Bahria University, Islamabad

Home Services

Khawaja Talal 01-135221-022

Raja Bahram Shafiq 01-135221-045

Bachelors of Science in Information Technology

Supervisor: Jawwad Ijaz

Department of Computer Science
Bahria University, Islamabad

Certificate

We accept the work contained in the report titled **"Home Services App"**, written by **Mr. Khawaja Talal** and **Mr. Raja Bahram Shafiq** as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Computer Science.

Approved by . . . :

Supervisor:

_____

Jawwad Ijaz

Internal Examiner:

_____

External Examiner:

_____

Project Coordinator:

_____

Head of the Department:

_____

Abstract

The Home Services App is a mobile-based platform designed to connect customers with verified professionals for a wide range of household services, including plumbing, electrical work, and general maintenance. The app aims to streamline the process of finding and hiring reliable service providers by offering features such as AI-powered chat assistance for problem diagnosis, real-time bidding for transparent pricing, and location tracking for efficient on-site services.

Customers can post job requests, compare bids, communicate directly with professionals, and track service appointments in real time. Professionals can manage their profiles, submit bids, and respond to job offers, while an integrated admin panel enables system-wide oversight and professional verification. The application leverages Flutter for cross-platform development, Supabase Supabase Database for real-time database management, and Google Maps API for navigation and live tracking.

This project provides a modern solution to the challenges of inconsistent service quality, price ambiguity, and lack of trust in the home services sector. By offering verified professionals, competitive bidding, AI-driven insights, and a seamless user experience, the Home Services App serves as a reliable, efficient, and transparent marketplace for both customers and service providers.

Acknowledgments

In the name of the most merciful and benevolent Allah. We are incredibly appreciative to the One who made us and bestowed upon us a life of privilege. We want to express our gratitude to our parents, who have always been a source of courage and encouragement. We are grateful to our parents for teaching us that perseverance, dedication, and pursuing your goals with unadulterated intention can lead to success.

Khawaja Talal & Raja Bahram Shafiq
Islamabad, Pakistan

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

The Home Services is a cutting-edge mobile app developed using Flutter [1], designed to connect customers with verified home-service professionals such as plumbers, electricians, and other service providers. The app aims to simplify the process of finding reliable professionals by offering a transparent and efficient platform. With features like AI-powered chat assistance, real-time bidding, and Google Maps integration [2], the app ensures a seamless experience for both customers and professionals.

The platform empowers customers to post job requests, receive competitive bids, and track professionals in real time. Professionals, on the other hand, can showcase their expertise, bid for jobs, and provide on-site services. By integrating advanced technologies, the Home Services App addresses the challenges of the traditional service industry and sets a new standard for convenience and reliability.

1.2 Objective

The primary objective of this project is to develop a dependable and user-friendly platform that:

Enables real-time home service bidding, guaranteeing competitive pricing. improves decision-making by offering AI-guided price estimation and problem diagnosis. Connects clients with certified experts to guarantee transparency. Provides real-time location tracking and on-demand on-site support. Uses in-app messaging to improve customer-professional communication. By developing a dependable, effective, and transparent solution that benefits both clients and service providers, this project seeks to completely transform the home services sector.

1.3 Problem Description

The traditional process of hiring home service professionals is fraught with challenges, including:

The conventional approach of recruiting professionals of home services is characterized by difficulties, including:

Unauthenticated Professionals: Customers usually have difficulties in locating reliable and professionals who are under-skilled, which results in the low quality of service.

Inequality of Pricing: Cus- does not have a standardized pricing. tomers to plan their budgets well, which mostly leads to excessive or unacknowledged expenses. Restricted Availability: The customers have challenges in accessing the close-by service. providers, in particular, in emergencies. Communication Barriers: It has poor communication channels among the customers. and professionals result in miscommunication and time wastage. Absence of Transparency: There is less visibility on the customer on the bidding process, and it is difficult to compare and make informed decisions.

These issues often result in customer dissatisfaction and inefficiencies in the service industry. The Home Services App addresses these challenges by providing:

Verified professionals have profile badges to ensure trust and reliability. AI-powered assistance helps diagnose issues and suggests price ranges based on the market. There is a real-time bidding system that encourages competitive pricing and transparency. Google Maps integration allows for real-time tracking and on-site visit services. In-app messaging enables seamless communication between customers and professionals.

1.4 Methodology

The construction of the Home Services App has a clear and detailed strategy, as it is based on the latest technologies and practices. The major ones are: Requirement Analysis: The analysis of the requirements of customers and professionals to define the features of the app and its functionality. Design: The user-friendly interface and easy workflows should be designed in a way to provide the user with a smooth experience. Development: Flutter as a cross-platform development tool, Supabase Supabase Database as a real-time database manager and AI/ML tools as a chat-based problem analyzer. Testing: It involves carrying out rigorous testing to fix bugs, security, and performance under different conditions. Deployment: The app should be deployed on Android and iOS platforms and monitored and updated on a regular basis. The application will have three main elements: User Panel: It will enable the customers to create job requests, to browse and compare bids, request on-site services and appointments. Professional Panel: This helps professionals to maintain their profiles, bid, as well as respond to the job requests and monitor their earnings. Admin Panel: Gives system administrators the means to control user accounts, track jobs and avert the successful running of the platform.

1.5 Project Scope

The scope of this project encompasses the development of a comprehensive, AI-assisted home services platform designed to connect customers with skilled service providers through a transparent, efficient, and user-friendly mobile application. It includes multi-role authentication, real-time job posting and bidding, secure payment processing, AI-guided diagnostics, location-based service tracking, and a complete communication system between customers and professionals. The project further includes an administrative dashboard for system oversight, detailed reporting, and analytics. The scope covers both front-end and back-end development, third-party integrations, database design, real-time services, and the deployment of the application across mobile platforms.

1.5.1 System Features

The **User Management System** provides multi-role authentication for customers, providers, and administrators using Supabase Auth with email and password login capabilities. It includes full profile management, allowing users to upload avatars, add bios, and update contact information. Role-based access control ensures that each user type can access only the features relevant to their role. The system also supports secure account verification through email confirmation and phone validation.

The **Job Management** module is divided into customer and provider functionalities. Customers can post jobs by adding a title, description, budget, and location, and can select categories such as plumbing, electrical, or cleaning. They can also attach photos or videos for clarity. Jobs support fixed or hourly budgets and progress through statuses including Open, Bidding, Assigned, In Progress, and Completed. After completion, customers may leave reviews and ratings. Providers, on the other hand, can browse available jobs using filters like category, location, and budget. They can place bids, accept job offers, update job statuses during progress, and track their earnings through a dedicated dashboard.

The **Bidding System** enables real-time bid submissions, allowing providers to compete for open jobs. Customers receive bid notifications and can compare bids along with provider ratings before selecting a winner. Once a bid is accepted, both the winning and unsuccessful bidders are automatically notified. Bidding windows are time-limited to ensure timely responses.

The platform includes **Payment Integration** through Stripe for secure card payments using Payment Intents, card validation, and webhook-based status updates. It supports test mode for development, multi-currency processing, and both full and partial refunds. In addition to online payments, the system supports cash payments, enabling users to record transactions manually, generate digital receipts, and track payment statuses such as Pending, Paid, and Completed. All cash transactions remain visible to administrators for oversight.

The **Messaging System** allows seamless communication between customers and providers through in-app chat, with each conversation linked to a specific job. It supports real-time messaging, media sharing, read receipts, and persistent chat history to maintain communication records.

The **Notification System** delivers timely alerts through push notifications powered by Supabase Cloud Messaging and an in-app notification center. Users receive notifications for new bids, job updates, payments, and messages, and can customize their notification preferences. Unread alerts are indicated through badge counts.

The **Admin Dashboard** provides system-wide management tools. Administrators can view and manage all users, including actions such as banning, suspending, verifying accounts, and assigning roles. They can also oversee all jobs, enforce status changes, and access analytics such as completion rates and average budgets. Payment management tools allow administrators to handle Stripe and cash transactions, process refunds, track revenue, and export reports in CSV or PDF format. Additional analytics offer insights into revenue trends, user growth, and job completion metrics.

The **Review and Rating System** enables customers to leave 5-star ratings and written feedback after job completion, with administrators having the option to moderate and control the visibility of reviews.

The **Search and Filter** module allows users to find jobs using keywords, categories,

location filters, budget ranges, and sorting options such as date, price, or rating.

Finally, the **Media Management System** supports image uploads for job photos and profile pictures using Supabase Storage. It ensures secure cloud storage with optimized compression, supports multiple formats such as JPEG, PNG, and PDF, and provides a gallery view for browsing job-related media.

1.6 Feasibility Study

1.6.1 Technical Feasibility

The project is technically feasible because it leverages modern and readily available technologies. Flutter ensures a smooth, efficient, and reliable user experience across both iOS and Android platforms. Supabase Database provides a scalable, real-time backend solution capable of handling authentication, storage, and data operations. Google Maps API supports accurate location tracking and navigation features essential for service-based applications. Additionally, AI and machine learning tools enable intelligent features such as chat-based problem analysis and automated price estimation, enhancing user convenience and decision-making.

1.6.2 Risks Involved

Several risks are involved in the development and deployment of the system. These include ensuring strong data security to protect user information and payment transactions, maintaining the accuracy and reliability of AI-based problem diagnosis and cost predictions, and guaranteeing the stability of real-time tracking and notification services.

1.6.3 Resource Requirements

The resource requirements for building the system include modern development tools such as Flutter, Supabase Database, Google Maps API, and AI/ML frameworks. The project also requires a team skilled in mobile app development, database management, and AI/ML integration. Basic hardware, including laptops and multiple mobile devices, is necessary for development, debugging, and testing across various environments.

1.6.4 Solution Application Areas

The Home Services App has a wide range of application areas within the service industry. It is designed for residential customers who need trustworthy professionals for tasks such as plumbing, electrical work, carpentry, and general home maintenance. It also caters to the commercial sector, serving businesses that require regular maintenance or repair services for their facilities. Additionally, the app benefits service providers by offering a platform where skilled professionals can showcase their expertise, access job opportunities, and expand their business.

1.7 Methodology (Detailed)

The development uses Flutter, Supabase for real-time database management, and Google Maps API. AI-based diagnosis relies on foundational NLP and machine learning princi-

ples.The methodology is divided into several key phases:

1.7.1 Requirement Analysis

Understanding user needs through surveys and interviews. Defining key features like AI-powered chat assistance, real-time bidding, and Google Maps integration. Conducting competitor analysis to find gaps.

1.7.2 System Design

Creating UI wireframes and prototypes. Designing the database schema with Supabase. Defining a three-tier architecture, which includes Presentation, Business Logic, and Data.

1.7.3 Development

Implementing the app using Flutter and Supabase. Integrating AI/ML models and Google Maps API.

1.7.4 Testing

Unit, integration, performance, and user acceptance testing.

1.8 Tools and Technology

The project's **frontend development** is built using the Flutter SDK (version 3.7+), supported by the Dart programming language and Material Design 3 for modern UI styling. State management is handled through Riverpod and Equatable, ensuring predictable and scalable application state flow. Navigation across the application is managed using GoRouter. The UI incorporates various styling and animation packages, including Cupertino Icons, Flutter SVG, Google Fonts, Loading Animation Widget, Flutter Spinkit, Shimmer, and Lottie, which collectively enhance the visual appeal and user experience.

On the **backend and database** side, the system relies on Supabase as a cloud-based backend-as-a-service solution. This includes Supabase Flutter SDK (version 2.3.4), PostgreSQL as the primary database, and features such as Supabase Auth, Supabase Storage, Supabase Realtime, and Edge Functions built with Deno. Database operations are secured through Row Level Security (RLS), and management is handled via pgAdmin or the Supabase Dashboard.

The **payment integration** is powered by Stripe, using Flutter Stripe (version 11.5.0) to support features such as the Stripe Payment Sheet, web payments, webhooks, and centralized monitoring through the Stripe Dashboard. For **maps and location services**, the system uses Google Maps Flutter, the web variant of Google Maps Flutter, and the Google Maps API. Location accuracy is enhanced through packages like Geolocator, Geocoding, and Permission Handler.

To support **HTTP and API communication**, the application uses both the HTTP and Dio packages, enabling REST API calls and WebSocket-based real-time communication. **Forms and validation** are implemented using Flutter Form Builder and the Form

Builder Validators package. **Image and file handling** features are enabled through Image Picker and Cached Network Image for image processing, while File Picker and Path Provider support file management on different platforms.

For **storage and persistence**, the application uses Shared Preferences for local data storage and Supabase Storage for cloud-based file handling. Additional utilities such as UUID, Timeago, Intl, and URL Launcher provide essential helper functionalities across the app. The **notification system** combines Flutter Local Notifications, the Timezone package, and Supabase Realtime triggers to deliver scheduled and real-time alerts to users.

The development process is supported by a robust set of **development tools**, including Visual Studio Code with Flutter and Dart extensions, GitLens, Error Lens, and Thunder Client. Version control is managed using Git and GitHub, while code quality is enforced through Flutter Lints, analysis options, and the Dart Analyzer. **Testing** is carried out using Flutter Test for widget testing, Mocktail for mocking, and integration tests, with Stripe's Test Mode enabling safe payment testing.

For **build and deployment**, Android builds rely on Android Studio, Gradle, Kotlin, the Android SDK, and deployment through the Google Play Console. iOS builds utilize Xcode, CocoaPods, Swift, TestFlight, and App Store Connect. Web deployment uses Flutter Web, Chrome DevTools, Progressive Web App (PWA) standards, and manifest configuration. Desktop builds are supported through Windows SDK, macOS SDK, and Linux GTK3.

The project also incorporates **DevOps and CI/CD** pipelines through GitHub Actions and Codemagic, with environment variables managed via .env files and the Supabase CLI. **Design and assets** are created using Figma and Canva, with assets organized under images, icons, and animations directories. Platform-specific tools include the Android Emulator, iOS Simulator, Instruments, App Transport Security, Flutter DevTools, Lighthouse, and Chrome Testing.

External **third-party services** supporting the system include the Stripe account, a Supabase project, and Google Cloud Platform for Maps API usage. **Communication and collaboration** across the team are handled using Git, GitHub Issues, Markdown documentation, and platforms such as WhatsApp, Discord, or Slack. The development environment spans both Windows and macOS operating systems, with command-line tools like PowerShell, Flutter CLI, Dart CLI, and Git Bash used throughout the development lifecycle.

1.9 Expertise of the Team Members

Member 1: Full-Stack Flutter Developer served as the primary frontend developer for the project, bringing a strong foundation in Flutter and Dart. Their expertise includes state management using Riverpod, building responsive user interfaces with Material Design, implementing multi-screen navigation with GoRouter, integrating APIs with the Supabase backend, and performing basic Stripe payment integration following documentation guidelines. Git was used for version control and collaborative development. Member 1's main responsibilities included frontend development (approximately 70% of the work), implementing UI/UX designs, setting up state management, creating navigation flows, developing the real-time chat interface, and designing job and bid management screens. They also handled testing and debugging to ensure smooth app performance.

Through this role, Member 1 mastered clean architecture patterns, Provider/Riverpod patterns, asynchronous programming and Streams, cross-platform development considerations, and payment SDK integration.

Member 2: Backend/Database Developer acted as the backend and database specialist, focusing on Supabase project management and PostgreSQL database design and optimization. Their expertise includes implementing Supabase Auth, managing Row Level Security (RLS) policies, configuring file storage and upload policies, and setting up real-time database subscriptions. Git was also used for version control and collaboration. Member 2's responsibilities accounted for approximately 30% of the project workload and included database schema design, RLS policy implementation, storage bucket configuration, real-time subscription setup, development of edge functions if needed, API endpoint testing, and supporting frontend integration. Through this role, Member 2 gained in-depth knowledge of the Supabase ecosystem, database security policies, real-time database concepts, file storage management, and SQL optimization techniques.

1.10 Milestones

The Home Services App development is broken down into key milestones to ensure steady progress and timely completion. Each milestone marks an important stage in the project lifecycle.

Table 1.1: Project Timeline

Task	**Start Date**	**End Date**	**Duration (Days)**
Project Proposal	9th Feb. 2025	14th Feb. 2025	6
Project Defence	24th Feb. 2025	29th Feb. 2025	4
Requirement Gathering	1st Mar. 2025	5th Mar. 2025	5
Front-end Development	7th Mar. 2025	23rd Mar. 2025	17
Reviewing	24th May. 2025	6th June. 2025	14
Back-end Development	8th June. 2025	28th June. 2025	21
Integration	1st July. 2025	10th July. 2025	10
Final Testing	12th July. 2025	13th July. 2025	2
Final Submission	24th Nov. 2025	28th Nov. 2025	5

Chapter 2

Literature Review

The home services industry has seen significant advancements with the integration of technology, yet challenges such as lack of transparency, inconsistent pricing, and unverified professionals persist. This literature review explores existing platforms, their limitations, and how the proposed Home Services App addresses these gaps.

2.1 Overview of the Home Services Industry

The home services industry encompasses a wide range of services, including plumbing, electrical work, carpentry, general maintenance, and more. With the rise of digital applications, customers have shifted from traditional methods of searching for professionals to using online platforms and mobile applications.

Despite technological improvements, existing platforms still fail to address critical issues such as:

- Lack of verified professionals, leading to trust and quality concerns.
- Inconsistent pricing models that create confusion for customers.
- Limited communication channels between customers and service providers.
- Absence of real-time tracking and bidding systems.

The proposed Home Services App addresses these challenges by integrating essential features such as AI-powered chat assistance, real-time bidding, and Google Maps–based navigation.

2.2 Related Work

This section reviews the work of existing online home service platforms and examines their features, limitations, and how they compare with the proposed Home Services App.

2.2.1 UrbanClap (Now Urban Company)

Urban Company (formerly UrbanClap) [7] is a major Indian multinational home services provider founded in 2014, operating as a two-sided marketplace that connects customers with verified professionals. The platform offers a wide and expanding range of at-home

services, including beauty and wellness (salon, spa, massage), home repairs and maintenance (electrician, plumbing, AC repair), and cleaning and pest control.

It has adopted a "full-stack" business model, meaning it goes beyond being a simple directory by actively managing the supply side through training, certification, providing quality kits, and micro-financing to its service partners, ensuring quality control and a standardized customer experience. This model, which generates revenue primarily through commissions on services and lead fees for customized jobs, has driven its expansion into international markets, including the UAE, Singapore, and Australia.

Features

- Verified service professionals.
- Fixed pricing for selected services.
- In-app booking and payment.

Limitations

- Limited flexibility due to fixed pricing.
- No real-time bidding option.
- No AI-powered diagnosis or price estimation.

Comparison with Home Services App

The Home Services App offers AI-based assistance and real-time bidding, enabling customers to compare multiple professional offers, improving pricing transparency.

2.2.2 TaskRabbit

TaskRabbit [8] connects customers with freelancers for home repairs, cleaning, and various physical tasks.

Features

- Broad task categories.
- Hourly pricing model.
- Customer reviews and ratings.

Limitations

- Professionals are not verified thoroughly.
- No real-time tracking.
- No AI-based support.

Comparison with Home Services App

Unlike TaskRabbit, the Home Services App focuses entirely on home services with verified professionals and introduces AI-powered diagnostics and bidding.

2.2.3 Thumbtack

Thumbtack [9] helps customers find professionals in multiple categories by allowing providers to bid on jobs.

Features

- Bidding-based pricing.
- Professional profiles and ratings.
- Wide range of services.

Limitations

- No real-time tracking.
- No AI-driven price estimation.
- Limited transparency in bidding.

Comparison with Home Services App

The Home Services App improves the bidding process using AI-guided price ranges and adds real-time tracking for service delivery.

2.2.4 Handy

Handy [10] provides cleaning and handyman services with fixed and transparent pricing.

Features

- Fixed pricing.
- Customer reviews and scheduling.
- In-app booking.

Limitations

- Limited service categories.
- No AI or bidding mechanism.
- No real-time tracking.

Comparison with Home Services App

The proposed platform expands service categories and introduces competitive bidding with AI support.

2.2.5 Local Directories and Word-of-Mouth

Before digital platforms, customers primarily relied on local directories or recommendations.

Features

- Accessible for users without digital access.
- No platform usage costs.

Limitations

- No verification of professionals.
- No fixed or transparent pricing.
- Time-consuming to contact multiple providers.

Comparison with Home Services App

The Home Services App overcomes these problems by offering verified professionals, transparent pricing, and seamless in-app communication.

2.3 Comparative Analysis of Home Services Platforms

To better understand the strengths and weaknesses of existing platforms, a detailed comparative analysis is presented in Table 2.1.

Table 2.1: Comparative Analysis of Home Services Platforms

Feature / Platform	**Home Services App**	**Urban Clap**	**Task Rabbit**	**Thumb tack**	**Handy**	**Local Directories**
Professional Verification	Yes (Verified badges)	Yes	No	No	No	No
AI-Powered Assistance	Yes (AI chat + estimation)	No	No	No	No	No
Real-Time Bidding	Yes (AI-guided bidding)	No	No	Yes	No	No
Google Maps Integration	Yes (Real-time tracking)	No	No	No	No	No
In-App Messaging	Yes	Yes	No	No	No	No
Service Range	Wide	Wide	Broad	Broad	Limited	Varies
Pricing Model	Transparent (Bidding + AI)	Fixed	Hourly	Bidding	Fixed	No standardization
Customer Reviews	Yes	Yes	Yes	Yes	Yes	No
Real-Time Notifications	Yes	Yes	No	No	No	No
Payment Integration	Yes (Secure online)	Yes	No	No	Yes	No
Target Audience	Residential + Commercial	Residential + Commercial	Freelancers + Customers	Residential + Commercial	Residential	General Public
Unique Features	AI assistance, bidding, tracking, verification	Verification	Broad tasks	Bidding focus	Fixed pricing	Offline simple

2.4 Key Takeaways from the Table

- The Home Services App stands out with its AI-powered assistance, real-time bidding, and Google Maps integration—features not simultaneously offered by competitors.
- Unlike UrbanClap and Handy, the app provides flexible pricing through bidding,

ensuring competitive rates for customers.

- Professional verification and in-app messaging improve trust and communication, addressing limitations in TaskRabbit and Thumbtack.
- Local directories lack the transparency and modern features introduced by the Home Services App.

Chapter 3

Requirements Specification

3.1 Proposed System

The proposed Home Services App is designed to address the challenges associated with finding reliable, verified, and affordable home service professionals. The system uses modern technologies such as Supabase, Flutter, Google Maps API, AI-driven diagnostics, and real-time bidding to streamline the process of connecting customers with professionals.

The system consists of three main components:

- **Customer Module** – Allows users to browse jobs, request services, chat with professionals, track job status, and make secure payments.
- **Service Provider Module** – Enables verified professionals to bid on jobs, manage schedules, communicate with customers, and handle payments.
- **Admin Module** – Provides backend management tools for user verification, job monitoring, payment auditing, dispute resolution, and system analytics.

The system architecture ensures scalability, real-time updates, and a seamless user experience across Android and iOS devices.

3.2 Functional Requirements

The functional requirements describe the core capabilities of the system:

- User registration and secure Supabase authentication.
- Customer job posting with category, budget, description, and media.
- Real-time bidding from service providers.
- In-app chat between customers and providers.
- Google Maps–based live tracking.
- Stripe payment integration for card and online payments.
- Rating and review system after job completion.
- Admin management features for monitoring the entire platform.

3.3 Non-Functional Requirements

3.3.1 Availability

The system must be operational and accessible 24/7. The backend is deployed on cloud infrastructure ensuring high availability and automatic failover.

3.3.2 Usability

The user interface must be intuitive, clean, and easy to navigate. Both customers and service providers should be able to perform tasks without extensive training.

3.3.3 Performance

The system must respond to user actions within acceptable time limits. All API responses should be delivered within 2 seconds under normal load.

3.3.4 Reliability

All transactions, including job bids, payments, and chat messages, must be reliably stored in Supabase with real-time backups and replication.

3.3.5 Security

User data, passwords, and payment information must be secured using industry-standard encryption. Stripe ensures PCI-compliant online payments, and Supabase provides secure authentication.

3.3.6 Scalability

The system should handle increased user traffic by auto-scaling. Real-time database updates ensure smooth operation even under heavy usage.

3.3.7 Compatibility

The mobile app must run smoothly on both Android and iOS. Google Maps, Supabase, and Stripe integrations must remain stable across platforms.

3.4 System Features

The system includes the following key features:

- Real-time job posting and bidding.
- In-app messaging with image sharing.
- Live tracking of professionals.
- AI-powered problem analysis and price estimation.

- Secure payments using Stripe.
- Appointment scheduling.
- Professional verification workflow.

3.5 Use Case Diagram

UC-001: User Registration

Shows how a new guest user creates an account in the app by submitting basic details, choosing a role, and getting a verification email through Supabase Auth.

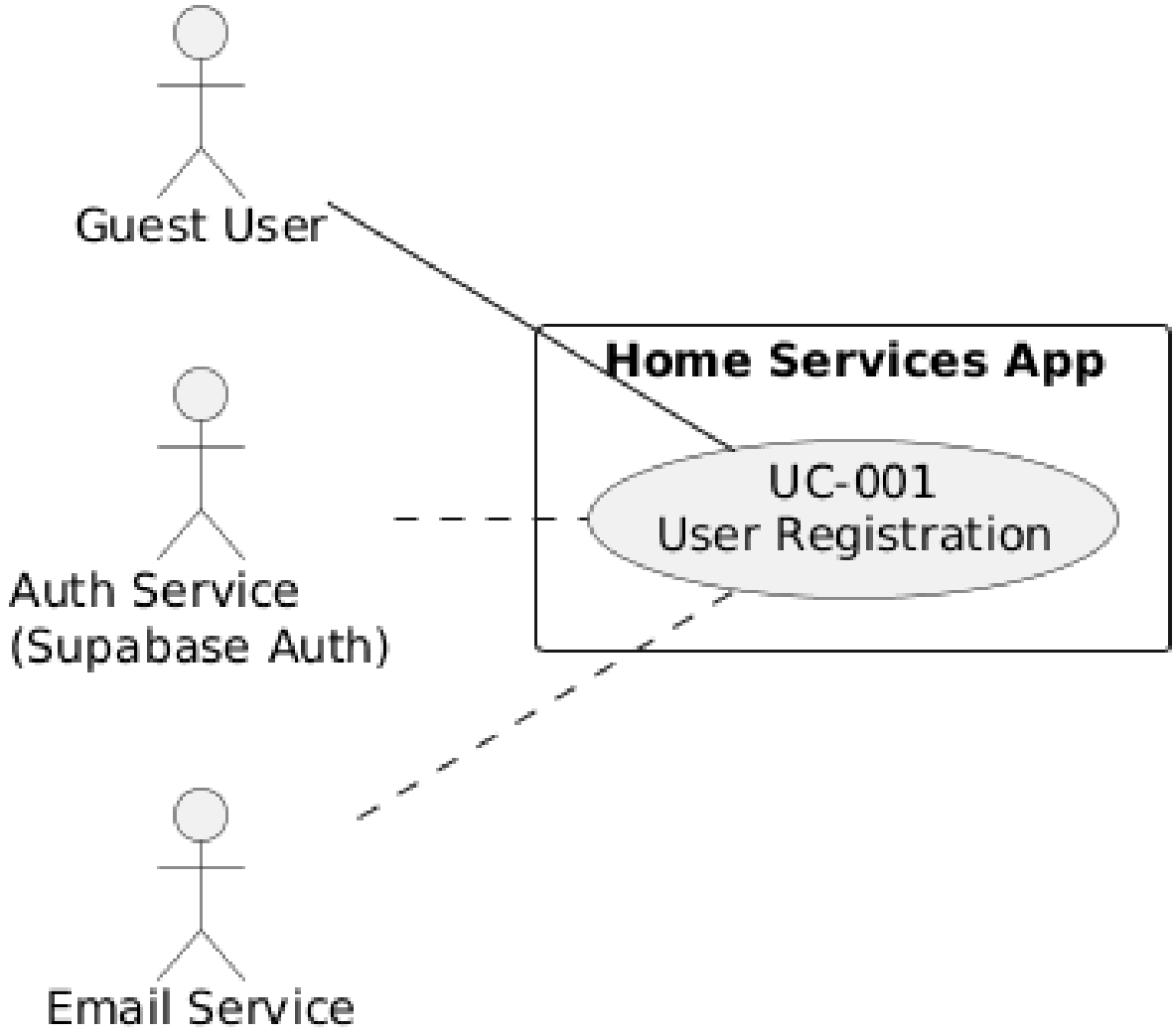

Figure 3.1: Use Case Diagram for UC-001: User Registration

Table 3.1: Use Case UC-001: User Registration

Use Case ID	UC-001
Title	User Registration
Description	This use case allows a new user to create an account in the Home Services application by providing required personal information.
Actor	Guest User
Pre-Conditions	• User is not already registered. • User has a valid email address. • User has access to the registration screen.
Basic Flow	1. User opens the application. 2. User navigates to the registration page. 3. User enters full name, email, and password. 4. User selects role (Customer / Service Provider). 5. User agrees to Terms and Conditions. 6. User clicks the "Register" button. 7. System validates the provided information. 8. System saves user details to the database. 9. System sends email verification link. 10. System displays successful registration message.
Alternate Flow	• **Invalid Email:** System shows "Email already exists". • **Weak Password:** System requests password correction. • **Missing Fields:** System highlights required fields. • **Server Error:** System displays "Registration failed".

