**Mashki Online**

**SPROJ Report**



**Mohammad Aaraiz Hassan 23100300**

**Muhammad Ali Raza 24100298**

**Muhammad Ahmed Hussain Wyne 23100112**

**Abdul Ahad 24100223**

**Muhammad Usman Zafar 24100213**

**Advisor: Waqar Ahmad  
School of Science and Engineering  
Lahore University of Management Sciences  
Submission Date**

**Acknowledgement and Dedication**

We would like to express our sincere gratitude to everyone who has contributed to the completion of this project. Without their support, guidance, and encouragement, this project would not have been possible. We extend our heartfelt thanks to:

* Our supervisor/mentor, Waqar, for their invaluable guidance, support, and mentorship throughout the project. Their expertise and feedback have been instrumental in shaping the direction and quality of the project.
* The faculty members and staff of LUMS for providing the necessary resources, facilities, and infrastructure for conducting research and carrying out the project work.
* Our classmates, friends, and family for their unwavering support, encouragement, and understanding during the challenging phases of the project. Their moral support and motivation kept us inspired and focused throughout the journey.

We are deeply grateful to all the individuals mentioned above and anyone else who has contributed in any way to the successful completion of this project.

We dedicate this project to our families, whose unwavering love, support, and encouragement have been our guiding light throughout our academic journey. Their belief in our abilities and their constant encouragement inspired us to overcome challenges, pursue excellence, and strive for success. This project is a testament to their enduring influence on our lives and academic pursuits. Thank you for being our source of strength and inspiration.

**Certificate**

I certify that the senior project titled “**Mashki Online**” was completed under my supervision by the following students:

Mohammad Aaraiz Hassan, Muhammad Ali Raza,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Muhammad Ahmed Hussain Wyne, Abdul Ahad,  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Muhammad Usman Zafar  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

and the project deliverables meet the requirements of the program.

------------------------------------- Date:

**Advisor (Signature)**

------------------------------------- Date:

**Co-advisor (if any)**

**Table of Contents**

[List of Figures 10](#)

[1. Introduction 11](#)

[a. Introduction 11](#)

[b. Objective and Scope 11](#)

[c. Development Methodology 12](#)

[d. Contributions 13](#)

[2. System Requirements 13](#)

[a. System Actors 13](#)

[b. Functional Requirements 15](#)

[Customer Module: 15](#_557u3whp03n5)

[Local Mineral Water Plant Module: 16](#_qwq4c3r8o256)

[Transporter Module: 16](#_4pk4xyzbpbai)

[Testing Laboratory Module: 17](#_82jdy7a6k7v4)

[Admin Module: 17](#_bbofiru20nt6)

[c. Non-functional Requirements 18](#)

[3. System Architecture 19](#)

[a. Architecture Diagram 19](#_gjdgxs)

[b. Architecture Description 19](#_30j0zll)

[c. Justification of the Architecture 21](#_1fob9te)

[Pros: 21](#_hfk8g3oxd6c0)

[Cons: 21](#_imxydmkesam3)

[Justification for the Architecture: 22](#_9y5ykwnlsojs)

[Non-Functional Requirements: 23](#_w8ve2vbj50fc)

[d. Tools and Technologies 23](#)

[4. Requirements Specifications 25](#)

[a. Use Cases 26](#)

[Admin - Adjust Pricing 27](#_orgcs54gho09)

[Admin - View Inflows and Outflows 28](#_snpeub5m0xaz)

[Admin - Manage Water Plants 30](#_4cfaqk85zkhb)

[Admin - Add a New Water Plant 31](#_2d2zfc5xwluu)

[Admin - Delete an Existing Water Plant 32](#_4iovaj59cg0r)

[Customer - Sign Up 33](#_wu1x3ck0uw7x)

[Customer - Log in 34](#_3gqzuaytd0y)

[Customer - Add/Select Delivery Address 36](#_1j3s48nokbnq)

[Customer - Track Order Status 37](#_e9wj2r3wgn5u)

[Customer - Give Rating/Feedback 38](#_ue28onpxlrbf)

[Customer - Place an Order 39](#_wq31z7wqyp21)

[Transporter - Receive Orders 40](#_r0qo6p3nh24l)

[Transporter - Update Order Status 41](#_wrvku11py8o0)

[Transporter - Receive Delivery Routes 42](#_c77izbnkgb7w)

[Local Mineral Water Plants - Check Inflows and Outflows 43](#_7xdc3zvr8db4)

[Local Mineral Water Plants - Update Production Capacity 45](#_jmgeogq1gd6z)

[Local Mineral Water Plants - Update Inventory 46](#_7j4qyol2berk)

[Local Mineral Water Plants - Get a list of ongoing orders 47](#_m08j8jz1sp9m)

[Testing Laboratories - Receive Testing Instructions 48](#_vv544xxnpg0i)

[Testing Laboratories - Enter Test results 49](#_pgdw5g2xbu7b)

[b. Class Diagram 50](#_2et92p0)

[Admin: 50](#_c0s0vxnqe5i1)

[Customer: 51](#_xp5rm5wnpzyo)

[Plant: 51](#_rnjtoox0exm6)

[Labs: 51](#_z1tzdkffbef)

[Payment: 52](#_6wdd47w6vxc5)

[Order: 52](#_i58kkf4u3d13)

[Transporter: 52](#_zqmclrafxw1)

[c. Sequence Diagrams 53](#_tyjcwt)

[Admin - Adjust Pricing 53](#_ql27xk32x7qs)

[Admin - View Inflows and Outflows 54](#_8qkqv5sp5w1)

[Admin - Manage Water Plants (Add and Delete Included) 55](#_8zprvogcfcoc)

[Admin - Add a New Water Plant 56](#_kx9twvfplaw2)

[Admin - Delete an Existing Water Plant 56](#_a4fxybshlgxx)

[Customer - Sign Up 56](#_e1cxyklujhs4)

[Customer - Log in 57](#_xq6gav1mymqc)

[Customer - Add/Select Delivery Address 58](#_li9ien2l33ar)

[Customer - Track Order Status 59](#_v4g19hvp48v8)

[Customer - Give Rating/Feedback 60](#_fmqb20jfektm)

[Customer - Place an Order 61](#_eyqftnomm6jo)

[Transporter - Receive Orders 62](#_val6pjsetv1q)

[Transporter - Update Order Status 63](#_tsn9ivwk9lvn)

[Transporter - Receive Delivery Routes 64](#_7g0gzp99kmsu)

[Local Mineral Water Plants - Check Inflows and Outflows 65](#_u0s23n6goe9z)

[Local Mineral Water Plants - Update Production Capacity 66](#_8484w0l022cw)

[Local Mineral Water Plants - Update Inventory 67](#_3bq8c63pu832)

[Local Mineral Water Plants - Get a list of ongoing orders 68](#_2474subo51o5)

[Testing Laboratories - Receive Testing Instructions 69](#_41hf7uea54nv)

[Testing Laboratories - Enter Test results 69](#_t92nj91jkn63)

[5. Software Development Methodology and Plan 69](#)

[a. Software Process Selection 70](#_3dy6vkm)

[1. Waterfall 70](#_syq99arlmso8)

[2. Agile(Scrum) 70](#_ubbvx5xsaftj)

[b. Gantt Chart 72](#_1t3h5sf)

[6. Database Design and Web Services 72](#)

[a. Database Design 72](#)

[b. API Specification 80](#)

[7. System User Interface 81](#)

[8. Project Security 86](#)

[a. Project Threats 86](#)

[b. Potential Losses 86](#)

[c. Security Controls 86](#)

[d. Static and Dynamic Security Scanning Tools 87](#)

[9. Risk Management 88](#)

[Potential Risks and Mitigation Strategies 88](#_4d34og8)

[10. Testing and Evaluation 91](#)

[11. Deployment Guidelines 92](#)

[12. Conclusion 92](#)

[a. Summary 92](#)

[b. Challenges 92](#)

[c. Future 93](#)

[13. Review checklist 94](#_2s8eyo1)

[14. References 95](#)

# List of Figures

# Introduction

## Introduction

Describe the application that you are developing. Also discuss:

1. Domain of your application
2. Target users of this application
3. What have you produced: a mobile app, a web app etc.?

MashkiOnline's scope extends to the complete supply chain management of water bottle delivery. It includes the integration of local mineral water plants, transporters, testing laboratories, and end-users through a user-friendly mobile (Android) application. This ambitious project segments urban areas into zones, each with its production facility.

## Objective and Scope

In this subsection, please define the objectives of your application, why you chose to develop this application and how it would impact/enhance the business operations in the selected domain.

MashkiOnline is a groundbreaking service poised to redefine the way 19-liter water bottles are delivered to homes, offices, and various other locations. At its core, MashkiOnline is driven by a singular mission: to provide access to safe, high-quality drinking water at a price point that's affordable for everyone. In a world where the cost of pure, potable water can often be expensive, MashkiOnline will be a name of affordability and convenience.

Unlike traditional water bottle providers like Nestle, Aquafina, Gourmet, or Dasani, MashkiOnline takes a different approach. It doesn't produce water itself, nor does it rely on a single centralized production facility. The soaring cost of water bottles from these providers is partly due to the long-distance transportation involved, forcing customers to bear the burden of these costs. Even local alternatives often provide scheduled deliveries, leaving customers in a lurch when they have urgent needs for water, like during events such as Eid.

MashkiOnline aims to build a relationship of trust with customers by offering on-demand water bottle delivery without the premium price tag. How will it achieve this? By leveraging a network of local mineral water plants, each tailored to serve specific zones within a city. This decentralization slashes transportation costs, ensuring that pure water reaches customers without breaking the bank. The system is further fine-tuned by an advanced routing algorithm, optimizing delivery routes for transporters to guarantee cost-effectiveness and timeliness.  
  
Central to MashkiOnline's operation is the unwavering commitment to water quality. To guarantee the highest standards of purity, the service collaborates with expert laboratories specializing in water quality testing. These laboratories will be integrated into the supply chain, conducting random testing at various stages, from production to transportation and even after delivery to the customer.

MashkiOnline is a multifaceted ecosystem where various stakeholders come together for a common purpose. This includes local mineral water plants, transporters, testing laboratories, and, of course, the end-users - individuals and organizations in need of clean drinking water. To facilitate seamless operations, each stakeholder is provided with specialized tools and interfaces. Plant managers have their dashboards for production management, transporters have optimized route planning tools, laboratories have user-friendly interfaces for test data entry, and customers have an intuitive platform for ordering and tracking deliveries.

## Development Methodology

Our project was conducted using an Agile framework, specifically utilizing a sprint-based approach. This allowed our team to work iteratively and incrementally, adapting to changes and feedback rapidly. The team consisted of students filling various roles such as developers, project managers, and testers, which were rotated to ensure all members gained experience in different aspects of the project.

Throughout the project, we documented our processes and code extensively. This documentation included technical details for future development phases and user guides to assist end-users in navigating our software.

## Contributions

Our application automates the task of receiving orders and deploying delivery riders. It also helps in managing water plants and making sure that the quality standards are met. Water plants can be added or deleted easily which helps in management. Customers can give reviews and track their order status. Delivery riders can look at all the orders that are to be delivered and then select their route accordingly. Admin can set the price of the products which will be visible to all the customers. This helps in maintaining customer relations as customers are always kept up-to-date about the prices and delivery status.

# System Requirements

Brief introduction of this chapter in a paragraph highlighting the content

## System Actors

List down the actor names and give a 2-3 lines description of the role of each actor

| **Actor Name** | **Description** |
| --- | --- |
| Customers | Individuals and organizations requiring 19-liter water bottle deliveries. |
| Admin | System administrators responsible for managing system-wide operations, including pricing adjustments, zone creation, and other administrative tasks. |
| Transporters | Individuals or companies responsible for delivering water bottles from production facilities to customers. |
| Testing Laboratories | Expert laboratories responsible for testing water quality at various stages of the supply chain. |
| Local Mineral Water Plants | Water production facilities distributed across different city zones, responsible for supplying high-quality water. |

## Functional Requirements

List down system requirements. You may group requirements according to actors or modules

### Customer Module:

| **Requirements** | |
| --- | --- |
| **Sr#** | **Requirement** |
| 1 | As a customer, I want to place an order for 19-liter water bottles through the MashkiOnline mobile or web application |
| 2 | As a customer, I want to specify my delivery location and schedule to receive water bottles at my convenience. |
| 3 | As a customer, I want to track the status of my order, including its production, transportation, and delivery stages. |
| 4 | As a customer, I want the option to provide feedback and report issues related to water quality or delivery. |

## 

### Local Mineral Water Plant Module:

| **Requirements** | |
| --- | --- |
| **Sr#** | **Requirement** |
| 5 | As a plant manager, I want access to a dashboard application to manage and monitor water production, quality control, and inventory for my specific zone. |

### Transporter Module:

| **Requirements** | |
| --- | --- |
| **Sr#** | **Requirement** |
| 6 | As a transporter, I want to receive orders through a mobile application. |
| 7 | As a transporter, I want to receive delivery instructions through a mobile application. |
| 8 | As a transporter, I want optimized routes for efficient delivery. |

## 

### Testing Laboratory Module:

| **Requirements** | |
| --- | --- |
| **Sr#** | **Requirement** |
| 9 | As a testing laboratory, I want a user-friendly dashboard interface to enter and update water quality test results. |
| 10 | As a testing laboratory, I want a user-friendly dashboard interface to select the stage at which the test was conducted. |

### Admin Module:

| **Requirements** | |
| --- | --- |
| **Sr#** | **Requirement** |
| 11 | As an admin, I want to access a comprehensive dashboard for system-wide management, which includes adjusting pricing for water bottles, creating and managing delivery zones, viewing inflows and outflows of water bottles, and managing user access and permissions |
| 12 | The admin module should have robust security measures, including user authentication, role-based access control, and encryption of sensitive data. |

## Non-functional Requirements

List down non-functional requirements.

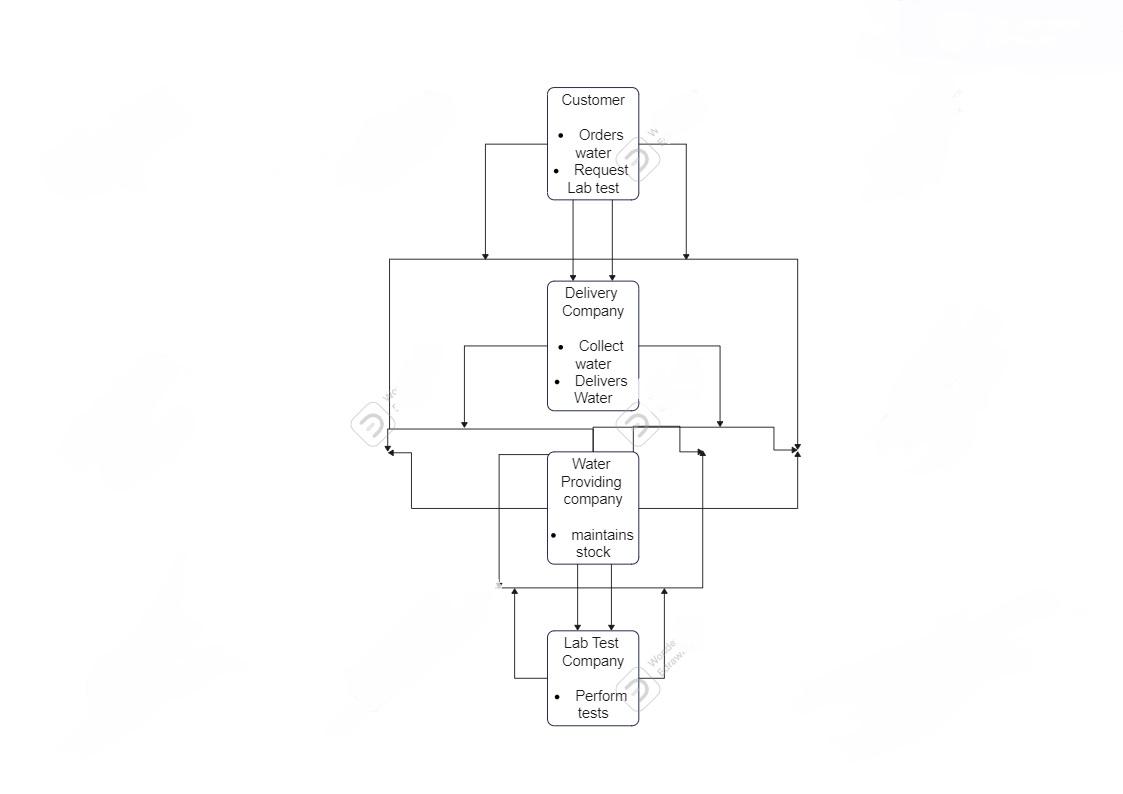
| **Sr#** | **Requirements** |
| --- | --- |
| 1 | Login Time: The login process for a user, from entering credentials to accessing the dashboard or main screen, should not exceed 20 seconds, given a stable internet connection with a minimum speed of 2 Mbps. |
| 2 | Data should be backed up daily to implement robust recovery plans in case of failure. |
| 3 | Every screen should be loaded within 10 seconds on an average smartphone |
| 4 | The app should be able to cater to thousands of concurrent users without performance degradation. |
| 5 | The system should maintain a minimum availability rate of 98% during normal operating hours. |
| 6 | The user interface should be intuitive, requiring minimal user training or guidance. |
| 7 | System should run on android devices. |
| 8 | Transactions related to bottles, money going from one stakeholder to the other will be managed within one day of delivery. |

# System Architecture

Brief introduction of this chapter in a paragraph highlighting the content

## Architecture Diagram

Draw a diagram of the system architecture.



## Architecture Description

Give description of each subsystem in the architecture diagram above. Moreover, give description of how subsystems interact with each other.

1. Customer Interface:

* Customers use this interface to place orders for water bottles. They can specify the quantity, delivery location, and payment method.
* Cashless payments can be made through the application, ensuring a convenient and secure transaction process.
* Customers can also request water quality testing services through this interface.

1. Water Provider Interface:

* Orders placed by customers are immediately visible in the Water Provider interface. The Water Provider is responsible for managing their stock of water bottles.
* The interface displays the requested quantity of water bottles and helps the provider ensure that the requested stock is available.
* Payment information received from the customer's interface is integrated here, streamlining the payment tracking process.

1. Delivery Company Interface:

* The Delivery Company interface presents delivery personnel with a clear list of tasks. This includes pickup and dropoff locations and the quantity of water bottles to be transported.
* It provides real-time route optimization and navigation assistance, helping delivery personnel efficiently execute orders.
* Payment information and delivery details are tracked here to ensure successful deliveries.

1. Lab Test Company Interface:

* When a customer requests a water quality test, this request is displayed in the Lab Test Company interface.
* The location of the Water Provider supplying the water for testing is also available here, aiding in test sample collection logistics.

## Justification of the Architecture

List down pros and cons of the architecture you have defined. These pros and cons must be discussed in the context of your system. Moreover, give a justification of why this architecture is appropriate for your system. Make sure that you also discuss how this architecture helps in implementation of your system’s non-functional requirements.

