

Final Year Project

HYDROHITCH

# Project Team:

M Rehmatullah Wadi Wala

2012300

Anish Kumar

2012107

# Project Supervisor:

Ma’am Ayesha Ghayas

27th August 2025

Submitted in partial fulfillment of the requirements for the degree of

# Bachelor of Science in Computer Science

in the

# Faculty of Computing and Engineering Sciences

Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology, Karachi.

# Declaration of Authorship

We, Anish kumar (2012107) and M rehmatullah wadi wala (2012300), declare that this report titled, “Hydro Hitch” and the work presented in it are our own. We confirm that:

This work was done wholly or mainly while in candidature for a bachelor’s degree at this University.

Where any part of this report has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.

Where we have consulted the published work of others, this is always clearly attributed.

Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this report is entirely our own work.

We have acknowledged all main sources of help.

Where the report is based on work done by us jointly with others, we have made clear exactly what was done by others and what we have contributed ourselves.

Name: Anish kumar

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dated: August 26, 2025

Name: M Rehmatullah wadi wala

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dated: August 26, 2025

# Plagiarism Free Certificate

This is to certify that the project titled "HydroHitch" has been successfully completed and submitted

by the undersigned students as part of their Final Year Project at SZABIST University. The

project was carried out under the supervision of Ayesha Ghayas.

We hereby declare that all the content in this report is completely original and has not been copied

from any external source. Wherever external references or materials were used, they have been

properly cited and acknowledged.

This project follows the academic and ethical standards set by SZABIST University. We also

confirm that this work has not been submitted anywhere else for the award of any degree or

qualification.

Name: **\_\_\_Anish Kumar\_\_\_** Name: **M Rehmatullah wadi wala**

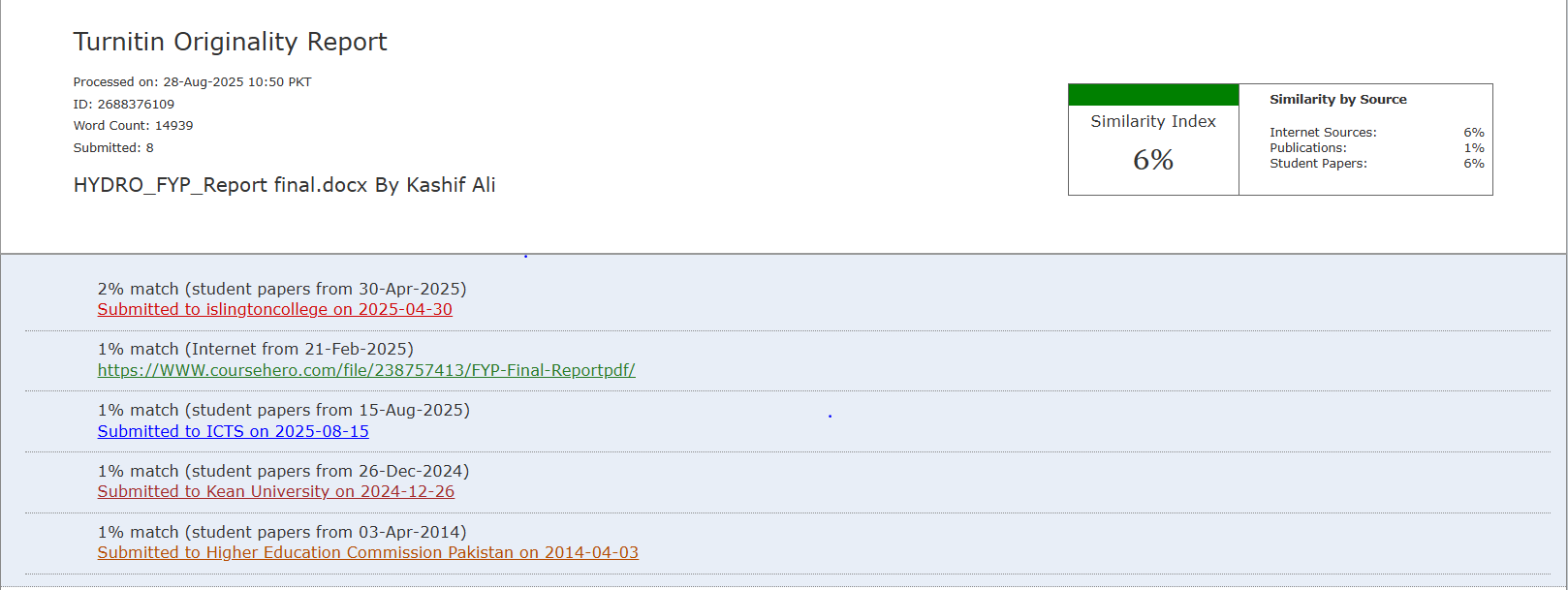
Registration #: **\_2012107\_** Registration #: **\_\_2012300\_**

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Supervisor Name:  **Ayesha Ghayas\_**

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Plagiarism Certificate



# Project Description

HydroHitch is a multivendor water delivery application designed to simplify water procurement for users while enhancing transparency and efficiency in the water distribution industry. Built using the MERN stack (MongoDB, Express.js, React.js, Node.js), the application features a user-friendly interface with three panels: User, Vendor, and Super Admin. It supports seamless interactions for purchasing, managing, and delivering water tankers, making it a robust solution for users, vendors, and administrators.

The **User Panel** allows users to register, log in, and manage profiles, enabling features like browsing and ordering water tankers, accessing detailed water quality reports, and calculating delivery charges based on location and tanker weight. Users can post queries, track order statuses, and view order history for convenience. The **Vendor Panel** enables vendors to manage listings, track and update orders, respond to user queries, and report water quality standards, ensuring high-quality service and transparent communication.

The **Super Admin Panel** provides comprehensive control, with features to manage users and vendors, oversee water quality compliance, and resolve customer queries. Super admins monitor delivery statuses, address complaints, and generate performance and financial reports. By integrating user, vendor, and admin functionalities, HydroHitch offers a scalable, efficient, and transparent platform for water delivery management.

# Acknowledgement

In the name of Allah, the most compassionate and merciful, who bestowed upon us the knowledge and strength to embark on this research endeavor, we express my deepest gratitude and appreciation.

First and foremost, I extend my heartfelt thanks to my esteemed supervisor, **Ma’am Ayesha Ghayas**, from the Computer Science faculty at Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology. Her unwavering support and guidance were invaluable, always providing an open door to address any challenges or queries I faced during my research. His consistent encouragement and motivation played a

vital role in the successful completion of this work.

We are also indebted to our dedicated teachers who shared their profound knowledge and experience with us, nurturing our growth and development in the field of study.

We extend my sincerest appreciation to my loving parents and family members whose unwavering support and encouragement have been a constant source of strength throughout this journey.

Lastly, we are grateful to Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology for providing a stimulating and inspiring environment. The institution's commitment to delivering quality education and its supportive faculty members have continuously boosted the confidence and enthusiasm of every student at the university.

We would also like to express my thanks to all those who generously shared their knowledge and resources, contributing to the success of this research.

Once again, thank you to everyone who has been a part of this journey and helped make this research possible.

Contents

Declaration Of authorship …………………………………………………………..……………………………………………………………. I

Plagiarism Free Certificate ………………………………………………………………………………………………………………………II

Plagiarism Certificate ……………………………………………………………………………………………………………………………. III

Project Description …………………………………………………………………………………………………………………………………IV

Acknowledgement ………………………………………………………………………………………………………………………………….V

[PROJECT PROPOSAL 1](#_bookmark0)

1. [Introduction 2](#_bookmark1)
2. [Objective 2](#_bookmark2)
3. [Problem Description 2](#_bookmark3)
4. [Methodology 2](#_bookmark4)
5. [Project Scope 3](#_bookmark5)
6. [Feasibility Study 3](#_bookmark6)
7. [Solution Application Areas 3](#_bookmark7)
8. [Tools/Technology 3](#_bookmark8)
9. [Expertise of the Team Members 4](#_bookmark9)
10. [Milestones 4](#_bookmark10)
11. [Work Breakdown Structure 4](#_bookmark11)
12. [Work Breakdown Structure 5](#_bookmark12)
13. [References 5](#_bookmark13)

[SOFTWARE REQUIREMENTS SPECIFICATION 6](#_bookmark14)

1. [Introduction 7](#_bookmark15)
   1. [Purpose 7](#_bookmark16)
   2. [Document Conventions 7](#_bookmark17)
   3. [Intended Audience and Reading Suggestions 7](#_bookmark18)
   4. [Product Scope 7](#_bookmark19)

[FYP 1 7](#_bookmark20)

[FYP 2 7](#_bookmark21)

* 1. [References 8](#_bookmark22)

1. [Overall Description 8](#_bookmark23)
   1. [Product Perspective 8](#_bookmark24)
   2. [Product Functions 8](#_bookmark25)
   3. [User Classes and Characteristics 9](#_bookmark26)
   4. [Operating Environment 9](#_bookmark27)
   5. [Design and Implementation Constraints 9](#_bookmark28)
   6. [User Documentation 9](#_bookmark29)

[2.6. Assumptions and Dependencies 9](#_bookmark30)

1. [External Interface Requirements 10](#_bookmark31)
   1. [User Interfaces 10](#_bookmark32)
   2. [Hardware Interfaces 2](#_bookmark33)4
   3. [Software Interfaces 2](#_bookmark34)4
   4. [Communication Interfaces 2](#_bookmark35)4
2. [System Features 2](#_bookmark36)4
   1. [User Registration 2](#_bookmark37)4
      1. [Description 2](#_bookmark38)4
      2. [Stimulus/Response Sequences 2](#_bookmark39)5
      3. [Functional Requirements 26](#_bookmark40)
   2. [Login 26](#_bookmark41)
      1. [Description 26](#_bookmark42)
      2. [Stimulus/Response Sequences 26](#_bookmark43)
      3. [Functional Requirements](#_bookmark44) 27
   3. [Tanker Management 28](#_bookmark45)
      1. [Description 28](#_bookmark46)
      2. [Stimulus/Response Sequences 28](#_bookmark47)
      3. [Functional Requirements 29](#_bookmark48)
   4. [Feedback System 29](#_bookmark49)
      1. [Description 29](#_bookmark50)
      2. [Stimulus/Response Sequences 29](#_bookmark51)
      3. [Functional Requirements 3](#_bookmark52)0
   5. [Order Management 3](#_bookmark53)1
      1. [Description 3](#_bookmark54)1
      2. [Stimulus/Response Sequences 3](#_bookmark55)1
      3. [Functional Requirements 3](#_bookmark56)2
   6. [Water Quality Reports 3](#_bookmark57)2
      1. [Description 3](#_bookmark58)2
      2. [Stimulus/Response Sequences 3](#_bookmark59)2
      3. [Functional Requirements 3](#_bookmark60)3
   7. [Order Tracking 3](#_bookmark61)3
      1. [Description 3](#_bookmark62)3
      2. [Stimulus/Response Sequences 3](#_bookmark63)3
      3. [Functional Requirements 3](#_bookmark64)4
   8. [Delivery Charge Calculation 3](#_bookmark65)5
      1. [Description 3](#_bookmark66)5
      2. [Stimulus/Response Sequences 3](#_bookmark67)5
      3. [Functional Requirements 36](#_bookmark68)
   9. [Location-Based Vendor Access 36](#_bookmark69)
      1. [Description 36](#_bookmark70)
      2. [Stimulus/Response Sequences 36](#_bookmark71)
      3. [Functional Requirements 37](#_bookmark72)
   10. [Emergency Water Service 37](#_bookmark73)
       1. [Description 37](#_bookmark74)
       2. [Stimulus/Response Sequences 37](#_bookmark75)
       3. [Functional Requirements 38](#_bookmark76)
   11. [Respond to User Queries 39](#_bookmark77)
       1. [Description 39](#_bookmark78)
       2. [Stimulus/Response Sequences 39](#_bookmark79)
       3. [Functional Requirements 4](#_bookmark80)0
3. [Other Non-functional Requirements 4](#_bookmark81)0
   1. [Performance Requirements 4](#_bookmark82)0
   2. [Safety Requirements 4](#_bookmark83)0
   3. [Security Requirements 4](#_bookmark84)0
   4. [Software Quality Attributes 4](#_bookmark85)1
   5. [Business Rules 4](#_bookmark86)1
4. [Other Requirements 4](#_bookmark87)1
   1. [Appendix A: Glossary 4](#_bookmark88)1

[SOFTWARE DESIGN SPECIFICATION 4](#_bookmark93)2

1. [Introduction 4](#_bookmark94)3
   1. [Purpose of this document 4](#_bookmark95)3
   2. [Scope of the development project 4](#_bookmark96)3
   3. [Definition, acronyms, and abbreviations 4](#_bookmark97)3
   4. [References 4](#_bookmark98)3
   5. [Overview of Document 4](#_bookmark99)3
2. [System Architecture Description 4](#_bookmark100)4
   1. [Section overview 4](#_bookmark101)4
   2. [General Constraints 4](#_bookmark102)4
   3. [Data Design 4](#_bookmark103)4
   4. [Program Structure 4](#_bookmark104)4
   5. [Alternatives Considered 4](#_bookmark105)4
3. [Detailed description of components 4](#_bookmark106)4
   1. [Section overview 4](#_bookmark107)4
   2. [Registration 4](#_bookmark108)5
   3. [Login 4](#_bookmark109)5
   4. [Ordering Tankers 46](#_bookmark110)
   5. [Order management 46](#_bookmark111)
   6. [Feedback 47](#_bookmark112)
   7. [Query System 47](#_bookmark113)
   8. [Water Quality Reports 48](#_bookmark114)
   9. [Emergency Orders 48](#_bookmark115)
   10. [Delivery Charge Calculation 49](#_bookmark116)
4. [User interface design 49](#_bookmark117)
   1. [Section overview 49](#_bookmark118)
   2. [Interface design rules 49](#_bookmark119)
   3. [GUI components 49](#_bookmark120)
   4. [Detailed description 5](#_bookmark121)0
5. [Reuse and relationships to other products 5](#_bookmark122)0
6. [Design decision and tradeoffs 5](#_bookmark123)0
7. [Pseudo code for components 5](#_bookmark124)0
   1. [Registration 5](#_bookmark125)0
   2. [Login 5](#_bookmark126)1
   3. [Ordering the tanker 5](#_bookmark127)2
   4. [Order management 5](#_bookmark128)2
   5. [Feedback 5](#_bookmark129)3
   6. [Query system 5](#_bookmark130)3
   7. [Emergency Request 5](#_bookmark131)3
   8. [Cost calculator 5](#_bookmark132)4
   9. [Order Tracking 5](#_bookmark133)4
8. [Appendices 5](#_bookmark134)5
   1. [State Chart Diagram 5](#_bookmark135)5
      1. [User 5](#_bookmark136)5
      2. [Vendor 56](#_bookmark137)
      3. [Admin 56](#_bookmark138)
   2. [Activity Diagram 57](#_bookmark139)
      1. [Registration 57](#_bookmark140)
      2. [Login 58](#_bookmark141)
      3. [Tanker Ordering 59](#_bookmark142)
      4. [Order Management 6](#_bookmark143)0
      5. [Emergency System 6](#_bookmark144)1
      6. [Feedback 6](#_bookmark145)2
      7. [Query System 6](#_bookmark146)3
      8. [Quality Reports 6](#_bookmark147)4
   3. [Component Diagram 6](#_bookmark148)5
      1. [Admin 6](#_bookmark149)5
      2. [User 66](#_bookmark150)
      3. [Vendor 67](#_bookmark151)
   4. [Sequence Diagram 68](#_bookmark152)
      1. [Registration 68](#_bookmark153)
      2. [Login 68](#_bookmark154)
      3. [Order Management 69](#_bookmark155)
      4. [Tanker Ordering 69](#_bookmark156)
      5. [Emergency System 7](#_bookmark157)0
      6. [Feedback 7](#_bookmark158)0
      7. [Query System 7](#_bookmark159)1
      8. [Quality Reports 7](#_bookmark160)2
   5. [Object Diagram 7](#_bookmark161)3
   6. [Deployment Diagram 7](#_bookmark162)4
   7. [Collaboration Diagram 7](#_bookmark163)5
      1. [Registration 7](#_bookmark164)5
      2. [Login 7](#_bookmark165)5
      3. [Order Management 76](#_bookmark166)
      4. [Tanker Ordering 76](#_bookmark167)
      5. [Emergency System 77](#_bookmark168)
      6. [Quality Report 77](#_bookmark169)
      7. [Feedback 78](#_bookmark170)
      8. [Query Service 78](#_bookmark171)
      9. [Vendor Profile 79](#_bookmark172)

8.7.10 [Use case Diagram 79](#_bookmark90)

8.7.11 [Entity Relationship Diagram 8](#_bookmark91)2

8.7.12 [Class Diagram 8](#_bookmark92)3

[SOFTWARE TESTING DOCUMENT 8](#_bookmark173)4

1. [Introduction 8](#_bookmark174)5
   1. [Purpose of this Document 8](#_bookmark175)5
   2. [Intended Audience 8](#_bookmark176)5
   3. [Document Convention 8](#_bookmark177)5
   4. [Project Overview 8](#_bookmark178)5
2. [Test Cases 86](#_bookmark179)
   1. [Registration-1 86](#_bookmark180)
   2. [Registration-2 86](#_bookmark181)
   3. [Registration-3 87](#_bookmark182)
   4. [Login-1 87](#_bookmark183)
   5. [Login-2 88](#_bookmark184)
   6. [Login-2 88](#_bookmark185)
   7. [Tanker Ordering-1 89](#_bookmark186)
   8. [Tanker Ordering-2 89](#_bookmark187)
   9. [Tanker Ordering-3 9](#_bookmark188)0
   10. [Order Management-1 9](#_bookmark189)0
   11. [Order Management-2 9](#_bookmark190)1
   12. [Order Management-3 9](#_bookmark191)1
   13. [Feedback-1 9](#_bookmark192)2
   14. [Feedback-2 9](#_bookmark193)2
   15. [Feedback-3 9](#_bookmark194)3
   16. [Query System-1 9](#_bookmark195)3
   17. [Query System-2 9](#_bookmark196)4
   18. [Query System-3 9](#_bookmark197)4
   19. [Quality Report-1 9](#_bookmark198)5
   20. [Quality Report-2 9](#_bookmark199)5
   21. [Quality Report-3 96](#_bookmark200)
   22. [Emergency System-1 96](#_bookmark201)
   23. [Emergency System-2 97](#_bookmark202)
   24. [Emergency System-3 97](#_bookmark203)
   25. [Cost Calculator-1 98](#_bookmark204)
   26. [Cost Calculator-2 98](#_bookmark205)

[USER MANUAL 99](#_bookmark206)

1. [Website Screens 1](#_bookmark207)00
   1. [Splashes 1](#_bookmark208)04
   2. [Register as user 1](#_bookmark209)05
   3. [Register as Vendor 1](#_bookmark210)06
   4. [Login as Vendor 1](#_bookmark211)07
   5. [Login as User 1](#_bookmark212)07
   6. [Admin dashboard 1](#_bookmark213)08
   7. [Vendor report 1](#_bookmark214)09
   8. [Water Quality Report 1](#_bookmark215)10
   9. [Feedback 1](#_bookmark215)10
   10. [Order booking 1](#_bookmark216)11
   11. [User query 1](#_bookmark217)12
   12. [Add product 1](#_bookmark218)13
   13. [User reports 1](#_bookmark219)14
   14. [User management 1](#_bookmark220)14
   15. [Order management 1](#_bookmark221)15
   16. [Admin Login 1](#_bookmark222)16

**HYDROHITCH**

# PROJECT PROPOSAL

#### Introduction

HydroHitch is a multi-vendor water delivery platform designed to meet the growing demand for dependable and high-quality water delivery services. By bringing together users, vendors, and administrators on a unified platform, the application facilitates smooth interactions and transactions within the water distribution industry.

With its user-friendly interface and powerful features, this enables users to order water tankers, access quality reports, and calculate delivery costs effortlessly. Vendors benefit from tools to manage listings, track orders, and provide detailed quality reports. Meanwhile, super admins have dedicated controls to oversee operations and ensure efficient service delivery.

Built on the scalable and flexible MERN stack, it provides tailored interfaces for each user type, streamlining communication and management. The app addresses key challenges in water delivery by prioritizing quality assurance, user convenience, and effective vendor oversight, positioning itself as a reliable solution for meeting community water needs.

#### Objective

The objective of the \*\*HydroHitch\*\* project is to develop a scalable, user-friendly multi-vendor water delivery platform that addresses the growing need for reliable water services. It aims to ensure user convenience by providing features like water ordering, quality reporting, and real-time tracking. The application will enhance vendor management with tools for listing, order tracking, and quality assurance. Scalability and security are prioritized through a robust MERN stack architecture and data protection protocols. Usability testing will ensure the platform is accessible and responsive to users' diverse needs. Ultimately, HydroHitch seeks to provide a dependable, transparent, and efficient solution for water distribution.

#### Problem Description

With growing populations, the demand for clean and safe water delivery services is on the rise. However, the industry often lacks transparency in water quality and vendor selection, with existing solutions offering limited features. HydroHitch bridges this gap by delivering a reliable platform that integrates essential features for water delivery. HydroHitch provides water quality reports, location-based vendor options, and real-time order tracking, ensuring convenience, safety, and transparency. By consolidating these services into a single platform, it simplifies water procurement, empowering users with a seamless and trustworthy experience for all their water delivery needs.

#### Methodology

The development of HydroHitch will follow an agile methodology, ensuring flexibility and iterative improvements through user feedback. The project will be executed in the following phases:

* 1. **Requirements Analysis**: Gather insights through user interviews, surveys, and market research to align the app with user and industry needs.
  2. **Design**: Develop wireframes, prototypes, and user interface designs for the user, vendor, and super admin panels.
  3. **Development**: Build the front-end and back-end using the MERN stack, incorporating robust security features and APIs for seamless functionality.
  4. **Testing**: Perform comprehensive functional, usability, performance, and security testing to ensure reliability across various scenarios.
  5. **User Feedback**: Collect and analyze user feedback to implement iterative enhancements and improve the overall user experience.
  6. **Deployment**: Launch the application in a live environment, focusing on scalability and accessibility for all stakeholders.
  7. **Documentation**: Create detailed user, vendor, and admin guides alongside technical documentation to support future maintenance and updates.

#### Project Scope

HydroHitch is a comprehensive water delivery solution that incorporates various features to streamline ordering, quality assurance, and delivery. The project aims to provide a platform that connects users with local water suppliers and ensures access to quality water. Key features include:

* 1. **User Registration and Profiles:** Allow users to create, edit, and manage their profiles.
  2. **Search and Vendor Selection:** Enable users to view and select water vendors based on location and user ratings.
  3. **Water Order Placement:** Allow users to order water tankers of varying capacities.
  4. **Quality Reporting:** Provide detailed water quality reports, including pH levels and contamination information.
  5. **Delivery Charges Calculation:** Calculate charges based on location, volume, and vendor pricing.
  6. **Order Tracking:** Enable real-time tracking of water deliveries.
  7. **Query and Feedback System:** Allow users to post queries and receive responses from vendors or super admins.
  8. **Emergency Service Feature:** Offer a quick access feature for immediate water service requests.

##### Feasibility Study

The outlined scope of HydroHitch suggests a well-rounded solution for users, but it also presents several challenges:

**Risks Involved:** The primary risk is user adoption, as the success of the app depends on its appeal and accessibility to a wide audience. Building trust through transparent water quality reporting and consistent service will be key.

**Resource Requirements:** Essential resources include domain hosting for platform deployment and APIs for real-time location tracking and quality reporting.

##### Solution Application Areas

The HydroHitch platform is designed to serve various user groups, including households, businesses, and vendors, ensuring reliable water delivery with verified quality reports. Key application areas are:

**Home Water Delivery:** Enable easy access to safe, quality water for households.

**Business and Office Use:** Cater to businesses requiring bulk water delivery.

**Vendor Services:** Help water vendors manage orders, deliveries, and quality assurance.

##### Tools/Technology

The development stack for **HydroHitch** includes:

* **Front-End**: React Native (for cross-platform mobile app development)
* **Back-End**: Node.js and Express.js
* **Database**: MongoDB (for managing vendor, user, and order data)
* **Additional Tools**: Visual Studio Code, Android Studio, Google Maps API (for location tracking), and AWS for hosting and storage.

##### Expertise of the Team Members

* + Anish kumar: Experienced in React JS and Node.js, with strong skills in back-end development and API integration.
  + Rehmatullah: Skilled in UI/UX design, app development, and implementing secure, responsive applications. Our team has a solid understanding of our combined strengths and previous project experience, enabling us to collaborate effectively to deliver this project.

##### Milestones

1. **User Pane:**
   * **Order Water Tankers**: Browse tanker options, view details, and place orders.
   * **Water Quality Reports**: Access quality reports, including pH levels, contamination tests, and source details.
   * **Location-Based Vendor Access**: View vendors near the user's location for quicker delivery.
   * **Delivery Charges Calculation**: Calculate charges based on weight and destination.
   * **Post Queries & View Replies**: Submit queries and receive responses from vendors and super admins.
   * **Order Tracking**: Track the status of orders in real-time, from dispatch to delivery.
2. **Vendor Panel:**
   * **Add and Manage Tankers**: List tankers, update information, and manage availability.
   * **Order Management**: Track and update order statuses for users.
   * **Respond to Queries**: Address user questions promptly.
   * **Water Quality Management**: Manage and update water quality reports as needed.
   * **Location-Based Service**: Provide accurate location details for effective service.
3. **Super Admin Panel:**
   * **User and Vendor Management**: Add, update, and delete users and vendors as needed.
   * **Water Quality Oversight**: Review and verify water quality reports submitted by vendors.
   * **Query Handling**: Address user and vendor queries, escalating when needed.
   * **Order and Delivery Tracking**: Monitor delivery statuses and address any delays.
   * **Reporting and Analytics**: Generate performance reports on orders, vendor activity, and customer satisfaction.

##### Work Breakdown Structure

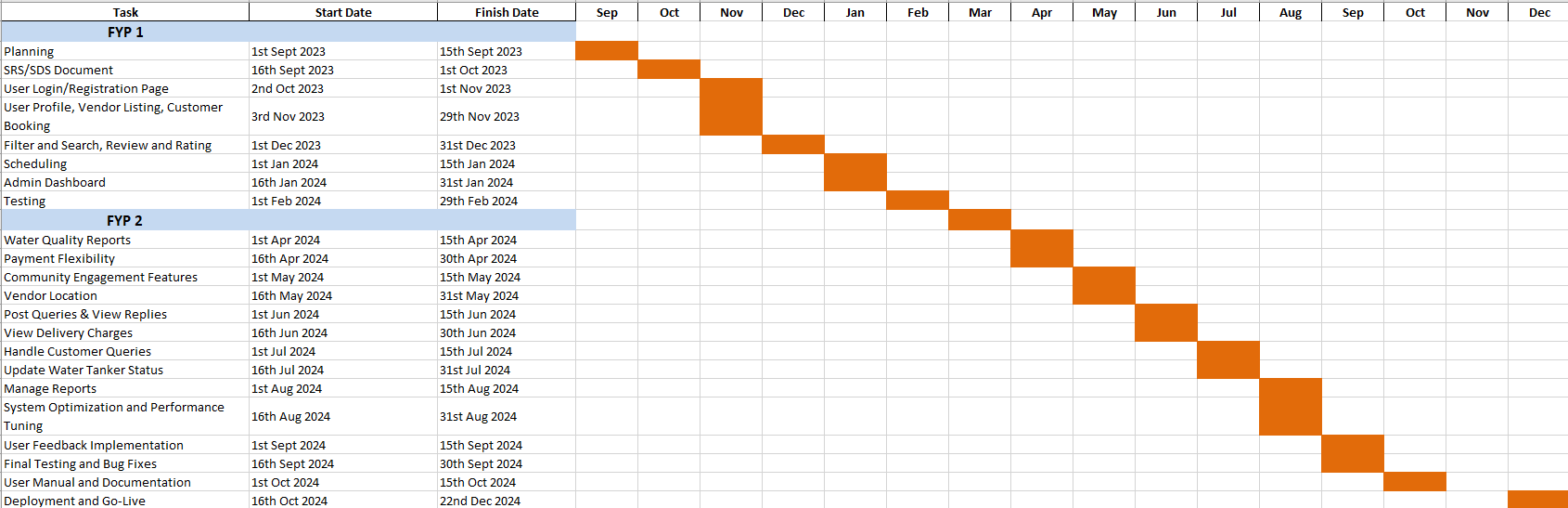
**Work for FYP-1**

* + Research & Planning
  + UI/UX Design & Prototyping
  + Technical Documentation
  + Backend Development

**Work for FYP-2**

* + Frontend Development & Integration
  + Feature Development (User Registration, Order Management, Quality Reporting)
  + Testing (Functional, Usability, and Security)
  + Finalization & Deployment

##### Work Breakdown Structure

****

##### References

1. <https://www.mongodb.com/>
2. <https://expressjs.com/>
3. <https://reactjs.org/>
4. <https://nodejs.org/>
5. <https://www.w3schools.com/>
6. <https://developer.mozilla.org/>
7. <https://www.codecademy.com/>
8. <https://tailwindcss.com/>

### SOFTWARE REQUIREMENTS SPECIFICATION

#### Introduction

##### Purpose:

The purpose of the HydroHitch project is to develop a comprehensive multi-vendor water delivery platform that addresses the challenges of water quality assurance, user convenience, and vendor management. HydroHitch aims to simplify the process of ordering water tankers, ensure transparency in water quality, and provide a seamless interaction platform for users, vendors, and administrators. This application serves as a dependable solution for communities, businesses, and households, ensuring safe, reliable, and efficient water delivery services.

##### Document Conventions:

|  |  |  |
| --- | --- | --- |
|  | **FONT NAME** | **FONT SIZE** |
| Headings 1 | Times New Roman (Bold) | 18 |
| Headings 2 | Times New Roman (Bold) | 16 |
| Headings 3 | Times New Roman (Bold) | 12 |
| Paragraphs | Times | 11 |

**Table No: 1.2 Document Conventions**

##### Intended Audience and Reading Suggestions:

This SRS intends to attract multiple types of audience that includes project team, our advisor, our FYP coordinator and many more.

**Project team:** Anish kumar and Rehmatullah

**Advisor:** Ma’am Ayesha Ghayas

**Fyp Coordinator:** Ma’am Ayesha Ghayas

**Other**s**:** FYP evaluators, Users, and testers

This document provides the reader with complete knowledge of the project.

##### Product Scope:

FYP 1:

1. **Requirement Analysis:** Collect and document user and system requirements through surveys and market research.
2. **UI/UX Design:** Develop prototypes, wireframes, and user interface designs for the platform.
3. **Backend Setup:** Define database schemas in MongoDB for users, vendors, and orders.
4. **Basic Frontend Development:** Implement the core UI screens using React Native.
5. **API Development:** Begin building foundational APIs for essential functions like registration and order management.
6. **Documentation:** Create technical and user documentation for the initial phases of the project.

FYP 2:

1. **Feature Development:** Add advanced features such as order tracking, quality reporting, and billing calculations.
2. **Integration:** Connect the frontend and backend, ensuring seamless communication between components.
3. **Testing:** Conduct thorough functional, usability, and security testing.
4. **Feedback Iterations:** Collect user feedback and make iterative improvements.
5. **Deployment:** Deploy the application to cloud hosting platforms and mobile app stores.
6. **Marketing:** Launch a campaign to promote app adoption.

##### References:

1. <https://www.mongodb.com/>
2. <https://expressjs.com/>
3. <https://reactjs.org/>
4. <https://nodejs.org/>
5. <https://www.w3schools.com/>
6. <https://developer.mozilla.org/>
7. <https://www.codecademy.com/>
8. <https://tailwindcss.com/>

#### Overall Description

##### Product Perspective:

HydroHitch is envisioned as a centralized platform for water delivery services, bridging the gap between users and vendors. It leverages the MERN stack (MongoDB, Express.js, React Native, Node.js) to ensure scalability, responsiveness, and cross-platform functionality. By integrating advanced features such as real-time tracking, location-based vendor selection, and water quality assurance, HydroHitch positions itself as a modern solution to meet the increasing demand for reliable water delivery.

##### Product Functions:

1. **User Functions:**
   * Register and manage profiles.
   * Search and select vendors based on location and ratings.
   * Place orders for water tankers.
   * View water quality reports.
   * Calculate delivery charges.
   * Track orders in real-time.
   * Submit queries and feedback.
2. **Vendor Functions:**
   * Register and list water tankers.
   * Update tanker availability and pricing.
   * Manage orders and delivery schedules.
   * Submit and update water quality reports.
   * Respond to user queries.
3. **Super Admin Functions:**
   * Oversee user and vendor activities.
   * Validate and monitor water quality reports.
   * Manage queries and escalate issues.
   * Generate analytics and performance reports.
   * Monitor and address delays in order fulfillment.

##### User Classes and Characteristics:

1. **End Users (Consumers):**
   * Individuals and households requiring water delivery.
   * Businesses and offices with bulk water needs.
   * Characteristics: Varying technical proficiency, requiring an intuitive and responsive interface.
2. **Vendors:**
   * Water suppliers managing tankers and delivery logistics.
   * Characteristics: Intermediate technical skills; need tools for order management and quality reporting.
3. **Super Administrators:**
   * Platform managers overseeing operations and ensuring quality and compliance.
   * Characteristics: Technically proficient, requiring access to analytics and system controls.

##### Operating Environment:

1. **Client Devices:**
   * Mobile devices (Android and iOS) for users and vendors.
   * Desktop browsers for super admins.
2. **Server Environment:**
   * Cloud-based hosting for backend services and databases.
   * Utilizes AWS for scalability and reliability.
3. **APIs and Tools:**
   * Google Maps API for location tracking.
   * Secure payment gateway for transactions.

