# 

**Development of Auraluxe: A Single Vendor E-commerce Application for Luxury Building Materials**

Md. Moniruzzaman

ID: 22103018

A Practicum in the Partial Fulfillment of the Requirements

for the Award of Bachelor of Computer Science and Engineering (BCSE)



Department of Computer Science and Engineering

College of Engineering and Technology

IUBAT International University of Business Agriculture and Technology

Fall 2025

**Auraluxe: A Single Vendor E-commerce Application for Luxury Building Materials**

Md. Moniruzzaman

A Practicum in the Partial Fulfillment of the Requirements for the Award of Bachelor of Computer Science and Engineering (BCSE)

The practicum has been examined and approved,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Dr. Utpal Kanti Das

Chairman

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Shahinur Alam

Co-supervisor, Coordinator and Assistant Professor

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dipta Mohon Das

Supervisor and Lecturer

Department of Computer Science and Engineering

College of Engineering and Technology

IUBAT International University of Business Agriculture and Technology

Fall 2025

## **Letter of Transmittal**

14 January 2026

The Chair

Practicum Defense Committee

Department of Computer Science and Engineering

IUBAT International University of Business Agriculture and Technology

4 Embankment Drive Road, Sector 10, Uttara Model Town

Dhaka 1230, Bangladesh.

**Subject:** Letter of Transmittal.

Dear Sir,

With due respect, I submit my project report titled **“Neighbourly: A Neighbourhood Service Finder Application”**, completed as part of the practicum requirement for the Bachelor of Computer Science and Engineering (BCSE) degree. The project, conducted under the supervision of FiroTech, focuses on developing a hyperlocal service marketplace that connects community members with nearby service providers using modern software architecture.

This work provided practical exposure to ASP.NET Core, Clean Architecture, and Geolocation and Hugging Face APIs, strengthening my full-stack development and problem-solving skills. I sincerely appreciate your time in reviewing this report and hope it meets the required academic and professional standards.

Yours sincerely,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Abdullah Al Mamun

ID: 22103014

## **Organization’s Certificate**

## **Student’s Declaration**

I hereby declare that the project report entitled **“Neighbourly: A Neighbourhood Service Finder Application”** is my original work and has been prepared by me in partial fulfillment of the requirements for the practicum. All sources of information and references used in this report have been properly acknowledged.

I confirm that the contents of this report are original and that no part of this work has been submitted previously for any degree or academic purpose. I have adhered to ethical guidelines in the collection and presentation of data and take full responsibility for the authenticity of this work.

I sincerely hope that this report meets the standards and expectations of the Practicum Defense Committee and respectfully submit it for review and evaluation.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Abdullah Al Mamun

ID: 22103014

## **Supervisor’s Certification**

This is to certify that the project report entitled **“Neighbourly – A Neighbourhood Service Finder Application”** has been prepared by **Abdullah Al Mamun**, **ID: 22103014**, in partial fulfillment of the requirements for the practicum.

I have reviewed the report and confirm that the work has been carried out under my supervision. All sources, references, and materials used in this report have been properly cited.

I hereby certify that the report meets the academic standards required by the Department of Computer Science and Engineering, IUBAT, and is suitable for submission and evaluation.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tanzina Tasnim Bithi

Supervisor and Assistant Professor

Department of Computer Science and Engineering

IUBAT International University of Business Agriculture and Technology

## **Abstract**

This practicum report presents the design and implementation of "**Neighbourly**," a hyperlocal service marketplace developed to bridge the gap between community members and local service providers. The project was conducted during a software development internship at **FiroTech** to address the inefficiencies, lack of transparency, and trust issues prevalent in traditional service discovery methods. "Neighbourly" utilizes a robust web-based platform built on **ASP.NET Core 8**, adhering to **Clean Architecture** principles to ensure scalability and maintainability. The system integrates **LocationIQ** APIs for precise geolocation services, allowing users to discover providers within specific radii, and leverages the **Hugging Face Inference API** to generate AI-powered professional service descriptions. Key functionalities include a dual-registration system for users and providers, a comprehensive booking lifecycle with secure **SSLCommerz** payment integration, and a rigorous provider verification workflow. The development followed the **Incremental Process Model**, enabling the iterative testing and refinement of core modules. The resulting application successfully digitizes the service marketplace, reducing the time required to find trusted help and providing a structured environment for local economic transactions. However, the current system is limited by its dependency on continuous internet connectivity and the use of a sandbox environment for payments. Future work will focus on developing a native mobile application, implementing offline capabilities, and integrating predictive analytics to further enhance service matching. This project demonstrates the practical application of modern software engineering standards in solving real-world community challenges.

## **Acknowledgments**

I would like to take this opportunity to sincerely thank all those who supported and encouraged me throughout the completion of my practicum program and the preparation of the project report entitled **“Neighbourly: A Neighbourhood Service Finder Application.”** First and foremost, I express my gratitude to the Almighty for granting me the patience, strength, and perseverance necessary to complete this work. I am also thankful to the International University of Business, Agriculture and Technology (IUBAT) for providing an academic environment that fosters both personal and professional growth. I would like to pay special tribute to **Late Prof. Dr. M. Alimullah Miyan**, the Founder of IUBAT, whose visionary leadership opened new avenues for higher education in the non-government sector of Bangladesh. I also extend my sincere appreciation to the honorable Vice-Chancellor, **Prof. Abdur Rab**, for his continued leadership and guidance, which have greatly benefited the institution and its students. My heartfelt thanks go to **Prof. Dr. Utpal Kanti Das**, Professor and Chair, for his valuable advice and encouragement throughout this academic journey. Above all, I am deeply indebted to my supervisor, **Tanzina Tasnim Bithi**, Assistant Professor, Department of Computer Science and Engineering, IUBAT, for his constant guidance, constructive feedback, and encouragement, which were instrumental in shaping this work. Completing this practicum has enhanced my understanding of the subject matter, strengthened my technical skills, and provided valuable experience in research and report writing. I am truly grateful for the opportunity to work on this project, which has been both academically enriching and professionally rewarding.

**Table of Contents**

**Letter of Transmittal** iii

**Organization’s Certificate** iv

**Student’s Declaration** v

**Supervisor’s Certification** vi

**Abstract** vii

**Acknowledgments** viii

**List of Figures** xiv

**List of Tables** xvii

**Chapter 1.** **Introduction** 1

1.1 Background of the Study 2

1.2 Methodology 3

1.2.1 Primary Sources. 3

1.2.2 Secondary Sources. 4

1.3 Objectives 5

1.3.1 Broad Objective. 5

1.3.2 Specific Objectives. 5

1.4 Process Model 6

1.4.1 Reasons for Choosing the Incremental Process Model. 6

1.5 Feasibility Study 7

1.5.1 Technical Feasibility. 7

1.5.2 Economic Feasibility. 7

1.5.3 Operational Feasibility. 8

1.6 Structure of the Report 8

**Chapter 2.** **Organizational Overview** 9

2.1 Organization Vision 10

2.2 Organization Mission 10

2.3 Organization Services 10

2.4 Organizational Structure 11

2.5 My Position in this Organization 12

2.6 Address of the Organization 12

**Chapter 3.** **Requirement Engineering** 13

3.1 Requirement Analysis 14

3.2 Requirement Engineering 14

3.2.1 User and System Requirements 14

3.2.2 Functional Requirements 17

3.2.3 Non-Functional Requirements 18

3.3 Use Case Diagram of the System 20

**Chapter 4.** **Analysis** 22

4.1 Software Analysis Pattern 23

4.2 Activity Diagrams 24

4.2.1 Activity Diagram for User Registration 24

4.2.2 Activity Diagram of Provider Verification 25

4.2.3 Activity Diagram of Booking Lifecycle 26

4.2.4 Activity Diagram of Service Creation 27

4.2.5 Activity Diagram of User Management 28

4.2.6 Activity Diagram of Review Submission 29

4.2.7 Activity Diagram of Report Generation 30

4.3 Swim-lane Diagrams 30

4.3.1 Swim-lane Diagram for User Registration 31

4.3.2 Swim-lane Diagram for Service Creation 31

4.3.3 Swim-lane Diagram for Booking Services 32

4.3.4 Swim-lane Diagram for Reschedule Services 32

4.3.5 Swim-lane Diagram for Provider Verification 33

4.3.6 Swim-lane Diagram for User Management 33

4.4 Sequence Diagrams 34

4.4.1 Sequence Diagram for Registration 34

4.4.2 Sequence Diagram for Login 34

4.4.3 Sequence Diagram for Booking Services 35

4.4.4 Sequence Diagram for Payment 35

4.4.5 Sequence Diagram for Provider Verification 36

4.4.6 Sequence Diagram for Service Creation 36

4.4.7 Sequence Diagram for Service Reschedule 37

4.5 Class Diagram of Neighbourly System 37

**Chapter 5.** **Project Management** 39

5.1 Risk Identification 40

5.2 Risk Analysis 41

5.3 Risk Planning 42

5.4 The RMMM Plan 43

**Chapter 6.** **Project Planning** 45

6.1 System Project Estimation 46

6.2 Function-Oriented Metrics 47

6.3 Identifying Complexity 47

6.3.1 Identifying Complexity of Transaction Functions 47

6.3.2 Identifying Complexity of Data Functions 48

6.3.3 Unadjusted Function Point Contribution 48

6.3.4 Unadjusted Function Point Contribution 49

6.3.5 Performance and Environmental Impact 50

6.3.6 Counting Adjusted Function Point 51

6.4 Project Schedule Chart (12- week timeframe) 51

6.5 Accounts Table 52

6.5.1 Personnel Cost 52

6.5.2 Expected Hardware Cost 52

6.5.3 Expected Software Cost 53

6.5.4 Expected Other Cost 53

6.6 Clarification of Academic vs. Industry Estimates 54

**Chapter 7.** **Designing** 55

7.1 Interface Design 56

7.1.1 Non-Logged in User Features 56

7.1.2 Provider Features 59

7.1.3 Users Features 61

7.1.4 Admin Features 63

7.2 Data Flow Diagram (DFD) 65

7.2.1 Context-Level DFD (Level 0) 66

7.2.2 DFD Level 1 67

7.2.3 DFD Level 2 68

7.3 Database Design 70

7.3.1 Normalization of Data Fields 71

7.3.2 Entity Relationship (ER) Model 71

7.3.3 Database Table Structure 73

7.4 ER Diagram for Neighbourly System 77

**Chapter 8.** **System Quality and Testing** 79

8.1 System Quality Management 80

8.1.1 Software Quality Management Process 80

8.1.2 Software Test Cases 81

**Chapter 9.** **Ethical Consideration** 83

9.1 Ethical Considerations in the Software Development Process 84

9.2 Sustainability in the Software Development Process 86

**Chapter 10.** **Conclusion** 88

10.1 Brief Overview of the Project 89

10.2 Proposed System Benefits 89

10.3 Limitations of the Project 89

10.4 Practicum and Its Value 90

10.5 Future Plan 91

**Bibliography** 92

## **List of Figures**

Figure 1.1: Incremental Process Model 6

Figure 2.1 Organizational Structure 11

Figure 3.1: Use Case Diagram 20

Figure 4.1: Clean Code Architecture in ASP .NET Core 23

Figure 4.2: Activity Diagram for User Registration 24

Figure 4.3: Activity Diagram of Provider Verification 25

Figure 4.4: Activity Diagram of Booking Lifecycle 26

Figure 4.5: Activity Diagram of Service Creation 27

Figure 4.6: Activity Diagram of User Management 28

Figure 4.7: Activity Diagram of Review Submission 29

Figure 4.8: Activity Diagram of Report Generation 30

Figure 4.9: Swim-lane Diagram for User Registration 31

Figure 4.10: Swim-lane Diagram for Service Creation 31

Figure 4.11: Swim-lane Diagram for Booking Services 32

Figure 4.12: Swim-lane Diagram for Reschedule Services 32

Figure 4.13: Swim-lane Diagram for Provider Verification 33

Figure 4.14: Swim-lane Diagram for User Management 33

Figure 4.15: Sequence Diagram for Registration 34

Figure 4.16: Sequence Diagram for Login 34

Figure 4.17: Sequence Diagram for Booking Services 35

Figure 4.18: Sequence Diagram for Payment 35

Figure 4.19: Sequence Diagram for Provider Verification 36

Figure 4.20: Sequence Diagram for Service Creation 36

Figure 4.21: Sequence Diagram for Service Reschedule 37

Figure 4.22: Class Diagram of Neighbourly System 38

Figure 6.1: Project Schedule Gantt Chart 51

Figure 7.1: Hero Section in Home Page 56

Figure 7.2: Featured Services in Home page 57

Figure 7.3: Search page 57

Figure 7.4: Service Details Page 57

Figure 7.5: Review Section in Service Details page 58

Figure 7.6: More services from same Provider 58

Figure 7.7: Similar Service Section 58

Figure 7.8: Providers Verification Request Form 59

Figure 7.9: Providers Dashboard 59

Figure 7.10: Provider Active Services Page 59

Figure 7.11: Providers booking page 60

Figure 7.12: Providers Payment History Page 60

Figure 7.13: Providers My Review Page 60

Figure 7.14: Providers Earning Report page 61

Figure 7.15: Users Dashboard 61

Figure 7.16: Users My booking page 61

Figure 7.17: SSLCommerz payment page 62

Figure 7.18: Users Payment History page 62

Figure 7.19: Users My Review page 62

Figure 7.20: Users Activity Report Page 63

Figure 7.21: Admin Dashboard 63

Figures 7.22: User Management 63

Figure 7.23: Provider Management 64

Figure 7.24: Service Category Management 64

Figure 7.25: Platform Analytics & Reports 64

Figure 7.26: Admin Audit Log 65

Figure 7.27: Context Level DFD (Level 0) 66

Figure 7.28: Level 1 DFD 67

Figure 7.29: DFD Level 2 Process 1 68

Figure 7.30: DFD Level 2 Process 2 68

Figure 7.31: DFD Level 2 Process 3 69

Figure 7.32: DFD Level 2 Process 4 69

Figure 7.33: DFD Level 2 Process 5 69

Figure 7.34: DFD Level 2 Process 6 70

Figure 7.35: DFD Level 2 Process 7 70

Figure 7.36: DFD Level 2 Process 8 70

Figure 7.37: ER Diagram of Neighbourly System 78

Figure 9.1: Sustainability Considerations in Software Development 87

## **List of Tables**

Table 5.1: Risk Identification Table 40

Table 5.2: Risk Analysis Table 41

Table 5.3: Risk Planning Table 42

Table 5.4: Risk Mitigation, Monitoring, and Management Table 44

Table 6.1: Unadjusted Function Point Contribution (Transaction Function) 48

Table 6.2: Unadjusted Function Point Contribution (Data Function) 49

Table 6.3: Performance and Environmental Impact (GSC) 50

Table 6.4: Personal Cost 52

Table 6.5: Expected Hardware Cost 52

Table 6.6: Expected Software Cost 53

Table 6.7: Expected Other Cost 53

Table 7.1: ApplicationUser Table 73

Table 7.2: Service Table 73

Table 7.3: Booking Table 74

Table 7.4: ServiceCategory Table 74

Table 7.5: Review Table 74

Table 7.6: VerificationRequest Table 75

Table 7.7: Notification Table 75

Table 7.8: AuditLog Table 75

Table 7.9: ProviderTimeSlot Table 76

Table 7.10: RescheduleProposal Table 76

Table 7.11: ServiceBlock Table 77

Table 7.12: CategoryRequest Table 77

Table 8.1: Test Cases for the System 81

## **Chapter 1.**

## **Introduction**

The digital transformation of the building materials and interior solutions industry has created unprecedented opportunities for businesses to reach wider audiences and streamline their operations. Auraluxe represents a comprehensive e-commerce platform developed to address the growing demand for premium building materials and interior solutions in the digital marketplace. This practicum project was undertaken at Tilottoma, a leading premium building materials and interior solutions provider in Bangladesh, where the practical application of modern web technologies was employed to create a robust, scalable, and user-friendly e-commerce solution.

The development of Auraluxe encompasses the complete lifecycle of a modern web application, from requirement analysis and system design to implementation and testing. This project demonstrates the integration of cutting-edge technologies including Next.js, NestJS, PostgreSQL, and Prisma ORM to create a single-vendor e-commerce platform that serves both administrative and customer needs effectively. The platform facilitates seamless product management, inventory control, order processing, and secure payment transactions through SSLCommerz integration.

This practicum experience provided invaluable hands-on exposure to professional software development practices, agile methodologies, and real-world problem-solving in the context of e-commerce solutions. The project not only fulfills the academic requirements of the Bachelor of Computer Science and Engineering program but also contributes a functional business solution that addresses genuine market needs in the premium building materials sector.

#### 1.1 Background of the Study

The building materials and interior solutions industry in Bangladesh has experienced significant growth over the past decade, driven by rapid urbanization, infrastructure development, and increasing consumer awareness about quality construction materials. Traditional brick-and-mortar showrooms, while effective, present limitations in terms of accessibility, product information dissemination, and customer convenience. The COVID-19 pandemic further accelerated the need for digital transformation in this sector, as customers increasingly prefer online browsing and purchasing options.

Tilottoma, established in 1980, has been at the forefront of providing premium imported luxury building materials and interior solutions in Bangladesh. With over four decades of experience, the organization recognized the imperative need to establish a strong digital presence to complement its physical operations and serve the evolving needs of modern customers. The absence of a comprehensive e-commerce platform limited the company's ability to showcase its extensive product catalog, reach geographically dispersed customers, and provide detailed product information that aids informed purchasing decisions.

The e-commerce sector in Bangladesh has witnessed exponential growth, with the digital payment infrastructure improving significantly through services like SSLCommerz, bKash, and Nagad. This technological advancement has made online transactions more secure and accessible to a broader demographic. However, the building materials industry has been slower to adopt comprehensive digital solutions compared to other retail sectors, presenting both a challenge and an opportunity.

This practicum addresses the critical need for a sophisticated, feature-rich e-commerce platform specifically tailored for the premium building materials sector. The platform must handle complex product categorizations, detailed specifications, brand management, project portfolios, and secure payment processing while maintaining an intuitive user experience. The development of Auraluxe represents a practical application of modern software engineering principles to solve real-world business challenges, bridging the gap between traditional business operations and contemporary digital commerce requirements.

Furthermore, this practicum provides an opportunity to explore the full-stack development ecosystem using industry-standard technologies such as the MERN stack principles adapted with Next.js and NestJS, demonstrating proficiency in both frontend and backend development, database design, API architecture, and payment gateway integration. The project serves as a comprehensive learning experience that encompasses system analysis, software design patterns, security implementation, and deployment strategies essential for professional software development careers.

#### 1.2 Methodology

The development of the Auraluxe e-commerce platform employed a systematic approach utilizing both primary and secondary data sources to ensure comprehensive understanding of requirements, effective implementation, and successful project delivery. The methodology encompassed various research techniques, development practices, and validation processes.

##### 1.2.1 Primary Sources

The primary data sources provided firsthand information and direct insights essential for the project development:

* **Stakeholder Interviews**: Conducted detailed discussions with Tilottoma management, sales personnel, and warehouse staff to understand business processes, product management workflows, and customer interaction patterns.
* **Customer Surveys**: Gathered feedback from existing customers regarding their preferences for online shopping features, payment methods, and product information requirements.
* **Requirement Gathering Sessions**: Organized multiple sessions with project supervisors and organizational representatives to define functional and non-functional requirements clearly.
* **Direct Observation**: Observed current manual processes for inventory management, order processing, and customer service operations to identify pain points and automation opportunities.
* **User Testing**: Conducted usability testing sessions with potential end-users to validate interface design, navigation flow, and feature accessibility.
* **Performance Monitoring**: Implemented analytics and monitoring tools to gather real-time data on application performance, user behavior, and system bottlenecks.
* **Supervisor Feedback**: Regular consultations with academic and organizational supervisors to ensure alignment with project objectives and industry standards.

##### 1.2.2 Secondary Sources

Secondary data sources supplemented primary research with theoretical foundations and industry best practices:

* **Technical Documentation**: Extensive review of official documentation for Next.js, NestJS, Prisma ORM, PostgreSQL, React Hook Form, TanStack Query, and SSLCommerz to understand capabilities and implementation patterns.
* **Academic Literature**: Studied research papers and academic publications on e-commerce systems, software engineering methodologies, database design principles, and user experience optimization.
* **Industry Reports**: Analyzed market research reports on e-commerce trends in Bangladesh, building materials industry growth patterns, and digital payment adoption statistics.
* **Case Studies**: Examined successful e-commerce implementations in similar industries to identify best practices, common pitfalls, and innovative features.
* **Online Resources**: Utilized Stack Overflow, GitHub repositories, Medium articles, and developer communities for problem-solving, code examples, and architectural patterns.
* **E-commerce Standards**: Reviewed industry standards for online payment security (PCI DSS), data protection, accessibility guidelines (WCAG), and performance benchmarks.
* **Competitive Analysis**: Studied existing e-commerce platforms in the building materials sector to identify market gaps, feature expectations, and differentiation opportunities.
* **Technology Blogs**: Followed technology blogs and newsletters to stay updated on emerging trends, security vulnerabilities, and framework updates.

#### 1.3 Objectives

The objectives of the Auraluxe e-commerce platform development have been categorized into broad and specific objectives to provide clear direction and measurable outcomes for the practicum work.

##### 1.3.1 Broad Objective

To design, develop, and deploy a comprehensive single-vendor e-commerce platform for premium building materials and interior solutions that enhances customer experience, streamlines administrative operations, and facilitates secure online transactions while demonstrating proficiency in full-stack web development using modern technologies.

