****

**Tribhuvan University**

**Institute of Science and Technology**

**A Project Report On**

**“Restaurant Management System”**

**Submitted to:**

Department of Computer Science and Information Technology

**Asian College of Higher Studies**

Ekantakuna, Lalitpur

***In partial fulfillment of the requirements***

***For the Bachelors of Science in Computer Science and Information Technology***

**Submitted by:**

Biplove Gautam(79011059)

Laya Pandey(79011066)

Shishir Aryal(79011085)

Babin Rana (79011057)

Semester: VI

**November 3, 2025**

**Under the supervision of:**

Mr. Bidur Sapkota

**Tribhuvan University**

**Institute of Science and Technology**

****

**SUPERVISOR’S RECOMMENDATION**

I hereby recommend that this project is prepared by **Biplove Gautam, Laya Pandey, Shishir Aryal** and **Babin Rana** under the supervision of **Mr. Bidur Sapkota** entitled “Restaurant Management System” in partial fulfillment of the requirements for the degree of Bachelor of Computer Science and Information Technology be processed for the final evaluation.

---------------------------------------

Bidur Sapkota

Project Supervisor

Department of Computer Science and Information Technology

# LETTER OF APPROVAL

This is to certify that this project is prepared by Biplove Gautam, Laya Pandey, Shishir Aryal and Babin Rana entitled “Restaurant Management System” in partial fulfillment of the requirements for the degree of Bachelor in Computer Science and Information Technology has been evaluated. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

**….………………………….. …………………………**

**Mr. Bidur Sapkota, ACHS External Examiner**

# ACKNOWLEDGEMENT

We would like to extend our deepest gratitude to **Mr. Bidur Sapkota** for his invaluable assistance and thorough review of this report. His insightful feedback and unwavering support have been instrumental in shaping the final outcome, and we are truly appreciative of his time and expertise.

We also wish to thank our friends and all the individuals who contributed in various ways, whether through encouragement, discussions, or practical help. Your collective efforts have made this work possible, and we are grateful for your kindness and generosity.

**With respect,**

Biplove Gautam

Laya Pandey

Shishir Aryal

Babin Rana

# ABSTRACT

The Restaurant Management System is a Spring Boot application that uses a RESTful API supported by JPA and MySQL to manage menu items, staff, tables, and customer orders. This report details the design, implementation, and testing of the system. Controllers, services, and repositories are all part of the system's MVC layered architecture. The requirements, system analysis, object-oriented design artifacts (ER, class, sequence diagrams etc), implementation details, tests, and appendices with important code snippets are all included in the report.  
  
Keywords: JPA, REST API, OOP, UML, Spring Boot, Restaurant Management.

**Table Of Contents**

**[LETTER OF APPROVAL III](#_Toc32178)**

**[ACKNOWLEDGEMENT IV](#_Toc9930)**

**[ABSTRACT V](#_Toc13096)**

**[LIST OF ABBREVIATIONS VII](#_Toc16588)**

**[LIST OF FIGURES VIII](#_Toc25243)**

**[Chapter 1. Introduction 1](#_Toc13355)**

[1.1 Introduction 1](#_Toc5366)

[1.2 Objectives 1](#_Toc17104)

[1.3 Scope and Limitations 1](#_Toc5762)

[1.4 Development Methodology 1](#_Toc23004)

[1.5 Report Organization 3](#_Toc4589)

**[Chapter 2. System Analysis 4](#_Toc27407)**

[2.1 System Analysis 4](#_Toc1175)

[2.1.1 Requirement Analysis 4](#_Toc3681)

[2.1.2 Feasibility Analysis 6](#_Toc24145)

[2.1.3 Analysis 6](#_Toc5452)

**[Chapter 3. System Design 12](#_Toc28646)**

[3.1 Component Diagram 12](#_Toc17829)

**[Chapter 4. Implementation and Testing 13](#_Toc10031)**

[4.1 Implementation 13](#_Toc24561)

[4.1.1 Tools Used 13](#_Toc19580)

[4.2 Testing 14](#_Toc30098)

[4.2.1 Test Cases for Unit Testing 14](#_Toc21661)

[4.3 Result Analysis 17](#_Toc30025)

**[5. Work Done and Conclusion 18](#_Toc13260)**

[5.1 Work Done 18](#_Toc18655)

[5.2 Conclusion 18](#_Toc16707)

**[REFERENCES 19](#_Toc29935)**

**[Appendices 20](#_Toc30801)**

# LIST OF ABBREVIATIONS

API : Application Programming Interface

CRUD : Create, Read, Update, Delete

DB : Database

DTO : Data Transfer Object

JPA : Java Persistence API

REST : Representational State Transfer

# LIST OF FIGURES

Figure 1.1 Agile Methodology

Figure 2.1 Use Case Diagram

Figure 2.2 Data Modeling using ER Diagram

Figure 2.3 Context Diagram

Figure 2.4 Process Modeling using DFD level 1

Figure 2.5 Object Modeling using Class Diagram

Figure 2.6 Class Diagram for Controllers, Services, Repository and DTO

Figure 2.7 Dynamic Modeling using Sequence Diagram

Figure 3.1 Component Diagram for Restaurant Management System

# 

# Chapter 1. Introduction

## Introduction

The Restaurant Management System is a backend app built with Spring Boot (Java 17) that handles the normal tasks of running a restaurant, like managing menu items, orders, tables, and staff. It gives you access to CRUD operations through REST endpoints and uses Spring Data JPA to store data in a MySQL database.

## Objectives

* Provide RESTful APIs to create, retrieve, update and delete MenuItems, OrderLists, Tables and Employees.
* Implement business rules and validations (uniqueness checks, existence checks) and global exception handling.
* Model the domain using OOP principles (entities, services, repositories).

## Scope and Limitations

**Scope:**

1. Backend API for running a restaurant.
2. JPA mapping and chains for orders and items on orders.

**Limitations:**

1. There isn't an authentication or authorization layer in place yet (the security dependency is commented out in the pom).
2. The app stores your credentials. Secret management based on the environment is suggested for production; properties in development.

## Development Methodology

The project follows an agile approach:

**1. Requirements (User Stories)**: Capture user needs as short, simple User Stories in the Product Backlog. Prioritized by the Product Owner based on value and feedback. Continuously refined and updated throughout the project.

**2. Design**: Create lightweight designs (wireframes, architecture) just for the current sprint. Focus on just enough to start development, not full upfront planning. Collaborative effort involving designers, developers, and stakeholders.