##### Design and Implementation Constraints:

1. **Budget Limitations:** Restricted budget for hosting, development tools, and marketing.
2. **Time Constraints:** Defined timelines for FYP 1 and FYP 2 deliverables.
3. **Platform Requirements:** Cross-platform compatibility using React Native.
4. **Data Security:** Adherence to data protection regulations for user privacy.
5. **Scalability:** Backend design must support future growth and high traffic.

##### User Documentation:

1. **User Guide:** Detailed instructions for end users, including registration, ordering, and tracking features.
2. **Vendor Manual:** Guides vendors on listing management, order handling, and quality reporting.
3. **Admin Documentation:** Comprehensive reference for super admins, including system controls and analytics.
4. **FAQs:** Common troubleshooting and usage tips for all user classes.

##### Assumptions and Dependencies:

###### Assumptions

* + - Users and vendors will have reliable internet access.
    - The application will be compatible with most modern smartphones and desktops.
    - Vendors will provide accurate water quality data.

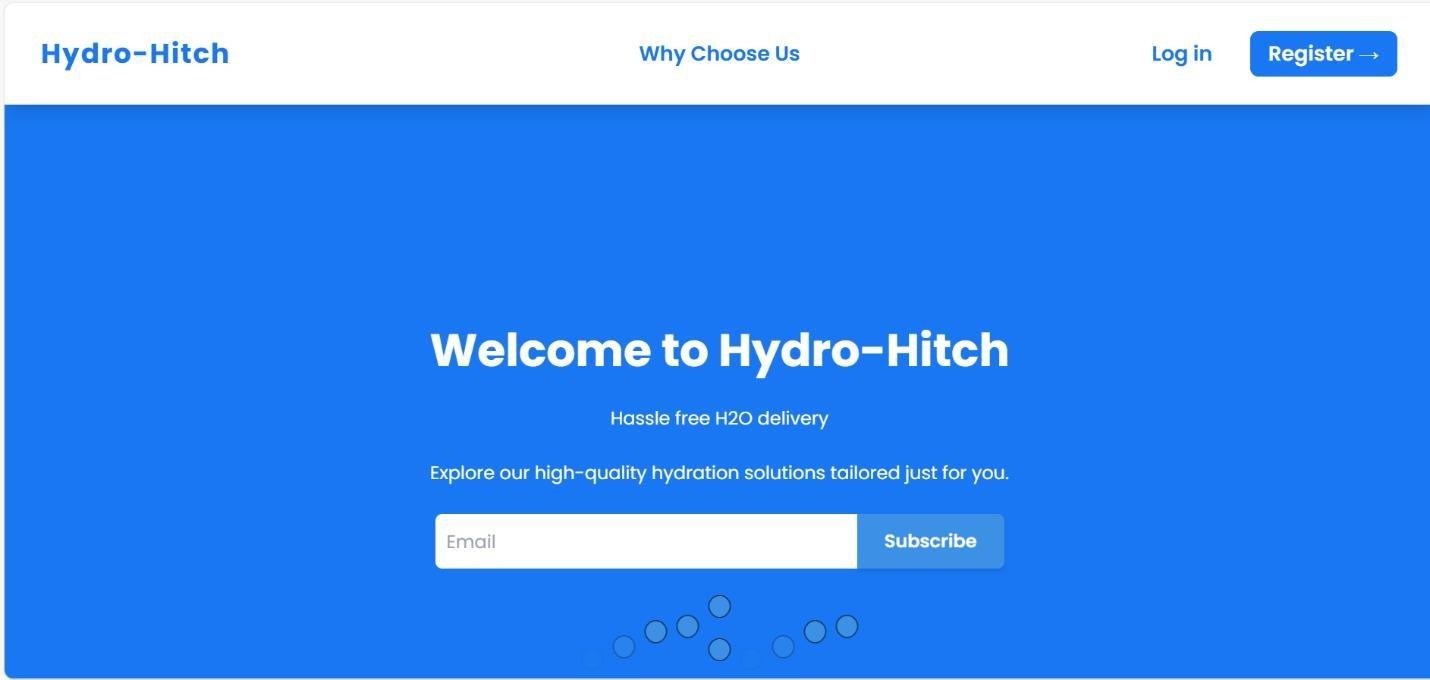
###### Dependencies

* + - Google Maps API for location-based services.
    - Payment gateways for secure transaction handling.
    - Hosting and database services (AWS and MongoDB) for application deployment.
    - User feedback for iterative improvements.

#### External Interface Requirements

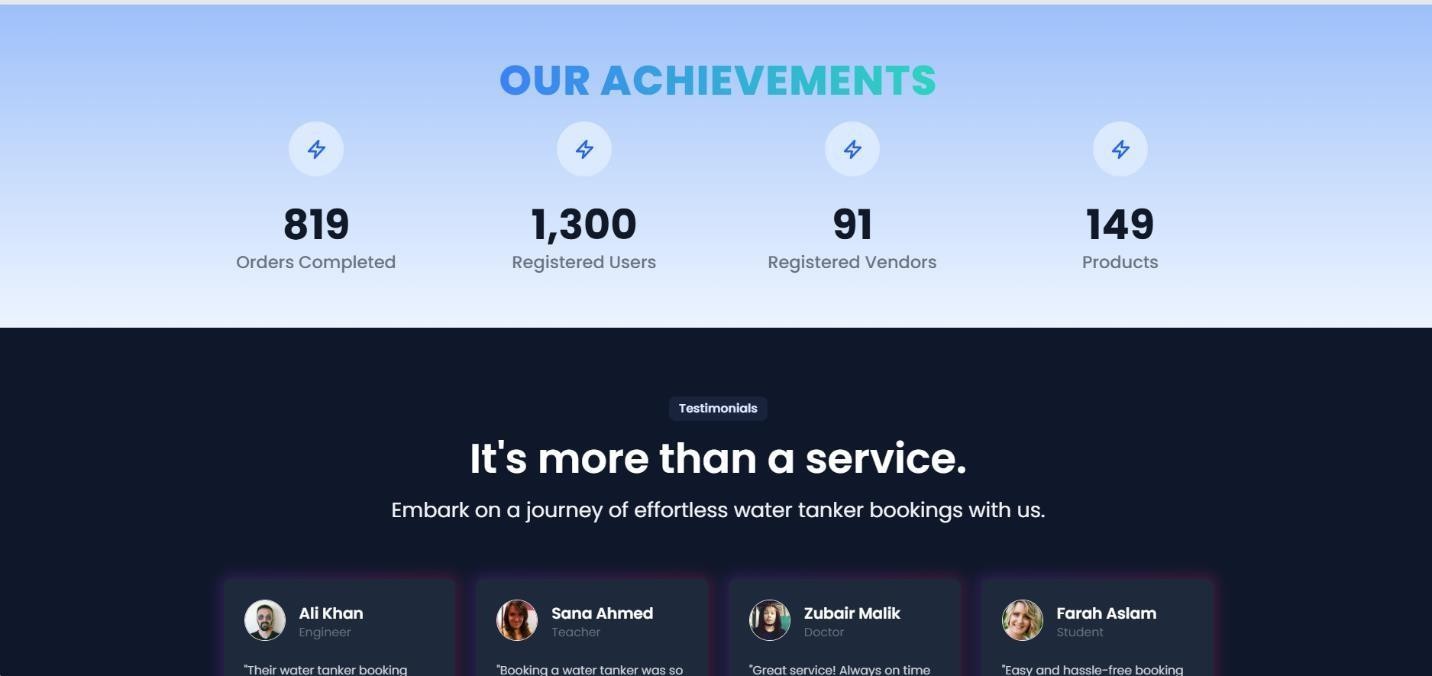
##### User Interfaces:

1. **User Panel:**
   * A mobile-friendly interface for users to browse vendors, place orders, view quality reports, and track deliveries.
   * Features intuitive navigation with minimal learning curve for varying technical proficiency levels.
2. **Vendor Panel:**
   * A dashboard for vendors to manage orders, submit water quality data, and track service performance.
   * Mobile and web-based interfaces for convenience.
3. **Admin Panel:**
   * A comprehensive control panel for super admins to monitor system operations, validate quality reports, and generate analytics.
   * Desktop interface with advanced tools for system oversight.

