TROY UNIVERSITY

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION AND COMMUNICATIONS TECHNOLOGY

FINAL PROJECT

COURSE: SOFTWARE ENGINEERING I

Taskera Home Service Booking Platform

Group: **[GROUP]**

Supervisor: **Assoc. Prof. Dr. Cao Tuan Dung**

List of students:

No.	Full name	Student ID
1	[STUDENT_1]	[ID_1]
2	[STUDENT_2]	[ID_2]
3	[STUDENT_3]	[ID_3]
4	[STUDENT_4]	[ID_4]
5	[STUDENT_5]	[ID_5]

HANOI, June 2026

Troy University Resources

Contents

Preface **8**

Group Member Task Assignment **9**

1 Problem Investigation **10**

1.1 Problem Requirements Description 10
1.2 Business Requirements Elicitation 10
1.3 Identification of Basic Business Information 11
1.4 Business Process Diagram and Functional Decomposition Diagram 12
1.5 Simple Project Plan 16

2 PROBLEM REQUIREMENTS SPECIFICATION **19**

2.1 General Introduction 19
2.2 Use Case Diagram 21
2.2.1 General Use Case Diagram 21
2.2.2 Level-2 Decomposed Use Case Diagram 21
2.3 Use Case Specification 25
2.4 Non-functional Requirements 47
2.4.1 Functionality 47
2.4.2 Usability 48
2.4.3 Reliability 48
2.4.4 Performance 48
2.4.5 Security 48
2.4.6 Maintainability & Portability 49
2.4.7 Design Constraints 49

3 REQUIREMENTS ANALYSIS **50**

3.1 Analysis Class Identification 50
3.2 Sequence Diagrams 59
3.3 Entity-Relationship Diagram (ERD) 65
3.3.1 Main data objects 66
3.3.2 ERD diagrams 66
3.4 Description of relationships between data objects 66
3.4.1 Account and Authentication Module 66
3.4.2 Service Request Module 66
3.4.3 Provider, Dispatch, and Job Module 69
3.4.4 Communication, Feedback, and Administration Module 69

4 SYSTEM DESIGN 70
4.1 Architectural Design . . . 70
4.1.1 Selection of Software Architecture . . . 70
4.1.2 Application of the Architecture to Taskera . . . 70
4.1.3 UML Package Diagram and Package Dependencies . . . 71
4.2 Database Design . . . 74
4.2.1 Detailed Specifications of Selected Database Tables . . . 78
4.3 Detailed Package Design . . . 88
4.4 Detailed Class Design . . . 92
4.5 Detailed Class Diagram . . . 99

5 IMPLEMENTATION OF THE DEMONSTRATION PROGRAM 101
5.1 Libraries and Tools Used . . . 101
5.2 Demonstration Program Results . . . 101
5.2.1 System Packaging structure . . . 102
5.2.2 Packaged Product 1: Backend (Server) . . . 102
5.2.3 Packaged Product 2: Frontend Admin . . . 102
5.2.4 Packaged Product 3: Mobile Client (Flutter App) . . . 103
5.2.5 Packaged Product 4: Database & Services . . . 103
5.3 User Interfaces Illustrating Program Functions . . . 104

6 PROGRAM TESTING 121
6.1 Testing approach . . . 121
6.2 Testing environment . . . 121
6.3 Functional testing . . . 121
6.3.1 Test cases for customer features . . . 121
6.3.2 Test cases for provider features . . . 122
6.3.3 Test cases for chat and admin features . . . 123
6.4 Non-functional testing . . . 124
6.5 Testing conclusion . . . 124

7 INSTALLATION AND USER GUIDE 125
7.1 Installation Guide . . . 125
7.2 Users and Scope Of Use . . . 125
7.3 Installation Requirements . . . 125
7.4 Detailed Installation Steps . . . 125
7.5 Software User Guide . . . 125

Conclusion and Future Development 126

References 127

Troy University Resources

List of Figures

1-1 Activity Diagram of Taskera Service Booking and Fulfillment Process 13
1-2 Functional Decomposition Diagram of Taskera System 15
2-1 General Use Case Diagram of Taskera System 22
2-2 Admin Module Use Case Diagram 23
2-3 Customer Module Use Case Diagram 24
2-4 Provider Module Use Case Diagram 25
3-1 Sequence Diagram - UC-01 Register Customer Account 60
3-2 Sequence Diagram - UC-02 Log in to the System 60
3-3 Sequence Diagram - UC-03 Recover Forgotten Password 61
3-4 Sequence Diagram - UC-04 Book a Home Service 61
3-5 Sequence Diagram - UC-05 Manage Service Request History 62
3-6 Sequence Diagram - UC-06 Register as a Service Provider 62
3-7 Sequence Diagram - UC-07 Accept or Decline a Dispatched Job 63
3-8 Sequence Diagram - UC-08 Track and Update Job Progress 63
3-9 Sequence Diagram - UC-09 Review a Completed Service 64
3-10 Sequence Diagram - UC-10 Approve or Reject Provider Application 64
3-11 Sequence Diagram - UC-11 Send and Receive Chat Messages 65
3-12 Sequence Diagram - UC-12 Manage Notifications and Preferences 65
3-13 ERD - Account and Authentication Module 66
3-14 Service Request Management Entity Relationship Diagram 67
3-15 ERD - Service Request Module 67
3-16 ERD - Provider Dispatch and Job Module 68
3-17 ERD - Communication Feedback and Administration Module 68
4-1 Taskera Three-Tier System Architecture Diagram 71
4-2 UML Package Diagram - Three-Tier Overview 72
4-3 UML Package Diagram - Presentation Tier 73
4-4 UML Package Diagram - Application and Data Tier 74
4-5 Relational Database Schema - Account, Profile and Authentication Module 79
4-6 Relational Database Schema - Service Request and Dispatch Module 80
4-7 Relational Database Schema - Job Management and Provider Operations Module 80
4-8 Relational Database Schema - Communication, Notification and Feedback Module . . . 81
4-9 Relationship Diagram between Tables in Taskera Database 82
4-10 Detailed Package Design - Authentication Package 89
4-11 Detailed Package Design - Service Request and Dispatch Packages 90
4-12 Detailed Package Design - Job Management Package 91
4-13 Detailed Package Design - Communication and Notification Packages 92
4-14 Admin Management Package 93
5-1 Login Page Interface 104
5-2 Register Page Interface for Customer 105

5-3 Register Page Interface for Provider 105
5-4 Provider Home Page Interface 106
5-5 Completed Jobs and Rating 107
5-6 Provider's Profile Interface 107
5-7 Provider's Update Profile Interface 108
5-8 Provider's Manage Skills Interface 109
5-9 Provider's Area Settings Interface 109
5-10 Provider's Notification Settings Interface 110
5-11 New Job Popup Interface 110
5-12 Job Progress Page Interface 111
5-13 Real-time Chat Page Interface 112
5-14 Customer's Main Page Interface 113
5-15 Customer's Notification Interface 113
5-16 Customer's View Profile Interface 114
5-17 Customer's Profile Interface 114
5-18 Customer's Update Information Interface 115
5-19 Customer's Manage Address Interface 115
5-20 Customer's Booking History Interface 116
5-21 Customer's Notification Settings Interface 116
5-22 Customer's Job Finding Interface 117
5-23 Customer's Service Request Form Interface 118
5-24 Customer’s Cleaning Service Detail Description Interface 118
5-25 Customer’s Cleaning Service Schedule Selection Interface 119
5-26 Customer’s Cleaning Service Confirmation Interface 120

Troy University Resources

List of Tables

1-1 Identification of Basic Business Information 11
1-3 Functional Feasibility Assessment of Taskera System 13
1-5 Simple Project Plan 16
1-7 Project Risk Management Plan 17
2-1 System Actors and Descriptions 19
2-3 Overview of Core Use Cases 19
2-5 Use Case Specification - UC-01 Register Customer Account 25
2-8 Use Case Specification - UC-02 Log in to the System 27
2-11 Use Case Specification - UC-03 Recover Forgotten Password 27
2-14 Use Case Specification - UC-04 Book a Home Service 28
2-16 Use Case Specification - UC-05 Manage Service Request History 29
2-18 Use Case Specification - UC-06 Register as a Service Provider 29
2-21 Use Case Specification - UC-07 Accept or Decline a Dispatched Job 31
2-24 Use Case Specification - UC-08 Track and Update Job Progress 32
2-27 Use Case Specification - UC-09 Review a Completed Service 33
2-30 Use Case Specification - UC-10 Approve or Reject Provider Application 34
2-33 Use Case Specification - UC-11 Send and Receive Chat Messages 35
2-36 Use Case Specification - UC-12 Manage Notifications and Preferences 35
2-38 Use Case Specification - UC-13 Manage Customer Profile and Addresses 36
2-41 Use Case Specification - UC-14 Manage Provider Profile and Availability 37
2-44 Use Case Specification - UC-15 Browse Available Service Requests 38
2-47 Use Case Specification - UC-16 View Job History 39
2-50 Use Case Specification - UC-17 Manage Provider Posts 40
2-53 Use Case Specification - UC-18 View Provider Public Profile 40
2-56 Use Case Specification - UC-19 Manage Service Categories 42
2-59 Use Case Specification - UC-20 Admin Manage Users 42
2-62 Use Case Specification - UC-21 Admin Manage Jobs and Requests 44
2-65 Use Case Specification - UC-22 Switch Provider for a Job 45
2-68 Use Case Specification - UC-23 Upload Images 46
2-71 Use Case Specification - UC-24 Register FCM Device Token 47
4-2 Specification of Table “users” 78
4-4 Specification of Table “cust_profiles” 83
4-6 Specification of Table “customer_addresses” 83
4-8 Specification of Table “provider_profiles” 84
4-10 Specification of Table “service_categories” 84
4-12 Specification of Table “service_requests” 85
4-14 Specification of Table “dispatch_logs” 85
4-16 Specification of Table “dispatch_candidates” 86
4-18 Specification of Table “jobs” 86

4-20 Specification of Table “job_assignments” 87
4-22 Specification of Table “messages” 87
4-24 Specification of Table “notifications” 87
4-26 Specification of Table “reviews” 88
5-1 Libraries and Tools Used 101

Troy University Resources

Preface

Taskera Home Service Booking Platform was chosen because the demand for home services, such as cleaning, electrical repair, plumbing repair, and maintenance, has become increasingly common in daily life. However, many customers still find these services through informal or fragmented channels, which can make it difficult to check worker availability, compare service quality, and build trust. At the same time, independent service providers may also face difficulties in finding stable customers and managing their work schedules effectively.

For these reasons, our group decided to develop Taskera as a digital platform that connects customers with suitable service providers in a more organized and convenient way. The system supports the service-booking process by allowing customers to create requests, helping providers receive and manage jobs, and enabling both sides to track service progress more clearly. By applying software engineering knowledge to a real-life problem, this project gives our group an opportunity to practice requirement analysis, system design, implementation, and testing while building a useful solution for the home service market.

During the development process, our group first analyzed the current problems in traditional home-service booking. Then, we identified the main users of the system, defined functional and non-functional requirements, built use case models, designed the database and system architecture, and implemented a demonstration version of the application. Through each stage, the project was gradually improved to better support the main workflow of booking services, dispatching providers, tracking jobs, communicating between users, and reviewing completed services.

Group Member Task Assignment

Full name	Student ID	Summary of completed work	Evaluation
[STUDENT_1]	[ID_1]	Group leader. Controlled the progress of team members. Managed project documents and coordinated task assignment. Reviewed source code and merged pull requests. Backend: completed remaining backend functions and connected the database. DevOps: prepared deployment, Docker configuration. Fixed program bugs. Prepared report sections: database design, program implementation, testing, and user guide.	10 / 10
[STUDENT_2]	[ID_2]	Frontend developer. Designed reusable UI components for the mobile application. Frontend: implemented the customer flow, including home page, service selection, service request creation, and job tracking. Improved UI/UX and fixed frontend bugs.	9.5 / 10
[STUDENT_3]	[ID_3]	Backend developer. Developed complex logic and real-time features. Frontend: implemented the real-time chat screen using WebSocket. Built the provider flow, including receiving requests, accepting or declining jobs, and updating job status. Supported bug fixing and code review.	9.5 / 10
[STUDENT_4]	[ID_4]	Frontend developer and Tester. Frontend: implemented authentication functions, including login, registration, forgot password, and OTP verification. Built the notification module and integrated Firebase Push Notification. Performed functional testing. Detected, reported, and fixed program bugs.	9 / 10
[STUDENT_5]	[ID_5]	Frontend developer and Business Analyst. Frontend: implemented review functions, review list display, profile screen, and settings screen. Managed project documents. Prepared report sections: user guide, README, system diagrams, and project documentation. Supported testing and bug fixing.	9 / 10

Troy University Resources

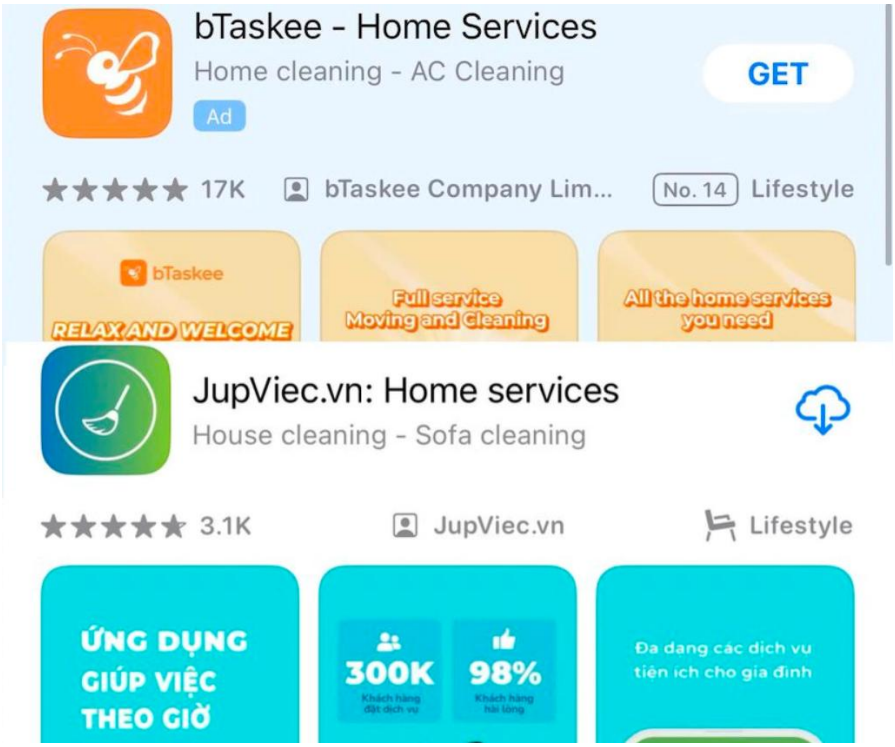
CHAPTER 1. Problem Investigation

1.1. Problem Requirements Description

In recent years, the demand for finding workers who provide home services, such as electrical and plumbing repair, cleaning, and equipment maintenance, has increased significantly. However, the process of connecting customers with workers often faces many difficulties, including verifying information, negotiating prices, and tracking work progress. Based on this situation, the Taskera system is proposed as a centralized platform that helps automate the booking process and improve the efficiency of service management.

1.2. Business Requirements Elicitation

An initial survey of the current market landscape indicates a substantial and growing demand for domestic maintenance and cleaning services. However, the existing mechanisms for connecting customers with service providers contains significant limitations. Empirical observations reveal that customers frequently encounter difficulties in verifying the professional competence, reliability, and background of independent workers. Simultaneously, service providers often lack access to a stable and centralized channel to reach potential clients, leading to inconsistent employment opportunities and inefficient time management. Based on the identification of these critical points, the project group established the core business requirements for the Taskera platform. The system is designed to resolve these connectivity issues by acting as a reliable intermediary that standardizes the service request and fulfillment lifecycle. By defining clear operational workflows, the project seeks to optimize the scheduling and coordination of services, ensuring that both consumers and providers can interact within a transparent, structured, and beneficial digital environment.





1.3. Identification of Basic Business Information

Table 1-1: Identification of Basic Business Information

Input	Process	Output
Customer account information, including full name, phone number, email address, password, and contact information	Register and manage customer accounts	A verified customer account stored in the databaseCustomer profile information available for booking and service tracking
Customer profile and address information, including default address, detailed location, phone number, and saved service addresses	Manage customer profile and addresses	Updated customer profileList of saved customer addresses used for future service requests
Provider registration information, including personal information, contact information, service area, years of experience, and service categories	Register and manage provider profiles	Provider profile created in the systemProvider information available for admin verification and job assignment
Provider verification information, including identity documents, certificates, professional experience, and service capacity	Review and approve provider applications	Provider application status updated as approved or rejectedVerified provider list available for dispatching service requests
Service request information, including selected service type, service description, attached images, address, location, preferred time, and customer notes	Create and manage service booking requests	A new service request recordInitial booking status generated and stored in the system
Service category information, booking schema, estimated service requirements, and customer-selected options	Process service booking details	Structured booking dataService request details prepared for provider matching and job coordination
Booking request information, customer location, service category, schedule, provider availability, and provider online/offline status	Dispatch suitable providers for a service request	List of suitable provider candidatesJob offer notifications sent to providersProvider assignment result
Provider response information, including acceptance, rejection, response note, and response time	Process provider acceptance or rejection	Accepted provider assigned to the jobRejected or expired offers recordedRequest redispatched if no provider accepts

Troy University Resources

Input	Process	Output
Job status information, including accepted, on the way, in progress, completed, cancelled, or provider-switching status	Track and update job progress	Updated job status recordReal-time progress information displayed to the customer and provider
Communication information, including sender, receiver, message content, image attachment, timestamp, and related job ID	Manage in-app chat between customer and provider	Chat history stored for each jobMessages displayed in the conversation screen
Notification information, including user device token, notification type, job status update, and recipient information	Send real-time system notifications through FCM	Notification records stored in the systemCustomers and providers receive updates about booking, dispatching, and job progress
Payment information, including job ID, service price, final price, payment method, payment status, and payment time	Manage service payment after job completion	Payment record createdPayment status updated for the completed job
Review information, including rating score, customer comment, attached images, provider ID, and related job ID	Manage customer reviews and provider ratings	Customer review stored in the systemProvider rating and service quality information updated
Administrative information, including user accounts, provider applications, service categories, job records, reports, and disputes	Manage system administration activities	Updated user, provider, service category, and job recordsAdministrative decisions recorded in the system
System operation data, including number of bookings, completed jobs, cancelled jobs, provider performance, customer reviews, and payment records	Generate operational statistics and management reports	Summary statistics for system monitoringData foundation for service improvement and future business decisions

1.4. Business Process Diagram and Functional Decomposition Diagram

The workflow begins in the Customer lane, where the customer browses and selects a home service, enters the required booking details, and submits the service request. The process then moves to the System lane, where the system validates the submitted information. If the request data is invalid, the system returns a validation error and the customer must correct the booking details before resubmitting the request. If the request is valid, the system creates a new service request, matches suitable candidate providers, and dispatches a job offer notification to an available provider.

After receiving the job offer, the workflow continues in the Provider lane, where the provider reviews the request details and decides whether to accept or decline the job. If the provider declines the offer, the system updates the offer status, evaluates the next available candidate, and continues the dispatching process. If the provider accepts the job, the provider starts traveling to the customer's location, updates the job status and progress during service execution, and marks the job as completed when the work is finished.

Finally, the process returns to the Customer and System lanes. The system requests customer confirmation, while the customer tracks job progress, confirms service completion, makes payment, and reviews the completed service. The diagram clearly shows the sequence of actions, decision points, and interactions among the Customer, System, and Provider, providing a comprehensive view of the end-to-end business process from service booking to job fulfillment and post-service evaluation.

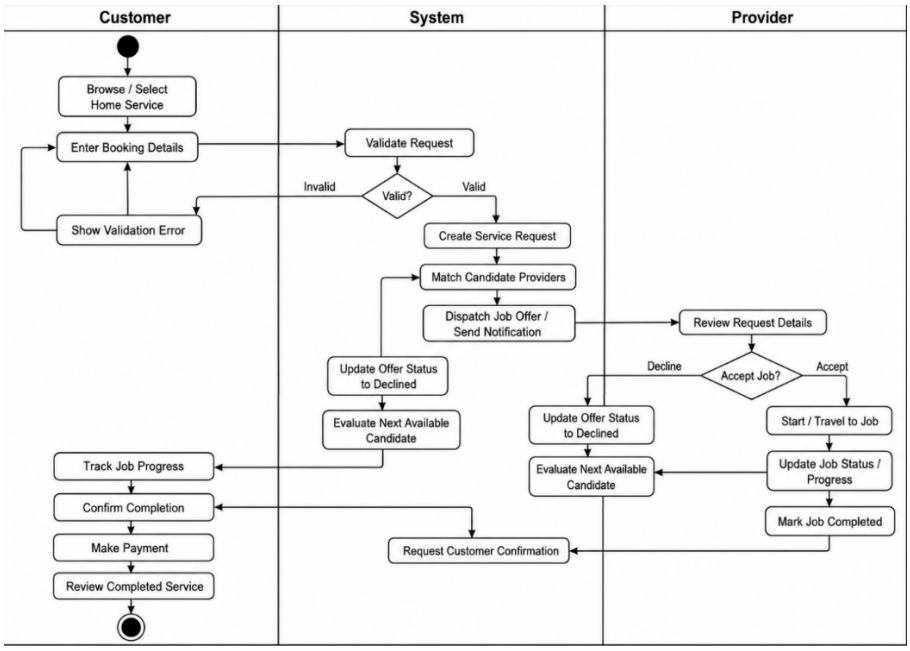


Figure 1-1: Activity Diagram of Taskera Service Booking and Fulfillment Process

Table 1-3: Functional Feasibility Assessment of Taskera System

Function name	Description	Feasibility assessment (People, Time, Technology, Environment)
Authentication	Handle account registration, login, password recovery, token refresh, and logout for customers, providers, and administrators.	People: Team has experience with frontend and backend authentication flows. Time: Medium, because OTP, password reset, and token handling require careful implementation. Technology: Supported by NestJS, JWT, validation DTOs, and Flutter screens. Environment: Suitable for mobile and web-based service platforms.
Customer Service Booking	Allow customers to browse home services, enter booking details, upload related images, and submit service requests.	People: Requires coordination between UI, backend APIs, and database design. Time: Medium–high due to dynamic booking forms and validation rules. Technology: Supported by Flutter, NestJS REST APIs, Prisma, and file upload services. Environment: Core function of an on-demand home service platform.
Service Request History	Allow customers to view previous and current service requests, inspect request details, and cancel requests when allowed.	People: Team can divide frontend list/detail screens and backend query logic. Time: Medium. Technology: Supported by REST APIs, relational database queries, and request status management. Environment: Necessary for tracking customer activity in Taskera.
Job Tracking and Fulfillment	Allow customers and providers to track job progress from acceptance to completion, including status updates and timeline information.	People: Requires understanding of business workflow and job status transitions. Time: Medium–high because multiple actors interact in the same process. Technology: Supported by backend job APIs, status logs, notifications, and Flutter tracking screens. Environment: Essential for real-time service coordination.

Troy University Resources

Function name	Description	Feasibility assessment (People, Time, Technology, Environment)
Review Management	Allow customers to review completed services and allow the system to store ratings and feedback for providers.	People: Easy to implement once job completion logic is available. Time: Low–medium. Technology: Supported by CRUD APIs, database relations, and simple mobile UI forms. Environment: Common and suitable for service marketplace applications.
Provider Profile Management	Allow providers to manage profile information, certificates, service areas, skills, availability, and working status.	People: Requires both domain modeling and frontend form handling. Time: Medium–high due to multiple provider-related data fields. Technology: Supported by Prisma models, NestJS services, validation DTOs, and Flutter provider pages. Environment: Important for maintaining provider reliability and service quality.
Job Offer Management	Allow providers to review dispatched job offers and accept or decline them based on availability and job details.	People: Requires clear understanding of dispatching workflow. Time: Medium. Technology: Supported by dispatch APIs, provider offer screens, and notification logic. Environment: Fits Taskera's provider-matching and job assignment process.
Job Status Updating	Allow providers to update job status, progress, location, and completion state during service execution.	People: Requires careful handling of state transitions and access control. Time: Medium. Technology: Supported by job status APIs, backend validation, and mobile job tracking screens. Environment: Necessary for controlling the service lifecycle.
Communication	Allow customers and providers to send and receive chat messages related to a job or service request.	People: Requires frontend chat UI and backend messaging logic. Time: Medium–high if real-time communication is included. Technology: Supported by Socket.IO, REST message history APIs, and mobile chat screens. Environment: Useful for clarifying service details during booking and fulfillment.
Notification Management	Manage push notifications and notification preferences for users, including job updates, chat alerts, and system messages.	People: Requires understanding of mobile notification flow. Time: Medium. Technology: Supported by Firebase Cloud Messaging, device tokens, notification APIs, and preferences.

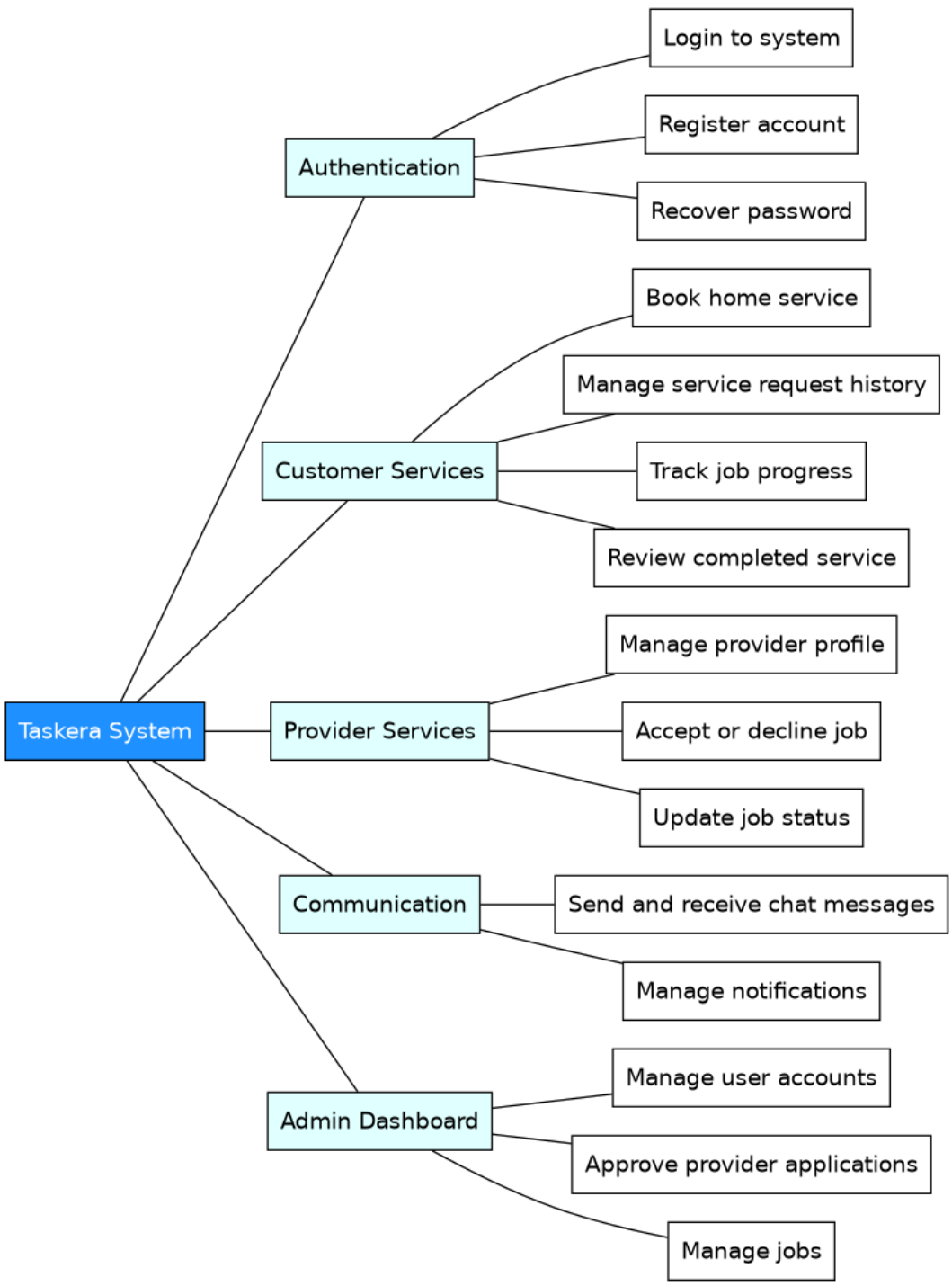


Figure 1-2: Functional Decomposition Diagram of Taskera System

Troy University Resources

1.5. Simple Project Plan

Table 1-5: Simple Project Plan

Task	Time (hours)	Number of people
1. Analysis and Overall Design	35	5
1.1 Survey home-service booking platforms and user needs	8	3
1.2 Analyze Taskera business process and main user roles	7	5
1.3 Identify functional and non-functional requirements	8	4
1.4 Design overall system architecture	8	3
1.5 Create detailed task allocation plan	4	5
2. Requirements Modeling and Use Case Design	40	4
2.1 Identify system actors and major use cases	8	4
2.2 Build overview use case diagram	6	2
2.3 Build decomposed use case diagrams	8	2
2.4 Write detailed use case specifications	12	4
2.5 Review requirements with business workflow	6	4
3. Data Modeling and Database Design	45	3
3.1 Design entities, enums, and relationships	12	2
3.2 Design Prisma schema and database tables	12	2
3.3 Build ERD and analyze data dependencies	8	2
3.4 Design DTOs and validation rules	8	2
3.5 Review data model with use cases	5	3
4. Backend Core Development	75	3
4.1 Build NestJS module structure	8	2
4.2 Implement authentication and user management APIs	12	2
4.3 Implement customer service request APIs	15	2
4.4 Implement provider profile, skills, and availability APIs	12	2
4.5 Implement job dispatching and job status management	16	3
4.6 Backend testing and bug fixing	12	3
5. Mobile Frontend Development	70	3
5.1 Design main Flutter screen structure and navigation	10	2
5.2 Build authentication and profile screens	12	2
5.3 Build customer booking and request history screens	16	2
5.4 Build provider dashboard and job offer screens	14	2
5.5 Connect frontend with backend APIs	10	3
5.6 UI testing and user flow validation	8	3
6. Communication and Notification Features	45	3
6.1 Implement real-time chat between customer and provider	14	2
6.2 Implement message history and unread message status	8	2
6.3 Implement Firebase Cloud Messaging device token registration	8	2
6.4 Implement job and chat notifications	10	3
6.5 Test notification and chat behavior	5	3
7. Admin Dashboard and Provider Verification	40	2
7.1 Build admin provider approval interface	10	1
7.2 Implement provider application approval/rejection APIs	10	1
7.3 Build admin user and job management functions	12	2
7.4 Test admin permissions and workflow	8	2
8. System Integration and Testing	35	5
8.1 Integrate mobile app, backend, database, and notification services	10	5
8.2 Perform end-to-end testing for main workflows	10	5
8.3 Test error handling and alternative flows	8	4
8.4 Fix bugs and improve system stability	7	5

Task	Time (hours)	Number of people
9. Report Writing and Defense Preparation	40	5
9.1 Write problem investigation and requirements sections	8	3
9.2 Write analysis, design, and database sections	10	3
9.3 Write implementation, testing, and installation guide	10	4
9.4 Prepare diagrams, screenshots, and appendix	6	3
9.5 Prepare presentation slides and practice Q&A	6	5

Table 1-7: Project Risk Management Plan

Task / Activity	Hazard	Risk	Level	Strategy	Mitigation
Requirement analysis	Requirements are unclear or change during development	Some functions may not match real user needs	High	Risk mitigation	Hold team discussions, define clear use cases, and confirm requirements with the lecturer
Business process design	Booking and job fulfillment flow is incomplete	The system workflow may become inconsistent or hard to follow	High	Risk mitigation	Draw activity diagrams, review normal and alternative flows, and update the workflow before coding
Use case specification	Use cases are too detailed or overlapping	The report becomes confusing and difficult to maintain	Medium	Risk mitigation	Group related use cases, keep only important use cases as full specifications, and review actor responsibilities
Database design	Incorrect entity relationships	Data inconsistency between users, providers, jobs, requests, and reviews	High	Risk mitigation	Design ERD carefully, review Prisma schema, and test relationships with sample data
Backend development	Logic errors in service requests, dispatches, or job status modules	Jobs may be assigned incorrectly, or status transitions may be invalid	High	Risk mitigation	Test each API, review business rules, and validate job status transitions carefully
Frontend development	Poor user interface or confusing navigation	Customers and providers may find the app difficult to use	Medium	Risk mitigation	Build simple screens, test main user flows early, and improve based on feedback
Authentication and authorization	Incorrect role checking	Customers, providers, or admins may access unauthorized functions	High	Risk elimination	Apply role-based access control, protect APIs, and test permissions for each role
Provider verification	Incomplete checking of provider documents	Unverified or unsuitable providers may be approved	High	Risk mitigation	Require identity documents and certificates, provide admin approval screens, and store provider status clearly
Job dispatching	No suitable provider is found or providers decline requests	Customer requests may remain unresolved	High	Risk mitigation	Implement candidate matching, retry dispatching, and allow the system to evaluate the next available provider
Chat and communication	Real-time messages are delayed or not delivered	Customers and providers may fail to coordinate service details	Medium	Risk mitigation	Store message history, use socket connection carefully, and test chat behavior in different screens

Troy University Resources

Task / Activity	Hazard	Risk	Level	Strategy	Mitigation
Notification management	Push notifications are not received	Users may miss job offers, job updates, or chat messages	Medium	Risk mitigation	Register FCM device tokens, handle notification preferences, and test notifications on real devices
File and image upload	Uploaded failed images or invalid files are accepted	Booking evidence or provider documents may be missing	Medium	Risk mitigation	Validate file type and size, handle upload errors, and display clear feedback to users
FE–BE integration	API contract mismatch between Flutter and NestJS	The application may fail during real user flows	High	Risk mitigation	Define API endpoints clearly, use consistent DTOs, and perform integration testing regularly
Testing	Not enough time for complete testing	Many bugs may remain in the final demo	Medium	Risk mitigation	Reserve dedicated testing time, prioritize core workflows, and prepare test cases from use cases
Time management	Development takes longer than expected	The team may not finish all planned features before the deadline	High	Risk mitigation	Create a project plan, track progress weekly, and prioritize core functions first
Human resources	Team members are busy or unavailable	Some modules may be delayed	Medium	Risk transfer	Assign backup members for important tasks and distribute work among all 5 members
Report writing	Report is written too late	The final report may lack detail, diagrams, or consistency	Medium	Risk mitigation	Write the report gradually, update diagrams during development
Demo preparation	Demo environment is unstable	The system may fail during presentation	High	Risk mitigation	Prepare test accounts, sample data, backup screenshots, and a stable local/cloud demo environment

CHAPTER 2. PROBLEM REQUIREMENTS SPECIFICATION

2.1. General Introduction

The Taskera system involves three human actors: Customer, Provider, and Admin. In addition, the system interacts with supporting external services such as OTP/Email Service, File Storage, WebSocket Gateway, Matching/Dispatch Service, and FCM Service. Based on the collected business requirements, the team identified a total of 24 core Use Cases, covering the entire lifecycle of a service request. These Use Cases serve as the foundation for the next phases of architecture design and source code implementation.

Table 2-1: System Actors and Descriptions

No.	Actor name	Actor description
1	Customer	End user who uses the Taskera application to search for and book home services.
2	Provider	Service provider who registers skills, accepts jobs, and completes service tasks.
3	Admin	System administrator who manages users, providers, categories, and resolves disputes.
4	System	Backend server that processes requests, manages business logic, and sends notifications.

Table 2-3: Overview of Core Use Cases

No.	Use case ID	Use case name	Use case description	Interacting actor	Complexity
1	UC-01	Register customer account	Allows the Customer to register and verify customer account. APIs: POST /auth/register, POST /auth/verify-otp, POST /auth/resend-otp.	Customer	Medium
2	UC-02	Log in to the system	Allows the Customer to login to the system. APIs: POST /auth/login, POST /auth/refresh, POST /auth/logout.	Customer, Provider, Admin	Medium
3	UC-03	Recover forgotten password	Allows the Customer to recover forgotten password. APIs: POST /auth/forgot-password, POST /auth/reset-password, GET /auth/reset-password-page.	Customer, Provider	Medium
4	UC-04	Book a home service	Allows the Customer to book a home service. APIs: GET /service-categories/:id, PATCH /service-categories/:id, GET /service-categories/:id/booking-schema (+3 more).	Customer	High
5	UC-05	Manage service request history	Allows the Customer to manage service request history. APIs: GET /requests/my, GET /requests/:id, PATCH /requests/:id/cancel.	Customer	Medium
6	UC-06	Register as a service provider	Allows the Provider to register as a service provider. APIs: POST /auth/register, GET /providers/me, PATCH /providers/me (+2 more).	Provider	High
7	UC-07	Accept or decline a dispatched job	Allows the Provider to accept or decline a dispatched job. APIs: POST /dispatch/:requestId/run, GET /dispatch/:requestId/candidates, GET /dispatch/my-offers (+2 more).	Provider	High

Troy University Resources

No.	Use case ID	Use case name	Use case description	Interacting actor	Complexity
8	UC-08	Track and update job progress	Allows the Provider to track and update job progress. APIs: GET /jobs/:id, PATCH /jobs/:id/status, PATCH /jobs/:id/progress (+3 more).	Provider, Customer	High
9	UC-09	Review a completed service	Allows the Customer to review a completed service. APIs: POST /jobs/:jobId/reviews, GET /jobs/:jobId/reviews, GET /providers/:providerProfileId/reviews (+1 more).	Customer	Medium
10	UC-10	Approve or reject provider application	Allows the Admin to approve or reject provider application. APIs: GET /admin/providers, GET /admin/providers/pending, PATCH /admin/providers/:id/approve (+3 more).	Admin	High
11	UC-11	Send and receive chat messages	Allows the Customer to send and receive chat messages. APIs: GET /chat/conversations, GET /chat/unread, GET /jobs/:jobId/messages (+3 more).	Customer, Provider	High
12	UC-12	Manage notifications and preferences	Allows the Customer to manage notifications and preferences. APIs: POST /fcm/device-token, DELETE /fcm/device-token, GET /notifications/me (+5 more).	Customer, Provider	High
13	UC-13	Manage customer profile and addresses	Allows the Customer to manage customer profile and addresses. APIs: GET /customers/me, PATCH /customers/me, GET /customers/addresses (+5 more).	Customer	High
14	UC-14	Manage provider profile and availability	Allows the Provider to manage provider profile and availability. APIs: GET /providers/me, PATCH /providers/me, GET /providers/dashboard-stats (+7 more).	Provider	High
15	UC-15	Browse available service requests (job board)	Allows the Provider to browse available service requests (job board). APIs: GET /requests/available, GET /requests/:id.	Provider	Medium
16	UC-16	View job history	Allows the Provider to view job history. APIs: GET /jobs/my, GET /jobs/my-customer.	Provider, Customer	Medium
17	UC-17	Manage provider posts (portfolio)	Allows the Provider to manage provider posts (portfolio). APIs: POST /providers/posts, GET /providers/posts/me, PATCH /providers/posts/:id (+3 more).	Provider	High
18	UC-18	View provider public profile	Allows the Customer to view provider public profile. APIs: GET /providers/top, GET /providers/:id, GET /providers/:id/stats.	Customer	Medium
19	UC-19	Manage service categories	Allows the Admin to manage service categories. APIs: GET /service-categories/:id, PATCH /service-categories/:id.	Admin	Medium

No.	Use case ID	Use case name	Use case description	Interacting actor	Complexity
20	UC-20	Admin manage users	Allows the Admin to admin manage users. APIs: GET /admin/users, GET /admin/users/:id, PATCH /admin/users/:id/suspend (+4 more).	Admin	High
21	UC-21	Admin manage jobs and requests	Allows the Admin to admin manage jobs and requests. APIs: GET /admin/requests, POST /admin/requests/:id/cancel, GET /admin/jobs (+4 more).	Admin	High
22	UC-22	Switch provider for a job	Allows the Customer to switch provider for a job. APIs: POST /dispatch/:requestId/retry, POST /dispatch/:requestId/manual-assign, POST /jobs/:id/switch-provider.	Customer	Medium
23	UC-23	Upload images	Allows the Customer to upload images. APIs: POST /upload/single, POST /upload/multiple, GET /upload/signed-url.	Customer, Provider	Medium
24	UC-24	Register FCM device token	Allows the System to register fcm device token. APIs: POST /fcm/device-token, DELETE /fcm/device-token.	System	Low

2.2. Use Case Diagram

2.2.1. General Use Case Diagram

This diagram identifies the primary human actors, including Guest, Customer, Provider Applicant, Provider, and Admin, together with supporting system services such as OTP/Email Service, File Storage, WebSocket Gateway, Matching/Dispatch Service, and FCM Service. These actors represent the different roles involved in the service booking platform, from account registration and authentication to service request management, provider operation, communication, notification, and administrative control.
Moreover, guests can register a customer account or apply to become a service provider, while registered users can log in, recover forgotten passwords, and manage notifications. Customers mainly interact with booking-related functions, including booking a home service, tracking job progress, reviewing completed services, and managing service request history. Providers are responsible for managing their profiles, accepting or declining dispatched jobs, and updating job status during service execution. Admins manage platform operations such as provider approval, job and request management, and provider reassignment. External system services support key technical operations, including OTP verification, file uploading, real-time messaging, provider matching, dispatching, and push notifications. Therefore, the diagram gives a high-level view of how Taskera organizes its main actors and core system functions.
Before logging in, a guest can register a new account, log in, request a password reset when the password is forgotten, and reset the password after receiving reset instructions by email. When the guest logs in successfully, the system creates a menu containing the functions corresponding to the user group to which that user belongs. When the guest requests a password reset, the system generates a token and sends password reset instructions by email. After logging in, users can view and update their personal information and change their password.

