Izmir University of Economics

SE 360

Food Ordering System

Delicious Bites

20190601010 - Ömer Beyazkaz

20230601055 - Celal Kurt

1. Introduction

This project is a food ordering system developed using Java. The aim of the project is to practice object-oriented programming, database usage, and client-server communication using socket programming.

The system provides two separate interfaces: a Customer GUI for browsing restaurants, adding items to a basket and placing orders, and a Restaurant Manager GUI for managing menus and viewing incoming orders. Orders are stored in a SQLite database and transmitted to a server application via TCP sockets.

2. What the Project Does

This section explains the main functionalities of the project. The system is designed to simulate a food ordering application with both customer and restaurant management capabilities.

2.1 Implemented Features

Customer Interface:

- Restaurants are stored in a database and loaded when the program starts
- Each restaurant has its own menu with food items (name, type, price)
- Users can add food items to a basket and adjust quantities
- Total price is calculated automatically with wallet balance validation
- Orders can be placed through the GUI and are saved to the database
- Order history can be viewed and previous orders can be reordered
- Auto-refresh timer (15 seconds) synchronizes menu prices from database
- Light/Night theme toggle for user preference

Restaurant Manager Interface:

- Managers can select and switch between restaurants
- Add new food items to the restaurant menu
- Edit existing food items (name, type, price)
- Delete food items from the menu
- View incoming orders for the selected restaurant
- Auto-refresh timer (10 seconds) updates order list in real-time

Client-Server Communication:

- Multi-threaded server handles concurrent client connections
- Structured protocol for order transmission (ORDER, RESTAURANT_ID, ITEMS, END_ORDER)
- Server stores orders in database and sends confirmation response
- Fallback mechanism: orders stored locally if server is unavailable
- Orders from multiple restaurants in basket are split into separate orders

Other Features:

- Database schema migration (automatically adds new columns to existing tables)
- Cross-platform font detection (Mac, Windows, Linux)
- CSV data import for initial restaurant and menu data
- Transaction handling with rollback support for database operations

2.2 Not Implemented Features

The following features are considered out of scope for this project:

- User login and authentication system
- Online payment integration
- Real-time order tracking with status updates
- Mobile or web version of the application

3. Design and Architecture

This project uses a layered architecture with separation of concerns. The system consists of presentation layer (GUIs), business logic layer (model classes), and data access layer (DatabaseManager).

Look at the *.svg* files in the diagrams.*zip* file also.

3.1 Architectural Overview

The system has four main components:

- **Customer GUI (FoodOrderingGUI.java):** Handles customer interaction for browsing and ordering
- **Restaurant GUI (RestaurantGUI.java):** Handles restaurant management operations
- **Order Server (OrderServer.java):** Multi-threaded TCP server receiving orders via sockets

- **Database Manager (DatabaseManager.java):** Handles all database operations and client-server communication

3.2 Client-Server Communication Protocol

The client sends orders to the server using a structured text protocol over TCP sockets:

```
ORDER
RESTAURANT_ID:<id>
TOTAL:<amount>
ITEMS:<count>
ITEM:<name>|<quantity>|<price>
END_ORDER
```

The server responds with "OK:Order stored successfully" or "ERROR:<message>" after processing.

3.3 Database Design

The application uses SQLite database with the following tables:

Table: restaurants

Column	Type	Description
id	INTEGER	Primary key
name	TEXT	Restaurant name

Table: food

Column	Type	Description
id	INTEGER	Primary key
name	TEXT	Food item name
type	TEXT	Food category
price	REAL	Price
restaurant_id	INTEGER	Foreign key to restaurants

Table: orders

Column	Type	Description
id	INTEGER	Primary key
date	TEXT	Order timestamp
total_amount	REAL	Total order price
restaurant_id	INTEGER	Foreign key to restaurants

Table: order_items

Column	Type	Description
id	INTEGER	Primary key
order_id	INTEGER	Foreign key to orders
food_name	TEXT	Name of ordered food
quantity	INTEGER	Ordered quantity
price	REAL	Price per item

