Software Requirements Specification
for
PDQ Routing Subsystem

Version 1.2 approved

Prepared by:

Joshua Alkins 190908

Antonio Workman 190907

30/10/2020

Contents

1. Introduction

1.1. Purpose

This is a Requirements Specification document for the PDQ routing subsystem. This document describes the scope, objectives, and goals of the subsystem, as well as the subsystem's functional and nonfunctional requirements. This document is intended to direct the design and implementation the subsystem, and document changes to the requirements as the project progresses.

1.2. Document Conventions

This document was partially based on the IEEE template for System Requirements Specification documents. This document also incorporates software modelling techniques taught in UWI's SWEN program.

Any terminology unique to the project can be found in glossary.

1.3. Intended Audience and Reading Suggestions

This document is intended for:

- The project development team
- The lead project manger
- Upper management on the project steering committee
- Dr. Xu

1.4. Product Scope

The project aims to create a system to direct delivery divers between pizza factories and delivery locations. This system must include a mobile app that can be installed on delivery drivers' phones that will serve as the interface for the drivers and the subsystem, and a database to store delivery information.

Additional details can be found in the project scope statement included with this document.

1.5. References

Project GitHub page:

IEEE template for System Requirements Specification Documents: https://goo.gl/nsUFwy

2. Overall Description

2.1. Product Perspective

The product aims to provide a means for the logistics subsystem to send order information to the routing subsystem in JSON format via a REST API provide by the routing subsystem. The system then aims to use the provided order information to direct delivery drivers to the correct pizza factory to pick up an order and then direct the delivery driver to the correct delivery location.

2.2. Product Functions

The subsystem should provide the following features:

1. Allow drivers to request a delivery
2. Allow drivers to accept a delivery
3. Provide the driver with directions to the pizza factory
4. Provide the driver with directions to the delivery location
5. Provide a means for the logistic subsystem to add orders to be delivered
6. Allow administrators to add new drivers
7. Allow administrators to edit driver information

2.3. User Classes and Characteristics

Delivery Driver – PDQ employees that are responsible for collecting and delivering pizza orders.

Administrator – PDQ employees that are responsible for employing drivers.

2.4. Operating Environment

The mobile application should be able to be run on iOS devices.

2.5. Design and Implementation Constraints

CON-1 The subsystem must use a NoSQL database

2.6. Assumptions and Dependencies

Apple Maps -

3. External Interfaces

3.1. User Interface

The user interface for the system should follow the PDQ brand's colour scheme and iconography where possible.

Login Screen
The mobile app must include a login screen where drivers can enter their credentials before being granted access to the system.

Local Area Map
The mobile app must include a map screen where drivers can see the local area. This screen will be where the route to locations and directions appear.

Delivery Details Screen
The mobile app must include a screen that shows the details of a delivery before the driver and provides an option to accept the delivery.

3.2. Subsystem Interface

API
The system must provide some form of API that can be utilized by other subsystems to add order information into the routing subsystem.