### Pros:

* Modularity and Scalability: The architecture allows for the modular development of different components, making it easy to scale and evolve each interface independently.
* Clear Separation of Concerns: Each interface serves a distinct purpose, promoting a clear separation of concerns. This simplifies development, maintenance, and updates.
* Efficient Information Flow: Orders and relevant information flow seamlessly between users, enhancing overall system efficiency. Real-time updates between interfaces ensure that all stakeholders have access to the latest data.
* Interoperability: Interfaces between customer, water provider, delivery company, and lab test company facilitate interoperability, enabling them to work together seamlessly.
* Security and Access Control: Access controls and security measures can be tailored to each interface, ensuring data privacy and integrity.
* Performance Optimization: By separating components, you can optimize the performance of each interface independently, allowing for efficient resource allocation and fine-tuning.

### Cons:

* Complexity of Integration: Integrating multiple interfaces can be complex, particularly ensuring that data flows smoothly between components.
* Maintenance Overhead: Maintaining and updating multiple interfaces may require more effort and resources, as changes in one component can impact others.
* Potential for Data Inconsistencies: Managing data consistency and synchronization between interfaces can be challenging, leading to potential data discrepancies.
* Increased Development Time: Building and maintaining separate interfaces may extend the development timeline compared to a monolithic approach.
* Security Concerns: The handling of payment information introduces potential security risks. Security measures, such as encryption and compliance with payment processing standards, are essential.

### Justification for the Architecture:

This architecture is appropriate for a water delivery system for several reasons:

* Specialization: Each interface caters to a specific stakeholder or function, allowing for specialization in design and functionality. This specialization can lead to more efficient and user-friendly interfaces tailored to the needs of each user group.
* Flexibility: The modularity of this architecture allows for the addition or removal of components as the system evolves. It's adaptable to changing business requirements and can accommodate new partners or features.
* Scalability: If any one aspect of the system experiences increased demand (e.g., more customers, additional delivery companies), you can scale that specific component independently without affecting the others.

### Non-Functional Requirements:

* Performance: Each interface can be optimized for its specific use case, ensuring fast response times for customers, accurate water testing for labs, and efficient coordination for delivery companies.
* Security: Security measures should be a top priority, especially considering the handling of payment data. This includes encryption for data in transit and at rest, access control, and compliance with industry-specific security standards. Regular security audits, vulnerability assessments, and training can help maintain a high level of security.
* Reliability: With a clear separation of concerns, it's easier to isolate issues and troubleshoot, improving system reliability.
* Scalability: The architecture's modular nature makes it easier to scale and adapt to changing loads and requirements, meeting scalability non-functional requirements.

In summary, this architecture is well-suited to a water delivery system, allowing for flexibility, specialization, and scalability. However, it does introduce some complexity in terms of integration and maintenance, which must be carefully managed.

## Tools and Technologies

List down development stack, tools and technologies etc. that you have used for development and deployment. Make sure that you mention name and version of the tools.

* React will be used for frontend. Version will be React 18
* React-native
* Node.js for backend
* Mongodb Atlas Cloud for database
* Vercel will be used for deployment
* Microsoft Azure and Aws (Amazon web services) for deployment

# 

# Requirements Specifications

Brief introduction of this chapter in a paragraph highlighting the content

This Requirements Specification section acts as a navigational compass for the MashkiOnline project, providing clarity, structure, and direction to all those involved. Here's a breakdown of its core purposes:

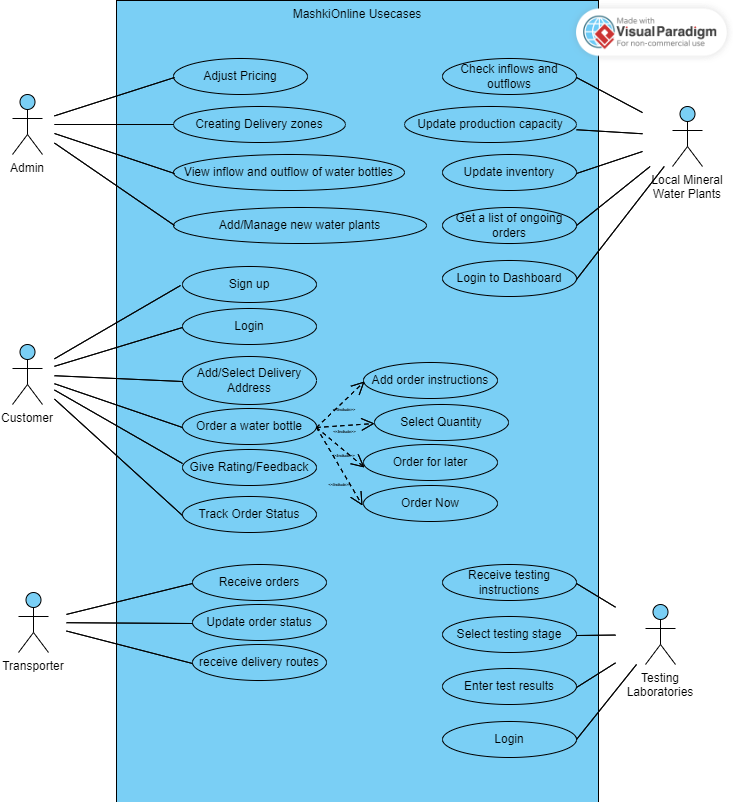
This section sheds light on specific use cases. By walking through potential scenarios, stakeholders gain a clearer understanding of system behavior, potential challenges, and the desired outcomes. It's like creating a story for every feature, ensuring its purpose and function are well-understood.

Sometimes, words alone can't do justice. That's where our diagrams come in. With UML notations and clear graphical representations, this document offers a visual guide to the system's architecture. It's a bit like having a map along with directions – ensuring clarity in both the broader view and the finer details.

By combining descriptive use cases with illustrative diagrams, this document acts as a bridge, fostering communication among stakeholders. This can be used by a developer who is looking to understand a feature or a manager wanting to see the bigger picture. This document captures the system's requirements in its current state, but it's also flexible. As feedback pours in and as market needs change, this document can be revisited and refined. Think of it as a living entity, adapting and growing with MashkiOnline.

Lastly, this isn't just a technical manual. It's a comprehensive reference guide. From understanding the role of each actor to visualizing the flow of processes with sequence diagrams, this document serves as the go-to resource for all things related to MashkiOnline.

## Use Cases

Draw use case diagrams of your system using standard UML notation. Moreover, give description of 10 core use cases of your system

### Admin - Adjust Pricing

| **Identifier** | | Admin-001 |
| --- | --- | --- |
| **Purpose** | | To allow the admin to modify the pricing structure for the water bottles in response to market demand, operational costs, or promotional campaigns. |
| **Pre-conditions** | | 1. The admin must be logged in to the administrative dashboard of MashkiOnline. 2. The admin has appropriate permissions to modify pricing details. |
| **Post-conditions** | | The new price is applied to the bottles in the user interfaces, and the new calcualtions/transactions are done based on the new price |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The admin navigates to the "Pricing Management" section of the dashboard. | |
| 2. | The system displays the current pricing for the water bottles. | |
| 3. | The admin selects the option to "Adjust Price" next to the relevant product or service. | |
| 4. | The system prompts the admin to enter the new price. | |
| 5. | The admin enters the desired price. | |
| 6. | The system validates the input to ensure it's a valid monetary amount. | |
| 7. | The system displays a confirmation message, asking the admin to verify and confirm the change. | |
| 8. | The admin confirms the new price. | |
| 9. | The system updates the price in the database and displays a success message. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | 1. At step 5, the admin may choose to enter a promotional price that lasts for a specified duration. In this case:    1. The admin enters the promotional price and the start and end date for the promotion.    2. The system validates the input and updates the price as a temporary change for the specified period.    3. After the promotional period, the system reverts the price back to its original amount before the promotion. 2. At any point before step 8, the admin can opt to cancel the process and revert to the initial pricing page. | |
| **Step #** | **Exception Paths** | |
|  | 1. In step 6, if the admin enters an invalid amount (e.g., negative value, non-numeric character), the system displays an error message prompting the admin to enter a valid amount and redirects them back to step 4. 2. In the alternate course of action 1a, if the end date for the promotion is before the start date or if it's an invalid date, the system will display an error message and prompt the admin to provide valid date entries. | |
|  |  |  |

### Admin - View Inflows and Outflows

| **Identifier** | | Admin-002 |
| --- | --- | --- |
| **Purpose** | | To allow the admin to monitor the volume of bottles/money coming into and leaving the local mineral water plants, ensuring optimal inventory management. |
| **Pre-conditions** | | 1. Admin is logged into the MashkiOnline administrative dashboard. 2. Admin has the necessary permissions to view inflow and outflow data. |
| **Post-conditions** | | Data representation of inflows and outflows for selected date range or plant is available for admin review. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | Admin navigates to the "Inflow/Outflow Monitoring" section from the dashboard | |
| 2. | System displays a visual representation (e.g., charts, graphs) of inflow and outflow metrics for each mineral water plant. | |
| 3. | Admin can select a specific date range or plant for detailed information. | |
| 4. | System updates the display based on admin's selection. | |
| 5. | Admin reviews the data. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | 1. Admin might export the data for further analysis. | |
| **Step #** | **Exception Paths** | |
|  | 1. If there's a system error fetching data, an error message is displayed. | |

### Admin - Manage Water Plants

| **Identifier** | | Admin-003 |
| --- | --- | --- |
| **Purpose** | | To facilitate the admin in overseeing and updating details about various local mineral water plants in the network. |
| **Pre-conditions** | | 1. Admin is logged into the MashkiOnline administrative dashboard. 2. Admin has the necessary permissions to manage water plant data.. |
| **Post-conditions** | | The mineral water plant's details are updated in the system. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | Admin navigates to the "Water Plant Management" section. | |
| 2. | System displays a list of all mineral water plants with essential details. | |
| 3. | Admin can select a specific plant to view its details and edit if necessary. | |
| 4. | System presents the selected plant's profile. | |
| 5. | Admin makes necessary changes and updates. | |
| 6. | System validates and saves the changes, providing a confirmation message. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | 1. Admin might opt to view the list based on certain criteria (e.g., location, capacity). | |
| **Step #** | **Exception Paths** | |
|  | 1. If invalid data is input during the update, an error message prompts correction. | |

### Admin - Add a New Water Plant

| **Identifier** | | Admin-004 |
| --- | --- | --- |
| **Purpose** | | To facilitate the admin in adding new local mineral water plants to the MashkiOnline network. |
| **Pre-conditions** | | 1. Admin is logged into the MashkiOnline administrative dashboard. 2. Admin has the necessary permissions to add new water plant data. |
| **Post-conditions** | | A new water plant is added to the MashkiOnline network and is available for monitoring and water distribution tasks. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | Admin navigates to the "Add New Plant" option in the "Water Plant Management" section. | |
| 2. | System presents a form for entering plant details like name, location, production capacity, and contact details. | |
| 3. | Admin fills out the form. | |
| 4. | System validates the input data. | |
| 5. | Upon validation, the system adds the new plant to the database and confirms the addition to the admin. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | None | |
| **Step #** | **Exception Paths** | |
|  | 1. If the admin provides invalid or incomplete data, the system displays an error message and prompts correction. | |

### Admin - Delete an Existing Water Plant

| **Identifier** | | Admin-005 |
| --- | --- | --- |
| **Purpose** | | To remove a local mineral water plant from the network. |
| **Pre-conditions** | | 1. Admin is logged into the administrative dashboard |
| **Post-conditions** | | The selected water plant is no longer part of the MashkiOnline network and all associated data is removed from the system. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | Admin navigates to the "Water Plant Management" section. | |
| 2. | System displays a list of all mineral water plants. | |
| 3. | Admin selects the plant they wish to remove and clicks the "Delete" option. | |
| 4. | System prompts a confirmation message to ensure the admin's intent. | |
| 5. | Admin confirms the deletion. | |
| 6. | System removes the plant from the database and provides a success message. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | None | |
| **Step #** | **Exception Paths** | |
|  | 1. If there's a system error during deletion, an error message is displayed. | |

### Customer - Sign Up

| **Identifier** | | Customer-001 |
| --- | --- | --- |
| **Purpose** | | To register a new customer account for ordering water and accessing other services on MashkiOnline. |
| **Pre-conditions** | | 1. The customer has downloaded the MAshkiOnline application on their mobile device or is accessing MashkiOnline website for registration 2. The customer does not have an existing account. |
| **Post-conditions** | | A new customer account is successfully created and the customer is logged in. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | Customer selects the "Sign Up" option. | |
| 2. | System presents a registration form asking for details like name, email, password, phone number, etc. | |
| 3. | Customer fills out the form and submits. | |
| 4. | System validates the data and checks for unique email/username. | |
| 5. | Upon successful validation, the system creates a new account and sends a confirmation email/message. | |
| 6. | Customer verifies the account via the confirmation link/message. | |
| 7. | Customer is shown a prompt saying account is created | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | 1. Customer may choose to sign up using third-party authentication (e.g., Google, Facebook). | |
| **Step #** | **Exception Paths** | |
|  | 1. If provided email/username already exists or data is invalid, system displays an error message. | |

### 

### Customer - Log in

| **Identifier** | | Customer-002 |
| --- | --- | --- |
| **Purpose** | | To allow existing customers to access their accounts on MashkiOnline. |
| **Pre-conditions** | | 1. The customer has downloaded the MashkiOnline application on their mobile device or is accessing MashkiOnline website for registration 2. The customer have an existing account. |
| **Post-conditions** | | Customer successfully logs into their account and can access access features of MashkiOnline Platform. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | Customer selects the "Login" option. | |
| 2. | System presents a Login form asking for details like email/username and password, or using phone number and an OTP. | |
| 3. | Customer enter credentials | |
| 4. | System validates the credentails | |
| 5. | Upon successful validation, the system logs the customer in | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | 1. Customer may choose to log in using third-party authentication. (like Google, Facebook) 2. Customer selects the "Forgot Password" option if they can't remember their password. | |
| **Step #** | **Exception Paths** | |
|  | 1. If credentials are incorrect, the system displays an error message. | |

### 

### Customer - Add/Select Delivery Address

| **Identifier** | | Customer-003 |
| --- | --- | --- |
| **Purpose** | | To let customers add a new delivery address or choose from previously saved addresses. |
| **Pre-conditions** | | 1. Customer is logged into their account. 2. Customer is placing an order or updating profile settings. |
| **Post-conditions** | | The selected delivery address is saved/updated for the customer's order/profile. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | During order placement or profile update, the customer chooses to add/select a delivery address. | |
| 2. | System displays previously saved addresses. | |
| 3. | Customer can select an existing address or opts to add a new one. | |
| 4. | If adding, system presents a form for address details. | |
| 5. | Customer fills out the form and submits. | |
| 6. | System saves/updates the address for the customer's account. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | 1. Customer can delete an existing address. | |
| **Step #** | **Exception Paths** | |
|  | 1. If address details are incomplete or invalid, the system displays an error message. | |

### 

### Customer - Track Order Status

| **Identifier** | | Customer-004 |
| --- | --- | --- |
| **Purpose** | | To allow customers to monitor the current status of their water bottle orders. |
| **Pre-conditions** | | 1. Customer is logged into their account. 2. Customer has placed at least one order |
| **Post-conditions** | | Customer can view the real-time status of their orders. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | From the order history, the customer selects the "My orders" option. | |
| 2. | System displays a list of recent orders with their respective statuses. | |
| 3. | Customer selects a specific order to view detailed tracking. | |
| 4. | System provides real-time tracking details, such as "Preparing," "In Transit," "Delivered," etc. and other details like phone number of the delivery person is shared too | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | 1. Customer may select to view past orders and their final statuses. | |
| **Step #** | **Exception Paths** | |
|  | 1. If there's an error fetching order details, the system displays an error message. 2. If there are any exceptional delays, customers are told that using a message | |

### Customer - Give Rating/Feedback

| **Identifier** | | Customer-005 |
| --- | --- | --- |
| **Purpose** | | To allow customers to provide feedback and rate the service or specific water delivery. |
| **Pre-conditions** | | 1. Customer is logged into their account. 2. Customer has placed at least one order and it was completed |
| **Post-conditions** | | Feedback and rating have been submitted and saved to the system. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | After receiving the order or from the order history, the customer selects the "Rate/Feedback" option. | |
| 2. | System prompts the customer to rate the service on a scale (e.g., 1 to 5 stars). | |
| 3. | System offers an optional text box for detailed feedback. | |
| 4. | Customer provides a rating and optionally enters feedback. | |
| 5. | System saves the rating and feedback. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | 1. Customer may choose to provide feedback without rating. | |
| **Step #** | **Exception Paths** | |
|  | 1. If there's a system error saving the feedback/rating, an error message is displayed. | |

### 

### Customer - Place an Order

| **Identifier** | | Customer-006 |
| --- | --- | --- |
| **Purpose** | | To allow customers to order water bottles for delivery. |
| **Pre-conditions** | | 1. Customer is logged into their account. 2. Customer has a valid delivery address. |
| **Post-conditions** | | An order is placed successfully and is now pending for processing and delivery. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | Customer navigates to the order page. | |
| 2. | System displays water bottle options, quantities, prices, and delivery time slots. | |
| 3. | Customer selects the desired quantity and other order preferences. | |
| 4. | System shows a summary including total cost. | |
| 5. | Customer confirms and proceeds to payment | |
| 6. | After successful payment/confirmation, the system acknowledges the order and updates its status to "Order Received." | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | 1. Customer can choose "Order Now" for immediate delivery. 2. Customer can choose "Order for Later" to schedule a delivery. 3. Customer can modify quantities or add special instructions before confirming. | |
| **Step #** | **Exception Paths** | |
|  | 1. If payment fails or there's a system error during order placement, an error message is displayed and order might be put on hold | |

### 

### Transporter - Receive Orders

| **Identifier** | | Transporter-001 |
| --- | --- | --- |
| **Purpose** | | To allow transporters to view and accept new delivery orders assigned to them. |
| **Pre-conditions** | | 1. Transporter is logged into their dedicated interface or app. 2. There are new orders assigned to the transporter by the system based on location and availability. |
| **Post-conditions** | | Transporter has successfully viewed and accepted the assigned orders, ready for delivery. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The transporter sees a notification or a list of new orders assigned. | |
| 2. | Transporter views details of each order, including destination, quantity, and any special instructions. | |
| 3. | System confirms the acceptance and updates the order status as "Accepted by [Transporter Name]." | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | 1. Transporter can choose to Navigate for the order that has the highest priority 2. The list order may be updated if new orders are received by this transporter and the transporter has more than two orders. | |
| **Step #** | **Exception Paths** | |
|  | 1. If there's a system error during the order acceptance process, an error message is displayed. | |

### 

### Transporter - Update Order Status

| **Identifier** | | Transporter-002 |
| --- | --- | --- |
| **Purpose** | | To allow transporters to update the status of orders they're delivering, keeping the system and customers informed in real-time. |
| **Pre-conditions** | | 1. Transporter is logged into their interface or app. 2. Transporter has orders ready for delivery. |
| **Post-conditions** | | Order status has been updated, reflecting the current phase of the delivery process. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | Transporter navigates to the "Ongoing Deliveries" section. | |
| 2. | A list of current orders is displayed with options to update status. | |
| 3. | As the transporter progresses with the delivery, they select the relevant status update, such as "En Route," "Arrived at Destination," or "Delivered." | |
| 4. | System instantly updates the order status, making it visible to customers and the central system. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | 1. Transporter can can select tags while updating the status, e.g., "Traffic delay." | |
| **Step #** | **Exception Paths** | |
|  | 1. If there's a system error during the order status update, an error message is displayed. | |