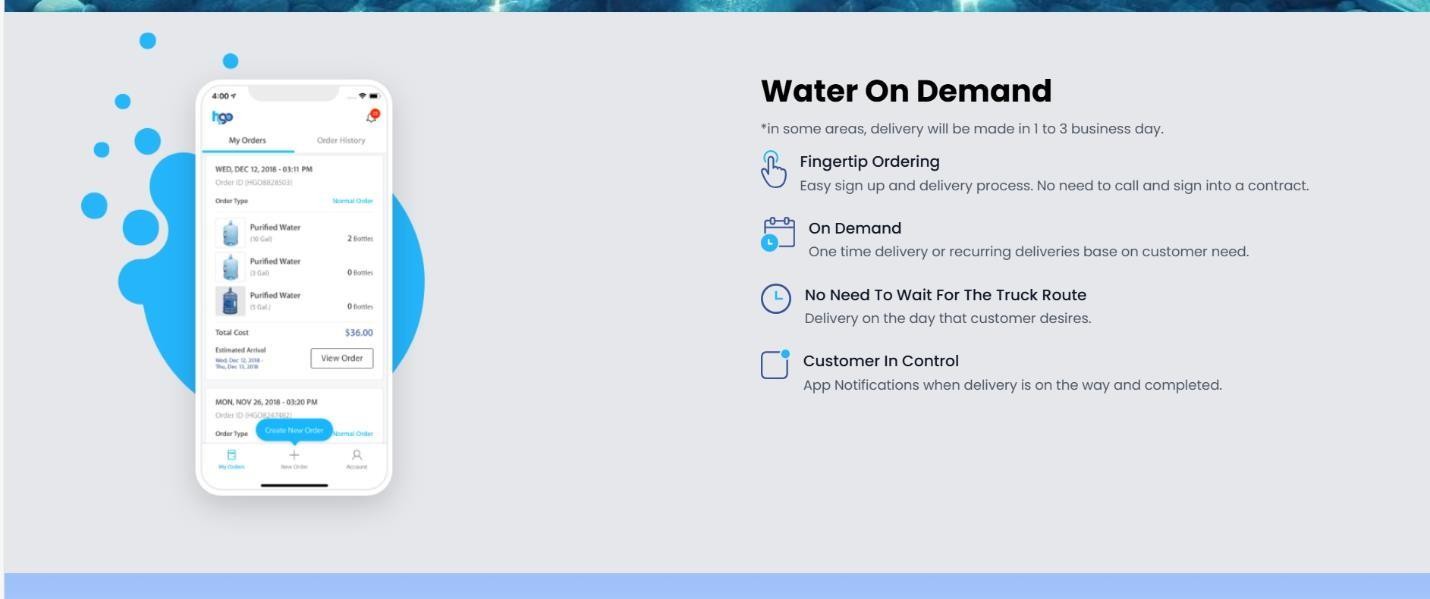
**Landing Page**

*****Figure 1. Homepage*

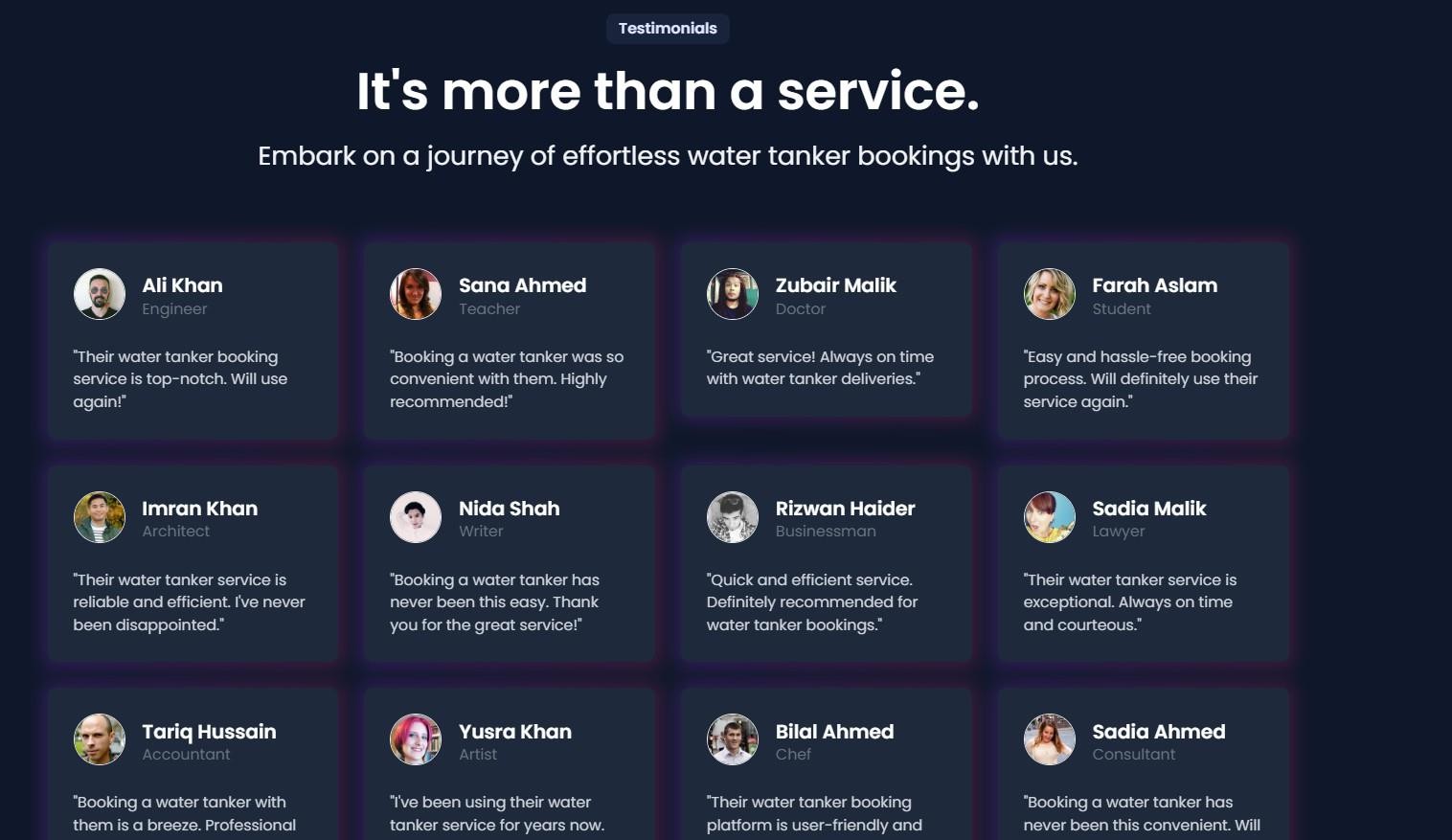


****

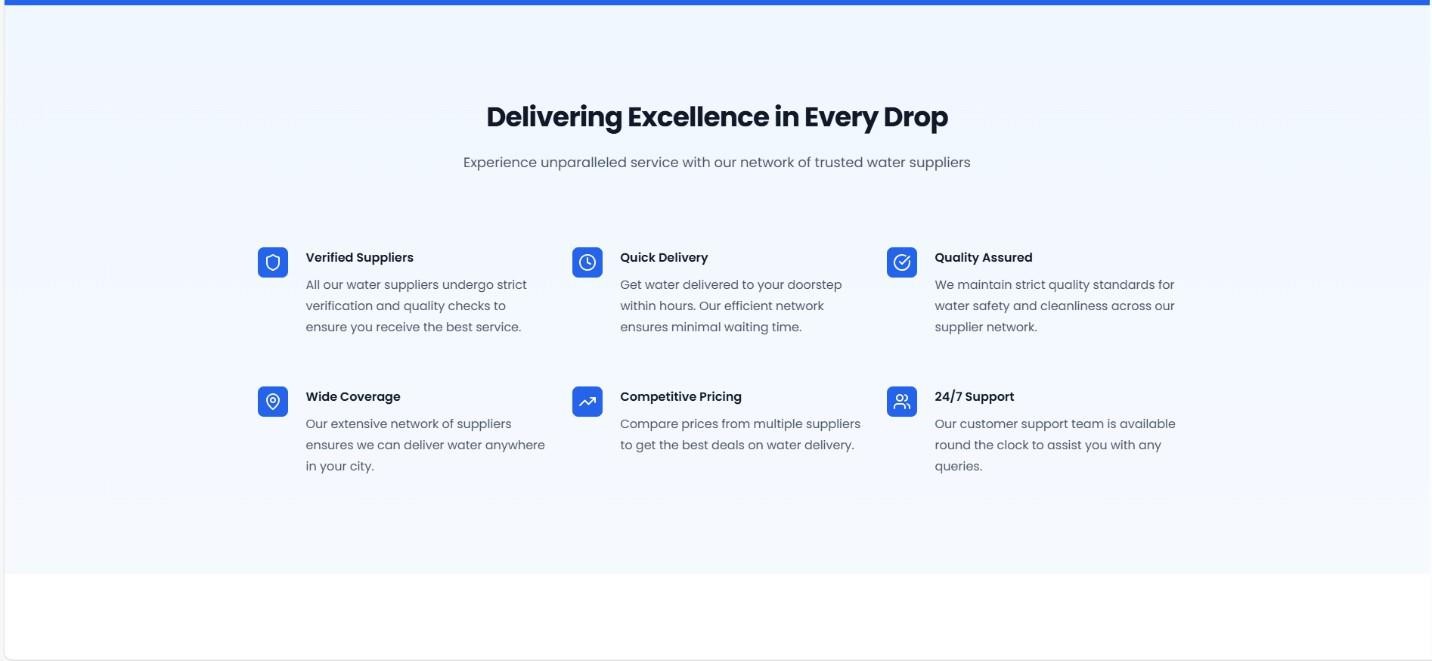
*Figure 3. Homepage3*

****

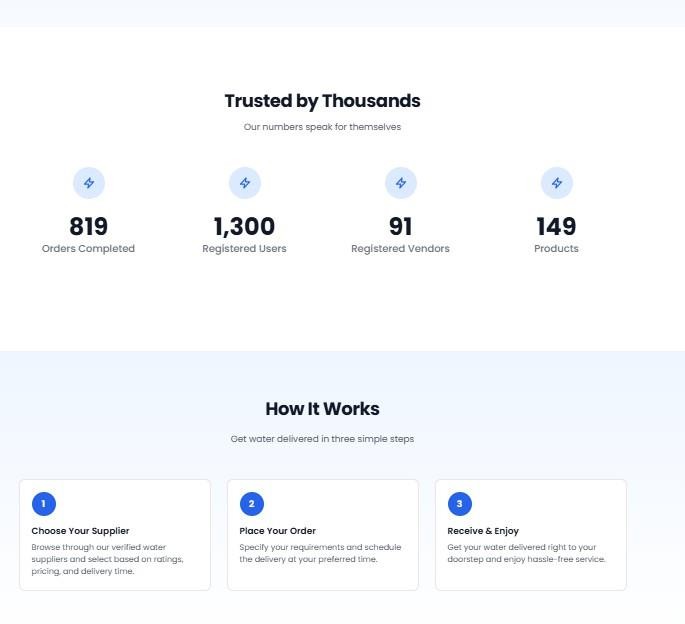
*Figure 4. HomePage4*

****

*Figure 5. Homepage 5*

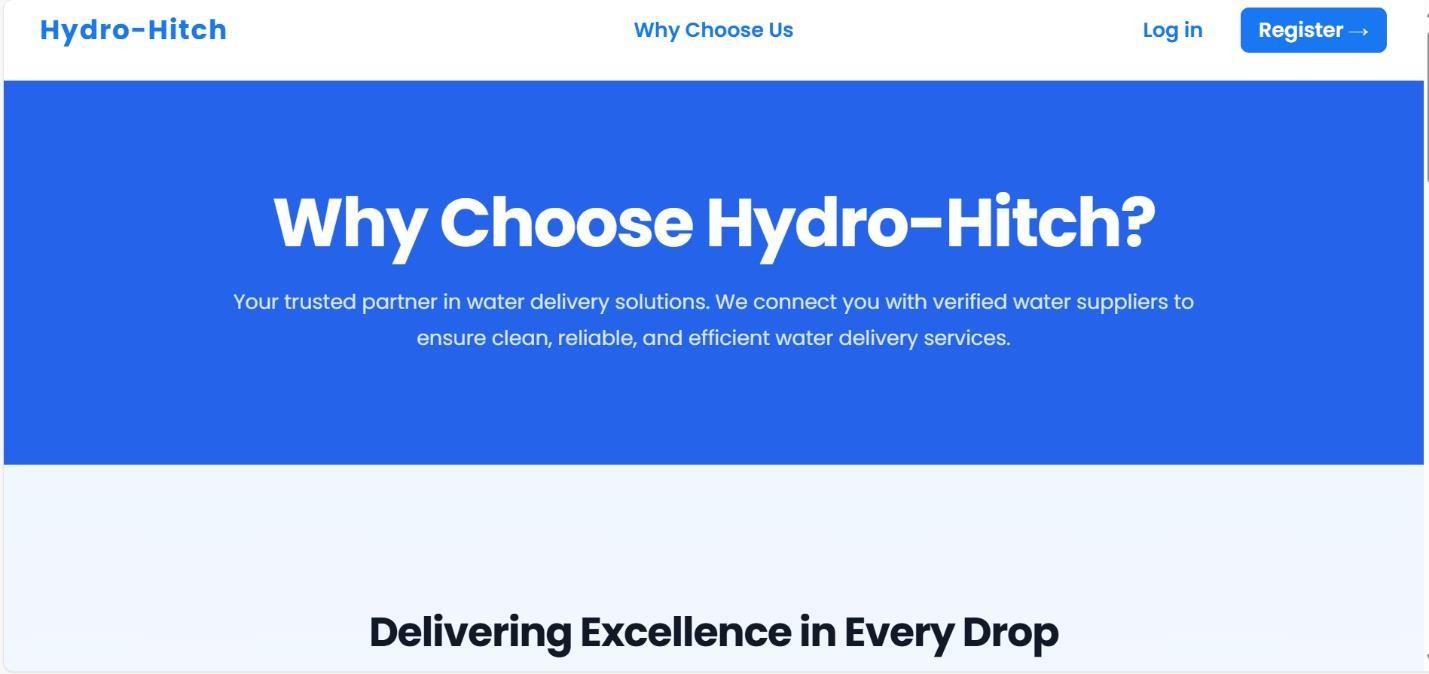
****

*F Figure 6. Homepage 6*

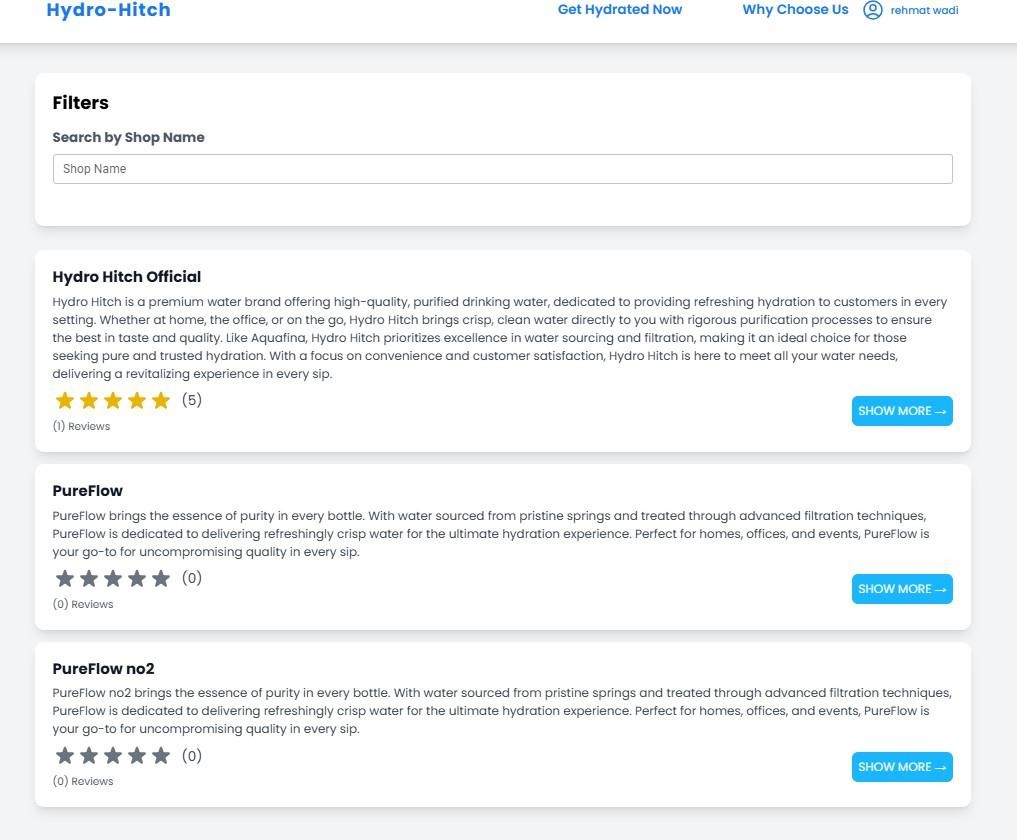
****

*Figure7. Homepage7*

*Figure8. Homepage8*

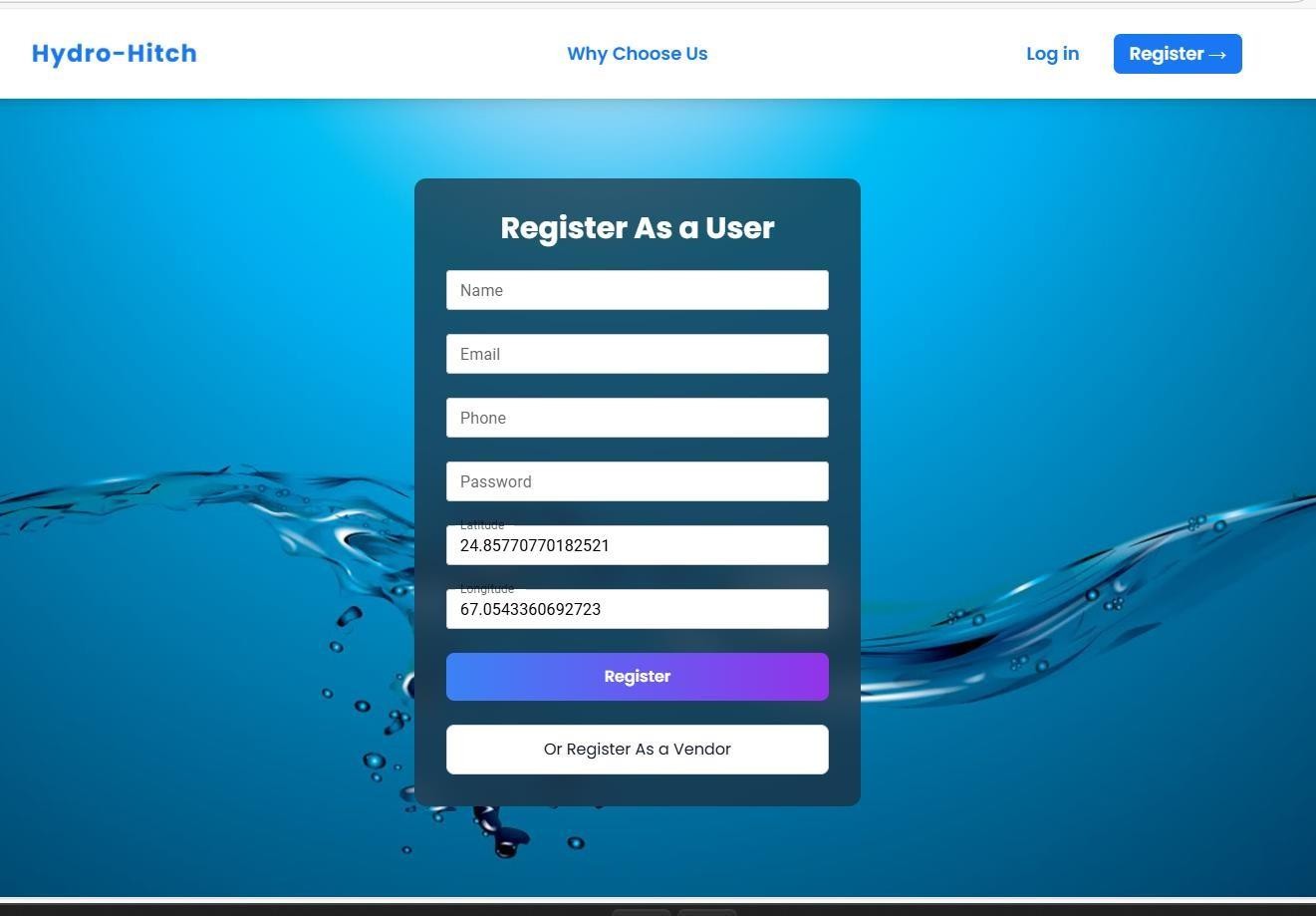
****

**Vendor Listing**



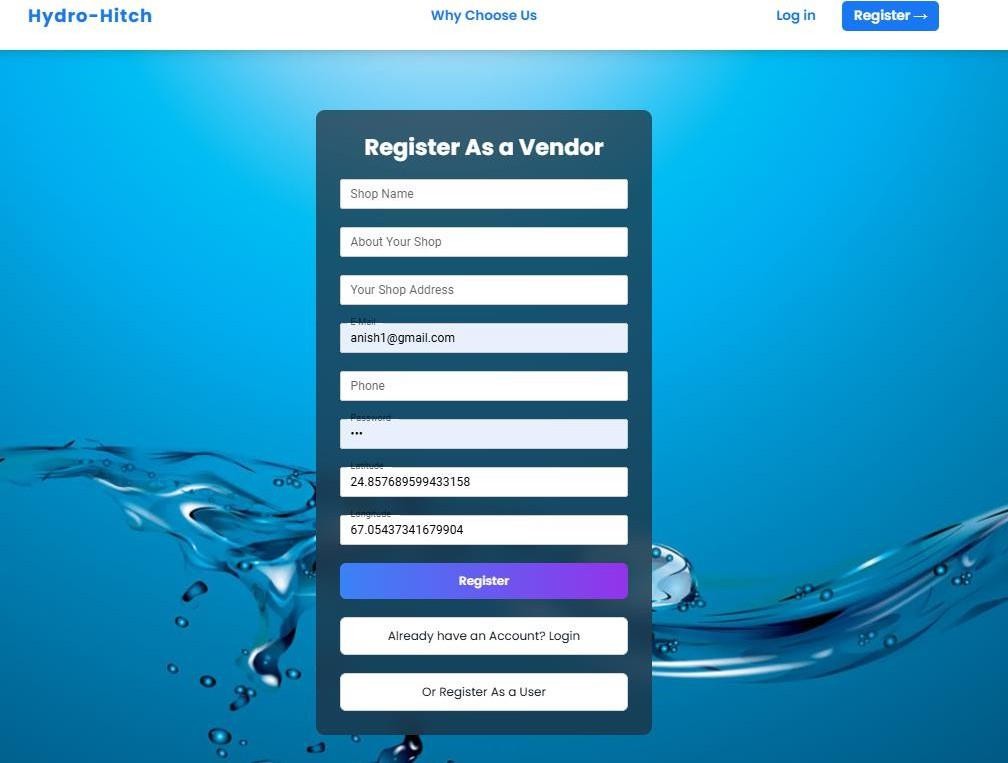
*Figure 9. Vendor Listing*

##### User Registration

****

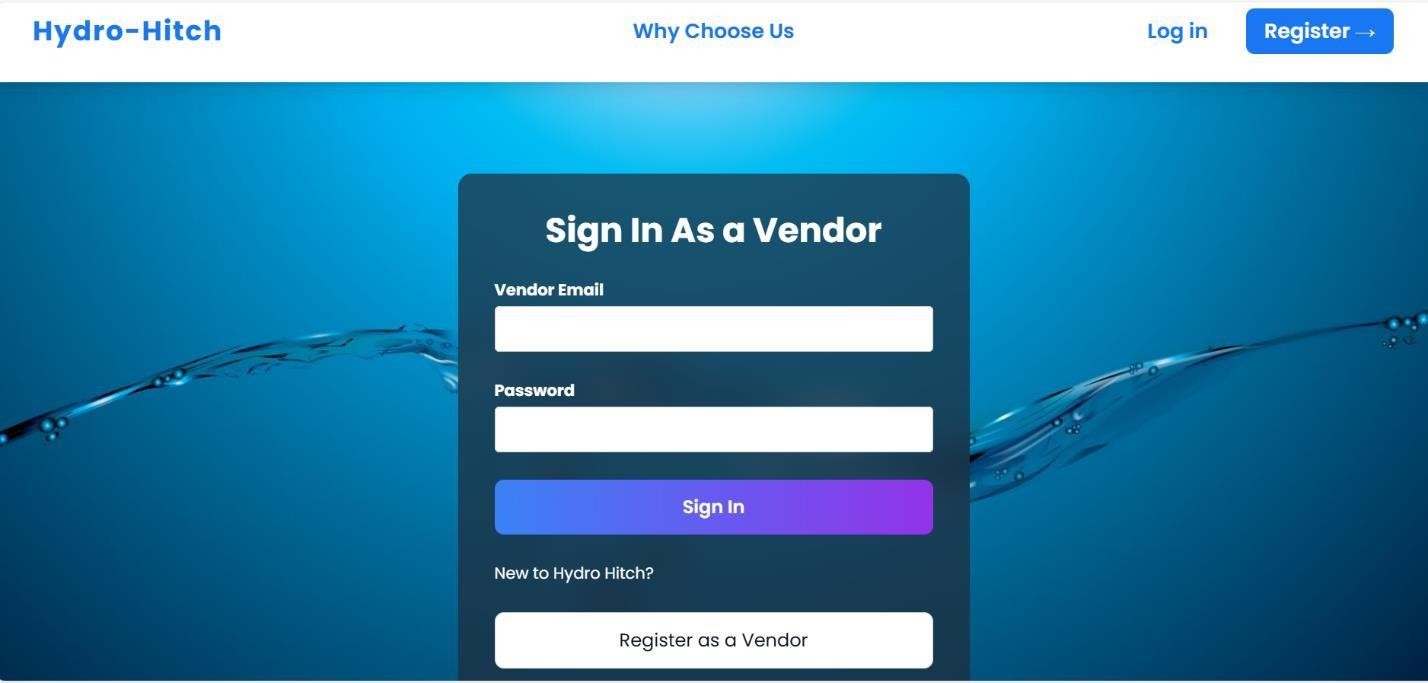
*Figure 10. User Registration*

##### Vendor Registration

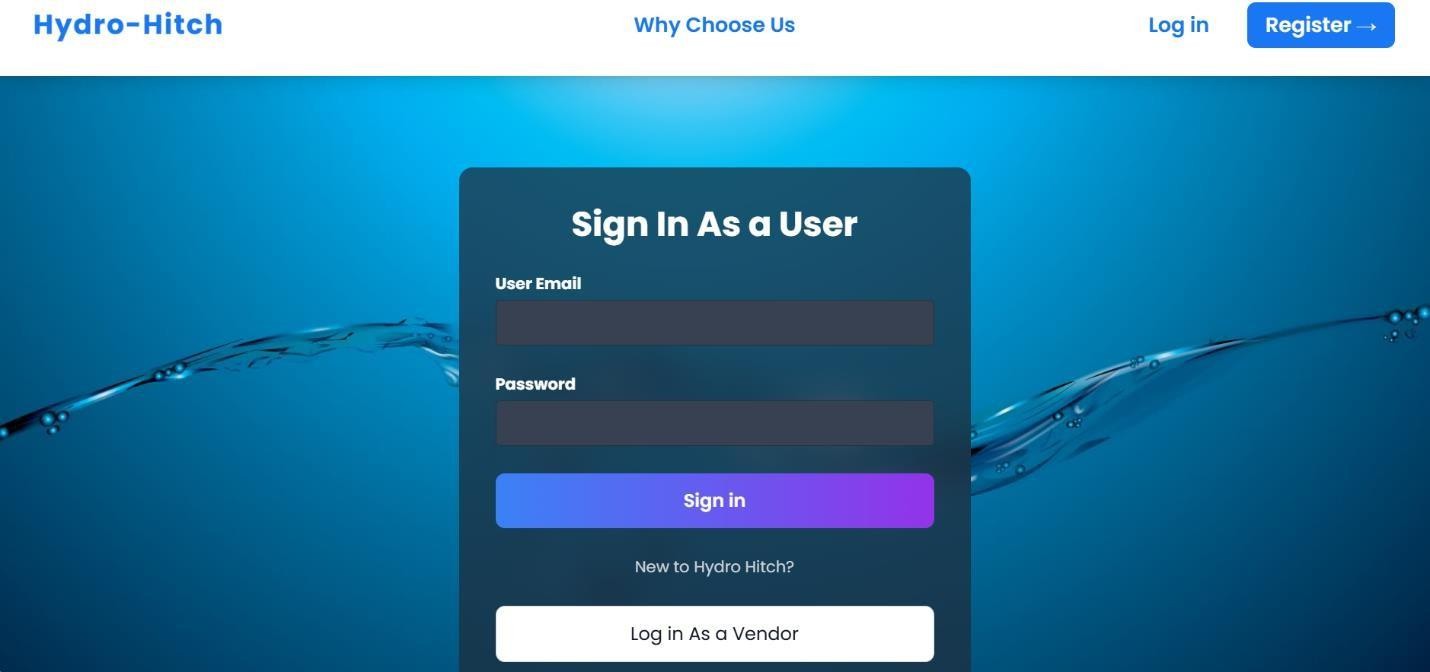
****

*Figure 11. Vendor Registration*

**Login Screens**

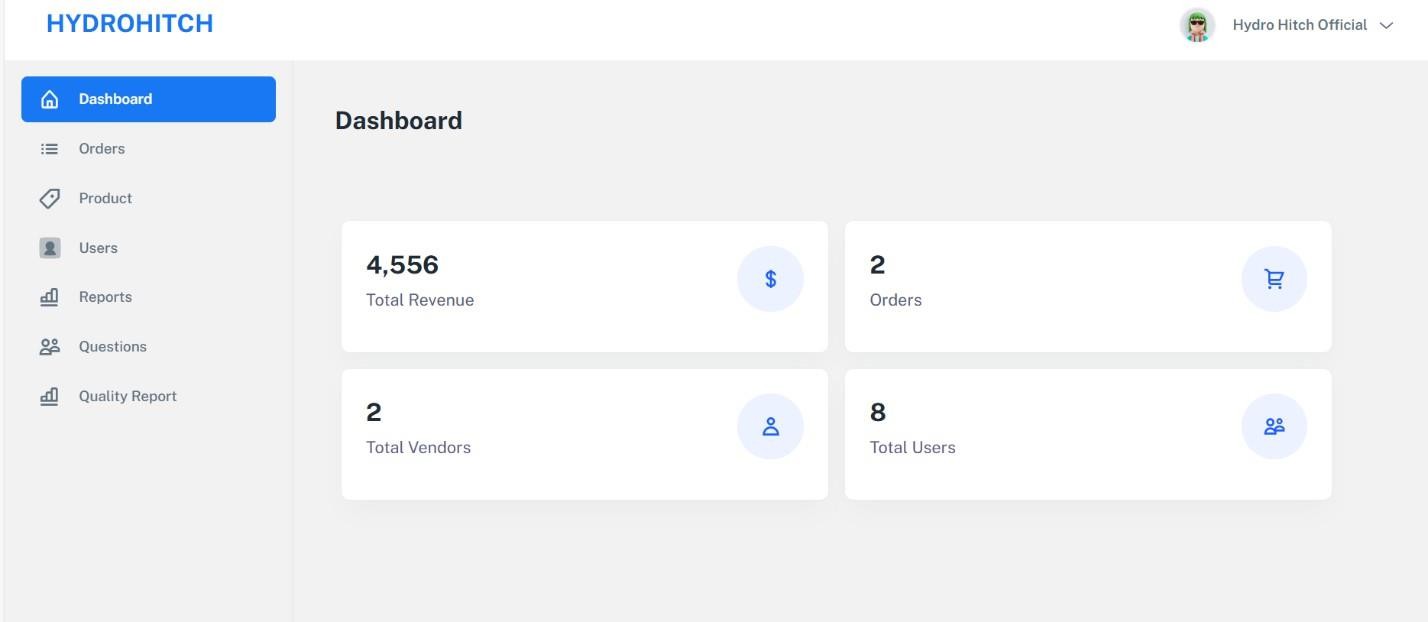


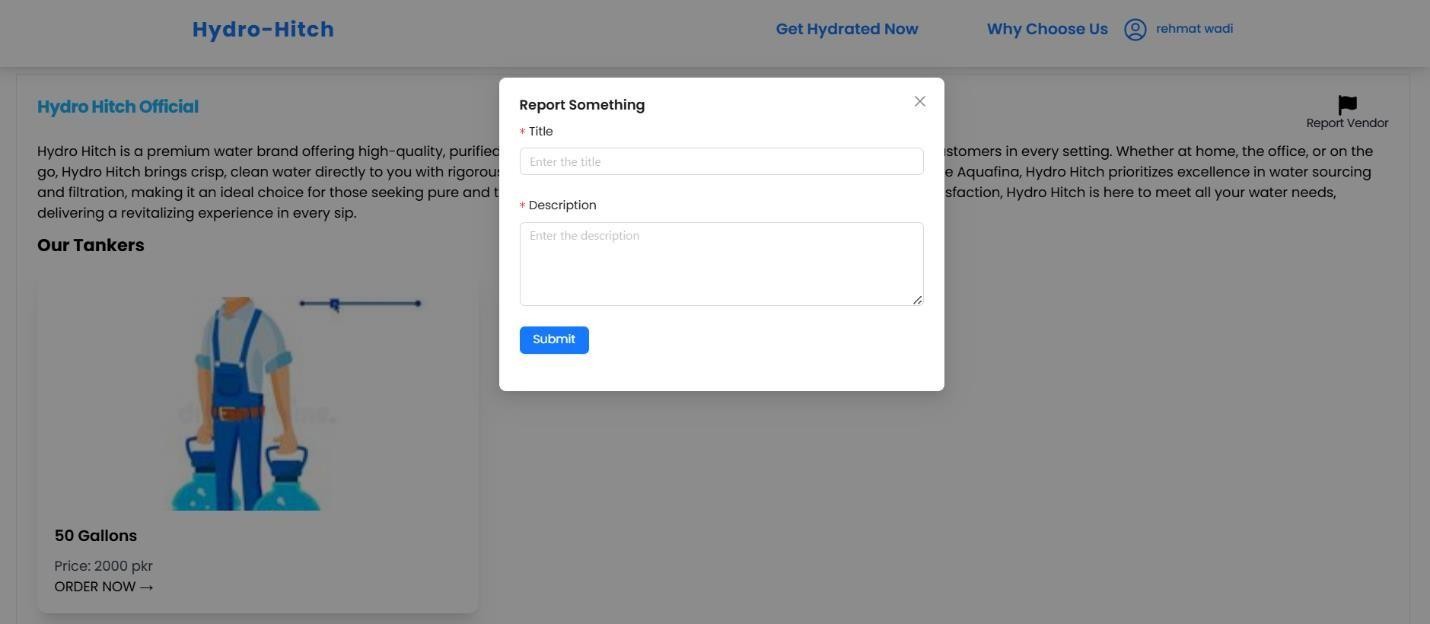
*Figure 12. Vendor Login*

****

*Figure 13. User Login*

##### Admin dashboard



 *Figure 14. Admin Dashboard*

*Figure 15. Vendor Report*

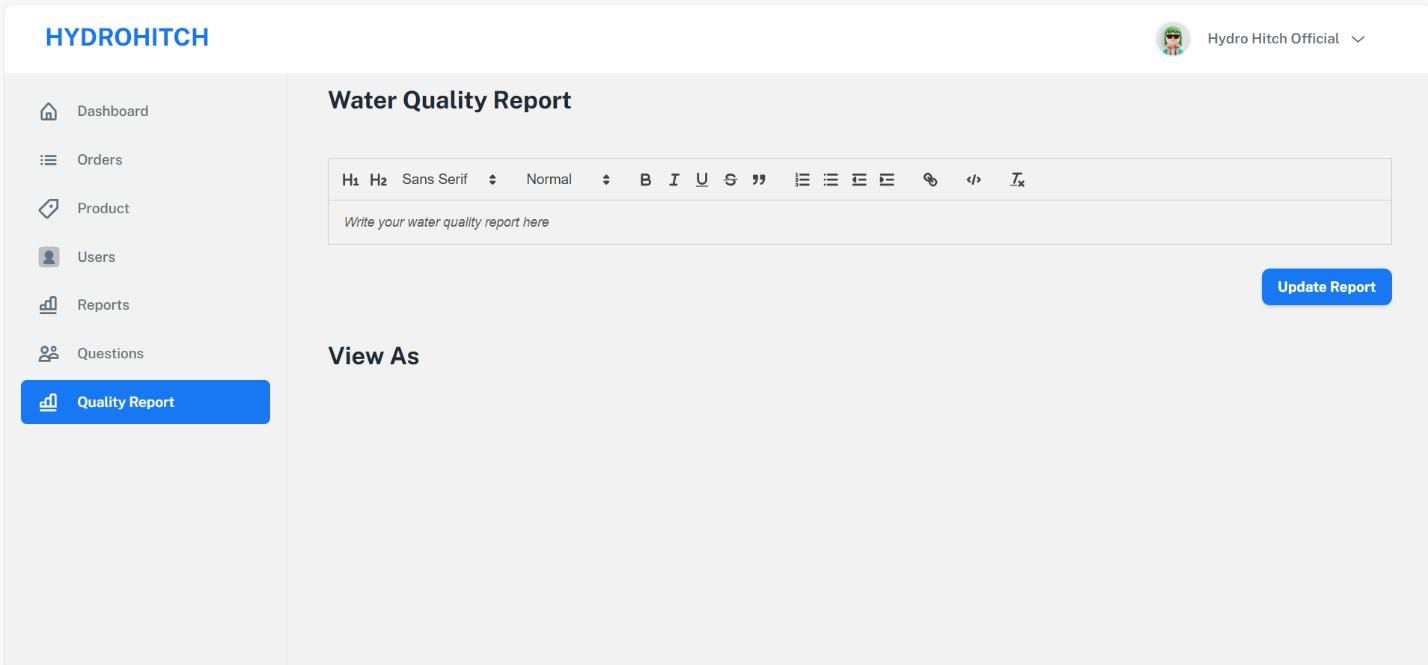
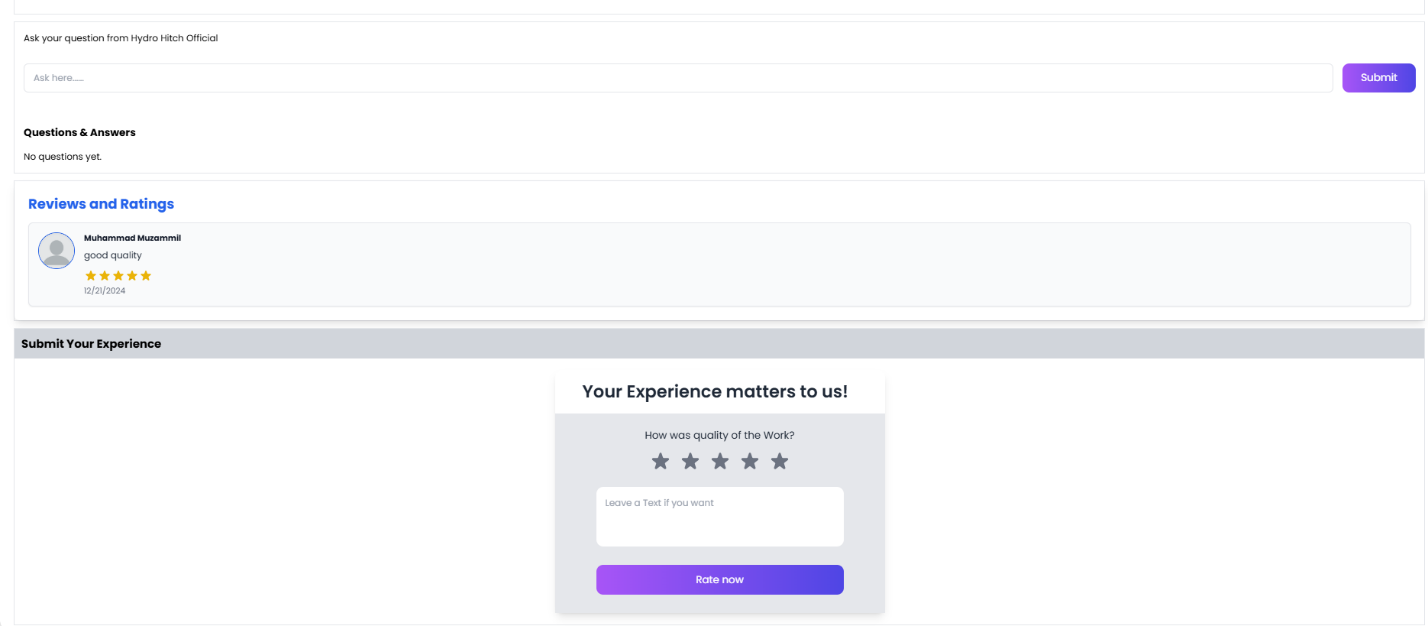
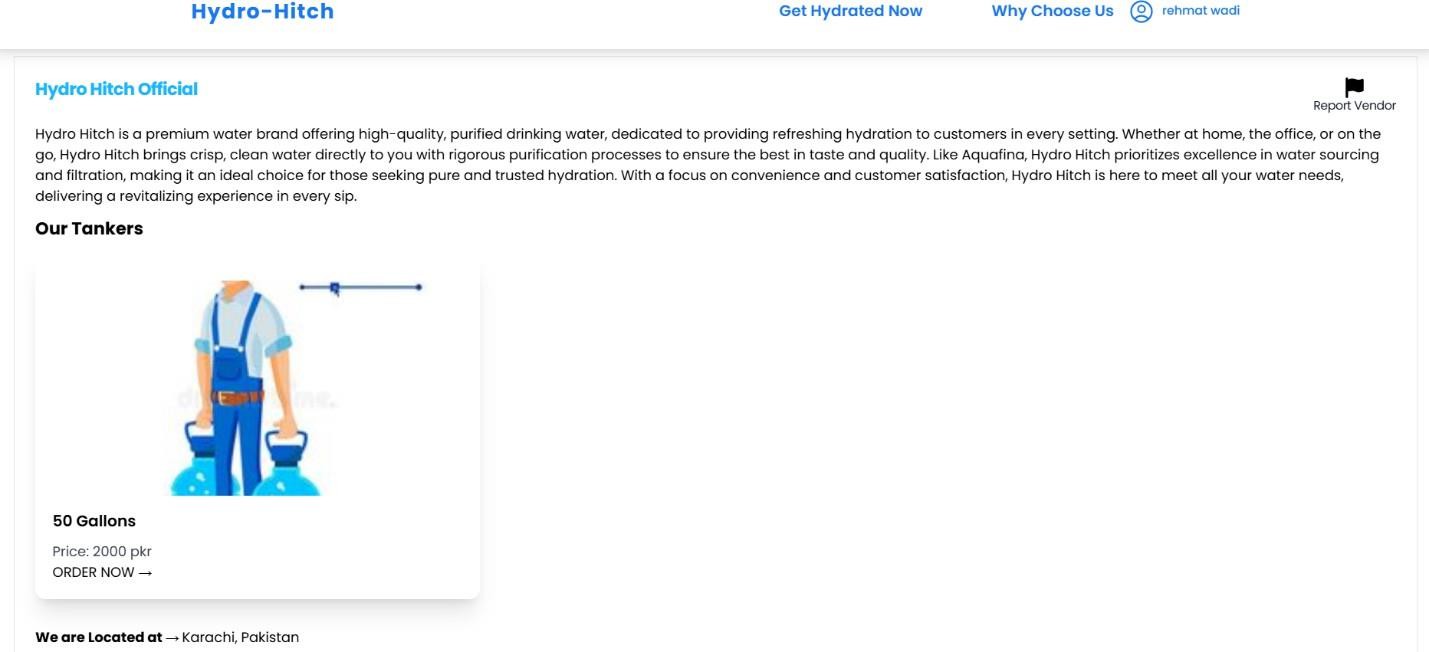


Figure 16. Water Quality Report

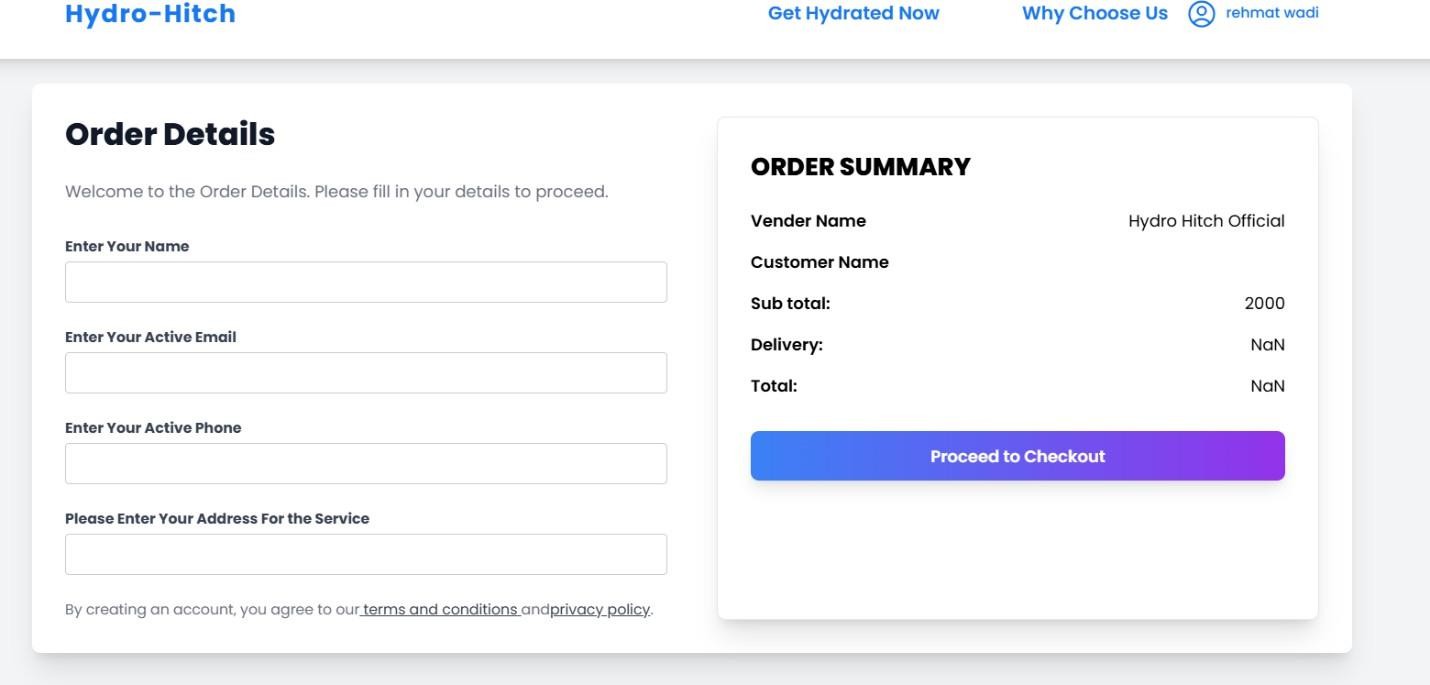
****

*Figure 17. Reviews*

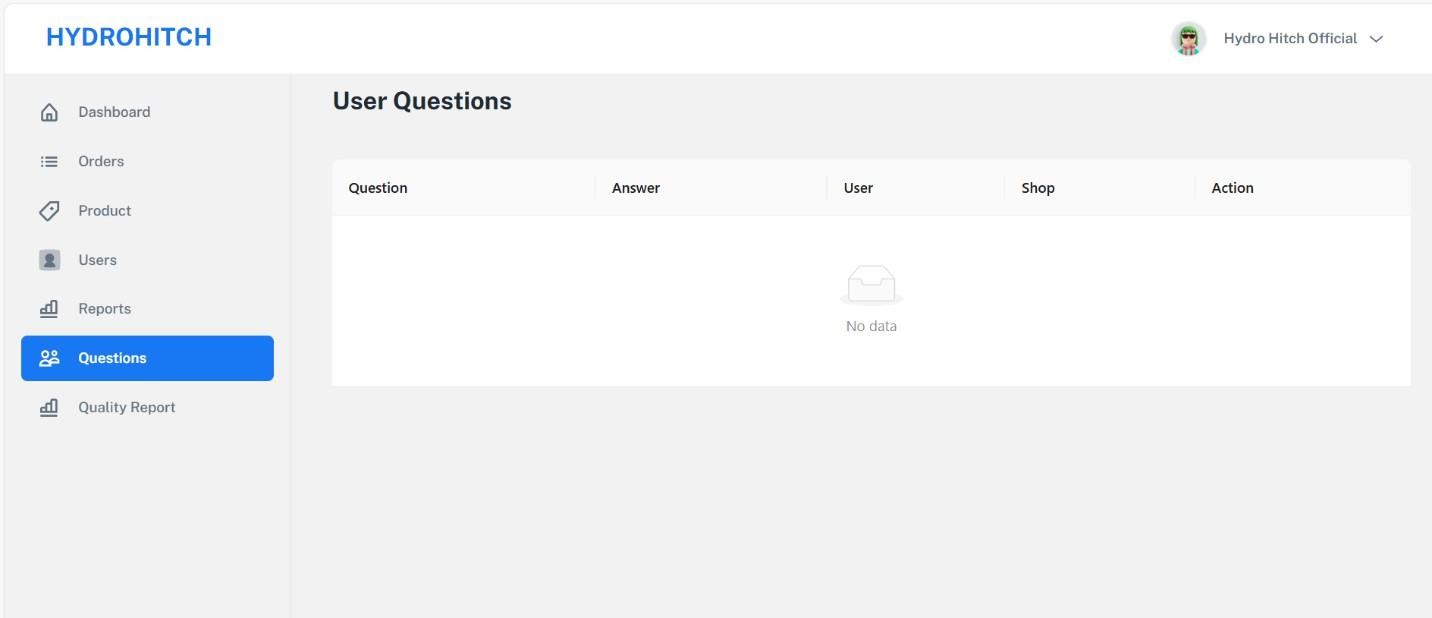
##### 

****

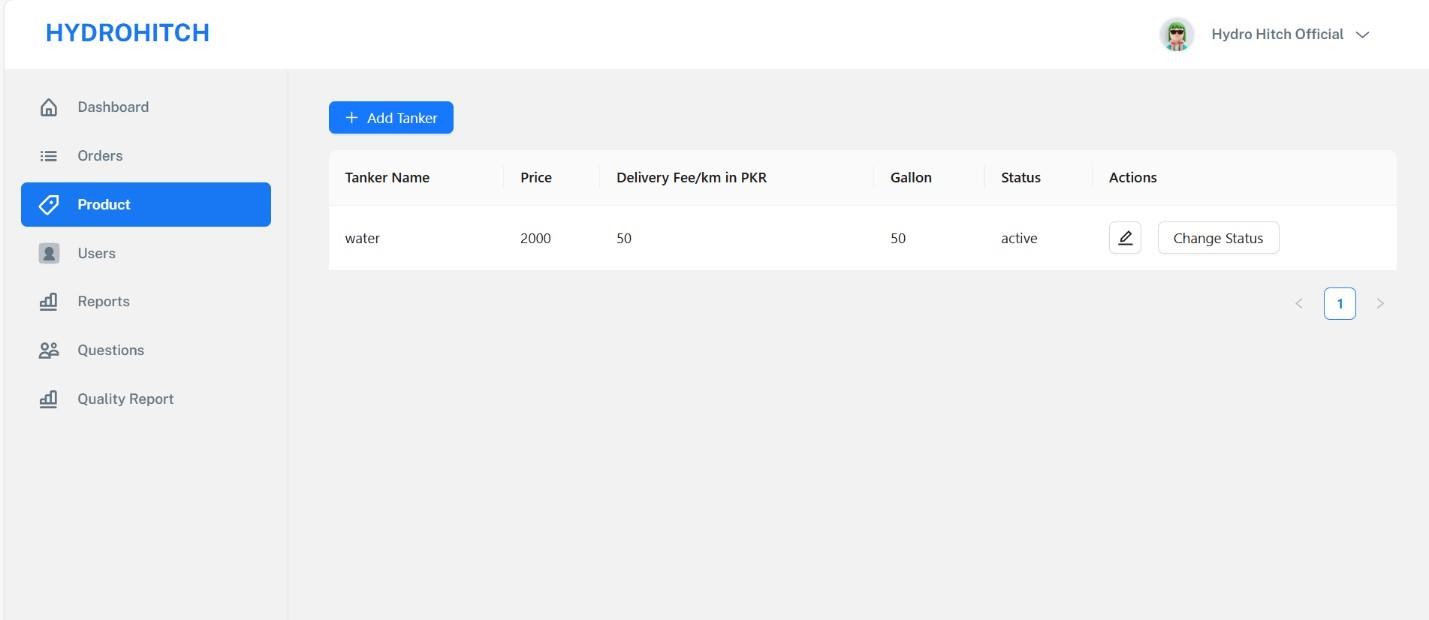
##### *Figure 18. Vendor List*

****

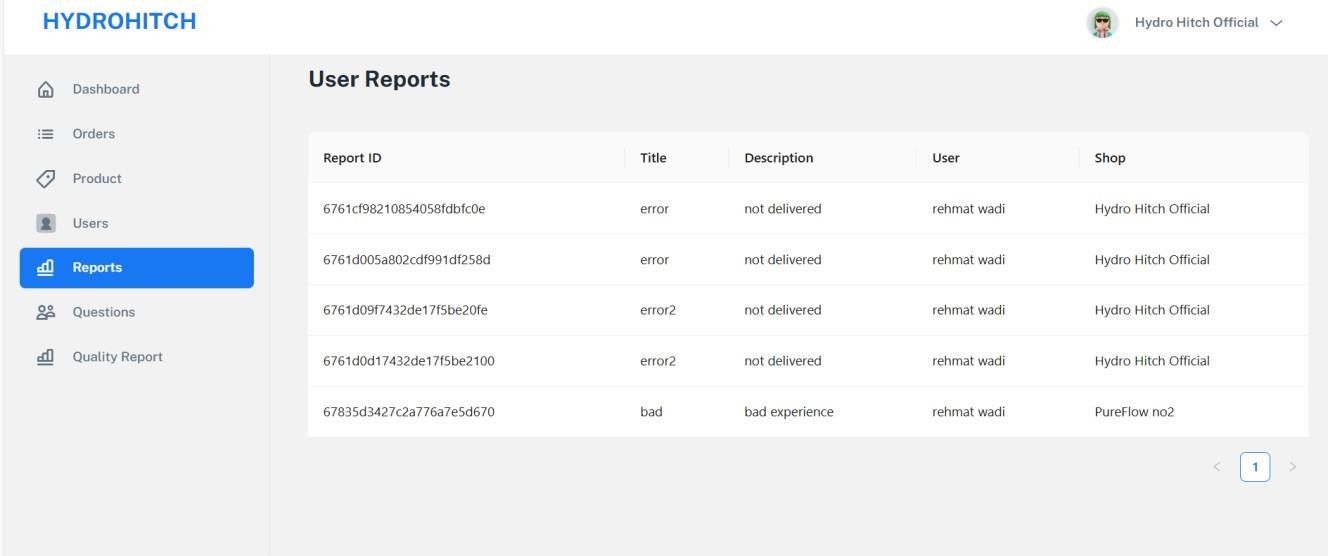
##### *Figure 18. Order Detail*

****

*Figure 20. UserQuery*

****

*Figure 21. Add Products*

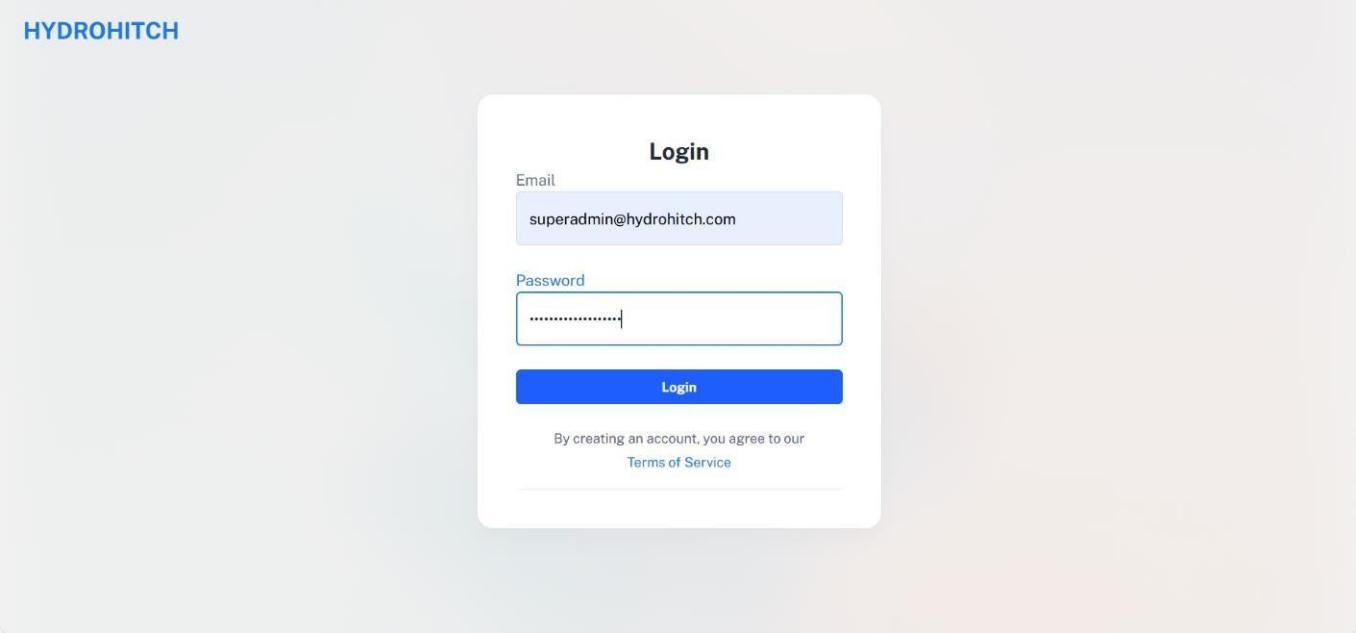
****

*Figure 22. User Report*

##### 

*Figure 23. User Management*

##### *Figure 24. Order Management*



*Figure 25. Admin Login*

##### Hardware Interfaces:

1. **Mobile Devices:**
   * Compatible with smartphones and tablets running Android or iOS.
2. **Servers:**
   * Cloud-hosted backend servers to handle API requests and data storage.
3. **GPS Hardware:**
   * Integration with GPS-enabled devices for real-time location tracking of water tankers.

##### Software Interfaces:

1. **Google Maps API:**
   * For location-based vendor selection and route optimization.
2. **Payment Gateway:**
   * Integration with secure payment systems to facilitate transactions.
3. **Authentication Services:**
   * APIs for user authentication and authorization.
4. **Database:**
   * MongoDB for managing user, vendor, and order data.

##### Communication Interfaces:

1. **Push Notifications:**
   * Real-time updates on order status and notifications for offers or alerts.
2. **Email and SMS:**
   * For account verification, order confirmations, and system communications.
3. **API Communication:**
   * RESTful APIs for interaction between the frontend and backend services.
4. **Real-Time Messaging:**
   * A messaging feature for users to communicate with vendors or administrators directly within the app.

