

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANASANGAMA, BELAGAVI – 590018**



**Mini Project Report
on**

WATER PUMPS COMPANY MANAGEMENT SYSTEM

Submitted in partial fulfillment for the award of degree of

**Bachelor of Engineering
In
Artificial Intelligence and Machine Learning**

Submitted by
Pavithra M
1BG21AI079



Vidyayāmṛuthamashnuthe

B.N.M. Institute of Technology

An Autonomous Institution under VTU

Department of Artificial Intelligence and Machine Learning

2022-23

B.N.M. Institute of Technology

An Autonomous Institution under VTU

Department of Artificial Intelligence and Machine Learning



Vidyayāmruthamashnuthu

CERTIFICATE

Certified that the Mini Project entitled **Water Pumps Company Management System** carried out by Ms. **Pavithra M** USN **1BG21AI079** a bonafide student of IV Semester B.E., **B.N.M Institute of Technology** in partial fulfillment for the Bachelor of Engineering in **ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING** of the **Visvesvaraya Technological University**, Belagavi during the year 2022-23. It is certified that all corrections / suggestions indicated for Internal Assessment have been incorporated in the report. The Mini project report has been approved as it satisfies the academic requirements in respect of Database Management System prescribed for the said Degree.

Mrs. Poornima Manjunath
Assistant Professor
Department of AIML
BNMIT, Bengaluru

Dr. Sheba Selvam
Professor and HOD
Department of AIML
BNMIT, Bengaluru

ABSTRACT

The "Water Pumps Company Database Management System" serves as a pivotal tool for optimizing the operations of a water pumps manufacturing and distribution company. This project employs a robust relational database, enriched with careful normalization and efficient functional dependencies, to seamlessly handle customer interactions, streamline order processing, manage supplier relationships, and maintain a comprehensive audit trail. Through a user-friendly interface, authorized personnel can effortlessly navigate and harness the power of this system, which not only ensures data integrity but also lays the foundation for future scalability and enhancement. In a dynamic industry landscape, this system stands as a testament to effective database management, bolstering productivity, transparency, and growth for the company.

ACKNOWLEDGEMENT

I would like to thank **Shri Narayan Rao R. Maanay**, Secretary, BNMEI, Bengaluru for providing an excellent academic environment in the College.

I would like to sincerely thank **Prof. T. J. Rama Murthy**, Director, BNMIT, Bengaluru, for having extended his support and encouraging me during the course of the work.

I would like to sincerely thank **Dr. S.Y. Kulkarni**, Additional Director, BNMIT, Bengaluru for having extended his support and encouraging me during the course of the work.

I would like to express my gratitude to **Prof. Eishwar N. Maanay**, Dean, BNMIT, Bengaluru for his relentless support, guidance and assistance.

I would like to thank **Dr. Krishnamurthy G.N**, Principal, BNMIT, Bengaluru for his constant encouragement.

I would like to thank **Dr. Sheba Selvam**, Professor and Head of the Department of Artificial Intelligence and Machine Learning, BNMIT, Bengaluru who has shared her opinions and thoughts which helped me in the completion of my project successfully.

I would also like to thank my Course teachers **Mrs. Poornima Manjunath**, Assistant Professor and **Mrs. Pankaja R**, Assistant Professor, Department of Artificial Intelligence and Machine Learning, BNMIT, Bengaluru for guiding in a systematic manner.

Finally, I would like to thank all technical and non-technical faculty members of Department of Artificial Intelligence and Machine Learning, BNMIT, Bengaluru, for their support. I would like to thank my Family and Friends for their unfailing moral support and encouragement.

PAVITHRA M

(1BG21AI079)

TABLE OF CONTENTS

CONTENTS	Page No.
ABSTRACT	I
ACKNOWLEDGEMENT	II
1. INTRODUCTION	
1.1. Overview of Database Management System	1
1.2. Problem Statement	1
1.3. Objective	2
1.4. Dataset Description	2
2. SYSTEM REQUIREMENTS	
2.1. Hardware Requirements	3
2.2. Software Requirements	3
3. SYSTEM DESIGN	
3.1. ER Diagram	4
3.2. ER to Relation Mapping	4
3.3. Schema Diagram	6
3.4. Over view of GUI	7
3.5. Normalization	8
4. IMPLEMENTATION	
4.1. Table Creation	10
4.2. Description of tables	11
4.3. Populated Tables	12
4.4. SQL Triggers and Stored Procedures	14
4.5. Database Connectivity	17
4.6. Modules	18
5. RESULTS	20
6. CONCLUSION AND FUTURE ENHANCEMENTS	29
REFERENCES	

LIST OF FIGURES

FIGURES	Page No.
3.1. ER Diagram of the water pumps company database management system.	4
3.2. All the entities and attributes	5
3.3. Schema Diagram	7
4.2. Description of tables	
4.2.1 Description of customers table	11
4.2.2 Description of supplier table	12
4.2.3 Description of pumps table	12
4.2.4 Description of orders table	12
4.2.5 Description of logs table	12
4.3 Populated tables	
4.3.1 Values in customers table	13
4.3.2 Values in supplier table	13
4.3.3 Values in pumps table	13
4.3.4 Values in orders table	14
4.3.5 Values in logs table	14
4.4. Trigger	15
4.6. Modules and dataflow design	19
5. Results	
5.1. Login Page	20
5.2. Dashboard Page	20
5.3. Home page (1)	21
5.4. home page (2) with customer information	21

5.5. Add customer information (1)	21
5.6 Add customer information (2)	22
5.7. View customer Orders (1)	22
5.8. view customer orders (2)	22
5.9. new order (1)	23
5.10. new order (2)	23
5.11. new order (3)	23
5.12. details (1)	24
5.13. details (2) (trigger)	24
5.14. stored details(stored procedure)(1)	24
5.15. stored details (2)	25
5.16. Add/search items(1)	25
5.17. Add/search items(2)	25
5.18. pumps list	26
5.19. suppliers list	26
5.20. searching	26
5.21. Adding	27
5.22. pump added	27
5.23. About us(1)	27
5.24. About us(2)	28
5.25. Logged out	28

Chapter 1

INTRODUCTION

1.1 Overview of Database Management System

A Database Management System (DBMS) is a software application that facilitates the management, organization, storage, and retrieval of data in a structured and efficient manner. It serves as an intermediary between users and the physical data stored in a database, enabling users to interact with the data without needing to understand the underlying complexities of storage and retrieval.

DBMS plays a pivotal role in modern information systems, offering several key advantages. It provides data integrity and security by enforcing access controls and ensuring that data is accurate and consistent. Additionally, DBMS allows for data sharing among multiple users and applications, promoting collaboration and reducing data redundancy. It offers a structured way to define, manipulate, and query data through a structured query language (SQL), making it easier for users to extract valuable insights and information from the stored data.

As technology has evolved, so has the concept of DBMS. The transition from DBMS 2 to DBMS 3 marked a significant advancement. DBMS 3 introduced the concept of a relational database, where data is organized into tables with rows and columns, fostering a more flexible and scalable approach to data management. This shift allowed for the development of complex queries and joins, enabling users to retrieve specific information from multiple tables. Furthermore, DBMS 3 introduced the ACID (Atomicity, Consistency, Isolation, Durability) properties to ensure data integrity even in the face of system failures, making it suitable for critical applications like banking and finance.

1.2 Problem Statement

The Water Pumps Company currently faces operational challenges due to inefficient data management, manual record-keeping, and disjointed processes. These issues hinder effective inventory management, customer interactions, order processing, and maintenance scheduling.

To address these challenges, there is a need to develop a robust and user-friendly Database Management System (DBMS) tailored to the company's specific requirements. The DBMS should centralize data, automate processes, and provide real-time access to accurate and comprehensive information, aiming to streamline operations, enhance customer satisfaction, and optimize overall business performance.

1.3 Objective

The Water Pumps Company DBMS project seeks to address operational inefficiencies through the development of a streamlined data management system. This system aims to centralize data related to water pump products, customers, orders, suppliers, and maintenance records. By doing so, the project aims to enhance inventory management, optimize order processing, and improve customer engagement. Furthermore, the project aims to foster collaboration with suppliers by creating a module for managing supplier details and procurement processes. The DBMS will also facilitate efficient maintenance scheduling, minimizing equipment downtime. Informed decision-making is another objective, with tools to analyse sales trends and generate insightful reports for strategic planning. The DBMS will provide a user-friendly interface for data input and retrieval, ensuring ease of use and efficiency. Data security will be prioritized through measures such as user authentication and encryption, safeguarding sensitive information. The architecture will be designed with scalability in mind, accommodating future growth. To ensure uninterrupted operations, automated backup and recovery mechanisms will be implemented. Comprehensive user training and ongoing technical support will empower effective system utilization, contributing to enhanced operational efficiency and customer satisfaction.

1.4 Dataset Description

The dataset underpinning the water pumps company's database management system is a structured compilation of vital information crucial for seamless operations. It encompasses diverse entities, including customers, suppliers, pumps, orders, and activity logs. Customers are identified by their unique IDs and characterized by their names, ages, contact details, and addresses. Suppliers are similarly distinguished by IDs and their respective names. Pumps, essential to the company's offerings, are designated by distinct IDs and descriptive names.

Chapter 2

SYSTEM REQUIREMENTS

In this section, we discuss the hardware and software components that contribute to the successful implementation and operation of our Flask web application. The combination of suitable hardware and software ensures optimal performance and user experience.

2.1 Hardware Requirements

- Processor: Intel Core i5 or equivalent
- RAM: 8 GB or more
- Hard Disk: 1 TB or sufficient space for database and files

2.2 Software Requirements

- Operating System: Windows 10 or higher (64-bit)
- Front-End: HTML, CSS, JavaScript
- Server-Side Language: Python with Flask framework
- Back-End Database: MySQL
- Web Server: Apache
- Browser: Chrome or Firefox
- Application Software: XAMPP

Chapter 3

SYSTEM DESIGN

3.1 E R Diagram

The Entity-Relationship (ER) diagram is a visual representation of the relationships between different entities in our database. It provides an overview of how entities are related and interact with each other within the system. The ER diagram helps us understand the data structure, dependencies, and flow of information in our Flask web application, contributing to effective database design and management.