##### 1.3.2 Specific Objectives

The specific objectives that guide the development and implementation of the Auraluxe platform are:

* **Develop a robust product management system** that enables administrators to efficiently create, update, and organize products with comprehensive details including specifications, pricing, images, categories, and brand associations.
* **Implement a user-friendly customer interface** that allows customers to browse products seamlessly, search and filter based on multiple criteria, view detailed product information, and access project portfolios showcasing real-world applications.
* **Create a secure authentication and authorization system** that manages user registration, login, password recovery, and role-based access control to ensure data security and appropriate feature access for different user types.
* **Build an efficient shopping cart and order management system** that enables customers to add products to cart, modify quantities, proceed through checkout, and track order status while allowing administrators to manage orders and update fulfillment stages.
* **Integrate SSLCommerz payment gateway** to provide secure, reliable, and diverse payment options including credit/debit cards, mobile banking, and internet banking, ensuring transaction security and customer confidence.
* **Design and implement a scalable database schema** using PostgreSQL and Prisma ORM that maintains data integrity, supports complex relationships between entities, and optimizes query performance for enhanced application responsiveness.
* **Develop RESTful APIs using NestJS** that follow industry best practices, implement proper error handling, validation, and documentation to ensure maintainable and extensible backend architecture.
* **Implement responsive frontend design** using Next.js and TypeScript that provides optimal viewing experience across devices, implements server-side rendering for improved performance and SEO, and utilizes modern state management techniques.
* **Establish comprehensive testing and quality assurance procedures** including form validation using Zod, API testing, and user acceptance testing to ensure reliability, functionality, and user satisfaction.
* **Deploy the application in a production environment** with proper configuration, security measures, monitoring, and documentation to demonstrate readiness for real-world usage.

#### 1.4 Process Model

The development of the Auraluxe e-commerce platform follows the **Incremental Process Model**. This approach was selected to accommodate the complex nature of e-commerce systems, enable progressive delivery of functional modules, and ensure systematic validation of each component before proceeding to subsequent development phases.

The Incremental Process Model is a software development methodology where the system is designed, implemented, and tested incrementally until the complete product is achieved. Unlike the waterfall model that delivers the entire system at once, the incremental model delivers the system in parts, called increments. Each increment represents a portion of the system's functionality, and when integrated, these increments form the complete application.

In the context of Auraluxe development, the system was divided into multiple increments, each representing distinct functional modules such as user authentication, product catalog management, shopping cart functionality, order processing, payment integration, and content management. Each increment went through the complete software development lifecycle including requirements analysis, design, implementation, testing, and deployment before the next increment began.

##### ****1.4.1 Key characteristics of the Incremental Process Model implementation:****

* **Modular Development Approach**: The entire e-commerce platform was decomposed into logical functional modules (authentication module, product management module, cart module, order module, payment module, and content management module), each developed as a separate increment.
* **Sequential Increment Delivery**: Development followed a planned sequence starting with foundational features (user authentication and database setup) progressing to core business logic (product catalog and categories) and culminating in advanced features (payment gateway integration and order tracking).
* **Early Functional Delivery**: The first increment delivered a minimal viable product with basic user authentication and product listing capabilities, providing early validation of the core concept and allowing stakeholder feedback on fundamental functionality.
* **Integration After Each Increment**: Upon completion of each increment, it was integrated with previously developed increments, ensuring compatibility and smooth interaction between different modules of the system.
* **Independent Testing Phases**: Each increment underwent rigorous unit testing, integration testing, and validation before being accepted, ensuring quality and functionality at every stage of development.
* **Progressive Requirement Refinement**: Requirements for subsequent increments could be refined based on insights gained from earlier increments, allowing for realistic planning and expectation management.

##### 1.4.2 Reasons for Choosing the Incremental Process Model

* **Risk Reduction**: By developing and testing the system in small increments, technical and functional risks are identified and mitigated early in the development process. If an increment fails, only that specific module needs revision rather than the entire system.
* **Early Functionality Demonstration**: Delivering working increments early in the development cycle allows stakeholders to see tangible progress, validate requirements, and provide feedback when changes are still relatively easy to implement.
* **Easier Testing and Debugging**: Smaller increments are easier to test comprehensively. Isolating bugs and issues is more straightforward when working with focused functional modules rather than the entire complex system.
* **Flexible Priority Management**: Critical features can be developed in early increments ensuring that if time or resource constraints arise, the system will still have its most essential functionality operational.
* **Better Resource Allocation**: Development resources can be allocated efficiently across increments, with team members focusing on specific modules according to their expertise and project timeline requirements.
* **Stakeholder Engagement**: Regular delivery of working increments maintains stakeholder interest and confidence in the project, demonstrating continuous progress and value delivery throughout the development period.
* **Manageable Complexity**: Breaking down the complex e-commerce system into manageable increments makes the overall development process less overwhelming and more organized, facilitating better project management and progress tracking.
* **Accommodation of Learning Curve**: As this is a practicum project, the incremental approach allows for learning and skill development in stages, with each increment building upon knowledge and experience gained from previous ones.
* **Parallel Development Opportunities**: Once the architectural foundation is established in early increments, certain independent modules can be developed in parallel, optimizing the development timeline without compromising integration integrity.
* **Easier Maintenance and Enhancement**: The modular nature of incremental development results in a well-structured codebase where individual modules can be maintained, updated, or enhanced independently without affecting the entire system.

#### 1.5 Feasibility Study

The feasibility study for the Auraluxe e-commerce platform was conducted to assess the viability of the project from technical, economic, and operational perspectives. This comprehensive analysis ensured that the project objectives could be achieved within available resources, constraints, and organizational context.

##### 1.5.1 Technical Feasibility

The Auraluxe platform is technically feasible based on the following considerations. All required technologies including Next.js, NestJS, PostgreSQL, Prisma ORM, TypeScript, React Hook Form, Zod, TanStack Query, Axios, and TanStack Table are open-source, well-documented, and widely supported by active developer communities with proven track records in production environments. The development team possesses adequate knowledge and skills in JavaScript/TypeScript, React ecosystem, Node.js backend development, database design, and RESTful API architecture, with any knowledge gaps addressable through available documentation and online resources. The project requires standard web hosting infrastructure supporting Node.js applications and PostgreSQL databases, readily available through cost-effective cloud platforms such as Vercel for Next.js frontend and Railway or Heroku for NestJS backend with managed PostgreSQL services. SSLCommerz provides comprehensive API documentation and sandbox environments specifically designed for the Bangladesh market with established integration patterns for Node.js applications. The chosen technology stack supports both horizontal and vertical scaling, with PostgreSQL handling large datasets efficiently, Next.js providing optimized rendering strategies, and NestJS architecture supporting future microservices expansion. Modern development tools including Visual Studio Code, Git, GitHub, Postman, and Prisma Studio are freely available and sufficient for all project requirements. The technologies selected are capable of meeting performance requirements including page load times under 3 seconds, API response times under 1 second, and support for concurrent users as specified in system requirements, while providing built-in security features including JWT authentication, password hashing, input validation, SQL injection prevention through Prisma ORM, and HTTPS encryption for data transmission.

##### 1.5.2 Economic Feasibility

The Auraluxe platform is economically feasible with minimal financial investment. The project primarily requires time and effort rather than substantial financial investment, as all core technologies are open-source and free to use, eliminating licensing costs, with development conducted as part of the academic practicum under university and organizational supervision. Cloud hosting services offer generous free tiers and student programs suitable for development and initial deployment, with Vercel providing free hosting for Next.js applications and services like Railway or Render offering free PostgreSQL databases, resulting in estimated production-level hosting costs of only $10-30 monthly. SSLCommerz operates on a transaction-based fee model charging 2-3% per successful transaction with no upfront setup fees or monthly subscriptions, making it economically viable with pay-as-you-grow pricing. Ongoing maintenance costs are minimal as the platform uses stable, well-maintained open-source technologies where updates and security patches are typically free and easily applied. For the host organization, the platform provides significant value by expanding market reach, reducing manual operational overhead, enabling 24/7 product showcase, and facilitating direct customer engagement, with the digital platform potentially increasing sales opportunities without proportional increases in operational costs. The benefits of digital presence, improved customer experience, streamlined operations, and competitive advantage far outweigh the minimal development and operational costs, making the project highly economically feasible.

##### 1.5.3 Operational Feasibility

The Auraluxe platform is operationally feasible as it seamlessly integrates with Tilottoma's existing business workflows without requiring major operational overhauls. The administrative interface is designed to be intuitive and user-friendly, requiring minimal training for staff members to manage products, process orders, and monitor inventory effectively. The platform complements rather than replaces current operations, allowing the organization to maintain their established processes while benefiting from digital automation and improved efficiency. The host organization has demonstrated strong commitment to this digital transformation initiative by providing necessary support, resources, and stakeholder availability throughout the development process. Customer acceptance is expected to be high as the platform addresses genuine needs for convenient online access to product information, detailed specifications, and secure payment options. The well-structured codebase following industry best practices ensures long-term maintainability, and the modular architecture supports future enhancements and scalability as business needs evolve, making the platform a sustainable operational asset for the organization.

##### 1.6 Structure of the Report

The report is structured systematically into four chapters to provide a comprehensive overview of the Auraluxe e-commerce platform development project. Following this Introduction, Chapter 2 provides an in-depth overview of Tilottoma, the host organization, including its vision, mission, services, organizational structure, and the specific role undertaken during the practicum period. Chapter 3 presents the Methods and System Implementation, detailing the requirement engineering process, system architecture and design, database schema development, frontend and backend implementation strategies, API development, payment gateway integration, security measures, and comprehensive testing methodologies employed throughout the development lifecycle. Finally, Chapter 4 concludes the report with a detailed discussion of project outcomes, challenges encountered and solutions implemented, lessons learned, achievement of objectives, system limitations, and recommendations for future enhancements and scalability considerations.

## **Chapter 2.**

## **Organizational Overview**

ATI Limited is a leading software development company that has been delivering innovative technology solutions since 1998. Established with a vision to transform businesses through cutting-edge software development, ATI Limited has grown into a trusted partner for organizations seeking robust, scalable, and efficient digital solutions. The company specializes in custom software development, web application development, enterprise resource planning (ERP) systems, and mobile application development, serving clients across multiple industries and geographical locations.

#### 2.1 Organization Vision

To be the most trusted technology partner for businesses worldwide, delivering innovative software solutions that drive digital transformation and sustainable growth.

#### 2.2 Organization Mission

ATI Limited is committed to providing end-to-end custom software solutions that streamline operations, enhance productivity, and drive innovation. The company focuses on understanding client requirements deeply and delivering scalable, high-quality software solutions from development to deployment. Through continuous innovation and a client-centric approach, ATI Limited aims to help businesses leverage technology to achieve their strategic objectives and maintain competitive advantage in their respective industries.

#### 2.3 Organization Services

ATI Limited provides a range of services, including:

1. **Custom Software Development:** Development of software solutions designed for specific business requirements across industries.
2. **Enterprise Resource Planning (ERP) Solutions:** Implementation of ERP systems to integrate operations, increase efficiency, and support decision-making.
3. **Mobile Application Development:** Recognizing the growing importance of mobile technology, ATI Limited develops innovative mobile applications for iOS and Android platforms
4. **E-commerce Platform Development:** Creation of e-commerce websites and mobile applications that support online transactions and business growth.

#### 2.4 Organizational Structure

ATI Limited operates with a hierarchical organizational structure designed to ensure effective project management, quality assurance, and client satisfaction. The company is led by experienced management professionals who oversee various departments including Software Development, Web Development, Mobile Development, Quality Assurance, Project Management, and Client Relations. Each department is headed by senior technical leads who manage teams of developers, designers, and analysts. The organizational structure promotes collaboration, knowledge sharing, and adherence to software engineering best practices.

#### 2.5 My Position in this Organization

During my practicum period at ATI Limited, I am working as a **Junior Frontend Developer** within the Web Development department. In this role, I am responsible for developing and implementing the Auraluxe e-commerce platform, working under the supervision of senior developers and the project lead.

#### 2.6 Address of the Organization

ATI Centre, House-07

Gareeb-E-Nawaz Avenue, Sector-11

Uttara, Dhaka-1230, Bangladesh

## **Chapter 3.**

## **Requirement Engineering**

#### 3.1 Requirement Analysis

Requirement analysis is a pivotal phase that defines the scope, functionality, and user experience of **Auraluxe**. The analysis focused on identifying the core problem in the building material industry: the lack of a centralized, digital platform where customers can view imported luxury materials and real-world project inspirations simultaneously.

Currently, customers often have to visit physical showrooms to see premium materials or understand how they look in a finished project. **Auraluxe** addresses this by offering a "Single Vendor" e-commerce solution where users can browse materials (stones, tiles, fittings), view completed architectural projects for inspiration, and order products directly. The analysis of the reference site (Tilottoma) highlighted the need for features like "Project Showcases," "Brand-wise filtering," and "Material Categorization" to enhance the digital purchasing experience.

#### 3.2 Requirement Engineering

The requirements were categorized into User, System, Functional, and Non-Functional requirements to ensure a holistic definition of the system.

##### 3.2.1 User and System Requirements

This section maps high-level user needs to specific, testable system requirements.

**1. Authentication and Account Management**

* 1. The system shall provide a secure registration system allowing customers to create accounts using email and password.
  2. The system shall enforce role-based access control (RBAC) covering **User** and **Admin** roles, as defined in the database schema (UserRole enum).
  3. The system shall ensure secure login functionality using **NestJS** backend authentication strategies (JWT).
  4. The system shall allow users to manage their profiles, including viewing their order history and updating personal details.

**2. Product and Project Discovery**

* 1. The system shall allow users to browse products filtered by Category, Brand, or Material type.
  2. The system shall implement a Project Showcase section where users can view architectural projects (including Project Name, Location, Architects, and Images) to understand how materials are applied in real life.
  3. The system shall provide a search bar allowing keyword-based queries for products and brands.
  4. The system shall display detailed product cards including Product Name, Price, Specifications, Key Features, and Stock Quantity.

1. **Cart and Order Management**
   1. The system shall enable users to add products to a Cart, supporting multiple items with adjustable quantities.
   2. The system shall validate stock availability (quantity in Product model) before allowing an item to be added to the cart.
   3. The system shall allow users to proceed to checkout and generate an order with a unique ID.
   4. The system shall implement a CartItem model to track unit prices and quantities for each session.

**4. Payments and Transactions**

* 1. The system shall integrate **SSLCommerz** for handling secure online payments.
  2. The system shall ensure transactional integrity by updating the order status only upon successful payment confirmation.
  3. The system shall maintain a transaction history accessible to both the User (My Orders) and the Admin.

**5. Administration and Content Management**

**5.1** The Admin shall have a dashboard to view total users, active orders, and inventory status.

**5.2** The system shall provide **CRUD (Create, Read, Update, Delete)** capabilities for the Admin to manage:

* **Products:** Uploading images, setting prices, and linking to Brands/Categories.
* **Projects:** Managing portfolio entries with details like Client, Architects, and Location.
* **Home Banners:** Managing the dynamic slider images on the frontend (homeBanner model).

**5.3** The Admin shall be able to manage **Brands** and **Brand Types**, including uploading logos and descriptions.

##### 3.2.2 Functional Requirements

Functional requirements define the specific behaviors of the Auraluxe system.

**User Functions:**

* **Registration/Login:** Secure entry using Zod-validated forms.
* **Browse Inventory:** View products using TanStack Table or Grid views.
* **Manage Cart:** Add/Remove items and view total cost calculations.
* **Place Order:** Complete purchases via SSLCommerz gateway.
* **View Projects:** Access the "Inspiration" section to see Project and Material relations.

**Admin Functions:**

* **Inventory Management:** Monitor quantity levels and update stock.
* **Order Fulfillment:** View incoming orders and update their status (e.g., Pending to Delivered).
* **Content Management:** Update "Our Featured" and "Home Banner" sections to keep the site fresh.
* **Data Entry:** Add new Materials, Project Types, and Key Brands into the database.

##### 3.2.3 Non-Functional Requirements

Non-functional requirements define system quality and performance standards.

1. **Security:** The system shall ensure secure handling of user data by hashing passwords using **bcrypt** and managing sessions via **JSON Web Tokens (JWT)** strategies in NestJS. Role-based authorization (User, Admin, Vendor) shall restrict access to sensitive routes, and all form inputs shall be validated on both client and server sides using **Zod** schemas to prevent SQL injection and XSS attacks.
2. **Performance:** The system shall deliver high-speed performance by leveraging **Next.js Server-Side Rendering (SSR)** to ensure initial page loads occur under 2 seconds. API data fetching shall be optimized using **TanStack Query** to cache server states and minimize redundant network requests, while high-resolution product images shall be optimized automatically using the Next.js Image component.
3. **Scalability:** The system shall be designed to support future business growth by utilizing the modular architecture of **NestJS**, allowing independent scaling of services. The **PostgreSQL** database shall be structured with **Prisma ORM**, ensuring that complex relationships between products, brands, and projects remain performant as the dataset grows.
4. **Reliability and Availability:** The platform shall ensure data integrity during critical operations, such as inventory deduction and payment processing, by adhering to **ACID principles** within PostgreSQL transactions. The system shall handle API failures gracefully using global exception filters to prevent application crashes and ensure continuous availability.
5. **Usability:** The user interface shall be fully responsive across mobile, tablet, and desktop devices, implemented using **Tailwind CSS**. The system shall provide immediate visual feedback for user interactions (such as "Added to Cart" or form errors) to ensure a seamless, premium shopping experience consistent with a luxury brand.
6. **Maintainability:** The system shall be developed using a strictly typed **TypeScript** codebase to reduce runtime errors and improve code readability. The backend structure shall separate concerns into Controllers, Services, and Repositories, while the database schema shall be centrally managed and documented via the **Prisma Schema** file.

#### 3.3 Use Case Diagram of the System

Figure 3.1: Use Case Diagram

* **Actor: User** -> Uses cases: Register, Login, Search Product, View Project Showcase, Add to Cart, Pay via SSLCommerz.
* **Actor: Admin** -> Uses cases: Login, Manage Products, Manage Projects, Manage Brands, View Orders, Update Home Banners.

## **Chapter 4.**

## **Analysis**

System analysis is the process of breaking down the Auraluxe e-commerce platform into its component parts to understand their relationships and interactions. In this phase, the requirements gathered in Chapter 3 were transformed into visual models and logical structures. The analysis focuses on understanding the flow of data between the Next.js frontend, the NestJS backend, and the PostgreSQL database, ensuring that the architecture effectively supports both luxury product sales and the unique "Project Inspiration" showcase features.

#### 4.1 Software Analysis Pattern

We adopted a Modular Layered Architecture (aligned with Clean Architecture principles) for Auraluxe using the NestJS framework. This pattern organizes code into distinct modules and layers, ensuring that the core business logic remains independent of external frameworks or database specifics. This separation of concerns promotes the longevity, testability, and maintainability of the codebase.

