Tribhuvan University

Institute of Science and Technology

A Project Report On

"Food Ordering 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 18 , 2025

Under the supervision of:

Mr. Nabin Man Malla

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. Nabin Man Malla** entitled "**Food Ordering System**" in partial fulfillment of the requirements for the degree of Bachelor of Computer Science and Information Technology to be processed for the final evaluation.

Nabin Man Malla

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 "**Food Ordering 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. Nabin Man Malla, ACHS **External Examiner**

ACKNOWLEDGEMENT

We would like to extend our deepest gratitude to **Mr. Nabin Man Malla** 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 ecommerce_college project is a full-stack E-Commerce Food Ordering System that includes:

- A React-based customer-facing client application (client/),
- A Vite + React-based admin interface (admin/), and
- A Node.js + Express backend API (server/) with a MySQL datastore.

The project's goals, design and architecture, implementation, testing, and deployment guidelines are all covered in this report. For managing products (food), processing carts and orders, and managing users, the backend offers RESTful endpoints. These APIs are used by the client and admin frontends, which offer user interfaces for clients and administrators, respectively.

Keywords: Node.js, Express, React, Vite, REST API, MySQL, JWT (optional), MVC, Full-Stack.

Table of Contents

LIST OF ABBREVIATIONS

API — Application Programming Interface

CRUD — Create, Read, Update, Delete

DB — Database

JSON — JavaScript Object Notation

JWT — JSON Web Token

MVC — Model–View–Controller

REST — Representational State Transfer

LIST OF FIGURES

Chapter 1. Introduction

1.1 Introduction

The ecommerce_college project implements an online food ordering and e-commerce platform for ordering meals and managing products.

The system provides two user interfaces: an admin-facing Vite + React application for managing products, orders, and basic administration, and a customer-facing React client for product browsing, cart management, and order placement. The backend, which runs business logic and communicates with a relational database (MySQL), is a Node.js + Express REST API. The architecture employs a tiered strategy: Routes (Controllers) → Business Logic Services → Data Layer Models.

1.2 Objectives

- Build a RESTful backend to manage products (food items), users, carts, and orders.
- Provide a responsive React client for customers to browse products, add to cart and place orders.
- Provide an admin UI for managing products and orders (CRUD operations).
- Persist data in MySQL and provide clear API contracts for frontends.
- Implement input validation, error handling, and basic security best practices.

1.3 Scope and Limitations

Scope:

- Payment gateway and transactional flows: combine safe webhooks, payment reconciliation, and refund processing with a production payment provider.
- Full CRUD for products (food items), user registration/login (simple), cart operations, and orders.
- Admin panel for product and order management.
- Persistence using MySQL and project-level scripts to seed or configure DB.

Limitations:

- The project uses a simple auth middleware pattern (see `server/middlewares/auth.js`), but production-grade authentication/authorization, role enforcement, and secrets management are not fully implemented.
- Scalability concerns (caching, horizontal scaling) are out of scope for the current implementation.
- The Java Maven module at the repository root appears scaffolded and not integrated with the Node.js backend. The primary backend in active use is the `server/` Node app.

1.4 Development Methodology

The project follows an agile approach:

I. **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.

II. **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.

III. **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).

IV. **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.

V. **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.

VI. **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.

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

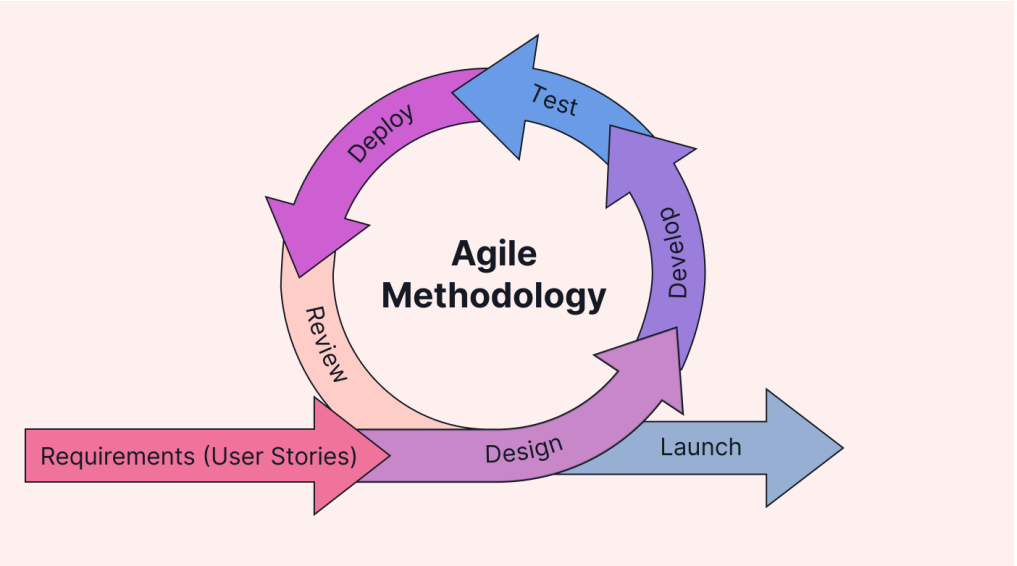


Figure 1.1 Agile Methodology

1.5 Report Organization

The design, implementation, and assessment of the Food Ordering System are presented in this report. It is structured so that the reader is guided from high-level requirements and motivation through design, implementation details, and verification, before concluding with conclusions and next steps.

Chapter 1: Introduction This chapter gives a brief overview of the project, explains the problem and motivation, lays out the goals, and describes the project's constraints and scope. Additionally, it explains the project's Agile development methodology.

Chapter 2: System Analysis In order to make sure the project satisfies the needs of the client and the employees, this chapter focuses on conducting a system analysis. It covers feasibility analysis (including technical, operational, economic, and schedule aspects), requirement analysis (including functional and non-functional requirements, such as performance, security, and maintainability), and detailed analysis using diagrams like ER diagrams, context diagrams, class diagrams, and sequence diagrams.

Chapter 3: System Design An outline of the system's architecture is given in this chapter, along with the Food Ordering System Deployment Diagram.

Chapter 4: Implementation This chapter explains the system's development and the different tools that were used. This includes CASE tools such as the version control system Git, the diagramming tool Draw.io, and the Integrated Development Environments (IDEs) IntelliJ IDEA and VS-Code. Additionally, it identifies MySQL as the database

platform and lists the programming languages that are used, including Java, Javascript, HTML, and CSS.

Chapter 5: Key Features and Implementation The main user-facing features and the underlying technical architecture are described in detail in this section. The persistent shopping cart, the product add feature for users with the "Seller" role, and the payment gateway integration demonstration using a Khalti dummy implementation are examples of Core User Features that are covered. Additionally, it describes the Technical and Security Architecture, which includes Role-Based Access Control (RBAC), JWT Verification for API security, User Authentication, and the utilization of the MySQL relational database for transactional data storage.

Chapter 6: Work Done and Conclusion A full-stack prototype with separate frontends (client/admin) and a REST backend, controllers for products, carts, orders, and users, UI components, and basic middleware for authentication and error handling are all included in this last chapter, which provides an overview of the work completed. It provides the project's conclusion, pointing out that keeping frontends and backends separate makes development easier. It also recommends actions that must be taken for a full production release, including CI/CD pipelines, monitoring, database migration tools, and strong security.

Chapter 2. System Analysis

2.1 System Analysis

Performing a system analysis for a Food Ordering 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