Post-Conditions	• New user account is created. • Verification email is sent to user. • User may log in after verification.

UC-002: User Login

Describes how a registered user logs in with email and password, gets validated by Supabase Auth, and is redirected to the correct dashboard based on role.

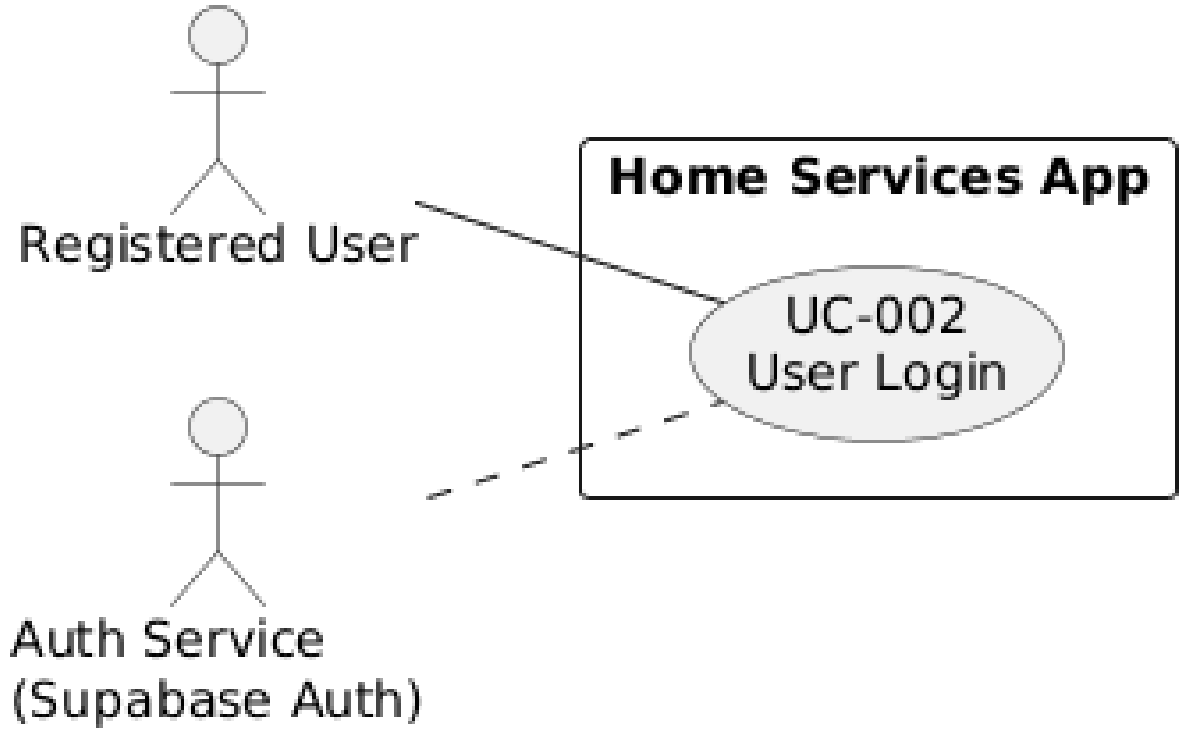

Figure 3.2: Use Case Diagram for UC-002: User Login

Table 3.2: Use Case UC-002: User Login

Use Case ID	UC-002
Title	User Login
Description	This use case allows a registered user to log in to the Home Services application using valid credentials.
Actor	Registered User (Customer / Service Provider / Admin)
Pre-Conditions	• User is already registered in the system. • User has a verified and active account. • User has access to the login screen.
Basic Flow	1. User opens the application. 2. User navigates to the login page (if not already there). 3. User enters registered email. 4. User enters password. 5. User clicks the "Login" button. 6. System validates the email and password. 7. System checks if the account is active and verified. 8. System creates a session for the user. 9. System redirects user to the respective dashboard (Customer / Provider / Admin).
Alternate Flow	• **Invalid Credentials:** System shows "Invalid email or password" and asks user to try again. • **Unverified Account:** System shows "Please verify your email before login". • **Suspended Account:** System shows "Your account has been suspended. Contact support". • **Forgot Password:** User clicks "Forgot Password" and is redirected to password reset flow.

Post-Conditions	• User is successfully logged in. • Active session is created for the user. • User can access all allowed features according to their role.

UC-003: User Logout

Explains how a logged-in user securely logs out, ending the active session and clearing stored credentials from the app.

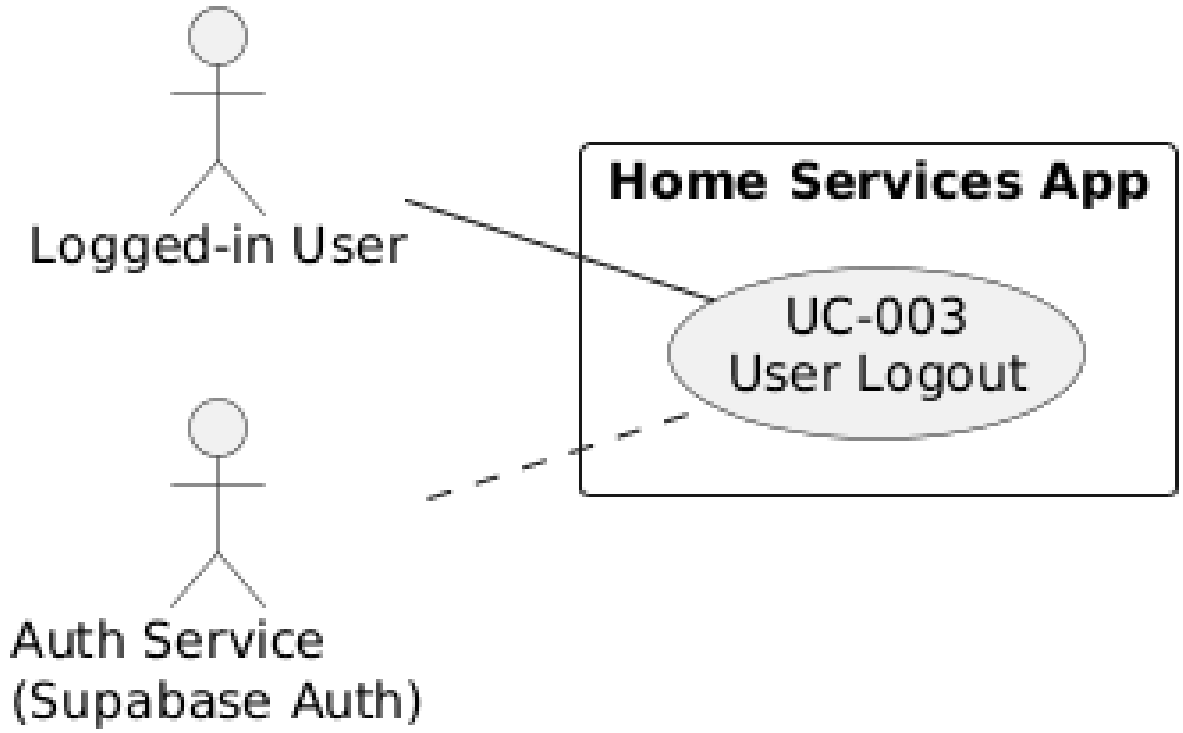

Figure 3.3: Use Case Diagram for UC-003: User Logout

Table 3.3: Use Case UC-003: User Logout

Use Case ID	UC-003
Title	User Logout
Description	This use case allows a logged-in user to securely log out from the Home Services application and end their active session.
Actor	Registered User (Customer / Service Provider / Admin)
Pre-Conditions	• User must be logged in. • User is on any authenticated screen of the application.
Basic Flow	1. User opens the menu or profile section. 2. User clicks the "Logout" button. 3. System asks for confirmation (optional). 4. User confirms logout. 5. System terminates the active session. 6. System clears stored user data and tokens. 7. System redirects user to the login or home screen.
Alternate Flow	• **User Cancels Logout:** User cancels confirmation dialog and remains logged in. • **Network Issue:** Session termination may delay; system retries automatically.
Post-Conditions	• User session is fully terminated. • User is redirected to the login/home page. • User must log in again to access protected features.

UC-004: Update User Profile

Shows how a logged-in user views and edits their profile information (name, contact, address, picture) and saves the updated data in the system.

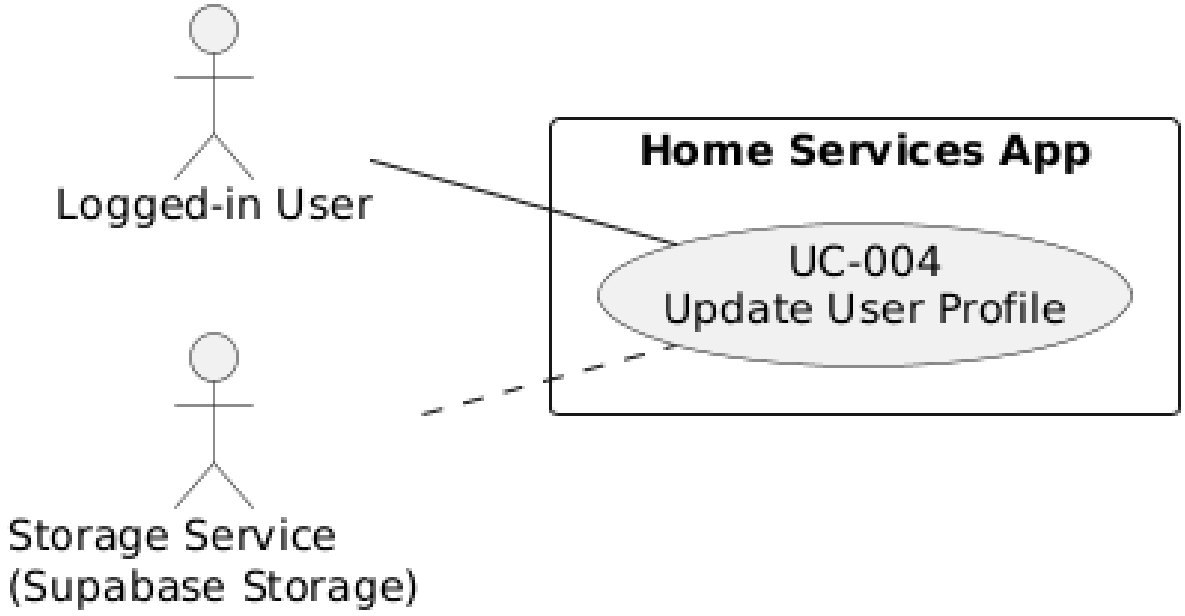

Figure 3.4: Use Case Diagram for UC-004: Update User Profile

Table 3.4: Use Case UC-004: Update User Profile

Use Case ID	UC-004
Title	Update User Profile
Description	This use case allows a logged-in user to update their personal information and profile details in the Home Services application.
Actor	Registered User (Customer / Service Provider)
Pre-Conditions	• User is logged in. • User has access to the profile screen.
Basic Flow	1. User navigates to the "Profile" section. 2. System displays current profile information (name, phone, address, etc.). 3. User clicks the "Edit Profile" button. 4. User updates required fields (e.g., name, contact number, address, short bio). 5. User optionally updates profile picture. 6. User clicks the "Save" button. 7. System validates the entered data. 8. System saves updated information in the database. 9. System shows a success message confirming the update.
Alternate Flow	• **Invalid Data:** If any field is invalid (e.g., wrong phone format), system highlights the field and asks the user to correct it. • **Image Upload Failed:** If profile picture upload fails, system keeps old picture and shows an error message, but other data can still be saved. • **User Cancels:** User cancels editing; no changes are saved and old data remains.

Post-Conditions	• User profile information is updated in the system. • Updated data is reflected across the application wherever profile is shown.

UC-005: Post a Job (Customer)

Describes how a customer creates a new job request by entering details, budget, location, and images, after which the job is stored as "Open" for providers.

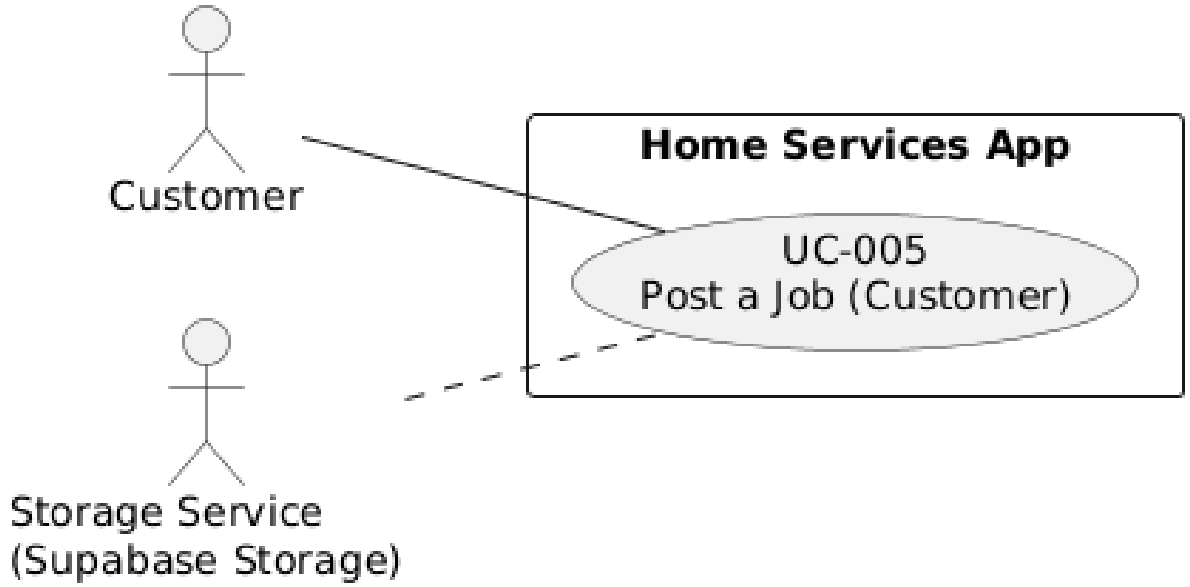

Figure 3.5: Use Case Diagram for UC-005: Post a Job

Table 3.5: Use Case UC-005: Post a Job

Use Case ID	UC-005
Title	Post a Job
Description	This use case allows a customer to create and post a new job request in the Home Services application. The posted job becomes visible to service providers.
Actor	Customer
Pre-Conditions	• User is logged in as a Customer. • User has access to the "Post Job" screen.
Basic Flow	1. Customer navigates to the "Post Job" page. 2. Customer enters job title. 3. Customer adds job description. 4. Customer selects job category (e.g., plumbing, cleaning, electrical). 5. Customer enters estimated budget. 6. Customer enters location/address. 7. Customer optionally uploads images (e.g., 1–5 photos). 8. Customer clicks the "Submit" or "Post Job" button. 9. System validates all required fields. 10. System saves the job information to the database. 11. System sets job status to "Open". 12. System sends notifications to available providers (if applicable). 13. System redirects customer to the job details page.

Alternate Flow	• **Missing Required Fields:** System highlights the missing fields and asks the customer to correct them. • **Invalid Budget:** System requests a valid numeric budget. • **Image Upload Failure:** Images fail to upload; system posts the job without images and shows a warning. • **Customer Cancels:** Job is not posted and no data is saved.
Post-Conditions	• A new job is created in the system with status "Open". • Providers can now view and bid on the job. • Job appears in the customer's "My Jobs" list.

UC-006: Browse Available Jobs (Provider)

Shows how a service provider views the list of open jobs, filters them by category, budget, or location, and opens any job to see full details.

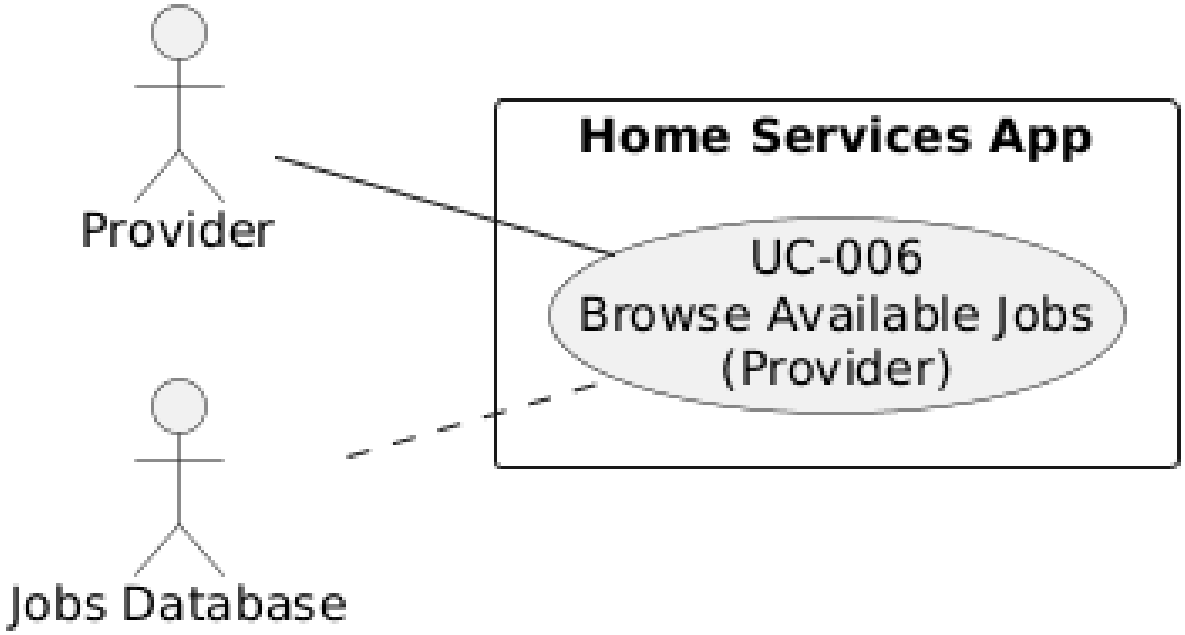

Figure 3.6: Use Case Diagram for UC-006: Browse Available Jobs

Table 3.6: Use Case UC-006: Browse Available Jobs

Use Case ID	UC-006
Title	Browse Available Jobs
Description	This use case allows a service provider to view and explore all open job postings submitted by customers. Providers can check job details before placing a bid.
Actor	Service Provider
Pre-Conditions	• Provider is logged in. • Provider account is active and approved.
Basic Flow	1. Provider navigates to the "Available Jobs" section. 2. System displays a list of all open jobs. 3. Provider scrolls through the job list. 4. Provider applies filters (category, budget, location) if needed. 5. System updates job listings based on applied filters. 6. Provider selects a job to view details. 7. System displays full job details including images, budget, description, and location. 8. Provider decides whether to place a bid or return to the list.
Alternate Flow	• **No Jobs Available:** System shows a message such as "No jobs available at the moment." • **No Filter Results:** System displays "No jobs match your criteria." • **Job Closed Before Viewing:** If the job is no longer available, system displays "This job is no longer open."
Post-Conditions	• Provider gains access to a list of open jobs. • Provider may proceed to place a bid using another use case.

UC-007: Place Bid on Job (Provider)

Explains how a provider submits a bid on an open job with a proposed price, estimated time, and message, and how the customer is notified of the new bid.

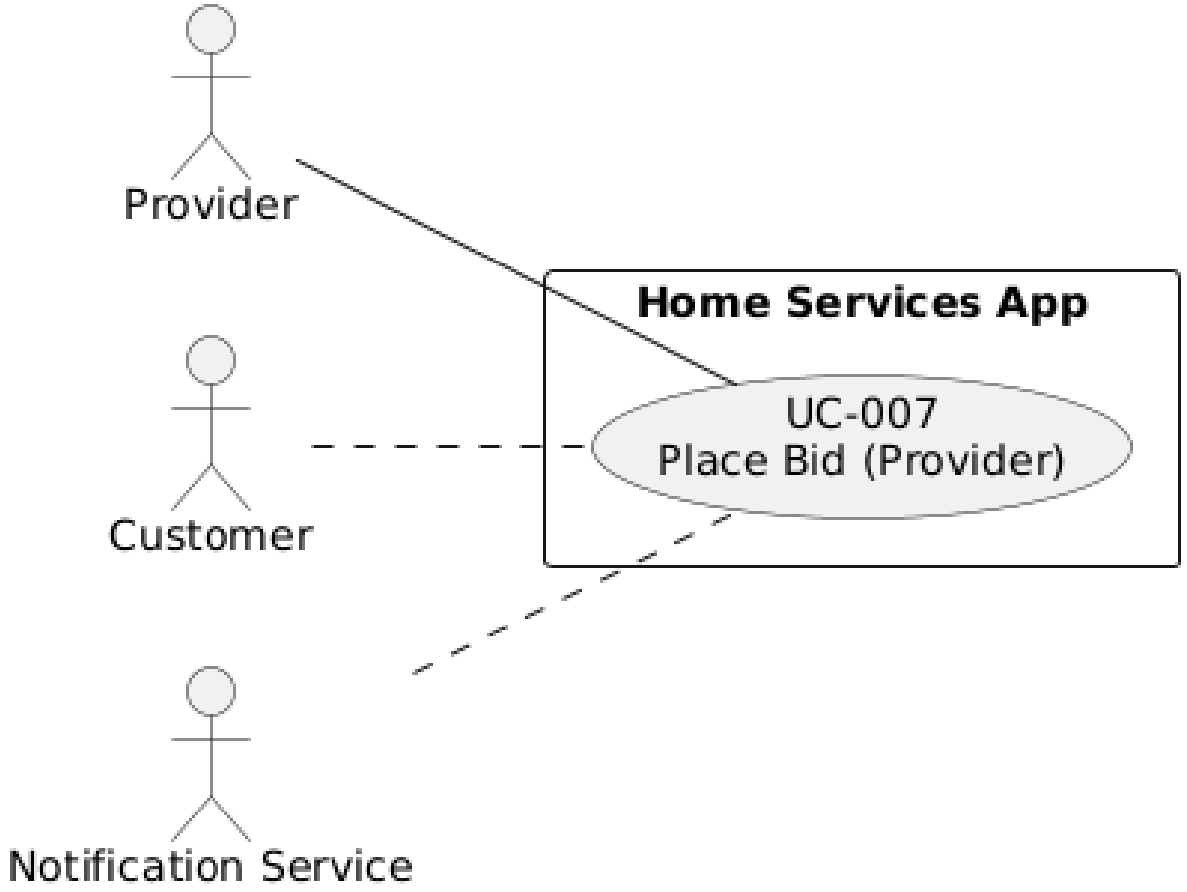

Figure 3.7: Use Case Diagram for UC-007: Place Bid on Job

Table 3.7: Use Case UC-007: Place Bid on Job

Use Case ID	UC-007
Title	Place Bid on Job
Description	This use case allows a service provider to place a bid on an open job posted by a customer, including proposed price and time estimate.
Actor	Service Provider
Pre-Conditions	• Provider is logged in. • Provider account is active and verified. • There is at least one job with status "Open". • Provider can view the job details screen.
Basic Flow	1. Provider opens the job details page of an open job. 2. System displays full job information (title, description, budget, location, images). 3. Provider clicks the "Place Bid" button. 4. System shows the bid form. 5. Provider enters proposed bid amount. 6. Provider enters estimated completion time or date. 7. Provider optionally adds a short message or proposal note for the customer. 8. Provider clicks the "Submit Bid" button. 9. System validates the entered data (e.g., amount format). 10. System saves the bid in the database and links it to the job and provider. 11. System updates the bid count for the job. 12. System notifies the customer that a new bid has been received. 13. System shows a success message to the provider.

Alternate Flow	• **Invalid Amount:** If bid amount is invalid or below a minimum limit, system shows an error and asks provider to correct it. • **Job Already Closed:** If the job status becomes "Assigned" or "Closed" before bid submission, system shows "This job is no longer accepting bids." • **Duplicate Bid:** If provider already placed a bid on this job, system may prevent duplicate bids or allow editing the previous bid (as per system rules). • **Network/Server Error:** System fails to save the bid and shows an error message; provider may retry.
Post-Conditions	• A new bid is stored in the system for the selected job. • Customer can view the bid in the list of received bids. • Job's bid count is updated.

UC-008: View and Compare Bids (Customer)

Describes how a customer opens their job, views all received bids with provider ratings and prices, and compares them before making a decision.

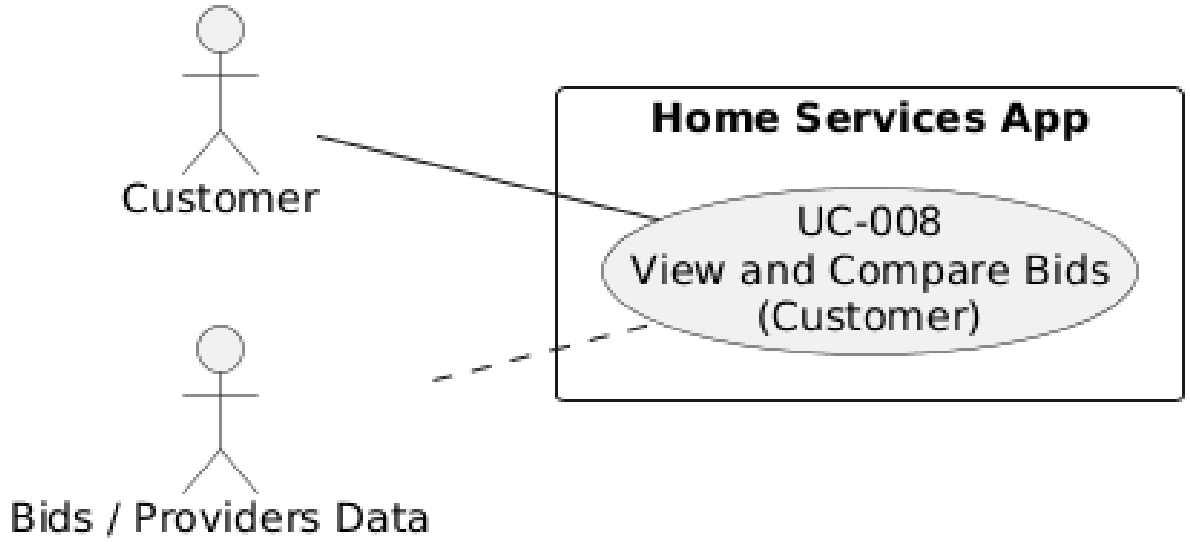

Figure 3.8: Use Case Diagram for UC-008: View and Compare Bids

Table 3.8: Use Case UC-008: View and Compare Bids

Use Case ID	UC-008
Title	View and Compare Bids
Description	This use case allows a customer to view all bids submitted by service providers for a specific job and compare them based on pricing, estimated time, and provider ratings.
Actor	Customer
Pre-Conditions	• Customer is logged in. • Customer has posted at least one job. • Providers have submitted bids on the selected job.
Basic Flow	1. Customer navigates to the "My Jobs" section. 2. Customer selects a specific job with active bids. 3. System displays a list of all received bids. 4. Each bid displays provider name, rating, proposed price, and estimated time. 5. Customer clicks any bid to see full details. 6. Customer compares bids to determine the most suitable provider. 7. Customer may shortlist or mark preferred bids (optional feature).
Alternate Flow	• **No Bids Received:** System displays the message "No bids received yet." • **Job Closed Before Viewing:** System shows "This job is no longer accepting bids." • **Provider Withdraws Bid:** System updates the list and hides removed bids.
Post-Conditions	• Customer gains a clear view of all submitted bids. • Customer may proceed to assign a provider using another use case.

UC-009: Assign Job to Provider

Shows how a customer accepts one bid, assigns the job to that provider, updates the job status to "Assigned," and notifies both selected and unselected providers.

