TSE2101
Individual Report

for

Nursery Plant Shopping System

Version <1.0>

Tutorial Section: TT3L

Group No.: Group 3

EMILY PHANG RU YING **1211102687**

Date: 08/02/2024

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Version 1.0	Emily Phang Ru Ying	This version includes the system overview of the nursery plant shopping system focusing on delivery man. It includes the description of the system, delivery man use cases and assumptions of the system from the delivery man role. It contains a use case diagram followed by class diagram, state diagram and a data flow diagram. It also includes the use case descriptions with sequence diagrams. It contains the development of the database design with a data dictionary and data structures. Architecture design and interface design are also provided. Subsystem components with more detailed activity diagrams are included. For the implementation of the system, this version includes the development environment, main program codes and sample screens of the system. For testing of the system, the report includes the test data used for testing, acceptance test and test results. Lastly, to conclude the report, project achievements, quality assurance, problems encountered and remarks were included.	07/02/2024

1 System Overview

1.1 Description

The Nursery Plant Shopping System serves as a platform that facilitates interactions among four primary user roles: administrator, guests, customers, and delivery men. I am in charge of the delivery man. Delivery men are able to use the system to streamline order processing, from accepting delivery orders to confirming successful deliveries, making sure orders reach customers smoothly. Delivery men are also able to create a user account and edit their profile details or change their passwords. This nursery plant shopping system combines plant management, order processing, customer engagement, and delivery logistics, ensuring a fluid and user-friendly experience for all parties involved.

1.2 Use Cases

Table 1.1 shows the use cases of the delivery man within the system. Delivery Men are involved in the delivery order process. They are able to streamline order processing, from accepting delivery orders to confirming successful deliveries, making sure orders reach customers smoothly. Besides, delivery men are also able to create a user account and edit their profile details or change their passwords.

Table 1.1: Actor and Respective Use Cases

Actor	Use Cases
Delivery Man	A delivery man creates a user account.
	A delivery man changes their password.
	A delivery man can edit profile details.
	A delivery man accepts a delivery order.
	A delivery man views accepted deliveries and confirms delivery.

1.3 Assumptions and Dependencies

In this project, there are several assumptions that we have made:

1. Regardless of the number of times a delivery order is declined, eventually, there will be a deliveryman who accepts and successfully completes the order.

2. Customers are always expected to check their order status to see if their order has been delivered and out for delivery.

2 Requirements

2.1 Use Case Diagram

Figure 2.1 is a use-case diagram for a Nursery Plant Shopping System, depicting the interactions between the Delivery Man and the system. The Delivery Man's role is focused on managing delivery orders, editing personal information, and creating a delivery man's account.

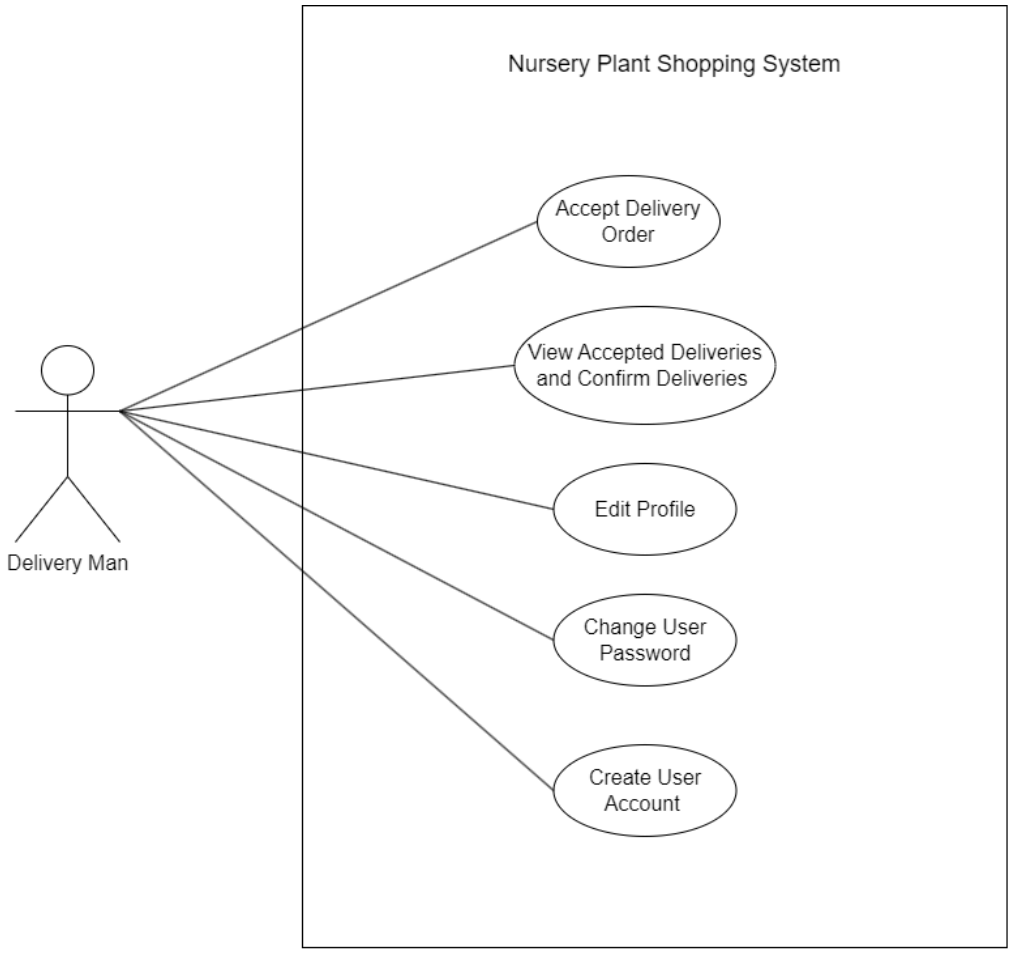


Figure 2.1: Nursery Plant Shopping System Delivery Man Use Case Diagram

2.2 Class Diagram

Figure 2.2 shows the nursery plant shopping system delivery man class diagram. The system contains several classes with specific attributes and methods, forming a network of interactions primarily for a Delivery Man role in the nursery plant shopping platform.

The DeliveryMan class is defined with multiple attributes, including delman_ID, deliveryman_name, deliveryman_address, deliveryman_state, deliveryman_ic, and deliveryman_phone_number, all of which are essential for capturing the delivery personnel's personal and contact information. The class is equipped with several operations to manage the delivery process effectively, each method enabling the delivery person to interact with the system's different facets.

The diagram illustrates that a DeliveryMan is a specialized type of User, as indicated by the inheritance relationship. This relationship suggests that the delivery personnel inherit properties and methods from the 'User' class, which includes attributes like user_ID, email, password, and boolean flags to indicate whether the user is a customer, deliveryman, or admin.

The DeliveryMan class is associated with the Order class via a one-to-many relationship, indicated by the manages relationship. This implies that one delivery person can manage multiple orders, but each order is managed by only one delivery person.

Furthermore, the Order class holds key information, such as order_ID, customer_ID, payment_ID, delman_ID, and order_status, and is linked to the 'Payment' class to track financial transactions associated with each order.

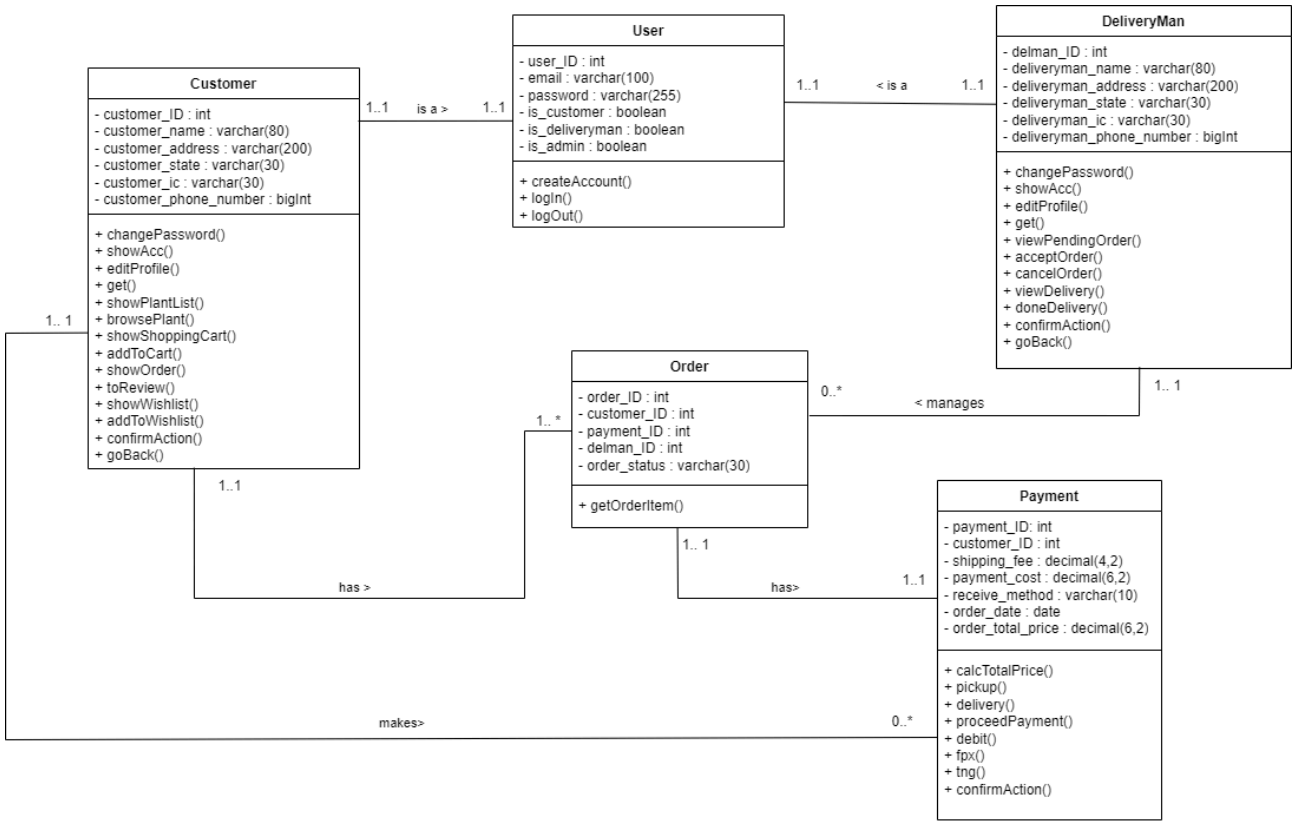


Figure 2.2: Nursery Plant Shopping System Delivery Man Class Diagram

2.3 State Diagrams

Figure 2.3 illustrates the State Transition Diagram for a Nursery Plant Shopping System. The user will begin with the “TLET Nursery Plant Shopping System Screen” as the initial state, from which users can navigate to the “Login Screen”. Upon successful login, the system transitions to different home screens based on the user's role which are “Administrator Home Screen” for admin login, “Customer Home Screen” for customer login, and “Delivery Man Home Screen” for delivery personnel login. Each home screen provides a “Logout” transition leading back to the “Login Screen”.

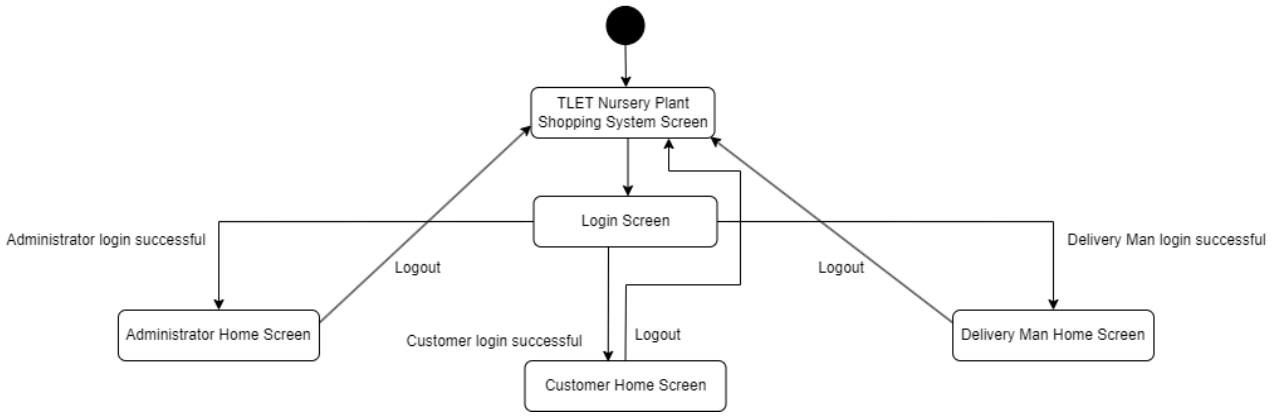


Figure 2.3: State Transition Diagram for Nursery Plant Shopping System

2.3.1 Delivery Man State Transition Diagram

Figure 2.4 presents the "State Transition Diagram for Delivery Man" in the TLET Nursery Plant Shopping System. It begins at the “TLET Nursery Plant Shopping System Screen” and transitions to the “Login Screen”, where upon successful login, the Delivery Man reaches the “Delivery Man Home Screen”. After a successful login, the delivery man can navigate through various system states. Selecting the "Pending Orders section" takes the delivery man to a "Pending Order screen" where they can accept or reject orders. They can also navigate to the "Accepted Deliveries screen" to view the orders that are accepted by clicking the "Accepted Deliveries section". After successfully delivering the order, the Delivery Man chooses the respective delivery order and clicks the "Confirm Delivery" button at the "Accepted Deliveries screen". The delivery man can also access "Account Settings" and be directed to the “Account Settings Screen” for personal information and security settings, including password changes. Successful password updates return them to the “Account Settings Screen”.

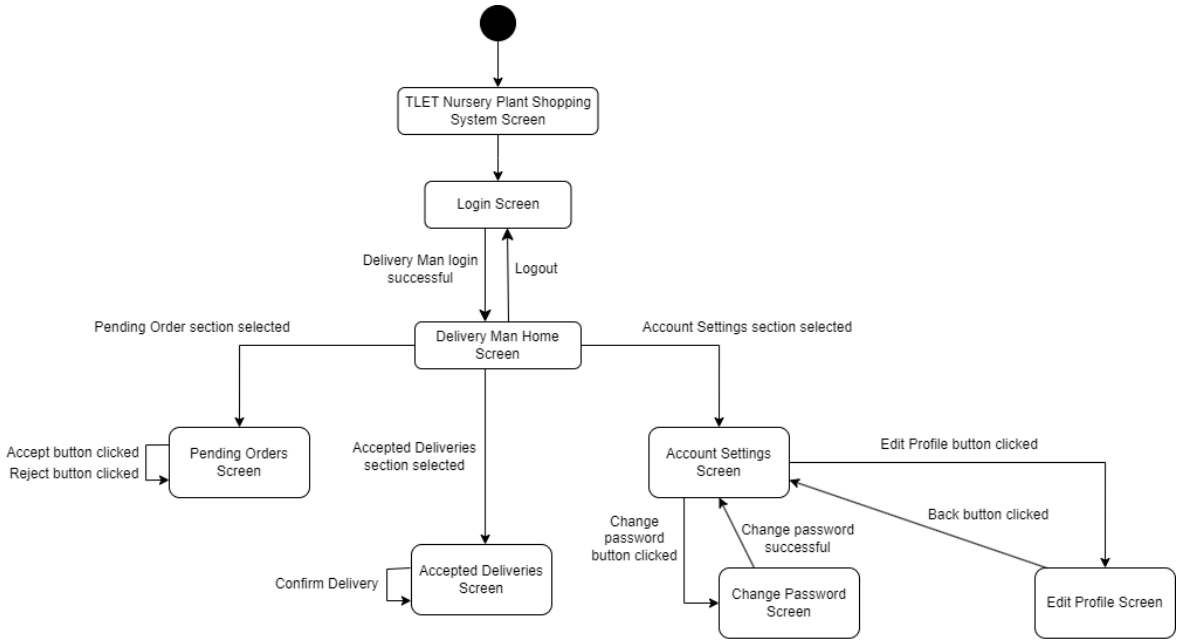


Figure 2.4: State Transition Diagram for Delivery Man

2.4 Data Flow Diagram

Figure 2.5 shows the Data Flow Diagram for the Nursery Plant Shopping System, highlighted through various data flows and system processes. In this system, the 'Delivery Man' is a critical agent responsible for the execution of order deliveries. Initially, the 'Delivery Man' enters the system by creating a user account, supplying essential details such as their name, address, and contact information, which are stored in the 'User Database'. They maintain their account's integrity through features that allow them to edit their profile and change their password, ensuring that personal and system security is upheld.

The core responsibilities of the 'Delivery Man' are encapsulated within the order delivery process. An 'Administrator' assigns orders to the 'Delivery Man', which marks the beginning of the delivery lifecycle. The 'Delivery Man' interacts with the 'Order Database' to retrieve the details of assigned deliveries, updating their status to 'Ready' or 'Completed' as they progress. The diagram illustrates the 'Delivery Man' accepting the delivery order, updating the status to 'Out for Delivery', and, upon successful delivery, confirming the completion of the delivery. This confirmation updates the order's status to 'Completed' in the 'Order Database', signifying the end of the delivery process. This signifies the 'Delivery Man's' direct involvement in the order management system, where they bridge the gap between the plant nursery and the customer, ensuring timely and accurate deliveries.

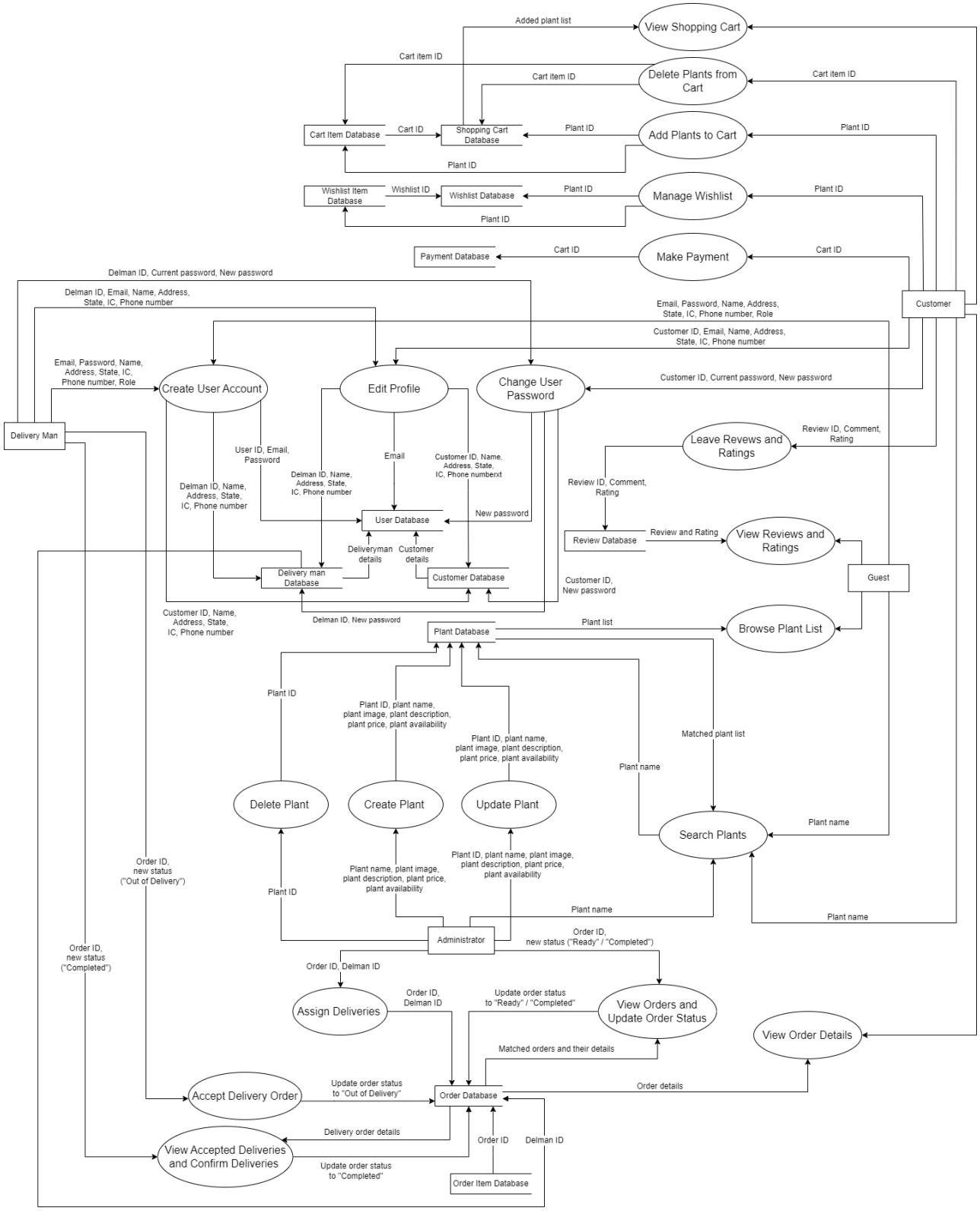


Figure 2.5: Data Flow Diagram for Nursery Plant Shopping System

3 Design

3.1 Use Cases

3.1.1 A delivery man creates a user account.

Table 3.1: A Delivery Man Creates A User Account Use Case Description

Use Case Name:	A delivery man creates a user account.
Description:	A delivery man begins by clicking the “Sign Up” button and is directed to the “Register” page. The delivery man fills out the registration form with their personal details. Upon clicking the “Submit” button, the system validates the input and saves the new user details to persistent storage. A confirmation message is then displayed, indicating successful account creation, and the delivery man now has access to the system with their new credentials.
Primary Actor:	Delivery Man
Precondition	• The delivery man is not logged into the system.
Postcondition	• The delivery man is successfully registered and has a new user account in the system. The provided information (full name, password, email, address, state, IC No, phone number, and role) is stored in persistent storage, and the delivery man can now log in to the system using their newly created credentials.
Main Success Scenario:	S01: The delivery man navigates to the login page of the website. S02: On the login page, the delivery man clicks on the "Sign Up" button. S03: The system displays a registration form with fields such as full name, password, email, address, state, IC No, phone number, and role. S04: The delivery man fills up the fields (full name, password, email, address, state, IC No, phone number) accordingly. S05: The delivery man chooses the role “Delivery Man” via a dropdown menu. S06: The delivery man clicks the “Submit” button. S07: The system validates the entered information. S08: The system displays a confirmation message, which shows the successful creation of the user account.

Alternative Scenario:	-
Exception Scenario:	A01: @S07 The system detects the invalid email format. A01.08: The system displays an error message next to the email field, prompting the delivery man to correct the information. A01.09: The delivery man corrects the information. A01.10: RESUME @S05

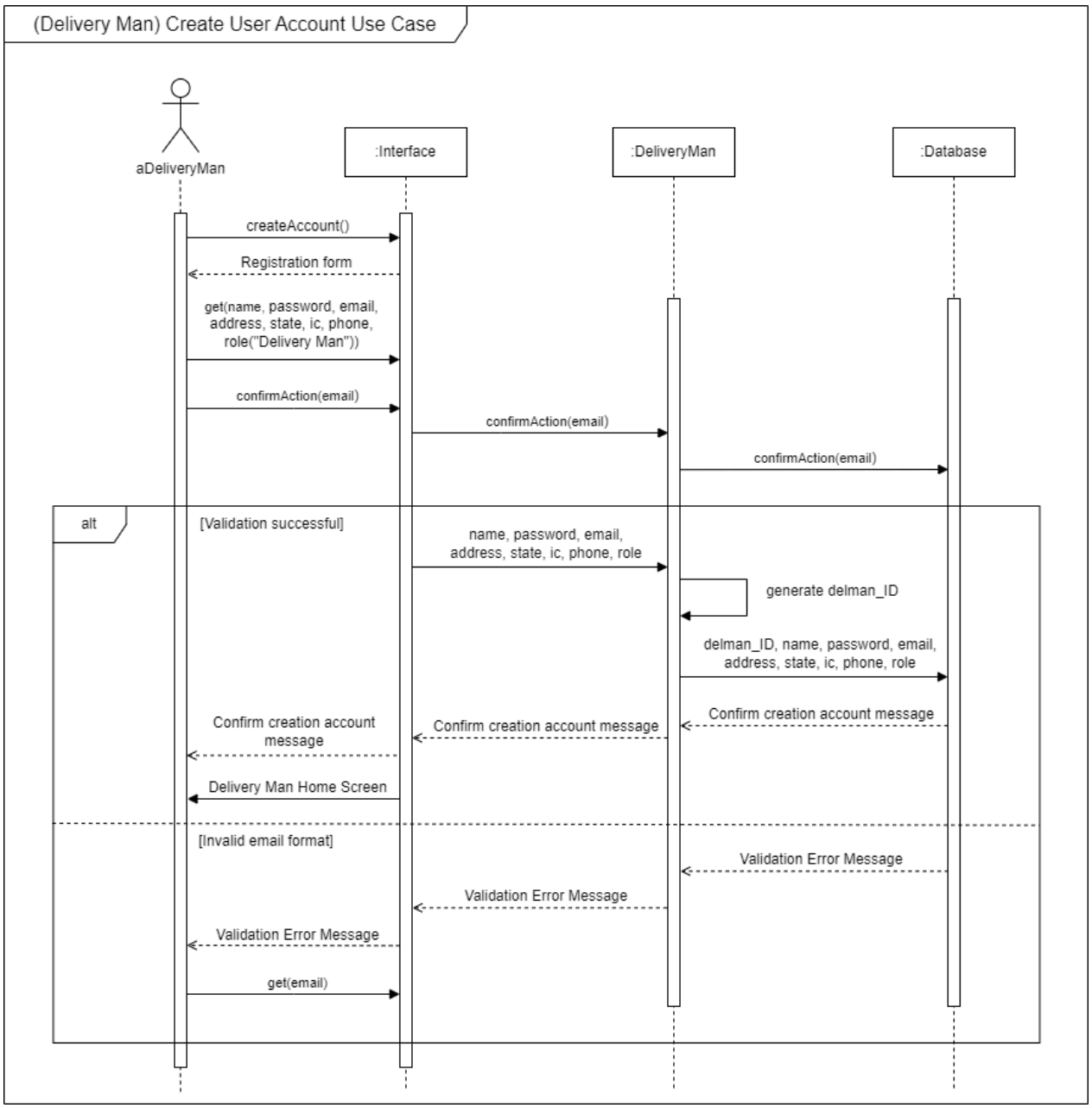


Figure 3.1: Create User Account Sequence Diagram

3.1.2 A delivery man changes their password.

Table 3.2: A Delivery Man Changes Their Password Use Case Description

Use Case Name:	A delivery man changes their password.
Description:	The registered delivery man can change his or her password. The system validates the entered current password and ensures that the new password is distinct. Upon successful validation, the system updates the delivery man's account with the new password, storing it securely in the database. In case of validation failures, error messages prompt corrections.
Primary Actor:	Delivery Man
Precondition	● The delivery man is logged into the system. ● The delivery man has the necessary permissions to change their own password.
Postcondition	● The delivery man's password is successfully changed, and the delivery man can continue using the system with the new password.
Main Success Scenario:	S01: The delivery man clicks on the "Account Settings" section within the main system interface. S02: The system displays account settings, including detailed information such as full name, email, phone number, address, state, and IC No, along with the "Change Password" button. S03: The delivery man clicks on the "Change Password" button. S04: The system displays fields for entering the current password and the new password. S05: The delivery man enters the current password in the "Current Password" field. S06: The delivery man enters the new password in the "New Password" field. S07: The delivery man clicks the "Confirm" button. S08: The system checks if the entered current password matches the stored password for the delivery man's account. S09: If the old password validation succeeds, the system checks if the new password is different from the entered current password. S10: If the new password validation succeeds, the system updates the delivery man's account with the new password and stores the new password in the database.

