SAVITRIBAI PHULE PUNE UNIVERSITY

MINI- PROJECT REPORT

OF THE PROJECT

FUEL STATION DELIVERY

***(A project to fulfill the requirements of WT lab*)**

THIRD YEAR ENGINEERING

As prescribed by

By

Saurabh raut	**64**
Pranav Shimpi	**66**
Sanket Shirsath	**69**
Parimal Thakre	**74**

Under the Guidance of

Prof. Reshma Dhurjad Mam

DEPARTMENT OF COMPUTER ENGINEERING

K. K. WAGH INSTITUTE OF ENGINEERING EDUCTION AND RESEARCH

(Academic Year: 2021 – 2022)

CONTENTS

Abstract:

The increasing demand for efficient and streamlined fuel delivery services has led to the development of advanced technologies to automate the process. This abstract presents a Fuel Station Delivery Management System (FSDMS) that utilizes the MERN (MongoDB, Express.js, React.js, Node.js) stack to create a robust and scalable solution for fuel station deliveries.

The FSDMS aims to enhance the overall delivery process by integrating multiple stakeholders, including fuel station owners, delivery drivers, and customers, into a unified platform. By leveraging the power of the MERN stack, the system provides real-time monitoring, order management, and data analytics capabilities, thereby ensuring seamless fuel delivery operations.

Introduction

The fuel industry plays a critical role in keeping transportation and businesses running smoothly. With the increasing demand for efficient fuel delivery services, there is a growing need for innovative technologies to streamline the process and enhance customer satisfaction. This introduction presents a Fuel Station Delivery system developed using the MERN (MongoDB, Express.js, React.js, Node.js) stack, offering a robust and scalable solution to meet the evolving needs of fuel stations and their customers.

Traditionally, fuel station deliveries have relied on manual processes, leading to challenges such as inefficient resource allocation, lack of real-time visibility, and limited data insights. The MERN stack provides a comprehensive and interconnected technology framework that addresses these challenges, enabling seamless communication and integration across various components of the system.

The Fuel Station Delivery system leverages MongoDB, a NoSQL database, to store and manage critical data related to fuel station inventory, customer information, and delivery orders. The flexibility and scalability of MongoDB allow for efficient data handling, accommodating the dynamic nature of fuel delivery operations.

3. PROJECT REQUIREMENTS

SOFTWARE REQUIREMENT SPECIFICATION :

- Technology : REACT, NodeJs, ExpressJs
- Web-Technologies : HTML,Javascript,CSS
- Web Server : NODE JS
- Backend Database : MONGODB
- Text Editor : Visual Studio Code

TECHNOLOGY OVERVIEW

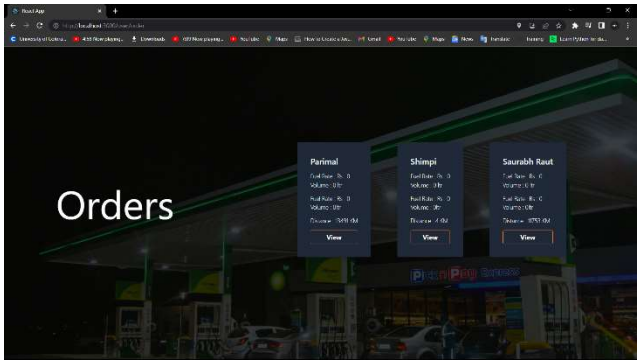
The Fuel Station Delivery system, built using the MERN (MongoDB, Express.js, React.js, Node.js) stack, incorporates a range of technologies to provide a comprehensive and efficient solution for fuel station delivery operations. This technology overview highlights the key components and their roles in the system.

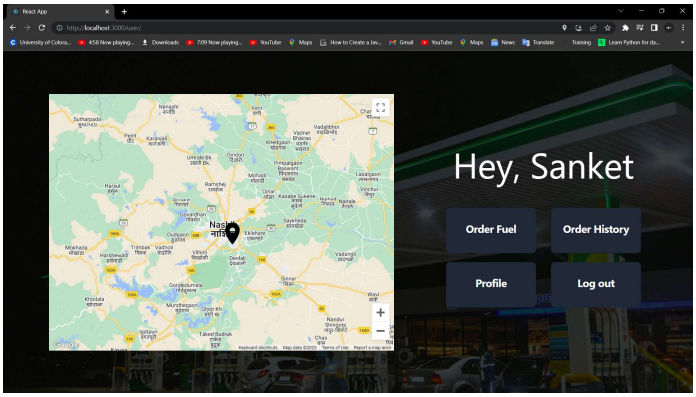
1. **MongoDB**: MongoDB, a NoSQL document-oriented database, serves as the system's backend data store. It allows for flexible and scalable data storage, accommodating the dynamic nature of fuel station inventory, customer information, and delivery orders. MongoDB's JSON-like document structure enables easy integration with the JavaScript-based MERN stack.
2. **Express**.js: Express.js is a robust web application framework for Node.js that forms the backend of the Fuel Station Delivery system. It provides a foundation for handling HTTP requests, routing, and middleware

