Faculty of Engineering Department of Informatics Engineering- Software and Information Systems Engineering

الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

Home Care System

senior Project 2- Completed the requirements for obtaining a bachelor's degree in Informatics Engineering - Software Engineering and Information Systems

Prepared by

Mohammad Alhendi **Kasem Almaleh** **Mohammad Nour**

Supervised By

Dr. Riad Sonbol

Eng. Anas Abdulaziz

January 2026

كلية الهندسة
قسم الهندسة المعلوماتية
هندسة البرمجيات ونظم المعلومات

نظام العناية في المنزل

مشروع تخرج 2 – قدم استكمالا لمتطلبات الحصول على درجة البكالوريوس في هندسة المعلوماتية– هندسة البرمجيات ونظم المعلومات

إعداد

محمد إبراهيم نور كاسم المليح محمد الهندي

إشـراف

الدكتور رياض سنبل

المهندس أنس عبد العزيز

2026 يناير

Supervisor certification

I certify that the preparation of this project entitled Home Care System, prepared by **Kasem Almaleh**, **Mohammad Alhendi**, and **Mohammad Noor** and was made under my supervision at Department of Software and Information Systems Engineering in partial of Bachelors of Degree the for Requirements the of fulfillment Software and Information Systems Engineering.

Name: Dr. Riad Sonbol Signature:……..…. Date: ….……

Name: Eng. Anas Abdulaziz Signature:……..…. Date: ….……

Abstract

This project introduces a comprehensive digital system for home care that aims to simplify and improve the process of requesting and managing home maintenance services. Traditional methods suffer from a lack of transparency, efficiency, and ease of use, which negatively affects customer satisfaction and service provider engagement.

The proposed system allows customers to submit maintenance requests, upload images of issues, and compare service offers. It also supports features such as role-based access control, notifications, and ratings. It enables service providers to enhance their visibility based on the quality of their work.

By fully digitizing the workflow, the system enhances reliability, promotes high-quality service delivery, and increases customer satisfaction and trust.

الملخص

يقدّم هذا المشروع نظامًا رقميًا متكاملًا للعناية بالمنزل، يهدف إلى تبسيط وتحسين عملية طلب خدمات الصيانة المنزلية وإدارتها. تعاني الأساليب التقليدية من ضعف في الشفافية والكفاءة وسهولة الاستخدام، مما ينعكس سلبًا على رضا العملاء وتفاعل مقدمي الخدمات.

يتيح النظام المقترح للعملاء تقديم طلبات الصيانة، تحميل صور للمشكلات، ومقارنة العروض المقدّمة. كما يدعم ميزات متعددة تشمل التحكم في الوصول بناءً على الأدوار، الإشعارات، والتقييمات. ويساعد مقدمي الخدمات على تعزيز ظهورهم بناءً على جودة خدماتهم.

من خلال رقمنة دورة العمل بالكامل، يسهم النظام في تعزيز الموثوقية، رفع جودة الخدمات المقدمة، وزيادة رضا وثقة العملاء.

Table of Content

SUPERVISOR CERTIFICATION **3**

ABSTRACT **4**

الملخص **5**

CHAPTER 1 INTRODUCTION **12**

1. **INTRODUCTION** 13
2. **PROBLEM STATEMENT** 13
3. **PROJECT OBJECTIVE** 14
4. **PROPOSED SYSTEM** 14
5. **REPORT ORGANIZATION** 15
6. **REPORT STRUCTURE AND PURPOSES** 15
7. **SUMMARY** 17

CHAPTER 2 FUNDAMENTAL CONCEPTS AND LITERATURE REVIEW **18**

1. **INTRODUCTION** 19
2. **LITERATURE REVIEW** 19
2.1 ***Maintenance / Home Care Platforms*** *19*
2.2 ***Trust and Reputation Systems*** *20*
2.3 ***Intelligent Request Handling*** *21*
2.4 ***Multi-Task Learning for Automated Assessment*** *22*
2.5 ***Support and Automation*** *22*
2.6 ***Proposed System*** *22*

CHAPTER 3 PROJECT MANAGEMENT **26**

1. **INTRODUCTION:** 27
2. **PROJECT CHARTER** 27
3. **THE SOW DOCUMENT:** 30
4. **GANTT CHART:** 33
5. **RISK MANAGEMENT** 35
6. **SUMMARY** 37

CHAPTER 4 SYSTEM ANALYSIS **38**

1. **INTRODUCTION** 39
2. **PROJECT TIMELINE** 39
3. **SOFTWARE REQUIREMENT SPECIFICATION DOCUMENT (SRS)** 41
4. **REQUIREMENTS MODELING** 62
5. **INITIAL TEST CASES** 99
6. **INITIAL REQUIREMENT TRACEABILITY MATRIX (RTM)** 107

CHAPTER 5 SYSTEM DESIGN **110**

1. **INTRODUCTION** 111
1. **SYSTEM ARCHITECTURE** 111

2. MICROSERVICES BREAKDOWN 112
1. User Mgmt. Service *112*
2. Service Management & Request *114*
3. Notification Service *119*
4. Rating, Review, and Trust Scoring Service *122*
3. DESIGN PATTERNS 125
1. Repository Design Pattern *125*
2. Façade Design Pattern: Simplifying the Interface *126*
3. Strategy Pattern: Runtime Flexibility *126*
4. Singleton Pattern: Efficient Resource Management *127*
5. Adapter Pattern: Interface Translation *128*
6. Observer Pattern: Decoupled Event Handling *128*
7. Pattern Summary Table *129*
4. COMMUNICATION BETWEEN COMPONENTS 129
1. External Communication: Frontend–Backend (North-South Traffic) *129*
2. Internal Communication: Backend–Backend (East-West Traffic) *130*
3. Comparison of Communication Flow *131*
5. MULTI-TASK DEEP LEARNING FOR AUTOMATED HOME SERVICE CLASSIFICATION AND COST ESTIMATION 131
6. COMPONENT DIAGRAM 131
7. ERD DIAGRAMS 133

CHAPTER 6 MULTI-TASK DEEP LEARNING FOR AUTOMATED HOME SERVICE CLASSIFICATION AND COST ESTIMATION **138**

ABSTRACT 139
1. INTRODUCTION 139
2. LITERATURE REVIEW 140
2.1 Image-Based Cost Estimation *140*
2.2 Multi-Task Learning (MTL) in Computer Vision *140*
2.3 Transfer Learning with EfficientNet *140*
2.4 Regression from Images *141*
2.5 Learning with Limited Data *141*
3. DATASET DESCRIPTION 141
4. SYSTEM ARCHITECTURE 142
4.1 Backbone & Feature Extraction *142*
4.2 Auxiliary Feature Integration *142*
4.3 Multi-Task Heads *142*
5. MATHEMATICAL BACKGROUND 143
5.1 Loss Functions *143*
6. MODEL VERSIONS & ABLATION STUDY 143
7. EXPERIMENTS & RESULTS 145

7.1 Experimental Setup *145*
7.2 Cross-Validation Results (Version 5) *145*
7.3 Final Evaluation on Unseen Test Data *145*
8. DISCUSSION 146
9. RECOMMENDATIONS & FUTURE WORK 146
10. CONCLUSION 147

CHAPTER 7 PRACTICAL IMPLEMENTATION **148**

1. INTRODUCTION 149
2. USED TOOLS 149
2.1 Core Development Frameworks *149*
2.2 Workflow Orchestration & AI Stack *149*
2.3 Asynchronous Processing & Messaging *150*
2.4 Performance & Load Testing Suite *150*
2.5 Infrastructure & API Documentation *151*
2.6 Collaboration & IDE *151*
2.7 Summary of Tooling Purpose *152*
3. SYSTEM INTERFACES 152
4. SYSTEM TESTING, VALIDATION, AND PERFORMANCE ENGINEERING 161
4.1 Overview and Testing Strategy *161*
4.2 Functional Testing: Logic Verification *162*
4.3 Performance Engineering: The Optimization Journey *162*
4.4 Scalability Validation (Stress Testing) *167*
4.5 Challenges & Quality Assurance *167*
4.6 Conclusion *167*
5. TEST CASES EXECUTION 168
6. LAST VERSION OF RTM 175

CHAPTER 8 CONCLUSION & FUTURE WORK **180**

1. INTRODUCTION 181
2. FUTURE WORK 181
3. CONCLUSION 182

REFERENCES **183**

LIST OF ABBREVIATIONS **184**

List of Figures

Figure 1 gantt chart – senior I 33
Figure 2 gantt chart - senior I 33
Figure 3 gantt chart - senior II 34
Figure 4 gantt chart - senior II 35
Figure 5 gantt chart - senior II 35
Figure 6 use case diagram 62
Figure 7 register sequence 63
Figure 8 register activity 64
Figure 9 log in sequence 65
Figure 10 log in activity 66
Figure 11 account info mgmt. sequence 68
Figure 12 account info mgmt. activity 69
Figure 13 submit custom service offer sequence 70
Figure 14 submit custom service offer activity 71
Figure 15 view available services sequence 72
Figure 16 view available services activity 72
Figure 17 submit service request manually sequence 74
Figure 18 submit service request manually activity 75
Figure 19 create service request via ai-assistant sequence 76
Figure 20 create service request via ai-assistant activity 77
Figure 21 create service request via chatbot sequence 78
Figure 22 create service request via chatbot 79
Figure 23 update & accept service request draft sequence 80
Figure 24 update & accept service request draft activity 81
Figure 25 view requests sequence 82
Figure 26 view requests activity 83
Figure 27 Review and Accept Service Provider Offer sequence 85
Figure 28 Review and Accept Service Provider Offer activity 85
Figure 29 submit rating & review sequence 87
Figure 30 submit rating & review activity 87
Figure 31 view provider ratings sequence 89
Figure 32 view provider ratings activity 89
Figure 33 edit / delete own review sequence 91
Figure 34 edit / delete own review activity 91
Figure 35 admin moderation sequence 92
Figure 36 admin moderation activity 93
Figure 37 provider notification sequence 94
Figure 38 provider notification sequence 95
Figure 39 provider average rating sequence 96
Figure 40 provider average rating activity 97
Figure 41 Class Diagram 98
Figure 42 user management architecture 113
Figure 43 service request architecture 116
Figure 44 Detailed Architecture of the AI Automation Layer 117
Figure 45 Architectural Diagram of the Notification Service 121

Figure 46 component diagram - rating & review....124
Figure 47 Component Diagram....132
Figure 48 Erd- User Mgmt.....133
Figure 49 Erd- Service Mgmt. & Request....134
Figure 50 Erd- Notification Service....135
Figure 51 rating & review service....136
Figure 52 Erd- Notification with Service Request....137
Figure 53 home page....152
Figure 54 how it works page....153
Figure 55 Request and offers page – customer....153
Figure 56 Featured Services page....154
Figure 57 Sign up page....155
Figure 58 Login page....155
Figure 59 Update profile page....156
Figure 60 Account settings page....156
Figure 61 Create request - s1....156
Figure 62 Create request – s2....157
Figure 63 Create request – s3....158
Figure 64 Create request – s4....158
Figure 65 Create request – s5....159
Figure 66 Requests list page....159
Figure 67 Provider dashboard....160
Figure 68 Request details page - provider....160
Figure 69 Offers list page - Provider....161
Figure 70 Baseline load test performance metrics.....164
Figure 71 Before Query Optimization....164
Figure 72 Impact of Query Optimization.....165
Figure 73 k6 network summary report for Phase 3. Before optimization....166
Figure 74 Payload & Network Efficiency....166

List of Tables

Table 1 Comparative Analysis of Similar Systems and the Target System 24
Table 2 Roles and Responsibilities 29
Table 3 Risk management 37
Table 4 register 62
Table 5 log in 65
Table 6 account info mgmt. 67
Table 7 submit custom service offer 69
Table 8 view available services 71
Table 9 submit service request manually 73
Table 10 create service request via ai-assistant 75
Table 11 create service request via chatbot 77
Table 12 update & accept service request draft 80
Table 13 view requests 82
Table 14 Review and Accept Service Provider Offer 84
Table 15 submit rating & review 86
Table 16 view provider ratings 88
Table 17 edit / delete own review 90
Table 18 admin moderation 92
Table 19 provider notification 94
Table 20 provider average rating update 96
Table 21 service management test cases 99
Table 22 service request test cases 100
Table 23 offer management test cases 101
Table 24 notification service test cases 102
Table 25 user management service test cases 103
Table 26 rating & review service test cases 105
Table 27 Initial Requirement Traceability Matrix (RTM) 107
Table 28 Pattern Summary Table 129
Table 29 comparison of communication flow 131
Table 30 service management test cases execution 168
Table 31 service request test cases execution 168
Table 32 offer management test cases execution 170
Table 33 notification service test cases execution 170
Table 34 user management service test cases execution 171
Table 35 rating & review service test cases 173
Table 36 Last Version of RTM 175
Table 37 list of abbreviations 184

Chapter 1 Introduction

1. Introduction

This chapter introduces the core elements of the **Home Care System** project, highlighting the problems in traditional service platforms, the objectives of the proposed solution, and the innovative features that distinguish it. The system is designed as a smart, role-based platform that not only manages service requests and offers but also integrates advanced features such as trust scoring, image-to-text request generation, in-app communication, and intelligent support. Together, these enhancements establish a strong foundation for improving efficiency, reliability, and user experience in home maintenance services.

2. Problem Statement

Traditional home maintenance service systems face persistent issues such as lack of transparency, delayed or unclear communication, inefficient request handling, biased rating systems, and difficulty in dispute management. These challenges result in customer dissatisfaction, mistrust, and poor service experiences.

Furthermore, existing platforms rely heavily on manual descriptions from customers, which are often vague and incomplete, leading to mismatches between requests and the services offered. Providers also struggle to build trust quickly, as rating systems are typically simplistic and easily biased. In addition, limited communication channels between customers and providers often force them to rely on external apps, reducing accountability. Finally, admins spend significant time resolving repetitive questions and disputes manually.

There is, therefore, a growing need for a modern, AI-enhanced digital platform that streamlines service requests, automates descriptions from images, improves trust through fair scoring, integrates direct communication, and provides instant support—ultimately boosting transparency, efficiency, and user satisfaction. Moreover, the manual assessment of home damages is inherently subjective and time-consuming. Traditional platforms rely on users to provide textual descriptions, which are often vague and prone to inconsistency. This leads to significant bottlenecks in providing fair cost estimates and deploying the correct trade professionals. There is a critical need for an automated system that can objectively evaluate damage and predict costs directly from visual evidence.

3. Project Objective

The main objective of this project is to develop a digital Home Care System that simplifies and optimizes the process of requesting, managing, and monitoring home maintenance services. To achieve this, the system sets out the following objectives:

- **Simplify Service Requests:** Allow customers to submit maintenance requests with images, and automatically generate structured descriptions using Vision-Language Models (VLMs).
- **Build Trust and Transparency:** Replace biased rating-only evaluations with an Trust Score Engine that considers multiple signals (response speed, fairness of offers, completion rates, sentiment from reviews, and visual evidence).
- **Automate Service Assessment and Costing:** Develop and integrate a multi-task deep learning framework capable of simultaneously identifying the required service category (e.g., Plumbing, Carpentry) and predicting a preliminary repair cost directly from user-uploaded images.
- **Introduce AI-Driven Service Request Automation:**
  Design and implement an automated workflow that supports a two-phase request lifecycle (Draft → Accept), allowing AI to identify service category, estimate cost, suggest service timing, and generate complete service requests while maintaining full user control.
- **Automate Service Requests via Chatbot:** Introduce a chatbot that replaces traditional service request forms by enabling customers to create complete home maintenance requests through natural language interaction, streamlining the request process and enhancing user experience.
- **Ensure Efficient Oversight:** Support admins with analytics, reporting, and dispute resolution tools to maintain accountability and platform reliability.

4. Proposed System

The proposed Home Care System is a smart, role-based platform that digitizes the entire workflow of home maintenance services. Customers can submit requests with images, and the system can automatically generate clear descriptions using AI. Providers can respond with offers, and customers can compare these offers transparently.

The platform goes beyond traditional systems by introducing several innovative features:

- **AI-Driven Service Request Automation:** Automates the creation of service requests by analyzing uploaded images and contextual information. The system generates draft requests, estimates budgets, suggests suitable service times, and supports a two-phase approval process (Draft → Accept), ensuring both efficiency and user control.
- **Trust Score Engine:** for more holistic and fair provider evaluation.
- **Image-to-Text Request Generator:** that leverages VLMs to reduce vague descriptions.
- **Support Chatbot:** that provides instant answers to common questions.
- **AI-Powered Cost Estimator:** Leverages a multi-task vision model to provide instant, objective cost estimations, reducing the dependency on manual provider assessments.

  In addition, the system includes real-time notifications, analytics dashboards, rating and review modules, and dispute management services, ensuring a seamless, reliable, and transparent user experience. By integrating these advanced features, the proposed system not only improves operational efficiency but also addresses long-standing problems of trust, miscommunication, and inefficiency in home maintenance platforms.

5. Report Organization

The report is structured as follows:

Chapter 1: Introduction

Chapter 2: Basic Concepts and Reference Study

Chapter 3: Project Management

Chapter 4: System Analysis

Chapter 5: System Design

Chapter 6: Practical Implementation

Chapter 7: Report Overview

6. Report structure and purposes

- **Chapter 1: Introduction**

  This chapter lays the foundation of the report by introducing the project, identifying the problem statement, and defining the objectives. It also

provides an overview of the proposed system and explains how the report is organized.

- **Chapter 2: Fundamental Concepts and Literature Review**

  This chapter presents the theoretical background and existing research relevant to the project. It covers key fundamental concepts related to the system and a literature review analyzing similar systems, technologies, and methodologies that influenced the project.

- **Chapter 3: Project Management**

  This section outlines the project management aspects, including the project charter, scope of work (SOW) document, Gantt chart for project planning, and risk management strategies. It ensures that the project is well-structured and managed efficiently.

- **Chapter 4: System Analysis**

  This chapter focuses on the analysis phase of the system, starting with the project timeline, followed by the Software Requirements Specification (SRS) document. It also includes requirements modeling, initial test cases, and the initial Requirement Traceability Matrix (RTM), ensuring comprehensive requirement analysis.

- **Chapter 5: System Design**

  This section provides a detailed explanation of the system design, including system architecture and the detailed design of system components. It serves as a blueprint for system implementation, ensuring a structured approach to development.

- **Chapter 6: Practical Implementation**

  This chapter details the practical aspects of the project, including the tools and technologies used, system interfaces, AI-based components, and test case execution. It also includes the final version of the Requirement Traceability Matrix (RTM), demonstrating the alignment between requirements and implementation.

7. Summary

This chapter has outlined the need for an Home Care System, explained the objectives of the project, introduced the proposed solution at a high level, and provided an overview of the report's structure. This introduction sets the stage for a deeper exploration of each aspect of the project in the following chapters.

Chapter 2 Fundamental Concepts and Literature Review

1. Introduction

This chapter provides a foundation for understanding the Home Care System by exploring fundamental concepts and reviewing related studies. It examines essential terminology, key theories, and the principles behind the technologies used in the system. along with insights from existing solutions to guide and support the development of the proposed system.

2. Literature Review

This section reviews existing digital platforms that manage home maintenance services, comparing their **features, approaches, strengths, and limitations** to highlight the **research gap our system fills**.

2.1 Maintenance / Home Care Platforms

In recent years, several mobile applications and digital platforms have been developed to facilitate **home maintenance and repair services**, aiming to connect homeowners or tenants with skilled service providers. One notable example is the **HOMERESC mobile application**, developed in Malaysia, which serves as a structured platform for booking home repair services and improving access to trusted professionals.

HOMERESC was created in response to challenges faced during the **COVID-19 Movement Control Order (MCO)** in Malaysia, which restricted mobility and made it difficult for residents to find reputable service providers. The platform enables users—particularly newcomers in residential areas—to discover and book services such as plumbing, cleaning, furniture assembly, and electrical work through a mobile app interface. It uses **an agile development methodology** and includes features like service search, viewing provider catalogs, in-app booking, rating/review after service completion, and profile management for both users and providers.

Additionally, HOMERESC integrates **basic communication and security features**, including chat activity, user registration/login, password protection, file restrictions, and email verification. These mechanisms aim to increase user trust and data protection. However, the study identifies several **limitations in current home service apps**:

- Lack of **detailed provider information** (e.g., qualifications, communication preferences), which affects user trust.
- Inconsistent or mixed ratings and reviews, making it difficult for users to judge provider reliability.
- Limited payment options, which may exclude certain user groups.
- Some systems allow browsing without secure login, raising privacy concerns.
- Absence of advanced trust mechanisms beyond star ratings, leaving space for biased or fraudulent reviews.

Beyond HOMERESC, other platforms such as **ServisHero** and **Zonar** have also been developed to address daily service needs. ServisHero is a Malaysian app that allows users to select service issues and book skilled professionals, focusing on ease of identification and

booking. Zonar emphasizes **price comparison and freelancer visibility**, while Carousell provides a broader consumer-to-consumer marketplace, including home services.

While these systems **simplify service discovery and booking**, they mostly rely on **manual textual descriptions** provided by customers and **basic rating systems** for trust assessment. This leaves gaps in automation, transparency, and intelligent matching between requests and service providers.

Your **Home Care System** aims to address these gaps through several innovations:

- **AI-powered Image-to-Text Request Generator** using VLMs to reduce reliance on vague textual inputs.
- **Trust Score Engine** to complement and improve upon traditional rating systems.
- **Context-aware communication (chat service)** and **FAQ chatbot** for improved interaction and support.

By integrating these technologies, our system moves beyond the static service marketplace model toward a **smarter, more transparent, and more user-centric maintenance platform**.

2.2 Trust and Reputation Systems

Trust and reputation are central to the success of **digital service platforms**, since users depend on signals such as ratings and reviews (R&R) to evaluate service providers. In **multi-sided platforms (MSPs)** like Amazon, Alibaba, or app marketplaces, R&R mechanisms are designed to reduce information asymmetry between providers and customers. However, literature highlights three persistent problems: **review shortage**, where insufficient feedback limits reliability; **review bias**, where extreme or selective feedback skews perceptions; and **review fraudulence**, where fake or manipulated reviews undermine credibility. These challenges show that R&R systems often fall short of their intended role as governance mechanisms for building trust. Researchers have proposed enhancements to R&R by viewing platforms not merely as **repositories of opinions**, but as **knowledge co-creation hubs**. This perspective emphasizes mechanisms that generate high-quality, reliable product or service knowledge rather than simplistic star ratings. Still, most current platforms rely heavily on user-generated reviews that are often incomplete, misleading, or inconsistent, leaving users uncertain about provider reliability.

Beyond e-commerce, **trust management research** in IoT and multi-agent systems provides advanced models that could inspire new solutions. For example, **COBRA (Context-aware Bernoulli Neural Network)** introduces a framework that aggregates both **first-hand and second-hand evidence** while preserving privacy and contextual information, achieving more accurate trust assessments than traditional binary or star-based systems. Similarly, **Trust2Vec** leverages **network embeddings** to analyze trust relationships at scale, effectively detecting malicious behaviors such as **self-promotion and bad-mouthing attacks** that can distort trust scores. Although these frameworks were designed for IoT and networked environments, their principles—multi-factor evaluation, robustness against manipulation, and context-awareness—are relevant to any platform requiring reliable trust signals.

In contrast, most existing **home service and maintenance apps** (e.g., HOMERESC, ServisHero) still use **basic star ratings and textual reviews**. These methods are vulnerable to bias, fraud, and lack of detail, providing little more than a superficial measure of trust. Customers struggle to distinguish truly reliable providers, while providers may be unfairly judged by a small set of biased reviews.

Research Gap: Despite advances in trust management in other domains, **home maintenance platforms have not integrated AI-driven trust models**. This gap motivates the introduction of a **Trust Score Engine** in the Home Care System, which aggregates multiple factors such as response speed, offer fairness, completion rates, customer sentiment, and even image-based evidence. By doing so, the system addresses the limitations of conventional rating methods and provides customers with a **holistic and tamper-resistant indicator of provider quality**.

2.3 Intelligent Request Handling

An essential component of digital home service platforms is the mechanism by which customers **submit service requests** and providers **interpret and respond** to them. In most existing applications, requests are created by filling out **manual text descriptions** accompanied by optional images. While this approach is simple to implement, it suffers from several limitations: customers may provide **vague, incomplete, or inconsistent descriptions**, making it difficult for providers to understand the scope of work. This miscommunication often results in **incorrect offers, delayed responses, or service dissatisfaction**.

Several studies in related domains have attempted to improve request handling through **natural language processing (NLP)**. For example, building maintenance systems have explored using NLP models to **classify free-text descriptions of issues** into predefined categories, thereby reducing ambiguity and enabling automated routing of requests. While effective in structured environments, such approaches remain dependent on the **quality of the textual input** provided by the user, and they are limited when users submit only images without clear descriptions.

Recent advances in **Vision-Language Models (VLMs)** open up new possibilities for handling unstructured customer input. Unlike traditional NLP or image recognition alone, VLMs can process an image and generate a **coherent textual description**, bridging the gap between visual evidence and structured request information. For example, a user uploading a photo of a leaking pipe would automatically receive a generated description such as *"Leaking water pipe under the kitchen sink"*. This reduces the burden on the customer to articulate the problem and minimizes errors due to vague input.

Several modern applications already leverage image recognition or vision-assisted features to enhance user support. For instance, **AI Repair: DIY Home Fix Guide** allows users to upload photos of damaged household items (such as plumbing, furniture, or appliances) and returns a tailored repair guide, including possible causes, tools required, and step-by-step instructions. Similarly, **Proto's AI assistant** in the EV charging network **chargEV** enables customers to upload photos of error codes or screenshots during support chats, with the AI interpreting these images to provide solutions, thus reducing reliance on human agents. Another example, **Homescope.ai**, builds maintenance profiles of houses by analyzing images of home systems (e.g., HVAC, roof, water heaters) and generates personalized maintenance plans. These systems demonstrate the growing role of **vision-based AI in service contexts**, but they

primarily focus on **diagnosis, planning, or visualization** rather than converting images into **structured service request descriptions** within a home maintenance marketplace.

Research Gap: While NLP-driven classification has been explored in building maintenance, and vision-based AI has been applied in adjacent service domains, **no home service platform has yet integrated VLM-based request generation**. This represents a novel contribution of the Home Care System, which will utilize a **VLM-powered Image-to-Text Service Request Generator** to standardize input quality, reduce miscommunication, and improve the accuracy of provider matching.

2.4 Multi-Task Learning for Automated Assessment

While existing platforms like *HOMERESC* and *ServisHero* simplify service discovery, they remain limited by manual request entry and basic rating systems. Our system bridges this gap by utilizing **Multi-Task Learning (MTL)**. Unlike single-task models, MTL allows the system to share representations between related tasks—classification of service type and regression for cost estimation—improving the efficiency and accuracy of both. We employ an **EfficientNet-B4** backbone, known for its optimal balance between computational efficiency and accuracy, to extract high-level visual features from damage images (more details in Chapter 6).

2.5 Support and Automation

Customer support is a critical aspect of digital service platforms, yet many home maintenance applications rely on manual admin responses or external FAQs, which can be time-consuming and inconsistent. To improve efficiency, platforms in e-commerce and travel industries (e.g., **Amazon, Booking.com**) have adopted **chatbots and automated FAQs** that handle routine queries such as account setup, payment methods, or service policies. These lightweight systems reduce the burden on human staff while providing users with immediate guidance.

In the home service domain, however, support automation remains underdeveloped. Most applications either lack integrated help tools or redirect users to static FAQ pages, which limits usability and increases reliance on admin intervention.

Research Gap: Despite the proven benefits of FAQ chatbots in other industries, **home maintenance platforms rarely integrate automated support mechanisms**. The proposed Home Care System addresses this by incorporating a **simple FAQ chatbot** that answers common customer and provider questions directly within the application. This ensures **faster problem resolution, reduced admin workload, and smoother onboarding** for new users.

2.6 Proposed System

The review of existing literature highlights that while current home service platforms such as HOMERESC and ServisHero simplify service discovery and booking, they remain limited by **basic rating systems, manual request entry, fragmented communication, and weak support tools**. Research in trust management (e.g., COBRA, Trust2Vec) and advances in AI-driven request handling demonstrate promising directions, yet these innovations have not been fully applied to the home maintenance domain. Similarly, while e-commerce and freelance platforms integrate in-app communication and structured dispute processes, most home care

systems continue to rely on informal channels. Finally, automated support, common in industries like retail and travel, is rarely present in home service apps.

These gaps collectively justify the need for a **smarter, AI-enhanced Home Care System** that integrates advanced trust scoring, image-to-text request generation, and. Building on the limitations identified in prior studies, the proposed system aims to deliver a more transparent, efficient, and user-centric approach to home maintenance services.

- ## Axxess HomeCare

  Axxess HomeCare is a cloud-based solution designed to manage various aspects of home care services, including intake, scheduling, and billing.

  ### Advantages:

  1. **Comprehensive Feature Set:** Offers a wide array of tools covering client intake, scheduling, human resources, and billing.
  2. **User-Friendly Design:** Emphasizes ease of use with an intuitive interface, reducing training time.
  3. **Family Portal:** Includes a portal for family members to stay informed about their loved one's care.

  ### Disadvantages:

  1. **Integration Challenges:** May face limitations when integrating with certain third-party applications.
  2. **Feature Overlap:** Some features may overlap, leading to potential redundancy and confusion.

  ### Main Features:

  - **Intake Management:** Streamlines the process of onboarding new clients.
  - **Human Resources Management:** Manages caregiver credentials, training, and performance evaluations.
  - **Electronic Visit Verification (EVV):** Ensures compliance with visit verification requirements through electronic tracking.

- ## ShiftCare

  ShiftCare is a home care agency software focusing on scheduling, client management, and compliance tracking.

  ### Advantages:

1. **Intuitive Scheduling:** Provides drag-and-drop scheduling with real-time updates, enhancing operational efficiency.
2. **Compliance Management:** Assists in maintaining compliance with industry standards and regulations.
3. **Mobile App:** Offers a mobile application for caregivers to access schedules and client information on the go.

Disadvantages:

1. **Limited Advanced Features:** May lack some advanced functionalities found in more comprehensive systems.
2. **Scalability Concerns:** May be better suited for small to mid-sized agencies, with potential limitations for larger organizations.

Main Features:

- **Client Management:** Maintains detailed client profiles, including care plans and preferences.
- **Document Management:** Stores and manages important documents securely.
- **Progress Notes:** Allows caregivers to record and share progress notes, facilitating continuity of care.

In conclusion, the analysis of these home care management systems reveals a range of features and functionalities aimed at enhancing the efficiency and quality of home care services. While each system offers unique advantages, common elements such as scheduling, compliance management, and mobile accessibility are prevalent. Identified disadvantages, including customization limitations and integration challenges, highlight areas for improvement. In developing a new home care system, incorporating robust user authentication, real-time tracking, comprehensive service management, and user-friendly interfaces will be crucial to address existing gaps and meet the evolving needs of home care agencies.

Here's a summary of the standout service for each of the 4 Home Care System mentioned

Table 1 Comparative Analysis of Similar Systems and the Target System

System Feature	Carecenta	WellSky Personal Care	Axxess HomeCare	ShiftCare	Our System
User Authentication & Access Control	✓	✓	✓	✓	✓

Registration & Account Management	✓	✓	✓	✓	✓
Worker Management	✓	✓	✓	✓	✓
Service Management	✓	✓	✓	✓	✓
AI-Driven Service Request Automation	✗	✗	✗	✗	✓
Image-to-Text Description Generation (VLM)	✗	✗	✗	✗	✓
Draft & User Approval Workflow	✗	✗	✗	✗	✓
AI-Based Cost Estimation	✗	✗	✗	✗	✓
Real-Time Tracking	✓	✓	✓	✓	✗
Notifications	✓	✓	✓	✓	✓
Dashboard & Analytics	✗	✓	✓	✓	✓
Ratings & Reviews	✗	✗	✗	✗	✓
Trust Scoring	✗	✗	✗	✗	✓
Payment System	✓	✗	✓	✗	✓ *
Multi-Language Support	✗	✗	✗	✗	✓ *
(*) means features that will be achieved in the future, outside the scope.					

Chapter 3 Project Management

1. Introduction:

In this chapter, we will dive into the management phase of the project, which is a critical aspect of ensuring the project's success. We will examine the project charter, project plan, Statement of Work (SOW) document, stakeholder analysis, and risk management strategies to effectively guide and control the project from initiation to completion.

2. Project charter

A project charter is a formal document that serves as an official authorization for the start of a project. It acts as a reference point throughout the project, providing a clear understanding of the project's purpose and establishing a foundation for decision-making and project governance.

❖ **Total Project Duration:** April 06, 2025 – January 12, 2026

❖ **Project Schedule & Key Milestones:**

Phase	Milestone	Start Date	Finish Date	Status
Phase I	Project Kickoff, User Mgmt. Service Ready, Service Request & Mgmt. Ready, Notification Service Ready, Project Delivery	April 06, 2025	July 13, 2025	**Completed**
Phase II	Project Kickoff, Integrating VLM, Core Rating & Review Service Complete, Trust Score Engine Ready, Automation Sprint Complete, Project Delivery	October 18, 2025	January 12, 2026	**Completed**

❖ **Project Objectives:**

- ❖ Develop a flexible and extensible digital platform for managing home maintenance service requests, designed to support both core service management functionalities and advanced AI-driven automation.
- ❖ Enable customizable request submission and provider matching, allowing users to upload images and specify issues while allowing service providers to offer competitive solutions.
- ❖ Enhance communication and workflow tracking through role-based access, real-time notifications, and status updates throughout the request lifecycle.
- ❖ Improve service transparency and quality control by supporting user ratings and feedback for each completed service, through structured ratings, user feedback, and intelligent trust-related indicators that contribute to more informed decision-making.
- ❖ Ensure scalability and data security by implementing a modular architecture with access control based on user roles (e.g., admin, customer, service provider).

- **Approach:**
  - ❖ Conduct a needs assessment to understand user expectations for home maintenance services.
  - ❖ Design a modular system architecture that supports request management, image uploads, role-based access, and service provider-customer communication.
  - ❖ Use internal development teams to handle system design, implementation, and testing.
  - ❖ Implement and test notification flows, request tracking, and service rating features to ensure smooth and transparent operations.

- **Roles and Responsibilities:**

Name	Role	Responsibility
Dr.Riad Sonbol	Project Manager &Sponsor	Provide strategic direction, oversee project funding and resources, resolve high-level issues, and ensure the project meets organizational objectives.
Eng.Anas Abdulaziz	Project Supervisor	Manage day-to-day project execution, coordinate with team members, track progress, manage risks, and report on project status to Dr. Riad Sonbol.
Mohammed Noor	SE & Mobile App Developer [Flutter]	Design and implement the cross-platform mobile interface, integrate backend APIs for real-time service tracking, and ensure a seamless mobile user experience.
Kasem Almaleh	SE & Backend [Django]	Architect and implement the core backend infrastructure using Django, focusing on the **AI Inference Service** layer. This includes managing model concurrency, optimizing system performance, and developing robust RESTful APIs for mobile and external service integration.
Mohammad Alhendi	SE & Frontend [React]	Design and implement responsive user interfaces, integrate frontend applications with backend APIs, and contribute to deep learning model development through data preparation interfaces, visualization of training results, and model evaluation support.

Table 2 Roles and Responsibilities

Membership Transition Note:

- **Mohammed Noor:** joined the team at the start of Phase II (Senior Project II).
- **Mohammad Bazzallah:** departed the team upon the successful completion of Phase I (Senior Project I).

3. The SOW document:

Statement of Work is a comprehensive document that defines the scope of work for a project. It outlines the specific tasks, deliverables, timeline, and responsibilities. The SOW document provides a clear understanding of what needs to be accomplished, the project's objectives, and the criteria for success.

1. Project Description and Objectives:

The project aims to develop a Home Care system that facilitates the management of home care services. The system will assist care teams in organizing, scheduling, and delivering services effectively while improving client satisfaction.

2. Project Scope:

The Home Care System is designed to digitize and enhance the complete lifecycle of home maintenance service management. The system enables customers to submit service requests using images and minimal input, while leveraging AI-driven automation to generate structured service descriptions, estimate costs, and suggest suitable service times. Customers can review, edit, and approve AI-generated drafts before finalizing requests, ensuring both accuracy and user control. The platform facilitates transparent interaction between customers and service providers by allowing providers to submit offers and enabling customers to compare these offers based on price, trust indicators, and service suitability. The system also supports real-time notifications, and comprehensive request status tracking. Additionally, trust and service quality are enhanced through ratings, reviews, and trust evaluation mechanisms. Service providers benefit from increased visibility and fair opportunity based on performance and reliability rather than manual or biased selection.

3. Project Goals:

- Develop a smart, role-based digital platform that simplifies and automates home maintenance service management for customers, service providers, and administrators.
- Integrate AI-driven technologies to assist in service request creation, including image-to-text analysis, automated cost estimation, identifying category of issue need to fix, intelligent scheduling, and draft-based user approval workflows.
- Enable intuitive and user-friendly interfaces that allow customers to request services easily and allow service providers to manage offers, availability, and appointments efficiently.
- Improve transparency, trust, and service quality through trust scoring mechanism, structured ratings, and user feedback mechanisms.
- Provide real-time notifications, analytics dashboards, and detailed reports to support operational monitoring, service performance evaluation, and informed decision-making.

Deliverables:

- Project plan.
- SRS documentation.
- Final project report.
- Backend and frontend components – the Home Care system itself.

4. Project Requirements:

Technology and Tools:

- Programming Languages: Python, JavaScript, HTML, CSS, Dart.
- Frameworks: Django (backend), React (frontend), Flutter (mobile app).
- Database: MySQL.

5. Assumptions:

- Regular availability of project team members and stakeholders for feedback.
- Continuous feedback and supervision from the project manager.
- Hardware and software resources required for development and deployment.

6. Project Resources:

Human Resources:

- Dr. Riad Sonbol: Project Manager
- Eng.Anas-Abdulaziz: Project Supervisor
- Kasem Almaleh: SE-Developer (Backend – Django)
- Mohammed Noor: SE-Developer (Mobile App – Flutter)
- Mohammad Alhendi: SE-Developer (Frontend – React)

7. Schedule:

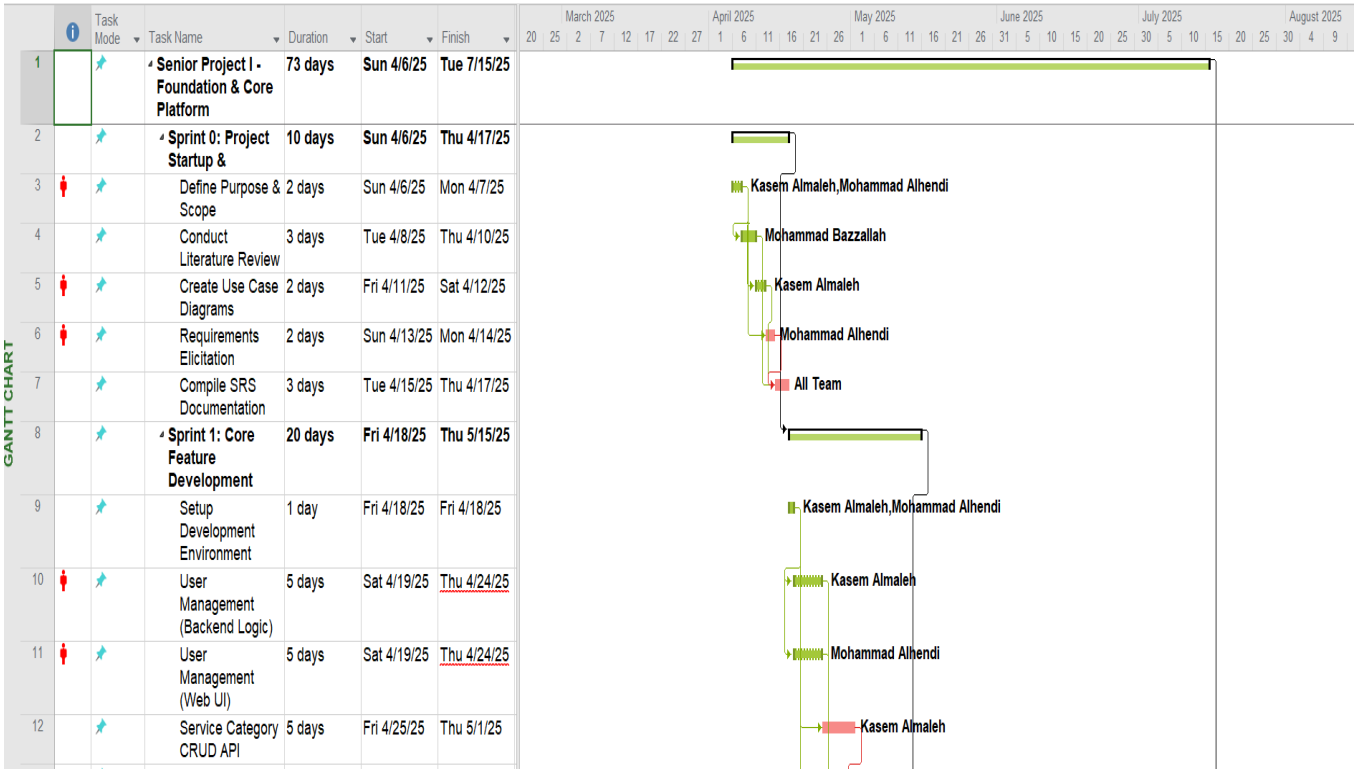
❖ Phase I (First Senior):

- Project Start Date: April 06, 2025
- First Seminar: April 16, 2025
- Second Seminar: ***
- Final Seminar: July 12, 2025
- Project Finish Date: July 06, 2025

❖ Phase II (Second Senior):

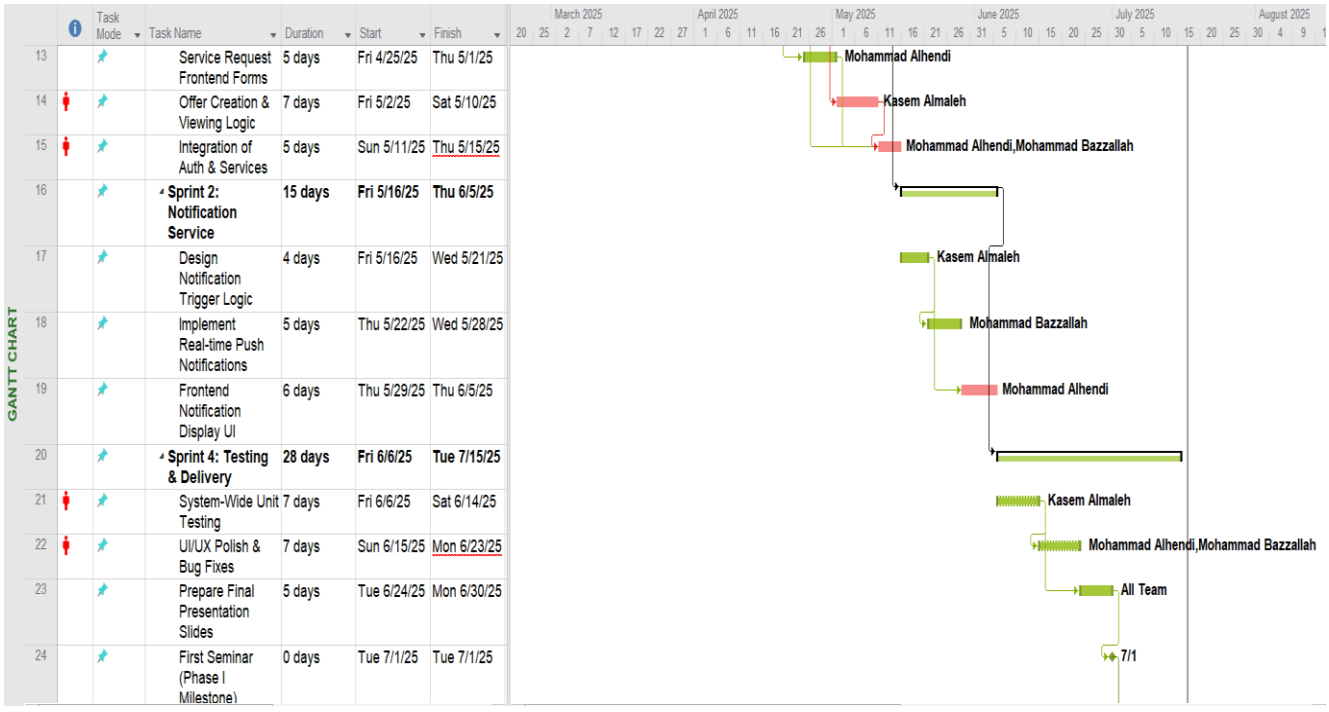
- Project Start Date: October 18, 2025
- First Seminar: November 12, 2025
- Second Seminar: December 20, 2025
- Final Seminar: January 31, 2026
- Project Finish Date: January 12, 2026

