**P-05 Ujret Mobile Application**

**SPROJ Report**



**Malik M. Moaz (24100163)  
Abdul Moez (24100244)**

**Abdullah Ehsan (24100144)  
Talha Husnain (24100245)  
Jahanzaib Khursheed (24100257)**

**Advisor:   
Dr Waqar Ahmad  
Co-Advisor:  
Dr. Sohaib Ayub**

**School of Science and Engineering**  
**Lahore University of Management Sciences**  
**Submission Date: 1st May, 2024**

**Acknowledgement and Dedication**

**Certificate**

I certify that the senior project titled “**Ujret Mobile Application**” was completed under my supervision by the following students:

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

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

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

and the project deliverables meet the requirements of the program.

------------------------------------- Date:

**Advisor (Signature)**

------------------------------------- Date:

**Co-advisor (if any)**

# Table of Contents

[Table of Contents](#_heading=)

[List of Figures](#_heading=)

[1. Introduction](#_heading=)

[a. Introduction](#_heading=)

[b. Objective and Scope](#_heading=)

[c. Development Methodology](#_heading=)

[d. Contributions](#_heading=)

[2. System Requirements](#_heading=)

[a. System Actors](#_heading=)

[b. Functional Requirements](#_heading=)

[c. Non-functional Requirements](#_heading=)

[Non-functional Requirements / Quality Attributes](#_heading=h.mulorf322csg)

[3. System Architecture](#_heading=)

[a. Architecture Diagram](#_heading=h.gjdgxs)

[b. Architecture Description](#_heading=h.30j0zll)

[c. Justification of the Architecture](#_heading=h.1fob9te)

[d. Tools and Technologies](#_heading=)

[4. Requirements Specifications](#_heading=h.3znysh7)

[a. Use Cases](#_heading=)

[b. Class Diagram](#_heading=h.2et92p0)

[c. Sequence Diagrams](#_heading=h.tyjcwt)

[5. Software Development Methodology and Plan](#_heading=)

[a. Software Process Selection](#_heading=h.3dy6vkm)

[b. Gantt Chart](#_heading=h.1t3h5sf)

[6. Database Design and Web Services](#_heading=)

[a. Database Design](#_heading=)

[b. API Specification](#_heading=)

[7. System User Interface](#_heading=)

[8. Project Security](#_heading=)

[11. Risk Management](#_heading=)

[Potential Risks and Mitigation Strategies](#_heading=h.4d34og8)

[12. Testing and Evaluation](#_heading=)

[13. Deployment Guidelines](#_heading=)

[14. Conclusion](#_heading=)

[a. Challenges](#_heading=)

[b. Future](#_heading=)

[15. Review checklist](#_heading=h.2s8eyo1)

[16. References](#_heading=)

# Introduction

The Ujret Mobile App is a pioneering platform in the Consumer-to-Consumer (C2C) market that has been successfully developed to address distinct needs within the service and rental sectors. Originally planned with a Handyman and a Carpooling module, the final development focused exclusively on the Handyman services and introduced a new Renting module. This shift in development underscores the app’s flexibility and its commitment to meeting user demands and market trends. The application serves as a versatile tool where users can either offer or secure services and rentals efficiently.

**Domain of the Application**

Ujret Mobile App’s domain encompasses service provision and personal rental transactions, placing it firmly within the gig and sharing economy. By focusing on these two areas, the app facilitates essential daily interactions—be it through hiring a service or renting personal items. This dual-functionality positions the app as a comprehensive resource for a broad user base looking to optimize their everyday tasks and possessions.

**Target Users of the Application**

* Handyman Module:

Service Seekers: Includes individuals and businesses needing professional services such as home repairs, installations, and other skilled tasks.

Service Providers: Comprises professionals and tradespeople who offer their expertise to a local customer base, leveraging the app to expand their reach and streamline their operations.

* Renting Module:

Item Lenders: Users who own items that are infrequently used and wish to monetize these by offering them for rent.

Item Renters: Individuals looking for short-term use of specific items without the need to purchase them, benefiting from cost savings and convenience.

**Type of Application**

The Ujret Mobile App has been developed as a mobile application, ensuring high accessibility and usability across various mobile devices. This choice supports the dynamic nature of the services and rentals offered, allowing users to engage with the platform from anywhere at any time.

**Handyman Module Features:**

Interactive Booking System:

Service Seeker Experience: Service seekers start by selecting a service category, such as plumbing, electrical, carpentry, etc., upon logging in. They are prompted to fill out a detailed task post form that includes descriptions of the service needed. Upon submitting the form by clicking the "Schedule Task" button, the request becomes visible to all available service providers in the selected category.

Service Provider Experience: Service providers see incoming tasks in real time if they are online and can bid on tasks they are interested in handling. This approach keeps the process dynamic and responsive, relying on the availability and interest of providers at the time the task is posted.

Transparent Pricing:

Service providers submit bids that include their pricing for the task, allowing them to compete based on their rates and expertise. Service seekers can review these bids and select the provider that best meets their needs in terms of price and provider profile.

User Reviews and Ratings:

Following the completion of a task, both the service seeker and the provider have the opportunity to rate and review each other. This feedback is vital for maintaining a trustworthy community and aids all users in making informed choices based on the reviews and ratings of past interactions.

**Renting Module Features:**

Interactive Catalog of Items:

Lender Experience: Lenders can post items they wish to rent out, providing details such as item description, suggested rental price, and available rental periods. The platform allows them to manage their listings directly, updating availability or modifying information as needed to attract potential renters.

Renter Experience: Renters can easily browse through the app’s categorized catalog of rentable items, which includes a wide range of products from household tools to outdoor equipment. The search functionality is designed to help users find specific items based on their needs, ensuring a quick and efficient rental process.

Peer-to-Peer Connection:

Once renters find the items they need, they can directly connect with the lenders through the app. This direct line of communication facilitates the discussion of rental terms, availability, and any other pertinent details. Since the app does not handle payments, users are encouraged to arrange their payment terms privately, allowing for flexibility and personal interaction.

User Reviews and Ratings:

After the rental transaction concludes, both lenders and renters are prompted to review and rate each other. This feature is critical in building a trustworthy community; it provides valuable insights into the reliability and quality of both the renters and the items listed. Reviews help future users make informed decisions and maintain high standards within the community.

**User Experience and System Interaction**

The features of the Ujret Mobile App are crafted to ensure ease of use, transparency, and efficiency, catering to the needs of both service and rental market segments. By focusing on an intuitive booking system and robust review mechanisms, the app supports a vibrant community of users engaged in service provision and item rentals. The absence of an in-app payment system for rentals and the real-time bidding process for handyman services highlight the app's flexible approach, adapting to user preferences and operational requirements. This flexibility is integral to fostering user confidence and satisfaction, making Ujret a preferred platform in the sharing economy.

**Impact and Evaluation**

Since its launch, the Ujret Mobile App has significantly influenced how people access handyman services and engage in personal item rentals. The application's user-focused design ensures a seamless integration of service provision and item rental processes into daily life, making it a valuable tool for a wide user demographic.

Enhancements through Feedback:

* Continuous Improvement: User feedback has been instrumental in refining app features and functionalities, ensuring they meet the evolving needs of the community.
* Increased Engagement: The app’s ease of use and comprehensive features have led to high user engagement and retention rates.

The Ujret Mobile App exemplifies innovation and adaptability in the C2C market by effectively combining handyman services with a personal item rental module. As the app continues to evolve, it remains dedicated to enhancing user experiences and expanding its offerings to meet and anticipate the needs of its diverse user base. With its robust functionality and community-focused features, Ujret stands out as a key player in the sharing economy, fostering a culture of cooperation and mutual benefit among its users.

## Objective and Scope

**Objectives of the Application**

The primary objective of the Ujret Mobile App is to simplify and enhance the way individuals and businesses access handyman services and engage in personal item rentals within the Consumer-to-Consumer (C2C) marketplace. The app aims to achieve this by creating a reliable, user-friendly platform that connects service providers and seekers, as well as item lenders and renters, through an intuitive and efficient online system. Specific objectives include:

* Facilitate Efficient Transactions: Streamline the process of finding and booking handyman services, and renting personal items, making these transactions as swift and simple as possible.
* Expand Market Access: Provide service providers and item lenders with broader access to potential clients and customers, thereby increasing their opportunities for business and income.
* Enhance User Experience: Offer a seamless user experience through an intuitive interface and user-centered design that caters to both tech-savvy and less tech-inclined users.
* Build Community Trust: Establish a trusted community by incorporating robust review and rating systems that ensure transparency and accountability among all users.
* Promote Economic and Environmental Benefits: Encourage the sharing economy model which maximizes resource utilization and reduces waste, contributing to economic savings and environmental benefits.

**Reasons for Development**

The decision to develop the Ujret Mobile App was driven by a clear need in the market for a more cohesive and accessible platform that could address the following aspects:

* Lack of Centralized Services: Existing platforms were either too niche, focusing solely on one type of service or item, or too broad without the necessary depth in any one area, making it difficult for users to find a reliable one-stop solution.
* Inefficiency in Current Systems: Many traditional methods for securing handyman services and renting items are cumbersome and time-consuming, involving multiple steps and platforms.
* Growing Demand for Flexible Work Options: The gig economy is expanding, and with it, the number of individuals looking to leverage their skills and assets in a flexible, income-generating manner.

**Impact and Enhancement on Business Operations**

By addressing these needs, the Ujret Mobile App significantly impacts and enhances business operations in its domain.

* Operational Efficiency: Businesses and individuals can reduce the time and effort spent on finding service providers and renting items. For businesses, this means quicker turnaround times for maintenance issues and lower costs for equipment through rentals.
* Economic Impact: Service providers and item lenders can tap into a larger market, increasing their potential earnings. The app's efficient marketplace also helps users save money, which can be redirected into other areas of the economy.
* Environmental Sustainability: By facilitating item rentals, the app encourages users to share resources, thereby reducing the need to purchase new items. This contributes to waste reduction and lessens the environmental footprint associated with manufacturing and disposing of goods.
* Community Building: The platform fosters a sense of community by connecting like-minded users, encouraging interactions that can lead to ongoing business relationships and community strengthening.

In conclusion, the Ujret Mobile App is designed to modernize and optimize the way services and rentals are managed within the C2C sector, delivering substantial benefits to users and contributing positively to the broader economic and environmental landscape.

## Development Methodology

The development of the Ujret Mobile App was carried out using an Agile development methodology, particularly focusing on the principles of Scrum. This approach was chosen due to its flexibility, efficiency, and effectiveness in managing large-scale software projects with numerous variables and evolving requirements. Here is a brief outline of the methodology used:

**1. Planning and Requirement Analysis**

At the project's outset, the development team conducted extensive market research to identify the specific needs of potential users and stakeholders. This phase involved gathering requirements through surveys, focus groups, and one-on-one interviews with prospective service providers, renters, and service seekers. The gathered data helped in defining clear and actionable user stories and project scope, forming the foundation for all subsequent development efforts.

**2. Sprint Planning**

The project was divided into two-week sprints. Each sprint began with a planning meeting where the team prioritized the user stories and defined a sprint goal. Tasks were assigned based on the team's capacity and the complexity of the features to be developed. This iterative planning allowed the team to adapt the project scope based on feedback and testing outcomes from previous sprints.

**3. Daily Stand-ups**

Each day of the sprint, a short stand-up meeting was held to discuss progress and any obstacles that might impede the workflow. These meetings ensured that all team members were aligned on the project goals and could collaborate effectively to resolve any issues quickly.

**4. Development and Testing**

During each sprint, developers worked on coding the application while simultaneously conducting unit tests and integration tests. This overlap of development and testing ensured that any bugs or issues were identified and resolved early in the process. The Agile methodology's emphasis on testing at every stage helped in maintaining high quality and functionality of the application.

