Fuel Stock Tracking System

Submitted By

Swarna Rani Das
(201-35-548)
Department of Software Engineering

Supervised By

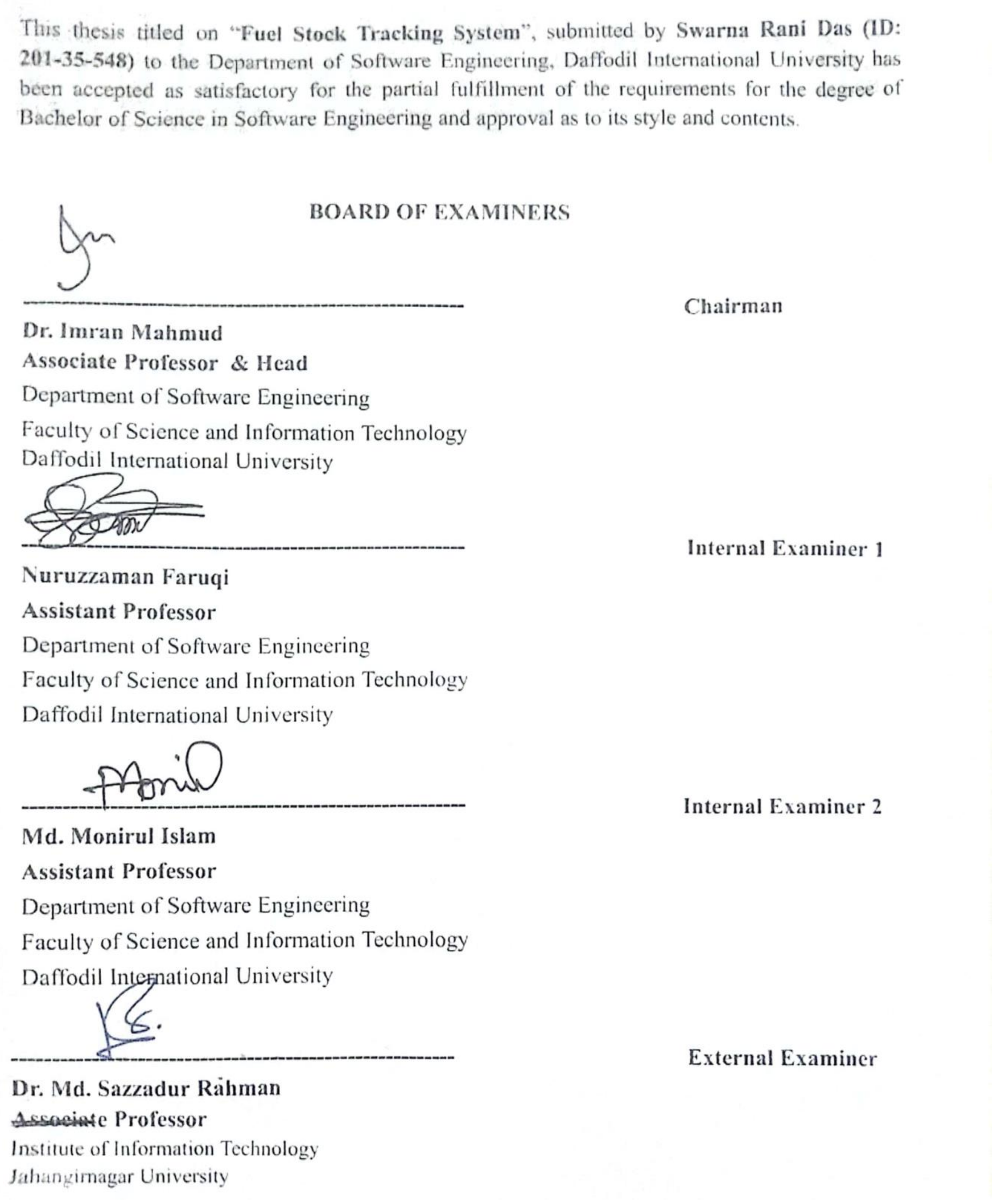
Md. Monirul Islam
Assistant Professor
Department of Software Engineering

A thesis submitted in partial fulfillment of the requirement for the degree of

Bachelor of Science in Software Engineering

Fall 2023

APPROVAL



This thesis titled on **"Fuel Stock Tracking System"**, submitted by **Swarna Rani Das (ID: 201-35-548)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS

[illegible]

-- **Chairman**

Dr. Imran Mahmud
Associate Professor & Head
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

[illegible]

-- **Internal Examiner 1**

Nuruzzaman Faruqi
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

[illegible]

-- **Internal Examiner 2**

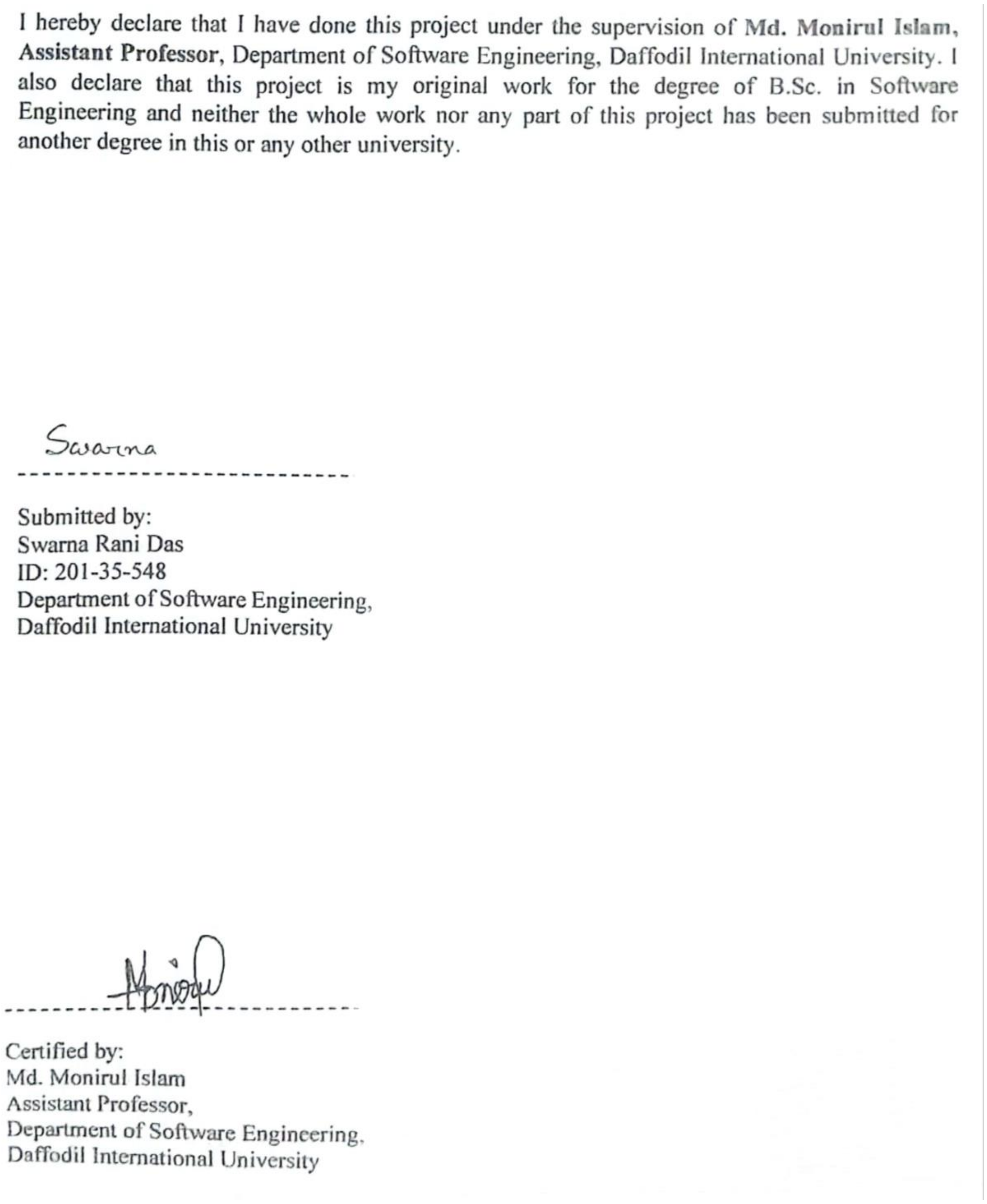
Md. Monirul Islam
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

[illegible]

-- **External Examiner**

Dr. Md. Sazzadur Rahman
~~Associate~~ Professor
Institute of Information Technology
Jahangirnagar University

DECLARATION



I hereby declare that I have done this project under the supervision of **Md. Monirul Islam, Assistant Professor**, Department of Software Engineering, Daffodil International University. I also declare that this project is my original work for the degree of B.Sc. in Software Engineering and neither the whole work nor any part of this project has been submitted for another degree in this or any other university.

Swarna

Submitted by:
Swarna Rani Das
ID: 201-35-548
Department of Software Engineering,
Daffodil International University

[illegible]

Certified by:
Md. Monirul Islam
Assistant Professor,
Department of Software Engineering,
Daffodil International University

ACKNOWLEDGEMENT

My work on this project was intense. But without the kind support of many people, it would not be feasible. Maybe I should say how much I value each of them and I appreciate the help I received from Daffodil International University and the continuous supervision of Mr. Md. Monirul Islam, Assistant Professor, Daffodil International University's Department of Software Engineering, who provided me with the necessary material to grasp the concept and helped me complete my tasks. I would like to thank all of our DIU members and my parents for their cooperation and kind support in getting me through this project.

ABSTRACT

The "Fuel Stock Tracking System" is a cutting-edge solution designed to revolutionize fuel monitoring across industries. Leveraging Node.js, Next.js, React.js, and MongoDB, this system enables real-time tracking and analysis of fuel consumption. It aims to promote responsible usage, optimize fuel efficiency, and reduce costs by implementing personalized insights and seamless integration with existing fleet management systems. The paper details the system's technological framework, hardware setup, and comprehensive testing strategies. Anticipated benefits include enhanced decision-making, reduced operational costs, and sustainable fuel inventory practices, making it a pivotal innovation for industries reliant on fuel resources.

Table of Contents

Chapter 1 Introduction

In the dynamic landscape of industries, the efficient management of fuel stocks is crucial for the smooth operation of various businesses, ranging from transportation and logistics to manufacturing and power generation. The "Fuel Stock Tracking System" emerges as a sophisticated solution to address the challenges associated with monitoring, controlling, and optimizing fuel inventory.

1.1 Project Overview

Fuel Stock Tracking System is a proposed project aimed at developing and implementing a comprehensive fuel tracker system in Bangladesh. This system will address the challenges faced during fuel crises by providing real-time information on fuel availability, queue status at filling stations, and alternative transportation options. By empowering individuals to make informed decisions, the Fuel Stock Tracking System aims to improve fuel management, and real-time updates, reduce wastage, and enhance overall transportation efficiency, time consumption, and tracking.

1.2 Project Purpose

The core purpose of this system is to address the multifaceted challenges inherent in traditional fuel management systems. By automating data collection, analysis, and reporting processes, it aims to eliminate the drawbacks associated with manual monitoring, including human errors, data discrepancies, and delayed reporting. Furthermore, this system intends to empower stakeholders with real-time insights to make informed decisions, reduce fuel-related costs, and enhance operational efficiency.

1.2.1 Proposed System

The impetus behind developing the Online Fuel Management Monitoring System arises from the increasingly complex demands and inefficiencies in fuel management practices. Conventional systems often lack the agility to adapt to dynamic fuel usage patterns, leading to suboptimal resource utilization, increased operational costs, and environmental impact. This system emerges as a response to these challenges, offering an intelligent and adaptable framework to modernize fuel management practices.

The primary objectives of the Online Fuel Management Monitoring System are:

1. **Real-Time Monitoring:** Develop a web-based interface that facilitates real-time tracking of fuel consumption for suppliers, fuel stations, equipment, and machinery.

2. **Enhance fuel management:** Promote responsible fuel consumption by offering personalized fuel consumption monitoring and tips for efficient usage.

3. **Data Analysis:** Implement advanced algorithms to analyze historical fuel consumption data, offering insights into trends and optimization opportunities.

4. **Cost Efficiency:** Provide tools to identify fuel wastage, leading to cost reduction and improved resource allocation.

5. **User-Friendly Interface:** Design a user interface that offers visualizations, customizable reports, and alerts for intuitive data interpretation and provides real-time information on fuel availability, queue lengths

6. **Integration:** Ensure seamless integration with existing fleet management systems for data sharing and enhanced functionality.

1.3 Project Outcome

The purpose of the Fuel Stock Tracking System Web Application project is to design, develop, and implement a comprehensive system that addresses the critical needs of organizations involved in managing fuel resources.

1. **Efficient Fuel Management:** Provide organizations with a centralized platform to record, monitor, and analyze fuel transactions and stock levels efficiently.
2. **Real-time Visibility:** Offer real-time visibility into fuel stock levels, transactions, and historical data, enabling informed decision-making and proactive management.
3. **User-Friendly Interface:** Develop an intuitive and user-friendly web application interface that accommodates users with varying levels of technical expertise.
4. **Accurate Data Recording:** Implement mechanisms to ensure accurate and reliable data recording, minimizing errors in fuel transaction entries and stock updates.

"As I conclude my discussion on Introduction, I will now delve into System Analysis."