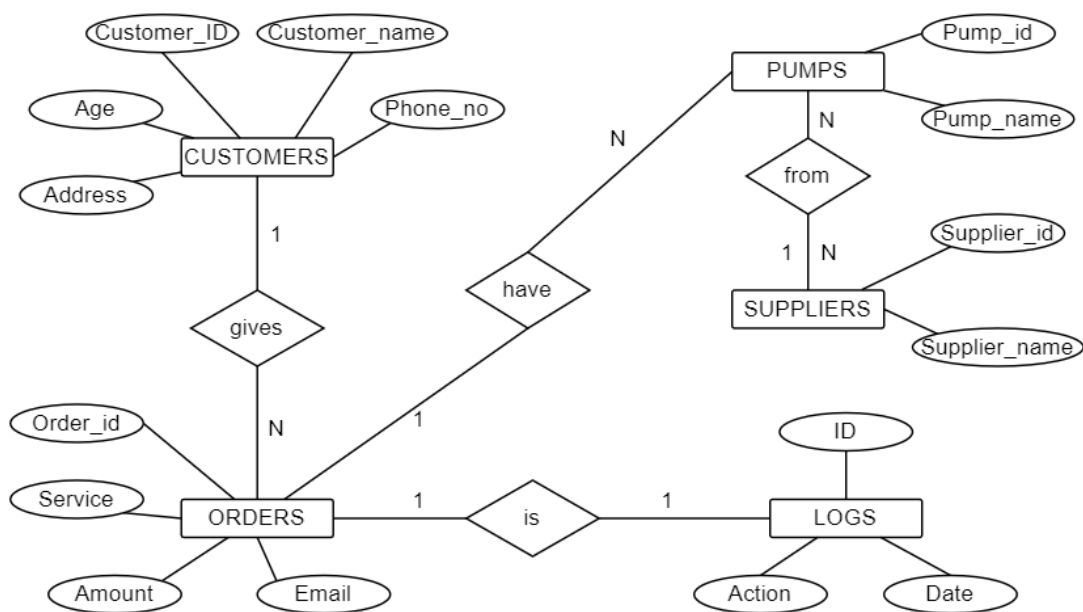


Fig.3.1. ER Diagram of the water pumps company database management system.

3.2 ER to Relation Mapping

The Entity-Relationship (ER) diagram serves as a graphical representation of the relationships among various entities in our project. To translate this visual representation into a functional database design, we perform ER to Relation Mapping. This process involves creating

structured tables that represent entities, their attributes, and the relationships between them.

- **Mapping the regular entities**

The entities do not include the foreign and relational attributes. The relations that are created from the mapping of entity types are sometimes called entity relations because each tuple represents an entity instance.

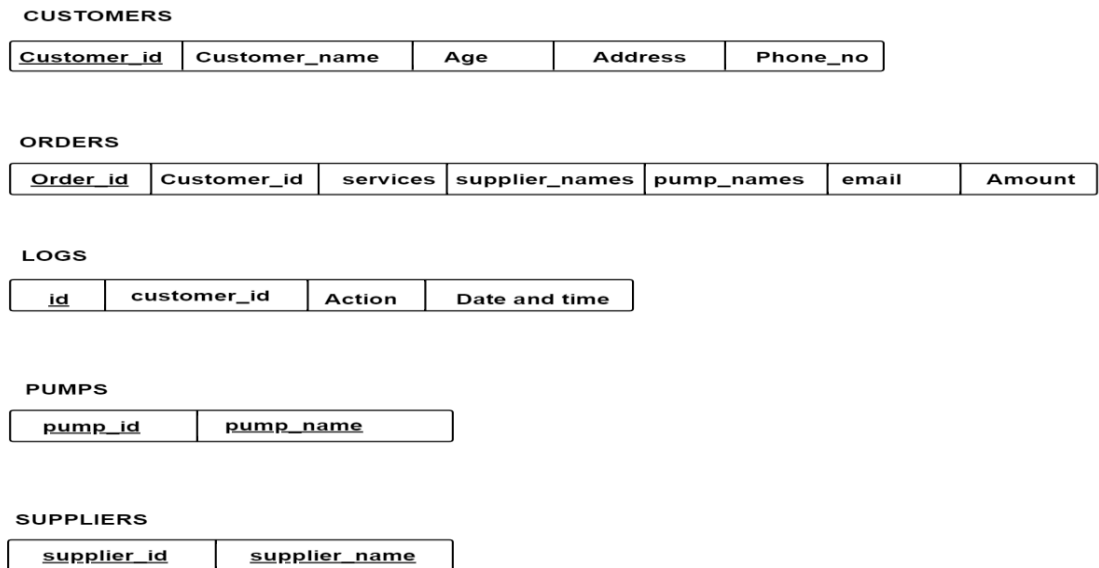


Fig.3.2. All the entities and attributes

- **Mapping of weak entity types**

We do not have weak entities in our ER Diagram. An entity is called "weak" when its existence depends on the existence of another (i.e., strong) entity.

- **Mapping of 1:1 cardinality relationships**

In a 1:1 relationship, one record in one table is associated with exactly one record in another table, and vice versa. In your project, you have a 1:1 relationship between the ORDERS table and the LOGS table. Each customer (record in the ORDERS table) can have one corresponding log entry (record in the LOGS table) to track their actions, and each log entry is associated with a specific customer.

- **Mapping of 1:N cardinality Relationship (One-to-Many)**

A 1:N relationship signifies that one record in a table is related to multiple records in another table. In your project, you have several 1:N relationships:

Each customer (record in CUSTOMERS) can place multiple orders (records in ORDERS), but each order is associated with only one customer. This represents a 1:N relationship between CUSTOMERS and ORDERS.

Similarly, each supplier (record in SUPPLIER) can supply multiple orders (records in ORDERS), but each order is supplied by one supplier. This is another 1:N relationship between SUPPLIER and ORDERS.

- **Mapping of M:N cardinality Relationship (Many-to-Many)**

In your specific project schema, there is no explicit M:N relationship. An M:N relationship would occur if multiple records in one table were associated with multiple records in another table. For example, if each customer could place multiple orders, and each order could be associated with multiple customers, that would constitute an M:N relationship.

- **Foreign Keys and Relationships:**

The ORDERS table has foreign keys to both the SUPPLIER and PUMP tables, creating a connection between orders, suppliers, and pumps. This reflects the concept that each order is tied to a specific supplier and pump.

The LOGS table has a foreign key to the CUSTOMERS table, establishing a link between log entries and the customers who performed the actions.

3.3 Schema Diagram

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and the relations among them are associated. It formulates all the constraints that are to be applied on data. A database schema defines its entities and relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It shows the various relations, references between entities. The arrows point towards the primary key and the other side will be the foreign keys in the relations respectively. The underlined attributes are the primary keys of the relations

respectively. They are used to uniquely identify the rows of a table. Thus, a row that needs to be uniquely identified, the key constraint is set as the Primary key to that field.

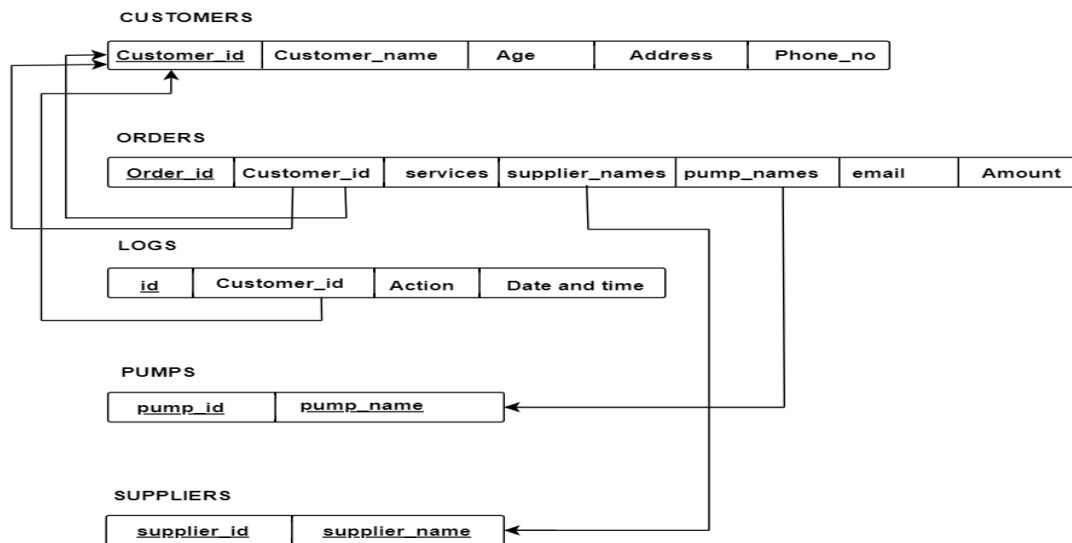


Fig. 3.3. Schema Diagram

3.4 Over view of GUI

A GUI is a programme interface that uses the visual capabilities of the computer to simplify the usage of the programme. An effective graphical user interface can save the user from having to learn difficult command languages. On the other hand, many users discover that a command-driven interface helps them work more productively, particularly if they are already familiar with the command language.

In the field of human-computer interaction, designing a GUI's visual layout and temporal behaviour is a crucial component of software application programming. Its objective is to improve a stored program's underlying logical design.

- The markup language used to create web pages and web applications is called HTML (Hypertext Markup Language). Together with JavaScript, they make up a trio of key web-enabling technologies. Web browsers transform HTML documents into multimedia web pages after receiving them from a web server or local storage. HTML originally featured signals for the appearance of the content and semantically explains the structure of a web page.

- Cascading Style Sheets (CSS) is a language for creating style sheets that describe how a document presented in a markup language like HTML. Along with HTML and JavaScript, CSS is a key component of the World Wide Web. Layout, colour, and font choices can all be differentiated between presentation and content using CSS. By specifying the pertinent CSS in a separate.css file, multiple web pages can share formatting, which can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, and reduce complexity and repetition in the structural content.