	S11: The system displays a confirmation message, indicating the successful password change.
Alternative Scenario:	-
Exception Scenario:	A01: @S07 The old password validation fails. A01.08: The system displays an error message indicating that the entered current password is incorrect. A01.09: The delivery man corrects the current password. A01.10: RESUME @S05 A02: @S08 The new password validation fails. A02.09: The system displays an error message, which shows the new password is the same as the entered current password. A02.10: The delivery man corrects the new password. A02.11: RESUME @S06

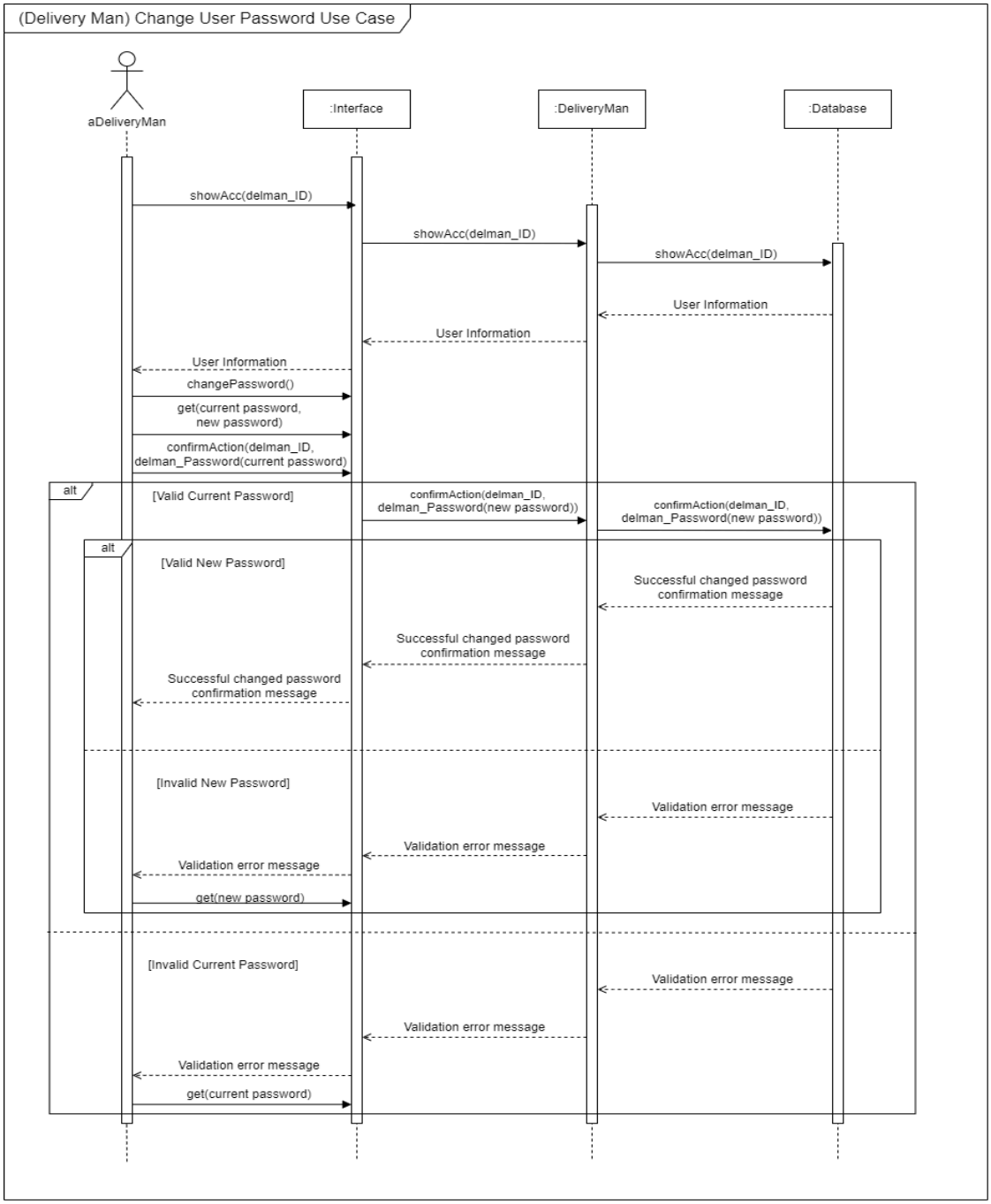


Figure 3.2: Change User Password Sequence Diagram

3.1.3 A delivery man can edit profile details.

Table 3.3: A Delivery Man Can Edit Profile Details Use Case Description

Use Case Name:	A delivery man can edit profile details
Description:	A delivery man can update their profile information by navigating to the 'Account Settings' section, clicking 'Edit Profile', and modifying details such as full name, email, phone number, address, state, and IC number. After confirming changes with the "Confirm" button, the system updates the database and notifies the delivery man of the successful profile update.
Primary Actor:	Delivery Man
Precondition	● The delivery man is logged into the system. ● The delivery man desires to change their profile details.
Postcondition	● The delivery man's profile details have been successfully updated in the system.
Main Success Scenario:	S01: The delivery man navigates to the "Account Settings" section within the system. S02: The system displays account settings, including detailed information such as full name, email, phone number, address, state, and IC No, along with the "Edit Profile" button. S03: The delivery man clicks on the "Edit Profile" button. S04: The system displays a form with editable fields for the delivery man's personal information. S05: The delivery man modifies the necessary information in the fields provided. S06: The delivery man submits the updates by clicking the "Confirm" button. S07: The system validates and saves the updated information. S08: The system confirms the updates with a message to the customer.
Alternative Scenario:	A01: @S04 The delivery man decides to cancel the update. A01.05: The delivery man clicks the "Back" button without saving changes.

	A01.06: The delivery man is then redirected back to the Account Settings page.
Exception Scenario:	A02: @S07 The system detects the invalid email format. A02.08: The system displays an error message next to the email field, prompting the delivery man to correct the information. A02.09: The delivery man corrects the information. A02.10: RESUME @S06.

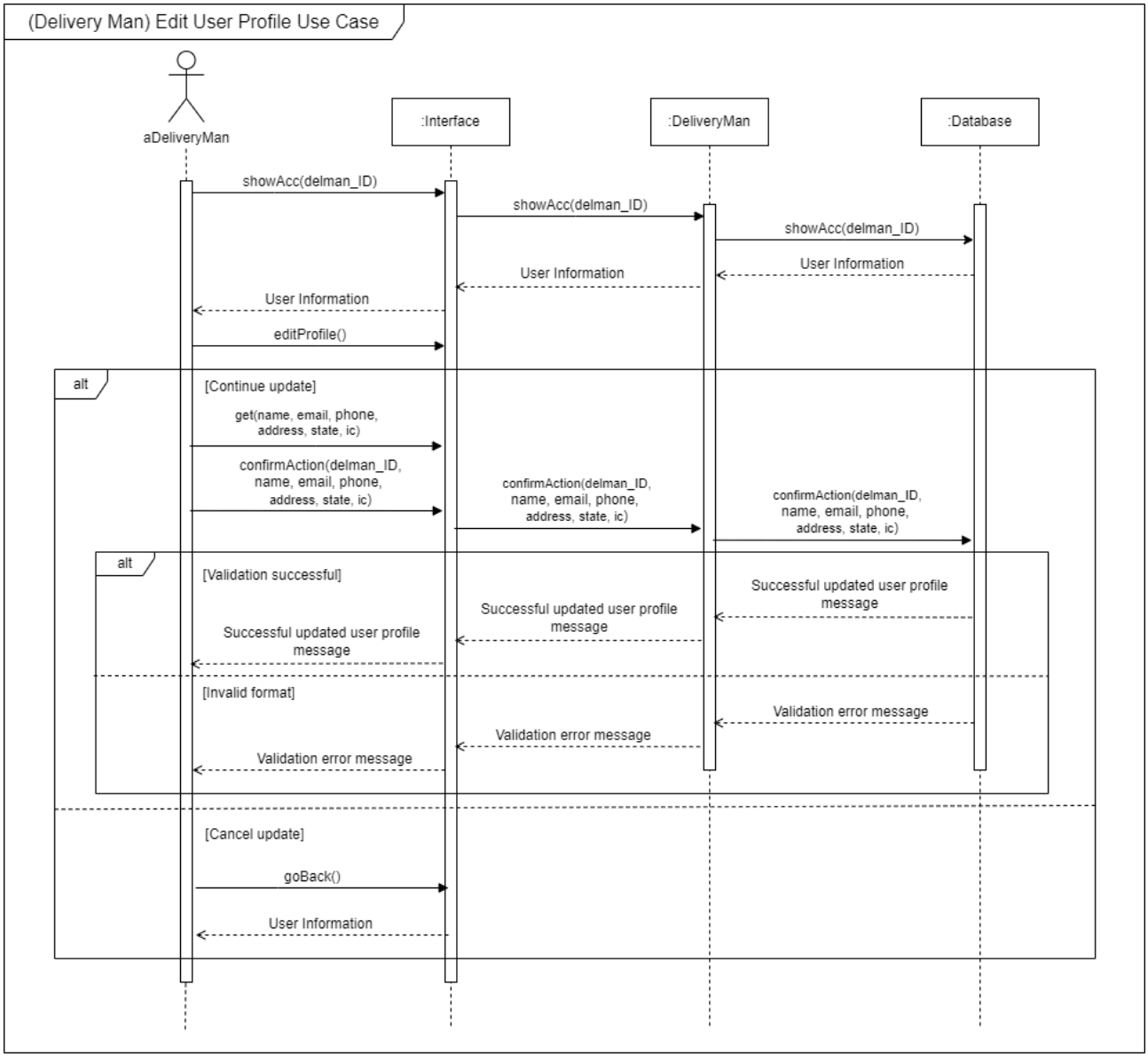


Figure 3.3: Edit User Profile Sequence Diagram

3.1.4 A delivery man accepts a delivery order.

Table 3.4: A Delivery Man Accepts A Delivery Order Use Case Description

Use Case Name:	A delivery man accepts a delivery order.
Description:	The Delivery Man can accept a pending delivery order that is assigned by the admin. In the event that no pending delivery orders are available during the search, the system communicates this to the Delivery Man, who can check back later for new orders.
Primary Actor:	Delivery Man
Precondition	• The Delivery Man is logged into the system.
Postcondition	• The Delivery Man has successfully accepted the delivery order, and the order status is now "Out for Delivery."
Main Success Scenario:	S01: The Delivery Man navigates to the "Pending Orders" section within the system. S02: The system displays a list of pending delivery orders that are assigned by admin with details such as delivery order ID, customer name, customer ID, phone number, and delivery address. S03: The Delivery Man chooses a specific delivery order that they wish to accept. S04: The Delivery Man clicks the "Accept" button. S05: The system updates the order status to "Out For Delivery" and stores the delivery status in the database.
Alternative Scenario:	-
Exception Scenario:	A01: @S02 There are no pending delivery orders to be accepted. A01.03: The system displays a message indicating that no pending delivery orders are available to the Delivery Man. A01.04: The Delivery Man may check back later. A02: @S04 The Delivery Man chooses to reject the assigned delivery order. A02.05: The Delivery Man clicks the "Reject" button. A02.06: The system removes the delivery order from the list of pending orders. A02.07: END

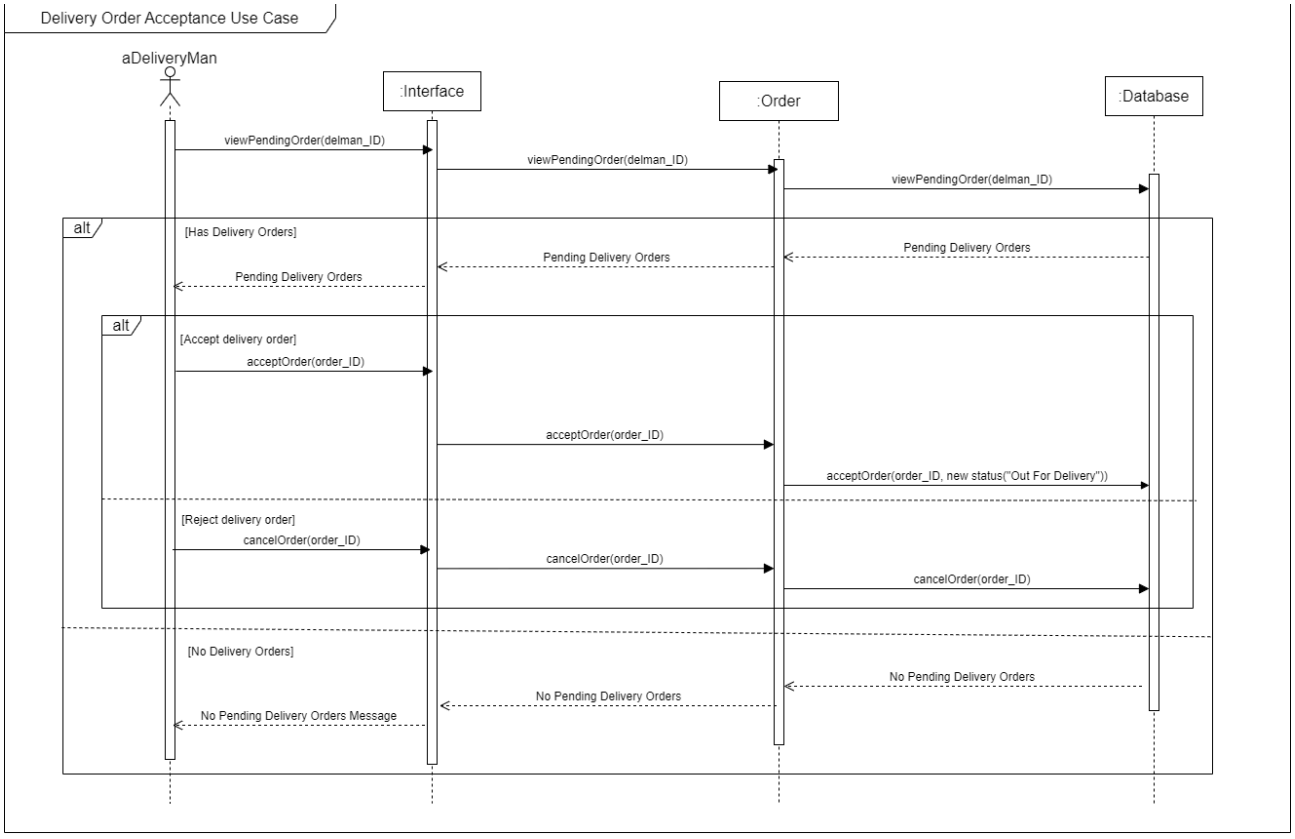


Figure 3.4: Delivery Order Acceptance Sequence Diagram

3.1.5 A delivery man views accepted deliveries and confirms delivery.

Table 3.5: A Delivery Man Views and Confirms Delivery Use Case Description

Use Case Name:	A delivery man views accepted deliveries and confirms delivery.
Description:	The Delivery Man can view a list of accepted plant deliveries. The Delivery Man can also confirm successful deliveries by updating the delivery status to “Completed”. In the absence of accepted deliveries during the search, the system communicates this to the Delivery Man, who may check back later for new assignments.
Primary Actor:	Delivery Man
Precondition	• The Delivery Man is logged into the system.
Postcondition	• The Delivery Man has successfully viewed the details of accepted plant deliveries and confirmed the successful delivery. The system then reflects the updated status in the database.

Main Success Scenario:	S01: The Delivery Man navigates to the "Accepted Deliveries" section within the system. S02: The system displays a list of accepted plant deliveries with details such as delivery order ID, customer name, phone number, and delivery address. S03: After successfully delivering an order, the Delivery Man chooses a delivery order and clicks the "Confirm Delivery" button. S04: The system updates the status of the selected delivery to "Completed" and stores the delivery status in the database.
Alternative Scenario:	-
Exception Scenario:	A01: @S02 No accepted deliveries are found. A01.03: The system displays a message indicating that no deliveries are currently accepted to the Delivery Man. A01.04: The Delivery Man may check back later.

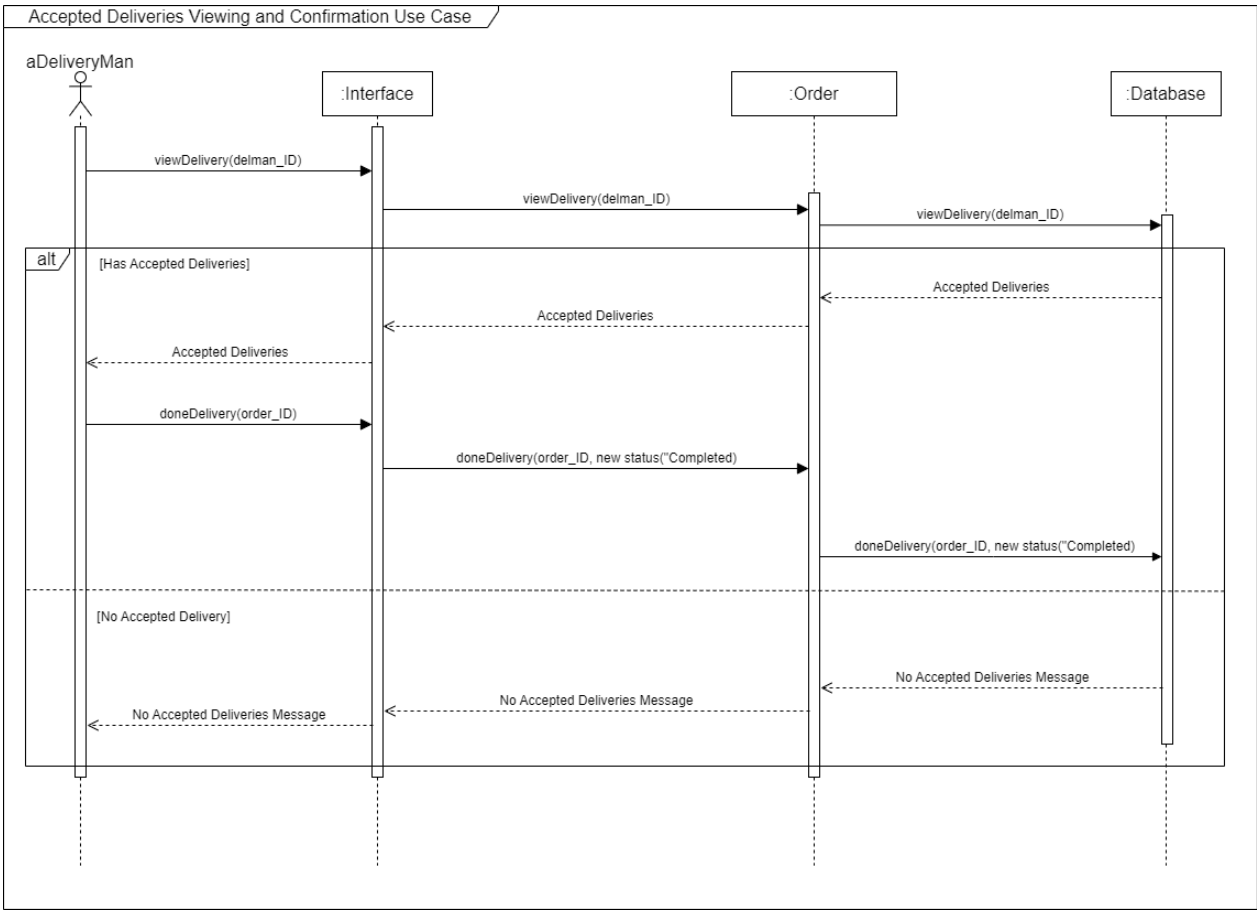


Figure 3.5: Accepted Delivery Viewing and Confirmation Sequence Diagram

3.2 Data Dictionary

The User Table Data Dictionary, shown in Table 3.6, is a key component for organizing user information in the Nursery Plant Shopping System. It details essential attributes like user unique identifier(user_ID), which uniquely identifies each user, user email (email), and user password (password) for login credentials, and role-defining booleans (is_customer, is_deliveryman, and is_admin) to distinguish between customers, delivery personnel, and administrators. The user_ID serves as the primary key, ensuring that each record is unique and easily retrievable.

Table 3.6: User Table Data Dictionary

Table Name	Attribute Name	Contents	Type	Format	Range	Required	PK/FK	FK Referenced Table
User	user_ID	user ID	int	1	1-9999	Y	PK	-
	email	user email	varchar(100)	eelin@gmail.com	-	Y	-	-
	password	user password	varchar(255)	123456	-	Y	-	-
	is_customer	is customer	boolean	True	-	Y	-	-
	is_deliveryman	is deliveryman	boolean	True	-	Y	-	-
	is_admin	is administrator	boolean	True	-	Y	-	-

The DeliveryMan Table Data Dictionary shown as Table 3.7 serves as a comprehensive reference guide for managing delivery man records within the database. It contains attributes such as the delivery man's unique identifier (delman_ID), their name (delman_Name), address (delman_Address), state (delman_State), identification card number (delman_IC), and phone number (delman_phone_number). These attributes are crucial for tracking and managing delivery personnel within the system. The data dictionary also designates delman_ID as the primary key (PK), facilitating efficient record referencing within the table.

Table 3.7: Delivery Man Table Data Dictionary

Table Name	Attribute Name	Contents	Type	Format	Range	Required	PK/FK	FK Referenced Table
DeliveryMan	delman_ID	delivery man ID	int	90006	90006-90999	Y	PK	-
	delman_Name	delivery man name	varchar(80)	Siti Fatimah	-	Y	-	-
	delman_Address	delivery man address	varchar(200)	35, Jalan BPU6 Taman Indah, Puchong	-	Y	-	-
	delman_State	delivery man state	varchar(30)	Selangor	-	Y	-	-
	delman_IC	delivery man's identification card number	varchar(30)	920504018888	-	Y	-	-
	delman_phone_number	delivery man phone number	bigInt	60147894163	-	Y	-	-

The Customer Table Data Dictionary shown as Table 3.8 provides a comprehensive overview of the attributes and specifications for managing customer records within the database. It includes essential details such as the customer's unique identifier (customer_ID), their name (customer_Name), address (customer_Address), state (customer_State), identification card number (customer_IC), and phone number (customer_phone_number). These attributes are essential for customer account management and interactions within the system. The data dictionary also identifies customer_ID as the primary key (PK) for efficient record referencing within the table.

Table 3.8: Customer Table Data Dictionary

Table Name	Attribute Name	Contents	Type	Format	Range	Required	PK/ FK	FK Referenced Table
Customer	customer_ID	customer ID	int	91000	91000-94999	Y	PK	-
	customer_Name	customer name	varchar(80)	Chin Yu Feng	-	Y	-	-
	customer_Address	customer address	varchar(200)	20, Taman Maju, Section 3/2a, Cheras	-	Y	-	-
	customer_State	customer state	varchar(30)	Selangor	-	Y	-	-
	customer_IC	customer's identification card number	varchar(30)	040202016450	-	Y	-	-
	customer_phone_number	customer phone number	bigInt	60128457865	-	Y	-	-

The Payment Table Data Dictionary shown as Table 3.9 provides an overview for managing payment transactions within the system. It contains attributes such as the payment's unique identifier (payment_ID), the associated customer's identifier (customer_ID), shipping fee (shipping_Fee), total cost for the payment (payment_Cost), the method for receiving the order (receive_Method), the date of order placement (order_Date), and the order's total price (order_total_price). These attributes are crucial for tracking and managing payment-related information and order details within the system. The data dictionary designates payment_ID as the primary key (PK), and identifies customer_ID as a foreign key (FK) referencing the "Customer" table, creating a direct link between payments and individual customers.

Table 3.9: Payment Table Data Dictionary

Table Name	Attribute Name	Contents	Type	Format	Range	Required	PK/FK	FK Referenced Table
Payment	payment_ID	payment ID	int	60000	60000-69999	Y	PK	-
	customer_ID	customer ID	int	91000	91000-94999	Y	FK	Customer
	shipping_Fee	shipping fee	decimal(4,2)	5.00	0.00-99.99	Y	-	-
	payment_Cost	total cost for payment	decimal(6,2)	200.00	0.00-9999.99	Y	-	-
	receive_Method	method to receive order	varchar(10)	Delivery	-	Y	-	-
	order_Date	date of order placement	date	YYYY-MM-DD	-	Y	-	-
	order_total_price	order total price	decimal(6,2)	200.00	0.00-9999.99	Y	-	-

The Order Table Data Dictionary shown as Table 3.10 serves as a reference guide for managing customer orders within the system. It encompasses essential attributes such as the order's unique identifier (order_ID), the associated customer's identifier (customer_ID), payment identifier (payment_ID), delivery man identifier (delman_ID), and the order status (order_Status). These attributes are fundamental for tracking and managing customer orders and related order details. The data dictionary designates order_ID as the primary key (PK), and identifies customer_ID as a foreign key (FK) referencing the “Customer” table, creating a direct link between orders and individual customers. Similarly, payment_ID and delman_ID are foreign keys (FK) referencing the “Payment” and “DeliveryMan” tables, respectively, establishing connections with payment and delivery information.

Table 3.10: Order Table Data Dictionary

Table Name	Attribute Name	Contents	Type	Format	Range	Required	PK/FK	FK Referenced Table
Order	order_ID	order ID	int	70000	70000-79999	Y	PK	-
	customer_ID	customer ID	int	91000	91000-94999	Y	FK	Customer
	payment_ID	payment ID	int	60000	60000-69999	Y	FK	Payment
	delman_ID	delivery man ID	int	90006	90006-90999	Y	FK	DeliveryMan
	order_Status	order status	varchar(30)	Ready	-	Y	-	-

3.3 Data Structures

3.3.1 User Array

Table 3.11 presents the User Array Data Structures, a fundamental element for managing user data within the Nursery Plant Shopping System. This array holds critical information for distinguishing between different user roles and their credentials. The user ID is an integer that uniquely identifies each user. The user email and user password are stored as varchar(100) and varchar(255) specifically for security and authentication purposes. The boolean flags (is customer, is deliveryman, and is administrator) are to specify the user's role within the system. These attributes ensure that user data is neatly categorized and accessible, simplifying the process of identifying and managing users' interactions and responsibilities.

Table 3.11: User Array Data Structures

Attribute	Data Type
user ID	int
user email	varchar(100)
user password	varchar(255)
is customer	boolean
is deliveryman	boolean
is administrator	boolean

3.3.2 DeliveryMan Array

Table 3.12 presents the Delivery Man Array Data Structures. The Delivery Man array is used for efficient management of essential data regarding delivery personnel within the system. The Delivery Man ID, represented as an integer, assigns a distinctive identifier to each delivery man. Delivery Man Name, as a varchar(80), facilitates easy user identification and improves the presentation of their profiles. Moreover, Delivery Man Address and Delivery Man State, both varchars (200) and (30) respectively, collect contact details and location information, enriching the understanding of delivery personnel's operational context. The Delivery Man’s Identification Card Number, a varchar(30), stores unique identification data crucial for comprehensive identity verification. Additionally, the use of the bigInt data type for the Delivery Man Phone Number accommodates the storage of contact information, ensuring efficient communication within the delivery team.

Table 3.12: Delivery Man Array Data Structures

Attribute	Data Type
Delivery man ID	int
Delivery man name	varchar(80)
Delivery man address	varchar(200)
Delivery man state	varchar(30)
Delivery man's identification card number	varchar(30)
Delivery man phone number	bigInt

3.3.3 Customer Array

Table 3.13 below shows Customer Array Data Structures. The Customer array serves as a fundamental data structure for managing crucial customer information within a system. The Customer ID, represented as an integer, acts as a unique identifier for each customer. Customer Name, a varchar(80), stores customer names, enhancing user recognition and presentation. Customer Email, a varchar(100), captures email addresses needed for user account validation. The Customer Password, a varchar(255), used for user authentication. Customer Address and Customer State, both varchars (200) and (30) respectively, record customer contact details and location information for order delivery. The Customer's Identification Card Number, a varchar(30), stores unique identification data, and the bigInt data type is used for the Customer Phone Number, enabling the storage of contact numbers.

Table 3.13: Customer Array Data Structures

Attribute	Data Type
Customer ID	int
Customer Name	varchar(80)
Customer Email	varchar(100)
Customer Password	varchar(255)
Customer address	varchar(200)
Customer state	varchar(30)
Customer's identification card number	varchar(30)
Customer phone number	bigInt

3.3.4 Payment Array

Table 3.14 presents the Payment Array Data Structures. The Payment array serves as a vital data structure for managing payment transactions within the system. Payment ID, an integer, assigns a unique identifier to each payment, ensuring precise tracking and reference. Customer ID, also an integer, links each payment to a specific customer, facilitating accurate association with corresponding customers. Shipping Fee, formatted as a decimal(4,2), captures the cost of shipping associated with the payment. Total Cost for Payment, a decimal(6,2), represents the overall cost encompassing the order and shipping fees. Method to Receive Order, a varchar(10), records the chosen method for receiving the order. Date of Order Placement, in date format, denotes the precise date when the order was placed, aiding in order scheduling. Order Total Price, formatted as a decimal(6,2), represents the total cost of the order, including products and excluding shipping fee.

Table 3.14: Payment Array Data Structures

Attribute	Data Type
Payment ID	int
Customer ID	int
Shipping fee	decimal(4,2)
Total cost for payment	decimal(6,2)
Method to receive order	varchar(10)
Date of order placement	date
Order total price	decimal(6,2)

3.3.5 Order Array

Table 3.15 presents the Order Array Data Structures. The Order array serves as a pivotal data structure for managing customer orders within the system. Order ID, represented as an integer, assigns a unique identifier to each order, facilitating precise tracking and reference. Customer ID, also an integer, links each order to a specific customer, ensuring accurate association with individual users. Payment ID, an integer, connects each order to a corresponding payment. Delivery Man ID, an integer, associates orders with specific delivery personnel. Order Status, a varchar(30), captures the current status of each order, which can be “waiting”, “ready”, “out of delivery” or “completed”.

Table 3.15: Order Array Data Structures

Attribute	Data Type
Order ID	int
Customer ID	int
Payment ID	int
Delivery man ID	int
Order status	varchar(30)

3.4 Subsystem Architecture

Figure 3.6 below is the Component-based Architecture Diagram for the Delivery Man Subsystem. Figure 3.6 illustrates the functional areas tailored for the delivery personnel. This subsystem allows a delivery man to manage their delivery operations effectively, with components for viewing “Pending Orders” and taking action to “Accept Delivery Order”. After an order has been accepted by the delivery man, that order will be transferred to the “Accepted Deliveries” list for active delivery tracking. The delivery man can “Confirm Delivery” once the plants reach the customers. Besides, personal account management is facilitated through “Account Settings”, which includes options to “Change Password” and “Edit Profile”, ensuring that delivery men can also keep their personal account information up-to-date. This targeted subsystem is used to streamline the delivery process and provide delivery men with the necessary tools to perform their roles efficiently in the plant shopping system.