4. Gantt Chart:



	Task Mode	Task Name	Duration	Start	Finish
1		**Senior Project I - Foundation & Core Platform**	**73 days**	**Sun 4/6/25**	**Tue 7/15/25**
2		**Sprint 0: Project Startup &**	**10 days**	**Sun 4/6/25**	**Thu 4/17/25**
3		Define Purpose & Scope	2 days	Sun 4/6/25	Mon 4/7/25
4		Conduct Literature Review	3 days	Tue 4/8/25	Thu 4/10/25
5		Create Use Case Diagrams	2 days	Fri 4/11/25	Sat 4/12/25
6		Requirements Elicitation	2 days	Sun 4/13/25	Mon 4/14/25
7		Compile SRS Documentation	3 days	Tue 4/15/25	Thu 4/17/25
8		**Sprint 1: Core Feature Development**	**20 days**	**Fri 4/18/25**	**Thu 5/15/25**
9		Setup Development Environment	1 day	Fri 4/18/25	Fri 4/18/25
10		User Management (Backend Logic)	5 days	Sat 4/19/25	Thu 4/24/25
11		User Management (Web UI)	5 days	Sat 4/19/25	Thu 4/24/25
12		Service Category CRUD API	5 days	Fri 4/25/25	Thu 5/1/25

Figure 1 gantt chart – senior I



	Task Mode	Task Name	Duration	Start	Finish
13		Service Request Frontend Forms	5 days	Fri 4/25/25	Thu 5/1/25
14		Offer Creation & Viewing Logic	7 days	Fri 5/2/25	Sat 5/10/25
15		Integration of Auth & Services	5 days	Sun 5/11/25	Thu 5/15/25
16		**Sprint 2: Notification Service**	**15 days**	**Fri 5/16/25**	**Thu 6/5/25**
17		Design Notification Trigger Logic	4 days	Fri 5/16/25	Wed 5/21/25
18		Implement Real-time Push Notifications	5 days	Thu 5/22/25	Wed 5/28/25
19		Frontend Notification Display UI	6 days	Thu 5/29/25	Thu 6/5/25
20		**Sprint 4: Testing & Delivery**	**28 days**	**Fri 6/6/25**	**Tue 7/15/25**
21		System-Wide Unit Testing	7 days	Fri 6/6/25	Sat 6/14/25
22		UI/UX Polish & Bug Fixes	7 days	Sun 6/15/25	Mon 6/23/25
23		Prepare Final Presentation Slides	5 days	Tue 6/24/25	Mon 6/30/25
24		First Seminar (Phase I Milestone)	0 days	Tue 7/1/25	Tue 7/1/25

Figure 2 gantt chart - senior I

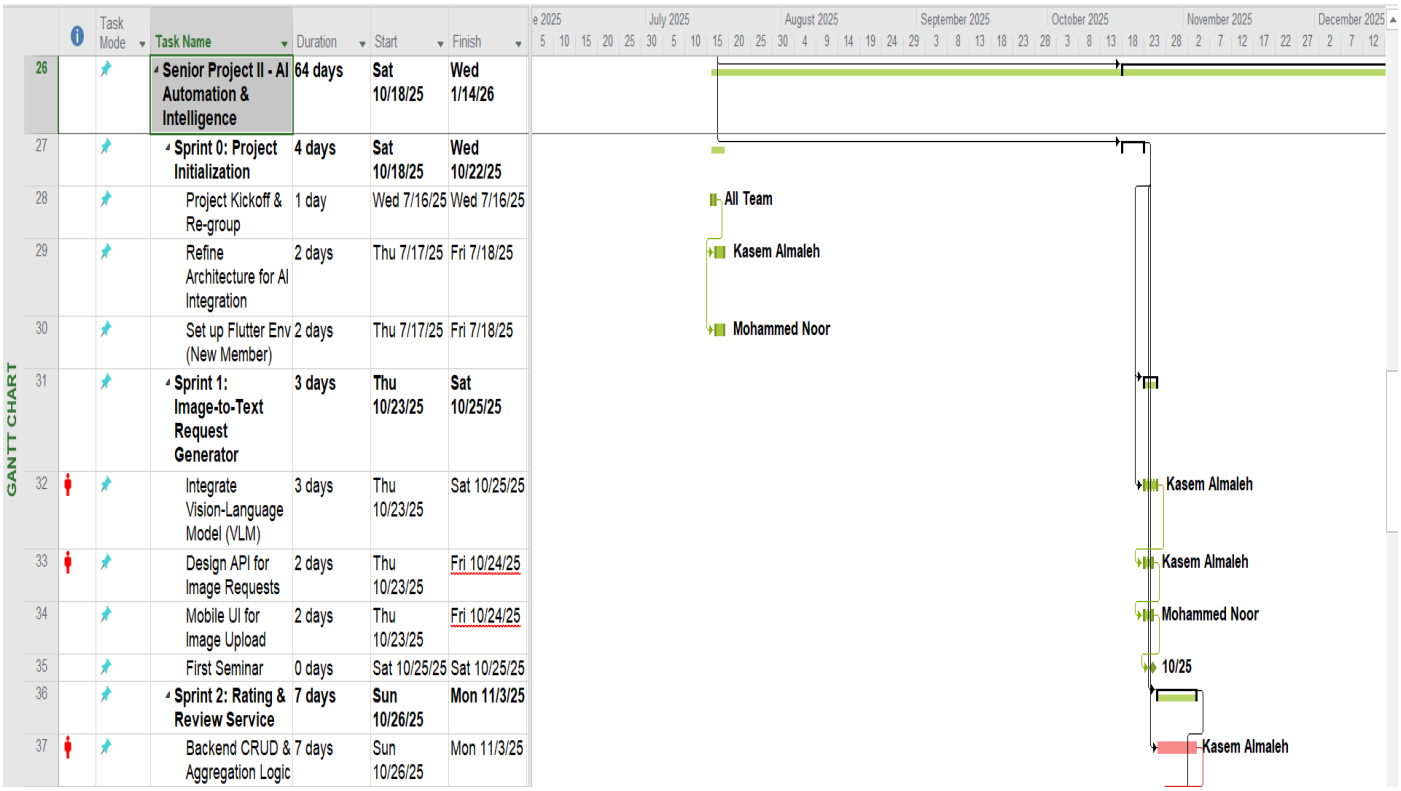


Figure 3 gantt chart - senior II

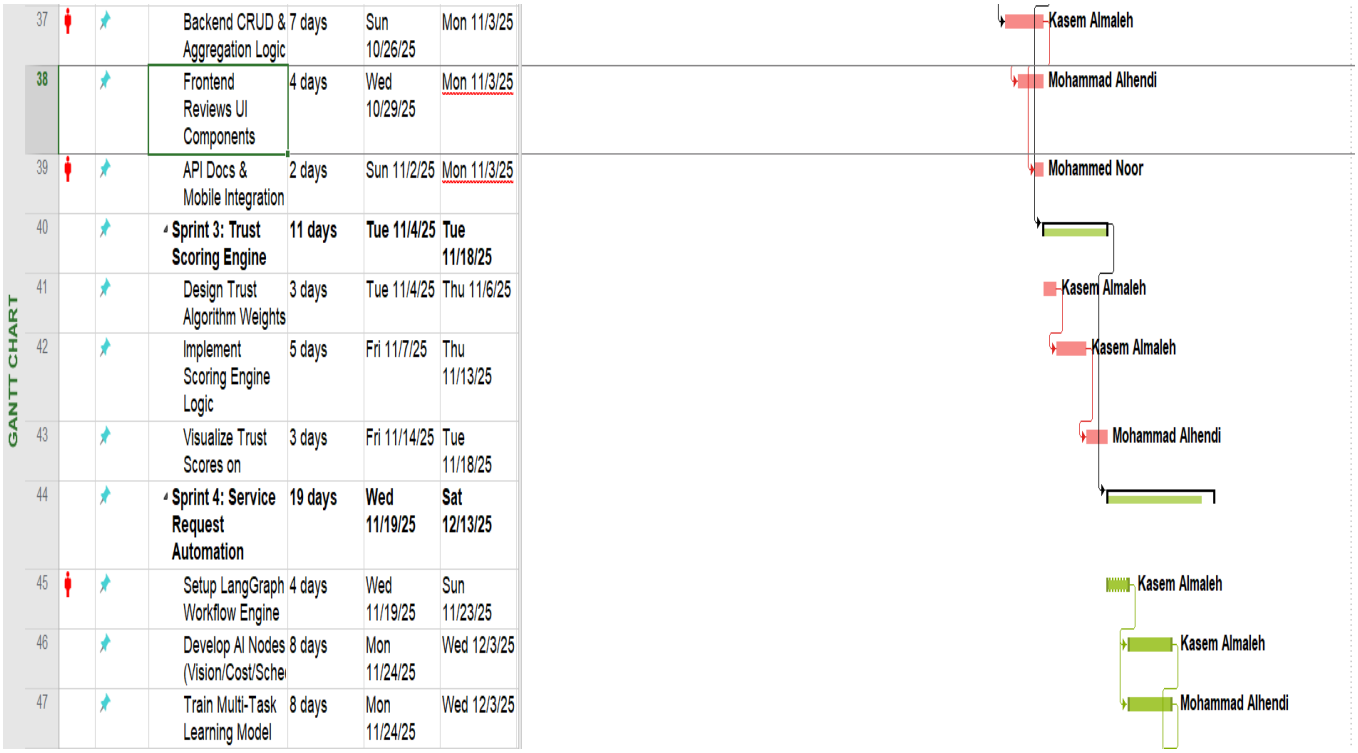


Figure 4 gantt chart - senior II

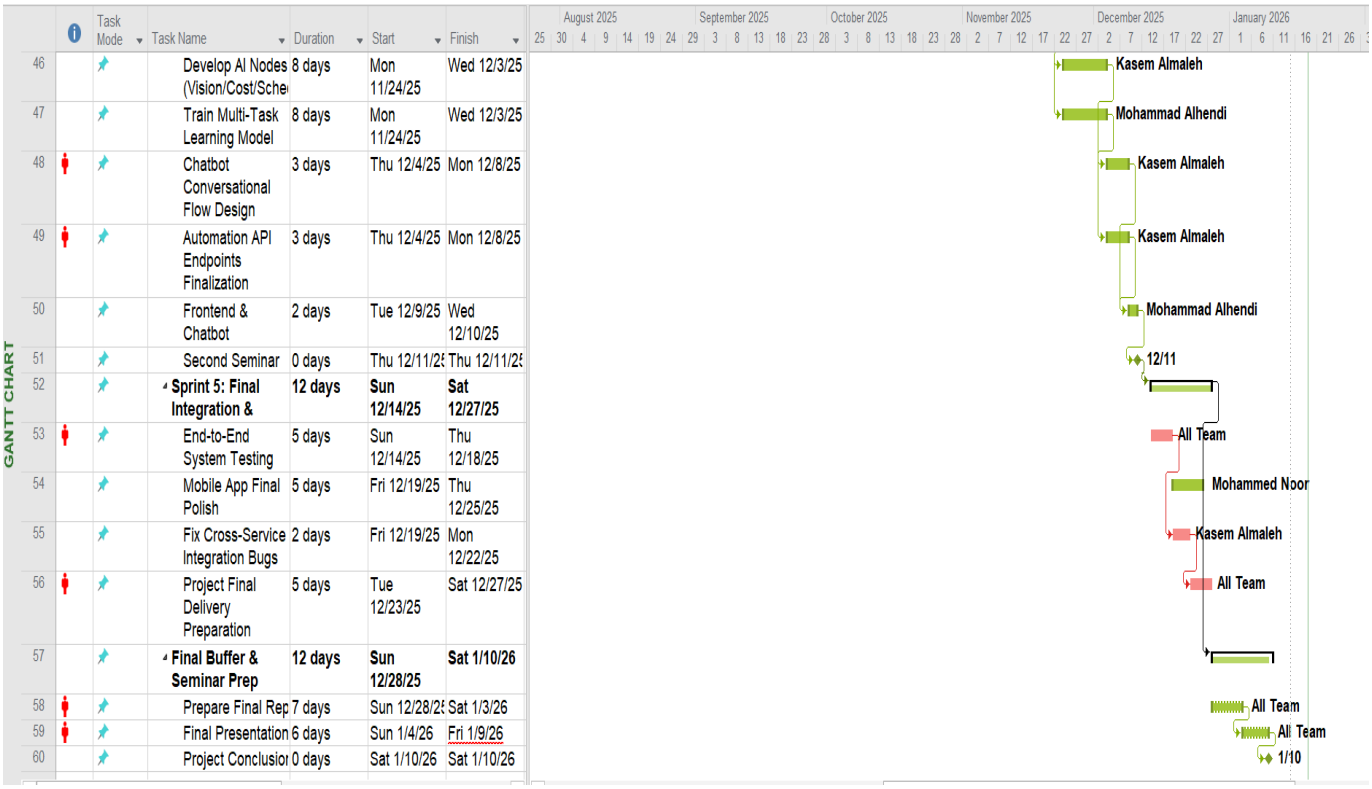


Figure 5 gantt chart - senior II

5. Risk management

Risk Title	Risk Description	Raised Date	Tracking Frequency	State	Impact	Mitigation Plan
Team of only three students	If one of the students stops working for some reason, the whole project progress will be impacted.		Weekly	Active	High	The one who finishes early will assist the other with their tasks.
Fuzzy project scope	Misunderstanding the project scope at the beginning of the work can lead to major mistakes in implementation.		Daily	Closed	Medium	Conduct regular team discussions and meetings at the end of each milestone to clarify objectives.
Learning new technologies	Using AI components with limited knowledge can create challenges in development.		Weekly	Under Mitigation	High	Allocate dedicated learning sessions and use online resources to build understanding.
Separation of work	One member is responsible for the frontend and the other for the backend, which can cause coordination gaps.		Daily	Active	Medium	Regular integration testing and progress sync-ups to ensure compatibility between backend and frontend.
Deadline pressure	Tight deadlines may lead to shortcuts in development, causing potential quality issues.		Weekly	Active	Medium	Break tasks into smaller deliverables and prioritize critical functionality to ensure quality work within deadlines.

Lack of testing expertise	Limited experience with testing AI models could result in poorly validated functionality or errors.		Weekly	Active	High	Assign specific time for testing and involve external peers to review testing plans.
Requirement changes	Late changes in requirements from instructors or stakeholders can disrupt the project timeline and scope.		Weekly	Active	High	Allocate buffer time in the schedule and ensure effective communication with stakeholders to manage changes.

Table 3 Risk management

6. Summary

In conclusion, good project management is vital for successful software system development. It provides structure, ensuring projects are completed within scope and schedule. Overall, strong project management enhances the delivery of high-quality software systems that meet the goals of the project.

Chapter 4 System Analysis

1. Introduction

This chapter focuses on the detailed analysis of the Home Care System. It defines the system's requirements, functionality, and architecture by examining user needs and the operational context. The chapter includes an exploration of functional and non-functional requirements, use cases, and workflows to provide a comprehensive understanding of how the system will address the identified challenges and achieve the project objectives. This analysis forms the foundation for designing and implementing the proposed solution.

2. Project Timeline

The project was executed following a structured timeline, organized into distinct sprints to ensure steady progress and timely completion of key milestones. The schedule was visualized using a Gantt chart, which outlined all major phases from initial planning to final delivery.

➢ Senior Project I – Foundation & Core Platform

Focus: Building a stable, functional Home Care System with essential features and clean architecture.

Included Phases & Sprints:

- **Sprint 0: Project Startup & Analysis**
  - This initial phase focused on defining the project's foundation and gathering requirements.
  - Key activities included defining the Purpose & Scope, conducting a Literature Review, creating Use Case Diagrams, performing Requirements Elicitation, and compiling the SRS Documentation.
- **Sprint 1: Core Feature Development**
  - This sprint was divided into two parallel development tracks: User Management and Service Management.
  - **User Management** tasks included implementing user login/logout, profile management, and registration handling.
  - **Service Management** tasks involved creating functionalities for service categories (CRUD), submitting service requests, and managing offer creation and viewing.
- **Sprint 2: Notification Service**
  - This sprint was dedicated entirely to developing the system's notification capabilities.
  - Work included implementing the logic to trigger notifications for various events and ensuring notifications could be displayed and marked as read by the user.
- **Sprint 4: Testing & Delivery**
  - The final phase concentrated on quality assurance and project wrap-up.

- This involved comprehensive, system-wide testing of all features and the preparation of the final project documentation.

- **Major Milestones**
  - Throughout this timeline, the project was also structured to meet key academic checkpoints as defined in the Statement of Work, including a First Seminar, Second Seminar, and a Final Seminar, all leading up to the final project finish date.

➢ Senior Project II – AI Automation & Intelligence

Focus: Transforming the system from manual → intelligent & automated.

Included Phases & Sprints:

- **Sprint 0: Project Initialization & Architecture Definition**
  - This initial sprint focused on establishing the technical and architectural foundation of the project.
  - Key activities included project initialization, finalizing system requirements, refining the system architecture, and conducting the project kickoff.
  - This phase ensured alignment between system objectives, architectural design, and implementation planning before development began.

- **Sprint 1: Image-to-Text Request Generator**
  - This sprint focused on enabling intelligent service request creation using images.
  - Key tasks included implementing the Image-to-Text Request Generator, integrating a Vision-Language Model (VLM), and designing APIs to support image-based service requests.
  - The AI module was initialized and validated to ensure accurate extraction of structured service descriptions from uploaded images.

- **Sprint 2: Rating & Review Service**
  - This sprint was dedicated to developing the Rating and Review Service.
  - Activities included implementing backend CRUD operations, review aggregation logic, and frontend API integration.
  - The sprint concluded with the completion of the core rating and review functionality, supporting service transparency and quality assessment.

- **Sprint 3: Trust Scoring Engine**
  - This sprint focused on the design and implementation of the Trust Scoring Engine.
  - The engine aggregated multiple evaluation signals, such as user ratings, service history, and behavioral indicators, to provide a holistic and fair trust score for service providers.
  - The sprint concluded with the Trust Scoring Engine being fully implemented and validated.

Sprint 4: Service Request Automation & Conversational Support

- This sprint represented a major milestone by introducing AI-driven automation and conversational interaction.
- Key activities included:
    - Implementing the Service Request Automation module.
    - Developing the automation workflow engine using LangGraph.
    - Creating workflow nodes and integration tools for vision analysis, cost estimation, and scheduling.
    - Integrating a multi-task learning model for automated service assessment and cost estimation.
    - Developing a support chatbot to assist users during the service request process and answer common questions using natural language interaction.
- Automation and chatbot API endpoints were implemented and integrated with the frontend, enabling both form-based and conversational request creation.

- **Sprint 5: Final Integration & Delivery**

    - The final sprint focused on system-wide integration, testing, and stabilization.
    - Tasks included frontend integration with automation and chatbot APIs, end-to-end testing of automated workflows, and resolving cross-service integration issues.
    - The sprint concluded with system finalization and preparation for final project delivery.

- **Major Milestones**

    - Throughout the project timeline, development activities were aligned with key academic milestones defined in the Statement of Work.
    - These milestones included the First Seminar, Second Seminar, and Final Seminar, culminating in the successful delivery of the Home Care System with AI-driven automation and conversational support capabilities.

3. Software Requirement Specification document (SRS)

1. Introduction

1.1 Purpose

This document specifies the software requirements and high-level design for the **Home Care System**, a smart, AI-enhanced platform designed to simplify, automate, and optimize the process of requesting, managing, and monitoring home maintenance services. The system aims to improve user experience, service accuracy, transparency, and operational efficiency by integrating intelligent automation and decision-support capabilities.

1.2 Project Scope

The **Home Care System** is designed to digitize and enhance the full lifecycle of home maintenance services. It enables customers to request services using images and natural language, supports service assessment, and facilitates transparent interaction between customers and service providers. The system leverages **Artificial Intelligence (AI)** to automatically generate structured service requests from images, estimate preliminary repair costs, suggest service categories and suitable times, and support a two-phase approval process (Draft → Accept) that ensures both automation efficiency and user control.

Additionally, the platform:

- Enhances trust and service quality through implementing trust scoring mechanisms and user reviews.
- Supports real-time notifications, analytics.
- Benefits service providers by increasing visibility and request opportunities based on performance and trust metrics.
- Supports create service request using natural language.

1.2.1 High-Level Requirements:

- **User Authentication and Access Control**: The system must provide secure authentication for administrators, workers, and customers.
  Role-based access control must be implemented to restrict functionalities based on user Role
- **Service Management**: The system should allow administrators to define and categorize maintenance services.
- **Profile management:** Users can manage their account information, including editing and deleting their profiles.
- **Registration:** The system allows new users to register by providing necessary information during the sign-up process.
- **Notifications System:** Customers and workers should receive notifications about service status, offers.

- **Service Request and Management:**
    - Customers should be able to request services by providing details and uploading images of the issue.
    - The system should allow customers to obtain a description through AI if they wish.
    - The system must be able to analyze uploaded images using Vision-Language Models (VLMs) to generate structured service descriptions.
    - The system must support a **two-phase service request workflow (Draft → Accept)**, allowing customers to review, edit, and confirm AI-generated requests.
    - The system must enable customers to create service requests through two methods: manual entry or AI assistance.

- **Offer Management:**
    - Service providers must be able to create offers, and customers must be able to compare and select suitable offers transparently.
- **Trust and Rating System**
    - The system must support customer ratings and reviews for completed services.
    - A Trust Score Engine must aggregate multiple evaluation signals (ratings, service history, behavior metrics) to compute a holistic trust score for service providers.
    - The system shall maintain aggregated rating statistics for each service provider.
- **Conversational Support (Chatbot)**
    - The system must provide a chatbot interface to assist users in creating service requests using natural language.
    - The chatbot must answer common questions and guide users through the request creation and approval process.

1.2.2 Actors:

- **Administrator**: Responsible for overseeing and managing the entire system. Manages users, services. Has the highest level of access and control. View analytics and reports on statistics.
- **Customer:** Users who request home maintenance services.
  Can browse services, submit requests, Write a review after the completion of the service request, and rate services.
- **Service Providers:** Provides maintenance services requested by customers. Can create offers for service requests, provide price estimations, read reviews, and receive ratings. Benefits from increased service requests based on performance.

2. Overall Description

2.1 product perspective

The Home Care System specified in this document is a new, self-contained product designed to streamline customer request and complaint management.
It is not part of an existing product family, nor is it a replacement for any current system.

2.2 Product Features

- **User Authentication and Access Control:**
  - Create, update, delete user profiles (Customers, Service Providers, Administrators), activate, and deactivate accounts.
  - Role-based access control.
- **User Management:**
  - Administrators can onboard, verify, and manage users.
  - User can manage profile.
- **Service Management:**
  - Customers can browse services by category (e.g., plumbing, electrical).
  - Administrators define service types.

- **Offer Management:**
    - Service Providers create offers for service request.
    - Customers receive personalized offers via notifications from service providers.
- **Trust, Rating, and Review System:**
    - Customers can rate and review service providers after service completion.
    - A **Trust Score Engine** aggregates ratings, service history, and behavioral indicators to generate a holistic trust score for providers.
    - The system shall maintain aggregated rating statistics for each service provider.
- **Notifications System:**
    - Push alerts for service requests, offers.
- **Service Request Handling and Automation:**
    - Customers submit requests with descriptions and images of the issue, system notifies eligible service providers, then service providers will create offers, customer will select best offer and the system will help customer.
    - The system analyzes uploaded images using **Vision-Language Models (VLMs)** to generate structured service descriptions.
    - An **AI-driven automation workflow** generates a draft service request, provide relevant service categories, estimates a preliminary budget, and suggests suitable service times.
    - The system supports a **two-phase approval process (Draft → Accept)**, allowing customers to review, edit, and confirm AI-generated requests before final submission.
    - Eligible service providers are notified once a request is accepted.
- **Conversational Service Request Creation:**
    - Customers can create complete service requests through an AI-powered chatbot using natural language interaction.
    - The chatbot collects service details, problem descriptions, preferred time, location, and optional images, and automatically submits structured requests to the backend system.

2.3 User Classes and Characteristics

2.3.1 Administrator

Responsibilities:

- Manages system-wide settings, service providers, and services.
- Monitors fraud, resolves disputes, and generates financial reports

Privileges:

- Create services.
- View Analytics.

2.3.2 Customer

Responsibilities:

- Submit home care service requests with necessary details such as service type, location, preferred time, and any additional notes, Attach images or documents to clarify the service need.
- Complete payments for the requested services.
- Provide service ratings and feedback after the service is delivered.

Privileges:

- Create new service requests.
- Edit or delete their own requests before they are assigned or processed.
- Submit rating and write review after the completion of service request.
- Manage comments related to their own requests.
- Manage their personal account information (e.g., name, phone number, address, and password).

2.3.3 Service Provider

Responsibilities:

- Responding to requests with offers.

Privileges:

- Access requests sent to them, create service offers, view their job history, view rating statistics.

3. System features

3.1 functional requirements

3.1.1 User Authentication and Access Control

REQ-1.1:

The system shall allow customers, service providers, and administrators to log in through their respective interfaces.

REQ-1.2:

The system shall validate credentials using secure password hashing.

REQ-1.3:

The system shall implement role-based access control to restrict functionalities based on the user's role.

REQ-1.4:

The system shall provide a logout option that securely terminates the user session.

REQ-1.5:

The system shall automatically log out inactive users after a predefined period.

3.1.2 Registration Management

REQ-2.1:

The system shall allow new customers to register.

REQ-2.2:

The system shall allow new service providers to register.

3.1.3 Account Management

REQ-3.1:

The system shall allow users to update their account information.

REQ-3.2:

The system shall allow users to delete their accounts.

REQ-3.3:

The system shall allow users to change password.

REQ-3.4:

The system shall allow users to view their profile.

3.1.4 User Management

REQ-4.1:

The system shall allow administrators to view the list of service providers.

REQ-4.2:

The system shall allow administrators to view the list of customers.

REQ-4.3:

The system shall allow administrators to activate and deactivate users.

3.1.5 Service Management

REQ-5.1:

The system shall allow administrators to create service categories.

REQ-5.2:

The system shall allow administrators to update service categories.

REQ-5.3:

The system shall allow administrators to delete service categories.

REQ-5.4:

The system shall allow customers to browse the list of available services.

3.1.6 Service Request and Management

REQ-6.1:

The system shall allow customers to request services by providing issue details and uploading images.

REQ-6.2:

The system shall support flexible and specific scheduling logic.

REQ-6.3:

The system shall allow service providers to view requests by service type.

REQ-6.4:

The system shall support filtering to show new requests that have not received any offer.

REQ-6.5:

The system shall allow service providers to mark service request is completed.

3.1.7 AI-Driven Service Request Automation

REQ-7.1:

The system shall analyze uploaded images using Vision-Language Models (VLMs) to generate structured service descriptions.

REQ-7.2:

The system shall generate a draft service request based on AI analysis before final submission.

REQ-7.3:

The system shall allow customers to review and edit AI-generated service request drafts.

REQ-7.4:

The system shall require customer confirmation (Accept) before converting a draft into a final service request.

REQ-7.5:

The system shall automatically notify eligible service providers only after a service request is accepted.

REQ-7.6:

The system shall maintain a stateful workflow for service request creation, tracking request status through defined states (e.g., Draft, Accepted).

REQ-7.7:

The system shall suggest suitable service time windows based on request context and system-defined scheduling rules.

REQ-7.8:

The system shall allow customers to modify AI-generated budget and scheduling suggestions before accepting the service request.

REQ-7.10:

The system shall orchestrate service request automation using a workflow engine that coordinates multiple tasks.

REQ-7.11:

The system shall log automation workflow steps and outcomes for monitoring, debugging, and future analytics.

3.1.8 Offer Management

REQ-8.1:

The system shall allow service providers to create offers on specific requests.

REQ-8.2:

The system shall allow service providers to view offers they created.

REQ-8.3:

The system shall allow customers to view offers they received.

REQ-8.4:

The system shall support automatic offer expiration after 7 days.

REQ-8.5:

The system shall allow customers to accept an offer.

3.1.9 Notifications System

REQ-9.1:

The system shall support the creation of notifications to service providers when a new service request is created.

REQ-9.2:

The system shall support the creation of notifications to customers when they receive an offer on a request.

REQ-9.3:

The system shall support the creation of notifications to service providers when a customer accepts an offer on a specific request.

REQ-9.4:

The system shall support the creation of notifications to service providers when their offer is rejected on a specific request.

REQ-9.5:

The system shall mark the notification as read when the user views it.

REQ-9.6:

The system shall support the creation of notifications to service providers when they received rating and review from customers.

3.1.10 AI-Powered Cost Estimation

REQ-10.1:

The system shall generate a preliminary cost estimation for each service request using AI-based analysis.

REQ-10.2:

The system shall present cost estimates as decision-support information rather than final pricing.

3.1.11 Trust, Rating, and Review System

REQ-11.1:

The system shall allow customers to rate service providers after service completion.

REQ-11.2:

The system shall allow customers to submit textual reviews for completed services.

REQ-11.3:

The system shall calculate a trust score for each service provider based on ratings, service history, and behavioral indicators.

REQ-11.4:

The system shall associate each rating and review with a specific customer, service provider, and service request.

REQ-11.5:

The system shall restrict rating values to a valid range from 1 to 5.

REQ-11.6:

The system shall enforce a maximum length constraint on textual reviews.

REQ-11.7:

The system shall prevent a customer from submitting more than one review for the same service request.

REQ-11.8:

The system shall store the creation timestamp for each submitted review.

REQ-11.9:

The system shall store the last modification timestamp for each review.

REQ-11.10:

The system shall allow administrators to mark reviews as visible or hidden without deleting them.

REQ-11.11:

The system shall preserve hidden reviews for auditing and historical purposes.

REQ-11.12:

The system shall support flagging reviews that violate platform rules or appear suspicious.

REQ-11.13:

The system shall allow flagged reviews to include an optional reason for flagging.

REQ-11.14:

The system shall compute and store a sentiment score derived from the textual content of reviews.

REQ-11.15:

The system shall support the use of sentiment scores for trust evaluation and moderation purposes.

REQ-11.16:

The system shall maintain aggregated rating statistics for each service provider.

REQ-11.17:

The system shall calculate and store the average rating for each service provider.

REQ-11.18:

The system shall store the total number of reviews associated with each service provider.

REQ-11.19:

The system shall update provider rating aggregates automatically when new reviews are submitted or modified.

REQ-11.20:

The system shall store review-level and provider-level data required for trust score computation.

REQ-11.21:

The system shall expose rating, review, and sentiment data for integration with trust-scoring engines.

REQ-11.22:

The system shall support weighted rating and AI-based trust evaluation models.

3.1.12 Conversational Support (Chatbot)

REQ-12.1:

The system shall provide a chatbot interface to assist users in creating service requests using natural language.

REQ-12.2:

The chatbot shall guide users through the service request creation and approval process.

REQ-12.3:

The chatbot shall provide instant responses to common user inquiries related to services and requests.

3.1.13 Additional Requirements

REQ-13.1:

The system shall support pagination.

REQ-13.2 (Image-Based Classification):

The system shall automatically classify service requests into categories with at least 95% accuracy using deep learning analysis of uploaded images.

4 .Non-Functional Requirements

4.1 Scalability

NF-1.1:

The system shall be scalable to support a growing number of users (customers, service providers, and administrators) without performance degradation.

NF-1.2:

The system architecture shall support horizontal scaling of backend services to handle increased workloads during peak service request periods.

NF-1.3:

The system shall support independent scaling of microservices, including Service Request Automation and Rating & Review services.

NF-1.4:

The system shall support high-volume AI workloads by executing computationally intensive tasks asynchronously.

4.2 Availability

NF-2.1:

The system shall maintain at least 99.5% service availability for core functionalities, including service requests, automation workflows, and notifications.

NF-2.2:

The system shall ensure fault isolation between microservices to prevent failures in one service from affecting others.

NF-2.3:

Long-running tasks (e.g., AI analysis and sentiment processing) shall not block user-facing operations.

4.3 Interoperability

NF-3.1:

The system shall support integration with external services using RESTful APIs and JSON data formats.

NF-3.2:

The system shall use standard APIs (e.g., REST, JSON) to ensure compatibility with external platforms or mobile apps.

4.4 Security

NF-4.1:

The system shall enforce secure login using hashed passwords and secure token-based session management**.**

NF-4.2:

The system shall implement role-based access control (RBAC) to prevent unauthorized access to protected resources.

NF-4.3:

The system shall protect AI and automation endpoints from unauthorized access and misuse.

NF-4.4:

The system shall validate all inputs, including uploaded images and textual data, to prevent malicious payloads.

4.5 Extensibility

NF-5.1:

The system shall be designed using a modular architecture to allow easy addition of new features and services.

NF-5.2:

The system shall support future integration of AI components such as dynamic pricing, service prioritization, and advanced conversational agents without major architectural changes.

NF-5.3:

The system shall support extending automation workflows by adding or modifying workflow nodes.

4.6 Performance

NF-6.1:

The system shall provide a response time of less than 2 seconds for 95% of all user requests under normal load.

NF-6.2:

AI-intensive tasks such as image analysis, cost estimation, and sentiment analysis shall be processed asynchronously using background workers.

NF-6.3:

The system shall use caching mechanisms to reduce database load and improve response times for frequently accessed data.

NF-6.4:

The system shall utilize database indexing to optimize query performance for service requests, offers, and reviews.

4.7 Usability

NF-7.1:

The system interfaces (Customer, Service Provider, Admin) interfaces shall follow standard usability principles, with intuitive navigation and responsive design.

NF-7.2:

AI-generated outputs (e.g., service descriptions, cost estimates) shall be presented clearly and allow user review and adjustment.

NF-7.3:

The system shall provide clear feedback to users during long-running operations, such as AI analysis or request automation.

4.8 Maintainability

NF-8.1:

The system codebase shall follow clean coding standards and be organized into modular components.

NF-8.2:

Each microservice shall be independently maintainable and deployable.

NF-8.3:

The system shall support structured logging and monitoring to facilitate debugging and maintenance.

4.9 Reliability and Data Integrity

NF-9.1:

The system shall enforce data integrity through validation rules and database constraints.

NF-9.2:

The system shall ensure consistency of rating, review, and trust-related data across services.

NF-9.3:

The system shall preserve historical data for auditing and analytics purposes.

4.10 AI Readiness and Automation Support

NF-10.1:

The system shall be designed to support AI-driven workflows without requiring schema changes.

NF-10.2:

The system shall store intermediate and final AI outputs for traceability and future model improvements.

NF-10.3:

The system shall allow AI models and automation logic to be updated independently of core business logic.

5. System Requirements

Req-ID	Requirement Title	Category	Priority
REQ-1.1	The system shall allow customers, service providers, and administrators to log in through their respective interfaces.	User Authentication and Access Control	High
REQ-1.2	The system shall validate credentials using secure password hashing.	User Authentication and Access Control	High
REQ-1.3	The system shall implement role-based access control to restrict functionalities based on the user's role.	User Authentication and Access Control	High
REQ-1.4	The system shall provide a logout option that securely terminates the user session.	User Authentication and Access Control	High
REQ-1.5	The system shall automatically log out inactive users after a predefined period.	User Authentication and Access Control	Medium
REQ-2.1	The system shall allow new customers to register.	Registration Management	High
REQ-2.2	The system shall allow new service providers to register.	Registration Management	High
REQ-3.1	The system shall allow users to update their account information.	Account Management	High
REQ-3.2	The system shall allow users to delete their accounts.	Account Management	High
REQ-3.3	The system shall allow users to change their password.	Account Management	High
REQ-3.4	The system shall allow users to view their profile.	Account Management	High
REQ-4.1	The system shall allow administrators to view the list of service providers.	User Management	High
REQ-4.2	The system shall allow administrators to view the list of customers.	User Management	High
REQ-4.3	The system shall allow administrators to activate and deactivate users.	User Management	High
REQ-5.1	The system shall allow administrators to create service categories.	Service Management	High
REQ-5.2	The system shall allow administrators to update service categories.	Service Management	High
REQ-5.3	The system shall allow administrators to delete service categories.	Service Management	High
REQ-5.4	The system shall allow customers to browse the list of available services.	Service Management	High
REQ-6.1	The system shall allow customers to request services by providing issue details and uploading images.	Service Request and Management	High
REQ-6.2	The system shall support flexible and specific scheduling logic.	Service Request and Management	High
REQ-6.3	The system shall allow service providers to view requests by service type.	Service Request and Management	High
REQ-6.4	The system shall support filtering to show new requests that have not received any offer.	Service Request and Management	High

REQ-6.5	The system shall allow service providers to mark service requests as completed.	Service Request and Management	High
REQ-7.1	The system shall analyze uploaded images using Vision-Language Models (VLMs) to generate structured service descriptions.	AI-Driven Service Request Automation	High
REQ-7.2	The system shall generate a draft service request based on AI analysis before final submission.	AI-Driven Service Request Automation	High
REQ-7.3	The system shall allow customers to review and edit AI-generated service request drafts.	AI-Driven Service Request Automation	High
REQ-7.4	The system shall require customer confirmation before converting a draft into a final service request.	AI-Driven Service Request Automation	High
REQ-7.5	The system shall notify eligible service providers only after a service request is accepted.	AI-Driven Service Request Automation	High
REQ-7.6	The system shall maintain a stateful workflow for service request creation.	AI-Driven Service Request Automation	High
REQ-7.7	The system shall suggest suitable service time windows based on request context.	AI-Driven Service Request Automation	Medium
REQ-7.8	The system shall allow customers to modify AI-generated budget and scheduling suggestions.	AI-Driven Service Request Automation	High
REQ-7.10	The system shall orchestrate service request automation using a workflow engine.	AI-Driven Service Request Automation	High
REQ-7.11	The system shall log automation workflow steps for monitoring and analytics.	AI-Driven Service Request Automation	Medium
REQ-8.1	The system shall allow service providers to create offers on specific requests.	Offer Management	High
REQ-8.2	The system shall allow service providers to view offers they created.	Offer Management	High
REQ-8.3	The system shall allow customers to view offers they received.	Offer Management	High
REQ-8.4	The system shall support automatic offer expiration after 7 days.	Offer Management	Low
REQ-8.5	The system shall allow customers to accept an offer.	Offer Management	High
REQ-9.1	The system shall notify service providers when a new service request is created.	Notifications System	High
REQ-9.2	The system shall notify customers when they receive an offer.	Notifications System	High
REQ-9.3	The system shall notify service providers when a customer accepts an offer.	Notifications System	High
REQ-9.4	The system shall notify service providers when their offer is rejected.	Notifications System	High
REQ-9.5	The system shall mark notifications as read when viewed.	Notifications System	High
REQ-9.6	The system shall notify service providers when they receive ratings and reviews.	Notifications System	Medium
REQ-10.1	The system shall generate a preliminary AI-based cost estimation for each service request.	AI-Powered Cost Estimation	High

REQ-10.2	The system shall present cost estimates as decision-support information.	AI-Powered Cost Estimation	Medium
REQ-11.1	The system shall allow customers to rate service providers after service completion.	Trust, Rating, and Review System	High
REQ-11.2	The system shall allow customers to submit textual reviews for completed services.	Trust, Rating, and Review System	High
REQ-11.3	The system shall calculate a trust score for each service provider.	Trust, Rating, and Review System	High
REQ-11.4	The system shall associate each rating and review with a customer, provider, and request.	Trust, Rating, and Review System	High
REQ-11.5	The system shall restrict rating values to the range 1–5.	Trust, Rating, and Review System	High
REQ-11.6	The system shall enforce a maximum length for textual reviews.	Trust, Rating, and Review System	Medium
REQ-11.7	The system shall prevent duplicate reviews for the same service request.	Trust, Rating, and Review System	High
REQ-11.8	The system shall store review creation timestamps.	Trust, Rating, and Review System	Medium
REQ-11.9	The system shall store review modification timestamps.	Trust, Rating, and Review System	Medium
REQ-11.10	The system shall allow administrators to hide or show reviews.	Trust, Rating, and Review System	High
REQ-11.11	The system shall preserve hidden reviews for auditing.	Trust, Rating, and Review System	Medium
REQ-11.12	The system shall support flagging suspicious or violating reviews.	Trust, Rating, and Review System	High
REQ-11.13	The system shall allow specifying a reason for review flagging.	Trust, Rating, and Review System	Medium
REQ-11.14	The system shall compute and store sentiment scores for reviews.	Trust, Rating, and Review System	High
REQ-11.15	The system shall use sentiment scores for moderation and trust evaluation.	Trust, Rating, and Review System	Medium
REQ-11.16	The system shall maintain aggregated rating statistics per provider.	Trust, Rating, and Review System	High
REQ-11.17	The system shall calculate and store average provider ratings.	Trust, Rating, and Review System	High
REQ-11.18	The system shall store the total number of reviews per provider.	Trust, Rating, and Review System	Medium
REQ-11.19	The system shall automatically update rating aggregates.	Trust, Rating, and Review System	High
REQ-11.20	The system shall store data required for trust score computation.	Trust, Rating, and Review System	High
REQ-11.21	The system shall expose rating and review data via APIs.	Trust, Rating, and Review System	High
REQ-11.22	The system shall support weighted and AI-based trust models.	Trust, Rating, and Review System	Medium
REQ-12.1	The system shall provide a chatbot interface for service request creation.	Conversational Support	High

REQ-12.2	The chatbot shall guide users through request creation and approval.	Conversational Support	High
REQ-12.3	The chatbot shall respond to common service-related inquiries.	Conversational Support	Medium
REQ-13.1	The system shall support pagination.	Additional Requirements	High
REQ-13.2	The system shall classify service requests using image-based deep learning models with high accuracy.	Additional Requirements	High

4. Requirements modeling

3.2 Basic UML Diagrams

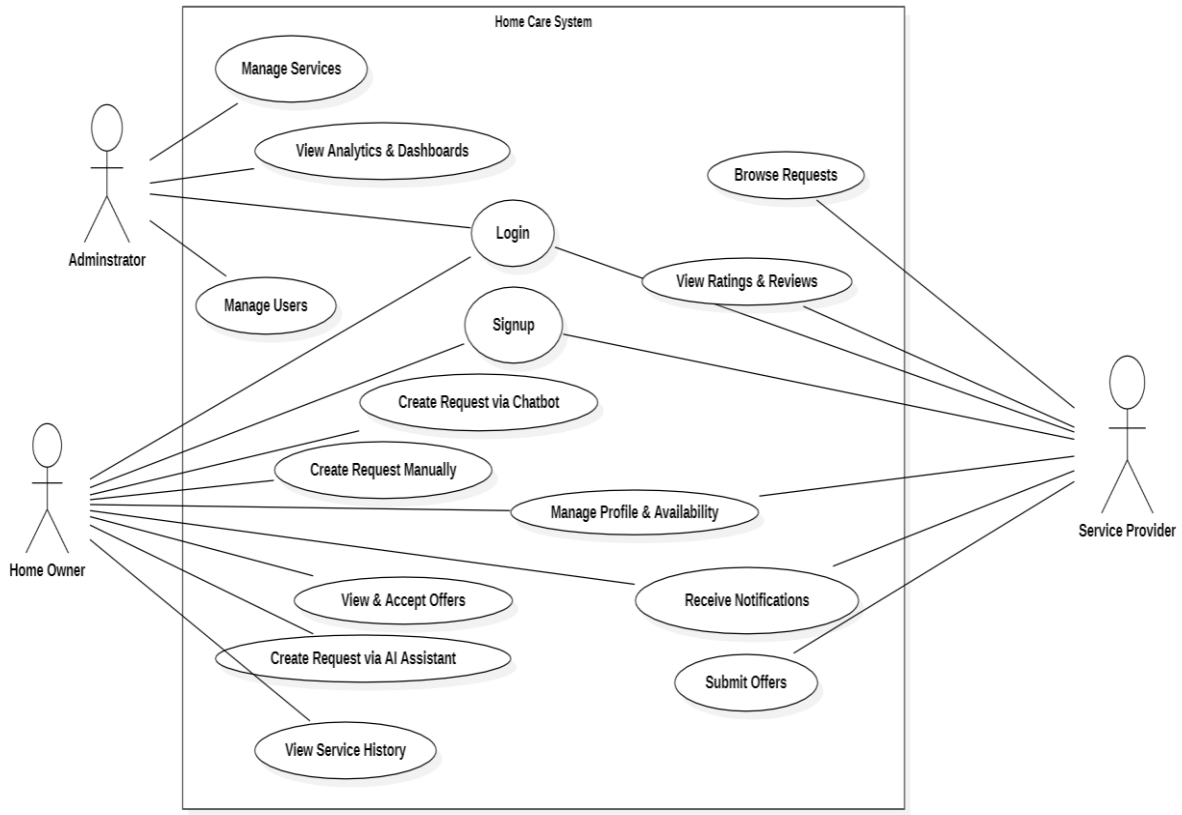


Figure 6 use case diagram

System features (use case specifications - Diagrams):

- **Registration**

Table 4 register

Use Case Name	**Register**
Actor	Client
Precondition	The customer must have access to the application or website.
Main Scenario	1. The customer launches the application and opens the registration page. 2. The customer fills in the required details (e.g., name, email, password, and other relevant fields). 3. The customer clicks on the "Register" button. 4. The system validates the input. 5. The system creates the account and displays a success message.