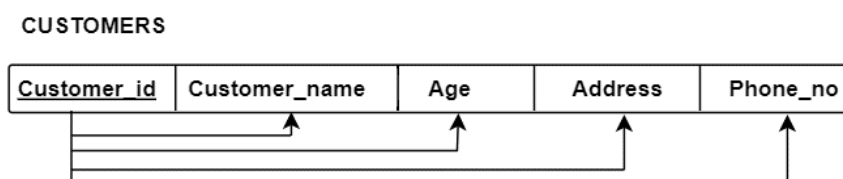
3.5 Normalization

Normalization is a multi-step process that involves breaking down a relational database into smaller, well-organized tables to eliminate data redundancy and anomalies. In your project, this process ensures that data is stored efficiently, improving data integrity and overall system performance.

Customers Table:

- Attributes: customer_id, customer_name, age, phone_no, address
- Primary Key: customer_id
- Functional Dependencies:

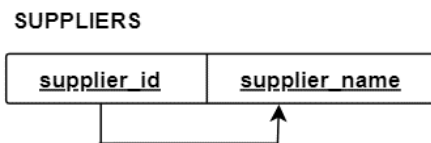
Customer ID → Customer Name, Age, Phone Number, Address



Suppliers Table:

- Attributes: supplier_id, supplier_name
- Primary Key: supplier_id
- Functional Dependencies:

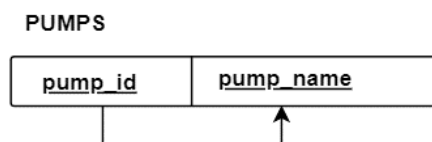
Supplier ID \rightarrow Supplier Name



Pump Table:

- Attributes: pump_id, pump_name
- Primary Key: pump_id
- Functional Dependencies:

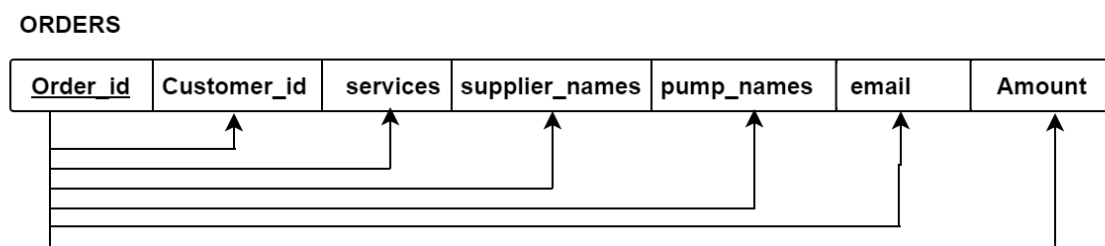
Pump ID \rightarrow Pump Name



Orders Table:

- Attributes: order_id, amount, supplier_names, pump_names, service, email, customer_id
- Primary Key: order_id
- Functional Dependencies:

Order ID \rightarrow Amount, Service, Supplier names, Pump names, Email, Customer ID



Chapter 4

IMPLEMENTATION

4.1 Table creation

```
CREATE TABLE customers (  
    customer_id INT AUTO_INCREMENT PRIMARY KEY,  
    customer_name VARCHAR(80) NOT NULL,  
    age INT NOT NULL,  
    phone_no VARCHAR(200) NOT NULL,  
    address VARCHAR(120) NOT NULL  
);
```

```
CREATE TABLE suppliers (  
    supplier_id INT AUTO_INCREMENT PRIMARY KEY,  
    supplier_name VARCHAR(500) NOT NULL  
);
```

```
CREATE TABLE pumps (  
    pump_id INT AUTO_INCREMENT PRIMARY KEY,  
    pump_name VARCHAR(200) NOT NULL  
);
```

```
CREATE TABLE orders (  
    order_id INT AUTO_INCREMENT PRIMARY KEY,  
    amount INT NOT NULL,
```

```

service VARCHAR(500) NOT NULL,

email VARCHAR(120) NOT NULL,

customer_id INT,

supplier_names VARCHAR(500) NOT NULL,

pump_names VARCHAR(200) NOT NULL,

FOREIGN KEY (customer_id) REFERENCES customers(customer_id),

FOREIGN KEY (supplier_name) REFERENCES supplier(supplier_name),

FOREIGN KEY (pump_name) REFERENCES pump(pump_name)

);

CREATE TABLE logs (

id INT AUTO_INCREMENT PRIMARY KEY,

customer_id INT,

action VARCHAR(30) NOT NULL,

date VARCHAR(100) NOT NULL,

FOREIGN KEY (customer_id) REFERENCES customers(customer_id)

);

```

4.2 Description of tables

desc customers;

















#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 customer_id 	int(11)			No	None		AUTO_INCREMENT	 Change  Drop  More
<input type="checkbox"/>	2 customer_name	varchar(100)	utf8mb4_general_ci		No	None			 Change  Drop  More
<input type="checkbox"/>	3 age	int(11)			No	None			 Change  Drop  More
<input type="checkbox"/>	4 phone_no	varchar(20)	utf8mb4_general_ci		No	None			 Change  Drop  More
<input type="checkbox"/>	5 address	varchar(50)	utf8mb4_general_ci		No	None			 Change  Drop  More

Fig. 4.2.1 Description of customers table

Water Pumps Company Management System

desc supplier;








#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 supplier_id	 int(11)			No	None		AUTO_INCREMENT	 Change  Drop  More
<input type="checkbox"/>	2 supplier_name	varchar(500)	utf8mb4_general_ci		No	None			 Change  Drop  More

Fig. 4.2.2 Description of supplier table

desc pumps;






#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 pump_id	 int(11)			No	None		AUTO_INCREMENT	 Change  Drop  More
<input type="checkbox"/>	2 pump_name	varchar(200)	utf8mb4_general_ci		No	None			 Change  Drop  More

Fig. 4.2.3 Description of pumps table

desc orders;























#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 order_id	 int(11)			No	None		AUTO_INCREMENT	 Change  Drop  More
<input type="checkbox"/>	2 amount	int(11)			No	None			 Change  Drop  More
<input type="checkbox"/>	3 supplier_names	varchar(100)	utf8mb4_general_ci		No	None			 Change  Drop  More
<input type="checkbox"/>	4 pump_names	varchar(500)	utf8mb4_general_ci		No	None			 Change  Drop  More
<input type="checkbox"/>	5 service	varchar(500)	utf8mb4_general_ci		No	None			 Change  Drop  More
<input type="checkbox"/>	6 email	varchar(50)	utf8mb4_general_ci		No	None			 Change  Drop  More
<input type="checkbox"/>	7 customer_id	varchar(50)	utf8mb4_general_ci		No	None			 Change  Drop  More

Fig. 4.2.4 Description of orders table

desc logs;














#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 id	 int(11)			No	None		AUTO_INCREMENT	 Change  Drop  More
<input type="checkbox"/>	2 customer_id	int(11)			No	None			 Change  Drop  More
<input type="checkbox"/>	3 action	varchar(500)	utf8mb4_general_ci		No	None			 Change  Drop  More
<input type="checkbox"/>	4 date	varchar(500)	utf8mb4_general_ci		No	None			 Change  Drop  More

Fig. 4.2.5 Description of logs table

4.3 Populated Tables

Select * from customers;

		customer_id	customer_name	age	phone_no	address
<input type="checkbox"/>	Edit Copy Delete	1000	anitha	35	7236872387	bangalore
<input type="checkbox"/>	Edit Copy Delete	1001	Pavithra M	20	8748007123	Mangalore
<input type="checkbox"/>	Edit Copy Delete	1002	Gayathri K R	47	07550236039	Chennai
<input type="checkbox"/>	Edit Copy Delete	1004	Kavitha	45	987654321	Bangalore
<input type="checkbox"/>	Edit Copy Delete	1005	GAYATHRI K R	56	07550236039	#380/1 10th cross
<input type="checkbox"/>	Edit Copy Delete	1007	jyothi	40	9740930889	Udupi
<input type="checkbox"/>	Edit Copy Delete	1008	om	23	6783544536	Kolkata
<input type="checkbox"/>	Edit Copy Delete	1009	srihith	20	8776667287	bangalore

Fig 4.3.1 Values in customers table

Select * from suppliers;

		supplier_id	supplier_name
<input type="checkbox"/>	Edit Copy Delete	12	KSB
<input type="checkbox"/>	Edit Copy Delete	13	Suguna
<input type="checkbox"/>	Edit Copy Delete	14	Ocleg
<input type="checkbox"/>	Edit Copy Delete	15	Jelco
<input type="checkbox"/>	Edit Copy Delete	16	Texmo
<input type="checkbox"/>	Edit Copy Delete	17	Finolex
<input type="checkbox"/>	Edit Copy Delete	18	dec

Fig 4.3.2 Values in supplier table

Select * form pumps;

		pump_id	pump_name
<input type="checkbox"/>	Edit Copy Delete	6	Submersible
<input type="checkbox"/>	Edit Copy Delete	7	MonoBloc
<input type="checkbox"/>	Edit Copy Delete	8	Sewagepump
<input type="checkbox"/>	Edit Copy Delete	9	3HP/50phase
<input type="checkbox"/>	Edit Copy Delete	10	1HP
<input type="checkbox"/>	Edit Copy Delete	11	borewell pump
<input type="checkbox"/>	Edit Copy Delete	12	borewell pump

Fig 4.3.3 Values in pumps table

select * from orders;













				order_id	amount	supplier_names	pump_names	service	email	customer_id
<input type="checkbox"/>	 Edit	 Copy	 Delete	8	40000	KSB	Submersible 1HP	installation	kvkavitha988@gmail.com	1002
<input type="checkbox"/>	 Edit	 Copy	 Delete	11	45000	Jelco	submersible	installation	21aiml025@bnmit.in	1004
<input type="checkbox"/>	 Edit	 Copy	 Delete	12	45000	KSB	submersible MonoBloc 1HP	transportation	pavithra863m@gamil.com	1001
<input type="checkbox"/>	 Edit	 Copy	 Delete	13	56000	KSB	submersible	installation	pavithra368m@gmail.com	1002

Fig 4.3.4 Values in orders table

select * from logs;
























				id	customer_id	action	date
<input type="checkbox"/>	 Edit	 Copy	 Delete	13	1002	INSERTED	2023-08-16 00:17:05
<input type="checkbox"/>	 Edit	 Copy	 Delete	14	1003	INSERTED	2023-08-16 00:17:48
<input type="checkbox"/>	 Edit	 Copy	 Delete	15	1002	INSERTED	2023-08-16 00:18:33
<input type="checkbox"/>	 Edit	 Copy	 Delete	16	1004	INSERTED	2023-08-16 00:19:12
<input type="checkbox"/>	 Edit	 Copy	 Delete	17	1001	INSERTED	2023-08-16 00:20:30
<input type="checkbox"/>	 Edit	 Copy	 Delete	18	1002	DELETED	2023-08-16 00:22:36
<input type="checkbox"/>	 Edit	 Copy	 Delete	19	1003	DELETED	2023-08-16 01:56:57
<input type="checkbox"/>	 Edit	 Copy	 Delete	20	1002	INSERTED	2023-08-16 10:01:44

Fig 4.3.5 Values in logs table

4.4. SQL Triggers and Stored Procedures

Triggers

A database trigger is procedural code that is automatically executed in response to certain events on a particular table or view in a database when a DML event occurs.

