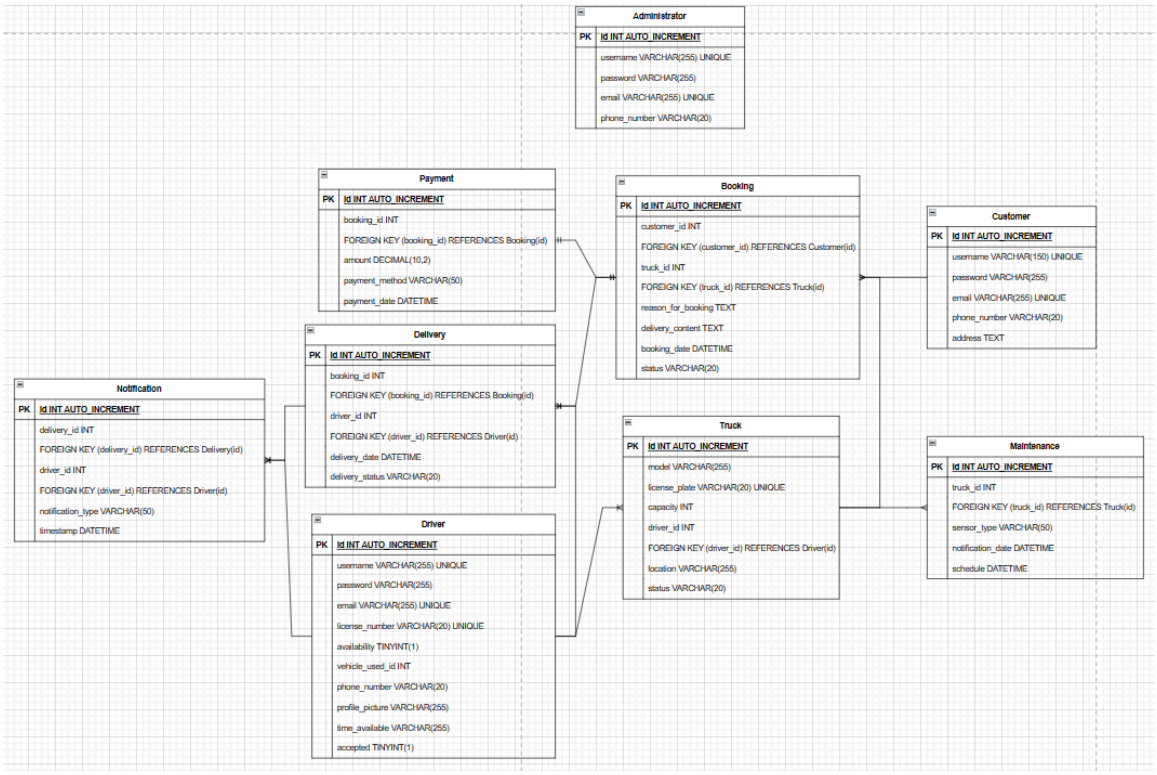
Figure

2. B. Use-Case Scenario

a. Customers will login into the application using their account, If the Customer does not have an account, they must create one. Then Login.
b. After Login, The Customer is directed to the homepage
c. The Customer can then book, explaining the details, such as: schedule and cargo
d. The system then searches for a driver, if a driver is found the customers are given the option to accept or decline the order, If the Driver declines, the customer will continue searching for a driver until they find one that accepts the order
e. The Customer then pays using their payment method of choice
f. The Driver and Customer Meetup and finalize the details of the order and starts the journey
g. During the trucks journey, Customers get live notifications of events during delivery and location data,
h. After the delivery is completed the driver will contact the customer and give them the proof via text messages.
i. The Booking Order is completed

II.3.3 Data Requirement from the user's perspective

3. A. ERD (Entity Relationship Diagram)



Figure

4. B. Data Dictionary.

Customer Table:

Attribute	Data Type	Index	Constraint
id	INT	Primary	Auto-increment

username	VARCHAR(150)	Unique	
password	VARCHAR(255)		
email	VARCHAR(255)	Unique	
phone_number	VARCHAR(20)		
address	TEXT		

Truck Table

Attribute	Data Type	Index	Constraint
id	INT	Primary	Auto-increment
model	VARCHAR(255)		
license_plate	VARCHAR(20)	Unique	
capacity	INT		

driver_id	INT		Foreign Key (Driver)
location	VARCHAR(255)		

Driver Table

Attribute	Data Type	Index	Constraint
user	INT	Primary	Foreign Key (User)
username	VARCHAR(255)	Unique	
password	VARCHAR(255)		
email	VARCHAR(255)	Unique	
license_number	VARCHAR(20)	Unique	
availability	TINYINT(1)		

accepted	TINYINT(1)		
profile_picture	VARCHAR(255)		
time_available	VARCHAR(255)		

Administrator Table:

Attribute	Data Type	Index	Constraint
id	INT	Primary	Auto-increment
username	VARCHAR(255)	Unique	
password	VARCHAR(255)		
email	VARCHAR(255)	Unique	
phone_number	VARCHAR(20)		

Booking Table:

Attribute	Data Type	Index	Constraint
id	INT	Primary	Auto-increment
customer_id	INT		Foreign Key (Customer)
truck_id	INT		Foreign Key (Truck)
reason_for_booking	TEXT		
delivery_content	TEXT		
booking_date	DATETIME		
status	VARCHAR(20)		

Payment Table:

Attribute	Data Type	Index	Constraint

id	INT	Primary	Auto-increment
booking_id	INT		Foreign Key (Booking)
amount	DECIMAL(10,2)		
payment_method	VARCHAR(50)		
payment_date	DATETIME		

Delivery Table:

Attribute	Data Type	Index	Constraint
id	INT	Primary	Auto-increment
booking_id	INT		Foreign Key (Booking)
driver_id	INT		Foreign Key (Driver)
delivery_date	DATETIME		
delivery_status	VARCHAR(20)		

Notification Table:

Attribute	Data Type	Index	Constraint
id	**INT**	**Primary**	**Auto-increment**
delivery_id	INT		Foreign Key (Delivery)

driver_id	INT		Foreign Key (Driver)
notification_type	VARCHAR(50)		
timestamp	DATETIME		

Maintenance Table

Attribute	Data Type	Index	Constraint
id	INT	Primary	Auto-increment
truck_id	INT		Foreign Key (Truck)
sensor_type	VARCHAR(50)		
notification_date	DATETIME		
status	VARCHAR(20)		

5. C. Entity tables

- **Administrator**:

  Represents the data of the Administrator type user, needed for authentication of the administrator, and information for contacting the admin.

- **Customer**:

  Represents the data of the Customer type user, will be responsible for the customer's profile which affects the recommendation for the customer.

- **Driver**:

  Represents the data of the Driver type user, will be responsible for the driver’s profile which affects the recommendation and filtering for the customer.

- **Truck**:

  Represents the data of the truck. Shows who is the driver, the location of it, the model, etc.

- **Maintenance**:

  Represents each maintenance for a truck is recorded here. This also records the condition of the truck with each sensor that might be attached to the said truck.

- **Booking**:

  For each booking details, including the customer and truck related will be recorded.

- **Payment**:

The record of the payment for the load delivery service will be here. This will be useful to create the invoice.

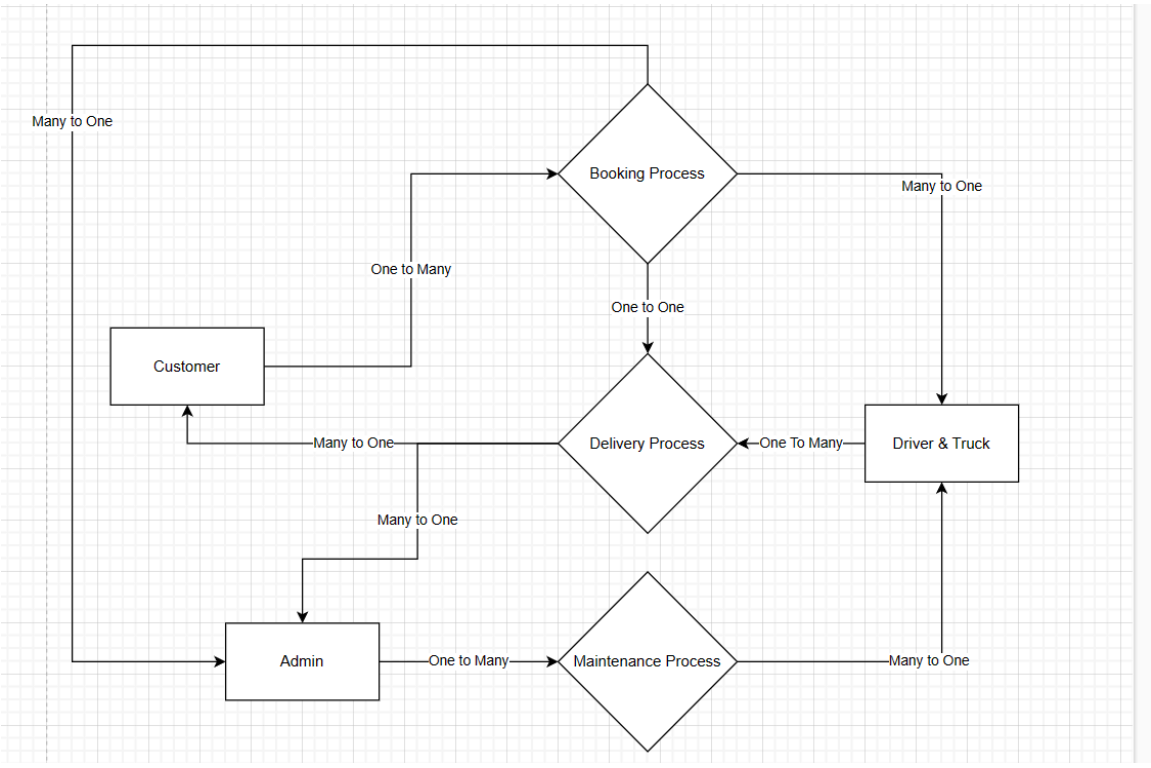
- **Delivery**:

  The delivery process will be recorded here. This includes the status of the delivery, whether it is completed or not.

- **Notification**:

  Notification of the events or troubles encountered by the driver during delivery, which if there are no events or troubles encountered, could be filled with simple announcements of the progress of the delivery.

6. D. Relations with cardinality of each Relation



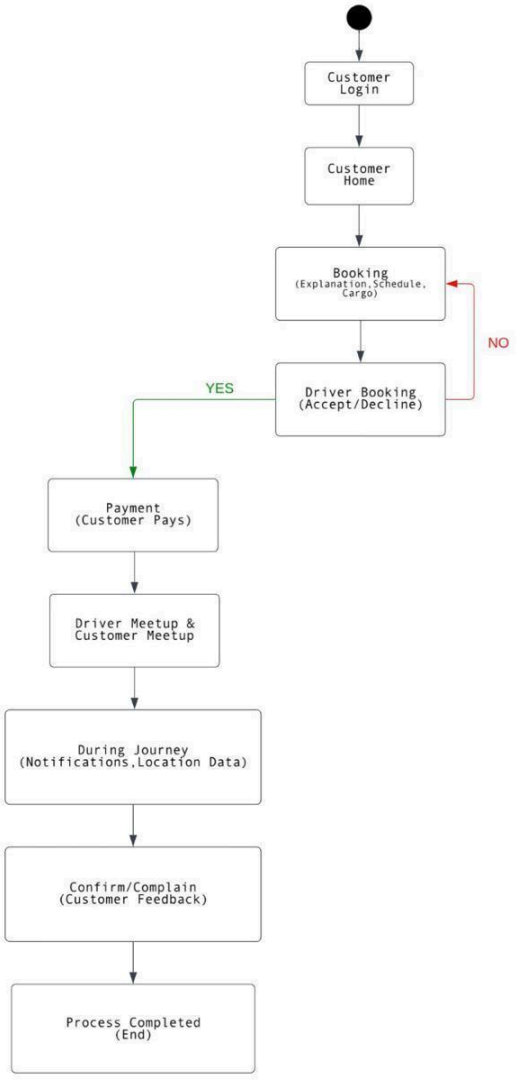
Note: Administrator can access CRUD every entity

II.3.4 Functional Requirement from the user's perspective

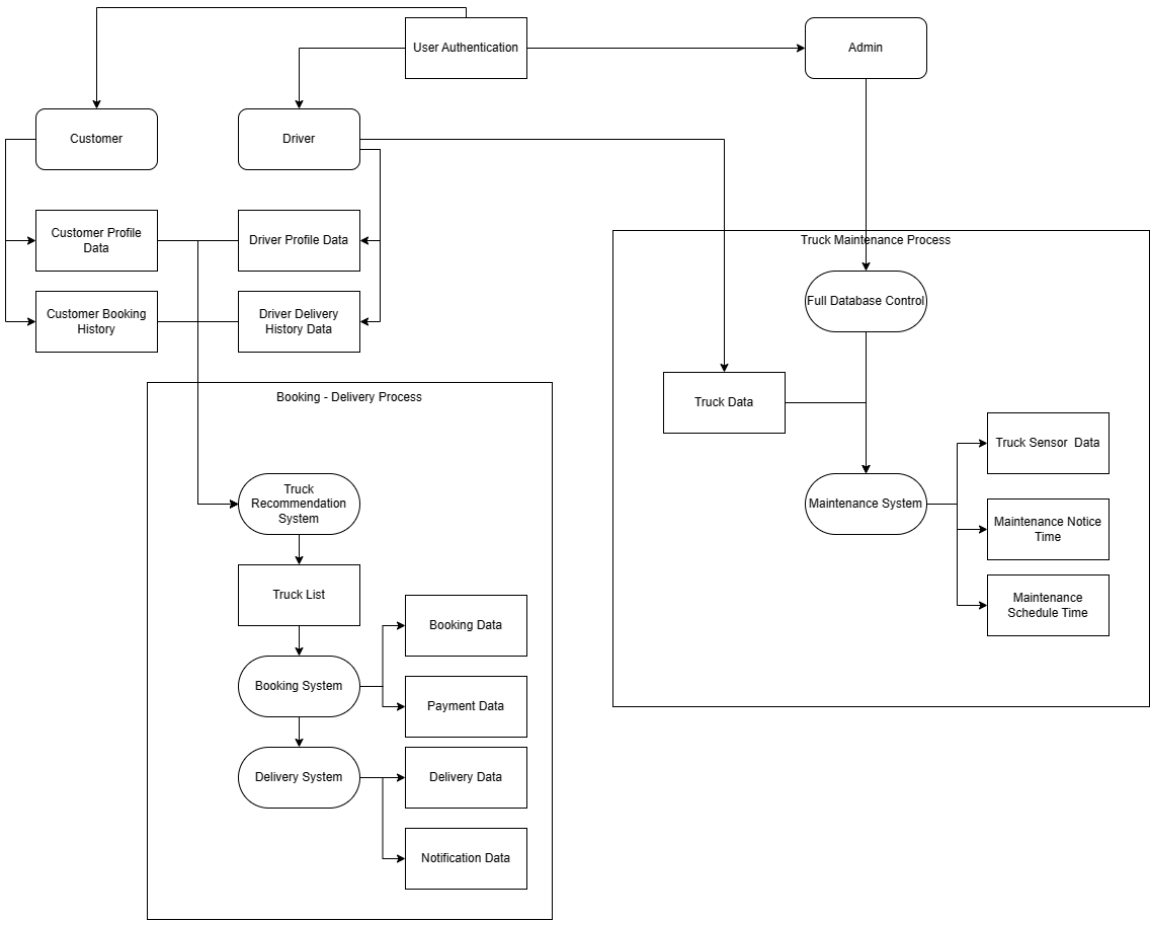
B. Activity Diagram and Data Flow Diagram

- **Activity Diagram**

Booking-Delivery Process:


Customer Login
Customer Home
Booking (Explanation,Schedule, Cargo)
NO
YES
Driver Booking (Accept/Decline)
Payment (Customer Pays)
Driver Meetup & Customer Meetup
During Journey (Notifications,Location Data)
Confirm/Complain (Customer Feedback)
Process Completed (End)

- **Data Flow Diagram**



7. C. Describe in detail the process inside the Activity Diagram and Data Flow Diagram

Activity Diagram

Booking-Delivery Process:

First, the customer will log in to the website, where they will choose a truck and book it. Then, the driver responsible for the truck will receive the booking request and choose whether to accept or reject. If reject, the customer have to choose another truck. If accept, customer will proceed to the payment process. After it is done, the customer will be given the driver's number, where they will discuss the delivery planning, and then meet up at the agreed time. When the delivery is in process, the driver will notify the customer and admin the encounters during the delivery. The truck will also give the location in which the customer and admin could check its location. When the delivery is done, the driver will give notice and proof, then the customer will have a maximum 2 days to whether complain or complete the delivery. The customer will be able to give feedback afterwards, and the process is complete.

Maintenance Process:

The first step in the maintenance process is gathering sensor data for the machine learning system. Administrators are alerted and maintenance is scheduled if a possible issue is found. The maintenance driver receives notifications and has the option to accept or decline the work. If authorized, the driver visits the site to carry out the necessary maintenance; afterward, a status update is entered into the database. If the job is turned down, the administrator modifies the timeline and sends out more notifications until it is approved. This method makes it possible to maintain ML systems effectively and on schedule.

Data Flow Diagram

There will be 2 scenarios in which the first is the booking delivery process, Truck Recommendation system will receive the data from Customer profile and their booking history, and also Driver profile and their delivery history. Then it will give the truck list which are recommendations to the customer, which when chosen they will proceed to the booking system and delivery system.

Another is the maintenance process, where the truck data(including its condition) will be given to the system. The system will have machine learning, and if it detects that the truck needs maintenance, it will give maintenance notification data, the sensor data. The admin then will add the scheduling data to the maintenance system, which the driver will receive all of the maintenance data later.

II.3.5 Non-Functional Requirement

- **Accessible From Various Browser:**

Modern web browsers such as Chrome, Firefox, Safari, and Edge should be able to access the system, ensuring that a broad user base can utilize the program without facing technological difficulties.

- **Nice User Interface:**

The application should have an intuitive and user-friendly interface to interact with users. It should provide clear instructions and guidance for users to navigate and use the features effectively.

- **Varying Payment Methods:**

The application should support various payment methods, according to the payment platform we will use, MidTrans. This ensures flexibility for users to choose their preferred payment option.

- **Device Compatibility:**

The application should be compatible with a wide range of devices, including desktops, laptops, tablets, and smartphones. Responsive design principles should be applied to ensure an optimal user experience across different screen sizes.

- **Fast Real-Time Delivery Data**:

The delivery data such as location and notification should be fast and accurate. Notifications by the driver will be scheduled sequentially. This is to ensure the customers that the delivery process is going smoothly or not.

- **Good Connection:**

The connection must be good so that the application can provide fluent and smooth delivery information to the customer and admin, especially the notification from the driver and location of the truck.

II.4 Specification Testing

Truck/Driver Recommendation System:

a. Objective: Verify that the recommendation system provides an accurate truck/driver suggestions list to the customers.

b. Method of Testing: Test whether the recommendation system could give minimum 3 accurate suggestions of trucks/drivers based on user preferences.

c. Quantity to be Measured: Successful and fitting recommendation for each customer.

Payment Processing:

a. Objective: Confirm that the payment process can be successful.

b. Method of Testing: Execute payment transactions at least 3 times.

c. Quantity to be Measured: Successful completion of payment transactions.

Vehicle Condition Testing:

3. Vehicle Condition Testing:

a. Objective: Test the condition of the vehicle by having users input their own data from Obd-2 Sensors.

b. Method of Testing: Validate the vehicle condition on based on input given by users with their own Obd 2 sensors.

c. Quantity to be Measured: Successful vehicle maintenance for each vehicle.

Vehicle Tracking:

a. Objective: Verify the system's ability to provide correct real-time vehicle locations.

b. Method of Testing: Test vehicle tracking on a minimal set of three(3) 4 or more-tyre vehicles, ensuring accurate location reporting.

c. Quantity to be Measured: Accuracy of vehicle location tracking for each vehicle.

Driver Quick-Medical Checkup:

a. Objective: Assess the system's accuracy in determining driver health based on facial expressions and eye conditions.

b. Method of Testing: Execute medical checkup tests on a minimum of 5 drivers, focusing on different facial expressions and eye conditions.

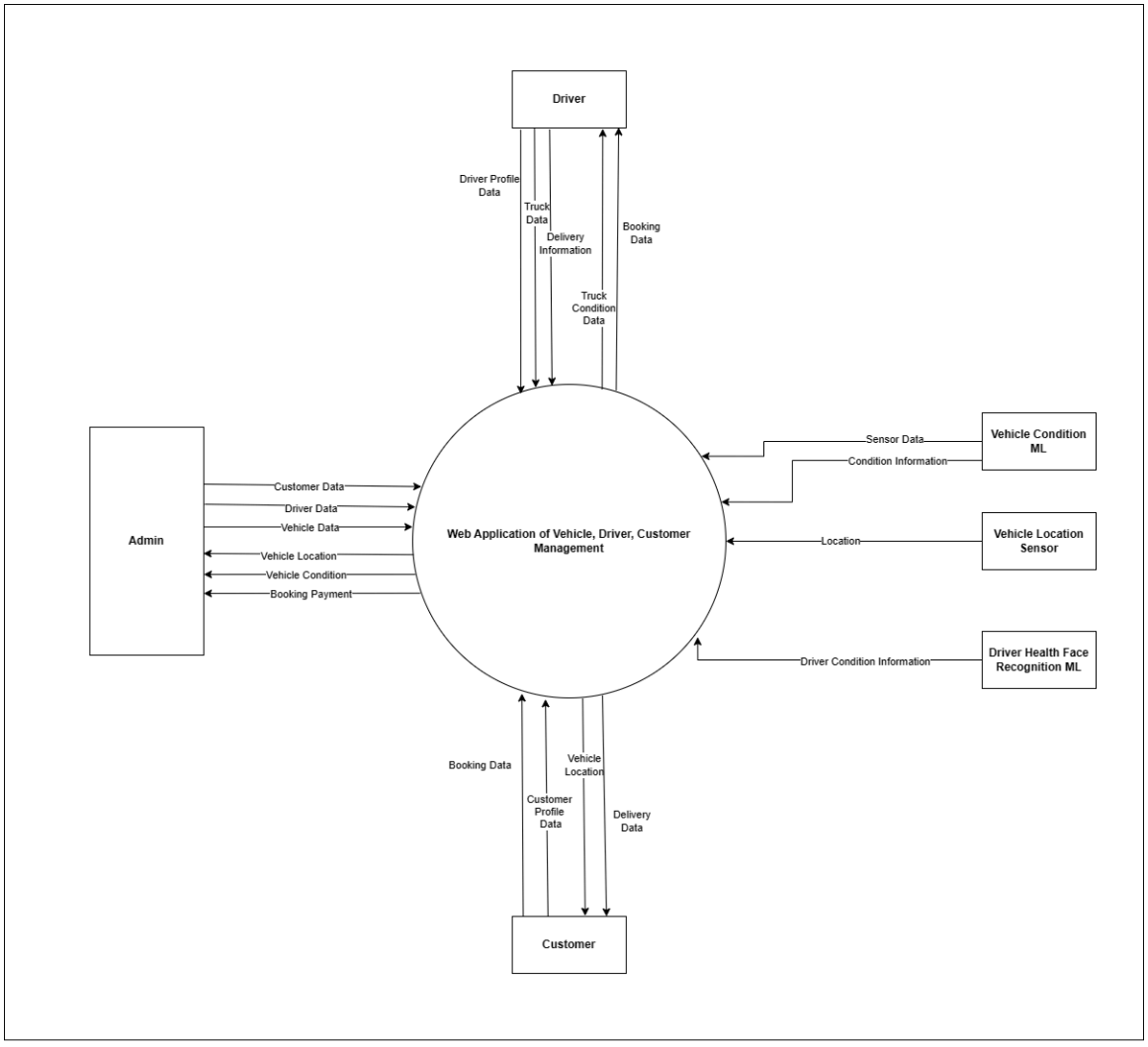
c. Quantity to be Measured: Successful health assessment for each driver tested.

