National College of Ireland

BSHCSD4

Software Development

Academic Year 2022/2023

Vedran Badrov

x19458636@student.ncirl.ie

ShipIT Web Application

Technical Report

Contents

Executive Summary

This report presents the motivation, idea, and architecture behind the “ShipIT” web application. This report outlines the background of this web application and what it aims to achieve. It provides insight into the technologies and resources that have been used for the completion of requirements. In this report, the “ShipIT” web application is broken down into its functional, data, environmental and usability requirements. Accompanied are diagrams for each use case, in a prioritised order. This is followed by the description of the design and architecture of the “ShipIT” web application which includes an architecture diagram. This section goes into detailed description and aspects of main algorithms, functions, data

structures and data base used in the Delivery Web Application. The GUI is presented using screenshot of each page, and its functions explained.

1.0 Introduction

1.1. Background

I am undertaking this project because I wish to make my attempt at creating a straightforward and convenient delivery application focused on enabling its users to create new orders and have their goods on the way to the destination as quick as possible. I have personally faced uncertainty and frustration when it comes to sending and receiving deliveries while finding it hard to track my orders and be confident about their security. I want to create a system that will be easy to understand by anyone who uses the application while providing the best of the functionality it stands for. The application will provide easy access to the world of online trading, regardless of the amount of experience they have.

1.2. Aims

- Delivery Web Application aims to enable the user to create orders and organize transport for any type of good.
- Enable flexibility by letting the sender chose the pick-up and drop-off point, time of pick up.
- Enable the sender/receiver to view order details and costs.
- Allow the sender to change details about the order, such as time, start and end destinations, quantity, and weight of the goods, prior to the order start.
- It aims to enable its senders/receivers to calculate costs prior to creating an order and while the order is active, to plan for these expenditures.
- The Delivery Web Application aims to find the shortest path between the start and final destinations of an order, making currier deliveries more efficient and faster to complete.
- The application aim is to achieve a secure completion of orders between the currier and receiver.

1.3. Technology

- I will use Ruby on Rails to achieve the main functionalities of my application. Using Ruby on Rails, I will:
  1. Make use of external GEMs such as Devise for user registration and authentication.
  2. Create relationships between the senders, curriers and receivers making use of the application and the orders they are involved with.

- HTML, CSS, Bootstrap for the design of the GUI.

- System Database for storing sender/receiver accounts, currier account, parcel information (Includes start destination, destination, stops if crossing national borders, tracking number, type of shipment, volume of shipment, weight of shipment, journey length)

1.4. Structure

1. **System** –

- Functional requirements of the ShipIT web application, in a ranked order.
- Functional requirements describe, in detail, capabilities and functions of the ShipIT web application.
- Each functional requirement is accompanied by a description, use case and a corresponding diagram.
- The system part goes further to into data requirements where the database is broken down to each variable and its use explained within the ShipIT system.
- Furthermore, the user requirements provide the ideas of what the ShipIT system enables its users to accomplish while using the ShipIT web application and what the goals the users may accomplish with their actions.

2. **Conclusions –**

- Goes over the strengths, weaknesses, advantages, and disadvantages of the ShipIT web application.
- Outlines what are its strongest points but also criticizes the weaknesses, which would be the focus of further development.

3. **Further Development or Research –**

- Outlines what the main focuses of further development would be and what direction ShipIT web application would take.

4. **Appendices –**

- Includes the initial Project Proposal and the Monthly reflective journals.
- Outlines the thoughts and progress through the development of ShipIT.

2.0 System

2.1. Requirements

2.1.1. Functional Requirements

1. Creating a new order – enable the user to create and submit new orders.

2. View “My Orders” – enable the user to list and view their orders currently in progress.
3. Cost calculator – enable users to calculate the total cost of the order based on the volume, weight, type of goods and distance between start and end destinations.
4. Currier Portal – Provides Currier Users to use it as a hub to locate orders that are pending as well as view their accepted, active, cancelled or completed orders.
5. Currier Accepted Orders – Enable curriers to view all orders that they are interacting with. All orders that have order status of accepted and in progress and belong to the currier are viewed in this page.
6. Tracking an order – enable users to see the progress of their order and its current order status. This includes an order number.
7. Manage orders – enable users to apply for changes or cancellation of orders. Enable curriers to manage orders that they interact with by beginning, cancelling, and completing orders.
8. Receive items – enable users to be receivers of orders and view information about any incoming orders. The most important details about the order are displayed to the receivers such as order status and order number.
9. Orders history – allow users to see the history of all their previous orders.
10. Registration and Log-in – Enable a user to create an account.
11. Manage my account – enable the user to view and change their account and personal details.

2.1.1.1. “Creating a new order” Use Case Diagram

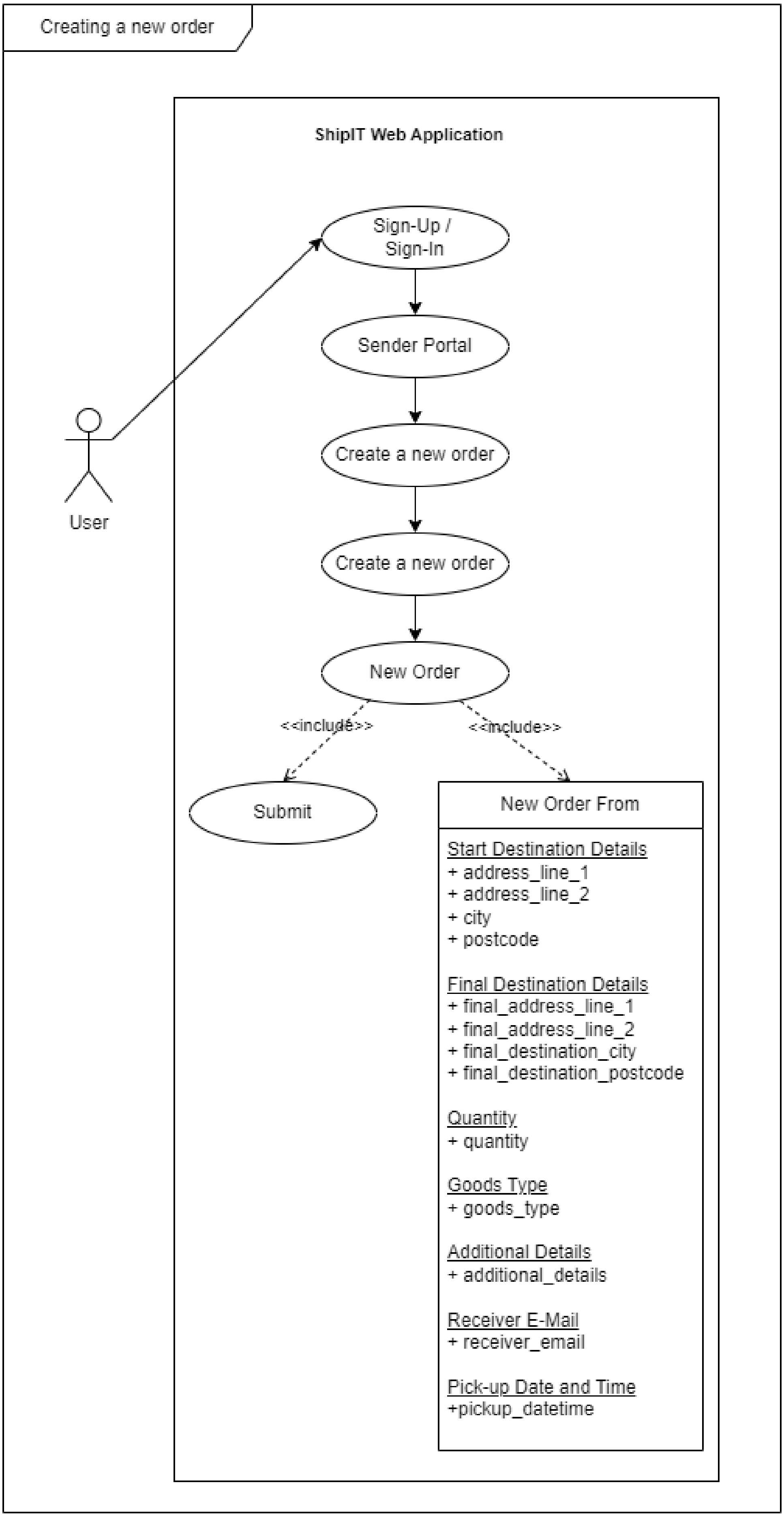

Creating a new order
ShipIT Web Application
Sign-Up / Sign-In
Sender Portal
Create a new order
Create a new order
New Order
User
<<include>>
<<include>>
Submit
New Order From
Start Destination Details
+ address_line_1
+ address_line_2
+ city
+ postcode
Final Destination Details
+ final_address_line_1
+ final_address_line_2
+ final_destination_city
+ final_destination_postcode
Quantity
+ quantity
Goods Type
+ goods_type
Additional Details
+ additional_details
Receiver E-Mail
+ receiver_email
Pick-up Date and Time
+pickup_datetime

Figure 1Creating a new order

2.1.1.2. Requirement 1 Creating a new order.

The heading of this section should read, e.g., “Requirement 1: User registration” or “Requirements 1: Participant takes test”.

2.1.1.3. Description & Priority

A description of the requirement and its priority. Describes how essential this requirement is to the overall system.

Enabling users to create new orders is very important in a system with one its main goals being the initialization of orders. Without such a feature a sender could possibly render this application almost useless.

2.1.1.4. Use Case

Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.

Scope

The scope of this use case is to enable the sender to create a new order.

Description

This use case describes the interaction between the sender and the web application and the process the sender must go through to create a new order.

Use Case Diagram

Flow Description

Precondition

The system is in initialisation mode when the user has access to the web application and is on the main home page.

Activation

This use case starts when a sender logs-in or creates account and proceeds to orders page.

Main flow

1. The system is running.
2. The Sender creates a new account or is logged-in
3. The system authenticates the account.
4. Sender gains access to the “Sender portal’ (See A1)
5. The sender wishes to create a new order and clicks on “Create new order”.
6. The system opens a the “New Order” page that includes the New Order Form.
7. The sender provides details of the order by filling out a provided form.

8. The system checks the validity of the information provided (See A2) (See E1)
9. The sender submits the order.
10. The sender is redirected to the "Confirmation Page" where the user is displayed with the new order alongside a generated "Order number" for the specific order.

Alternate flow

- A1: Unauthenticated account
    1. The system renders the account unauthenticated or non-existent.
    2. The sender must create a new account or apply for account recovery.
    3. The use case continues at position 4 of the main flow.

- A2: Unexpected order details
    1. The system does not successfully validify the information provided for the order to be created.
    2. The user must check the information and re-enter it where needed.
    3. The use case continues at position 9 of the main flow.

Exceptional flow

- E1: Invalid information
    4. The system information provided is not successfully validated.
    5. The does not provide the required fields of the form.
    6. The use case continues at position 10.

Termination

The system redirects the user to an order confirmation page.

Post condition

The system goes into a wait state.

2.1.1.5. "My orders" Use Case Diagram

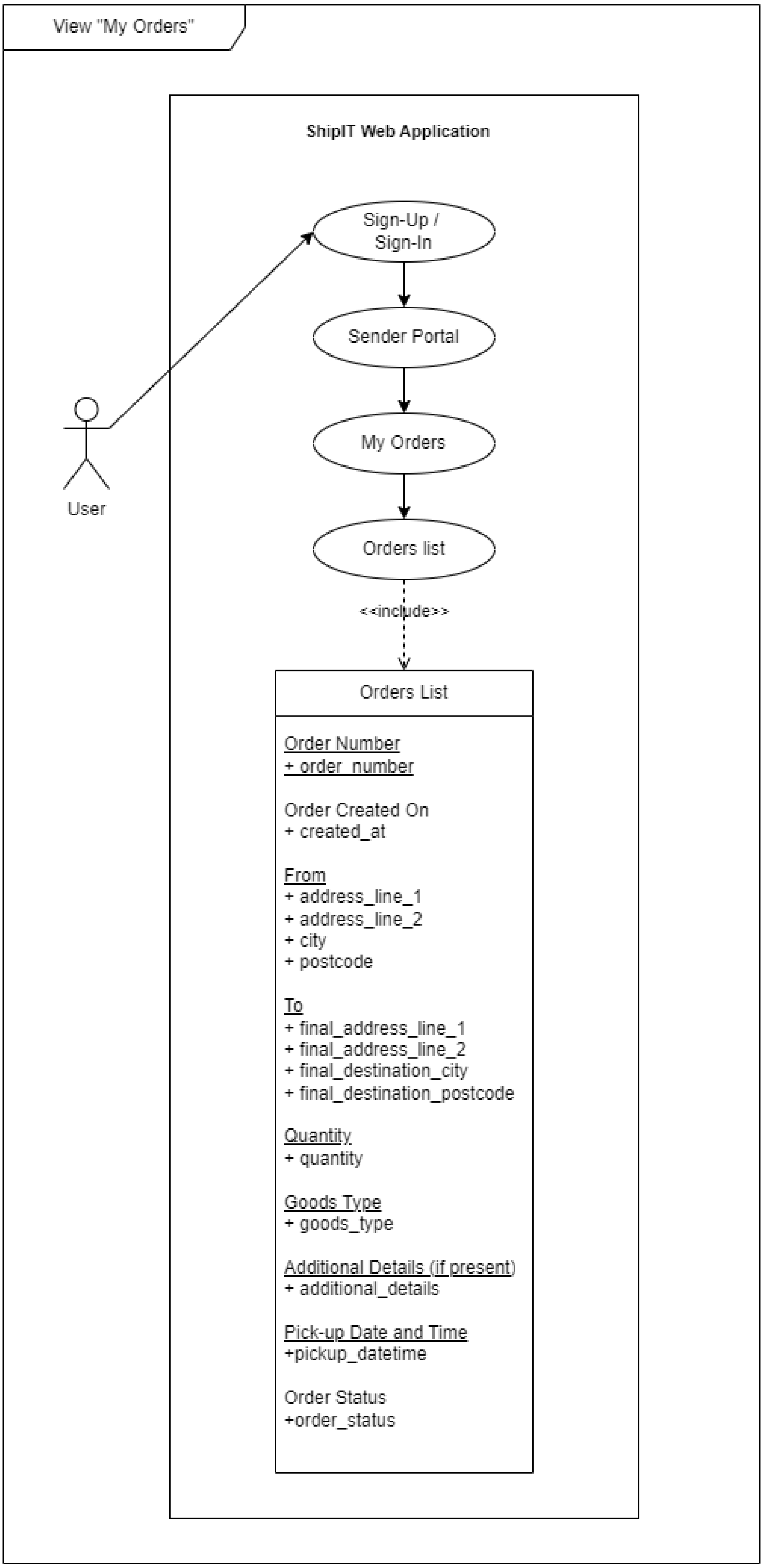

View "My Orders"
ShipIT Web Application
Sign-Up / Sign-In
Sender Portal
My Orders
Orders list
<<include>>
User
Orders List
Order Number
+ order_number
Order Created On
+ created_at
From
+ address_line_1
+ address_line_2
+ city
+ postcode
To
+ final_address_line_1
+ final_address_line_2
+ final_destination_city
+ final_destination_postcode
Quantity
+ quantity
Goods Type
+ goods_type
Additional Details (if present)
+ additional_details
Pick-up Date and Time
+pickup_datetime
Order Status
+order_status

Figure 2 View "My Orders"

2.1.1.6. Requirement 2 My orders

2.1.1.7. Description & Priority

Description & Priority

This is second highest priority as it closely ties with creating new orders. The user must be able to view their orders in progress especially if they intend to verify if they correctly created an order. They also gain access to information such as Order Number of the Order, when the order was created and what the current status of the order is.

2.1.1.8. Use Case

Use Case

Requirement 2 – View "My Orders"

Scope

The scope of this use case is to enable the sender to view all their currently active orders.

Description

This use case interaction between the sender and the system and the process they must go through to view their active orders.

Flow Description

- **Precondition**

  The system is in initialisation mode when the user has access to the web application and is on the main home page.

- **Activation**

  This use case starts when a sender logs-in or creates account and proceeds to orders page.

- **Main flow**

  1. The system is running.
  2. The Sender creates a new account or logs in
  3. The system authenticates the account.
  4. Sender gains access to the "Orders' page
  5. The Sender wishes to list all their active (pending, accepted or in progress) orders and clicks on "My Orders" page.
  6. The system opens a the "My Orders" page listing all active orders (See A1) (See A2) (See E1)

7. The sender is satisfied to verify the information and the status of their order.
8. The sender leaves the “My Orders” page.

Alternate flow

- A1: Change details
  1. The sender is not satisfied with the information included in the order and they must apply for a change of details.
  2. The system opens a form.
  3. the user to specifies the details they desire to be changed.
  4. The changes are validated, and the details of the order have been changed.
  5. The use case continues at position 7.

- A2: Decline an order.
  1. The sender wishes to decline an active order.
  2. The system opens an order declining submission page.
  3. The sender must provide valid account details.
  4. The system declines the order and sends it to the history of orders table.
  5. The use case continues at position 8.

Exceptional flow

- E1: Order not present
  1. The sender does not see a currently active order or an order that they just created.
  2. The system provides contact information for the support of the web application.
  3. The use case continues at position 8.