I. Functional Requirement

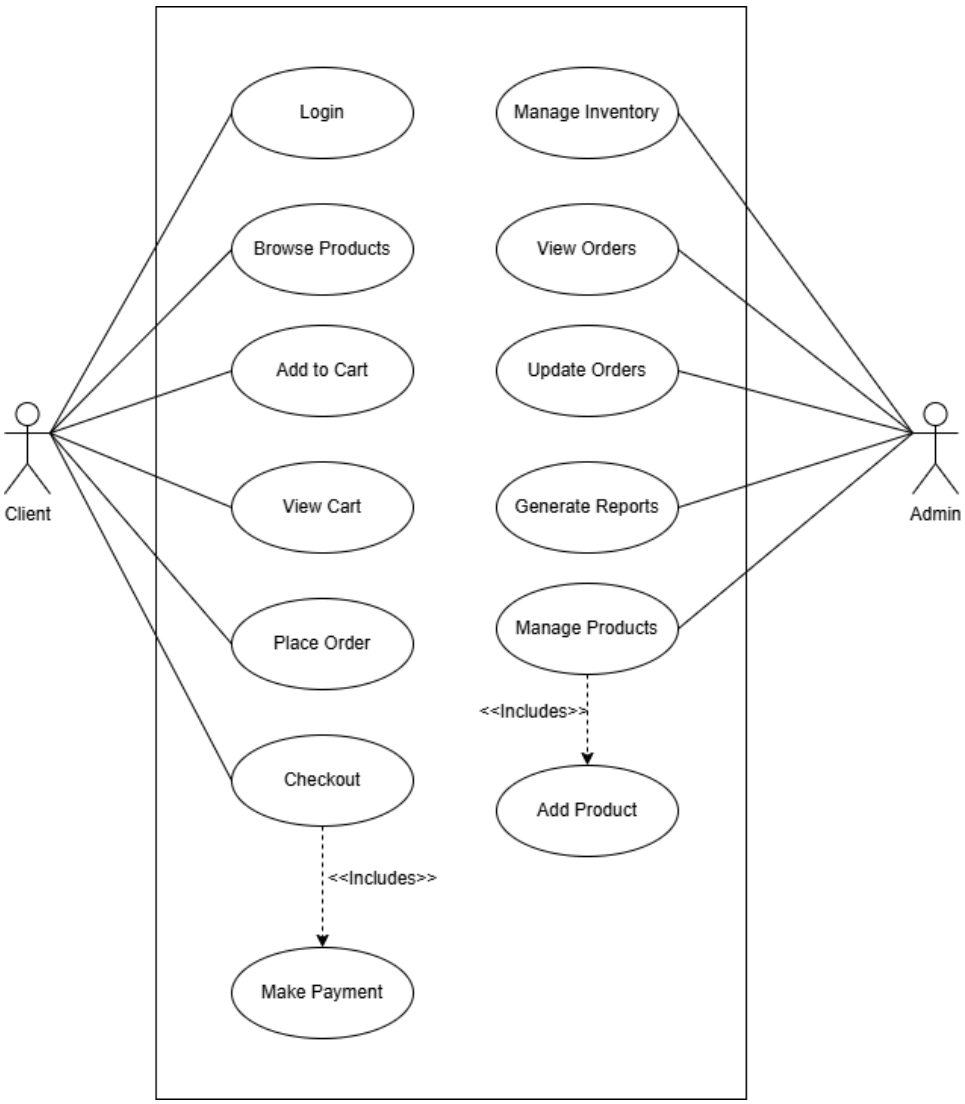


Figure 2.1 Use Case Diagram

The use case diagram captures the functional requirements of the Food Ordering System for two actors: Client and Admin. The Client can login, browse products, add to cart, view cart, checkout, place order, and make payment, with «include» relationships linking checkout to payment. The Admin manages inventory, views and updates orders, generates reports, and manages products. Adding a product is included in managing products, ensuring modular functionality. This diagram defines the system's scope and user interactions at a high level.

II. Non Functional Requirement

- Performance: API endpoints should respond quickly under expected loads; frontends should be responsive.
- Security: Basic input validation and middleware to protect certain routes. Use HTTPS and secret management in production.
- Maintainability: Layered architecture and modular code organization for easier maintenance.

2.1.2 Feasibility Analysis

- Technical : Stack (Node.js + Express, React, MySQL) is mature and fits the requirements. Core features are straightforward to implement; integrations (payments, real-time) add moderate effort.
- Operational: Admins and customers can use separated UIs with minimal training; maintenance is reasonable due to modular code. Production requires monitoring and basic runbooks.
- Economic: Development costs are low (open-source stack); hosting and managed DB are modest recurring costs. Paid services (payments, SMS, monitoring) add variable monthly fees (only if).
- Schedule: A focused MVP can be delivered in ~6–8 weeks for a small team (2–3 students) with iterative sprints. Scope control is essential to meet semester deadlines.

2.1.3 Analysis

I. Data Modeling using ER Diagrams

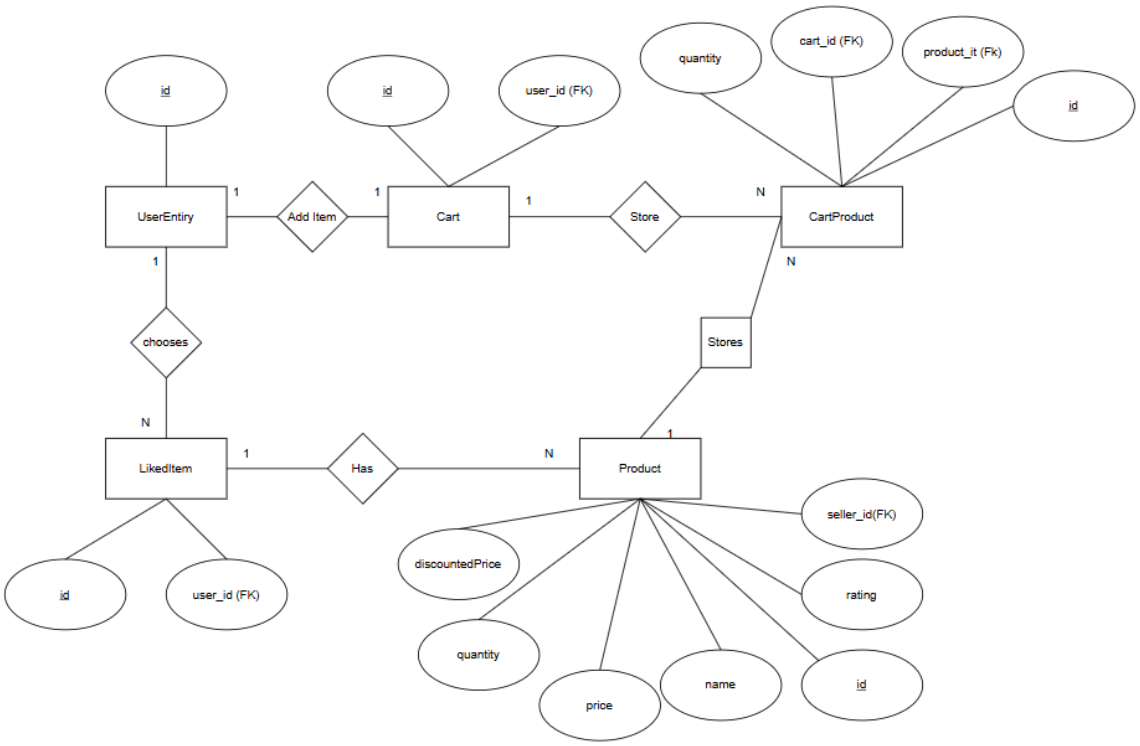


Figure 2.2 Data Modeling using ER Diagram

The ER diagram models the database structure of the Food Ordering System with five core entities. UserEntity owns one Cart and can like multiple LikedItem entries, linked via foreign keys. Cart stores multiple CartProduct items with quantity and cart_id (FK), while Product holds item details like name, price, discountPrice, and rating. CartProduct connects Cart and Product in an N-to-N relationship through product_id (FK). This design supports efficient cart management, product browsing, and user preferences with referential integrity.

II. Context Diagram

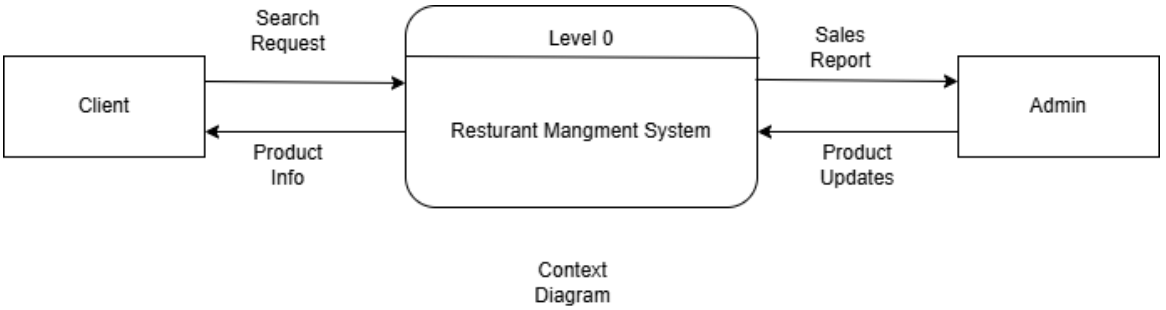


Figure 2.3 Context Diagram for Food Ordering System

The context diagram represents the Food Ordering System as a Level 0 process interacting with two external entities. The Client sends Search Requests and receives Product Info from the system. The system provides Sales Reports to the Admin and

receives Product Updates. This high-level view defines the system boundary and external data flows. It establishes the scope of interactions between users, administrators, and the core restaurant management system.

