Software Requirements Specification

for

SpeedyDrop

Version 1.1 approved

Prepared by Mommin Amir, Qaisar Mateen, Abdullah Dar, Usman Faisal

31-March-2024

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
SpeedyDrop-1.0	30-3-2024	Vague description	1.0

1. Introduction

1.1 Purpose

The Purpose of this Software Requirements Specification (SRS) document for SpeedyDrop is to outline the specific objectives and goals of the app's development. It serves to define the scope of the e-commerce platform, specify its functionalities, and establish clear criteria for successful implementation. This document aims to provide a comprehensive understanding of the requirements and expectations for SpeedyDrop, ensuring that all stakeholders have a unified vision throughout the development process.

1.2 Document Conventions

Standard Font used throughout the document is Arial font. The font size for the paragraph test is 11 and for the Heading's level wise is 18, 14 and 12 respectively. Priorities for higher-level requirements are assumed to be inherited by detailed requirements

1.3 Intended Audience and Reading Suggestions

The intended audience for this Software Requirements Specification (SRS) document includes developers, project managers, marketing staff, users, testers, and documentation writers. Developers will find detailed technical requirements and system architecture information. Project managers will gain insights into project scope, timelines, and resource allocation. Marketing staff will understand the app's features and target audience. Users will learn about app functionalities and how to use them. Testers will find test cases and scenarios. Documentation writers will use this document as a reference for user manuals and help guides.

The document is organized into sections covering introduction, overall description, specific requirements, system features, external interfaces, non-functional requirements, and appendices. It is suggested to start with the overview sections to understand the app's purpose and scope, then proceed to specific requirements relevant to each reader type. Developers should focus on technical details, while project managers and marketing staff should review overall description and system features. Users can refer to functional requirements, and testers can delve into test cases and scenarios provided.

1.4 Product Scope

The product scope of SpeedyDrop encompasses an e-commerce mobile app designed as a marketplace for shop owners to list their products, enabling users to browse, purchase, and have items delivered by riders. The app's purpose is to facilitate seamless online shopping experiences, offering convenience and accessibility to both sellers and buyers. Its objectives include increasing sales opportunities for shop owners, providing a user-friendly interface for customers, and optimizing delivery logistics through rider integration. This aligns with corporate goals of expanding market reach, enhancing customer satisfaction, and streamlining business operations in the e-commerce sector. For a more detailed vision and scope, please refer to the dedicated document.

2. Overall Description

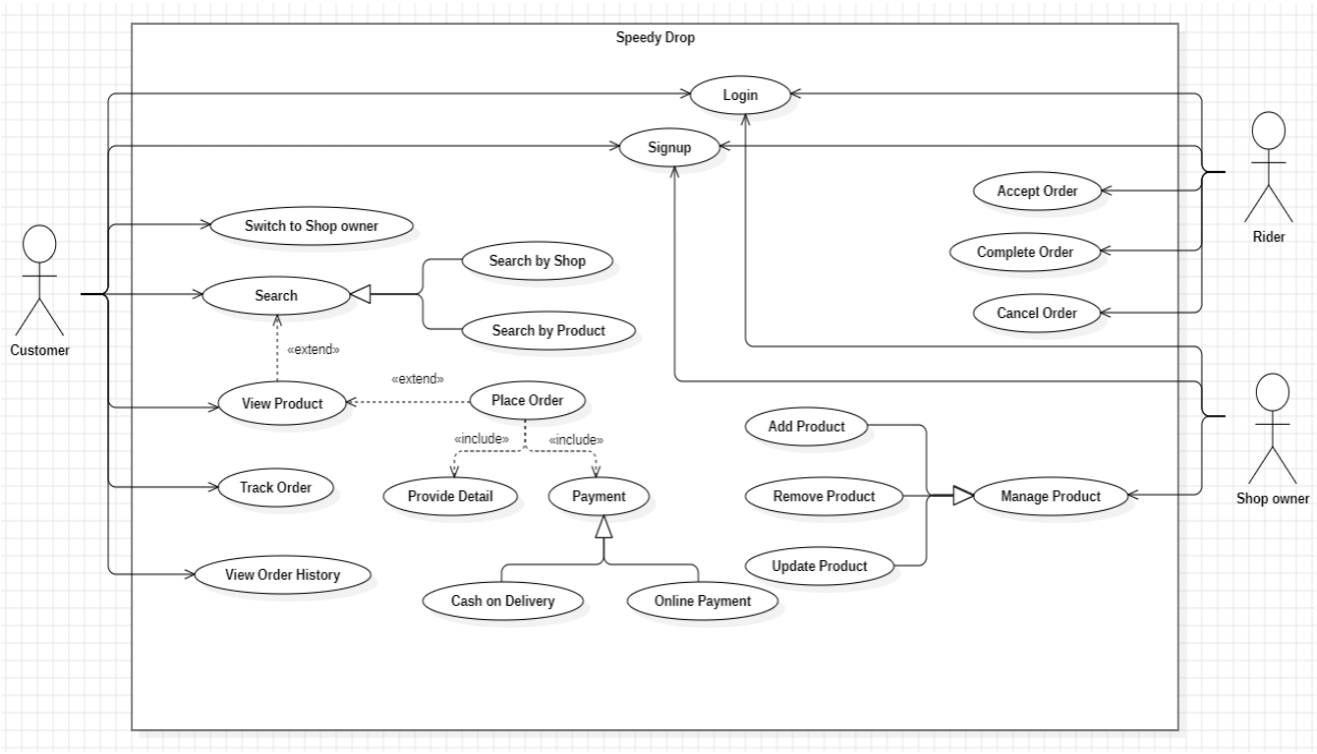
2.1 Product Perspective

The product perspective of SpeedyDrop within this Software Requirements Specification (SRS) outlines its position as a standalone e-commerce mobile app, distinct from any existing systems. It is not a replacement for other systems but rather a new, self-contained product aimed at providing a comprehensive solution for online shopping and delivery services. SpeedyDrop is designed to operate independently, connecting users directly with shop owners and delivery riders through its platform.

