System Documentation

for

Online Petrol Delivery System

Version 3.0

Tutorial Section: T12L

Group No.: 6

Nur Iman binti Mohamad Idros	1231303620
Maryam binti Norazman	1211111809
Tan Jie Ting	1221102916
Fikrul Amsyar bin Azmin	241UC24167

Date: 12 February 2025

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
SRS in Part 1(as Ver 1.0) SDS in Part 2(as Ver 2.0.X) *System Documentation in Part 3 (as Ver 3.0) Draft Type and Number	Nur Iman binti Mohamad Idros Tan Jie Ting Maryam binti Norazman Fikrul Amsyar bin Azmin	Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded.	12/02/25

1 Project Management

1.1 Team Members

Name	Actor/Processes
Nur Iman binti Mohamad Idros	Supplier
Maryam binti Norazman	Admin
Tan Jie Ting	Customer
Fikrul Amsyar bin Azmin	Driver

1.2 Problem statement

The development of the online petrol delivery system has been challenging due to our limited understanding of the project's full scope and technical requirements. Without a deep understanding of the system's complexities, it is hard to make an informed design and implementation decisions. There is multiple tasks and roles that increases the work needed to be done. Additionally, coordinating work within the group has proven to be a major hurdle, as unclear task assignments have led to inefficiencies and delays. Project implementation also affected our group as we found it hard to get the desired results and outcomes These challenges have slowed our progress and highlighted the need for better planning, clearer communication, and a stronger foundational understanding of the project.

1.3 Project Plan

Progress / Task	W 1	W 2	W 3	W 4	W 5	W 6	W 7	W 8	W 9	W 10	W 11	W 12	W 13	W 14
Sprint 1 : Project Planning	🟩	🟩												
Define requirements	🟩	🟩												
System overview	🟩	🟩												
Sprint 2 : Requirements Analysis			🟩	🟩	🟩									
Gather requirements			🟩	🟩	🟩									
Finalise user and acceptance criteria			🟩	🟩	🟩									
Sprint 3 : Project Design						🟩	🟩							
Create wireframes						🟩	🟩							
Develop architecture design						🟩	🟩							
Develop database design						🟩	🟩							
Sprint 4 : System Authentication								🟩	🟩					
Customer Registration								🟩	🟩					
Driver Registration								🟩	🟩					
Sprint 5 : Furl Ordering and Price Calculation									🟩	🟩				
Order Placement									🟩	🟩				
Price Calculation									🟩	🟩				
Sprint 6 : Order Delivery and Real - Ttime Tracking										🟩	🟩			
Order Delivery										🟩	🟩			
Real - Time Tracking										🟩	🟩			
Sprint 7 : Testing and Feedback												🟩		
Perform Testing												🟩		
Incoporate feedback and refine features												🟩		
Sprint 8 : Project Deployment												🟩	🟩	
Deploy for real - world testing												🟩	🟩	
Conduct more feedback												🟩	🟩	
Sprint 9 : Project Reporting and Presentation													🟩	🟩
Project Reporting													🟩	🟩
													🟩	🟩
Project Presentation													🟩	🟩

Figure 1.3 : Project Gantt Chart

2 System Overview

2.1 Description

The Online Petrol Delivery System is designed to improve fuel delivery services for individuals, businesses, and industries. It has four primary customers: Admin, Customer, Driver, and Supplier, each have a distinct role in this system. The system facilitates customer and driver registration with authentication to ensure safety and reliability. Customers can compare petrol prices based on delivery distance and volume, as well as evaluate delivery times across available drivers to make informed decisions. Once orders are placed, the system verifies petrol and delivery purchases, enabling real-time tracking to ensure customers have full visibility over their orders. Additionally, customers can cancel orders and receive refunds, adding convenience to the service.

The system has efficient inventory management by keeping track of petrol stock levels and ensuring steady supply to meet demand from customer. The supplier also responsible for providing petrol pricing data so that customer would be able to have a fast and well updated pricing data. Admins oversee the entire operation so that the system can runs smoothly without causing issue. Drivers can confirm completion, providing a seamless end-to-end service. The system outputs include registered customer and driver accounts, price and delivery time comparisons, authenticated transactions, real-time tracking updates, inventory status, and order cancellation with refunds. By combining convenience, transparency, and efficiency, the Online Petrol Delivery System ensures a reliable and customer-centric fuel delivery experience.

2.2 Actors

Actor	Use Case
Customer	1. Registration 2. Login 3. Profile management 4. Order placement 5. Make payment 6. Track order 7. Refund request
Driver	1. Registration and authentication 2. Login 3. Track order
Admin	1. Login 2. Manage driver authentication 3. Manage refund request 4. Manage customer profile
Supplier	1. Login 2. Prepare order 3. Update pricing data and stock inventory

Figure 2.2 : Actors Table

2.3 Assumptions and Dependencies

The system assumes the availability of reliable third-party APIs, such as payment gateways as this is critical for smooth operation. If these APIs become unavailable and unreliable the system's functionality and customer experience could be negatively impacted. Additionally, stable internet connectivity is presumed for all customers, drivers, suppliers and administrators. Poor or inconsistent internet connections could lessen the system's performance, hinder real-time tracking, and reduce customer satisfaction. Another important assumption is hardware and device compatibility. Customers, drivers, suppliers and administrators will have access to devices capable of running the application or web interface. If customers rely on incompatible or outdated devices, it could severely affect system performance or prevent the system from functioning on the device.

The system also has key dependencies that influence its operation. It relies on secure and efficient payment gateway services, such as FPX Net, to handle transactions. Furthermore, the system is heavily reliant on the availability of drivers and suppliers. Any shortage or lack of availability in these roles could disrupt operations and lessen the system's ability to meet customer demands effectively.

2.4 Use Case Diagram

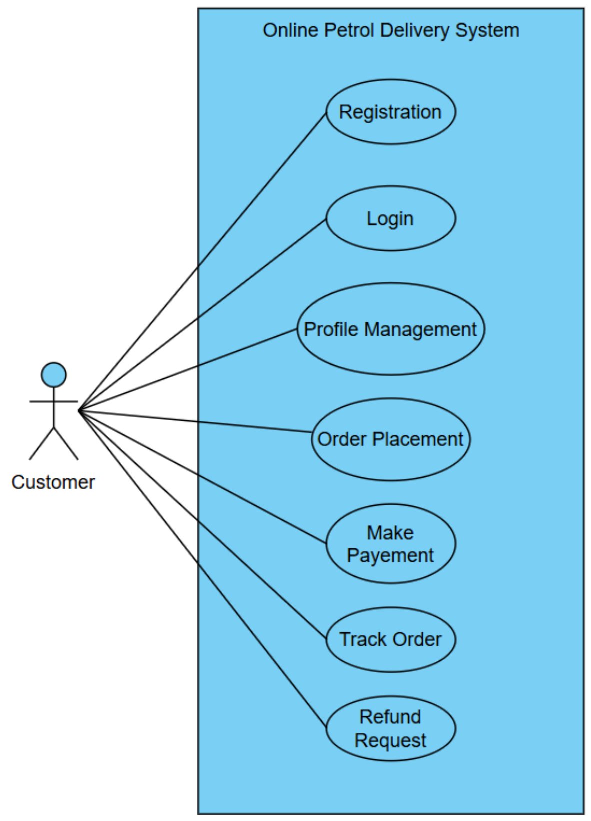


Figure 2.4.1 Customer Use Case Diagram

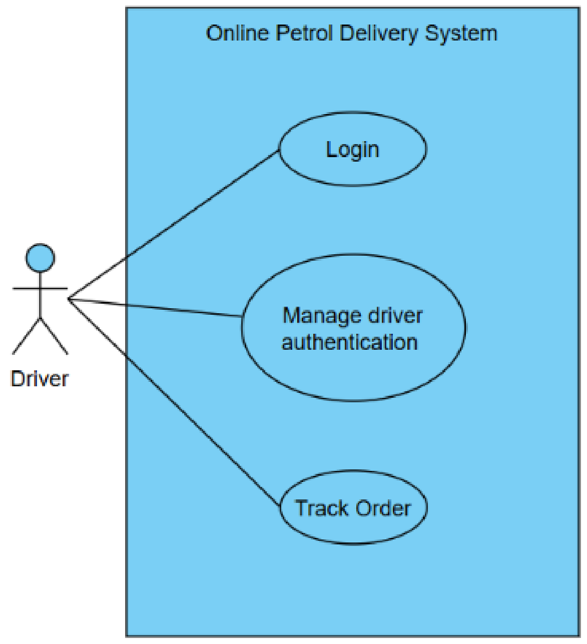


Figure 2.4.2 Driver Use Case Diagram

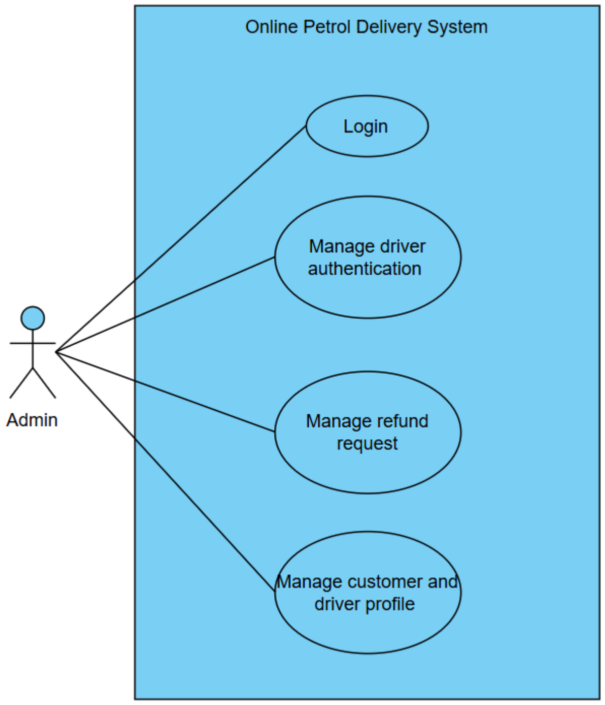

Online Petrol Delivery System
Login
Manage driver authentication
Manage refund request
Manage customer and driver profile
Admin

Figure 2.4.3 Admin Use Case Diagram

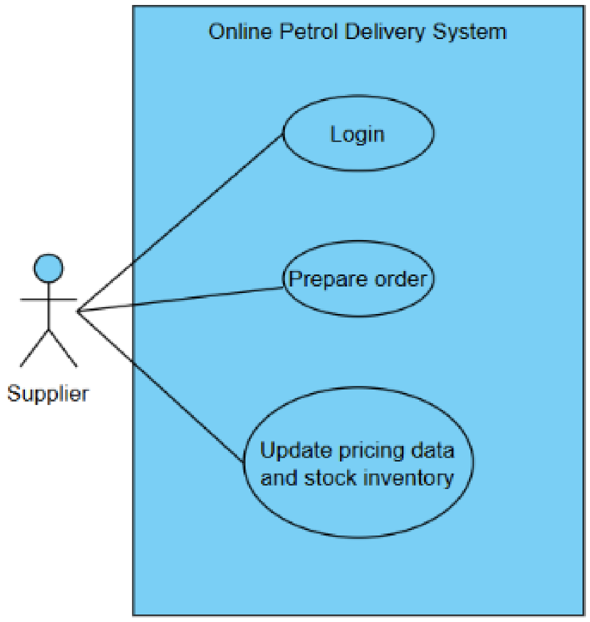

Online Petrol Delivery System
Login
Prepare order
Update pricing data and stock inventory
Supplier

Figure 2.4.4 Supplier Use Case Diagram

3 Requirements

3.1 Class Diagrams / ERD

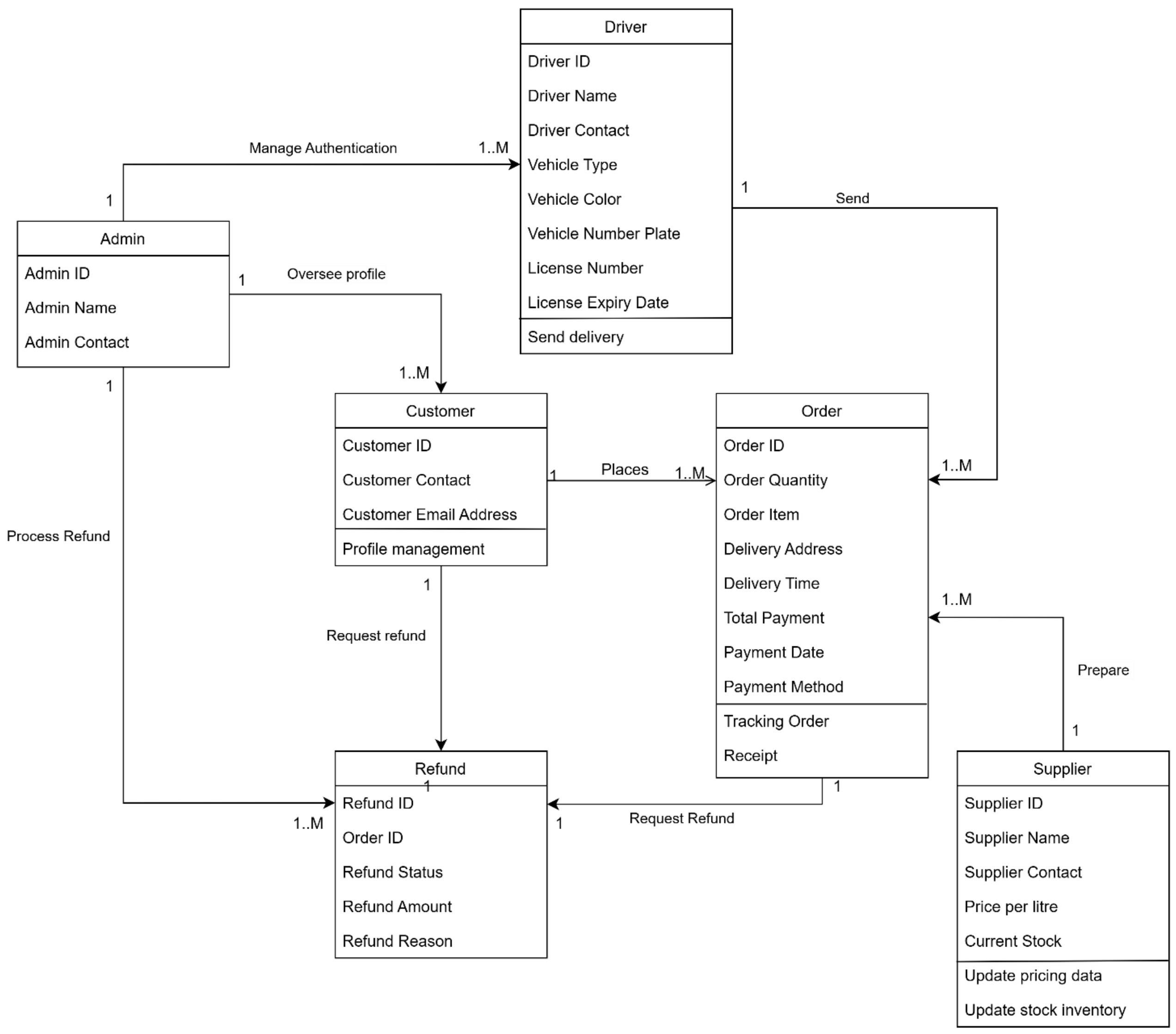
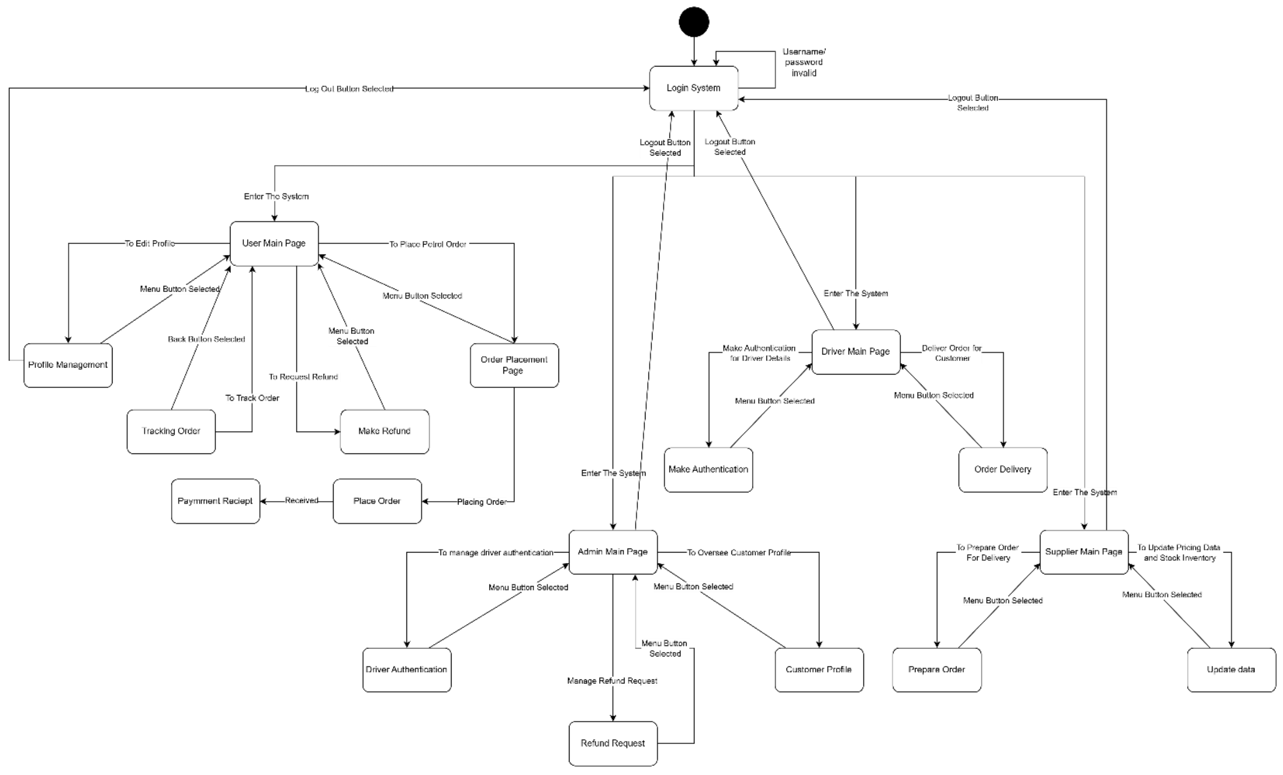


Figure 3.1 : Class Diagram

3.2 State Diagrams



4

Figure 3.2 : State Diagram

5 Design

5.1 Data Dictionary

Attribute Name	Data type	Format	Description	PK or FK	FK Referencing Table
DriverID	Varchar (5)	D0001	Unique identifier for driver	PK	-
DriverName	String	Text	Full name of the driver	-	-
DriverContact	Varchar (12)	Phone number	Contact number of the driver	-	-
VehicleType	Varchar (15)	Alphanumeric	Type of vehicle (e.g., car, bike)	-	-
VehicleColor	string	Text	Color of the vehicle	-	-
VehicleNumberPlate	Varchar (10)	Alphanumeric	License plate number of the vehicle	-	-
License Number	Varchar (7)	Alphanumeric	Driver's license number	-	-
LicenseExpiry Date	Date	DD/MM/YYYY	Expiry date of the driver's license	-	-

Table 4.1.1 : Driver Table

Attribute Name	Data type	Format	Description	PK or FK	FK Referencing Table
AdminID	Varchar (5)	A0001	Unique identifier for admin	PK	-
AdminName	String	Text	Full name of the admin	-	-
AdminContact	Varchar (12)	Phone number	Contact number of the admin	-	-

Table 4.1.2 : Admin Table

Attribute Name	Data type	Format	Description	PK or FK	FK Referencing Table
CustomerID	Varchar (5)	C0001	Unique identifier for customer	PK	-
CustomerName	String	Text	Full name of the customer	-	-
CustomerContact	Varchar (12)	Phone number	Contact number of the customer	-	-
EmailAddress	Varchar (15)	Alphanumeric	Email address of the customer	-	-
DeliveryAddress	Varchar (60)	Text	Address for order delivery	-	-

Table 4.1.3 : Customer Table

Attribute Name	Data type	Format	Description	PK or FK	FK Referencing Table
OrderID	Varchar (5)	OD001	Unique identifier for order	PK	-
CustomerID	Varchar (5)	C0001	Identifier linking to the customer	FK	Customer
DriverID	Varchar (5)	D0001	Identifier linking to the driver	FK	Driver
OrderItem	Varchar (15)	Alphanumeric	Name of the item ordered	-	-
OrderQuantity	Integer	Numeric	Quantity of the item ordered	-	-
DeliveryTime	DateTime	YYYY-MM-DD HH:MM	Total payment for the order	-	-
TotalPayment	Decimal (5,2)	Numeric	Driver's license number	-	-
PaymentMethod	String	Text	Method of payment (e.g., cash, card)	-	-
PaymentDate	Date	DD/MM/YYYY	Date of payment being made	-	-

Table 4.1.4 : Order Table

Attribute Name	Data type	Format	Description	PK or FK	FK Referencing Table
RefundID	Varchar (5)	R0001	Unique identifier for refund	PK	-
OrderID	Varchar (5)	OD001	Identifier linking to the order	FK	Order
RefundStatus	String	Text	Status of the refund (e.g., completed)	-	-
RefundAmout	Decimal (5,2)	Numeric	Amount to be refunded	-	-
RefundReason	String	Text	Reason for the refund request	-	-

Table 4.1.5 : Refund Table

Attribute Name	Data type	Format	Description	PK or FK	FK Referencing Table
SupplierID	Varchar (5)	S0001	Unique identifier for supplier	PK	-
SupplierName	String	Text	Full name of the supplier	-	-
SupplierContact	Varchar (12)	Phone number	Contact number of the supplier	-	-
PricePerLitre	Decimal (3,2)	Numeric	Price per litre of the product	-	-
CurrentStock	Varchar (10)	Numeric	Current stock level of the product	-	-

Table 4.1.6 : Supplier Table

5.2 Software Architecture

The application-level architecture describes the overall structure of the system, including the frontend, backend, and data layer. This architecture ensures that the system is scalable, maintainable, and efficient.

5.2.1 High Level Diagram

The system is divided into multiple layers, including the presentation layer (frontend), business logic layer (backend), and data layer. This separation ensures modularity and allows for independent development and scaling of each layer.