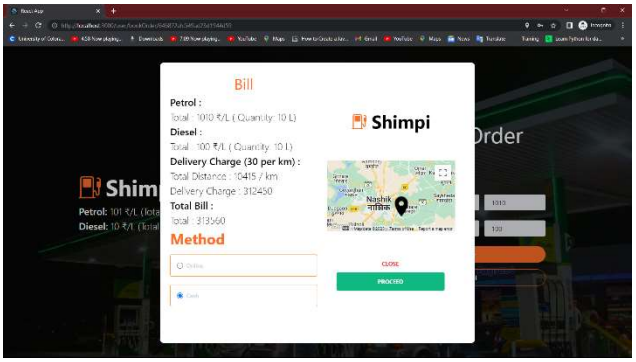
functions. Express.js simplifies the development of server-side logic, allowing seamless communication with the frontend, handling authentication, and integrating with external APIs for fuel inventory management.

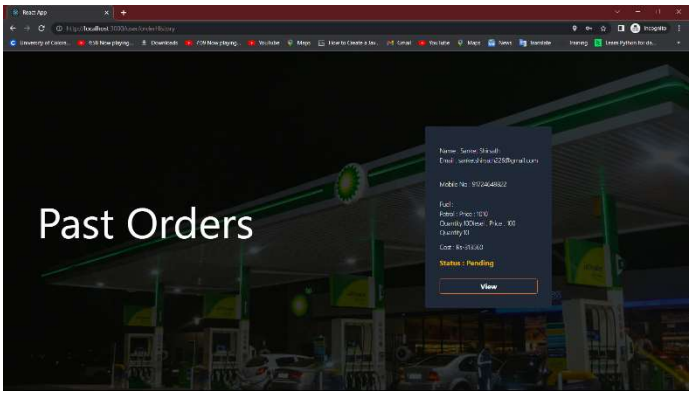
3. **React**.js: React.js is a popular JavaScript library for building user interfaces. It is used to develop the frontend of the Fuel Station Delivery system, providing a responsive and interactive user interface for fuel station owners, delivery drivers, and customers. React.js facilitates the creation of reusable UI components and efficient rendering, enhancing the overall user experience.
4. **Node**.js: Node.js is a JavaScript runtime environment that allows server-side execution of JavaScript code. It serves as the backbone of the system, facilitating the integration of Express.js with MongoDB and React.js. Node.js enables event-driven and non-blocking I/O operations, making it ideal for building scalable and high-performance web applications.
5. **RESTful APIs**: The Fuel Station Delivery system utilizes RESTful APIs (Application Programming Interfaces) to enable communication and data exchange between various components. APIs allow fuel station owners, delivery drivers, and customers to interact with the system, place orders, track deliveries, and access relevant information. Express.js and Node.js facilitate the development and management of these APIs, ensuring smooth integration with the frontend and backend components.
6. **Cloud-based Infrastructure**: The MERN stack Fuel Station Delivery system can leverage cloud-based infrastructure services such as Amazon Web Services (AWS) or Microsoft Azure. Cloud platforms provide scalability, reliability, and cost-efficiency, allowing the system to handle varying workloads and ensuring high availability and data security.