2.2 Product Functions

To understand the Product Functions of this application, the use case diagram, DFD diagrams along with the state diagrams are given below

2.2.1 Use-Case Diagram



2.3 User Classes and Characteristics

SpeedyDrop is designed to cater to different user classes, each with distinct characteristics and usage patterns. Shop Owners represent a key user class, engaging regularly with the app for tasks such as managing products, inventory, orders, and analytics. Their technical expertise may vary, from basic to advanced understanding of e-commerce platforms. Customers, another critical user

class, use the app frequently for browsing and purchasing products. They typically have basic technical skills and expect intuitive interfaces, secure payment options, order tracking, and personalized recommendations. Riders, while essential for delivery services, have a usage frequency that aligns with delivery requests, route optimization, real-time tracking, and communication tools. While all user classes are important, the primary focus is on Shop Owners and Customers due to their direct engagement with core app functionalities.

2.4 Operating Environment

The Application will be designed for smartphone more specifically for Android phones and it will co-exist with every other application in the mobile

2.5 Design and Implementation Constraints

The design and implementation constraints for SpeedyDrop in this SRS include:

- *Corporate policies and regulatory requirements*
- *Hardware limitations and compatibility considerations*
- *Integration with external applications and services*
- *Specific technologies, tools, and databases to be used*
- *Parallel processing for efficiency*
- *Security measures like encryption and access controls*
- *Adherence to design conventions and programming standards for maintainability.*

These constraints guide the development process and inform decisions related to architecture, technology stack, security measures, and coding practices throughout the lifecycle of SpeedyDrop.

2.6 User Documentation

There will be no User manual provided for this application. Instead for training purposes we will send our representative Mommin Amir, who will professionally train the clients for the specified time frame.

2.7 Assumptions and Dependencies

The assumptions and dependencies for SpeedyDrop outlined in this SRS are critical considerations for the project's success. Assumptions include factors like the availability and compatibility of third-party components essential for payment processing, mapping services, and communication tools. Additionally, assumptions extend to the presence of adequate server infrastructure and stable networks to ensure optimal app performance, along with compliance with legal and regulatory standards for e-commerce operations.

On the other hand, dependencies encompass the integration with existing APIs for seamless functionality such as payment gateways, mapping services, and delivery tracking. Moreover, the project relies on reusing software components from previous projects for specific functionalities, unless these dependencies are already documented elsewhere. Ensuring the accuracy and continuity of these assumptions and dependencies is crucial to avoiding potential project setbacks or disruptions.

3. External Interface Requirements

3.1 User Interfaces

The User Interface for this application will follow the modern mobile application layout and will be sleek and light, The GUI image for each module and page will be provide later. That GUI will be in Figma that will be used by the frontend developer in the application

3.2 Hardware Interfaces

The GUI will interact with the Android Kernel for rendering of widgets and will be expected to run on all the Android mobile devices. And for the communication between modules and interface the standard protocols will be used.

3.3 Software Interfaces

In this application the Firebase will used be used as a Database and the mobile application will be developed in Flutter Framework, No software components from other previous projects have been reused in the project. For the message in and out flow refer to the 2.2.2, 2.2.3 & 2.2.4 section.

3.4 Communications Interfaces

The backend database for the application will be Firebase. Which is a cloud-based Data Base. And for the communication with the Firebase and our app's backend the fire base API will be used and user will be expected to be connected to the internet to fully utilize the app.

4. System Features

All the Main functional requirements detail for this application are described in detail below.