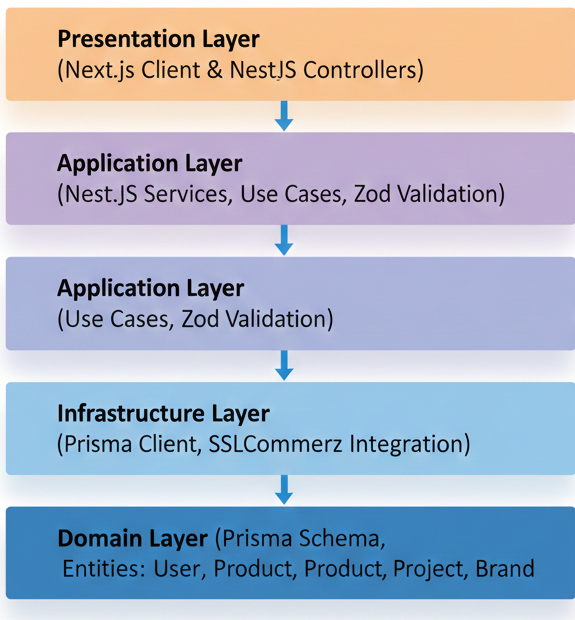


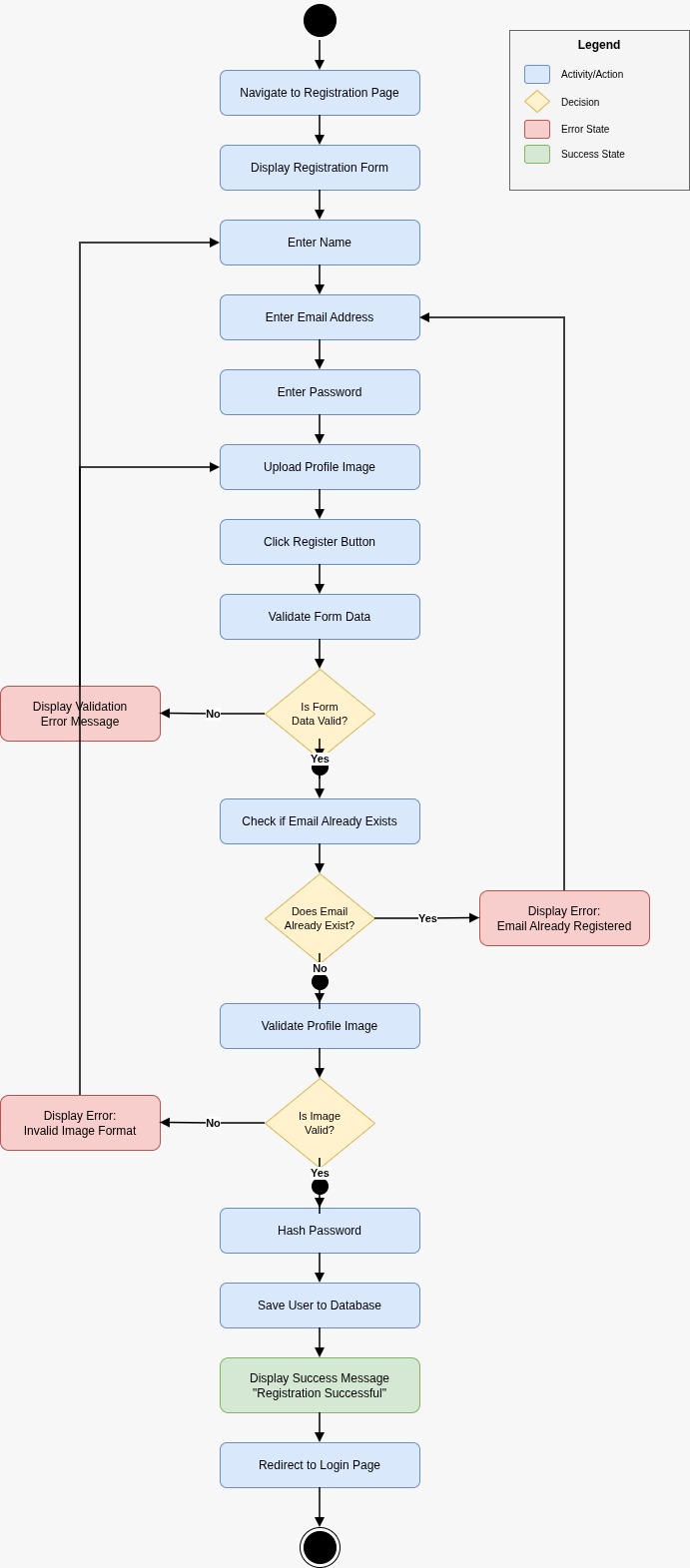
Figure 4.1: Clean Code Architecture

* **Domain Layer (Entities):** The core of the system containing the Enterprise Logic and Data Models (User, Product, Brand, Project). In our system, these are defined via the Prisma Schema, serving as the single source of truth for our data structure without external dependencies.
* **Application Layer (Services):** Contains the business rules and Use Cases. It depends on the Domain layer and orchestrates the flow of data. This layer (implemented as NestJS Services) handles logic such as stock verification, Zod validation, and order processing before interacting with the database.
* **Infrastructure Layer:** Implements the interfaces and data access mechanisms required by the Application layer. In Auraluxe, this includes the Prisma Client (for PostgreSQL access) and external integrations like the SSLCommerz payment gateway. It depends on the Application layer to know what data to fetch or save.
* **Presentation Layer (Controllers & Client):** The entry point for the application. It consists of NestJS Controllers (which handle incoming API requests) and the Next.js Frontend (which presents the UI to the user). This layer depends on the Application layer to retrieve data and return responses to the client.

#### 4.2 Activity Diagrams

Activity diagrams illustrate the dynamic nature of the system by modeling the flow of control from activity to activity.

##### 4.2.1 Activity Diagram for User Registration

Figure 4.2: Activity Diagram for User Registration

##### 4.2.2 Activity Diagram of Admin

Figure 4.3: Activity Diagram of Admin

##### 4.2.3 Activity Diagram of product purchase

Figure 4.4: Activity Diagram of product purchase

#### 4.3 Swim-lane Diagrams

Swim-lane diagrams delineate specific responsibilities within a process, clarifying the interaction between the User, System, Database, and External Services.

##### 4.3.1 Swim-lane Diagram for User Registration

Figure 4.9: Swim-lane Diagram for User Registration

##### 4.3.2 Swim-lane Diagram for Admin Dashboard

Figure 4.9: Swim-lane Diagram for Admin Dashboard

##### 4.3.3 Swim-lane Diagram for Product Purchase

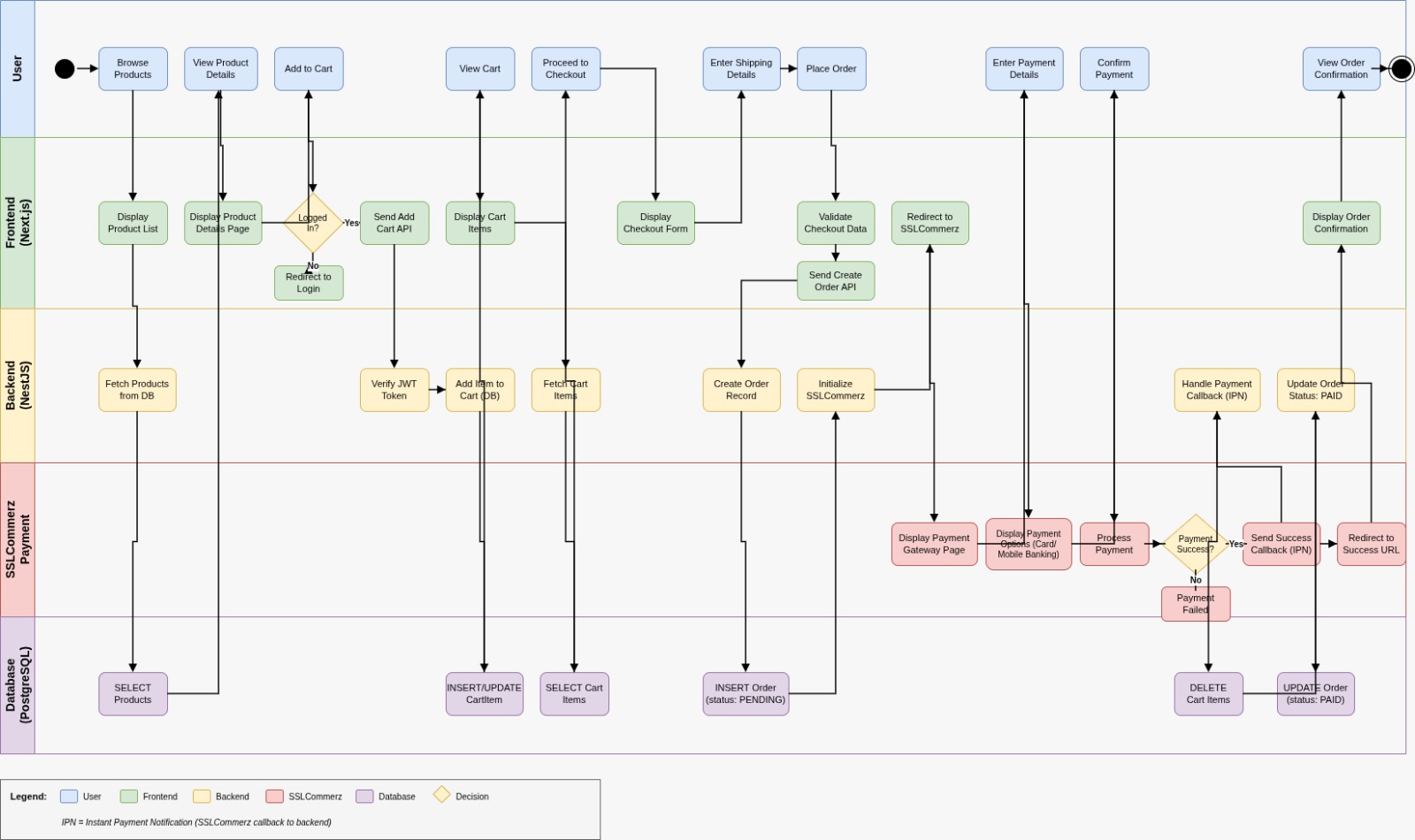
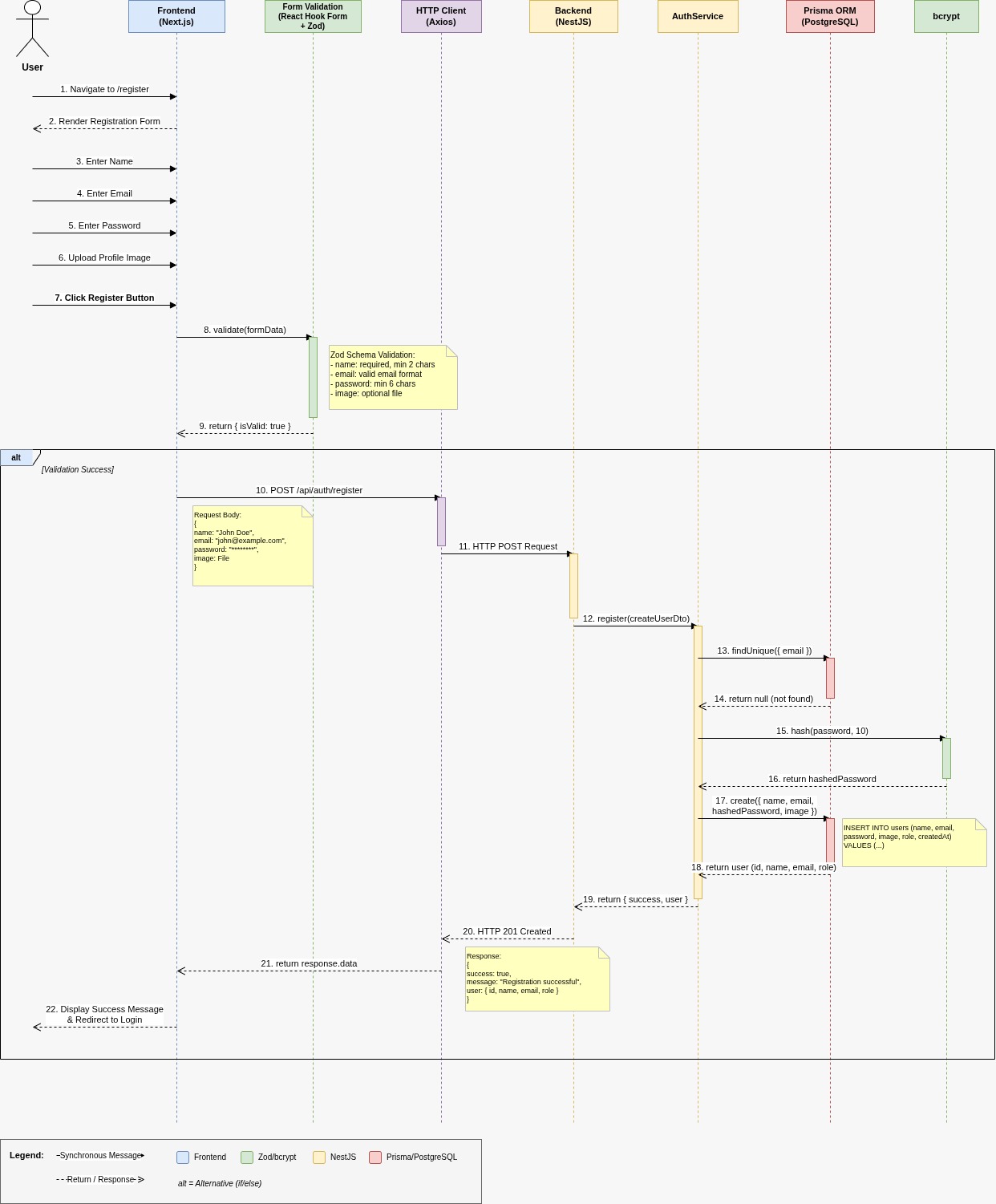


Figure 4.9: Swim-lane Diagram for Product Purchase

#### 4.4 Sequence Diagrams

Sequence diagrams capture the detailed interaction between objects over time for specific use cases.

##### 4.4.1 Sequence Diagram for Registration

Figure 4.16: Sequence Diagram for Registration 

##### 4.4.2 Sequence Diagram for Admin Dashboard

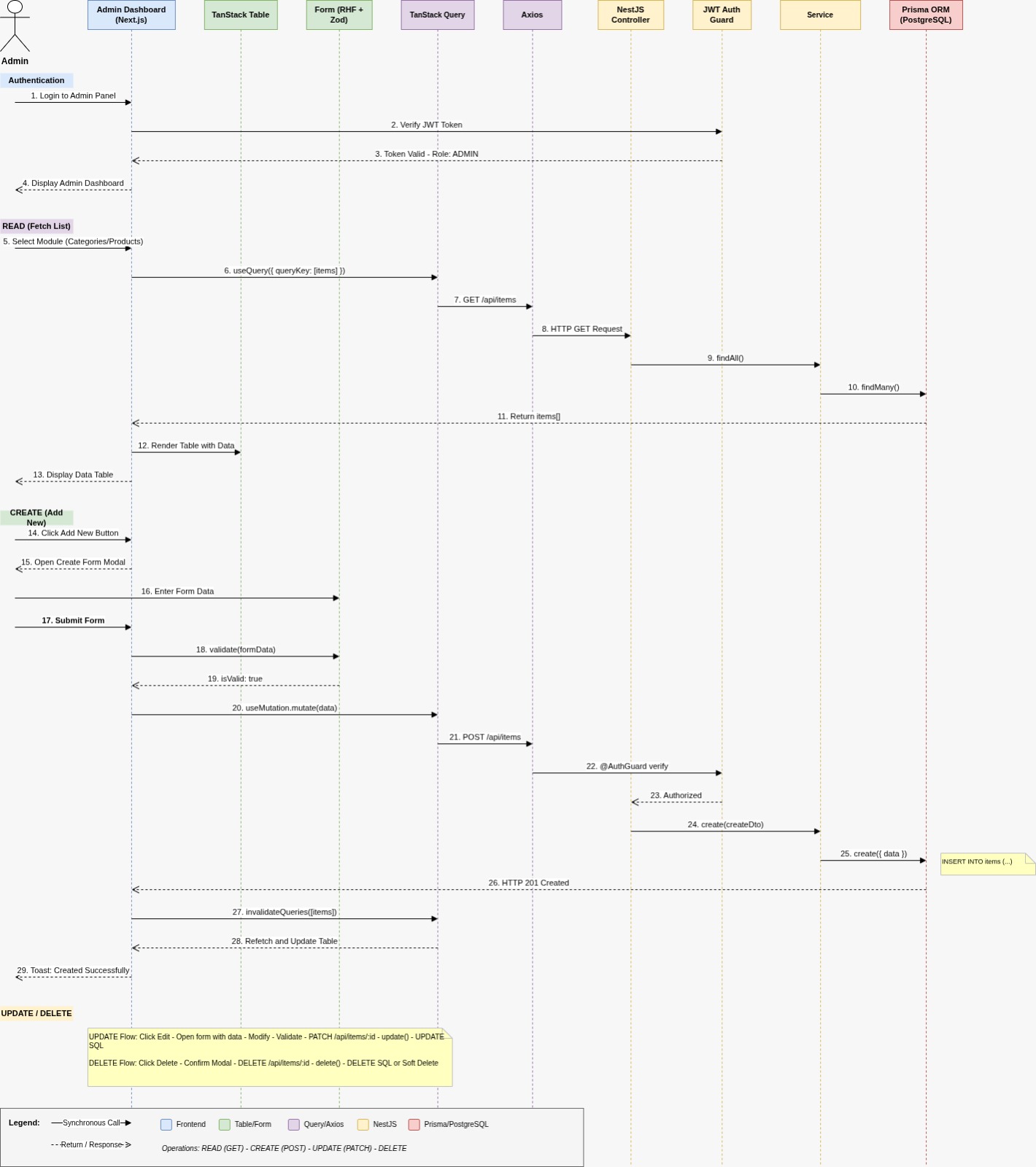


Figure 4.15: Sequence Diagram for Admin Dashboard

##### 4.4.3 Sequence Diagram for Product Purchase

Figure 4.15: Sequence Diagram for Product Purchase

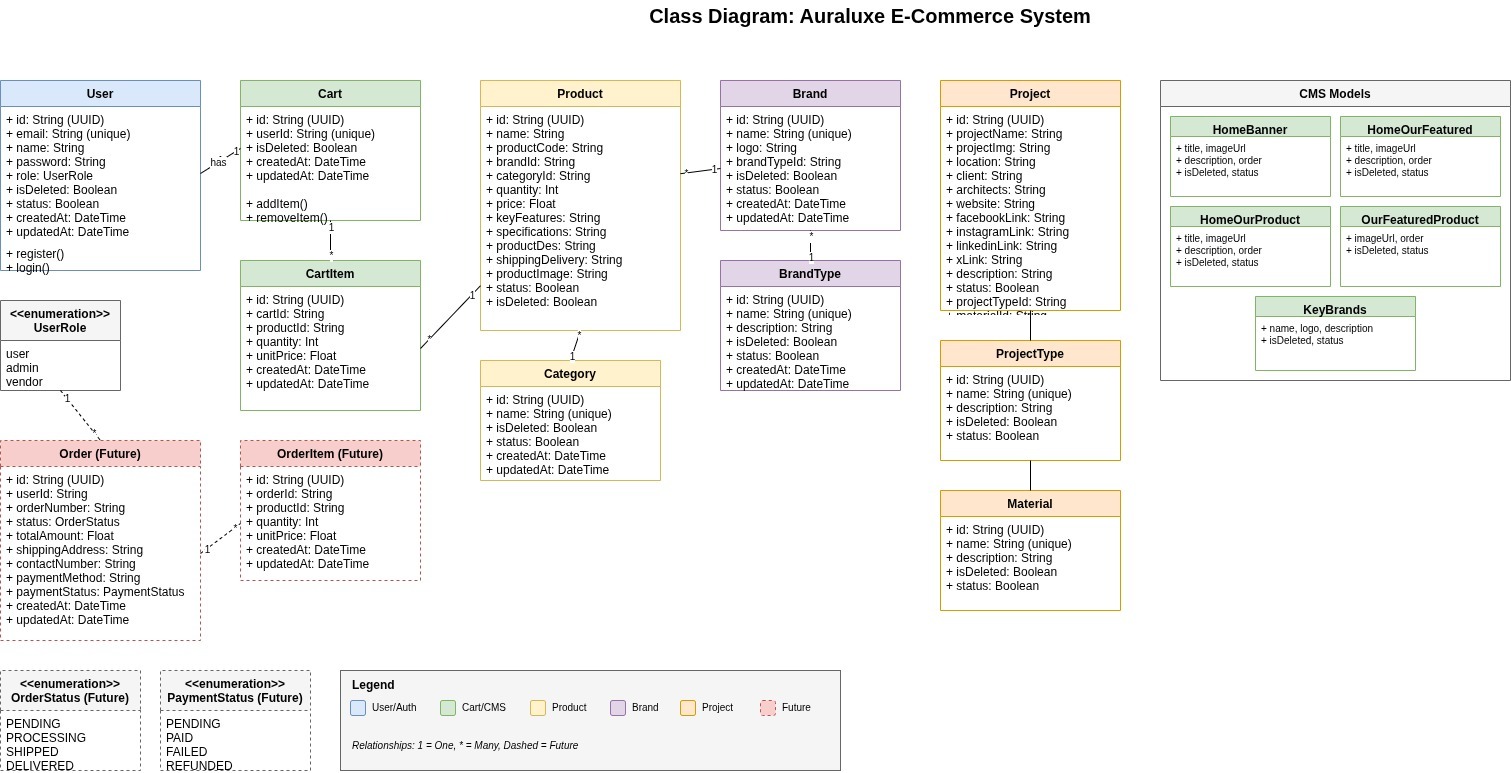
#### 4.5 Class Diagram of the Auraluxe System

The Class Diagram represents the static structure of the system, showing classes, attributes, operations, and relationships.

* User Class: Id, Name, Email, Password, Role (User/Admin/Vendor), Status, CreatedAt, UpdatedAt.
* Product Class: Id, Name, ProductCode, Price, Quantity, KeyFeatures, Specifications, ProductDes, ShippingDelivery, ProductImage, Status.
* Brand Class: Id, Name, Logo, BrandTypeId, Status.
* Category Class: Id, Name, Status.
* Project Class: Id, ProjectName, ProjectImg, Location, Client, Architects, Website, Description, ProjectTypeId, MaterialId.
* Cart Class: Id, UserId, IsDeleted.
* CartItem Class: Id, CartId, ProductId, Quantity, UnitPrice.

**Relationships:**

* User has one Cart (1-to-1).
* Brand has many Products (1-to-Many).
* Category has many Products (1-to-Many).
* Cart has many CartItems (1-to-Many).
* Project belongs to one ProjectType and one Material (Many-to-1).
* Brand belongs to one BrandType (Many-to-1).
* Figure 4.22: Class Diagram of Neighbourly System

Figure 4.22: Class Diagram of Auraluxe 

## **Chapter 5.**

## **Project Management**

#### 5.1 Risk Identification

Risk identification is the first step in the risk management process, involving the discovery and documentation of potential threats that could negatively impact the development or deployment of the Auraluxe system. Given the nature of a premium e-commerce platform dealing with high-value luxury materials, the risks were categorized into technical, operational, and external factors.

