**1.INTRODUCTION**

MealStack is a comprehensive, web-based meal and food service management system designed to automate and streamline daily operations in institutional canteens, corporate cafeterias, and similar food service environments. The system integrates meal inventory management, daily menu handling, order processing, and user management into a unified platform, enabling efficient coordination between administrators and end users. By digitizing traditionally manual processes, MealStack reduces operational overhead, minimizes errors, and improves overall service efficiency.

The application is developed using React for the frontend, providing a responsive, component-based user interface that enhances usability and interaction. The backend is implemented using Spring Boot, which offers a robust and scalable RESTful API architecture to handle business logic, data processing, and system workflows. Secure authentication and authorization are enforced using JWT (JSON Web Token) integrated with Spring Security, ensuring stateless and role-based access control across the system.

MealStack is designed to support real-time meal availability tracking, daily menu resets, automated order handling, and secure user interactions. The system caters to the operational requirements of food service administrators while delivering a seamless ordering experience to users. By combining modern full-stack technologies with a layered architectural approach, MealStack provides a scalable and maintainable solution for efficient food service management.

MealStack is specifically tailored to meet the unique needs of educational canteen environments, ensuring real-time menu updates, automated stock management, wallet-based transactions, and a user-friendly interface for both students and administrators. This integration of inventory management with a digital ordering system helps canteens reduce manual errors, eliminate cash handling, improve operational efficiency, and provide superior customer service to students.

In the MealStack system, both Student and Admin roles play crucial roles in the management and operation of the platform. Each role comes with specific responsibilities, permissions, and access levels to ensure that the system operates smoothly and securely.

**System Roles:**

**Student Role**: Students can register, login, browse daily menus, place orders using wallet balance, track order status, view order history, recharge wallet, and manage profile.

**Admin Role**: Administrators manage student accounts, maintain item catalog, set daily menus with quantities, process orders, mark orders as served, and view reports and analytics.

**Key Features:**

• Real-time menu browsing with availability

• Wallet-based cashless payments

• Automated inventory tracking

• Order processing workflow

• Comprehensive reporting

• Mobile-responsive design

**Security**: JWT-based authentication, BCrypt password hashing, role-based access control, SQL injection prevention, XSS protection, and CORS configuration.

* 1. **PURPOSE**

MealStack digitizes canteen operations to:

**For Students:**

• Enable cashless, convenient ordering

• Reduce waiting times

• Provide transparent pricing and availability

• Maintain digital transaction records

**For Administrators:**

• Streamline order processing

• Eliminate cash handling

• Provide real-time inventory visibility

• Enable data-driven decisions

• Reduce food wastage

**Overall Goals:**

• Complete digitization of canteen operations

• Paperless, eco-friendly system

• Improved efficiency and customer satisfaction

• Scalable for growing populations

**1.2 Scope**

**Functional Modules:**

1. User Management: Registration, authentication, profile management

2. Menu Management: Item catalog, categorization, pricing

3. Inventory Management: Daily stock, real-time tracking, alerts

4. Order Management: Cart, placement, tracking, history

5. Wallet & Payment: Balance management, recharge, transactions

6. Reporting: Sales, inventory, and analytics

**Technical Stack:**

• Frontend: React.js, Redux, Material-UI, Axios

• Backend: Spring Boot, Spring Security, JPA/Hibernate, MySQL

• Security: JWT, BCrypt, CORS

• Integration: Payment gateway, Email service

**Users:** Students and Canteen Administrators

Deployment: Single institution, cloud or on-premise

Out of Scope: Physical equipment, payroll, supplier management

* 1. **Objectives:**

**1.** **Efficient Inventory Management**: Real-time tracking, automated stock updates, low-stock alerts, wastage minimization

**2. Streamlined Order Processing:** Instant processing, elimination of manual errors, automated calculations

**3. Cashless Transactions:** Digital wallet system, payment gateway integration, complete transaction history

**4. Enhanced User Experience**: Intuitive interfaces, quick browsing, mobile-responsive design

**5. Robust Security:** JWT authentication, role-based access, data encryption, vulnerability prevention

**6. Comprehensive Analytics:** Sales reports, inventory analysis, student spending patterns, business insights

**7. Operational Efficiency:** Reduced manual effort, automated tasks, optimized workflows

**1.4 KEY FUNCTIONALITIES**

**User Management:**

• Student registration with email validation

• JWT-based login and authentication

• Profile management and password change

• Role-based access control (Student/Admin)

**Menu & Inventory (Admin):**

• Item master catalog with categories (Breakfast, Lunch, Snacks, Dinner, Beverages)

• Cuisine types (South Indian, North Indian, Oriental, Maharashtrian)

• Daily menu setup with quantities

• Real-time stock tracking and updates

• Low-stock alerts and manual adjustments

**Order Management:**

• Browse menu with real-time availability

• Shopping cart with quantity management

• Order placement with balance validation

• Unique transaction ID generation

• Order status tracking (Pending/Served)

• Complete order history

**Wallet & Payment:**

• View wallet balance

• Recharge through payment gateway

• Automatic payment deduction

• Transaction history with receipts

**Admin Functions:**

• Student account management

• Pending order processing

• Mark items/orders as served

• Sales and inventory reports

• Dashboard with key metrics

**Additional Features:**

• Responsive design for all devices

• Email notifications

• Data export (Excel/PDF)

• Audit logging

• Performance optimization

**2. SOFTWARE REQUIREMENT SPECIFICATION**

**2.1 FUNCTIONAL REQUIREMENTS**

**1: User Management**

• Student registration with name, email, password, mobile, DOB, course

• Email uniqueness validation and BCrypt password hashing

• JWT token generation on login (24-hour expiry)

• Profile view/update and password change

• Role-based access control (STUDENT/ADMIN)

**2: Menu Item Management (Admin)**

• Add items with name, price, category, genre, image (unique name constraint)

• Update item details and pricing

• Delete items (with validation for active references)

• Search, filter, and sort items

**3: Daily Inventory (Admin**)

• Set daily menu from master catalog with quantities

• Unique constraint on (date, item\_id)

• Real-time stock tracking (Available = Initial - Sold)

• Automatic quantity reduction on orders

• Manual inventory adjustments with logging

• Low-stock alerts

**4: Order Management**

• Shopping cart with add/modify/remove items

• Order placement with balance and stock validation

• Atomic transaction: deduct balance, create order, update inventory

• Generate unique transaction ID

• View pending/completed orders

• Mark orders as served

• Order history with filtering

**5: Wallet & Payment**

• View wallet balance

• Recharge with min/max limits through payment gateway

• Instant balance update on payment

• Transaction history with receipts

• Automatic payment deduction for orders

**6**: **Student Management (Admin)**

• View all students with details

• Add/edit/delete student accounts

• Search and filter by course/balance

• View student statistics

**7: Reporting (Admin)**

• Daily sales reports (item-wise, category-wise)

• Inventory status and turnover reports

• Student spending analytics

• Dashboard with key metrics

**8: Security**

• JWT authentication with role validation

• BCrypt password hashing

• Input validation and sanitization

• Audit logging

**2.2 Non-Functional Requirements**

**Performance:**

• Response time < 2 seconds

• Support 100 concurrent users

• Process 50 orders/minute during peak

• Database queries < 100ms

**Reliability:**

• 99.5% uptime

• Daily automated backups

• Transaction rollback on failures

• Graceful error handling

**Usability:**

• Intuitive navigation (max 3 clicks)

• Mobile responsive design

• No training required for students

• Clear error messages

**Security:**

• Strong password policy (min 8 chars)

• HTTPS for all communications

• SQL injection and XSS prevention

• CSRF protection

• Audit trails

**Maintainability:**

• Modular MVC architecture

• 70%+ code coverage

• Comprehensive documentation

• Version control with Git

**Scalability:**

• Horizontal scaling support

• Database sharding capability

• Caching for frequent queries

• Load balancing ready

**2.3 Hardware and Software Requirements**

**HARDWARE**

**Server (Minimum):**

• Processor: Intel i5/AMD Ryzen 5 (Quad-core, 2.5 GHz)

• RAM: 8 GB

• Storage: 50 GB SSD

• Network: 100 Mbps

**Client (Minimum):**

• Processor: Dual-core 1.5 GHz

• RAM: 4 GB

• Display: 1024x768

• Network: 4G/10 Mbps

**SOFTWARE**

**Backend:**

• OS: Ubuntu 22.04/Windows Server 2019

• JDK: 17+

• Database: MySQL 8.0

• Server: Embedded Tomcat (Spring Boot)

• Build: Maven 3.8+

**Backend Frameworks:**

• Spring Boot 3.9.18

• Spring Security 6.x

• Spring Data JPA 3.x

• Hibernate 6.x

• JWT, Lombok, Swagger

**Frontend:**

• Browsers: Chrome 100+, Firefox 95+, Edge 100+, Safari 15+

• Node.js 18.x, npm 8.x

**Frontend Frameworks:**

• React 18.x

• Redux 4.x

• Material-UI 5.x

• React Router 6.x

• Axios 1.x

**Development Tools:**

• IDE: VS Code/IntelliJ/Eclipse/Spring Suite

• Version Control: Git

• API Testing: Postman/Swagger ui

• Database: MySQL Workbench

**Additional:**

• Payment Gateway: Razorpay/Stripe

• Email: SMTP (Gmail/SendGrid)