**5. Sprint Review and Retrospectives**

At the end of each sprint, the team conducted a review meeting to demonstrate the completed features to stakeholders and gather feedback. This feedback was crucial for adjusting the project trajectory and making necessary improvements. Following the review, a retrospective meeting was held to reflect on the sprint process and identify areas for process improvement.

**6. Release and Iteration**

After several sprints, once enough features were developed and tested, the application was released to a selected group of users for beta testing. The feedback from this phase was used to make final adjustments before a full public launch. Post-launch, the team continued to iterate on the product, adding new features and making improvements based on user feedback and emerging market trends.

This Agile Scrum methodology not only facilitated a responsive and adaptive development environment but also ensured that the final product was closely aligned with user needs and expectations. The collaborative and iterative approach helped the Ujret Mobile App to effectively meet its objectives and offer a robust platform for its users.

## Contributions

The development of the Ujret Mobile App has brought several significant contributions to the Consumer-to-Consumer (C2C) marketplace, particularly in the areas of service provision and personal item rentals. The app introduces innovative features and solutions that not only enhance user experience but also improve operational efficiency and community engagement. Here's a detailed look at these contributions and innovations:

**1. Integrated Dual-Functionality**

Unlike many platforms that focus exclusively on either service provision or item rentals, Ujret seamlessly integrates both functionalities within a single application. This dual-functionality allows users to switch between hiring services and renting items without the need to use multiple platforms, thus providing a comprehensive solution that caters to a broader range of user needs.

**2. Real-Time Bidding System for Services**

The Ujret app introduces a real-time bidding system for handyman services, where service providers can place bids on tasks as soon as they are posted. This dynamic marketplace enables service seekers to receive competitive bids quickly and makes the process of selecting a service provider transparent and efficient. This feature not only speeds up the hiring process but also empowers both parties to negotiate fair prices based on real-time market demand.

**3. User-Driven Transparency and Community Trust**

Both the Handyman and Renting modules incorporate a robust user review and rating system. This system ensures that all transactions are transparent and that both parties can trust each other. Feedback from previous interactions is openly available, helping users make informed decisions based on the reliability and quality of the service providers and item lenders.

**4. Flexibility in Payment and Transactions**

Recognizing the diverse needs and preferences of its users, Ujret does not restrict payment methods by excluding an in-app payment system for rentals. Instead, it allows users to arrange their financial transactions directly, offering flexibility and promoting personal interaction between lenders and renters. This approach caters to a global audience with varied access to payment methods and preferences.

**Comparative Advantage Over Similar Solutions**

When compared to other solutions in the market, the Ujret Mobile App stands out due to its user-centric design and innovative features:

* Enhanced Usability: The app's interface is designed to be intuitive and easy to navigate, reducing the learning curve for new users and enhancing the overall user experience.
* Greater Flexibility: The flexibility in scheduling tasks and managing rentals allows users to use the services at their convenience, adapting to their personal schedules and needs.
* Community Building: The emphasis on building a trusted community through a robust rating system and transparent reviews creates a safer and more reliable environment for transactions.
* Sustainable Impact: By promoting the sharing economy through its dual modules, Ujret contributes to environmental sustainability by reducing the need for new purchases and decreasing waste.

Overall, the Ujret Mobile App not only addresses the existing gaps in the C2C marketplace but does so with innovative solutions that prioritize efficiency, transparency, and user satisfaction. Its approach to integrating multiple services under one platform, coupled with its commitment to community and sustainability, makes it a superior choice for users looking to maximize their resources and capabilities.

# System Requirements

This chapter delves into the fundamental system requirements essential for the Ujret Mobile App, outlining the technical specifications, functionalities, and operational standards necessary for the platform's optimal performance. It serves as a blueprint for understanding the capabilities and constraints of the system, addressing both the handyman and renting modules. The requirements are categorized into hardware and software needs, data management protocols, security measures, user interface designs, and compliance with relevant regulations. This comprehensive overview ensures that stakeholders, including developers, system administrators, and end-users, have a clear understanding of what the application necessitates to function effectively and efficiently in the C2C marketplace. Through this exploration, we establish a roadmap for the current infrastructure and pave the way for future enhancements, ensuring the app remains robust, scalable, and user-centric.

## System Actors

| **Actor Name** | **Description** |
| --- | --- |
| Handyman Service Provider | Responsible for posting handyman jobs on the platform for handyman service seekers. |
| Handyman Service Seeker | A freelance operator who will be working on handyman jobs posted by a handyman service provider. |
| Item Lender | Posts items available for rent on the platform, providing descriptions, rental terms, and conditions. |
| Items Renter | Browses the platform for items to rent, initiates rental agreements, and communicates with lenders. |
| System Administrator | A user with full admin privileges across all our services. This user may create, read, update and delete data. |

## Functional Requirements

| **Requirements** | |
| --- | --- |
| **Sr#** | **Requirement** |
| 1.1 | As an admin, I want to edit user profiles to manage user information across handyman and renting modules. |
| 1.2 | As an admin, I want to delete user profiles to maintain a healthy and active user base efficiently. |
| 1.3 | As an admin, I want to provide chat support to users to assist them in both the handyman and renting modules when needed. |
| 2.1 | As a general user, I want to have control over my profile information, allowing me to edit, delete, or add details to ensure my information is up to date across all modules. |
| 2.2 | As a general user, I want to securely log in to the app with proper authentication and authorization. |
| 3.1 | As a handyman service seeker, I want to select a task category to find relevant services easily. |
| 3.2 | As a handyman service seeker, I want to post task details in real-time so providers can bid on my job. |
| 3.3 | As a handyman service seeker, I want to view bids from service providers in real-time based on my posted task. |
| 3.4 | As a handyman service seeker, I want to engage in real-time chat with service providers to discuss task specifics through WhatApp. |
| 3.5 | As a handyman service seeker, I want to view detailed profiles of service providers, including ratings and reviews, before making a selection. |
| 3.6 | As a handyman service seeker, I want to rate and review service providers after task completion to contribute to the community's trust. |
| 3.7 | As a handyman service seeker, I want to have a history of tasks/jobs I've ordered to keep track of my service engagements. |
| 4.1 | As a handyman service provider, I want to list my areas of expertise or specialization to attract relevant job postings. |
| 4.2 | As a handyman service provider, I want to bid on jobs in real-time to offer my services to seekers. |
| 4.3 | As a handyman service provider, I want to have a history of tasks/jobs I have completed to maintain a record of my work and performance. |
| 4.4 | As an item lender, I want to post items available for rent on the platform, providing descriptions and terms to attract renters. |
| 5.1 | As an item lender, I want to manage my rental postings, updating item availability and rental conditions as needed. |
| 5.2 | As an item lender, I want to receive notifications when users express interest in or inquire about my rental items. |
| 5.3 | As an item lender, I want to rate and review renters after the rental period to build trust within the community. |
| 5.4 | As an item renter, I want to browse the catalog for items to rent and view details like pricing, rental terms, and lender ratings. |
| 5.5 | As an item renter, I want to directly communicate with item lenders to finalize rental agreements and arrange logistics. |
| 6.1 | As an item renter, I want to rate and review item lenders after the rental period to share my experience with the community. |
| 6.2 | As a user engaged in services or rentals, I want to communicate about my transactions within the app to ensure clarity and record-keeping. |
| 6.3 | As a user of the handyman or renting modules, I want to access my history of engagements, including past services and rentals, to keep track of my activities. |

## Non-functional Requirements

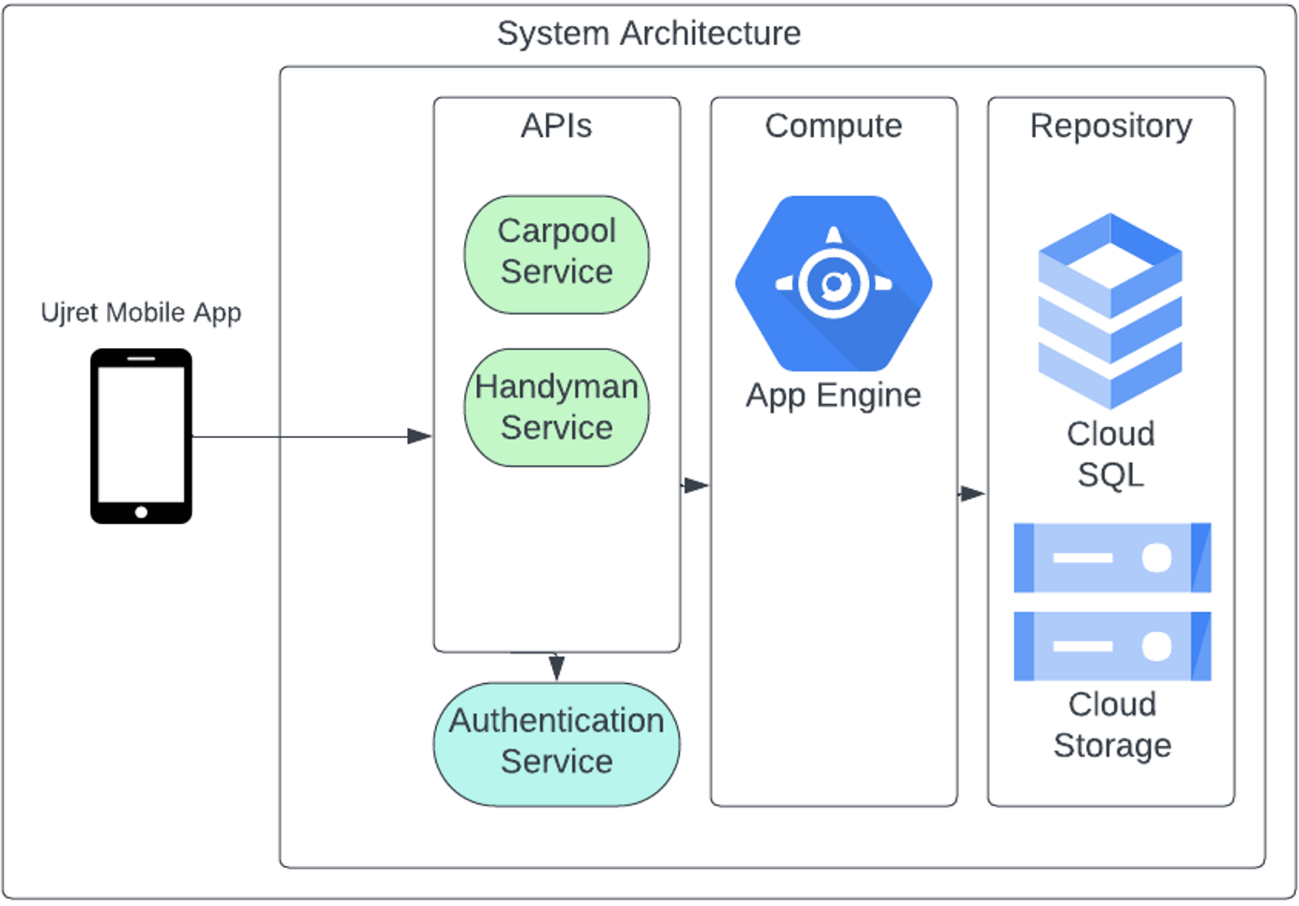
Following is the List of non-functional requirements Quality Attributes

