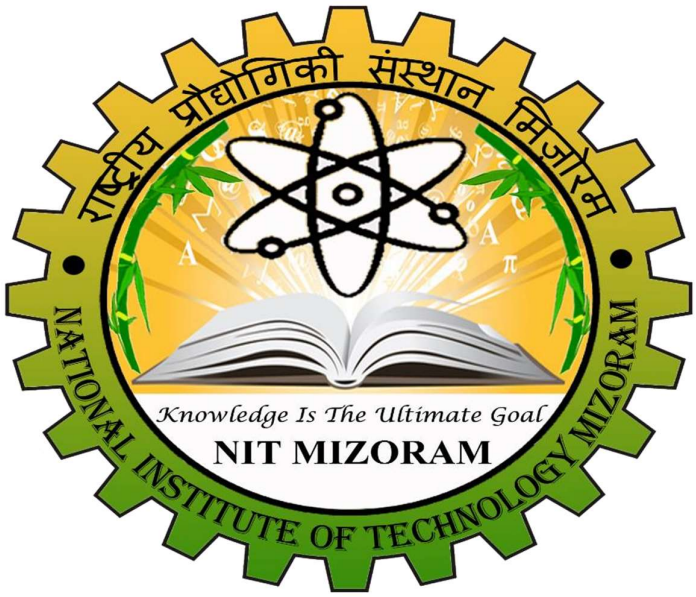
A DBMS PROJECT REPORT

ON AQUA WATER REFILING MANAGEMENT SYSTEM

A report submitted in the fulfilment of the requirement for B.Tech.



Submitted To

Dr. Ashish Singh Patel
Assistant Professor,
Department of Computer Science and Engineering
National Institute of Technology Mizoram

Submitted By

PRIYANSHU BARIYAR

BT22CS017 V SEM/3rd Year

DBMS

B.Tech., Department of Computer Science and Engineering National Institute of Technology Mizoram

Table of Contents

---

1. Introduction

The **Aqua Water Refiling Management System** is a web-based solution that organizes the operations of a water management company. It features a structured database to handle customers, products, orders, deliveries, payments, suppliers, and inventory, alongside role-based access for admins and basic users.

---

2. Objectives

- **Data Centralization**: Consolidate company data into a single relational database.
- **Access Control**: Implement role-based access to maintain security.
- **User-Friendly Interface**: Develop an intuitive web application using Flask.
- **Scalability and Efficiency**: Ensure the system can grow with the company.

---

3. System Overview

This system uses a **MySQL** database for data storage and **Flask** for the web interface. It allows users to perform CRUD operations on various entities, with permissions depending on their role (admin or basic user).

---

4. Database Design

The database schema is organized into several tables as outlined below.

4.1 Customer Table

- **Purpose**: To store customer information including contact details, type, and balance.
- **Schema**:
    - customerID (INT, Primary Key, Auto-Incremented)
    - first_name (VARCHAR(50))
    - middle_name (VARCHAR(50))
    - last_name (VARCHAR(50))
    - email (VARCHAR(50))
    - phone_number (BIGINT)
    - customer_type (VARCHAR(50))
    - balance (INT)
    - address (VARCHAR(255))

4.2 Product Table

- **Purpose**: To store product information, including prices and stock.
- **Schema**:
    - product_ID (INT, Primary Key)
    - product_name (VARCHAR(255))
    - price_per_unit (FLOAT)
    - stock_quantity (INT)

4.3 Delivery Person Table

- **Purpose**: To maintain records of delivery personnel and their availability status.
- **Schema**:
  - delivery_person_ID (INT, Primary Key)
  - person_name (VARCHAR(255))
  - phone_number (BIGINT)
  - availability (VARCHAR(255))

4.4 OrderDetails Table

- **Purpose**: To manage orders placed by customers, with links to delivery and payment.
- **Schema**:
  - orderID (INT, Primary Key)
  - customerID (INT, Foreign Key referencing Customer)
  - orderDate (DATETIME)
  - deliveryDate (DATETIME)
  - amount (FLOAT)
  - orderStatus (VARCHAR(50))