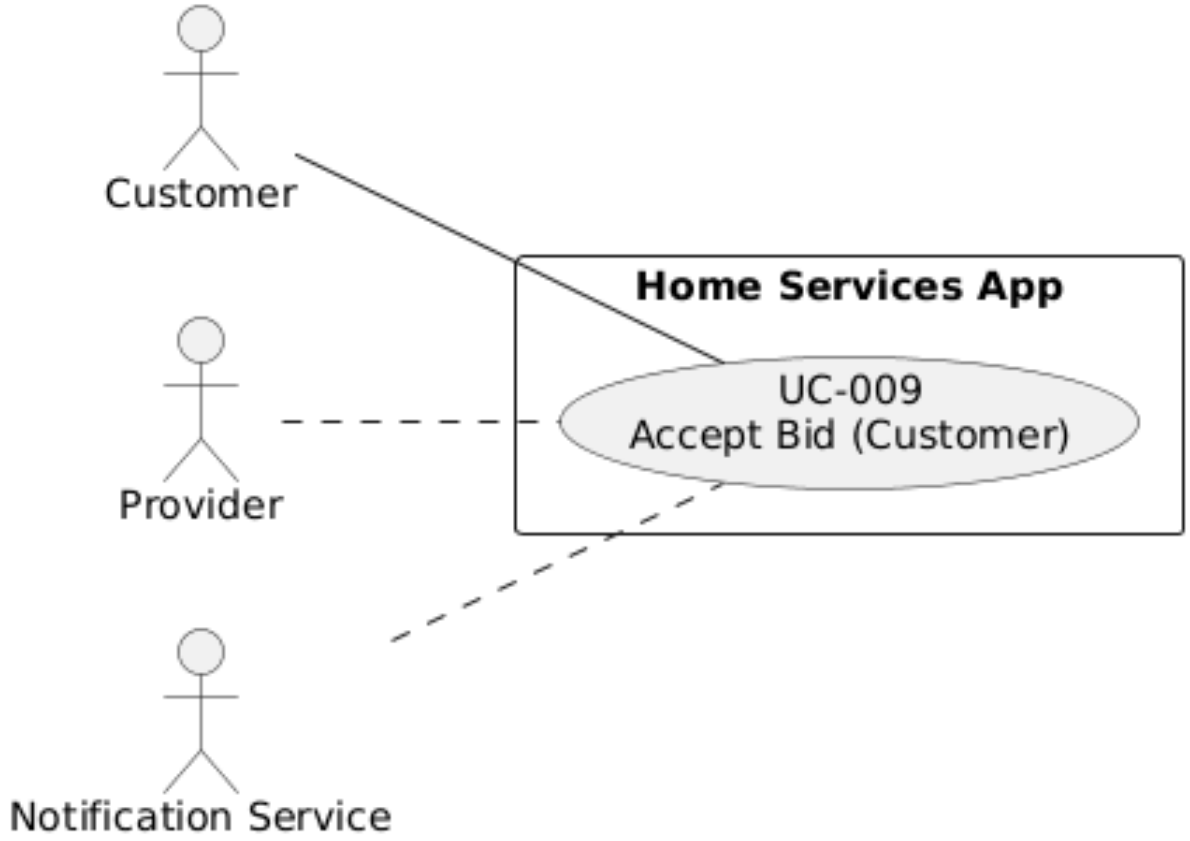

Figure 3.9: Use Case Diagram for UC-009: Assign Job to Provider

Table 3.9: Use Case UC-009: Assign Job to Provider

Use Case ID	UC-009
Title	Assign Job to Provider
Description	This use case allows a customer to assign a job to a selected service provider by approving one of the received bids.
Actor	Customer
Pre-Conditions	• Customer is logged in. • Job has at least one active bid. • Customer has viewed the list of bids.
Basic Flow	1. Customer opens “My Jobs”. 2. Customer selects a job with bids. 3. System displays list of received bids. 4. Customer selects a bid to review. 5. Customer clicks “Assign Job” button. 6. System asks for confirmation. 7. Customer confirms the assignment. 8. System updates job status to “Assigned”. 9. System notifies the selected provider about the assignment. 10. System notifies unselected providers that the job is closed. 11. System displays success message to the customer.

Alternate Flow	• **Customer Cancels Assignment:** No changes made; system returns to bid list. • **Bid Withdrawn Before Assignment:** System shows "This bid is no longer available." • **Provider Account Suspended:** System prevents assignment and notifies customer. • **Job Already Assigned:** System shows "Job already assigned to another provider."
Post-Conditions	• Job is assigned to the selected provider. • Job status becomes "Assigned". • Provider can now begin or schedule the job.

UC-010: Update Job Status (Provider)

Explains how the assigned provider updates job progress (e.g., In Progress, Completed), optionally adds notes or photos, and triggers notifications to the customer.

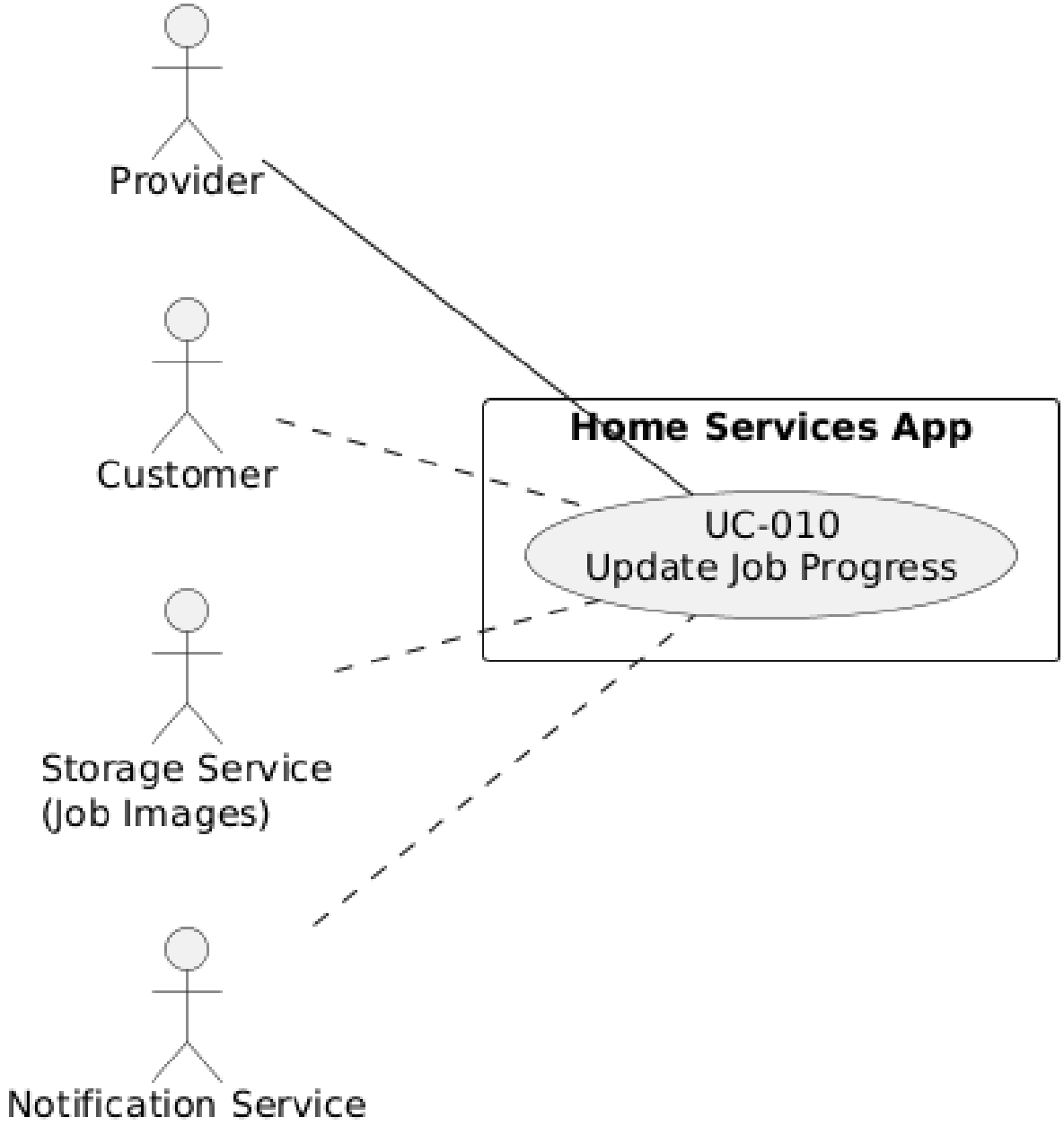

Figure 3.10: Use Case Diagram for UC-010: Update Job Status

Table 3.10: Use Case UC-010: Update Job Status

Use Case ID	UC-010
Title	Update Job Status
Description	This use case allows a service provider to update the current status of an assigned job (e.g., In Progress, Completed).
Actor	Service Provider
Pre-Conditions	• Provider is logged in. • Provider has been assigned a job. • Job status is currently "Assigned".
Basic Flow	1. Provider navigates to the "My Jobs" or "Assigned Jobs" section. 2. Provider selects an assigned job. 3. System displays job details including current status. 4. Provider clicks "Update Status". 5. Provider selects a new status (e.g., In Progress, Completed). 6. Provider optionally adds a note or attachment (e.g., proof of work). 7. Provider confirms the action. 8. System updates the job status in the database. 9. System notifies the customer about the status update. 10. System displays a success message.
Alternate Flow	• **Invalid Status Sequence:** If provider tries to skip stages (e.g., move directly to Completed), system shows "Invalid status update". • **Job Already Completed/Cancelled:** System prevents updates and notifies provider. • **Network/Server Issue:** Status update fails; provider is asked to retry.

Post-Conditions	• Job status is updated to the new state. • Customer is informed about the status change. • Updated status reflects in both customer and provider dashboards.

UC-011: Cancel Job (Customer)

Describes how a customer cancels an open or assigned job, changes its status to "Cancelled," and notifies all affected providers.

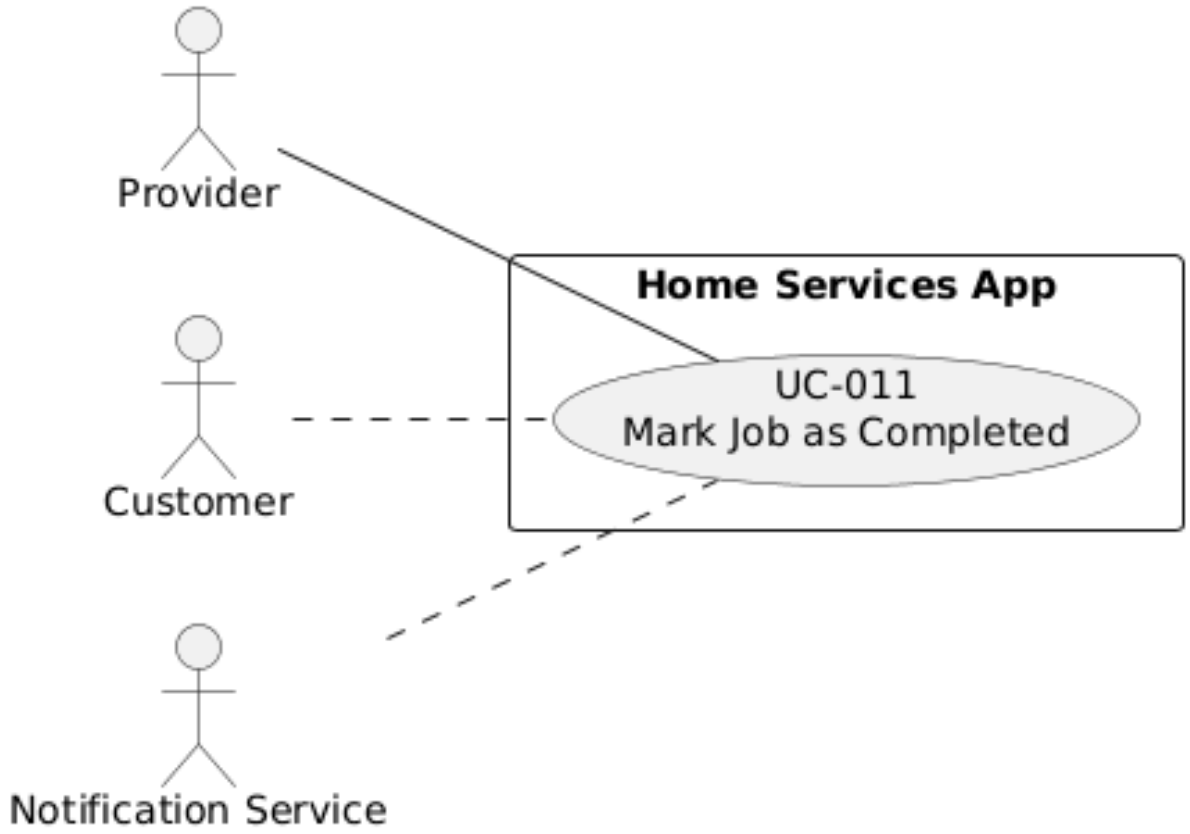

Figure 3.11: Use Case Diagram for UC-011: Cancel Job

Table 3.11: Use Case UC-011: Cancel Job

Use Case ID	UC-011
Title	Cancel Job
Description	This use case allows a customer to cancel a job they previously created, whether it is still open for bids or already assigned to a provider.
Actor	Customer
Pre-Conditions	• Customer is logged in. • Customer has created the job. • Job is in "Open" or "Assigned" status (not completed).
Basic Flow	1. Customer navigates to the "My Jobs" section. 2. Customer selects a job they want to cancel. 3. System displays job details including current status. 4. Customer clicks the "Cancel Job" button. 5. System asks for cancellation confirmation. 6. Customer confirms the cancellation. 7. System updates job status to "Cancelled". 8. System notifies any providers who placed bids. 9. If job was assigned, system sends a cancellation notice to the assigned provider. 10. System displays a success message to the customer.

Alternate Flow	• **Customer Cancels the Action:** Customer dismisses the confirmation dialog and job remains unchanged. • **Job Already Completed:** System prevents cancellation and shows "Job cannot be cancelled after completion." • **Admin-Locked Job:** System prevents cancellation and informs customer to contact support. • **Network/Server Issue:** System cannot update status and asks customer to retry.
Post-Conditions	• Job status becomes "Cancelled". • Providers are notified of the cancellation. • Job no longer appears in provider job listings.

UC-012: Chat Between Customer and Provider

Shows how customer and provider use in-app messaging linked to a job to exchange text (and media), with messages delivered and stored in real-time.

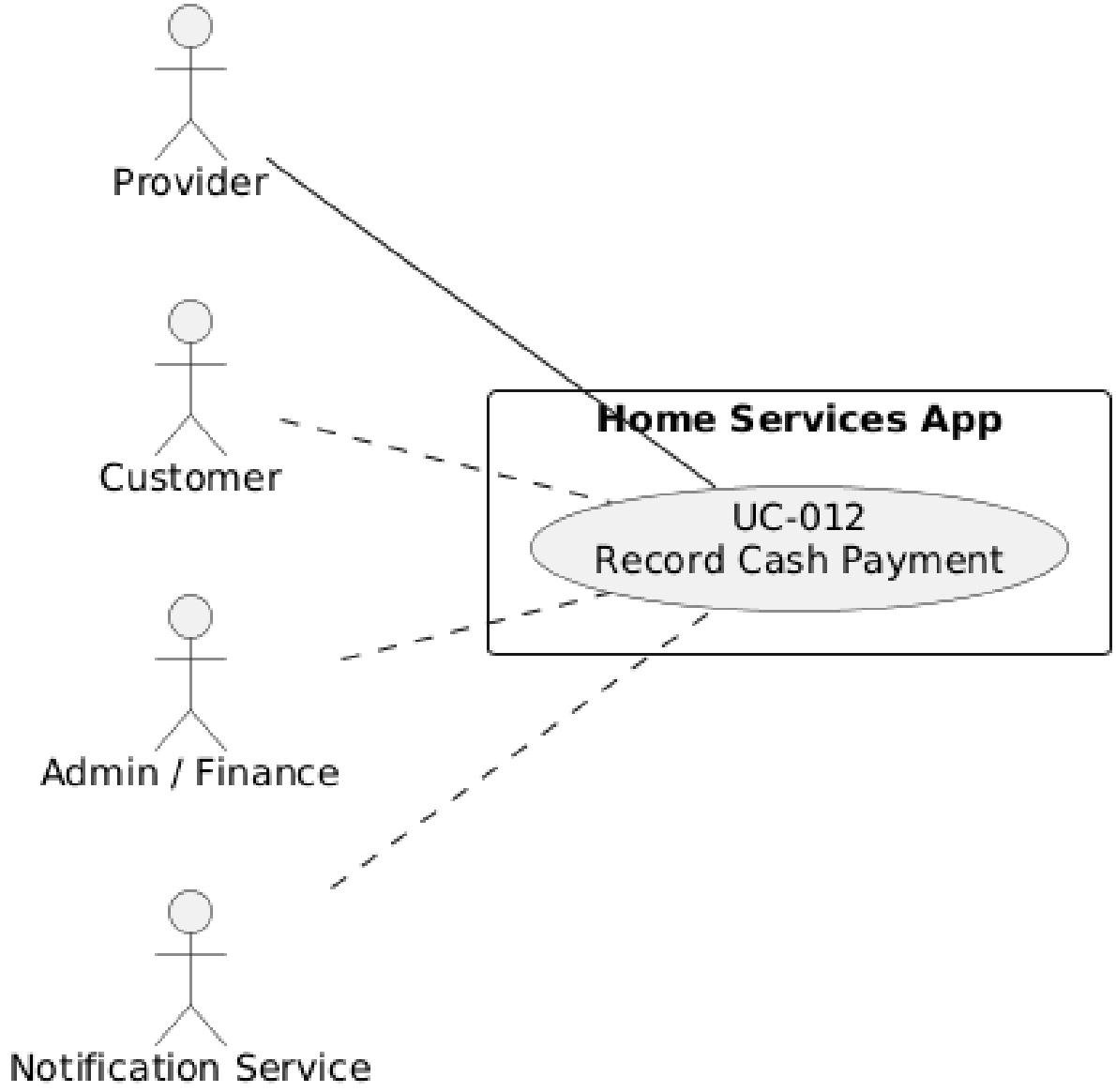

Figure 3.12: Use Case Diagram for UC-012: Chat Between Customer and Provider

Table 3.12: Use Case UC-012: Chat Between Customer and Provider

Use Case ID	UC-012
Title	Chat Between Customer and Provider
Description	This use case allows a customer and a service provider to communicate through an in-app chat system regarding a job or service request.
Actor	Customer, Service Provider
Pre-Conditions	• Both users are logged in. • A job exists between the two parties (open, assigned, or in progress). • Chat feature is enabled in the system.
Basic Flow	1. User opens the application. 2. User navigates to the "Messages" or "Chat" section. 3. System displays a list of active conversations related to jobs. 4. User selects a conversation (Customer to Provider). 5. System loads full chat history. 6. User types a message in the chat box. 7. User clicks "Send". 8. System delivers the message in real-time. 9. The other user receives a chat notification. 10. Chat updates instantly for both users.

Alternate Flow	• **No Internet Connection:** Message not sent; system displays "Connection error". • **Provider or Customer Blocked:** Chat disabled, system shows "You cannot send messages to this user". • **Job Cancelled or Closed:** Chat becomes read-only. • **Message Delivery Failure:** System retries sending automatically.
Post-Conditions	• Message is saved in the chat history. • Both users can view the updated conversation. • Chat remains available for future reference.

UC-013: Rate and Review Provider

Explains how, after job completion, a customer gives a star rating and written review, which updates the provider's overall rating and becomes visible to others.

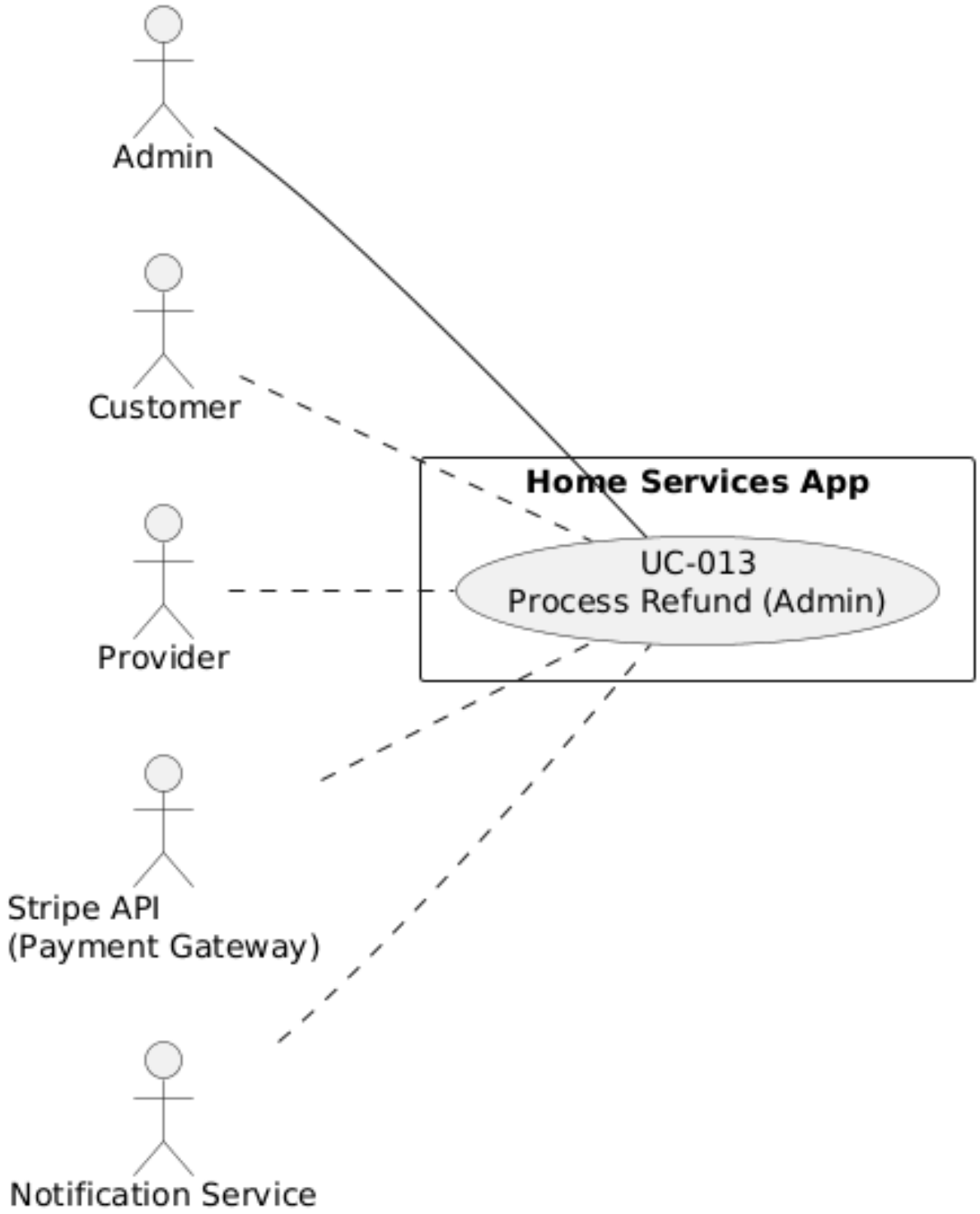

Figure 3.13: Use Case Diagram for UC-013: Rate and Review Provider

Table 3.13: Use Case UC-013: Rate and Review Provider

Use Case ID	UC-013
Title	Rate and Review Provider
Description	This use case allows a customer to rate and review a service provider after the successful completion of a job. The feedback influences the provider's profile rating.
Actor	Customer
Pre-Conditions	• Customer is logged in. • Job must be completed by the provider. • Customer has not already submitted a review for this job.
Basic Flow	1. Customer opens the "My Jobs" section. 2. Customer selects a completed job. 3. System displays the provider details and job summary. 4. Customer clicks the "Rate and Review" button. 5. Customer selects a star rating (1–5). 6. Customer writes a review comment describing the experience. 7. Customer clicks "Submit Review". 8. System validates the review details. 9. System saves the rating and review in the database. 10. System updates the provider's overall rating. 11. System shows a confirmation message to the customer.

Alternate Flow	• **Missing Rating or Comment:** System shows an error requesting required fields. • **Review Already Submitted:** System prevents additional reviews and shows "Review already submitted for this job". • **Job Not Completed:** System disables the review option. • **Network Failure:** System is unable to save review and asks customer to retry.
Post-Conditions	• A new rating and review are added for the provider. • Provider's profile updates to reflect new average rating. • Review becomes visible to other customers.

UC-014: Payment Processing

Describes how a customer pays for a completed job using the integrated payment gateway (e.g., Stripe), and how the app confirms and records the transaction.

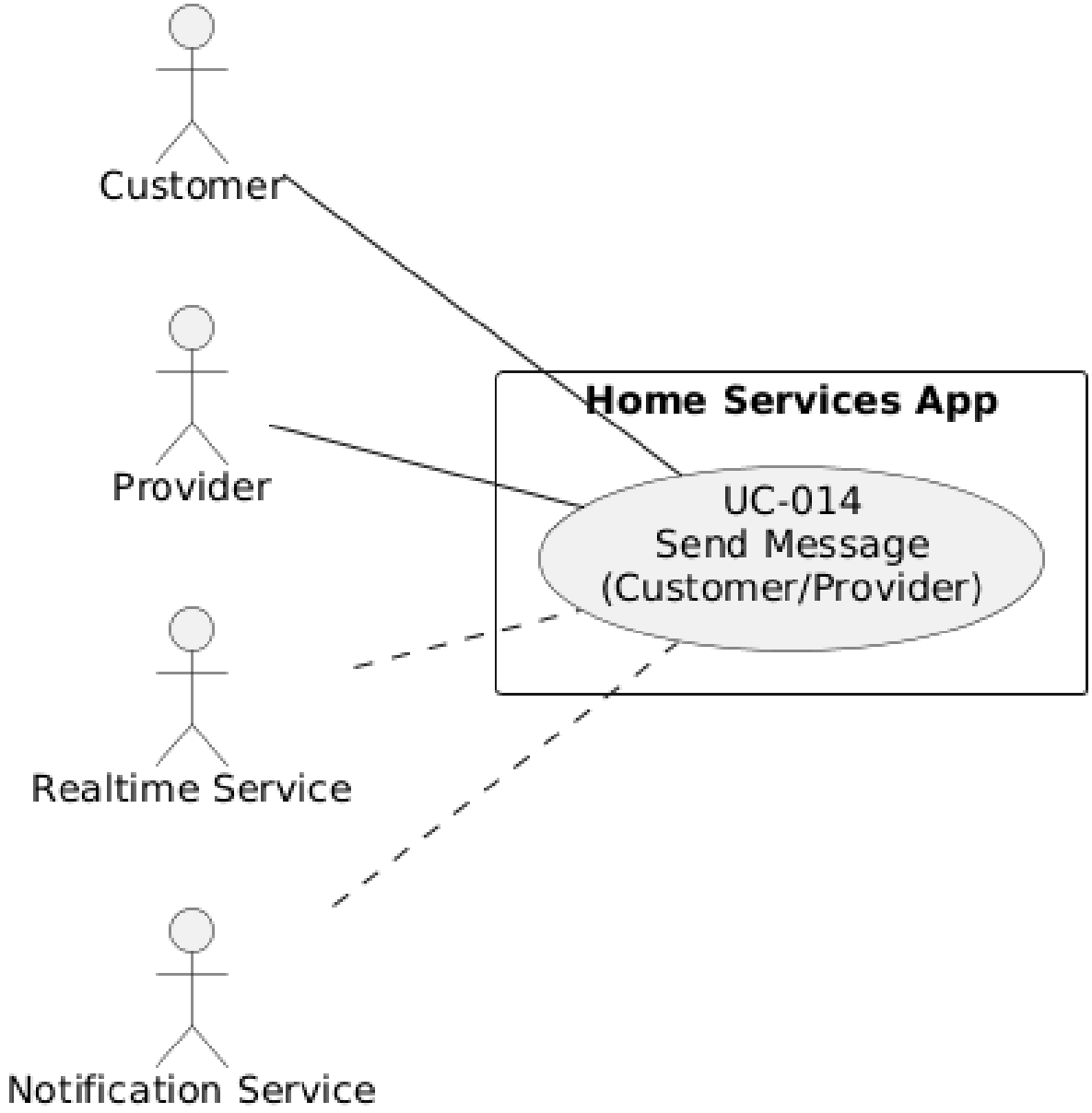

Figure 3.14: Use Case Diagram for UC-014: Payment Processing

Table 3.14: Use Case UC-014: Payment Processing

Use Case ID	UC-014
Title	Payment Processing
Description	This use case allows a customer to make a secure payment for the completed job using available payment methods integrated within the Home Services application.
Actor	Customer
Pre-Conditions	• Job has been completed by the provider. • Customer is logged in. • Payment gateway is active and functional. • Customer has a valid payment method available.
Basic Flow	1. Customer opens the "My Jobs" section. 2. Customer selects a completed job requiring payment. 3. System displays total payable amount and job summary. 4. Customer clicks on "Make Payment". 5. System displays available payment methods (Card, Wallet, Cash Option if allowed). 6. Customer selects a payment method. 7. Customer enters necessary details (e.g., card info). 8. Customer confirms the payment. 9. System processes the payment through the payment gateway. 10. System verifies and confirms successful payment. 11. System updates job status to "Paid". 12. System notifies provider about payment confirmation. 13. System shows a success receipt to the customer.

Alternate Flow	• **Payment Failed:** System displays "Payment unsuccessful" and prompts customer to retry or use another method. • **Insufficient Balance:** System shows "Transaction declined". • **Invalid Card Details:** System requests correction of incorrect information. • **Gateway Timeout:** System shows "Connection error" and pauses transaction. • **Customer Cancels:** Payment is not processed; job remains unpaid.
Post-Conditions	• Payment is processed and recorded in the system. • Job status is updated to "Paid". • Provider receives payment confirmation. • Customer receives a digital receipt.