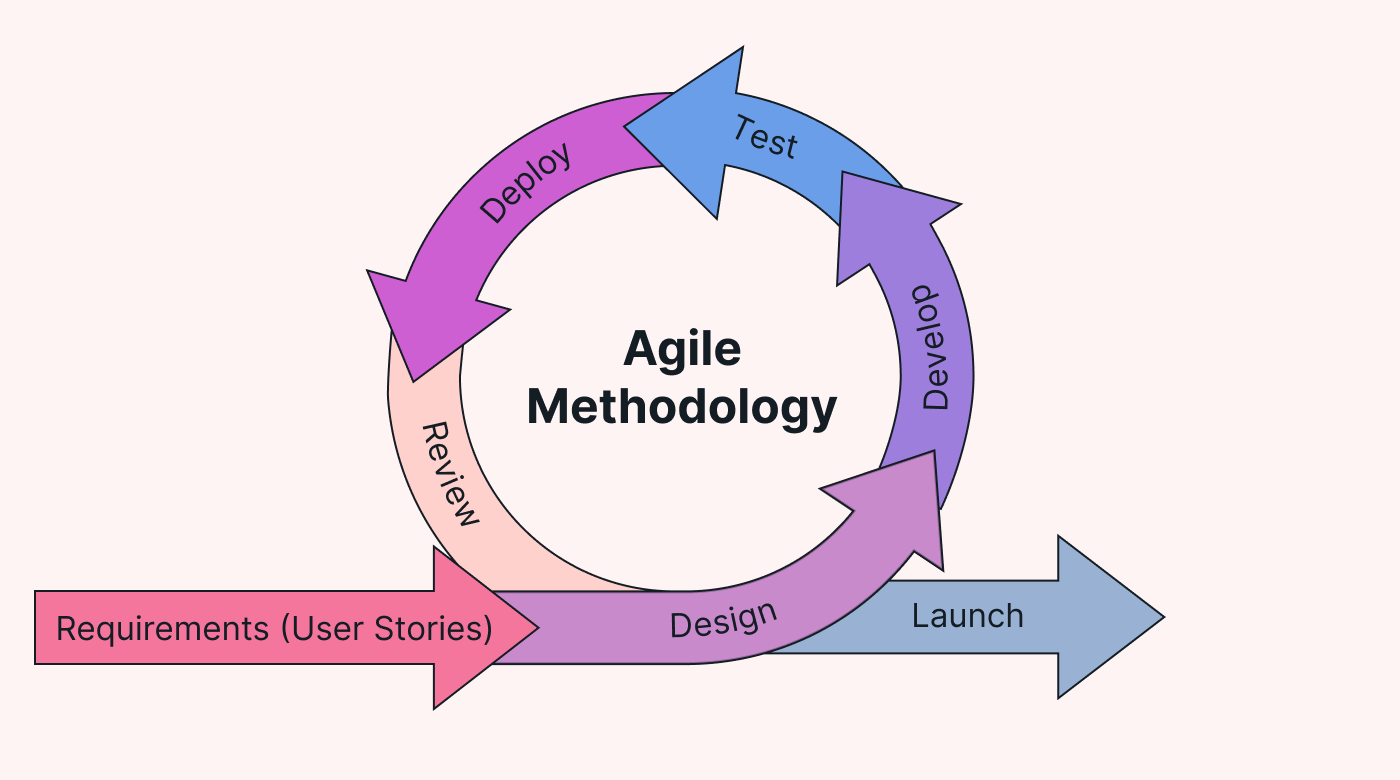
**3. Develop**: Team builds working software in short sprints (1–4 weeks). Daily stand-ups keep progress aligned and blockers resolved quickly. Code is written following the Definition of Done (DoD).

**4. Test Run**: unit, integration, and automated tests during the sprint. Quality is ensured before marking a story as complete. Includes exploratory and acceptance testing with stakeholders.

**5. Deploy**: Use CI/CD pipelines to deploy code to staging or production frequently. Features can be released incrementally or toggled with feature flags. Goal: Deliver working software to users often and safely.

**6. Review**: A quick look back is done and a lot of things are checked whether any changes are required. If any changes are required then the loop is done again for a number of times.

**7. Launch**: Represents a major release or public rollout of the product. Not a one-time event — ongoing cycles continue post-launch.



**Figure 1.1 Agile Methodology**

The codebase follows layered architecture : Controller → Service → Repository.

## Report Organization

* **Chapter 1: Introduction** outlines the problem statement, objectives, scope, limitations, and the Agile development methodology adopted for the project.
* **Chapter 2: System Analysis** details the functional and non-functional requirements, feasibility study (technical, operational, economic), requirement analysis, use case diagram, ER diagram, and data flow diagrams (context and Level 1).
* **Chapter 3: System Design** describes the component diagram.
* **Chapter 4: Implementation and Testing** explains the tools and technologies used (Java, MySQL, IntelliJ IDEA, Draw.io, Postman, Swagger), module descriptions (Employee, Menu Item, Order, Table, Billing, Report), testing procedures, unit tests, and result analysis with screenshots.
* **Chapter 5: Work Done and Conclusion** summarizes the achievements, and overall conclusions of the project.

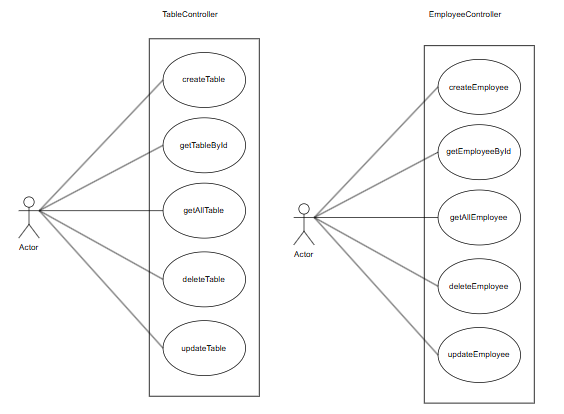
# Chapter 2. System Analysis

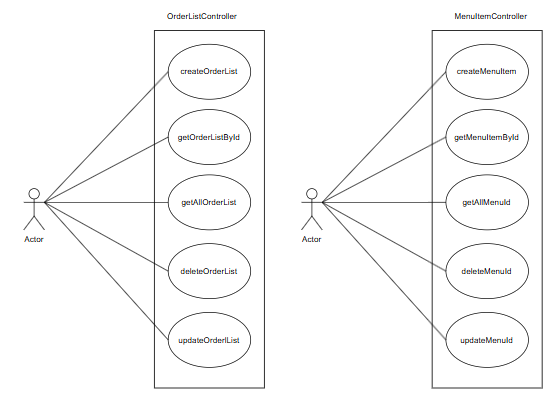
## 2.1 System Analysis

Performing a system analysis for a Restaurant Management System involves examining different parts of the system which will ensure it meets the requirement for both the client and the staff.

### 2.1.1 Requirement Analysis

1. **Functional Requirement**

****

****

**Figure 2.1 Use Case Diagram**

A use case diagram depicted above shows the interaction between a single Actor and four distinct system entities: Employee, MenuItem, OrderList, and Table.

The Actor manages Employee records by performing createEmployee, getEmployeeByID, getAllEmployee, deleteEmployee, and updateEmployee operations.

Similarly, the Actor handles MenuItem management through createMenuItem, getMenuItem, getAllMenuItem, deleteMenuItem, and updateMenuItem.

For OrderList, the Actor can createOrderList, getOrderListById, getAllOrderList, deleteOrderList, and updateOrderList.

Finally, the Actor controls Table resources with createTable, getTableById, getAllTable, and deleteTable actions.

This diagram represents a restaurant management system where the Actor performs full CRUD (Create, Read, Update, Delete) operations across core entities: staff, menu, orders, and tables.

