



Debre Berhan University

College of Computing

Department of Software Engineering Internship

Report

**PROJECT TITL: CAFÉ MEAL ORDER MANAGEMENT
SYSTEM**

By: Yenenesh Dabot

Mentor: Mr. Erku K.

Company Supervisor: Mr. Zerabruk Assefa

September 2025

Mentor: Mr. Erku Kifle

Table of Contents

Executive Summary	1
1. Introduction	2
1.1. Background of the Company	3
1.2. Mission of the Company	3
1.3. Vision of the Company	3
1.4. Main Products and Services of the Company	3
1.5. Main Customers and End Users of the Company	4
2. Overall Internship Description and Activities	5
2.1. Objective of the Internship.....	5
2.1.1. General Objective.....	5
2.1.2. Specific Objectives.....	5
2.2. Nature of Work / Tasks Carried Out	6
2.2.1. Week One Internship Report.....	6
2.2.2. Week Two Internship Report.....	7
2.2.3. Week Three Internship Report	8
2.2.4. Week Four Internship Report	8
2.2.5. Week Five Internship Report.....	9
2.2.6. Week Six Internship Report	10
2.2.7. Week Seven Internship Report.....	11
2.2.8. Week Eight Internship Report	12
2.2.9. Week Nine Internship Report.....	13
2.2.10. Week Ten Internship Report.....	13
2.2.11. Week Eleven Internship Report.....	13
2.2.12. Week Twelve Internship Report	14
2.2.13. Week Thirteen Internship Report	15
2.2.14. Week Fourteen Internship Report	16
2.3. Section I have been working.....	16

3. Experience and Transferable Skills.....	18
3.1. Work Experience	18
3.1.1. Technical Experience	18
3.1.2. Non-Technical Experience	19
3.2. Benefits of Experience	19
3.3. Skills Gained (Soft and Hard Skills).....	20
3.4. Challenges Faced and Solutions Taken.....	20
4. General Overview of the project	22
4.1. Executive Summary	22
4.2. Background of the Project	22
4.3. Statement of the Problem.....	23
4.4. Project Significance	24
4.5. Objective the project	24
4.5.1. General Objective.....	24
4.5.2. Specific Objectives.....	24
4.6. Scope of the Project	25
4.7. Functional and nonfunctional requirement	26
4.7.1. Functional requirement	26
4.7.2. Non-Functional Requirements	27
4.8. Methodology	28
4.8.1. Development Tools.....	28
4.9. System Analysis and Modeling.....	29
4.9.1. Use Case Diagram.....	29
4.9.2. Class Diagram	37
4.9.3. Sequence Diagrams	39
4.9.4. State Chart Diagram	46
4.9.5. Activity Diagrams	49
4.9.6. Collaboration Diagrams	53
4.9.7. Database Design.....	54
4.9.8. Deployment Diagram	57

5. Conclusion and Recommendation	59
6. Appendix	61
6.1. Appendix A: Project Folder Structure	61
6.2. Appendix B: API Source Code	62
6.3. Appendix C: Source Code Demo	65
7. Reference	65

List of Figure

Figure 1 Use case diagram	30
Figure 2 Class Diagram	38
Figure 3 user registration Sequence	40
Figure 4 User login Sequence	40
Figure 5 Add new user Sequence	41
Figure 6 Edit user Sequence	43
Figure 7 Delete user Sequence	44
Figure 8 Order processing Sequence	45
Figure 9 Registration state chart diagram	46
Figure 10 Add user state chart diagram	47
Figure 11 Add product state chart diagram	48
Figure 12 Order state chart diagram	49
Figure 13 Registration activity diagram	50
Figure 14 Login activity diagram	51
Figure 15 Add user activity diagram	52
Figure 16 Add product activity diagram	52
Figure 17 Collaboration Diagrams	54
Figure 18 Deployment diagram	58

List of Tables

Table 1 Assign IDs for Use Cases	30
Table 2 UC-01: Register	31
Table 3 UC-02: Login	32
Table 4 UC-04: Manage User	33
Table 5 UC-05: Approve Register	33
Table 6 UC-06: Manage Product	34

Table 7 UC-07: Manage Category.....	34
Table 8 UC-08: View Order Status.....	35
Table 9 UC-09: Change Order Status.....	35
Table 10 UC-10: View Request Order.....	36
Table 11 UC-11: Distribute Order.....	36
Table 12 UC-12: Generate Receipt.....	36
Table 13 UC-13: View Report.....	37
Table 14 Users Table.....	55
Table 15 Roles Table.....	55
Table 16 Products Table.....	56
Table 17 Orders Table.....	56
Table 18 Receipts (Bill) Table.....	57

List of Acronyms

Acronym	Full Form
CMOMS	Cafe Meal Order Management System
OTMS	Oromiya transport management system
DB	Database
ERD	Entity Relationship Diagram
UI	User Interface
UX	User Experience
PK	Primary Key
FK	Foreign Key
SQL	Structured Query Language
UML	Unified Modeling Language
API	Application Programming Interface
CRUD	Create, Read, Update, Delete
SRS	Software requirement specification
ID	Identifier
UC	Use case

Executive Summary

The report contains general and specific ideas, background, main products, customers, objectives, mission, and vision of DAFTech Social ICT Solutions PLC. In addition to the company profile, the report details the objectives of the internship, the knowledge gained during the on-job training—both practical and theoretical—and the tasks performed throughout the internship period. The problems faced during the internship, identified issues, and proposed solutions are also discussed. The designed project on the selected problem, along with the measures taken, is included in this report.

1. Introduction

DAFTech Social ICT Solutions PLC is a prominent software company in Ethiopia, dedicated to providing innovative IT solutions that tackle challenges across various sectors. The company specializes in developing advanced software systems, mobile applications, artificial intelligence solutions, and comprehensive IT services.

DAFTech focuses on empowering IT companies by providing tools and resources to scale engineering capacity, achieve optimal growth, and meet customer expectations in a rapidly evolving technology landscape.

Core Activities:

- **Innovative Solutions:** Creating software solutions to address real-world challenges faced in Ethiopia.
- **System Design:** Specializes in architectural design, component/module design, database design, UI design, and performance & scalability design.
- **Mobile Design:** Offers UX/UI design, responsive and interaction design, prototyping, testing, and handoff for intuitive and user-friendly mobile apps.
- **Artificial Intelligence:** Conducts AI research, data handling, algorithm development, model creation, training, integration, and continuous innovation.
- **Consultation & Support:** Provides expert advice, technical support, and creative design services to clients.

1.1. Background of the Company

DAFTech Social ICT Solutions PLC is an Ethiopian software company committed to delivering cutting-edge IT services. The company has established a strong track record in providing high-quality software solutions, mobile applications, and AI systems. With over 2,000 satisfied clients, 20 completed projects, and a team of skilled experts, DAFTech has become a trusted name in Ethiopia's IT sector.

1.2. Mission of the Company

DAFTech Social ICT Solutions PLC's mission is to empower IT companies and individuals by providing the tools, resources, and support needed to innovate, scale software solutions, and achieve sustainable growth in the digital economy.

1.3. Vision of the Company

DAFTech envisions becoming a leading software company in Ethiopia and Africa, driving technological innovation, delivering world-class IT solutions, and enabling clients to leverage technology for maximum business impact. The company seeks to foster a tech ecosystem where innovative solutions contribute to economic development and digital transformation.

1.4. Main Products and Services of the Company

DAFTech Social ICT Solutions PLC provides a wide range of innovative IT products and services aimed at addressing challenges in Ethiopia and beyond. These include:

- **Innovative Solutions:** DAFTech delivers advanced software solutions tailored to solve real-world problems, helping businesses and institutions achieve operational efficiency and technological advancement.
- **System Design:** The company specializes in architectural design, component and module design, database design, user interface design, and performance and scalability optimization to ensure high-quality and reliable software systems.

- **Mobile Application Design:** DAFTech’s mobile app development process includes UX and UI design, responsive design, interaction design, prototyping, testing, iterative design, and seamless collaboration with development teams to produce intuitive and user-friendly applications.
- **Artificial Intelligence (AI) Solutions:** The Company adopts a comprehensive AI development approach that covers research and planning, data management, algorithm selection, model creation, training and evaluation, integration, and ongoing innovation to deliver smart, data-driven solutions.

These services position DAFTech as a leader in software development, mobile applications, and AI technologies, providing end-to-end IT solutions for clients across multiple sectors.

1.5. Main Customers and End Users of the Company

DAFTech serves a diverse set of clients, including:

- Government agencies and institutions requiring IT solutions and system optimization.
- Private sector companies seeking software, mobile, and AI solutions.
- Educational and research institutions requiring digital tools and application support.
- International organizations collaborating on technological projects.

End-users

The end users of DAFTech’s services include IT companies, entrepreneurs, students, business professionals, and the broader tech community in Ethiopia. Clients benefit from tailored IT solutions, software systems, and mobile applications that enhance productivity, efficiency, and innovation.

2. Overall Internship Description and Activities

2.1. Objective of the Internship

2.1.1. General Objective

The objective of the internship is to provide practical exposure and reflection on experiences gained while bridging academic knowledge with real-world applications. The internship offers students an opportunity to explore their career interests before committing to permanent roles, develop technical and professional skills, and enhance interpersonal abilities. Interns gain hands-on experience that complements their academic learning and prepares them for future employment in the Information Technology sector. Participation in an internship increases confidence, practical skills, and employability for future professional opportunities.

2.1.2. Specific Objectives

- To provide intensive field experience with hands-on work in software development and IT solutions.
- To receive exposure to the operations of an IT company and understand its organizational design and structure.
- To develop knowledge of management and technical skills, communication methods, and decision-making processes within an IT organization.
- To apply theoretical knowledge, technical skills, and professionalism gained during academic coursework to real-world projects.
- To allow DAFTech Social ICT Solutions PLC to benefit from a trainee's up-to-date understanding of technology and provide an objective perspective.
- To help the intern make informed decisions about a future career in Information Systems and software development.
- To prepare the intern for future non-internship employment by gaining practical experience, developing professional documents, and learning workplace protocols.

2.2. Nature of Work / Tasks Carried Out

DAFTech Social ICT Solutions PLC assigns interns to various technical teams based on their area of interest. For this internship, I was assigned to the web design and development team.

Since there were no ongoing client projects during my internship period, I was given a Library Management System as a project to work on. My responsibilities included:

- Designing the frontend of the system using Angular.
- Implementing backend logic using C#/.NET for CRUD operations.
- Integrating database functionality to store book, member, and transaction information.
- Applying best practices in UI/UX design, ensuring a responsive and user-friendly interface.
- Testing the system for functionality, security, and data integrity.

This hands-on project provided me with practical experience in full-stack development and a deeper understanding of how DAFTech designs and implements software solutions for clients.

2.2.1. Week One Internship Report

During the first week of my internship, I focused on setting up my development environment and learning the fundamentals of Angular. I researched frameworks relevant to my project, particularly Angular, and compared it with React to understand their differences.

I explored Angular concepts such as components, templates, forms, and data binding. I also researched best practices for structuring Angular projects to maintain scalability.

Towards the end of the week, I added more functionality to the app, such as deleting tasks, marking them as completed, and incorporating RxJS to handle asynchronous data. Over the weekend, I completed additional Angular tutorials on platforms like Codecademy and freeCodeCamp, and practiced routing and lifecycle hooks for better navigation and state management.

Challenges Faced

- Difficulties installing Angular CLI due to version conflicts with Node.js and npm, which I resolved by updating the necessary software.
- Initial confusion with Angular's two-way data binding, which I overcame by working through examples and consulting documentation.

Achievements

- Successfully installed and configured Angular for development.
- Built the initial structure of a To-Do List app with CRUD operations.
- Gained experience with TypeScript, data binding, routing, and RxJS.
- Strengthened my understanding of Angular components, forms, and project organization.

2.2.2. Week Two Internship Report

Activities

During the second week, I developed the basic structure of the Todo List app and implemented task addition features. I spent time debugging UI issues to improve responsiveness and researched TypeScript concepts to strengthen my Angular knowledge. I also studied Angular's state management approaches and applied best practices to ensure smooth data flow within the app. By the end of the week, I enhanced the UI/UX design and took a short pause on Angular routing to better balance my workload.

Challenges Faced

- Debugging UI responsiveness issues while managing state across components.
- Initial difficulty understanding advanced TypeScript features in the Angular context.

Achievements

- Successfully created the base structure of the Todo List app.
- Implemented task addition functionality and improved responsiveness.
- Enhanced understanding of state management in Angular.
- Improved UI/UX design of the app.

2.2.3. Week Three Internship Report

This week, I continued strengthening my front-end development skills using Angular. I focused on component interaction, data binding, and refactoring existing code for better readability. I built a Todo List app with features like adding, deleting, and marking tasks as done. Midweek, I faced challenges understanding Angular routing and lazy loading, so I paused briefly to avoid burnout and refreshed my core knowledge. I also explored TypeScript interfaces and began sketching ideas for my Angular-based portfolio site. These efforts gave me a stronger grasp of Angular's architecture and real-world front-end development practices.

Challenges Faced

- Difficulty grasping Angular routing and lazy loading concepts.
- Balancing progress on the Todo List app while managing mental fatigue.

Achievements

- Strengthened knowledge of Angular component interaction and data binding.
- Successfully implemented and refined Todo List app features.
- Improved code structure through refactoring and TypeScript interfaces.
- Designed an initial layout for an Angular-based portfolio project.

2.2.4. Week Four Internship Report

Overview

This week marked an important step in my learning journey, as I began exploring backend development using C# and the .NET framework. It was both challenging and exciting to

transition into a new environment and start understanding how backend systems are built and maintained.

- Introduced to C# programming and the .NET backend framework.
- Practiced writing backend logic and structuring basic .NET applications.
- Explored how APIs are developed and how data is handled on the server side.

Challenges Faced

- Adjusting to the syntax and architectural design of .NET compared to frontend frameworks I had used before.
- Needing additional time to understand API routing and model binding concepts.

Achievements

- Successfully set up and worked with a basic .NET backend environment.
- Strengthened my programming knowledge by applying C# to real backend scenarios.
- Built confidence in understanding how APIs connect the frontend and backend.

2.2.5. Week Five Internship Report

Overview

This week, I focused on backend development for the Cafe Meal Order Management System , implementing authentication and user management features using C# and .NET. The primary goal was to secure the application by integrating JWT-based authentication and role-based authorization.

Challenges Faced

- Token Security: Initially stored JWT in localStorage, which posed XSS risks. Began migrating to HTTP-only cookies.

- Password Reset Flow: Emailed plaintext passwords, which was insecure. Started designing a token-based reset system with expiry.

Achievements

- Built a secure authentication workflow with JWT.
- Enabled admin control over user accounts with new management endpoints.
- Improved backend knowledge of secure API practices.

2.2.6. Week Six Internship Report

Overview

This week, I shifted my focus to the Angular frontend of the Cafe Meal Order Management System. My main tasks involved integrating the authentication features from the backend, enhancing user experience, and improving application security.

Activities & Tasks

- Built reactive forms for login and signup with robust validation using `ReactiveFormsModule`.
- Integrated JWT token handling with `HttpInterceptor`.
- Created an admin dashboard to approve or disapprove users dynamically.
- Added route guards (`AuthGuard`, `AdminGuard`) to secure access to restricted routes.
- Began implementing HTTP-only cookie storage for JWT tokens (ongoing).