4. System Features

Overview

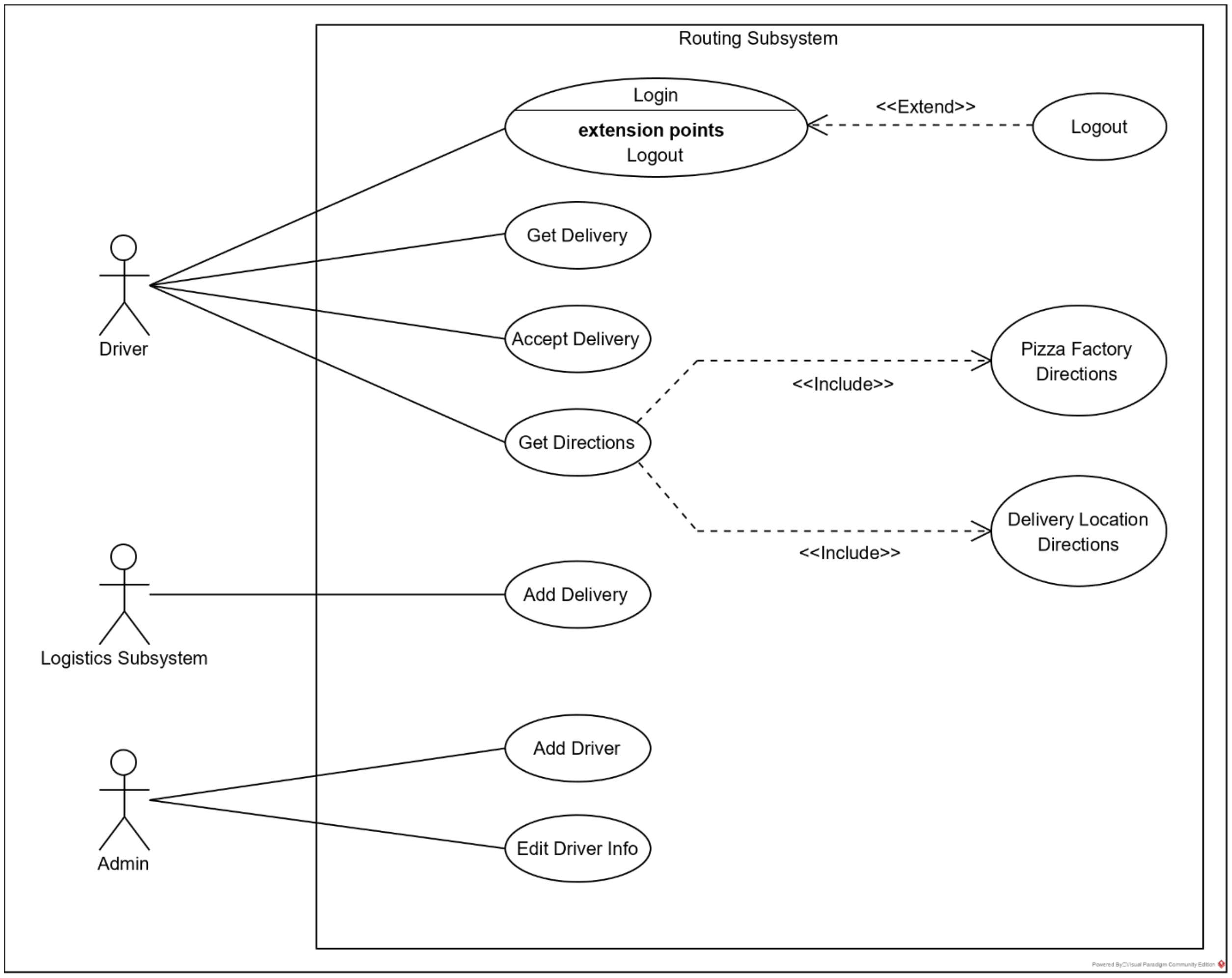

Figure 4.1 Use Case Overview Diagram

4.1. Login

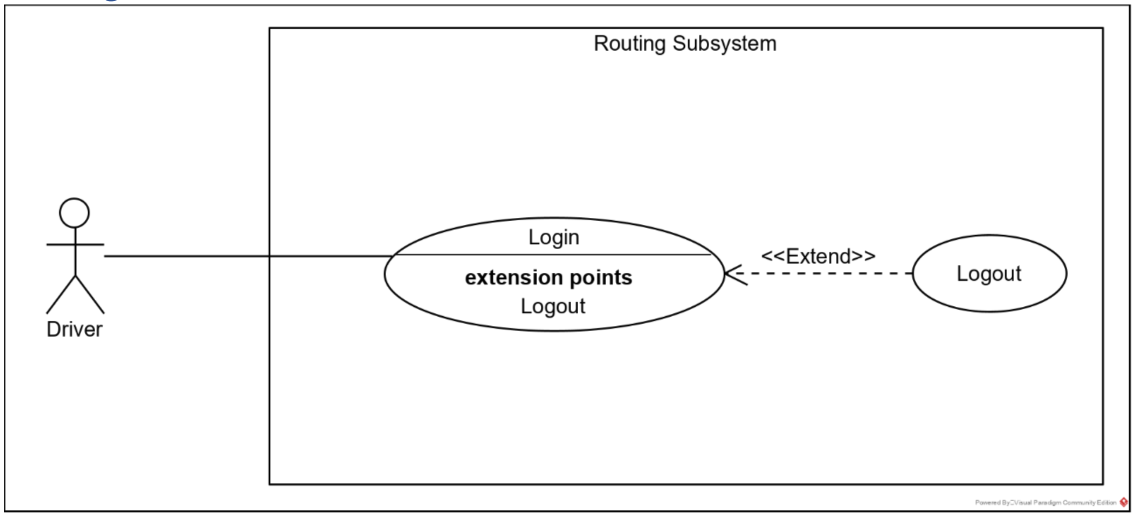

Figure 4.1.1. Login Use Case Diagram

Title	Login
Use Case ID	UC-1
Description	The driver enters their login credentials to gain access to the system.
Primary Actors	Driver
Preconditions	• The driver is registered and has an account.
Postconditions	• The driver is given access to the system.
Main Flow	1. The driver enters their login credentials 2. The system queries the database for a user with matching credentials 3. The system grants the user access and navigates to the next screen
Alternate Flows	2.1. The system cannot find an account with matching credentials 2.2. The system prompts the driver to reenter their login credentials and ensure they are correct 3.1. The driver selects logout 3.2. The system navigates to the login screen
Non-Functional Requirements	**QA-1** The system should take no more than 0.5 seconds to verify login credentials. **QA-8** Actors should not be able to access the database without logging in.

Table 4.1.1 UC-1 Login Use Case

4.1.1. Description and Priority

Drivers will be required to login to access the system’s functionality. This is to provide security for the system and so driver performance can be tracked.

Priority: Medium

4.1.2. Stimulus and Response