4.1 Order Product

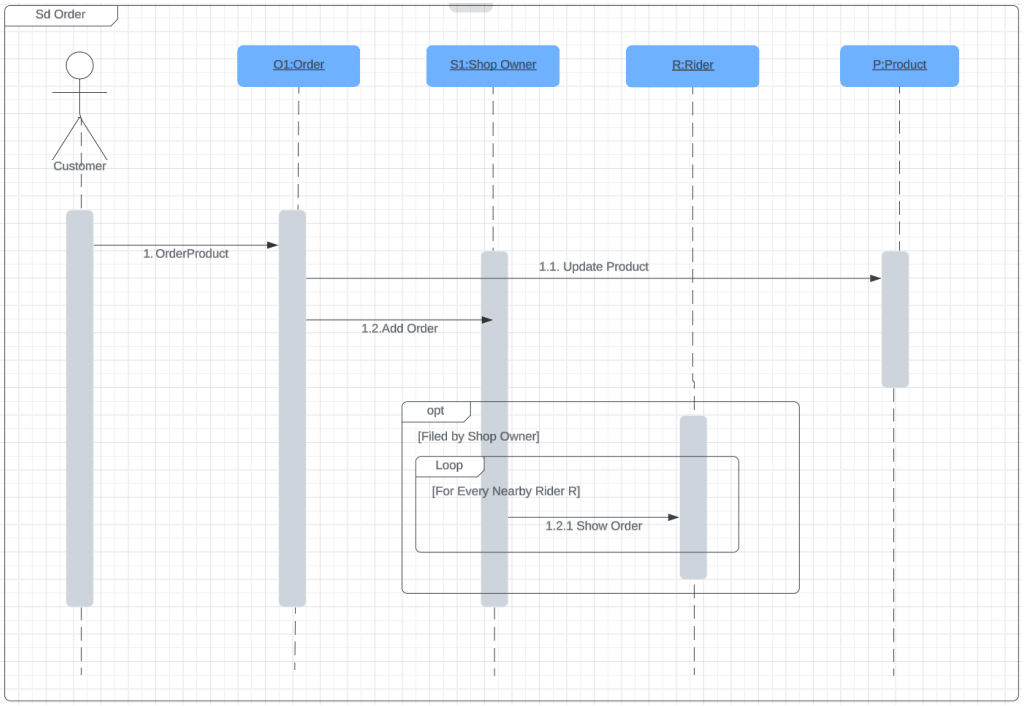
The customer will be able to order products they want from the app; the precondition will be that they have to be logged in to be able to do that

4.1.1 Use Case Description.

Use Case#1	Order Product
Description	It involves customer ordering product from the app, and shop owner getting notice and Order available display to relevant riders
Actors	Customer, Shop Owner, Rider
Goals	Giving customer option to buy products, Giving Shop owner a way to sell products

Pre-Condition	User logged in, Have active internet connection
Post-Condition	The user successfully orders the product
Basic Flow	1- Customer order a Product by tapping the order button on a Product 2- System checks and update the products inventory in data base and add order to the Orders table 3- Order added to Shop owner 4- Shop owner declare order ready to pick 5- All riders nearby are notified about the availability of order
Alternative Flow	1- If system is down or user connection is slow then error is shown to user 2- If product is not available then user is notified

4.1.2 Sequences Diagram.



4.1.3 Functional Requirements

This feature full fills the requirement that a user could be able to place order. In order to implement this requirement, the database should be working and active, a payment method should be added and shop owner panel should be completed to. In case of any error the product should redirect the user to previous page and display the error message. The requirement fulfilled are:

REQ-1: Customer should be able to buy products

REQ-2: Shop Owner should be able to sell products

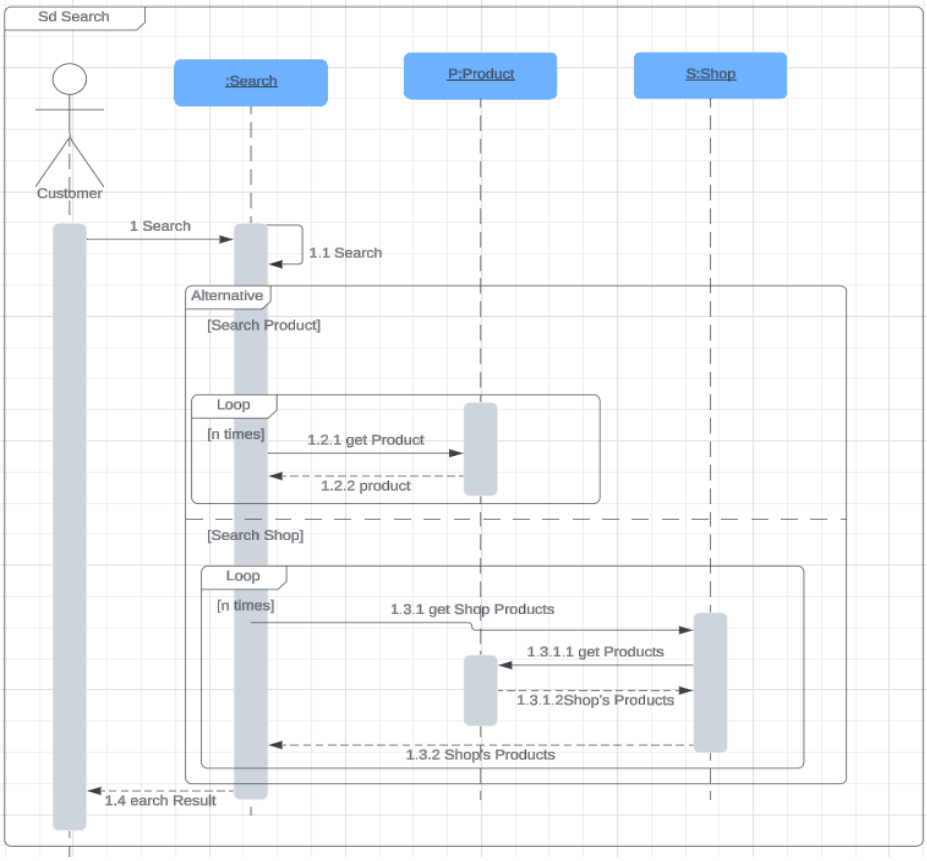
4.2 Search Product

The customer will be able to search products or search shops for their products to save their time.