Challenges Faced

- Needed to carefully manage role-based routing and route guards.
- Ensuring JWT storage security while transitioning from `localStorage` to HTTP-only cookies.

Achievements

- Successfully integrated backend authentication with the Angular frontend.
- Built an admin dashboard with user approval/disapproval functionality.
- Strengthened security by introducing route guards and beginning migration to cookie-based storage.

2.2.7. Week Seven Internship Report

Overview

This week, I actively participated in the development of a real company project — the Transportation Management System (TMS) Dashboard for the training center. My responsibilities included working on both the frontend and backend components. The dashboard serves as a core tool that provides essential visual insights and operational metrics to manage vehicles, zones, and organizational data effectively. This contribution directly supports the company's transport coordination and decision-making processes.

Frontend Development Tasks

- Dashboard Layout Design: Designed a responsive and user-friendly UI layout using Angular. Created modular UI components to display summary stats (e.g., Permanent Vehicles, Temporary Vehicles, Total Organizations).
- Chart Integration: Developed dynamic bar charts to display vehicles by month and donut charts to visualize vehicle distribution by zone. Implemented dynamic color schemes and labels for better user experience.
- User Interface Enhancements: Added personalized greetings, breadcrumb navigation, and ensured RTL (Right-to-Left) compatibility. Applied Angular directives (*ngIf, bindings) for real-time, data-driven displays.

Backend Development Tasks

- **Data Fetching & Integration:** Integrated backend services to fetch vehicle counter statistics, monthly issuance data, and vehicle distribution by zone. Ensured secure data retrieval using the logged-in user's credentials.
- **API Consumption:** Connected Angular UI with backend APIs using HTTP services. Handled API responses for counters, reports, and zonal data.
- **Error Handling & Optimization:** Implemented basic error logging and toast notifications for user feedback. Optimized data mapping from backend responses into chart-friendly structures.

Achievements

- Delivered a fully functional and dynamic dashboard interface with real-time vehicle and organizational data.
- Ensured smooth integration between frontend components and backend services.
- Built a visually informative system with easy-to-understand analytics for transport oversight.

2.2.8. Week Eight Internship Report

Overview

This week, I focused on analyzing existing transportation workflows within the Oromiya Transport Management System. My primary responsibility was to study how driver and vehicle data is currently managed and propose improvements through structured migration modules. This work was critical in laying the foundation for efficient integration of driver data across the system.

Backend Development Tasks

- **System Analysis:** Conducted a detailed review of the current driver data handling processes and identified inconsistencies and gaps in the migration workflow.
- **Data Flow Design:** Drafted a data flow diagram linking drivers, vehicles, and routes, and suggested standardization methods to streamline data integration.

Achievements

- Gained a deeper understanding of transportation processes and the challenges in managing driver and vehicle data.
- Prepared a comprehensive blueprint for the seamless migration and integration of driver and vehicle records.

2.2.9. Week Nine Internship Report

Overview

This week, I focused on designing and implementing the database schema for the Transport Management System (TMS). I also began developing the core models required for driver and vehicle data migration, laying the groundwork for smooth integration across the system.

2.2.10. Week Ten Internship Report

Overview

This week, I focused on implementing the **Driver Migration Module** for the Transport Management System. My primary task was to develop APIs to fetch driver data with filtering options, ensuring accurate and consistent data handling.

API Consumption & Testing

- Verified API outputs with dummy frontend requests.
- Ensured response mapping matched the correct DTO formats.

2.2.11. Week Eleven Internship Report

Overview

This week, I focused on implementing **route assignment features** for the Oromiya Transport Management System (OTMS) and resolving recurring issues with model references. These tasks were essential for ensuring accurate driver–vehicle–route associations and reducing system errors.

Backend Development Tasks

- **Route Assignment Module:** The route assignment module was implemented to link drivers to vehicles and specific routes. The system was designed to prevent conflicting assignments, ensuring that a single driver could not be assigned to multiple routes at the same time.
- **Error Handling & Debugging:** Driver Migration Data not found errors were resolved by correcting namespace imports. In addition, exception handling was enhanced with clear and informative error logs to assist in faster debugging and system reliability.
- **Achievements:** The driver–route assignment functionality was successfully integrated into the OTMS. System stability was improved, and runtime errors were reduced through consistent namespace management and proper error handling.

2.2.12. Week Twelve Internship Report

Overview

This week, I focused on scheduling integration and improving system efficiency in allocating drivers to routes. The aim was to ensure fair and balanced distribution of workloads while maintaining accurate migration data.

Backend Development Tasks

- **Scheduling Logic:** Algorithms were developed to assign drivers based on their availability and workload. This ensured fairness in the scheduling process and provided a balanced distribution of routes among drivers.

- **Migration Response Update:** The migration response message was extended to include scheduling results, providing clearer and more detailed feedback to the system and its users.
- **Testing:** Unit tests were conducted to verify the accuracy of the scheduling logic. Additionally, multiple migration scenarios were simulated to confirm the reliability and consistency of the scheduling process under different conditions.

2.2.13. Week Thirteen Internship Report

Overview

This week, I focused on system security, authentication, and documentation for the migration modules. The goal was to protect sensitive migration data and provide clear technical guidance for future developers.

Backend Development Tasks

- **Authentication & Authorization:** Custom authentication filters were added specifically for migration APIs to ensure secure access. These filters guaranteed that only authorized users could retrieve or interact with sensitive migration data.
- **Documentation:** Detailed technical notes were prepared to describe the driver migration processes in depth. The documentation also emphasized scalability considerations, ensuring that the system could handle large volumes of data effectively.
- **Security Testing:** Comprehensive testing was carried out on endpoint access under different user roles, such as administrators and regular users. Unauthorized access attempts were logged and monitored to strengthen the system's security posture.

Achievements

Migration endpoints were successfully secured through role-based access, providing an additional layer of protection. Furthermore, clear and comprehensive documentation was produced to guide future developers in maintaining and extending the migration system.

2.2.14. Week Fourteen Internship Report

Overview

This week focused on the finalization and testing of the Driver Migration Module. I worked on ensuring data accuracy, improving system performance, and preparing deployment guidelines to make the module production-ready.

Backend Development Tasks

- **Finalization:** The integration of driver, vehicle, and route modules was completed successfully. Data migration integrity was thoroughly verified through multiple iterations to ensure consistency and accuracy across the system.
- **System-Wide Testing:** End-to-end testing was conducted, covering the driver migration, route assignment, and scheduling modules. During this process, final bugs such as mismatched DTO mappings were identified and resolved, resulting in a more stable system.
- **Deployment Preparation:** Deployment notes and guidelines were drafted to assist in smooth rollout and maintenance. In addition, proposals were made for future expansion, including the development of passenger ticketing and analytics reporting modules to enhance system functionality.

2.3. Section I have been working

During my internship at DAFTech IT Solutions PLC, I was actively involved in the development and enhancement of the Oromia Transport Management System (OTMS), a collaborative project with the Oromia Transport Authority. My role as a Software Development Intern allowed me to contribute significantly across multiple critical areas of the system.

Key Contributions and Responsibilities

Trainer ID Card Update and Barcode Implementation

- I worked on updating trainer ID cards and integrating barcode functionality.
- This task ensured data accuracy, improved identification processes, and enhanced overall system efficiency.

Training Center and Driver Data Migration Validation

- I participated in validating data migrations for training centers and driver information.
- This process required detailed verification to maintain database integrity and ensure correct data mapping during migration.

Training Center Dashboard Development

- I contributed to the design and development of the training center dashboard.
- My work focused on improving the user interface, enhancing functionality, and ensuring a seamless experience for end-users.

Driver Detail Updates and Additions

- I was responsible for updating and adding driver details, covering both heavy-duty drivers and international drivers.
- This involved careful data management and implementation of diverse data handling requirements to maintain comprehensive and accurate records.

Duration of Involvement

My involvement spanned the full course of the internship, with initial tasks focused on data validation and updates, followed by dashboard development and implementation of functional improvements.

3. Experience and Transferable Skills

3.1. Work Experience

3.1.1. Technical Experience

During my internship I significantly improved my technical knowledge and practical skills in full-stack development:

- **Frontend Development:** Gained hands-on experience with Angular and TypeScript, focusing on component-based design, routing, two-way data binding, lazy loading, and state management. I also improved my ability to design responsive UI and dashboards.
- **Backend Development:** Worked with ASP.NET Core to implement APIs, handle authentication, and manage complex workflows. I learned how to apply model binding, dependency injection, and routing effectively in backend systems.
- **Database Management:** Strengthened my skills in MSSQL, including designing schemas, performing data migrations, and validating data integrity.
- **Version Control & Collaboration:** Actively used GitHub for version control, contributing to collaborative development and learning how to handle branching, pull requests, and conflict resolution.

- Security Practices: Applied secure coding practices by addressing JWT token storage vulnerabilities and implementing secure password reset flows using token-based mechanisms.

3.1.2. Non-Technical Experience

In addition to technical growth, I developed important soft skills that are highly valuable in professional environments:

- Teamwork: Collaborated closely with senior developers, interns, and supervisors, learning how to divide tasks and contribute effectively to a group project.
- Communication: Improved both written and verbal communication, particularly in documenting requirements, presenting weekly progress, and discussing technical challenges.
- Time Management: Learned to prioritize tasks effectively, balance competing deadlines, and ensure steady progress on assigned modules.
- Problem-Solving: Faced and overcame technical issues systematically, applying logical reasoning and research to find solutions.

3.2. Benefits of Experience

The internship provided a bridge between theoretical knowledge from university courses and practical, real-world applications. I was able to:

- Apply software engineering principles in actual development, such as requirement gathering, system design, development
- Strengthen my coding discipline by adhering to best practices in version control, debugging, and documentation.
- Improve my adaptability to new technologies, especially when shifting between frontend (Angular/TypeScript) and backend (.NET/MSSQL) environments.
- Gain exposure to working in a professional software development company setting, preparing me for future career opportunities.

3.3. Skills Gained (Soft and Hard Skills)

- **Hard Skills:** Angular, TypeScript, ASP.NET Core, MSSQL, GitHub, REST API design, database migration, authentication & authorization, dashboard development.
- **Soft Skills:** Team collaboration, effective communication, critical thinking, leadership in task ownership, and time management.
- **Knowledge Upgrade:** Gained deeper understanding of secure software design, enterprise-level system workflows, and real-world software lifecycle management.

3.4. Challenges Faced and Solutions Taken

Challenges Faced

During my internship, I encountered several technical and personal challenges that pushed me to grow as a developer. On the backend, I struggled with security concerns because JWT tokens were initially stored in local Storage, creating vulnerabilities to XSS attacks, and the password reset flow relied on sending plaintext passwords. Adapting to ASP.NET Core also proved difficult since its architecture, especially API routing, model binding, and debugging, was unfamiliar compared to frontend frameworks I had used before. On the frontend, Angular and Type Script presented complexities such as lazy loading, routing, and state management across components, while ensuring responsive design added further difficulty. Setting up the development environment delayed my progress due to version conflicts with Angular CLI, Node.js, and npm. In addition to technical hurdles, I experienced mental fatigue from balancing my assigned work on the Oromia Transport Management System with personal projects. Specific project-related challenges also emerged, including validating large-scale data migrations for training centers and drivers, which required accuracy and consistency, and developing the training center dashboard, which demanded a balance between usability and functional requirements.

Solutions Taken

To overcome these challenges, I applied both technical improvements and personal strategies. I enhanced backend security by migrating JWT storage from local Storage to HTTP-only cookies and redesigned the password reset flow into a secure token-based system with expiry times. To adapt to ASP.NET Core, I studied official documentation, worked on small practice projects, and sought guidance from senior developers, which helped me build confidence. On the frontend, I approached Angular challenges by breaking down complex tasks into smaller subtasks, leveraging online resources, and collaborating with teammates. I resolved environment setup issues by upgrading Node.js and npm to stable versions and documented the steps for future interns. For UI responsiveness and state management, I applied Angular services along with responsive CSS frameworks to create a consistent user experience. To reduce mental fatigue, I improved my time management by setting realistic goals, prioritizing key tasks, and incorporating short breaks. For project-specific tasks, I validated data migration using SQL scripts and manual sampling to maintain data integrity, and I built the dashboard through an iterative process of prototyping, testing, and refining based on supervisor feedback.

4. General Overview of the project

4.1. Executive Summary

The Cafe Meal Order Management System aims to digitize and simplify daily cafe operations: order management, cashier operations, and managerial reporting. The current implementation supports a simulated online ordering flow, cashier-assisted ordering for walk-in customers, and a manager dashboard that displays sales statistics and revenue by product and category for daily, weekly, and monthly time ranges. The system generates receipts (PDFs) and maintains order statuses to keep customers and staff informed.

4.2. Background of the Project

The Cafe Meal Order Management System (CMOMS) is a web-based solution designed to modernize and automate the core operations of a cafe. In many small to medium-sized cafes, activities such as taking customer orders, generating receipts, managing inventory, and tracking daily sales are still performed manually. While functional, these manual processes are often slow, error-prone, and inefficient, especially during peak service hours, affecting both staff productivity and customer satisfaction.

To address these challenges, CMOMS provides a centralized digital platform accessible through web browsers, allowing staff to seamlessly manage orders, categorize menu items, and generate

receipts automatically. The system operates in real time, enabling instant order tracking, accurate sales summaries, and the generation of daily, weekly, or monthly reports. These reports offer valuable insights for cafe managers and owners, helping them monitor performance, analyze sales trends, and make informed, data-driven decisions.

Beyond order and billing automation, CMOMS enhances inventory management by tracking product usage, alerting staff to low stock levels, and supporting optimal reordering processes. This prevents stock-outs, minimizes wastage, and ensures popular menu items are consistently available. By implementing this web-based system, cafes gain a scalable, reliable, and user-friendly platform that reduces human errors, saves time, and provides faster, more accurate service to customers. Ultimately, CMOMS improves operational efficiency, strengthens managerial decision-making, and enhances the overall customer experience, laying the foundation for a modern, technology-driven cafe environment.

4.3. Statement of the Problem

In many cafes, daily operations such as taking customer orders, generating receipts, and tracking sales are still handled manually. These manual processes are prone to human errors, especially during peak business hours, leading to misplaced orders, incorrect receipting, and slower service. Dependence on a larger number of wait staff increases labor costs and makes it challenging to maintain consistent service quality, which directly affects customer satisfaction.

From an administrative perspective, manually recording and analyzing sales and inventory data is time-consuming and often inaccurate. Without a centralized system, managers cannot easily access real-time information about daily, weekly, or monthly sales, product performance, or stock levels. This lack of timely and reliable data hinders effective decision-making, such as identifying popular items, planning restocks, or designing promotions, limiting the overall operational efficiency of the cafe.

This system addresses these challenges by providing a digital, centralized platform that automates order management, receipting, inventory tracking, and reporting. By reducing human errors, supporting multiple payment types, and generating accurate real-time reports, CMOMS enables

faster service, optimizes staff utilization, and empowers management with actionable insights. Ultimately, the system enhances operational efficiency, improves customer experience, and provides a scalable solution for modern cafe management.

4.4. Project Significance

The system aims to deliver an integrated and automated solution for handling the daily operations of a café. It is designed to assist staff members such as cashiers and managers by simplifying tasks like order management, receipt generation, inventory monitoring, and report preparation.

