**Logo, company name

Description automatically generated**

**COMSATS University Islamabad (CUI)**

Software Design Description   
(SDS DOCUMENT)

for

**Stock Logistics**

Version 1.0

***By***

**Waqas Ahmad CIIT/FA19-BCS-087/ISB**

**Ahmed Habib Pandit CIIT/FA19-BCS-094/ISB**

***Supervisor*Mr. Asif Muhammad**

*Bachelor of Science in Computer Science (2019-2023)*

**Table of Contents**

1. Introduction 1

2. Design Methodology and Software Process Model 1

3. System Overview 2

3.1 Architectural Design 2

4. Design Models 4

5. Data Design 42

5.1 Data Dictionary 46

6. Human Interface Design 50

6.1 Screen Images 50

6.2 Screen Objects and Actions 54

7. Implementation 55

7.1 Algorithms 55

7.2 External APIs/SDKs 55

7.3 User Interface 56

8. Testing and Evaluation 63

8.1 Unit Testing 63

8.2 Functional Testing. 66

8.3 Integration Testing 67

9. Plagiarism Report 68

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

Application Evaluation History

|  |  |
| --- | --- |
| **Comments (by committee)**  **\*Include the ones given at scope time both in doc and presentation** | **Action Taken** |
|  |  |
|  |  |

Supervised by

Mr. Asif Muhammad

Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Introduction

This project aims to provide a platform for small businesses storing, shipping, and selling solution to maintain good service and fast shipping. It’s based on the concept of distributed warehouse networks where any person with space can open a warehouse to accommodate a business inventory. Warehouses will also be required to ship the business inventory to the required customers. Warehouse will be able to sell their own inventory to customers along with other services. Stock Logistic also provide the opportunity for the people to have space and are looking to earn profit from it.

A supplier will be able to source product and will send it to warehouse. Supplier can register with multiple warehouses so that their inventory will be able to wide range of customers. After selecting the warehouse, the supplier can source the product and our system will keep track of all the inventory. Warehouse can have multiple suppliers and it will be the responsibility of warehouse manager to keep track of the supplier inventory. Supplier can convert account to seller account in order to sell directly to customers on stock logistics marketplace. Store logistic will manage all the activities and will charge 10% of revenue and $40 per month. Store logistic will recommend products to buyers. System will analyze data and will recommend warehouse managers to open a new branch in a specific area depending on order traffic.

# Design Methodology and Software Process Model

System will be developed on Enhanced waterfall model. System will be developed completely from requirements to deployment in one go for base requirements like waterfall model. We chose this methodology because base requirements of system are simple and can be easily implemented using waterfall model and system can be updated over time to add new functionality.

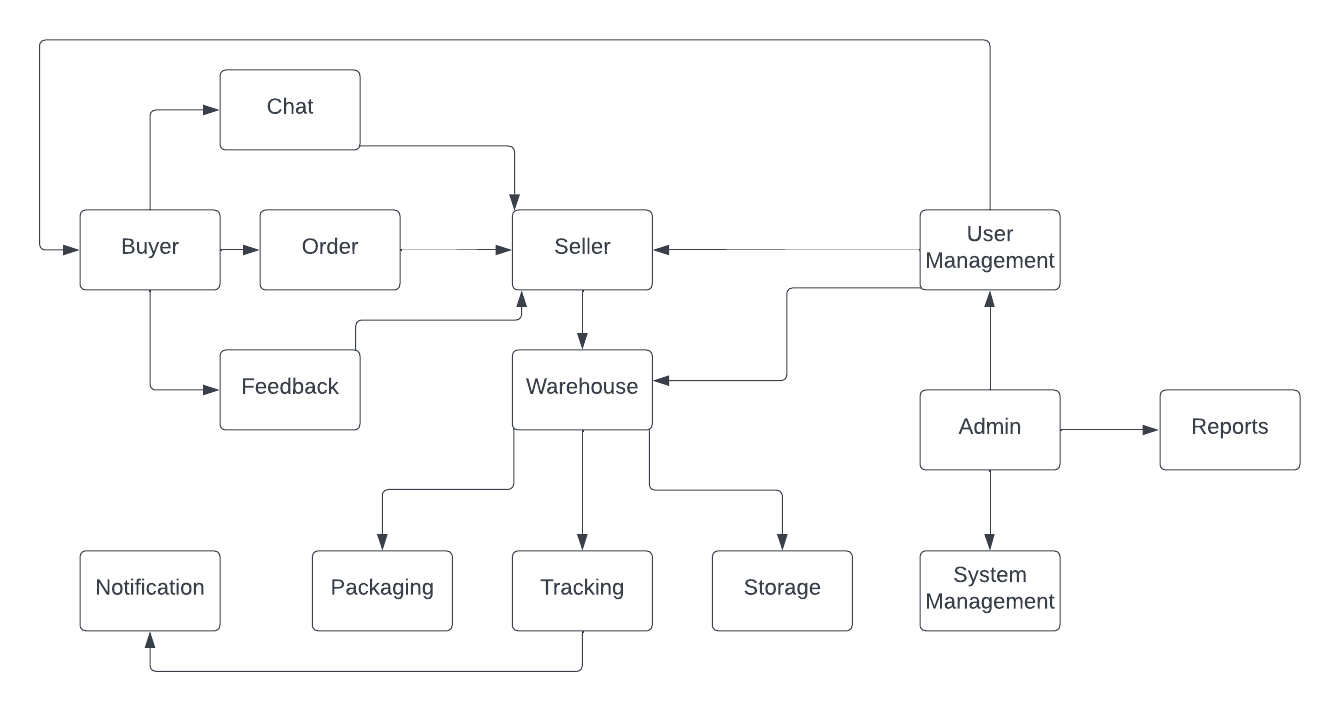
System design will be modulars. The purpose of this approach is to separate each part of functionality into layers and modules so that changes in each module causes least effect in functionality of remaining system which makes it easier to add more functionalities and make changes based on requirements and trends in future.

# System Overview

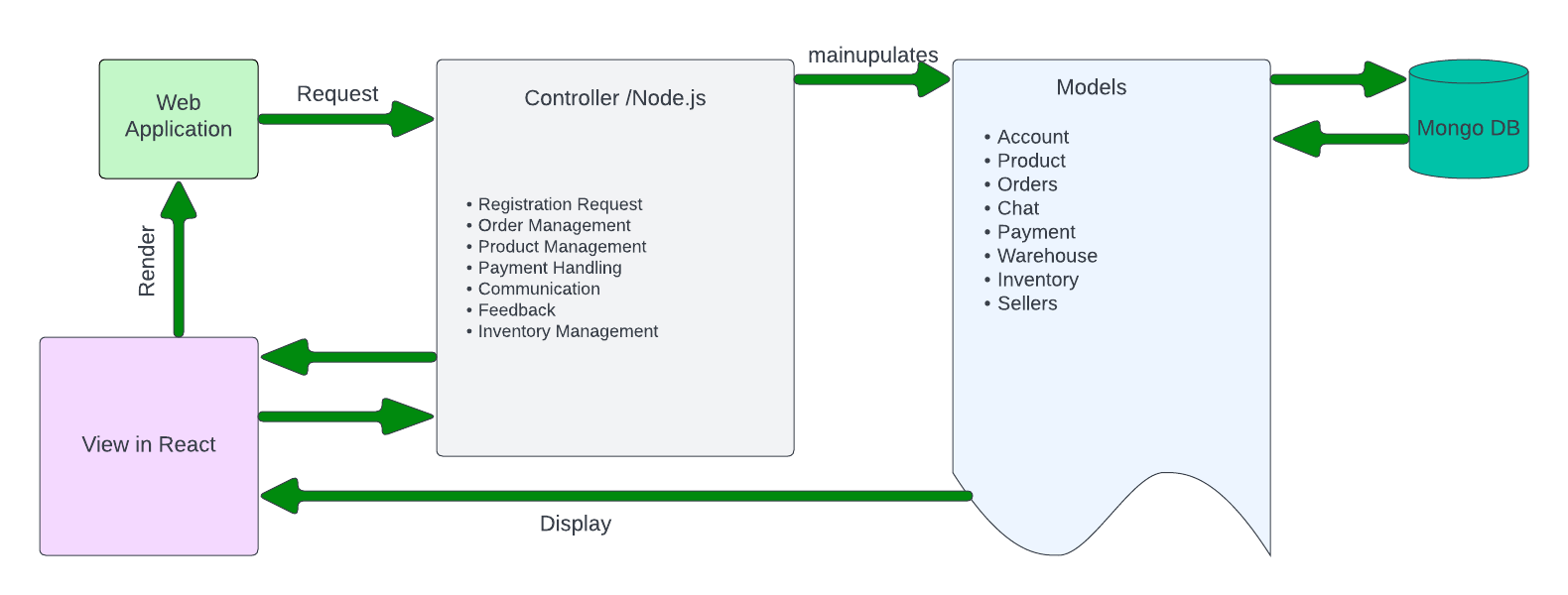
This project aims to provide a platform for small businesses to store, ship and sell their product. It’s based on the concept of distributed warehouse networks where any person with space can open a warehouse to accommodate a business inventory and, in the process, can earn money. A Seller will be able to source product and will send it to warehouse. Seller can register with multiple warehouses so that their inventory will be able to wide range of customers. After selecting the warehouse, the Seller can source the product and our system will keep track of all the inventory. Now, it will be the responsibility of warehouse management to store, ship according to requirements of Seller. Warehouses can manage multiple sellers. Admin of the system will be able to manage buyers, sellers and warehouses registered in the system and will also keep track of the business analytics.

## Architectural Design

For our proposed system, we are using MVC architecture. The reason behind using MVC architecture is that it converts the project into a more organized form. Separate layers of Model, View and Controller make it easy to plan and maintain code and fasten the development process. In our MVC architecture, the model will be MongoDB database and controller will be node server. As this is both mobile and web-based application, we will use React as view for web and React Native for mobile.