4.2.1 Use Case Description.

Use Case#2	Search Product
Description	It will allow customer to search products or products of specific shop
Actors	Customer
Goals	Giving Customer option to browser products by searching
Pre-Condition	Have active internet connection
Basic Flow	1- User Enter the Search text and filter 2- If Filter is Search for Products, then system make a list of all the products matching the text. 3- If Filter is Search for Shop, Then the system makes a list of all the products that shop sells 4- It returns the list of Products to the Customer's Interface to display it
Alternative Flow	1- If system is down or user connection is slow then error is shown to user

4.2.2 Sequences Diagram.



4.2.3 Functional Requirements

This function covers oner functional requirement that is 'Customer will be able to search the products via products or Shops filter'

REQ-1: Search for Products via filter

4.3 Add New Product

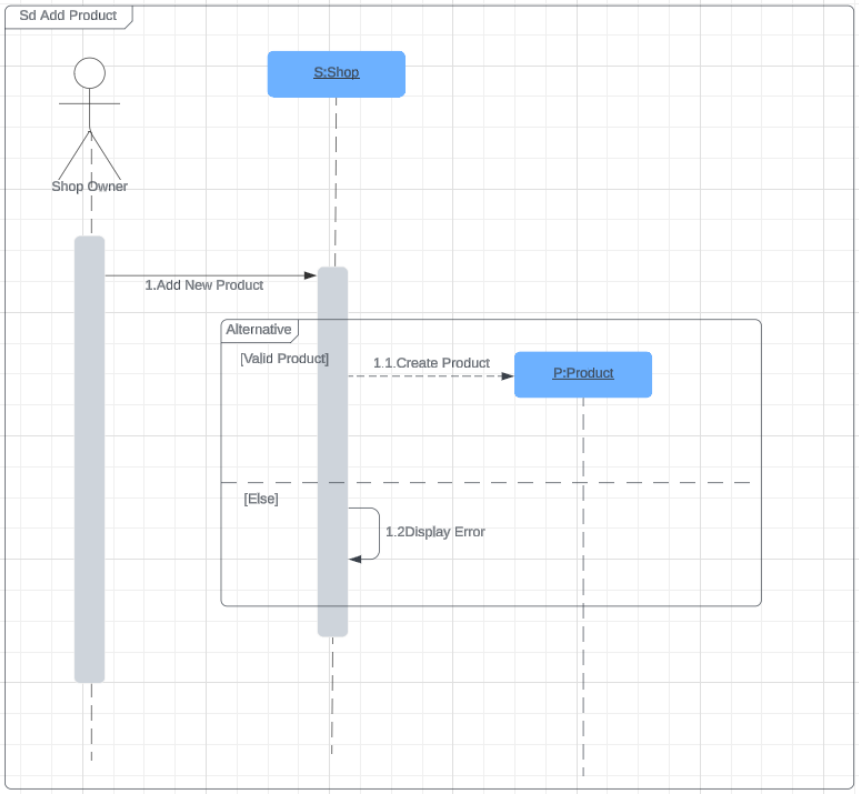
This functionality will allow the shop owner to add new products to his virtual Shop on the application.

4.3.1 Use Case Description.

Use Case#3	Update Product

Description	It will allow shop owner to manage their products by adding new products
Actors	Shop Owner
Goals	Helping Shop Owner to Maintain and Manage his shop's products
Pre-Condition	User logged in, Have active internet connection
Post-Condition	New product will be successfully be added
Basic Flow	1- The Shop owner goes to the manage app section and tap the add button 2- He will be directed to add product panel 3- He will fill data in the provided field and tap create button 4- If data is valid new product will be created and added in the data base 5- Shop owner will receive a success notification
Alternative Flow	1- If system is down or user connection is slow then error is shown to user 2- If the entered data is not valid then it will display error