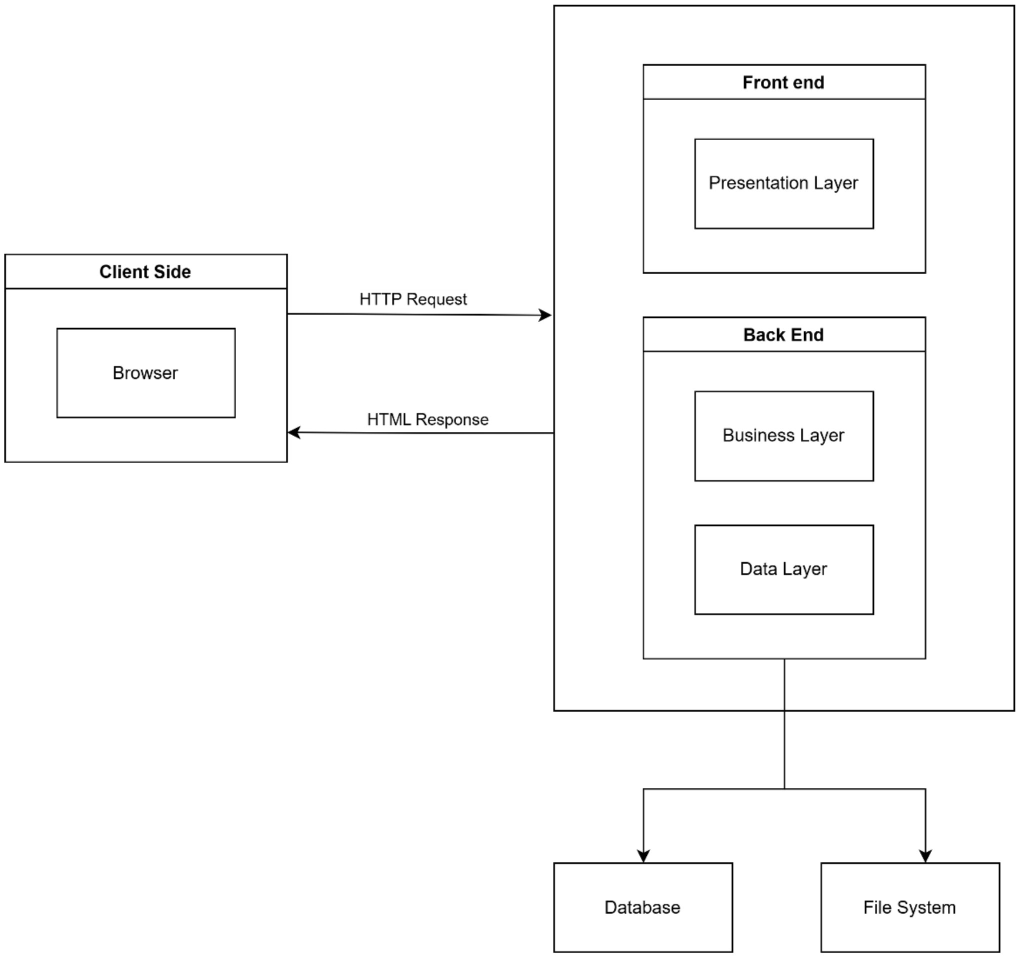


Figure 5.1.1 : High Level Diagram

1. **Client Side**
   - Represents the user's device and interface.
   - Contains a **Browser**, which sends HTTP requests and receives HTML responses.
2. **Front End**
   - The **Presentation Layer** is responsible for displaying data to users.
   - It handles user interactions and sends requests to the backend.
3. B**ack End**

- The **Business Layer** processes requests, applies logic, and manages communication between the frontend and data storage.
- The **Data Layer** manages database interactions and file storage.

4. **Database**
   - Stores structured data such as user information, orders, and transactions.
5. **File System**
   - Stores unstructured data like images, logs, and documents.
6. **Arrows (HTTP Request & HTML Response)**
   - Represents communication between the client and backend via HTTP requests and responses.

5.2.2 Frontend Architecture Diagram

The system is divided into multiple layers, including the presentation layer (frontend), business logic layer (backend), and data layer. This separation ensures modularity and allows for independent development and scaling of each layer.

5.2.3 Backend Architecture Diagram

The backend handles the business logic, such as processing orders, managing refunds, updating inventory, and authenticating users. It consists of multiple microservices or modules, each responsible for specific functionalities (e.g., order management, payment processing, etc.).

5.2.4 Data Layer Architecture Diagram

The data layer manages the storage and retrieval of data. It includes databases for storing user profiles, order details, inventory, payment records, and more. The data layer ensures data consistency, security, and scalability

5.2.5 Overall Architecture Diagram

The system is further divided into subsystems based on user roles and functionalities. Each subsystem is assigned to a team member or a group of team members for development and maintenance.

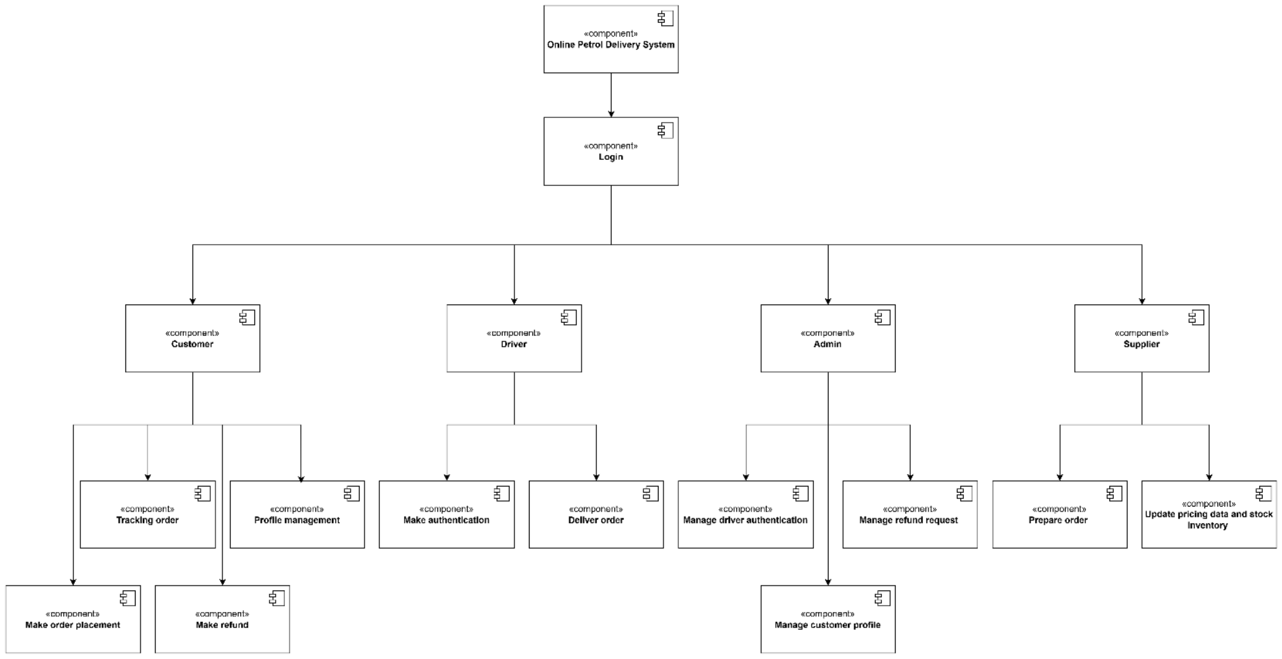

«component» Online Petrol Delivery System
«component» Login
«component» Customer
«component» Driver
«component» Admin
«component» Supplier
«component» Tracking order
«component» Profile management
«component» Make authentication
«component» Deliver order
«component» Manage driver authentication
«component» Manage refund request
«component» Prepare order
«component» Update pricing data and stock inventory
«component» Make order placement
«component» Make refund
«component» Manage customer profile

Figure 5.1.2 : Overall Architecture Diagram

Subsystem	Team members
Admin	Maryam binti Norazman
Customer	Tan Jie Ting
Supplier	Nur Iman binti Mohamad Idros
Driver	Fikrul Amsyar Azmin

5.2.6 Login Subsystem

This subsystem handles user authentication and authorization. It ensures that users (customers, drivers, admins, and suppliers) can securely log in and access their respective dashboards. Features include password management, session handling, and role-based access control.

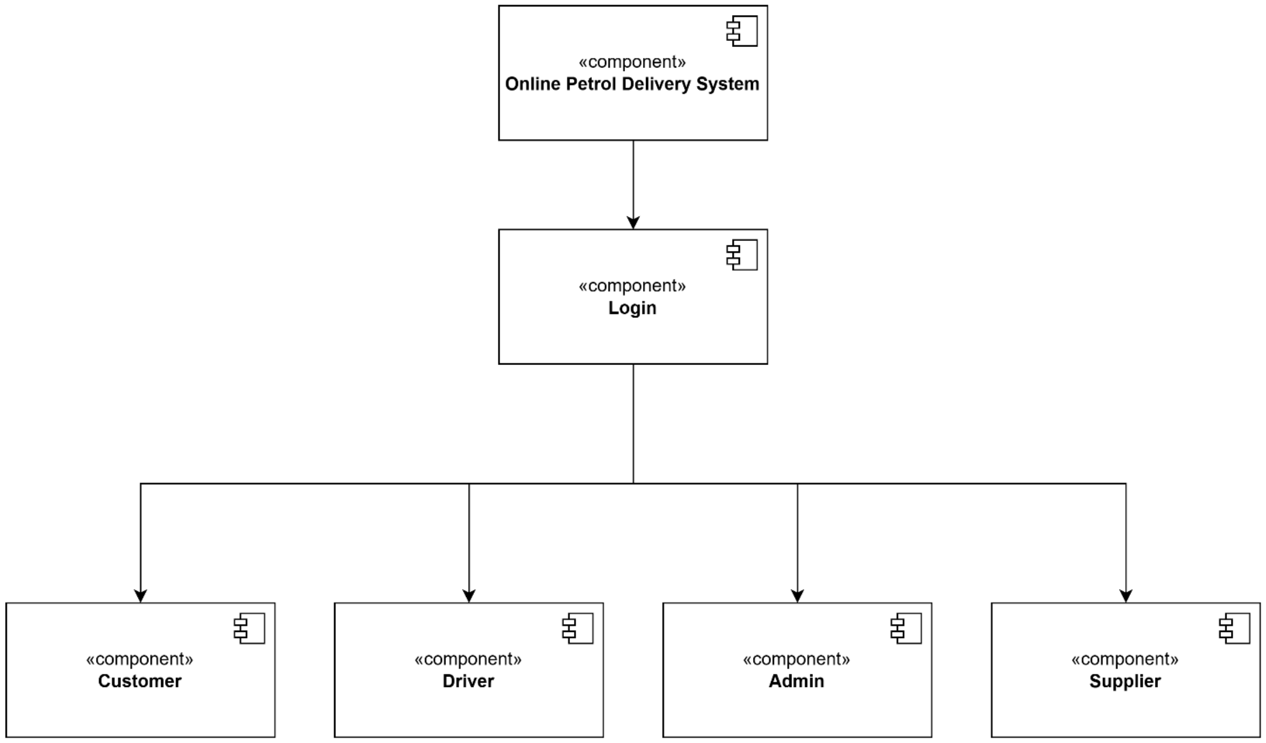


Figure 5.2.6 Login Subsystem

5.2.7 Customer Subsystem

The customer subsystem allows users to place orders, track their delivery status, request refunds, and manage their profiles. It includes features like order placement, payment processing, and real-time order tracking.

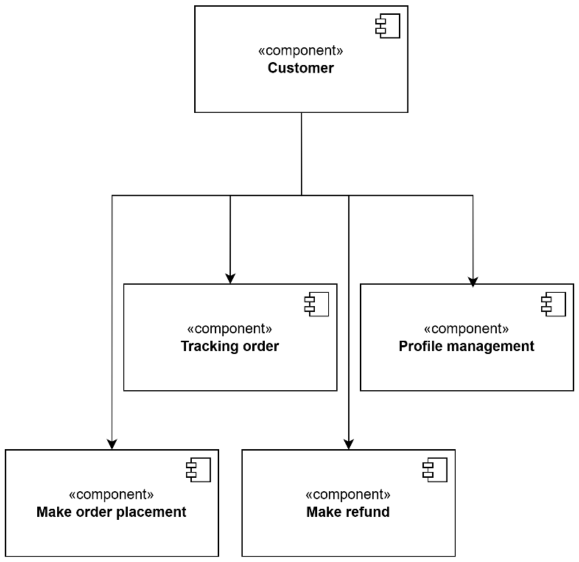


Figure 5.2.7 Architecture Diagram (Customer)

5.2.8 Driver Subsystem

The driver subsystem allows drivers to view assigned orders, update delivery status, and manage their profiles. It includes features like order tracking, route optimization, and delivery confirmation.

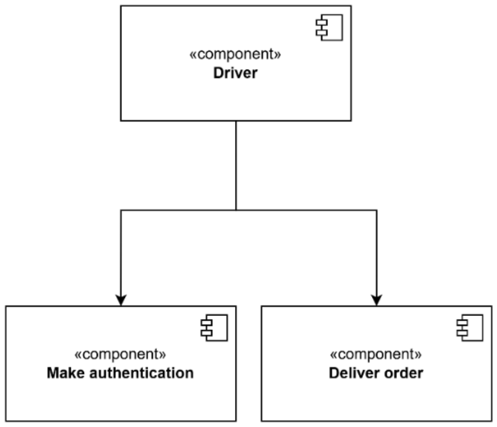


Figure 5.2.8 Architecture Diagram (Driver)

5.2.9 Admin Subsystem

The admin subsystem is responsible for managing the entire system. Admins can oversee user profiles, manage refund requests, update inventory, and monitor driver authentication. This subsystem ensures that the platform runs smoothly and securely.

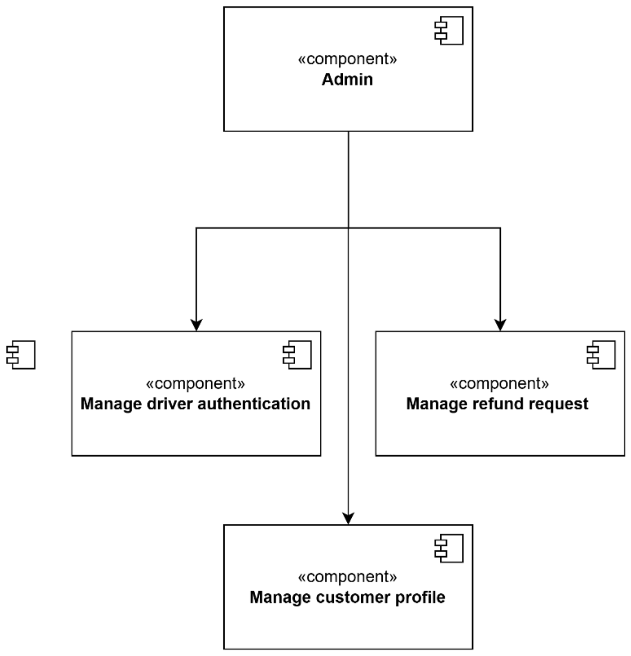


Figure 5.2.9 Architecture Diagram (Admin)

5.2.10 Supplier Subsystem

The supplier subsystem enables suppliers to manage inventory, update pricing, and prepare orders for delivery. It ensures that suppliers can efficiently fulfill customer orders and maintain accurate stock levels.

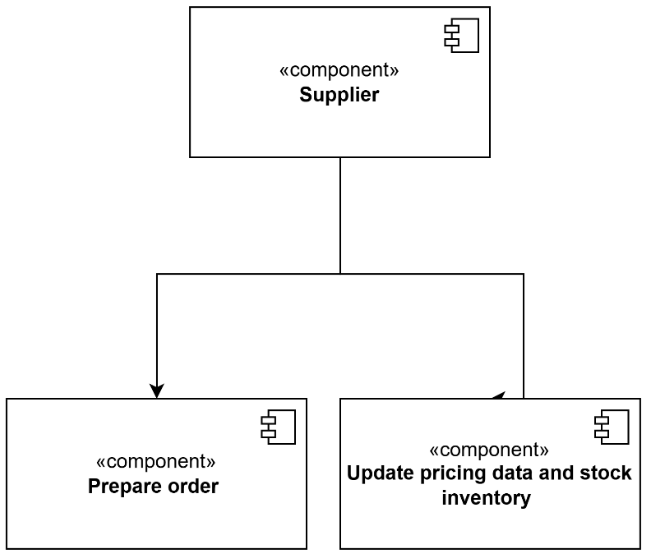


Figure 5.2.10 Architecture Diagram (Supplier)

5.3 Main Screens

5.3.1 Login Page

Figure 5.1 illustrates a login interface designed for all users, providing a simple and welcoming way to access their accounts. The interface is user-friendly and focuses on ease of use.

Key components include:
1. Welcome Message: A friendly greeting ("Hi! Welcome") creates a positive first impression.
2. Input Fields: Users can enter their username, email, or phone number along with their password to log in.
3. Log In Button: A button to submit the login credentials and access the account.
4. Sign Up Option: A prompt for users who do not have an account to sign up, providing an easy transition to the registration process.

The design is straightforward and intuitive, ensuring that users can quickly and securely access their accounts. This approach is essential for maintaining a seamless user experience and encouraging continued use of the platform.

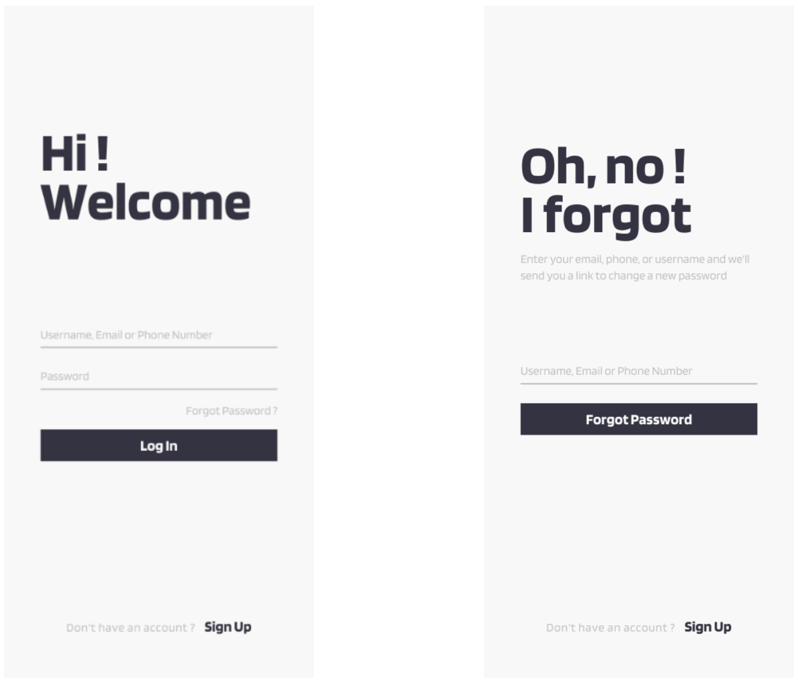

Hi ! Welcome
Username, Email or Phone Number
Password
Forgot Password ?
Log In
Don't have an account ? Sign Up
Oh, no ! I forgot
Enter your email, phone, or username and we'll send you a link to change a new password
Username, Email or Phone Number
Forgot Password
Don't have an account ? Sign Up

Figure 5.1 Login Page

<TO DO: Describe the main screens of the system and place the screen designs here.>

5.3.2 Registration Page

Figure 5.2 illustrates a registration interface designed for all types of users, including regular users and drivers. The interface is welcoming and straightforward, guiding users through the account creation process.

Key components include:
1. Welcome Message: A friendly greeting ("Hi! Welcome") and a prompt to create an account, setting a positive tone for the registration process.
2. Input Fields: Users are required to enter their email or phone number, full name, username, and password. The password fields include guidelines (e.g., must contain a number and be at least 8 characters) to ensure security.
3. User Type Selection: Users can specify their role (e.g., User, Driver), ensuring that the system can tailor the experience based on their needs.
4. Sign Up Button: A button to complete the registration process, allowing users to create their account.
5. Login Option: A prompt for users who already have an account to log in, providing an alternative to registration.

The design is user-friendly and inclusive, ensuring that all users can easily create an account and access the platform. This approach is essential for facilitating a smooth onboarding experience and encouraging user engagement.

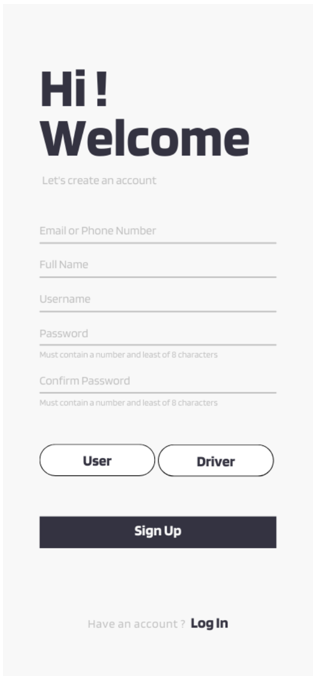


Figure 5.2 Registration Page

5.3.3 User Profile Page

Figure 5.3 shows a user profile interface designed to be versatile and accessible for all types of users, including drivers, administrators, suppliers, and customers. The interface is simple and user-friendly, focusing on providing essential profile management options.

Key components include:
1. Username: Displays the user's unique identifier (e.g., S001), helping to personalize the experience.
2. Full Name: Allows users to view and potentially edit their full name, ensuring their profile information is accurate.
3. Contact Number: Provides a field for users to update their contact information, which is crucial for communication and account security.
4. Settings: Offers access to additional settings, allowing users to customize their experience and preferences.
5. Log Out: A clear option to log out, ensuring users can securely end their session.

The design is straightforward and consistent across different user roles, ensuring that all users can easily manage their profiles and access necessary features. This approach is essential for maintaining a seamless and secure user experience across the platform.

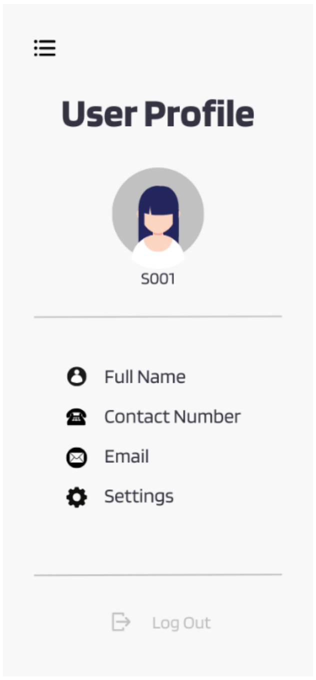


Figure 5.3 User Profile

5.4 Admin Screens

5.4.1 Home Page (Admin)

Figure 5.4 illustrates a home page interface designed for administrators, providing them with quick access to essential management functions. The interface is organized to help administrators efficiently oversee various aspects of the system.

Key features include:
1. Profile: Allows administrators to edit their latest information, ensuring their details are up-to-date.
2. Manage Driver: Provides options to view and manage driver details, facilitating the oversight of driver-related activities.

3. Refund Details: Enables administrators to view and handle refund requests, ensuring customer issues are addressed promptly.
4. Profile Management: Offers access to user data, allowing administrators to monitor and manage user profiles and activities.

The design emphasizes ease of navigation and quick access to critical functions, ensuring that administrators can manage their tasks efficiently. This streamlined approach is crucial for maintaining smooth operations and effective system management.

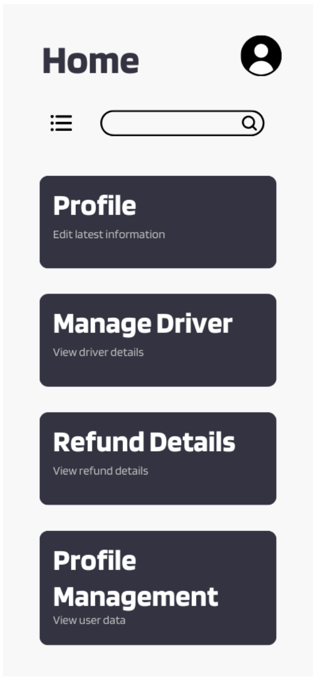


Figure 5.4 Home Page (Admin)

5.4.2 Manage Driver Authentication (Admin)

Figure 5.5 depicts a driver authentication management interface designed for administrators, allowing them to review and manage driver registration and authentication requests. The interface is structured to provide clear and actionable information for each driver.