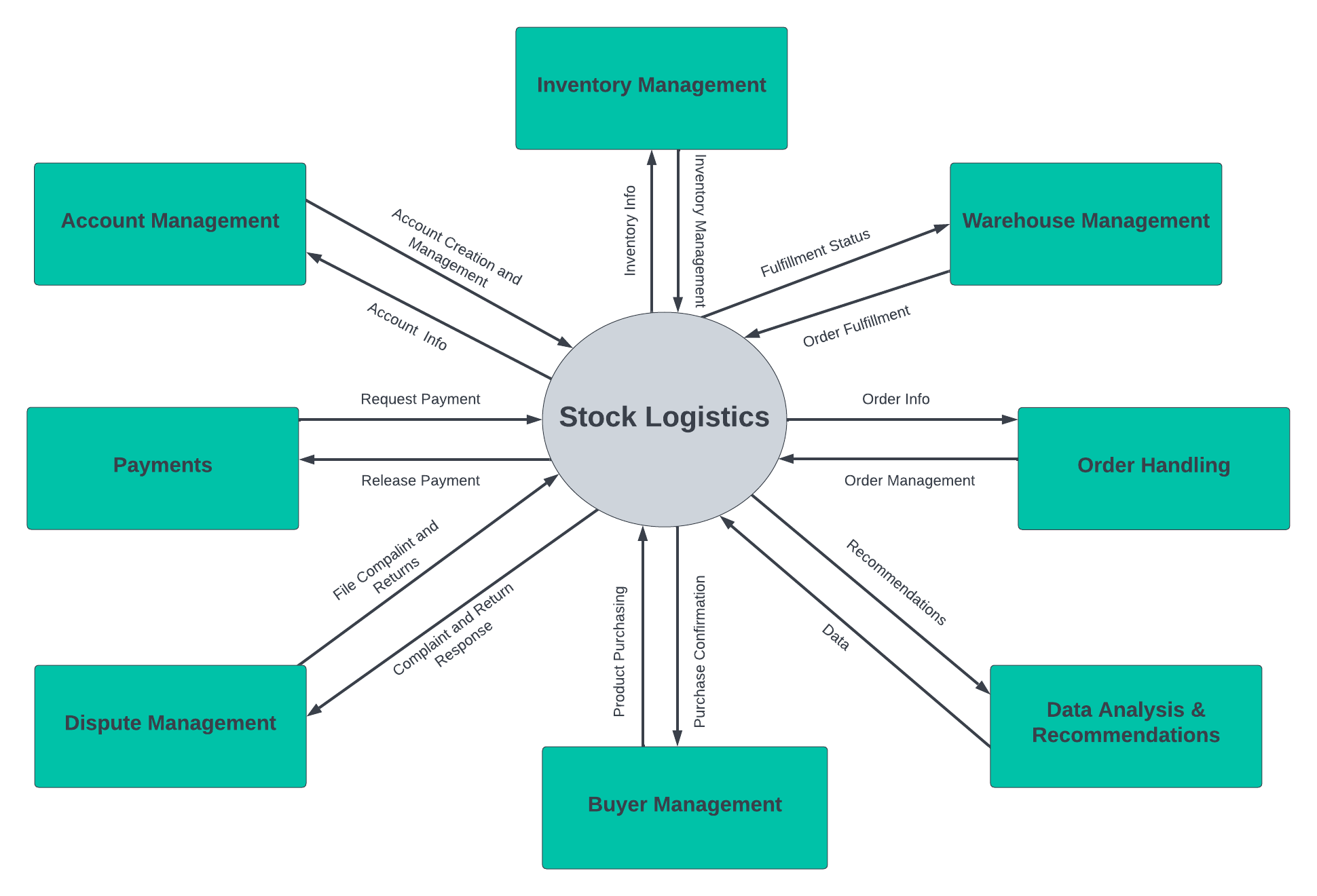
**Figure 1: Box & Line**

**MVC FRAMEWORK**

**Figure 2: MVC**

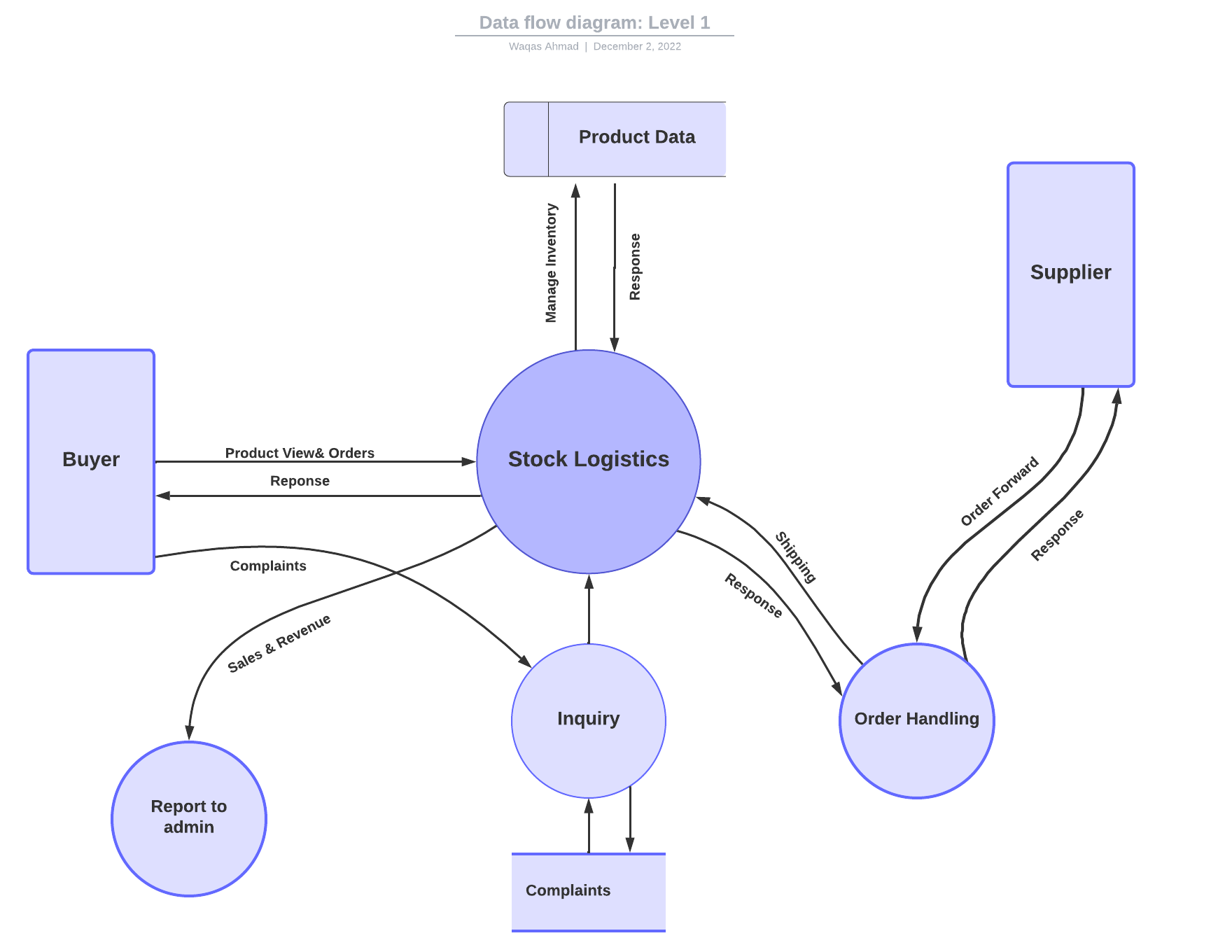
# Design Models

* 1. **Data Flow Diagram**
     1. **Context Diagram**

****Figure 3 is level 0 Data flow diagram for stock logistics.

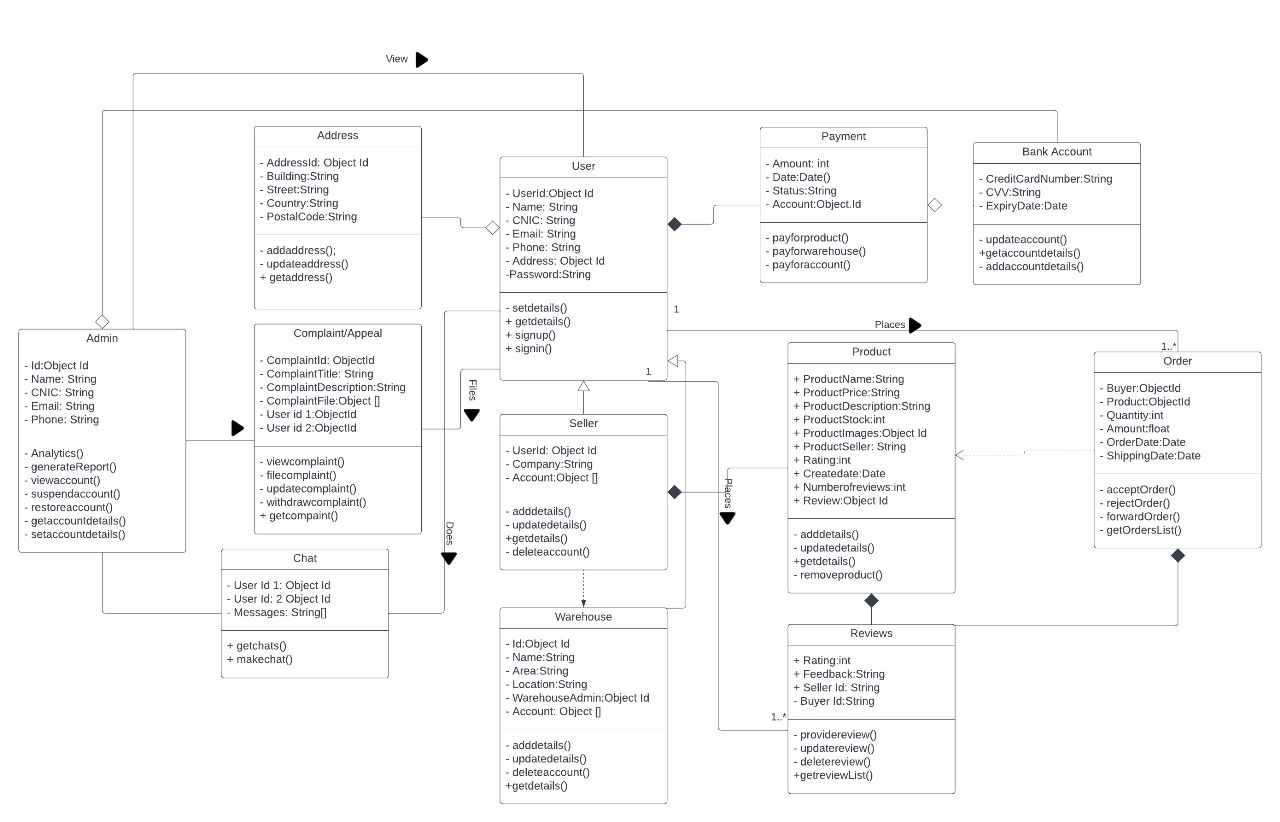
**Figure 3: Context Diagram**

* + 1. **Data Flow Diagram**

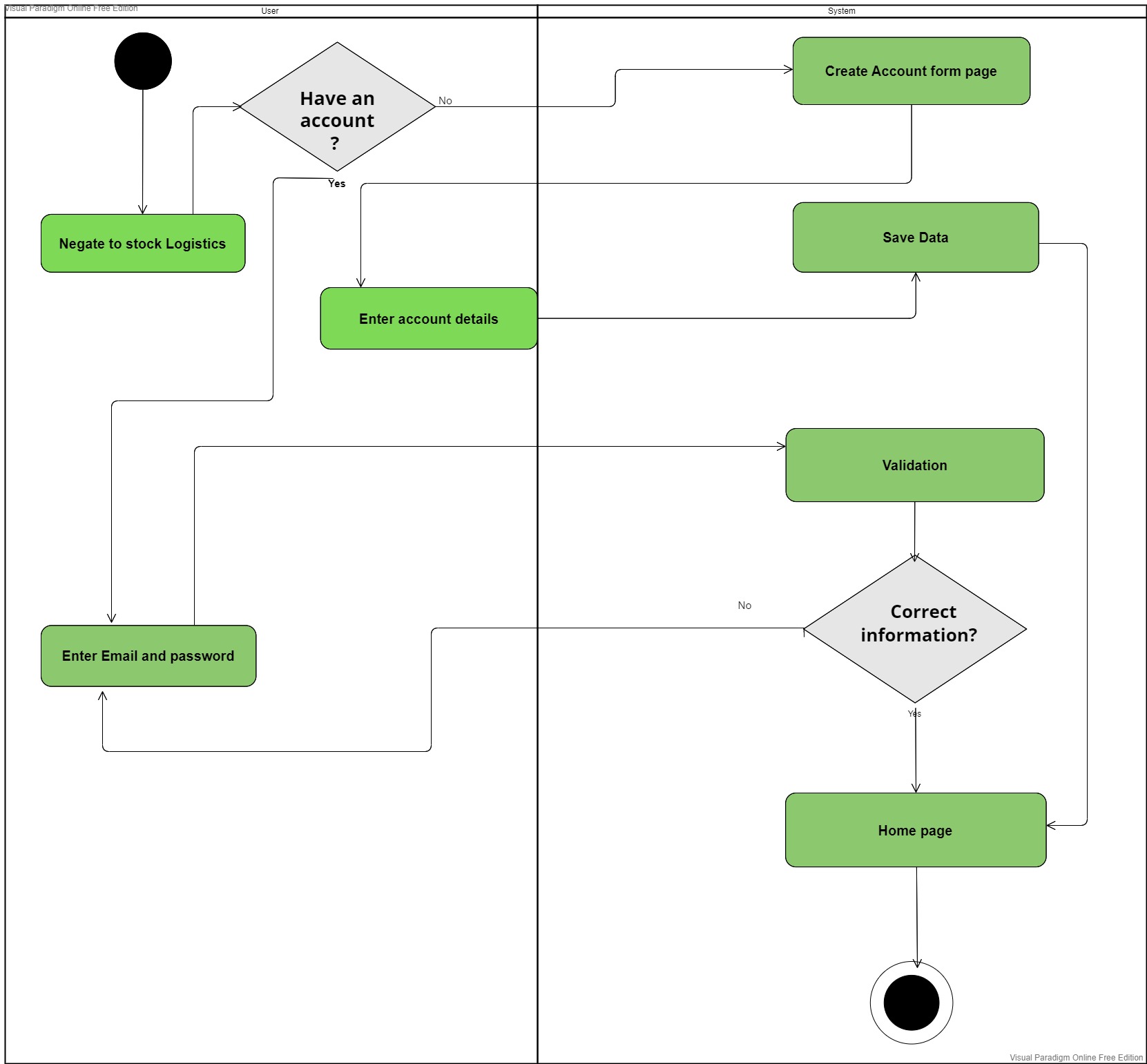
****Figure 4 is level 1 Data flow diagram for stock logistics**.**

