**Restaurant Management System**

**1. Project Title**

Restaurant Management System

**2. Objective**

To design and implement a Java-based Restaurant Management System that streamlines various restaurant operations such as menu handling, order processing, user role management, table bookings, payment processing, and collecting feedback and ratings. The project aims to reduce manual workload, improve customer service, and enhance the overall efficiency of restaurant management.

**3. Scope**

This system caters to restaurants of small to medium scale, aiming to computerize the following areas:

* Menu management (add/edit/delete dishes with prices and images).
* User account creation and role-based login.
* Customer order placement and kitchen status tracking.
* Secure and flexible payment options.
* Table booking and availability tracking.
* Collection and analysis of customer feedback and dish ratings.
* Admin dashboards for monitoring orders, users, and feedback.

**4. Technologies Used**

* Programming Language: Java
* Database: MySQL
* User Interface: JavaFX (Assumed for UI support)
* Styling: CSS
* Security: SHA-256 hashing for password encryption
* Development Tools: Eclipse IDE or IntelliJ IDEA, MySQL Workbench

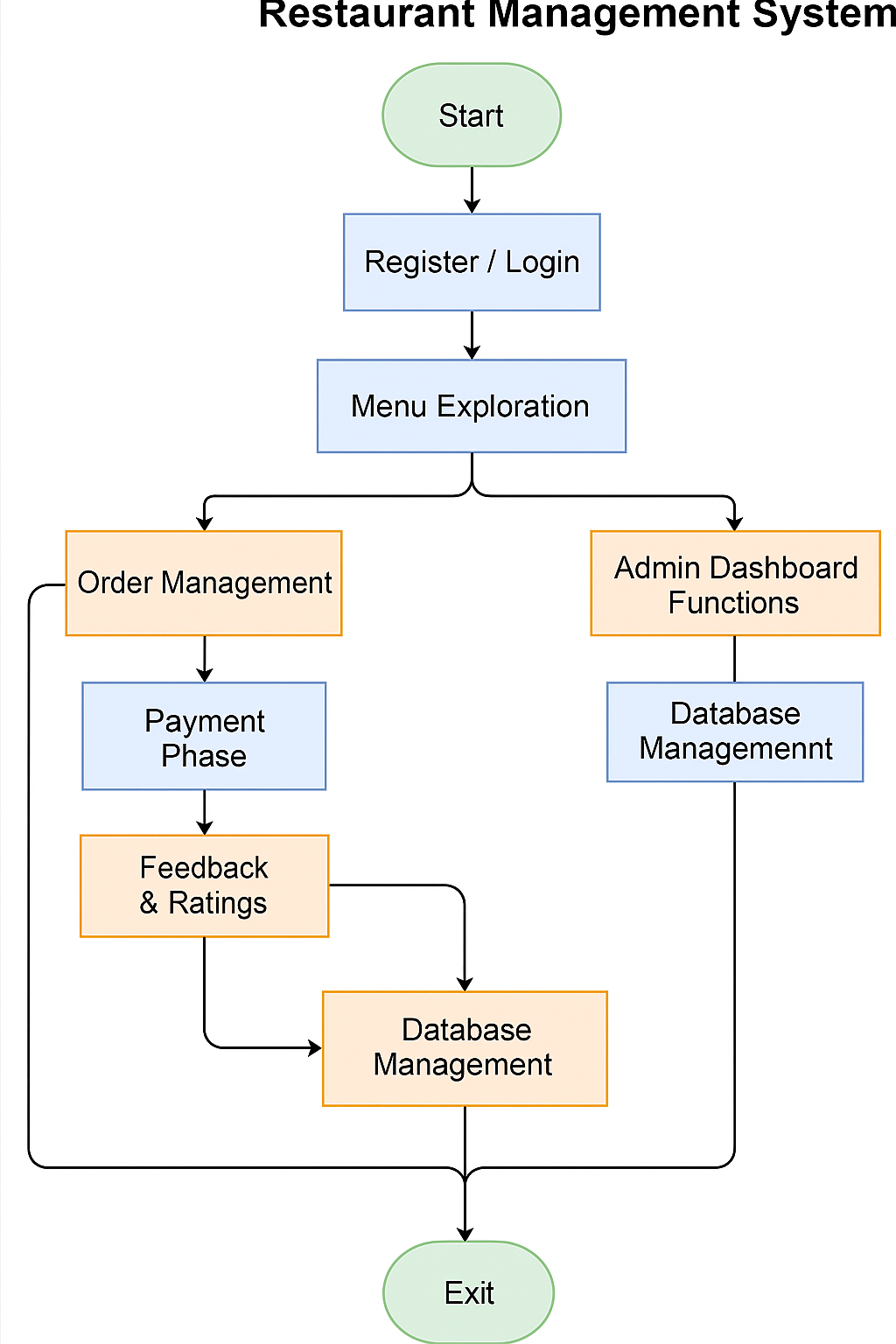
**5. System Requirements**

**Software Requirements**

* JDK 8 or above
* MySQL Server
* Eclipse/IntelliJ IDEA
* MySQL Workbench (optional)
* JavaFX SDK

**Hardware Requirements**

* Processor: Intel i3 or above
* RAM: 4 GB minimum
* Storage: Minimum 500 MB free space



**6. Functional Requirements**

* Admins can add/edit/delete menu items.
* Customers can register and securely log in.
* Customers can browse the menu, place orders, and book tables.
* Real-time update of order statuses by staff.
* Payments are processed with simulated methods (e.g., card, UPI).
* Feedback and ratings can be submitted and viewed.
* Role-based access control across different user types.

**7. Non-Functional Requirements**

* High reliability and availability.
* Data integrity and secure storage of sensitive information.
* Good performance and responsiveness.
* User-friendly and intuitive interface.
* Maintainable and scalable code structure.