4.3.2 Sequences Diagram.



4.3.3 Functional Requirements

This function partially satisfies one functional requirement

REQ-1: Shop Owner can manage Products

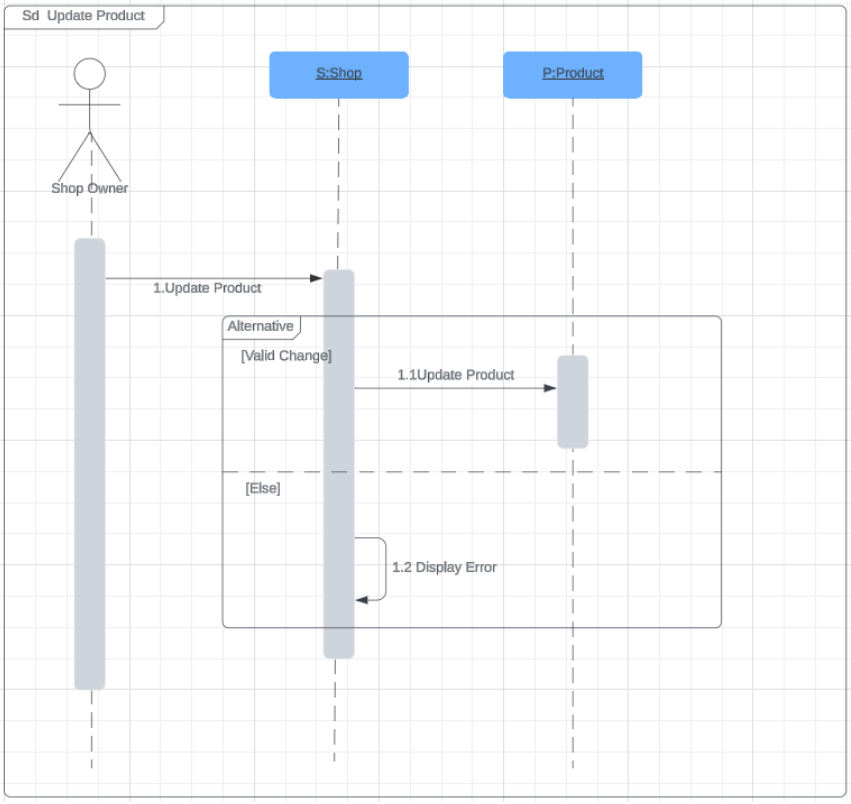
4.4 Update Product

This functionality will allow the shop owner to Update existing products to his virtual Shop on the application.

4.4.1 Use Case Description.

Use Case#4	Update Product
Description	It will allow shop owner to manage their products by updating existing products
Actors	Shop Owner
Goals	Helping Shop Owner to Maintain and Manage his shop's products
Pre-Condition	User logged in, Have active internet connection
Post-Condition	Product will be updated Successfully
Basic Flow	1- The Shop owner goes to the manage app section and tap the product to update and then tap the update product button 2- A new panel will be open where shop owner will update the product data. 3- When the shop owner is satisfied then he will tap the save button 4- Now if the data is valid the product will be updated in the database 5- The Shop owner will get success message
Alternative Flow	1- If system is down or user connection is slow then error is shown to user 2- If the entered data is not valid then it will display error

4.4.2 Sequences Diagram.



4.4.3 Functional Requirements

This function partially satisfies one functional requirement

REQ-1: Shop Owner can manage Products

4.5 Remove Product

This functionality will allow the shop owner to remove products from his virtual Shop on the application.

4.5.1 Use Case Description.

Use Case#5	Remove Product
Description	It will allow shop owner to manage their products by removing products
Actors	Shop Owner
Goals	Helping Shop Owner to Maintain and Manage his shop’s products
Pre-Condition	User logged in, Have active internet connection
Post-Condition	Product will be removed from the shop

Basic Flow	1- The Shop owner goes to the manage app section 2- The shop Owner tap the product he wants to remove 3- Then he taps the remove button on that product 4- The product will be removed
Alternative Flow	1- If system is down or user connection is slow then error is shown to user 2- If the entered data is not valid then it will display error

4.5.2 Sequences Diagram.

No sequence diagram for this functionality

4.5.3 Functional Requirements

This function partially satisfies one functional requirement

REQ-1: Shop Owner can manage Products

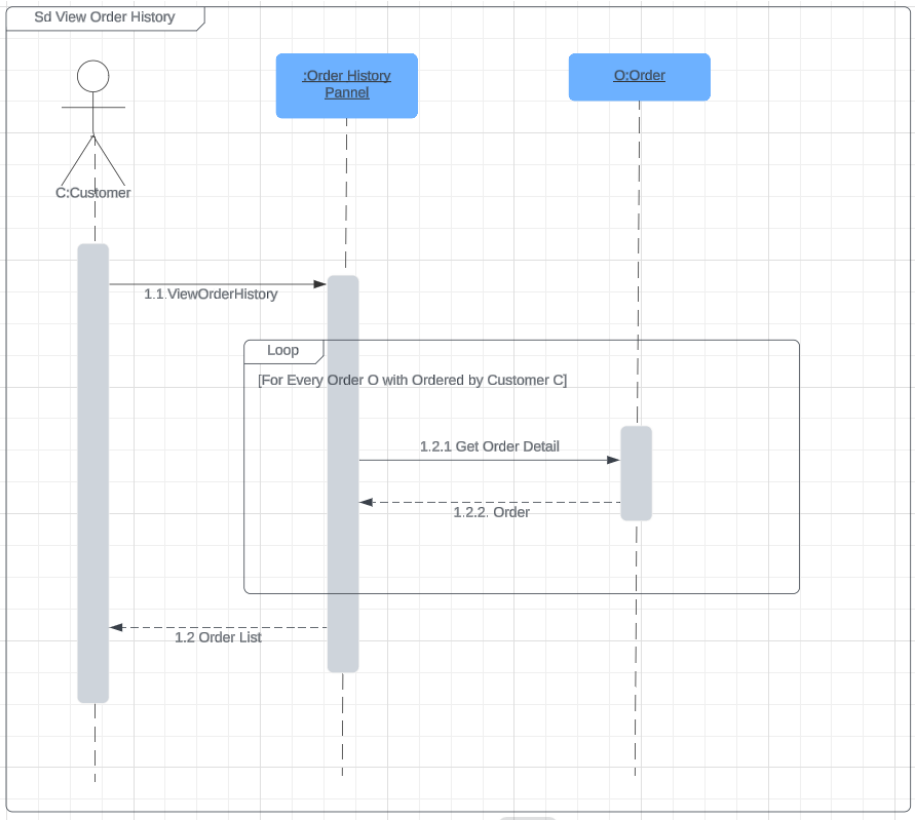
4.6 View Order History

This functionality will allow the customer to view his order history in the app.