UC-015: Withdraw Earnings (Provider)

Shows how a provider views their wallet balance, requests a withdrawal to a linked payout method, and how the system records and processes the withdrawal.

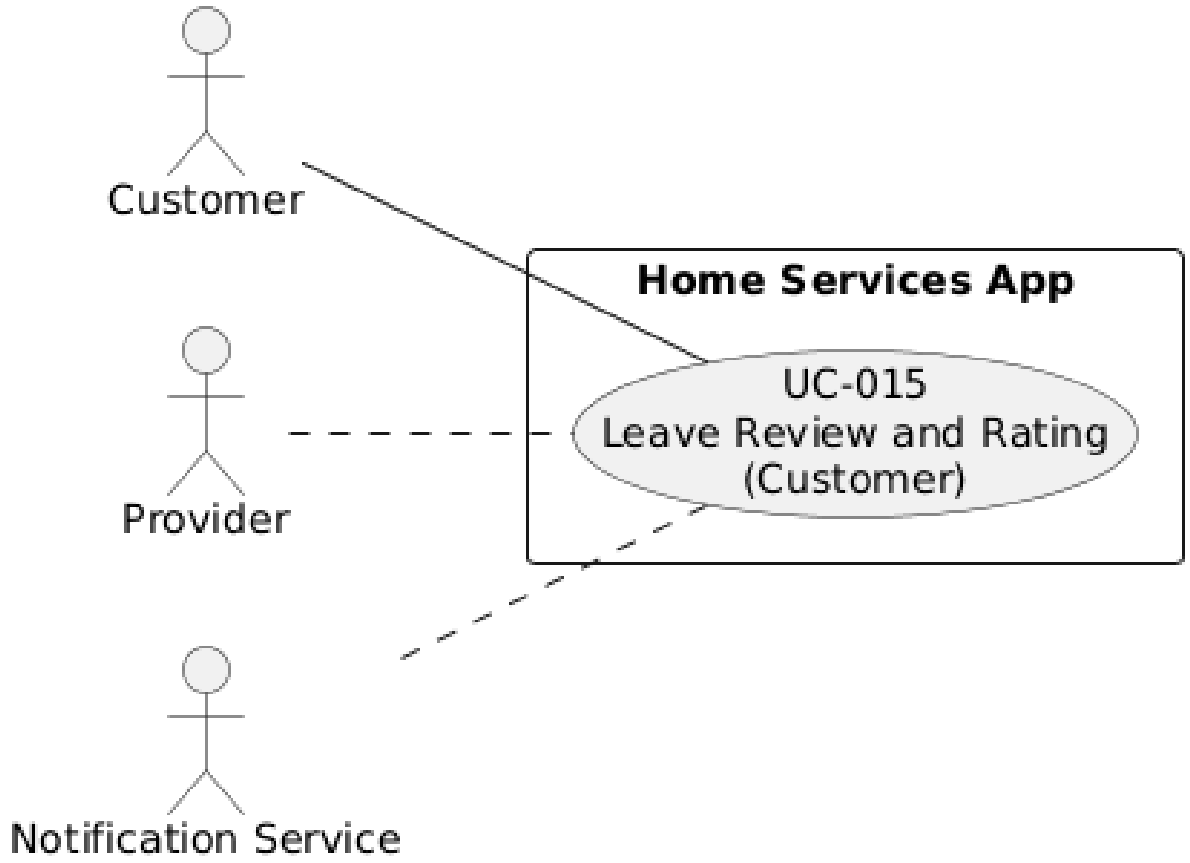

Figure 3.15: Use Case Diagram for UC-015: Withdraw Earnings

Table 3.15: Use Case UC-015: Withdraw Earnings

Use Case ID	UC-015
Title	Withdraw Earnings
Description	This use case allows a service provider to withdraw their available earnings from the Home Services application to a linked bank account, wallet, or other supported payout method.
Actor	Service Provider
Pre-Conditions	• Provider is logged in. • Provider has a verified payout method (e.g., bank account or wallet). • Provider has sufficient available balance in their earnings wallet.
Basic Flow	1. Provider opens the application. 2. Provider navigates to the "Wallet" or "Earnings" section. 3. System displays total earnings, pending amount, and available balance. 4. Provider clicks on the "Withdraw" button. 5. System shows withdrawal form with current payout method and available amount. 6. Provider enters withdrawal amount (less than or equal to available balance). 7. Provider confirms or selects a payout method (if multiple are available). 8. Provider clicks "Confirm Withdrawal". 9. System validates the amount and payout details. 10. System creates a withdrawal request and records it in the database. 11. System sends the payout request to the payment/transfer gateway.

Alternate Flow	• **Insufficient Balance:** If the entered amount is greater than available balance, system shows "Insufficient funds" and prevents request. • **Invalid Amount:** If the amount is zero, negative, or below minimum withdrawal limit, system shows an error and asks for correction. • **No Payout Method Linked:** System prompts provider to add and verify a payout method before continuing. • **Gateway/Bank Error:** If external payout fails, system marks request as "Failed" and restores the amount to available balance. • **Provider Cancels:** Provider cancels the withdrawal before confirmation; no changes are made.
Post-Conditions	• A withdrawal request is created and stored in the system. • Provider's available balance is reduced by the withdrawal amount (or restored if payout fails). • Withdrawal appears in the provider's transaction/withdrawal history.

UC-016: Update Provider Availability

Explains how a provider sets their status to Available, Busy, or Offline so that customers see accurate availability when choosing professionals.

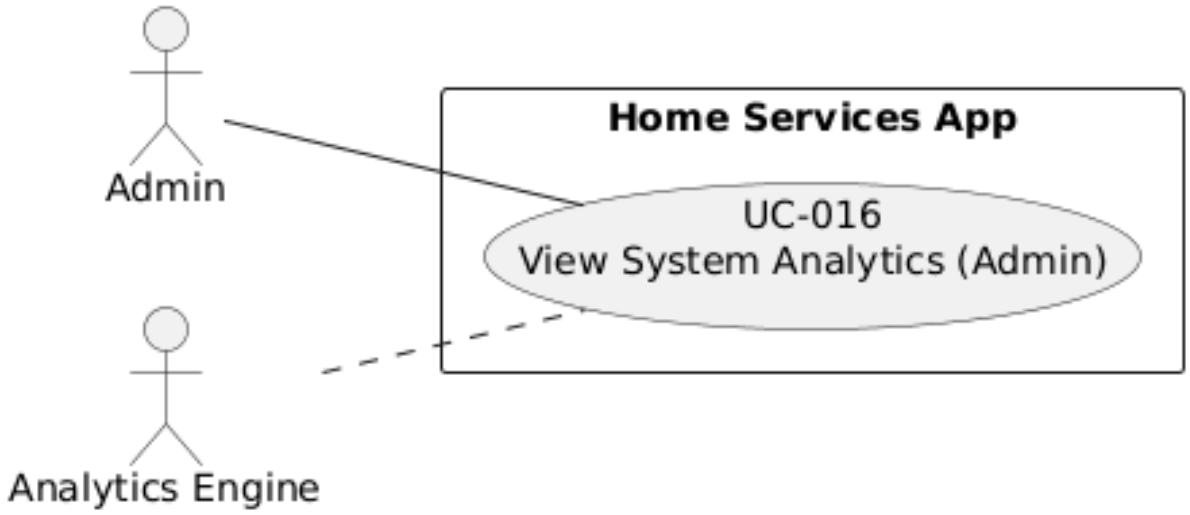

Figure 3.16: Use Case Diagram for UC-016: Update Provider Availability

Table 3.16: Use Case UC-016: Update Provider Availability

Use Case ID	UC-016
Title	Update Provider Availability
Description	This use case allows a service provider to update their availability status (Available / Busy / Offline) in the Home Services application.
Actor	Service Provider
Pre-Conditions	• Provider is logged in. • Provider profile is active and verified.
Basic Flow	1. Provider navigates to the "Profile" or "Availability" section. 2. System displays current availability status. 3. Provider selects a new status (Available, Busy, or Offline). 4. Provider clicks the "Save" or "Update" button. 5. System updates the provider's availability in the database. 6. System reflects the new status in real-time for customers browsing providers. 7. System displays a success message.
Alternate Flow	• **Network Error:** System cannot update status and asks provider to try again. • **Invalid Status:** System resets dropdown and displays an error. • **Provider Cancels:** No changes are saved.
Post-Conditions	• Provider's availability status is updated. • Customers see the updated status when selecting providers.

Table 3.17: Use Case UC-016: Update Provider Availability

UC-017: View System Analytics (Admin)

Describes how an admin opens the analytics dashboard to see key stats like total users, jobs, and revenue, with options to filter data by time period.

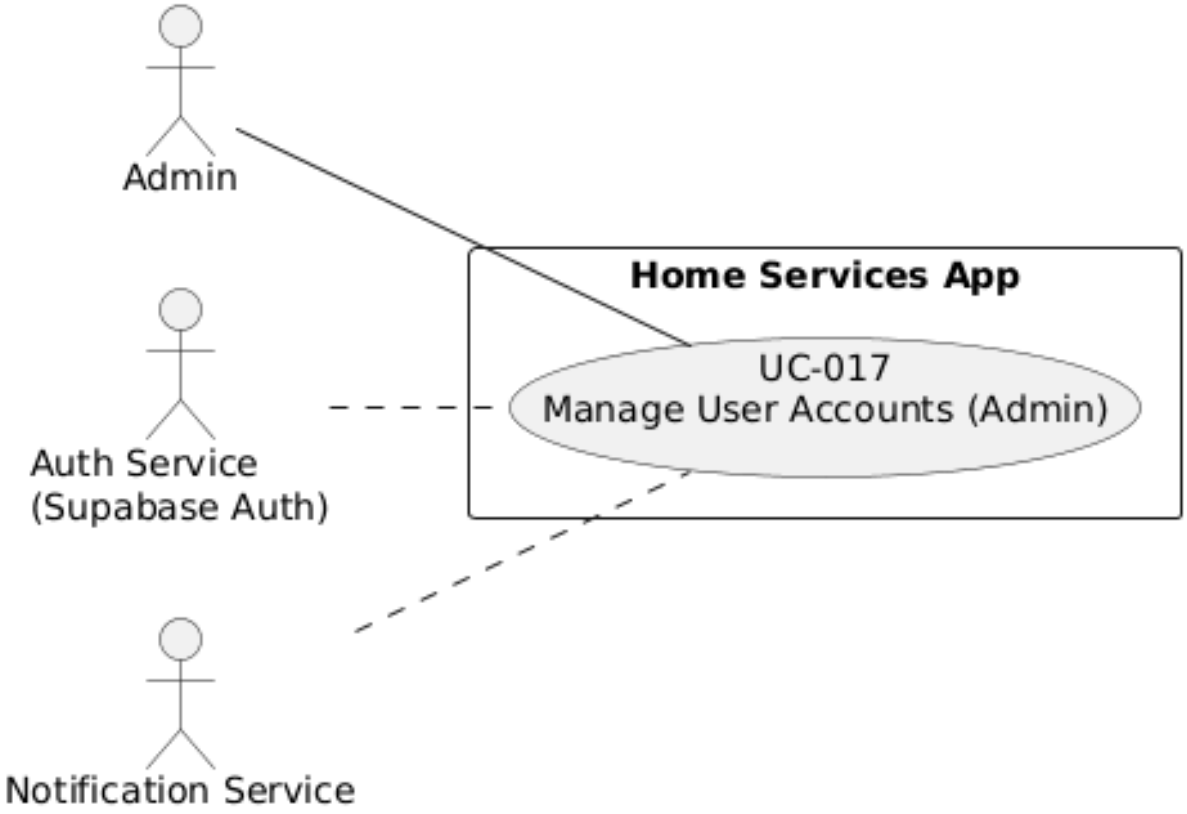

Figure 3.17: Use Case Diagram for UC-017: View System Analytics

Table 3.18: Use Case UC-017: View System Analytics

Use Case ID	UC-017
Title	View System Analytics
Description	This use case allows an admin to view system-wide analytics such as total users, active providers, jobs posted, jobs completed, earnings, and other important platform KPIs.
Actor	Admin
Pre-Conditions	• Admin is logged in. • Admin has permission to access analytics. • Analytics data exists in the system.

Basic Flow	1. Admin logs in to the system.
	2. Admin navigates to the "Admin Dashboard" or "Analytics" section.
	3. System retrieves analytics data from the database.
	4. System displays summary statistics such as:
	  • Total number of users
	  • Total number of service providers
	  • Total jobs posted
	  • Jobs in progress and completed
	  • Total earnings or revenue
	5. Admin applies filters (daily, weekly, monthly, yearly, custom).
	6. System updates analytics charts and tables based on selected filters.
	7. Admin reviews analytics for decision-making and monitoring.

Alternate Flow	• **No Data Available:** System displays "Not enough data to show analytics." • **Permission Denied:** If the user is not an admin, system blocks access. • **Database Error:** System shows "Unable to load analytics. Please try again later."
Post-Conditions	• Admin successfully views platform analytics. • No system data is modified; admin only views analytics.

UC-018: Manage Users (Admin)

Shows how an admin views user lists and profiles and can edit details, verify providers, suspend, reactivate, or delete accounts.

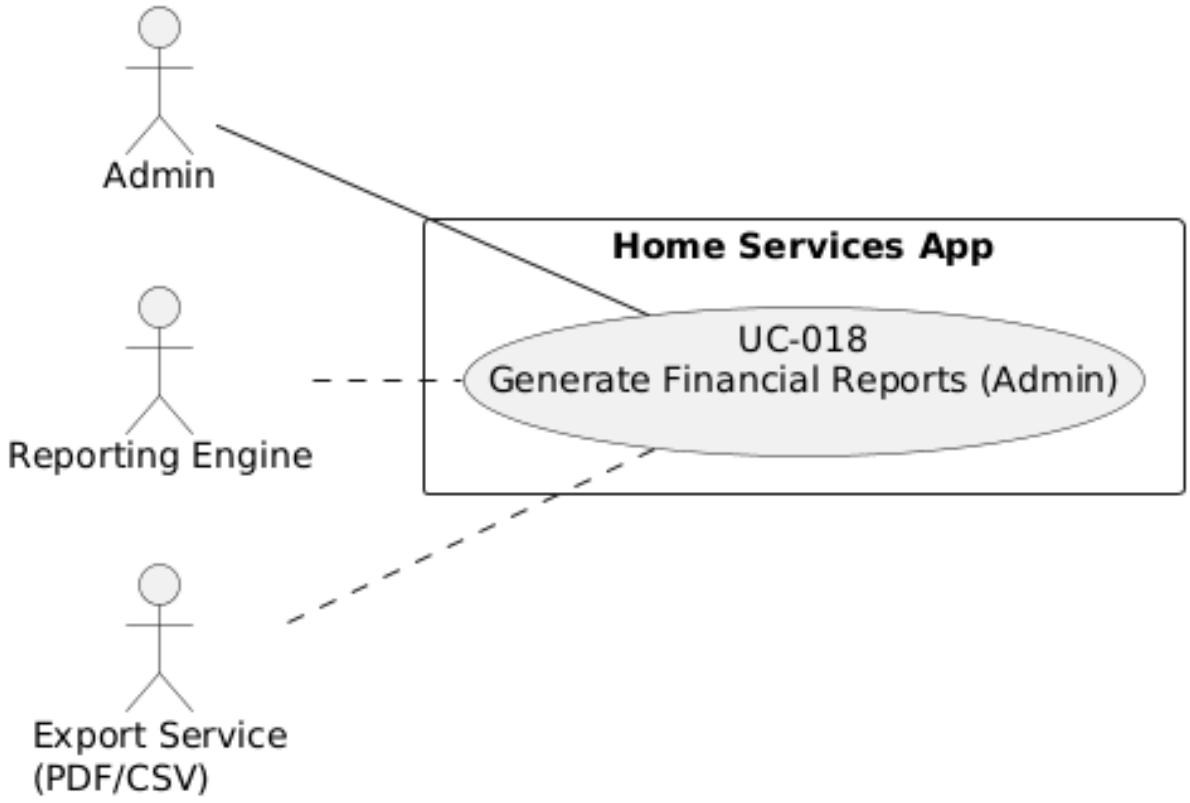

Figure 3.18: Use Case Diagram for UC-018: Manage Users

Table 3.19: Use Case UC-018: Manage Users

Use Case ID	UC-018
Title	Manage Users
Description	This use case allows an admin to manage all users registered on the platform, including viewing, editing, suspending, or removing user accounts.
Actor	Admin
Pre-Conditions	• Admin is logged in. • Admin has appropriate permissions for user management. • User account records exist in the system.

Basic Flow	1. Admin logs into the application.
	2. Admin navigates to the "User Management" section.
	3. System displays a list of all users (Customers and Providers).
	4. Admin selects a specific user to view full profile details.
	5. Admin may choose one of the actions:
	  • View user profile
	  • Edit user information
	  • Suspend or deactivate the user
	  • Reactivate suspended users
	  • Delete user account
	6. Admin confirms the selected action.
	7. System validates the request.
	8. System updates the user's status or information in the database.
	9. System displays a confirmation message.

Alternate Flow	• **Unauthorized Admin:** System blocks access and shows "You do not have permission." • **User Already Suspended:** System prevents duplicate suspension. • **User Not Found:** User may have been deleted earlier. • **Admin Cancels Action:** No changes are made.
Post-Conditions	• Selected updates to user accounts are reflected in the system. • User status or details are updated for customer and provider views.

UC-019: Manage Complaints and Reports

Explains how an admin reviews user complaints, checks evidence, takes actions (warnings, suspensions, resolution), and updates complaint status.

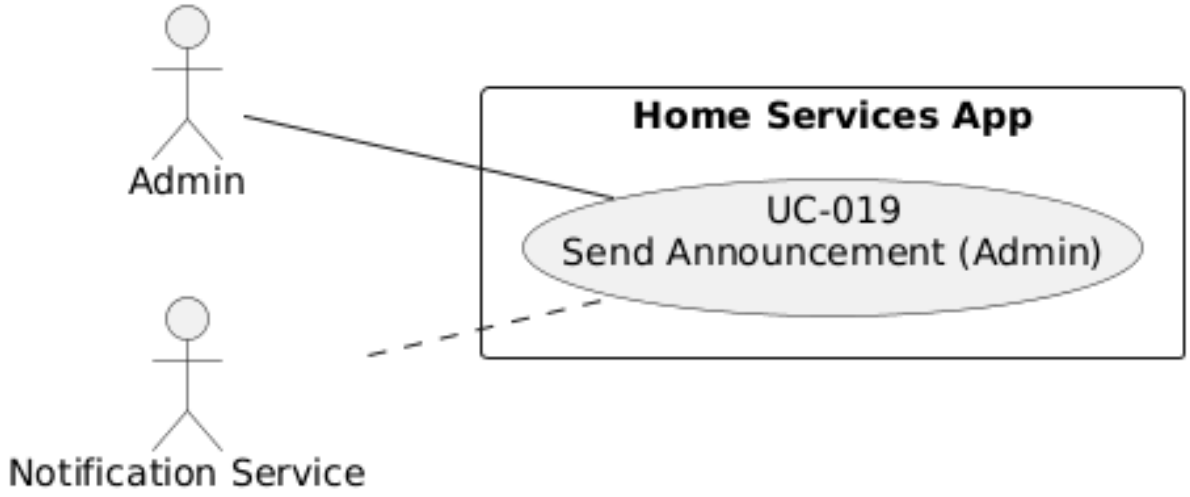

Figure 3.19: Use Case Diagram for UC-019: Manage Complaints and Reports

Table 3.20: Use Case UC-019: Manage Complaints and Reports

Use Case ID	UC-019
Title	Manage Complaints and Reports
Description	This use case allows an admin to view, investigate, and take action on complaints or reports submitted by users regarding jobs, providers, customers, or misconduct on the platform.
Actor	Admin
Pre-Conditions	• Admin is logged in with complaint-management permissions. • At least one complaint or report exists in the system.

Basic Flow	1. Admin navigates to the "Complaints and Reports" section.
	2. System displays a list of all complaints submitted by users.
	3. Admin selects a specific complaint to view details.
	4. System displays complaint information (user, job, type of issue, description, timestamp).
	5. Admin reviews any attached evidence (screenshots, chat logs, etc.).
	6. Admin takes an action:
	  • Mark as resolved
	  • Issue warning to user
	  • Suspend user account
	  • Request additional information
	  • Escalate to higher authority (optional)
	7. Admin confirms selected action.
	8. System logs the action in the admin activity log.
	9. System updates the complaint status (e.g., Resolved, In Review, Escalated).
	10. System notifies the concerned user(s) about the resolution.

Alternate Flow	• **Complaint Already Resolved:** System prevents duplicate actions. • **Invalid Complaint:** If the complaint lacks evidence, admin can mark it as invalid. • **User Account Deleted:** System updates complaint status and prevents actions like warnings. • **Admin Cancels Action:** Complaint remains unchanged.
Post-Conditions	• Complaint is updated to the new status. • All actions are logged for audit purposes. • Users involved are notified about the outcome.

UC-020: Manage Services and Categories

Describes how an admin adds, edits, activates/deactivates, or removes service categories and sub-services that customers use when posting jobs.

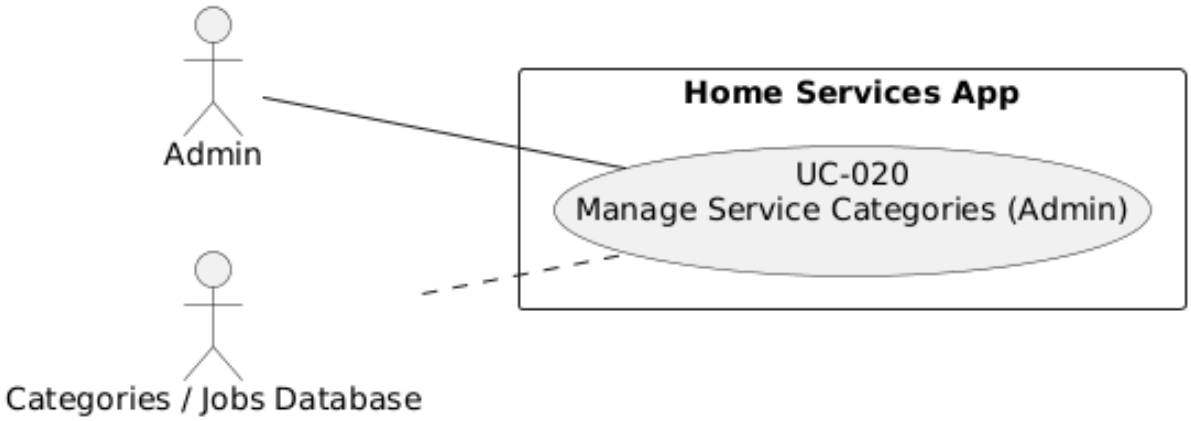

Figure 3.20: Use Case Diagram for UC-020: Manage Services and Categories

Table 3.21: Use Case UC-020: Manage Services and Categories

Use Case ID	UC-020
Title	Manage Services and Categories
Description	This use case allows an admin to manage service categories and sub-categories offered on the platform, including adding, editing, updating, activating, or removing services.
Actor	Admin
Pre-Conditions	• Admin is logged in. • Admin has service-management permissions. • Service and category data exist in the system.

Basic Flow	1. Admin navigates to the "Service Management" or "Categories" section.
	2. System displays existing service categories and sub-services.
	3. Admin selects an action:
	  • Add a new service category
	  • Add a new sub-category
	  • Edit an existing service
	  • Deactivate / activate a service
	  • Delete a service (if not in use)
	4. Admin enters or updates the required details (name, description, icon/image).
	5. Admin clicks "Save" or "Update".
	6. System validates the details.
	7. System saves changes to the database.
	8. System updates the service list for customers and providers.
	9. System displays a success message confirming changes.

Alternate Flow	• **Duplicate Name:** If service/category name already exists, system shows an error. • **Invalid Data:** System rejects blank or invalid entries. • **Category in Use:** Delete is blocked if existing jobs depend on it. • **Admin Cancels Action:** No changes are saved.
Post-Conditions	• The updated service list is stored in the system. • Customers and providers see the updated service categories. • System ensures consistency of job posting and browsing interfaces.

Chapter 4

System Design

This chapter provides an in-depth explanation of the system design for the Home Services App. It includes the system architecture, high-level design, and graphical representations such as sequence diagrams, activity diagrams, and GUI designs. The design ensures that the system meets the functional and non-functional requirements outlined in the previous chapter.

4.1 System Architecture

The Home Services App uses a three-tier architecture, consistent with standard software engineering practices [11]. The architecture is divided into the following layers:

1. **Presentation Layer**
    - This layer represents the user interface through which users interact with the application.
    - It includes the mobile app interface for customers and professionals, as well as the admin dashboard.
    - The presentation layer is developed using Flutter, ensuring a consistent and responsive experience across Android and iOS platforms.
2. **Business Logic Layer**
    - This layer contains the core functionalities and processes of the application.
    - It handles tasks such as processing job requests and bids, managing user authentication and verification, AI-powered problem diagnosis and price estimation, and real-time notifications and messaging.
    - The business logic layer is implemented using Supabase Functions and AI/ML models.
3. **Data Layer**
    - This layer is responsible for managing and storing data.
    - It includes a Supabase Database for real-time data management, storing user profiles, job requests, bids, and payment transactions.
    - The data layer ensures secure and efficient data retrieval and storage.

4.2 System Architecture Diagram

The system architecture diagram illustrates the overall system architecture of the proposed mobile-based service platform, redesigned with Supabase as the backend instead of Firebase. The system consists of multiple Flutter-based frontend applications, including a Customer App, a Professional App, and an Admin Web Panel. Each interface serves a distinct role: customers use the app to browse services, place bookings, make payments, and submit feedback; professionals manage job requests, share real-time location, and track their work; while administrators oversee user management, service approvals, and system monitoring through the web panel.

At the core of the architecture is the Supabase Backend, which functions as the central communication and data management layer. All frontend applications interact with Supabase rather than directly accessing databases or external services. Supabase ensures secure authentication, real-time data synchronization, and controlled access to system resources. This centralized backend design improves scalability, security, and consistency across all user roles.

The system relies on Supabase services to handle essential backend operations. Supabase Authentication (OTP) is used for secure user login and identity verification through phone numbers or email-based one-time passwords. Supabase Database (PostgreSQL) stores structured data such as user profiles, bookings, service records, feedback, and ratings. Supabase Realtime enables live updates for booking status, job assignments, and location sharing. Supabase Storage is used to store files such as profile images, verification documents, and service-related media. Additionally, Supabase Edge Functions manage server-side logic, automate workflows, and act as secure connectors to external APIs.

An integrated AI Engine is connected to the system to provide an AI-based chat assistant that helps users with service selection, diagnostics, and price estimation. The AI engine communicates with the backend via secure APIs and Edge Functions, ensuring that sensitive data and authentication remain protected. This component enhances user experience by delivering instant, intelligent support within the application.

The architecture also incorporates third-party APIs to extend functionality. A Payment Gateway API handles secure online transactions, while the Google Maps API enables real-time location tracking and navigation for professionals. An SMS/Email OTP service supports verification and notification delivery. All third-party integrations are routed through Supabase Edge Functions, ensuring secure data exchange and proper access control.

Overall, the Supabase-based system architecture presents a modular, scalable, and secure design, where Flutter applications form the frontend, Supabase provides a powerful backend with real-time capabilities, AI adds intelligent assistance, and third-party APIs enable practical, real-world features. This layered architecture ensures efficient data flow, real-time interaction, and a seamless experience for customers, professionals, and administrators alike.

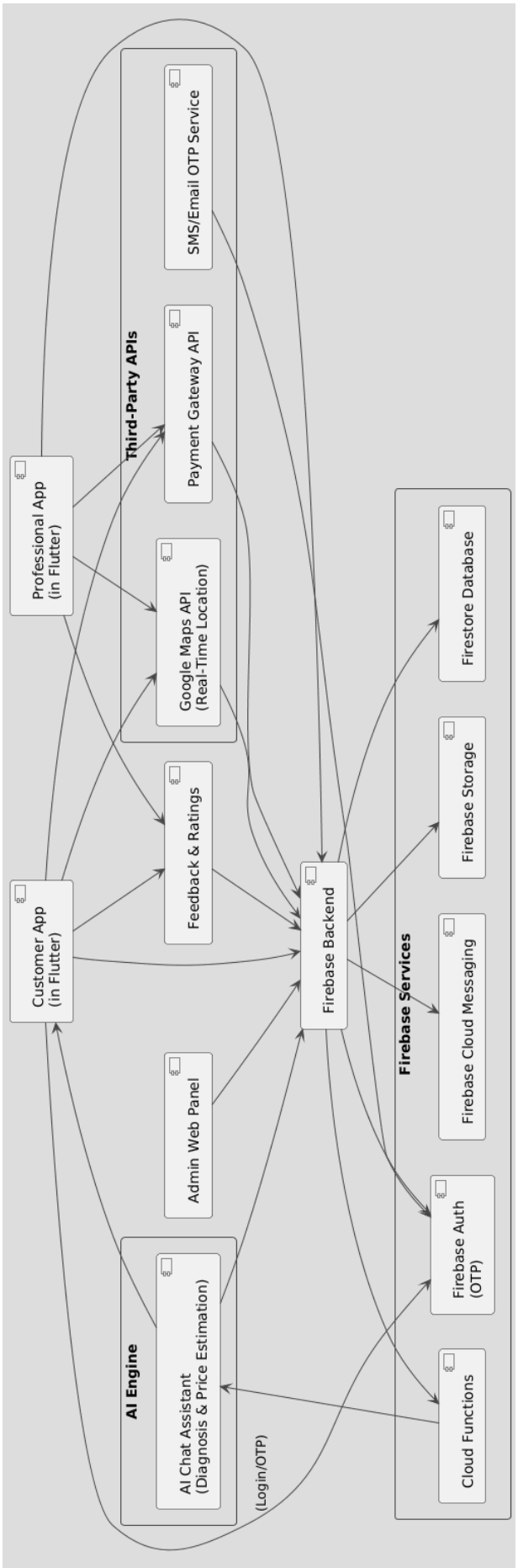

Figure 4.1: System Architecture Diagram

4.3 High-Level Design

The high-level design provides an overview of how the system components interact to deliver the required functionalities. It includes sequence diagrams, activity diagrams, and use case workflows.