#### System Features

System Features of **HYDRO** App:

##### User Registration:

* + 1. Description:

User sign-up feature allows new users to create an account by providing required details such as their name, email, password, and other optional information. This is the first step for users to access the platform

Stimulus/Response Sequences:

|  |
| --- |
| **Use Case Name:** Sign Up |
| **Description:** User can sign up for an account on “HYDRO.” |
| **Actors:** User (End User) / System (HYDRO Application) |
| **Goal:** To successfully create a user account and provide access to the application’s features. |
| **Pre-Conditions:**   * User must have access to a device with an internet connection. * User must have a valid email address and/or phone number. * The HYDRO application must be operational. |
| **Basic Course of Events /Main Flow**   1. **User Action (Step 1):**   The user selects the "Sign Up" option on the HYDRO application.  **System Response:**  The system displays a registration form requesting required information (e.g., full name, email, phone number, and password).   1. **User Action (Step 2)**   The user fills in the required fields and clicks the "Submit" or "Register" button.  **System Response:**  The system validates the inputs and checks for duplicate accounts.   1. **User Action (Step 3):**   The user verifies their email or phone number via a confirmation code sent by the system.  **System Response:**  The system confirms the verification and creates the user account. |
| **Alternate Flow (Invalid Inputs)**  **User Action**:  The user enters invalid or incomplete information (e.g., invalid email format, weak password, or missing fields).  **System Response**:  The system displays error messages indicating the specific issues and prompts the user to correct Them. |
| **Exception Flow (System Errors)**  **System Failure:**  The HYDRO application encounters a system issue (e.g., server downtime or connectivity problems).  **System Response:**  The system displays a generic error message (e.g., "We are experiencing technical difficulties. Please try again later.") and logs the error for further analysis. |

|  |
| --- |
| **Post Conditions:**   1. A new user account is created and stored in the database. 2. The user is redirected to the login page or their dashboard after successful registration. 3. The user receives a confirmation email or SMS for account creation. |

**Table 1. User signup Stimulus/ Response sequence**

* + 1. Functional Requirements:

|  |  |  |  |
| --- | --- | --- | --- |
| **Req.ID** | **Description** | **Priority** | **Status** |
| Req 1 | User must provide a valid email address | High | Pending |
| Req 2 | Password must meet security standards | High | Pending |
| Req 3 | Verification code must be unique. | High | Pending |
| Req 4 | System must handle duplicate emails | Medium | Pending |
| Req 5 | Welcome email must be sent post-signup | Medium | Pending |
| Req 6 | Provide feedback for invalid inputs | Low | Pending |
| Req 7 | Support password recovery mechanism | High | Pending |

**Table 2. User signup Functional Requirements**

##### Login:

* + 1. Description:

The User login feature allows existing users to access their accounts by entering their registered email/phone number and password. This is essential for users to gain access to the platform’s features and manage their bike services.

* + 1. Stimulus/Response Sequences:

|  |
| --- |
| **Use Case Name:** Login |
| **Description:** User can Login to an account on “HYDRO”. |
| **Actors:** User (End User) / System (HYDRO Application) |
| **Goal:** To login on the platform and get app experience**.** |
| **Pre-Conditions:** User must have an account on the platform. |
| **Basic Course of Events /Main Flow**  **1. User Action (Step 1):**  The user opens the HYDRO app and selects the Login option.  **System Response:**  The system displays the login form, asking for the user’s registered email or phone number and password. |

|  |
| --- |
| 1. **User Action (Step 2)**   The user enters their registered email/phone number and password, then clicks the Login button.  **System Response:**  The system validates the entered credentials, checking if the email/phone number exists and if the password matches.   1. **User Action (Step 3):**   If the credentials are valid, the user confirms the login.  **System Response:**  The system successfully logs in the user, creates a session, and redirects them to the app's main dashboard or home screen. |
| **Alternate Flow (Invalid Inputs)**  **User Action**:  The user selects the Forgot Password option due to multiple failed attempts or forgotten credentials.  **System Response**:  The system prompts the user to enter their registered email or phone number and provides in instructions for resetting the password (e.g., sending a password reset link or OTP verification). |
| **Exception Flow (System Errors) System Failure:**  The user submits the login credentials, but there is a network failure preventing communication with  the server.  **System Response:**  The system displays an error message like "Network error. Please check your connection and try again." The login process is paused until a stable connection is restored. |
| **Post Conditions:** User is logged into the account. |

**Table 3. User login Stimulus/ Response sequence**

* + 1. Functional Requirements:

|  |  |  |  |
| --- | --- | --- | --- |
| **Req.ID** | **Description** | **Priority** | **Status** |
| Req 1 | The system shall display a login form with fields for email/phone number and password. | High | Pending |
| Req 2 | The system shall validate the entered email/phone number and  password before allowing form submission. | High | Pending |
| Req 3 | The system shall check if the provided email/phone number exists in the database. | High | Pending |
| Req 4 | The system shall provide an option for users to reset their password if they forget it. | Medium | Pending |
| Req 5 | The system shall log failed login attempts for security auditing  purposes. | Medium | Pending |
| Req 6 | The system shall provide a “Remember Me” option to allow users to  stay logged in on their devices. | Low | Pending |
| Req 7 | The system shall ensure that all user credentials are securely  transmitted and stored. | High | Pending |

**Table 4. User login Functional Requirements**

##### Tanker Management:

* + 1. Description:

Tanker Management allows users to sign up for an account on the HydroHitch platform, enabling them to access water delivery services and manage orders efficiently.

* + 1. Stimulus/Response Sequences:

|  |
| --- |
| **Use Case Name:** Tanker management |
| **Description:** Users can sign up for an account on “HYDRO” and manage water tanker orders seamlessly. |
| **Actors:** User, System, Vendor, Super Admin |
| **Goal:** To enable users to create accounts and manage water tanker services effectively, including ordering, tracking, and managing queries. |
| **Pre-Conditions:**   * User must have internet access. * The HYDRO application must be installed on the user’s device or accessible via the web. |
| **Basic Course of Events / Main Flow**   1. **User Action (Step 1):**   The user opens the HYDRO application and selects the option to sign up.  **System Response:**  The system displays a registration form requesting user details such as name, email, password, and location.   1. **User Action (Step 2):**   The user fills in the form and submits the registration details.  **System Response:**  system validates the input and creates a user account, sending a confirmation email or SMS.   1. **User Action (Step 3):**   The user logs into the system and navigates to the Tanker Management section.  **System Response:**  The system displays available tankers, pricing, and quality reports, allowing the user to place an order. |
| **Alternate Flow (Invalid Inputs):**  **User Action (Invalid Input):**  The user attempts to add or update a bike with invalid input, such as an incorrect registration number format, missing mandatory fields, or invalid model year.  **System Response:**  The system detects the invalid input and displays error messages indicating the specific issues, such as "Invalid registration number format," "Model year cannot be empty," or "Please fill in all required fields." The user is prompted to correct the errors before proceeding |

**Post-Condition:** The user's bike information is successfully saved or updated in the system, and the user is notified of the completion.

**Exception Flow (System Errors):**

**System Failure:**

The system encounters an error while adding, updating, or retrieving bike information due to server downtime, database connection issues, or other internal errors.

**System Response:**

The system displays a generic error message, such as "An error occurred while processing your request. Please try again later." The error is logged for further diagnosis, and the user is advised to retry or contact support if the problem persists.

**Table 5. Tanker management Stimulus/ Response sequence**

* + 1. Functional Requirements:

|  |  |  |  |
| --- | --- | --- | --- |
| **Req.ID** | **Description** | **Priority** | **Status** |
| Req 1 | The system shall allow users to add a new bike by entering details such as model, year, and registration number. . | High | Pending |
| Req 2 | The system shall display a list of all bikes added by the user, along  with their relevant details. | High | Pending |
| Req 3 | The system shall validate all bike details before allowing submission to ensure accuracy (e.g., valid registration number format). | High | Pending |
| Req 4 | The system shall allow users to edit bike information, such as  updating the registration number or year. | Medium | Pending |
| Req 5 | The system shall provide users with the ability to delete a bike from  their profile. | Medium | Pending |
| Req 6 | The system shall maintain a service history for each bike, showing all  past services performed. | High | Pending |
| Req 7 | The system shall allow users to view and manage multiple bikes in their profile. | Medium | Pending |

**Table 2. Tanker management Functional Requirements**

##### Feedback System:

* + 1. Description:

The Feedback System allows users to submit queries or suggestions and receive responses from vendors or administrators within the HydroHitch platform, enhancing user engagement and issue resolution.

* + 1. Stimulus/Response Sequences:

|  |
| --- |
| **Use Case Name:** Feedback System |
| **Description:** Users can provide feedback, report issues, or ask queries about the services. |
| **Actors:** User, Vendor, Super Admin |
| **Goal:** To facilitate effective communication and issue resolution between users and service providers. |

|  |
| --- |
| **Pre-Conditions:**   * The user must be logged into their HydroHitch account. * A valid network connection is required. |
| **Basic Course of Events / Main Flow**   1. **User Action (Step 1):**   Navigate to the feedback section and click on "Submit Feedback."  **System Response:**  Displays a feedback form with fields for query description, category, and optional attachment.   1. **User Action (Step 2):**   Fill out the feedback form and click "Submit."  **System Response:**  Saves the feedback, assigns a unique ID, and notifies the appropriate vendor or admin.   1. **User Action (Step 3):**   View feedback status or reply to responses in the "My Feedback" section.  **System Response:**  Updates the feedback status and displays responses from vendors or administrators. |
| **Alternate Flow (Invalid Inputs):**  **User Action (Invalid Input):**  Attempt to submit incomplete feedback (e.g., without a comment).  **System Response:**  Prompts the user to complete all required fields before submission. |
| **Exception Flow (System Errors):**  **User Action (Invalid Input):**  Feedback submission fails due to a server error.  **System Response:**  Displays an error message: "Unable to submit feedback. Please try again later." |
| **Post-Conditions:**   * Feedback is successfully saved in the database. * Admins are notified of new feedback submissions for review. |

**Table 7. Feedback System Stimulus/ Response sequence**

* + 1. Functional Requirements:

|  |  |  |  |
| --- | --- | --- | --- |
| **Req.ID** | **Description** | **Priority** | **Status** |
| Req 1 | Users can submit feedback forms. | High | Pending |
| Req 2 | Feedback is stored securely. | High | Pending |
| Req 3 | Admins can view user feedback. | High | Pending |
| Req 4 | Users can view their feedback. | Medium | Pending |
| Req 5 | Feedback supports star ratings. | Medium | Pending |

**Table 8. Feedback System Functional Requirements**

##### Order Management:

* + 1. Description:

This allows users to sign up for an account on the HYDRO platform and manage their orders efficiently.

* + 1. Stimulus/Response Sequences:

|  |
| --- |
| **Use Case Name:** Order Management |
| **Description:** Users can register for an account on HYDRO and place/manage water tanker orders seamlessly. |
| **Actors:** User, System, Vendor, Admin |
| **Goal:** To enable users to create accounts and place/manage orders for water delivery. |
| **Pre-Conditions:**   * The user must have access to a smartphone or desktop with internet connectivity. * The HYDRO application or website must be operational. |
| **Basic Course of Events / Main Flow**   1. **User Action (Step 1):**   The user opens the HYDRO application or website and clicks on the 'Sign Up' button.  **System Response:**  The system displays a registration form for the user to input their details.   1. **User Action (Step 2):**   The user enters their name, email, phone number, and password, then submits the form.  **System Response:**  The system validates the input, creates the user account, and sends a confirmation email/SMS.   1. **User Action (Step 3):**   The user confirms their registration by clicking on the verification link in the email/SMS.  **System Response:**  The system activates the user account and displays a confirmation message, allowing the user to log in. |
| **Alternate Flow (Invalid Inputs):**  **User Action (Invalid Input):**  The user enters incorrect or incomplete information in the registration form.  **System Response:**  The system displays error messages highlighting the fields that need correction. |
| **Exception Flow (System Errors):**  **System Failure:**  The system encounters an error while creating the account.  **System Response:**  Displays an error message, apologizes for the inconvenience, and prompts the user to retry later. |
| **Post-Condition:**   * The user account is created and verified. * The user can log in and access the platform's features. |

**Table 9. Order Management Stimulus/ Response sequence**

* + 1. Functional Requirements:

|  |  |  |  |
| --- | --- | --- | --- |
| **Req.ID** | **Description** | **Priority** | **Status** |
| Req 1 | User can create an account | High | Pending |
| Req 2 | Validate user input for signup | High | Pending |
| Req 3 | Send confirmation email/SMS | Medium | Pending |
| Req 4 | Verify account via email/SMS | High | Pending |
| Req 5 | Allow user login post-verification | Medium | Pending |
| Req 6 | Handle invalid input gracefully | Low | Pending |
| Req 7 | Notify user of system errors | High | Pending |

**Table 10. Order Management Functional Requirements**

1.

##### Water Quality Reports:

* + 1. Description:

The Water Quality Reports feature provides users with detailed information about the quality of water, including pH levels, contamination tests, and source details, ensuring transparency and reliability.

* + 1. Stimulus/Response Sequences:

|  |
| --- |
| **Use Case Name:** Water Quality Reports |
| **Description:** Users can sign up for an account on “HYDRO” to access water quality reports provided by vendors. |
| **Actors:** End User / Vendor / System |
| **Goal:** To provide users with verified water quality information for informed decision-making. |
| **Pre-Conditions:**   * User must have access to the HydroHitch platform. * Vendors must submit verified water quality data. |
| **Basic Course of Events / Main Flow**   1. **User Action (Step 1):**   User logs into their account and navigates to the water quality reports section.  **System Response:**  The system displays a list of available water quality reports based on the user’s location and selected vendors.   1. **User Action (Step 2):**   User selects a specific vendor to view the detailed water quality report.  **System Response:**  The system retrieves and displays the selected vendor’s water quality report, including pH levels, contamination data, and source information.   1. **User Action (Step 3):**   User saves or shares the water quality report.  **System Response:**  The system confirms the action and provides sharing or downloading options. |

|  |
| --- |
| **Alternate Flow (Invalid Inputs):**  **User Action (Invalid Input):**  The user searches for vendors outside their immediate location.  **System Response:**  The system expands the search radius and displays relevant vendors with water quality reports. |
| **Exception Flow (System Errors):**  **System Failure:**  The system cannot retrieve water quality reports due to a server error.  **System Response:**  Displays an error message: “Unable to retrieve water quality reports. Please try again later.” |
| **Post-Condition:**   * Users access reliable water quality information. Vendors maintain up-to-date and accurate quality data. |

**Table 11. Water Quality Reports Stimulus/ Response sequence**

* + 1. Functional Requirements:

|  |  |  |  |
| --- | --- | --- | --- |
| **Req.ID** | **Description** | **Priority** | **Status** |
| Req 1 | Allow vendors to upload reports | High | Pending |
| Req 2 | Allow vendors to upload reports | High | Pending |
| Req 3 | Provide detailed contamination data | High | Pending |
| Req 4 | Include pH levels in reports | Medium | Pending |
| Req 5 | Offer report download functionality | Medium | Pending |
| Req 6 | Display vendor-specific reports | Low | Pending |
| Req 7 | Ensure data integrity and security | High | Pending |

**Table 12. Water Quality Reports Functional Requirement**

##### Order Tracking:

* + 1. Description:

Enables users to monitor the real-time status and progress of their water delivery orders through the HydroHitch platform.

* + 1. Stimulus/Response Sequences:

|  |
| --- |
| **Use Case Name:** Order Tracking |
| **Description:** Users can sign up for an account on “HYDRO” to access and track the status of their water delivery orders in real-time. |
| **Actors:** End User / Vendor / System |
| **Goal:** Enable users to log in, access their service history, and submit ratings and reviews for services. |

|  |
| --- |
| **Pre-Conditions:**   * User must have an active account on HydroHitch. * A water delivery order must be placed. |
| **Basic Course of Events / Main Flow**   1. **User Action (Step 1):**   User logs into their HydroHitch account.  **System Response:**  Displays the dashboard with a list of active and past orders.   1. **User Action (Step 2):**   User selects an active order to view details.  **System Response:**  Displays the real-time status of the order, including location, estimated delivery time, and vendor contact information.   1. **User Action (Step 3):**   User opts to receive live updates or notifications.  **System Response:**  Sends push notifications for order updates, such as dispatch, in transit, and delivery. |
| **Alternate Flow (Invalid Inputs):**  **User Action (Invalid Input):**  User selects an archived order for reference.  **System Response:**  Displays the past order details, including delivery history and quality reports. |
| **Exception Flow (System Errors):**  **System Failure:**  Backend server is temporarily down.  **System Response:**  Displays an error message: "We are experiencing technical difficulties. Please try again later." Ensures that data is preserved for future updates. |
| **Post-Condition:**   * User receives accurate and timely updates on order status. * System logs all tracking interactions for analytics and future reference. |

**Table 13. Order Tracking Stimulus/ Response sequence**

* + 1. Functional Requirements:

|  |  |  |  |
| --- | --- | --- | --- |
| **Req.ID** | **Description** | **Priority** | **Status** |
| Req 1 | Enable users to track active orders. | High | Pending |
| Req 2 | Provide real-time updates via notifications. | High | Pending |
| Req 3 | Allow access to archived order history. | High | Pending |
| Req 4 | Display estimated delivery times. | Medium | Pending |
| Req 5 | Show vendor contact details. | Medium | Pending |
| Req 6 | Handle offline access gracefully. | Low | Pending |
| Req 7 | Log interactions for future reference. | High | Pending |

**Table 14. Order Tracking Functional Requirements**

##### Delivery Charge Calculation:

* + 1. Description:

The Delivery Charge Calculation feature allows users to estimate delivery costs based on tanker size, distance, and vendor pricing. This ensures transparency and helps users make informed decisions before placing orders.

* + 1. Stimulus/Response Sequences:

|  |
| --- |
| **Use Case Name:** Delivery Charge Calculation |
| **Description:** User can calculate delivery charges before placing an order. |
| **Actors:** User, System |
| **Goal:** To provide an accurate delivery cost based on input parameters. |
| **Pre-Conditions:**   * User must be logged into their HydroHitch account. * User has selected a water tanker and entered their location details. |
| **Basic Course of Events / Main Flow**   1. **User Action (Step 1):**   The user sets User selects a water tanker from the available options.  **System Response:**  The system System displays the selected tanker details and prompts for the delivery address.   1. **User Action (Step 2):**   User enters the delivery address and confirms the location.  **System Response:**  System calculates the distance from the vendor's location to the user's address. |
| **Alternate Flow (Invalid Inputs):**  **User Action (Invalid Input):**  User enters an incorrect address or missing information.  **System Response:**  The system prompts the user to correct the address or provide missing information. |
| **Exception Flow (System Errors):**  **User Action (Invalid Input):**  The system is unable to calculate delivery charges due to a server error.  **System Response:**  Displays an error message: "Unable to calculate delivery charges at the moment. Please try again later." |
| **Post-Condition:**   * Delivery charges are displayed to the user. * The order summary is updated with the calculated charges.   The system ensures data accuracy and logs the transaction for vendor review. |

**Table 15. Delivery Charge Calculation Stimulus/ Response sequence**

* + 1. Functional Requirements:

|  |  |  |  |
| --- | --- | --- | --- |
| **Req.ID** | **Description** | **Priority** | **Status** |
| Req 1 | User sign-up functionality | High | Pending |
| Req 2 | Delivery charge calculation logic | High | Pending |
| Req 3 | Location-based calculation | High | Pending |
| Req 4 | Error handling for invalid inputs | Medium | Pending |
| Req 5 | User account management | Medium | Pending |
| Req 6 | User feedback on system errors | Low | Pending |
| Req 7 | Support for multiple addresses | High | Pending |

**Table 16. Delivery Charge Calculation Functional Requirements**

##### Location-Based Vendor Access:

* + 1. Description:

This feature allows users to sign up for an account on "HYDRO" and grants vendors access based on the user's location, ensuring relevant vendors are displayed.

* + 1. Stimulus/Response Sequences:

|  |
| --- |
| **Use Case Name:** Location-Based Vendor Access |
| **Description:** User can sign up for an account on "HYDRO" and gain access to location-based vendors. |
| **Actors:** User, Vendor, System |
| **Goal:** Allow users to view and interact with vendors based on their location. |
| **Pre-Conditions:**   * User has access to the "HYDRO" platform. * User has enabled location services. |
| **Basic Course of Events / Main Flow**   1. **User Action (Step 1):**   User visits the "HYDRO" platform and selects "Sign Up."  **System Response:**  The system displays a sign-up form for account creation.   1. **User Action (Step 2):**   User enters necessary information (e.g., email, password, location).  **System Response:**  The system processes the user’s information and verifies location.   1. **User Action (Step 3):**   User submits the form to create an account.  **System Response:**  The system creates the account and grants access to location-based vendor listings. |

|  |
| --- |
| **Alternate Flow (Invalid Inputs):**  **User Action (Invalid Input):**  User opts to skip location permission during sign-up.  **System Response:**  System displays a prompt to enable location access for vendor recommendations. |
| **Exception Flow (System Errors):**  **System Failure:**  User cannot sign up due to location service failure.  **System Response:**  System displays an error message asking the user to enable location services. |
| **Post-Condition:**   * User account is created and linked to a specific location. * User has access to the relevant vendors based on their location. |

**Table 17. Location-Based Vendor Access Stimulus/ Response sequence**

* + 1. Functional Requirements:

|  |  |  |  |
| --- | --- | --- | --- |
| **Req.ID** | **Description** | **Priority** | **Status** |
| Req 1 | User Sign-Up | High | Pending |
| Req 2 | Location Services | High | Pending |
| Req 3 | Vendor Access | High | Pending |
| Req 4 | Account Creation | Medium | Pending |
| Req 5 | Location Permission | Medium | Pending |
| Req 6 | Vendor Filtering | Low | Pending |
| Req 7 | Error Handling | High | Pending |

**Table 18. Location-Based Vendor Access Functional Requirements**

##### Emergency Water Service:

* + 1. Description:

Users can sign up for an account on "HYDRO" to request emergency water services.

* + 1. Stimulus/Response Sequences:

|  |
| --- |
| **Use Case Name:** Emergency Water Service |
| **Description:** Users can sign up for an account on "HYDRO" to request emergency water services. |
| **Actors:** User, Vendor, System |
| **Goal:** Enable users to quickly access water delivery during emergencies. |
| **Pre-Conditions:**   * User has internet access. * "HYDRO" platform is functional and available. |

|  |
| --- |
| **Basic Course of Events / Main Flow**   1. **User Action (Step 1):**   User navigates to the sign-up page.  **System Response:**  Displays the registration form.   1. **User Action (Step 2):**   User fills out required details (name, email, address, etc.) and submits the form.  **System Response:**  Validates inputs and creates the user account.   1. **User Action (Step 3):**   User receives confirmation and logs into their account.  **System Response:**  Grants access to emergency water service request options. |
| **Alternate Flow (Invalid Inputs):**  **User Action (Invalid Input):**  User enters invalid details.  **System Response:**  Displays an error message with prompts to correct inputs. |
| **Exception Flow (System Errors):**  **System Failure:**  Server downtime or network issues.  **System Response:**  Displays an error message and suggests trying again later. |
| **Post-Condition:**   * User account is successfully created. * User can access and request emergency water services through the platform. |

**Table 19. Emergency Water Service Stimulus/ Response sequence**

* + 1. Functional Requirements:

|  |  |  |  |
| --- | --- | --- | --- |
| **Req.ID** | **Description** | **Priority** | **Status** |
| Req 1 | User can sign up for an account. | High | Pending |
| Req 2 | Validate user inputs. | High | Pending |
| Req 3 | Display error messages for invalid inputs. | High | Pending |
| Req 4 | User receives confirmation upon successful registration. | Medium | Pending |
| Req 5 | Enable account login. | Medium | Pending |
| Req 6 | Handle server or network errors gracefully. | Low | Pending |
| Req 7 | Ensure data security during sign-up. | High | Pending |

**Table 20. Emergency Water Service Functional Requirements**

##### Respond to User Queries:

* + 1. Description:

This feature allows users to interact with the "HYDRO" platform by signing up and receiving system assistance or information. The system ensures smooth communication, enhancing user satisfaction.

* + 1. Stimulus/Response Sequences:

|  |
| --- |
| **Use Case Name:** Respond to User Queries |
| **Description:** Users can register for an account on "HYDRO" to gain access to its query-response functionality. |
| **Actors:** User / Super Admin |
| **Goal:** Enable users to register and interact with the "HYDRO" system to receive timely and accurate responses to their queries. |
| **Pre-Conditions:**   * The "HYDRO" platform is operational and accessible. * The user has internet connectivity and a compatible device. |
| **Basic Course of Events / Main Flow**   1. **User Action (Step 1):**   The user accesses the "HYDRO" platform and initiates the signup process by entering their email address and creating a password.  **System Response:**  The system validates the entered information and displays a confirmation message prompting the user to verify their email.   1. **User Action (Step 2):**   The user clicks the verification link sent to their email.  **System Response:**  The system confirms email verification and activates the user account, providing access to the query interface.   1. **User Action (Step 3):**   The user submits a query via the "HYDRO" interface.  **System Response:**  The system processes the query, retrieves relevant information, and presents the response to the user. |
| **Alternate Flow (Invalid Inputs):**  **User Action (Invalid Input):**  The user enters incorrect or incomplete signup information.  **System Response:**  The system highlights the invalid fields and prompts the user to correct the information before proceeding. |

|  |
| --- |
| **Exception Flow (System Errors):**  **User Action (Invalid Input):**  If the system experiences downtime or connectivity issues, users may not access the signup page or submit queries.  **System Response:**  The system displays an error message: “Service temporarily unavailable. Please try again later.” |
| **Post-Condition:**   * The user successfully registers on the platform and submits queries. * The system securely stores user data and provides accurate responses. |

**Table 21. Respond to User Queries Stimulus/ Response sequence**

* + 1. Functional Requirements:

|  |  |  |  |
| --- | --- | --- | --- |
| **Req.ID** | **Description** | **Priority** | **Status** |
| Req 1 | Implement user signup feature | High | Pending |
| Req 2 | Add email verification process | High | Pending |
| Req 3 | Enable query submission | High | Pending |
| Req 4 | Develop error handling | Medium | Pending |
| Req 5 | Enhance user interface design | Medium | Pending |

**Table 22. Respond to User Queries Functional Requirements**

#### Other Non-functional Requirements

##### Performance Requirements:

HydroHitch must support up to 10,000 concurrent users without significant performance degradation. The system should process user requests, such as order placement or delivery charge calculation, within 3 seconds on average. Database queries for vendor or order data should execute in under 2 seconds. The application should maintain 99.9% uptime to ensure availability. All modules must efficiently handle increased loads during peak usage periods.

##### Safety Requirements:

The application must validate all input data to prevent incorrect calculations or unintended actions. Users should be warned when providing incomplete or inconsistent delivery information. The system should ensure secure water quality data uploads to prevent tampering. Emergency requests must prioritize user safety by ensuring immediate notification to vendors. Regular audits should identify and mitigate potential hazards in system operations.

##### Security Requirements:

HydroHitch must encrypt all sensitive user data, including login credentials and payment details, using industry-standard encryption protocols. Role-based access control (RBAC) should restrict access to sensitive functionalities like quality report uploads and user data management. The system must include safeguards against common vulnerabilities such as SQL injection and cross-site scripting (XSS). User

sessions should auto-expire after 15 minutes of inactivity to prevent unauthorized access. A robust logging mechanism should record all critical operations for security audits.

##### Software Quality Attributes:

The application must be reliable, ensuring that all functionalities perform as intended under defined conditions. It should maintain high usability with an intuitive interface for users, vendors, and administrators. Scalability is crucial, allowing the platform to accommodate a growing user base without impacting performance. Maintainability should be prioritized, with clear documentation and modular code for easy updates. Interoperability should enable seamless integration with third-party tools like Google Maps and payment gateways.

##### Business Rules:

1. Vendors must provide accurate water quality reports and update them regularly.
2. Delivery charges must be calculated based on distance, tanker size, and vendor-specific pricing.
3. Users should be allowed to modify or cancel orders up to 15 minutes before scheduled delivery.
4. Super admins have the authority to review and resolve disputes between users and vendors.
5. Emergency service requests must be addressed within the shortest time frame, with vendor prioritization.

#### Other Requirements

Plan for ongoing updates and improvements to keep our platform competitive and aligned with evolving industry trends.

##### Appendix A: Glossary

1. GPS (Global Positioning System)
2. API (Application Programming Interface)
3. SRS (Software Requirements Specification)
4. SDS (Software Design Specification)
5. STS (Software Test Specification)
6. UM (User Manu

**SOFTWARE DESIGN SPECIFICATION**

#### Introduction

##### Purpose of this document:

The purpose of this document is to provide a comprehensive overview of the HydroHitch project, detailing its objectives, features, and technical requirements. It serves as a blueprint for developers, designers, and stakeholders, ensuring clarity on project expectations and deliverables. The document outlines the functional and non-functional requirements, system architecture, and implementation strategies. It aims to standardize the development process, facilitate communication, and streamline decision-making. By capturing all relevant details, this document ensures alignment among team members and stakeholders. Additionally, it provides a reference point for evaluating the project's progress and success.

##### Scope of the development project:

The scope of the HydroHitch project encompasses the design and development of a multivendor water delivery platform that connects users, vendors, and administrators. It includes creating user-friendly interfaces for ordering water, viewing quality reports, and tracking deliveries. Vendors will manage tanker listings, orders, and water quality data, while administrators oversee the platform's operations. The system must handle high user traffic, integrate with mapping tools for location-based services, and ensure secure transactions. The project also includes testing, deployment, and maintenance to ensure reliability and scalability. Ultimately, the project aims to address challenges in water delivery with a focus on transparency, convenience, and quality assurance.

##### Definition, acronyms, and abbreviations:

Terms and Description

ERD Entity Relationship

SDS Software Design Specification

SRS Software Requirements Specification

##### References:

* + 1. https:/[/www.mongodb.com/](http://www.mongodb.com/)
    2. https://expressjs.com/
    3. https://reactjs.org/
    4. https://nodejs.org/
    5. https:/[/www.w3schools.com/](http://www.w3schools.com/)
    6. https://developer.mozilla.org/
    7. https:/[/www.codecademy.com/](http://www.codecademy.com/)
    8. https://tailwindcss.com/

##### Overview of Document:

This document serves as a detailed guide to the HydroHitch development project, covering its purpose, scope, and features. It begins with an introduction to the project, followed by a breakdown of its objectives, technical requirements, and development methodology. Key sections include performance metrics, safety and security considerations, and a detailed description of user, vendor, and admin functionalities. The document also outlines use cases, diagrams, and system architecture to provide a visual representation of the project. Additionally, it specifies timelines, milestones, and resource

requirements to ensure structured development. This document is essential for aligning all stakeholders and ensuring successful project execution.

#### System Architecture Description

##### Section overview:

This section provides a detailed description of the system architecture for HydroHitch, outlining how the application components interact and integrate. It explains the design principles, structural organization, and key technologies used. The focus is on ensuring scalability, reliability, and ease of maintenance. By detailing each architectural component, the section ensures clarity for developers and stakeholders. Visual representations, such as diagrams, are included for a better understanding of the overall architecture.

##### General Constraints:

The architecture must support up to 10,000 concurrent users with minimal latency. It is constrained by budget limitations, necessitating efficient use of hosting and resources. The system must adhere to security and privacy regulations, such as GDPR, for user data protection. Integration with third-party APIs, like Google Maps and payment gateways, must be seamless. Additionally, it should be deployable on cloud platforms like AWS or Azure for scalability.

##### Data Design:

The data design includes a MongoDB-based schema to manage users, vendors, orders, and quality reports. Relationships between entities are structured to allow efficient querying and real-time updates. Data redundancy is minimized using normalized structures, while critical data like user credentials are encrypted. Data partitioning techniques ensure optimal performance under heavy loads. Backup and recovery mechanisms are also integrated to safeguard against data loss.

##### Program Structure:

The program structure follows a modular design with a clear separation of concerns. The MERN stack is utilized: React handles the frontend, Node.js and Express.js manage the backend, and MongoDB serves as the database. APIs are designed to be RESTful, enabling efficient communication between components. The system includes three modules: user panel, vendor panel, and admin panel, each with distinct responsibilities. The modularity allows for easier debugging, testing, and future feature integration.

##### Alternatives Considered:

Several architectural alternatives were evaluated, including monolithic vs. microservices architectures. A monolithic approach was dismissed due to scalability concerns, while microservices were deemed too complex for the project's current scope. Relational databases like MySQL were considered but replaced by MongoDB for its flexibility with unstructured data. Firebase was explored for real-time capabilities but was rejected due to limited scalability for this application. The chosen architecture balances simplicity, scalability, and cost-effectiveness.

#### Detailed description of components

##### Section overview:

This section provides an in-depth analysis of the core modules within the **HYDRO** app. It outlines the functionality and interaction of key components, including the user interface, service booking system, payment integration, and real-time updates. Each component is described in terms of its role, behavior, and how it communicates with other parts of the system. This section ensures a clear understanding of the app’s architecture and functionality.