Chapter 2 System Analysis

System analysis of a "Fuel Stock Tracking System" involves a detailed examination of the system's requirements, components, and functionalities. The goal is to understand the system's scope, define user needs, and establish a foundation for the system design.

2.1 Feasibility Analysis, Technical Feasibility, Operational Feasibility

An essential first phase in any project's life cycle is feasibility analysis, which establishes the project's viability and value. Technical and operational are the three primary facets of feasibility that should be taken into consideration for a "Fuel Stock Tracking System Web Project."

1. Technical Feasibility:

Technology Requirements:

- Examine the infrastructure currently in place to see if it is capable of supporting the suggested fuel stock tracking system.
- Assess the platforms and technological instruments (such as databases, web servers, and programming languages) that are necessary for development in terms of their compatibility and availability.
- Think about if the team has the necessary technical know-how or if other resources are required.

Development Complexity:

- Evaluate how difficult it will be to construct the system while taking into account all necessary features, points of integration, and probable technological difficulties.
- Evaluate whether integrating the system with currently in-use technologies or systems is feasible.

Scalability and Performance:

- Determine whether the system can process the anticipated number of transactions and data over time.
- Think about scalability possibilities and make sure the system can expand to handle higher traffic.

Security and Compliance:

- Determine the necessary security measures and determine whether putting strong security measures in place is feasible to safeguard confidential fuel stock information.
- Assure adherence to pertinent rules and guidelines in the petroleum sector.

2. Operational Feasibility

Acceptance by Users:

- Participate in the planning and development process with end users to gauge their possible acceptance.
- For system adoption, take into account the design of the user interface, convenience of use, and necessary training.

Management of Change:

- Evaluate how the new system will affect the current workflows and processes.
- Create a change management strategy to facilitate the shift and reduce interference.

Support and Maintenance:

- Determine whether it is feasible to give the system continuous support and maintenance.
- Think about the procedures and resources available for managing updates, problem fixes, and user support.

2.2 Functional Requirements

All of the project's implementable components are referred to as functional requirements. The functional requirements guide the system's real actions.

The functional specifications are provided below-

• User Authentication and Authorization

• Admin Dashboard

• Inventory Management

• Alerts and Notifications

• See Stock of Supplier

• Purchase and Consumption Logging

2.3 System Requirements

Here are the system requirements for a "Fuel Stock Tracking System Web Project," focusing on functional and non-functional aspects:

- The system is built using a specific web framework (React, Node.js).
- A particular database system (MongoDB) is used for data storage.
- The system is integrated with external systems using APIs (fuel suppliers, financial systems).
- Compatibility with various web browsers and devices is ensured.
- The system adheres to relevant industry standards and regulations.
- Compliance with data privacy laws and regulations.

2.4 Non-functional Requirements

Non-functional specifications that specify how the system should function. Additionally, it describes a system's quality characteristics.
The following non-functional requirements are provided-

- Performance
- Quality
- Stability
- Usability
- Response Time
- Reliability

2.5 Performance

Performance is a critical aspect of any web-based system, and the "Fuel Stock Tracking System Web Project" is no exception. Here are key considerations for ensuring optimal performance:

- **Response Time:** Ensure that the system responds promptly to user interactions.
- **Scalability:** Design the system to handle increased loads gracefully.
- **Resource Utilization:** Optimize resource usage to prevent system slowdowns or crashes.
- **Concurrency and Transaction Handling:** Support multiple concurrent users and transactions without performance degradation.
- **Load Testing:** Validate the system's performance under various load conditions.
- **Security Performance:** Ensure that security measures do not significantly impact performance.
- **Database Performance:** Optimize database operations for efficient data retrieval and storage.
- **Monitoring and Logging:** Continuously monitor system performance and identify issues in real time.

"The insights gained from System Analysis set the stage for my exploration of System Design."

Chapter 3 System Design

Designing a Fuel Stock Tracking System involves creating a blueprint for the software, specifying how different components will interact, and ensuring that the system meets the identified requirements.

3.1 Development Model

I decided to build my project using an iterative enhancement method which gives many advantages of this approach. With this idea, a parallel development can be envisaged. All risks will be found and fixed during the iteration. During shorter iterations, testing and troubleshooting will be straightforward. This iteration is ideal for implementation since even the smallest errors can be detected.

This iteration model integrates with the waterfall paradigm and permits multiple iterations throughout the development process.

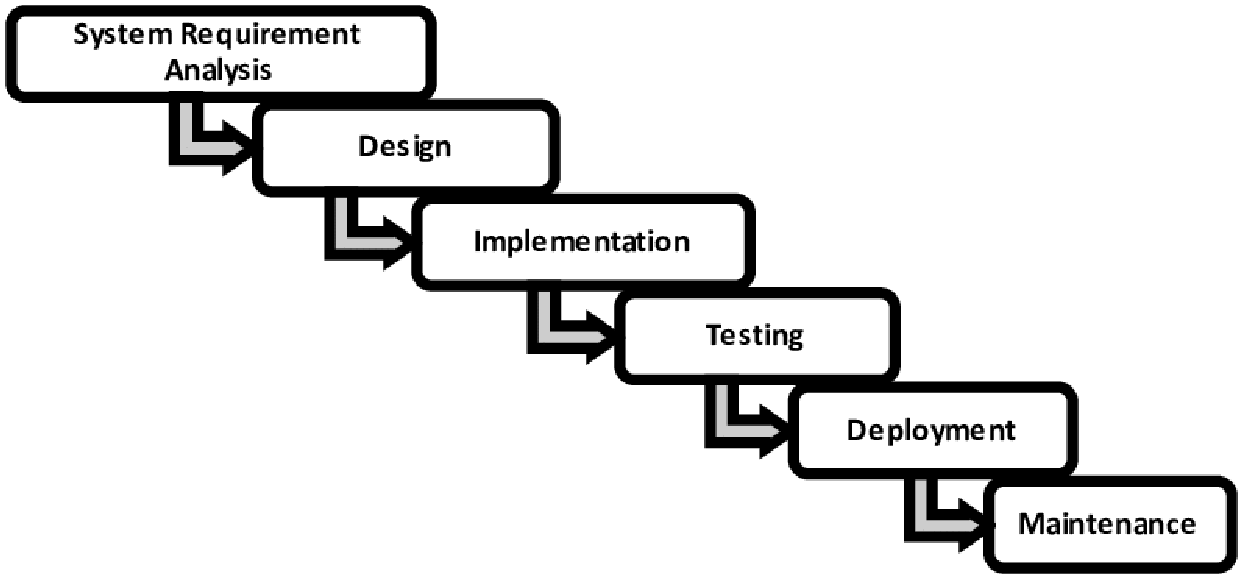


Figure: 3.1 Model of iterative improvement

3.1.1 Use Case Diagram

A use case diagram is a visual representation that illustrates how users interact with a system and the various functionalities of the system. In the context of a Fuel Stock Tracking System, here is a simplified use case diagram:

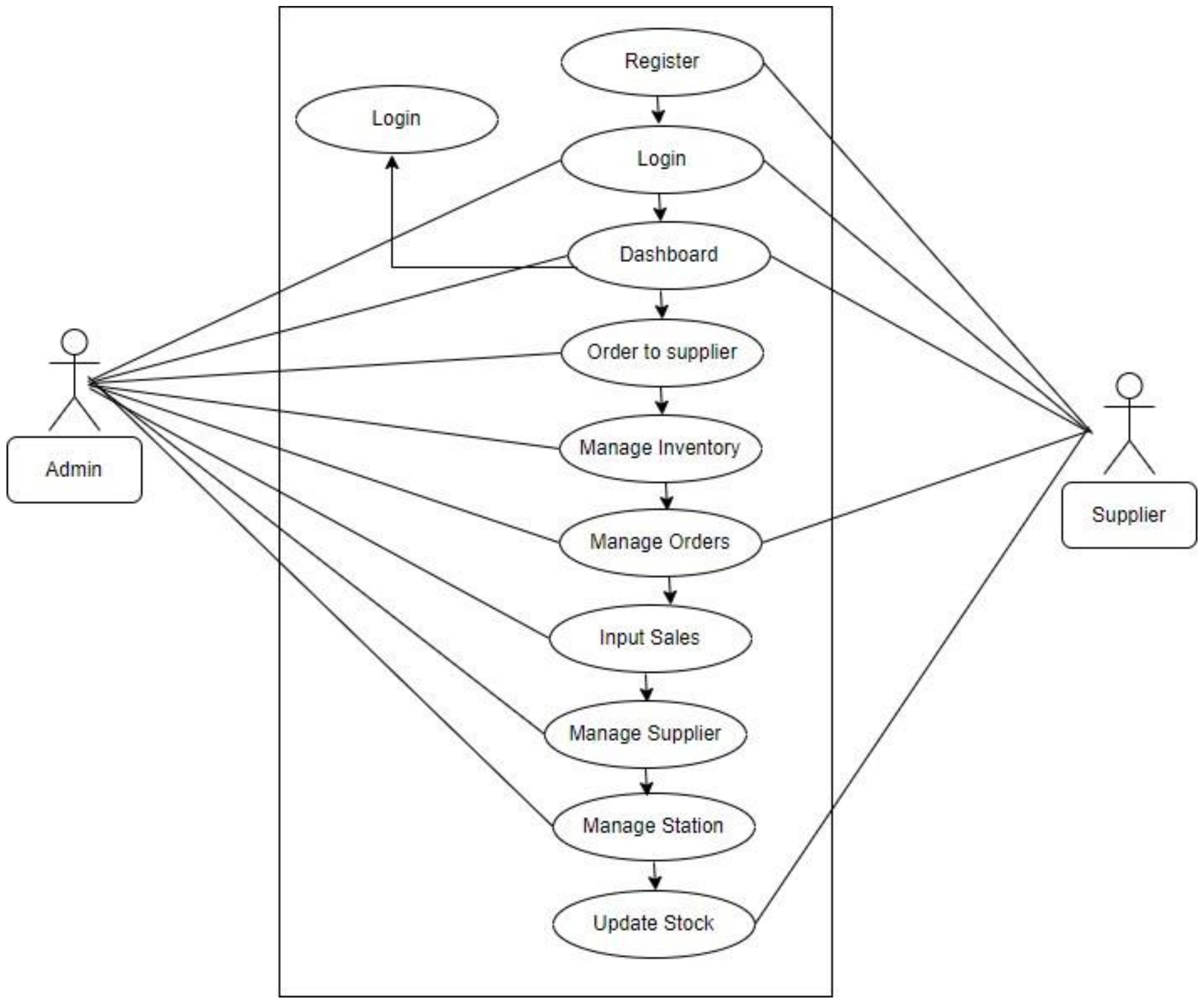


Figure: 3.1.1 Use Case Diagram for Fuel Stock Tracking System

3.1.2 Use Case Descriptions

A use case description is a detailed and textual representation of how a specific interaction or scenario unfolds between a user (or an external system) and a system, typically described within the context of a use case. Let's provide a brief description for each of the use cases mentioned in the Fuel Stock Tracking System use case diagram:

Registration

Description	This module will provide a form. On that form, the user will put their personal information. Then the user can enter into the system. **Actor:** Supplier

Login

Description	By providing user's personal information such as email and password, users can enter into the system. So this login module acts like this. **Actor:** Admin, Supplier

Dashboard

Description	After logging into the system, which interface the user will find the dashboard. Admin will get the admin dashboard and the supplier will get the supplier dashboard. **Actor:** Admin, Supplier

Order to Supplier

Description	Admin or station owner can check the stock of the supplier and according to the stock, admin can place orders to the supplier. **Actor:** Admin

Manage Inventory

Description	The Station owner can manage the inventory throughout the system. It means they can update new stock, delete stock, and everything. **Actor:** Admin

Manage Orders

Description	Admin and station owner both can manage orders. When a supplier gets an order from the admin side, the supplier can confirm or cancel the order. After placing an order admin can also edit the order or cancel it but before the supplier confirms the order. **Actor:** Admin, Supplier

Input Sales

Description	Admin can input sales information into the system. By this time, the stock will update automatically. **Actor:** Admin

Manage Supplier

Description	When suppliers register on the website the admin can approve the registration request. After accepting the supplier, the supplier can enter into the system. Also, the admin can remove the supplier anytime. **Actor:** Admin

Manage Station

Description	This system can handle multiple stations' inventory. So the admin can add or remove stations.

	Actor: Admin

Update Stock

Description	Suppliers can update their available stock for selling into the system. Before placing an order, the admin will check the stock and then place the order. **Admin:** Supplier

3.2 Activity Diagram

An activity diagram is a type of UML (Unified Modeling Language) diagram that illustrates the dynamic aspects of a system by modeling the flow of activities, actions, and transitions between them. It is particularly useful for depicting the workflow within a system or a specific use case.