4.5. Objective the project

4.5.1. General Objective

To develop a digital Cafe Meal Order Management System that automates order processing, receipting, and sales tracking, while providing accurate and actionable reports to help managers make informed decisions and improve the overall operational efficiency of the cafe.

4.5.2. Specific Objectives

The specific objectives of the System are to:

- Enable user registration and login for Managers, Chefs, and Cashiers, ensuring secure and role-based access to the system.
- Automate order management by allowing staff to record, update, and track customer orders efficiently and in real time.
- Enable cashiers to record orders and payments for walk-in customers.
- Streamline receipting processes by automatically generating accurate receipts for each order, reducing errors and saving service time.

- Digitize sales and inventory tracking to monitor daily, weekly, and monthly revenue, product usage, and stock levels.
- Generate timely and actionable reports for managers, including sales trends, inventory status, and product performance, supporting informed decision-making.

4.6. Scope of the Project

In-Scope

The system focuses on automating and streamlining core cafe operations. The key features included in the current iteration are:

- User authentication and role-based access for Chefs, Cashiers and Managers ensuring secure and appropriate system privileges.
- Change password and forgot password functionality for all users, providing secure options to update or recover account credentials.
- Product and category management with full CRUD (Create, Read, Update, Delete) functionality for menu items and categories.
- Order management, allowing staff to create new orders, update order status, and view detailed order information in real time.
- Dashboard reporting, providing aggregated sales and product performance data on a daily, weekly, and monthly basis, categorized by product and category.
- PDF receipt generation using I Text or similar tools, enabling staff to produce accurate customer receipts automatically.

Out-of-Scope

Certain features are not included in the current version of CMOMS but are planned for future updates:

- Real payment gateway integration (e.g., Stripe, PayPal) for processing online payments.
- Inventory re-order automation, which will enable automatic stock replenishment based on usage and thresholds.
- Mobile native application, as the current system is web-first; however, a responsive UI is recommended for mobile access.

4.7. Functional and nonfunctional requirement

4.7.1. Functional requirement

User Registration & Authentication

- Managers and Admins can create, manage, and deactivate employee accounts.
- Employees can log in securely using their credentials.
- Role-based access ensures appropriate permissions for Admins, Managers, Cashiers, and Chefs.
- Admins have the ability to grant or revoke employee access.
- Users can change passwords and recover accounts through a forgot password feature.

Order Management

- Admins and Cashiers can create, edit, and delete customer orders.
- Orders can be assigned to specific employees for processing (e.g., a cashier handling the order).
- The system tracks and displays the order status in real time, including: Placed, Accepted, Preparing, Ready, Served/Completed, and Cancelled.

Cashier Point of Sale

- Cashiers can create orders for walk-in customers.
- Apply discounts and record payments via cash, card, or simulated online methods.

- Generate and print PDF receipts for each order.

Manager Dashboard & Reporting

- View sales metrics: sold items by category, number of orders, revenue per product.
- Provide filters for daily, weekly, and monthly reports.
- Export reports

Product & Category Management

- Admins can create, update, and delete products and categories.
- Upload product images and set availability.

4.7.2. Non-Functional Requirements

- **Performance:** The system must support multiple simultaneous users without noticeable lag, ensuring smooth operation during peak periods. All pages and report generation processes must complete within 3 seconds under normal conditions.
- **Security:** User credentials must be securely stored using hashed passwords (e.g., bcrypt) and authenticated via JWT or secure cookies. The system must enforce role-based access control for Admins, Managers, Cashiers, and Chefs, and all sensitive data, including sales, orders, and employee information, must be protected against unauthorized access.
- **Maintainability:** The system must have a clear and organized project structure with modular backend services, allowing for easy updates, bug fixes, and feature additions.
- **Scalability:** The system must be designed to scale horizontally, with the ability to add web server instances and read-replicas for reporting as needed to handle increased load.
- **Availability:** The system must be available 24/7 for internal users during working hours. Backup and recovery mechanisms must ensure that data loss is minimized in case of system failures.

- **Usability:** The system must have an intuitive, user-friendly interface, enabling staff to perform tasks efficiently with minimal training. Clear feedback and notifications must be provided for all key actions, including order placement, payment processing, and report generation.
- **Portability:** The system must be web-based and compatible with all major browsers (Chrome, Firefox, Edge). The user interface must be responsive, allowing access from desktops, tablets, and mobile devices.

4.8. Methodology

Agile methodology allows iterative development, frequent feedback, and continuous improvement. This is crucial for the CMOMS, where requirements may evolve based on real user feedback.

Agile Approach in This Project

- Requirements were gathered and prioritized based on cafe operations.
- Development was done in sprints, with each sprint focusing on a specific module (e.g., order management, receipting, reporting).
- Continuous testing and feedback ensured the system met user expectations.

4.8.1. Development Tools

The development of the System leverages a modern and robust technology stack, ensuring scalability, maintainability, and security. The tools and frameworks used include:

Frontend Development:

- **Angular with TypeScript:** is used for building the user interface, ensuring a dynamic, responsive, and user-friendly experience.


- Bootstrap CSS is employed for styling, enabling rapid design prototyping and maintaining a clean, modern UI.
- UI Libraries: Angular Material, custom CSS

Backend Development:

- C# with ASP.NET Core Web API is used to implement the backend REST APIs, ensuring high performance and scalability.

Database

- Microsoft SQL Server: Serves as the relational database for structured storage of user, vehicle, services, and transactions data.

 **IDE/Editors:** Visual Studio 2022 (backend), Visual Studio Code (frontend)

 **Version Control:** Git with GitHub

 **Testing Tools:** Postman (API)

4.9. System Analysis and Modeling

4.9.1. Use Case Diagram

The Use Case Model describes the functional interactions between system actors (users) and the Cafe Meal Order Management System. It defines how different roles (Customer, Cashier, Manager, and Admin) engage with the system to achieve specific goals.

Objectives of Use Case Modeling

- To capture the functional requirements of Cafe Meal Order Management System.
- To provide a clear understanding of user interactions with the system.
- To serve as a foundation for further modeling (e.g., class, sequence, and activity diagrams).

Actors

- Admin/Manager: Responsible for high-level oversight, managing employees, services, financial records, viewing reports, and overall system configuration.
- Cashier: Creates orders for walk-in customers, processes payments, and distributes orders to the Chef.

- Chef : Views assigned orders, updates food preparation progress, and marks orders as ready.

Figure 1 Use case diagram

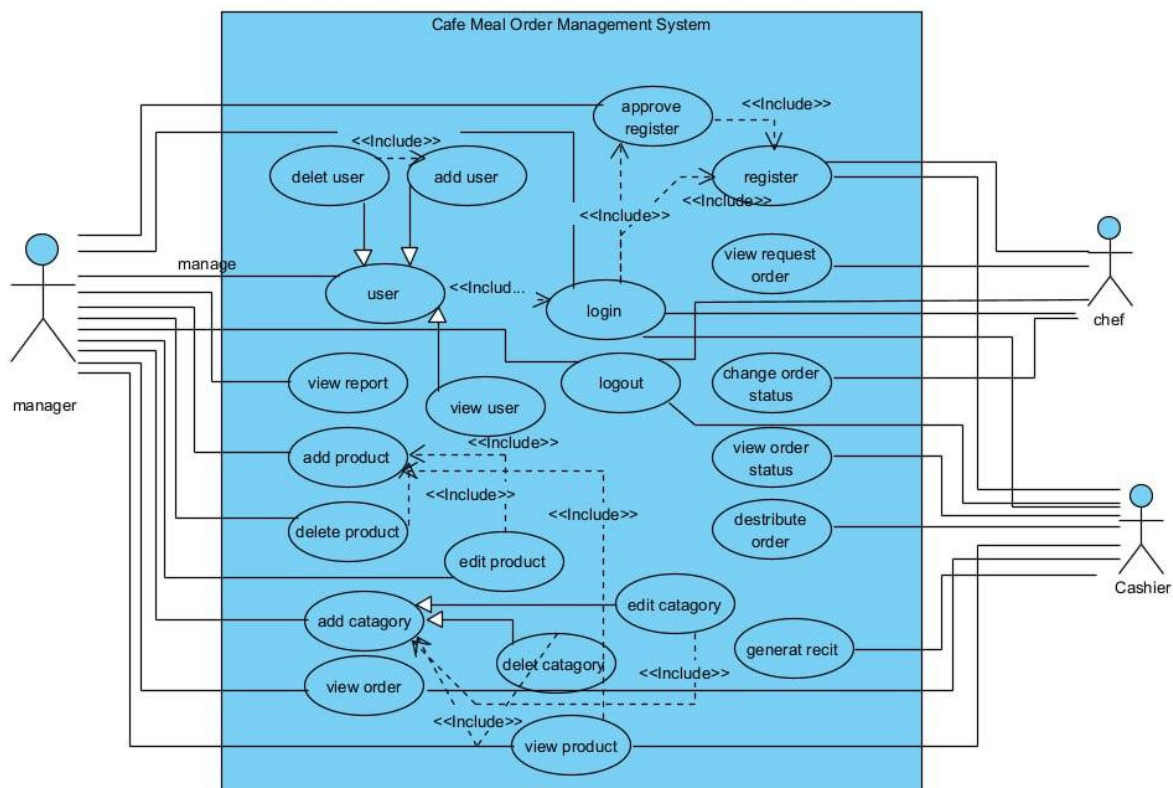


Table 1 Assign IDs for Use Cases

No	UC-id	UC-name	UC description
1	UC-01	Register	A new user (customer/cashier/chef/manager) creates an account in the system.
2	UC-02	Login	A user provides valid credentials to access the system features.
3	UC-03	Logout	A user logs out of the system to end the session.

4	UC-04	Manage User	Manager adds a new user, View User, Edit user and delete user account manually.
5	UC-05	Approve Register	The manager approves a new user's registration request.
6	UC-06	Manage Product	The manager adds a new product, edits, Delete, View existing product to the menu.
7	UC-07	Manage Category	The manager adds a new product category, edits, Delete and View an existing category.
8	UC-08	View Order Status	A customer or cashier checks the status of an order.
9	UC-09	Change Order Status	The cashier updates the order status (e.g., paid, pending).
10	UC-10	View Request Order	The chef views incoming order requests.
11	UC-11	Distribute Order	The chef marks the order as distributed/served.
12	UC-12	Generate Receipt	The cashier generates a receipt after payment.
13	UC-13	View Report	The manager views sales or performance reports.

Table 2 UC-01: Register

Field	Description
UC-Name	Register
Actor	Customer, Cashier, Chef, Manager
Description	A new user (customer/cashier/chef/manager) creates an account in the system.
Entry condition	User is not logged in.
Post conditions	A new user account is created and stored in the system.
Flow of event	<ol style="list-style-type: none"> 1. User selects "Register". 2. System displays registration form. 3. User fills in details and submits the form. 4. System validates the data.

5. System creates the new account.
6. System confirms successful registration.

Table 3 UC-02: Login

Field	Description
UC-Name	Login
Actor	Customer, Cashier, Chef, Manager
Description	A user provides valid credentials to access the system features.
Entry condition	User has an existing account.
Post conditions	User is authenticated and redirected to their dashboard.
Flow of event	<ol style="list-style-type: none"> 1. User selects "Login". 2. System displays login form. 3. User enters credentials. 4. System validates the credentials. 5. If valid, system grants access and redirects to the dashboard.

UC-03: Logout

Field	Description
UC-Name	Logout

Actor	Customer, Cashier, Chef, Manager
Description	A user logs out of the system to end the session.
Entry condition	User is logged in.
Post conditions	User session is terminated.
Flow of event	<ol style="list-style-type: none"> 1. User clicks "Logout". 2. System ends the session and returns to the login/home page.

Table 4 UC-04: Manage User

Field	Description
UC-Name	Manage User
Actor	Manager
Description	Manager adds a new user, views user details, edits user info, or deletes a user account manually.
Entry condition	Manager is logged in.
Post conditions	User records are updated in the system.
Flow of event	<ol style="list-style-type: none"> 1. Manager selects "Manage User". 2. System displays user management options. 3. Manager chooses add/view/edit/delete. 4. System validates and applies the changes.

Table 5 UC-05: Approve Register

Field	Description
UC-Name	Approve Register
Actor	Manager
Description	The manager approves a new user's registration request.

Entry condition	User has requested registration; manager is logged in.
Post conditions	Registration is approved and the user can access the system.
Flow of event	<ol style="list-style-type: none"> 1. Manager selects "Approve Register". 2. System shows pending requests. 3. Manager approves or rejects request. 4. System updates user status.

Table 6 UC-06: Manage Product

Field	Description
UC-Name	Manage Product
Actor	Manager
Description	Manager adds, edits, deletes, and views products in the menu.
Entry condition	Manager is logged in.
Post conditions	Product information is updated in the system.
Flow of event	<ol style="list-style-type: none"> 1. Manager selects "Manage Product". 2. System shows product list. 3. Manager adds/edits/deletes product. 4. System validates and saves changes.

Table 7 UC-07: Manage Category

Field	Description
UC-Name	Manage Category
Actor	Manager
Description	Manager adds, edits, deletes, and views product categories.
Entry condition	Manager is logged in.
Post conditions	Category information is updated in the system.

Flow of event	<ol style="list-style-type: none"> 1. Manager selects "Manage Category". 2. System displays category list. 3. Manager adds/edits/deletes category. 4. System validates and updates changes.
----------------------	---

Table 8 UC-08: View Order Status

Field	Description
UC-Name	View Order Status
Actor	Customer, Cashier
Description	A customer or cashier checks the status of an order.
Entry condition	Order exists in the system.
Post conditions	Order status is displayed.
Flow of event	<ol style="list-style-type: none"> 1. Actor selects "View Order Status". 2. System prompts for order details. 3. System retrieves and displays status.

Table 9 UC-09: Change Order Status

Field	Description
UC-Name	Change Order Status
Actor	Cashier
Description	The cashier updates the order status (e.g., paid, pending).
Entry condition	Cashier is logged in and order exists.
Post conditions	Updated order status is saved.
Flow of event	<ol style="list-style-type: none"> 1. Cashier selects "Change Order Status". 2. System displays order. 3. Cashier updates status. 4. System validates and saves changes.

Table 10 UC-10: View Request Order

Field	Description
UC-Name	View Request Order
Actor	Chef
Description	The chef views incoming order requests.
Entry condition	Chef is logged in; pending orders exist.
Post conditions	Chef can see list of new orders.
Flow of event	1. Chef selects "View Request Order". 2. System displays list of new orders.

Table 11 UC-11: Distribute Order

Field	Description
UC-Name	Distribute Order
Actor	Chef
Description	The chef marks the order as distributed/served.
Entry condition	Chef is logged in and order is prepared.
Post conditions	Order marked as distributed in the system.
Flow of event	1. Chef selects order. 2. Chef marks it as distributed. 3. System updates order status.

Table 12 UC-12: Generate Receipt

Field	Description
UC-Name	Generate Receipt
Actor	Cashier
Description	The cashier generates a receipt after payment.
Entry condition	Payment is completed.
Post conditions	Receipt generated and saved.