CHAPTER III
F300

DESIGN

III.1 Alternative Solution Designs

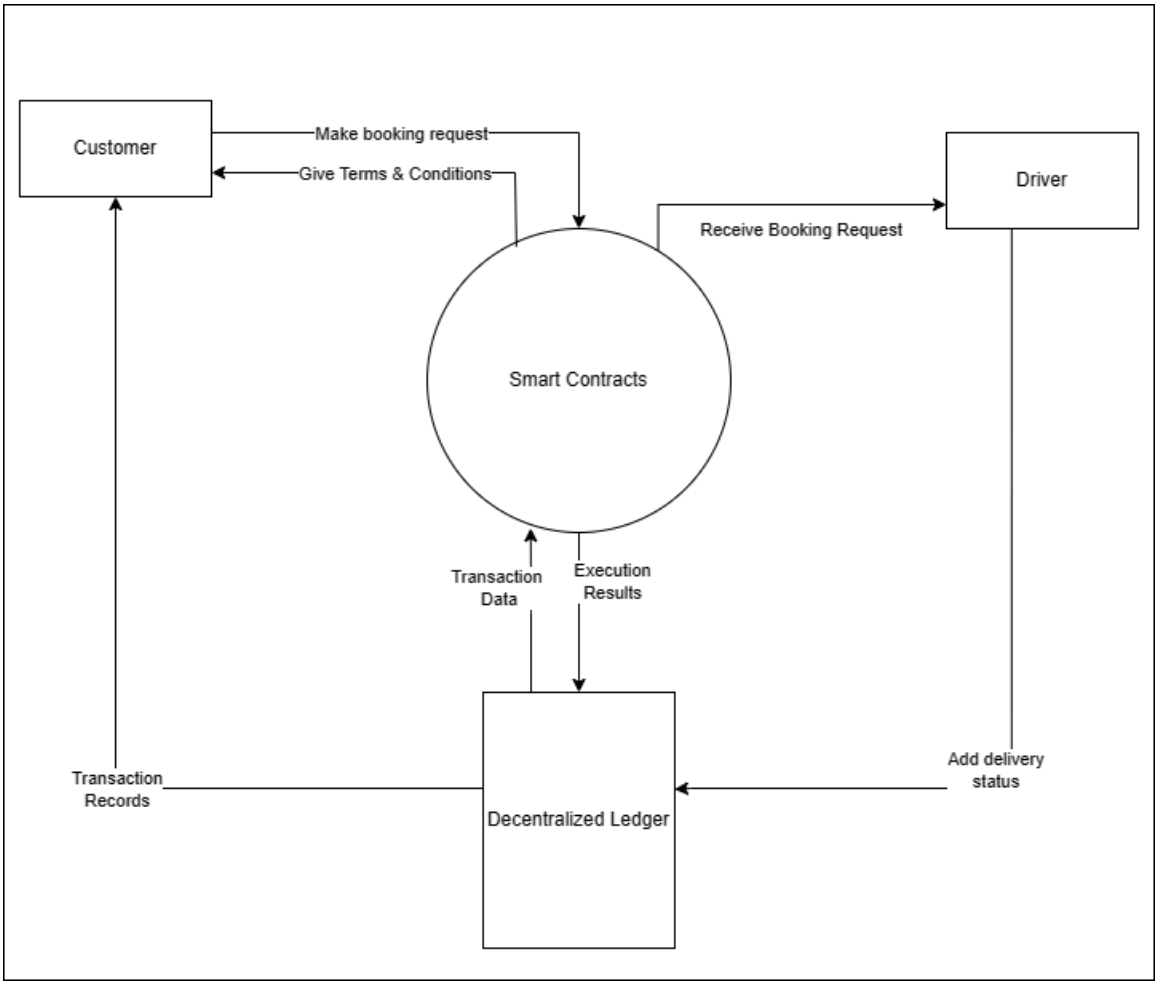
Alternative Solution 1: [*Web Based Trucking Management System*]



Alternative Solution 2

The solution is given to the problem.

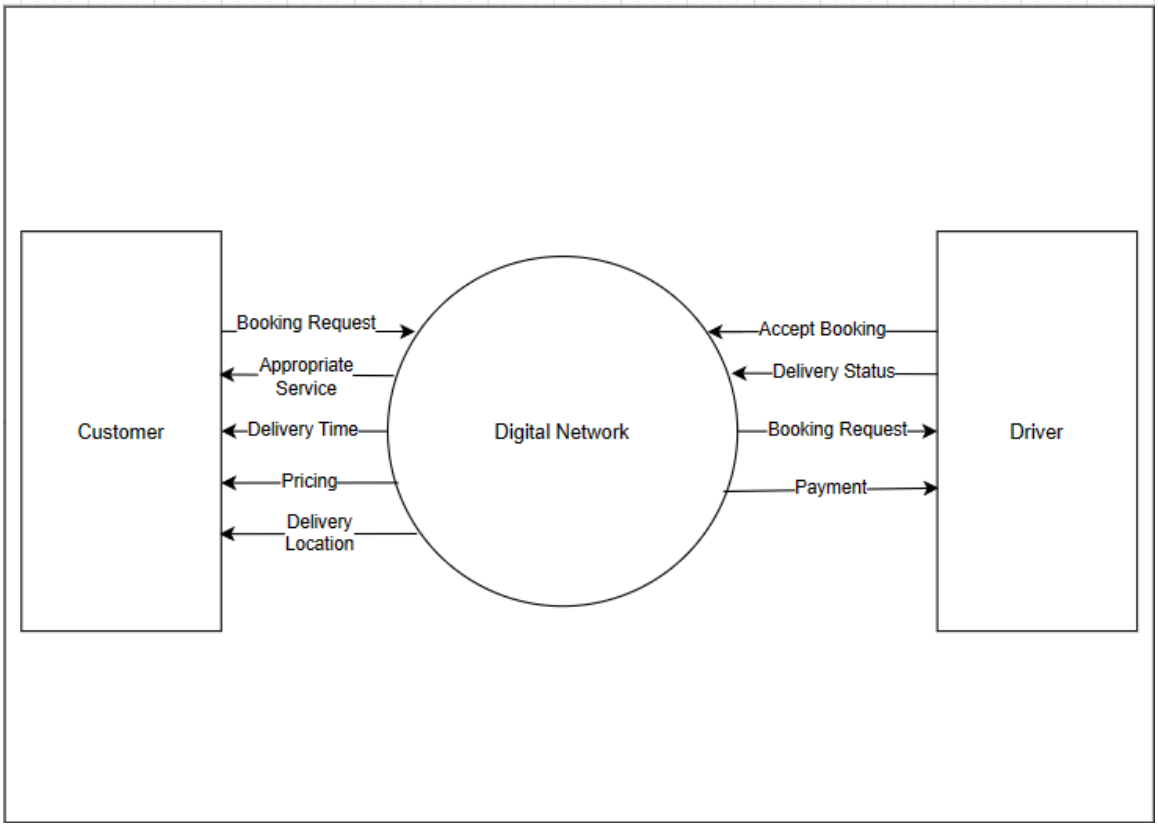
[*A blockchain-based platform*]



Alternative Solution 3

The solution is given to the problem.

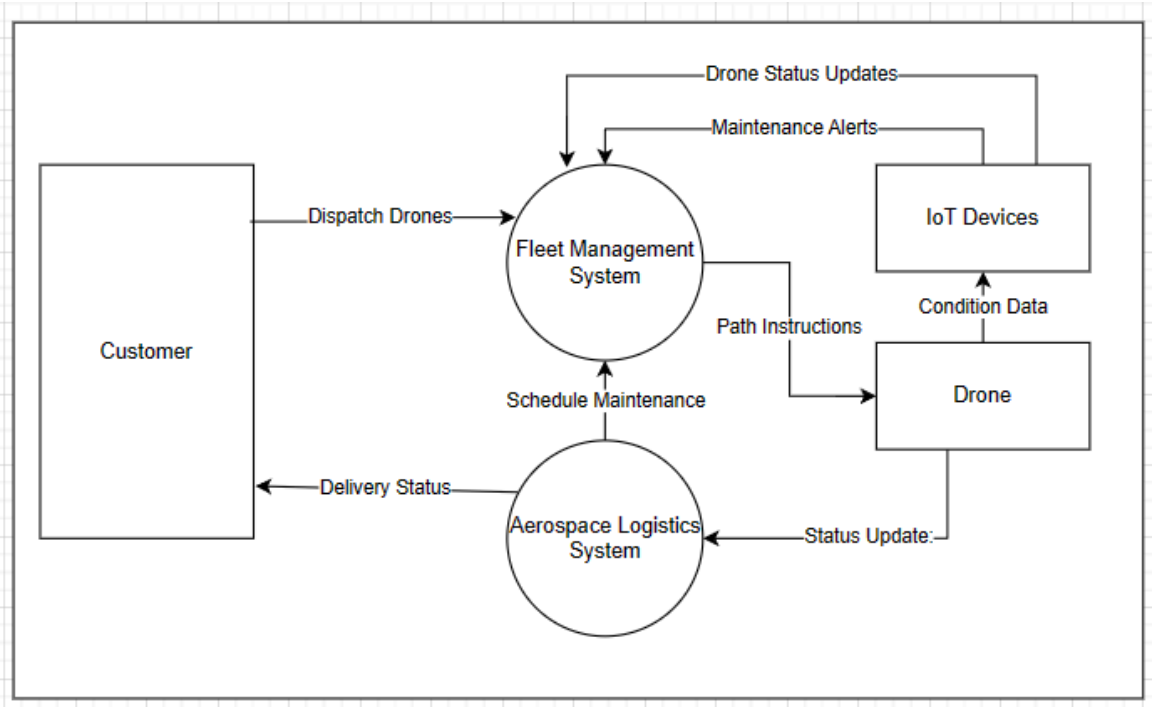
[*Digital Network connecting between haulage and freight forwarding companies*]



Alternative Solution 4

The solution is given to the problem.

[*Aerospace Logistics with Drone Technology*]



III.2 Rational/Systematic Design

To see which is the better solution for the problem of this project, some comparisons are necessary to be carried out. In which we compare 7 topics for these 4 solutions.

1. Technological Basis: The basis of what technology the product is, this summaries what technology is the solution about.
2. Core Concept: The core concept of the solution, summaries on how the technology basis will be executed.
3. Main Focus: What are the solutions and their respective applications focused at.

4. Key Technologies: The technologies that will be implemented for the alternative solution.
5. Main Goal: The target of the finished solution's application or products.
6. Potential Challenges: The potential troubles, problems, and challenges for the solution.
7. Cost Efficiency: How efficient will the cost for the solution's technologies be, a comparison on which will burn the budget the most.

III.2.1 *COMPARISON TABLE*

No.	**Comparison Entity**	**Alternative Solution 1**	**Alternative Solution 2**	**Alternative Solution 3**	**Alternative Solution 4**
1.	Technological Basis	Web Based application with advance features for multiple user types	Blockchain based platform for secure Transactions	Digital Network Between companies Mobile applications	Aerospace Logistics with Drone Technology
2.	Core Concept	Comprehensive management of logistics and trucking	Decentralized Ledger for secure Transactions	Digital Network connecting between haulage and logistic companies	Delivery using Aeroships and Drones

3.	Main Focus	Focus on ground logistics operations	Blockchain's decentralized and secure nature	Optimizing route planning	Focus on aerial logistic operations
4.	Key Technologies	real time tracking data, Data Analytics, Machine Learning	Blockchain, Smart Contracts	IoT Device, Data Analytics, Machine Learning	Drones, Airships, IoT Devices
5.	Main Goal	Customer, Driver, and Trucks Management with advanced features	Trust and Transparency in transactions	Integrated Booking and Scheduling System between multiple companies.	Transparency of Delivery items and Speed of delivery
6.	Potential Challenges	Integration of multiple modules for multiple user types. Each user type adaptation	Adoption and acceptance of blockchain in logistics industry	Getting the networking of multiple companies to make collaboration with the application	Government regulations, and local residents' acceptance. Delivery range too far.

7.	Cost Efficiency	Cost on Sensors and API	Cost on blockchain	Cost on IoT devices	Cost on Drones, Sensors, and API

III.2.2 *QUANTITATIVE SOLUTION SELECTION*

With scores ranging from 1 to 5, here provided the solution selection by scores, quantitatively

Rubics:

Criteria	**Scale**
Technological Basis	**1**. The solution relies on outdated or inefficient technologies, lacking innovation and sophistication. **2**. The solution uses basic or common technologies, but fails to leverage more advanced or cutting-edge options available. **3**. The solution adopts some modern technologies but lacks comprehensive integration or utilization of advanced features. **4**. The solution demonstrates a good understanding and implementation of contemporary technologies, incorporating some innovative elements.

	5. The solution is built on state-of-the-art and highly advanced technologies, showcasing a deep understanding of modern tools and methodologies.
Core Concept	**1**.The core concept is poorly defined or lacks coherence, making it difficult to understand or implement effectively. **2**. The core concept is somewhat clear but lacks depth or originality, resulting in a somewhat generic or unremarkable approach. **3.** The core concept is adequately defined and offers a reasonable basis for the solution, but may lack some uniqueness or innovation. **4**. The core concept is well-defined and innovative, offering a fresh perspective or unique approach to addressing the problem at hand. **5**. The core concept is exceptionally well-defined and highly innovative, representing a groundbreaking or revolutionary approach to solving the problem.
Main Focus	**1**. The solution's focus is unclear or scattered, lacking a coherent direction or purpose. **2**. The solution has some focus but may lack clarity or specificity, leading to ambiguity in its objectives. **3**. The solution's focus is reasonably clear, targeting specific aspects or areas of interest, but may lack comprehensive coverage. **4**. The solution demonstrates a strong focus on key areas or objectives, with clear priorities and well-defined goals.

	5. The solution's focus is exceptional, with a clear and comprehensive approach to addressing all relevant aspects of the problem.
Key Technologies	**1**. The solution relies on outdated or inefficient technologies, lacking consideration for more modern or effective options. **2**. The solution utilizes some basic or common technologies, but overlooks more advanced or specialized tools that could enhance its performance. **3**. The solution incorporates a mix of modern and traditional technologies, but may not fully leverage the most cutting-edge options available. **4**. The solution adopts a range of modern technologies, effectively leveraging their capabilities to enhance its functionality and performance. **5**. The solution employs state-of-the-art technologies across the board, utilizing the latest advancements to achieve optimal results and efficiency.
Main Goal	**1**. The main goal is vague or unrealistic, lacking specificity or feasibility. **2**. The main goal is somewhat defined but may lack clarity or achievable targets, leading to uncertainty in its implementation. **3**. The main goal is reasonably clear and achievable, but may lack ambition or fail to address all relevant aspects of the problem. **4**. The main goal is well-defined and ambitious, representing a clear vision for the solution's intended outcomes and impact.

	5. The main goal is exceptionally well-defined and highly ambitious, setting a bold vision for the solution's transformative potential and long-term success.
Potential Challenge	**1**. The solution is likely to encounter numerous significant challenges or obstacles that could hinder its development or implementation. **2**. The solution may face some challenges or obstacles, but they are relatively minor or manageable with proper planning and execution. **3**. The solution is expected to encounter a moderate level of challenges or obstacles, requiring some effort to overcome but not insurmountable. **4**. The solution may face few challenges or obstacles, with most issues easily addressed through effective strategies or adaptations. **5**. The solution is unlikely to encounter any significant challenges or obstacles, with potential issues easily mitigated or resolved through proactive measures and robust contingency plans.
Cost Efficiency	**1**. The solution is highly inefficient in terms of cost, requiring significant resources with little return on investment or value generated. **2**. The solution is somewhat inefficient in terms of cost, with expenses outweighing the benefits or value provided to some extent. **3**. The solution demonstrates average cost efficiency, with expenses and benefits balanced to some degree, but room for improvement exists.

	4. The solution is relatively cost-efficient, with expenses justified by the value and benefits delivered, resulting in a reasonable return on investment. **5**. The solution is highly cost-efficient, delivering exceptional value and benefits relative to the resources invested, resulting in a significant return on investment and overall affordability.

8. **Solution 1:**

No.	**Quantitative Solution Selection**	**Score**
1.	Technological Basis	3
2.	Core Concept	4
3.	Main Focus	4
4.	Key Technologies	4
5.	Main Goal	4
6.	Potential Challenges level of convenience	4
7.	Cost Efficiency	3
	Total	**26**

9. **Solution 2:**

No.	Quantitative Solution Selection	Score
1.	Technological Basis	4
2.	Core Concept	3
3.	Main Focus	4
4.	Key Technologies	4
5.	Main Goal	3
6.	Potential Challenges level of convenience	3
7.	Cost Efficiency	2
	Total	**23**

10. **Solution 3:**

No.	Quantitative Solution Selection	Score
1.	Technological Basis	2
2.	Core Concept	4
3.	Main Focus	2

4.	Key Technologies	3
5.	Main Goal	3
6.	Potential Challenges level of convenience	4
7.	Cost Efficiency	4
	Total	**22**

-
- **Solution 4:**

No.	**Quantitative Solution Selection**	**Score**
1.	Technological Basis	5
2.	Core Concept	2
3.	Main Focus	3
4.	Key Technologies	4
5.	Main Goal	4
6.	Potential Challenges level of convenience	2
7.	Cost Efficiency	3
	Total	**23**

III.2.3 *SOLUTION SELECTION*

The selection of alternative solution 1 is based on a thorough assessment of our projects' needs, specifications, and objectives. It also has the best score in evaluation among all the comparisons of different alternative solutions. We are aiming for a platform for logistics and trucking management, and solution 1 is great for numerous reasons.

This solution excels in having an easy to use interface for customers, especially because the recommendations of trucks for the customers, it will help them pick the vehicles they might need. There is also a filtering method to filter which pair of vehicles and drivers are appropriate for the client's aim. Moreover, solution 1 fulfills the need for active vehicle maintenance notice by using sensor data online and machine learning.

The transparent communication between customers and drivers met our goal of enhancing information sharing, which is vital for deliveries. Then, the Administrator or staff is beneficial for monitoring and giving schedules for maintenance, as well as managing the database. The administrator or staff will need to actively make contributions in all situations, rather than only handling and managing the database.

The focus of solution 1 is on ground-based logistics, not aerial strategies or drones like solution 4. Then, despite not using the blockchain like solution 2, we strategically implemented transaction functions for ease and convenient payments, catering to the financial aspects of the industry. Also, instead of connecting customers with logistics companies, as in solution 3, this solution is a web application that connects customers with drivers that may use our application.

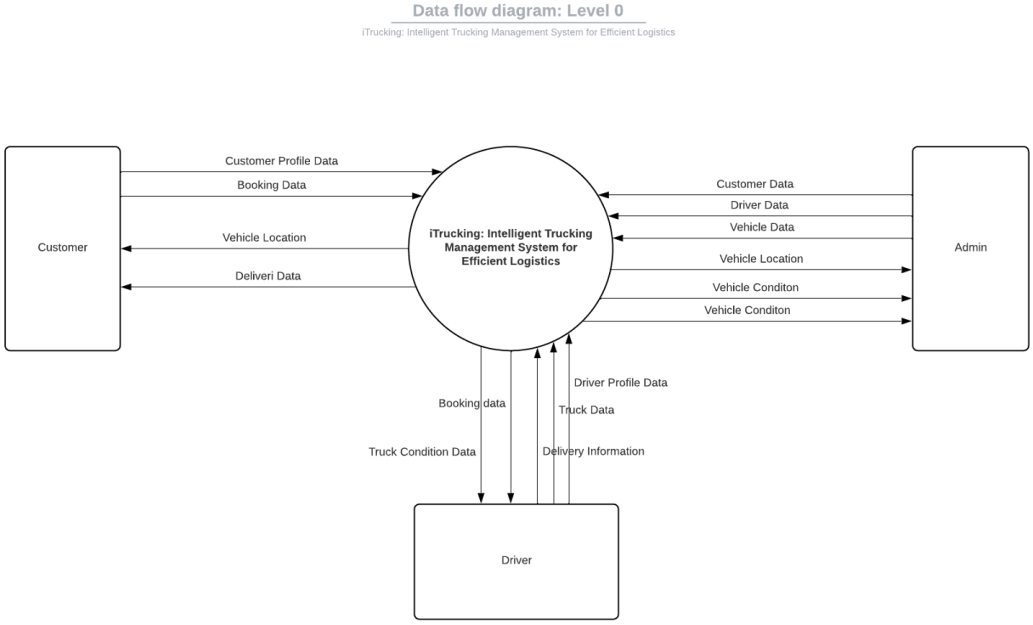
This is why our choice of answer 1 became the end result of a thoughtful evaluation of our undertaking's desires and constraints. Using a combination of capabilities, it provides a complete logistics and trucking management system tailored to the specific needs of the industry.

III.3 Hierarchical/IterativeDesign

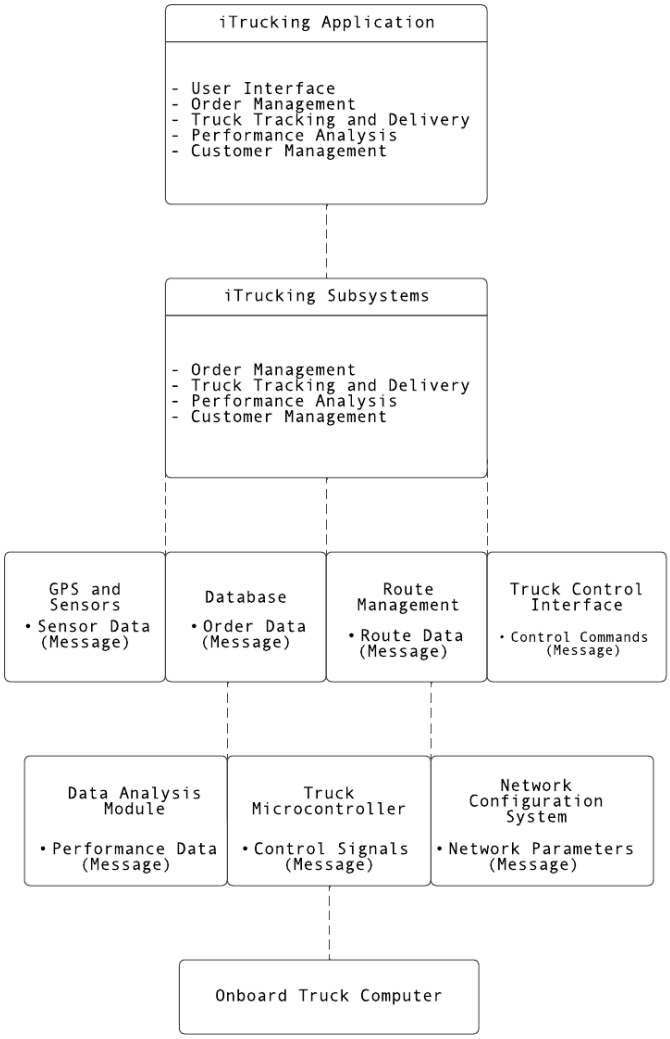
ii. 1. Data Flow Diagram iTrucking: Intelligent Trucking Management System for Efficient Logistics.

On the data flow diagram we used the structured system analysis and design method (SSADM) by use Yourdoun and coad it is for the process to use a circle. Here's i show the requirement of yourdoun and coad.

A. Data Flow Diagram Level 0



iii. 2. Interface information between block diagrams. (example: Global variable, message passing, parameter passing).



1. GPS and Sensors Subsystem - Database Subsystem:
   Interface: Message Passing
   Information Flow: Sensor data from GPS and sensors subsystem is sent as messages to the database subsystem. This includes data such as location information and order details.

2. Database Subsystem - Route Management Subsystem:
   Interface: Message Passing
   Information Flow: Order data stored in the database subsystem is passed as messages to the route management subsystem to optimize route planning based on current orders.

3. Route Management Subsystem - Truck Control Interface:
   Interface: Message Passing
   Information Flow: Route data generated by the route management subsystem is sent as messages to the truck control interface to provide guidance to trucks.

4. Data Analysis Module - Truck Microcontroller:
   Interface: Message Passing
   Information Flow: Performance data analyzed by the data analysis module is passed as messages to the truck microcontroller to optimize truck performance and efficiency.

5. Data Analysis Module - Network Configuration System:
   Interface: Parameter Passing
   Information Flow: Network parameters needed for data analysis are passed from the network configuration system to the data analysis module to ensure proper connectivity and data access.

iv. 3. Implement steps in Software Engineering Design as your design documentation.