Termination

The sender/receiver leaves the “My Orders” page.

Post condition

The system goes into a wait state.

2.1.1.9. “Cost Calculator” Use Case Diagram

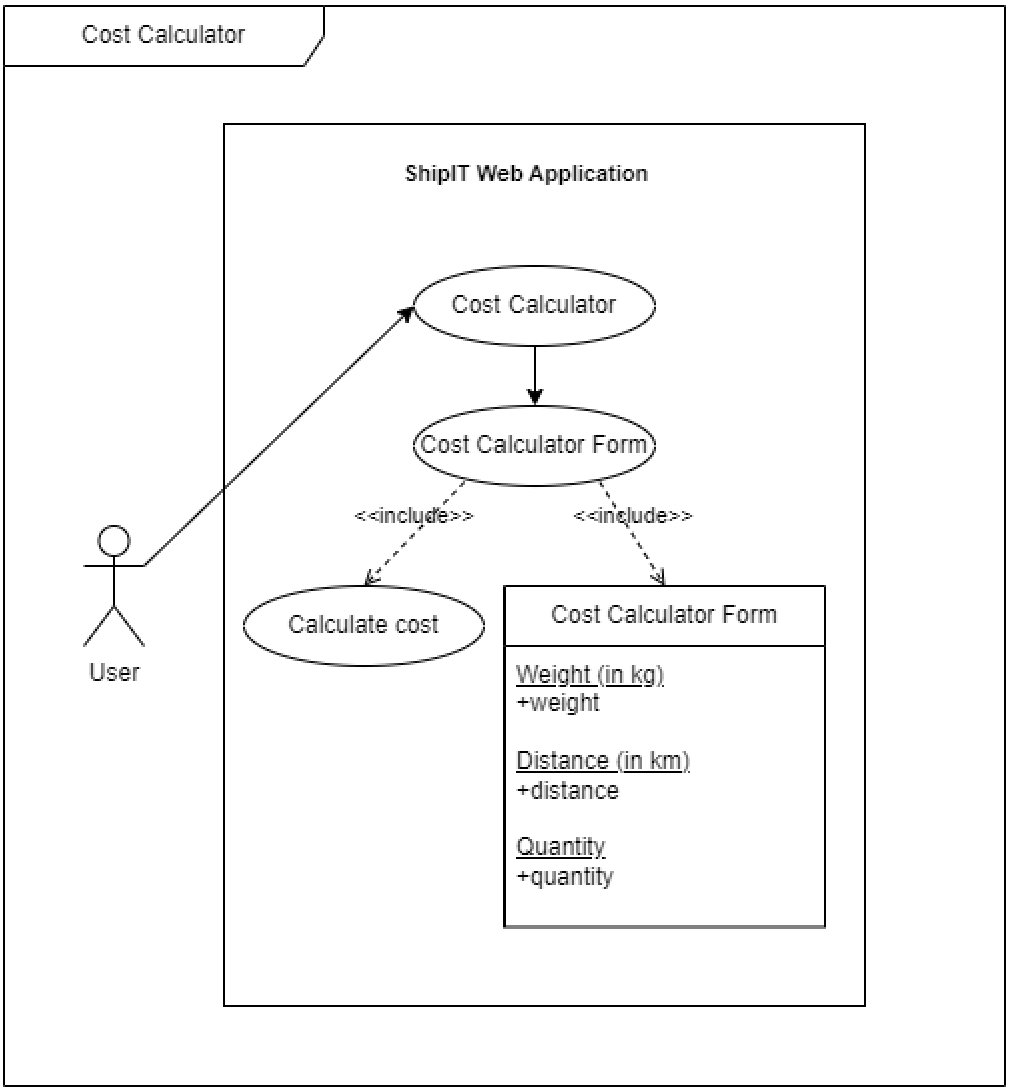


Figure 3 Cost Calculator

2.1.1.10. Requirement 3 Cost Calculator

2.1.1.11. Description & Priority

The cost calculator comes in at the third priority. It is an important part of the application as it provides a unique way to interact with the ShipIT web application. The cost is calculated based on pre-set costs by ShipIT based on weight, distance, and quantity of items.

2.1.1.12. Use Case

Use Case 3 – Cost Calculator

Scope

The Scope of this use case is to show how a user interacts with the Cost Calculator as a part of the ShipIT web application and makes use of it for costs planning.

Description

This use case goes through the Main, Alternate and Exceptional flows for the Cost Calculator, alongside the preconditions, termination, and post condition.

Flow Description

- **Precondition**
  The User has the web application running on their system.
- **Activation**
  The use case activates when the user goes to the Cost Calculator page on the web application.
- **Main Flow**
  1. The system is running.
  2. The user navigates to the navigation bar and selects the "Cost Calculator".
  3. The user is redirected to the Cost Calculator page and presented with the calculator form.
  4. The user fills out the calculator form with details.
  5. The user selects the "Calculate cost" button**.**
  6. **The application calculates the cost based on pre-set costs of weight, distance, and quantity, multiplying them with the user inputs.**
  7. **The Cost calculator outputs the cost for the inputted details.**

- **Alternate Flow**
  1. The user creates a new account or logs-in with an existing one.
  2. The user navigates to the navigation bar and selects the "Cost calculator" button.
  3. The use case continues at Main flow 3.

- **Exceptional flow**
  1. The user navigates to the "Cost Calculator" in the navigation bar.
  2. The user is presented with the calculator form.
  3. The user inputs a letter instead of a number for Weight, Distance or Quantity.
  4. The system notifies the user that the input must be a number.
  5. The user fixes the wrong input, and the use case continues at Main Flow 5.

- **Termination**

  The use case ends when the user views the calculated cost based on the inputted values.

- **Post Condition**

The system is in a wait state

2.1.1.13. “Currier Portal” Use Case Diagram

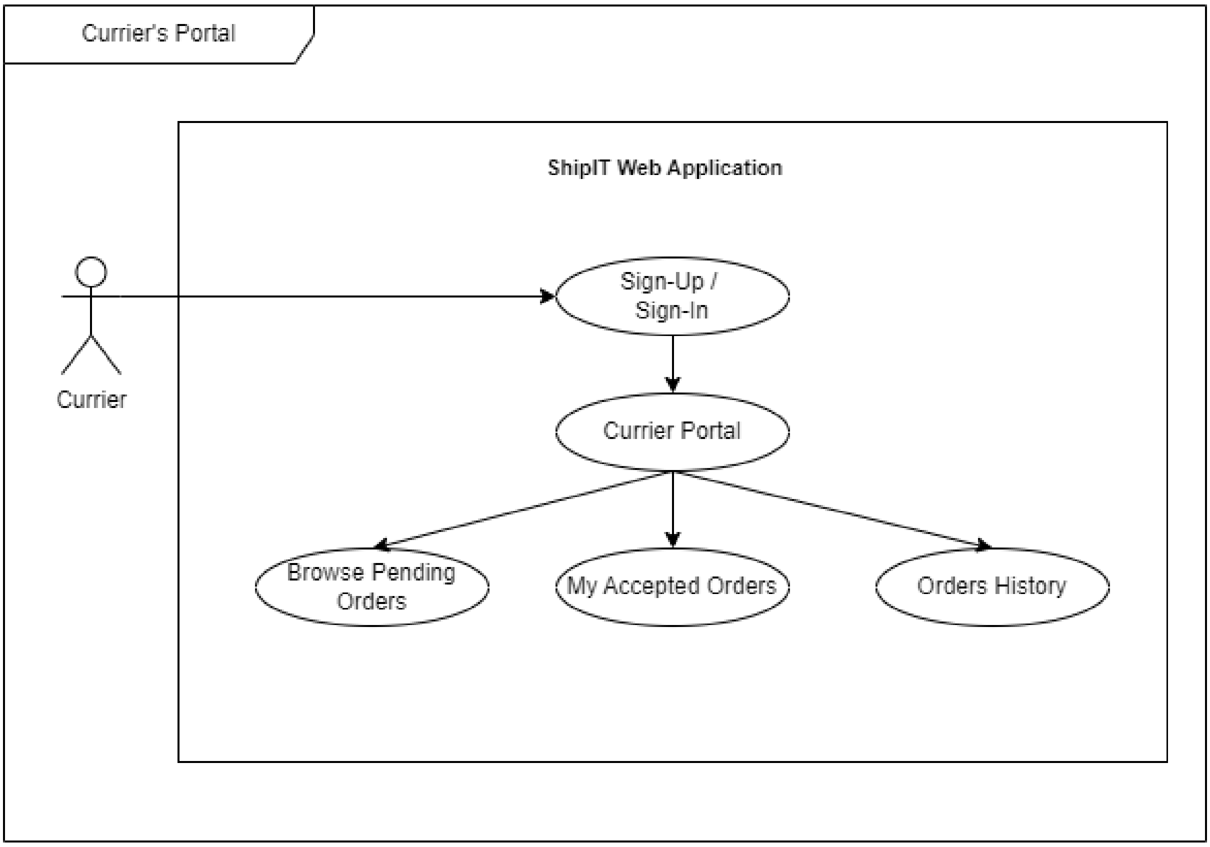


Figure 4 Currier Portal

2.1.1.14. Requirement 4 Currier Portal

2.1.1.15. Description & Priority

Currier portal comes at the 4th place in importance and enables the web application to have multiple user roles interacting with it. Users with the role of currier have their own dedicated portal where they may look for newly created pending orders or view orders that belong to them and are accepted or in progress. The curriers may also view their past orders in a separate tab where the order status is completed or cancelled.

2.1.1.16. Use Case

Use Case 4 – Currier Portal

- **Scope**

  Enable the users with role of “currier” to view “Pending Orders”, “My accepted Orders” and “Orders History”.

- **Description**

  This use case highlights how curriers interact with the ShipIT web application and where they may locate pending, accepted, in progress, cancelled and completed orders.

- **Flow Description**

- **Precondition**
  The user account must have role of "currier" and the user must be logged in.

- **Activation**
  The use case begins when the user is logged in and wants to view their currier orders.

- **Main flow**

  1. **The currier navigates to the navigation bar and locates the "Currier portal" button.**
  2. **The currier selects the "Currier portal" and is redirected to the Currier portal page.**
  3. **The currier can select to "Browse pending orders", "My Accepted Orders", or "Orders History".**

- **Alternate flow**

  1. **The currier navigates to the home page of the ShipIT web application.**
  2. **The currier navigates to the "Currier portal" button on the home page.**
  3. **The use case continues Main flow 3.**

- **Exceptional flow**

  1. **The current user is not currently logged-in with an account that has the role "currier".**
  2. **The user can not view the link to currier portal in the navigation bar or the home page.**
  3. **The user may check their role on top of the home page.**
  4. **In case the role displayed is not currier, the current user does not have permission to access the currier portal pages.**

- **Termination**

  Once the currier is satisfied that they have the option to view pending orders, active orders and orders history, this use case ends.

- **Post condition**

  The system goes into a wait state.

2.1.1.17. “Browse Pending Orders” Use Case Diagram

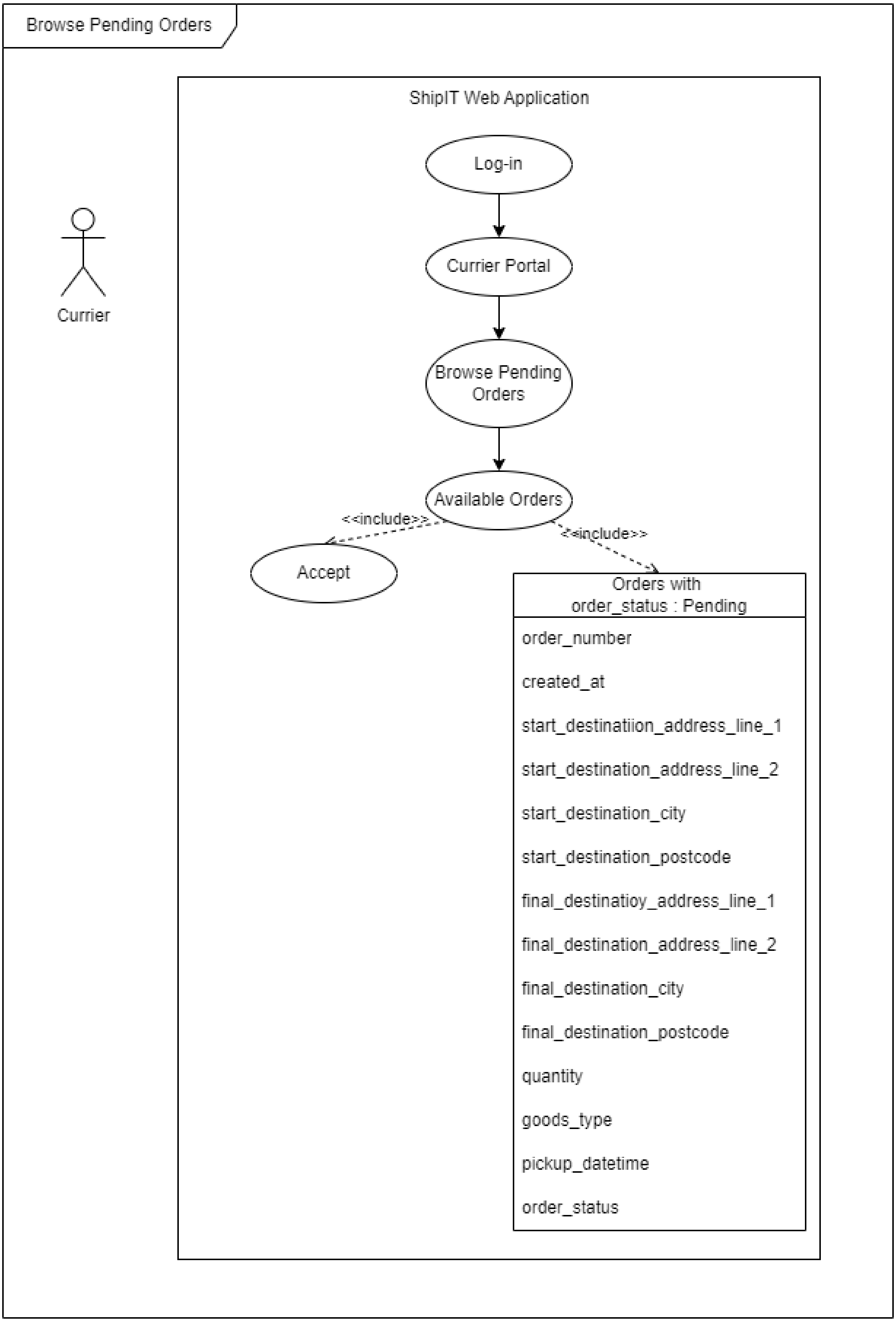


Figure 5 Browse Pending Orders

2.1.1.18. Requirement 5 Browse Pending Orders

2.1.1.19. Description & Priority

The 5th Requirement is to enable curriers to Browse all pending orders that are currently available. This connects users and curriers, enabling curriers to chose what they wish to work on while making use of the ShipIT web application. They may see all pending orders in this tab alongside important details of each order.

2.1.1.20. Use Case

- **Scope**

  The scope of the use case is how the user navigates and views all currently pending orders.

- **Description**
  This use case goes through how the currier navigates to all currently pending orders and how they interact with what is displayed.

- **Flow Description**

- **Precondition**
  The user must be logged-in with an account that has the role of "currier" and be able to access the Currier Portal.

- **Activation**
  The use case begins when the currier wants to view orders that are currently pending and need to be accepted.

- **Main flow**
  1. The user navigates to the curriers portal and selects the "Browse Pending Orders".
  2. The system redirects the user to "Available Orders" page, where orders with current status of "pending" are displayed.
  3. The currier may browse these orders and accept anyone that they wish to accept.
  4. The currier accepts an order.
  5. The system redirects the currier to "My Orders" page where the currier may see their accepted order with an updated order status of "Accepted".

- **Alternate flow**
  1. The currier navigates to the home page where they can see a "Browse pending orders" button and selects it.
  2. The system redirects the currier to browse "Available Orders" accompanied with an accept button.
  3. The use case continues at Main Flow 3.

- **Exceptional flow**
  1. The user accepts an order that they did not intend to accept.
  2. The user navigates to “My orders” page and selects “Manage Order”.
  3. The system redirects the user to “Manage” page of the specific order.
  4. The currier selects the “Cancel” button and the order is removed from “My orders” page and is no longer “Accepted”.

- **Termination**

The user is satisfied with browsing pending orders and leaves the page.

- **Post condition**

The system goes into a wait state.

