VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY
INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

WEB APPLICATION DEVELOPMENT

FINAL REPORT

Course by Assoc. Prof. Nguyen Van Sinh

TOPIC:
CAKE STORE WEBSITE

TEAM MEMBERS

Name	ID	Contribution
Ngô Thị Thương	ITCSIU21160	50%
Nguyễn Trần Hoàng Hạ	ITITIU21127	50%

Link Github Code Backend: https://github.com/thuongngo050902/Cake-Store-Website-Project

Link Github Code Frontend: https://github.com/hahoang03/Cake-Store-Website-FE

Link Report:
https://docs.google.com/document/d/1msfZ4V_VLz_uOdb8Kvv8QuBpCGhcN-3h/edit
Link Slide Presentation:
https://www.canva.com/design/DAG8EInS4Qk/650NEzdtwxaCIB_REJtvWw/edit?utm_content=DAG8EInS4Qk&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. Executive Summary

The Cake Store Website project was developed to provide a complete and user-friendly online bakery e-commerce platform. The primary objective of the system is to allow customers to browse bakery products, place orders, and submit product reviews, while enabling administrators to manage users, products, categories, and orders efficiently.

The project focuses on simulating a real-world e-commerce workflow, addressing common challenges such as unclear ordering processes, insecure authentication, and improper role separation. Core functionalities include user registration and login, product browsing, shopping cart management, order placement with payment logic, and a review system with purchase verification.

Throughout the development process, several milestones were successfully achieved, including the implementation of secure authentication using JWT, a structured order placement flow, and a role-based access control mechanism distinguishing Guest, Registered User, and Administrator roles.

The final outcome is a fully functional Cake Store web application that delivers a seamless shopping experience. Users can explore products, manage their orders, and write reviews only for products they have purchased. Administrators can efficiently control system data and moderate content. While the system fulfills its core requirements, future improvements may include integration with real payment gateways, notification services, and delivery tracking systems.

II. Introduction

With the rapid growth of e-commerce, consumers increasingly expect convenient, secure, and efficient online shopping experiences. In the food and bakery sector, customers demand clear product information, smooth ordering processes, and trustworthy review systems. Recognizing these needs, this project aims to develop a Cake Store Website that models a practical online bakery system.

The main goal of this project is to design and implement a web application that supports product browsing, order processing, and user interaction through reviews, while ensuring proper authentication and authorization. The system is designed to be intuitive, scalable, and aligned with common business rules in real-world e-commerce platforms.

In addition to customer-oriented features, the project emphasizes administrative functionality. Administrators are provided with tools to manage users, categories, products, and orders, ensuring data consistency and operational efficiency.

This report presents a comprehensive overview of the Cake Store Website project, including system requirements, design architecture, development process, testing strategy, and deployment considerations. It concludes with reflections on lessons learned and recommendations for future enhancements.

III. Project Planning

- **Back-end Developer:** Ngô Thị Thương
- **Front-end Developer:** Nguyễn Trần Hoàng Hạ

Stage	Date	Tasks
Planning	20/10/2025 - 31/10/2025	Define the topic and technologies.
		Identify key functional requirements.
		Design user interface.
Learning Stage	01/11/2025 - 10/11/2025	Learn defined technologies.
Development Stage	10/11/2025 - 10/12/2025	Develop the web application.
Improvement Stage	10/12/2025 - 15/12/2025	Conduct testing and improve functionalities.
Complete	20/12/2025	Finish developing the web application.

Table 3.1. Project Timeline and Tasks Schedule

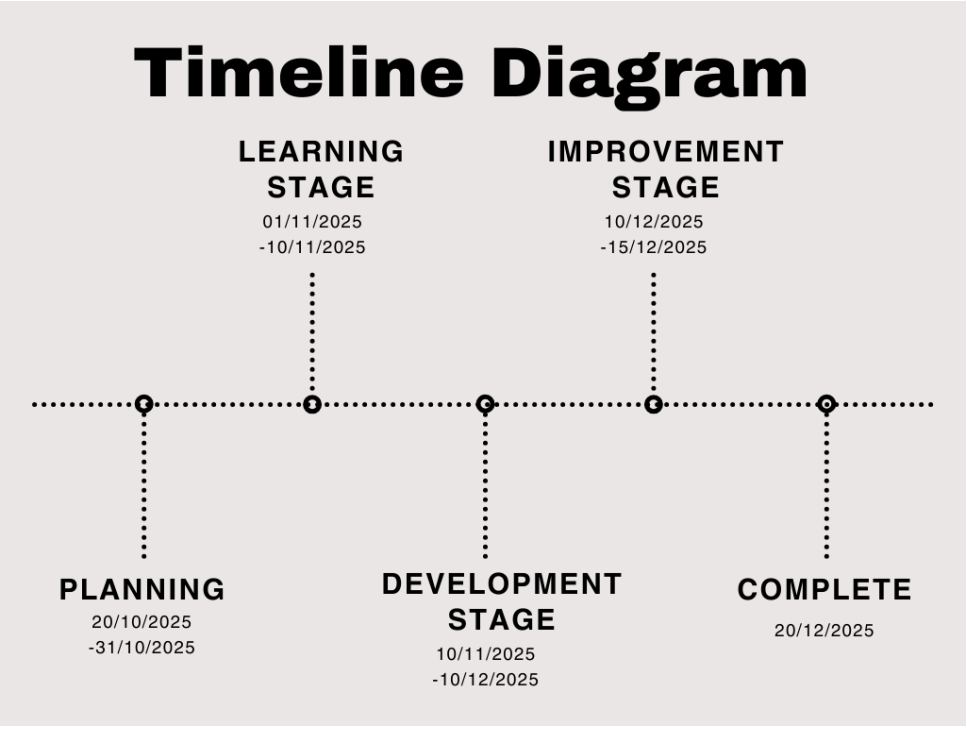

Figure 3.1. Project Timeline

IV. Requirements Analysis

4.1. Project Scope

4.1.1. Description

The Cake Shop H&T is an online bakery e-commerce system that allows users to register, log in, browse products, place orders, and write reviews. Administrators can manage system data through a secured admin interface.

4.1.2. Goals

1. Provide a user-friendly platform for online cake shopping.
2. Ensure secure authentication and role-based access control.
3. Support a complete order lifecycle from cart to delivery.
4. Allow only verified customers to write product reviews.
5. Enable efficient system management for administrators.

4.1.3. Objectives

1. Implement full CRUD operations for products and categories.
2. Ensure accurate order and inventory management.
3. Enforce review authenticity through purchase verification.
4. Design a scalable and maintainable backend architecture.

4.2. Solution Requirement

4.2.1. Functional Requirement

4.2.1.1. Guest (Unauthenticated User):

1. Browse product list.
2. View product details.
3. Search and filter products by category.
4. Register an account.
5. Login to the system.

4.2.1.2. Customer (Registered User):

1. Register and log in securely.
2. Browse and search products.
3. Add products to cart.
4. Place orders.
5. View order history and order details.
6. Write, edit, and delete user's reviews for purchased products only.

4.2.1.3. Administrator:

1. Manage users.
2. Manage categories and products (CRUD).
3. Manage orders and update order status.
4. Delete inappropriate reviews.

4.2.2. Non-Functional Requirement

4.2.2.1 Security

1. Secure authentication using JSON Web Tokens (JWT).
2. Role-Based Access Control (RBAC) for all system roles.
3. Protection of sensitive user and admin data.

4.2.2.2 Performance

1. Fast response time for product browsing and searching.
2. Responsive checkout and order viewing.
3. Stable system behavior under normal load.

4.2.2.3 Availability and Reliability

1. High system availability during usage.
2. Reliable order creation and status updates.
3. Consistent system behavior during admin operations.

4.2.2.4 Data Integrity

1. Orders and order items must remain consistent.
2. Product stock must be updated correctly after order placement.
3. Reviews must always be linked to valid users and products.

4.3. User story

Epic 1: User Authentication

Epic 1: User Authentication	**Priority: Must-have**	**Estimate: 1 week**
User story 1. As a **guest**, I want to register for an account so that I can become a registered user. 2. As a **registered user**, I want to log in so that I can access my account. 3. As a **registered user**, I want to log out so that I can secure my account.		

1. User story 1.1: Guest Registration: **As a guest**, I want to register for an account so that I can have a personalized shopping experience.	
Given	The guest is on the registration page and provides valid information,
When	They click the “Đăng ký” button,
Then	A new user account is created and a success message is displayed.

2. User story 1.2: User Login **As a registered user**, I want to log in so that I can access my profile and place orders.

Given	The user already has a registered account,
When	They enter valid credentials and click the “Đăng nhập” button,
Then	They are authenticated and redirected to the homepage.

3. User story 1.3: User Logout **As a registered user**, I want to log out so that my account remains secure.	
Given	The user is logged in,
When	They click the "Logout" button,
Then	They are logged out and redirected to the homepage.

Epic 2: Product Browsing

Epic 2: Product Browsing	**Priority: Must-have**	**Estimate: 2 week**
User story 1. As a **guest**, I want to view products so that I can explore the store. 2. As a **registered user**, I want to search and filter products to find suitable items.		

1. **User story 2.1:** View Product List **As a guest**, I want to view all products so that I can decide what to buy.	
Given	The user is on the homepage,
When	They access the product list,
Then	All available products are displayed.

2. **User story 2.2:** View Product Details **As a guest**, I want to view product details so that I can see price, description, and reviews.	
Given	The user is viewing the product list,
When	They select a product,
Then	Product details are displayed.

3. **User story 2.3:** Search Products **As a guest**, I want to view product details so that I can see price, description, and

reviews.	
Given	The user is on the product page,
When	They search by keyword or filter by category,
Then	Matching products are displayed.

Epic 3: Shopping Cart & Order Placement

Epic 3: Shopping Cart & Order Placement	**Priority: Must-have**	**Estimate: 2 week**
User story 1. As a registered user, I want to add products to cart. 2. As a registered user, I want to place an order.		

1. **User story 3.1:** Add Product to Cart As a registered user, I want to add products to my cart so that I can purchase them later.	
Given	The user is logged in and viewing a product,

When	They click “Tiến hành đặt hàng”,
Then	The product is added to the shopping cart.

2. **User story 3.2:** : Place Order **As a registered user**, I want to place an order so that I can buy selected products.	
Given	The user has items in the cart,
When	They proceed to checkout and confirm the order,
Then	An order is created with status *Unpaid*.

Epic 4: Payment & Order Processing

Epic 4: Payment & Order Processing	**Priority: Must-have**	**Estimate: 2 week**
User story 1. As a **registered user**, I want to choose a payment method. 2. As an **admin**, I want to confirm payment and delivery.		

1. **User story 4.1:** Select Payment Method **As a registered user**, I want to select a payment method (cash or bank transfer) so that I can complete my order.	
Given	The user is on the checkout page,
When	They select a payment method and confirm,
Then	The order is submitted for admin confirmation.

2. **User story 4.2:** Admin Confirm Payment **As an admin**, I want to confirm payment so that the order can be processed.	
Given	An order has been placed,
When	The admin confirms the payment,
Then	The order status is updated to *Paid*.

3. **User story 4.3:** Admin Confirm Delivery
As an admin, I want to confirm delivery so that the order lifecycle is completed.

Given	An order has been paid,
When	The admin marks it as delivered,
Then	The order status is updated to *Delivered*.

Epic 5: Review Management

Epic 5: Review Management	**Priority: Must-have**	**Estimate: 2 week**
User story 1. As a **registered user**, I want to write reviews for purchased products. 2. As an **admin**, I want to delete inappropriate reviews.		

1. **User story 5.1:** Write Review
As a registered user, I want to write a review so that I can share my feedback.

Given	The user has purchased the product and the order is paid,
When	They submit a rating and comment,
Then	The review is saved and displayed publicly.