Key components include:
1. Driver Information: Displays details about the driver, such as their name (e.g., Misha) and unique identifier (e.g., UCI23h).
2. Vehicle Details: Includes information about the driver's vehicle, such as the model (e.g., Perodua Myvi) and license plate number (e.g., CDF1208), ensuring the vehicle is verified.
3. Registration Date: Shows the date the driver registered (e.g., 7 Jan 2025), helping administrators track the timeline of the registration process.
4. Status: Indicates the status of the driver's authentication request (e.g., REJECTED, APPROVED), allowing administrators to take appropriate actions.

The design is straightforward and efficient, enabling administrators to easily review and manage driver authentication requests. This approach is essential for maintaining a secure and reliable system, ensuring that only verified drivers are approved for service.

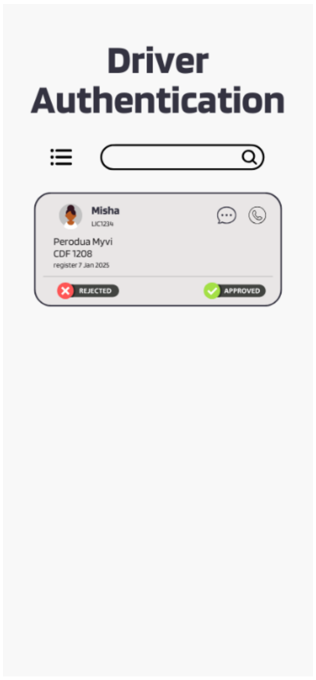


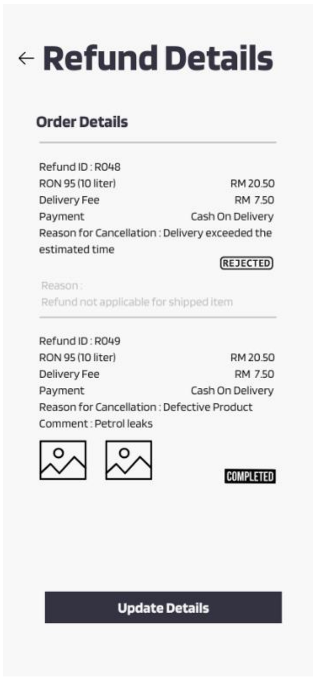
Figure 5.5 Manage Driver Authentication (Admin)

5.4.3 Manage Refund Request (Admin)

Figure 5.6 illustrates a refund management interface designed for administrators, allowing them to handle and process refund requests efficiently. The interface provides detailed information about each refund request to facilitate informed decision-making.

Key components include:
1. Order Details: Displays specific information about the order, such as the item (e.g., RON95, 10 liters) and delivery fee, ensuring administrators know which order the refund request pertains to.
2. Payment Information: Indicates the payment method used, which is important for processing the refund correctly.
3. Reason for Cancellation: Lists the reason provided by the customer for the refund request (e.g., delivery exceeded time, defective product), helping administrators understand the issue.
4. Customer Comments: Includes any additional comments from the customer (e.g., "Petrol leaks"), providing further context for the refund request.
5. Update Details: An option to update the refund details, allowing administrators to make necessary changes or notes as they process the request.
6. Refund ID: A unique identifier (e.g., R048) for each refund request, helping administrators track and manage refunds systematically.

The design is clear and comprehensive, ensuring that administrators have all the information they need to handle refund requests effectively. This approach is crucial for maintaining customer satisfaction and ensuring a smooth refund process.



Refund Details

Order Details

Refund ID : R048	
RON 95 (10 liter)	RM 20.50
Delivery Fee	RM 7.50
Payment	Cash On Delivery

Reason for Cancellation : Delivery exceeded the estimated time

REJECTED

Reason :
Refund not applicable for shipped item

Refund ID : R049	
RON 95 (10 liter)	RM 20.50
Delivery Fee	RM 7.50
Payment	Cash On Delivery

Reason for Cancellation : Defective Product
Comment : Petrol leaks

COMPLETED

Figure 5.6 Manage Refund Request (Admin)

5.4.4 Manage Customer and Driver Profile (Admin)

Figure 5.7 shows a profile management interface designed for administrators, allowing them to manage and monitor user profiles within the system. The interface is structured to provide a clear overview of user information and activity.

Key components include:
1. User Profiles: Displays a list of users (e.g., Amyra Natasha, Charlotte) along with their roles (e.g., Customer) and unique identifiers (e.g., 5001).
2. Activity Status: Indicates the last active date for each user (e.g., 31 Jan 2020, 1 December 2024, 3 Oct 2019), helping administrators track user engagement and activity.

The design is straightforward and efficient, enabling administrators to easily access and manage user information. This approach is essential for maintaining an organized and secure system, ensuring that administrators can effectively monitor and support users.

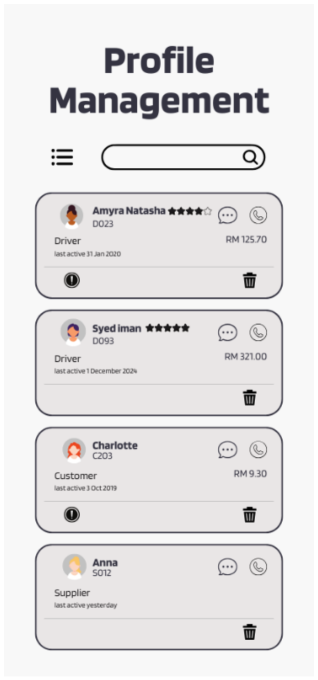


Figure 5.7 Profile Management (Customer, Driver) Page

5.5 Customer Screens

5.5.1 Home Page (Customer)

Figure 5.8 illustrates a home page interface designed for customers, providing them with easy access to essential features and services. The interface is user-friendly and organized to help customers navigate the platform efficiently.

Key features include:
1. Profile: Allows customers to edit their latest information, ensuring their details are up-to-date.
2. Make Order: Provides an option to purchase items, directing customers to the order placement section.
3. Tracking Order: Enables customers to track the status of their orders, offering transparency and real-time updates.
4. Refund: Allows customers to request refunds, providing a straightforward process for addressing any issues with their orders.

The design focuses on simplicity and ease of navigation, ensuring that customers can quickly access the features they need. This approach is crucial for enhancing user experience and ensuring customer satisfaction.

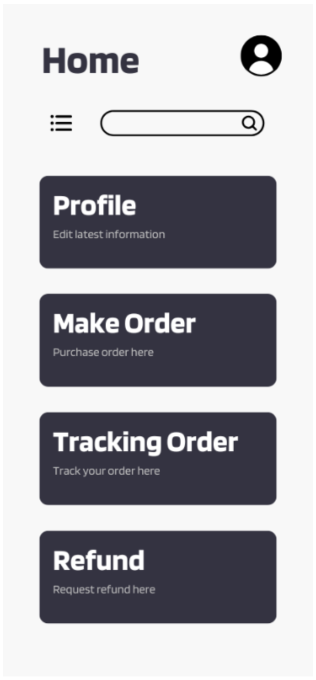


Figure 5.8 Home Page (Customer)

5.5.2 Make Order (Customer)

Figure 5.9 illustrates an order placement interface designed for customers, allowing them to easily select items and quantities for purchase. The interface is straightforward and user-friendly, focusing on simplifying the ordering process.

Key features include:
1. Item Selection: Customers can choose from a list of items (e.g., RON95, RON97, RON100, DieselB7, DieselB10) to add to their order.
2. Quantity Input: For each selected item, customers can specify the quantity in liters, ensuring they order the exact amount they need.
3. Checkout Button: A "Checkout" button is provided to finalize the order, allowing customers to proceed to payment and delivery details.

The design emphasizes ease of use and clarity, ensuring that customers can quickly and efficiently place their orders. This approach is crucial for providing a seamless and satisfying shopping experience.

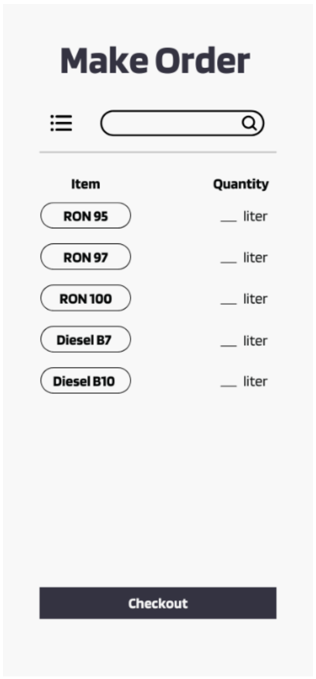


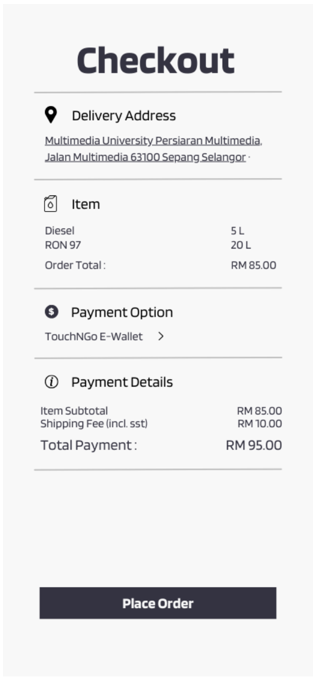
Figure 5.9 Make Order (Customer)

5.5.3 Checkout Page (Customer)

Figure 5.10 depicts a checkout interface designed for customers, allowing them to review and confirm their order details before finalizing the purchase. The interface is structured to ensure a smooth and transparent checkout process.

Key components include:
1. Delivery Address: The customer's delivery address (e.g., Multimedia University, Persiaran Multimedia, Jalan Multimedia, 63100 Sepang, Selangor) is displayed, ensuring the order is delivered to the correct location.
2. Payment Option: Customers can select their preferred payment method (e.g., TouchNGo E-Wallet), providing flexibility and convenience.
3. Payment Details: A breakdown of the costs is provided, including the item subtotal (e.g., RM85.00) and shipping fee (e.g., RM10.00), ensuring transparency in the total amount to be paid.
4. Place Order Button: A "Place Order" button allows customers to finalize their purchase, completing the checkout process.

The design emphasizes clarity and ease of use, ensuring that customers can easily review their order details and complete the purchase with confidence. This approach is essential for providing a seamless and satisfying shopping experience.


Checkout

Delivery Address
Multimedia University Persiaran Multimedia, Jalan Multimedia 63100 Sepang Selangor

Item

Diesel	5 L
RON 97	20 L
Order Total :	RM 85.00

Payment Option
TouchNGo E-Wallet >

Payment Details

Item Subtotal	RM 85.00
Shipping Fee (incl. sst)	RM 10.00
Total Payment :	RM 95.00

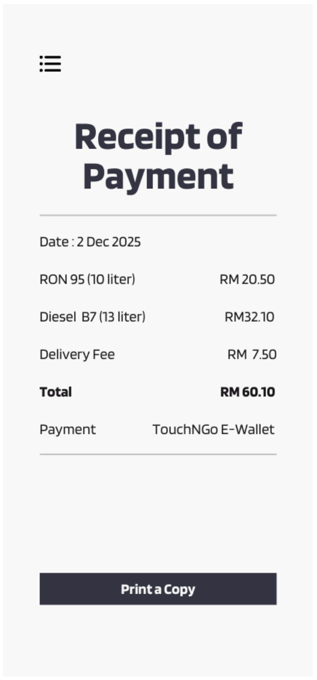
Figure 5.10 Checkout Page (Customer)

5.5.4 Receipt Page (Customer)

Figure 5.11 displays a receipt interface designed for customers, providing a detailed summary of their payment and order details. The interface is clear and concise, ensuring that customers can easily review their transaction information.

Key components include:
1. Payment Date: The date of the transaction (e.g., 2 Dec 2025) is displayed, helping customers keep track of their purchase history.
2. Order Details: Specific items and quantities (e.g., RON95, 10 liters; Diesel B7, 13 liters) are listed along with their respective prices, providing a clear breakdown of the order.
3. Delivery Fee: The delivery fee (e.g., RM 7.50) is included, ensuring transparency in the total cost.
4. Total Amount: The total amount paid (e.g., RM60.10) is prominently displayed, summarizing the overall cost.
5. Payment Method: The payment method used (e.g., TouchNGo E-Wallet) is indicated, which is useful for customers' records.
6. Print Option: An option to print a copy of the receipt is provided, allowing customers to keep a physical record if needed.

The design is straightforward and user-friendly, ensuring that customers can easily access and understand their receipt details. This approach is essential for maintaining transparency and trust in the transaction process.



Receipt of Payment

Date : 2 Dec 2025	
RON 95 (10 liter)	RM 20.50
Diesel B7 (13 liter)	RM32.10
Delivery Fee	RM 7.50
Total	**RM 60.10**
Payment	TouchNGo E-Wallet

Figure 5.11 Receipt Page (Customer)

5.5.5 Tracking Order Page (Customer)

Figure 5.12 illustrates a tracking order interface designed for customers, providing them with real-time updates and essential details about their delivery. The interface is user-friendly and focuses on keeping customers informed about the status of their order.

Key features include:
1. Order Status: A clear message (e.g., "Driver is on their way!") informs the customer about the current status of their delivery, enhancing transparency and trust.
2. Delivery Address: The destination address (e.g., Seksyen 10 Bandar Baru Bangi) is displayed, ensuring the customer knows where the order is being delivered.
3. Order Details: Specific items and quantities (e.g., RON95, 10 liters) are listed, providing a quick overview of what was ordered.
4. Delivery Fee: The fee for the delivery is shown, which is useful for the customer's reference.
5. Additional Options: Customers have the option to download their receipt online, rate the service, and re-order, enhancing the overall user experience.

The design emphasizes clarity and ease of use, ensuring that customers can easily track their orders and access relevant information. This approach is crucial for maintaining customer satisfaction and providing a seamless ordering experience.

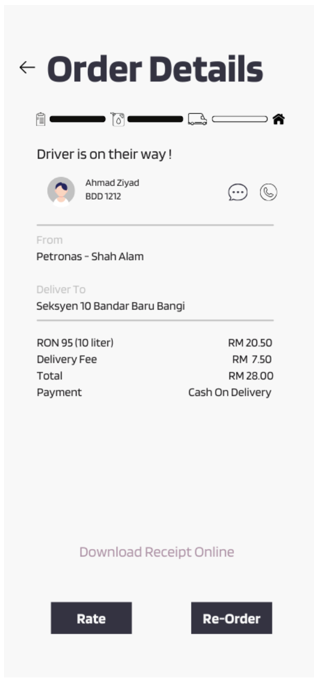


Figure 5.12 Tracking Order Page (Customer)

5.5.6 Request Refund Page (Customer)

Figure 5.13 shows a refund request interface designed for customers, allowing them to easily request a refund for their orders. The interface is structured to gather all necessary information to process the refund efficiently.

Key components include:
1. Order Details: Displays specific information about the order, such as the item (e.g., RON95, 10 liters) and delivery fee, ensuring the customer knows which order they are requesting a refund for.
2. Payment Information: Indicates the payment method used (e.g., Cash On Delivery), which is important for processing the refund correctly.
3. Reason for Refund: Provides a list of common reasons for requesting a refund (e.g., delayed delivery, incorrect item, defective product) and an option for other reasons, helping customers specify the issue accurately.
4. Details of Refund: A text field allows customers to provide additional comments or details about their refund request, ensuring that all relevant information is captured.
5. Send Request Button: A button to submit the refund request, completing the process.

The design is user-friendly and ensures that customers can easily provide all necessary information to facilitate a smooth refund process. This approach is essential for maintaining customer satisfaction and trust in the service.

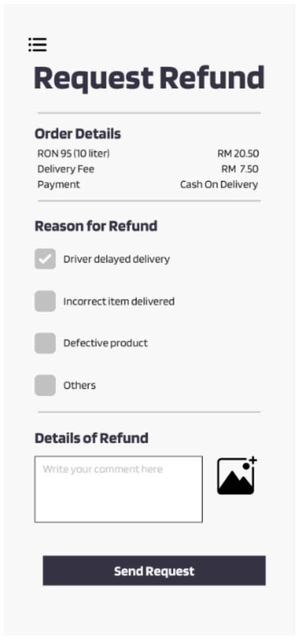


Figure 5.13 Request Refund Page (Customer)

5.6 Driver Screens

5.6.1 Home Page (Driver)

Figure 5.14 illustrates a home page interface designed for drivers, providing them with quick access to essential functions and information needed for their delivery tasks. The interface is user-friendly and focuses on key tasks that drivers need to perform regularly.

Key features include:
1. Profile: Allows drivers to edit their latest information, ensuring that their details are up-to-date.
2. Delivery: Provides access to available orders that need to be delivered, helping drivers manage their tasks efficiently.
3. Driver Details: Enables drivers to update their authentication information, ensuring that their credentials and vehicle details are current and accurate.

The design emphasizes ease of navigation and quick access to critical functions, ensuring that drivers can manage their tasks efficiently. This streamlined approach is crucial for maintaining smooth operations and timely order processing in a delivery context.

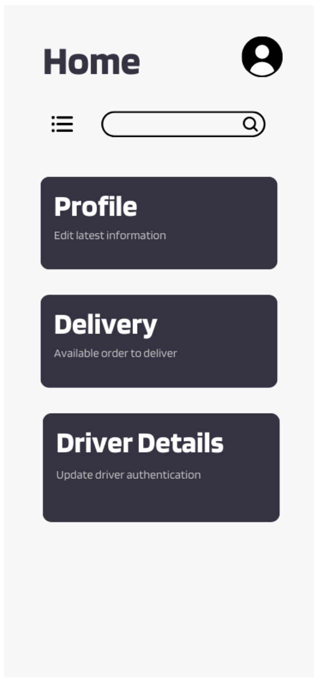


Figure 5.14 Home Page (Driver)

5.6.2 Driver Authentication Page (Driver)

Figure 5.15 depicts a driver authentication interface designed to verify and register driver information securely. The interface is straightforward, ensuring that drivers can easily input the required details for authentication.

Key elements include:
1. Driver Details: Drivers are prompted to enter necessary personal and professional details to verify their identity.
2. License Number: A field is provided for drivers to input their license number, which is crucial for validating their driving credentials.
3. Vehicle Number Plate: Drivers must enter their vehicle's number plate, ensuring that the vehicle used for deliveries is registered and recognized.
4. Register Button: A "Register" button is available for drivers to submit their information, completing the authentication process.

The design emphasizes simplicity and security, ensuring that all necessary information is collected efficiently. This approach is essential for maintaining a reliable and secure system for driver authentication, which is critical for the safety and integrity of delivery operations.

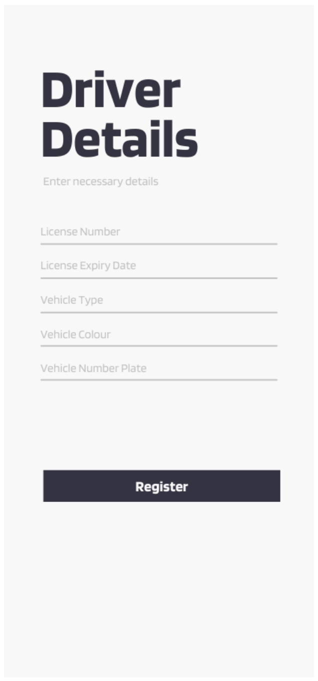


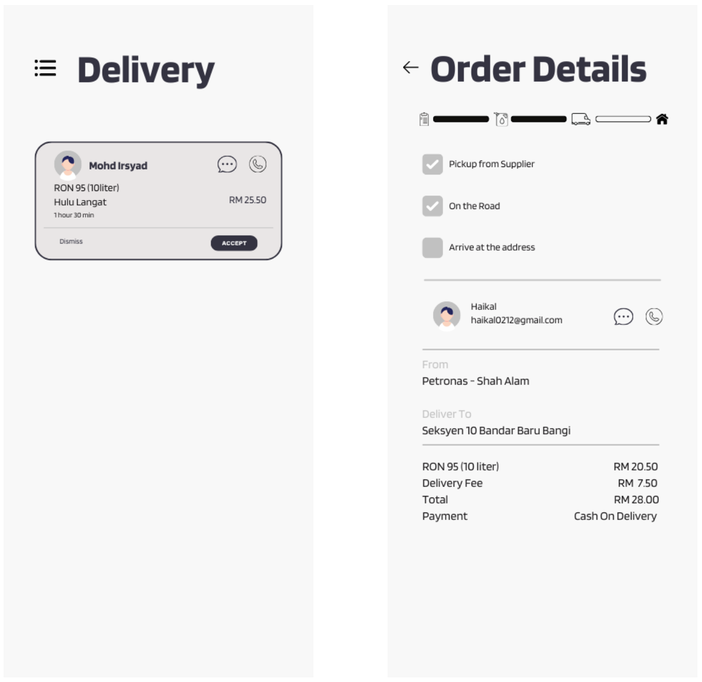
Figure 5.15 Driver Authentication Page (Driver)

5.6.3 Deliver Order Page (Driver)

Figure 5.16 illustrates a delivery order interface designed specifically for drivers, providing them with essential information to complete their delivery tasks efficiently. The interface is streamlined to ensure drivers can quickly access and understand the details they need.

Key components include:
1. Customer Information: The driver is provided with the customer's name (e.g., Mohd Irsyad) to personalize the delivery process.
2. Order Details: Specific items and quantities (e.g., RON 95, 10 liters) are listed, ensuring the driver knows exactly what to deliver.
3. Delivery Location: The destination (e.g., Hulu Langat) is clearly indicated, helping the driver navigate to the correct address.
4. Delivery Fee: The fee for the delivery (e.g., RM25.50) is displayed, which is useful for transactions and record-keeping.

The design focuses on clarity and ease of use, enabling drivers to quickly grasp the necessary information and complete their deliveries efficiently. This approach is crucial for maintaining timely and accurate delivery services, enhancing overall customer satisfaction.



Mohd Irsyad
RON 95 (10liter)
Hulu Langat RM 25.50
1 hour 30 min

✓ Pickup from Supplier
✓ On the Road
Arrive at the address

Haikal
haikal0212@gmail.com

From
Petronas - Shah Alam

Deliver To
Seksyen 10 Bandar Baru Bangi

RON 95 (10 liter)	RM 20.50
Delivery Fee	RM 7.50
Total	RM 28.00
Payment	Cash On Delivery

Figure 5.16 Deliver Order Page (Driver)

5.7 Supplier Screens

5.7.1 Home Page (Supplier)

Figure 5.17 shows a home page interface designed for suppliers, providing quick access to essential functions for managing their operations. The interface is simple and user-friendly, focusing on key tasks that suppliers need to perform regularly.

Key features include:
1. Profile: Allows suppliers to edit their latest information, ensuring that their details are up-to-date.
2. Inventory: Provides options to update stock levels and prices, helping suppliers keep their inventory data accurate and current.
3. Prepare Order: Enables suppliers to pack items based on customer orders, facilitating efficient order fulfillment.

The design emphasizes ease of navigation and quick access to critical functions, ensuring that suppliers can manage their tasks efficiently. This streamlined approach is crucial for maintaining smooth operations and timely order processing in a supply chain context.