**Figure 4: Data Flow Diagram**

* 1. **Class Diagram**

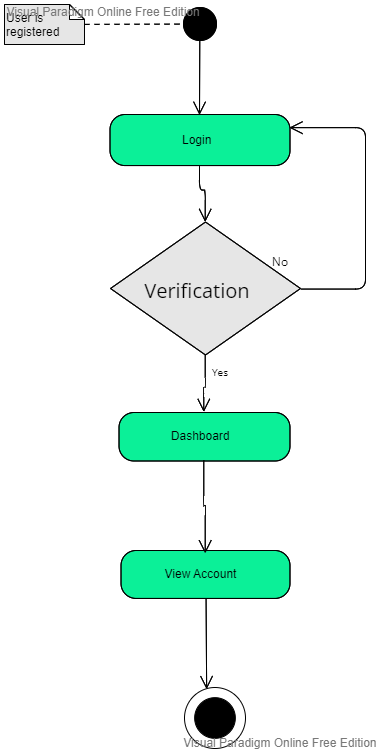


**Figure 5: Class Diagram**

* 1. **Activity Diagram**
     1. **Sign up & Sign in**

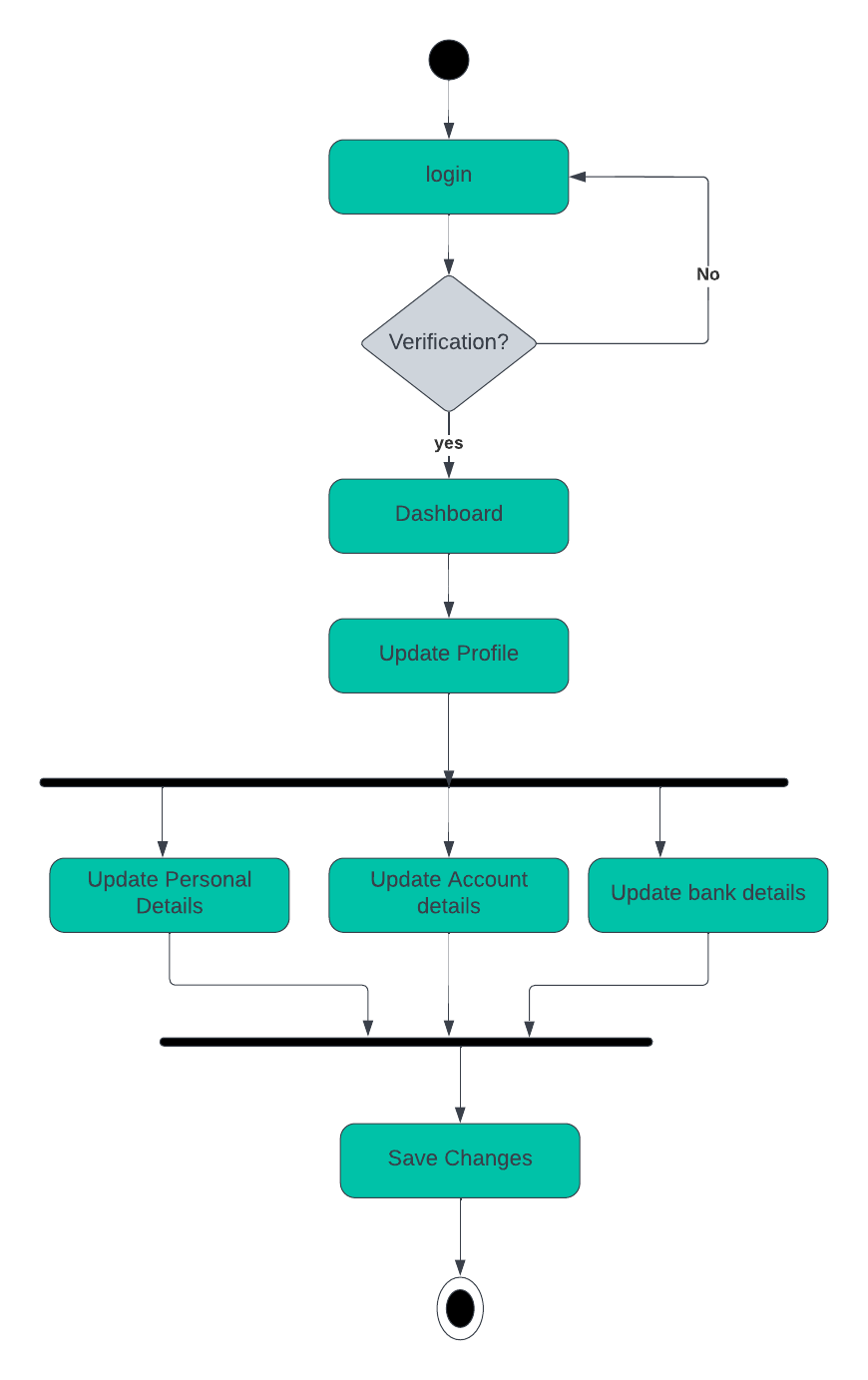
**Figure 6: Signup & Sign in**

* + 1. **View Account**



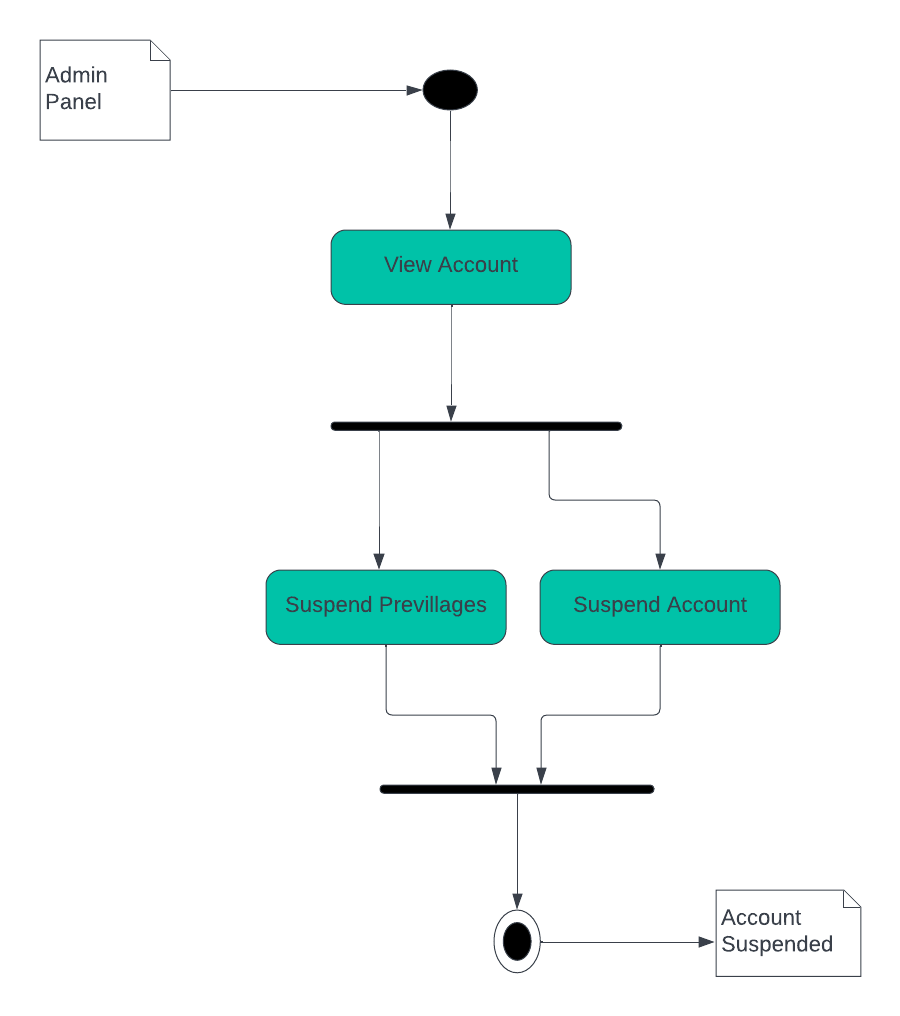
**Figure 7: View Account**

* + 1. **Update Account**

****

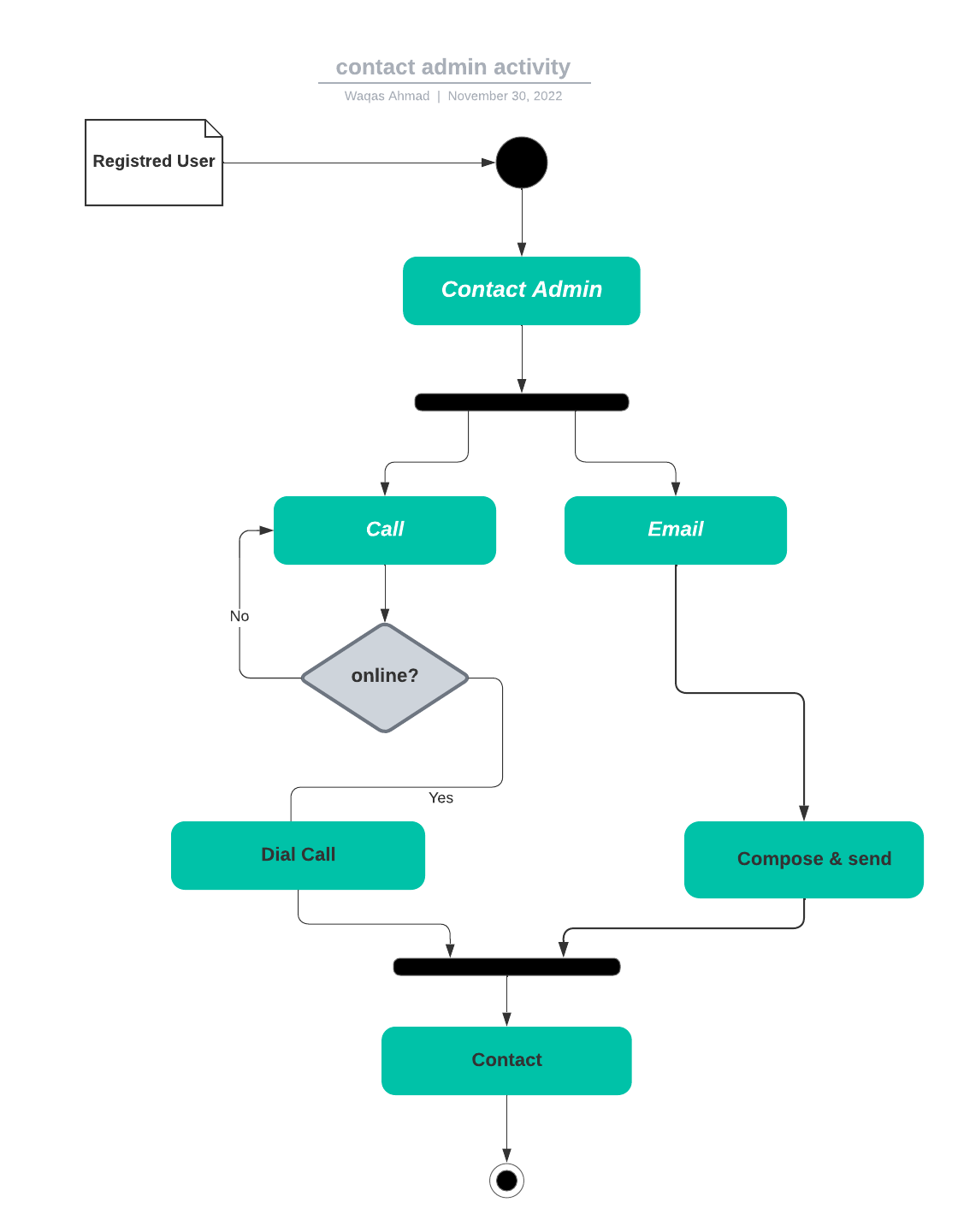
**Figure 8: Update Account Profile**

* + 1. **Suspend Account**

****

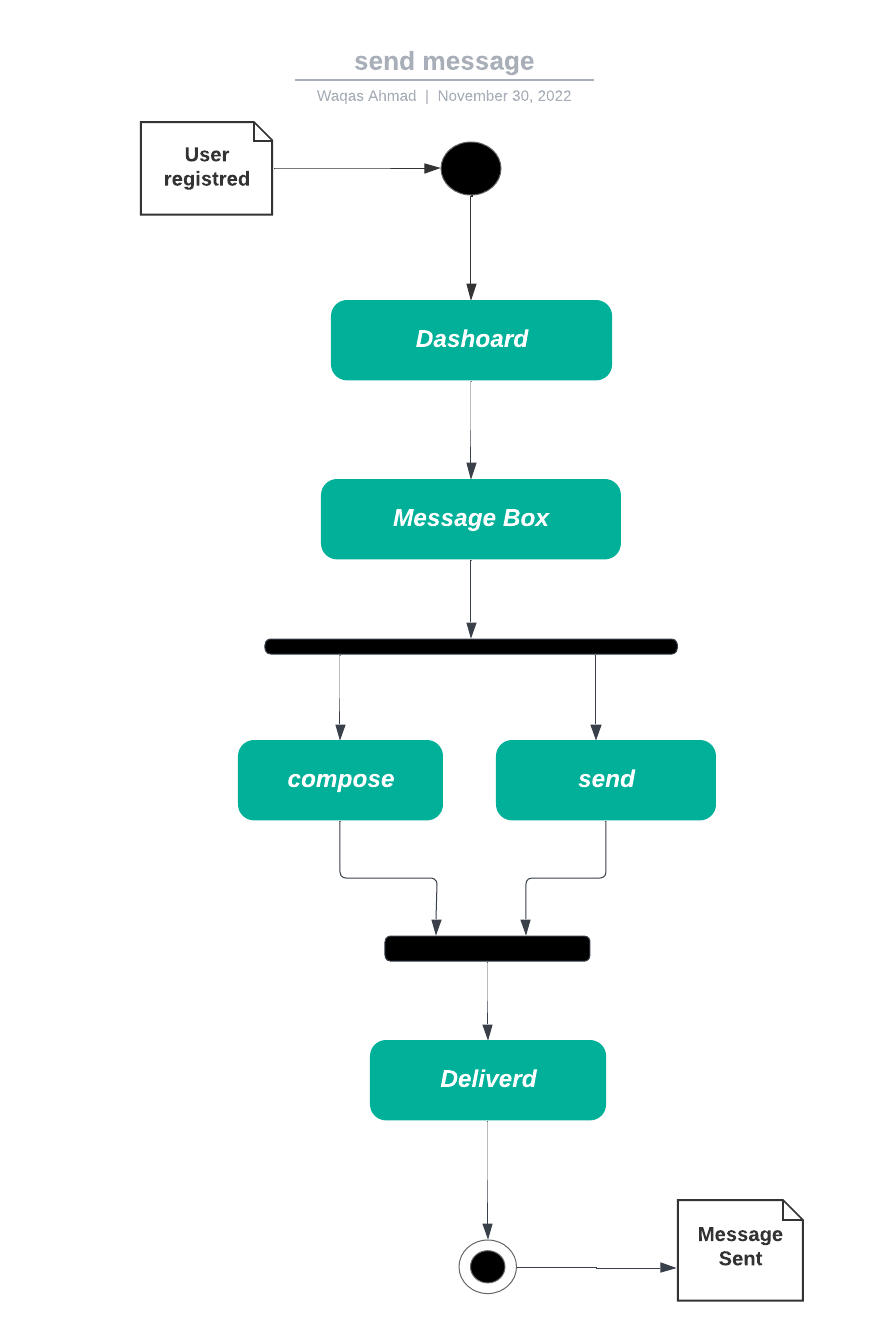
**Figure 9: Suspend Account**

* + 1. **Contact Admin**

****

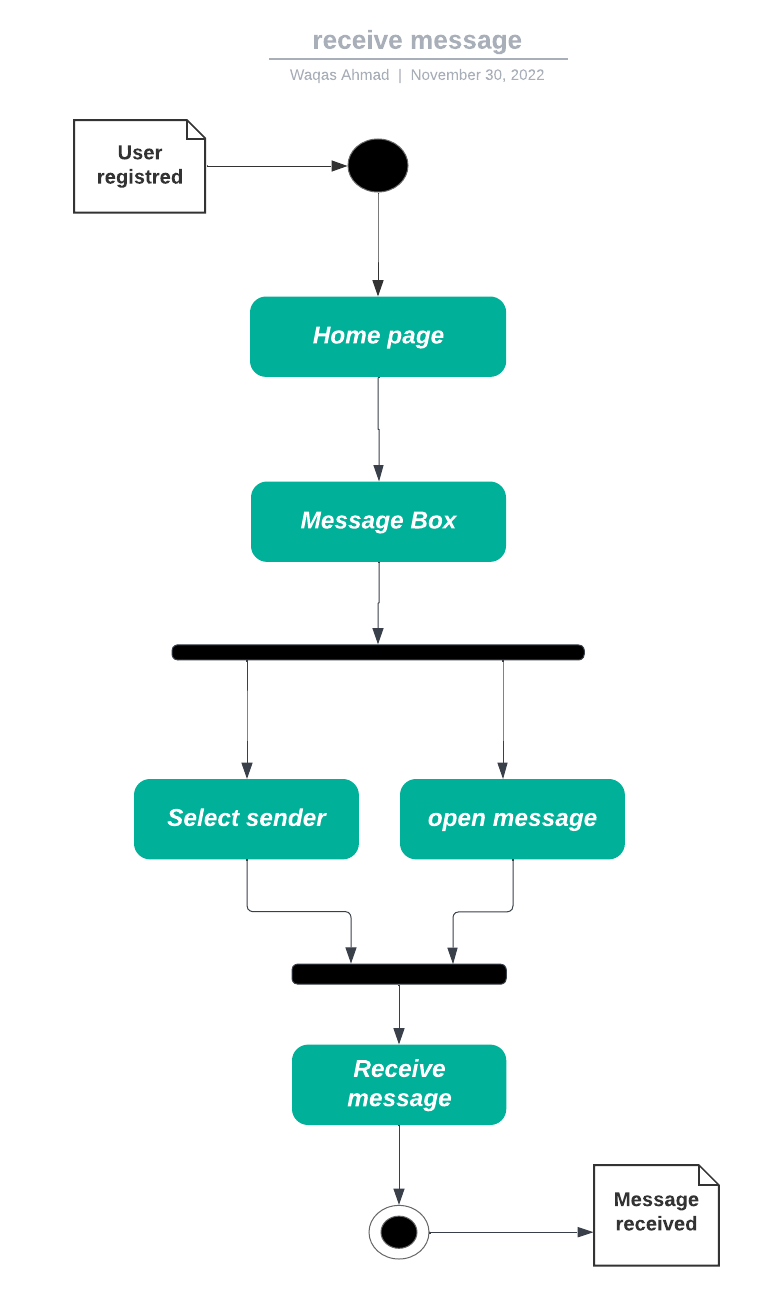
**Figure 10: Contact Admin**

* + 1. **Send Message**

****

**Figure 11: Send Message**

* + 1. **Receive Message**

****

**Figure 12: Receive Message**

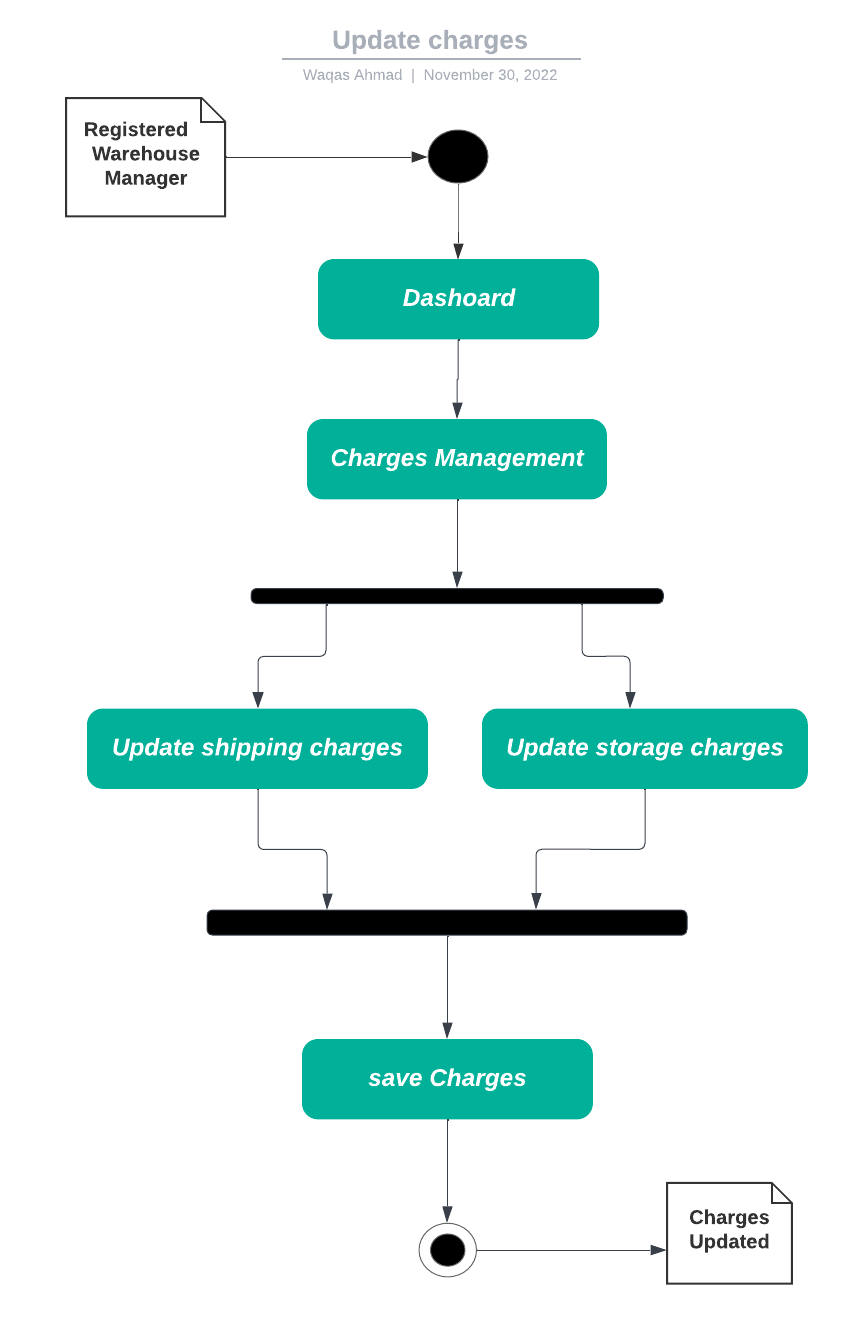
* + 1. **Add Service Charges**

**Diagram

Description automatically generated**

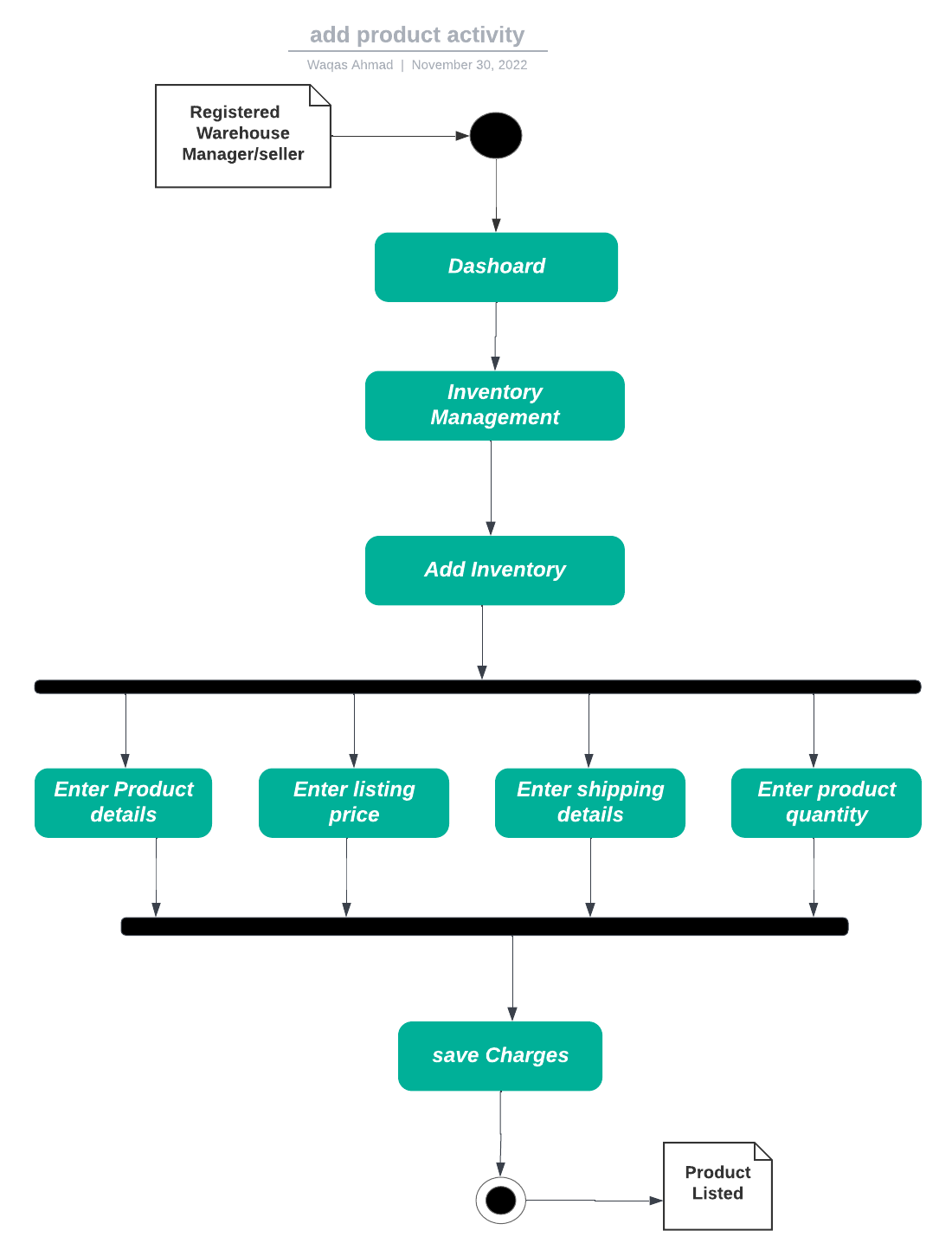
**Figure 13: Add service Charges**

* + 1. **Update Service Charges**

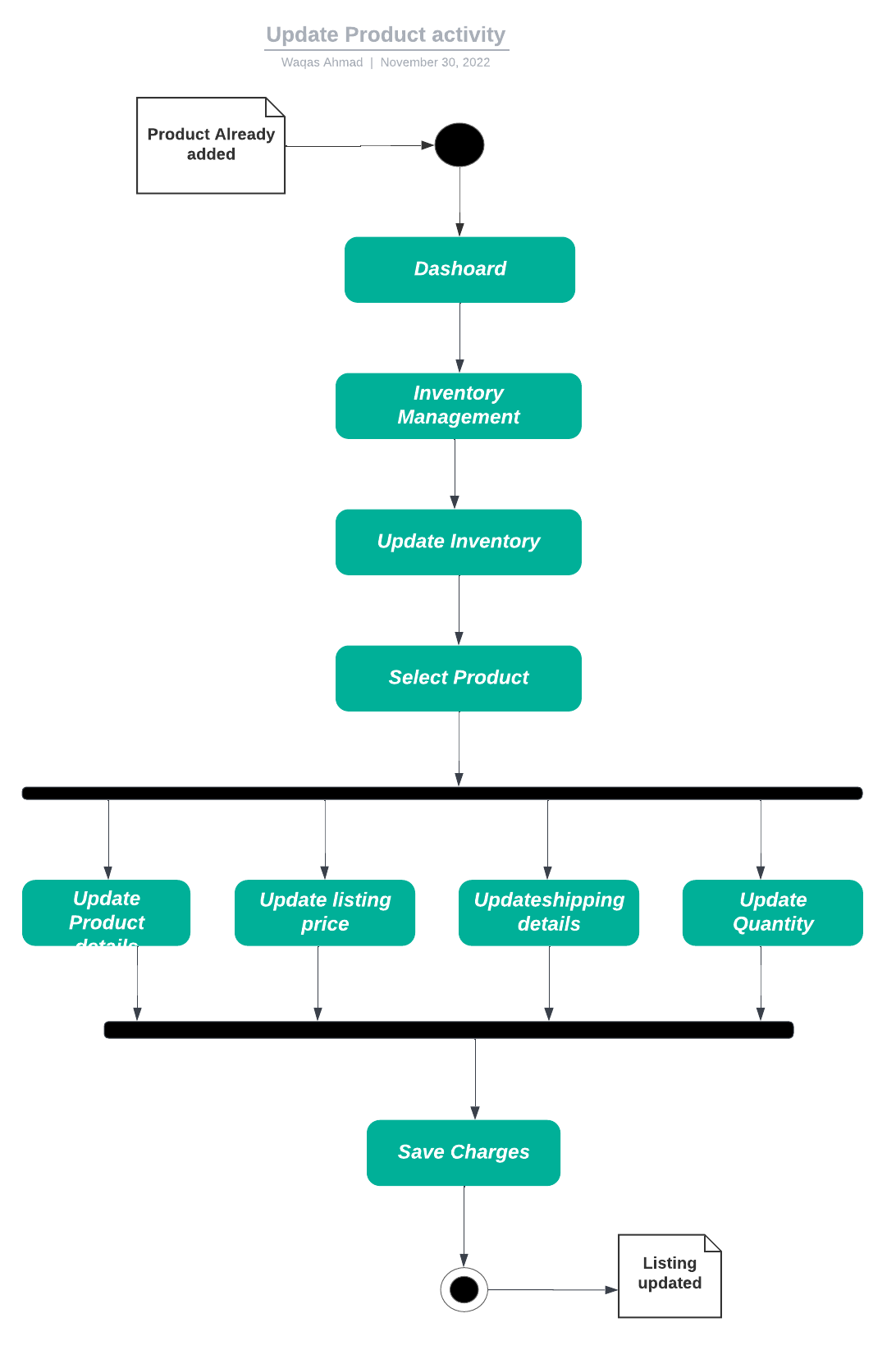


**Figure 14: Update Charges**

* + 1. **Add Product**

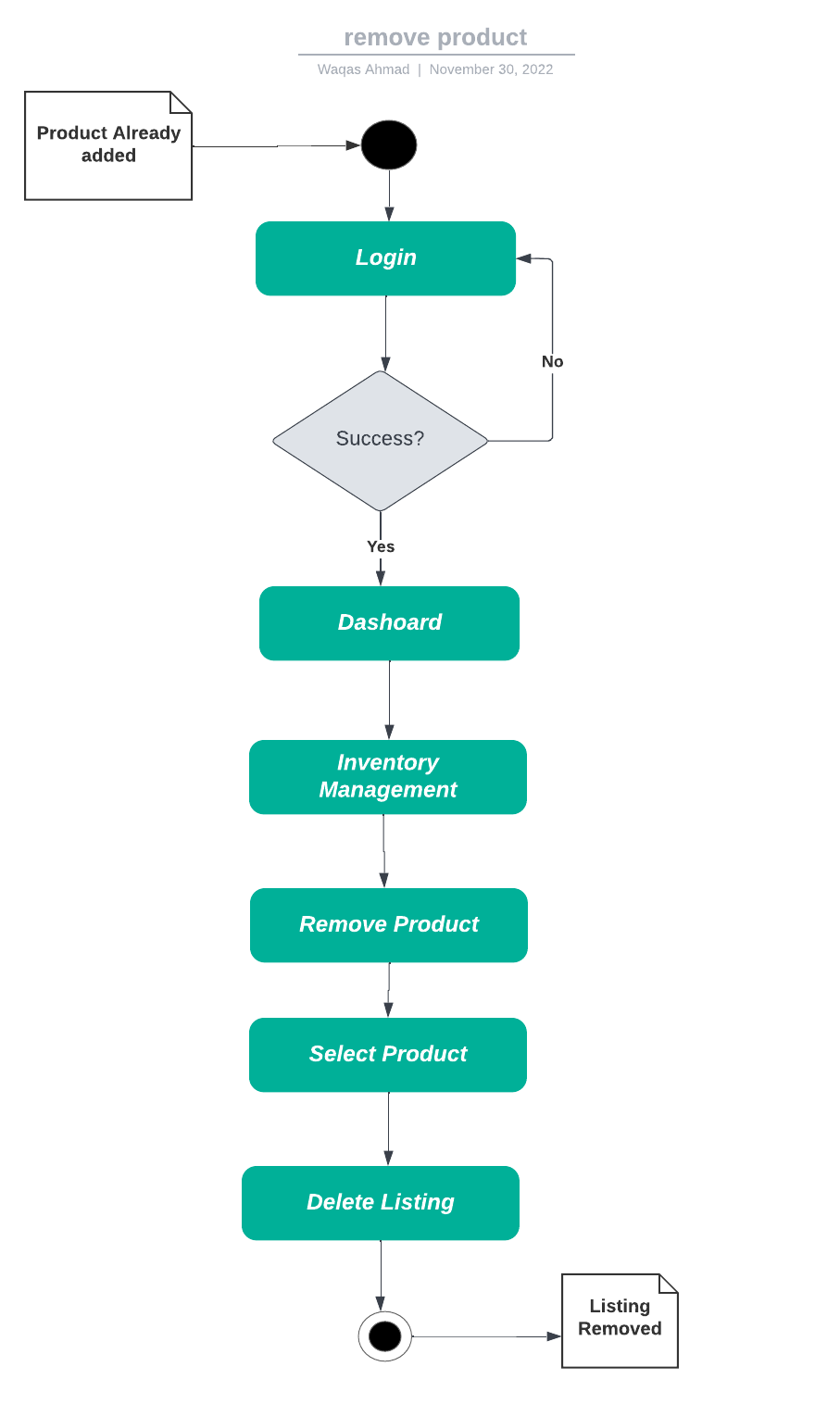
****

**Figure 15: Add Product**

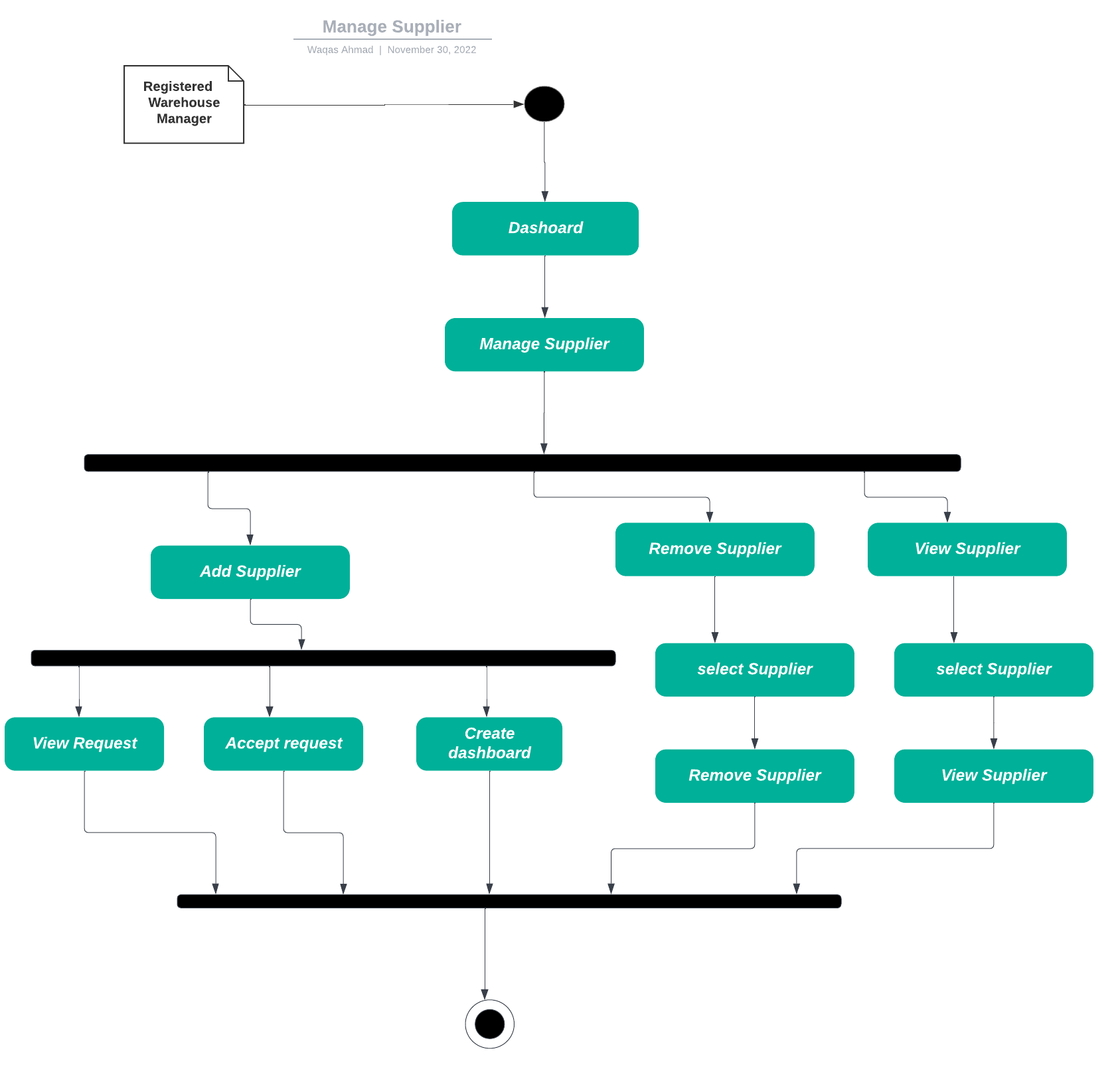
* + 1. **Update Product**

**Figure 16: Update Product**

* + 1. **Remove Product**

****

**Figure 17: Remove Product**

* + 1. **Manage suppliers**

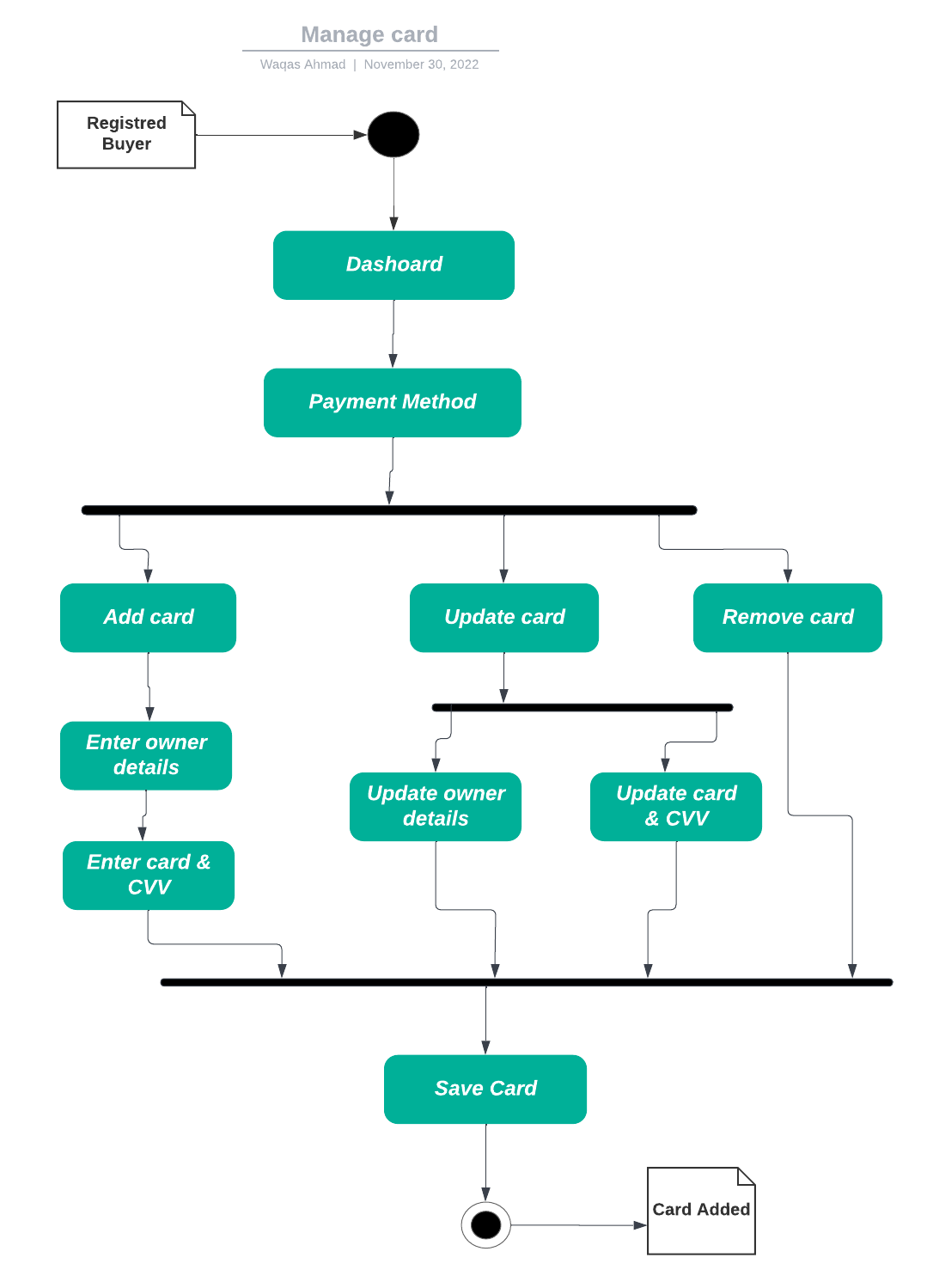
**Figure 18: Manage Supplier**

* + 1. **Place order**

**Diagram

Description automatically generated**

**Figure 19: Place Order**

* + 1. **Manage Card**

**Figure 20: Manage Card**

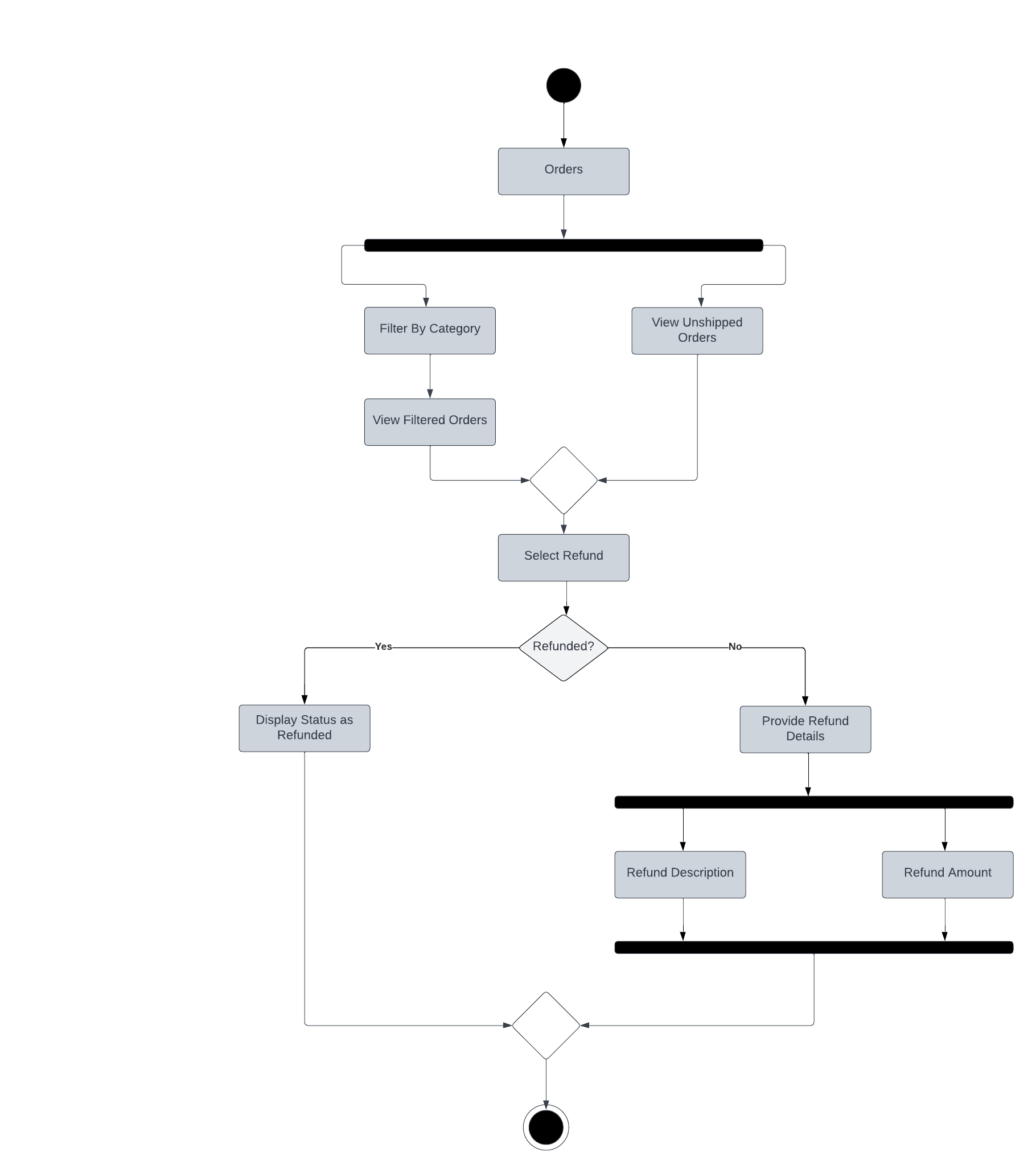
* + 1. **File Appeal**

**Diagram

Description automatically generated**

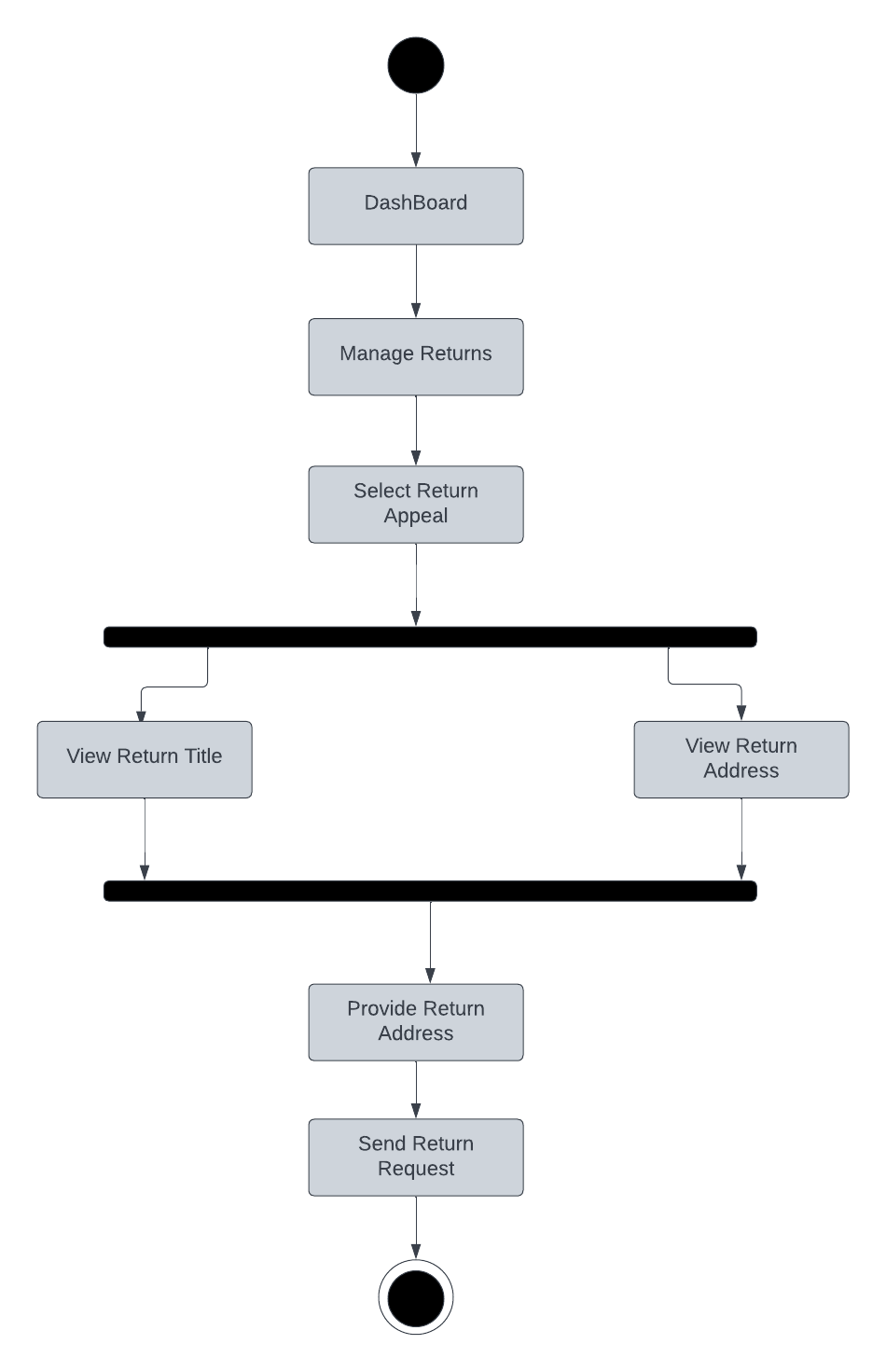
**Figure 21: File Appeal**

* + 1. **Manage Refunds**

****

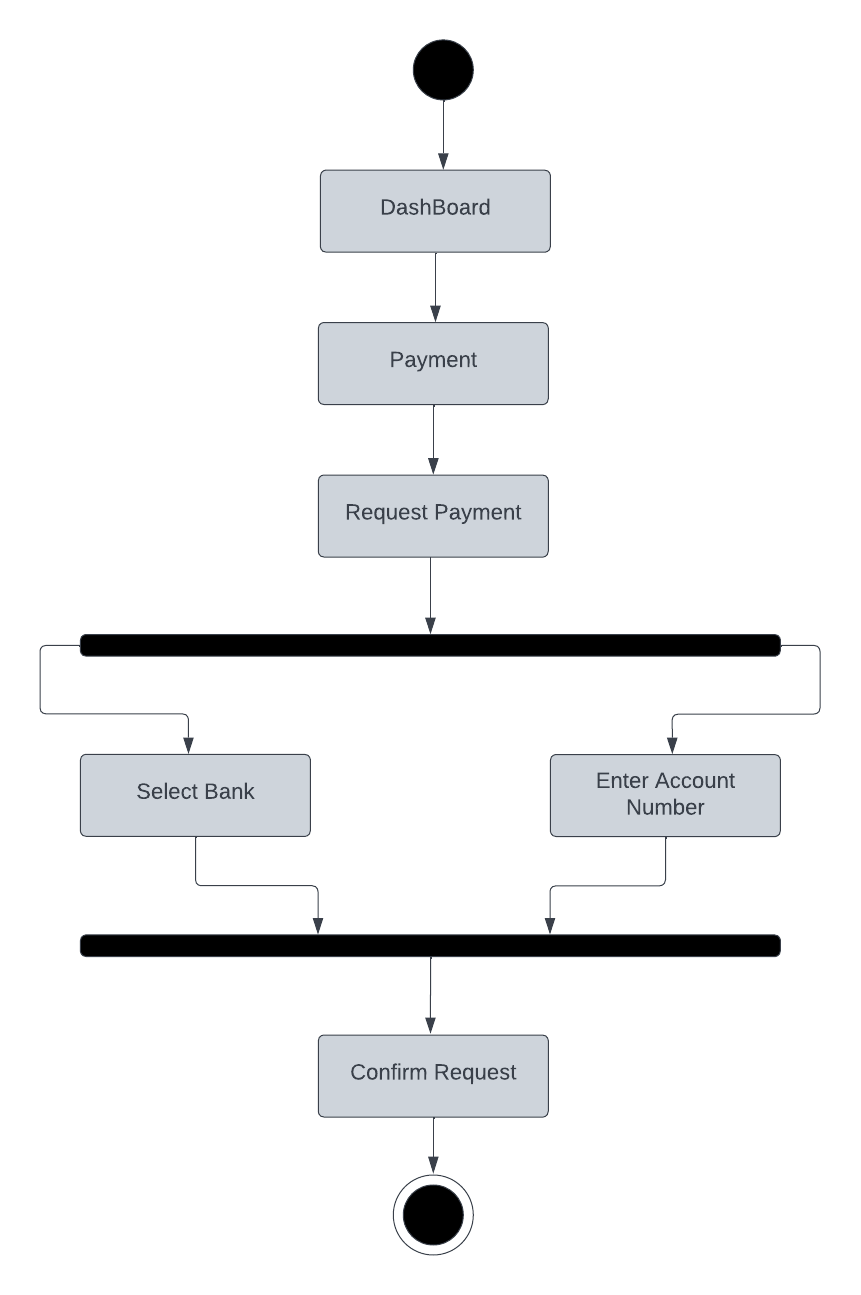
**Figure 22: Manage Refunds**

* + 1. **Manage Returns**

****

**Figure 23: Manage Returns**

* + 1. **Request Payment**

****

**Figure 24: Request Payment**

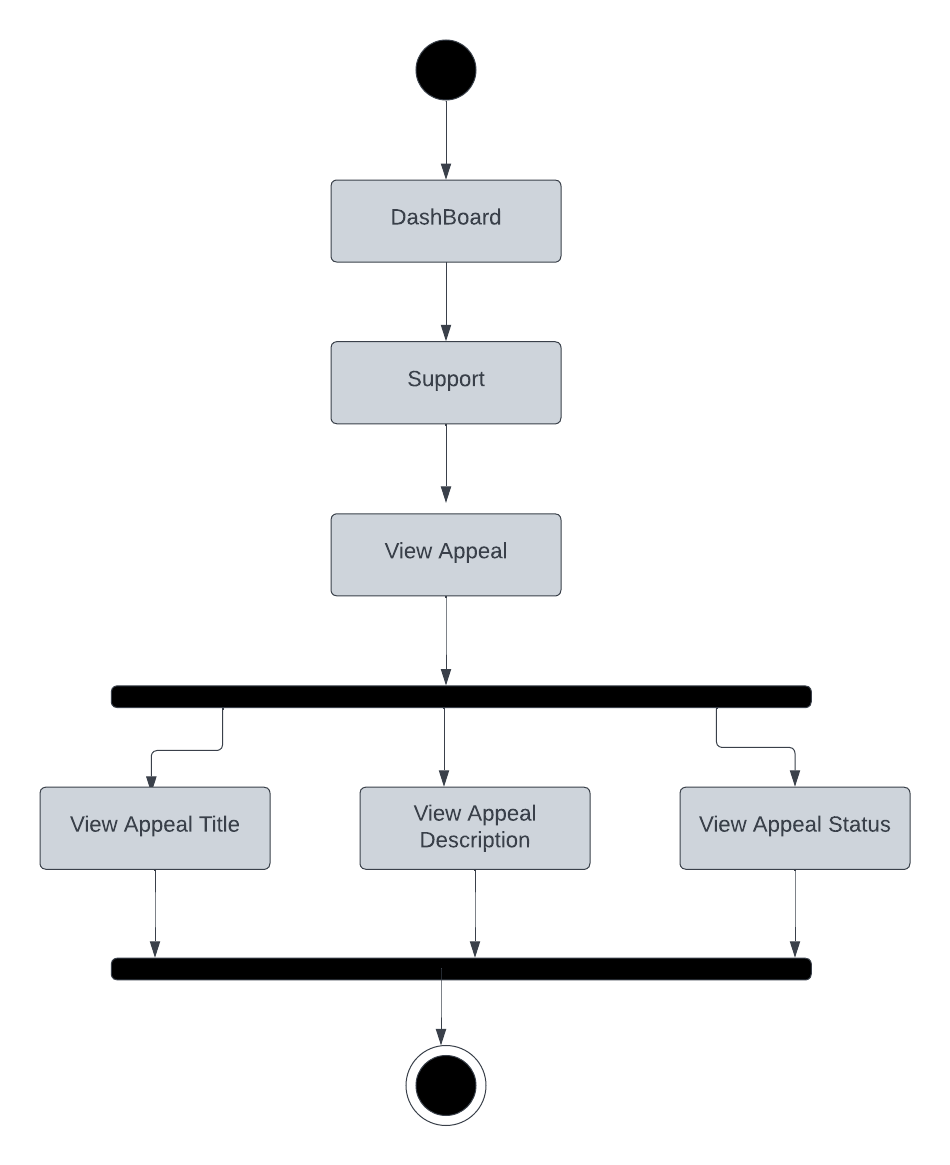
* + 1. **View Order**

**Diagram

Description automatically generated**

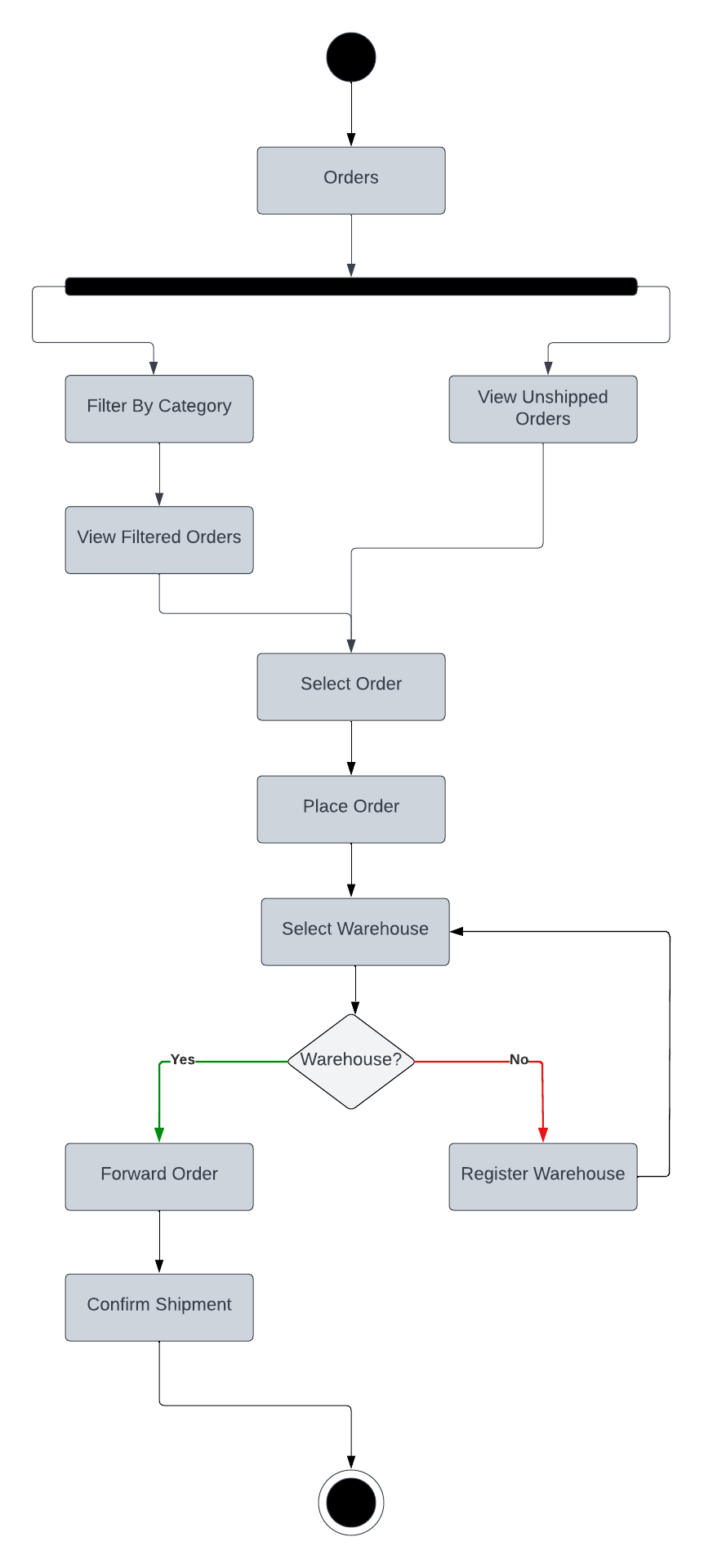
**Figure 25: View Order**

* + 1. **View Appeal**

****

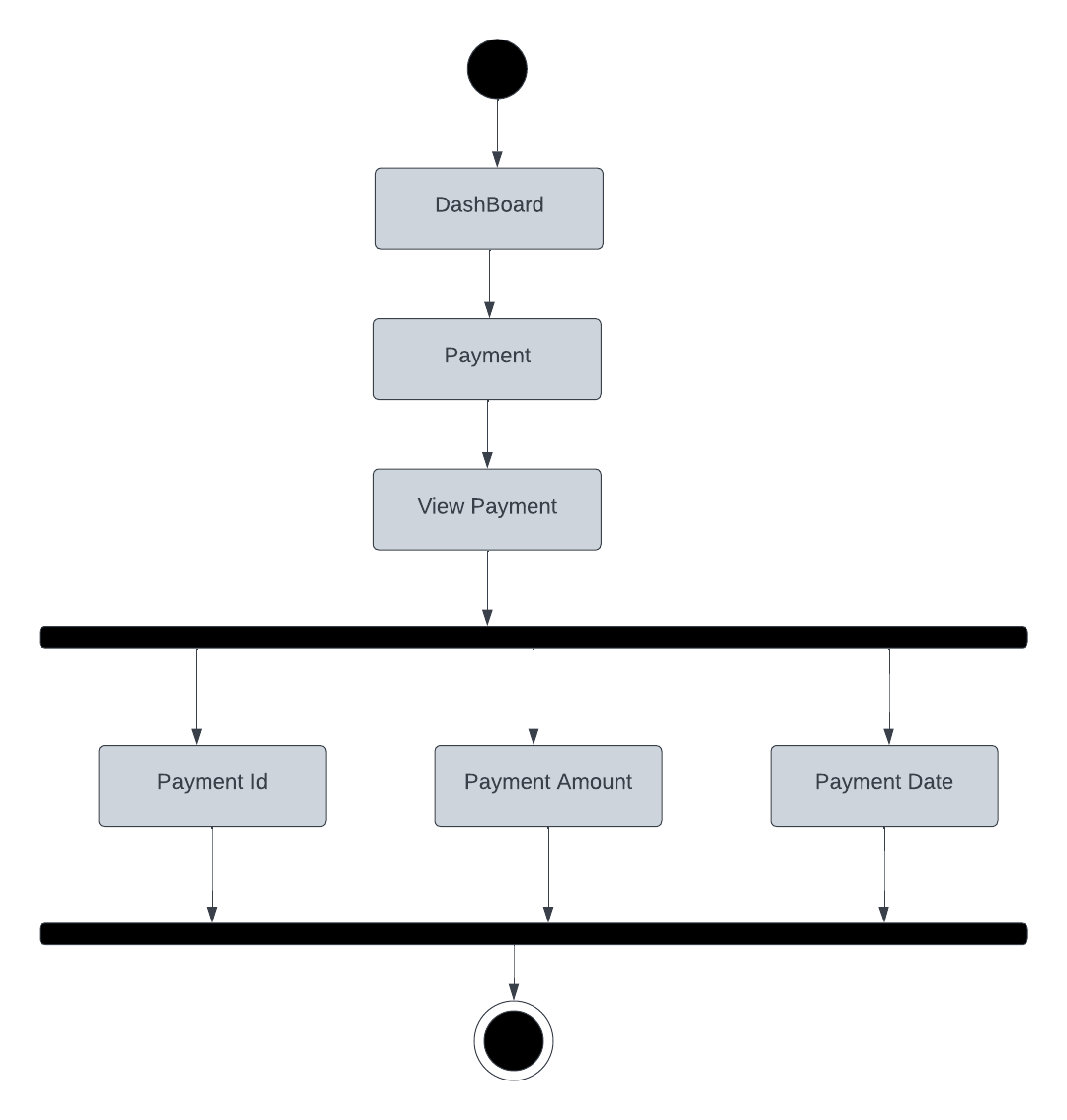
**Figure 26: View Appeal**

* + 1. **Forward Order Details**