Upon opening the application if the driver has not already logged in, they will be presented with the login screen. They will be prompted to enter their ID and password, once this is done the driver can select submit and the system will verify the entered login information.

4.1.3. Functional Requirements

REQ-1 Drivers must be able to login to the system via the mobile app.

4.2. Get Delivery

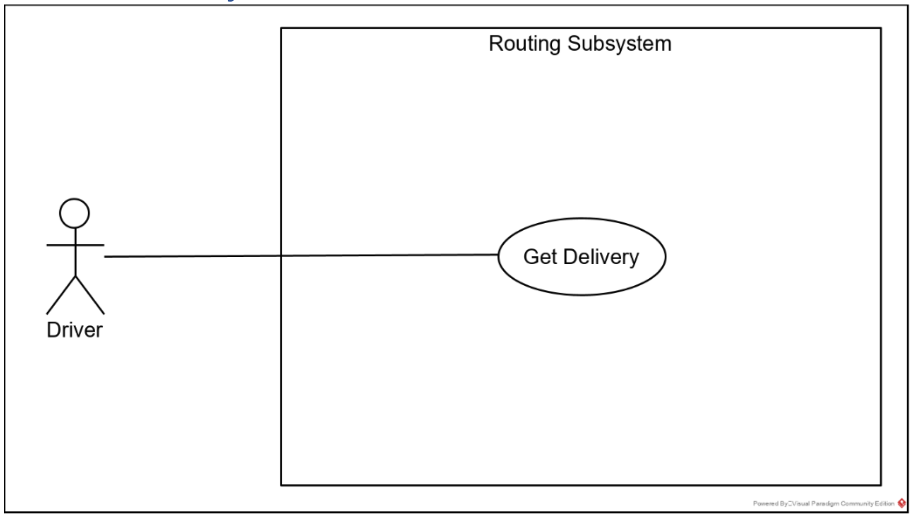

Figure 4.2.1. Get Delivery Use Case Diagram

Title	Get Delivery
Use Case ID	UC-2
Description	The driver requests a delivery from the system and is sent the information of the next available delivery.
Primary Actors	Driver
Preconditions	• The driver has logged in
Postconditions	• The driver is presented with the next available delivery
Main Flow	1. The driver selects a button to request delivery 2. The system queries the database to find the most urgent delivery 3. The system sends the driver the delivery information 4. The system presents the delivery information to the driver 5. The system prompts the driver to accept the delivery
Alternate Flows	2.1. There are no deliveries available 2.2. The system repeatedly checks the database until a delivery is found 2.2.1. The driver cancels the request 2.2.2. The system returns to the map screen 5.1. The driver declines the delivery 5.2. The system makes the delivery available to other drivers
Non-Functional Requirements	**QA-2** The system should take no more than 0.5 seconds to respond to the initial request.