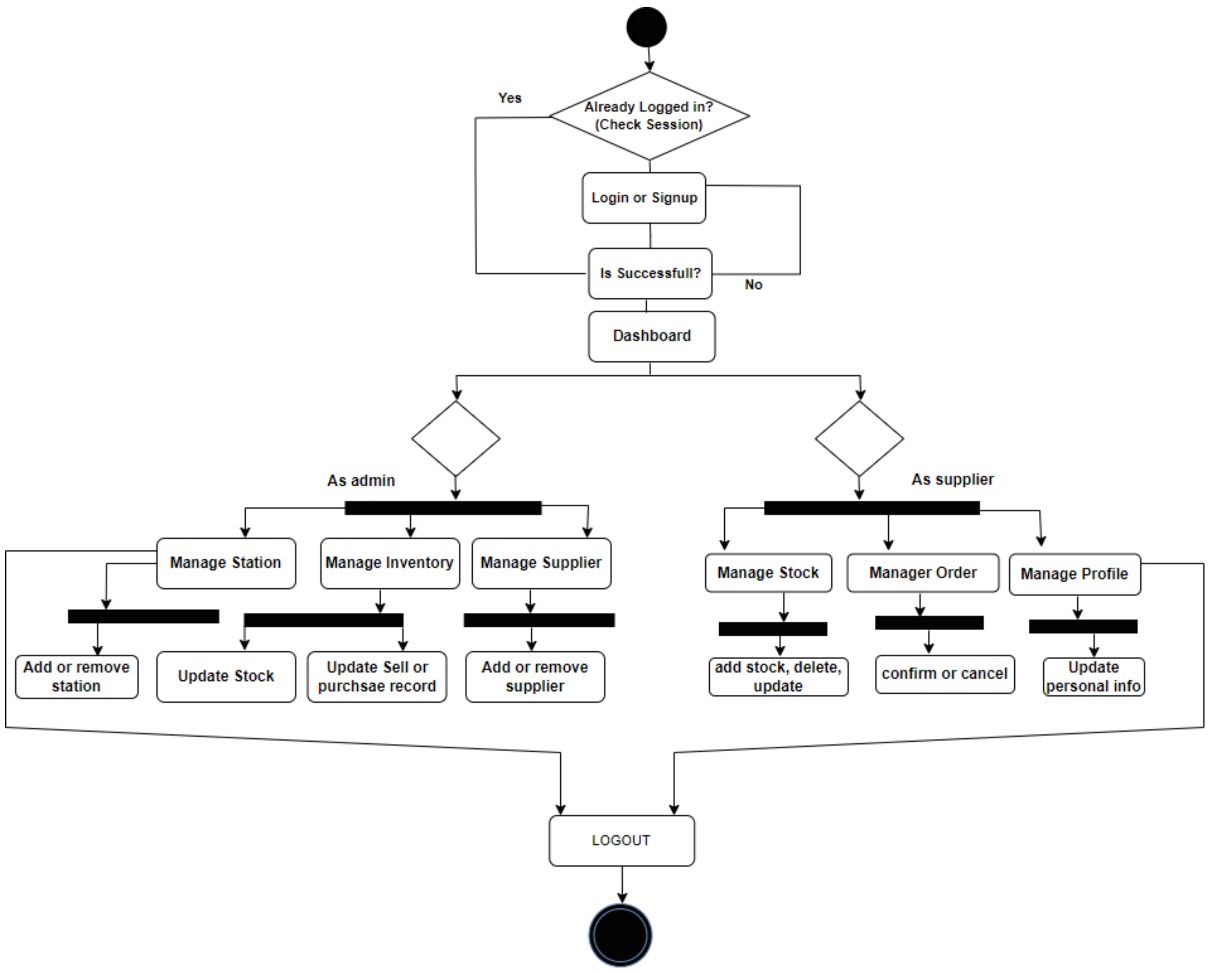


Figure 3.2.1: Activity Diagram for The Entire_Fuel Stock Tracking System

3.3 Sequence Diagram

A sequence diagram is another type of UML (Unified Modeling Language) diagram that illustrates the interactions between different objects or components in a system over time.

Sequence Diagram (Admin)

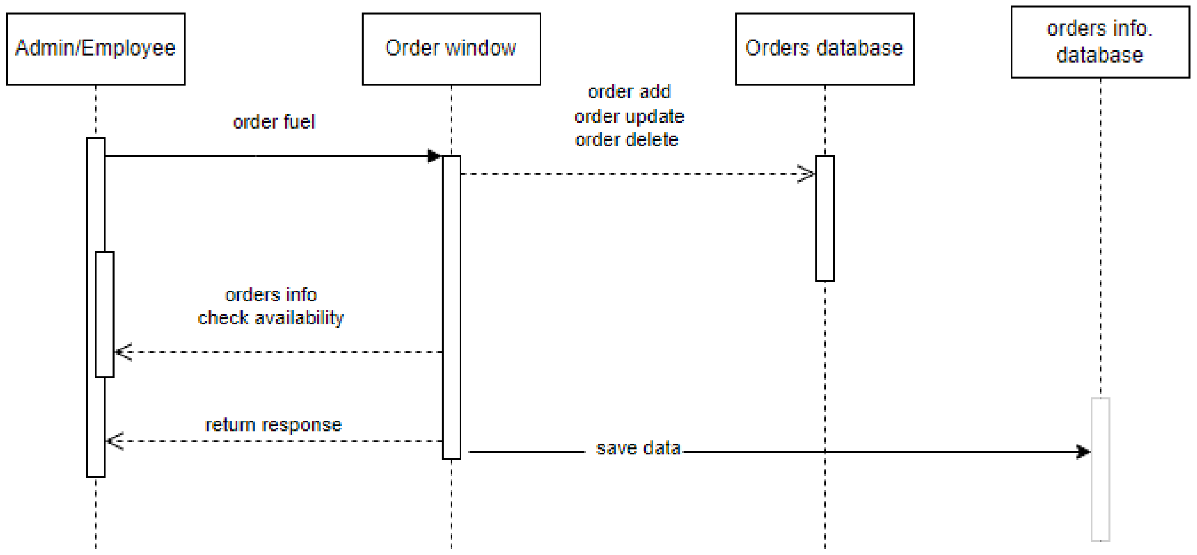


Figure 3.3.1: Sequence Diagram for Admin

Sequence Diagram (Supplier)

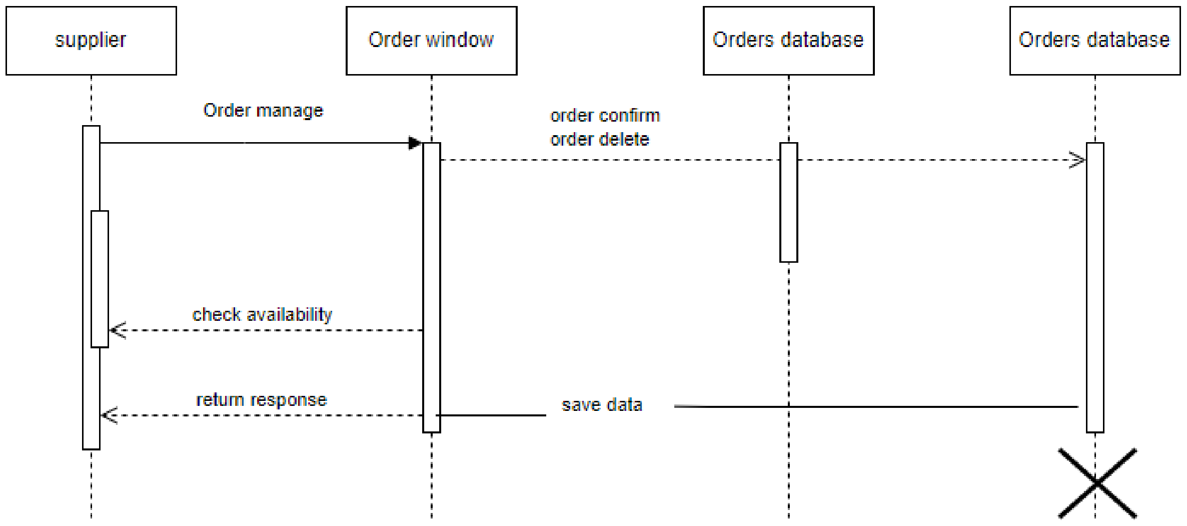


Figure 3.3.2: Sequence Diagram for Supplier

3.4 ERD Diagram

An Entity-Relationship (ER) diagram is a visual representation of the entities (objects or concepts) within a system and the relationships between them. It's commonly used in database design to illustrate the structure of a database and the connections between different entities.

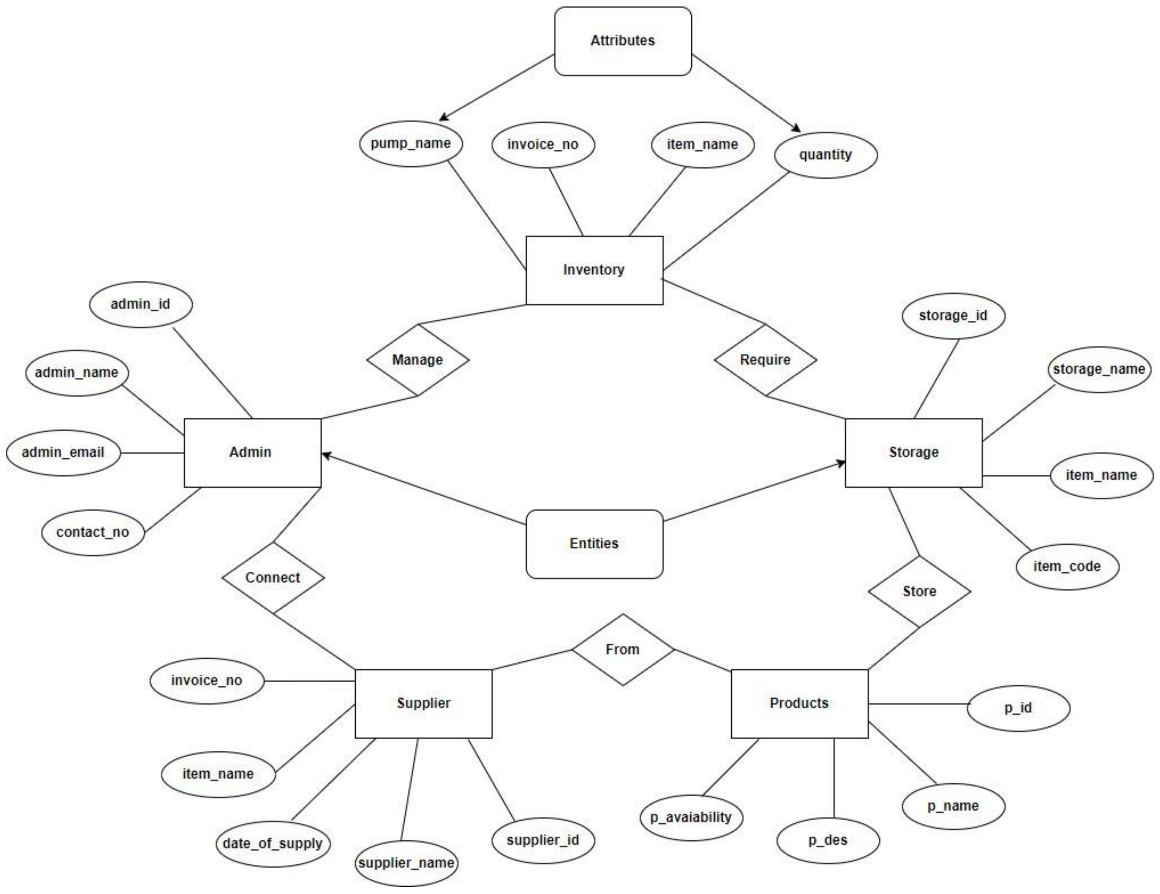


Figure 3.4.1: ER Diagram for The Entire Fuel Stock Tracking System

3.5 Class Diagram

A class diagram is a type of UML (Unified Modeling Language) diagram that represents the static structure of a system by showing the classes, their attributes, methods, and relationships. Here is a simplified class diagram for a Fuel Stock Tracking System:

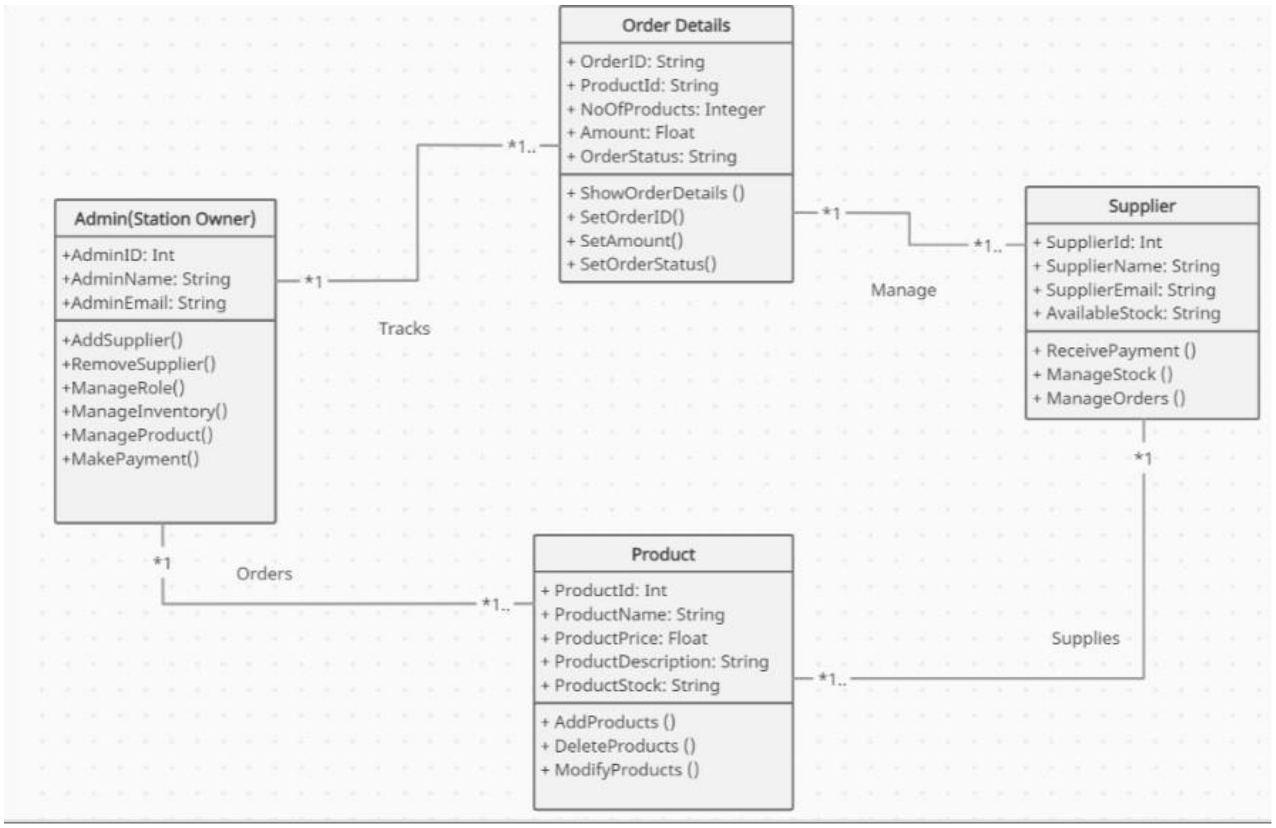


Figure 3.5.1: Class Diagram for The Entire Fuel Stock Tracking System

"Now that I've covered the fundamentals of System Analysis and design. The next chapter holds the answers to which technology I used to build up the system"

Chapter 4 Development Tool & Technology

The choice of development tools and technologies for building a Fuel Stock Tracking System can depend on various factors, including the development team's expertise, the project's requirements, scalability needs, and integration capabilities.