Software Engineering Design process using the waterfall

1. Requirements Analysis:
   - Identify and gather requirements from stakeholders regarding the functionalities and features of the iTrucking application.
   - Define user stories or use cases based on the requirements provided.
   - Analyze the requirements specifically for the following functionalities:
     - User authentication (login and account creation)
     - Homepage navigation
     - Booking process (including details such as schedule and cargo)
     - Driver assignment and order acceptance
     - Payment processing
     - Live notifications during delivery
     - Proof of delivery via text messages

2. System Design:
    - Design the overall system architecture, including the frontend and backend components.
    - Design the database schema to store user account information, booking details, driver information, payment details, etc.
    - Design the user interface for login, homepage, booking process, payment, and notification displays.
    - Define the algorithms and logic for driver assignment and order acceptance.
    - Design the communication protocols for sending live notifications to customers during delivery.
    - Determine the integration points with external services for payment processing and SMS messaging for proof of delivery.

3. Implementation:
    - Develop the frontend components including login/signup pages, homepage, booking form, payment form, and notification displays using appropriate technologies (e.g., HTML, CSS, JavaScript).
    - Develop the backend components including server-side logic, database interactions, and integration with external services using suitable programming languages and frameworks (e.g., Node.js, Express.js, Django, Flask).
    - Implement user authentication functionalities for login and account creation.
    - Implement the booking process logic to allow customers to book trucks, enter details, and submit orders.

- Implement the driver assignment algorithm to match drivers with orders and handle order acceptance/decline.
- Integrate payment processing functionality using payment gateways or APIs.
- Implement live notification mechanisms to send updates to customers during delivery.

4. Testing:
    - Perform unit testing to ensure each component functions correctly in isolation.
    - Conduct integration testing to verify that the frontend and backend components interact properly.
    - Perform system testing to validate the overall system behavior against the requirements.
    - Conduct user acceptance testing (UAT) with stakeholders to ensure the application meets their expectations.

5. Deployment:
    - Deploy the application to a production environment, ensuring scalability, reliability, and security.
    - Monitor the application for any issues or performance bottlenecks post-deployment and address them accordingly.

6. Maintenance:
    - Provide ongoing support and maintenance for the application, addressing any bugs or issues reported by users.

- Implement feature enhancements or updates based on user feedback or changing requirements.
- Regularly review and update the application to ensure it remains secure and up-to-date with technology advancements.

v. 4. Provide the details of component references and libraries used.

List of used components or tools

Here provided is the list of Hardware components and tools that will be used for the project to run well

- UbloxNeo-6m: A very capable GPS for the price and size. Compatible with UART capable devices including Arduino, Raspberry Pi, MSP430, and MSP432. Power the module and it will automatically acquire satellite signals and a position fix. Once it has a position fix the module will blink the on board LED
- SIM800L GSM Module: This is a compact and powerful device that enables communication over GSM networks. It is a key component in many IoT and remote communication projects due to its versatility and ease of use.

List of used library and frameworks

Here provided is the list of the frameworks and libraries, which are software tools for the coding side of the project in order for it to run and work

- asgiref version 3.7.2: Framework for building async web apps, enabling handling of multiple requests without blocking resources.

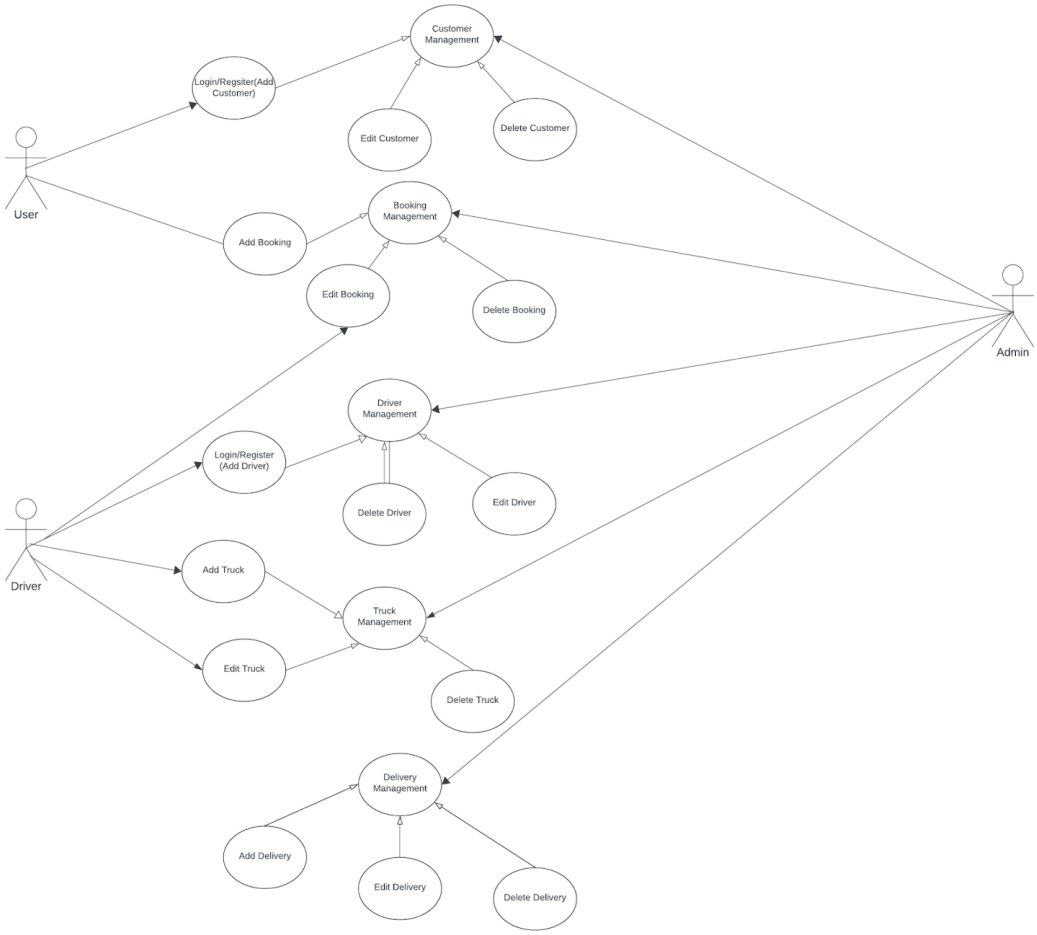
- Authlib version 1.3.0: Python library supporting authentication mechanisms like OAuth and JWT for securing web applications.
- certifi version 2023.11.17: Library ensuring secure SSL/TLS connections by providing a collection of trusted CA certificates.
- cffi version 1.16.0: Python package facilitating the calling of C functions from Python for improved performance and flexibility.
- charset-normalizer version 3.3.2: Python library for normalizing character encodings, ensuring consistent handling of text data.
- cryptography version 41.0.7: Python library offering cryptographic functions like encryption, decryption, and secure communication protocols.
- django-chartjs version 2.3.0: Integration of Chart.js into Django projects for creating dynamic and interactive charts.
- django-extensions version 3.2.3: Django package providing various useful extensions and utilities to streamline development.
- django-jazzmin version 2.6.0: Customizable Django admin interface with modern design and additional features for enhanced user experience.
- djangorestframework version 3.14.0: Toolkit for building Web APIs in Django, simplifying the creation of RESTful endpoints.
- Django 5.x: Latest version of the Django web framework, renowned for its simplicity and scalability in building web applications.
- Faker version 22.2.0: Python library generating fake data for testing and development purposes.
- faker-vehicle version 0.2.0: Faker extension specializing in generating fake vehicle-related data for testing scenarios.
- idna version 3.6: Library handling Internationalized Domain Names (IDNA) in Python applications, ensuring proper domain name handling.

- model-bakery version 1.17.0: Tool for creating model instances in Django tests, simplifying the setup of test data.
- mysqlclient version 2.2.1: MySQL database adapter for Python, enabling interaction with MySQL databases from Python code.
- pillow version 10.2.0: Python Imaging Library (PIL) fork providing image processing capabilities for Python applications.
- pyserial version 2.7: Python library for serial communication, enabling interaction with serial ports and devices.
- python-dateutil version 2.8.2: Python library for parsing, manipulating, and working with dates and times.
- python-dotenv version 1.0.0: Python library for managing environment variables from a .env file, simplifying configuration management.
- pytz version 2023.3.post1: Python library for working with time zones, providing functionality for converting and handling time zone-aware datetime objects.
- requests version 2.31.0: Python library for making HTTP requests, simplifying interaction with web servers and APIs.
- six version 1.16.0: Python compatibility library for bridging differences between Python 2 and Python 3 codebases.
- sqlparse version 0.4.4: Python library for parsing SQL queries, facilitating analysis and manipulation of SQL code.
- tzdata version 2023.4: Time zone database for Python applications, ensuring accurate handling of time zone information.
- urllib3 version 2.1.0: Python library providing a higher-level interface for making HTTP requests, built on top of the standard library's urllib.
- scikit-learn: for machine learning for the data obtained from the obd-2 sensor data online

- Tensorflow
- opencv-contrib-python
- numpy
- SoftwareSerial : SIM800L GSM Module and NeoGPS Module using the appropriate libraries (SoftwareSerial for SIM800L and AltSoftSerial for NeoGPS).
- AltSoftSerial, TinyGPS: using the NeoGPS module for GPS data and the SIM800L GSM module for GSM communication.

■

vi. 5. Use -Case Diagrams



III.4 Verification Demonstration and Proof of Design Concept

Here provided is a simulation on how the application would work. Testing will simulate the following scenario acted by three different users (Admin, Driver, and Customer). Tested by 3 people.

Actor	Simulation Scenario	Simulation Result
Admin	1. Registration and Login process, inputting the fields using correct credentials of an admin. 2. Viewing the customer profile data inside the administration. 3. Check the driver data and see whether there are any driver requests. 4. Interview the requestor. 5. Check whether the drivers had submitted the required documents.	1. User is thrown into the administration page and asked to log in as an admin by inputting the previous admin credentials. 2. Correct view of customer data. 3. There is someone who requests a driver. 4. Interview runs perfectly, the requestor is accepted as a driver, the driver then is asked to submit the remaining documents and pictures. 5. The drivers had submitted the required documents, making the driver available.
	Simulation Process: *Booking-Delivery*	
	6. Check whether there are any bookings ongoing, verify the payment of the ongoing booking. 7. Create a chatting channel for customer and driver using a generic chatting	6. Payment verified and money received successfully. 7. Customer and driver will converse about the delivery inside the said channel.

	application specific for conversing about the delivery. 8. Before delivery, the driver will have a quick checkup. Verify on the checkup. 9. During delivery, monitor the map, and check the vehicle. Expect to receive messages from the driver when the truck stops for a long duration. 10. After the delivery is complete, verify with the driver and wait for 2-5 days for confirmation from the customer.	8. Checkup verified, driver will proceed for the delivery. 9. Map will show the geographical location and the truck location. Receive message from the driver with a minimum of 4 times containing information about the encounters during the delivery. 10. Customer confirmed the delivery, delivery process complete.
	Simulation Process: *Truck Maintenance*	
	11. Check whether there are any vehicle maintenance notifications. 12. Turn off the availability of the truck and driver, and message the driver for a schedule to do the maintenance.	11. Vehicle maintenance notification shows, check on the vehicle's condition and notify the driver. 12. Turn off the availability of the truck and driver, and message the driver on a schedule to do the maintenance.
Driver	1. Register as a driver. 2. Interview completed and user is accepted as driver, now the user is logging in as driver.	1. Registration completed successfully, wait for the news from admin for the interview.

	3. In the home page, there are pages which are delivery, booking list, delivery history, maintenance schedule, and view truck.	2. Getting inside the website home, driver's side. 3. In the home page, there are pages which are delivery, booking list, delivery history, maintenance schedule, and view truck. Delivery will show the information of current delivery, it will be empty if no delivery is in progress. Booking list will show a list of bookings received which could be accepted or declined, as well as the information of the booking. Delivery history will show the list of delivery information of the past deliveries. Maintenance schedule will show the schedule for vehicle maintenance if exists. View vehicle will show the conditions and information of the driver's truck.
	Simulation Process: *Booking-Delivery*	
	4. Open booking list to check if there are any customer asking for a delivery. 5. Talk with the customer about the delivery in the chat group made by admin.	4. A customer asked for a booking, view the booking information. It is acceptable, so accept the job.

	6. Before delivery, the driver will have a quick checkup. Use the application to check on the driver's current condition based on their face. 7. Drive safe during delivery. Every stop, give message to both the customer and admin minimum 4 times for each of them. 8. After the delivery is complete, take some picture as a proof of delivery completion.	5. Talk about the details on the delivery with the customer, schedule a meetup for the delivery. 6. The machine shows that the driver is fit for the delivery, proceed with the delivery. 7. Delivery complete with both the customer and admin is notified every stop. 8. Wait for 2-5 days for any customer complaints. If there exists, the admin will notify the driver and all actors will try to resolve the issue.
	Simulation Process: *Truck Maintenance*	
	9. Occasionally check on the vehicle condition by uploading excel file, if it is bad, expect the admin to contact soon 10. Decline any bookings received, and schedule a maintenance date with the admin.	9. Vehicle condition is bad, received a notification from the admin. 10. Maintenance will proceed as scheduled with the admin, if there are nothing interferes with the plan.
Customer	1. Register and login as a customer with correct credentials. 2. Check the recommendation page.	1. User sent to customer home page. 2. Get recommendations of trucks based on the customer's database and driver

	3. Open the truck list page, and use the filter to find the suitable truck. 4. Choose the truck. View the truck's driver overview. 5. Fill in the booking form. Then wait for the driver to accept. 6. Pay according to the distance of the delivery. 7. Talk with the driver about the delivery details inside the group chat or private chat. 8. During delivery, monitor the movement of the truck with the map. Expect minimal 4 messages from the driver at every stop. 9. After the delivery is complete, expect the driver to give the proof. Customer could complain within 2-5 days, if no complains, confirm the completion of delivery.	database, as well as the vehicle database. Customers find trucks that fit their preferences, but still want to choose the trucks theirself. 3. Customers find their preferred truck and click it. 4. The driver seems fit for the job, and the customer proceeds to book. 5. The driver accepted the booking. 6. Payment will be verified by the admin. 7. The chat contains delivery details, meetup schedule, or anything related to the delivery and booking. 8. Map will provide a real time location of the truck, and messages are received a minimum 4 times during each long stop of the truck. 9. Verify the delivery in 2-5 days in which the customer could file a complaint of the delivery, the load, or even the lack of messages received to the admin. The admin will then forward the message to the driver. Confirm the completion otherwise.

This simulation scenario contains the main function of the application, which is the whole booking and delivery functions, and also the truck maintenance function. These functions will then include the modules stated in the first document (F100).

III.5 Standards Used

For a truck management system, we have specific standards and considerations.
This project has a complex set of requirements of regulations, so we have provided a list tailored to it.

Telematics standards:

SAE J1939: Follow with the same old truck and bus control communications to permit actual-time tracking and communication between various vehicle structures.

Electronic Logging Device (ELD) standards:

Adhere to the requirements set by the Federal Motor Carrier Safety Management (FMCSA) to ensure compliance with electronic logging requirements.

Fleet management system (FMS) standards:

ISO 15143-3: put into effect the usual for records communique in a fleet management machine, ensuring interoperability among wi-fi additives.

Maintenance and Diagnostics standards:

SAE J1708 and SAE J1587: observe those standards for truck diagnostics and protection information conversation.

Geospatial data requirements:

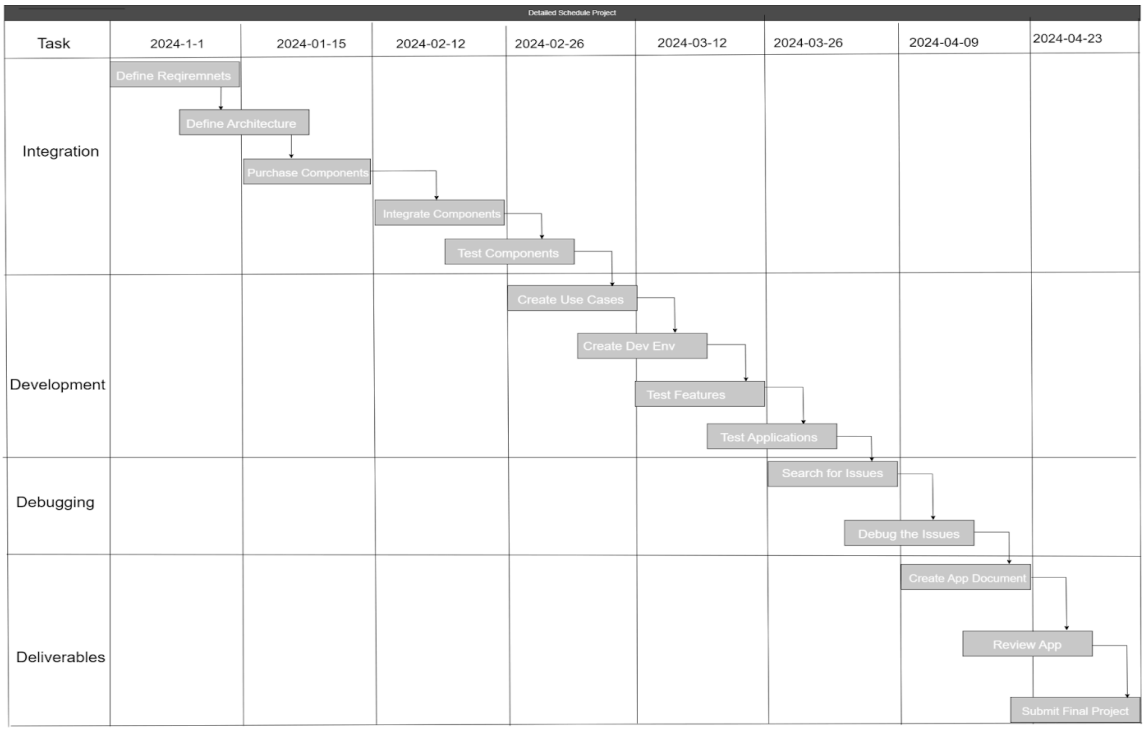
Use standardized geospatial records codecs like GeoJSON for place-primarily based facts and mapping.

Additionally, Below is the standard list used for the document design:

- UML Diagram 2.0
- Data Flow Diagram

III.6 Implementation and Testings Plans

III.6.1 *Gantt Chart*



III.6.2 *Expected Outcome*

No	Task	Description	Date Started	Project Completion Percentage
1.	Define Requirements	Define the requirements of our project	2024/01/01	0%
2.	Define Architecture	Define the architecture of our project and adjust it with our project requirement	2024/01/8	0%
3.	Django Studying	Study the necessary knowledge to make the website	2024/01/15	5%
4.	Create Website	Start to create the website using the knowledge learnt before	2024/02/12	10%

6.	Create Use Cases	Create and define various use cases for our applications using the components purchased. (Such as: users, drivers, etc.)	2024/02/26	15%
7.	Create Module Integration	Start to integrate each of the other member's module into the website	2024/02/26	70%
8.	Test Features	Test all the features that will be inserted into the main project, ensure that all features are working properly	2024/03/12	72%
9.	Test Applications	Test the entire application with the features added.	2024/03/12	78%
10.	Search for Issues	Search for issues in the applications that have been looked over in development	2024/03/26	80%

11.	Debugging the Issues	Debug issues found in the application in the previous step.	2024/03/26	90%
12.	Create Application Document	After Fixing all the bugs, create detailed documentation about our application.	2024/04/09	95%

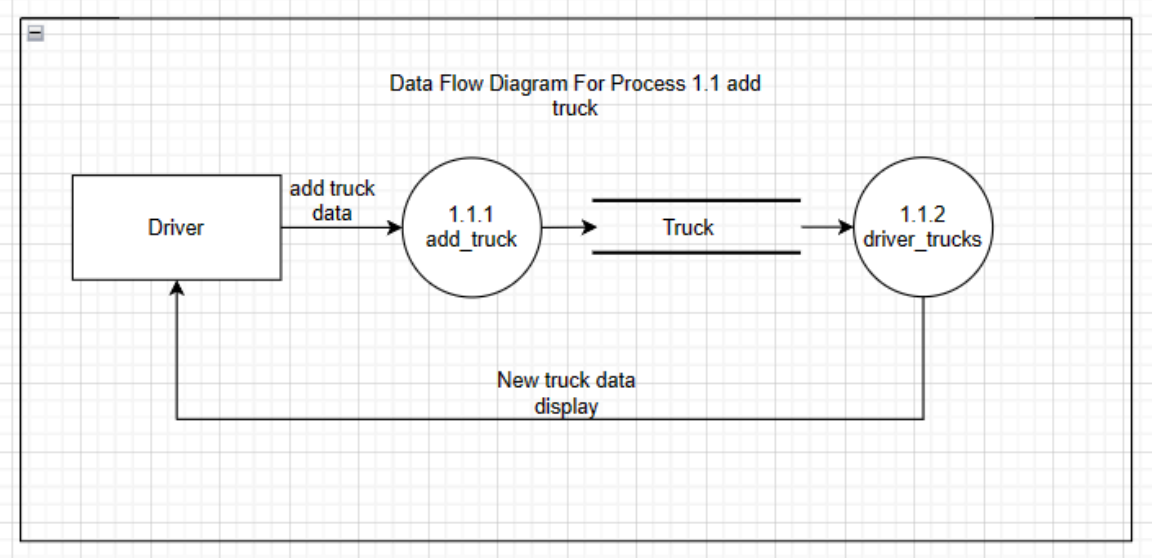
CHAPTER IV

IMPLEMENTATION

IV.1 Designs Implementation

IV.1.1 Functions/Procedure/Class

1. DFD Level 3 for add truck data



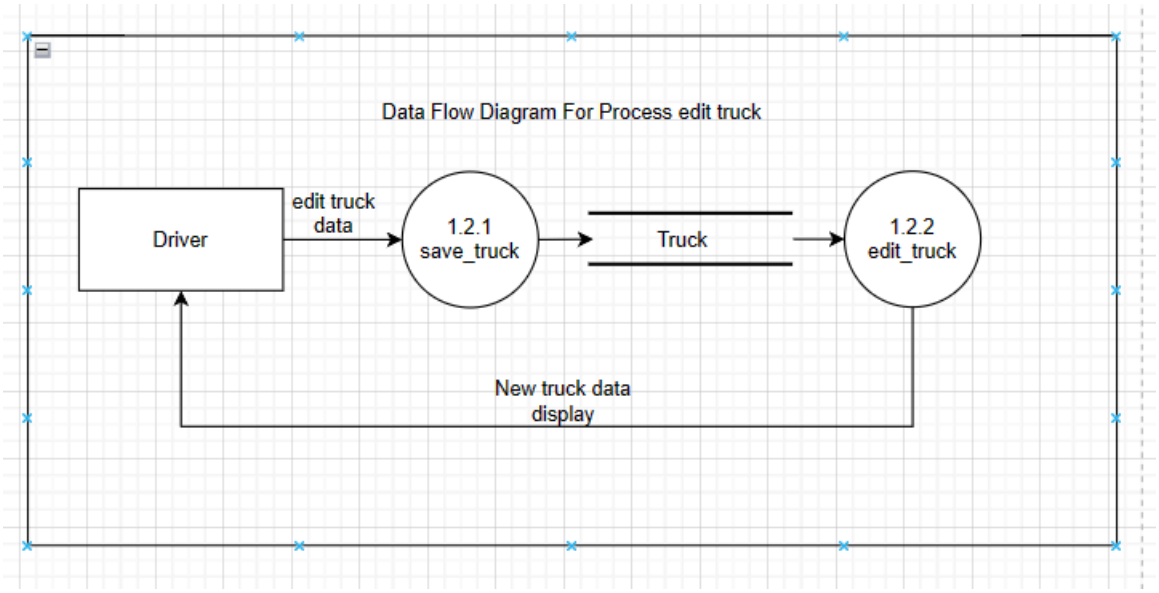
a. code:

```
@login_required(login_url = 'login')
@user_passes_test(im_driver, login_url='/')
def add_truck(request):
    if request.method == 'POST':
        form = TruckForm(request.POST, request.FILES)
        if form.is_valid():
            if form.is_valid():
                truck = form.save()

                return JsonResponse({
                    'status': 'success',
                    'truck': {
                        'id': truck.id,
```

```
                'overall_view_url': truck.overall_view.url,
                'front_view_url': truck.front_view.url,
                'side_view_url': truck.side_view.url,
                'back_view_url': truck.back_view.url,
                'top_view_url': truck.top_view.url,
                'truck_available': truck.truck_available,
                'truck_accepted': truck.truck_accepted,
                'status': truck.status,
                'truck_model': truck.truck_model,
                'license_plate': truck.license_plate,
                'capacity': truck.capacity,
            }
        })
    else:
        return JsonResponse({'status': 'error', 'errors': form.errors})
else:
    form = TruckForm()
return render(request, 'driver/driver_truck.html', {'form': form})
```

Explanation:
login_required requires the user (which is driver) to log in first in order to be able to execute the function above. Then the user_passes_test, is a function that tests if the user is a driver or not, if it is not a driver, then the user can't proceed to the function. if request.method == 'POST': This checks if the HTTP method of the request is POST. form = TruckForm(request.POST, request.FILES): This creates a new instance of TruckForm, populated with the POST data and files from the request. If form.is_valid(): This checks if the form data is valid. If it is, the form data is saved as a new Truck object, and a JSON response is returned with the details of the new truck. If the form data is not valid, a JSON response is returned with the status 'error' and the form errors. else: form = TruckForm(): If the HTTP method of the request is not POST (i.e., it's a GET request), a new, empty TruckForm is created. return render(request, 'driver/driver_truck.html', {'form': form}): This renders the 'driver/driver_truck.html' template with the form as context. This is the response to a GET request.