Table 4.2.1. UC-2 Get Delivery Use Case

4.2.1. Description and Priority

While the driver has no active delivery, they should be able to request a new delivery from the system. Once the request is made the system should return the delivery information for the driver to review.

Priority: High

4.2.2. Stimulus and Response

While a driver has no active delivery, they will have the option to request a delivery in the form of a button accessible from the map screen. Once pressed the system should query the database to find the most urgent delivery the driver is in the area of, then present the delivery information to the driver.

4.2.3. Functional Requirements

REQ-2 Drivers must be able to get delivery information form the system.

REQ-11 The system must automatically refresh the request until a delivery is found or the driver cancels the search.

4.3. Accept Delivery

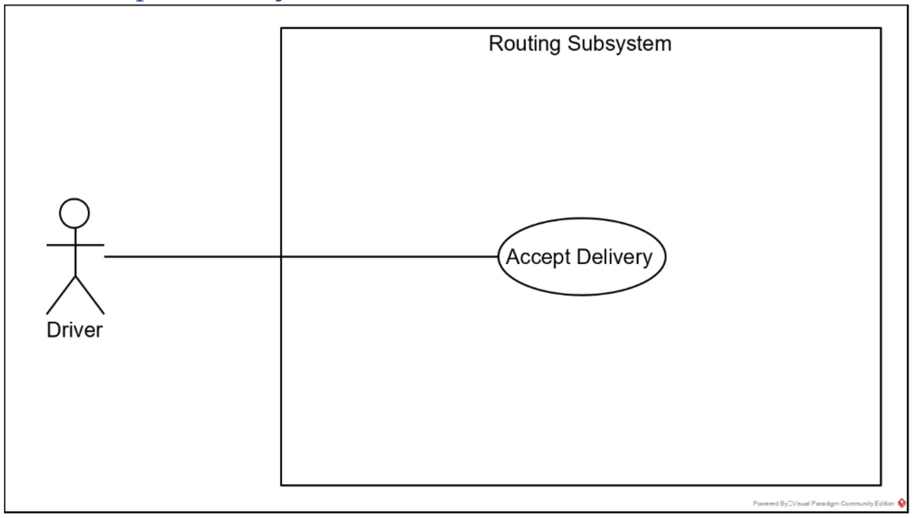

Figure 4.3.1. Accept Delivery Use Case Diagram

Title	Accept Delivery
Use Case ID	UC-3
Description	The driver agrees to deliver a specified order.
Primary Actors	Driver
Preconditions	• The driver has requested a delivery • The system has returned a delivery
Postconditions	• The system navigates to the map screen to show directions
Main Flow	1. The driver selects the accept delivery option 2. The system navigates to the map screen 3. The system updates the delivery status to show it has been accepted
Alternate Flows	1.1. The driver denies the delivery

	1.2. The system navigates to the previous screen 1.3. The system updates the delivery status to make it available to other drivers
Non-Functional Requirements	-

Table 4.3.1. UC-3 Accept Delivery Use Case

4.3.1. Description and Priority

After requesting a delivery, the driver should be given the option to accept or deny the delivery, based on whether or not they are currently able to perform the delivery.

Priority: Medium

4.3.2. Stimulus and Response

Once the system has returned delivery information at the request of the driver, the driver will be presented with the option to accept or decline the delivery. Once accept option is selected the system navigates to the map screen to provide directions and the system will update the database to reflect that the order has been accepted and record the ID of the driver responsible for the delivery. If the decline option is selected the system navigates to the previous screen.

4.3.3. Functional Requirements

REQ-3 Drivers must be able to accept deliveries.

REQ-4 Drivers must be able to decline deliveries.

REQ-5 The system must record the status of each delivery.