4.3.1 Sequence Diagrams

The sequence diagrams illustrate the interactions between users (customers, professionals, and admins) and the system for each key use case. They show the order of operations and message flows.

Sequence Diagram for UC-001: User Registration

This sequence diagram illustrates the user registration process for a Home Services App, which involves three main stages: Open Registration, Fill and Submit Form, and Email Verification. The process coordinates interactions between the Guest User, the Home Services App, an Auth Service (Supabase Auth), a Database (profiles), and an Email Service to create an account, store the user profile, and verify the user's email address.

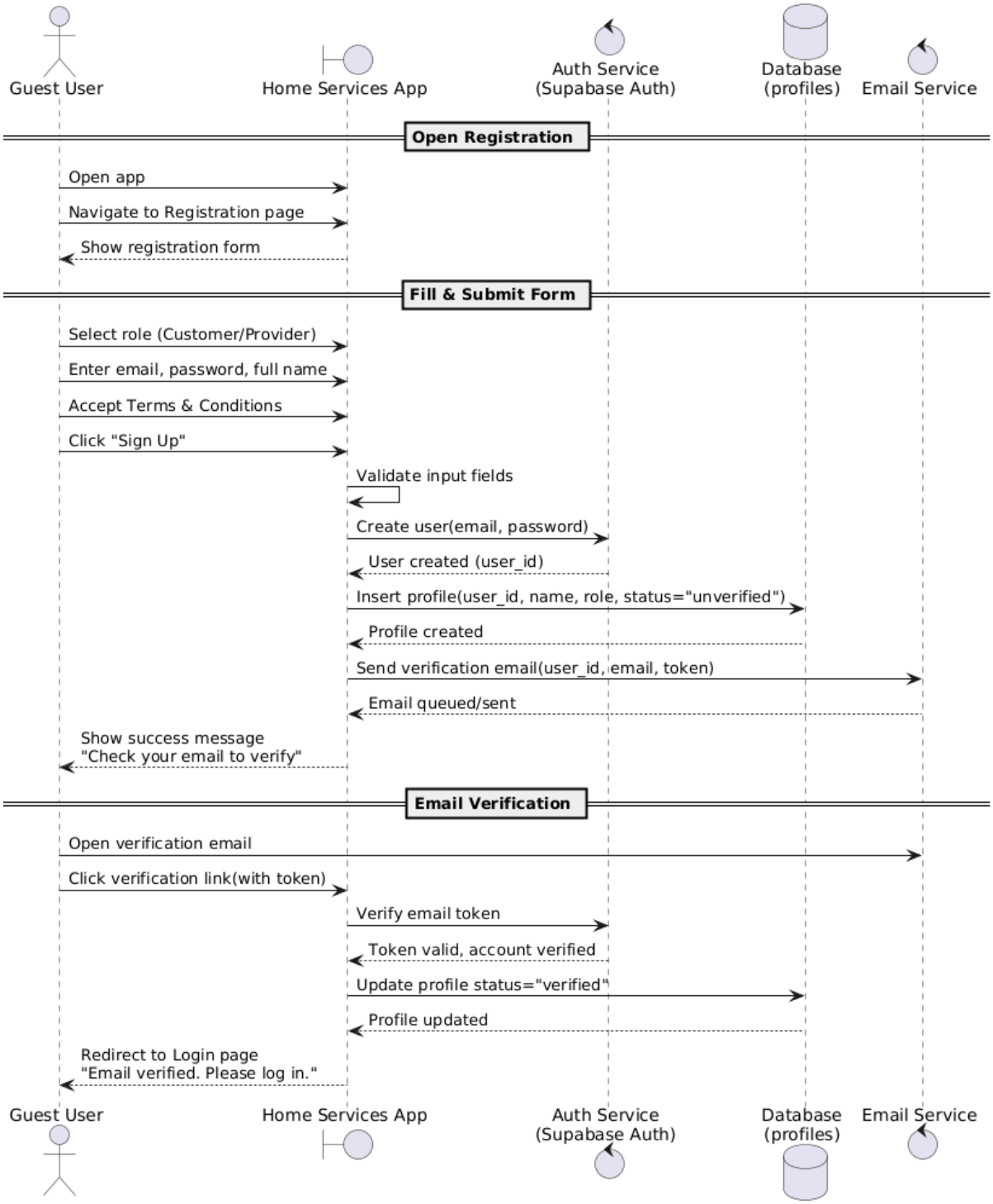

Figure 4.2: Sequence Diagram for UC-001: User Registration

Sequence Diagram for UC-002: User Login

This sequence diagram models the user login process for the Home Services App, detailing the interactions when a user attempts to access the application. The process begins with Open Login and proceeds to Submit Credentials, where the Registered User provides their email and password to the Home Services App. The app delegates authentication

to the Auth Service (Supabase Auth). Following successful authentication, the app fetches the user's profile from the Database to determine the user's role and verification status, subsequently either redirecting them to a role-based dashboard, prompting them to verify their email, or showing an error if login failed.

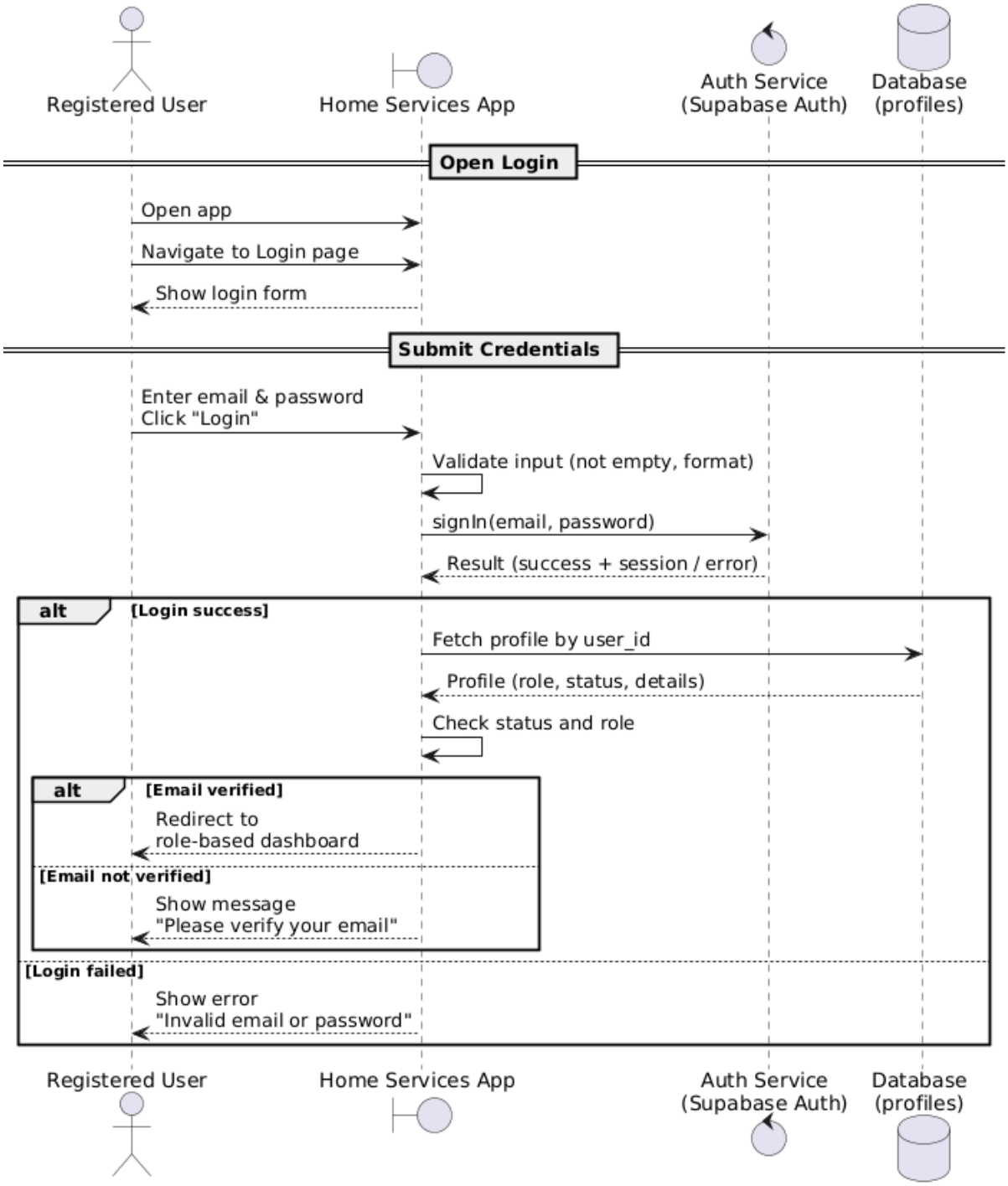

Figure 4.3: Sequence Diagram for UC-002: User Login

Sequence Diagram for UC-003: User Logout

This sequence diagram details the user logout process for the Home Services App, which is executed in two main stages: Initiate Logout and Invalidate Session. The process begins when a Logged-in User clicks the "Logout" button, prompting the Home Services App to visually prepare for the action and then instruct the Auth Service (Supabase Auth) to signOut(). Upon a successful result, the app clears all local session data and redirects the user to the Login / Welcome screen; if logout fails, an error message is shown to the user.

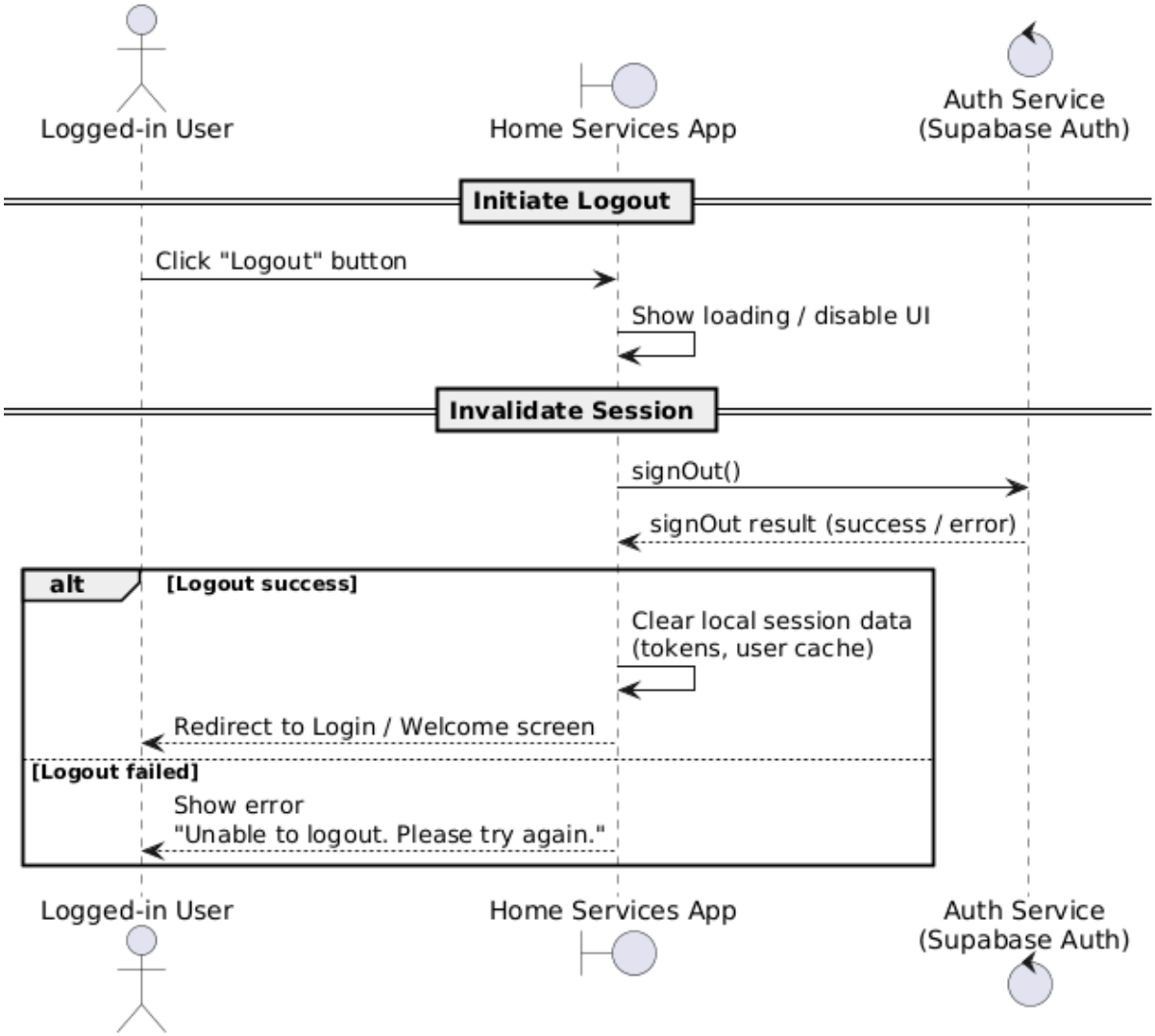

Figure 4.4: Sequence Diagram for UC-003: User Logout

Sequence Diagram for UC-004: Update User Profile

This sequence diagram illustrates the process of a Logged-in User viewing and updating their profile, which is broken down into two stages: Open Profile Page and Edit and Submit Profile. Initially, the Home Services App fetches existing profile data from the Database (profiles) to display to the user. When the user submits changes, the App validates the input; if valid, it checks for an avatar update, uploading any new image

to the Storage Service (Supabase Storage) before persisting all profile changes (including the new avatar URL, if applicable) in the Database, and finally confirming the successful update to the user.

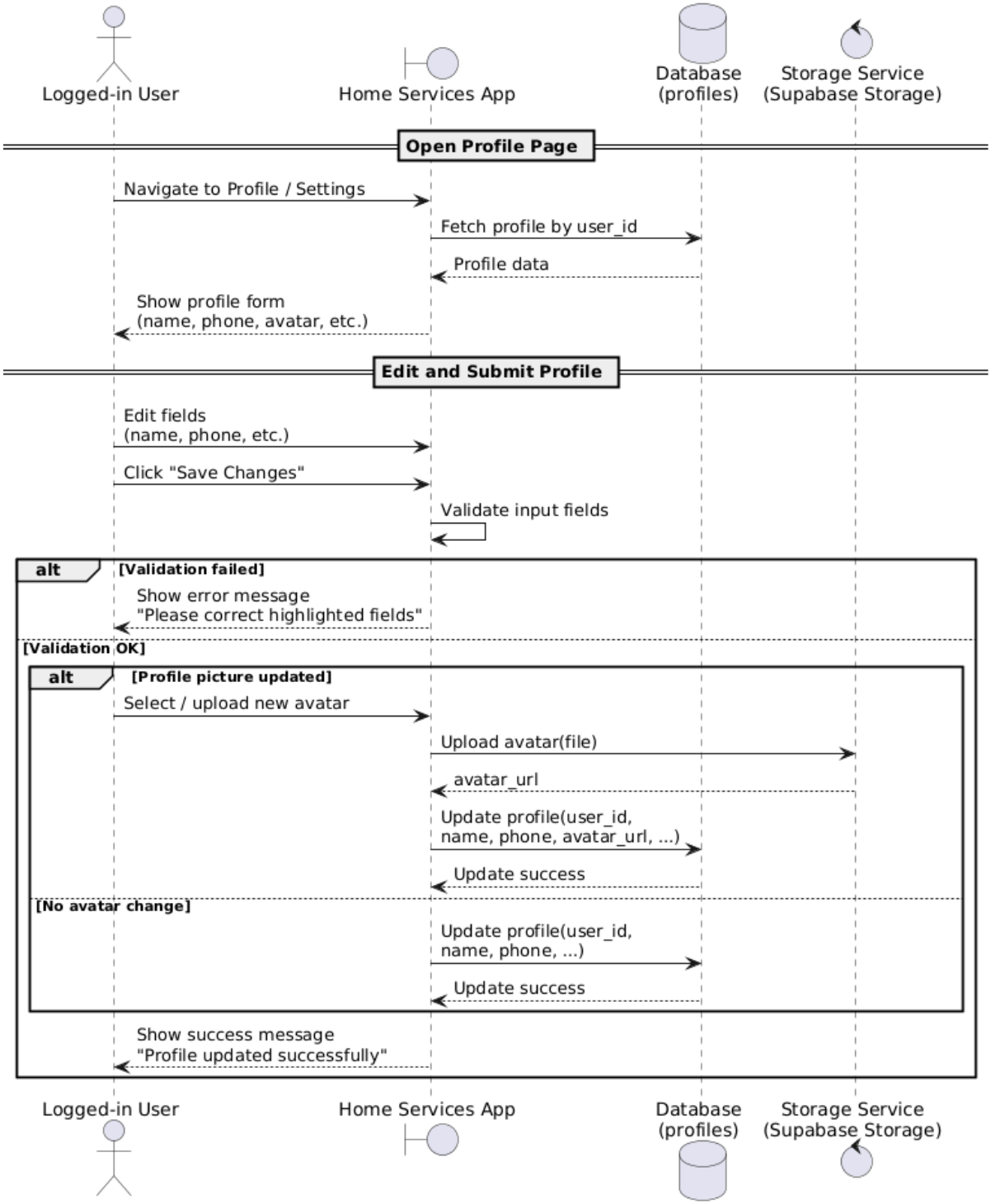

Figure 4.5: Sequence Diagram for UC-004: Update User Profile

Sequence Diagram for UC-005: Post a Job (Customer)

This sequence diagram outlines the process for a Customer to post a new job within the Home Services App, spanning two main phases: Open Post Job Screen and Fill Details. First, the Home Services App fetches available job categories from the Database and displays the job form. Once the user submits the job details, the App validates the input; if successful, it checks for attached media, uploads any files to the Storage Service (Supabase Storage), and then inserts the complete job record (including media URLs) into the Database with an "Open" status, finally confirming the successful posting to the user.

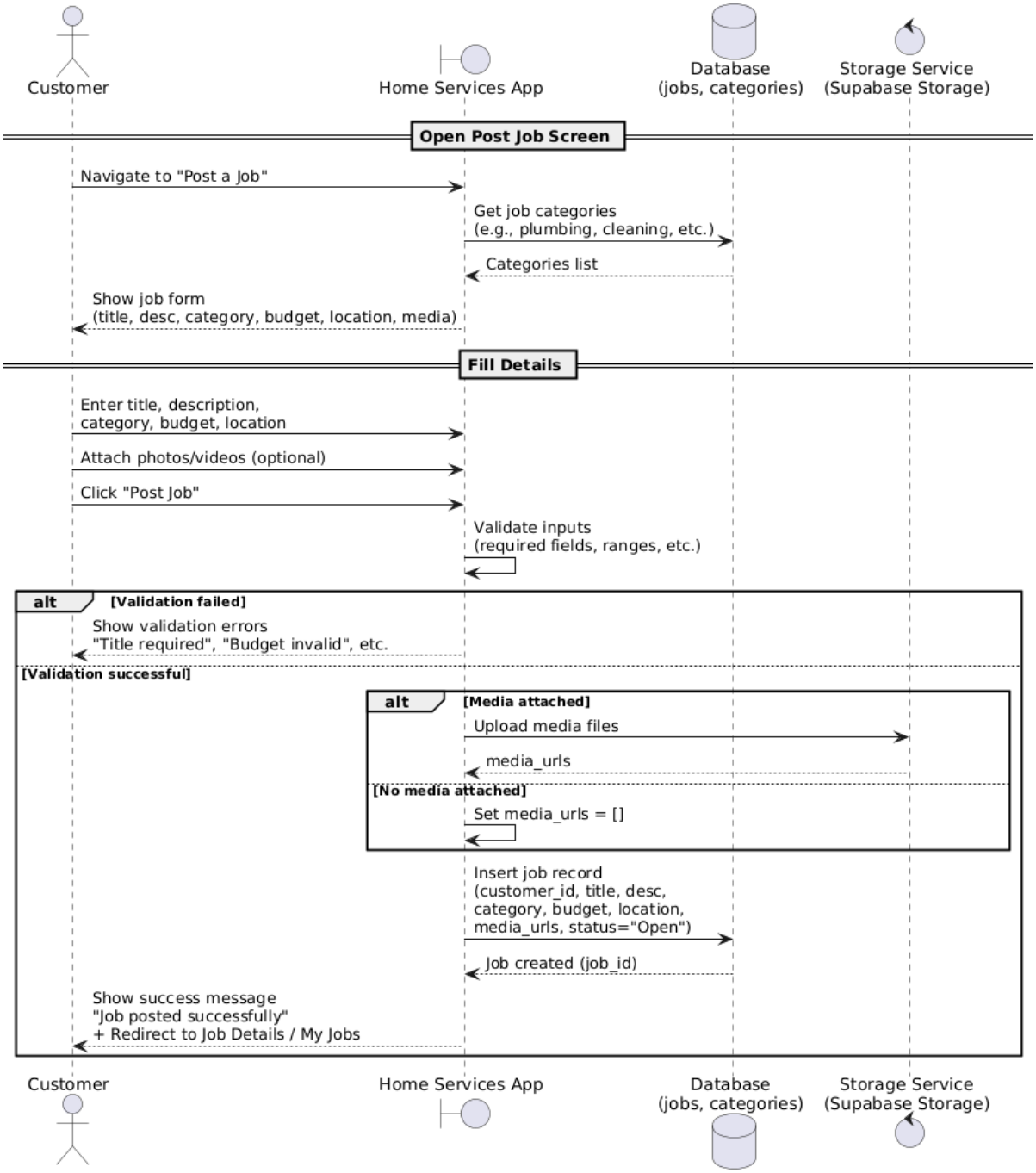

Figure 4.6: Sequence Diagram for UC-005: Post a Job (Customer)

Sequence Diagram for UC-006: Browse Available Jobs (Provider)

This sequence diagram illustrates how a Provider browses and views available jobs on the Home Services App, detailing the process in three main stages: Open Browse Jobs, Apply Filters / Search, and View Job Details (Optional). Initially, the Home Services App fetches job categories and a default list of open jobs from the Database for display. When the Provider applies filters or a search, the App queries the Database to fetch the filtered jobs list matching the criteria. Finally, if the Provider selects a job, the App

retrieves the specific job details by job id from the Database to show the details screen, which includes an option to bid.

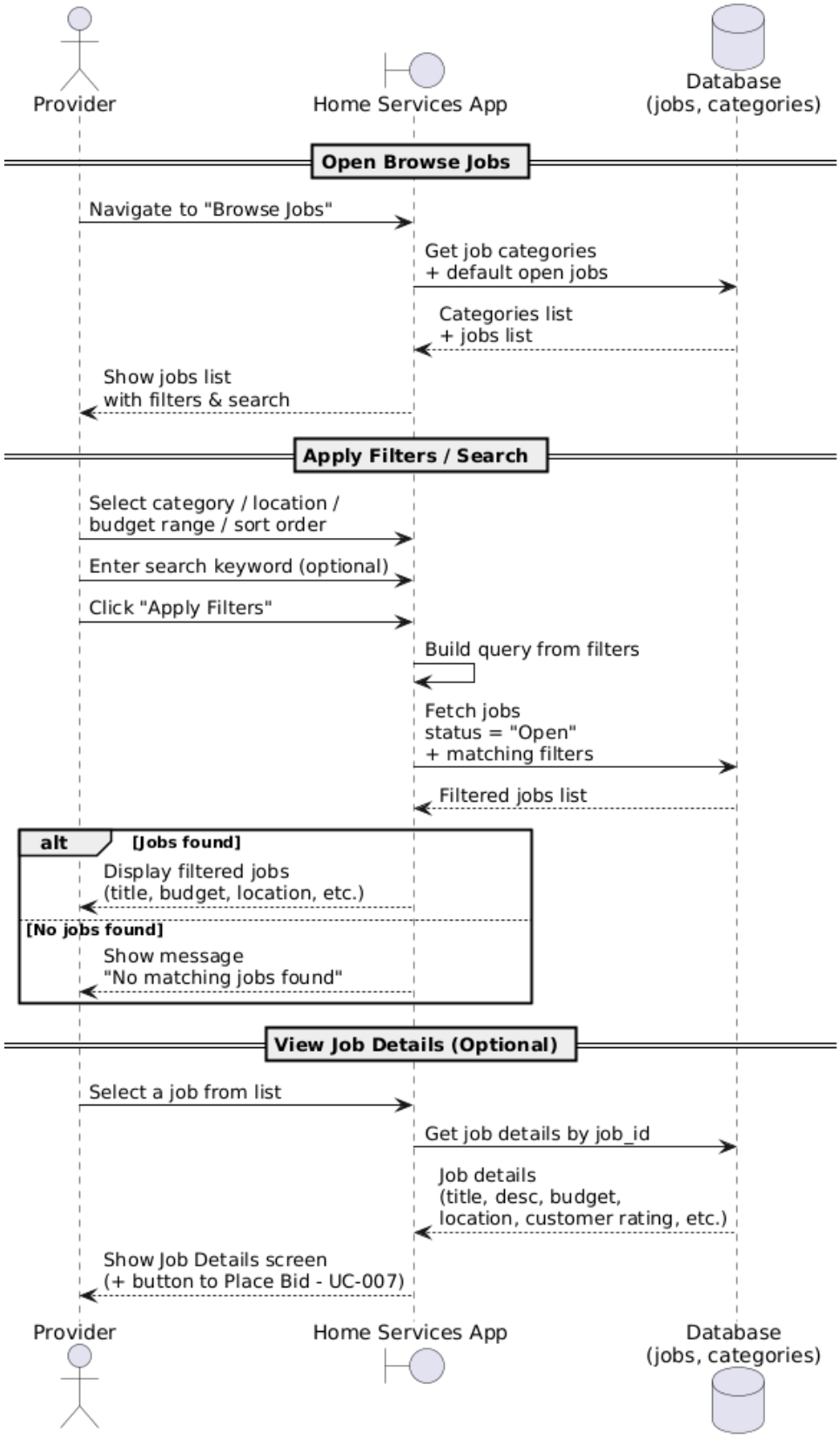

Figure 4.7: Sequence Diagram for UC-006: Browse Available Jobs (Provider)

Sequence Diagram for UC-007: Place Bid on Job (Provider)

This sequence diagram illustrates the process of a Provider placing a bid on an open job, which involves two primary phases: Open Job Details and Start Bid. First, when the Provider opens the Job Details screen, the Home Services App fetches the current job details from the Database (jobs, bids). When the Provider clicks "Place Bid" and submits the bid form, the App validates the input to ensure the job is still available and the bid details are valid. If successful, the App inserts the new bid record with a "Pending" status and updates the job's activity timestamp in the Database, then triggers the Notification Service to inform the customer that a new bid has been received.