4.1 Integrated Development Environment (IDE)

When developing a web application for a fuel stock tracking system, there are several options for choosing an Integrated Development Environment (IDE) based on preferred programming languages and frameworks. But I choose “Visual Studio Code”.

Visual Studio Code:

- Languages/Frameworks: Suitable for a wide range of languages including JavaScript, TypeScript, HTML, CSS, and various frameworks (Node.js, React, Angular, etc.).
- Features: Lightweight, extensible, and supports a variety of plugins for different languages and frameworks. It has built-in Git support and a robust ecosystem.

4.2 Programming Language

Choosing JavaScript for my fuel stock tracking system web application comes with several advantages, making it a popular and versatile choice for web development. Here are some reasons why I might consider JavaScript:

- **Full-Stack Development:** JavaScript allows us to use the same language for both frontend and backend development. With Node.js, we can execute JavaScript on the server side, providing a unified and consistent development experience.
- **Large Developer Community:** JavaScript has one of the largest and most active developer communities. This means there is a wealth of resources, libraries, and frameworks available, making it easier to find solutions to common problems and stay updated on best practices.
- **Rich Ecosystem of Frameworks and Libraries:** JavaScript has a vast ecosystem of frameworks and libraries that facilitate rapid development. Frameworks like React.js, Angular, and Vue.js are widely used for building dynamic and interactive user interfaces, while Node.js is popular for server-side development.

Frontend Technology: ReactJS

ReactJS is a popular JavaScript library for building user interfaces. It is a component-based library that lets us build complex UIs from small, reusable pieces called components. This makes it a great choice for building dynamic and interactive web applications.

Here are some of the benefits of using ReactJS for front-end development:

- **Component-based architecture:** Makes code more modular and reusable, leading to easier maintenance and development.
- **Virtual DOM**: Provides a performant way to update the UI without re-rendering the entire page.
- Large community and ecosystem: Many libraries and tools are available to help us build React applications.
- **Declarative:** We tell React what we want to see on the screen, and it takes care of updating the DOM.

Backend Framework: NodeJS

Absolutely! Node.js is a fantastic choice for building the backend of my ReactJS application. Here's why:

Node.js and ReactJS: A Perfect Match

- **Shared Language:** Both Node.js and ReactJS use JavaScript, leading to a more cohesive development experience and easier code reuse. We can leverage our JavaScript knowledge across the entire stack.
- **Performance:** Node.js's asynchronous nature and event-driven architecture make it highly performant, especially for real-time applications. This pairs well with ReactJS's virtual DOM, leading to a smooth and responsive user experience.
- **Scalability:** Both technologies are known for their scalability. Node.js can handle high traffic volumes efficiently, while ReactJS can handle complex UIs without performance bottlenecks.
- **Large Community and Ecosystem:** Both Node.js and ReactJS have massive communities and ecosystems. We'll find numerous libraries, tools, and frameworks to support our development process.

4.3 User Interface Design

A well-designed user interface (UI) is crucial for any fuel stock tracking system. It should be intuitive, and efficient, and provide clear insights into fuel levels and usage. Here are some UI design ideas to consider:

Dashboard:

- **Fuel levels visualization:** Prominently display current fuel levels in tanks with gauges, charts, or clear numerical values. Use color-coding for easy identification of critical levels.
- **Consumption trends:** Show graphs and charts depicting fuel usage over time. Identify periods of peak consumption and potential areas for optimization.
- **Delivery and refill information:** Log delivery dates, quantities, and supplier details for each tank. Track fuel received and consumed to ensure accurate inventory management.
- **Alerts and notifications:** Set up automated alerts for low fuel levels, discrepancies, or unusual consumption patterns.
- **Real-time data monitoring:** Display live updates on fuel levels and sensor readings. Allow users to remotely monitor tanks and identify potential issues.

Additional Features:

- **Reporting and analytics:** Generate reports on fuel consumption, deliveries, and costs. Export data for further analysis and decision-making.
- **User roles and permissions:** Implement access controls based on user roles and responsibilities. Ensure data security and prevent unauthorized actions.
- **Mobile app integration:** Develop a mobile app for convenient access to fuel stock data and functionalities on the go.

Design Principles:

- **Simplicity and clarity:** Prioritize intuitive design elements and avoid unnecessary complexity. Use clear labels, icons, and consistent layouts.
- **Data visualization:** Leverage charts, graphs, and color-coding to effectively communicate fuel levels and trends. Make data easily interpretable for users of all technical backgrounds.

- **Responsiveness and accessibility:** Ensure the UI adapts seamlessly to different screen sizes and devices. Make it accessible for users with disabilities.
- **User-friendliness:** Conduct usability testing to ensure the UI is intuitive and efficient for users to navigate and interact with.

4.4 Database

Database: MongoDB

Since I'm building my application with ReactJS on the front end and Node.js on the back end, MongoDB emerges as a fantastic choice for a database. Here's why:

MongoDB's Strengths for the Application:

- **Document-oriented:** Stores data in flexible JSON-like documents, perfectly aligned with the object-oriented nature of JavaScript used in ReactJS and Node.js.
- **Scalability:** Handles massive data volumes and high traffic efficiently, ideal for future growth.
- **Performance:** Offers fast querying and data manipulation thanks to its flexible schema and indexing capabilities.
- **Rich query language:** Supports complex queries using operators and aggregation frameworks, enabling powerful data analysis.
- **Cloud-ready**: Deploys easily on cloud platforms like MongoDB Atlas, simplifying deployment and management.

4.5 Deploy and hosting

Deploying and hosting our fuel stock tracking system website with JavaScript involves several steps:

1. Choose a hosting platform:

- Cloud hosting: Scalable and flexible, ideal for high-traffic websites with varying demands. Popular options include Heroku, AWS, and Google Cloud Platform.

2. Prepare website for deployment:

- Build your front-end with ReactJS: Ensure the code is clean, optimized, and production-ready. Consider using tools like Webpack for bundling and minifying our code.
- Set up your Node.js backend: Ensure the server code is stable and secure. Use frameworks like Express.js for building APIs and handling data interactions.
- Connect to your database: Configure our database connection in Node.js code. Choose a secure and reliable hosting provider for the MongoDB database.

3. Deploy your website:

- VPS or Cloud hosting: Follow the specific instructions provided by the hosting provider.

4. Secure your website:

- Implement secure authentication and authorization: Use mechanisms like logins, user roles, and API keys to restrict access to sensitive data.
- Use HTTPS and encryption: Protect data transmission between websites and users' browsers.
- Regularly update software and dependencies: Keep website code, server software, and libraries up to date to patch vulnerabilities.

5. Monitor and maintain website:

- Track website traffic and performance: Use tools like Google Analytics to monitor user behavior and identify potential issues.
- Backup your data regularly: Protect against data loss due to server failures or cyberattacks.

“As I conclude my discussion on Development Tools and Technology, the next chapter promises a deep dive into the realm of testing”

Chapter 5 System Testing

System testing is a phase in the software testing life cycle where the entire integrated software system is tested to ensure that it meets specified requirements. The goal of system testing is to evaluate the system's compliance with functional and non-functional requirements, identify defects, and ensure that it performs as expected in its intended environment.

5.1 Testing Features

Testing plays a crucial role in ensuring the functionality, performance, and user experience of the fuel stock tracking system.

5.1.1 Feature to be tested

Choosing the features to test for the fuel stock tracking system will depend on the specific system's functionality and priorities. However, here are some key features that should be considered for testing:

Features	**Priority**	**Description**
Login and authentication	1	Verify user login, role-based access control, and session management
Navigation and dashboard:	1	Test ease of navigation, clarity of information, and intuitiveness of controls.
Visualization and charts	3	Ensure fuel level trends, consumption patterns, and alerts are displayed clearly and accurately.
Responsiveness	2	Test the website's functionality and layout on different devices and screen sizes.

Data entry	1	Test adding, editing, and deleting fuel level data for different tanks.
Sensor data integration	3	Ensure live sensor readings update accurately on the dashboard.
Calculations	1	Verify calculations for fuel consumption, remaining stock, and refill quantities.
Notifications and alerts	1	Test the delivery of low fuel level and consumption anomaly alerts to users.
Contact	1	Admin can manage contact from the user contact section
Security measures	1	Test login security, data encryption, and access controls to prevent unauthorized access.
Logout	1	After logging out, the session needs to be terminated.

5.2 Testing Strategies

5.2.1 Test Approach

Two types of testing have been utilized to guarantee the system's quality. It primarily focuses on White Box and Black Box testing.

- Functional testing, another name is black box testing. It is a testing approach that concentrates solely on the output, ignoring internal procedures. Some outputs are produced for a given input. These results are then compared to the anticipated result. The function is accepted if they match.

- Structural testing, another name is White Box Testing. This testing technique takes into account the system's internal mechanism.

5.2.2 Pass/Fail Criteria

Data Management and Accuracy:

- Pass:
  - All data entries (add, edit, delete) are successfully processed and reflected in the system.
  - Live sensor readings update accurately on the dashboard within a specified timeframe (e.g., 5 seconds).
  - Calculations for fuel consumption, remaining stock, and refill quantities are accurate within a defined tolerance (e.g., 1% error).
- Fail:
  - Data entries fail to save or are not reflected correctly.
  - Live sensor readings have lag or display inconsistent values.
  - Calculations have significant discrepancies exceeding the tolerance limit.

User Interface and Experience (UI/UX):

- Pass:
  - Login and authentication function correctly, with clear error messages for invalid attempts.
  - Navigation is intuitive and user-friendly across different sections of the dashboard.
  - Information on the dashboard is clear, concise, and easy to understand.
  - Visualizations and charts display data accurately and effectively.
- Fail:
  - The login process is buggy or error messages are unclear.
  - Navigation is confusing or difficult to follow.
  - Information on the dashboard is overwhelming, cluttered, or ambiguous.
  - Visualizations and charts are misleading or difficult to interpret.
  - The website layout breaks or functions poorly on certain devices or screen sizes.

System Functionality and Security:

- Pass:
    - API interactions are successful and data is retrieved, updated, and authorized correctly.
    - Error handling mechanisms provide informative messages and prevent crashes.
    - Security measures effectively prevent unauthorized access and data breaches.
    - Backup and restore functionalities work seamlessly and ensure data recovery.
- Fail:
    - API interactions fail or return inaccurate data.
    - Error handling messages are missing or unhelpful.
    - Security vulnerabilities allow unauthorized access or data leaks.
    - Backup and restore processes fail or result in data loss.

Additional Features:

- Pass:
    - Notifications and alerts are delivered promptly and accurately for low fuel levels and consumption anomalies.
    - Mobile app integration functions seamlessly with the website and offers a similar user experience.
    - Integrations with other systems work as expected and exchange data accurately.
- Fail:
    - Notifications and alerts are delayed, inaccurate, or not delivered at all.
    - Mobile app integration is buggy or has significant functionality limitations.
    - Integrations with other systems fail or cause data inconsistencies.

5.2.3 Testing Schedule

Test Phase	Time

Testing Plan Create	1 week
Unit testing	During development time
Component Test	During development time
Testing user interface	1 week
Performance testing Accessibility Testing	1 week
Accessibility Testing	1 week

5.3 Test Cases

Test Case ID	**Test Case Description**	**Test Steps**	**Test Data**	**Expected Results**	**Actual Results**	**Pass/Fail**
1	Check user Registration with valid Data	Go to Site Enter Name Enter Email Enter Password Enter Contact Click Submit	User-Name = Name User-Email = Email User-Password = Password User-Contact	Users should Register for the application	As expected,	Pass

2	Check user Login with valid Data	Go to Site Enter Email Enter Password Click Submit	User-Email = Email User-Password = Password	User should log into the applicati on	As expected,	Pass
3	Add Fuel Level	Go to Site Click on Add Stock Select Fuel Type Select Pump Enter Stock Submit	Fuel-Type = Fuel Pump-Name = Pump Fuel-Stock = Stock	Enter a valid fuel level for a specific tank and verify it's added correctly	As expected,	Pass

4	Edit Fuel Stock	Go to Site Click on Edit Stock Rewrite Stock Submit	Fuel-Stock = Stock	Edit an existing fuel level entry and verify if the changes are reflected correctly	As expected,	Pass

"In wrapping up my discussion on testing methodologies, I pave the way for the next chapter dedicated to user guidance."