• SSL: Let's Encrypt

**3.DIAGRAMS**

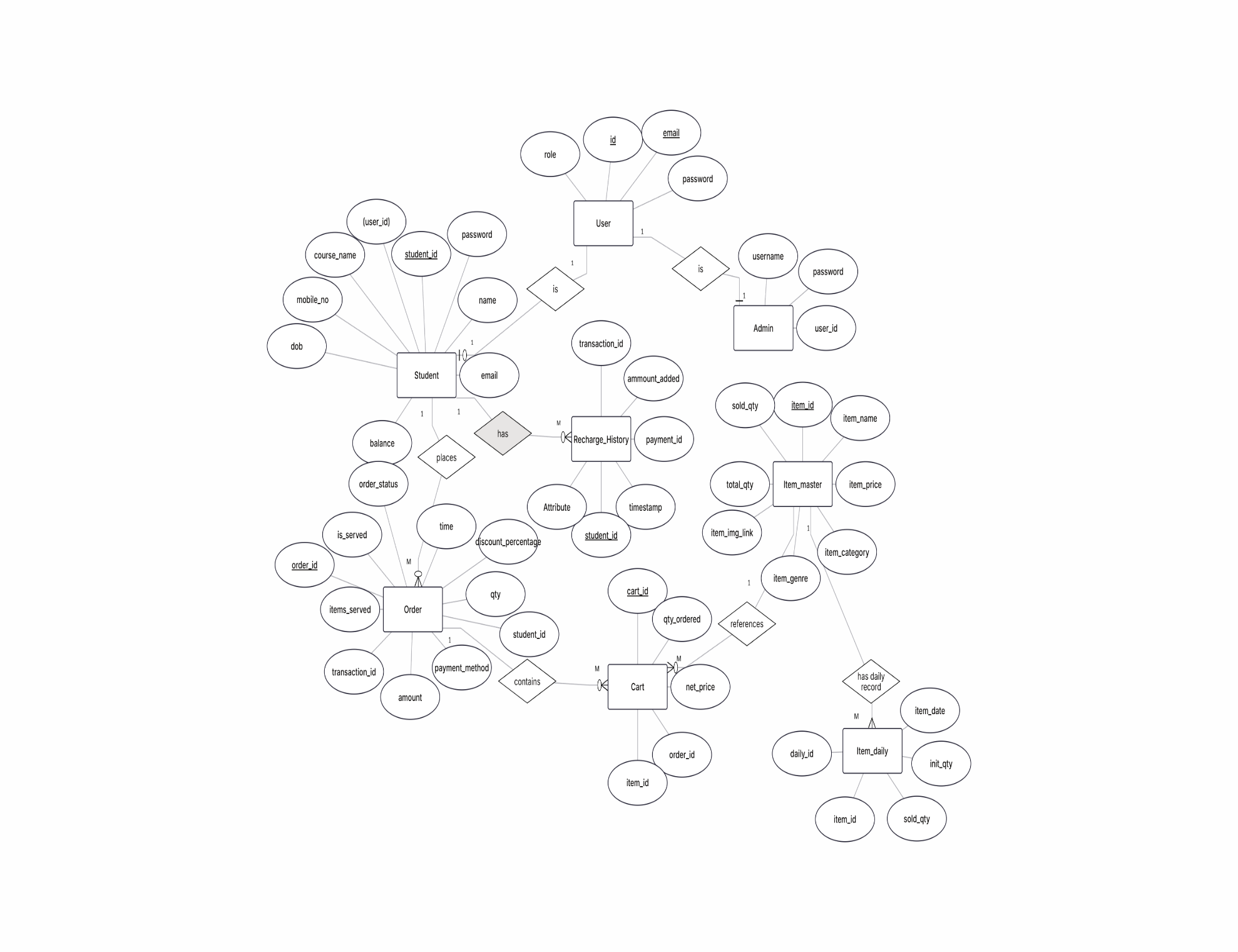
**3.1 Entity Relationship Diagram:**

Figure 1 ER Diagram for MealStack-Canteen management System

**3.2 Use Case Diagram:**

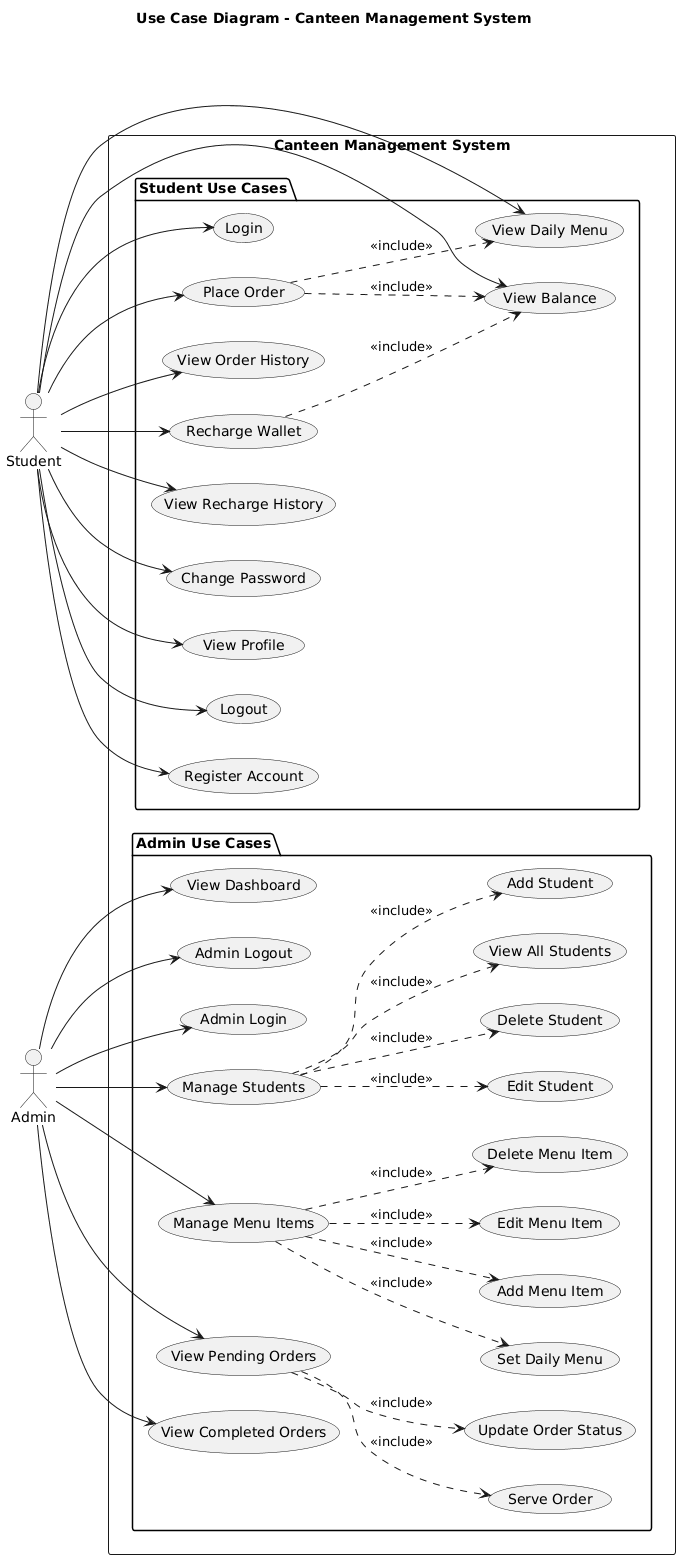


Figure 2 Use Case Diagram for MealStack-Canteen Management System

**3.3 Data Flow Diagram:**

**DFD Level 0:**

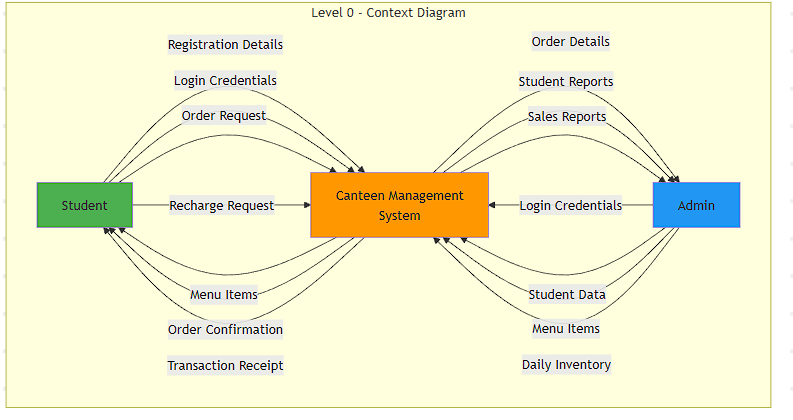


Figure 3 Data Flow 0

**DFD Level 1:**

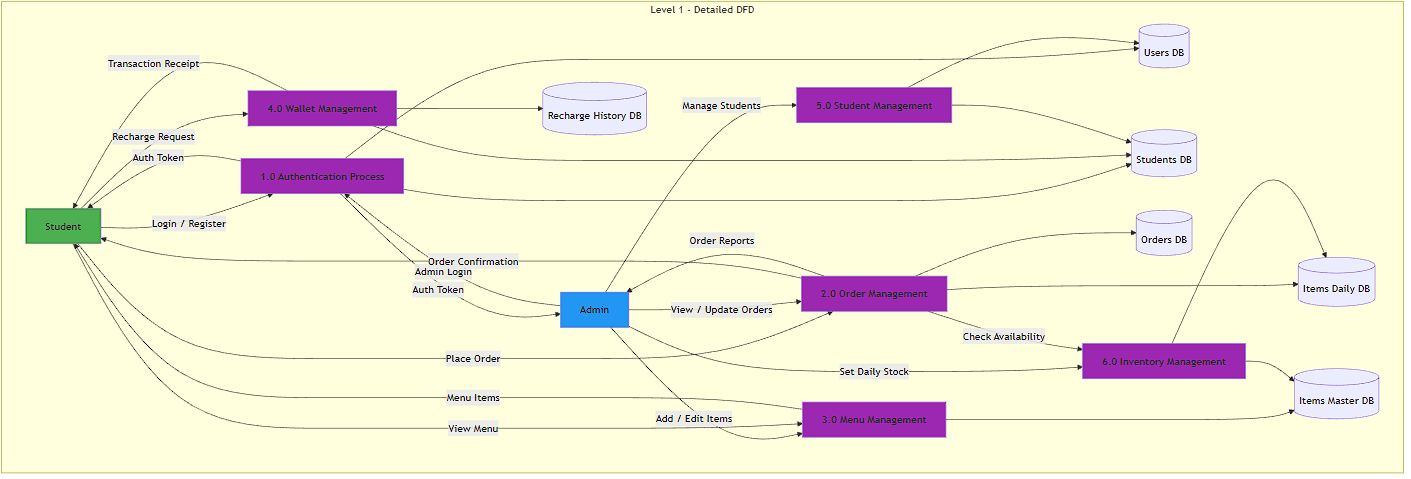


Figure 4 Data flow 2

**3.4 Activity Diagram:**

1.Login Activity Diagram

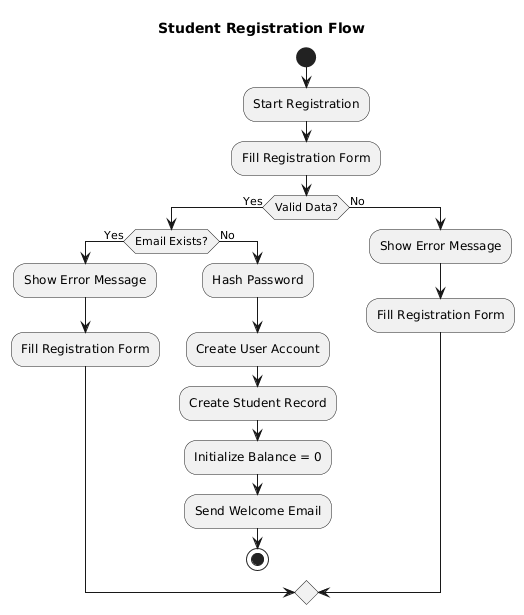


Figure 5 Login Activity

2.Admin Activity Diagram:

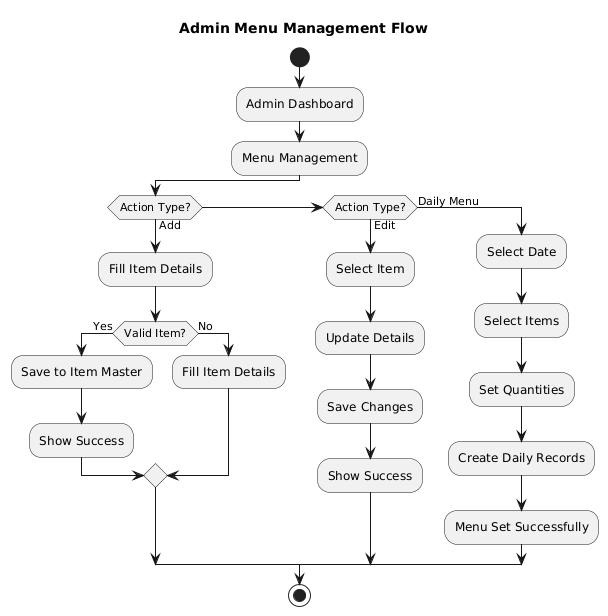


Figure 6 Admin Activity

3.Admin Order Processing Activity Diagram



Figure 7 Admin Order

4.Student Order Activity Diagram

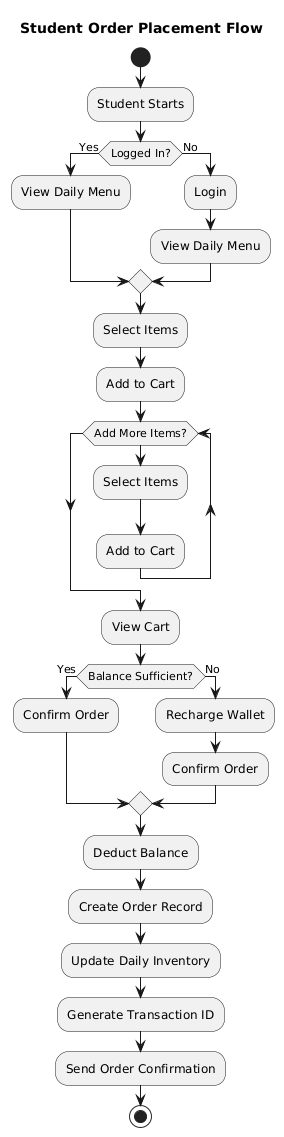
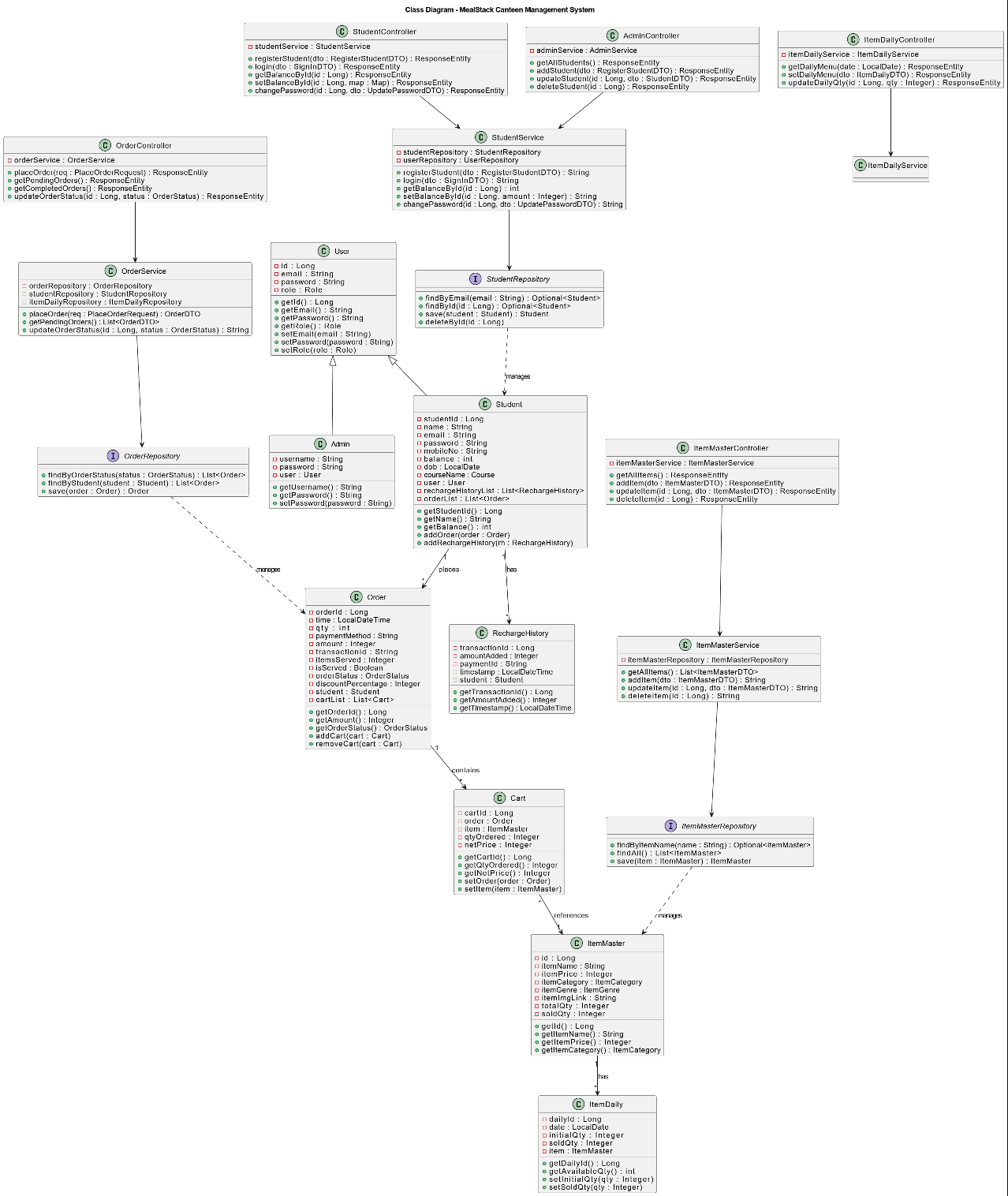