****

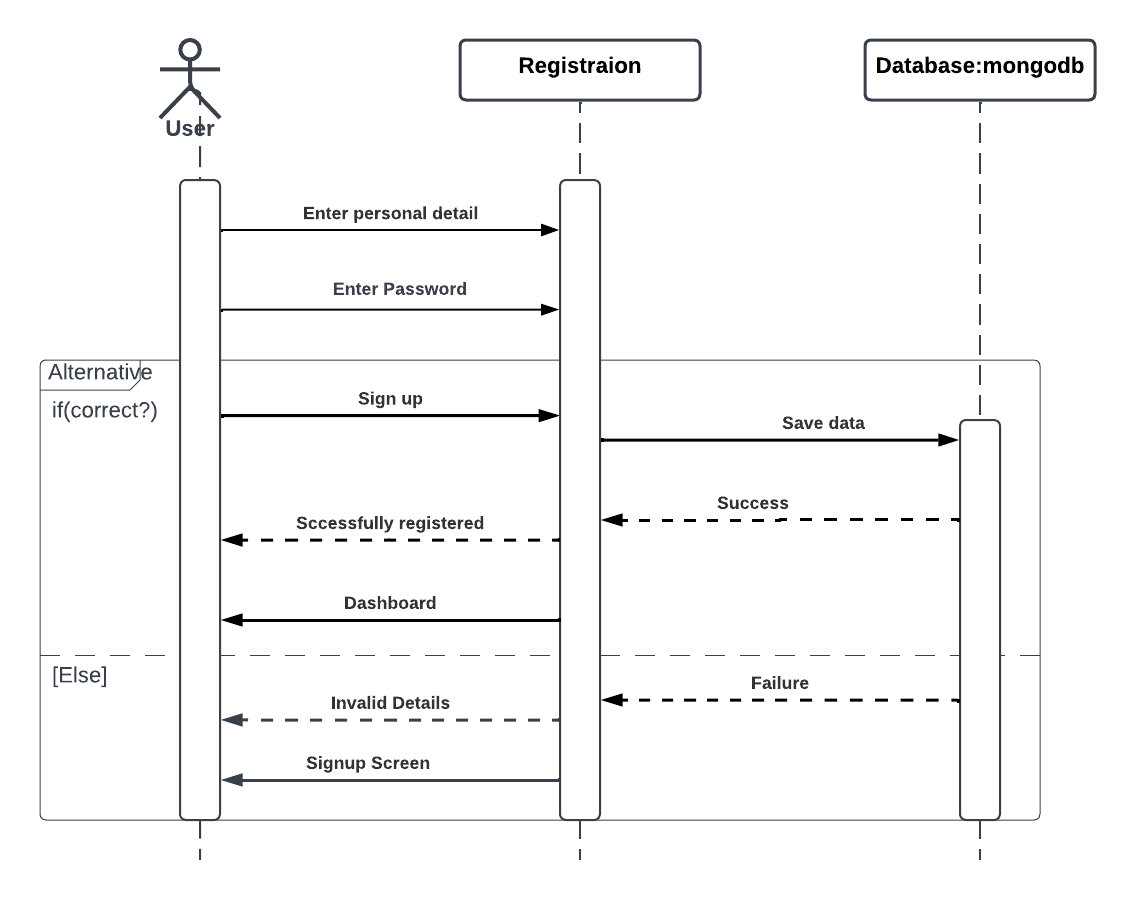
**Figure 27: Forward Order Details**

* + 1. **View Payments**

****

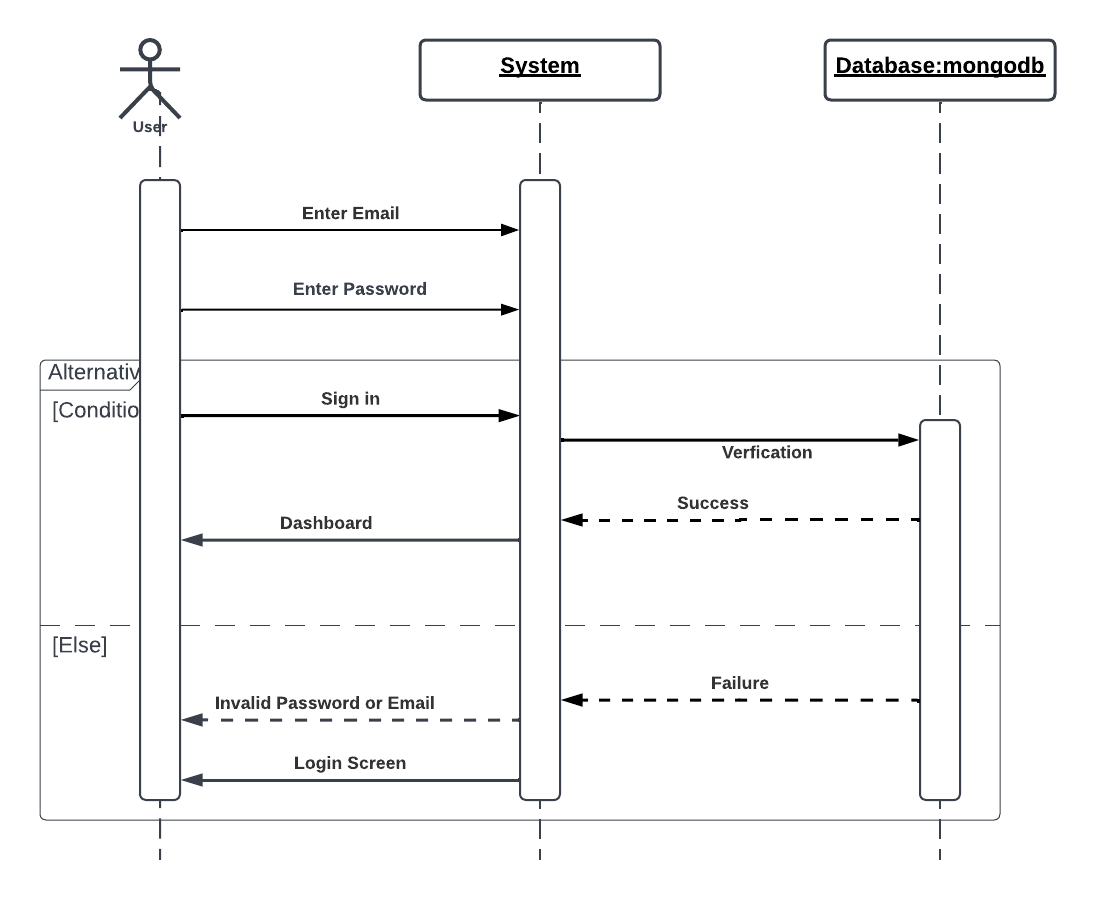
**Figure 28: View Payments**

* 1. **Sequence Diagram** 
     1. **Sign up**

****

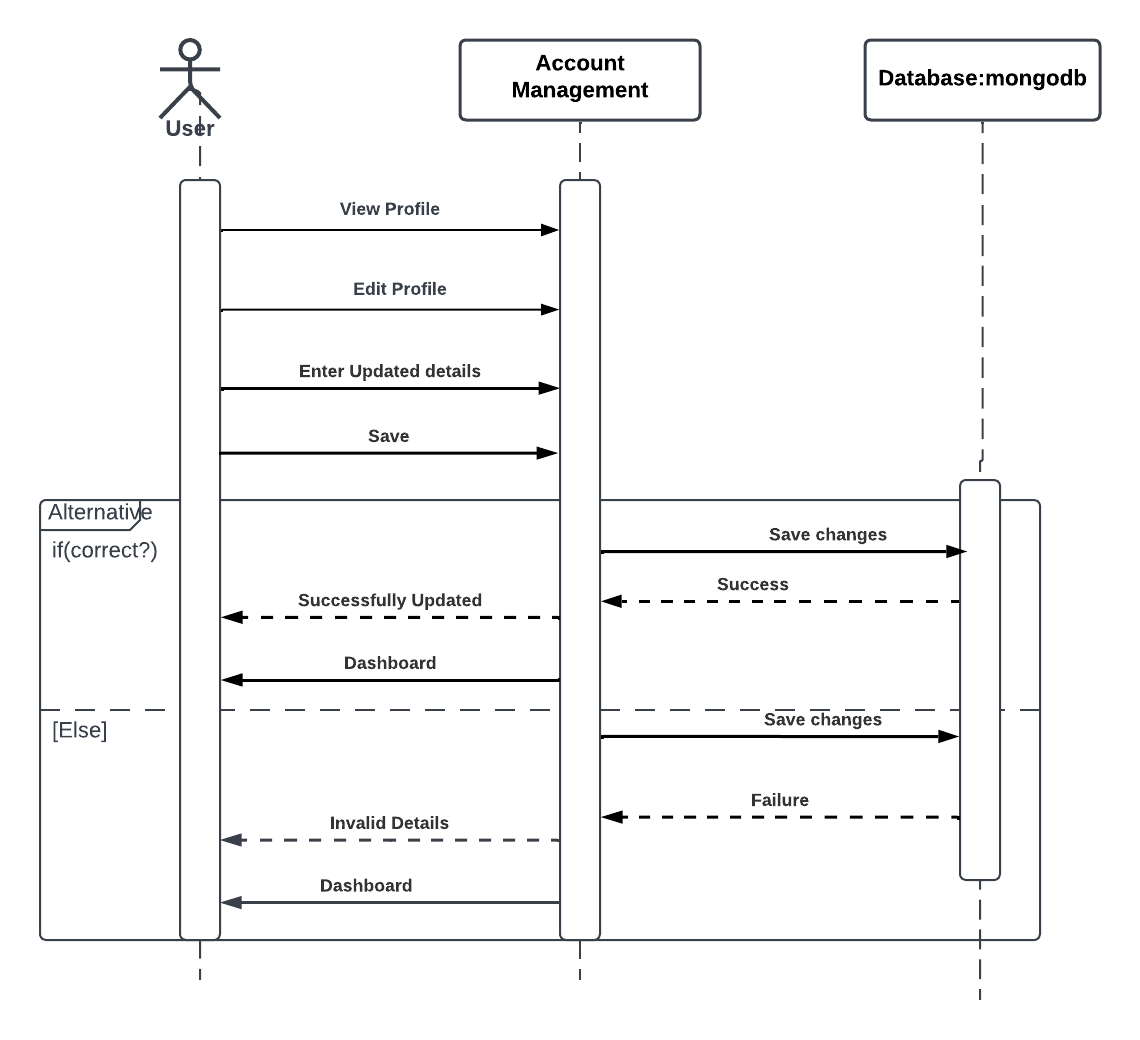
**Figure 29: Sign up**

* + 1. **Sign in**

****

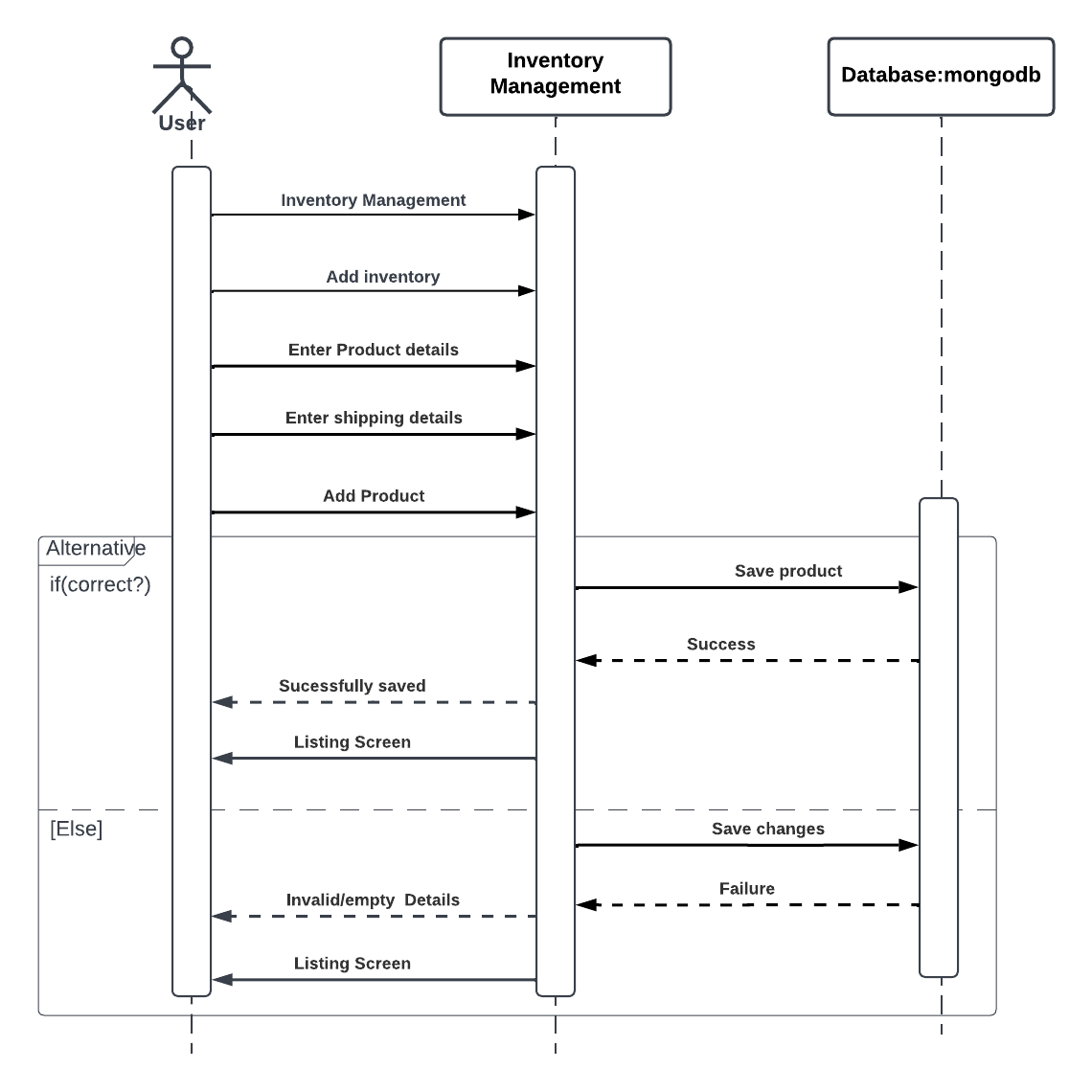
**Figure 30: Sign in**

* + 1. **Update Profile**

****

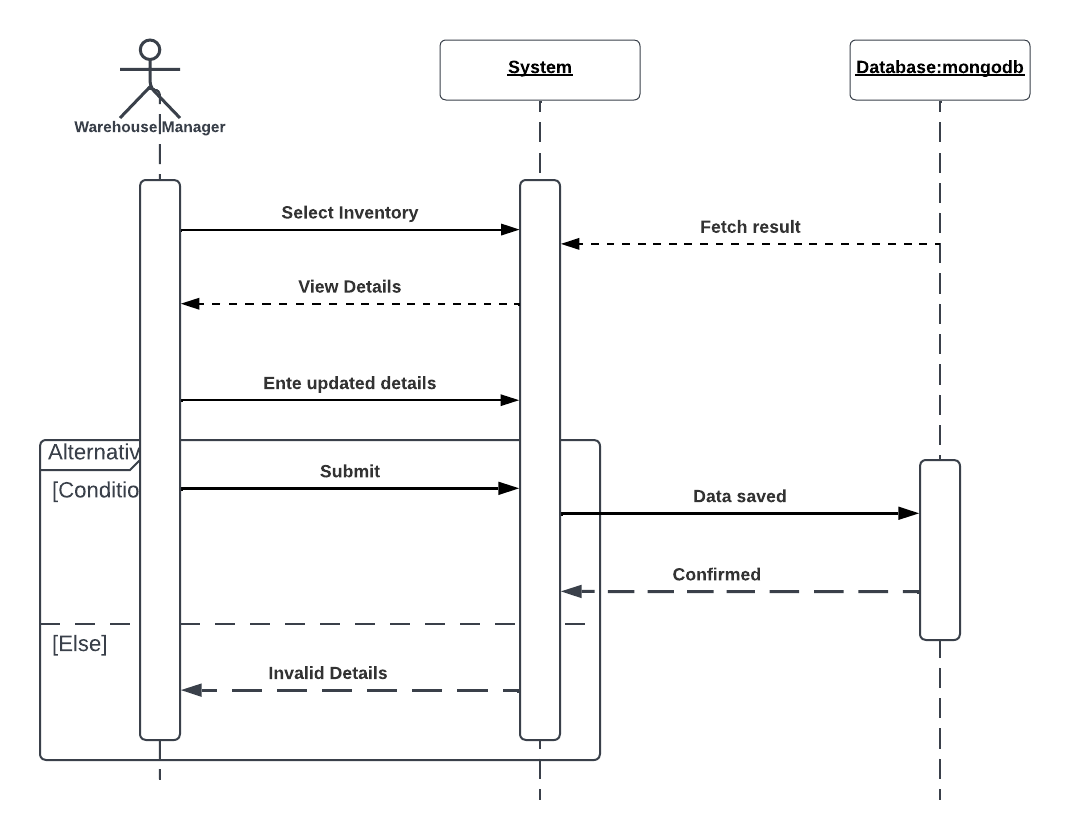
**Figure 31: Update Profile**

* + 1. **Add Inventory**

****

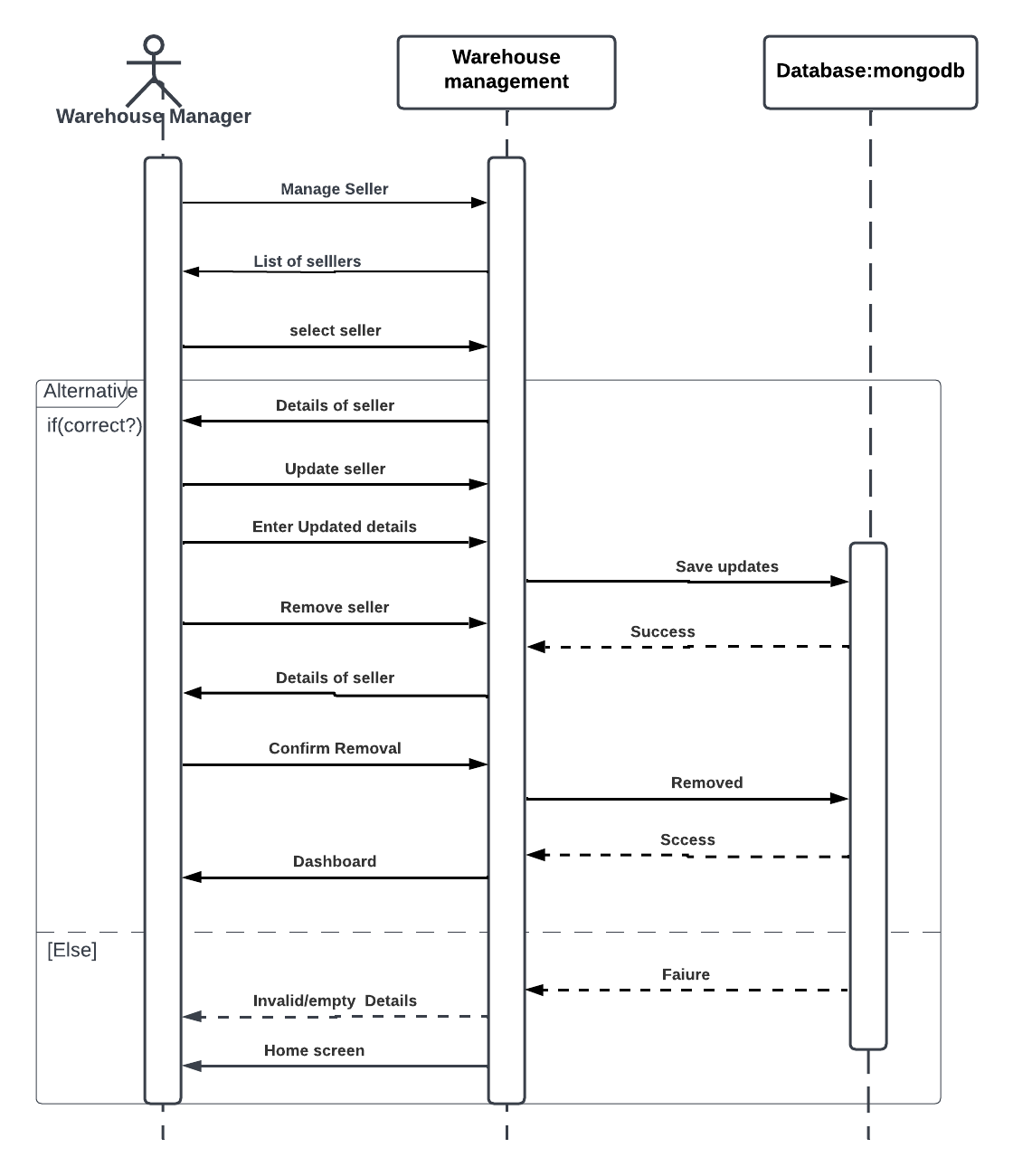
**Figure 32: Add Inventory**

* + 1. **Update Inventory**

****

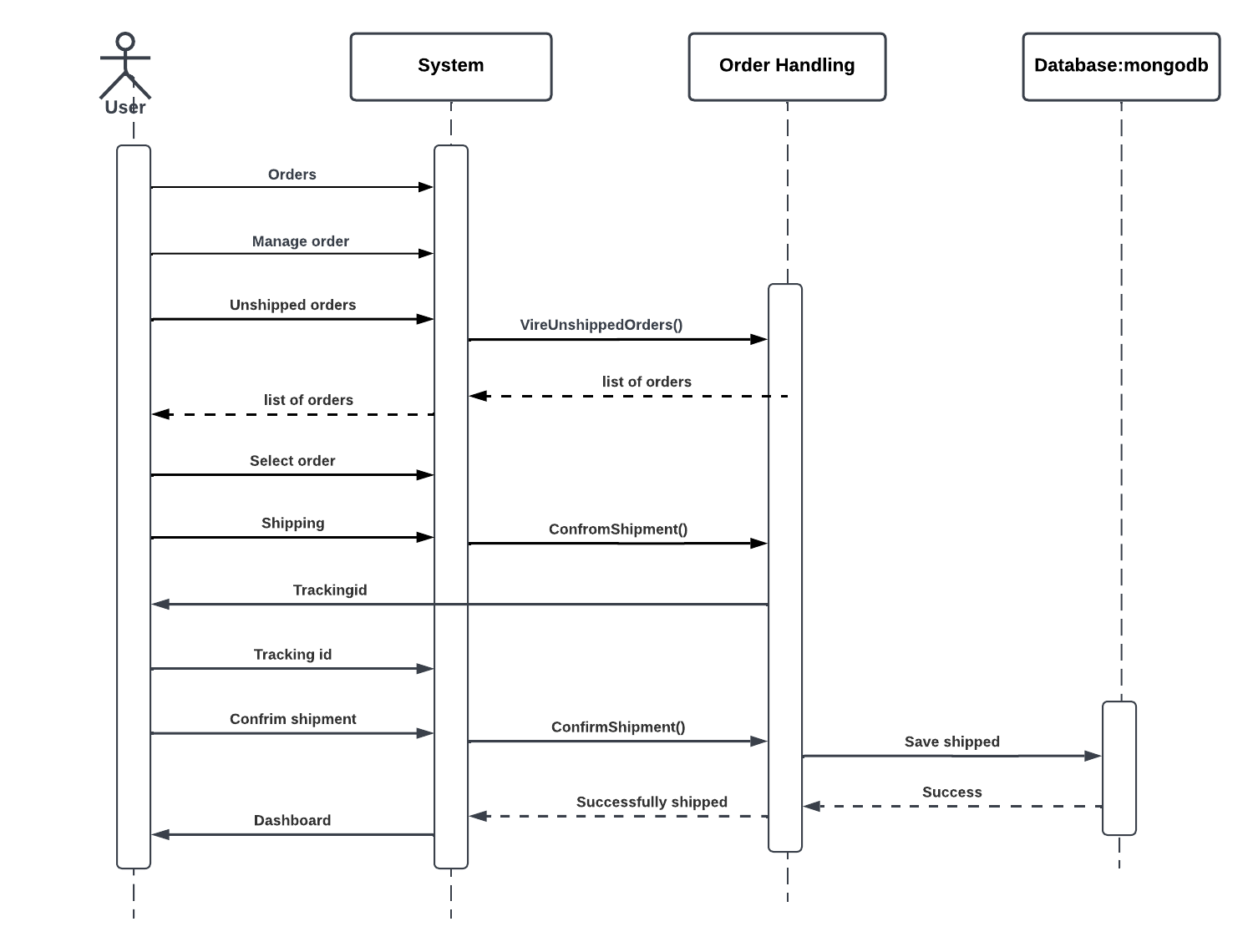
**Figure 33: Update Inventory**

* + 1. **Manage seller**

****

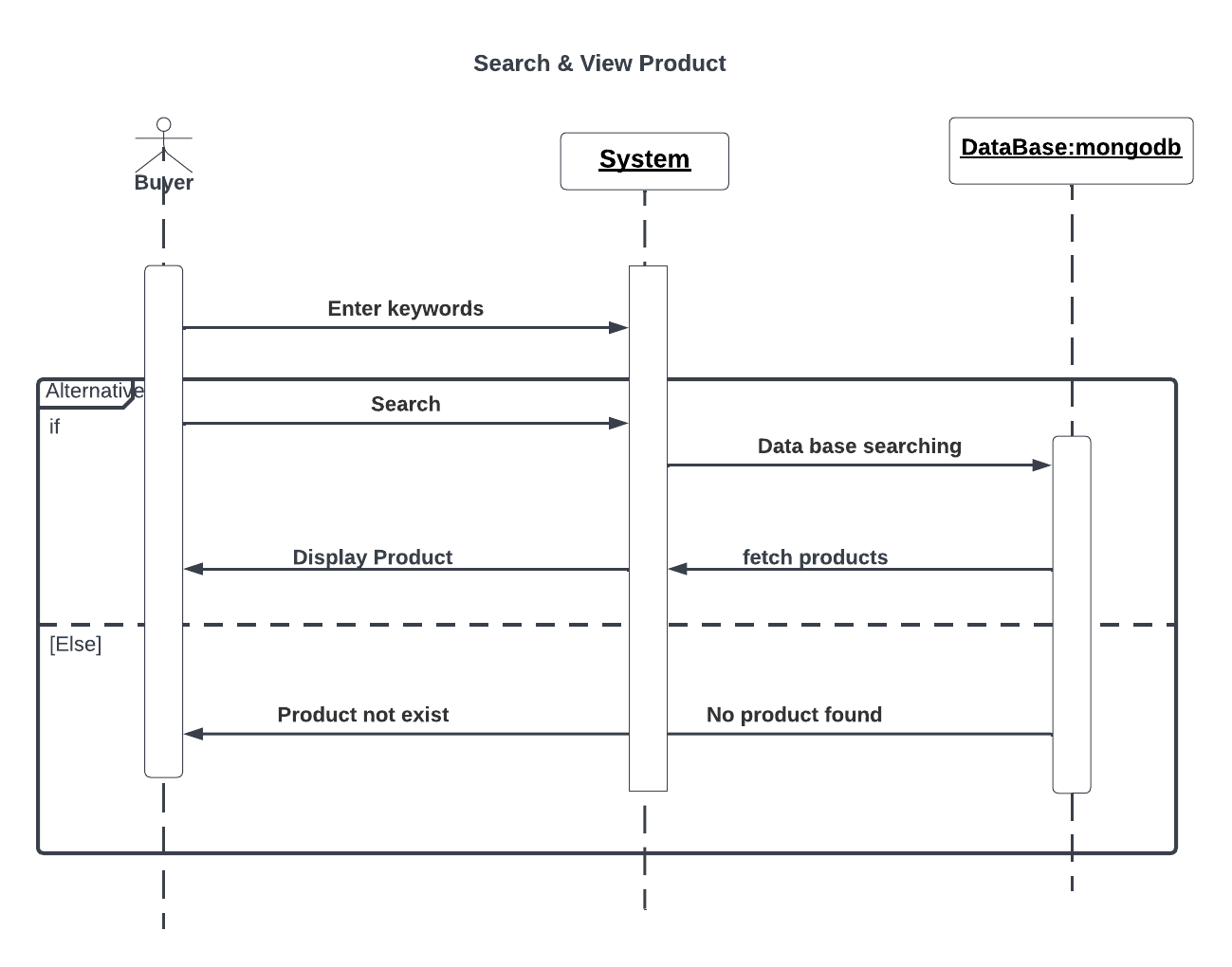
**Figure 34: Manage seller**

* + 1. **Confirm shipment**

****

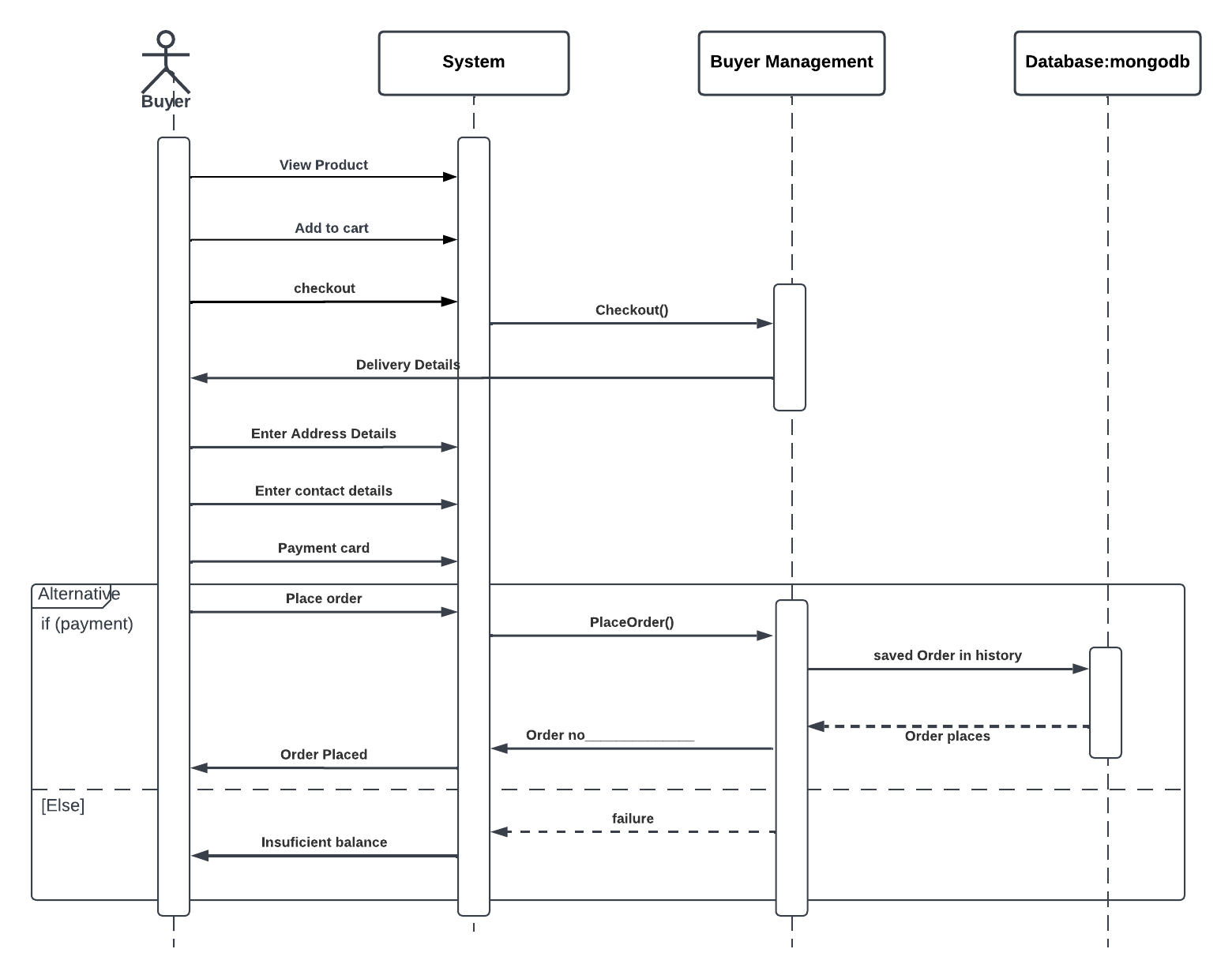
**Figure 35: Confirm Shipment**

* + 1. **Search & View Product**

****

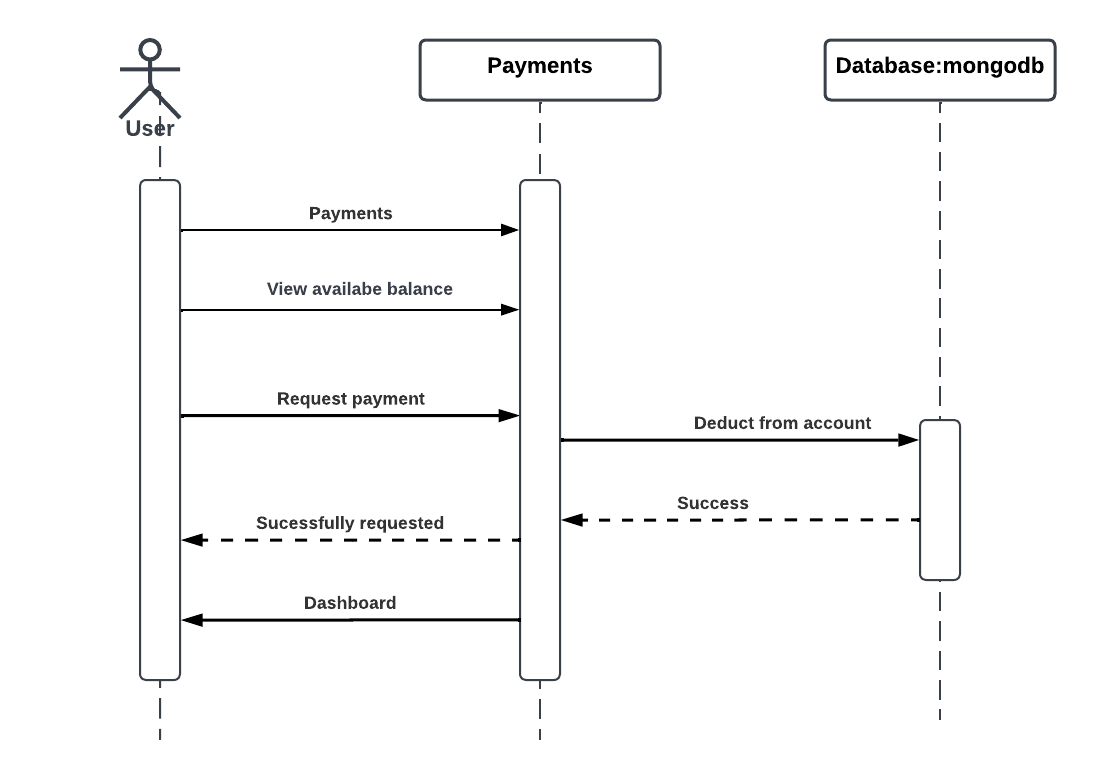
**Figure 36: Search & View Products**

* + 1. **Place order**

****

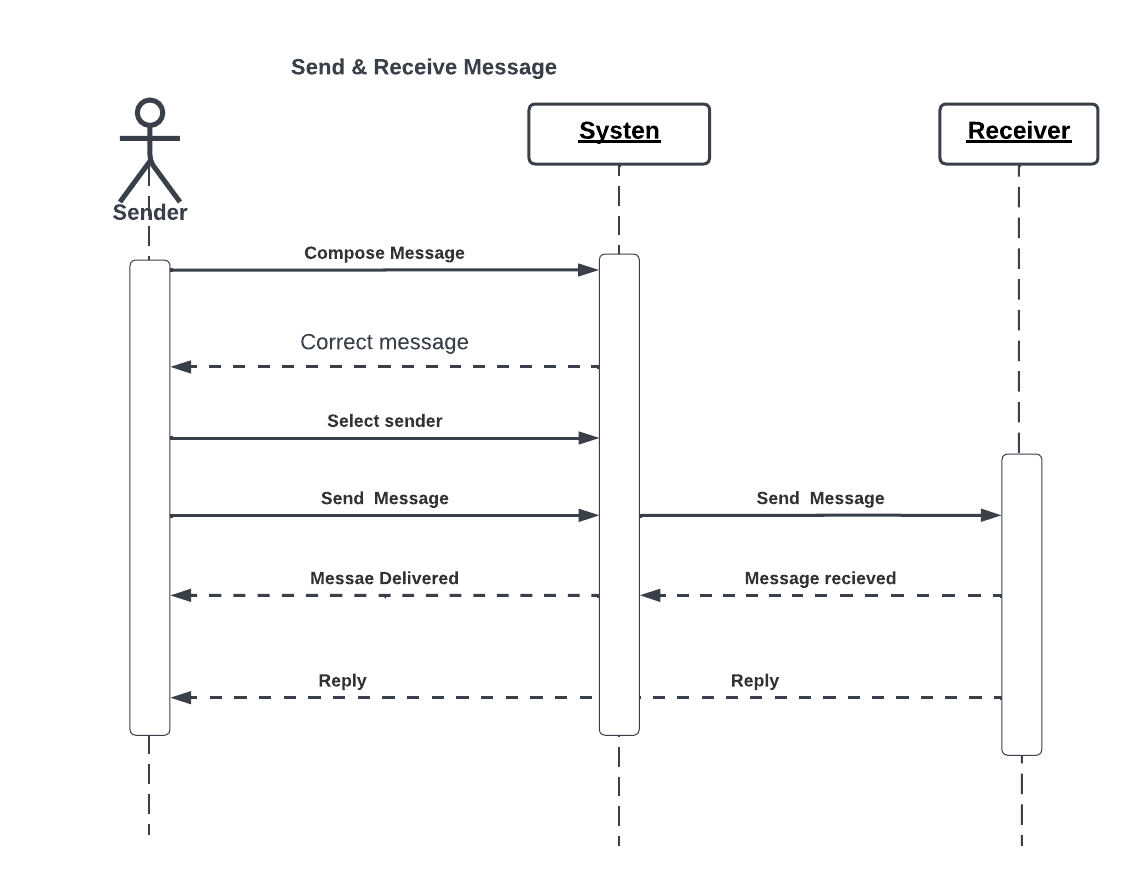
**Figure 37: Place Order**

* + 1. **Request Payments**

****

**Figure 38: Request Payments**

* + 1. **Send & Receive Message**



**Figure 39: Send & Receive Message**

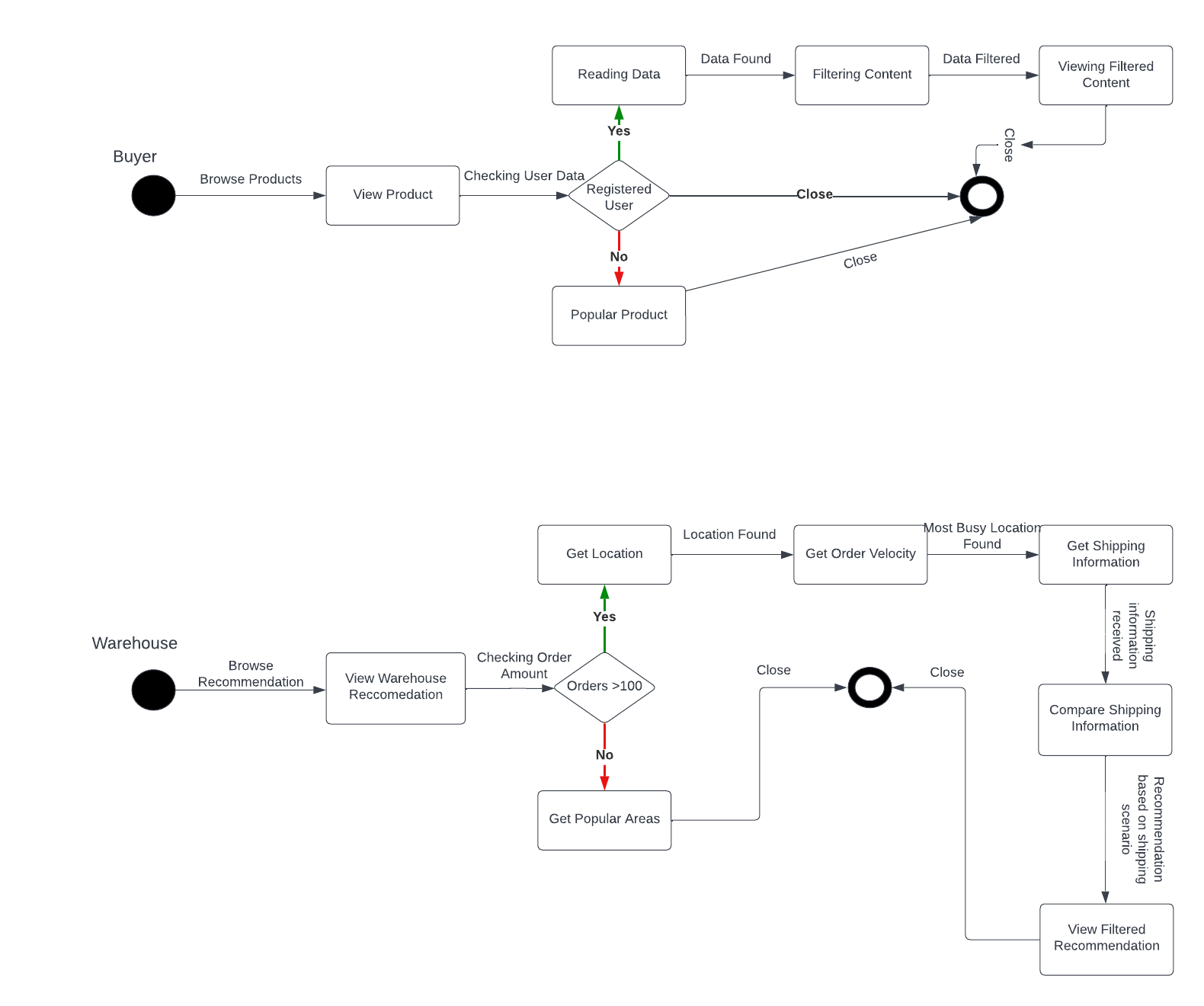
* 1. **State Transition Diagram** 
     1. **Product Recommendations**

**Diagram

Description automatically generated**

**Figure 40: Product Recommendations**

* + 1. **Warehouse Recommendations**



**Figure 41: Warehouse Recommendations**

# Data Design

Data is stored in a non-relational database; the database schema is written below.

“**Address**”: {

“AddressId”:”Schema.Types.ObjectId”

“Building”:”String”,

“Street”:”String”,

“City”:”String”,

“Country”:”String”,

“PostalCode”:” String”

},

“**Reviews**”: {

“Rating”:”Number”,

“Feedback”:”String”,

“ShipperId”:” Schema.Types.ObjectId”,

“BuyerId”:” Schema.Types.ObjectId”,

},

“**Bank Account**” :{

“CreditNumber”:” String”,

“CVV”:”Number”,

“ExpiryDate()”:” Date()”;

},

“**Payment**”: {

“Amount”:”Number”,

“Token”:”String”,

“Datetime”:”String”,

“Status”:”String”

},

“**Chat**”: {

“SellerId:” Schema.Types.ObjectId”

“UserId:” Schema.Types.ObjectId”

“Messages”:” String []”,

},

“**Product**”:{

productname:{

type:String,

required:true,

trim:true,