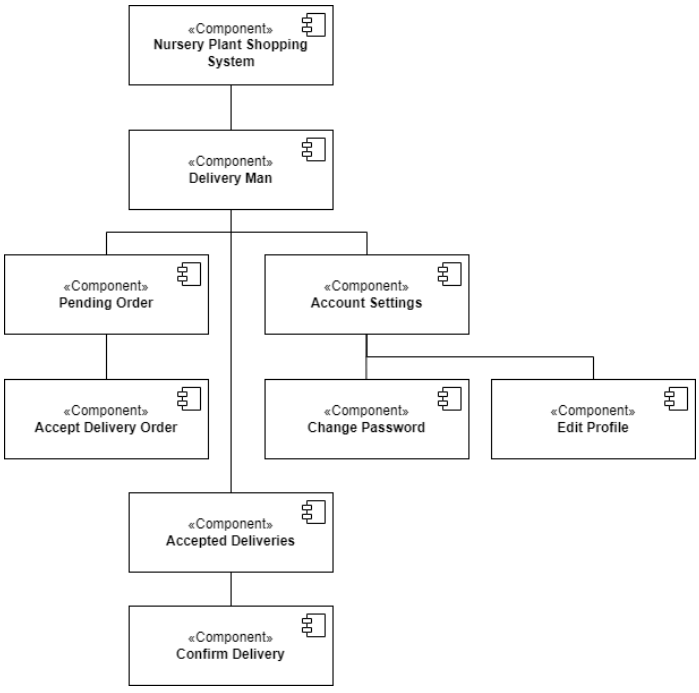


Figure 3.6: Component-based Architecture Diagram for Delivery Man Subsystem

3.5 Subsystem Screens

3.5.1 Delivery Man Home Screen

Figure 3.7 displays the "Delivery Man Home Screen" of the TLET Nursery Plant Shopping System. Upon login, the delivery man will be presented with a home screen shown in Figure 3.7 featuring a navigation bar with options for 'Pending Order', 'Accepted Deliveries', 'Account Settings', and a 'Logout' button. Below the navigation, a welcoming message marks the entry to the delivery man dashboard.

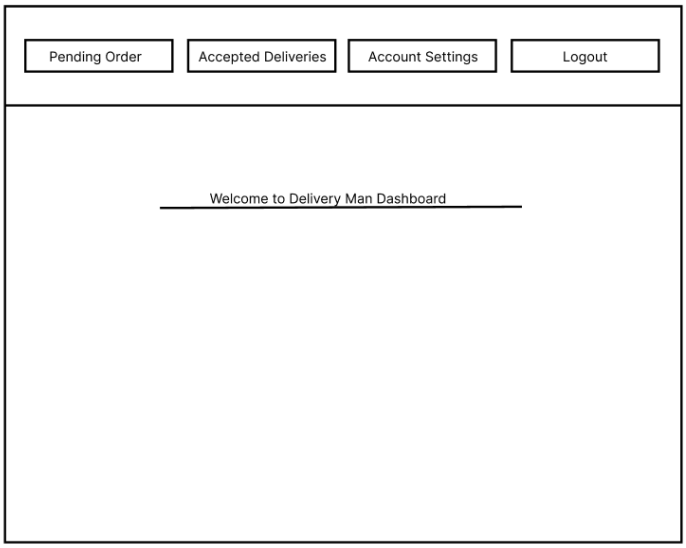


Figure 3.7: Delivery Man Home Screen

3.5.2 Delivery Man Creates a User Account Screen

Figure 3.9 shows the 'Create User Account Screen', which illustrates the process for delivery men to create a new account on the TLET Nursery Plant Shopping System. Upon accessing the system, the delivery man will first encounter the guest home screen in Figure 3.8, which is also the "TLET Nursery Plant Shopping System Screen" before creating a user account. The delivery man can proceed by clicking on the "Sign Up" button and this action takes the delivery man to the 'Creates User Account Screen' shown in Figure 3.9, where the delivery man fills out the form with personal and contact details. Upon clicking the “Submit” button, the system validates the input and saves the new user details to persistent storage. Successful registration is confirmed with a message. Delivery men who have an account can quickly get started by clicking the "Login" button, which takes them to a Login page in Figure 3.10.

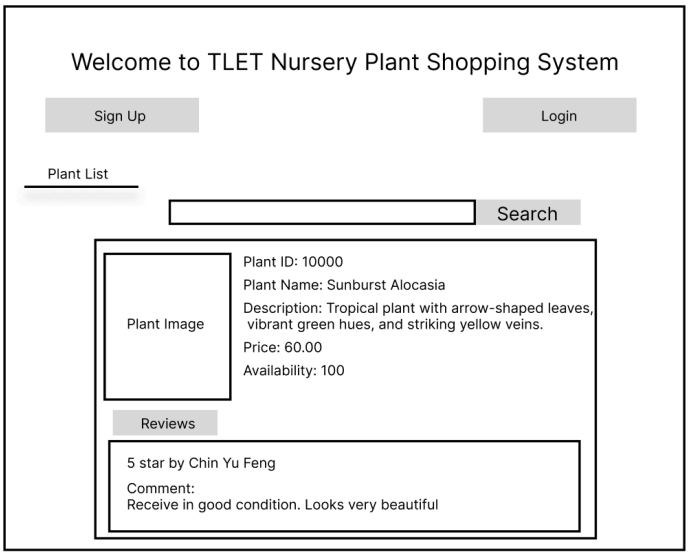


Figure 3.8: Guest Home Screen

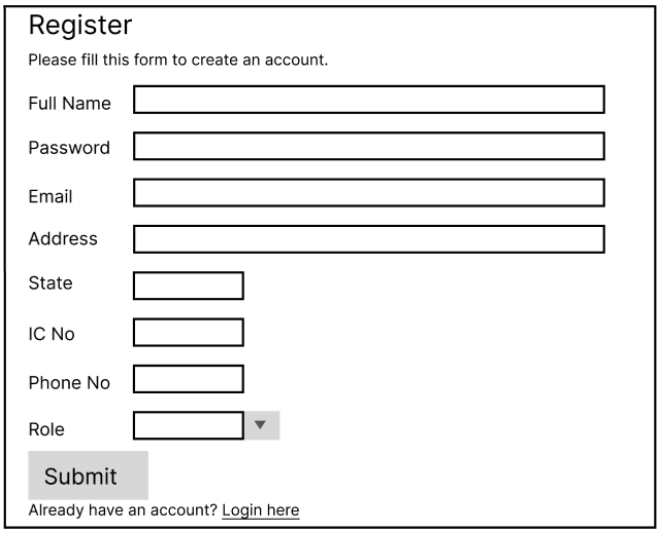


Figure 3.9:Create User Account Screen

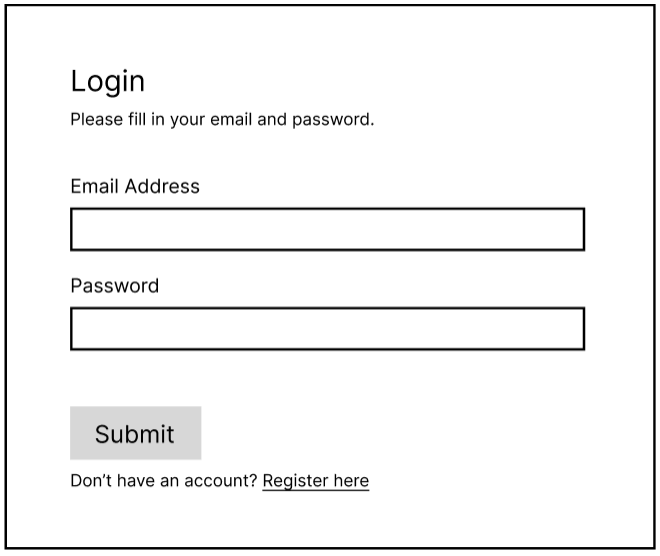


Figure 3.10: TLET Nursery Plant Shopping System Login Screen

3.5.3 Delivery Changes Their Password Screen

Figure 3.12 shows the "Change Password Screen", which presents the interface for delivery man to update their password on the TLET Nursery Plant Shopping System. The process is initiated from the “Delivery Man Home Screen” as shown in Figure 3.7, where the delivery man selects the 'Account Settings' button. This action takes them to the "Account Settings Screen" displayed in Figure 3.11, where the delivery man details are listed. By selecting the 'Change Password' button, the delivery man is directed to the screen shown in Figure 3.12, where they are required to enter their current password followed by the new password they wish to set. After submitting the new password by clicking 'Confirm', the system verifies the new password. If the update is successful, a confirmation message is displayed, and the delivery man is taken back to the "Account Settings Screen" shown in Figure 3.11, completing the password change process.

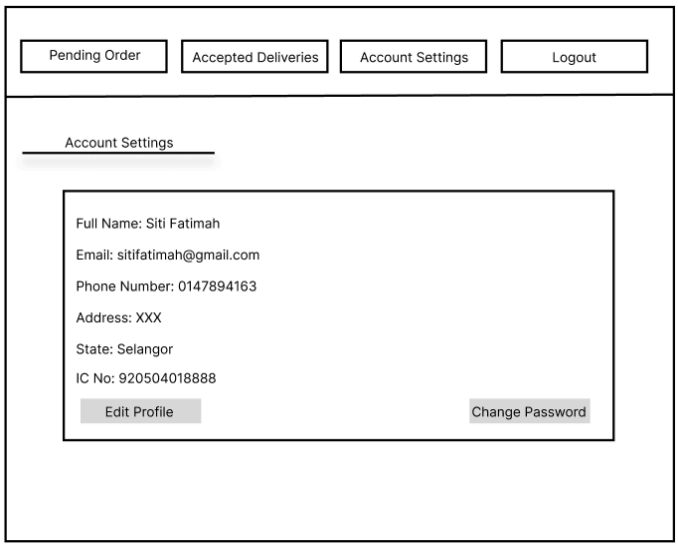


Figure 3.11: Account Settings Screen

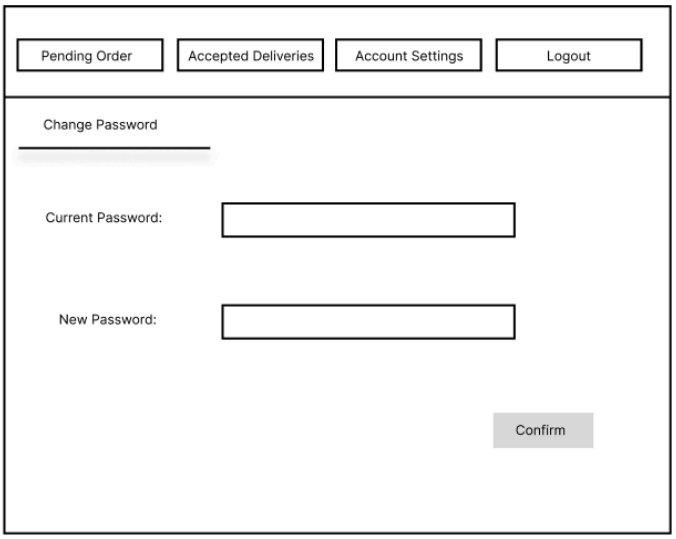


Figure 3.12: Change Password Screen

3.5.4 Delivery Man Can Edit Profile Details Screen

Figure 3.13 shows 'Edit Profile Details Screen', where a delivery man can update their personal information on the TLET Nursery Plant Shopping System. The process is initiated from the "Delivery Man Home Screen" as shown in Figure 3.7, where the delivery man selects the 'Account Settings' button. This action takes them to the "Account Settings Screen" displayed in Figure 3.11, where the delivery man's details are listed. By selecting the 'Edit Profile' button, the delivery man is directed to the screen shown in Figure 3.13, featuring fields to modify the delivery man's personal information. Changes are submitted with the 'Confirm' button, and the system displays a confirmation message. If the delivery man chooses not to update the personal information, clicking the 'Back' button redirects them to the “Account Settings Screen” in Figure 3.11, without making any updates.

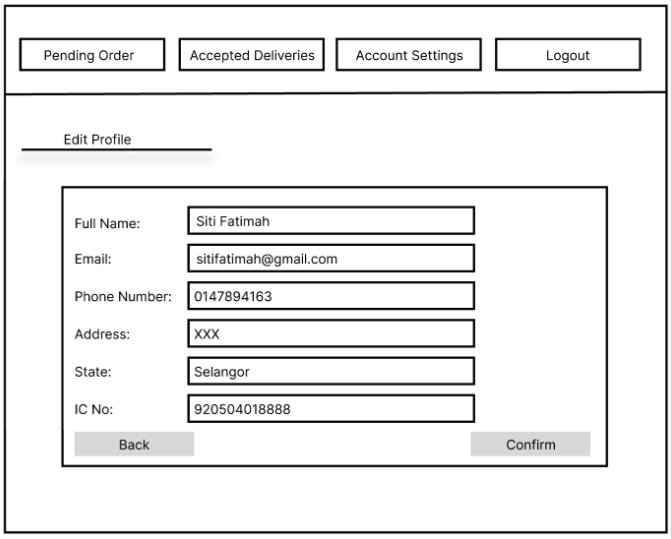


Figure 3.13: Edit Profile Screen

3.5.5 Delivery Man Accepts a Delivery Order Screen

Figure 3.14 shows the "Accepts a Delivery Order Screen" from the TLET Nursery Plant Shopping System, used by the delivery man to manage incoming orders. The process is initiated from the "Delivery Man Home Screen" as shown in Figure 3.7, where the delivery man selects the “Pending Orders” button. This action takes them to the "Accepts a Delivery Order Screen" displayed in Figure 3.14. On this screen, the delivery man can view all pending orders with detailed information including the Delivery Order ID, customer names, customer IDs, contact numbers, and delivery addresses. They have the option to either accept or reject these orders directly from this interface. Once an order is accepted, it is marked as out of delivery. If the Delivery Man clicks the "Reject" button, the system removes the delivery order from the list of pending orders.

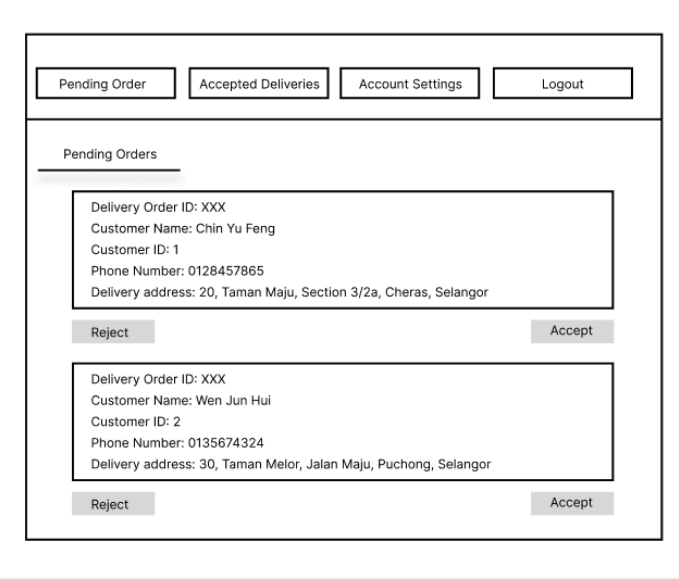


Figure 3.14: Accepts a Delivery Order Screen

3.5.6 Delivery Views Accepted Deliveries and Confirm Delivery Screen

Figure 3.15 presents the "Accepts a Deliveries and Confirm Delivery Screen" of the TLET Nursery Plant Shopping System. The process is initiated from the "Delivery Man Home Screen" as shown in Figure 3.7, where the delivery man selects the 'Accepted Deliveries' button. This action takes them to the "Accepts a Deliveries and Confirm Delivery Screen" displayed in Figure 3.15 which allows delivery men to view a list of all plant deliveries they have accepted, complete with details such as delivery order ID, customer names, phone numbers, and delivery addresses. After the successful delivery of an order, the delivery man can confirm the completion of delivery by clicking the "Confirm Delivery" button provided alongside each order. This confirmation updates the order's status to "Completed" in the system's database, ensuring that the delivery process is accurately tracked.

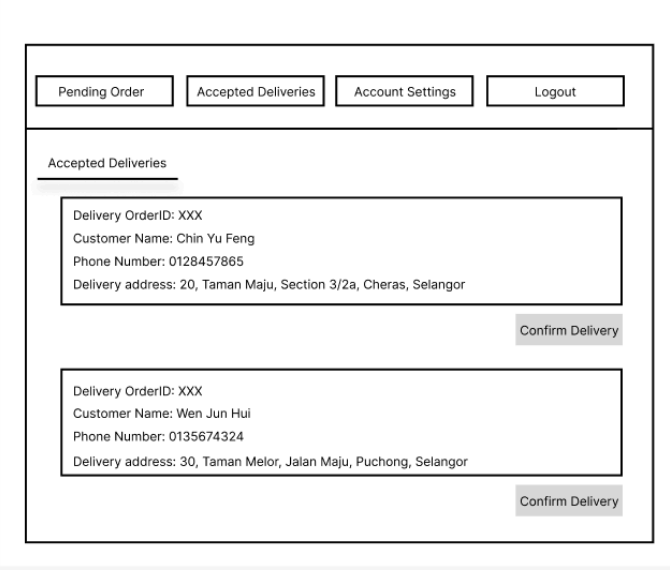


Figure 3.15: Accepts a Deliveries and Confirm Delivery Screen

3.6 Subsystem Components

Table 3.16 shows the use cases of the delivery man within the system. The subsystem dedicated to the Delivery Man encompasses modules for account management, profile customization, and delivery processing. Integration is seamless, allowing the delivery personnel to create and manage their user accounts, change passwords, and edit personal profile details for accurate system representation. Additionally, the system equips them with functionalities to accept delivery orders and track these assignments. After completing deliveries, they can view and confirm them in the system, which closes the loop of the delivery process and ensures accountability and efficiency in the subsystem's operation.

Table 3.16: Actor and Respective Use Cases

Actor	Use Cases
Delivery Man	A delivery man creates a user account.
	A delivery man changes their password.
	A delivery man can edit profile details.
	A delivery man accepts a delivery order.
	A delivery man views accepted deliveries and confirms delivery.

3.6.1 A delivery man creates a user account.

Figure 3.16 illustrates the activity diagram where a delivery man begins the account creation process by clicking the "Sign Up" button within the TLET Nursery Shopping System. It then leads to a registration form where the delivery man must provide personal details such as full name, password, email, address, state, IC number, phone number, and select a role. The delivery man fills in these details and specifically chooses the "Delivery Man" role from a dropdown menu. After submitting the form, the system checks the validity of the email format. If the email format is incorrect, an error message prompts the delivery man to make the necessary corrections. Once the information is validated, a confirmation message is displayed, signifying the successful creation of the delivery man's user account, and the information is stored in the system's database.

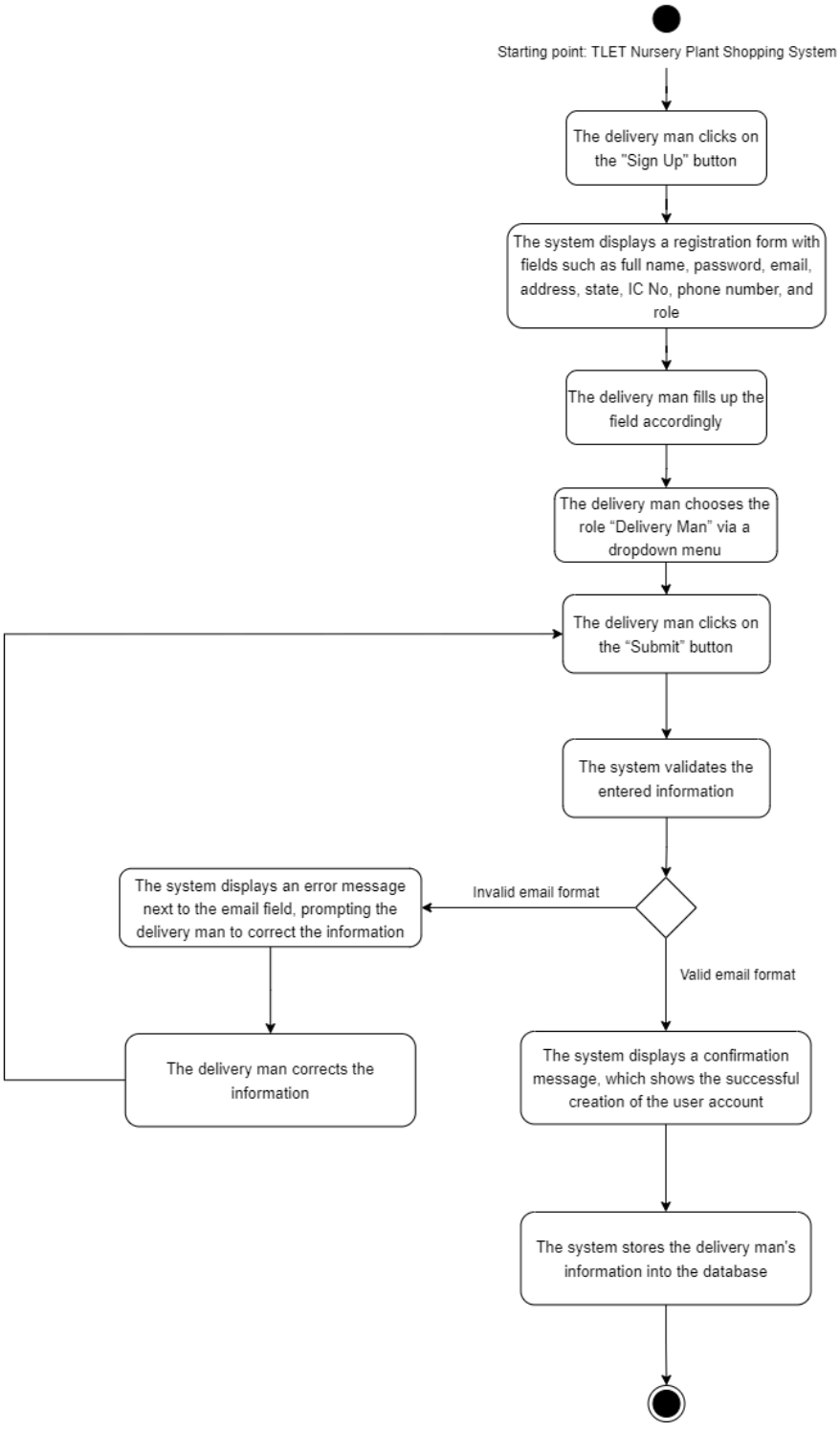


Figure 3.16: Create User Account Activity Diagram

3.6.2 A delivery man changes their password.

Figure 3.17 illustrates the activity diagram when a delivery man updates his/her password within the TLET Nursery Plant Shopping System. Beginning at the Delivery Man home screen, the delivery man navigates to the "Account Settings" section and selects the "Change Password" button. The system then displays the "Change Password" page, which prompts the delivery man to input his current password and the new password he wishes to set. Upon entering both passwords and clicking the "Confirm" button, the system verifies if the entered current password matches the one that is stored on the database. If it doesn't match, an error message is displayed, and the delivery man must correct the current password. After the system verifies the current password is correct, it then checks to ensure the new password is different from the current password. If the new password is the same as the current one, an error message appears, prompting a correction. If the new password is indeed different and meets the system's criteria, the system proceeds to update the account with the new password and securely stores it in the database.

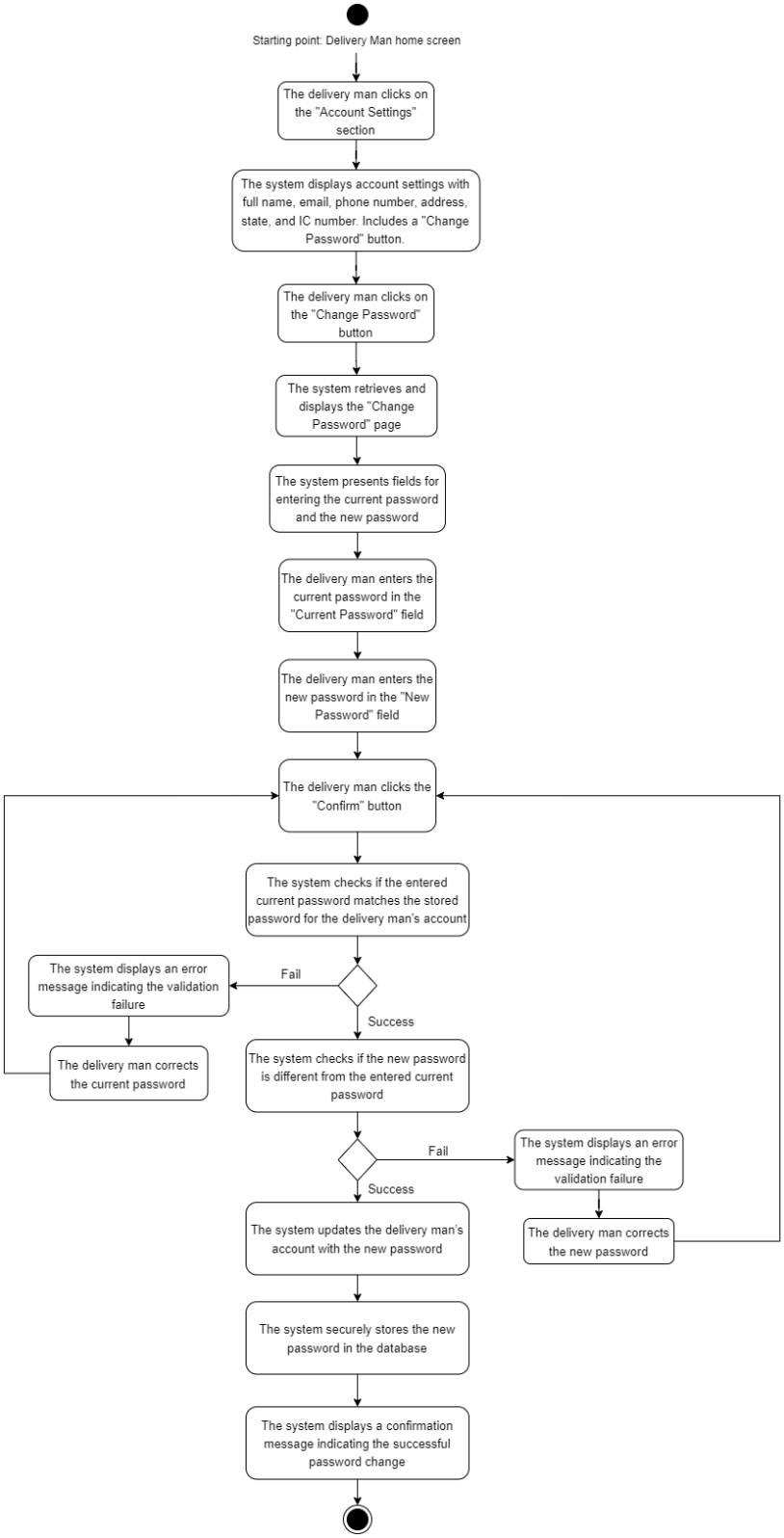


Figure 3.17: Change User Password Activity Diagram

3.6.3 A delivery man can edit profile details.

Figure 3.18 illustrates the activity diagram when a delivery man updates personal information in the TLET Nursery Plant Shopping System. Starting from the Delivery Man home screen, the delivery man selects the "Account Settings" section and then clicks on the "Edit Profile" button. The system displays the "Edit Profile" page with editable fields, allowing the delivery man to change details such as full name, email, phone number, address, state, and IC number. If the delivery man chooses to proceed, he/she updates the necessary fields and clicks "Confirm" to submit the changes. The system then validates the format for the email address. If an error is detected, the system prompts for the correction. Upon successful validation, the system saves and stores the updated information in the database. The delivery man will receive a confirmation message from the system, affirming the successful update of his/her profile. If the delivery man decides to cancel the update, he/she can click the "Back" button to return to the "Account Settings" screen.