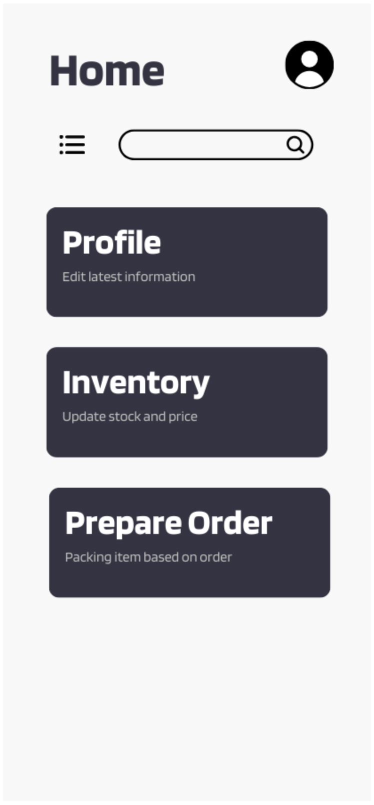


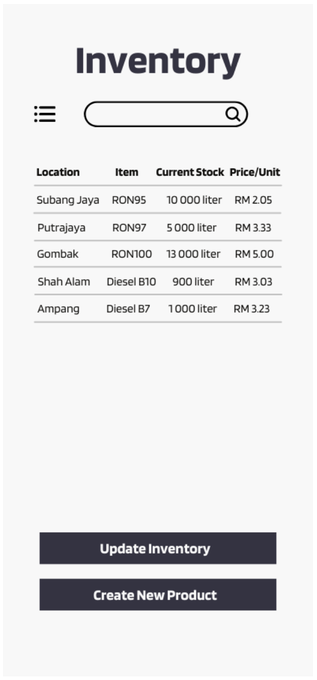
Figure 5.17 Home Page (Supplier)

5.7.2 Update Inventory and Price Page (Supplier)

Figure 5.18 illustrates an inventory management interface designed to help users track and update stock levels and product details across different locations. The interface is organized to display essential information clearly and concisely.

Key elements include:
1. Location: Each entry specifies the location of the inventory, such as Subang Jaya or Putrajaya, helping users identify where products are stored.
2. Item and Stock Levels: The interface lists items (e.g., RON95, RON97) along with their current stock levels (e.g., 10,000 liters, 5,000 liters), providing a quick overview of available inventory.
3. Price per Unit: The price per unit for each item is displayed (e.g., RM2.05, RM3.33), which is useful for financial tracking and sales planning.
4. Actions: Options like "Update Inventory" and "Create New Product" are provided, allowing supplier to manage and expand their inventory efficiently.

The design is straightforward and user-friendly, ensuring that supplier can easily access and manage inventory information. This approach is essential for maintaining accurate stock levels and supporting operational efficiency in various business contexts.



Location	Item	Current Stock	Price/Unit
Subang Jaya	RON95	10 000 liter	RM 2.05
Putrajaya	RON97	5 000 liter	RM 3.33
Gombak	RON100	13 000 liter	RM 5.00
Shah Alam	Diesel B10	900 liter	RM 3.03
Ampang	Diesel B7	1 000 liter	RM 3.23

Figure 5.18 Update Inventory and Price Page (Supplier)

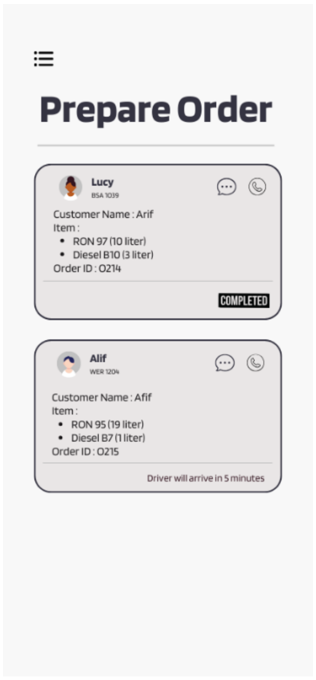
5.7.3 Prepare Order Page (Supplier)

Figure 5.19 illustrates a prepare order interface designed for suppliers, providing them with essential details to fulfill customer orders efficiently. The interface is structured to ensure clarity and ease of use.

Key components include:

1. Customer Information: Displays the customer's name (e.g., Arif) to help suppliers identify and organize orders.
2. Order Details: Lists specific items and quantities (e.g., RON 97, 10 liters; Diesel B10, 3 liters) to ensure accurate preparation.
3. Order ID: A unique identifier (e.g., O214) for each order, aiding in tracking and reference.
4. Contact Information: Provides a contact option for any necessary communication regarding the order.
5. Delivery Status: For ongoing orders, real-time updates (e.g., "Driver will arrive in 5 minutes") are provided to keep the supplier informed about the delivery progress.

The design emphasizes clarity and efficiency, ensuring that suppliers can quickly access and act on the information they need. This approach is crucial for maintaining smooth operations and timely order fulfillment in a supply chain context.



Prepare Order

Lucy
BSA 1039

Customer Name : Arif
Item :
- RON 97 (10 liter)
- Diesel B10 (3 liter)

Order ID : O214

COMPLETED

Alif
WER 1204

Customer Name : Afif
Item :
- RON 95 (19 liter)
- Diesel B7 (1 liter)

Order ID : O215

Driver will arrive in 5 minutes

Figure 5.19 Prepare Order (Supplier)

5.8 Main Components

All Users: Can log in to the system.
Customers: Can register, manage profiles, place and pay for orders, and track their orders.
Drivers: Must register and authenticate themselves and can track orders assigned to them.
Admins: Manage driver registrations, refund requests, and customer profiles.
Suppliers: Responsible for order preparation and updating pricing and stock inventory.

Actor	Components
All User	Login
Customer	Registration
	Profile management
	Order placement and payment
	Track order
Driver	Registration and authentication
	Track Order
Admin	Manage driver registration
	Manage refund request
	Manage customer profile
Supplier	Prepare Order
	Update pricing data and stock inventory

Figure 5.8 :Main Component Table

5.8.1 Login Component

All user will log into the system using their credentials.Once system verifies the details they can enter their respective main page. The system validates the input and updates the database accordingly. If any error occurs during this process, the system will notify the user.

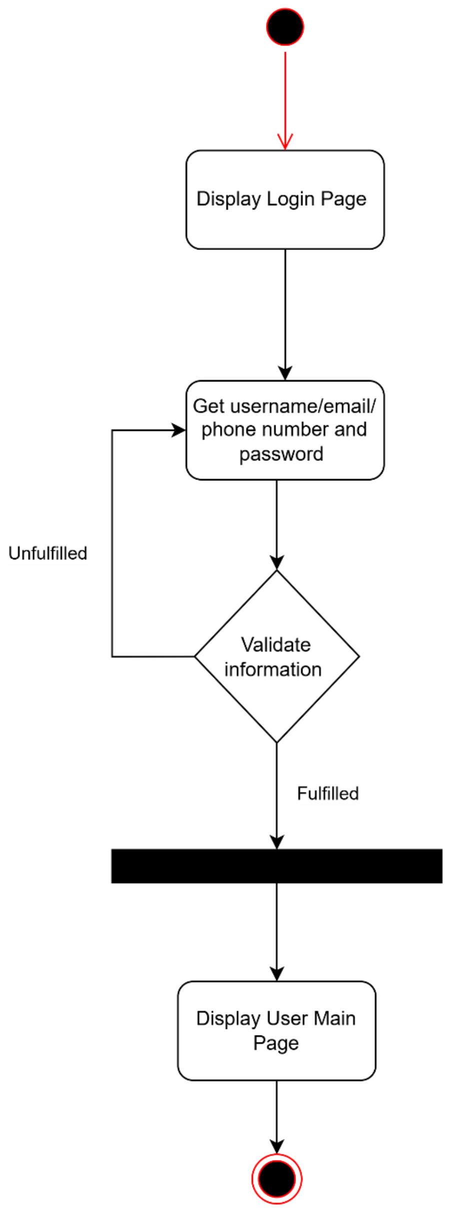


Figure 5.8.1.1 : Login Component Diagram

```
BEGIN

DISPLAY "Login Page"

WHILE true DO
   GET user_input (username/email/phone, password)

   IF user_input is not empty THEN
      CALL validate_information(user_input)

      IF validation is successful THEN
         DISPLAY "User Main Page"
         BREAK
      ELSE
         DISPLAY "Invalid credentials. Please try again."
   ELSE
      DISPLAY "Please enter all required information."
   ENDIF
ENDWHILE

END
```

Figure 5.8.1.2 : Login Component Pseudocode

5.8.2 Registration Component

Customers and drivers enter required details to register in the system to open an account. The system then validates the information and update the database. If succesful an account will be created otherwise the user is asked to correct errors.

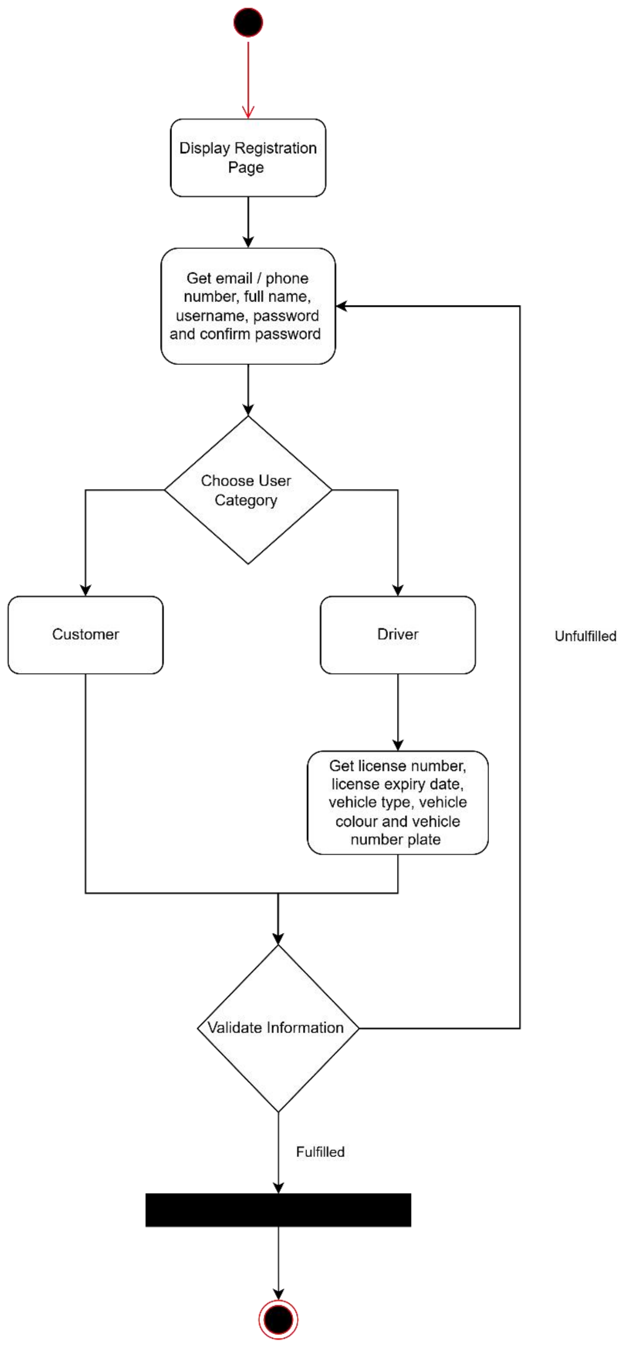

Display Registration Page
Get email / phone number, full name, username, password and confirm password
Choose User Category
Customer
Driver
Unfulfilled
Get license number, license expiry date, vehicle type, vehicle colour and vehicle number plate
Validate Information
Fulfilled

Figure 5.8.2.1 : Registration Component Diagram

```
BEGIN
  Display Registration Page

  Get user input:
    email
    phone number
    full name
    username
    password
    confirm password

  Display "Choose User Category"

  IF User selects "Customer" THEN
    Proceed to validation

  ELSE IF User selects "Driver" THEN
    Get driver details:
      license number
      license expiry date
      vehicle type
      vehicle color
      vehicle number plate
    Proceed to validation

  ELSE
    Display "Unfulfilled Registration"
    END PROCESS

  Validate Information

  IF Validation is successful THEN
    Registration is Fulfilled
  ELSE
    Display "Invalid Information. Please try again."
    Restart process

END
```

Figure 5.8.2.2 : Registration Component Pseudocode

5.8.3 Order Placement Component

The customers selects item then the system calculates the total price and customer then confirms the order and selects a payment method. The order is placed.

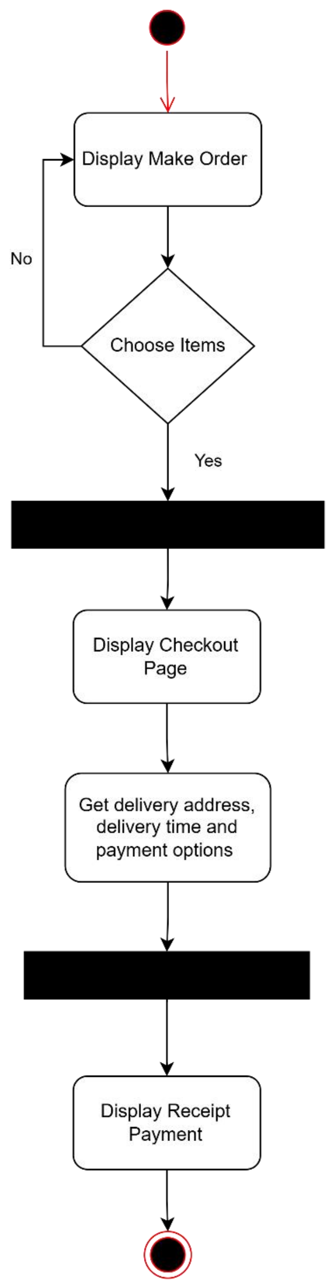


Figure 5.8.3.1 : Order Placement Component Diagram

```
BEGIN
   Display "Make Order" Page

   Display "Choose Items?"

   IF User selects "No" THEN
      END PROCESS

   ELSE IF User selects "Yes" THEN
      Proceed to Checkout
```

```
Display Checkout Page

Get user input:
    Delivery address
    Delivery time
    Payment options

Confirm Order

Display Receipt Payment

END PROCESS
```

Figure 5.8.3.2 : Order Placement Component Pseudocode

5.8.4 Track Order Component

Once order placed the system assign driver. The system updates the order status . Customer can track their order's progress in real time.

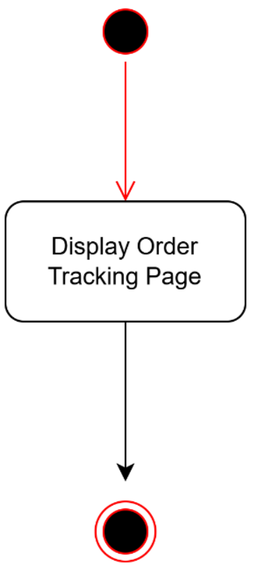


Figure 5.8.4.1 : Track Order Component Diagram

```
BEGIN
    Display Order Tracking Page
END
```

Figure 5.8.4.2 : Track Order Component Pseudocode

5.8.5 Refund Request Component

The process starts with a customer submitting a refund request. The request is reviewed, and if approved, the refund is processed. If rejected, the customer is notified of the denial.

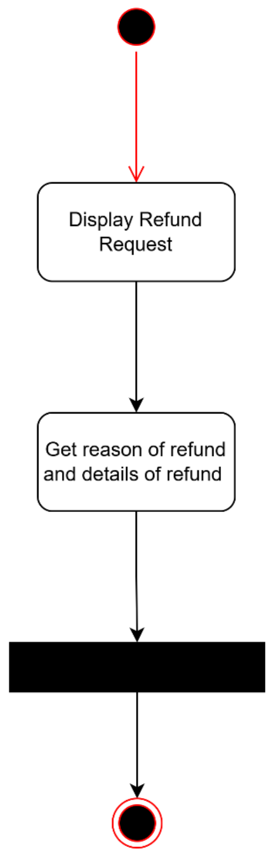


Figure 5.8.5.1 : Refund Request Component Diagram

```
BEGIN
   Display Refund Request Page
   Get reason and details of refund
   Process refund request
END
```

Figure 5.8.5.2 : Refund Request Component Pseudocode

5.8.6 Manage Driver Authentication Component

Drivers enter their login details. The system verifies credentials and role. If authentication is successful, the driver is granted access to manage deliveries

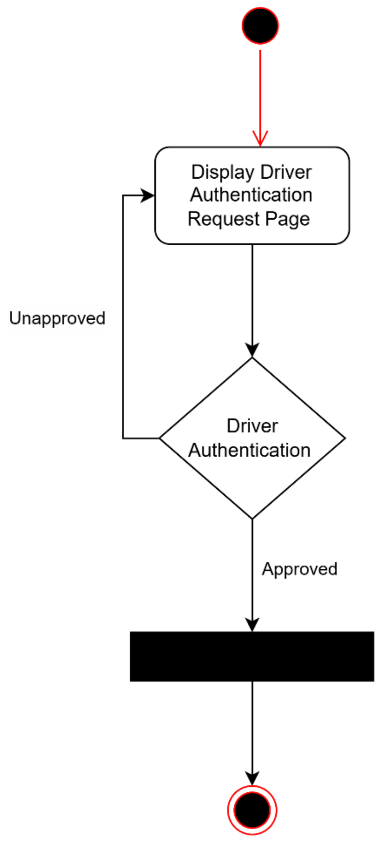


Figure 5.8.6.1: Manage Driver Authentication Component Diagram

```
BEGIN
   Display Driver Authentication Request Page
   IF Driver Authentication Approved THEN
      Process Authentication
   ELSE
      Reject Authentication
   ENDIF
END
```

Figure 5.8.6.2: Manage Driver Authentication Component Pseudocode

5.8.7 Manage Customer Profile Component

The admin can manage customers and driver profile and terminated the profile if it’s violating the system rules.

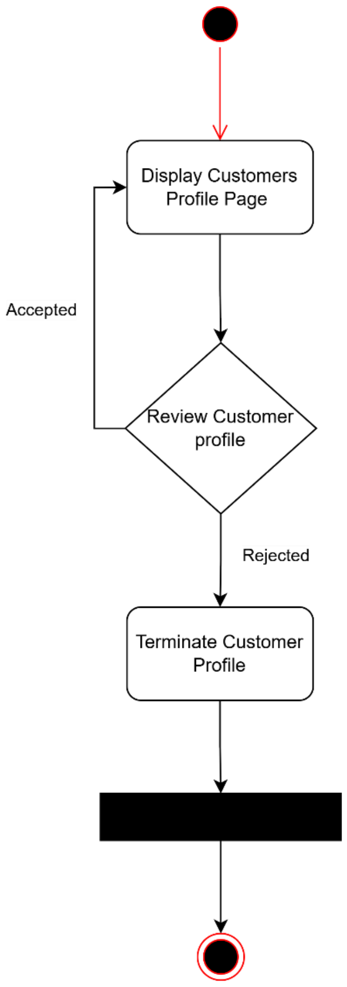


Figure 5.8.7.1: Manage Customer Profile Component Diagram

```
BEGIN
   Display Customer Profile Page
   Review Customer Profile

   IF Profile is Accepted THEN
      CONTINUE
   ELSE
      Terminate Customer Profile
   ENDIF

END
```

Figure 5.8.7.2: Manage Customer Profile Component Pseudocode

5.8.8 Manage Refund Request Component

The admin handle customer's refund request. The customer submits a refund request, which is then reviewed. If the request is approved, the refund is processed. Otherwise, the request is denied and customer can’t get refunded

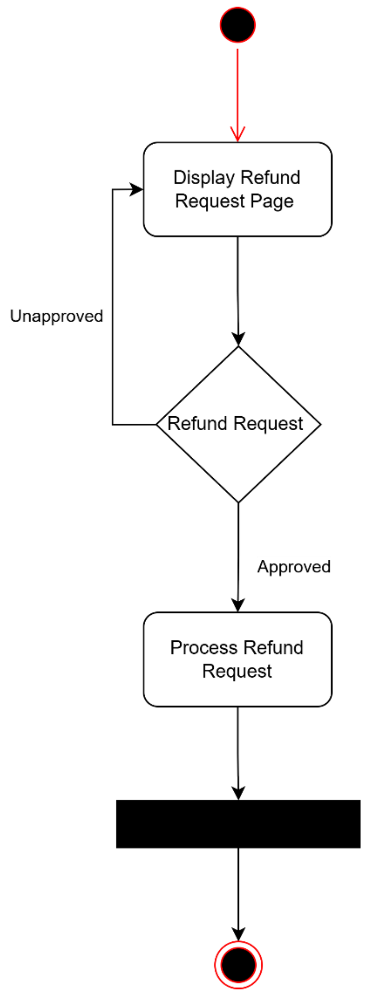


Figure 5.8.8.1: Manage Refund Request Component Diagram

```
BEGIN
   Display Refund Request Page
   INPUT Refund Request

   IF Refund Request is Approved THEN
      Process Refund Request
   ELSE
      RETURN to Refund Request Page
   ENDIF

END
```

Figure 5.8.8.2: Manage Refund Request Component Pseudocode

5.8.9 Update Stock and Pricing Data Component

The system allows supplier to access the stock and pricing database, where they can modify or update inventory and pricing information as necessary.

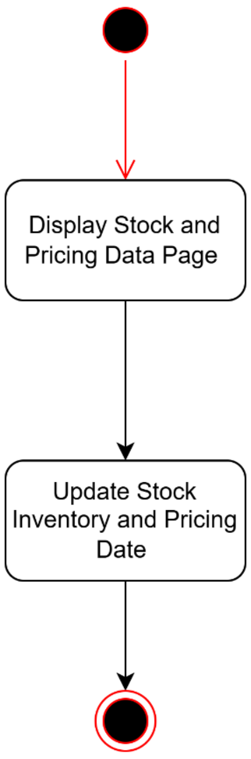

Display Stock and Pricing Data Page
Update Stock Inventory and Pricing Date

Figure 5.8.9.1: Update Stock and Pricing Data Component Diagram

```
BEGIN
   Display Stock and Pricing Data Page
   Update Stock Inventory and Pricing Data
END
```

Figure 5.8.9.2: Update Stock and Pricing Data Component Pseudocode

5.8.10 Prepare Order Component

The supplier receives details and prepare order according to it.