maxLength:100,

},

**productprice**:{

type:Number,

required:true,

default:0.0

},

**productdescription**:{

type:String,

required:true,

},

**productstock**:{

type:Number,

required:true,

default:0

},

productimages:{

type:[String],

required:true,

},

productcategory:{

type:String,

// required:true,

productseller:{

type: String,

// required:true,

},

rating:{

type:Number,

default:0,

},

numberofreview:{

type:Number,

default:0,

},

createdAt: {

type: Date,

default: new Date(),

},

reviews:[

{

name:{

type:String,

required:true,

},

}

]

})

};

“**Order**”: {

“Buyer”:” Schema.Types.ObjectId”,

“Quantity”:”Number”,

“Amount”:”Number”,

“OrderDate”:”Date()”,

“ShippingDate”:”Date()”,

},

“**User**”: {

“name”:”String”,

“cnic”:”String”,

“email”:”String”,

“phone”:”String”,

“address”:” Schema.Types.ObjectId”

“account”:” Schema.Types.ObjectId”,

};

“**WarehouseAdmin**”:{

“Id”:” Schema.Types.ObjectId”,

},

“Warehouse”:{

“name”:”String”,

“Area”:”String”,

“Location”:”Address”,

“WarehouseAdmin”: “Schema.Types.ObjectId”

“account”:” Schema.Types.ObjectId”,

}

“**System Admin**”: {

“Id”:” Schema.Types.ObjectId”,

“account”:” Schema.Types.ObjectId”,

“Sellers”:” Schema.Types.ObjectId[]”,

“Buyers”:” Schema.Types.ObjectId[]”,

“WarehouseAdmins”:” Schema.Types.ObjectId[]”

},

“**Seller**”: {

“Id”:” Schema.Types.ObjectId”,

“Address”:” Schema.Types.ObjectId”,

“Company":” Schema.Types.ObjectId”,

“account”:” Schema.Types.ObjectId”,},

## Data Dictionary

**Table 1: Data Dictionary**

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity** | **Attributes** | **Data type** | **Description** |
| Address | Building | String | Building Name or number |
| Address | Street | String | Street Name or number |
| Address | City | String | City name |
| Address | Country | String | Country name |
| Address | Postal Code | String | Zip Postal area code |
| Reviews | Rating | Number | Seller rating |
| Reviews | Feed Back | String | Seller feedback |
| Reviews | Buyer Id | Schema.Types.ObjectId | Buyer Id |
| Reviews | Shipper Id | Schema.Types.ObjectId | Shipper Id |
| Product | Product Name | String | Name of Product |
| Product | Product Price | Number | Price of Product |
| Product | Product Description | String | Description of Product |
| Product | Product Stock | Number | Amount of Product |