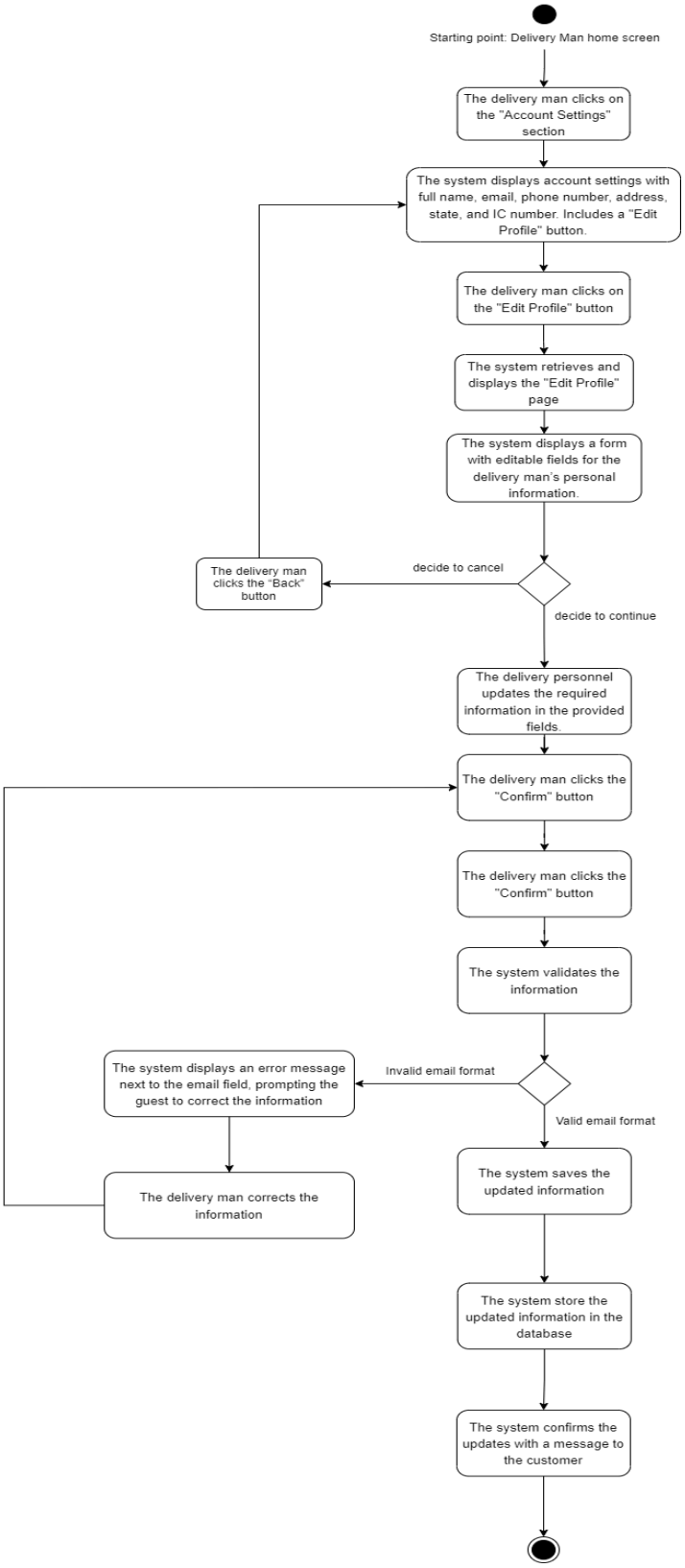


Figure 3.18: Edit User Profile Activity Diagram

3.6.4 A delivery man accepts a delivery order.

Figure 3.19 shows the activity diagram when a delivery man accepts delivery orders within the TLET Nursery Plant Shopping System. Starting from the Delivery Man home screen, when the delivery man clicks on the "Pending Orders" section, the system will fetch a list of pending delivery orders that are assigned by the administrator. If no orders are assigned, an error message is displayed to inform the delivery man that there are no deliveries to process. If the pending orders exist, the system displays a list including details such as delivery order ID, customer name, customer ID, phone number, and delivery address. The delivery man then has the option to select and accept a specific order. Upon clicking the "Accept" button, the system validates this action, changes the order status to "Out For Delivery" and updates the database accordingly. Conversely, if the delivery man chooses to reject an order by clicking the "Reject" button, the system removes the order from the pending order list.

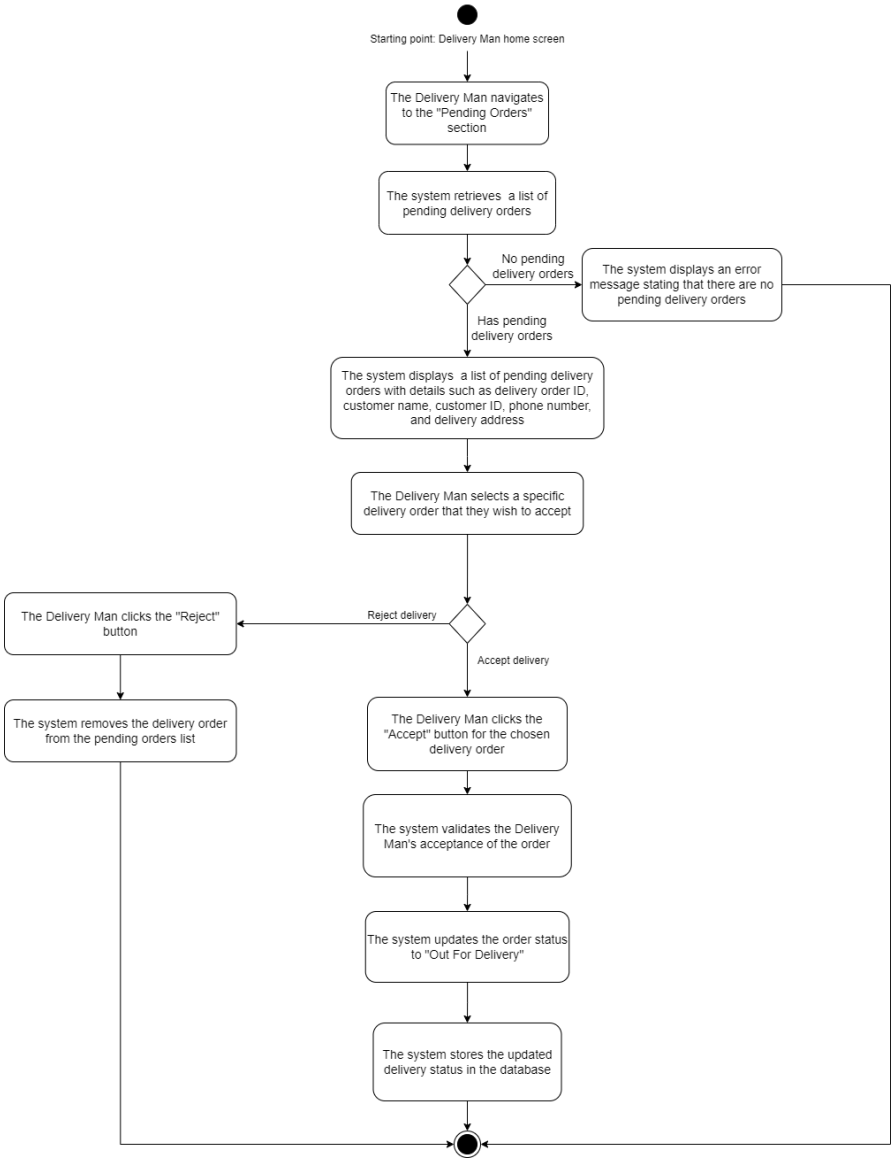


Figure 3.19: Delivery Order Acceptance Activity Diagram

3.6.5 A delivery man views accepted deliveries and confirms delivery.

Figure 3.20 illustrates a delivery man managing the accepted plant delivery orders within the TLET Nursery Plant Shopping System. Beginning at the Delivery Man home screen, the delivery man navigates to the "Accepted Deliveries" section where the system will display a list of currently accepted plant deliveries. If there are no accepted deliveries, the system will display an error message stating there are no accepted plant deliveries. If there are deliveries accepted, the delivery man will be displayed with details such as the delivery order ID, customer name, phone number, and delivery address by the system. After successfully delivering an order, the delivery man selects a delivery order to confirm its completion by clicking the "Confirm Delivery" button. The system then validates the confirmation, updates the status of the delivery order to "Completed" and stores this updated status in the database.

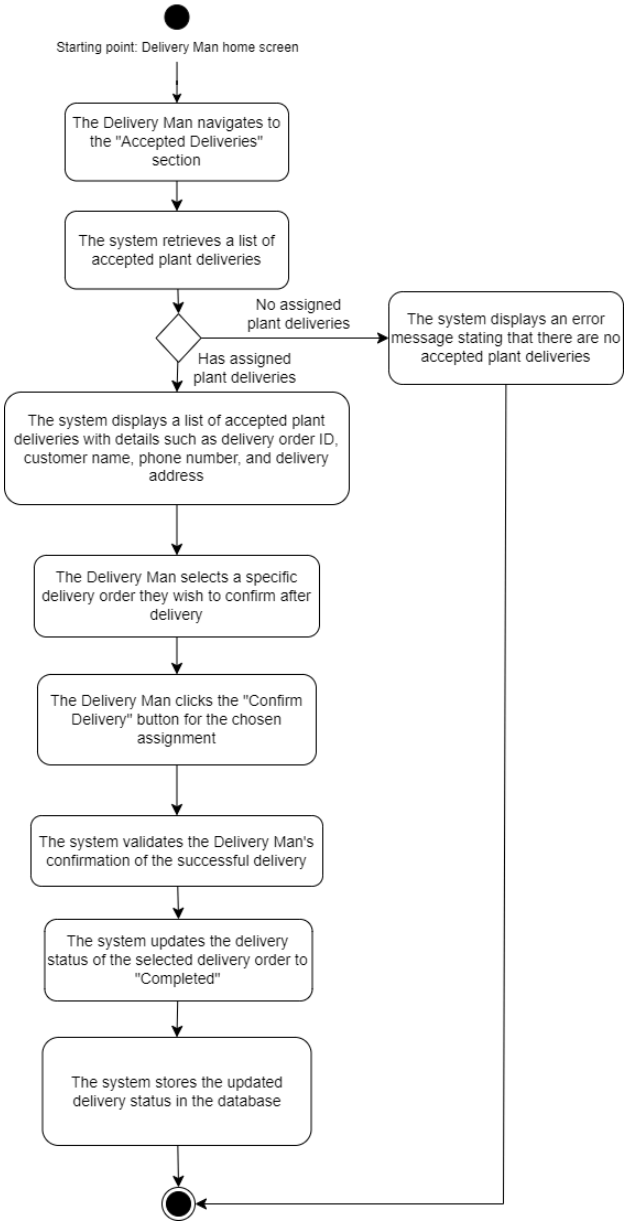


Figure 3.20: Accepted Delivery Viewing and Confirmation Activity Diagram

4 Implementation

4.1 Development Environment

Figure 4.1 showcases the structure of a Django project designed for a delivery man in the Nursery Plant Shopping System. The project is organized into a typical Django application structure, including the main project folder TruePltSys, the Django application app, and other subdirectories such as migrations for database schema changes, static for CSS or JS, and templates for HTML files. Specific HTML templates such as acceptedDelivery.html, accountSetting.html, changePassword.html, deliveryDashboard.html, editProfile.html, and pendingOrder.html correspond to the various use cases for a delivery personnel. The models.py contains the data structures for delivery man's user accounts and orders, while views.py would contain the logic to handle requests and responses. The urls.py file would define the URL patterns associated with these views.

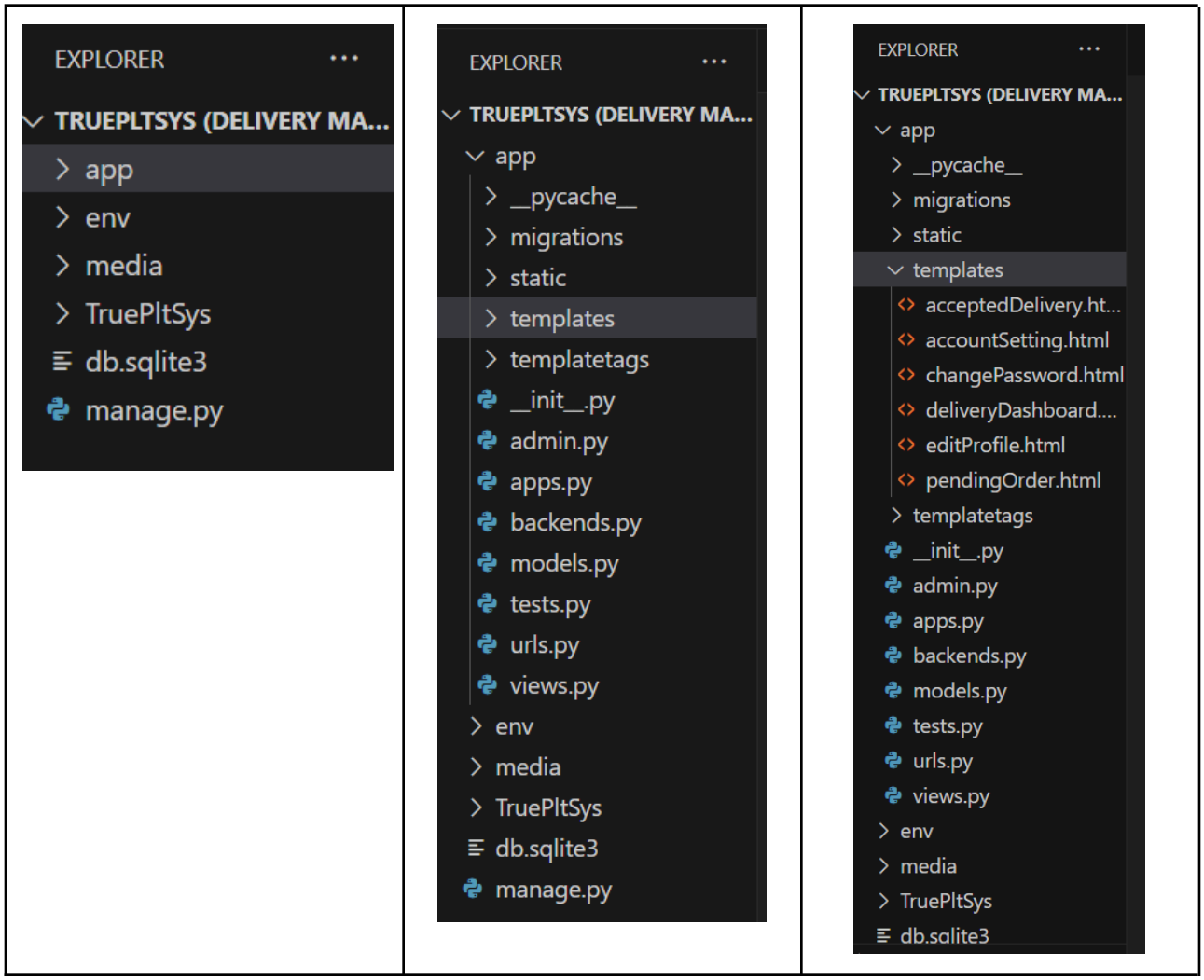


Figure 4.1: Nursery Plant Shopping System (Delivery Man's) Project Explorer

4.2 Main Program Codes

For the delivery man's interaction with the TLET Nursery Plant Shopping System, the main file handling the logic is views.py, which resides within a single Django application. This file hosts a suite of functions that facilitate the delivery workflow, such as displaying the dashboard for the delivery man (deliveryDashboard.html), handling delivery acceptance (pendingOrder.html) and (acceptedDelivery.html), and managing personal account settings (accountSetting.html). These are the main source code that are involved in the delivery man's function.

4.2.1 Delivery man dashboard.

Figure 4.2 illustrates the deliveryDashboard function within the views.py module, which is the linchpin for the Delivery Man Dashboard in the Nursery Plant Shopping System. This function begins by retrieving the delivery person's email from the session data and checks for any alert messages to be displayed. If the email is registered, it fetches the delivery man's details from the database using the DeliveryMan model and prepares these details to be displayed on the dashboard. The corresponding HTML template which is shown in Figure 4.3, is then rendered with this information.

```python
# Delivery man
def deliveryDashboard(request):
    user_email = request.session.get('user_email')
    show_alert = request.session.get('show_alert')
    request.session['show_alert'] = False
    if user_email:
        try:
            user = DeliveryMan.objects.get(user = User.objects.get(email = user_email))
            context = {
                'full_name': user.deliveryman_name,
                'email': user_email,
                'address': user.deliveryman_address,
                'state': user.deliveryman_state,
                'ic': user.deliveryman_ic,
                'phone': user.deliveryman_phone_number,
                'show_alert': show_alert
            }
            return render(request, 'deliveryDashboard.html', context)

        except DeliveryMan.DoesNotExist: #If user not found
            messages.error(request, 'User does not exist')
            return redirect('loginPage')
    else:
        return redirect('loginPage') # Back to login page
```

Figure 4.2: deliveryDashboard/view.py/app

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Delivery Man Dashboard</title>
  <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
  <style>
    .vertical-center {
      min-height: 100vh;
      min-height: calc(100vh - 56px);
      display: flex;
      align-items: center;
      justify-content: center;
    }
  </style>
</head>
```

```
<body>
  <nav class="navbar navbar-expand-md navbar-dark" style="background-color: #333333;">
    <div class="container-fluid">
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav"
      aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
          <li class="nav-item deliveryman-section">
            <a class="nav-link active" href="{% url 'deliveryDashboard' %}">Delivery Man</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'pendingOrder' %}">Pending Order</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'acceptedDelivery' %}">Accepted Deliveries</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'accountSetting' %}">Account Settings</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'logout' %}">Logout</a>
          </li>
        </ul>
        <div class="ml-auto navbar-text">
          Welcome, {{ full_name }}
        </div>
```

```
        </div>
      </div>
    </nav>
    <div id="successMessage" class="alert alert-success" style="display: none;">
      Your account was successfully created.
    </div>
    <div class="vertical-center">
      <h1 class="display-4">Welcome to Delivery Man Dashboard</h1>
    </div>
    {% if show_alert %}
    <script>
      window.onload = function() {
          var successMessageDiv = document.getElementById('successMessage');
          successMessageDiv.style.display = 'block';

          setTimeout(function() {
              successMessageDiv.style.display = 'none';
          }, 5000);
      };
    </script>
    {% endif %}
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
</body>

</html>
```

Figure 4.3: deliveryDashboard.html/templates/app

4.2.2 A delivery man changes their password.

Figure 4.4 and Figure 4.5 showcase the process for a delivery man to change their password within the Nursery Plant Shopping System. The changePassword function in Figure 4.4 starts by checking if the delivery man is logged in via their email. If they are, and they submit a 'POST' request, the function checks if the new password is different from the current one. If it passes this check, and the user is authenticated, the password is updated in the database. If the current password is incorrect or the new password isn't suitable, it shows an error message. The changePassword.html page as shown in Figure 4.5 provides a form where the delivery man can enter their current and new passwords.

```python
def changePassword(request):
    user_email = request.session.get('user_email')
    request.session['show_alert'] = False
    if user_email:

        if request.method == 'POST':
            password = request.POST.get('currentPassword')
            passwordNew = request.POST.get('newPassword')

            if (password == passwordNew):
                messages = 'New password cannot be the same as current password'
                return render(request, 'changePassword.html',{'messages': messages})

            user = authenticate(request, email = user_email, password = password)
            if user is not None:
                user.set_password(passwordNew)
                user.save()
                request.session['show_alert'] = True
                return redirect('accountSetting')
            else:
                invalid_password_messages = 'Invalid current password'
                return render(request, 'changePassword.html',{'invalid_password_messages': invalid_password_messages})

        return render(request, 'changePassword.html')
    else:
        return redirect('loginPage') # Back to login page
```

Figure 4.4: changePassword/view.py/app

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Change Password</title>
  <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
  <style>
    .vertical-center {
      min-height: 100vh;
      min-height: calc(90vh - 120px);
      display: flex;
      align-items: center;
      justify-content: center;
    }
    .change-password-container {
      width: 100%;
      max-width: 800px;
    }
  </style>
</head>
```

```
<body>
  <nav class="navbar navbar-expand-md navbar-dark" style="background-color: #333333;">
    <div class="container-fluid">
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav"
      aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
          <li class="nav-item deliveryman-section">
            <a class="nav-link active" href="{% url 'deliveryDashboard' %}">Delivery Man</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'pendingOrder' %}">Pending Order</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'acceptedDelivery' %}">Accepted Deliveries</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'accountSetting' %}">Account Settings</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'logout' %}">Logout</a>
          </li>
        </ul>
      </div>
    </div>
```

```
<div class="container mt-4">
  <h2 class="title">Change Password</h2>
  <div class="vertical-center">
    <div class="change-password-container">
      <form id="changePasswordForm" method="post" action="{% url 'changePassword' %}" onsubmit="return validateForm()">
        {% csrf_token %}
        <div class="form-group">
          <label for="currentPassword">Current Password:</label>
          <input type="password" class="form-control" id="currentPassword" name="currentPassword" required>
        </div>

        <div class="form-group">
          <label for="newPassword">New Password:</label>
          <input type="password" class="form-control" id="newPassword" name="newPassword" required>
        </div>
        <p class="mt-3 text text-danger" id="errorNewPassword"></p>
        {% if messages %}
            <p class="mt-3 text text-danger">{{ messages }}</p>
        {% endif %}
        {% if invalid_password_messages %}
            <p class="mt-3 text text-danger">{{ invalid_password_messages }}</p>
        {% endif %}
        <div class="d-grid gap-2">
          <button type="submit" class="btn btn-primary">Confirm</button>
        </div>
      </form>
    </div>
  </div>
```

```
  <script>
    function validateForm() {
      var password = document.getElementById('newPassword').value;
      var errorNewPasswordElement = document.getElementById('errorNewPassword');

      // Validate password
      var passwordPattern = /^(?=.*[a-zA-Z])(?=.*\d)(?=.*[!@#$%^&*()<>?/\\|}{~:]).{8,}$/;
      if (!passwordPattern.test(password)) {
          if (password.length < 8) {
            errorNewPasswordElement.textContent = 'Password must be at least 8 characters long.';
          }
          if (/^[a-zA-Z]+$/.test(password)) {
            errorNewPasswordElement.textContent += 'Password must contain at least one digit.';
          }
          if (/^\d+$/.test(password)) {
            errorNewPasswordElement.textContent += 'Password must contain at least one alphabet.';
          }
          return false;
      }
      return true;
    }
  </script>

</body>
</html>
```

Figure 4.5: changPassword.html/templates/app

4.2.3 A delivery man can edit profile details.

Figure 4.6 and Figure 4.7 outline the implementation of the profile editing feature for a delivery man in the Nursery Plant Shopping System. The editProfile function within Figure 4.6 orchestrates the retrieval and update of profile information. Initially, it checks if the delivery man is logged in by verifying their email. Once authenticated, it allows for the retrieval of the delivery man's current profile details from the database and displays them on the editProfile.html page. This HTML page includes a form populated with the delivery man's existing information. The delivery man can update fields such as name, phone number, and address.

```
UserModel = get_user_model()
def editProfile(request):
    user_email = request.session.get('user_email')
    alert = request.session.get('alert')
    request.session['alert'] = False
    if user_email:
        try:
            if User.objects.get(email=user_email).is_deliveryman: # Delivery man's edit profile
                deliveryman = DeliveryMan.objects.get(user = User.objects.get(email=user_email))
                context = {
                    'full_name': deliveryman.deliveryman_name,
                    'email': user_email,
                    'address': deliveryman.deliveryman_address,
                    'state': deliveryman.deliveryman_state,
                    'ic': deliveryman.deliveryman_ic,
                    'phone': deliveryman.deliveryman_phone_number,
                    'alert': alert,
                }
```

```
            if request.method == 'POST':
                name = request.POST.get('name')
                email = request.POST.get('email')
                phone = request.POST.get('phone')
                address = request.POST.get('address')
                state = request.POST.get('state')
                ic = request.POST.get('ic')
                with transaction.atomic():
                    # Update DeliveryMan details
                    deliveryman.deliveryman_name = name
                    deliveryman.deliveryman_address = address
                    deliveryman.deliveryman_state = state
                    deliveryman.deliveryman_ic = ic
                    deliveryman.deliveryman_phone_number = phone
                    deliveryman.save()
                    context['update_success'] = True

                    # Update User email
                    user = UserModel.objects.get(email=user_email)
                    user.email = email
                    user.save()
                    request.session['user_email'] = email

                request.session['alert'] = True
                return redirect('editProfile')

            return render(request, 'editProfile.html', context)
```

```
        except DeliveryMan.DoesNotExist:
            messages.error(request, 'User does not exist')
            return redirect('loginPage')

    else:
        return redirect('loginPage')
```

Figure 4.6: editProfile/view.py/app

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Edit Profile</title>
  <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
  <style>
    .vertical-center {
      min-height: 100vh;
      min-height: calc(90vh - 120px);
      display: flex;
      align-items: center;
      justify-content: center;
    }
```

```css
.form-container {
  width: 100%;
  max-width: 960px;
  padding: 2rem;
  background-color: white;
  border: 1px solid #dee2e6;
  border-radius: 0.25rem;
}
.form-container label {
    display: block;
    margin-top: 0.5rem;
}
.form-container input {
    width: 100%;
    max-width: 100%;
}
@media (max-width: 576px) {
    .buttons .form-btn {
        width: 100%;
        margin-bottom: 0.5rem;
    }
    .buttons a {
        padding: 0.375rem 0.75rem;
    }
}
```

```css
.buttons {
  display: flex;
  justify-content: space-between;
  margin-top: 1rem;
}
```

```
</style>
</head>
<body>
<nav class="navbar navbar-expand-md navbar-dark" style="background-color: #333333;">
<div class="container-fluid">
<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
<span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarNav">
<ul class="navbar-nav">
<li class="nav-item deliveryman-section">
<a class="nav-link active" href="{% url 'deliveryDashboard' %}">Delivery Man</a>
</li>
<li class="nav-item">
<a class="nav-link" href="{% url 'pendingOrder' %}">Pending Order</a>
</li>
<li class="nav-item">
<a class="nav-link" href="{% url 'acceptedDelivery' %}">Accepted Deliveries</a>
</li>
<li class="nav-item">
<a class="nav-link" href="{% url 'accountSetting' %}">Account Settings</a>
</li>
<li class="nav-item">
<a class="nav-link" href="{% url 'logout' %}">Logout</a>
</li>
</ul>
</div>
</div>
</nav>
```

```
<div class="container mt-4">
  <h2 class="title">Edit Profile</h2>
  <div id="successMessage" class="alert alert-success" style="display: none;">
    Your profile was successfully updated.
  </div>
    <div class="vertical-center">
      <div class="form-container">
        <form id="editProfileForm" method="POST" action="{% url 'editProfile' %}" onsubmit="return validateForm()">
          {% csrf_token %}
          <label for="id_name">Full Name:</label>
          <input type="text" id="id_name" name="name" value="{{ full_name }}" required>
          <p class="mt-3 text text-danger" id="errorName"></p>

          <label for="id_email">Email:</label>
          <input type="email" id="id_email" name="email" value="{{ email }}" required>
          <p class="mt-3 text text-danger" id="errorEmail"></p>

          <label for="id_phone">Phone Number:</label>
          <input type="text" id="id_phone" name="phone" value="{{ phone }}" required>
          <p class="mt-3 text text-danger" id="errorPhone"></p>

          <label for="id_address">Address:</label>
          <input type="text" id="id_address" name="address" value="{{ address }}" required>

          <label for="id_state">State:</label>
          <input type="text" id="id_state" name="state" value="{{ state }}" required>

          <label for="id_ic">IC Number:</label>
          <input type="text" id="id_ic" name="ic" value="{{ ic }}" required>
          <p class="mt-3 text text-danger" id="errorIC"></p>

          <div class="buttons">
            <button type="button" class="btn btn-primary form-btn" onclick="location.href='{% url 'accountSetting' %}'">Back</button>
            <button type="submit" class="btn btn-primary form-btn confirm">Confirm</button>
          </div>
        </form>
      </div>
    </div>
  </div>
```

```
{% if alert %}
    <script>
      window.onload = function() {
          var successMessageDiv = document.getElementById('successMessage');
          successMessageDiv.style.display = 'block';

          setTimeout(function() {
              successMessageDiv.style.display = 'none';
          }, 5000);
      };
    </script>
{% endif %}
<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
<script>
  function validateForm() {
    var fullName = document.getElementById('id_name').value;
    var icNumber = document.getElementById('id_ic').value;
    var phoneNumber = document.getElementById('id_phone').value;
    var email = document.getElementById('id_email').value;
    var role = "delivery_man";

    var errorNameElement = document.getElementById('errorName');
    var errorICElement = document.getElementById('errorIC');
    var errorPhoneElement = document.getElementById('errorPhone');
    var errorEmailElement = document.getElementById('errorEmail');

    // Reset existing error messages
    errorNameElement.textContent = '';
    errorICElement.textContent = '';
    errorPhoneElement.textContent = '';
    errorEmailElement.textContent = '';

    var msgBool = true;

    // Validate full name
    var namePattern = /^[a-zA-Z\s]*$/;
    if (!namePattern.test(fullName)) {
        errorNameElement.textContent = 'Full name can only contain letters.';
        msgBool = false;
    }
```

```
        // Validate IC number (should contain only digits)
        var icPattern = /^\d+$/;
        if (!icPattern.test(icNumber)) {
            errorICElement.textContent = 'IC number can only contain digits.';
            msgBool = false;
        }

        // Validate phone number (should contain only digits)
        var phonePattern = /^\d+$/;
        if (!phonePattern.test(phoneNumber)) {
            errorPhoneElement.textContent = 'Phone number can only contain digits.';
            msgBool = false;
        }
```

```
        $.ajax({
            type: 'GET',
            url: '{% url 'checkEditProfile' %}',
            data: {
                fullName: fullName,
                icNumber: icNumber,
                email: email,
                role: role
            },
            success: function (response) {
                if (response.exists) {
                    // Display error messages based on which data already exists
                    if (response.exists_name) {
                        errorNameElement.textContent = 'This name already exists. Please choose a different name.';
                        msgBool = false;
                    }

                    if (response.exists_ic) {
                        errorICElement.textContent = 'This IC number already exists. Please choose a different IC number.';
                        msgBool = false;
                    }

                    if (response.exists_email) {
                        errorEmailElement.textContent = 'This email already exists. Please choose a different email.';
                        msgBool = false;
                    }
                }
            }
        });

        if (msgBool == false) {
            return false;
        }

        // If all validations pass, the form will be submitted
        return true;
    }
    </script>

</body>
</html>
```

Figure 4.7: editProfile.html/templates/app

4.2.4 A delivery man accepts a delivery order.

Figure 4.8 and 4.9 details the software component interactions within the Nursery Plant Shopping System when a delivery man accepts a delivery order. The pendingOrder function in Figure 4.8 initiates by checking the logged-in status through the delivery man's email. Once verified, it fetches

all 'Ready' orders assigned to the delivery man. The HTML page in Figure 4.9 displays these orders in a structured list, with each order offering 'Accept' or 'Reject' options. If the delivery man chooses to accept an order, the corresponding 'Accept' button triggers an AJAX POST request, which updates the order status to "Out Of Delivery" in the database.