2. **User story 5.2:** Admin Delete Review
As an admin, I want to delete inappropriate reviews so that content quality is maintained.

Given	A review violates system rules,
When	The admin deletes it,
Then	The review is removed from the system.

3. **User story 5.3:** Admin Delete Review
As an admin, I want to delete inappropriate reviews so that content quality is maintained.

Given	A review violates system rules,
When	The admin deletes it,
Then	The review is removed from the system.

Epic 6: Administration Management

Epic 6: Review Management	**Priority: Must-have**	**Estimate: 2 week**
User story 1. As an admin, I want to manage users, products, categories, orders, and reviews so that the system operates correctly and business rules are enforced.		

1. **User story 6.1:** Manage Products and Categories As an admin, I want to create, update, and delete products and categories so that the store content remains accurate and consistent.	
Given	The admin is authenticated,
When	They perform create, update, or delete actions on products or categories,
Then	The system saves changes to the database, If a product has no existing orders, it is permanently deleted, If a product has existing orders, it is soft-deleted by setting is_active = false, Inactive products are hidden from customers but remain visible to admins.

2. **User story 6.2:** Manage Users **As an admin**, I want to manage user accounts so that system security is maintained.	
Given	The admin is authenticated,
When	They view or manage users,

Then	User data is updated safely.

2. **User story 6.3:** Manage Orders **As an admin**, I want to manage customer orders so that payments and deliveries are processed correctly.	
Given	The admin is authenticated,
When	They view orders or update order status,
Then	The admin can mark orders as paid, The admin can mark orders as delivered, Order timestamps (paid_at, delivered_at) are recorded accordingly.

2. **User story 6.2:** Manage Reviews **As an admin,** I want to delete inappropriate reviews so that the review system remains trustworthy.	
Given	The admin is authenticated,
When	They delete a review,
Then	The review is removed from the system, Product rating and review count are recalculated, The admin cannot edit review content, only delete.

V. Design and Architecture

5.1 System Architecture Overview

The system architecture of the Cake Store web application is designed to provide a cohesive, scalable, and maintainable environment for managing the core functionalities of an e-commerce platform. Based on the functional and non-functional requirements defined earlier, the system is structured around several interconnected management modules that together support the complete user journey from product browsing to order fulfillment.

At the core of the system is the Product and Category Management module, which is responsible for organizing and maintaining the product catalog. This module allows administrators to create, update, and manage products and categories, while customers can browse and search available products. Products that have been previously ordered cannot be physically deleted and are instead soft-deleted using an is_active flag to preserve historical order data.

The User Account Management module handles user registration, authentication, and profile management. This includes secure sign-up and login processes using JSON Web Tokens (JWT), as well as role-based access control (RBAC) to distinguish between Guest users, Registered Users, and Administrators.

The Order Management module manages the entire order lifecycle, including cart handling, order creation, payment confirmation, and delivery status updates. Unlike real-time payment gateway integration, the system supports a manual payment flow, where users select a payment method (e.g., cash or bank transfer), and administrators confirm payment and delivery statuses.

Additionally, the Review Management module enables customers to submit reviews only for products they have purchased and paid for. This ensures review authenticity while allowing administrators to moderate reviews by deleting inappropriate content.

Overall, this modular architecture ensures clear separation of responsibilities, data integrity, and a streamlined user experience from product discovery to post-purchase interaction.

5.2 Context diagram of system

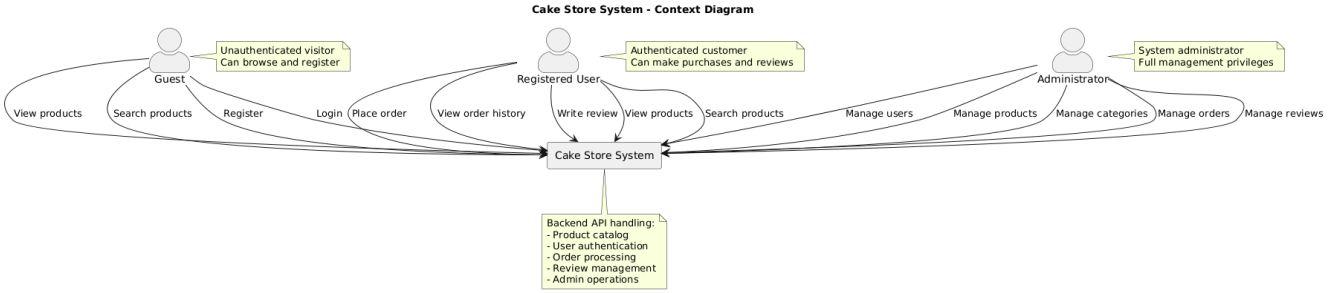

Figure 5.1. Context diagram of system

The context diagram illustrates the interaction between the Cake Store system and external actors, including Guests, Registered Users, and Administrators. It highlights how users interact with the system to browse products, place orders, and manage content, while administrators oversee system operations

5.3 **Use case diagram**

The use case diagram presents a high-level view of the system's functional requirements and actor interactions. For Guests, the system supports browsing and searching products, as well as account registration and login. Registered Users can place orders, view order history, and write reviews for purchased products. Administrators are responsible for managing users, products, categories, orders, and reviews.

Key business rules, such as soft deletion of products and purchase-based review verification, are represented through notes and constraints within the diagram.

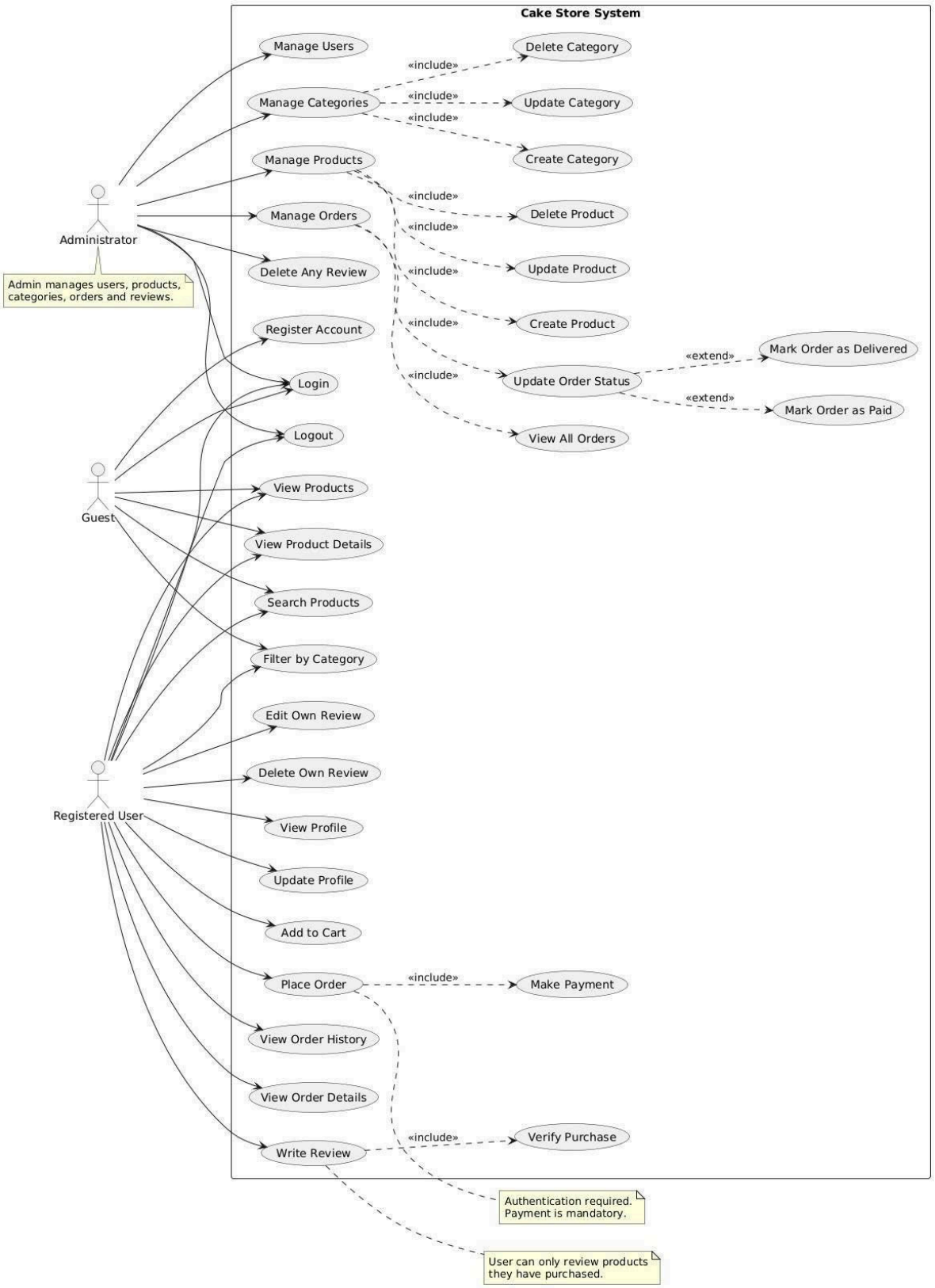

Figure 5.2. Use case diagram

5.4 User flow

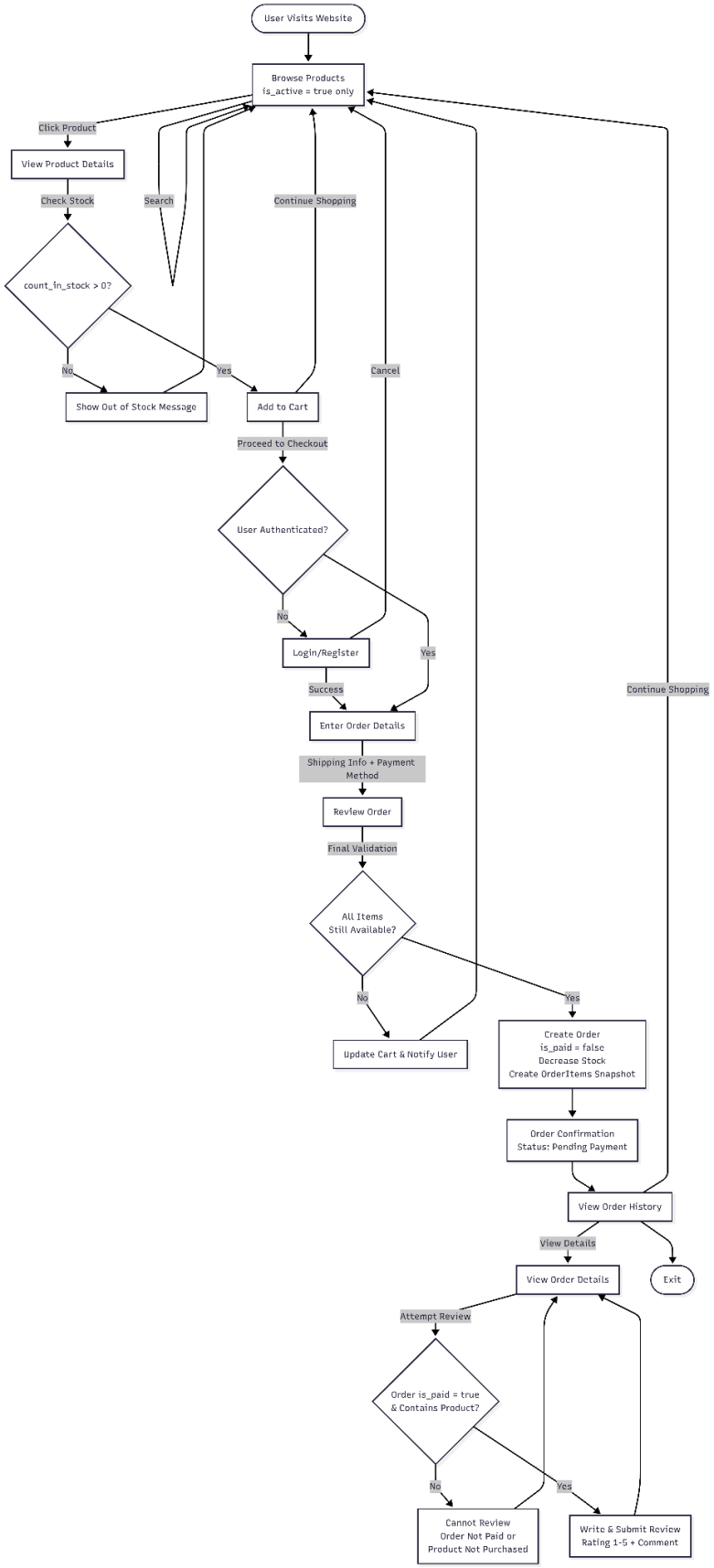

Figure 5.3. User flow diagram

The user flow diagram demonstrates the step-by-step interaction of a customer with the Cake Store website. The flow begins with browsing or searching for products, followed by adding items to the cart, placing an order, selecting a payment method, and finally viewing order history. This diagram helps ensure that the navigation and interaction paths are intuitive and consistent.