2.1.1.21. “My Accepted Orders” Use Case Diagram

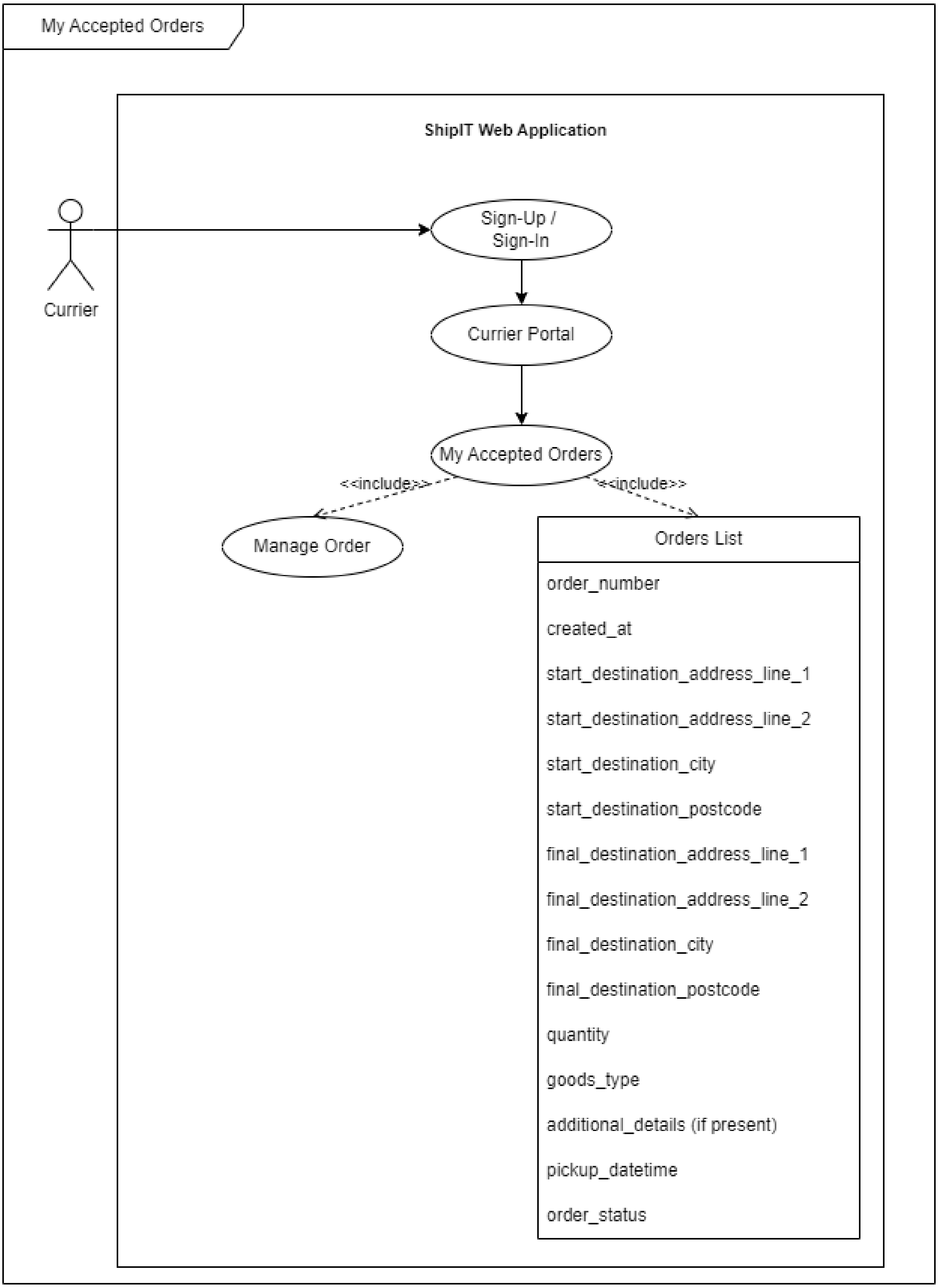


Figure 6 My Accepted Orders

2.1.1.22. Requirement 6 My Accepted Orders

2.1.1.23. Description & Priority

This use case describes how the currier navigates and makes use of “My Accepted Orders” page. It shows how the currier interacts with the ShipIT web application in order to view all Accepted orders and also manage these further.

2.1.1.24. Use Case

- Scope

The scope of this use case is to show the interaction between a currier user and the ShipIT web application so that the currier may view all their orders with order_status of accepted and in progress.

- Description

This use case describes how the currier gains access to viewing their accepted orders and how they navigate to managing them.

<u>Flow Description</u>

- Precondition

The user must have ShipIT web application open and be logged-in with an account that has role of “currier” and be able to access Currier Portal.

- Activation

The use case begins when the currier wants to view orders that are currently active for the currier.

- Main flow

1. The user navigates to the navigation bar and selects the currier portal.

2. The system redirects the currier to the currier portal.

3. The currier selects “My accepted orders”

4. The system displays the “Orders list” where all Accepted and In Progress orders are visible.

5. The user selects the “Manage Order” button to further manage the order.

- Alternate flow

N/A

- Exceptional flow

1. The user Is not logged in with an account that has the role of “currier” and thus can’t locate the currier portal.

2. The user navigates to the home page and checks their role.
3. If the role is not “currier” the user does not have permission to view the Currier portal.

- Termination

The use case terminates when the currier is happy enough with their order list and leaves the page.

- Post condition

The system goes into a wait state.

2.1.1.25. “Manage Order” Use Case Diagram

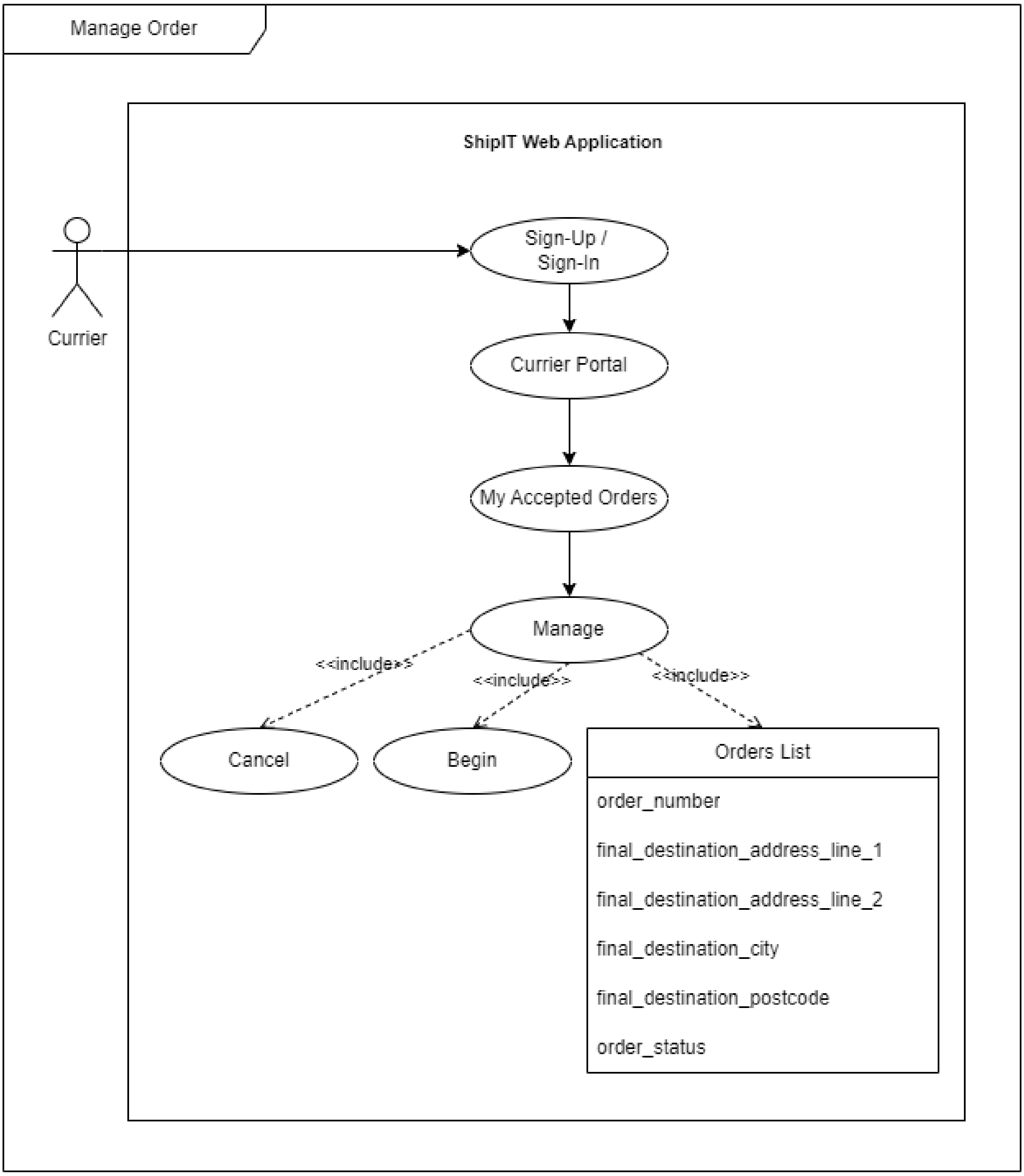


Figure 7 Manage Order

2.1.1.26. Requirement 7 Manage Order

2.1.1.27. Description & Priority

This use case goes through how the currier navigates and makes use of “Manage Order” function within a specific order and what options are provided for the currier.

2.1.1.28. Use Case

- Scope

The scope of this use case is to show how the currier makes use of the “Manage Order” function for a specific order.

- Description

The use case goes through the steps a currier must go through to manage an order.

Flow Description

- Precondition

The user must have the ShipIT web application open and be logged-in with an account that has role of “currier”.

- Activation

The use case begins when the currier wants to manage a specific order.

- Main flow

1. The currier navigates to the navigation bar and selects the “Currier Portal”.
2. The system redirects the currier to the “Currier Portal” page.
3. The currier selects to view “My accepted orders”.
4. The system redirects the user to the “Orders list” where all orders have order status of accepted or in progress.
5. The currier navigates to “Manage Order” button at the bottom of a specific order.
6. The system redirects the user to “Manage Order” page.
7. The currier has options to “Cancel” or “Begin” the specific order when they wish to.
8. The currier selects “Begin” to begin a specific order and the order status is changed to “In progress”.
9. The system redirects the user to “My accepted orders” where orders with order status of Accepted and In Progress are located.
10. The currier may now view that the order that they began is now “in Progress”.

- Alternate flow

1. The currier selects the “Cancel” option to cancel a specific order when they do not wish to complete this order.
2. The system redirects the user to “Currier history” page where the user can see that the order has a status of “Cancelled”.

- Exceptional flow

N/A

- Termination

The user Began or Cancelled the order and leaves the page they have been redirected to.

- Post condition

  The system goes int a wait state.

2.1.1.29. “Order history” Use Case Diagram

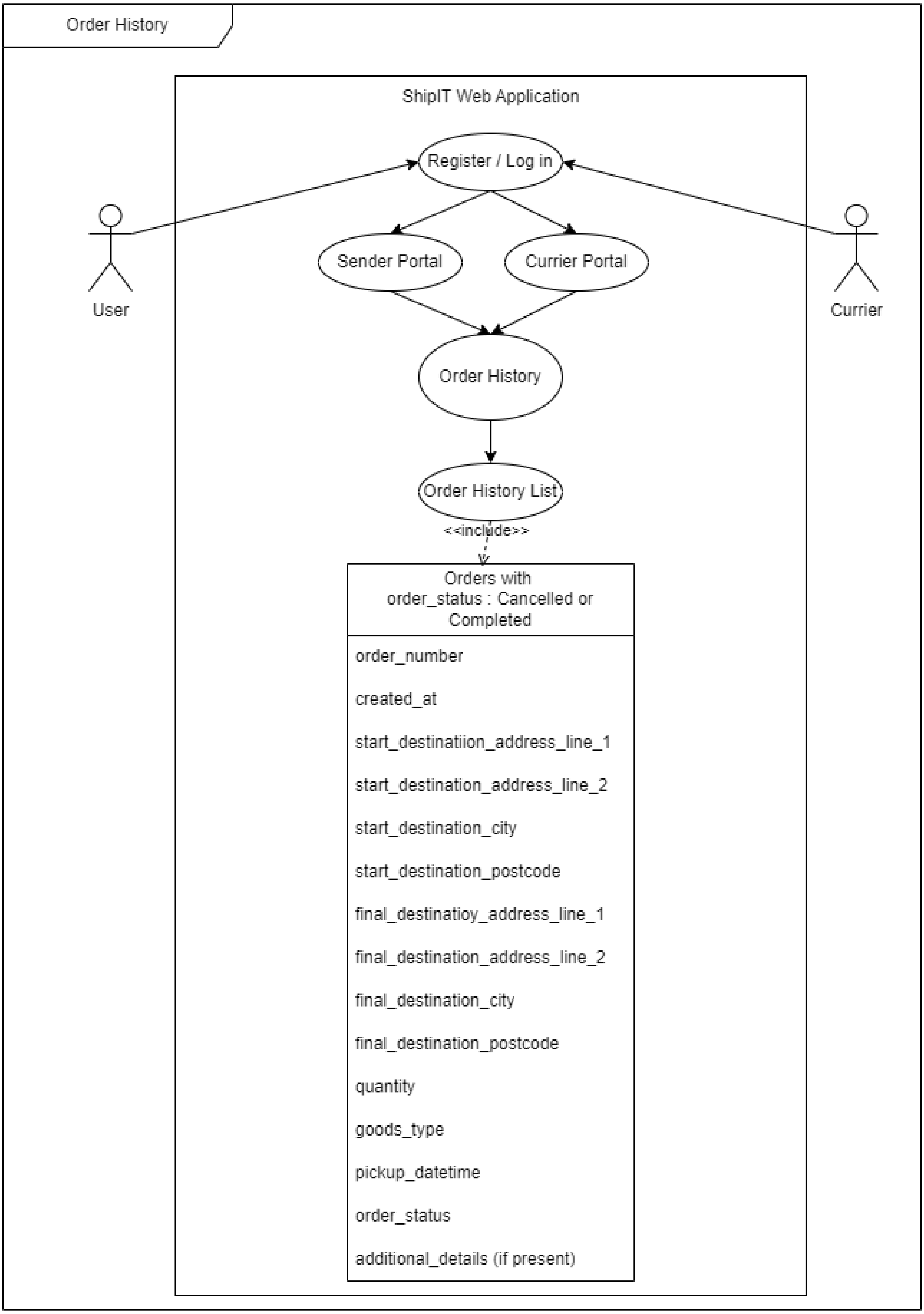


Figure 8 Order History

2.1.1.30. Requirement 7 Order History

2.1.1.31. Description & Priority

Order history comes at 7th place in priority as it displays orders that are cancelled or completed for both regular users and currier users of the ShipIT application. The makes it a useful tool to see what has happened to orders and to view orders after some time of completion.

2.1.1.32. Use Case

- Scope

- Description

  This use case goes through how regular and currier users alike go through the web application, navigate, and make use of "Order history".

  Flow Description

- Precondition

  The user has the ShipIT web application open and is logged in as either a user or currier.

- Activation

  The use case begins when the user wants to see the history of their previous orders.

- Main flow

  1. The user navigates to the navigation bar and selects "Sender portal".
  2. The system redirects the user to the Sender Portal.
  3. The user selects "Order history" to access the page.
  4. The system redirects the user to "Order History" page where a list is displayed of all orders that are tied to the user and have completed or cancelled order status.
  5. The user leaves the page.

- Alternate flow

  1. The user has the role of "currier".
  2. The currier navigates to the navigation bar and selects "Currier portal".
  3. The system redirects the user to the currier portal, the use case continues at main flow 3.

- Exceptional flow

  N/A

- Termination
  The user is satisfied with their order history and leaves the page.

- Post condition

  The system goes into a wait state.