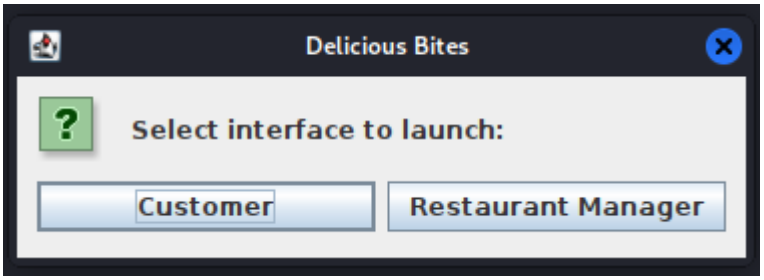
4. Java Technologies Used

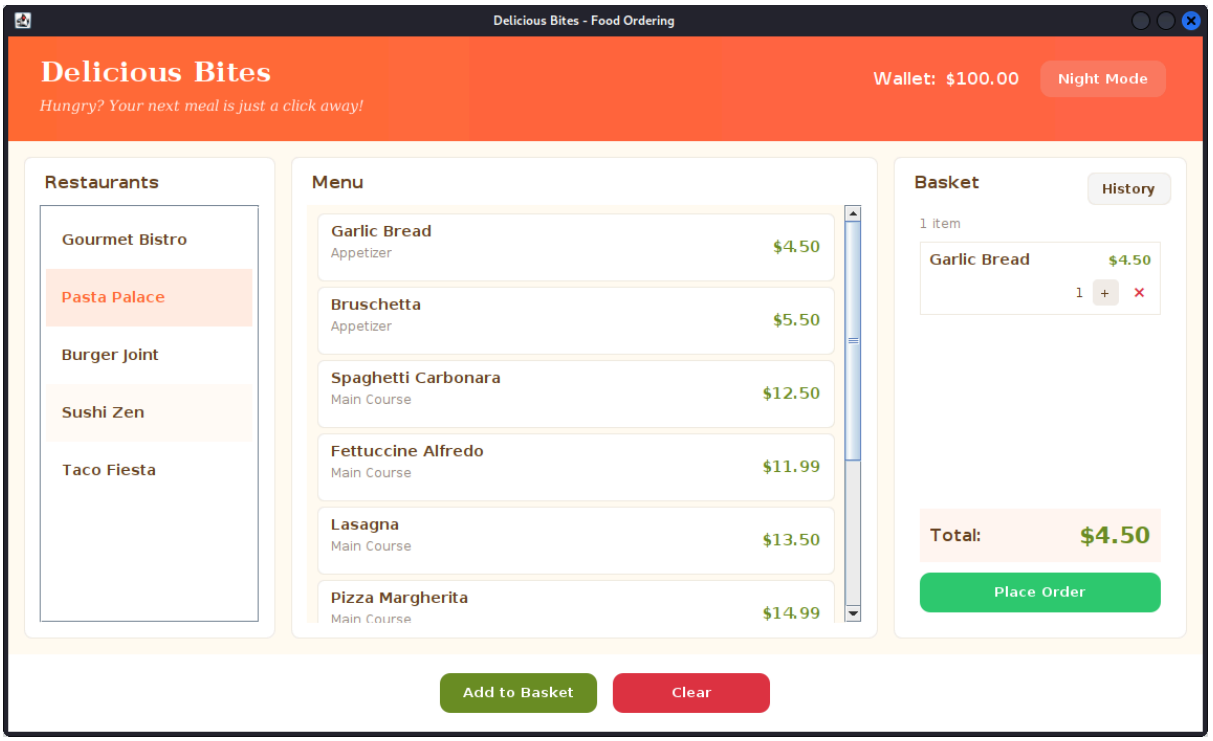
This project is implemented using standard Java technologies:

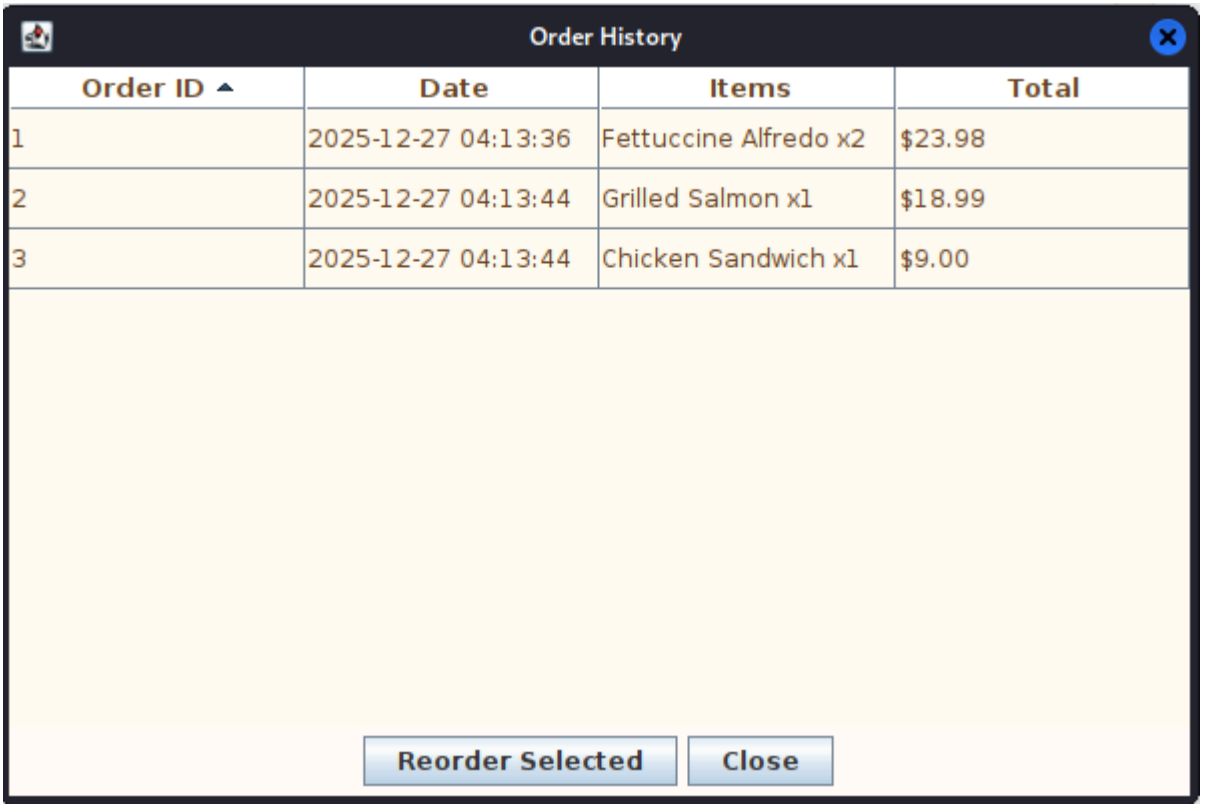
- **Java SE:** Core language for object-oriented programming
- **Java Swing:** GUI framework with custom rendering for themed components
- **SQLite + JDBC:** Embedded database with Java Database Connectivity
- **Java Socket API:** TCP socket communication (Socket, ServerSocket)
- **Java Concurrency:** Multi-threaded server using Thread class
- **Java I/O:** BufferedReader, PrintWriter for socket streams; FileReader for CSV import
- **Java Timer:** javax.swing.Timer for auto-refresh functionality

5. Screenshots

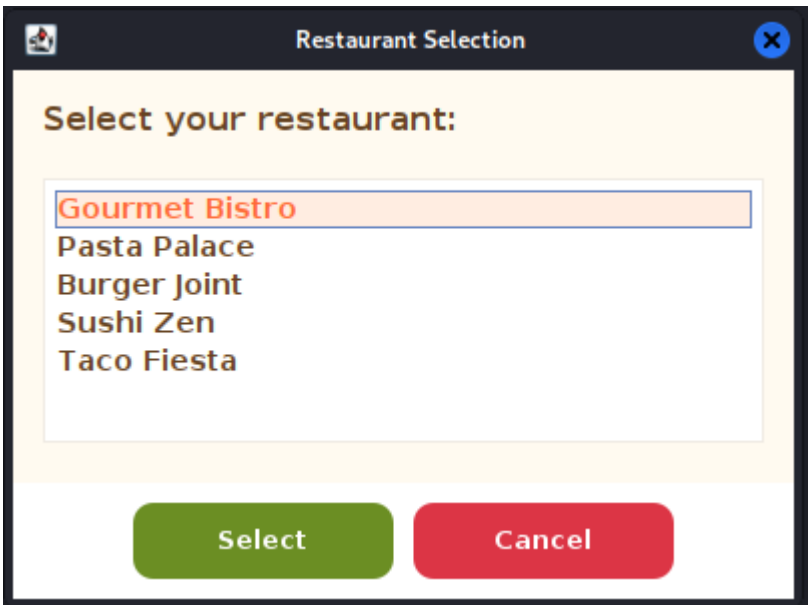
Screenshots of the running application demonstrating the main functionalities should be added here. Include: interface selection dialog, customer GUI (light), restaurant manager GUI, order history, and server console output.