### 

### Transporter - Receive Delivery Routes

| **Identifier** | | Transporter-003 |
| --- | --- | --- |
| **Purpose** | | To provide transporters with the most efficient routes for their deliveries, optimizing time and fuel. |
| **Pre-conditions** | | 1. Transporter is presented with an optimized delivery route for their orders. |
| **Post-conditions** | | Order status has been updated, reflecting the current phase of the delivery process. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | Upon receivingorders, or on demand, the system calculates the best route considering traffic, distance, and order sequence. | |
| 2. | Transporter selects "View Map/ Navigate" for their deliveries. | |
| 3. | System displays a detailed map with the optimized route for the current order in progress with each delivery point added as a stop in the correct order | |
| 4. | Transporter follows the suggested route for efficient deliveries. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | 1. Transporter can adjust or override the suggested route based on personal knowledge or unforeseen circumstances. | |
| **Step #** | **Exception Paths** | |
|  | 1. If there's a system error in generating or displaying the route, an error message is displayed. | |

### 

### Local Mineral Water Plants - Check Inflows and Outflows

| **Identifier** | | Local Mineral Water Plants-001 |
| --- | --- | --- |
| **Purpose** | | To allow plant managers to monitor the quantity of Bottles coming into the plant (inflows) and the amount being dispatched for orders (outflows). |
| **Pre-conditions** | | 1. Plant manager is logged into their dedicated dashboard.. |
| **Post-conditions** | | Plant manager has an updated view of the water inflows and outflows. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | Plant manager navigates to the "Inflows and Outflows" section. Clicks the graph which shows this is pictorial form. | |
| 2. | System displays a visual representation (graphs, charts) and data about current and historical inflows and outflows. | |
| 3. | Plant manager reviews the data for insights and inventory management. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | 1. Plant manager can download the inflow and outflow data for external analysis. 2. Plant manager can view different time frames (daily, weekly, monthly) for inflow and outflow data. | |
| **Step #** | **Exception Paths** | |
|  | 1. If there's a system error retrieving data, an error message is displayed. | |

### 

### Local Mineral Water Plants - Update Production Capacity

| **Identifier** | | Local Mineral Water Plants-002 |
| --- | --- | --- |
| **Purpose** | | To allow plant managers to update the production capacity of their plant as it changes due to maintenance, upgrades, or other factors. It is important because it the system needs to know the plant’s capacity to tell the estimated delivery time |
| **Pre-conditions** | | 1. Plant manager is logged into their dedicated dashboard.. |
| **Post-conditions** | | Plant production capacity is updated in the system and this may change the estimated delivery time calculation |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | Plant manager navigates to the "Production Capacity" section. | |
| 2. | System displays the current set production capacity. | |
| 3. | Plant manager enters the new capacity and saves the changes. | |
| 4. | System confirms the update. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | 1. Plant manager can revert to previous production capacity settings if required. 2. Plant manager can set future changes in production capacity to be updated automatically on a specified date. | |
| **Step #** | **Exception Paths** | |
|  | 1. If there's a system error during update, an error message is displayed. | |

### 

### Local Mineral Water Plants - Update Inventory

| **Identifier** | | Local Mineral Water Plants-003 |
| --- | --- | --- |
| **Purpose** | | To let plant managers update the number of ready-to-deliver water bottles in their inventory. |
| **Pre-conditions** | | 1. Plant manager is logged into their dedicated dashboard.. |
| **Post-conditions** | | The plant's inventory count is updated in the system. That how many bottles are ready to be delivered |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | Plant manager navigates to the "Inventory" section. | |
| 2. | System displays the current inventory count. | |
| 3. | Plant manager enters the updated count and saves the changes. | |
| 4. | System confirms the inventory update. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | 1. Plant manager can set periodic reminders to update inventory at regular intervals. | |
| **Step #** | **Exception Paths** | |
|  | 1. If there's a system error during update, an error message is displayed. | |

### 

### 

### Local Mineral Water Plants - Get a list of ongoing orders

| **Identifier** | | Local Mineral Water Plants-003 |
| --- | --- | --- |
| **Purpose** | | To let plant managers update the number of ready-to-deliver water bottles in their inventory. |
| **Pre-conditions** | | 1. Plant manager is logged into their dedicated dashboard.. |
| **Post-conditions** | | The plant's inventory count is updated in the system. That how many bottles are ready to be delivered |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | Plant manager navigates to the "Inventory" section. | |
| 2. | System displays the current inventory count. | |
| 3. | Plant manager enters the updated count and saves the changes. | |
| 4. | System confirms the inventory update. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | 1. Plant manager can filter the order list based on various criteria like order status, delivery date, or volume. 2. Plant manager can export the ongoing order list for further analysis or for sharing with other team members. | |
| **Step #** | **Exception Paths** | |
|  | 1. If there's a system error retrieving the list, an error message is displayed. | |

### 

### Testing Laboratories - Receive Testing Instructions

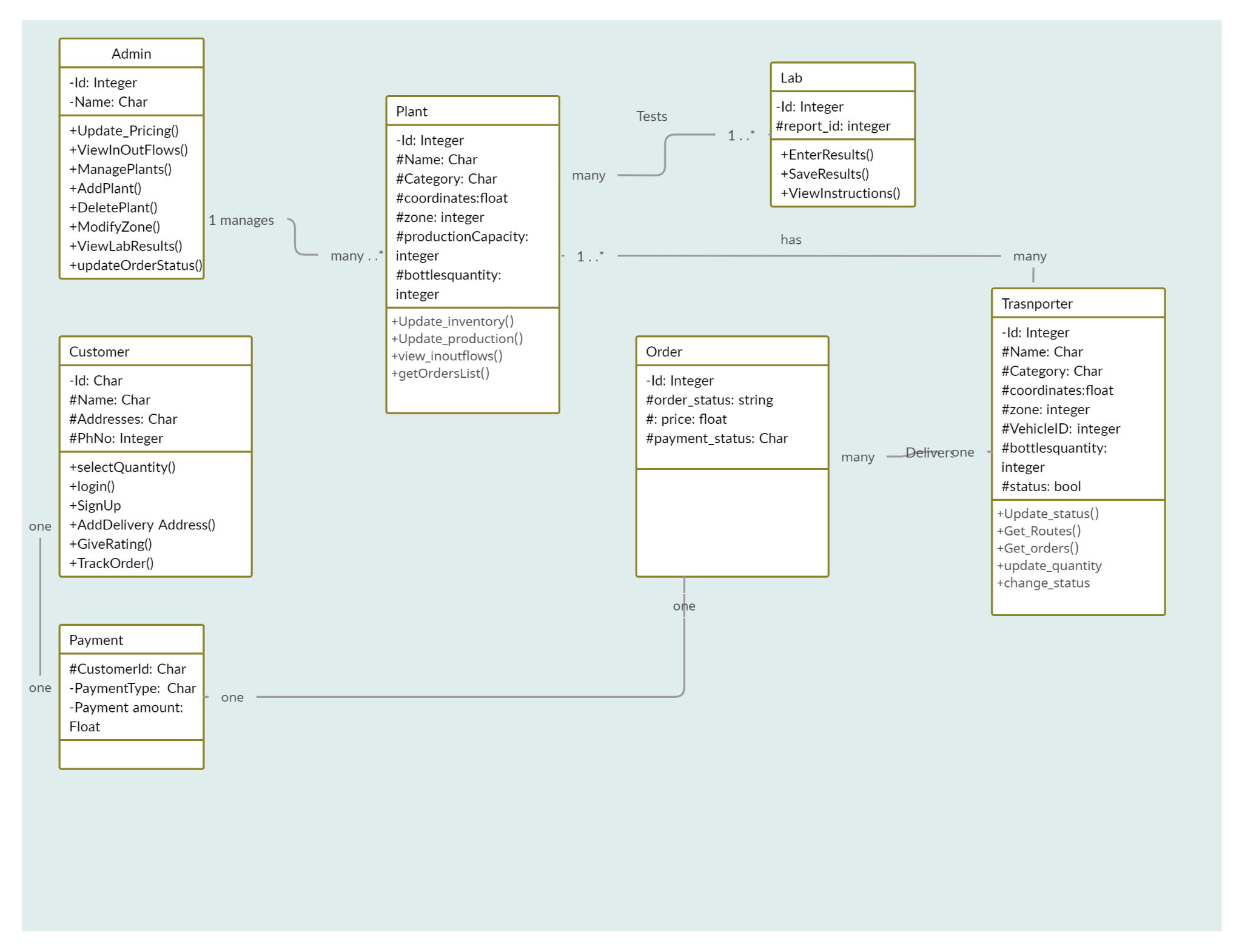
| **Identifier** | | Testing Laboratories-001 |
| --- | --- | --- |
| **Purpose** | | To provide laboratories with specific instructions regarding the parameters and methods for water quality testing. |
| **Pre-conditions** | | 1. Laboratory technician or representative is logged into their interface or app. 2. Laboratory is affiliated with MashkiOnline and has been assigned water samples for testing. |
| **Post-conditions** | | Laboratory has received and understood the detailed instructions for conducting the water test. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | Upon being assigned a water sample, the system a list of the tests that are assigned to the laboratory | |
| 2. | Laboratory technician selects "View Instructions" for the specified test | |
| 3. | System displays specific parameters to be tested, like which plant/home, and any other relevant instructions. | |
| 4. | Laboratory technician follows the provided instructions to conduct the test. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | 1. Laboratory technician can save or print the instructions for offline reference. | |
| **Step #** | **Exception Paths** | |
|  | 1. If there's a system error in displaying the list pof tests to be comducted, an error message is displayed. | |

### Testing Laboratories - Enter Test results

| **Identifier** | | Testing Laboratories-002 |
| --- | --- | --- |
| **Purpose** | | To allow laboratories to input the results of the water quality tests into the system. |
| **Pre-conditions** | | 1. Laboratory technician is logged into their interface. 2. Water testing has been completed as per the instructions. |
| **Post-conditions** | | Test results are stored and can be accessed by MashkiOnline for quality assurance and feedback to plants and customers. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | Laboratory technician navigates to the "Enter Test Results" section. | |
| 2. | System provides an interface for entering test parameters and their respective results. | |
| 3. | Laboratory technician inputs the results of each parameter tested. | |
| 4. | After entering all results, technician submits the data. | |
| 5. | System confirms the successful entry of test results. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | 1. Laboratory technician can save results midway and continue entering them at a later time. 2. Laboratory technician can attach additional files (like pdf of the test) or notes along with the test results. | |
| **Step #** | **Exception Paths** | |
|  | 1. If there's a system error during result entry or submission, an error message is displayed. | |

### 

## Class Diagram



### Admin:

At the heart of the MashkiOnline system is the Admin, responsible for the holistic management and oversight of the platform. With unique identification attributes like name and ID, the admin's functions span from dynamically updating product pricing to ensure competitive market rates, to overseeing the flow of products across the platform.

They maintain an eagle-eye view of product movement with functions like viewInflowsOutflows(). Furthermore, as MashkiOnline grows, the admin has the ability to manage water plants by adding or removing them. Adjusting service zones, accessing lab results, and monitoring the status of orders are also within the admin's purview, ensuring efficient operations and high-quality service delivery.

### Customer:

The most important actor of MashkiOnline is its customer base, uniquely identified through attributes like ID, name, and phone number. They navigate the platform with ease, selecting the quantity of products they desire, securely accessing their profiles through the login mechanism, and even onboarding through the signup process. Their flexibility is evident as they can specify delivery locations, offer valuable feedback through ratings, and stay updated by tracking their orders in real-time.

### Plant:

The water plants, the very source of the products, have been meticulously integrated into MashkiOnline. Identified by attributes like name, coordinates, and production capacity, these plants not only supply products but also update their inventory and production capacities as needed. They have a window into the system through functions that allow them to view inflows and outflows and even get a list of ongoing orders, ensuring seamless alignment with demand.

### Labs:

Ensuring quality standards, the Lab entity plays a pivotal role in MashkiOnline's commitment to excellence. Identified uniquely, the labs are equipped to enter and save test results, ensuring the highest quality of products. Moreover, they can view instructions, ensuring standard procedures are maintained, solidifying the platform's reputation for safety and quality.

### Payment:

Serving as the financial backbone, the Payment entity captures each transaction's essence through unique payment IDs and the associated monetary amount. It ensures that every financial exchange on MashkiOnline is tracked, secured, and accounted for.

### Order:

The Order entity is a testament to the platform's customer-centric design. With unique order IDs, customers can keep a tab on their order status, know the exact price, and stay informed about their payment status. It acts as a bridge between customer intent and service fulfillment.

### Transporter:

Essential to fulfilling orders are the transporters, each uniquely identified with details like name, category, and zone. Their role goes beyond just transportation; they have specific operational details like vehicle IDs and bottle quantities, and their current status (active or inactive) ensuring that the customer gets a smooth experience.

## Sequence Diagrams

Draw sequence diagrams of 10 core use cases. Draw the diagrams using standard UML notation