2. DFD Level 3 for edit truck data



a. code:

```
@login_required(login_url = 'login')
@user_passes_test(im_driver, login_url='/')
@require_GET
def edit_truck(request, truck_id):
    try:
        driver = Driver.objects.get(user=request.user)
    except ObjectDoesNotExist:
        # Handle the case where the driver does not exist
        return redirect('go_home')

    request.session['truck_id'] = truck_id
    truck = Truck.objects.get(id=truck_id)
    if truck.driver_id != driver.user_id:
        return redirect('driver_trucks')

    request.session['truck_model'] = truck.truck_model
    request.session['license_plate'] = truck.license_plate
    request.session['driver'] = truck.driver.username() if truck.driver else None
    request.session['capacity'] = truck.capacity
    request.session['location'] = truck.location
    request.session['status'] = truck.status
```

```
    request.session['last_maintained'] = truck.last_maintained.isoformat() if
truck.last_maintained else None
    request.session['front_view'] = truck.front_view.url if truck.front_view else None
    request.session['side_view'] = truck.side_view.url if truck.side_view else None
    request.session['back_view'] = truck.back_view.url if truck.back_view else None
    request.session['top_view'] = truck.top_view.url if truck.top_view else None
    request.session['overall_view'] = truck.overall_view.url if truck.overall_view else
None
    request.session['truck_accepted'] = truck.truck_accepted
    request.session['truck_available'] = truck.truck_available

    return render(request, 'pages/trucks/driver/edit_truck.html' )
```

login_required requires the user (which is driver) to log in first in order to be able to execute the function above. Then the user_passes_test, is a function that tests if the user is a driver or not, if it is not a driver, then the user can't proceed to the function. try: driver = Driver.objects.get(user=request.user): This attempts to get the Driver object associated with the currently logged-in user.

except ObjectDoesNotExist: return redirect('go_home'): If the Driver object does not exist, the user is redirected to the 'go_home' page.

request.session['editTruckId'] = truck_id: This stores the truck_id in the user's session.

truck = Truck.objects.get(id=truck_id): This gets the Truck object with the given ID.

if truck.driver_id != driver.user_id: return redirect('driver_trucks'): If the Truck object is not associated with the current driver, the user is redirected to the 'driver_trucks' page.

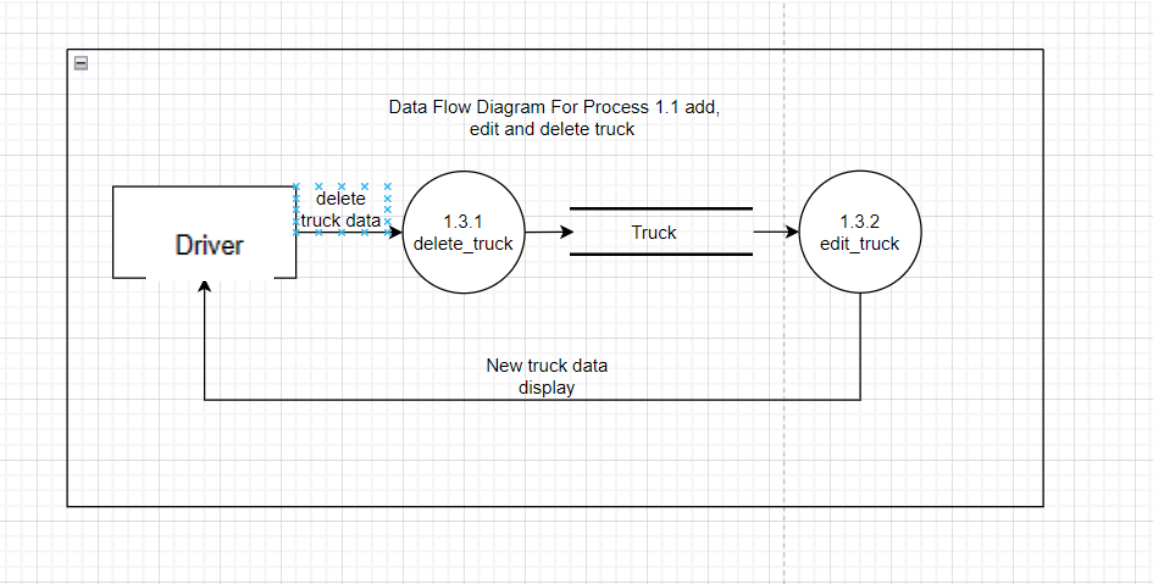
if request.method == 'POST': This checks if the HTTP method of the request is POST.

form = TruckForm(request.POST, request.FILES): This creates a new instance of TruckForm, populated with the POST data and files from the request.

if form.is_valid(): This checks if the form data is valid. If it is, the form data is saved to the Truck object (without committing the save to the database), the truck's fields are updated with the new data, and the Truck object is saved to the database.

return redirect('edit_truck'): After the Truck object is saved, the user is redirected to the 'edit_truck' page.

DFD Level 3 for delete car data



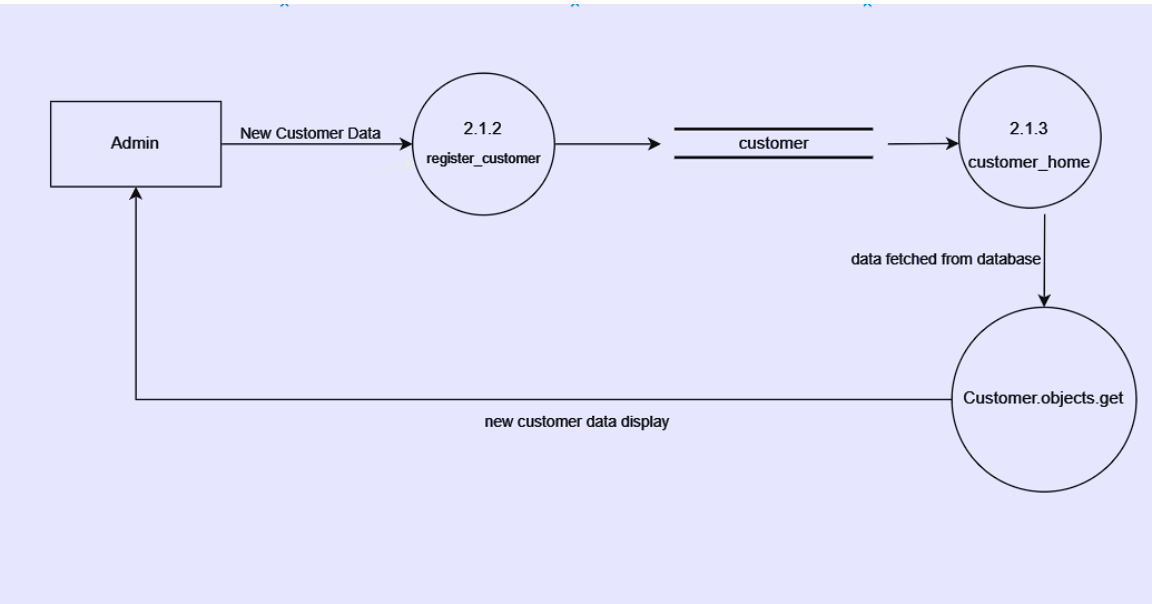
a. code:

```
@login_required(login_url = 'login')
@user_passes_test(im_driver, login_url='/')
@csrf_exempt
def delete_truck(request):
    if request.method == 'POST':
        truck_id = request.POST.get('truck_id')
        Truck.objects.filter(id=truck_id).delete()
        return JsonResponse({'status':'success'})
```

explanation:
login_required requires the user (which is driver) to log in first in order to be able to execute the function above. Then the user_passes_test, is a function that tests if the user is a driver or not, if it is not a driver, then the user can't proceed to the function. csrf_exempt means that the view can accept POST requests without the need for a CSRF token. The delete function checks if there are any request POST, if exists, it gets the truck_id gained from the template's javascript, and deletes the truck object.

User Login Functions (Views & URL):

1.4. DFD Level 3 for add customer data



code:

```
def register_customer(request):
    if request.user.is_authenticated:
        return redirect ('go_home')
    if request.method == 'POST':
        firstname = request.POST['firstname']
        lastname = request.POST['lastname']
```

```
        username = request.POST['username']
        phone_number = request.POST['phone_number']
        email = request.POST['email']
        # address = request.POST['address']
        password = request.POST['password']
        confirm_password = request.POST['confirm_password']

        request.session['firstname'] = firstname
        request.session['lastname'] = lastname
        request.session['username'] = username
        request.session['phone_number'] = phone_number
        request.session['email'] = email
        # request.session['address'] = address

        if password == confirm_password:
            if User.objects.filter(username=username).exists():
                messages.error(request, 'Username already exists!')
                return redirect('register_customer')
            else:
                if User.objects.filter(email=email).exists():
                    messages.error(request, 'Email already exists!')
                    return redirect('register_customer')
                else:
                    user = User.objects.create_user(first_name=firstname, last_name=lastname,
email=email,phone_number = phone_number , username=username,
password=password, is_customer = True)
                    user.save()
                    customer = Customer.objects.create(user=user)
                    customer.save()
                #   messages.success(request, 'You are registered successfully.')
                    auth.login(request, user)
                    # messages.success(request, 'You are now logged in.')
                    return redirect('customer_home')

        else:
            messages.error(request, 'Passwords do not match')
            return redirect('register_customer')

    return render(request, 'accounts/register_customer.html')
```

explanation:

if request.user.is_authenticated: return redirect ('go_home'): If the user is already authenticated (logged in), they are redirected to the 'go_home' page.

if request.method == 'POST': This checks if the HTTP method of the request is POST, which would indicate that the registration form has been submitted.

The next several lines retrieve the form data from the POST request and store it in the user's session. This includes the user's first name, last name, username, phone number, email, and password.

if password == confirm_password: This checks if the password and confirm password fields match. If they don't, an error message is displayed and the user is redirected back to the 'register_customer' page.

if User.objects.filter(username=username).exists(): This checks if a user with the given username already exists. If they do, an error message is displayed and the user is redirected back to the 'register_customer' page.

if User.objects.filter(email=email).exists(): This checks if a user with the given email already exists. If they do, an error message is displayed and the user is redirected back to the 'register_customer' page.

user = User.objects.create_user(...): If the username and email are unique and the passwords match, a new User object is created with the given data.

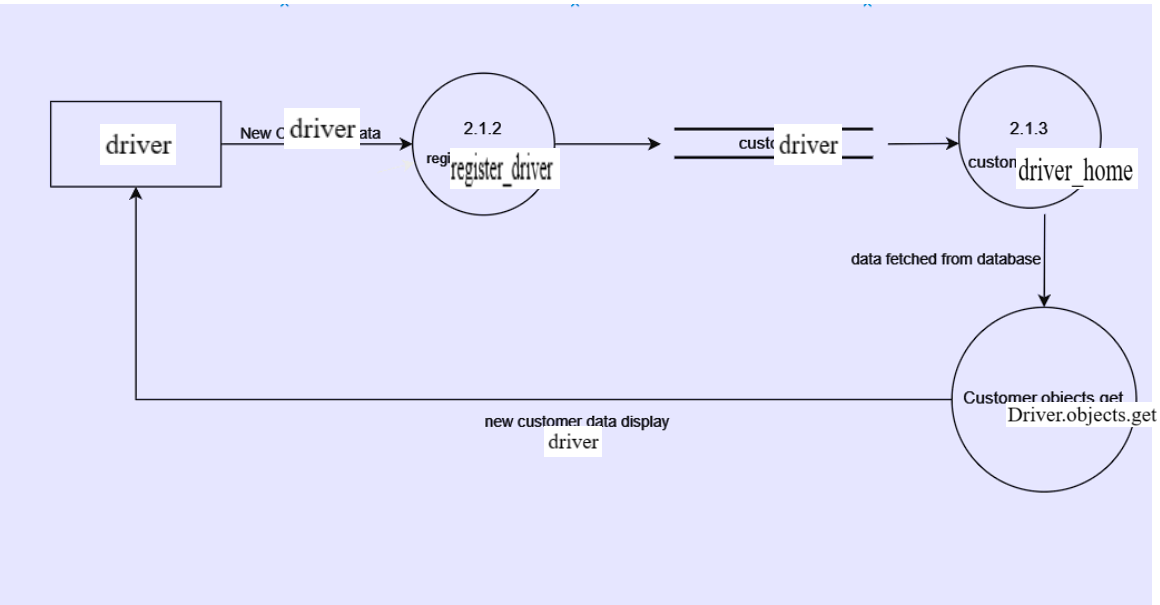
customer = Customer.objects.create(user=user): A new Customer object is created, associated with the new User object.

auth.login(request, user): The new user is logged in.

return redirect('customer_home'): The user is redirected to the 'customer_home' page.

return render(request, 'accounts/register_customer.html'): If the HTTP method of the request is not POST (i.e., it's a GET request), the 'accounts/register_customer.html' template is rendered. This would display the registration form to the user.

DFD Level 3 for add driver data



Code:

```
def register_driver(request):
    if request.user.is_authenticated:
        return redirect ('go_home')
    if request.method == 'POST':
        firstname = request.POST['firstname']
        lastname = request.POST['lastname']
        username = request.POST['username']
        phone_number = request.POST['phone_number']
        id_card = request.POST['id_card']
        license_card = request.POST['license_card']
        email = request.POST['email']
        password = request.POST['password']
        confirm_password = request.POST['confirm_password']

        request.session['firstname'] = firstname
        request.session['lastname'] = lastname
        request.session['username'] = username
        request.session['phone_number'] = phone_number
        request.session['email'] = email
```

```
        request.session['id_card'] = id_card
        request.session['license_card']=license_card

        if password == confirm_password:
            if User.objects.filter(username=username).exists():
                messages.error(request, 'Username already exists!')
                return redirect('register_driver')
            else:
                if User.objects.filter(email=email).exists():
                    messages.error(request, 'Email already exists!')
                    return redirect('register_driver')
                else:
                    user = User.objects.create_user(first_name=firstname, last_name=lastname,
email=email,phone_number = phone_number , username=username,
password=password, is_driver = True)
                    user.save()
                    driver = Driver.objects.create(user= user,id_card = id_card, license_card =
license_card)
                    driver.save()
                    # messages.success(request, 'You are registered successfully. Please wait for
confirmation from the admin.')
                    return render(request, 'pages/registration_thanks.html')
        else:
            messages.error(request, 'Password do not match')
            return redirect('register_driver')

    return render(request, 'accounts/register_driver.html')
```

Explanation:

The function above, register_driver(request), handles user registration requests. When receiving a request, it checks if the user is already authenticated; if they are, it then redirects to the home page. If the request method is POST, which means a form submission, it retrieves the user input data which are first name, last name, username, phone number, ID card details, license card details, email, password, and confirmation password. These inputs are also stored in the session for later use.

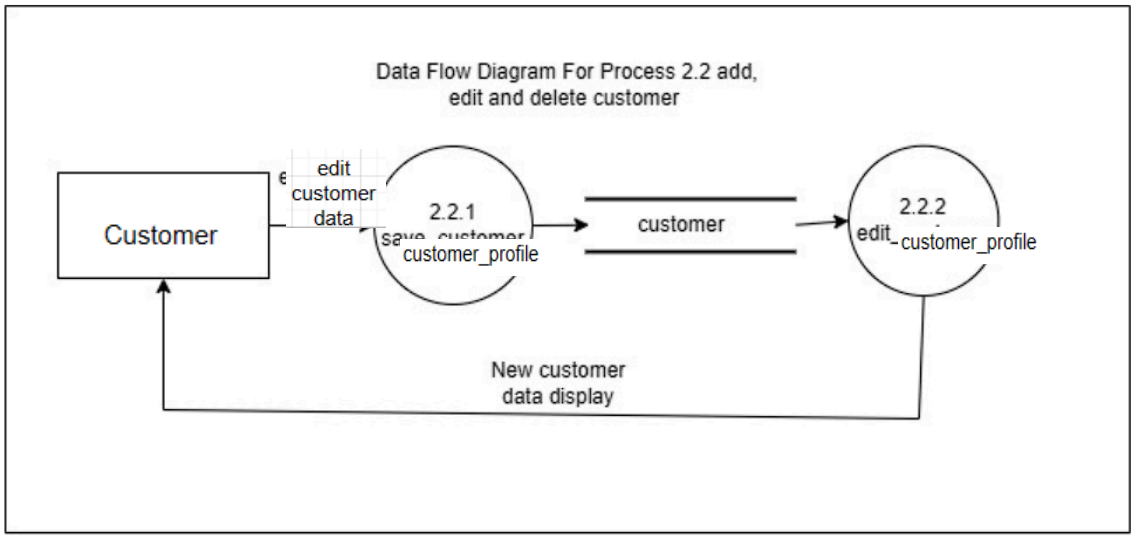
Then, it validates the input data, ensuring that the passwords match and that the provided username and email are not already in use by existing users. If the validation fails, the

error messages are displayed, and the user is redirected back to the registration page. If all validation passes, a User is created with the provided user details, marked as a driver, and saved to the database. Additionally, a Driver object associated with this user is created, containing specific driver-related information such as ID card and license card details.

Finally, upon a successful registration, the user is redirected to a "thank you" page which, acknowledges their registration.

1.5. User Management Functions (Views & URL):

DFD Level 3 for edit customer data



code:

```
@login_required(login_url = 'login')
@user_passes_test(im_customer, login_url='/')
def customer_profile(request):
    try:
        customer = Customer.objects.get(user=request.user)
    except ObjectDoesNotExist:
        # Handle the case where the customer does not exist
        return redirect('go_home')
```

```
    # Save the customer's data in the session
    request.session['firstname'] = customer.first_name()
    request.session['lastname'] = customer.last_name()
    request.session['username'] = customer.username()
    request.session['phone_number'] = customer.phone_number()
    request.session['email'] = customer.email()
    request.session['address'] = customer.address
    request.session['profile_picture'] = customer.profile_picture.url if
customer.profile_picture else None

    if request.method == 'POST':
        firstname = request.POST['firstname']
        lastname = request.POST['lastname']
        username = request.POST['username']
        phone_number = request.POST['phone_number']
        email = request.POST['email']
        address = request.POST['address']

        if
User.objects.filter(username=username).exclude(username=request.user.username).exists
():
            messages.error(request, 'Username already exists!')
            return redirect('customer_profile')
        else:
            if User.objects.filter(email=email).exclude(email=request.user.email).exists():
                messages.error(request, 'Email already exists!')
                return redirect('customer_profile')
            else:
                request.user.first_name = firstname
                request.user.last_name = lastname
                request.user.email = email
                request.user.phone_number = phone_number
                request.user.username = username
                request.user.save()

                customer.address = address
                # Assuming the profile picture is being sent in the 'profile_picture' field
                if 'profile_picture' in request.FILES:
```

```
            customer.profile_picture = request.FILES['profile_picture']
        customer.save()

        messages.success(request, 'Profile updated successfully.')
        return redirect('customer_profile')

    return render(request, 'pages/profile/customer_profile.html')
```

Explanation:
login_required requires the user (which is customer) to log in first in order to be able to execute the function above. Then the user_passes_test, is a function that tests if the user is a customer or not, if it is not a driver, then the user can't proceed to the function.

try: customer = Customer.objects.get(user=request.user): This attempts to get the Customer object associated with the currently logged-in user.

except ObjectDoesNotExist: return redirect('go_home'): If the Customer object does not exist, the user is redirected to the 'go_home' page.

The next several lines store the customer's current profile information in the session.

if request.method == 'POST': This checks if the HTTP method of the request is POST, which would indicate that the profile update form has been submitted.

The next several lines retrieve the updated profile information from the POST request.

if User.objects.filter(username=username).exclude(username=request.user.username).exists(): This checks if the updated username is already taken by another user. If it is, an error message is displayed and the user is redirected back to the 'customer_profile' page.

if User.objects.filter(email=email).exclude(email=request.user.email).exists(): This checks if the updated email is already taken by another user. If it is, an error message is displayed and the user is redirected back to the 'customer_profile' page.

If the updated username and email are unique, the user's profile information is updated and saved.

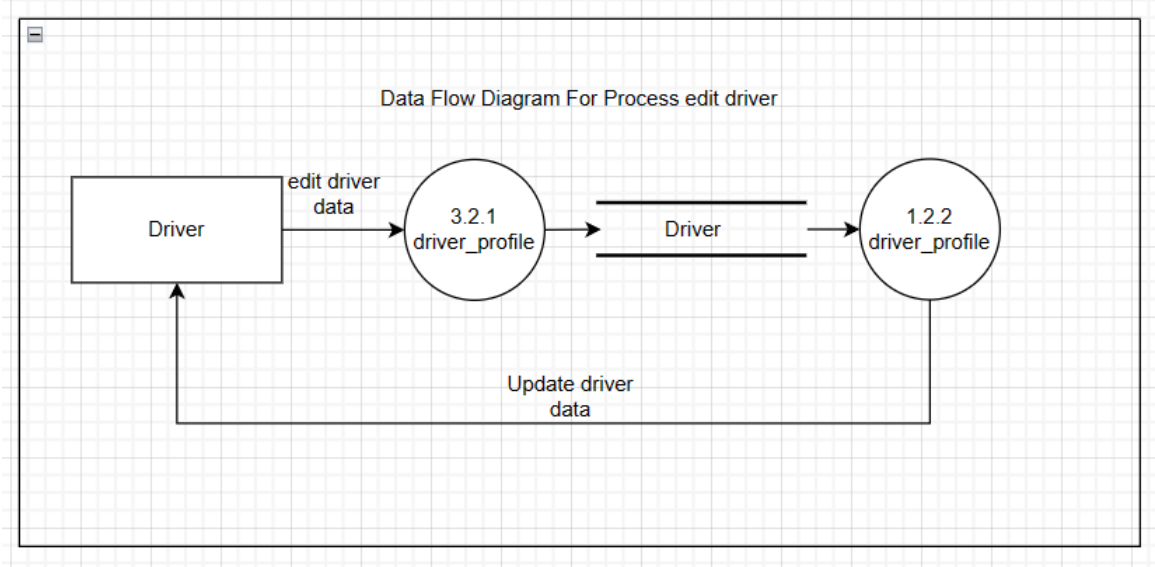
if 'profile_picture' in request.FILES: This checks if a new profile picture has been uploaded. If it has, the profile picture is updated.

messages.success(request, 'Profile updated successfully.'): A success message is displayed.

return redirect('customer_profile'): The user is redirected back to the 'customer_profile' page.

return render(request, 'pages/profile/customer_profile.html'): If the HTTP method of the request is not POST (i.e., it's a GET request), the 'pages/profile/customer_profile.html' template is rendered. This would display the current profile information to the user.