III. Process Modeling using DFD level 1

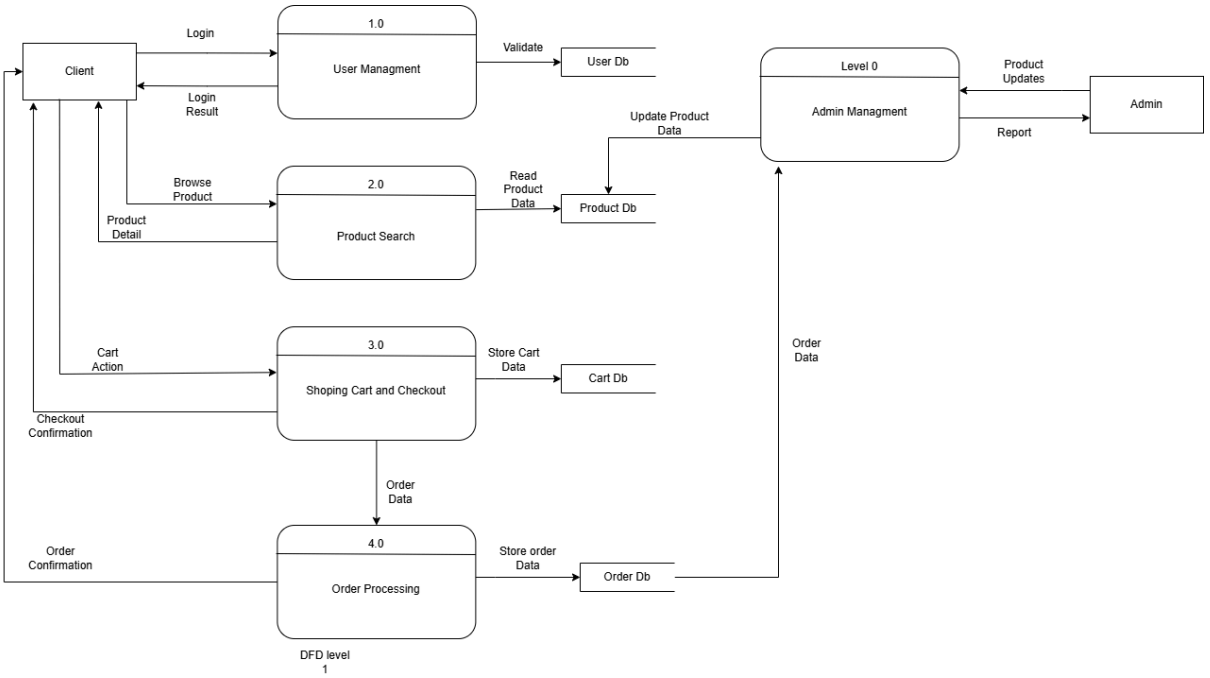


Figure 2.4 Process Modeling using DFD level 1

The DFD level 1 diagram models the core processes of the Food Ordering System across four subsystems.

Process 1.0 handles user login via User Management, validating credentials against User DB.

Process 2.0 enables product browsing and search, reading data from Product DB.

Process 3.0 manages shopping cart and checkout, storing cart details in Cart DB.

Process 4.0 processes orders, storing final data in Order DB, while Level 0 Admin Management updates products and generates reports.

IV. Class Diagram for Entities

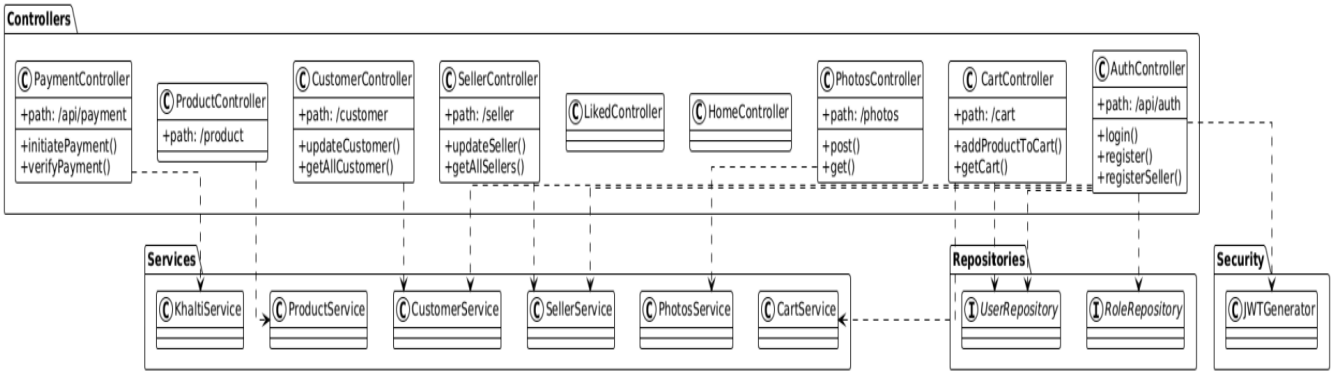


Figure 2.5 Class Diagram for Controllers

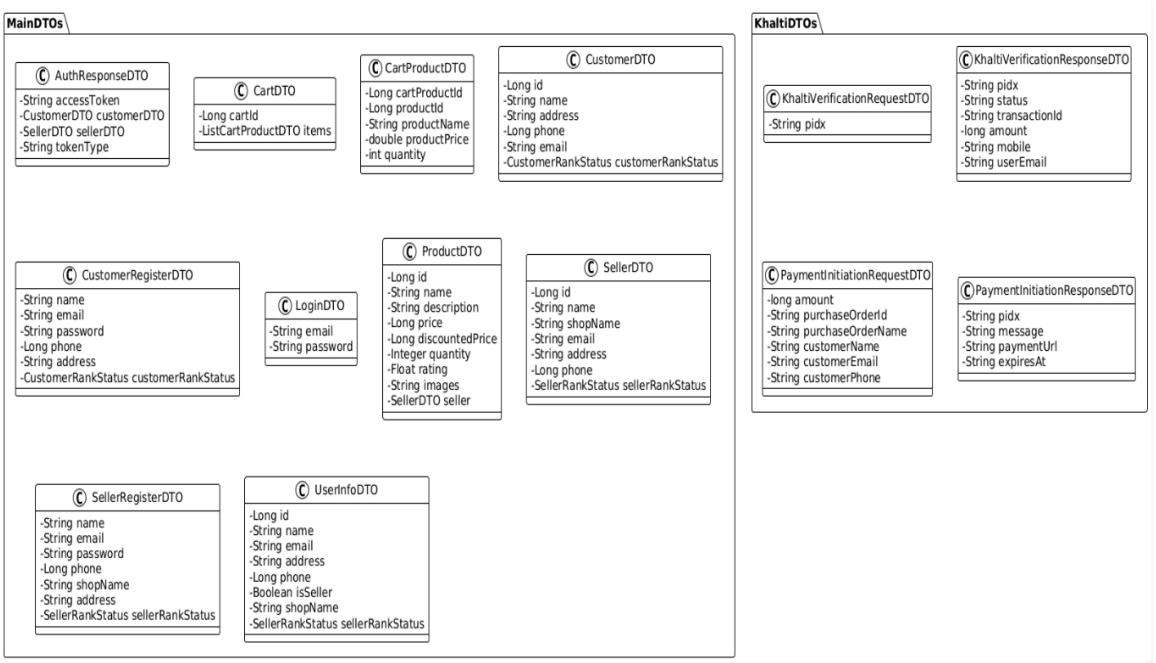


Figure 2.6 Class diagram for DTO (Data Transfer Object)