**8. Project Modules & Description**

Each module of the system is represented by an individual Java class:

**8.1 Main.java**

* Entry point of the application; initializes components and triggers startup logic.

**8.2 DBConnection.java**

* Provides database connection setup to MySQL using JDBC**.**

**8.3 DatabaseManager.java**

* Core manager class for database transactions: CRUD operations for all entities.

**8.4 MenuItem.java**

* Model class to represent menu dishes with name, price, and optional image URL.

**8.5 User.java**

* Captures user details such as full name, username, encrypted password, and role.

**8.6 PasswordUtil.java**

* Utility class to hash and verify passwords using SHA-256 algorithm.

**8.7 UserRole.java (Enum)**

* Defines three roles: ADMIN, CUSTOMER, STAFF.

**8.8 Order.java**

* Represents a food order. Stores list of items, quantities, and status.

**8.9 OrderStatus.java (Enum)**

* Tracks different order lifecycle states: PLACED, PREPARING, SERVED, CANCELLED**.**

**8.10 PaymentMethod.java & PaymentStatus.java (Enums)**

* Defines modes of payment and their corresponding statuses.

**8.11 CreditCard.java**

* Simulates the use of a credit card for payment processing.

**8.12 TableBooking.java**

* Allows reservation of restaurant tables by users.

**8.13 TableType.java (Enum)**

* Distinguishes types of tables: SINGLE, FAMILY, OUTDOOR.

**8.14 Feedback.java**

* Collects customer feedback content and links it to user and order IDs.

**8.15 DishRating.java**

* Allows rating of individual dishes using a star-based system.

**8.16 style.css**

* Provides CSS-based styling for UI components.

**9. Database Schema Overview**

The database includes the following main tables:

* users: Stores user login and profile details.
* menu\_items: Stores dishes and prices.
* orders: Contains customer order details.
* feedback: Contains customer comments.
* dish\_ratings: Tracks dish ratings.
* table\_bookings: Stores reservation information.

**10. Security Measures**

* Passwords are securely hashed using SHA-256.
* Role-based access ensures separation of responsibilities.
* Input validations and database abstraction recommended to prevent SQL injection.