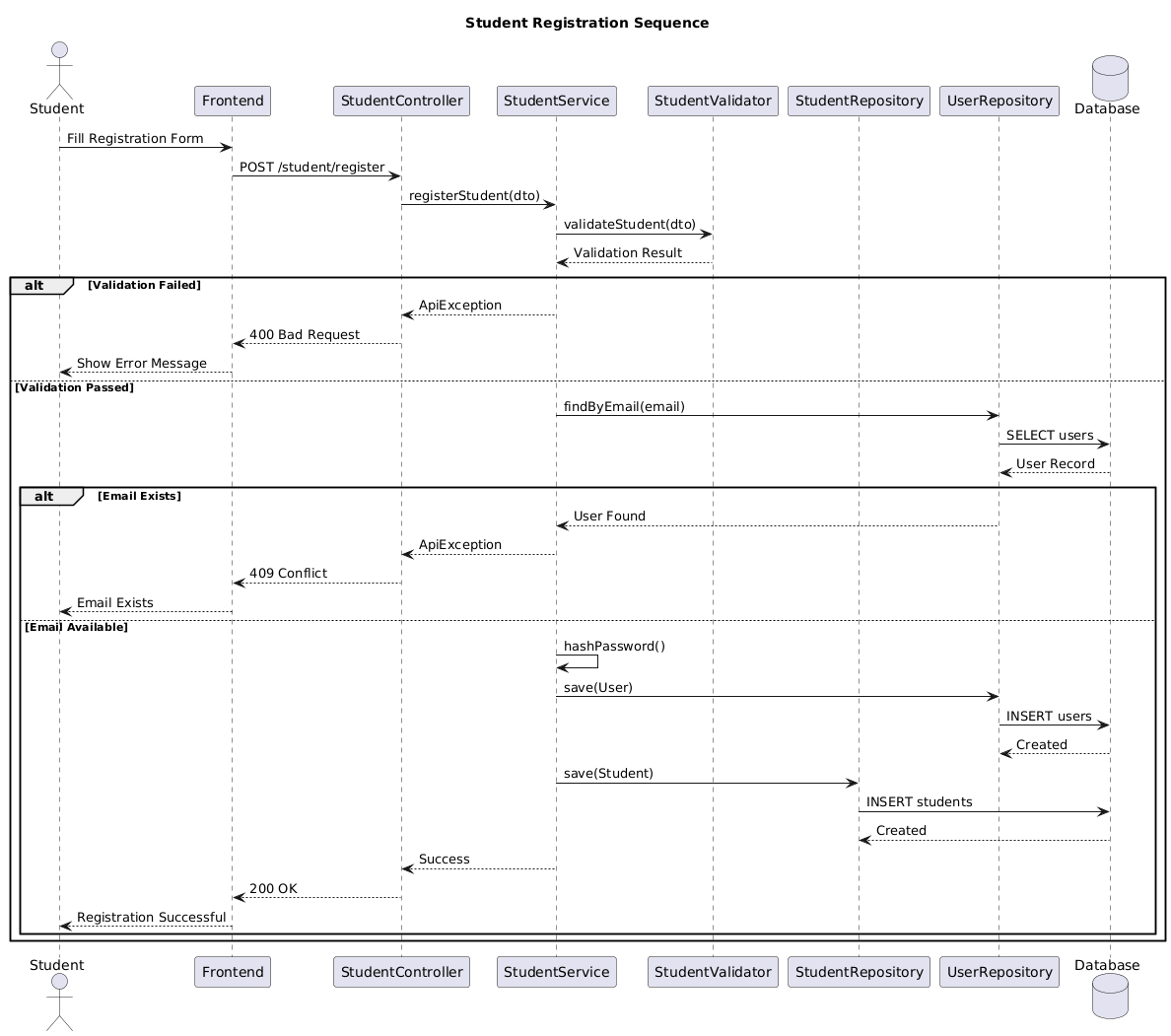
Figure 8 Student Order

**3.5 Class Diagram:**

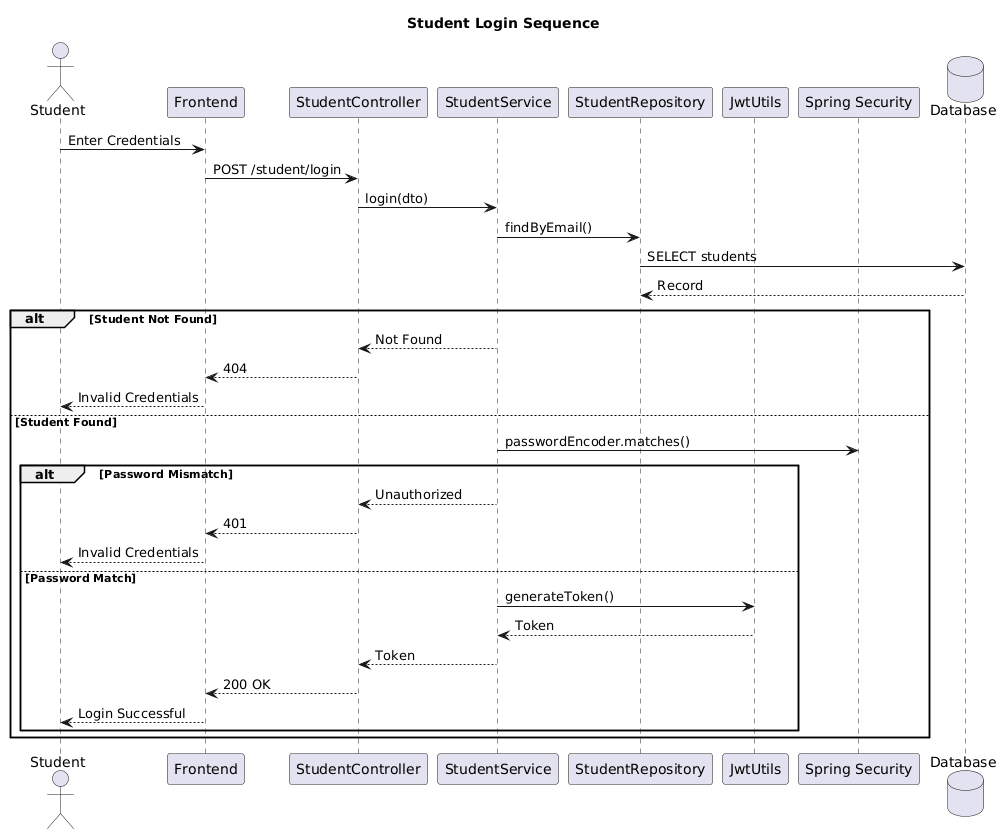


**3.6 Sequence Diagram**

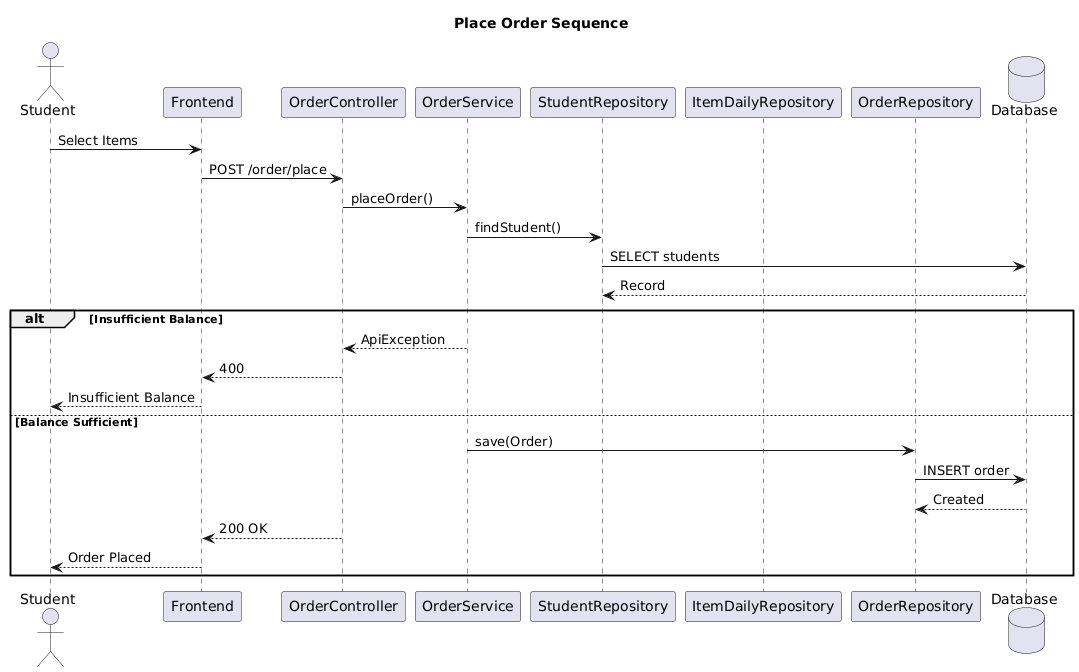
1.Student registration Sequence



2.Student Login Sequence



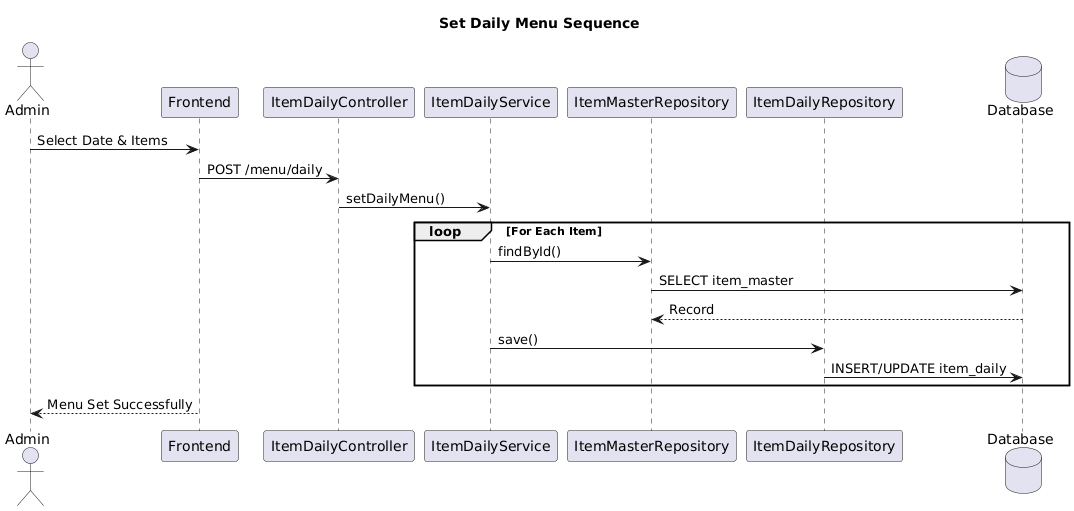
3.Place order sequence



4.Wallet sequence

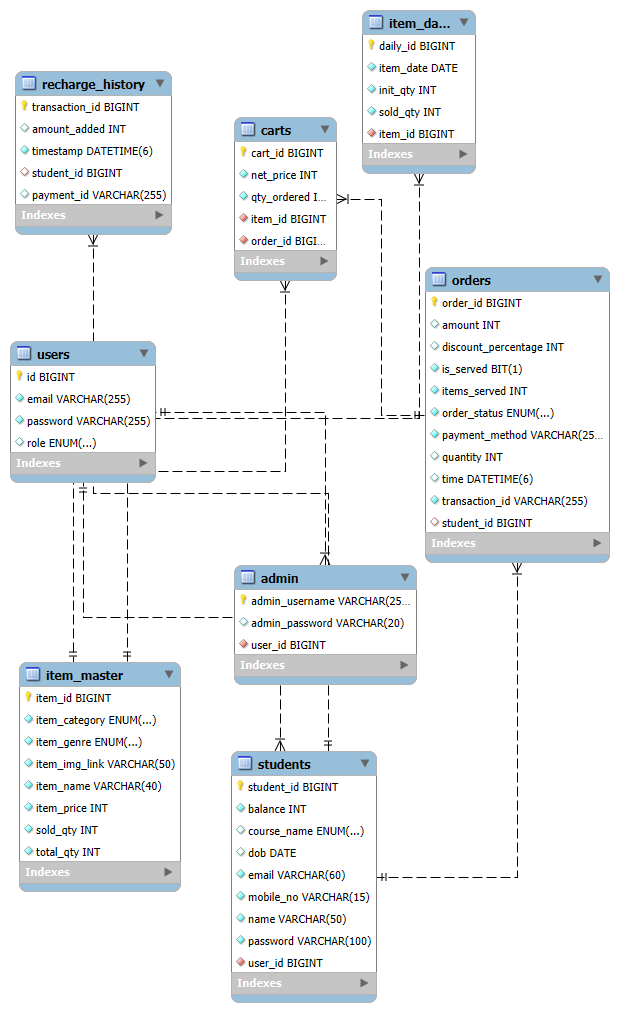


5.Set daily sequence



**4. DATABASE DESIGN**

**4.1 Design:**



**4.2 Tables:**

The following table structures depict the database design