2.2.2. Level-2 Decomposed Use Case Diagram

The Admin Module Use Case Diagram presents the main administrative functions of the Taskera system. In this diagram, the Admin actor is responsible for managing platform operations, including managing jobs and service requests, reassigning providers when necessary, and approving or rejecting provider

Troy University Resources

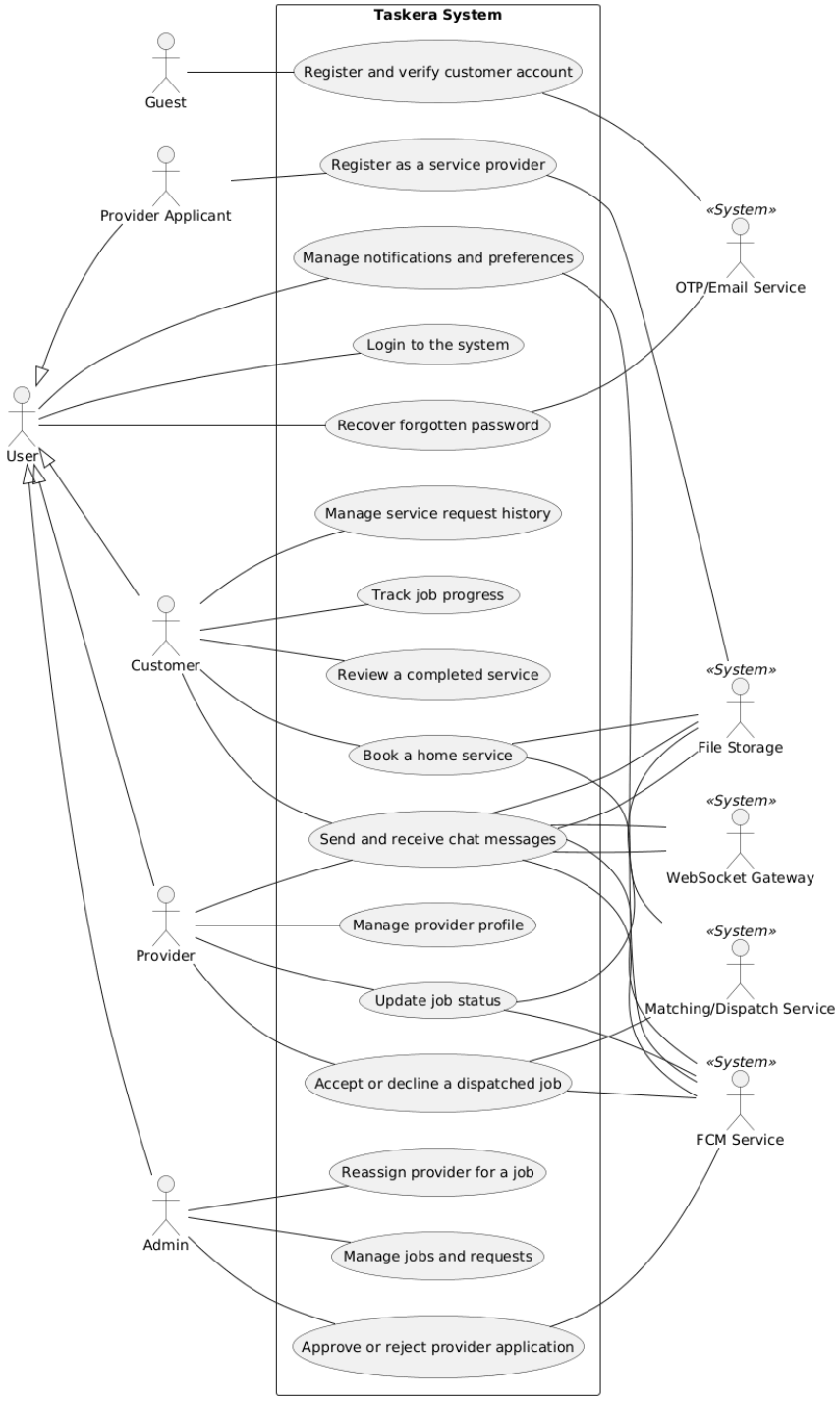


Figure 2-1: General Use Case Diagram of Taskera System

applications. These functions allow the administrator to supervise the service fulfillment process and maintain the quality of providers participating in the platform.

The diagram also shows that Admin is a specialized type of User. Therefore, the Admin can access common user functions such as logging in to the system, recovering a forgotten password, and managing notification preferences. In addition, the module interacts with supporting system services, including the FCM Service for notification delivery and the OTP/Email Service for password recovery. Overall, this diagram clarifies the administrative scope of Taskera and shows how admin operations are connected with user authentication and system notification services.

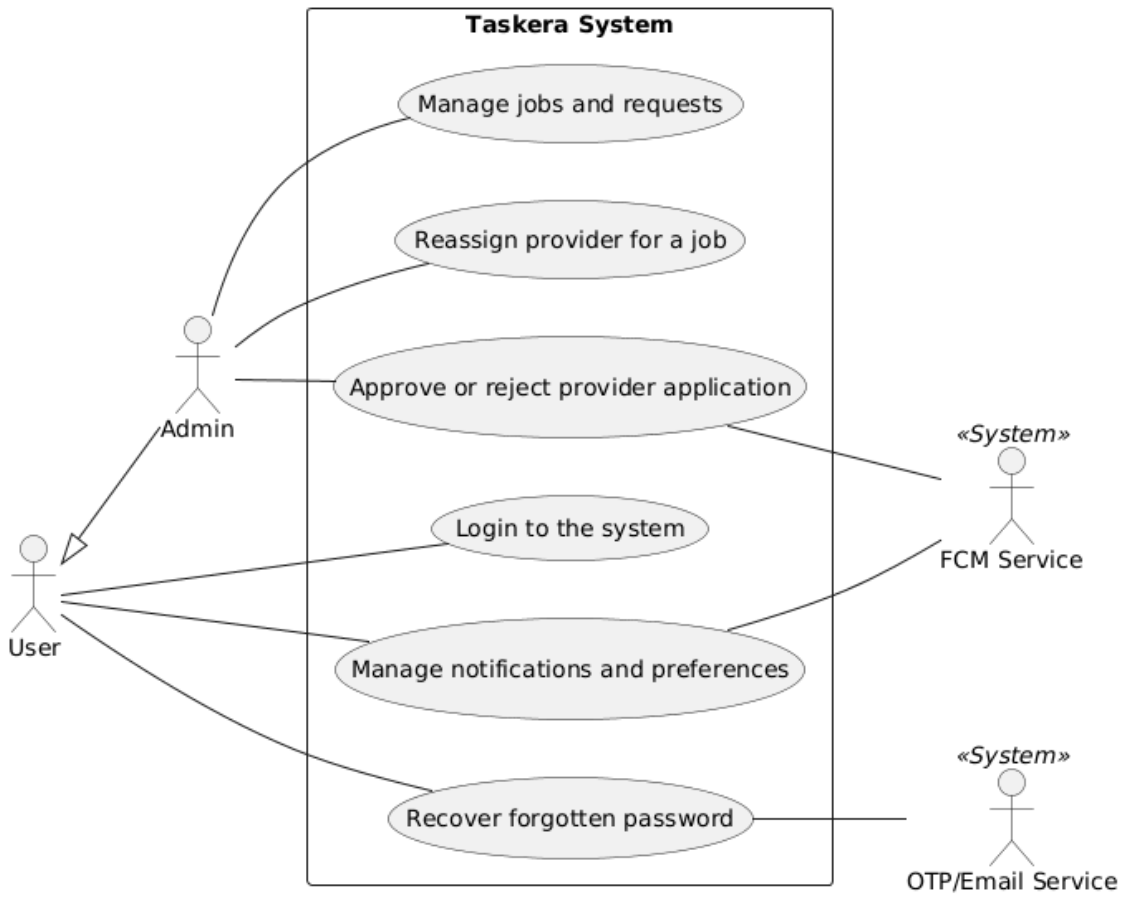


Figure 2-2: Admin Module Use Case Diagram

The Customer Module Use Case Diagram describes the main functions available to customers in the Taskera system. Before using the platform, a guest can register and verify a customer account through the OTP/Email Service. After becoming a registered user, the customer can log in to the system, recover a forgotten password when necessary, and manage notification preferences for receiving service-related updates.

The central functions of this module focus on the customer service lifecycle. Customers can book a home service, manage their service request history, track job progress, and review a completed service after the job has been finished. The diagram also shows the interaction between customer functions and supporting system services. The Matching/Dispatch Service supports provider matching during service booking, File Storage is used when customers upload images or attachments, the WebSocket Gateway supports real-time chat messages, and the FCM Service delivers push notifications. Overall, this diagram clarifies how customers interact with Taskera from account registration to service booking, job tracking, communication, notification handling, and post-service review.

The Provider Module Use Case Diagram describes the main functions related to service providers in the Taskera system. A Provider Applicant can register as a service provider by submitting the required profile

Troy University Resources

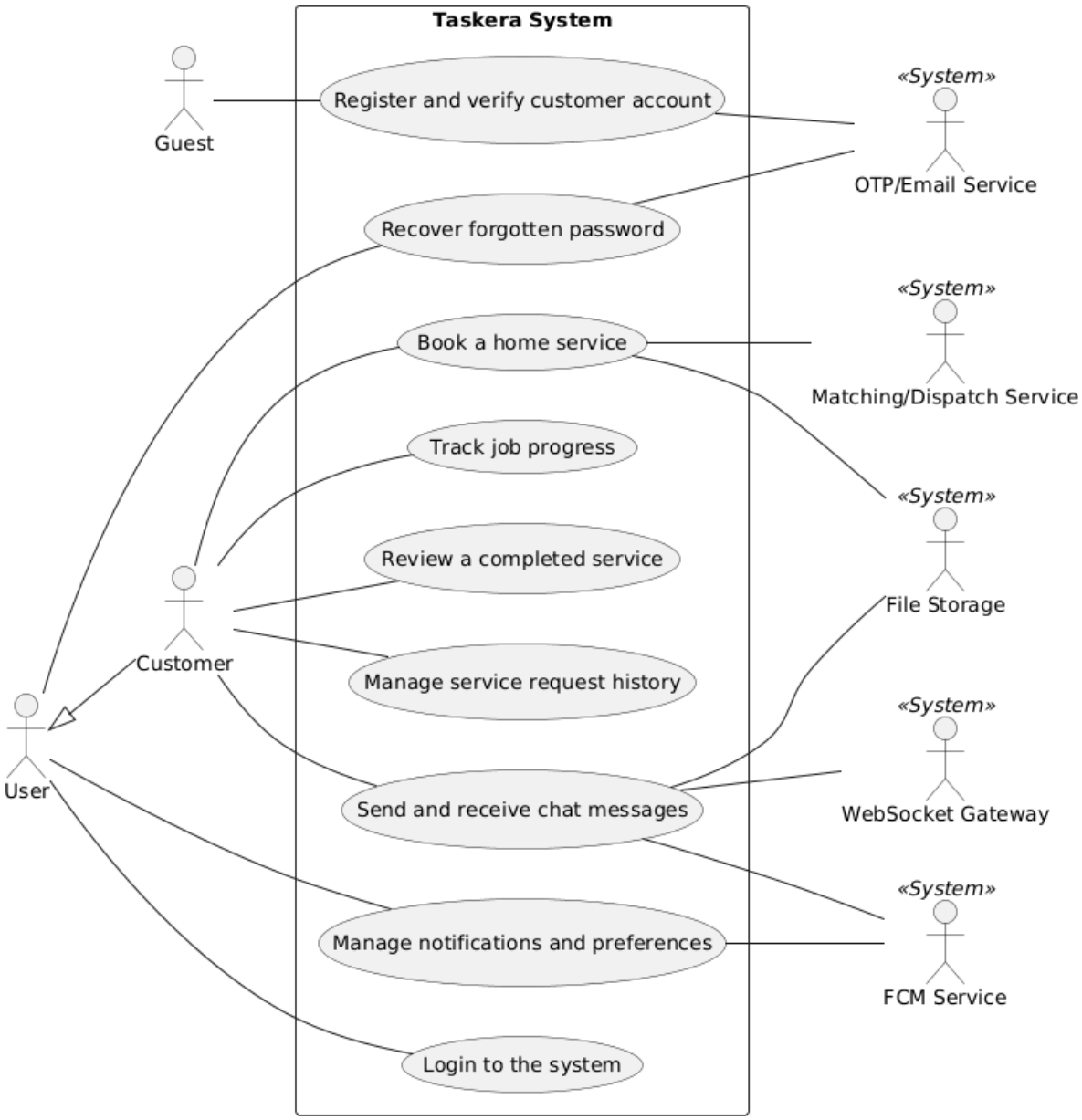


Figure 2-3: Customer Module Use Case Diagram

and verification information. After the registration process, a provider can log in to the system, recover a forgotten password when necessary, and manage notification preferences as a registered user.
The main provider-related functions focus on job fulfillment and service operation. Providers can manage their service profile, accept or decline dispatched jobs, update job status during the service process, and communicate with customers through chat messages. The diagram also shows the supporting system services involved in provider operations. The Matching/Dispatch Service supports job assignment, the FCM Service delivers job offer and status notifications, the WebSocket Gateway supports real-time communication, and File Storage is used for uploaded documents or related service images. Overall, this diagram clarifies how providers participate in Taskera from registration and profile management to job acceptance, progress updating, communication, and notification handling.

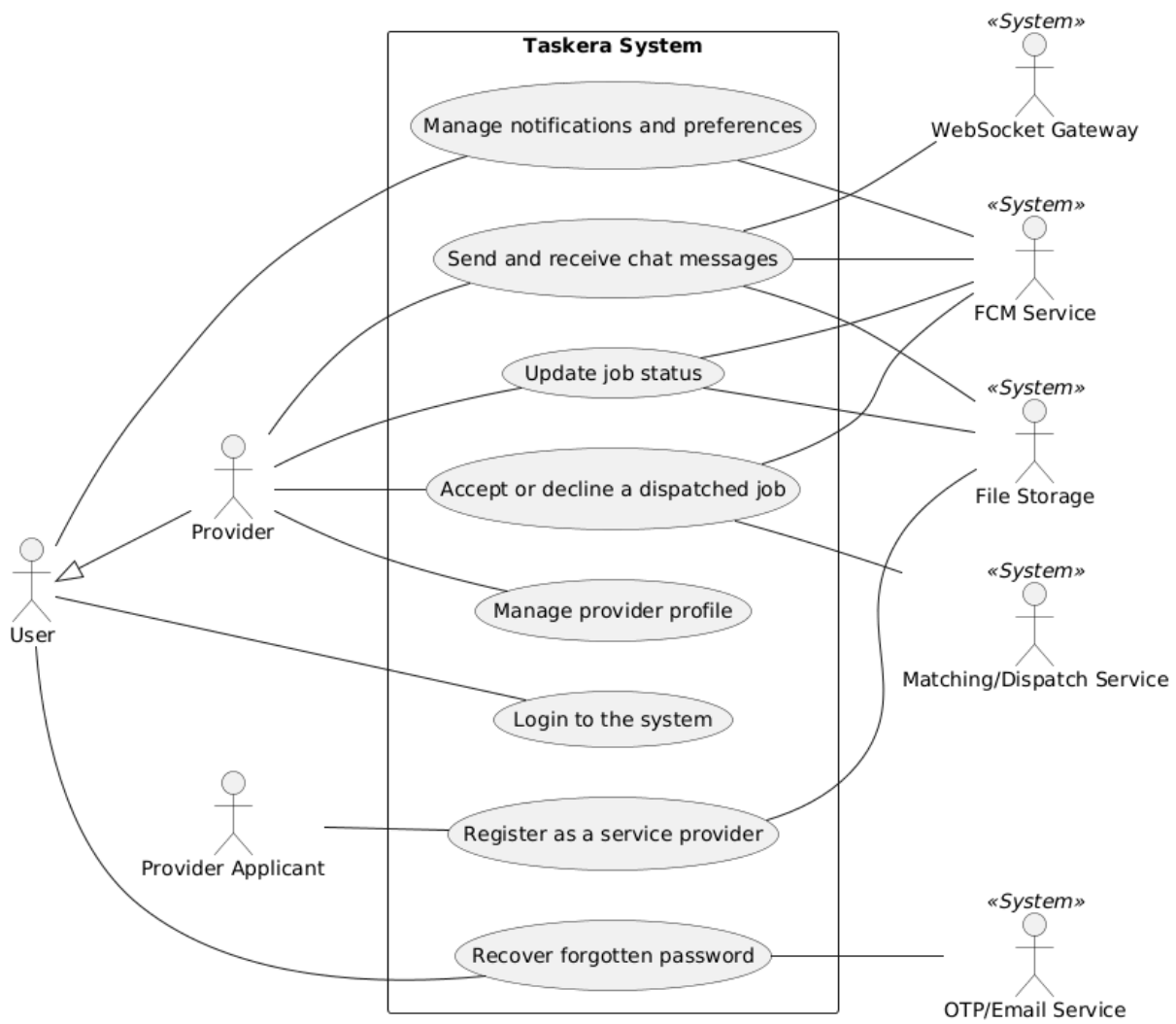


Figure 2-4: Provider Module Use Case Diagram

2.3. Use Case Specification

Table 2-5: Use Case Specification - UC-01 Register Customer Account

Use case ID	UC-01	**Use case name**	Register customer account
Actor	Customer		
Description	Allows the Customer to register and verify customer account. APIs: POST /auth/register, POST /auth/verify-otp, POST /auth/resend-otp.		
Precondition	The customer has not registered an account or has not completed the OTP verification process.		

Troy University Resources

Main flow (Success)

No.	Performed by	Action
1	Customer	The customer opens the registration screen.
2	Customer	The customer enters account information and submits the registration form.
3	System	The system validates the input data and creates a pending customer account.
4	System	The system sends an OTP code to the customer's email address.
5	Customer	The customer enters the OTP code for verification.
6	System	The system verifies the OTP code and activates the customer account.

Alternative flow

No.	Performed by	Action
3a	System	If the input data is invalid, the system displays an error message and asks the customer to re-enter the information.
3b	System	If the email is already registered, the system displays an error message indicating that the email is currently in use.
5a	System	If the OTP code is invalid or expired, the system displays an error message and allows the customer to request a new OTP code.

Postcondition A new customer account is created and verified successfully in the system.

* The input data for personal information includes the following data fields:

No.	Data field	Description	Required?	Validation condition	Example
1	Full Name	The full name of the customer	Yes	Must be a valid string, maximum 50 characters	John Doe
2	Email	The email address of the customer	Yes	Must be a valid email format	johndoe@example.com
3	Phone Number	The contact phone number of the customer	Yes	Must be a valid phone number format	+84901234567
4	Password	The password for the new account	Yes	Minimum 8 characters	SecurePass123!
5	OTP Code	The one-time password sent to the customer's email	Yes	Must be a 6-digit numeric code	123456

Table 2-8: Use Case Specification - UC-02 Log in to the System

Use case ID	UC-02	**Use case name**	Log in to the system
Actor	Customer, Provider, Admin		
Description	Allows the Customer to login to the system. APIs: POST /auth/login, POST /auth/refresh, POST /auth/logout.		
Precondition	The user has a registered and verified account in the system.		

	No.	Performed by	Action
Main flow (Success)	1	User	The user opens the login screen.
	2	User	The user enters their email address and password, then submits the login request.
	3	System	The system validates the provided email and password against the database.
	4	System	The system generates authentication tokens for the user session.
	5	System	The system authenticates the user and redirects them to the appropriate home screen based on their role.

	No.	Performed by	Action
Alternative flow	3a	System	If the email or password is incorrect, the system displays an error message indicating invalid credentials and asks the user to try again.
	3b	System	If the user account is not verified, the system displays a notification and prompts the user to complete the email verification process.

Postcondition	The user is successfully authenticated and granted access to the system.

* The input data for personal information includes the following data fields:

No.	Data field	Description	Required?	Validation condition	Example
1	Email	The registered email address of the user.	Yes	Must be a valid email format.	user@example.com
2	Password	The password associated with the user account.	Yes	Must not be empty.	SecurePass123!

Table 2-11: Use Case Specification - UC-03 Recover Forgotten Password

Use case ID	UC-03	**Use case name**	Recover forgotten password
Actor	Customer, Provider		
Description	Allows the Customer to recover forgotten password. APIs: POST /auth/forgot-password, POST /auth/reset-password, GET /auth/reset-password-page.		
Precondition	The user has a registered account in the system and has forgotten their password.		

	No.	Performed by	Action
Main flow (Success)	1	User	The user navigates to the forgot password screen.
	2	User	The user enters their registered email address and submits the request.
	3	System	The system verifies the email address exists and sends a password reset OTP to the provided email.
	4	User	The user enters the received OTP and a new password.
	5	System	The system validates the OTP and updates the user's password.

Troy University Resources

Alternative flow	No.	Performed by	Action
	3a	System	If the email address is not found in the system, the system displays an error message and prompts the user to enter a valid email address.
	5a	System	If the OTP is invalid or expired, the system displays an error message and requests the user to enter a valid OTP or request a new one.
Postcondition	The user's password is successfully updated in the system, allowing them to log in with the new credentials.		

* The input data for personal information includes the following data fields:

No.	Data field	Description	Required?	Validation condition	Example
1	Email	The registered email address of the user.	Yes	Must be a valid email format.	customer@example.com
2	OTP Code	The one-time password sent to the user's email.	Yes	Must be a numeric string of the correct length.	123456
3	New Password	The new password chosen by the user.	Yes	Must meet minimum length and complexity requirements.	NewSecurePass123!

Table 2-14: Use Case Specification - UC-04 Book a Home Service

Use case ID	UC-04	**Use case name**	Book a home service
Actor	Customer		
Description	Allows the Customer to book a home service. APIs: GET /service-categories/:id, PATCH /service-categories/:id, GET /service-categories/:id/booking-schema (+3 more).		
Precondition	Customer is authenticated and has the appropriate permissions.		
Main flow (Success)	No.	Performed by	Action
	1	Customer	Opens the Service Booking Page screen.
	2	System	Validates the input data.
	3	System	Processes the request and updates the database.
	4	System	Returns a success response to the client.
	5	System	Updates the UI to reflect the changes.
Alternative flow	No.	Performed by	Action
	1a	System	If the resource is not found (Customer profile not found), the system returns HTTP 404.
	2a	System	If the request is invalid (Invalid or inactive category), the system returns HTTP 400.
	3a	System	If the request is invalid, the system returns HTTP 400.
	4a	System	If the resource is not found (Service request not found), the system returns HTTP 404.
	5a	System	If access is denied (Access denied), the system returns HTTP 403.
	6a	System	If access is denied (Only the customer can cancel their request), the system returns HTTP 403.
	7a	System	If access is denied (Only providers can view available jobs), the system returns HTTP 403.
Postcondition	The system state is updated successfully according to the request.		

Table 2-16: Use Case Specification - UC-05 Manage Service Request History

Use case ID	UC-05		**Use case name**	Manage service request history
Actor	Customer			
Description	Allows the Customer to manage service request history. APIs: GET /requests/my, GET /requests/:id, PATCH /requests/:id/cancel.			
Precondition	Customer is authenticated and has the appropriate permissions.			
Main flow (Success)	No.	Performed by	Action	
	1	Customer	Initiates the request.	
	2	System	Validates the input data.	
	3	System	Processes the request and updates the database.	
	4	System	Returns a success response to the client.	
Alternative flow	No.	Performed by	Action	
	1a	System	If the resource is not found (Customer profile not found), the system returns HTTP 404.	
	2a	System	If the request is invalid (Invalid or inactive category), the system returns HTTP 400.	
	3a	System	If the request is invalid, the system returns HTTP 400.	
	4a	System	If the resource is not found (Service request not found), the system returns HTTP 404.	
	5a	System	If access is denied (Access denied), the system returns HTTP 403.	
	6a	System	If access is denied (Only the customer can cancel their request), the system returns HTTP 403.	
	7a	System	If access is denied (Only providers can view available jobs), the system returns HTTP 403.	
Postcondition	The system state is updated successfully according to the request.			

Table 2-18: Use Case Specification - UC-06 Register as a Service Provider

Use case ID	UC-06		**Use case name**	Register as a service provider
Actor	Provider			
Description	Allows the Provider to register as a service provider. APIs: POST /auth/register, GET /providers/me, PATCH /providers/me (+2 more).			
Precondition	The user has logged into the system and navigated to the provider registration module.			
Main flow (Success)	No.	Performed by	Action	
	1	User	Selects the option to register as a service provider.	
	2	System	Displays the provider onboarding and registration form.	
	3	User	Enters profile details, uploads identity documents, and provides a professional biography.	
	4	User	Selects service categories, specific skills, and defines the service area.	
	5	User	Submits the provider registration request.	
	6	System	Validates the submitted data, including document formats, selected categories, and service areas.	
	7	System	Creates the provider profile, saves the associated skills and service areas, and updates the user role.	
	8	System	Displays a success message and redirects the user to the provider dashboard.	

Troy University Resources

	No.	Performed by	Action
	6a	System	If the submitted data is incomplete or invalid, the system displays an error message and highlights the fields requiring correction.
Alternative flow	6b	System	If the selected service category or skill does not exist in the system, the system displays an error message and prompts the user to select valid options.
	7a	System	If a provider profile already exists for the user, the system aborts the creation process, notifies the user, and redirects them to their existing profile.
Postcondition	A service provider profile is successfully created and stored in the system, and the user is granted provider access.		

* The input data for personal information includes the following data fields:

No.	**Data field**	**Description**	**Required?**	**Validation condition**	**Example**
1	displayName	The public display name of the service provider	Yes	String, maximum 100 characters	John Doe Plumbing
2	bio	A short professional biography or introduction	No	String, maximum 500 characters	Expert plumber with over 10 years of experience in residential repairs.
3	idDocType	The type of identity document provided	Yes	Valid enum value (for example, National ID, Passport)	National ID
4	idFrontUrl	URL to the front image of the identity document	Yes	Valid URL format	https: //storage.example.com/id-front-123.jpg
5	idBackUrl	URL to the back image of the identity document	Yes	Valid URL format	https: //storage.example.com/id-back-123.jpg
6	categoryId	The unique identifier of the primary service category	Yes	Valid UUID	550e8400-e29b-41d4-a716-446655440000
7	years Experience	Number of years of professional experience	Yes	Numeric, greater than or equal to 0	5
8	certificateUrls	URLs to uploaded professional certificates	No	Array of valid URLs	[”https: //storage.example.com/cert-1.jpg”]
9	areaName	The name of the primary service area	Yes	String, maximum 100 characters	District 1, Ho Chi Minh City
10	radiusKm	The maximum travel distance for providing services	Yes	Numeric, greater than 0	15
11	isNationwide	Indicates if the provider offers services nationwide	No	Boolean	false

Table 2-21: Use Case Specification - UC-07 Accept or Decline a Dispatched Job

Use case ID	UC-07	**Use case name**	Accept or decline a dispatched job
Actor	Provider		
Description	Allows the Provider to accept or decline a dispatched job. APIs: POST /dispatch/:requestId/run, GET /dispatch/:requestId/candidates, GET /dispatch/my-offers (+2 more).		
Precondition	The provider has logged into the system and has received a pending job dispatch offer.		

Main flow (Success)

No.	Performed by	Action
1	Provider	Navigates to the job offers screen.
2	System	Displays the list of pending job dispatch offers.
3	Provider	Selects a specific job offer to view its details.
4	System	Displays the job offer details, including service type, location, and schedule.
5	Provider	Confirms the acceptance of the job offer.
6	System	Validates the provider's availability and the current status of the job offer.
7	System	Assigns the service request to the provider and updates the offer status to accepted.
8	System	Updates the user interface to reflect the accepted job and removes it from the pending offers list.

Alternative flow

No.	Performed by	Action
5a	Provider	If the provider chooses to decline the job offer, they select the decline option and optionally enter a response note.
5b	System	Updates the offer status to declined for this provider and evaluates the next available candidate.
6a	System	If the job offer has already been accepted by another provider or has expired, the system displays an error message and removes the offer from the view.
6b	System	If the provider has a conflicting ongoing job, the system displays a conflict error message and prevents the acceptance.
6c	System	If the service request or provider profile is not found, the system displays an error message and aborts the operation.

Postcondition	The system updates the dispatch offer status. If accepted, the service request is assigned to the provider; if declined, the system proceeds to the next available candidate.

* The input data for personal information includes the following data fields:

No.	Data field	Description	Required?	Validation condition	Example
1	providerProfileId	The unique identifier of the provider responding to the offer.	Yes	Must be a valid UUID.	a1b2c3d4-e5f6-7890-1234-56789abcdef0
2	responseNote	Optional note provided by the provider when declining the job offer.	No	String, maximum length constraints apply.	Distance is too far from my current location.
3	Note	Additional notes submitted during the acceptance or rejection process.	No	String.	I can arrive in 15 minutes.
4	topN	Configuration for the maximum number of candidates to dispatch to (used during dispatch generation).	No	Numeric integer.	5

Troy University Resources

5	timeoutSeconds	The duration in seconds before the dispatch offer expires.	No	Numeric integer, greater than 0.	300

Table 2-24: Use Case Specification - UC-08 Track and Update Job Progress

Use case ID	UC-08		**Use case name**	Track and update job progress
Actor	Provider, Customer			
Description	Allows the Provider to track and update job progress. APIs: GET /jobs/:id, PATCH /jobs/:id/status, PATCH /jobs/:id/progress (+3 more).			
Precondition	The provider or customer has logged into the system and an active job request exists.			
Main flow (Success)	No.	Performed by	Action	
	1	Provider	Accesses the job tracking screen for an assigned job.	
	2	System	Retrieves and displays the current job details, timeline, and location.	
	3	Provider	Submits updated job information, such as status, progress percentage, location, or final price.	
	4	System	Validates the submitted job update information and user permissions.	
	5	System	Updates the job record in the database and logs the status change.	
	6	System	Refreshes the tracking interface to display the updated job progress.	
Alternative flow	No.	Performed by	Action	
	2a	System	If the job record or associated user profile is not found, the system displays an error message and redirects the user to the job list.	
	4a	System	If an unassigned provider attempts to update the job location, the system denies the request and displays an error message.	
	4b	System	If the provider attempts to update the location while the job status is not 'On the way', the system rejects the update and displays an error message.	
	4c	System	If the provider attempts to update the progress percentage while the job status is not 'In progress', the system rejects the update and displays an error message.	
	4d	System	If an unauthorized user attempts to update the final price, the system denies the request and displays an error message.	
	4e	System	If the customer attempts to switch the provider while the provider is on the way without providing a reason, the system prompts the customer to enter a cancellation reason.	
Postcondition	The job progress, status, location, or price is updated in the system, and the changes are reflected on the tracking interface.			

* The input data for personal information includes the following data fields:

No.	Data field	Description	Required?	Validation condition	Example
1	status	The current status of the job	Yes	Must be a valid job status enumeration (for example, ON_THE_WAY, IN_PROGRESS, COMPLETED)	IN_PROGRESS

2	note	Additional notes regarding the status update	No	Must be a string, maximum length of 500 characters	Arrived at the customer's location and starting the inspection.
3	progressPercent	The completion percentage of the job	No	Must be a numeric value between 0 and 100	50
4	cancelReason	The reason for cancelling the job or switching the provider	Yes (if cancelling or switching provider)	Must be a non-empty string	Provider is unresponsive.
5	latitude	The current latitude coordinate of the provider	Yes (if updating location)	Must be a valid latitude numeric format	10.762622
6	longitude	The current longitude coordinate of the provider	Yes (if updating location)	Must be a valid longitude numeric format	106.660172
7	finalPrice	The final agreed price for the completed service	No	Must be a numeric value greater than or equal to 0	150000

Table 2-27: Use Case Specification - UC-09 Review a Completed Service

Use case ID	UC-09	**Use case name**	Review a completed service
Actor	Customer		
Description	Allows the Customer to review a completed service. APIs: POST /jobs/:jobId/reviews, GET /jobs/:jobId/reviews, GET /providers/:providerProfileId/reviews (+1 more).		
Precondition	The customer has logged into the system and has at least one completed service job.		

Main flow (Success)

No.	Performed by	Action
1	Customer	The customer selects a completed service job to review.
2	System	The system displays the review submission form.
3	Customer	The customer enters the rating, comment, and optionally uploads images.
4	Customer	The customer submits the review.
5	System	The system validates the provided information and verifies that the job is completed and not yet reviewed.
6	System	The system saves the review and updates the provider's overall rating.
7	System	The system displays a success message to the customer.

Alternative flow

No.	Performed by	Action
5a	System	If the job is not found, the system displays an error message and aborts the process.
5b	System	If the customer is not the owner of the job, the system denies access and displays an error message.
5c	System	If the job is not in the completed state, the system displays an error message stating that only completed jobs can be reviewed.
5d	System	If the job has already been reviewed, the system displays an error message indicating that a review already exists for this job.

Postcondition	The review is successfully submitted, linked to the completed job, and visible on the provider's profile.

* The input data for personal information includes the following data fields:

No.	Data field	Description	Required?	Validation condition	Example
1	rating	The score given to the provider for the service	Yes	Numeric value between 1 and 5	5

Troy University Resources

2	comment	Textual feedback about the service quality	No	String, maximum length 500 characters	Great service, very professional and on time.
3	imageUrls	List of image URLs attached to the review	No	Array of valid URL strings	[”https: //example.com/images/review1.jpg”]

Table 2-30: Use Case Specification - UC-10 Approve or Reject Provider Application

Use case ID	UC-10	**Use case name**	Approve or reject provider application
Actor	Admin		
Description	Allows the Admin to approve or reject provider application. APIs: GET /admin/providers, GET /admin/providers/pending, PATCH /admin/providers/:id/approve (+3 more).		
Precondition	The Admin has logged into the system and accessed the provider management dashboard.		

Main flow (Success)

No.	Performed by	Action
1	Admin	Navigates to the pending provider applications screen.
2	System	Retrieves and displays a list of pending provider applications.
3	Admin	Selects a specific provider application to review.
4	System	Displays the provider's detailed information, including identity documents and certificates.
5	Admin	Selects the option to approve the application.
6	System	Updates the provider's status to approved in the database.
7	System	Displays a success message and removes the application from the pending list.

Alternative flow

No.	Performed by	Action
4a	System	If the selected provider application is not found, the System displays an error message and returns the Admin to the pending list.
5a	Admin	If the Admin decides to reject the application, the Admin selects the reject option and provides a rejection reason.
5b	System	The System updates the provider's status to rejected in the database.
5c	System	The System displays a success message and removes the application from the pending list.

Postcondition	The provider application is processed, and the provider's status is updated to approved or rejected.

* The input data for personal information includes the following data fields:

No.	Data field	Description	Required?	Validation condition	Example
1	status	The approval decision for the provider application.	Yes	Must be a valid status value (for example, APPROVED, REJECTED).	APPROVED
2	rejectionReason	The reason provided by the Admin if the application is rejected.	No	String, maximum 500 characters.	Identity document is expired or blurry.

Table 2-33: Use Case Specification - UC-11 Send and Receive Chat Messages

Use case ID	UC-11	**Use case name**	Send and receive chat messages
Actor	Customer, Provider		
Description	Allows the Customer to send and receive chat messages. APIs: GET /chat/conversations, GET /chat/unread, GET /jobs/:jobId/messages (+3 more).		
Precondition	The user (Customer or Provider) has logged into the system and is part of an active job.		

	No.	Performed by	Action
Main flow (Success)	1	User	Selects an active job conversation from the chat list.
	2	System	Displays the chat history for the selected job.
	3	User	Enters the message content or attaches an image.
	4	User	Submits the message.
	5	System	Validates the message content and verifies the user's access to the job chat.
	6	System	Saves the message to the database and updates the conversation.
	7	System	Displays the newly sent message in the chat interface.

	No.	Performed by	Action
Alternative flow	5a	System	If the job is not found, the system displays an error message indicating the job does not exist and aborts the operation.
	5b	System	If the user does not have access to the job chat, the system displays an access denied error message and aborts the operation.

Postcondition	The chat message is successfully sent, stored in the system, and delivered to the recipient.

* The input data for personal information includes the following data fields:

No.	Data field	Description	Required?	Validation condition	Example
1	content	The text content of the chat message	No	Must be a string, maximum length constraints apply	I will arrive in 10 minutes.
2	imageUrl	The URL of the attached image	No	Must be a valid URL format	https://example.com/images/chat-attachment.jpg

Table 2-36: Use Case Specification - UC-12 Manage Notifications and Preferences

Use case ID	UC-12	**Use case name**	Manage notifications and preferences
Actor	Customer, Provider		
Description	Allows the Customer to manage notifications and preferences. APIs: POST /fcm/device-token, DELETE /fcm/device-token, GET /notifications/me (+5 more).		
Precondition	User is authenticated and has the appropriate permissions.		

	No.	Performed by	Action
Main flow (Success)	1	User	Initiates the request.
	2	System	Validates the input data.
	3	System	Processes the request and updates the database.
	4	System	Returns a success response to the client.
Alternative flow			

Postcondition	The system state is updated successfully according to the request.

Troy University Resources

Table 2-38: Use Case Specification - UC-13 Manage Customer Profile and Addresses

Use case ID	UC-13		**Use case name**	Manage customer profile and addresses
Actor	Customer			
Description	Allows the Customer to manage customer profile and addresses. APIs: GET /customers/me, PATCH /customers/me, GET /customers/addresses (+5 more).			
Precondition	The Customer has logged in to the system with a valid account.			
Main flow (Success)	No.	Performed by	Action	
	1	Customer	Opens the profile and address management screen.	
	2	System	Retrieves and displays the current customer profile and saved addresses.	
	3	Customer	Selects the profile information or address information to update.	
	4	Customer	Enters updated information such as full name, phone number, default address, detailed address, or location.	
	5	System	Validates the submitted profile or address information.	
	6	System	Updates the customer profile or address record in the database.	
	7	System	Displays the updated profile and address list to the Customer.	
Alternative flow	No.	Performed by	Action	
	1	System	If required information is missing or invalid, displays validation errors and asks the Customer to correct the data.	
	2	System	If the selected address is being used by an active service request, prevents deletion and displays an error message.	
	3	System	If the database update fails, keeps the previous information and displays an error message.	
Postcondition	Customer profile information and saved addresses are updated successfully, or the previous data remains unchanged if the operation fails.			

* The input data for personal information includes the following data fields:

No.	Data field	Description	Required?	Validation condition	Example
1	customerId	Unique identifier of the customer	Yes	Must exist in the system	CUS001
2	fullName	Customer’s full name	Yes	2–100 characters	Nguyen Van A
3	phoneNumber	Customer’s phone number	Yes	Valid phone number format	0912345678
4	email	Customer’s email address	No	Valid email format if provided	customer@example.com
5	addressTitle	Name of saved address	No	Maximum 100 characters	Home
6	detailedAddress	Detailed service address	Yes	Must not be empty	123 Tran Dai Nghia, Hanoi
7	latitude	Latitude of address location	No	Valid latitude value	21.005
8	longitude	Longitude of address location	No	Valid longitude value	105.843
9	isDefault	Whether this address is the default address	No	Boolean value	true

Table 2-41: Use Case Specification - UC-14 Manage Provider Profile and Availability

Use case ID	UC-14	**Use case name**	Manage provider profile and availability
Actor	Provider		
Description	Allows the Provider to manage provider profile and availability. APIs: GET /providers/me, PATCH /providers/me, GET /providers/dashboard-stats (+7 more).		
Precondition	The Provider has logged in and has an existing provider profile.		

Main flow (Success)	No.	Performed by	Action
	1	Provider	Opens the provider profile or availability screen.
	2	System	Retrieves and displays the current provider profile, service categories, skills, service areas, certificates, and availability status.
	3	Provider	Updates profile information, professional experience, service area, skills, or service categories.
	4	Provider	Changes online/offline availability if necessary.
	5	System	Validates the updated provider information.
	6	System	Saves the updated provider profile and availability data.
	7	System	Displays the updated provider information and dashboard statistics.

Alternative flow	No.	Performed by	Action
	1	System	If required provider information is missing, displays validation errors and rejects the update.
	2	System	If the provider profile has not been approved, restricts online availability and job-receiving functions.
	3	System	If the Provider attempts to go online without required service information, prevents the action and displays a warning message.

Postcondition	Provider profile, service information, and availability status are updated successfully in the system.

* The input data for personal information includes the following data fields:

No.	Data field	Description	Required?	Validation condition	Example
1	providerId	Unique identifier of the provider	Yes	Must exist in the system	PRO001
2	bio	Provider introduction	No	Maximum 500 characters	Experienced electrician
3	yearsOf Experience	Provider’s working experience	No	Must be a non-negative number	3
4	service Categories	Categories that the provider can serve	Yes	Must contain at least one valid category	Electrical repair
5	skills	Provider’s professional skills	No	Text or list format	Wiring, socket repair
6	serviceArea	Working area of the provider	Yes	Must be a supported area	Thanh Xuan, Hanoi
7	certificateUrls	Uploaded certificate images or files	No	Must be valid file URLs	certificate1.jpg
8	isOnline	Provider availability status	Yes	Boolean value	true

Troy University Resources

Table 2-44: Use Case Specification - UC-15 Browse Available Service Requests

Use case ID	UC-15		**Use case name**	Browse available service requests (job board)
Actor	Provider			
Description	Allows the Provider to browse available service requests (job board). APIs: GET /requests/available, GET /requests/:id.			
Precondition	The Provider has logged in, has an approved provider profile, and is allowed to view available service requests.			
Main flow (Success)	No.	Performed by	Action	
	1	Provider	Opens the available service requests screen.	
	2	System	Checks the Provider's approval status, service categories, service area, and availability.	
	3	System	Retrieves available service requests that match the Provider's service information.	
	4	System	Displays the list of available service requests with basic information.	
	5	Provider	Selects a service request to view details.	
	6	System	Displays detailed request information, including service type, address area, preferred time, description, and estimated price.	
	7	Provider	Reviews the request and decides whether to take further action according to the job workflow.	
Alternative flow	No.	Performed by	Action	
	1	System	If no suitable request is available, displays an empty-state message.	
	2	System	If the Provider is not approved, denies access to the available request list.	
	3	System	If the selected request is no longer available, displays a message and refreshes the list.	
Postcondition	The Provider can view matching service requests and inspect request details. No request data is changed unless the Provider performs another action.			

* The input data for personal information includes the following data fields:

No.	Data field	Description	Required?	Validation condition	Example
1	providerId	Unique identifier of the provider	Yes	Must exist in the system	PRO001
2	serviceCategoryId	Service category filter	No	Must be a valid category ID	CAT_ELECTRIC
3	serviceArea	Area used to filter requests	No	Must be a supported area	Thanh Xuan
4	requestStatus	Status of service requests to display	No	Must be an available status	PENDING
5	page	Page number for pagination	No	Must be greater than 0	1
6	limit	Number of records per page	No	Must be within allowed range	10

- **Use case specification UC-16 "View job history"**

Table 2-47: Use Case Specification - UC-16 View Job History

Use case ID	UC-16	**Use case name**	View job history
Actor	Provider, Customer		
Description	Allows the Provider to view job history. APIs: GET /jobs/my, GET /jobs/my-customer.		
Precondition	The user has logged in to the system as a Customer or Provider.		

Main flow (Success)

No.	Performed by	Action
1	User	Opens the job history screen.
2	System	Identifies the user role as Customer or Provider.
3	System	Retrieves jobs related to the current user.
4	System	Displays the job history list with service type, time, status, and related user information.
5	User	Selects a job from the list.
6	System	Displays detailed job information, including request details, job progress, status, price, and review information if available.