### Admin - Adjust Pricing

## 

## 

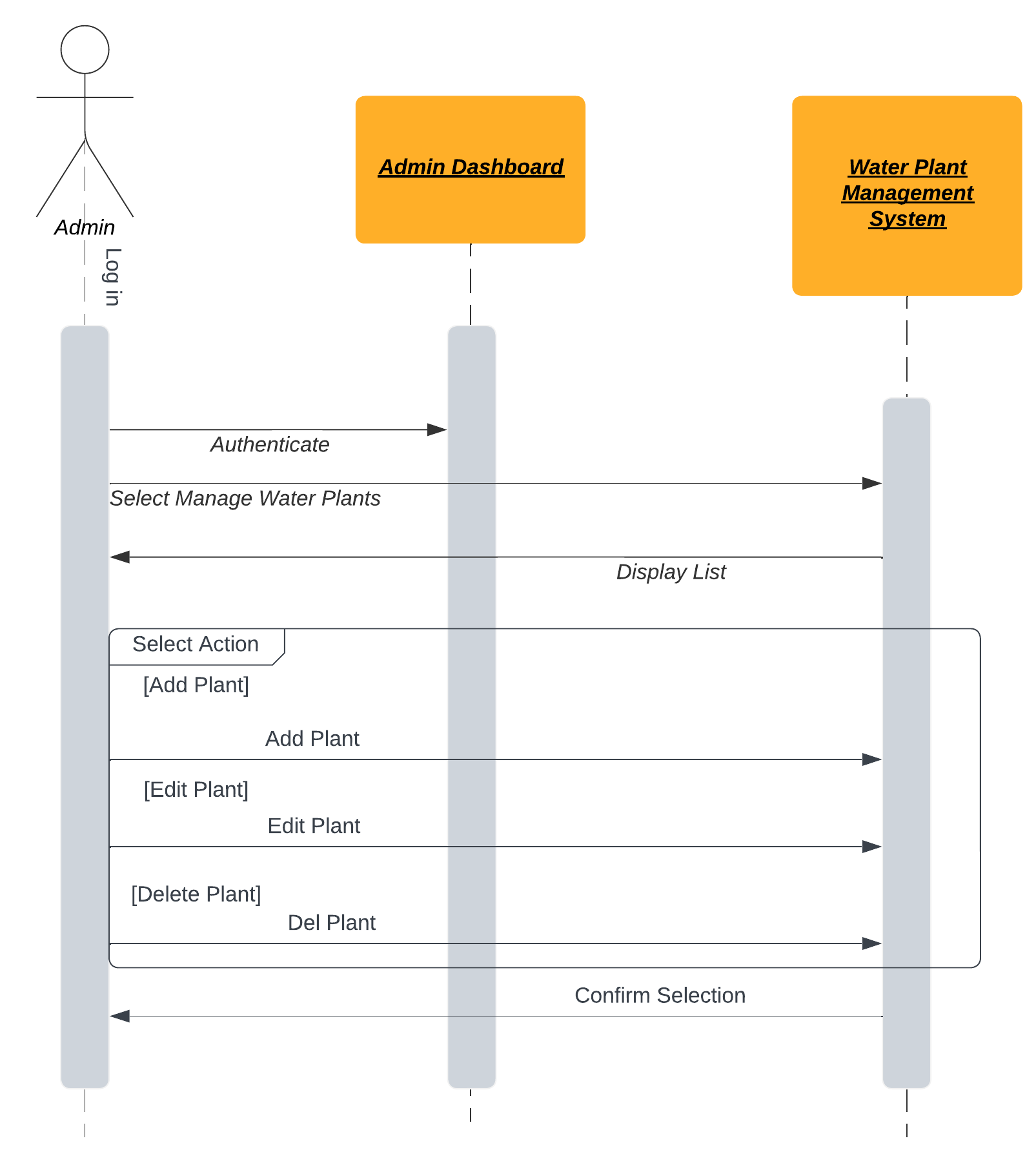
### 

### 

### Admin - View Inflows and Outflows

## 

### Admin - Manage Water Plants (Add and Delete Included)



## 

### 

### Admin - Add a New Water Plant

Included in Admin - Manage Water Plant

### Admin - Delete an Existing Water Plant

Included in Admin - Manage Water Plant

### Customer - Sign Up

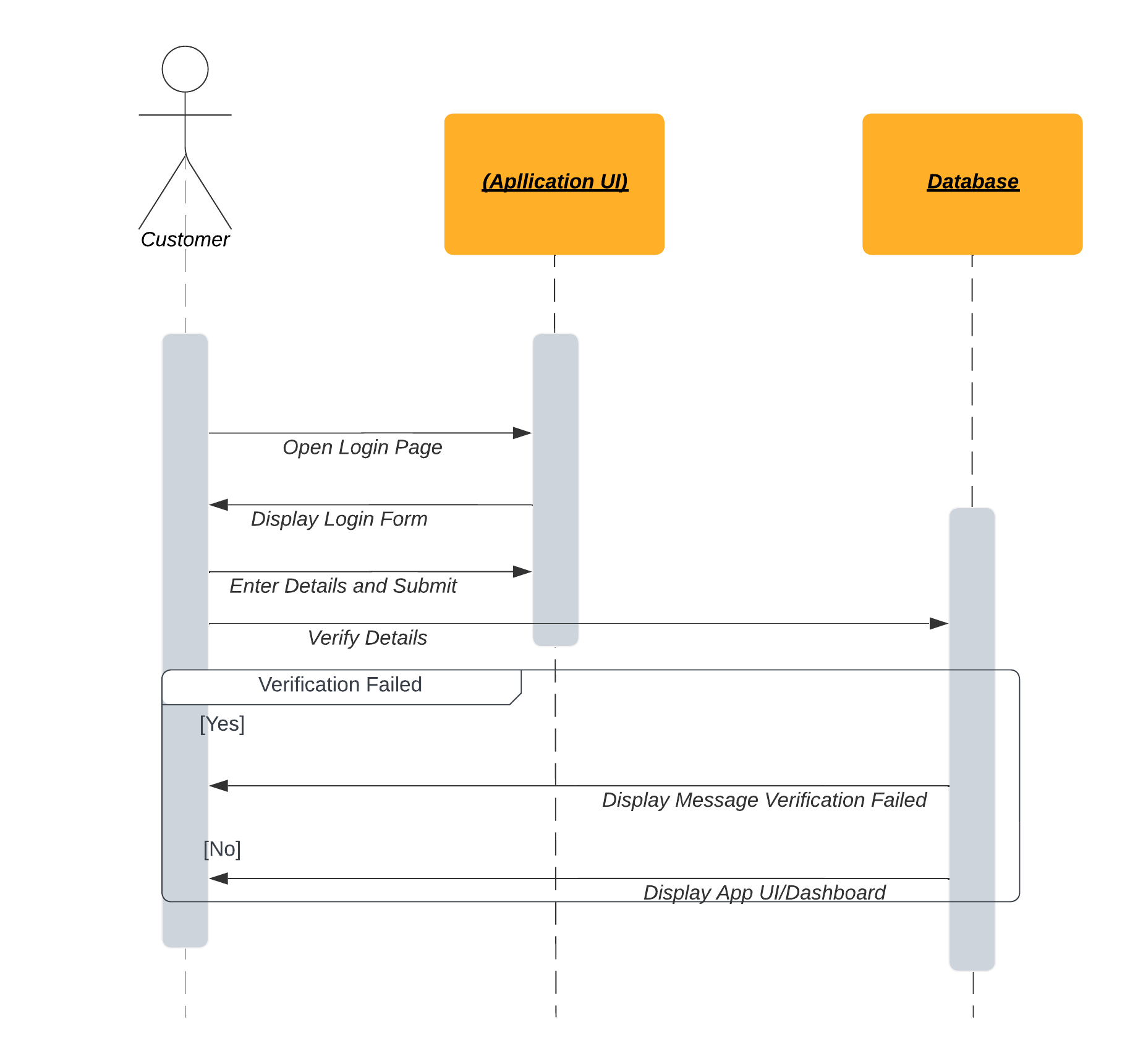
## 

## 

## 

### 

### Customer - Log in



## 

## 

## 

## 

### Customer - Add/Select Delivery Address

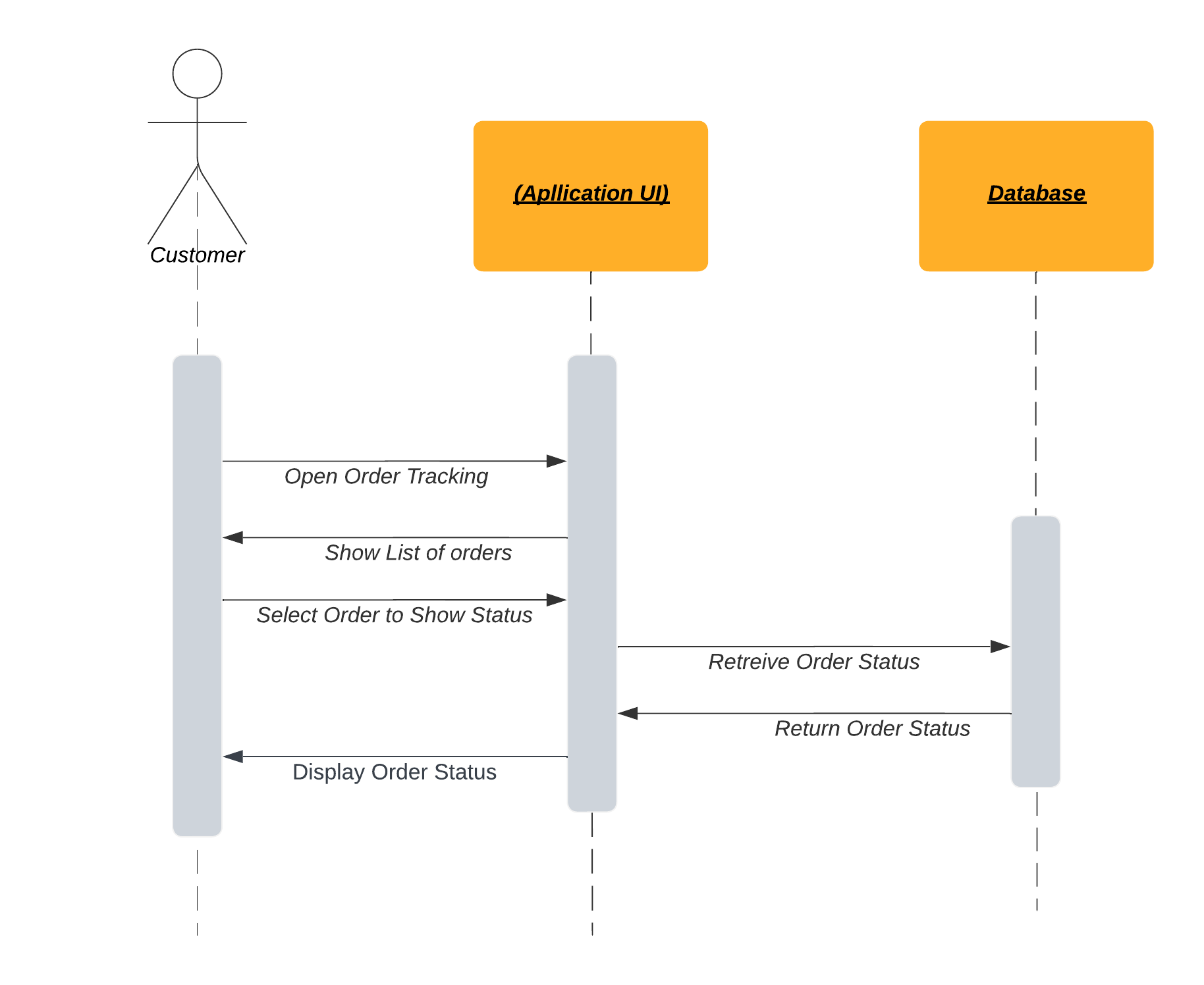
## 

## 

## 

## 

### Customer - Track Order Status



## 

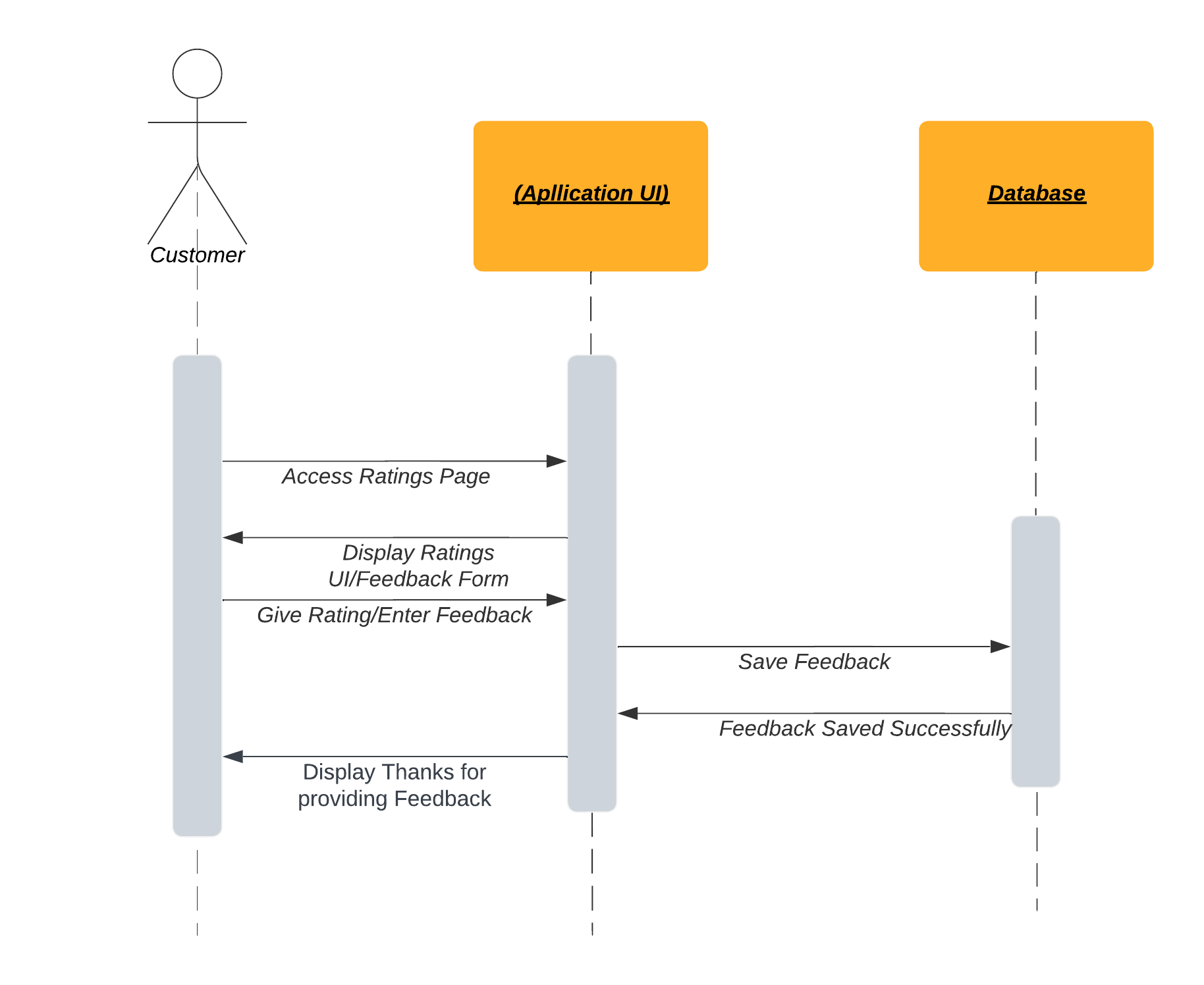
## 

## 

### 

### 

### Customer - Give Rating/Feedback



## 

## 

## 

### Customer - Place an Order

## 

## 

## 

## 

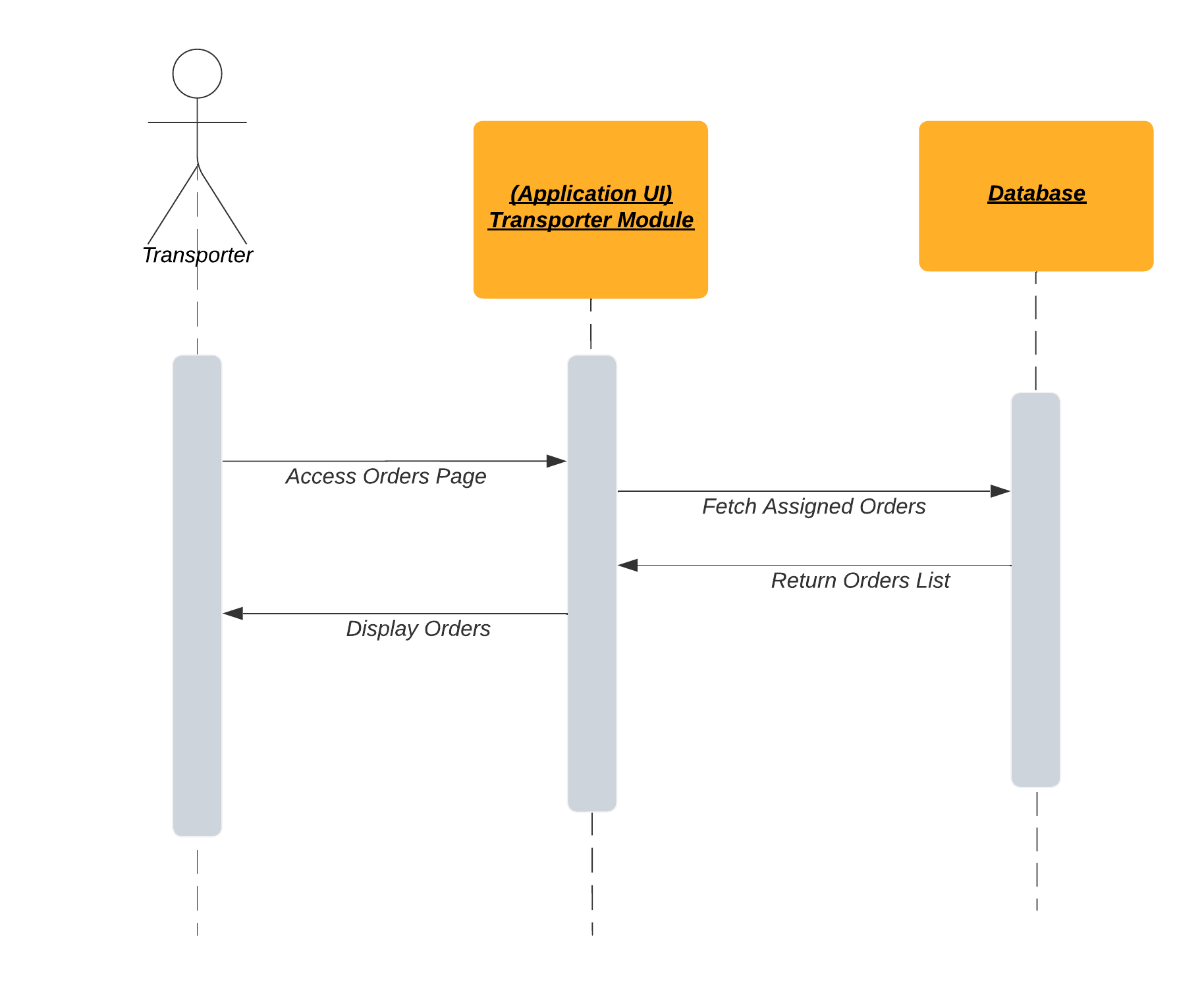
## 

## 

## 

## 

### Transporter - Receive Orders



## 

## 

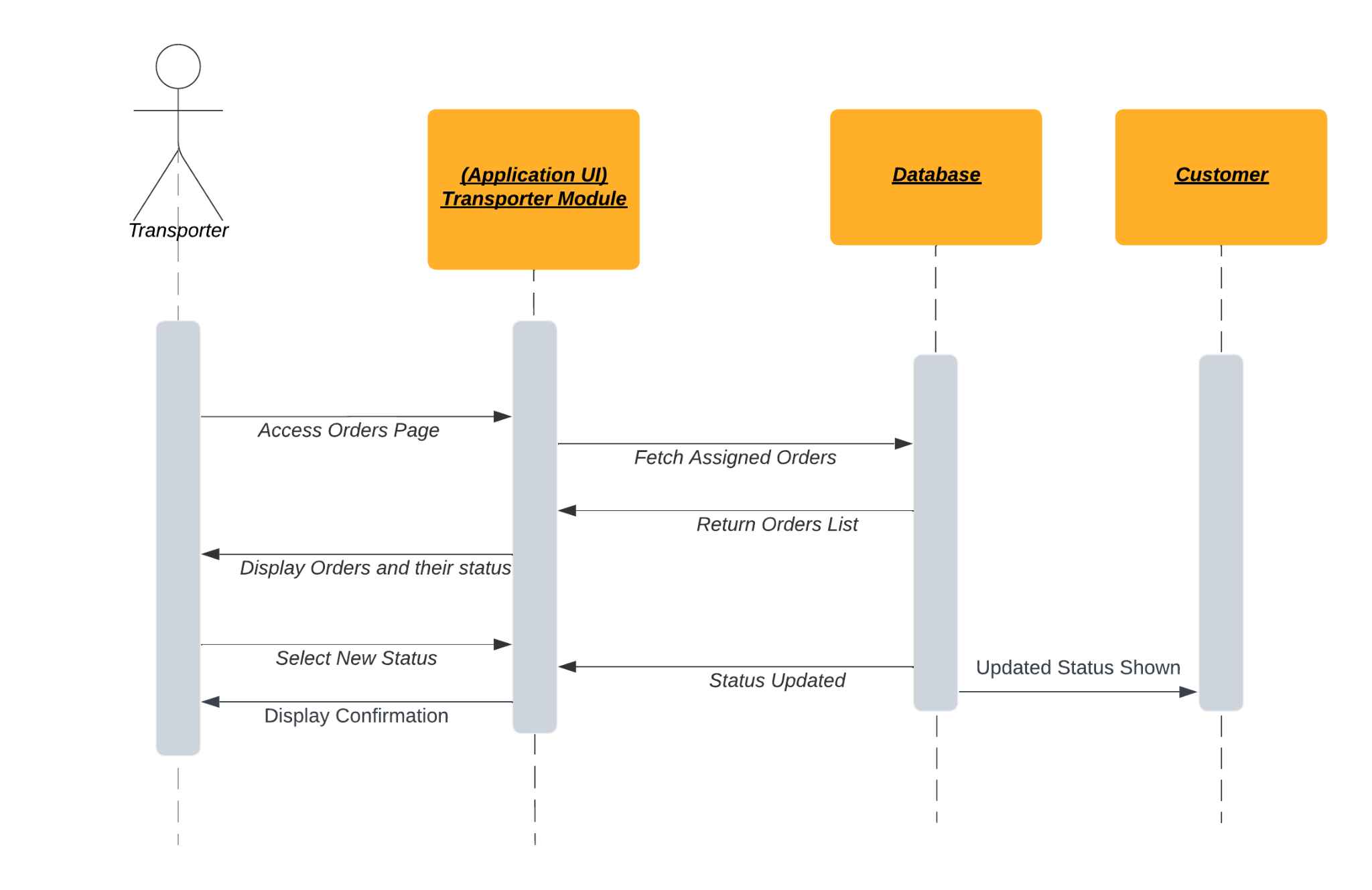
## 

## 

## 

### 

### Transporter - Update Order Status



## 

## 

## 

## 

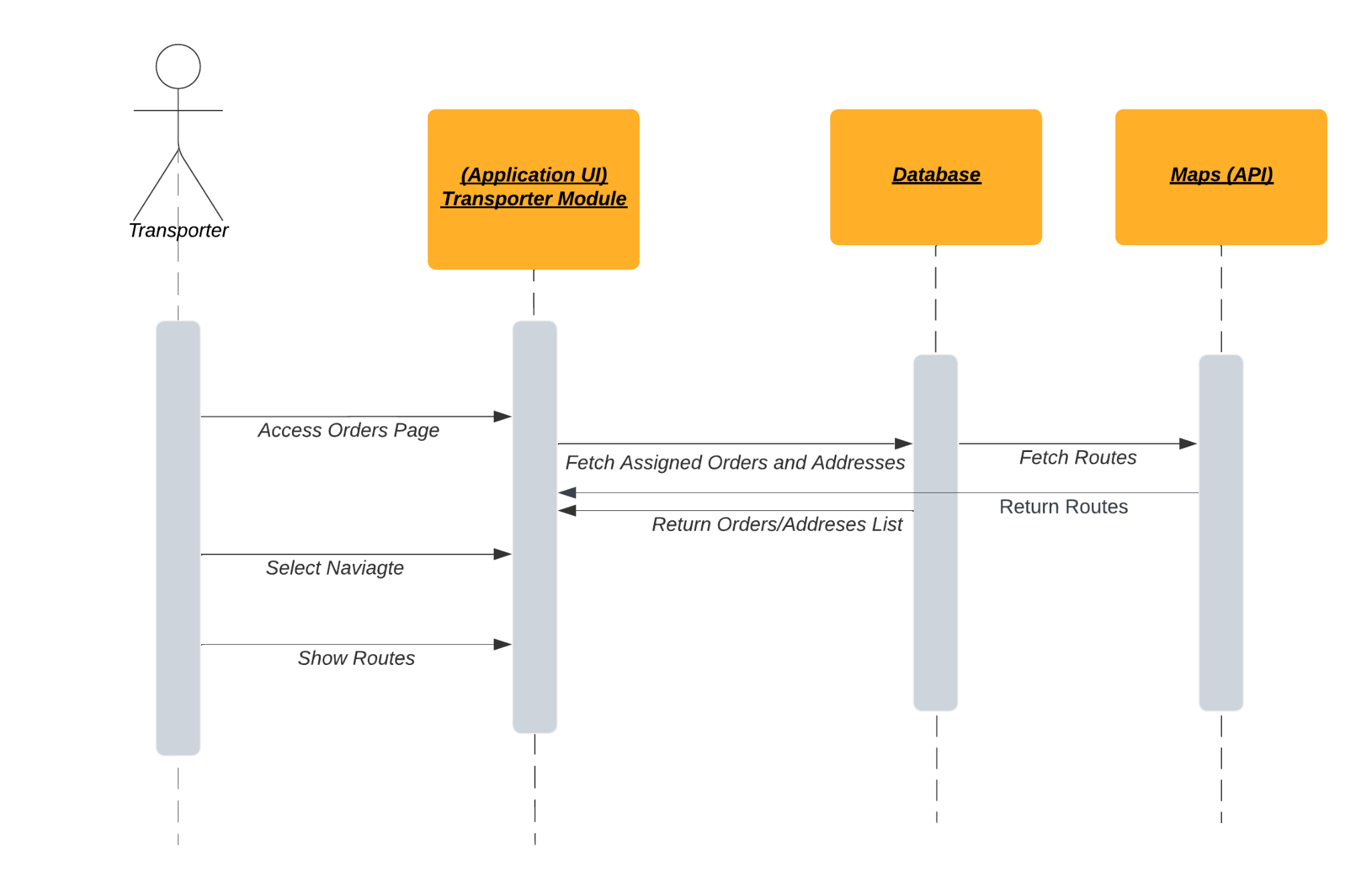