4.4. Get Directions

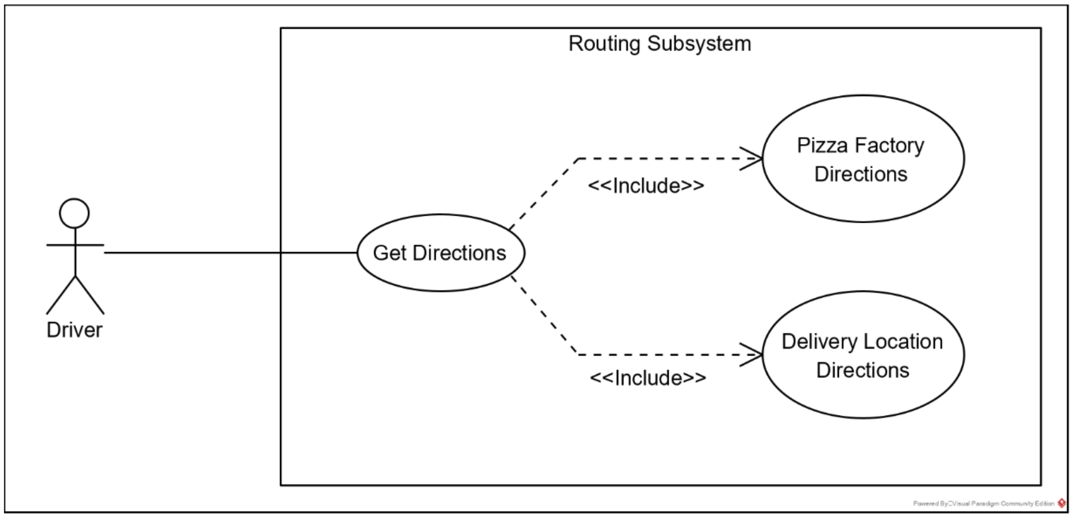

Figure 4.4.1. Get Directions Use Case Diagram

Title	Get Directions
Use Case ID	UC-4
Description	The driver is presented with directions to the correct pizza factory and then to the delivery location.
Primary Actors	Driver
Preconditions	• The driver has accepted a delivery
Postconditions	• The system navigates to back to the map screen

	• The system presents no directions • The driver is able to request a new delivery
Main Flow	1. The system uses the delivery information to calculate a route to the pizza factory 2. The system displays the route to the pizza factory 3. The driver navigates to the pizza factory 4. The system displays a button to collect the order 5. The driver selects the prompt 6. The system displays the order information 7. The system displays a button to confirm the order has been collected 8. The system returns to the map screen 9. The system displays the route to the delivery location 10. The user navigates to the delivery location 11. The system displays a button to confirm the order has been delivered 12. The user selects the button 13. The system returns to the maps screen without directions present
Alternate Flows	-
Non-Functional Requirements	-

Table 4.4.1. UC-4 Get Directions Use Case

4.4.1. Description and Priority

Once a delivery has been accepted the system will present direction to the appropriate pizza factory to collect the order, and then to the delivery location to drop off the order.

Priority: Very High

4.4.2. Stimulus and Response

After a delivery is accepted by the driver, directions to the appropriate pizza factory will be displayed on the map. Once the driver navigates to the pizza factory, they will be given the option to confirm they have collected the order. After this option is selected the system will present directions to the delivery location, and the system will reflect that the order is in transit. Once the driver has navigated to the delivery location, they will be presented the option to confirm that the order has been delivered, which once selected will return the driver to the map screen without directions, ready to request a new order, and the system will update the database to reflect that the order has been delivered.

4.4.3. Functional Requirements

REQ-6 The system must provide directions to a delivery’s corresponding pizza factory.

REQ-7 The system must provide directions to a delivery’s corresponding delivery location.

