MediQuick Medical Delivery Web Application

1. Overview

The MediQuick Medical Delivery Web Application is designed to facilitate the efficient and secure ordering and deliv
It connects users with pharmacies while ensuring stock management and seamless transactions. The system follows a
architecture and is developed using modern web technologies.

This system is divided into four primary modules:

1. User Management
2. Medicine Management
3. Cart & Orders Management
4. Membership Management

2. Assumptions

- The system is deployed on a cloud-based environment with MySQL as the database.
- Role-based access control for Admin, Customers, and Pharmacists.
- ORM tools (Sequelize for Node.js) are used for database interactions.
- A responsive and user-friendly interface is implemented using React.js.

3. **Module-Level Design**

3.1 User Management Module

Purpose: Handles user registration and profile management.

Controller:

- UserController

- registerUser()

- updateUserProfile()

- getUserDetails()

Model:

- User Entity

- userId (PK)

- username

- password (hashed)

- email

- role (ADMIN, CUSTOMER, PHARMACIST)

3.2 Medicine Management Module

Purpose: Manages medicine listings, stock updates, and pricing.

Controller:

- MedicineController

- addMedicine()

- updateMedicine()

- getMedicineById()

- getAllMedicines()

- deleteMedicine()

Model:

- Medicine Entity

- medicineId (PK)

- name

- category

- price

- stockQuantity

- expiryDate

3.3 Cart & Orders Management Module

Purpose: Allows users to add medicines to their cart and place orders.

Controller:

- CartController

- addToCart()

- removeFromCart()

- getCartItems()

- OrderController

- placeOrder()

- getOrderStatus()

- cancelOrder()

Model:

- Cart Entity

- cartId (PK)

- userId (FK)

- medicineId (FK)

- quantity

- Order Entity

- orderId (PK)

- userId (FK)

- totalAmount

- orderDate

- status

3.4 Membership Management Module

Purpose: Handles premium memberships and discount offers.

Controller:

- MembershipController

- subscribeMembership() - getMembershipDetails()

Model:

- Membership Entity

- membershipId (PK)

- userId (FK)

- subscriptionDate

- expiryDate

- status

4. Database Schema

```
CREATE TABLE User (     userId INT
AUTO_INCREMENT PRIMARY KEY,     username
VARCHAR(100),     password VARCHAR(255),     email
VARCHAR(100),     role ENUM('ADMIN', 'CUSTOMER',
'PHARMACIST')
);

CREATE TABLE Medicine (     medicineId INT
AUTO_INCREMENT PRIMARY KEY,     name
VARCHAR(100),     category VARCHAR(50),     price
DECIMAL(10,2),     stockQuantity INT,     expiryDate
DATE
);

CREATE TABLE Order (     orderId INT
AUTO_INCREMENT PRIMARY KEY,
   userId INT,     totalAmount DECIMAL(10,2),     orderDate DATE,     status
ENUM('PENDING', 'CONFIRMED', 'SHIPPED', 'DELIVERED', 'CANCELED'),
FOREIGN KEY (userId) REFERENCES User(userId)
);
```

5. Deployment Details

1. Ensure MySQL is installed and configured.

2. Install necessary dependencies using Node.js (Express.js, Sequelize).

3. Configure database connections in the environment settings.

4. Deploy frontend using React.js and backend using Node.js on cloud platforms.

6. Conclusion

The MediQuick system offers a robust digital healthcare solution with secure transactions, efficient medicine tracking Future updates will enhance service efficiency and AI-driven prescription recommendations.