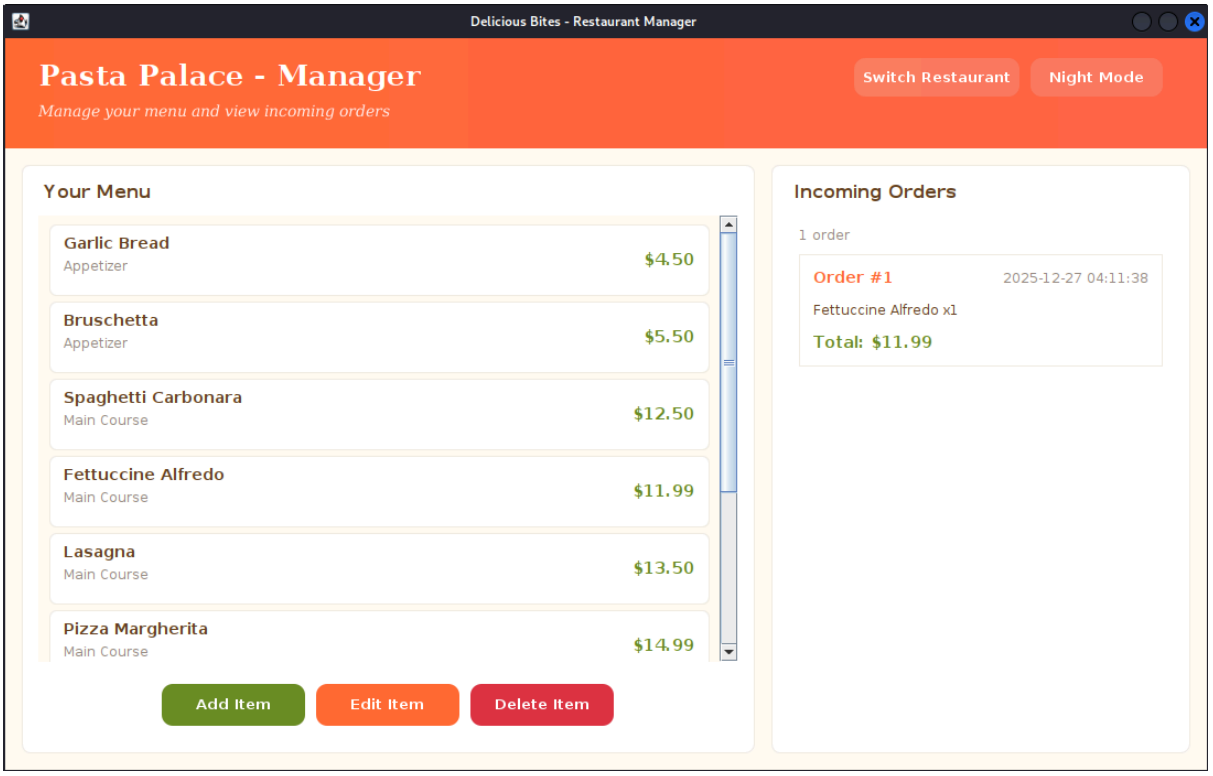





Order History
Order ID
Date
Items
Total
1
2025-12-27 04:13:36
Fettuccine Alfredo x2
$23.98
2
2025-12-27 04:13:44
Grilled Salmon x1
$18.99
3
2025-12-27 04:13:44
Chicken Sandwich x1
$9.00
Reorder Selected
Close


└─$ java -jar OrderServer.jar
Order Server started on port 6000
Database: jdbc:sqlite:/home/kebapci/.food_ordering_app/food_ordering.db
Client connected: /127.0.0.1
New Order Stored:
Restaurant ID: 3
Total: $9.00
Items:
• Chicken Sandwich x1 ($9.00)





6. User Manual

6.1 Starting the Application

1. Run OrderServer class to start the server (optional argument: port number)
2. Run Main class to start the application
3. Select "Customer" or "Restaurant Manager" from the dialog

6.2 Customer Interface

1. Select a restaurant from the left panel
2. Browse menu items and select a food item
3. Click "Add to Basket" to add the item
4. Use +/- buttons to adjust quantities
5. Click "Place Order" to complete (requires sufficient wallet balance)
6. Click "History" to view past orders or reorder
7. Use theme toggle button for Light/Night mode

6.3 Restaurant Manager Interface

8. Select your restaurant from the dialog
9. View your menu on the left panel
10. Click "Add Item" to add new food items
11. Select an item and click "Edit Item" to modify
12. Select an item and click "Delete Item" to remove
13. View incoming orders on the right panel (auto-refreshes every 10 seconds)
14. Click "Switch Restaurant" to manage a different restaurant

7. Tools Used

7.1 Implementation

The project was developed using standard Java development tools and IDE. AI assistance (Claude) was '**highly**' used for code implementation guidance and debugging.

7.2 Report Writing

This report was prepared using AI assistance (Claude) for document generation and formatting.

8. Conclusion

This project successfully implements a Java-based food ordering system with both customer and restaurant management interfaces. The system demonstrates practical application of object-oriented programming, database operations, GUI development, and client-server communication using socket programming.

Key achievements include a multi-threaded server for handling concurrent orders, real-time synchronization between customer and restaurant views, and a robust fallback mechanism for offline operation. The project meets the given requirements and provides a functional demonstration of core Java technologies.