Table 1 Tables

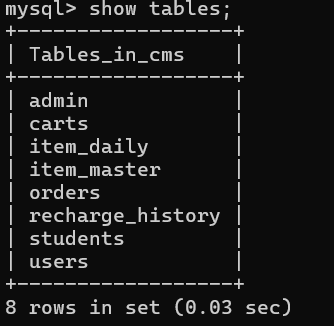


Table 2 Users

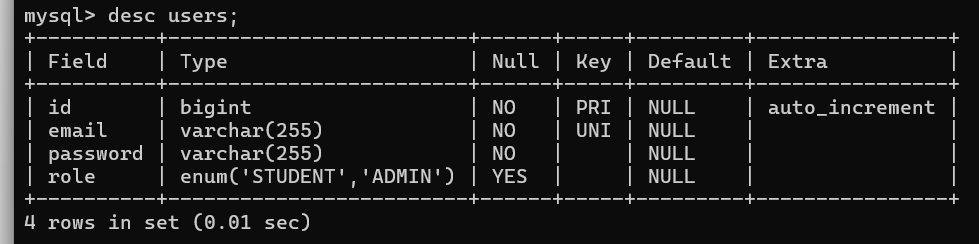


Table 3 Students

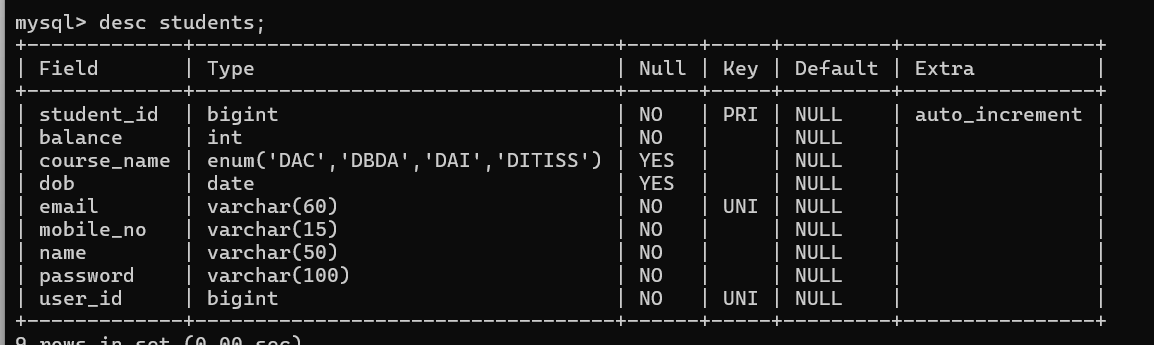


Table 4 admin

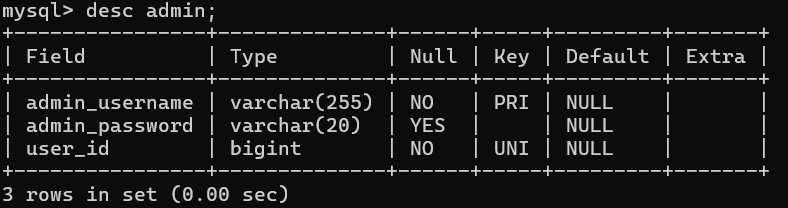


Table 5 orders



Table 6 recharge

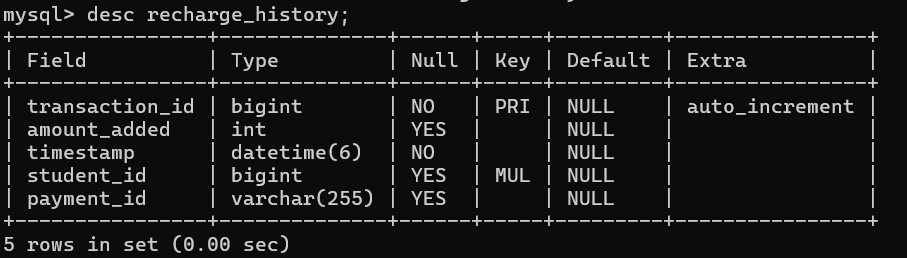


Table 7 item\_daily

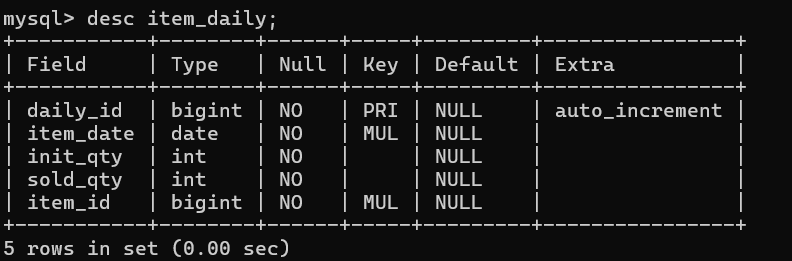


Table 8 item\_master

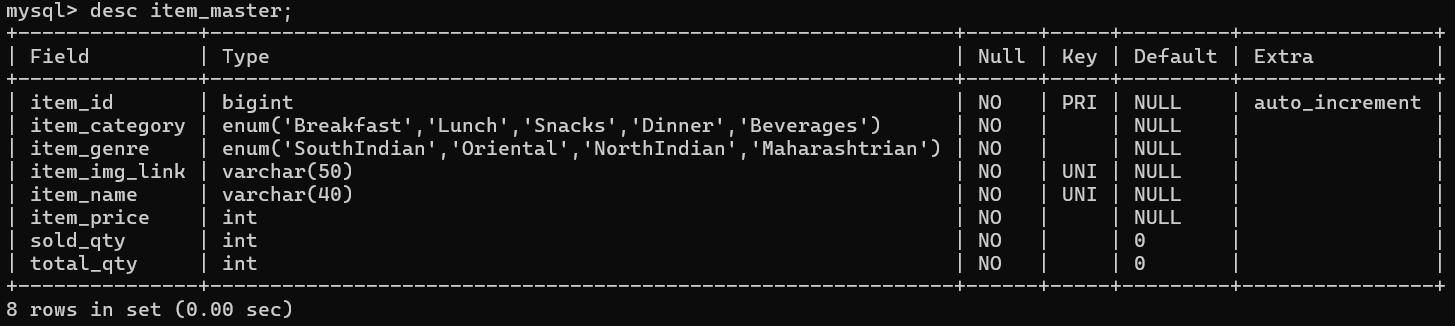
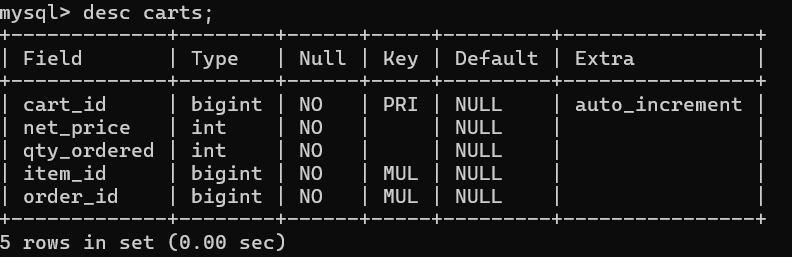
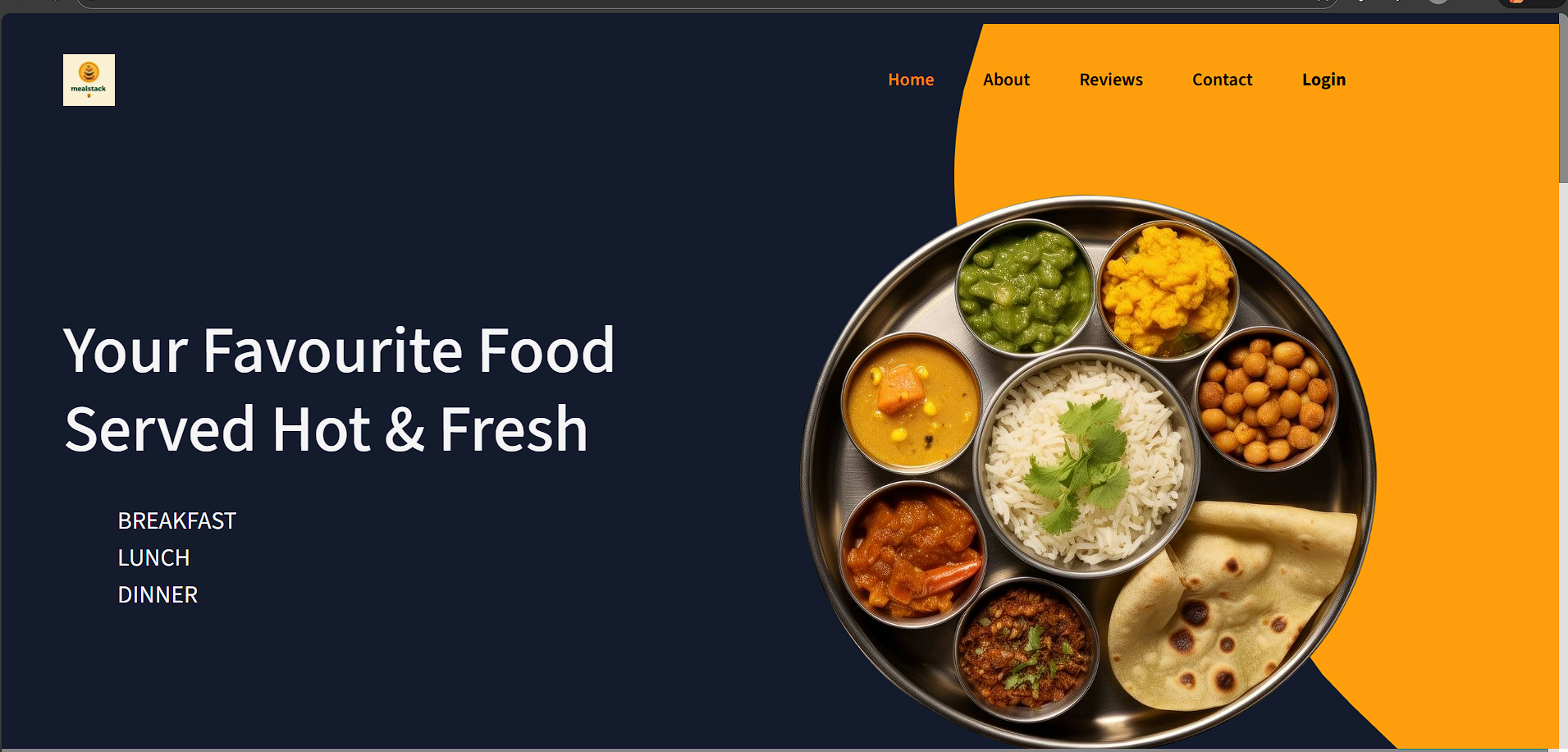