- **Delete Trigger:**

Trigger Name: Delete

Event: BEFORE DELETE on table orders

Purpose: This trigger is executed before a record is deleted from the orders table. It inserts a log entry into the Logs table, capturing the action of a record being deleted and

the associated customer_id and timestamp.

- **Insert Trigger:**

Trigger Name: Insert

Event: AFTER INSERT on table orders

Purpose: This trigger is executed after a new record is inserted into the orders table. It inserts a log entry into the Logs table, capturing the action of a record being inserted and the associated customer_id and timestamp.

- **Update Trigger:**

Trigger Name: Update

Event: AFTER UPDATE on table orders

Purpose: This trigger is executed after a record in the orders table is updated. It inserts a log entry into the Logs table, capturing the action of a record being updated and the associated customer_id and timestamp.

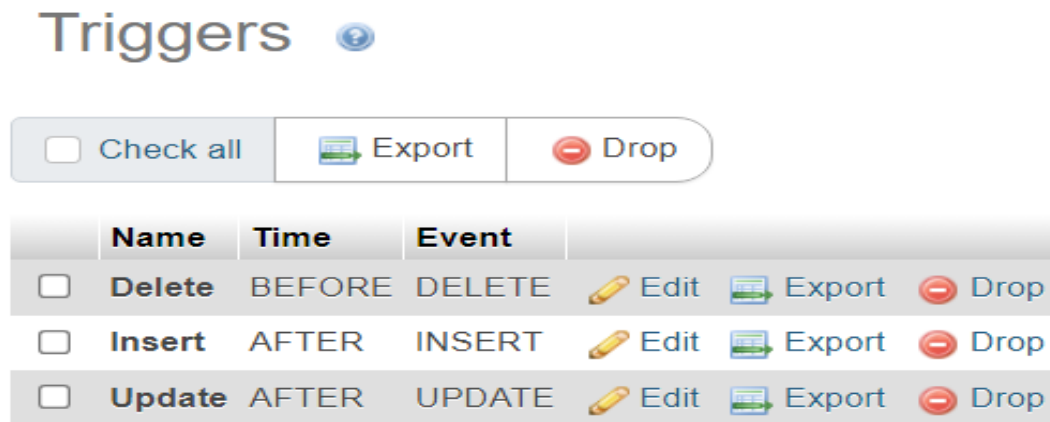


Fig 4.4. Trigger

Stored Procedure

We have used SQLAlchemy to execute raw SQL queries, which include the calls to the stored procedures. Here are the relevant sections of your code where you interact with the stored procedures:

Displaying Orders using Stored Procedure (list route):

We retrieve orders for a specific customer using a stored procedure named GetOrdersForCustomer:

```
@app.route("/list", methods=['GET', 'POST'])

def list():

    if 'user' in session and session['user'] == params['user']

        # Assuming you have a stored procedure named 'GetOrdersForCustomer'

        customer_id = get_customer_id() # Your logic to retrieve customer ID

        orders      =      db.session.execute("CALL      GetOrdersForCustomer(:customer_id)",
        {'customer_id': customer_id})

        return render_template('post.html', params=params, orders=orders)
```

Saving Orders using Stored Procedure (supplier_name route):

We save orders using a stored procedure named InsertOrder:

```
@app.route("/supplier_name", methods=['GET', 'POST'])

def supplier_name():

    if request.method == 'POST':

        # Assuming you have a stored procedure named 'InsertOrder'

        customer_id = request.form.get('customer_id')

        service = request.form.get('service')

        supplier_names = request.form.get('supplier_names')

        pump_names = request.form.get('pump_names')

        email = request.form.get('email')

        amount = request.form.get('amount')
```

```
query = "CALL InsertOrder(:customer_id, :service, :supplier_names, :pump_names,
:email, :amount)"

db.session.execute(query, {

    'customer_id': customer_id,

    'service': service,

    'supplier_names': supplier_names,

    'pump_names': pump_names,

    'email': email,

    'amount': amount

})

db.session.commit()

flash("Data Added Successfully", "primary")

return render_template('supplier_name.html', params=params)
```

In both cases, you are using SQLAlchemy's 'execute' method to call the stored procedures with the appropriate parameters. The stored procedures themselves should be defined and created directly in our MySQL database using SQL statements. The Python code then interacts with these stored procedures using SQLAlchemy's capabilities for executing raw SQL queries.

4.5 Database connectivity

A cornerstone of our project's functionality is the establishment of a robust and reliable database connectivity layer. This enables seamless communication between our web application and the MySQL database, facilitating efficient data management and retrieval.

Flask and SQLAlchemy Integration:

Database connectivity is established by configuring the SQLAlchemy extension within our Flask application. The database connection details, including the database URI, authentication credentials, and host information, are specified in the application's configuration.


```
app.config['SQLALCHEMY_DATABASE_URI']='mysql://username:password@host/database_name'
```

Creating the Database Instance:

We create a database instance using SQLAlchemy, which serves as a bridge between our application and the MySQL database. This instance manages the database sessions and facilitates data interactions.

```
from flask_sqlalchemy import SQLAlchemy
```

```
app = Flask(__name__)
```

```
app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql://username:password@host/database_name'
```

```
db = SQLAlchemy(app)
```

Executing Queries

Our project leverages the database connectivity to execute SQL queries. SQLAlchemy's execute method allows us to send SQL statements to the database, fetch results, and perform data manipulation.

```
#executing a raw SQL query
```

```
result = db.session.execute("SELECT * FROM orders WHERE customer_id = :customer_id",  
{ 'customer_id': customer_id })
```

Use returned data –

```
@app.route('/home')
```

```
def home():
```

```
    return render_template("home.html");
```

4.6 Modules

The below flowchart explains how the system runs in the real world. The system can easily be implemented under various situations. Reusability is possible as and when required in this

application. There is feasibility in all the modules which makes task of the user easier.

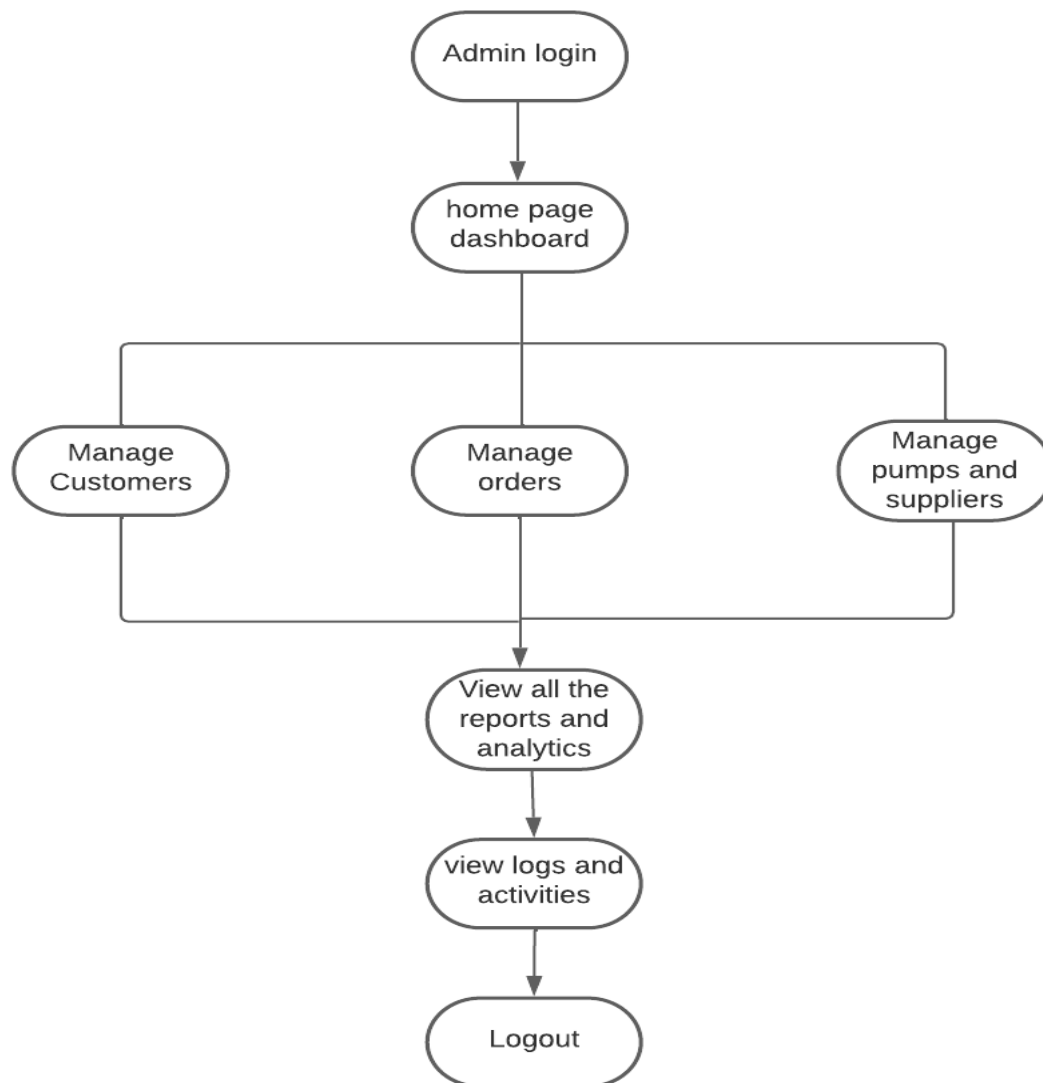


Fig 4.6. Modules and dataflow design

Chapter 5

RESULTS

This chapter contains GUI built using CSS and HTML. The screenshots contains various html pages.