Alternative flow

No.	Performed by	Action
1	System	If the user has no job records, displays an empty-state message.
2	System	If the user attempts to view a job that does not belong to them, denies access.
3	System	If the selected job cannot be loaded, displays an error message.

Postcondition	The user can view their job history and inspect the details of selected jobs.

* The input data for personal information includes the following data fields:

No.	Data field	Description	Required?	Validation condition	Example
1	userId	Unique identifier of the current user	Yes	Must exist in the system	USR001
2	userRole	Role of the current user	Yes	Must be Customer or Provider	Customer
3	jobId	Selected job identifier	No	Must belong to current user	JOB001
4	statusFilter	Job status filter	No	Must be a valid job status	COMPLETED
5	page	Page number for pagination	No	Must be greater than 0	1
6	limit	Number of records per page	No	Must be within allowed range	10

Troy University Resources

Table 2-50: Use Case Specification - UC-17 Manage Provider Posts

Use case ID	UC-17		**Use case name**	Manage provider posts (portfolio)
Actor	Provider			
Description	Allows the Provider to manage provider posts (portfolio). APIs: POST /providers/posts, GET /providers/posts/me, PATCH /providers/posts/:id (+3 more).			
Precondition	The Provider has logged in and has an existing provider profile.			
Main flow (Success)	No.	Performed by	Action	
	1	Provider	Opens the provider post management screen.	
	2	System	Displays the list of posts created by the Provider.	
	3	Provider	Chooses to create a new post or edit an existing post.	
	4	Provider	Enters post title, content, service category, and optional images.	
	5	System	Validates the post content and uploaded images.	
	6	System	Uploads images to file storage if images are provided.	
	7	System	Creates or updates the provider post in the database.	
	8	System	Displays the updated provider post list.	
Alternative flow	No.	Performed by	Action	
	1	System	If the title or content is missing, displays validation errors.	
	2	System	If an uploaded image has an unsupported type or exceeds the size limit, rejects the file.	
	3	Provider	Selects an existing post and confirms deletion.	
	4	System	Deletes the selected post and updates the post list.	
Postcondition	Provider posts are created, updated, or deleted successfully.			

* The input data for personal information includes the following data fields:

No.	Data field	Description	Required?	Validation condition	Example
1	providerId	Unique identifier of the provider	Yes	Must exist in the system	PRO001
2	postId	Unique identifier of the post	No	Required when updating or deleting	POST001
3	title	Title of the provider post	Yes	2–150 characters	Before and after repair
4	content	Main content of the post	Yes	Must not be empty	Completed electrical repair for customer
5	serviceCategory Id	Related service category	No	Must be a valid category ID	CAT_ELECTRIC
6	imageUrls	Images attached to the post	No	Must be valid image URLs	post1.jpg

Table 2-53: Use Case Specification - UC-18 View Provider Public Profile

Use case ID	UC-18		**Use case name**	View provider public profile
Actor	Customer			
Description	Allows the Customer to view provider public profile. APIs: GET /providers/top, GET /providers/:id, GET /providers/:id/stats.			
Precondition	The Customer has access to provider information from a provider list, job detail screen, booking screen, or review screen.			

Main flow (Success)	No.	Performed by	Action
	1	Customer	Selects a provider profile from a provider list, job detail screen, or recommendation section.
	2	System	Retrieves the selected provider's public profile information.
	3	System	Retrieves provider statistics, rating, completed jobs, reviews, and posts if available.
	4	System	Displays provider name, experience, service categories, rating, reviews, and public portfolio information.
	5	Customer	Reviews the provider information to evaluate service quality and reliability.
Alternative flow	No.	Performed by	Action
	1	System	If the selected provider profile does not exist, displays an error message.
	2	System	If the provider is not approved or not public, hides the profile and displays a message.
	3	System	If the provider has no reviews or posts, displays an empty-state message for those sections.
Postcondition	The Customer can view the provider's public information without modifying system data.		

* The input data for personal information includes the following data fields:

No.	Data field	Description	Required?	Validation condition	Example
1	providerProfileId	Unique identifier of the provider profile	Yes	Must exist in the system	PRO001
2	customerId	Unique identifier of the viewing customer	No	Must exist if user is logged in	CUS001
3	reviewPage	Page number for review pagination	No	Must be greater than 0	1
4	reviewLimit	Number of reviews per page	No	Must be within allowed range	5
5	includeStats	Whether to include provider statistics	No	Boolean value	true

Troy University Resources

Table 2-56: Use Case Specification - UC-19 Manage Service Categories

Use case ID	UC-19	**Use case name**	Manage service categories
Actor	Admin		
Description	Allows the Admin to manage service categories. APIs: GET /service-categories/:id, PATCH /service-categories/:id.		
Precondition	The Admin has logged in and has permission to manage service categories.		

Main flow (Success)

No.	Performed by	Action
1	Admin	Opens the service category management screen.
2	System	Displays the list of existing service categories.
3	Admin	Selects a category to view, create, update, disable, or delete.
4	Admin	Enters category information such as name, description, icon, status, and booking schema.
5	System	Validates the category information and booking schema.
6	System	Saves the new or updated service category information.
7	System	Displays the updated service category list.

Alternative flow

No.	Performed by	Action
1	System	If the category name already exists, rejects the update and displays an error message.
2	System	If the category is linked to existing service requests or jobs, prevents deletion and recommends disabling the category instead.
3	System	If the booking schema format is invalid, rejects the update and displays validation errors.

Postcondition	Service category data and booking schema are created, updated, disabled, or deleted according to Admin permissions.

* The input data for personal information includes the following data fields:

No.	Data field	Description	Required?	Validation condition	Example
1	adminId	Unique identifier of the Admin	Yes	Must have Admin role	ADM001
2	categoryId	Unique identifier of the service category	No	Required when updating or deleting	CAT_CLEANING
3	categoryName	Name of the service category	Yes	Must be unique	Home Cleaning
4	description	Description of the service category	No	Maximum 500 characters	Cleaning service for homes
5	iconUrl	Icon representing the category	No	Must be a valid URL	cleaning.png
6	isActive	Whether the category is available	Yes	Boolean value	true
7	bookingSchema	Form schema used for booking details	No	Must be valid schema format	{duration, area}

Table 2-59: Use Case Specification - UC-20 Admin Manage Users

Use case ID	UC-20	**Use case name**	Admin manage users
Actor	Admin		
Description	Allows the Admin to admin manage users. APIs: GET /admin/users, GET /admin/users/:id, PATCH /admin/users/:id/suspend (+4 more).		
Precondition	The Admin has logged in and has user management permission.		

Main flow (Success)	No.	Performed by	Action
	1	Admin	Opens the user management screen.
	2	System	Displays the list of users with role, status, and basic account information.
	3	Admin	Searches or filters users by role, status, email, phone number, or name.
	4	Admin	Selects a user account to view details.
	5	System	Displays account information, profile information, and account status.
	6	Admin	Updates user status, suspends an account, reactivates an account, or edits allowed administrative fields.
	7	System	Validates the admin action and updates the user record.
	8	System	Displays the updated user information.
Alternative flow	No.	Performed by	Action
	1	System	If the selected user account does not exist, displays an error message.
	2	System	If the Admin tries to modify a restricted account or unauthorized field, denies the request.
	3	System	If the user has active jobs, displays a warning and may require confirmation before suspension.
Postcondition	User account information or status is updated according to the Admin's authorized action.		

* The input data for personal information includes the following data fields:

No.	Data field	Description	Required?	Validation condition	Example
1	adminId	Unique identifier of the Admin	Yes	Must have Admin role	ADM001
2	userId	Unique identifier of the selected user	Yes	Must exist in the system	USR001
3	roleFilter	Role used for filtering users	No	Must be a valid role	Customer
4	statusFilter	Account status filter	No	Must be a valid status	Active
5	searchKeyword	Keyword for searching users	No	Maximum 100 characters	nguyen
6	suspension Reason	Reason for suspending account	No	Required when suspending	Policy violation
7	updatedStatus	New status of the account	No	Must be valid account status	Suspended

Troy University Resources

Table 2-62: Use Case Specification - UC-21 Admin Manage Jobs and Requests

Use case ID	UC-21		**Use case name**	Admin manage jobs and requests
Actor	Admin			
Description	Allows the Admin to admin manage jobs and requests. APIs: GET /admin/requests, POST /admin/requests/:id/cancel, GET /admin/jobs (+4 more).			
Precondition	The Admin has logged in and has permission to manage jobs and service requests.			
Main flow (Success)	No.	Performed by	Action	
	1	Admin	Opens the job and request management screen.	
	2	System	Displays service requests and jobs with status, customer, provider, category, schedule, and created time.	
	3	Admin	Filters or searches records by status, service category, customer, provider, or time range.	
	4	Admin	Selects a request or job to view detailed information.	
	5	System	Displays request details, job status, assigned provider, dispatching information, and related logs.	
	6	Admin	Performs an allowed management action such as cancellation, status review, or provider reassignment support.	
	7	System	Validates the admin action.	
	8	System	Updates the request or job record and sends notifications to related users if necessary.	
Alternative flow	No.	Performed by	Action	
	1	System	If the selected request or job does not exist, displays an error message.	
	2	System	If the action is not allowed for the current job status, rejects the action and displays a warning.	
	3	System	If notification delivery fails, records the administrative action and logs the notification error.	
Postcondition	Job or request records are updated according to the Admin’s authorized action.			

* The input data for personal information includes the following data fields:

No.	Data field	Description	Required?	Validation condition	Example
1	adminId	Unique identifier of the Admin	Yes	Must have Admin role	ADM001
2	requestId	Unique identifier of the service request	No	Required for request actions	REQ001
3	jobId	Unique identifier of the job	No	Required for job actions	JOB001
4	statusFilter	Status used for filtering records	No	Must be valid request/job status	DISPATCHING
5	actionType	Type of admin action	Yes	Must be an allowed action	Cancel request
6	cancellation Reason	Reason for cancelling request/job	No	Required when cancelling	Customer request
7	updatedStatus	New status after admin action	No	Must be a valid status	CANCELLED

Table 2-65: Use Case Specification - UC-22 Switch Provider for a Job

Use case ID	UC-22	**Use case name**	Switch provider for a job
Actor	Customer		
Description	Allows the Customer to switch provider for a job. APIs: POST /dispatch/:requestId/retry, POST /dispatch/:requestId/manual-assign, POST /jobs/:id/switch-provider.		
Precondition	The Customer has logged in and has an active job assigned to a Provider. The job status allows provider switching.		

Main flow (Success)

No.	Performed by	Action
1	Customer	Opens the active job tracking screen.
2	Customer	Selects the option to switch provider.
3	System	Displays a confirmation message and asks for a switch reason if necessary.
4	Customer	Confirms the switch provider request.
5	System	Validates whether the job is eligible for provider switching.
6	System	Removes or cancels the current provider assignment according to business rules.
7	System	Changes the related service request back to dispatching or provider-matching status.
8	System	Searches for another suitable provider.
9	System	Sends job offer notifications to candidate providers.
10	System	Displays the updated job or request status to the Customer.

Alternative flow

No.	Performed by	Action
1	System	If the job is completed, cancelled, or not assigned, rejects the switch request.
2	System	If no replacement provider is available, keeps the request in waiting or dispatching status and notifies the Customer.
3	System	If automatic switching is not allowed by business rules, asks for Admin support.

Postcondition	The current provider assignment is removed or marked inactive, and the service request is redispatched or prepared for reassignment.

* The input data for personal information includes the following data fields:

No.	Data field	Description	Required?	Validation condition	Example
1	customerId	Unique identifier of the Customer	Yes	Must exist in the system	CUS001
2	jobId	Unique identifier of the active job	Yes	Must belong to current Customer	JOB001
3	requestId	Related service request identifier	Yes	Must be linked to the job	REQ001
4	currentProviderId	Identifier of current assigned Provider	Yes	Must be assigned to the job	PRO001
5	switchReason	Reason for switching provider	No	Maximum 500 characters	Provider cannot arrive on time
6	currentJobStatus	Current status of the job	Yes	Must allow switching	ACCEPTED

Troy University Resources

Table 2-68: Use Case Specification - UC-23 Upload Images

Use case ID	UC-23		**Use case name**	Upload images	
Actor	Customer, Provider				
Description	Allows the Customer to upload images. APIs: POST /upload/single, POST /upload/multiple, GET /upload/signed-url.				
Precondition	The user has logged in and is performing a function that allows image upload.				
Main flow (Success)	No.	Performed by	Action		
	1	User	Selects one or more image files from the device.		
	2	System	Receives the upload request and checks user authentication.		
	3	System	Validates file type, file size, and number of uploaded files.		
	4	System	Uploads valid image files to file storage.		
	5	System	Receives file URLs or file identifiers from file storage.		
	6	System	Stores or returns uploaded image information to the related function.		
	7	System	Displays uploaded images or image previews to the user.		
Alternative flow	No.	Performed by	Action		
	1	System	If the selected file is not an allowed image type, rejects the upload and displays an error message.		
	2	System	If the file size exceeds the limit, rejects the upload and displays an error message.		
	3	System	If file storage fails, cancels the upload and asks the user to try again.		
	4	System	If multiple upload partially fails, keeps successfully uploaded files and reports failed files.		
Postcondition	Valid images are uploaded and linked to the relevant service request, message, provider profile, post, or review.				

* The input data for personal information includes the following data fields:

No.	Data field	Description	Required?	Validation condition	Example
1	userId	Unique identifier of the uploading user	Yes	Must exist in the system	USR001
2	fileName	Name of the uploaded file	Yes	Must not be empty	repair_image.jpg
3	fileType	MIME type of the uploaded file	Yes	Must be an allowed image type	image/jpeg
4	fileSize	Size of the uploaded file	Yes	Must not exceed size limit	2 MB
5	uploadPurpose	Purpose of the upload	Yes	Must be a valid purpose	SERVICE_ REQUEST
6	relatedEntityId	ID of related request, job, post, or review	No	Required when linking image to entity	REQ001
7	imageFile	Actual image file selected by user	Yes	Must be a valid image file	photo.jpg

Table 2-71: Use Case Specification - UC-24 Register FCM Device Token

Use case ID	UC-24	**Use case name**	Register FCM device token
Actor	System		
Description	Allows the System to register fcm device token. APIs: POST /fcm/device-token, DELETE /fcm/device-token.		
Precondition	The user has installed the mobile application and the application has obtained an FCM device token.		

	No.	Performed by	Action
Main flow (Success)	1	Mobile application	Obtains an FCM device token from Firebase Cloud Messaging.
	2	Customer / Provider	Logs in to the Taskera application.
	3	Mobile application	Sends the device token, platform, and device information to the backend system.
	4	System	Validates the authenticated user and token information.
	5	System	Stores or updates the device token associated with the user account.
	6	System	Uses the registered token to send push notifications for job offers, job updates, chat messages, and system alerts.
	7	System	Returns a success response to the mobile application.

	No.	Performed by	Action
Alternative flow	1	System	If the device token is missing or invalid, rejects the request and returns a validation error.
	2	System	If notification permission is denied, records that the user cannot receive push notifications on the device.
	3	Mobile application	Sends a delete token request when the user logs out or disables notifications.
	4	System	Removes the device token from the user account.

Postcondition	The FCM device token is stored, updated, or removed successfully. The user can receive push notifications if notification permission is enabled.

* The input data for personal information includes the following data fields:

No.	Data field	Description	Required?	Validation condition	Example
1	userId	Unique identifier of the current user	Yes	Must exist in the system	USR001
2	deviceToken	FCM token of the device	Yes	Must not be empty	fcm_token_abc123
3	platform	Device platform	Yes	Must be Android, iOS, or Web	Android
4	deviceId	Identifier of the user device	No	Must be unique if provided	SM-S901U1
5	notification Permission	Notification permission status	No	Must be granted or denied	granted
6	actionType	Register or remove token action	Yes	Must be valid action type	REGISTER

2.4. Non-functional Requirements

2.4.1. Functionality

- In all use case event flows, if there is a database operation failure (for example, connection time-out or query error), a specific error message must be displayed to inform the actor that it is a system/database issue rather than a user input error.

Troy University Resources

- Use cases intended for Administrators, Customers, or Providers require the Actor to be authenticated and logged in with the corresponding role.
- The system must handle real-time state synchronization seamlessly using WebSockets (for example, instant chat message delivery, live GPS location tracking, and dynamic job status updates).
- If a service request is accepted by another Provider or cancelled by the Customer while a Provider is viewing the offer, the viewing Provider must be notified immediately via real-time events.
- General display formatting constraints:
  - Numbers must be right-aligned.
  - Text must be left-aligned.
  - Font: Arial 14, black color.
  - Background: White.

2.4.2. Usability

- The user interface must be mobile-friendly, highly responsive, and optimized for both Android and iOS platforms.
- Workflows (like the Provider Flow) must be intuitive and require minimal taps. Critical actions should have clear, unmistakable UI components (for example, ”Slide to accept” for job offers).
- Error messages must be specific, actionable, and user-friendly. The system should highlight exactly which field caused the error and provide clear instructions on how to resolve it.
- The real-time chat interface must visually distinguish between pending, sent, delivered, and read messages (for example, using single and double checkmark icons) to provide clear feedback to users.

2.4.3. Reliability

- The system must operate continuously 24 hours a day, 7 days a week, with a targeted system uptime of 99.9%.
- The WebSocket service must implement automatic background reconnection logic if the user's network connection drops.
- The chat feature must support ”Optimistic UI” and offline queueing; if a message fails to send due to network issues, the user must be prompted with a ”Retry” option.

2.4.4. Performance

- The system must support up to 1,000 concurrent WebSocket connections (active Customers and Providers) without significant degradation in real-time tracking accuracy or chat performance.
- Real-time chat messages must be delivered with a latency of less than 500 milliseconds under stable network conditions.
- API endpoints fetching pending service requests or provider dashboards must execute and return data with a latency of no more than 2 seconds.
- The system must be capable of processing 90% of standard API transactions within 3 seconds.

2.4.5. Security

- All client-server communications (HTTP and WebSockets) must be encrypted using HTTPS/WSS (TLS).
- Authentication for all API requests and WebSocket connections must be secured using JWT (JSON Web Tokens).
- Sensitive user data, including passwords, must be heavily hashed (for example, using bcrypt) before being stored in the database.

2.4.6. Maintainability & Portability

- The mobile application must maintain a clean architectural pattern (for example, separation of UI, Services, and Models) to facilitate easy maintenance and future scaling.
- The backend infrastructure should be containerized (for example, using Docker) to ensure portability and ease of deployment across different cloud hosting environments.

2.4.7. Design Constraints

- The mobile application must be built using the Flutter framework to ensure a unified codebase for both iOS and Android.
- The backend system must be developed using Node.js (NestJS) and utilize a PostgreSQL relational database (RDBMS) managed via Prisma ORM.
- The system must integrate seamlessly with third-party Mapping APIs (for example, Google Maps or Goong Maps) to support location-based services and live tracking.
- The system must provide a Web-based interface (Admin Dashboard) for management purposes.

Troy University Resources

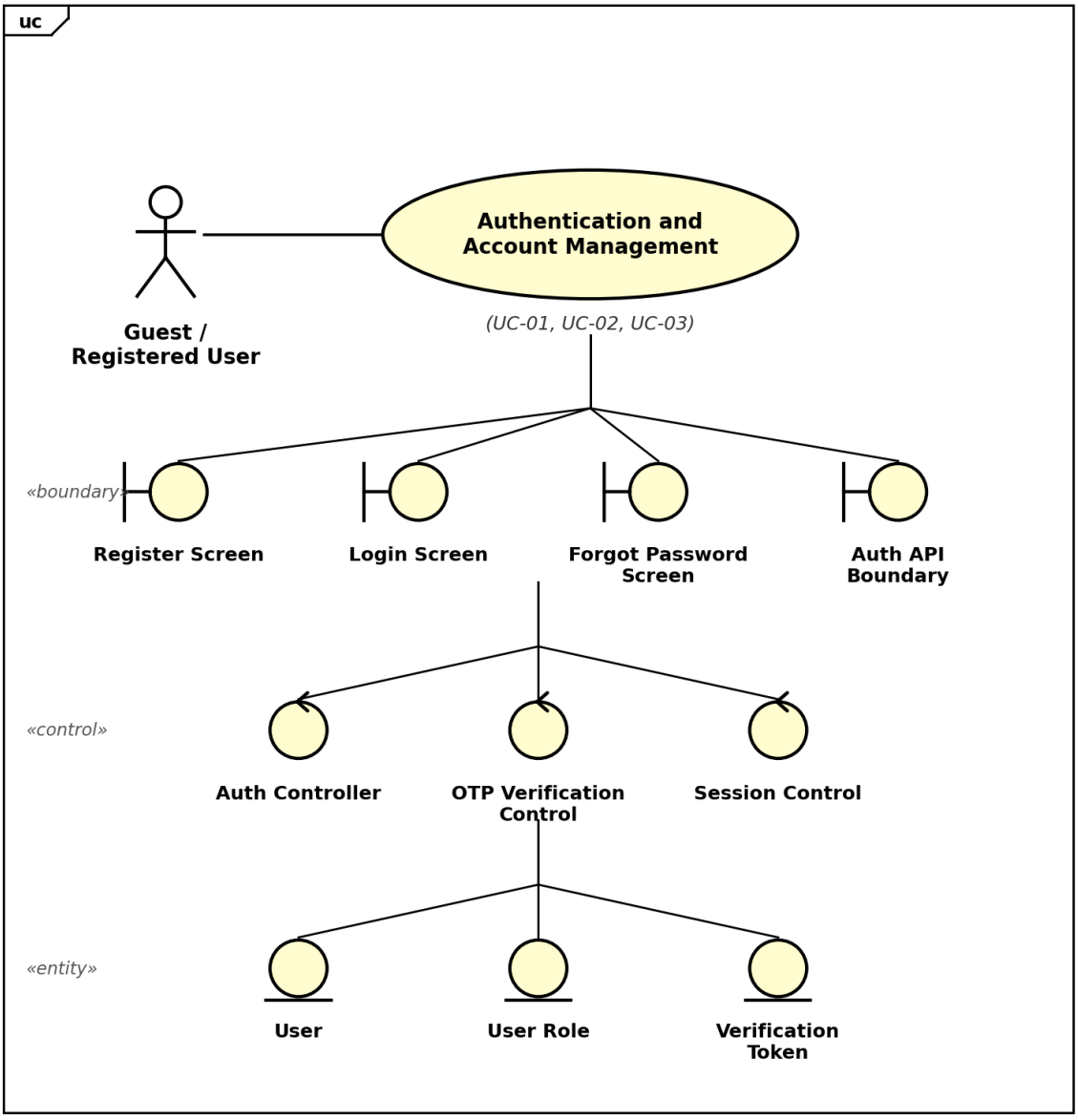
CHAPTER 3. REQUIREMENTS ANALYSIS

3.1. Analysis Class Identification

Each Use Case is examined to determine the Boundary classes, Control classes, and Entity classes involved in the corresponding business flow. Boundary classes represent the user interface or API entry points, Control classes handle business logic and workflow coordination, while Entity classes represent persistent business data used by the system.

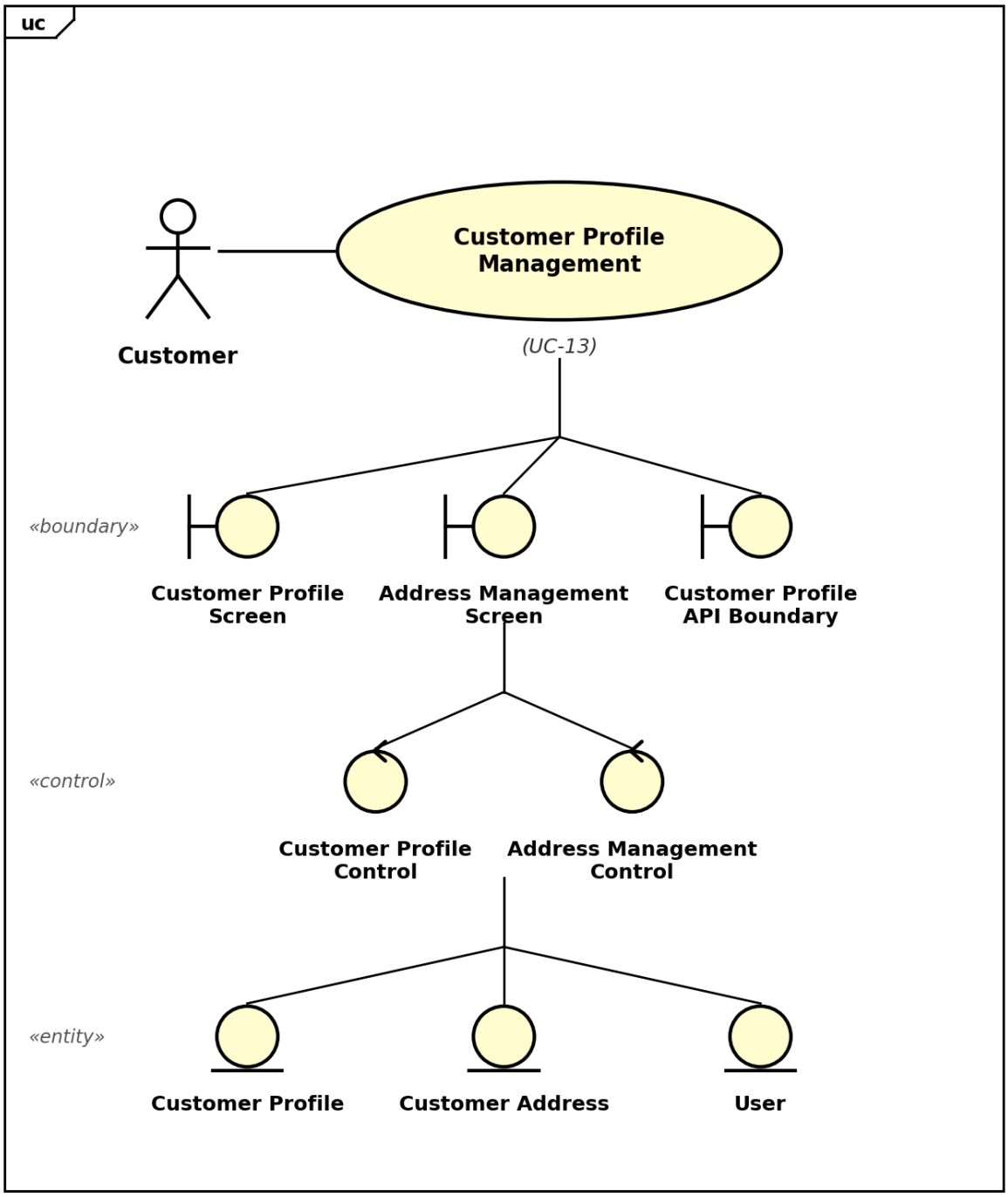
1. **Result of the initial decomposition of use case group ”Authentication and Account Management”**

This group covers UC-01, UC-02, UC-03.



2. **Result of the initial decomposition of use case group ”Customer Profile Management”**

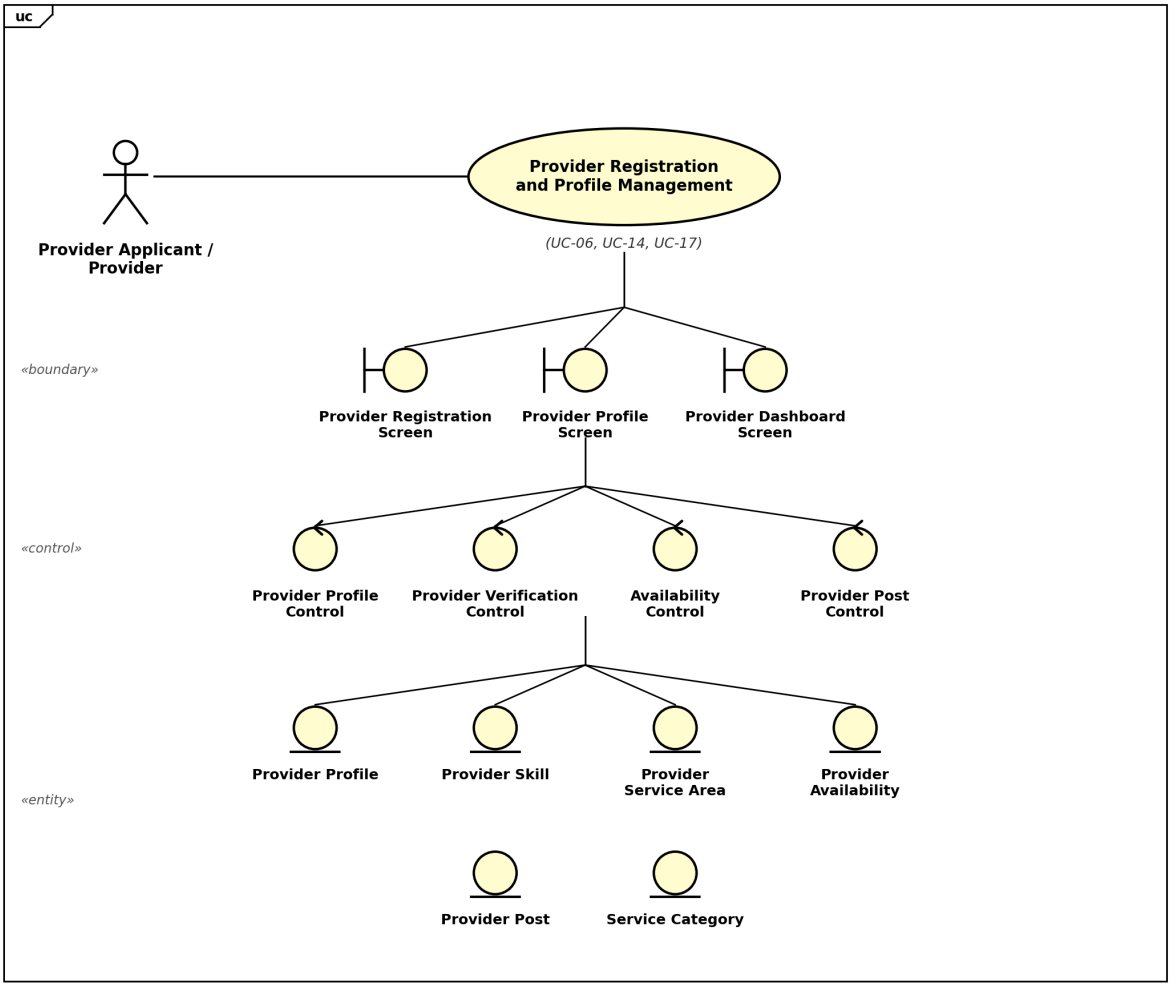
This group covers UC-13.



3. **Result of the initial decomposition of use case group ”Provider Registration and Profile Management”**

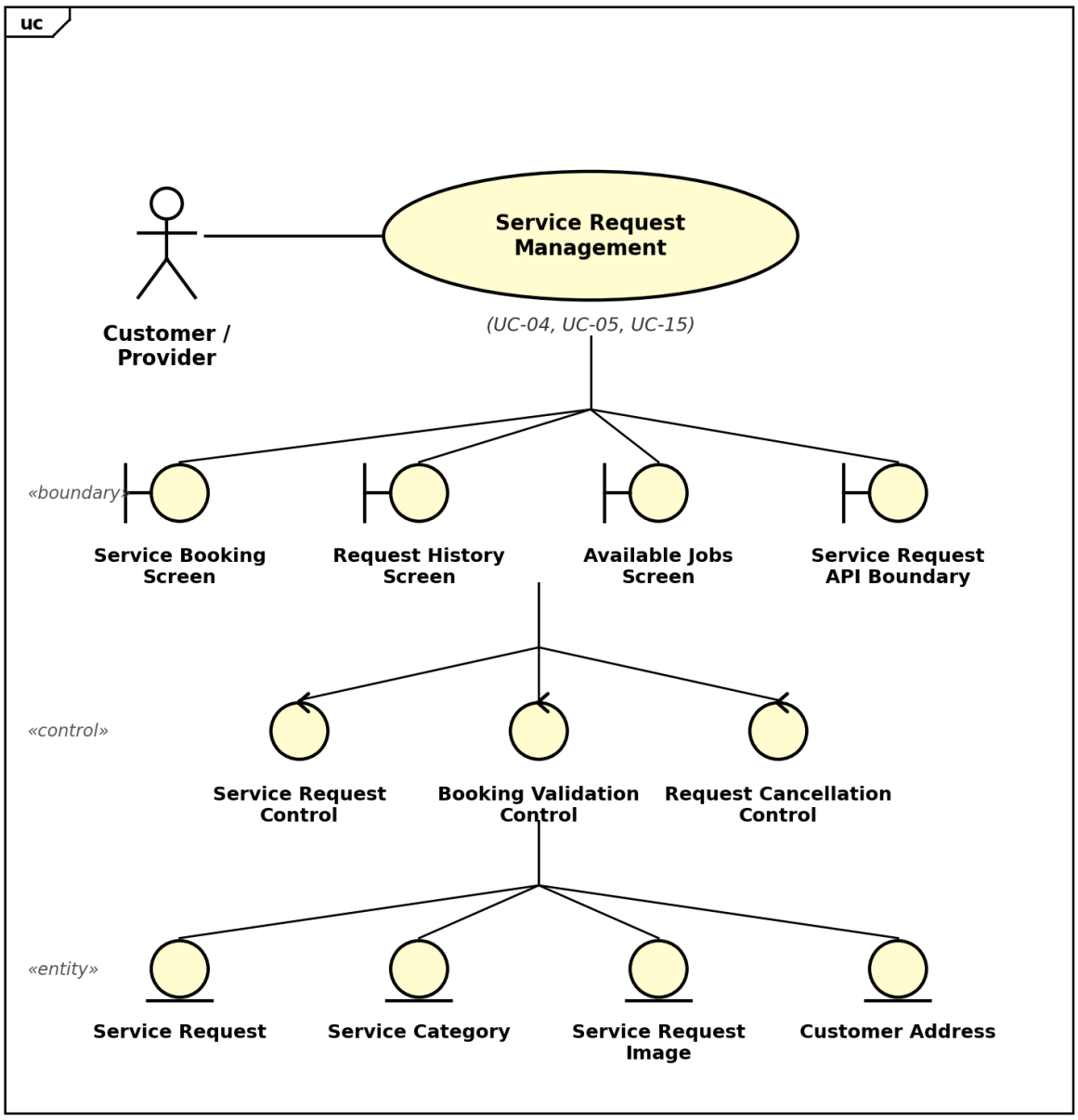
This group covers UC-06, UC-14, UC-17.

Troy University Resources



4. **Result of the initial decomposition of use case group ”Service Request Management”**

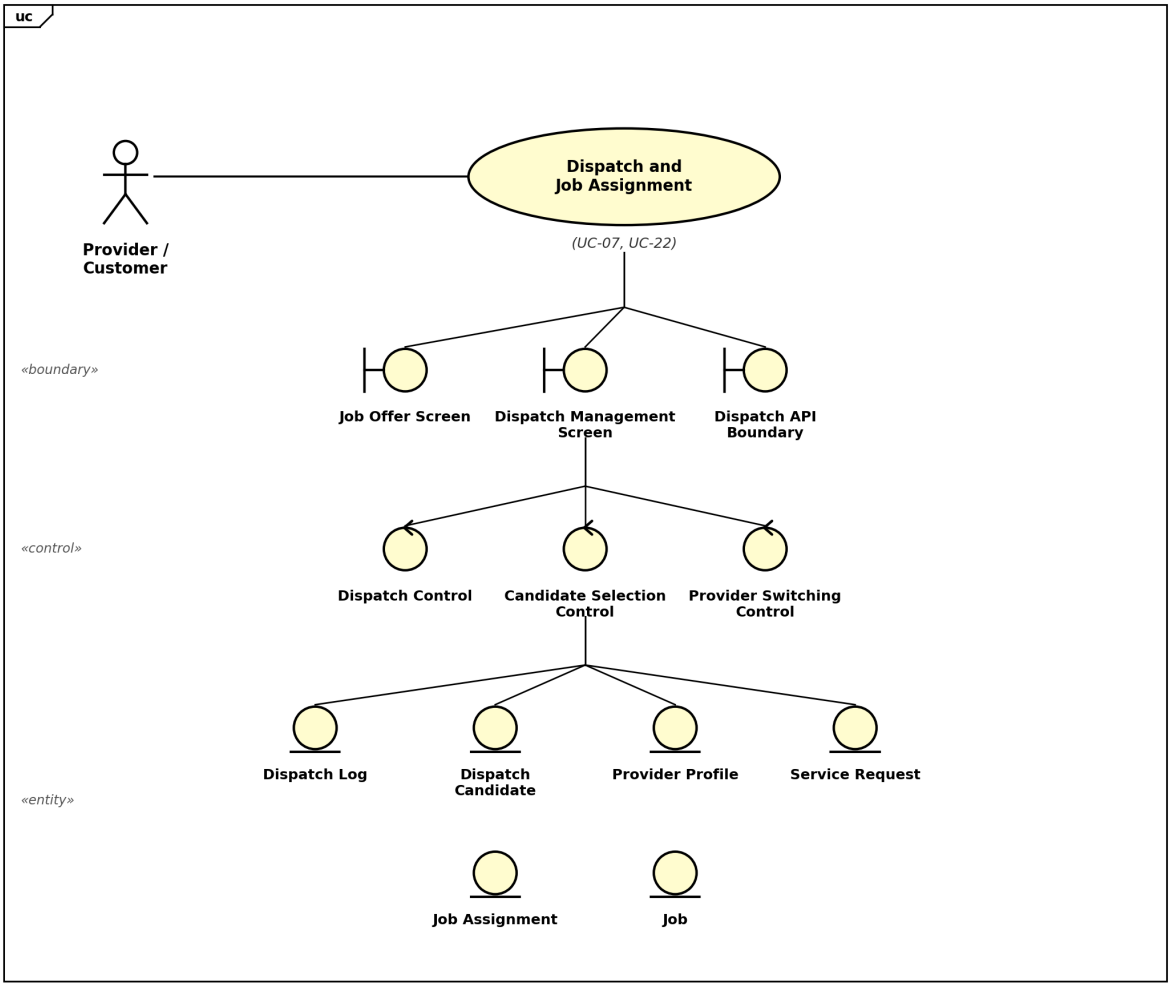
This group covers UC-04, UC-05, UC-15.



5. **Result of the initial decomposition of use case group ”Dispatch and Job Assignment”**

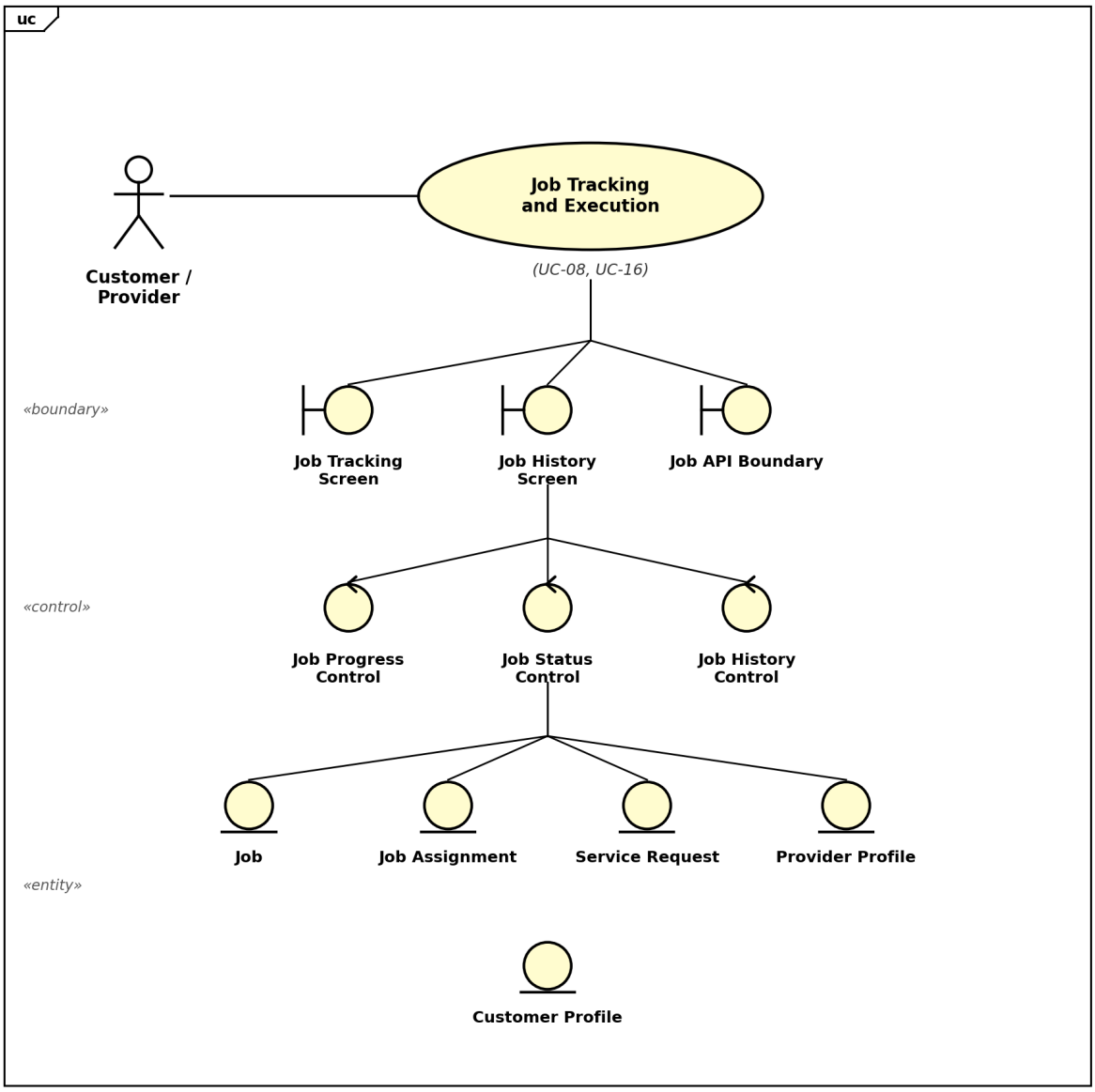
This group covers UC-07, UC-22.