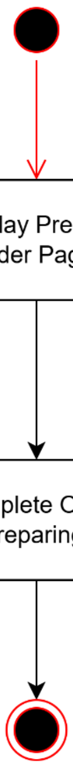


Figure 5.8.10.1: Prepare Order Component Diagram

```
BEGIN
   Display Prepare Order Page
   Complete Order Preparing
END
```

Figure 5.8.10.2: Prepare Order Component Pseudocode

5.9 Deployment Diagram

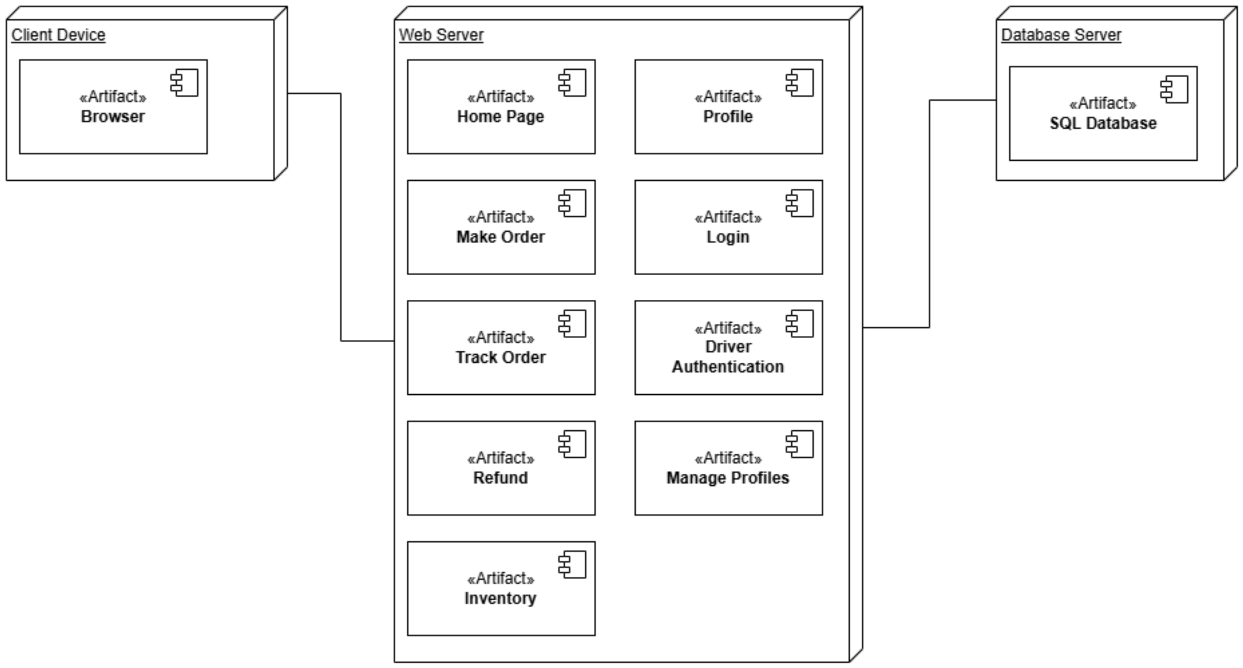


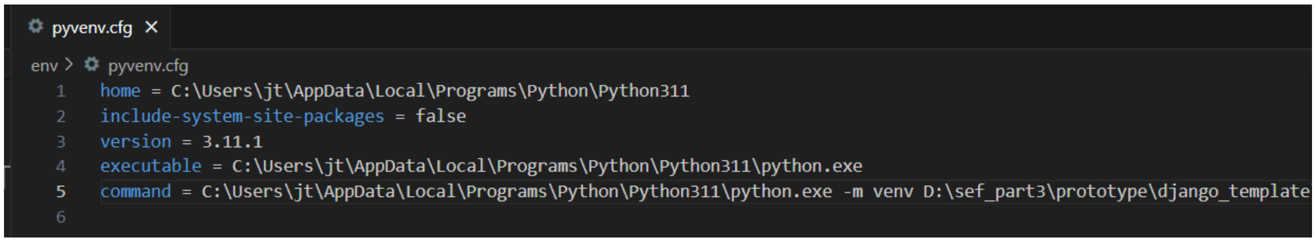
Figure 4.9 : Deploment Diagram

For deployment diagram, HTML is chosen as the front-end interface, allowing users to interact with the system through their mobile devices using web browsers for convenience. The application leverages PHP as the server-side scripting language to process requests and establish a connection to the back-end SQL database, ensuring easier data management and retrieval.

6 Implementation

6.1 Development Environment

The online petrol delivery system is developed using Visual Code Studio, HTML and Python Django Framework. Visual Code Studio is used as the source code edtor for this project while HTML is the frontend language and Python Django is used as the backend language. The pyvenv.cfg is used to setup the enviroment for the project.


pyvenv.cfg

env > pyvenv.cfg

```
home = C:\Users\jt\AppData\Local\Programs\Python\Python311
include-system-site-packages = false
version = 3.11.1
executable = C:\Users\jt\AppData\Local\Programs\Python\Python311\python.exe
command = C:\Users\jt\AppData\Local\Programs\Python\Python311\python.exe -m venv D:\sef_part3\prototype\django_template
```

6.2 Software Integration

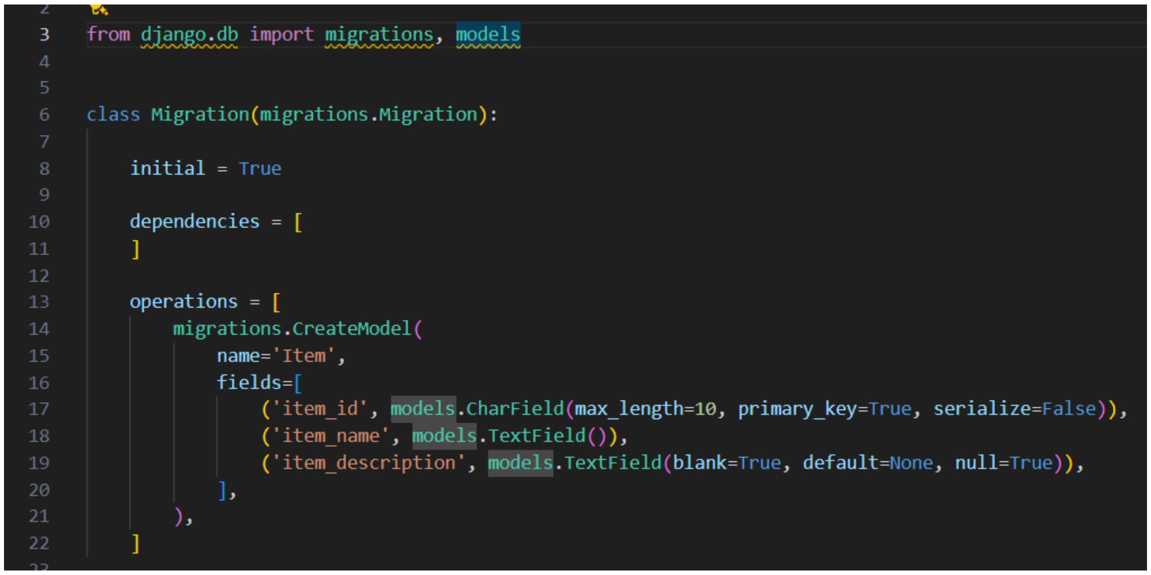
<TO DO: Describe a strategy to integrate all the subsystems here with relevant images.>

File	Description

6.3 Database

6.3.1 Database items

This is a Django database migration file. It is automatically generated when run the makemigrations command in Django. It defines the changes that should be applied to the database schema.

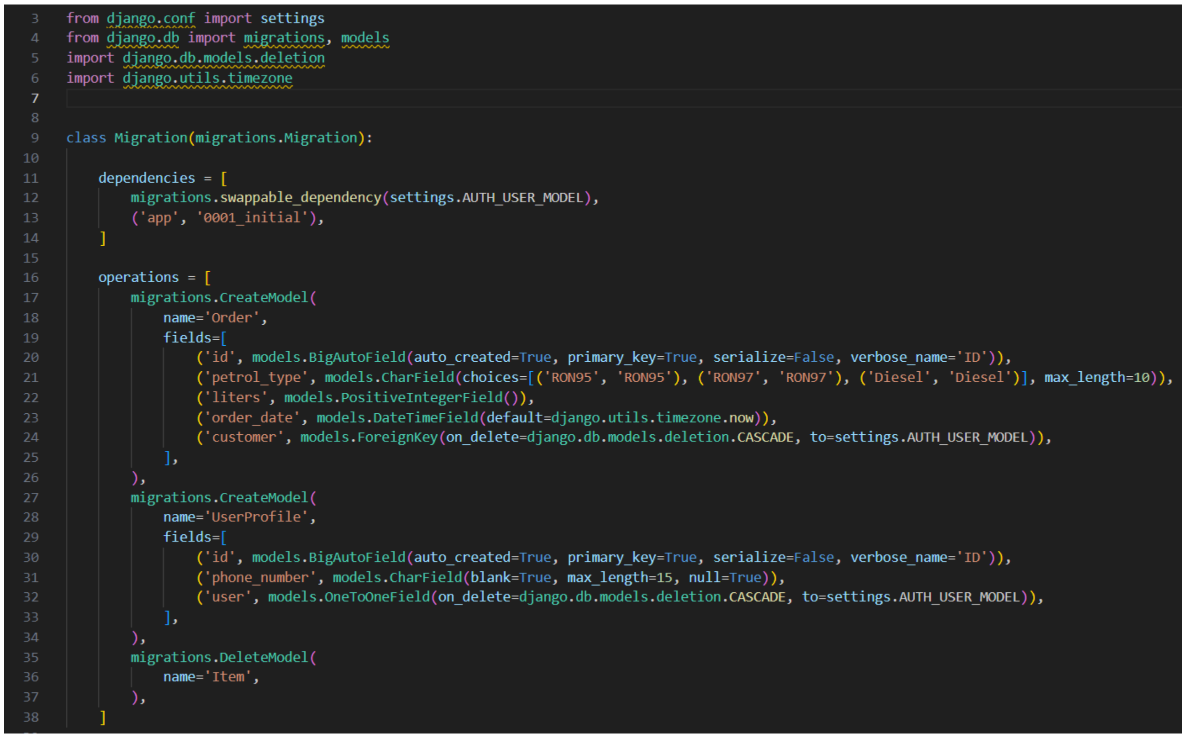


```
from django.db import migrations, models


class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='Item',
            fields=[
                ('item_id', models.CharField(max_length=10, primary_key=True, serialize=False)),
                ('item_name', models.TextField()),
                ('item_description', models.TextField(blank=True, default=None, null=True)),
            ],
        ),
    ]
```

6.3.2 Create Order and User Profile

This Django migration file defines **database schema changes**, specifically creating two new models (Order and UserProfile) and deleting the old Item model. This is what it does:

1. Creates an Order table (storing petrol purchase orders).
2. Creates a UserProfile table (storing extra user details).
3. Deletes the old Item table (likely no longer needed).

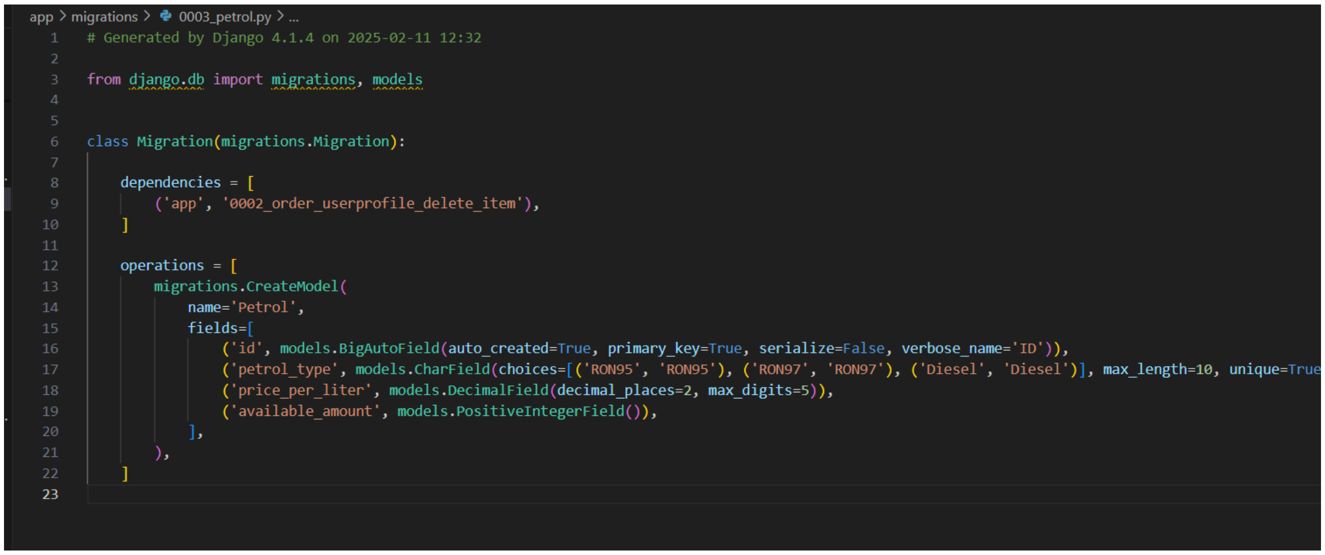


```python
from django.conf import settings
from django.db import migrations, models
import django.db.models.deletion
import django.utils.timezone


class Migration(migrations.Migration):

    dependencies = [
        migrations.swappable_dependency(settings.AUTH_USER_MODEL),
        ('app', '0001_initial'),
    ]

    operations = [
        migrations.CreateModel(
            name='Order',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('petrol_type', models.CharField(choices=[('RON95', 'RON95'), ('RON97', 'RON97'), ('Diesel', 'Diesel')], max_length=10)),
                ('liters', models.PositiveIntegerField()),
                ('order_date', models.DateTimeField(default=django.utils.timezone.now)),
                ('customer', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to=settings.AUTH_USER_MODEL)),
            ],
        ),
        migrations.CreateModel(
            name='UserProfile',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('phone_number', models.CharField(blank=True, max_length=15, null=True)),
                ('user', models.OneToOneField(on_delete=django.db.models.deletion.CASCADE, to=settings.AUTH_USER_MODEL)),
            ],
        ),
        migrations.DeleteModel(
            name='Item',
        ),
    ]
```

6.3.3 Petrol

This migration creates a new Petrol model, which stores petrol-related information. This is what it does :

1. Creates a Petrol table to store fuel types, pricing, and stock levels.
2. Ensures each petrol type is unique (no duplicate entries for RON95, RON97, or Diesel).
3. Keeps track of fuel availability in liters.

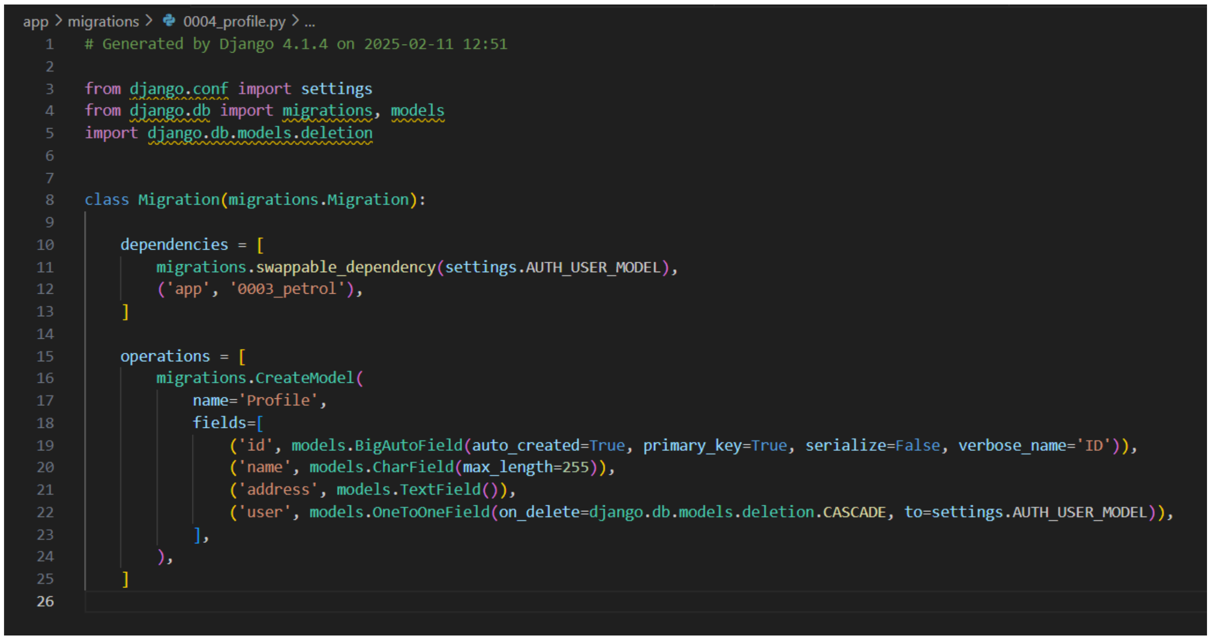


```python
# Generated by Django 4.1.4 on 2025-02-11 12:32

from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('app', '0002_order_userprofile_delete_item'),
    ]

    operations = [
        migrations.CreateModel(
            name='Petrol',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('petrol_type', models.CharField(choices=[('RON95', 'RON95'), ('RON97', 'RON97'), ('Diesel', 'Diesel')], max_length=10, unique=True
                ('price_per_liter', models.DecimalField(decimal_places=2, max_digits=5)),
                ('available_amount', models.PositiveIntegerField()),
            ],
        ),
    ]
```

6.3.4 Profile

This migration creates a new Profile model, which extends user information with additional details like name and address. What it does.

1. Creates a Profile table to store additional user details.
2. Links each Profile to a unique User (One-to-One relationship).
3. Ensures profile data is deleted if the user is removed



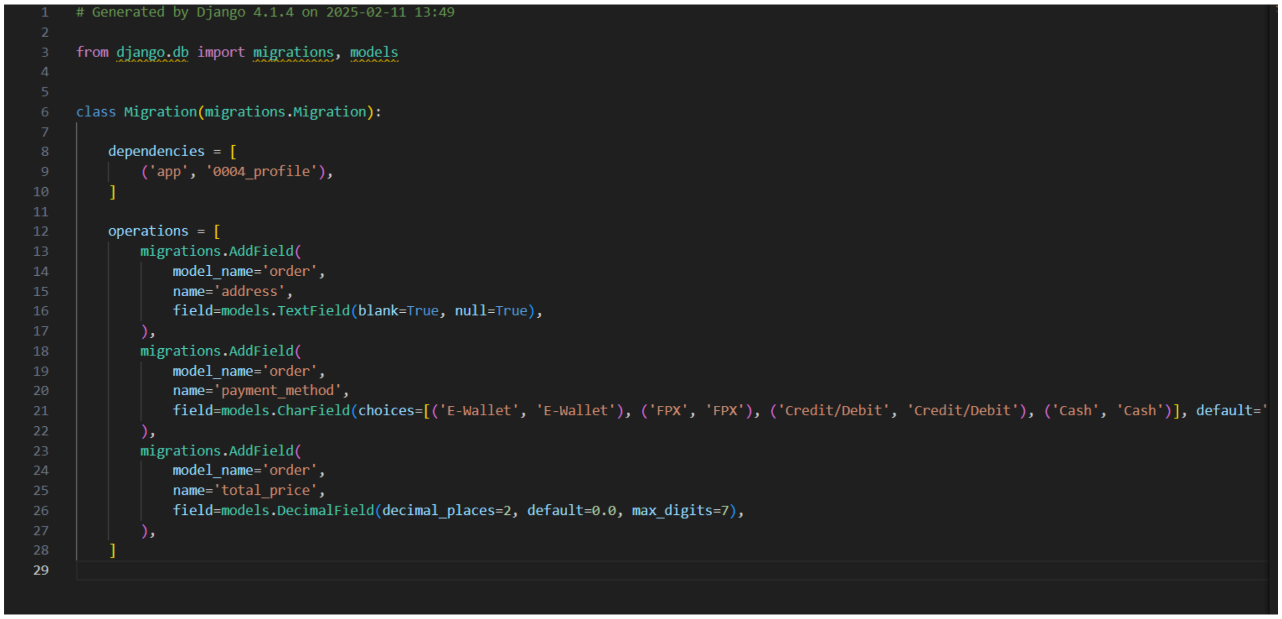
app > migrations > 0004_profile.py > ...

```python
# Generated by Django 4.1.4 on 2025-02-11 12:51

from django.conf import settings
from django.db import migrations, models
import django.db.models.deletion


class Migration(migrations.Migration):

    dependencies = [
        migrations.swappable_dependency(settings.AUTH_USER_MODEL),
        ('app', '0003_petrol'),
    ]

    operations = [
        migrations.CreateModel(
            name='Profile',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('name', models.CharField(max_length=255)),
                ('address', models.TextField()),
                ('user', models.OneToOneField(on_delete=django.db.models.deletion.CASCADE, to=settings.AUTH_USER_MODEL)),
            ],
        ),
    ]
```

6.3.5 Order address, payment method and total price

This migration adds three new fields (address, payment_method, and total_price) to the Order model, expanding its functionality.What it does :

1. Allows storing an address (optional) for each order.
2. Adds a payment method field with predefined choices.
3. Stores the total price of the order as a decimal value.

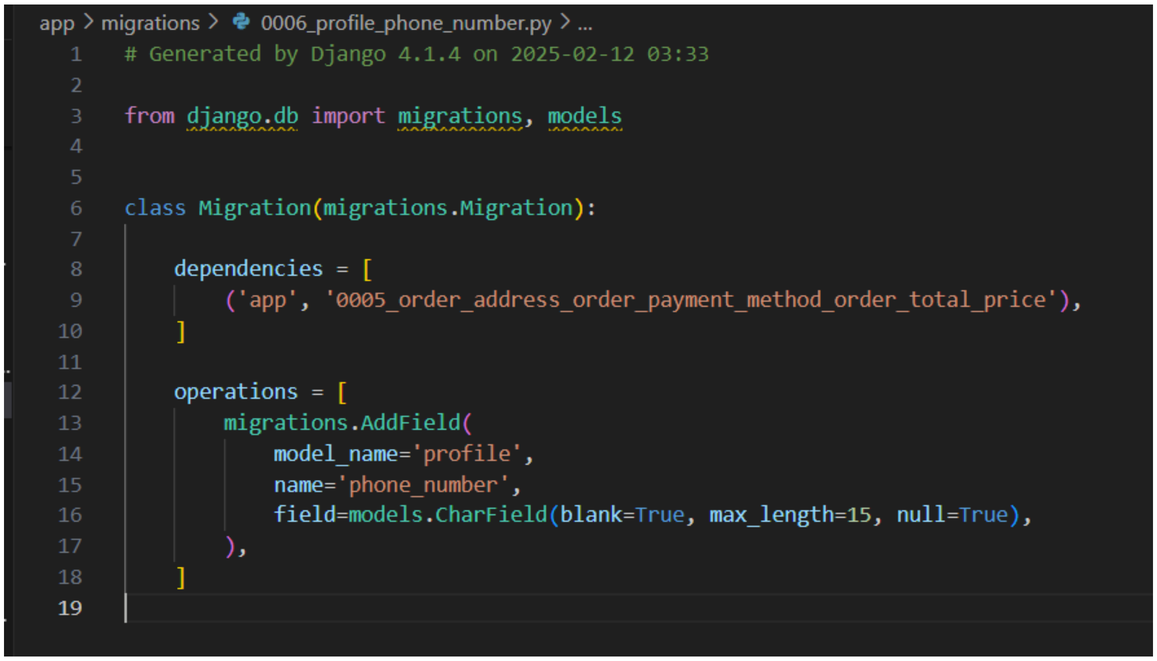


```python
# Generated by Django 4.1.4 on 2025-02-11 13:49

from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('app', '0004_profile'),
    ]

    operations = [
        migrations.AddField(
            model_name='order',
            name='address',
            field=models.TextField(blank=True, null=True),
        ),
        migrations.AddField(
            model_name='order',
            name='payment_method',
            field=models.CharField(choices=[('E-Wallet', 'E-Wallet'), ('FPX', 'FPX'), ('Credit/Debit', 'Credit/Debit'), ('Cash', 'Cash')], default='
        ),
        migrations.AddField(
            model_name='order',
            name='total_price',
            field=models.DecimalField(decimal_places=2, default=0.0, max_digits=7),
        ),
    ]

```

6.3.6 Profile phone number

This migration adds a phone_number field to the Profile model, allowing users to store their contact number.

1. Adds a phone_number field to the Profile model.
2. Makes it optional, so existing users won’t be affected.
3. Allows storing phone numbers up to 15 characters.



