

DBMS LAB (CSL333) PROJECT

EATWISE RESTAURANT MANAGEMENT SYSTEM

Aravind A Kamath (CEC23CS041)

Arjun Manoj (CEC23CS043)

Anandhakrishnan S (CEC23CS033)

Aravind Sajeev (CEC23CS042)

Computer Science & Engineering (CSE)

Semester 5 (S5)

College of Engineering Cherthala

INDEX

1. Introduction
2. Implementation
3. Code
4. Results
5. Conclusion

INTRODUCTION

- EatWise is a small restaurant management application to demonstrate relational database design, CRUD operations, and use of triggers to maintain derived/calculated columns
- Objective: to simplify the process of restaurant management (order/suborder/review/menu/customer handling) and improve its efficiency
- Tools and technologies used => ReactJS (HTML, CSS, JavaScript, JSX), ExpressJS (NodeJS), MySQL
- Purpose/motivation: to alleviate hassles like mixing up of orders, incorrect cash calculations, outdated menu items and other such issues in restaurants

IMPLEMENTATION

- A simple client-server restaurant management app: ReactJS frontend communicating with an ExpressJS backend using axios/CORS to manage menu, orders, users, reviews with CRUD flows and role-based homepages

MODULES

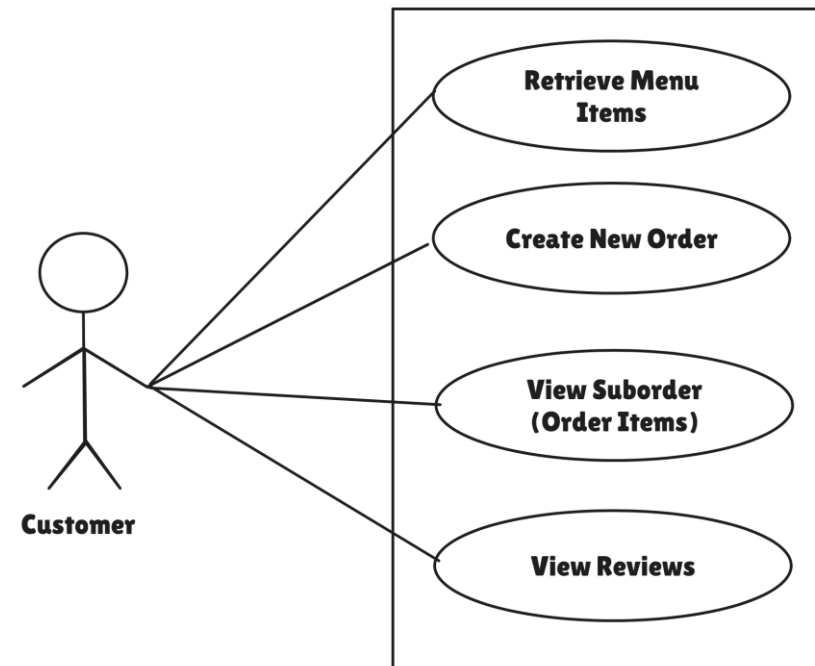
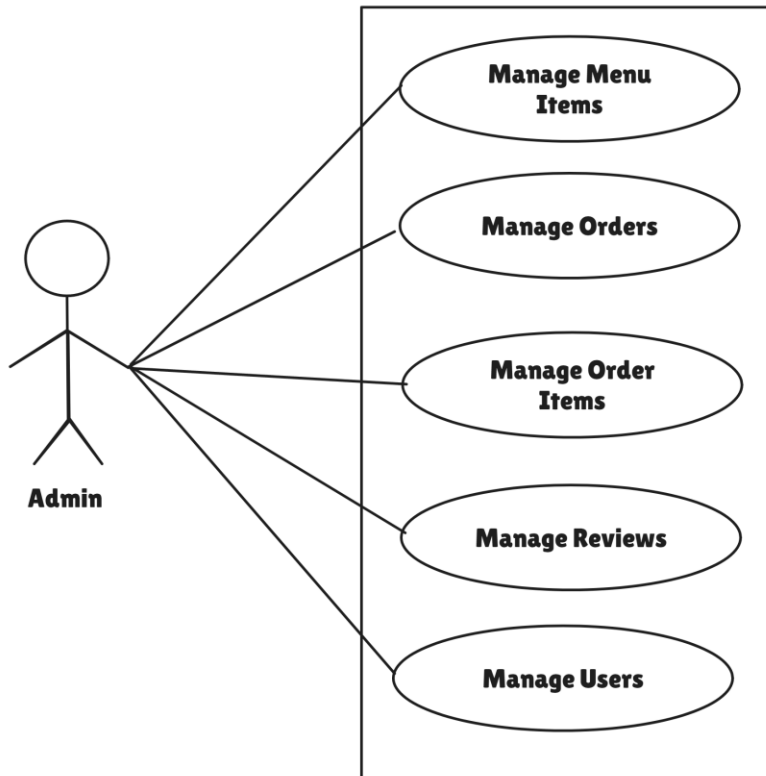
1. Frontend -> App, Homepage, Hero, Login, Review, Credits, AdminLanding, CustomerLanding
2. Backend -> index.js (single backend file)