**II. Non Functional Requirement**

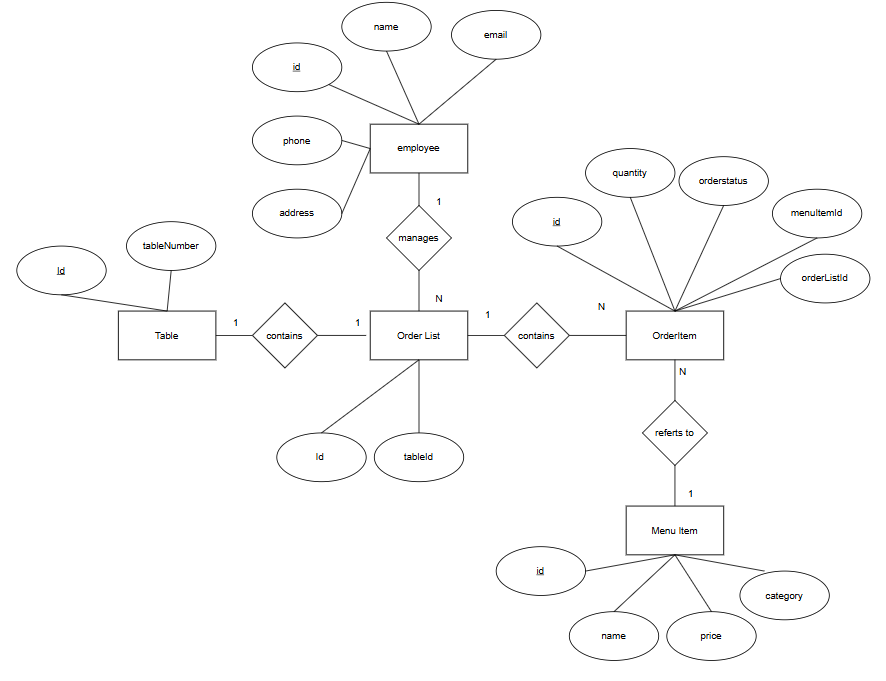
1. **Performance:** API responses should be under 200 ms under normal load (target).
2. **Maintenance:** Layered structure and DTOs for controller/service separation.
3. **Reliability:** Transactional handling for order creation; cascading ensures consistency.

### 2.1.2 Feasibility Analysis

1. **Technical:** The project uses Spring Boot 3.5.4, Java 17, Spring Data JPA and MySQL. It is technically feasible on typical development machines and on any server with Java 17 and MySQL.
2. **Operational:** The REST API can be consumed by web frontends or mobile apps. Deployment requires a Java runtime and database.
3. **Economic:** Low-cost to operate with existing infrastructure; main costs are hosting DB and server.
4. **Schedule:** Basic MVP with core CRUD and order flows can be delivered in a short sprint (2–4 weeks). Additional features (auth, reporting) require extra time.

### 2.1.3 Analysis

**I. Data Modeling using ER Diagrams**

****

**Figure 2.2 Data Modeling using ER Diagram**

The ER diagram describes the relationship between the entities in a restaurant management system, showing how core business components are interconnected through one-to-many and many-to-one relationships.

In the system, one Employee manages multiple Order Lists, represented by a one-to-many relationship via the **managed** association. Each Order List is linked to one Table, forming a many-to-one relationship through the contains association (one table can have many order lists, but each order list belongs to one table).

Each Order List contains multiple Order Items, represented by a one-to-many relationship via the contains association. Every Order Item refers to exactly one Menu Item, forming a many-to-one relationship through the refers to association (many order items can reference the same menu item).

All entities have attributes with primary keys ensuring uniqueness and enabling relationships:

**Employee:** id (PK), name, email, phone, address

**Table:** id (PK), tableNumber

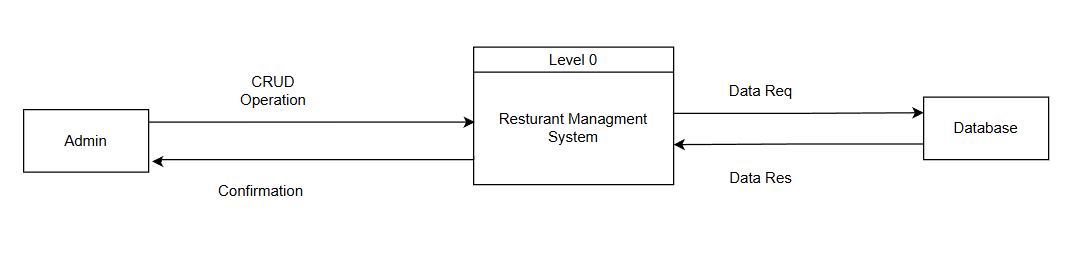
**Order List**: id (PK), tableId (FK)

**Order Item:** id (PK), quantity, orderStatus, menuItemId (FK), orderListId (FK)

**Menu Item:** id (PK), name, price, category

This structure supports order tracking, menu management and table assignment in a restaurant environment.

**II. Context Diagram**

****

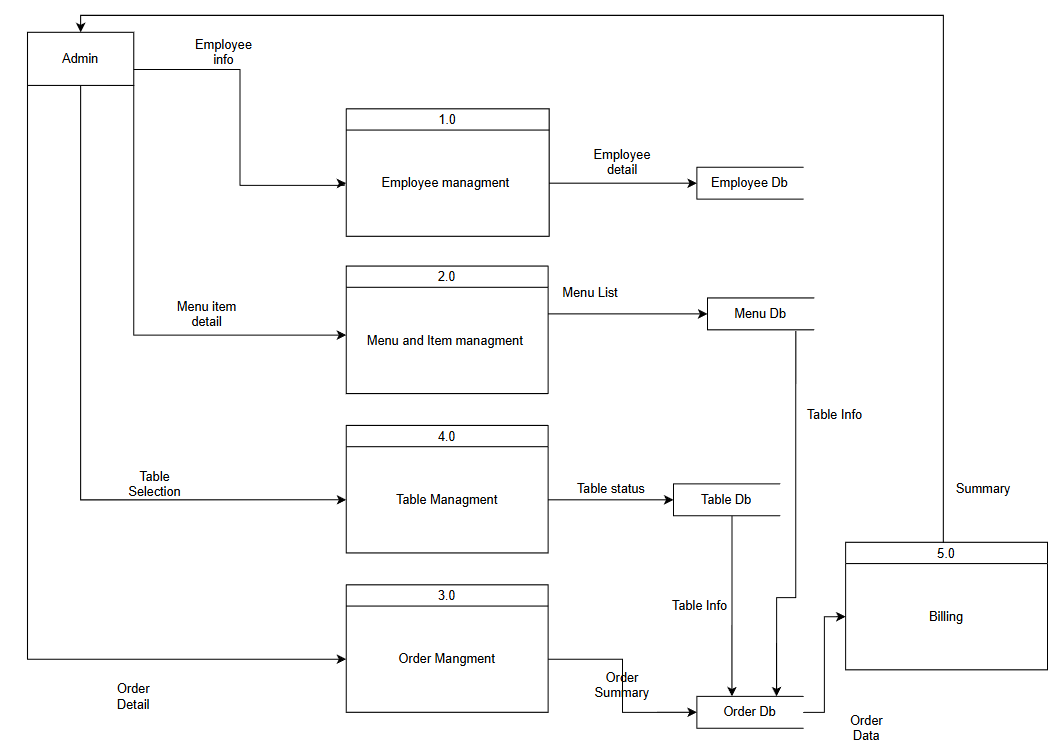
**Figure 2.3 Context Diagram**

An overview of the admin's interactions with the restaurant management system is shown in this diagram.