## 

## 

## 

### 

### Transporter - Receive Delivery Routes



## 

## 

## 

## 

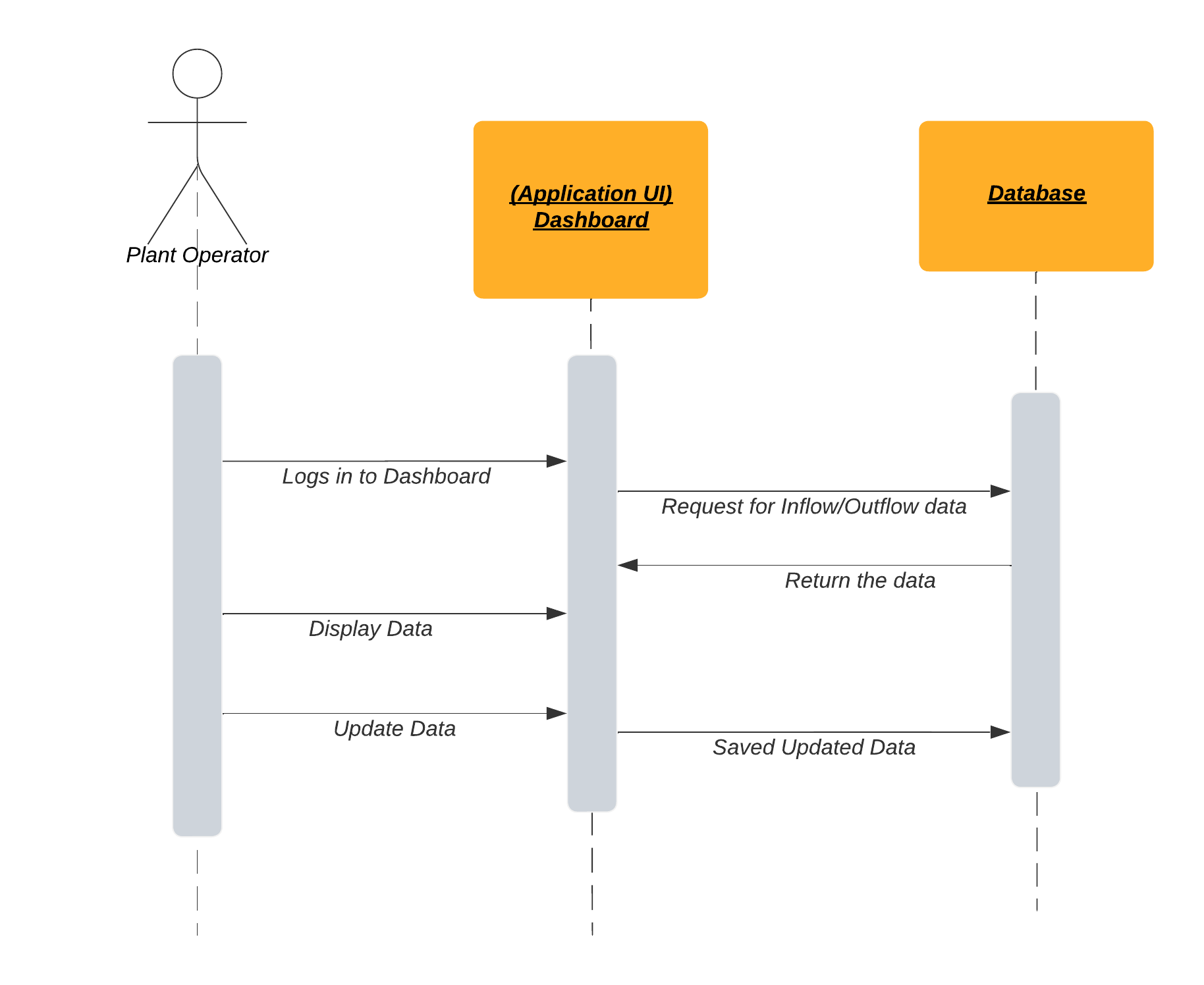
## 

## 

## 

### 

### Local Mineral Water Plants - Check Inflows and Outflows



## 

## 

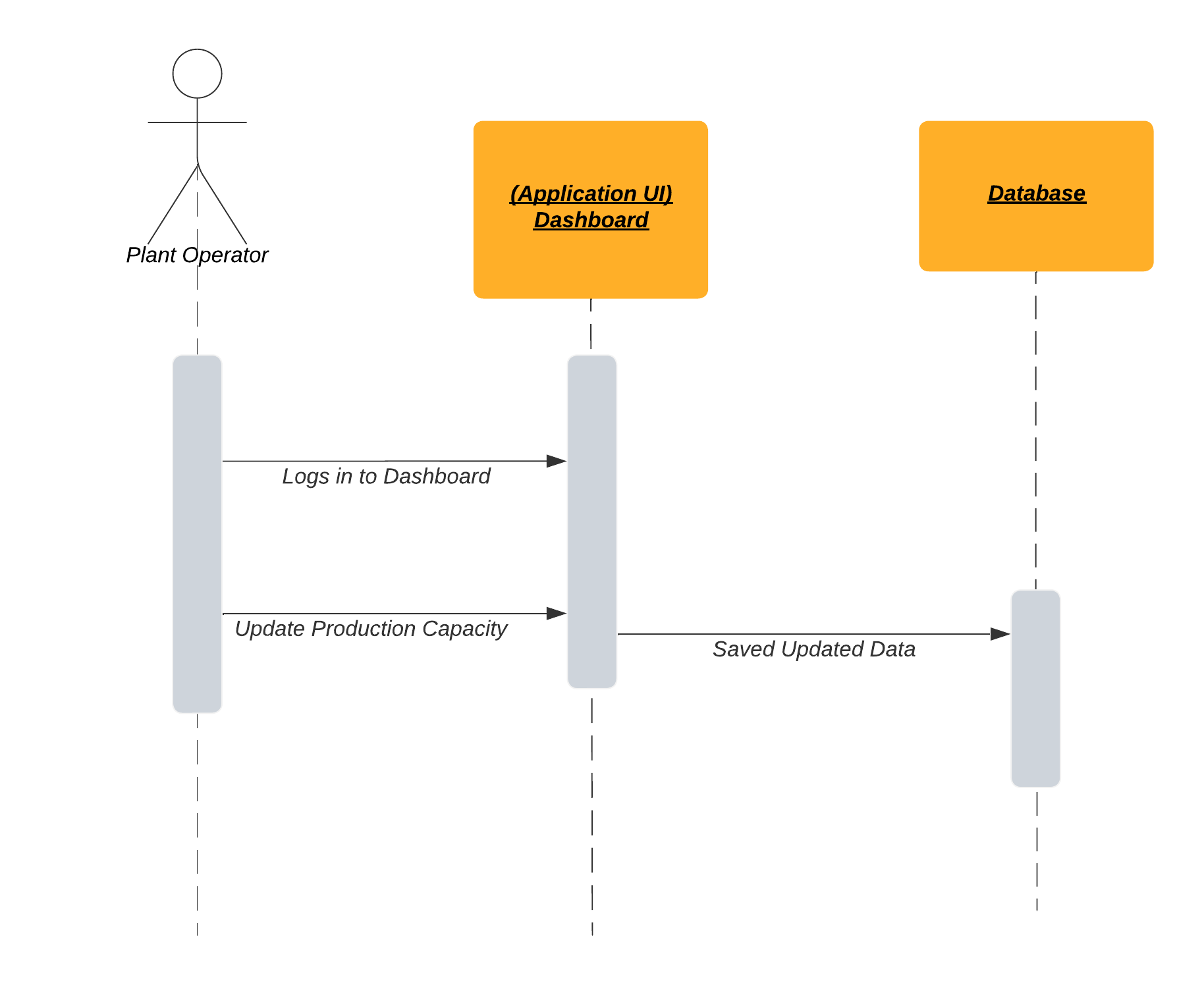
## 

## 

### 

### 

### Local Mineral Water Plants - Update Production Capacity



## 

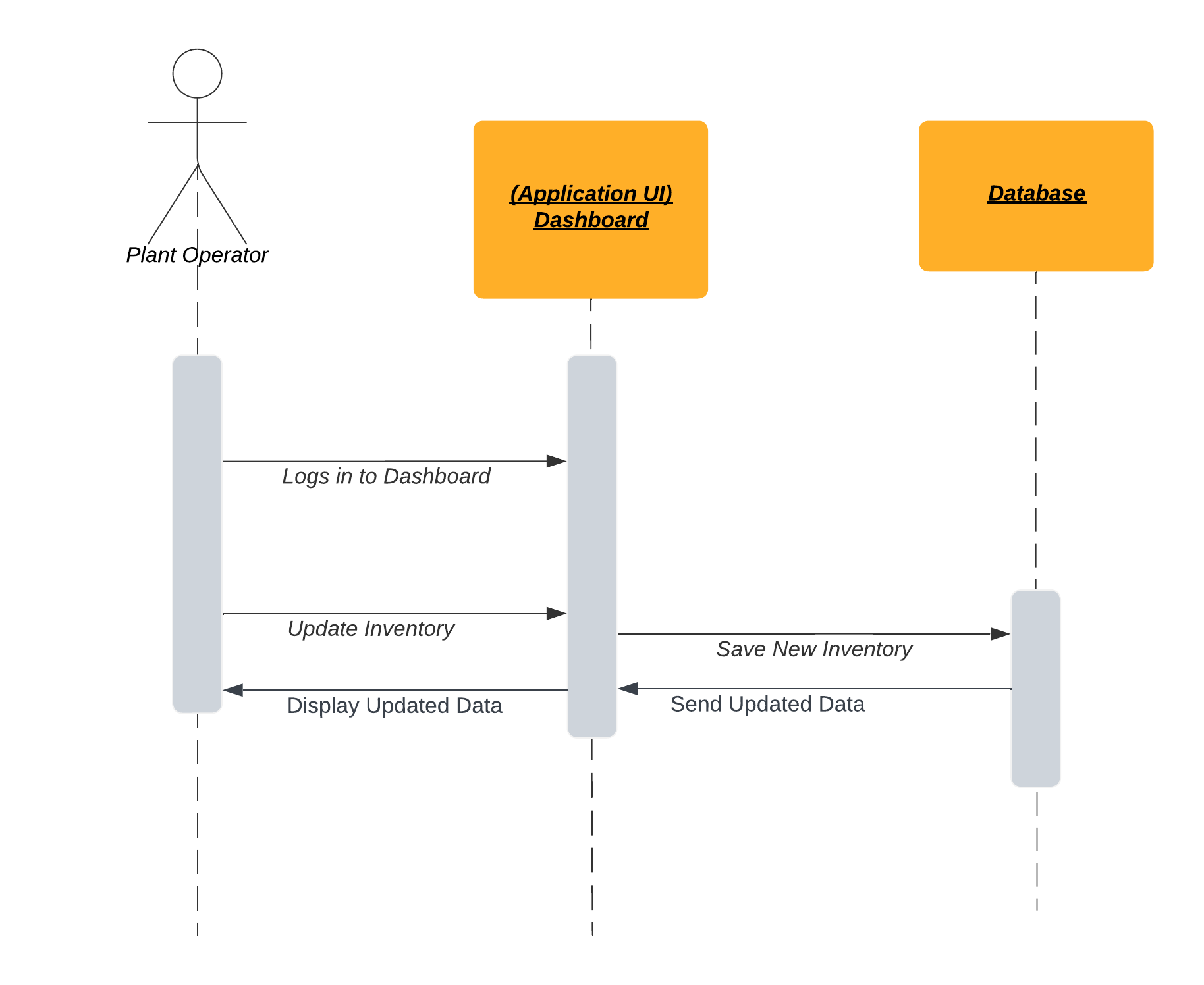
## 

## 

## 

### 

### Local Mineral Water Plants - Update Inventory



## 

## 

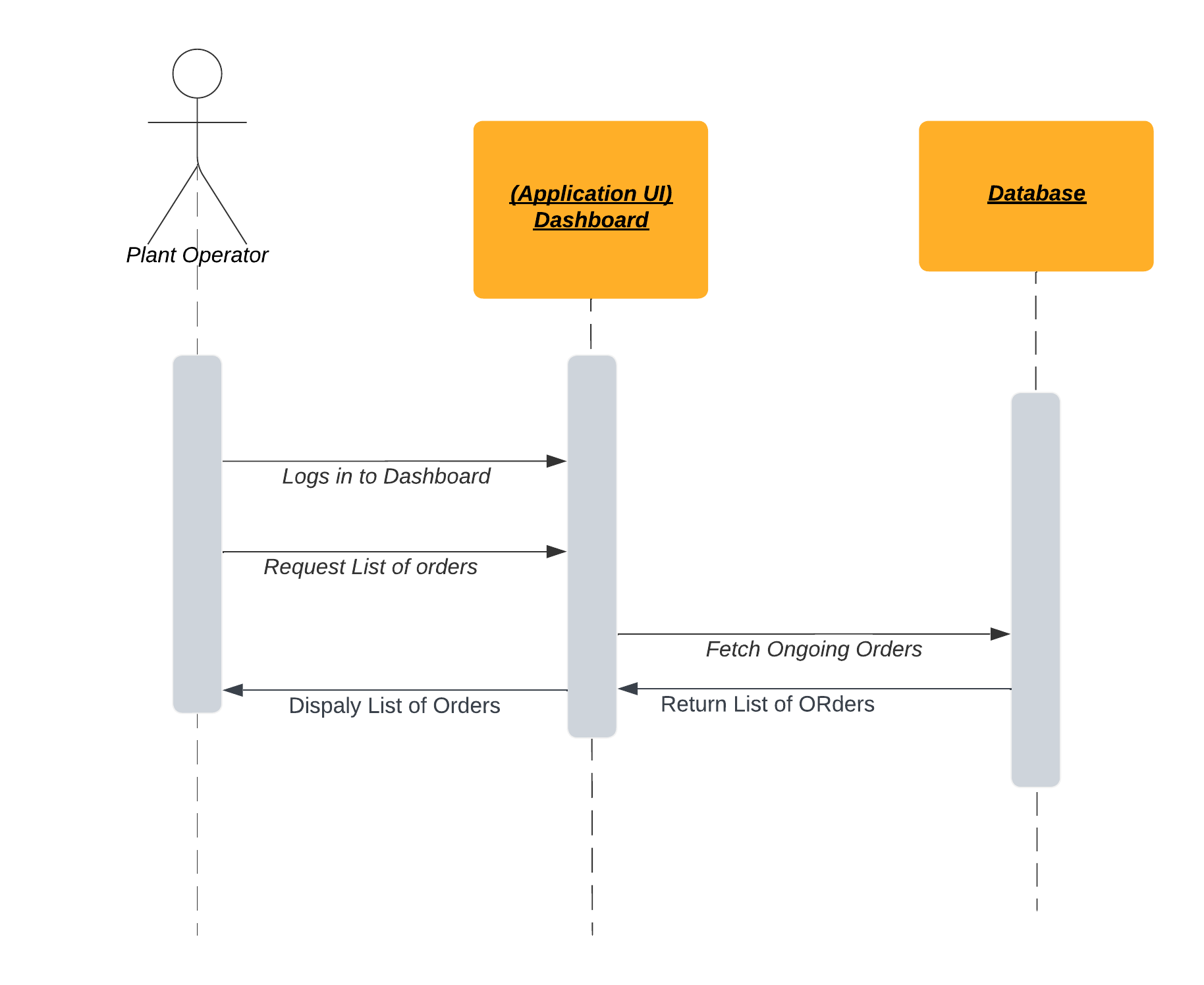
## 

## 

## 

### 

### Local Mineral Water Plants - Get a list of ongoing orders



## 

## 

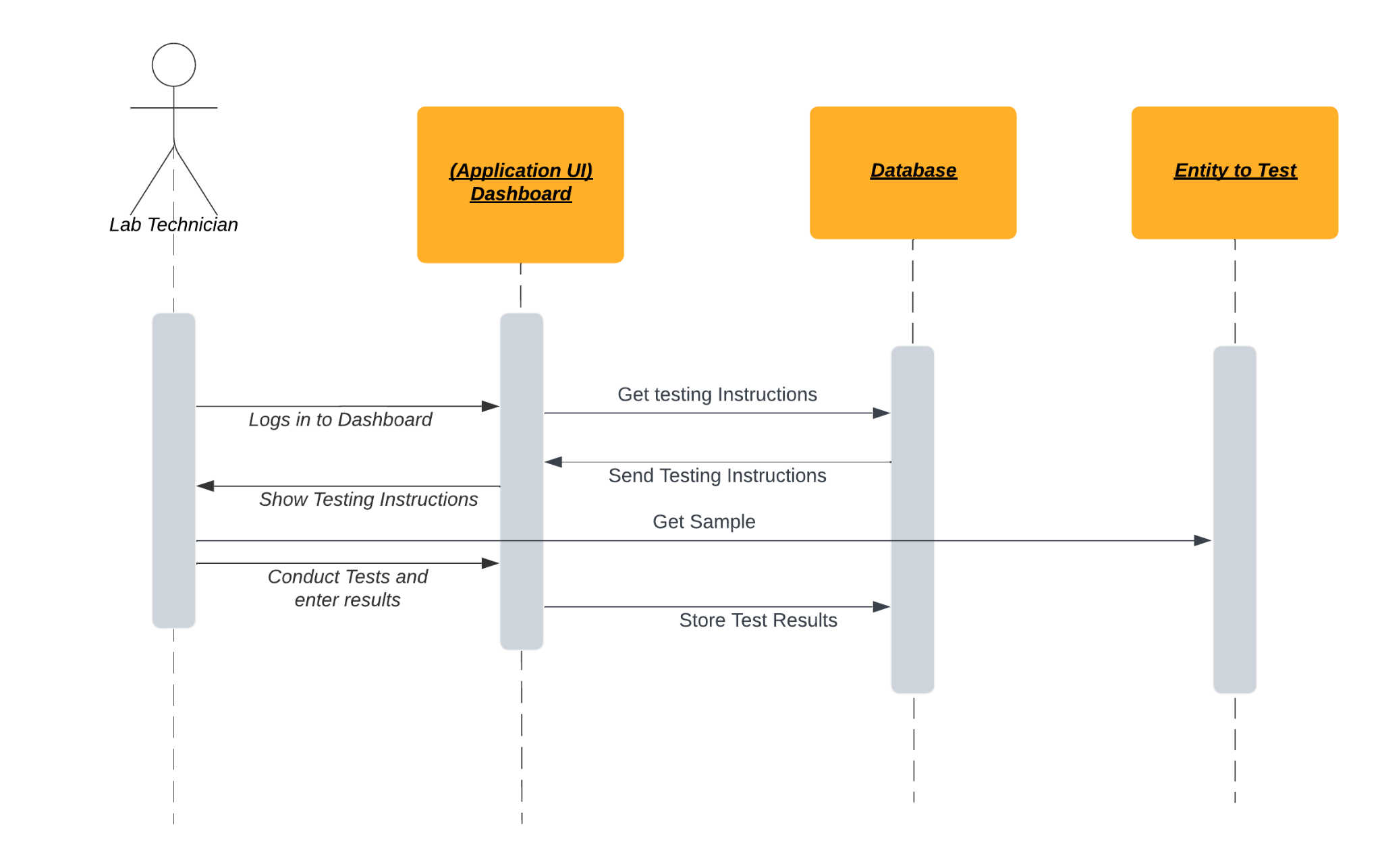
## 

## 

## 

### 

### Testing Laboratories - Receive Testing Instructions



### Testing Laboratories - Enter Test results

## 

# Software Development Methodology and Plan

Brief introduction of this chapter in a paragraph highlighting the content

## Software Process Selection

Discuss a pros and cons of waterfall and agile (scrum) processes in your own words.