Alternative Scenario	If the user provides invalid or incomplete input (e.g., missing fields, weak password, duplicate email), the system displays an appropriate error message and prompts the user to correct the input.
Postcondition	The customer account is created successfully, and the user can proceed to log in.

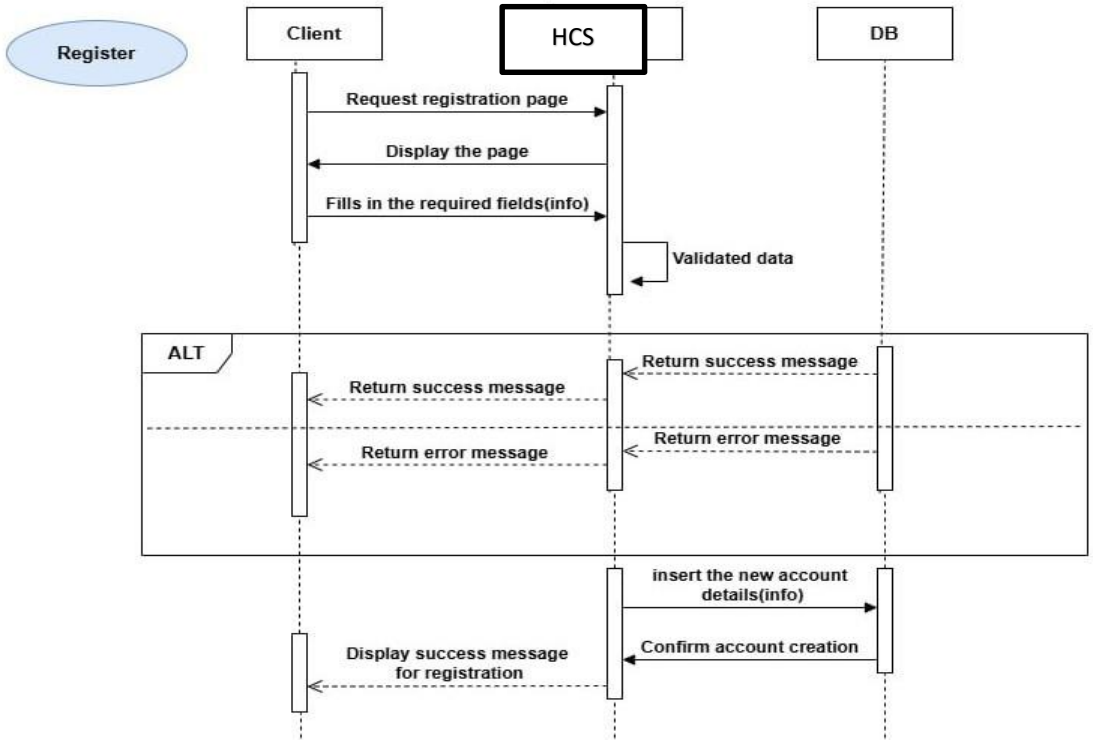


Figure 7 register sequence

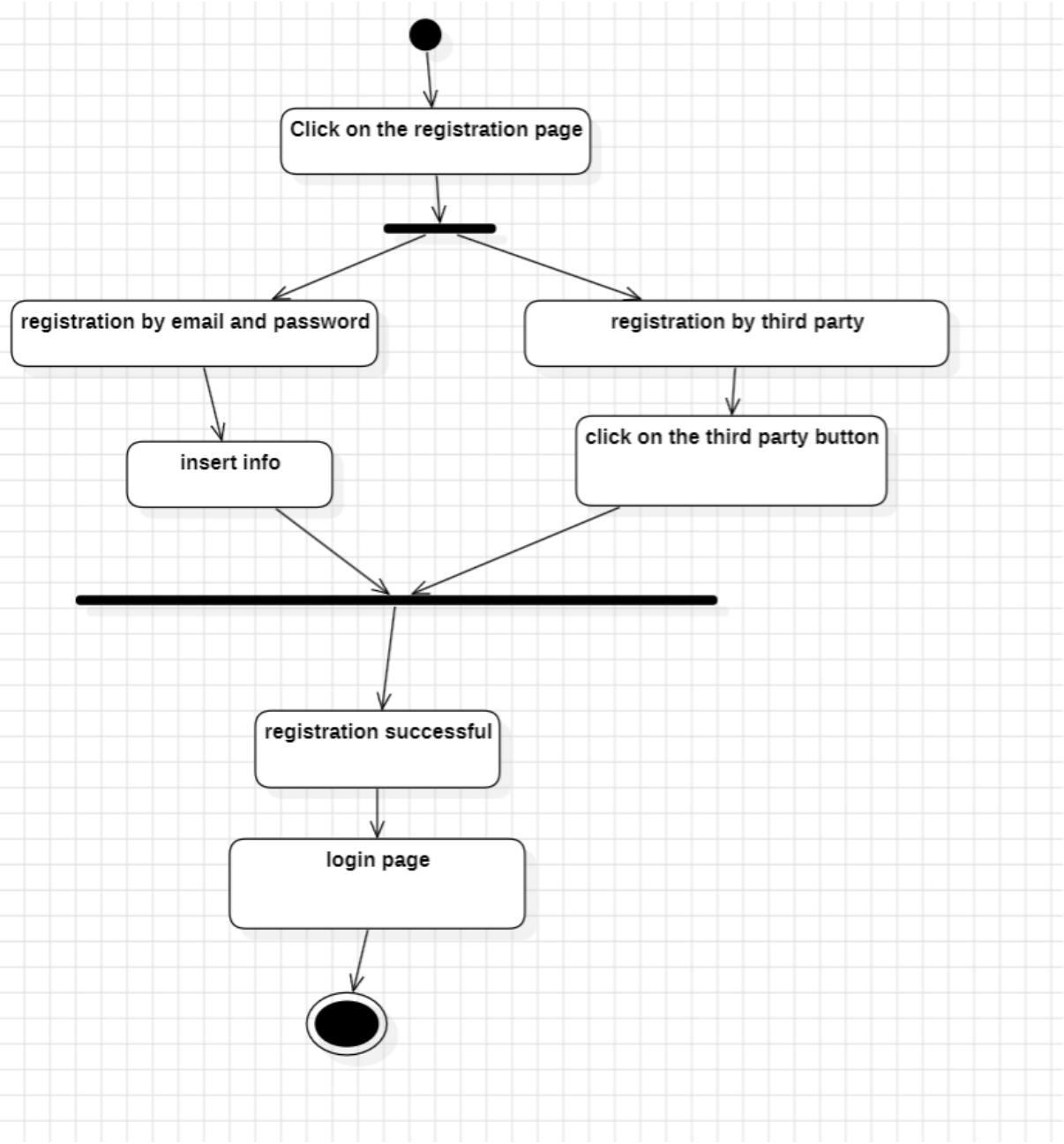


Figure 8 register activity

- **log in**

Table 5 log in

Use Case Name	**log in**
Actor	All users
Precondition	The user must have access to the application or website.
Main Scenario	1. The user launches the application and opens the login page. 2. The user enters their email and password. 3. The user clicks on the "Sign in" button. 4. The system validates the entered credentials. The system logs the user in successfully.
Alternative Scenario	If the email or password is incorrect, the system displays an error message ("Invalid credentials. Please try again.") and does not log the user in.
Postcondition	The user is successfully logged into the system.

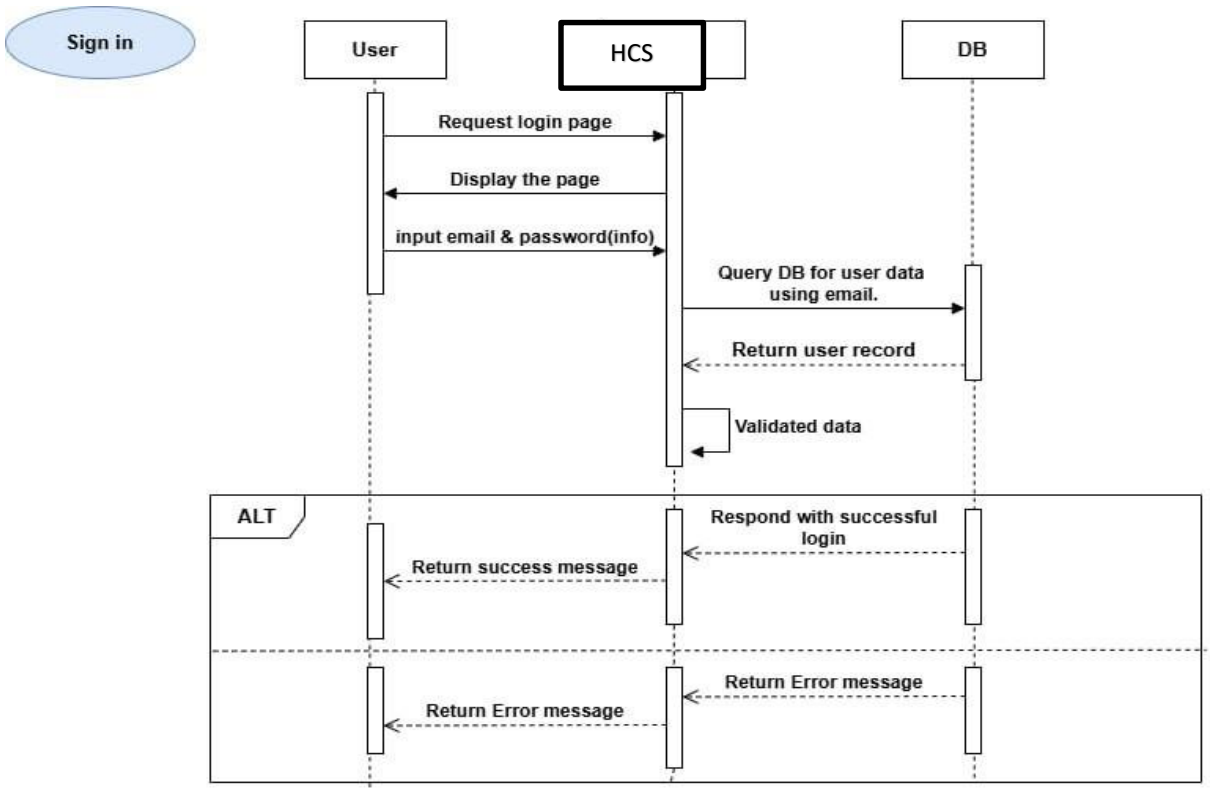


Figure 9 log in sequence

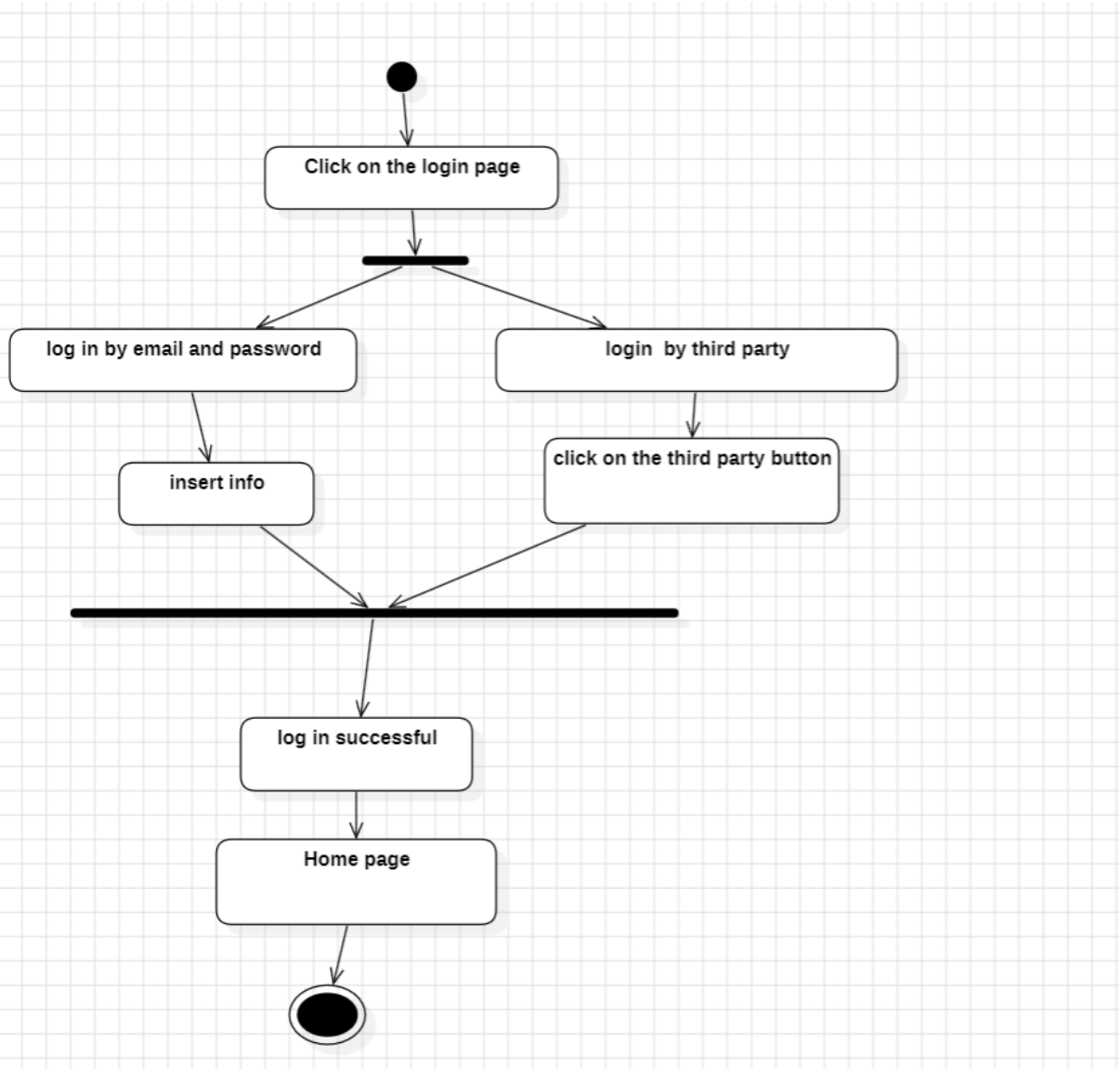


Figure 10 log in activity

- **Account info management**

Table 6 account info mgmt.

Use case name	**Account Info management**
Description	This use case allows users to manage their account information.
Actors	All users
Preconditions	user must be logged into the system.
Main Scenario	▪ User navigates to the **Profile information page.** **A. If User request to update its own account:** 1. The user navigates to the **Profile information** page. 2. The system displays the user's current personal information. 3. The user modifies the desired fields directly on the page. 4. The user clicks the **"Edit My Profile"** button to save the changes. 5. The system validates the updated information for correctness and completeness. 6. After successful validation, the system saves the updated information in the database. 7. The system displays a success message: **"Your account information has been updated successfully."** **B. If User request to delete its own account:** 1. The user navigates to the **Account Info Management** page. 2. The user clicks the **"Delete Account"** button. 3. The system deletes the user's account from the database. 4. The system logs the user out automatically. 5. The system displays a success message: **"Your account has been successfully deleted."**
Alternative Scenario	**Invalid Input:** 1. If the user enters invalid information (e.g., an incorrectly formatted email or missing required fields), the system highlights the errors and displays an error message. 2. The user revises the information and clicks the **"Edit My Profile"** button again to save the changes.
Postconditions	The user's account information is successfully updated or deleted based on the action performed.

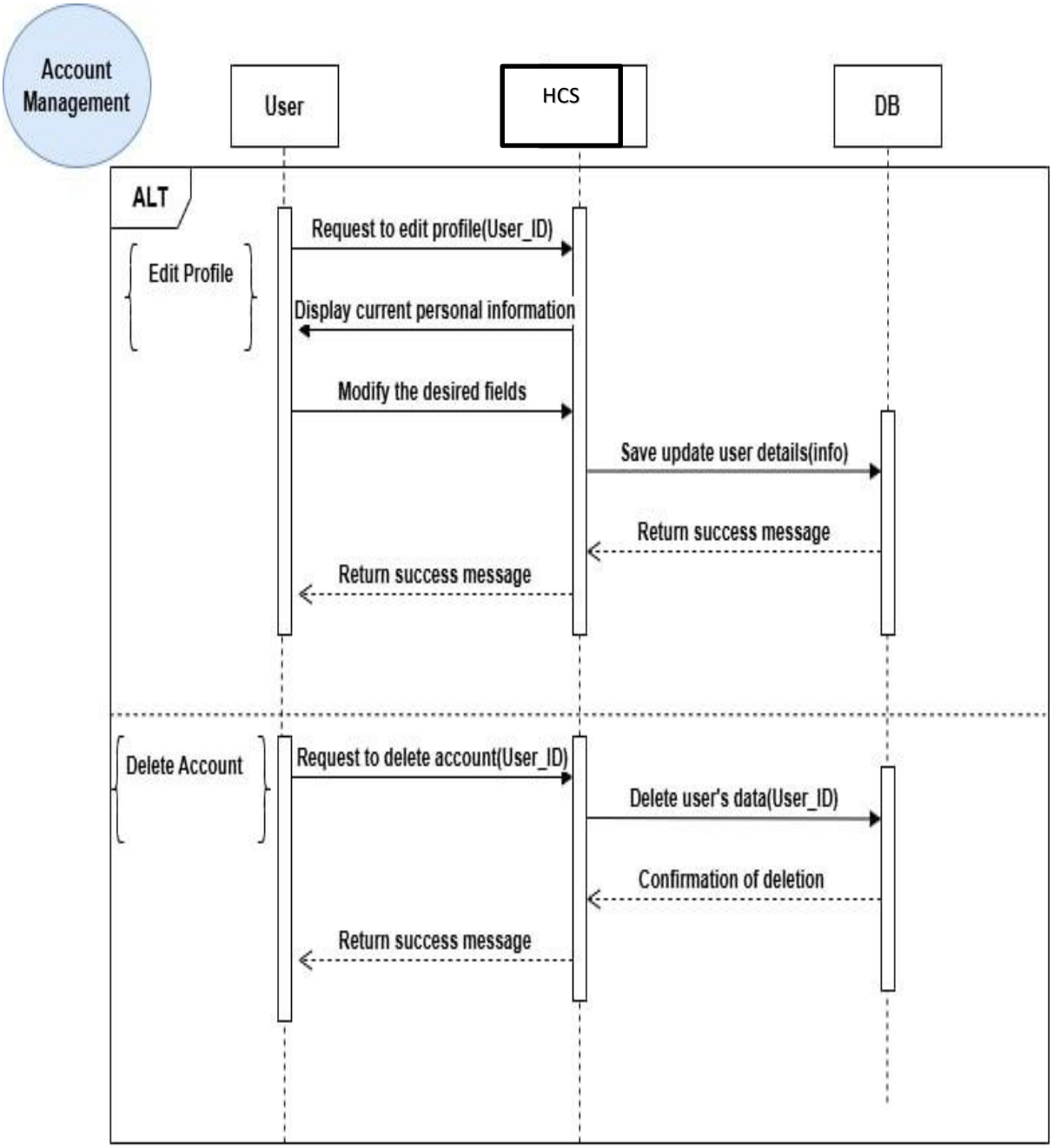


Figure 11 account info mgmt. sequence

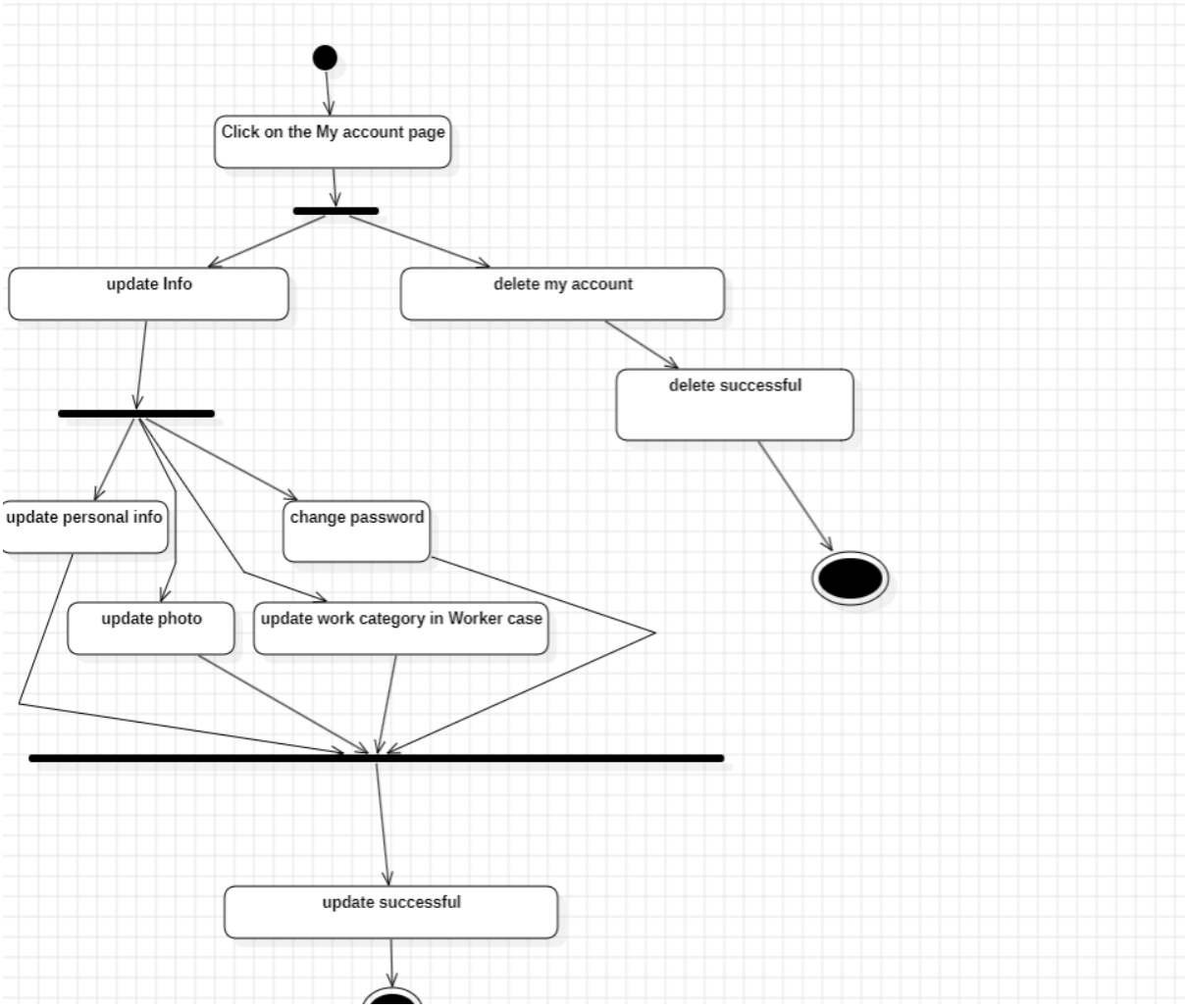


Figure 12 account info mgmt. activity

- **Submit Custom Service Offer**

Table 7 submit custom service offer

Use case name	**Submit Custom Service Offer**
Description	This use case Allow a service provider to respond to a service request with a custom offer (price, time, and optional notes)..
Actors	Service Provider
Preconditions	A service request has been received and the service provider is eligible to respond.

Main Scenario	1. Service provider receives notification about a new service request. 2. Opens request details (description, images, customer city, preferred time). 3. Chooses to create a service offer. 4. Fills out offer form: Proposed price, estimated time to start, Optional note/message. 5. System stores the offer linked to the request and service provider. 6. System notifies the customer about the new offer.
Alternative Scenario	• If the service provider closes the request without offering, no data is saved. • If system fails to store the offer, service provider is notified with an error message.
Postconditions	A new offer is submitted and stored; the customer can now view it in their offers list.

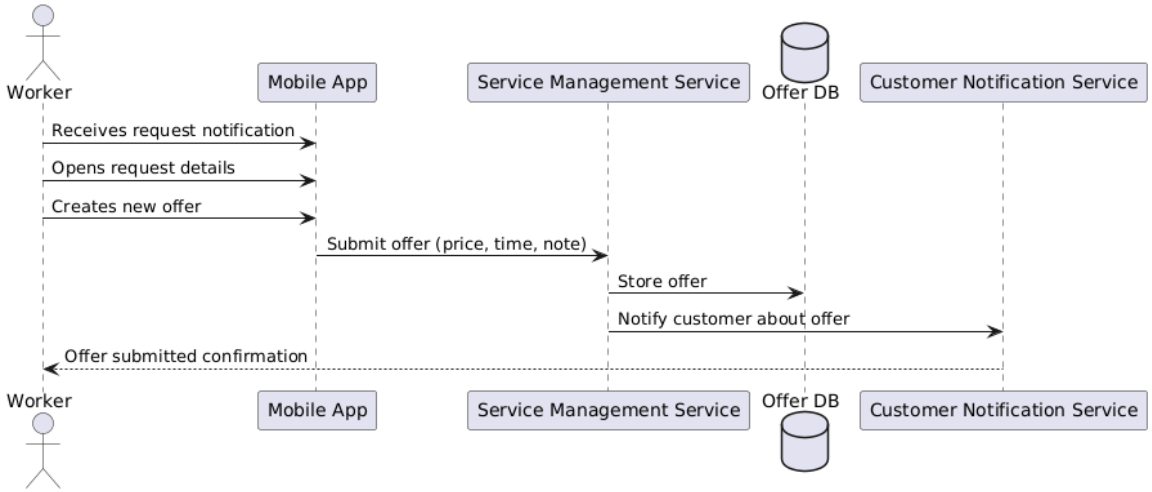


Figure 13 submit custom service offer sequence

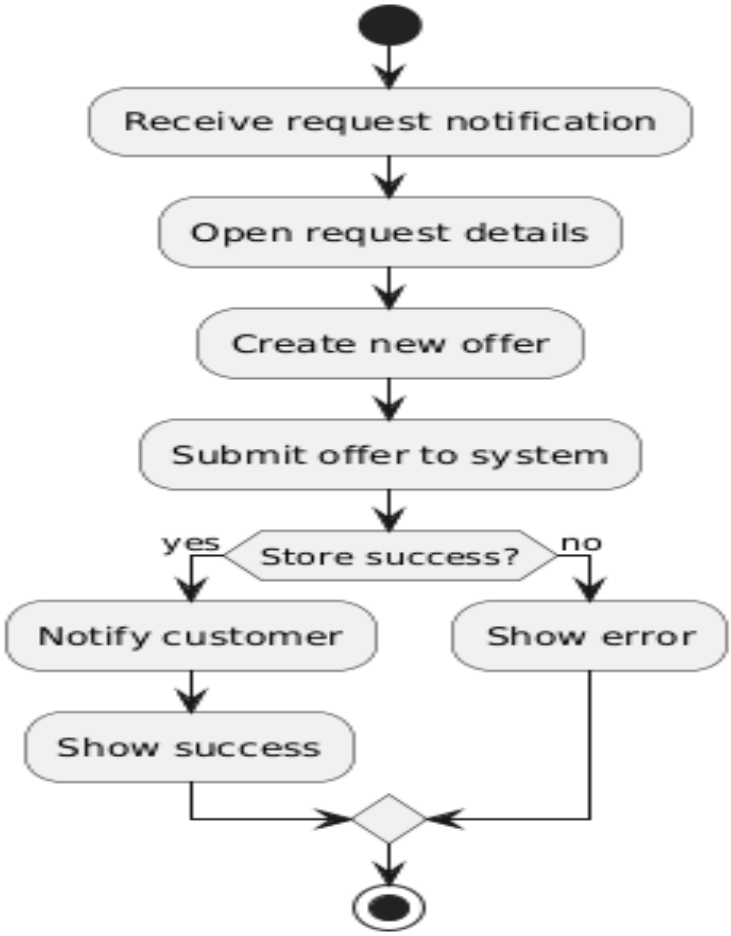


Figure 14 submit custom service offer activity

- **View Available Services**

Table 8 view available services

Use Case Name	**View Available Services**
Description	Let customers browse the list of services available in the system.
Actor	Customer
Precondition	Customer is authenticated.
Main Scenario	1. Customer opens the app. 2. Navigates to 'Available Services'. 3. System fetches the service categories from the database. 4. Customer browses and selects a service to view more information.
Alternative Scenario	• If no services are available, system displays a friendly empty state.
Postcondition	Customer is informed of available services and can proceed to request one.

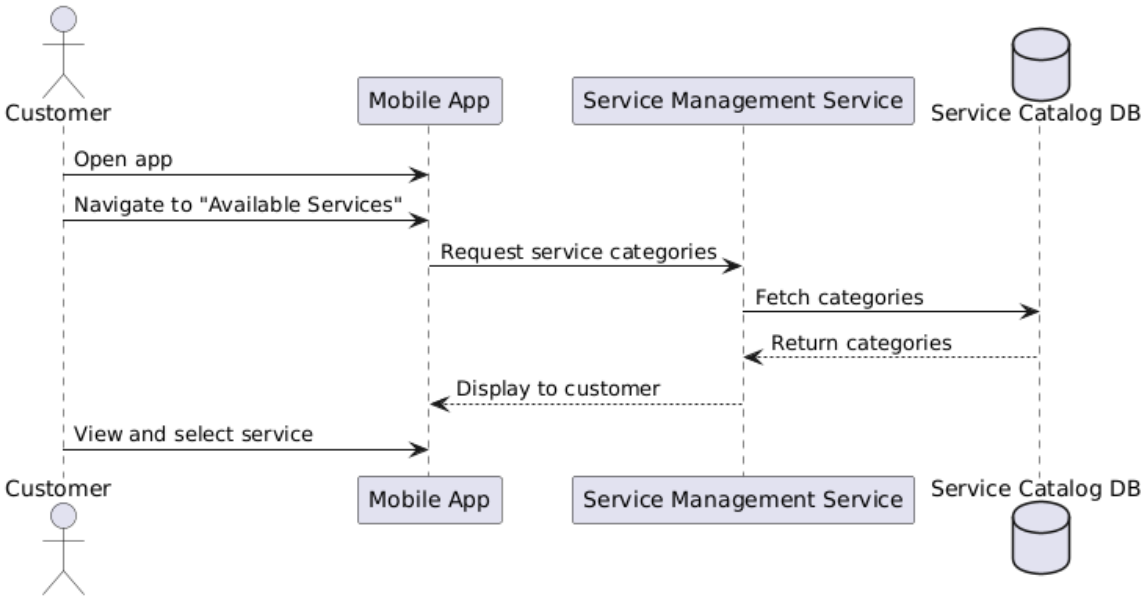


Figure 15 view available services sequence



Figure 16 view available services activity

- **Submit Service Request Manually**

Table 9 submit service request manually

Use Case Name	**Submit Service Request Manually**
Description	Allow the customer to submit a new service request.
Actor	Customer
Precondition	Customer is authenticated and has browsed/select a service.
Main Scenario	1. Customer selects a service from the available list. 2. Customer reaches the Description field and chooses to Upload Image for Auto-Description. 3. Customer uploads an image 4. System sends the image to the **VLM Service**. 5. VLM Service returns a generated description; System populates the description field. 6. Customer reviews/edits the generated text and selects a preferred time. 7. System stores the request and image in the Service Request table. 8. System identifies and notifies nearby eligible service providers.
Alternative Scenario	• **If image upload fails**, system retries or continues without the image. • **If no service provider is found**, system queues request for retry or future match. • **Manual Entry:** Customer bypasses AI and types the description directly. • **AI Generation Fails:** If the VLM service is unavailable or fails to analyze the image, the system prompts the customer to enter the description manually.
Postcondition	A service request is created and dispatched to service providers.

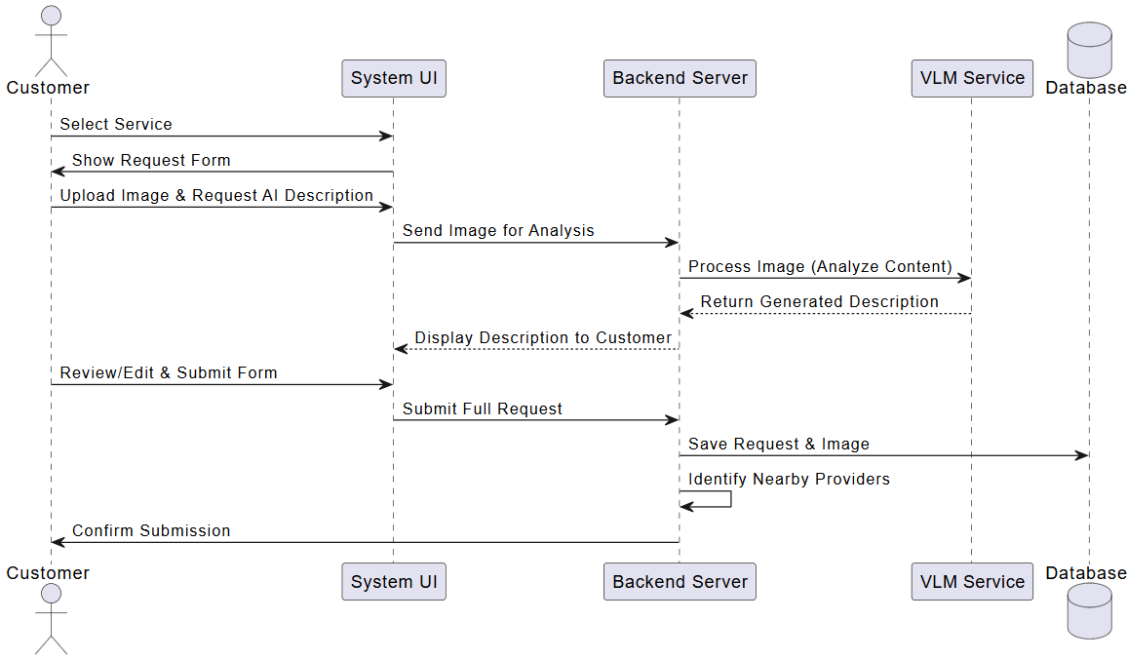


Figure 17 submit service request manually sequence

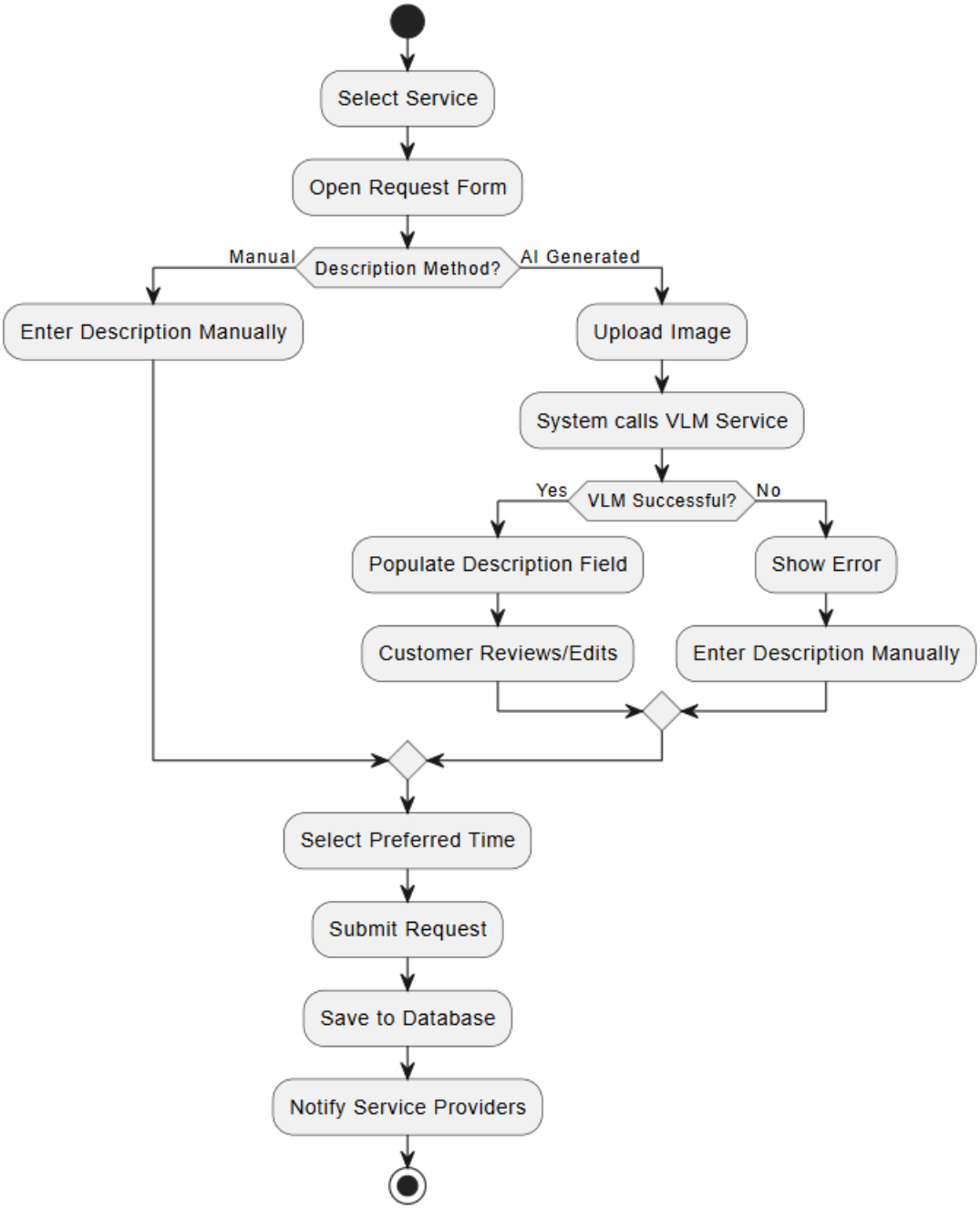


Figure 18 submit service request manually activity

- **Create Service Request via AI Assistant**

Table 10 create service request via ai-assistant

Use Case Name	**Create Service Request via AI Assistant**
Actor	Customer
Precondition	The customer must be authenticated and have access to the service request feature.
Main Scenario	1. The customer opens the service request page. 2. The customer uploads one or more images of the issue.

	3. The customer provides optional contextual information (e.g., location, urgency). 4. The customer initiates AI analysis. 5. The system analyzes images using Vision-Language Models (VLMs). 6. The system generates a draft service request (description, category, estimated cost, suggested time). 7. The system displays the generated draft to the customer.
Alternative Scenario	• If image analysis fails or images are invalid, the system displays an error message and requests new input.
Postcondition	A draft service request is created and stored in **Draft** state.

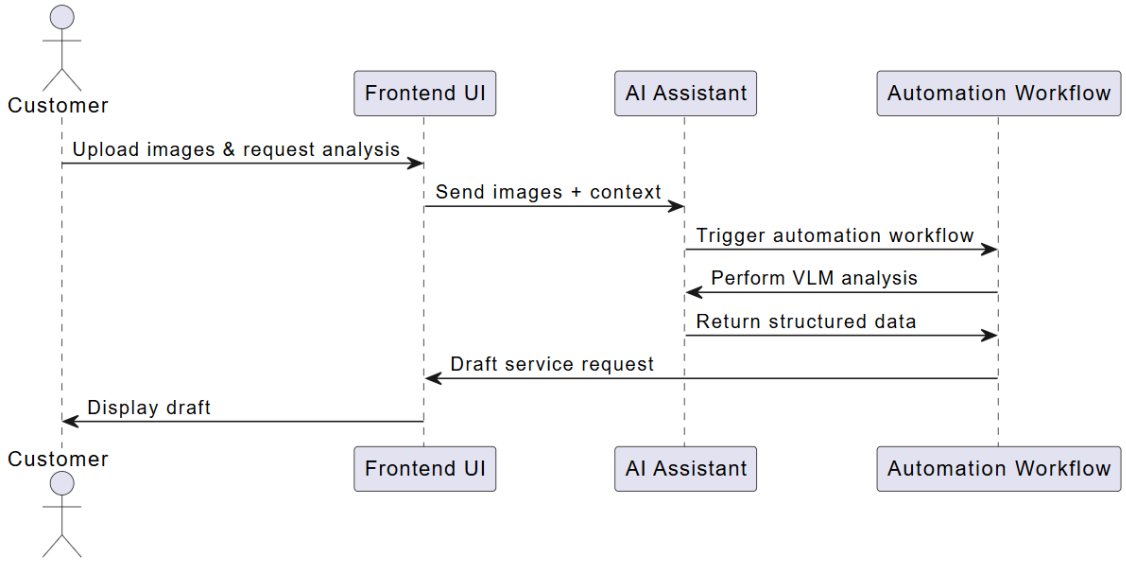


Figure 19 create service request via ai-assistant sequence

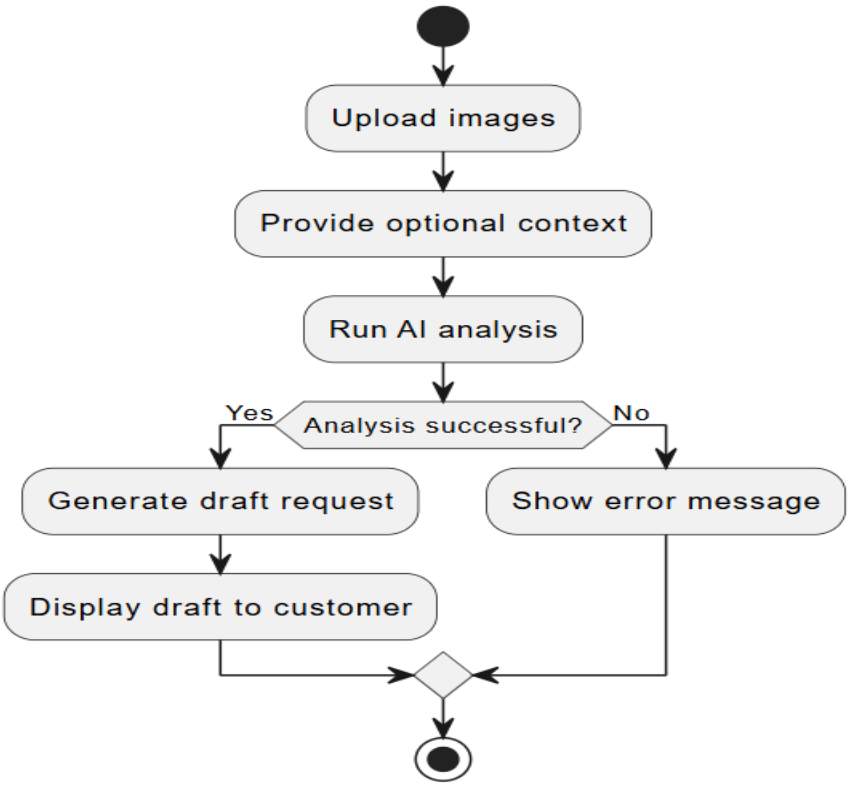


Figure 20 create service request via ai-assistant activity

- **Create Service Request via Chatbot**

Table 11 create service request via chatbot

Use Case Name	**Create Service Request via Chatbot**
Actor	Customer
Precondition	The customer must be authenticated and have access to the chatbot interface.
Main Scenario	1. The customer opens the chatbot interface. 2. The customer describes the issue using natural language. 3. The chatbot asks follow-up questions (images, urgency, location). 4. The customer uploads images if requested. 5. The chatbot sends collected data to the AI automation engine. 6. The system generates a draft service request. 7. The chatbot presents the draft for review.
Alternative Scenario	• If the customer provides unclear or incomplete information, the chatbot asks clarification questions.
Postcondition	A draft service request is created in **Draft** state via conversational input.