5.5 Admin flow

To support administrative operations, the Cake Store system defines three dedicated admin flows: **Admin Overview**, **Order Processing**, and **Product Management**. These flows ensure that all management actions are clearly structured, secure, and aligned with business rules.

5.5.1 Admin Overview Flow

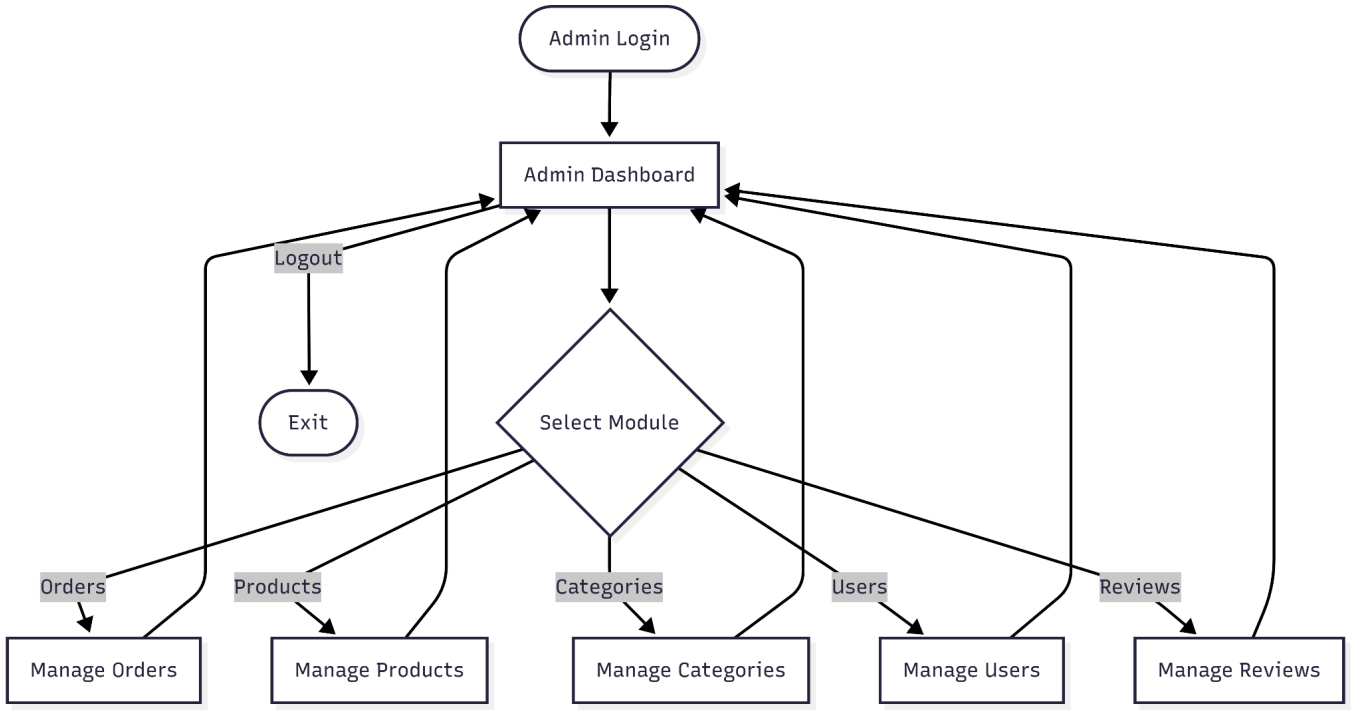

Figure 5.4 Admin Overview flow

The Admin Overview flow provides a high-level view of how an administrator interacts with the system.

After successful login, the admin is directed to the Admin Dashboard, which acts as a central control panel. From this dashboard, the administrator can select one of the main management modules: Orders, Products, Categories, Users, or Reviews.

Each module can be accessed independently and, after completing any operation, the admin can return to the dashboard. The flow also supports a clear logout action, ensuring secure session termination.

This overview flow emphasizes modularity and simplifies navigation for administrative users.

5.5.2 Admin Order Processing Flow

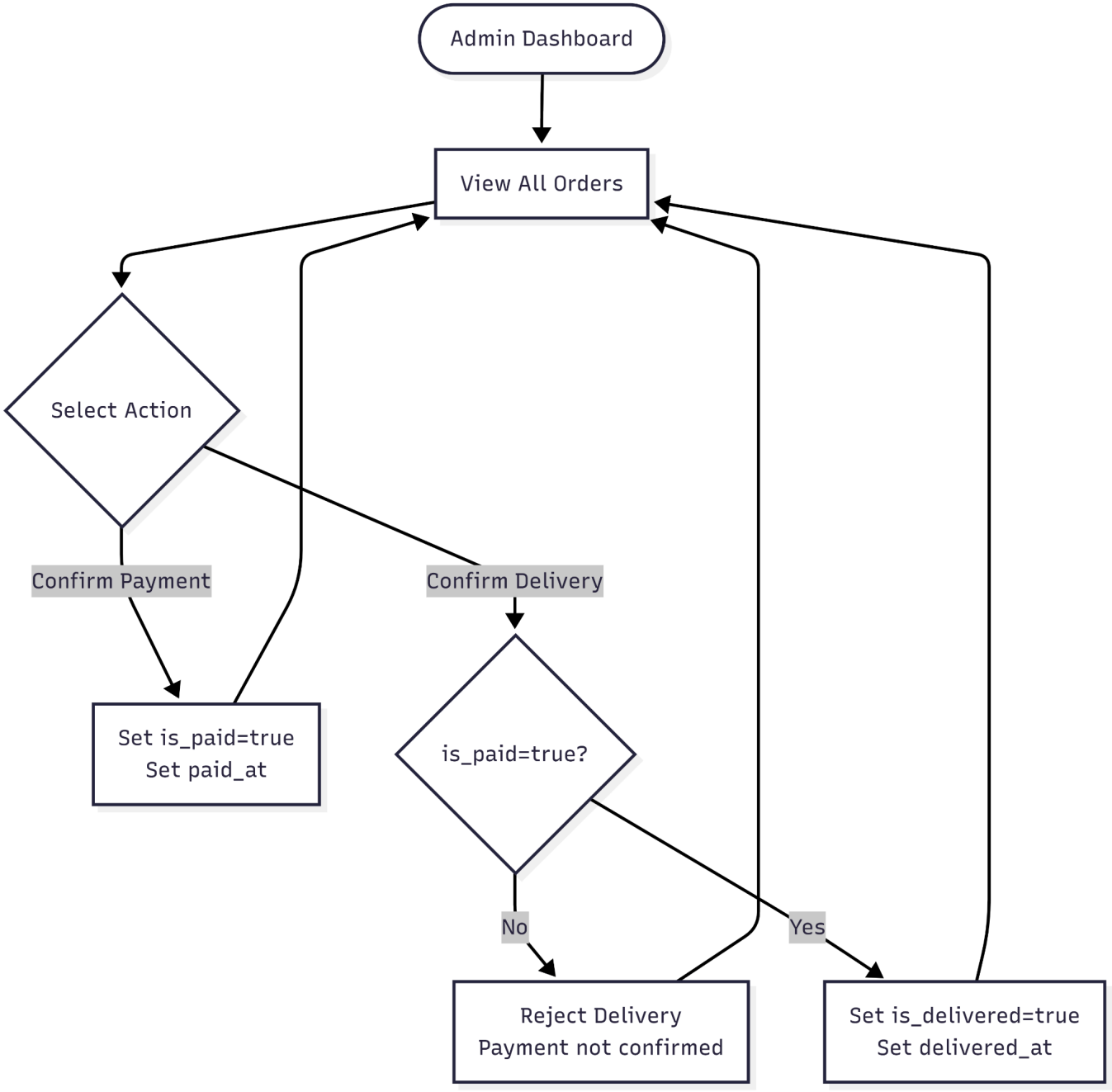

Figure 5.5 Admin Order Processing Flow

The Admin Order Processing flow focuses on managing the lifecycle of customer orders.

The administrator begins by viewing all orders and selecting a specific action.

Two key actions are supported:

- Confirm Payment: The system updates the order status by setting is_paid = true and

recording the paid_at timestamp.

- Confirm Delivery: Before delivery confirmation, the system verifies whether the order has already been paid.

If payment is not confirmed, delivery is rejected.
If payment is confirmed, the system sets is_delivered = true and records the delivered_at timestamp.
After each action, the admin is returned to the order list, allowing efficient batch processing and strict enforcement of order business rules.