2.1.1.33. “Receive items” Use Case Diagram

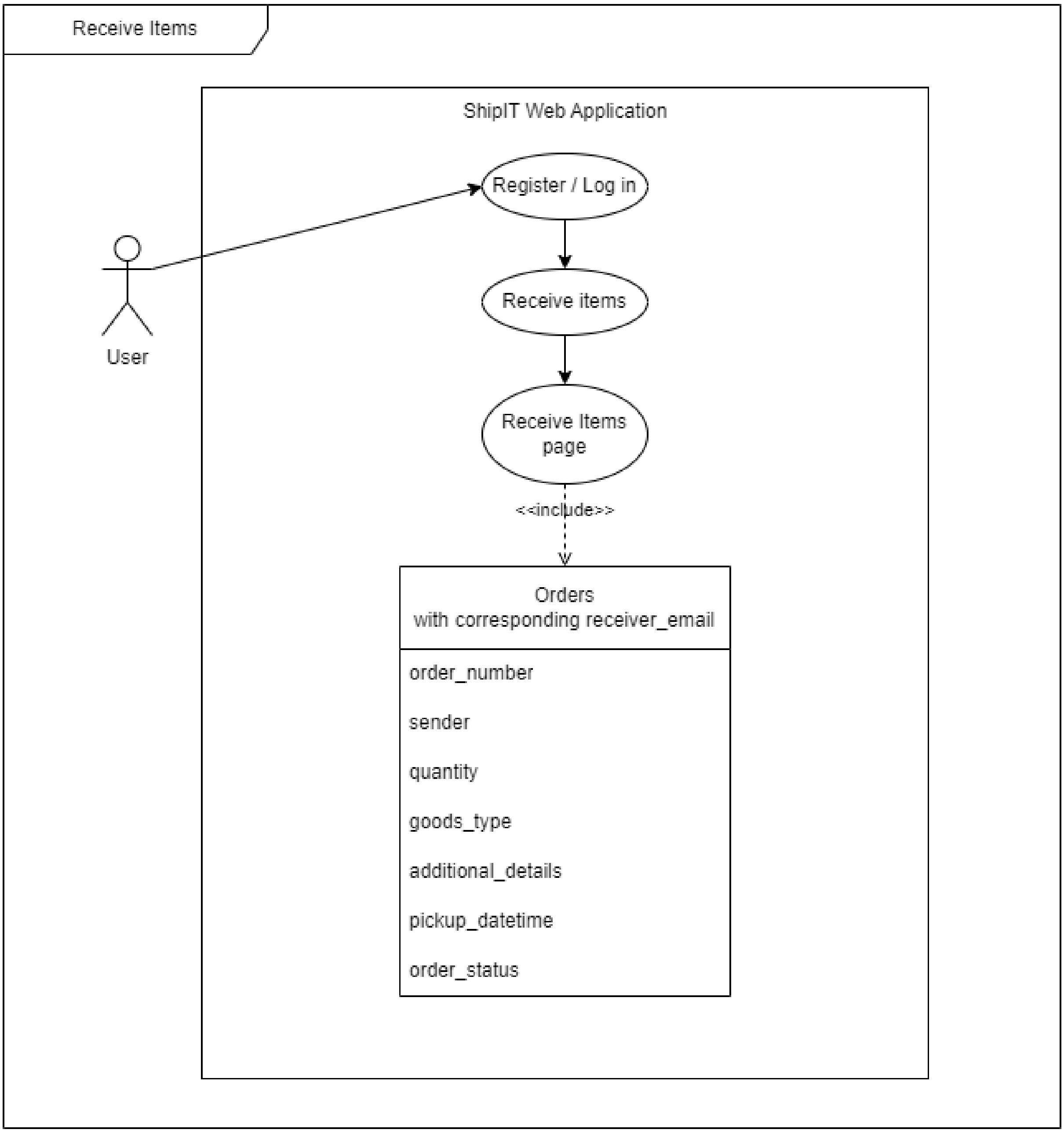


Figure 9 Receive Items

2.1.1.34. Requirement 8 Receive Items

2.1.1.35. Description & Priority

Receive items is an important part of ShipIT web application as it allows users to view orders that are being sent to them, they can see the sender, what time the order has been picked up, order status and more information about the order they are receiving.

2.1.1.36. Use Case

- Scope

The scope of this use case is to show how users interact with the “Receive items” part of the ShipIT web application.

- Description

This use case goes over how the user navigates to and makes use of the “Receive items” page.

Flow Description

- Precondition

The user must have the ShipIT web application open and be logged-in to their account.

- Activation

The use case starts when the user expects a delivery made to them and wishes to check for incoming orders.

- Main flow
  1. The user makes use of the navigation bar and selects the “Receive Items”.
  2. The system redirects the user to the “Receive items” page and displays a table showing all incoming orders to this user.
  3. The user makes use of the information in order to plan ahead and know when their orders are coming and what their status is.

- Alternate flow

- Exceptional flow

- Termination

The user is satisfied with the information provided by the “Receive Items” page and leaves the page.

- Post condition

The system goes into a wait state.

2.1.2. Data Requirements

To store the data and create the database of ShipIT web application, I made use of Ruby on Rails Active Record and migrations to add new and rollback unwanted changes.

The data requirements consist of the "Orders" table and "Users" table.

- "Orders" table data requirements:

  1. Order_id: integer – Created by default when an order is created in the database. Unique identifier within the database.
  2. Order_number: string – Unique identifier for each new order that is created using ShipIT web application. It makes finding a specific order easy and is generated upon creating a new order.
  3. Start_destination: string – Represents the sender input of a location where the goods are to be picked-up by the currier. The start destination includes Address Line 1, Address Line 2, City and Postcode.
  4. Final_destination: string – Represents the sender input for a location where the goods are to be dropped-off by the currier. Includes Address Line 1, Address Line 2, City and Postcode of the destination.
  5. Quantity: integer – Number of items that is being sent from start to final destination.
  6. Goods_type: string – Name of the item that is being sent.
  7. Additional_details: text – Field for any additional information that the currier or receiver might make use of when completing the order.
  8. User_id: integer – Connects the created order with the current user. Also connects the curriers with their accepted orders. Connects users and curriers with cancelled and completed orders that belong to them.
  9. Order_status: integer – Defines how the order is progressing. Order status is set to "Pending" by default (0). When accepted by a currier the status is set to "Accepted" (1) and when the currier begins the order it is changed to "In Progress" (3). If the order is completed the order status is set to "Completed" (4) and if it is cancelled, the order status is "Cancelled" (5).
  10. Weight: float – Used for the cost calculator functionality of the ShipIT web application. In kilograms.
  11. Distance: float – Used for the cost calculator functionality of the ShipIT web application. In kilometres.
  12. Created_at: datetime – Created by default with the orders table and marks the time when the order was created in the database.
  13. Updated_at: datetime – Created by default with the orders table and marks the time when the order was updated in the database.
  14. Pickup_datetime: datetime – User inputted date time for when the order is to be picked-up at the start destination.
  15. Receiver_email: string – User input for the receiver e-mail. Connects the created order with the receiver enabling them to see incoming orders.

- "Users" table data requirements:

1. Email: string – created by the users when they create a new account. A required field which without the user can’t access main functions of the ShipIT web application.
2. Encrypted password: string – created by the user when creating a new account. Used to log-in to existing accounts.

2.1.3. User Requirements

The ShipIT system must enable users to carry out several tasks to make use of ShipIT web application as intended.

1. Creating a new account – The users of ShipIT web application must be able to create a new account where they select their own e-mail and password.

2. Log-in – The user must be able to make use of their account by being able to log in with an existing account by providing its e-mail and password.

3. Manage account- The user must be able manage their account, including changing their password or cancelling their account by selecting “Manage my account”.

4. Sign-out – The user must be able to log out of their account by pressing the “Sign out” button.

5. Sender portal – The user must be able to access and view the “Sender portal” if they are logged-in with their account.

6. Creating a new order – Users must be able to access the new order form from the sender portal. The user must be able to fill out all the required fields of the form including start destination details, final destination details, Quantity, good type, receiver e-mail and pick-up date time.

7. Submit new order - The user must be able to submit the form using the “Submit” button, creating the new order.

8. My orders - The user must be able to access “My orders” page and view all orders that are created by them. The “My orders” page must display all important information for the user including order number and order status.

9. Order history - The user must be able to view orders that have order status of completed or cancelled by accessing “Order History” page within the Sender portal.

10. Receive items - The user must be able to access “Receive items” page and view all incoming orders. The receive items page must display order number, sender, quantity, good type, any additional details, pick-up date and time and order status.

11. The user must be able to make use of “Cost Calculator” page. The system must display a form for the users and enable the user to fill in weight, distance, and quantity. The user must be able to click the “Calculate cost” button and the system must output the calculated cost based on the inputted values.

12. Currier portal – The users with role of “currier” must be able to access the Currier portal.

13. Browse pending orders – Curriers must be able to access and browse pending orders and the system must display all orders with order status of pending.

14. My accepted orders – Curriers must be able to access “My accepted Orders” page and the system must display all orders that belong to the currier and have order status of “accepted” and “in progress”.

15. Manage order – curriers must be able to access “Manage order” page and be able to “Begin” if the order status is “accepted”, “Complete” if the order status is “in progress”, and “Cancel”. The system must respond to the currier’s selection and change the order status accordingly.

16. Currier order history – The currier must be able to view orders that have order status of completed or cancelled by accessing “Order History” page within the Currier portal.

2.1.4. Environmental Requirements

2.1.5. Usability Requirements

- GUI: The user interface is intuitive, simple, and easy to navigate for its users. The colour schemes are consistent across the different pages of the web application.
- Navigation: Pages of the ShipIT web application provide for usability through its easy to navigate and logical layouts. The user is quickly able to find what they are looking for by using the navigation bar at the top of the ShipIT web pages.
- Input validations: present input validations for registrations, sign-up, new order forms and cost calculator forms. The user’s input is validated, and the user is notified if information is missing or incorrect.
- Compatibility: The web application is compatible with different operating systems such ash Linux, macOS and Windows. It is also compatible with multiple web browsers such as Chrome, Opera and Firefox.
- Performance: The web application’s simplicity provides for minimal lag between the application and its users. The ShipIT web application is fast and responsive.

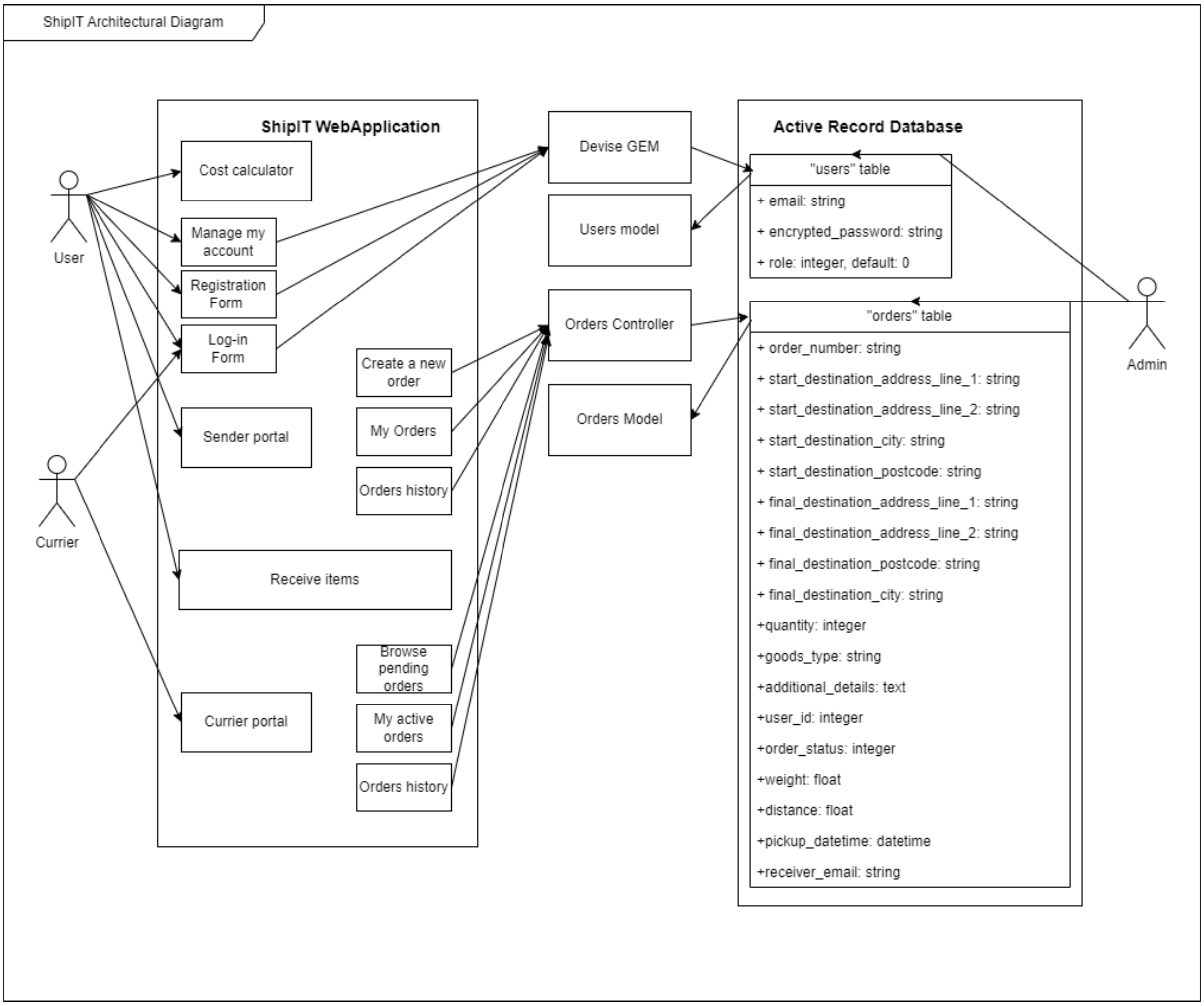
2.2. Design & Architecture

The Delivery Web Application will be comprised of the web application where the users will access features available for them to carry out their deliveries. The web application will initially enable users learn about ShipIT web application, to register or log in if they already have an account. Users will also be able to make use of the ShipIT cost calculator based on the costs set out by ShipIT to plan for costs.

Once the user has an account, they will be able to create new orders as well as keep track of existing, completed, and canceled orders.

The Delivery Web Application will have its system database and it will store user accounts, orders, and details of each. The senders, curriers and receivers would interact with the database through the web application.

Architecture diagram:



- The ShipIT Web Application provides users with functions through which they interact with the database.
- Devis GEM ensures password encryption and user account security.

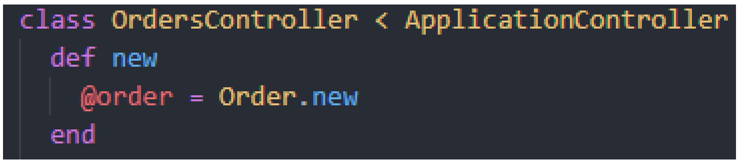
- The Users model sets out some of the rules of how the users table behaves, setting out default values for user roles, e-mail and password validations and user associations with the orders.
- Orders controller provides for important orders functionalities. It defines how orders are created, shown. It also defines what orders should be considered as “historic”. The order controller connects the currier with the order when the order is accepted, changes the order status based on the currier actions.
- The Orders model defines defaults of orders, enumerates order status and validates presence of a user that is assigned to the order.
- The active record database makes sure that all the user and orders information is stored in its assigned field. It helps connect users and orders table and enables users to manipulate and view orders.

2.3. Implementation

Controllers:

Orders controller: The most important class of the ShipIT web application as it accounts for the main functionalities and enabling users to work with orders. The orders controller also controls the behaviour of orders based on the actions of the users. Several different functions are defined withing the orders controller, including:

1. New

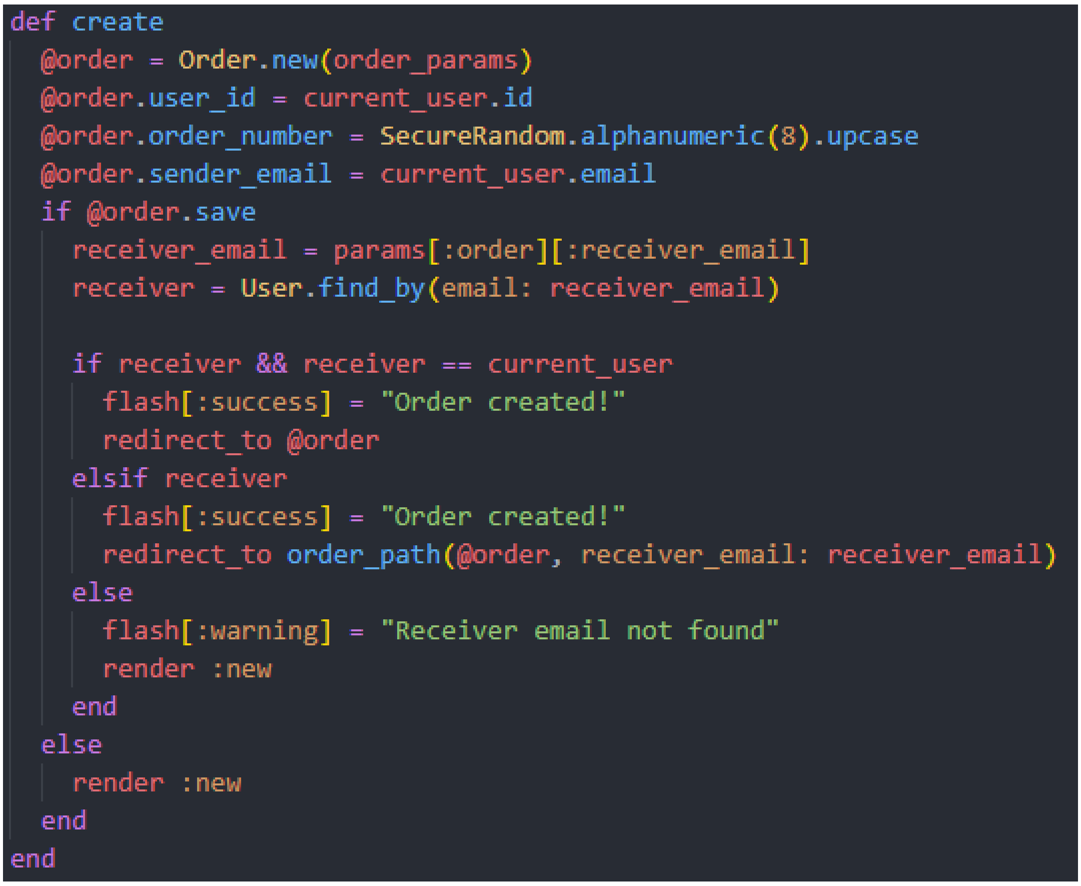


```
class OrdersController < ApplicationController
  def new
    @order = Order.new
  end
```

Figure 10 Orders controller - New Function

- The “New” function creates a new instance of the “Order” model and assigns it to the variable @order.