- User roles – Customer and Admin

- `users` (user_id PK, username, password, email, role)
- `menu_items` (item_id PK, item_name, category, price, item_picture)
- `orders` (order_id PK, user_id FK -> users.user_id, order_date_time, total_amount)
- `order_items` (order_items_id PK, order_id FK -> orders.order_id, item_id FK -> menu_items.item_id, quantity, suborder_price)
- `reviews` (review_id PK, user_id FK -> users.user_id, description, date, rating)

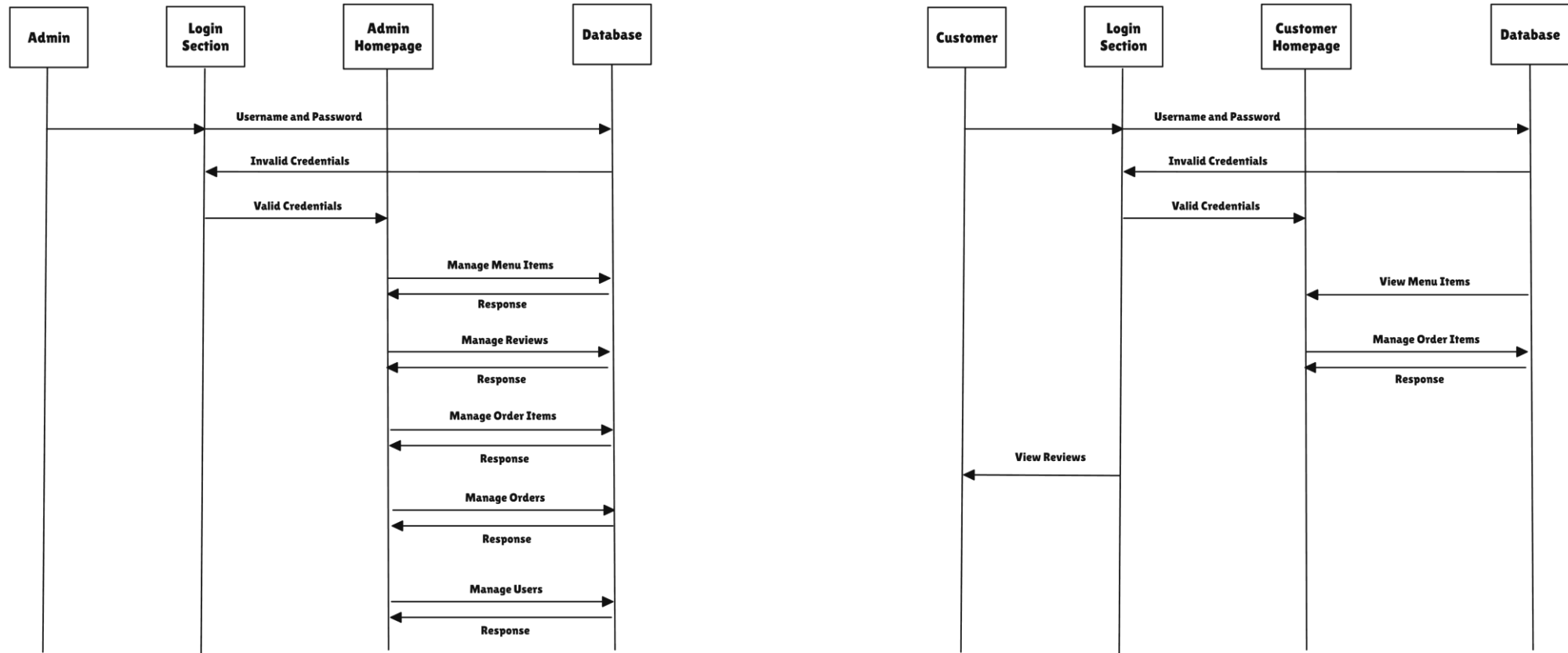
USE CASE DIAGRAM

- Customer is able to view menu items and order
- Admin is able manage all 5 tables with create/retrieve/update/delete (CRUD) operations on the database