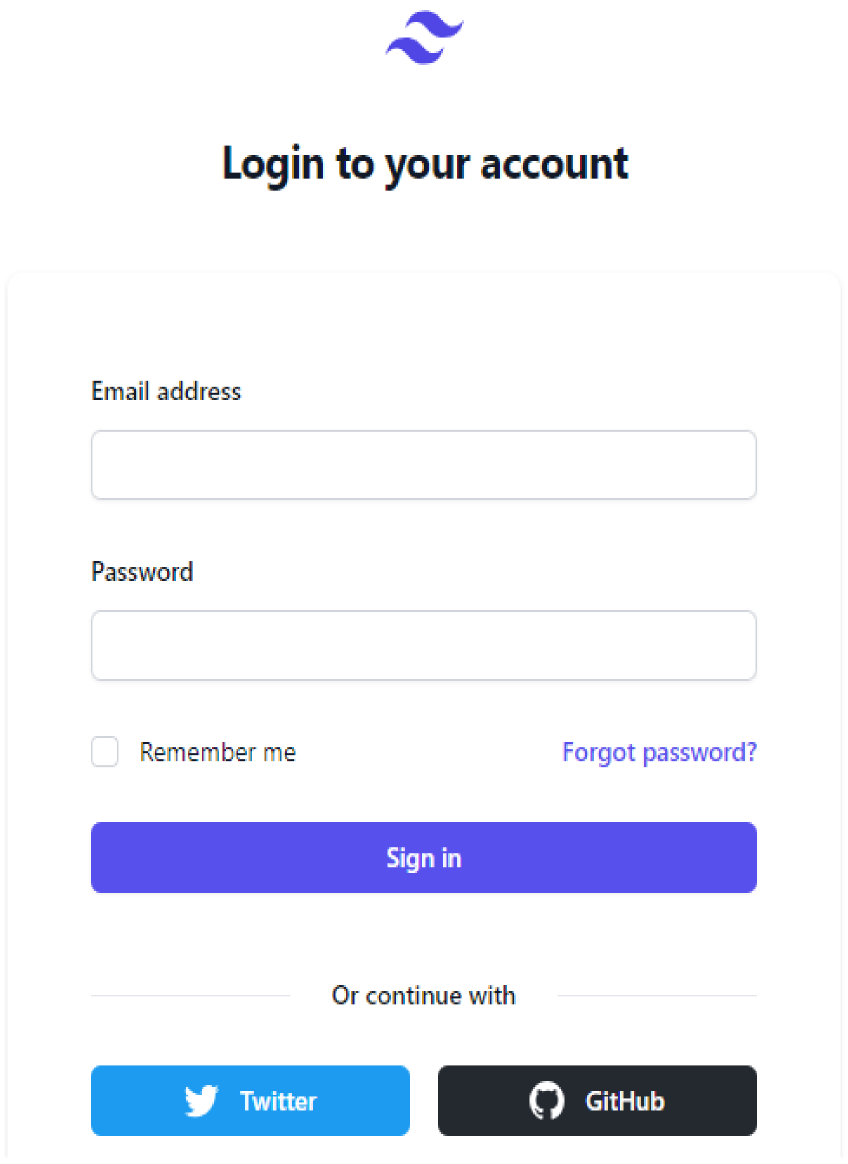
Chapter 6 User Manual

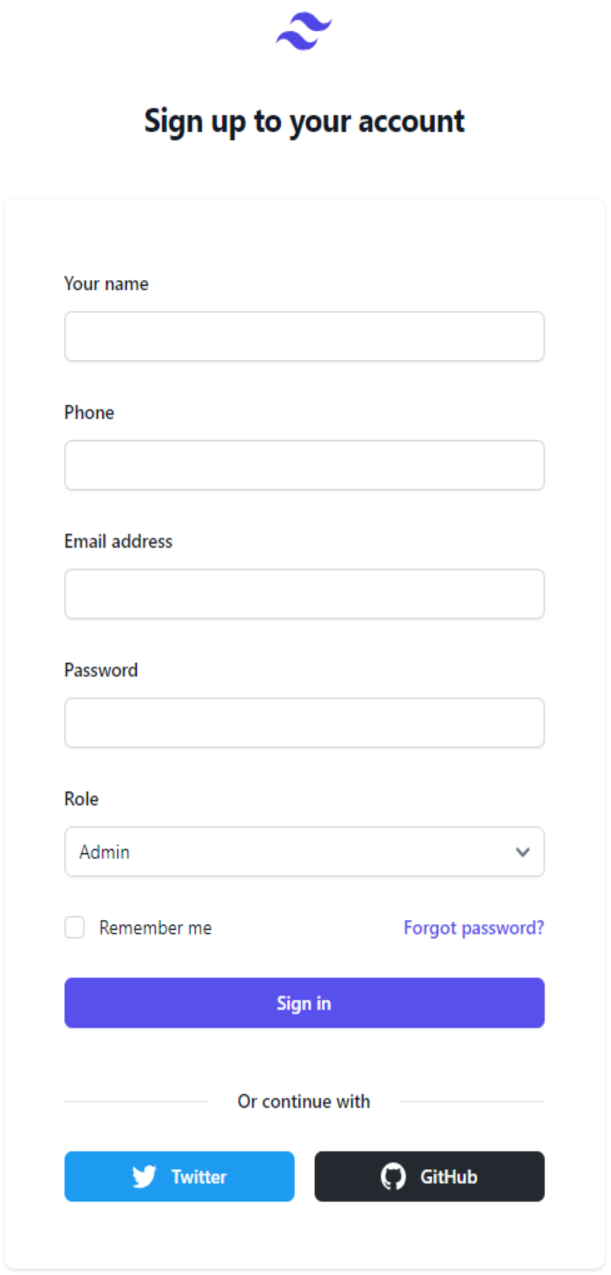
Creating a comprehensive user manual is essential to help users understand how to use a system efficiently and effectively. Below is a general outline for creating a user manual for a Fuel Stock Tracking System.

6.1 Landing Page

Registration & Login

First of all, we have to register from this page, with name email, and new password, and then log in





Sign up to your account

Your name

Phone

Email address

Password

Role

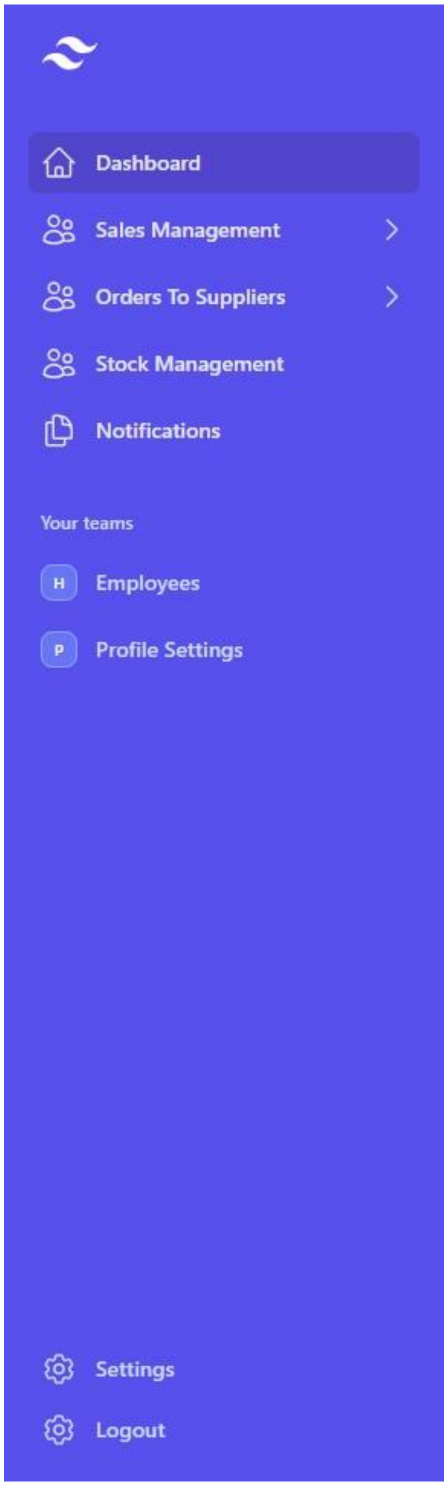
Admin

Remember me

Forgot password?

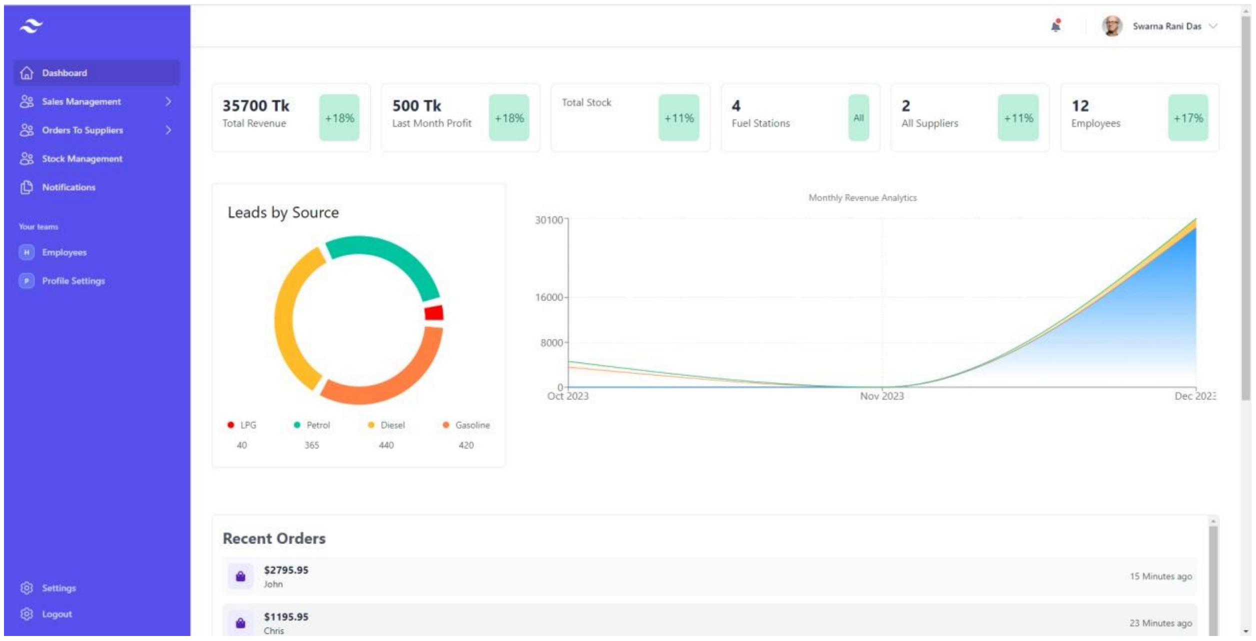
Side Navigation Menu

This is the sidebar menu of the system which contains the primary content of this system.



Admin Dashboard

This is the admin Dashboard, from where the admin can handle the whole system.



35700 Tk Total Revenue +18%

500 Tk Last Month Profit +18%

Total Stock +11%

4 Fuel Stations All

2 All Suppliers +11%

12 Employees +17%

LPG	Petrol	Diesel	Gasoline
40	365	440	420

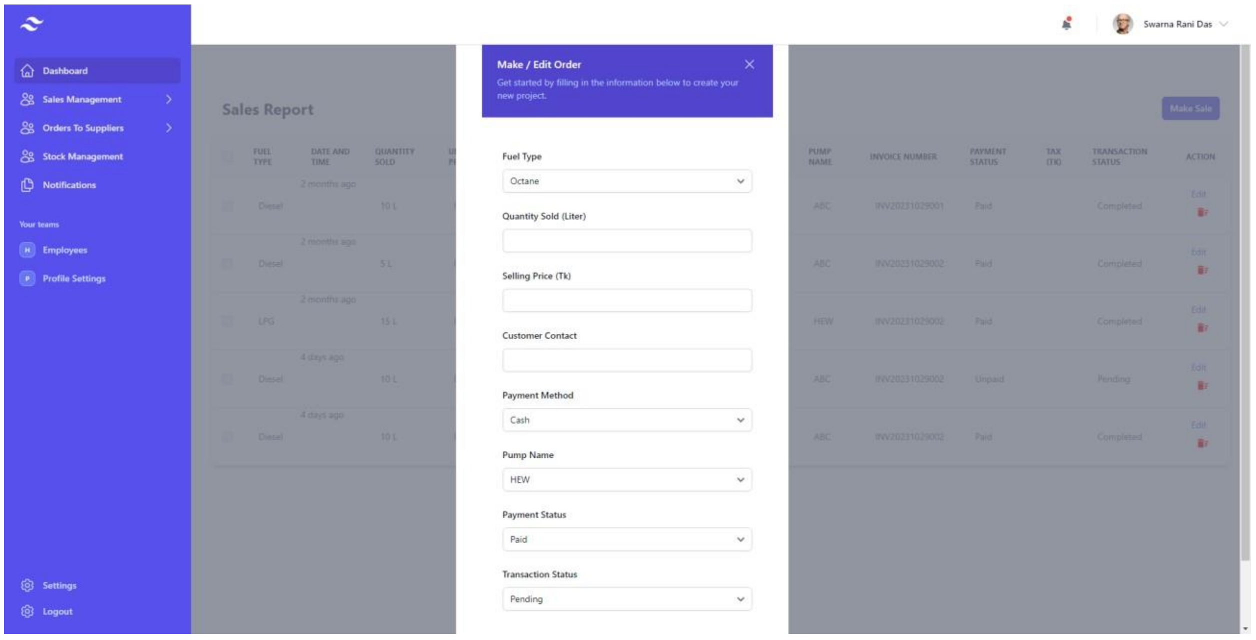
Recent Orders

$2795.95 John — 15 Minutes ago

$1195.95 Chris — 23 Minutes ago

Sale data Update

Admin can update the selling data from there.