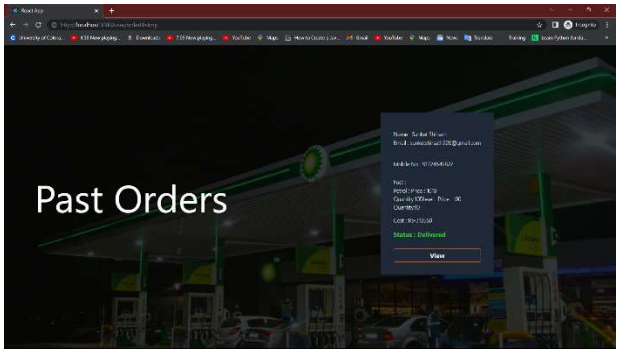
4. Graphical User Interface -

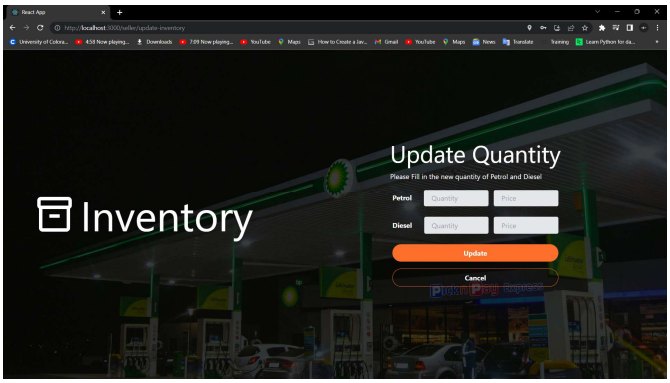


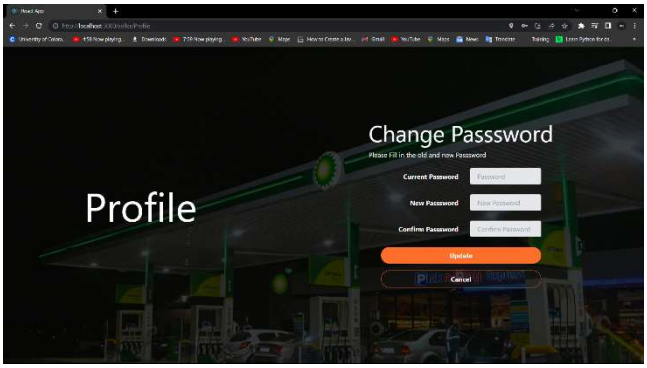


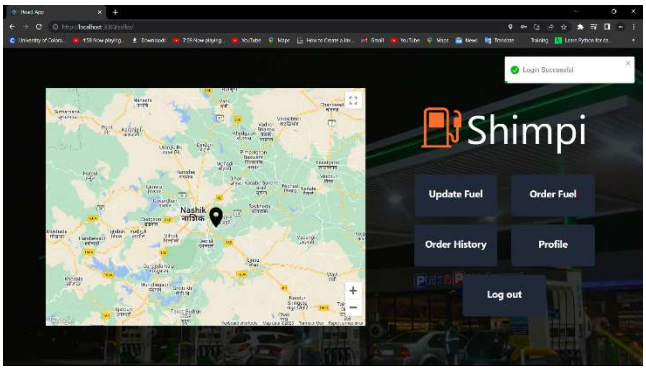


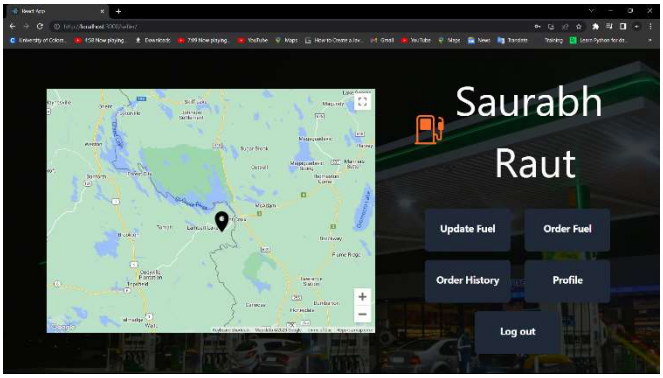


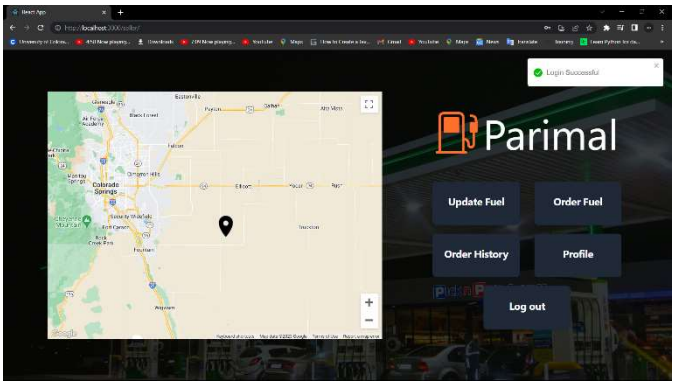


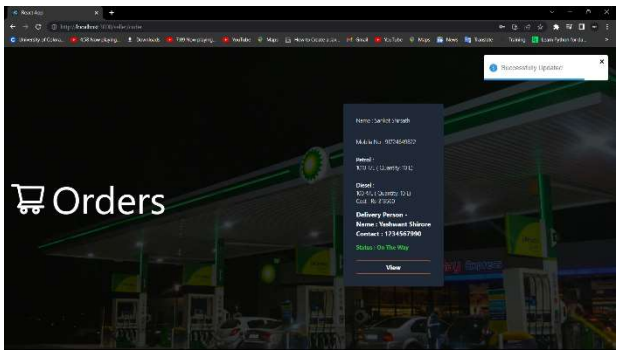

Inventory
Update Quantity
Please Fill in the new quantity of Petrol and Diesel
Petrol
Diesel
Update
Cancel


Profile
Change Passsword
Please Fill in the old and new Password
Current Password
New Password
Confirm Password
Update
Cancel


Login Successful
Shimpi
Update Fuel
Order Fuel
Order History
Profile
Log out


Saurabh
Raut
Update Fuel
Order Fuel
Order History
Profile
Log out


Login Successful
Parimal
Update Fuel
Order Fuel
Order History
Profile
Log out


Orders
View

Test Development:

1. Test case Development (check list)

2. Test Procedure preparation. (Description of the Test cases).

Result Analysis:

1. Expected value: is nothing but expected behaviorof application.

2. Actual value: is nothing but actual behavior ofapplication

Bug Tracing: Collect all the failed cases, prepare documents.

Reporting: Prepare document (status of the application)

TCD (Test Case Document):

Test Case Document Contains

- **Test Scope (or) Test objective**
- **Test Scenario**
- **Test Procedure**
- **Test case**

This is the sample test case document for the Academic details of student project:

Test scope:

- Testing plays a crucial role in ensuring the quality, functionality, and reliability of the Fuel Station Delivery system developed using the MERN (MongoDB, Express.js, React.js, Node.js) stack.
- The test scope encompasses various aspects of the system, covering both functional and non-functional requirements.

Test Procedure:

- The procedure for testing this screen is planned in such a way that the data entry, status calculation functionality, saving and quitting operations are tested in terms of Gui testing, Positive testing, Negative testing using the corresponding Gui test cases, Positive test cases, Negative test cases respectively

Test Cases:

• Template for Test Case

T.C.No	Description	Exp	Act	Result
1	Enter user name and password	True/false	True	Home page
2	Enter valid name to store in the database	Accurate/Valid data	Valid name	Data stored successfully

Guidelines for Test Cases:

1. GUI Test Cases:

- Total no of features that need to be check
- Look & Feel