Table 9 carts

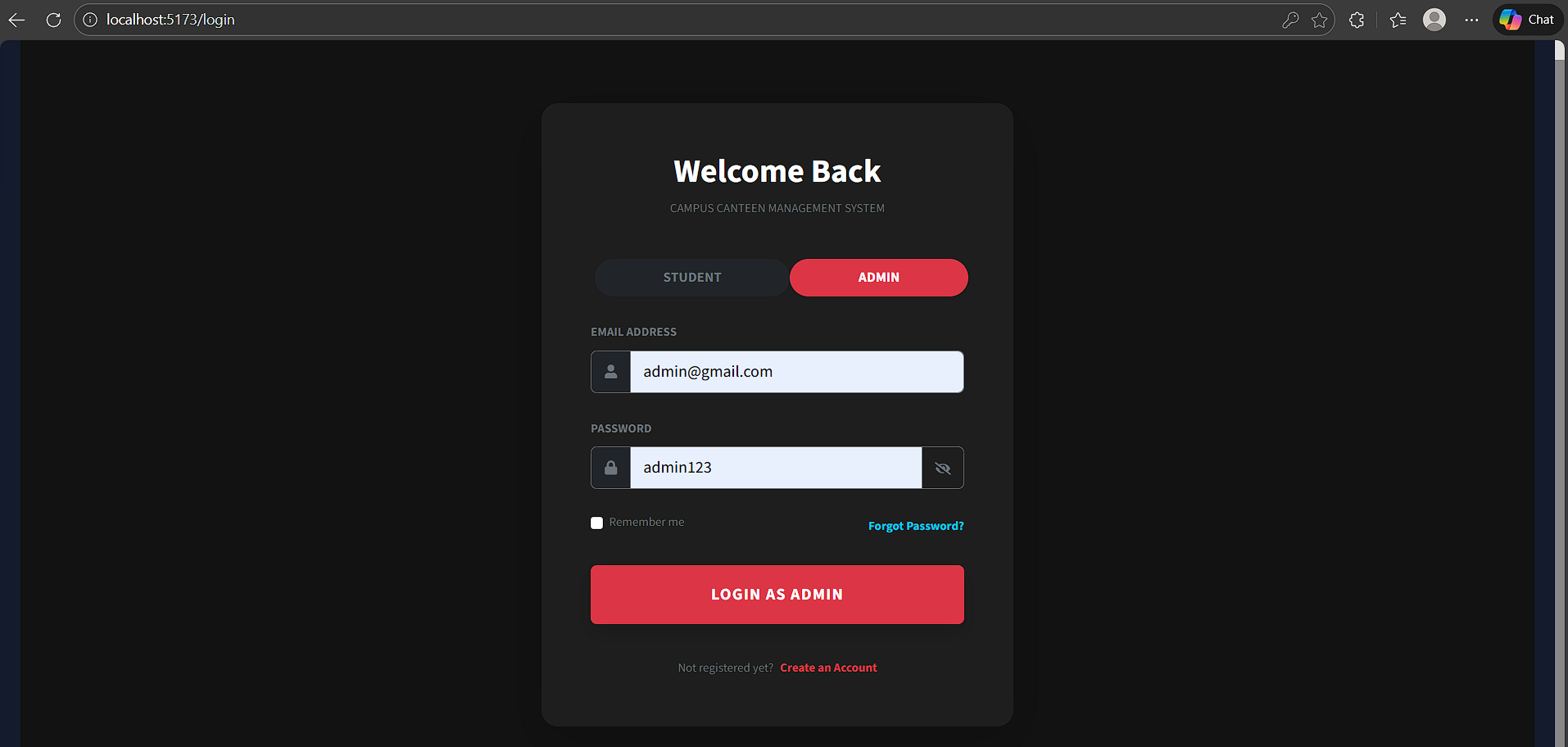


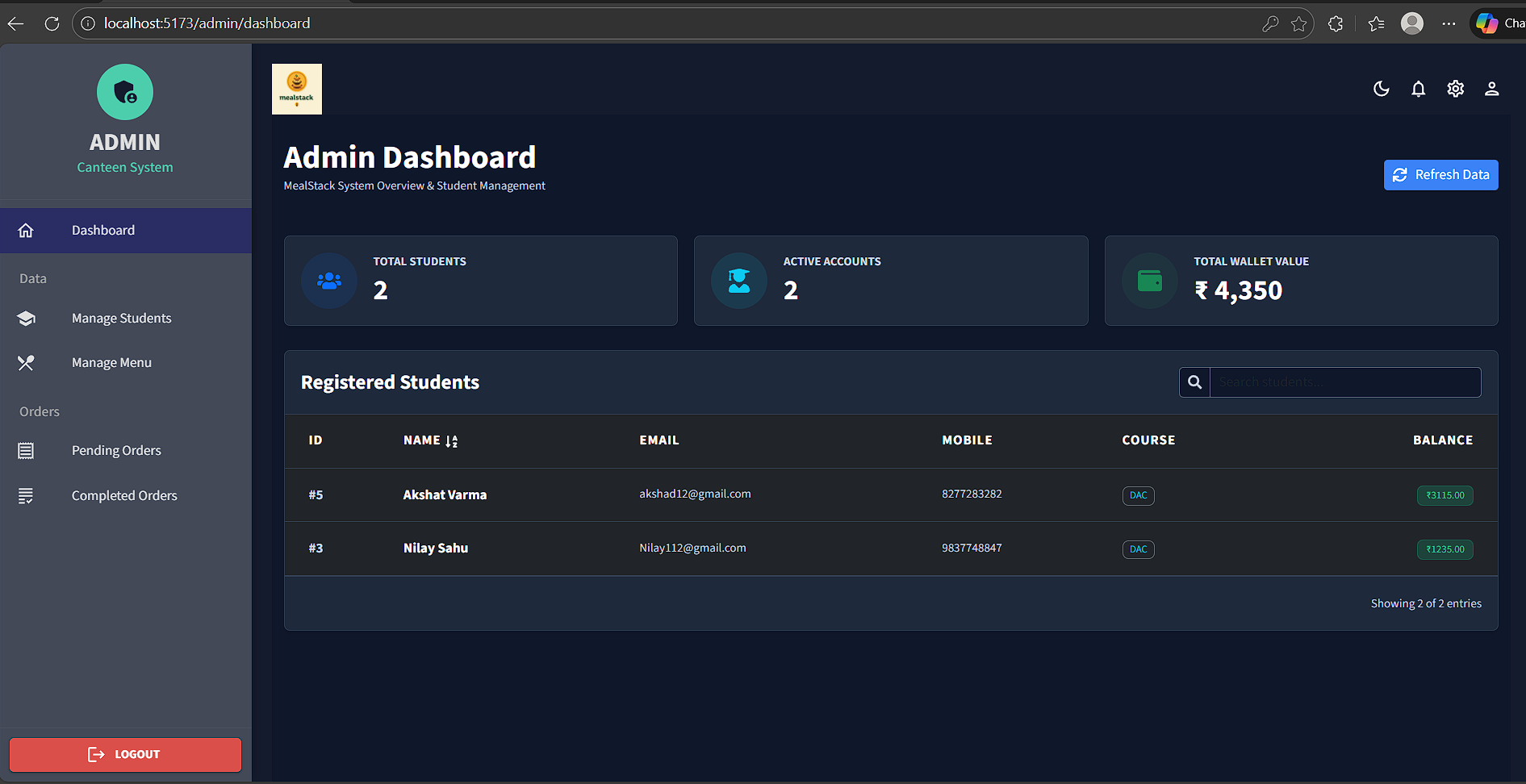
**5.SNAPSHOTS**

**Home page:**

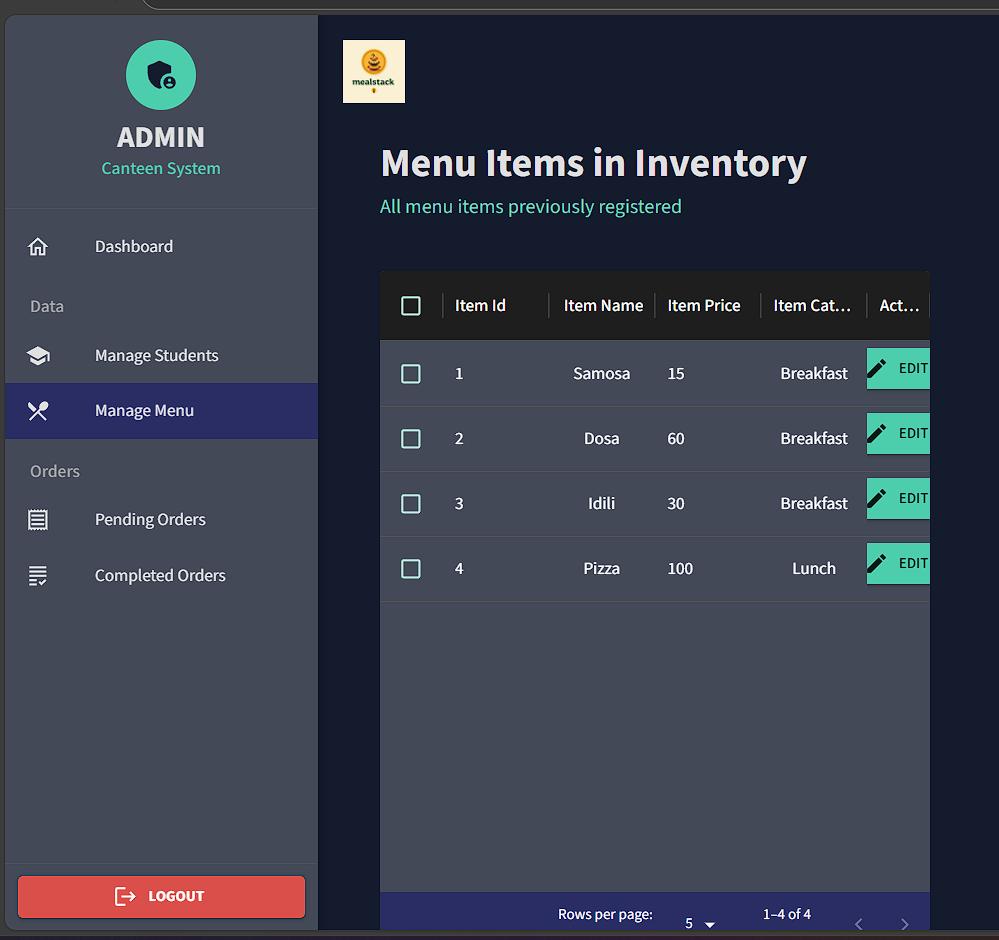


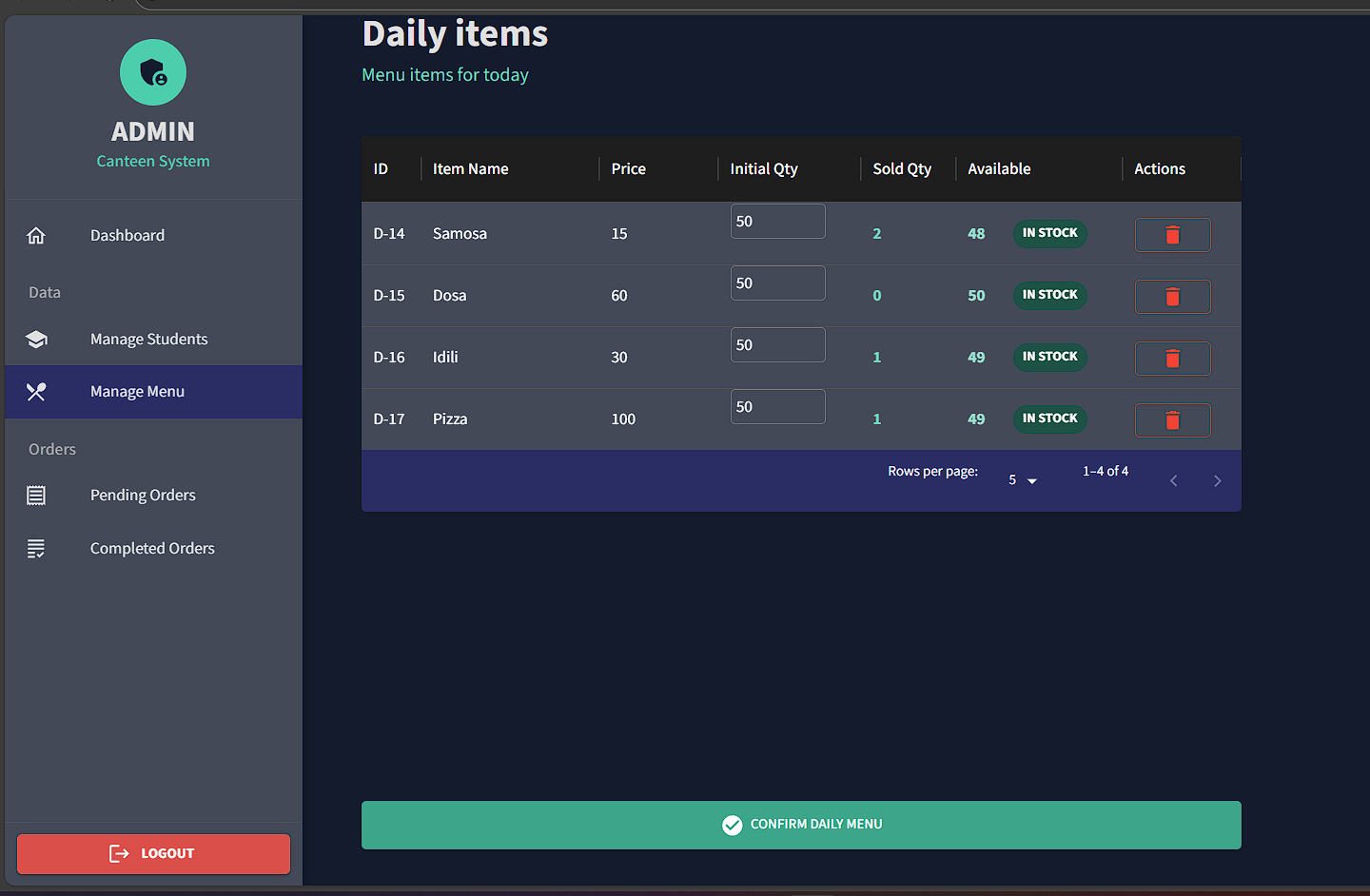
**Admin DashBoard:**

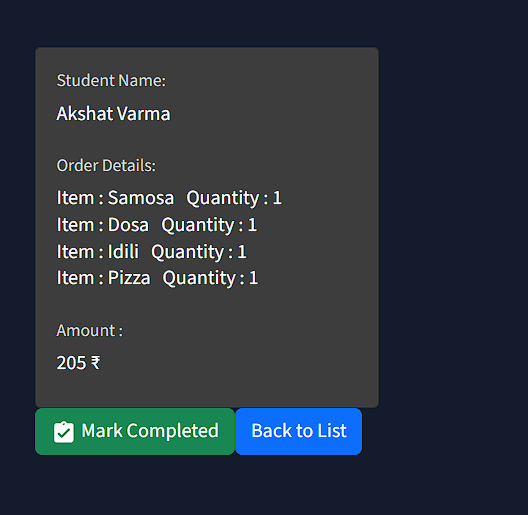


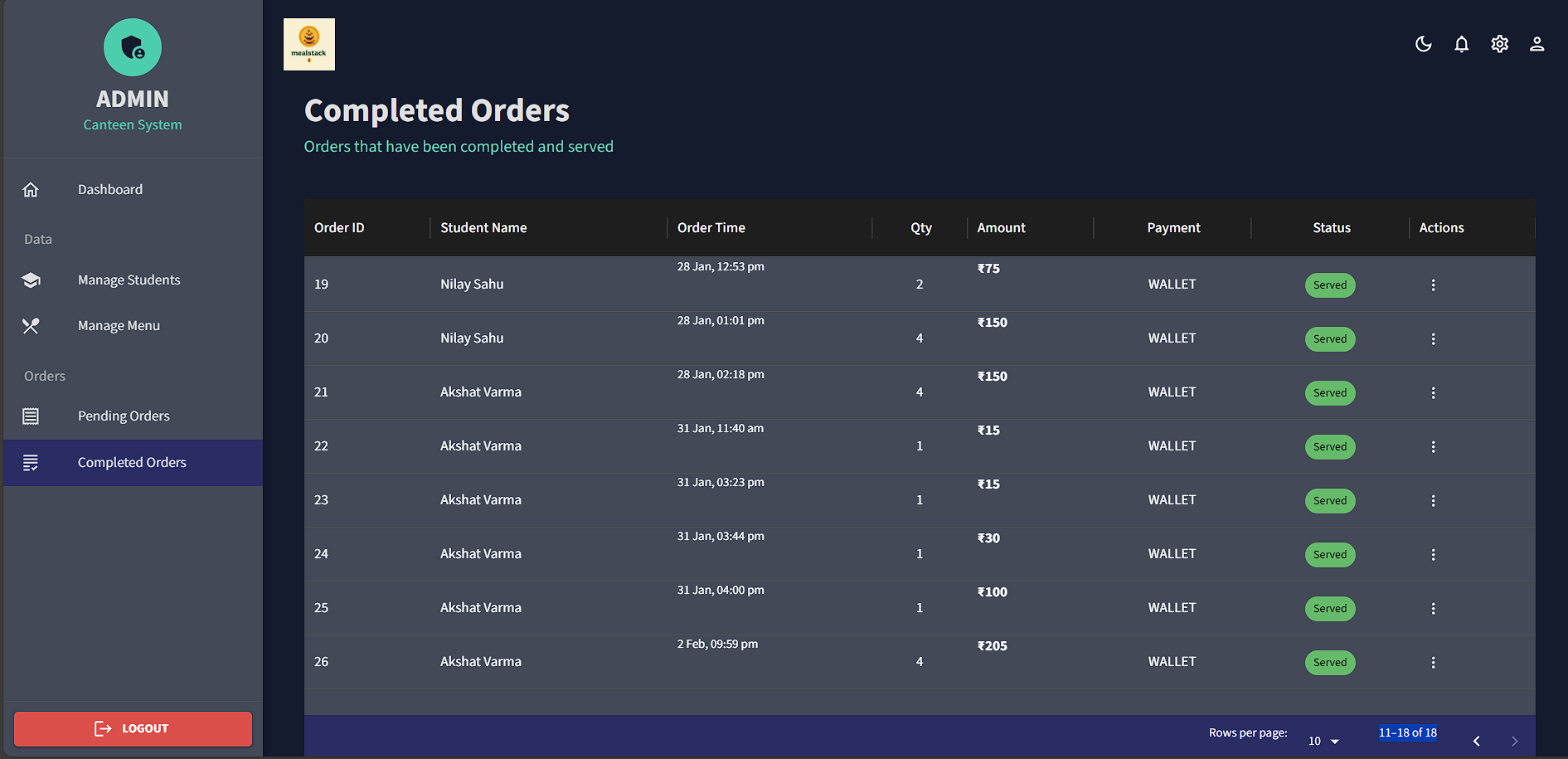




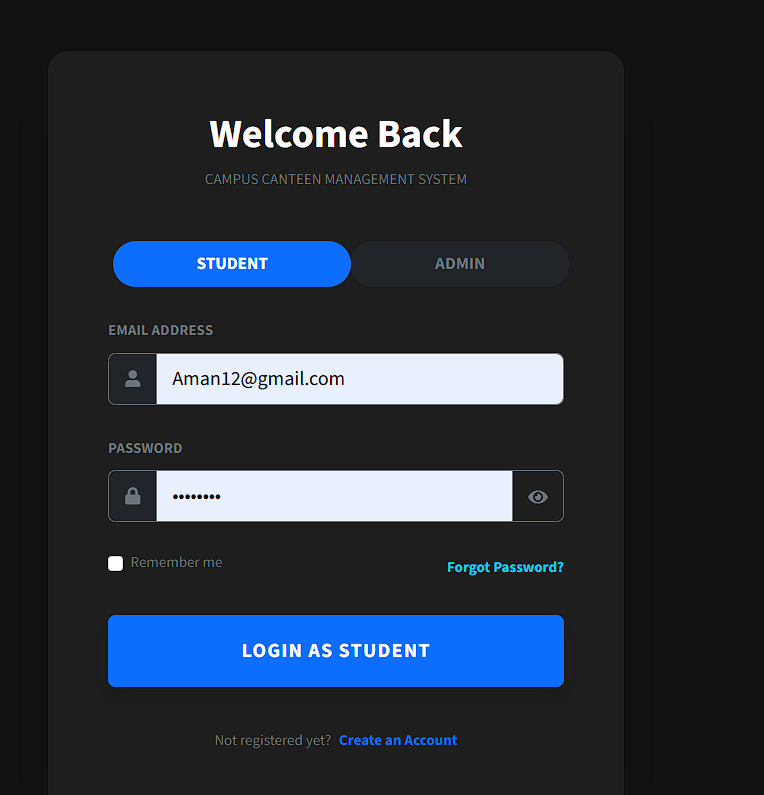


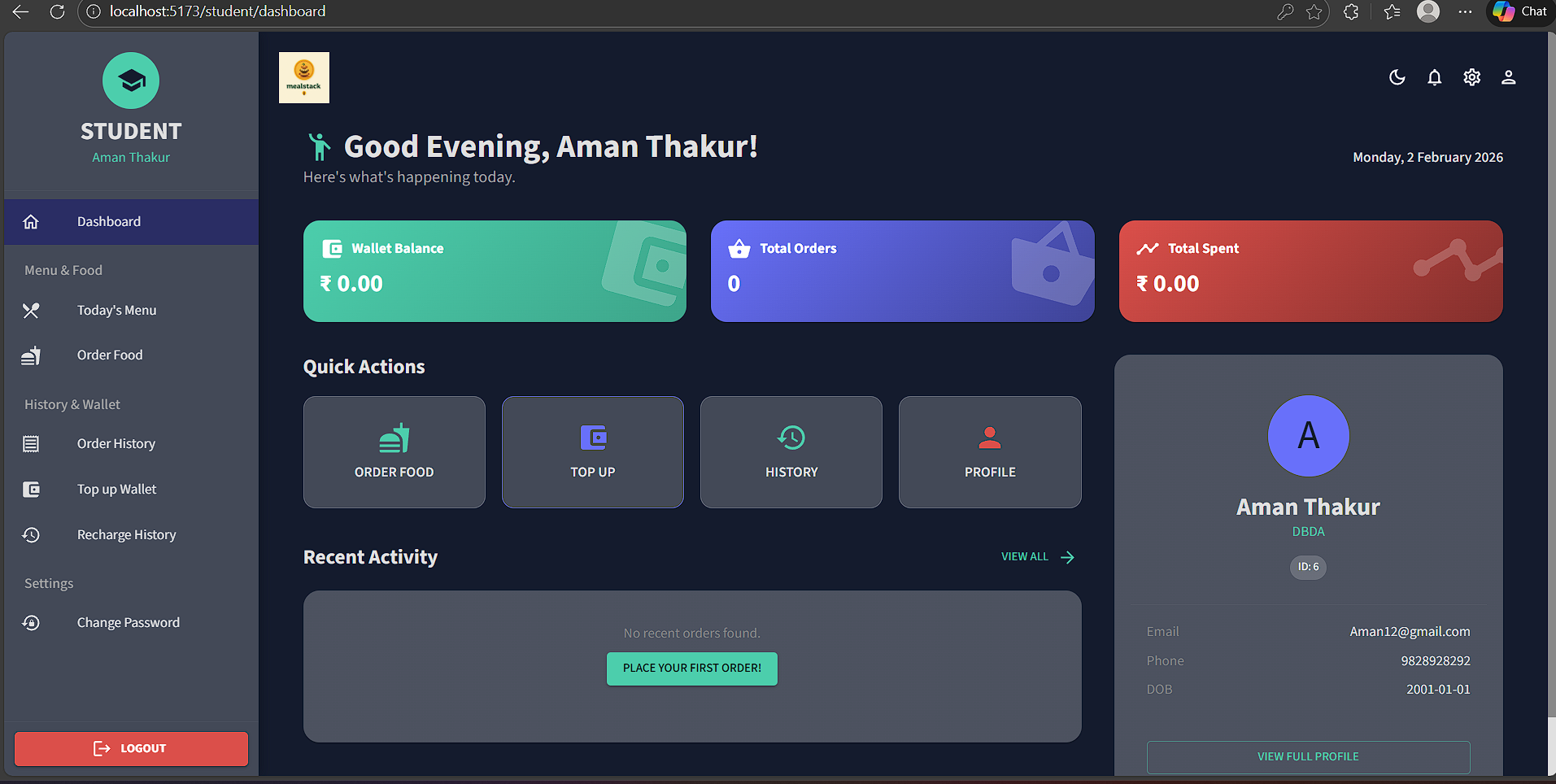


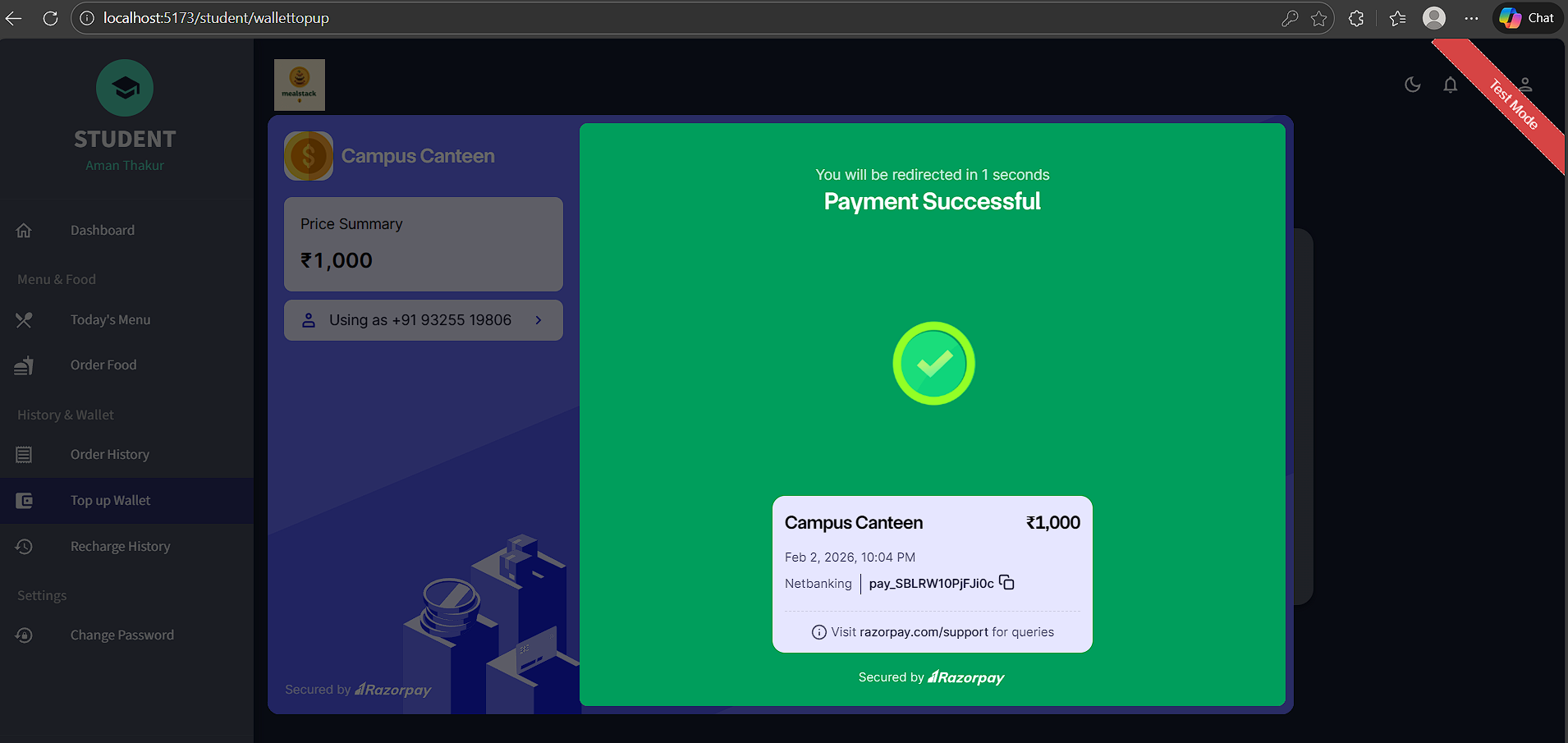


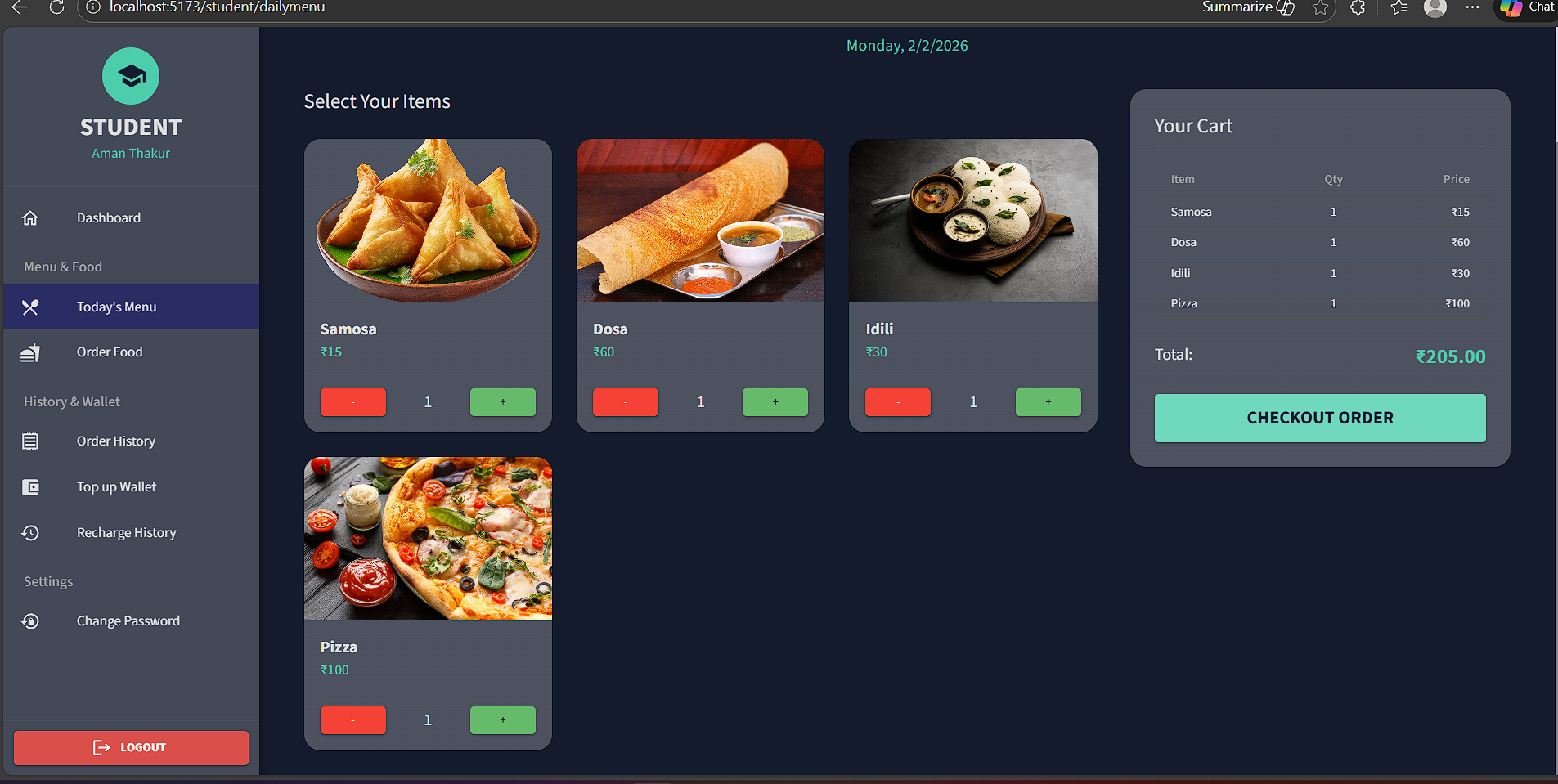


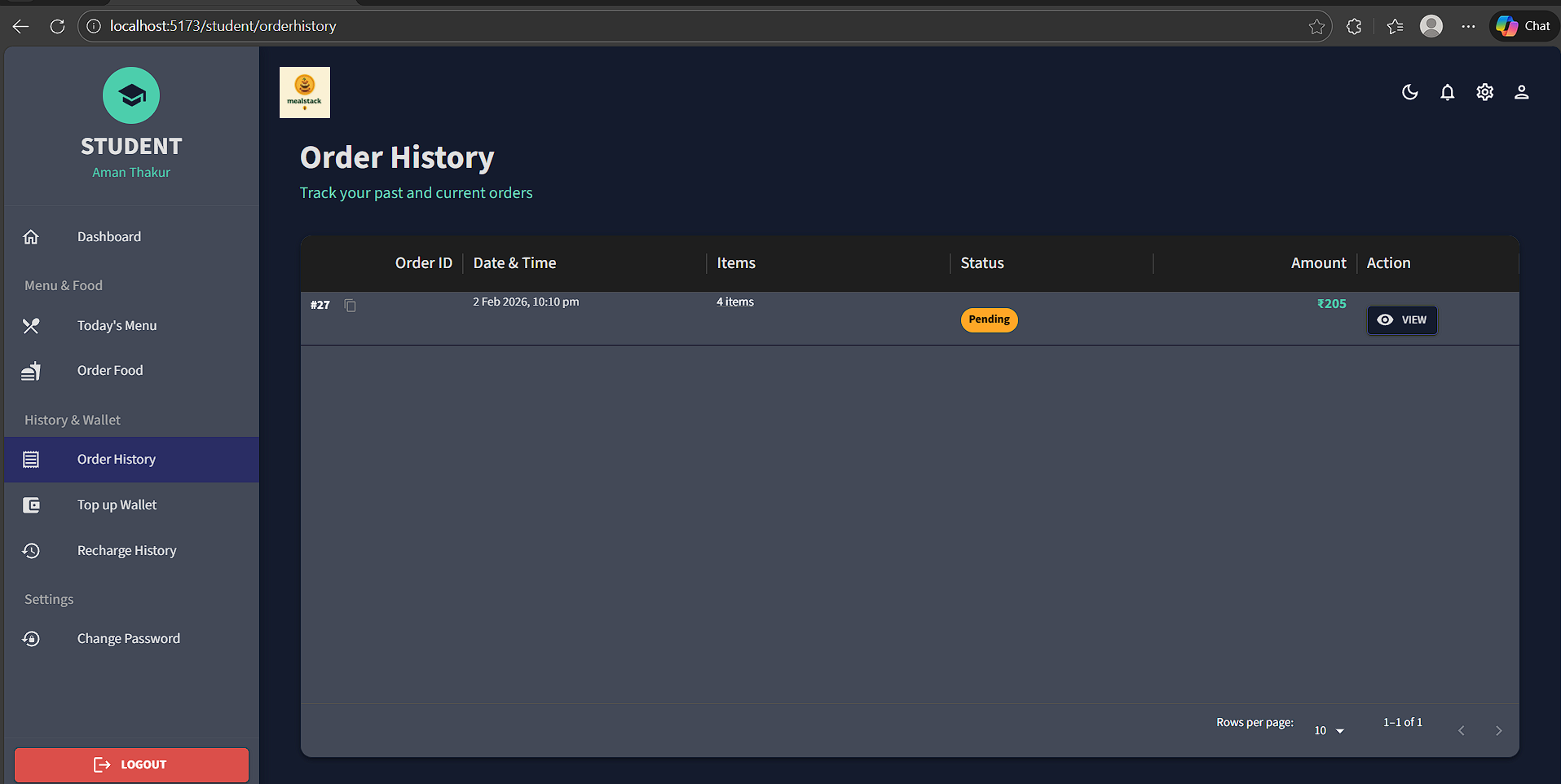
**Student Dashboard:**

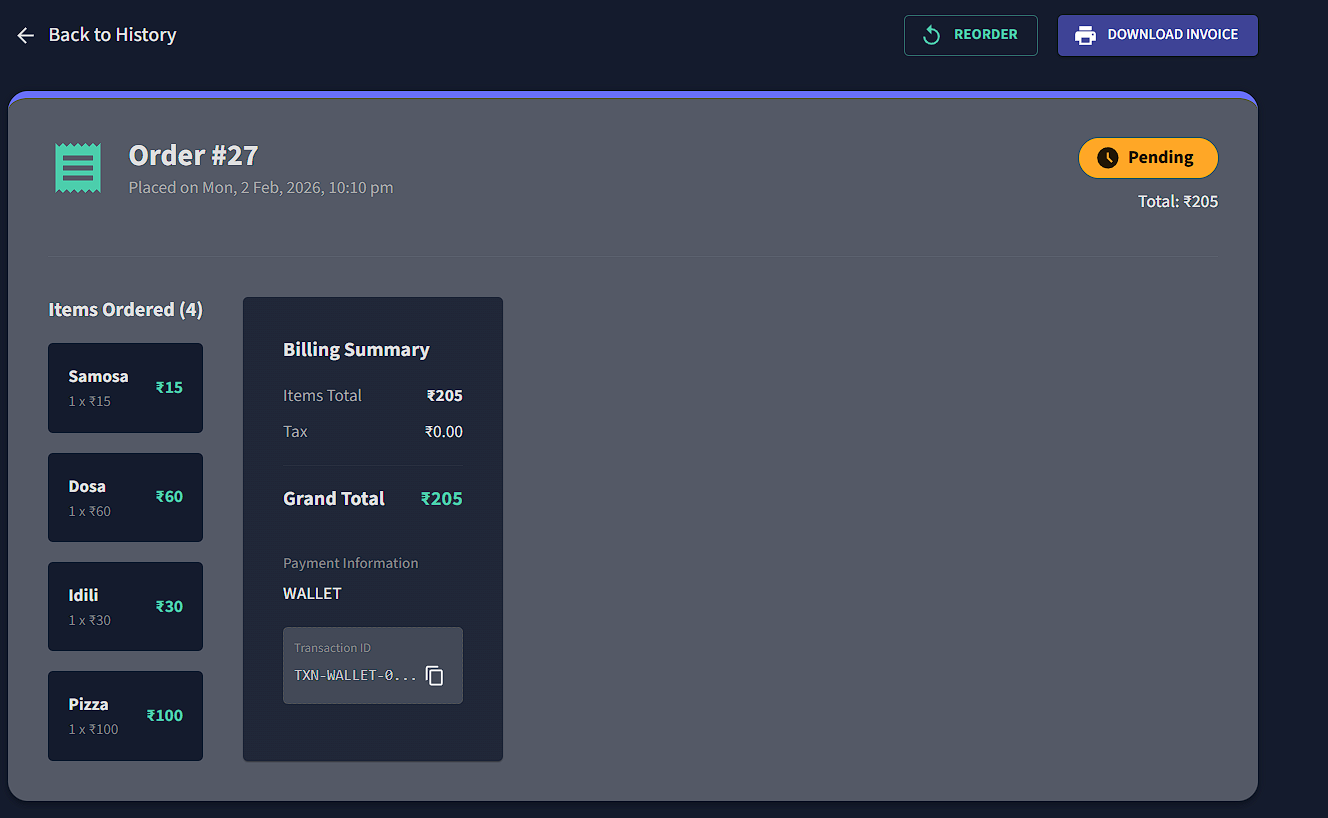


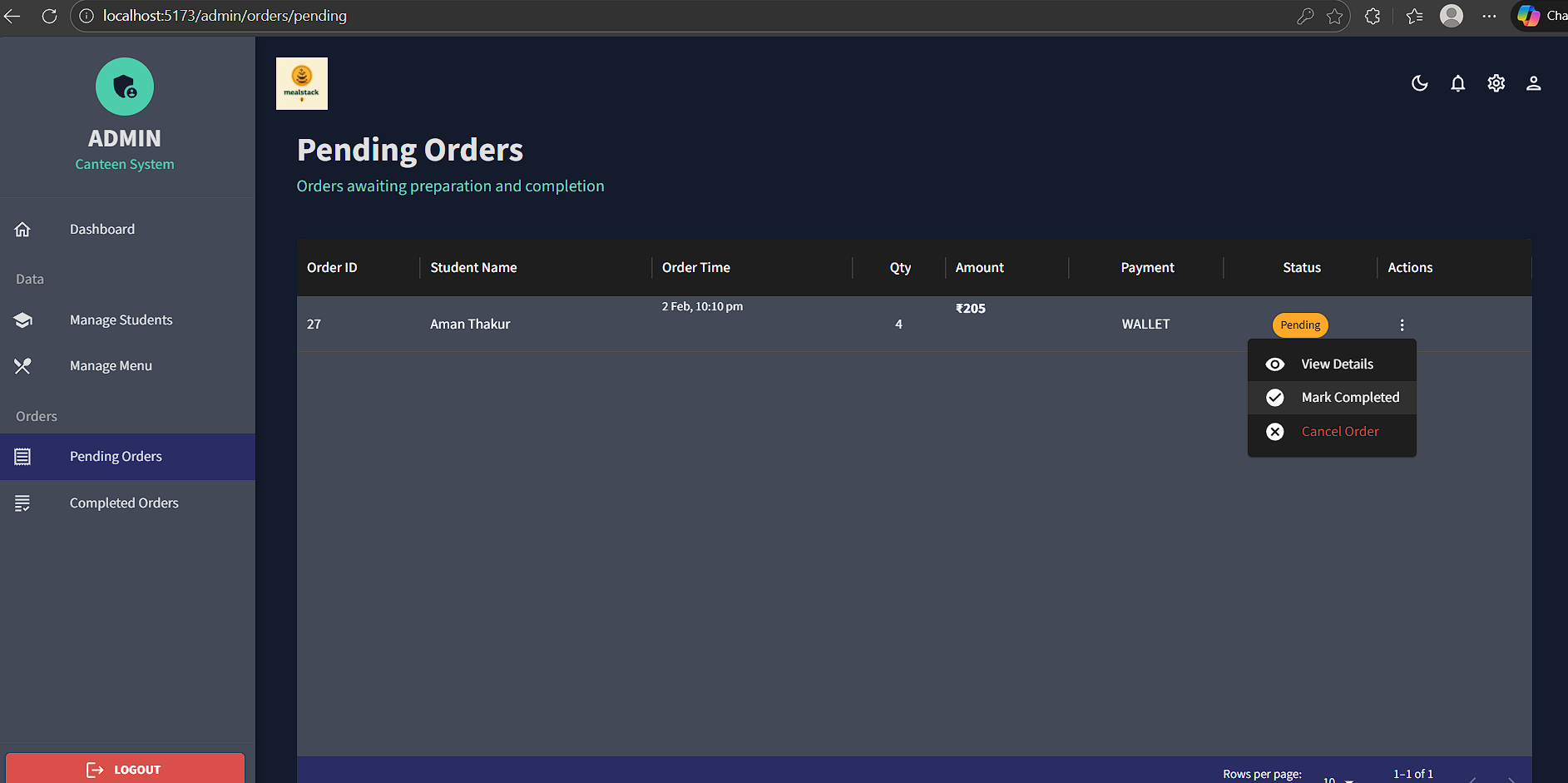


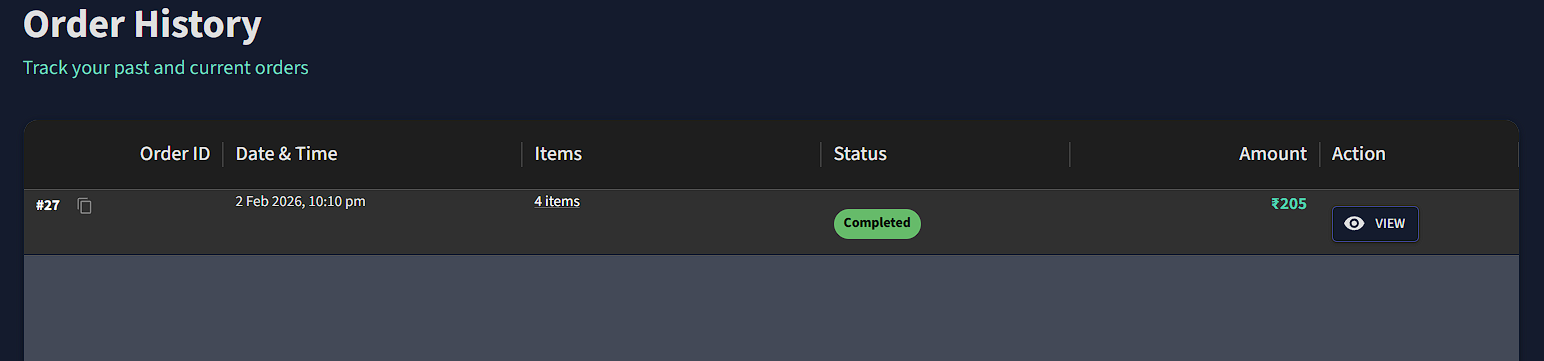












**6.CONCLUSION**

MealStack successfully digitizes canteen operations, providing an efficient platform for food ordering and inventory management in educational institutions.

**Key Achievements:**

• Complete digitization eliminating manual processes and cash handling

• Order processing reduced from minutes to seconds