4.5 Delivery Table

- **Purpose**: To track deliveries associated with orders and delivery personnel.
- **Schema**:
  - delivery_ID (INT, Primary Key)
  - order_ID (INT, Foreign Key referencing OrderDetails)
  - del_date (DATETIME)
  - del_status (VARCHAR(255))
  - delivery_personID (INT, Foreign Key referencing Delivery Person)

4.6 Payment Table

- **Purpose**: To record payments for each order.
- **Schema**:
  - payment_ID (INT, Primary Key)
  - order_id (INT, Foreign Key referencing OrderDetails)
  - payment_date (DATETIME)
  - amount_paid (BIGINT)
  - method (VARCHAR(255))
  - status (VARCHAR(255))

4.7 Supplier Table

- **Purpose**: To manage supplier details and associated products.
- **Schema**:
  - supplier_ID (INT, Primary Key)
  - name (VARCHAR(255))
  - contact_number (BIGINT)
  - email (VARCHAR(255))
  - address (VARCHAR(255))
  - supplied_product_ID (INT, Foreign Key referencing Product)

4.8 Inventory Table

- **Purpose**: To manage inventory levels and reorder points.
- **Schema**:
  - inventory_ID (INT, Primary Key)
  - product_ID (INT, Foreign Key referencing Product)
  - quantity_available (FLOAT)
  - reorder_level (INT)
  - last_updated (DATETIME)

---

5. Application Design

The application uses Flask to provide a web-based user interface. Admin users have full access, while basic users can access limited functions.

- **Admin Functions**:
  - Create, update, delete, and view customers, orders, and products.
  - Manage deliveries and inventory.
- **Basic User Functions**:
  - View customer information, place orders, and access delivery updates.

---

6. Implementation

- **Languages/Frameworks**: Python (Flask), MySQL, HTML, CSS
- **Libraries Used**:
  - flask_mysqldb for database connectivity.
  - Flask for routing and rendering HTML templates.

---

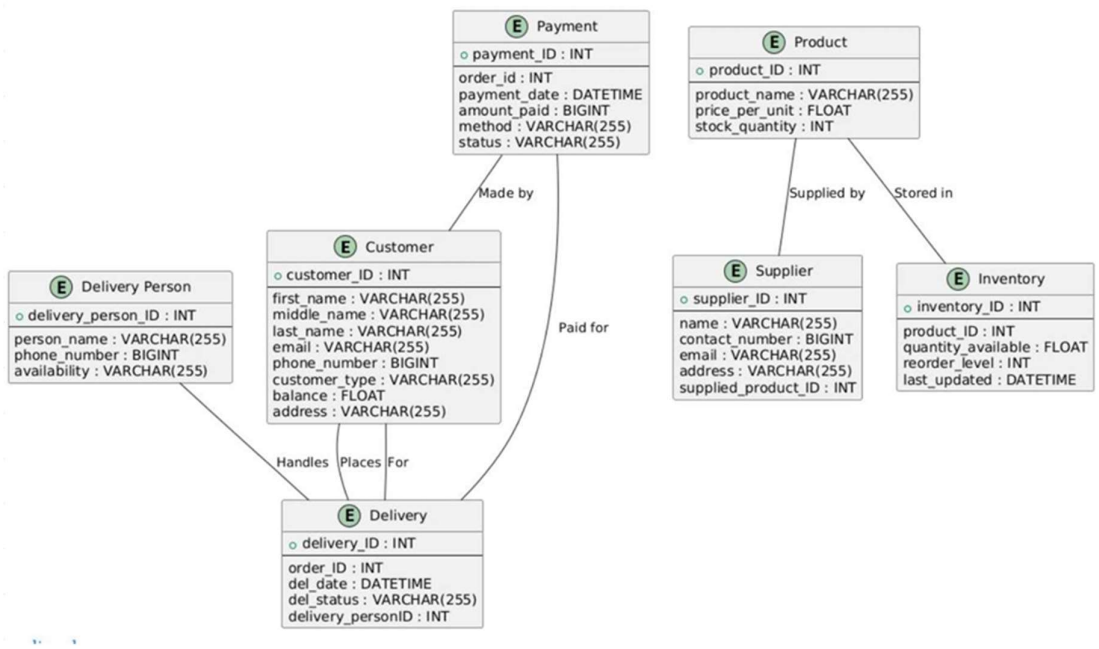
7. Testing

The system underwent unit testing for each feature:

- **Customer Management**: Add, view, and delete functions were tested.
- **Order and Delivery Management**: Verified the relational integrity and workflow between orders and deliveries.
- **Payments and Inventory**: Tested for accuracy and synchronization.