Troy University Resources



6. **Result of the initial decomposition of use case group ”Job Tracking and Execution”**

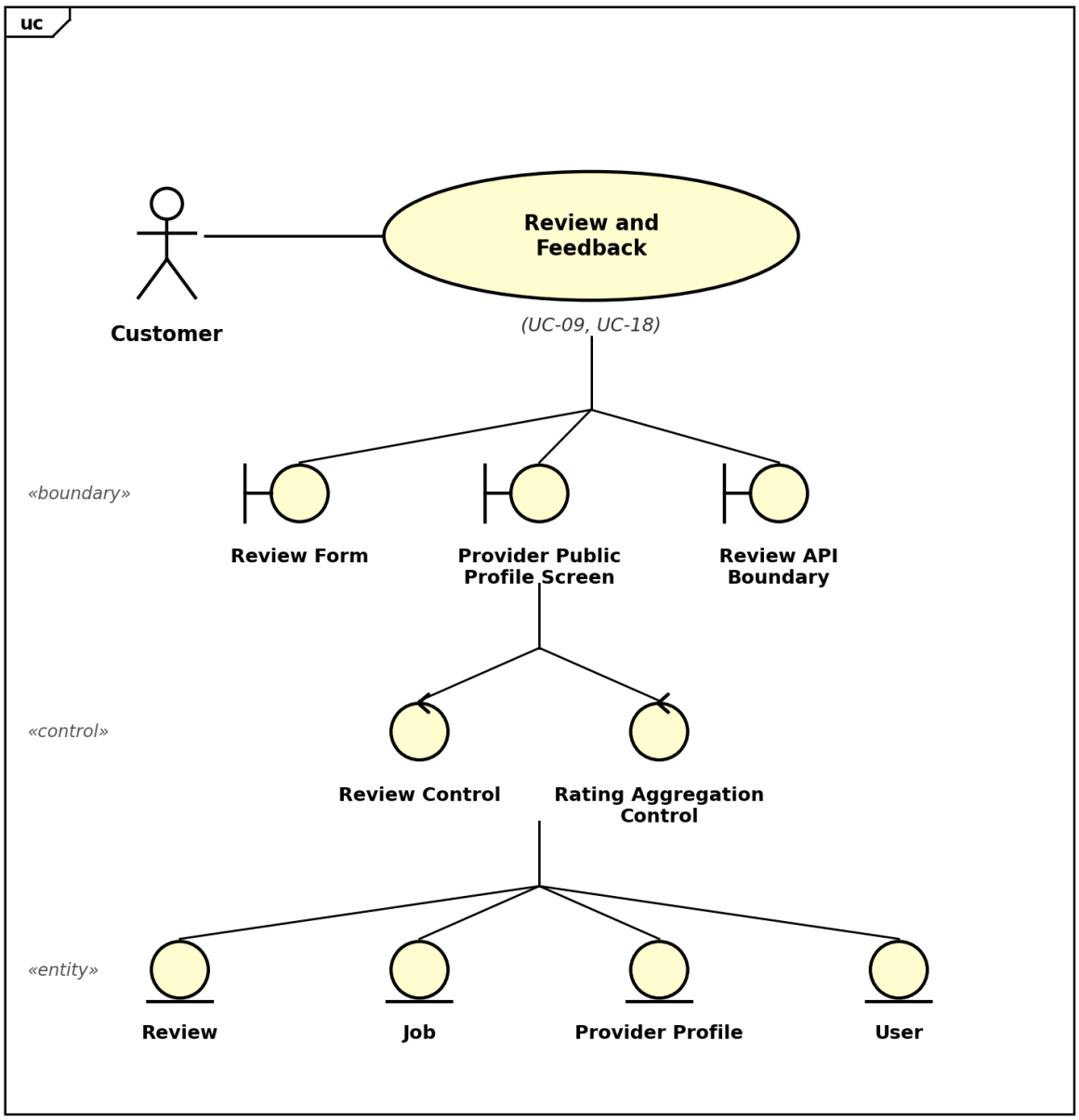
This group covers UC-08, UC-16.



7. **Result of the initial decomposition of use case group ”Review and Feedback”**

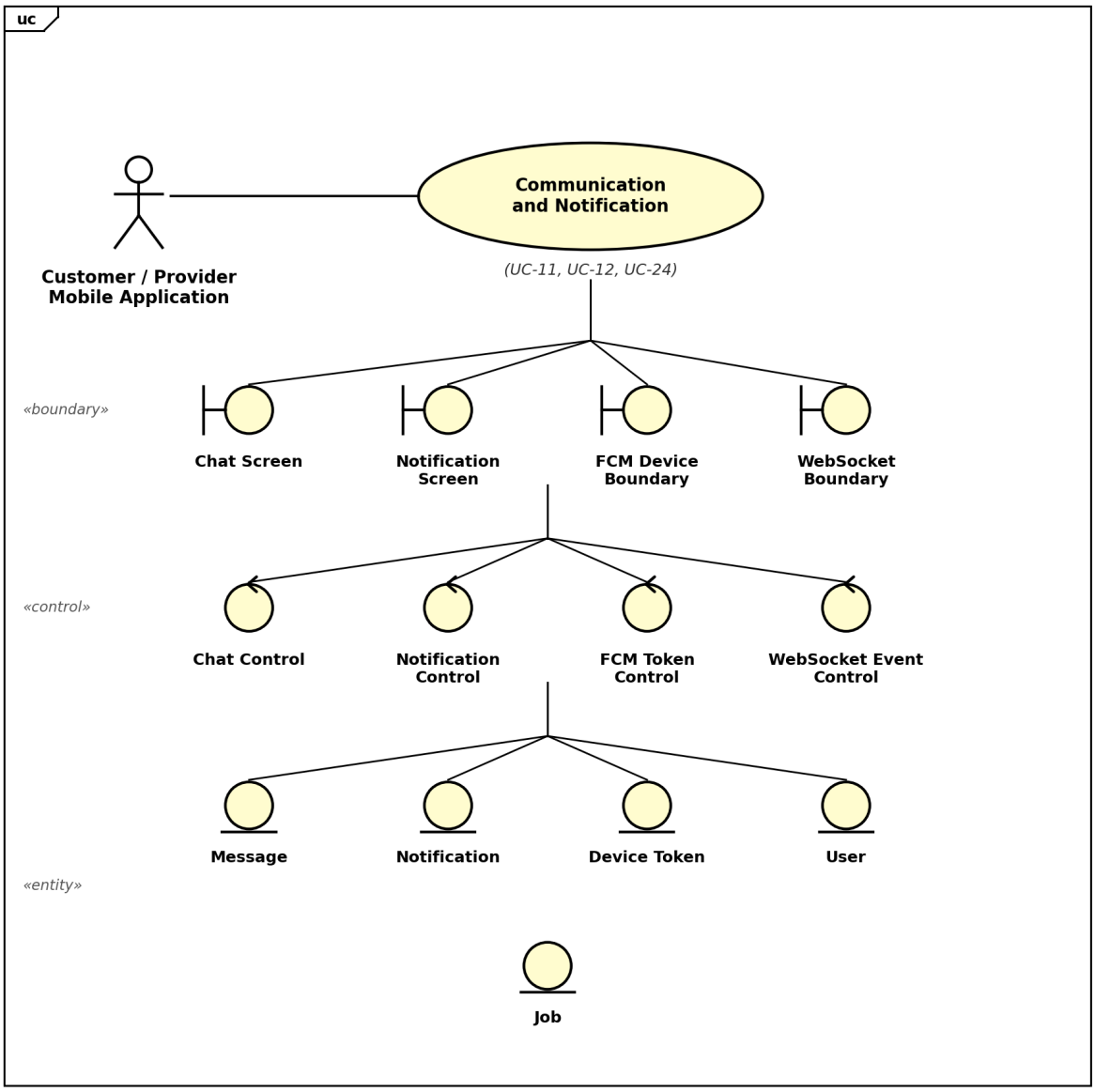
This group covers UC-09, UC-18.

Troy University Resources



8. **Result of the initial decomposition of use case group ”Communication and Notification”**

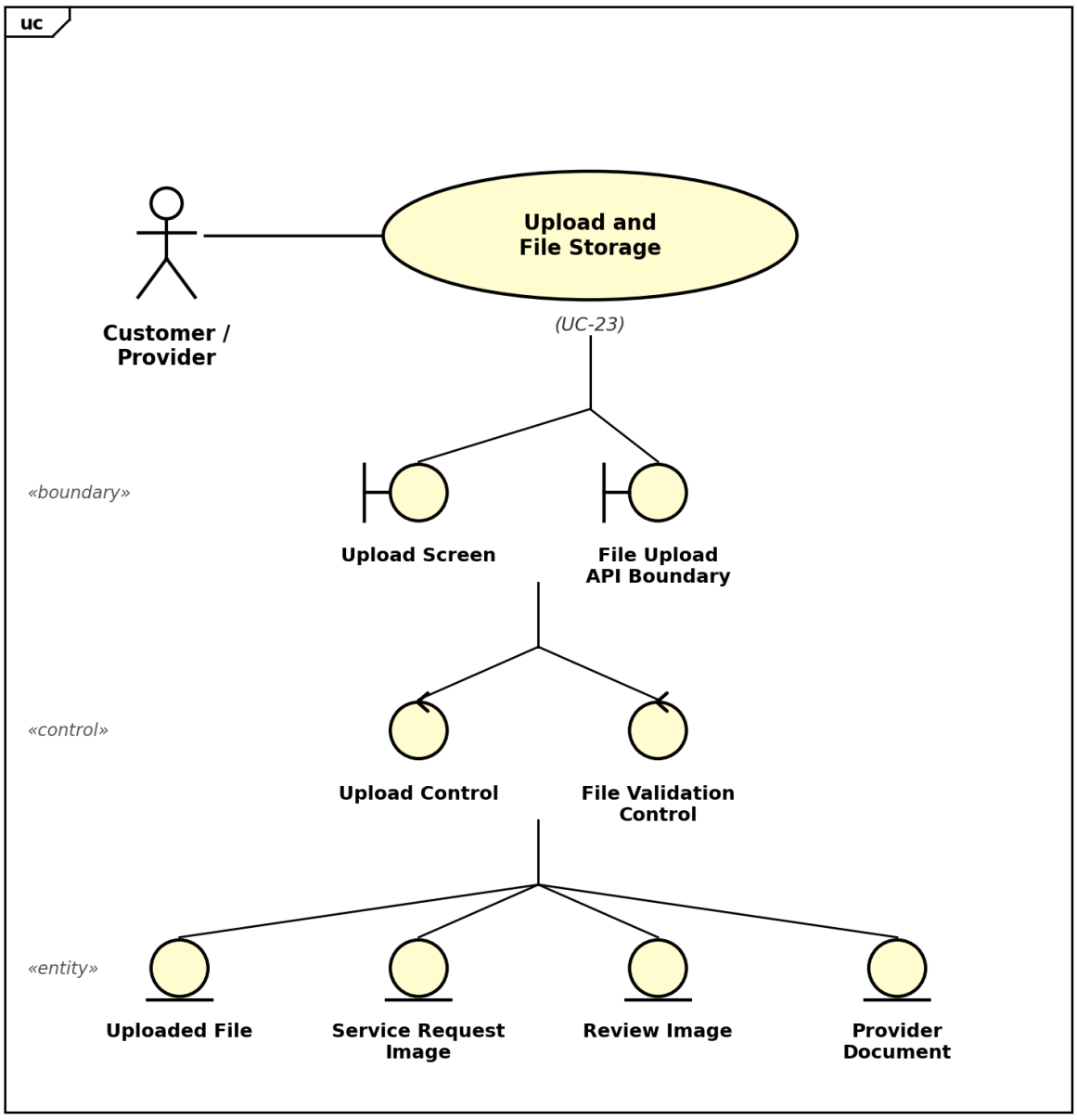
This group covers UC-11, UC-12, UC-24.



9. **Result of the initial decomposition of use case group ”Upload and File Storage”**

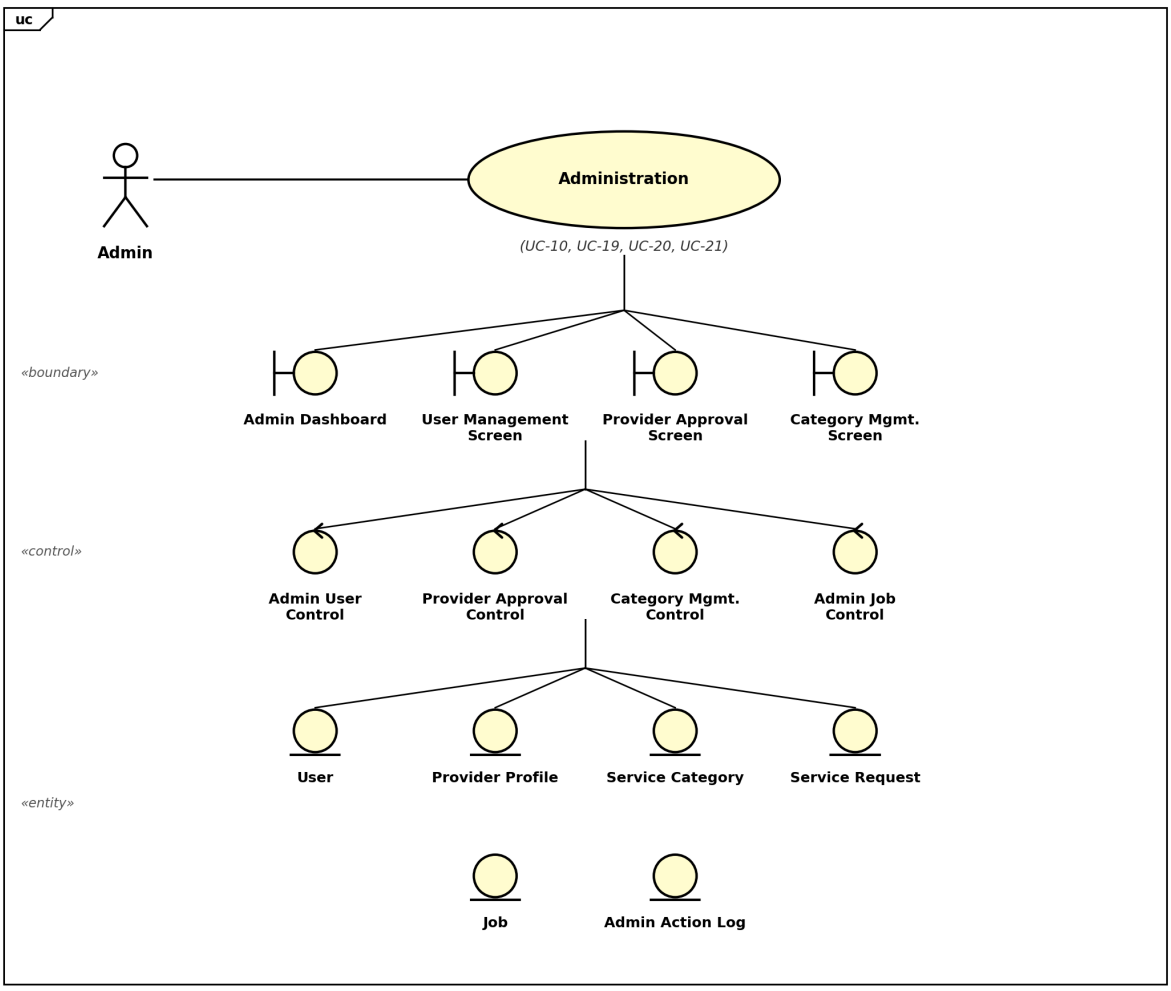
This group covers UC-23.

Troy University Resources



10. **Result of the initial decomposition of use case group ”Administration”**

This group covers UC-10, UC-19, UC-20, UC-21.



After identifying the initial analysis classes, our team refined the model by merging classes with overlapping responsibilities. For example, separate login, registration, and password recovery interfaces were grouped under the authentication boundary layer, while duplicated provider location and availability concepts were consolidated into ProviderAvailability and ProviderServiceArea so that we can reduce the model complexity.

3.2. Sequence Diagrams

- Sequence Diagram for use case “UC-01: Register Customer Account”
- Sequence Diagram for use case “UC-02: Log in to the System”
- Sequence Diagram for use case “UC-03: Recover Forgotten Password”
- Sequence Diagram for use case “UC-04: Book a Home Service”
- Sequence Diagram for use case “UC-05: Manage Service Request History”
- Sequence Diagram for use case “UC-06: Register as a Service Provider”
- Sequence Diagram for use case “UC-07: Accept or Decline a Dispatched Job”
- Sequence Diagram for use case “UC-08: Track and Update Job Progress”
- Sequence Diagram for use case “UC-09: Review a Completed Service”
- Sequence Diagram for use case “UC-10: Approve or Reject Provider Application”

Troy University Resources

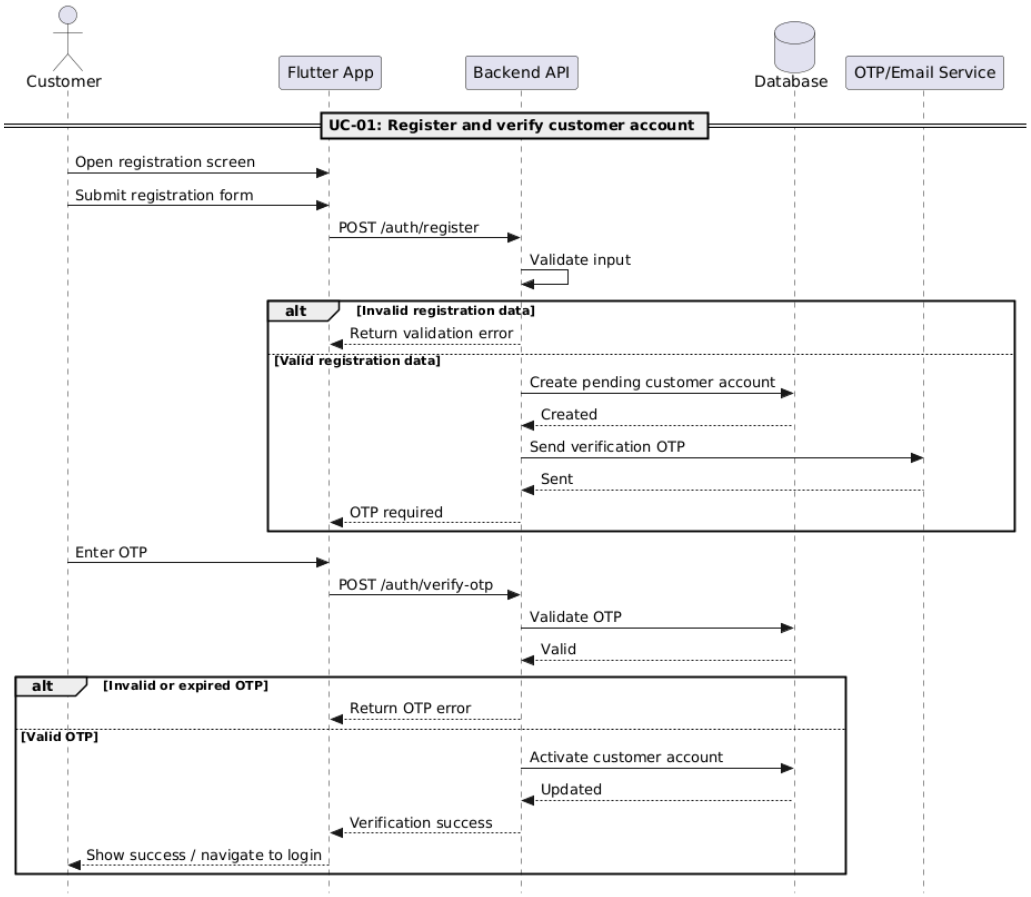


Figure 3-1: Sequence Diagram - UC-01 Register Customer Account

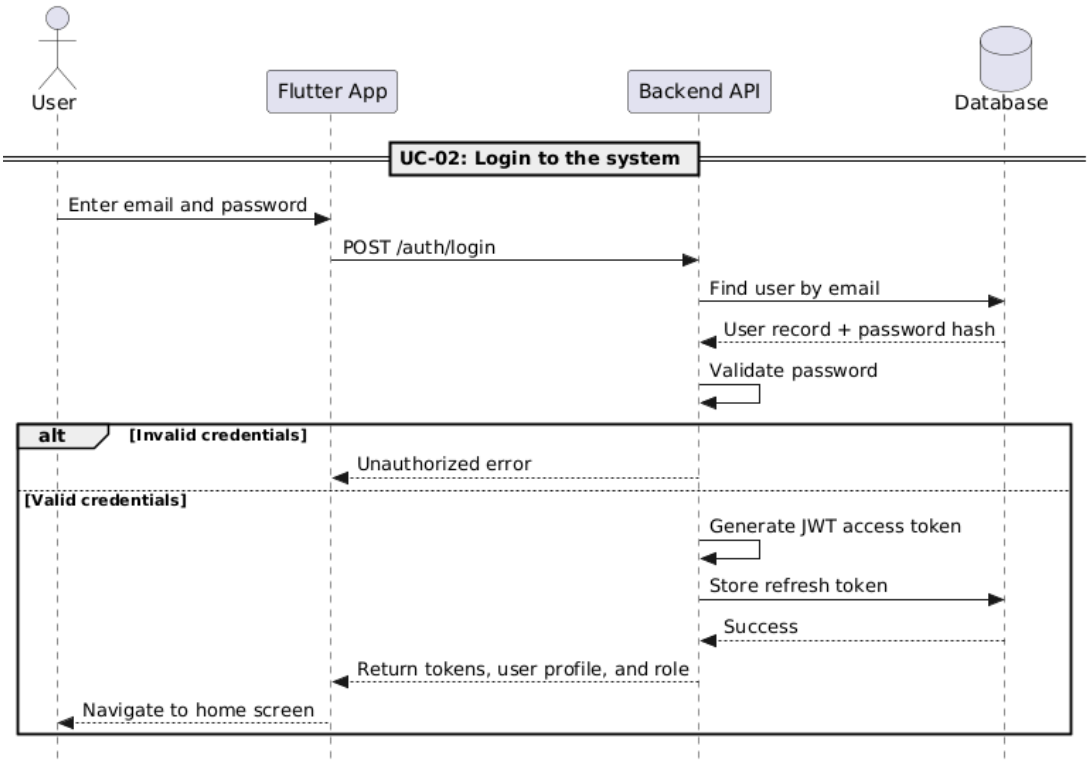


Figure 3-2: Sequence Diagram - UC-02 Log in to the System

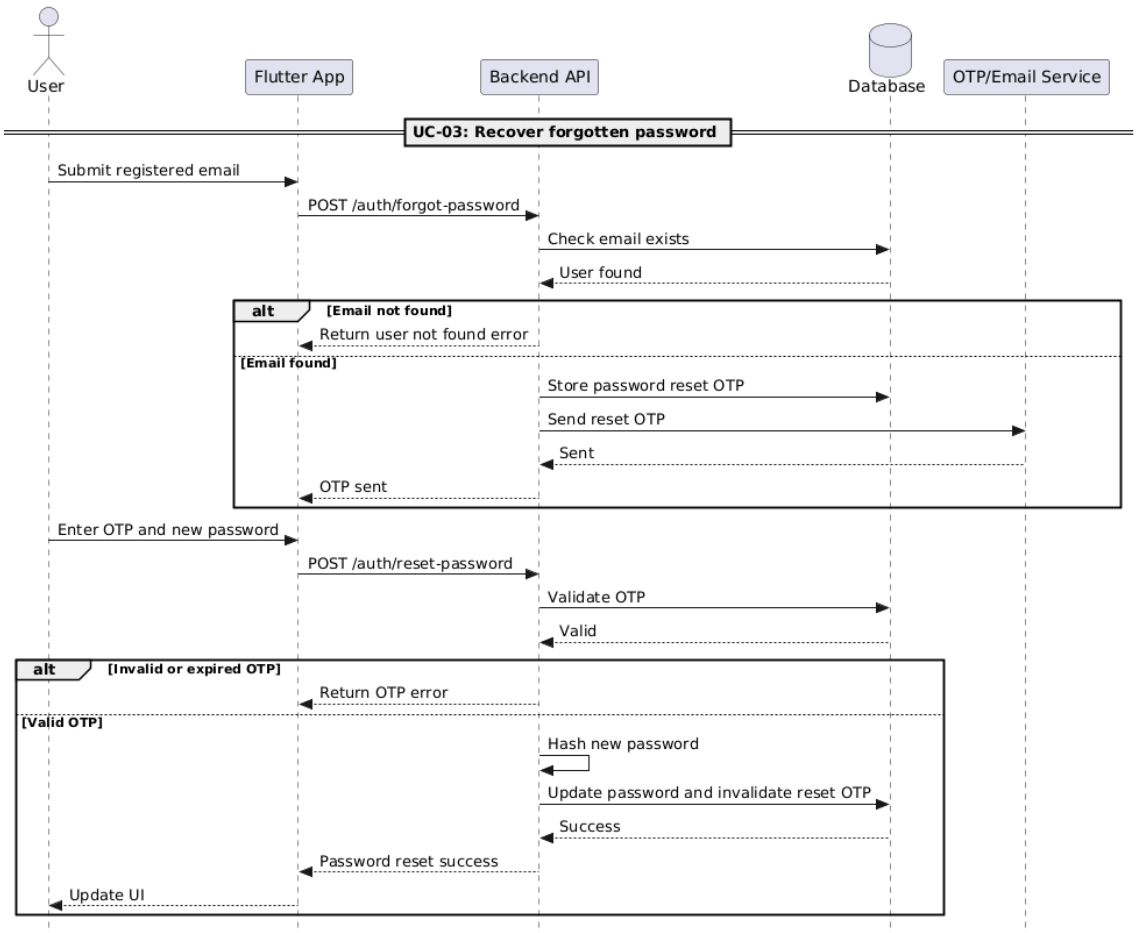


Figure 3-3: Sequence Diagram - UC-03 Recover Forgotten Password

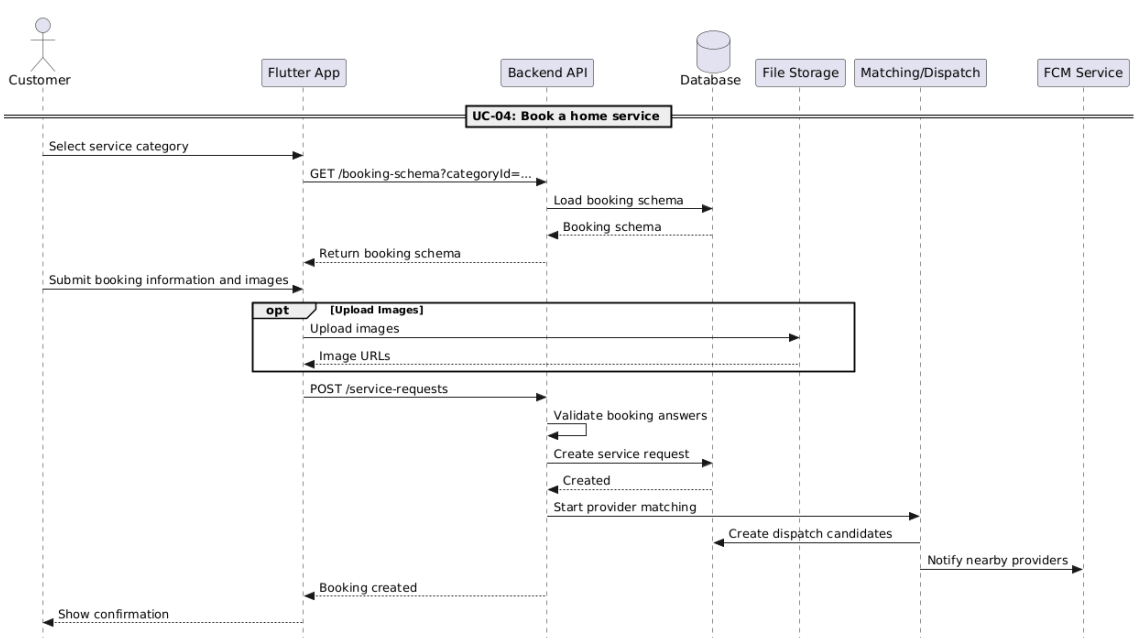


Figure 3-4: Sequence Diagram - UC-04 Book a Home Service

Troy University Resources

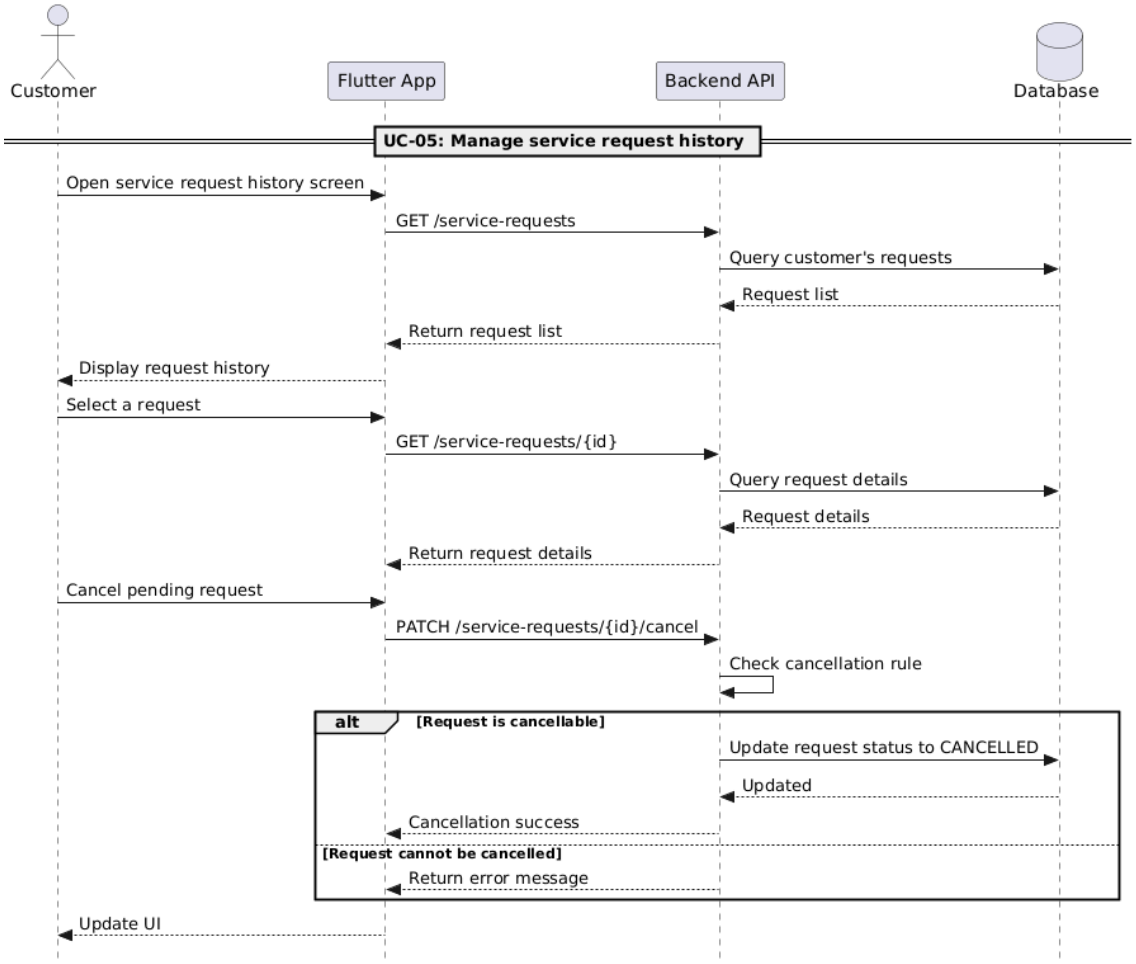


Figure 3-5: Sequence Diagram - UC-05 Manage Service Request History

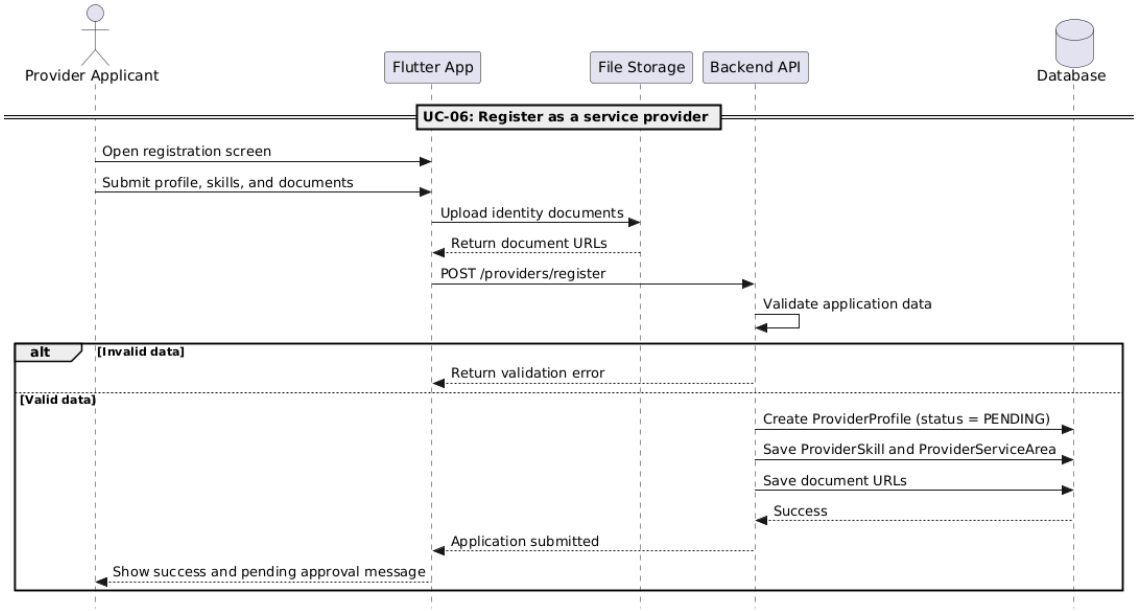


Figure 3-6: Sequence Diagram - UC-06 Register as a Service Provider

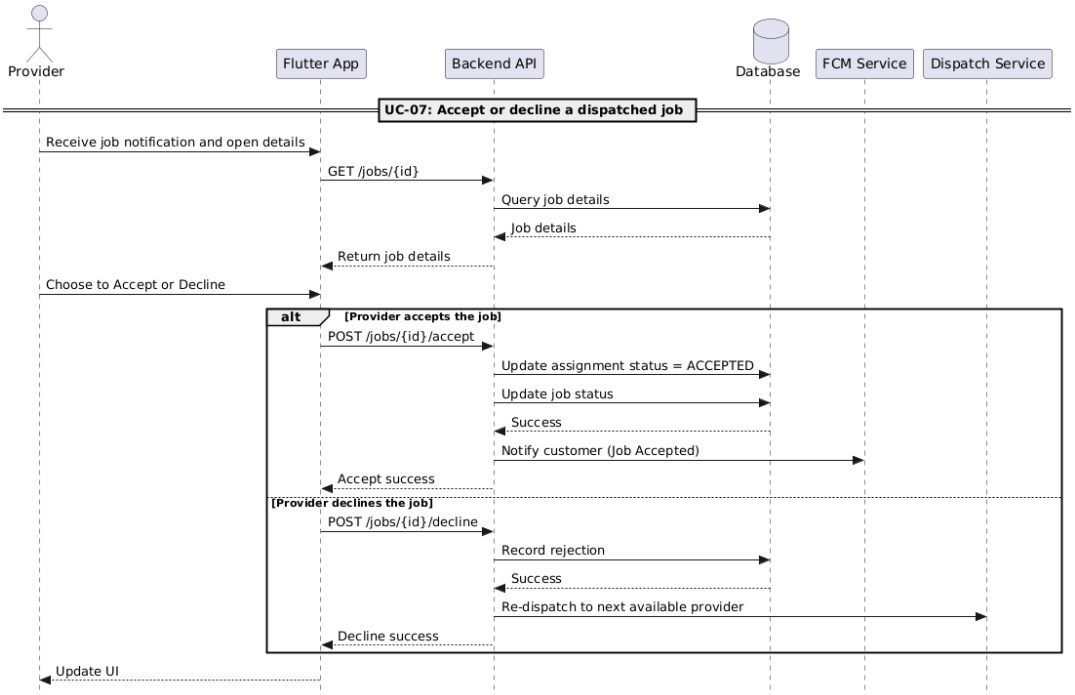


Figure 3-7: Sequence Diagram - UC-07 Accept or Decline a Dispatched Job

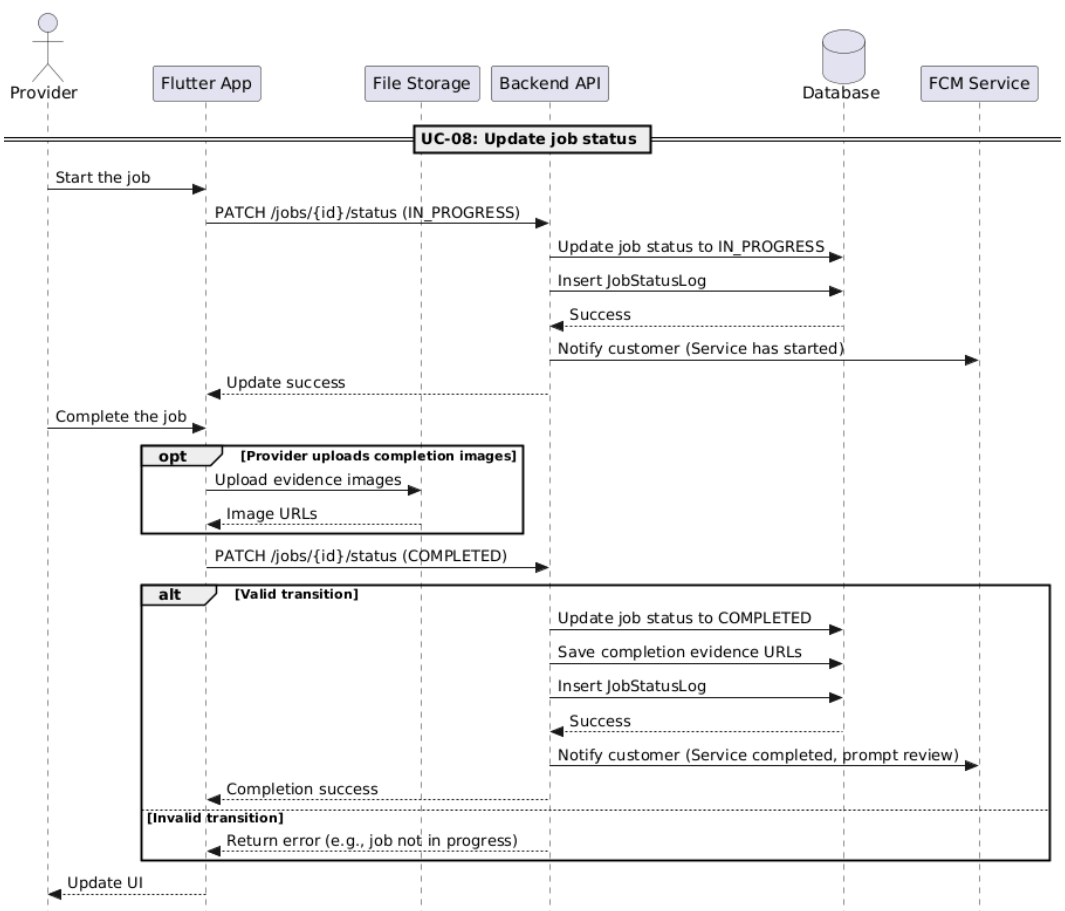


Figure 3-8: Sequence Diagram - UC-08 Track and Update Job Progress

Troy University Resources

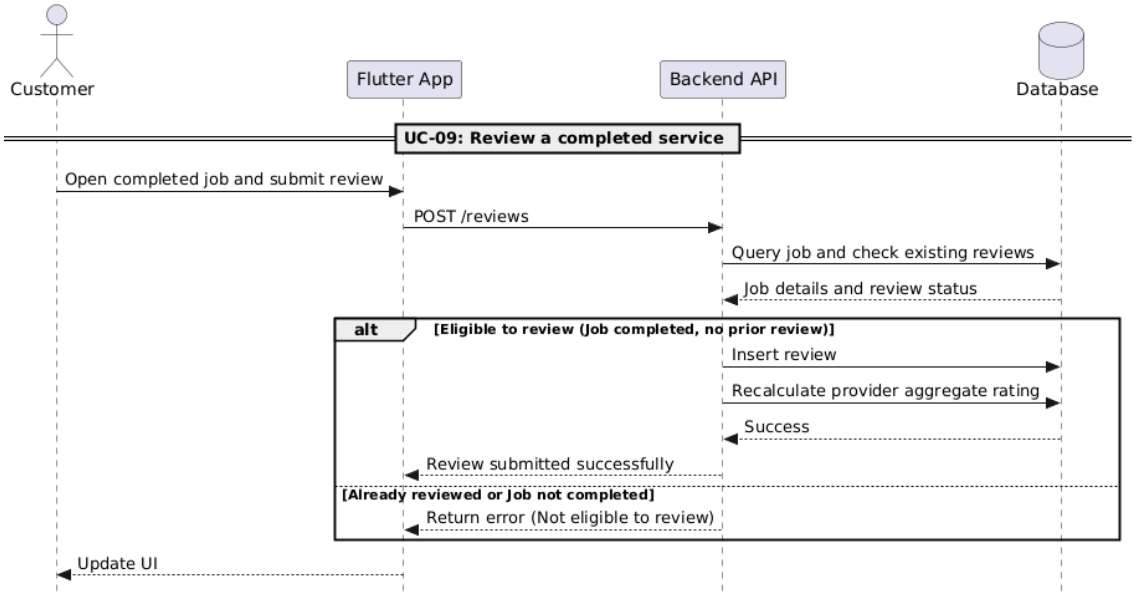


Figure 3-9: Sequence Diagram - UC-09 Review a Completed Service

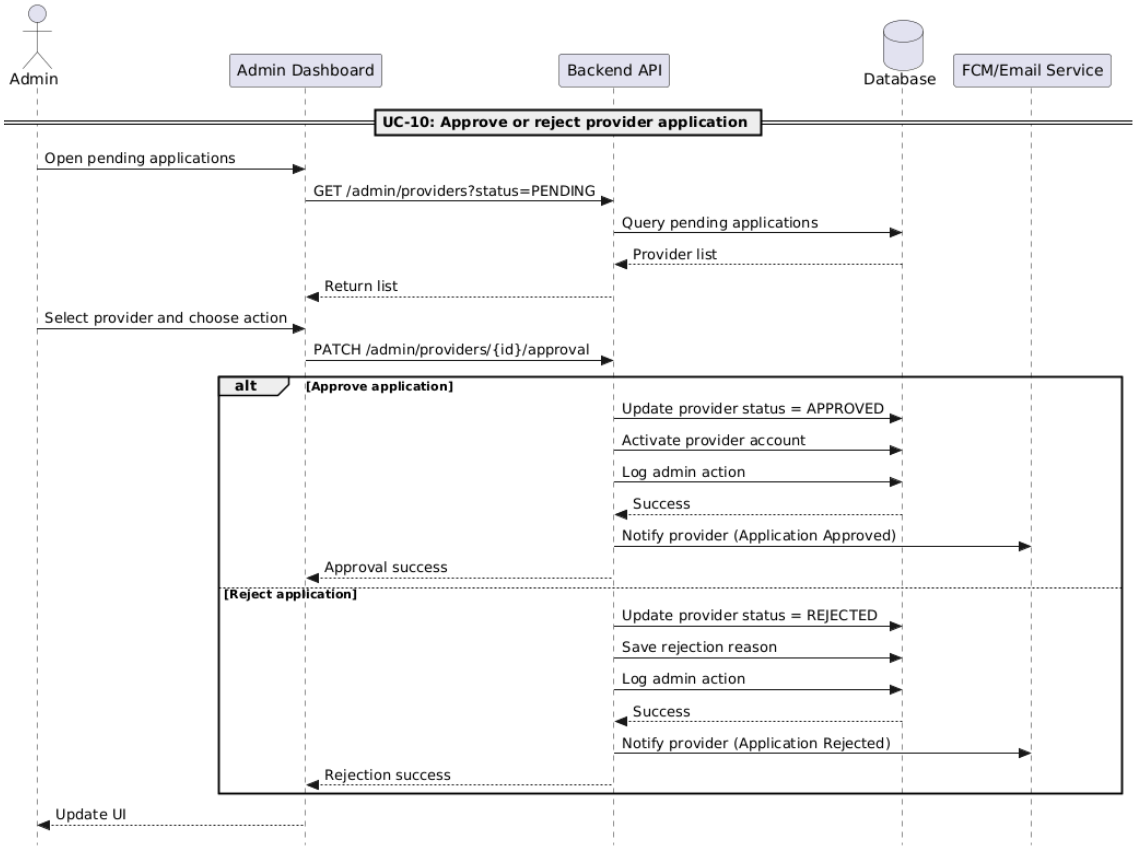


Figure 3-10: Sequence Diagram - UC-10 Approve or Reject Provider Application

- Sequence Diagram for use case “UC-11: Send and Receive Chat Messages”