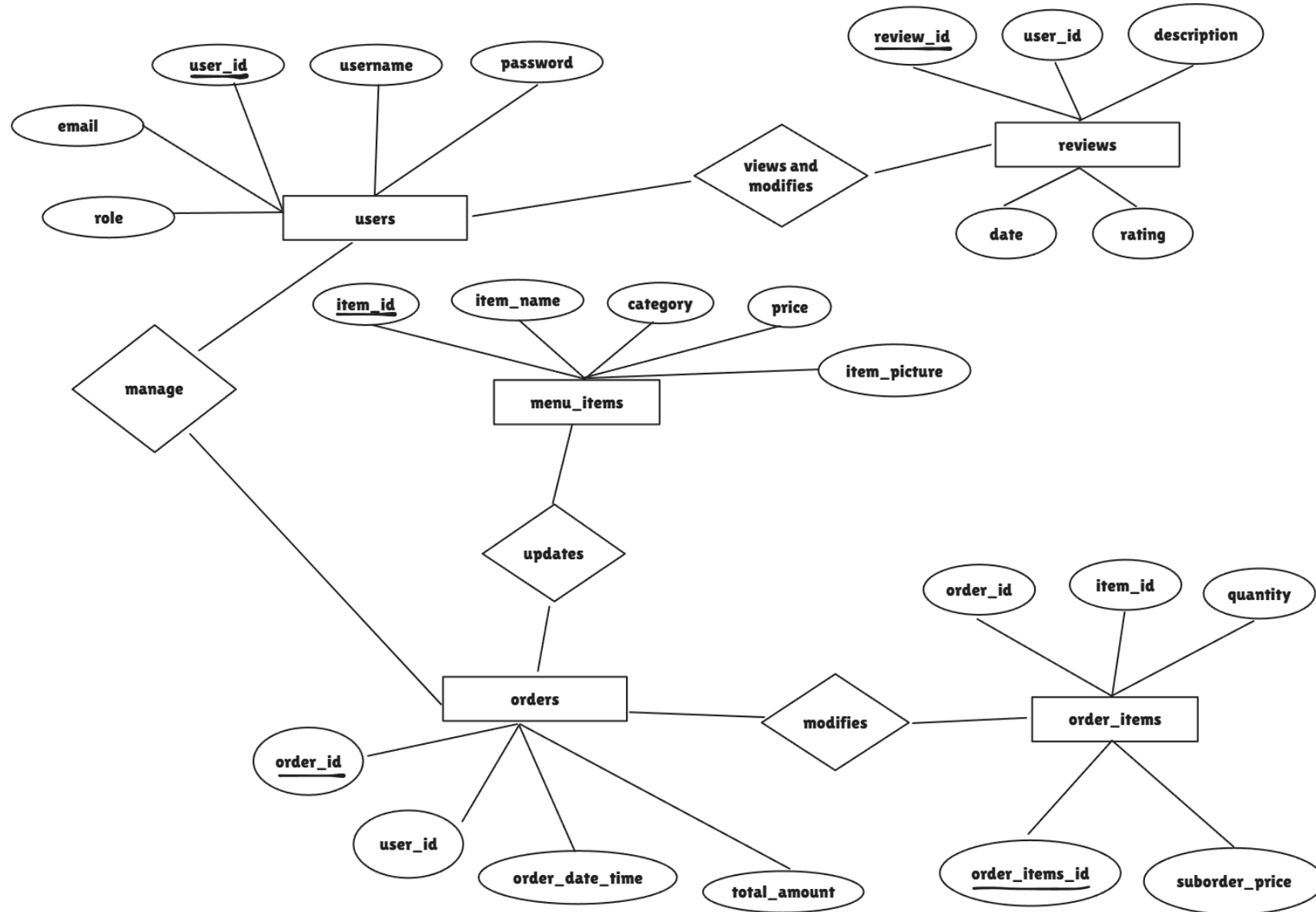


SEQUENCE DIAGRAM

- Customer and admin can view their respective homepages if login is successful
- If not, they will stay in main homepage



ER DIAGRAM



CODE

index.js (backend)




```
1 // Login route
2 app.post('/login', (req, res) => {
3   const { username, password } = req.body;
4   const q = "SELECT * FROM users WHERE username = ? AND password = ?";
5   db.query(q, [username, password], (err, results) => {
6     if (err) return res.status(500).json({ error: 'Database error' });
7     if (results.length === 0) {
8       return res.status(401).json({ error: 'Invalid credentials' });
9     }
10    // Return role and user info
11    const user = results[0];
12    return res.json({ role: user.role, user_id: user.user_id, username: user.username });
13  });
14 });
```


CODE

index.js (backend)