• Real-time inventory control with automatic updates

• Secure JWT authentication and role-based access

• Intuitive, mobile-responsive interface

• Comprehensive analytics for data-driven decisions

**Impact:**

For Students: Convenient cashless ordering, transparent pricing, reduced wait times, complete transaction history

For Administrators: Reduced manual effort, better inventory control, valuable business insights, eliminated cash reconciliation

For Institution: Improved satisfaction, operational efficiency, successful digital transformation

**Technical Success**:

Effective integration of Spring Boot, React, Redux, Material-UI, and MySQL demonstrating modern full-stack development with RESTful architecture and modular design.

**Future Enhancements**:

QR code pickup, push notifications, AI recommendations, nutrition tracking, mobile apps, multi-canteen support, predictive analytics, ERP integration

MealStack demonstrates successful application of software engineering principles to real-world problems, exceeding functional requirements in usability, security, and performance. The modular architecture ensures easy maintenance and future enhancements.

**7.REFERENCES**

1. Spring Boot Documentation - https://docs.spring.io/spring-boot/

2. Spring Security Reference - https://docs.spring.io/spring-security/

3. React Documentation - https://react.dev/

4. Material-UI Documentation - https://mui.com/

5. "Spring Boot in Action" by Craig Walls, Manning Publications

6. "Learning React" by Alex Banks, O'Reilly Media

7. "Database Systems" by Garcia-Molina, Ullman, Widom

8. Codd, E.F. (1970) "Relational Model of Data", ACM

9. Fielding, R.T. (2000) "REST Architecture", UC Irvine

10. MySQL Documentation - https://dev.mysql.com/doc/

11. JWT.io - https://jwt.io/

12. OWASP Security - https://owasp.org/

13. Baeldung Spring Tutorials - https://www.baeldung.com/

14. MDN Web Docs - https://developer.mozilla.org/

15. Razorpay Documentation - https://razorpay.com/doc