5.5.3 Admin Product Management Flow

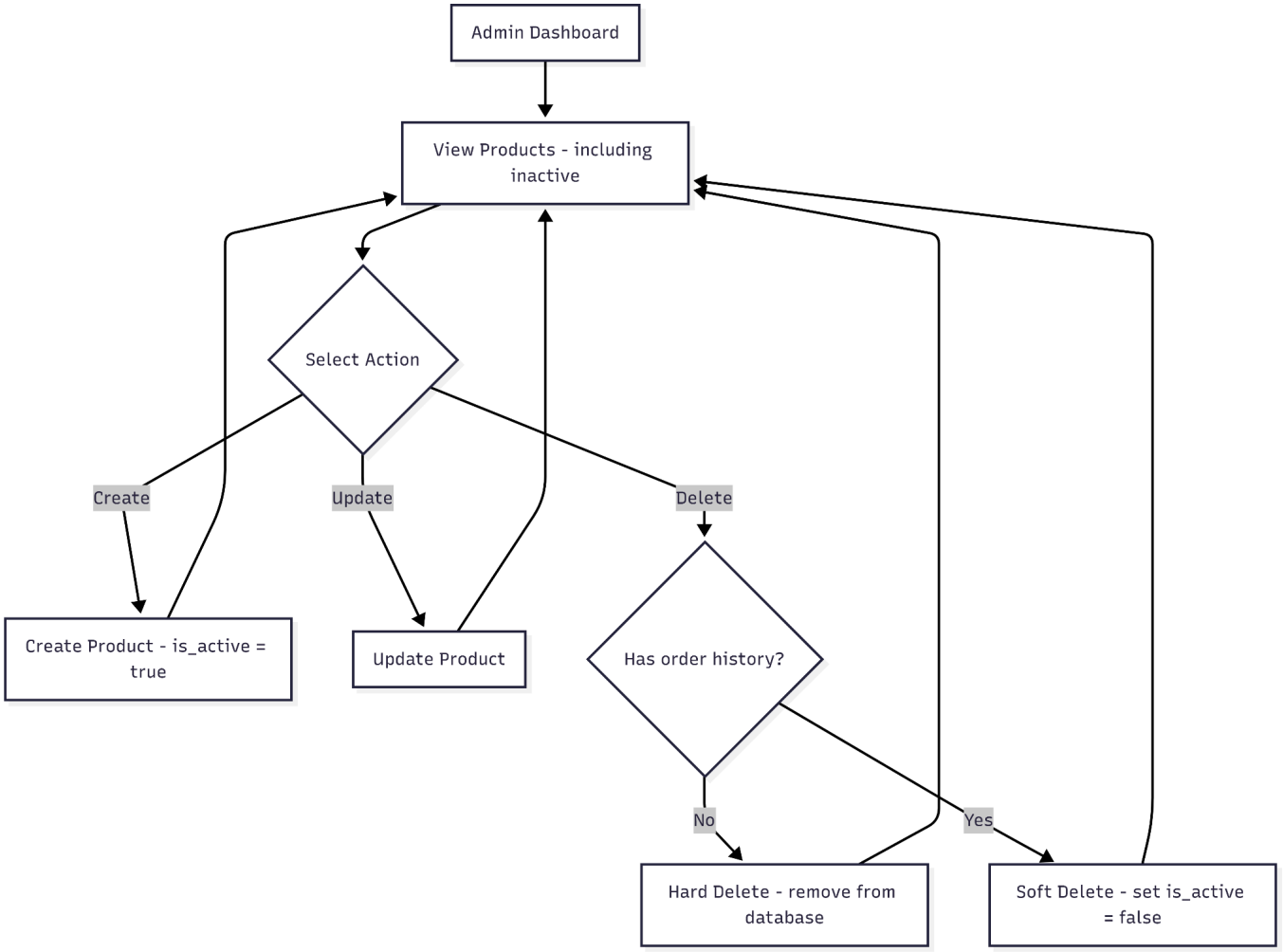

Figure 5.6 Admin Product Management Flow

The Admin Product Management flow handles product lifecycle operations, including the soft delete mechanism.
Administrators can view all products, including inactive ones, and choose between Create, Update, or Delete actions.
Create Product: A new product is created with is_active = true by default.
Update Product: Product information is modified without affecting its availability

history.

Delete Product: If the product has no order history, it is hard deleted and removed from the database.

If the product has been ordered, it is soft deleted by setting is_active = false, ensuring historical order integrity.
This flow preserves data consistency while allowing administrators to manage the product catalog safely.

5.6 Various UML diagrams

Various UML diagrams are utilized to provide a visual representation and a clear understanding of the different processes and interactions that occur within the Cake Store system. These diagrams help the development team validate business rules (authentication, stock validation, payment lifecycle, soft delete protection) and ensure consistent behavior across the API and UI.

5.6.1 Entity Relationship Diagram (ERD)

The Entity Relationship Diagram (ERD) represents the database schema of the Cake Store system and the relationships between tables.
Key relationships include:

- users – orders: One-to-many relationship, where a user can place multiple orders.

orders – order_items: One-to-many relationship, ensuring that each order contains at least one order item.

- products – order_items: One-to-many relationship, allowing products to appear in multiple orders.
- products – reviews: One-to-many relationship, enabling customers to review purchased products.
- categories – products: One-to-many relationship for organizing the product catalog.

The ERD also enforces data integrity through foreign keys and unique constraints, such as ensuring that each user can submit only one review per product. This diagram ensures consistency between the logical data model and the physical database implementation.

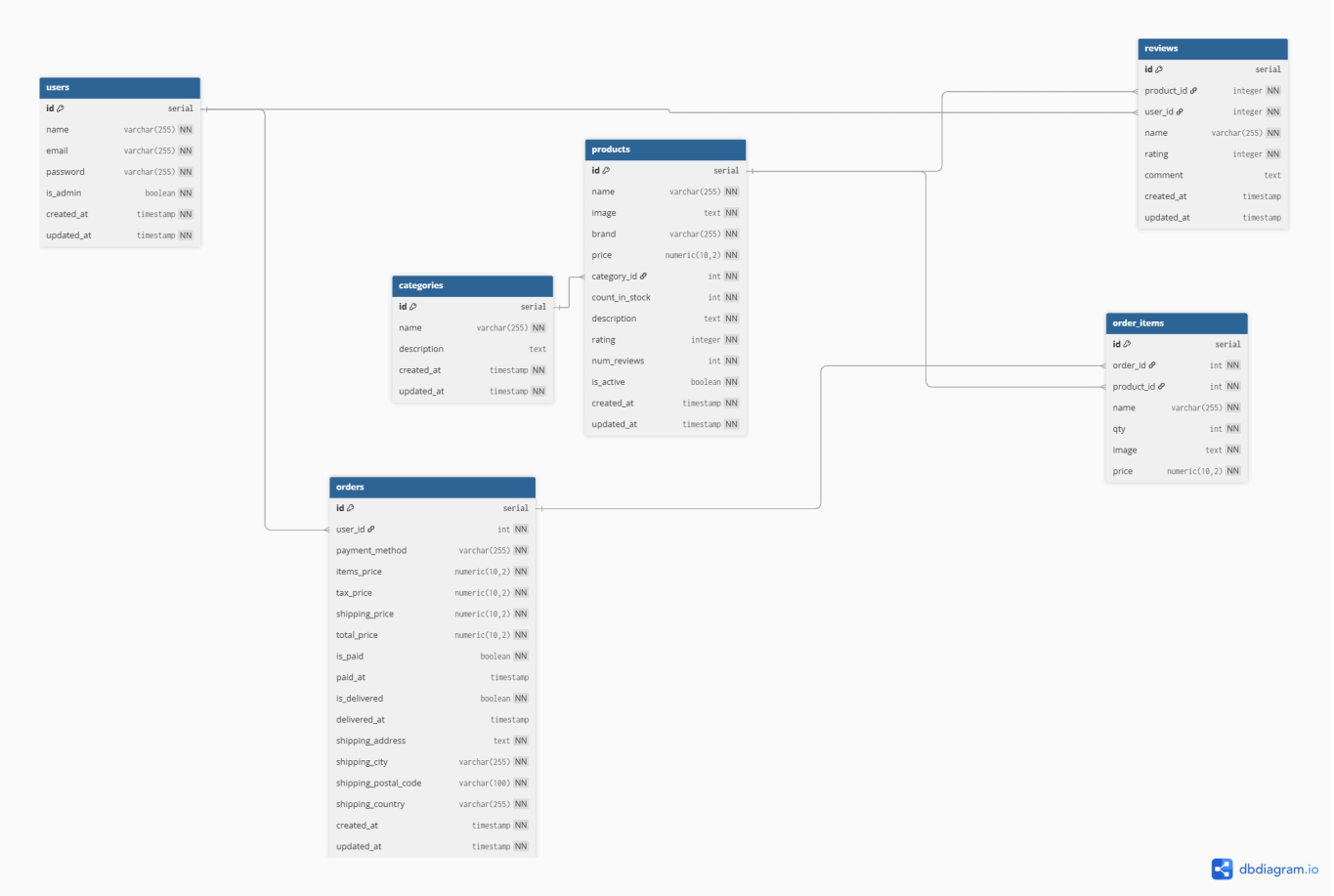

Figure 5.7 Entity Relationship - Schema Diagram

5.6.2 Class Diagram

The class diagram illustrates the static structure of the Cake Store system by defining the main domain entities and the relationships between them. It provides a clear view of how data objects are organized and how they interact within the system.

The core entities include User, Category, Product, Order, OrderItem, and Review.

A User can place multiple Orders and write Reviews for purchased products. Each Order contains one or more OrderItems, which store snapshot information of products at the time of purchase to ensure historical data integrity. Products belong to a Category and can receive multiple Reviews, from which the system automatically calculates product ratings.

To preserve order history, products that have been ordered cannot be permanently removed. Instead, a soft delete mechanism is implemented using the is_active attribute. When is_active is set to false, the product is hidden from customer views but remains accessible to administrators.

This class diagram supports clear separation of concerns, enforces business rules, and serves as a blueprint for backend implementation.

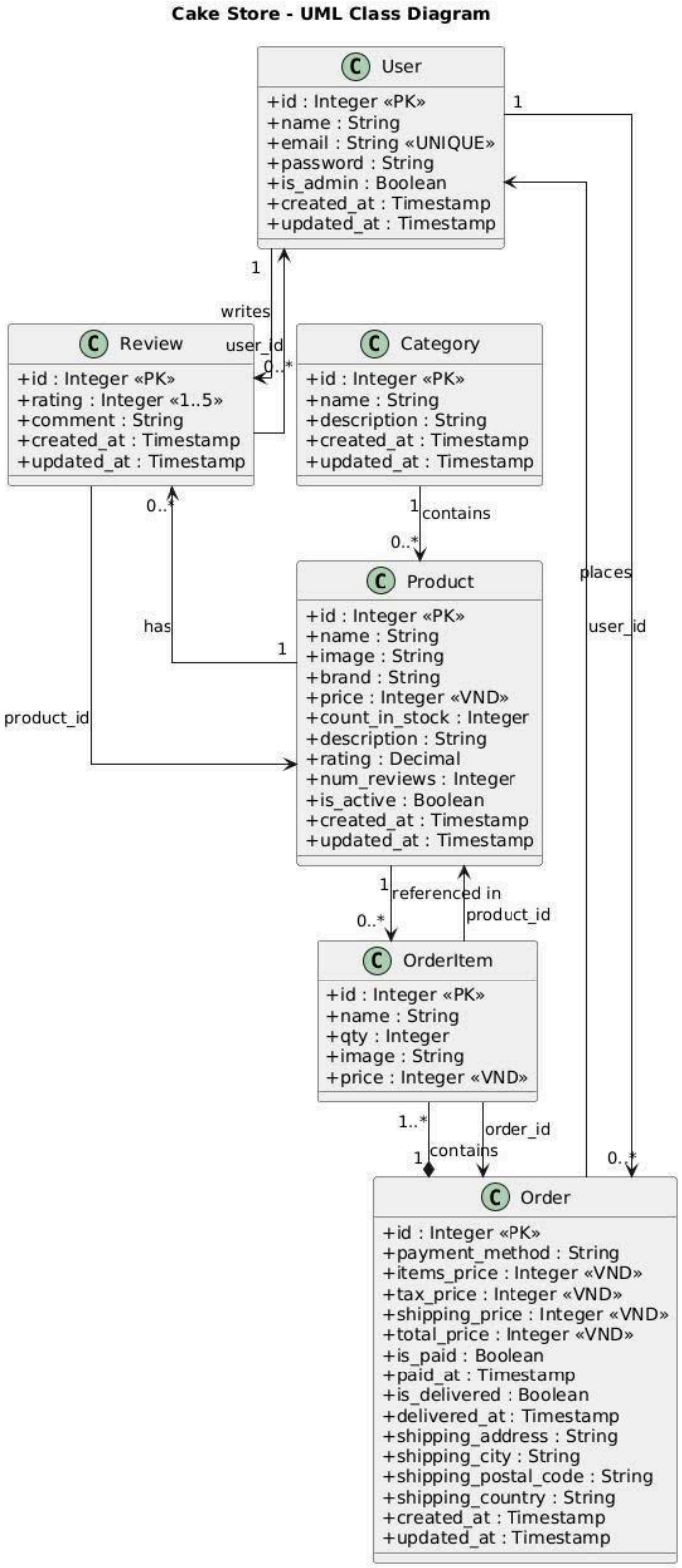

Figure 5.8 Class diagram

5.6.3 Activity Diagrams

To further illustrate the internal workflows of the Cake Store system, activity diagrams are used to model the dynamic behavior of the system during key user and administrative operations. These diagrams focus on the control flow, decision points, and conditional logic that govern how actions are processed within the system.