4.6.1 Use Case Description.

Use Case#6	View Order History
Description	It will allow Customer to view his order history
Actors	Customer
Goals	Helping Customer to View his past data in the app
Pre-Condition	User logged in, Have active internet connection
Basic Flow	1- The customer goes to the Order History section in the app 2- The system will run a loop to get all the orders with receiver being the customer him self from the order data base 3- The system will store these orders in the list 4- The system will display the Order on the users Screen
Alternative Flow	1- If system is down or user connection is slow then error is shown to user 2- If there are no order then it will display an empty screen image

4.6.2 Sequences Diagram.



4.6.3 Functional Requirements

This function fully satisfies one functional requirement

REQ-1: Customer must be able to view his previous order history

4.7 Accept Order

This functionality will allow the rider to accept the order being shown to him for delivery

4.7.1 Use Case Description.

Use Case#7	Accept Order
Description	It will allow the rider to accept the orders
Actors	Rider
Goals	Helping Rider interact with incoming orders for delivery
Pre-Condition	User logged in, Have active internet connection
Post-Condition	The order will be assigned to the Rider
Basic Flow	1- The Rider will go to available order section

	2- Here he will be able to view available orders 3- By pressing the accept button he will accept the order 4- The system will update the order status in the data base and assign it to the rider
Alternative Flow	1- If system is down or user connection is slow then error is shown to user 2- If someone else accepted the order first then it will display the message

4.7.2 Sequences Diagram.

No sequence diagram for this functionality

4.7.3 Functional Requirements

This function satisfies one requirement

REQ-1: Rider can accept orders

4.8 Cancel Order

This functionality will allow the rider to accept the order being shown to him for delivery

4.8.1 Use Case Description.

Use Case#8	Cancel Order
Description	It will allow the rider to Cancel the orders
Actors	Rider
Goals	Helping Rider interact with incoming orders for delivery
Pre-Condition	User logged in, Have active internet connection
Post-Condition	The order will be un-assigned to the Rider
Basic Flow	1- The Rider will go to active order section 2- Here he will be able to view active orders 3- He will press the cancel and confirm button 4- The system will check if the order can be canceled 5- If yes then the order will be cancelled
Alternative Flow	1- If system is down or user connection is slow then error is shown to user 2- If order cant be cancelled then it will display it to rider

4.8.2 Sequences Diagram.

No sequence diagram for this functionality

4.8.3 Functional Requirements

This function satisfies one requirement

REQ-1: Rider can cancel orders

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The performance requirements for SpeedyDrop in this SRS specify the expected performance under various circumstances. These include response time for user actions such as product browsing, order placement, and delivery tracking. The rationale behind these requirements is to ensure a smooth and efficient user experience, minimizing wait times and delays. Specific timing relationships for real-time systems, such as real-time order updates and notifications, are also defined to meet user expectations for immediacy and accuracy. These requirements guide developers in making design choices that prioritize speed, responsiveness, and reliability in the app's functionality.

5.2 Safety Requirements

Safety requirements for SpeedyDrop in this SRS focus on mitigating potential risks and ensuring user safety during product use. This includes safeguards to prevent loss, damage, or harm that could result from app usage, such as secure payment processing to protect financial information and encryption measures to safeguard user data. Actions must be taken to ensure data privacy and secure communication channels to prevent unauthorized access.

5.3 Security Requirements

The security requirements for SpeedyDrop outlined in this SRS focus on protecting user data and ensuring a secure environment for app usage. This includes implementing encryption for data transmission and storage, establishing secure authentication mechanisms for user identity verification, and enforcing access controls to prevent unauthorized access to sensitive information.

5.4 Software Quality Attributes

The software quality attributes for SpeedyDrop in this SRS encompass adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. These attributes are crucial for both customers and developers. Specific, quantitative, and verifiable metrics will be established to measure each attribute's performance. For

example, adaptability will be measured by the app's ability to accommodate changing user needs, while usability will be assessed through user feedback and testing. The relative preferences for these attributes prioritize reliability, usability, and maintainability to ensure a seamless and efficient user experience while facilitating easy maintenance and future updates.

5.5 Business Rules

The business rules for SpeedyDrop outlined in this SRS specify operating principles related to the product. These include defining which individuals or roles can perform specific functions under specific circumstances, such as shop owners listing products, customers placing orders, and riders delivering products. While these rules are not functional requirements themselves, they imply certain functional requirements to enforce the rules effectively and ensure smooth operation of the app according to business guidelines and practices.

Appendix A: Glossary

SRS (Software Requirements Specification): *A document that outlines the requirements, functionalities, and constraints of a software system.*

SpeedyDrop: *The e-commerce mobile app being developed, facilitating online shopping and delivery services.*

Shop Owners*: Users who list their products on SpeedyDrop for sale.*

Customers*: Users who browse, purchase, and track orders on SpeedyDrop.*

Riders: *Users responsible for delivering products to customers.*

Third-party Components*: Software or services provided by external vendors for integration into SpeedyDrop, such as payment gateways and mapping services.*

APIs (Application Programming Interfaces): *Interfaces that allow SpeedyDrop to communicate and integrate with external systems and services.*