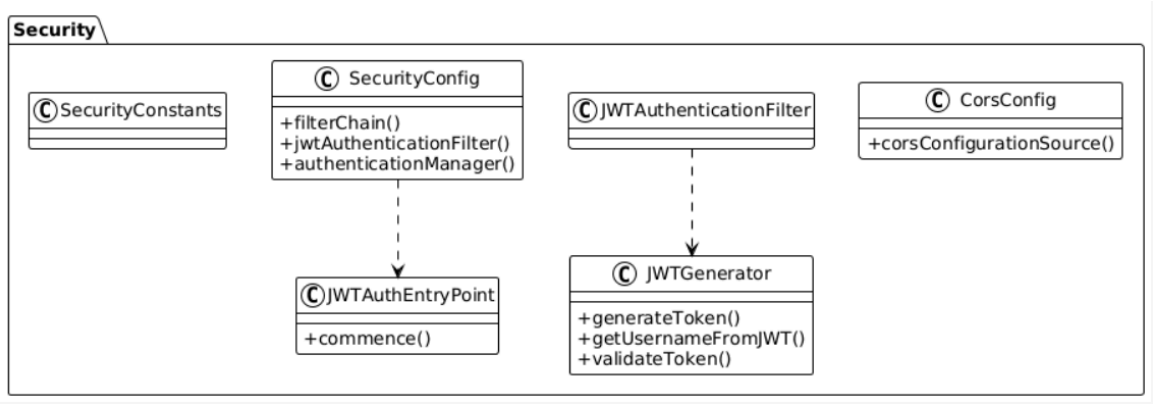


Figure 2.7 Class diagram for Security

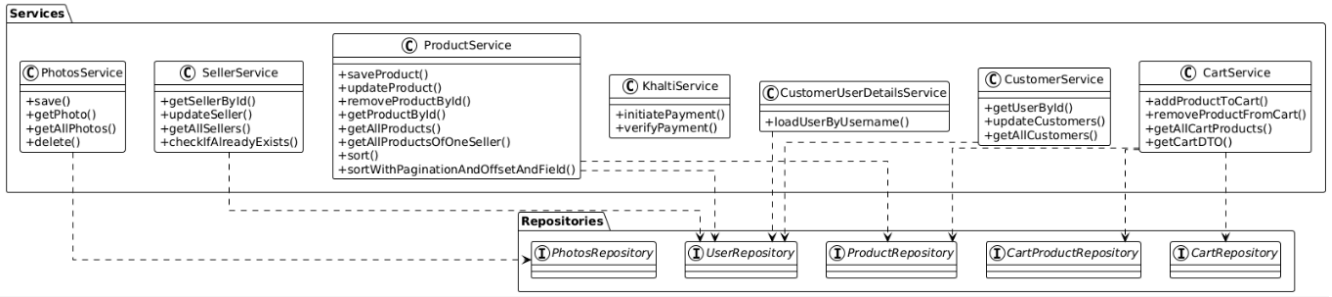


Figure 2.8 Class diagram for Services

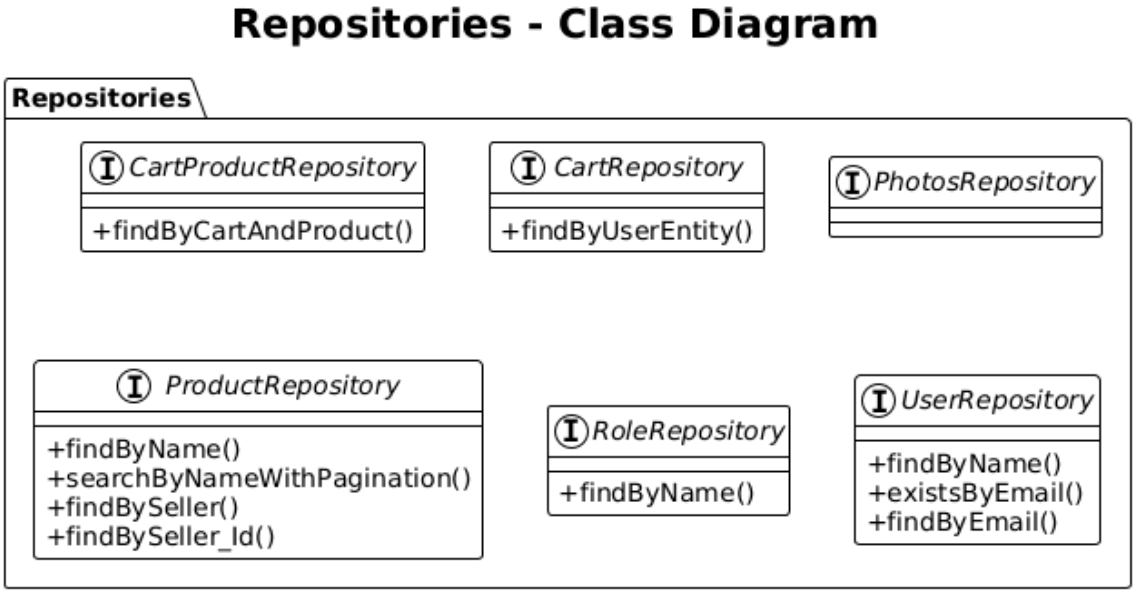


Figure 2.9 Class Diagram for Repositories

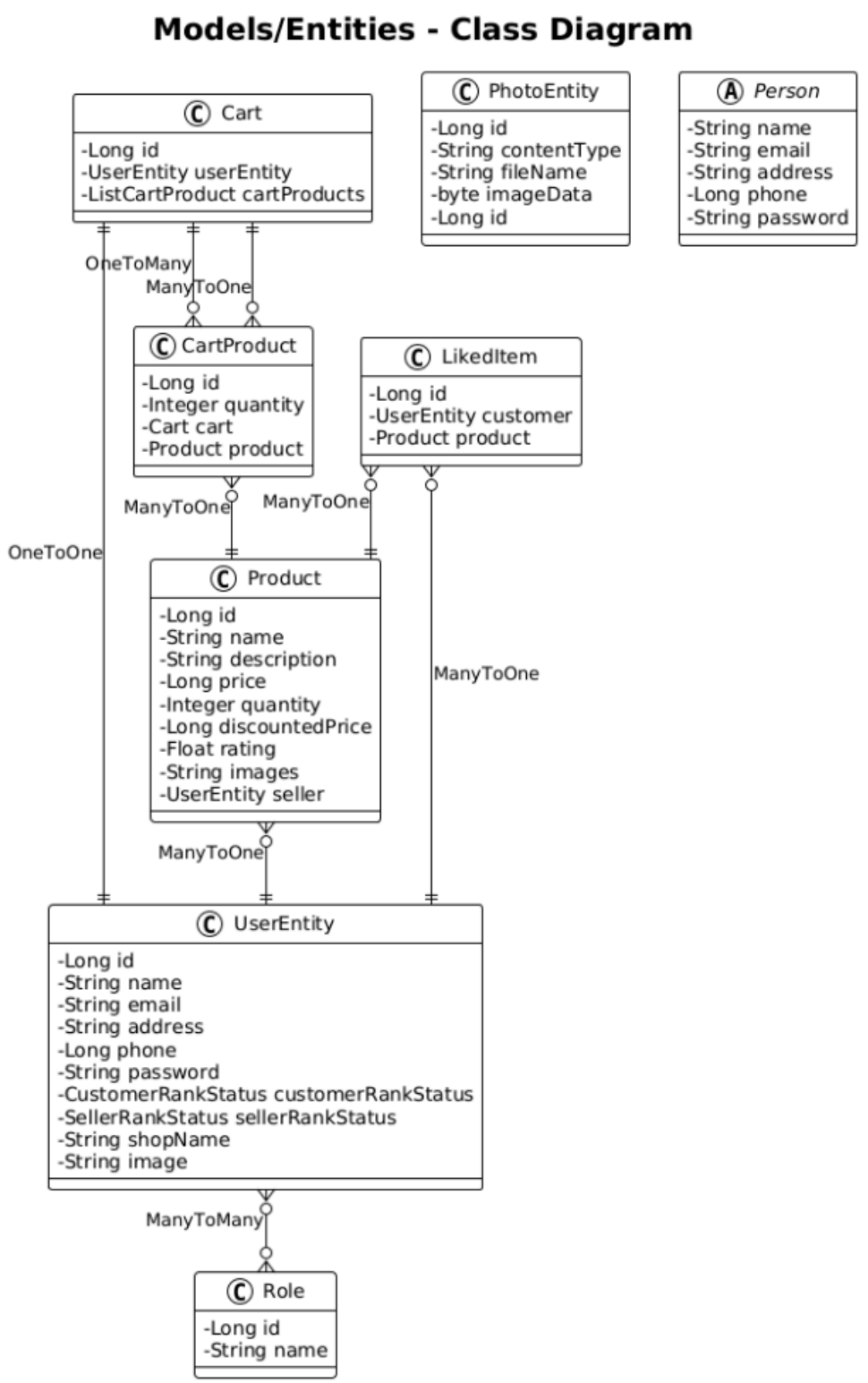


Figure 2.10 Class Diagram for Models

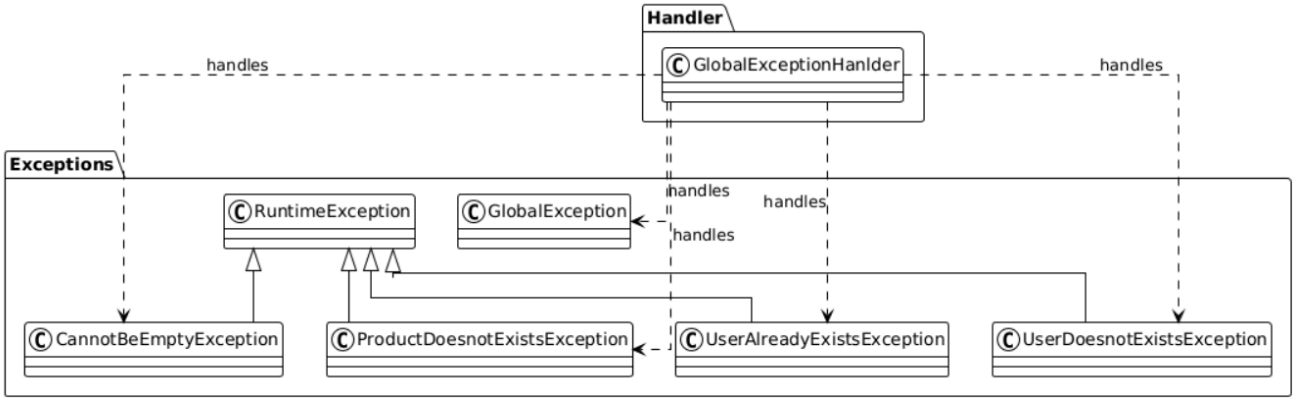


Figure 2.11 Class Diagram for Exception

V. Dynamic Modeling using Sequence Diagram

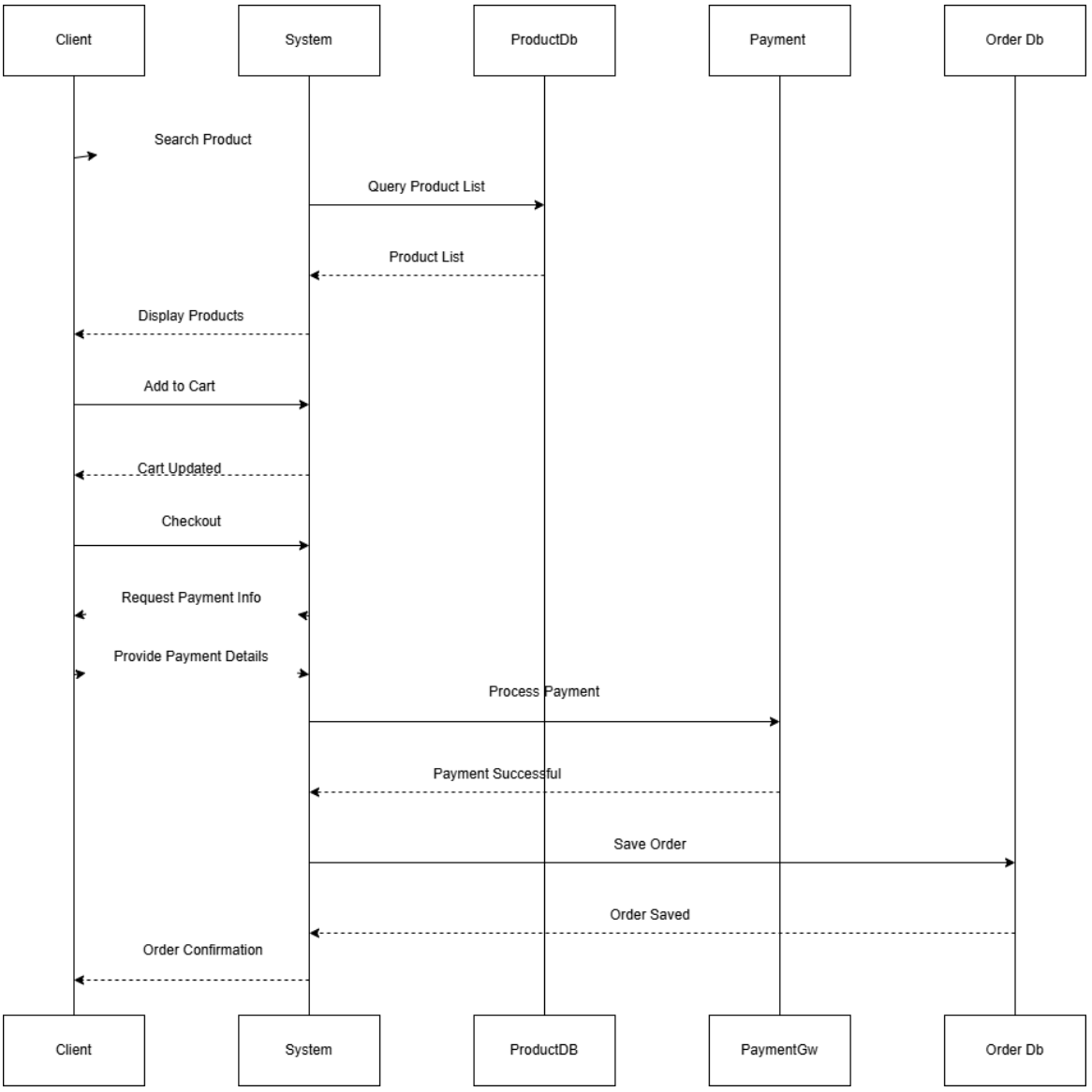


Figure 2.12 Dynamic Modeling using Sequence Diagram

The sequence diagram illustrates the interaction flow for placing an order in the Food Ordering System. The Client searches products, and the System queries the ProductDB to return and display a product list. The Client adds items to cart, receives cart updates, and proceeds to checkout, requesting payment details. The System processes payment via PaymentGW, then saves the order in OrderDB upon success. Finally, the Client receives an order confirmation, ensuring a complete, secure transaction lifecycle.

Chapter 3. System Design

3.1 Deployment Diagram

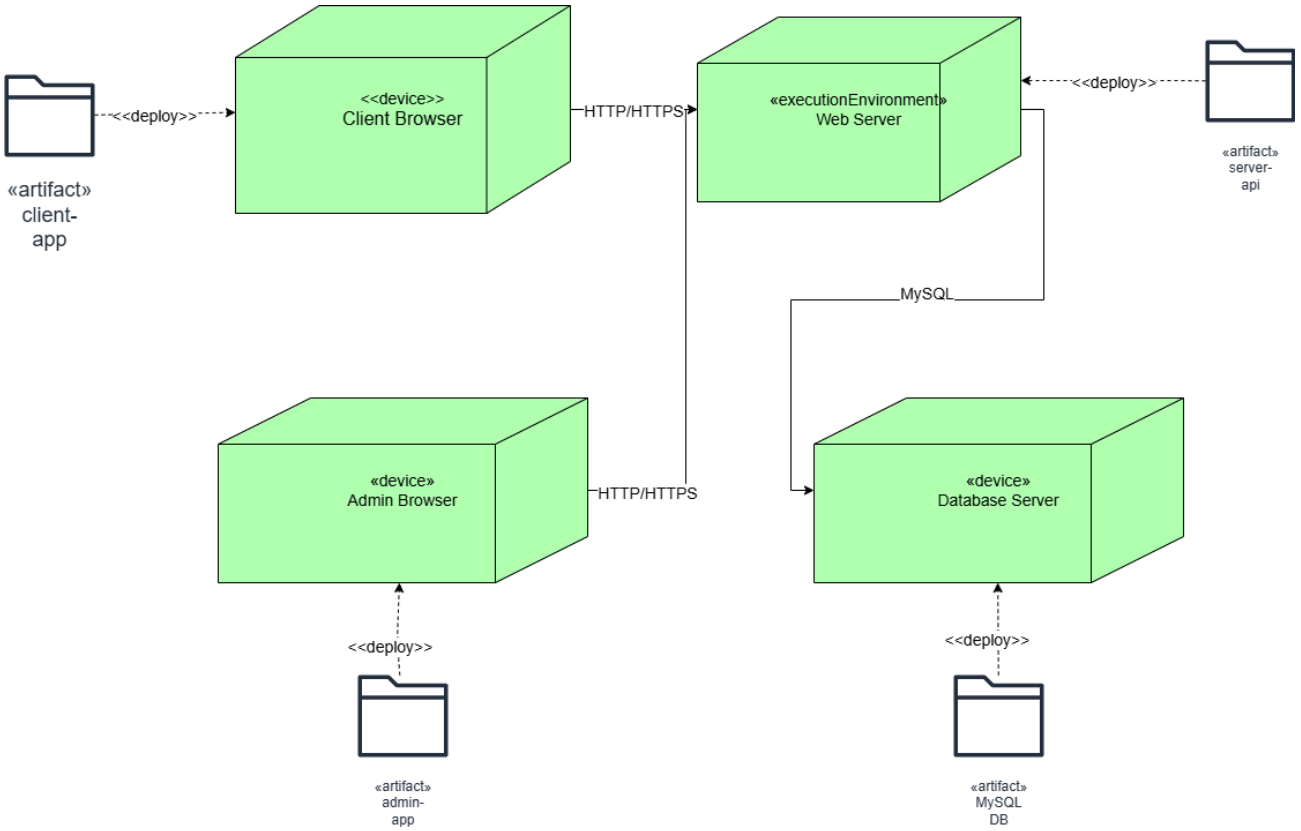


Figure 3.1 Deployment Diagram for Food Ordering System

The deployment diagram shows the Food Ordering System's components distributed across four nodes. The `client-app` and `admin-app` are deployed as artifacts on `«device»` Client and Admin Browsers, accessing the system via HTTP/HTTPS. The `server-api` (Node.js + Express) runs on the `«executionEnvironment»` Web Server. The Web Server connects to the `«device»` Database Server using MySQL (TCP 3306). The `MySQL DB` artifact is deployed on the Database Server to store all persistent data.

Chapter 4. Implementation

4.1 Implementation

4.1.1 Tools Used

I. 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:

A. Integrated Development Environment (IDE):

- **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.
- **VS-Code:** Visual Studio Code is a powerful, lightweight, and free source code editor developed by Microsoft. It's a popular and versatile tool used across many disciplines, from code editing and web or app development to data science and machine learning.

B. Diagramming Tools:

- 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.

C. Version Control Tool:

- **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.

II. 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.

- **Javascript:** JavaScript is a versatile programming language used mainly for web development. It allows developers to make web pages interactive and dynamic. JavaScript runs directly in the browser without needing extra software. It works alongside HTML and CSS to create modern, responsive websites.
- **HTML:** HTML stands for Hyper Text Markup Language, which is the core language used to structure content on the web. It organizes text, images, links, and media using tags and elements that browsers can interpret.
- **CSS:** Cascading Style Sheets is a language used to style and format the presentation of HTML elements on web pages. It allows developers to control the layout, colors, fonts, and spacing of elements, making websites visually appealing and responsive. CSS can be applied in three ways: inline, internal, and external, with external stylesheets being the most efficient for maintaining large projects. It follows a cascade principle, prioritizing styles based on specificity, importance, and source order.