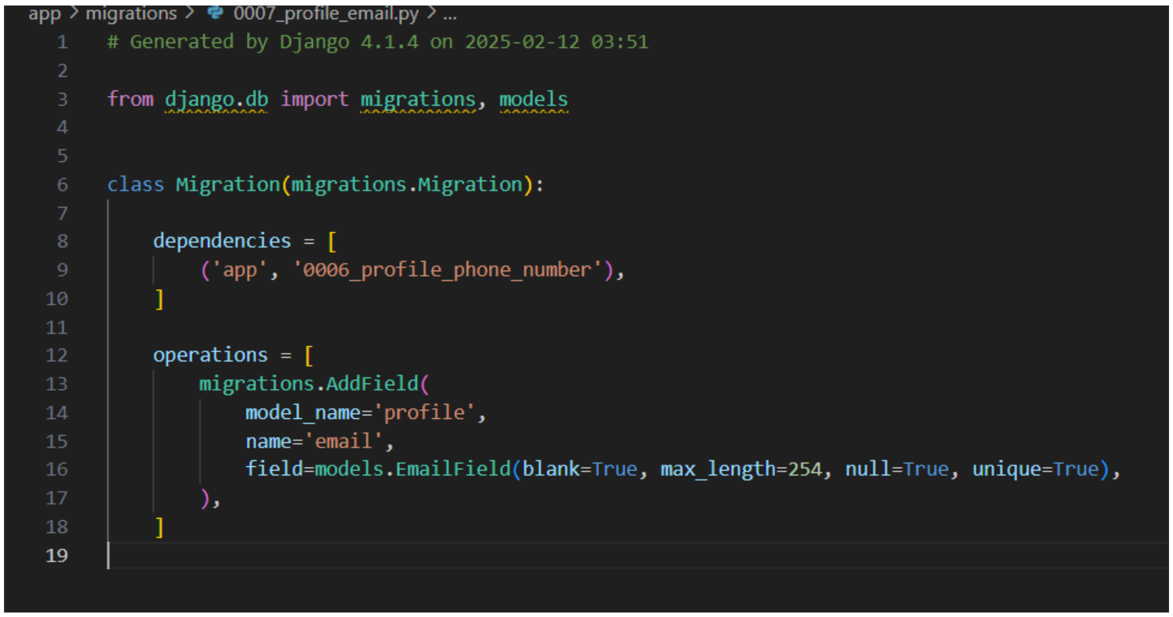
app > migrations > 0006_profile_phone_number.py > ...

```python
# Generated by Django 4.1.4 on 2025-02-12 03:33

from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('app', '0005_order_address_order_payment_method_order_total_price'),
    ]

    operations = [
        migrations.AddField(
            model_name='profile',
            name='phone_number',
            field=models.CharField(blank=True, max_length=15, null=True),
        ),
    ]

```

6.3.7 Profile email

This migration adds an email field to the Profile model, allowing users to store a unique email address.

1. Adds an email field to the Profile model.
2. Enforces uniqueness, preventing duplicate email entries.
3. Allows users to store an email address optionally.



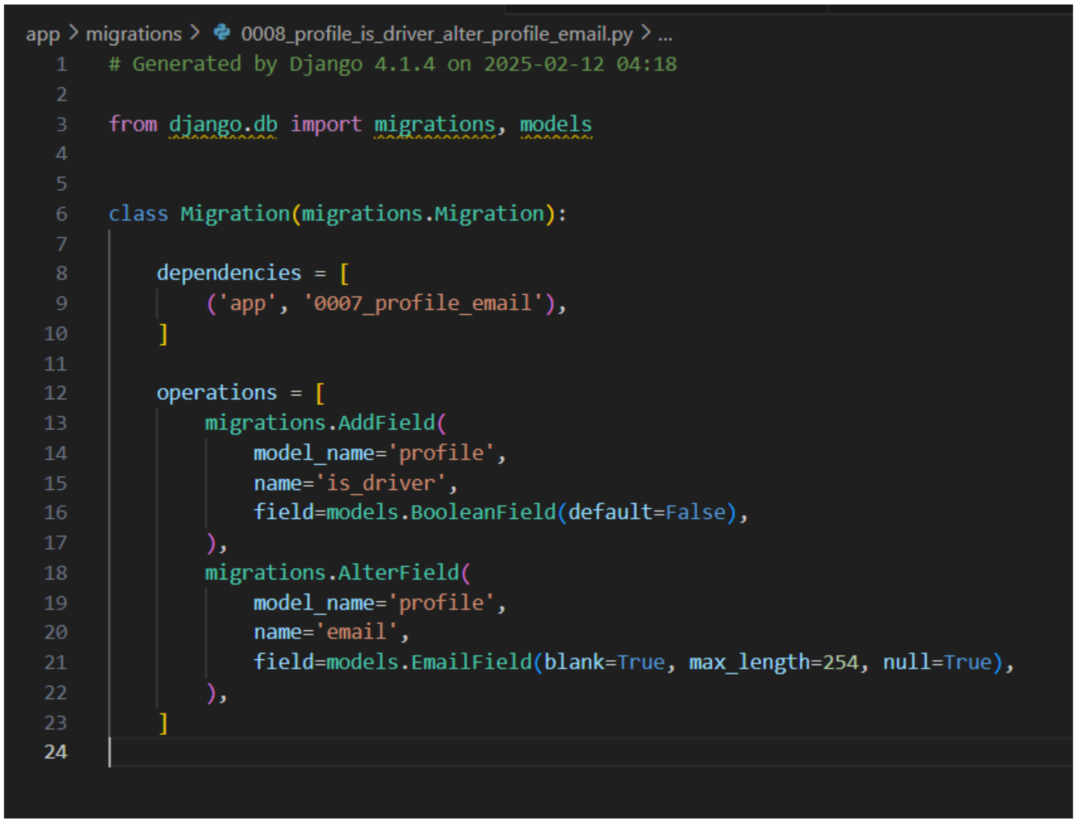
app > migrations > 0007_profile_email.py > ...

```python
# Generated by Django 4.1.4 on 2025-02-12 03:51

from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('app', '0006_profile_phone_number'),
    ]

    operations = [
        migrations.AddField(
            model_name='profile',
            name='email',
            field=models.EmailField(blank=True, max_length=254, null=True, unique=True),
        ),
    ]
```

6.3.8 Driver profile

This migration adds an is_driver field to the Profile model and modifies the email field.

1. Adds an is_driver field to track whether a user is a driver.
2. Defaults to False, meaning regular users are not drivers unless specified.
3. Modifies the email field to allow duplicates.



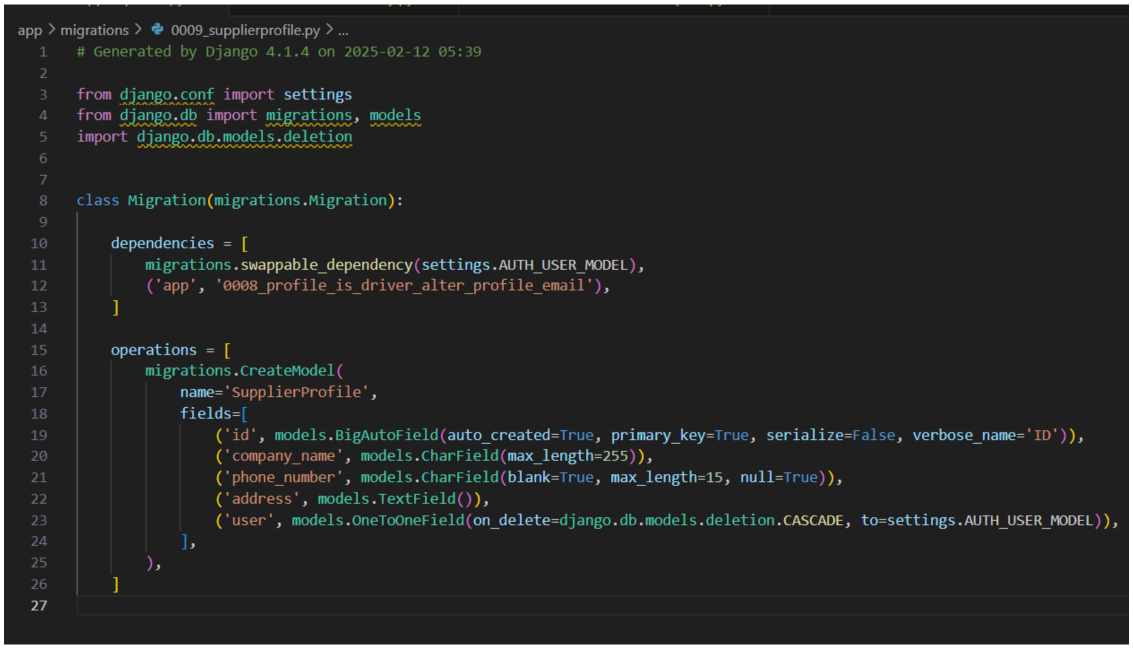
app > migrations > 0008_profile_is_driver_alter_profile_email.py > ...

```python
# Generated by Django 4.1.4 on 2025-02-12 04:18

from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('app', '0007_profile_email'),
    ]

    operations = [
        migrations.AddField(
            model_name='profile',
            name='is_driver',
            field=models.BooleanField(default=False),
        ),
        migrations.AlterField(
            model_name='profile',
            name='email',
            field=models.EmailField(blank=True, max_length=254, null=True),
        ),
    ]

```

6.3.9 Supplier Profile

This migration creates a SupplierProfile model, allowing users to be associated with supplier details such as company name, phone number, and address.

1. Creates a SupplierProfile model to store supplier details.
2. Links each supplier to a unique user (One-to-One relationship).
3. Ensures that supplier data is removed if the user is deleted.

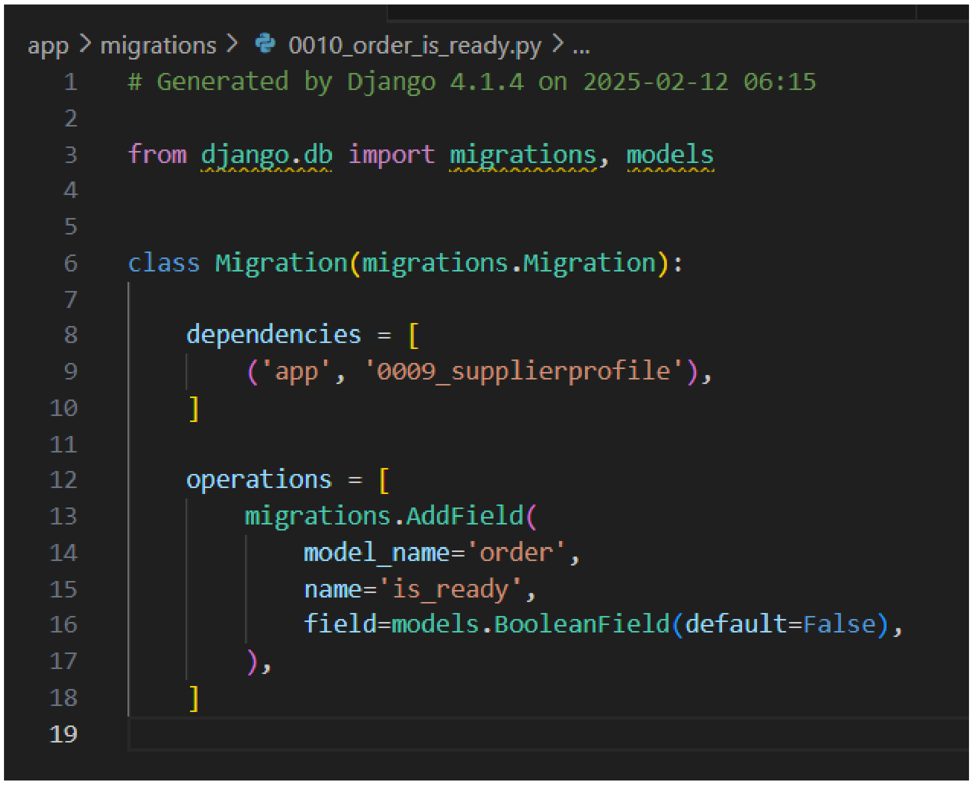


```
app > migrations > 0009_supplierprofile.py > ...
# Generated by Django 4.1.4 on 2025-02-12 05:39

from django.conf import settings
from django.db import migrations, models
import django.db.models.deletion


class Migration(migrations.Migration):

    dependencies = [
        migrations.swappable_dependency(settings.AUTH_USER_MODEL),
        ('app', '0008_profile_is_driver_alter_profile_email'),
    ]

    operations = [
        migrations.CreateModel(
            name='SupplierProfile',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('company_name', models.CharField(max_length=255)),
                ('phone_number', models.CharField(blank=True, max_length=15, null=True)),
                ('address', models.TextField()),
                ('user', models.OneToOneField(on_delete=django.db.models.deletion.CASCADE, to=settings.AUTH_USER_MODEL)),
            ],
        ),
    ]
```

6.3.10 Order ready

This migration adds an is_ready field to the Order model to indicate whether an order is ready for pickup or delivery.

1. Adds an is_ready field to track order readiness.
2. Defaults to False, meaning orders are not ready until marked otherwise.

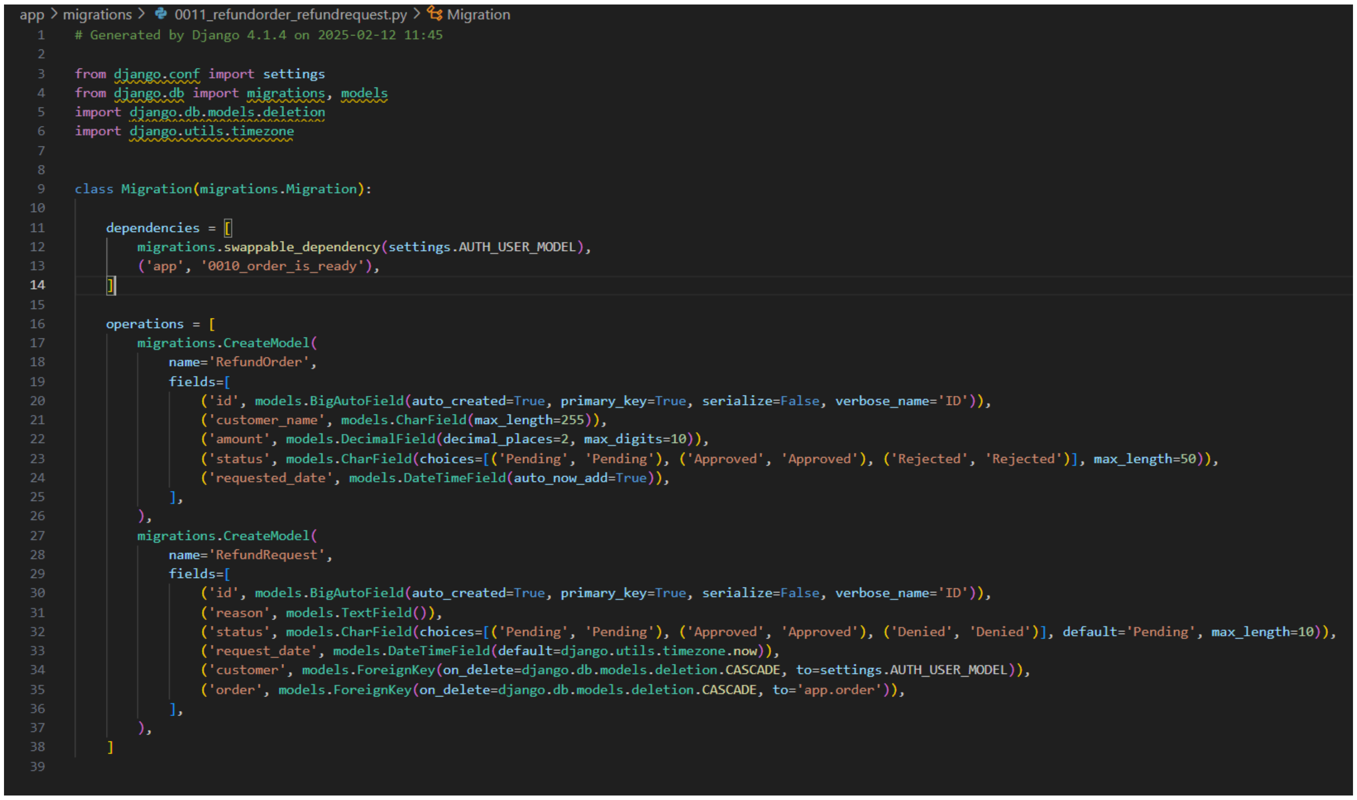


```
app > migrations > 0010_order_is_ready.py > ...
# Generated by Django 4.1.4 on 2025-02-12 06:15

from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('app', '0009_supplierprofile'),
    ]

    operations = [
        migrations.AddField(
            model_name='order',
            name='is_ready',
            field=models.BooleanField(default=False),
        ),
    ]
```

6.3.11 Prepare Order Component

This migration creates two new models: RefundOrder and RefundRequest, allowing users to request refunds and track their status.

1. Creates a RefundOrder model to track refund amounts and statuses.
2. Creates a RefundRequest model to allow users to submit refund requests.
3. Links refund requests to customers and orders for better tracking.



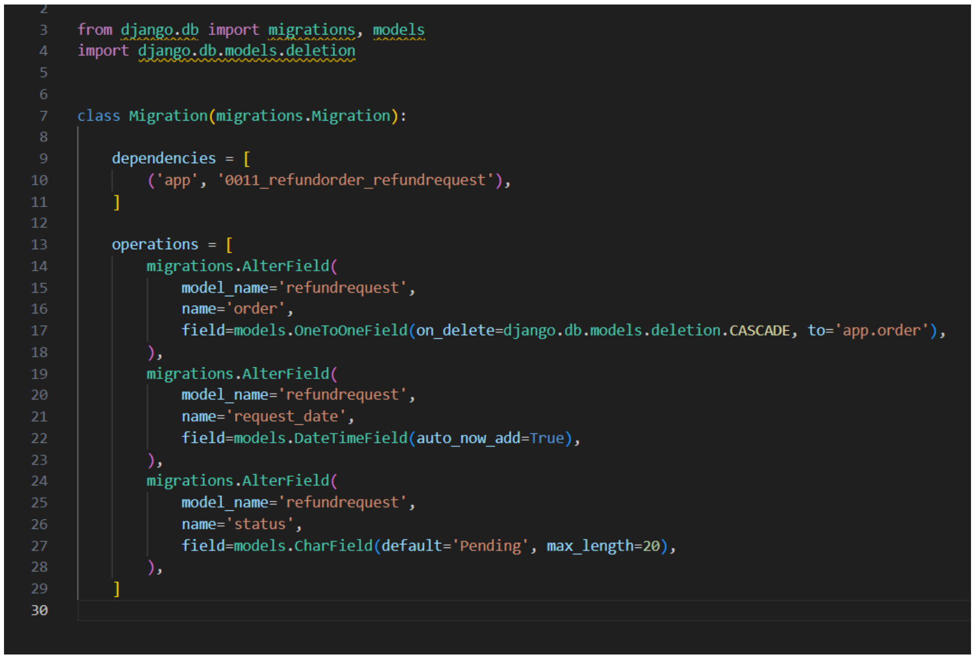
app > migrations > 0011_refundorder_refundrequest.py > Migration

```python
# Generated by Django 4.1.4 on 2025-02-12 11:45

from django.conf import settings
from django.db import migrations, models
import django.db.models.deletion
import django.utils.timezone


class Migration(migrations.Migration):

    dependencies = [
        migrations.swappable_dependency(settings.AUTH_USER_MODEL),
        ('app', '0010_order_is_ready'),
    ]

    operations = [
        migrations.CreateModel(
            name='RefundOrder',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('customer_name', models.CharField(max_length=255)),
                ('amount', models.DecimalField(decimal_places=2, max_digits=10)),
                ('status', models.CharField(choices=[('Pending', 'Pending'), ('Approved', 'Approved'), ('Rejected', 'Rejected')], max_length=50)),
                ('requested_date', models.DateTimeField(auto_now_add=True)),
            ],
        ),
        migrations.CreateModel(
            name='RefundRequest',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('reason', models.TextField()),
                ('status', models.CharField(choices=[('Pending', 'Pending'), ('Approved', 'Approved'), ('Denied', 'Denied')], default='Pending', max_length=10)),
                ('request_date', models.DateTimeField(default=django.utils.timezone.now)),
                ('customer', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to=settings.AUTH_USER_MODEL)),
                ('order', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='app.order')),
            ],
        ),
    ]
```

6.3.12 Alter refund request

This migration modifies the RefundRequest model by altering three fields. This is what it does :

1. Ensures each Order can have only one RefundRequest
2. Automatically records the request date when a refund is created
3. Sets the default refund status to "Pending"



```python
from django.db import migrations, models
import django.db.models.deletion


class Migration(migrations.Migration):

    dependencies = [
        ('app', '0011_refundorder_refundrequest'),
    ]

    operations = [
        migrations.AlterField(
            model_name='refundrequest',
            name='order',
            field=models.OneToOneField(on_delete=django.db.models.deletion.CASCADE, to='app.order'),
        ),
        migrations.AlterField(
            model_name='refundrequest',
            name='request_date',
            field=models.DateTimeField(auto_now_add=True),
        ),
        migrations.AlterField(
            model_name='refundrequest',
            name='status',
            field=models.CharField(default='Pending', max_length=20),
        ),
    ]
```

7 Testing

7.1 Testing Strategy

7.1.1 Testing Strategy Overview

The testing strategy aims to ensure that all system components function correctly and integrate seamlessly. The approach includes unit testing, integration testing, system testing, and user acceptance testing (UAT)

7.1.2 Testing Approach

1. **Unit Testing** – Individual components (e.g., login, order placement, refund request) are tested separately.
2. **Integration Testing** – Ensures modules (customer, driver, admin, supplier) work together as expected.
3. **System Testing** – Validates the entire system against requirements.
4. **User Acceptance Testing (UAT)** – Real users test the system for usability and performance.

7.1.3 Integration Testing Plan

Integration testing ensures proper interaction between system components. The key areas covered:

- **Customer & Order System** – Verify that customers can place, track, and cancel orders.
- **Payment Gateway & Refund Processing** – Test transactions, refunds, and failed payments.
- **Driver & Order Assignment** – Ensure drivers receive and update delivery status correctly.
- **Admin & User Management** – Validate admin control over customer and driver profiles.
- **Supplier & Inventory Management** – Check order processing, stock updates, and pricing.

7.2 Test Data

7.2.1 Registration Test Data

Scenario	Input	Expected Outcome
Valid registration	Email or Phone Number : alice@gmail.com Full Name : Lee Alice Username : alice Password Pass1234 Confirm Password : Pass1234	Successful registration as all details is fills out.
Invalid registration	Email or Phone Number : alice@gmail.com Full Name : Lee Alice Username : alice Password Pass12 Confirm Password : Pass12	Unsuccesful registration as the password does not contains at least 8 characters.