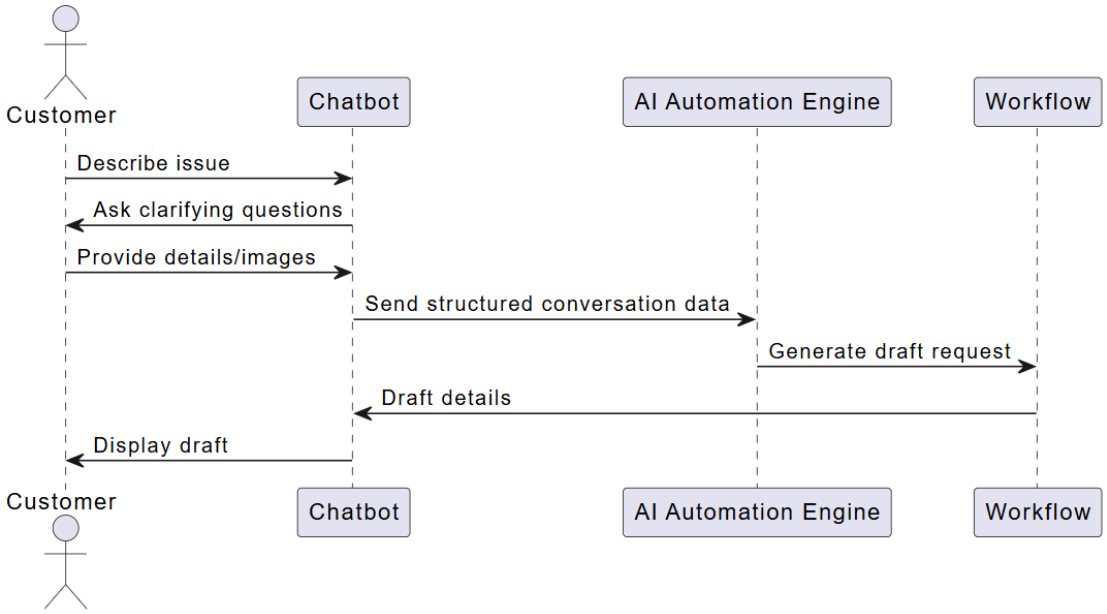


Figure 21 create service request via chatbot sequence

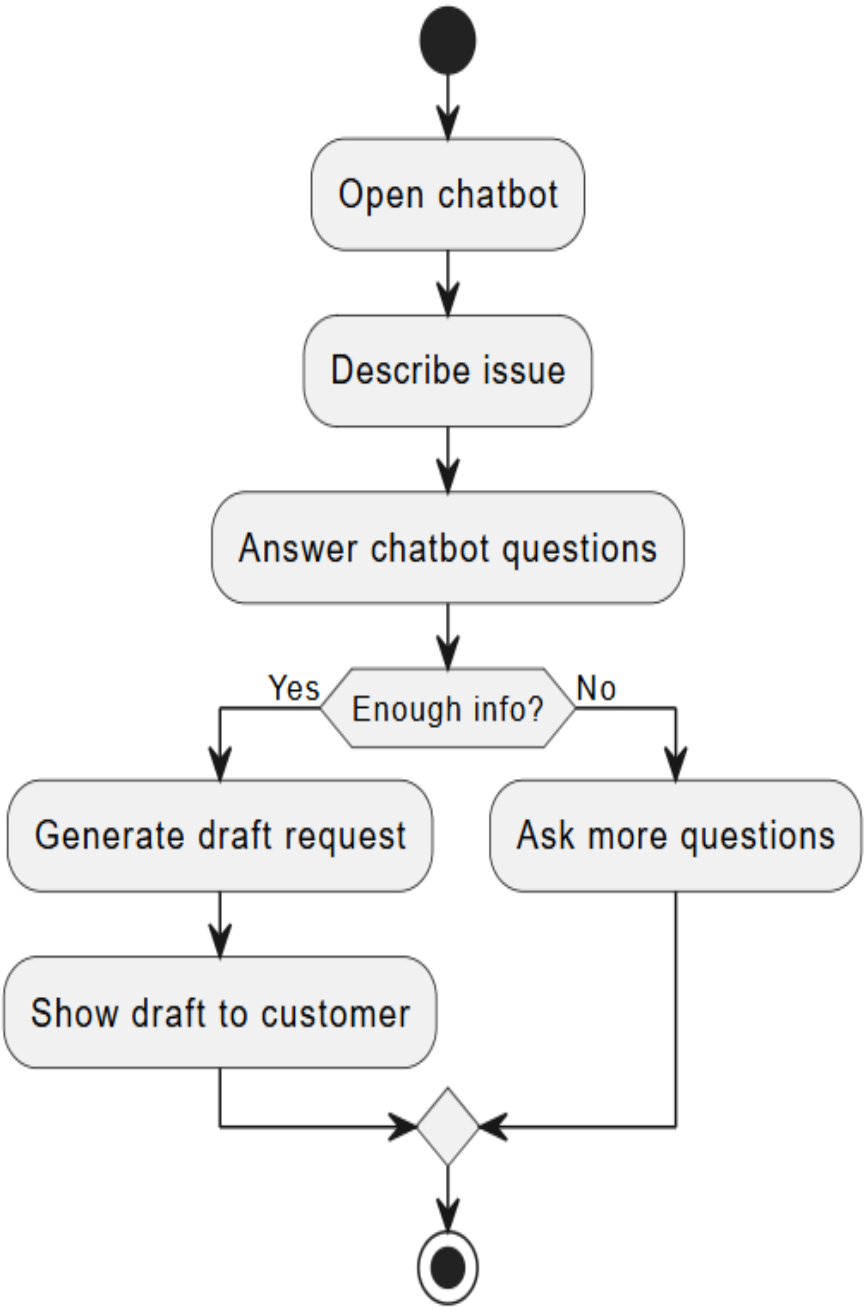


Figure 22 create service request via chatbot

- **Update & Accept Service Request Draft**

Table 12 update & accept service request draft

Use Case Name	Update and Accept Service Request Draft
Actor	Customer
Precondition	A draft service request must already exist.
Main Scenario	1. The customer views the draft service request. 2. The customer edits the description, budget, or suggested time if needed. 3. The system updates the draft request. 4. The customer confirms (Accept) the request. 5. The system finalizes the request. 6. The system notifies eligible service providers.
Alternative Scenario	• If the customer cancels the process, the draft remains saved but not finalized.
Postcondition	The service request is finalized and becomes visible to service providers.

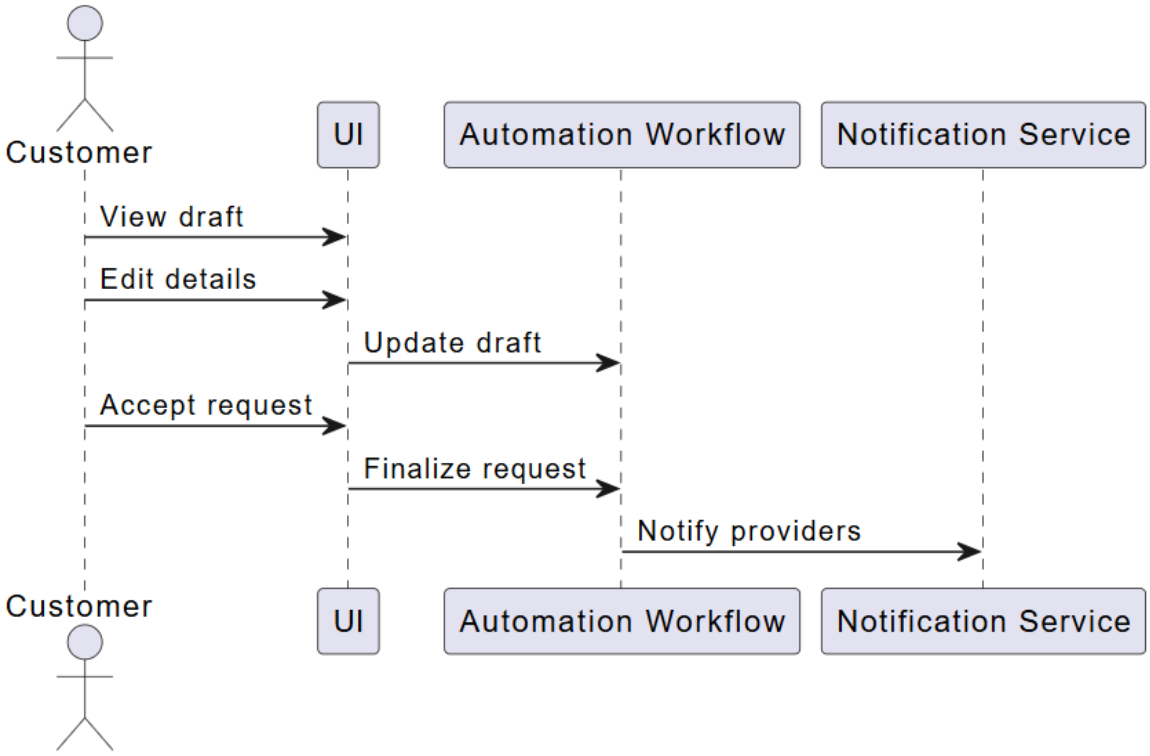


Figure 23 update & accept service request draft sequence

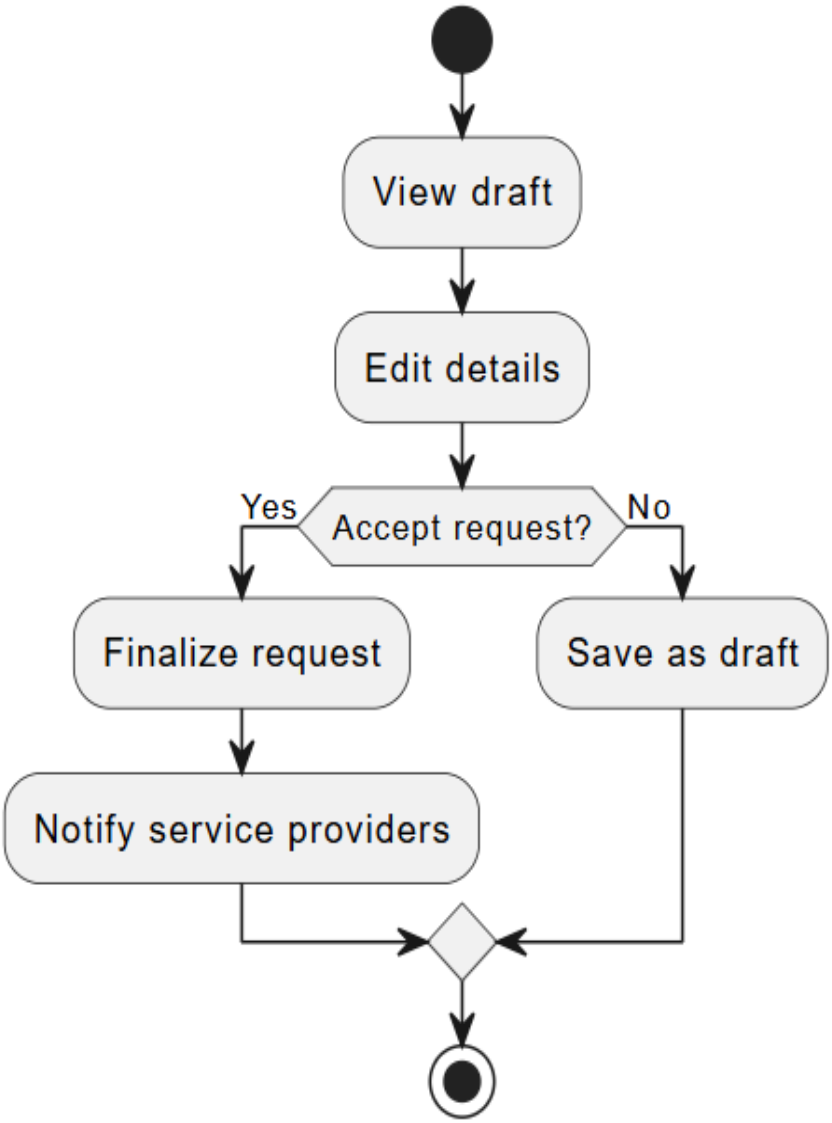


Figure 24 update & accept service request draft activity

- **View Requests**

Table 13 view requests

Use Case Name	**View Requests**
Description	Allow customers to track and review their service history.
Actor	Customer
Precondition	Customer Dashboard Features
Main Scenario	1. Customer opens 'My Requests'. 2. System retrieves a list of past and current service requests with statuses. 3. Customer selects any request to view details (problem description, images, assigned service provider, status). 4. If the request is completed, the customer may submit a rating and review.
Alternative Scenario	• If no requests exist, system displays a message like 'You haven't requested any services yet.
Postcondition	Customer has access to a full request history and may engage in post-service feedback.

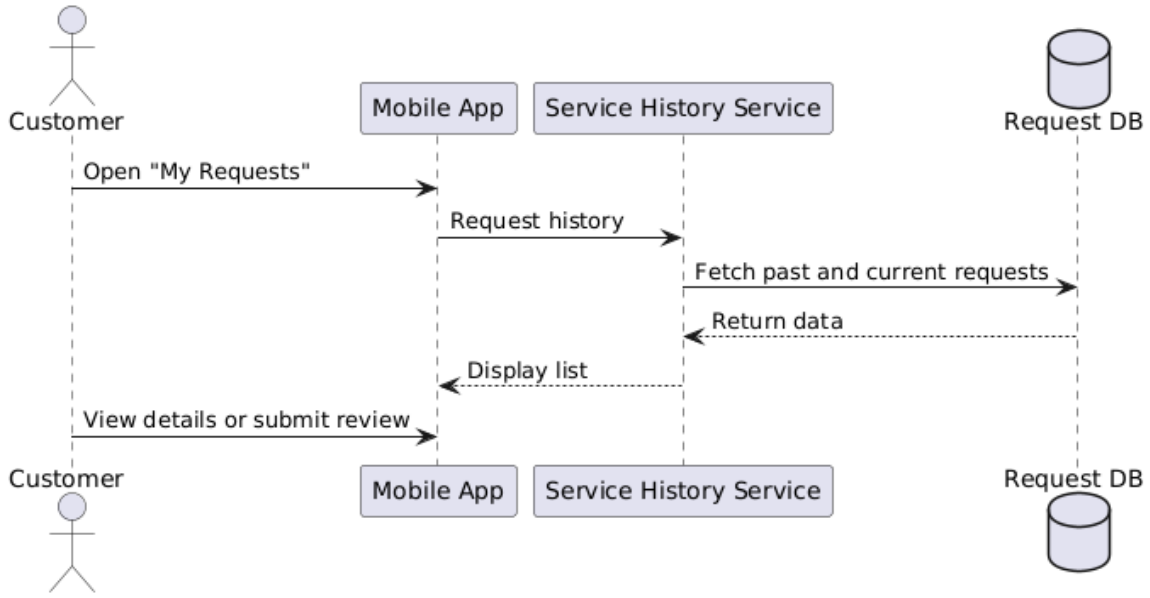


Figure 25 view requests sequence

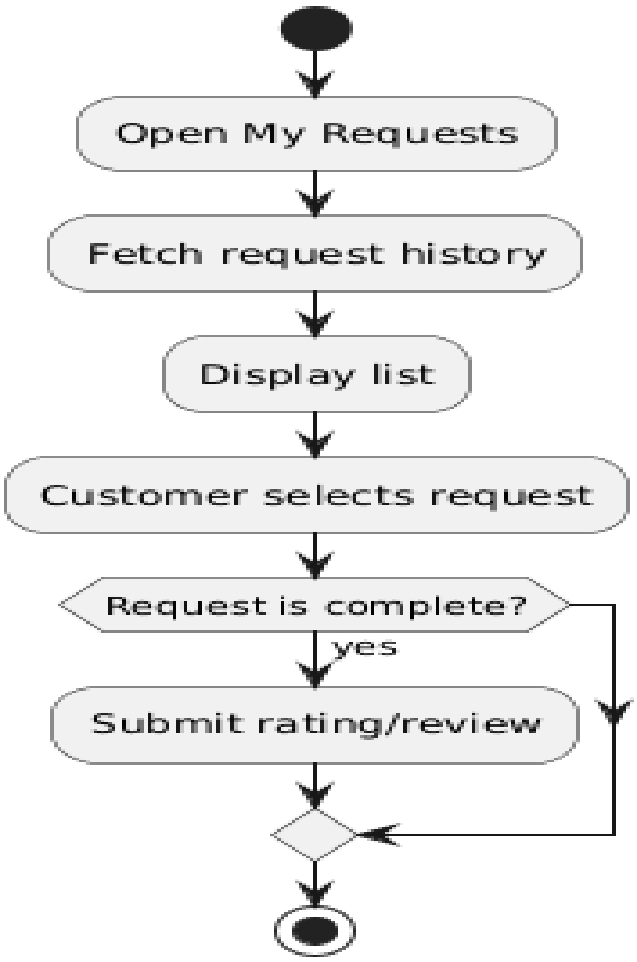


Figure 26 view requests activity

- **Review and Accept Service Provider Offer**

Table 14 Review and Accept Service Provider Offer

Use Case Name	**Review and Accept Service provider Offer**
Description	Allow the customer to view offers from service providers and accept one to proceed with service.
Actor	Customer
Precondition	Customer has submitted a service request; at least one service provider offer exists.
Main Scenario	1. Customer opens the app and navigates to their active service request. 2. System fetches all submitted offers for the request. 3. Customer reviews each offer (price, estimated time, service provider profile, notes). 4. Customer selects one offer and confirms it. 5. System updates the request: Marks the selected offer as Accepted and assigns the request to the corresponding service provider. 6. Other service providers are notified that their offers were not selected.
Alternative Scenario	• If no offers are submitted within a time limit, system suggests customer retry or cancel request. • If customer cancels before accepting, system closes the request and archives pending offers.
Postcondition	The selected service provider is assigned, and the request is now in progress

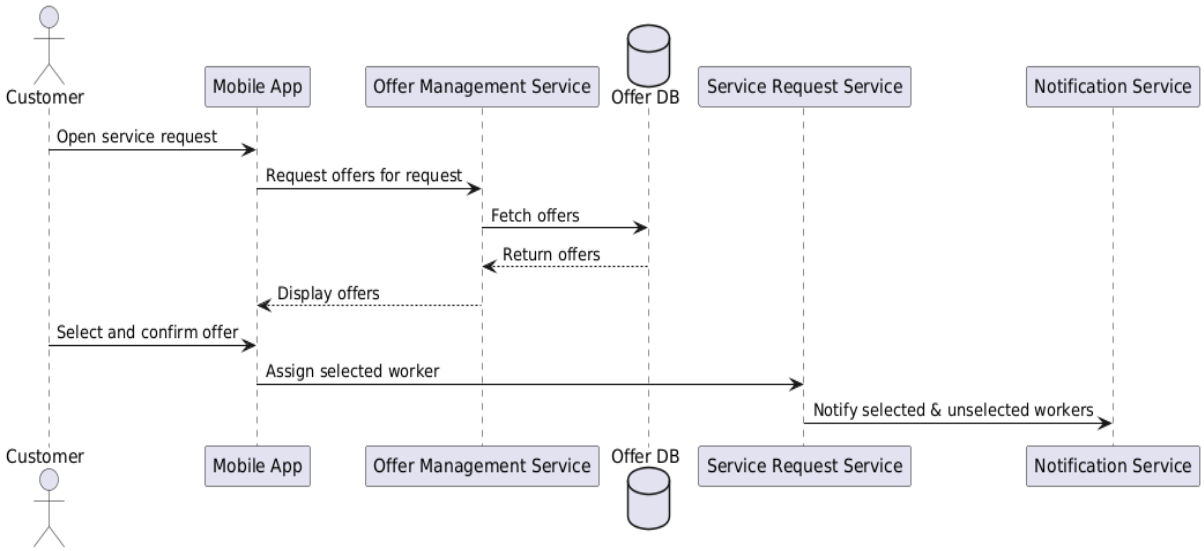


Figure 27 Review and Accept Service Provider Offer sequence

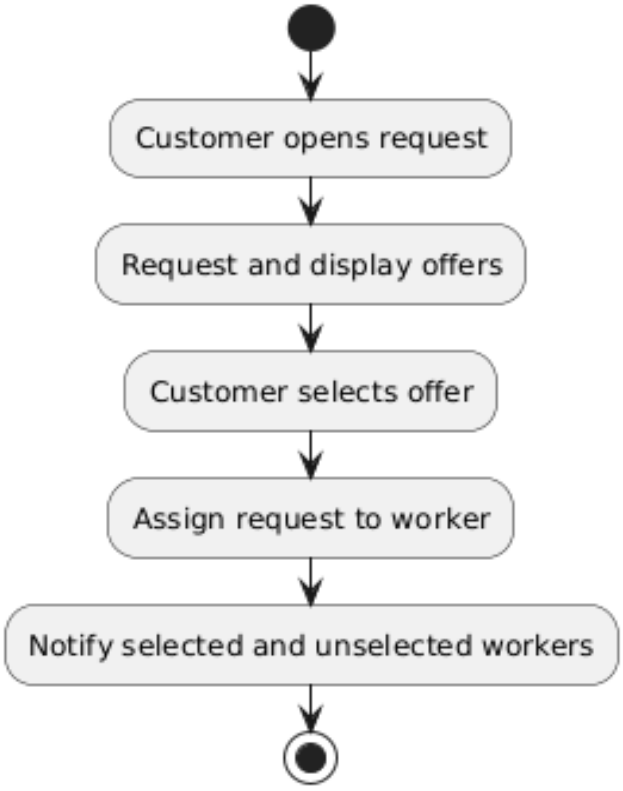


Figure 28 Review and Accept Service Provider Offer activity

- **Submit Rating & Review**

Table 15 submit rating & review

Use Case Name	**Submit Rating & Review**
Actor	Customer
Precondition	The service request is completed (status = completed). Customer is authenticated and was the requester for that service.
Main Scenario	1. Customer opens the completed request detail. 2. Customer clicks "Rate & Review". 3. Customer selects rating (1–5). 4. Customer optionally enters review text (max length enforced). 5. Customer submits. 6. API validates input, verifies request completion and customer ownership. 7. Business logic checks duplicate review constraint (one review per request). 8. System stores the review and rating in ratings table. 9. System updates provider aggregate (provider average, count). 10. System publishes notification to Notification Service and publishes aggregate to Analytics Service. 11. System returns success confirmation to customer.
Alternative Scenario(s)	• If customer is not owner of request or request not completed → API returns error (403/400). • If rating is out of bounds or text too long → validation error, show form errors. • If duplicate review exists → show message with option to edit existing review. • If external Notification fails → accept review save; queue.
Postcondition	• A new rating/review record exists. • provider aggregate updated. • provider notified • customer shown confirmation.

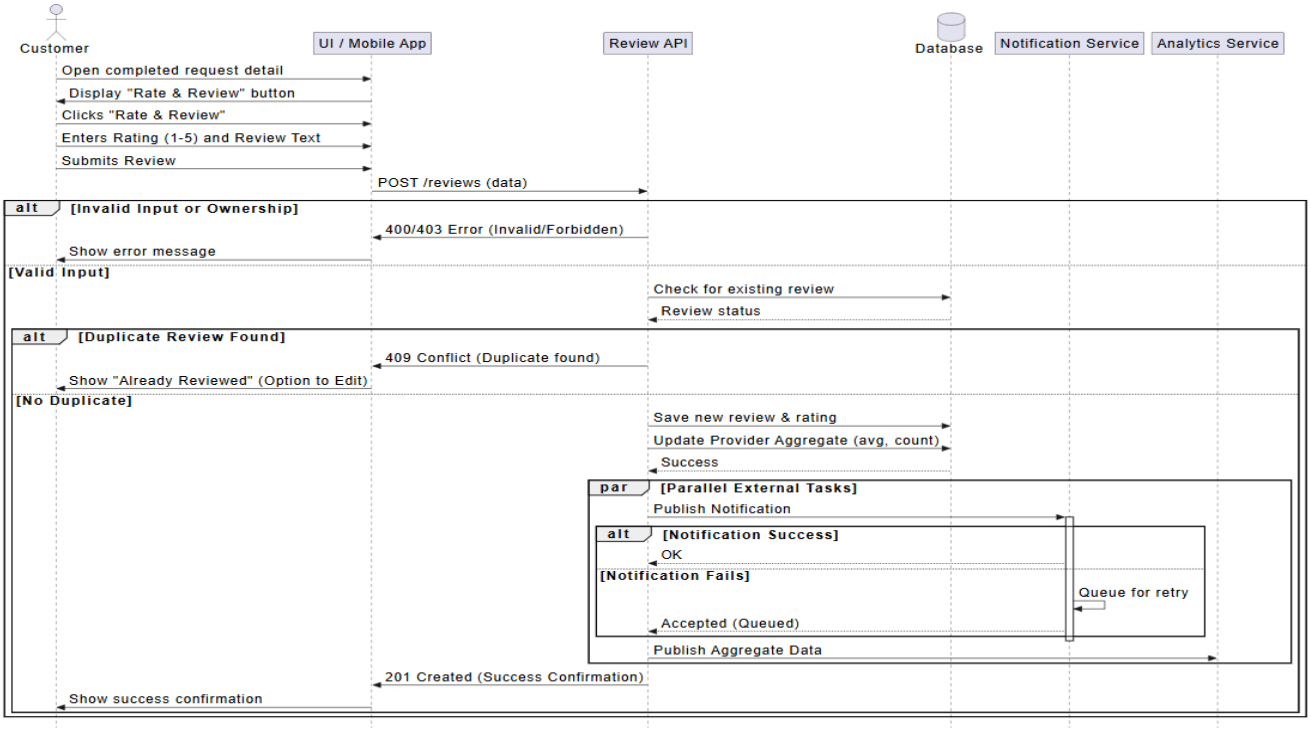


Figure 29 submit rating & review sequence

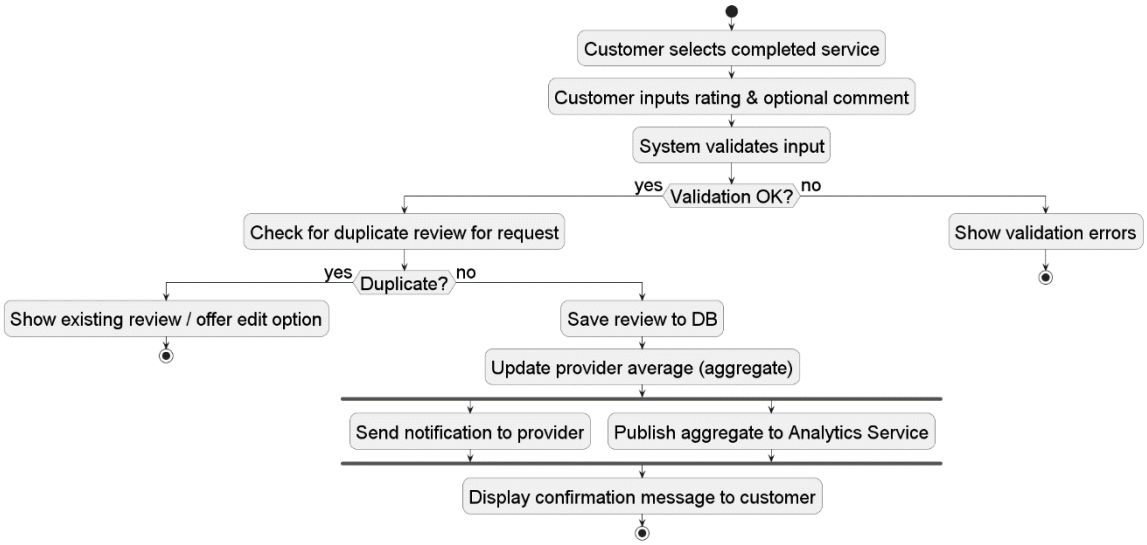


Figure 30 submit rating & review activity

- **View Provider Ratings**

Table 16 view provider ratings

Use Case Name	View Provider Ratings
Actor	Provider
Precondition	Provider profile exists. ProviderAggregate record may exist (or be computed on demand).
Main Scenario	1. User navigates to provider profile page. 2. UI calls Rating & Review API: /providers/{id}/reviews and /providers/{id}/aggregate. 3. API fetches provider aggregate (average rating, total reviews) from provider_aggregates or computes on the fly via Repository. 4. API returns: average rating, rating breakdown, paginated review list, visibility flags. 5. UI displays aggregate, rating histogram, and paginated list of visible reviews sorted by recency or helpfulness.
Alternative Scenario(s)	• If provider has no reviews → display "No reviews yet" state and encourage feedback. • If cached aggregate missing → repository computes aggregate from ratings table and caches result.
Postcondition	User sees provider average and paginated reviews.

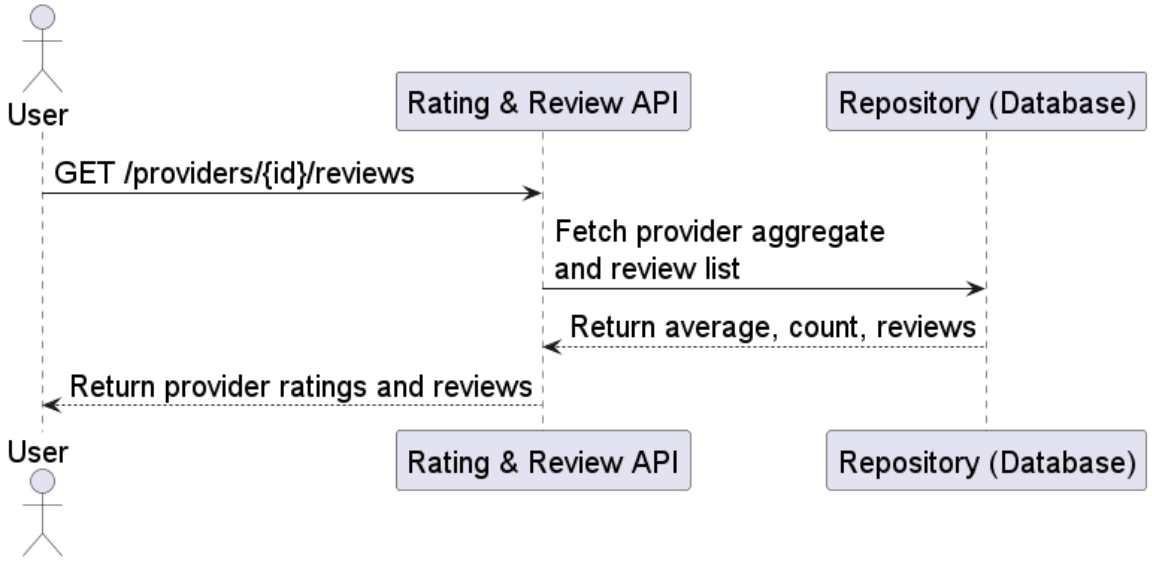


Figure 31 view provider ratings sequence

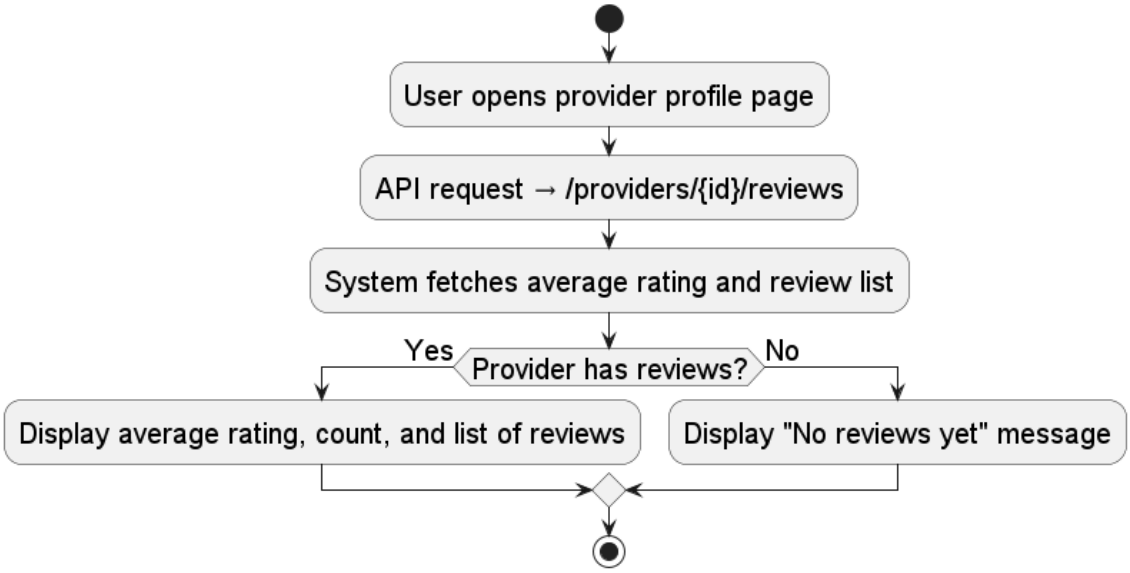


Figure 32 view provider ratings activity

- **Edit / Delete Own Review**

Table 17 edit / delete own review

Use Case Name	**Edit / Delete Own Review**
Actor	Customer
Precondition	Customer authored a review for the request; the allowed edit/delete window has not expired (policy configurable). Customer authenticated.
Main Scenario	1. Customer views their review on request or profile. 2. Customer chooses Edit or Delete. 3. For Edit: customer modifies rating/text and submits. API validates, verifies ownership and time-window, updates review record and timestamps. 4. For Delete: customer confirms deletion; API marks review as deleted or removes it depending on retention policy. 5. Business Logic triggers provider aggregate recalculation and notifies Analytics Service. 6. System returns confirmation to customer.
Alternative Scenario(s)	• If customer attempts to edit after allowed time-window → API rejects with message and suggests contacting support. • If external services unavailable → perform local update and schedule async aggregate/notification updates (with retry).
Postcondition	Review updated or deleted; provider aggregate updated; audit trail preserved (if soft-delete).

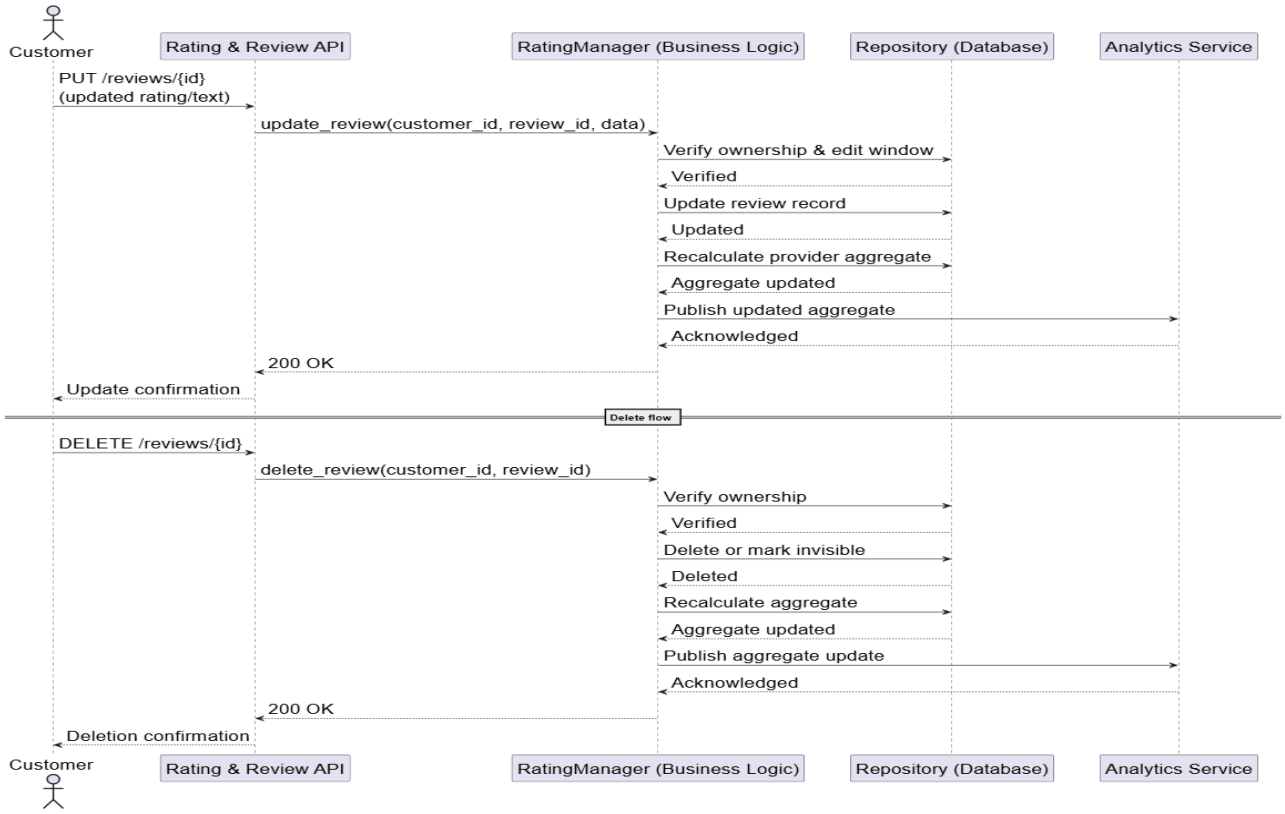


Figure 33 edit / delete own review sequence

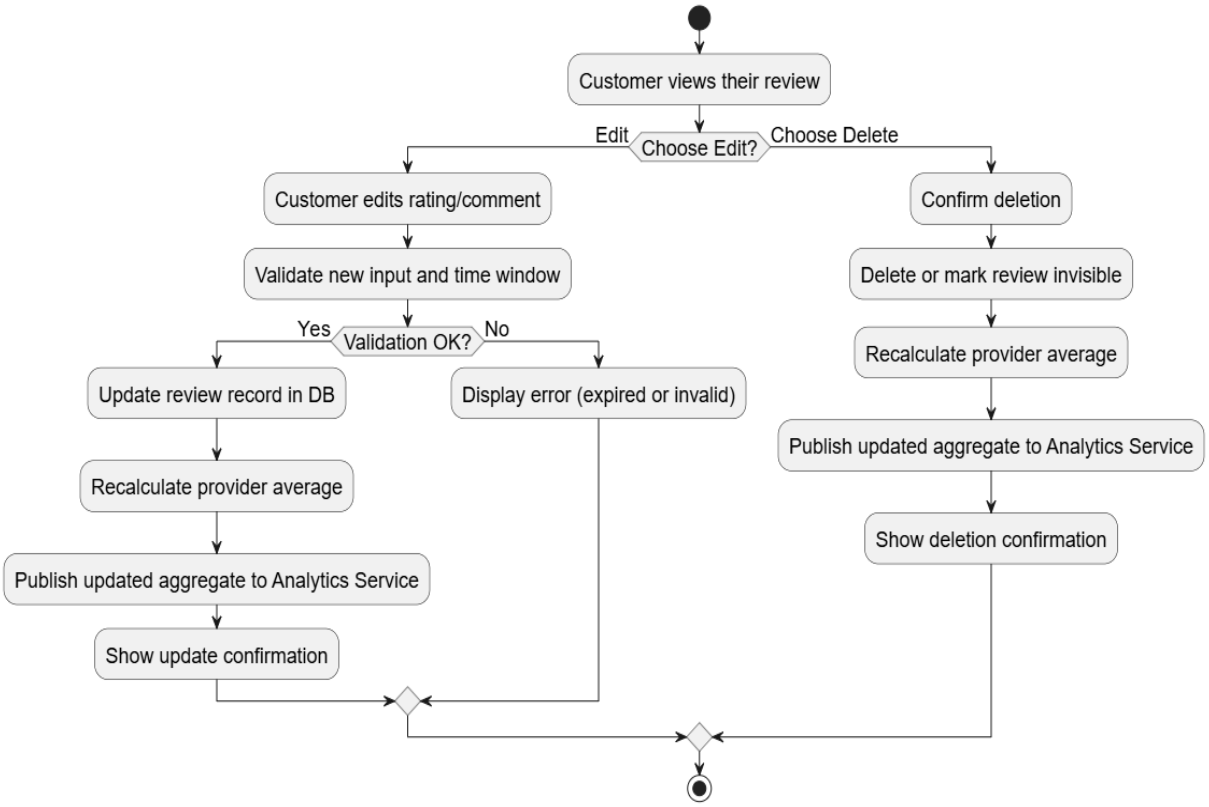


Figure 34 edit / delete own review activity

- **Admin Moderation**

Table 18 admin moderation

Use Case Name	**Admin Moderation**
Actor	Admin
Precondition	Admin authenticated with moderation privileges; reviews exist with visibility flags.
Main Scenario	1. Admin views flagged/inappropriate reviews (via moderation UI). 2. Admin selects a review and chooses Hide or Delete. 3. System marks review as is_visible=false (hide) or deletes/archives per retention policy. 4. System triggers provider aggregate recalculation and notifies Analytics Service. 5. System logs moderation action (who, when, reason). 6. UI shows moderation success.
Alternative Scenario(s)	• Admin cannot edit the review text; only hide or delete (policy).
Postcondition	Review is hidden or removed; provider aggregate recalculated; moderation audit log recorded.

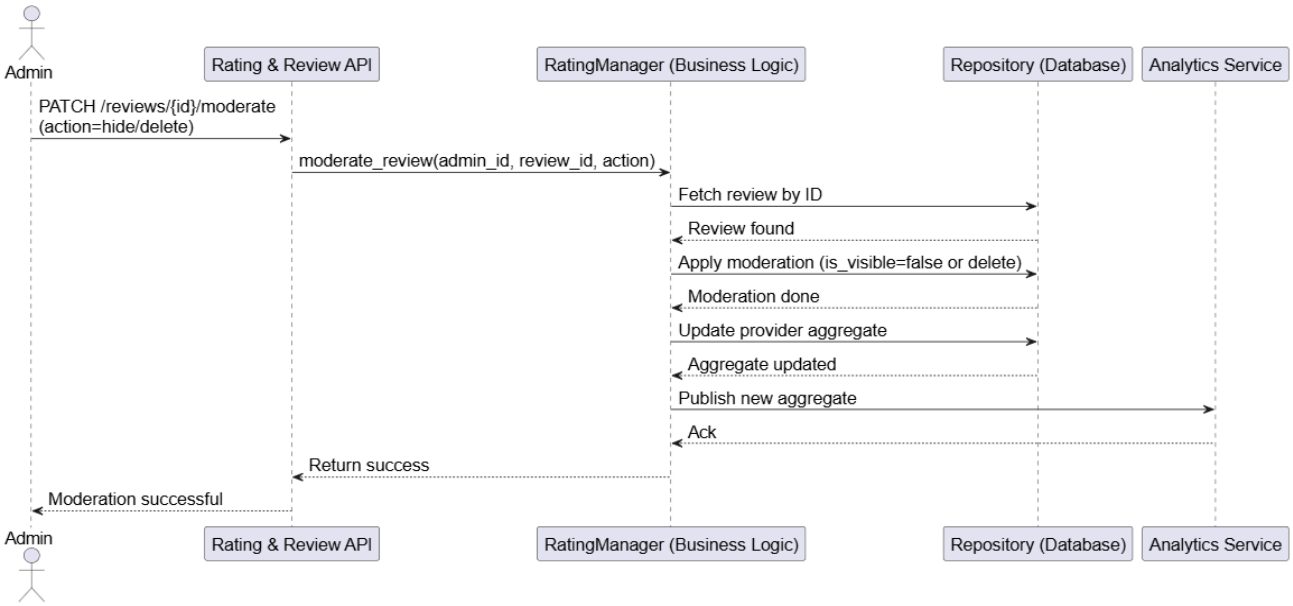


Figure 35 admin moderation sequence

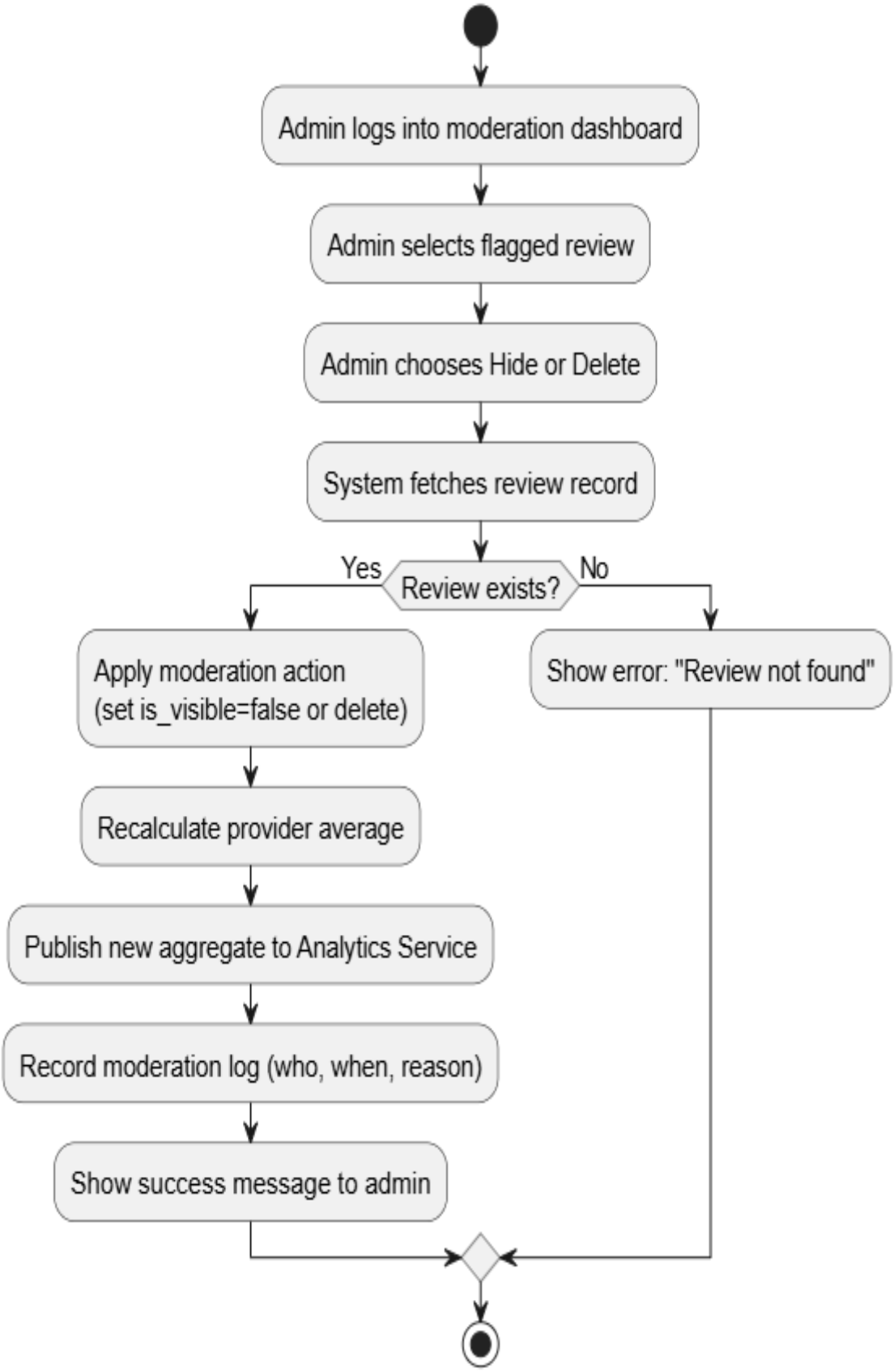


Figure 36 admin moderation activity

- **Provider Notification**

Table 19 provider notification

Use Case Name	**Provider Notification**
Actor	Notification Service
Precondition	A new rating/review is accepted and stored. Notification Service is running.
Main Scenario	1. After saving review, RatingManager composes notification payload (provider_id, rating, text). 2. RatingManager calls Notification Service client. 3. Notification Service sends the notification to provider. 4. Notification Service returns delivery status. 5. Notification Service logs notification outcome and returns success to caller.
Alternative Scenario(s)	• If Notification Service returns failure → retry policy: exponential backoff or enqueue to retry queue; record failure for admin.
Postcondition	Provider receives notification; system recorded notification event.