Give proper justification of the software development process that you have used for your project.

### Waterfall

* 1. Pros:
     + Spending more time on earlier phases to prevent problems coming up later on. This is the reason why we spend more time in collecting requirements and documenting them properly.
     + It has a structure which helps as everyone knows what to do next.
     + It is easier to judge the progress of the project.
     + Documentation is given importance.
  2. Cons:
     + It is difficult to respond to changing requirements.
     + It is used for huge projects.
     + Sudden or unexpected risks will be difficult to judge and handle.
     + You get a working system at a later stage of the development process.

### Agile(Scrum)

* 1. Pros:
     + Can easily respond to changing requirements of customers.
     + The team can chose how to fulfill the responsibilities.
  2. Cons:
     + It has a lack of documentation as compared to Waterfall model.
     + Communication is heavily relied upon.
     + Settling contracts with customers can be difficult as the requirements keep changing.
     + If there are multiple stakeholders for the same project, then it becomes difficult to decide which requirement to prioritize first.

We would go with the Agile process for our project development.

Reasoning:

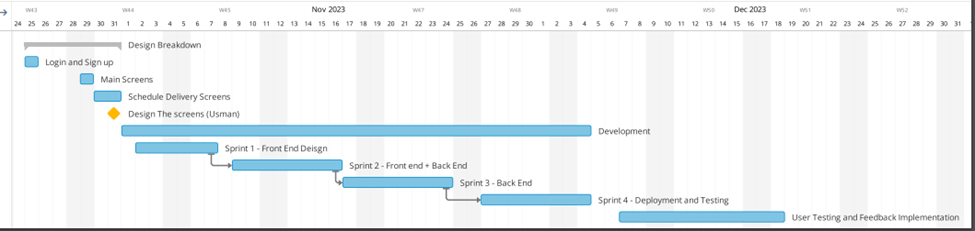
* With time our business logic will be changing so it will be better to go with the Agile development process as we can cater to changing requirements. This is the reason we can not go with the Waterfall development process as we expect many changing requirements. We expect to have changes in our design and processes and functions within our project. We will prefer having iterations as feedback would be important to change certain functions and processes. There can be huge losses due to defects and bugs. Developers have the required experience to develop this project. Rate of change of requirements is expected to be very high as discussed above. The team size is 5. The culture is adaptive to changes. There is slight pressure to develop early releases as feedback is important. Although communication is considered a drawback of Agile development processes, we will use it for our benefit as communication is an important aspect for successful completion of our project.

# 

## Gantt Chart

Draw a Gantt chart that illustrates your project’s schedule. The Gantt chart should show at least the following

* Tasks (tasks should not be too small or too large)
* Duration (in weeks)
* Milestones
* Team member names who have worked on each task.



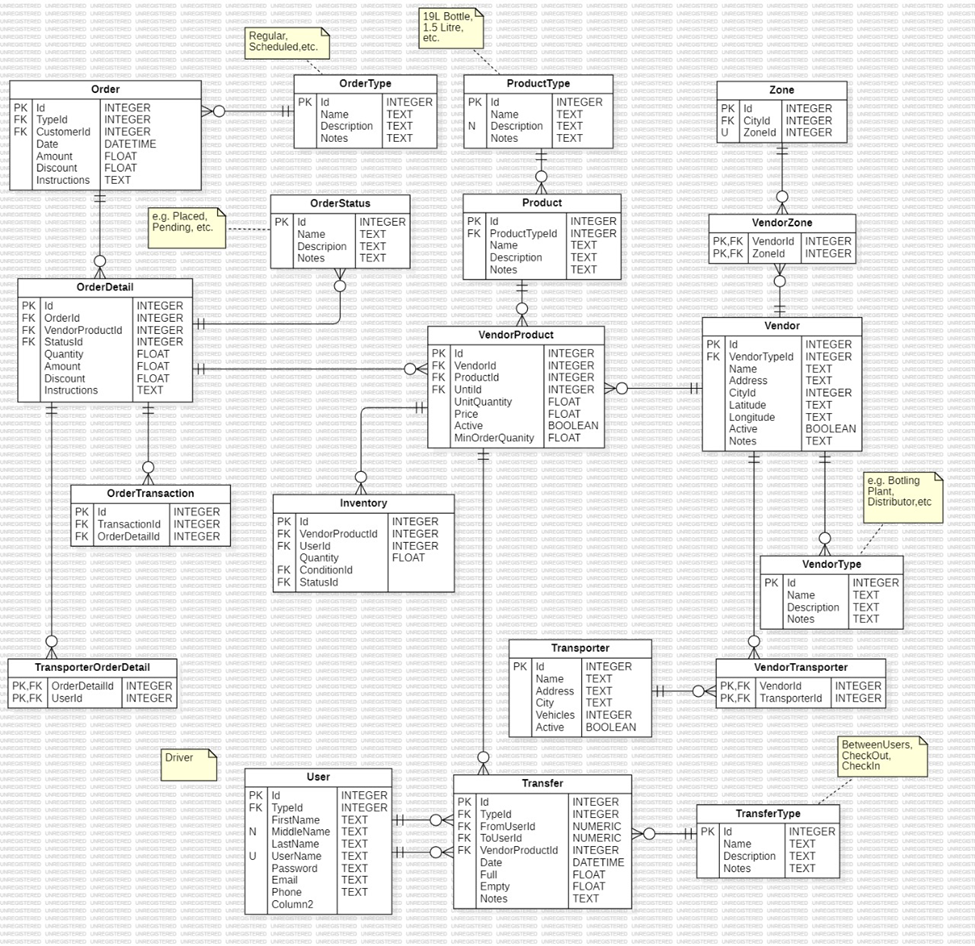
# Database Design and Web Services

Brief introduction of this chapter in a paragraph

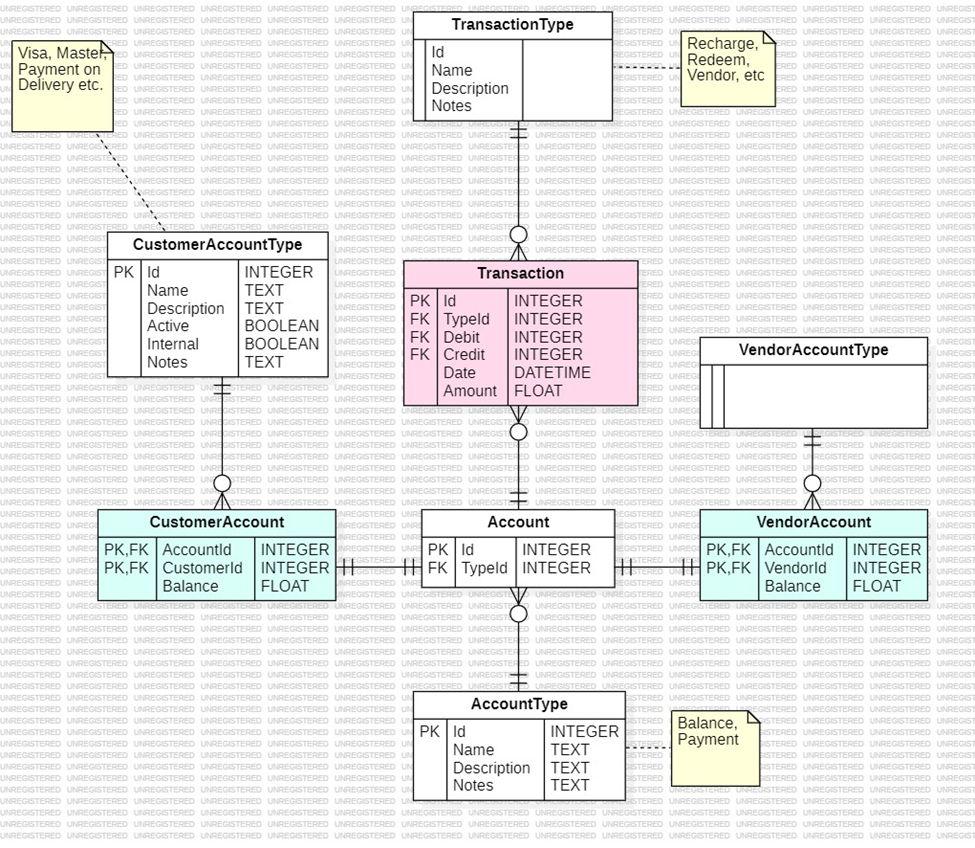
## Database Design

Create data model of your system. For example, E/R diagram. Give brief description of entities and data fields.

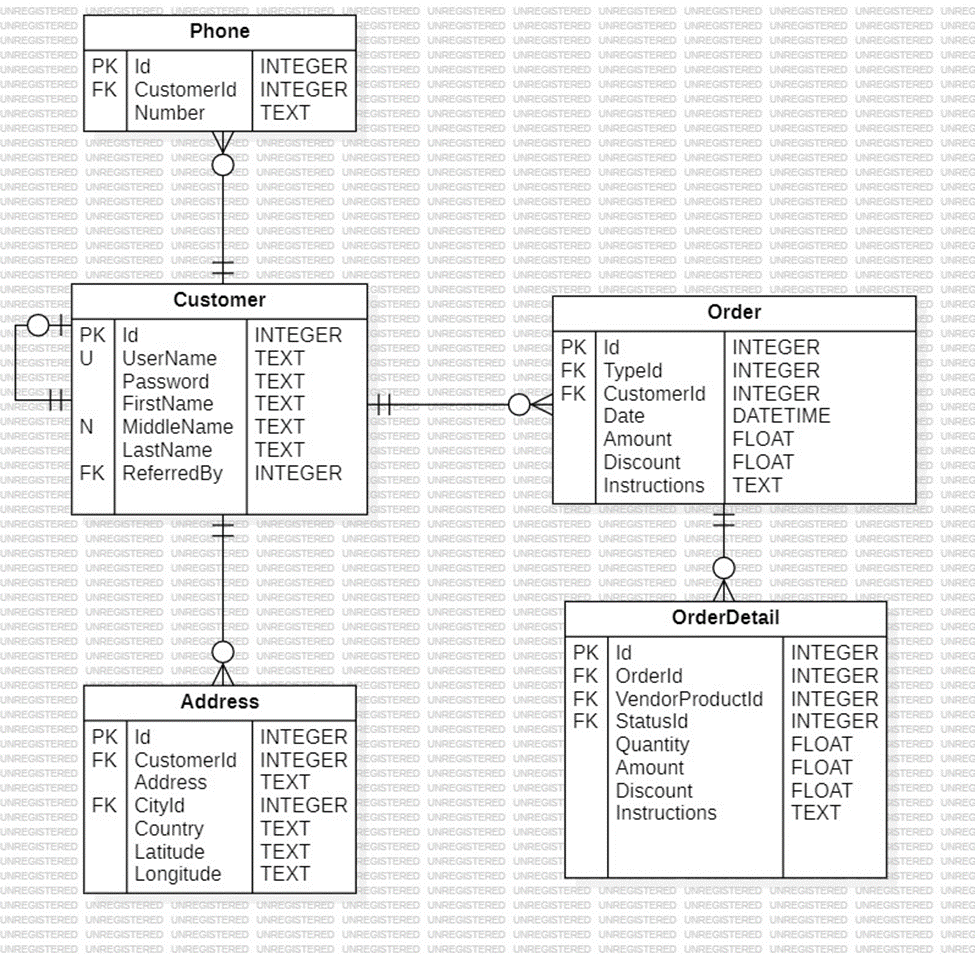
This is the ER diagram for the process of an order, it contains data around the type of order, the product type, the transaction related to that order and the transporter and the zone that is connected with a particular order.



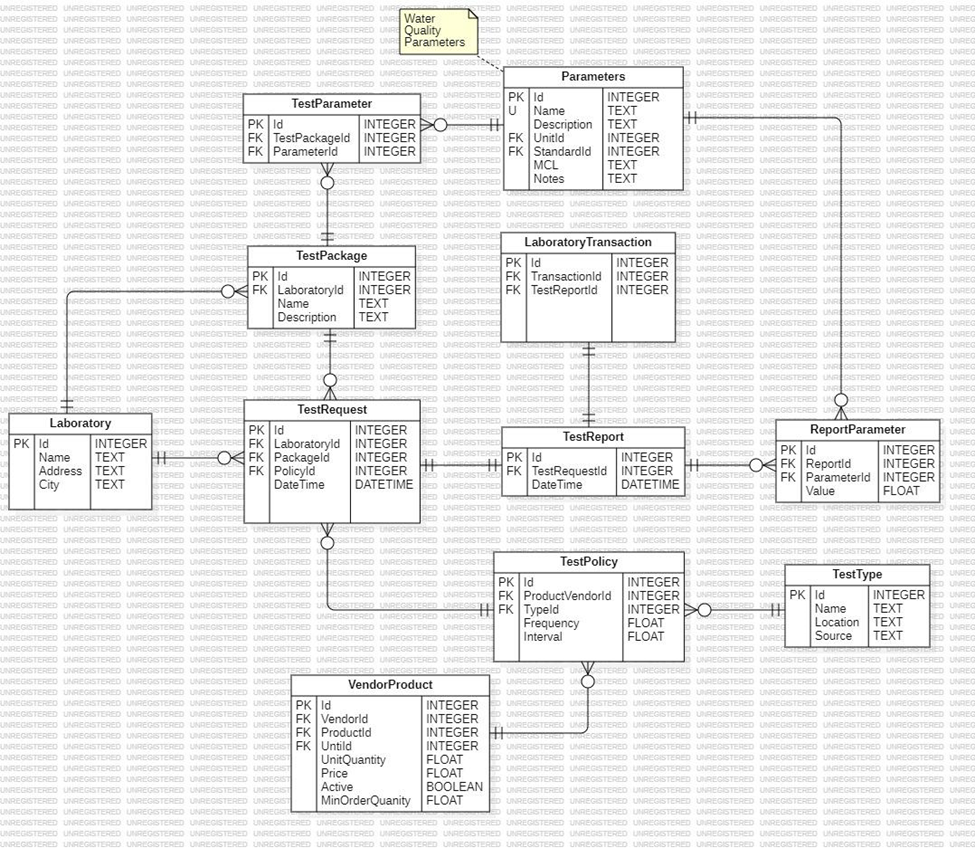
This is ER diagram is explaining the transactions and the data related to a transaction



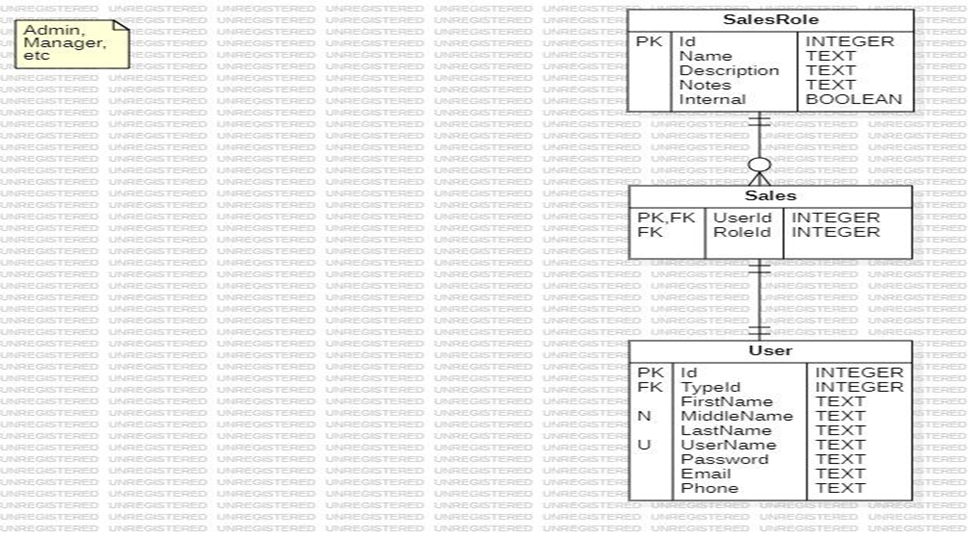
This is an ER diagram is explaining the relationship between the customer and the order.