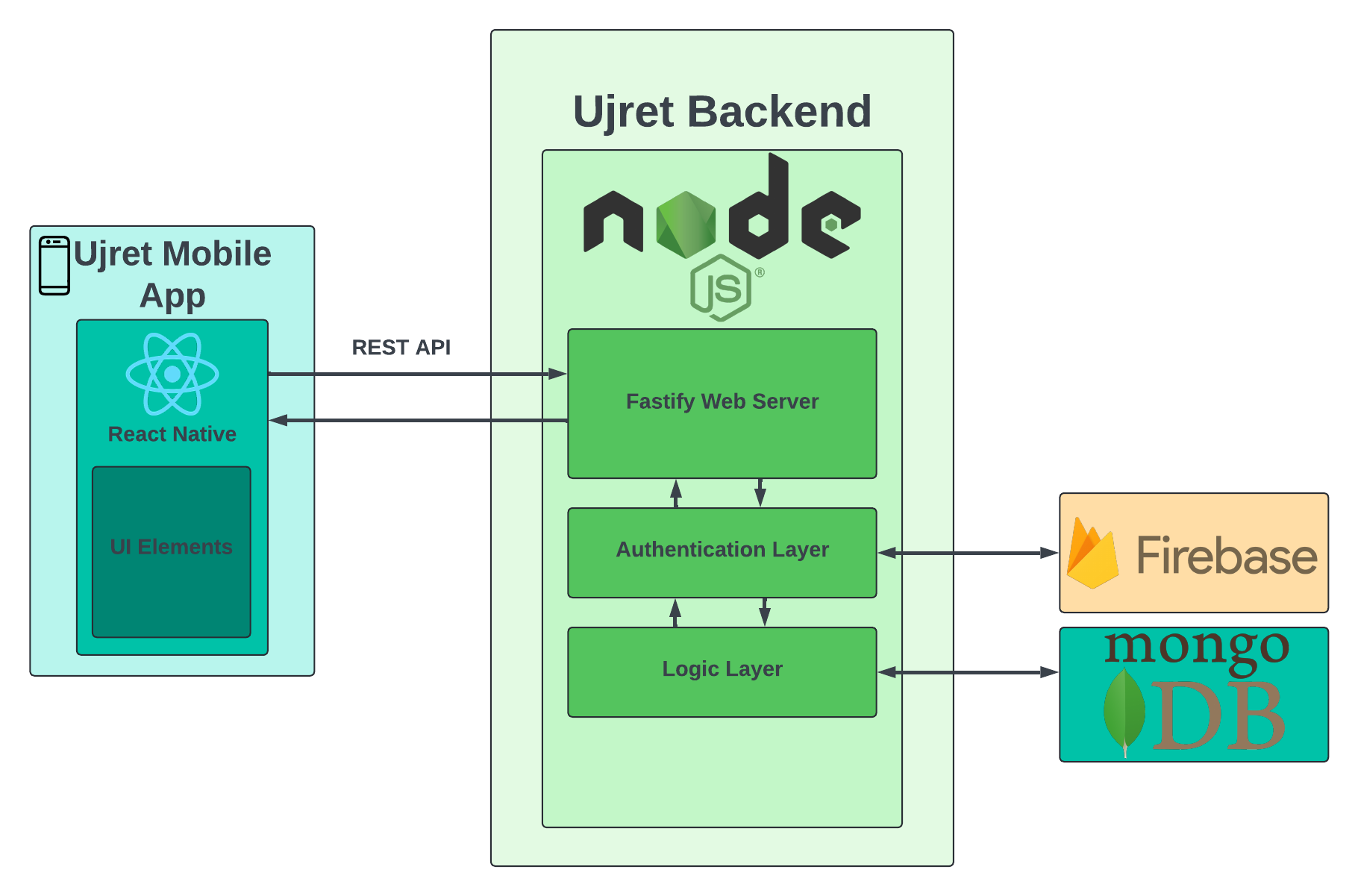
| **Sr#** | **Requirements** |
| --- | --- |
| 1 | The system should exhibit robust reliability, not exceeding more than 3 unexpected downtimes every 24 hours. In the event of a system failure, it must be capable of recovering to operational status within 5 minutes. |
| 2 | The system should support dynamic horizontal scaling to efficiently handle varying traffic loads, ensuring a seamless user experience. While there isn't an explicit upper user limit, the infrastructure should be architected to accommodate a continually expanding user base.  Performance benchmarks, such as response times and concurrency handling, should be established and regularly tested. |
| 3 | The platform must prioritize high usability to accommodate users with varying degrees of tech proficiency. Regular user experience testing and feedback loops through surveys and usability sessions will be essential in measuring and enhancing this attribute. |
| 4 | The app must be fully operational on a broad range of devices, including models up to five years old, ensuring accessibility for users with older technology. This includes compatibility with at least the two most recent major versions of popular mobile operating systems such as iOS and Android. Testing will be conducted across a spectrum of devices to validate this requirement. |
| 5 | Response times for user interactions should not exceed 2 seconds for most operations within the app, contributing to a fluid and responsive user experience. |
| 6 | As the user base and data volume grow, the system should maintain its performance without significant degradation. Scalability will be monitored via load testing and by analyzing key performance metrics at various levels of user and data volume. |
| 7 | All user data, particularly sensitive personal and financial information, must be secured through robust encryption protocols. The system should employ industry-standard encryption methods to protect against data breaches and unauthorized access. |
| 8 | The system must be designed with maintainability in mind, ensuring that future updates, bug fixes, and enhancements can be implemented with minimal disruption. |
| 9 | The platform must adhere to accessibility standards to ensure that users with disabilities can navigate and utilize the app effectively. |
| 10 | The app should comply with all relevant legal and regulatory standards, including data protection laws such as GDPR, CCPA, and any sector-specific regulations. |
| 11 | A comprehensive disaster recovery plan should be in place to handle major system outages or data loss incidents, with the capability to restore full system functionality as per defined recovery time objectives (RTOs). |

# System Architecture

Brief introduction of this chapter in a paragraph highlighting the content

## Architecture Diagram





## Architecture Description

**Frontend**

**React Native**

A framework for building native apps using React, a JavaScript library for building user interfaces. It allows the development of cross-platform mobile apps that can run on both Android and iOS from a single codebase.

The UI elements are the components that create the user interface of the mobile app, such as buttons, text fields, images, etc. They are built using React Native components, which are rendered as native platform widgets.

**Backend**

**Node.js**

An open-source, cross-platform JavaScript runtime environment that executes JavaScript code outside a web browser. It handles HTTP requests and responses, routing, and can plug in various middleware for tasks like parsing, security, etc.

**Authentication Layer**

This layer handles the authentication of users, ensuring that only authorized users can access certain functions within the app.

It uses Firebase for verifying user credentials and managing sessions.

**Logic Layer**

Contains the business logic of the application, such as processing data, handling application workflows, and implementing the rules of the business domain.

It interacts with the database to retrieve, update, and store data based on the application’s operations.

**Databases**

**Firebase**

A platform developed by Google for creating mobile and web applications.

In this architecture, it’s used primarily for user authentication and storing user data.

**MongoDB**

A NoSQL database known for its flexibility and scalability.

Stores application data in a non-relational format, which is beneficial for the performance and development speed.

**Interactions:**

The React Native app sends HTTP requests via a REST API, in JSON format, to the Fastify web server in the backend.

The Fastify web server routes these requests to the appropriate handlers, which involves authentication checks against Firebase or business logic processing in the Logic Layer.

After processing a request, the Logic Layer uses the Mongo Database to fetch or store data, or with Firebase for operations requiring user data or authentication status.

The results are then sent back through the Fastify web server to the React Native app, completing the request-response cycle.

## Justification of the Architecture

This architecture is appropriate for systems that require:

* Scalability: to handle growth in user numbers or data volume.
* Cross-platform support: to cater to both iOS and Android users without duplicating efforts.
* Rapid Development: using popular frameworks can leverage community support and existing libraries.

Non-functional requirements it supports include:

* Performance: Fastify and Node.js can handle high loads, and React Native ensures smooth UI rendering.
* Reliability: Firebase authentication adds a layer of security and reliability.
* Maintainability: A clear separation of concerns makes the system easier to maintain and update.

In conclusion, this architecture is suited for a modern mobile application that requires robust backend processing, secure authentication, and the ability to scale, all while maintaining a consistent user experience across different mobile platforms.

## Tools and Technologies

**Backend Development Stack:**

1. **MongoDB**:

Description: A NoSQL database program, MongoDB uses JSON-like documents with optional schemas.

1. **Node.js:**

Description: A JavaScript runtime built on Chrome's V8 JavaScript engine for building fast, scalable network applications.

**Frontend Development:**

1. **React Native CLI:**

Description: A command-line tool for creating and managing React Native apps that run natively on both Android and iOS devices.

1. **Firebase Authentication:**

Description: A service provided by Google for authenticating users via passwords, phone numbers, popular federated identity providers like Google, Facebook and Twitter, and more.

Other Technologies and Tools:

1. **GitHub:**

Description: A Git repository hosting service providing distributed version control and source code management functionality.

1. **Visual Studio Code:**

Description: A source-code editor made by Microsoft for Windows, Linux and macOS.

1. **Postman:**

Description: A popular API client that makes it easy for developers to create, share, test and document APIs.

1. **ESLint (for JavaScript linting):**

Description: A static code analysis tool for identifying problematic patterns found in JavaScript code.

1. **Pylint (for Python linting):**

Description: A static code analysis tool for improving code quality and finding bugs in Python code.

**Backend Versions**

* MongoDB (Database): Version: 5.0.6
* Node.js (JavaScript Runtime): Version: 20.12.1
* Fastify (Node.js Web Application Framework): Version: 4.17.1
* DataGrip (Database IDE by JetBrains): Version: 2022.1
* Flask (Python Web Framework): Version: 2.1.0
* PostgreSQL (Database): Version: 14.1
* Ngrok (Secure Tunneling): Version: 2.3.40

**Frontend Versions**

* React Native (Mobile Application Framework): Version: 0.66.3
* React Native CLI (Command Line Interface): Version: 2.0.1
* Firebase (Authentication and Backend Services): Version: 9.6.1

**Deployment and Version Control versions**

* Git (Version Control System): Version: 2.44.1
* GitHub (Hosting for Software Development and Version Control): Version: 3.12.2

**Development Tools**

* Visual Studio Code (Source-code Editor): Version: 1.62
* Postman (API Development Environment): Version: 9.2.0

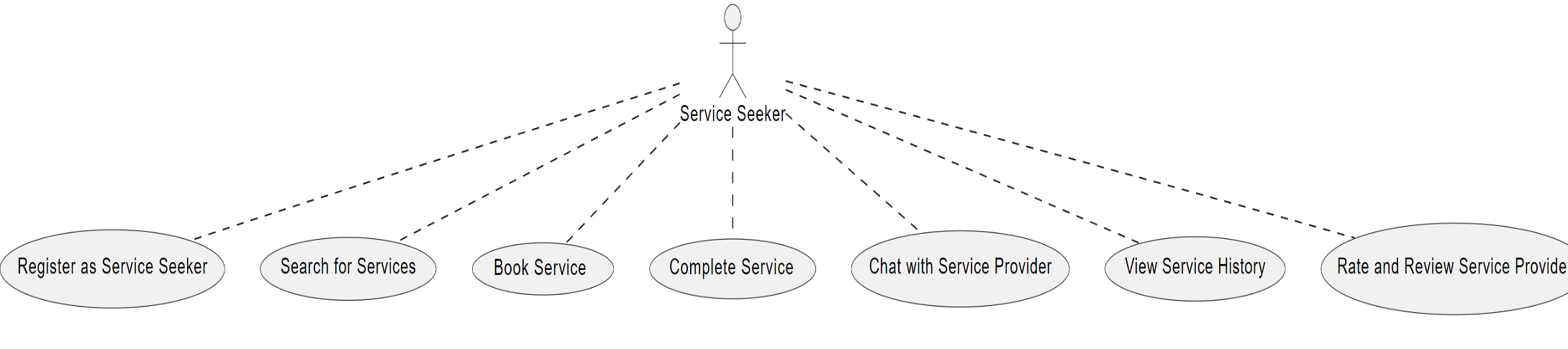
Additional Tools and Libraries

* Bcrypt (Password Hashing): Version: 5.0.1
* JWT (JSON Web Tokens for Authentication): Version: 8.5.1
* Axios (HTTP Client for Browser and Node.js): Version: 0.24.0
* Redux (State Management): Version: 4.1.2