```python
def pendingOrder(request):
    user_email = request.session.get('user_email')
    if user_email:
        try:
            user = DeliveryMan.objects.get(user=User.objects.get(email=user_email))
            delivery_orders = Order.objects.filter(delman=user, order_status='Ready')

            no_orders = delivery_orders.count() == 0

            if request.method == 'POST':
                action = request.POST.get('action')
                order_id = request.POST.get('orderId')

                if action == 'accept':
                    orders = Order.objects.filter(delman=user, id=order_id)
                    orders.update(order_status="Out Of Delivery")
                    return redirect('pendingOrder')

                elif action == 'reject':
                    orders = Order.objects.filter(delman=user, id=order_id)
                    orders.update(delman=None)
                    return redirect('pendingOrder')

            return render(request, "pendingOrder.html", {"delivery_orders": delivery_orders, "no_orders": no_orders})

        except DeliveryMan.DoesNotExist:
            return redirect('loginPage')
    else:
        return redirect('loginPage')
```

Figure 4.8: pendingOrder/view.py/app

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Pending Orders</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <style>
        .button-container {
            display: flex;
            justify-content: space-between;
            margin-top: 1rem;
        }
        .vertical-center {
            min-height: 100vh;
            min-height: calc(90vh - 120px);
            display: flex;
            align-items: center;
            justify-content: center;
        }
    </style>
    </head>
<body>
```

```
<nav class="navbar navbar-expand-md navbar-dark" style="background-color: #333333;">
    <div class="container-fluid">
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false"
      aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
          <li class="nav-item deliveryman-section">
            <a class="nav-link active" href="{% url 'deliveryDashboard' %}">Delivery Man</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'pendingOrder' %}">Pending Order</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'acceptedDelivery' %}">Accepted Deliveries</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'accountSetting' %}">Account Settings</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% url 'logout' %}">Logout</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>
```

```
<div class="container mt-4">
    <h2>Pending Orders</h2>
    {% if no_orders %}
    <div class="text-center mb-4 alert alert-light">
        You have no pending orders at the moment.
    </div>
    {% else %}
    {% for order in delivery_orders %}
    <div class="card mb-3">
        <div class="card-body">
            <p class="card-text"><strong>Delivery Order ID:</strong> {{ order.id }}</p>
            <p class="card-text"><strong>Customer Name:</strong> {{ order.customer.customer_name }}</p>
            <p class="card-text"><strong>Customer ID:</strong> {{ order.customer.user.id }}</p>
            <p class="card-text"><strong>Phone Number:</strong> {{ order.customer.customer_phone_number }}</p>
            <p class="card-text"><strong>Delivery address:</strong> {{ order.customer.customer_address }}</p>
            <div class="button-container">
                <button class="btn btn-primary" onclick="handleOrder('reject', {{ order.id }})">Reject</button>
                <button class="btn btn-primary" onclick="handleOrder('accept', {{ order.id }})">Accept</button>
            </div>
        </div>
    </div>
    {% endfor %}
    {% endif %}
</div>
```

```
    <script src="https://code.jquery.com/jquery-3.6.4.min.js"></script>
    <script>
        // When click the button js will link to python to do the action
        function handleOrder(action, orderId) {
            $.ajax({
                type: 'POST',
                url: '{% url 'pendingOrder' %}',
                data: {
                    action: action,
                    orderId: orderId,
                    csrfmiddlewaretoken: '{{ csrf_token }}',
                },
                success: function(data) {
                    location.reload();
                },
                error: function(error) {
                    console.log(error);
                }
            });
        }
    </script>

</body>
</html>
```

Figure 4.9: pendingOrder.html/templates/app

4.2.5 A delivery man views accepted deliveries and confirms delivery.

Figure 4.10 demonstrates the software design for a delivery man when viewing and confirming accepted deliveries. The acceptedDelivery function in Figure 4.10 first confirms the delivery man's login status through their email. If logged in, it fetches all orders with the status "Out Of Delivery" associated with that delivery man. The HTML page, acceptedDelivery.html, displays these orders, and for each one, provides a 'Confirm Delivery' button. When this button is clicked, the confirmDelivery JavaScript function sends an AJAX POST request back to the server to update the order status to "Completed".

```python
def acceptedDelivery(request):
    user_email = request.session.get('user_email')
    if user_email:
        try:
            user = DeliveryMan.objects.get(user=User.objects.get(email=user_email))
            delivery_orders = Order.objects.filter(delman=user, order_status='Out Of Delivery')

            no_orders = delivery_orders.count() == 0

            if request.method == 'POST':
                action = request.POST.get('action')
                order_id = request.POST.get('orderId')

                if action == 'confirm':
                    orders = Order.objects.filter(delman=user, id=order_id)
                    orders.update(order_status="Completed")
                    return redirect('acceptedDelivery')

            return render(request, "acceptedDelivery.html", {"delivery_orders": delivery_orders, "no_orders": no_orders})

        except DeliveryMan.DoesNotExist:
            return redirect('loginPage')
    else:
        return redirect('loginPage')
```

Figure 4.10: acceptedDelivery/view.py/app

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Accepted Deliveries</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <style>
        .vertical-center {
          min-height: 100vh;
          min-height: calc(90vh - 120px);
          display: flex;
          align-items: center;
          justify-content: center;
        }
    </style>
</head>
<body>
    <nav class="navbar navbar-expand-md navbar-dark" style="background-color: #333333;">
        <div class="container-fluid">
            <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav"
            aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav">
                    <li class="nav-item deliveryman-section">
                        <a class="nav-link active" href="{% url 'deliveryDashboard' %}">Delivery Man</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'pendingOrder' %}">Pending Order</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'acceptedDelivery' %}">Accepted Deliveries</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'accountSetting' %}">Account Settings</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'logout' %}">Logout</a>
                    </li>
                </ul>
            </div>
        </div>
```