Login Page:

If correct password is entered it will login to home page or else is will display wrong password.

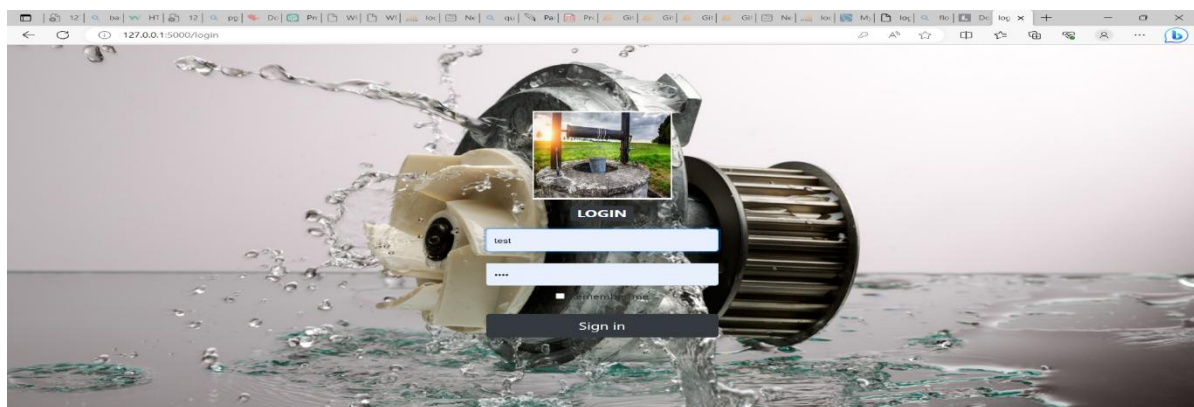


Fig 5.1. Login Page

Dashboard Page:

Once logged in it will take us to the home page.

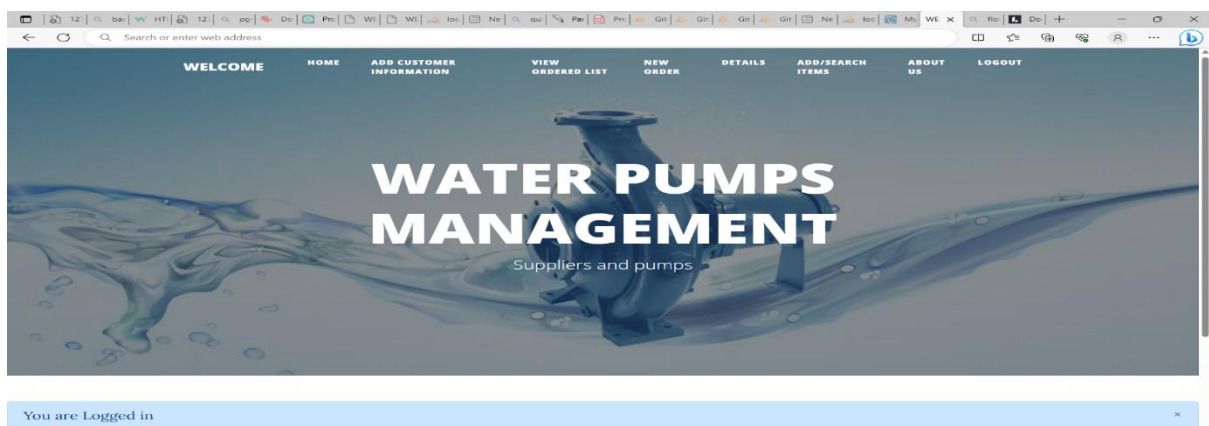


Fig 5.2. Dashboard Page

Home:

Once we click home the pages which views the info of the customers is displayed it is editable and deleteable.

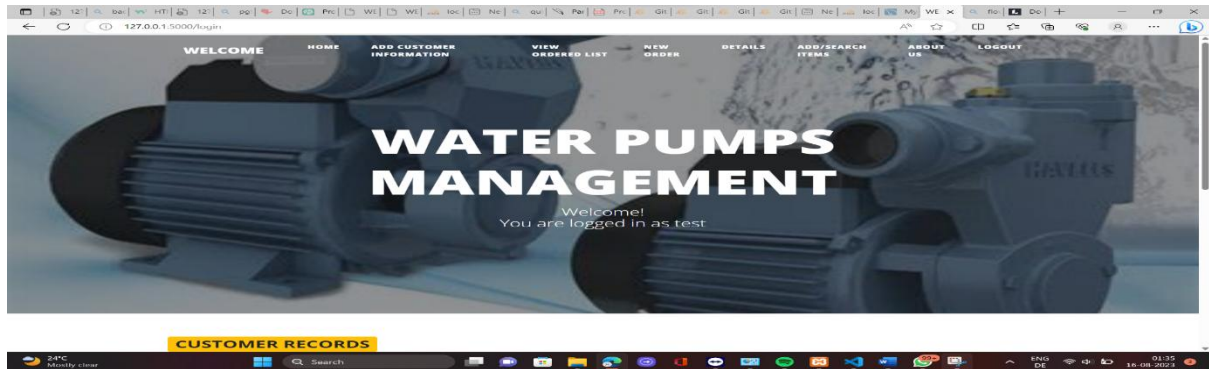


Fig 5.3. Home page (1)

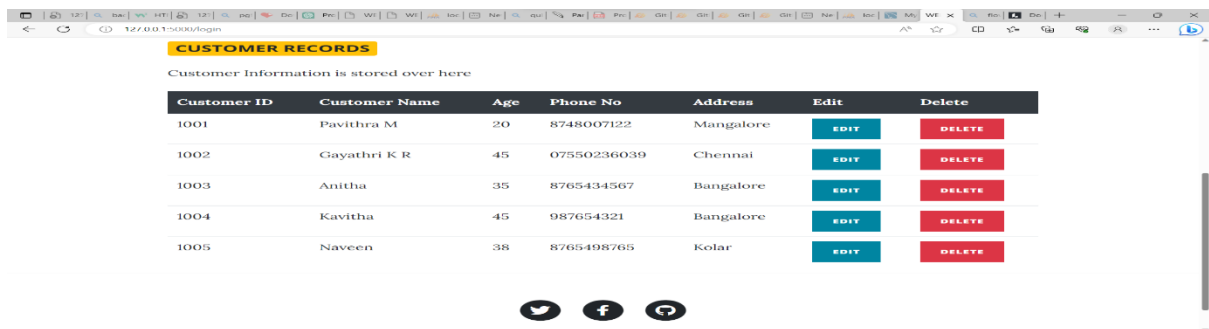


Fig 5.4. home page (2) with customer information

Add Customers information:

Once we click this in the status a page appears where we can add new customers and it gets stored and displayed on the home page again.

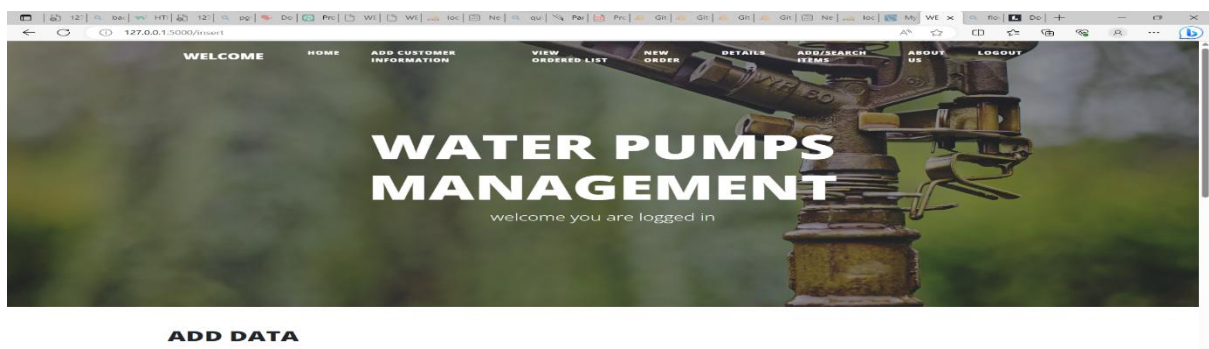


Fig 5.5. Add customer information (1)

The screenshot shows a web browser window with the URL 127.0.0.1:5000/insert. The page has a title 'ADD DATA'. Below the title are five input fields: 'enter Customer id', 'enter customer name', 'Enter age', 'Enter phone number', and 'enter Address'. At the bottom of the form is a blue button labeled 'INSERT DATA'. Below the form are three social media icons: Twitter, Facebook, and YouTube.

Fig 5.6. Add customer information (2)

View Orders:

Once we click this the orders list will be viewed and the orders can be deleted as well.

The screenshot shows a web browser window with the URL 127.0.0.1:5000/list. The page has a header with a navigation menu: WELCOME, HOME, ADD CUSTOMER INFORMATION, VIEW ORDERED LIST, NEW ORDER, DETAILS, ADD/SEARCH ITEMS, ABOUT US, and LOGOUT. The main content area has a background image of a water pump and the text 'View Customers Orders' and 'order records'. Below this is a table with the following data:

Customer ID	Supplier Name	Pump Name	Amount	Delete
1002	KSB	Submersible IHP	40000	DELETE

Fig 5.7. View customer Orders (1)

The screenshot shows a web browser window with the URL 127.0.0.1:5000/list. The page has a header with a navigation menu: WELCOME, HOME, ADD CUSTOMER INFORMATION, VIEW ORDERED LIST, NEW ORDER, DETAILS, ADD/SEARCH ITEMS, ABOUT US, and LOGOUT. The main content area has a background image of a water pump and the text 'View Customers Orders' and 'order records'. Below this is a table with the following data:

Customer ID	Supplier Name	Pump Name	Amount	Delete
1002	KSB	Submersible IHP	40000	DELETE
1003	Ocleg	MonoBloc	56000	DELETE
1004	Jelco	submersible	45000	DELETE
1001	KSB	submersible MonoBloc IHP	45000	DELETE

Fig 5.8. view customer orders (2)

New Order:

In this we can add the new orders.