2. Create

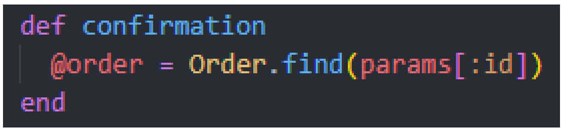


```ruby
def create
  @order = Order.new(order_params)
  @order.user_id = current_user.id
  @order.order_number = SecureRandom.alphanumeric(8).upcase
  @order.sender_email = current_user.email
  if @order.save
    receiver_email = params[:order][:receiver_email]
    receiver = User.find_by(email: receiver_email)

    if receiver && receiver == current_user
      flash[:success] = "Order created!"
      redirect_to @order
    elsif receiver
      flash[:success] = "Order created!"
      redirect_to order_path(@order, receiver_email: receiver_email)
    else
      flash[:warning] = "Receiver email not found"
      render :new
    end
  else
    render :new
  end
end
```

Figure 11 Orders controller - Create Function

- The create function is used to create a new instance of "Order" model with parameters passed from the form filled out by the user.
- The newly created orders are linked to the current user upon creation of the order. The create function also generates a random order number, assigning it to a specific order and acting as a unique identifier for the order.
- The create function makes sure that the receiver email is existing and is present and notifies the user if that is not the case.
- If the receiver email is not found the function renders the "new" action again.
- When a user saves the order, they are redirected to the confirmation page.

3. Confirmation

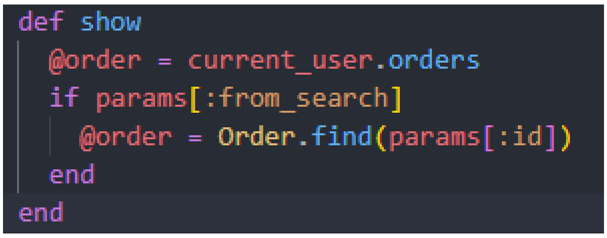


```ruby
def confirmation
  @order = Order.find(params[:id])
end
```

Figure 12 Orders controller - Confimration Function

- Used to show the confirmation page after an order has been successfully created and finds the order by passing the order id in the URL.

4. Show

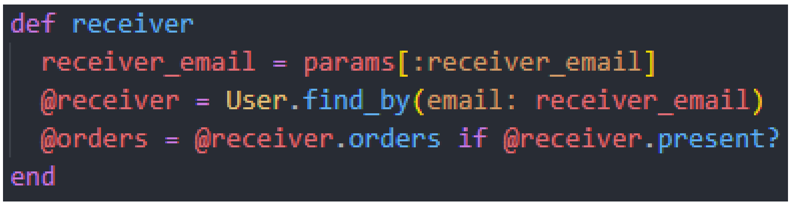


```ruby
def show
  @order = current_user.orders
  if params[:from_search]
    @order = Order.find(params[:id])
  end
end
```

Figure 13 Orders controller - Show Function

- The show function is used to display all orders from the current user. If the parameter ‘from_search’ is present it finds and displays a specific order.

5. Receiver

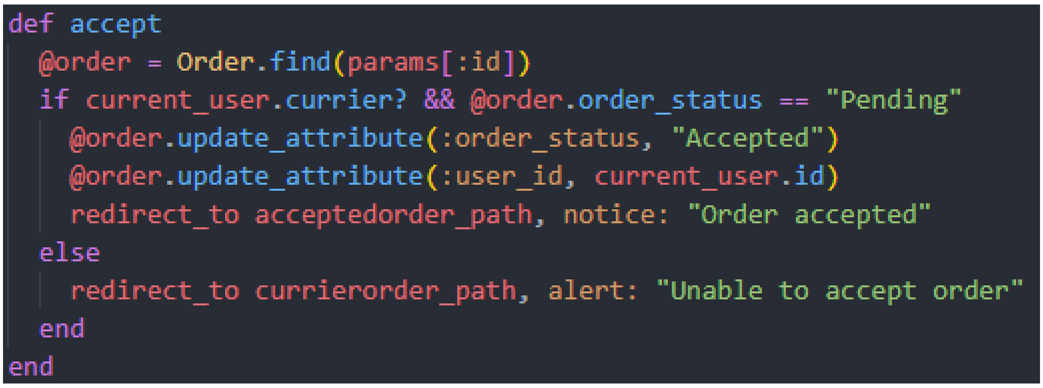


```ruby
def receiver
  receiver_email = params[:receiver_email]
  @receiver = User.find_by(email: receiver_email)
  @orders = @receiver.orders if @receiver.present?
end
```

Figure 14 Orders controller - Receiver Function

- The function is used to display all orders that have been sent to a specific ‘receiver_email’. It finds the user by the email address passed and shows all orders associated with the specific user.

6. Accept

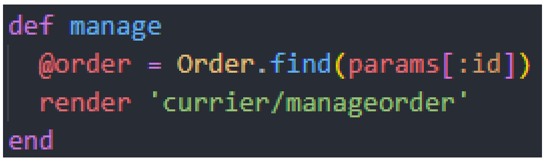


```ruby
def accept
  @order = Order.find(params[:id])
  if current_user.currier? && @order.order_status == "Pending"
    @order.update_attribute(:order_status, "Accepted")
    @order.update_attribute(:user_id, current_user.id)
    redirect_to acceptedorder_path, notice: "Order accepted"
  else
    redirect_to currierorder_path, alert: "Unable to accept order"
  end
end
```

Figure 15 Orders controller - Accept Function

- The accept function is used to update the order status to ‘Accepted’ if a currier accepts an order that is currently ‘Pending’. It also assigns the accepted order to the currier and notifies them that the order has been accepted.
- In case the currier is unable to accept the order, the system notifies the currier.

7. Manage

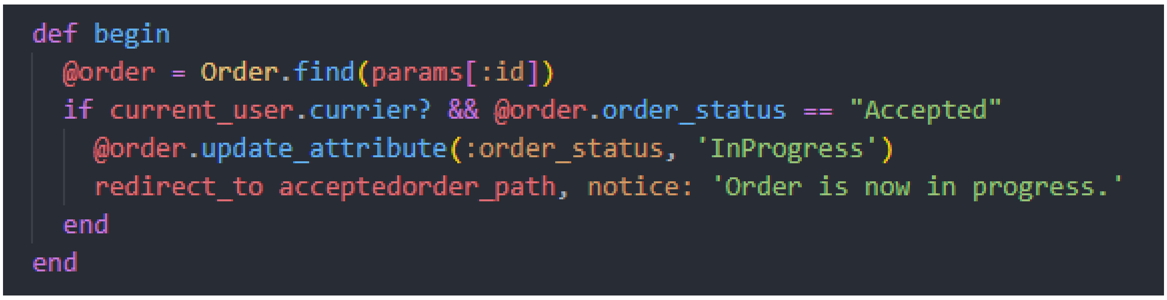


```ruby
def manage
  @order = Order.find(params[:id])
  render 'currier/manageorder'
end
```

Figure 16 Orders controller - Manage Function

- Used to redirect the currier to the “Manage order” page and finds the order by the ID passed in the URL.

8. Begin

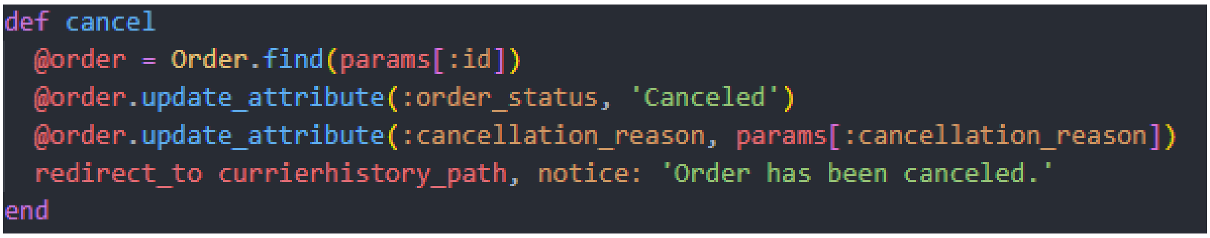


```ruby
def begin
  @order = Order.find(params[:id])
  if current_user.currier? && @order.order_status == "Accepted"
    @order.update_attribute(:order_status, 'InProgress')
    redirect_to acceptedorder_path, notice: 'Order is now in progress.'
  end
end
```

Figure 17 Orders controller - Begin Function

- The begin function is used to change the order status to ‘InProgress’ when the curriers begin the order.
- The order must be accepted, and the current user must be a currier.
- When the currier begins the order, the system notifies the curriers that the ‘Order is now in progress’.

9. Cancel

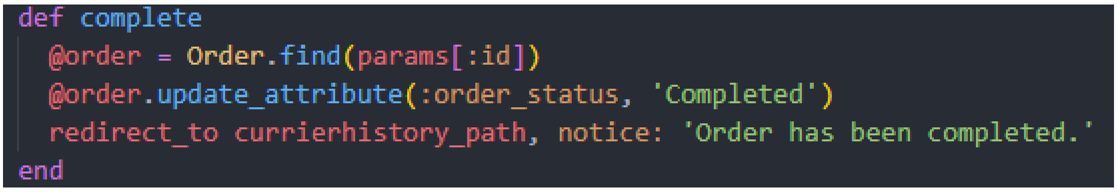


```ruby
def cancel
  @order = Order.find(params[:id])
  @order.update_attribute(:order_status, 'Canceled')
  @order.update_attribute(:cancellation_reason, params[:cancellation_reason])
  redirect_to currierhistory_path, notice: 'Order has been canceled.'
end
```

Figure 18 Orders controller - Cancel Function

- The cancel function is used to change the order status attribute to ‘Canceled’ when the order is being cancelled and updates the cancellation reason attribute.
- It finds the order by the ID passed in the URL.
- The system redirects the user to “Currier history” page and notifies the user that the ‘Order has been canceled’.

10. Complete

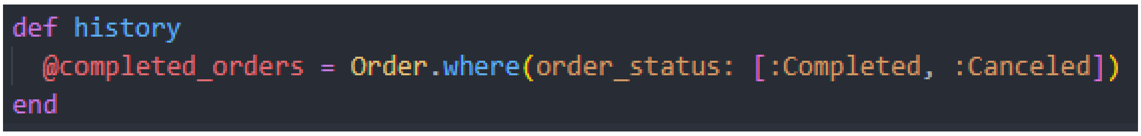


```ruby
def complete
  @order = Order.find(params[:id])
  @order.update_attribute(:order_status, 'Completed')
  redirect_to currierhistory_path, notice: 'Order has been completed.'
end
```

Figure 19 Orders controller - Complete Function

- Similarly to “Cancel” function, the “Complete” function is used to update the order status attribute to ‘Completed’ when the order is being completed.
- The user is redirected to ‘Currier history’ page and notified that the ‘Order has been completed’.

11. History

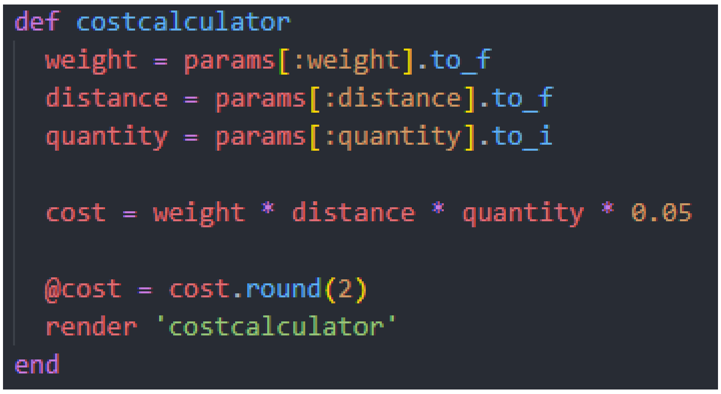


```
def history
  @completed_orders = Order.where(order_status: [:Completed, :Canceled])
end
```

Figure 20 Orders controller - History Function

- Function used to display all orders where the order status attribute is 'Completed' or 'Canceled'.

12. Costcalculator



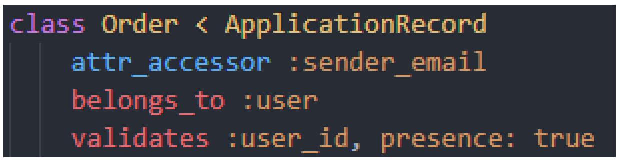
```
def costcalculator
  weight = params[:weight].to_f
  distance = params[:distance].to_f
  quantity = params[:quantity].to_i

  cost = weight * distance * quantity * 0.05

  @cost = cost.round(2)
  render 'costcalculator'
end
```

Figure 21 Orders controller - Costcalculator Function

- Costcalculator function calculates the cost of the order based on the weight, distance, quantity of goods. It calculates and renders the cost calculator page with the cost output.

Models:

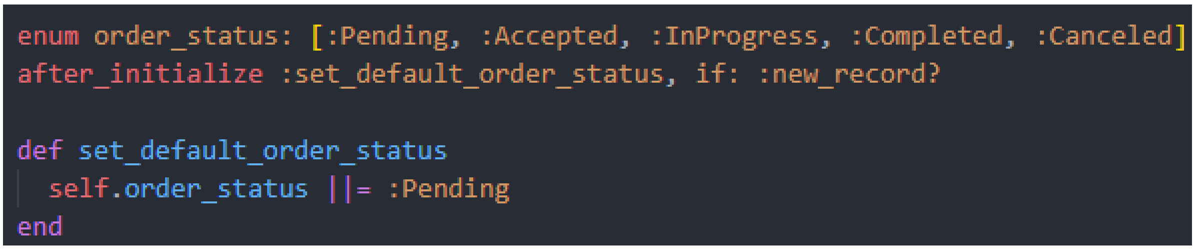
Model "Order.rb": This model represents the order object in the ShipIT web application and is associated through the 'belongs_to' association. It also validates that each order object has an user_id associated with it before a new order object can be saved to the database.



```
class Order < ApplicationRecord
    attr_accessor :sender_email
    belongs_to :user
    validates :user_id, presence: true
```

Figure 22 Orders model - associations

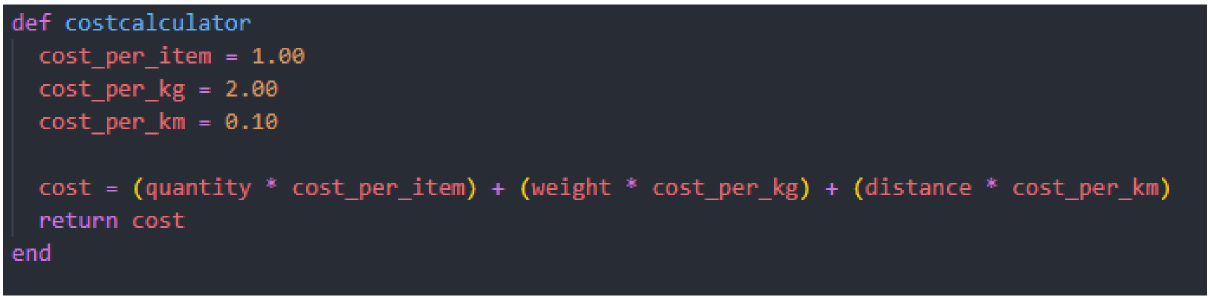
The 'Order.rb' model also declares and enumerates 'order_status' attribute of the orders where 0 = Pending, 1 = Accepted, 2 = InProgress, 3 = Completed and 4 = Canceled.
A default value 0 = Pending is set when a new order is initialized.



```
enum order_status: [:Pending, :Accepted, :InProgress, :Completed, :Canceled]
after_initialize :set_default_order_status, if: :new_record?

def set_default_order_status
  self.order_status ||= :Pending
end
```

Figure 23 Orders model - Enumeration and Defaults

In the ‘Order.rb’ model the cost calculator and the calculation logic is also defined.

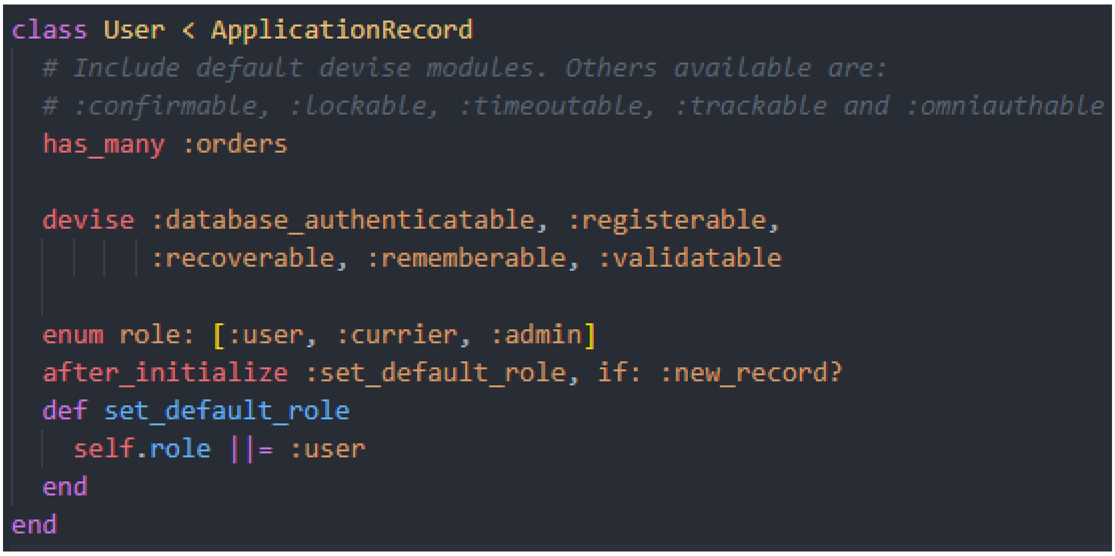


```
def costcalculator
  cost_per_item = 1.00
  cost_per_kg = 2.00
  cost_per_km = 0.10

  cost = (quantity * cost_per_item) + (weight * cost_per_kg) + (distance * cost_per_km)
  return cost
end
```

Figure 24 Orders model - Costcalculator

The cost is returned upon calculating the cost of an order.