Creation of table menu_items



```
1 CREATE TABLE `menu_items` (  
2   `item_id` int NOT NULL,  
3   `item_name` varchar(45) DEFAULT NULL,  
4   `category` varchar(45) DEFAULT NULL,  
5   `price` int DEFAULT NULL,  
6   `item_picture` varchar(100) DEFAULT NULL,  
7   PRIMARY KEY (`item_id`),  
8   UNIQUE KEY `item_id_UNIQUE` (`item_id`)  
9 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

CODE

index.js (backend) Updation of menu_items

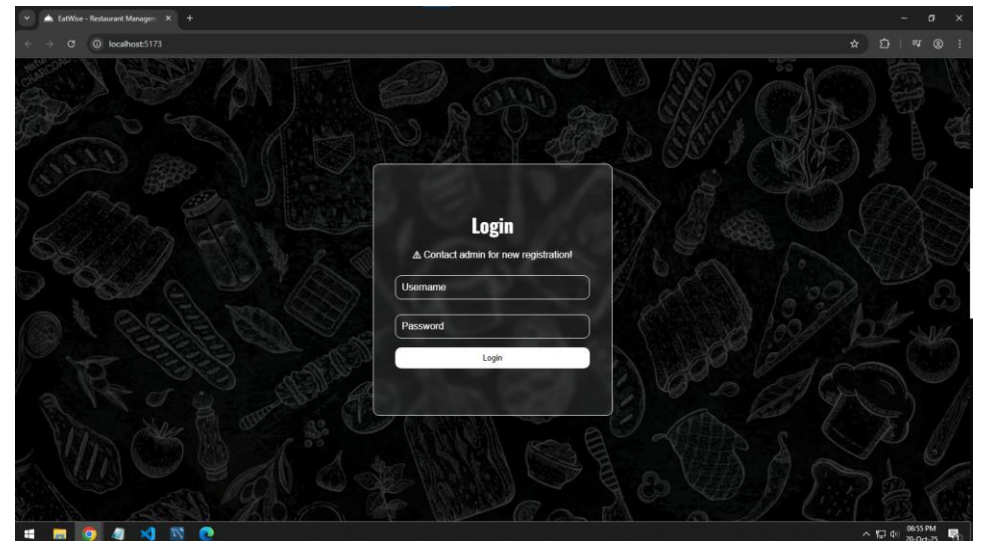
```
1 // Update menu item by id
2 app.put('/menu_items/:id', (req, res) => {
3     const menuItemId = req.params.id;
4     const q = "UPDATE menu_items SET `item_name` = ?, `category` = ?, `price` = ?, `item_picture` = ? WHERE item_id = ?";