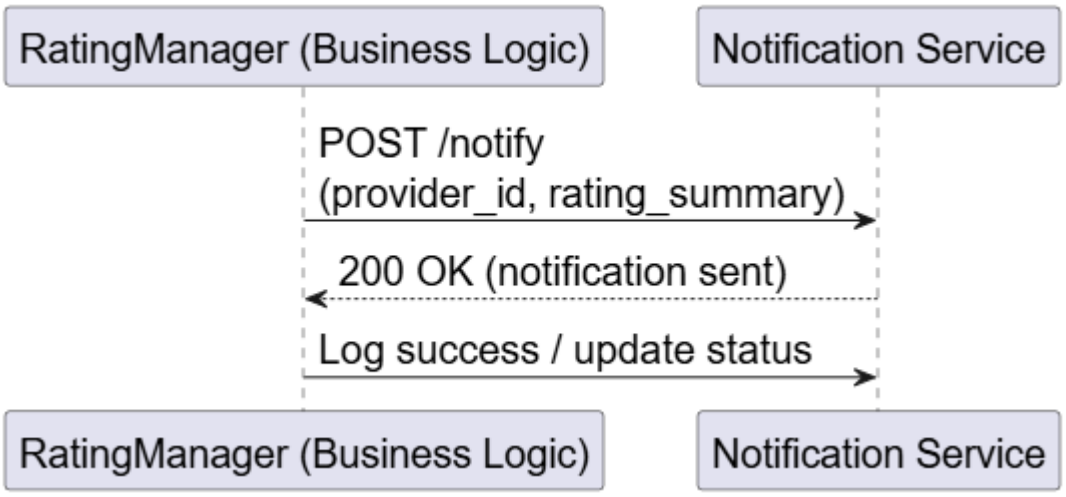


Figure 37 provider notification sequence

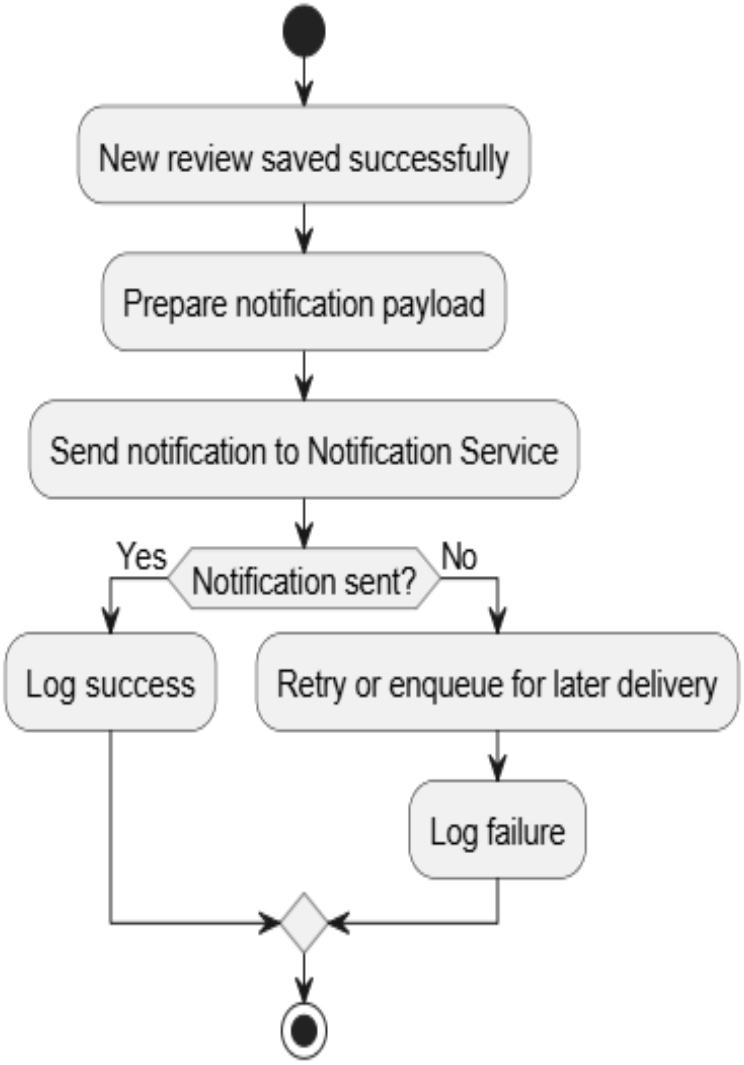


Figure 38 provider notification sequence

- **Provider Average Rating Update**

Table 20 provider average rating update

Use Case Name	**Provider Average Rating Update**
Actor	
Precondition	A rating is created, edited, or deleted. provider_aggregates table exists to store averages and counts.
Main Scenario	1. After review change, RatingManager calls Repository to update aggregate. 2. Repository recalculates (avg, count), ideally via incremental update or SQL aggregate query. 3. Repository persists updated provider_aggregates record. 4. RatingManager returns confirmation.
Alternative Scenario(s)	• Use incremental update: store sum and count and compute avg = sum/count to avoid full-table scans. • If heavy load, use event-driven architecture: write review, emit event, have a background worker recompute aggregates and publish.
Postcondition	Provider aggregate (average rating, review count, rating distribution) is updated and available for read APIs and analytics.

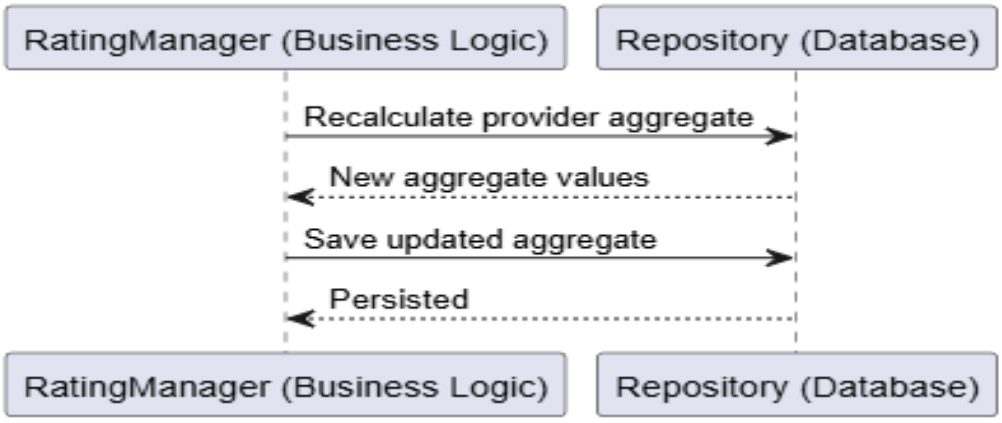


Figure 39 provider average rating sequence

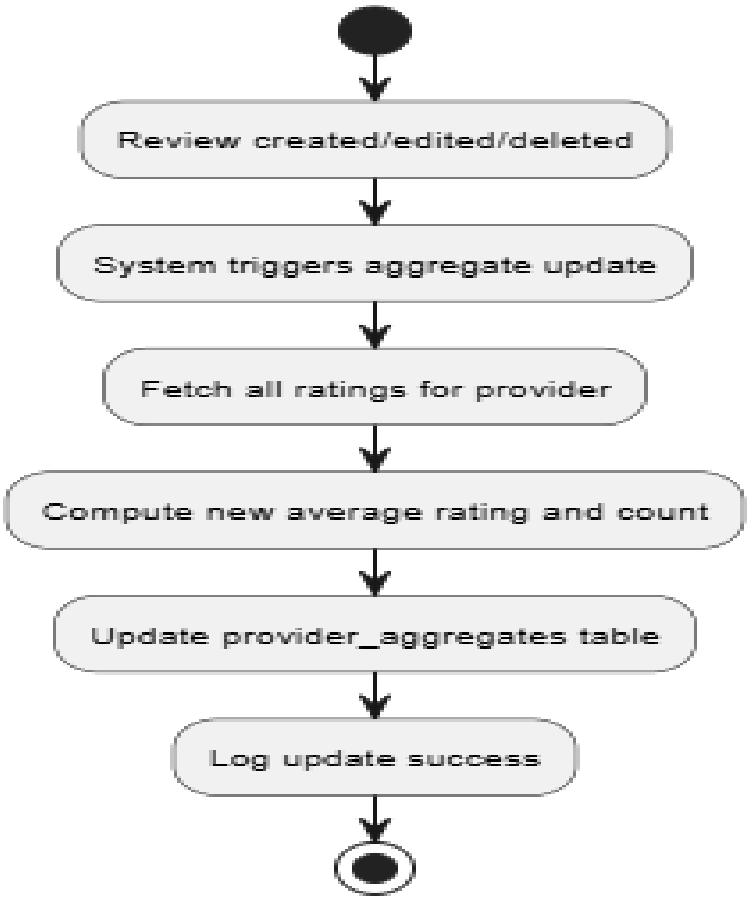


Figure 40 provider average rating activity

3.3 Analysis Class Diagram

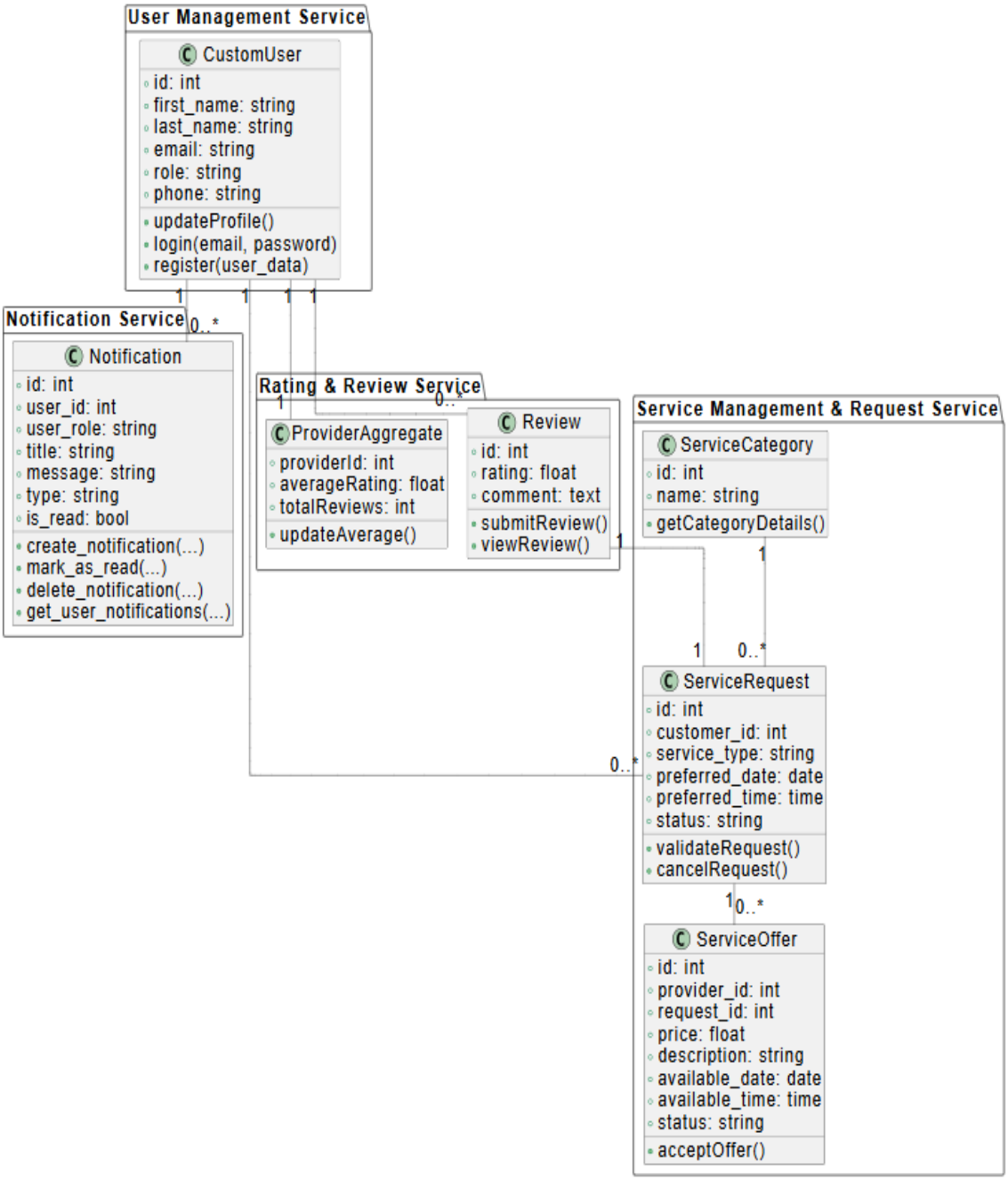


Figure 41 Class Diagram

5. Initial Test Cases

Table 21 service management test cases

Test Case ID	Test Case Title	Test Steps	Expected Result
TC-SM-001	Admin creates a new service category	1. Login as Administrator 2. Navigate to "Service Categories" 3. Click "Add Category" +4. Enter valid name 5. Click "Save"	New category is created and appears in the list.
TC-SM-002	Admin updates an existing service category	1. Login as Administrator 2. Go to "Service Categories" 3. Select a category 4. Click "Edit" 5. Modify name 6. Click "Save"	Category name is updated and reflected in the list.
TC-SM-003	Admin deletes a service category	1. Login as Administrator 2. Go to "Service Categories" 3. Select a category 4. Click "Delete" 5. Confirm deletion	Category is removed from the list and no longer selectable.
TC-SM-004	Customer browses available services	1. Login as Customer 2. Navigate to "Browse Services" page	List of service categories and available services is displayed to the customer.

Table 22 service request test cases

Test Case ID	Test Case Title	Test Steps	Expected Result
TC-SR-001	Customer submits a new service request	1. Login as Customer 2. Navigate to "Request Service" 3. Select service category 4. Enter issue details 5. Upload image (optional) 6. Submit	Service request is created and visible in the customer's request history.
TC-SR-002	Customer uploads invalid image format	1. Login as Customer 2. Navigate to "Request Service" 3. Attempt to upload unsupported file format (e.g., .exe)	Error message is displayed; file is rejected.
TC-SR-003	Request without required issue description	1. Login as Customer 2. Leave issue description blank 3. Attempt to submit	Form validation prevents submission and highlights required field.
TC-SR-004	Service provider views service requests by type	1. Login as Service Provider 2. Go to "Incoming Requests" 3. Select service category from filter	List of matching service requests is displayed.
TC-SR-005	Service provider filters unoffered requests only	1. Login as Service Provider 2. Enable "Unoffered Only" filter	Only requests that have received no offers are displayed.
TC-SR-006	Customer schedules service during restricted hours	1. Login as Customer 2. Attempt to select time slot outside allowed scheduling window	Error or warning message is displayed, and scheduling is blocked.
TC-SR-007	Customer creates service request via AI Assistant	1. Login as Customer 2. Navigate to "AI Service Request" 3. Upload issue images 4. Trigger AI analysis	AI-generated draft service request is created and displayed to the customer.
TC-SR-008	AI Assistant generates structured service description	1. Login as Customer 2. Upload valid issue images 3. Start AI analysis	System generates a structured service description, category, and suggestions.
TC-SR-009	Customer edits AI-generated service request draft	1. Login as Customer 2. Open AI-generated draft request 3. Modify description,	Draft request is updated successfully with customer modifications.

		budget, or schedule 4. Save changes	
TC-SR-010	Customer accepts AI-generated service request draft	1. Login as Customer 2. Open draft request 3. Click “Accept”	Service request is finalized and becomes visible to service providers.
TC-SR-011	Draft request is not visible to service providers	1. Login as Service Provider 2. View incoming requests	Draft service requests are not displayed to service providers.
TC-SR-012	Customer creates service request via Chatbot	1. Login as Customer 2. Open chatbot interface 3. Describe issue in natural language 4. Provide requested details and images	Draft service request is created via chatbot interaction.
TC-SR-013	Chatbot requests clarification for incomplete input	1. Login as Customer 2. Open chatbot 3. Provide vague or incomplete issue description	Chatbot asks follow-up questions to gather required information.
TC-SR-014	Customer accepts chatbot-generated service request	1. Login as Customer 2. Review chatbot-generated draft 3. Click “Accept”	Service request is finalized and service providers are notified.
TC-SR-015	AI analysis failure during request creation	1. Login as Customer 2. Upload corrupted or unclear images 3. Trigger AI analysis	System displays an error message and prompts user to retry or upload new images.

Table 23 offer management test cases

Test Case ID	Test Case Title	Test Steps	Expected Result
TC-OM-001	Service provider creates an offer on a request	1. Login as Service Provider 2. View a new request from the list 3. Click “Create Offer” 4. Enter offer details (price, availability, notes) 5. Submit the offer	Offer is successfully saved and associated with the selected request.
TC-OM-002	Service provider views own offer	1. Login as Service Provider 2. Go to “My Offers”	Offer details are displayed correctly.

		3. Select an offer from the list	
TC-OM-003	Customer views received offers	1. Login as Customer 2. Navigate to “My Requests” 3. Select a request with offers 4. View list of received offers	All relevant offers related to the request are listed.
TC-OM-004	Offer auto-expires after 7 days	1. Submit an offer 2. Wait 7 days without customer action	Offer status is automatically marked as expired.
TC-OM-005	Customer accepts an offer	1. Login as Customer 2. Go to “My Requests” 3. Open the request with pending offers 4. Click “Accept” on a specific offer 5. Confirm acceptance	Offer is accepted, status is updated, and service provider is notified.

Table 24 notification service test cases

Test Case ID	Test Case Title	Test Steps	Expected Result
TC-NT-001	Notify provider on new service request	1. Customer submits a new service request 2. Provider is subscribed to the relevant service category	Provider receives a notification about the new request.
TC-NT-002	Notify customer when they receive a new offer	1. Service provider submits an offer on a customer’s request 2. Notification service is active	Customer receives a notification about the new offer.
TC-NT-003	Notify provider when customer accepts an offer	1. Customer accepts a service provider’s offer 2. Notification logic is triggered	Service provider receives a notification that their offer was accepted.
TC-NT-004	Notify provider when offer is rejected	1. Customer rejects a service provider’s offer 2. System handles update state	Provider receives a notification that their offer was rejected.

Table 25 user management service test cases

Test Case ID	Test Case Title	Test Steps	Expected Result
TC-NT-001	Secure password hashing	Register or login with a known password. Verify password is stored hashed in the database. Login should succeed with correct password only.	Password is securely hashed and validated on login.
TC-NT-002	Role-based access control	Login as each user role (admin, provider, customer). Attempt access to restricted areas for each role.	Only allowed pages are accessible per role; access denied otherwise.
TC-NT-003	Secure logout	Login as any user. Click logout. Try accessing dashboard again.	User is logged out and denied access to protected areas.
TC-NT-004	Auto logout after inactivity	Login and remain inactive for session timeout period. Attempt any action post timeout.	Session ends automatically and user is redirected to login.
TC-NT-005	Customer registration	Fill registration form with valid customer data. Submit the form. Try to register with duplicate email.	User is registered once; duplicates are rejected.
TC-NT-006	Service provider registration	Fill registration form with valid customer data. Submit the form. Try to register with duplicate email.	User is registered once; duplicates are rejected.
TC-NT-007	Update account information	Perform actions related to the requirement. Observe system response.	Requirement is fulfilled as expected.
TC-NT-008	Delete account	Perform actions related to the requirement. Observe system response.	Requirement is fulfilled as expected.
TC-NT-	Change password	Perform actions related to the requirement.	Requirement is fulfilled as expected.

009		Observe system response.	
TC-NT-0010	View profile	Perform actions related to the requirement. Observe system response.	Requirement is fulfilled as expected.
TC-NT-0011	View service providers	Perform actions related to the requirement. Observe system response.	Requirement is fulfilled as expected.
TC-NT-0012	View customers	Perform actions related to the requirement. Observe system response.	Requirement is fulfilled as expected.
TC-NT-0013	Activate/Deactivate users	Perform actions related to the requirement. Observe system response.	Requirement is fulfilled as expected.
TC-NT-0014	Pagination support	Perform actions related to the requirement. Observe system response.	Requirement is fulfilled as expected.
TC-NT-0015	Update user name and contact info	Perform actions related to the requirement. Observe system response.	Requirement is fulfilled as expected.
TC-NT-0016	Update user address/location	Perform actions related to the requirement. Observe system response.	Requirement is fulfilled as expected.
TC-NT-0017	Update password	Perform actions related to the requirement. Observe system response.	Requirement is fulfilled as expected.
TC-NT-0018	Delete account and logout	Login as any user. Click logout. Try accessing dashboard again.	User is logged out and denied access to protected areas.

Table 26 rating & review service test cases

Test Case ID	Test Case Title	Test Steps	Expected Result
TC-RR-001	Customer submits rating and review for completed service	1. Login as Customer 2. Open completed service request 3. Click “Rate & Review” 4. Select rating (1–5) 5. Enter optional review text 6. Submit	Rating and review are saved successfully; confirmation message is displayed.
TC-RR-002	Rating submission for incomplete service	1. Login as Customer 2. Open a service request with status ≠ completed 3. Attempt to submit rating	System rejects submission and displays an error message.
TC-RR-003	Rating value out of allowed range	1. Login as Customer 2. Open completed request 3. Enter rating outside range (e.g., 0 or 6) 4. Submit	Validation error is displayed; submission is blocked.
TC-RR-004	Review text exceeds maximum length	1. Login as Customer 2. Open completed request 3. Enter review text exceeding allowed length 4. Submit	Validation error is displayed and review is not saved.
TC-RR-005	Customer submits duplicate review	1. Login as Customer 2. Submit rating for completed request 3. Attempt to submit another review for same request	System prevents duplicate submission and displays appropriate message.
TC-RR-006	Provider views own ratings and reviews	1. Login as Service Provider 2. Navigate to provider profile 3. View ratings and reviews	Provider average rating, review count, and paginated reviews are displayed.
TC-RR-007	Provider with no reviews views ratings page	1. Login as Service Provider 2. Navigate to profile ratings section	System displays “No reviews yet” message.

TC-RR-008	Customer edits own review	1. Login as Customer 2. Navigate to previously submitted review 3. Click “Edit” 4. Modify rating or text 5. Submit	Review is updated successfully and timestamps are refreshed.
TC-RR-009	Customer deletes own review	1. Login as Customer 2. Navigate to own review 3. Click “Delete” 4. Confirm deletion	Review is deleted or hidden according to retention policy.
TC-RR-010	Customer edits review after allowed time window	1. Login as Customer 2. Attempt to edit review after allowed time period	System rejects edit and displays policy-related message.
TC-RR-011	Admin hides inappropriate review	1. Login as Admin 2. Open moderation panel 3. Select flagged review 4. Choose “Hide”	Review visibility is set to hidden and removed from public view.
TC-RR-012	Admin deletes review	1. Login as Admin 2. Open moderation panel 3. Select review 4. Choose “Delete”	Review is deleted or archived; action is logged.
TC-RR-013	Admin moderation audit logging	1. Login as Admin 2. Hide or delete a review	Moderation action is logged with admin ID, timestamp, and reason.
TC-RR-014	Provider receives notification for new review	1. Customer submits rating and review 2. Notification Service processes event	Provider receives notification successfully.
TC-RR-015	Notification service failure handling	1. Submit rating while Notification Service is unavailable	Review is saved; notification is queued for retry.
TC-RR-016	Provider average rating calculation	1. Submit new review 2. Trigger aggregate update	Provider average rating and review count are updated correctly.
TC-RR-017	Provider aggregate update after review edit	1. Edit existing review rating 2. Submit changes	Provider aggregate is recalculated and updated.
TC-RR-018	Provider aggregate update after review deletion	1. Delete an existing review	Provider average and review count are recalculated correctly.

6. Initial Requirement Traceability Matrix (RTM)

Table 27 Initial Requirement Traceability Matrix (RTM)

Req_ID	Requirement Title	SRS Section	System Design (Module)	Analysis (Use Case)	Test Case(s)	Coding
REQ-1.1	Login through interfaces	3.1		Table 5 log in	TC-NT-001	
REQ-1.2	Secure password hashing	3.1		Table 5 log in	TC-NT-001	
REQ-1.3	Role-based access control	3.1		Table 5 log in	TC-NT-002	
REQ-1.4	Secure logout	3.1		Table 5 log in	TC-NT-003	
REQ-1.5	Auto logout (Inactivity)	3.1		Table 5 log in	TC-NT-004	
REQ-2.1	Customer registration	3.1		register	TC-NT-005	
REQ-2.2	Provider registration	3.1		register	TC-NT-006	
REQ-3.1	Update account info	3.1		Manage Profile	TC-NT-007, TC-NT-015	
REQ-3.2	Delete account	3.1		Manage Profile	TC-NT-008, TC-NT-018	
REQ-3.3	Change password	3.1		Manage Profile	TC-NT-009, TC-NT-017	
REQ-3.4	View profile	3.1		Manage Profile	TC-NT-010	
REQ-4.1	Admin: View providers	3.1		User Mgmt.	TC-NT-011	
REQ-4.2	Admin: View customers	3.1		User Mgmt.	TC-NT-012	
REQ-4.3	Activate/Deactivate users	3.1		User Mgmt.	TC-NT-013	
REQ-5.1	Create service categories	3.1		Manage Category	TC-SM-001	
REQ-5.2	Update service categories	3.1		Manage Category	TC-SM-002	
REQ-5.3	Delete service categories	3.1		Manage Category	TC-SM-003	
REQ-5.4	Browse services	3.1		Browse Services	TC-SM-004	
REQ-6.1	Submit service requests	3.1		Request Service	TC-SR-001	

REQ-6.2	Scheduling logic	3.1		Schedule Service	TC-SR-006	
REQ-6.3	View requests by type	3.1		View Requests	TC-SR-004	
REQ-6.4	Filter unoffered requests	3.1		Filter Requests	TC-SR-005	
REQ-7.1	VLM Image Analysis	3.2.1		create service request via ai-assistant	TC-SR-007, TC-SR-008	
REQ-7.2	Generate AI Draft	3.2.1		create service request via ai-assistant	TC-SR-007	
REQ-7.3	Edit AI-generated draft	3.2.1		Edit Draft	TC-SR-009	
REQ-7.4	Customer confirmation	3.2.1		Finalize Request	TC-SR-010	
REQ-7.6	Stateful Workflow	3.2.2		Process Request	TC-SR-011	
REQ-8.1	Create service offers	3.1.12		Create Offer	TC-OM-001	
REQ-8.2	Provider: View own offers	3.1.12		Manage Offers	TC-OM-002	
REQ-8.3	Customer: View offers	3.1.12		View Offers	TC-OM-003	
REQ-8.4	Auto-expire offers (7 days)	3.1.12		System Task	TC-OM-004	
REQ-8.5	Accept an offer	3.1.12		Accept Offer	TC-OM-005	
REQ-9.1	Notify: New Request	3.1.13		Notify User	TC-NT-001	
REQ-9.2	Notify: New Offer	3.1.13		Notify User	TC-NT-002	
REQ-9.3	Notify: Offer Accepted	3.1.13		Notify User	TC-NT-003	
REQ-9.4	Notify: Offer Rejected	3.1.13		Notify User	TC-NT-004	
REQ-10.1	AI Cost Estimation	3.2.3		Estimate Cost	TC-SR-008, TC-SR-009	
REQ-11.1	Rate service providers	3.1.14		Rate & Review	TC-RR-001	
REQ-11.2	Submit textual reviews	3.1.14		Rate & Review	TC-RR-001	
REQ-11.3	Calculate Trust Score	3.1.14		Update Trust	TC-RR-016, TC-RR-018	

REQ-11.5	Restricted range (1-5)	3.1.14		Rate & Review	TC-RR-003	
REQ-11.7	Prevent duplicate reviews	3.1.14		Rate & Review	TC-RR-005	
REQ-11.10	Admin: Hide/Show reviews	3.1.14		Moderate Reviews	TC-RR-011	
REQ-11.14	Sentiment Score Logic	3.1.14		Analyze Sentiment	TC-RR-013	
REQ-12.1	Chatbot Interface	3.2.4		Use Chatbot	TC-SR-012	
REQ-12.2	Chatbot guided creation	3.2.4		Create via Chat	TC-SR-012, TC-SR-014	
REQ-12.3	Chatbot Inquiries	3.2.4		Inquiry	TC-SR-013	
REQ-13.1	Pagination Support	3.1.15		UI List	TC-NT-014	

Chapter 5 System Design

1. Introduction

This chapter presents a comprehensive overview of the **system design phase**, which translates the requirements defined in previous chapters into a structured and implementable technical solution. The design phase focuses on defining the **overall system architecture**, identifying system components, and modeling the interactions between them to ensure scalability, maintainability, and reliability.

The chapter covers both **high-level architectural decisions** and **detailed component design**, emphasizing how modern software engineering practices—such as microservices, asynchronous processing, and AI-driven automation—were applied. Special attention is given to the integration of **artificial intelligence services**, including image-based request analysis, cost estimation, trust scoring, and conversational support via a chatbot.

By adopting a modular and extensible design, the system enables continuous enhancement, supports independent deployment of services, and ensures that new features can be integrated with minimal impact on existing components.

1. System Architecture

The Home Care System is designed using a **microservices-based architecture**, where each core functionality is implemented as an independent service with a clearly defined responsibility. This architectural approach was chosen to improve **scalability, fault isolation, maintainability, and development flexibility**.

Each microservice operates independently, maintains its own database where applicable, and communicates with other services through **RESTful APIs** using standard data formats (JSON). This ensures loose coupling between services and allows individual components to be scaled, updated, or redeployed without affecting the entire system.

The architecture also incorporates:

- **Asynchronous communication** using background workers (Celery) for time-consuming tasks such as AI inference, sentiment analysis, and notification delivery.

- **Caching and indexing mechanisms** to improve system responsiveness and reduce database load.
- **Workflow orchestration** to manage complex AI-driven processes, such as automated service request creation and validation.

At a high level, the system consists of:

- Client applications (Web Interface)
- Core business microservices
- AI and automation services
- Supporting infrastructure services (notifications, authentication, analytics)

This layered and distributed design ensures that the system meets both **functional requirements** (e.g., service requests, offers, ratings) and **non-functional requirements** (e.g., performance, scalability, availability).

2. Microservices Breakdown

The Home Care System is composed of multiple specialized microservices, each responsible for a distinct domain of functionality. The following sections describe the main microservices and their responsibilities.

1. User Mgmt. Service

1. Service Overview

The **User Management Service** serves as the centralized authority for identity and access management within the system. It is responsible for handling all user-related operations, ensuring secure entry points for authentication, and enforcing authorization policies across the microservices ecosystem.

2. Architectural Design

The service is built upon a **Service-Repository Pattern** within a Django framework. This architectural choice promotes a strict separation of concerns, ensuring that business logic is decoupled from data access and presentation layers.

- **Presentation Layer (Views & Serializers):** Handles HTTP requests and data validation. It separates distinct logical flows for standard Users, Providers, and Technical Profiles.
- **Business Logic Layer (Services):** Encapsulates core business rules (e.g., user_services.py, tech_profile_services.py). This layer uses an **Event Emitter** utility to handle asynchronous tasks and side effects, ensuring the core service remains responsive.
- **Data Access Layer (Repositories):** Abstracts the database interactions. Modules such as user_repositories.py and tech_profile_repositories.py handle direct ORM queries, providing a clean API for the service layer.

3. Component Diagram

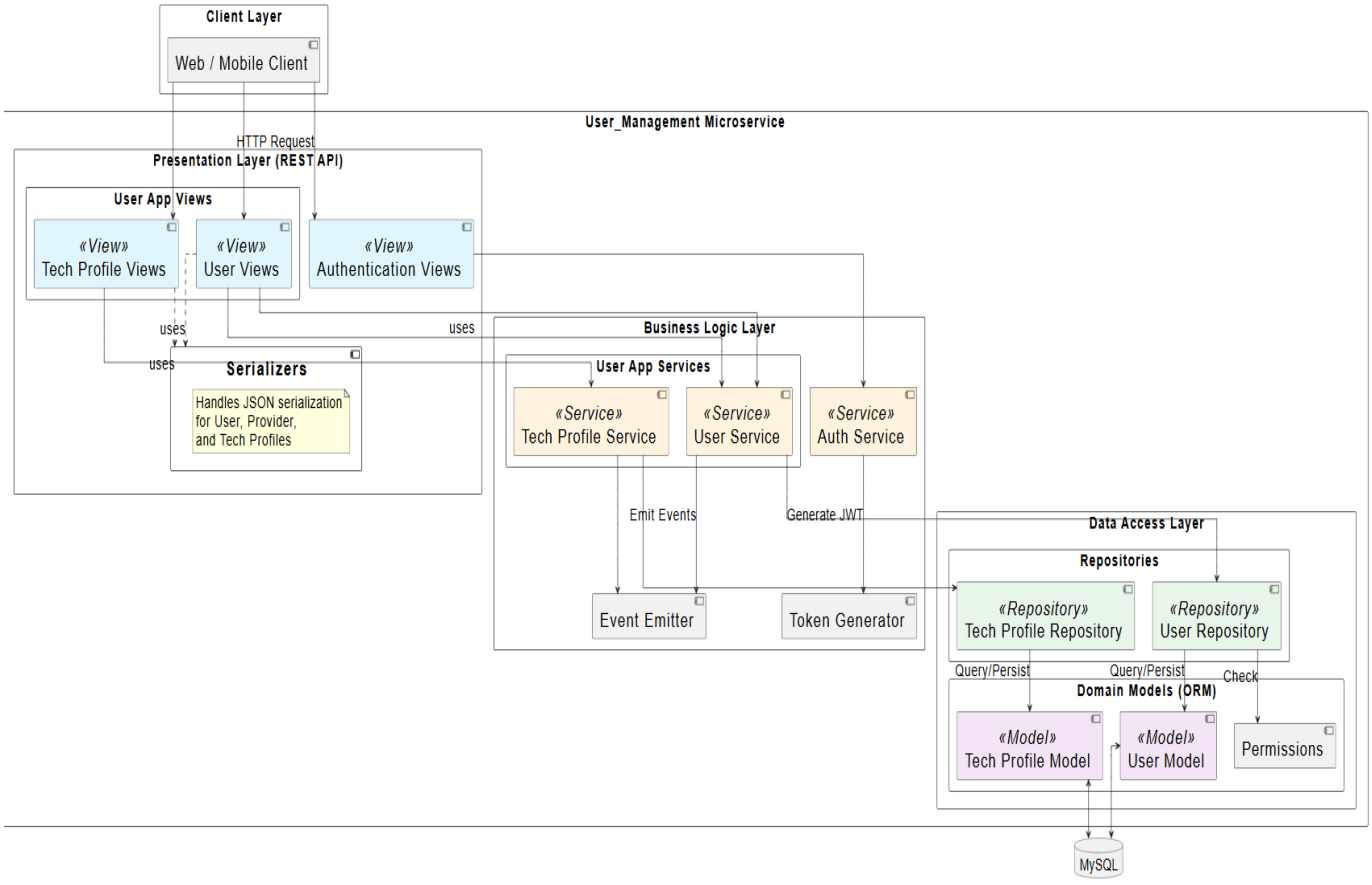


Figure 42 user management architecture

4. Key Responsibilities & Capabilities

Based on the implementation, the service delivers the following core functionalities:

- **Multi-Role User Registration:** Specialized flows for Customers, Service Providers, and Administrators, supported by dedicated serializers for data validation.
- **Authentication & Session Management:** Secure login/logout functionality utilizing **JWT (JSON Web Token)** generation for stateless authentication.
- **Role-Based Access Control (RBAC):** Granular permission enforcement handled at the repository and view levels to restrict resource access based on user roles.
- **Profile Management:**
    - **User Profiles:** standard account details.
    - **Technical Profiles:** specialized data structures for service providers (e.g., years in business, technical qualifications).
- **Event-Driven Architecture:** Utilization of an internal **Event Emitter** to trigger system-wide actions (e.g., notifications, logging) without blocking the main request thread.
- **Account Administration:** centralized control for administrators to activate, deactivate, or audit user accounts.

2. Service Management & Request

1. Service Overview

The **Service Management and Request Service** acts as the operational backbone of the platform. It is responsible for the end-to-end lifecycle of service requests, managing the catalog of service categories, and facilitating the negotiation process (offers and acceptances) between customers and providers. Unlike a traditional CRUD service, this microservice integrates an **AI-driven Automation Engine** to assist users in defining complex requests through conversational interfaces and visual analysis.

2. Architectural Design

The service implements a **Layered Architecture** enhanced with **Event-Driven** and **Workflow-based** patterns:

- **Presentation Layer:** Split between standard REST **Views** (for manual CRUD operations) and **Consumers** (likely for async updates).
- **AI Automation Layer (`/automation` & `/chatbot`):** This is a unique architectural component. It uses a **Node-based Workflow Engine** (likely a graph or state machine) to process complex requests. Specific nodes handle distinct tasks:
  - **VLM Node:** Uses Vision-Language Models to analyze uploaded images (e.g., identifying "pipe damage").
  - **Budget & Time Nodes:** Automate estimation and scheduling constraints.
  - **Orchestrator & Intent Classifier:** Manages the conversational flow and determines user intent using LLMs and Redis for state management.
- **Business Logic Layer (`/services`):** Encapsulates core domain logic (e.g., `request_service.py`, `inference_engine.py`) and utilizes specialized tools like the **Price Engine** and **Scheduler Engine**.
- **Data Access Layer (`/repositories`):** Abstracts database interactions using the **Repository Pattern** (e.g., `ServiceRequestRepository`), decoupling business logic from the specific ORM implementation.

3. Component Diagram

The following diagram illustrates the architecture, highlighting the separation between the Standard Request Flow and the AI Automation Workflow.

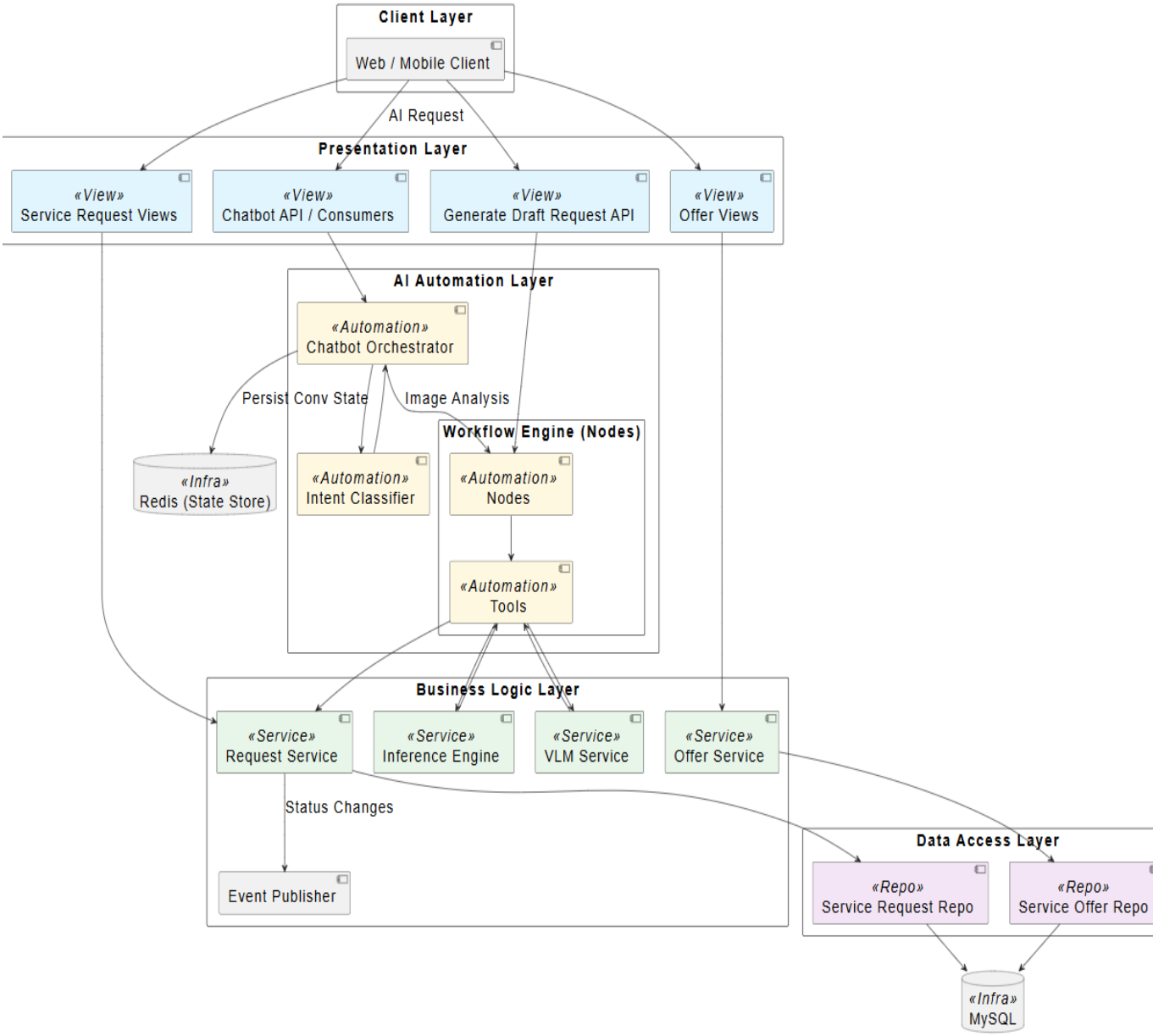


Figure 43 service request architecture

4. Detailed AI Automation and Request Workflow

The core innovation of this service lies in its Automation Layer (illustrated in Figure [39]), which utilizes LangGraph for sophisticated task orchestration. As shown in the detailed workflow, the system transitions from an API Gateway through various specialized Workflow Nodes—including VLM Analysis for visual processing and automated Budget/Time estimation—before finalizing the payload and interacting with the Service Layer and external integration tools.