```
<div class="container mt-4">
    <h2>Accepted Deliveries</h2>
    {% if no_orders %}
    <div class="text-center mb-4 alert alert-light">
        You have no accepted deliveries at the moment.
    </div>
    {% else %}
    {% for order in delivery_orders %}
    <div class="card mb-3">
        <div class="card-body">
        <p class="card-text"><strong>Delivery OrderID:</strong> {{ order.id }}</p>
        <p class="card-text"><strong>Customer Name:</strong> {{ order.customer.customer_name }}</p>
        <p class="card-text"><strong>Phone Number:</strong> {{ order.customer.customer_phone_number }}</p>
        <p class="card-text"><strong>Delivery address:</strong> {{ order.customer.customer_address }}</p>
        <button class="btn btn-primary" onclick="confirmDelivery('confirm', {{ order.id }})">Confirm Delivery</button>
        </div>
    </div>
    {% endfor %}
    {% endif %}
</div>
```

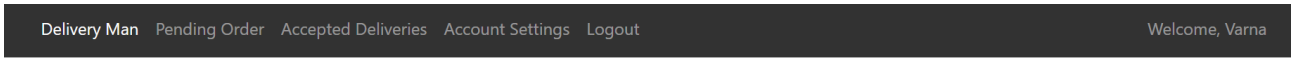
```
<script src="https://code.jquery.com/jquery-3.6.4.min.js"></script>
<script>
    function confirmDelivery(action, orderId) {
        $.ajax({
            type: 'POST',
            url: '{% url 'acceptedDelivery' %}',
            data: {
                action: action,
                orderId: orderId,
                csrfmiddlewaretoken: '{{ csrf_token }}',
            },
            success: function(data) {
                location.reload();
            },
            error: function(error) {
                // Handle the error
                console.log(error);
            }
        });
    }
</script>
</body>
</html>
```

Figure 4.11: acceptedDelivery.html/templates/app

4.3 Sample Screens

4.3.1 Delivery Man Home Screen

Figure 4.12 displays the "Delivery Man Home Screen" of the TLET Nursery Plant Shopping System. Upon login, the delivery man will be presented with a home screen shown in Figure 4.12 featuring a navigation bar with options for 'Pending Order', 'Accepted Deliveries', 'Account Settings', 'Logout' button and a message displaying “Welcome, username”. Below the navigation, a welcoming message marks the entry to the delivery man dashboard.



Welcome to Delivery Man Dashboard

Figure 4.12: Delivery Man Home Screen

4.3.2 Delivery Man Creates a User Account Screen

Figure 4.14 shows the 'Create User Account Screen', which illustrates the process for delivery men to create a new account on the TLET Nursery Plant Shopping System. Upon accessing the system, the delivery man will first encounter the guest home screen in Figure 4.13, which is also the "TLET Nursery Plant Shopping System Screen" before creating a user account. The delivery man can proceed by clicking on the "Sign Up" button and this action takes the delivery man to the 'Creates User Account Screen' shown in Figure 4.14, where the delivery man fills out the form with personal and contact details. Upon clicking the “Submit” button, the system validates the input and saves the new user details to persistent storage. Successful registration is confirmed with a message. Delivery men who have an account can quickly get started by clicking the "Login" button, which takes them to a Login page in Figure 4.15.

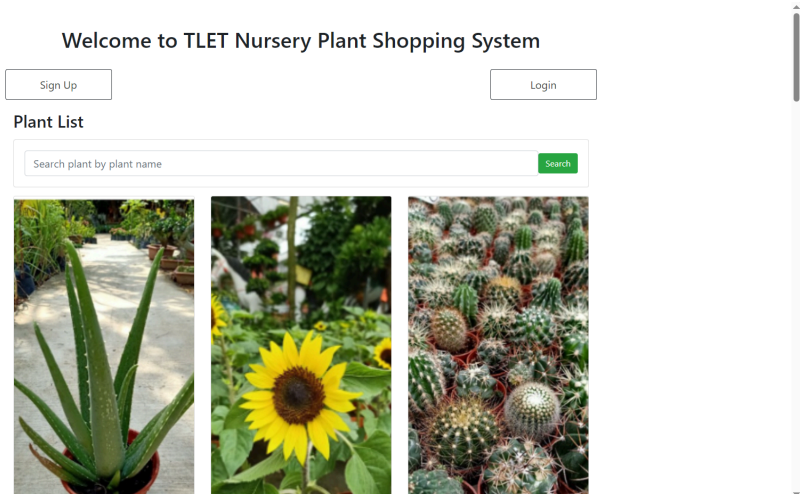


Figure 4.13: Guest Home Screen

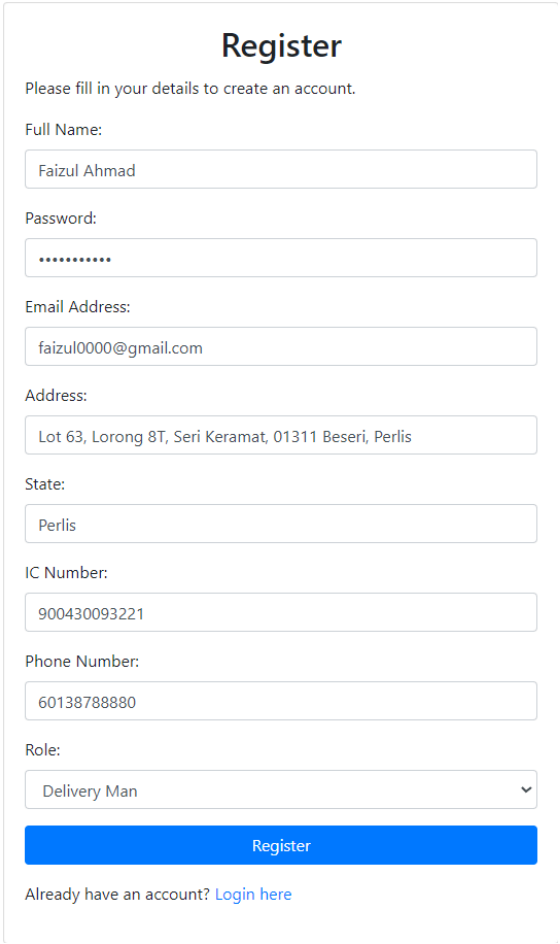


Figure 4.14:Create User Account Screen

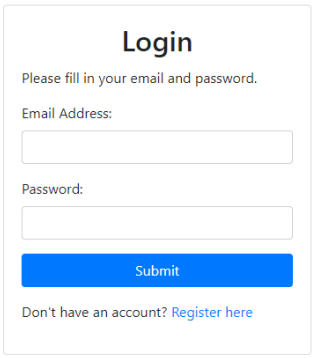


Figure 4.15: TLET Nursery Plant Shopping System Login Screen

4.3.3 Delivery Changes Their Password Screen

Figure 4.17 shows the "Change Password Screen", which presents the interface for delivery man to update their password on the TLET Nursery Plant Shopping System. The process is initiated from the “Delivery Man Home Screen” as shown in Figure 4.12, where the delivery man selects the 'Account Settings' button. This action takes them to the "Account Settings Screen" displayed in Figure 4.16, where the delivery man details are listed. By selecting the 'Change Password' button, the delivery man is directed to the screen shown in Figure 4.17, where they are required to enter their current password followed by the new password they wish to set. After submitting the new password by clicking 'Confirm', the system verifies the new password. If the update is successful, a confirmation message is displayed, and the delivery man is taken back to the "Account Settings Screen" shown in Figure 4.16, completing the password change process.

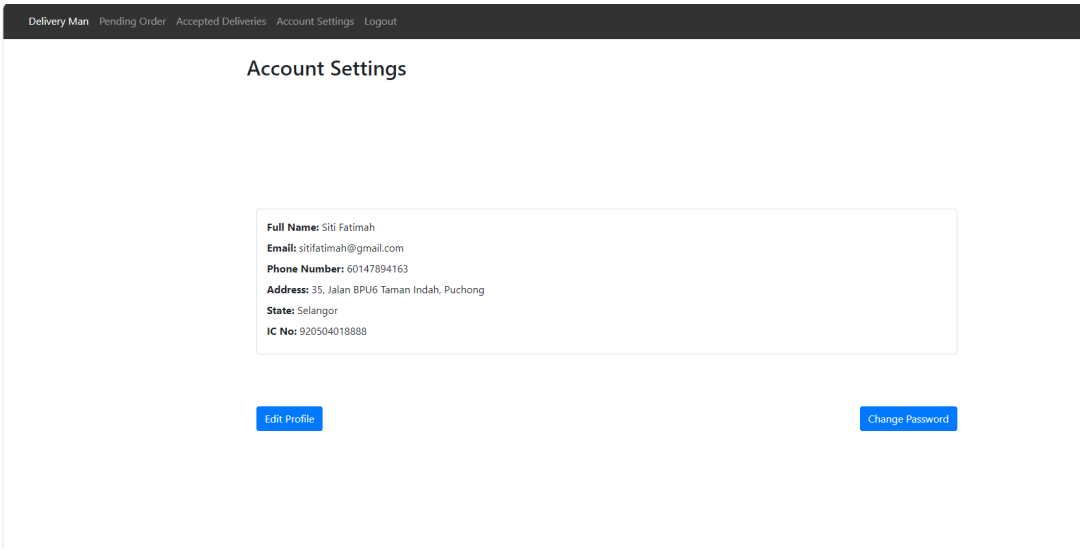


Figure 4.16: Account Settings Screen

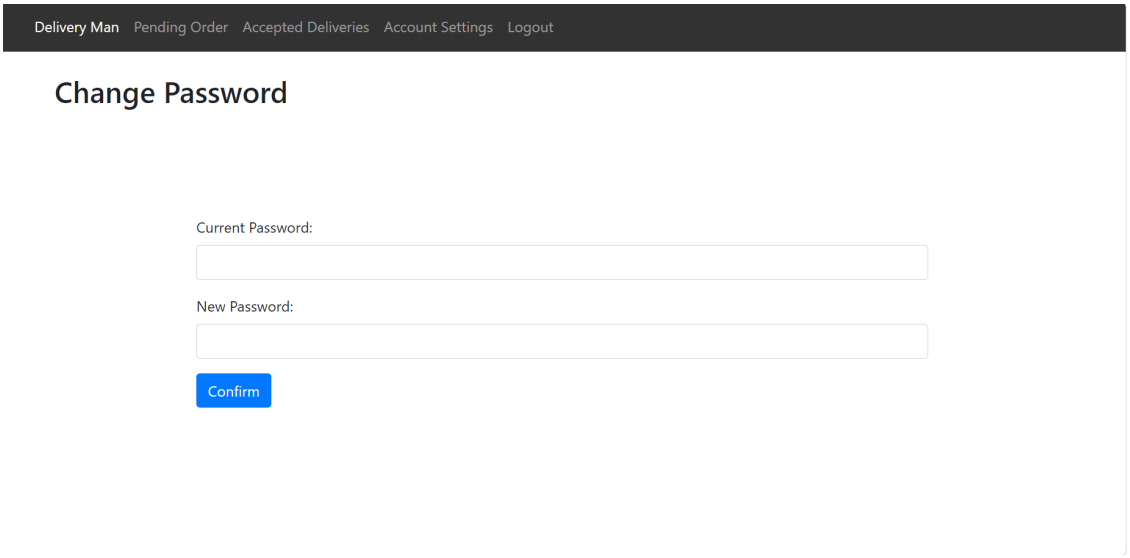


Figure 4.17: Change Password Screen

4.3.4 Delivery Man Can Edit Profile Details Screen

Figure 4.18 shows 'Edit Profile Details Screen', where a delivery man can update their personal information on the TLET Nursery Plant Shopping System. The process is initiated from the "Delivery Man Home Screen" as shown in Figure 4.12, where the delivery man selects the 'Account Settings' button. This action takes them to the "Account Settings Screen" displayed in Figure 4.16, where the delivery man's details are listed. By selecting the 'Edit Profile' button, the delivery man is directed to the screen shown in Figure 4.18, featuring fields to modify the delivery man's personal information. Changes are submitted with the 'Confirm' button, and the system displays a confirmation message. If the delivery man chooses not to update the personal

information, clicking the 'Back' button redirects them to the “Account Settings Screen” in Figure 4.16, without making any updates.

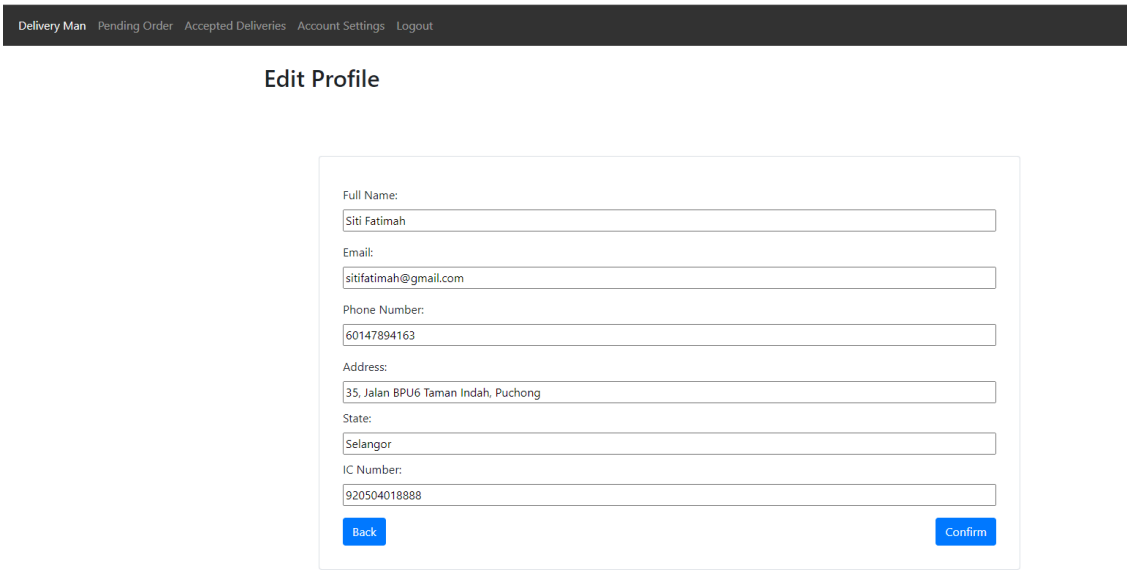


Figure 4.18: Edit Profile Screen

4.3.5 Delivery Man Accepts a Delivery Order Screen

Figure 4.19 shows the "Accepts a Delivery Order Screen" from the TLET Nursery Plant Shopping System, used by the delivery man to manage incoming orders. The process is initiated from the "Delivery Man Home Screen" as shown in Figure 4.12, where the delivery man selects the “Pending Orders” button. This action takes them to the "Accepts a Delivery Order Screen" displayed in Figure 4.19. On this screen, the delivery man can view all pending orders with detailed information including the Delivery Order ID, customer names, customer IDs, contact numbers, and delivery addresses. They have the option to either accept or reject these orders directly from this interface. Once an order is accepted, it is marked as out of delivery. If the Delivery Man clicks the "Reject" button, the system removes the delivery order from the list of pending orders.

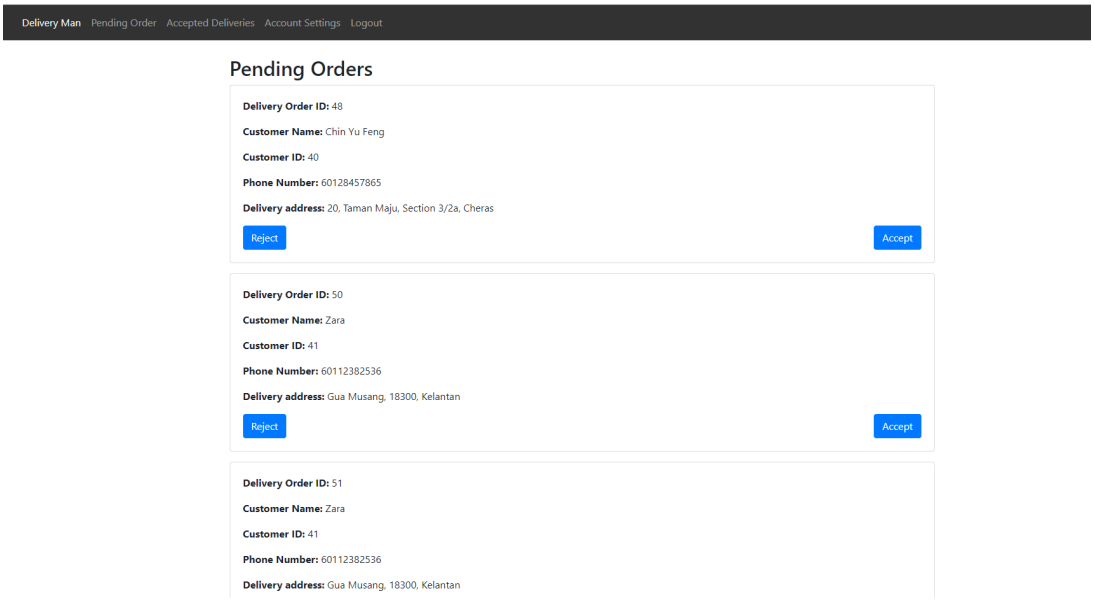


Figure 4.19: Accepts a Delivery Order Screen

4.3.6 Delivery Views Accepted Deliveries and Confirm Delivery Screen

Figure 4.20 presents the "Accepts a Deliveries and Confirm Delivery Screen" of the TLET Nursery Plant Shopping System. The process is initiated from the "Delivery Man Home Screen" as shown in Figure 4.12, where the delivery man selects the 'Accepted Deliveries' button. This action takes them to the "Accepts a Deliveries and Confirm Delivery Screen" displayed in Figure 4.20 which allows delivery men to view a list of all plant deliveries they have accepted, complete with details such as delivery order ID, customer names, phone numbers, and delivery addresses. After the successful delivery of an order, the delivery man can confirm the completion of delivery by clicking the "Confirm Delivery" button provided alongside each order. This confirmation updates the order's status to "Completed" in the system's database, ensuring that the delivery process is accurately tracked.

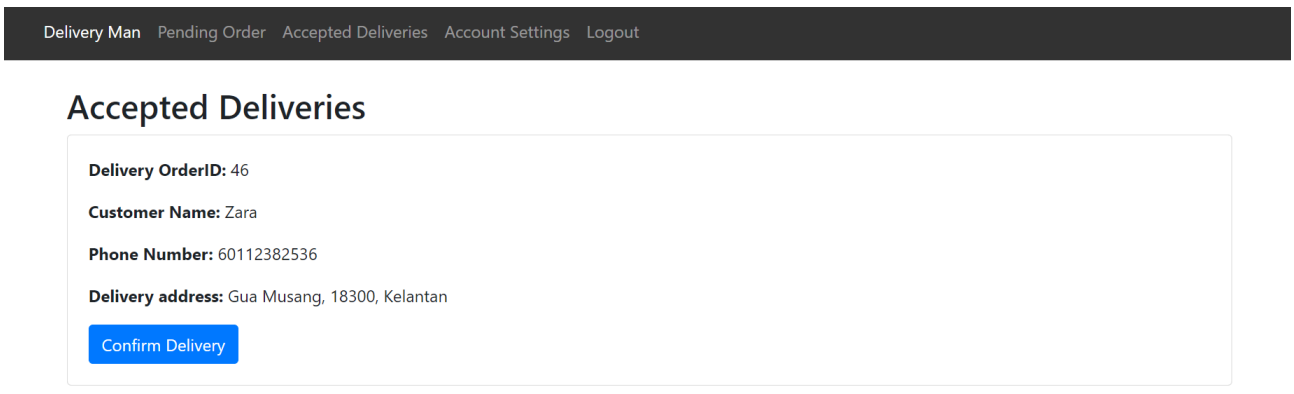


Figure 4.20: Accepts a Deliveries and Confirm Delivery Screen

5 Testing

5.1 Test Data

Table 5.1 shows the test data for creating a delivery man user account. The delivery man is not able to create an account if they have numbers in their name. A delivery man's password must be at least 8 characters long and contain at least one alphabet. They can't create an account if their email address does not have'@'. Lastly, they are not able to create an account if their IC and phone number have characters.

Table 5.1: Create User Account Test Data

Delivery Man Name	**Delivery Man Password**	**Delivery Man Email**	**Delivery Man Address**	**Delivery Man State**	**Delivery Man IC**	**Delivery ManPhone Number**
Faizul Ahmad123	faizul@34**	faizul0000@gmail.com	Lot 63, Lorong 8T, Seri Keramat, 01311 Beseri, Perlis	Perlis	900430093221	60138788880
Faizul Ahmad	faizul@34**	faizul0000gmail.com	Lot 63, Lorong 8T, Seri Keramat, 01311 Beseri, Perlis	Perlis	900430093221	60138788880
Faizul Ahmad	faizul@34**	faizul0000@gmail.com	Lot 63, Lorong 8T, Seri Keramat, 01311 Beseri, Perlis	Perlis	900430093221my	60138788880
Faizul Ahmad	faizul@34**	faizul0000@gmail.com	Lot 63, Lorong 8T, Seri Keramat, 01311 Beseri, Perlis	Perlis	900430093221	60138788880abc

Faizul Ahmad	123	faizul0000@gmail.com	Lot 63, Lorong 8T, Seri Keramat, 01311 Beseri, Perlis	Perlis	900430093221	60138788880
Faizul Ahmad	faizul@34**	faizul0000@gmail.com	Lot 63, Lorong 8T, Seri Keramat, 01311 Beseri, Perlis	Perlis	900430093221	60138788880

Table 5.2 shows the test data for changing a delivery man's password. The delivery man's new password must be at least 8 characters long and contain at least one alphabet. A delivery man's current password must be correct in order to change to the new password. A delivery man's new password cannot be the same as the current password.

Table 5.2: Change User Password Test Data

Delivery Man Name	**Delivery Man Email**	**Delivery Man Current Password**	**Delivery Man New Password**
Faizul Ahmad	faizul0000@gmail.com	faizul@34**	1234
Faizul Ahmad	faizul0000@gmail.com	faizul@34	faizul@34**12
Faizul Ahmad	faizul0000@gmail.com	faizul@34**	faizul@34**
Faizul Ahmad	faizul0000@gmail.com	faizul@34**	faizul@34**12

Table 5.3 shows the test data for editing a delivery man's user profile. The delivery man is not able to edit their profile if they have numbers in their name. They can't edit their profile if their email address does not have'@'. Lastly, they are not able to edit their profile if their IC and phone number have characters.

Table 5.3: Edit User Profile Test Data

Delivery Man Name	**Delivery Man Email**	**Delivery ManPhone Number**	**Delivery Man Address**	**Delivery Man State**	**Delivery Man IC**
Faizul Ahmad345	faizul0000@gmail.com	60138788880	Lot 63, Lorong 8T, Seri Keramat, 01311 Beseri, Perlis	Perlis	900430093221
Faizul Ahmad	faizul0000gmail.com	60138788880	Lot 63, Lorong 8T, Seri Keramat, 01311 Beseri, Perlis	Perlis	900430093221
Faizul Ahmad	faizul0000@gmail.com	60138788880abc	Lot 63, Lorong 8T, Seri Keramat, 01311 Beseri, Perlis	Perlis	900430093221