While the system exchanges data requests and responses with the database, the administrator completes CRUD activities and receives confirmation.

The system is represented as a single process with external data flow in this Level 0 (Context) DFD.

**III. Process Modeling using DFD level 1**

****

**Figure 2.4 Process Modeling using DFD level 1**

The process begins with the Admin providing inputs to manage system components.

**The Employee Management** (1.0) process accepts Employee info and stores/retrieves data via Employee Db.

**Menu and Item Management** (2.0) handles Menu item detail and maintains records in Menu Db.

**Order Management** (3.0) processes Order Detail, generates Order Summary, and interacts with Order Db.

**Table Management** (4.0) receives Table Selection, updates Table status, and accesses Table Db.

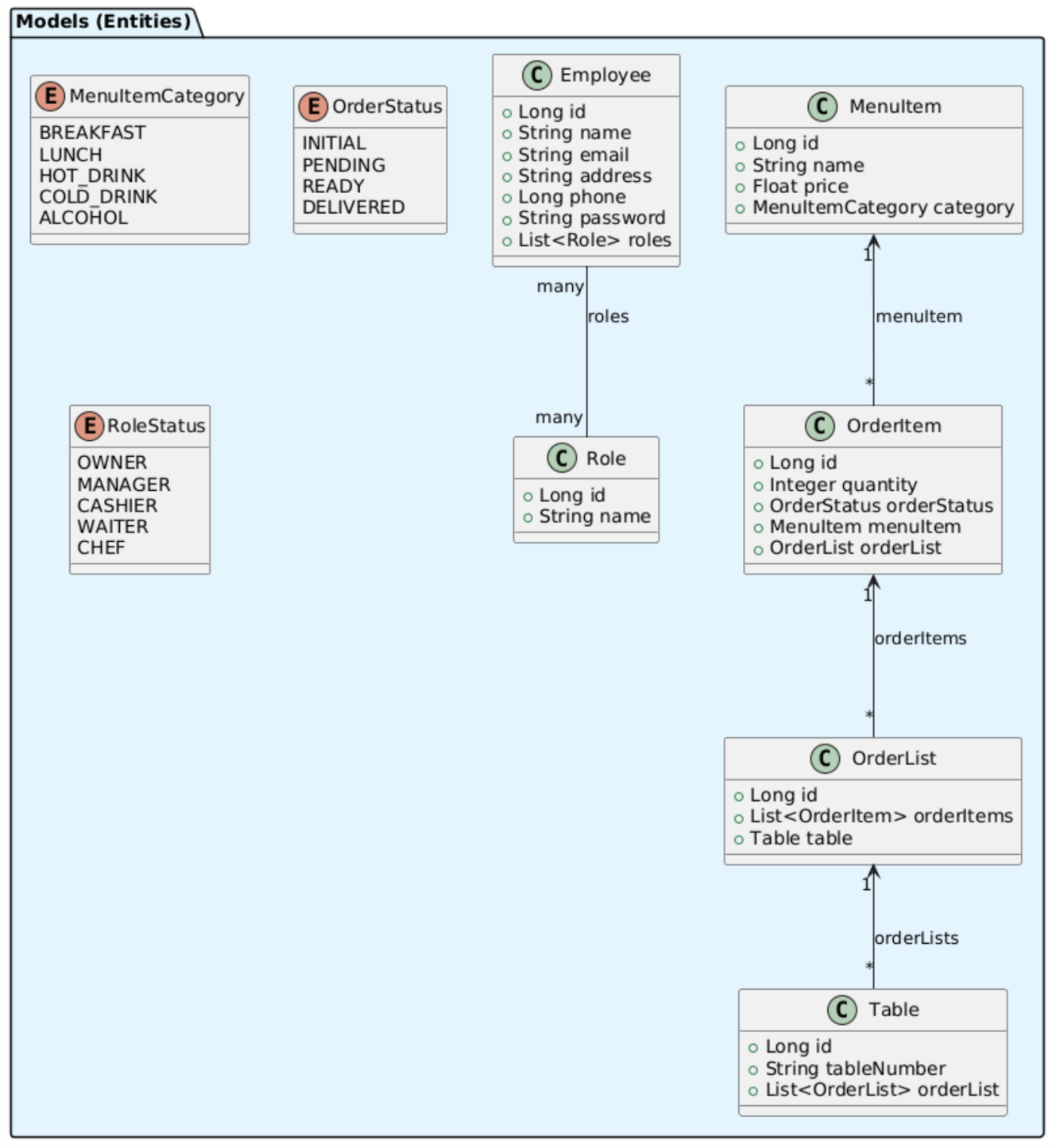
**Billing** (5.0) uses Order Data and Table info to generate the final bill.

This Level 1 DFD decomposes the restaurant system into core functional processes with data flow between admin, processes, and databases.

It ensures efficient tracking of staff, menu, orders, tables, and payments in real time.

All processes operate independently but share data through centralized databases.

**IV. Class Diagram for Entities**

****

**Figure 2.5 Object Modeling using Class Diagram**

This class diagram models the core entities of a Restaurant Management System using object-oriented design.

Employee has attributes (`id`, `name`, `email`, etc.) and holds many Roles, while each Role links to one RoleStatus (e.g., WAITER, CHEF).