III. 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.

Chapter 5. Key Features and Technical Implementation

This section details the primary user-facing features and the underlying technical architecture that ensures security, scalability, and robust data management for the e-commerce web application.

5.1 Core User Features

- **Shopping Cart:**The application incorporates a fully functional and persistent shopping cart system. This feature allows authenticated users to aggregate multiple products, manage item quantities, and dynamically view the calculated subtotal before initiating the checkout process. The cart state is maintained across sessions, providing a seamless shopping experience.
- **Product Add Feature (Seller Dashboard)**:A dedicated, secure interface is provided for users assigned the 'Seller' role. This feature facilitates the efficient listing of new products into the catalog, enabling sellers to input comprehensive details such as product name, description, pricing, discount and initial inventory levels.
- **Payment Gateway Integration (Khalti Dummy)**:The checkout workflow integrates with a payment service using a Khalti payment implementation. This integration demonstrates the successful handshaking with a third-party payment API, handling secure transmission of payment details, managing transaction callbacks, and updating the order status upon simulated successful or failed payment attempts.

5.2 Technical and Security Architecture

- **User Authentication and Session Management:** Secure user onboarding and access control are foundational elements of the application. The system supports standard user registration and login processes, leveraging strong hashing algorithms to securely store credentials and verify user identity before granting access to personalized services.
- **JWT Verification for API Security:** Following successful authentication, session management is handled exclusively through JSON Web Tokens (JWTs). Each API request to protected endpoints requires a valid JWT, which is verified against a

secret key for integrity and checked for expiration. This approach ensures stateless, token-based authorization, effectively mitigating risks associated with traditional session management.

- **Role-Based Access Control (RBAC):** A robust RBAC model is enforced to strictly define and manage user permissions. This ensures the principle of least privilege, where user access is limited based on their assigned role (e.g., Customer, Seller). RBAC is applied at the API level, restricting unauthorized access to critical functions like product creation (Seller-only).
- **MySQL Database Storage:** The application utilizes a MySQL relational database as the primary data store. This choice provides the necessary reliability, structure, and ACID (Atomicity, Consistency, Isolation, Durability) compliance required for transactional e-commerce data, efficiently managing and querying large volumes of information including user records, product catalogs, and complex order relationships.

Chapter 6. Work Done and Conclusion

6.1 Work Done

- Implemented full-stack prototype with separated frontends (client/admin) and REST backend.
- Built controllers for products, carts, orders and users.
- Provided UI components for browsing products, adding to cart, and placing orders.
- Implemented basic middleware for auth and error handling.

6.2 Conclusion

The Food Ordering project provides a practical full-stack web application for food ordering and e-commerce use cases. The separation between frontends and backend simplifies development and enables independent scaling and deployment. For a complete production release, add robust security (JWT & refresh tokens, RBAC), database migration tooling, CI/CD pipelines, and monitoring.

REFERENCES

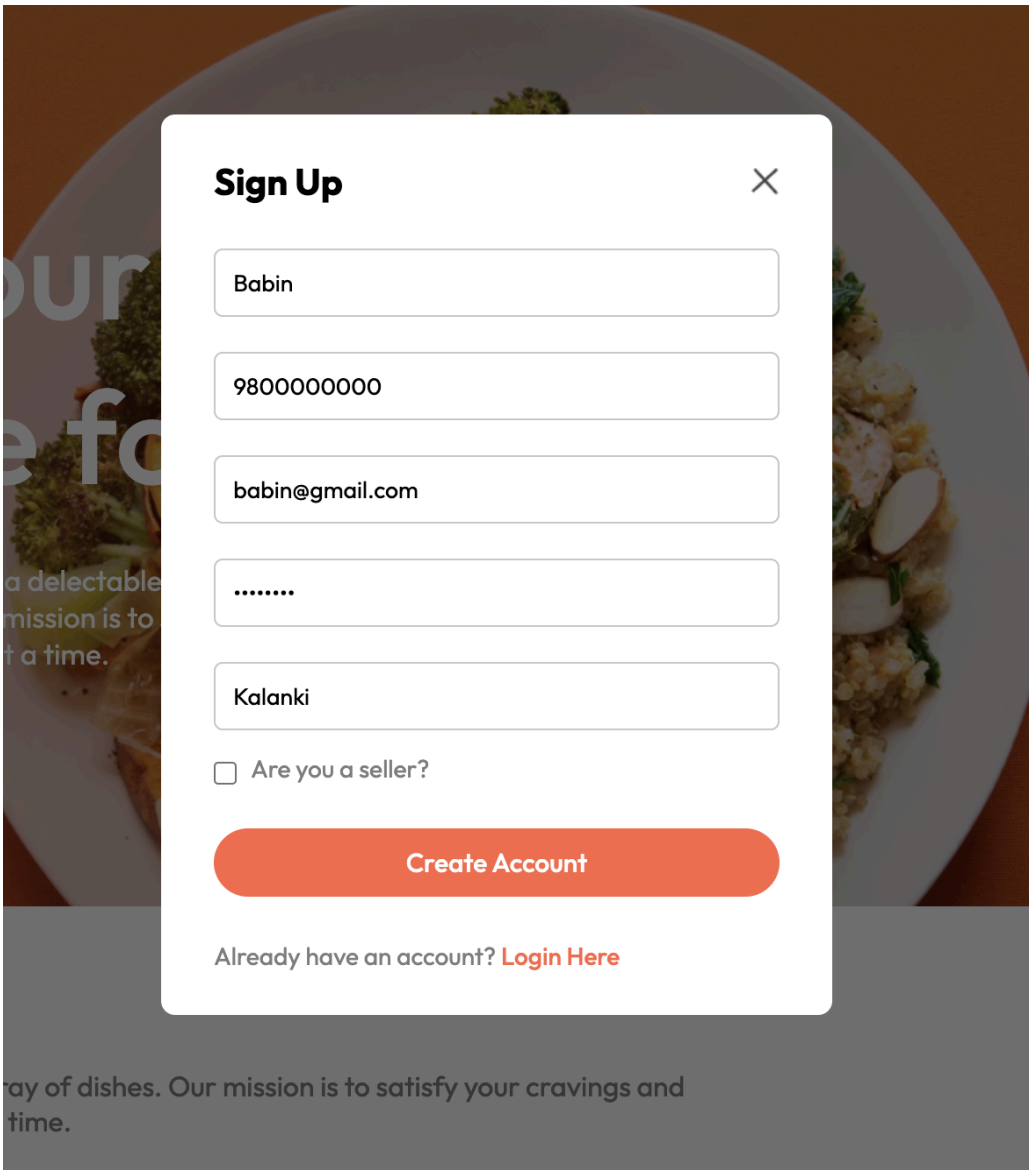
[1] Nurjihan, Saniyyah Wafa, Nafis Faturrahman, Indra Maki Wiguna, Aditya Wicaksono, and Muhammad Nasir. "Design and Implementation of an Online Food Ordering System Using the Agile Scrum." *Journal of Informatics and Communication Technology (JICT)* 7, no. 1 (2025): 47-58. [Online]. Available: View of Design and Implementation of an Online Food Ordering System Using the Agile Scrum

[2] Mahmud, Bayezid. "Easy order app: Making food ordering system easier and simpler." (2024). [Online]. Available: Mahmud_Bayezid.pdf

[3] Yang, Yu. "Design and Implementation of Online Food Ordering System Based on Springcloud." *Information Systems and Economics (2022) Clausius Scientific Press, Canada. DOI* 10 (2022). [Online]. Available: article_1670317352.pdf

APPENDICES

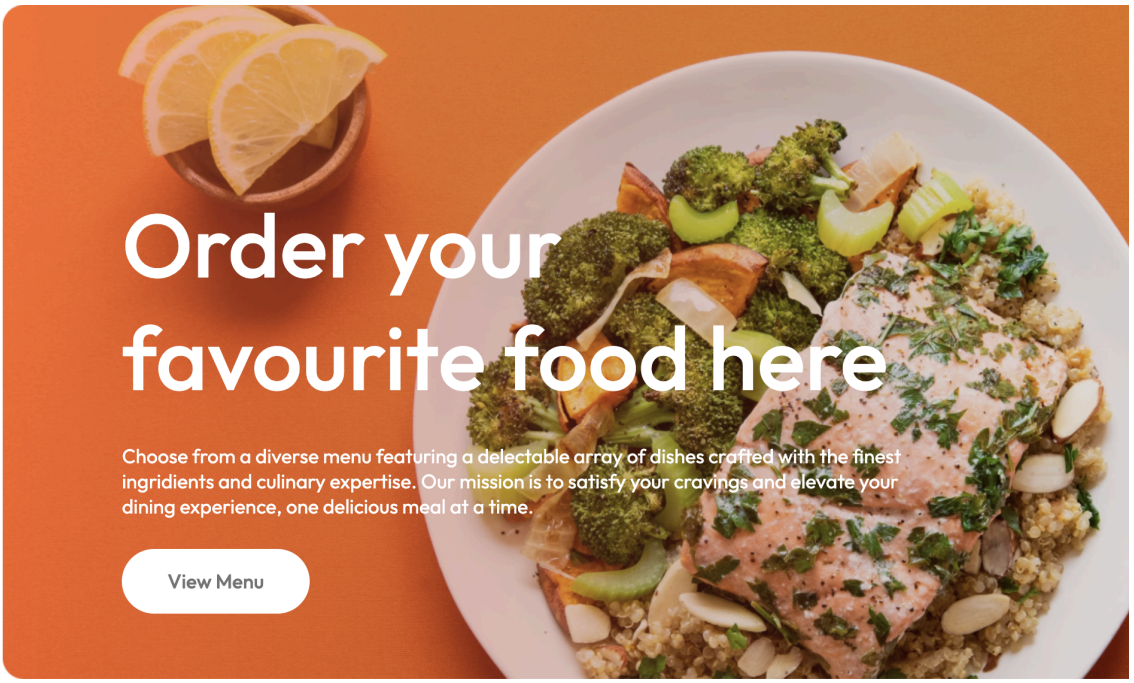
Buyer's experience:



Signup page

Tomato.