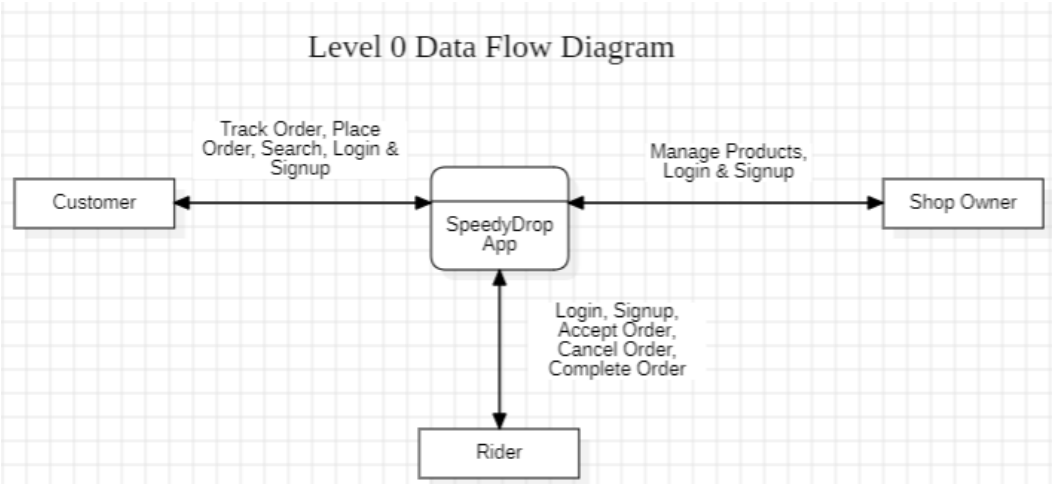
Server Infrastructure*: Hardware and software resources used to host and run SpeedyDrop.*

Compliance*: Adherence to legal and regulatory standards governing e-commerce operations, data privacy, and consumer protection.*

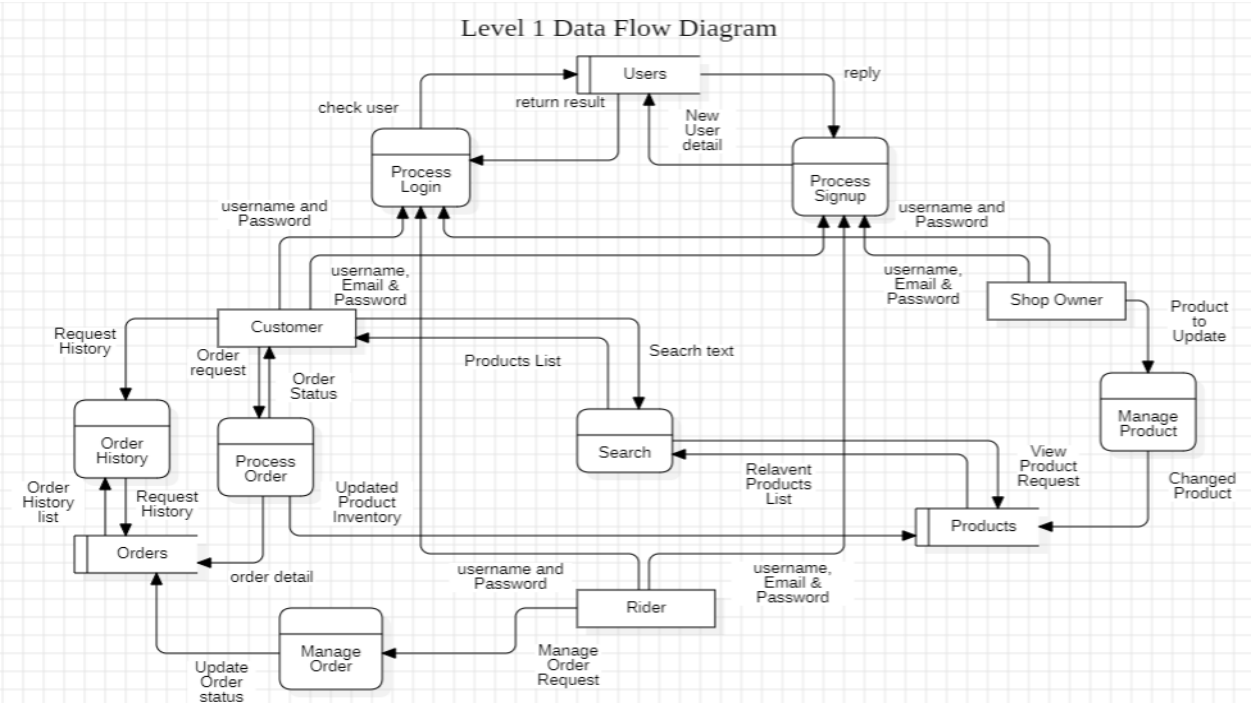
Dependencies*: External factors or components that SpeedyDrop relies on for functionality, such as APIs and reusable software components.*