DFD Level 3 for edit driver data:



code:

```
@login_required(login_url = 'login')
@user_passes_test(im_driver, login_url='/')
def driver_profile(request):
    try:
        driver = Driver.objects.get(user=request.user)
```

```
        if driver.accepted != 1:
            return render(request, 'pages/registration_thanks.html')
    except ObjectDoesNotExist:
        # Handle the case where the driver does not exist
        return redirect('go_home')

    # Save the driver's data in the session
    request.session['firstname'] = driver.first_name()
    request.session['lastname'] = driver.last_name()
    request.session['username'] = driver.username()
    request.session['phone_number'] = driver.phone_number()
    request.session['email'] = driver.email()
    request.session['license_number'] = driver.license_number
    request.session['availability'] = driver.availability
    request.session['profile_picture'] = driver.profile_picture.url if
driver.profile_picture else None
    request.session['id_card'] = driver.id_card.url if driver.id_card else None
    request.session['license_card'] = driver.license_card.url if driver.license_card
else None
    request.session['profile_picture_confirmed'] = driver.profile_picture_confirmed
    request.session['accepted'] = driver.accepted
    request.session['vehicle_available'] = driver.vehicle_available

    if request.method == 'POST':
        firstname = request.POST['firstname']
        lastname = request.POST['lastname']
        username = request.POST['username']
        phone_number = request.POST['phone_number']
        email = request.POST['email']

        if
User.objects.filter(email=email).exclude(email=request.user.email).exists():
            messages.error(request, 'Email already exists!')
            return redirect('driver_profile')
        else:
            request.user.first_name = firstname
            request.user.last_name = lastname
            request.user.email = email
            request.user.phone_number = phone_number
```

```
            request.user.username = username
            request.user.save()

            driver.license_number = request.POST['license_number']
            driver.availability = request.POST['availability']

            # Assuming the profile picture, id card, and license card are being sent in
the respective fields
            if 'profile_picture' in request.FILES:
                driver.profile_picture = request.FILES['profile_picture']
            if 'id_card' in request.FILES:
                driver.id_card = request.FILES['id_card']
            if 'license_card' in request.FILES:
                driver.license_card = request.FILES['license_card']
            driver.save()

            messages.success(request, 'Profile updated successfully.')
            return redirect('driver_profile')

    return render(request, 'pages/profile/driver_profile.html')
```

Explanation:

try: driver = Driver.objects.get(user=request.user): This attempts to get the Driver object associated with the currently logged-in user.

if driver.accepted != 1: return render(request, 'pages/registration_thanks.html'): If the driver's application has not been accepted, they are shown a "thank you for registering" page.

except ObjectDoesNotExist: return redirect('go_home'): If the Driver object does not exist, the user is redirected to the 'go_home' page.

The next several lines store the driver's current profile information in the session.

if request.method == 'POST': This checks if the HTTP method of the request is POST, which would indicate that the profile update form has been submitted.

The next several lines retrieve the updated profile information from the POST request.

if User.objects.filter(email=email).exclude(email=request.user.email).exists(): This checks if the updated email is already taken by another user. If it is, an error message is displayed and the user is redirected back to the 'driver_profile' page.

If the updated email is unique, the user's profile information is updated and saved.

The driver's license number and availability are updated from the POST data.

if 'profile_picture' in request.FILES, if 'id_card' in request.FILES, and if 'license_card' in request.FILES: These check if new files have been uploaded for the profile picture, ID card, and license card. If they have, the respective fields are updated.

messages.success(request, 'Profile updated successfully.'): A success message is displayed.

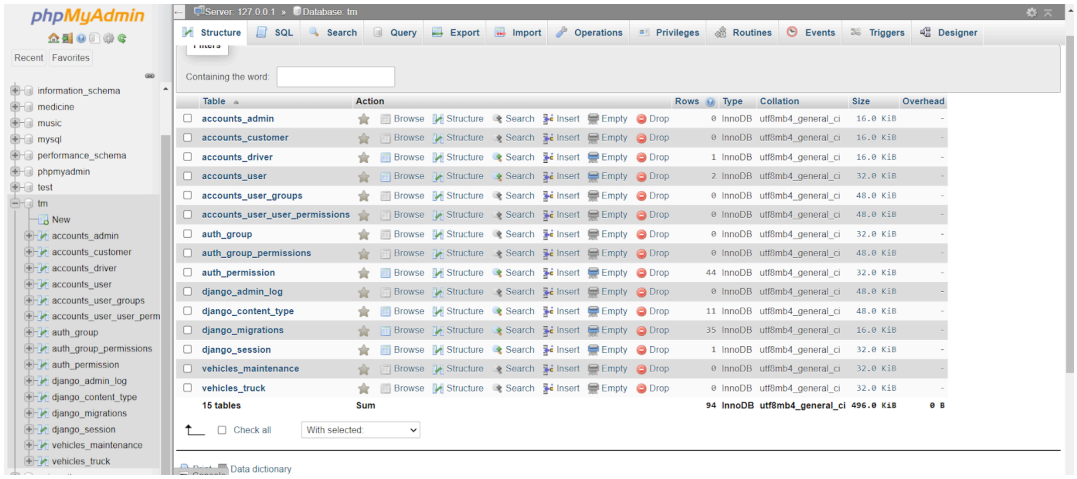
return redirect('driver_profile'): The user is redirected back to the 'driver_profile' page.

return render(request, 'pages/profile/driver_profile.html'): If the HTTP method of the request is not POST (i.e., it's a GET request), the 'pages/profile/driver_profile.html' template is rendered. This would display the current profile information to the user.

Truck Management Functions (Views & URL):

Admin Functions (URL):

IV.1.2 Database implementation



User:

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**id**	bigint			No	*None*		AUTO_INCREMENT	Change Drop More
☐	2	**password**	varchar(128)	utf8mb4_0900_ai_ci		No	*None*			Change Drop More
☐	3	**last_login**	datetime(6)			Yes	*NULL*			Change Drop More
☐	4	**is_superuser**	tinyint(1)			No	*None*			Change Drop More
☐	5	**username**	varchar(150)	utf8mb4_0900_ai_ci		No	*None*			Change Drop More
☐	6	**first_name**	varchar(150)	utf8mb4_0900_ai_ci		No	*None*			Change Drop More
☐	7	**last_name**	varchar(150)	utf8mb4_0900_ai_ci		No	*None*			Change Drop More
☐	8	**email**	varchar(254)	utf8mb4_0900_ai_ci		No	*None*			Change Drop More
☐	9	**is_staff**	tinyint(1)			No	*None*			Change Drop More
☐	10	**is_active**	tinyint(1)			No	*None*			Change Drop More
☐	11	**date_joined**	datetime(6)			No	*None*			Change Drop More
☐	12	**is_customer**	tinyint(1)			No	*None*			Change Drop More
☐	13	**is_driver**	tinyint(1)			No	*None*			Change Drop More
☐	14	**phone_number**	varchar(20)	utf8mb4_0900_ai_ci		No	*None*			Change Drop More

Customer:

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**user_id**	bigint			No	*None*			Change Drop More
☐	2	**address**	longtext	utf8mb4_0900_ai_ci		No	*None*			Change Drop More
☐	3	**profile_picture**	varchar(100)	utf8mb4_0900_ai_ci		Yes	*NULL*			Change Drop More

Driver:

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**user_id** 🔑	bigint			No	*None*			Change Drop More
☐	2	**license_number**	varchar(20)	utf8mb4_0900_ai_ci		No	*None*			Change Drop More
☐	3	**availability**	tinyint(1)			No	*None*			Change Drop More
☐	4	**profile_picture**	varchar(100)	utf8mb4_0900_ai_ci		Yes	*NULL*			Change Drop More
☐	5	**id_card**	varchar(100)	utf8mb4_0900_ai_ci		Yes	*NULL*			Change Drop More
☐	6	**license_card**	varchar(100)	utf8mb4_0900_ai_ci		Yes	*NULL*			Change Drop More
☐	7	**profile_picture_confirmed**	tinyint(1)			No	*None*			Change Drop More
☐	8	**accepted**	tinyint(1)			No	*None*			Change Drop More
☐	9	**vehicle_available**	tinyint(1)			No	*None*			Change Drop More

Truck:

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**id**	bigint			No	*None*		AUTO_INCREMENT	Change Drop More
☐	2	**truck_model**	varchar(255)	utf8mb4_0900_ai_ci		No	*None*			Change Drop More
☐	3	**license_plate**	varchar(20)	utf8mb4_0900_ai_ci		No	*None*			Change Drop More
☐	4	**capacity**	int			Yes	*NULL*			Change Drop More
☐	5	**location**	varchar(255)	utf8mb4_0900_ai_ci		Yes	*NULL*			Change Drop More
☐	6	**last_maintained**	date			Yes	*NULL*			Change Drop More
☐	7	**status**	varchar(10)	utf8mb4_0900_ai_ci		No	*None*			Change Drop More
☐	8	**back_view**	varchar(100)	utf8mb4_0900_ai_ci		Yes	*NULL*			Change Drop More
☐	9	**front_view**	varchar(100)	utf8mb4_0900_ai_ci		Yes	*NULL*			Change Drop More
☐	10	**side_view**	varchar(100)	utf8mb4_0900_ai_ci		Yes	*NULL*			Change Drop More
☐	11	**top_view**	varchar(100)	utf8mb4_0900_ai_ci		Yes	*NULL*			Change Drop More
☐	12	**overall_view**	varchar(100)	utf8mb4_0900_ai_ci		Yes	*NULL*			Change Drop More
☐	13	**truck_accepted**	tinyint(1)			No	*None*			Change Drop More
☐	14	**truck_available**	tinyint(1)			No	*None*			Change Drop More
☐	15	**driver_id**	bigint			Yes	*NULL*			Change Drop More

1. Hardware implementation (Hardwares are not needed in my module, since I only make the website that serves to connect all of the modules all the other members, as well as managing the database).

IV.1.3 Implementation verification of Parts 1, 2, 3, and 4 explained above.

No.	Scenario	Every Possible Input	Expected Output	Output Result

1.	User logins	Login Page Display: When users access the login page, they should see a form with fields for username and password. Form Submission: Users should be able to submit the login form with their username and password. Form Validation: If users try to submit the form without filling out all the required fields, they might see error messages indicating which fields need to be filled. Authentication: After submitting the form, the system should authenticate the user based on the provided	A login page displaying a form with fields for username and password. Form validation to ensure both fields are filled out before submission. Error messages displayed if there are validation errors or authentication failures. Successful login redirects the user to the appropriate page.	The actual output will depend on how users interact with the login page and the correctness of the credentials they provide. If users fill out the form correctly and their credentials are valid, they should be redirected to the appropriate page. If there are errors in the form submission or authentication process, users will see relevant error messages indicating the issue.

		username and password. Error Messages: If there are authentication errors, such as incorrect username or password, users should see appropriate error messages. Successful Login: If authentication is successful, users should be redirected to the appropriate page, such as a dashboard or homepage.		If the login process encounters technical issues, such as server errors, users might see generic error messages or be unable to log in.
2	Register customer	The web page displaying the customer registration form. Validation errors if any fields are not filled correctly or do not meet the criteria.	If all fields are filled correctly and the form is submitted, the user will receive a success message confirming the	User is able to register as customer

		Success message if the registration is successful. Error message if there is an internal issue, such as a server problem.	successful registration. If there are validation errors, such as incorrectly filled fields, the user will see an error message explaining the issue. If there is an internal error, such as a server problem, the user may see an error message stating that the registration cannot be processed at the moment.	
3	Register driver	Open Driver Registration Page: When users access the driver registration	A driver registration page displaying a form	User is able to register as driver.

		page, they should see the registration form with various fields to fill out. Form Validation: If users try to submit the form without filling out all the required fields, they might see error messages informing them about which fields need to be filled. Error Messages: If there are validation errors or other issues when users submit the form, they might see error messages related to things like form input errors or server issues. Successful Registration: If users successfully submit	with required fields. Form validation to ensure all required fields are filled out correctly. Relevant error messages if there are issues with form submission. Confirmation of successful registration after the form is submitted correctly.	

		the form with valid data, they should receive confirmation that the driver registration was successful.		
4.	Edit Profile Customer	Accessing the customer profile page as an authenticated user. Attempting to access the customer profile page without being logged in (redirects to the login page). Accessing the customer profile page as a user who does not meet the criteria to be a customer (redirects to the home page). Submitting a POST request to update the customer profile with	If the user is not logged in, they are redirected to the login page. If the user is not a customer (as per the im_customer function), they are redirected to the home page. If the customer profile exists, it is rendered with the customer's information pre-populated. If the customer profile does not	Rendering of the customer profile page with pre-populated information. Redirecting to the login page if not authenticated. Redirecting to the home page if not meeting customer criteria or if the profile does not exist. Displaying success or error messages upon

		various combinations of valid and invalid data.	exist, the user is redirected to the home page. If the submitted data in the POST request is valid, the customer's profile information is updated, and a success message is displayed. If the submitted data contains a username or email that already exists in the database for a different user, an error message is displayed, and the user is redirected back to the profile page.	updating the profile.
5.	Edit Profile Driver	Accessing the driver profile page as an authenticated user.	If the user is not logged in, they are	Rendering of the driver profile page with

		Attempting to access the driver profile page without being logged in (redirects to the login page). Accessing the driver profile page as a user who does not meet the criteria to be a driver (redirects to the home page). Submitting a POST request to update the driver profile with various combinations of valid and invalid data.	redirected to the login page. If the user is not a driver (as per the im_driver function), they are redirected to the home page. If the driver's accepted field is not equal to 1, they are redirected to a "registration thanks" page. If the driver profile exists, it is rendered with the driver's information pre-populated. If the driver profile does not exist, the user is redirected to the home page.	pre-populated information. Redirecting to the login page if not authenticated. Redirecting to the home page if not meeting driver criteria or if the profile does not exist. Redirecting to the "registration thanks" page if the driver is not accepted. Displaying success or error messages upon updating the profile.

			If the submitted data in the POST request is valid, the driver's profile information is updated, and a success message is displayed. If the submitted data contains an email that already exists in the database for a different user, an error message is displayed, and the user is redirected back to the profile page.	
6.	Add Truck	Accessing the truck addition page as an authenticated user. Attempting to access the truck addition page without being	If the user is not logged in, they are redirected to the login page. If the user is not a driver (as per the	Rendering of the truck addition form page with the form displayed.

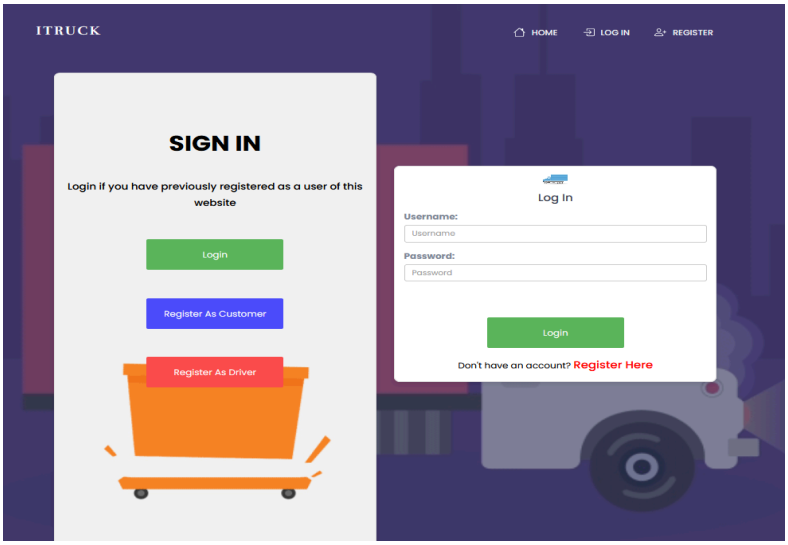
		logged in (redirects to the login page). Accessing the truck addition page as a user who does not meet the criteria to be a driver (redirects to the home page). Submitting a POST request to add a new truck with various combinations of valid and invalid data.	im_driver function), they are redirected to the home page. If the submitted form data is valid, a new truck is added to the database, and a JSON response containing information about the newly added truck is returned. If the submitted form data is invalid, an error JSON response containing form validation errors is returned. If the request method is not POST, the form	Redirecting to the login page if not authenticated. Redirecting to the home page if not meeting driver criteria. Adding a new truck to the database and returning a success JSON response if the form data is valid. Returning an error JSON response containing validation errors if the form data is invalid.

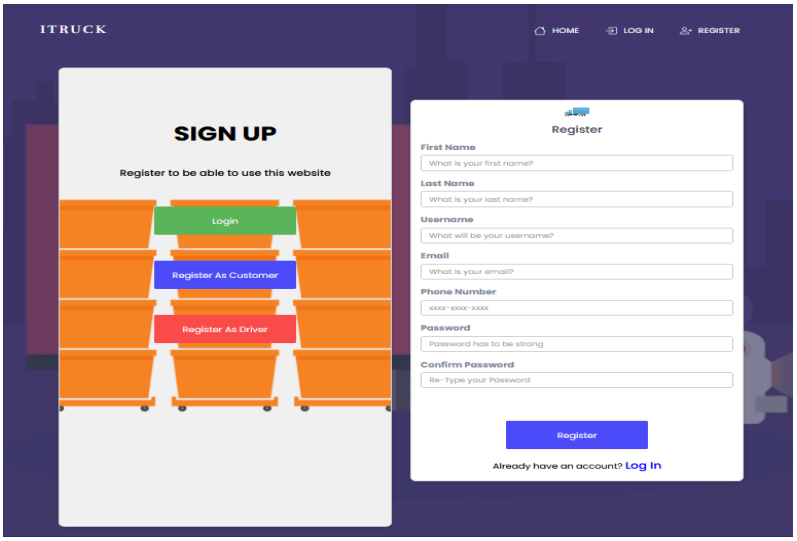
			for adding a truck is rendered.	
7	Edit Truck	Accessing the truck editing page as an authenticated user. Attempting to access the truck editing page without being logged in (redirects to the login page). Accessing the truck editing page as a user who does not meet the criteria to be a driver (redirects to the home page). Accessing the editing page for a truck that does not belong to the authenticated driver. Successfully accessing the editing page for a truck owned by the authenticated driver.	If the user is not logged in, they are redirected to the login page. If the user is not a driver (as per the im_driver function), they are redirected to the home page. If the truck being edited does not belong to the authenticated driver, the user is redirected to the page displaying the driver's trucks. If the truck being edited belongs to the authenticated driver, the editing page is rendered	Rendering of the truck editing page with the truck's details pre-populated if the truck belongs to the authenticated driver. Redirecting to the login page if not authenticated. Redirecting to the home page if not meeting driver criteria or if the truck does not belong to the authenticated driver.

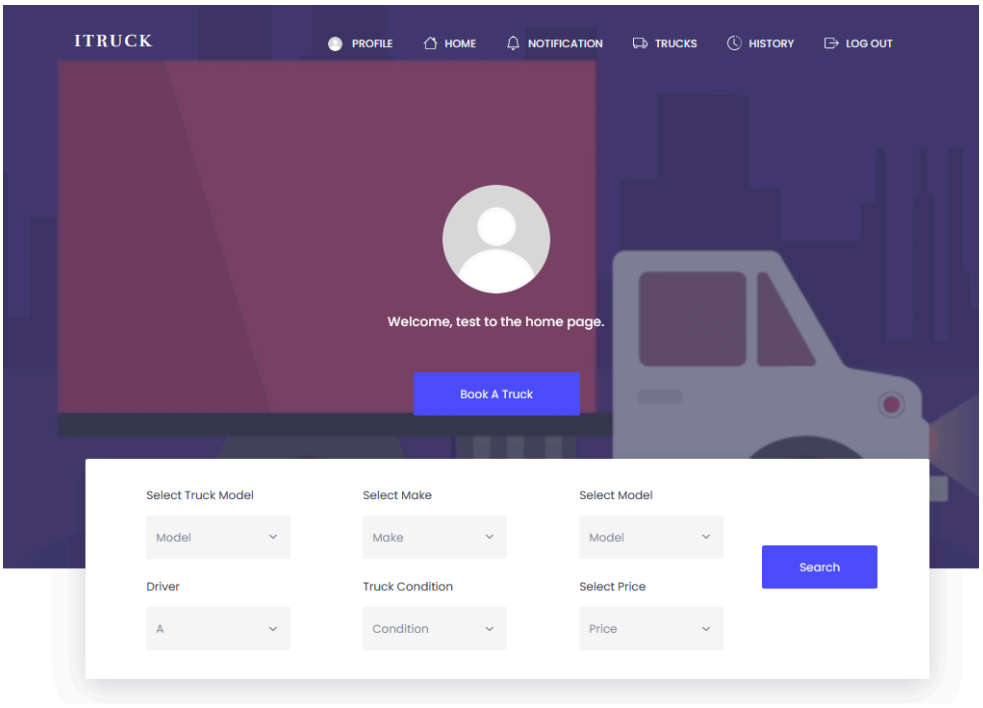
			with the truck's details pre-populated in the session. The editing page displays the details of the truck fetched from the database.	

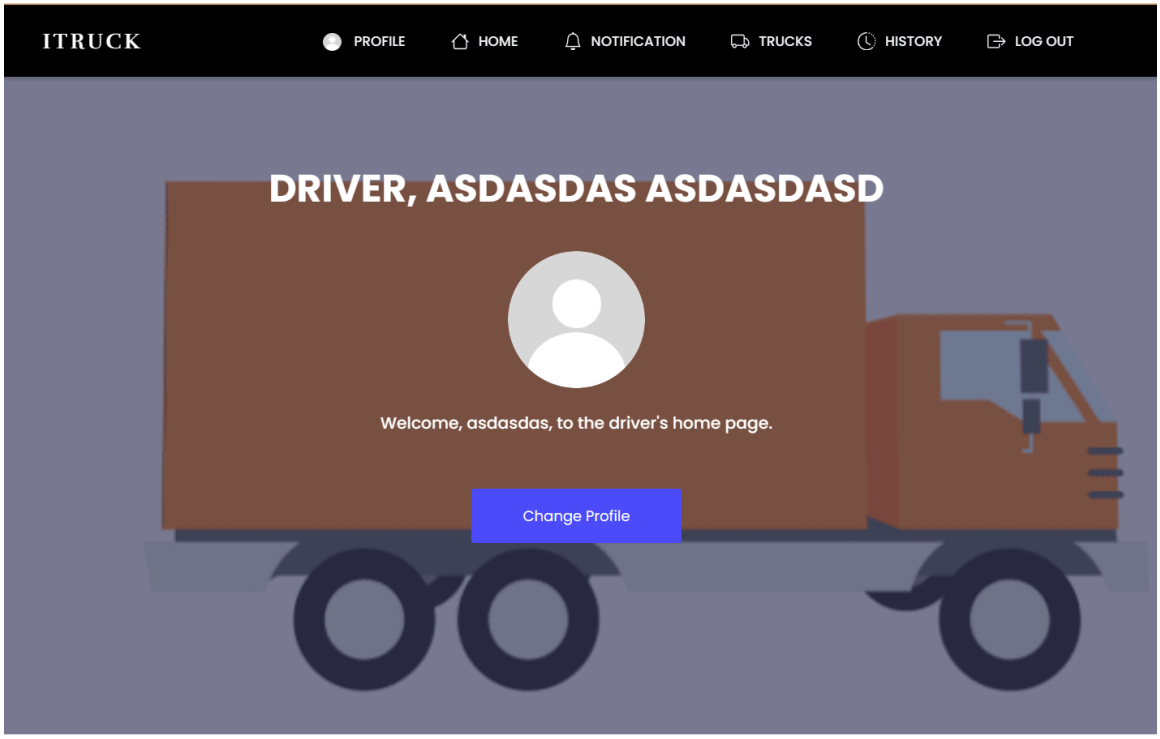
IV.2 Product Display The Reference Material

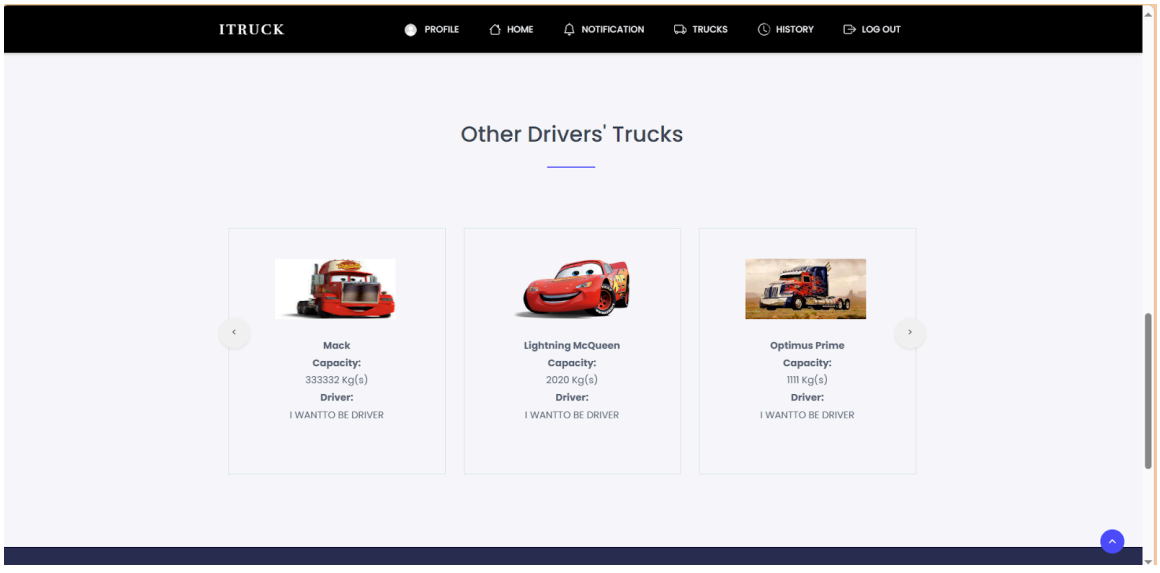
IV.2.1 SOFTWARE PRODUCT DISPLAY






ITRUCK
PROFILE
HOME
NOTIFICATION
TRUCKS
HISTORY
LOG OUT
Welcome, test to the home page.
Book A Truck
Select Truck Model
Model
Select Make
Make
Select Model
Model
Search
Driver
A
Truck Condition
Condition
Select Price
Price


ITRUCK
PROFILE
HOME
NOTIFICATION
TRUCKS
HISTORY
LOG OUT
DRIVER, ASDASDAS ASDASDASD
Welcome, asdasdas, to the driver's home page.
Change Profile


ITRUCK
PROFILE
HOME
NOTIFICATION
TRUCKS
HISTORY
LOG OUT
Other Drivers' Trucks
Mack
Capacity:
333332 Kg(s)
Driver:
I WANTTO BE DRIVER
Lightning McQueen
Capacity:
2020 Kg(s)
Driver:
I WANTTO BE DRIVER
Optimus Prime
Capacity:
1111 Kg(s)
Driver:
I WANTTO BE DRIVER

IV.2.2 HARDWARE PRODUCT DISPLAY

Hardwares are not needed in my module, since I only make the website that serves to connect all of the modules all the other members, as well as managing the database

IV.3 Manual Guide

IV.3.1 SYSTEM BUILD DOCUMENTATION FROM THE SOURCE vii.