Table 5.1: Risk Identification Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk ID** | **Risk Description** | **Category** | **Probability** | **Impact** |
| **R1** | The risk of SSLCommerz API downtime or integration errors, preventing users from completing high-value transactions. | Technical | Low | High |
| **R2** | The risk of concurrency issues where two users try to buy the last unit of a luxury item simultaneously, leading to negative inventory. | Technical | Medium | High |
| **R3** | High-resolution project/product images causing slow page loads in Next.js, leading to poor user experience (UX) and higher bounce rates. | Technical | High | Medium |
| **R4** | Since Auraluxe is a single-vendor system, unauthorized access to the Admin Dashboard could lead to catastrophic data manipulation (e.g., changing prices of luxury items). | Technical | Low | Critical |
| **R5** | Potential connection timeouts between the NestJS backend and the PostgreSQL database during peak traffic. | Technical | Low | Medium |

#### 5.2 Risk Analysis

Following identification, each risk was analyzed to determine its probability and impact. This allowed us to calculate the Risk Exposure and assign a Priority level to focus mitigation efforts effectively.

Table 5.2: Risk Analysis Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk ID | **Probability (P)** | **Impact (I)** | **Risk Exposure (P × I)** | **Priority** |
| R1 | Low | High | Medium | 2 |
| R2 | Medium | High | High | 1 |
| R3 | High | Medium | High | 1 |
| R4 | Low | Critical | Medium-High | 2 |
| R5 | Low | Medium | Low | 3 |

#### 5.3 Risk Planning

For each identified risk, a specific response strategy was developed. This planning phase ensured that the team was prepared to act immediately should any risk materialize.

Table 5.3: Risk Planning Table

|  |  |
| --- | --- |
| **Risk ID** | **Planned Response / Mitigation Strategy** |
| **R1** | Implement robust error handling in the NestJS backend to catch SSLCommerz failures and return user-friendly error messages instead of crashing the application. |
| **R2** | Use Prisma Transactions (Interactive Transactions) to lock database rows during checkout, ensuring stock is only deducted if the payment is confirmed. |
| **R3** | Utilize Next.js Image Optimization (<Image /> component) to automatically serve resized and WebP-formatted images. Implement lazy loading for the "Project Showcase" section. |
| **R4** | Enforce strict Role-Based Access Control (RBAC) using JWT Guards. Store admin passwords using strong hashing (bcrypt) and implement session timeouts. |
| **R5** | Configure a connection pool in PostgreSQL to manage multiple concurrent connections efficiently. |

#### 5.4 The RMMM Plan

The Risk Mitigation, Monitoring, and Management (RMMM) plan provided a framework for ongoing risk oversight throughout the practicum lifecycle.

1. **Risk Mitigation**

* **Incremental Development:** Adopted an agile approach, delivering the "Product Catalog" and "Admin Dashboard" modules first before attempting complex integrations like SSLCommerz.
* **Type Safety:** Used TypeScript across both frontend and backend to catch type-related errors (e.g., pricing calculations) at compile time rather than runtime.
* **Validation:** Implemented Zod schemas to validate all incoming data, preventing malformed requests from corrupting the database.

1. **Risk Monitoring**

* **API Logging:** Used NestJS interceptors to log the response time and status of every API call.
* **Database Monitoring:** Monitored Prisma query logs during development to identify slow queries (N+1 problems) and optimize them using include or select.
* **Visual Testing:** Regularly tested the UI on mobile devices to ensure the "Project Showcase" was responsive and images loaded correctly.

1. **Risk Management**

* **Fallback Provision:** Created static "Placeholder" images for products in case the dynamic image URL failed to load.
* **Transaction Rollback:** Configured the checkout logic to automatically rollback the entire database transaction if any step (e.g., Payment or Stock Update) failed, preserving data integrity.

Table 5.4: Risk Mitigation, Monitoring, and Management Table

|  |  |  |  |
| --- | --- | --- | --- |
| Risk ID | **Category** | **Mitigation Strategy** | **Monitoring Method** |
| **R1** | Technical (Payment) | Use SSLCommerz Sandbox for all dev testing; Validate IPN callbacks rigorously. | Check server logs for "Payment Failed" events. |
| **R2** | Data Integrity | Use Prisma $transaction API for atomic updates. | Monitor "Order Created" vs "Stock Deducted" counts. |
| **R3** | Performance | Use Next.js SSR for initial load; Cache data with TanStack Query. | Run Lighthouse performance audits on Home and Product pages. |
| **R4** | Security | Protect Admin routes with UseGuards(JwtAuthGuard, RolesGuard). | Log all failed Admin login attempts. |

## **Chapter 6.**

## **Project Planning**

#### Chapter 6: Project Planning

To ensure successful execution, the Auraluxe project required a structured project plan. This involved defining project objectives, system scope, and critical milestones such as requirements analysis, database schema design (Prisma), API development (NestJS), frontend implementation (Next.js), and testing. The plan also addresses resource allocation and risk management strategies. Key deliverables include administrative features (inventory management, project portfolio showcasing, order fulfillment), and user features (product filtering, cart management, secure SSLCommerz payments, and project inspiration viewing). The system ensures a premium user experience with server-side rendering for performance and secure role-based access control.

The following activities of project planning were followed in this system:

1. Estimation of project effort and resources.
2. Task prioritization based on dependencies (e.g., Backend APIs before Frontend UI).
3. Personnel requirements for development and testing.
4. Estimation of project cost and schedule.
5. Version control (Git/GitHub) for proper code and document management.

#### 6.1 System Project Estimation

The accuracy of estimation in the Auraluxe System depends on:

* **Estimating product size:** Scope includes modules such as User/Admin Authentication, Product Catalog Management (with Brands/Categories), Project/Inspiration Showcase, Cart & Order Lifecycle, Payment Gateway Integration, and the Admin Dashboard.
* **Effort, Calendar Time, and Cost:** Time and cost are estimated considering system complexity (specifically relational data integrity between Projects/Materials and the Order transaction flow), number of modules, and skill requirements. This involves effort for strict type-checking (TypeScript), Zod validation setup, and responsive UI implementation.
* **Consistency of requirements:** Requirements are kept stable throughout development, focused on the core "Single Vendor E-commerce" value proposition.

#### 6.2 Function-Oriented Metrics

The project estimation is based on function point analysis, which considers both data functions and transactional functions.

1. **Data Functions**

* **Internal Logical Files (ILF):** Data maintained within the system via PostgreSQL, such as Users, Products, Projects, Brands, Carts, Orders, and Home Banners.
* **External Interface Files (EIF):** Interfaces connecting to external services like SSLCommerz (Payment Gateway) and potentially external Image Hosting/CDN services.

1. **Transactional Functions**

* **External Inputs (EI):** User actions such as registering, adding items to cart, placing an order, and Admin actions like creating products, uploading project details, or updating banner images.
* **External Outputs (EO):** Order Invoices, Payment Success/Fail Messages, Admin Sales Reports.
* **External Inquiries (EQ):** Real-time queries such as filtering products by Brand/Category, searching for products, and viewing Project details.

#### 6.3 Identifying Complexity

##### 6.3.1 Identifying Complexity of Transaction Functions

* **Low complexity:** Simple inputs such as user login, logout, subscribing to newsletters, or viewing static "About Us" pages.
* **Medium complexity:** Product creation (requires linking Brand/Category + Image upload), updating Cart quantities, and filtering products by multiple attributes.
* **High complexity:** The Checkout process (involves stock validation -> order creation -> payment redirect), Secure Payment Handling (SSLCommerz IPN callbacks), and Dynamic Project Showcasing (linking Materials to Projects).

##### 6.3.2 Identifying Complexity of Data Functions

* **Low complexity:** Simple lookup tables such as BrandType, Category, Material, or ProjectType.
* **Medium complexity:** User profiles (Info + Role + Status) and HomeBanner configurations.
* **High complexity:** Product (links to Brand, Category, CartItems, OrderItems), Project (links to ProjectType, Material, Images), and Order (links to User, Payments, Status History).

##### 6.3.3 Unadjusted Function Point Contribution

Table 6.1: Unadjusted Function Point Contribution (Transaction Function)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Transaction Functions** | **FTRs** | **DETs** | **Complexity** | **UFP** |
| User/Provider Registration (EI) | 2 | 15 | Average | 4 |
| Authentication (Login/Reset) (EI) | 1 | 5 | Low | 3 |
| Search Products (Filter/Sort) (EQ | 2 | 12 | High | 6 |
| View Project Details (EQ) | 2 | 15 | Average | 4 |
| Admin: Create/Edit Product (EI) | 4 | 20 | High | 6 |
| Admin: Create/Edit Project (EI) | 3 | 15 | High | 6 |
| Add to Cart / Update Cart (EI) | 2 | 8 | Average | 4 |
| Checkout & Payment Init (EI) | 3 | 15 | High | 6 |
| Payment Verification (SSL) (EI) | 2 | 10 | High | 6 |
| Admin: Update Order Status (EI) | 2 | 5 | Average | 4 |
| Admin: Dashboard Stats (EO) | 2 | 10 | Average | 5 |
| **Total (Transaction)** |  |  |  | **54** |

##### 6.3.4 Unadjusted Function Point Contribution

Table 6.2: Unadjusted Function Point Contribution (Data Function)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Function (ILF/EIF)** | **FTRs** | **DETs** | **Complexity** | **UFP** |
| Model: User (ILF) | 2 | 10 | Average | 10 |
| Model: Product (ILF) | 4 | 20 | High | 15 |
| Model: Project (ILF) | 3 | 15 | High | 15 |
| Model: Order/Cart (ILF) | 3 | 12 | High | 15 |
| Model: Brand/Category (ILF) | 1 | 5 | Low | 7 |
| Model: HomeBanner (ILF) | 1 | 6 | Low | 7 |
| SSLCommerz API (EIF) | 1 | 15 | Average | 7 |
| **Total (Data)** |  |  |  | 76 |

##### 6.3.5 Performance and Environmental Impact

Table 6.3: Performance and Environmental Impact (GSC)

|  |  |
| --- | --- |
| **General System Characteristic** | **Rating** |
| Data Communications (API Calls) | 4 |
| Distributed Data Processing | 2 |
| Performance (SSR/SEO requirements) | 5 |
| Heavily Used Configuration | 2 |
| Transaction Rate | 4 |
| Online Data Entry | 4 |
| End-user Efficiency | 4 |
| Online Update | 3 |
| Complex Processing (Geospatial) | 3 |
| Reusability | 4 |
| Installation Ease | 3 |
| Operational Ease | 3 |
| Multiple Site | 1 |
| Facilitate Change | 4 |
| Total Degree of Influence (TDI) | 46 |

##### 6.3.6 Counting Adjusted Function Point

**Function Point Calculation**

* UFP for Transaction = 54
* UFP for Data = 76
* Total UFP = 130
* Value Adjustment Factor (VAF) = 0.65 + (0.01 × 46) = 1.11
* **Adjusted FP (AFP) = 130 × 1.11 = 144.3 (approx 144)**

**Effort & Schedule Calculation**

* Effort = AFP ÷ Productivity
  + Note: Standard productivity for a full-stack developer using modern tools (MERN) is approx 15-20 FP/month.
  + Effort = 144 ÷ 18 ≈ 8.0 person-months

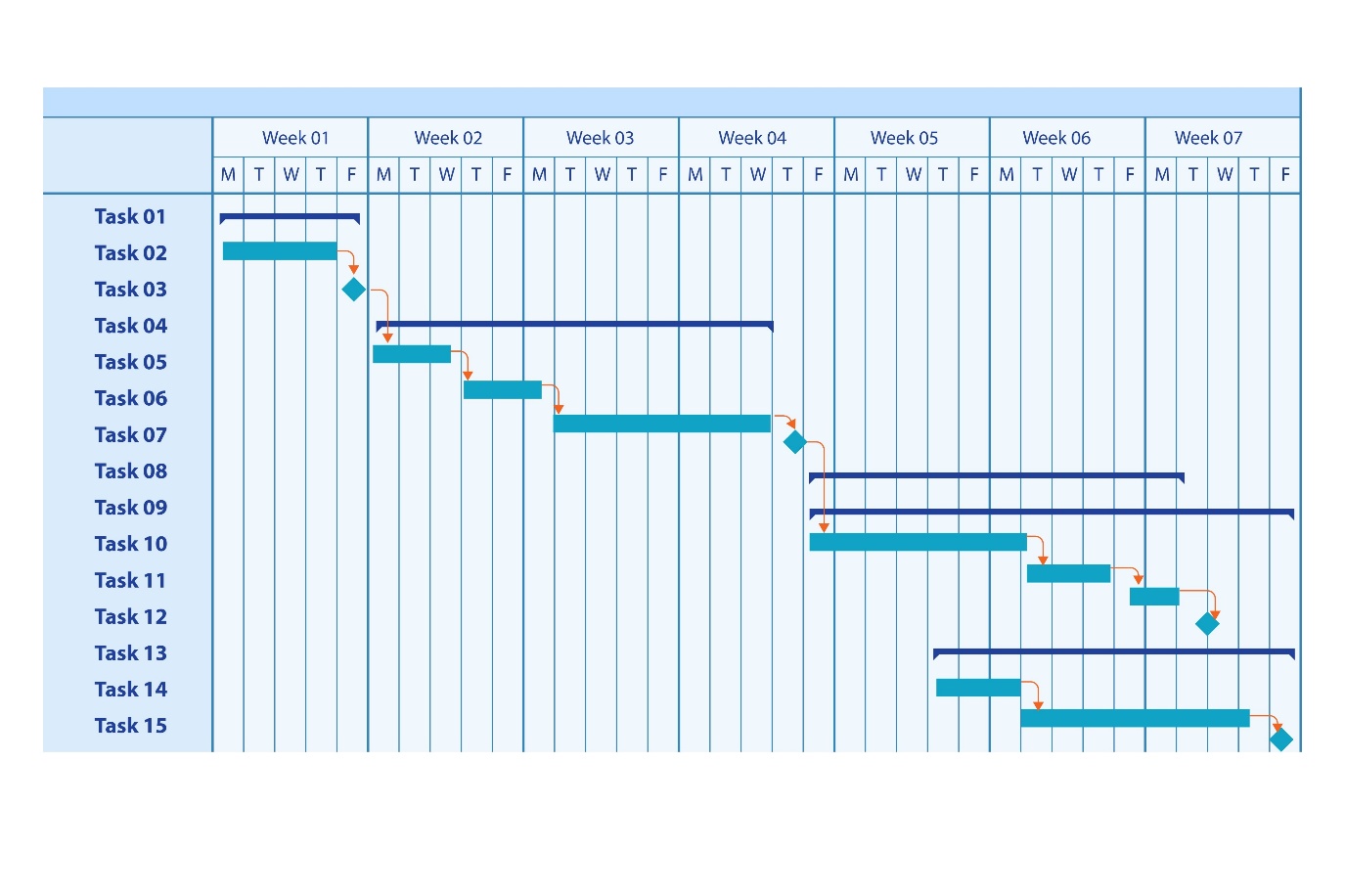
**Schedule (Duration):**

* Since I am working as a single developer for this practicum, but focusing on a specific subset of features (MVP) for the report:
* Target Duration = 3 to 4 months (Standard Semester Duration).
* This confirms the project scope is appropriate for a comprehensive semester-long internship project.

#### 6.4 Project Scheduling (Gantt Chart)

Project scheduling involves sequencing activities and assigning timeframes to ensure the timely completion of the Auraluxe platform. The schedule was designed to fit within a 14-week internship period, dividing the workload into distinct phases: Analysis, Design, Development, Integration, and Testing.

The timeline uses a Gantt Chart approach to visualize the critical path. The backend (NestJS) and database (Prisma) were prioritized to ensure the API was ready before the frontend (Next.js) integration began.

Figure 6.1: Project Schedule Gantt Chart

Development is carried out in three incremental 2-week phases: Phase 1 implements core Authentication and Role-Based Access Control to secure Admin and User portals, Phase 2 delivers the dynamic Product Catalog and the unique Project Inspiration Showcase, and Phase 3 completes the Shopping Cart logic and secure SSLCommerz Payment Integration. The schedule concludes with a 1-week Testing phase for validation and bug fixes, and a final 1-week Deployment phase for documentation and presentation, ensuring all dependencies are addressed.

#### 6.5 Accounts Table

Project costs are estimated based on a standard development environment for a medium-scale commercial e-commerce application.

##### 6.5.1 Personnel Cost

The largest portion of the budget is dedicated to personnel, as skilled manpower is essential for successful full-stack development and deployment.

Table 6.5: Personnel Cost

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Role** | **No. of Persons** | **Est. Cost (Monthly)** | **Duration** | **Total** |
| Project Manager | 1 | 80,000 BDT | 3 Mos | 240,000 BDT |
| Backend Developer | 1 | 60,000 BDT | 3 Mos | 180,000 BDT |
| Frontend Developer | 1 | 40,000 BDT | 3 Mos | 80,000 BDT |
| Total Personnel |  |  |  | 500,000 BDT |

#### 6.5.2 Expected Hardware Cost

To support development, testing, and hosting, certain hardware resources are required.

Table 6.5: Expected Hardware Cost

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Quantity** | **Unit Price** | **Total Cost** |
| Development Laptops | 3 | (Existing Assets) | 0 BDT |
| Local Server (Testing) | 1 | 5,000 BDT | 5,000 BDT |
| **Total Hardware** |  |  | **5,000 BDT** |

##### 6.5.3 Expected Software Cost

Licensed software and development tools are also considered in the cost structure.

Table 6.6: Expected Software Cost

|  |  |  |
| --- | --- | --- |
| **Item** | **Type** | **Cost** |
| Visual Studio Code | IDE | Free (Community) |
| PostgreSQL (Local/Neon) | Database | Free Tier |
| Vercel Pro (Deployment) | Hosting | 2,200 BDT (Monthly) |
| SSLCommerz Sandbox | Payment Gateway | Free |
| Domain Name (.com) | Web | 1,200 BDT |
| **Total Software** |  | **34,00 BDT** |

#### 6.5.4 Expected Other Cost

Other costs include miscellaneous but essential expenses.

Table 6.7: Expected Other Cost

|  |  |  |
| --- | --- | --- |
| **Item** | **Description** | **Cost** |
| Internet & Utility | High-speed connection | 6,000 BDT |
| Documentation | Printing & Binding | 2,000 BDT |
| Miscellaneous | Contingency fund | 5,000 BDT |
| **Total Other** |  | **13,000 BDT** |

**Grand Total Estimated Project Cost:** 521,400 BDT

#### 6.6 Clarification of Academic vs. Industry Estimates

It is important to note that this budget represents an industry implementation scenario for a company like "Tilottoma." In the context of this practicum report, the actual cost was close to zero because the project was completed as an academic exercise using free open-source tools (NestJS, Next.js, Postgres) and personal resources. The detailed accounts table therefore illustrates what an organization might expect to spend if deploying the Auraluxe system commercially, while the student project itself required minimal financial investment.

## **Chapter 7.**

## **Designing**

The design phase translated the requirements into detailed technical specifications, focusing on usability, data flow, and database integrity.

#### 7.1 Interface Design

The User Interface (UI) follows a minimalist, mobile-first design language, acknowledging that most users will access the platform via smartphones. Key design principles include:

1. **Simplicity:** Clear navigation menus for Dashboard, Reports, Search etc.
2. **Consistency:** Uniform design across all modules (fonts, colors, buttons).
3. **Feedback:** Confirmation pop-ups after booking, cancellation, payment etc.
4. **Accessibility:** Mobile-friendly design adaptable to multiple screen sizes.