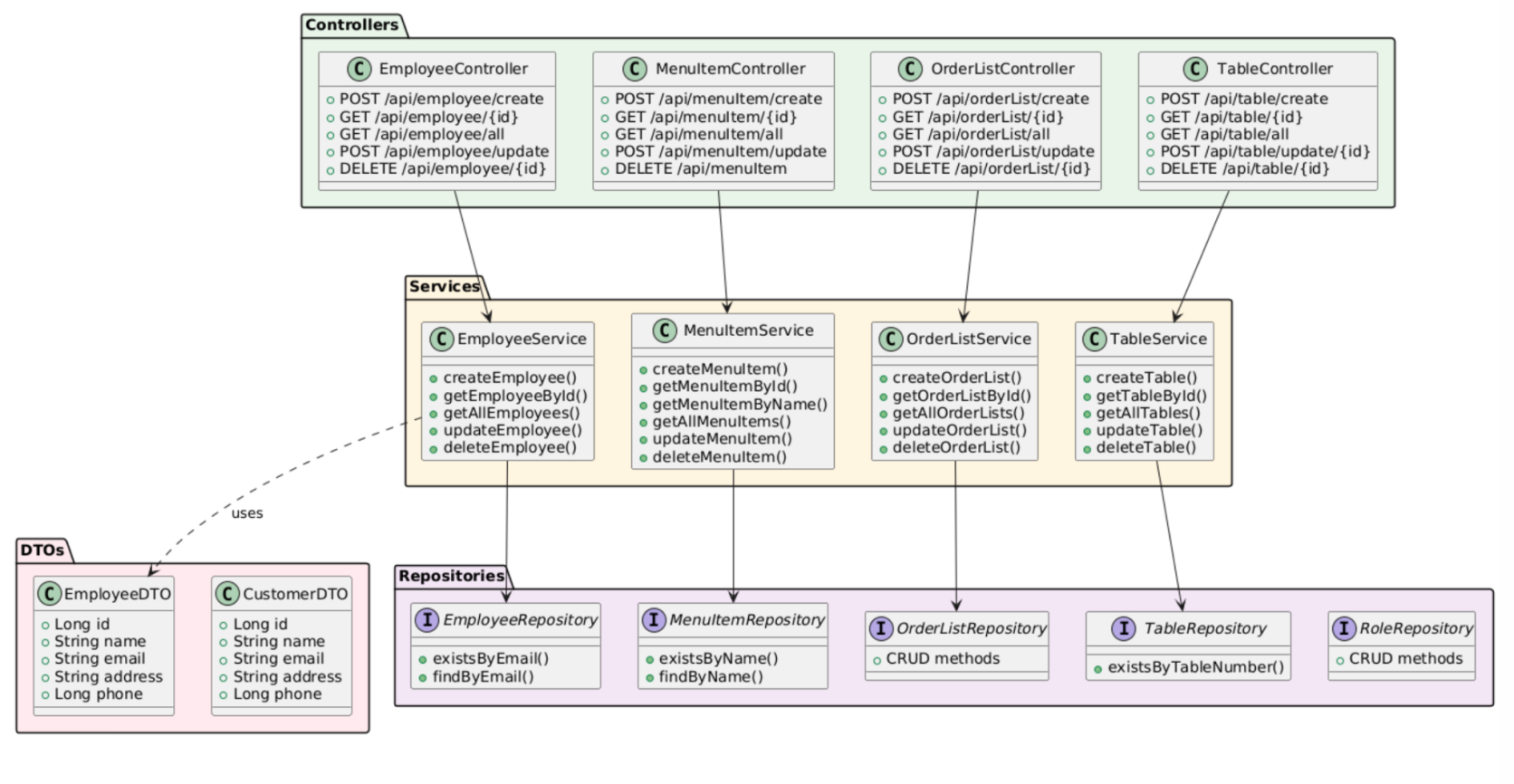
MenuItem belongs to a MenuItemCategory (e.g., HOT\_DRINK) and is part of many OrderItems, which track `quantity` and `orderStatus`.

OrderItem connects to OrderList, which aggregates multiple items and links to a Table via `tableNumber`.

The diagram supports role-based access, menu categorization, order tracking, and table assignment with clear one-to-many and many-to-many relationships.

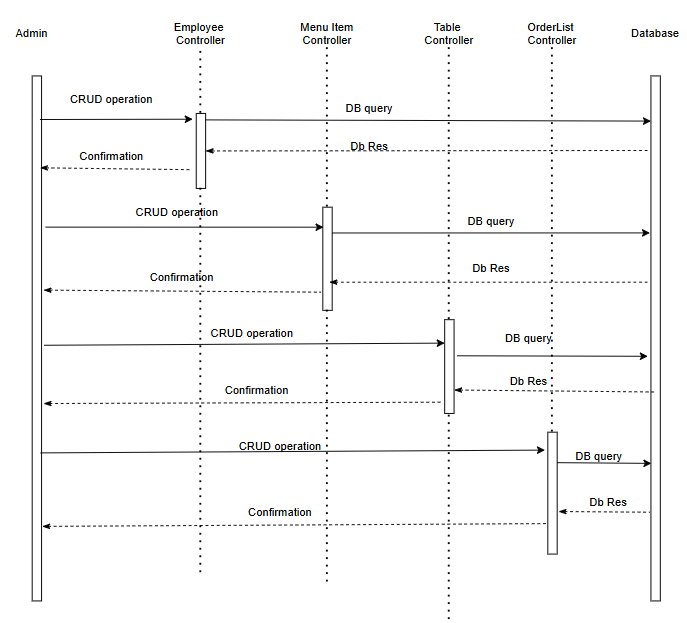
It forms the foundation for implementing business logic in an object-oriented programming language.

**V. Class Diagram for Controllers, Services, Repositories and Data Transfer Object (DTO)**



**Figure 2.6 Class Diagram for Controllers, Services, Repository and DTO**

**VI. Dynamic Modeling using Sequence Diagram**

****

**Figure 2.7 Dynamic Modeling using Sequence Diagram**

This sequence diagram illustrates the Admin performing CRUD operations on Employee, Menu Item, Table, and OrderList entities via dedicated Controllers.

The Admin sends a CRUD operation to the respective Controller, which forwards a DB query to the Database.

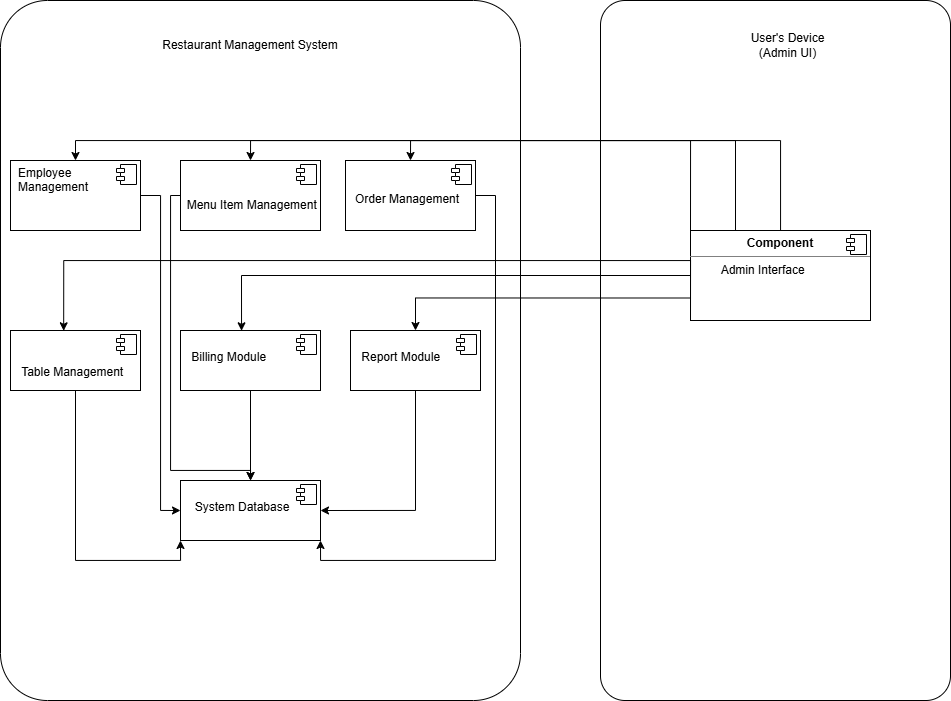
The Database processes the query and returns a DB Response to the Controller, which then sends a Confirmation back to the Admin.

This interaction repeats sequentially for each entity type, ensuring data consistency and user feedback.

It models a centralized, controller-based architecture with clear separation of concerns and database interaction.

# Chapter 3. System Design

## 3.1 Component Diagram

****

**Figure 3.1 Component Diagram for Restaurant Management System**

The Component Diagram illustrates the modular structure of the Restaurant Management System, with seven reusable components encapsulated within the system boundary.

The Admin Interface on the user’s device interacts with six management modules (Employee, Menu Item, Order, Table, Billing, Report), which all depend on the centralized System Database for data persistence.

This design promotes separation of concerns, scalability, and easy maintenance by isolating business logic and UI from data storage.