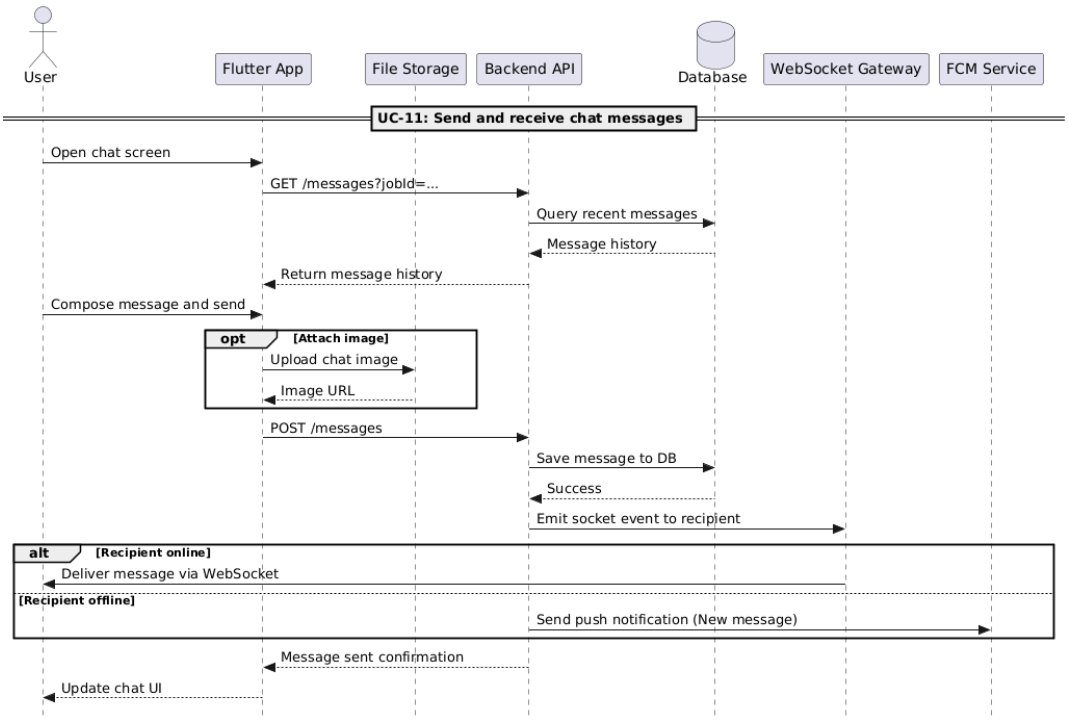


Figure 3-11: Sequence Diagram - UC-11 Send and Receive Chat Messages

- Sequence Diagram for use case “UC-12: Manage Notifications and Preferences”

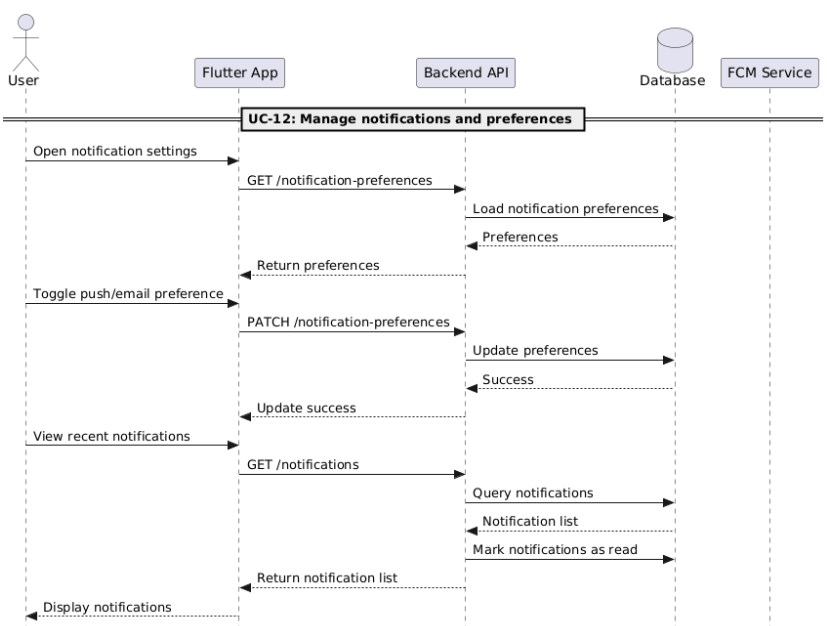


Figure 3-12: Sequence Diagram - UC-12 Manage Notifications and Preferences

3.3. Entity-Relationship Diagram (ERD)

This section presents the Entity-Relationship Diagram of the Taskera system. The ERD is constructed by identifying the main data objects, their key attributes, and the relationships among these objects. To keep the diagrams readable, the ERD is divided into four modules: account and authentication, service request management, provider dispatch and job management, and communication feedback and administration.

Troy University Resources

3.3.1. Main data objects

The main data objects of Taskera include User, User Role, Verification Token, Device Token, Notification Preference, Customer Profile, Customer Address, Service Category, Service Request, Service Request Image, Provider Profile, Provider Skill, Provider Service Area, Provider Availability, Dispatch Log, Dispatch Candidate, Job, Job Assignment, Message, Notification, Review, Provider Post, and Admin Action Log. These objects cover the core data needed for account management, booking, dispatching, job execution, communication, feedback, and administration.

3.3.2. ERD diagrams

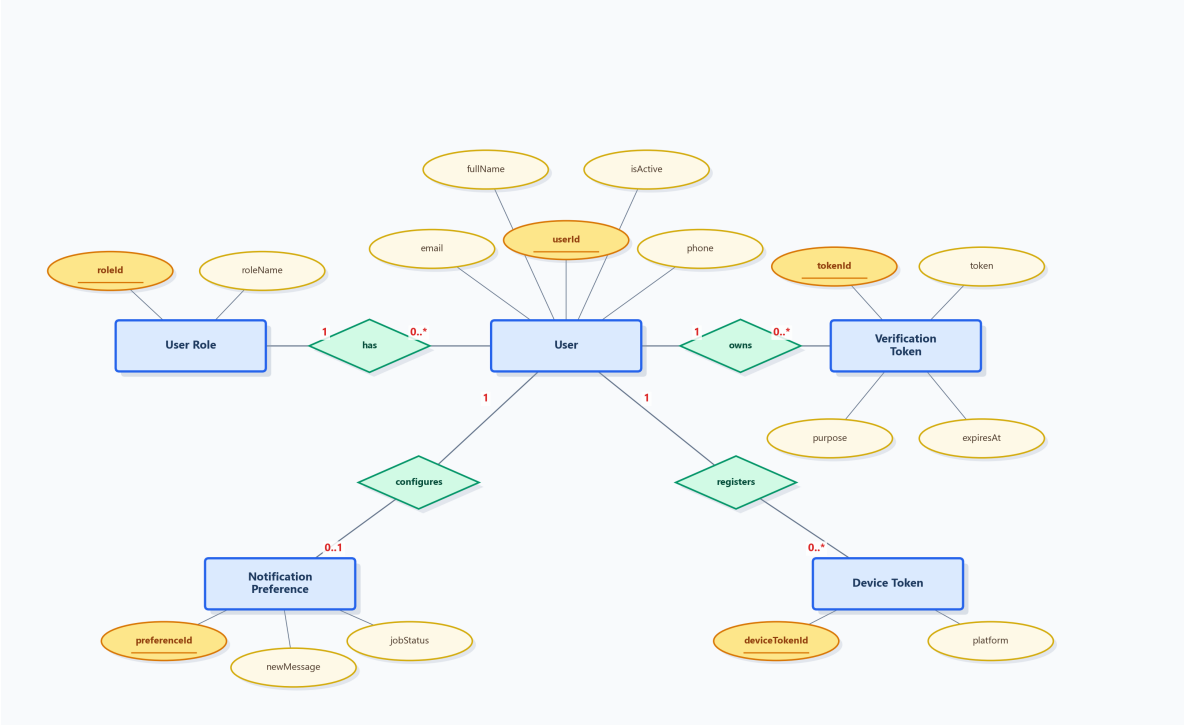


Figure 3-13: ERD - Account and Authentication Module

3.4. Description of relationships between data objects

3.4.1. Account and Authentication Module

- Each User Role can be assigned to many Users, while each User has one main role. This relationship supports role-based access control for customers, providers, and administrators.
- Each User may own many Verification Tokens. These tokens are used for account verification, OTP confirmation, and password recovery. A token belongs to exactly one user and becomes invalid after it expires or is consumed.
- Each User may register multiple Device Tokens because the same account can be used on different devices. These tokens allow the system to deliver push notifications through FCM.
- Each User may configure one Notification Preference record. This record stores whether the user wants to receive job updates, chat alerts, review messages, and system alerts.

3.4.2. Service Request Module

- Each Customer Profile can create many Service Requests, while each Service Request is created by exactly one customer profile. This relationship represents the main booking action in Taskera.
- Each Customer Profile can save many Customer Addresses. A saved address can later be selected when creating a new service request.

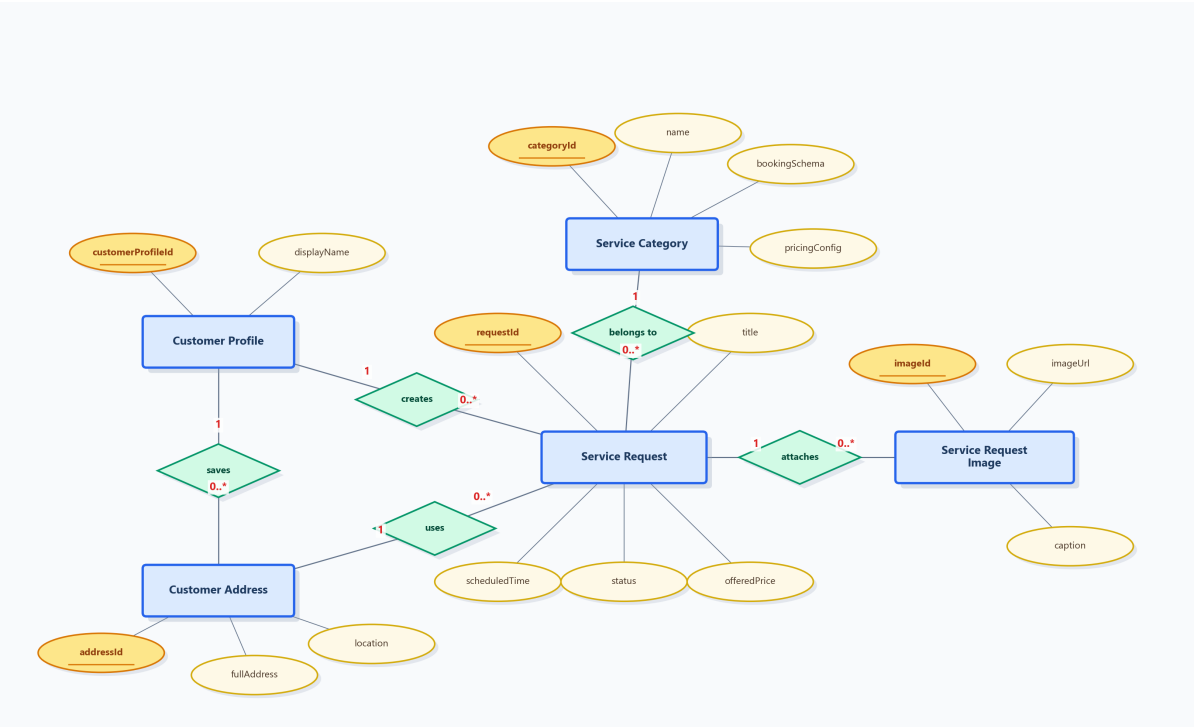


Figure 3-14: Service Request Management Entity Relationship Diagram

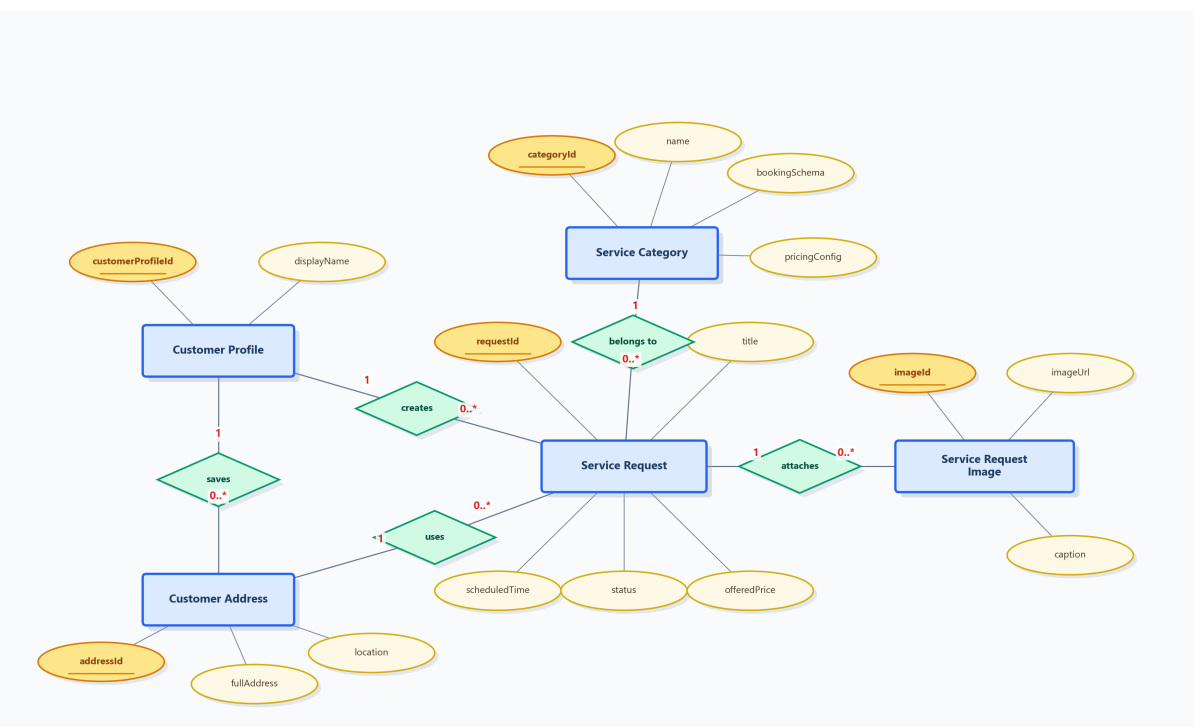


Figure 3-15: ERD - Service Request Module

Troy University Resources

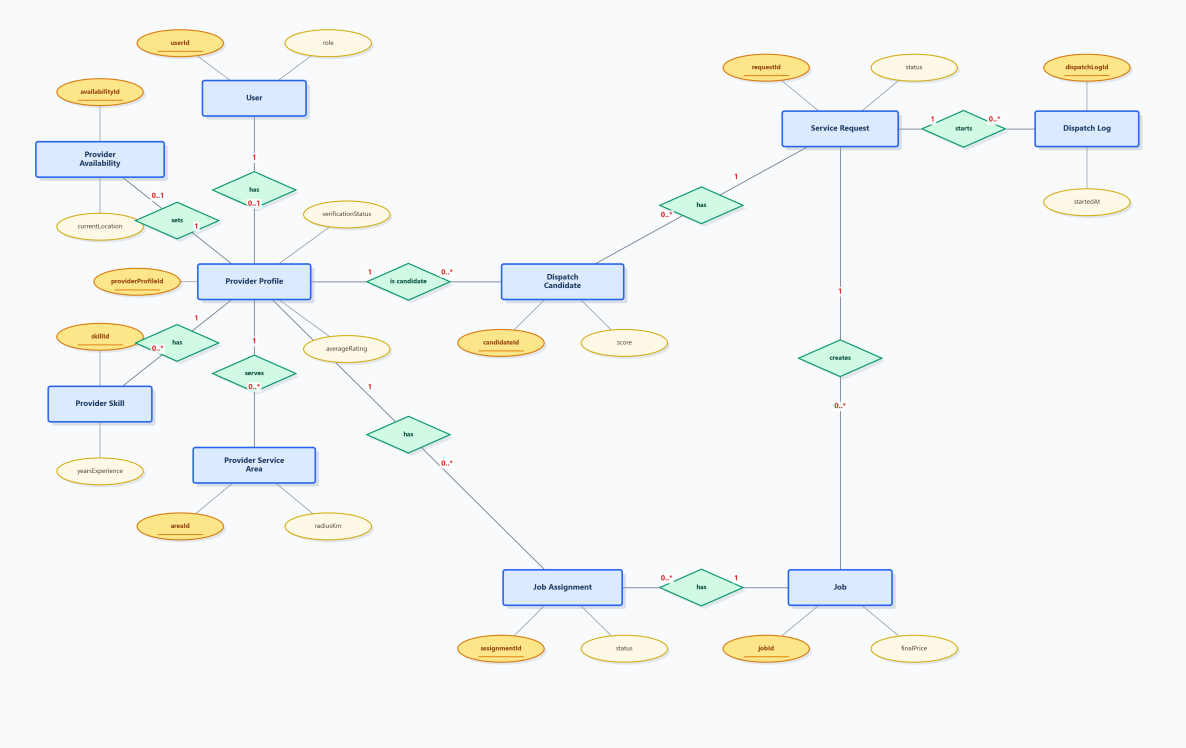


Figure 3-16: ERD - Provider Dispatch and Job Module

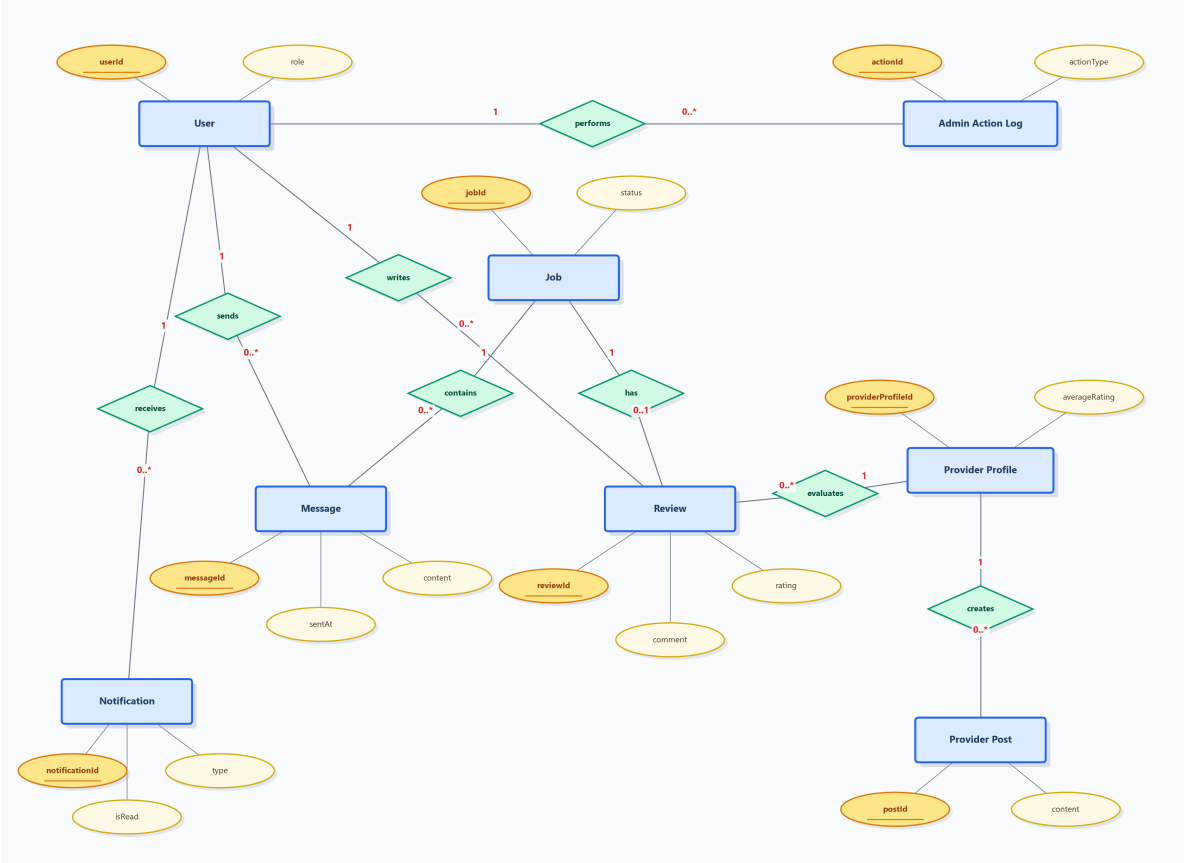


Figure 3-17: ERD - Communication Feedback and Administration Module

- Each Service Request uses one Customer Address as the service location. One saved address may be reused by many service requests over time.
- Each Service Request belongs to one Service Category, such as home cleaning, electrical repair, plumbing repair, or painting. A service category can be linked to many service requests.
- Each Service Request can have zero or more Service Request Images. These images provide visual evidence or additional information about the requested work.

3.4.3. Provider, Dispatch, and Job Module

- Each User may have one Provider Profile after registering as a service provider. The Provider Profile stores professional information, verification status, rating, skills, and availability data.
- Each Provider Profile can have many Provider Skills and many Provider Service Areas. These relationships help the dispatching process determine whether a provider is suitable for a customer request.
- Each Provider Profile may have one current Provider Availability record. This record indicates whether the provider is online, available, and located within a suitable service area.
- Each Service Request can start one or more Dispatch Logs. A dispatch log records one attempt to find a suitable provider for the request.
- Each Dispatch Log evaluates many Dispatch Candidates. Each candidate refers to one provider profile and stores matching information such as distance, score, status, and response time.
- Each Service Request can create one or more Jobs depending on assignment and provider-switching rules. Each Job can have one or more Job Assignments, while each assignment links the job to a provider profile.

3.4.4. Communication, Feedback, and Administration Module

- Each Job can contain many Messages. Each Message is sent by one User and belongs to one job conversation, allowing customers and providers to communicate during service execution.
- Each User can receive many Notifications. Notifications inform users about job offers, booking updates, chat messages, job status changes, and system events.
- Each completed Job may have one Review from the customer. Each Review evaluates one Provider Profile and contributes to the provider rating.
- Each Provider Profile can create many Provider Posts. These posts present provider experience, work images, or portfolio content to customers.
- Admin Action Log records administrative operations performed by an Admin user. Each log may refer to a target object such as a User, Provider Profile, Service Request, Job, or Service Category, depending on the action type.

Troy University Resources

CHAPTER 4. SYSTEM DESIGN

By following common software engineering principles, the design clearly separates the presentation layer, business logic layer, and data access layer, making the source code easier to manage and the system modules more organized. This modular structure also allows different parts of the system to be developed, tested, and improved independently, which helps reduce integration issues and supports the implementation of more complex business features. As a result, the proposed design not only meets the current requirements of Taskera but also provides a flexible foundation for future enhancements and technology integration.

4.1. Architectural Design

4.1.1. Selection of Software Architecture

The Taskera platform uses a three-tier client–server architecture. This architecture is selected because Taskera is a mobile service-booking system in which different users, including customers, providers, and administrators, interact with a central backend server through a mobile application. In this model, the system is divided into three main tiers: the presentation tier, the application tier, and the data tier.
The presentation tier is responsible for displaying the user interface and receiving user interactions. The application tier processes business logic, validates data, manages user authentication, handles service booking workflows, dispatches providers, manages communication, and sends notifications. The data tier stores and manages persistent system data. This architectural style is suitable for Taskera because it clearly separates user interface processing, business processing, and data management. As a result, the system becomes easier to develop, test, maintain, and extend.
Taskera does not apply a microservice architecture because the current project scope does not require independently deployed services for each business module. Instead, the system uses a centralized backend server with clearly separated internal modules. RESTful APIs, WebSocket, Firebase Cloud Messaging, file storage, and OTP/email services are used as supporting communication mechanisms and external services, not as separate architectural styles.

4.1.2. Application of the Architecture to Taskera

The presentation tier is implemented by the Flutter mobile application. This tier provides separate interfaces for customers, providers, and administrators through the Customer App, Provider App, and Admin Interface. Customers can register accounts, book home services, track job progress, chat with providers, receive notifications, and review completed services. Providers can manage their profiles, receive job offers, accept or decline dispatched jobs, update job status, and communicate with customers. Administrators can manage users, review provider applications, monitor jobs, and handle platform operations.
The application tier is implemented by the NestJS backend server. This tier acts as the central processing layer of the system. It contains the main business modules of Taskera, including Authentication, Service Request Management, Job and Dispatch Management, Chat and Notification, Review Management, and Admin Management. These modules receive requests from the mobile application, apply business rules, validate input data, coordinate workflows, and access the database through data access components.
The data tier is implemented using a PostgreSQL relational database. This tier stores and manages the main data objects of the system, such as users, customer profiles, provider profiles, service categories, service requests, jobs, messages, notifications, reviews, device tokens, and uploaded file references. PostgreSQL is suitable for this project because it supports relational constraints, structured queries, and data consistency between related entities.
The frontend communicates with the backend mainly through RESTful APIs for standard operations such as login, booking creation, profile updates, job management, and review submission. For real-time functions, the system uses WebSocket to support instant chat messages and live job status updates. In addition, Firebase Cloud Messaging is integrated to send push notifications to mobile devices, while

OTP/email services support account verification and password recovery. These technologies support the selected three-tier architecture and help Taskera provide real-time and asynchronous features without changing the main architectural model.

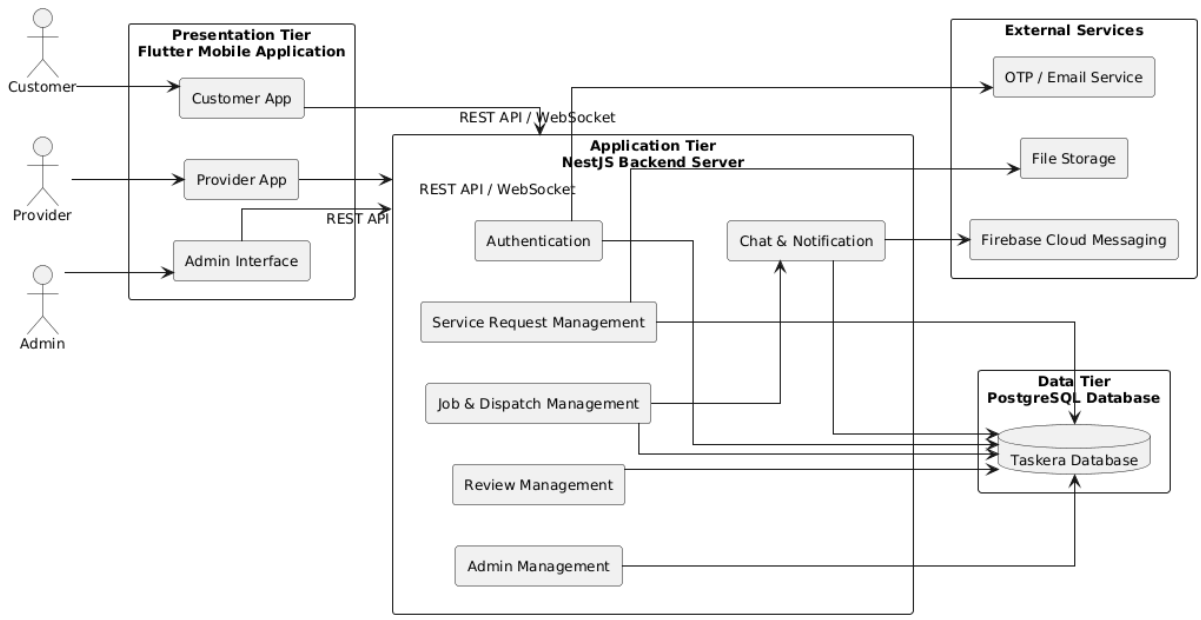


Figure 4-1: Taskera Three-Tier System Architecture Diagram

4.1.3. UML Package Diagram and Package Dependencies

The packages are arranged by layer to avoid a disorganized structure. Packages in the presentation tier are responsible for user interface screens and client-side communication. Packages in the application tier are responsible for backend business logic. Packages in the data tier are responsible for data access and persistence.

The dependencies between packages follow the layered design rule. Packages in the presentation tier depend on backend packages through API and WebSocket communication. Packages in the application tier depend on the data access package to read and write data. However, packages in lower tiers do not depend on packages in higher tiers. For example, the database package does not depend on the backend business modules, and backend modules do not depend on Flutter UI packages. This dependency direction reduces coupling, prevents circular dependencies, and makes the system easier to maintain.

1. Communication Flow

- The Flutter mobile application communicates with the NestJS backend server through REST API and WebSocket.
- Backend controllers receive incoming requests and pass them to services.
- Services process business logic and access data through repositories.
- Repositories interact with the PostgreSQL database.
- Services can also connect to external supporting services such as OTP/Email Service, Firebase Cloud Messaging, and File Storage Service.

2. Design Rules Compliance

- Lower layers do not depend on upper layers.
  - For example, repositories do not call controllers, and the database does not depend on services.
- No layer skipping.

Troy University Resources

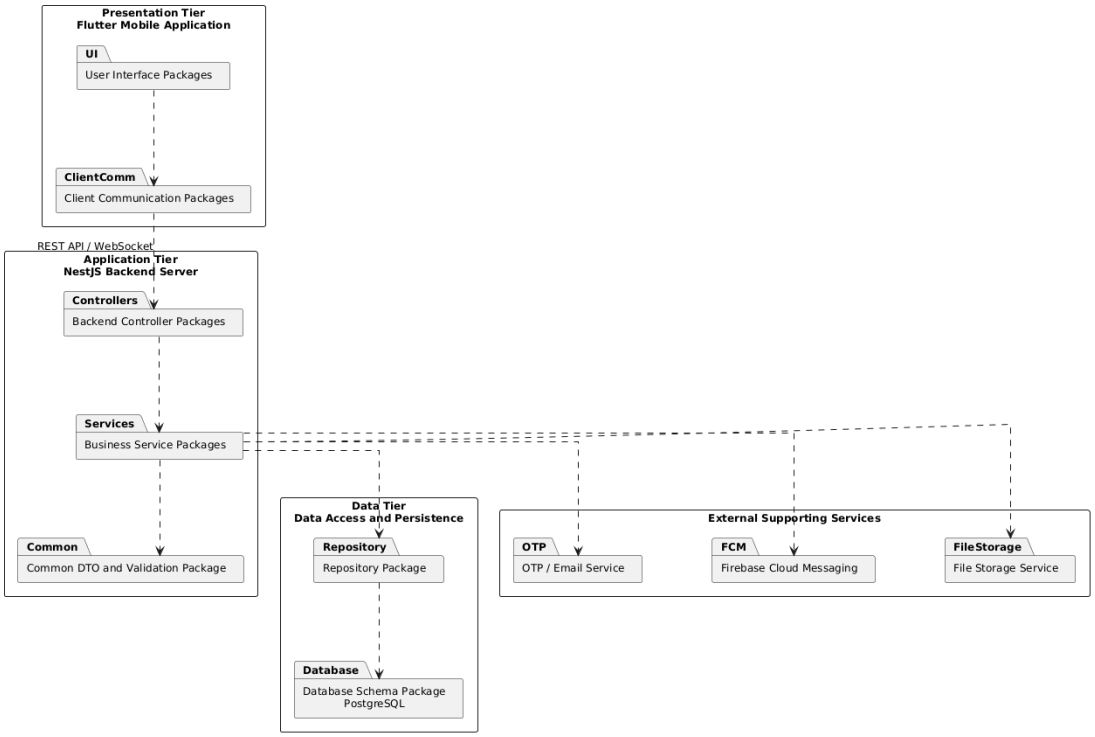


Figure 4-2: UML Package Diagram - Three-Tier Overview

  - The normal flow is: Flutter Mobile App → Controllers → Services → Repository → Database.

- No circular dependencies.
  - Each layer has a clear direction of dependency.
- Each package has a clear responsibility.
  - UI handles interface display.
  - ClientComm handles communication.
  - Controllers handle incoming requests.
  - Services handle business logic.
  - Repository handles database access.
  - External services handle OTP, notification, and file storage.

1. Presentation Tier

- Flutter Mobile Application: contains all mobile client packages used by different user roles.
- ProviderUI: contains UI pages and screens for service providers.
- CustomerUI: contains UI pages and screens for customers.
- AdminUI: contains UI pages and screens for administrators.
- AuthUI: contains authentication-related screens such as login, register, and account access pages.
- SharedUI: contains reusable UI components used by multiple UI packages, such as common buttons, input fields, layouts, cards, and shared widgets.
- APIClient: handles REST API communication between the Flutter application and the backend API.
- SocketClient: handles real-time WebSocket communication, mainly for chat or live updates.
- NotificationClient: handles notification-related communication between the mobile application and the backend notification package.

2. Communication Flow

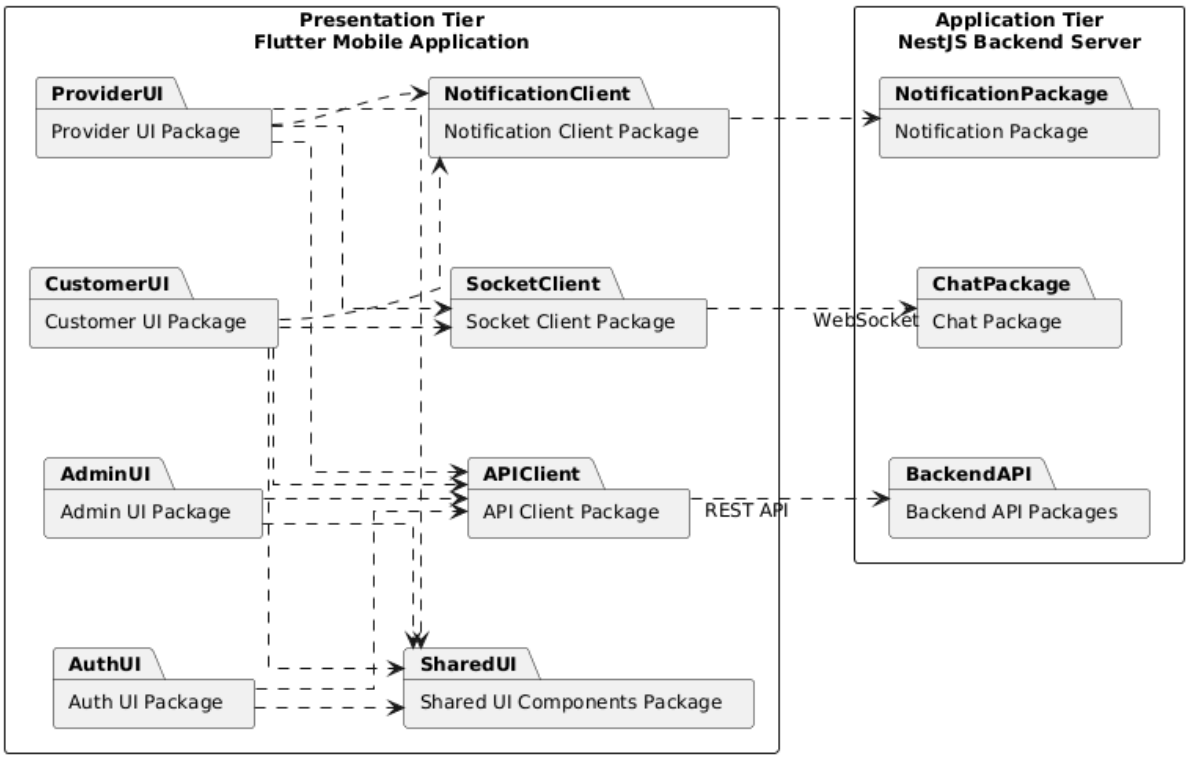


Figure 4-3: UML Package Diagram - Presentation Tier

- The UI packages such as ProviderUI, CustomerUI, AdminUI, and AuthUI use shared components from SharedUI to keep the interface consistent.
- The frontend communicates with the backend through different client packages depending on the feature.
- APIClient sends REST API requests to BackendAPI.
- SocketClient communicates with ChatPackage through WebSocket for real-time features.
- NotificationClient communicates with NotificationPackage to support notification-related features.
- Each UI package can use the required client package depending on its function, such as API requests, chat communication, or notifications.

1. Application Tier

- Admin: contains the administration package used for admin-related functions, such as managing users, service requests, jobs, and system-level operations.
- Request: contains the service request package. This package handles customer service requests, including creating requests, updating request status, and coordinating with job dispatch logic.
- User: contains the user management package. It manages user information, user profiles, roles, and account-related operations.
- Job: contains the job and dispatch package. It handles job assignment, dispatching service providers, and managing job progress.
- Auth: contains the authentication package. It handles login, registration, authentication, authorization, and account verification.
- Communication: contains the communication package. It supports communication-related features such as messaging, notifications, and interaction between users and providers.

2. Data Tier

- Repository: acts as the repository layer between the backend packages and the database. It provides database access methods for reading, creating, updating, and deleting data.

Troy University Resources

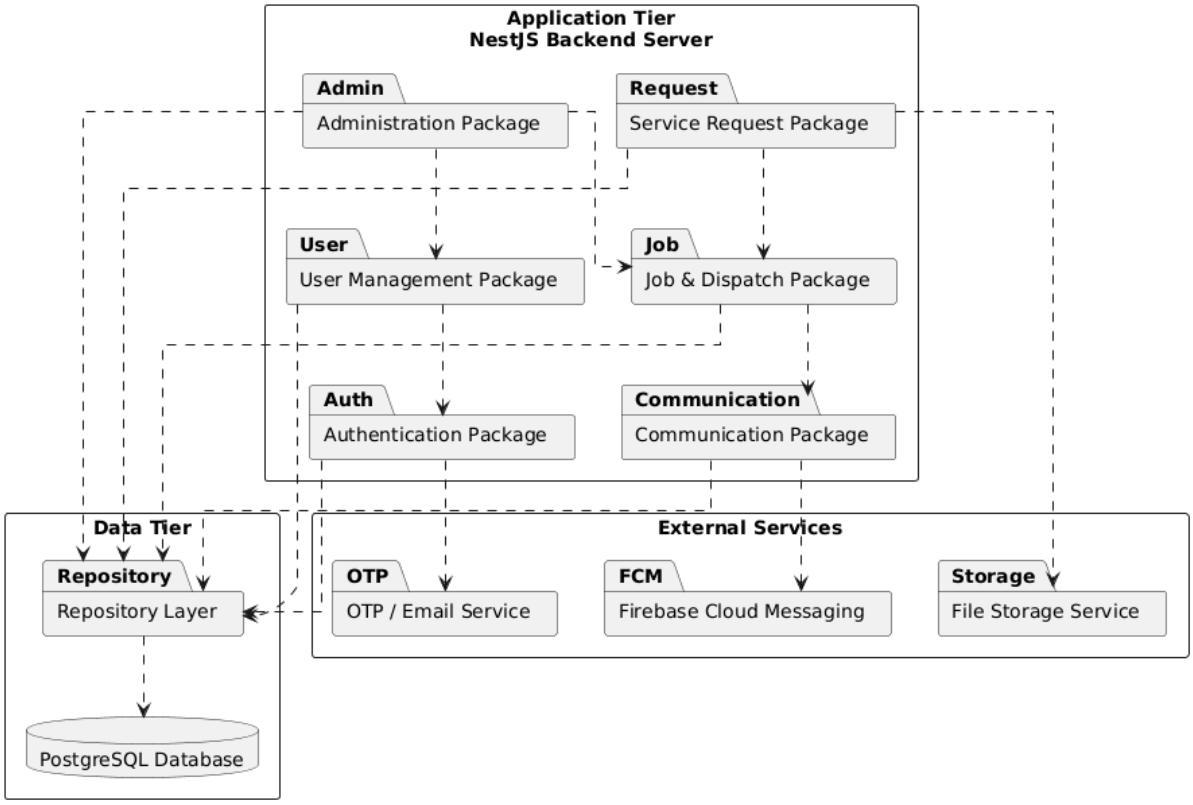


Figure 4-4: UML Package Diagram - Application and Data Tier

- PostgreSQL Database: stores the main system data, including users, service requests, jobs, authentication data, and other application records.

3. External Services

- OTP / Email Service: supports OTP verification and email-based communication, mainly used by the authentication package.
- FCM: uses Firebase Cloud Messaging to send push notifications to users.
- Storage: provides file storage service for storing uploaded files, images, and other related documents.

4.2. Database Design

The database design of the Taskera system is organized into several functional groups: account and authentication management, customer management, provider management, service catalog management, service request and dispatch management, job execution and review management, and communication and notification management. These groups are designed to support the full workflow of the platform, from user registration and service booking to provider dispatching, job completion, messaging, notification, and customer feedback.

1. Account and Authentication Group

- The account and authentication group stores the basic information of system users and supports login, verification, password reset, notification, and device-token management.
- The users table stores the main account information of all users in the system, including email, phone number, full name, password hash, role, verification status, and account status. The role field is used to distinguish between different user types, such as customer, provider, and administrator.
- The verification_tokens table stores verification tokens used for account verification or authentication-related confirmation. Each verification token belongs to exactly one user.