Flow of event	<ol style="list-style-type: none"> 1. Cashier selects order. 2. Cashier confirms payment. 3. System generates receipt. 4. System displays/prints receipt.
----------------------	---

Table 13 UC-13: View Report

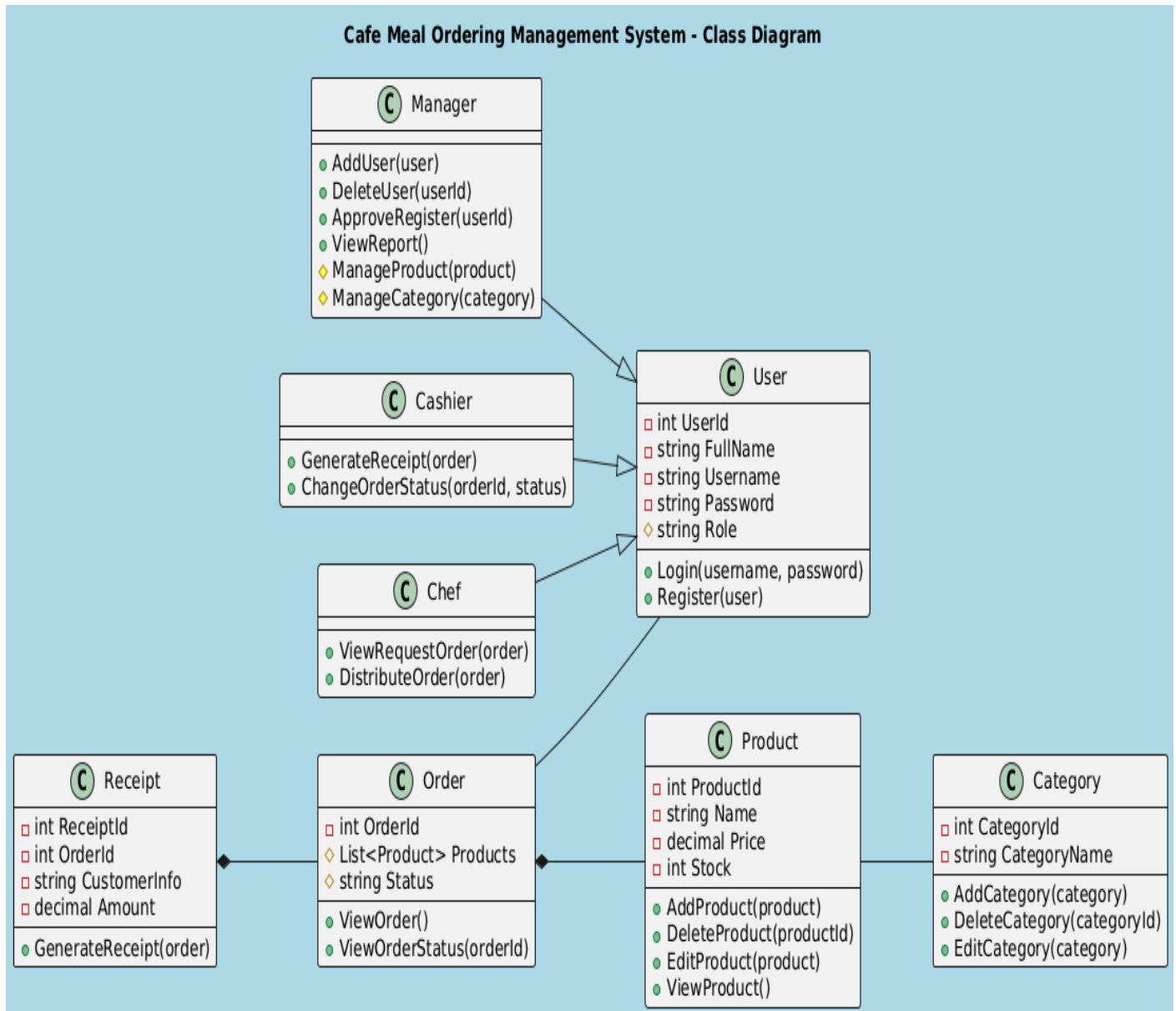
Field	Description
UC-Name	View Report
Actor	Manager
Description	The manager views sales or performance reports.
Entry condition	Manager is logged in; report data exists.
Post conditions	Reports are displayed.
Flow of event	<ol style="list-style-type: none"> 1. Manager selects "View Report". 2. System retrieves sales/performance data. 3. System displays report.

4.9.2. Class Diagram

The Class Diagram represents the structural design of the CMOMS system. It shows classes, their attributes, operations, and the relationships between them.

- Core Classes: User, Manager, Order, Cashier, Chef, Product, Category, Order, Receipt and Report
- Relationships: Manager, Cashier, Chef, and inherit from User, Associations and Multiplicity

Figure 2 Class Diagram



4.9.3. Sequence Diagrams

The Sequence Diagrams describe the interaction between objects in a time sequence for each use case.