Make / Edit Order

Get started by filling in the information below to create your new project.

Fuel Type: Octane

Quantity Sold (Liter):

Selling Price (Tk):

Customer Contact:

Payment Method: Cash

Pump Name: HEW

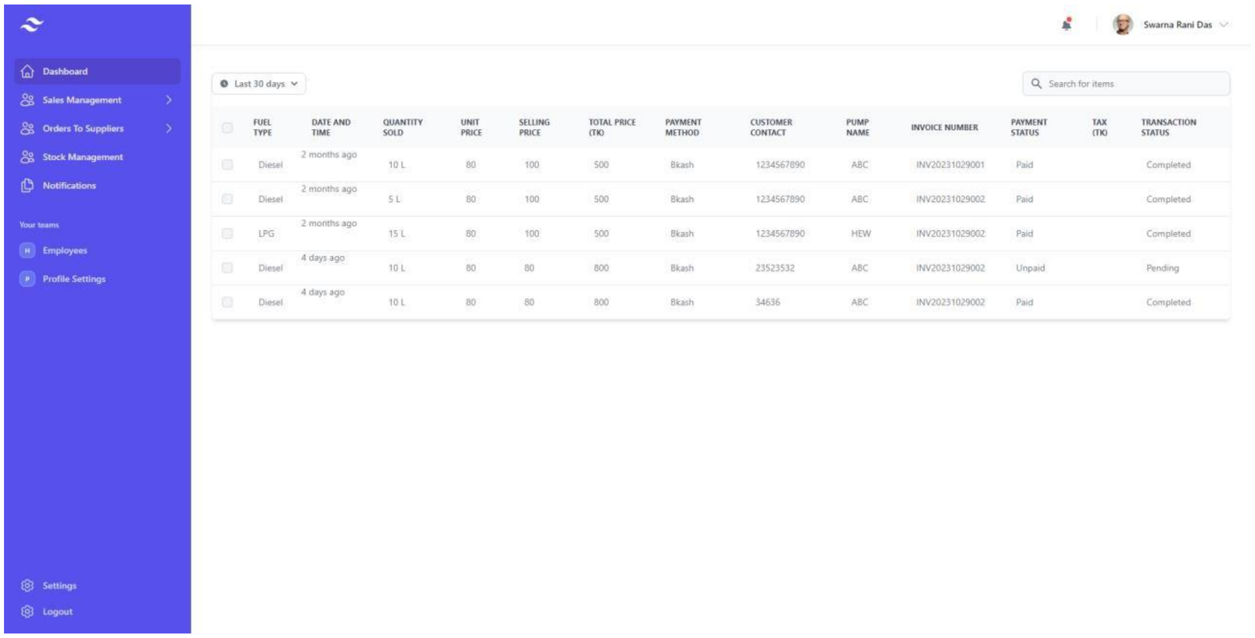
Payment Status: Paid

Transaction Status: Pending

FUEL TYPE	DATE AND TIME	QUANTITY SOLD	PUMP NAME	INVOICE NUMBER	PAYMENT STATUS	TAX (TK)	TRANSACTION STATUS	ACTION
Diesel	2 months ago	10 L	ABC	INV20231029001	Paid		Completed	Edit
Diesel	2 months ago	5 L	ABC	INV20231029002	Paid		Completed	Edit
LPG	2 months ago	15 L	HEW	INV20231029002	Paid		Completed	Edit
Diesel	4 days ago	10 L	ABC	INV20231029002	Unpaid		Pending	Edit
Diesel	4 days ago	10 L	ABC	INV20231029002	Paid		Completed	Edit

Check Sell History

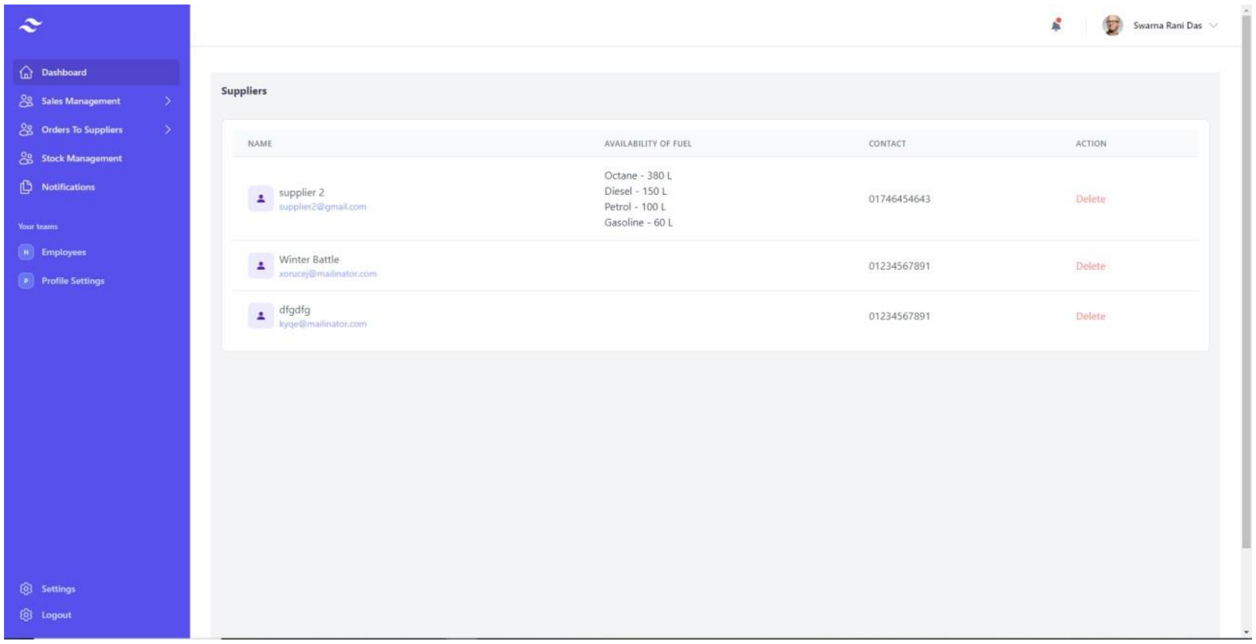
Admin can check Sell History from there.



FUEL TYPE	DATE AND TIME	QUANTITY SOLD	UNIT PRICE	SELLING PRICE	TOTAL PRICE (TK)	PAYMENT METHOD	CUSTOMER CONTACT	PUMP NAME	INVOICE NUMBER	PAYMENT STATUS	TAX (TK)	TRANSACTION STATUS
Diesel	2 months ago	10 L	80	100	500	Bkash	1234567890	ABC	INV20231029001	Paid		Completed
Diesel	2 months ago	5 L	80	100	500	Bkash	1234567890	ABC	INV20231029002	Paid		Completed
LPG	2 months ago	15 L	80	100	500	Bkash	1234567890	HEW	INV20231029002	Paid		Completed
Diesel	4 days ago	10 L	80	80	800	Bkash	23523532	ABC	INV20231029002	Unpaid		Pending
Diesel	4 days ago	10 L	80	80	800	Bkash	34636	ABC	INV20231029002	Paid		Completed

Supplier's List & Their Stock

Admin can see the list of the suppliers and the fuel stock.

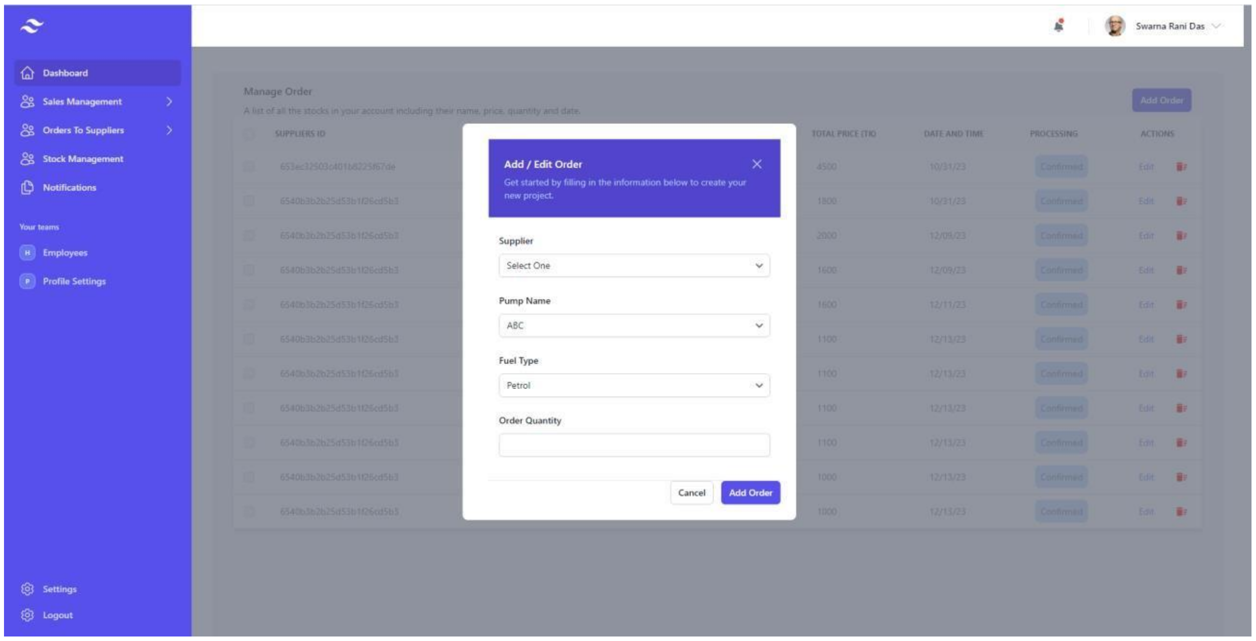


Suppliers

NAME	AVAILABILITY OF FUEL	CONTACT	ACTION
supplier 2 supplier2@gmail.com	Octane - 380 L Diesel - 150 L Petrol - 100 L Gasoline - 60 L	01746454643	Delete
Winter Battle xorucej@mailinator.com		01234567891	Delete
dfgdfg kyqe@mailinator.com		01234567891	Delete

Placing Order to The Supplier

Admin can place orders from there.



Add / Edit Order

Get started by filling in the information below to create your new project.

Supplier: Select One

Pump Name: ABC

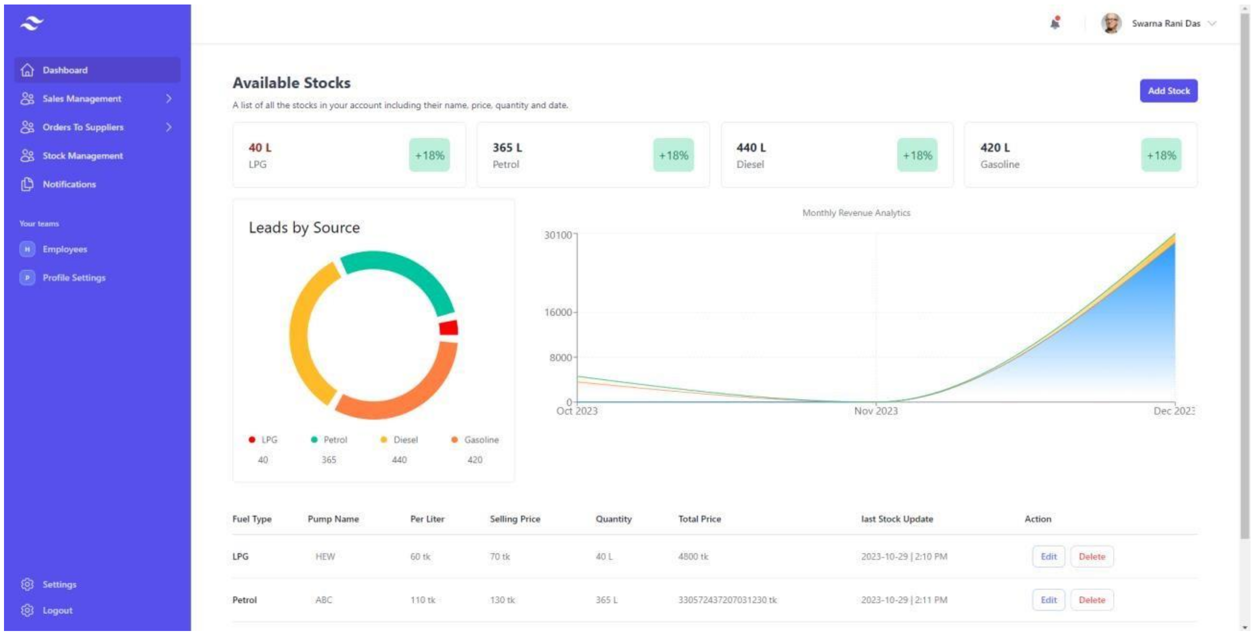
Fuel Type: Petrol

Order Quantity:

Cancel | Add Order

Fuel Stock of The Stations

Admin can check his own stock from there.