7.2.2 Login Test Data

Scenario	Input	Expected Outcome
Valid login	Username : alice Password Pass1234	Successful login as all details is correct.
Invalid login	Username : alice Password Pass4321	Unsuccesful registration as the password is wrong

7.2.3 Driver Authentication Test Data

Scenario	Input	Expected Outcome
Valid authentication	License number : 123456789876 License expiry date : 12/12/2025 Vehicle type : Car Vehicle colour :Red Vehicle number plate : BNP 4321	Successful authentication as all details is fill up.
Invalid authentication	License number : 123456789876 License expiry date : Vehicle type : Car Vehicle colour :Red Vehicle number plate : BNP 4321	Unsuccesful authentication as the details is not fill up

7.2.4 Refund Request Test Data

Scenario	Input	Expected Outcome
Request accepted	Reason for refund : Incorrect item delivered Details of refund : Request for 20 litre of RON 95 instead get 10 litre of RON 95	Successful refund request as it is a valid reason for refund.
Request rejected	Reason for refund : Other Details of refund : Changed mind after payment	Unsuccesful refund request as it is not a valid reason

7.2.5 Update Stock and Pricing Data Test Data

Scenario	Input	Expected Outcome
Updating stock and pricing data	Location : Ampang Item : Diesel B7 Current Stock : 2000 litre Price/Unit : RM 3.23	Stock and pricing data updated

7.3 Acceptance Testing

7.3.1 Customer

Criteria	Fulfilled	Remarks
Registration		
Login		
Profile management		
Order placement		
Make payment		
Track order		
Refund request		

Date tested : ____________
% Complete : ___________
Tested by : _____________
*Verified by :*_____________

7.3.2 Driver

Criteria	Fulfilled	Remarks
Registration and authentication		
Login		
Track order		

Date tested : ____________
% Complete : ___________
Tested by : _____________
*Verified by :*_____________

7.3.3 Admin

Criteria	Fulfilled	Remarks
Login		

Manage driver authentication		
Manage refund request		
Manage customer and driver profile		

Date tested : ____________
% Complete : ___________
Tested by : _____________
Verified by :_____________

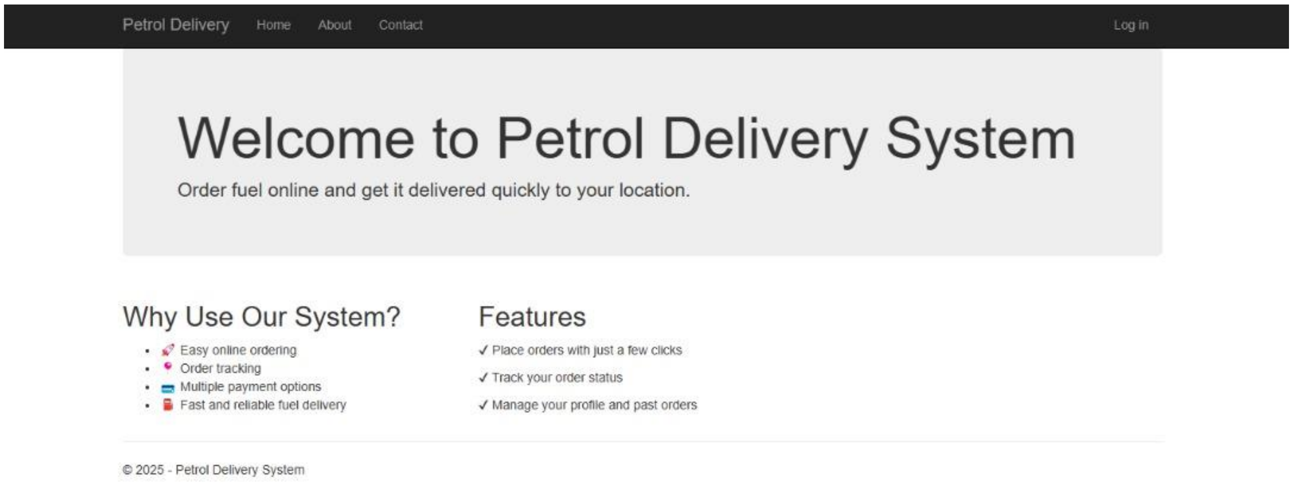
7.3.4 Supplier

Criteria	Fulfilled	Remarks
Login		
Prepare order		
Update pricing data and stock inventory		

Date tested : ____________
% Complete : ___________
Tested by : _____________
Verified by :_____________

8 Sample Screens

8.1 Main Screen



Welcome to Petrol Delivery System

Order fuel online and get it delivered quickly to your location.

Why Use Our System?

- Easy online ordering
- Order tracking
- Multiple payment options
- Fast and reliable fuel delivery

Features

✓ Place orders with just a few clicks
✓ Track your order status
✓ Manage your profile and past orders

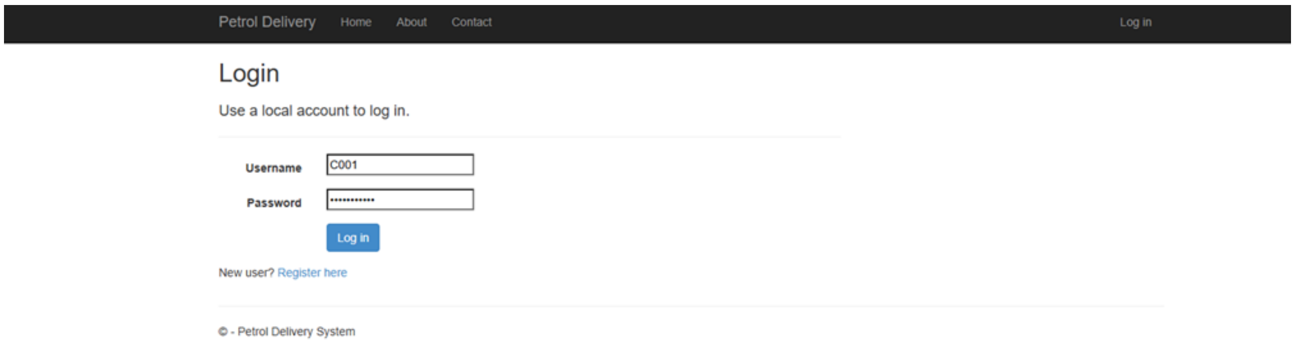
© 2025 - Petrol Delivery System

Figure 8.1 System Main Screen

This is the online petrol delivery system main screen that will load when the using the system.

8.2 Subsystem Screen

8.2.1 Customer screen



Login

Use a local account to log in.

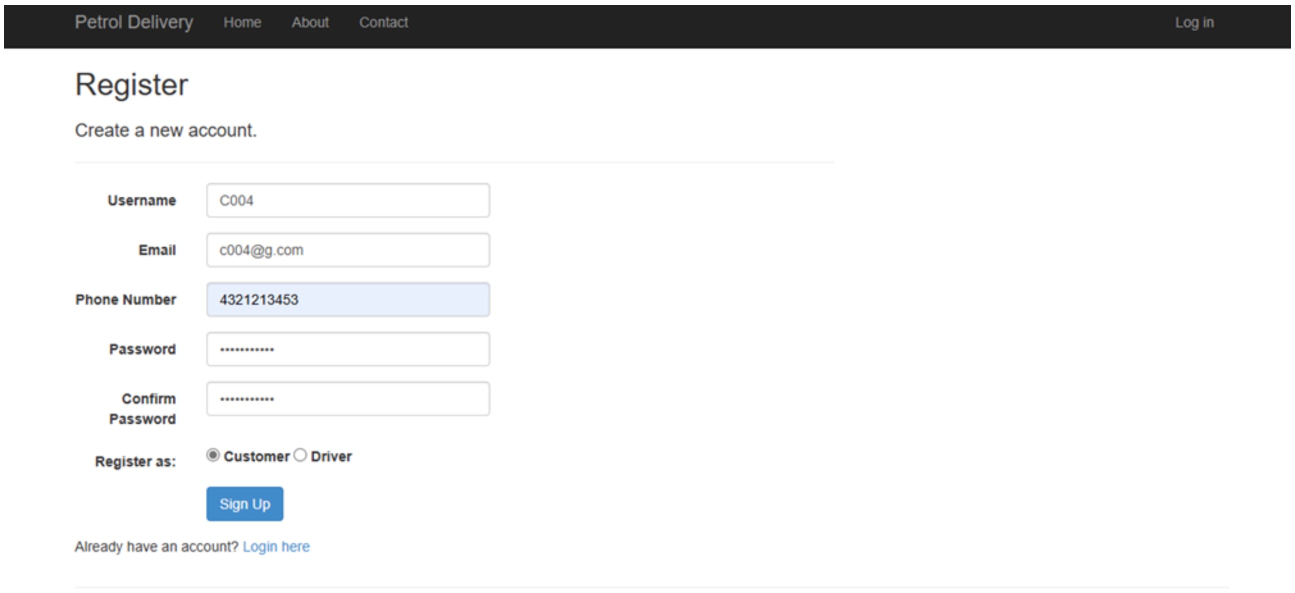
Username: C001

Password: ••••••••••••

New user? Register here

© - Petrol Delivery System

Figure 8.2.1.1: login screen



Register

Create a new account.

Username	C004
Email	c004@g.com
Phone Number	4321213453
Password	•••••••••••
Confirm Password	•••••••••••
Register as:	[✓ Customer] Driver

Already have an account? Login here

Figure 8.2.1.2: register screen

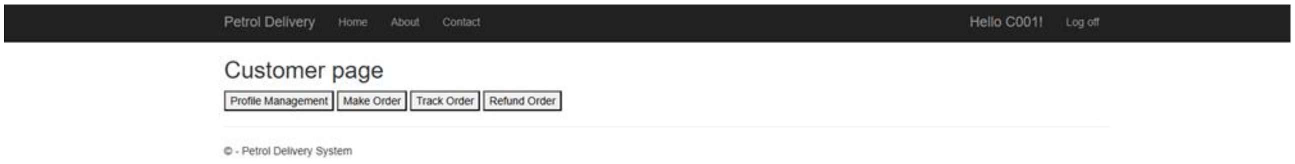
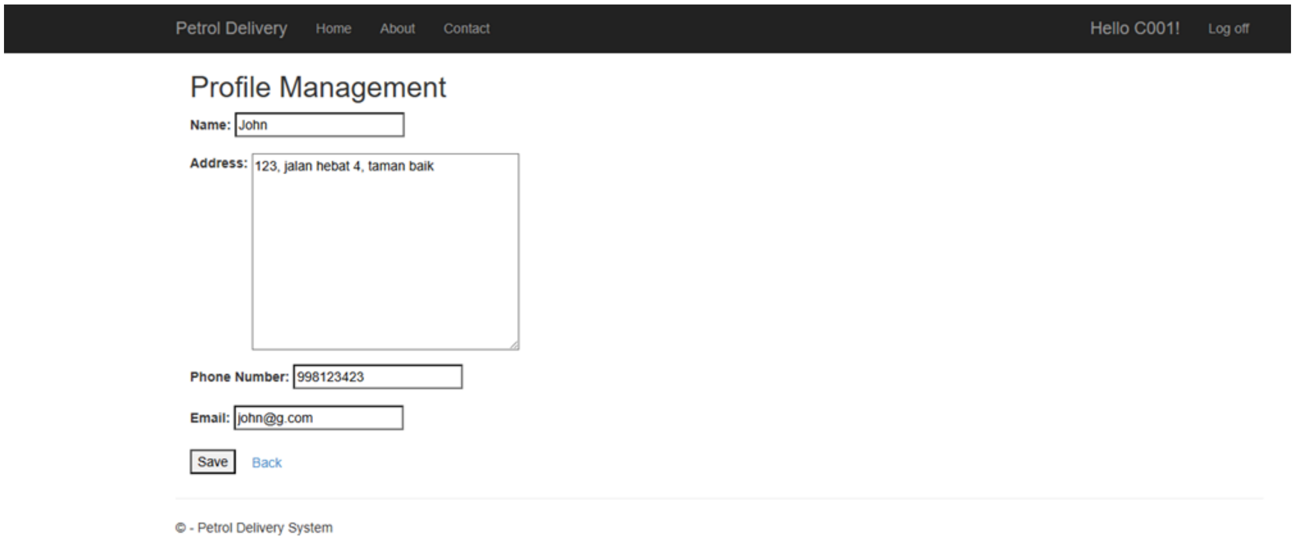


Figure 8.2.1.3: main page for customer



Profile Management

Name:	John
Address:	123, jalan hebat 4, taman baik
Phone Number:	998123423
Email:	john@g.com

Figure 8.2.1.4: profile management screen

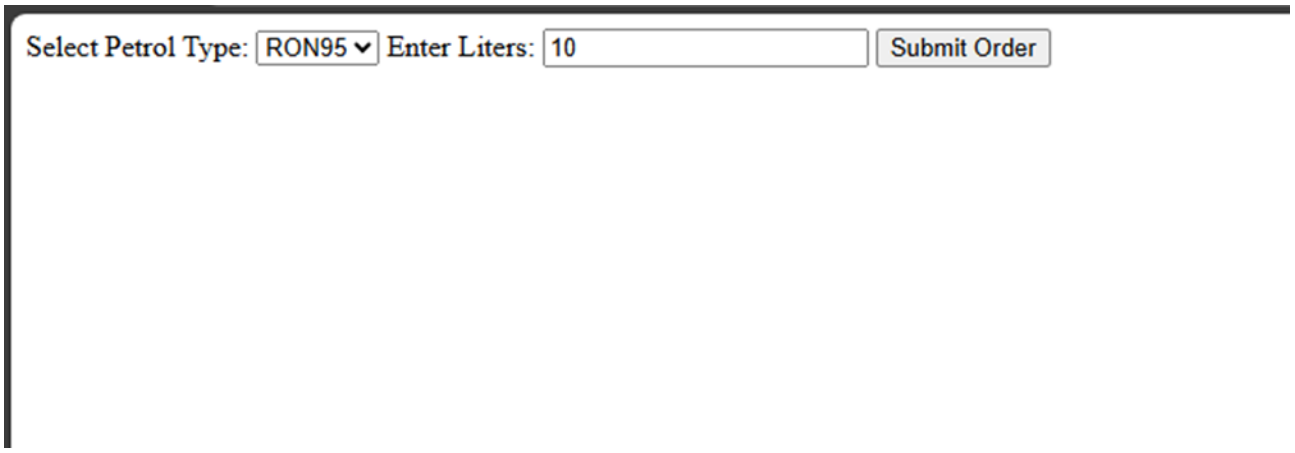
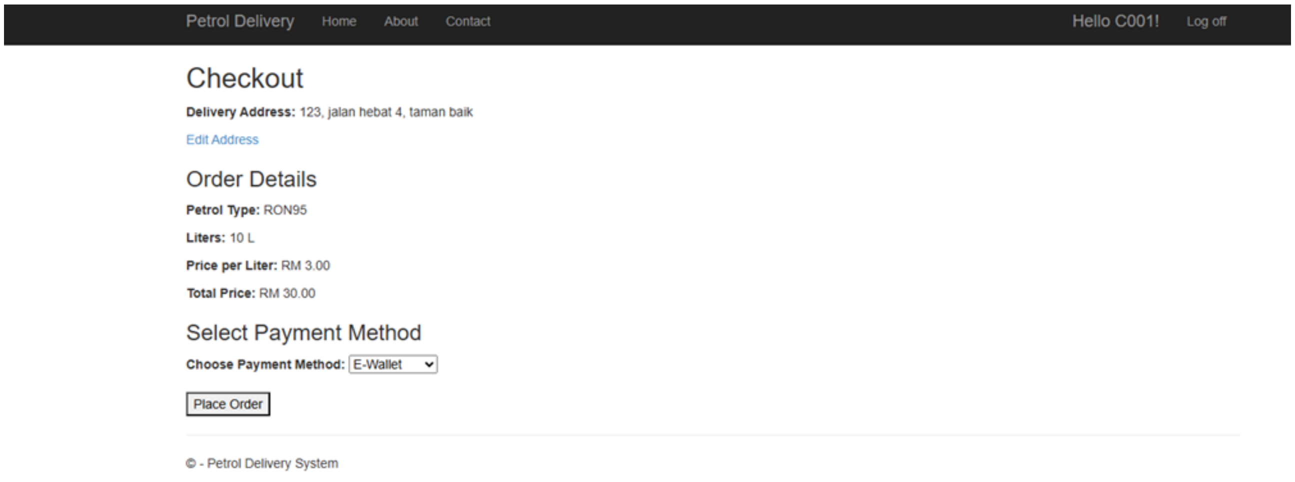

Select Petrol Type: RON95 Enter Liters: 10 Submit Order

Figure 8.2.1.5: make order screen



Checkout

Delivery Address: 123, jalan hebat 4, taman baik

Edit Address

Order Details

Petrol Type: RON95

Liters: 10 L

Price per Liter: RM 3.00

Total Price: RM 30.00

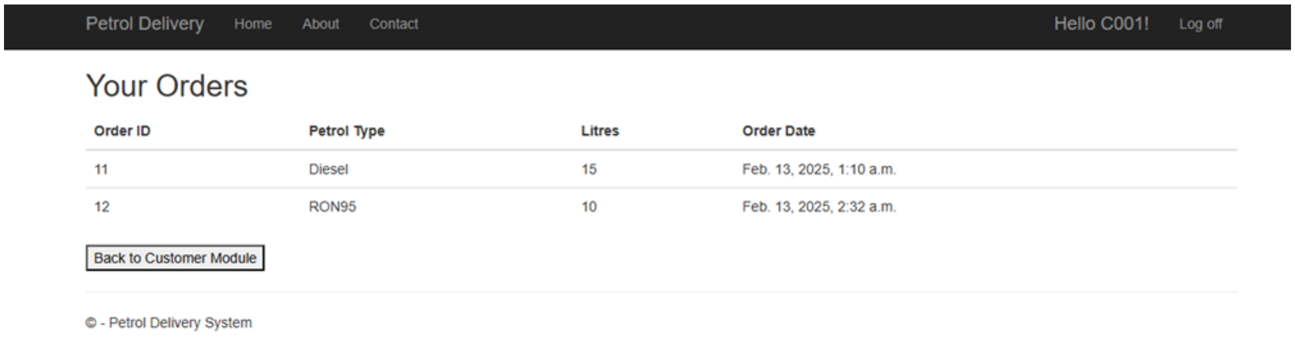
Select Payment Method

Choose Payment Method: E-Wallet

Place Order

© - Petrol Delivery System

Figure 8.2.1.6: checkout screen



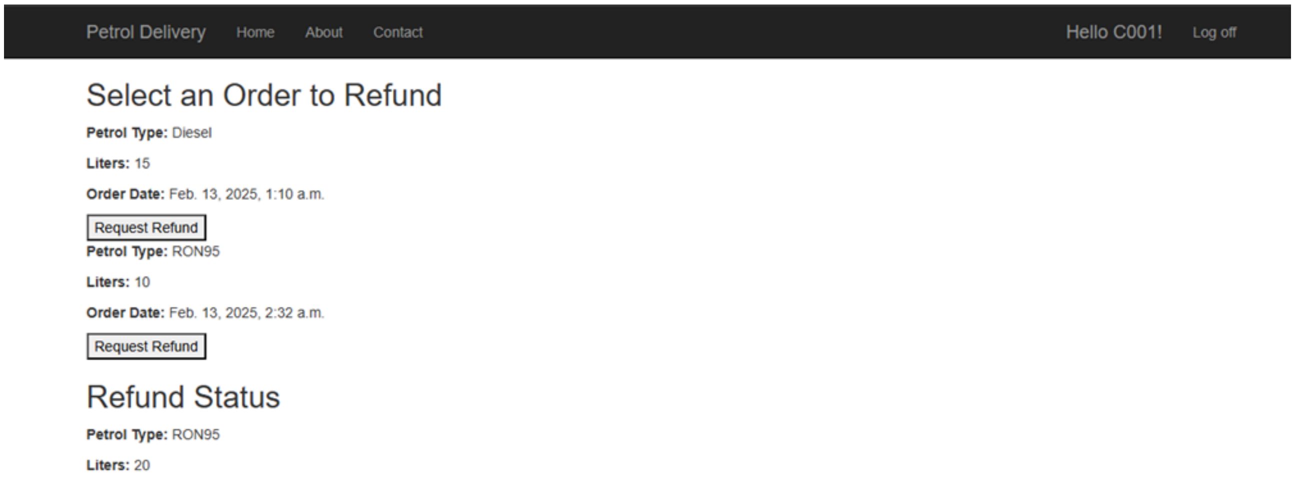
Your Orders

Order ID	Petrol Type	Litres	Order Date
11	Diesel	15	Feb. 13, 2025, 1:10 a.m.
12	RON95	10	Feb. 13, 2025, 2:32 a.m.

Back to Customer Module

© - Petrol Delivery System

Figure 8.2.1.7: track order screen



Select an Order to Refund

Petrol Type: Diesel

Liters: 15

Order Date: Feb. 13, 2025, 1:10 a.m.

Request Refund

Petrol Type: RON95

Liters: 10

Order Date: Feb. 13, 2025, 2:32 a.m.

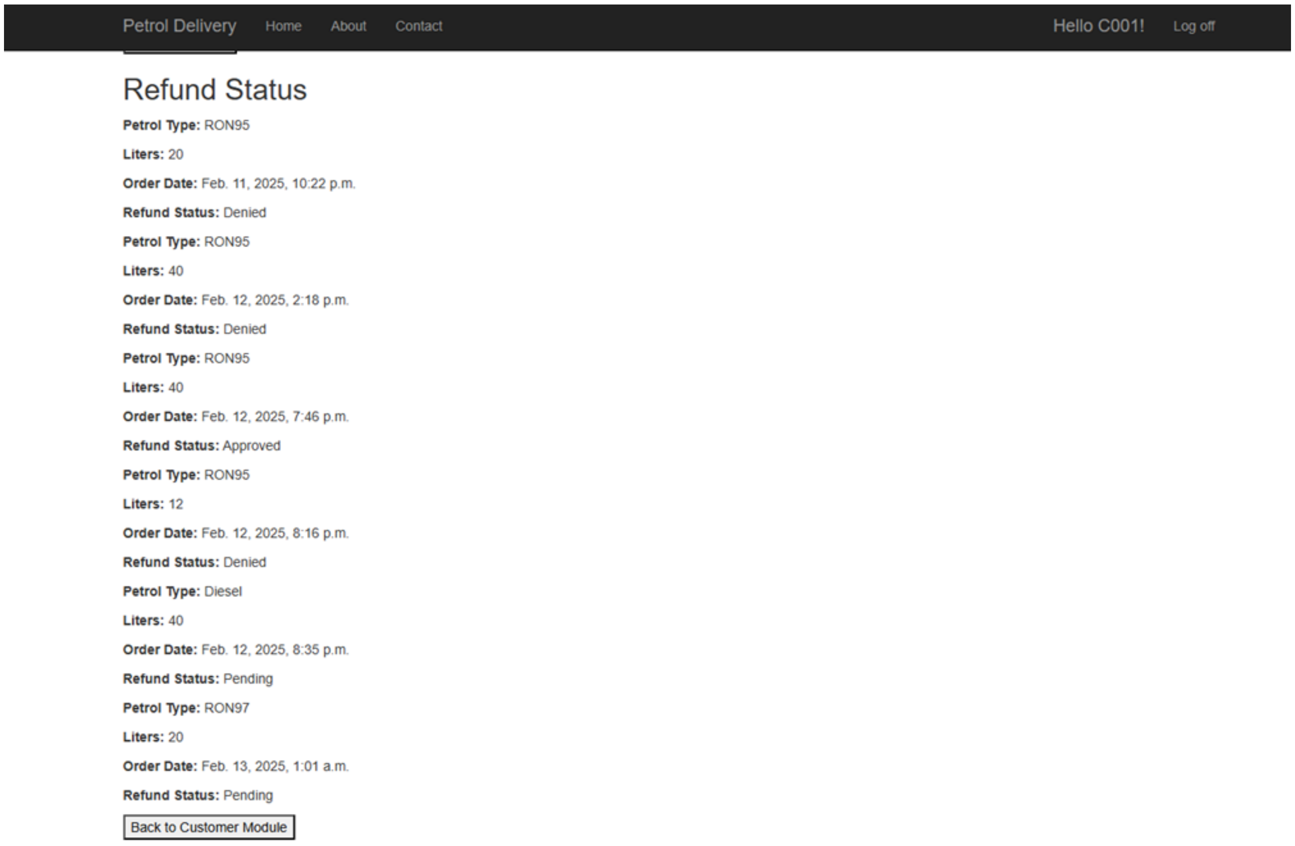
Request Refund

Refund Status

Petrol Type: RON95

Liters: 20

Figure 8.2.1.8 refund order screen 1



Refund Status

Petrol Type: RON95

Liters: 20

Order Date: Feb. 11, 2025, 10:22 p.m.