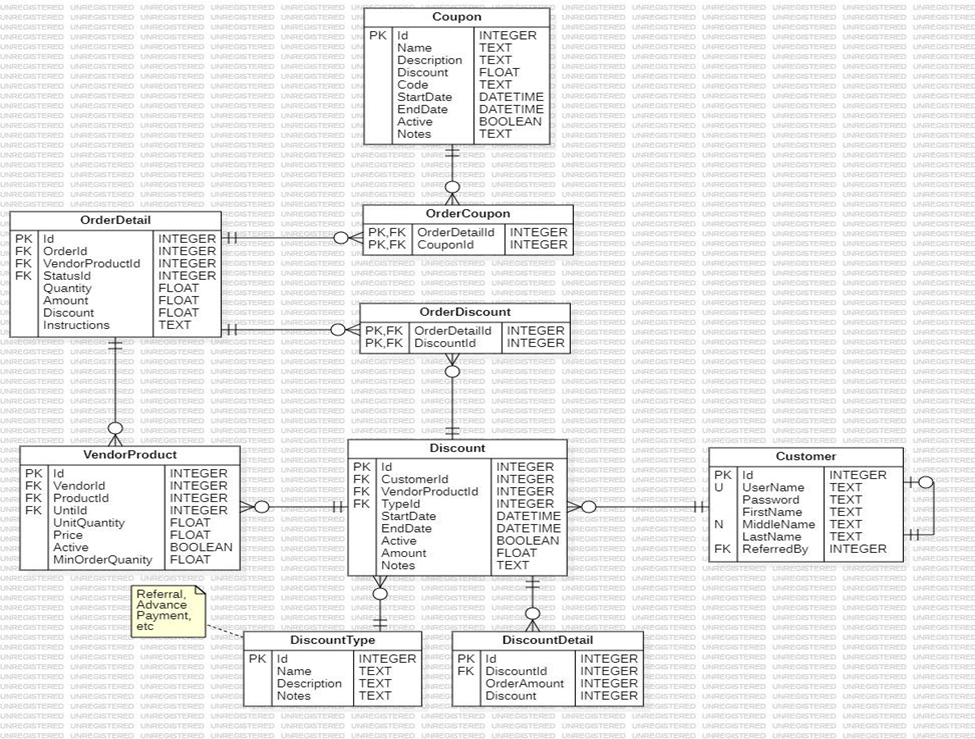
This is the diagram explaining the data related to a laboratory test, we have certain parameters on which a test is conducted. This also incorporates the transaction which goes to the laboratory



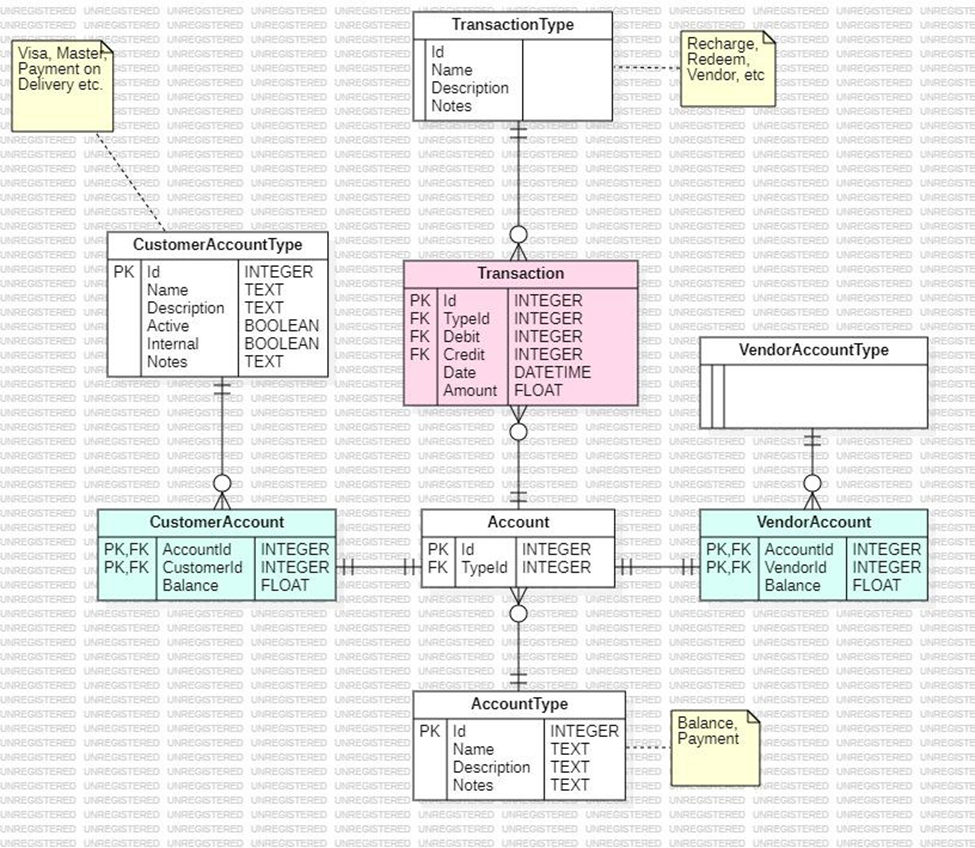
This Er diagram contains how the data around the admin side is like, they are associated with a role which each role and each person having a sales figure as well.



This below is the ER diagram of the the process related to coupons and discount, this functionality will be implemented at later stages of the development cycle



This is the diagram showing the data and relationship between different extities that are related to a transaction.



## API Specification

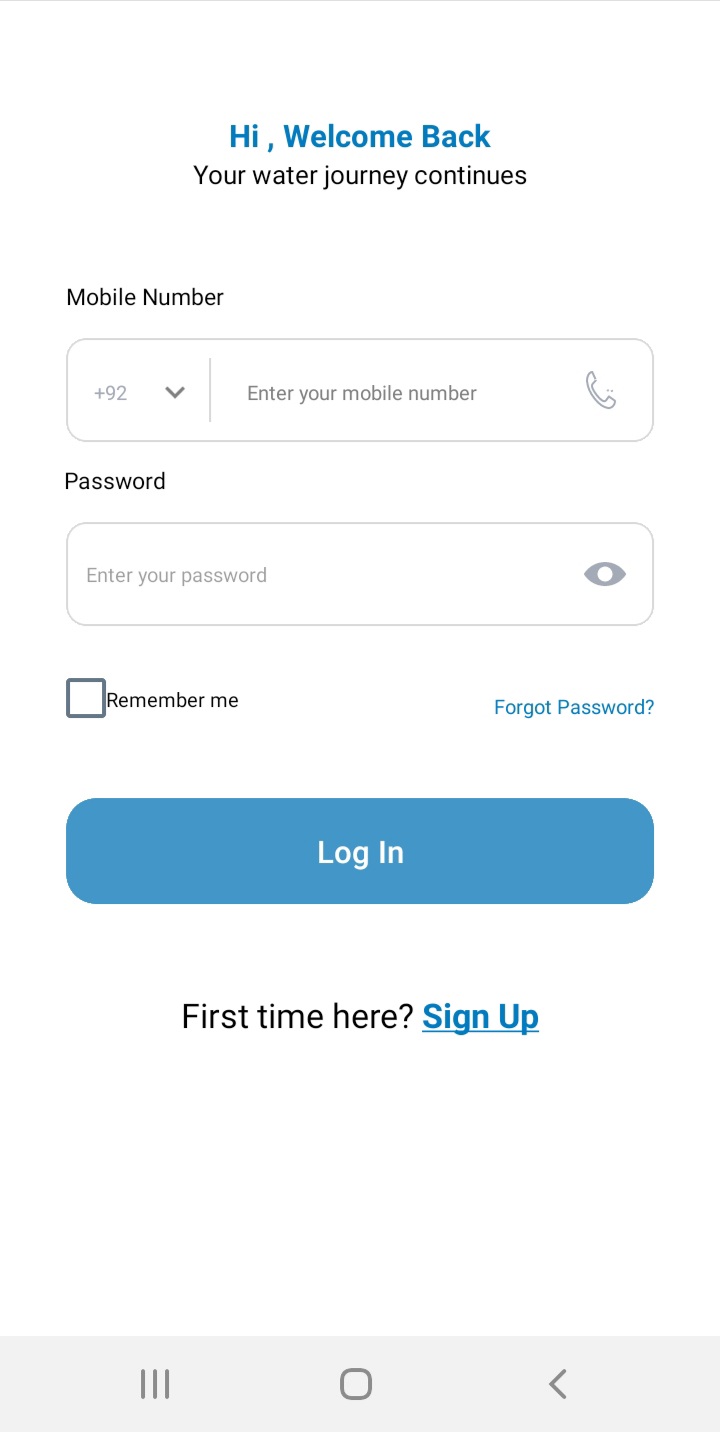
This sub-section will contain the list of external APIs’ that you have used in your project.

# System User Interface

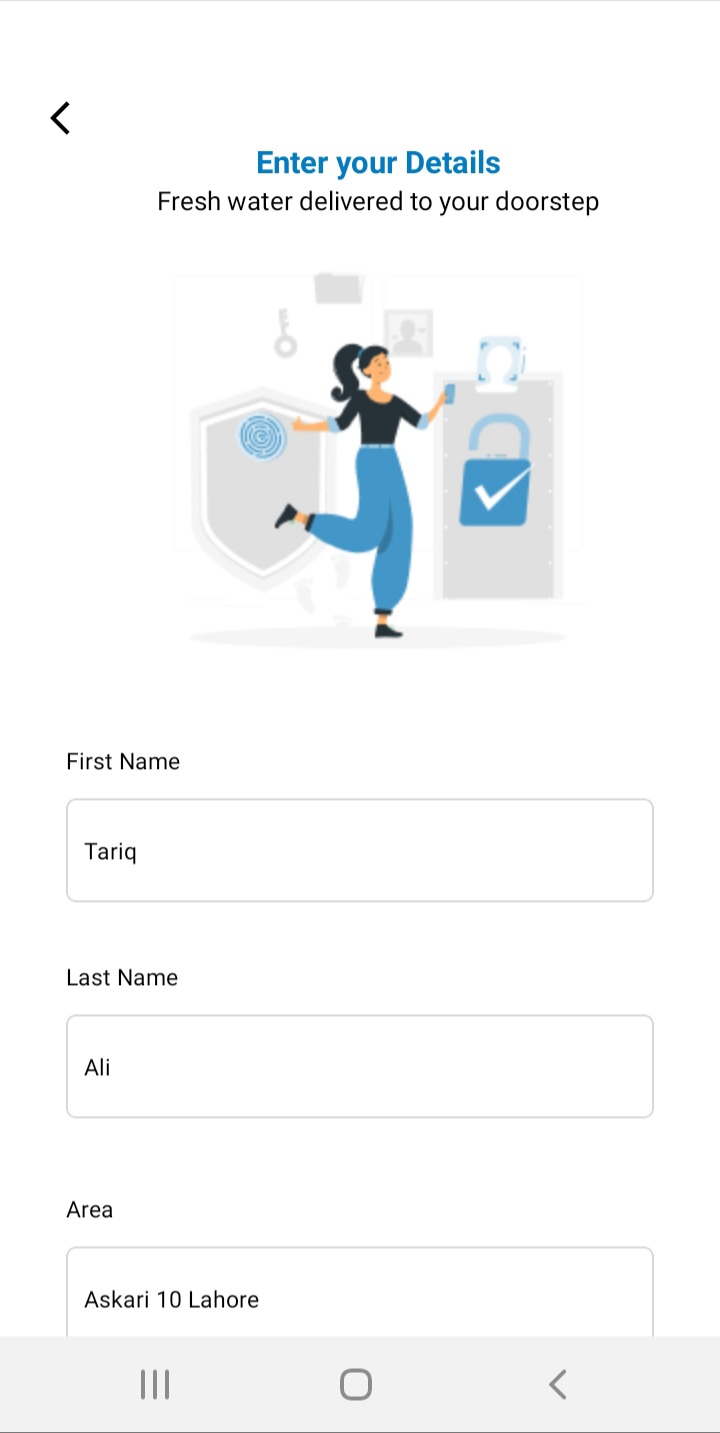
Brief introduction of this chapter in a paragraph

This sub-section should explain the functionality of your application to the end user with supporting screenshots.

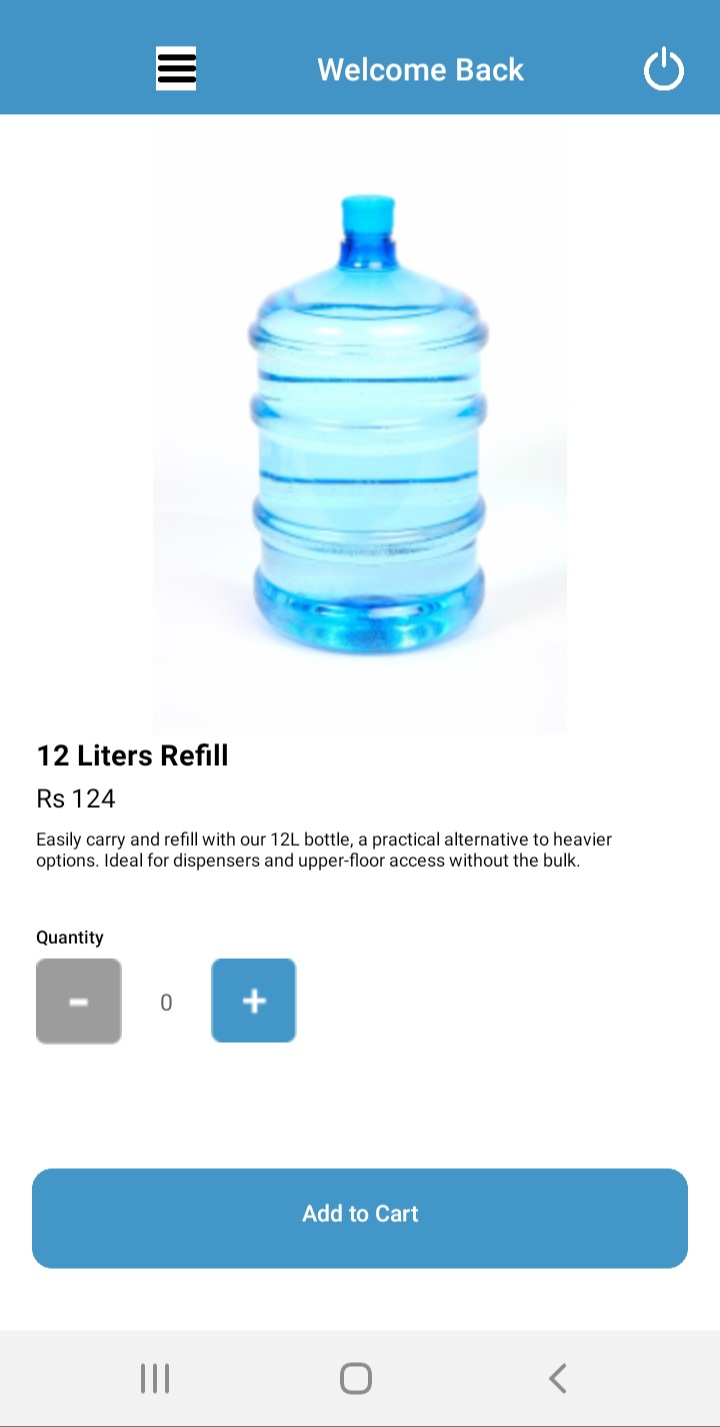
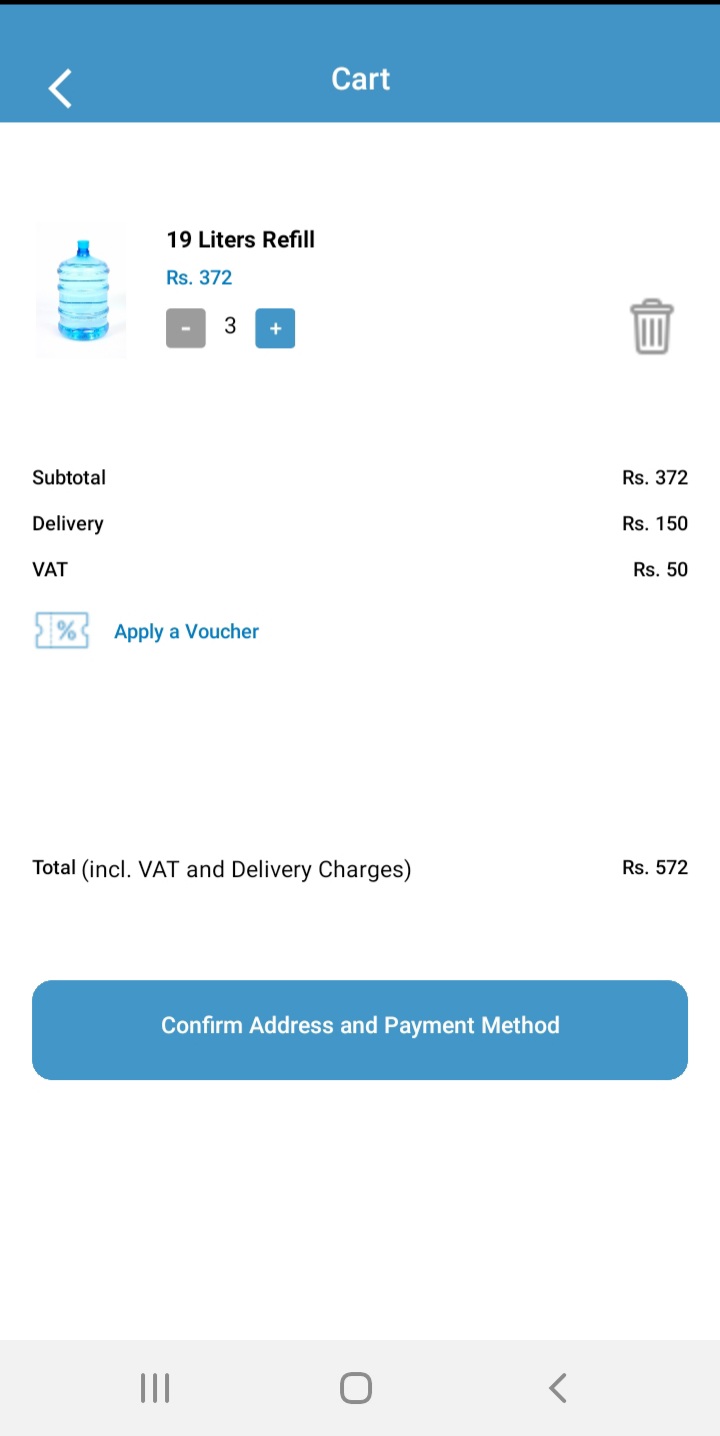
The application opens up to a login screen. Here, a registered user is required to input their phone number, and password then press “Log In” to login. If it is their first time using the application, they can press “Sign up” to be redirected to the sign up page.