<u>Model “User.rb”:</u>



```
class User < ApplicationRecord
  # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable, :trackable and :omniauthable
  has_many :orders

  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :validatable

  enum role: [:user, :currier, :admin]
  after_initialize :set_default_role, if: :new_record?
  def set_default_role
    self.role ||= :user
  end
end
```

Figure 25 User model - Associations, Defaults, Enumeration

The ‘User.rb’ model has one-to-many association with the Order model through the has_many :orders method.

This model uses devise for authentication and authorization of users.

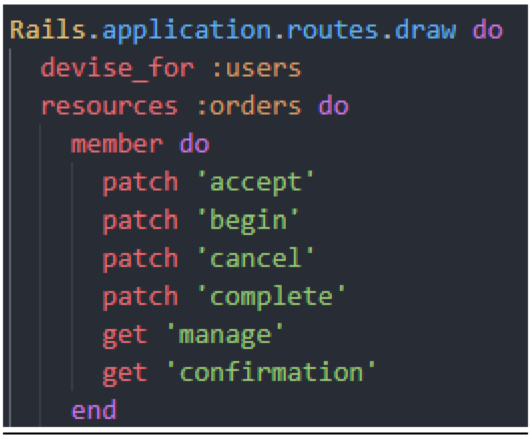
It also enumerates and initialized roles for the user of the ShipIT web application, where 0 = user, 1 = currier and 2 = admin.

A default role of 0 = user is set when a new user account is created.

Routes.rb:

Devise_for: users – sets up routes for the devise gem, providing authorization and authentication for the user accounts of the ShipIT web application.

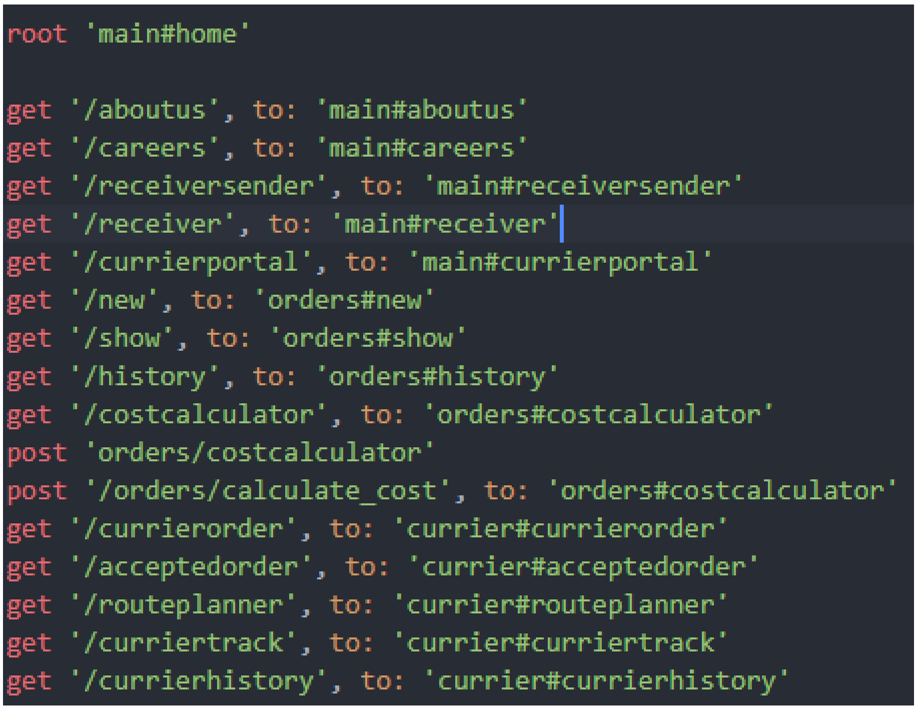
Member do – sets up routes and defines actions for a single 'order' resource, identifies by its ID.
Patch 'accept' – updates the order status to "Accepted".
Patch 'begin' – updates the order status to "InProgress".
Patch 'cancel' – updates the order status to "Canceled".
Patch 'complete' – updates the orders status to 'Completed".
Get 'manage' – shows the view form managing a specific order.
Get 'confirmation' – shows a confirmation page for the created order.



```
Rails.application.routes.draw do
  devise_for :users
  resources :orders do
    member do
      patch 'accept'
      patch 'begin'
      patch 'cancel'
      patch 'complete'
      get 'manage'
      get 'confirmation'
    end
```

Figure 26 Single order - Routes.rb

Root 'main#home' sets the root URL to 'home' action in the 'main' controller. The remaining routes define custom URLs for various actions of the ShipIT web application.



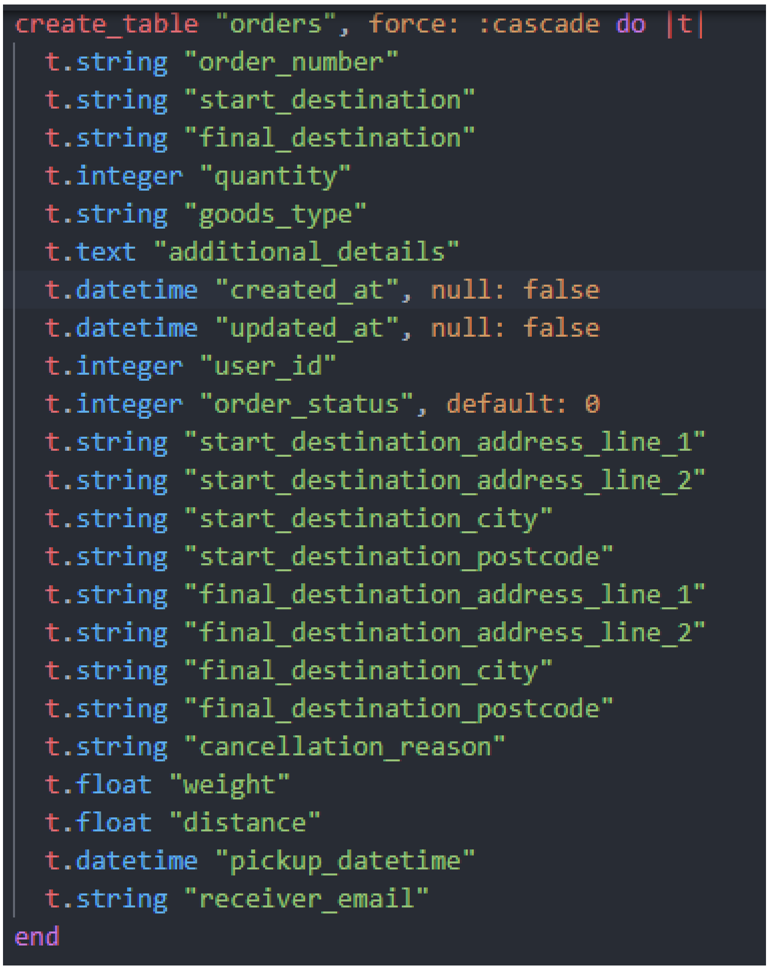
```ruby
root 'main#home'

get '/aboutus', to: 'main#aboutus'
get '/careers', to: 'main#careers'
get '/receiversender', to: 'main#receiversender'
get '/receiver', to: 'main#receiver'
get '/currierportal', to: 'main#currierportal'
get '/new', to: 'orders#new'
get '/show', to: 'orders#show'
get '/history', to: 'orders#history'
get '/costcalculator', to: 'orders#costcalculator'
post 'orders/costcalculator'
post '/orders/calculate_cost', to: 'orders#costcalculator'
get '/currierorder', to: 'currier#currierorder'
get '/acceptedorder', to: 'currier#acceptedorder'
get '/routeplanner', to: 'currier#routeplanner'
get '/curriertrack', to: 'currier#curriertrack'
get '/currierhistory', to: 'currier#currierhistory'
```

Figure 27 Root & other Routes.rb

<u>Database- Schema.rb:</u>

- The schema.rb consists of “Orders” and “Users tables. The “Orders” table is focused on storing various information of the orders that are created and used by ShipIT web application.

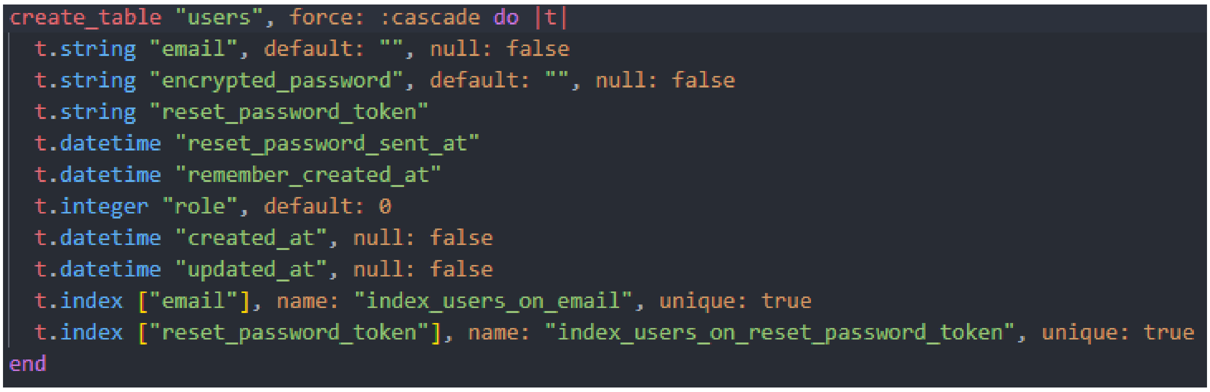
- The important part of the orders table is user_id integer, which effectively connects the users with the orders enabling associations between the two.



```ruby
create_table "orders", force: :cascade do |t|
  t.string "order_number"
  t.string "start_destination"
  t.string "final_destination"
  t.integer "quantity"
  t.string "goods_type"
  t.text "additional_details"
  t.datetime "created_at", null: false
  t.datetime "updated_at", null: false
  t.integer "user_id"
  t.integer "order_status", default: 0
  t.string "start_destination_address_line_1"
  t.string "start_destination_address_line_2"
  t.string "start_destination_city"
  t.string "start_destination_postcode"
  t.string "final_destination_address_line_1"
  t.string "final_destination_address_line_2"
  t.string "final_destination_city"
  t.string "final_destination_postcode"
  t.string "cancellation_reason"
  t.float "weight"
  t.float "distance"
  t.datetime "pickup_datetime"
  t.string "receiver_email"
end
```

Figure 28 Orders table - schema.rb

- The “Users” table stores information about the users such as their e-mail and encrypted passwords as well as roles.

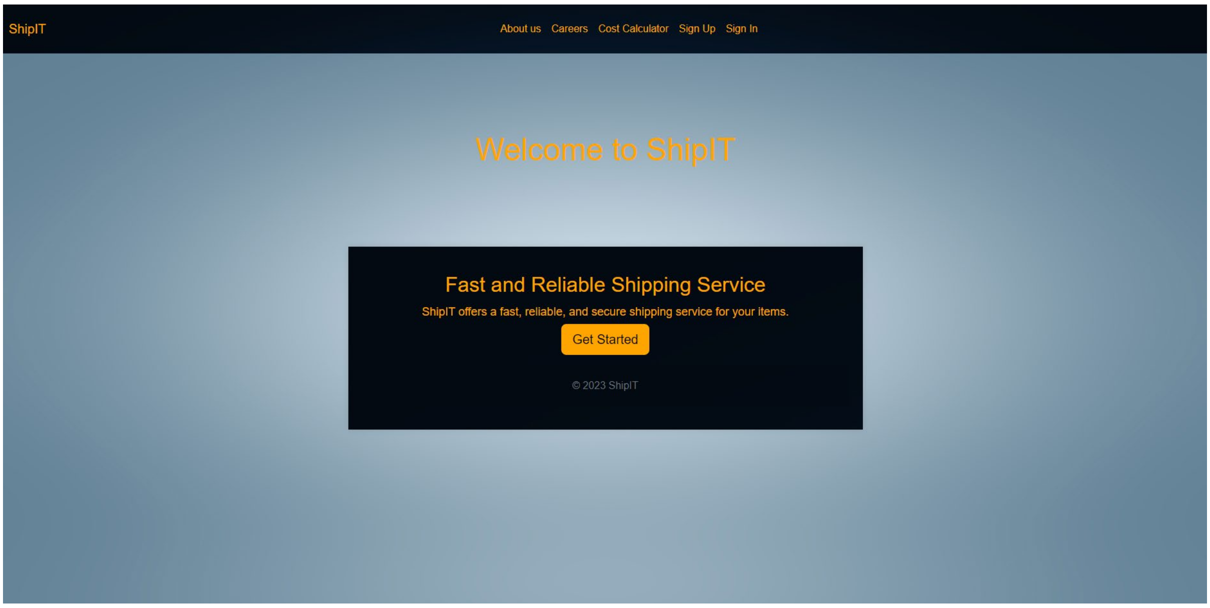


```ruby
create_table "users", force: :cascade do |t|
  t.string "email", default: "", null: false
  t.string "encrypted_password", default: "", null: false
  t.string "reset_password_token"
  t.datetime "reset_password_sent_at"
  t.datetime "remember_created_at"
  t.integer "role", default: 0
  t.datetime "created_at", null: false
  t.datetime "updated_at", null: false
  t.index ["email"], name: "index_users_on_email", unique: true
  t.index ["reset_password_token"], name: "index_users_on_reset_password_token", unique: true
end
```

Figure 29 Users table - schema.rb

2.4. Graphical User Interface (GUI)

Navigation bar and home page:



ShipIT

About us Careers Cost Calculator Sign Up Sign In

Welcome to ShipIT

Fast and Reliable Shipping Service

ShipIT offers a fast, reliable, and secure shipping service for your items.

Get Started

© 2023 ShipIT

Figure 30 Home page & Navbar

Navigation bar provides various links for the user:

Access to the home page by click in “ShipIT”,

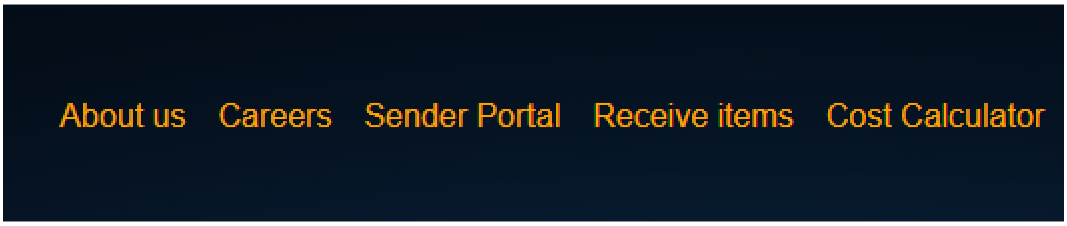
learn about the ShipIT by clicking “About us”,

learn about possibilities of working with the company by clicking “Careers”,

access the cost calculator via the “Cost Calculator” link.

To access further functionality, the user may “Sign Up” to create a new account or “Sign in” to access the application with an existing account.

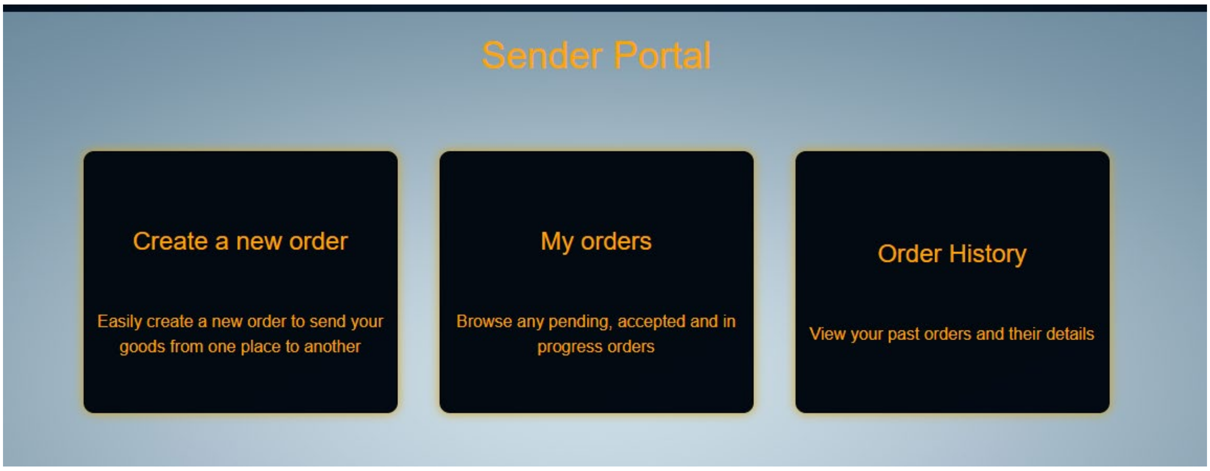
When the user is signed-in to the application, they have new links on the navigation bar:



About us Careers Sender Portal Receive items Cost Calculator

Figure 31 Navbar

Sender portal link which will redirect the user to a new page, enabling them to “Create a new order”, view “My orders”, or view their “Order History”.