# Requirements Specifications

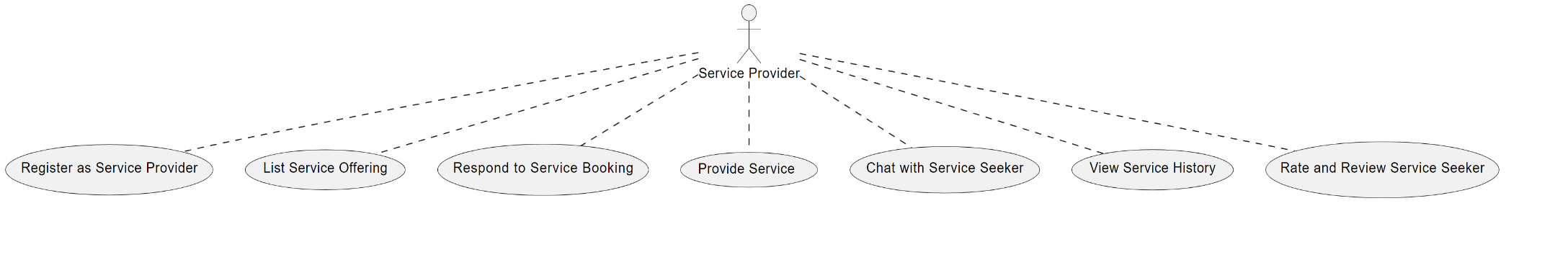
Brief introduction of this chapter in a paragraph highlighting the content

## Use Cases



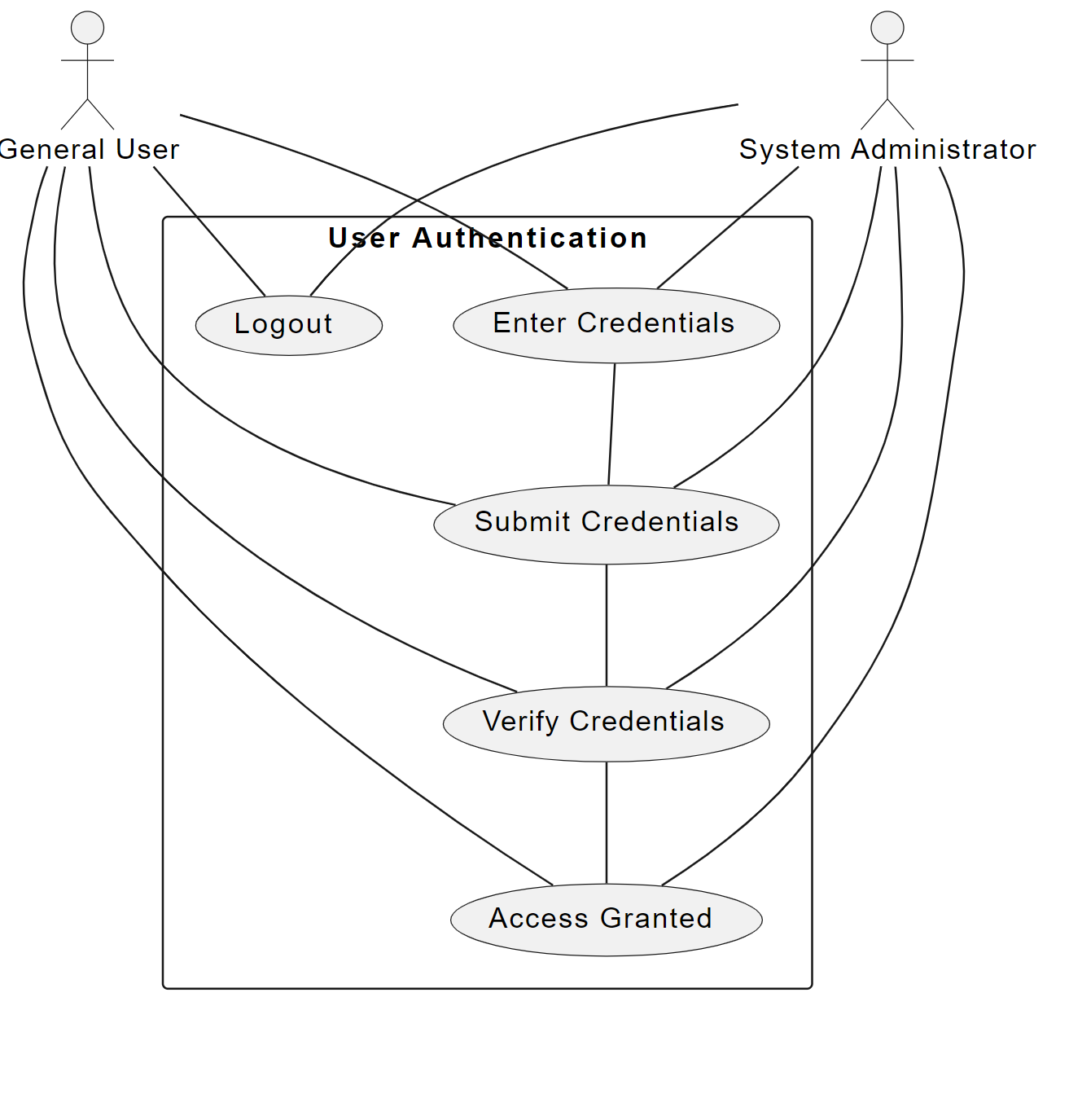
**Handyman Service Seeker**

* Register as Service Seeker: The user creates an account to access services.
* Search for Services: The user searches the app's database for available handyman services.
* Book Service: The user selects and books a specific service.
* Complete Service: The user marks a service as completed after it is fulfilled.
* Chat with Service Provider: The user communicates directly with the service provider to discuss service details.
* View Service History: The user reviews their past service bookings and completions.
* Rate and Review Service Provider: The user provides feedback on the service received through ratings and written reviews.



**Handyman Service Provider**

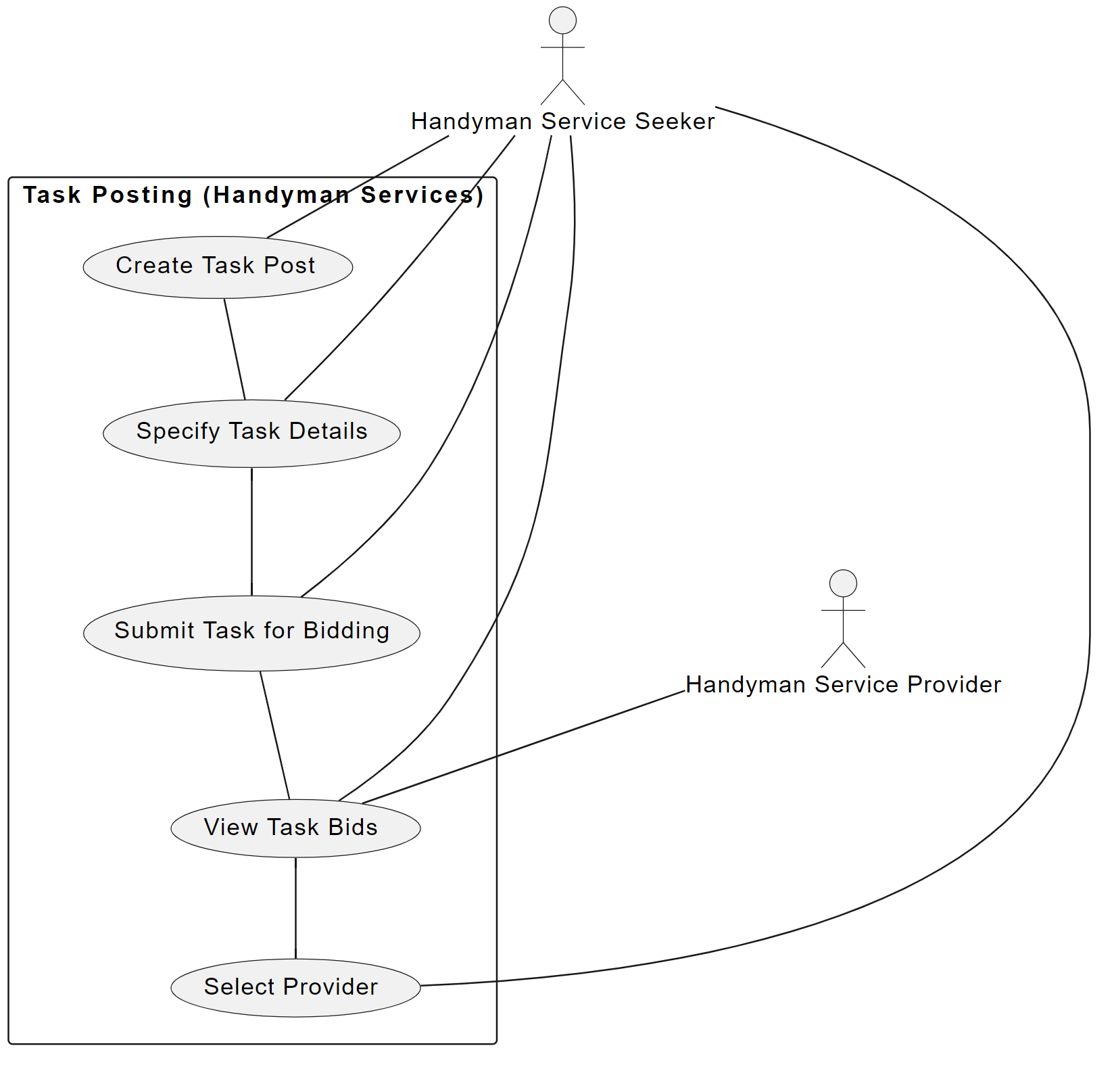
* Register as Service Provider: The service provider sets up an account to offer services.
* List Service Offering: The provider lists the services they offer, with details for seekers to view.
* Respond to Service Booking: The provider responds to booking inquiries from seekers.
* Provide Service: The provider completes the service as per the agreement with the seeker.
* Chat with Service Seeker: The provider engages in direct communication with the service seeker.
* View Service History: The provider checks their history of services provided.
* Rate and Review Service Seeker: The provider rates and reviews the service seeker post-service.



**Description of the Use Case: User Registration**

This diagram (or use case) depicts the following key steps in the user registration process for the Ujret Mobile App:

* Access Registration Page: The new user initiates the process by selecting the registration option upon opening the app.
* Enter Details: The user is prompted to input necessary personal and contact information,
* Role-specific details (if applicable, e.g., licensure for service providers or item details for lenders)
* Submit Details: After filling in their information, the user submits the registration form to create their account.
* Verify Contact Information: To ensure the authenticity of the provided contact details, the system sends a verification code via email or SMS.
* Enter Verification Code: The user enters the verification code received to confirm their contact details.
* Complete Registration: Upon successful verification, the registration process is completed, and the user's account is activated, granting them access to use the platform according to their role (e.g., service seeker, service provider, item lender, item renter).