It reflects the CRUD operations and data flow shown in earlier DFD and sequence diagrams.

# Chapter 4. Implementation and Testing

## 4.1 Implementation

### 4.1.1 Tools Used

#### CASE Tools

Software projects like Restaurant Management Systems can benefit from the use of CASE (Computer-Aided Software Engineering) tools for development, upkeep, and documentation. For this project, the following tools are utilized:

* **Integrated Development Environment(IDE):**

1. IntelliJ IDEA: It is a powerful, intelligent Java IDE developed by JetBrains, offering advanced code completion, refactoring, and debugging tools. It supports multiple languages, frameworks (like Spring, Java EE), and seamless integration with version control and build tools.

* **Diagramming Tool:**

1. [Draw.io](http://draw.io) : Draw.io (now diagrams.net) is a free, open-source online diagramming tool for creating flowcharts, UML, ERDs, BPMN, AWS architecture, wireframes, and more, with no login required.

* **Version Control Tool:**

1. Git: Git is used for version control to track changes to our code base over time, collaborate on code development and manage releases and deployment.

#### Programming Languages

* Java: Java is a high-level, object-oriented, platform-independent language that runs on the JVM (Write Once, Run Anywhere). Widely used in enterprise apps, Android, web, and big data, it ensures robustness, security, and scalability. Supports multithreading, garbage collection, and exception handling with a rich standard library.

#### Database Platform

* MySQL: MySQL is an open-source relational database management system (RDBMS) using SQL for data storage and retrieval. Supports ACID transactions, multi-user access, and high performance with indexing, replication, and clustering.

#### Development and Testing Tools

* POSTMAN: A collaborative API development and testing platform used to design, test, document, and monitor RESTful APIs during backend development.
* SWAGGER: A framework for designing, documenting, and testing RESTful APIs using a machine-readable specification, enabling interactive API documentation and client SDK generation.

## 4.2 Testing

### 4.2.1 Test Cases for Unit Testing

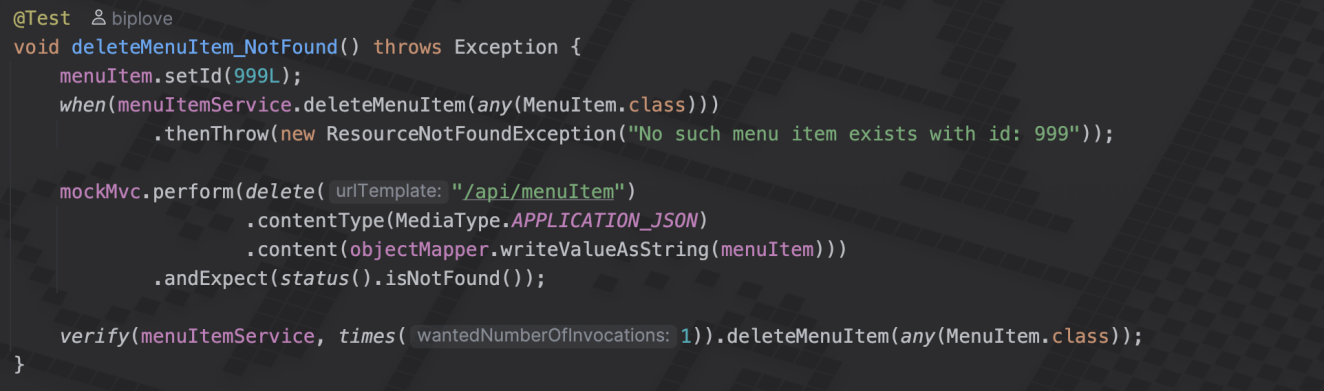
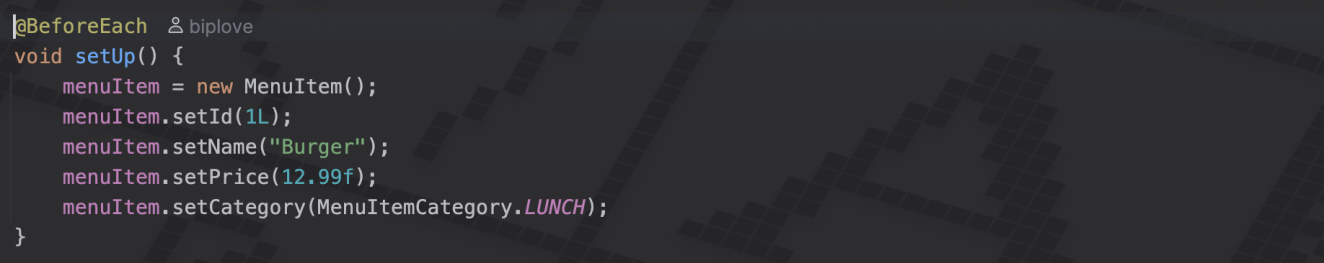
Unit testing was conducted to verify the accuracy and dependability of individual system modules.

It validated component behavior, including the collaborative filtering recommendation algorithm and database operations.

Each module was tested in isolation across multiple scenarios to ensure robustness and precision.

The following images are a few of the many tests that we wrote for our controllers (containing API endpoints) and services (containing Business Logic) in our project. The tests were written for testing the robustness and consistency of the program. The test code follows the following pattern.

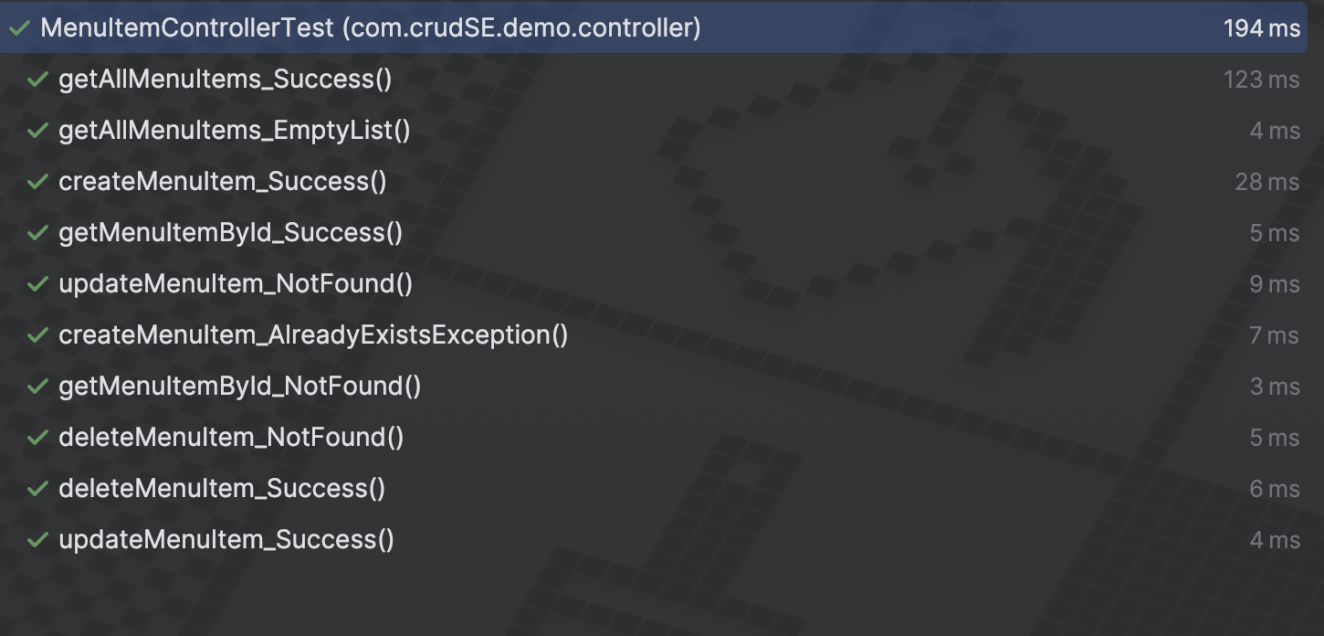
**Testing Done during the code for Controllers:**

****

## 4.3 Result Analysis

The test showed a positive outcome from the tests we conducted for the controllers of our app. All the controllers displayed expected outcomes with given input body or parameters. The program has passed all our unit tests cases.