Appendix B: Analysis Models

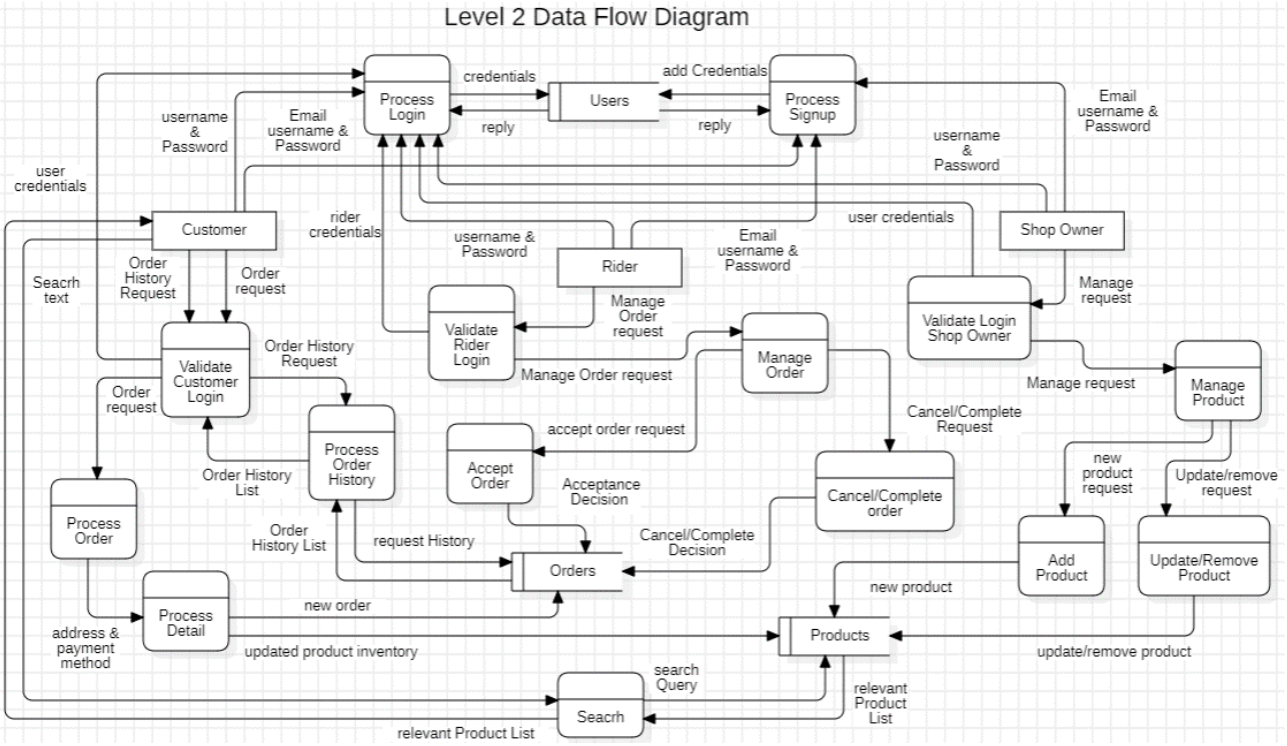
DFD Level 0:



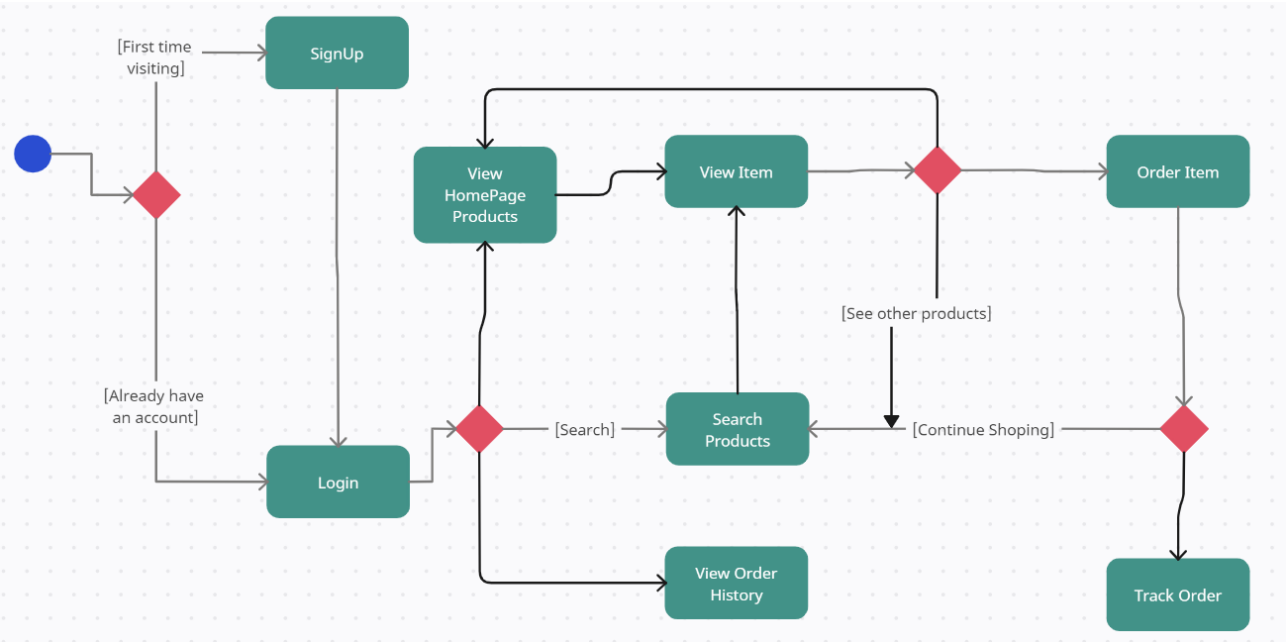
DFD Level 1



DFD Level 2



State Diagram:



Appendix C: To Be Determined List

- *The Figma Screen Shots for the Frontend of the application*
- *The API provider for maps*
- *The algorithm for the order tracking*
- *Sequence diagram of some of the functionalities (cut short by the board)*