Figure 3 user registration Sequence

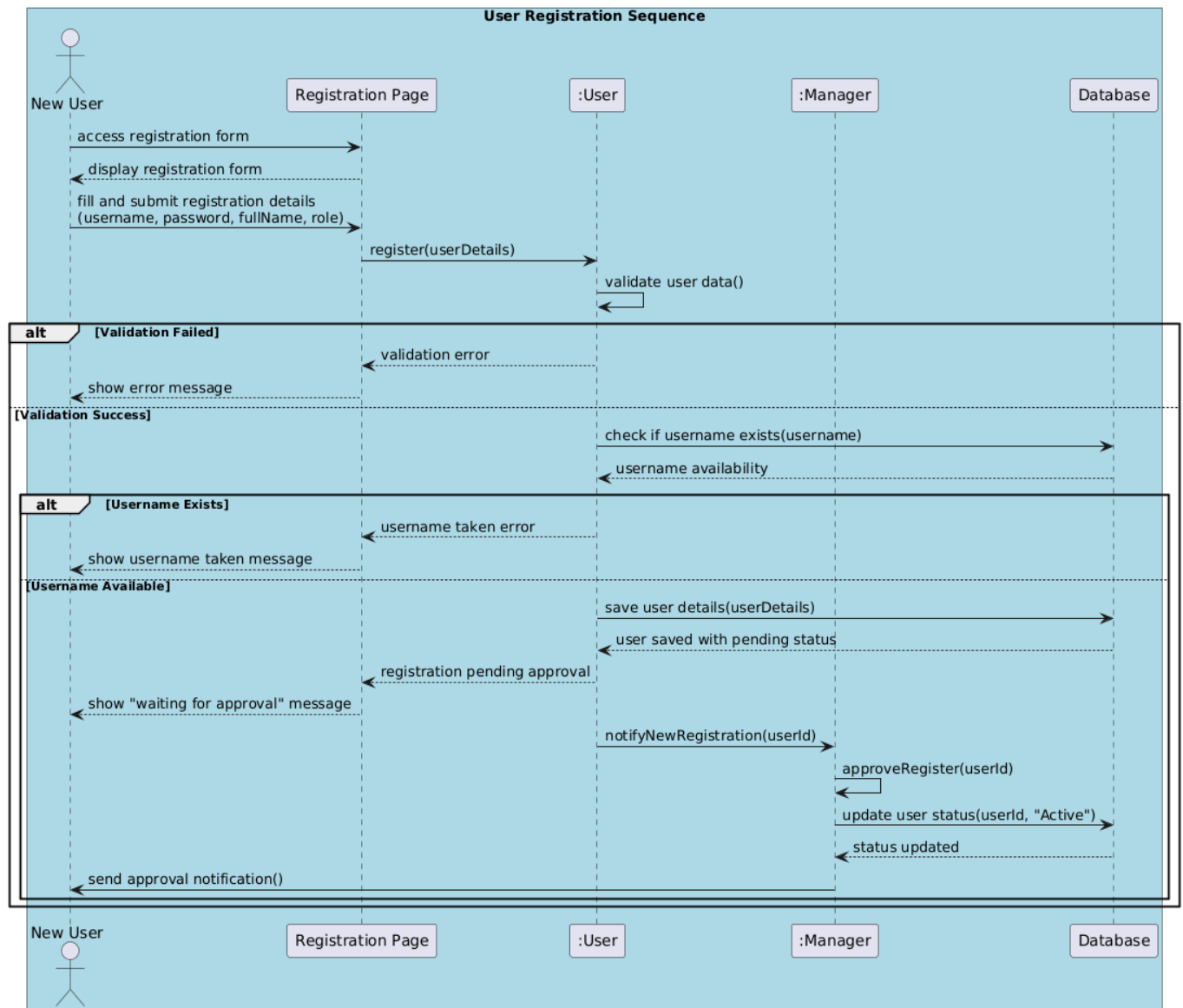


Figure 4 User login Sequence

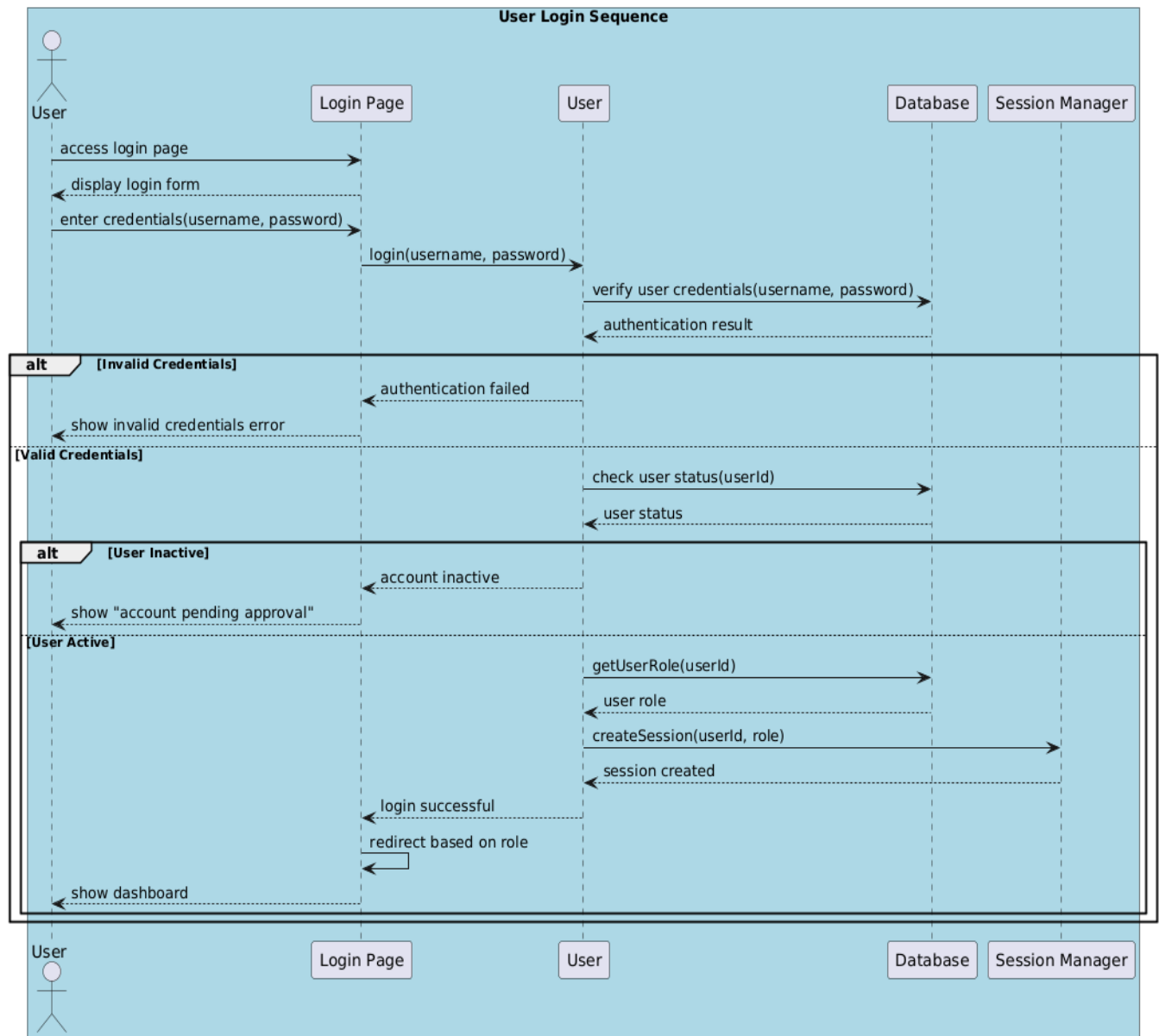


Figure 5 Add new user Sequence

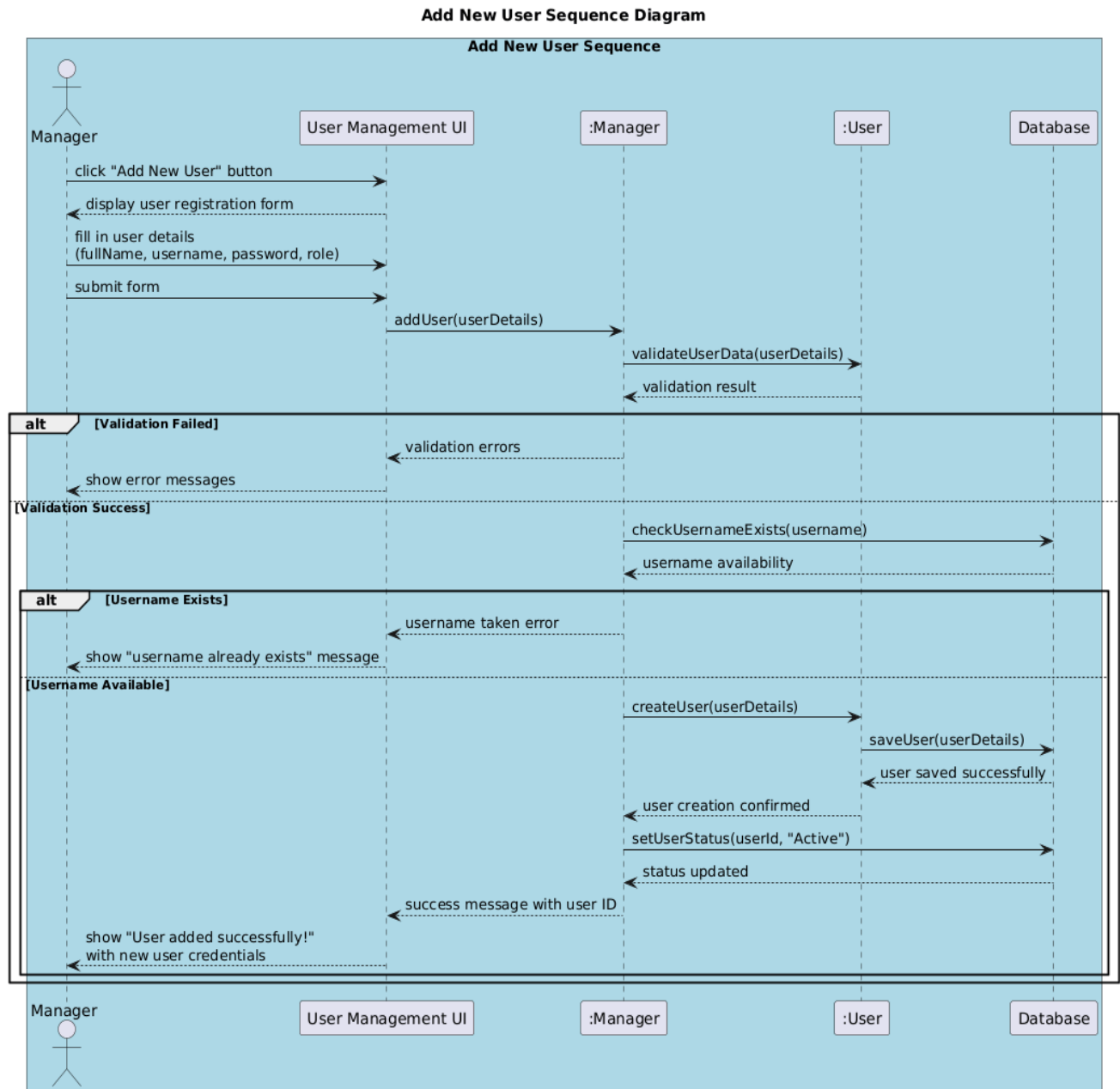


Figure 6 Edit user Sequence

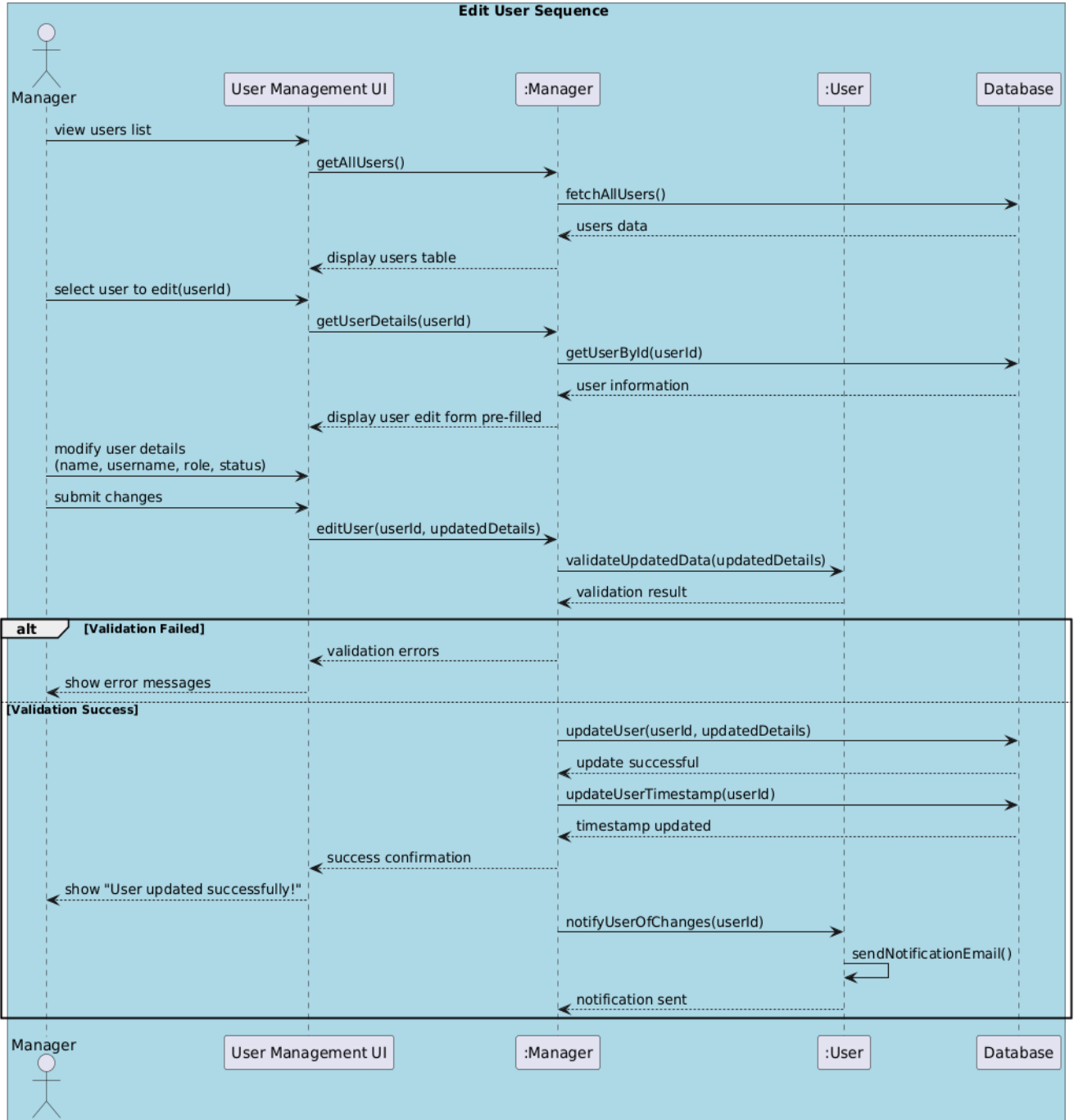


Figure 7 Delete user Sequence

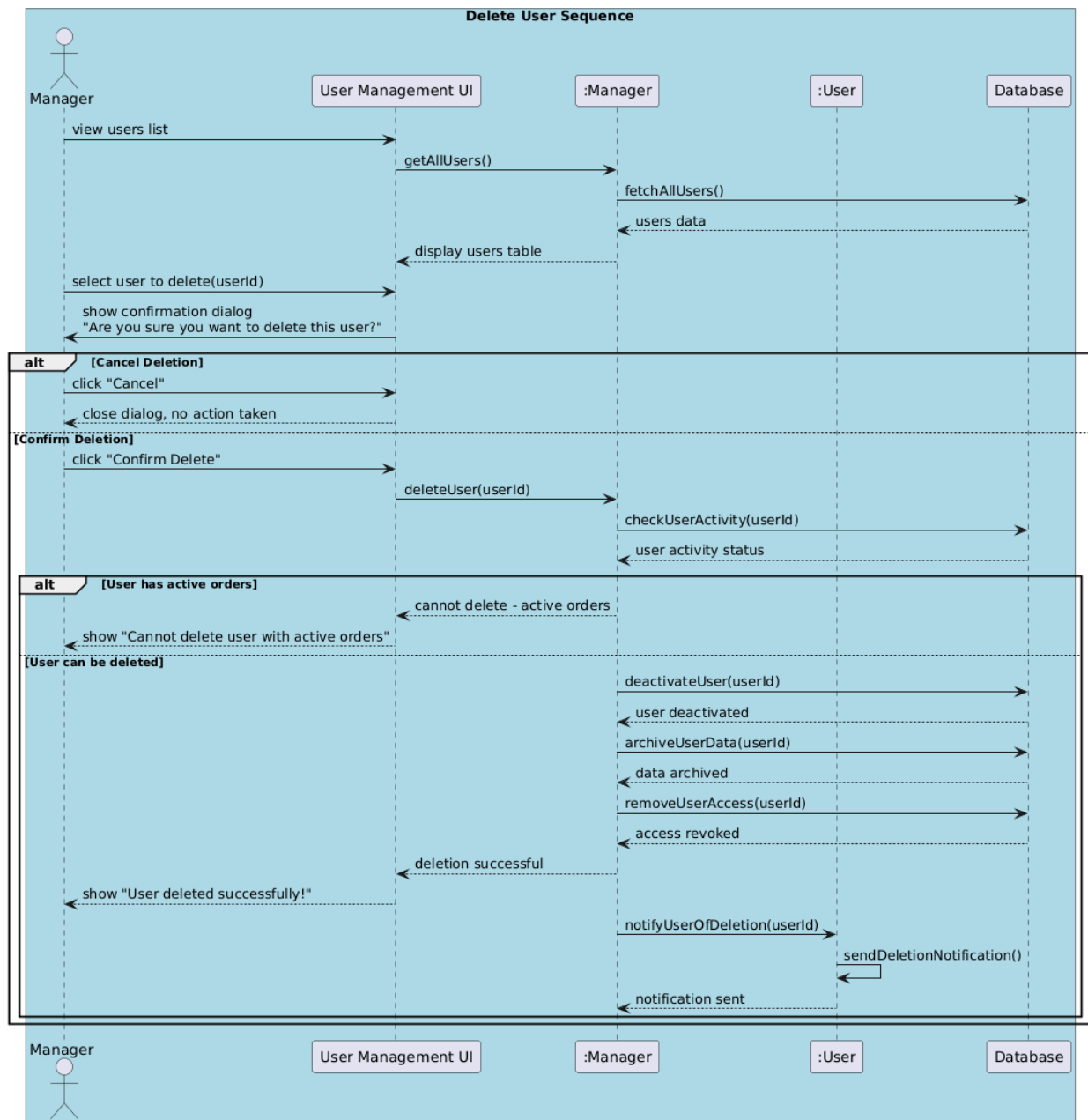
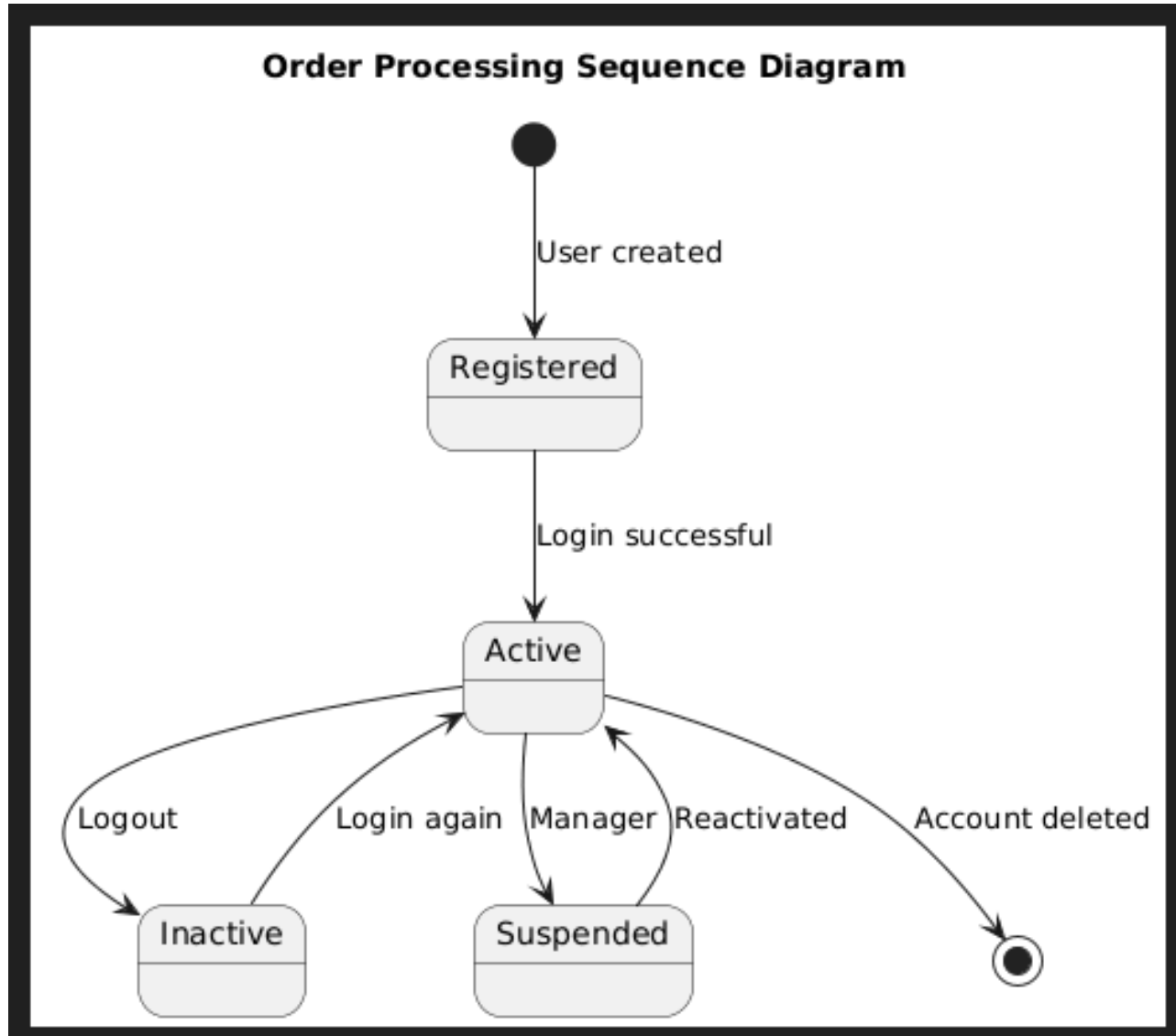


Figure 8 Order processing Sequence



4.9.4. State Chart Diagram

The State Chart Diagram illustrates the lifecycle of major entities in the system