1. Install Python and pip.
2. Download the project from **https://github.com/GeraldSutarso/Trucking-Management**
3. Assuming Python and pip are already installed, open the cmd of the downloaded project's folder
4. type "py -m venv env" to create virtual environment
5. Activate it by navigating to the >env/Scripts and opening the activate.bat, can be done by continuing the command prompt from before.
   >cd env/Scripts
   >activate.bat
   You can also do this instead: .\env\Scripts\activate
6. After that, go back to the project folder and install the required dependencies, can be done by continuing the command prompt from before.
   >cd..
   >cd..
   >pip install -r requirements.txt

7. To setting the database, create a MySql Database named tm (you may change the setting in tm\settings.py), then run this inside the “tm “ folder.
   >py manage.py migrate
8. To run the project, simply go to the tm folder

```
|___ Trucking-Management
     |___ .git
     |___ env
     |___ tm  <--- This folder
          |___ tm
               |___ __pycache__
               |___ static
               |___ __init__.py
               |___ asgi.py
               |___ settings.py
               |___ urls.py
               |___ wsgi.py
          |___ "Application folders"
          |___ db.sqlite3
          |___ manage.py
     |___ .gitignore
     |___ readme.md
     |___ requirements
```

9. open cmd at that (tm) folder, then:
   >py manage.py runserver
10. The web app could be opened from localhost in a browser. It is using Django version 5.x so further details regarding development using Django could be found in **https://docs.djangoproject.com/en/5.0/**

■

IV.3.2 SYSTEM INSTALLATION

■

Admin:

1. Admin will need to have the required components listed in Chapter C above.
2. After deploying the server(steps could be done from the System Build), the admin could create a new superuser by using the command “py manage.py createsuperuser”.
3. Browser for website is needed (Newest version is recommended)
4. No further installation required for Administrator, access to the website is free from the web browser

-Driver:

1. Browser for website is needed (Newest version is recommended)
2. No further installation required for Driver, access to the website is free from the web browser

-Customer:

1. Browser for website is needed (Newest version is recommended)
2. No further installation required for Customer, access to the website is free from the web browser.

IV.3.3 USER GUIDE PER USER ROLE

IV.3.3.1 ADMIN

■

User Guide: Admin

1. Registration and Login Process:
- Use the provided admin credentials to log in to the administration page.

2. Viewing Customer Profile Data:
- Navigate to the customer profile section to view customer data.

3. Checking Driver Data and Requests:
- Access the driver management section to review driver requests and profiles.

4. Interviewing Requestor:
- Conduct interviews with driver requesters and verify their suitability.

5. Document Verification:
- Ensure that drivers have submitted all required documents for approval.

6. Verifying Ongoing Bookings and Payments:
- Check ongoing bookings and verify payment status.

7. Creating Chat Channels:

- Use the provided chat application to create channels for customer-driver communication.

8. Monitoring Driver Checkups and Deliveries:
- Keep track of driver checkups before delivery and monitor delivery progress.

9. Monitoring Vehicle Maintenance:
- Receive and act upon vehicle maintenance notifications promptly.

10. Coordinating Maintenance Schedules:
- Communicate with drivers to schedule vehicle maintenance as necessary.

IV.3.3.2 Driver

■

User Guide: Driver

1. Registration and Interview Process:
- Register as a driver and await notification for the interview process.

2. Accessing Driver Dashboard:
- Upon acceptance, log in to the driver dashboard to access various features.

3. Navigating Dashboard Sections:

- Utilize sections like delivery, booking list, delivery history, maintenance schedule, and view truck.

4. Checking and Accepting Bookings:
- Review booking requests and accept suitable jobs.

5. Communicating with Customers:
- Use the provided chat platform to discuss delivery details with customers.

6. Completing Checkups and Deliveries:
- Undergo necessary checkups and proceed with deliveries once verified.

7. Updating Progress:
- Keep both customers and admin informed of delivery progress with regular updates.

8. Handling Complaints:
- Address any customer complaints within the specified timeframe.

9. Responding to Maintenance Notifications:
- Act on maintenance notifications and coordinate with admin for scheduling.

IV.3.3.3 Customer

■

User Guide: Customer

1. Registration and Login:
- Register as a customer and log in with provided credentials.

2. Browsing Recommendations:
- Explore recommended trucks based on preferences or search for specific criteria.

3. Choosing a Truck:
- Select a suitable truck and view driver details.

4. Booking Process:
- Fill in the booking form and wait for driver acceptance.

5. Payment Verification:
- Await payment verification by the admin.

6. Communication with Driver:
- Engage in communication with the driver regarding delivery details.

7. Tracking Delivery Progress:
- Monitor delivery progress using the provided map and expect regular updates from the driver.

8. Confirming Delivery:
- Confirm delivery completion within the specified timeframe, providing feedback if necessary.

IV.4 Video Demonstration

All video of Demonstration can be accessed in this link below

Video Guide
https://drive.google.com/drive/folders/1wqq7WmPdjt4obpQCOP20HNMqPzWvX5-r?usp=sharing

CHAPTER V

TESTING

V.1 Functional testing

V.1.1 *Testing results of every function in the specification (based on PART C in F100, PART C NO.4 in F200, PART D in F200)*

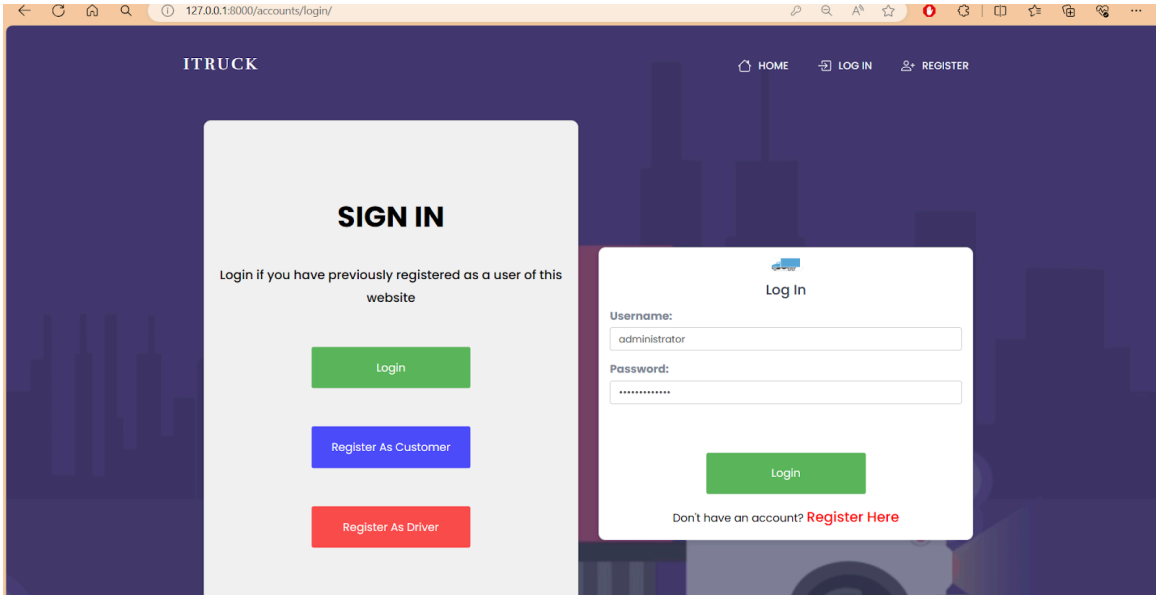
CHAPTER VI

Checking whether the user (Administrator, Driver, Customer) Management module could successfully manage the user objects.

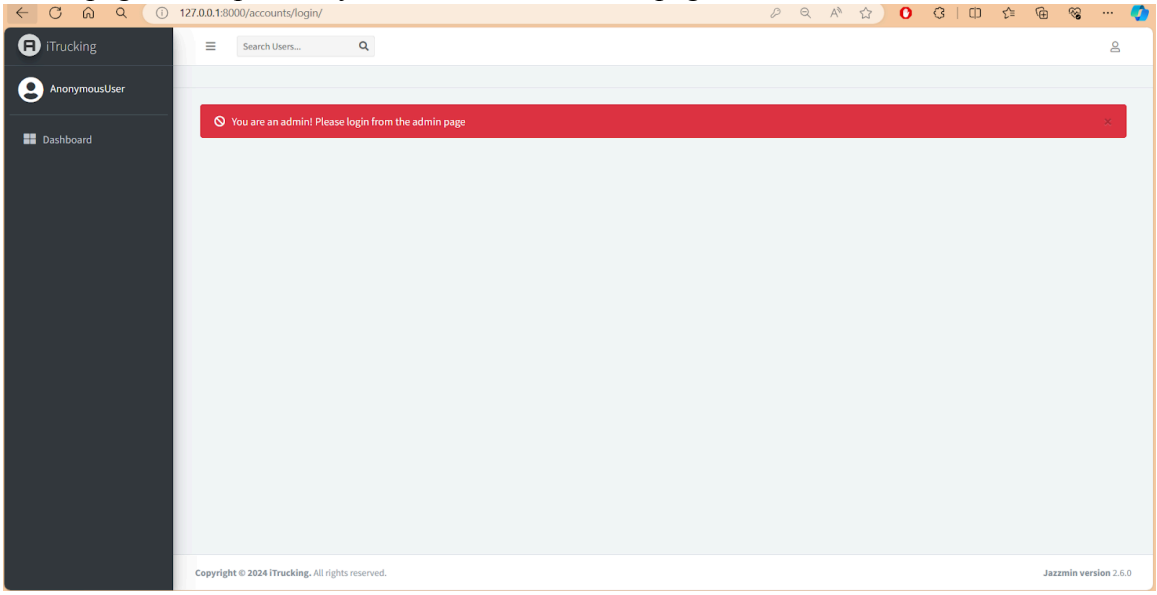
- Administrator should be able to manage the whole database by logging in as an administrator. The superuser (administrator is registered through the django administrator feature.

```
(env) C:\Kuliah\CAPSTONE\Project\Trucking-Management\tm>py manage.py createsuperuser

You have 1 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): booking
.
Run 'python manage.py migrate' to apply them.
Username: administrator
Email address: administrator@user.co.id
Password:
Password (again):
The password is too similar to the username.
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.

(env) C:\Kuliah\CAPSTONE\Project\Trucking-Management\tm>
```

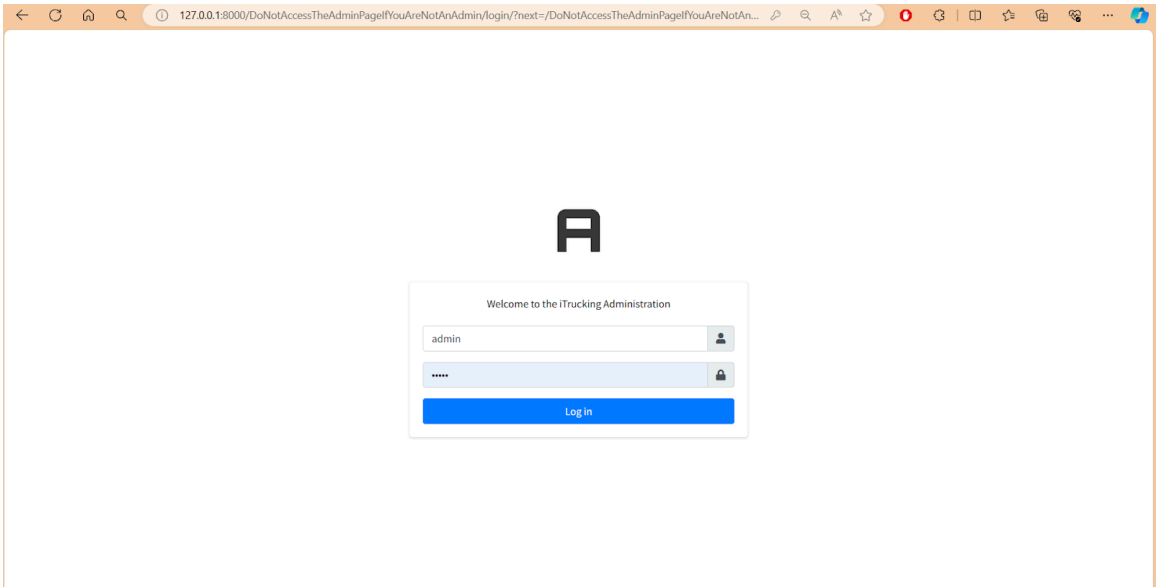
- After this, administrator could open the login page from the website and enter the correct username and password credentials.



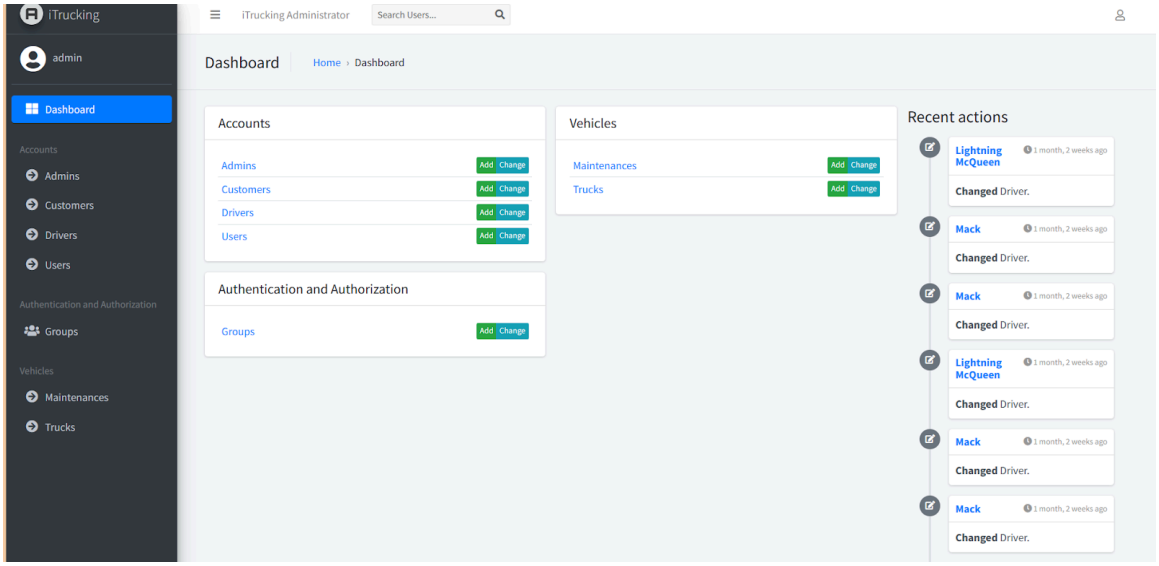
- Admin will be thrown into the admin page, and asked to re-login through the admin page, here, press any button available in the page.



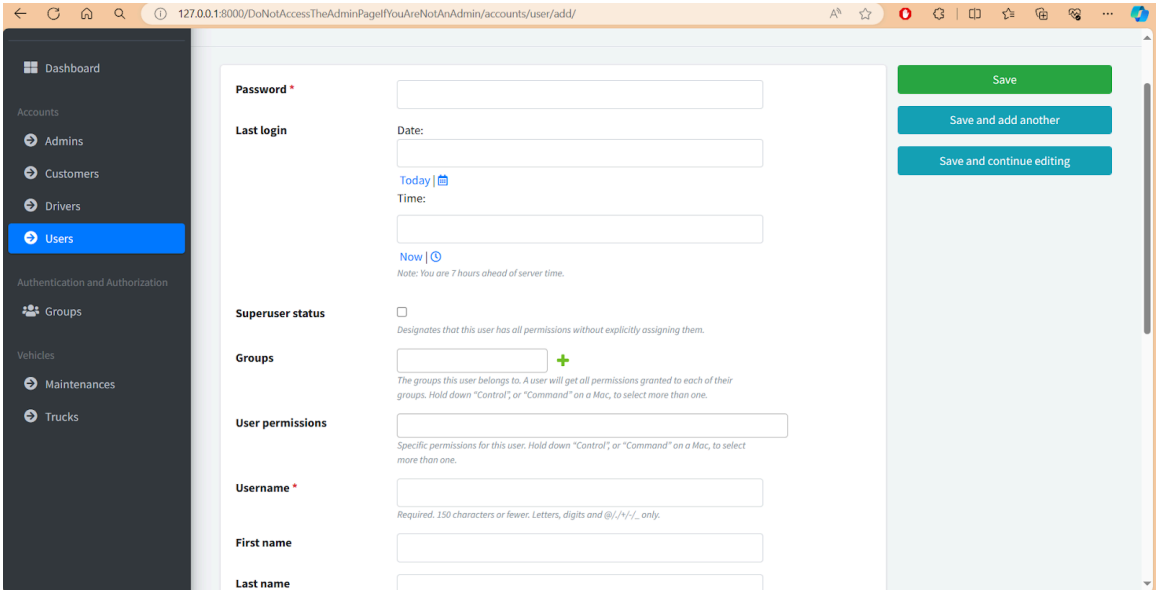
- Proceed on logging in again using the correct admin credentials.



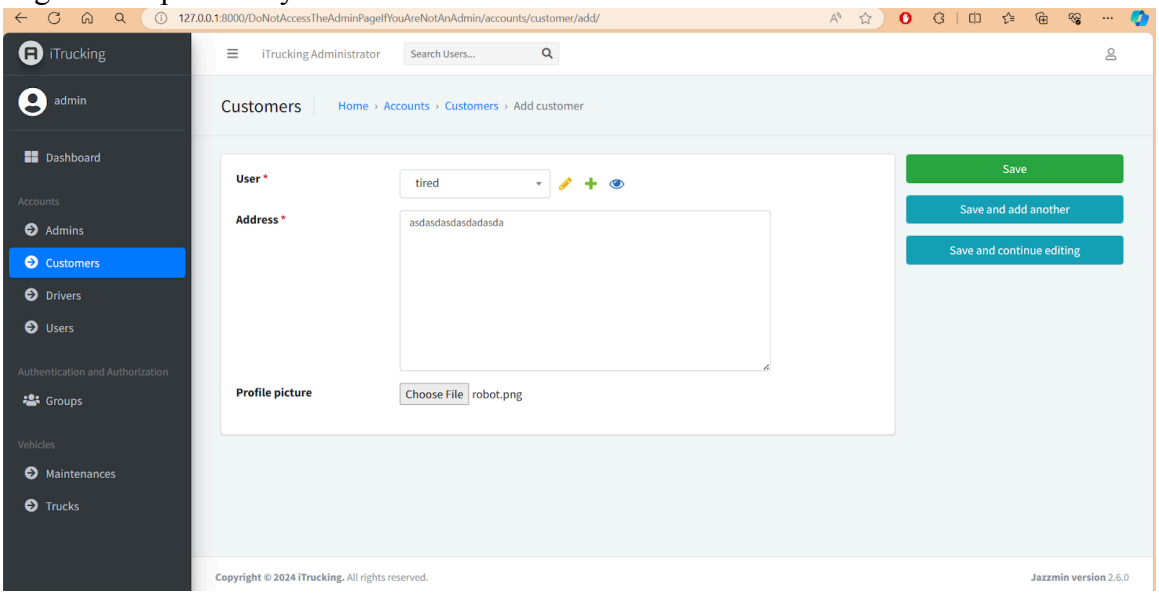
- Here, try to create a new customer user from the administrator page, see if the admin could manage the user through the admin page.



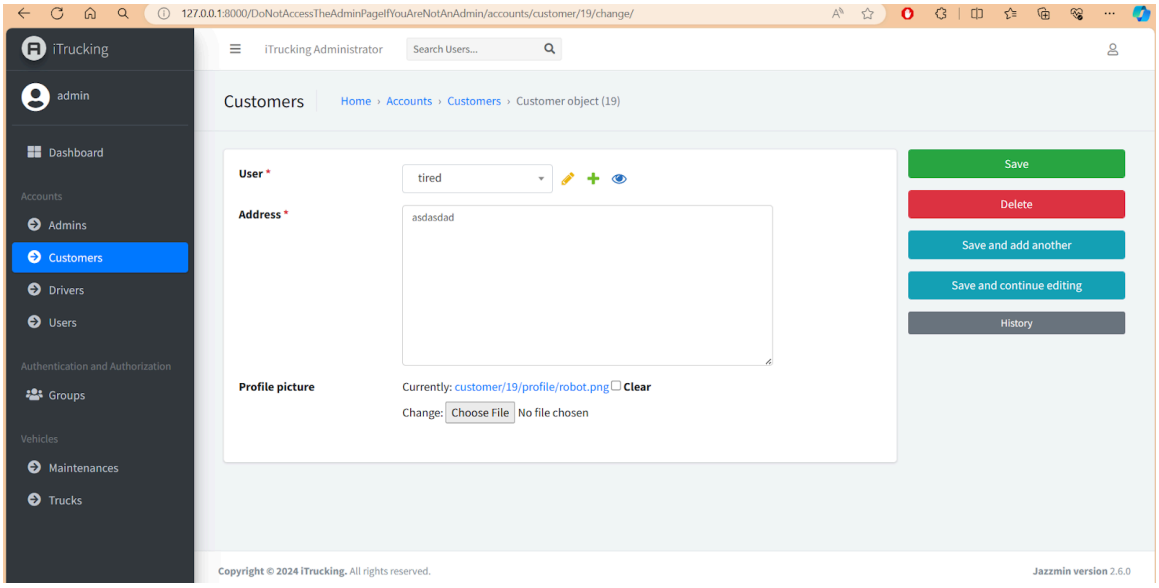
- Add new user by filling the form here



- Register the previously created user into the customer

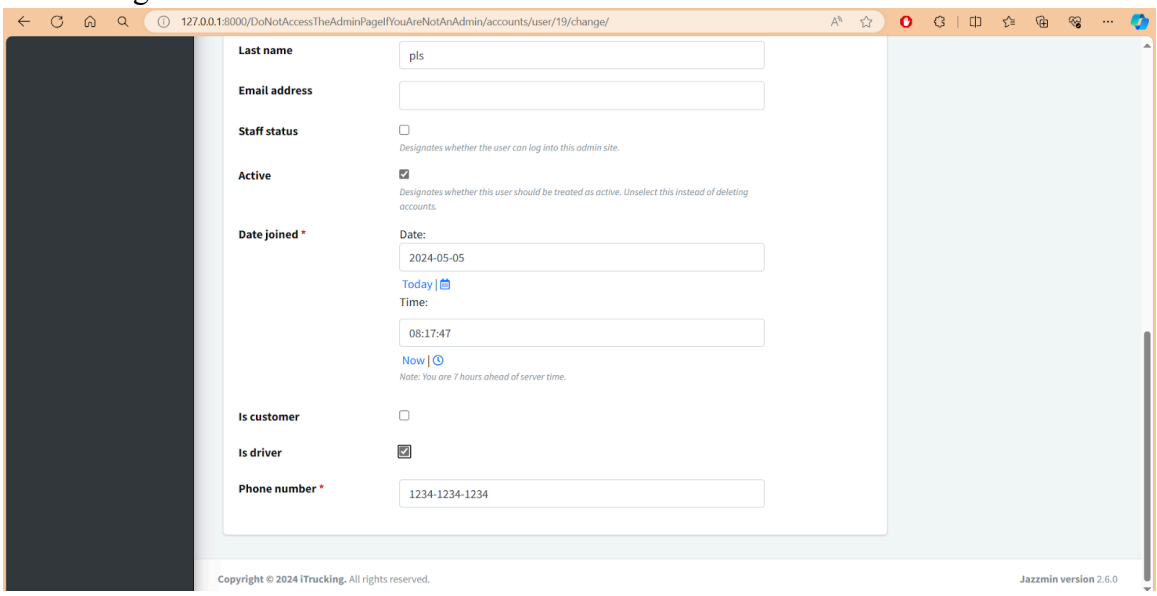


- Try to change the created object from before, which is the customer with username: tired, each object could be changed or deleted.