- The notifications table stores notification records sent to users. These records include notification title, body, type, read status, and creation time.
- The device_tokens table stores Firebase Cloud Messaging device tokens used to send push notifications to user devices. One user can have multiple device tokens because the same account may be used on different devices.

2. Customer Management Group

- The customer management group stores customer-specific information and saved addresses used during the service booking process.
- The cust_profiles table stores customer profile information, including display name, date of birth, and preferred language. Each customer profile belongs to exactly one user.
- The customer_addresses table stores customer address information, including label, street address, latitude, longitude, and default address status. One customer profile can have many saved addresses, and one address can be selected when creating a service request.

3. Provider Management Group

- The provider management group stores provider profile information, skills, working areas, availability, and portfolio posts.
- The provider_profiles table stores the main information of service providers, including display name, biography, working status, verification status, and average rating. This table is connected to the users table because each provider profile belongs to one registered user.
- The provider_skills table stores the service categories that a provider can perform. Because one provider can have many skills and one service category can be linked to many providers, this table helps represent the relationship between providers and service categories.
- The provider_service_areas table stores the areas where a provider can work, including area name, center location, and radius.
- The provider_availability table stores the availability status and working schedule of each provider. It helps the system determine whether a provider can receive job offers.
- The provider_posts table stores provider portfolio posts, including title, description, image URL, and creation time. These posts help customers view provider experience and previous work.

4. Service Catalog Group

- The service catalog group stores the list of home services that customers can book.
- The service_categories table stores service category information such as category name, slug, description, parent category, and booking schema update time. This table supports the display and organization of different service types, such as cleaning, plumbing repair, electrical repair, and maintenance.
- The service_categories table may also have a self-referencing relationship through parentCategory, allowing the system to organize services into parent and child categories when necessary.

5. Service Request and Dispatch Group

- The service request and dispatch group manages customer booking requests and the process of finding suitable providers.
- The service_requests table stores booking requests created by customers. It includes customer profile ID, service category ID, selected address ID, title, description, preferred time, offered price, status, and related scheduling information. This is one of the central tables of the system because it represents the customer's demand for a home service.
- The service_request_images table stores images attached to a service request. One service request can have multiple images, and these images help providers better understand the customer's problem before accepting the job.
- The dispatch_logs table stores dispatching attempts for a service request. It records information

Troy University Resources

such as request ID, dispatch round, status, timeout duration, start time, and completion time.

- The dispatch_candidates table stores the list of provider candidates selected during a dispatch round. It includes provider profile ID, score, distance, estimated fee, response time, and assignment status. This table supports the provider matching and dispatching process.

6. Job Execution and Review Group

- The job execution and review group manages assigned jobs, job progress, job assignment, and customer reviews.
- The job_assignments table stores assignment records between service requests and providers. It records whether the assignment is manual or automatically created by the system.
- The jobs table stores the actual job created after a provider is assigned. It includes job assignment ID, job status, start time, completion time, agreed fee, service fee percentage, and score fee. This table is used to track the lifecycle of a job from assignment to completion or cancellation.
- The job_status_logs table stores the status change history of jobs. It records the previous status, new status, and related job ID. This makes it easier to track the progress of a job over time.
- The reviews table stores customer reviews for completed jobs. It includes job ID, author ID, target user or provider ID, rating score, comment, and creation time. Each review is linked to a job and is used to evaluate provider service quality.

7. Communication and Notification Group

- The communication and notification group supports chat messages, user notifications, notification preferences, and uploaded files.
- The messages table stores chat messages between customers and providers. Each message is linked to a job assignment and a sender. It contains message content, image URL if any, and creation time.
- The notifications table stores notification records sent to users. Notifications may include job offers, job status updates, chat message alerts, provider approval results, and system alerts.
- The notification_preferences table stores user notification settings, such as job update notifications, chat message notifications, promotional notifications, system notifications, and push notification status. Each user has at most one notification preference record.
- The upload_files table stores uploaded file information, including uploader ID, related entity ID, file URL, file name, and creation time. It supports uploaded images for service requests, provider documents, messages, and review attachments.

The PostgreSQL database is designed through Prisma ORM to keep the data model consistent and maintainable. Important categorical fields, such as request status, job status, provider verification status, and notification type, are controlled by Enum types to prevent invalid values. User passwords are hashed before being stored to improve security. The schema also defines primary keys and foreign keys for entities such as users, profiles, service requests, jobs, messages, notifications, and reviews, helping maintain clear relationships and referential integrity across the system.

- Relationship diagram between tables:

The main relationships between data entities:

No.	Entity A	Entity B	Relationship	Description
1	users	cust_profiles	One-to-one	One user can have one customer profile, and each customer profile belongs to exactly one user.
2	users	provider_profiles	One-to-one	One user can have one provider profile, and each provider profile belongs to exactly one user.

No.	Entity A	Entity B	Relationship	Description
3	users	verification_tokens	One-to-many	One user can have many verification tokens over time.
4	users	notifications	One-to-many	One user can receive many notifications.
5	users	device_tokens	One-to-many	One user can register multiple device tokens.
6	cust_profiles	customer_addresses	One-to-many	One customer profile can store many addresses.
7	cust_profiles	service_requests	One-to-many	One customer can create many service requests.
8	service_categories	service_requests	One-to-many	One service category can be selected in many service requests.
9	service_categories	service_categories	Self-referencing	Service categories may reference parent and child categories.
10	provider_profiles	provider_skills	One-to-many	One provider can register many skills.
11	provider_profiles	provider_service_areas	One-to-many	One provider can define many working areas.
12	provider_profiles	provider_availability	One-to-one	Each provider has one availability record.
13	provider_profiles	provider_posts	One-to-many	One provider can create many portfolio posts.
14	service_requests	service_request_images	One-to-many	One service request can contain many attached images.
15	service_requests	dispatch_logs	One-to-many	One service request can go through many dispatching rounds.
16	dispatch_logs	dispatch_candidates	One-to-many	One dispatch log can contain many provider candidates.
17	provider_profiles	dispatch_candidates	One-to-many	One provider can appear in many dispatch candidate records.
18	service_requests	job_assignments	One-to-many	One service request can be assigned to one or more providers depending on the service requirement.
19	job_assignments	jobs	One-to-one	Each assignment can generate one actual job record.
20	jobs	job_status_logs	One-to-many	One job can have many status log records.
21	job_assignments	messages	One-to-many	One job assignment can contain many chat messages.
22	users	messages	One-to-many	One user can send many messages.
23	jobs	reviews	One-to-one	One completed job can receive one review.
24	users	reviews	One-to-many	One user can write many reviews.

Troy University Resources

No.	Entity A	Entity B	Relationship	Description
25	provider_profiles / users	reviews	One-to-many	Provider profiles or target users can receive many reviews, depending on how the review target is stored in the database.
26	users	upload_files	One-to-many	One user can upload many files.

4.2.1. Detailed Specifications of Selected Database Tables

The users table stores information about all application users, including customers, providers, and administrators. It contains login data, personal information, role information, verification status, and account status.

Table 4-2: Specification of Table “users”

Field Name	Data Type	Size	Integrity Constraints	Format	Notes
id	VARCHAR	255	PK, NOT NULL	UUID	Primary key, auto-generated
email	VARCHAR	255	UNIQUE, NOT NULL	Email	User email used for login
phone	VARCHAR	20	UNIQUE, NULL	Digits	User phone number
fullName	VARCHAR	255	NOT NULL	String	Full name of the user
passwordHash	VARCHAR	255	NOT NULL	Encrypted string	Hashed password
role	VARCHAR	50	NOT NULL	Enum	CUSTOMER, PROVIDER, ADMIN
isVerified	BOOLEAN		NOT NULL	true/false	Indicates whether the account is verified

Relations:

- 1–1 with cust_profiles
- 1–1 with provider_profiles
- 1–N with verification_tokens
- 1–N with notifications
- 1–N with device_tokens
- 1–N with messages
- 1–N with reviews

The cust_profiles table stores customer-specific profile information used for service booking and customer management.

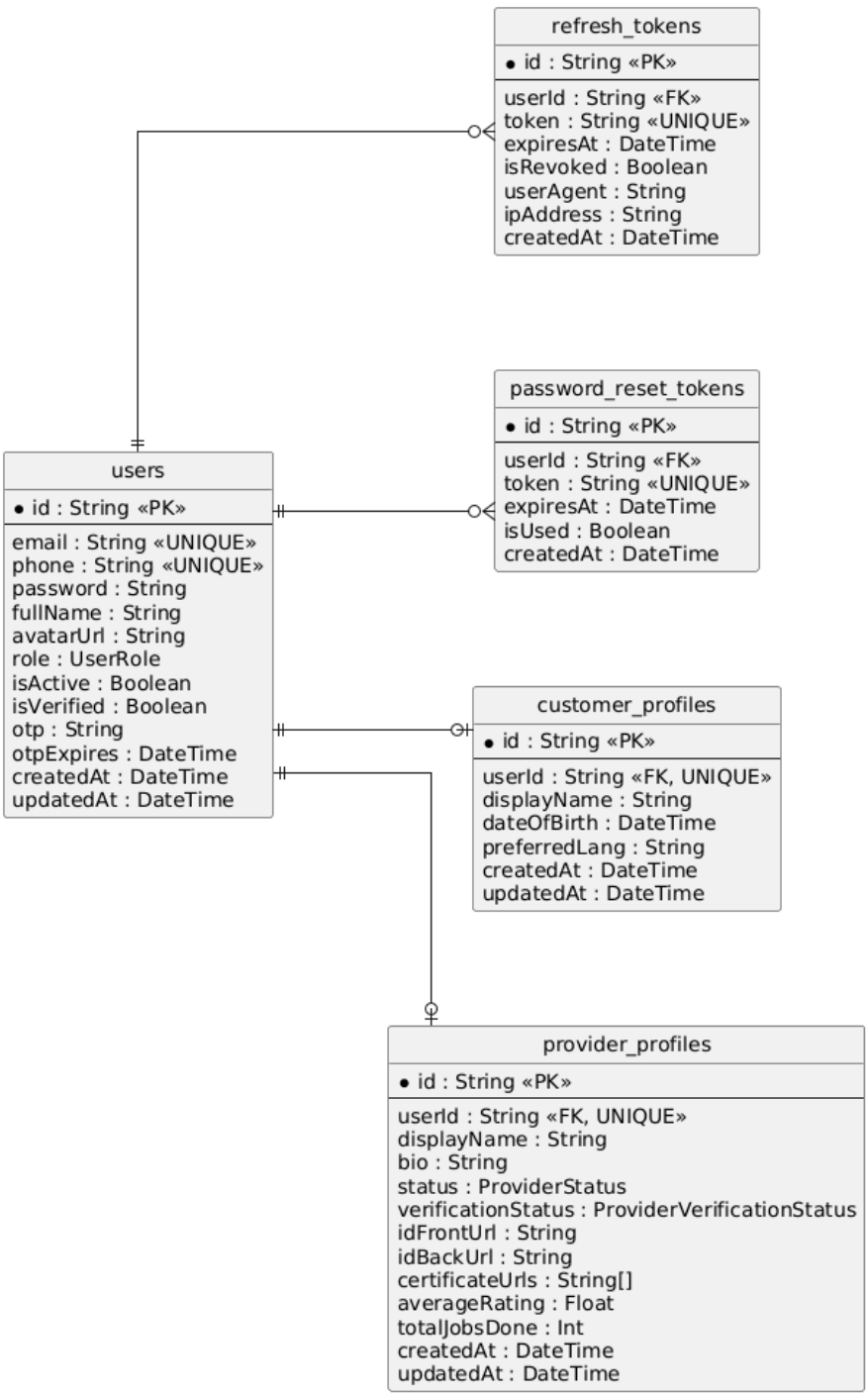


Figure 4-5: Relational Database Schema - Account, Profile and Authentication Module

Troy University Resources

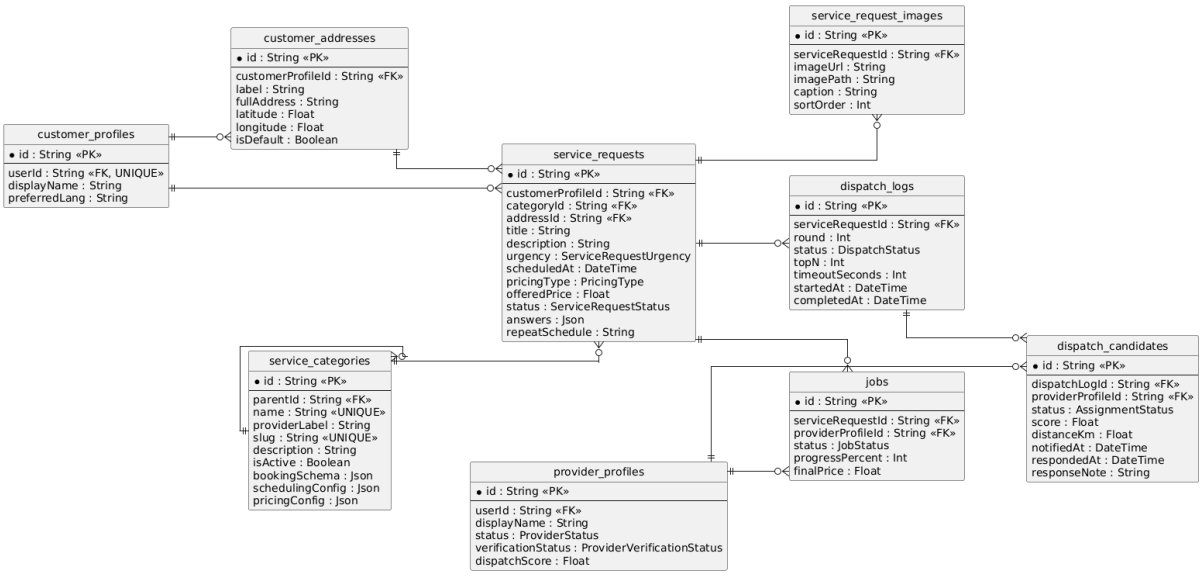


Figure 4-6: Relational Database Schema - Service Request and Dispatch Module

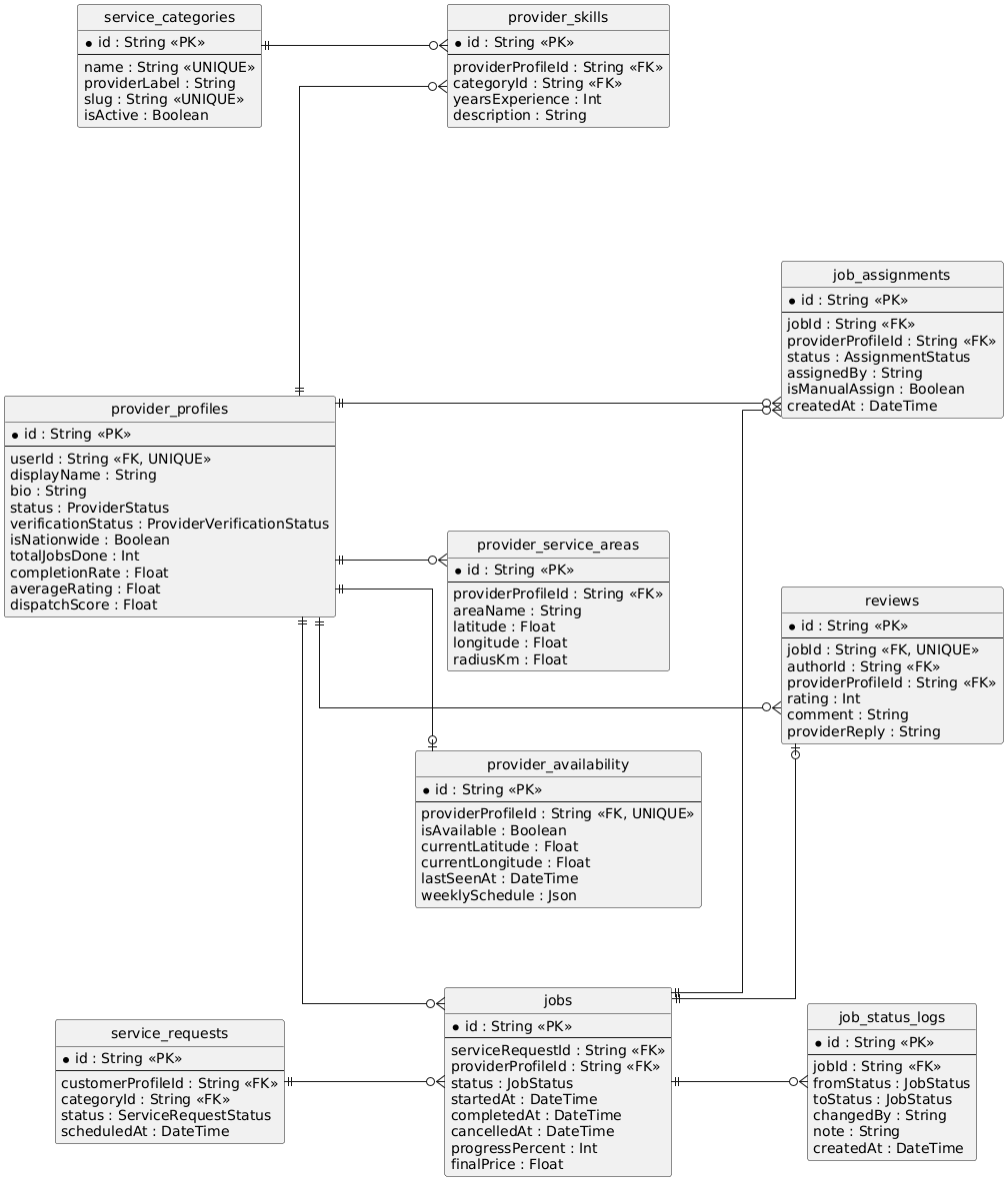


Figure 4-7: Relational Database Schema - Job Management and Provider Operations Module

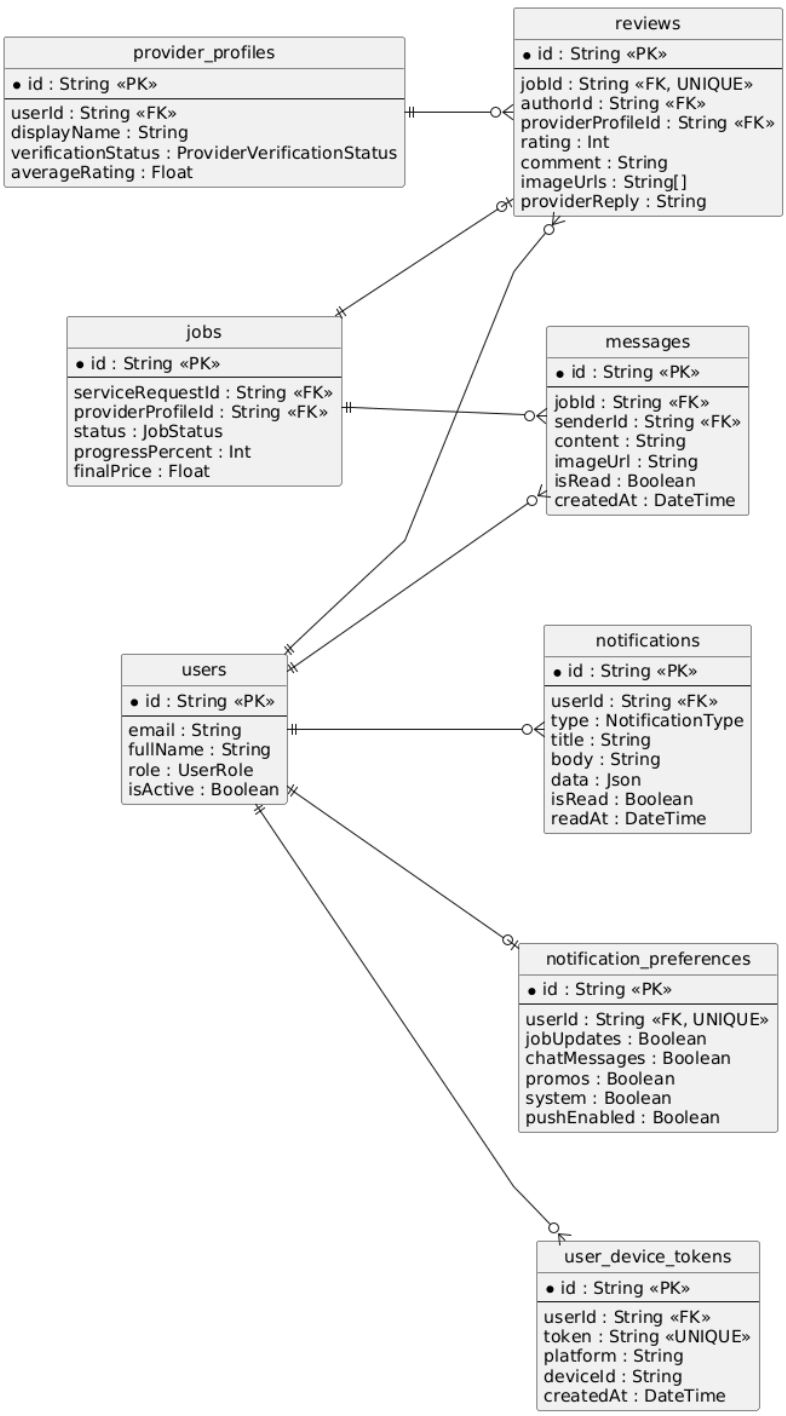


Figure 4-8: Relational Database Schema - Communication, Notification and Feedback Module

Troy University Resources

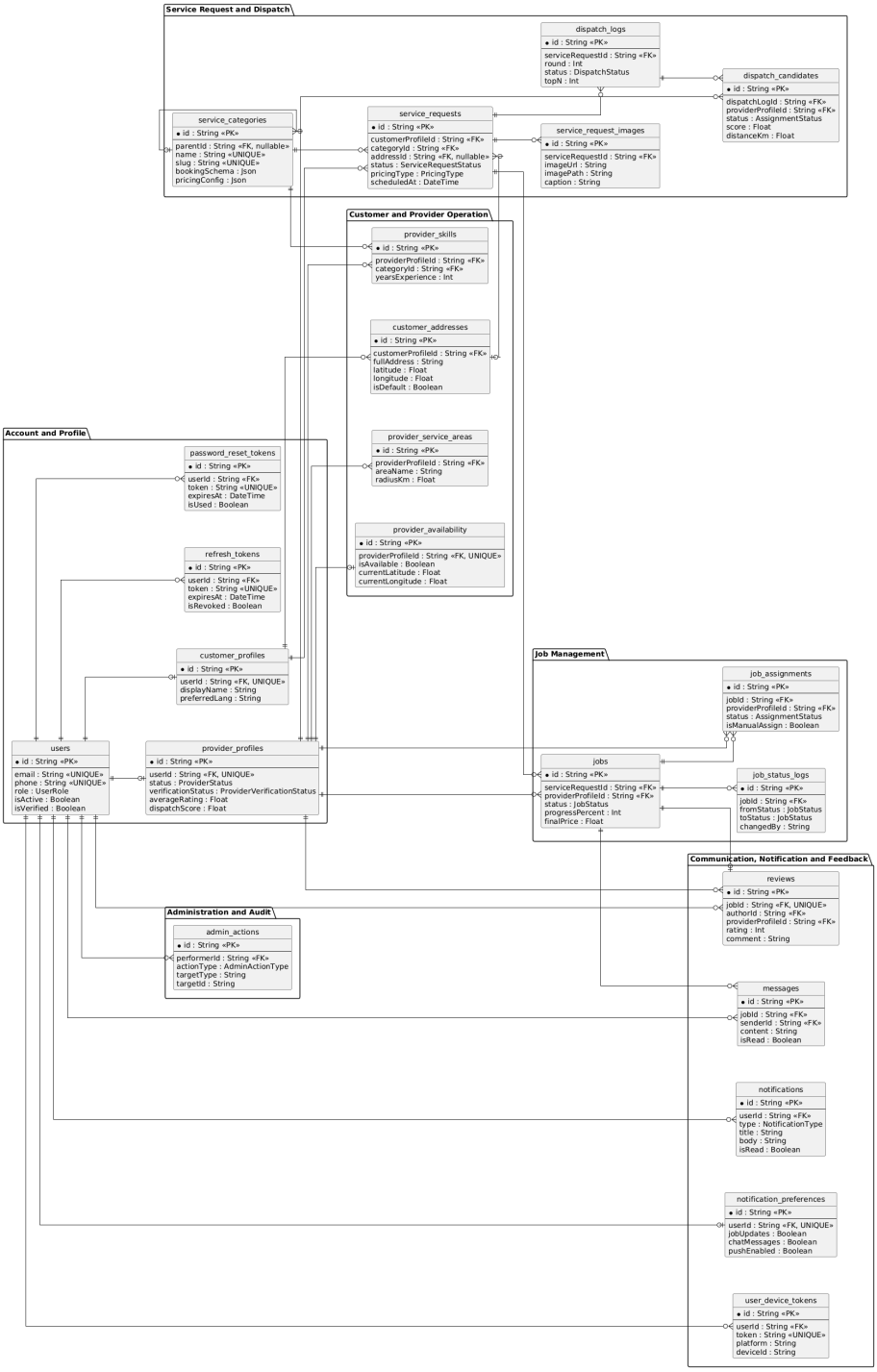


Figure 4-9: Relationship Diagram between Tables in Taskera Database

Table 4-4: Specification of Table “cust_profiles”

Field Name	Data Type	Size	Integrity Constraints	Format	Notes
id	VARCHAR	255	PK, NOT NULL	UUID	Primary key, auto-generated
userId	VARCHAR	255	FK, UNIQUE, NOT NULL	UUID	References users.id
displayName	VARCHAR	255	NULL	String	Customer display name
dateOfBirth	DATETIME		NULL	YYYY-MM-DD	Customer date of birth
preferredLang	VARCHAR	20	NULL	String	Preferred language

Relations:

- 1–1 with users
- 1–N with customer_addresses
- 1–N with service_requests

The customer_addresses table stores saved addresses of customers. These addresses can be selected when creating service requests.

Table 4-6: Specification of Table “customer_addresses”

Field Name	Data Type	Size	Integrity Constraints	Format	Notes
id	VARCHAR	255	PK, NOT NULL	UUID	Primary key, auto-generated
customerProfileId	VARCHAR	255	FK, NOT NULL	UUID	References cust_profiles.id
label	VARCHAR	100	NULL	String	Address label, such as Home or Work
streetAddress	VARCHAR	255	NOT NULL	String	Detailed address
latitude	FLOAT		NULL	Number	Address latitude
longitude	FLOAT		NULL	Number	Address longitude
isDefault	BOOLEAN		NOT NULL	true/false	Indicates whether this is the default address

Relations:

- N–1 with cust_profiles
- 1–N with service_requests

The provider_profiles table stores service provider information, verification status, working status, and rating data.

Troy University Resources

Table 4-8: Specification of Table “provider_profiles”

Field Name	Data Type	Size	Integrity Constraints	Format	Notes
id	VARCHAR	255	PK, NOT NULL	UUID	Primary key, auto-generated
userId	VARCHAR	255	FK, UNIQUE, NOT NULL	UUID	References users.id
displayName	VARCHAR	255	NULL	String	Provider display name
bio	TEXT		NULL	Text	Provider introduction
status	VARCHAR	50	NOT NULL	Enum	ONLINE, OFFLINE, BUSY
verificationStatus	VARCHAR	50	NOT NULL	Enum	PENDING, APPROVED, REJECTED
averageRating	FLOAT		NULL	Number	Average rating of the provider

Relations:

- 1–1 with users
- 1–N with provider_skills
- 1–N with provider_service_areas
- 1–1 with provider_availability
- 1–N with provider_posts
- 1–N with dispatch_candidates
- 1–N with job_assignments
- 1–N with reviews

The service_categories table stores service categories available in the Taskera system.

Table 4-10: Specification of Table “service_categories”

Field Name	Data Type	Size	Integrity Constraints	Format	Notes
id	VARCHAR	255	PK, NOT NULL	UUID	Primary key, auto-generated
parentCategoryId	VARCHAR	255	FK, NULL	UUID	References parent category
name	VARCHAR	255	NOT NULL	String	Service category name
slug	VARCHAR	255	UNIQUE, NULL	String	Category slug
description	TEXT		NULL	Text	Category description

Relations:

- Self-referencing relationship with service_categories
- 1–N with service_requests
- 1–N with provider_skills

The service_requests table stores service booking requests created by customers.

Field Name	Data Type	Size	Integrity Constraints	Format	Notes

Table 4-12: Specification of Table “service_requests”

Field Name	Data Type	Size	Integrity Constraints	Format	Notes
id	VARCHAR	255	PK, NOT NULL	UUID	Primary key, auto-generated
customerProfileId	VARCHAR	255	FK, NOT NULL	UUID	References cust_profiles.id
categoryId	VARCHAR	255	FK, NOT NULL	UUID	References service_categories.id
addressId	VARCHAR	255	FK, NULL	UUID	References customer_addresses.id
targetProviderId	VARCHAR	255	FK, NULL	UUID	References provider_profiles.id
title	VARCHAR	255	NOT NULL	String	Request title
description	TEXT		NOT NULL	Text	Request description
preferredDateTime	DATETIME		NULL	YYYY-MM-DD HH:mm:ss	Preferred service time
status	VARCHAR	50	NOT NULL	Enum	PENDING, DISPATCHING, ASSIGNED, COMPLETED, CANCELLED
offeredPrice	FLOAT		NULL	Number	Price offered by customer

Relations:

- N–1 with cust_profiles
- N–1 with service_categories
- N–1 with customer_addresses
- N–1 with provider_profiles through targetProviderId
- 1–N with service_request_images
- 1–N with dispatch_logs
- 1–N with job_assignments

The dispatch_logs table records dispatching rounds for service requests.

Table 4-14: Specification of Table “dispatch_logs”

Field Name	Data Type	Size	Integrity Constraints	Format	Notes
id	VARCHAR	255	PK, NOT NULL	UUID	Primary key, auto-generated
serviceRequestId	VARCHAR	255	FK, NOT NULL	UUID	References service_requests.id
round	INT		NOT NULL	Number	Dispatch round number
status	VARCHAR	50	NOT NULL	Enum	PENDING, SENT, ACCEPTED, TIMEOUT, CANCELLED
timeoutSeconds	INT		NOT NULL	Number	Timeout duration
startedAt	DATETIME		NOT NULL	YYYY-MM-DD HH:mm:ss	Dispatch start time

Relations:

- N–1 with service_requests
- 1–N with dispatch_candidates

The dispatch_candidates table stores provider candidates selected in a dispatching round.

Troy University Resources

Table 4-16: Specification of Table “dispatch_candidates”

Field Name	Data Type	Size	Integrity Constraints	Format	Notes
id	VARCHAR	255	PK, NOT NULL	UUID	Primary key, auto-generated
dispatchLogId	VARCHAR	255	FK, NOT NULL	UUID	References dispatch_logs.id
providerProfileId	VARCHAR	255	FK, NOT NULL	UUID	References provider_profiles.id
score	FLOAT		NULL	Number	Provider ranking score
distanceKm	FLOAT		NULL	Number	Distance between provider and customer
estimatedFee	FLOAT		NULL	Number	Estimated service fee
respondedAt	DATETIME		NULL	YYYY-MM-DD HH:mm:ss	Provider response time
assignmentStatus	VARCHAR	50	NOT NULL	Enum	PENDING, ACCEPTED, REJECTED, TIMEOUT

Relations:

- N–1 with dispatch_logs
- N–1 with provider_profiles

The jobs table stores actual jobs created after service assignment.

Table 4-18: Specification of Table “jobs”

Field Name	Data Type	Size	Integrity Constraints	Format	Notes
id	VARCHAR	255	PK, NOT NULL	UUID	Primary key, auto-generated
jobAssignmentId	VARCHAR	255	FK, NOT NULL	UUID	References job_assignments.id
status	VARCHAR	50	NOT NULL	Enum	ASSIGNED, ON_THE_WAY, IN_PROGRESS, COMPLETED, CANCELLED
startedAt	DATETIME		NULL	YYYY-MM-DD HH:mm:ss	Job start time
completedAt	DATETIME		NULL	YYYY-MM-DD HH:mm:ss	Job completion time
agreedFee	FLOAT		NULL	Number	Agreed service fee
serviceFeePercent	FLOAT		NULL	Number	Service fee percentage
scoreFee	FLOAT		NULL	Number	Score or calculated fee value

Relations:

- 1–1 with job_assignments
- 1–N with job_status_logs
- 1–1 with reviews

The job_assignments table stores assignment records between service requests and providers.

Field Name	Data Type	Size	Integrity Constraints	Format	Notes

Table 4-20: Specification of Table “job_assignments”

Field Name	Data Type	Size	Integrity Constraints	Format	Notes
id	VARCHAR	255	PK, NOT NULL	UUID	Primary key, auto-generated
serviceRequestId	VARCHAR	255	FK, NOT NULL	UUID	References service_requests.id
providerProfileId	VARCHAR	255	FK, NULL	UUID	References provider_profiles.id
assignedAt	DATETIME		NULL	YYYY-MM-DD HH:mm:ss	Assignment time
isManualAssign	BOOLEAN		NOT NULL	true/false	Indicates whether the assignment is created manually

Relations:

- N–1 with service_requests
- N–1 with provider_profiles
- 1–1 with jobs
- 1–N with messages

The messages table stores chat messages between users related to a job assignment.

Table 4-22: Specification of Table “messages”

Field Name	Data Type	Size	Integrity Constraints	Format	Notes
id	VARCHAR	255	PK, NOT NULL	UUID	Primary key, auto-generated
jobAssignmentId	VARCHAR	255	FK, NOT NULL	UUID	References job_assignments.id
senderId	VARCHAR	255	FK, NOT NULL	UUID	References users.id
content	TEXT		NOT NULL	Text	Message content
imageUrl	VARCHAR	255	NULL	URL	Optional image attachment
createdAt	DATETIME		NOT NULL	YYYY-MM-DD HH:mm:ss	Message creation time

Relations:

- N–1 with job_assignments
- N–1 with users

The notifications table stores system notification records sent to users.

Table 4-24: Specification of Table “notifications”

Field Name	Data Type	Size	Integrity Constraints	Format	Notes
id	VARCHAR	255	PK, NOT NULL	UUID	Primary key, auto-generated
userId	VARCHAR	255	FK, NOT NULL	UUID	References users.id
title	VARCHAR	255	NOT NULL	String	Notification title
body	TEXT		NULL	Text	Notification body
type	VARCHAR	50	NOT NULL	Enum	Notification type
isRead	BOOLEAN		NOT NULL	true/false	Read status

Troy University Resources

Field Name	Data Type	Size	Integrity Constraints	Format	Notes
createdAt	DATETIME		NOT NULL	YYYY-MM-DD HH:mm:ss	Notification creation time

Relations:

- N–1 with users

The reviews table stores customer reviews and feedback after job completion.

Table 4-26: Specification of Table “reviews”

Field Name	Data Type	Size	Integrity Constraints	Format	Notes
id	VARCHAR	255	PK, NOT NULL	UUID	Primary key, auto-generated
jobId	VARCHAR	255	FK, NOT NULL	UUID	References jobs.id
authorId	VARCHAR	255	FK, NOT NULL	UUID	References users.id
targetUserId	VARCHAR	255	FK, NULL	UUID	References reviewed user
rating	INT		NOT NULL	1–5	Rating score
comment	TEXT		NULL	Text	Review content
createdAt	DATETIME		NOT NULL	YYYY-MM-DD HH:mm:ss	Review creation time

Relations:

- N–1 with jobs
- N–1 with users as author
- N–1 with target user or provider profile

4.3. Detailed Package Design

This section presents the detailed design of several important backend packages in the Taskera system. While the architectural design section describes the overall package organization, this section focuses on the internal structure of selected packages. Each diagram shows the main classes inside one package or a group of related packages. Following the design requirement, only class names are shown in these diagrams, while detailed attributes and methods are described later in the detailed class design section.

The selected packages include Authentication, Service Request and Dispatch, Job Management, Communication and Notification, and Admin Management. These packages are selected because they represent the core workflow of Taskera, from user authentication and service booking to provider dispatching, job tracking, real-time communication, notification delivery, and administrative control.

The Authentication Package handles account registration, OTP verification, login, token refresh, logout, forgot password, and password reset. AuthController receives authentication requests from the client and forwards them to AuthService. DTO classes are used to validate input data before processing. AuthService depends on EmailService for sending OTP and password reset emails, and on PrismaService for reading and updating authentication-related data.

The Service Request Package manages customer booking requests, including creating requests, viewing request history, browsing available requests, viewing request details, and cancelling requests. The Dispatch Package is responsible for matching service requests with suitable providers. DispatchService checks provider availability, creates dispatch records, sends job offers, and processes provider acceptance or rejection. The scheduler classes support automatic dispatch behavior for timeout and scheduled requests.