**Description of the Use Case Diagram**

This diagram depicts the following key steps in the task posting process for handyman services:

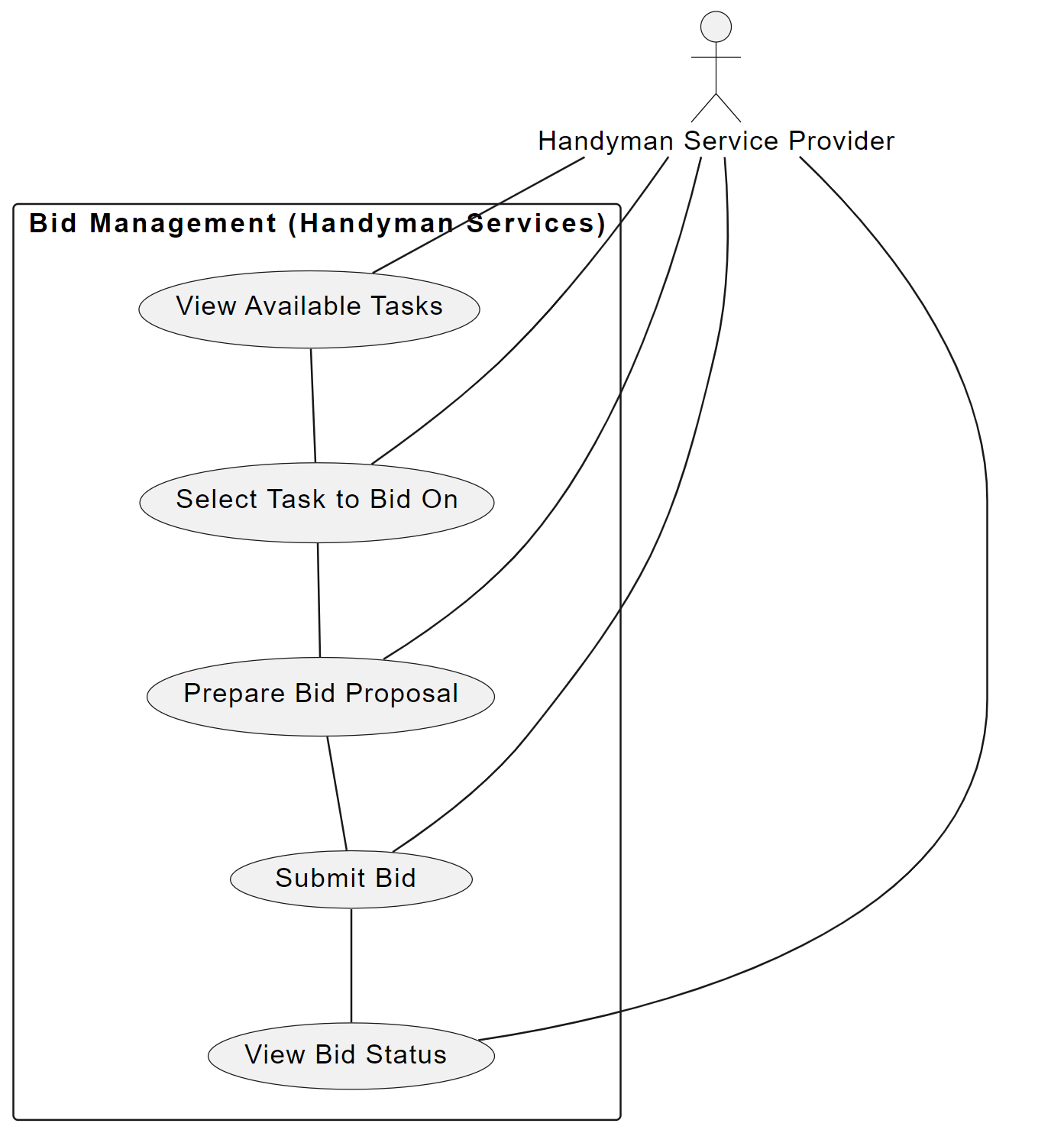
**Create Task Post:** The handyman service seeker initiates the process by creating a new task post.

Specify Task Details: The seeker provides detailed information about the task, such as the nature of the work, required skills, and expected completion time.

Submit Task for Bidding: Once the task details are specified, the seeker submits the task for bidding, making it visible to potential service providers.

View Task Bids: The seeker can view bids from various service providers, assessing each bid based on the provider's proposal and profile.

Select Provider: After reviewing the bids, the seeker selects the most suitable service provider for the task.



This diagram outlines the key steps involved in the bid management process for handyman service providers:

View Available Tasks: Service providers log into the platform and view a list of available tasks posted by service seekers.

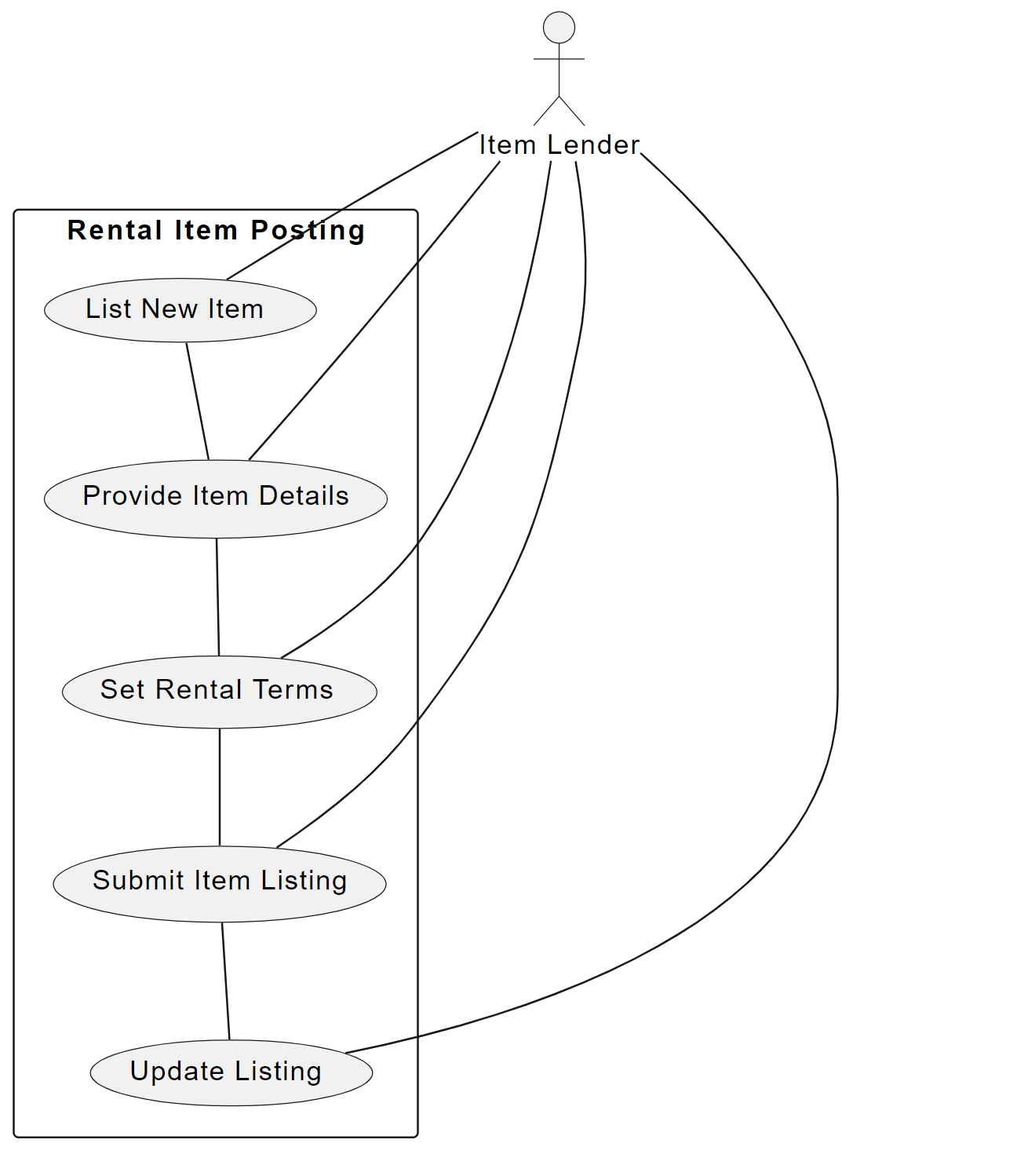
Select Task to Bid On: From the list, providers select a specific task that matches their skills and availability.

Prepare Bid Proposal: Providers prepare a detailed bid proposal, including cost estimates and time frames.

Submit Bid: The completed bid is submitted back to the platform for consideration by the service seeker.

View Bid Status: After submitting, providers can track the status of their bid, whether it's pending, accepted, or rejected.

This sequence facilitates a structured and competitive environment for service providers to offer their services effectively within the platform.



This diagram outlines the key steps involved in the process of posting rental items:

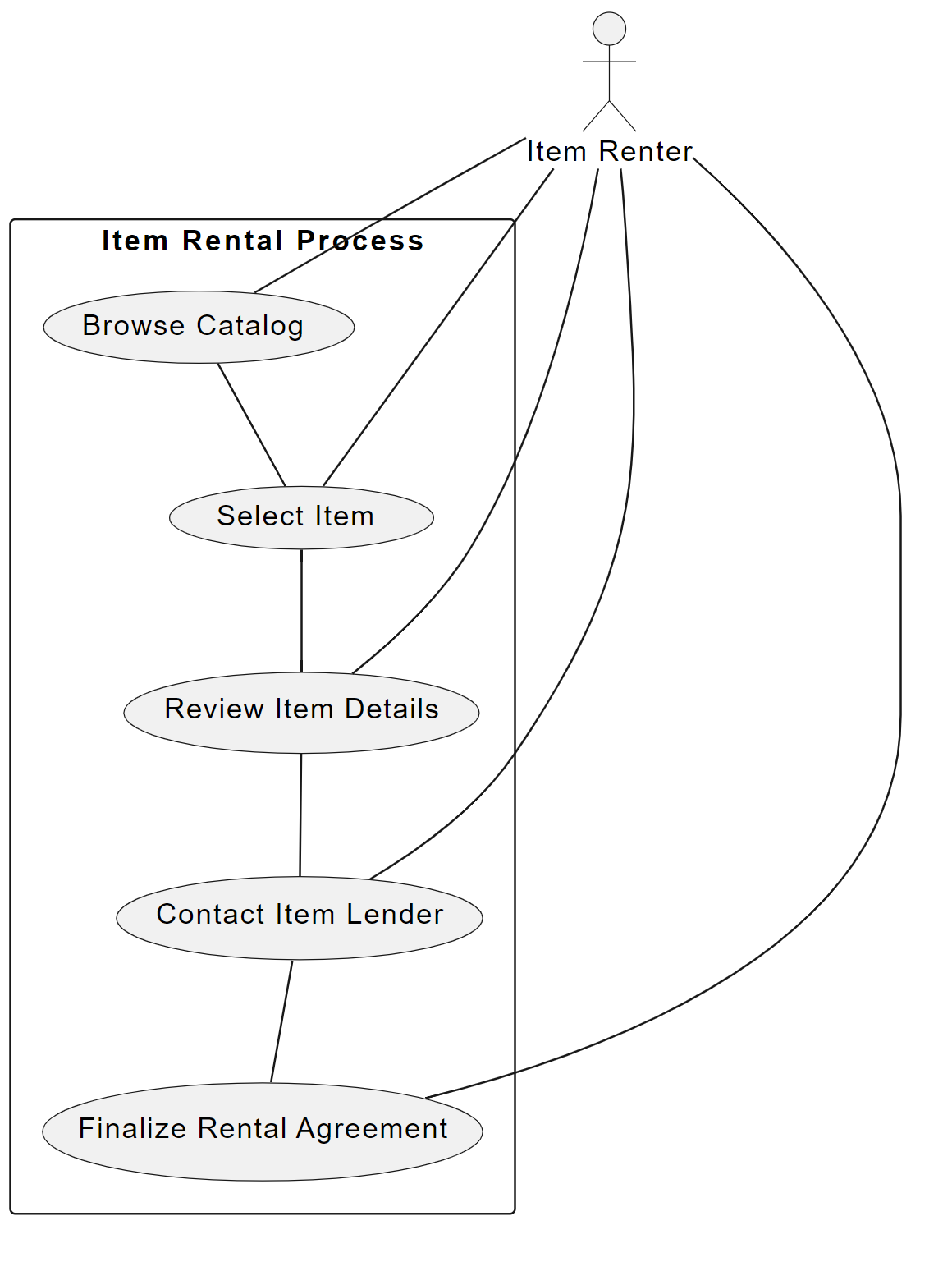
List New Item: The item lender initiates the process by choosing to list a new item available for rent.

Provide Item Details: The lender enters detailed information about the item, such as description, condition, and photos.

Set Rental Terms: The lender specifies the rental terms, including price, duration, and any deposit required.