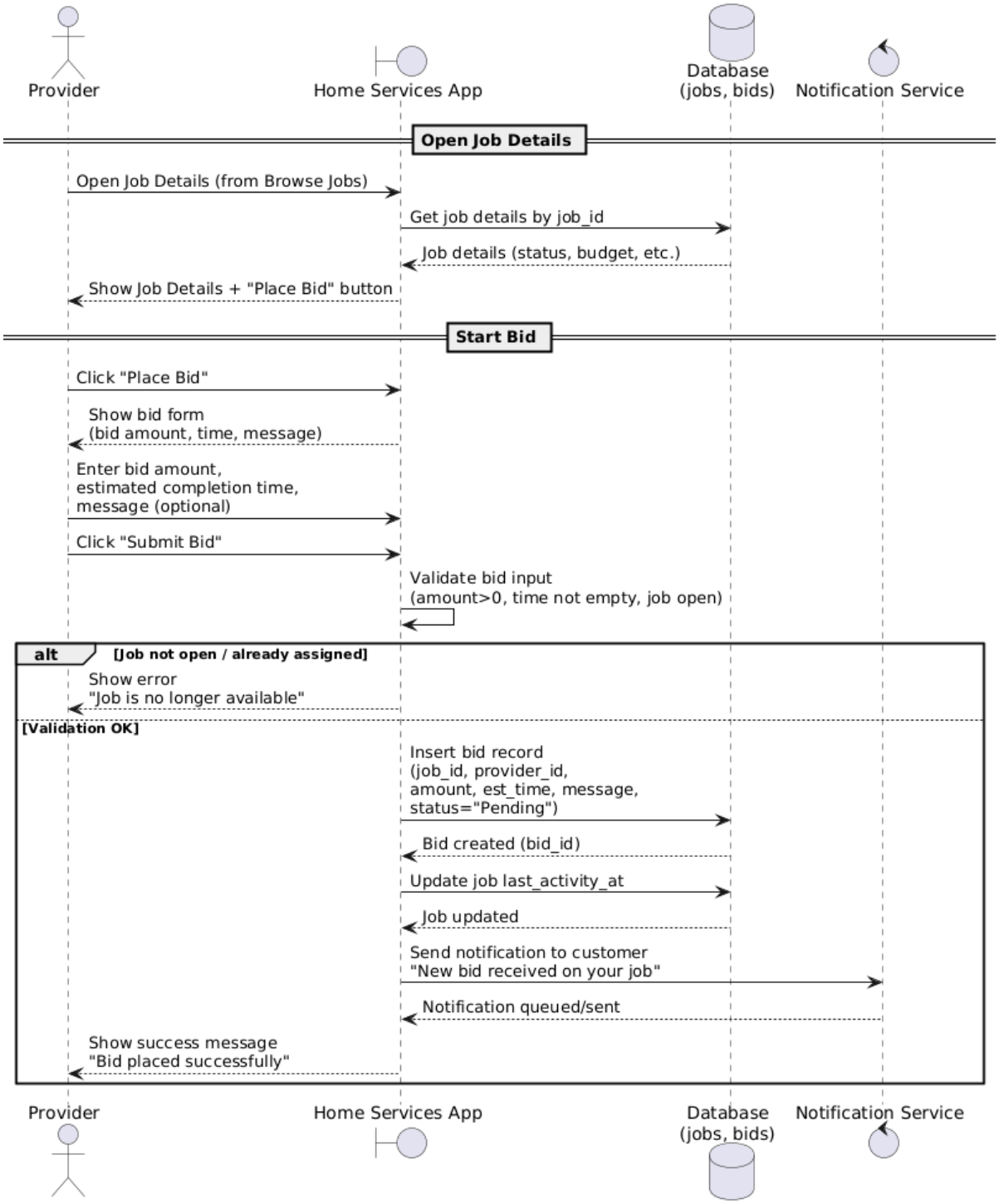

Figure 4.8: Sequence Diagram for UC-007: Place Bid on Job (Provider)

Sequence Diagram for UC-008: View and Compare Bids (Customer)

This sequence diagram details how a Customer manages and reviews bids for their posted jobs, encompassing four sequential stages: Open My Jobs, Open Job Bids, Sort / Filter / Compare, and Next Step (Optional). Initially, the Home Services App retrieves a list of the Customer's jobs from the Database. Upon selecting a job to view bids, the App

fetches the job details and any associated pending bids. The Customer can then sort, filter, or compare bids by selecting one or more, which triggers the App to fetch the corresponding provider profiles from the Database, ultimately leading to the optional Next Step of accepting a bid.

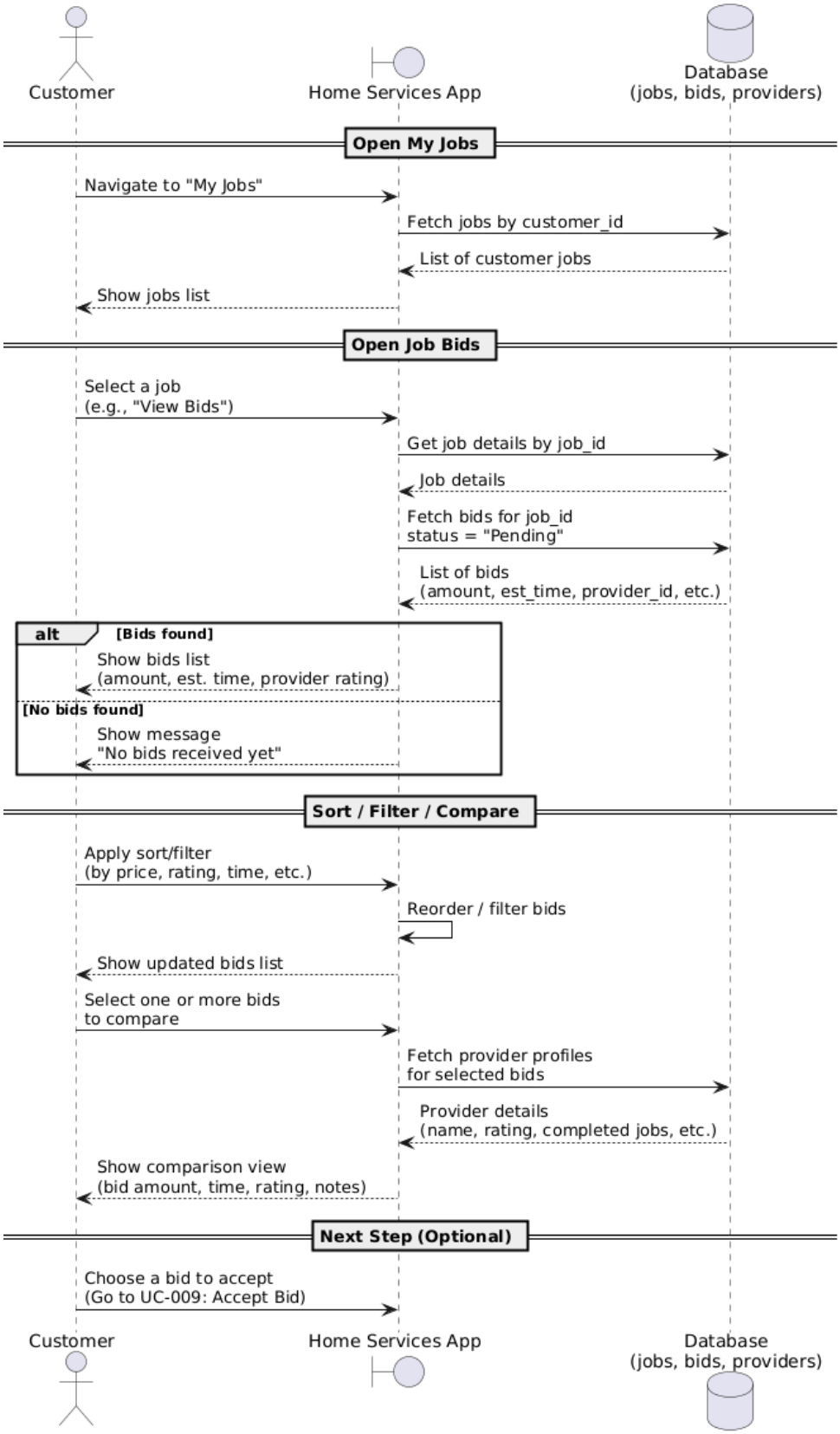

Figure 4.9: Sequence Diagram for UC-008: View and Compare Bids (Customer)

Sequence Diagram for UC-009: Assign Job to Provider (Customer)

This sequence diagram illustrates the process of a Customer accepting a bid for a job, which is divided into four stages: Open Bids for a Job, Select Bid to Accept, Update Bid and Job Status, and Notify Provider and Customer. After the Customer opens the bid list and confirms the selection of a bid, the Home Services App first verifies the job's current status from the Database. If the job is still open, the App proceeds to update the selected bid to "Accepted", updates all other bids for that job to "Rejected", and changes the job's status to "Assigned" along with the assigned provider id in the Database. Finally, the App uses the Notification Service to inform both the Provider and the Customer about the acceptance and assignment.

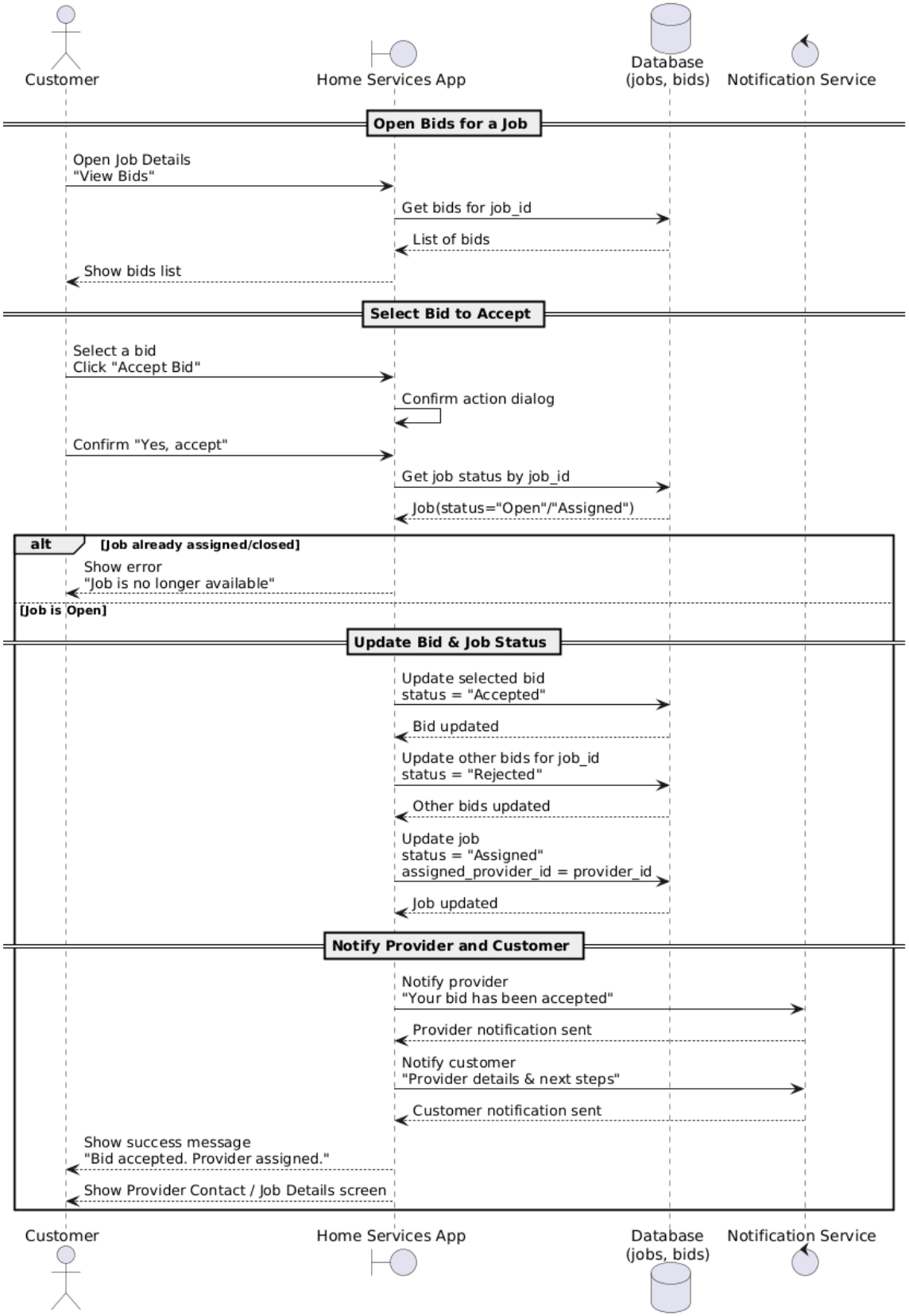

Figure 4.10: Sequence Diagram for UC-009: Assign Job to Provider (Customer)

Sequence Diagram for UC-010: Update Job Status (Provider)

This sequence diagram details the process for a Provider to submit a progress update on an assigned job, structured into three main phases: Open Assigned Job, Start Progress Update, and Submit Update. Initially, the Home Services App fetches the Provider's assigned jobs and specific job details from the Database. When the Provider starts the update, they can enter notes and optionally add up to five photos, which the App validates for format and size. Upon submitting, the App ensures notes are present, then uploads any valid photos to the Storage Service (Job Images), inserts a new job progress record into the Database, increments the job's progress count, and finally sends a notification via the Notification Service to inform the customer of the update.

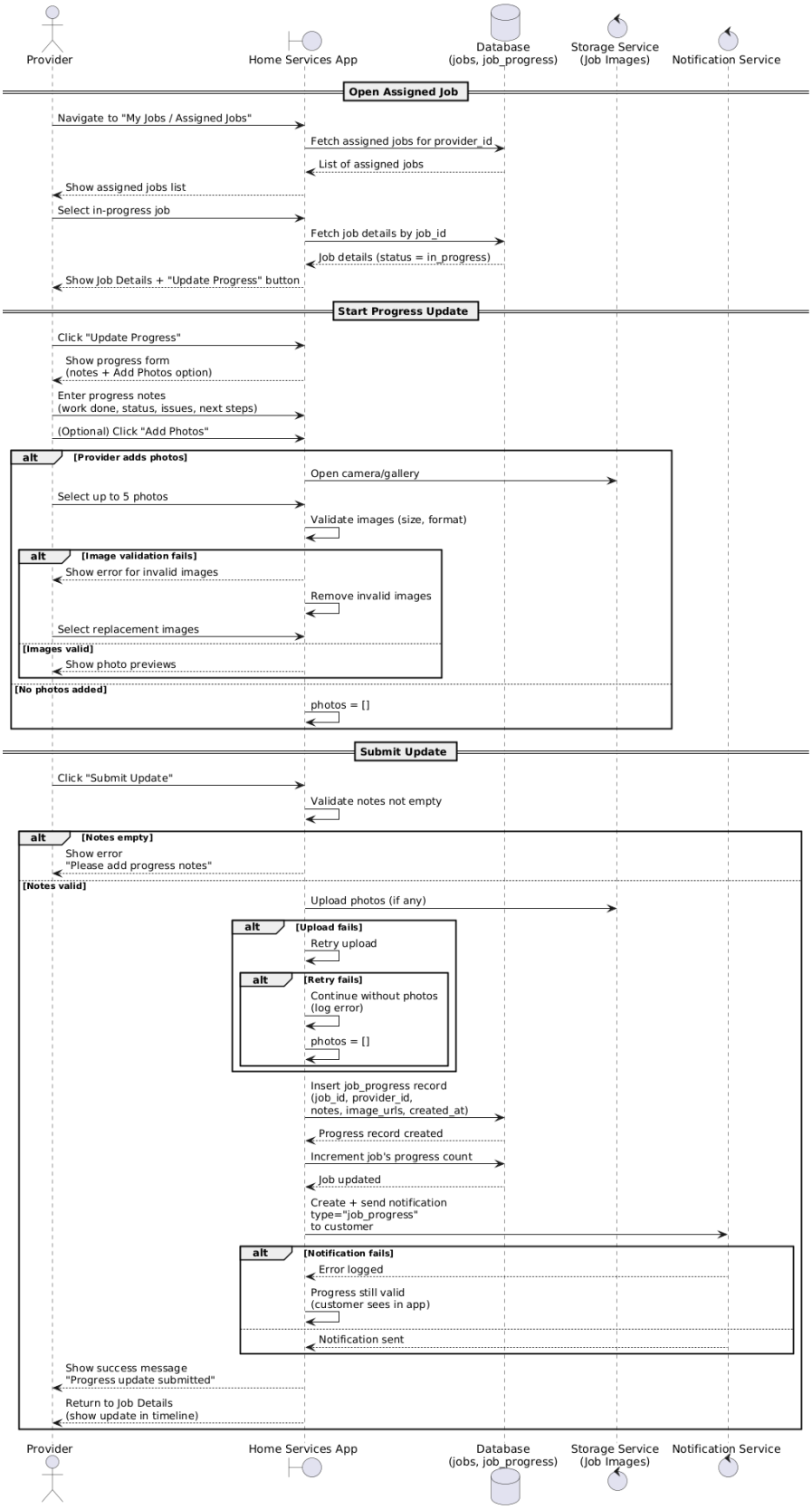

Figure 4.11: Sequence Diagram for UC-010: Update Job Status (Provider)

Sequence Diagram for UC-011: Cancel Job (Customer)

This sequence diagram depicts the process for a Provider to mark an assigned job as complete, structured into three main phases: Open Assigned Job, Start Completion Flow, and Submit Completion. First, the Home Services App displays the Provider's assigned jobs and specific job details fetched from the Database. When the Provider clicks "Mark as Completed," a form is shown for completion notes and optional final photos. Upon submission, the App validates the notes and checks the job status. If valid, it uploads photos to the Storage Service (Job Images), inserts a completion record in the Database, updates the job status to "Completed - Pending Confirmation," and sends a notification via the Notification Service to the customer that the job is complete and awaiting their confirmation.

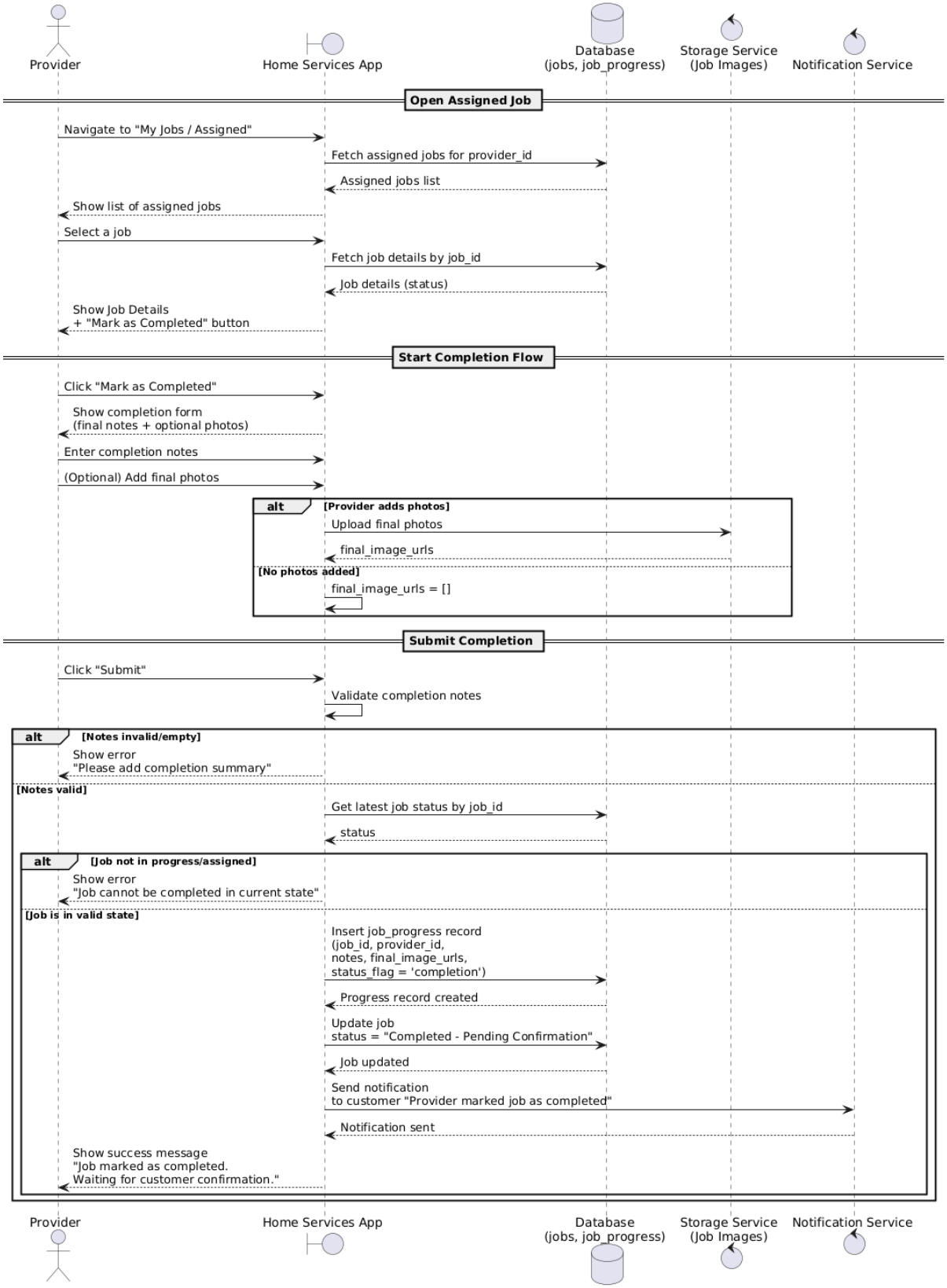

Figure 4.12: Sequence Diagram for UC-011: Cancel Job (Customer)

Sequence Diagram for UC-012: Chat Between Customer and Provider

This sequence diagram illustrates the process of a Provider recording a cash payment received for a completed job, which is divided into three main phases: Open Cash Payment Screen, Fill and Submit Form, and the resulting update sequence. Initially, the Home Services App fetches a list of the Provider's completed jobs from the Database. When the Provider submits the payment form, the App validates the input and confirms the job hasn't already been paid. If valid and not yet paid, the App handles the optional receipt photo upload to Receipts Storage, inserts a payment record into the Database, updates the job's payment status to "Paid (Cash)", and sends notifications to both the Customer and the Admin/Finance department via the Notification Service.

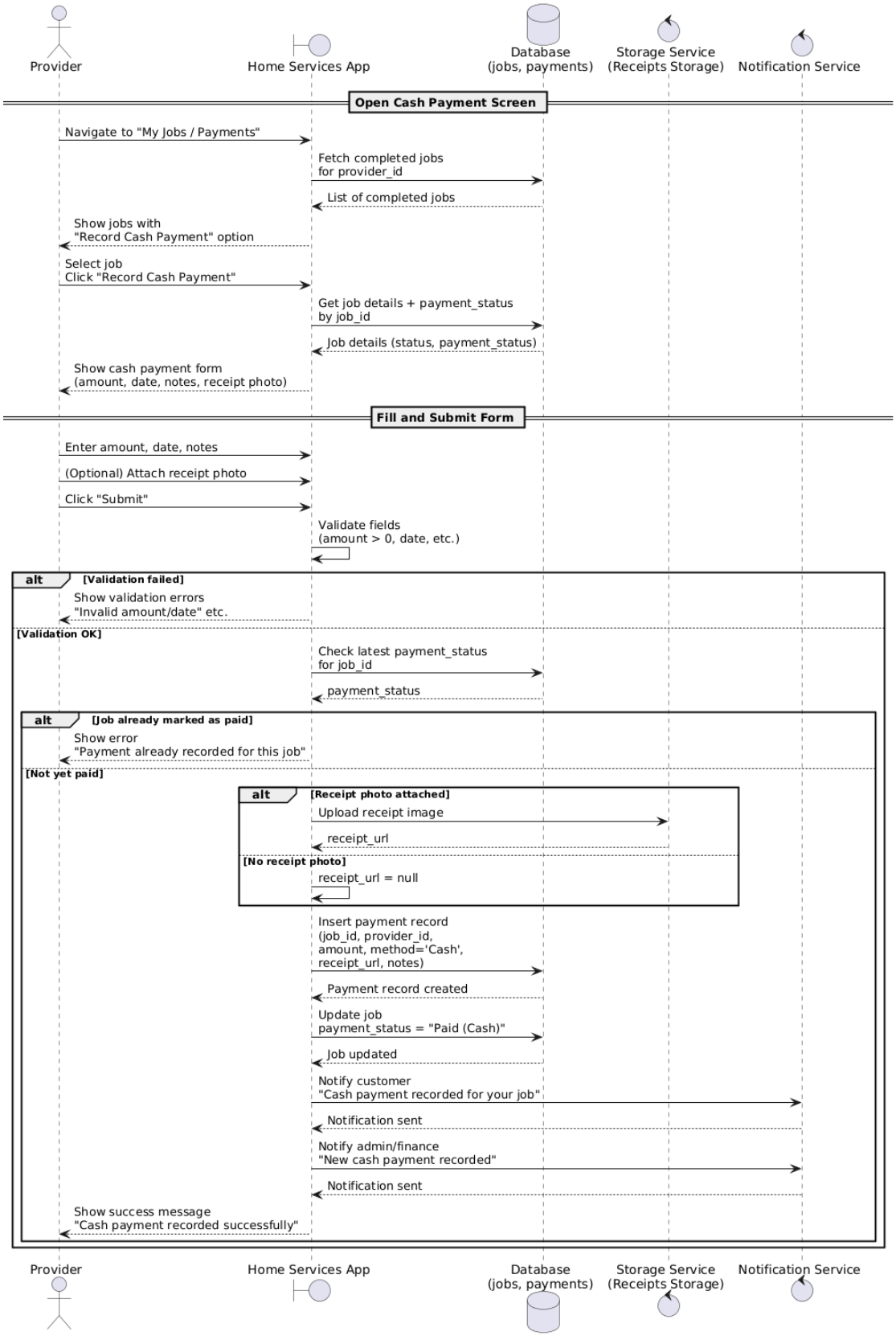

Figure 4.13: Sequence Diagram for UC-012: Chat Between Customer and Provider

Sequence Diagram for UC-013: Rate and Review Provider (Customer)

This sequence diagram illustrates the Admin-managed process for refunding a payment, which is executed in four main stages: Open Refund Management, Select Payment to Refund, Call Stripe Refund API, and Notify Customer and Provider. An Admin initiates the process by viewing payments and selecting one to refund. The Home Services App (Admin Panel) validates refund eligibility and the requested amount before calling the Stripe API (Payment Gateway) to process the refund. Upon success, the App updates the refund and payment records in the Database, sets the job status to "Refunded" or "Partially Refunded", and triggers the Notification Service to inform both the Customer and the Provider.

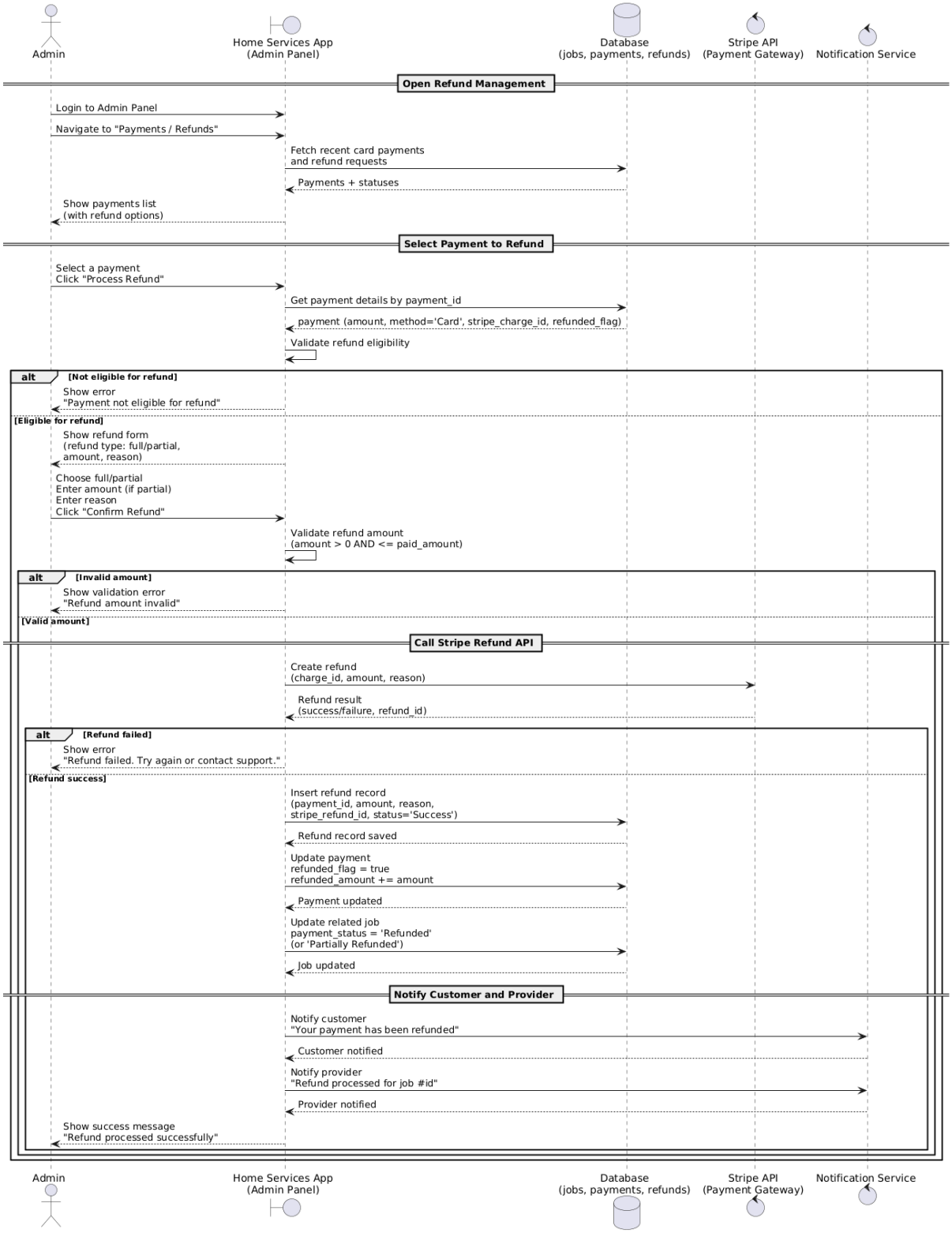

Figure 4.14: Sequence Diagram for UC-013: Rate and Review Provider (Customer)

Sequence Diagram for UC-014: Payment Processing

This sequence diagram illustrates the real-time chat and messaging process between a Customer or Provider within the Home Services App, detailing the two main phases: Open Chat and Type and Send Message. Initially, the Home Services App fetches the user's conversation list and message history for a selected chat from the Database. When a user sends a message, the App validates it, inserts the message record into the Database, publishes a "new message" event through the Realtime Service (Supabase Realtime) for instant delivery to other online clients, and simultaneously triggers the Notification Service to send a push notification to the receiver.