Faizul Ahmad	faizul0000@gmail.com	60138788880	Lot 63, Lorong 8T, Seri Keramat, 01311 Beseri, Perlis	Perlis	900430093221my
Faizul Mirul	faizul123@gmail.com	60138788000	14, Jalan 3F, PJU8, 10172 Tanjung Bungah, Penang	Penang	900430093221

Table 5.4 shows the test data for accepting delivery order. If the delivery man is new and has not been assigned any delivery orders, the pending order section will be empty. A delivery man who has pending orders can accept or reject the order. If a delivery man accepts a pending order, the order will move to the accepted deliveries section. If a delivery man rejects a pending order, the order will be returned to the admin and the admin will have to assign the order again.

Table 5.4: Accept Delivery Order Test Data

Delivery Man Name	Delivery Man Email	Delivery Man Password	Has Pending Order	Delivery Order ID	Customer Name	Accept/Reject Order	Order Status
Faizul Mirul	faizul123@gmail.com	faizul@34**12	No	-	-	-	-
Siti Fatimah	sitifatimah@gmail.com	qwerty156	Yes	48	Chin Yu Feng	Accept	Ready
Siti Fatimah	sitifatimah@gmail.com	qwerty156	Yes	50	Zara	Reject	Ready

Table 5.5 shows the test data for viewing accepted deliveries and confirm delivery. If the delivery man is new and has not accepted any delivery orders, the accepted deliveries section will be empty. A delivery man who has accepted deliveries can confirm delivery after they have successfully delivered the order. If a delivery man confirms delivery for an order, the order status in the database will be updated to Completed.

Table 5.5: View Accepted Deliveries and Confirms Delivery Test Data

Delivery Man Name	Delivery Man Email	Delivery Man Password	Has Accepted Deliveries	Delivery Order ID	Customer Name	Order Status
Faizul Mirul	faizul123@gmail.com	faizul@34**12	No	-	-	-
Siti Fatimah	sitifatimah@gmail.com	qwerty156	Yes	48	Chin Yu Feng	Out For Delivery

5.2 Acceptance Test

Table 5.6 shows the delivery man acceptance test with each of its use cases.

Module: Delivery Man
Developer: Emily Phang Ru Ying

Table 5.6: Delivery Man Acceptance Test

Criterial	**Fullfield?**	**Remarks**
A delivery man creates a user account.		
A delivery man changes their password.		
A delivery man can edit profile details.		
A delivery man accepts a delivery order.		
A delivery man views accepted deliveries and confirms delivery.		

Date tested:
% Completed:
Test by:
Verified by:

5.3 Test Results

Create User Account Test Result:

Figure 5.1 shows the test result when a delivery man tries to create an account but has numbers in their name. An error message will pop up stating that the full name can only contain letters.

Delivery Man Name	Delivery Man Password	Delivery Man Email	Delivery Man Address	Delivery Man State	Delivery Man IC	Delivery ManPhone Number
Faizul Ahmad123	faizul@34**	faizul0000@gmail.com	Lot 63, Lorong 8T, Seri Keramat, 01311 Beseri, Perlis	Perlis	900430093221	60138788880

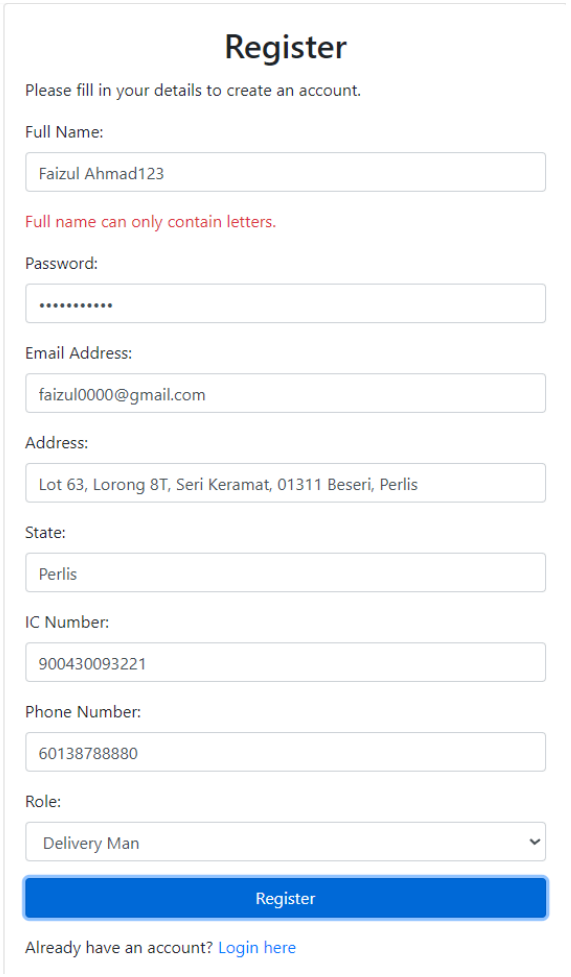


Figure 5.1: Create Delivery Man Account with Numbers in Name Test Result

Figure 5.2 shows the test result when a delivery man tries to create an account without '@' in their email. An error message will pop up asking the delivery man to include '@' in their email.

Delivery Man Name	**Delivery Man Password**	**Delivery Man Email**	**Delivery Man Address**	**Delivery Man State**	**Delivery Man IC**	**Delivery ManPhone Number**
Faizul Ahmad	faizul@34**	faizul0000gmail.com	Lot 63, Lorong 8T, Seri Keramat, 01311 Beseri, Perlis	Perlis	900430093221	60138788880

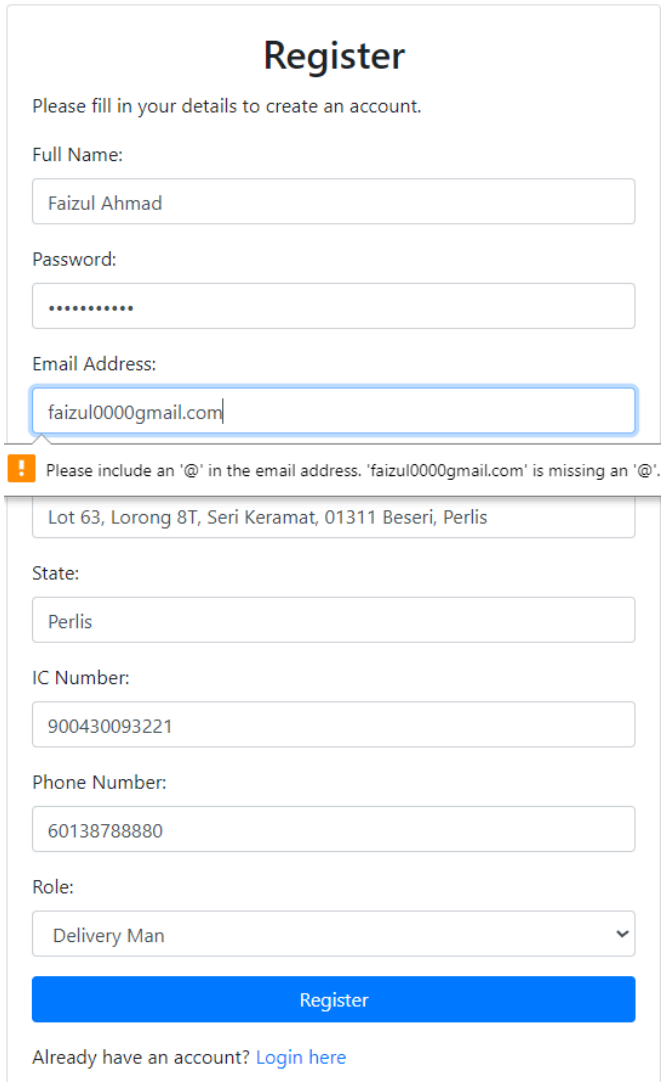


Figure 5.2: Create Delivery Man Account without '@' in Email Test Result

Figure 5.3 shows the test result when a delivery man tries to create an account with characters in their IC. An error message will pop up stating IC number can only contain digits.

Delivery Man Name	**Delivery Man Password**	**Delivery Man Email**	**Delivery Man Address**	**Delivery Man State**	**Delivery Man IC**	**Delivery ManPhone Number**
Faizul Ahmad	faizul@34**	faizul0000@gmail.com	Lot 63, Lorong 8T, Seri Keramat, 01311 Beseri, Perlis	Perlis	900430093221my	60138788880

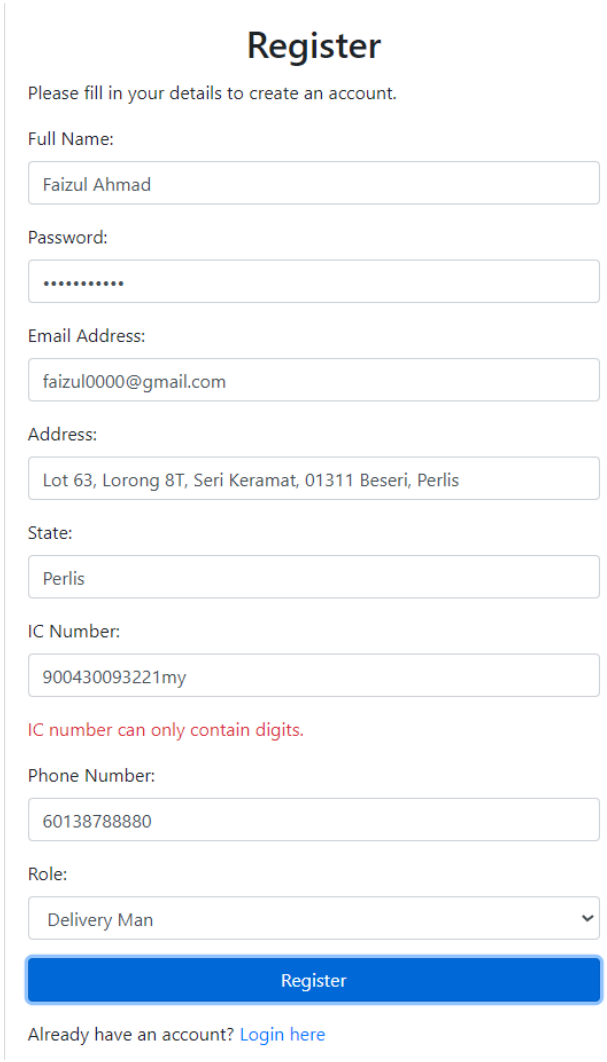


Figure 5.3: Create Delivery Man Account with characters in IC Test Result

Figure 5.4 shows the test result when a delivery man tries to create an account with characters in their Phone Number. An error message will pop up stating Phone Number can only contain digits.

Delivery Man Name	**Delivery Man Password**	**Delivery Man Email**	**Delivery Man Address**	**Delivery Man State**	**Delivery Man IC**	**Delivery ManPhone Number**
Faizul Ahmad	faizul@34**	faizul0000@gmail.com	Lot 63, Lorong 8T, Seri Keramat, 01311 Beseri, Perlis	Perlis	900430093221	60138788880abc

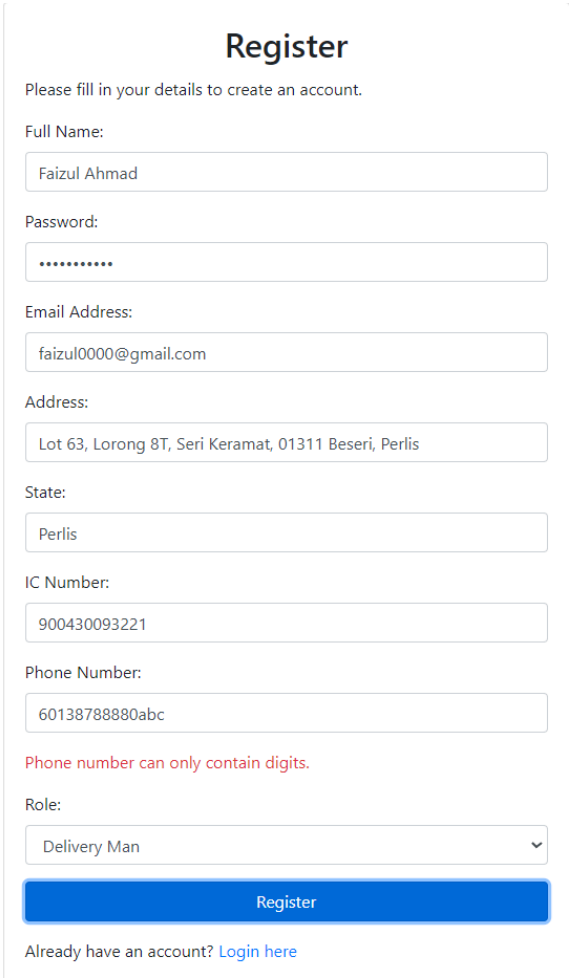


Figure 5.4: Create Delivery Man Account with characters in Phone Number Test Result

Figure 5.5 shows the test result when a delivery man tries to create an account with a short password and without an alphabet. An error message will pop up stating that the password must be at least 8 characters long and contain at least one alphabet.

Delivery Man Name	**Delivery Man Password**	**Delivery Man Email**	**Delivery Man Address**	**Delivery Man State**	**Delivery Man IC**	**Delivery ManPhone Number**
Faizul Ahmad	123	faizul0000@gmail.com	Lot 63, Lorong 8T, Seri Keramat, 01311 Beseri, Perlis	Perlis	900430093221	60138788880

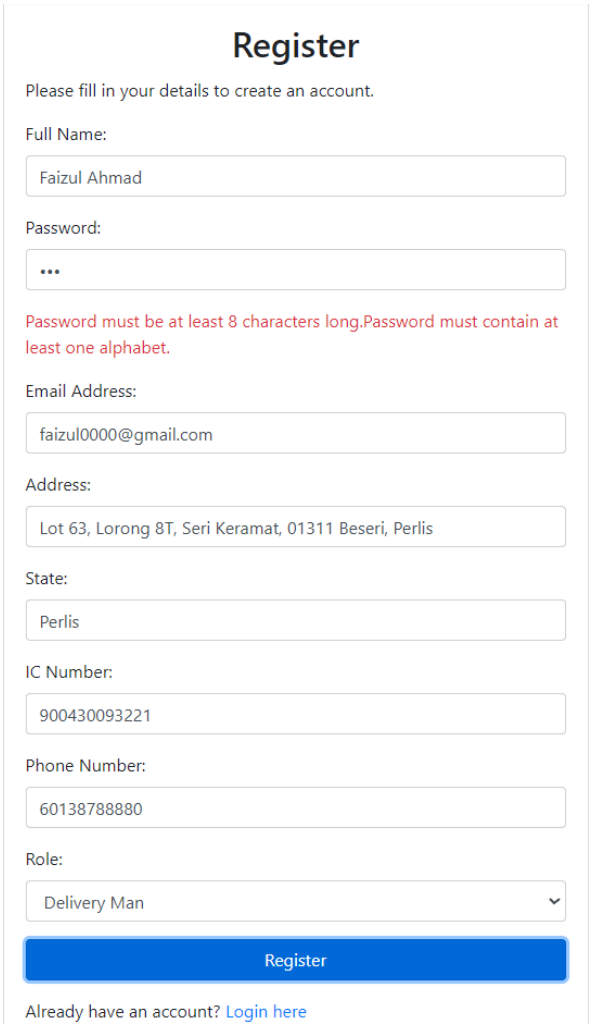


Figure 5.5: Create Delivery Man Account with Short Password and Without Alphabet Test Result

Figure 5.6 shows that the delivery man create an account with correct credentials. Figure 5.7 shows that after successfully creating an account, a message saying your account was successfully created will pop up. Figure 5.8 shows that the Django administration database has the new delivery man stored in it.

Delivery Man Name	**Delivery Man Password**	**Delivery Man Email**	**Delivery Man Address**	**Delivery Man State**	**Delivery Man IC**	**Delivery ManPhone Number**
Faizul Ahmad	faizul@34**	faizul0000@gmail.com	Lot 63, Lorong 8T, Seri Keramat, 01311 Beseri, Perlis	Perlis	900430093221	60138788880

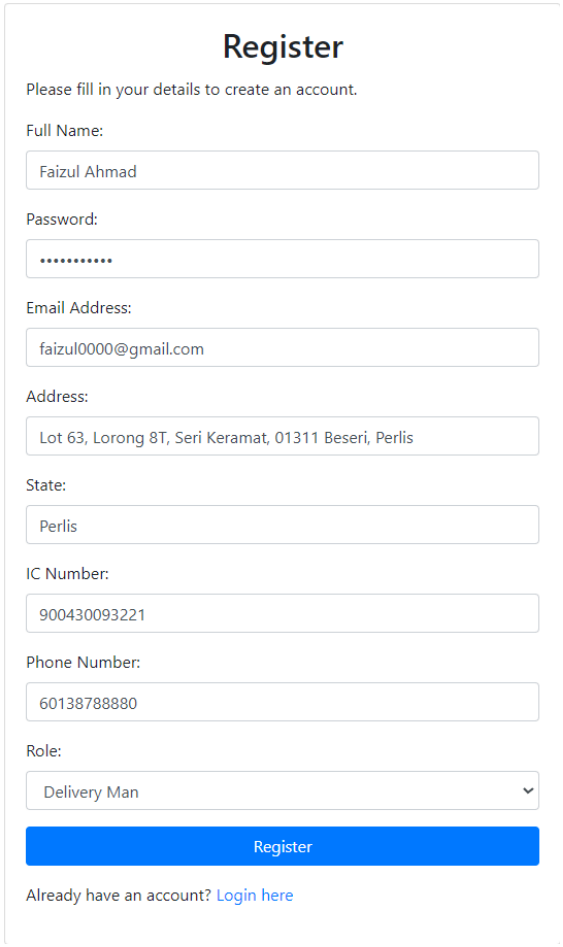
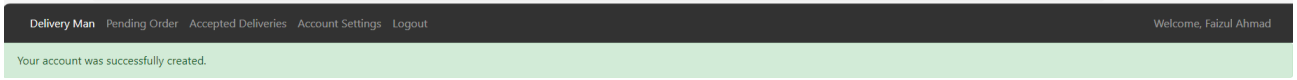


Figure 5.6: Create Delivery Man Account with Correct Credentials



Welcome to Delivery Man Dashboard

Figure 5.7: Delivery Man Dashboard

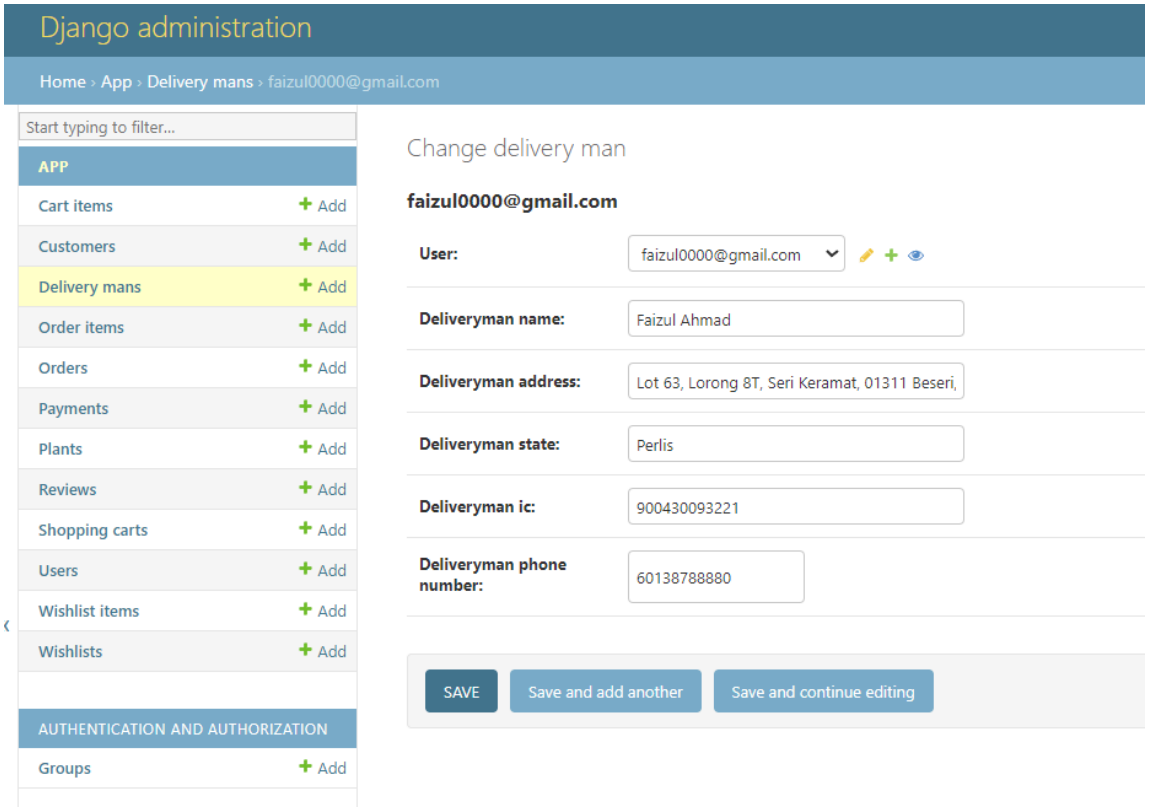


Figure 5.8: Django Administration Database

Change User Password Test Result:

Figure 5.9 shows that the delivery man is trying to change their password with a new short password without alphabets. An error message stating that the password must be at least 8 characters long and must contain at least one alphabet will pop up.

Delivery Man Name	**Delivery Man Email**	**Delivery Man Current Password**	**Delivery Man New Password**
Faizul Ahmad	faizul0000@gmail.com	faizul@34**	1234

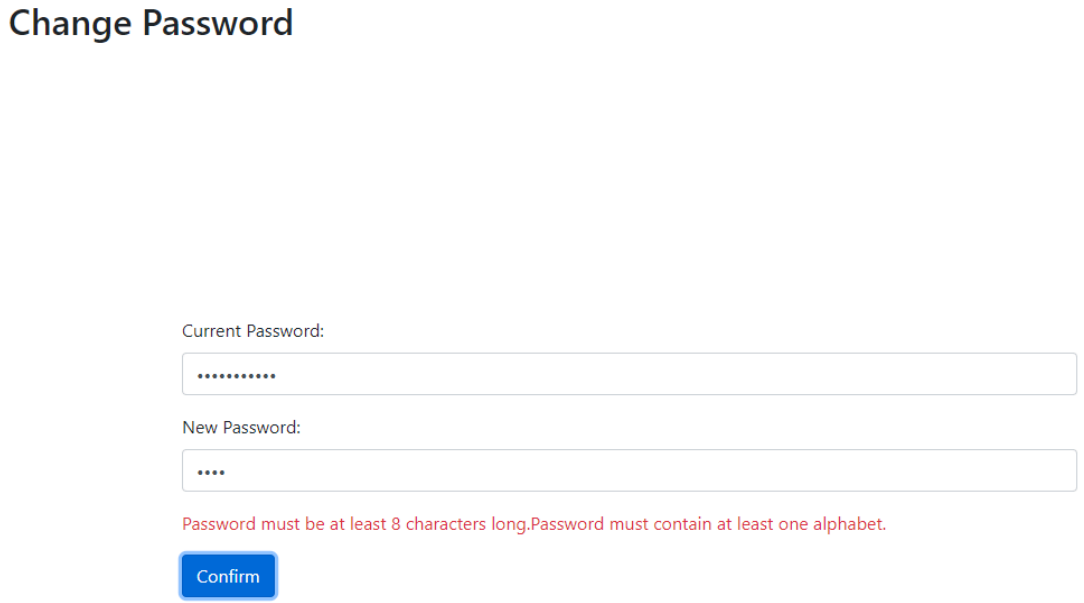


Figure 5.9: Password Too Short Test Result

Figure 5.10 shows that the delivery man is trying to change their password but entered the wrong current password. An error message stating that the current password is invalid will pop up.

Delivery Man Name	Delivery Man Email	Delivery Man Current Password	Delivery Man New Password
Faizul Ahmad	faizul0000@gmail.com	faizul@34	faizul@34**12

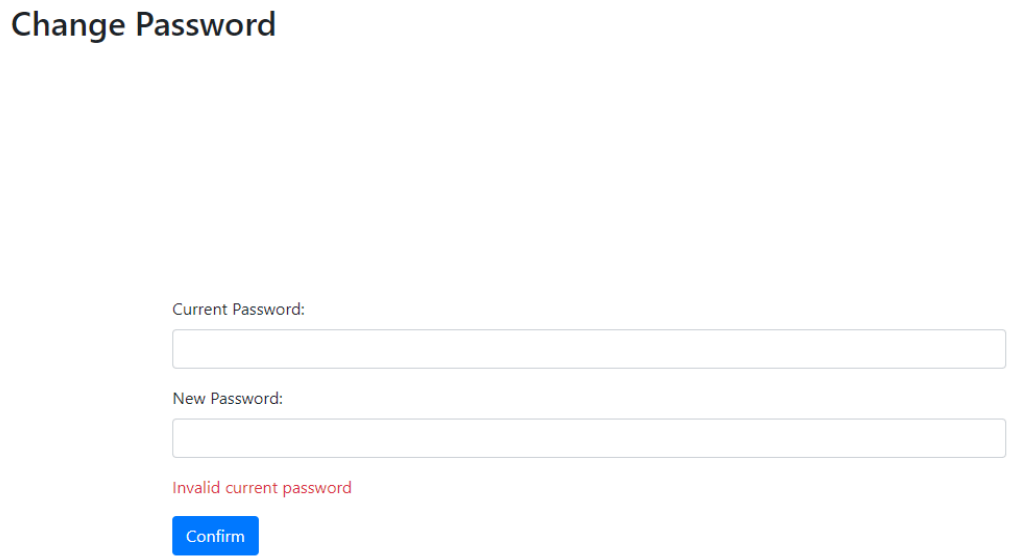


Figure 5.10: Wrong Current Password Test Result

Figure 5.11 shows that the delivery man is trying to change their password but entered the same new password as the current password. An error message stating that the new password cannot be the same as the current password will pop up.

Delivery Man Name	Delivery Man Email	Delivery Man Current Password	Delivery Man New Password
Faizul Ahmad	faizul0000@gmail.com	faizul@34**	faizul@34**

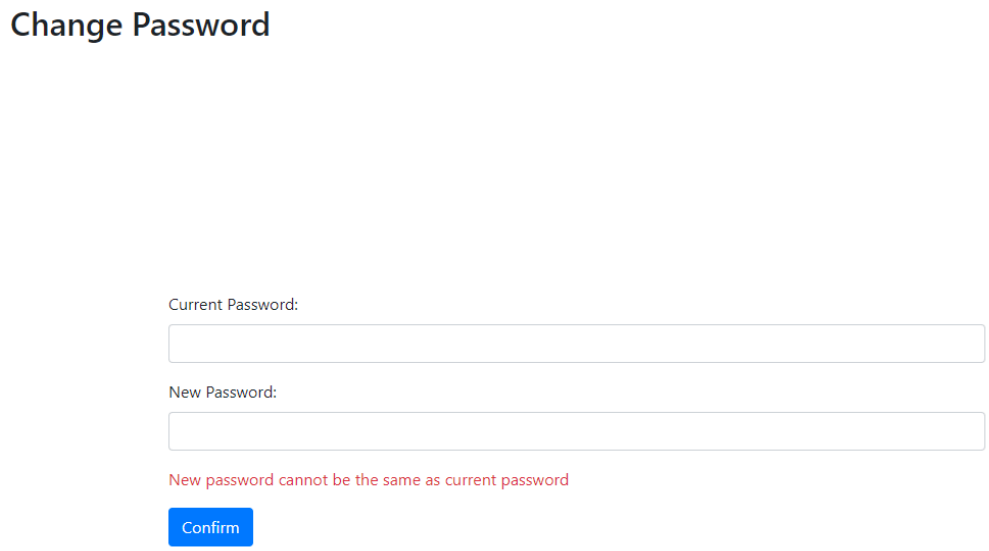


Figure 5.11: Wrong Current Password Test Result

Figure 5.12 shows that the delivery man has successfully changed their password by entering the correct current and new password. A message stating that the password has successfully changed will pop up. Figure 5.13 shows that the django administration's database for delivery man Faizul Ahmad has changed his password in the database.

Delivery Man Name	Delivery Man Email	Delivery Man Current Password	Delivery Man New Password
Faizul Ahmad	faizul0000@gmail.com	faizul@34**	faizul@34**12

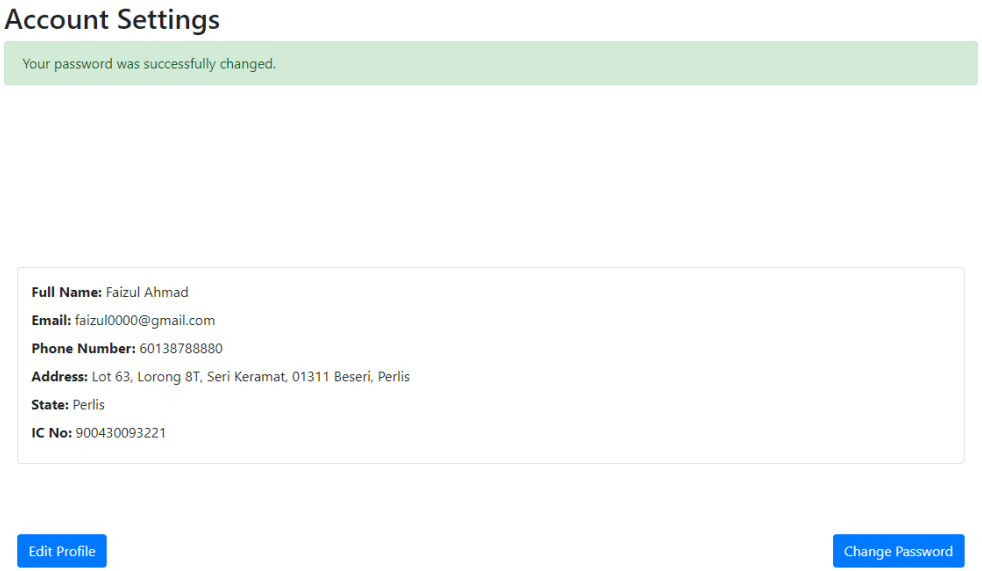


Figure 5.12: Delivery Man Successfully Changed Password

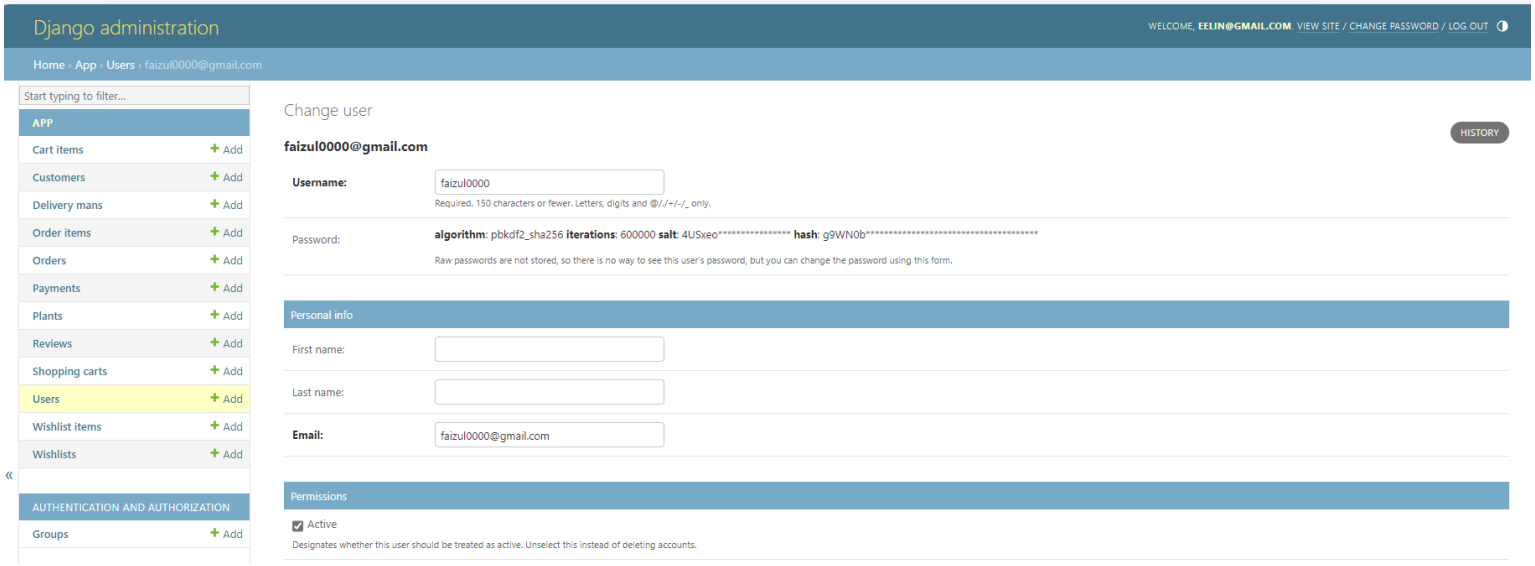


Figure 5.13: Django Administration Delivery Man Password Changed

Edit User Profile Test Result:

Figure 5.14 shows the test result when a delivery man tries to edit his profile but has numbers in his name. An error message will pop up stating that the full name can only contain letters.

Delivery Man Name	Delivery Man Email	Delivery ManPhone Number	Delivery Man Address	Delivery Man State	Delivery Man IC
Faizul Ahmad345	faizul0000@gmail.com	60138788880	Lot 63, Lorong 8T, Seri Keramat, 01311 Beseri, Perlis	Perlis	900430093221

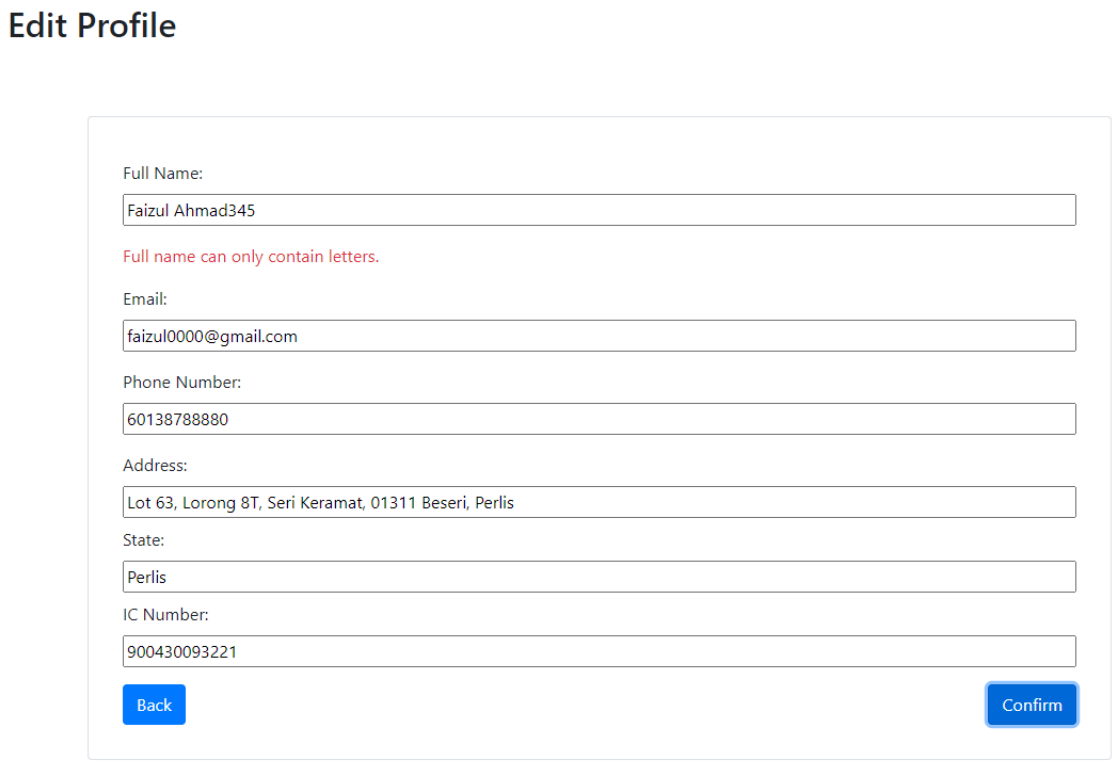


Figure 5.14: Full Name Has Numbers Test Result

Figure 5.15 shows the test result when a delivery man tries to edit his profile without '@' in their email. An error message will pop up asking the delivery man to include '@' in their email.

Delivery Man Name	Delivery Man Email	Delivery ManPhone Number	Delivery Man Address	Delivery Man State	Delivery Man IC
Faizul Ahmad	faizul0000gmail.com	60138788880	Lot 63, Lorong 8T, Seri Keramat, 01311 Beseri, Perlis	Perlis	900430093221

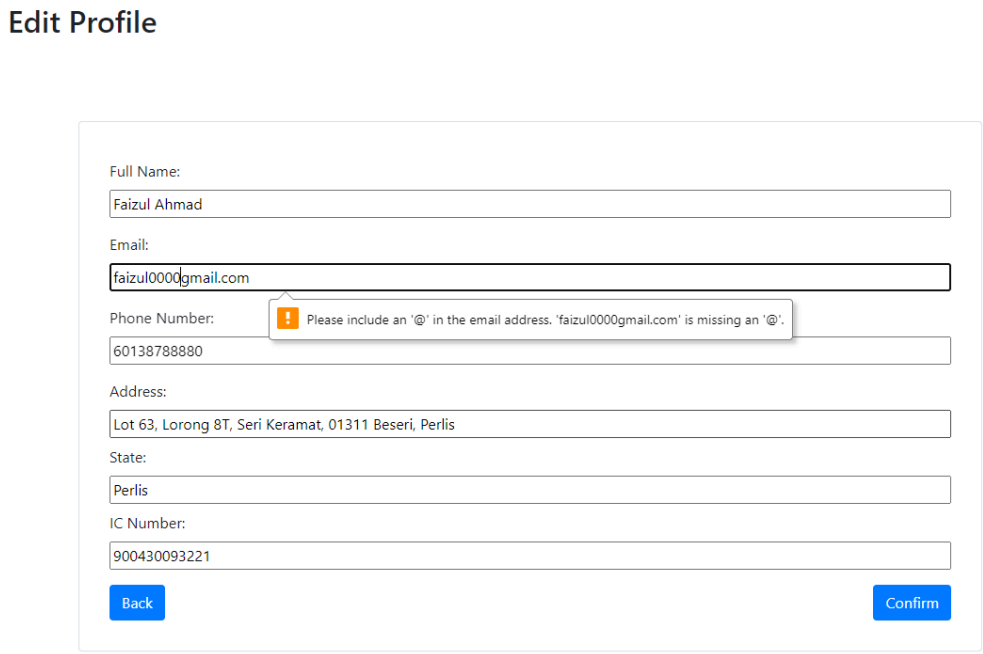


Figure 5.15: Email Does Not Include '@' Test Result

Figure 5.16 shows the test result when a delivery man tries to edit his profile but with characters in his Phone Number. An error message will pop up stating Phone Number can only contain digits.

Delivery Man Name	Delivery Man Email	Delivery ManPhone Number	Delivery Man Address	Delivery Man State	Delivery Man IC
Faizul Ahmad	faizul0000@gmail.com	60138788880abc	Lot 63, Lorong 8T, Seri Keramat, 01311 Beseri, Perlis	Perlis	900430093221

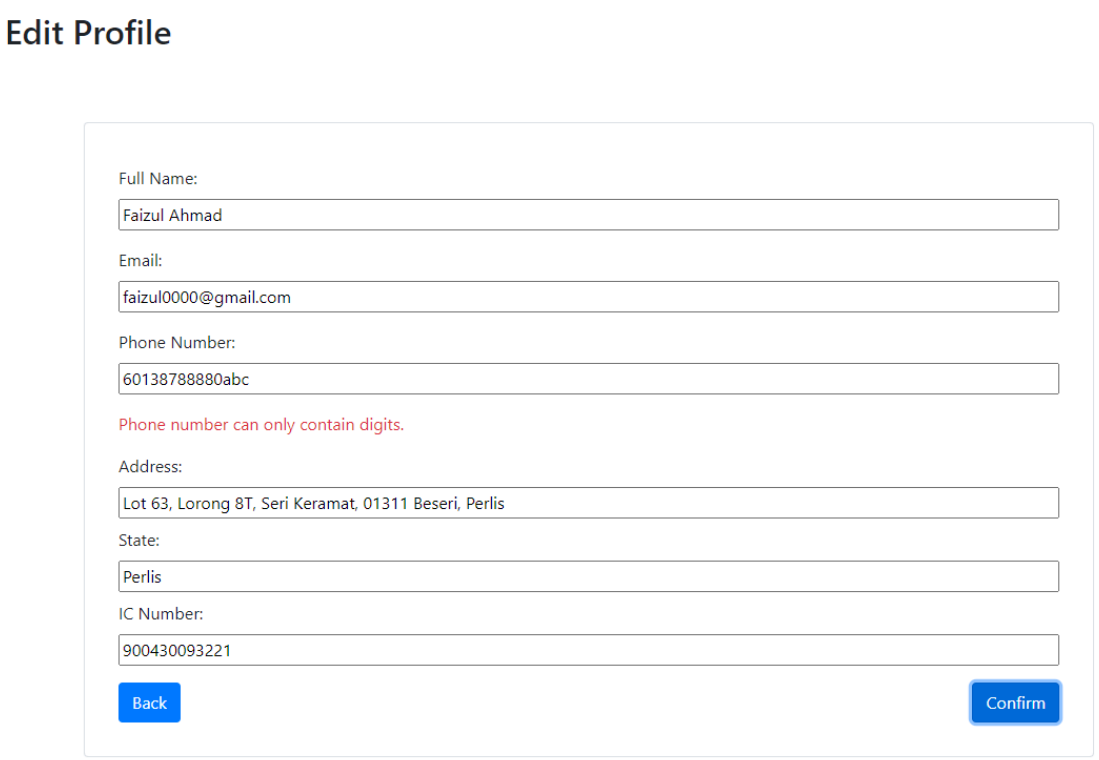


Figure 5.16: Phone Number Has Characters Test Result

Figure 5.17 shows the test result when a delivery man tries to edit his profile but with characters in their IC. An error message will pop up stating IC number can only contain digits.

Delivery Man Name	Delivery Man Email	Delivery ManPhone Number	Delivery Man Address	Delivery Man State	Delivery Man IC
Faizul Ahmad	faizul0000@gmail.com	60138788880	Lot 63, Lorong 8T, Seri Keramat, 01311 Beseri, Perlis	Perlis	900430093221my

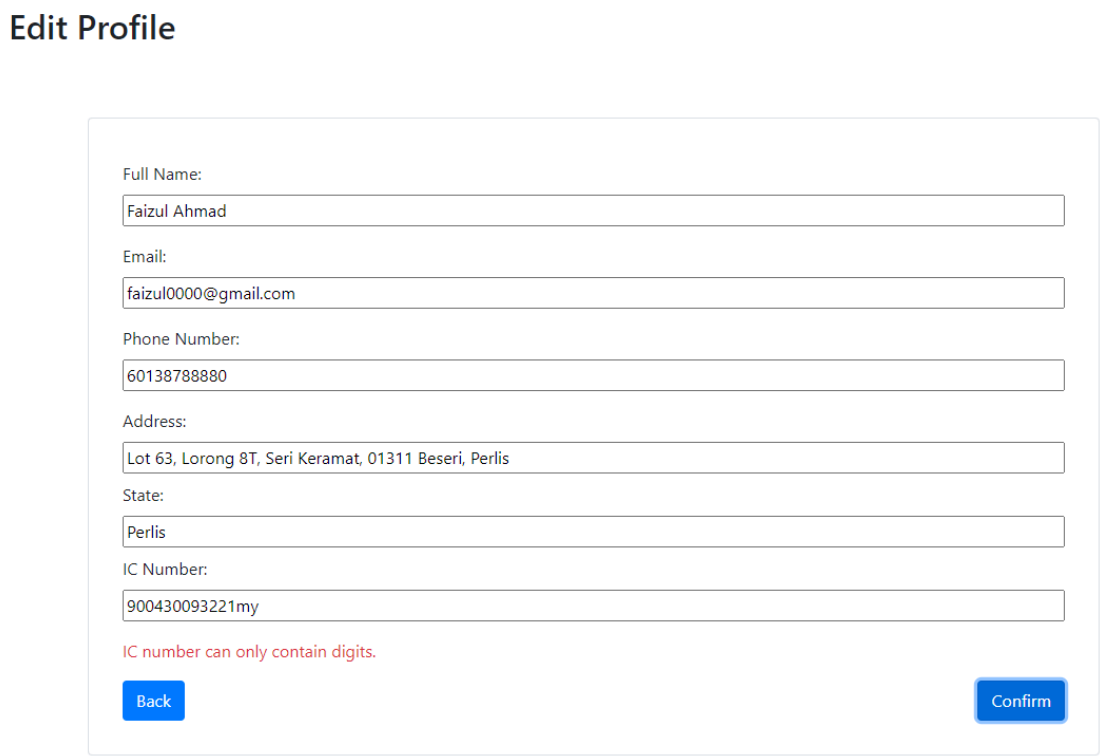


Figure 5.17: IC Has Characters Test Result