**Test output:**

****

# 5. Work Done and Conclusion

## 5.1 Work Done

7 implemented features:

1. **Employee Management:** CRUD operations with role assignment
2. **Menu Item Management:** Lifecycle management
3. **Order Management:** Orders with items and transactional integrity
4. **Table Management:** Table operations linked to orders
5. **Role-Based Access Control:** multiple roles per employee
6. **Data Transfer Objects (DTOs):** DTO pattern implementation
7. **Entity Relationships and Cascading:** JPA relationships with cascading

## 5.2 Conclusion

The Restaurant Management System provides a modular backend with clear separation of concerns and OOP-based domain modelling. It supports the core features required by the project brief. Future improvements include adding authentication/authorization (Spring Security), externalizing configuration for secrets, adding integration tests and more comprehensive API documentation (OpenAPI/Swagger).

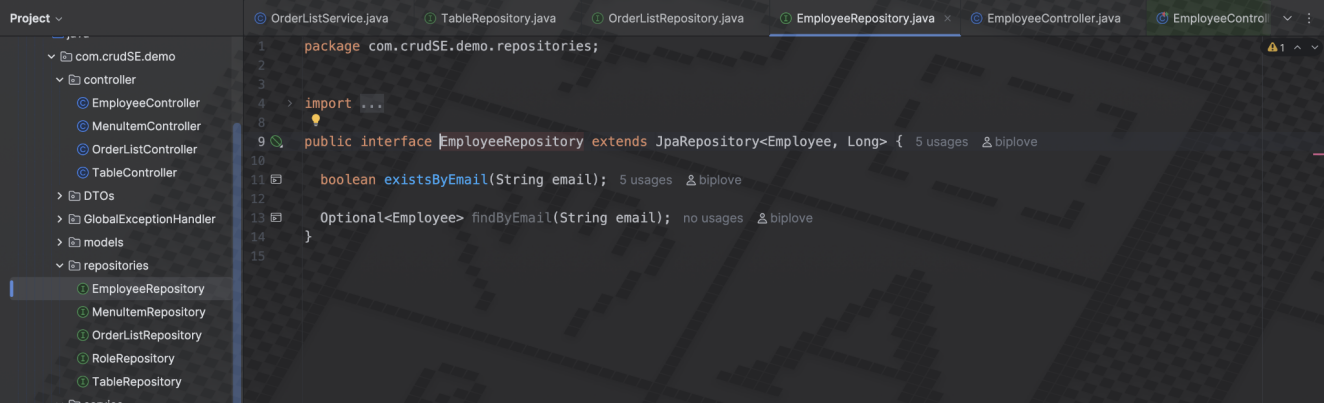
# REFERENCES

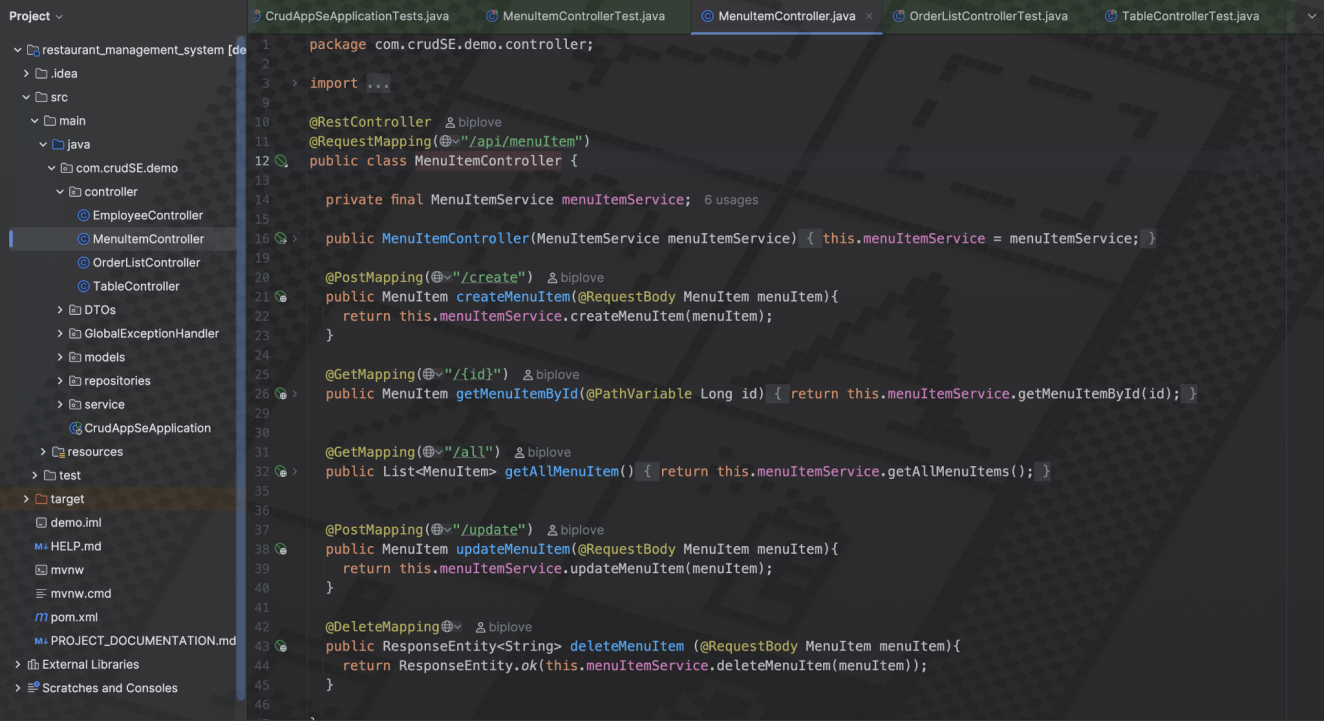
**[1] Karne, Prudveer. "Management System for a Restaurant." (2022). [Online]. Available: [Management System for a Restaurant](https://opus.govst.edu/cgi/viewcontent.cgi?article=1529&context=capstones)**