5.6.3.1 Activity Diagram of Login Feature

The activity diagram of the Login feature describes the authentication workflow for registered users and administrators. The process begins when a user enters login credentials and submits the login request. The system then validates the input, checks the existence of the user account, and verifies the password using secure hashing mechanisms.

If the authentication fails, the system returns an error message and prompts the user to retry. If authentication succeeds, a JSON Web Token (JWT) is generated and returned to the client, allowing the user to access protected features based on their role.

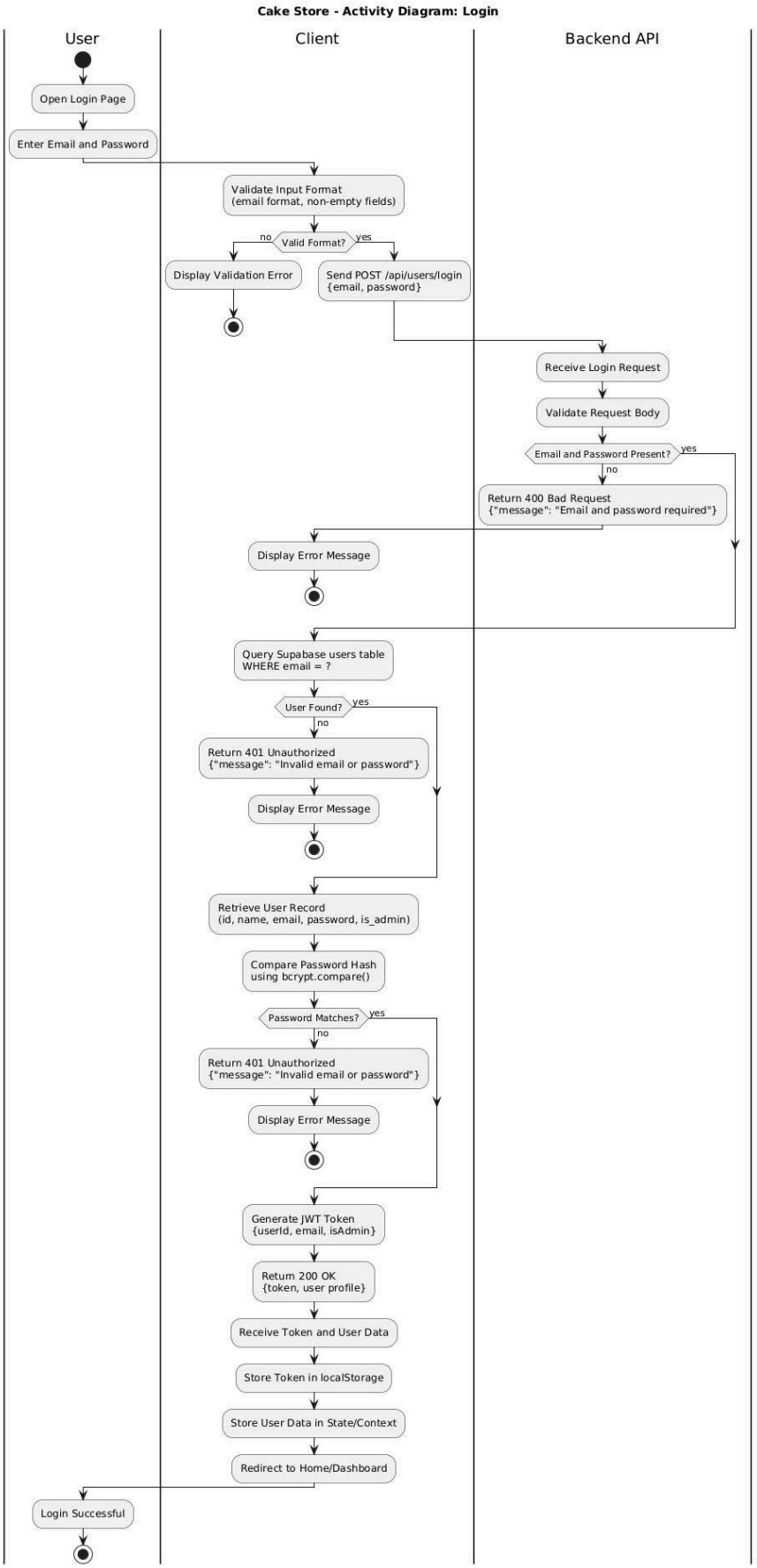

Figure 5.9. Activity diagram of Login feature

5.6.3.2 Activity Diagram of Place Order Feature

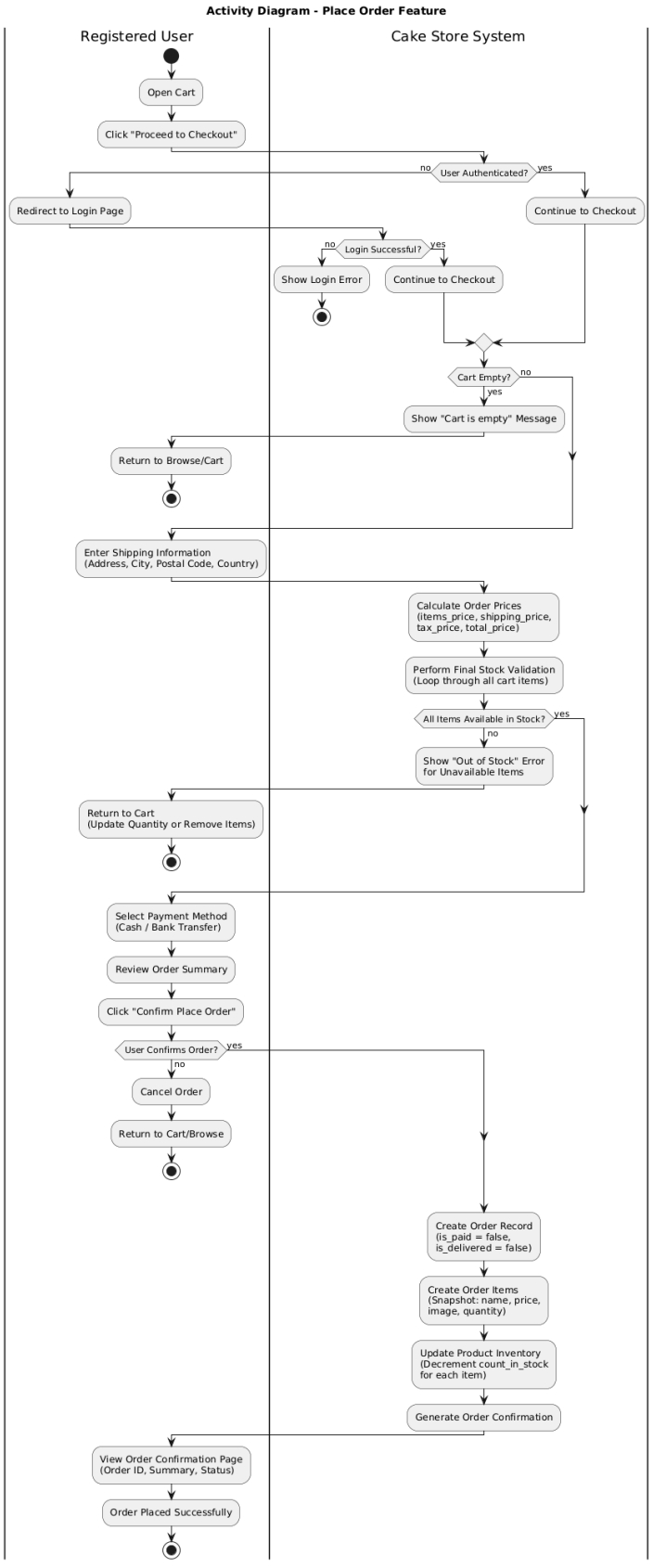

Figure 5.10. Activity diagram of Place Order feature

The activity diagram of the Place Order feature represents the checkout workflow of the Cake Store system. The process starts when a customer proceeds from the cart to checkout. The system verifies user authentication, validates shipping information, and checks product availability and stock levels.

All price calculations are performed server-side to ensure data integrity. Once the order is successfully created, the system records order details and updates product stock quantities. This diagram highlights important decision points such as authentication checks and stock validation before order confirmation.

5.6.4 Sequence Diagrams

Sequence diagrams are employed to provide a detailed view of the interactions between the client interface, backend services, and the database. These diagrams emphasize the chronological order of messages exchanged during system operations, helping to clarify responsibilities across system components.

5.6.4.1 Sequence Diagram of Login Feature

The sequence diagram of the Login feature illustrates the interaction between the user, client application, authentication controller, and database. It shows how login credentials are submitted, validated, and processed, and how the JWT token is generated and returned upon successful authentication.

This diagram ensures that authentication logic is clearly defined and securely implemented.