| Product | Product Images | Schema.Types.ObjectId [] | Images of Product |
| Product | Product Category | String | Category of Product |
| Product | Product Seller | Schema.Types.ObjectId | Seller of Product |
| Product | Product Rating | Number | Rating of Product |
| Product | Number of Reviews | Number | Number of Reviews on Product |
| Product | Created At | Date | Listing Date |
| Product | Reviews | Schema.Types.ObjectId[] | Reviews on Product |
| Chat | Buyer Id | Schema.Types.ObjectId | Id of Buyer |
| Chat | Seller Id | Schema.Types.ObjectId | Id of Seller |
| Chat | Messages | String[] | All messages from chat |
| Orders | Buyer Id | Schema.Types.ObjectId | Id of Buyer |
| Orders | Quantity | Number | Quantity of products ordered. |
| Orders | Amount | Number | Total Cost of Order |
| Orders | Order Date | Date | Date of Order Placement |
| Orders | Shipping Date | Date | Shipping Date of Order |
| User | Name | String | Name of user. |
| User | CNIC | String | CNIC of user. |
| User | Email | String | Email if of user. |
| User | Phone | String | Phone number of user |
| User | Address | String | Address of User |
| User | Account | Schema.Types.ObjectId | Account details of user |
| Warehouse | Name | String | Name of Warehouse |
| Warehouse | Area | String | Area of Warehouse |
| Warehouse | Location | String | Location of Warehouse |
| Warehouse | Warehouse Admin | Schema.Types.ObjectId | Admin of Warehouse |
| Warehouse | Account | Schema.Types.ObjectId | Account Details of Warehouse |
| Admin | Id | Schema.Types.ObjectId | Admin Id |
| Admin | Account | Schema.Types.ObjectId | Account Details of Admin |
| Admin | Sellers | Schema.Types.ObjectId[] | Sellers of website. |
| Admin | Buyers | Schema.Types.ObjectId[] | Buyers of website |
| Admin | Warehouse Admin | Schema.Types.ObjectId[] | Warehouse associated with website |
| Seller | Id | Schema.Types.ObjectId | Seller Id |
| Seller | Address | String | Address of Seller |
| Seller | Company | String | Company of Seller |
| Seller | Account | Schema.Types.ObjectId | Account Details of Seller |
| Review | Rating | Number | Review Number |
| Review | Feedback | String | Feedback of Buyer |
| Review | Seller Id | Schema.Types.ObjectId | Seller Id |
| Review | Buyer Id | Schema.Types.ObjectId | Buyer which gives review |

# Human Interface Design

## Screen Images

* + 1. **Signup Screen**