Submit Item Listing: Once all details and terms are set, the lender submits the item listing to make it available on the platform.

Update Listing: The lender can return to update the listing with additional information or adjustments to terms as needed.



This diagram outlines the key steps involved in the item rental process:

Browse Catalog: Renters access the platform's catalog to browse through available rental items.

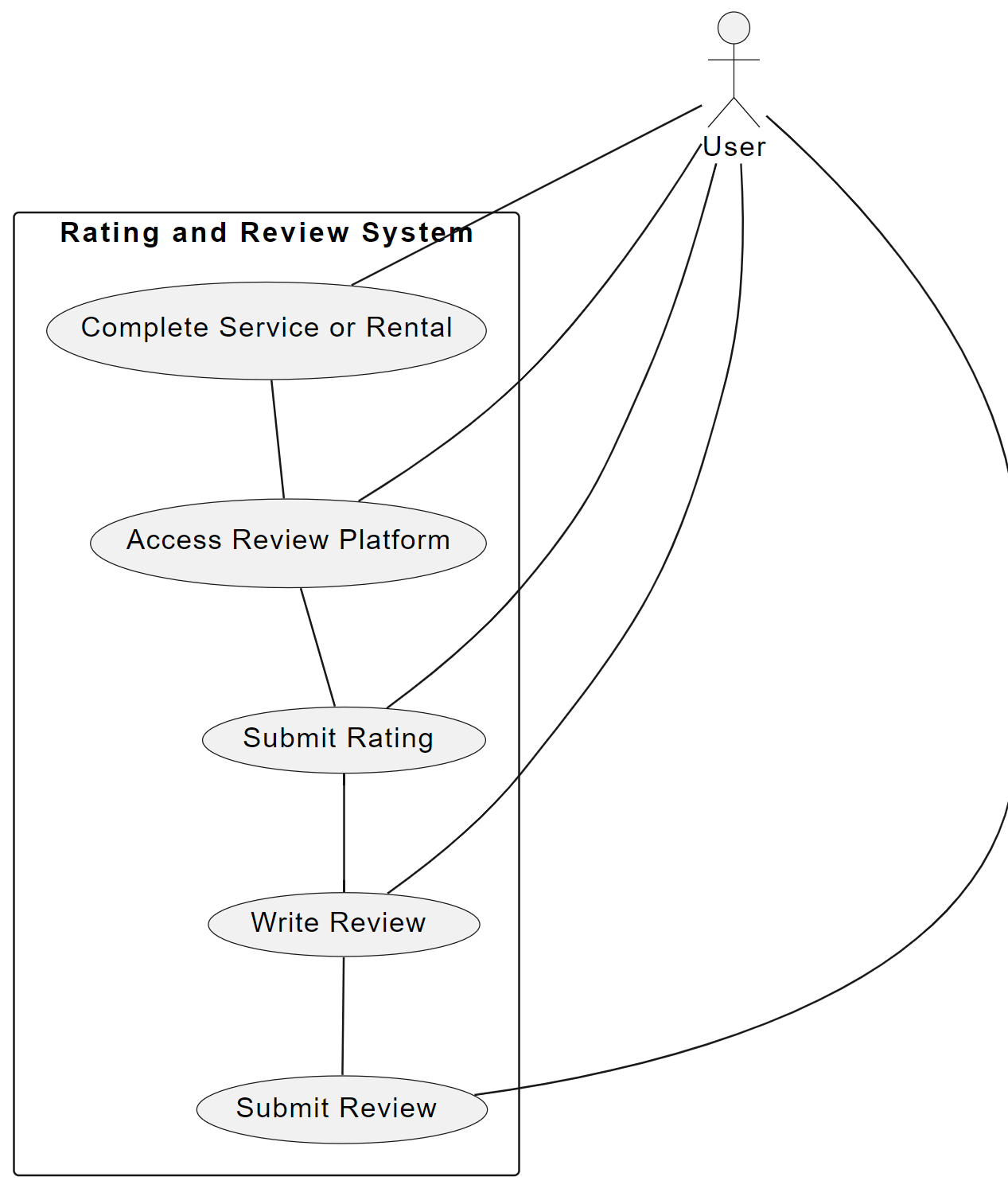
Select Item: Upon finding an item of interest, the renter selects it for more details.

Review Item Details: The renter reviews the detailed information provided about the item, including pricing, rental terms, and availability.

Contact Item Lender: If the item meets their needs, the renter initiates contact with the lender directly through the platform to discuss further details.

Finalize Rental Agreement: The renter and lender agree on the terms, and the rental agreement is finalized.

This sequence facilitates a smooth rental process, ensuring that renters can easily find and rent items while communicating effectively with lenders to secure their needs.



This diagram outlines the key steps involved in the rating and review process:

Complete Service or Rental: After the completion of a service or rental, the user initiates the review process.

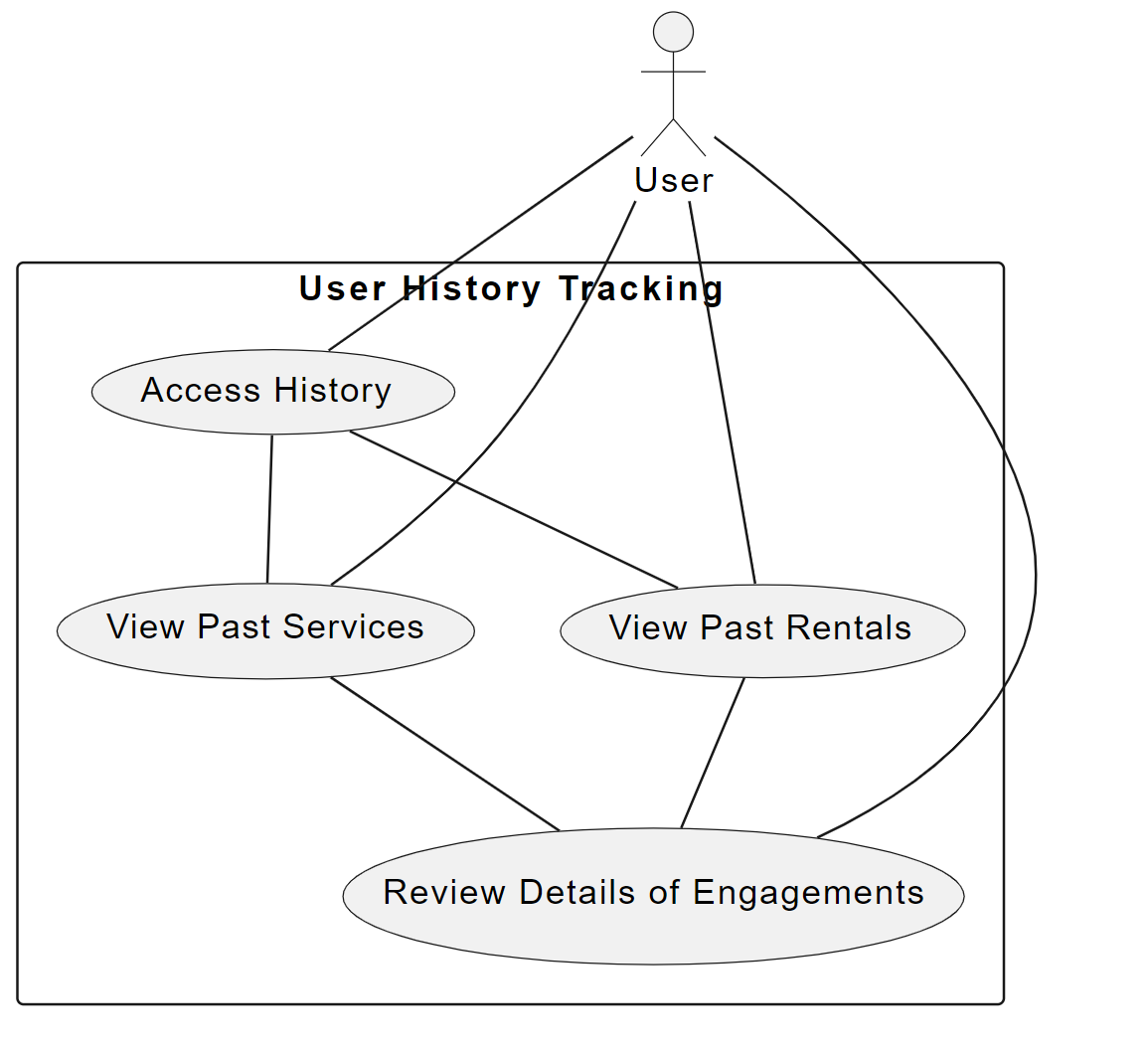
Access Review Platform: The user accesses the review section within the app designated for feedback on the completed transaction.

Submit Rating: The user submits a numeric rating reflecting their satisfaction with the service or rental.

Write Review: The user writes a textual review, detailing their experience and any feedback.

Submit Review: The review and rating are submitted and published, contributing to the community's trust and reliability.

This sequence ensures a structured process for providing feedback, which is crucial for maintaining a reliable and user-friendly community environment on the platform.

This diagram outlines the key steps involved in the user history tracking process:

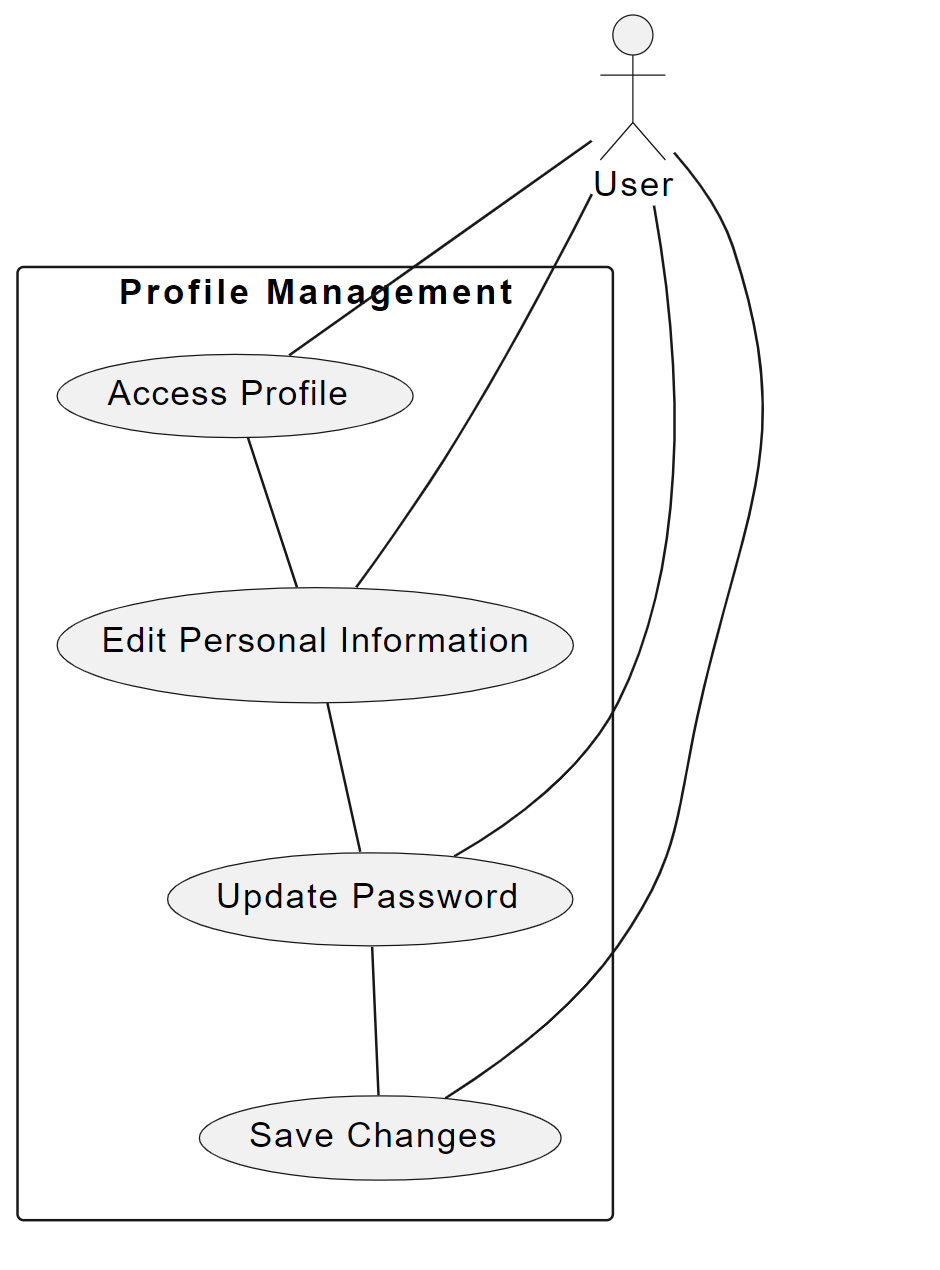
Access History: Users log into their account and access the history section of the app.