##### Registration:

|  |  |
| --- | --- |
| **Identification** | Register |
| **Type** | Form, method |
| **Purpose** | To enable new users to create an account, granting them access to the HYDRO platform and its services. |
| **Function** | Handles user input, validates registration information, and securely stores data required for account creation. |
| **Dependencies** | Requires integration with a MongoDB database for storing user data, email validation services for verification, and secure password handling protocols (such as encryption). |
| **Interfaces** | Interacts with the user registration forms in the app’s UI, backend APIs for data submission, and email services for verification purposes. |
| **Resources** | Utilizes database storage for user credentials, encryption libraries for password security, and user authentication modules for future login. |
| **Processing** | Validates the user's email, password, and any other registration inputs, checks for existing accounts to avoid duplicates, and securely encrypts  sensitive data like passwords. |
| **Data** | Stores user credentials such as email and password, optional details like phone number and address, and records the verification status of the account. |

##### Login:

|  |  |
| --- | --- |
| **Identification** | Login |
| **Type** | Form, method |
| **Purpose** | To allow existing users to securely access their accounts and the platform's features. |
| **Function** | Handles user input (email/phone number and password), validates credentials, and authorizes access. |
| **Dependencies** | Requires a database (MongoDB), password encryption/decryption libraries, and authentication services. |
| **Interfaces** | Interacts with the user login form, backend API for credential verification, and security systems for failed attempts. |
| **Resources** | Database storage, encryption libraries, authentication tokens (JWT or similar), and logging mechanisms. |
| **Processing** | Verifies user credentials against stored data, checks for account status (active/locked), and generates access tokens for authenticated sessions. |
| **Data** | Utilizes user credentials (email/phone number, password), login attempts, and session tokens for tracking authenticated sessions. |

##### Ordering Tankers:

|  |  |
| --- | --- |
| **Identification** | To order the tankers |
| **Type** | Form, method |
| **Purpose** | Enable users to order water tankers conveniently through the platform. |
| **Function** | Facilitate selection, scheduling, and confirmation of tanker orders based on user location and preferences. |
| **Dependencies** | Requires accurate user location, vendor availability, and pricing details. |
| **Interfaces** | Interacts with the user panel, vendor panel, and APIs. |
| **Resources** | Utilizes database storage for orders, real-time API for tracking, and Google Maps for location data. |
| **Processing** | Calculates delivery charges, validates order details, and updates vendor availability in real time. |
| **Data** | Includes user profiles, vendor data, order details, and payment transactions. |

##### Order management:

|  |  |
| --- | --- |
| **Identification** | Management of orders |
| **Type** | Form, method |
| **Purpose** | Facilitate seamless placement, tracking, and management of water tanker orders. |
| **Function** | Enable users to create, view, modify, and cancel orders while vendors manage delivery statuses. |
| **Dependencies** | Relies on user profiles, vendor availability, and delivery charge calculation modules. |
| **Interfaces** | Accessible through user, vendor, and admin panels, with API integration for order tracking. |
| **Resources** | Requires database storage for order details, server-side processing, and front-end rendering. |
| **Processing** | Handles order validation, assignment to vendors, and real-time updates on delivery status. |
| **Data** | Stores order IDs, user details, vendor information, tanker specifications, and delivery timelines. |

##### Feedback:

|  |  |
| --- | --- |
| **Identification** | Ratings/Review |
| **Type** | Form, method |
| **Purpose** | To allow users to provide feedback on their experience with the service or product. |
| **Function** | Collect, store, and display user feedback for analysis and improvement. |
| **Dependencies** | Requires a database for storing feedback and APIs for submission and retrieval. |
| **Interfaces** | Accessible via user, vendor, and admin panels on the application. |
| **Resources** | Feedback form, database storage, and backend processing modules. |
| **Processing** | Validates, stores, and organizes feedback for display and reporting. |
| **Data** | Includes user name, feedback content, timestamps, and response status. |

##### Query System:

|  |  |
| --- | --- |
| **Identification** | Query System |
| **Type** | Form, method |
| **Purpose** | To allow users to submit queries and receive responses from vendors or admins. |
| **Function** | Facilitates communication between users, vendors, and admins for issue resolution and service inquiries. |
| **Dependencies** | User and vendor profiles, query database, notification system. |
| **Interfaces** | User interface (query submission form), vendor/admin interface (query response panel). |
| **Resources** | Database for storing queries, backend server for processing requests, notification service. |
| **Processing** | User submits a query; backend stores it, notifies the vendor/admin; vendor/admin responds, and the system updates the user interface. |
| **Data** | Query content, user ID, vendor ID, timestamps, and response content. |

##### Water Quality Reports:

|  |  |
| --- | --- |
| **Identification** | Reports |
| **Type** | Form, method |
| **Purpose** | To provide users and vendors with accurate and detailed information about water quality, including pH levels and contamination status. |
| **Function** | Allows vendors to submit, update, and display water quality reports to users for transparency and decision-making. |
| **Dependencies** | Relies on vendor input, real-time data collection, and secure database management for accurate reporting. |
| **Interfaces** | Accessible through the user and vendor panels, where reports can be viewed and updated. |
| **Resources** | Utilizes the MongoDB database for storing report data, and external APIs for contamination detection and data verification. |
| **Processing** | Gathers and processes water quality data, calculates key metrics like pH levels, and generates user-friendly reports. |
| **Data** | Includes data on pH levels, contamination tests, water source, vendor information, and delivery details. |

##### Emergency Orders:

|  |  |
| --- | --- |
| **Identification** | Emergency Orders |
| **Type** | Form, method |
| **Purpose** | To provide users with a quick way to request urgent water deliveries. |
| **Function** | Allows users to place emergency water orders for immediate delivery. |
| **Dependencies** | Relies on user authentication, location tracking, and vendor availability. |
| **Interfaces** | User interface for placing emergency orders, vendor interface for managing requests. |
| **Resources** | Requires real-time location data, vendor availability data, and delivery routing. |
| **Processing** | Processes user location, calculates delivery charges, and triggers urgent order dispatch. |
| **Data** | User profile information, order details, delivery time, and vendor responses. |

##### Delivery Charge Calculation:

|  |  |
| --- | --- |
| **Identification** | Delivery Charge Calculator |
| **Type** | Form, method |
| **Purpose** | Calculate the cost of water delivery based on factors such as distance, volume, and vendor pricing. |
| **Function** | Dynamically compute delivery charges using predefined rates and location- based inputs. |
| **Dependencies** | Relies on Google Maps API for distance calculation and vendor pricing data from the database. |
| **Interfaces** | User app interface for input and display, and APIs for backend charge computation. |
| **Resources** | Vendor pricing details, location data, and delivery distance metrics. |
| **Processing** | Retrieves location and volume data, applies pricing algorithms, and calculates total cost. |
| **Data** | Includes user location, delivery distance, water volume, and vendor-specific pricing rates. |

#### User interface design

##### Section overview:

The user interface design focuses on creating an intuitive, user-friendly experience for HydroHitch users. It incorporates responsive layouts, ensuring accessibility across devices. Each user type—customer, vendor, and admin—has a dedicated panel tailored to their specific tasks. Consistency is achieved through unified design elements like color schemes and typography. Real-time feedback, such as loading indicators and notifications, enhances usability. The interface simplifies complex workflows like order placement, tracking, and vendor management.

##### Interface design rules:

Follow simplicity and minimalism to avoid overwhelming users. Ensure consistency in visual elements like fonts, colors, and button styles across the app. Prioritize accessibility with readable text, contrasting colors, and screen reader support. Use clear labels, tooltips, and icons to guide users through functions. Optimize for mobile-first design, ensuring seamless navigation on smaller screens. Employ real-time feedback for actions like button clicks and form submissions.

##### GUI components:

Interactive buttons for actions like order placement and vendor selection. Dropdown menus for filtering vendors based on location and ratings. Real-time progress bars and maps for order tracking. Form fields for user registration and order details input. Pop-up modals for displaying water quality reports and charge calculations. Navigation menus for quick access to primary features like orders, queries, and account settings.

##### Detailed description:

The user dashboard provides quick access to order history, active orders, and account settings. Vendor panels feature tools for managing tankers, updating quality reports, and viewing user queries. Super admin dashboards display analytics, vendor activity logs, and query escalation tools. Real-time tracking integrates Google Maps to display delivery status. Pop-ups and notifications inform users of updates like successful order placements or delays. The design incorporates icons and animations to improve engagement and clarity.

#### Reuse and relationships to other products:

Reuses modular design components like buttons, forms, and navigation bars to maintain consistency. Interfaces for vendor and admin panels share similar layouts to streamline development. Google Maps API and quality reporting modules can be adapted for related delivery platforms. Follows industry standards, allowing integration with third-party payment or logistics systems. Relationships exist with similar platforms offering delivery services, sharing best practices in design. Promotes scalability by reusing backend APIs for future feature expansions.

#### Design decision and tradeoffs:

Chose React Native for its cross-platform support, balancing development speed and performance. Opted for a minimalist design over feature-heavy layouts to improve usability and loading speeds. Prioritized mobile-first design but ensured desktop compatibility for vendor and admin panels. Decided against advanced animations to reduce complexity and ensure smooth performance on low-end devices. Implemented modular components for reuse but required additional time for initial development. Focused on essential features for launch, deferring advanced functionality to future updates.

#### Pseudo code for components:

##### Registration:

// Step 1: Collect User Input

Display "Enter your email/phone number" Input email/phone

Display "Enter your password" Input password

Display "Confirm your password" Input confirmPassword

Display "Enter your full name" Input fullName

Display "Enter your address (optional)" Input address

// Step 2: Validate Inputs

IF email/phone is invalid THEN

Display "Invalid email/phone number. Please try again." END Registration

IF password != confirmPassword THEN

Display "Passwords do not match. Please try again." END Registration

IF password is too weak THEN

Display "Password is too weak. Please choose a stronger password." END Registration

// Step 3: Check if Email/Phone already exists IF email/phone already exists in database THEN

Display "Email/phone number is already registered. Please log in." END Registration

// Step 4: Store User Data Encrypt password

Store email/phone, password, fullName, and address in the database

// Step 5: Send Verification Email/Message Send email/phone verification link

Display "Registration successful! Please check your email/phone for verification." END Registration

##### Login:

// Prompt user for login credentials DISPLAY "Enter Email/Phone Number:" INPUT email\_or\_phone

DISPLAY "Enter Password:" INPUT password

// Validate the entered credentials

IF email\_or\_phone exists in database THEN

IF password matches the stored password for the email\_or\_phone THEN DISPLAY "Login Successful"

Redirect to user dashboard ELSE

DISPLAY "Incorrect password" END IF

ELSE

DISPLAY "User not found" END IF

END

##### Ordering the tanker:

// Prompt user for tanker order details

DISPLAY "Select Tanker Size (e.g., Small, Medium, Large):" INPUT tanker\_size

DISPLAY "Enter Delivery Address:"

INPUT delivery\_address DISPLAY "Choose Delivery Time:" INPUT delivery\_time

// Calculate delivery charges

CALCULATE delivery\_charges based on tanker\_size and delivery\_address DISPLAY "Total Charges: " + delivery\_charges

DISPLAY "Confirm Order? (Yes/No):"

INPUT confirmation IF confirmation is "Yes" THEN SAVE order details to database

DISPLAY "Order Placed Successfully" Redirect to order tracking page

ELSE

DISPLAY "Order Cancelled" END IF

##### Order management:

// Prompt vendor for action

DISPLAY "Choose an action: [1] View Orders, [2] Update Order Status" INPUT action

IF action = 1 THEN

// Display all orders for the vendor

FETCH orders WHERE vendor\_id = current\_vendor\_id DISPLAY orders

ELSE IF action = 2 THEN

// Update the status of a specific order DISPLAY "Enter Order ID to update:" INPUT order\_id

DISPLAY "Enter new status (e.g., Pending, In Progress, Delivered):" INPUT new\_status

IF order\_id exists in vendor's orders THEN UPDATE order status TO new\_status DISPLAY "Order status updated successfully" ELSE

DISPLAY "Order ID not found" END IF

ELSE

DISPLAY "Invalid action selected" END IF

##### Feedback:

// Prompt user for feedback DISPLAY "Enter your feedback:" INPUT feedback

// Validate feedback input

IF feedback is not empty THEN STORE feedback in the database

DISPLAY "Thank you for your feedback!" ELSE

DISPLAY "Feedback cannot be empty" END IF

DISPLAY "Thank you for your feedback!"

##### Query system:

// Prompt user for query input DISPLAY "Enter your query:" INPUT user\_query

// Validate user session

IF user is logged in THEN

// Store the query in the database

SAVE user\_query in database with user\_id and timestamp

// Notify admin or vendor

SEND notification to relevant admin/vendor

DISPLAY "Your query has been submitted successfully" ELSE

DISPLAY "Please log in to submit a query" END IF

##### Emergency Request:

// Prompt user for emergency request details DISPLAY "Enter Location:"

INPUT location

DISPLAY "Enter Required Water Volume:" INPUT volume

// Validate inputs

IF location IS NOT EMPTY AND volume IS NOT EMPTY THEN

// Process the emergency request SAVE request details to database

DISPLAY "Emergency request submitted successfully" Notify nearest vendors of the emergency request

ELSE

DISPLAY "Invalid input. Please provide all required details." END IF

##### Cost calculator:

// Prompt user for order details DISPLAY "Enter delivery location:" INPUT location

DISPLAY "Enter water volume (in liters):" INPUT volume

DISPLAY "Select vendor:" INPUT vendor

// Retrieve vendor pricing and location data GET vendor\_price FROM database

GET distance FROM location\_service

// Calculate delivery charges

delivery\_cost = (vendor\_price \* volume) + (distance \* delivery\_rate)

// Display the calculated cost

DISPLAY "Total delivery cost: " + delivery\_cost END

##### Order Tracking:

// Prompt user for order tracking DISPLAY "Enter Order ID:" INPUT order\_id

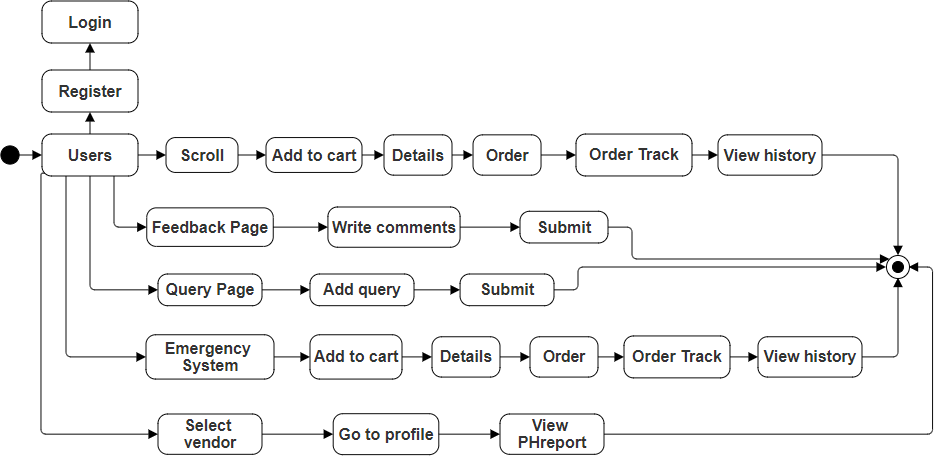
// Retrieve order status from the database IF order\_id exists in database THEN status = GET order status for order\_id DISPLAY "Order Status: " + status ELSE

DISPLAY "Order not found" END IF

#### Appendices:

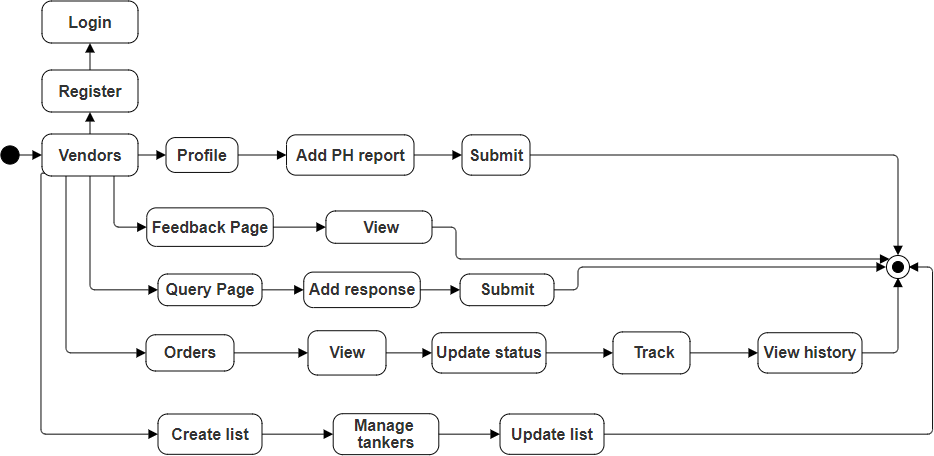
##### State Chart Diagram:

###### User:

****

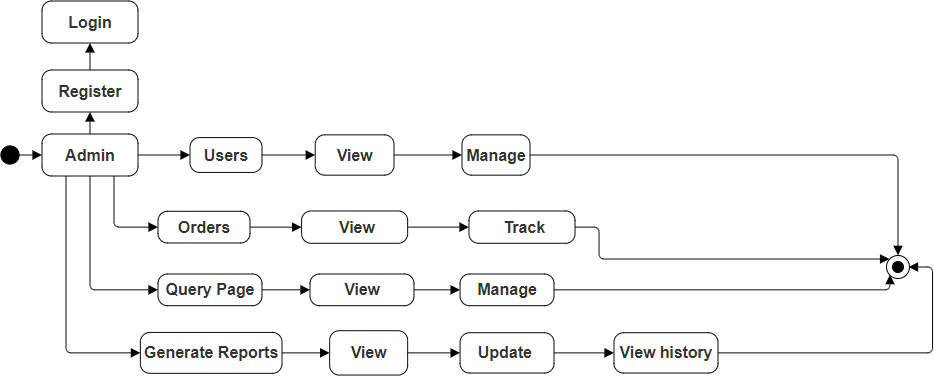
**Fig no: 8.1.1**

###### Vendor:

****

**Fig no: 8.1.2**

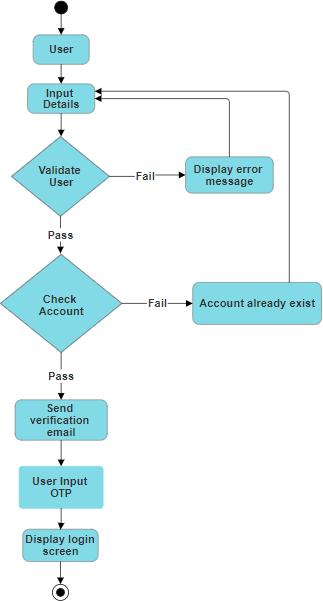
###### Admin:

****

**Fig no: 8.1.3**

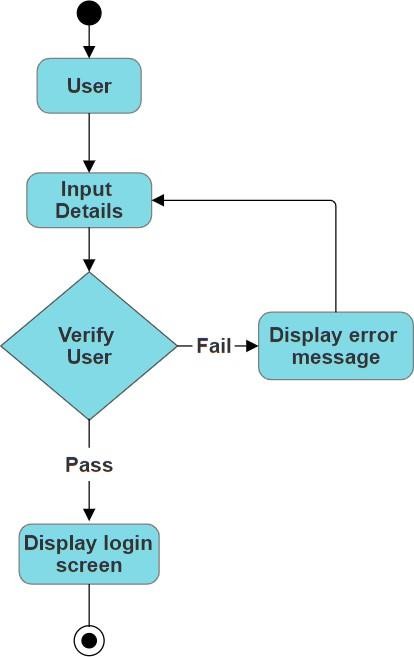
##### Activity Diagram:

###### Registration:

****

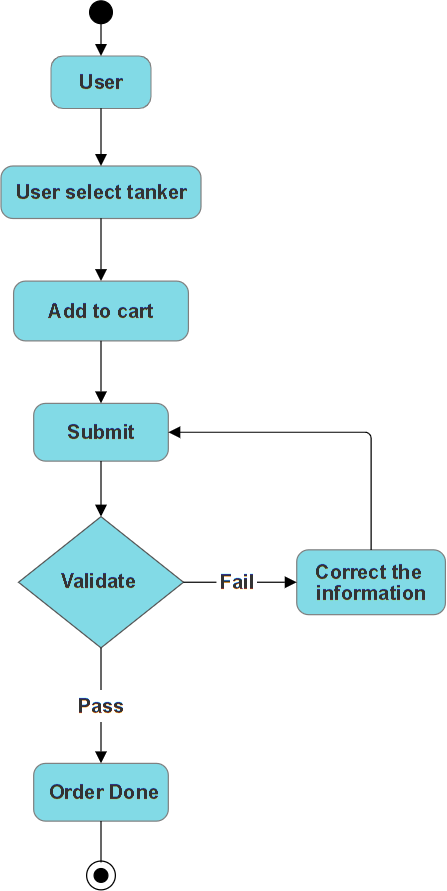
**Fig no: 8.2.1**

###### Login:

****

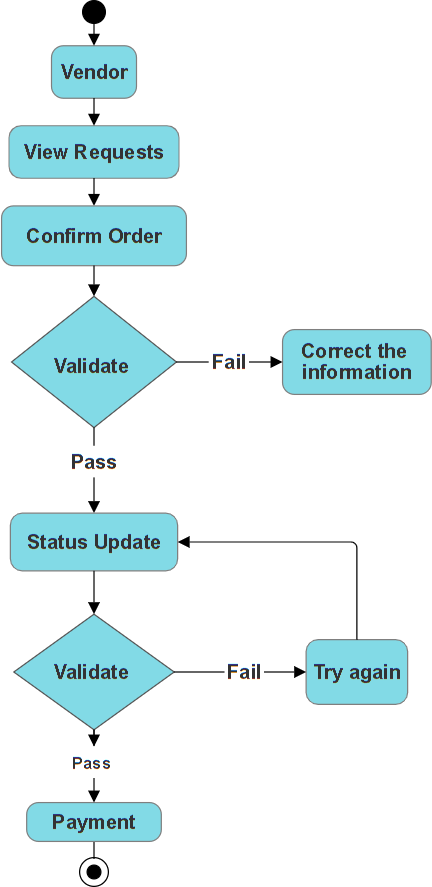
**Fig no: 8.2.2**

###### Tanker Ordering:

****

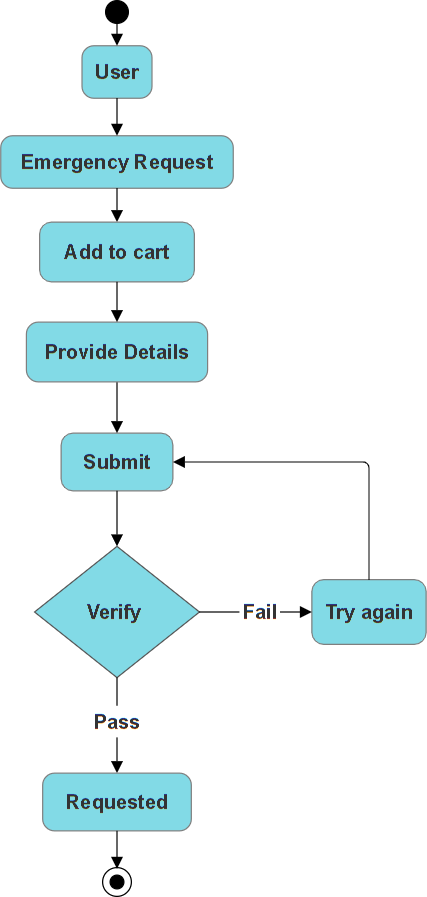
**Fig no: 8.2.3**

###### Order Management:

****

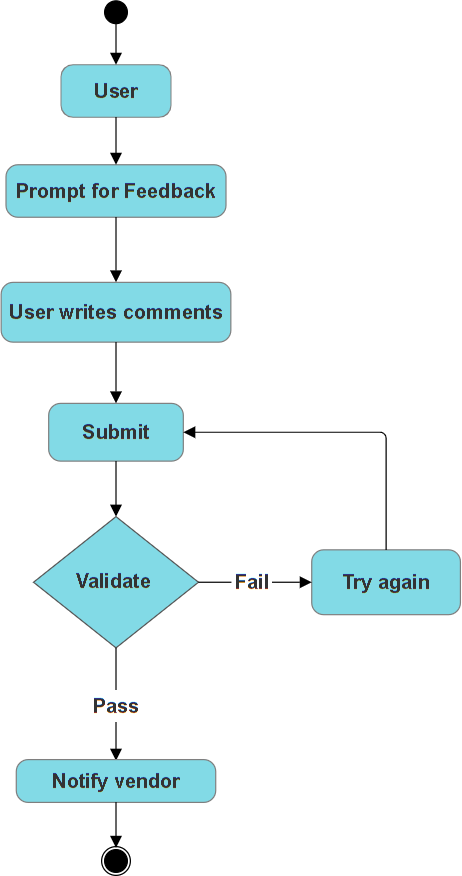
**Fig no: 8.2.4**

###### Emergency System:

****

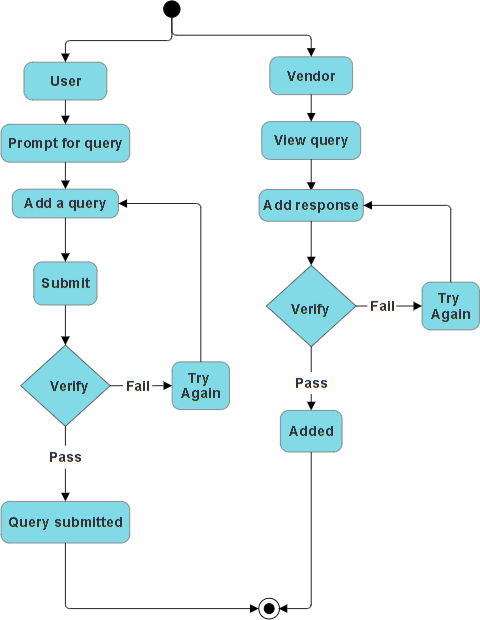
**Fig no: 8.2.5**

###### Feedback:

****

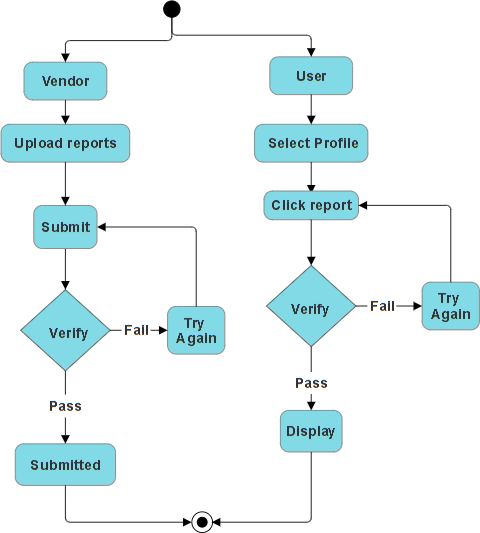
**Fig no: 8.2.6**

###### Query System:

****

**Fig no: 8.2.7**

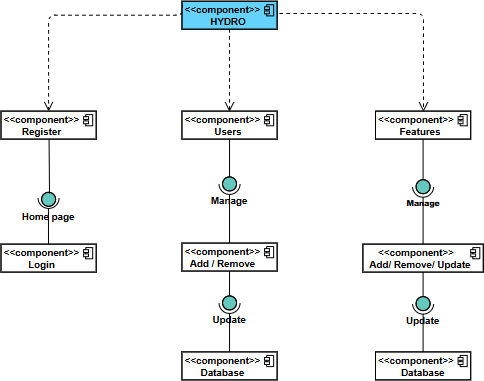
###### Quality Reports:

****

**Fig no: 8.2.8**

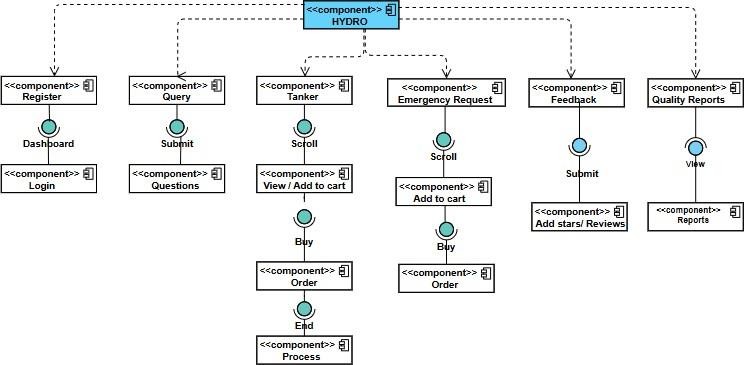
##### Component Diagram:

###### Admin:

****

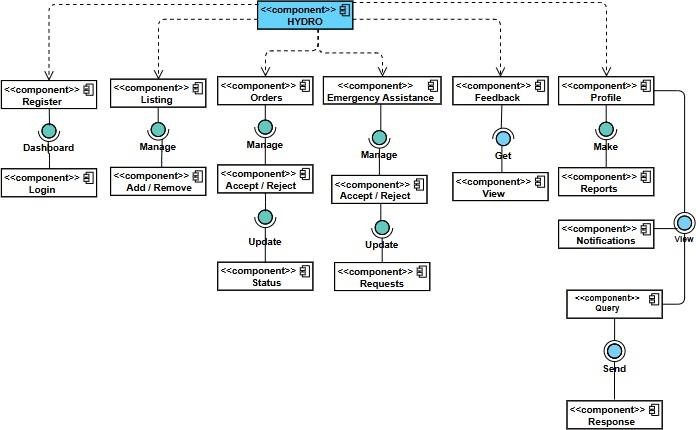
**Fig no: 8.3.1**

###### User:

****

**Fig no: 8.3.2**

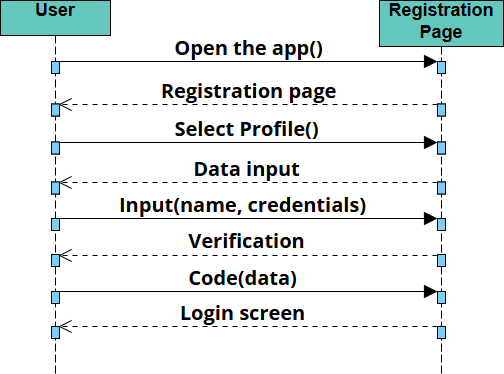
###### Vendor:

****

**Fig no: 8.3.3**

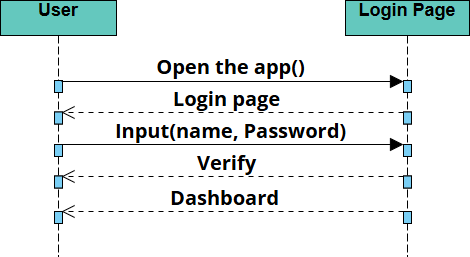
##### Sequence Diagram:

###### Registration:

****

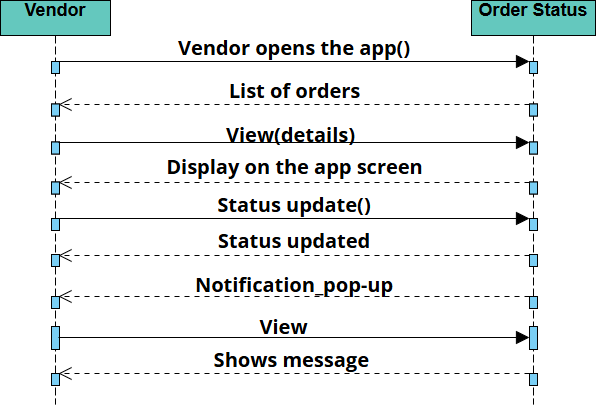
**Fig no: 8.4.1**

###### Login:

****

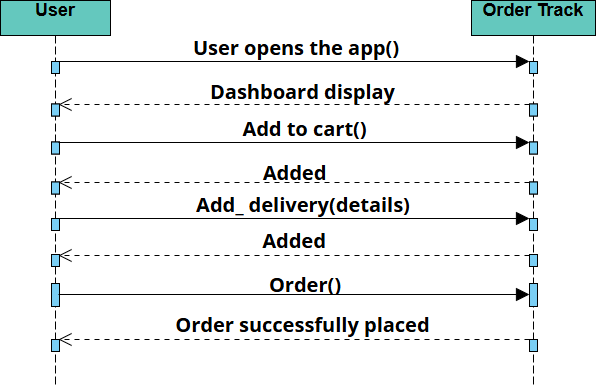
**Fig no: 8.4.2**

###### Order Management:

****

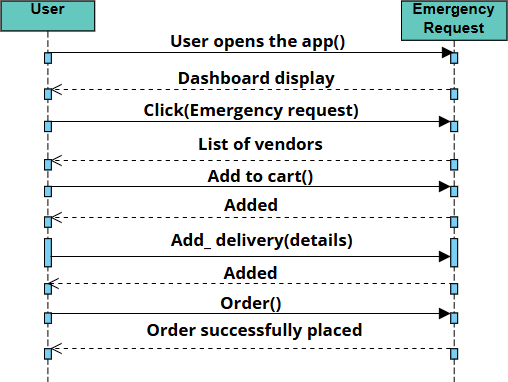
**Fig no: 8.4.3**

###### Tanker Ordering:

****

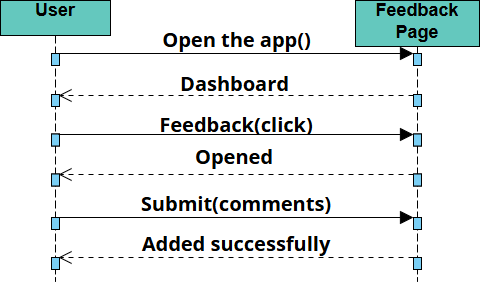
**Fig no: 8.4.4**

###### Emergency System:

****

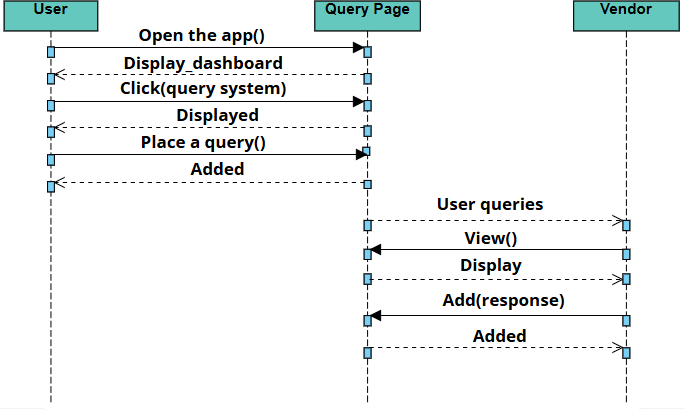
**Fig no: 8.4.5**

###### Feedback:

****

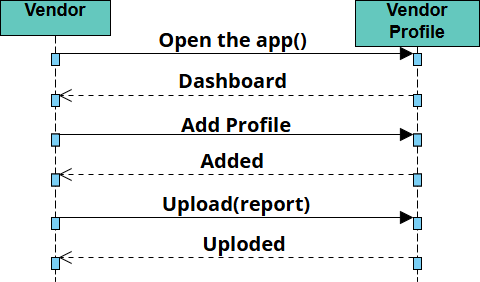
**Fig no: 8.4.6**

###### Query System:

****

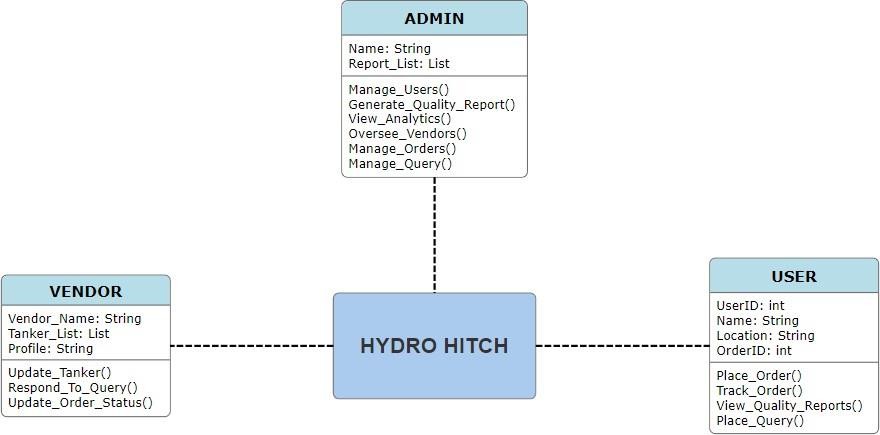
**Fig no: 8.4.7**

###### Quality Reports:

****

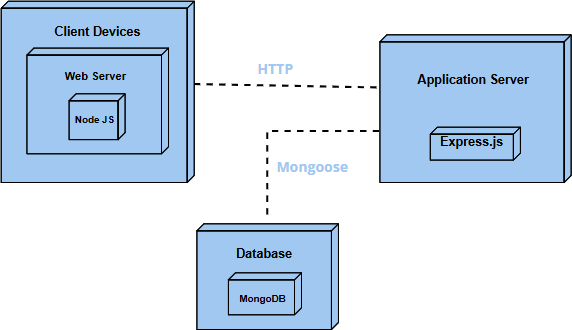
**Fig no: 8.4.8**

##### Object Diagram:

****

**Fig no: 8.5**

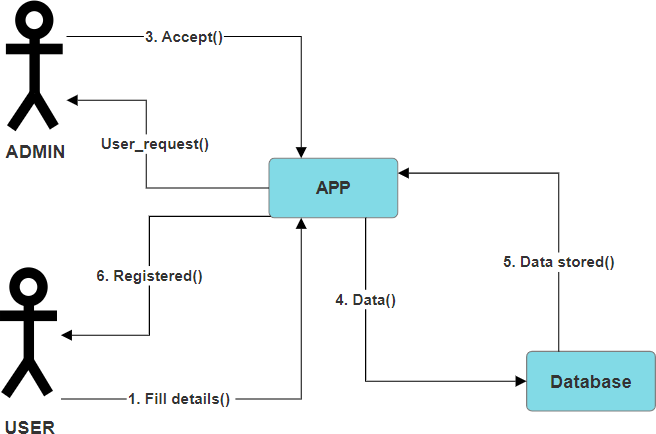
##### Deployment Diagram:

****

**Fig no: 8.6**

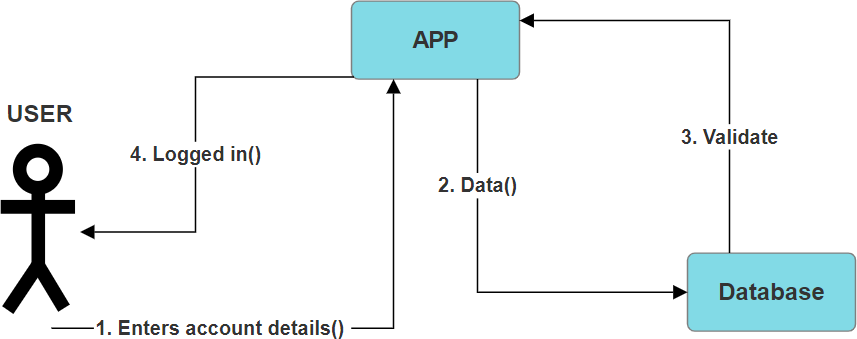
##### Collaboration Diagram:

###### Registration:

****

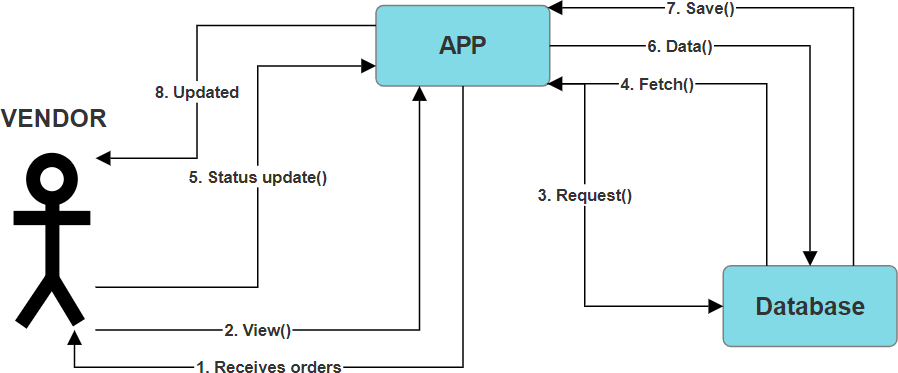
**Fig no: 8.7.1**

###### Login:

****

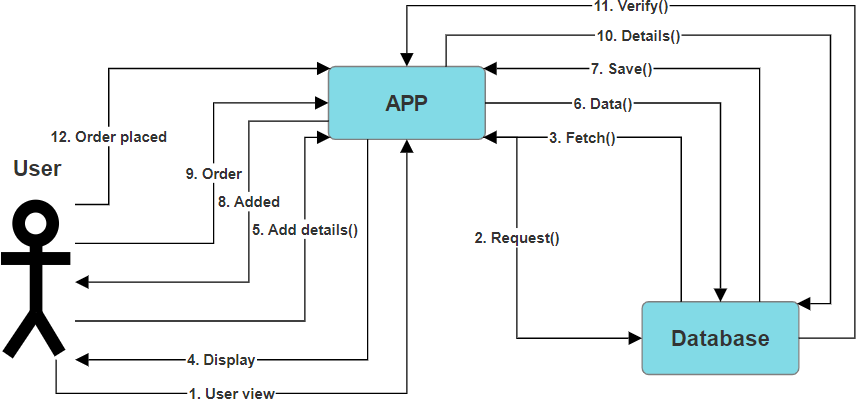
**Fig no: 8.7.2**

###### Order Management:

****

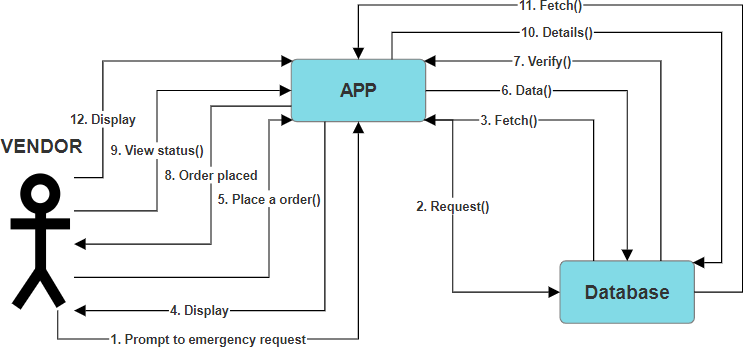
**Fig no: 8.7.3**

###### Tanker Ordering:

****

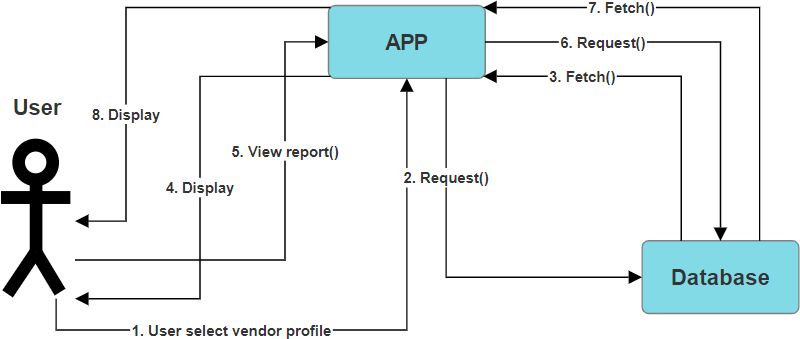
**Fig no: 8.7.4**

###### Emergency System:

****

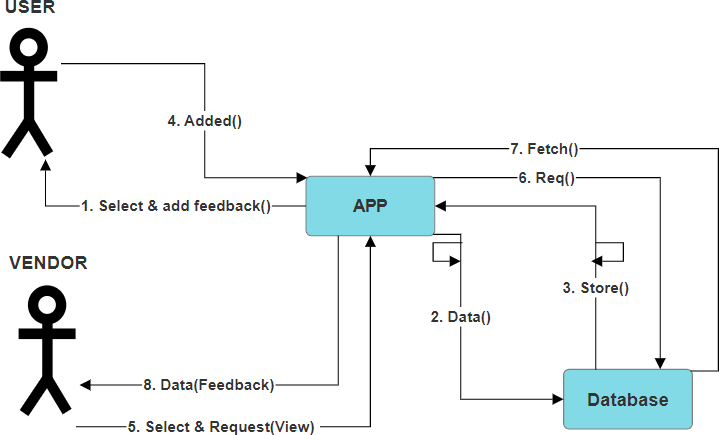
**Fig no: 8.7.5**

###### Quality Report:

****

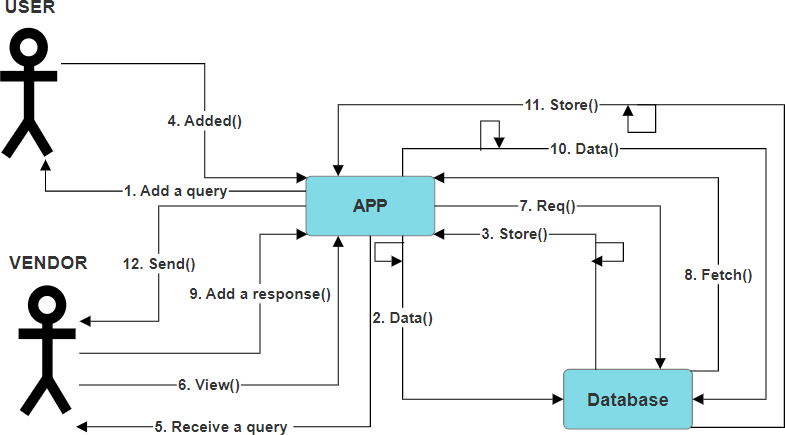
**Fig no: 8.7.6**

###### Feedback

****

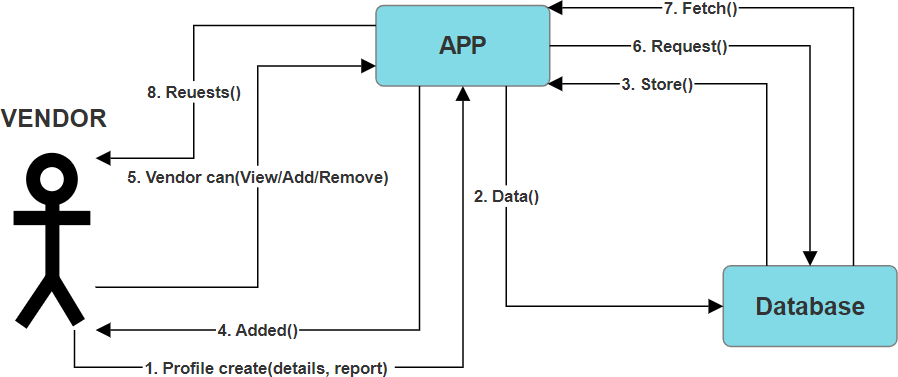
**Fig no: 8.7.7**

###### Query Service:

****

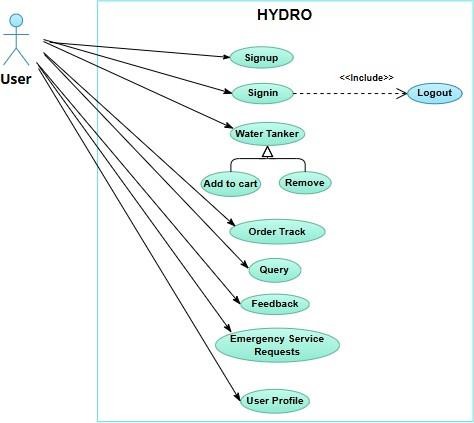
**Fig no: 8.7.8**

###### Vendor Profile:

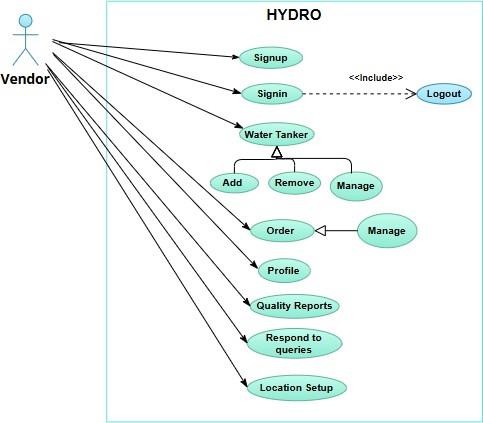
****

**Fig no: 8.7.9**

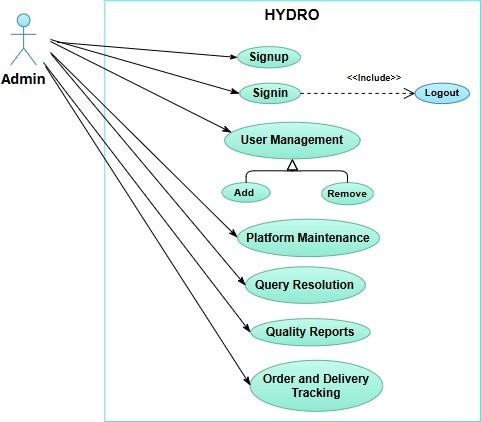
###### Use case Diagram

****

**Fig No: 8.7.10 A**

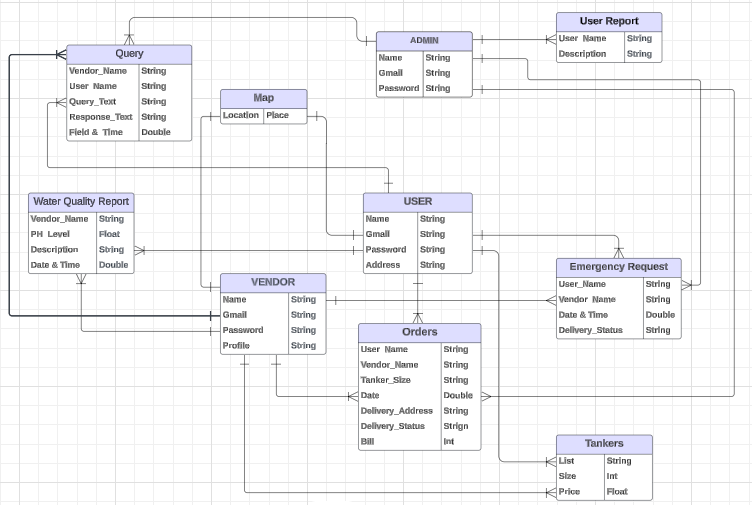


**Fig No: 8.7.10 B**



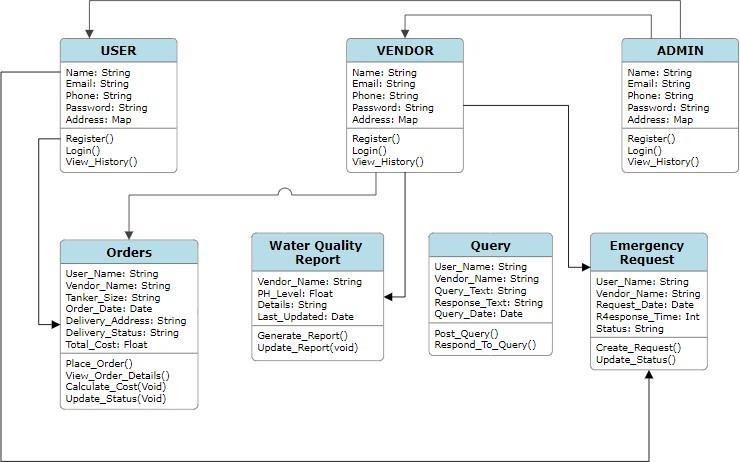
**Fig No: 8.7.10 C**

###### Entity Relation Diagram

****

**Fig No: 8.7.11**

###### Class Diagram

****

**Fig No: 8.7.12**

## SOFTWARE TESTING DOCUMENT

#### Introduction

Software testing is a critical phase in the software development lifecycle, ensuring that the developed software meets the specified requirements and is free of defects. This document outlines the approach, processes, and methodologies used for testing the HydroHitch platform, a multi-vendor water delivery application. The software will undergo various levels of testing, including functional, usability, performance, and security tests, to ensure that it operates smoothly and meets user expectations.

The testing will involve different stages, from unit testing of individual components to integration and system testing, followed by user acceptance testing (UAT). The overall objective is to ensure that HydroHitch is a reliable, efficient, and secure solution for water delivery services.

##### Purpose of this Document

The purpose of this document is to define the software testing strategy for the HydroHitch platform. It aims to:

Provide a detailed testing approach, including test phases, methodologies, and techniques.

Define the scope of testing, including functional and non-functional requirements to be validated.

Ensure that all stakeholders are aligned on testing objectives and processes.

Set clear expectations for the testing outcomes, including defect management and reporting procedures.

Identify key testing activities, responsibilities, and timelines to ensure timely delivery and quality of the software.

##### Intended Audience

Our different audience is our supervisor, Faculty, Team Members and Users.

##### Document Convention

Format: IEEE Standard Font Family:

Times New Roman Text Size: 18,16,14,11

Text Style: Bold Text Color: Black Text Alignment: Left Align

##### Project Overview

HydroHitch is a multi-vendor water delivery platform designed to meet the growing demand for dependable and high-quality water delivery services. By bringing together users, vendors, and administrators on a unified platform, the application facilitates smooth interactions and transactions within the water distribution industry. With its user-friendly interface and powerful features, this enables users to order water tankers, access quality reports, and calculate delivery costs effortlessly.

Vendors benefit from tools to manage listings, track orders, and provide detailed quality reports. Meanwhile, super admins have dedicated controls to oversee operations and ensure efficient service delivery. Built on the scalable and flexible MERN stack, it provides tailored interfaces for each user type, streamlining communication and management. The app addresses key challenges in water delivery by prioritizing quality assurance, user convenience, and effective vendor oversight, positioning itself as a reliable solution for meeting community water needs.

#### Test Cases

##### Registration-1

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 1 | **Test Name** | Registration |
| **Test Case Id** | TCI-1 | **Test Type** | Functionality |
| **Test Case Description** | This test case verifies the functionality of the user registration feature in the HYDRO app. The user should be able to create a new account by entering valid credentials such as name, email, phone number, and password. | | |
| **Test Steps** | 1. Open the HYDRO app. 2. Navigate to the Registration page. 3. Enter valid Name (e.g., John Doe). 4. Enter a valid Email (e.g., [manoj@example.com).](mailto:manoj@example.com) 5. Enter a valid Phone Number (e.g., +1234567890). 6. Enter a Password (e.g., P@ssword123). 7. Confirm the password. 8. Click the Register button. | | |
| **Expected Result** | The app should display a confirmation message: "Registration Successful." The user should be redirected to the Login page. The user's data should be stored in the database. | | |
| **Actual Result** | Account successfully created and redirected to the dashboard. | | |
| **Pass/Fail** | Pass | | |

##### Registration-2

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 1 | **Test Name** | Registration |
| **Test Case Id** | TCI-2 | **Test Type** | Functionality |
| **Test Case Description** | This test case verifies how the system handles invalid inputs during the user registration process. The user tries to register with missing or incorrect credentials (e.g., invalid email format, missing password). | | |
| **Test Steps** | 1. Open the HYDRO app. 2. Navigate to the Registration page. 3. Enter valid Name (e.g. Jatin kumar). 4. Enter an Invalid Email (e.g., [123@invalid.123).](mailto:123@invalid.123) 5. Enter a valid Phone Number (e.g., +1234567890). 6. Leave the Password field empty. 7. Click the Register button. | | |
| **Expected Result** | The app should display an error message: "Please enter a valid email address." The app should display another error message: "Password field cannot be empty." The registration should fail, and no data should be stored in the database. | | |
| **Actual Result** | Error message displayed. Registration is not allowed until all fields are properly filled. | | |
| **Pass/Fail** | Pass | | |

##### Registration-3

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 1 | **Test Name** | Registration |
| **Test Case Id** | TCI-3 | **Test Type** | Functionality |
| **Test Case Description** | Registration with Invalid Email Format | | |
| **Test Steps** | 1. Navigate to the registration page. 2. Enter an invalid email format (e.g., userexample.com, missing @). 3. Enter a valid password (Pass@123). 4. Confirm the password by re-entering it. 5. Fill in other required fields (name, phone number, etc.). 6. Click the "Register" button. | | |
| **Expected Result** | Error message displayed: "Invalid email format. Please enter a valid email." Registration is not allowed. | | |
| **Actual Result** | Registration accepted without validating the email format. | | |
| **Pass/Fail** | Fail | | |

##### Login-1

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 2 | **Test Name** | Login |
| **Test Case Id** | TCI-4 | **Test Type** | Functionality |
| **Test Case Description** | Verify the login functionality of the HYDRO app to ensure that users can successfully log in using valid credentials. | | |
| **Test Steps** | 1. Open the HYDRO app. 2. Navigate to the Login screen. 3. Enter a valid Email/Phone Number in the login field. 4. Enter the correct Password associated with the account. 5. Click the Login button. 6. Observe the system's response. | | |
| **Expected Result** | The user is successfully logged in and redirected to the app's dashboard.  A welcome message or dashboard appears, showing the user is authenticated. | | |
| **Actual Result** | The system displays a "Login Successful" message and redirects to the user dashboard. | | |
| **Pass/Fail** | Pass | | |

##### Login-2

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 2 | **Test Name** | Login |
| **Test Case Id** | TCI-5 | **Test Type** | Functionality |
| **Test Case Description** | Verify the login functionality of the HYDRO app to ensure that users can handle invalid credentials appropriately. | | |
| **Test Steps** | 1. Open the HYDRO app. 2. Navigate to the Login screen. 3. Enter an invalid Email/Phone Number in the login field or incorrect Password associated with the account. 4. Click the Login button. 5. Observe the system's response. 6. Observe the system's response for incorrect credentials. | | |
| **Expected Result** | An error message like "Invalid email or password" is displayed. The user remains on the login screen without being logged in. | | |
| **Actual Result** | The system displays an "Incorrect password / email" message without granting access to the dashboard. | | |
| **Pass/Fail** | Pass | | |

##### Login-2

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 2 | **Test Name** | Login |
| **Test Case Id** | TCI-6 | **Test Type** | Functionality |
| **Test Case Description** | Verify that the system handles login attempts with non-existent user credentials. | | |
| **Test Steps** | 1. Open the HydroHitch application. 2. Navigate to the login page. 3. Enter an invalid or non-registered email/phone number. 4. Enter any password. 5. Click the "Login" button. | | |
| **Expected Result** | The system displays a "User not found" message without granting access. | | |
| **Actual Result** | The system incorrectly allows login or does not display the appropriate error message. | | |
| **Pass/Fail** | Fail | | |

##### Tanker Ordering-1

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 3 | **Test Name** | Tanker Ordering |
| **Test Case Id** | TCI-7 | **Test Type** | Functionality |
| **Test Case Description** | Successful tanker order placement | | |
| **Test Steps** | 1. Log in as a registered user. 2. Navigate to the "Order Water Tanker" section. 3. Select the desired tanker capacity. 4. Enter the delivery address or confirm location via GPS. 5. Confirm the order details and submit. | | |
| **Expected Result** | The system should display a confirmation message with the order ID and estimated delivery time. | | |
| **Actual Result** | The order was placed successfully with order confirmation displayed. | | |
| **Pass/Fail** | Pass | | |

##### Tanker Ordering-2

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 3 | **Test Name** | Tanker Ordering |
| **Test Case Id** | TCI-8 | **Test Type** | Functionality |
| **Test Case Description** | Update order details | | |
| **Test Steps** | 1. Log in as a registered user. 2. Navigate to the "My Orders" section. 3. Select an active order from the order list. 4. Modify the delivery address or update tanker capacity. 5. Confirm the changes and save. | | |
| **Expected Result** | The system should update the order details and display a success message. The updated details should be reflected in the order summary. | | |
| **Actual Result** | The order details were updated correctly, and a confirmation message appeared. | | |
| **Pass/Fail** | Pass | | |

##### Tanker Ordering-3

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 3 | **Test Name** | Tanker Ordering |
| **Test Case Id** | TCI-9 | **Test Type** | Functionality |
| **Test Case Description** | Failed tanker order placement due to missing contact information | | |
| **Test Steps** | 1. Navigate to the HydroHitch app and log in as a registered user. 2. Go to the "Order Water Tanker" section. 3. Select the tanker capacity (e.g., 1500 gallons). 4. Choose a vendor from the list. 5. Leave the contact information field empty. 6. Enter a valid delivery address. 7. Click on the "Place Order" button. | | |
| **Expected Result** | The system should display an error message prompting the user to provide valid contact information before proceeding with the order. | | |
| **Actual Result** | The order was placed successfully with order confirmation displayed. | | |
| **Pass/Fail** | Fail | | |

##### Order Management-1

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 4 | **Test Name** | Order Management |
| **Test Case Id** | TCI-10 | **Test Type** | Functionality |
| **Test Case Description** | Verify that the vendor can update the order status correctly. | | |
| **Test Steps** | 1. Log in to the vendor panel using valid credentials. 2. Navigate to the "Manage Orders" section. 3. Select an order with the status "Processing." 4. Update the order status to "Dispatched" or "Delivered." Save the changes. 5. Verify that the updated status reflects in the user order tracking. | | |
| **Expected Result** | The order status is successfully updated, and the user is notified of the status change via a push notification or email. | | |
| **Actual Result** | Order status updated and reflected in the user dashboard. | | |
| **Pass/Fail** | Pass | | |

##### Order Management-2

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 4 | **Test Name** | Order Management |
| **Test Case Id** | TCI-11 | **Test Type** | Functionality |
| **Test Case Description** | Verify that the vendor can successfully add a new tanker to their fleet through the vendor panel. | | |
| **Test Steps** | 1. Log in as a vendor with valid credentials. 2. Navigate to the "Manage Tankers" section in the vendor panel. 3. Click on the "Add New Tanker" button. 4. Fill in the required details (e.g., tanker capacity, pricing, and availability). 5. Submit the form by clicking the "Save" button. 6. Verify that the new tanker appears in the tanker list. | | |
| **Expected Result** | The new tanker is successfully added, and a confirmation message is displayed. The tanker appears in the vendor's list of available tankers. | | |
| **Actual Result** | The tanker is added, and the vendor panel reflects the new tanker with correct details. | | |
| **Pass/Fail** | Pass | | |

##### Order Management-3

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 4 | **Test Name** | Order Management |
| **Test Case Id** | TCI-12 | **Test Type** | Functionality |
| **Test Case Description** | Verify that if a vendor cancels an order, the cancellation status is properly updated for the user. | | |
| **Test Steps** | 1. Log in as a vendor using valid credentials. 2. Navigate to the "Order Management" section. 3. Select an active order. 4. Change the order status to "Cancelled" and save the changes. 5. Check the user dashboard for the order status update. | | |
| **Expected Result** | The user should be notified that the order has been cancelled, and the order status should reflect "Cancelled" on their dashboard. | | |
| **Actual Result** | The user does not receive any notification, and the order status remains unchanged on their dashboard. | | |
| **Pass/Fail** | Fail | | |

##### Feedback-1

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 5 | **Test Name** | Feedback |
| **Test Case Id** | TCI-13 | **Test Type** | Functionality |
| **Test Case Description** | Submit feedback | | |
| **Test Steps** | 1. Log in to the user account. 2. Navigate to the "Feedback" section in the dashboard. 3. Enter a comment in the feedback text area (e.g., "Great service!"). 4. Submit the feedback by clicking the "Submit" button. 5. Verify if the confirmation message appears (e.g., "Feedback submitted successfully"). | | |
| **Expected Result** | Feedback is submitted, and a confirmation message is displayed. The feedback entry is recorded in the database. | | |
| **Actual Result** | Feedback was successfully recorded, and the confirmation message appeared. | | |
| **Pass/Fail** | Pass | | |

##### Feedback-2

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 5 | **Test Name** | Feedback |
| **Test Case Id** | TCI-14 | **Test Type** | Functionality |
| **Test Case Description** | Submit empty feedback | | |
| **Test Steps** | 1. Log in to the user account. 2. Navigate to the "Feedback" section in the dashboard. 3. Leave the feedback text area empty. 4. Click the "Submit" button. 5. Verify if an error message is displayed (e.g., "Feedback cannot be empty"). | | |
| **Expected Result** | An error message is displayed, and feedback submission is prevented. | | |
| **Actual Result** | An error message appeared, and no feedback was submitted. | | |
| **Pass/Fail** | Pass | | |

##### Feedback-3

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 5 | **Test Name** | Feedback |
| **Test Case Id** | TCI-15 | **Test Type** | Functionality |
| **Test Case Description** | Test the scenario where the user attempts to submit feedback with invalid characters (e.g., special characters or HTML tags) in the comment. | | |
| **Test Steps** | 1. Login as a user. 2. Navigate to the "Order History" section. 3. Select an order from the order list. 4. Click on the "Submit Feedback" button. 5. Enter a rating (e.g., 4 stars) and include invalid characters like   <script>alert("XSS")</script> or special symbols in the comment.   1. Click "Submit". | | |
| **Expected Result** | The system should display an error message indicating invalid characters are not allowed in the comment field. The feedback should not be submitted. | | |
| **Actual Result** | system accepts and submits the feedback with invalid characters | | |
| **Pass/Fail** | Fail | | |

##### Query System-1

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 6 | **Test Name** | Query System |
| **Test Case Id** | TCI-16 | **Test Type** | Functionality |
| **Test Case Description** | Submit a query | | |
| **Test Steps** | 1. Login to the system as a registered user. 2. Navigate to the "Query and Feedback" section. 3. Click on the "Submit a Query" button. 4. Enter a query in the input field (e.g., "What is the delivery time for Vendor X?"). 5. Click on the "Submit" button. | | |
| **Expected Result** | The query is successfully submitted, and a confirmation message ("Query submitted successfully") is displayed. | | |
| **Actual Result** | The query was submitted, and a confirmation message was displayed. | | |
| **Pass/Fail** | Pass | | |

##### Query System-2

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 6 | **Test Name** | Query System |
| **Test Case Id** | TCI-17 | **Test Type** | Functionality |
| **Test Case Description** | View vendor response to a query | | |
| **Test Steps** | 1. Login to the system as a registered user. 2. Navigate to the "Query and Feedback" section. 3. Locate a previously submitted query in the query history. 4. Click on the query to view the response. | | |
| **Expected Result** | The query details and the vendor’s response are displayed in a readable format. | | |
| **Actual Result** | The response was displayed as expected. | | |
| **Pass/Fail** | Pass | | |

##### Query System-3

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 6 | **Test Name** | Query System |
| **Test Case Id** | TCI-18 | **Test Type** | Functionality |
| **Test Case Description** | Test the behavior when a user attempts to submit a query without filling in the required fields. | | |
| **Test Steps** | 1. Login as a user. 2. Navigate to the "Query & Feedback" section. 3. Click on "Submit Query". 4. Leave the "Query Subject" and "Query Description" fields empty. 5. Click the "Submit" button. | | |
| **Expected Result** | The system should display an error message prompting the user to fill in the required fields ("Query Subject" and "Query Description"). | | |
| **Actual Result** | no validation message is shown or the query is submitted without the required fields) | | |
| **Pass/Fail** | Fail | | |

##### Quality Report-1

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 7 | **Test Name** | Quality Report |
| **Test Case Id** | TCI-19 | **Test Type** | Functionality |
| **Test Case Description** | Verify viewing of quality report | | |
| **Test Steps** | 1. Log in to the application with valid credentials. 2. Navigate to the "Orders" section. 3. Select an order with a completed status. 4. Click on the "View Quality Report" button for the selected order. | | |
| **Expected Result** | The quality report displays details such as pH level, contamination data, and source information for the selected order. | | |
| **Actual Result** | The water quality report is displayed with accurate pH level (7.5) and other details. | | |
| **Pass/Fail** | Pass | | |

##### Quality Report-2

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 7 | **Test Name** | Quality Report |
| **Test Case Id** | TCI-20 | **Test Type** | Functionality |
| **Test Case Description** | Verify error message for missing quality report | | |
| **Test Steps** | 1. Log in to the application with valid credentials. 2. Navigate to the "Orders" section. 3. Select an order that does not have a quality report available. 4. Click on the "View Quality Report" button for the selected order. | | |
| **Expected Result** | An error message appears: "Quality report not available for this order." | | |
| **Actual Result** | The system displayed the message: "Water quality report not available for this vendor. Please contact the vendor for more details." | | |
| **Pass/Fail** | Pass | | |