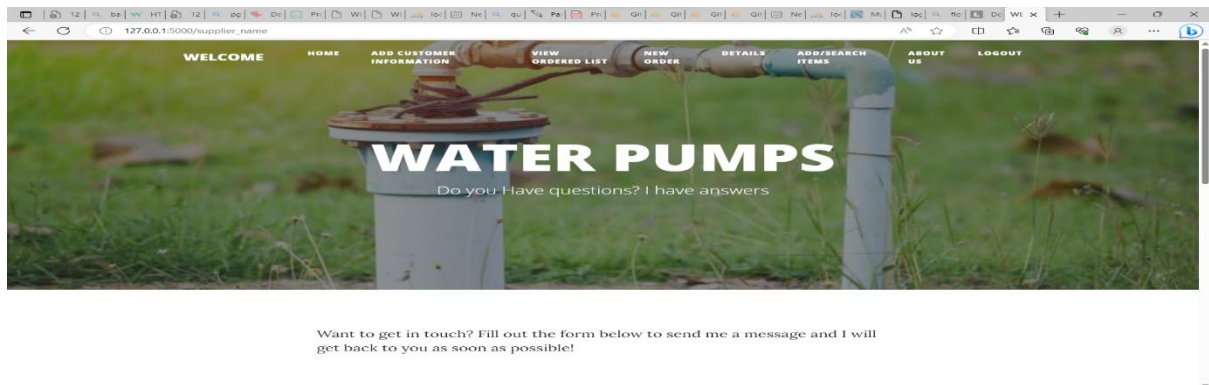


Fig 5.9. new order (1)

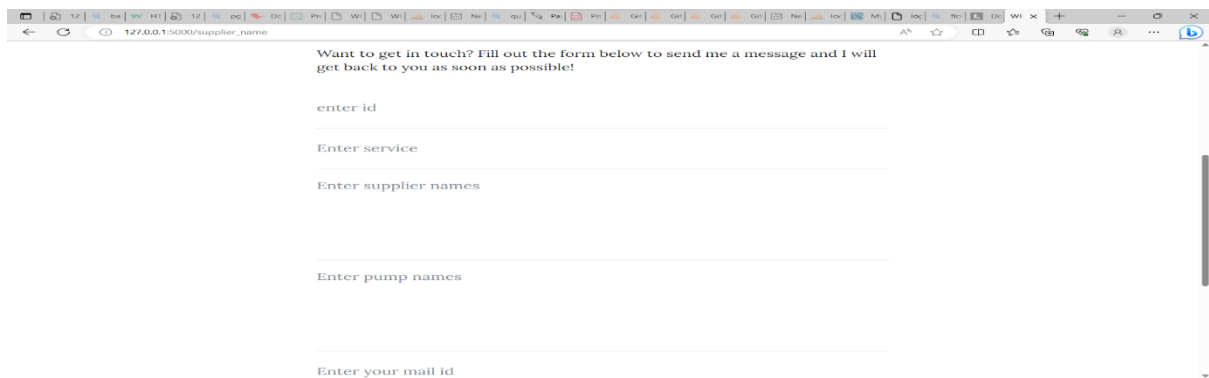


Fig 5.10. new order (2)

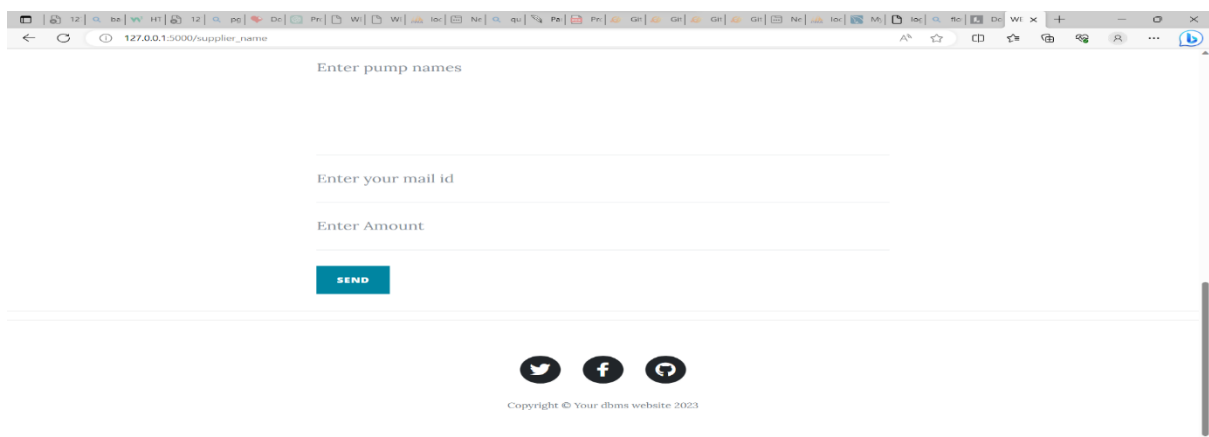


Fig 5.11. new order (3)

Details:

When we go to this we can the logs of the order that whether it is inserted or deleted and at what time it is done. There is also a button of stored details when we click it views the orders using stored procedure.

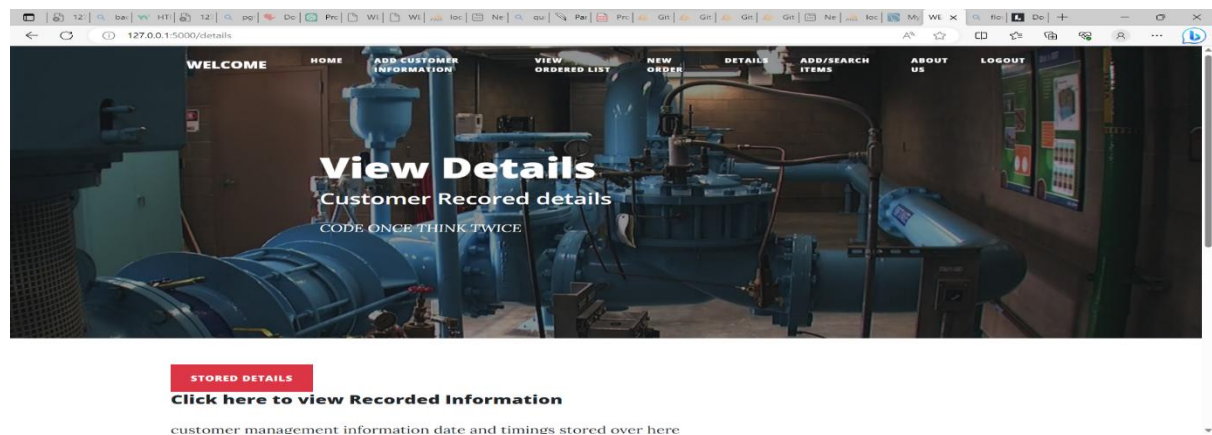


Fig 5.12. details (1)

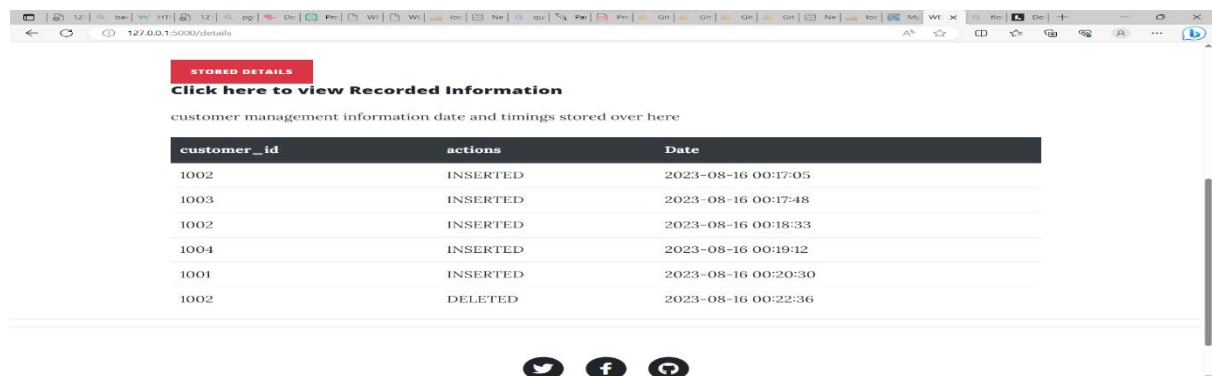


Fig 5.13. details (2)(trigger)

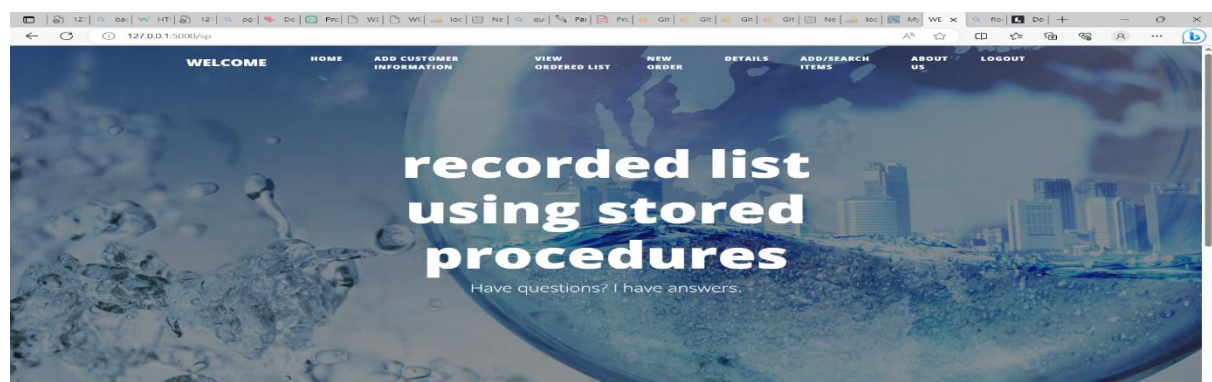
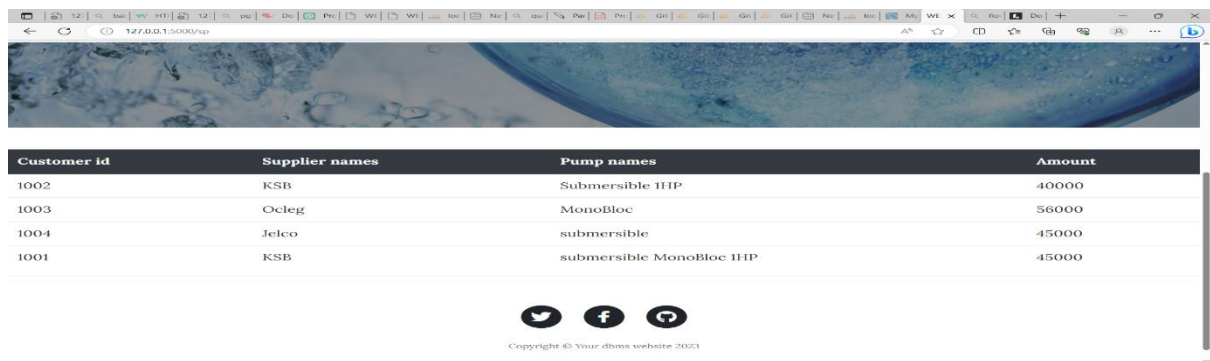


Fig 5.14. stored details(stored procedure)(1)



Customer id	Supplier names	Pump names	Amount
1002	KSB	Submersible IHP	40000
1003	Ocleg	MonoBloc	56000
1004	Jelco	submersible	45000
1001	KSB	submersible MonoBloc IHP	45000

Copyright © Your dhms website 2023

Fig 5.15. stored details (2)

Add/Search items:

In this We can add the new suppliers and new pumps, we can view the list of suppliers and pumps and we can also search for items. If the item is found it will show item found or it will show item not found.