Available Stocks

A list of all the stocks in your account including their name, price, quantity and date.

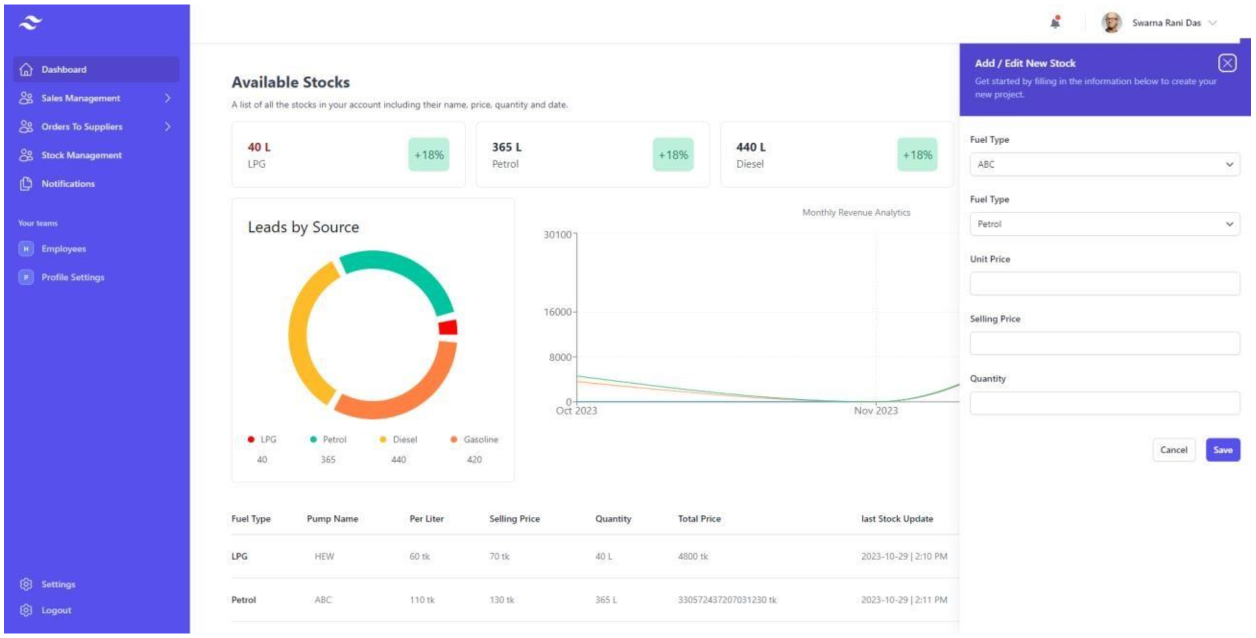
40 L LPG +18%	365 L Petrol +18%	440 L Diesel +18%	420 L Gasoline +18%

LPG	Petrol	Diesel	Gasoline
40	365	440	420

Fuel Type	Pump Name	Per Liter	Selling Price	Quantity	Total Price	last Stock Update	Action
LPG	HEW	60 tk	70 tk	40 L	4800 tk	2023-10-29 | 2:10 PM	Edit Delete
Petrol	ABC	110 tk	130 tk	365 L	330572437207031230 tk	2023-10-29 | 2:11 PM	Edit Delete

Add Stock

Admin can add new stock, edit stock, or delete stock from there.



Available Stocks

A list of all the stocks in your account including their name, price, quantity and date.

Add / Edit New Stock

Get started by filling in the information below to create your new project.

Fuel Type: ABC

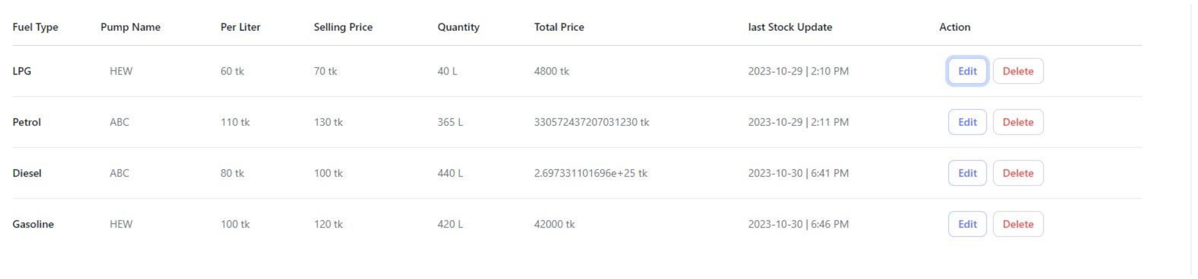
Fuel Type: Petrol

Unit Price

Selling Price

Quantity

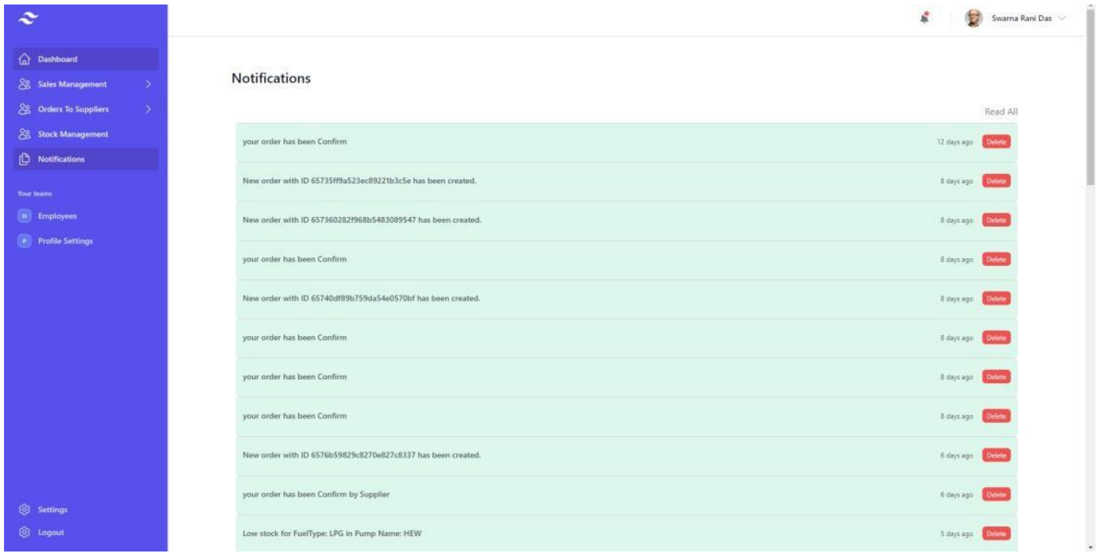
Fuel Type	Pump Name	Per Liter	Selling Price	Quantity	Total Price	last Stock Update
LPG	HEW	60 tk	70 tk	40 L	4800 tk	2023-10-29 | 2:10 PM
Petrol	ABC	110 tk	130 tk	365 L	330572437207031230 tk	2023-10-29 | 2:11 PM



Fuel Type	Pump Name	Per Liter	Selling Price	Quantity	Total Price	last Stock Update	Action
LPG	HEW	60 tk	70 tk	40 L	4800 tk	2023-10-29 | 2:10 PM	Edit Delete
Petrol	ABC	110 tk	130 tk	365 L	330572437207031230 tk	2023-10-29 | 2:11 PM	Edit Delete
Diesel	ABC	80 tk	100 tk	440 L	2.697331101696e+25 tk	2023-10-30 | 6:41 PM	Edit Delete
Gasoline	HEW	100 tk	120 tk	420 L	42000 tk	2023-10-30 | 6:46 PM	Edit Delete

Notifications

Admin and Supplier can check notifications from there



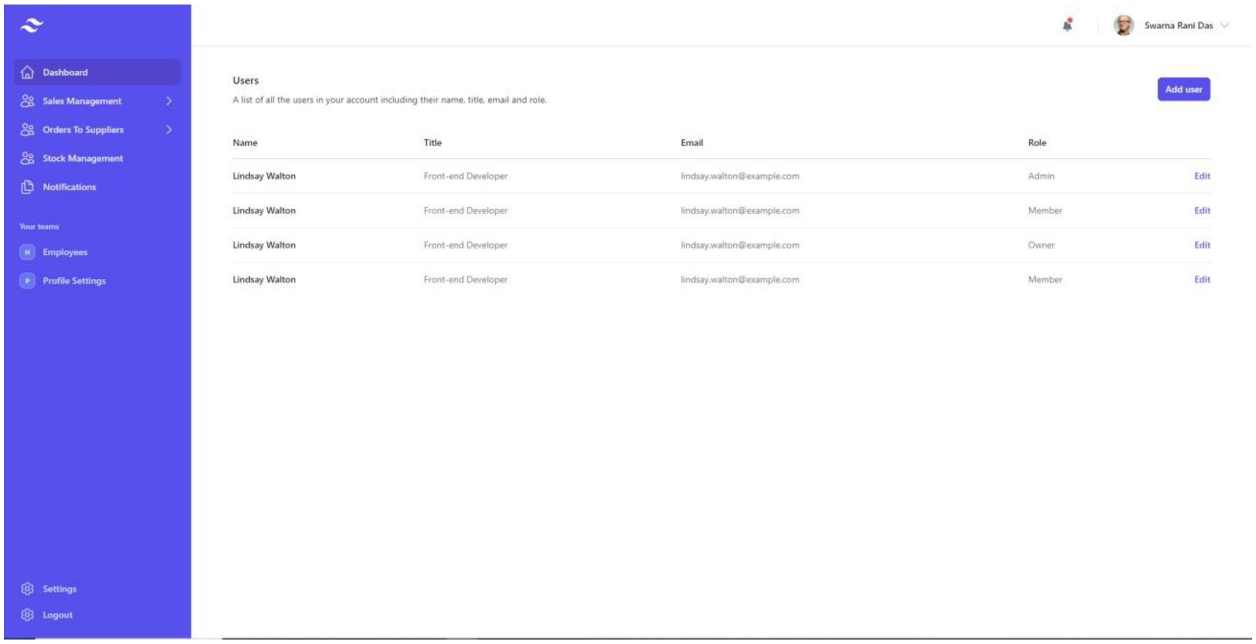
Notifications

Read All

your order has been Confirm	12 days ago	Delete
New order with ID 65735ff9a523ec89221b3c5e has been created.	8 days ago	Delete
New order with ID 657360282f968b5483089547 has been created.	8 days ago	Delete
your order has been Confirm	8 days ago	Delete
New order with ID 65740df89b759da54e0570bf has been created.	8 days ago	Delete
your order has been Confirm	8 days ago	Delete
your order has been Confirm	8 days ago	Delete
your order has been Confirm	8 days ago	Delete
New order with ID 6576b59829c8270e827c8337 has been created.	6 days ago	Delete
your order has been Confirm by Supplier	6 days ago	Delete
Low stock for FuelType: LPG in Pump Name: HEW	5 days ago	Delete

Manage User

Admin can add or remove members from there.



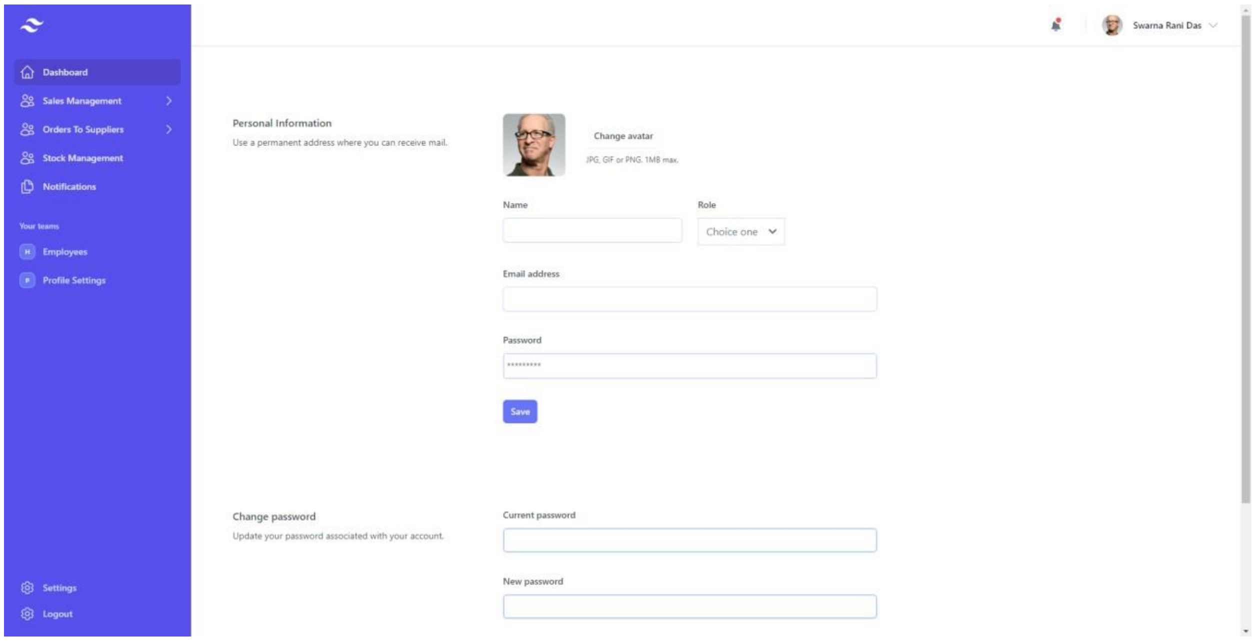
Users

A list of all the users in your account including their name, title, email and role.