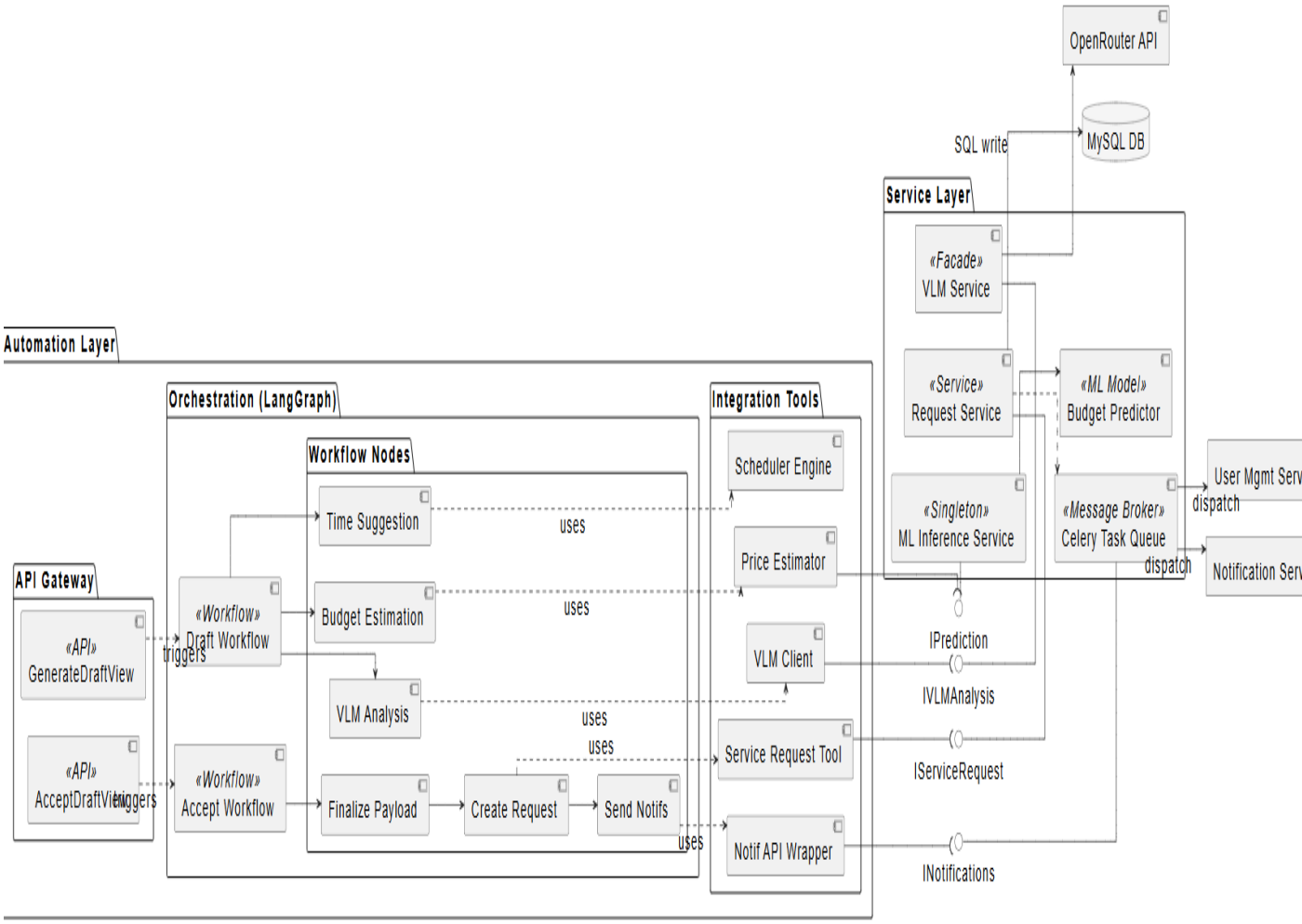


Figure 44 Detailed Architecture of the AI Automation Layer

5. Key Responsibilities & Capabilities

Based on the file structure analysis, the service delivers the following capabilities:

- **Advanced Service Request Management:**
  - **Manual Creation:** Standard REST endpoints for users to post requests.
  - Managing **t**he end-to-end **AI-driven service request workflow**
  - Creating draft service requests based on **AI analysis**
  - Maintaining a stateful workflow **(Draft → Accepted)**
  - Coordinating:
    - Image analysis
    - Cost estimation
    - Service categorization
    - Scheduling suggestions
  - Logging workflow steps and outcomes for monitoring and debugging
- **Intelligent Image Analysis:**
  - The `VLM Node` and `VLM Service` uses Vision-Language Models to automatically analyze images uploaded by customers (e.g., detecting damage types from photos) to populate request descriptions.
- **Automated Estimation & Categorization:**
  - Hosting a trained **multi-task vision model** (e.g., EfficientNet-based architecture)
  - Analyzing uploaded images to:
    - Identify the appropriate service category
    - Estimate preliminary repair costs
- **Scheduling:**
  - **Scheduler Engine** tools validate time constraints before a request is finalized.
- **Offer Lifecycle Management:**
  - Full support for providers to create, view, and expire offers, managed via the `ServiceOfferRepository`.
- **State & Workflow Management:**
  - Uses **Redis** to maintain chatbot conversation state.
  - Manages distinct request states (Draft, Accepted, In Progress, Completed) via strictly defined state transitions.

- **Conversational Support (Chatbot)**

  The **Chatbot Service** provides a natural language interface for interacting with the system.

  Key Responsibilities:

  - Allowing customers to create service requests using conversational input
  - Guiding users through the request creation and approval process
  - Asking follow-up questions when user input is incomplete or ambiguous
  - Integrating with the automation and AI inference services
  - Providing instant responses to common questions

  The chatbot improves accessibility and user experience by reducing reliance on traditional forms.

- **Event Propagation:**
  - The `Event Publisher` broadcasts lifecycle events (e.g., `REQUEST_CREATED`, `OFFER_ACCEPTED`) to other microservices (like Notifications or User Management) asynchronously.

3. Notification Service

1. Service Overview

The **Notification Service** is the communication hub of the platform. It is responsible for the orchestration and delivery of real-time, event-driven alerts to users across multiple channels. By decoupling notification logic from core business services, it ensures that system events (like new offers or service updates) are communicated instantly without impacting the performance of the originating microservice.

2. Architectural Design

The service is designed for high concurrency and low latency, utilizing an **Asynchronous Event-Driven Architecture**:

- **Presentation Layer (REST & WebSockets):**
    - **REST API:** Provides endpoints for users to retrieve notification history, filter through past alerts, and update notification states (e.g., marking as read).
    - **WebSocket Layer (ASGI):** Facilitates real-time "push" capabilities, allowing the system to transmit alerts to active clients instantly as events occur.
- **Asynchronous Task Layer:** Given the "asynchronous delivery" requirement, the service typically utilizes a **Task Queue (e.g., Celery)** and a **Message Broker (e.g., Redis)**. This ensures that even if an external delivery channel (like email or a mobile push provider) is slow, the platform remains responsive.
- **Service & Business Logic:** Centralizes the logic for "Notification Dispatching." This layer determines which users should receive an alert based on the event type and handles the formatting of notification templates.
- **Data Access Layer:** Manages the persistence of notification states, ensuring that "Read/Unread" statuses are synchronized across all user devices.

3. Component Diagram

The following diagram illustrates the flow from a system event to real-time delivery and persistent storage.

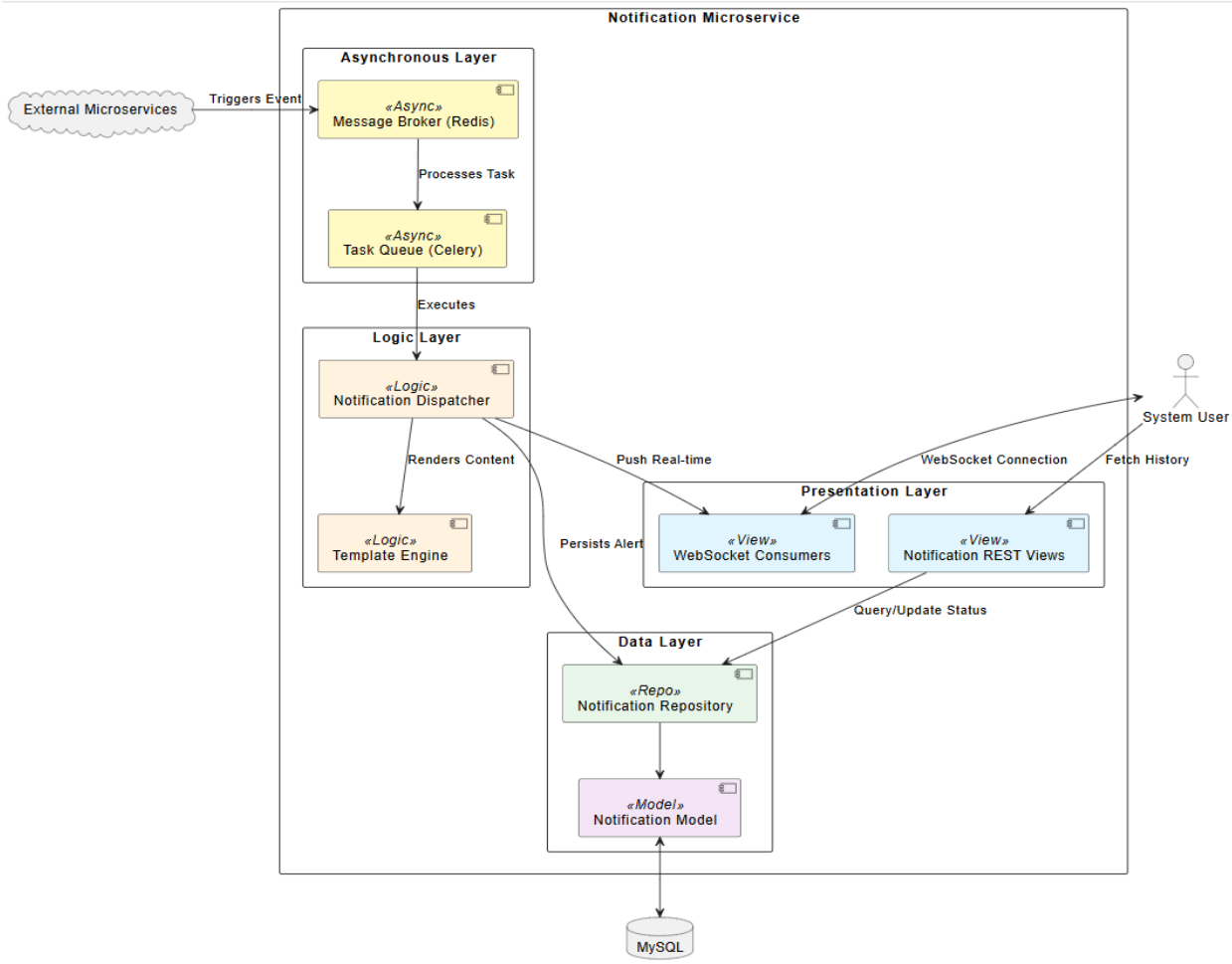


Figure 45 Architectural Diagram of the Notification Service

4. Key Responsibilities & Capabilities

- **Event-Driven Alert Generation:** Automatically generates notifications triggered by lifecycle events from other services, including:
    - **Marketplace Events:** New service requests and offer submissions.
    - **Negotiation Events:** Real-time updates on offer acceptance or rejection.
    - **Feedback Events:** Alerts when a rating or review is submitted for a completed service.

- **Real-Time Delivery (WebSockets):** Uses an ASGI-based framework to push notifications to the user's browser or app the moment they are generated, minimizing communication delay.
- **Intelligent History Management:** Supports advanced retrieval features including **filtering** (by category or importance) and **pagination** to ensure the user interface remains performant even with thousands of alerts.
- **State Tracking:** Provides robust mechanisms for users to manage their inbox, including bulk "mark as read" operations and persistent unread counts.
- **Asynchronous Reliability:** By utilizing a decoupled task layer, the service ensures that notification delivery is "fire-and-forget" for the rest of the system, preventing bottlenecks during peak activity.

4. Rating, Review, and Trust Scoring Service

1. Service Overview

The **Rating, Review, and Trust Scoring Service** acts as the platform's reputation and quality assurance engine. It manages the complete lifecycle of user feedback, from submission and automated sentiment processing to moderation and aggregation. By transforming qualitative textual reviews into quantitative data, this service provides the foundational metrics required for the platform's trust evaluation mechanisms and provider ranking systems.

2. Architectural Design

The service is structured to ensure data integrity and decoupling of concerns, utilizing a **Service-Repository** architecture:

- **Presentation Layer (Views & Serializers):** Handles incoming review submissions and status queries. It includes dedicated **Moderation Views** for administrative control and **Provider Stats Views** for retrieving aggregated performance metrics.
- **Business Logic Layer (Services):** The core logic is encapsulated in modules like `review_service.py`. This layer enforces complex business rules, such as the "one review per service request" constraint, and orchestrates the **Sentiment Analysis** workflow.

- **AI & Integration Layer:** Dedicated integration modules handle calls to sentiment analysis engines. This layer processes raw review text to extract emotional polarity (Positive/Neutral/Negative), which is then stored as structured metadata.
- **Data Access Layer (Repositories):** Uses the **Repository Pattern** (e.g., `ReviewRepository`, `ProviderStatsRepository`) to abstract database operations. This allows the service layer to query review history and update aggregated statistics without being coupled to the underlying ORM.

3. Component Diagram

The following diagram illustrates the internal architecture and its integration with AI and trust-scoring components:

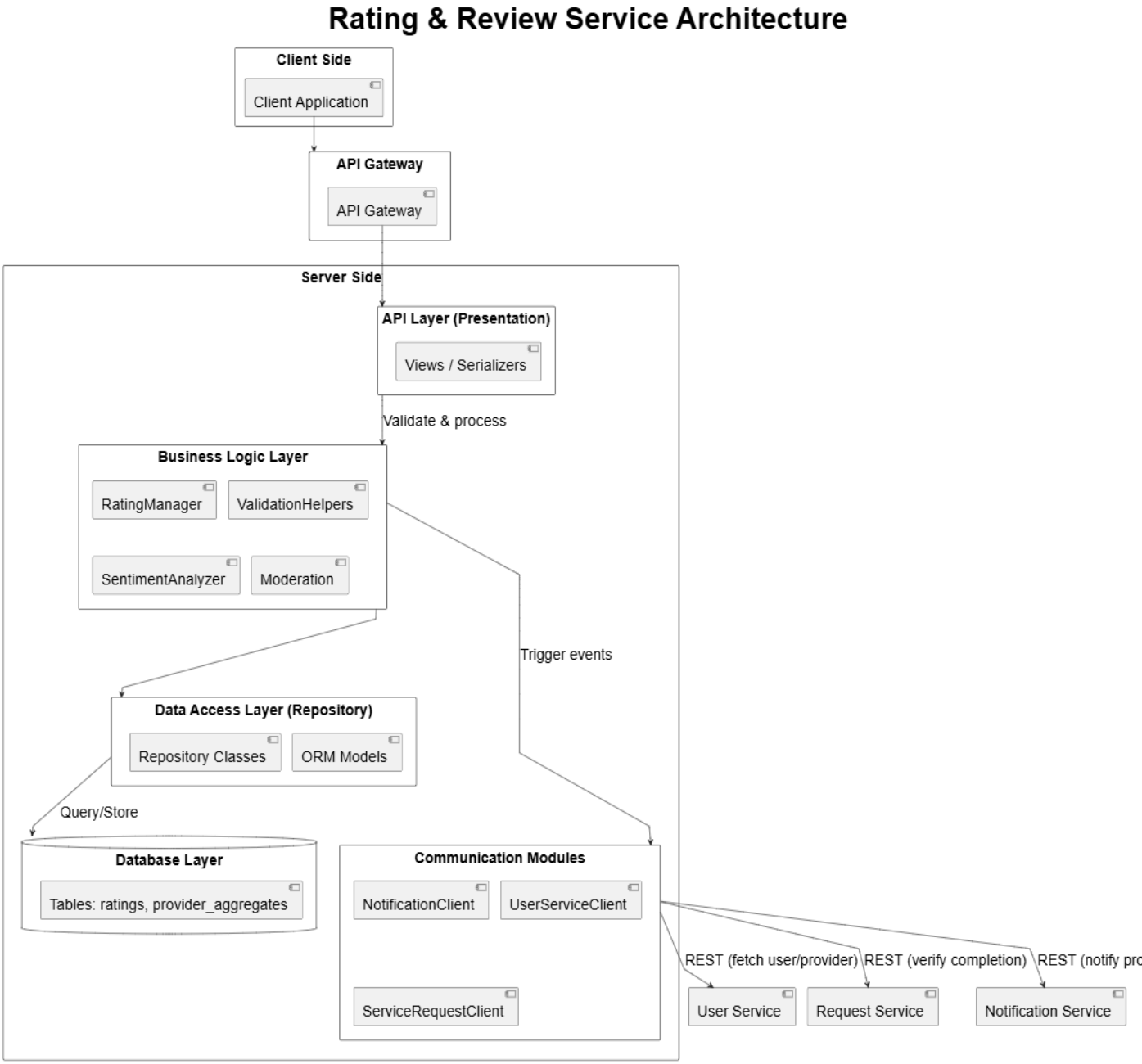


Figure 46 component diagram - rating & review

4. **Key Responsibilities & Capabilities**

- **Integrity-First Review Management:** Implements strict validation at the repository level to enforce a **one-review-per-service-request** policy, preventing feedback manipulation and ensuring authenticity.
- **AI-Driven Sentiment Analysis:** Automatically analyzes review content during the ingestion process to categorize user sentiment. This metadata is used to provide nuanced insights beyond simple numerical ratings.
- **Real-time Provider Statistics:** Maintains a high-performance **Aggregation Engine** that updates provider-level statistics (average rating, sentiment distribution, total reviews) immediately upon review approval.
- **Administrative Moderation Workflow:** Provides a robust set of tools for visibility control, allowing administrators to flag, hide, or delete reviews that violate community standards.
- **Trust Score Data Provisioning:** Functions as the primary data source for the platform's **Trust Evaluation System**. It exposes clean, processed feedback data and sentiment scores required for multi-factor trust computation.
- **Accountability and Transparency:** Ensures that every service interaction is closed with a transparent feedback loop, fostering a marketplace environment based on merit and quality.

3. Design Patterns

1. Repository Design Pattern

To ensure clean code and separation of concerns, the Repository Pattern is used for data access logic. To promote separation of concerns and clean architecture, the Repository Design Pattern is used extensively across services. Business logic is handled in views or service classes, while all database interactions are abstracted within repository classes. This allows for:

- Views handle coordination and validation.
- Repositories handle DB operations.
- This enhances maintainability and testability.
- Better testing and mocking
- Cleaner and more maintainable code
- Reusability of data-access logic

Example:

ServiceOfferRepository handles all queries related to service offers, isolating them from the views.

2. Façade Design Pattern: Simplifying the Interface

The **Facade Pattern** was implemented to provide a unified, simplified interface to the complex set of tools within the business logic layer.

- **Implementation:** The `PriceEstimator` class acts as the Facade. It shields the `BudgetNode` (the Orchestrator) from the complexities of the underlying subsystems.
- **VLM Service Context:** In the Vision-Language Model (VLM) service, the Facade handles the intricate coordination between raw image processing, prompt engineering, and the subsequent logic required to interpret model outputs.
- **Key Benefits:**
    - **Decoupling:** The Automation Workflow Layer doesn't need to know how prices are calculated; it only needs to call a single `calculate()` method.
    - **Reduced Complexity:** It wraps multiple internal steps (like temporary file handling and risk-to-severity mapping) into a clean API.

3. Strategy Pattern: Runtime Flexibility

To handle different pricing requirements—specifically the choice between high-precision Machine Learning and deterministic Rule-Based logic—the **Strategy Pattern** was utilized.

- **Implementation:**
    - **Context:** `PriceEstimator` maintains a reference to a `PricingStrategy`.
    - **Strategies:** Both `MLBasedPricingStrategy` and `RuleBasedPricingStrategy` implement a common `PricingStrategy` interface.

- **Usage:** This allows the system to switch algorithms at runtime based on the `AutomationState`. If the VLM identifies a standard, low-risk item, a rule-based approach is used; for complex, ambiguous cases, the ML-based strategy is triggered.
- **Key Benefits:**
    - **Open/Closed Principle:** New estimation strategies can be added in the future without modifying the existing `PriceEstimator` code.
    - **Clean Logic:** It eliminates messy conditional blocks (`if/else`) by encapsulating behaviors in dedicated classes.

4. Singleton Pattern: Efficient Resource Management

Because Machine Learning models and inference engines (like those used in the VLM Service) are resource-intensive and expensive to load, the **Singleton Pattern** was applied to the inference layer.

- **Implementation:** The `MLInferenceService` is implemented as a thread-safe Singleton using **Double-Checked Locking**.
- **Mechanism:** The `get_instance()` method ensures that only one instance of the `MLInferenceService` (and by extension, the `BudgetPredictor`) exists across the entire Django application.
- **Key Benefits:**
    - **Memory Efficiency:** Prevents the system from loading multiple instances of heavy ML models into RAM, which would otherwise lead to "Out of Memory" (OOM) errors.
    - **Consistency:** Ensures that all parts of the application share the same model state and configuration.
    - **Thread Safety:** The use of a `_lock` ensures that even in a high-concurrency environment, the service is initialized correctly only once.

5. Adapter Pattern: Interface Translation

The **Adapter Pattern** is used to bridge the gap between the VLM Service and the LangGraph automation nodes, ensuring they can communicate through a standardized API.

- **Implementation:** The `VLMClient` (located in **vlm_client.py**) adapts the **VLMService** for use within LangGraph nodes.
- **Why:** It keeps the automation nodes independent of specific service implementations. This makes it trivial to swap VLM providers (e.g., moving from OpenAI to a local model) without refactoring the core workflow logic.
- **Trade-off:** We ensure the adapter remains "thin," focusing strictly on mapping async calls and standardizing outputs, while leaving heavy business logic to the service layer.

6. Observer Pattern: Decoupled Event Handling

The **Observer Pattern** is implemented via **Django Signals** to handle side effects that occur after specific domain events.

- **Implementation:** A **post_save** signal on the **ServiceRequest** model (in **signals.py**) triggers provider notifications.
- **Execution:** The signal dispatches a Celery task (**notify_matching_providers_task**) to ensure notifications are sent asynchronously without slowing down the HTTP response.
- **Why:** This allows the "Service Management" module to remain decoupled from the "Notification" module. The core logic doesn't need to know how or who to notify; it simply broadcasts that a request was created.
- **Note:** We maintain **idempotency** in the side effects to ensure that retried Celery tasks do not result in duplicate notifications.

7. Pattern Summary Table

Table 28 Pattern Summary Table

Pattern	Category	Primary Role	Implementation Highlight
Facade	Structural	Entry point for VLM logic	PriceEstimator
Adapter	Structural	Standardizes VLM for LangGraph	VLMClient
Strategy	Behavioral	Interchangeable pricing logic	MLBasedPricingStrategy
Observer	Behavioral	Decoupled post-save reactions	Django post_save Signals
Singleton	Creational	Thread-safe model management	MLInferenceService
Repository	Creational/Structural	Abstraction of the Data Layer	ServiceRequestRepository

4. Communication Between Components

The system utilizes a hybrid communication model designed to optimize user experience through synchronous responsiveness while ensuring system-wide reliability through asynchronous processing.

1. External Communication: Frontend–Backend (North-South Traffic)

The interface between the client-side application and the server-side logic is governed by **RESTful API principles**, ensuring a stateless and scalable connection.

- **Mechanism:** The frontend communicates with backend microservices via secure **HTTPS** requests, utilizing **JSON** as the standard data interchange format.
- **Design Rationale:** * **Statelessness:** By adhering to REST conventions, each request contains all necessary information for processing, allowing the backend to scale horizontally without session affinity issues.
    - **Predictability:** The use of standard HTTP methods (GET, $POST$, PUT, $DELETE$) and status codes provides a consistent contract for frontend developers and simplifies debugging.
    - **Efficiency:** This direct communication path ensures low-latency responses for user-facing actions, such as form submissions or data retrieval.

2. Internal Communication: Backend–Backend (East-West Traffic)

To maintain strict decoupling between microservices, the system employs an **Event-Driven Architecture (EDA)** for inter-service orchestration.

- **Mechanism:** Asynchronous communication is facilitated by **RabbitMQ**, acting as a robust message broker. Services interact through a **Publisher-Subscriber (Pub/Sub)** model rather than direct, blocking calls.
- **Event Orchestration:**
    - **User Lifecycle Events:** When a "User Created" event is published, independent services (e.g., Notification Service, Profile Service) consume this message to trigger welcome emails or initialize default settings without the primary service waiting for completion.
    - **Transaction Workflows:** Events such as "Offer Accepted" are broadcast to the message bus, allowing the Payment and Scheduling services to react in parallel, ensuring high throughput.
- **Why this approach?**
    - **Fault Tolerance:** If a downstream service (like the Notification Service) is temporarily offline, RabbitMQ queues the message, ensuring no data is lost and the system maintains **eventual consistency**.

- **Reduced Latency:** The primary service can return a "Success" response to the user immediately after publishing an event, rather than waiting for all secondary side effects to finish.
- **Loose Coupling:** Microservices remain "unaware" of each other's internal implementations; they only need to agree on the message schema.

3. Comparison of Communication Flow

Table 29 comparison of communication flow

Communication Type	Protocol / Tool	Nature	Primary Benefit
Frontend to Backend	RESTful APIs (HTTP/JSON)	Synchronous	Immediate user feedback and simplicity.
Service to Service	RabbitMQ (AMQP)	Asynchronous	High availability and system decoupling.

5. Multi-Task Deep Learning for Automated Home Service Classification and Cost Estimation

Subsequent chapter: Multi-task Learning Model

6. Component Diagram

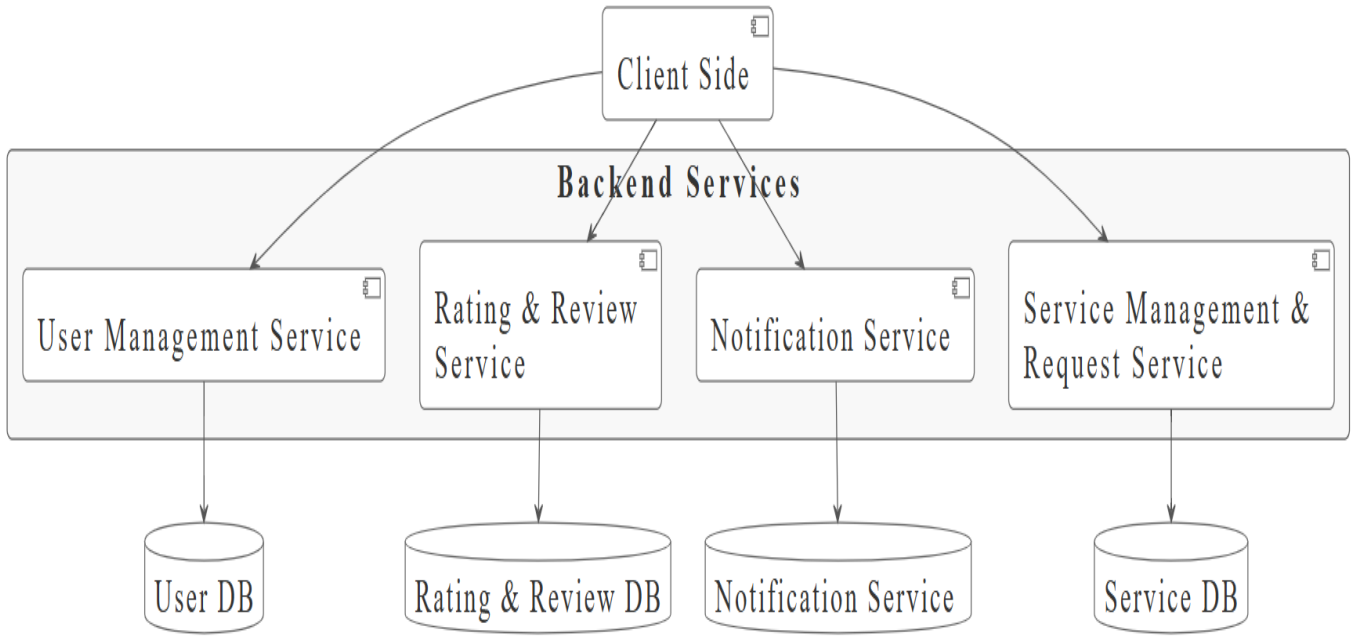


Figure 47 Component Diagram

7. ERD Diagrams

- User Management

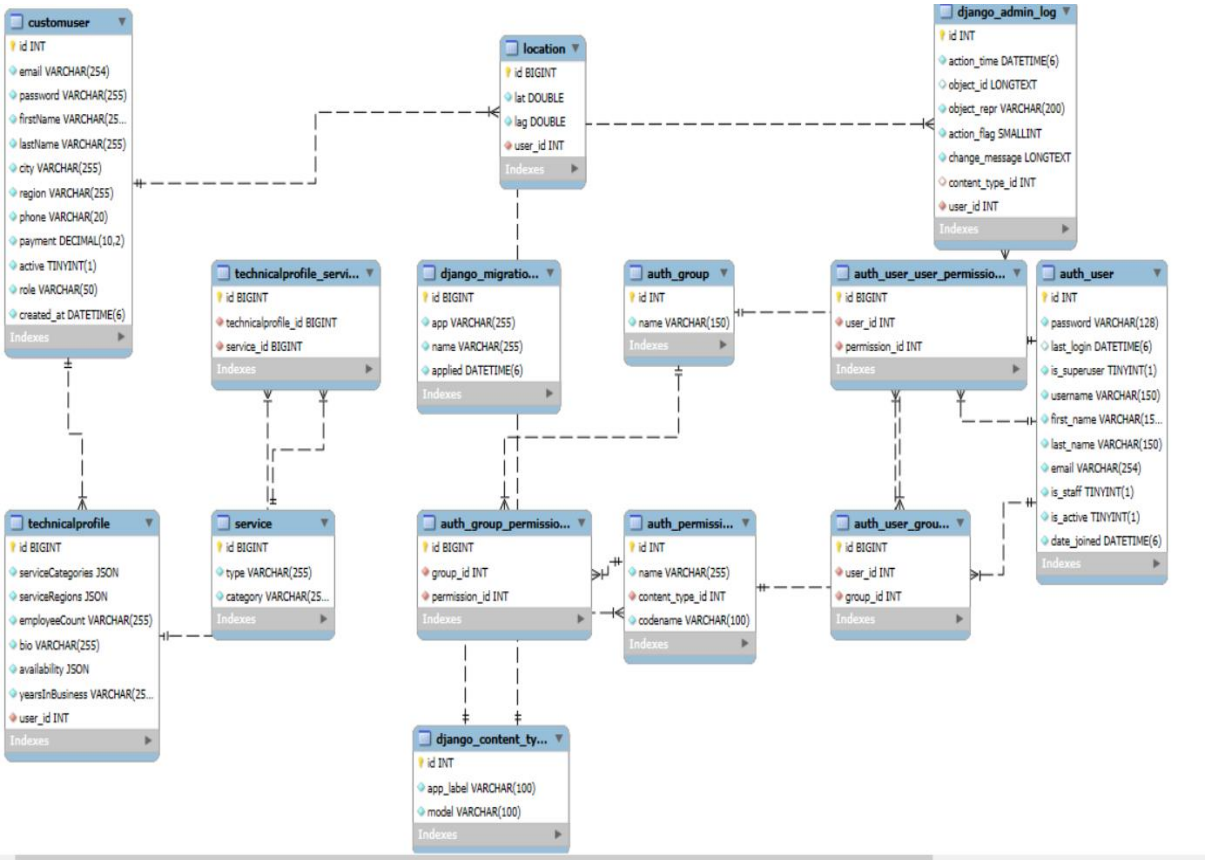


Figure 48 Erd- User Mgmt.

- Service Management & Request

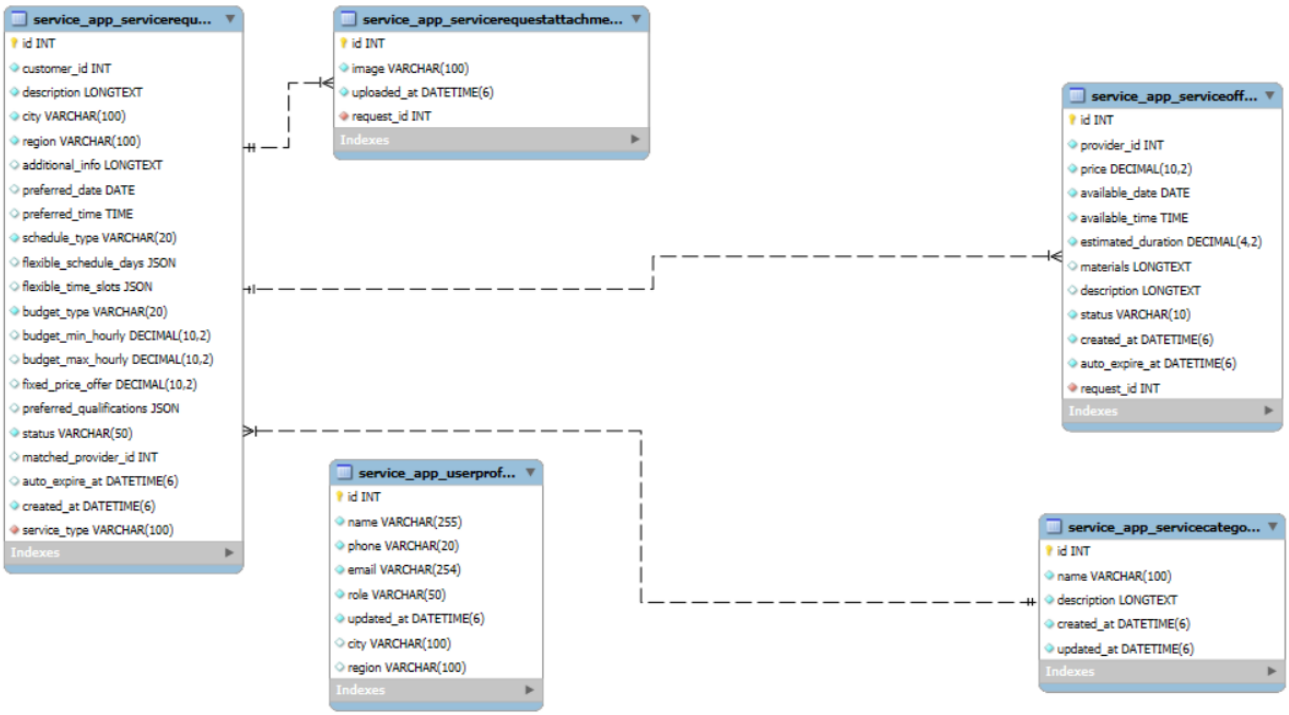


Figure 49 Erd- Service Mgmt. & Request

- Notification Service

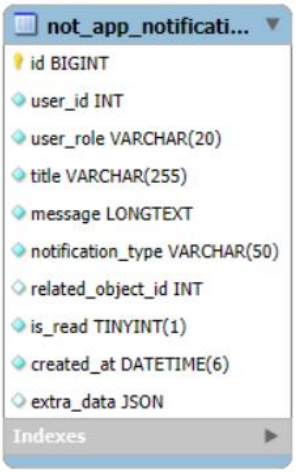


Figure 50 Erd- Notification Service

- Rating & Review Service

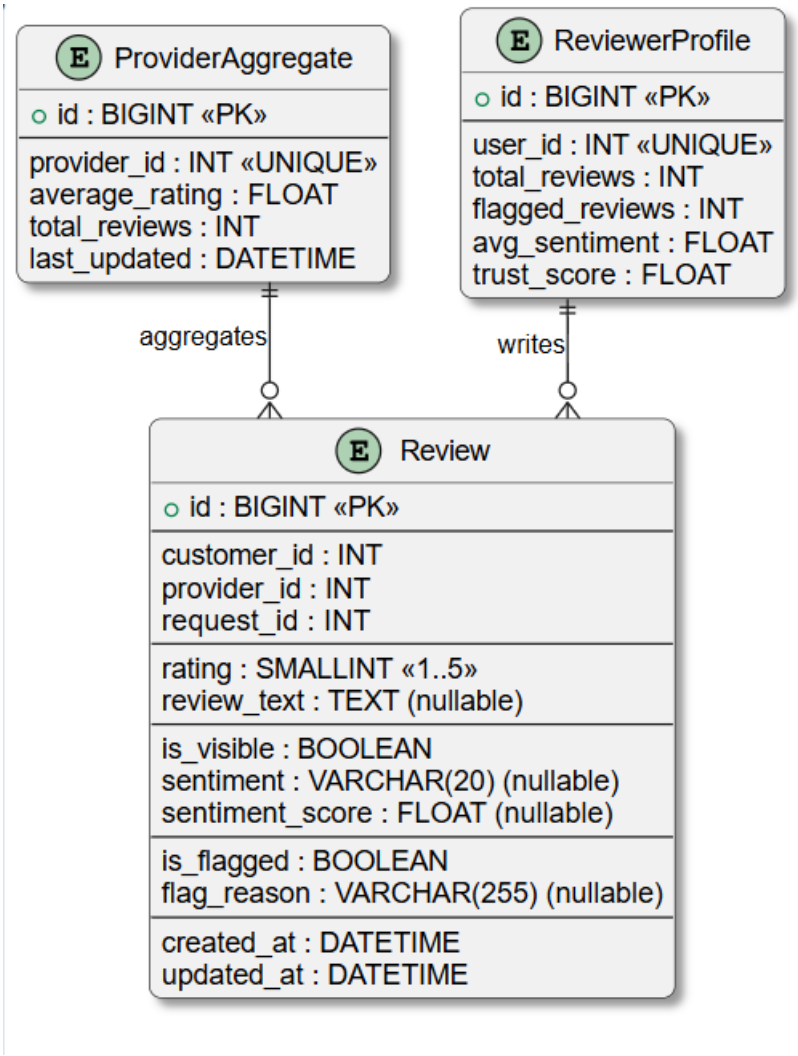


Figure 51 rating & review service

- Notification module with Service Request Service modules



Figure 52 Erd- Notification with Service Request

Chapter 6 Multi-Task Deep Learning for Automated Home Service Classification and Cost Estimation

Abstract

This project presents an automated computer vision system for classifying home service requirements and estimating repair costs from images. The manual assessment of home damages is often subjective, time-consuming, and prone to inconsistency. To address this, we developed a multi-task deep learning framework capable of simultaneously identifying the required service category (e.g., Plumbing, Carpentry) and predicting the associated cost. The system utilizes an EfficientNet-B4 backbone enhanced with feature embeddings for auxiliary inputs (severity, damage area, complexity). We conducted a rigorous ablation study across multiple model versions, evolving from a baseline simple regressor to a robust K-Fold validated architecture with dynamic loss weighting. Experimental results on a dataset of ~750 annotated images demonstrate that our final model achieves **95.5% classification accuracy** and an **R^2 score of 0.40** in cross-validation. Evaluation on an independent unseen test set yielded **100% accuracy** and an **R^2 of 0.46**, validating the system's generalization capability.

1. Introduction

The home services industry relies heavily on accurate initial assessments of damage to provide fair cost estimates and deploy the correct trade professionals. Traditionally, this process involves on-site visits or manual review of customer-submitted photos, which creates bottlenecks and inconsistencies in pricing.

Recent advancements in Deep Learning (DL) and Computer Vision (CV) offer the potential to automate this workflow. However, this specific domain presents unique challenges:

1.Dual Objective: The system must determine _*what*_ is wrong (Classification) and _*how expensive*_ it is to fix (Regression).
2.Small Data Regimes: High-quality annotated datasets for home damages are scarce compared to general object detection datasets.
3.Subjectivity: Cost depends not just on the visual appearance of damage but also on severity, complexity, and affected area.

This project proposes a **Multi-Task Learning (MTL)** approach to solve these challenges. By jointly training for classification and regression, the model learns robust feature representations that benefit both tasks. We systematically improved the system through five architectural iterations, incorporating transfer learning, auxiliary feature fusion, and uncertainty-based loss weighting.

2. Literature Review

2.1 Image-Based Cost Estimation

Visual inference for pricing is an emerging field. Preliminary studies in vehicle damage assessment have shown that Convolutional Neural Networks (CNNs) can correlate visual damage patterns with repair costs. However, mapping visual features directly to monetary values is difficult due to the high variance and non-linear relationship between visual extent and actual cost. Unlike classification, where decision boundaries are discrete, regression requires learning a continuous manifold, often necessitating robust loss functions like Huber or Log-Cosh to mitigate the impact of outliers.

2.2 Multi-Task Learning (MTL) in Computer Vision

MTL improves generalization by leveraging the domain-specific information contained in the training signals of related tasks.

- **Hard Parameter Sharing**: The most common approach, where the hidden layers are shared between all tasks, keeping task-specific output layers. This reduces the risk of overfitting on small datasets.
- **Joint Optimization**: Balancing the learning objectives is critical. Kendall et al. (2018) introduced uncertainty-based weighting, allowing the model to dynamically adjust the influence of each task during training, which we adopted in our final architecture.

2.3 Transfer Learning with EfficientNet

For limited datasets, training deep networks from scratch is impractical. Tan & Le (2019) introduced EfficientNet, which uses compound scaling to achieve state-of-the-art accuracy

with fewer parameters. EfficientNet-B4 was selected for this project due to its optimal balance of capacity and computational efficiency, making it suitable for extracting rich features from high-resolution texture images of home damages.

2.4 Regression from Images

Direct regression from high-dimensional image data is unstable. Common challenges include:

- **Scale Sensitivity**: Large variations in target values (costs ranging from $10 to $1000).
- **Metric Selection:** Mean Squared Error (MSE) is standard but widely criticized for being overly sensitive to outliers. Mean Absolute Error (MAE) and Log-Space regression are often preferred choices to stabilize training gradients.

2.5 Learning with Limited Data

Overfitting is a primary concern with small datasets (~750 images). Strategies such as **Cross-Validation**, **Data Augmentation** (geometric and color transformations), and **Regularization** (Dropout, Weight Decay) are essential. Our work emphasizes stratified K-Fold cross-validation to ensure reliable performance estimates.

3. Dataset Description

The dataset was constructed specifically for this project by scraping relevant images from Bing.

- **Total Size**: 750 Images (approx.) + 30 Independent Test Images.
- **Classes (6 Services)**:

1. Carpentry

2. Tiling
3. Glass
4. Electrician
5. Plumbing
6. Painting

- **Annotations**:
  - **Class Label:** One of the 6 services.
  - **Cost:** Numeric value (USD 10–1000).
  - **Auxiliary Attributes:** Severity (1-5), Damage Area Ratio (0-1), Complexity Level (1-4).
- **Distribution:** The dataset is balanced for classification but exhibits a long-tail distribution for costs, motivating the use of log-transformation.

4. System Architecture

The implemented system is an end-to-end Deep Learning pipeline implemented in PyTorch.

4.1 Backbone & Feature Extraction

We utilize **EfficientNet-B4** pretrained on ImageNet as the feature extractor. The final classification layer is removed, and the network outputs a feature vector of size 1792.

4.2 Auxiliary Feature Integration

To improve cost estimation, non-visual metadata is fused with visual features:

- **Severity**: Processed through a dedicated embedding layer (5 to 16dimensions).
- **Damage Area & Complexity**: Processed through a dense layer (2 to 8dimensions).
- **Fusion**: The visual vector (1792) is concatenated with these embeddings (1792 + 16 + 8 = 1816 dimensions).

4.3 Multi-Task Heads

The fused feature vector feeds into two separate branches:

1. **Classification Head**:
   - Structure: FC (256) to BatchNorm to ReLU to Dropout to FC (6).
   - Output: Logits for the 6 service classes.
2. **Regression Head**:
   - Structure: A deeper MLP (512 to 256 to 128 to 1).
   - Each stage includes Batch Normalization and Dropout (p=0.3) to prevent overfitting.
   - Output: Single scalar value (normalized log-cost).

5. Mathematical Background

5.1 Loss Functions

The network minimizes a joint loss function L_{total}:

1.**Classification Loss**: Cross Entropy with Label Smoothing (epsilon=0.1).

$$L_{cls} = -\sum_{i=1}^{C} y_i \log(p_i)$$

2.**Regression Loss**: Huber Loss is used for its robustness to outliers.

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta, \\ \delta(|a| - \frac{1}{2}\delta) & \text{otherwise.} \end{cases}$$

3.**Dynamic Weighting**:

Instead of manually tuning weights, we learn the optimal task balance using learnable parameters sigma_1, sigma_2 (homoscedastic uncertainty):

$$L_{total} = \frac{1}{2\sigma_1^2} L_{cls} + \frac{1}{2\sigma_2^2} L_{cost} + \log(\sigma_1) + \log(\sigma_2)$$

6. Model Versions & Ablation Study

A key contribution of this project is the iterative refinement of the model. We experimented with five distinct versions.

Version 1: The Baseline:

- **Architecture**: Basic EfficientNet-B4 with two simple linear heads.
- **Features**: Image input only. No auxiliary data.
- **Training**: Standard train/val split. No cross-validation. Simple MSE Loss.
- **Outcome**: High variance in cost prediction; prone to overfitting.

Version 2: Enhanced Features:

- **Improvement**: Introduction of **Auxiliary Features** (Severity, Complexity).
- **Architecture**: Added embedding layers for metadata.
- **Loss**: Switched to **Combined Loss** (Fixed weights).
- **Outcome**: improved R^2 significantly as severity is a strong predictor of cost.

Version 3: Cross-Validation:

- **Improvement**: Implemented **5-Fold Cross-Validation**.
- **Motivation**: Given the small dataset size, a single validation split was unreliable.
- **Outcome**: Provided a true estimate of model generalization performance.

Version 4: Architectural Search (ConvNeXt)

- **Experiment**: Tested **ConvNeXt-Tiny**as an alternative backbone.
- **Motivation**: To see if modern Vision Transformers/ConvNets outperformed EfficientNet.
- **Outcome**: Comparison showed EfficientNet-B4 remained superior for this specific dataset size, likely due to better inductive biases for texture.

Version 5: The Final Model:

- **The Culmination**: Combines all successful components:
  - EfficientNet-B4 Backbone.
  - Full Auxiliary Embeddings.

- **Dynamic Loss Weighting** (Kendall et al.).
- **Two-Stage Training**: Linear Warmup (Frozen Backbone) to Fine-tuning (Differential Learning Rates).
- **Log-Space Regression**: Predicting **log**(cost) to handle order-of-magnitude differences.

7. Experiments & Results

7.1 Experimental Setup

- **Environment**: Google Colab (GPU-accelerated).
- **Validation**: 5-Fold Stratified Cross-Validation.
- **Optimizer**: AdamW with Cosine Annealing.
- **Batch Size**: 8.