Home Contact Us

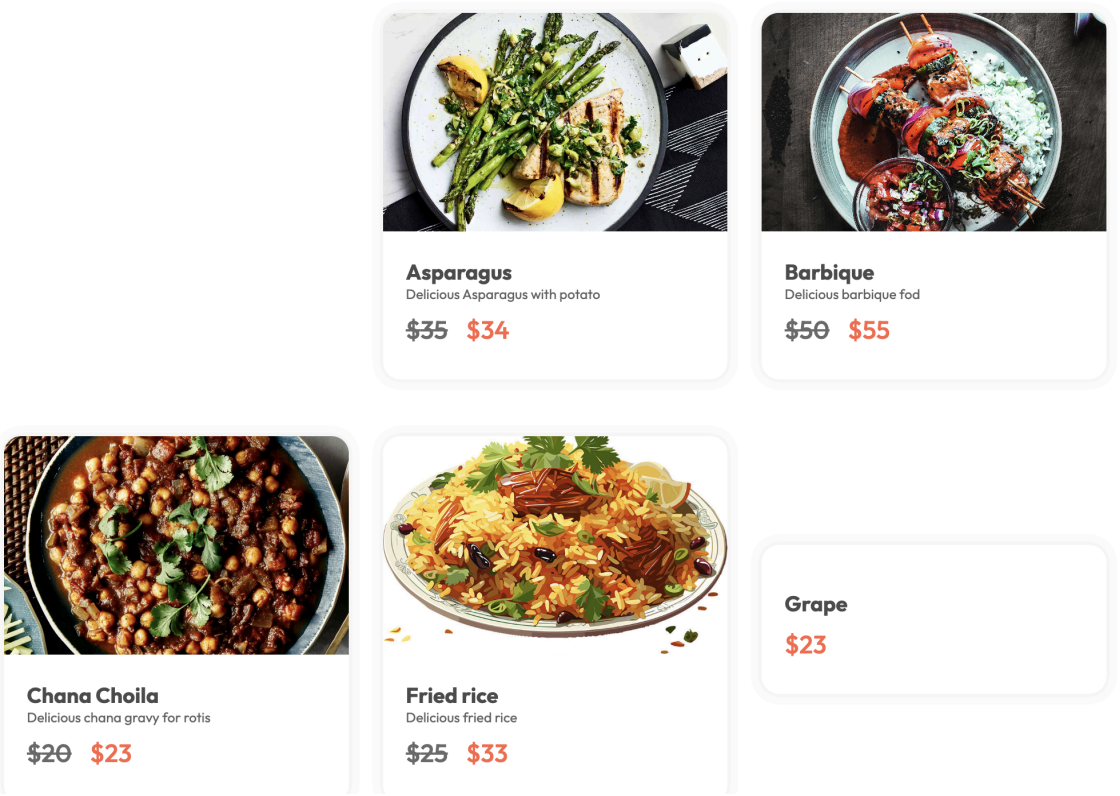


Explore Our Menu

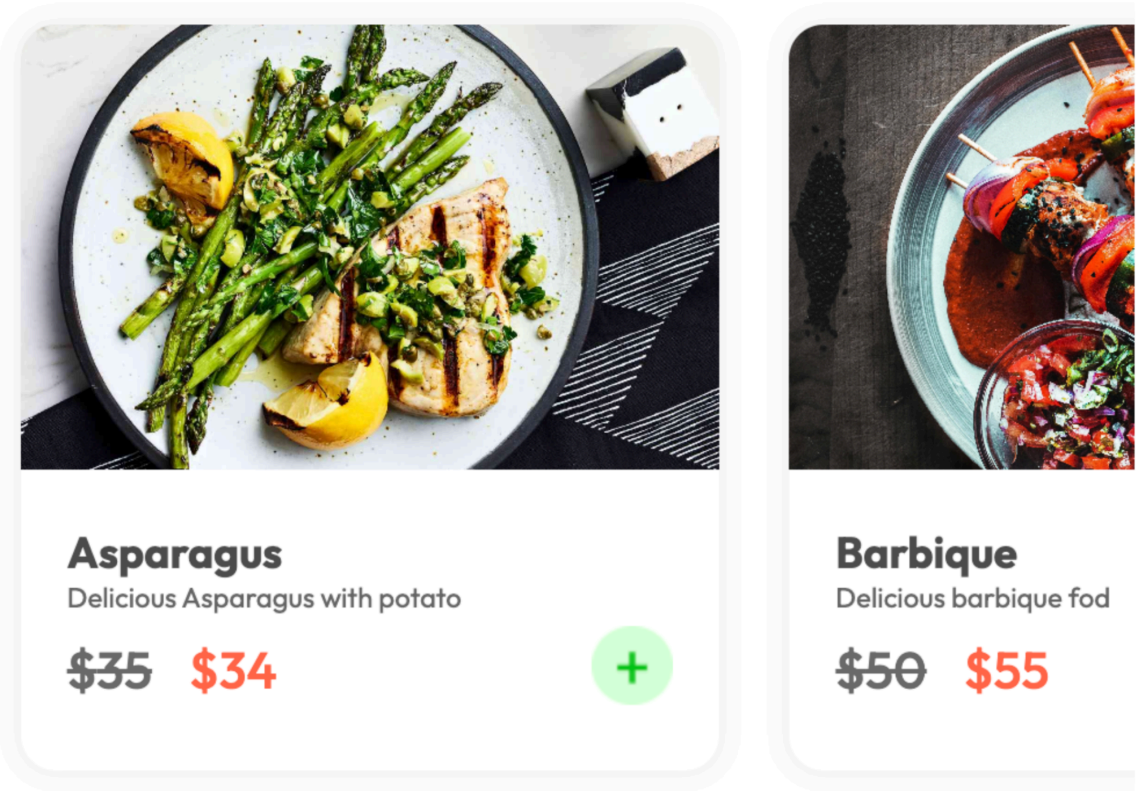
Choose from a diverse menu featuring a delectable array of dishes. Our mission is to satisfy your cravings and elevate your dining experience, one delicious meal at a time.

Top Dishes near you

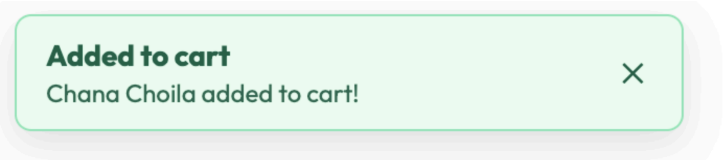
Homepage dashboard



Product list on homepage



Product with Add To Cart button



Successfully added to cart

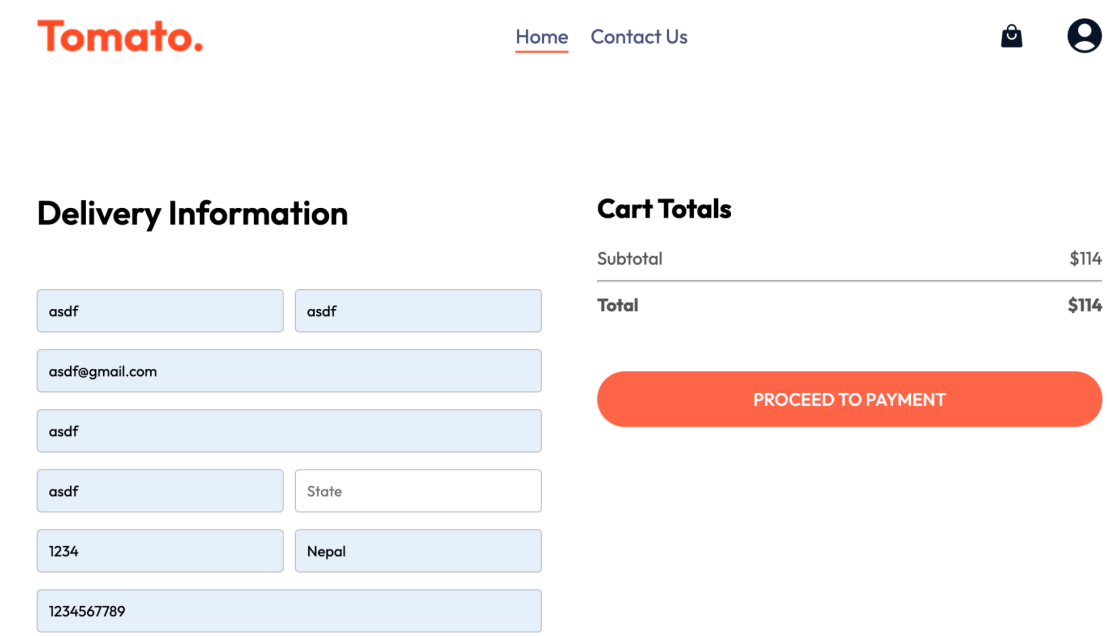
Items	Title	Price	Quantity	Total	Remove
	Asparagus	$34	1	$34	x
	Barbique	$55	1	$55	x
	Chana Choila	$23	1	$23	x