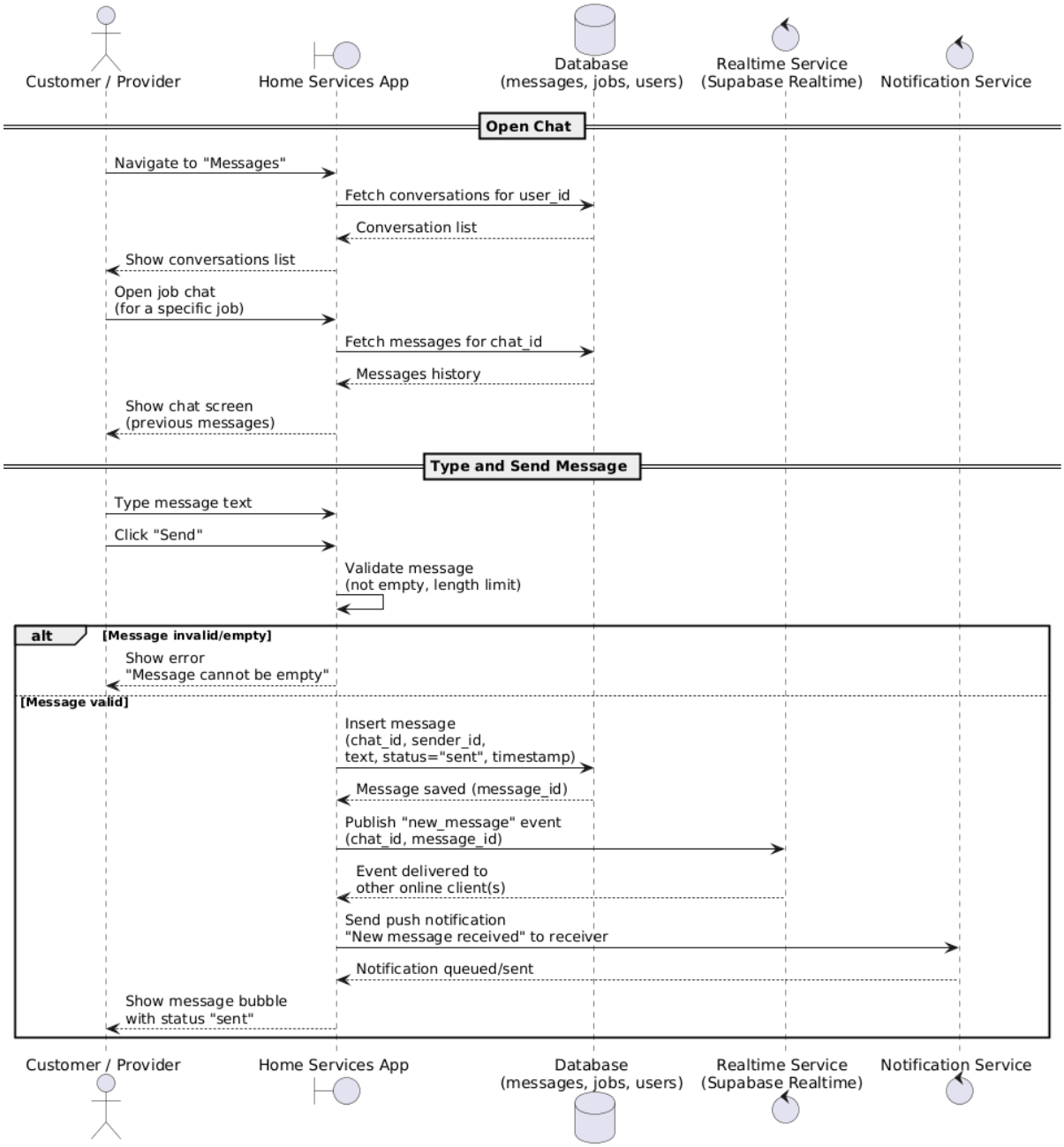

Figure 4.15: Sequence Diagram for UC-014: Payment Processing

Sequence Diagram for UC-015: Withdraw Earnings (Provider)

This sequence diagram details the process for a Customer to submit a review and rating for a completed job, which is organized into four sequential phases: Open Completed Jobs, Fill Review Form, Update Provider Overall Rating, and Notify Provider. After the Customer opens their "My Jobs" list and selects a completed job (and no review exists yet), they can fill out the form by selecting a rating (1-5 stars) and an optional comment. Upon valid submission, the Home Services App inserts the new review record and updates the job's review flag in the Database. Subsequently, it recalculates and updates the Provider's overall average rating in the Database, and finally sends a notification via the Notification Service to inform the Provider of the new review.

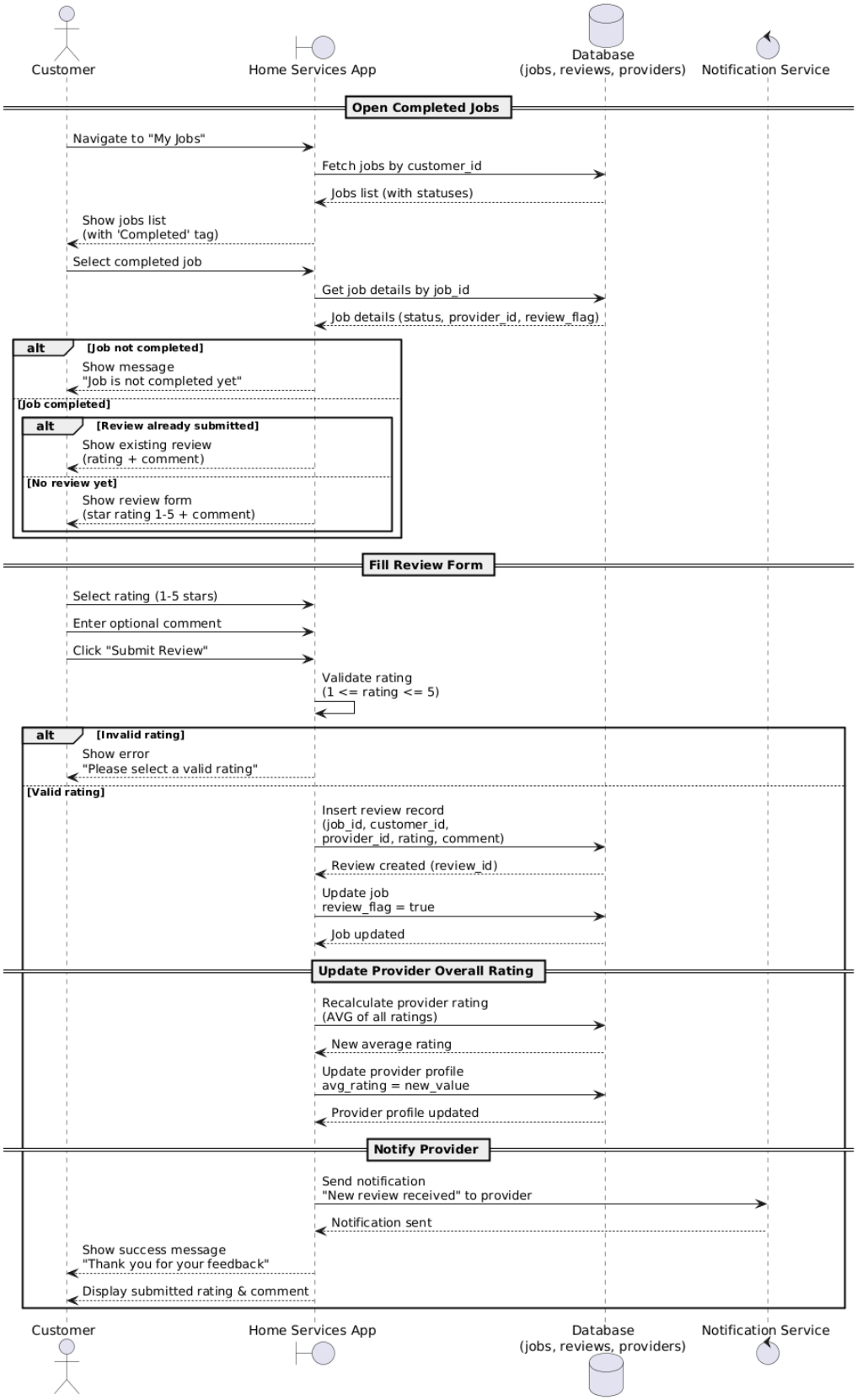

Figure 4.16: Sequence Diagram for UC-015: Withdraw Earnings (Provider)

Sequence Diagram for UC-016: Update Provider Availability

This sequence diagram illustrates the process of an Admin accessing and interacting with the analytics dashboard, which is structured into three main stages: Open Analytics Dashboard, Apply Filters / Drill-down, and Export (Optional). Initially, after logging in, the Home Services App (Admin Panel) requests default Key Performance Indicators (KPIs) from the Analytics Engine, which queries aggregated metrics from the Database to display the initial dashboard. When the Admin applies filters or selects a new view, the App requests filtered metrics from the Analytics Engine. Finally, the Admin can optionally click "Export Report", which prompts the Analytics Engine to generate a report by fetching detailed data from the Database before initiating the file download.

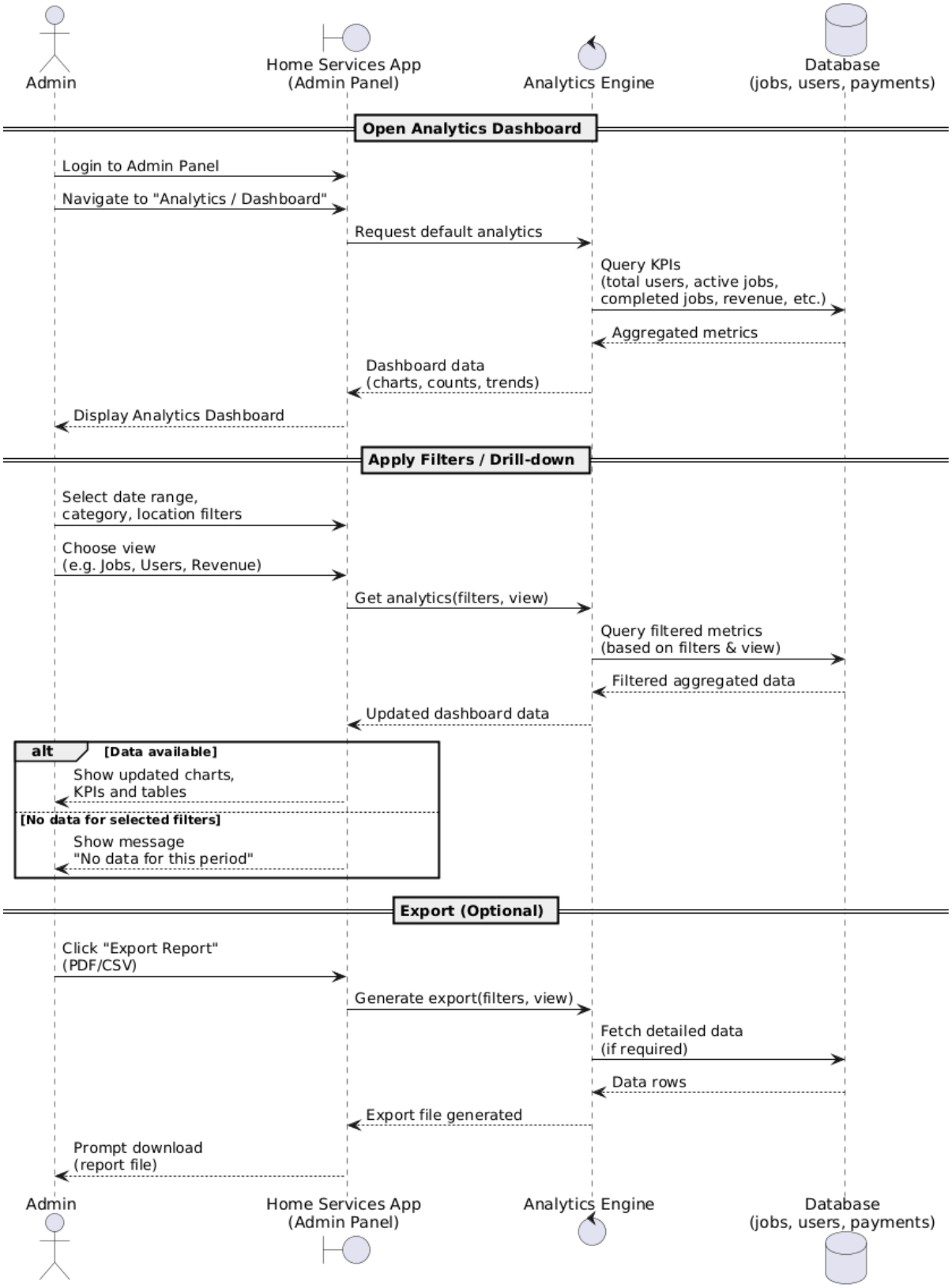

Figure 4.17: Sequence Diagram for UC-016: Update Provider Availability

Sequence Diagram for UC-017: View System Analytics (Admin)

This sequence diagram models the Admin management of user accounts within the Home Services App, which is executed in three main stages: Open User Management, View User Details, and Perform User Action. The Admin first views a list of all users fetched from the Database (users, profiles). Upon selecting a user, the Home Services App (Admin Panel) retrieves detailed profile and metadata. When the Admin selects an action (e.g., Disable, Enable, Verify Provider, Change Role), the App interacts with the Auth Service (Supabase Auth) to update the authentication status and then updates the user's profile status/role in the Database, finally using the Notification Service to inform the user of actions like disabling or verifying their account.

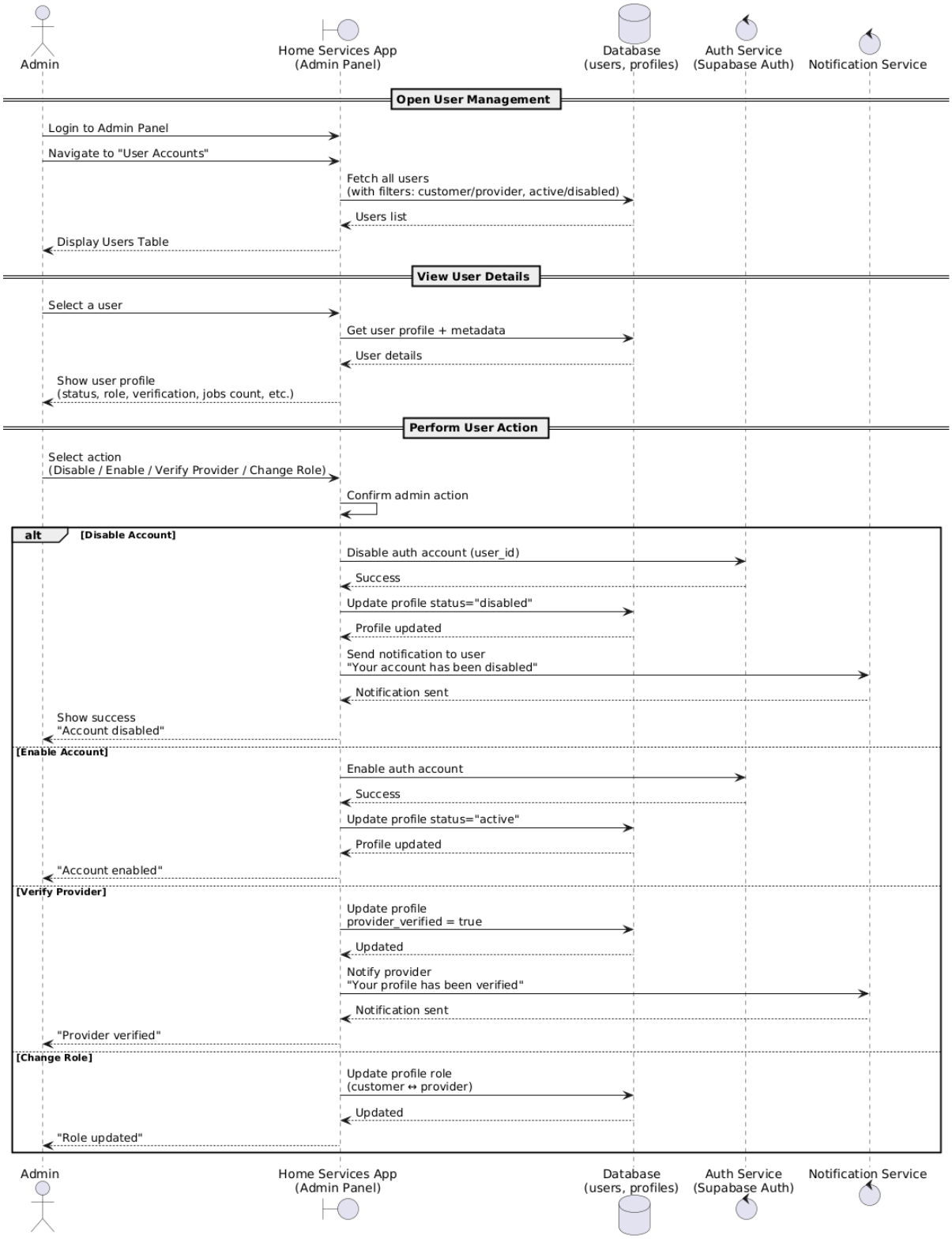

Figure 4.18: Sequence Diagram for UC-017: View System Analytics (Admin)

Sequence Diagram for UC-018: Manage Users (Admin)

This sequence diagram outlines the process for an Admin to access, filter, and export financial reports, which is structured into four main phases: Open Reports Dashboard, Apply Filters, View Detailed Report, and Export Report. Initially, the Home Services App (Admin Panel) requests default financial KPIs from the Reporting Engine, which queries summary metrics from the Database for display. The Admin can then apply filters (e.g., date range, payment method) to query and display filtered aggregated results. Optionally, the Admin can View Detailed Transactions to retrieve the complete dataset. Finally, the Admin can click "Export Report" to prompt the Reporting Engine and Export Service (PDF/CSV) to generate the file and trigger a download of the selected financial data.

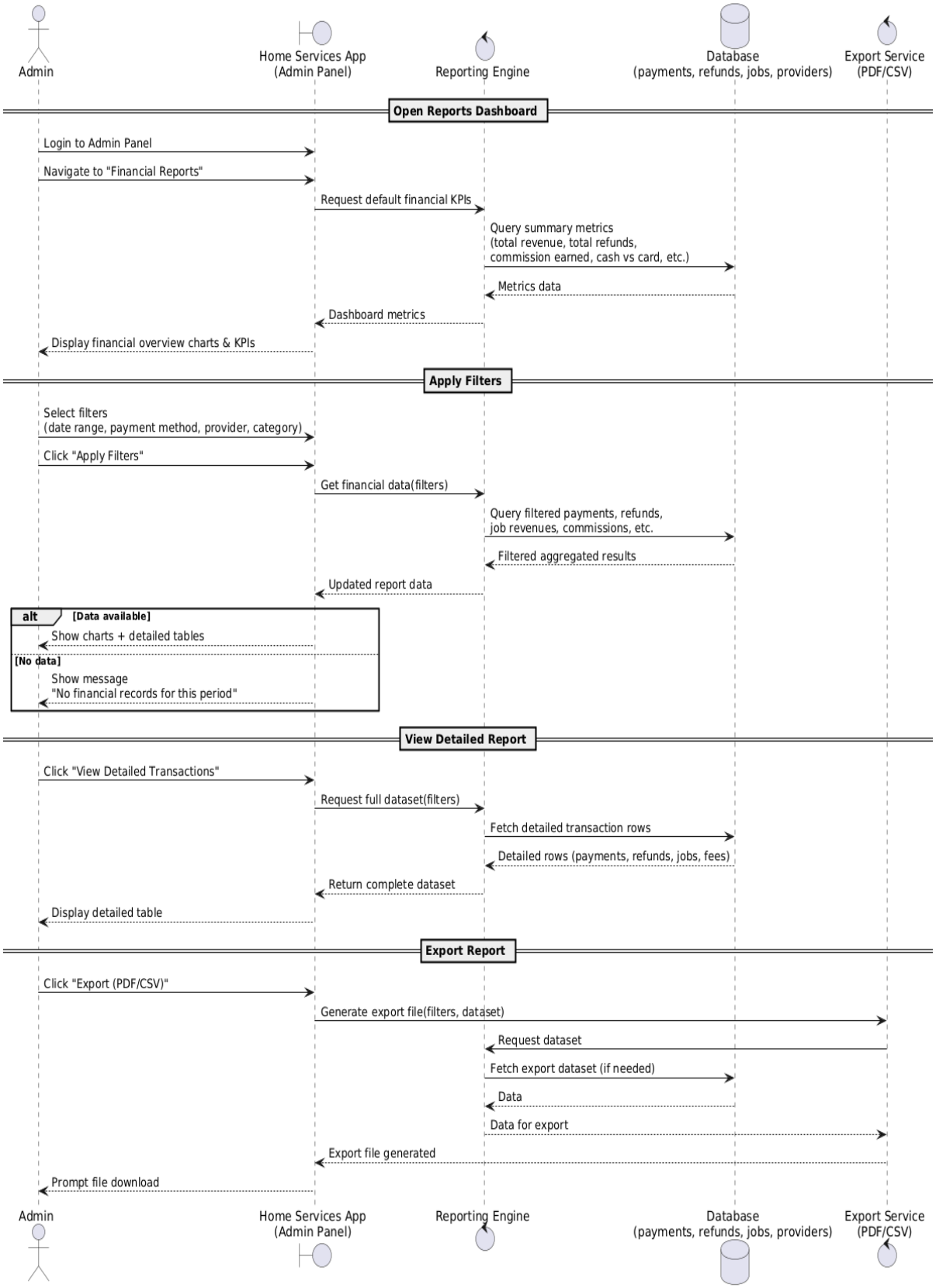

Figure 4.19: Sequence Diagram for UC-018: Manage Users (Admin)

Sequence Diagram for UC-019: Manage Complaints and Reports (Admin)

This sequence diagram illustrates the process of an Admin creating and sending an announcement within the Home Services App, which is executed in four main stages: Open Announcements Module, Create New Announcement, Validate Input, and Save and Deliver Announcement. Initially, the Admin Panel fetches a list of previous announcements from the Database. When the Admin submits a new announcement with a title, message, and selected audience, the App validates the input. Upon successful validation, the App inserts the announcement record into the Database, fetches the target user list based on the audience, and then uses the Notification Service to send a push notification of the announcement to the selected users.

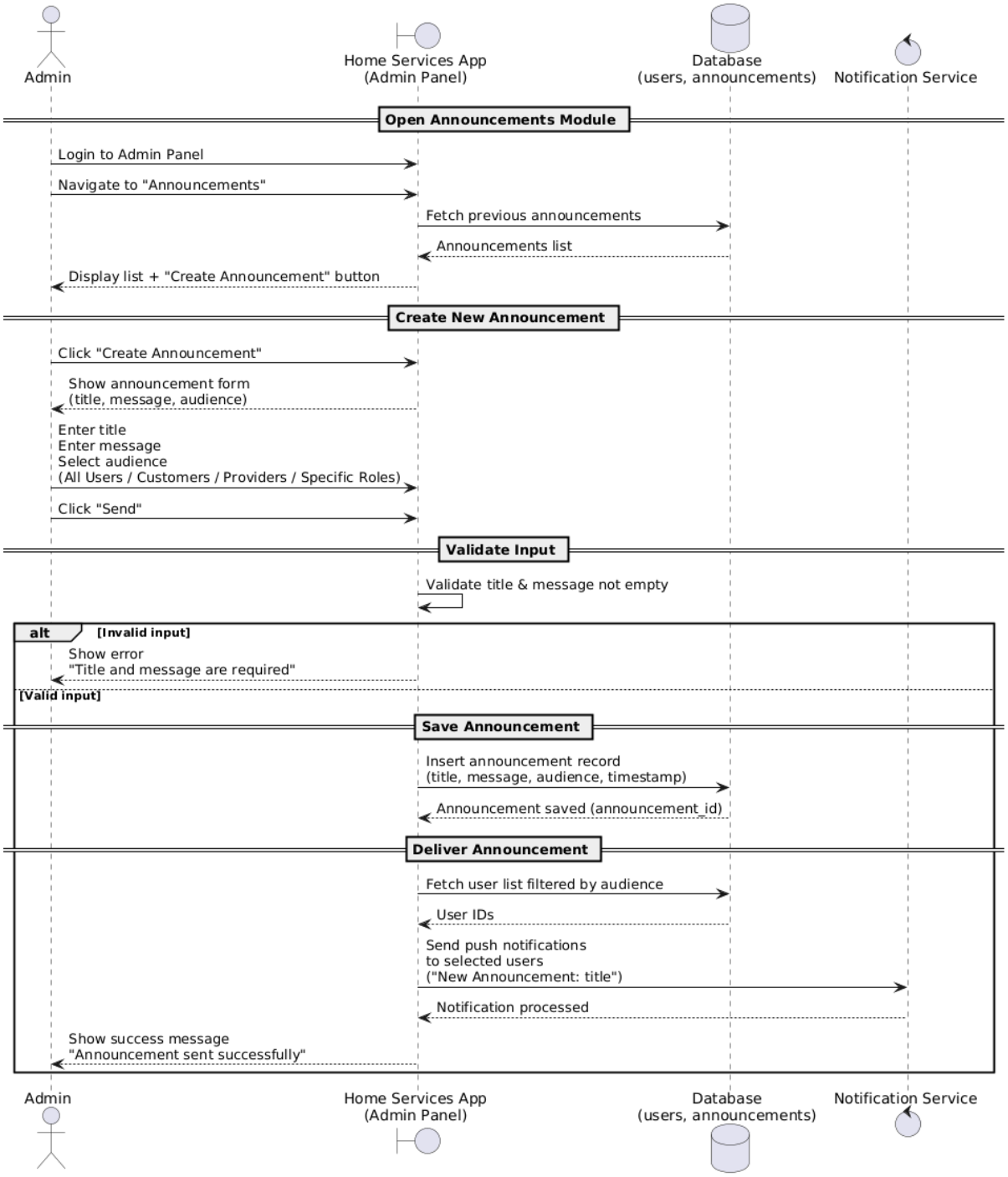

Figure 4.20: Sequence Diagram for UC-019: Manage Complaints and Reports (Admin)

Sequence Diagram for UC-020: Manage Services and Categories (Admin)

This sequence diagram illustrates the complete Admin management of service categories within the Home Services App, detailing four distinct processes: Open Categories Module, Add New Category, Edit Category, and Delete Category. Initially, the Admin Panel fetches all existing categories from the Database. To add a category, the App validates the name and inserts a new record upon success.