5     const values = [
6         req.body.item_name,
7         req.body.category,
8         req.body.price,
9         req.body.item_picture
10    ];
11    db.query(q, [...values, menuItemId], (err, data) => {
12        if (err) return res.json(err);
13        return res.json("Menu item updated successfully!");
14    });
15 })
```

RESULTS

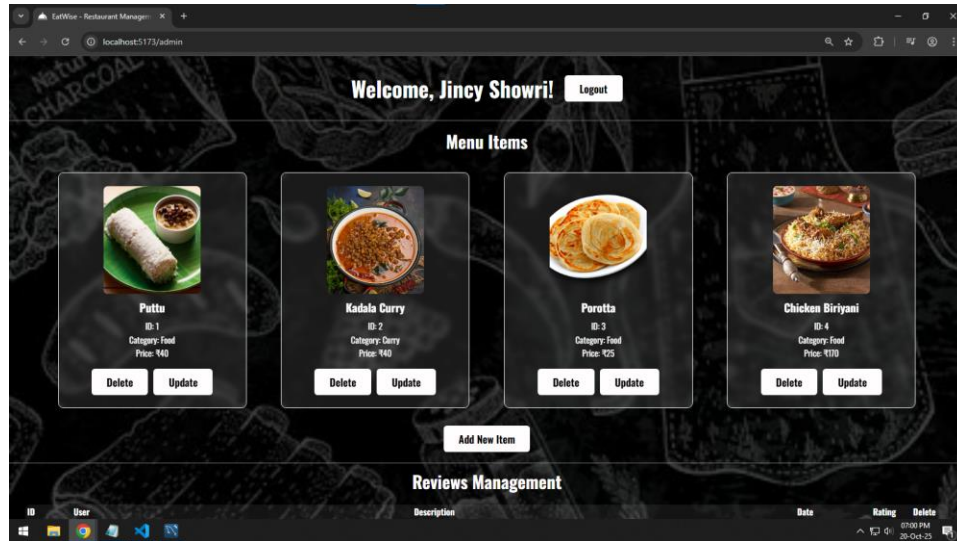


- Homepage – simple layout with CSS flexbox and relative/absolute positioning

- Login section with glassmorphism UI design with simple inputs and button

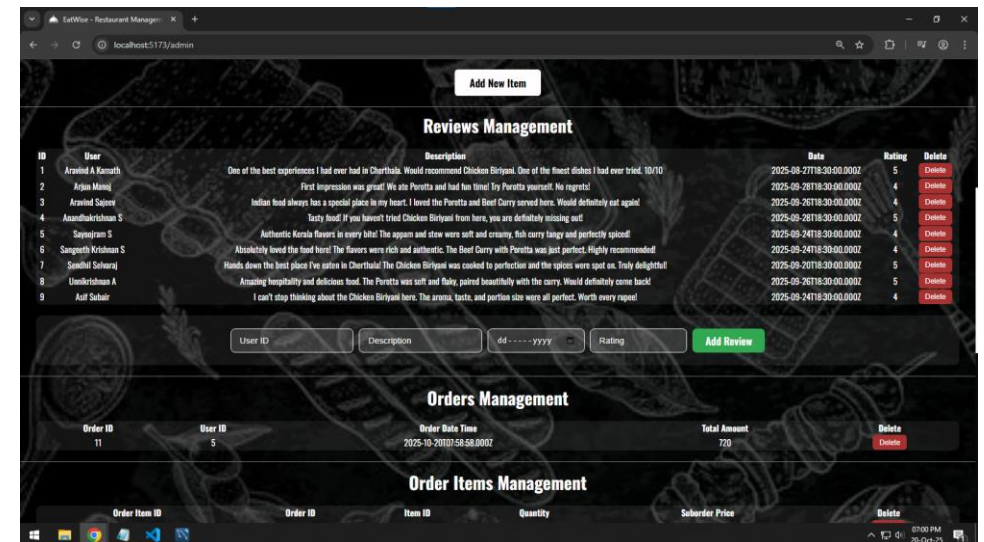


ADMIN HOMEPAGE

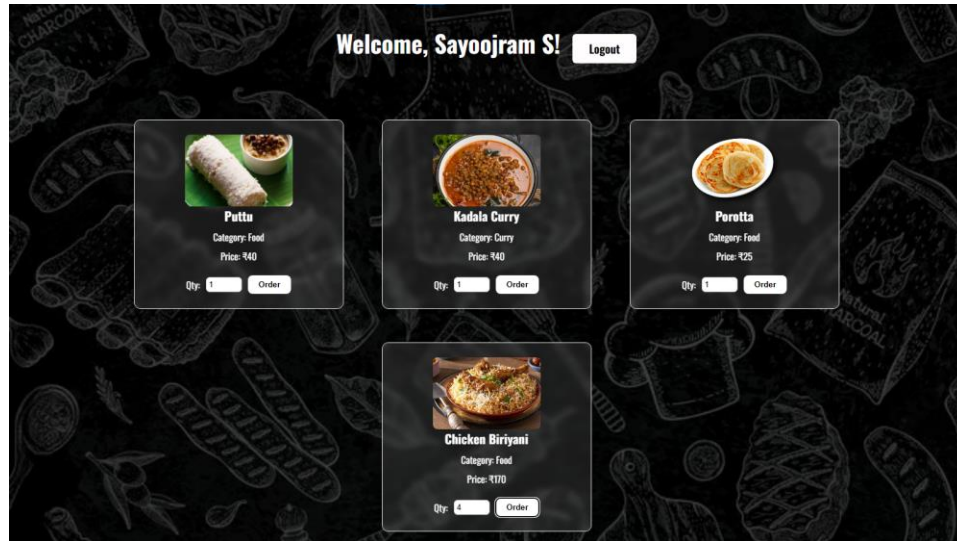


- Ability to modify all 5 tables' data (menu_items section shown in screenshot)