The Job Management Package handles job tracking and job status updates after a provider has been

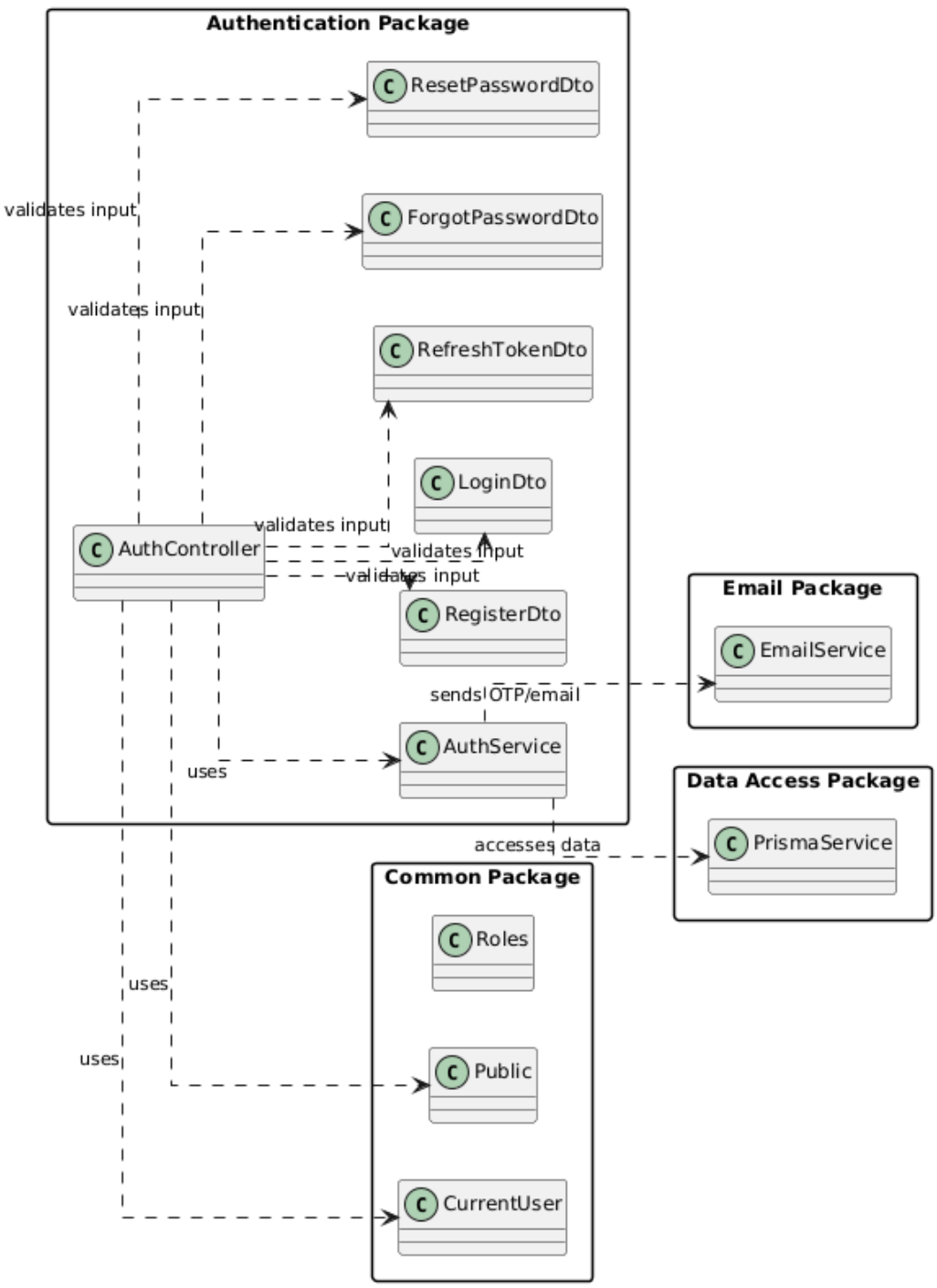


Figure 4-10: Detailed Package Design - Authentication Package

Troy University Resources

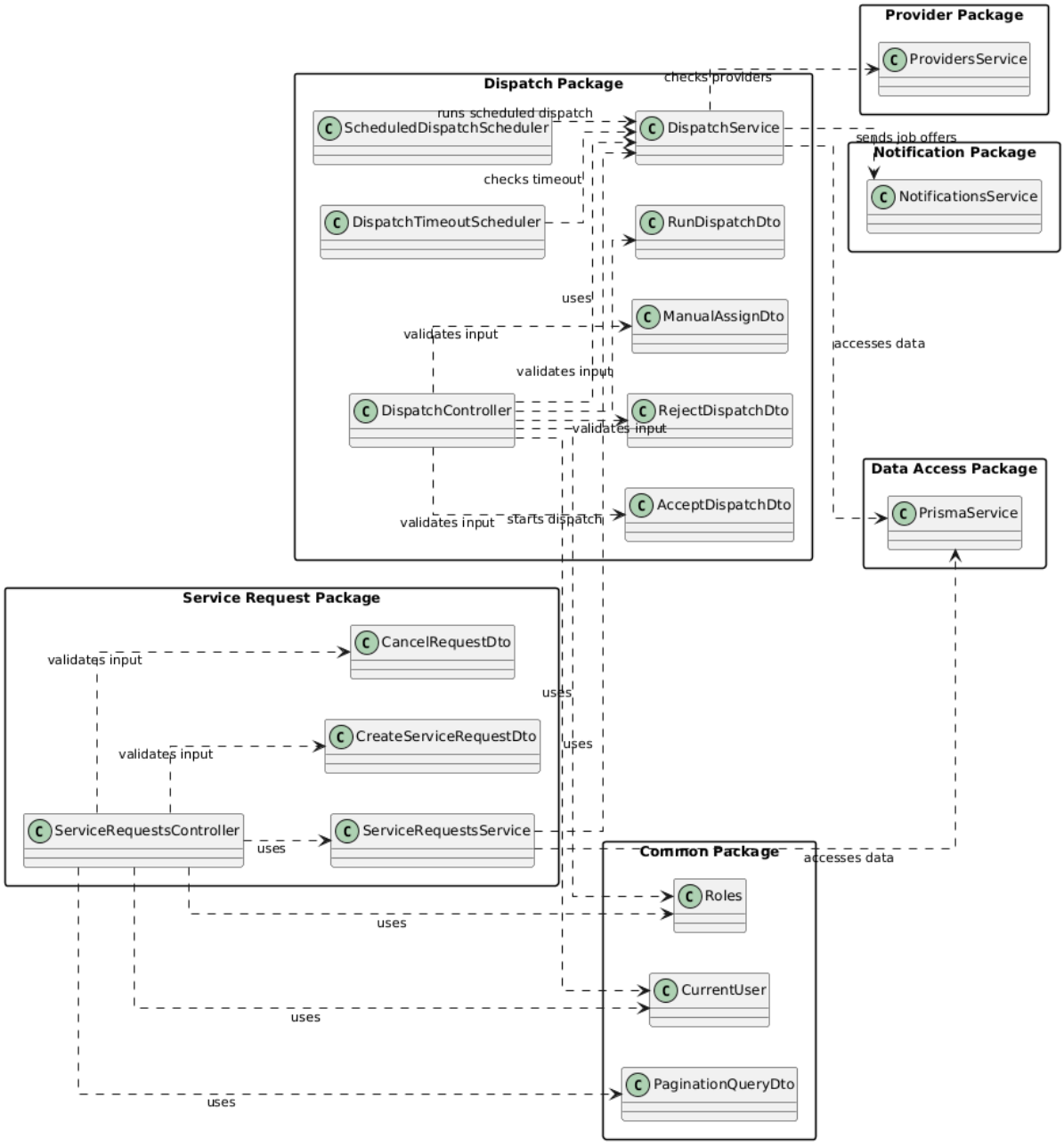


Figure 4-11: Detailed Package Design - Service Request and Dispatch Packages

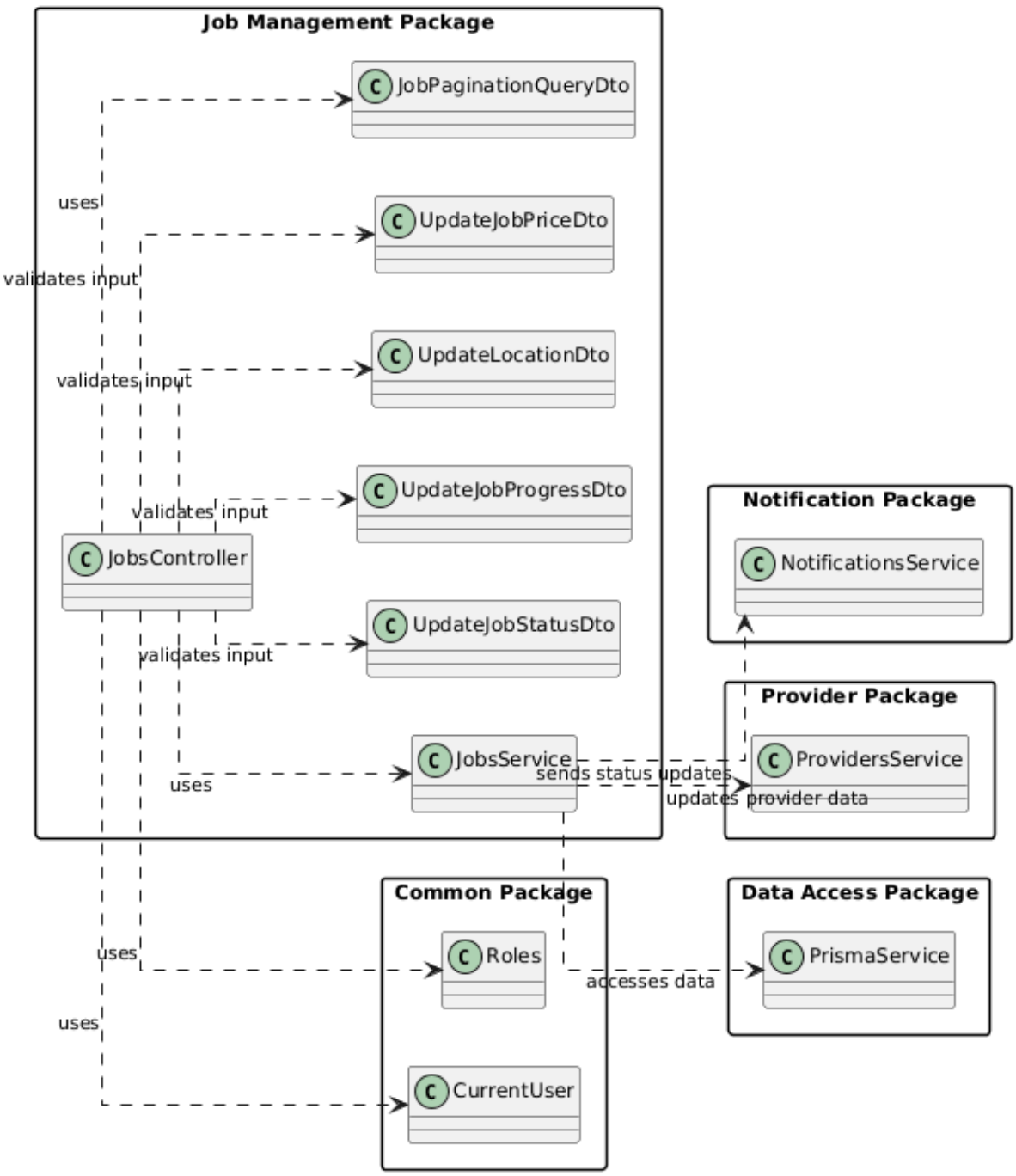


Figure 4-12: Detailed Package Design - Job Management Package

Troy University Resources

assigned. JobsController receives requests for viewing jobs, updating job status, updating progress, updating provider location, viewing job timeline, and setting the final price. JobsService contains the main job-processing logic and depends on PrismaService for database access. It also works with NotificationsService to notify users about important job status changes.

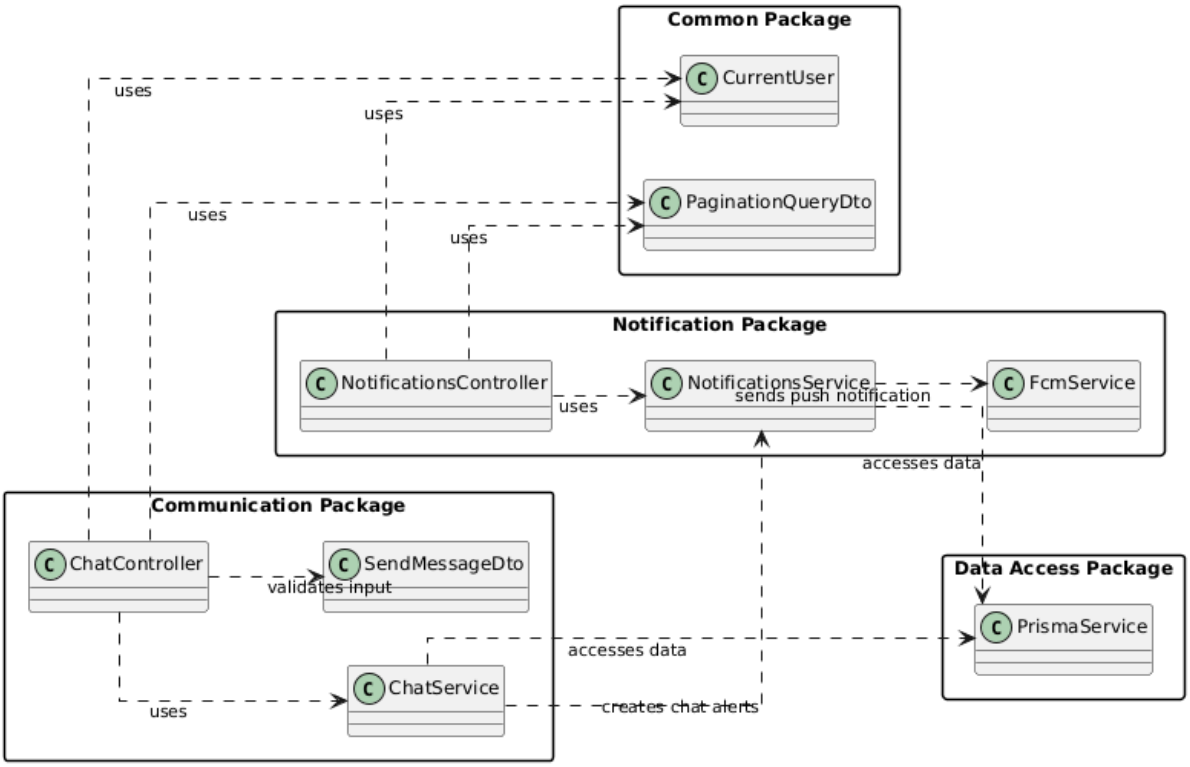


Figure 4-13: Detailed Package Design - Communication and Notification Packages

The Communication Package supports chat between customers and providers during a job. ChatController handles conversation listing, unread message count, message retrieval, message sending, marking messages as read, and deleting job messages. ChatService stores and retrieves messages through PrismaService and can trigger notification creation when a new message is sent. The Notification Package manages notification records, read status, notification preferences, and push notification delivery through FcmService.

The Admin Management Package supports administrative operations such as viewing providers, approving or rejecting provider applications, suspending or unsuspending providers, managing users, monitoring service requests, managing jobs, and reassigning providers when necessary. AdminController receives admin requests and delegates processing to AdminService. AdminService coordinates with provider, request, job, and dispatch-related packages to perform platform management tasks.

4.4. Detailed Class Design

- **User – Base Account Entity**

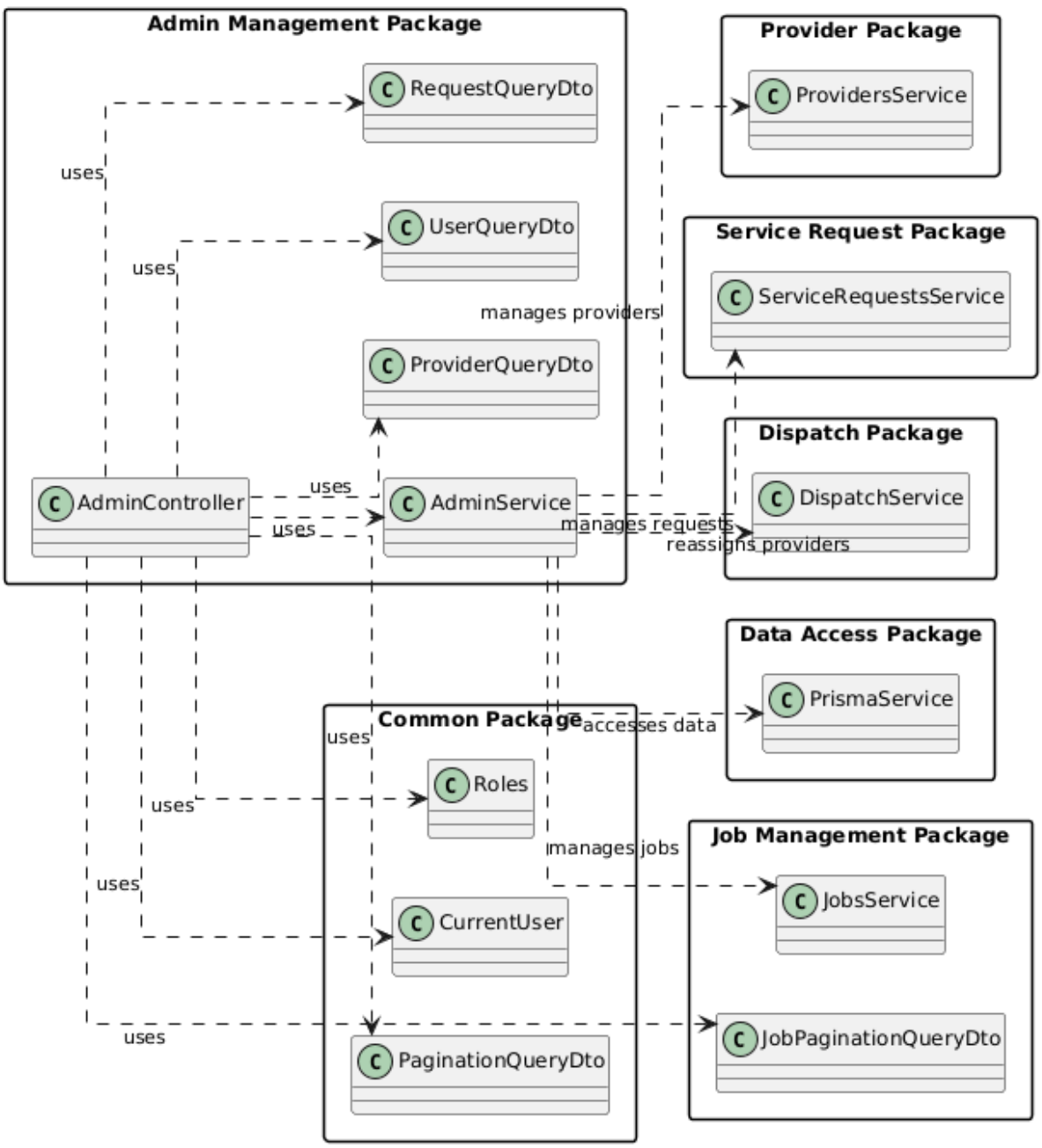


Figure 4-14: Admin Management Package

Troy University Resources

User «entity»	
• Stores common account information for Customer, Provider, and Admin. • Maintains authentication-related state such as active status, verification status, role, and credential information. • Provides account-level operations such as registration, OTP verification, password checking, password change, profile information update, and account suspension.	• CustomerProfile • ProviderProfile • RefreshToken • PasswordResetToken • Message • Notification • Review • AuthService

- **CustomerProfile – Customer Account Profile**

CustomerProfile «entity»	
• Stores customer-specific profile information such as display name, date of birth, and preferred language. • Manages saved service addresses and supports customer request history. • Provides operations for updating customer profile data, adding or removing addresses, setting a default address, and viewing service request history.	• User • CustomerAddress • ServiceRequest • RequestsService

- **CustomerAddress – Customer Service Address**

CustomerAddress «entity»	
• Stores a customer service address, including label, detailed street address, latitude, longitude, and default-address state. • Provides address update and default-address marking operations. • Checks whether the address can be used for a booking request.	• CustomerProfile • ServiceRequest

- **ProviderProfile – Provider Work Profile**

ProviderProfile «entity»	
• Stores provider information, work status, verification status, identity documents, certificates, average rating, and total completed jobs. • Determines whether a provider is eligible to receive new dispatched jobs. • Supports profile update, status update, verification submission, approval or rejection, and rating update operations.	• User • ProviderSkill • ProviderServiceArea • ProviderAvailability • DispatchCandidate • JobAssignment • Review

- **ProviderSkill – Provider Service Skill**

ProviderSkill «entity»	
• Stores a provider skill and the corresponding service category. • Records years of experience for matching and provider evaluation. • Supports skill addition, update, and removal.	• ProviderProfile • ServiceCategory • DispatchService

- **ProviderServiceArea – Provider Coverage Area**

ProviderServiceArea «entity»	
• Stores the geographical areas or radius in which a provider can accept jobs. • Supports area update and coverage checking. • Helps the dispatching process determine whether a provider can serve a customer location.	• ProviderProfile • ServiceRequest • DispatchService

- **ProviderAvailability – Provider Availability State**

ProviderAvailability «entity»	
• Stores provider availability state and current location. • Supports online/offline status changes and location updates. • Provides availability information for dispatch and job-offer decisions.	• ProviderProfile • DispatchService

- **ServiceCategory – Service Type Definition**

ServiceCategory «entity»	
• Stores service category information, including name, slug, description, and booking schema. • Provides booking schema for dynamic service request forms. • Supports service category creation, update, activation, and availability checking.	• ServiceRequest • ProviderSkill • AdminService

- **ServiceRequest – Customer Booking Request**

ServiceRequest «entity»	
• Represents a customer service booking request before it becomes a job. • Stores selected category, address, title, description, preferred time, offered price, images, target provider, and request status. • Supports submission, booking validation, cancellation, dispatch start, assignment marking, completion marking, and image attachment.	• CustomerProfile • CustomerAddress • ServiceCategory • DispatchLog • JobAssignment • RequestsService • DispatchService

- **DispatchLog – Dispatch Round Record**

DispatchLog «entity»	
• Records one dispatch round for a service request. • Stores round number, dispatch status, timeout duration, and start time. • Supports starting a round, adding candidates, marking accepted, marking timeout, and cancelling dispatch.	• ServiceRequest • DispatchCandidate • DispatchService

- **DispatchCandidate – Provider Offer Candidate**

Troy University Resources

DispatchCandidate «entity»	
• Represents a provider selected as a candidate in one dispatch round. • Stores candidate score, distance, estimated fee, response time, and assignment status. • Supports score calculation, offer sending, offer acceptance, offer rejection, and offer expiration.	• DispatchLog • ProviderProfile • JobAssignment • DispatchService

- **JobAssignment – Provider Assignment Record**

JobAssignment «entity»	
• Stores the assignment relationship between a service request and a provider. • Supports manual assignment and provider switching. • Creates a Job after a provider accepts the request and provides access to job messages.	• ServiceRequest • ProviderProfile • Job • Message • JobsService • DispatchService

- **Job – Service Job Lifecycle**

Job «entity»	
• Represents the actual service job after a provider has been assigned. • Stores job status, start time, completion time, agreed fee, service fee percentage, and final price. • Controls job lifecycle operations such as status update, progress update, final price setting, completion, cancellation, and valid status transition checking.	• JobAssignment • Message • Notification • Review • JobsService

- **Message – Job Chat Message**

Message «entity»	
• Stores a chat message exchanged between customer and provider within a job assignment. • Stores sender, content, image attachment, creation time, and read status. • Supports sending, marking as read, deletion, and job-ownership validation.	• User • JobAssignment • ChatService • ChatGateway • NotificationsService

- **Notification – User Notification Record**

Notification «entity»	
• Stores a system notification sent to a user. • Stores title, body, notification type, read state, and creation time. • Supports notification creation, read marking, push sending, and user-preference checking.	• User • NotificationsService • FcmService

- **Review – Completed Job Review**

Review «entity»	
• Stores a customer review after a completed service job. • Validates rating and ensures the review belongs to a completed job. • Updates provider rating information after a valid review is created.	• Job • User • ProviderProfile • ReviewsService

- **AuthController – Authentication API Boundary**

AuthController «boundary»	
• Receives authentication requests from the Flutter application or admin interface through REST API endpoints. • Delegates registration, OTP verification, login, token refresh, logout, forgot-password, and reset-password processing to AuthService. • Returns authentication responses and error messages to the client.	• AuthService • RegisterDto • LoginDto • VerifyOtpDto • ResetPasswordDto

- **AuthService – Authentication Business Logic**

AuthService «control»	
• Implements authentication business rules. • Handles registration, OTP generation and validation, credential checking, JWT access/refresh token issuance, logout, and password reset. • Coordinates persistence and external email delivery.	• User • RefreshToken • PasswordResetToken • PrismaService • EmailService • JwtService

- **RequestsService – Booking Request Management**

RequestsService «control»	
• Validates booking data and creates service requests. • Retrieves customer service request history and request details. • Cancels requests when business rules allow cancellation and starts dispatching when appropriate.	• ServiceRequest • CustomerProfile • CustomerAddress • ServiceCategory • DispatchService • PrismaService

- **DispatchService – Provider Dispatch Management**

DispatchService «control»	
• Finds suitable providers for a service request. • Calculates candidate scores based on provider skills, service area, availability, distance, rating, and job suitability. • Creates dispatch logs, sends job offers, processes provider acceptance or rejection, retries dispatching, and switches providers when needed.	• ServiceRequest • ProviderProfile • DispatchLog • DispatchCandidate • JobAssignment • NotificationsService • PrismaService

- **JobsService – Job Lifecycle Management**

Troy University Resources

JobsService «control»	
• Manages job details and job lifecycle after dispatching. • Validates status transitions and updates job progress, final price, completion, or cancellation. • Notifies related users about job state changes.	• Job • JobAssignment • Notification • NotificationsService • PrismaService

- **ChatService – Job Chat Management**

ChatService «control»	
• Stores and retrieves chat messages for job assignments. • Calculates unread counts and marks conversations as read. • Emits real-time messages through ChatGateway and triggers notifications when needed.	• Message • JobAssignment • User • ChatGateway • NotificationsService • PrismaService

- **NotificationsService – Notification Delivery Management**

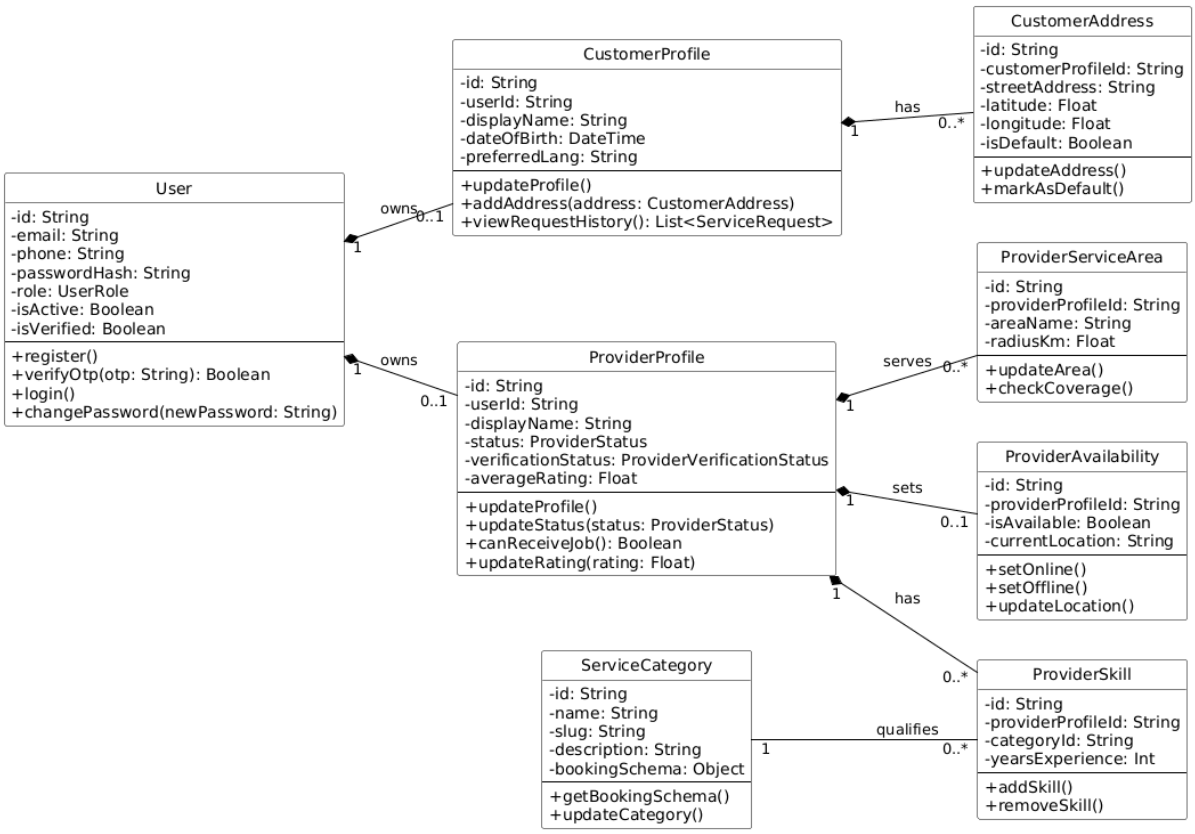
NotificationsService «control»	
• Creates notification records and applies notification preferences. • Registers or deletes FCM device tokens. • Sends push notifications through Firebase Cloud Messaging and updates read status.	• Notification • User • FcmService • PrismaService

- **AdminService – Platform Administration Management**

AdminService «control»	
• Provides administrative operations for supervising the Taskera platform. • Manages users, provider verification, service requests, jobs, and manual provider assignment. • Coordinates with DispatchService when an admin intervenes in dispatching.	• User • ProviderProfile • ServiceRequest • Job • DispatchService • PrismaService

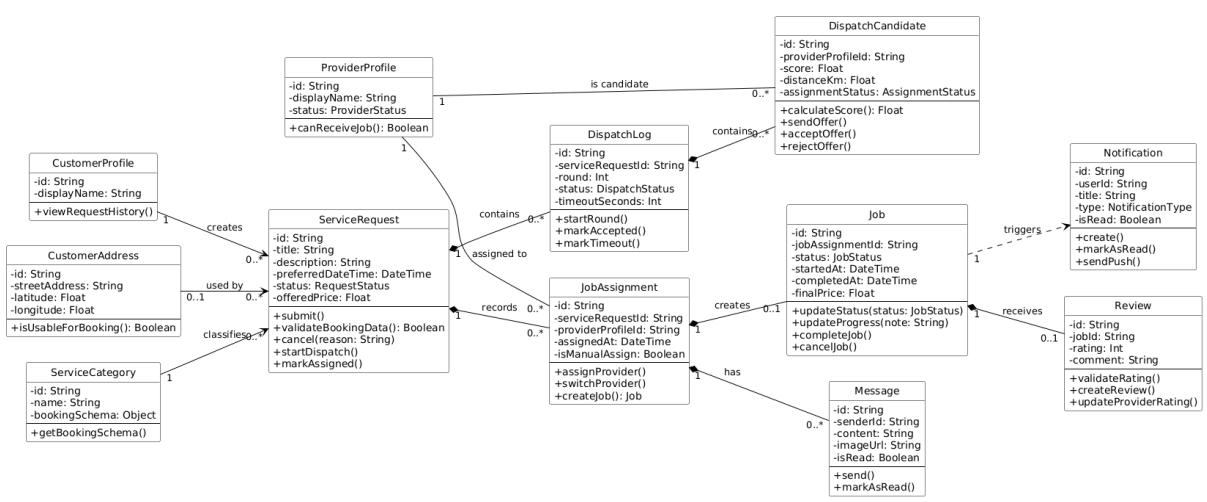
4.5. Detailed Class Diagram

- **Detailed Class Diagram - Account and Profile Classes**



This diagram shows User as the central account class. User compositionally owns optional CustomerProfile and ProviderProfile objects. CustomerProfile owns CustomerAddress objects. ProviderProfile owns ProviderSkill, ProviderServiceArea, and ProviderAvailability objects. ServiceCategory is associated with ProviderSkill to represent the service types that a provider can perform.

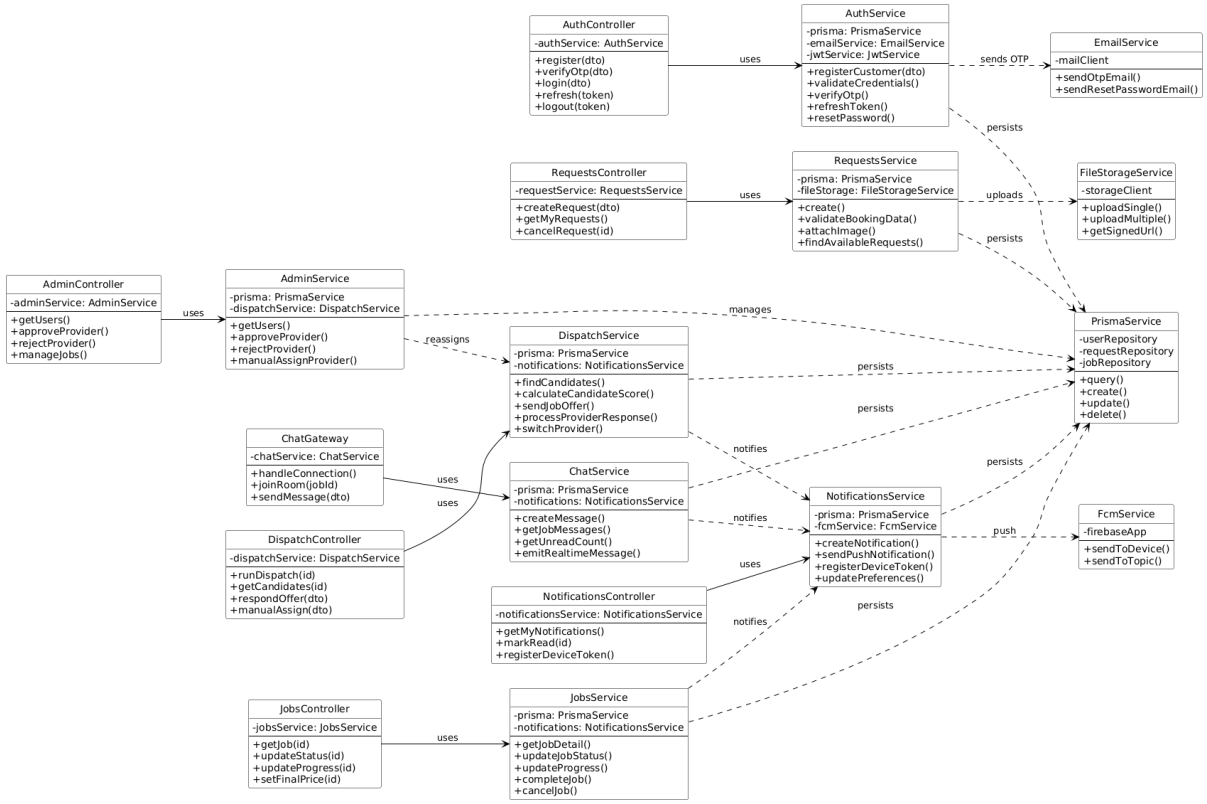
- **Detailed Class Diagram - Service Request, Dispatch, and Job Classes**



This diagram models the booking and dispatch workflow. CustomerProfile creates ServiceRequest objects using CustomerAddress and ServiceCategory. A ServiceRequest contains DispatchLog objects, and each DispatchLog contains DispatchCandidate objects. After a provider accepts an offer, JobAssignment is recorded and may create a Job. JobAssignment contains Message objects, while Job may receive Review objects and trigger Notification objects.

Troy University Resources

- **Detailed Class Diagram - Controller and Service Layer**



This diagram describes the backend control structure. Boundary classes such as controllers and gateways receive REST API or WebSocket requests and delegate processing to control classes such as AuthService, DispatchService, JobsService, ChatService, NotificationsService, and AdminService.

CHAPTER 5. IMPLEMENTATION OF THE DEMONSTRATION PROGRAM

5.1. Libraries and Tools Used

The Taskera system uses Flutter to develop a cross-platform mobile application for both customers and workers. The Backend is built using the NestJS framework, combined with Prisma ORM and the PostgreSQL database management system, to handle business logic and store data. In addition, the system integrates WebSocket for real-time messaging and Firebase Cloud Messaging, FCM, for sending push notifications.

Table 5-1: Libraries and Tools Used

Purpose	Tool	URL
Dev	VSCode	https://code.visualstudio.com
Mobile Frontend Framework	Flutter SDK	https://flutter.dev
Mobile Frontend Language	Dart	https://dart.dev
Backend Runtime	Node.js	https://nodejs.org
Backend Language	TypeScript	https://typescriptlang.org
Backend Framework	NestJS	https://nestjs.com
Backend Core Framework	@nestjs/core	https://npmjs.com/package/@nestjs/core
Backend Common Utilities	@nestjs/common	https://npmjs.com/package/@nestjs/common
Backend HTTP Platform	@nestjs/platform-express	https://npmjs.com/package/@nestjs/platform-express
Backend ORM	Prisma	https://prisma.io
Backend Database Client	@prisma/client	https://npmjs.com/package/@prisma/client
Backend WebSocket Support	@nestjs/websockets	https://npmjs.com/package/@nestjs/websockets
Backend Socket.IO Platform	@nestjs/platform-socket.io	https://npmjs.com/package/@nestjs/platform-socket.io
Backend Real-time Server	socket.io	https://npmjs.com/package/socket.io
Backend JWT Authentication	@nestjs/jwt	https://npmjs.com/package/@nestjs/jwt
Backend Supabase Integration	@supabase/supabase-js	https://npmjs.com/package/@supabase/supabase-js
Backend Push Notification	firebase-admin	https://npmjs.com/package/firebase-admin
Backend Email Service	nodemailer	https://npmjs.com/package/nodemailer
Backend Scheduled Tasks	@nestjs/schedule	https://npmjs.com/package/@nestjs/schedule
Backend API Documentation	@nestjs/swagger	https://npmjs.com/package/@nestjs/swagger
Backend File Upload	multer	https://npmjs.com/package/multer
Admin Font	Inter via Google Fonts	https://fonts.google.com

5.2. Demonstration Program Results

This section presents the outcomes achieved following the implementation of the Taskera system across both the mobile application and the web-based administrative interface. The current iteration of the system successfully realizes the core business workflows, including user account registration, service

Troy University Resources

scheduling, worker dispatching, and real-time task status tracking. These fundamental operations have been demonstrated to function stably within the defined scope of the project. The implementation validates the feasibility of the proposed architecture in handling essential home-service management tasks. However, it is important to note certain limitations inherent to the academic nature of this project. Specifically, the system has not yet been fully integrated with real-world payment gateways for financial transactions. Furthermore, advanced algorithms for optimizing travel routes and minimizing transit times for service workers remain unimplemented. These areas represent opportunities for future development and refinement before the system can be considered viable for deployment in a commercial production environment.
The statistics interface on the Admin web dashboard provides an overview of the operational status of the Taskera system. The charts visualize data such as the number of service requests, task completion rate, and revenue over different time periods. This function supports administrators in monitoring system performance, evaluating operational efficiency, and making appropriate management decisions.

5.2.1. System Packaging structure

The application is developed as a management system consisting of four main packaged components: Backend, Frontend Admin, Mobile Client (Flutter App), Database & Services.

5.2.2. Packaged Product 1: Backend (Server)

- Components:

  Includes REST APIs and WebSockets built with NestJS (TypeScript), handling core business logic for functions such as user authentication and registration, role-based access control (RBAC), customer/provider profiles, service category schemas, matching and dispatching workflows, job tracking, status history logs, reviews, chat messaging, and push notifications. The backend interacts with the database through Prisma ORM (with PostgreSQL adapter) and secures connection using Passport JWT, bcryptjs, Helmet, and Throttler rate limiting. It integrates with Firebase Admin SDK for FCM push notifications, Supabase for file uploads, and Nodemailer for SMTP email alerts.

- Role and Purpose:

  The backend is the core processing engine of the system. It receives and processes all HTTP REST requests and Socket.IO real-time events from the mobile and web clients. It validates input payloads, coordinates the automatic provider matching/dispatching algorithm, manages service transaction workflows, handles real-time message broadcasting, and ensures system security and data integrity.

- Main Technologies:

  NestJS, TypeScript, Prisma ORM, Socket.IO, Firebase Admin SDK, Nodemailer, Passport JWT, bcryptjs, PostgreSQL.

5.2.3. Packaged Product 2: Frontend Admin

- Components:

  The administrator user interface is built as a clean, lightweight web app using pure HTML5, CSS3 (using Google Fonts Inter and a custom styling sheet), and Vanilla JavaScript (ES6). It communicates with the NestJS API endpoints using the browser's native Fetch API. It contains functional views for statistics dashboards, provider approval management (viewing front/back documents, approving, rejecting, suspending), user accounts management, service category form settings, and active job/request auditing.

- Role and Purpose:

  The admin panel is the administrative control layer. It allows system operators to verify provider credentials, review and suspend accounts, monitor active jobs and system metrics, configure system fees, and manually intervene in matching requests when needed.

- Main Technologies:

  HTML5, CSS3, ES6 Vanilla JavaScript.

5.2.4. Packaged Product 3: Mobile Client (Flutter App)

- Components:

  The client user interface is built using the Flutter SDK and Dart programming language, targeting Android and iOS mobile platforms. It integrates a dual-portal design for:

- Customers: Onboarding, map-based address selection, dynamic category forms, finding matching providers, job status tracking, real-time messaging, review submissions, and history.
- Providers: Document upload onboarding, availability status toggle, skill categories and service area radius settings, pending job invitation boards, job navigation and location tracking, progress logging, and final price negotiation.
- Role and Purpose:

  The mobile application is the primary channel for end-users, enabling real-time interactions, live location updates using Geolocator/Google Maps, and instant notifications via WebSockets (Socket.IO client) and FCM push notifications.

- Main Technologies:

  Flutter SDK, Dart, Google Maps Flutter, Geolocator, Socket.IO Client, Firebase Core & Messaging.

5.2.5. Packaged Product 4: Database & Services

- Components:
  - PostgreSQL Database: containerized using Docker for development and deployed via Supabase in production. Stores relational tables for users, profiles, service categories, requests, jobs, logs, notifications, and device tokens.
  - Cloud Storage: Supabase Storage is used to store media files such as avatars, request photos, and provider certificates.
  - Redis Cache: used for caching and session management.
- Role and Purpose:

  Provides the long-term data persistence layer, ensuring transaction consistency and referential integrity, while the object storage manages high-availability media assets.

- Main Technologies:

  PostgreSQL, Supabase Storage, Redis.

Troy University Resources

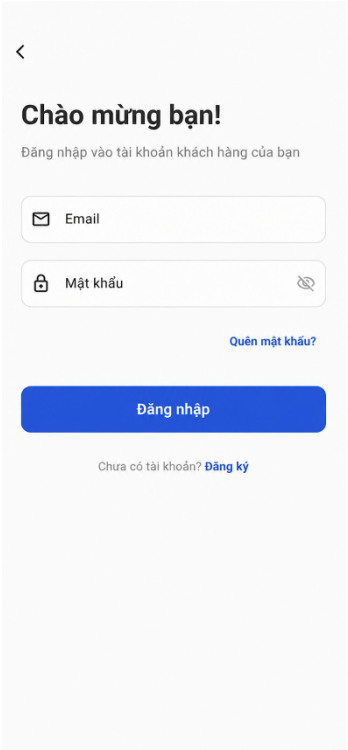


Figure 5-1: Login Page Interface

5.3. User Interfaces Illustrating Program Functions

The images show the service booking screen for customers on the mobile application. This interface allows users to select a service type, enter an address, choose the desired time, and add extra notes for the worker. After all valid information is provided, the system records the request and changes its status to waiting for dispatch.

1. Login page interface

Description:

- Back button to return to the previous screen.
- Login form with email and password fields.
- Button to show or hide the password using the eye icon.
- “Forgot password?” link for users who need to recover their password.
- Login button performs authentication.
- “Register” link for users who do not have an account.

1. Account Registration

For Customer:

Description:

- Back button to return to the previous screen.
- Registration form with full name, email, phone number, password, and confirm password fields.
- Buttons to show or hide the password and confirm password using the eye icons.
- Checkbox for agreeing to the Terms of Use and Privacy Policy.
- Terms of Use and Privacy Policy links allow users to view legal information before registering.
- Register button creates a new user account.

For Provider:

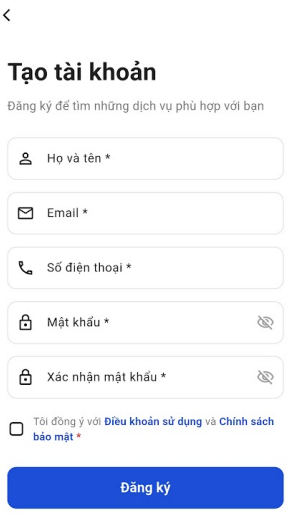


Figure 5-2: Register Page Interface for Customer

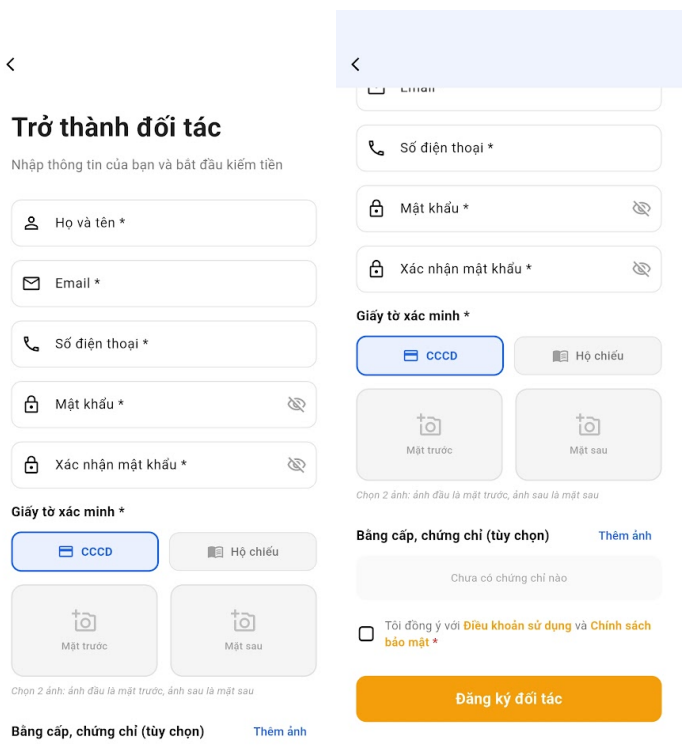


Figure 5-3: Register Page Interface for Provider

Troy University Resources

Description:

- Back button to return to the previous screen.
- Partner registration form with full name, email, phone number, password, and confirm password fields.
- Buttons to show or hide the password and confirm password using the eye icons.
- Identity document section allows users to choose between Citizen ID card and Passport.
- Image upload boxes allow users to upload the front and back sides of the identity document.
- Optional certificates section allows users to add qualifications or certificate images.
- Empty certificate state shows when no certificate image has been uploaded.
- Checkbox for agreeing to the Terms of Use and Privacy Policy.
  - Partner registration button submits the form to create a provider/partner account.

1. Provider’s main page interface

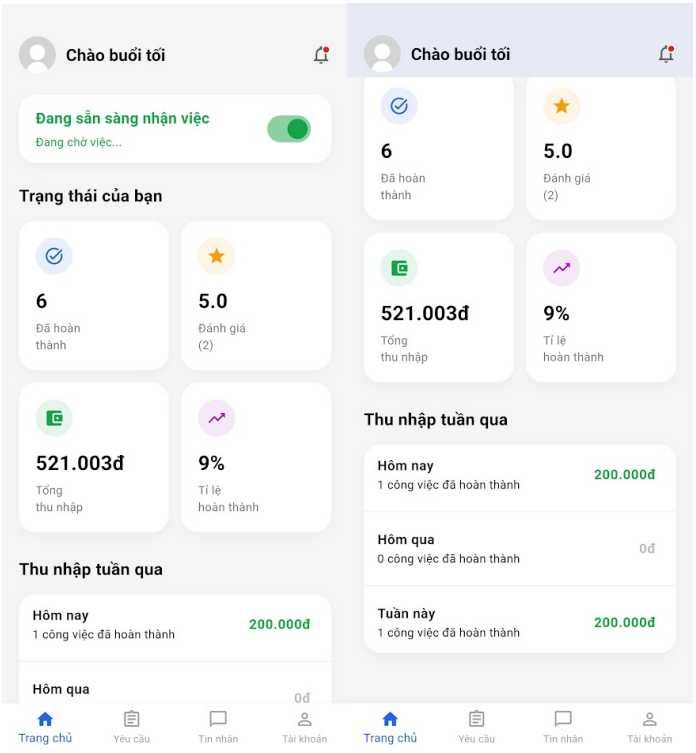


Figure 5-4: Provider Home Page Interface

Description:

- Notification icon shows new alerts or updates.
- Availability status card allows the provider to switch between available and unavailable for receiving jobs.
- Shows provider performance statistics, including completed jobs, rating, total income, and completion rate.