Figure 9 Registration state chart diagram

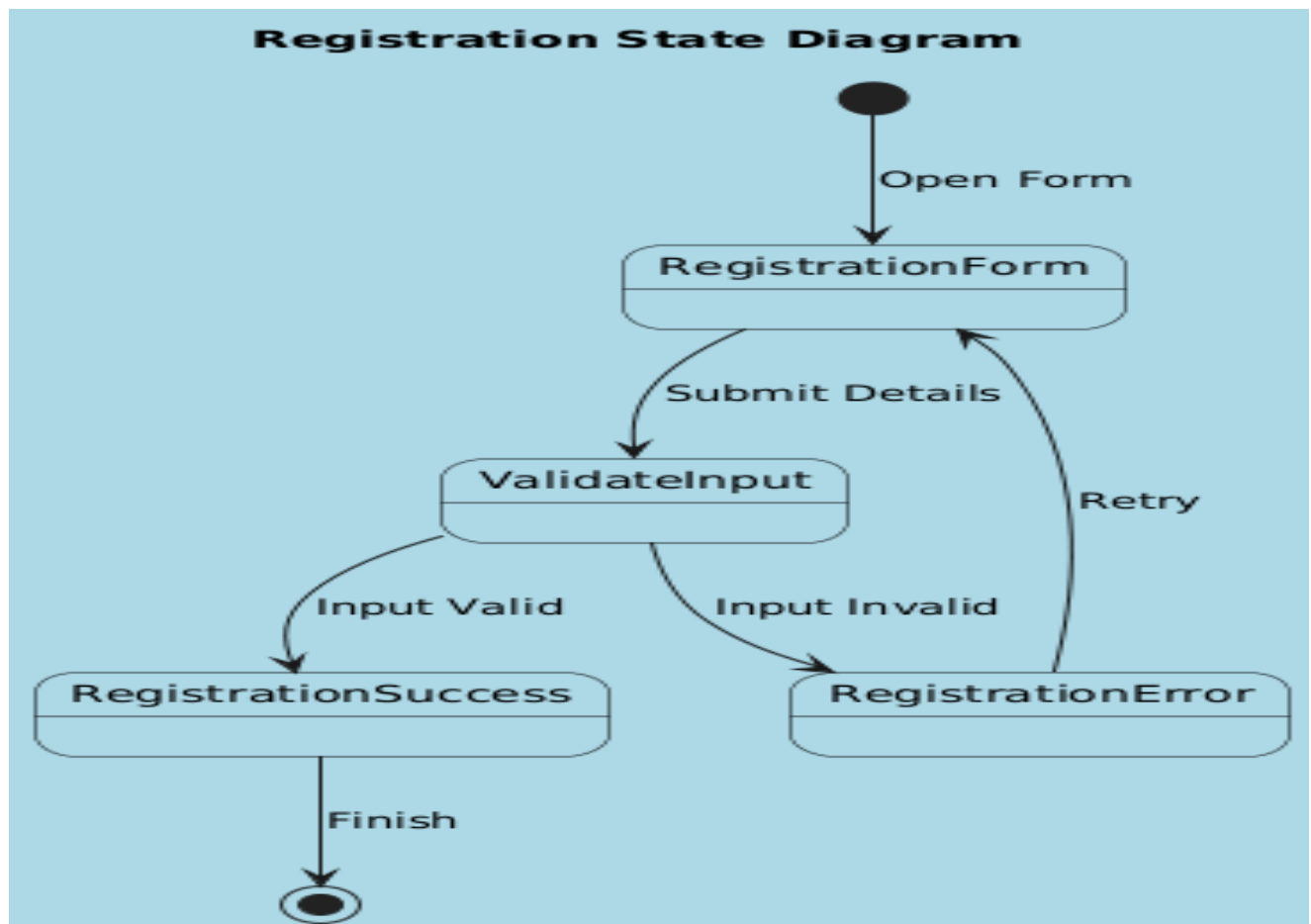


Figure 10 Add user state chart diagram

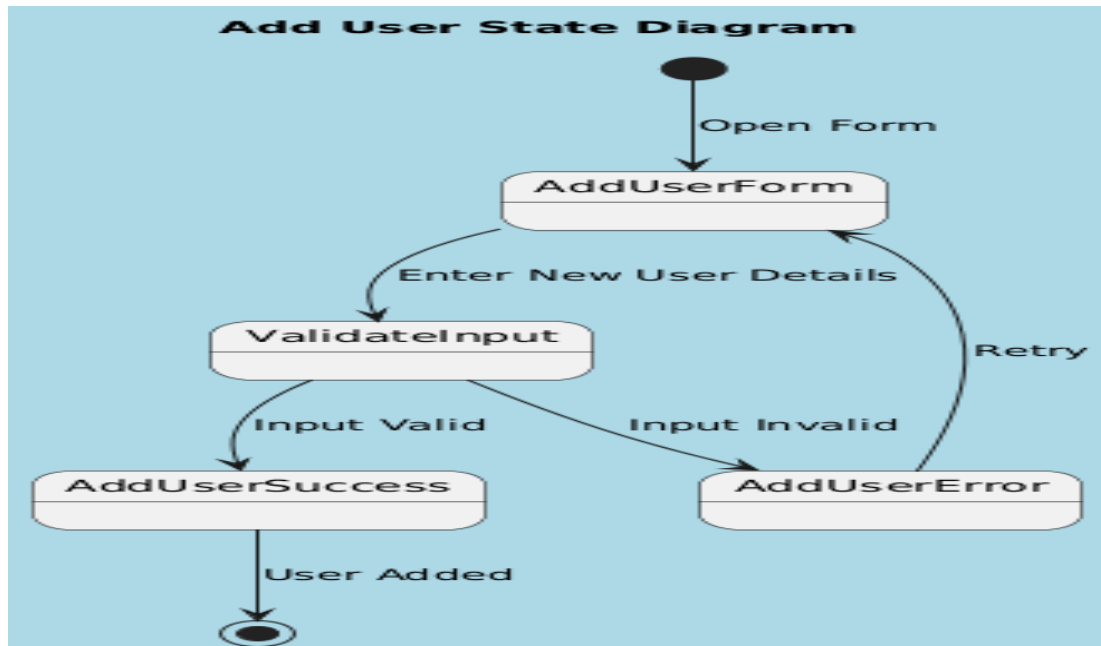


Figure 11 Add product state chart diagram

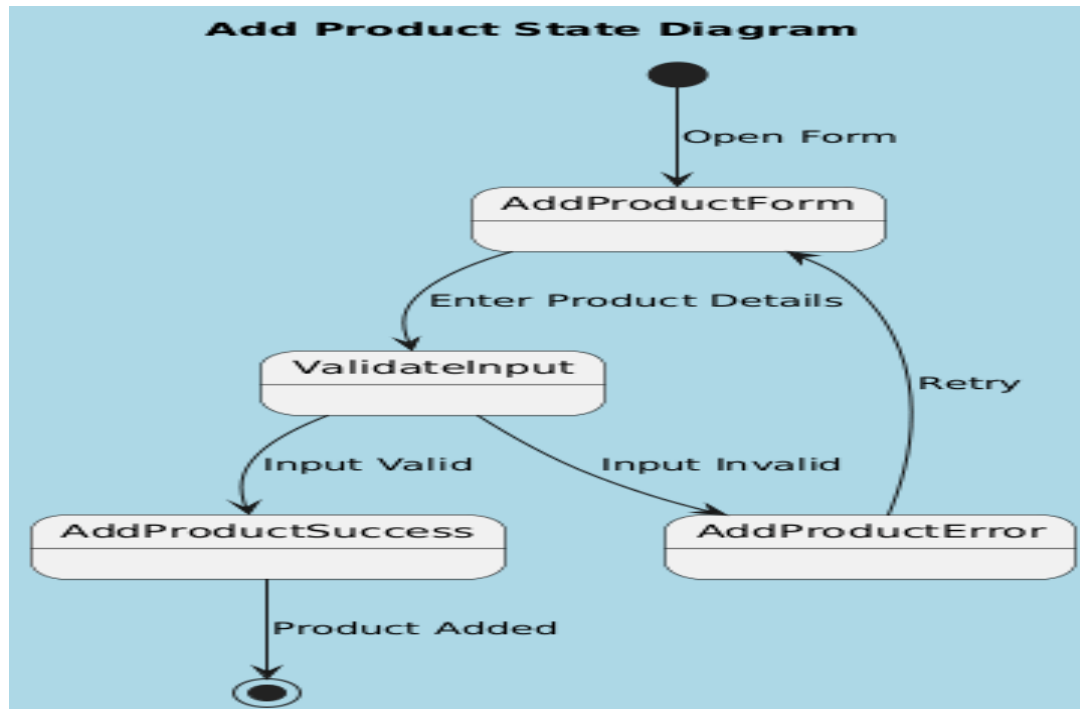
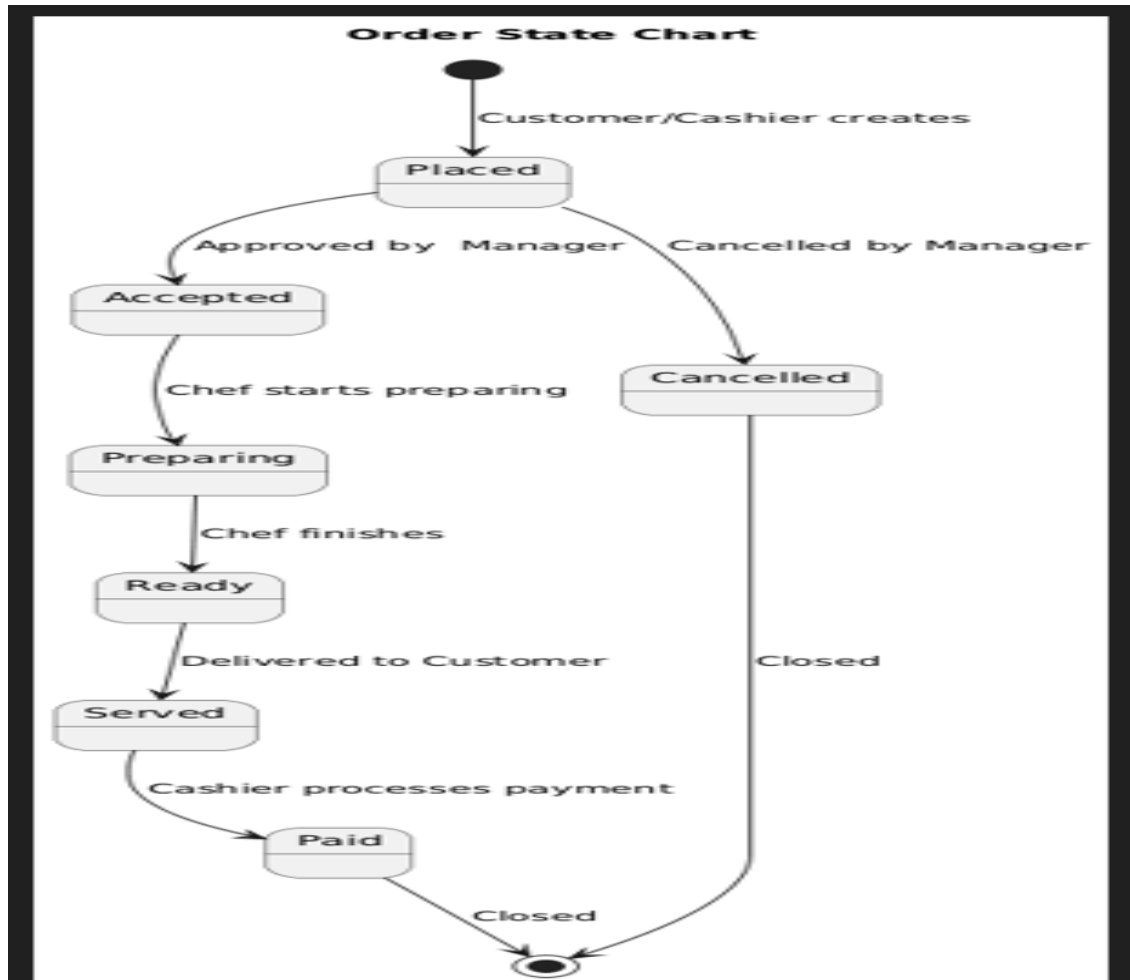


Figure 12 Order state chart diagram



4.9.5. Activity Diagrams

The Activity Diagrams illustrate workflows for major use cases, showing the flow of activities from start to end.

Figure 13 Registration activity diagram



Figure 14 Login activity diagram

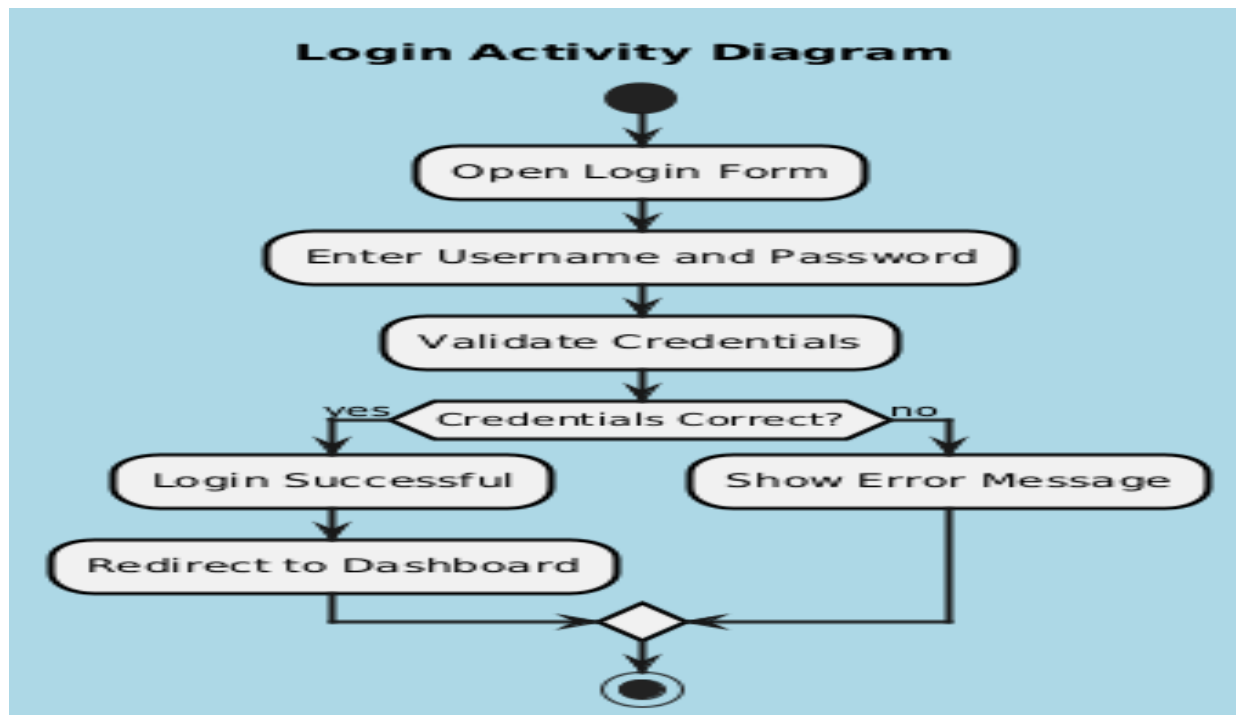


Figure 15 Add user activity diagram

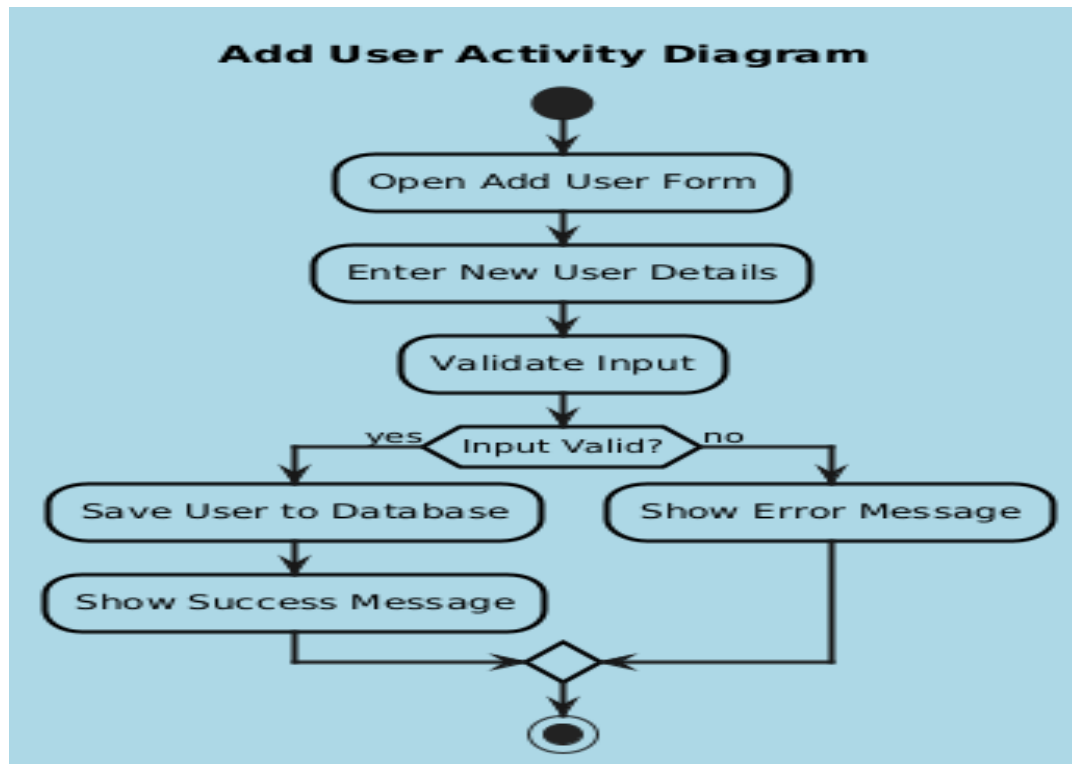
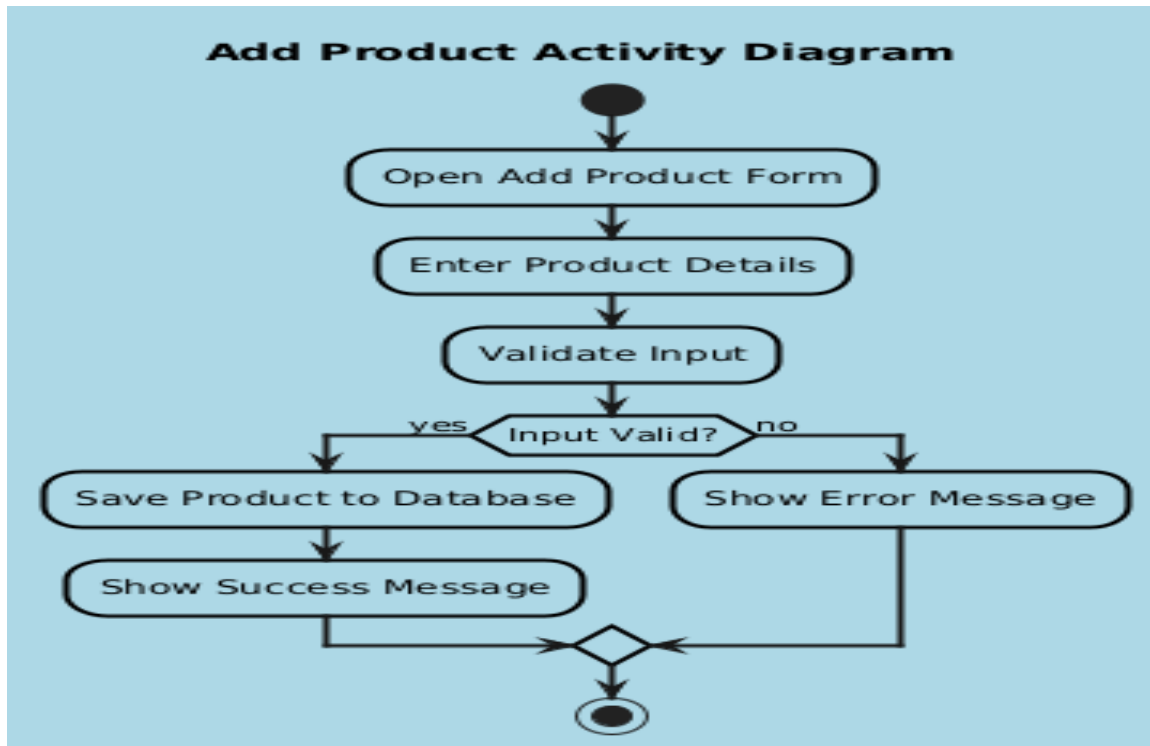