##### 7.1.1 Non-Logged in User Features

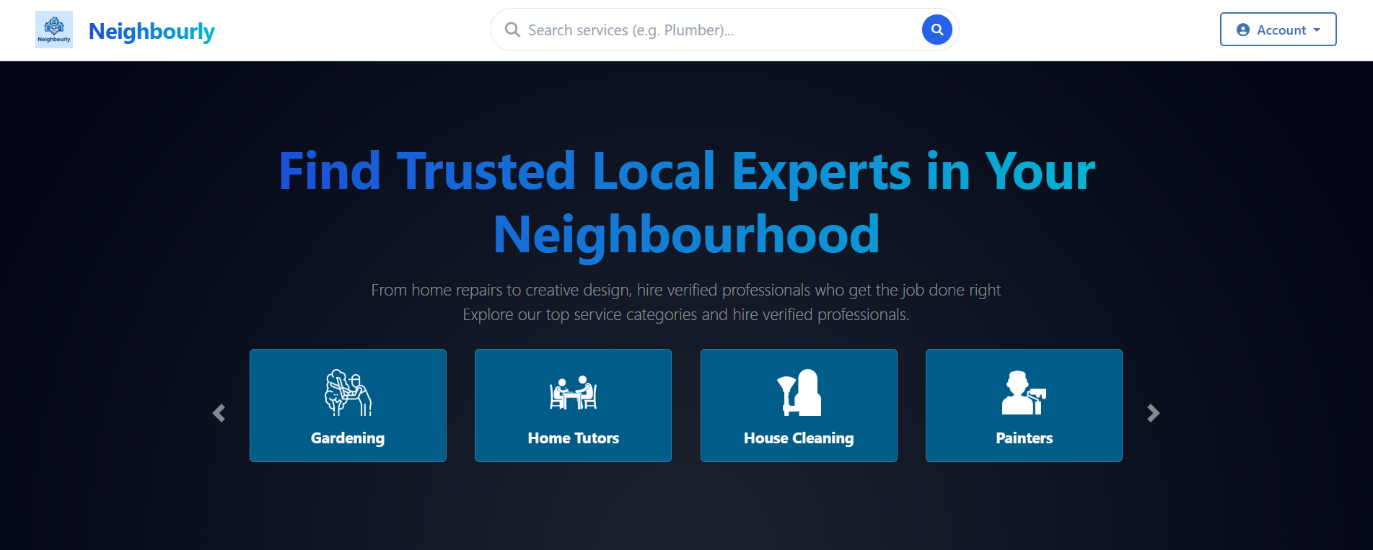
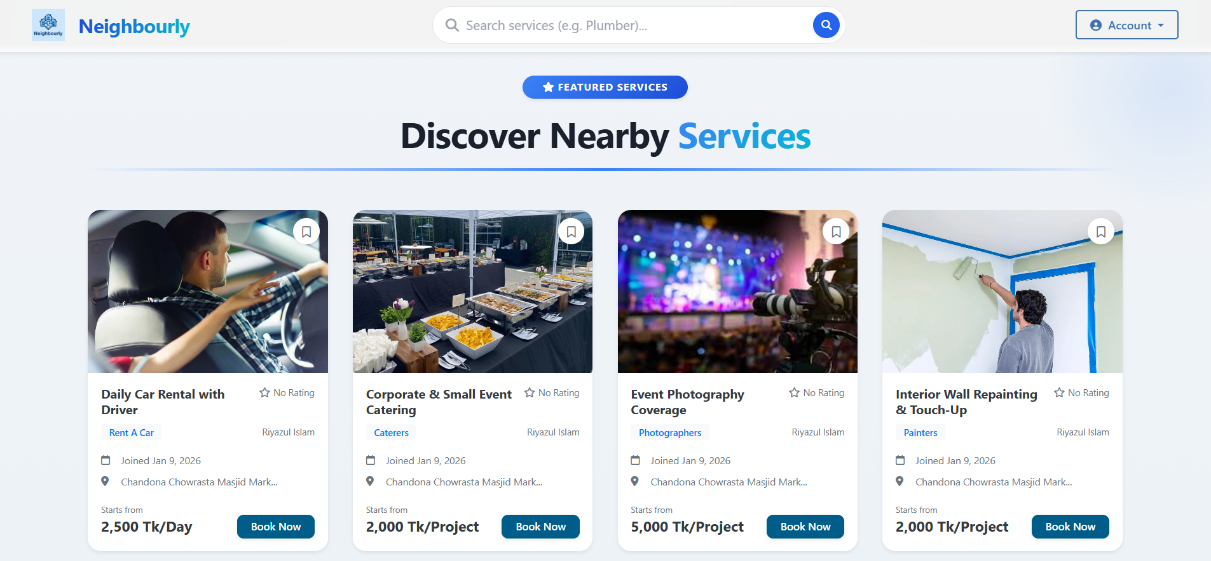
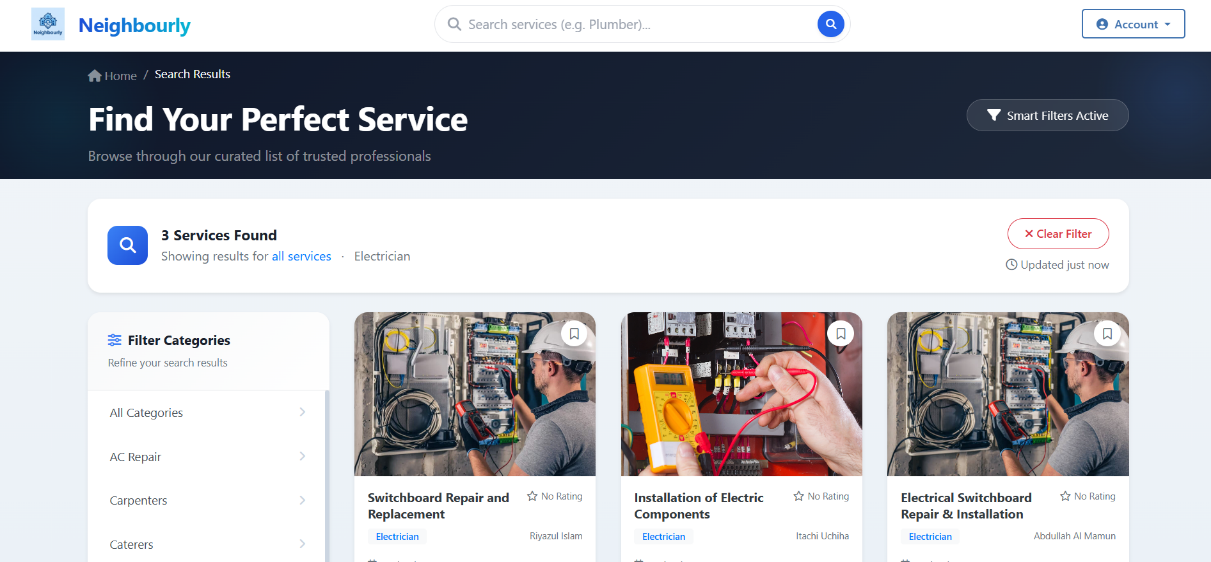
**Hero Section:** Hero Section in home page shows heading, sub-heading and a dynamic Category Carousal. Also, there are a Search bar and Account Drop Down button at the nav bar. So that users can easily Search for the services, and navigate to the Login/Register page.

Figure 7.1: Hero Section in Home Page

**Featured Service:** All the featured services show below the Hero Section. Initial two row of featured Service card shows. After Clicking Explore All Services button rest of the services shows up. Every service card loads via AJAX for a smooth user experience.

Figure 7.2: Featured Services in Home page

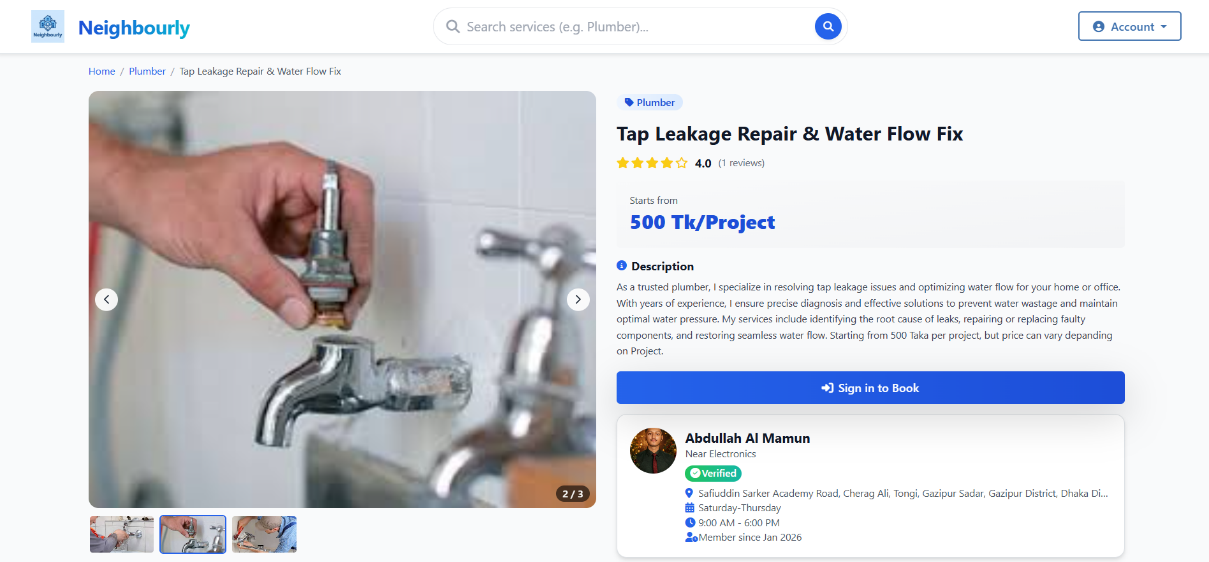
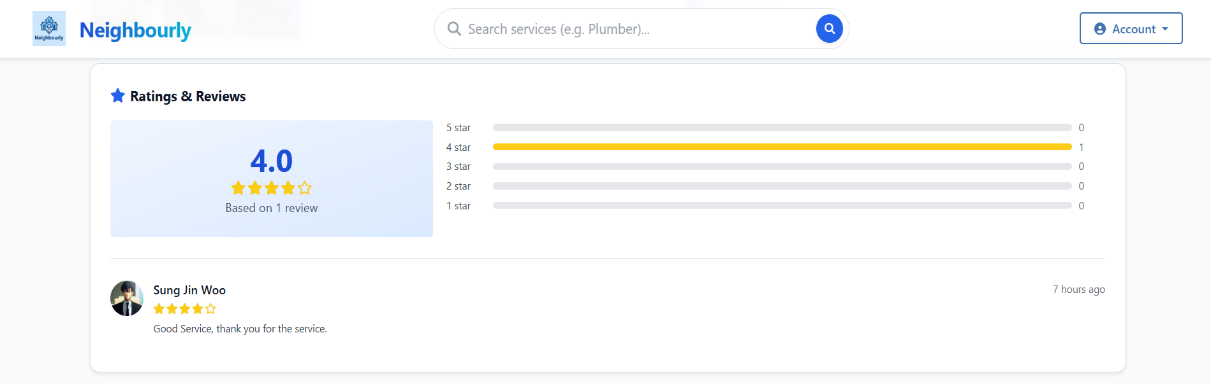
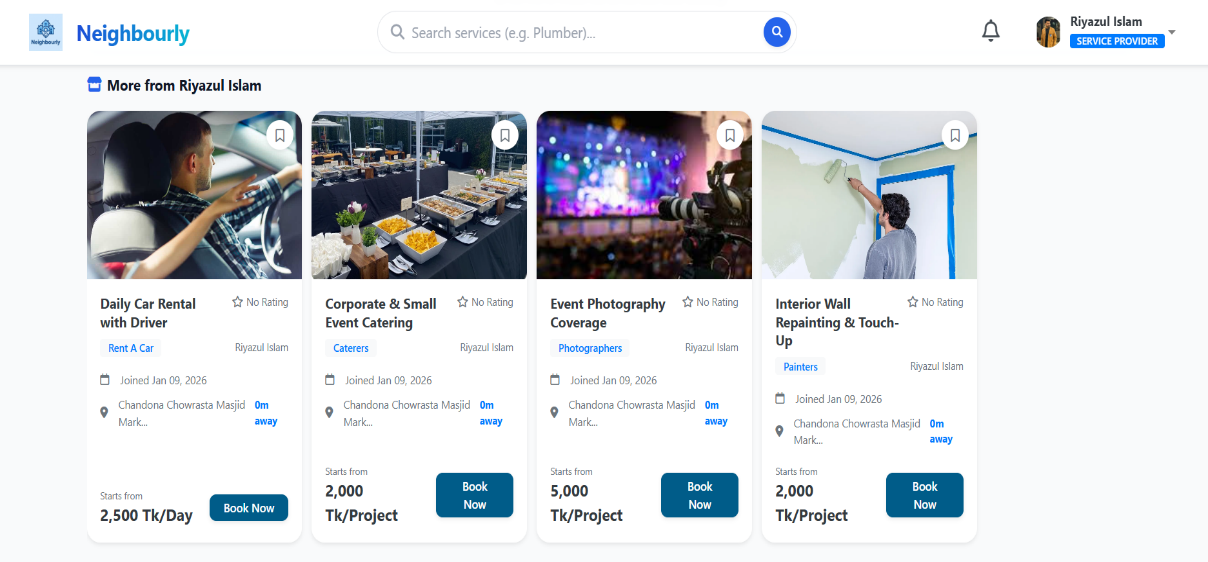
Figure 7.3: Search page

Figure 7.4: Service Details Page

Figure 7.5: Review Section in Service Details page

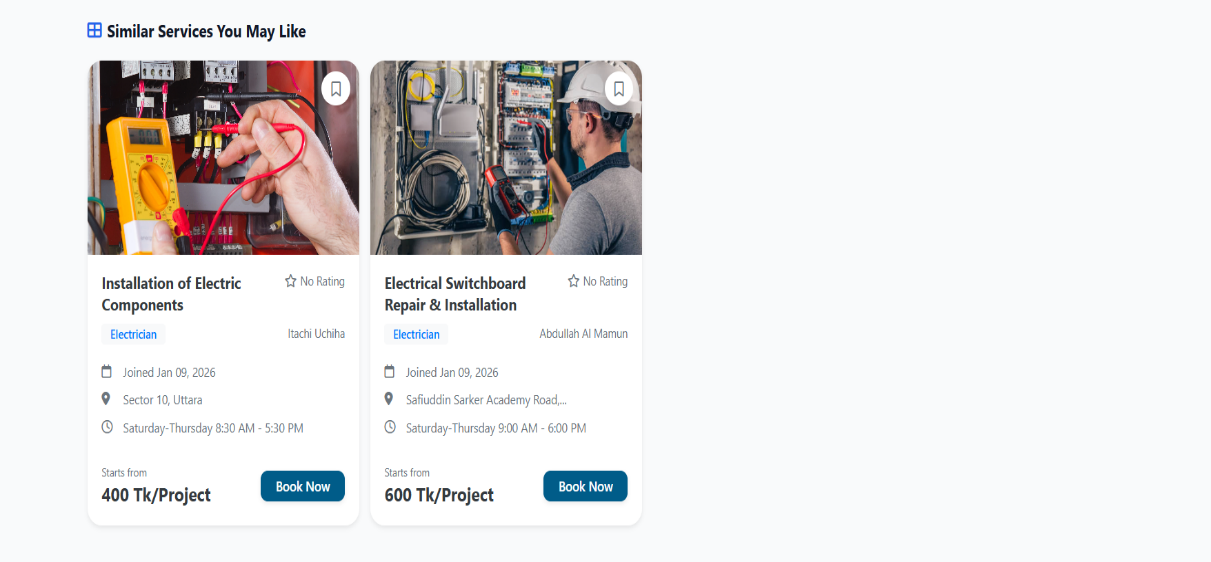
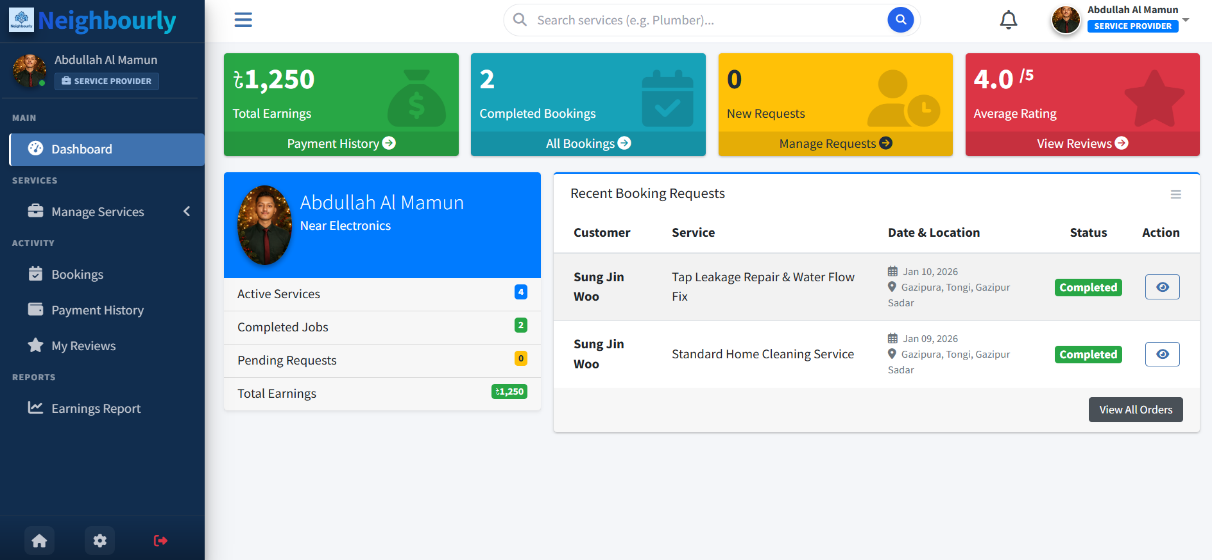
Figure 7.6: More services from same Provider

Figure 7.7: Similar Service Section

##### 7.1.2 Provider Features

Figure 7.8: Providers Verification Request Form

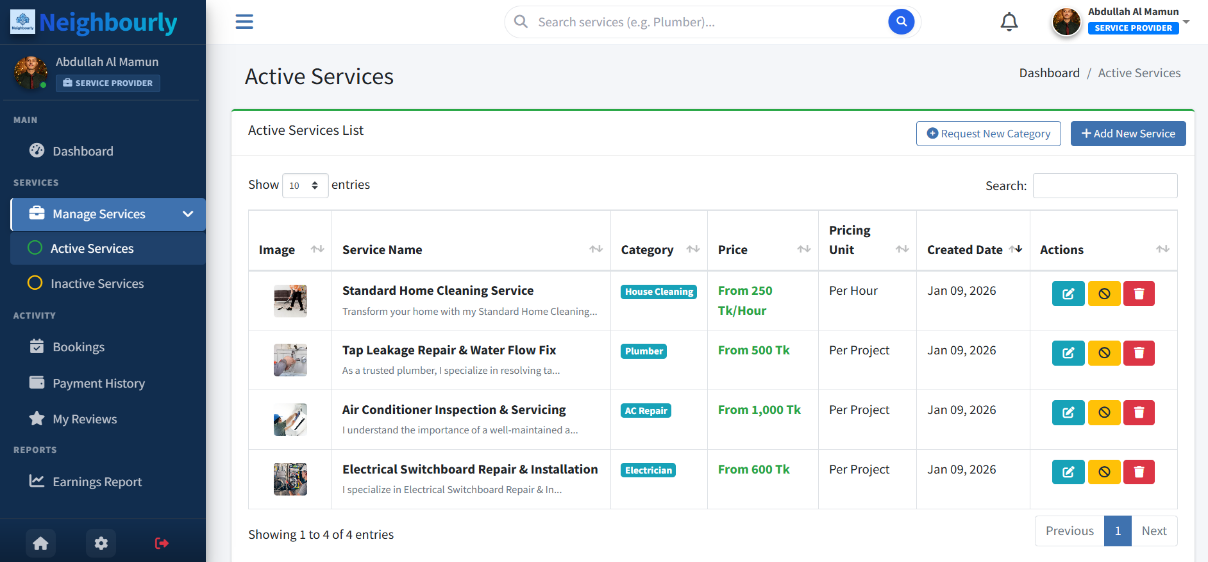
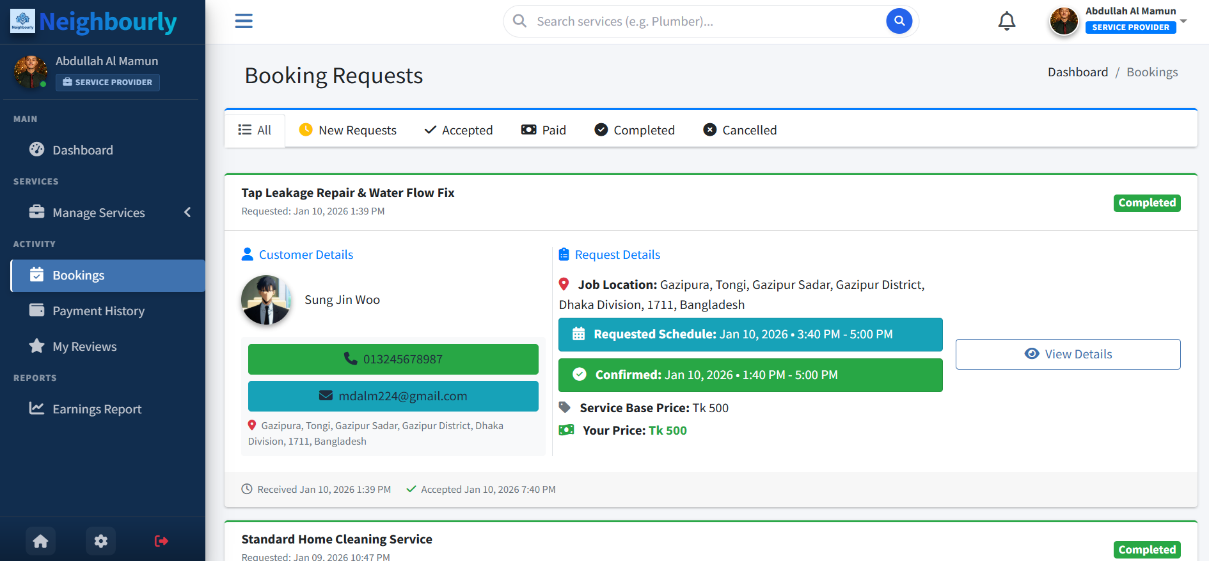
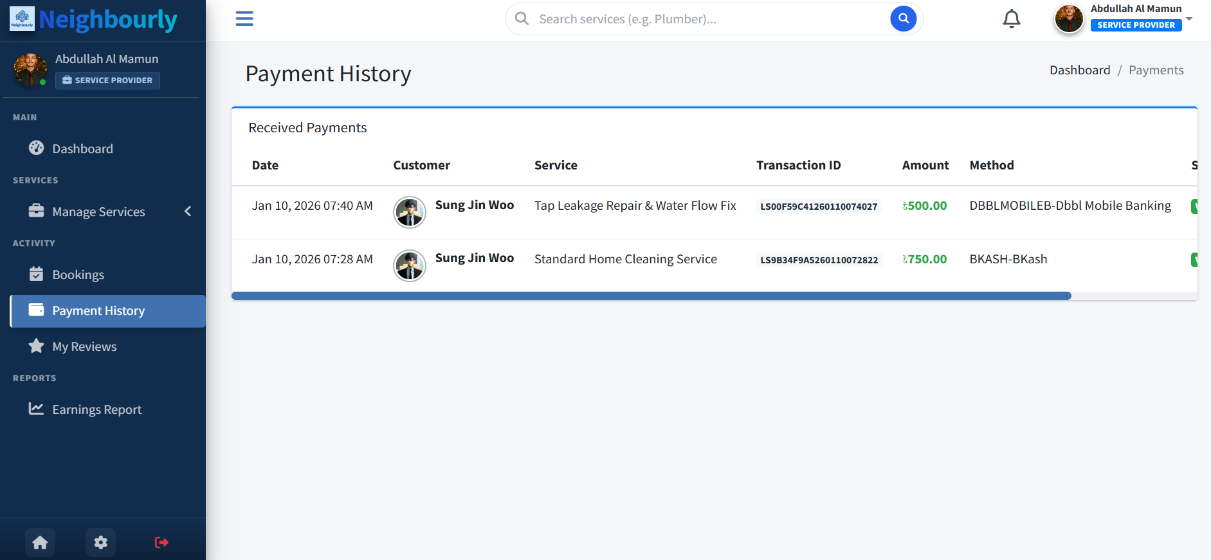
Figure 7.9: Providers Dashboard