View Past Services: Users can view details of past services they have used or offered, depending on their role (service seeker or provider).

View Past Rentals: Users can view details of past rentals, either as lenders or renters.

Review Details of Engagements: Users can review detailed information about each past engagement, including service or rental specifics, dates, and other participants involved.

This sequence allows users to effectively monitor and reflect on their past activities and interactions within the platform, enhancing user experience and trust.



This diagram outlines the key steps involved in managing a user's profile:

Access Profile: Users log into their account and access their profile settings.

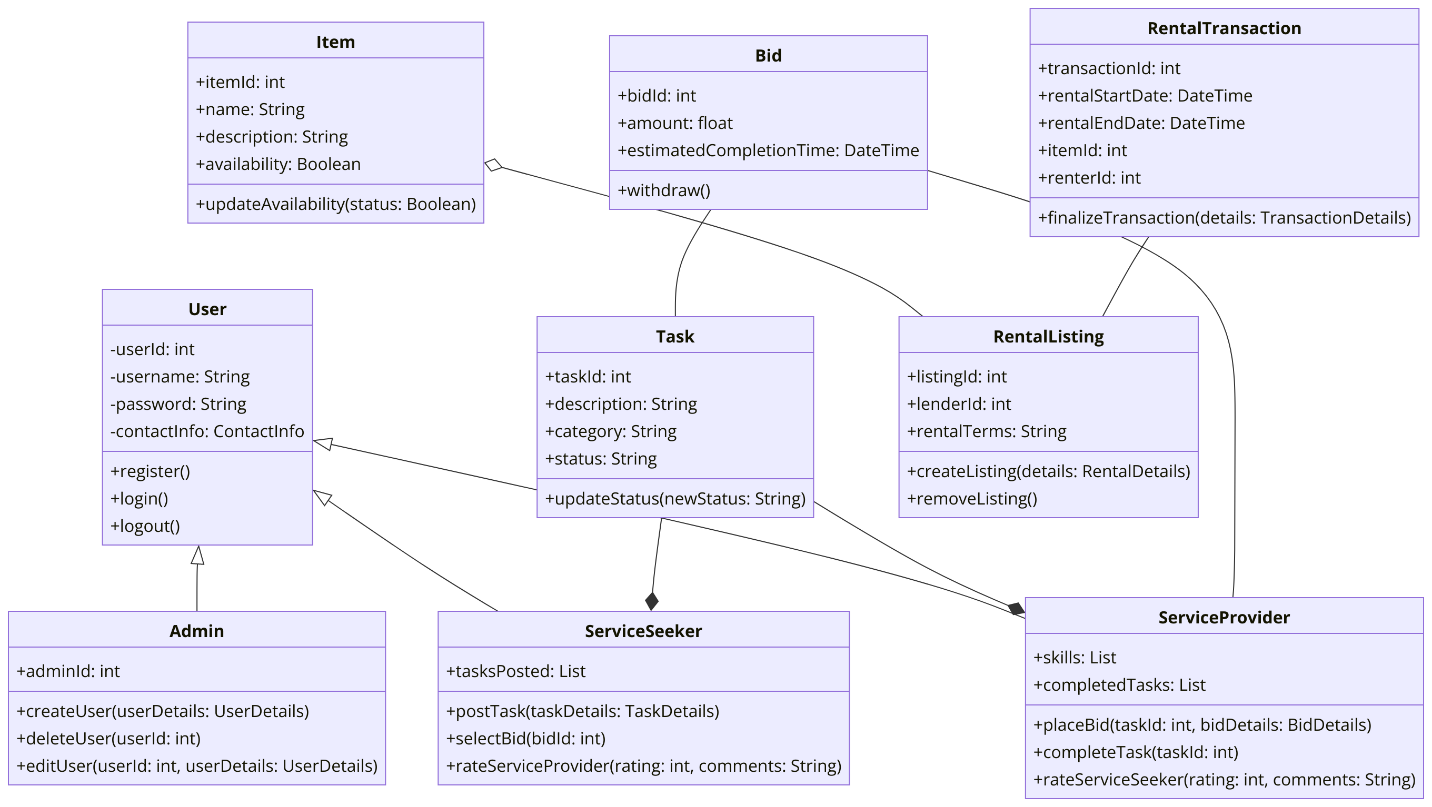
Edit Personal Information: Users can edit personal information such as their name, phone number, address, and other details.

Update Password: Users have the option to update their password for security reasons.

Save Changes: After making the desired changes, users save them to update their profile.

This sequence ensures that users have full control over their personal information and account security settings, enhancing their experience and trust in the platform.

## Class Diagram

sysUser:

* Description: This class represents a generic user within the system, serving as a base class for all types of users.
* Attributes:
* userId: A unique identifier for the user.
* username: The user's chosen name for login identification.
* password: A hashed representation of the user's password to ensure security.
* contactInfo: Structured data that includes email, phone number, etc.
* Operations:
* register(): Function to create a new user account.
* login(): Function that authenticates a user and starts a session.
* logout(): Function that ends the user's session.

Admin:

* Description: A specialized version of the User class with additional administrative privileges.
* Attributes:
* adminId: A unique identifier for an admin, differentiating from regular users.
* Operations:
* createUser(userDetails): Allows for the creation of new user accounts.
* deleteUser(userId): Permits the deletion of existing user accounts.
* editUser(userId, userDetails): Enables the editing of user account details.

ServiceSeeker:

* Description: Represents users who are looking for handyman services and wish to post tasks.
* Attributes:
* tasksPosted: A list of tasks that the user has posted.
* Operations:
* postTask(taskDetails): Allows users to post a new service request.
* selectBid(bidId): Enables the selection of a bid from a service provider.
* rateServiceProvider(rating, comments): Facilitates the rating of a service provider after task completion.

ServiceProvider:

* Description: Users who provide handyman services and respond to posted tasks with bids.
* Attributes:
* skills: A list of skills the provider has.
* completedTasks: A list of tasks the provider has completed.
* Operations:
* placeBid(taskId, bidDetails): Function to place a bid on an available task.
* completeTask(taskId): Marks a task as completed.
* rateServiceSeeker(rating, comments): Allows rating the service seeker post-task completion.

Task:

* Description: Represents a task posted by a service seeker that needs to be accomplished.
* Attributes:
* taskId: A unique identifier for the task.
* description: A detailed description of the task.
* category: The category of service required.
* status: The current status of the task (e.g., open, in progress, completed).
* Operations:
* updateStatus(newStatus): Update the current status of the task.

Bid:

* Description: Represents an offer made by a service provider to complete a posted task.
* Attributes:
* bidId: A unique identifier for the bid.
* amount: The proposed price to complete the task.
* estimatedCompletionTime: The time the provider estimates it will take to complete the task.
* Operations:
* withdraw(): A function that allows a service provider to withdraw their bid.

Item:

* Description: Represents items that can be rented through the platform.
* Attributes:
* itemId: A unique identifier for the item.
* name: The name of the item.
* description: A detailed description of the item.
* availability: Indicates whether the item is available for rent.
* Operations:
* updateAvailability(status): Changes the availability status of the item.

RentalListing:

* Description: Detailed listing information about an item available for rent.
* Attributes:
* listingId: A unique identifier for the listing.
* lenderId: The identifier of the user who is renting out the item.
* rentalTerms: The terms and conditions of the rental.
* Operations:
* createListing(details): Creates a new listing for an item.
* removeListing(): Removes an existing listing from the platform.

RentalTransaction:

* Description: Captures the details of a rental agreement between an item lender and renter.
* Attributes:
* transactionId: A unique identifier for the rental transaction.
* rentalStartDate: The date when the rental period starts.
* rentalEndDate: The date when the rental period ends.
* itemId: The identifier of the rented item.
* renterId: The identifier of the user who rents the item.
* Operations:
* finalizeTransaction(details): Confirms and finalizes the terms of the rental transaction.

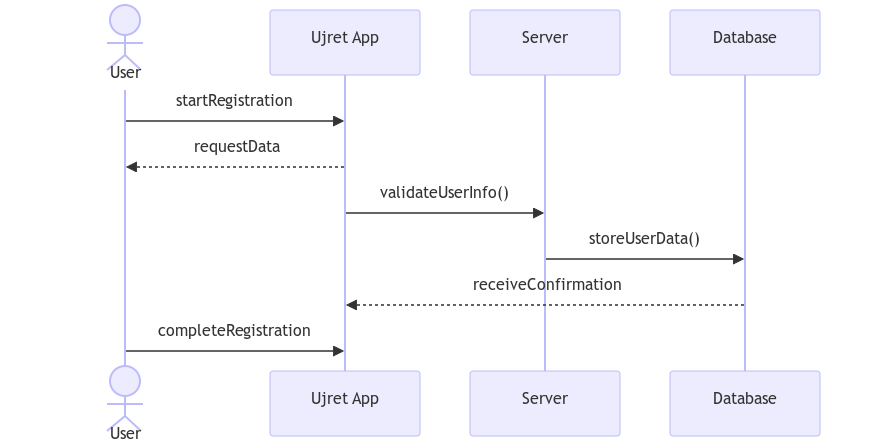
Each class in the diagram has been designed to encapsulate specific data and behaviors relevant to the user's role in the Ujret Mobile App ecosystem, fostering a modular and scalable system architecture.