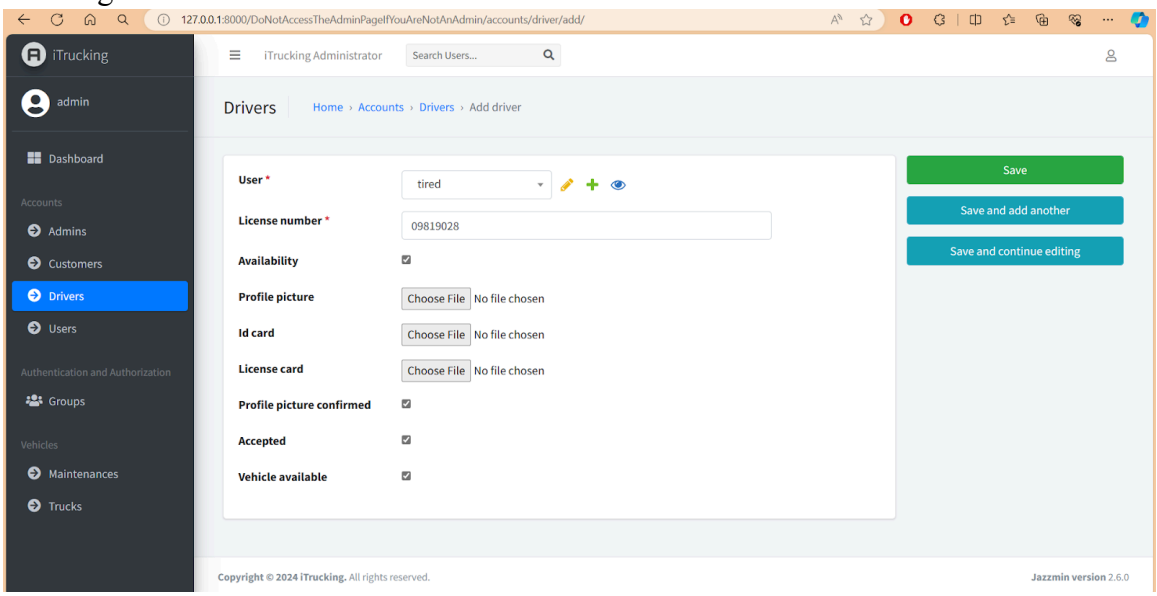


- The user could also be registered as a driver by changing the user, first delete the previous customer.

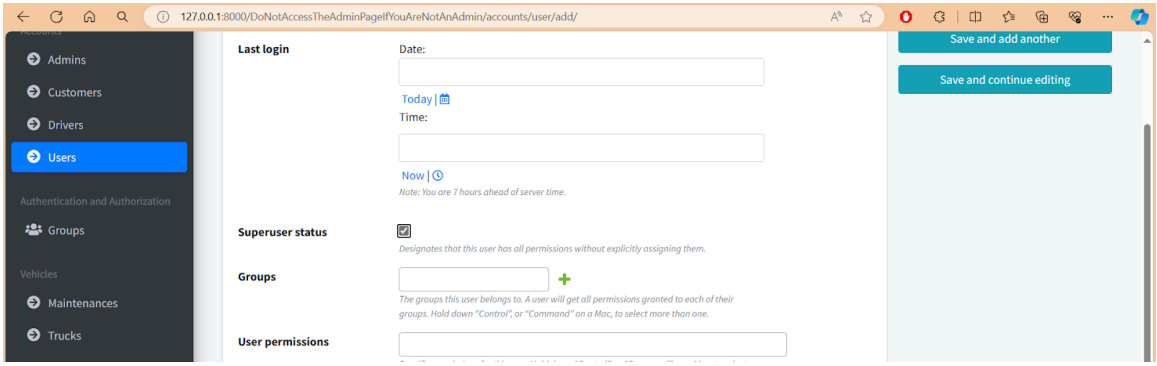
- Then change the user from is customer to is driver



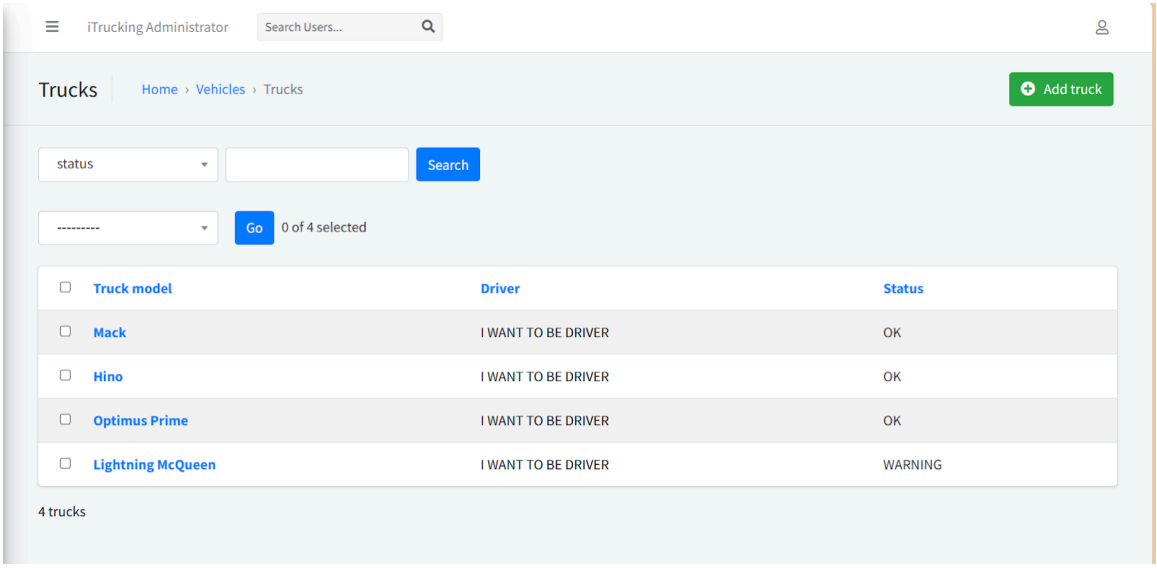
- Then register the user “tired” as driver.



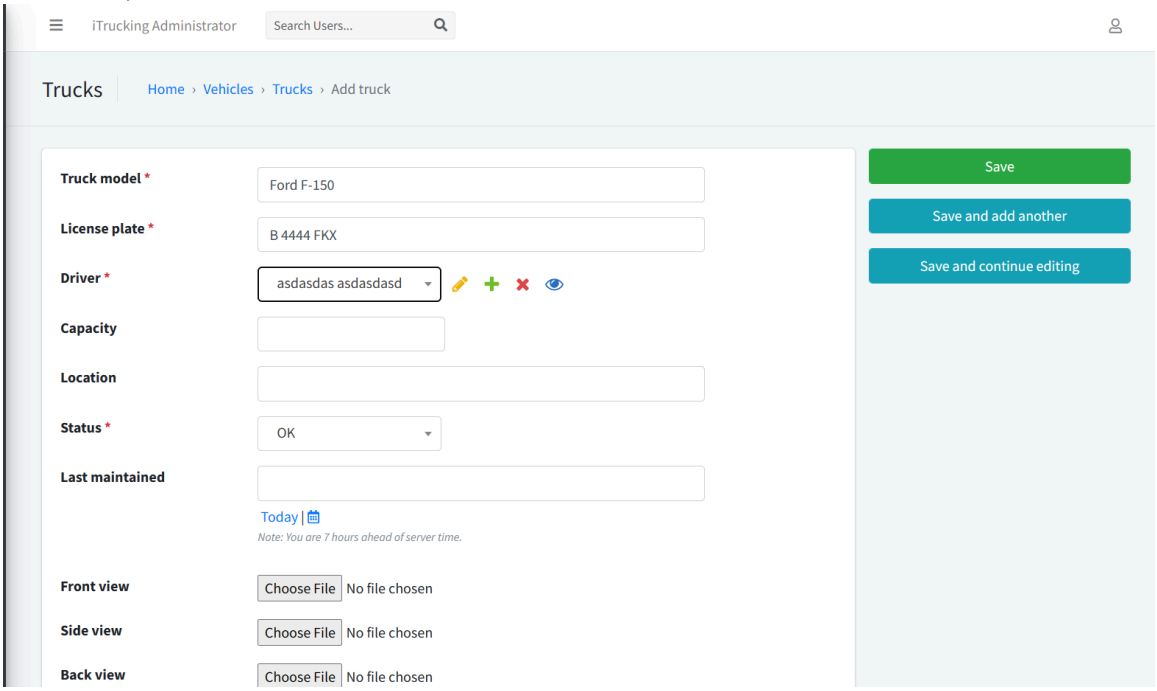
- The same applies for changing the user into superuser (administrator), could be done by changing the user status into superuser (by checking).



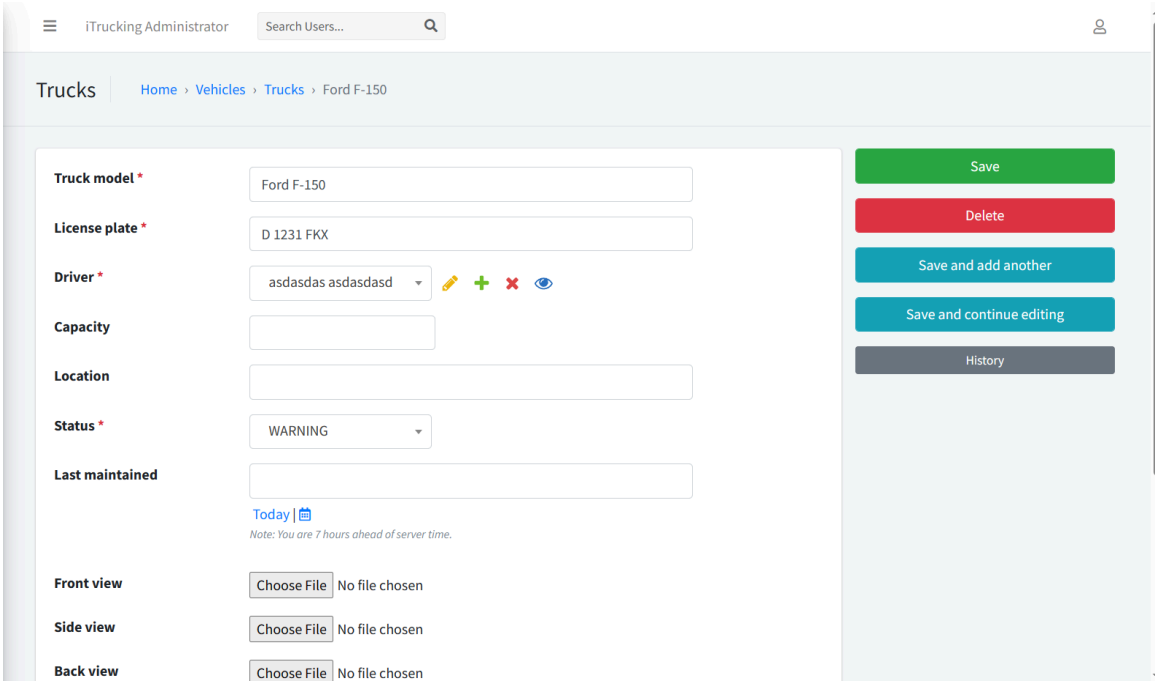
- Now for testing the truck management from the administrator side, Open the Truck menu, could filter by status, or could be searched.



- Test the add truck by creating a truck, and assigning it to a driver (asdasdas asdasdasd).

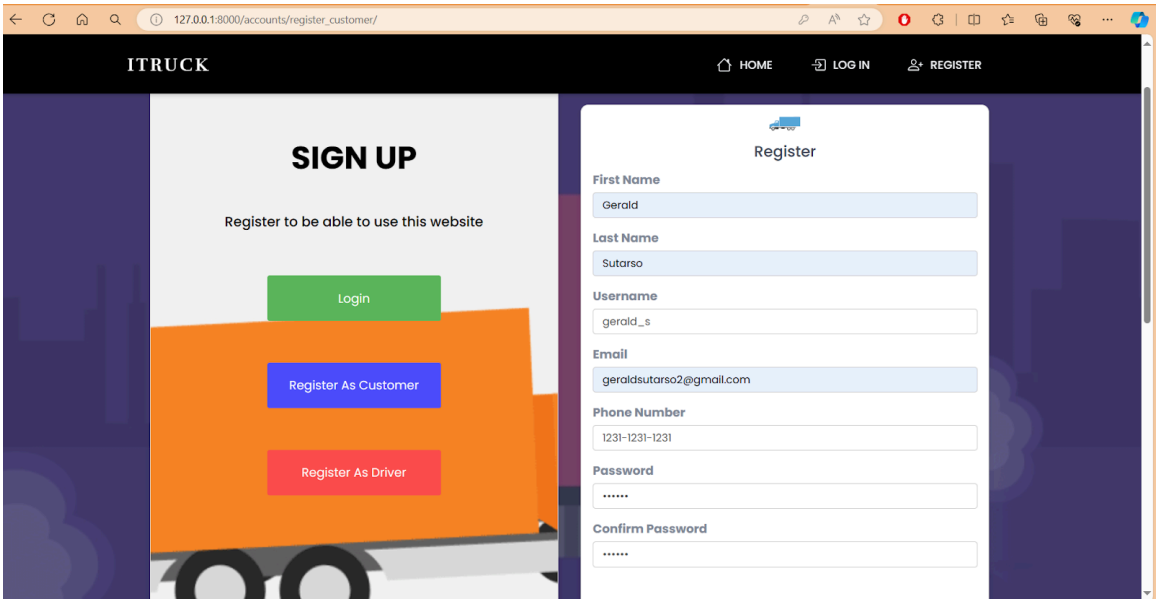


- Try editing the truck and deleting it.

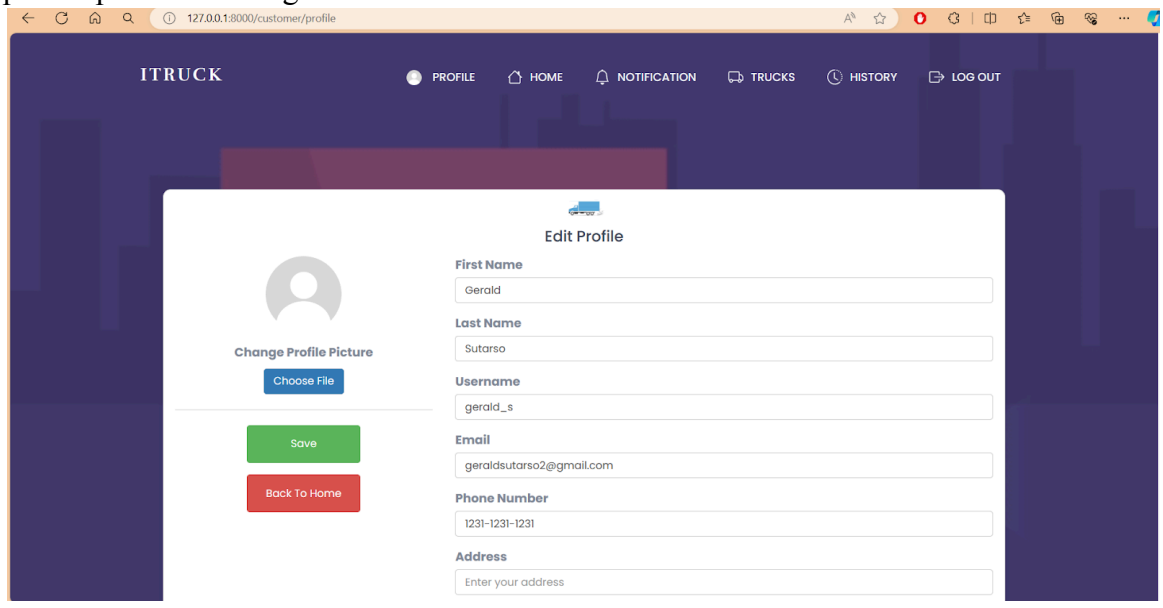


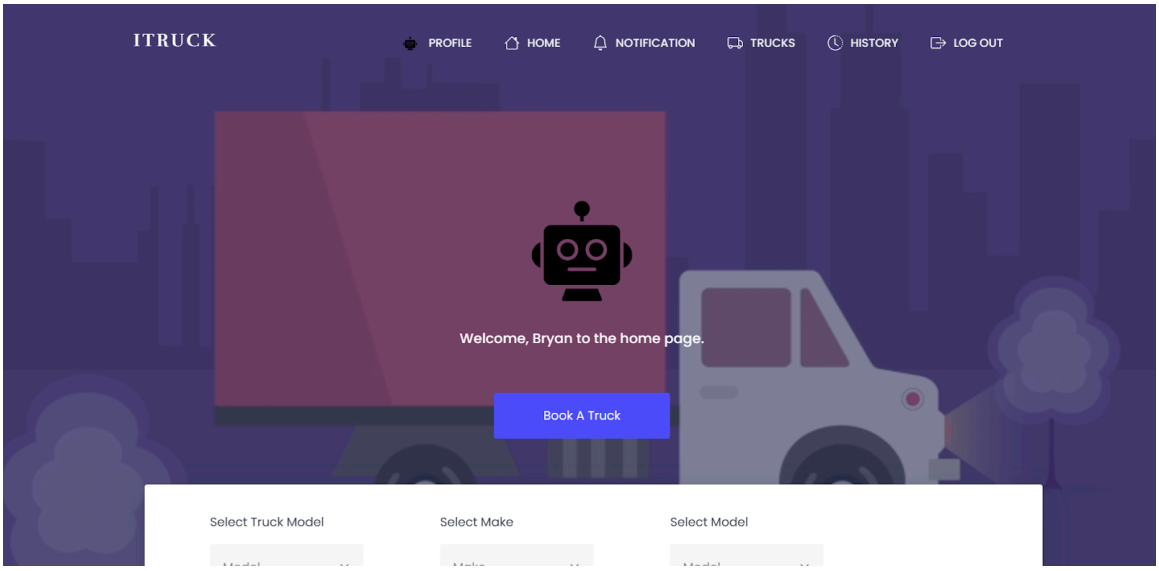
- After testing from the administrator side, test from customer side.

- Register as a customer to check whether “Create Customer Object” could work from the website’s customer side.

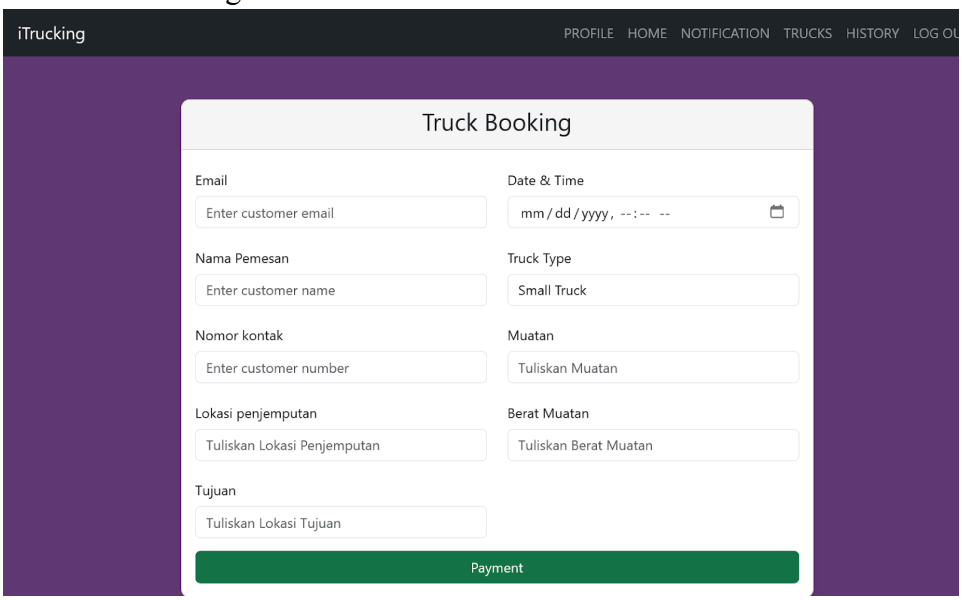


- Try to edit the user(customer) information through the profile(example: change name into Bryan) while also try to change the profile, checking whether the profile picture is changed.