Figure 5.6. Sequence Diagram of Login Feature - Cake Store

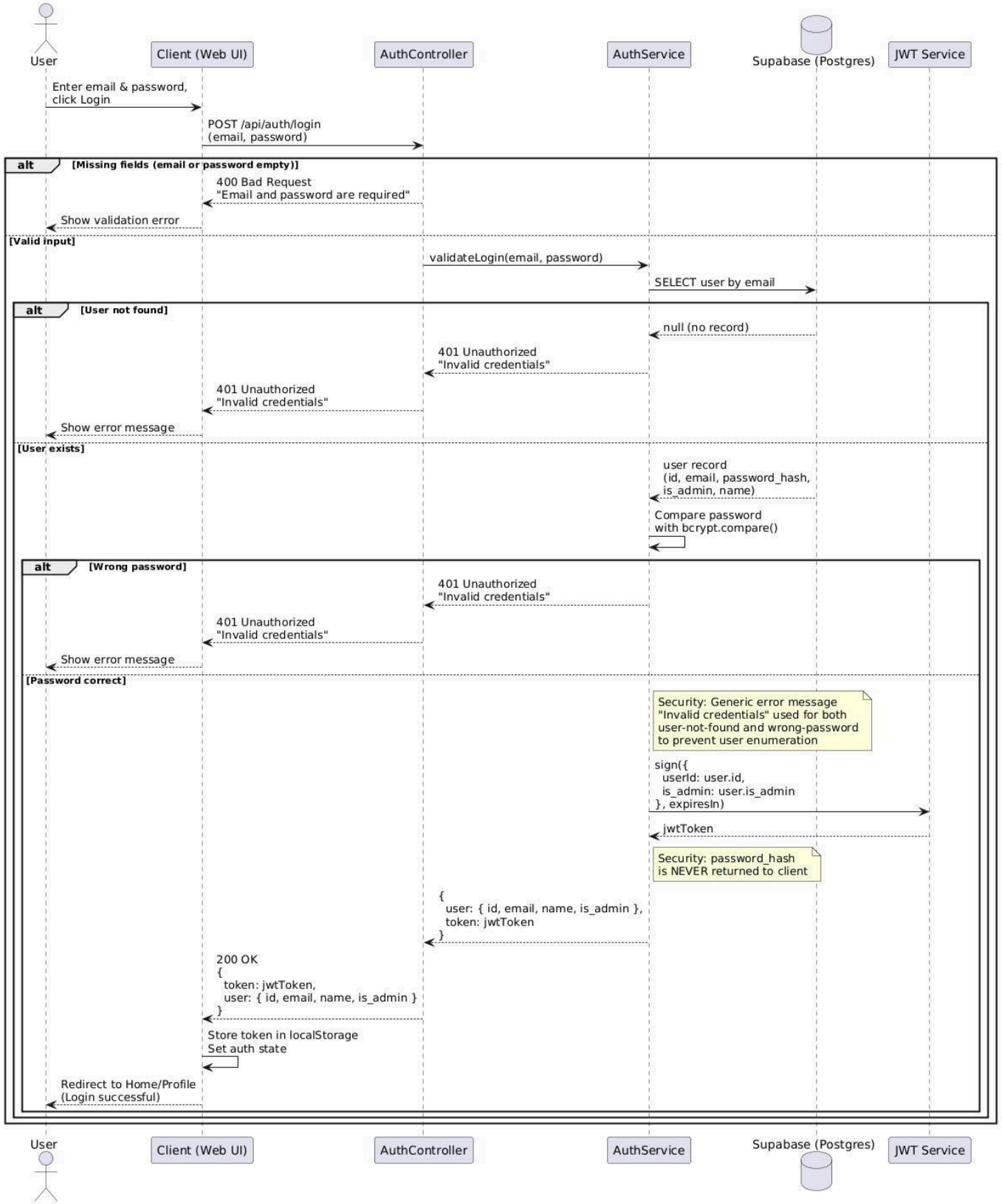

Figure 5.11. Sequence diagram of Login feature

5.6.4.2 Sequence Diagram of Place Order Feature

Figure 5.7. Sequence Diagram of Place Order Feature - Cake Store

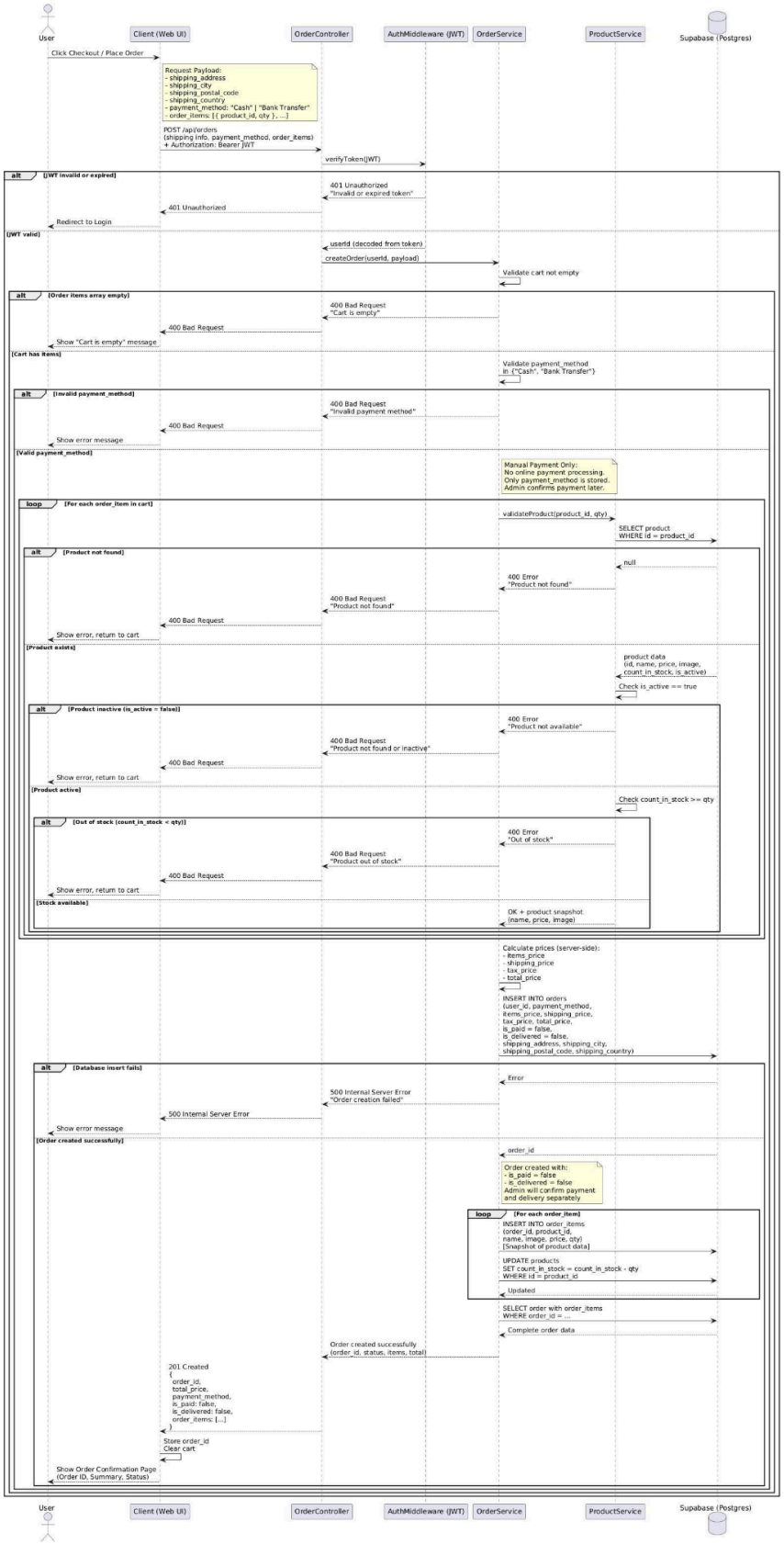

Figure 5.12. Sequence diagram of Place Order feature

The sequence diagram of the Place Order feature details the interaction flow during order creation. It includes authentication checks, product validation, stock verification, price computation, order creation, and inventory updates.

By modeling these interactions explicitly, the diagram ensures that critical business rules—such as preventing orders for inactive or out-of-stock products—are consistently enforced.

5.7 UI/UX Design Reference (Figma)

After completing the system design and UML diagrams, a Figma project was developed to visualize the user interface of the Cake Store application. The design includes screens for product browsing, cart management, checkout, and administrative dashboards.

These UI mockups help align the system design with user experience goals, ensuring that each interaction follows the defined workflows and business rules.

VI. Development

6.1. Programming Languages, Frameworks, and Tools Used

6.1.1. Backend

6.1.1.1. Languages:

JavaScript (Node.js runtime)

6.1.1.2. Framework:

Express.js – lightweight framework for building RESTful APIs and handling HTTP requests

6.1.1.3 Database:

PostgreSQL (via Supabase) – relational database used for storing users, products, orders, reviews, and categories

6.1.1.4. Backend-as-a-Service (BaaS):

Supabase – used for database management, authentication integration, and API communication

6.1.2. Frontend

6.1.2.1. Languages:

TypeScript (.ts, .tsx) - typed language, supported with type-checking and interfaces.

JavaScript (ES6+) - TypeScript compiles down to JS to run in the browser.

6.1.2.2. Framework:

React - component-based UI library.

Tailwind CSS - CSS framework for styling

6.1.2.3. Development Tool:

Vite - fast development server and bundler for React.

6.1.1.4. API

Axios (through api wrapper) - calling REST API endpoints (api.get, api.post, etc.).

6.1.3. Authentication

Authentication Method: JSON Web Tokens (JWT) – used for stateless authentication and role-based access control (RBAC)

Authorization: Role-based access control to distinguish Guest, Registered User, and Admin permissions

6.1.4. Version Control and Collaboration

Version Control System: Git – used to manage source code versions

Repository Hosting: GitHub used for source code hosting, collaboration, and issue tracking

6.2. Challenges Encountered During Development

Payment Gateway Integration: Ensuring seamless integration with the chosen payment gateway and handling various payment scenarios securely.

During the development of the Cake Store web application, several technical and design challenges were identified and addressed:

Product Deletion with Order History:

Products that had already been purchased could not be physically deleted due to database constraints. This was resolved by implementing a soft delete mechanism using an is_active flag, ensuring data integrity while hiding inactive products from customers.

Manual Payment Workflow:

Instead of real-time payment gateway integration, the system supports a manual payment confirmation process. Designing this workflow required strict enforcement of business rules to prevent delivery confirmation before payment confirmation.

State Management Complexity:

Managing cart data, authentication state, and order status across multiple pages required a centralized state management solution.

Role-Based Access Control (RBAC):

Ensuring that only administrators could access management features while customers had restricted permissions required careful backend authorization checks and frontend route protection.

6.3. Code Management and Version Control

To maintain code quality and ensure effective collaboration, the following practices were applied:

Version Control: Git was used to track changes and manage different versions of the codebase.

Branching Strategy: Two separate working branches, BE and FE, were developed separately and then merged.

A feature-based branching method was applied, where new features and bug fixes were developed on separate branches before being merged into the main branch.

Commit Practices: Meaningful commit messages were used to clearly describe changes and improve traceability.

Code Reviews: Changes were reviewed before merging to ensure consistency, correctness, and adherence to coding standards.

VII. Testing and Quality Assurance

The test results were systematically recorded and managed throughout the development process to ensure traceability and timely issue resolution. Identified bugs and defects were documented and prioritized based on their severity and impact on core system functionalities. This structured approach enabled the development team to address critical issues promptly and apply iterative improvements to the Cake Store system.

Quality assurance played a crucial role in maintaining system reliability and consistency. Multiple QA measures were implemented, including:
Unit Testing: Core business logic such as authentication, order creation, and admin operations was tested in isolation.
Integration Testing: API endpoints were tested to verify correct interaction between the frontend, backend, and database.
Manual Testing: Key user flows—including login, product browsing, order placement, and admin management—were manually tested to validate real user scenarios.
Business Rule Validation: Special attention was given to enforcing rules such as:
Only paid orders can be delivered
Only purchased products can be reviewed
Inactive products are hidden from customer views

By integrating these testing and quality assurance practices into the development workflow, the Cake Store application achieved a stable, secure, and user-friendly system that meets both technical requirements and user expectations.

VIII. Deployment and Implementation

The deployment and implementation phase of the Cake Store web application was conducted in a structured manner to ensure system stability, reliability, and ease of maintenance.

The deployment process followed a staged approach, beginning with configuring the backend API and database environment, followed by frontend deployment. Environment variables were carefully set to manage database connections, authentication secrets, and API endpoints. Automated build tools were used to reduce human error and ensure consistency across environments.

Before public release, smoke tests were executed to verify core functionalities such as user login, product browsing, order placement, and admin operations. These tests ensured that the system operated correctly in the production environment.

A gradual rollout strategy was adopted, starting with internal testing by the development team, followed by limited user testing. Feedback collected during this phase was used to refine usability and fix minor issues before full deployment.

Post-deployment, a maintenance plan was established to monitor system health, apply bug fixes, and update dependencies when necessary. This approach ensures long-term stability and scalability of the Cake Store platform.

IX. User Documentation

9.1. User Guide

This section provides guidance for end-users on how to use the Cake Store system effectively.

9.1.1. Sign up

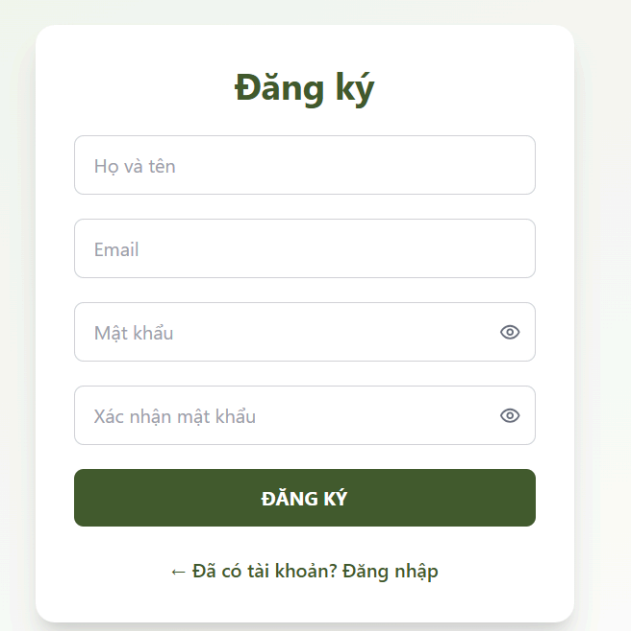

Figure 9.1. Screen of Sign up

ID	Name	Description
1	Username	User enter their username
2	Email	User fill the email
3	Password	User enter their password that they want
7	"Đăng ký" button	After finishing all of the above, User clicks Sign up completely.