##### Quality Report-3

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 7 | **Test Name** | Quality Report |
| **Test Case Id** | TCI-21 | **Test Type** | Functionality |
| **Test Case Description** | Verify Water Quality Report Display with Missing pH Level | | |
| **Test Steps** | 1. Log in as a user. 2. Search for a vendor by location. 3. Select a vendor from the list. 4. Click on the "Water Quality Report" option. 5. Verify that the report displays an appropriate message or indication when the pH level data is missing. | | |
| **Expected Result** | The system should display a message like "pH level data unavailable" or leave the pH level section blank with a note indicating missing information. | | |
| **Actual Result** | The water quality report displayed a blank pH level field, but no message or indication was shown that the pH data was missing. | | |
| **Pass/Fail** | Fail | | |

##### Emergency System-1

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 8 | **Test Name** | Emergency System |
| **Test Case Id** | TCI-22 | **Test Type** | Functionality |
| **Test Case Description** |  |  |  |
| **Test Case Description** | Emergency service request by user | | |
| **Test Steps** | 1. Open the HydroHitch app. 2. Navigate to the emergency service section. 3. Enter required details (e.g., address, water quantity). 4. Click "Submit" to request emergency water delivery. | | |
| **Expected Result** | The system should accept the emergency request and show a confirmation message with estimated delivery time. | | |
| **Actual Result** | The emergency service request was successfully submitted, and a confirmation message with an estimated delivery time of 2 minutes appeared. | | |
| **Pass/Fail** | Pass | | |

##### Emergency System-2

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 8 | **Test Name** | Emergency System |
| **Test Case Id** | TCI-23 | **Test Type** | Functionality |
| **Test Case Description** | Emergency service request without address | | |
| **Test Steps** | 1. Open the HydroHitch app. 2. Navigate to the emergency service section. 3. Leave the address field empty and submit the request. | | |
| **Expected Result** | The system should display an error message indicating that the address field is required before proceeding. | | |
| **Actual Result** | The system displayed an error message: "Please fill in the delivery address." and did not submit the request. | | |
| **Pass/Fail** | Pass | | |

##### Emergency System-3

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 8 | **Test Name** | Emergency System |
| **Test Case Id** | TCI-24 | **Test Type** | Functionality |
| **Test Case Description** | Test the system's behavior when there is a failure in submitting the emergency water delivery request due to server issues. | | |
| **Test Steps** | 1. Open the HydroHitch app. 2. Navigate to the "Emergency Service" section. 3. Enter the required details (e.g., delivery address, water volume). 4. Click on the "Submit Request" button. 5. Confirm that the request fails to be submitted due to server unavailability. | | |
| **Expected Result** | The system should display an error message like "Service unavailable. Please try again later." and prevent the submission. | | |
| **Actual Result** | The app showed a "Server Error" message and failed to submit the emergency request. | | |
| **Pass/Fail** | Fail | | |

##### Cost Calculator-1

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 9 | **Test Name** | Cost Calculator |
| **Test Case Id** | TCI-25 | **Test Type** | Functionality |
| **Test Case Description** | Test the delivery cost calculation for a valid user order based on location, volume, and vendor pricing. | | |
| **Test Steps** | 1. Open the HydroHitch app and log in as a user. 2. Navigate to the "Order Water" section. 3. Select a water vendor. 4. Input the required delivery volume (e.g., 1000 liters). 5. Enter the delivery location (e.g., Karachi). 6. Click on the "Calculate Delivery Cost" button. | | |
| **Expected Result** | The delivery cost is calculated correctly based on the vendor pricing, location, and volume. The cost is displayed in the "Cost Estimate" section. | | |
| **Actual Result** | The delivery cost was calculated as 1000 PKR based on the vendor's pricing and the location of Karachi. | | |
| **Pass/Fail** | Pass | | |

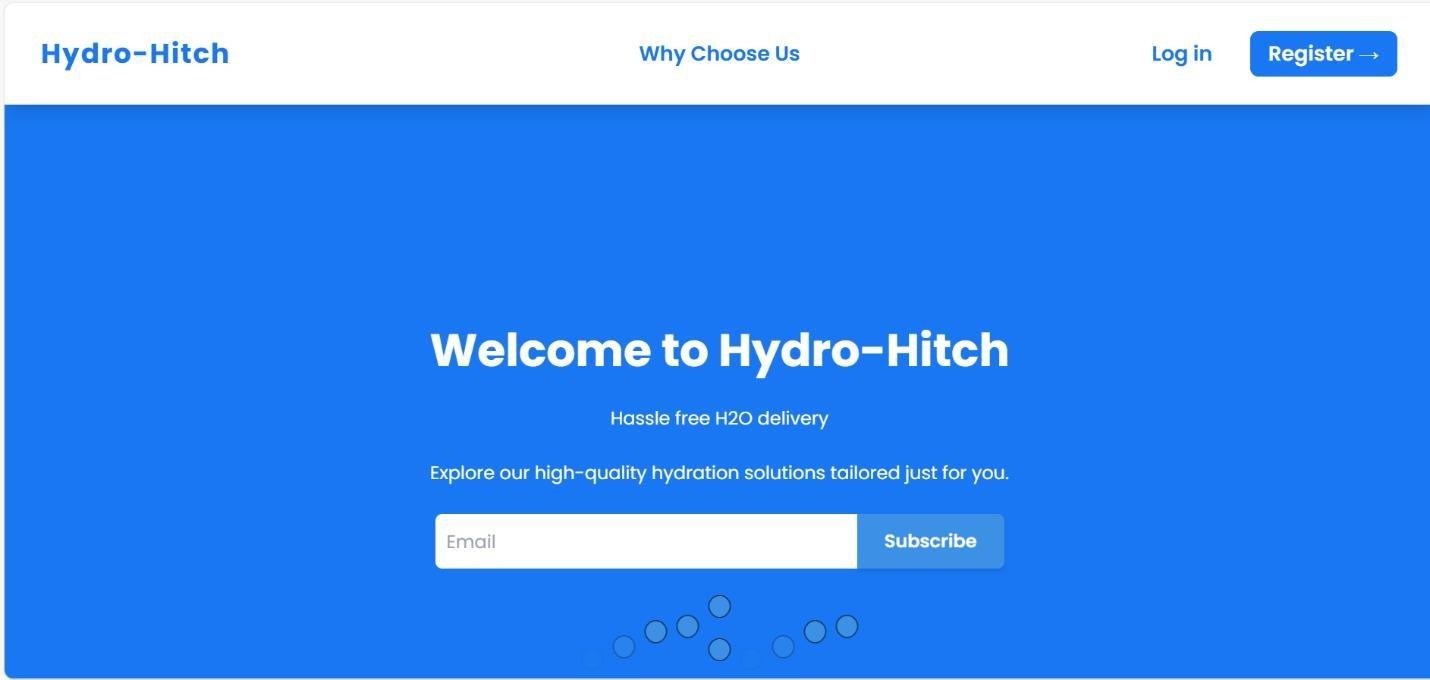
##### Cost Calculator-2

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement Reference** | 8 | **Test Name** | Cost Calculator |
| **Test Case Id** | TCI-26 | **Test Type** | Functionality |
| **Test Case Description** | Test the delivery cost calculation when the selected vendor is unavailable for the entered location. | | |
| **Test Steps** | 1. Open the HydroHitch app and log in as a user. 2. Navigate to the "Order Water" section. 3. Select a water vendor that does not service the entered location (e.g., selecting Vendor A for Lahore while located in Karachi). 4. Input the required delivery volume (e.g., 500 liters). 5. Enter the delivery location (e.g., Karachi). 6. Click on the "Calculate Delivery Cost" button. | | |
| **Expected Result** | An error message is displayed stating that the selected vendor does not provide service to the entered location, and no cost is calculated. | | |
| **Actual Result** | The error message "Vendor does not service this location" appeared, and no cost was calculated. | | |
| **Pass/Fail** | Fail | | |

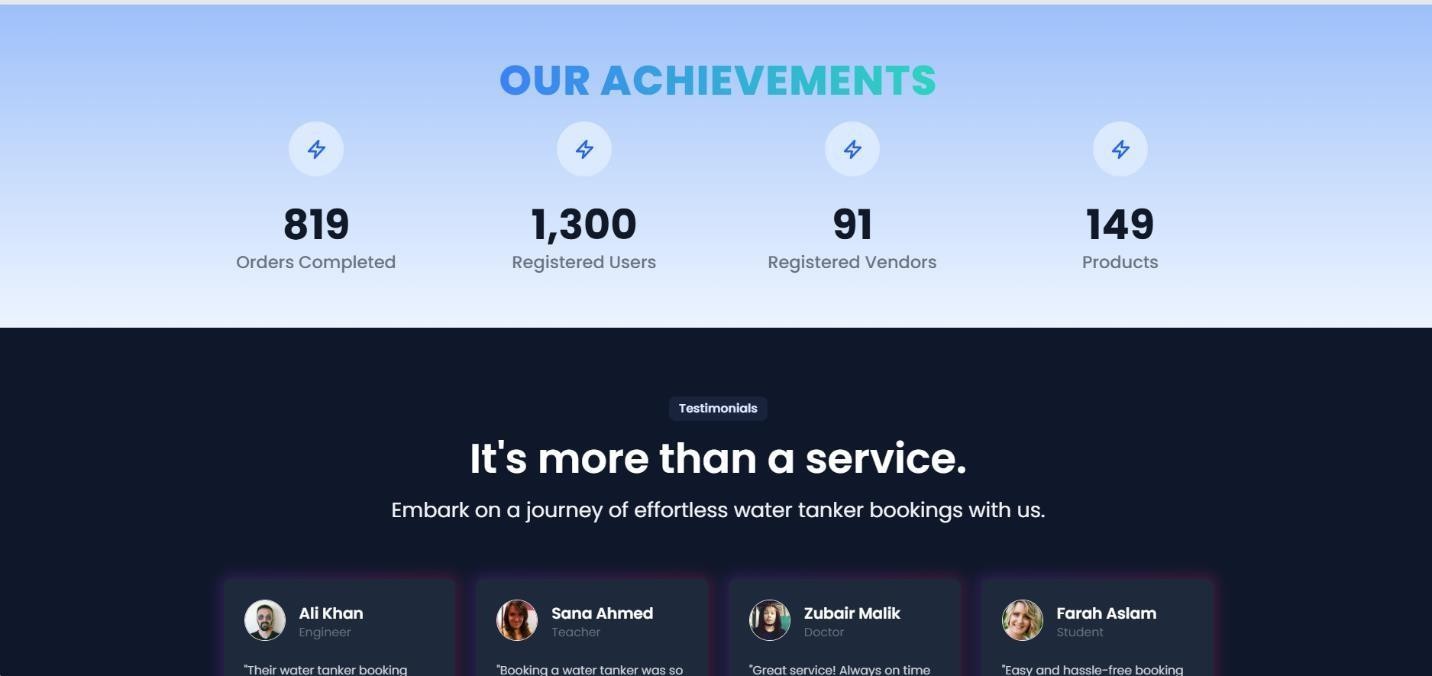
## USER MANUAL

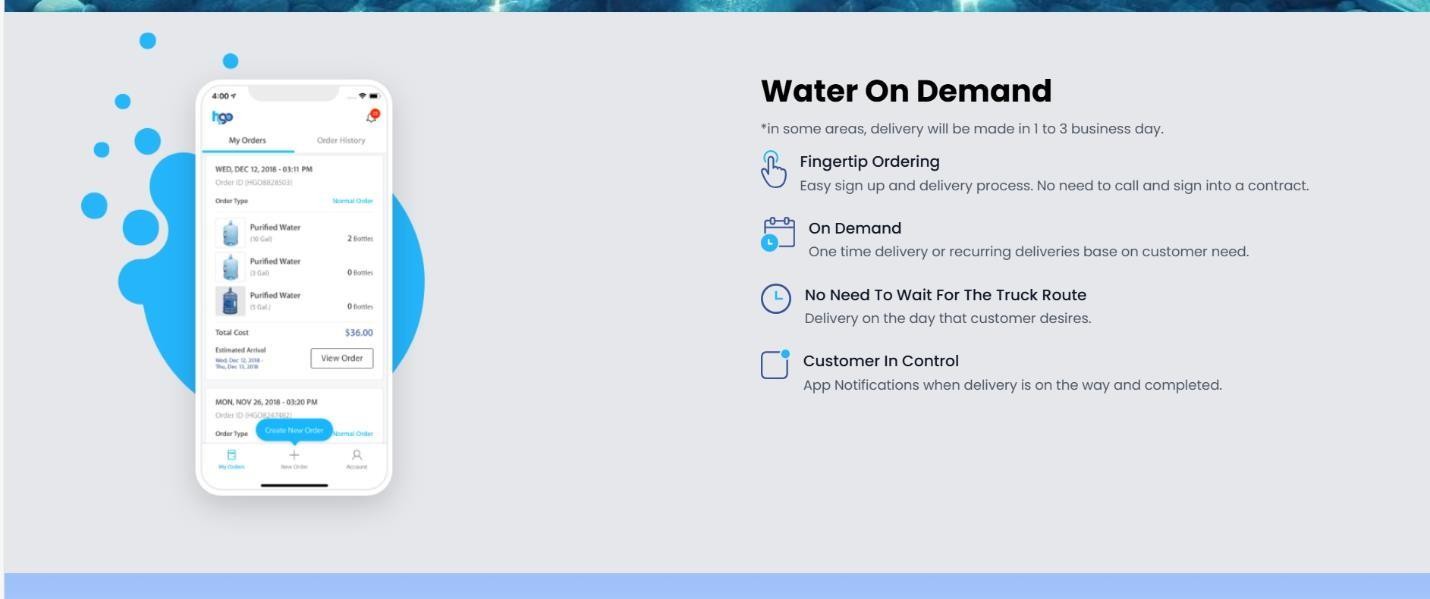
#### Website Screens

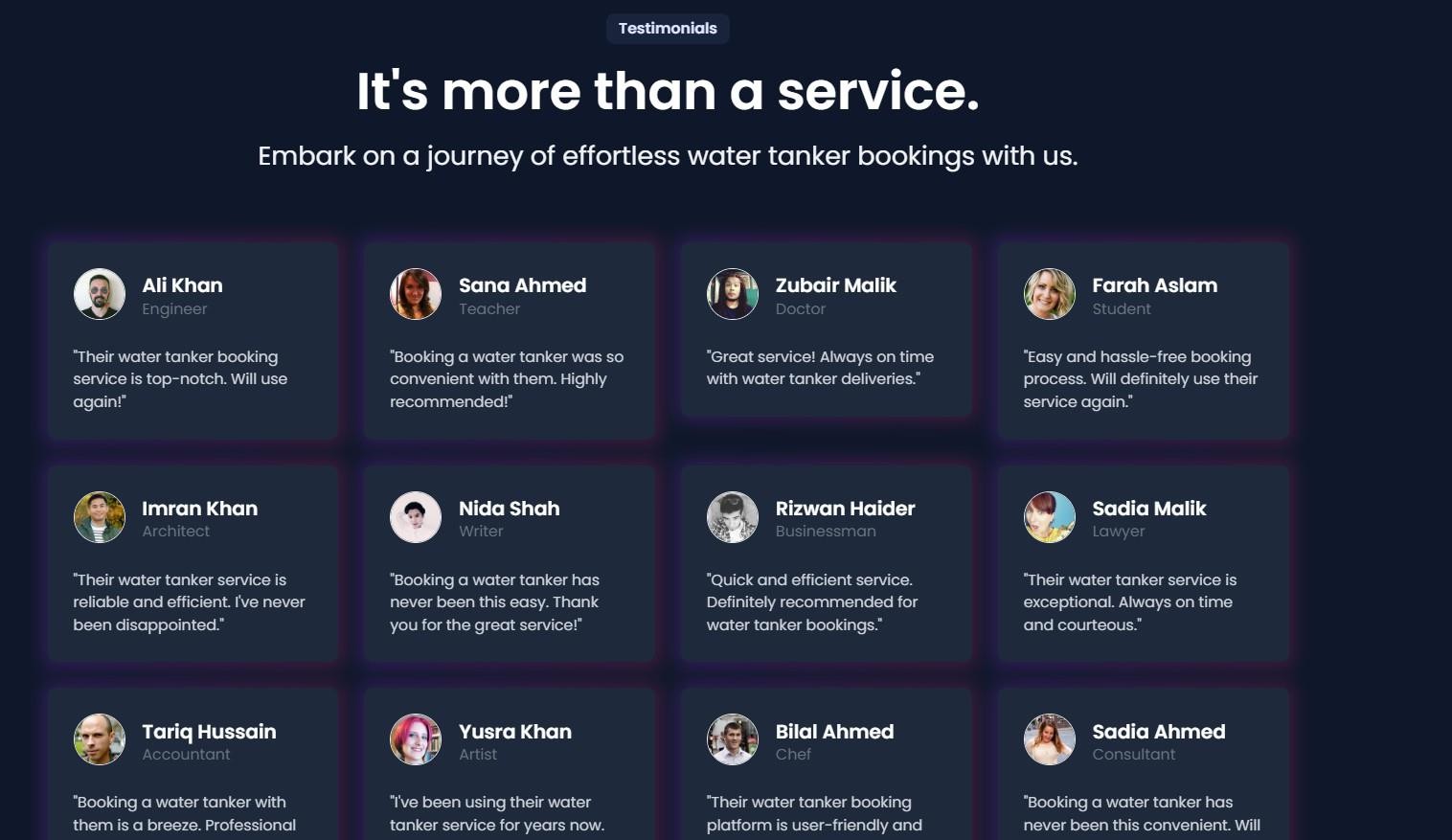
##### Splashes

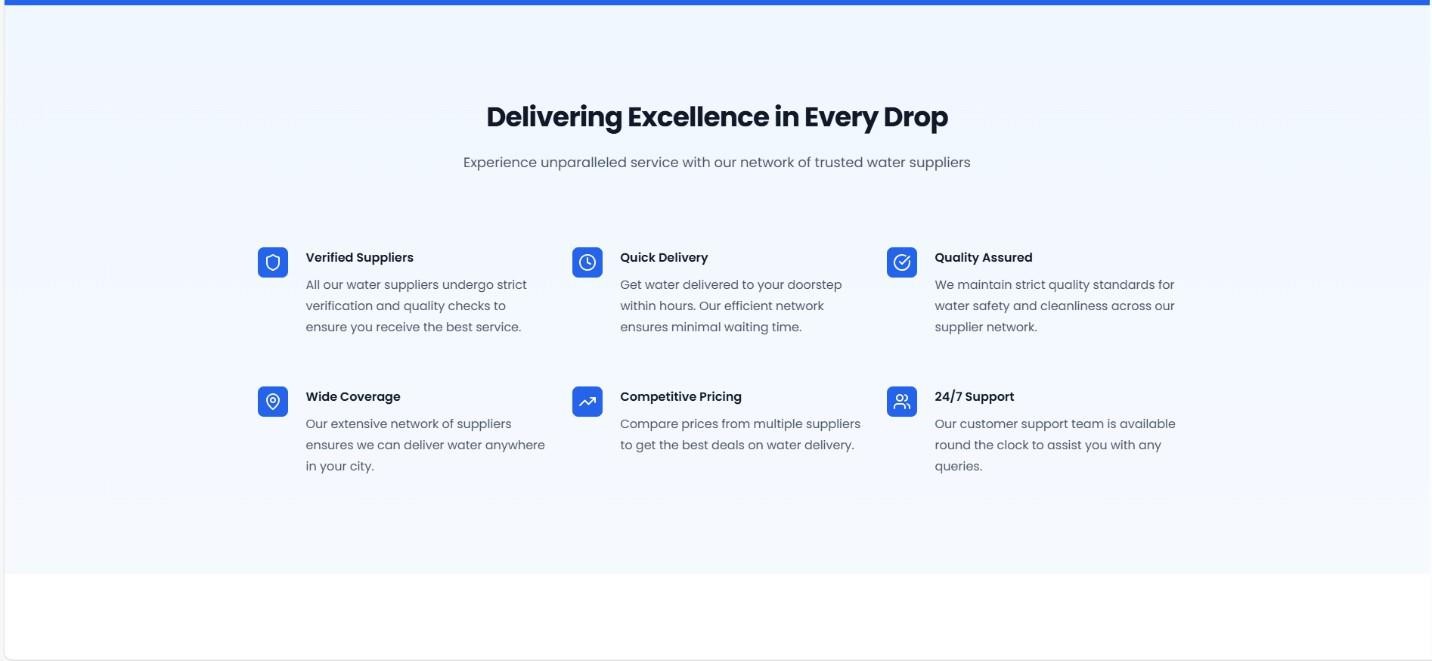
****

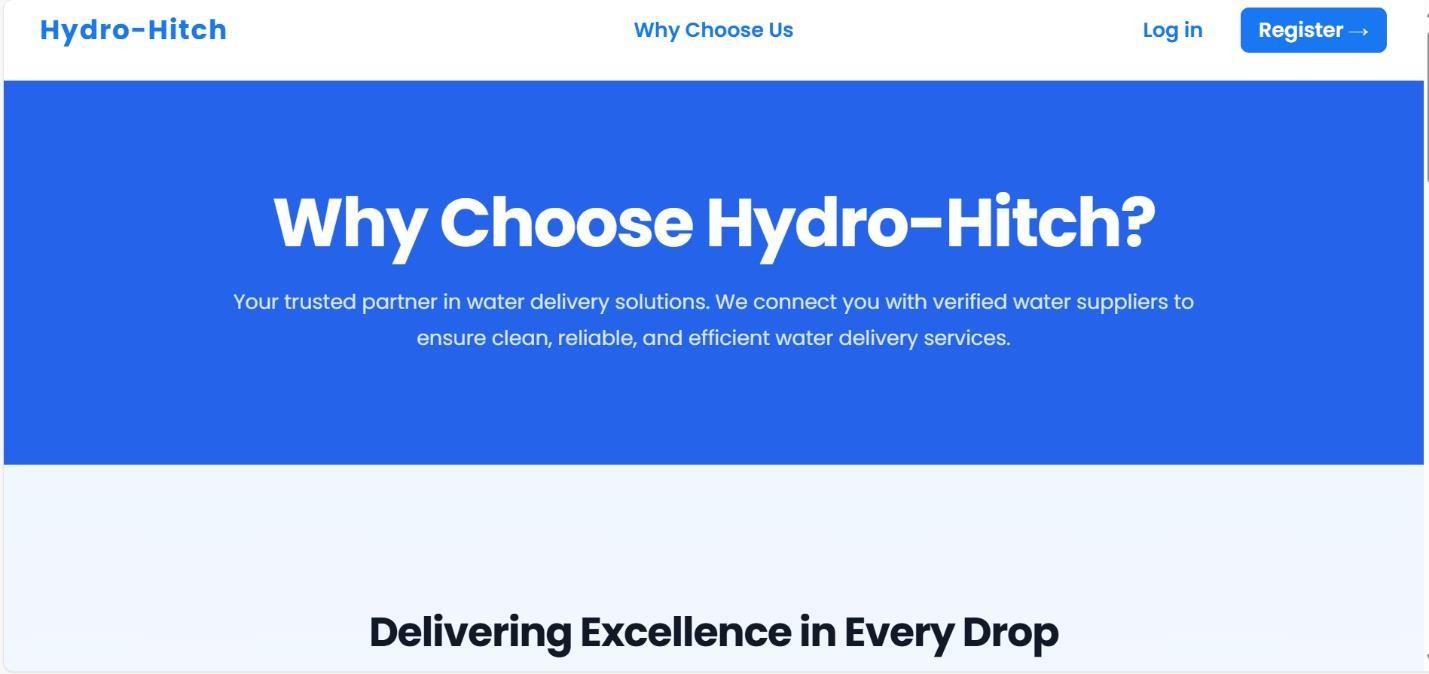
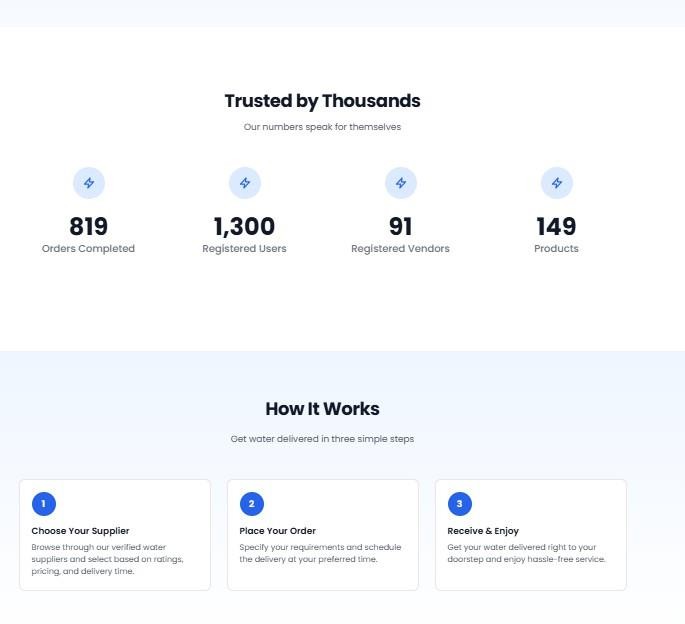
****

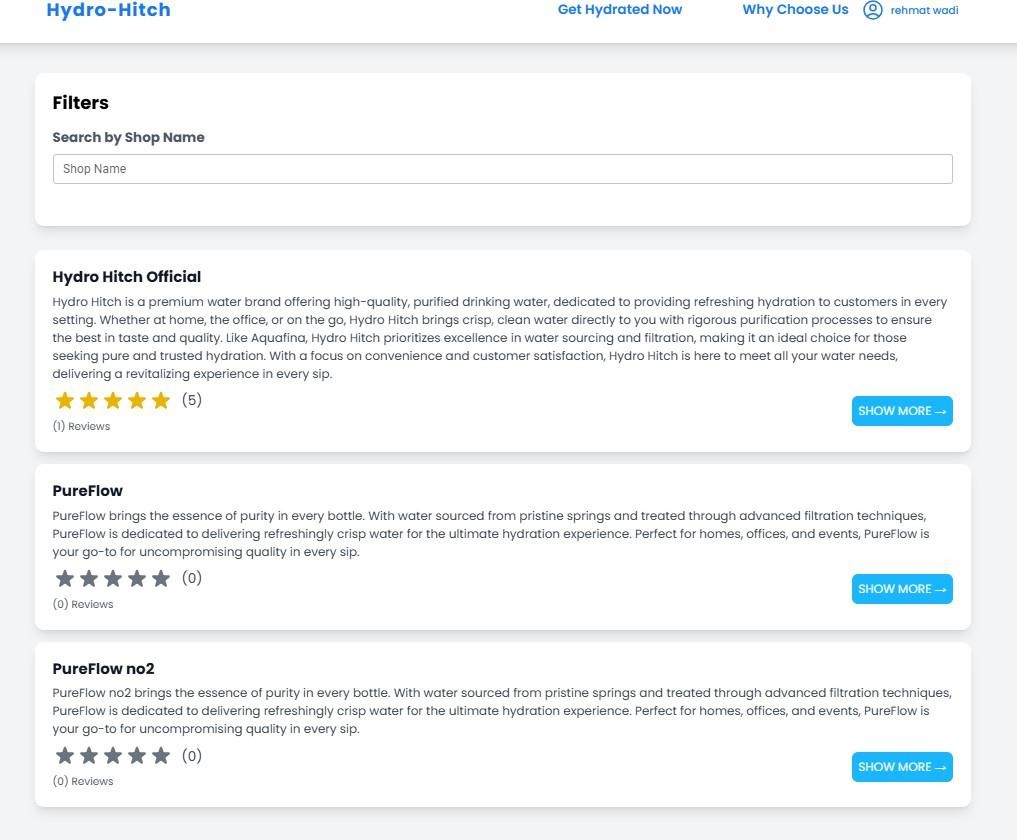
****

****

****

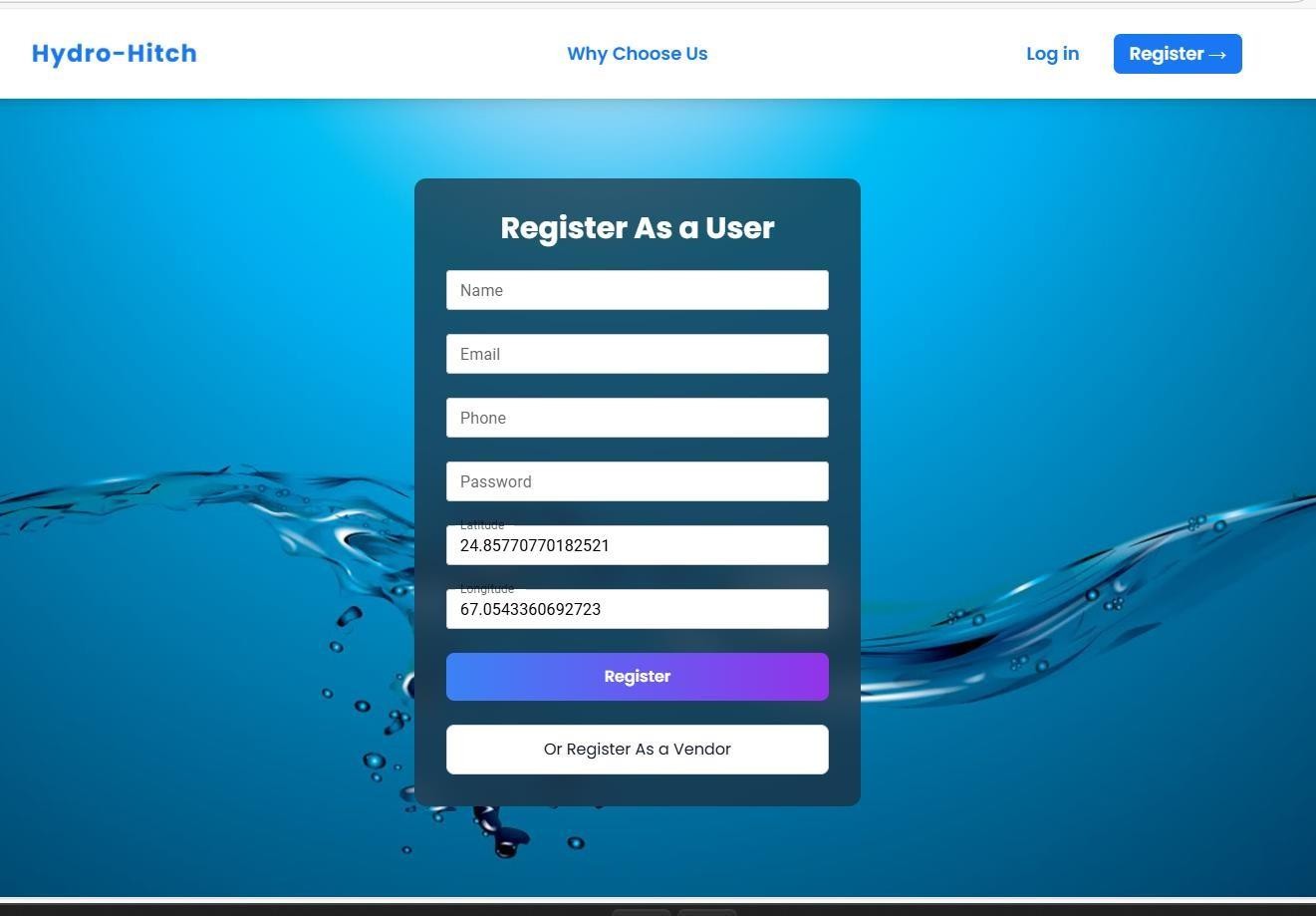
****

****



##### Register as user

* Go to the registration page.
* Fill in the required information, such as your name, email, Phone and password.
* Submit the form to create your user account.
* Validate the user account.