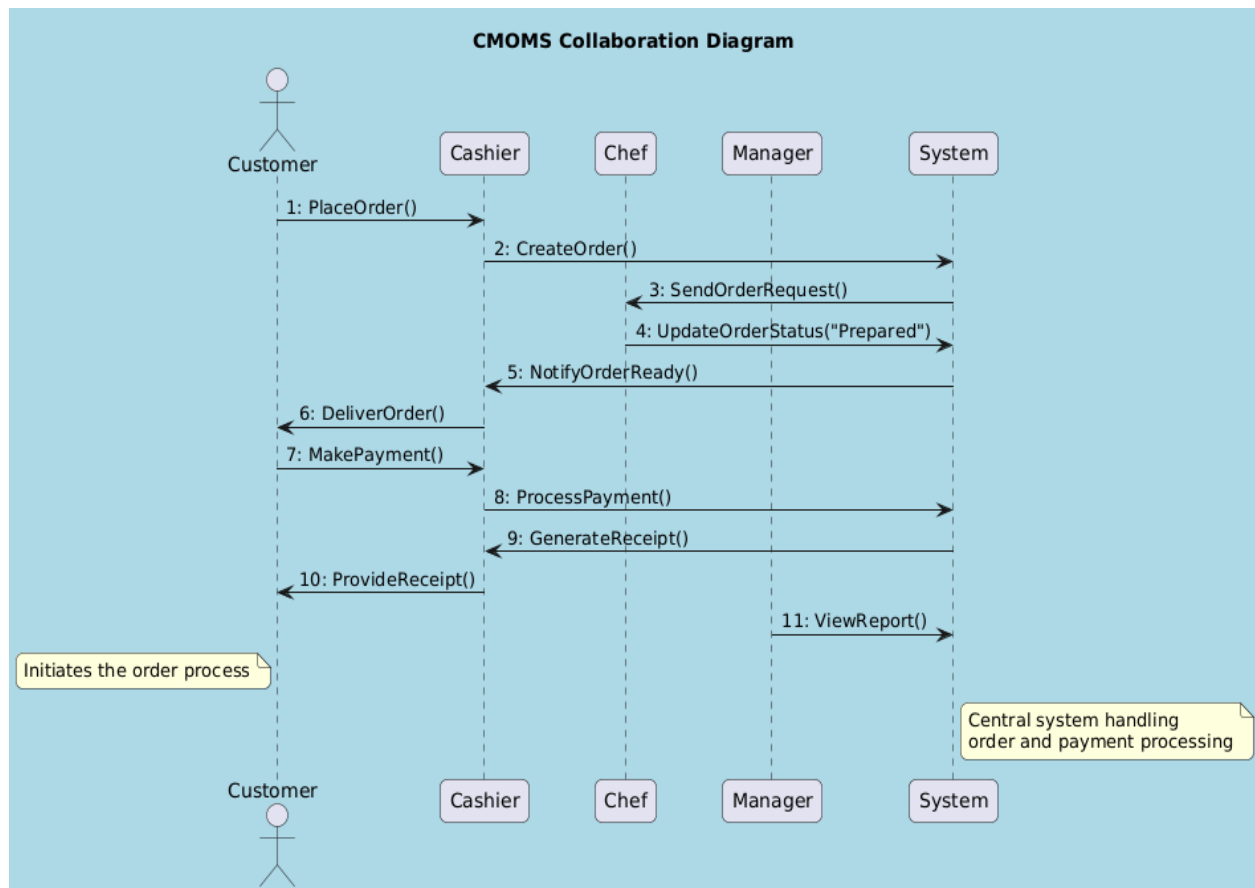
Figure 16 Add product activity diagram



4.9.6. Collaboration Diagrams

- Interaction between cashier, system, and database during order processing

Figure 17 Collaboration Diagrams



4.9.7. Database Design

The database design of the System provides the structural foundation for storing, managing, and retrieving data efficiently. It ensures that all system processes—such as user management, product and category handling, order processing, billing, and reporting—are supported in a secure, consistent, and scalable way.

🚦 Key objectives of the database design include:

- **Consistency:** Ensuring accurate relationships between customers, cashiers, chefs, managers, and their respective activities.
- **Scalability:** Supporting future growth, such as adding new roles, products, or reporting features without major redesign.

- **Data Integrity:** Using constraints and relationships to maintain correctness (e.g., an order must belong to a valid customer).
- **Security:** Storing sensitive user details (like passwords) securely and restricting unauthorized access.
- **Traceability:** Keeping historical data such as receipts and reports for accountability and auditing.

✚ The system have the following core entities are:

- **Users and Roles** – to handle different system actors (customers, cashiers, chefs, managers).
- **Products and Categories** – to manage menu items and their classifications.
- **Orders** – to record customer purchases and product details.
- **Receipts** – to finalize transactions and provide proof of payment.

Table 14 Users Table

Column	Data Type	Constraints
UserID (PK)	UNIQUEIDENTIFIER	DEFAULT NEWID()
FullName	VARCHAR(100)	NOT NULL
Email	VARCHAR(100)	UNIQUE, NOT NULL
ContactNumber	VARCHAR(20)	NULL
Username	VARCHAR(50)	UNIQUE, NOT NULL
Password	VARCHAR(255)	NOT NULL
RoleID (FK)	INT	FK → Roles(RoleID)
Status	VARCHAR(20)	DEFAULT 'Active'

Table 15 Roles Table

Column	Data Type	Constraints
--------	-----------	-------------

RoleID (PK)	INT	AUTO_INCREMENT
RoleName	VARCHAR(50)	UNIQUE (Customer, Cashier, Chef, Manager)

3. Categories Table

Column	Data Type	Constraints
CategoryID	INT	PK, AUTO_INCREMENT
CategoryName	VARCHAR(100)	UNIQUE
Description	TEXT	NULL

Table 16 Products Table

Column	Data Type	Constraints
ProductID	INT	PK, AUTO_INCREMENT
Name	VARCHAR(100)	NOT NULL
Price	DECIMAL(10,2)	NOT NULL
CategoryID	INT	FK → Categories(CategoryID)

Table 17 Orders Table

Column	Data Type	Constraints
OrderID (PK)	UNIQUEIDENTIFIER	DEFAULT NEWID()
CustomerID	UNIQUEIDENTIFIER	FK → Users(UserID)
CashierID	UNIQUEIDENTIFIER	FK → Users(UserID), NULL
OrderDate	DATETIME	DEFAULT CURRENT_TIMESTAMP
PaymentMethod	VARCHAR(50)	e.g., 'Cash','Card','MobileMoney'
TotalAmount	DECIMAL(10,2)	NOT NULL
Status	VARCHAR(20)	('Pending','Paid','Served','Cancelled')

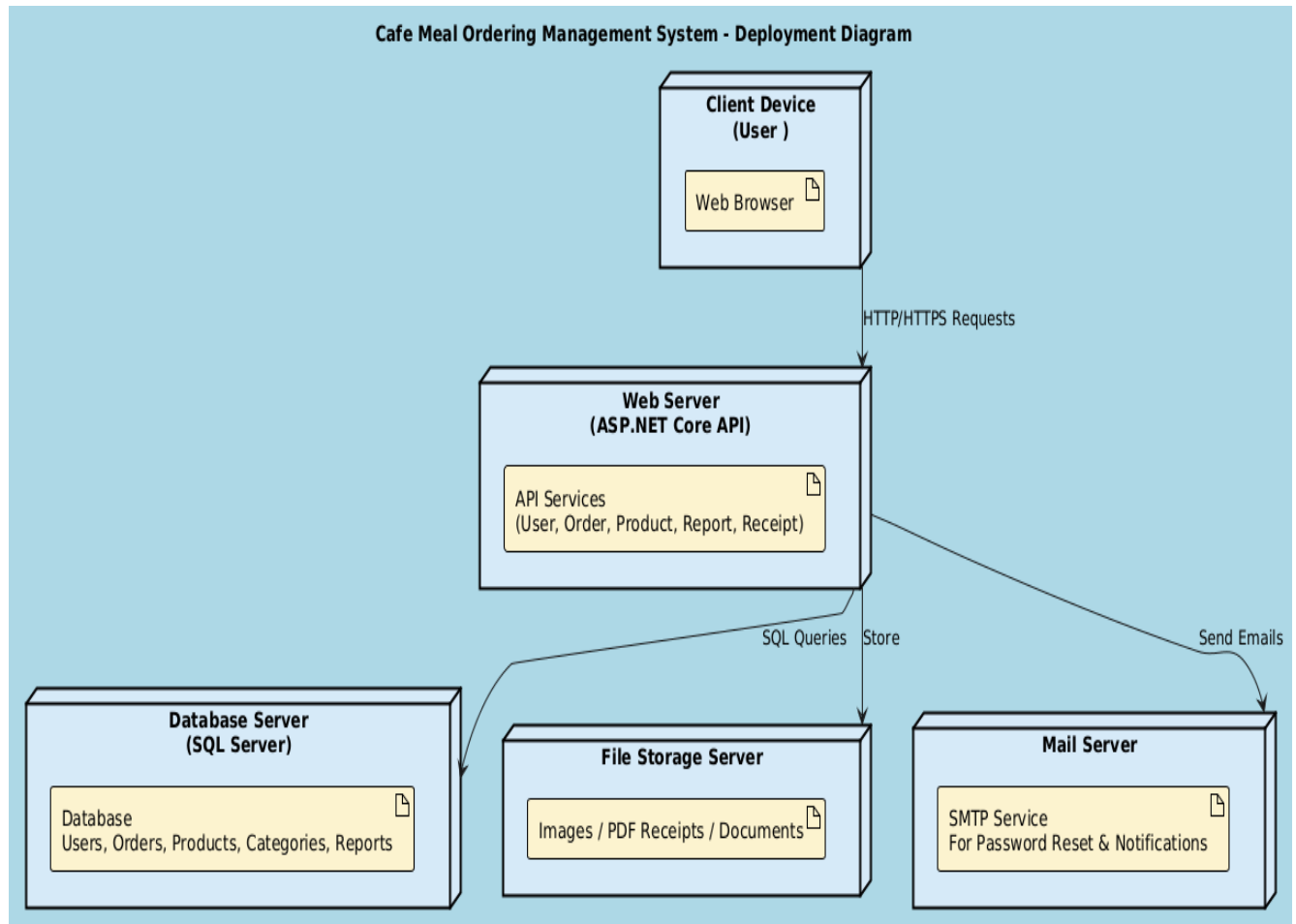
Table 18 Receipts (Bill) Table

Column	Data Type	Constraints
uuid (PK)	UNIQUEIDENTIFIER	DEFAULT NEWID()
OrderID	UNIQUEIDENTIFIER	FK → Orders(OrderID)
name	VARCHAR(100)	Customer/Order name
email	VARCHAR(100)	Customer email
contactNumber	VARCHAR(20)	Customer contact
paymentMethod	VARCHAR(50)	Cash/Card/etc
totalAmount	DECIMAL(10,2)	From Orders.TotalAmount
productDetail	TEXT	JSON or string list of purchased products
createdBy	UNIQUEIDENTIFIER	FK → Users(UserID) (cashier who generated it)
CreatedDate	DATETIME	DEFAULT CURRENT_TIMESTAMP

4.9.8. Deployment Diagram

- **Frontend:** Angular application on staff computers
- **Backend:** ASP.NET Core Web API on server
- **Database:** SQL Server hosted locally or on cloud

Figure 18 Deployment diagram



5. Conclusion and Recommendation

Conclusion

During my internship at **DAFTech**, I gained valuable hands-on experience in developing and managing a comprehensive software system, specifically the Cafe Meal Ordering Management System. This project allowed me to apply my knowledge of software development, database design, and web application frameworks in a real-world setting.

Through this experience, I enhanced my skills in ASP.NET Core Web API, Angular, database management, and UML modeling, while also improving my problem-solving and project management abilities. I am grateful to my supervisor zerabruk Assefa and the organization for their continuous guidance, support, and for providing me with the opportunity to contribute to a meaningful and functional system.

Recommendations

Based on my internship experience, I would like to offer several suggestions that could benefit the organization and future interns, as well as the department and university that supported my journey.

Recommendation for future internship

For future interns, I recommend taking a proactive approach to learning. Interns should try to immerse themselves in the project from the beginning, ask questions, and actively seek feedback from team members. It's also essential to take initiative in understanding the project's goals, even beyond the technical aspects, so that they can contribute more effectively. Regularly communicating with mentors and peers for guidance and support will enhance the internship experience and ensure personal growth.

Recommendation for hosting company

It would be beneficial to provide interns with more opportunities for hands-on learning through mini-projects or coding challenges before assigning them to larger tasks. Regular check-ins and feedback sessions would also help interns gauge their progress and make adjustments where

necessary. Offering a formal mentorship program could also be a valuable addition to the internship experience.

Recommendation for department

The Software Engineering Department at Debre Berhan University could provide more internship-related resources, such as partnerships with local companies or hosting career development workshops. A more structured internship program that bridges the gap between academic knowledge and industry experience would be highly beneficial. This could include organizing internship fairs or offering guidance in preparing for internships, such as resume workshops and mock interviews, to better equip students for the demands of the tech industry.