---

8. ER Diagram



- Ensure that each table's relationships and cardinalities are accurately represented.
- Key entities should be clearly identified, with primary and foreign keys noted.

---

9. Normalization

Each table in the Aqua Water Management System database is normalized to ensure data integrity and minimize redundancy. Below are the normalization levels:

9.1 First Normal Form (1NF)

- Each table contains atomic values, and there are no repeating groups.

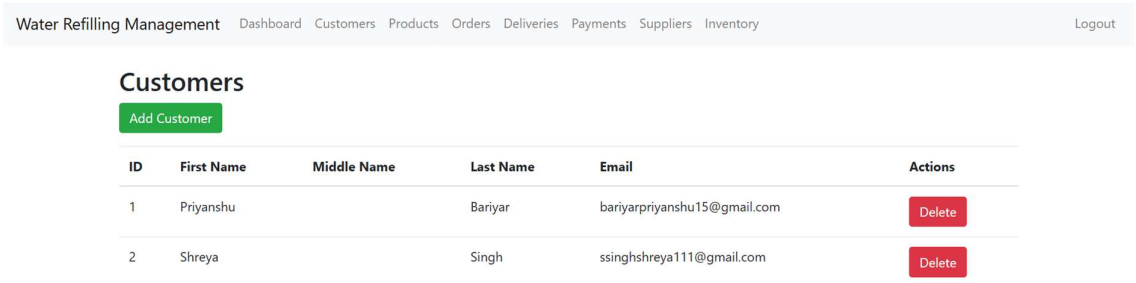
9.2 Second Normal Form (2NF)

- All tables meet 1NF requirements, and each non-key attribute is fully dependent on the primary key.