7.2 Cross-Validation Results (Version 5)

Metric	Mean Value	Std. Deviation
Accuracy	95.50%	± 0.92%
MAE (Mean Absolute Error)	$36.66	± $5.07
R^2 Score	0.402	± 0.046

The model demonstrates exceptional classification performance (>95%). The R^2 score of ~0.40 is respectable for the highly subjective task of cost estimation from images without depth or 3D information.

7.3 Final Evaluation on Unseen Test Data

A strictly held-out test set of 30 images was used to evaluate the final model ensemble. These images were never seen during training or CV.

Metric	Value
Test Accuracy	100.00%
Test R^2 Score	0.4661
Test MAE	$23.46
Test RMSE	$34.10

Analysis: The Test R^2 (0.46) is *_higher_* than the validation mean (0.40), indicating that the model has **not overfitted** and generalizes well to new query images. The accuracy of 100% on the test set confirms the robustness of the classification head.

8. Discussion

- **Impact of Auxiliary Features**: The ablation study confirmed that visual features alone are insufficient for accurate cost prediction. Metadata like "Severity" guides the regression head significantly.
- **Model Stability**: The transition to Version 5 (Dynamic Weighting + Log Transform) notably reduced the variance across folds (lower Std Dev), making the system reliable for deployment.
- **Limitations**: The dataset size (750 images) is the primary bottleneck. While transfer learning mitigates this, the model struggles with extremely high-cost outliers ($>800) due to their sparsity in the training data.

9. Recommendations & Future Work

1.**Dataset Expansion**: Increasing the dataset to 5,000+ images would likely boost R^2 scores above 0.60.

2.**Object Detection Integration**: Incorporating bounding box detection (e.g., YOLO) to localize damage before estimation could improve focus and accuracy.

3.**Uncertainty Quantification**: Implementing Bayesian approximation (MC Dropout) during inference would allow the system to output a confidence interval for the price (e.g., "$100 ± $20"), which is more valuable for business logic.

10. Conclusion

This project successfully demonstrated the feasibility of automating home service assessment using multi-task deep learning. By rigorous experimentation and architectural refinement, we developed a system that classifies service types with near-perfect accuracy and estimates costs with a promising degree of correlation. The final architecture—combining EfficientNet, auxiliary feature fusion, and dynamic uncertainty weighting—represents a robust baseline for future automated insurance and service dispatch systems.

Chapter 7 Practical Implementation

1. Introduction

In this chapter, we will present the practical implementation of the system, detailing the technologies and tools employed. Additionally, we will showcase the system interfaces and conclude by executing test cases to verify the system's functionalities across different scenarios.

2. Used tools

2.1 Core Development Frameworks

2.1.1 Backend: Django & Django REST Framework (DRF)

Django was selected as the primary backend framework for its "batteries-included" philosophy.

- **DRF Integration:** We utilized the **Django REST Framework** to build a standardized API layer. This allowed for seamless serialization of complex model data and the implementation of robust authentication policies.
- **Role:** Handles business logic, database ORM, and provides the interface for both the frontend and the AI orchestration nodes.

2.1.2 Frontend: React.js

React serves as the client-side engine, providing a responsive and dynamic user experience.

- **Component-Based Architecture:** By using reusable UI components, we ensured consistency across the service provider and customer dashboards.
- **Role:** Excels in creating interactive web applications where UI components update efficiently as underlying data changes.

2.1.3 Database: MySQL

A relational database management system (RDBMS) was chosen to ensure ACID compliance and data integrity.

- **Role:** Stores persistent data including user profiles, service request logs, and historical pricing data.

2.2 Workflow Orchestration & AI Stack

2.2.1 Orchestration: LangGraph

LangGraph is a library designed for building stateful, multi-actor applications.

- **Role:** It acts as the "orchestrator" for the service request lifecycle. Instead of simple linear code, LangGraph manages the **State** of a request as it moves through various nodes (e.g., VLM Analysis → Budget Estimation → Time Suggestion).
- **Benefit:** It allows for complex, cyclical logic and maintains a persistent "checkpoint" of the automation state, making the system's decision-making process highly observable and robust.

2.2.5 AI Backbone: PyTorch & TorchVision

These libraries form the foundation of our image analysis capabilities.

- **TorchVision:** Used for pre-processing service request images (resizing, normalization) to prepare them for the neural network.

2.2.6 EfficientNet-B4

Utilized as the pre-trained backbone for robust feature extraction.

- **Rationale:** It provides a state-of-the-art balance between accuracy and computational efficiency, allowing the system to "understand" the visual context of a service request photo.

2.3 Asynchronous Processing & Messaging

2.3.1 RabbitMQ & Celery

To prevent long-running tasks from blocking the user interface, we implemented an asynchronous task queue.

- **RabbitMQ:** Acts as the message broker, facilitating decoupled communication between microservices.
- **Celery:** The distributed task queue that processes events (like "User Created" or "Offer Accepted") in the background.

2.3.2 Redis

Redis is utilized as a high-performance, in-memory data store.

- **Role:** Serves as the **Result Backend** for Celery and facilitates fast data caching to reduce database load.

2.4 Performance & Load Testing Suite

2.4.1 Grafana k6

A modern load-testing tool used to script performance tests.

- **Role:** Specifically used to identify p(95) response times and detect bottlenecks, leading to the optimization of the N+1 query problem.

2.4.2 Locust

An open-source user load testing tool.

- **Role:** Used to simulate "swarms" of concurrent users to verify system stability under extreme load.

2.5 Infrastructure & API Documentation

2.5.1 Docker

The entire system is containerized to ensure "environment parity."

- **Role:** Ensures that the system runs identically across development, testing, and production environments.

2.5.2 Swagger & Postman

- **Swagger:** Provides interactive API documentation, allowing developers to test endpoints directly from the browser.
- **Postman:** Used for manual API validation and environment variable management.

2.6 Collaboration & IDE

2.6.1 Bitbucket

A Git-based source code repository hosting service.

- **Role:** Facilitated version control, collaborative code reviews, and integrated continuous integration (CI) workflows.

2.6.2 Visual Studio Code (VS Code)

The primary Integrated Development Environment (IDE) used for the entire project lifecycle, providing powerful debugging and IntelliSense for both Python and JavaScript.

2.7 Summary of Tooling Purpose

Category	Tools	Primary Benefit
Logic & UI	Django, React, MySQL	Reliability and rapid UI/UX development.
Intelligence	LangGraph, PyTorch, EfficientNet	Sophisticated, stateful AI orchestration.
Concurrency	RabbitMQ, Celery, Redis	Non-blocking, responsive user experience.
Validation	k6, Locust, Postman	Performance assurance and API integrity.
Infrastructure	Docker, Bitbucket, Swagger	Portability, version control, and documentation.

3. System interfaces

- ❖ Homepage Interface – Search and Navigation for Home Care

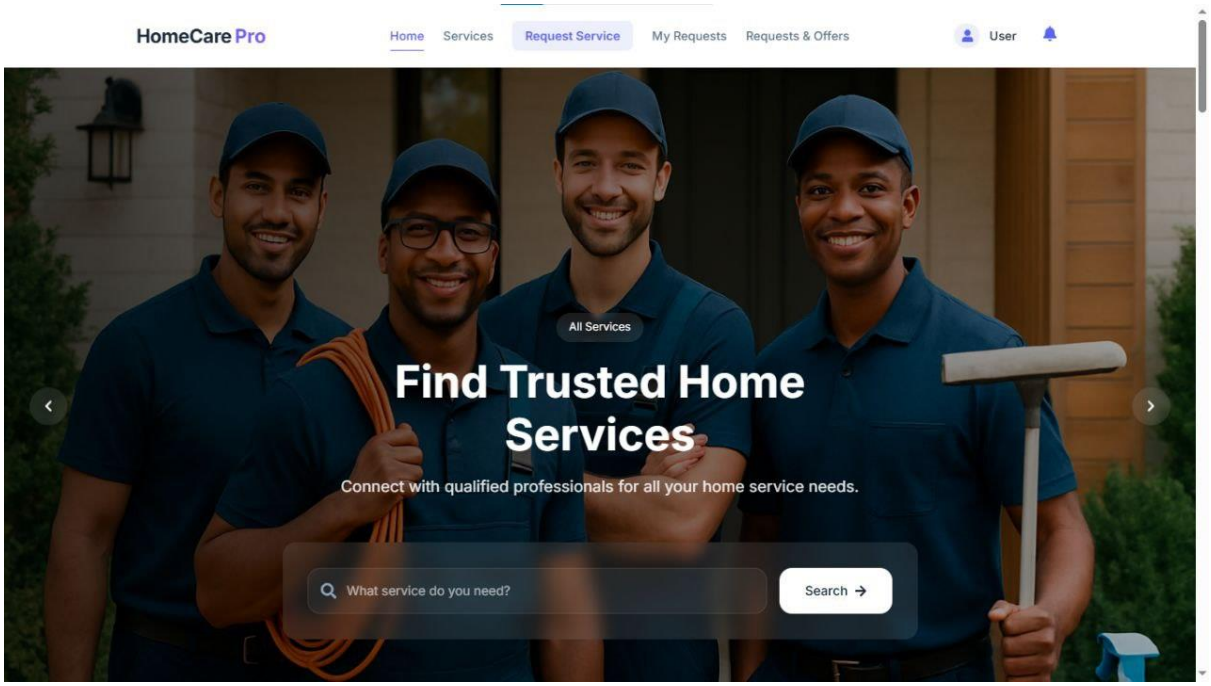


Figure 53 home page

❖ How It Works – Four-Step Service Journey for Home Care

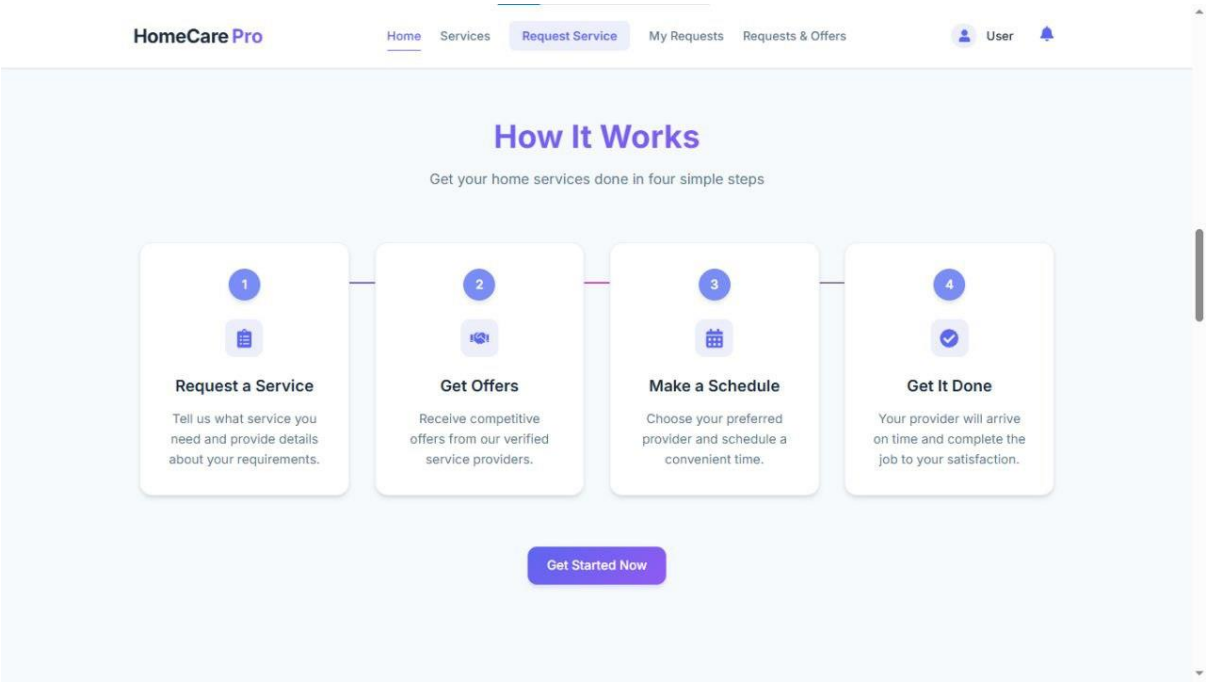


Figure 54 how it works page

❖ Customer Dashboard – View and Accept Provider Offers for Cleaning Services

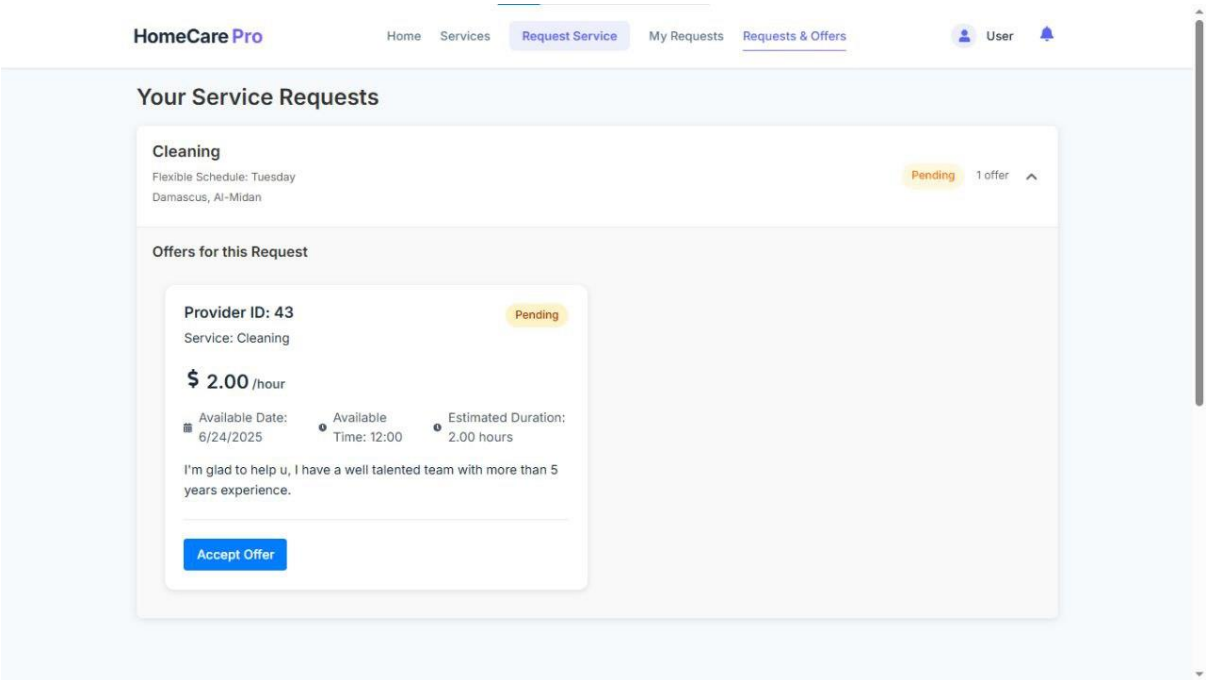


Figure 55 Request and offers page – customer

❖ Featured Services – Explore Home Care Offerings by Category

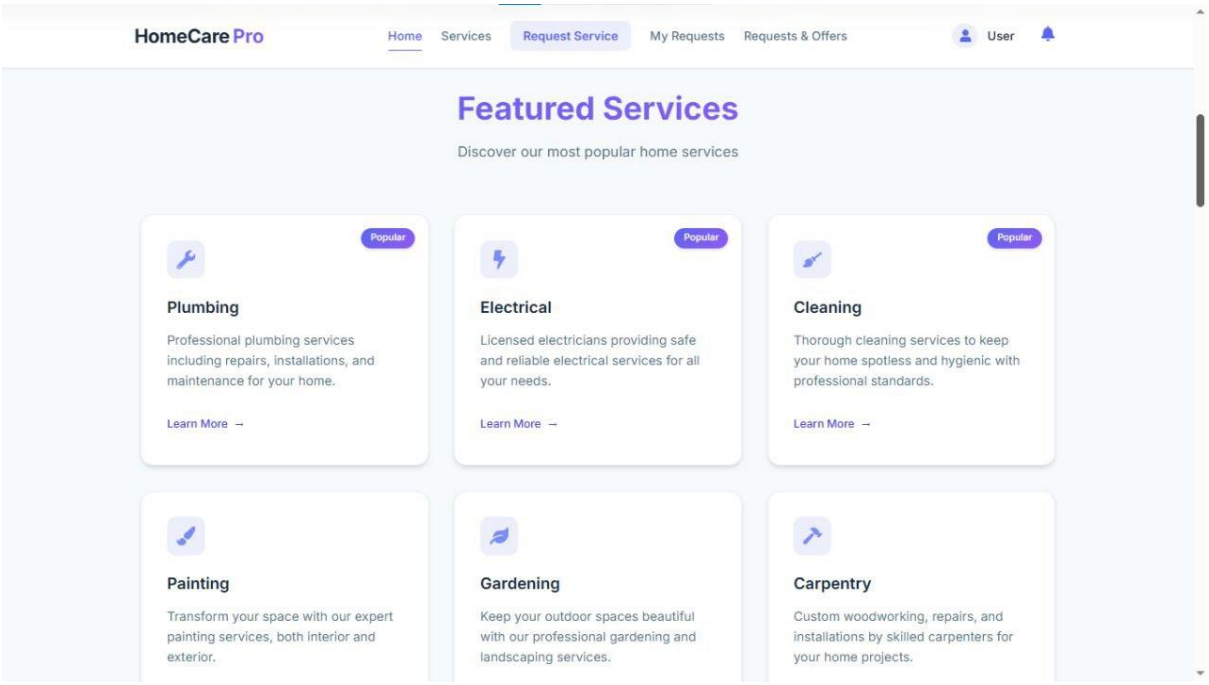


Figure 56 Featured Services page

❖ Sign-Up – Service Provider Account Creation (Basic Info Step)

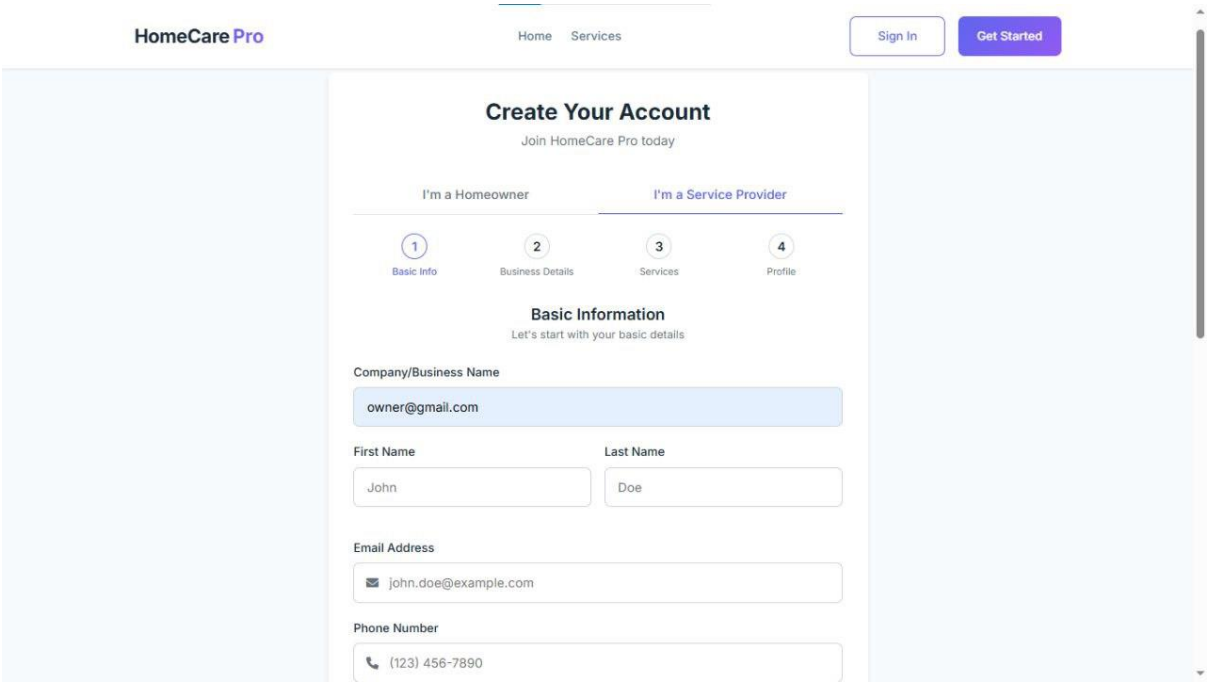


Figure 57 Sign up page

❖ Login Page – Secure Access to Home Care for Returning Users

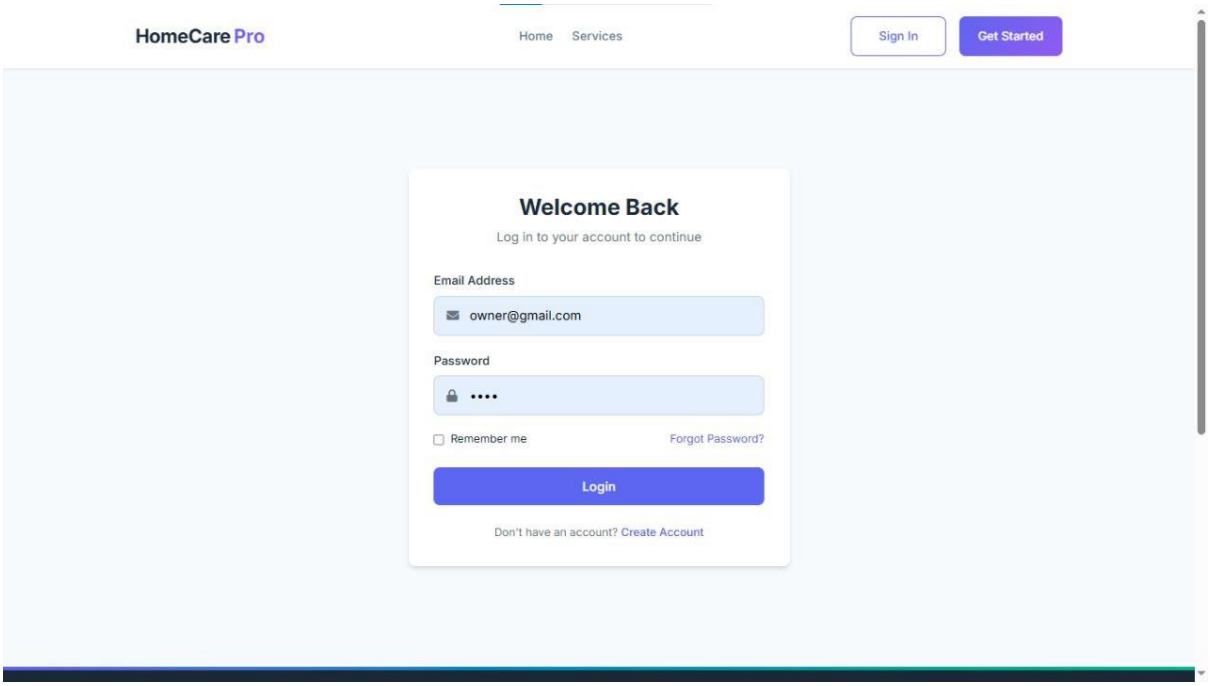


Figure 58 Login page

❖ Account Settings – Update Profile Details Interface (Customer Account)

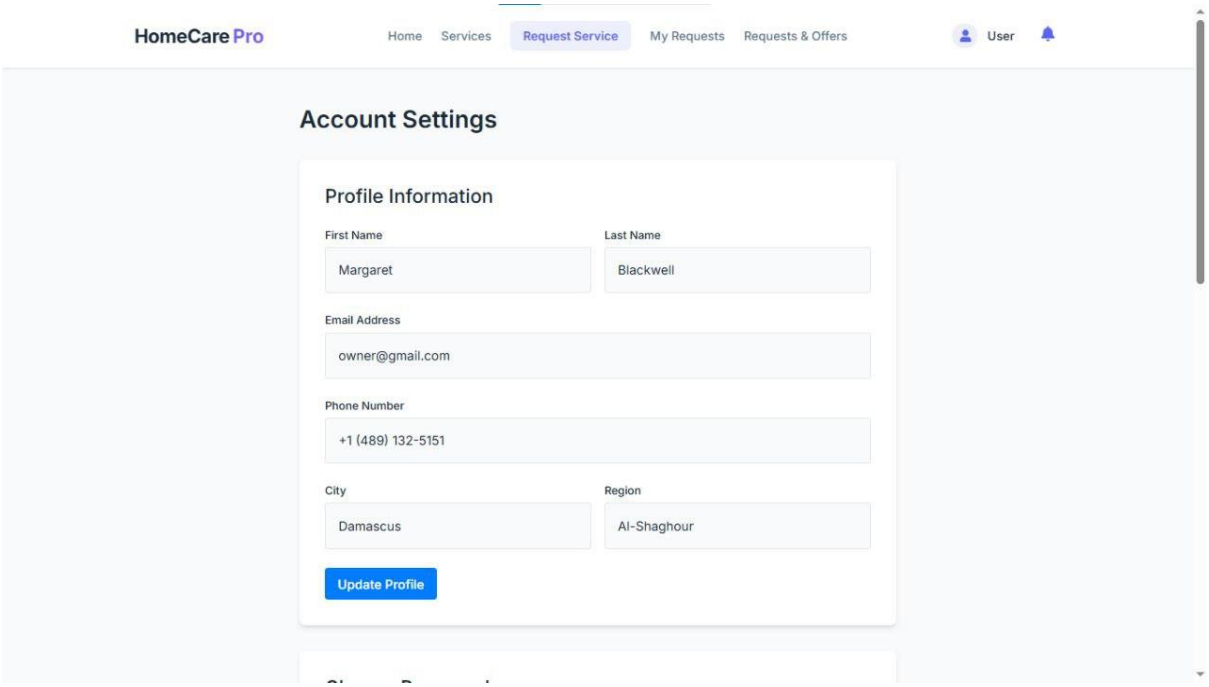


Figure 59 Update profile page

❖ Provider Account Settings – Profile Information for Home Care (Provider Account)