Sender Portal

Create a new order

Easily create a new order to send your goods from one place to another

My orders

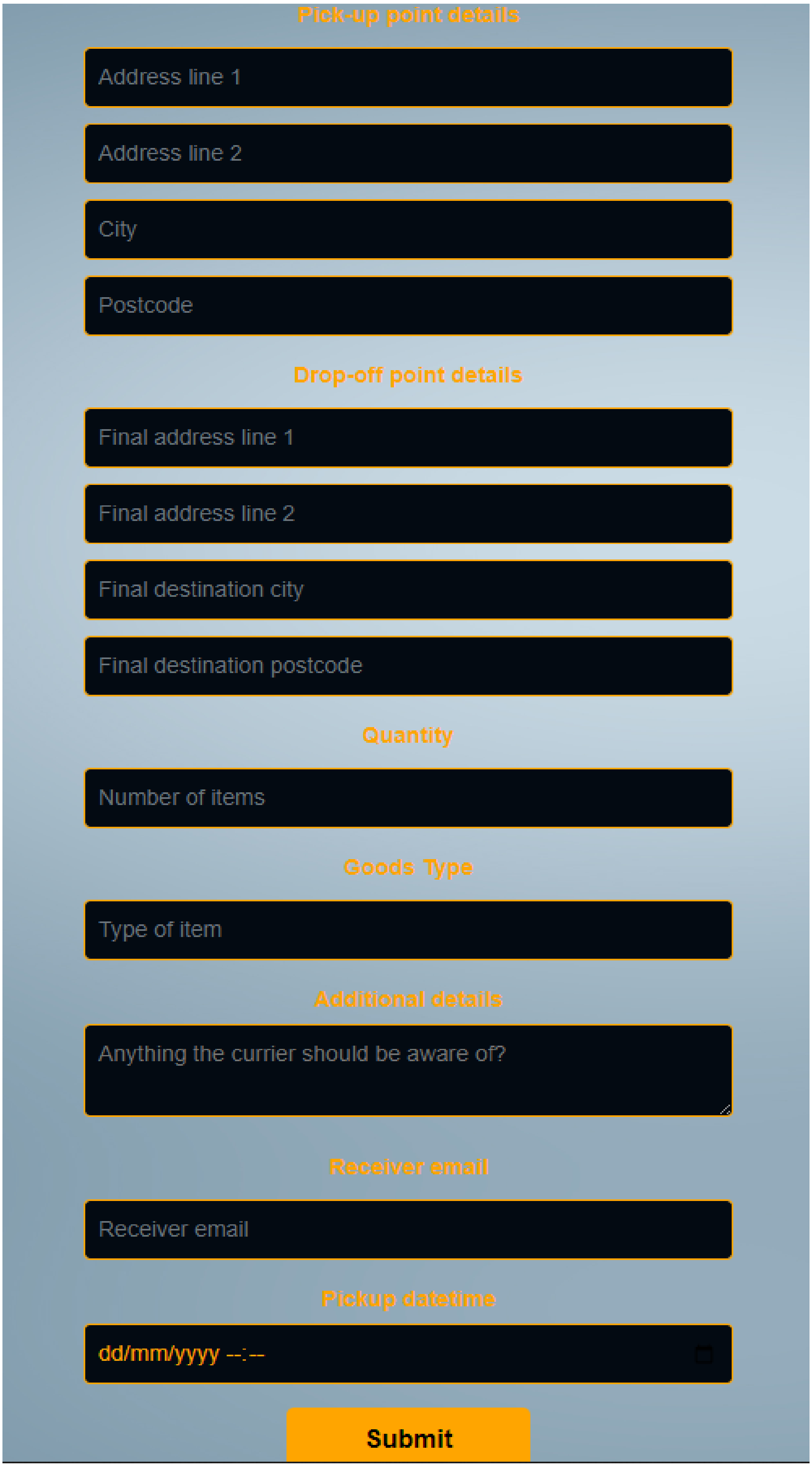
Browse any pending, accepted and in progress orders

Order History

View your past orders and their details

Figure 32 Sender portal

By selecting “Create a new order” the user will be presented with a from which they need to fill out to create a new order:



Pick-up point details

Address line 1

Address line 2

City

Postcode

Drop-off point details

Final address line 1

Final address line 2

Final destination city

Final destination postcode

Quantity

Number of items

Goods Type

Type of item

Additional details

Anything the currier should be aware of?

Receiver email

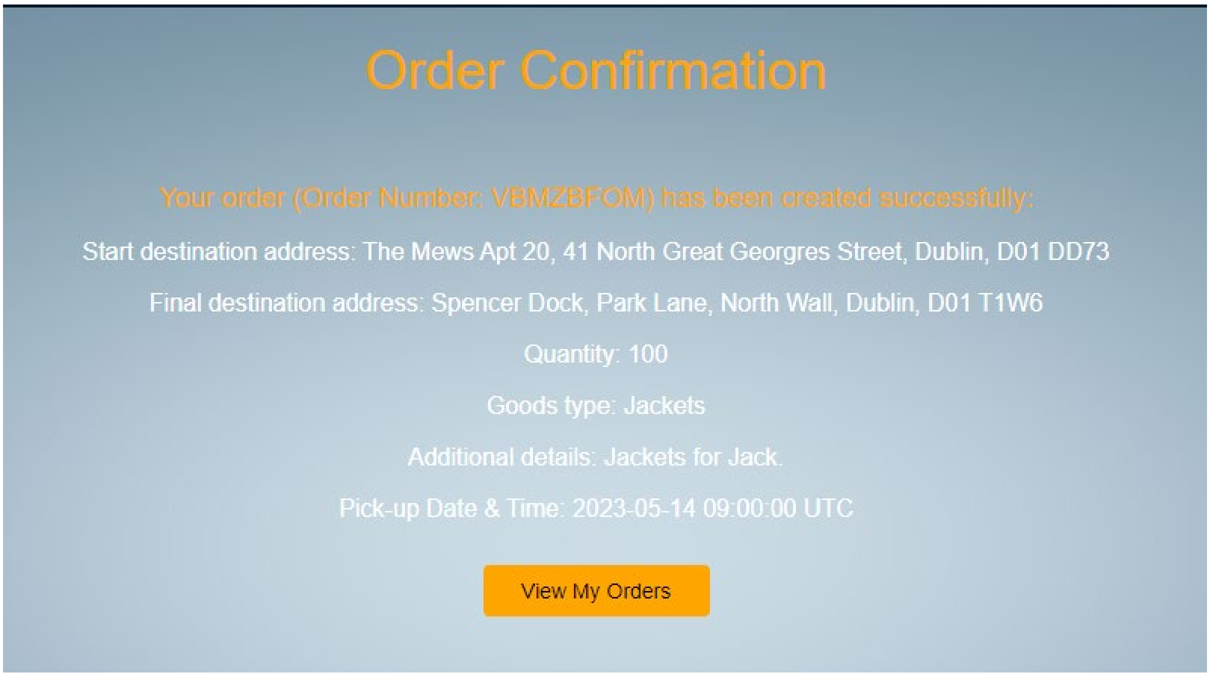
Receiver email

Pickup datetime

dd/mm/yyyy --:--

Figure 33 New order form

When the user fills out the form and clicks the “Submit” button, they are redirected to an “Confirmation” page where they are shown the following information:



Order Confirmation

Your order (Order Number: VBMZBFOM) has been created successfully:

Start destination address: The Mews Apt 20, 41 North Great Georgres Street, Dublin, D01 DD73

Final destination address: Spencer Dock, Park Lane, North Wall, Dublin, D01 T1W6

Quantity: 100

Goods type: Jackets

Additional details: Jackets for Jack.

Pick-up Date & Time: 2023-05-14 09:00:00 UTC

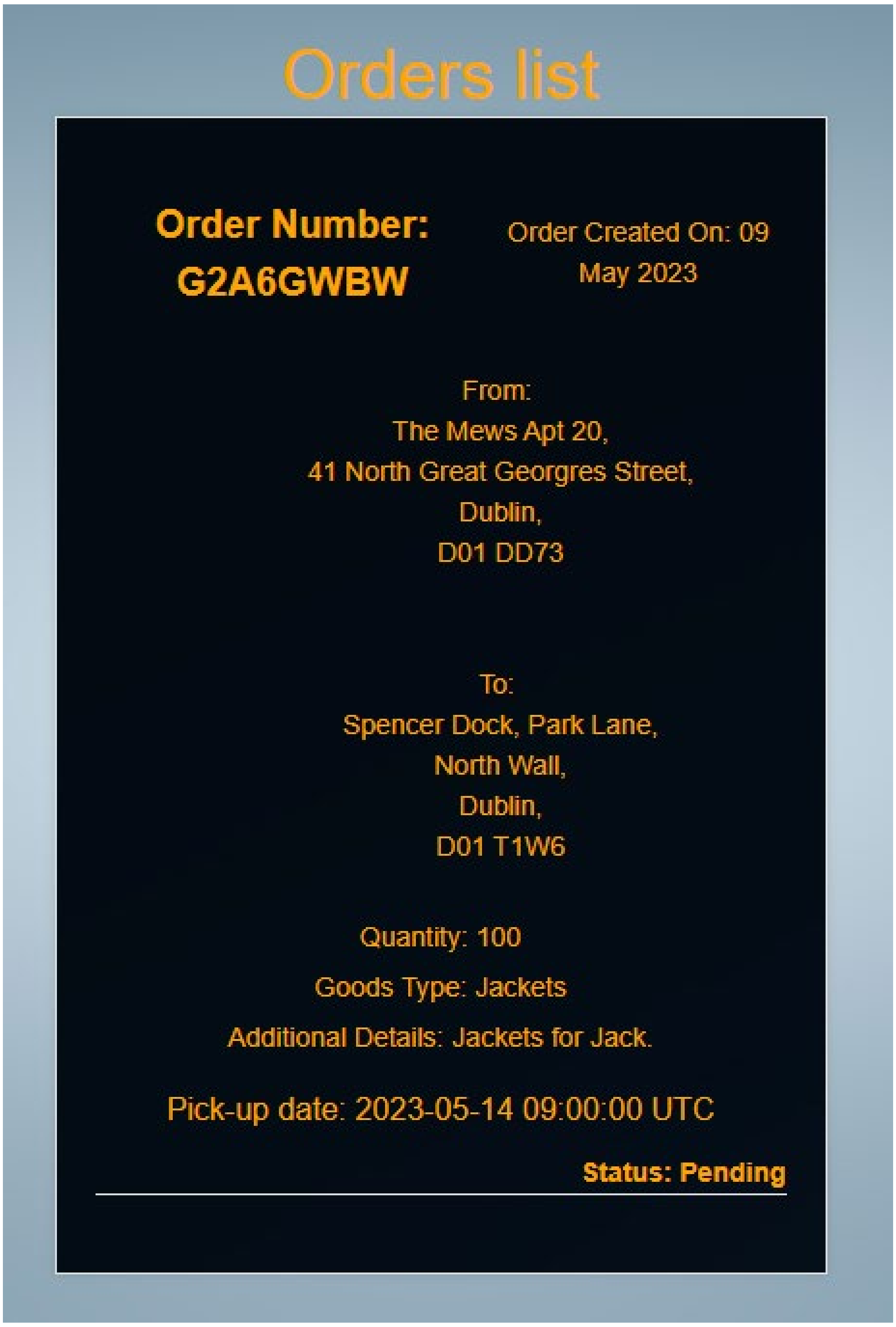
Figure 34 Order confirmation

This includes the newly generated order number which the user may use to navigate the order.

This page also includes the “View My Orders” link which will redirect the user to the page where all of their “Pending”, “Active” and “In progress” orders are located.

The user may also access this page via the “Sender portal” by selecting the “My orders” link.

Once the user is on the “My orders” page, they will see information about their order, including order number, date, and time the order has been created and the order status.



Orders list

Order Number: G2A6GWBW

Order Created On: 09 May 2023

From:
The Mews Apt 20,
41 North Great Georgres Street,
Dublin,
D01 DD73

To:
Spencer Dock, Park Lane,
North Wall,
Dublin,
D01 T1W6

Quantity: 100

Goods Type: Jackets

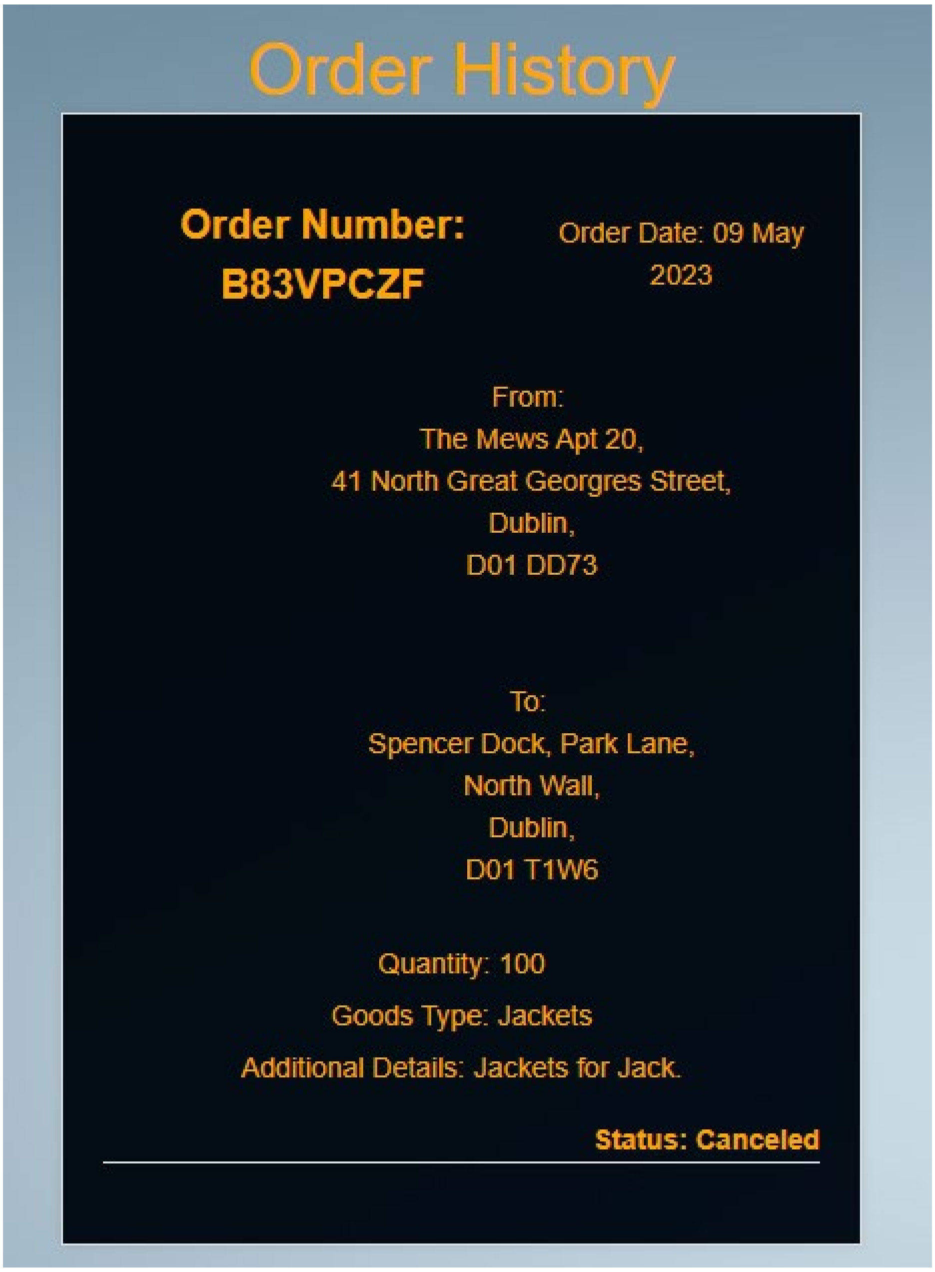
Additional Details: Jackets for Jack.

Pick-up date: 2023-05-14 09:00:00 UTC

Status: Pending

Figure 35 View "My orders"

In case the user wants to view orders that have been completed or cancelled, the user may navigate to “Orders history” via the Sender portal. In the “Orders history” page all orders that have the order status of “Canceled” or “Completed” will be displayed.



Order History

Order Number: B83VPCZF

Order Date: 09 May 2023

From:
The Mews Apt 20,
41 North Great Georgres Street,
Dublin,
D01 DD73

To:
Spencer Dock, Park Lane,
North Wall,
Dublin,
D01 T1W6

Quantity: 100

Goods Type: Jackets

Additional Details: Jackets for Jack.