Graphical user interface, application

Description automatically generated

**Figure 42: Sign up Screen**

* + 1. **Sign in Screen**

Graphical user interface

Description automatically generated

**Figure 43: Sign in Screen**

* + 1. **Admin Dashboard**

Graphical user interface, chart, application

Description automatically generated

**Figure 44: Admin Dashboard**

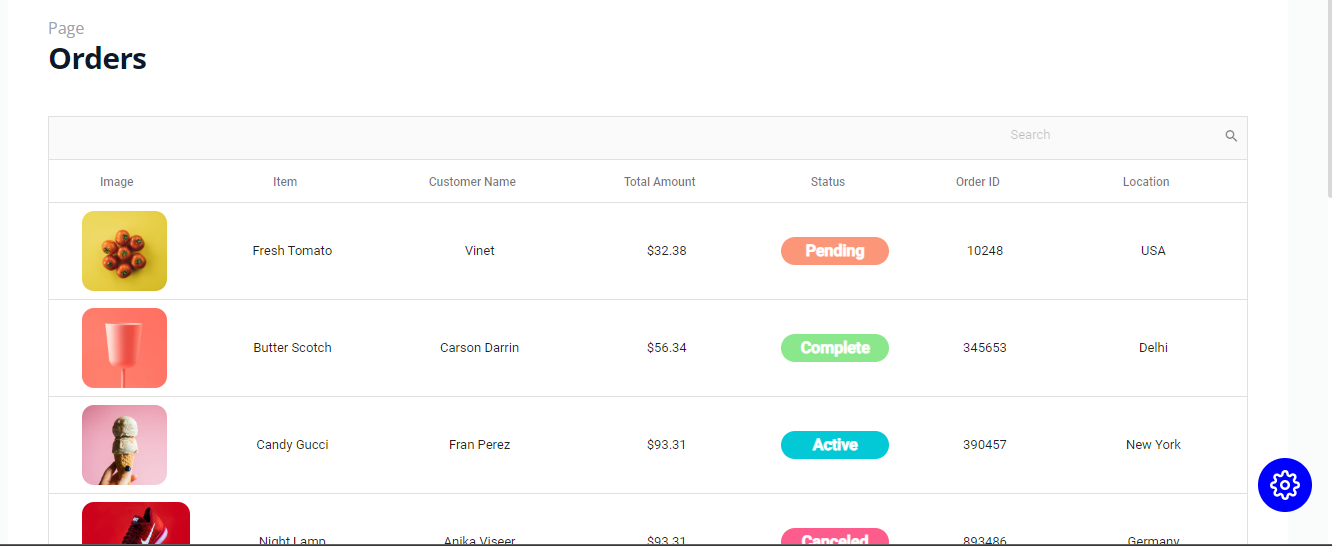
* + 1. **Seller Dashboard**

**Graphical user interface, chart

Description automatically generated**

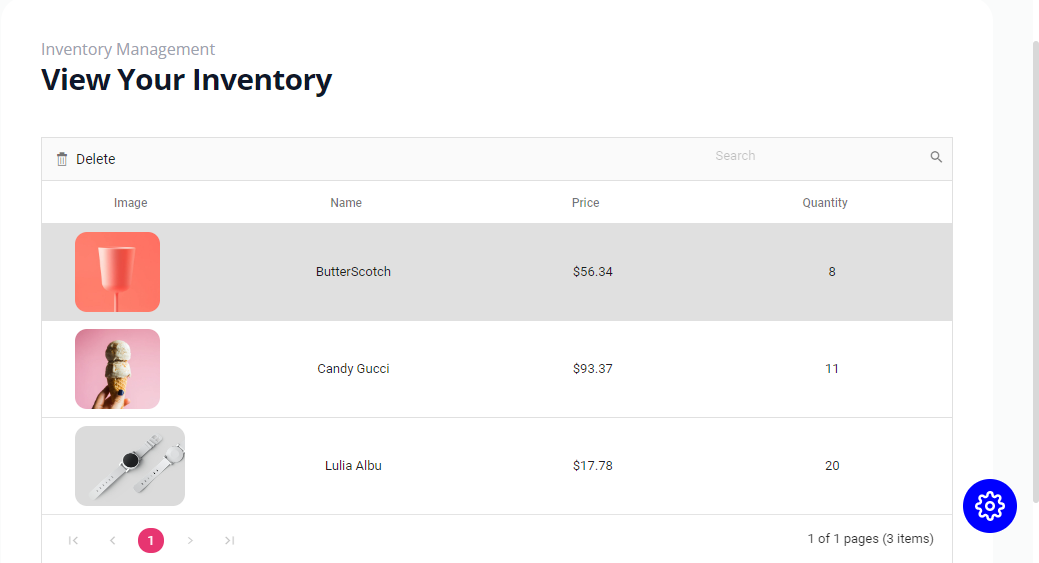
**Figure 45: Seller Dashboard**

* + 1. **Order Screen**



**Figure 46: Order Screen**

* + 1. **Inventory Screen**



**Figure 47: Inventory Screen**

## Screen Objects and Actions

Figure 42: It is the Signup screen.

Figure 43: It is the Sign in screen.

Figure 44: It is the Admin Dashboard screen of the application after user is navigated from sign in screen.

Figure 45: It is the Seller Dashboard screen of the application after user is navigated from sign in screen

Figure 46: It is the order screen

Figure 47: It is the inventory screen.

# Implementation



## Algorithms

**Machine Learning**

**Product Recommendation to Buyers**

User will log in to account

If (User Has account):

System will fetch the user details

System will analyze the user details.

System will display products on home screen which are according to the content the user consumes.

Else:

Popular Products will be displayed on user’s home screen.

**Warehouse Recommendation to Warehouse Manager:**

User will login to account.

If (User Has account):

If (Order Quantity>100):

System will fetch the order data.

System will fetch area from where most order received.

System will calculate cost to profit ratio of opening of warehouse.

System will recommend warehouse if profit is greater than cost.

Else:

Warehouse will be recommended in the area with most traffic.

Else:

No alternative route.

## External APIs/SDKs

**Table 2: Details of APIs used in the project**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name of API and version** | **Description of API** | **Purpose of usage** | **List down the API endpoint/function/class in which it is used** |
| Stripe (version 2020-08-27) | Credit Card payment integration | For order placement | stripe.paymentMethods.create |