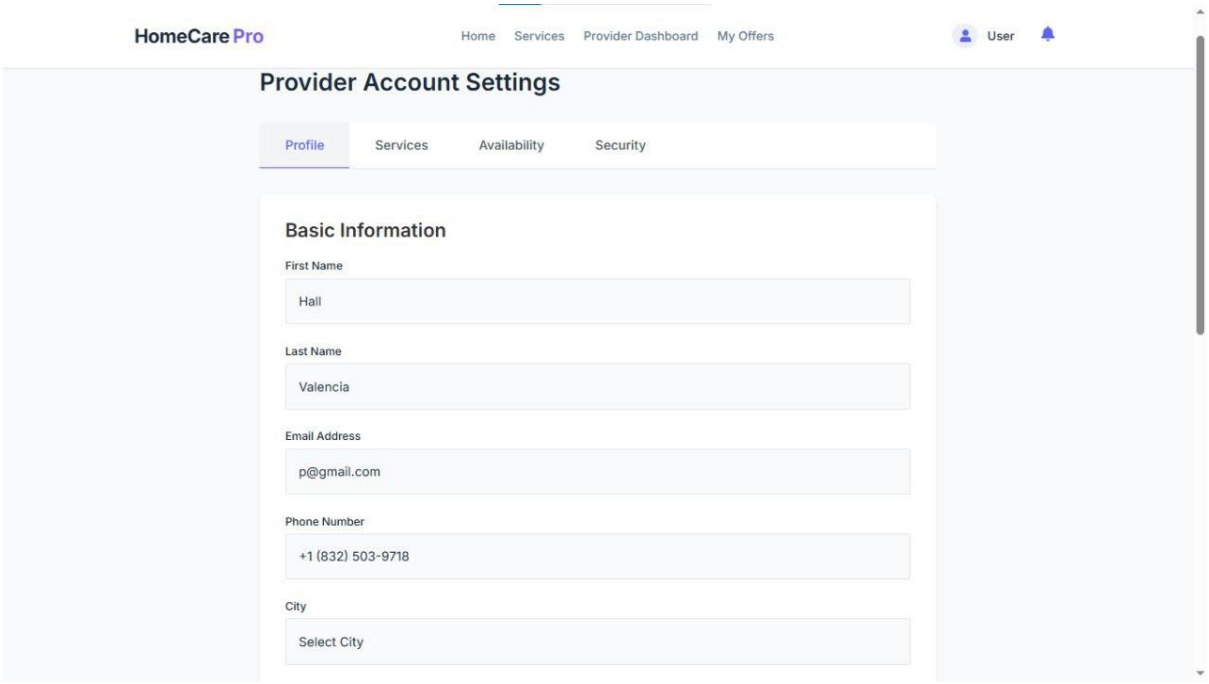


Figure 60 Account settings page

❖ Step 1 – Select a Service Category and Describe the Task

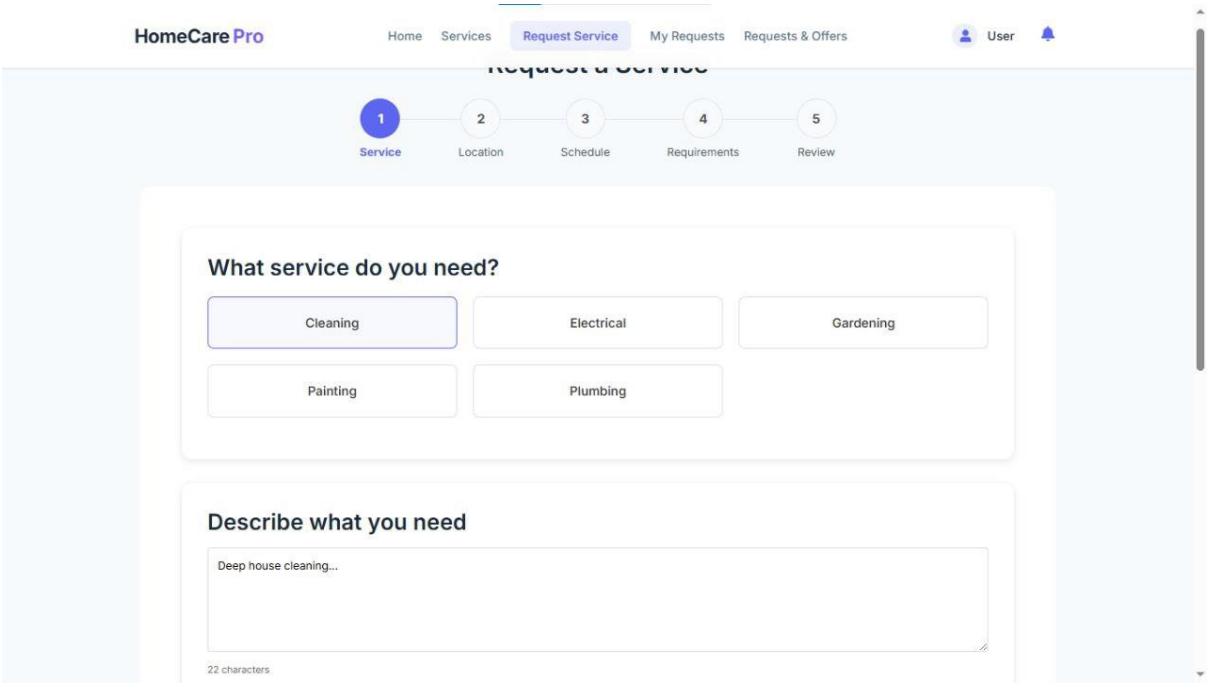


Figure 61 Create request - s1

❖ Request Service – Step 2: Location Information for Cleaning Request

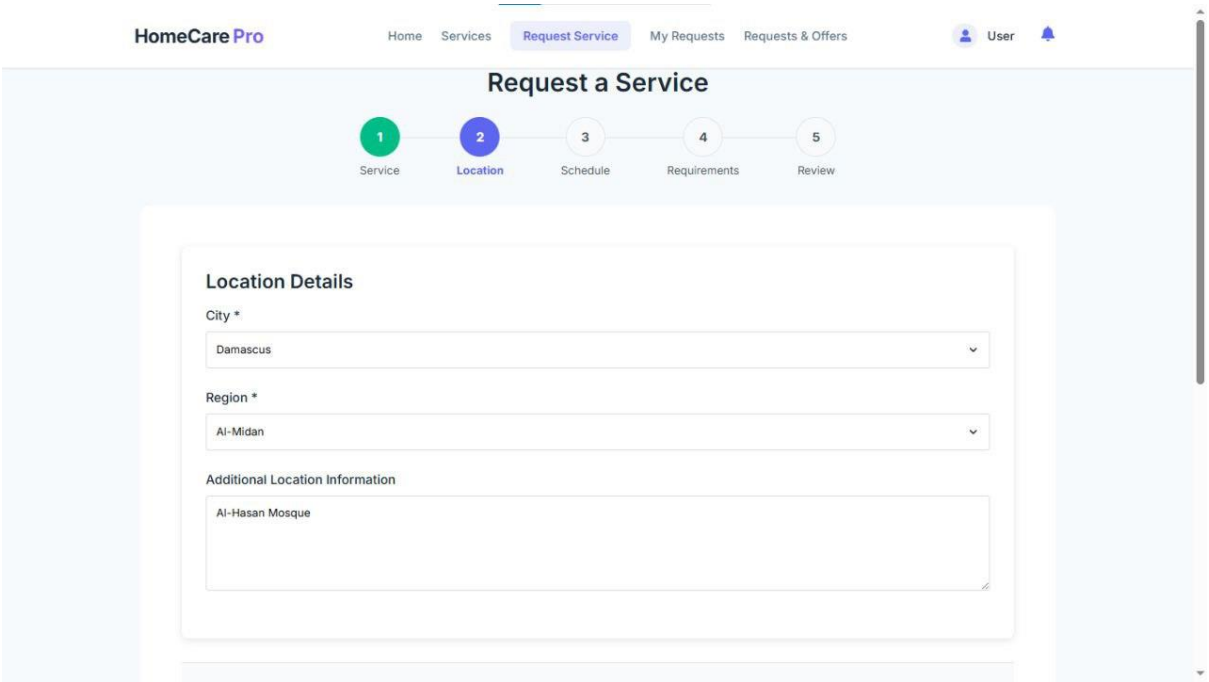


Figure 62 Create request – s2

❖ Step 3 – Schedule Your Preferred Availability for Cleaning Services

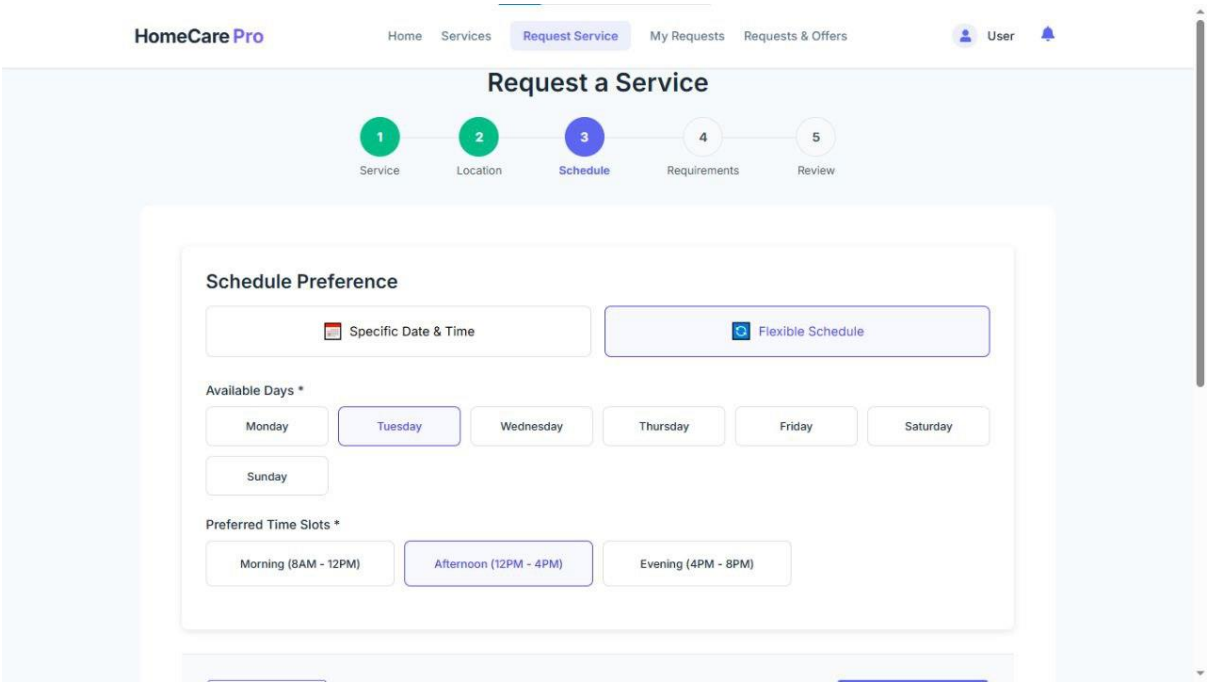


Figure 63 Create request – s3

❖ Request Service – Define Requirements for House Cleaning Task

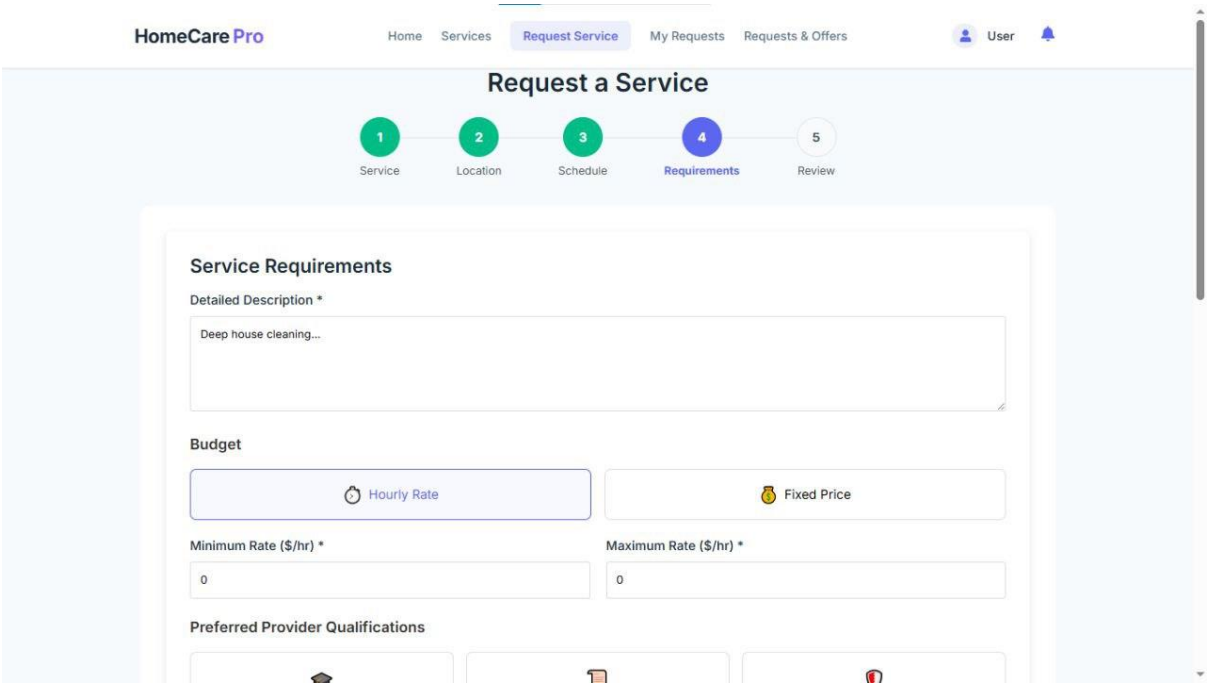


Figure 64 Create request – s4

❖ Step 5 – Review and Submit Final Cleaning Service Request

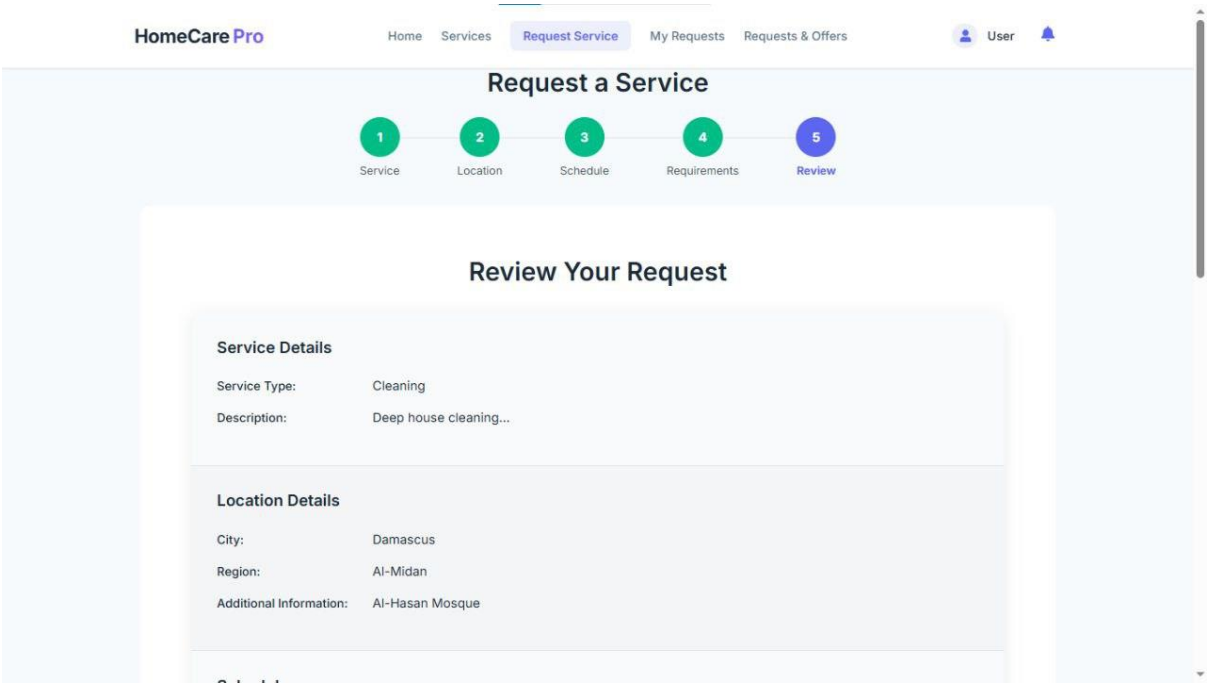


Figure 65 Create request – s5

❖ My Requests – Track and Manage Submitted Service Tasks

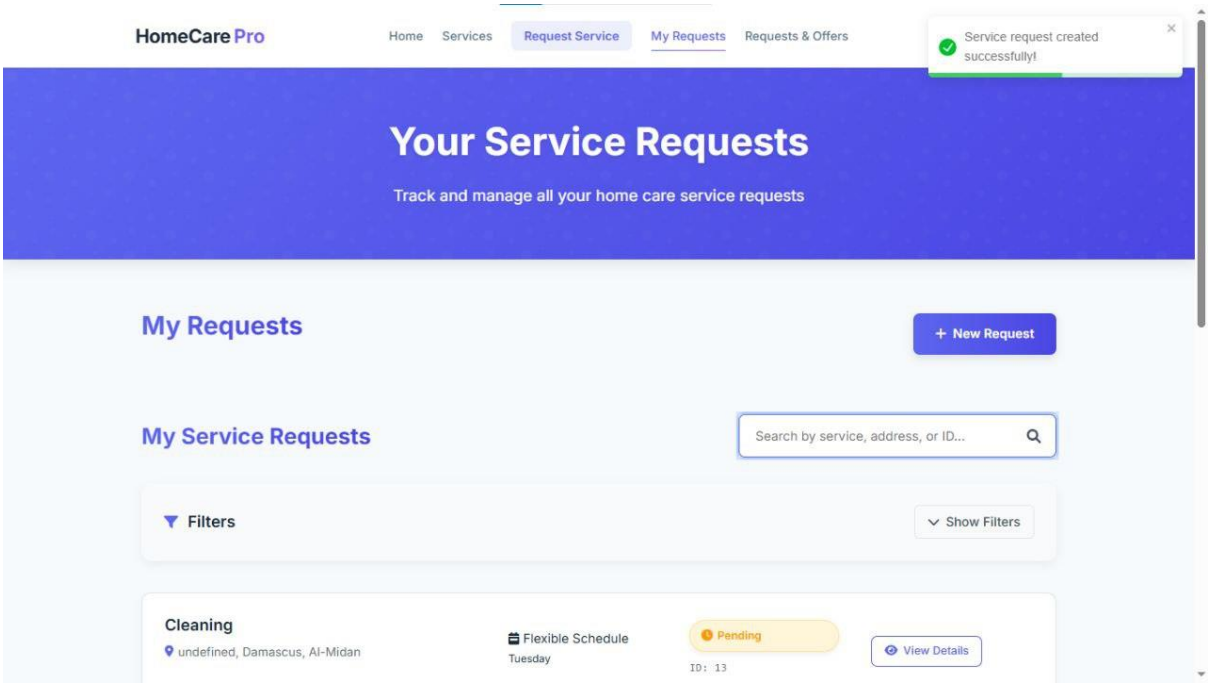


Figure 66 Requests list page

❖ Provider Dashboard Overview – Performance Metrics & Request Filtering

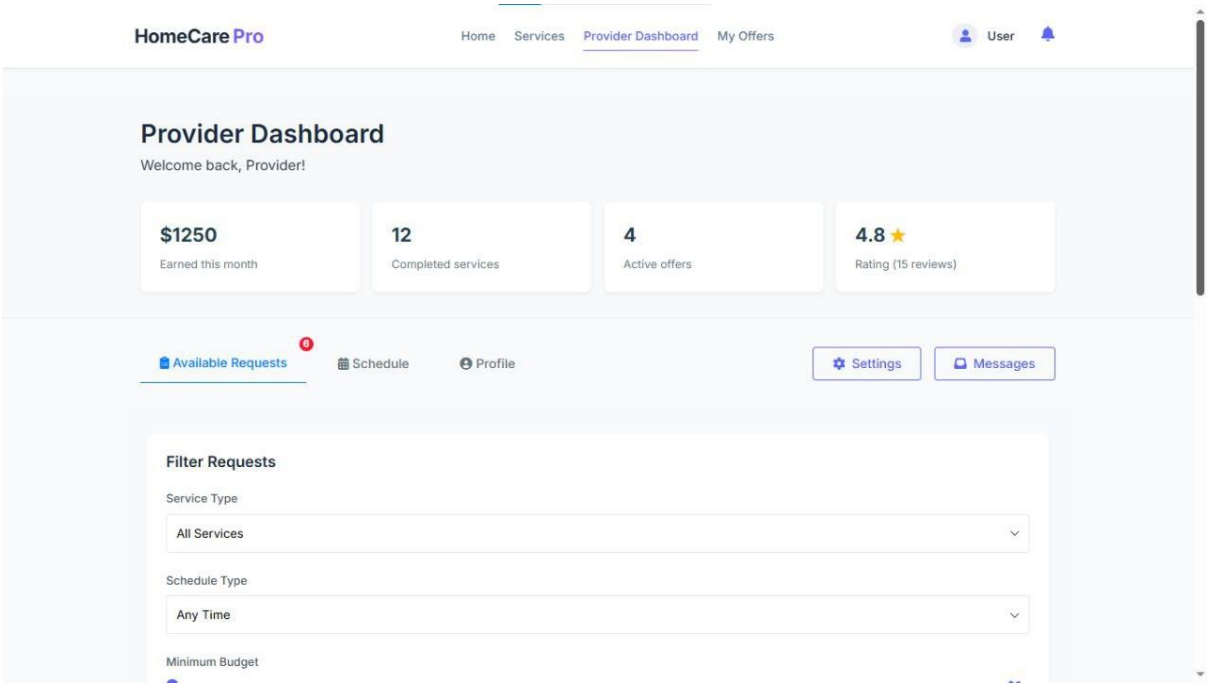


Figure 67 Provider dashboard

❖ Service Request Details – Provider Offer Screen for Cleaning Task

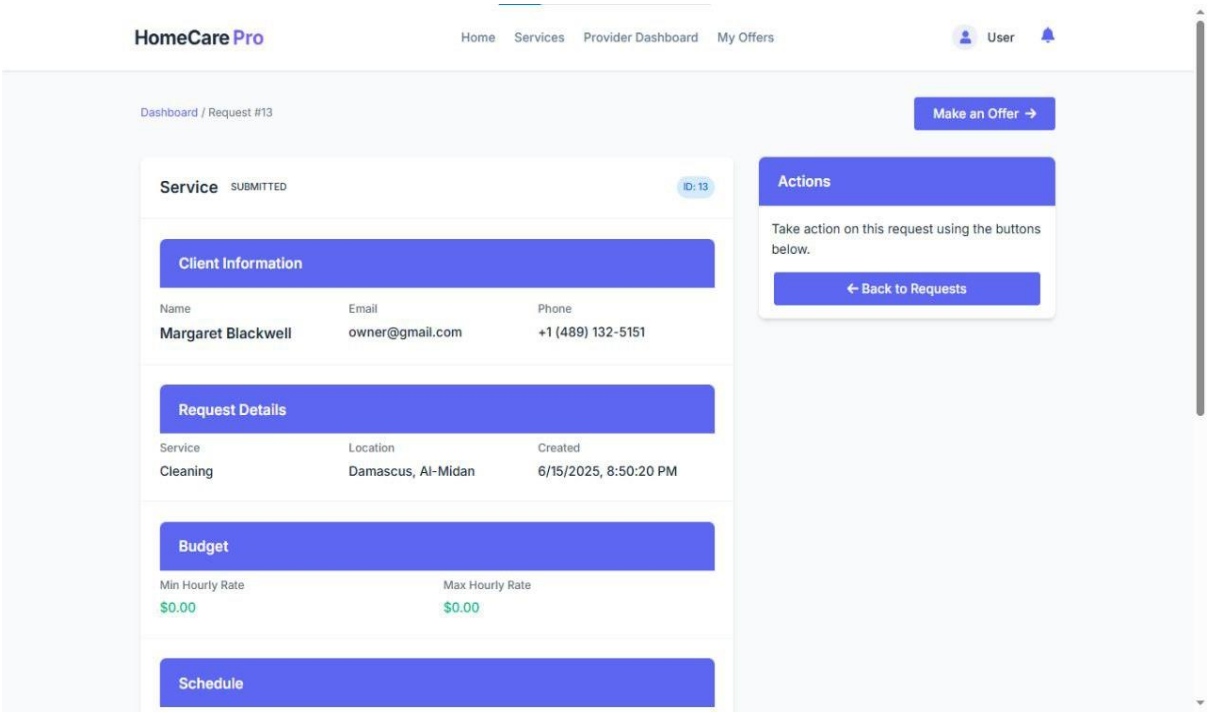


Figure 68 Request details page - provider

❖ My Offers – View Pending Provider Proposals with Offer Details

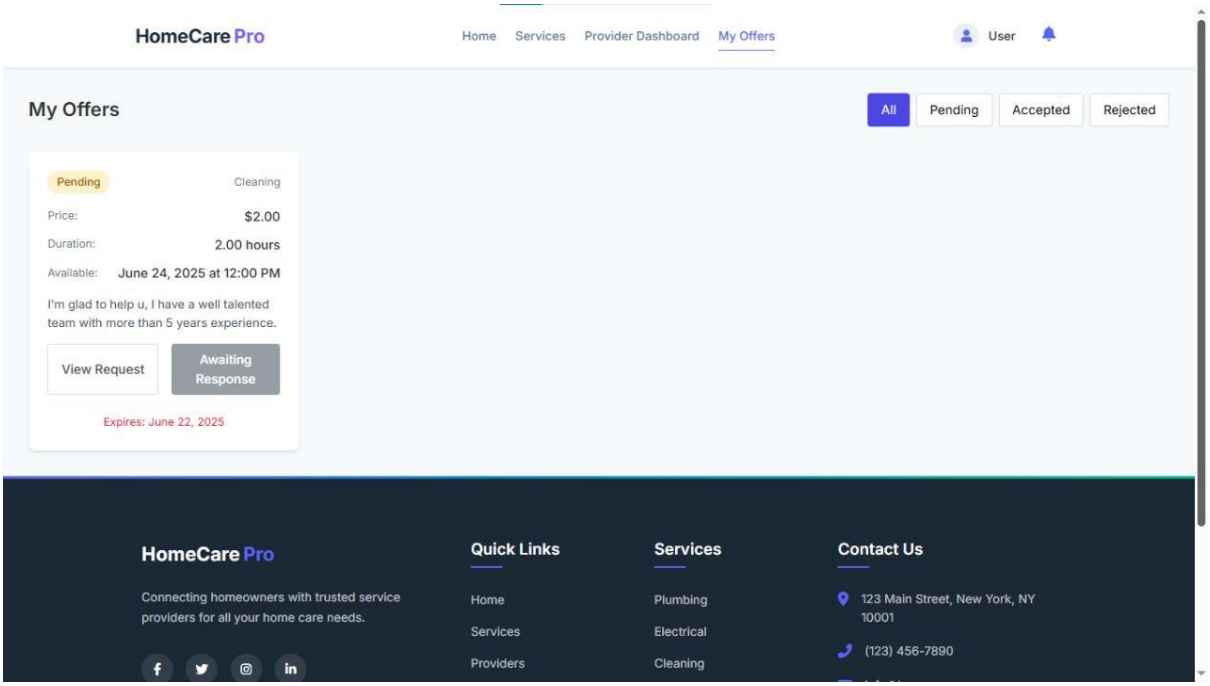


Figure 69 Offers list page - Provider

4. System Testing, Validation, and Performance Engineering

4.1 Overview and Testing Strategy

To ensure the **Home Care Service Management System** meets enterprise standards for reliability and scalability, a multi-layered testing strategy was implemented. This phase extended beyond simple bug detection to include architectural validation, ensuring that the **VLM Facade**, **Async Workflows**, and **Database Layers** function cohesively under stress.

The testing methodology follows the **"Test Pyramid"** approach:

1. **Unit Testing:** Isolating individual components (Nodes, Models) using mocks.
2. **Integration Testing:** Verifying communication between the API, RabbitMQ, and the Database.
3. **Performance Testing:** Subjecting the system to high concurrency to identify and resolve bottlenecks.

4.1.1 Tools and Environment

- **Framework:** Django TestCase & unittest.mock for logic verification.

- **Load Testing:** Grafana k6 (API Metrics) and Locust (User Behavior Simulation).
- **Containerization:** Docker for consistent test environments.

4.2 Functional Testing: Logic Verification

4.2.1 Automated Node Testing (Unit Level)

The core of the automation system relies on "Nodes" (VLM, Budget, Time). These were tested in isolation using mocks to ensure that business logic remains correct regardless of external service availability (e.g., if the AI model is down).

Code: Isolated Workflow Verification

4.2.2 API Contract Testing

To ensure the frontend communicates seamlessly with the backend, **Contract Tests** were implemented. These verify that API endpoints return the exact JSON structure expected by the client, including correct status codes and data types.

4.2.3 Asynchronous Integration Testing

Since the architecture heavily relies on **Celery** and **RabbitMQ**, standard synchronous tests are insufficient. We utilized unittest.mock to intercept calls to .delay(), allowing us to verify that tasks are scheduled correctly without requiring a running worker process during test execution.

4.3 Performance Engineering: The Optimization Journey

A critical objective was ensuring the API could handle **100+ Concurrent Virtual Users (VUs)** with a response time (p(95)) under **100ms**. Initial tests revealed significant bottlenecks, leading to a four-phase optimization process.

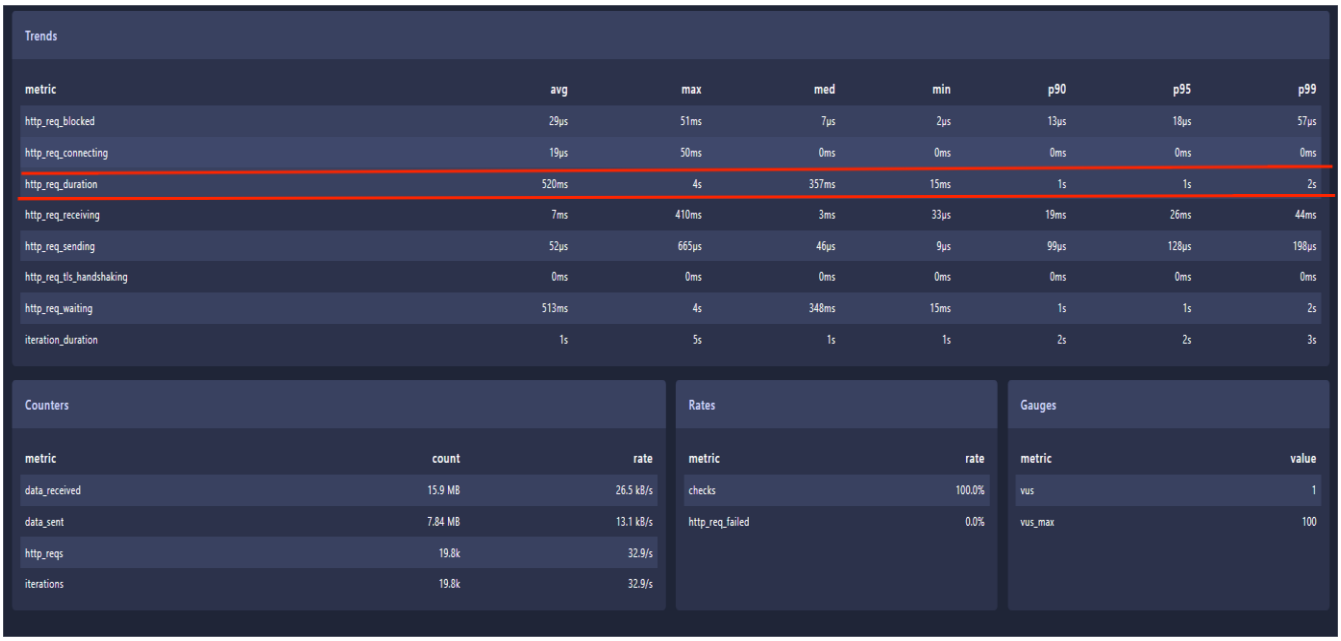
Key Metrics Tracked:

- **Response Time (p(95)):** The time within which 95% of requests are completed.
- **Throughput:** Requests processed per second.

- **Error Rate:** Percentage of failed requests.

4.3.1 Optimization Phases and Results

Phase	Scenario	Detected Bottleneck	Engineering Solution	Result (p(95))
0. Baseline	100 VUs	Initial configuration	N/A	**1.21s (Fail)**
1. Query Opt.	100 VUs	**N+1 Query Problem**	Applied prefetch_related	**511ms**
2. Stress Test	200 VUs	High CPU & Table Scans	**Database Indexing** (B-tree)	**2.05s (Saturation)**
3. Payload Opt.	200 VUs	Large JSON (493MB throughput)	**Lite Serializer** Pattern, **Caching**	**Data Reduced to 163MB, 985ms**
4. Final Scaling	200 VUs	Single-threaded blocking	**Gunicorn (5 Workers)**	**<100ms (Success)**



Trends

metric	avg	max	med	min	p90	p95	p99
http_req_blocked	29µs	51ms	7µs	2µs	13µs	18µs	57µs
http_req_connecting	19µs	50ms	0ms	0ms	0ms	0ms	0ms
http_req_duration	520ms	4s	357ms	15ms	1s	1s	2s
http_req_receiving	7ms	410ms	3ms	33µs	19ms	26ms	44ms
http_req_sending	52µs	665µs	46µs	9µs	99µs	128µs	198µs
http_req_tls_handshaking	0ms	0ms	0ms	0ms	0ms	0ms	0ms
http_req_waiting	513ms	4s	348ms	15ms	1s	1s	2s
iteration_duration	1s	5s	1s	1s	2s	2s	3s

Counters

metric	count	rate
data_received	15.9 MB	26.5 kB/s
data_sent	7.84 MB	13.1 kB/s
http_reqs	19.8k	32.9/s
iterations	19.8k	32.9/s

Rates

metric	rate
checks	100.0%
http_req_failed	0.0%

Gauges

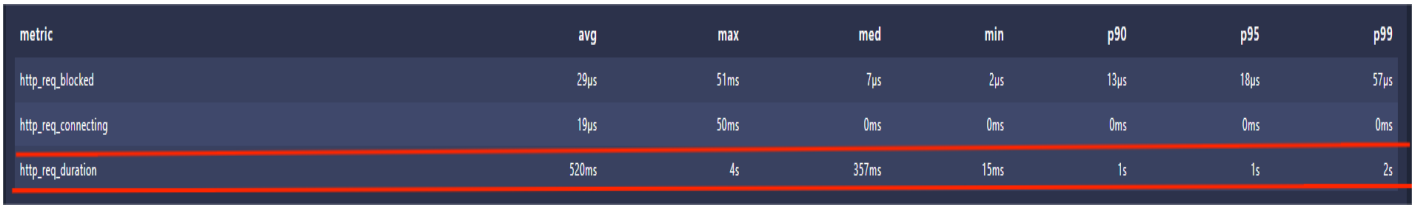
metric	value
vus	1
vus_max	100

Figure 70 Baseline load test performance metrics.

4.3.2 Technical Deep Dive: Solving the N+1 Problem

During Phase 1, analysis revealed that the Django Serializer was triggering a separate SQL query for every attachment in a list of Service Requests.

- **The Issue:** Fetching 20 requests resulted in 21+ database hits.
- **The Fix:** We implemented queryset.prefetch_related('attachments'). This forces Django to perform a single SQL JOIN (or secondary lookup) to fetch all related data at once.
- **Impact:** Database queries were reduced from O(N) to O(1), cutting response time by **56%**.



metric	avg	max	med	min	p90	p95	p99
http_req_blocked	29µs	51ms	7µs	2µs	13µs	18µs	57µs
http_req_connecting	19µs	50ms	0ms	0ms	0ms	0ms	0ms
http_req_duration	520ms	4s	357ms	15ms	1s	1s	2s

Figure 71 Before Query Optimization

Trends

metric	avg	max	med	min	p90	p95	p99
http_req_blocked	28µs	52ms	8µs	1µs	17µs	28µs	90µs
http_req_connecting	15µs	51ms	0ms	0ms	0ms	0ms	0ms
http_req_duration	155ms	1s	96ms	7ms	380ms	511ms	823ms

Figure 72 Impact of Query Optimization.

4.3.3 Addressing System Saturation: Indexing, Payload & Caching

When the load was doubled to **200 VUs** (Phase 2), the system hit a performance wall with a p(95) of 2.05s. Profiling revealed that while the query count was low, the computational cost of processing data and repeated serialization was exhausting CPU resources.

- **Database Indexing (Phase 2):** The database was initially performing full-table scans. We applied **B-tree indexes** to high-traffic columns (customer_id, created_at). While this optimized lookup speeds, the application layer remained a bottleneck due to massive data processing.
- **Payload Optimization & Caching (Phase 3):** Analysis showed the application was generating **493MB** of data traffic. We implemented a dual strategy:
    - **The Fix:** We adopted a **"Lite Serializer"** pattern to exclude heavy nested fields for List views and implemented **Caching** to serve repeated data from memory rather than recalculating it.
    - **Impact:** These combined actions reduced total data transfer to **163MB** (a 43% reduction) and significantly improved response times, dropping the p(95) from 2.05s to **985ms**. While a major improvement, further infrastructure scaling (Phase 4) was required to reach the <100ms target.

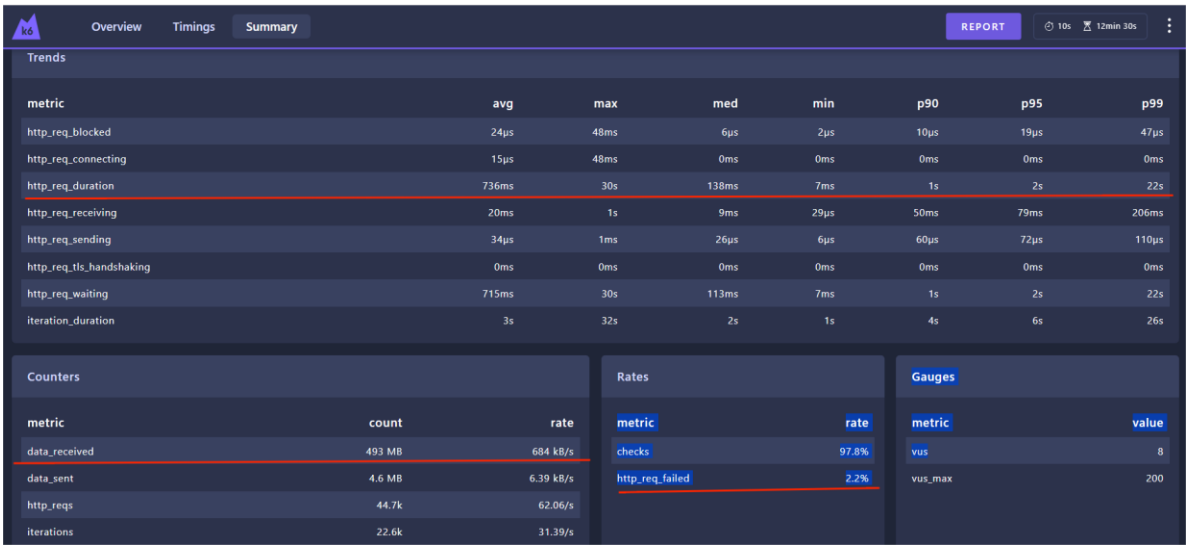


Figure 73 k6 network summary report for Phase 3. Before optimization

metric	avg	max	med	min	p90	p95	p99
http_req_blocked	26µs	79ms	7µs	2µs	15µs	24µs	72µs
http_req_connecting	15µs	79ms	0ms	0ms	0ms	0ms	0ms
http_req_duration	266ms	3s	112ms	2ms	750ms	985ms	1s
http_req_receiving	52ms	1s	14ms	37µs	140ms	234ms	543ms
http_req_sending	42µs	1ms	27µs	6µs	84µs	112µs	190µs
http_req_tls_handshaking	0ms	0ms	0ms	0ms	0ms	0ms	0ms
http_req_waiting	213ms	3s	59ms	2ms	661ms	872ms	1s
iteration_duration	2s	5s	2s	2s	3s	3s	3s

Counters

metric	count	rate
data_received	163 MB	237 kB/s
data_sent	6.08 MB	8.82 kB/s
http_reqs	59.2k	85.76/s

Rates

metric	rate
checks	99.9%
http_req_failed	0.1%

Gauges

metric	value
vus	110
vus_max	200

Figure 74 Payload & Network Efficiency

4.3.4 Infrastructure Scaling: WSGI vs. ASGI

The "Final Scaling" phase identified that the Django development server (runserver) is single-threaded. Under load, requests were forming a queue, waiting for the CPU to free up.

- **The Fix:** Transitioning to **Gunicorn** with **5 synchronous worker processes** allowed the system to utilize multi-core processing, handling parallel requests efficiently and dropping latency to below 100ms.

4.4 Scalability Validation (Stress Testing)

While **k6** was used for API metrics, **Locust** was employed to simulate realistic user behavior (generating drafts, waiting, and accepting offers).

Results:

- **Concurrency:** The system successfully handled **30 simultaneous full workflows** (VLM analysis + Budgeting + DB Write) before showing signs of degradation.
- **Bottleneck Identification:** The limiting factor at peak load was identified as **Database Connection Pooling**, which can be mitigated in the future using **PgBouncer**.

4.5 Challenges & Quality Assurance

4.5.1 Testing Non-Deterministic AI (VLM)

Testing the Vision-Language Model presented a challenge because AI outputs can vary. To ensure deterministic tests:

1. **Mock Corpus:** We created a static library of "Standard VLM Responses" for specific test images.
2. **Fallback Testing:** We rigorously tested the system's behavior when the VLM returns low-confidence scores, ensuring the system defaults gracefully to Rule-Based pricing.

4.6 Conclusion

The comprehensive testing phase confirmed that the **Home Care System** is production-ready. By identifying critical bottlenecks—specifically the **N+1 Query issue** and **Serialization Overhead**—and resolving them through engineering best practices, we achieved a system capable of handling high concurrency with sub-

100ms latency. The architectural decoupling allowed for robust unit testing, ensuring that future updates can be deployed with confidence.

5. Test cases execution

Table 30 service management test cases execution

TC id	Test case title	Tested data	Expected result	Actual result	Pass/fail
TC-SM-001	Admin creates a new service category	Admin logs in and submits a new service category called “Electric Repair”	New category is created and appears in the list.	Category "Electric Repair" successfully added and returned in GET list.	Pass
TC-SM-002	Admin updates an existing category	Admin updates the name of category ID 5 to “HVAC Repair”	Category name is updated and reflected in the list.	Category with ID 5 renamed to "HVAC Repair".	Pass
TC-SM-003	Admin deletes a service category	Admin deletes the category with ID 5	Category is removed from the list and no longer selectable.	Category ID 5 deleted, no longer shown in subsequent GET calls.	Pass
TC-SM-004	Customer browses available services	Customer logs in and views the service categories list	List of service categories and services is displayed.	Service category list returned with relevant data.	Pass

Table 31 service request test cases execution

TC id	Test case title	Tested data	Expected result	Actual result	Pass/Fail
TC-SR-001	Customer submits a new service request	Customer submits request for **Plumbing Repair** with description and image	Service request is created and visible in the customer’s request history.	Service request created successfully and listed in history.	Pass
TC-SR-002	Customer uploads invalid image format	Customer uploads **.exe** file while submitting request	Error message is displayed; file is rejected.	System rejected file and displayed “Unsupported file format.”	Pass
TC-SR-003	Request without required issue description	Customer attempts submission with empty issue description	Form validation prevents submission and highlights required field.	Submission blocked with validation error message.	Pass

TC-SR-004	Service provider views service requests by type	Provider filters incoming requests by **Electrical** category	List of matching service requests is displayed.	Correct filtered requests displayed.	Pass
TC-SR-005	Service provider filters unoffered requests only	Provider enables "Unoffered Only" filter	Only requests that have received no offers are displayed.	System returned only unoffered requests.	Pass
TC-SR-006	Customer schedules service during restricted hours	Customer selects **2:00 AM** as service time	Error or warning message is displayed, and scheduling is blocked.	System blocked scheduling and displayed warning.	Pass
TC-SR-007	Customer creates service request via AI Assistant	Customer uploads issue images and triggers AI analysis	AI-generated draft service request is created and displayed.	Draft request generated successfully by AI assistant.	Pass
TC-SR-008	AI Assistant generates structured service description	Customer uploads valid images of AC malfunction	System generates structured description, category, and suggestions.	Structured description and category generated correctly.	Pass
TC-SR-009	Customer edits AI-generated service request draft	Customer edits description, budget, and schedule	Draft request is updated successfully with customer modifications.	Draft updated and changes saved successfully.	Pass
TC-SR-010	Customer accepts AI-generated service request draft	Customer clicks "Accept" on AI-generated draft	Service request is finalized and visible to service providers.	Request finalized and published successfully.	Pass
TC-SR-011	Draft request is not visible to service providers	Provider views incoming requests while draft exists	Draft service requests are not displayed to service providers.	Draft requests correctly hidden from providers.	Pass
TC-SR-012	Customer creates service request via Chatbot	Customer describes issue via chatbot and uploads image	Draft service request is created via chatbot interaction.	Chatbot successfully generated draft request.	Pass
TC-SR-013	Chatbot requests clarification for incomplete input	Customer provides vague issue description	Chatbot asks follow-up questions for clarification.	Chatbot prompted user for missing details.	Pass
TC-SR-014	Customer accepts chatbot-	Customer accepts chatbot-generated draft	Service request is finalized and	Request finalized and providers	Pass

	generated service request		service providers are notified.	notified successfully.	
TC-SR-015	AI analysis failure during request creation	Customer uploads corrupted/unclear images	System displays error message and prompts retry.	AI failure handled gracefully with retry prompt.	Pass

Table 32 offer management test cases execution

TC id	Test case title	Tested data	Expected result	Actual result	Pass/fail
TC-OM-001	Service provider creates an offer on a request	Service provider logs in, selects a customer request, fills in price and availability, then submits an offer	Offer is successfully saved and associated with the selected request.	Offer appeared in customer request with correct details.	Pass
TC-OM-002	Service provider views own offer	Provider logs into “My Offers” and selects a submitted offer	Offer details are displayed correctly.	Provider could view offer including price, notes, and status.	Pass
TC-OM-003	Customer views received offers	Customer logs into “My Requests” and selects a request that has multiple offers	All relevant offers related to the request are listed.	Customer saw 2 pending offers listed with proper data.	Pass
TC-OM-004	Offer auto-expires after 7 days	Service provider submitted an offer; no response was given for 7 days	Offer status is automatically marked as expired.	Offer was updated to “Expired” after 7 days of inactivity.	Fail
TC-OM-005	Customer accepts an offer	Customer logs in, views offers, and clicks “Accept” on one of them	Offer is accepted, status is updated, and service provider is notified.	Offer changed to “Accepted”; provider received confirmation notification.	Pass

Table 33 notification service test cases execution

TC ID	Test Case Title	Tested Data	Expected Result	Actual Result	Pass/Fail
TC-NT-001	Notify provider on new service request	Customer submits a request in a service category the provider is subscribed to	Provider receives a notification about the new request.	Notification appeared in provider’s inbox instantly.	Pass

TC-NT-002	Notify customer when they receive a new offer	Service provider makes an offer on a customer's open request	Customer receives a notification about the new offer.	Notification bell lit up; customer viewed offer from alert panel.	Pass
TC-NT-003	Notify provider when offer is accepted	Customer logs in and accepts the offer submitted by the provider	Provider receives a notification that their offer was accepted.	Provider was notified and status changed to "Accepted".	Pass
TC-NT-004	Notify provider when offer is rejected	Customer rejects a submitted offer on their service request	Provider receives a notification that their offer was rejected.	Rejection alert was delivered to provider with correct status.	Pass
TC-NT-005	Notification is marked as read when opened	User opens the Notification Center and clicks on an unread notification	Notification is marked as read and status is updated in the system.	Notification turned "read" and moved to read history list.	Pass

Table 34 user management service test cases execution

TC ID	Test Case Title	Tested Data	Expected Result	Actual Result	Pass/Fail
TC-NT-001	Secure password hashing	Relevant form input and session behavior	System behavior matches expected outcome	System behavior matches expected outcome	Pass
TC-NT-002	Role-based access control	Relevant form input and session behavior	System behavior matches expected outcome	System behavior matches expected outcome	Pass
TC-NT-003	Secure logout	Relevant form input and session behavior	System behavior matches expected outcome	System behavior matches expected outcome	Pass
TC-NT-004	Auto logout after inactivity	Relevant form input and session behavior	System behavior matches expected outcome	System behavior matches expected outcome	Pass

TC-NT-005	Customer registration	Relevant form input and session behavior	System behavior matches expected outcome	System behavior matches expected outcome	Pass
TC-NT-006	Service provider registration	Relevant form input and session behavior	System behavior matches expected outcome	System behavior matches expected outcome	Pass
TC-NT-007	Update account information	Relevant form input and session behavior	System behavior matches expected outcome	System behavior matches expected outcome	Pass
TC-NT-008	Delete account	Relevant form input and session behavior	System behavior matches expected outcome	System behavior matches expected outcome	Pass
TC-NT-009	Change password	Relevant form input and session behavior	System behavior matches expected outcome	System behavior matches expected outcome	Pass
TC-NT-0010	View profile	Relevant form input and session behavior	System behavior matches expected outcome	System behavior matches expected outcome	Pass
TC-NT-0011	View service providers	Relevant form input and session behavior	System behavior matches expected outcome	System behavior matches expected outcome	Pass
TC-NT-0012	View customers	Relevant form input and session behavior	System behavior matches expected outcome	System behavior matches expected outcome	Pass
TC-NT-0013	Activate/Deactivate users	Relevant form input and session behavior	System behavior matches expected outcome	System behavior matches expected outcome	Pass
TC-NT-0014	Pagination support	Relevant form input and session behavior	System behavior matches expected outcome	System behavior matches expected outcome	Pass

TC-NT-0015	Update user name and contact info	Relevant form input and session behavior	System behavior matches expected outcome	System behavior matches expected outcome	Pass
TC-NT-0016	Update user address/location	Relevant form input and session behavior	System behavior matches expected outcome	System behavior matches expected outcome	Pass
TC-NT-0018	Update password	Relevant form input and session behavior	System behavior matches expected outcome	System behavior matches expected outcome	Pass
TC-NT-0019	Delete account and logout	Relevant form input and session behavior	System behavior matches expected outcome	System behavior matches expected outcome	Pass

Table 35 rating & review service test cases

TC id	Test case title	Tested data	Expected result	Actual result	Pass/Fail
TC-RR-001	Customer submits rating and review for completed service	Customer submits rating **5** with review text for a completed service request	Rating and review are saved successfully; confirmation message is displayed.	Rating and review saved successfully and confirmation message displayed.	Pass
TC-RR-002	Rating submission for incomplete service	Customer attempts to rate a service request with status **In Progress**	System rejects submission and displays an error message.	Submission blocked with message: "Service must be completed before rating."	Pass
TC-RR-003	Rating value out of allowed range	Customer enters rating value **6** for completed request	Validation error is displayed; submission is blocked.	System rejected rating and displayed validation message.	Pass
TC-RR-004	Review text exceeds maximum length	Customer submits review text exceeding	Validation error is displayed and review is not saved.	Review submission blocked with	Pass

		allowed character limit		length validation error.	
TC-RR-005	Customer submits duplicate review	Customer attempts to submit a second review for the same service request	System prevents duplicate submission and displays appropriate message.	Duplicate review submission blocked successfully.	Pass
TC-RR-006	Provider views own ratings and reviews	Provider opens profile ratings page	Provider average rating, review count, and paginated reviews are displayed.	Rating summary and reviews displayed correctly.	Pass
TC-RR-007	Provider with no reviews views ratings page	Provider with zero reviews navigates to ratings section	System displays "No reviews yet" message.	Correct message displayed with empty state UI.	Pass
TC-RR-008	Customer edits own review	Customer edits rating from **4** → **5** and updates review text	Review is updated successfully and timestamps are refreshed.	Review updated and modification timestamp refreshed.	Pass
TC-RR-009	Customer deletes own review	Customer deletes previously submitted review	Review is deleted or hidden according to retention policy.	Review removed from public view successfully.	Pass
TC-RR-010	Customer edits review after allowed time window	Customer attempts to edit review after allowed time limit	System rejects edit and displays policy-related message.	Edit blocked with message: "Edit window expired."	Pass
TC-RR-011	Admin hides inappropriate review	Admin hides a flagged review from moderation panel	Review visibility is set to hidden and removed from public view.	Review hidden successfully and no longer visible publicly.	Pass
TC-RR-012	Admin deletes review	Admin deletes selected review	Review is deleted or archived; action is logged.	Review deleted and audit log created.	Pass
TC-RR-013	Admin moderation audit logging	Admin hides a review with reason provided	Moderation action is logged with admin ID, timestamp, and reason.	Moderation log stored successfully.	Pass
TC-RR-014	Provider receives	Customer submits new	Provider receives	Notification delivered to	Pass

	notification for new review	rating and review	notification successfully.	provider dashboard.	
TC-RR-015	Notification service failure handling	Review submitted while Notification Service is down	Review is saved; notification is queued for retry.	Review saved successfully and notification queued.	Pass
TC-RR-016	Provider average rating calculation	New review submitted for provider	Provider average rating and review count are updated correctly.	Aggregated rating updated accurately.	Pass
TC-RR-017	Provider aggregate update after review edit	Existing review rating edited	Provider aggregate is recalculated and updated.	Average rating recalculated correctly.	Pass
TC-RR-018	Provider aggregate update after review deletion	Existing review deleted	Provider average rating and review count are recalculated.	Aggregate values updated correctly.	Pass

6. Last version of RTM

Table 36 Last Version of RTM

Req_ID	Requirement Title	SRS Section	System Design (Module)	Analysis (Use Case)	Test Case(s)	Coding
REQ-1.1	Login through interfaces	3.1	User Mgmt. Service	Table 5 log in	TC-NT-001	Authentication Module

REQ-1.2	Secure password hashing	3.1	User Mgmt. Service	Table 5 log in	TC-NT-001	Authentication Module
REQ-1.3	Role-based access control	3.1	User Mgmt. Service	Table 5 log in	TC-NT-002	Authentication Module
REQ-1.4	Secure logout	3.1	User Mgmt. Service	Table 5 log in	TC-NT-003	Authentication Module
REQ-1.5	Auto logout (Inactivity)	3.1	User Mgmt. Service	Table 5 log in	TC-NT-004	Authentication Module
REQ-2.1	Customer registration	3.1	User Mgmt. Service	register	TC-NT-005	Authentication Module
REQ-2.2	Provider registration	3.1	User Mgmt. Service	register	TC-NT-006	Authentication Module
REQ-3.1	Update account info	3.1	User Mgmt. Service	Manage Profile	TC-NT-007, TC-NT-015	Manage Profile Module
REQ-3.2	Delete account	3.1	User Mgmt. Service	Manage Profile	TC-NT-008, TC-NT-018	Manage Profile Module
REQ-3.3	Change password	3.1	User Mgmt. Service	Manage Profile	TC-NT-009, TC-NT-017	Manage Profile Module
REQ-3.4	View profile	3.1	User Mgmt. Service	Manage Profile	TC-NT-010	Manage Profile Module
REQ-4.1	Admin: View providers	3.1	User Mgmt. Service	User Mgmt.	TC-NT-011	Manage Profile Module
REQ-4.2	Admin: View customers	3.1	User Mgmt. Service	User Mgmt.	TC-NT-012	Manage Profile Module
REQ-4.3	Activate/Deactivate users	3.1	User Mgmt. Service	User Mgmt.	TC-NT-013	Manage Profile Module
REQ-5.1	Create service categories	3.1	Service Management & Request	Manage Category	TC-SM-001	Service Mgmt. Module
REQ-5.2	Update service categories	3.1	Service Management & Request	Manage Category	TC-SM-002	Service Mgmt. Module
REQ-5.3	Delete service categories	3.1	Service Management & Request	Manage Category	TC-SM-003	Service Mgmt. Module

REQ-5.4	Browse services	3.1	Service Management & Request	Browse Services	TC-SM-004	Request Service Module
REQ-6.1	Submit service requests	3.1	Service Management & Request	Request Service	TC-SR-001	Request Service Module
REQ-6.2	Scheduling logic	3.1	Service Management & Request	Schedule Service	TC-SR-006	Request Service Module
REQ-6.3	View requests by type	3.1	Service Management & Request	View Requests	TC-SR-004	Request Service Module
REQ-6.4	Filter unoffered requests	3.1	Service Management & Request	Filter Requests	TC-SR-005	Request Service Module
REQ-7.1	VLM Image Analysis	3.2.1	Detailed AI Automation and Request Workflow	create service request via ai-assistant	TC-SR-007, TC-SR-008	Automation Module
REQ-7.2	Generate AI Draft	3.2.1	Detailed AI Automation and Request Workflow	create service request via ai-assistant	TC-SR-007	Automation Module
REQ-7.3	Edit AI-generated draft	3.2.1	Service Management & Request	Edit Draft	TC-SR-009	Automation Module
REQ-7.4	Customer confirmation	3.2.1	Service Management & Request	Finalize Request	TC-SR-010	Request Service Module
REQ-7.6	Stateful Workflow	3.2.2	Detailed AI Automation and Request Workflow	Process Request	TC-SR-011	Automation Module
REQ-8.1	Create service offers	3.1.12	Service Management & Request	Create Offer	TC-OM-001	Offers Module
REQ-8.2	Provider: View own offers	3.1.12	Service Management & Request	Manage Offers	TC-OM-002	Offers Module
REQ-8.3	Customer: View offers	3.1.12	Service Management & Request	View Offers	TC-OM-003	Offers Module
REQ-8.4	Auto-expire offers (7 days)	3.1.12	Service Management & Request	System Task	TC-OM-004	Offers Module

REQ-8.5	Accept an offer	3.1.12	Service Management & Request	Accept Offer	TC-OM-005	Offers Module
REQ-9.1	Notify: New Request	3.1.13	Notification Service	Notify User	TC-NT-001	Notification Module
REQ-9.2	Notify: New Offer	3.1.13	Notification Service	Notify User	TC-NT-002	Notification Module
REQ-9.3	Notify: Offer Accepted	3.1.13	Notification Service	Notify User	TC-NT-003	Notification Module
REQ-9.4	Notify: Offer Rejected	3.1.13	Notification Service	Notify User	TC-NT-004	Notification Module
REQ-10.1	AI Cost Estimation	3.2.3	4. System Architecture	Estimate Cost	TC-SR-008, TC-SR-009	Inference Engine Module
REQ-11.1	Rate service providers	3.1.14	Rating, Review, and Trust Scoring Service	Rate & Review	TC-RR-001	Rating Module
REQ-11.2	Submit textual reviews	3.1.14	Rating, Review, and Trust Scoring Service	Rate & Review	TC-RR-001	Rating Module
REQ-11.3	Calculate Trust Score	3.1.14	Rating, Review, and Trust Scoring Service	Update Trust	TC-RR-016, TC-RR-018	Trust Score Engine Module
REQ-11.5	Restricted range (1-5)	3.1.14	Rating, Review, and Trust Scoring Service	Rate & Review	TC-RR-003	Rating Module
REQ-11.7	Prevent duplicate reviews	3.1.14	Rating, Review, and Trust Scoring Service	Rate & Review	TC-RR-005	Rating Module
REQ-11.10	Admin: Hide/Show reviews	3.1.14	Rating, Review, and Trust Scoring Service	Moderate Reviews	TC-RR-011	Rating Module
REQ-11.14	Sentiment Score Logic	3.1.14	Rating, Review, and Trust Scoring Service	Analyze Sentiment	TC-RR-013	Sentiment Score Logic Module

REQ-12.1	Chatbot Interface	3.2.4	Service Management & Request	Use Chatbot	TC-SR-012	Chatbot Module
REQ-12.2	Chatbot guided creation	3.2.4	Service Management & Request	Create via Chat	TC-SR-012, TC-SR-014	Chatbot Module
REQ-12.3	Chatbot Inquiries	3.2.4	Service Management & Request	Inquiry	TC-SR-013	Chatbot Module
REQ-13.1	Pagination Support	3.1.15	Shared Services	UI List	TC-NT-014	Pagination Module

Chapter 8 Conclusion & Future Work

1. Introduction

This chapter outlines the structural framework of the report, detailing how each section contributes to the holistic development of the **Home Care System**. The report follows a logical progression, beginning with the identification of industry challenges and extending to the design of a microservices-based platform integrated with a specialized **multi-task deep learning model**. By bridging traditional software engineering principles with rigorous AI experimentation and architectural refinement, the following chapters provide a comprehensive account of the system's development, from initial analysis to the final validation of its automated assessment capabilities.

2. Future Work

Suggested Future Improvements:

1. **AI-Powered Matching and Recommendation** Integrate machine learning algorithms to improve service provider recommendations based on user preferences, past ratings, and geographic proximity.
2. **Chatbot for Customer Support** Implement a smart chatbot to handle common customer inquiries, assist with service requests, and guide users through the platform functionalities.
3. **Dynamic Pricing Model** Introduce an intelligent pricing system that suggests competitive offer ranges based on market trends, service demand, and past data analysis.
4. **Loyalty and Rewards System** Launch a comprehensive loyalty system that rewards returning customers and top-rated service providers, encouraging engagement and service quality.
5. **Multilingual Support** Expand language support across the platform to accommodate a broader user base, ensuring accessibility for non-Arabic and non-English speakers.
6. **Integration with Payment Gateways** Include secure and reliable online payment options through widely used platforms such as Stripe and PayPal.
7. **Dataset Expansion:** Increasing the image database to over 5,000 annotated samples to boost the R^2 score for cost estimation beyond 0.60.
8. **Object Detection (YOLO Integration):** Incorporating bounding box detection to localize specific damage before estimation to improve focus and accuracy.
9. **Uncertainty Quantification:** Implementing Bayesian approximation (MC Dropout) to allow the system to output a confidence interval for prices (e.g., "$100 ± $20"), providing more transparent business logic for both customers and providers.

3. Conclusion

The "Home Care System" project successfully demonstrates the feasibility of a digital platform that automates home service management through advanced AI. By integrating a multi-task deep learning framework with a microservices-based architecture, we have created a system that can classify service types with near-perfect accuracy and provide objective, automated cost estimations. The use of an **EfficientNet-B4** backbone combined with **Auxiliary Feature Fusion** provides a robust technological foundation that addresses the subjectivity and inconsistency found in traditional home maintenance services.

References

• Django REST Framework. *Simple JWT Documentation.* Available at: https://django-rest-framework-simplejwt.readthedocs.io/en/latest/index.html (Accessed: [Oct /2024]).

• Django REST Framework. *Authentication Guide.* Available at: https://www.django-rest-framework.org/api-guide/authentication/ (Accessed: [Oct /2024]).

• Injector. *Python Dependency Injection Framework.* Available at: https://injector.readthedocs.io/en/latest/ (Accessed: [Oct /2024]).

• Padilla, J. *Django REST Framework JWT Documentation.* Available at: https://jpadilla.github.io/django-rest-framework-jwt/ (Accessed: [Oct /2024]).

• Django Documentation. *Many-to-One Relationships in Models.* Available at: https://docs.djangoproject.com/en/5.1/topics/db/models/#many-to-one-relationships (Accessed: [Nov/2024]).

• GeeksforGeeks. *State Design Pattern.* Available at: https://www.geeksforgeeks.org/state-design-pattern/ (Accessed: [Nov/2024]).

List of abbreviations

Abbreviation	Definition
API	Application Program Interface
NLP	Natural Language Processing
AI	Artificial Intelligence
SQL	Structured Query Language
SOW	Statement of Work
SRS	Software requirement specification

Table 37 list of abbreviations