9.1.2. Log in

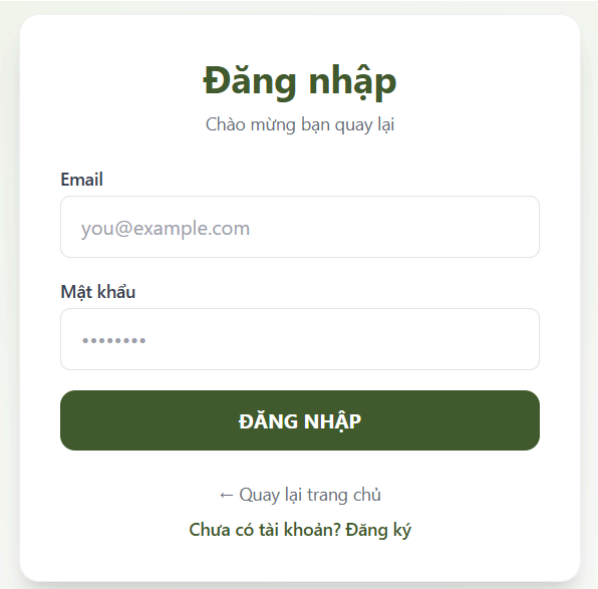

Figure 9.2. Screen of Login

ID	Name	Description
1	Username	User enter their username
2	Password	User enter the password that saved
3	Button “Đăng nhập”	User click button “Log in” to move homepage

9.1.3. Product Browsing and Search

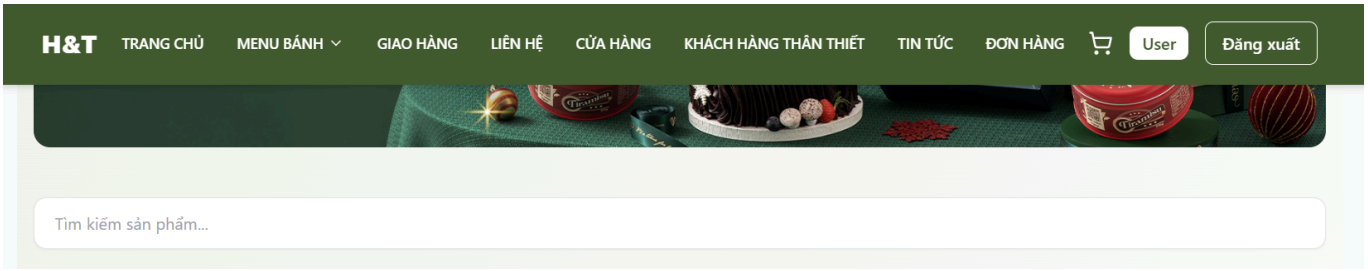

Figure 9.3. Screen of Search

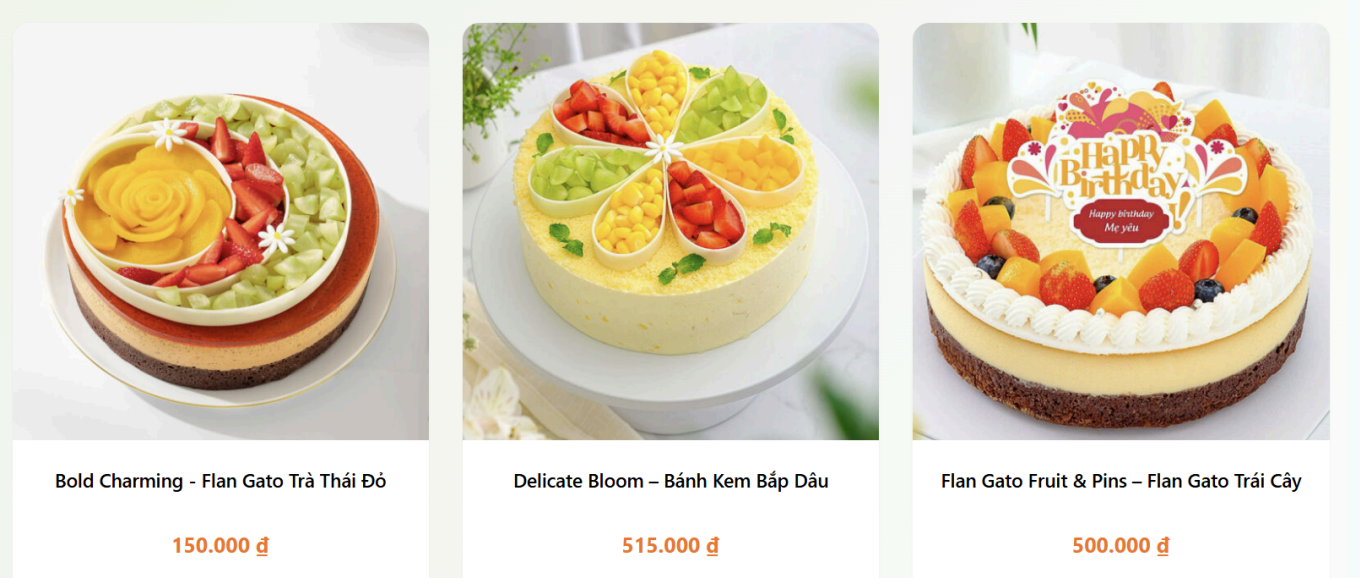

Figure 9.4. Screen of Product List

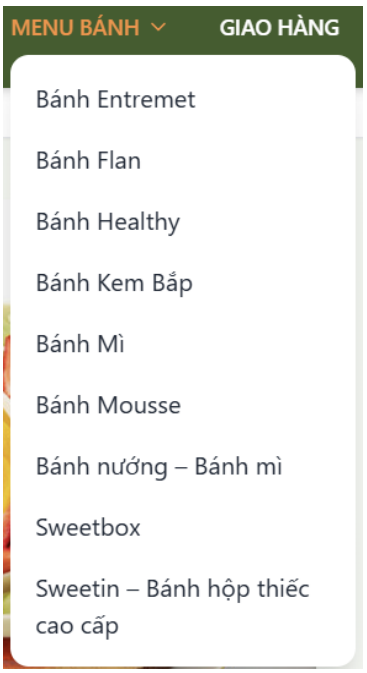

Figure 9.5. Screen of Filter Options

ID	Name	Description
1	Search Bar	User enters product keywords
2	Product List	Displays active products only
3	Filter Options	Filter by category or price