Figure 7.10: Provider Active Services Page

Figure 7.11: Providers booking page

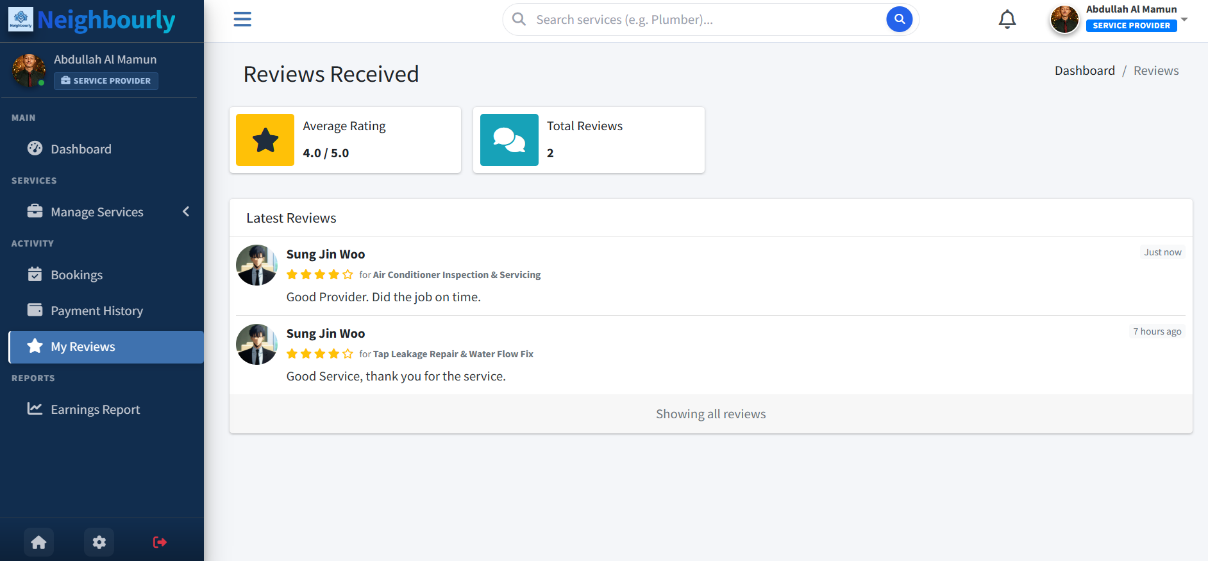
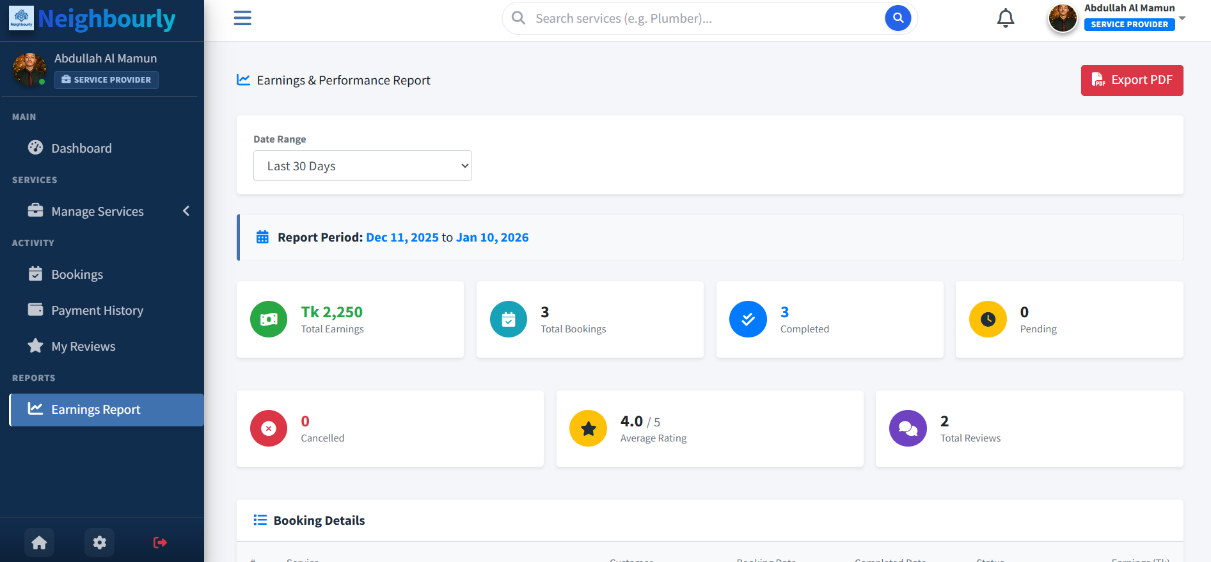
Figure 7.12: Providers Payment History Page

Figure 7.13: Providers My Review Page

Figure 7.14: Providers Earning Report page

##### 7.1.3 Users Features

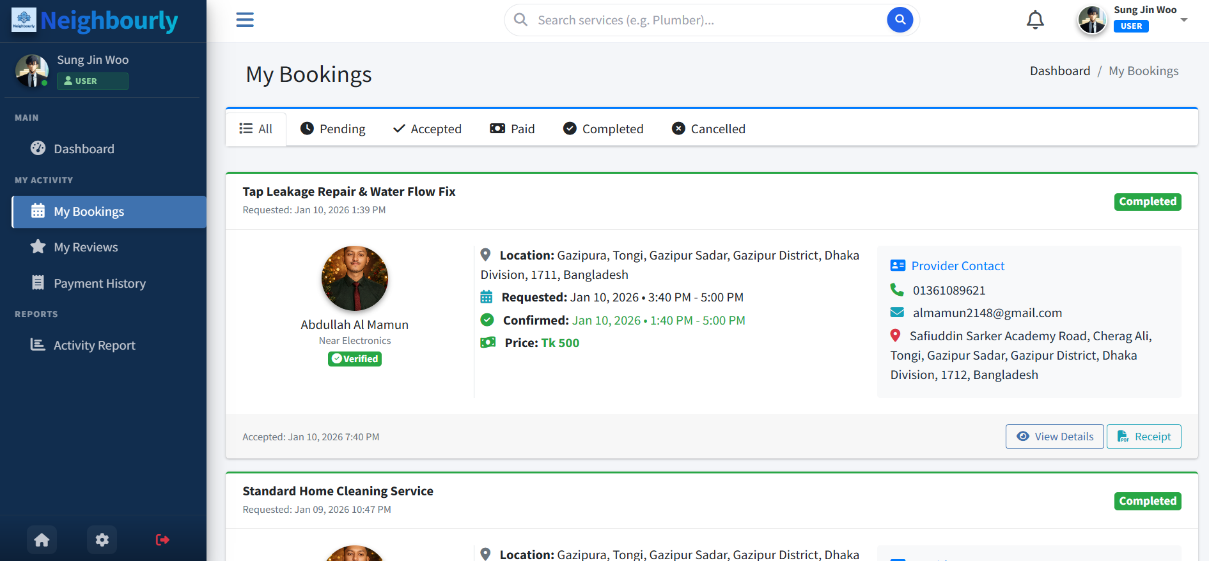
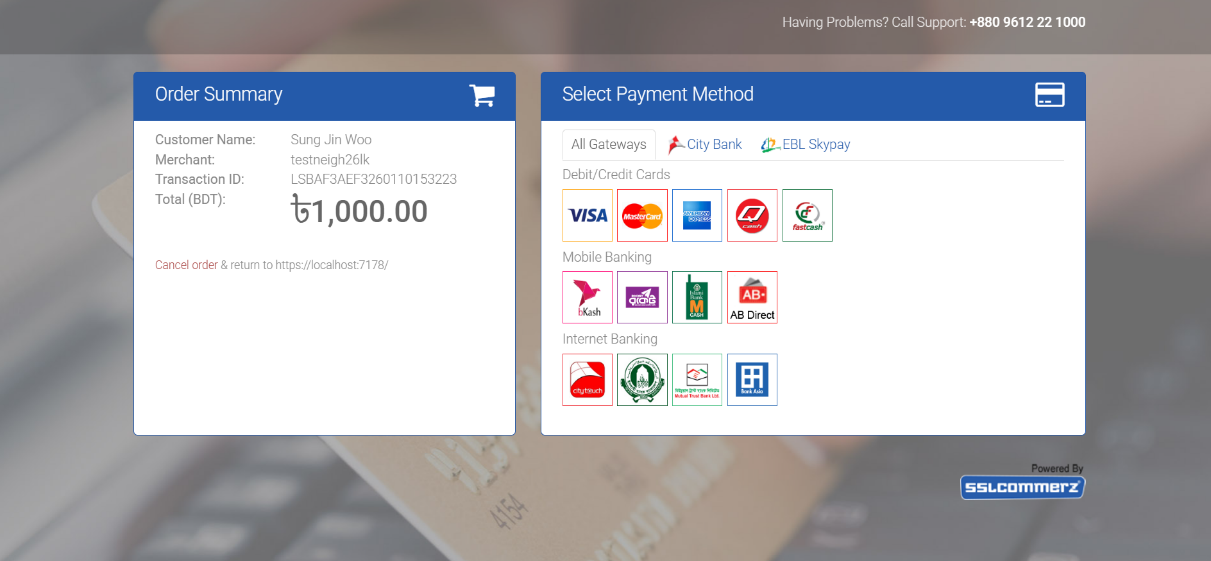
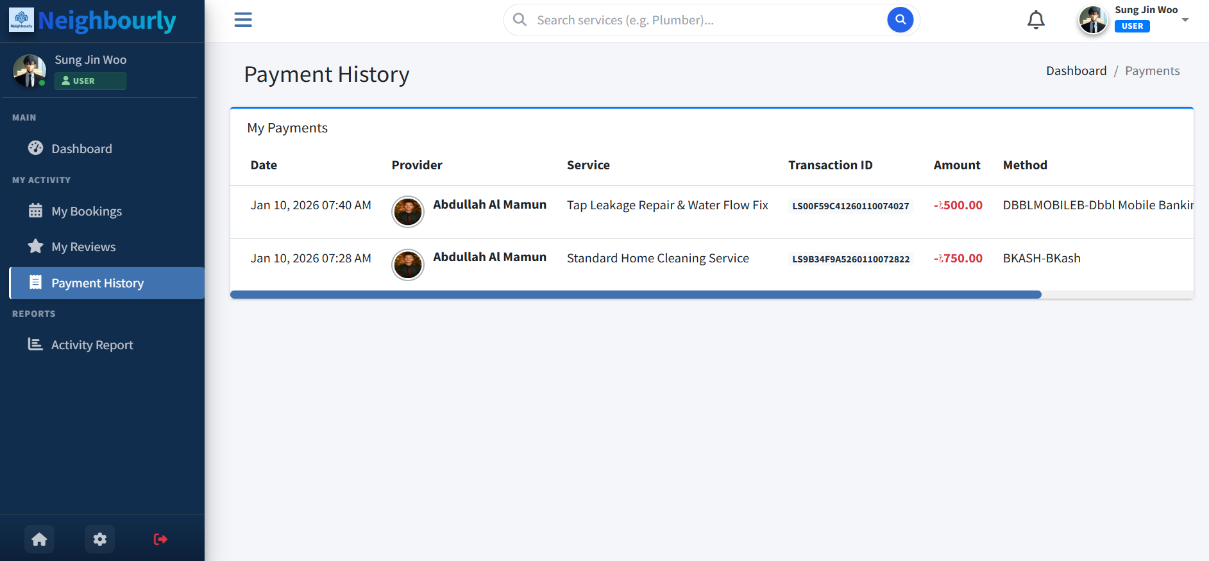
Figure 7.15: Users Dashboard

Figure 7.16: Users My booking page

Figure 7.17: SSLCommerz payment page

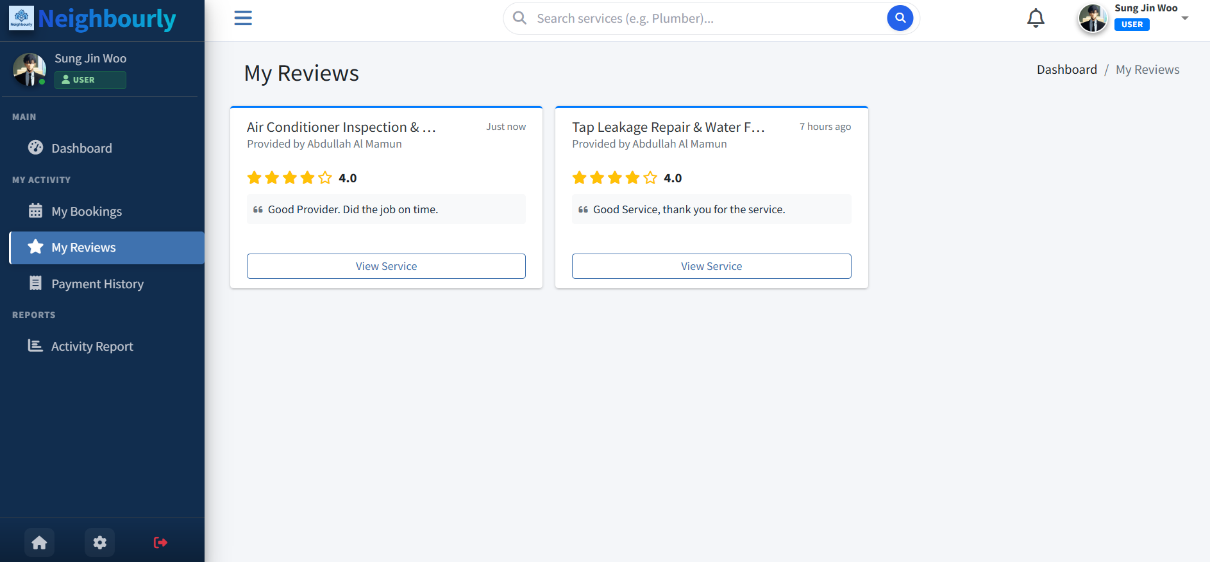
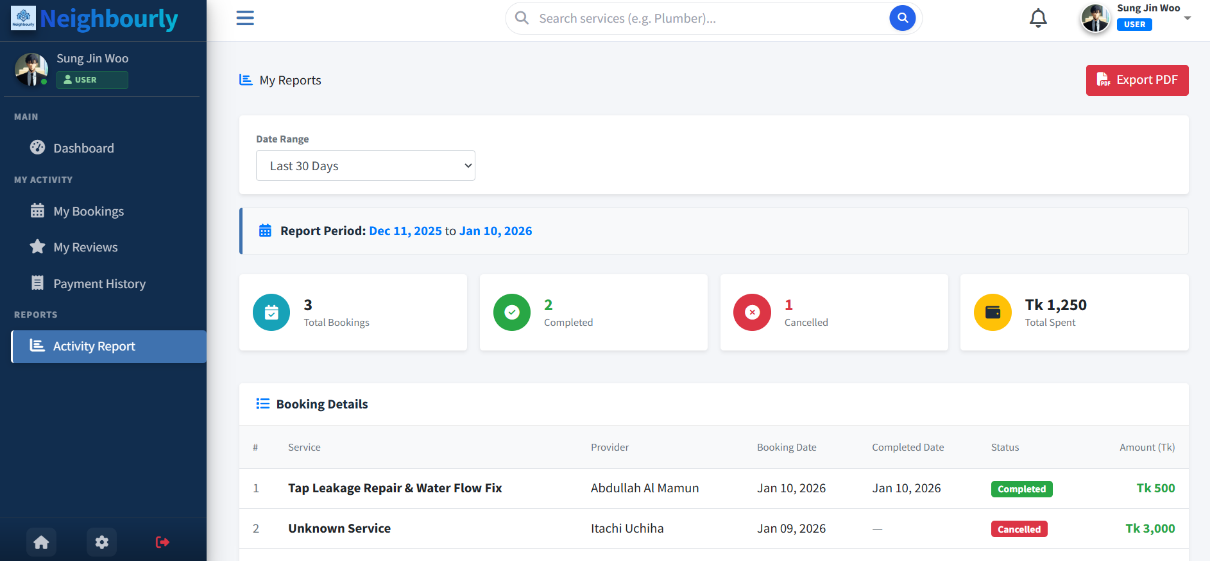
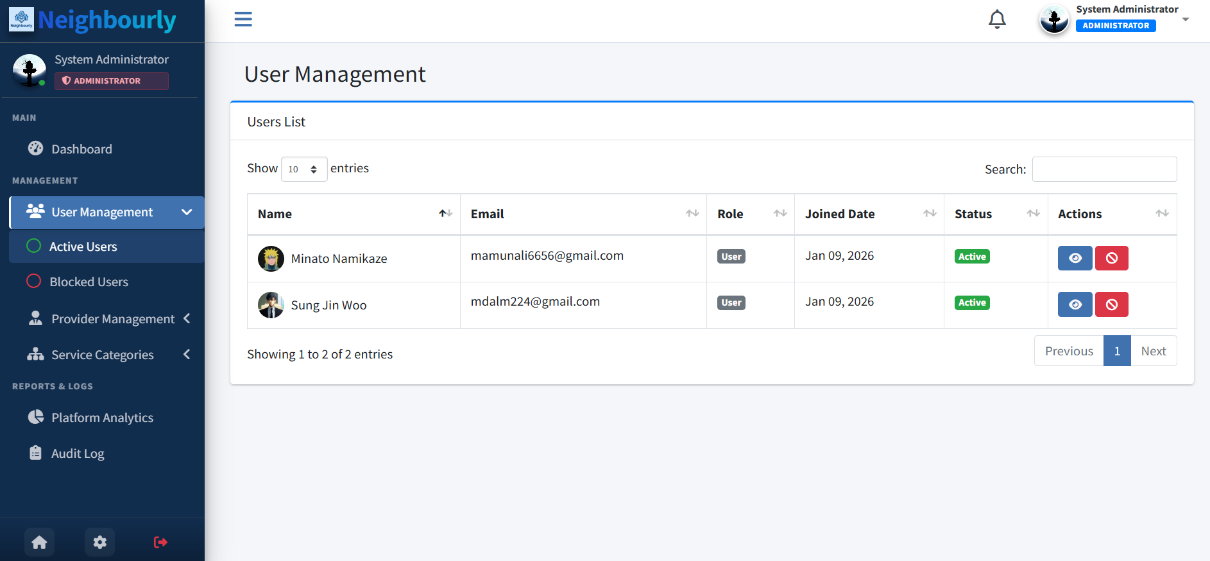
Figure 7.18: Users Payment History page