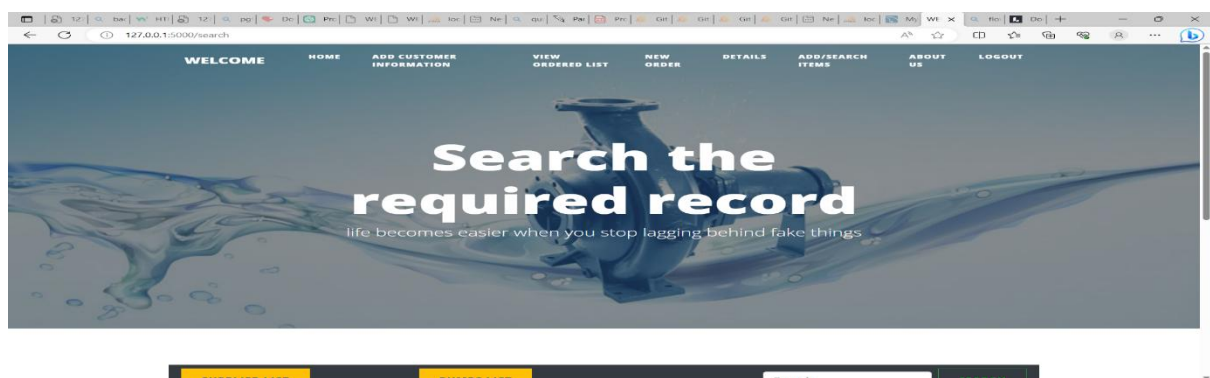


Fig 5.16.Add/search items(1)

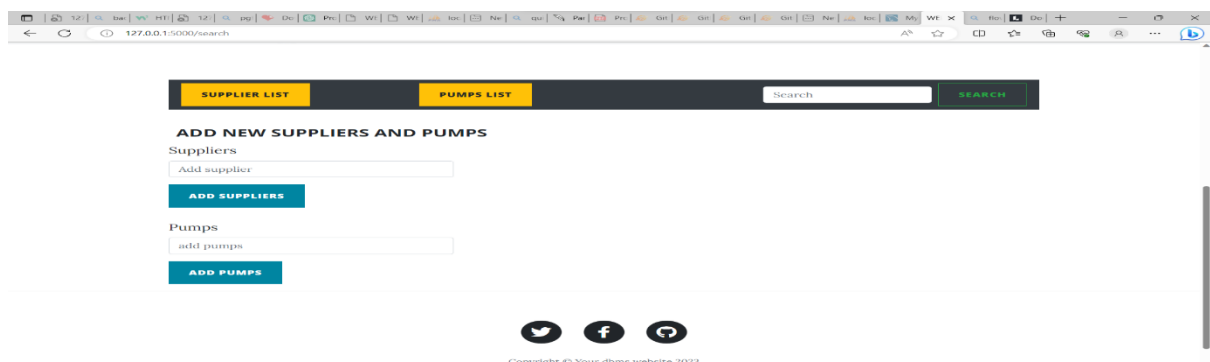


Fig 5.17. Add/search items(2)