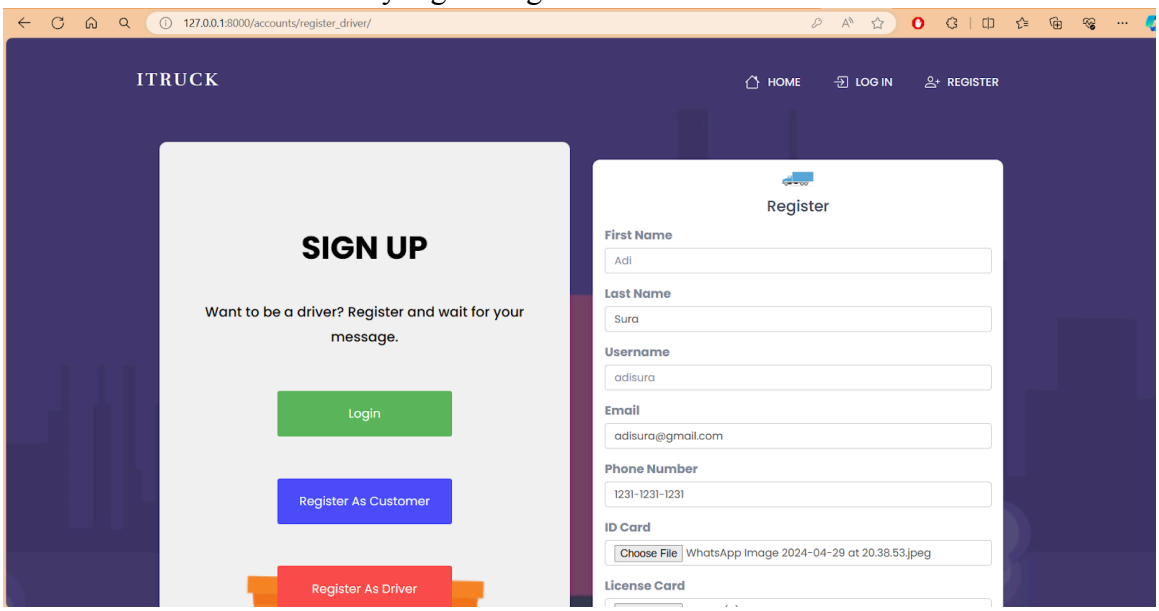




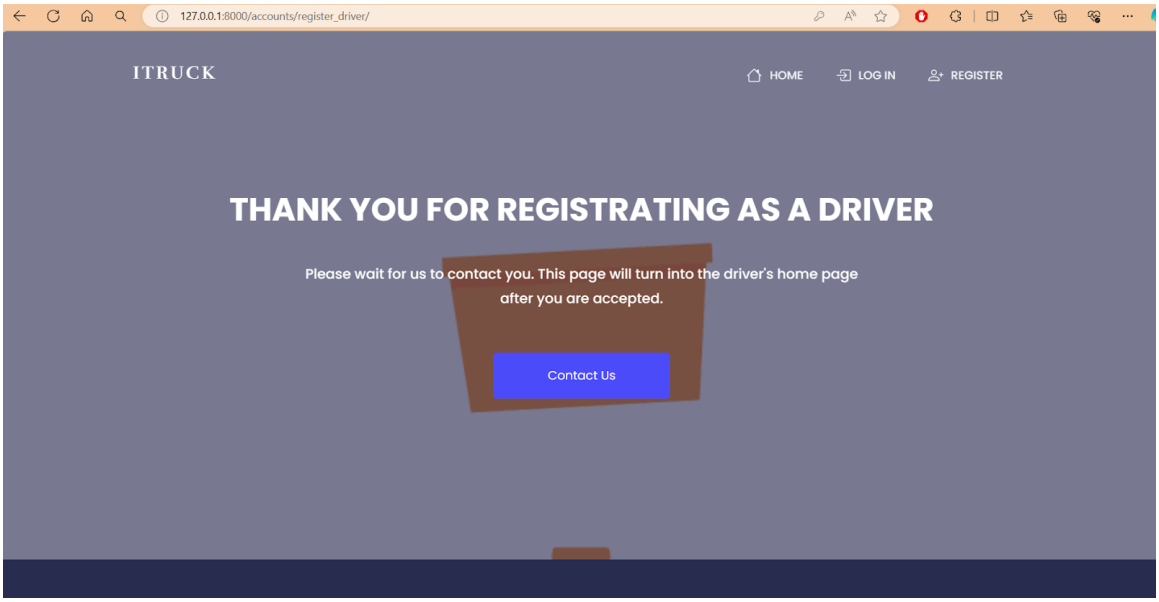
- Customer Booking the truck

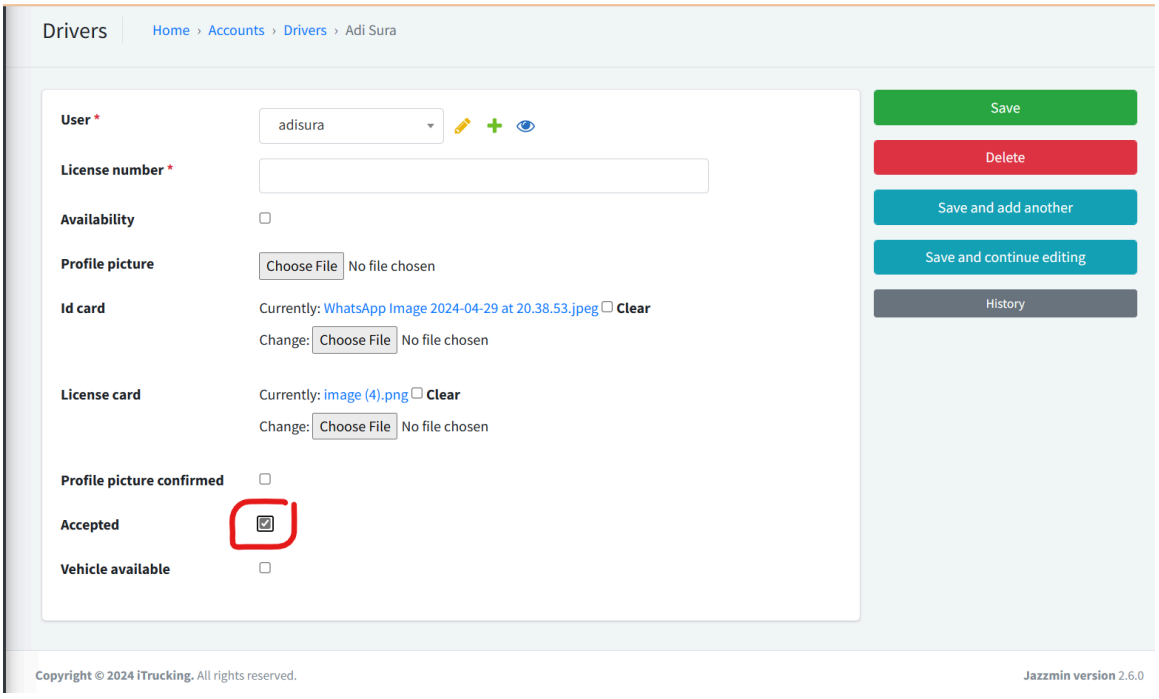


- **Now** check the “add driver” by registering as a driver from the website

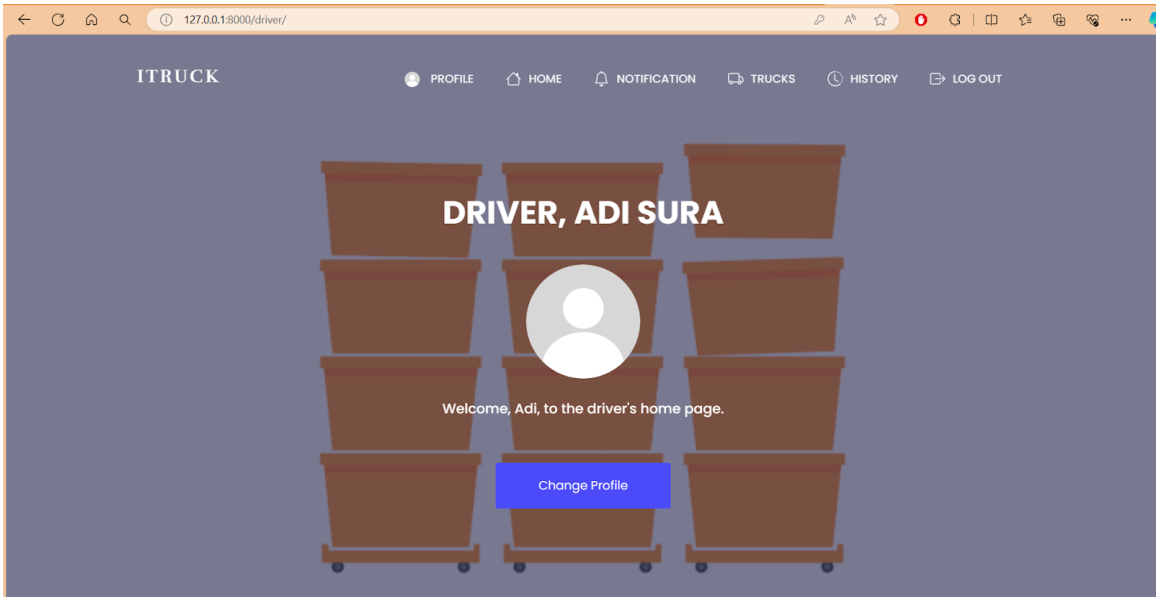


- Registered driver then should take an interview with the admin, to get accepted. To change the driver to be accepted, login as admin then check the accepted item in that driver’s object.

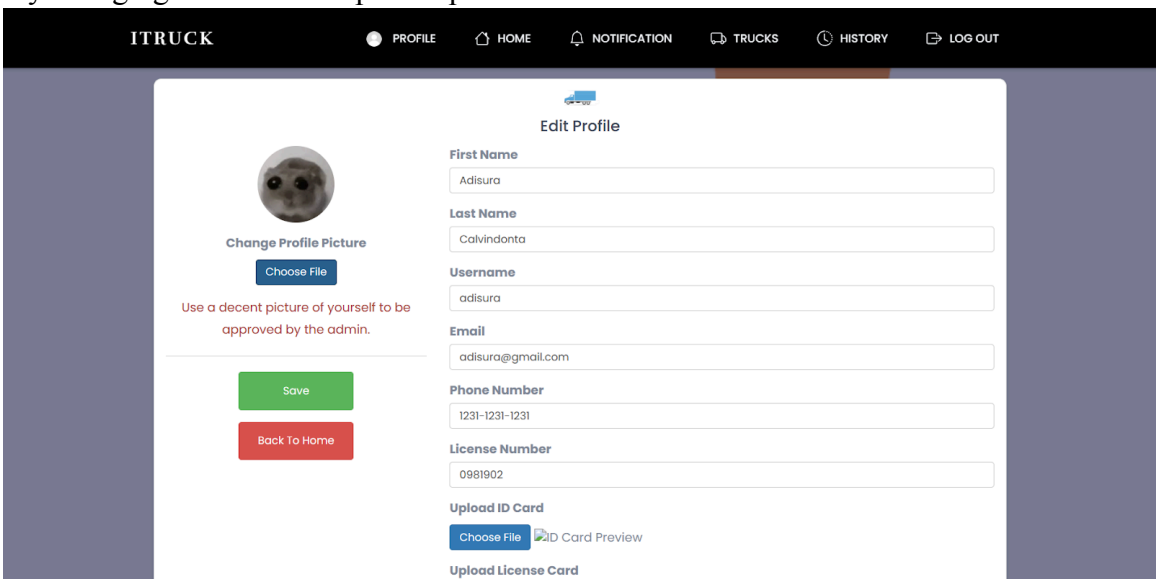

127.0.0.1:8000/accounts/register_driver/
ITRUCK
HOME
LOG IN
REGISTER
THANK YOU FOR REGISTRATING AS A DRIVER
Please wait for us to contact you. This page will turn into the driver's home page after you are accepted.
Contact Us

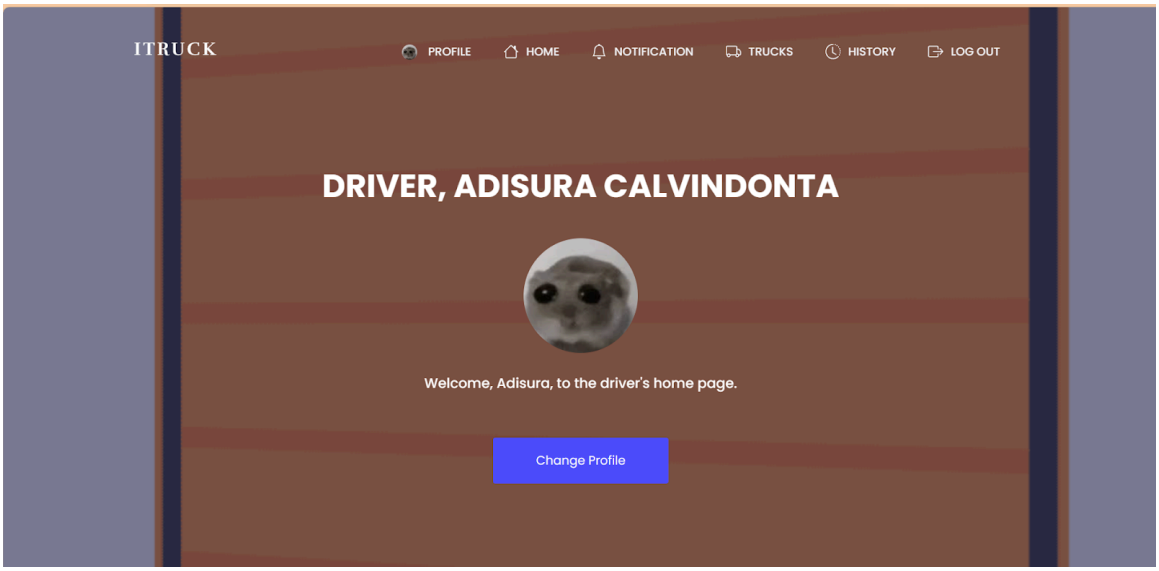

Drivers
Home › Accounts › Drivers › Adi Sura
User *
adisura
License number *
Availability
Profile picture
Choose File No file chosen
Id card
Currently: WhatsApp Image 2024-04-29 at 20.38.53.jpeg Clear
Change: Choose File No file chosen
License card
Currently: image (4).png Clear
Change: Choose File No file chosen
Profile picture confirmed
Accepted
Vehicle available
Save
Delete
Save and add another
Save and continue editing
History
Copyright © 2024 iTrucking. All rights reserved.
Jazzmin version 2.6.0

- Same testing as customer, check whether the profile could be changed from the driver side

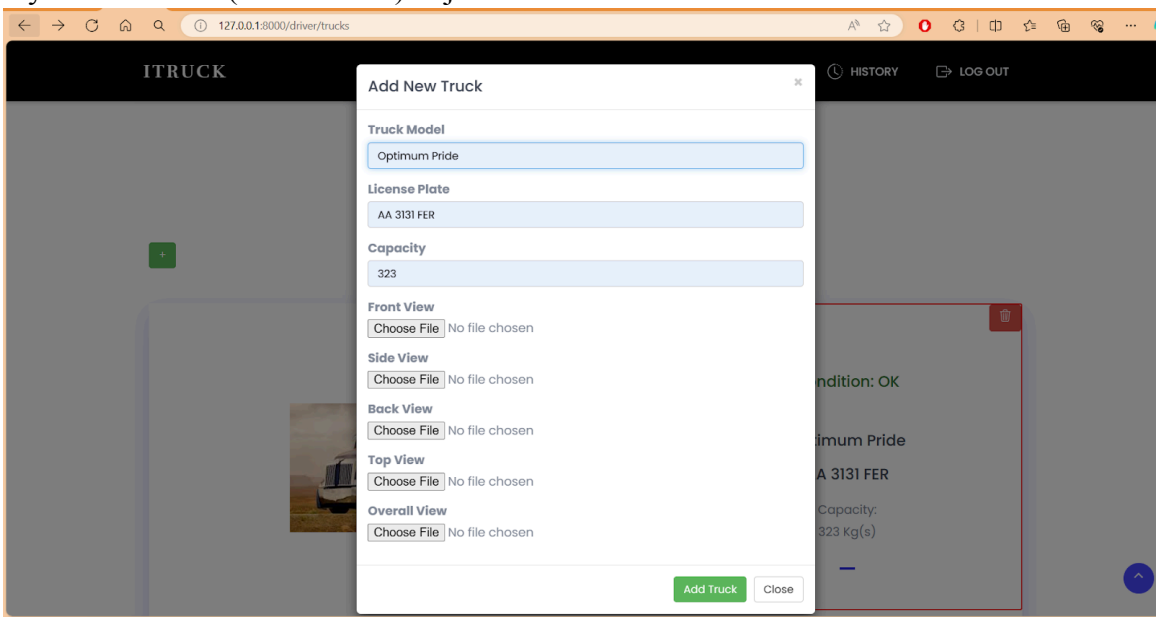


- Try changing the name and profile picture

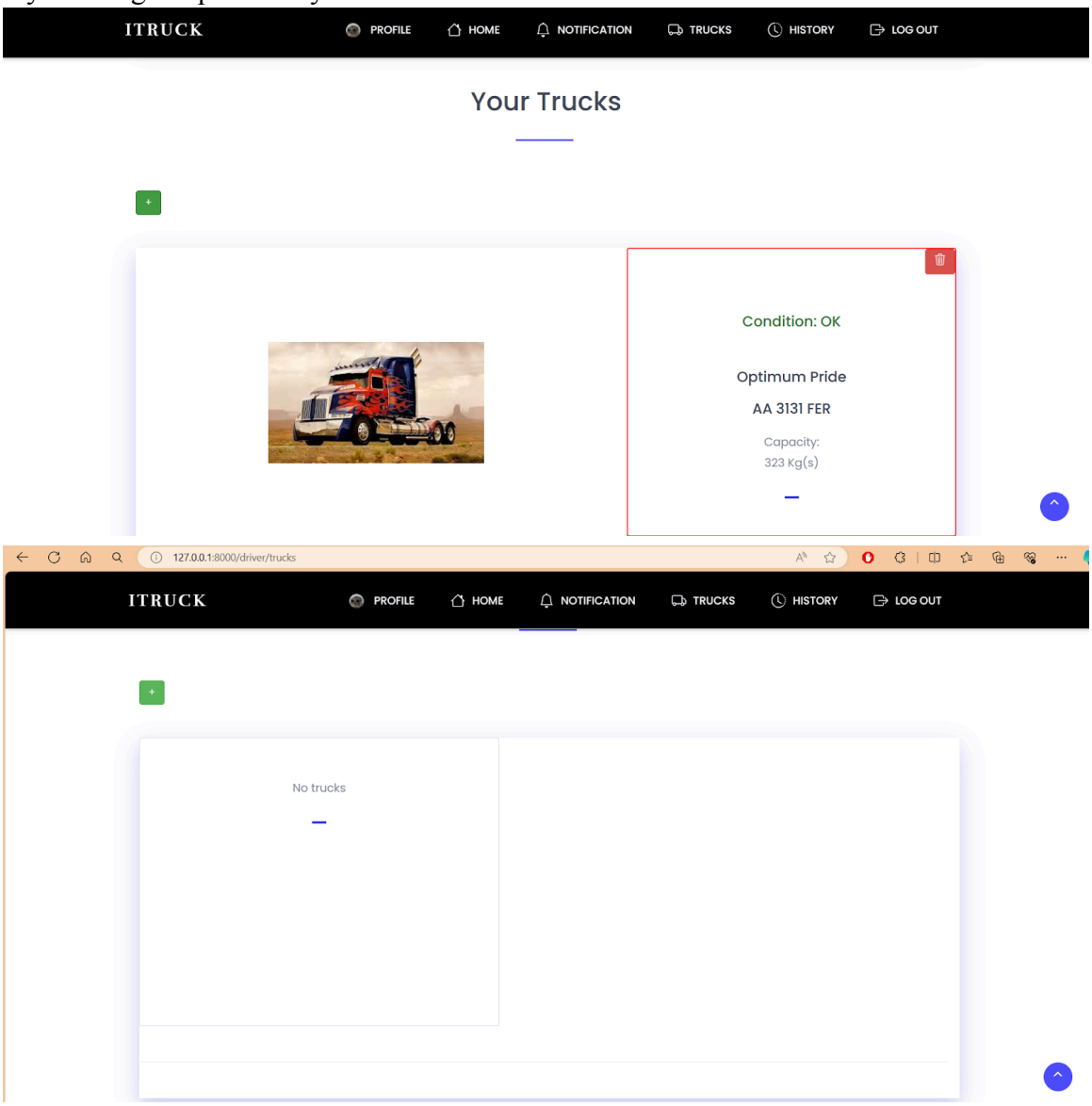




- Try the add truck (Create truck) object from the driver side



- Try deleting the previously created truck



VI.1 Testing other specifications

VI.1.1 *Non-functional specifications such as size, weight, etc., that are included in the document (PART C NO. 5 F200)*

1. Non-functional specifications

- Truck/Driver Recommendation System:

a. Objective: Verify that the recommendation system provides an accurate truck/driver suggestions list to the customers.

b. Method of Testing: Test whether the recommendation system could give minimum 3 accurate suggestions of trucks/drivers based on user preferences.

c. Quantity to be Measured: Successful and fitting recommendation for each customer.

- Payment Processing:

a. Objective: Confirm that the payment process can be successful.

b. Method of Testing: Execute payment transactions at least 3 times.

c. Quantity to be Measured: Successful completion of payment transactions.

- Vehicle Condition Testing:

a. Objective: Test the sensor's ability to transmit data to the computer remotely.

b. Method of Testing: Validate the vehicle condition sensor on a minimal set of three 4-tyre vehicles, ensuring remote data transmission capability.

c. Quantity to be Measured: Successful remote data transmission for each vehicle.

- Vehicle Tracking:

a. Objective: Verify the system's ability to provide correct real-time vehicle locations.

b. Method of Testing: Test vehicle tracking on a minimal set of three(3) 4 or more-tyre vehicles, ensuring accurate location reporting.

c. Quantity to be Measured: Accuracy of vehicle location tracking for each vehicle.

- Driver Quick-Medical Checkup:

a. Objective: Assess the system's accuracy in determining driver health based on facial expressions and eye conditions.

b. Method of Testing: Execute medical checkup tests on a minimum of 5 drivers, focusing on different facial expressions and eye conditions.

c. Quantity to be Measured: Successful health assessment for each driver tested.

1. *A photo/recording of the test is shown in the document*

VI.1.2 *Conclusion*

The conclusion of our project titled “iTrucking: Intelligent Trucking Management System for Efficient Logistics” tries to tackle issues with Indonesia's trucking industry with a web application.

In this project our group aimed to solve the problem statement From our F100 project and used the same specifications as stated in the F200 project. We have also done testing on our functionalities based on parts in F100 and F200.

Based on the testing conducted on our application, it can be concluded that the application demonstrates the functionality across multiple critical areas through testing, the application has exhibited great reliability. The testing process highlighted the functions of our applications including its user interface, and It’s multiple features which Include Payment, Truck Management, Fatigue Detection, Vehicle Tracking, and Machine Maintenance Detections damage, as well as its prompt notification system, underscores its effectiveness in facilitating timely response and maintenance efforts.

However, it's essential to acknowledge certain areas for potential improvement identified during testing, such as minor usability issues, occasional performance lags under high traffic loads, and the need for enhanced error detection and recovery mechanisms. Overall, the functional testing results affirm the Road Ranger Mobile Application's capability to fulfill its intended purpose of facilitating track and road damage tracking effectively. Continued efforts to address identified areas for enhancement will further strengthen the application's usability, reliability, and overall user experience.

REFERENCES

[1] NEO-6 series. (n.d.). u-blox. https://www.u-blox.com/en/product/neo-6-series

[2] GitHub - GeraldSutarso/Trucking-Management: Trucking Management web. (n.d.). GitHub. https://github.com/GeraldSutarso/Trucking-Management

[3] Difference between SIM800L, SIM800A, SIM900A. (2022, January 3). Hatchnhack Cart. https://www.hnhcart.com/blogs/sensors-modules/difference-between-sim800l-sim800a-sim900a#:~:text=The%20SIM800L%20is%20a%20GSM,GPRS,%20TCP,%20or%20IP.

[4] APA ITU ARDUINO UNO DAN KEGUNAANNYA | D3 Teknologi Telekomunikasi. (n.d.). D3 Teknologi Telekomunikasi. https://dte.telkomuniversity.ac.id/apa-itu-arduino-uno-dan-kegunaannya/

[5] Blockchain in Logistics & Transportation. (n.d.). PixelPlex. https://pixelplex.io/blog/blockchain-for-transport-and-logistics/

[6] Platforms and Blockchain Will Transform Logistics. (n.d.). Harvard Business Review. https://hbr.org/2019/06/platforms-and-blockchain-will-transform-logistics

[7] Acropolium spol. s r. o. (n.d.). Bespoke Software Development Company — Acropolium. https://acropolium.com/blog/how-to-develop-a-gps-tracking-system-and-why-do-you-need-one/

[8] Reed Smith LLP. (n.d.). Commercial drone delivery: A solution to last-mile logistics | Global Air Freight | Perspectives. https://www.reedsmith.com/en/perspectives/global-air-freight/2022/01/commercial-drone-delivery-a-solution-to-lastmile-logistics

[9] Convoy. (n.d.). Digital Freight Network - Connecting Shippers & Carriers. https://convoy.com/digital-freight-network/

[10] Maersk. (n.d.). On the road to digitalisation in freight forwarding. https://www.maersk.com/insights/digitalisation/2023/02/17/digitalisation-in-freight-forwarding

[11] Nada, Y. A. (2017). Design and Implementation for Trucks Tracking System Using GPS Based on Semantic Web. International Journal of Trend in Scientific Research and Development, Volume-1(Issue-4). https://doi.org/10.31142/ijtsrd169

[12] Antara News. (2019, May 27). Pengiriman Paket Barang Alami Peningkatan Saat Ramadhan. https://www.antaranews.com/berita/889719/pengiriman-paket-barang-alami-peningkatan-saat-ramadhan

[13] Bisnis Indonesia. (n.d.). Refleksi Kasus Pencurian Barang Konsumen yang Menyentil GoTo. https://bisnisindonesia.id/article/refleksi-kasus-pencurian-barang-konsumen-yang-menyentil-goto