Refund Status: Denied

Petrol Type: RON95

Liters: 40

Order Date: Feb. 12, 2025, 2:18 p.m.

Refund Status: Denied

Petrol Type: RON95

Liters: 40

Order Date: Feb. 12, 2025, 7:46 p.m.

Refund Status: Approved

Petrol Type: RON95

Liters: 12

Order Date: Feb. 12, 2025, 8:16 p.m.

Refund Status: Denied

Petrol Type: Diesel

Liters: 40

Order Date: Feb. 12, 2025, 8:35 p.m.

Refund Status: Pending

Petrol Type: RON97

Liters: 20

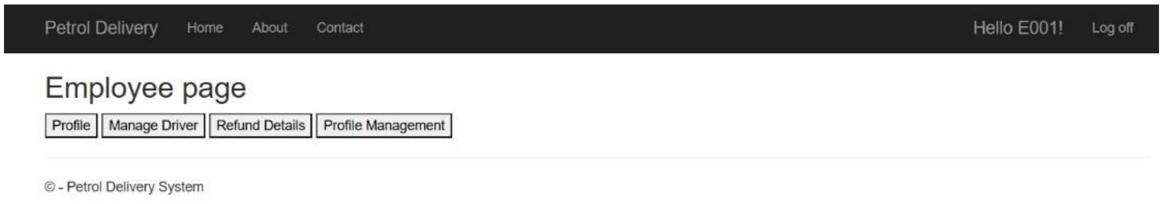
Order Date: Feb. 13, 2025, 1:01 a.m.

Refund Status: Pending

Back to Customer Module

Figure 8.2.1.9: refund order screen 2

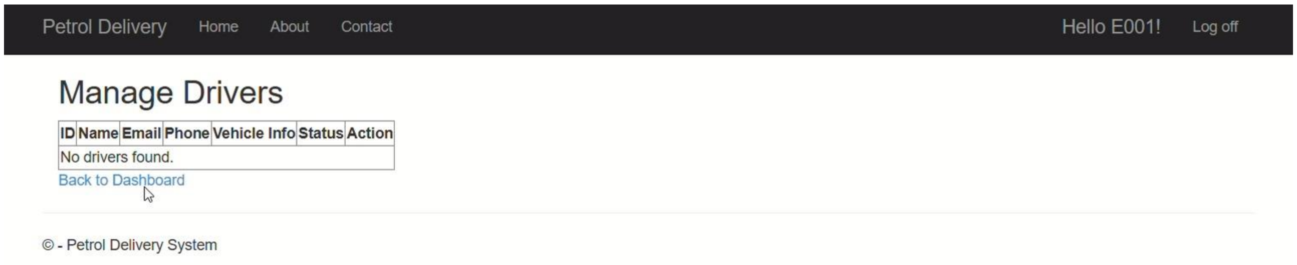
8.2.2 Admin Screens



Employee page

© - Petrol Delivery System

Figure 8.2.2.1 Home Page (Admin



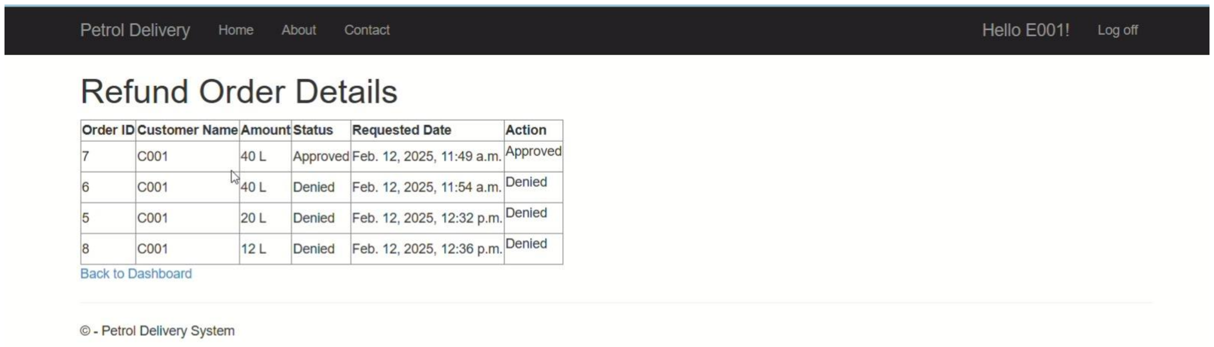
Manage Drivers

ID	Name	Email	Phone	Vehicle Info	Status	Action
No drivers found.						

Back to Dashboard

© - Petrol Delivery System

Figure 8.2.2.2 Manage Driver Authentication (Admin)



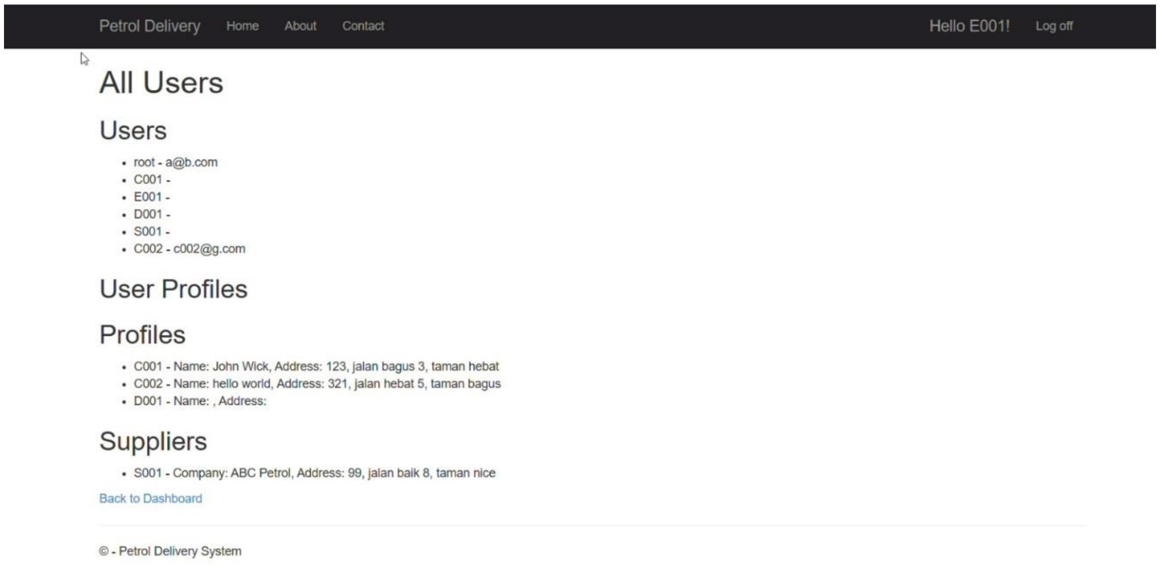
Refund Order Details

Order ID	Customer Name	Amount	Status	Requested Date	Action
7	C001	40 L	Approved	Feb. 12, 2025, 11:49 a.m.	Approved
6	C001	40 L	Denied	Feb. 12, 2025, 11:54 a.m.	Denied
5	C001	20 L	Denied	Feb. 12, 2025, 12:32 p.m.	Denied
8	C001	12 L	Denied	Feb. 12, 2025, 12:36 p.m.	Denied

Back to Dashboard

© - Petrol Delivery System

Figure 8.2.2.3 Manage Refund Request (Admin)



All Users

Users

- root - a@b.com
- C001 -
- E001 -
- D001 -
- S001 -
- C002 - c002@g.com

User Profiles

Profiles

- C001 - Name: John Wick, Address: 123, jalan bagus 3, taman hebat
- C002 - Name: hello world, Address: 321, jalan hebat 5, taman bagus
- D001 - Name: , Address:

Suppliers

- S001 - Company: ABC Petrol, Address: 99, jalan baik 8, taman nice

Back to Dashboard

© - Petrol Delivery System

Figure 8.2.2.4 Manage Customer and Driver Profile (Admin

8.2.3 Supplier Screens

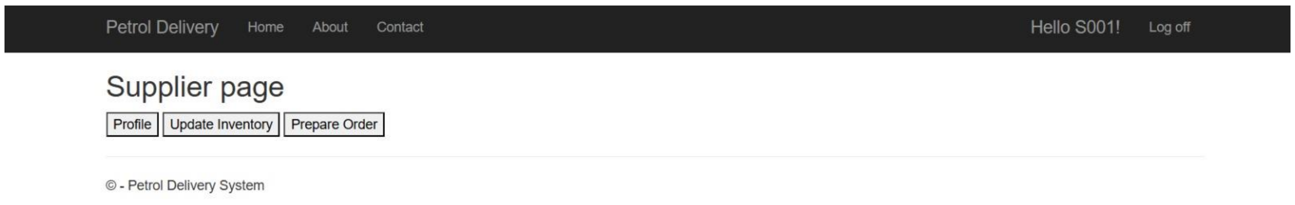
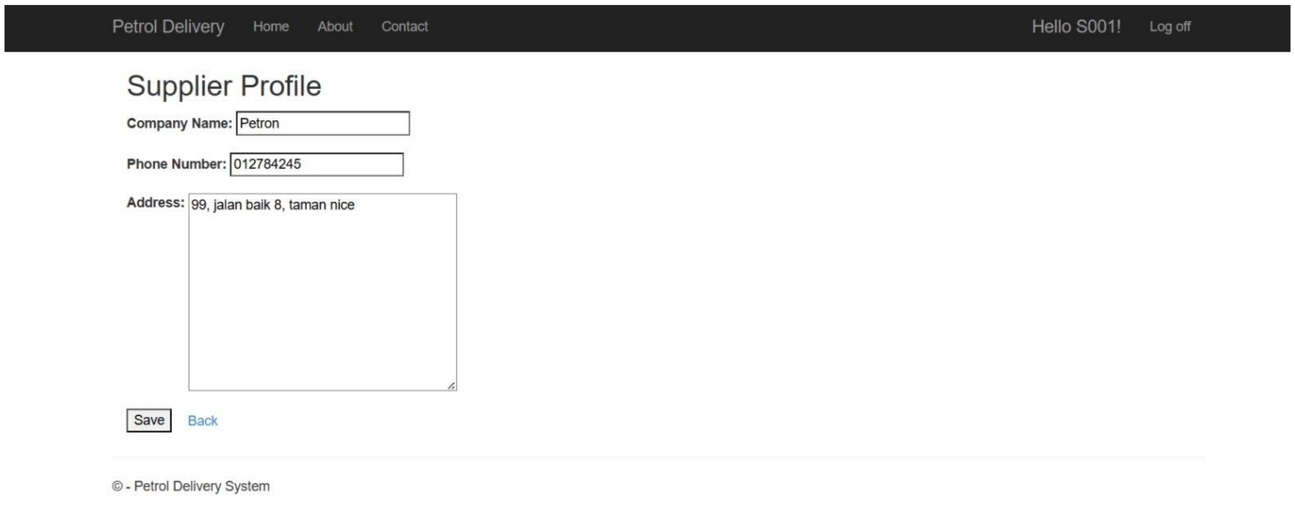


Figure 4.3.1 Supplier main page

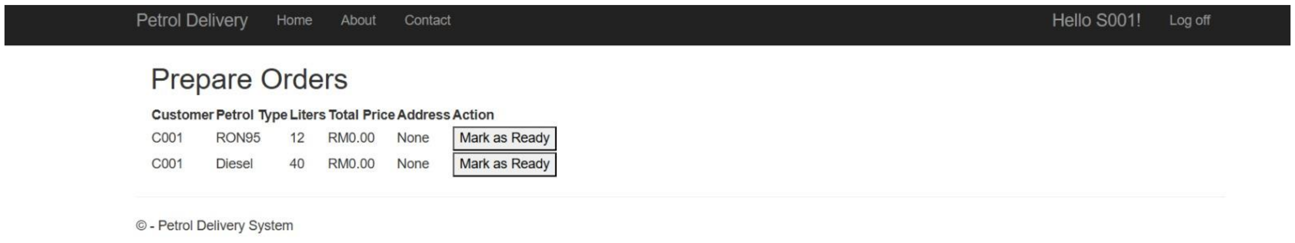


Company Name: Petron

Phone Number: 012784245

Address: 99, jalan baik 8, taman nice

Figure 8.2.3.1 Supplier profile management page

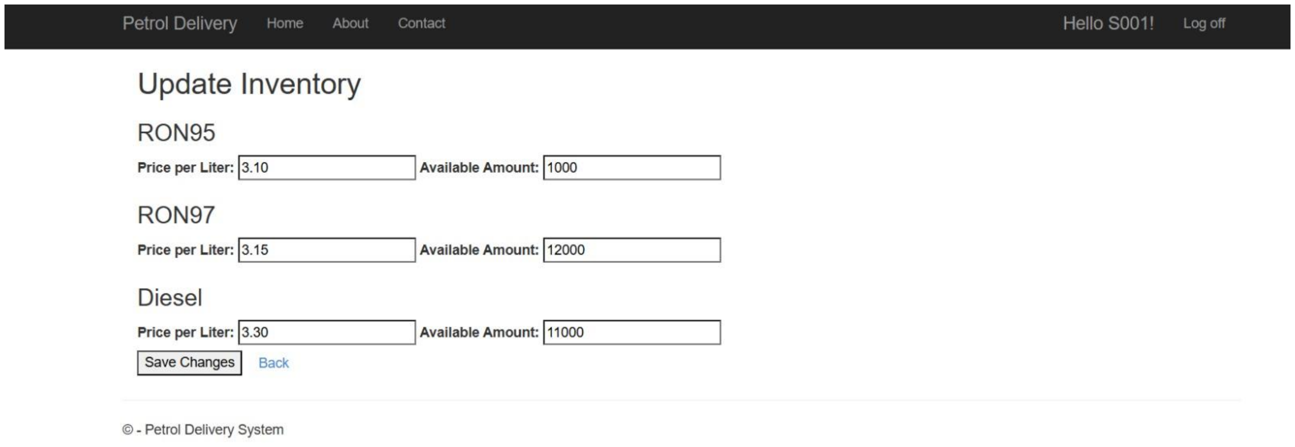


Prepare Orders

Customer	Petrol Type	Liters	Total Price	Address	Action
C001	RON95	12	RM0.00	None	Mark as Ready
C001	Diesel	40	RM0.00	None	Mark as Ready

© - Petrol Delivery System

Figure 8.2.3.2 Prepare order page



Update Inventory

RON95

Price per Liter: 3.10 Available Amount: 1000

RON97

Price per Liter: 3.15 Available Amount: 12000

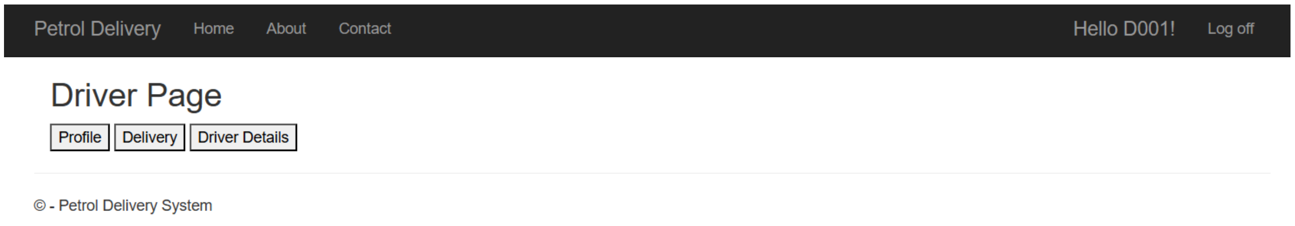
Diesel

Price per Liter: 3.30 Available Amount: 11000

© - Petrol Delivery System

Figure 8.2.3.3 Update stock inventory and pricing data

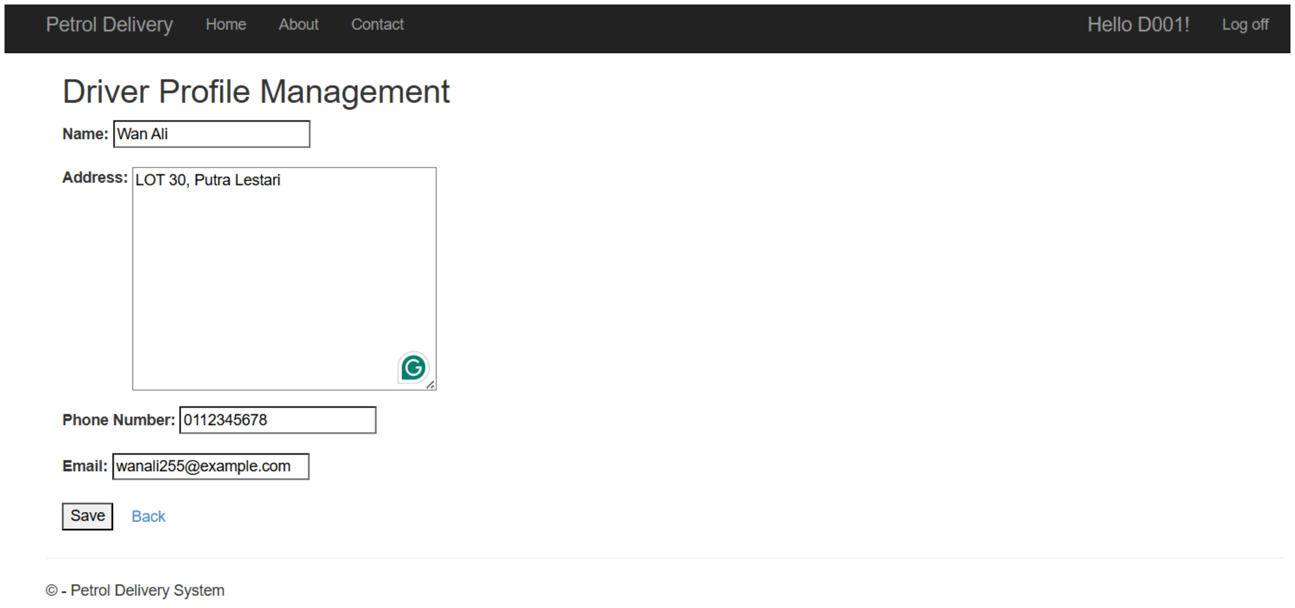
8.2.4 Driver Screens



Driver Page

© - Petrol Delivery System

Figure 8.2.4.1 Driver main page



Driver Profile Management

Name: Wan Ali

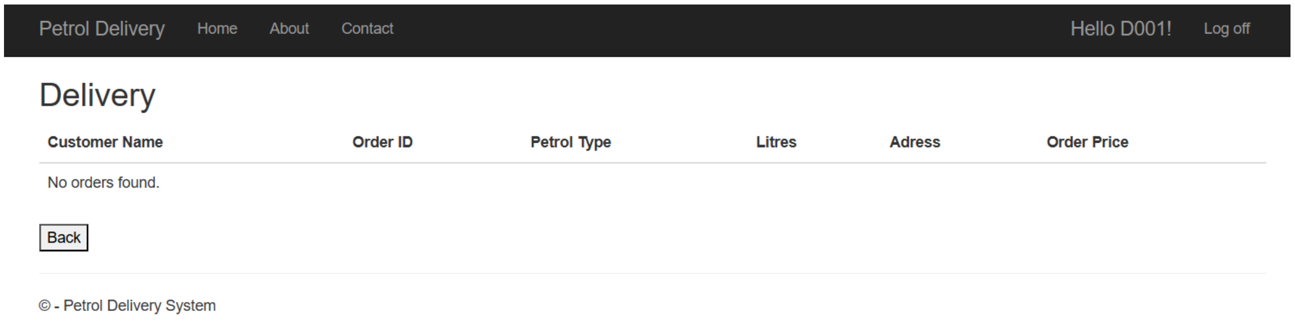
Address: LOT 30, Putra Lestari

Phone Number: 0112345678

Email: wanali255@example.com

© - Petrol Delivery System

Figure 8.2.4.2 Driver profile management

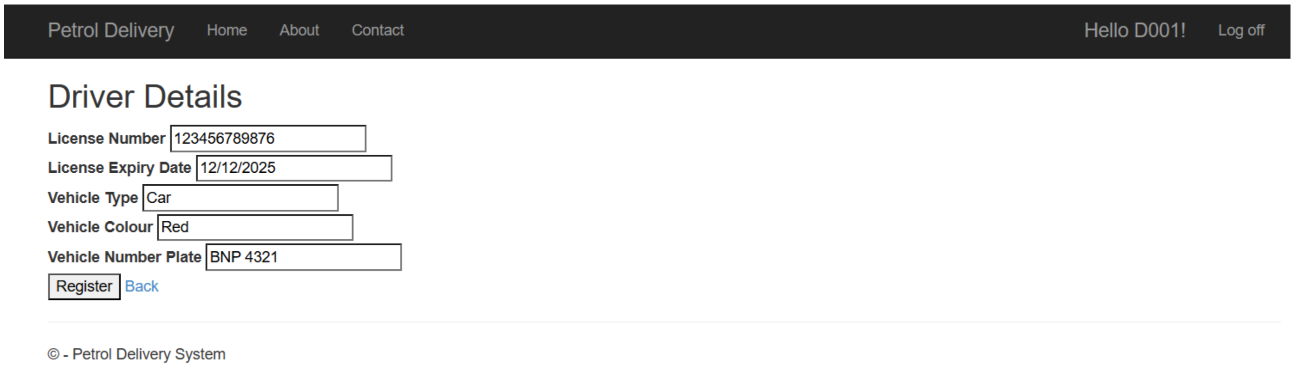


Delivery

Customer Name	Order ID	Petrol Type	Litres	Adress	Order Price
No orders found.					

© - Petrol Delivery System

Figure 8.2.4.3 Delivery Page



Driver Details

License Number	123456789876
License Expiry Date	12/12/2025
Vehicle Type	Car
Vehicle Colour	Red
Vehicle Number Plate	BNP 4321

Figure 8.2.4.4 Driver Authenthication Page

9 Conclusion

The Online Petrol Delivery System was completed, demonstrating the team's ability to plan, develop, test, and finalize a functional system. The project followed a structured approach to complete the project. The final product reflects the collective effort and dedication of the team, delivering a software solution that is required.

9.1 Completion of software

The software development process was carried out systematically, covering necessary stages, from requirement analysis and design to implementation and testing. The planned features and functionalities were developed and integrated into the system. The system hopefully met the project scope and user requirements.

9.2 Software Quality Assurance

Ensuring software quality was a critical aspect of the project. Bug tracking and debugging processes were followed to enhance system stability and performance. The final software product may met the industry standards.

9.3 Group Collaboration

Teamwork and collaboration is very important for the project success. Each team member was assigned specific roles and responsibilities. Regular discussions helped ensure alignment with project goals. Communication tools were utilized to maintain coordination and track progress. Undeniably there is diffculties in getting teamwork fully from group member at certain time.

9.4 Problems Encountered

Several challenges were encountered during the project, including technical issues, debugging complexities, group collaboration and time constraints. Some requirements needed additional research and troubleshooting, which extended development time. Coordination among team members was affected by scheduling conflicts, and from lack of effort. Despite these difficulties, the team adapted, ensuring that the project was completed.

10 User Guide

1. Unzip the folder
2. Run Visual Studio Code, and open the folder
3. Edit the file env\pyvenv.cfg according to the path of the programs
4. In Visual Studio Code, select Python interpreter with ‘env’
5. In the terminal, type in “Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope Process”, then “env\Scripts\activate”
6. Then enter the command “python manage.py runserver”
7. Then press ctrl and click the link in the terminal
8. Can login with credentials or create a new user
9. After login, can use the function provided to the actor in the system