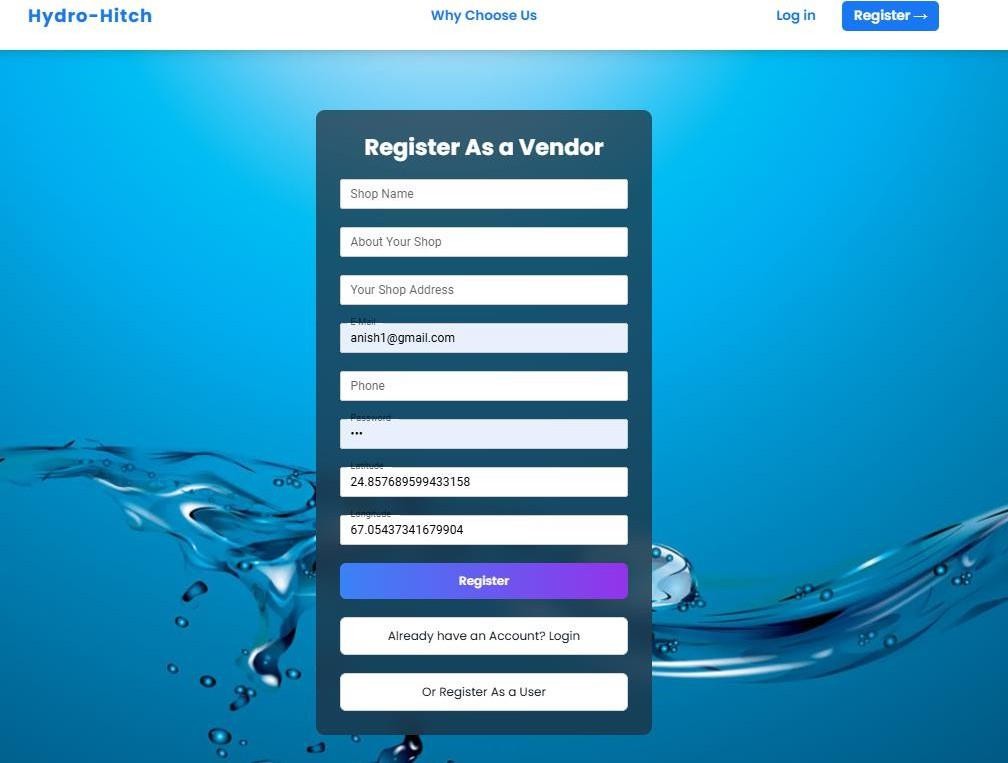
****

##### Register as Vendor

##### Go to the registration page.

##### Fill in the required vendor details.

##### Submit the form to create your vendor account.

****

##### Login as Vendor

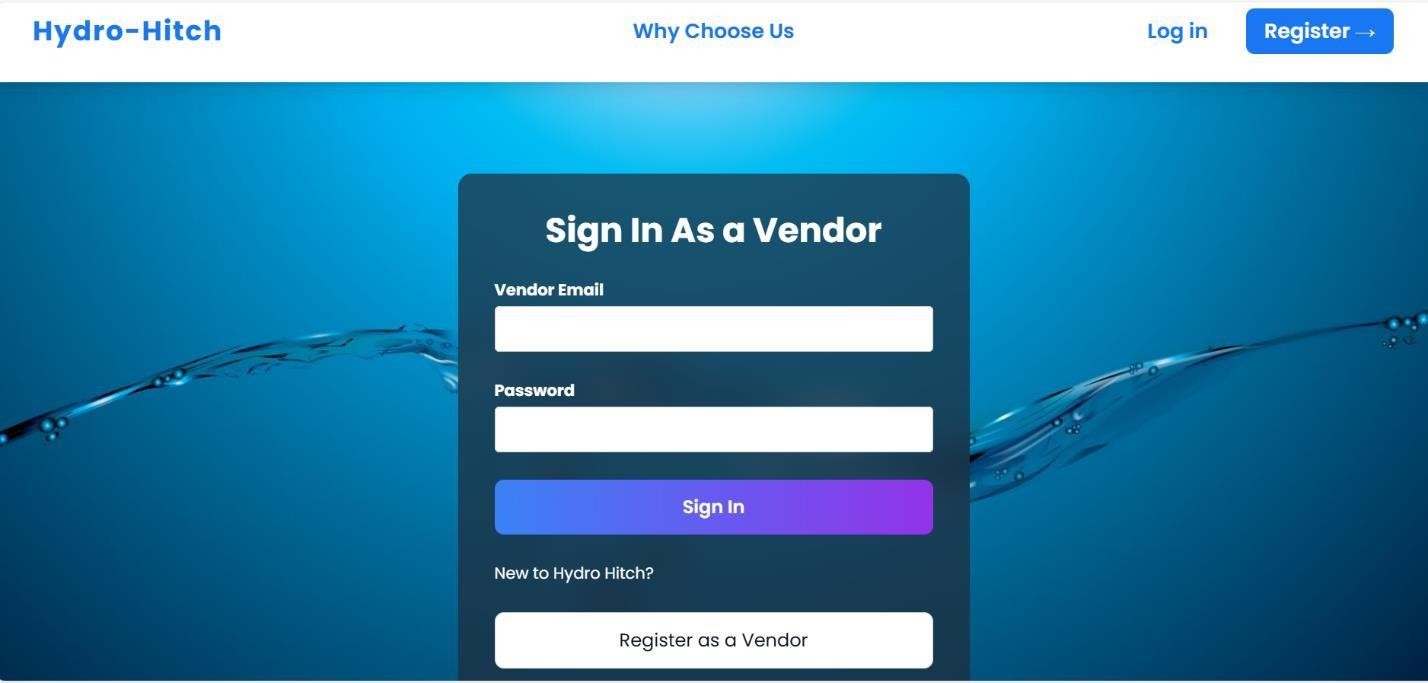
##### Open the app and go to the login screen.

##### Enter your registered email or phone number and your password.

##### Click the "Login" button.

##### The system will validate your credentials.

##### If your credentials are valid, the system will log you in and redirect you to the main dashboard.



##### Login as User

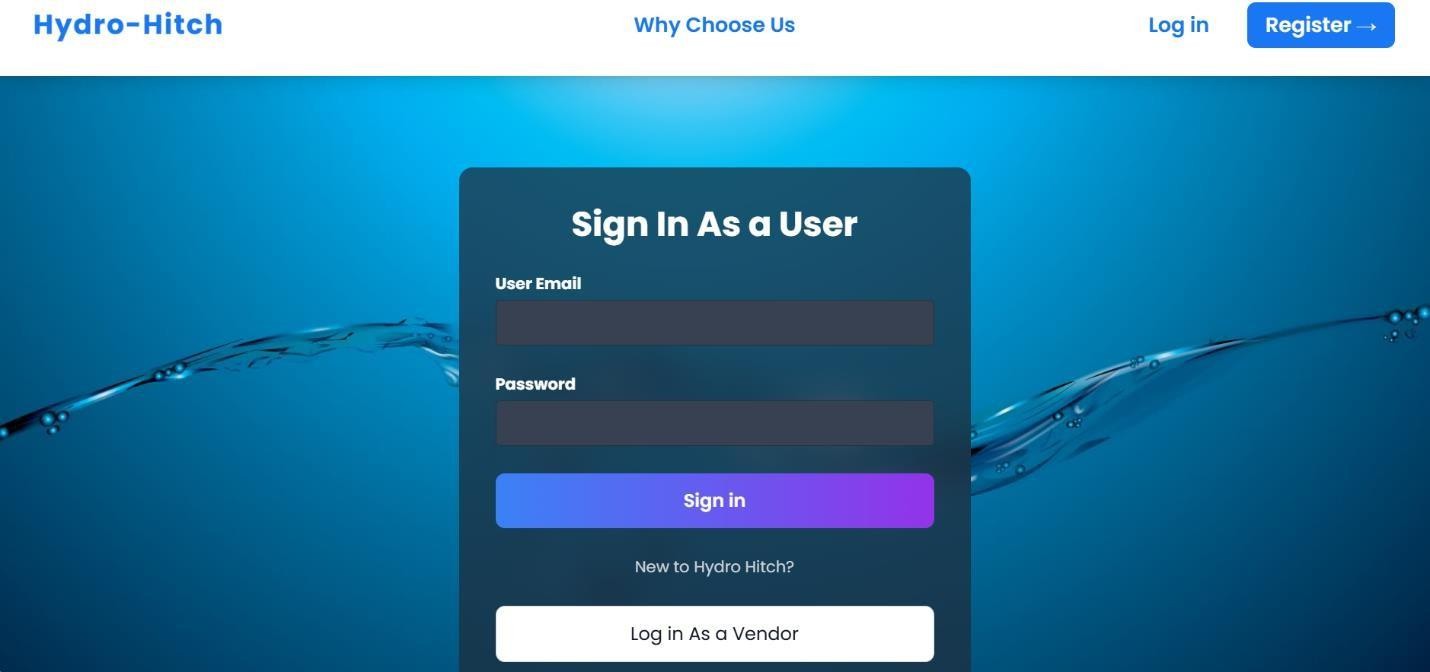
##### Open the app and go to the login screen.

##### Enter your registered email or phone number and your password.

##### Click the "Login" button.

##### The system will validate your credentials.

##### If your credentials are valid, the system will log you in and redirect you to the main dashboard.

****

##### Admin dashboard

##### The admin dashboard provides comprehensive control over the application. As an admin, you can perform the following actions:

##### User and Vendor Management: Add, update, and delete users and vendors as needed.

##### Water Quality Oversight: Review and verify water quality reports submitted by vendors.

##### Query Handling: Address user and vendor queries.

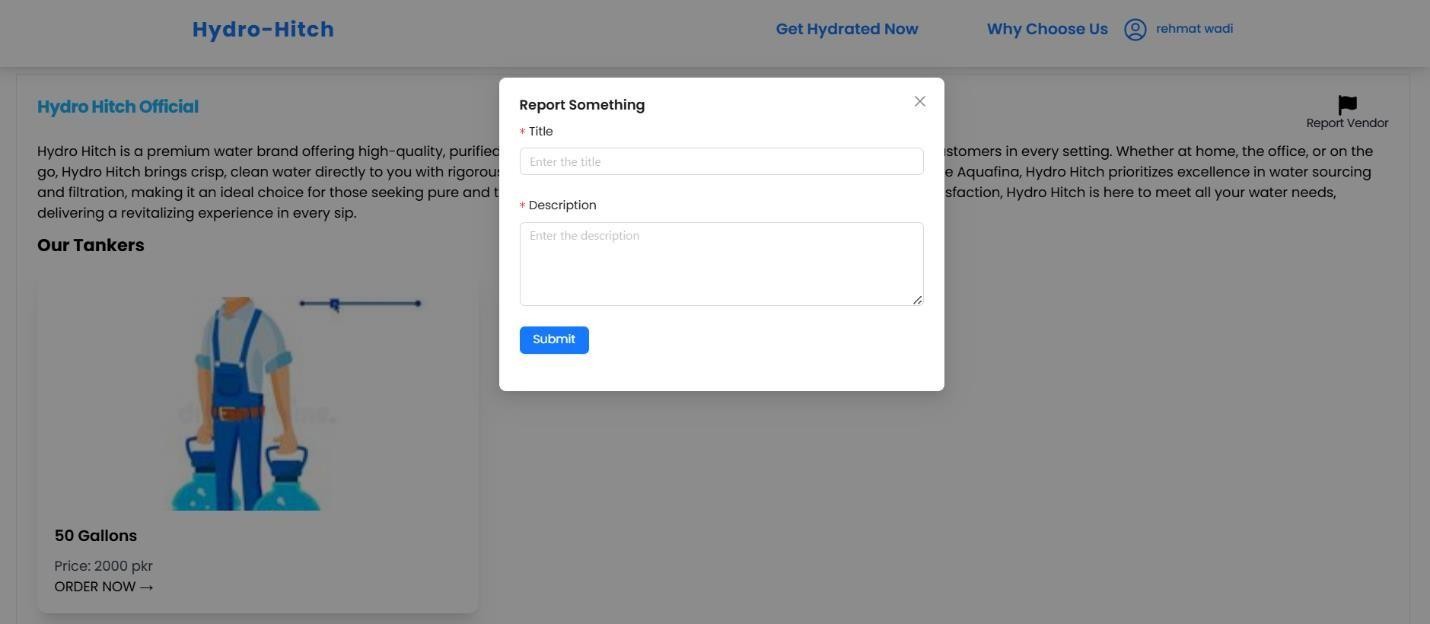
##### Order and Delivery Tracking: Monitor delivery statuses and address any delays.

##### Reporting and Analytics: Generate performance reports on orders, vendor activity, and customer satisfaction.

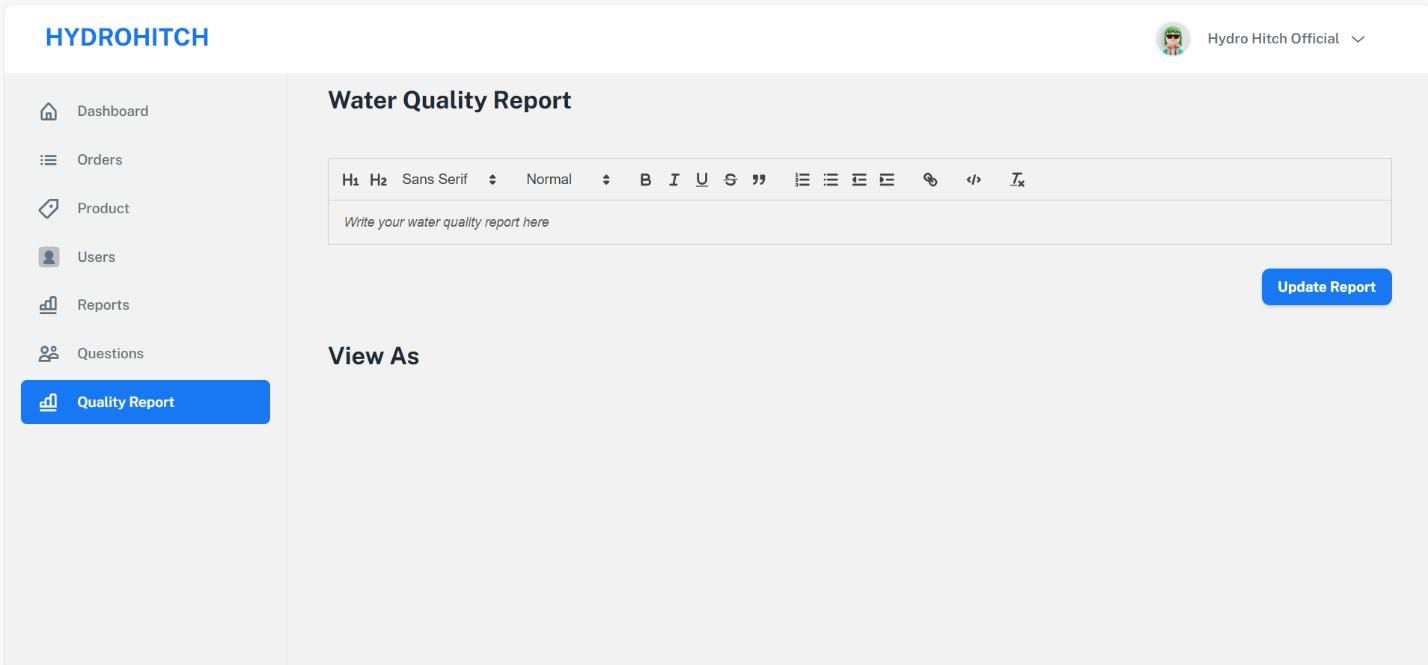
##### 

##### Vendor report

Vendor can do any report if they don’t get what they want or about the product they need to mention title and description .

****

* 1. **Water Quality Report**
* Go to "Quality Report" from the left menu.
* In the text box that says "Write your water quality report here", type or paste your report.
* Once finished, click the blue "Update Report" button to save your report.



##### Feedback

##### Log in to your account.

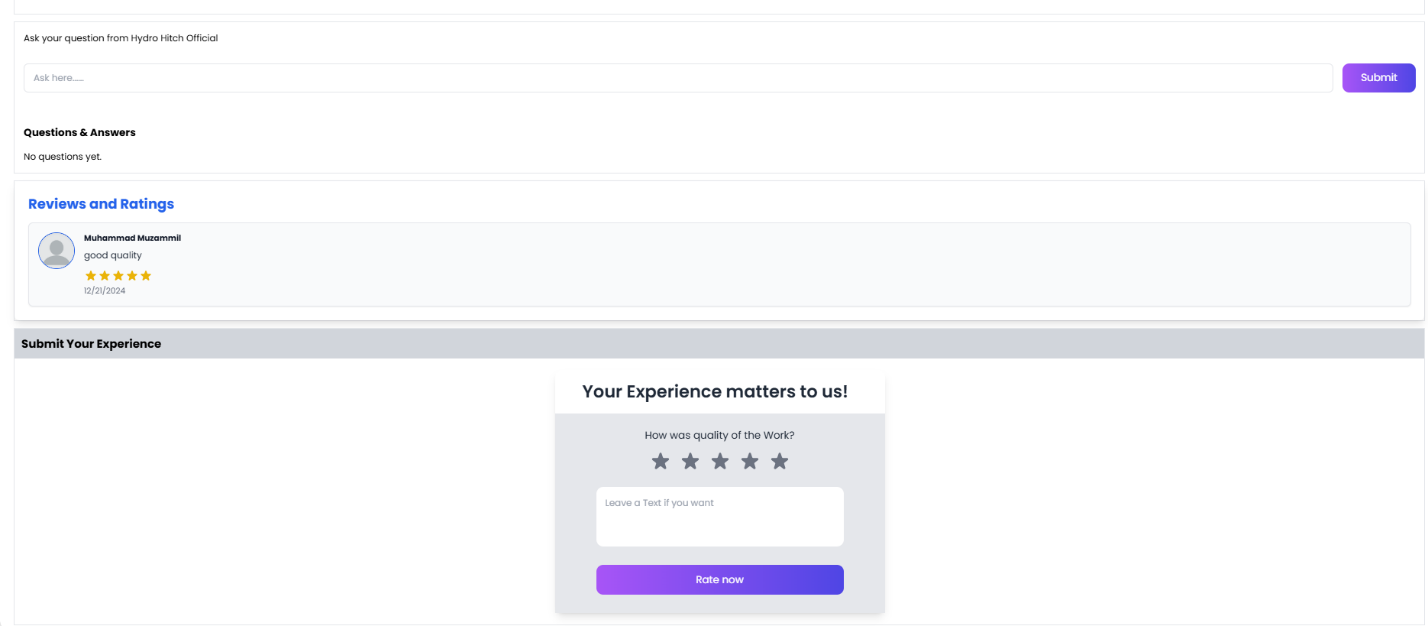
##### Navigate to the feedback section of the dashboard.

##### Enter a comment in the feedback text area.

##### Submit the feedback by clicking the "Submit" button.

##### A confirmation message will appear, and the feedback will be recorded in the database.

##### Admins can view and address this feedback.

****

##### Order booking

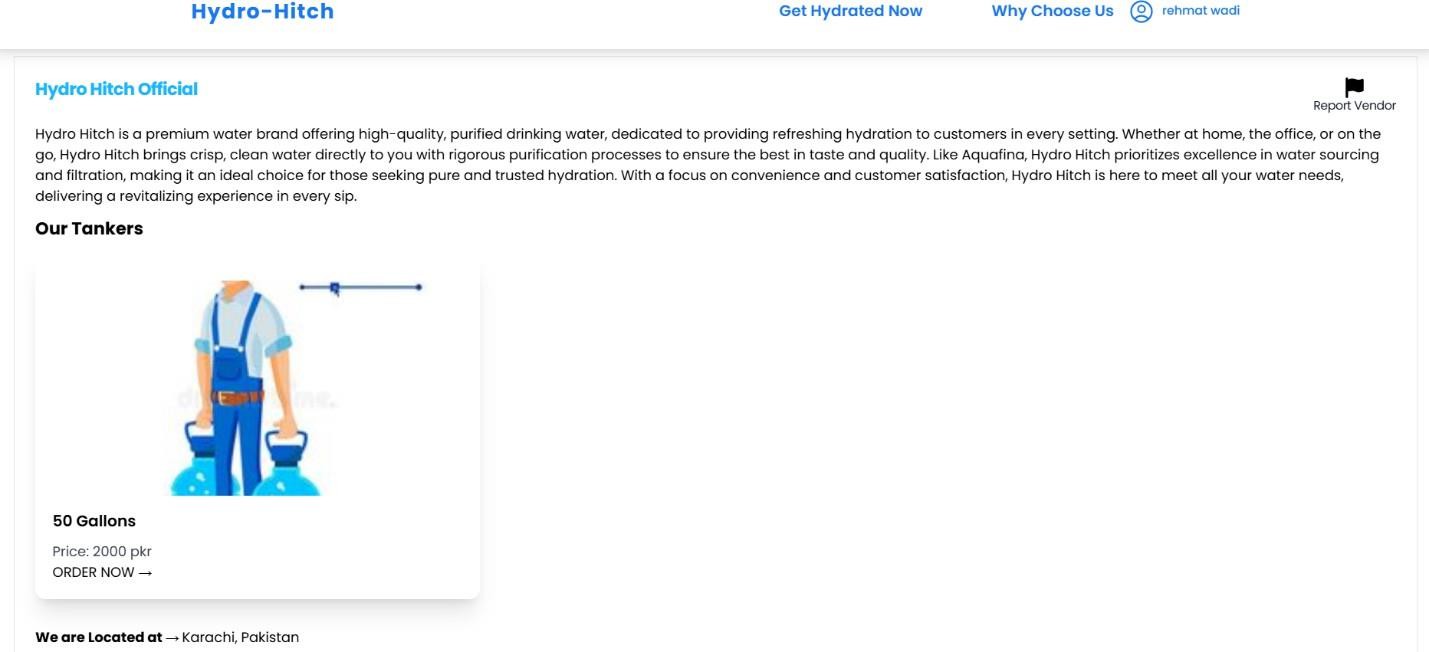
##### Login to the application and navigate to the "Tanker Management" or order section.

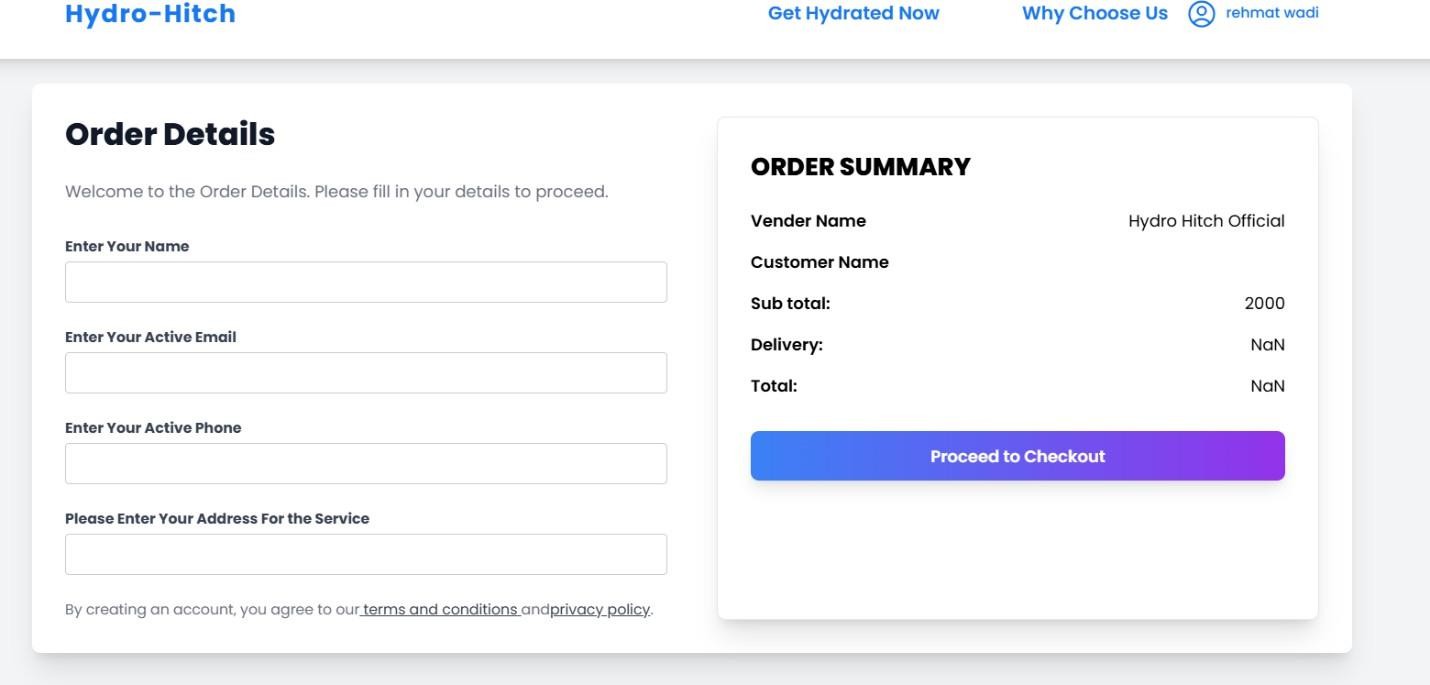
##### View available tankers, pricing, and quality reports.

##### Fill the details

* Select the tanker you want to order

##### Place the order.

****

****

##### User query

##### Log in to the application.

##### Go to the query interface.

##### Submit your query.

##### The system will process the query, retrieve relevant information, and display the response.

##### 

##### Add product

##### The "Add product" feature allows vendors to add a new tanker to their fleet. The process involves the following steps:

##### Log in as a vendor.

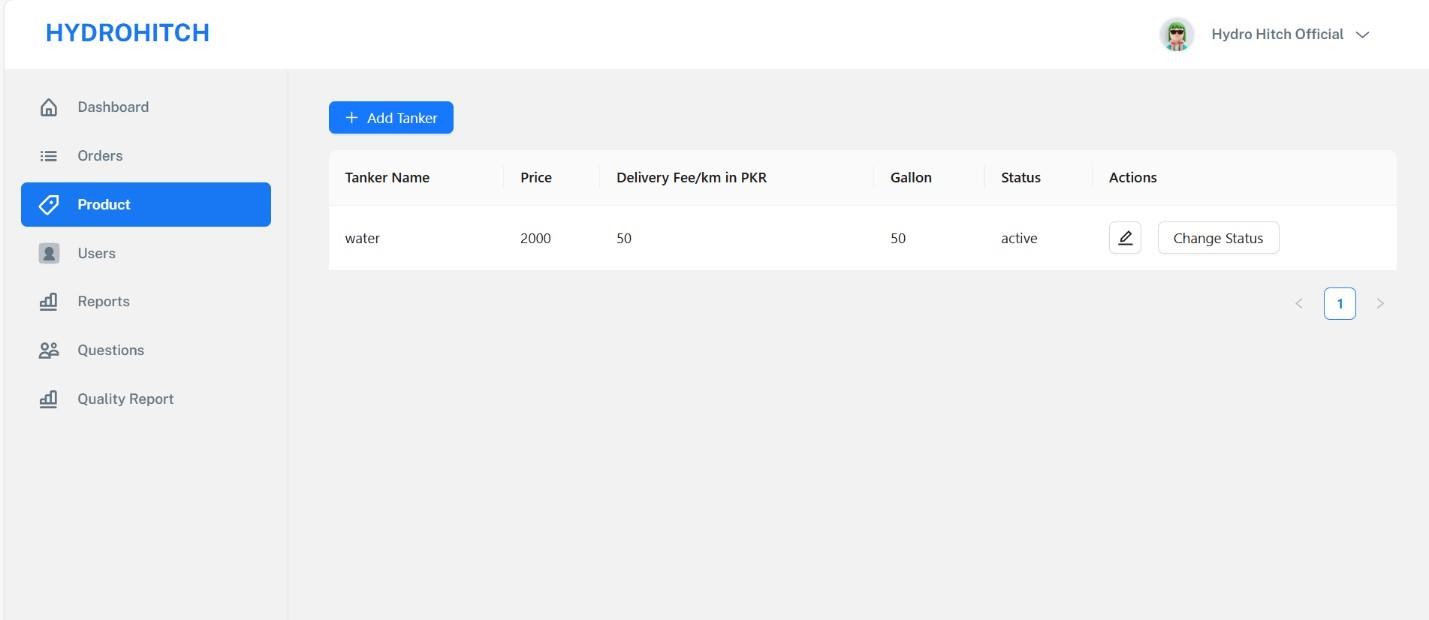
##### Go to the "Manage Tankers" section in the vendor panel.

##### Click the "Add New Tanker" button.

##### Fill in the required details, such as tanker capacity, pricing, and availability.

##### Submit the form to add the new tanker.

##### The application will then display a confirmation message, and the new tanker will appear in the vendor's list of available tankers.

****

##### User reports

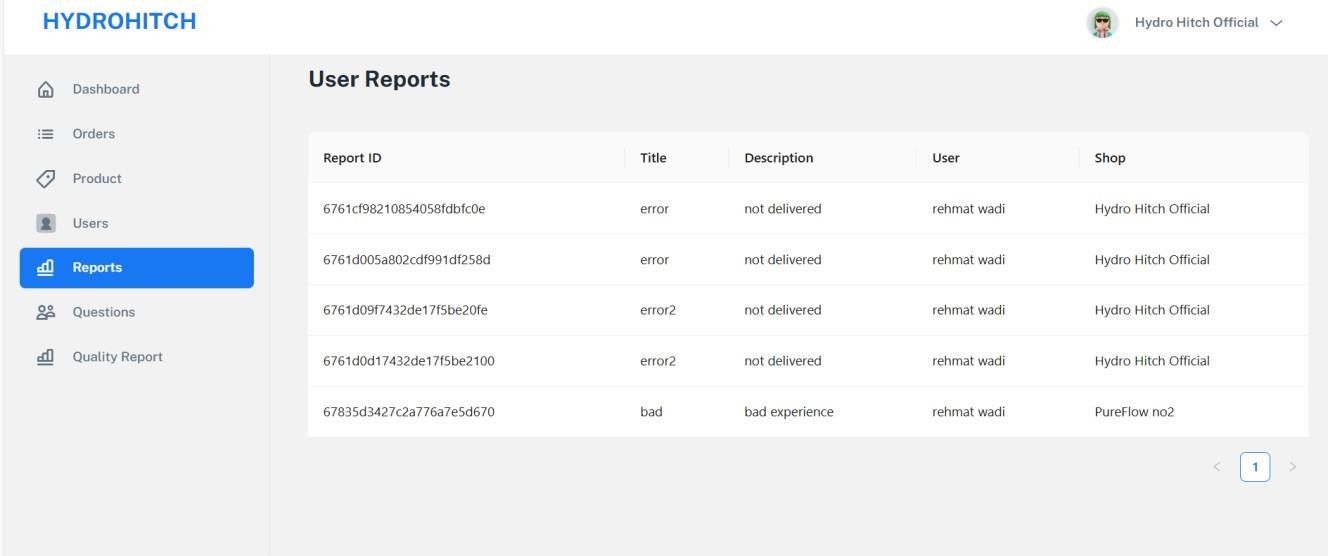
##### As an admin, you can access the user report to view and analyze key metrics. While the document does not provide a specific list of steps, the report is used to track and generate information on:

##### Vendor activity

##### Order performance

##### Customer satisfaction

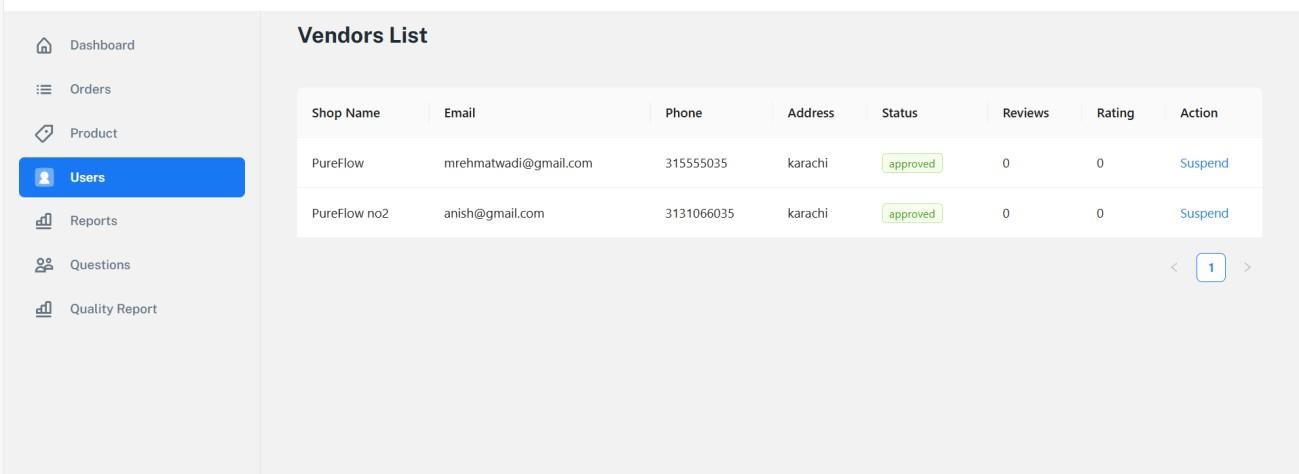
##### Delivery status

****

##### User management

User management is a function of the "Super Admin Panel". It allows an admin to control user and vendor accounts with the following capabilities:

* Add, update, and delete users and vendors as needed.
* Oversee all user and vendor activities.
* Generate reports and analytics on vendor activity

****

##### Order management

##### Log in as a vendor.

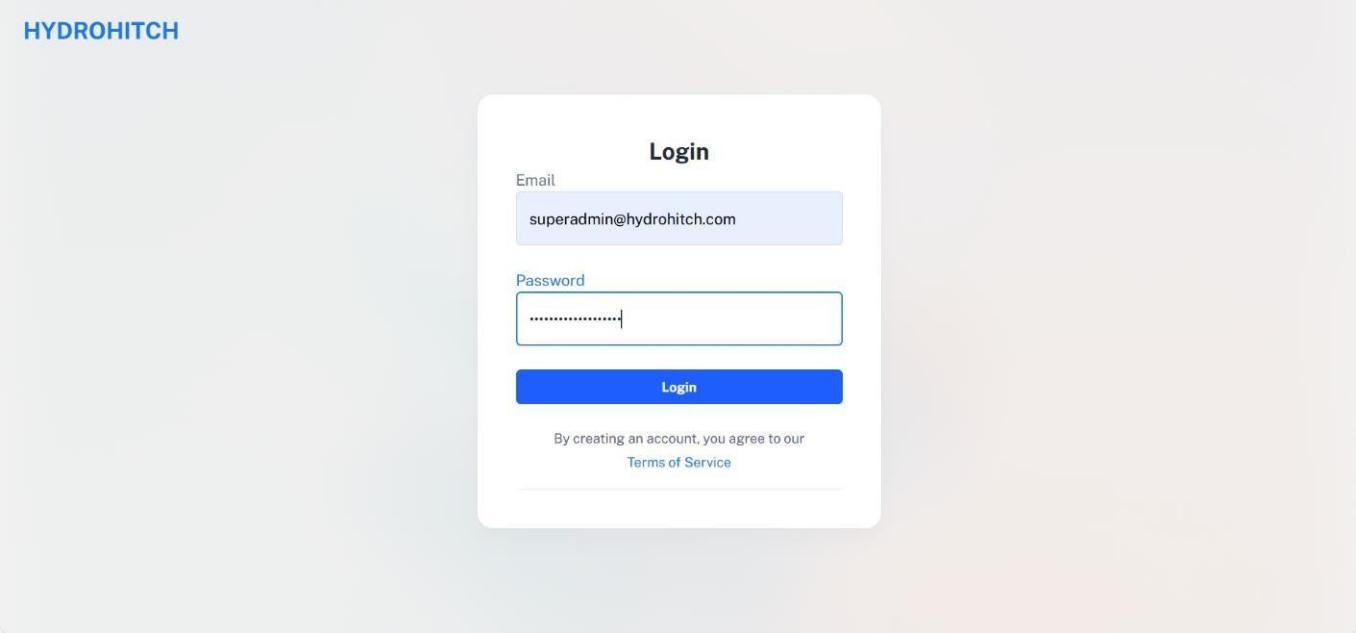
##### Go to the order management section.

##### You can view all orders submitted to you.

##### You can also update the status of an order (e.g., from "pending" to "in progress" or "delivered")

##### Admin Login

Admin Login with their id and password if its match it will navigated to the dashboard

****