****

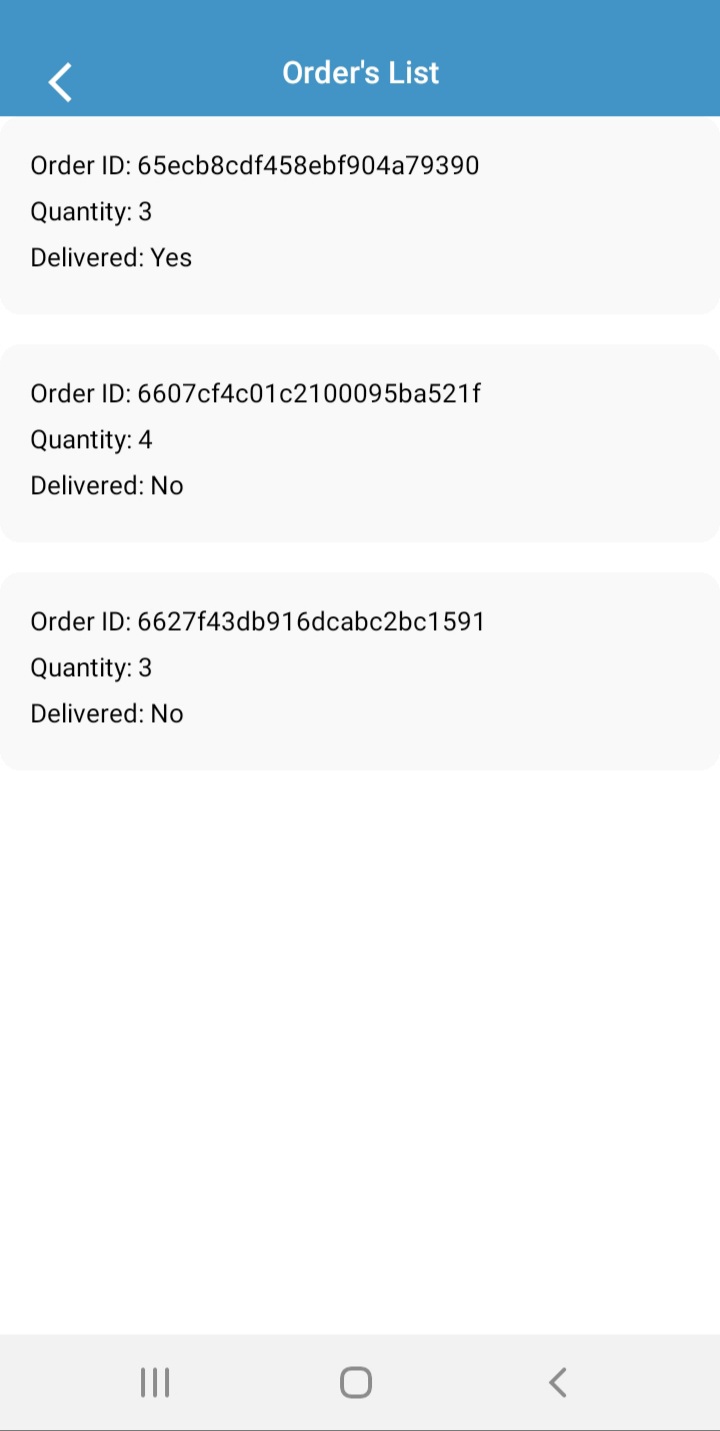
After being redirected upon clicking “Sign Up”, the user is asked whether they want to register as a customer, or as a delivery rider. User selects “Customer”, and is redirected to below form, where they enter their details, then they are redirected to enter their password, and confirm it. After this, the user is confirmed as a customer by the system



After successfully logging in, the home screen is displayed to the user, where the quantity of water, and price of it is displayed, along with a button to select how many bottles does the customer need. After selecting their quantity, customers are redirected to the Cart page, where they’re shown a breakdown of their order, along with prices with VAT and delivery charges. Upon confirming the order, an entry in the database, matching the details, is made, which include the phone Number of who placed the order, their area of residence, quantity of water bottles ordered, and delivery status of the order .

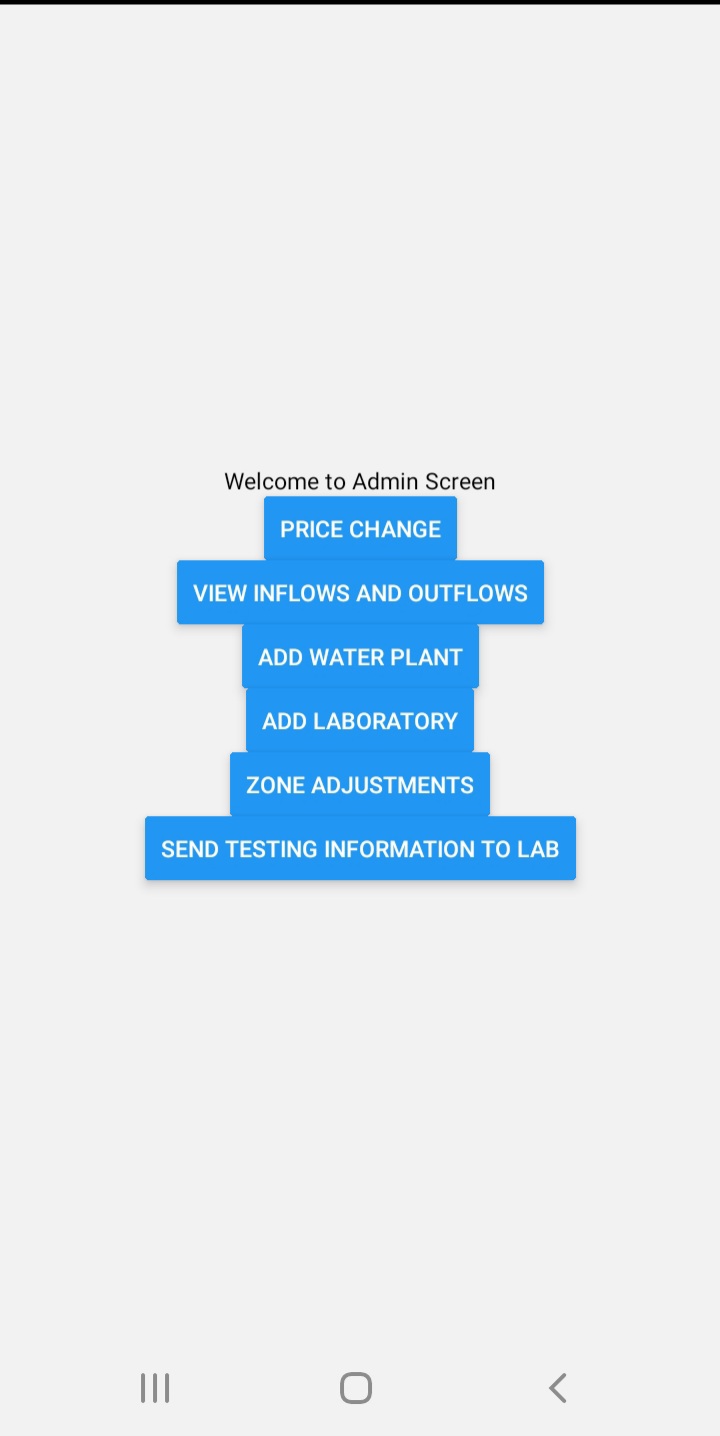
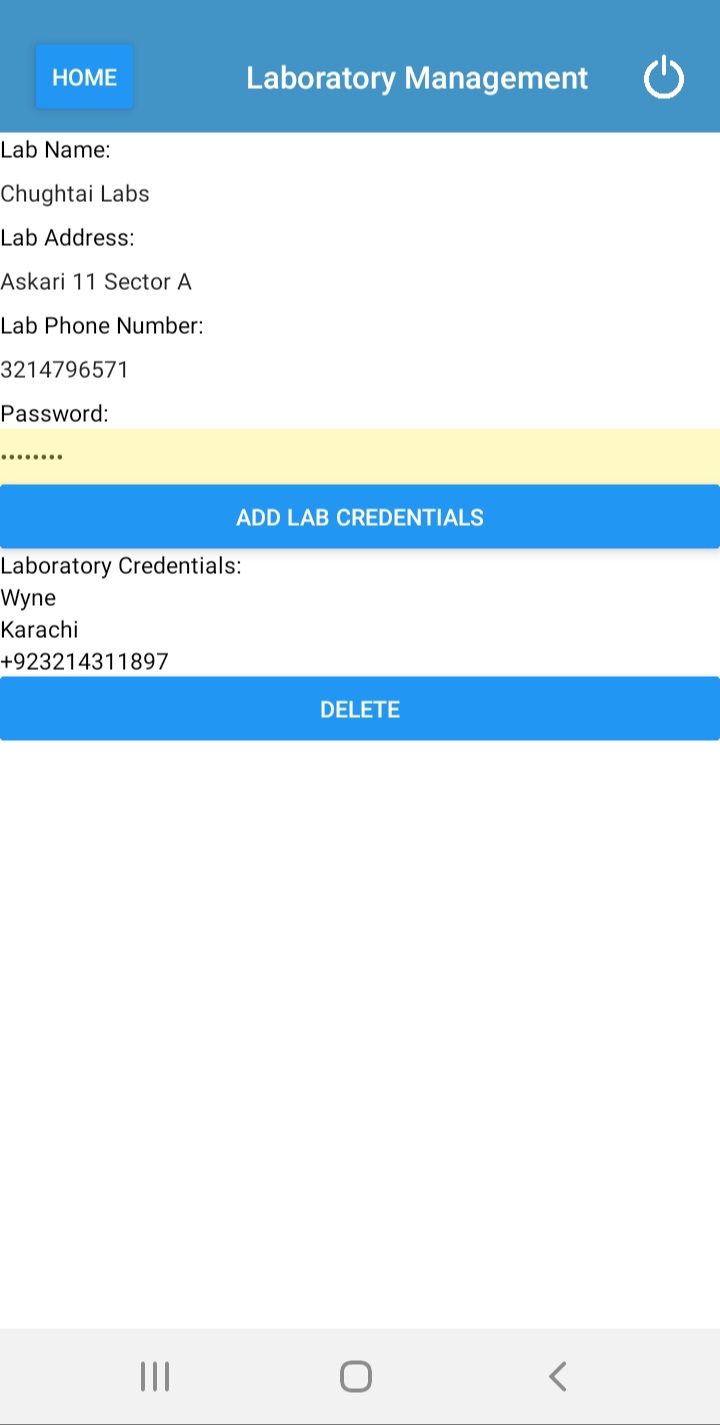


From the homescreen, customers can also select to display their order history, which displays them the order\_id, quantity of water bottles, and whether or not order was delivered.



Similarly, a transporter can view the orders placed, mark them as delivered or not by using the button “Mark as Undelivered/Delivered”, and from picker at top, can filter through the areas of customers.

A user with admin privileges can handle many aspects of the application, such as laboratories. They can add them by adding the matching details into form, and then the laboratories stored in database, are displayed below the form, from where the admin can delete a laboratory if they wish so.



# Project Security

Brief introduction of this chapter in a paragraph

## Project Threats

Identify security threats to your system. While doing this, carefully consider the information/functionality that is most vulnerable from security perspective in the context of your project.

* Insecure Design
* Cryptographic Failures
* Security Misconfiguration

## Potential Losses

For each security risk, identify **potential losses**  (e.g., financial loss, total business loss, litigation etc.) if you do not implement the above controls in your system.

* Total Business Loss
* Financial Loss

## Security Controls

Identify the **controls** (e.g., input validation, audit logs, multi-factor authentication, user roles etc.) that should be implemented in your system in order to address the above threats. Moreover, categorize the identified controls into one or more of the following: ***detective, protective, responsive, recovery***.

* Secure Development Procedures,
* Test the Code,
* Implement Threat Modeling
* Multi-Factor Authentication,
* Encryption
* Software Updates,
* Security Testing

## Static and Dynamic Security Scanning Tools

Explore and select **static and dynamic security scanning tools** for your project. The tools should be selected considering languages and technologies being used in your project.

# Risk Management

## Potential Risks and Mitigation Strategies

| **Sr.** | **Risk Description** | **Mitigation Strategy** |
| --- | --- | --- |
| 1. | Changes in requirements needed by the user, or business can cause disruptions in the project. | Maintain an open dialogue with stakeholders, and document any changes. Regularly, assess impact of any changes to the project, on the scope, budget and schedule. |
| 2. | Problems with servers, or network availability can cause delivery services to get disrupted. | Monitor, and maintain the infrastructure, have a failover plan in case of network availability going down. |
| 3. | Inadequate documentation can hinder maintenance of the software. | Create comprehensive documentation for deployment procedures, configurations, and troubleshooting. |
| 4. | Users may struggle to properly use the new software. | Provide user training, offering user support, as well as gathering user feedback for further improvements. |
| 5. | Data corruption, or loss of data can severely affect both the users and software’s functionality. | Regularly perform data backups, and recovery procedures. Utilise failover mechanism to deal with system failures. |
| 6. | Unforeseen issues, or scope changes can cause delays in schedule of project | Create realistic project schedules with the team, and identify potential bottlenecks in development, and have contingency plans to deal with any hindrances. As well as, regularly updating and communicating project timeline between team members. |
| 7. | Industry regulations, and standards might not be met. | Conduct thorough research and monitoring, maintain a comprehensive compliance checklist, perform regular internal audits, and seek expert consultation. Additionally, prioritise training, documentation, and third-party audits to ensure consistent compliance and contingency planning for unexpected regulatory changes. |
| 8. | Handling payment information poses security risks. | Implement end-to-end encryption, adhere to industry security standards like PCI DSS, and conduct regular security audits to identify and address vulnerabilities. |
| 9. | Since the application utilises third party services for transportation, there can be issues with payment processing and navigation when dealing with them. | Do due diligence in provider selection, robust SLAs, redundancy planning, real-time monitoring, and data security measures. Testing and strong vendor relationships are also crucial to ensure a seamless user experience and prompt issue resolution. |
| 10. | Managing data consistency between interfaces. | Establish a central data repository or database that serves as the single source of truth. Implement standardised data formats and conduct regular data reconciliation checks to ensure consistency and accuracy across interfaces. |

# Testing and Evaluation

Discuss your testing strategy. List down some sample test cases that your created. Moreover, list down the automation tools you used.

Testing is an essential part of ensuring our application, Mashki Online, works reliably and functions properly. We adopted a comprehensive unit testing, integration testing, end-to-end testing and database testing to cover all aspects of the application.

In unit testing, we ensured the forms for signup, login, laboratory and transporter’s order list were rendered correctly, and that the input fields were only accepting valid data. For example, for signup, only accepted phone numbers 10 digit long, the password had capital letters and numbers in it, otherwise the form rejected the inputs.

Important aspect of the application is testing to see whether the different components are interacting correctly, is data flowing correctly between them, and are we getting expected outputs. For example, here we tested that a customer placed an order of 3 liters of water, with phone number +923204301984, area Lahore. The component interacting with the customer's order was the transporter receiving these orders, and being able to mark the order as undelivered, or delivered. Then, on the customer's end, when viewing their order history, the delivery status should match whatever the transporter had set it at their end.

Database is a key aspect of REACT application, and we had to ensure data was being correctly handled and stored in MongoDB. For that, we tested CRUD operations. To test retrieval of data, we checked to see if the customer's order history was correctly retrieved and displayed to them. As stated before, had the transporter mark orders as delivered and ensured the order’s delivery status in the database was updated as being delivered. Then have this be updated on the customer's frontend. Another test done was deleting laboratories, and seeing if the database removed those entries, which it did.

# Deployment Guidelines

List down the steps for deployment of your system. Start from where the code (link of the github repository) should be picked and then mention all the steps for deployment in a production environment. Also mention the online link where your application is hosted along with access information (user/password etc.).

Github repository: <https://github.com/AaraizHassan/P02-MashkiOnline>

From the ‘sprint4’ folder, download the code files.

In the main folder of the code open a terminal.

Enter ‘npm i’ into the terminal

Then enter ‘npx expo start’

Now open another terminal in the ‘server’ folder.

Enter ‘npm i’ into the terminal.

Then enter ‘npm run dev’.

Scan expo QR code from an android device and wait for it to load.

You will have access to the application.

# Conclusion

## Summary

In summarizing the project report on MashkiOnline, the document provides insights into the team's approach and learning outcomes throughout the project development. Here’s a comprehensive summary of their strategy and key learnings:

**Project Approach:**

* **Agile Development Framework:** The project was conducted using an Agile methodology, specifically a sprint-based approach. This allowed the team to work iteratively and adapt to changes and feedback rapidly, which was crucial given the dynamic nature of software development.
* **Specialized Roles and Documentation:** The team consisted of members assuming various roles such as developers, project managers, and testers, with roles rotating to broaden experience and skill sets. Extensive documentation was maintained throughout the project, covering both technical specifications and user guidelines.
* **Comprehensive Testing Strategy:** Testing was an integral part of ensuring the application’s reliability. The team implemented a mix of unit, integration, end-to-end, and database testing to cover all aspects of the application. This was vital for validating the functionality and performance of MashkiOnline.
* **User-Centric Design and Feedback:** Emphasis was placed on understanding user needs through UX research, leading to a design that supports easy navigation and functionality. Regular user feedback helped refine the application’s features and interface.

**Learned Lessons**

* **Flexibility and Adaptability:** The team learned the importance of being flexible and adaptable, allowing them to efficiently address the challenges of changing project requirements and user feedback. This adaptability was facilitated by the Agile process, which supported continuous improvement.
* **The Importance of Testing:** The project highlighted the critical role of comprehensive testing in software development. The team learned to employ various testing methods to ensure each component functioned properly and met user expectations.
* **Team Collaboration and Role Rotation:** Working in a dynamic team environment with rotating roles provided team members with a holistic understanding of the project lifecycle. This experience underscored the importance of collaboration and effective communication in achieving project goals.
* **Challenges in Data Management:** Managing data effectively, particularly in syncing data between the front end and the database, posed significant challenges. The team improved their skills in API usage and database management, ensuring data integrity and flow.
* **Security and Performance:** The project emphasized the necessity of integrating robust security measures and optimizing performance. The team learned to implement security protocols and enhance system performance to handle multiple user interactions simultaneously.
* **Future Extensions and Scalability:** The project taught the team about designing applications with scalability in mind. They recognized the potential for future expansions, like integrating payment systems and extending service offerings beyond water delivery.

**Conclusion:**

The MashkiOnline project not only provided a practical learning platform for the team but also helped them understand the complexities involved in developing a full-fledged software application. From technical challenges to strategic planning and user engagement, the team gained invaluable insights that would aid their future endeavors in software development .

## Challenges

Elaborate the issues and challenges, both technical and non-technical, faced during this project and how you have addressed them.

First challenge which we faced earlier on was due to data being stored in a particular format in the database and then that format was not being followed for retrieval.

The second challenge we faced was on how to extract customer orders from the orders database using customer contact number from customer database.

The third challenge we faced was when the orders list for the customer was not loading from the database. We had to show the customer all the orders he/she ever placed but the screen was not showing the list. We had to figure out the API calls and make sure the correct format is being followed. Then we realized that it was because the orders list from the server was being send as a json object instead of an iterable list.

## Future

* **Advanced Routing Algorithm:** Optimize delivery routes in real-time for speed and efficiency.
* **Franchise Model Integration:** Expand through franchising; ensure standards across all locations.
* **Scalability and Customer Reach:** Increase customer base by extending service to more regions.
* **Integration of Similar Products:** Explore adding related products, leveraging existing infrastructure.
* **Zone-Based Expansion Strategy:** Implement zonal systems for efficient regional management.
* **Sustainable Practices and Community Impact:** Maintain commitment to sustainability and positive community contributions.

# Review checklist

Before submission of this report, the team must perform an internal review. Each team member will review one or more sections of the deliverable.

| **Chapter/Section Name** | **Reviewer Name(s)** |
| --- | --- |
| 1,2,3,4,5,7,8,11,12 | Mohammad Aaraiz Hassan |
| 6,7,9,10,11,12 | Abdul Ahad |
| 3,4,6,7,9,10,11,12 | Usman Zafar |
|  |  |

# References