Cart Totals

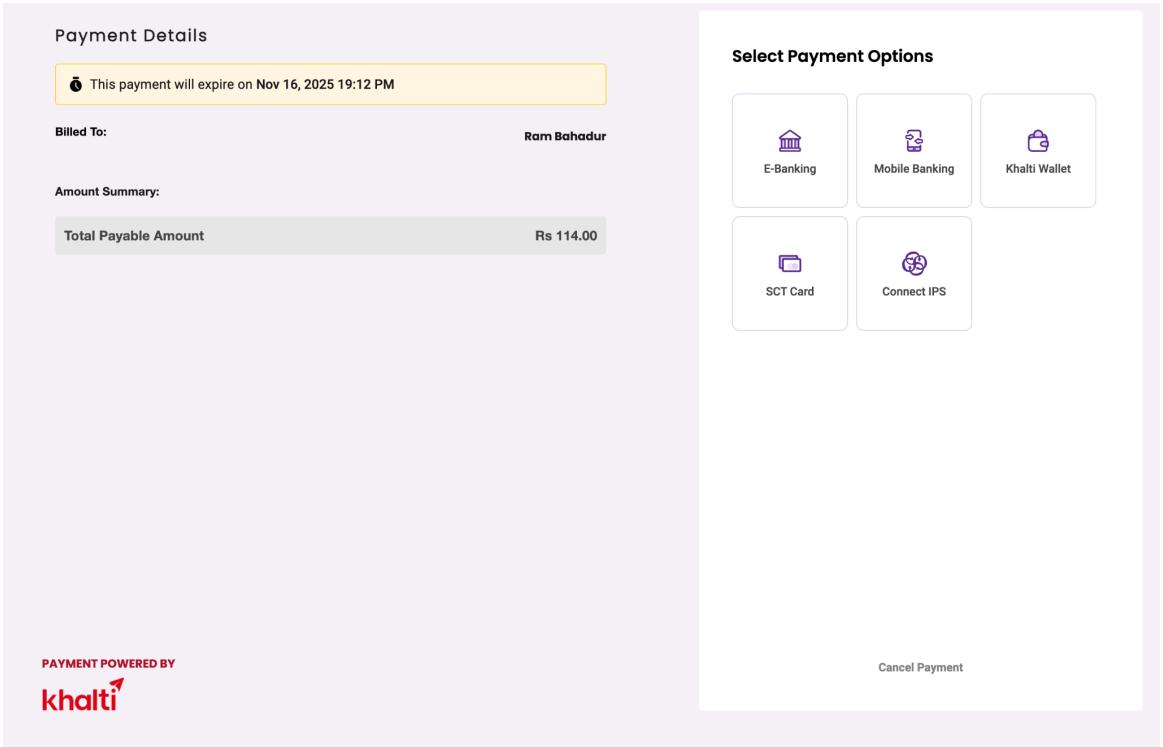
Subtotal	$112
Delivery Fee	$2
Total	**$114**

PROCEED TO CHECKOUT

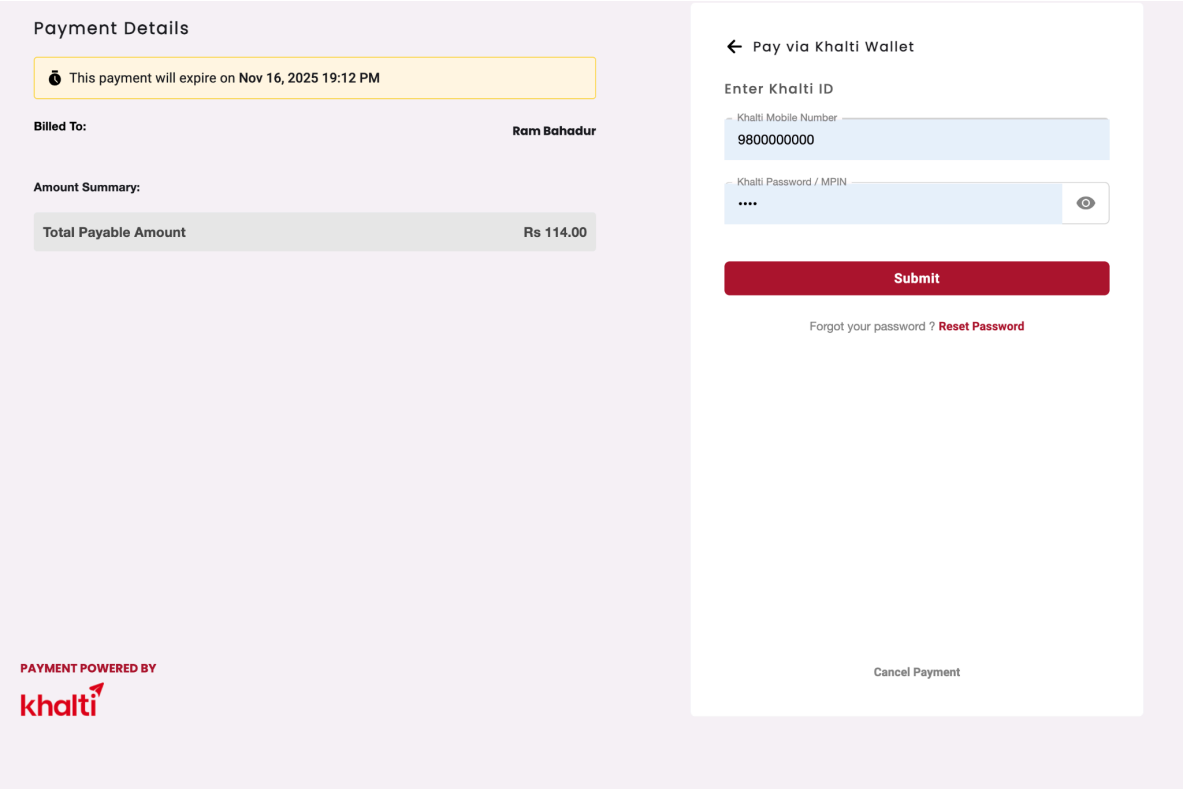
Cart Items



Checkout page

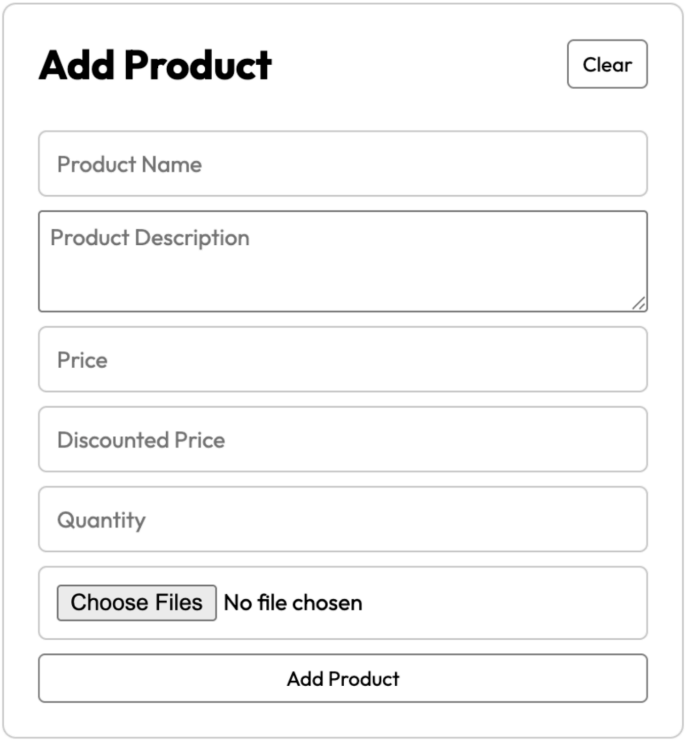


Khalti gateway



Khalti wallet payment process.

Seller's experience:



Add product UI for seller

Items	Title	Description	Price	Discount	Quantity	Edit	Remove
	Asparagus	Delicious Asparagus with potato	34	35	4		x
	Barbique	Delicious barbique fod	55	50	9		x
	Chana Choila	Delicious chana gravy for rotis	23	20	99		x
	Nepali Fish	Tastey fish from Malekhu	12	10	33		x

Added product List from Seller