Water Pumps Company Management System

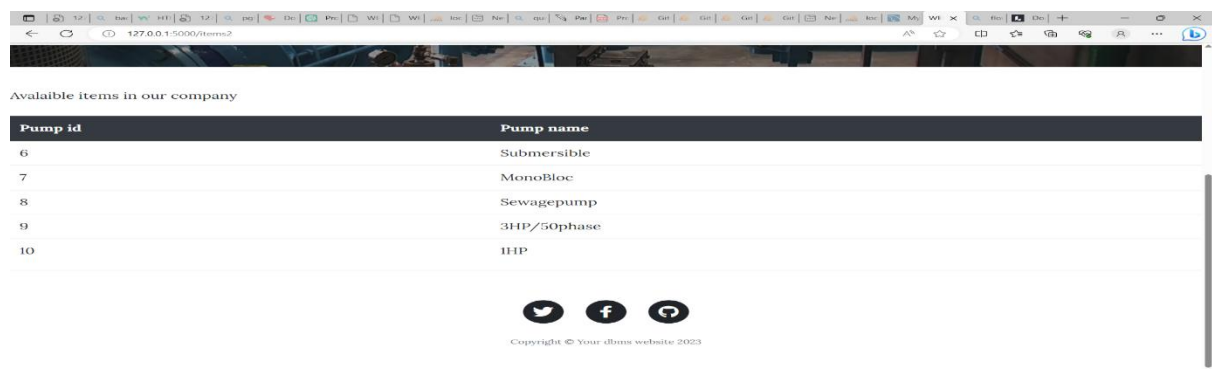


Fig 5.18. pumps list

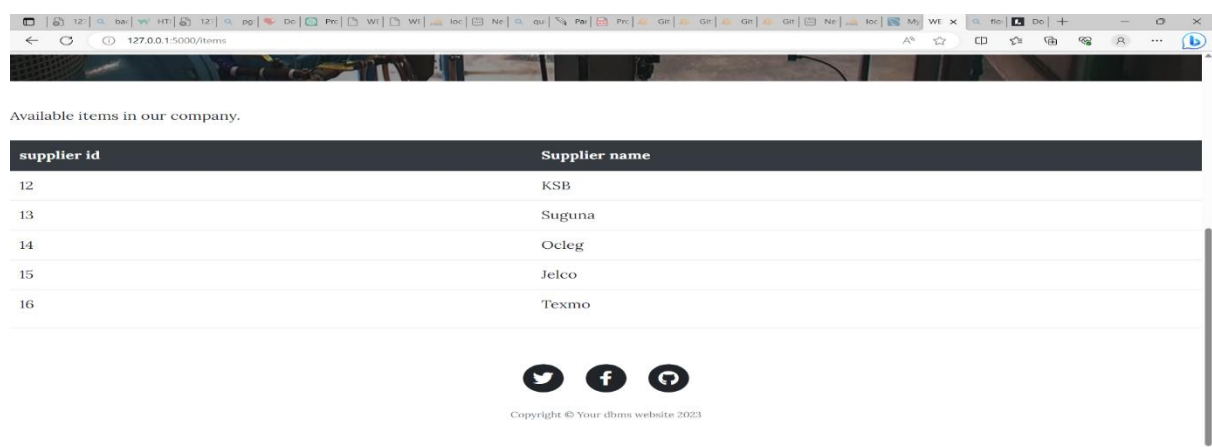


Fig 5.19. suppliers list

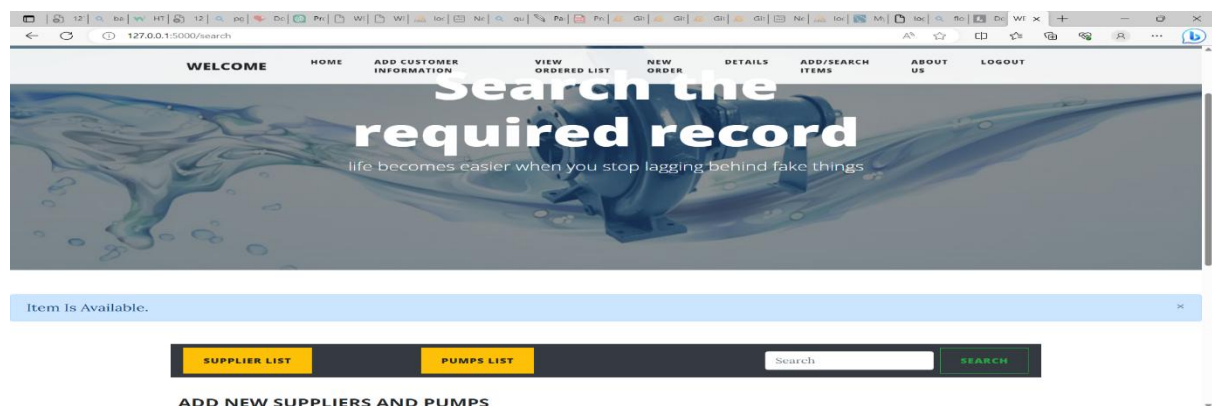


Fig 5.20. searching

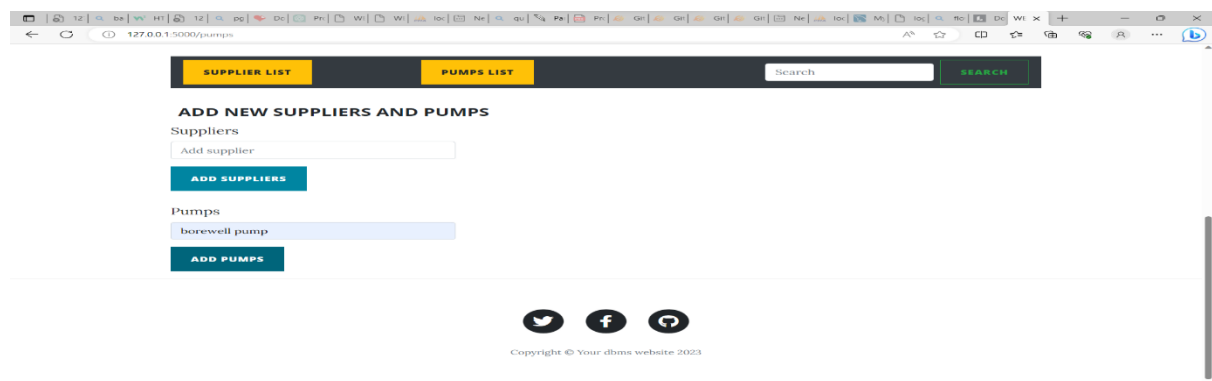


Fig 5.21. Adding

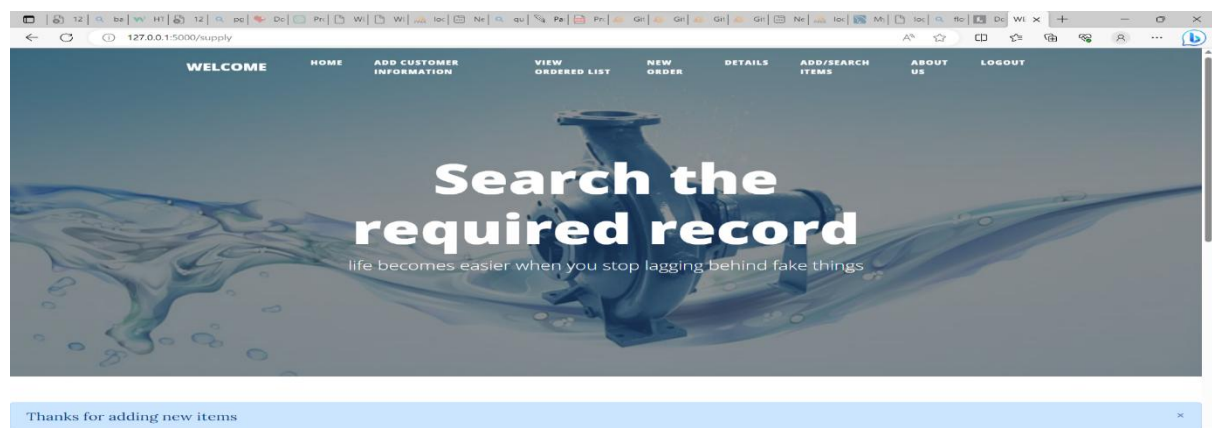


Fig 5.22. pump added

About us:

In this the information about the company is displayed.

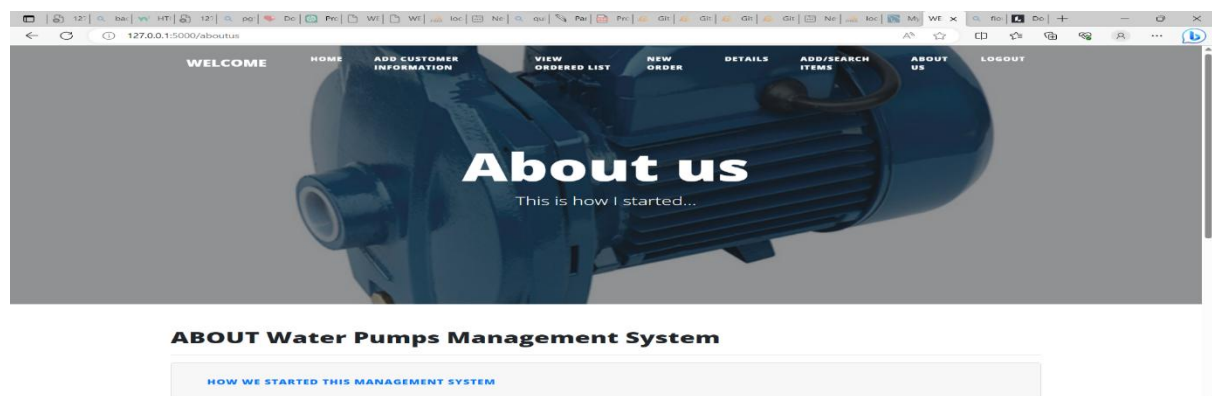


Fig 5.23. About us(1)

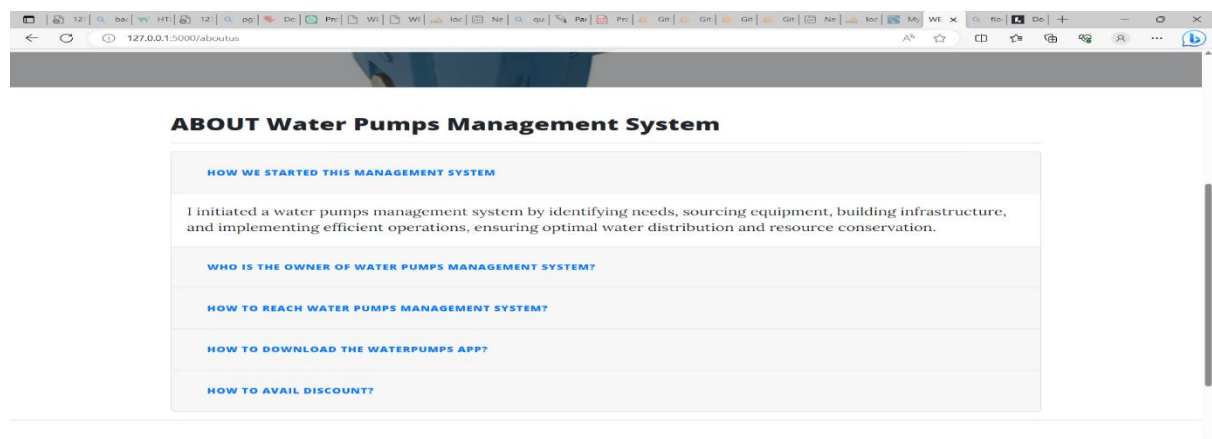


Fig 5.24. About us(2)

Logout:

When this is Clicked it takes us back to the login page.

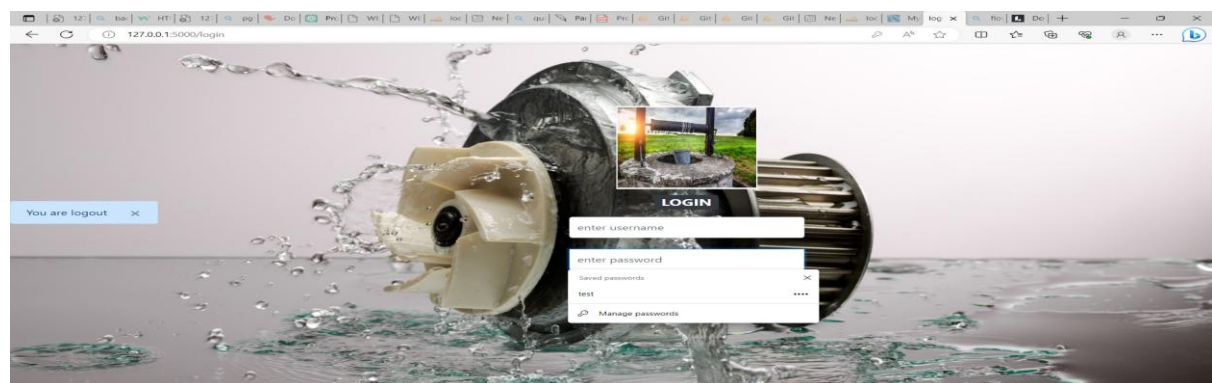


Fig 5.25. Logged out

Chapter 6

CONCLUSION AND FUTURE ENHANCEMENTS

6.1 Conclusion

In conclusion, the development of the admin-based project has resulted in the creation of a robust and user-friendly system that empowers administrators to efficiently manage and oversee various aspects of the business. The project's core features, including user authentication, supplier and pump management, order tracking, and comprehensive logs, have been successfully implemented to streamline administrative tasks and enhance transparency.

The project's architecture, which leverages Flask for web application development and SQLAlchemy for database management, has proven effective in delivering a seamless and intuitive user experience. The utilization of SQL triggers and stored procedures further enhances data integrity and provides valuable insights into system activities.

Through the development and implementation process, valuable skills in web development, database management, and user interface design have been acquired.

6.2 Future Enhancements

Looking ahead, our project presents exciting prospects for future enhancements aimed at further optimizing its capabilities. Integration of real-time analytics promises to deliver instantaneous insights into sales trends and performance metrics. The incorporation of automated alerts will ensure administrators stay promptly informed of critical events, enhancing responsiveness. Furthermore, the potential addition of a mobile app will extend the convenience of on-the-go management, while user feedback will provide invaluable insights for refining the user interface and overall user experience. Emphasis on security enhancements, exploration of integration options, introduction of multilingual support, and implementation of role-based access control will collectively solidify the project's adaptability, security, and user satisfaction in a dynamic business landscape.

References

[1] IEEE Xplore: Search for papers and articles related to "Database Management Systems" and "Project Development."

Link: <https://ieeexplore.ieee.org/>

[2] ACM Digital Library: Look for relevant papers, articles, and resources on database systems and software development.

Link: <https://dl.acm.org/>

[3] Google Scholar: Search for academic papers and articles using keywords like "single-user admin-based DBMS project."

Link: <https://scholar.google.com/>

[4] TechCrunch: Explore articles on technology trends, software development, and project management.

Link: <https://techcrunch.com/>

[5] TechRepublic: Look for articles discussing database management systems and project development.

Link: <https://www.techrepublic.com/>

[6] Elmasri, R., & Navathe, S. B. (2016). Fundamentals of Database Systems. Pearson.

[7] Connolly, T. M., & Begg, C. E. (2014). Database Systems: A Practical Approach to Design,

Implementation, and Management. Pearson.

[8] Coronel, C., Morris, S., & Rob, P. (2015). Database Systems: Design, Implementation, & Management. Cengage Learning.

[9] Date, C. J. (2003). An Introduction to Database Systems. Addison-Wesley.

[10] Ramakrishnan, R., & Gehrke, J. (2003). Database Management Systems. McGraw-Hill Education.