Figure 5.18 shows that the delivery man successfully edited his profile with correct information. A message saying your profile was successfully updated will pop up. Figure 5.19 shows the updated personal information in the Account Settings section. Figure 5.20 shows that the Django administration database has the updated profile details of the delivery man stored.

Delivery Man Name	Delivery Man Email	Delivery ManPhone Number	Delivery Man Address	Delivery Man State	Delivery Man IC
Faizul Mirul	faizul123@gmail.com	60138788000	14, Jalan 3F, PJU8, 10172 Tanjung Bungah, Penang	Penang	900430093221

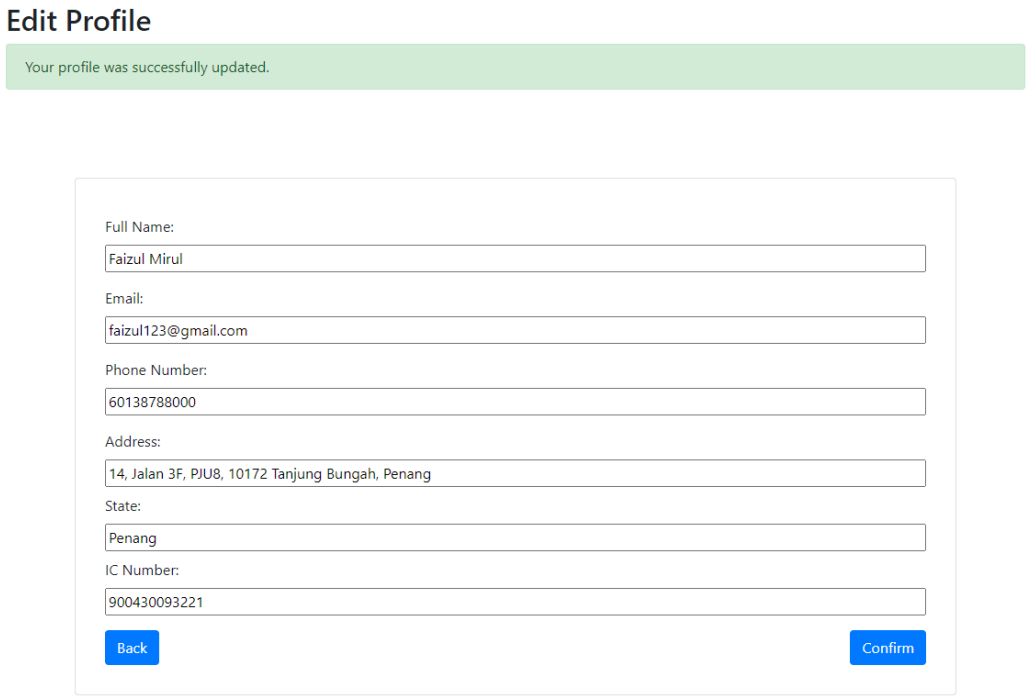


Figure 5.18: Successfully Edit Profile Test Result

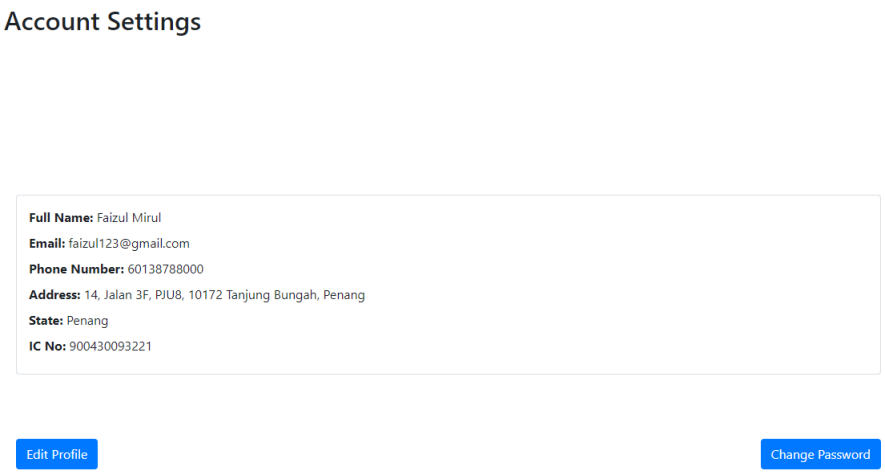


Figure 5.19: Profile at Account Settings Updated

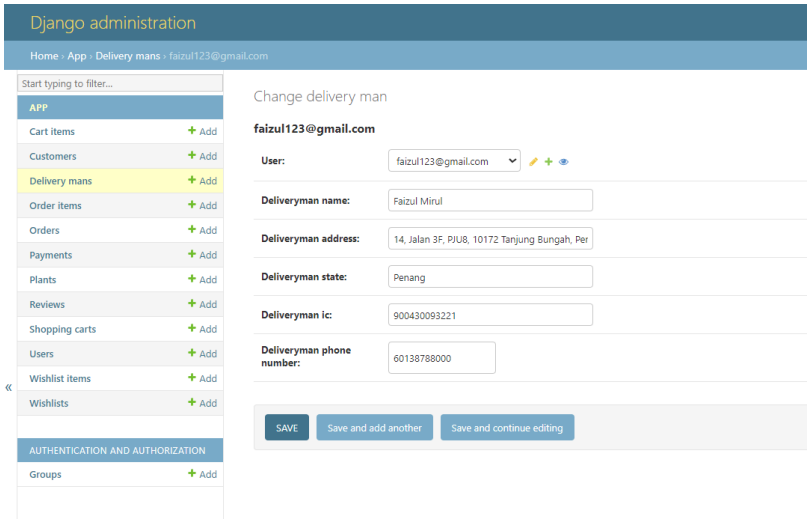


Figure 5.20: Faizul's Profile at Django Administration Updated

Accept Delivery Order Test Result:

Figure 5.21 shows that a new delivery man has no pending orders yet. A message stating that the delivery man has no pending orders at the moment will pop up.

Delivery Man Name	Delivery Man Email	Delivery Man Password	Has Pending Order	Delivery Order ID	Customer Name	Accept/Reject Order	Order Status
Faizul Mirul	faizul123@gmail.com	faizul@34**12	No	-	-	-	-

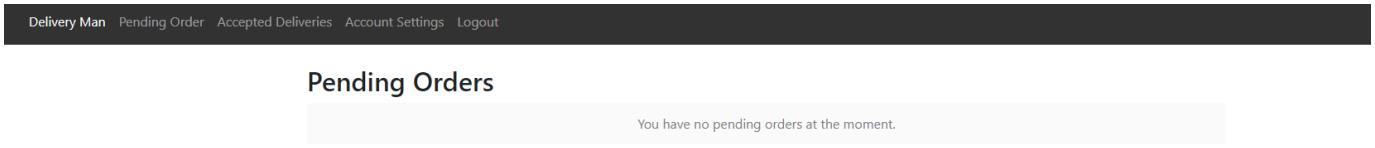


Figure 5.21: No Pending Orders Test Result

Figure 2.2 shows the pending orders before delivery man Siti accepts Delivery OrderID 48 in pending orders. Figure 5.23 shows the Django Administration where the order status for orderID 48 is Ready before being accepted by Siti. Figure 5.24 shows the pending orders section after Siti Accepts Delivery OrderID 48. This means after accepting a pending order it will be removed from the pending orders section. Figure 5.25 shows that the accepted delivery orderID 48 has moved to the accepted deliveries section after being accepted. Figure 5.26 shows that the Django Administration Order Status for Delivery OrderID 48 has changed to Out For Delivery after being accepted.

Delivery Man Name	Delivery Man Email	Delivery Man Password	Has Pending Order	Delivery Order ID	Customer Name	Accept/Reject Order	Order Status
Siti Fatimah	sitifatimah@gmail.com	qwerty156	Yes	48	Chin Yu Feng	Accept	Ready

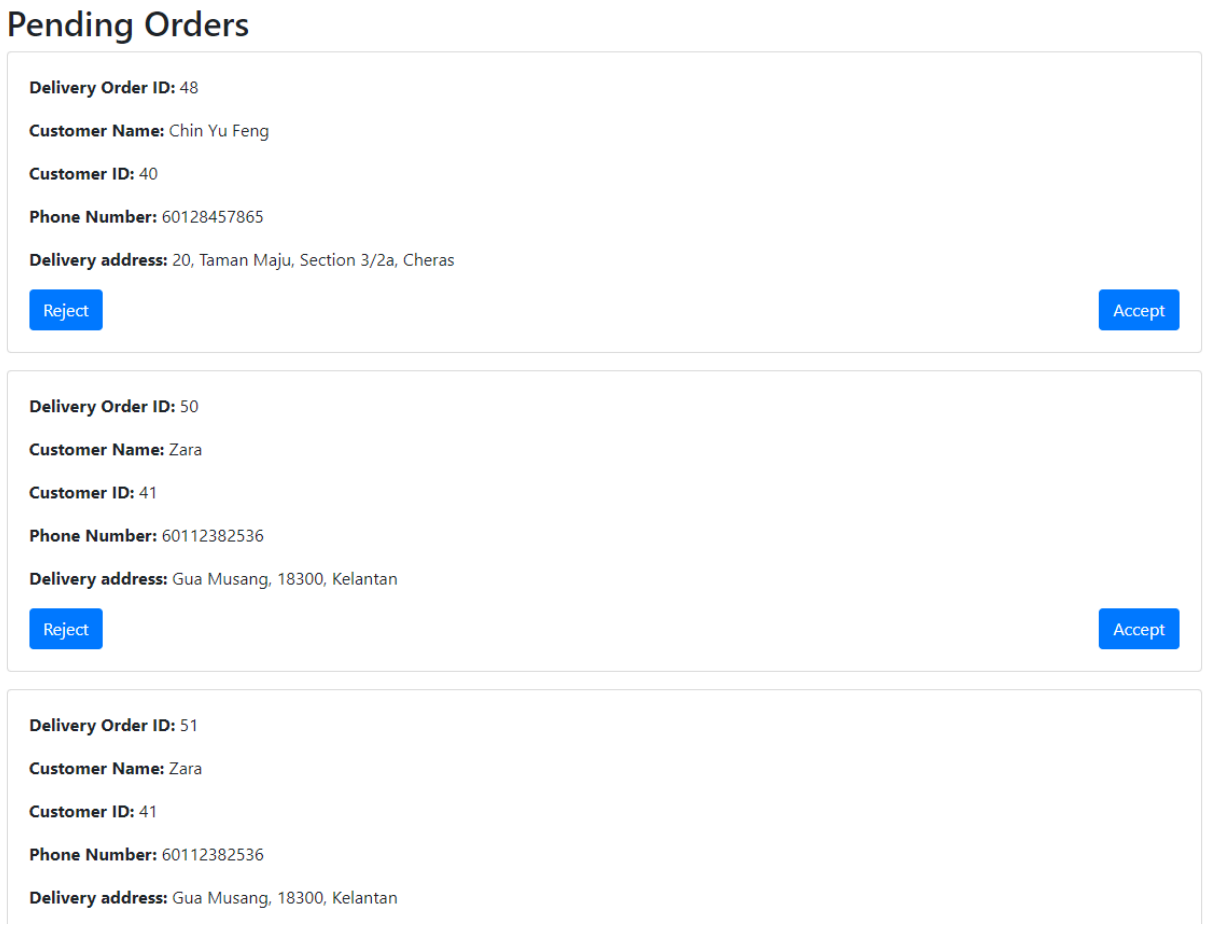


Figure 5.22: Pending Orders Before Accepting

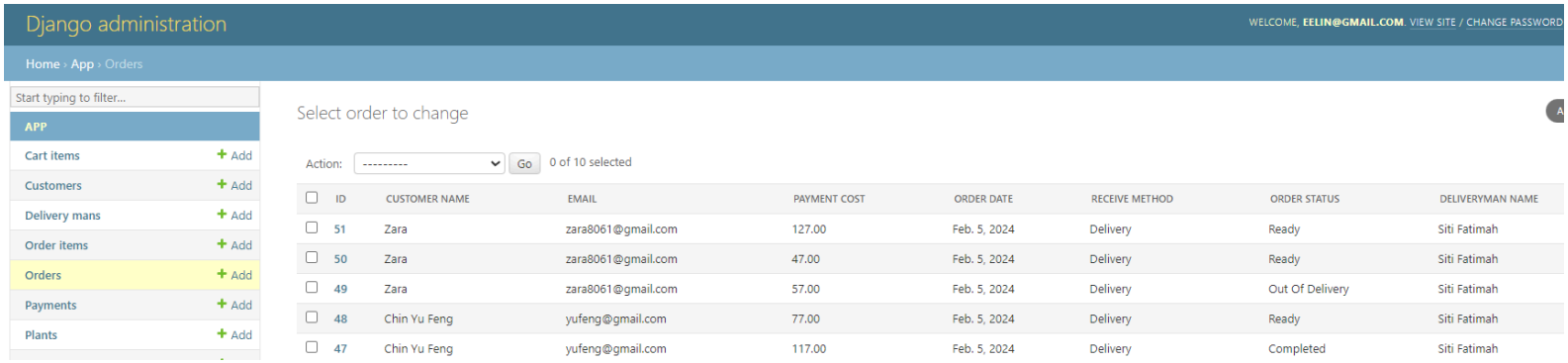


Figure 5.23: Django Administration Order Staus Before Accepting

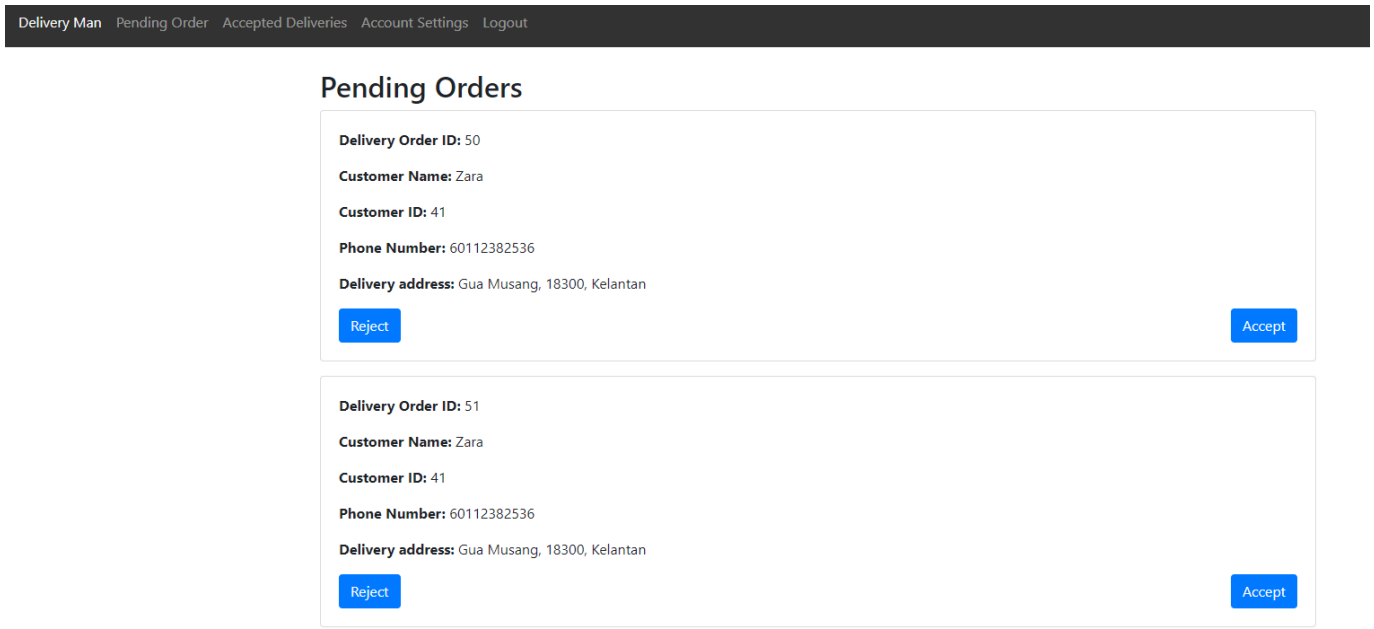


Figure 5.24: Pending Orders After Accepting Delivery OrderID 48

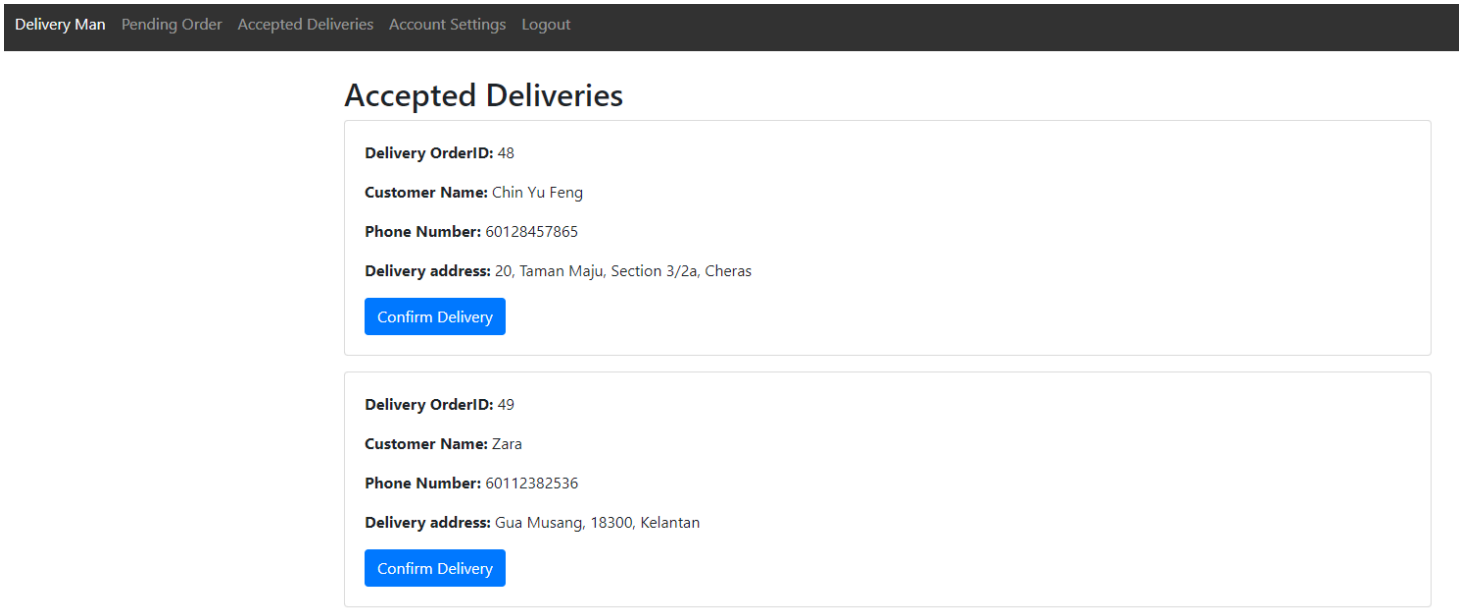


Figure 5.25: Accepted Deliveries Has Delivery OrderID 48

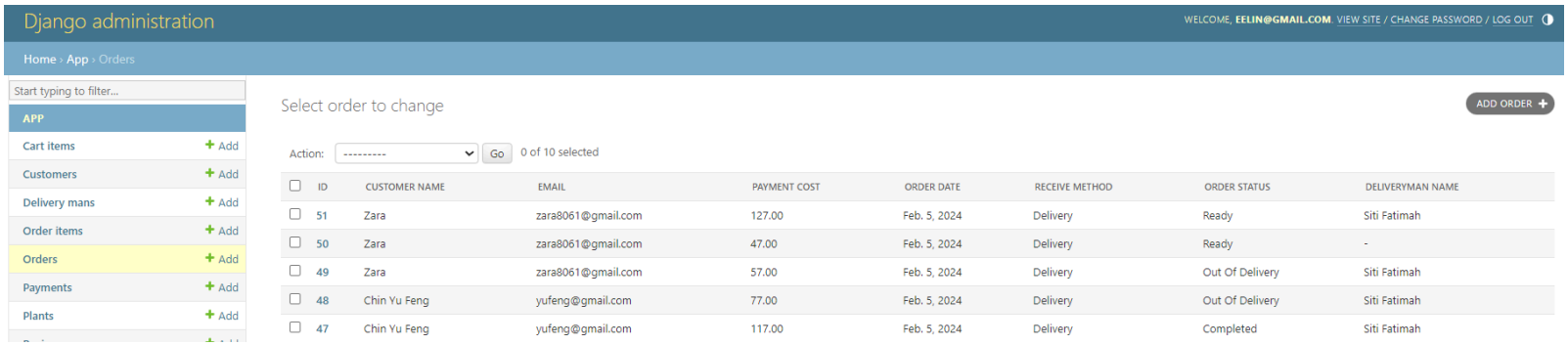


Figure 5.26: Django Administration Order Status After Accepting Delivery OrderID 48

Figure 5.27 shows the pending orders section before Siti rejects Delivery OrderID 50. Figure 5.28 shows the pending orders section after Siti rejects Delivery OrderID 50. This means that Delivery OrderID 50 was removed from the pending orders section after being rejected. Figure 5.29 shows that the Django Administration removed Siti Fatimah as the delivery man for orderID 50 after Siti rejected the order in pending orders.

Delivery Man Name	Delivery Man Email	Delivery Man Password	Has Pending Order	Delivery Order ID	Customer Name	Accept/Reject Order	Order Status
Siti Fatimah	sitifatimah@gmail.com	qwerty156	Yes	50	Zara	Reject	Ready

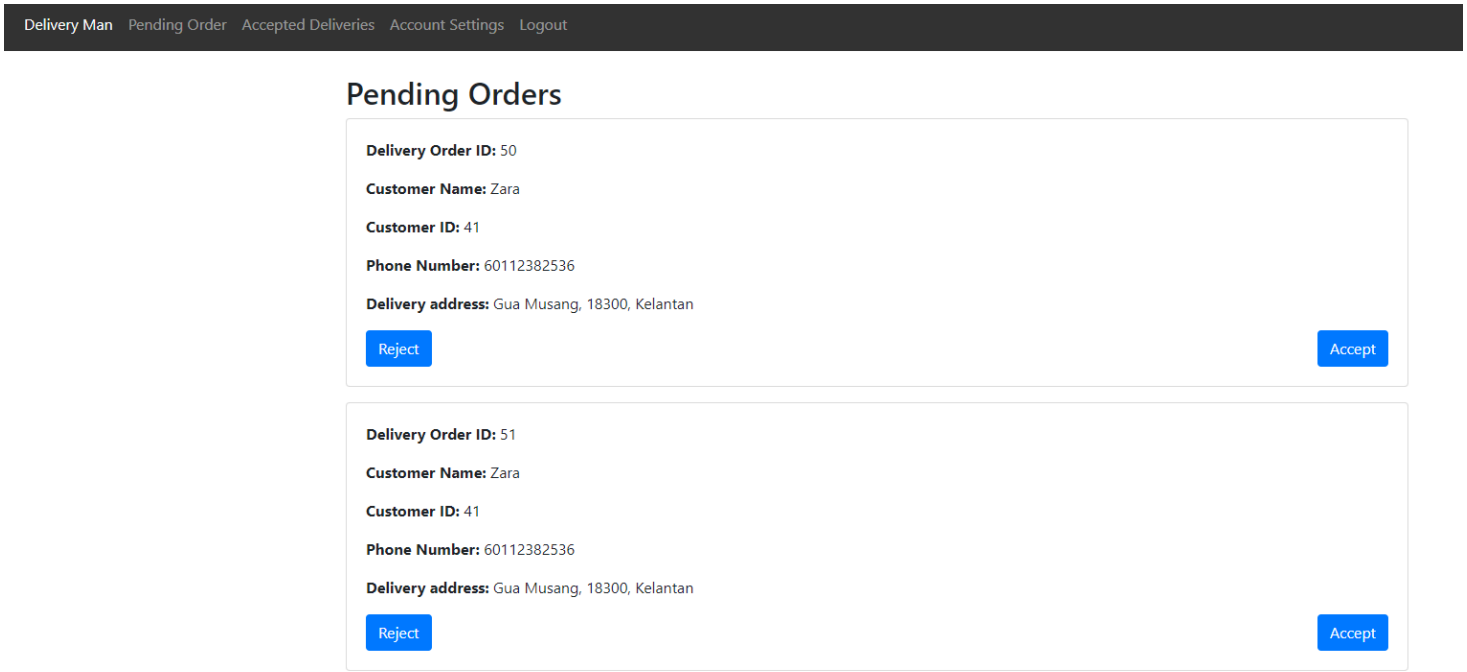


Figure 5.27: Pending Orders Before Rejecting

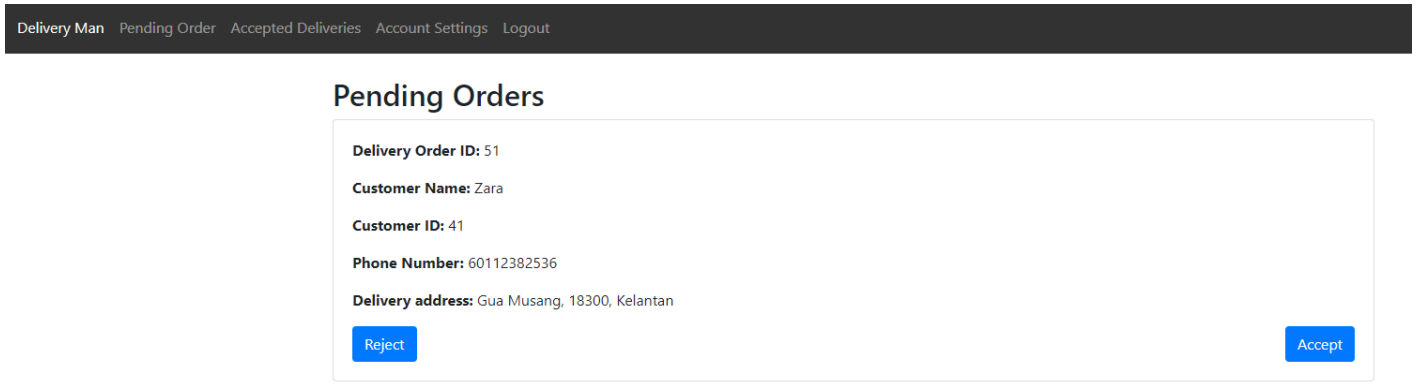


Figure 5.28: Pending Orders After Rejecting Delivery OrderID 50

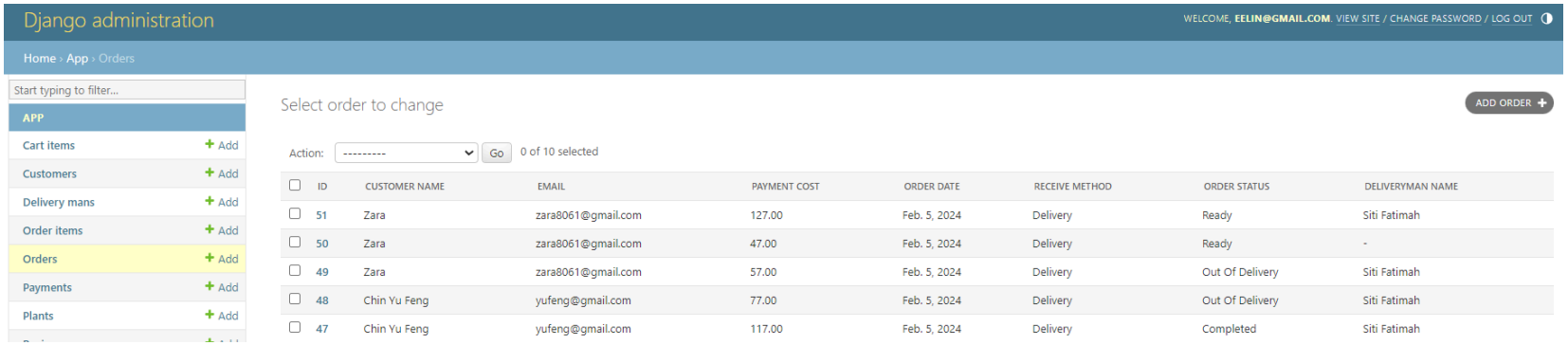


Figure 5.29: Django Administration DeliveryMan Name After Rejecting Delivery OrderID 50

View Accepted Deliveries and Confirms Delivery Test Result:

Figure 5.30 shows that a new delivery man has no accepted deliveries yet. A message stating that the delivery man has no accepted deliveries at the moment will pop up.

Delivery Man Name	Delivery Man Email	Delivery Man Password	Has Accepted Deliveries	Delivery Order ID	Customer Name	Order Status
Faizul Mirul	faizul123@gmail.com	faizul@34**12	No	-	-	-

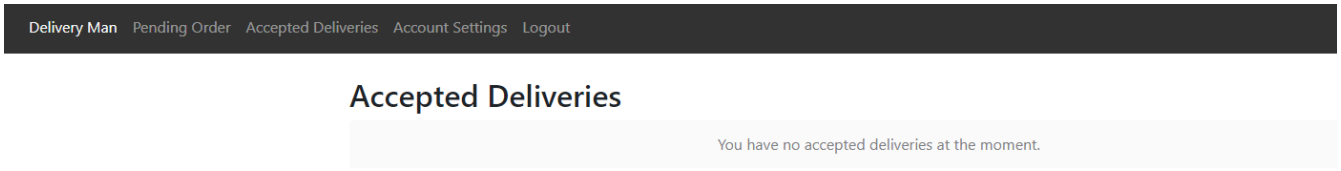


Figure 5.30: No Accepted Deliveries

Figure 5.31 shows the Django Administration order status for OrderID 48 before Siti confirms delivery. The order status for OrderID 48 is out for delivery. Figure 5.32 shows the accepted deliveries section before Siti confirms delivery for delivery orderID 48. Figure 5.33 shows the accepted deliveries section after Siti confirms delivery for delivery orderID 48. This means that the delivery order details will be removed from the accepted deliveries section after a delivery man confirms that they have delivered it. Figure 5.34 shows the Django Administration order status for OrderID 48 after Siti confirms delivery. The order status for orderID 48 changed from out for delivery to completed.

Delivery Man Name	Delivery Man Email	Delivery Man Password	Has Accepted Deliveries	Delivery Order ID	Customer Name	Order Status
Siti Fatimah	sitifatimah@gmail.com	qwerty156	Yes	48	Chin Yu Feng	Out For Delivery

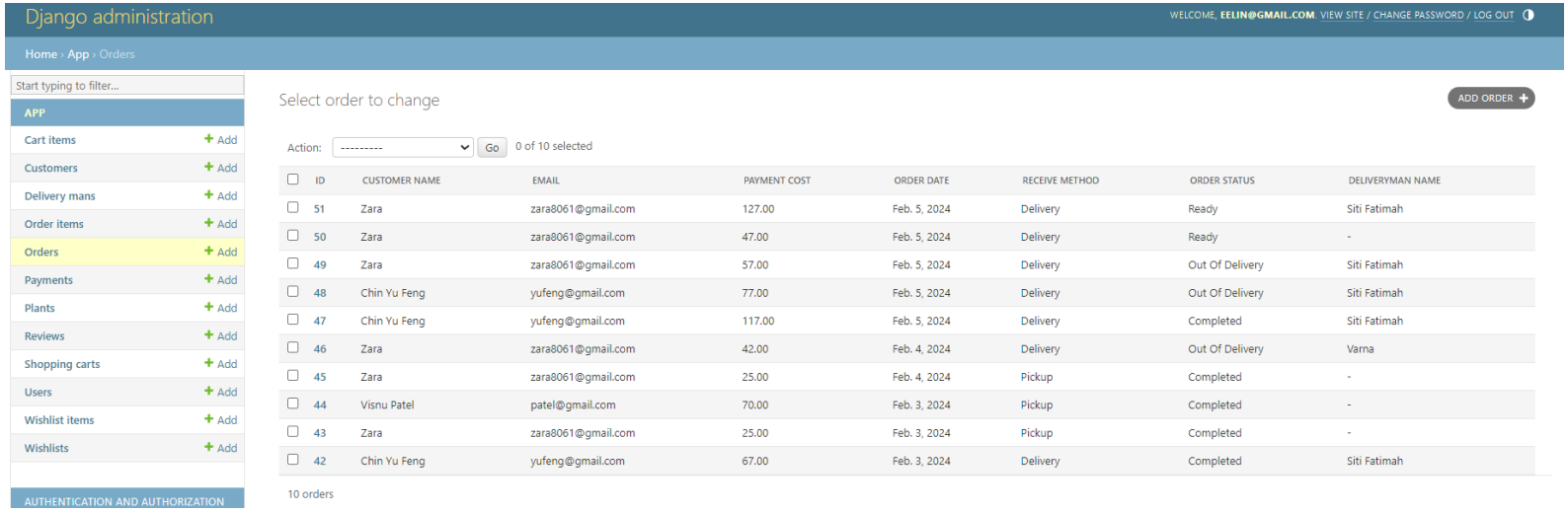


Figure 5.31: Django Administration Order Status for OrderID 48 Before Pressing Confirm Delivery

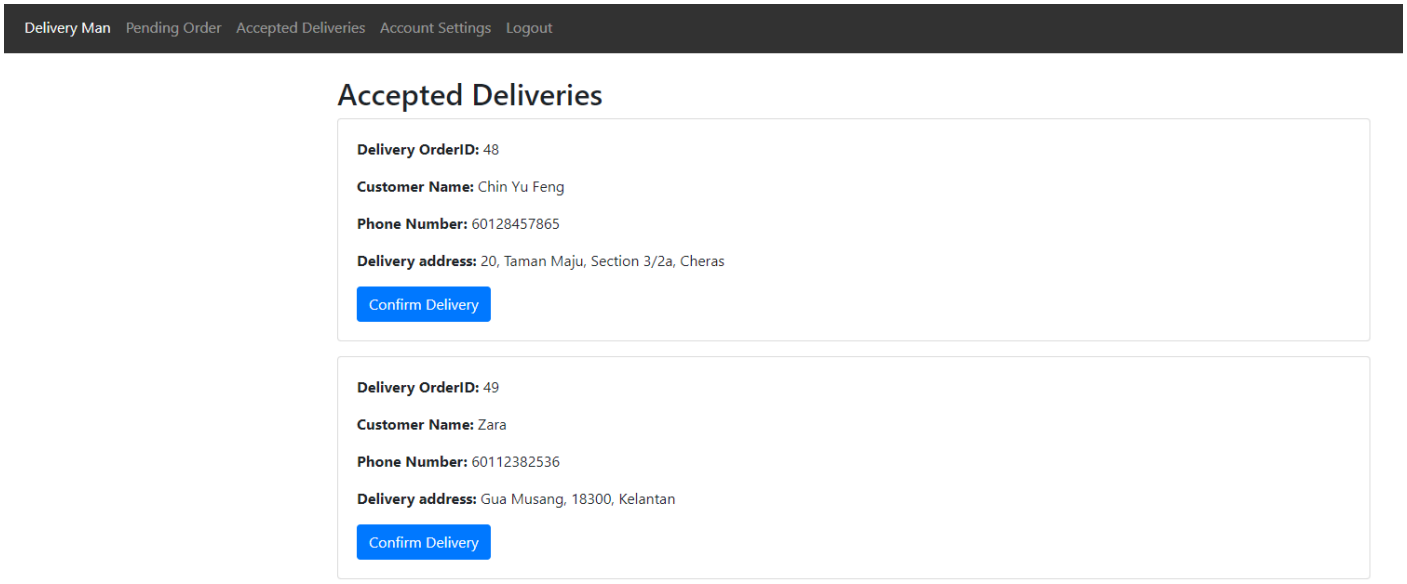


Figure 5.32: Accepted Deliveries Before Pressing Confirm Delivery

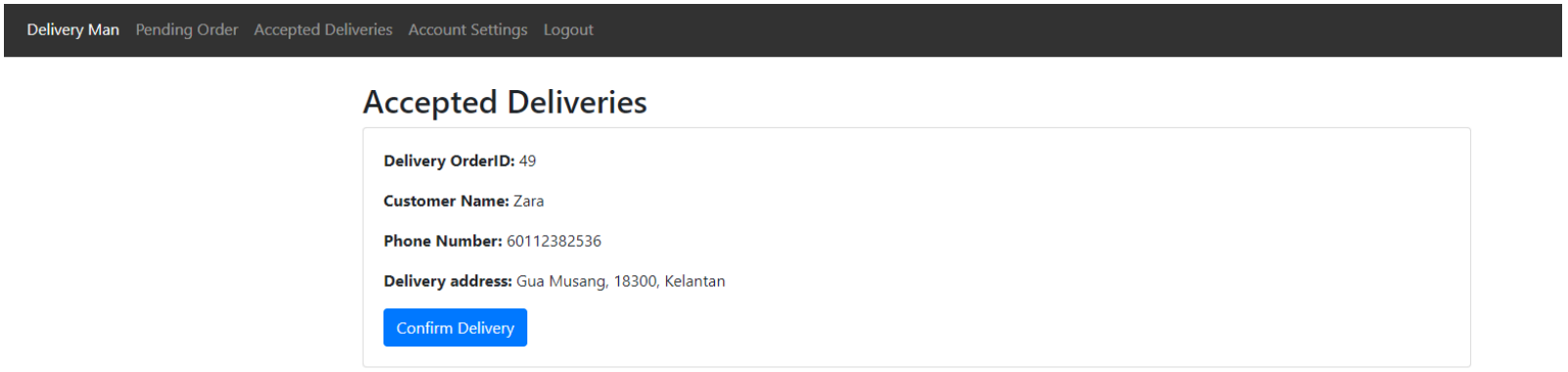


Figure 5.33: Accepted Deliveries After Pressing Confirm Delivery

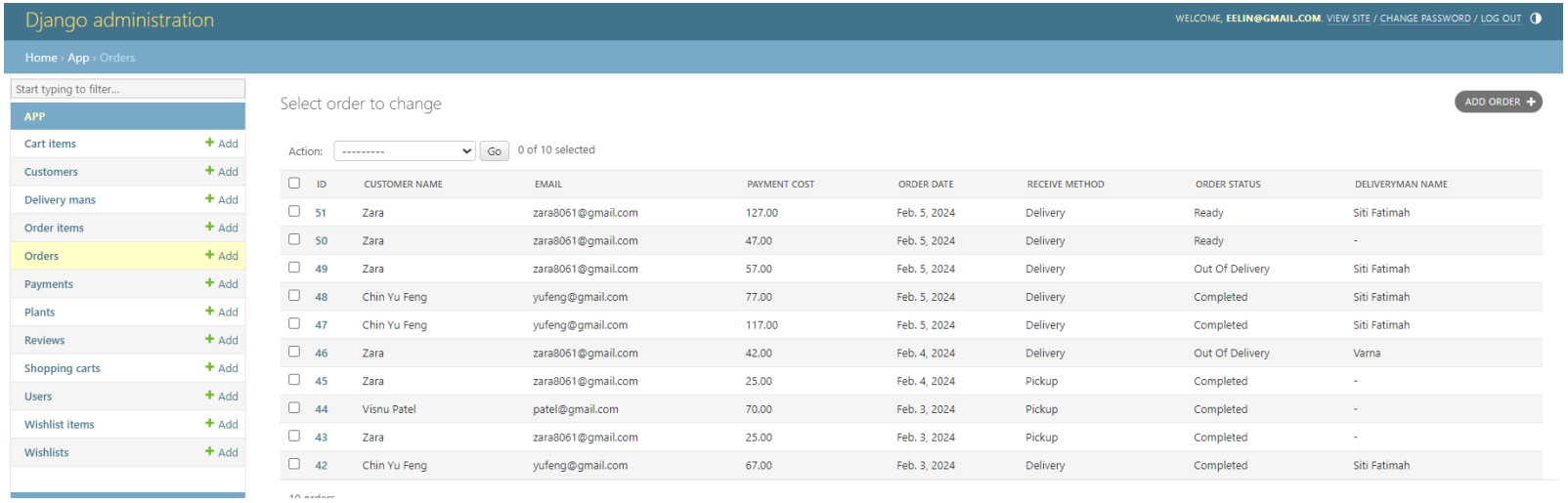


Figure 5.34: Django Administration Order Status for OrderID 48 After Pressing Confirm Delivery

6 Conclusion

6.1 Project Achievements

The nursery plant shopping system is completed and well integrated to allow all actors such as administrator, customer, guest, and delivery man to perform their specific tasks and use cases. Although I wasn't able to strictly follow our promised schedule, we managed to complete and submit the project on time and deliver a working nursery plant shopping system.

I was able to develop the delivery man's use cases in the system. The delivery man is able to create a user account as a delivery man. A delivery man is able to edit their profile and change their passwords. Lastly, a delivery man is able to accept delivery orders, view accepted deliveries, and confirm delivery.

The achievements that I was able to achieve through the development of this system are I was able to learn the basic skills of developing a system using the Django framework. I was also able to recap and improve my knowledge of Python as the last time I used it was during my foundation year. Not only that, I was also able to learn Bootstrap for an intuitive front-end experience. With all of this new and old knowledge, I believe that if I were to have to develop another system in the future, it would not be too hard and would be less time-consuming.

6.2 Quality Assurance

In our nursery plant shopping system project, our team took several steps to ensure good quality software. First, we carefully planned how each actor module should work and double-checked all the steps it needed to do. We tested every use case by different actors to make sure there were no problems.

We created a Project Plan Gantt Chart for our Nursery Plant Shopping System at the beginning of the project that includes techniques for quality and change management.When we finished an actor module, we all looked at it together to find any mistakes or things that could be better. This helped us see things we might have missed and fix them. During software development, we included test data and performed many reviews and inspections in order to ensure that we achieved our software's requirements. Not only that, we made sure that whatever we documented matched how our software works. Lastly, we always take notes and remind each other whenever we see problems or mistakes in our software. We made sure to always communicate with each other whenever there are problems encountered.

6.3 Problems Encountered

Throughout the development of our nursery plant shopping system, we encountered a number of challenges that tested our problem-solving skills and adaptability. One significant issue was we were encouraged to use Python and Django framework to develop the system. It was very challenging as we are completely new to Django and it is time-consuming as lots of research and studying is needed to be done to understand the language. Django framework has comprehensive documentation but insufficient guidelines plus there are not many references online. Thus, we have to do a lot of testing on the syntax to fully understand the language and fix our bugs.

Since it is our first time learning the Django framework and Bootstrap, the lack of experience made us fall behind our initial schedule. We were stressed out as there were also other assignments cramped together. A lot of time was used to study and understand the concept of the language and frameworks. Initially, we wanted to implement features like sort and filter. However, we had to scrape it off our plan as there wasn't much time left for us to do more extra features.

Lastly, coordinating as a team was sometimes challenging. Balancing our project with other assignments meant we had to be very organized with our time. We learned to communicate better, hold regular meetings, and set clear milestones to ensure we stayed on track. Despite these obstacles, each problem we overcame was a learning opportunity and contributed to the project's overall success.

6.4 Remarks/Comments

Overall, this project was very heavy and quite challenging for most of us as it is our first time learning how to develop a system using the Django framework. We were new to the concept and syntax of the framework and we don’t remember much about Python syntax. Therefore, we had to spend most of our time learning and understanding the implementation. However, I was happy to be able to gain exposure to developing a web application and I believe it will be a useful experience for me in the future.

After completing the project, we realized that communication is very important in team projects. When there’s a problem, we should help each other and solve the problem as soon as possible. Not only that, we also learn to be flexible in solving problems and not dwell on a problem for too long. We should solve the main problem first. Often, the solutions we initially envisioned were not the most effective, and we had to pivot and adapt our strategies. In conclusion, we are happy and grateful to be able to gain experience from this project and hope to be able to use it in the future.