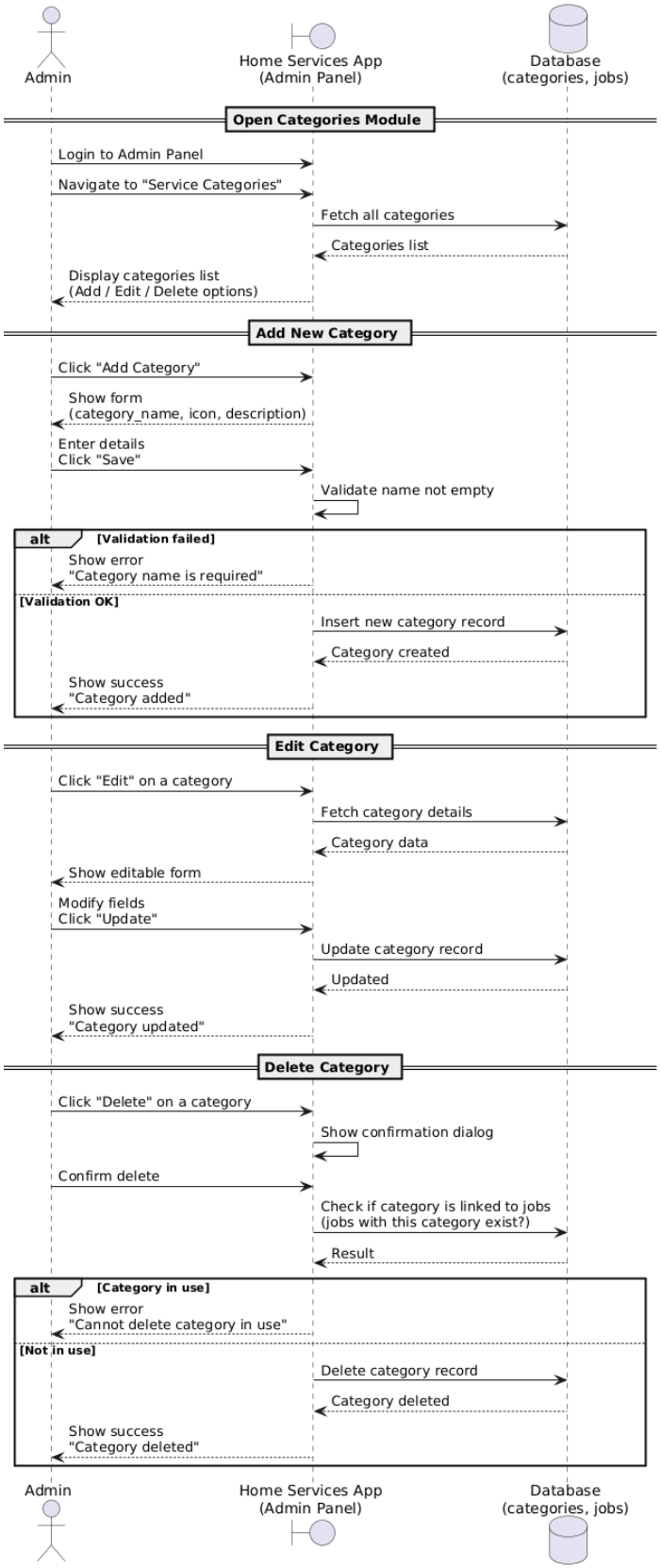

Figure 4.21: Sequence Diagram for UC-020: Manage Services and Categories (Admin)

4.3.2 Activity Diagram

An activity diagram represents the workflow of the system, showing the sequence of activities for specific use cases.

Activity Diagram for Job Request Workflow

When a customer needs a service, they begin by posting a job request on the platform, providing all relevant details such as the type of work, deadline, and any specific requirements. Once the job request is submitted, the system automatically notifies qualified professionals who are registered on the platform, informing them of the new opportunity. Interested professionals then review the job and submit their bids, which typically include their proposed price, timeline, and any additional notes. The customer evaluates the submitted bids and selects the professional they feel is best suited for the task, considering factors such as cost, expertise, and previous ratings. After selection, the professional carries out the work according to the agreed terms, ensuring quality and timely completion. Upon completion, the customer makes the payment through the platform and has the opportunity to provide feedback, which helps maintain accountability, informs future customers, and contributes to the professional's reputation.

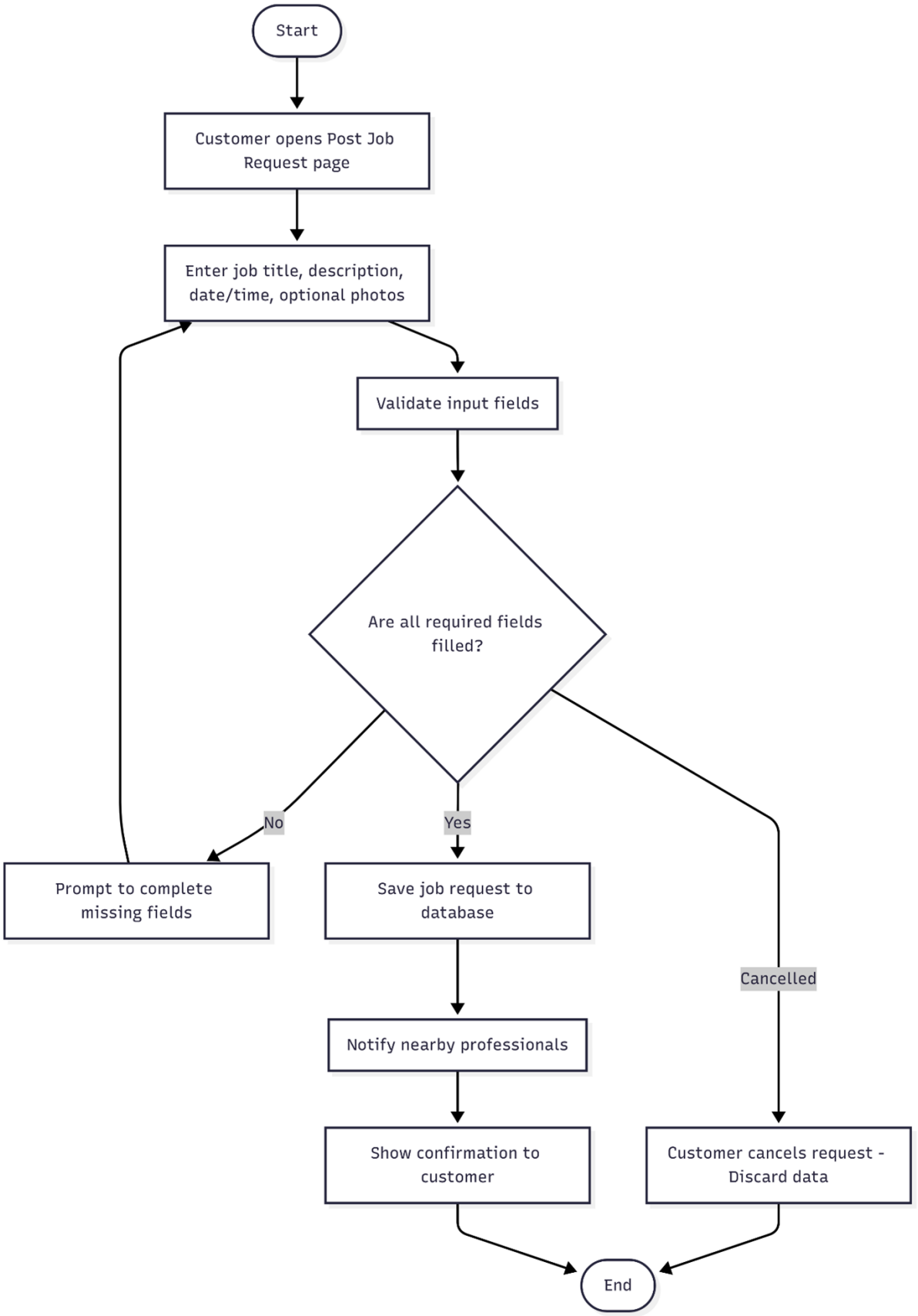

Figure 4.22: Activity Diagram for Job Request Workflow

4.4 GUI Design

The graphical user interface (GUI) design focuses on creating an intuitive and user-friendly experience for customers, professionals, and admins. The design ensures that users can easily navigate the app and access its features.

4.4.1 Login Page and Register Page

The login page allows users to log in using their phone number and one-time password (OTP). The register page allows customers and professionals to create an account by entering personal details, selecting a role, and uploading verification documents.

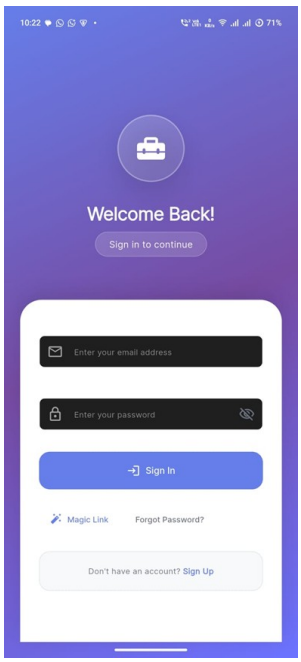

Figure 4.23: Login Page – Enter Phone Number and OTP

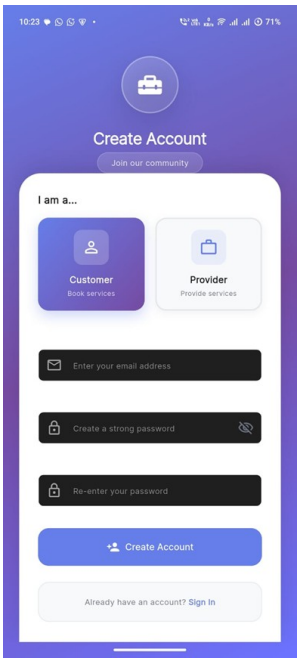

Figure 4.24: Register Page – Enter Personal Details and Upload Documents

4.4.2 Home Page and Job Request Page

The home page provides an overview of app features such as posting job requests, viewing bids, and tracking professionals. The job request page allows customers to post job requests by entering job details, preferred date/time, and uploading photos for clarity.

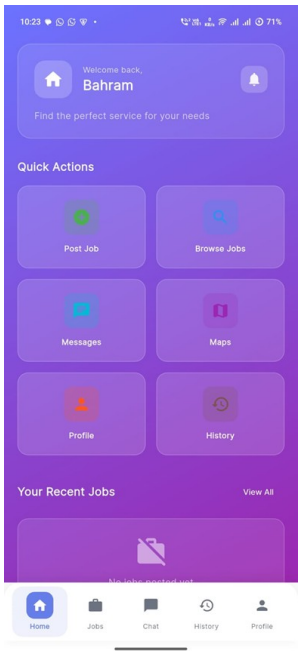

Figure 4.25: Home Page – Dashboard Overview

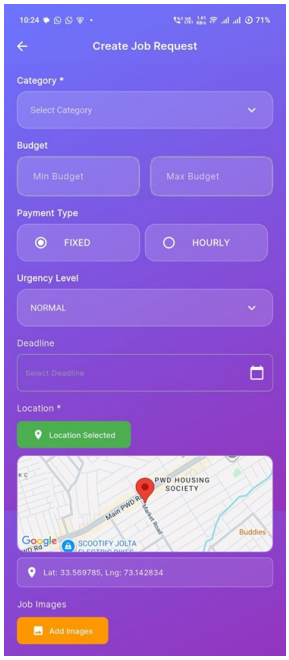

Figure 4.26: Job Request Page – Enter Job Details and Upload Photos

4.4.3 Bidding Page and Tracking Page

The bidding page displays bids from professionals, showing names, bid amounts, and ratings for comparison. The tracking page allows customers to see the real-time location of professionals and the estimated time of arrival (ETA).

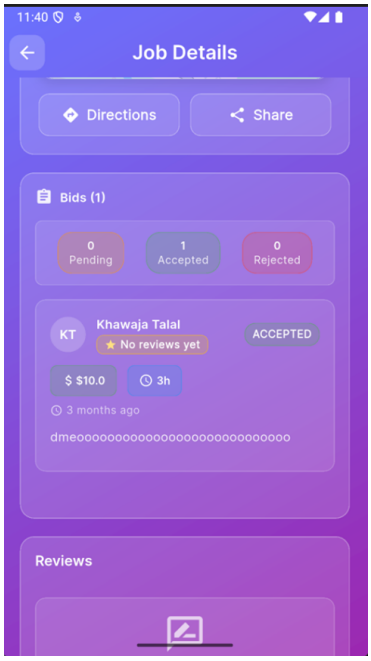

Figure 4.27: Bidding Page – List of Submitted Bids

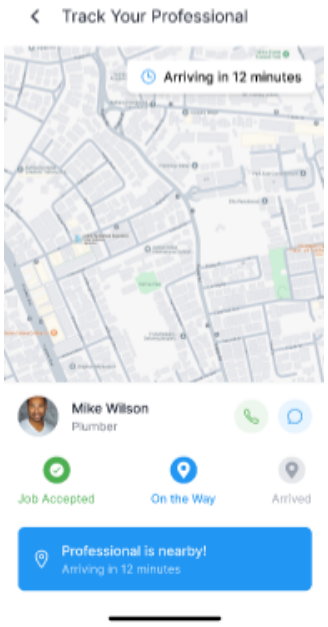

Figure 4.28: Tracking Page – Professional Location and ETA

4.4.4 Admin Dashboard

The admin dashboard provides tools for managing users, monitoring job requests, and accessing analytics. It includes sections for user verification, system monitoring, and sending notifications.

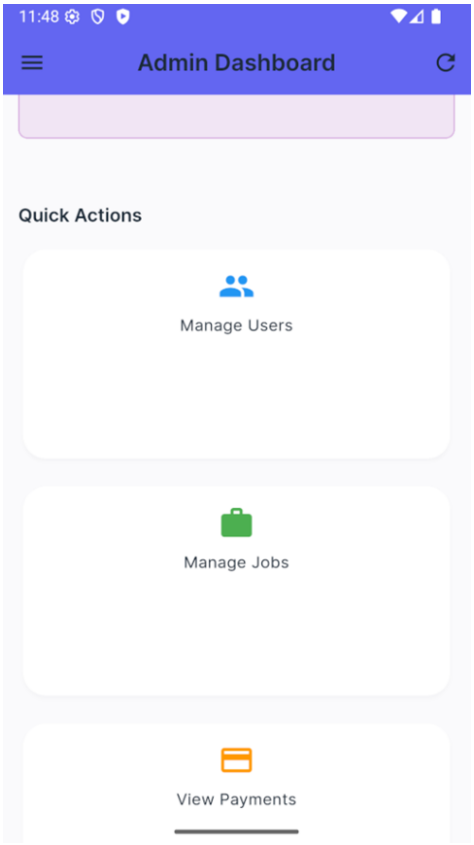

Figure 4.29: Admin Dashboard – Overview, User Management, and Analytics

4.5 System Components

The system is divided into several components, each responsible for specific functionalities:

1. **User Management**
   - Handles user registration, login, and profile management.
   - Ensures secure authentication using OTP-based verification.
2. **Job Management**
   - Allows customers to post job requests and professionals to submit bids.
   - Manages the lifecycle of a job request from posting to completion.
3. **AI-Powered Assistance**
   - Provides chat-based problem diagnosis and price estimation.

- Uses machine learning models to analyze user inputs and suggest solutions.

4. **Real-Time Tracking**
    - Integrates Google Maps API for tracking professionals during on-site visits.
    - Provides real-time updates and notifications to customers.
5. **Stripe Payment Integration**
    - Handles secure online payments using the Stripe payment gateway.
    - Supports multiple payment methods, including credit and debit cards and mobile wallets.
6. **Notification System**
    - Sends real-time notifications for job updates, bid statuses, and appointment reminders.
    - Uses Supabase Cloud Messaging for efficient delivery.

4.6 System Design Principles

The system design follows key principles to ensure reliability, scalability, and user satisfaction:

- **Modularity:** The system is divided into independent modules, making it easier to develop, test, and maintain.
- **Scalability:** The architecture supports future growth, including additional features and a larger user base.
- **Security:** User data and payment information are protected using encryption and secure authentication.
- **Usability:** The app is designed to be intuitive and user-friendly, ensuring a seamless experience for all users.

Chapter 5

System Implementation

This chapter outlines the implementation process of the Home Services App, detailing the tools, technologies, and methodologies used to convert the system design into a functional application. It also describes the internal components, their roles, and how they interact to deliver the app's features.

5.1 Introduction

The implementation phase involves translating the system design into a working application. This includes coding, integrating APIs, setting up the database, and ensuring the app meets the functional and non-functional requirements. The Home Services App is built using the MERN stack (Flutter, Supabase, Google Maps API, and AI/ML tools) to ensure scalability, performance, and cross-platform compatibility.

5.2 System Architecture

The system architecture is based on a three-tier model:

Presentation Layer: The user interface developed using Flutter ensures a consistent experience across Android and iOS platforms.

Business Logic Layer: Supabase Functions and AI/ML models handle the core functionalities, such as bidding, problem diagnosis, and notifications.

Data Layer: Supabase Supabase Database serves as the backend database, managing user data, job requests, bids, and payment transactions.

The architecture ensures modularity, scalability, and efficient communication between components.

5.3 System Internal Components

The Home Services App is divided into several internal components, each responsible for specific functionalities. These components work together to deliver a seamless user experience.

5.3.1 Registration and Supabase Auth Module

Purpose: To manage user registration and secure login.

Implementation:

- Email-based authentication using Supabase Supabase Auth.
- Role-based registration for customers and professionals.
- Validation of professional documents (e.g., CNIC, certifications) during registration.

Outcome: Ensures secure access and verified profiles for professionals.

5.3.2 Job Management Module

Purpose: To handle the lifecycle of job requests, from posting to completion.
Implementation:

- Customers can post job requests with details such as title, description, and photos.
- Professionals can view job requests and submit bids.
- The system notifies customers of new bids and allows them to select a professional.

Outcome: Streamlines the process of connecting customers with professionals.

5.3.3 AI-Powered Assistance Module

Purpose: To provide chat-based problem diagnosis and price estimation.
Implementation:

- AI/ML models analyze user inputs to identify the problem and suggest solutions. [5]
- Price estimation is based on market trends and historical data. [6].
- Natural Language Processing (NLP) enhances the chat system's ability to understand user queries.

Outcome: Improves user experience by offering quick and accurate assistance.

5.3.4 Real-Time Tracking Module

Purpose: To track professionals during on-site visits.
Implementation:

- Google Maps API is integrated to provide real-time location tracking.
- The system calculates the estimated time of arrival (ETA) and updates the customer. [2].
- Geofencing is used to notify customers when the professional is nearby.

Outcome: Enhances transparency and ensures timely service delivery.

5.3.5 Stripe Payment Integration Module

Purpose: To handle secure online payments.
Implementation:

- Integration of a Stripe Payment Gateway to support multiple payment methods (e.g., credit/debit cards, mobile wallets). [4].
- Encryption of payment data to ensure security.
- Automatic generation of invoices and receipts for completed transactions.

Outcome: Provides a seamless and secure payment experience for users.

5.3.6 Notification System

Purpose: To keep users informed about job updates, bid statuses, and reminders.
Implementation:

- Supabase Cloud Messaging (FCM) is used to send real-time notifications.
- Notifications are triggered for events such as new bids, job acceptance, and professional arrival.

Outcome: Keeps users engaged and informed throughout the service process.

5.3.7 Feedback and Review Module

Purpose: To allow customers to rate and review professionals.
Implementation:

- Customers can provide ratings and written feedback after job completion.
- Reviews are displayed on the professional's profile to enhance credibility.

Outcome: Promotes accountability and helps customers make informed decisions.

5.3.8 Admin Panel

Purpose: To manage users, monitor jobs, and ensure smooth platform operation.
Implementation:

- Tools for approving or rejecting professional verification requests.
- System-wide monitoring of job requests, bids, and user activity.
- Analytics dashboard for tracking platform performance.

Outcome: Ensures efficient management and oversight of the platform.

Chapter 6

System Testing and Evaluation

This chapter outlines the testing process for the Home Services ...e app performs reliably and delivers a seamless user experience.

6.1 System Testing

System testing is conducted to verify that the Home Services App...the entire system, including all modules and their interactions.

Objectives of System Testing

- Ensure the app meets functional requirements, such as job posting, bidding, and real-time tracking.
- Validate non-functional requirements, including performance, usability, and security.
- Identify and resolve any bugs or inconsistencies in the system.

Types of System Testing Conducted

- Functional Testing: Verifies that each feature works as expected.
- Integration Testing: Ensures seamless communication betwe...ifferent modules (e.g., job management and payment integration).
- Performance Testing: Evaluates the app's performance under various conditions, such as high user loads.
- Security Testing: Ensures user data and payment information are protected.
- Usability Testing: Assesses the app's user interface and overall user experience.

6.2 Testing Strategies

The following testing strategies were employed to ensure comprehensive coverage of the system:

1. Unit Testing

Focuses on testing individual components or modules, such as the registration module or bidding system. Ensures that each module functions correctly in isolation.

2. Integration Testing

Verifies that different modules work together seamlessly. Example: Ensuring the job management module communicates effectively with the notification system.

3. User Acceptance Testing (UAT)

Involves real users testing the app to ensure it meets their expectations. Feedback from UAT is used to make improvements before deployment.

4. Automated Testing

Tools like Selenium and JMeter were used to automate repetitive test cases, such as login and job posting workflows.

5. Manual Testing

Conducted for scenarios requiring human judgment, such as usability testing and visual design validation.

6.3 Test Cases

The following test cases were designed to validate the app's functionalities:

6.3.1 Registration and Login

Table 6.1: Test Cases for Registration and Login

Test Case ID	Description	Steps	Expected Result	Result
TC01	User Registration	1. Open the registration page.	User account is created successfully.	Pass
		2. Enter valid details and submit.		
TC02	Login with OTP	1. Open the login page.	User logs in successfully.	Pass
		2. Enter phone number and OTP.		

6.3.2 Job Posting

Table 6.2: Test Cases for Job Posting

Test Case ID	Description	Steps	Expected Result	Result
TC03	Post a Job Request	1. Navigate to the job posting page.	Job request is posted successfully.	Pass
		2. Enter job details and submit.		
TC04	Missing Job Details	1. Navigate to the job posting page.	System prompts user to fill required fields.	Pass
		2. Submit without entering details.		

6.3.3 Bidding System

Table 6.3: Test Cases for Bidding System

Test Case ID	Description	Steps	Expected Result	Result
TC05	View Bids	1. Navigate to the job request page.	Bids are displayed successfully.	Pass
		2. Select a job request with bids.		
TC06	Select a Bid	1. View bids for a job request.	Selected professional is notified.	Pass
		2. Choose a bid and confirm.		

6.3.4 Real-Time Tracking

Table 6.4: Test Cases for Real-Time Tracking

Test Case ID	Description	Steps	Expected Result	Result
TC07	Track Professional	1. Navigate to the tracking page.	Professional's location is displayed.	Pass
		2. View real-time location on the map.		

6.3.5 Stripe Payment Integration

Table 6.5: Test Cases for Stripe Payment Integration

Test Case ID	Description	Steps	Expected Result	Result
TC08	Make a Payment	1. Navigate to the payment page.	Payment is processed successfully.	Pass
		2. Enter payment details and confirm.		
TC09	Payment Failure	1. Navigate to the payment page.	System displays error message.	Pass
		2. Enter invalid payment details.		

6.4 Testing Results

The testing process yielded the following results:

- Functional Testing: All core functionalities, such as registration, job posting, and bidding, passed successfully.
- Integration Testing: Modules communicated seamlessly, with no data inconsistencies.
- Performance Testing: The app handled up to 1,000 concurrent users without significant performance degradation.
- Security Testing: User data and payment information were encrypted and protected against unauthorized access.
- Usability Testing: Users found the app intuitive and easy to navigate, with minor improvements suggested for the UI.

6.5 Exception Testing

Exception testing was conducted to ensure the app handles unexpected scenarios gracefully.

Examples of Exception Testing

Invalid Login Credentials:
Test: Enter an unregistered phone number during login.
Result: System displays an error message prompting the user to register.

Network Failure:
Test: Attempt to track a professional with no internet connection.
Result: System displays a message indicating network issues.

Payment Gateway Timeout:
Test: Simulate a timeout during payment processing.
Result: System notifies the user and allows them to retry.

6.6 Performance Testing

Performance testing was conducted to evaluate the app's responsiveness and stability under various conditions.

Key Metrics

- Response Time: Average response time for job posting and bidding was under 2 seconds.
- Concurrent Users: The app supported up to 1,000 concurrent users without crashes.
- Load Testing: The system maintained stability under high traffic conditions.

6.7 Usability Testing

Usability principles follow standard UX engineering guidelines [12].

Feedback from Users

Positive: Users appreciated the intuitive design and seamless navigation.
Suggestions: Minor improvements were suggested for the bidding page layout and notification settings.

6.8 Summary of Testing

The testing phase ensured that the Home Services App meet...nfidence in its reliability, performance, and user satisfaction.

Chapter 7

Conclusion

This chapter summarizes the outcomes of the Home Services App project, highlights its key achievements, and discusses potential future enhancements. The project successfully addresses the challenges in the home services industry by providing a reliable, efficient, and transparent platform for customers and professionals.

7.1 Summary of the Project

The Home Services App was developed to bridge the gap between customers and home service professionals by leveraging modern technologies such as AI, real-time bidding, and Google Maps integration. The app provides a seamless experience for users, ensuring transparency, reliability, and convenience.

Key Features Delivered

- **AI-Powered Assistance:** Chat-based problem diagnosis and price estimation.
- **Real-Time Bidding:** Competitive pricing through a transparent bidding system.
- **Professional Verification:** Verified badges to ensure trust and reliability.
- **Real-Time Tracking:** Google Maps integration for tracking professionals during on-site visits.
- **Secure Payments:** Integration of a Stripe Payment Gateway for hassle-free transactions.
- **Feedback System:** Ratings and reviews to promote accountability and quality service.

Project Outcomes

- The app successfully meets the functional and non-functional requirements outlined in the initial proposal.
- Comprehensive testing ensured the app's reliability, performance, and usability.
- The app is ready for deployment on Android and iOS platforms, providing a scalable solution for the home services industry.

7.2 Challenges Faced

The development process encountered several challenges, which were addressed through careful planning and problem-solving:

1. Data Security

Challenge: Ensuring the security of user data and payment information.
Solution: Implemented encryption and secure authentication mechanisms.

2. AI Model Accuracy

Challenge: Developing accurate AI models for problem diagnosis and price estimation.
Solution: Trained models using diverse datasets and conducted extensive testing.

3. Real-Time Tracking

Challenge: Ensuring reliable location tracking in areas with poor network connectivity.
Solution: Optimized the Google Maps API integration for better performance.

4. User Experience

Challenge: Designing an intuitive and user-friendly interface for diverse user groups.
Solution: Conducted usability testing and incorporated user feedback to improve the UI.

7.3 Key Learnings

The project provided valuable insights and learning opportunities for the team:

1. Technical Expertise

- Gained hands-on experience in using Flutter for cross-platform development.
- Enhanced knowledge of Supabase Database for real-time database management.
- Learned to integrate AI/ML models for chat-based assistance and price estimation.

2. Problem-Solving Skills

- Developed strategies to address challenges such as data security, AI accuracy, and real-time tracking.
- Improved the ability to troubleshoot and resolve technical issues during development.

3. Team Collaboration

- Strengthened teamwork and communication skills through effective collaboration and task distribution.
- Leveraged version control tools like GitHub to manage code and ensure smooth collaboration.

7.4 Future Enhancements

While the Home Services App meets its current objectives, there is always room for improvement and expansion. The following enhancements are proposed for future development:

7.4.1 Mobile Application Expansion

Objective: Develop a dedicated mobile application for tablets and other devices.
Benefit: Broaden the app's accessibility and usability across different platforms.

7.4.2 Live Chat Support

Objective: Introduce a live chat feature for real-time communication with customer support.
Benefit: Enhance user experience by providing instant assistance for queries and issues.

7.4.3 Advanced AI Features

Objective: Improve the AI-powered chat system with advanced Natural Language Processing (NLP) capabilities.
Benefit: Provide more accurate problem diagnosis and personalized recommendations.

7.4.4 Background Checks for Professionals

Objective: Implement a more rigorous verification process, including background checks for professionals.
Benefit: Increase trust and reliability for customers.

7.4.5 Expand Service Categories

Objective: Add more service categories, such as landscaping, pest control, and appliance repair.
Benefit: Cater to a wider audience and increase the app's market reach.

7.4.6 Multi-Language Support

Objective: Introduce support for multiple languages to cater to diverse user groups.
Benefit: Improve accessibility for non-English-speaking users.

7.4.7 Subscription Plans

Objective: Offer subscription plans for professionals to access premium features, such as priority bidding and enhanced profile visibility.
Benefit: Generate additional revenue and provide value-added services for professionals.

7.5 Conclusion

The Home Services App successfully addresses the challenges in the home services industry by providing a transparent, efficient, and user-friendly platform. By leveraging modern technologies such as AI, real-time bidding, and Google Maps integration, the app ensures a seamless experience for both customers and professionals.

The project has been a valuable learning experience for the team, enhancing their technical skills, problem-solving abilities, and teamwork. With the proposed future enhancements, the app has the potential to become a leading solution in the home services market, benefiting a wide range of users.

The successful completion of this project demonstrates the team's ability to deliver innovative and impactful solutions, setting the stage for future endeavors in the field of software development.

References

[1] Google, "Flutter Documentation," 2024. [Online]. Available: https://docs.flutter.dev/

[2] Google Developers, "Google Maps Platform Documentation," 2024. [Online]. Available: https://developers.google.com/maps

[3] Supabase, "Supabase Developer Documentation," 2024. [Online]. Available: https://supabase.com/docs

[4] Stripe, "Stripe API Reference," 2024. [Online]. Available: https://stripe.com/docs/api

[5] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed., Pearson, 2023.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[7] Urban Company, "About Urban Company," 2024. [Online]. Available: https://www.urbancompany.com/

[8] TaskRabbit, "How TaskRabbit Works," 2024. [Online]. Available: https://www.taskrabbit.com/

[9] Thumbtack, "Thumbtack Services Overview," 2024. [Online]. Available: https://www.thumbtack.com/

[10] Handy, "About Handy Home Services," 2024. [Online]. Available: https://www.handy.com/

[11] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 9th ed., McGraw-Hill, 2020.

[12] J. Nielsen, *Usability Engineering*. Morgan Kaufmann, 1994.