4.5. Add Delivery

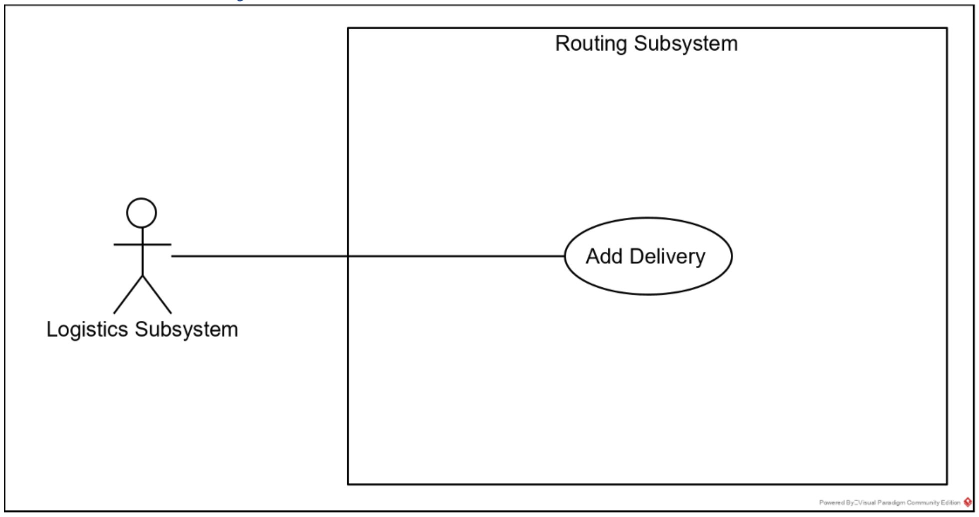

Figure 4.5.1. Add Delivery Use Case Diagram

Title	Add Delivery
Use Case ID	UC-5
Description	The logistic subsystem adds an order into the routing subsystem via an API.
Primary Actors	Logistics Subsystem
Preconditions	• Both subsystems are running
Postconditions	• The order is added to the routing subsystem's database • The logistics subsystem receives a confirmation code
Main Flow	1. The logistic subsystem sends the order information to the routing subsystem via the API 2. The routing subsystem checks the information 3. The routing subsystem adds the order to the database 4. The routing subsystem returns a confirmation code
Alternate Flows	2.1. The routing subsystem returns an error code 3.1. The routing subsystem returns an error code
Non-Functional Requirements	**QA-3** The system should take no longer than 0.1 seconds to send a response code.

Table 4.5.1. UC-5 Add Delivery Use Case

4.5.1. Description and Priority

Once the logistics subsystem has collected all the information related to an order, it will have to send the information to the routing subsystem to be delivered.

Priority: Very High

4.5.2. Stimulus and Response

The logistics subsystem calls the API provided by the routing subsystem and attaches the order information. The routing subsystem takes this information and adds it to the database and returns a confirmation code.

4.5.3. Functional Requirements

REQ-8 The logistics subsystem must be able to add deliveries into the routing subsystem database.

4.6. Add Driver

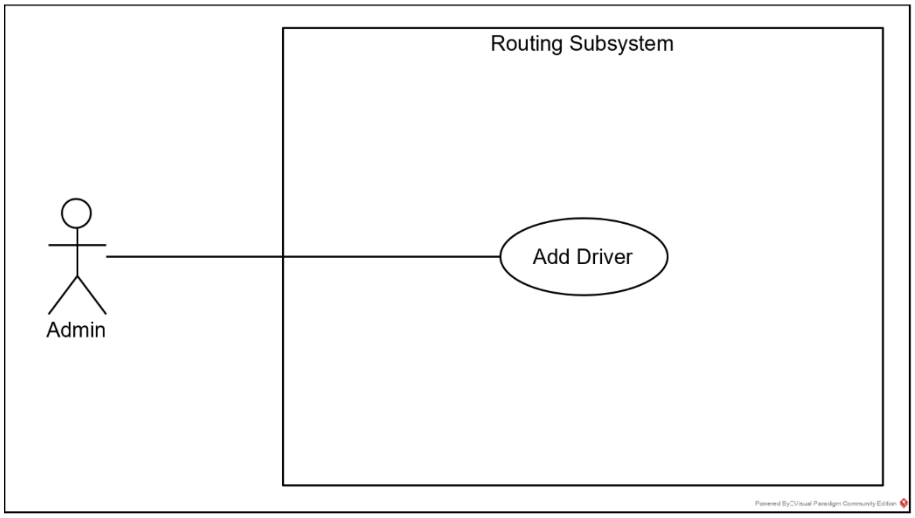

Figure 4.6.1. Add Driver Use Case Diagram

Title	Add Driver
Use Case ID	UC-6
Description	The administrator adds a new driver into the database.
Primary Actors	Administrator
Preconditions	• The administrator has access to the system
Postconditions	• The new driver's information is in the database • The new driver can login
Main Flow	1. The system presents a driver registration form 2. The administrator enters the new driver's information 3. The administrator selects submit 4. The system reviews the information 5. The system adds the information to the database 6. The system displays a success message
Alternate Flows	4.1. The displays an error message 4.2. The system the administrator to ensure all information was entered correctly
Non-Functional Requirements	**QA-4** The system should take no longer than 0.5 seconds to respond to a submission. **QA-5** The system should prevent information in an incorrect format from being submitted to the database.