Description:

- Weekly income section displays income summary for today, yesterday, and the current week.
- Bottom navigation bar allows users to switch between Home, Requests, Messages, and Account pages.

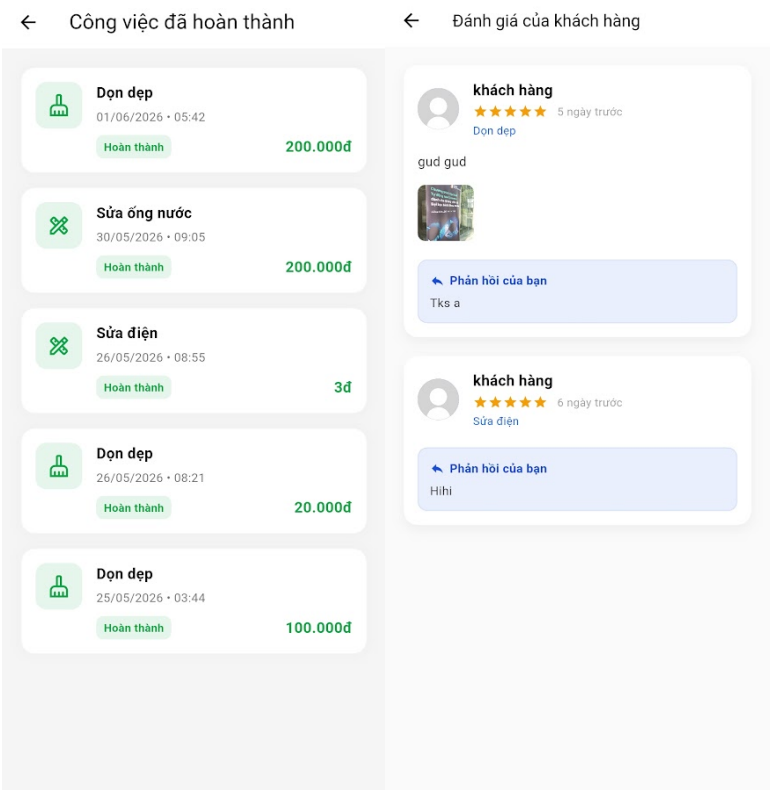


Figure 5-5: Completed Jobs and Rating



Figure 5-6: Provider's Profile Interface

Troy University Resources

1. Provider’s Profile interface

Description:

- Back button to return to the previous screen.
- Update Profile option allows the provider to edit personal information.

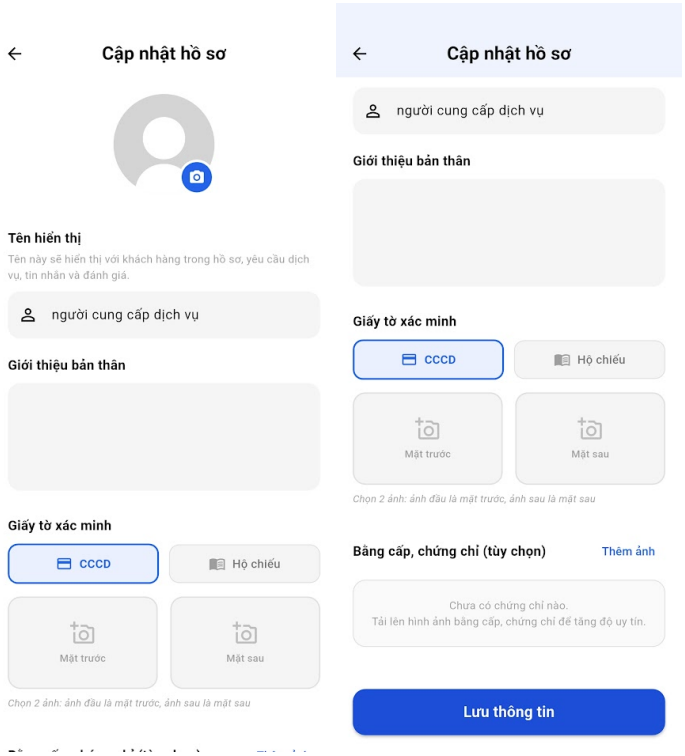


Figure 5-7: Provider's Update Profile Interface

Description:

- Manage Skills option allows the provider to add, edit, or manage service skills.

Description:

- Area Settings option allows the provider to configure working/service areas.

Description:

- Notification Settings option allows the provider to manage notification preferences.

Description:

- Logout button allows the provider to sign out of the account.

1. Provider’s feature interfaces

Description:

- Popup appears when a new job request is available for the provider.
- Title informs the provider that there is a new job.

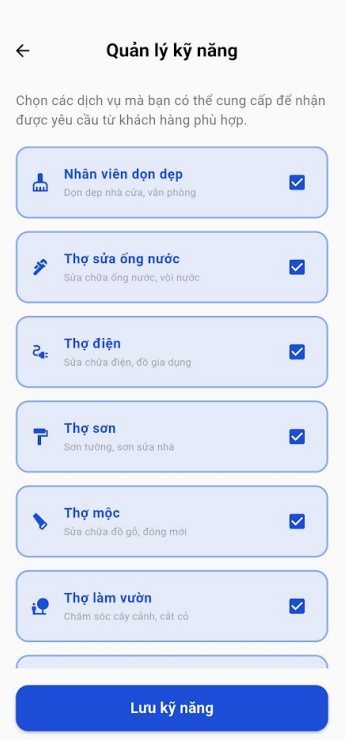


Figure 5-8: Provider's Manage Skills Interface

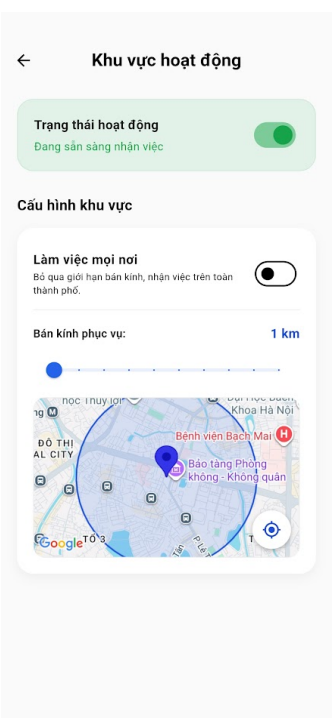


Figure 5-9: Provider's Area Settings Interface

Troy University Resources

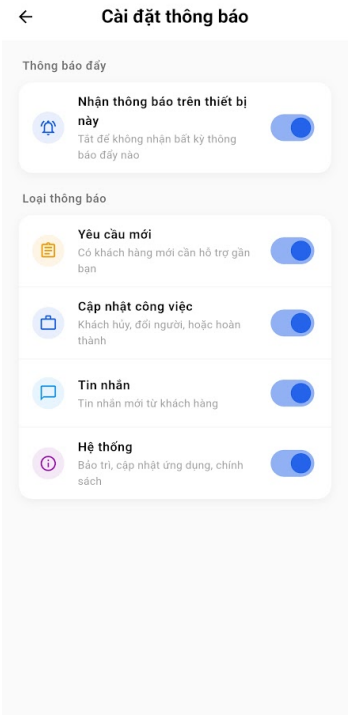


Figure 5-10: Provider's Notification Settings Interface

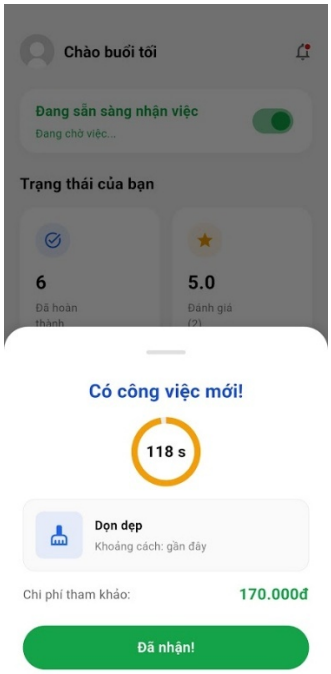


Figure 5-11: New Job Popup Interface

- Countdown timer shows the remaining time to accept the job request.
- Job card displays the service type, such as Cleaning.
- Distance information shows how close the job location is to the provider.
- Estimated cost section displays the reference service price.
- Accept button allows the provider to receive the new job request.

When taking the job and completing it:

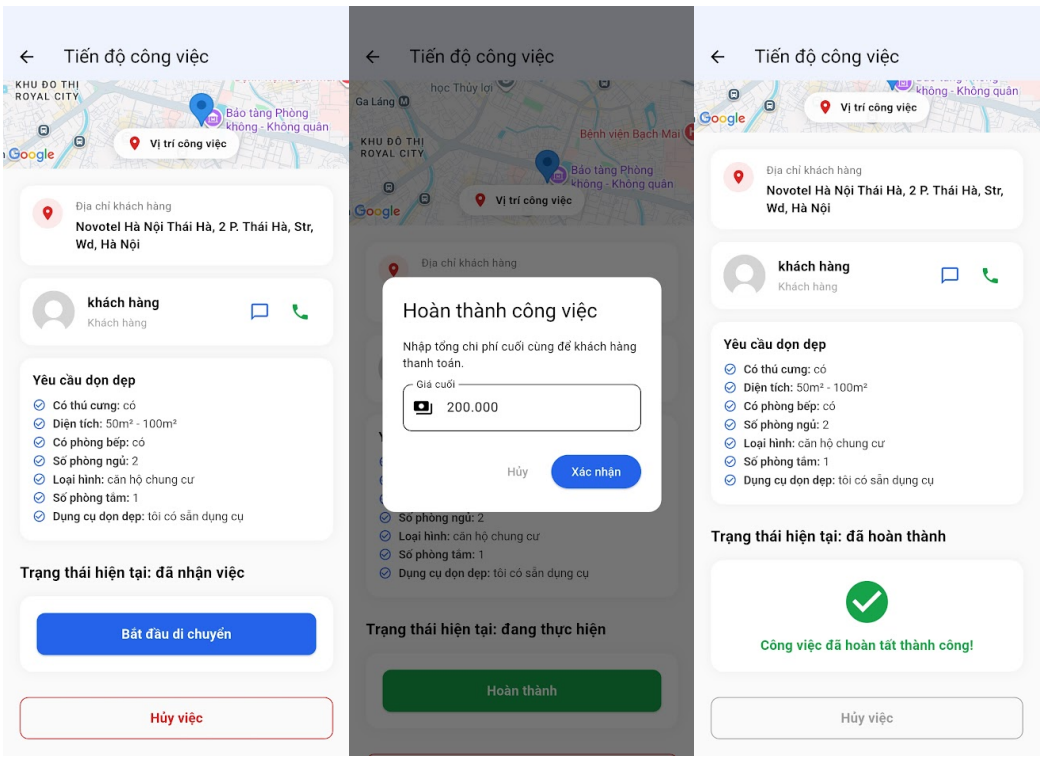


Figure 5-12: Job Progress Page Interface

Description:

- Back button to return to the previous screen.
- Page title shows the provider is viewing the current job progress.
- Map section displays the job location with a location marker.
- Customer address card shows the service location details.
- Customer information card displays the customer avatar, name, and role.
- Message icons allow the provider to chat with the customer.
- Phone icon allows the provider to call the customer.
- Request Details section shows the cleaning job requirements, including tools, area, kitchen cleaning, number of bedrooms, space type, number of bathrooms, and available cleaning equipment.
- Current Status section shows the job state, such as accepted, in progress, or completed.
- Start Moving button updates the job status when the provider begins traveling to the customer location.
- Complete button allows the provider to finish the job after the service is done.
- Completion popup asks the provider to enter the final total cost before confirming completion.
- Confirm button submits the final cost and marks the job as completed.
- Success message confirms that the job has been completed successfully.
- Cancel Job button allows the provider to cancel the job before completion.

1. Real-time chat

Troy University Resources

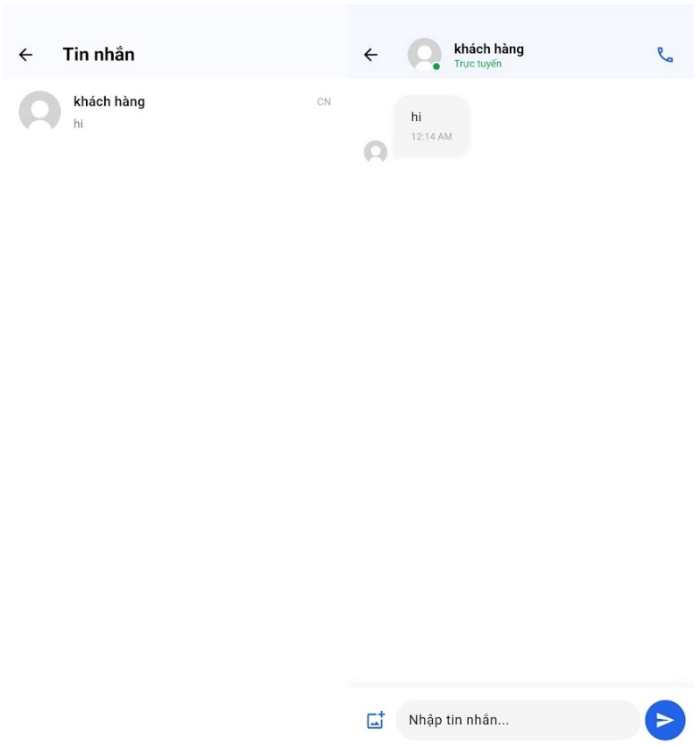


Figure 5-13: Real-time Chat Page Interface

Description:

- Back button to return to the previous screen.
- Message list page displays conversations with customers or providers.
- Conversation item shows the user avatar, name, latest message preview, and message time.
- Selecting a conversation opens the chat detail page.
- Chat detail header shows the receiver's avatar, name, and online status.
- Phone icons allow the user to call the other person.
- Chat area displays sent and received messages with message time.
- Image attachment button allows the user to send an image in the conversation.
- Message input field allows the user to type a new message.
- Send button sends the typed message to the receiver.

1. Customer's main page interface

Description:

- Notification icon shows alerts or new updates.

Description:

- Recent section shows recently viewed or contacted service providers.
- Provider card displays provider avatar, name, service type, and rating.
- View Profile button allows the user to open the provider's profile details.

Description:

- Service Categories section displays available service types such as cleaning, plumbing, electrical repair, painting, furniture repair, and gardening.
- "View all" link allows users to see the full list of recent providers, service categories, or top partners.
- Best Partners section recommends highly rated service providers.
- Create Service Request button allows users to start a new service booking request.

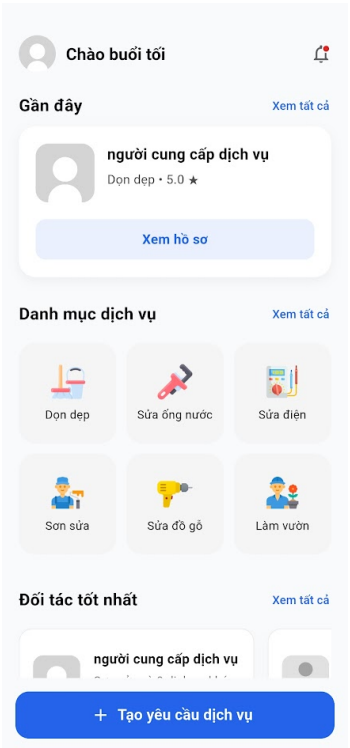


Figure 5-14: Customer's Main Page Interface



Figure 5-15: Customer's Notification Interface

Troy University Resources

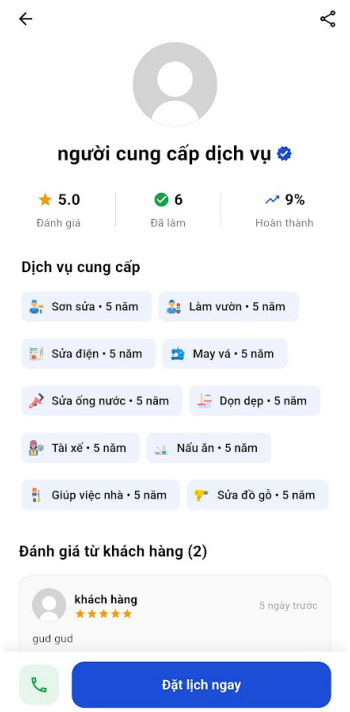


Figure 5-16: Customer's View Profile Interface

1. Customer’s Profile interface

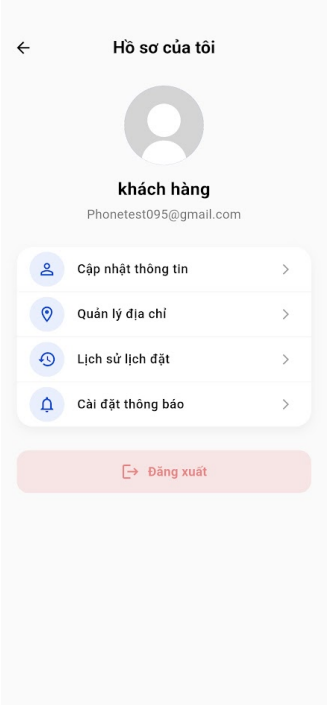


Figure 5-17: Customer's Profile Interface

Description:

- Back button to return to the previous screen.
- Update Information option allows the customer to edit account details.

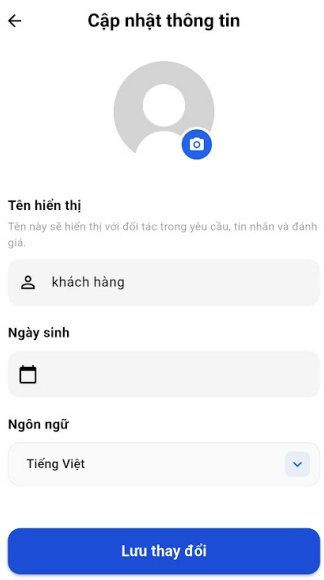


Figure 5-18: Customer's Update Information Interface

Description:

- Manage Address option allows the customer to add, edit, or remove saved addresses.

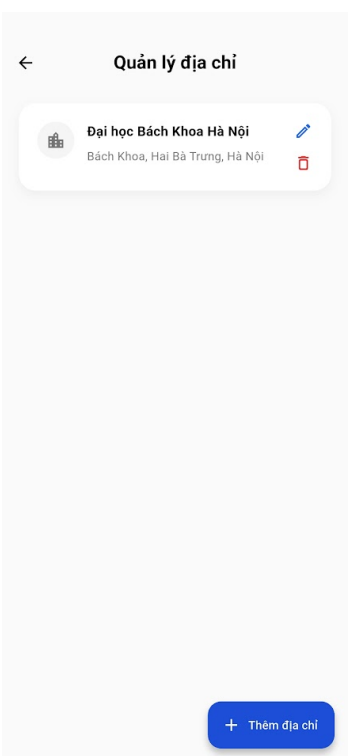


Figure 5-19: Customer's Manage Address Interface

Description:

- Booking History option allows the customer to view previous service bookings.

Troy University Resources

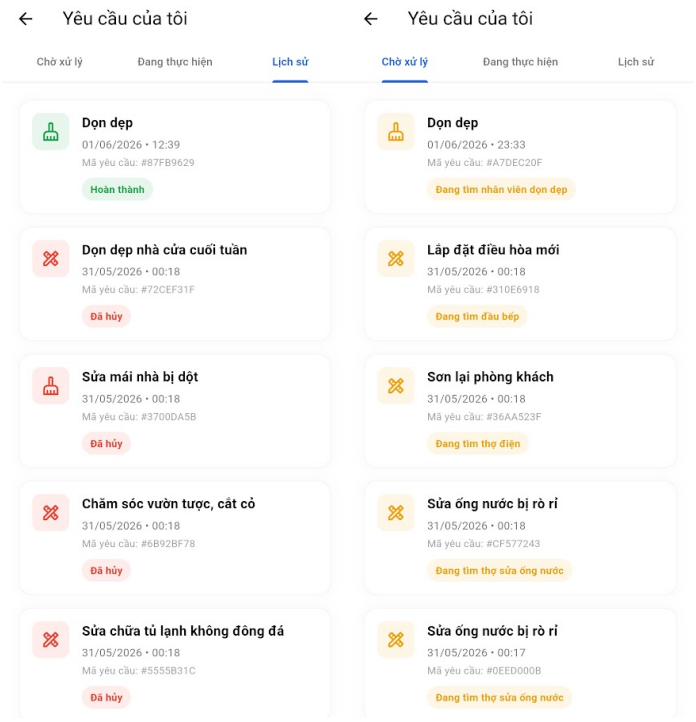


Figure 5-20: Customer's Booking History Interface

Description:

- Notification Settings option allows the customer to manage notification preferences.

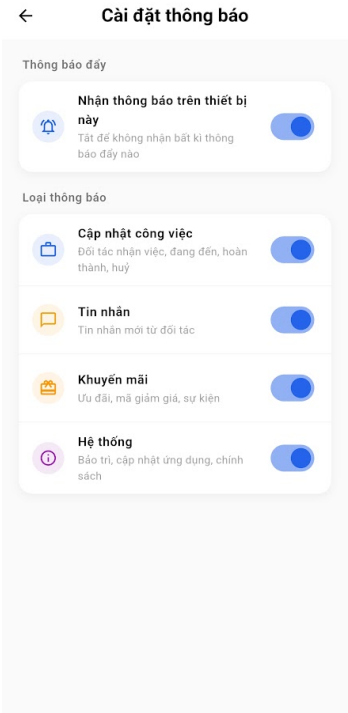


Figure 5-21: Customer's Notification Settings Interface

Description:

- Logout button allows the customer to sign out of the account.

1. Customer’s finding interface

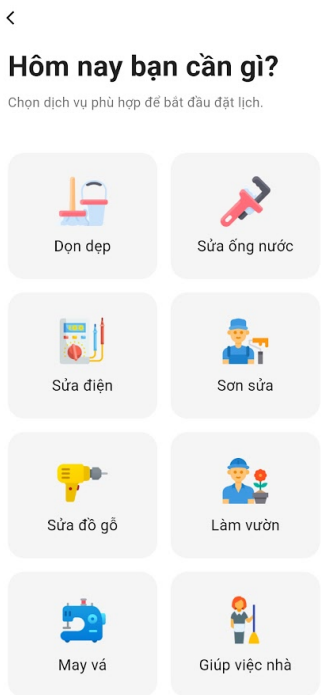


Figure 5-22: Customer's Job Finding Interface

Description:

- Back button to return to the previous screen.
- Page title asks the customer what service they need today.
- Subtitle guides the customer to choose a suitable service to start booking.
- Service category grid displays available service options.
- Each service card includes an icon and service name for easy recognition.
- Available categories include cleaning, plumbing repair, electrical repair, painting, furniture repair, gardening, sewing, and housework help.
- Selecting a service category opens the corresponding service request form.

When job is selected:

Description:

- Back button to return to the previous screen.
- Page title shows the selected service category as Cleaning.
- Form asks the customer what type of space needs to be cleaned.
- Space type options include apartment, house, villa, and office.
- Area dropdown allows the customer to select the size range of the space.
- Bedroom counter allows the customer to increase or decrease the number of bedrooms.
- Bathroom counter allows the customer to increase or decrease the number of bathrooms.
- Kitchen cleaning option allows the customer to choose whether kitchen cleaning is required.
- Pet option allows the customer to indicate whether there are pets in the space.
- Cleaning tools section allows the customer to choose whether tools are already available, whether the worker should bring tools, or whether the customer is unsure.
- Required fields are marked with an asterisk.
- Continue button moves the customer to the next step of creating the cleaning service request.

Troy University Resources

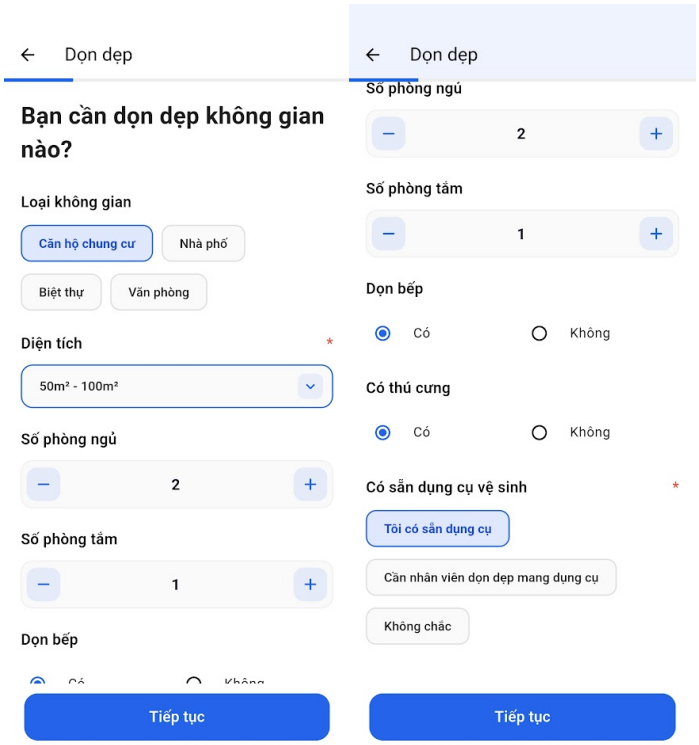


Figure 5-23: Customer's Service Request Form Interface

Specific Description for the job:

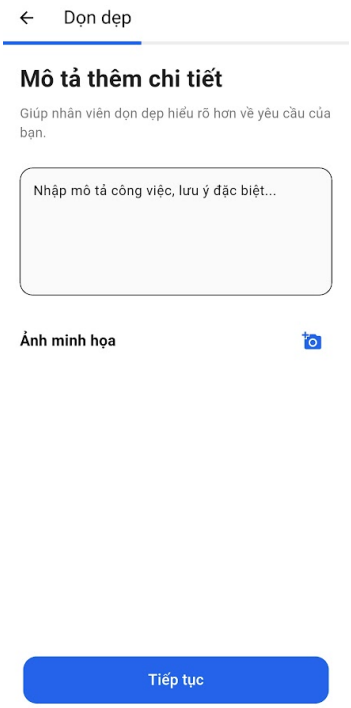


Figure 5-24: Customer’s Cleaning Service Detail Description Interface

Description:

- Back button to return to the previous screen.
- Page title shows the selected service category as Cleaning.

- Text input box allows users to enter job description, notes, or special requirements.
- Illustration image section allows the customer to upload supporting images for the service request.
- Camera icon button opens the image upload or camera selection function.
- Continue button moves the customer to the next step of creating the cleaning service request.

Desired Schedule for the job:



Figure 5-25: Customer's Cleaning Service Schedule Selection Interface

Description:

- Back button to return to the previous screen.
- Desired Time section allows the customer to choose when the service should be performed.
- Desired Date options allow the customer to select Today, Tomorrow, or choose a custom date.
- Time Slot section allows the customer to select an available working time range.
- Estimated Duration section allows the customer to choose the expected service duration, such as 2 hours, 3–4 hours, half day, or full day.
- Recurring Need section allows the customer to choose whether the service is one-time, weekly, or monthly.
- Continue button moves the customer to the next step of creating the cleaning service request.

Confirm and find Providers:

Description:

- Back button to return to the previous screen.
- Confirmation section summarizes the selected cleaning service information.
- Service card displays the cleaning service name and additional description.
- Desired Time card shows the selected date and time slot for the service.
- Address card displays the service location selected by the customer.
- Request Details section shows the selected cleaning requirements, including space type, area, number of bedrooms, number of bathrooms, kitchen cleaning, pet availability, and cleaning tools availability.

Troy University Resources

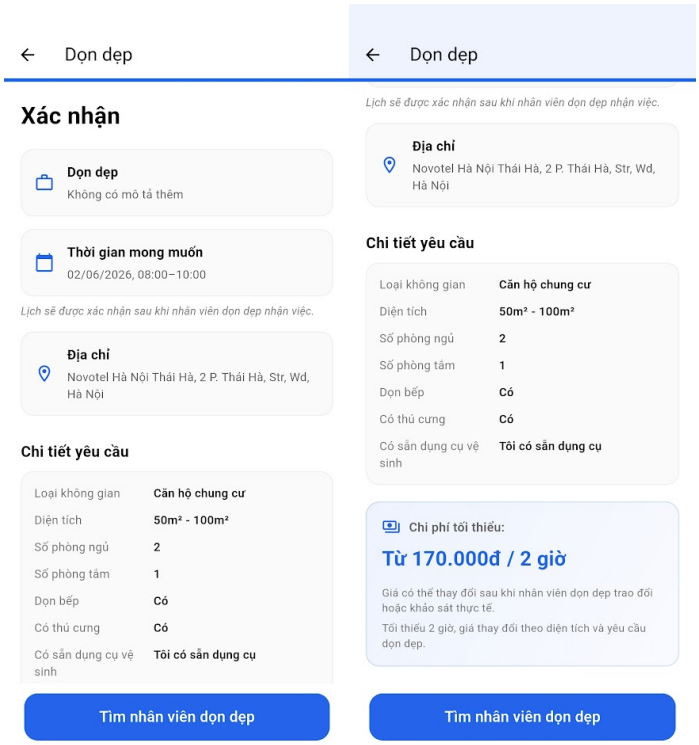


Figure 5-26: Customer's Cleaning Service Confirmation Interface

- Minimum Cost card displays the estimated starting price and minimum service duration.
- Note text informs the customer that the final price may change after discussion or real-site inspection.
- Find Cleaning Staff button submits the request and starts searching for an available cleaning provider.

CHAPTER 6. PROGRAM TESTING

6.1. Testing approach

The testing phase was conducted to verify whether the implemented prototype of Taskera satisfies the main functional and non-functional requirements. Since Taskera is developed as a course project demonstration, testing mainly focuses on manual black-box test cases derived from the system use cases. The tests check whether users can complete important workflows through the application interface without examining the internal source code.
Each test case is described by its test case ID, tested function, test objective, input condition, expected result, actual result, and status. The main roles considered in testing are Customer, Provider, and Admin. External services such as Supabase database, Firebase Cloud Messaging, and Socket.IO-based chat are also considered when they directly affect a tested feature.

6.2. Testing environment

Item	Description
Mobile application	Flutter / Dart application running on Android emulator or Android physical device
Backend service	REST API server used by the mobile application for authentication, booking, job, chat, and admin operations
Database	Supabase PostgreSQL database for storing users, provider profiles, service requests, jobs, messages, and admin logs
Real-time communication	Socket.IO client/server connection for job chat and room-based message delivery
Notification service	Firebase Cloud Messaging for push notifications related to job status and chat events
Testing method	Manual black-box testing based on use cases and visible UI behavior

6.3. Functional testing

Functional testing verifies that the implemented features work according to their intended use cases. The selected test cases focus on the core business flow of Taskera: customer authentication, service request creation, provider dispatching, job tracking, provider switching, real-time chat, and admin management.

6.3.1. Test cases for customer features

Test case ID	Function	Objective and input condition	Expected and actual result	Status
TC-CUS-01	Register account	Objective: Verify that a new customer can create an account. Input/condition: Customer enters valid name, phone/email, password, and OTP information.	Expected result: A customer account is created and the user can continue to the customer interface. Actual result: Same as expected.	Passed
TC-CUS-02	Login	Objective: Verify that a registered customer can log in successfully. Input/condition: Customer enters valid credentials.	Expected result: The system authenticates the account and op ens the customer home screen. Actual result: Same as expected.	Passed

Troy University Resources

Test case ID	Function	Objective and input condition	Expected and actual result	Status
TC-CUS-03	Create service request	Objective: Verify that a customer can submit a home-service booking request. Input/condition: Customer selects service category, enters address, description, images, and schedule information.	Expected result: A new service request is created and shown in dispatching or waiting status. Actual result: Same as expected.	Passed
TC-CUS-04	Track job status	Objective: Verify that the customer can follow the assigned job progress. Input/condition: Provider updates status to on the way, in progress, and completed.	Expected result: The customer tracking screen displays the latest job status and related information. Actual result: Same as expected.	Passed
TC-CUS-05	Switch provider	Objective: Verify that the customer can request another provider when needed. Input/condition: Customer selects switch provider on an assigned job.	Expected result: The current job is cancelled without penalty, the request returns to dispatching, and the previous provider is excluded from the next offer. Actual result: Same as expected.	Passed

6.3.2. Test cases for provider features

Test case ID	Function	Objective and input condition	Expected and actual result	Status
TC-PRO-01	Provider status	Objective: Verify that a provider can switch between online and offline states. Input/condition: Provider opens dashboard/settings and changes online status.	Expected result: The status is updated on the server and reflected correctly in the provider interface. Actual result: Same as expected.	Passed
TC-PRO-02	Receive job offer	Objective: Verify that an online provider can receive a matched job offer. Input/condition: A customer request is created while the provider is online and eligible.	Expected result: The provider receives a job offer dialog with service and customer information. Actual result: Same as expected.	Passed
TC-PRO-03	Accept job	Objective: Verify that the provider can accept an offered job. Input/condition: Provider presses accept on a valid job offer.	Expected result: The job is assigned to the provider and the customer can see assigned provider information. Actual result: Same as expected.	Passed

Test case ID	Function	Objective and input condition	Expected and actual result	Status
TC-PRO-04	Update job status	Objective: Verify that provider status updates change the job tracking flow. Input/condition: Provider updates job to on the way, in progress, and completed.	Expected result: The job status is saved and synchronized with the customer tracking screen. Actual result: Same as expected.	Passed

6.3.3. Test cases for chat and admin features

Test case ID	Function	Objective and input condition	Expected and actual result	Status
TC-CHAT-01	Real-time chat	Objective: Verify that customer and provider can exchange job-related messages. Input/condition: Both users open the chat room of the same job.	Expected result: Messages are sent, stored, and displayed in the correct conversation. Actual result: Same as expected.	Passed
TC-CHAT-02	Chat notification handling	Objective: Verify that chat notifications do not duplicate while user is already viewing the chat screen. Input/condition: User receives a message while the same chat room is active.	Expected result: The message appears in chat without unnecessary popup notification for the active room. Actual result: Same as expected.	Passed
TC-ADM-01	Provider approval	Objective: Verify that admin can review and approve provider profiles. Input/condition: Admin opens provider approval screen and checks CCCD/certificates.	Expected result: The provider approval status is updated and reflected in the provider account. Actual result: Same as expected.	Passed
TC-ADM-02	Service category management	Objective: Verify that admin can manage service categories used in booking. Input/condition: Admin creates, updates, or disables a service category.	Expected result: The service category data is stored and the customer booking flow uses the updated category list. Actual result: Same as expected.	Passed
TC-ADM-03	Admin action log	Objective: Verify that administrative actions can be recorded for later review. Input/condition: Admin performs an action that targets a user, provider profile, service request, job, or service category.	Expected result: The system stores an admin action log entry with action type, target type, target ID, actor, and timestamp. Actual result: Same as expected.	Passed

Troy University Resources

6.4. Non-functional testing

Non-functional testing evaluates general quality requirements that affect the usability, reliability, performance, security, and compatibility of the Taskera prototype. These tests are performed through observation during normal application usage.

Test case ID	Function	Objective and input condition	Expected and actual result	Status
NFT-01	Usability	Objective: Verify that the main screens are understandable for each user role. Input/condition: Customer, provider, and admin screens are tested through normal navigation.	Expected result: Customers, providers, and admins can identify their main actions without needing complex instructions. Actual result: Main actions are visible and separated by role.	Passed
NFT-02	Reliability	Objective: Verify that data remains consistent after important operations. Input/condition: Booking creation, provider assignment, job status update, and chat messages are performed.	Expected result: Important data is saved correctly and remains available after refresh or screen reopening. Actual result: Data remains available after refresh or screen reopening.	Passed
NFT-03	Performance	Objective: Verify that common operations respond within an acceptable time under normal network conditions. Input/condition: Login, booking creation, job loading, and chat message sending are executed.	Expected result: Common operations complete without noticeable delay. Actual result: The prototype responds acceptably in normal testing conditions.	Passed
NFT-04	Security and access control	Objective: Verify that users cannot use functions outside their roles. Input/condition: Customer, provider, and admin accounts are used to access role-based screens.	Expected result: Role-based screens prevent normal users from accessing unrelated functions. Actual result: Role-based navigation prevents access to unrelated functions.	Passed
NFT-05	Compatibility	Objective: Verify that the application can run on the selected Android testing environment. Input/condition: The Flutter application is installed and tested on Android emulator or physical Android device.	Expected result: Core screens and workflows display correctly on the tested device. Actual result: Same as expected.	Passed

6.5. Testing conclusion

The testing results show that the main implemented functions of Taskera operate correctly in normal scenarios. Core workflows such as account authentication, service request creation, provider job acceptance, job tracking, provider switching, chat, provider approval, service category management, and admin action logging were verified through manual black-box test cases.

Because the current version is a prototype for a course project, the testing scope is still limited. The report mainly focuses on functional testing and basic non-functional testing. Large-scale automated testing, stress testing, and full security penetration testing can be added in future development to improve reliability and maintainability.

CHAPTER 7. INSTALLATION AND USER GUIDE

7.1. Installation Guide

This section outlines the fundamental procedures required for the installation and deployment of the system within a local development environment. The comprehensive guide details the essential software prerequisites and dependencies necessary to initialize the project. It systematically covers the configuration of the PostgreSQL database, the setup and initialization of the backend Application Programming Interface (API) utilizing the NestJS framework, and the compilation and execution of the mobile application built with Flutter. Furthermore, it addresses the configuration of environment variables and network settings required to establish seamless communication between the client and server components. Strict adherence to these documented deployment steps is crucial for establishing a stable and reproducible operational environment. Ensuring that the system is correctly configured at this stage is a prerequisite for conducting rigorous functional and integration testing, thereby validating the system's operational capabilities before any potential transition to a production environment.

7.2. Users and Scope Of Use

The Taskera system serves three main groups of actors: customers, service workers, and administrators. The scope of the system focuses on digitalizing the process of service booking, task dispatching, status tracking, and home-service quality evaluation. Within the limitations of this major assignment, the system supports the core business flows but does not yet include complex online payment features or optimized automatic dispatching algorithms.

7.3. Installation Requirements

To deploy and operate the Taskera system, the server environment needs to have Node.js, the PostgreSQL database management system, and Redis installed to support real-time processing. For the mobile application development environment, Flutter SDK and the corresponding emulation tools, such as Android Studio or Xcode, are required. In addition, the developer’s computer should meet the minimum hardware requirement of 8GB RAM or higher to ensure stable compilation and testing.

7.4. Detailed Installation Steps

To deploy the Taskera system in the development environment, the installation process is divided into two main parts: Backend and Mobile App. For the Backend, developers need to install Node.js, set up the PostgreSQL database, configure environment variables, and run the Prisma initialization command. For the mobile application, the environment requires the Flutter SDK, an Android or iOS emulator, and then the build command can be executed so that the application starts and connects to the system API.

7.5. Software User Guide

The software user guide provides detailed, step-by-step operational instructions designed to enable various actors to interact effectively with the Taskera system. For customers and service providers, the documentation primarily focuses on the installation of the mobile application, the account registration process, and the execution of the core service request and acceptance of workflows. Conversely, for system administrators, the guide delineates the procedures for utilizing the web-based dashboard to review worker profiles, manage user accounts, and monitor platforma ctivities. By offering clear and structured guidance, this document aims to minimize the learning curve for end-users and ensure the correct utilization of the system functionalities. Furthermore, the inclusion of comprehensive usage instructions serves as a critical component of the software deployment phase, facilitating user adoption and providing a reference framework for troubleshooting common operational issues encountered during standard system interaction.

Troy University Resources

Conclusion and Future Development

After the process of analysis, design, and development, the team has completed the core functions of the Taskera system. The system can support the basic business flow of service booking and worker dispatching. However, the system still has some limitations, especially in handling large-scale user traffic and fully optimizing the worker's recommendation algorithm. In future versions, the system can be expanded with data analytics features and improvements to the overall user experience.

Regarding future development, the Taskera system possesses the potential for several strategic expansions to enhance its functionality and user experience. A primary objective for subsequent iterations is the integration of electronic payment gateways, which would diversify payment methods and facilitate secure, cashless transactions within the platform. Furthermore, future enhancements could focus on refining the worker dispatching algorithms. By incorporating real-time geolocation data and historical performance ratings, the system could more effectively match service providers with customer requests, thereby potentially reducing wait times and improving service quality. Additionally, the implementation of advanced data analytics mechanisms is recommended. Such analytical tools would enable system administrators to extract actionable insights from user behavior and transaction histories, facilitating more accurate forecasting of service demands. These proposed improvements would not only address current limitations but also contribute to the long-term scalability and adaptability of the platform in a dynamic market environment.

References

[1] I. Sommerville, Software Engineering, 9th ed. Boston, MA, USA: Addison-Wesley, 2011.
[2] R. S. Pressman, Software Engineering: A Practitioner's Approach, 7th ed. New York, NY, USA: McGraw-Hill Higher Education, 2010.
[3] Object Management Group, Unified Modeling Language, Version 2.5.1, formal specification, Dec. 2017. [Online]. Available: https://www.omg.org/spec/UML/2.5.1
[4] K. D. Moore, V. Ngo, S. Patterson, and A. Ulate Fallas, Flutter Apprentice: Learn to Build Cross-Platform Apps, 4th ed. Kodeco Inc., 2024. [Online]. Available: https://www.kodeco.com/books/flutter-apprentice/v4.0
[5] J. Sande, Dart Apprentice: Fundamentals: Modern Cross-Platform Programming With Dart. Kodeco Inc., 2022. [Online]. Available: https://www.kodeco.com/books/dart-apprentice-fundamentals/v1.0
[6] J. Sande, Dart Apprentice: Beyond the Basics: Object-Oriented Programming, Concurrency & More. Kodeco Inc., 2022. [Online]. Available: https://www.kodeco.com/books/dart-apprentice-beyond-the-basics/v1.0
[7] M. Jones, J. Bradley, and N. Sakimura, JSON Web Token (JWT), RFC 7519, Internet Engineering Task Force, May 2015. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7519

Troy University Resources