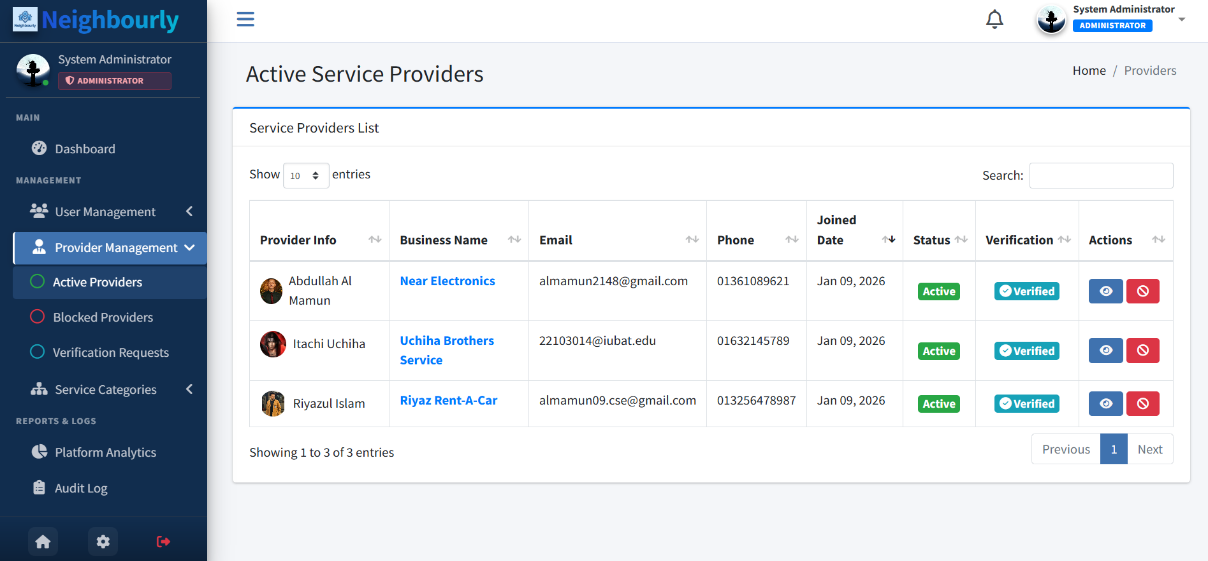
Figure 7.19: Users My Review page

Figure 7.20: Users Activity Report Page

##### 7.1.4 Admin Features

Figure 7.21: Admin Dashboard

Figures 7.22: User Management

Figure 7.23: Provider Management

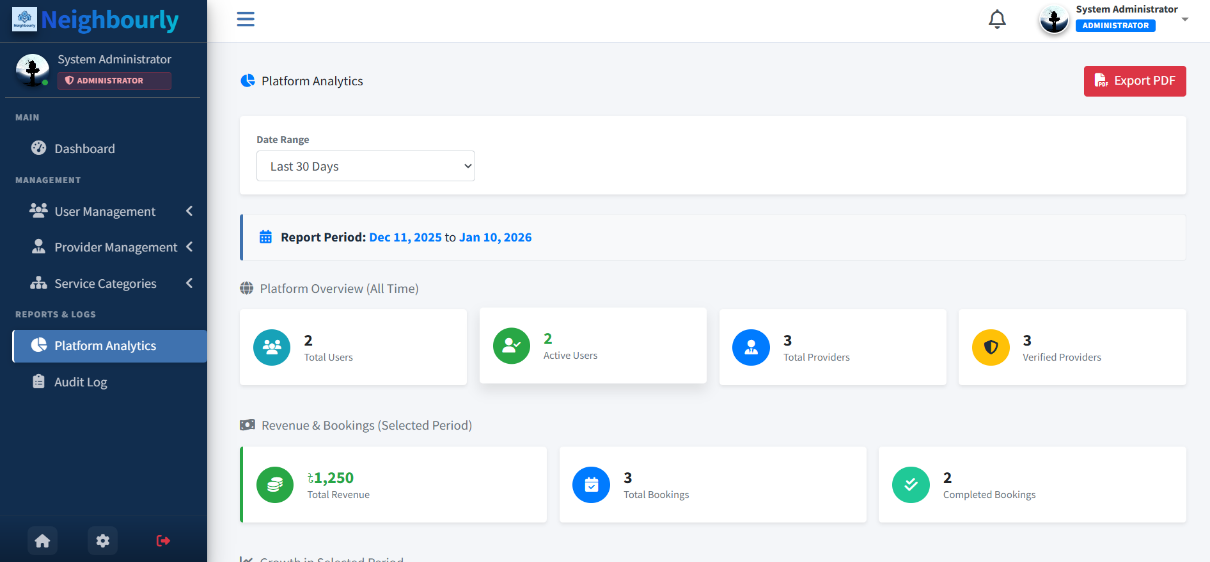
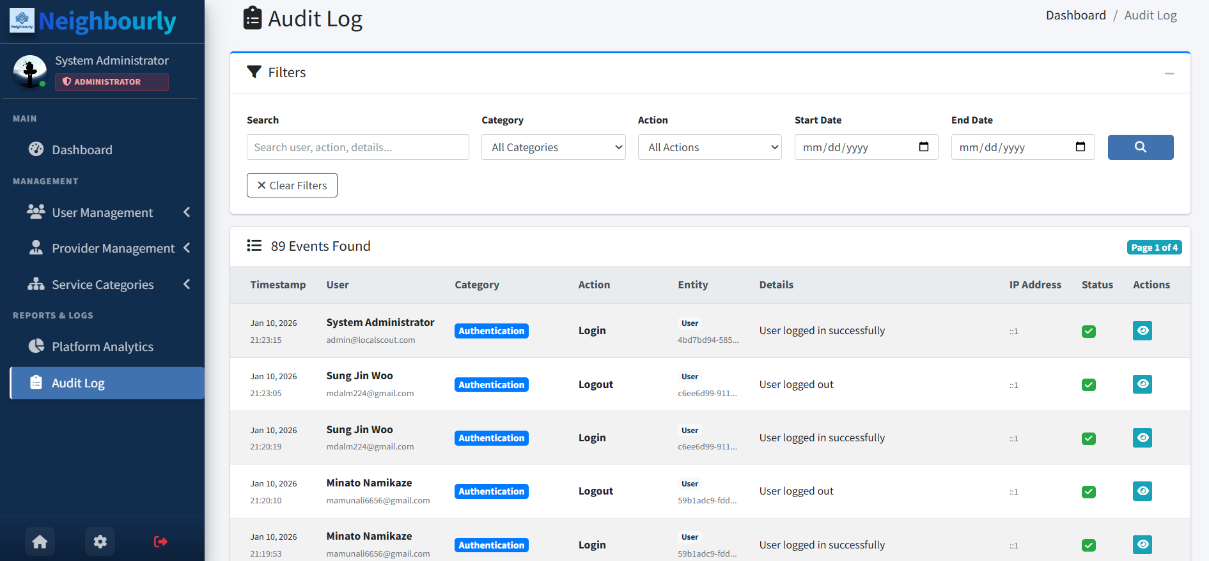
****Figure 7.24: Service Category Management

Figure 7.25: Platform Analytics & Reports

****Figure 7.26: Admin Audit Log

#### 7.2 Data Flow Diagram (DFD)

A **Data Flow Diagram (DFD)** is a graphical representation used to illustrate how data moves through an information system. It focuses on the flow of data between different system components rather than on program logic or technical implementation. DFDs help analysts and developers understand how data is received, processed, stored, and produced as output within the system.

A Data Flow Diagram is composed of four main elements: **external entities**, **processes**, **data stores**, and **data flows**. External entities represent users or external systems that interact with the system. Processes define the activities that transform input data into meaningful output. Data stores are repositories where data is saved for future use, such as databases or files. Data flows show the direction and type of data moving between entities, processes, and data stores.

DFDs are developed in multiple levels to provide increasing detail. The **Context Diagram (Level 0)** presents the entire system as a single process and shows its interaction with external entities. **Level 1 and Level 2 DFDs** further decompose the system into smaller, more detailed processes, making complex systems easier to understand.

Overall, Data Flow Diagrams are essential for system analysis and design, as they improve clarity, support requirement validation, and enhance communication among stakeholders.

##### 7.2.1 Context-Level DFD (Level 0)

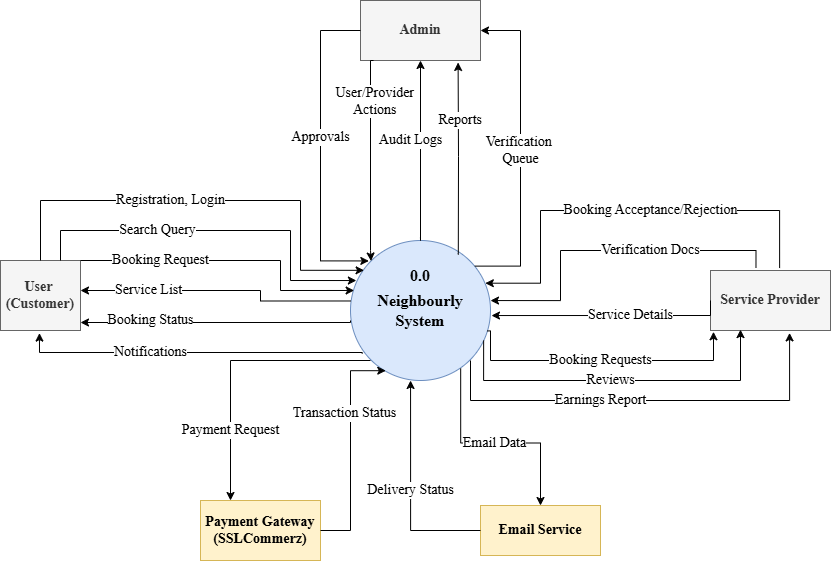
The Level 0 DFD represents the entire Neighbourly system as a single process interacting with external entities.

Figure 7.27: Context Level DFD (Level 0)

* **External Entities:**
  + **User:** Sends registration data, search queries, booking requests, and payments. Receives service results, booking confirmations, and notifications.
  + **Service Provider:** Sends business details, service listings, verification docs, and booking responses. Receives booking requests and earnings reports.
  + **Admin:** Sends verification approvals and blocking commands. Receives system reports and usage stats.
  + **Payment Gateway (SSLCommerz):** Receives payment requests. Sends transaction status.
  + **Map Service (LocationIQ):** Receives coordinates/addresses. Sends geolocation data.

##### 7.2.2 DFD Level 1

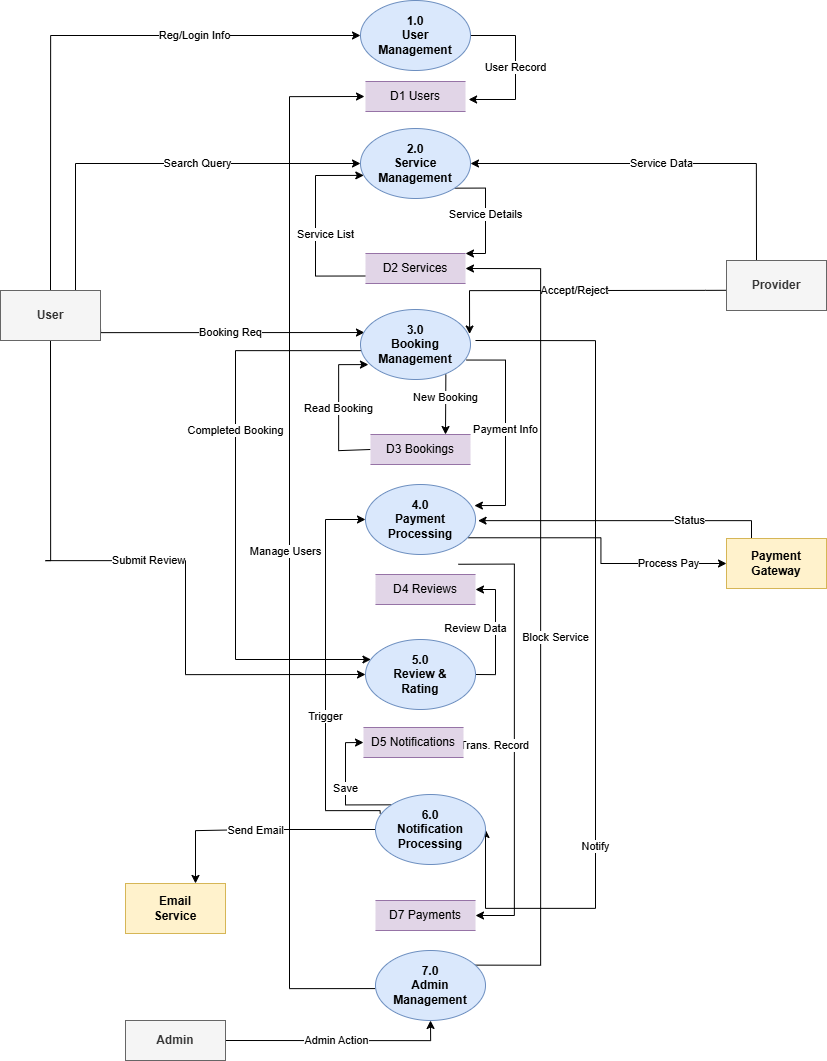
The Level 1 DFD decomposes the main system into its primary sub-processes:

Figure 7.28: Level 1 DFD

* **1.0 User Management:** Handles user registration, login, and maintenance of user records.
* **2.0 Service Management:** Manages service listings, service details, and processes user search queries.
* **3.0 Booking Management:** Handles service booking requests, provider accept/reject actions, and booking status updates.
* **4.0 Payment Processing:** Processes payment information, interacts with the payment gateway, and records transaction status.
* **5.0 Review & Rating:** Manages user reviews and ratings after service completion and stores feedback data.
* **6.0 Notification Processing:** Generates and sends booking, payment, and review notifications through the email service.
* **7.0 Admin Management:** Oversees system administration, including user management, service control, and payment monitoring.

##### 7.2.3 DFD Level 2

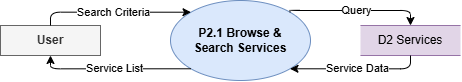
**P2.1 Browse & Search Services:** Allows users to search services using criteria and retrieves relevant service listings from the Services data store.

Figure 7.29: DFD Level 2 Process 1

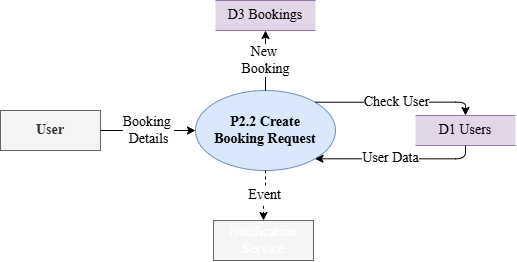
**P2.2 Create Booking Request:** Collects booking details from the user, validates user information, creates a new booking record, and triggers notification events.

Figure 7.30: DFD Level 2 Process 2

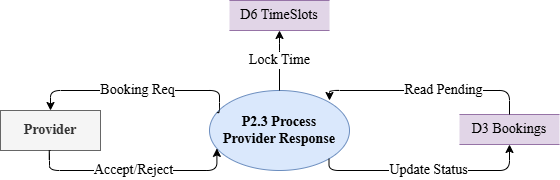
**P2.3 Process Provider Response:** Sends booking requests to providers, processes accept/reject responses, locks time slots, and updates booking status.

Figure 7.31: DFD Level 2 Process 3

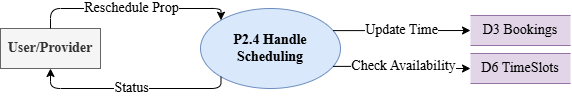
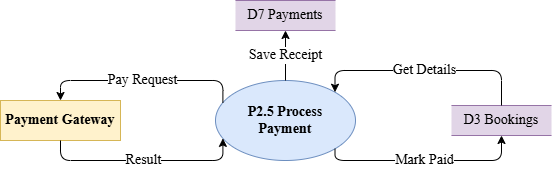
**P2.4 Handle Scheduling:** Manages rescheduling requests, checks time slot availability, updates booking time, and returns updated status to users and providers

Figure 7.32: DFD Level 2 Process 4

**P2.5 Process Payment:** Retrieves booking details, processes payment through the payment gateway, marks bookings as paid, and stores payment receipts.

Figure 7.33: DFD Level 2 Process 5

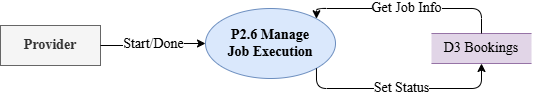
**P2.6 Manage Job Execution:** Tracks job starts and completion by providers, retrieves job information, and updates booking execution status.

Figure 7.34: DFD Level 2 Process 6

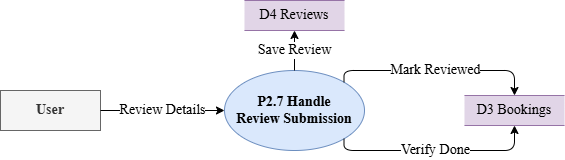
**P2.7 Handle Review Submission:** Accepts review details from users, verifies completed bookings, saves reviews, and marks bookings as reviewed.

Figure 7.35: DFD Level 2 Process 7

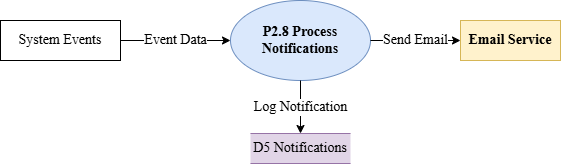
**P2.8 Process Notifications:** Processes system events, sends email notifications via the email service, and logs notification records.

Figure 7.36: DFD Level 2 Process 8

#### 7.3 Database Design

Database design plays a critical role in ensuring that the Neighbourly Service Booking System is efficient, scalable, and reliable. The database is designed to manage users, service providers, services, bookings, payments, reviews, notifications, and administrative operations. A relational database approach is adopted to maintain data integrity, minimize redundancy, and support complex relationships between entities such as users, services, bookings, and time slots.

The design supports core system functionalities including service discovery, booking management, payment processing, scheduling, reviews, verification, and notifications.

##### ****7.3.1 Normalization of Data Fields****

To ensure data consistency and eliminate redundancy, the Neighbourly database is normalized up to the **Third Normal Form (3NF)**.

1. **First Normal Form (1NF)**

* All tables contain atomic values.
* No repeating groups or multivalued attributes exist.
* Example: Image paths, working hours, and time slots are stored in separate fields or related tables instead of being embedded in a single column.

1. **Second Normal Form (2NF)**

* All non-key attributes fully depend on the primary key.
* Example: In the **Booking** table, attributes such as status, negotiated price, and requested date depend entirely on BookingId.

1. **Third Normal Form (3NF)**

* Non-key attributes are independent of each other.
* Example: User profile details (name, email, address) are stored only in the **ApplicationUser** table and not duplicated in Booking or Service tables.

This normalization approach improves data integrity, reduces update anomalies, and enhances database maintainability.

##### ****7.3.2 Entity Relationship (ER) Model****

The ER model of the Neighbourly system defines key entities and their relationships as follows:

* **ApplicationUser**: Represents both users and service providers. Stores personal information, location, verification status, and business details.
* **Service**: Represents services offered by providers. Each service belongs to a specific service category and is associated with one provider.
* **ServiceCategory**: Defines categories under which services are grouped (e.g., plumbing, cleaning).
* **Booking**: Represents service bookings made by users. A booking links a user, a service, a provider, and selected time slots.
* **ProviderTimeSlot**: Stores provider availability and ensures that booked slots are blocked during confirmed bookings.
* **Review**: Stores user feedback and ratings for completed bookings.
* **Payment**: Records payment transactions linked to bookings and users.
* **Notification:** Stores system-generated notifications sent to users.
* **VerificationRequest**: Manages provider verification submissions and admin review decisions.
* **CategoryRequest**: Allows providers to request new service categories.
* **AuditLog**: Tracks system activities for security and monitoring.
* **ServiceBlock & RescheduleProposal:** Handle service blocking by admin and booking rescheduling workflows.

The relationships include:

* One User to Many Bookings
* One Provider to Many Services
* One Booking to One Payment
* One Booking to One Review
* One Service to Many Bookings
* One Provider to Many Time Slots

##### 7.3.3 Database Table Structure

The following tables represent the core schema of the system:

Table 7.1: ApplicationUser Table

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| Id (PK) | NVARCHAR(450) | Unique user identifier (Identity) |
| FullName | NVARCHAR(MAX) | User's legal name |
| Email | NVARCHAR(256) | Unique email address |
| BusinessName | NVARCHAR(MAX) | Provider's business display name |
| Role | NVARCHAR(MAX) | User role (Admin, Provider, User) |
| Latitude | FLOAT | Location coordinate for search |
| Longitude | FLOAT | Location coordinate for search |
| IsVerified | BIT | Provider verification status |

Table 7.2: Service Table

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| ServiceId (PK) | UNIQUEIDENTIFIER | Unique service identifier |
| ServiceCategoryId (FK) | UNIQUEIDENTIFIER | Link to ServiceCategory |
| Id (FK) | NVARCHAR(450) | Link to Provider (ApplicationUser) |
| ServiceName | NVARCHAR(MAX) | Title of the service |
| MinPrice | DECIMAL(18,2) | Starting price for the service |
| IsActive | BIT | Visibility status |

Table 7.3: Booking Table

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| BookingId (PK) | UNIQUEIDENTIFIER | Unique booking reference |
| ServiceId (FK) | UNIQUEIDENTIFIER | Link to Service |
| UserId (FK) | NVARCHAR(450) | Link to Customer |
| ProviderId (FK) | NVARCHAR(450) | Link to Service Provider |
| Status | INT | Enum |
| NegotiatedPrice | DECIMAL(18,2) | Final agreed price |
| ConfirmedStartDateTime | DATETIME | Scheduled start time |

Table 7.4: ServiceCategory Table

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| ServiceCategoryId (PK) | UNIQUEIDENTIFIER | Unique category ID |
| CategoryName | NVARCHAR(MAX) | Name (e.g., Plumbing) |
| IconPath | NVARCHAR(MAX) | Path to category icon image |
| IsApproved | BIT | Admin approval status |

Table 7.5: Review Table

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| ReviewId (PK) | UNIQUEIDENTIFIER | Unique review ID |
| BookingId (FK) | UNIQUEIDENTIFIER | Link to completed booking |
| Rating | INT | Star rating (1-5) |
| Comment | NVARCHAR(MAX) | Text feedback |
| CreatedAt | DATETIME | Date of review |

Table 7.6: VerificationRequest Table

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| VerificationRequestId (PK) | UNIQUEIDENTIFIER | Unique request ID |
| ProviderId (FK) | NVARCHAR(450) | Link to Provider |
| DocumentPath | NVARCHAR(MAX) | Path to uploaded ID/License |
| Status | INT | Pending, Approved, Rejected |

Table 7.7: Notification Table

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| NotificationId (PK) | UNIQUEIDENTIFIER | Unique notification ID |
| UserId (FK) | NVARCHAR(450) | Recipient ID |
| Title | NVARCHAR(MAX) | Notification header |
| Message | NVARCHAR(MAX) | Notification body content |
| IsRead | BIT | Read/Unread status |

Table 7.8: AuditLog Table

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| AuditLogId (PK) | UNIQUEIDENTIFIER | Unique log ID |
| Action | NVARCHAR(MAX) | Description of activity |
| UserId | NVARCHAR(450) | User who performed action |
| IpAddress | NVARCHAR(MAX) | Source IP address |
| Timestamp | DATETIME | Time of occurrence |