Table 4.6.1. UC-6 Add Driver Use Case

4.6.1. Description and Priority

When hiring new delivery drivers, administrators will need to be able to create accounts for the drivers to access the system.

Priority: Low

4.6.2. Stimulus and Response

The administrator accesses the system via a web browser, selects add driver, and is presented with a form to enter the driver information. The administrator enters the information and selects submit. The system then verifies the information is entered correctly, adds the new driver to the database, and returns a success message. If any fields in the form were missing or written in the incorrect format the system will return an error message prompting the administrator to check the information is entered correctly.

4.6.3. Functional Requirements

REQ-9 Administrators must be able to add new drivers into the routing subsystem database.

4.7. Edit Driver Information

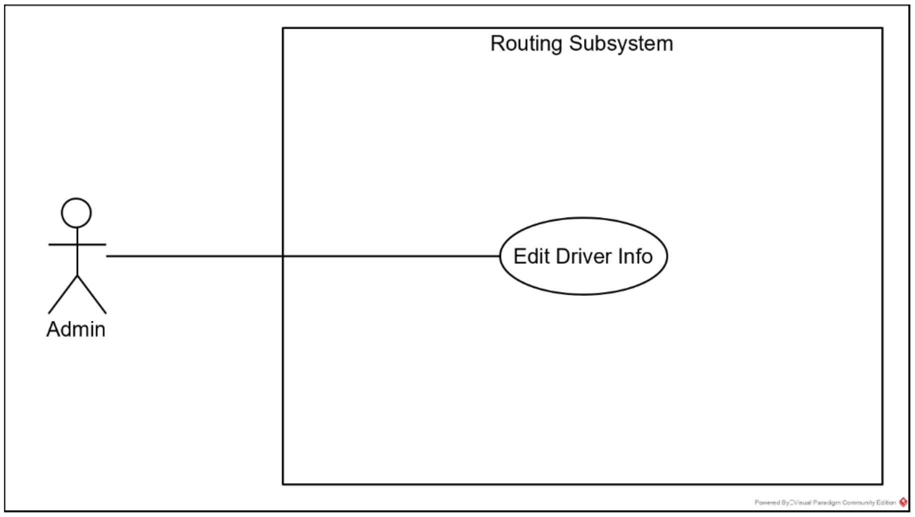

Figure 4.7.1. Edit Driver Information Use Case Diagram

Title	Edit Driver Information
Use Case ID	UC-7
Description	The administrator changes a driver's information that is stored in the database.
Primary Actors	Administrator
Preconditions	• The administrator has access to the system • The driver exists in the system
Postconditions	• The information in the database has been altered
Main Flow	1. The administrator enters the ID of a driver 2. The system queries the database for the driver 3. The system displays a form filled out with the driver's current information 4. The administrator edits the necessary fields 5. The administrator selects submit 6. The system reviews the entered data 7. The system updates the database 8. The system displays a success message
Alternate Flows	4.1. The administrator selects cancel 6.1. The system the administrator to ensure all information was entered correctly

Non-Functional Requirements	-

Table 4.7.1. UC-7 Edit Driver Information Use Case

4.7.1. Description and Priority

The administrator may need to change a driver’s information to reflect a change that has occurred or to correct an error.

Priority: Very Low

4.7.2. Stimulus and Response

The administrator accesses the system via a web browser, selects edit driver information, and is presented with a field to enter the driver’s ID, to search for that driver. The system then queries the database and return the driver’s current information in a form that can be edited. If the driver ID is not in the database, the system will return an error message saying the driver cannot be found. The administrator can then change the necessary information and select submit. The system then verifies the information and updates the database. If information is missing or entered in an incorrect format the system will return an error message prompting the administrator to ensure the data entered is correct.

4.7.3. Functional Requirements

REQ-10 Administrators must be able to edit driver information.

5. Other Nonfunctional Requirements

QA-6 The system should support 500 concurrent users.

QA-7 The system should store driver security information in an encrypted format.

QA-9 The system should be available 100% of the time during working hours 9am – 10pm Monday – Saturday.

Appendix A: Glossary

Pizza Factory – Locations that produce pizza for delivery

Logistic Subsystem – A subsystem of PDQ’s pizza order and delivery management system.

JSON – A format for storing and transporting data.

API – Application Programming Interface, allows software to interact with each other.