## User Interface

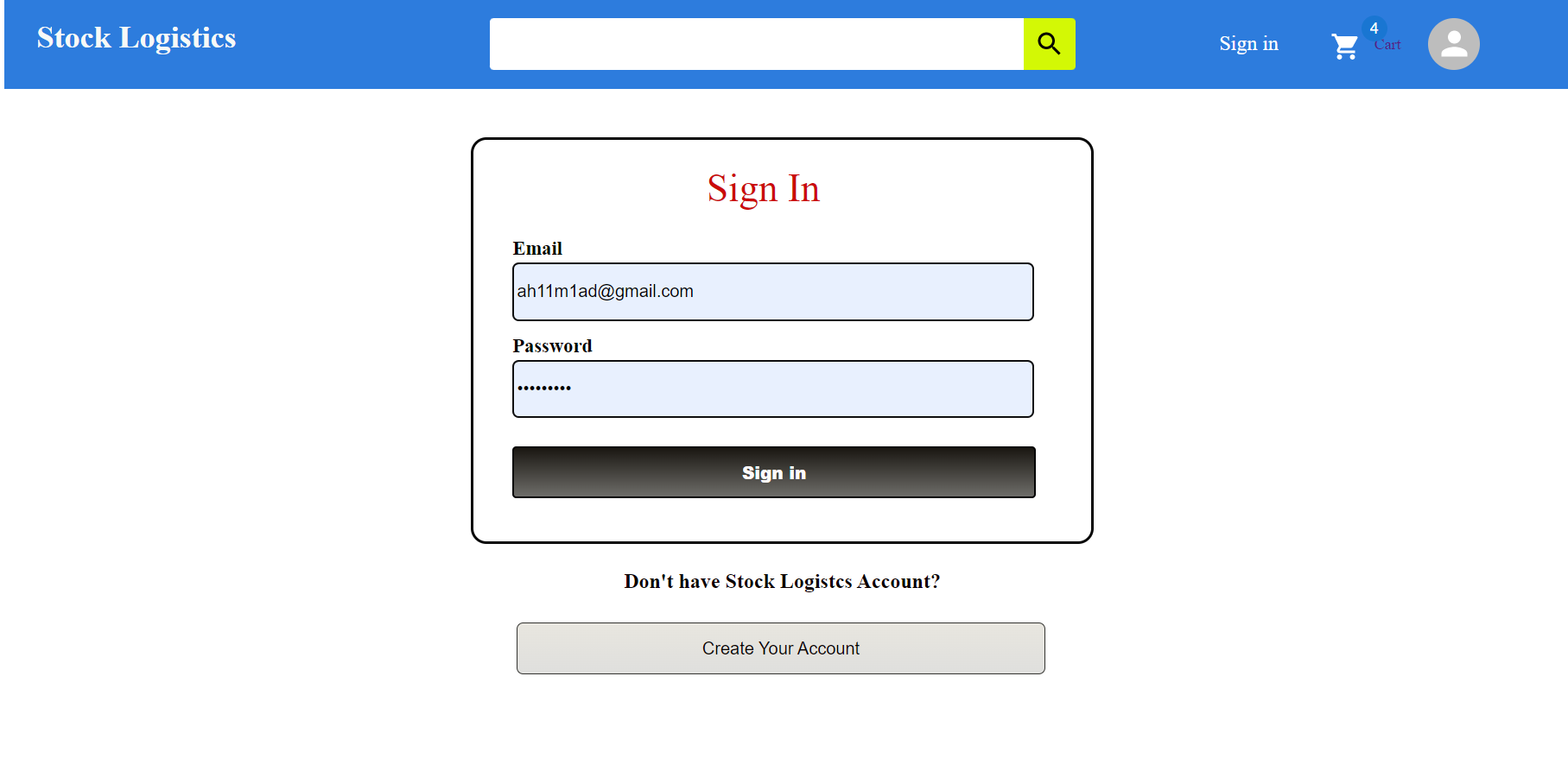
* + 1. **Signup Screen**

Graphical user interface, application

Description automatically generated

**Figure 48: Sign up Screen**

* + 1. **Sign in Screen**



**Figure 49: Sign in Screen**

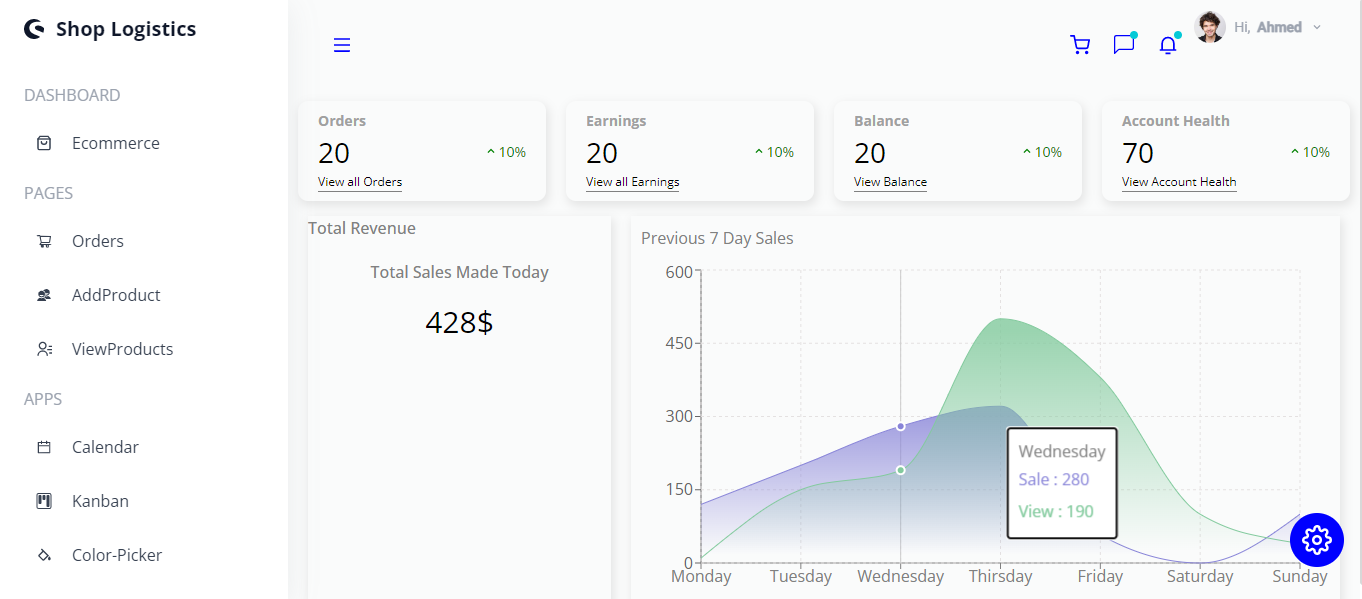
* + 1. **Admin Dashboard**

Graphical user interface, chart, application

Description automatically generated

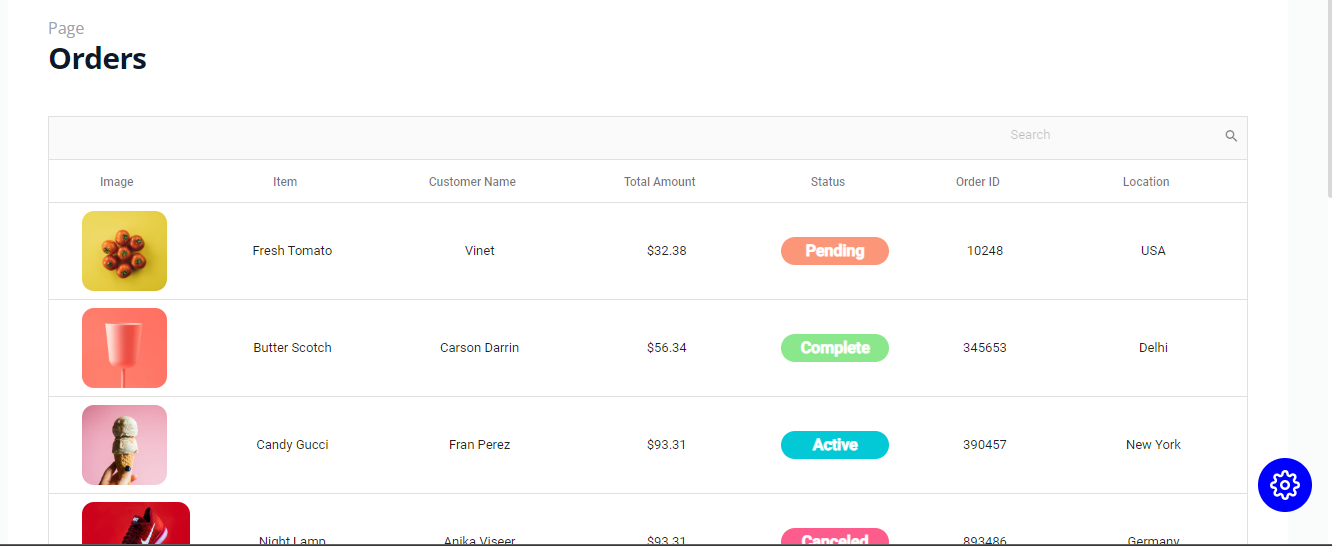
**Figure 50: Admin Dashboard**

* + 1. **Seller Dashboard**

****

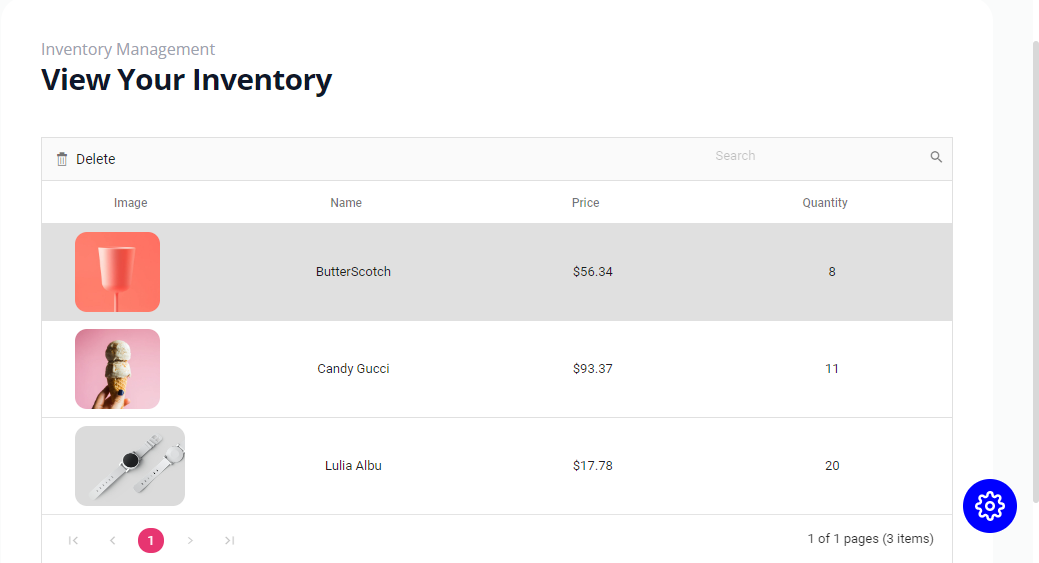
**Figure 51: Seller Dashboard**

* + 1. **Order Screen**



**Figure 49: Order Screen**

* + 1. **Inventory Screen**



**Figure 50: Inventory Screen**

* + 1. **Supplier Screen**

Graphical user interface, application

Description automatically generatedFigure 3 shows the mockup of supplier side of the Stock Logistics.

Figure 51: Supplier Screen

* + 1. **Buyer Screen**

Figure 4 shows the mockup of buyer side of the Stock Logistics.

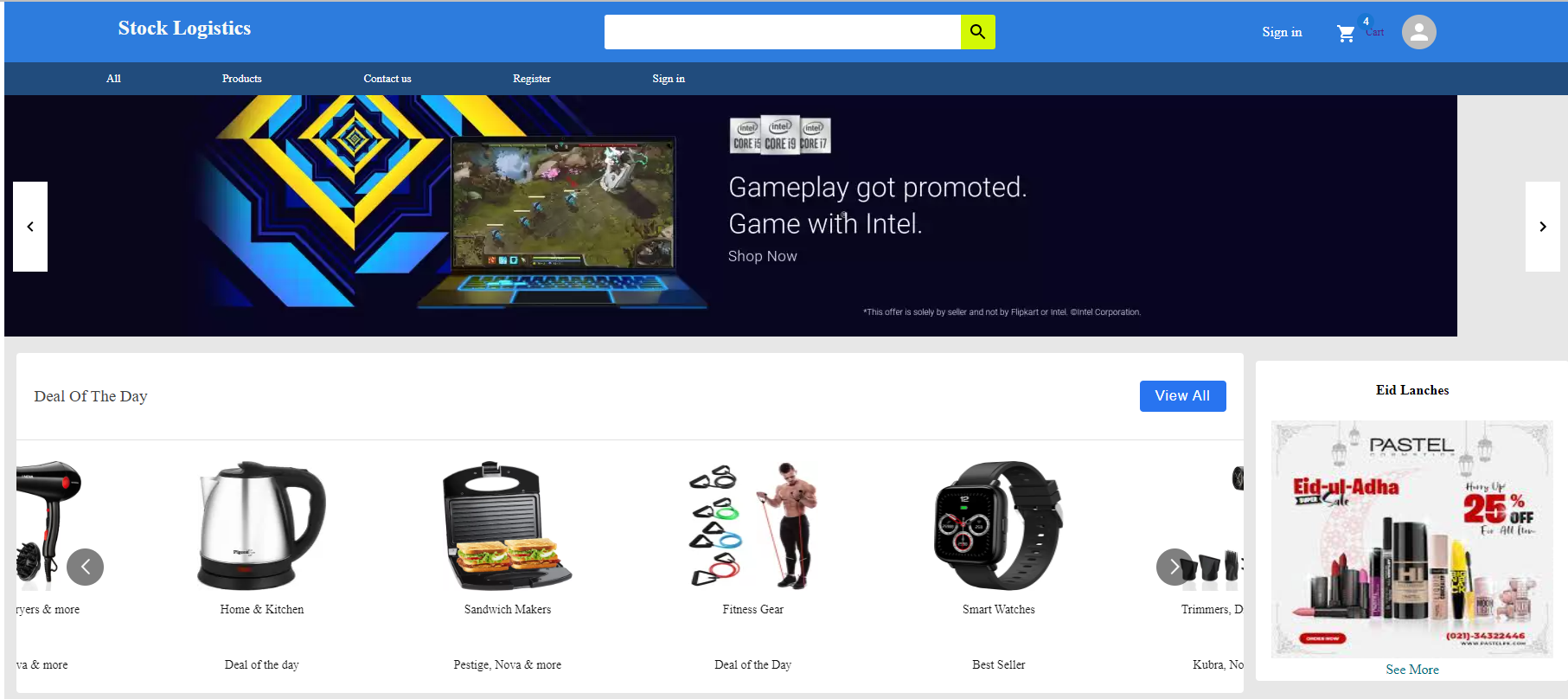
****

Figure 52: Buyer side

* + 1. **Checkout Screen**

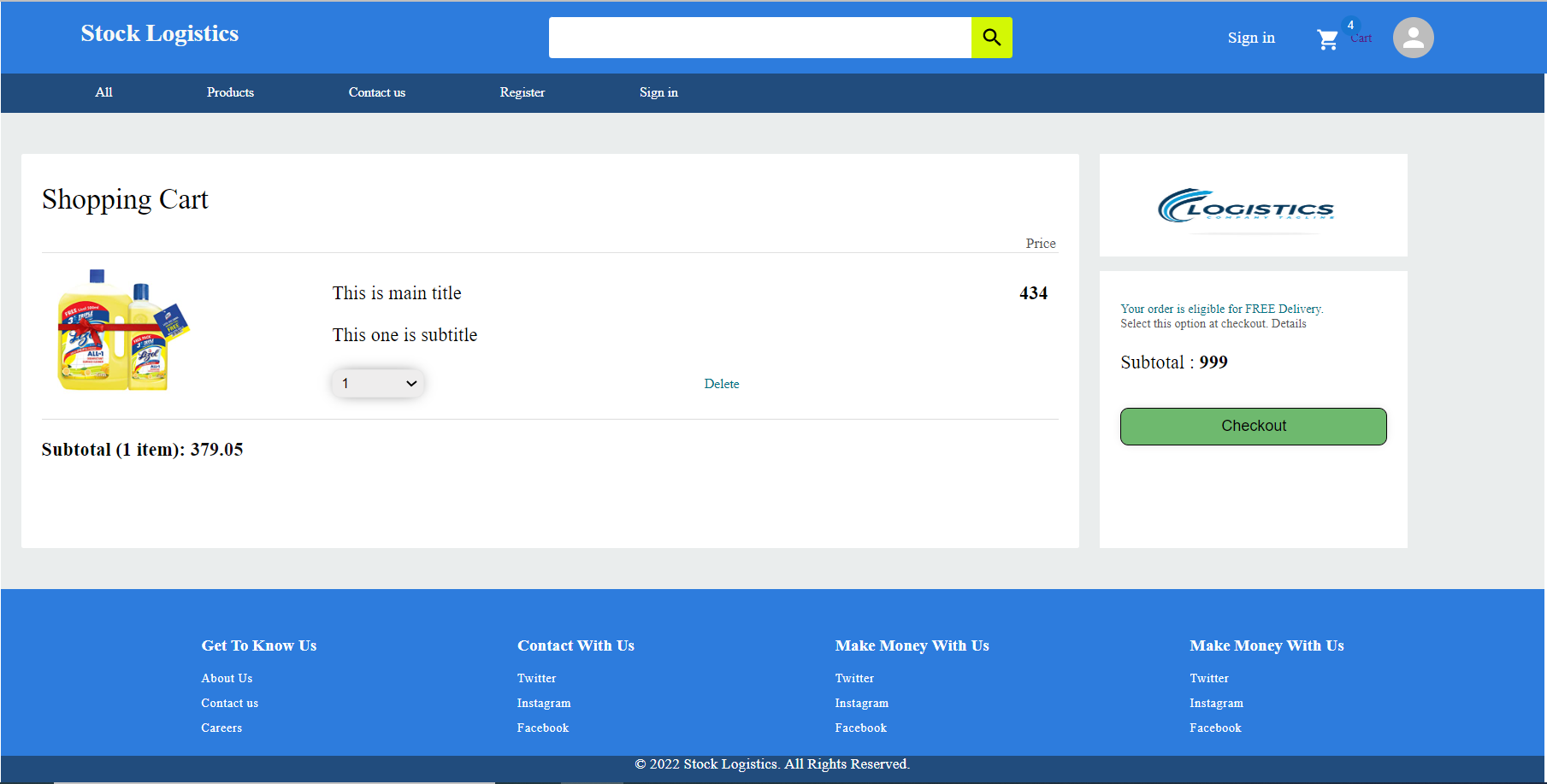
****

Figure 53: Buyer side

# Testing and Evaluation



## Unit Testing

**Unit Testing 1:** Login as buyer with valid and invalid Email

**Testing Objective:** To ensure the login form is working correctly with valid and invalid credentials/inputs.

**Table 3: Unit Testing 1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1 | Check the email field of login to validate that it takes proper email | Email: [waqasviki123@gmail.com](mailto:waqasviki123@gmail.com) | Validates email address and moves cursor to next textbox | Pass |
| 2 | Check the email field of login to validate that it takes proper email | Email: [ahmedpandit2@gmail.com](mailto:ahmedpandit2@gmail.com) | Validates email address and moves cursor to next textbox | Pass |
| 3. | Check the email field of login to validate that it displays error message. | Email:  Waqas123.gmail.com | Highlights field and displays error message | Pass |
| 4. | Check the email field of login to validate that it displays error message. | Email:  Sania+gmail.com | Highlights field and displays error message | Pass |

**Unit Testing 2:** Login as buyer with valid and invalid credentials

**Testing Objective:** To ensure the login form is working correctly with valid and invalid Password.

**Table 4: Unit Testing 2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
|  | Check that password is inserted in form or stars | Password:  Ali123@ | Password in stars: \*\*\*\*\*\*\* | Pass |
|  | Check that password is inserted in form or stars | Password:  @qwerty23 | Password in stars: \*\*\*\*\*\*\*\*\* | Pass |

**Unit Testing 3:** Login as buyer with valid and invalid credentials

**Testing Objective:** To ensure that login cannot be made with empty fields

**Table 5: Uni testing 3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
|  | Check whether sign in button give error message on empty fields or not? | Email:  Password: | Highlights field and displays error message that fields cannot be empty | Pass |

**Unit Testing 4:** Login as Seller with valid and invalid Email

**Testing Objective:** To ensure the login form is working correctly with valid and invalid credentials/inputs.

**Table 6: Unit Testing 4**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
| 1 | Check the email field of login to validate that it takes proper email | Email: [ahmadmir123@gmail.com](mailto:ahmadmir123@gmail.com) | Validates email address and moves cursor to next textbox | Pass |
| 2 | Check the email field of login to validate that it takes proper email | Email: [waqashassan98@gmail.com](mailto:waqashassan98@gmail.com) | Validates email address and moves cursor to next textbox | Pass |
| 3. | Check the email field of login to validate that it displays error message. | Email:  abc.gmail.com | Highlights field and displays error message | Pass |
| 4. | Check the email field of login to validate that it displays error message. | Email:  raza+gmail.com | Highlights field and displays error message | Pass |

**Unit Testing 5:** Login as Seller with valid and invalid credentials

**Testing Objective:** To ensure the login form is working correctly with valid and invalid Password.

**Table 7: Unit Testing 5**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
|  | Check that password is inserted in form or stars | Password:  I8r523@ | Password in stars: \*\*\*\*\*\*\* | Pass |
|  | Check that password is inserted in form or stars | Password:  @qwerty23 | Password in stars: \*\*\*\*\*\*\*\*\* | Pass |

**Unit Testing 6:** Login as Seller with valid and invalid credentials

**Testing Objective:** To ensure that login cannot be made with empty fields

**Table 8: Unit Testing 6**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Result** |
|  | Check whether sign in button give error message on empty fields or not? | Email:  Password: | Highlights field and displays error message that fields cannot be empty | Pass |

## Functional Testing.

**Functional Testing 1:** Complete login for buyer with different accounts.

**Objective**: To ensure that the Buyer can only login with correct credentials.

**Table 9: Functional testing 1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Actual result** | **Result** |
| 1. | Login as Buyer | Username waqasviki397@gmail.com  Password:  1234 | Home screen page for buyer | Logged in and redirected to home screen. | Pass |
| 2. | Login as Buyer | Username ahmedhabib12@gmail.com  Password:  9876 | Home screen page for buyer | Logged in and redirected to home screen. | Pass |
| 3. | Login as Buyer | Username ahmedhabib12@gmail.com  Password:  3456656 | Home screen page for buyer | Login failed – invalid credentials error | Fail |
| 4. | Login as Buyer | Username saniaali12@gmail.com  Password:  9876 | Home screen page for buyer | Login failed – invalid email error | Fail |

**Functional Testing 2:** Complete login for seller with different accounts.

**Objective**: To ensure that the Buyer can only login with correct credentials.

**Table 10: Functional Testing 2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Actual result** | **Result** |
| 1. | Login as Seller | Username waqasviki397@gmail.com  Password: 1234 | Home screen page for seller | Logged in and redirected to home screen. | Pass |
| 2. | Login as Seller | Username ahmedhabib12@gmail.com  Password: 9876 | Home screen page for seller | Logged in and redirected to home screen. | Pass |
| 3. | Login as Seller | Username ahmedhabib12@gmail.com  Password: 3456656 | Home screen page for seller | Login failed – invalid credentials error | Fail |
| 4. | Login as seller | Username saniaali12@gmail.com  Password: 9876 | Home screen page for seller | Login failed – invalid email error | Fail |

## Integration Testing

**Integration Testing 1:** Place order

**Testing Objective:** to ensure that buyer can place order.

**Table 11: Integration Testing 1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Actual result** | **Result** |
| 1. | Place order | Add clean shower to cart and checkout. | Successfully placed order with order number. | Order placed successfully | Pass |

**Integration Testing 2:** Search product

**Testing Objective:** to ensure that buyer can search product successfully.

**Table 12: Integration Testing 2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Test case/Test script** | **Attribute and value** | **Expected result** | **Actual result** | **Result** |
| 1. | Search product | Keyword:  Clean shower | View Products with similar keywords. | Product displayed | Pass |
| 2. | Search product | Keyword:  Camera | View Products with similar keywords. | Camera product | Fail |

# Plagiarism Report

Graphical user interface, text, application, email

Description automatically generated