Table 7.9: ProviderTimeSlot Table

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| TimeSlotId (PK) | UNIQUEIDENTIFIER | Unique slot ID |
| ProviderId (FK) | NVARCHAR(450) | Link to Provider |
| BookingId (FK) | UNIQUEIDENTIFIER | Link to Booking |
| StartDateTime | DATETIME | Slot start time |
| EndDateTime | DATETIME | Slot end time |

Table 7.10: RescheduleProposal Table

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| ProposalId (PK) | UNIQUEIDENTIFIER | Unique proposal ID |
| BookingId (FK) | UNIQUEIDENTIFIER | Link to Booking |
| ProposedBy | NVARCHAR(450) | Name of proposer |
| ProposedDate | DATETIME | Suggested new date |
| Status | INT | Pending, Accepted, Rejected |

Table 7.11: ServiceBlock Table

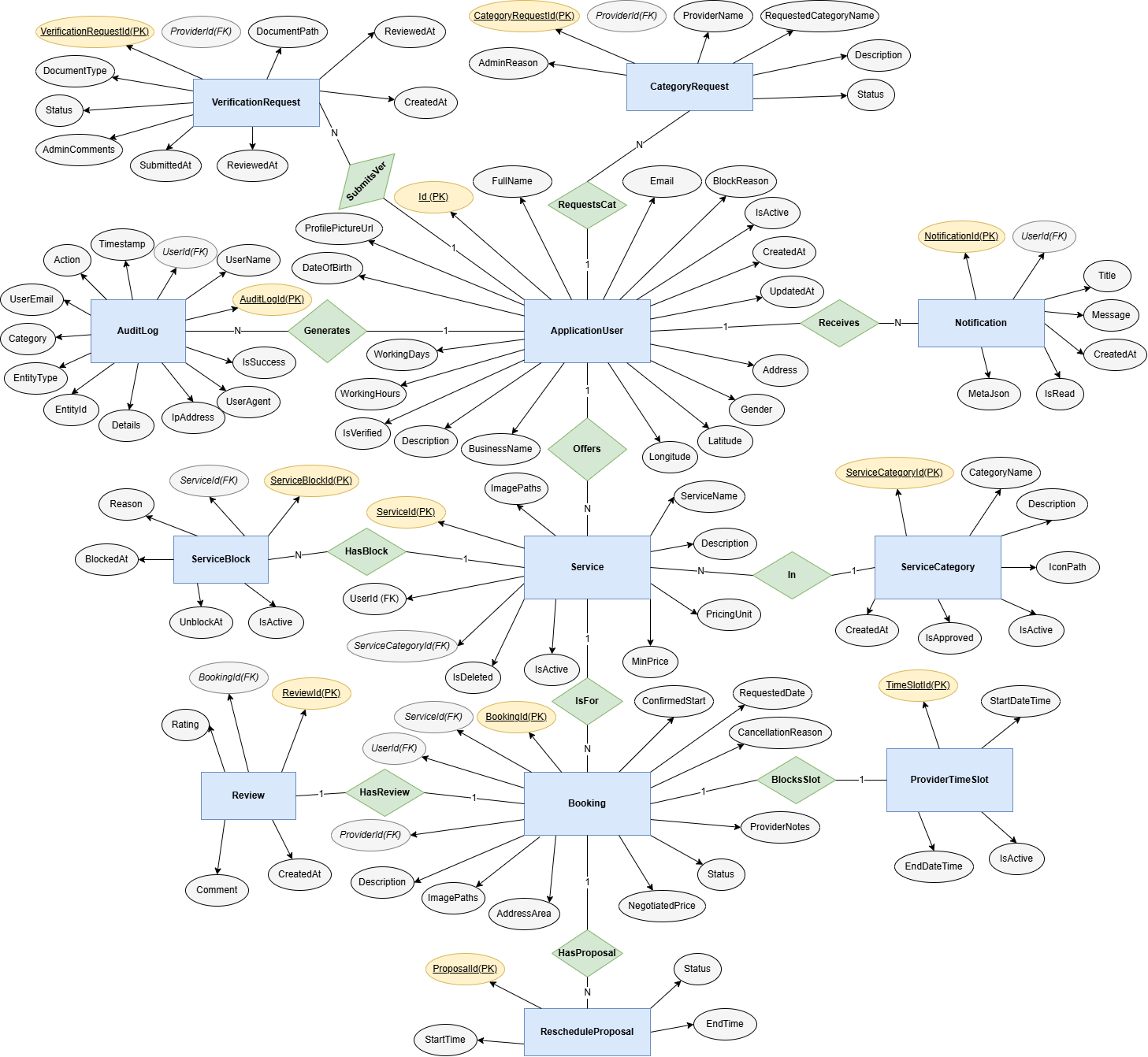
|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| ServiceBlockId (PK) | UNIQUEIDENTIFIER | Unique block ID |
| ServiceId (FK) | UNIQUEIDENTIFIER | Link to blocked service |
| Reason | NVARCHAR(MAX) | Reason for blocking |
| UnblockAt | DATETIME | Automatic unblock time |

Table 7.12: CategoryRequest Table

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| CategoryRequestId (PK) | UNIQUEIDENTIFIER | Unique request ID |
| ProviderId (FK) | NVARCHAR(450) | Requester ID |
| RequestedCategoryName | NVARCHAR(MAX) | Suggested name |
| Status | INT | Approval status |

#### 7.4 ER Diagram for Neighbourly System

The **Entity-Relationship (ER) Diagram** of the *Neighbourly* system visually represents the database structure and the relationships between key entities. It shows how users interact with services, including service creation and booking processes. The diagram also illustrates how bookings are associated with payments and reviews, ensuring accurate tracking of transactions and feedback. Additionally, it represents administrative functions such as user verification and service blocking, which support system security and quality control. Overall, the ER Diagram provides a clear understanding of data organization and relationships within the Neighbourly system, supporting efficient database design and system implementation.

Figure 7.37: ER Diagram of Neighbourly System

## **Chapter 8.**

## **System Quality and Testing**

#### 8.1 System Quality Management

System quality management ensures that the developed LocalScout System (Service Marketplace) meets its functional requirements, performance expectations, and user satisfaction goals. The quality of the system was managed throughout the project by applying structured testing methods, code reviews, and continuous validation of requirements.

##### 8.1.1 Software Quality Management Process

The quality management process followed these steps:

1. **Requirement Verification:** All Service Seeker and Service Provider requirements (service listing, extensive search, booking management, payment processing, cancellation, refund, reporting, etc.) were documented and confirmed with stakeholders before development (Mramba and Kaijage, 2018).
2. **Design Validation:** The database schema (Entity Relationship Diagram), interface design, and system architecture were validated to ensure consistency, normalization, and efficiency.
3. **Code Quality Assurance:** Coding standards (C# naming conventions, SOLID principles) were maintained and peer reviews were performed to avoid logical errors, redundancy, and maintainability issues (Pressman and Maxim, 2014).
4. **Testing:** Multiple types of testing, such as unit testing, integration testing, system testing, and user acceptance testing (UAT), were performed to verify the interaction between different modules (e.g., User, Booking, Service).
5. **Performance Evaluation:** Load testing was carried out to ensure the system could handle multiple simultaneous users searching for services and booking appointments without failures.
6. **Error Monitoring Debugging:** Bugs identified during testing (e.g., status transition errors, payment callback failures) were logged, categorized by severity, and fixed before final deployment.
7. **User Feedback Integration:** End users (Seekers and Providers) and admin testers were involved to provide feedback on usability, which was integrated into the final version.

##### 8.1.2 Software Test Cases

The system was tested using multiple scenarios to ensure functionality, security, and reliability. Ghezzi et al. (2002). Some key test cases are listed below:

Table 8.1: Test Cases for the System

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Feature Tested** | **Input / Action** | **Expected Result** | **Status** |
| **TC1** | User Registration | Enter new username, email, password, select role (Seeker/Provider) | New user account created, confirmation message shown, redirected to login | Passed |
| **TC2** | User Login | Enter valid username and password | Dashboard loads with appropriate options (Service Search for Seeker, Manage Services for Provider) | Passed |
| **TC3** | Invalid Login | Enter wrong password or unregistered email | Error message displayed ("Invalid credentials") | Passed |
| **TC4** | Service Booking | Select service category (e.g., Cleaner), choose provider, pick date/time, and confirm booking | Booking request created, status set to "PendingProviderReview", notification sent to provider | Passed |
| **TC5** | Payment Gateway | Process payment for confirmed booking using SSLCommerz sandbox | Payment successful, validation ID received, Booking status updates to "PaymentReceived" | Passed |
| **TC6** | Booking Cancellation | User cancels a confirmed booking before the cut-off time | Booking marked as "Cancelled", refund request initiated if applicable | Passed |
| **TC7** | Admin Report Generation | Admin requests monthly transaction report | PDF/Excel report generated with accurate booking and commission details | Passed |
| **TC8** | System Load Test | 100 users search for services simultaneously | System remains stable with no downtime; search results return within acceptable time | Passed |
| **TC9** | Security Test (SQL Injection) | Try inserting malicious SQL in search or login fields | System blocks attempt, sanitizes input, and shows generic error | Passed |
| **TC10** | Session Management | User logs out and presses back button | Session expired, requires re-login to access secured pages | Passed |

## **Chapter 9.**

## **Ethical Consideration**

#### 9.1 Ethical Considerations in the Software Development Process

Developing the **Neighbourly** platform involved navigating complex ethical landscapes, particularly given its nature as a location-based service connecting real-world individuals.

**1. Data Privacy and Security** In a hyper-local service marketplace, protecting user privacy is paramount. The system handles highly sensitive Personally Identifiable Information (PII), including precise home addresses, real-time geolocation coordinates (Latitude/Longitude), and identity verification documents (National IDs/Licenses). To ensure confidentiality and compliance with data protection standards:

* **Encryption:** Passwords are hashed using ASP.NET Core Identity's strong hashing algorithms. All data in transit is encrypted via HTTPS.
* **Access Control:** Strict Role-Based Access Control (RBAC) ensures that only authorized Admins can view verification documents, and Users can only see Provider details relevant to a booking.
* **Payment Security:** Financial data is never stored locally. The integration with **SSLCommerz** ensures that sensitive card details are handled by a PCI-DSS compliant gateway.
* **Minimization:** The system only requests location access when necessary for the search functionality. Sommerville (2016) Turban and Volonino (2010)

**2. Intellectual Property** The development of Neighbourly respected all intellectual property rights. The solution utilizes open-source technologies such as **ASP.NET Core 8.0**, **Entity Framework Core**, and **Tailwind CSS**, adhering strictly to their MIT or Apache 2.0 licenses. External APIs like **LocationIQ** and **Hugging Face** were integrated according to their Terms of Service, ensuring proper attribution and usage limits. The core business logic, database schema, and custom UI components remain the intellectual property of the development team and the client organization. Bass et al. (2012) Ghezzi et al. (2002)

**3. User Impact** The system significantly impacts the daily lives of local community members. For **Service Providers**, particularly freelancers and small business owners, the platform provides a critical source of income and visibility. For **Users**, it offers a convenient, stress-free way to solve household problems. The design prioritizes "Safety First"—features like Provider Verification and the ability to block/report users were implemented to prevent harm. Accessibility best practices were followed in the UI design to ensuring the app is usable by elderly or non-technical users seeking help. Sommerville (2011b) Sommerville (2011a)

**4. Bias and Fairness** Neighbourly is designed to be a meritocratic platform.

* **Algorithm Neutrality:** The search algorithms prioritize **Proximity** and **Rating**, ensuring that new providers in a local area have a fair chance of being discovered compared to established ones, provided they offer quality service.
* **AI Ethics:** The "Smart Description" feature powered by Hugging Face is carefully prompted to generate professional, neutral content, avoiding gendered or biased language when describing services.
* **Fair Access:** The platform does not discriminate based on the demographic or geographic background of the provider; anyone with valid skills and verification documents can join. Hevner and Chatterjee (2010)

**5. Responsibility to Stakeholders** The primary stakeholders are the Local Community (Customers), Service Professionals (Providers), and Platform Administrators.

* **Community:** The team is responsible for delivering a functional, bug-free experience where bookings are reliable.
* **Providers:** The system must accurately track bookings and earnings to ensure they are compensated fairly for their work.
* **Admins:** The platform empowers admins with tools to maintain order (e.g., verifying docs, moderating disputes) effectively. Regular feedback loops with beta testers helped align the software functionality with these diverse stakeholder needs. Hevner and Chatterjee (2010)

**6. Professional Responsibility** The development team adhered to the ACM/IEEE Software Engineering Code of Ethics. This specifically meant:

* **Honesty:** clearly communicating system limitations (e.g., location accuracy depends on browser hardware).
* **Competence:** ensuring the payment and security modules were implemented using industry-standard best practices to prevent financial loss.
* **Maintenance:** committing to resolving critical bugs (like potential booking conflicts) promptly to prevent user disruption. Hevner and Chatterjee (2010)

#### 9.2 Sustainability in the Software Development Process

Sustainability is integrated into the core architecture of Neighbourly, addressing environmental, economic, and social dimensions.

**1. Environmental Sustainability**

* **Paperless Workflow:** By digitizing the entire service lifecycle—from Booking requests to Digital Receipts and Invoices—Neighbourly significantly reduces paper waste involved in traditional contracting and billing.
* **Efficiency:** The verification process is entirely digital, eliminating the need for physical mail or travel for document submission.
* **Server Optimization:** Database queries are optimized using Entity Framework's specialized projection to fetch only necessary data, reducing CPU cycles and energy consumption in data centers.

**2. Economic Sustainability**

* **Gig Economy Support:** The platform directly fosters the local economy by lowering the barrier to entry for skilled workers (plumbers, tutors, technicians), allowing them to start businesses with minimal overhead costs.
* **Cost Efficiency:** For users, finding a local provider reduces need for long-distance travel, saving fuel and time costs.
* **Scalability:** The modular "Clean Architecture" design allows the system to scale to new cities without complete redevelopment, ensuring long-term economic viability of the software asset itself. Pressman and Maxim (2014).

**3. Social Sustainability**

* **Trust Building:** By implementing a verification and review system, Neighbourly builds social capital and trust between strangers in a community.
* **Inclusivity:** The platform provides equal opportunity for independent workers who might not have the capital for traditional advertising store-fronts.
* **Employment:** Long-term, the platform acts as an engine for local employment, helping skilled individuals find consistent work (Turban and Volonino, 2010).

Figure 9.1: Sustainability Considerations in Software Development

## **Chapter 10.**

## **Conclusion**

#### 10.1 Brief Overview of the Project

The **Neighbourly** system was designed to provide a comprehensive digital marketplace for connecting local community members with skilled service providers. It serves as a modern bridge between demand and supply in the gig economy, replacing fragmented methods of finding help (like word-of-mouth or unverified directories) with a centralized, verified platform. By introducing features such as geolocation-based search, AI-powered service descriptions, negotiation-friendly booking workflows, and secure digital payments, the project aims to professionalize local services. On the administrative side, the system offers robust tools for provider verification and platform moderation, ensuring a safe ecosystem for all users.

#### 10.2 Proposed System Benefits

The developed system offers a range of benefits for both End-Users and Service Providers:

1. **Localized Convenience:** Users can instantly find help (plumbers, electricians, tutors) within their specific vicinity using geolocation technology.
2. **Economic Empowerment:** Provides a professional platform for freelancers and small business owners to showcase their portfolios and earn income without high marketing costs.
3. **Trust & Safety:** The mandatory verification workflow and rating system ensures that users interact with legitimate, trusted professionals.
4. **Operational Efficiency:** Automated booking management, invoicing, and digital receipts reduce the administrative burden on providers.
5. **Secure Transactions:** Integration with **SSLCommerz** provides a regulated and secure environment for financial exchanges, minimizing fraud risk.
6. **AI Assistance:** Integration with **Hugging Face** lowers the barrier for providers to create professional-looking service listings by auto-generating content.

#### 10.3 Limitations of the Project

While the system meets its primary objectives, several limitations remain, which highlight areas for improvement:

1. **Browser-Based Location:** The system relies on browser APIs for geolocation, which can vary in accuracy depending on the user's device hardware and network.
2. **Internet Dependence:** As a cloud-hosted web application, it requires a stable internet connection, which may limit accessibility in areas with poor 4G/Broadband coverage.
3. **Payment Gateway Sandbox:** The current implementation uses the SSLCommerz Sandbox environment; a live production deployment would require rigorous compliance auditing and legal agreements.
4. **Lack of Native Mobile App:** The system is a Responsive Web App (RWA), which lacks native mobile features like push notifications for real-time booking updates.
5. **AI Rate Limits:** The reliance on the free tier of the Hugging Face Inference API means the "Smart Description" feature could face rate limits under heavy load.
6. **No Real-Time Chat:** Communication is currently handled via Booking Notes and Reschedule Proposals; a live chat system is missing for immediate coordination.

These limitations emphasize that while the project demonstrates strong architectural principles and functional value, further refinement is needed for a commercial-grade launch.

#### 10.4 Practicum and Its Value

The practicum offered meaningful real-world learning opportunities by connecting theoretical knowledge with practical application. The experience included:

1. **Technical Mastery:** Deepened expertise in the **ASP.NET Core 8.0** ecosystem, **Entity Framework Core**, and **Clean Architecture** patterns.
2. **Third-Party Integration:** Gained practical experience integrating diverse external services such as **LocationIQ** (Maps), **SSLCommerz** (FinTech), and **Hugging Face** (AI).
3. **Complex Logic Implementation:** Solved challenging problems related to booking state machines, time-slot negotiation, and concurrency handling.
4. **Professional Development:** Reinforced the importance of code reviews, Git version control, and adhering to strict project documentation standards. Overall, the practicum bridged the gap between academic concepts and industry-standard software engineering practices.

#### 10.5 Future Plan

Future development of the system will focus on addressing the identified limitations and preparing it for larger-scale deployment. Planned improvements include:

1. **Native Mobile Application:** Developing cross-platform apps (using .NET MAUI or React Native) to leverage native push notifications and background location tracking.
2. **Real-Time Chat Module:** Implementing a WebSocket-based (SignalR) chat system to allow users and providers to communicate instantly within the app.
3. **Advanced Recommendation Engine:** Using Machine Learning to suggest services to users based on their past booking history and search patterns.
4. **Multiple Payment Options:** Expanding beyond SSLCommerz to include bKash, Nagad, and Stripe APIs for broader payment accessibility.
5. **Infrastructure Scalability:** Migrating the monolithic architecture to microservices (separating Auth, Booking, and Notification services) to handle thousands of concurrent users.
6. **Enhanced AI Features:** Implementing image recognition to automatically categorize portfolio photos uploaded by providers.

These enhancements will help transition Neighbourly from a functional prototype to a market-ready platform capable of serving a city-wide user base.

## **Bibliography**

FORTUNE (2024). Hyperlocal Services Market Size, Share | Growth Report [2032]. [online] Fortunebusinessinsights.com

Folio3 (2025). A Guide to Hyperlocal Marketplaces and Ecommerce. [online] Folio3 Ecommerce.

Huggingface.co. (2025). Inference Providers. [online] Available at: https://huggingface.co/docs/inference-providers/en/index.

location-iq (2025). GitHub - location-iq/locationiq-csharp-client. [online] GitHub. Available at: https://github.com/location-iq/locationiq-csharp-client.

Mramba, B.P. and Kaijage, S.F. (2018). Design of an Interactive Mobile Application for Maternal, Neonatal and Infant Care Support for Tanzania. Journal of Software Engineering and Applications, 11(12), pp.569–584.

NDepend (2024). Clean Architecture in ASP.NET Core - NDepend Blog. [online] NDepend Blog. Available at: https://blog.ndepend.com/clean-architecture-for-asp-net-core-solution.

Raza, J. (2025). Building a Simple Text Generation App with Hugging Face Model APIs. [online] Medium. Available at: https://medium.com/@jhonraza/building-a-simple-text-generation-app-with-hugging-face-model-apis.

Shipturtle. (2025). Marketplace Trends for 2025 You Should Not Miss. [online] Available at: https://www.shipturtle.com/blog/online-marketplace-industry-trends.

www.rst.software. (n.d.). 6 common geolocation app development challenges and how to overcome them | RST Software. [online] Available at: https://www.rst.software/blog/6-common-geolocation-app-development-challenges-and-how-to-overcome-them.

Pressman, R. S. and Maxim, B. R. (2014), Software Engineering: A Practitioner’s Approach, 8th edn, McGraw-Hill Education, New York.

Ghezzi, C., Jazayeri, M. and Mandrioli, D. (2002), Fundamentals of Software Engineering, 2nd edn, Prentice Hall, Upper Saddle River.

Sommerville, I. (2011a), Requirements Engineering, 9th edn, Pearson.

Sommerville, I. (2011b), Requirements Engineering: From System Goals to UML Models to Software Specifications, Pearson Education, London.

Sommerville, I. (2016), Software Engineering, 10th edn, Pearson, Boston.

Bass, L., Clements, P. and Kazman, R. (2012), Software Architecture in Practice, 3rd edn, Addison-Wesley, Boston.

Turban, E. and Volonino, L. (2010), Information Technology for Management, 7th edn, John Wiley & Sons, Hoboken.

Hevner, A. R. and Chatterjee, S. (2010), Design Research in Information Systems: Theory and Practice, Springer, New York.