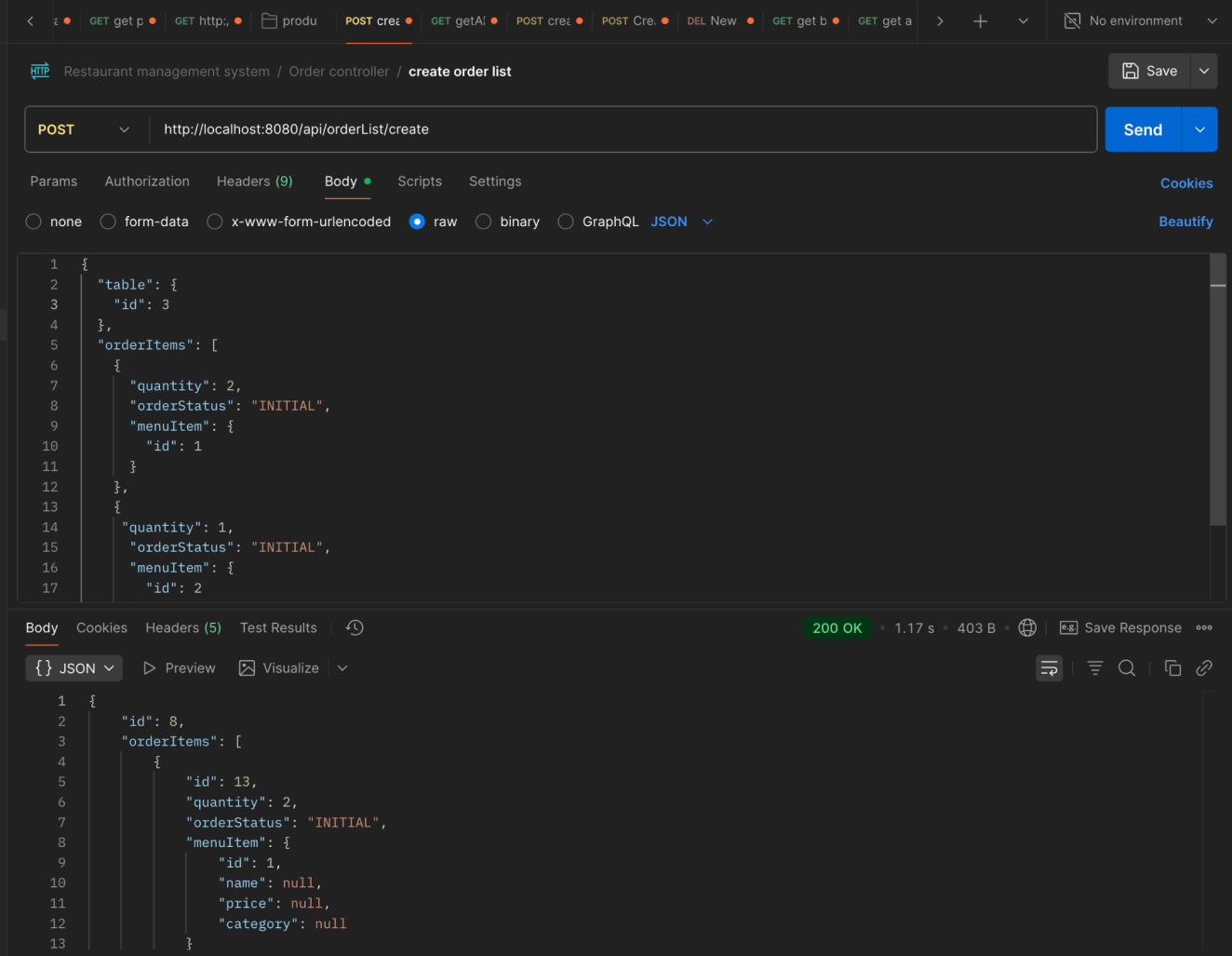
**[2] Macha, Srikar. "Management System for a Restaurant." (2022). [Online]. Available: [Management System for a Restaurant](https://opus.govst.edu/cgi/viewcontent.cgi?article=1531&context=capstones)**

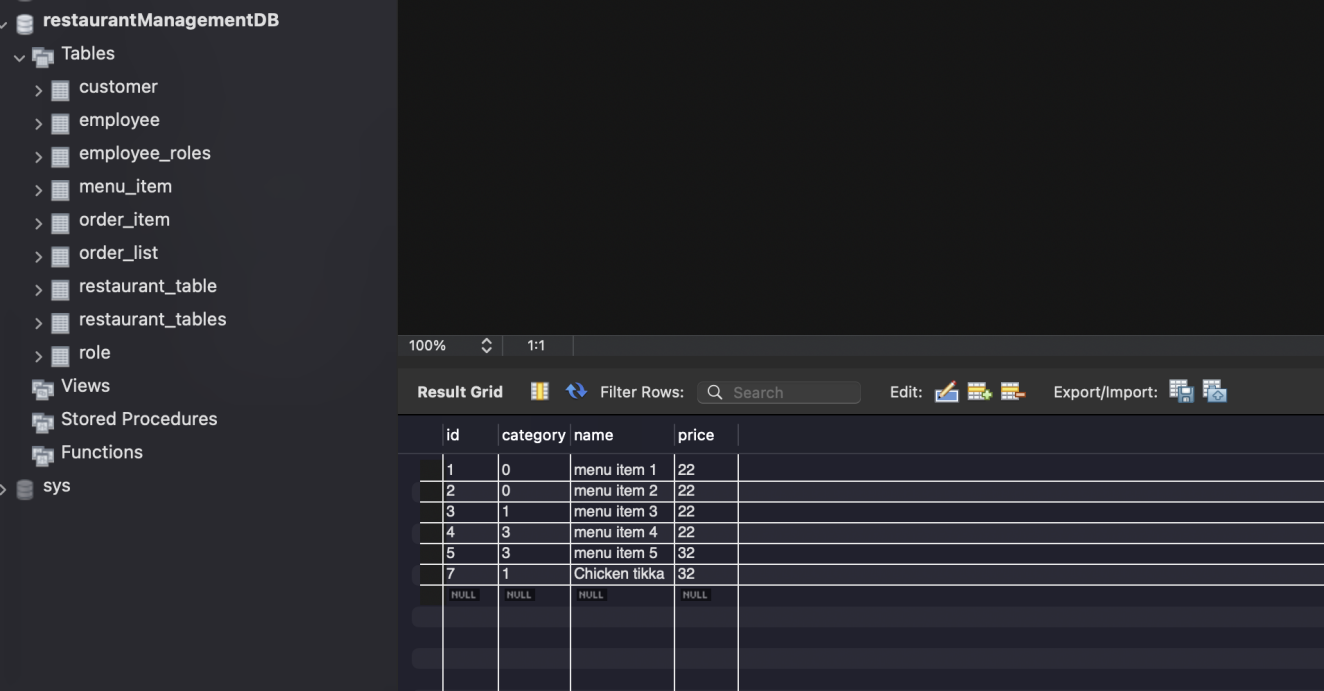
**[3] Chakraborty, Sourav. *An Undergraduate Internship/Project on Restaurant Management System*. Independent University, Bangladesh, 2023. [Online]. Available: [CSE499\_Autumn22\_ArchivableFinalReport\_1821411.pdf](https://ar.iub.edu.bd/bitstream/handle/11348/738/CSE499_Autumn22_ArchivableFinalReport_1821411.pdf?sequence=1&isAllowed=y)**

# Appendices

**Employee repository**

**Menu Item controller**

**Postman testing for Order List**

****

**MySql Database**