Name	Title	Email	Role	
Lindsay Walton	Front-end Developer	lindsay.walton@example.com	Admin	Edit
Lindsay Walton	Front-end Developer	lindsay.walton@example.com	Member	Edit
Lindsay Walton	Front-end Developer	lindsay.walton@example.com	Owner	Edit
Lindsay Walton	Front-end Developer	lindsay.walton@example.com	Member	Edit

Profile Setting Page

Users can update their profile from there.



Personal Information

Use a permanent address where you can receive mail.

JPG, GIF or PNG. 1MB max.

Name

Role: Choice one

Email address

Password: *********

Change password

Update your password associated with your account.

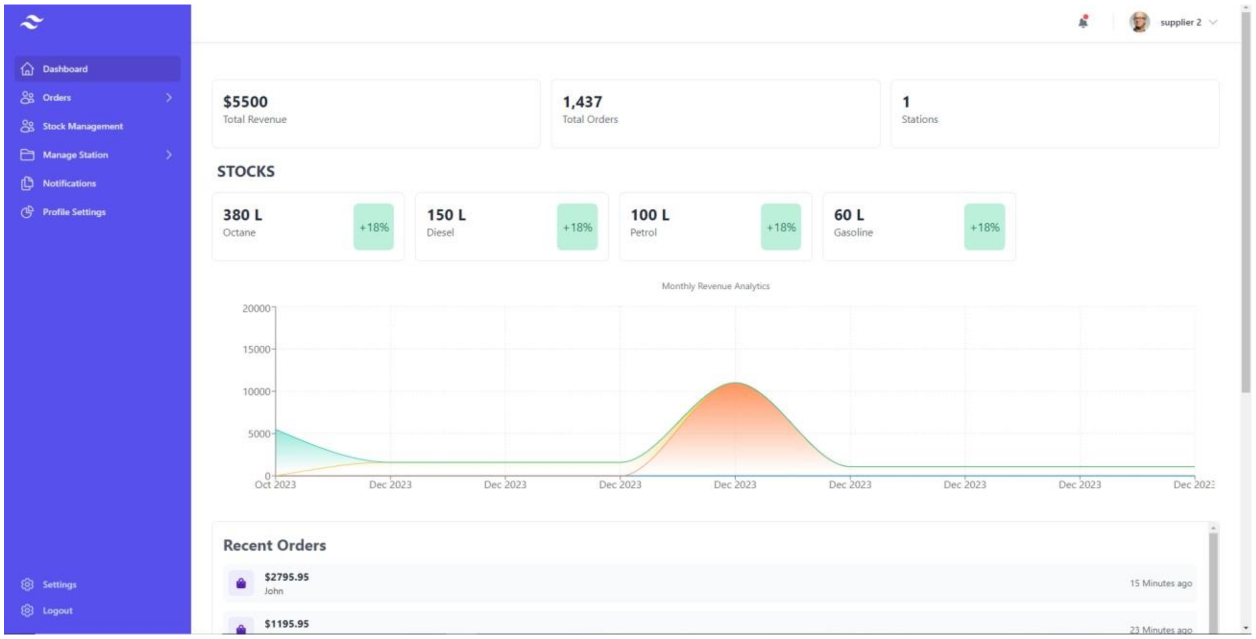
Current password

New password

6.2 Landing Page (Only Supplier)

Dashboard of Supplier

This is the dashboard of the supplier.



$5500 Total Revenue

1,437 Total Orders

1 Stations

STOCKS

380 L Octane +18%

150 L Diesel +18%

100 L Petrol +18%

60 L Gasoline +18%

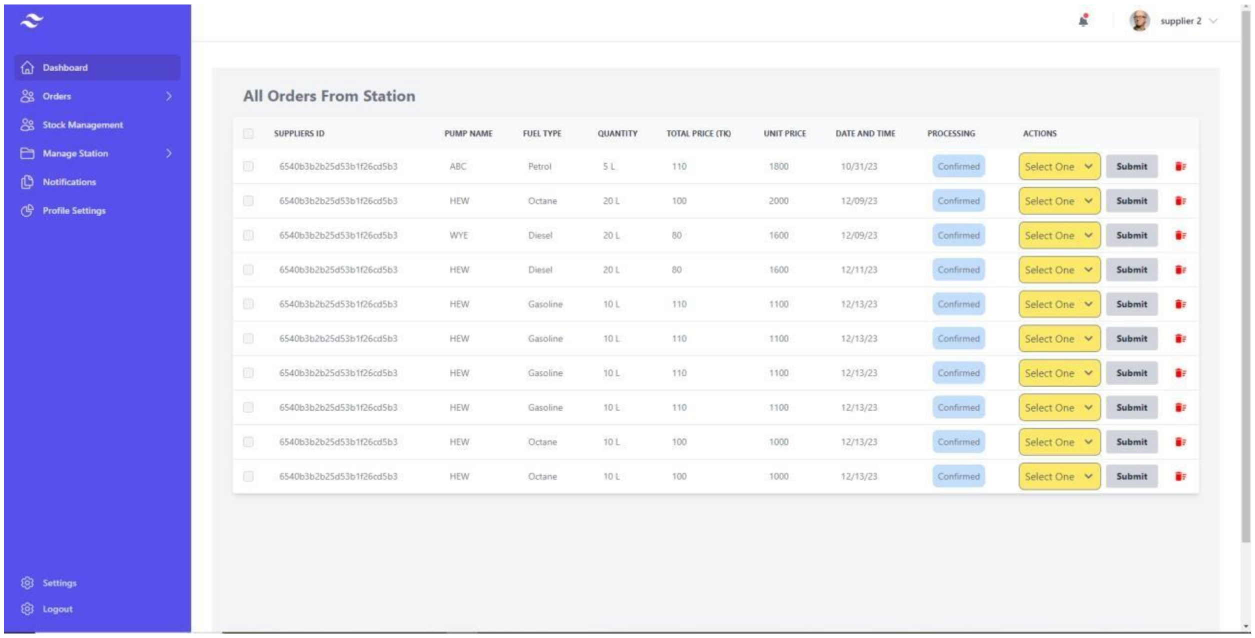
Recent Orders

$2795.95 John 15 Minutes ago

$1195.95 23 Minutes ago

Check Order and Manage

Suppliers can check the placed order from the station owner and confirm or cancel the order from there.

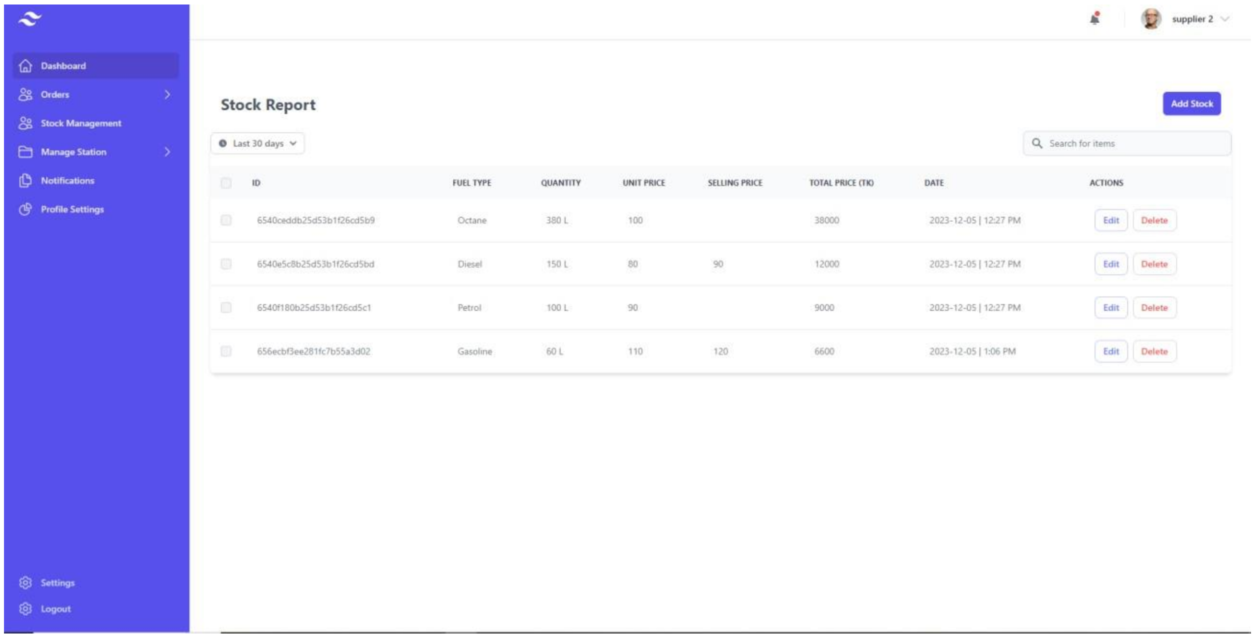


All Orders From Station

SUPPLIERS ID	PUMP NAME	FUEL TYPE	QUANTITY	TOTAL PRICE (TK)	UNIT PRICE	DATE AND TIME	PROCESSING	ACTIONS
6540b3b2b25d53b1f26cd5b3	ABC	Petrol	5 L	110	1800	10/31/23	Confirmed	Select One Submit
6540b3b2b25d53b1f26cd5b3	HEW	Octane	20 L	100	2000	12/09/23	Confirmed	Select One Submit
6540b3b2b25d53b1f26cd5b3	WYE	Diesel	20 L	80	1600	12/09/23	Confirmed	Select One Submit
6540b3b2b25d53b1f26cd5b3	HEW	Diesel	20 L	80	1600	12/11/23	Confirmed	Select One Submit
6540b3b2b25d53b1f26cd5b3	HEW	Gasoline	10 L	110	1100	12/13/23	Confirmed	Select One Submit
6540b3b2b25d53b1f26cd5b3	HEW	Gasoline	10 L	110	1100	12/13/23	Confirmed	Select One Submit
6540b3b2b25d53b1f26cd5b3	HEW	Gasoline	10 L	110	1100	12/13/23	Confirmed	Select One Submit
6540b3b2b25d53b1f26cd5b3	HEW	Gasoline	10 L	110	1100	12/13/23	Confirmed	Select One Submit
6540b3b2b25d53b1f26cd5b3	HEW	Octane	10 L	100	1000	12/13/23	Confirmed	Select One Submit
6540b3b2b25d53b1f26cd5b3	HEW	Octane	10 L	100	1000	12/13/23	Confirmed	Select One Submit

Add or Remove Stock

Suppliers can add or remove stock from there.



Stock Report

ID	FUEL TYPE	QUANTITY	UNIT PRICE	SELLING PRICE	TOTAL PRICE (TK)	DATE	ACTIONS
6540ceddb25d53b1f26cd5b9	Octane	380 L	100		38000	2023-12-05 | 12:27 PM	Edit Delete
6540e5c8b25d53b1f26cd5bd	Diesel	150 L	80	90	12000	2023-12-05 | 12:27 PM	Edit Delete
6540f180b25d53b1f26cd5c1	Petrol	100 L	90		9000	2023-12-05 | 12:27 PM	Edit Delete
656ecbf3ee281fc7b55a3d02	Gasoline	60 L	110	120	6600	2023-12-05 | 1:06 PM	Edit Delete

Chapter 7 Conclusion

In conclusion, the development and implementation of the Fuel Stock Tracking System web application have been a significant endeavor aimed at addressing the critical need for efficient fuel management. This project has successfully culminated in the creation of a robust and user-friendly system that facilitates seamless recording, monitoring, and analysis of fuel transactions.

7.1 Project Link

GitHub Link: https://github.com/swarnaDas01/fuel-tracking.git

7.2 Limitations

▪ Messaging option is not available on the system.
▪ The system is still in the earlier phase. That's why there will be more features that will be implemented.
▪ Registration and email verification is not available

7.3 Future Scope

The future of my fuel stock tracking system website with JavaScript holds exciting possibilities to expand its functionality and cater to evolving needs. Here are some potential areas to explore:

- Machine Learning: Integrate machine learning algorithms to predict future fuel consumption based on historical data and external factors like weather or traffic patterns.
- Automated Insights: Generate and deliver custom reports with actionable insights on fuel usage, cost optimization, and delivery schedules.
- Interactive Dashboards: Implement interactive dashboards with drag-and-drop features, customizable visualizations, and drill-down capabilities for deeper data exploration.
- Third-Party API Integration: Connect with fuel delivery services, fleet management systems, or accounting software for automated tasks and data exchange.
- Alert and Notification System: Enhance the notification system with customizable alerts for various scenarios, including critical fuel levels, sensor malfunctions, or unauthorized access attempts.
- Environmental Impact Tracking: Calculate and report the carbon footprint associated with fuel consumption for eco-conscious tracking and optimization.

Reference:

Project Idea: Filling Station Management System by ClikBD
https://www.clickbd.com/bangladesh/1655718-filling-station-management-system.html

- Development: Microsoft Visual Studio.
- Diagram: drawio.io - **https://app.diagrams.net**