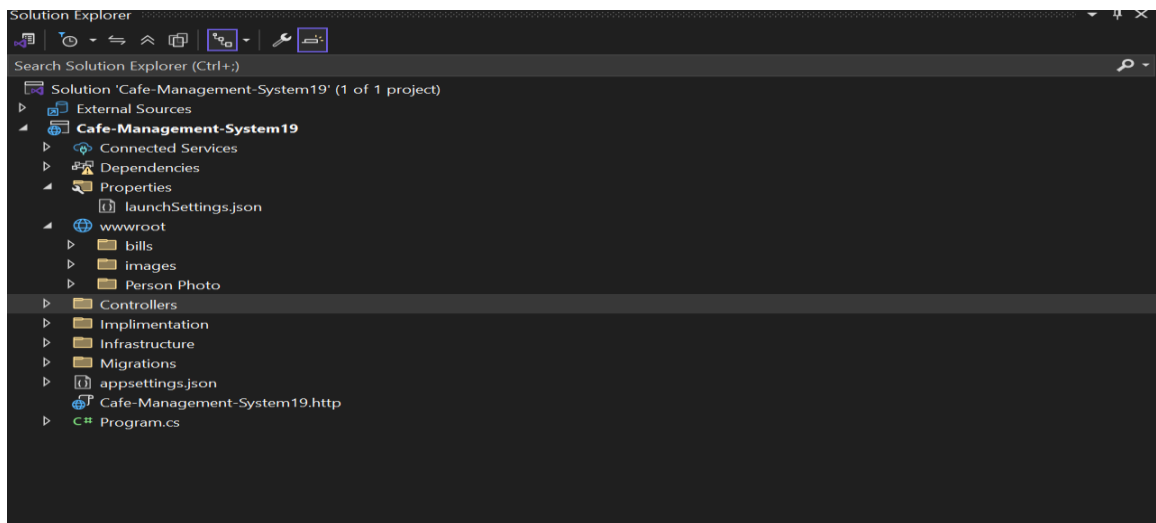
Recommendation for university

I recommend that Debre Berhan University continues to foster relationships with more local and international tech companies to create internship opportunities for students. The university could also work on enhancing the curriculum by integrating more practical, project-based learning experiences that align with industry standards. Providing additional career counseling services would help students navigate their internship search and make informed decisions about their professional paths.

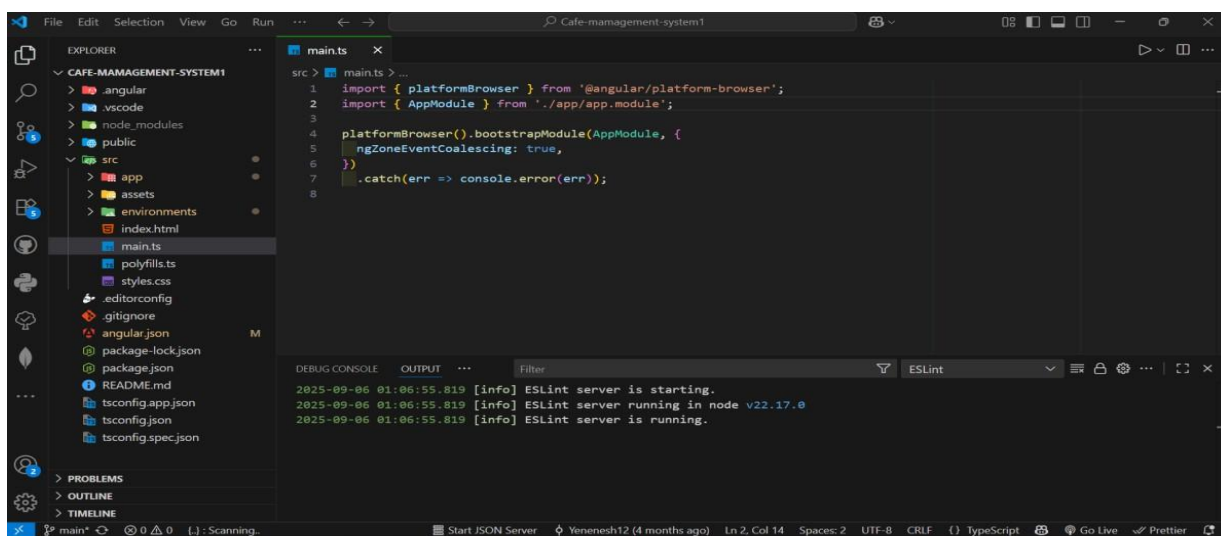
6. Appendix

6.1. Appendix A: Project Folder Structure

This section shows the organization of the CMOMS project folders for both frontend (Angular) and backend (ASP.NET Core Web API). **Backend (ASP.NET Core Web API)**



Frontend (Angular)



6.2. Appendix B: API Source Code

This section provides **key examples** of the backend API implementation.

UserController.cs

```
[Route("api/[controller]/[action]")]
[ApiController]
public class UserController : ControllerBase
{
    private readonly IUserService _userService;

    public UserController(IUserService userService)
    {
        _userService = userService;
    }

    // POST: api/User/SignUp
    [HttpPost]
    public async Task<IActionResult> SignUp([FromForm] UserCreateDto userDto)
    {
        var result = await _userService.CreateUserAsync(userDto);
        return result.Success ? Ok(result) : BadRequest(result);
    }

    // GET: api/User/GetAll
    [HttpGet]
    public async Task<IActionResult> GetAll()
    {
        var users = await _userService.GetUsersAsync();
        return Ok(users);
    }

    // GET: api/User/GetById/{id}
    [HttpGet("{id}")]
    public async Task<IActionResult> GetById(Guid id)
    {
        var user = await _userService.GetUserByIdAsync(id);
        return user != null ? Ok(user) : NotFound(new { Message = "User not found" });
    }

    // PUT: api/User/Update
    [HttpPut]
    public async Task<IActionResult> Update([FromForm] UpdateUserDto userDto)
    {
        var result = await _userService.UpdateUserAsync(userDto);
        return result.Success ? Ok(result) : BadRequest(result);
    }

    // DELETE: api/User/Delete/{id}
    [HttpDelete("{id}")]
    public async Task<IActionResult> Delete(Guid id)
    {
        var result = await _userService.DeleteUserAsync(id);
        return result.Success ? Ok(result) : BadRequest(result);
    }
}
```

```
[HttpPost]
public async Task<IActionResult> ChangePassword([FromForm] ChangePasswordDto dto)
{
    // get userId from claims
    var userIdClaim = User.FindFirst("userId")?.Value;
    if (!Guid.TryParse(userIdClaim, out Guid userId))
        return Unauthorized(new { message = "Invalid token" });

    var res = await _userService.ChangePasswordAsync(userId, dto);
    if (!res.Success) return BadRequest(res);
    return Ok(res);
}
```

ProductController.cs

```
[Route("api/[controller]")]
[ApiController]
public class ProductController : ControllerBase
{
    private readonly IProductService _productService;

    public ProductController(IProductService productService)
    {
        _productService = productService;
    }

    [HttpGet]
    public async Task<IActionResult> GetAll()
    {
        var products = await _productService.GetAllProductsAsync();
        return Ok(products);
    }

    [HttpPost("add")]
    public async Task<IActionResult> Add([FromForm] CreateProductDto createProductDto)
    {
        var result = await _productService.AddProductAsync(createProductDto);
        if (result.Success) return Ok(result);
        return BadRequest(result);
    }

    [HttpPost("update")]
    public async Task<IActionResult> Update([FromForm] UpdateProductDto updateProductDto)
    {
        var result = await _productService.UpdateProductAsync(updateProductDto);
        if (result.Success) return Ok(result);
        return BadRequest(result);
    }

    [HttpDelete("delete/{id}")]
    public async Task<IActionResult> Delete(Guid id)
    {
        var result = await _productService.DeleteProductAsync(id);
        if (result.Success) return Ok(result);
        return BadRequest(result);
    }
}
```

OrderController.cs

```
[Route("api/[controller]/[action]")]
[ApiController]
public class OrderController : ControllerBase
{
    private readonly IOrderService _orderService;

    public OrderController(IOrderService orderService)
    {
        _orderService = orderService;
    }

    // Create new order
    [HttpPost("create")]
    public async Task<IActionResult> CreateOrder([FromForm] OrderDTO orderDto)
    {
        // Await the async method to get the actual ResponseMessage
        var result = await _orderService.CreateOrder(orderDto);

        // Now you can access result.Success
        return result.Success ? Ok(result) : BadRequest(result);
    }

    // Get order by ID
    [HttpGet("{id}")]
    public async Task<IActionResult> GetOrder(Guid id)
    {
        var order = await _orderService.GetOrderById(id);
        if (order == null)
            return NotFound("Order not found.");

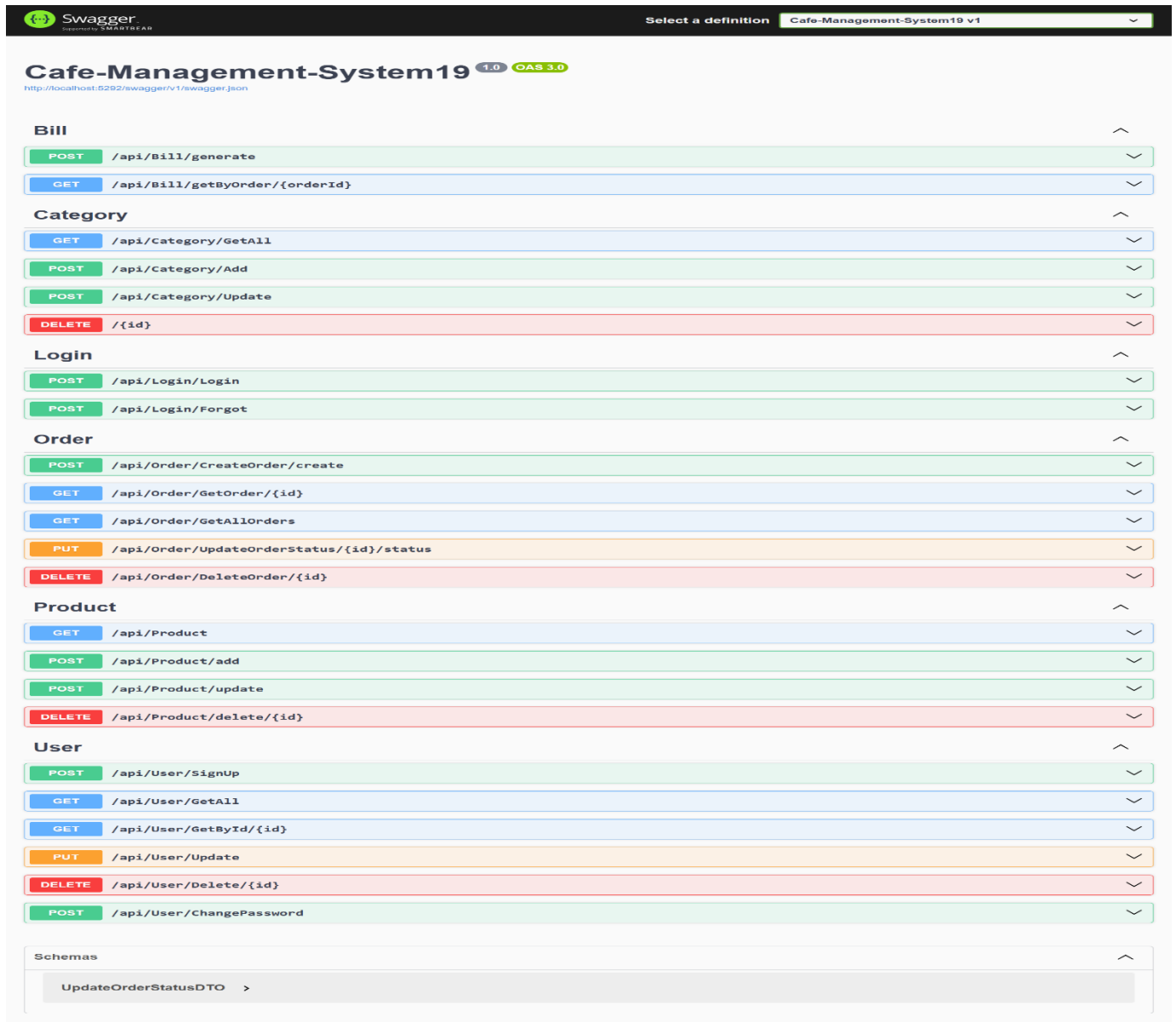
        return Ok(order);
    }

    [HttpGet]
    public async Task<IActionResult> GetAllOrders()
    {
        var orders = await _orderService.GetAllOrders();
        return Ok(orders);
    }

    // Update order status
    [HttpPut("{id}/status")]
    public async Task<IActionResult> UpdateOrderStatus(Guid id, [FromBody] UpdateOrderStatusDTO dto)
    {
        var result = await _orderService.UpdateOrderStatus(id, dto.Status);
        return result.Success ? Ok(result) : BadRequest(result);
    }

    [HttpDelete("{id}")]
    public async Task<IActionResult> DeleteOrder(Guid id)
    {
        var result = await _orderService.DeleteOrder(id);
        return result.Success ? Ok(result) : BadRequest(result);
    }
}
```

6.3. Appendix C: Source Code Demo



The image shows the Swagger UI for the 'Cafe-Management-System19 v1' API. The interface is organized into sections for different API endpoints, each with a color-coded header and a list of endpoints with their HTTP methods and paths. The endpoints are as follows:

- Bill**
 - POST /api/Bill/generate
 - GET /api/Bill/getByOrder/{orderId}
- Category**
 - GET /api/Category/GetAll
 - POST /api/Category/Add
 - POST /api/Category/Update
 - DELETE /{id}
- Login**
 - POST /api/Login/Login
 - POST /api/Login/Forgot
- Order**
 - POST /api/Order/CreateOrder/create
 - GET /api/Order/GetOrder/{id}
 - GET /api/Order/GetAllOrders
 - PUT /api/Order/UpdateOrderStatus/{id}/status
 - DELETE /api/Order/DeleteOrder/{id}
- Product**
 - GET /api/Product
 - POST /api/Product/add
 - POST /api/Product/update
 - DELETE /api/Product/delete/{id}
- User**
 - POST /api/User/SignUp
 - GET /api/User/GetAll
 - GET /api/User/GetById/{id}
 - PUT /api/User/Update
 - DELETE /api/User/Delete/{id}
 - POST /api/User/ChangePassword
- Schemas**
 - UpdateOrderStatusDTO

7. Reference

- <https://daftechsocialictsolution.com/>

- *Microsoft Documentation. (2025). ASP.NET Web API Documentation. Retrieved from:* □ <https://github.com/SnapDevelop/.NET-Project-Example-Database>, □ <https://docs.devmagic.com/sd2025/>
- *Angular Documentation. (2025). Angular Framework Guide. Retrieved from:* <https://angular.dev>
- *W3Schools. (2025). MSSQL Database Tutorial. Retrieved from:* <https://www.w3schools.com/sql>
- *https://chatgpt.com/c/*
- *SRS Guidelines: IEEE Standard for Software Requirements Specifications, IEEE 8301998*
- *https://editor.plantuml.com/*
- *https://grok.com/c/20e87e9c-d7f5-485e-8e5f-0043d3469531*