- Look for Default values if at all any (date & Time, if at all any require)
- Look for spell check

2. Positive Test Cases:

- Valid inputs must be used for testing
- Must have the positive perception to verify whether the requirements are justified.

3. Negative Test Cases:

- Must have negative perception.
- Invalid inputs must be used for test.

BENEFITS:

The project is identified by the merits of the system offered to the user. The merits of this project are as follows: -

- It's a web-enabled project.
- This project offers user to enter the data through simple and interactive forms. This is very helpful for the client to enter the desired information through so much simplicity.
- The user is mainly more concerned about the validity of the data, whatever he is entering. There are checks on every stage of any new creation, data entry or updating so that the user cannot enter the invalid data, which can create problems at later date.
- Sometimes the user finds in the later stages of using project that he needs to update some of the information that he entered earlier. There are options for him by which he can update the records. Moreover, there is restriction for his that he cannot change the primary data field. This keeps the validity of the data to longer extent.

LIMITATIONS:

- The size of the database increases day-by-day, increasing the load on the database back up and data maintenance activity.

- Training for simple computer operations is necessary for the users working on the system.

Conclusion:

The system has been developed for the given conditions and is found working effectively. The developed system is flexible and changes, if required, can be made easily. Using the facilities and functionalities of the software has been developed in a neat and simple manner, thereby reducing the operator's work.

The speed and accuracy are maintained in proper way. The user-friendly nature of this software makes it very easy to work with all categories of people from tech-savvy to people with little knowledge of computers.

The results obtained were fully satisfactory from the user point of view. The system was verified and validated in each manner. The system has been developed with an insight into the necessary modifications that may be required in the future. Hence the system can be maintained successfully without much effort.

Prof. Reshma Dhurjad Mam
(Project Guide)

Prof. S.S.Sane Sir
(HOD)