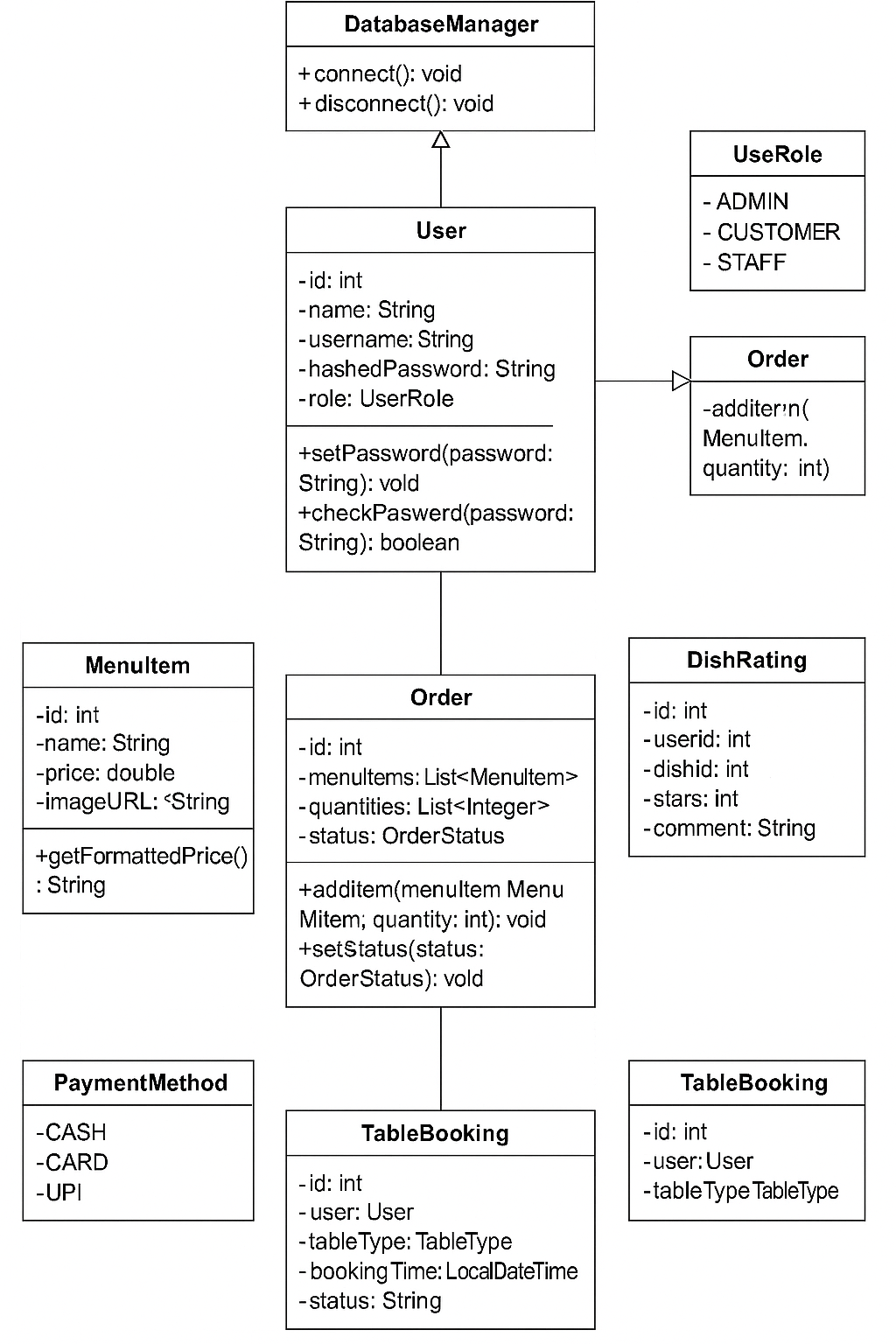
**11. User Roles**

* Admin: Full privileges including menu and user management.
* Customer: Can browse menu, place orders, and give feedback.
* Staff: Can update order status and view bookings.

**12. Use Case Diagram & Explanation**

The following actors interact with the system:

* **Admin**:
  + Manage users (add/edit/remove)
  + Manage menu items
  + View all orders and bookings
  + Analyze feedback and ratings
* **Customer**:
  + Register/Login
  + Browse menu and place orders
  + Book tables
  + Make payments
  + Submit feedback and ratings
* **Staff**:
  + View and update order status
  + Check and manage table bookings



**12. Future Enhancements**

* Integrate a full-featured JavaFX GUI.
* Enable real-time order tracking using sockets.
* Add user analytics and sales reports.
* Develop mobile app version.
* Connect with online food delivery APIs (Zomato/Swiggy).
* Add billing and invoice printing module.

**13. Installation & Setup Guide**

1. Install JDK and preferred IDE (e.g., Eclipse).
2. Install MySQL and create a project database.
3. Import necessary tables and seed data.
4. Update database credentials in DBConnection.java.
5. Compile and run Main.java.
6. (Optional) Add JavaFX dependencies for UI integration.

**14. Conclusion**

This project serves as a foundational tool for understanding how restaurant operations can be managed digitally using Java. It covers key operations such as order management, menu control, user role segregation, and basic customer interaction. The system is built using modular OOP design, providing flexibility to scale and integrate additional features such as web interfaces or mobile apps.