9.1.4. Order Placement

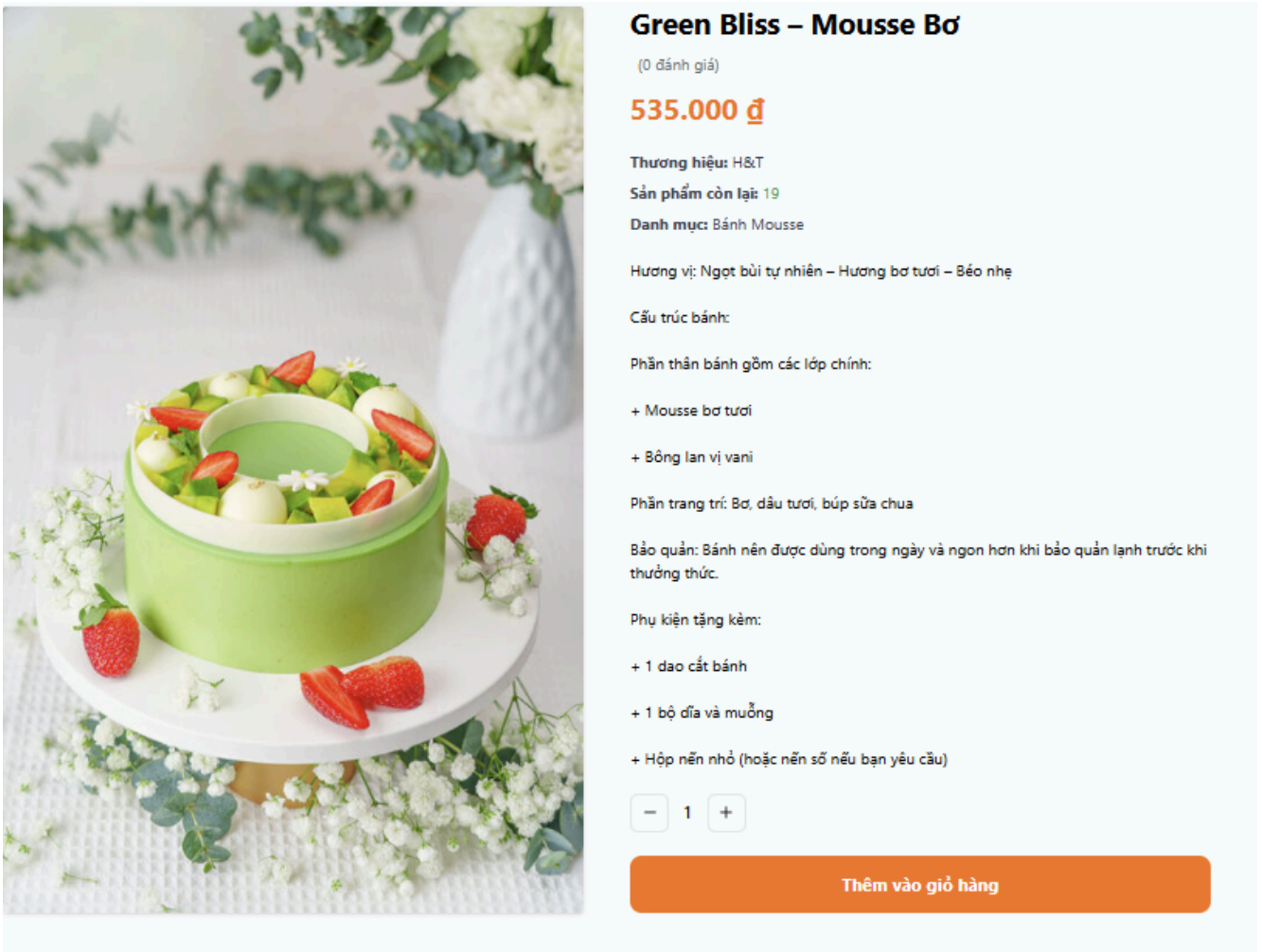

Figure 9.6. Screen of Product Detail

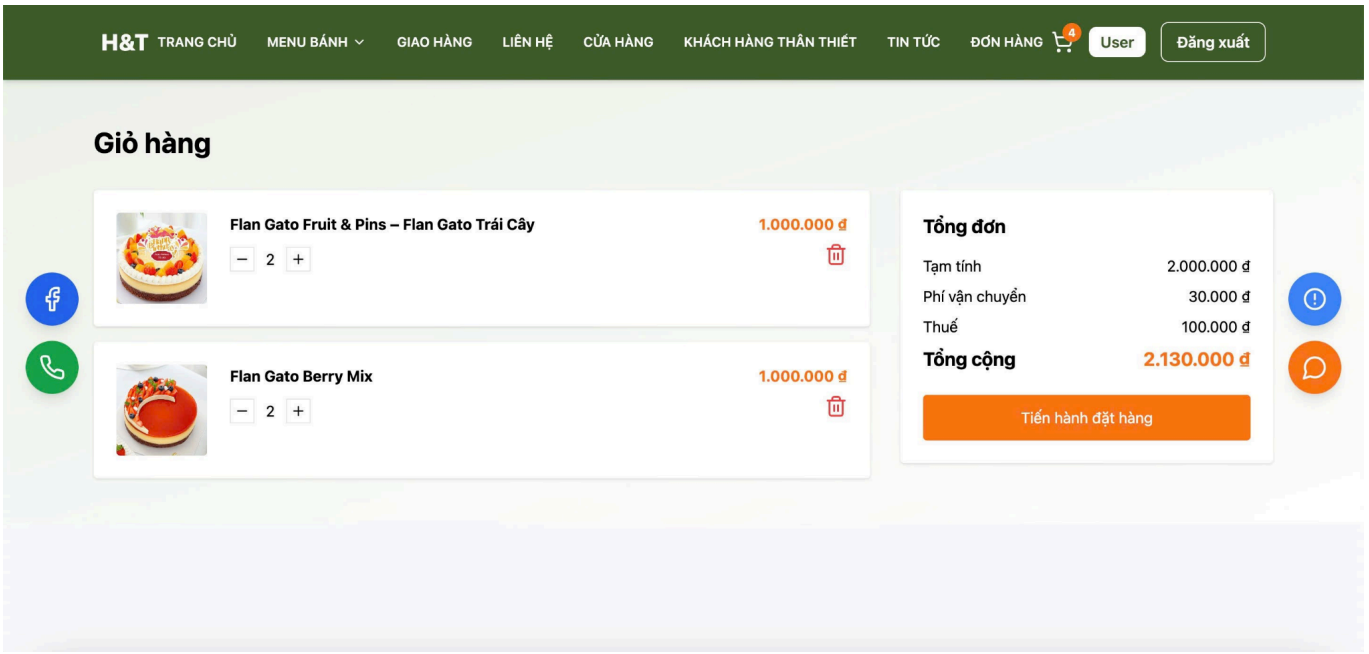

Figure 9.7. Screen of Add to Cart

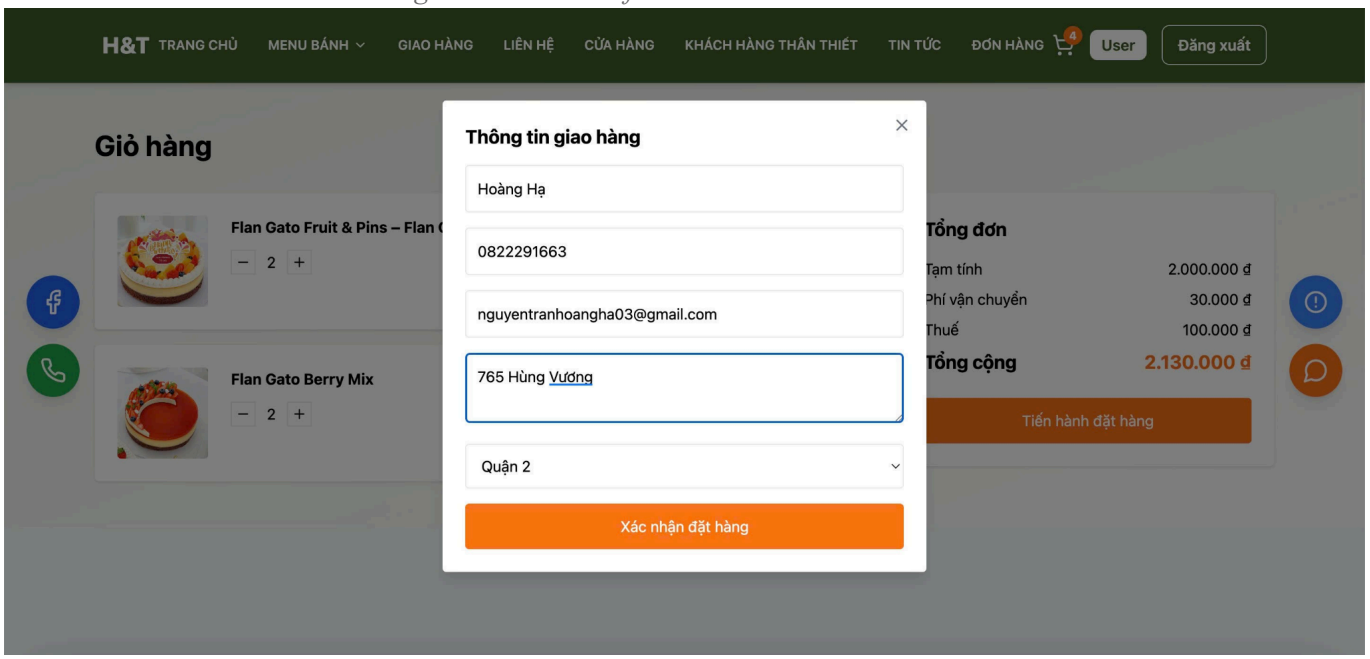

Figure 9.8. Screen of Place Order

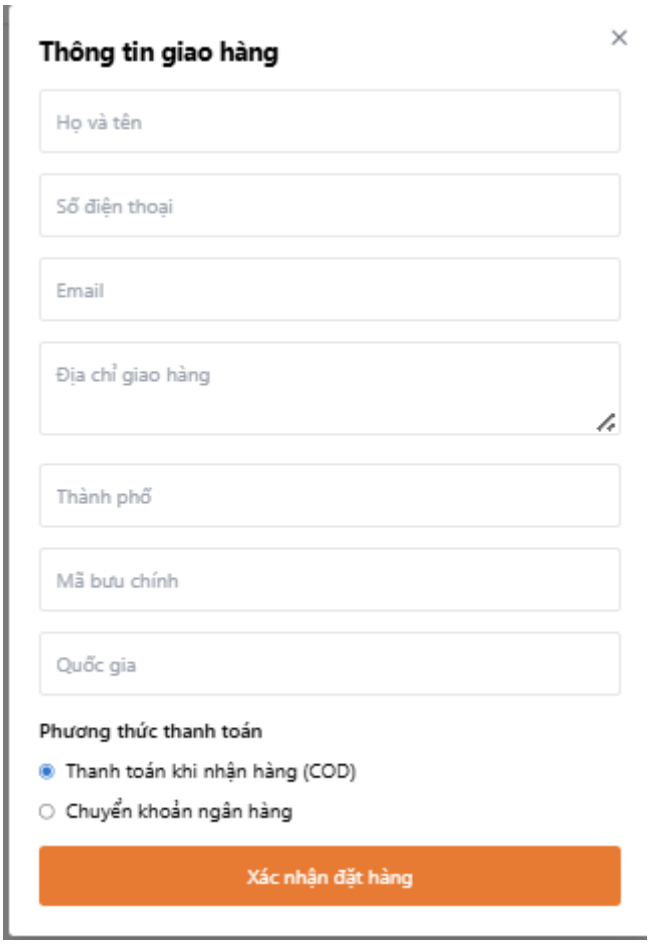

Figure 9.9. Screen of Shipping Information

ID	Name	Description
1	Product Detail	Displays product name, price, description
2	Add to Cart	Adds product to shopping cart
3	Place Order	Proceeds to checkout
4	Shipping Information	User enters delivery details

9.2. System requirements for end-users

9.2.1. Hardware Requirements:

- Personal Computer, Laptop, Tablet, or Smartphone
- Internet connectivity for online booking

9.2.2. Operating System

- Compatibility with popular operating systems such as Windows, macOS, Linux, Android, and iOS.

9.2.3. Web Browser

- Support for major web browsers such as Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.

9.2.4. User Interface:

- Intuitive and user-friendly interface for ease of use.
- Responsiveness for various screen sizes and resolutions.

9.2.5. Security

- Secure Sockets Layer (SSL) for data encryption during online transactions.
- Secure user authentication and authorization mechanisms.

9.2.6. Accessibility

- Compliance with accessibility standards (e.g., WCAG) to ensure usability for people with disabilities.

9.3. Troubleshooting tips

1. Login Issues:

   Problem: Users can't log in.

   Tip: Reset the password or check for typos in the username/password.

2. Order Errors

   Tip: Refresh page, Check product availability

3. Payment Issues:.

   Tip: Verify order status, Contact administrator if needed

4. Session Timeout:

   Problem: Users are unexpectedly logged out.

   Tip: Refresh the page, log in again, or adjust session timeout settings.

5. Slow Performance:

   Problem: System response is slow.

   Tip: Check internet connection, close unnecessary tabs, or try during non-peak hours.

X. Lessons Learned

10.1. Reflection on what went well and what could be improved.

Throughout the development of the Cake Store web application, several positive outcomes were achieved. Firstly, investing time in clear requirement analysis at the early stages helped establish a shared understanding of business rules among team members, particularly regarding order processing, product lifecycle management, and role-based access control. This significantly reduced ambiguity during implementation.

Secondly, regular communication and task coordination contributed to effective collaboration and transparency. Weekly discussions allowed the team to identify blockers early and align technical decisions with functional requirements.

However, the project also revealed areas for improvement. Inadequate documentation during the initial development phase made knowledge transfer more difficult, especially when revisiting earlier design decisions such as soft deletion and order status handling. This highlighted the need for more consistent technical documentation and diagram updates throughout the project lifecycle.

10.2. Insights gained from project challenges

One key insight gained from this project is the importance of maintaining a well-defined project scope. Clearly distinguishing between customer-facing features and administrative operations minimized misunderstandings and helped streamline development efforts.
Another significant insight is the value of user involvement. Early feedback from test users helped refine user flows, improve navigation, and clarify error-handling behavior, especially in the order placement process.
Additionally, the project emphasized that agility requires both technical and organizational readiness. While the system architecture supported iterative development, adapting quickly to requirement changes required improved planning flexibility and clearer communication channels.
Finally, the team learned that well-maintained documentation is essential for long-term maintainability. Updated diagrams, clear API descriptions, and consistent naming conventions proved invaluable when extending features or debugging complex workflows.

10.3. Recommendations for future projects

Based on the experiences gained, several recommendations can be made for future projects:

Prioritize comprehensive requirement discussions at the beginning of the project to establish a solid foundation.
Maintain continuous and structured communication among team members to ensure alignment.
Implement systematic user feedback mechanisms to validate design decisions early.
Foster a more agile mindset by adopting shorter development cycles and

iterative improvements.
Emphasize thorough documentation, covering system architecture, business rules, and code structure.
Strengthen project management practices by incorporating real-time progress tracking and proactive issue resolution.stipulated timeline.
We have seamlessly integrated a variety of key features, such as intuitive seat selection, real-time availability updates, and secure payment processing, resulting in a highly efficient booking system. The execution of a comprehensive testing strategy has ensured the platform's functionality, security, and performance, meeting our high- quality standards and exceeding user expectations.
Key features such as intuitive ticket selection, real-time availability updates, and secure payment processing have been seamlessly integrated, resulting in a highly efficient booking system. Comprehensive testing ensured the platform's functionality, security, and performance, exceeding user expectations.
Key achievements include implementing a scalable architecture, developing a thorough testing strategy, and executing a smooth deployment process. We also ensured a secure environment for users' data.
As we conclude this project, we reflect on the valuable insights gained and improvements made. We are committed to ongoing enhancements and dedicated support, ensuring we remain at the forefront of the digital entertainment industry.

XI. References

1. Tiệm bánh kem - bánh sinh nhật chất lượng cao, đẹp mắt tại HCM
2.

XII. Appendices

The appendices provide supplementary information for readers who wish to explore the technical details of the project further. These include:

Database Entity Relationship Diagram (ERD)
System Class Diagram
API Endpoint Specifications
Git workflow and commit history
This section supports the main report without overwhelming it with excessive technical detail.

XIII. Acknowledgments

Throughout the development of this project, the team gained invaluable experience in planning, collaboration, and problem-solving. Each member

contributed actively to their assigned tasks while continuously improving technical and communication skills.

Although challenges were encountered, particularly in documentation and requirement refinement, the project has strong potential for further development. With continued improvements, the Cake Store system could evolve into a production-ready e-commerce platform.

-THE END