- Ability to manage different tables including reviews, orders etc...

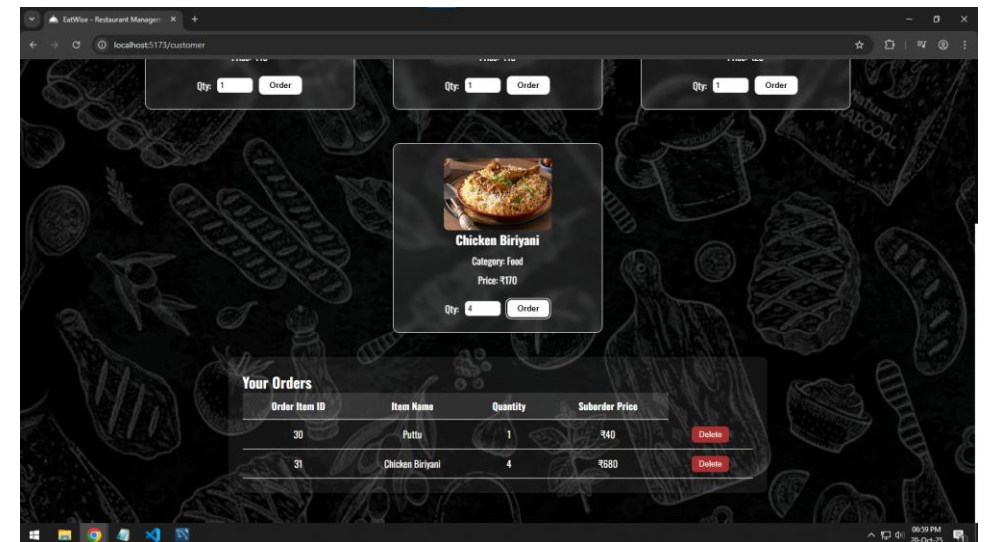


CUSTOMER HOMEPAGE



- Ability to view menu_items, select quantity and order

- Ability to view and delete suborders



CONCLUSION

- A basic but diversely feature-rich web app with vast technologies used for management of 5 tables with data consistency and objective to improve customer/admin experience in restaurant management system
- Technologies learned – ReactJS, React Router, React DOM, NodeJS, ExpressJS, MySQL, CORS, Nodemon, SQL Triggers
- Possible improvements – adding proper authentication with security, adding more options for table modifications