Status: Canceled

Figure 36 View "Orders history"

2.5. Testing

Various tests are carried out on the Delivery Web Application to verify if the system requirements have been met and are working as initially planned and they are:

Unit testing - each feature and its requirements will be tested as a part of an independent unit to test its functionality.

1. Ruby On Rails testing – Models – I have created couple of tests to test the user and orders model and if they are functioning as expected.

2. The tests include setting the default order status to “Pending” ensuring that the model is behaving as expected.
3. The cost calculator is also tested in this testing instance, as the correctness of the calculation is being evaluated.
4. User model is also being tested, some of the testing includes making sure that users are not being saved without an email. New user account also must have valid e-mail and passwords before being saved onto the database and default roles should be set as intended in the user model.

Unit test plan:

Completing tests in Ruby on Rails is a simple process:

1. Use the command console in the IDE of your choice.
2. Navigate to the directory where the ShipIT web application is stored.
3. Once in the directory, run the following commands:

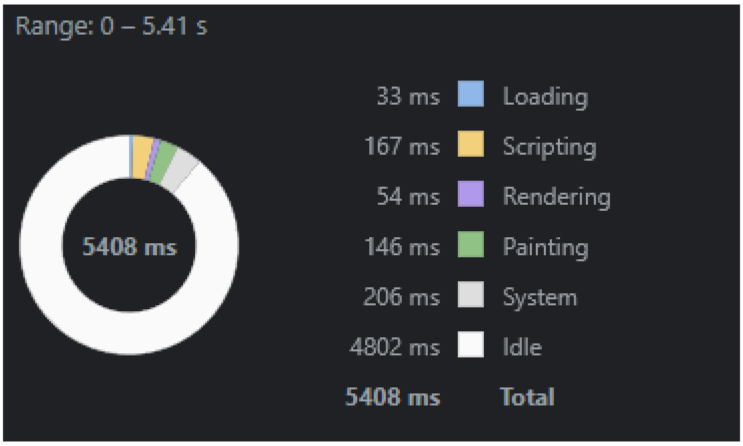
```
Rails test test/models/order_test.rb
Rails test test/models/user_test.rb
```

<u>Integration testing – test multiple features working together to evaluate if they are working as expected.</u>

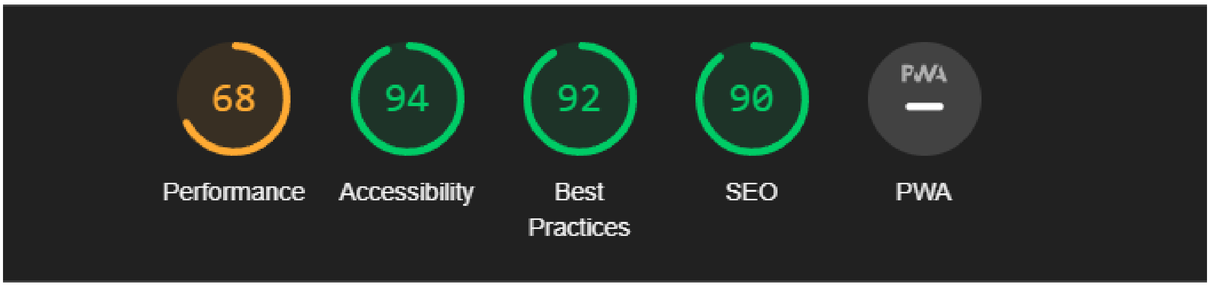
1. Cost Calculator Integration Testing – tests interaction between the controller and the view. The test posts a request to the controller and checks the response expecting HTML content, including the result of the cost calculation.
2. Creating a new order – this integration test evaluates the response of the views to creating a new order. When a new order is created the user should be redirected to a confirmation page.

- End-to-end testing will be carried out once all requirements are completed and will be used to check if the application is working as initially set out.
- Willing human participants over the age of 18 are going to be invited to participate in testing. They will be invited via their e-mail and will have to provide consent before participating in the testing of the Delivery Web Application. Once they feel they have a good understanding of the application they will be provided with a yes/no questionnaire and will include a text area for extra feedback. These questionnaires will be reviewed and will be taken into consideration while testing the Delivery Web Application.

2.6. Evaluation



```
Range: 0 – 5.41 s

             33 ms  Loading
            167 ms  Scripting
             54 ms  Rendering
5408 ms     146 ms  Painting
            206 ms  System
           4802 ms  Idle
           5408 ms  Total
```



I have used an open-source tool Lighthouse and Chrome Dev Tools to evaluate the performance of ShipIT Web Application, giving me insight into the performance, accessibility, practices, and semantics and search engine optimization. These ratings helped me understand what I could change and what parts to improve during further development of the ShipIT web application.

The 68 score on performance suggests that there is still room for improvement of the areas such as loading times and speed of the ShipIT Web application.

Accessibility is scored well as the pages are simple and there is not too much content for the user to take on one page.

Best practices and SEO score well as well, which when evaluated means that the ShipIT Web application currently follows good web development practices with solid HTML structure. This also helps with making the web application reliable and easy to maintain. The SEO on the other hand evaluates the page titles, meta descriptions, and absences of broken links. From the score of 90 it is evaluated that ShipIT follows good practices in terms of SEO optimization.

3.0 Conclusions

Advantages and strengths:

- The ShipIT web application is straightforward, simple to understand and navigate.
- It is compatible with multiple web browsers and operating systems and provides good performance for its users.
- User accounts are secured with encrypted passwords.
- Provides uniqueness with the cost calculator, enabling users to plan for costs.

Disadvantages and limitations:

- Does not provide users to select multiple stops while creating an order.
- Does not automatically calculate cost of the order, while the order is being created.
- Does not allow for map view when selecting start and final destinations.

4.0 Further Development or Research

With further development I would make improvements the GUI making the web application more responsive and enjoyable for the users.

I would also attempt to enable direct communication between the sender/currier/receivers and ideally positively impact user experience with the ShipIT web application.

I would further integrate the cost calculator functionality, to provide for further inputs and more complex calculations and outputs for the users. I would also attempt at automatizing the cost calculator so that it calculates the costs based on user inputs while creating the orders.

Provided additional time, I would consider attempting implementation of a live tracking system using a GPS. This would be costly but an option not many delivery operators provide for their customers which would make ShipIT stand out significantly.

5.0 References

Please include references throughout your document where appropriate. See here for a guide on referencing from the NCI library.

6.0 Appendices

6.1. Project Proposal

1.0 Objectives

This project is set out to simplify the trade of all sorts of goods between two or more individuals, corporations or between individuals and corporations.

The application aims to simplify the process of handling the deliveries and enable users to see exactly what the status of their package is, regardless of whether they are a sender, receiver, or the middleman. The status of the package is set out to include live tracking, route and stops planning and tracking as well as cost estimating and planning.

The project sets out to achieve a shortest route and stops planner and provide them to the curriers making use of the application.

A cost estimator feature which would be determined by the type, size of the goods as well as distance required for the delivery.

A delivery confirmation feature where both the currier and receiver must confirm the completion of the delivery.

Workload distribution and route analysis tools for larger organisations, where pre-planned routes can be reviewed, and the strategy may be optimized based on this feature with a goal of increasing productivity or income.

Include a scheduled pick-up feature, where drivers and users can agree on the same time of a package pick-up. More detailed information may be communicated using this feature e.g., how to access the building or the package, notes not to leave the package out in certain conditions etc.)

2.0 Background

I chose to undertake this project because I faced some of the delivery tracking and organizing challenges myself as an individual. This led to feelings of uncertainty and frustration and my aim is to improve upon existing ways of delivery planning, tracking and security. I think everyone involved in the delivery should have the right to know what exactly is happening with the package at any moment.

I will meet the objectives set out by working consistently for a set amount of time each week until the project is due. I will break down each feature into smaller requirements to keep the work needed to be done well distributed and achievable.

3.0 State of the Art

I have researched possible similar applications already on the market and there are some that may come close to what I wish to create. The applications that I have researched have been mostly targeted at large corporations for better route planning and driver tracking with less emphasis on an individual wishing to trade. This is where I believe my project would stand out and be distinguished as a new type of take on the todays delivery systems already existing, enabling an individual to schedule, plan, track their deliveries as well as have security involved in the application.

4.0 Technical Approach

I will attempt to build my application using an incremental approach as I believe this will provide me with more flexibility as well as more informed decision making when adding features to my project.

I will identify requirements of my project by breaking down the project to its core features that I wish to implement. Once I do this I will attempt and identify the main requirements for the core feature to function. Each requirement will be further broken down into separate tasks that I will keep track of using a software such as Click Up. I will add all resources used and note all progress made during the development of the requirement. The result of satisfying all requirements should be a developed and working feature of my Delivery Application. I will set deadlines for myself to keep progressing and avoid procrastination. Once all my main features are completed using the process outlined, I will attempt to either improve upon them or add additional features that I find may be useful or improve upon my existing project.

5.0 Technical Details

As for the development language I am considering using Ruby on Rails as it provides support of widely used web standards like JSON and XML for my data as well as HTML, CSS, and JavaScript that I will need for creating a user interface. Rails provides various frameworks and GEMs that I may use to create functionalities I will want to be included in my web application.

A database such as Postgres or MySQL for storing of accounts, tracking order numbers, order details and completed orders.

One of the core tools I will have to be using to achieve my project goals is to implement options for custom planning, routing, and tracking.

One important security algorithm I will be using for my web application is a Password Hash, as I want to keep the user accounts of my web application secure.

Dijkstra algorithm used for finding the shortest path from the start to end destination.

6.0 Special Resources Required

7.0 Project Plan

The goal of my web application is to provide an end user with a user interface, where they can log in and navigate the available features of the application. The goal is to have features of planning, creating, tracking, and securely closing a delivery using the application.

- User interface – an interactive user interface where the users can explore and read about the application itself to decide if they wish to try it. In case they wish to try the main features of the application they will have to create a new account or log-in with an existing account. During the development process this will represent the main skeleton of my Delivery Web Application into which I will integrate all the main features I want to include in my application. The planned amount of time needed to complete the User Interface is 2 weeks.
- Main features – core features that will define my Delivery Web application.

    1. Creating, reading, updating, and deleting an order by setting the date of the delivery and selecting a desired route resulting in the generation of a tracking number, enabling the sender, currier, and receiver to track the progress using maps and/or written updates. (2 weeks)

    2. Route planning using where the user will be able to set out their desired routes and have the deliveries tracked as they progress through the routes. This will include stops planning feature where the user can set their desired stops (4 weeks)

    3. Cost estimation where the cost will be calculated based on the amount of units or weight, type of unit and distance required and send out the output with the calculations to the sender, currier, and receiver. (1 – 2 weeks)

- Secure end of a delivery where the currier and the receiver both must confirm the completion delivery, with everyone included in the delivery to be notified of the completion. (1 – 2 weeks)

- Additional features – in case I successfully create, integrate, and test all the main features in the amount of time that is predicted to be needed I will attempt to work on additional features to improve the quality of the application.

1. Workload distribution – create an additional page for users with administrator privileges that can separate their workers as they wish, based on the amount of work that is needed in a specific area at the specific time. Different routes can be assigned to different drivers. (2 weeks)

2. Route analysis tool – where data is collected from the routes and calculates as a result of type of goods, number of stops, and time needed to complete deliveries. (2 weeks)

3. Scheduling pick-up – enable the senders, receivers, and curriers to arrange a scheduled pick-up using the Scheduling pick-up feature of the application where they can organize and schedule a specific time and a stop for a pickup of goods, by either the currier or the receiver. (1 – 2 weeks)

8.0 Testing

Unit testing – each feature and its requirements are to be tested as a part of an independent unit to test the features functionality.

Integration testing – test multiple features to determine if they work together as expected.

End-to-end testing to determine if the finally completed project is working as set out in the Project plan.

As for the finished project, the end users will be invited to try out and test the application via email. Participants will be selected voluntarily and will have to be over 18 years of age. Each willing human participant will have to provide their consent before participating and commencing the application test. Once they got a good understanding of the application they will be provided with a yes/no questionnaire in the email which will include place to give their own feedback as well.

6.1. Ethics Approval Application (only if required)

Reflective Journals

Supervision & Reflection Template

Student Name	Vedran Badrov
Student Number	X19458636

Course	BSHC Software Development 4
Supervisor	Abdul Razzaq

Month: October

What?

During the month of October, I worked on coming up with a project idea that I would enjoy working on for my final year. I had to think through couple of ideas before making my final decision as to identify the most suitable idea both for me and requirements. Once I identified an idea that felt will suit best, I started creating a document where I outlined the main features that I want to include in my project. Once I set out the main targets, I went into more details about each feature that I am planning to include.

So What?

Consider what that meant for your project progress. What were your successes? What challenges remain?

This was an important part of a project as this sets out the main idea, features, and goals of my project. If I did not have these aspects set out my project would likely shift outside my original idea, and I would not want that. The main challenge and decision I must make right now is what coding language I will use as well as identifying additional resources I will require for successfully completing my project.

Now What?

I will research all options and tools that I can before deciding and outlining the coding language and additional resources I will be using. I will read throughout details of each resource and attempt to understand how it would improve my project, to what extent and if it would help me out at all.

Student Signature	Vedran Badrov

Month: November

What?

Continuation of Project plan development and visualization of the project skeleton. I also expanded on the initial ideas of main features and underlying requirements. Minor developments of the code and exploration of needed resources.

So What?

Consider what that meant for your project progress. What were your successes? What challenges remain?

The steps I have taken in November are much needed for created a strong foundation to work from in the coming months. The main challenge now is to attempt to take the features and underlying requirements and break them into smaller tasks. This will enable me to start creating and coding my project.

Now What?

What can you do to address outstanding challenges?

I will attempt to use an online tool to break down tasks into smaller pieces, where I can also keep track of these tasks by documenting them.

Student Signature	Vedran Badrov

Month: December

What?

Reflect on what has happened in your project this month?

Completed major points of the project technical report including the executive summary, introductions to the background, aims and technology of the project as well as the structure of the technical report document. I also completed the requirements of the project, outlining main functional requirements accompanied with use case diagrams. Each requirement and use case diagram go through each step and include main, alternate and exceptional flows. The document goes through the testing of the web application and how it will be executed.

So What?

Consider what that meant for your project progress. What were your successes? What challenges remain?

The step of completing the major point of the technical report was an important step of understanding and outlining important parts of the Delivery Web Application. It helped me think about and have a better understanding of the potentials and limitations of the project.

Now What?

What can you do to address outstanding challenges?

I still have couple of parts to be completed in terms of the technical report and I am planning to do so in the month of January. I think it is important to go over the template once again and then look to start the initial build of the application while also consulting with my supervisor.

Student Signature	Vedran Badrov

Month: January

What?

Reflect on what has happened in your project this month?

During the month of January, I continued improving and completing the technical report template and finishing the parts that were missing in December. I’ve gone through the potential advantages, disadvantages, strengths, and limitations of the Delivery Web Application in more detail. Some of the missing requirements have been added alongside use cases for each.

So What?

Consider what that meant for your project progress. What were your successes? What challenges remain?

I think that it was important to go over the rest of the technical report to better prioritize and understand all upcoming requirements before commencing work on the actual web application.

Now What?

What can you do to address outstanding challenges?

Currently I am about to be starting with creating a prototype Delivery Web Application where I will attempt to implement each use case as intended in the technical report.

Student Signature	Vedran Badrov

Month: February

What?

The delivery web application got its name, and it will be called “ShipIT”. I chose this name as it has the potential to be catchy and explains well the purpose of my project. I also developed frameworks of the user interface and set up the project in Visual Studio Code.

So What?

The next challenges of the project include realising the wireframes and having the pages of the web application set up.

Now What?

During the month of February, I am starting the development of the wireframe using HTML, CSS and Ruby On Rails until they resemble the wireframes.

Student Signature	Vedran Badrov

Month: March

What?

During March I planned and developed the data requirements of the ShipIT web application. I developed the database so that it may support user account and orders. The application now enables users to register, log-in and manage their account as well as create, store, and view their orders.

So What?

This was an important part of the project as the first few important functionalities have been addressed regarding the project plan. With further development I hope that the ShipIT web application will keep representing the project plan as closely as possible.

Now What?

Future challenges include the development of different order statuses thus enabling users to view orders that are completed and closed in a separate view. This will also enable curriers to have some functionalities when it comes to orders and be able to manipulate them based on the new variables.

Student Signature	Vedran Badrov

Month: April

What?

During April I further developed the database and controllers and models of the ShipIT web application. Functions such as cost calculator, order status, associations between orders and users. Also, users have enumerated roles and different roles have different access to functionalities. Curriers have been enabled to manage orders and orders are now tied to users that created the order and curriers that accepted the order.

So What?

Important parts of the project have been developed throughout April, and the ShipIT web application looks much closer to the set-out project plan.

Now What?

As for further challenges, the ShipIT web application requires web design of the created functionalities and testing of as many as possible.	
Student Signature	Vedran Badrov

Vedran Badrov

____

Signature

6.2. Other materials used.

Any other reference material used in the project for example evaluation surveys etc.