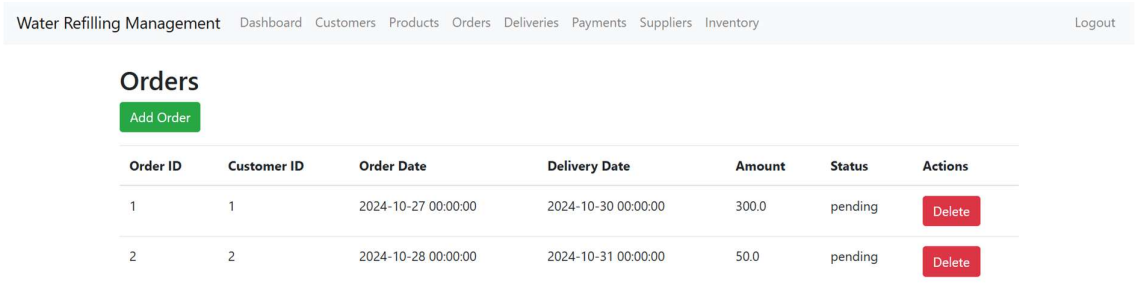
---

10. UI Screenshots

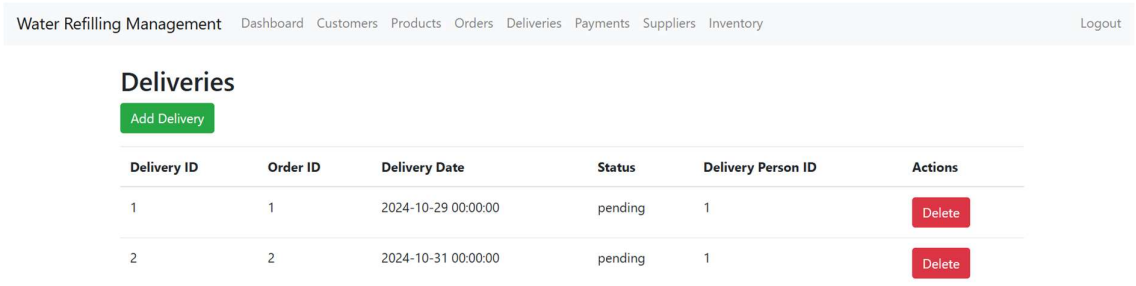
1. Customer Management Page



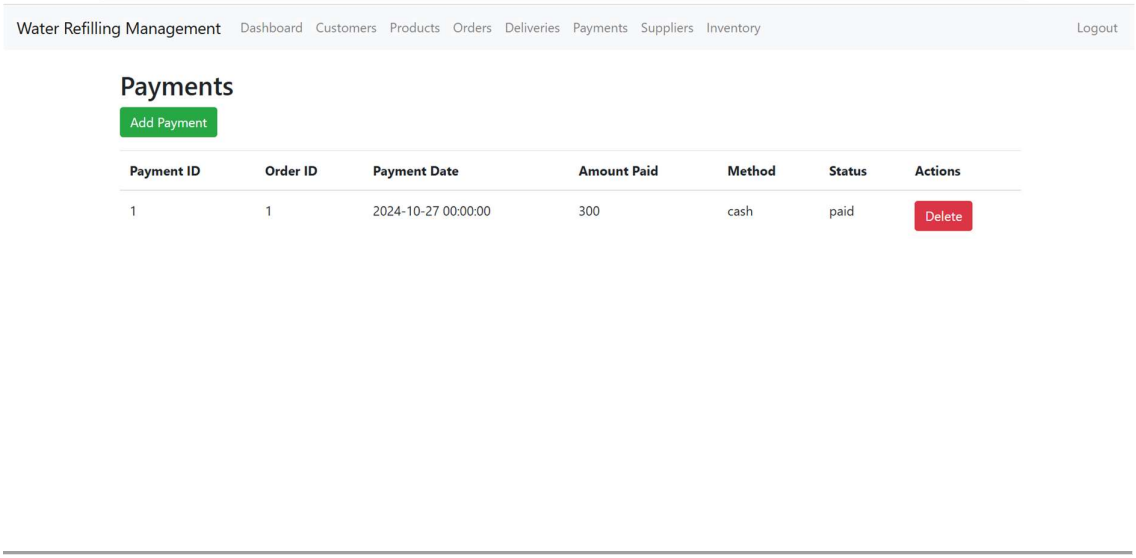
2. Order Management Page



3. Delivery Tracking Page



4. **Payment Processing Page**



11. Conclusion

The Aqua Water Management System successfully streamlines data management for a water distribution business. It provides a flexible solution with role-based access and a structured database schema.

---

12. Future Enhancements

- **Notifications**: Implement email or SMS notifications for order updates.
- **Reporting**: Add reporting tools to analyze sales and delivery performance.
- **Mobile Access**: Develop a mobile application interface for on-the-go access.

---

13. References

- Flask Documentation: https://flask.palletsprojects.com/
- MySQL Documentation: https://dev.mysql.com/doc/
- Aqua Water Management System Internal Documentation

---

14. Appendices

Appendix A: Sample Code Snippets

- Database Connection Code

```
app = Flask(__name__)

app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'Sg155@1505'
app.config['MYSQL_DB'] = 'aqua_water_refiling_management_system'
app.secret_key = 'your_secret_key'

mysql = MySQL(app)
```

- Role Management Code

```
def create_user(username, password, role):
    cursor = mysql.connection.cursor()
    cursor.execute("INSERT INTO users (username, password, role) VALUES (%s, %s, %s)",
                   (username, password, role))  # Password is not hashed anymore
    mysql.connection.commit()
    cursor.close()

def get_user(username):
    cursor = mysql.connection.cursor()
    cursor.execute("SELECT * FROM users WHERE username = %s", (username,))
    user = cursor.fetchone()
    cursor.close()
    return user
```

- Key Queries

```
ALTER TABLE orderdetails
    ADD COLUMN customerID INT NOT NULL,
    ADD COLUMN orderDate DATETIME,
    ADD COLUMN deliveryDate DATETIME,
    ADD COLUMN amount FLOAT,
    ADD COLUMN orderStatus VARCHAR(50);
    ALTER TABLE orderdetails
    ADD foreign key(customerID) references customer(customerID);
```

```
SELECT * FROM aqua_water_refiling_management_system.users;

INSERT INTO `aqua_water_refiling_management_system`.`users` (`user_id`, `username`, `password`, `role`)
VALUES ('2', 'shruti', 'mai_nhi_bataungi', 'admin');
```