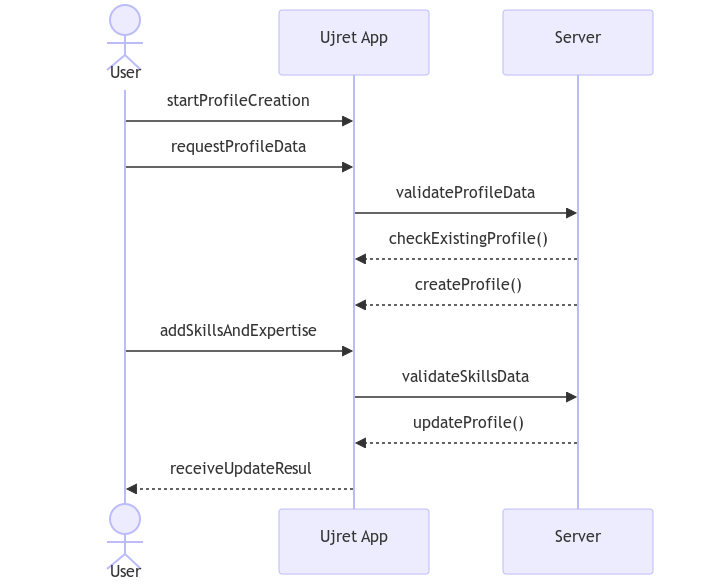
## Sequence Diagrams

Draw sequence diagrams of 10 core use cases. Draw the diagrams using standard UML notation

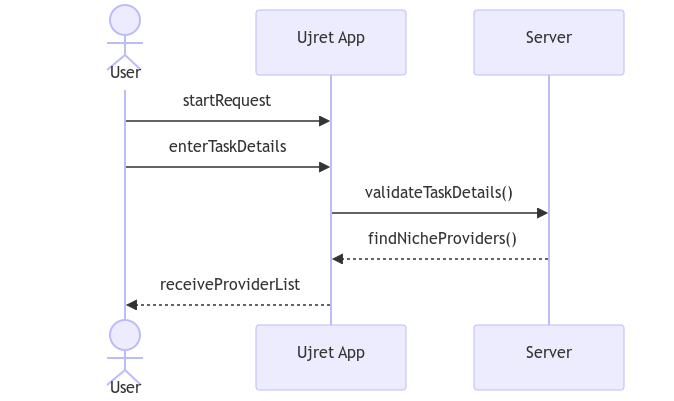
Handyman:

**User Registration:**

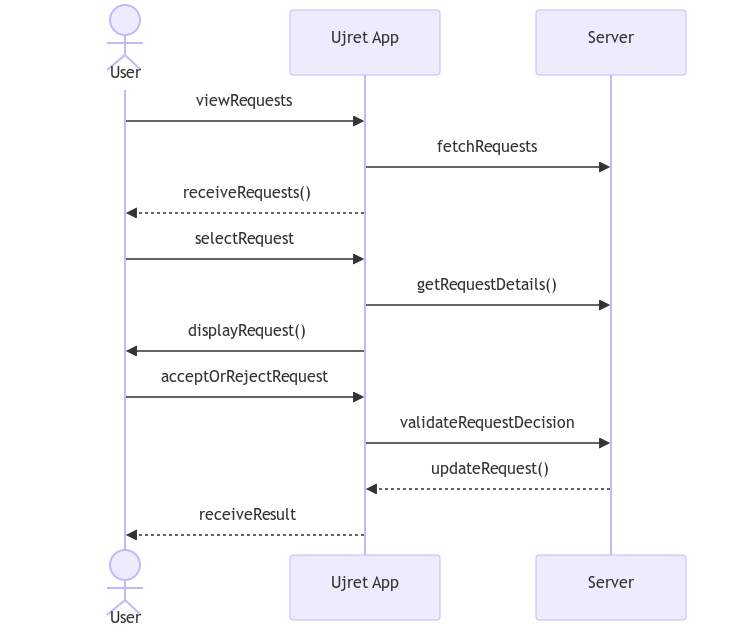
  
**Creates and manages the profile:**

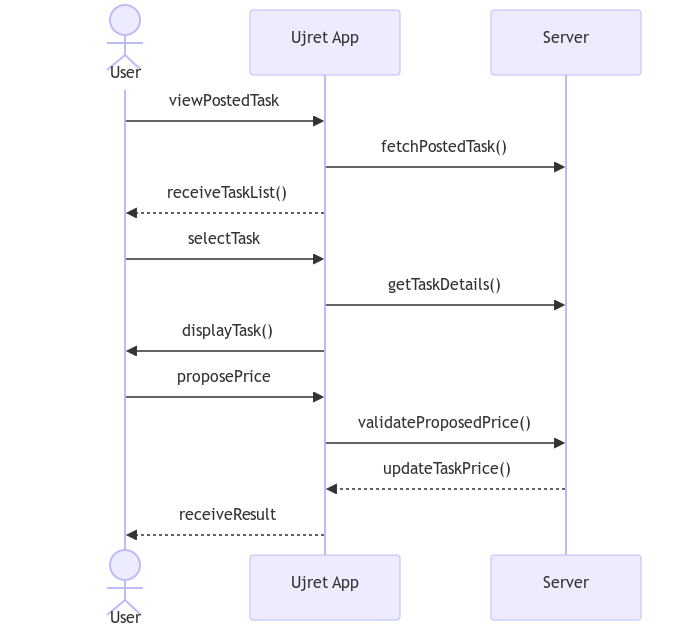


**Post request with task details:**



**Service Provider accepts or rejects service requests:**

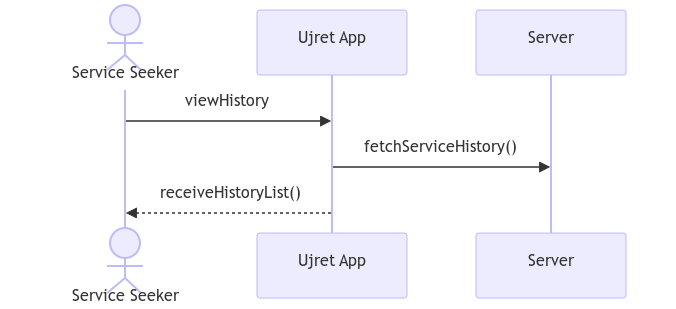


**Service Provider can propose his/her own price on posted task:**  


**Service Provider marks the service as completed:**

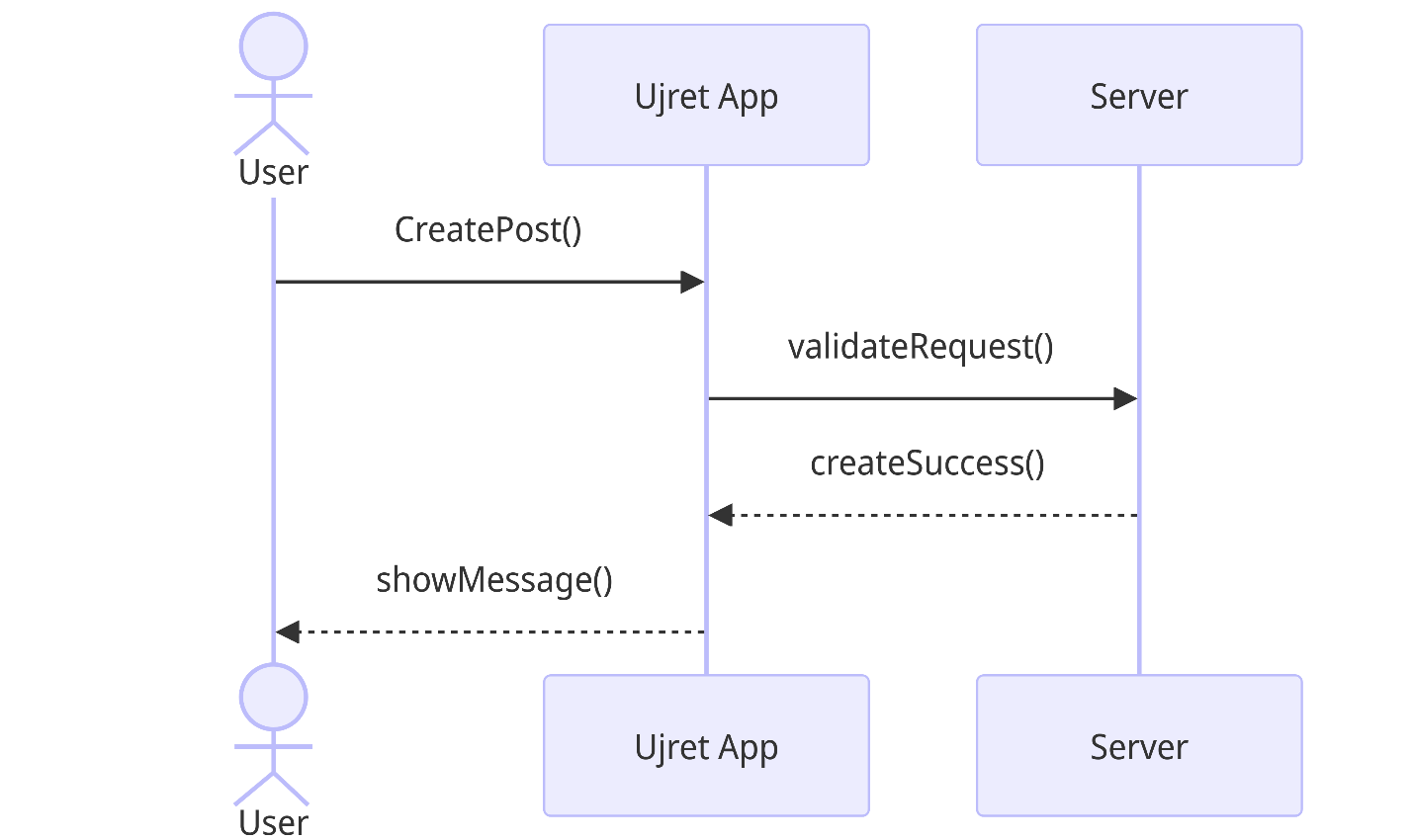


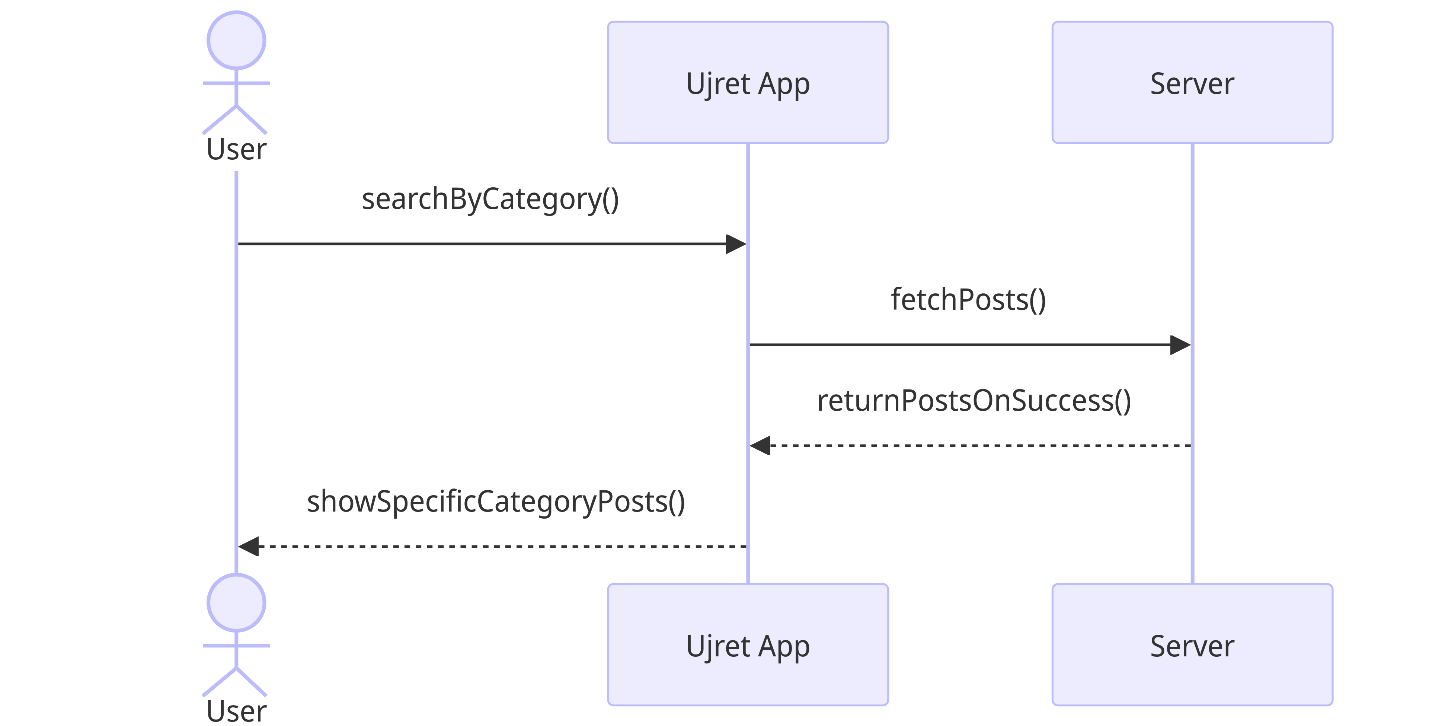
**Service Seeker views the history of services availed:**



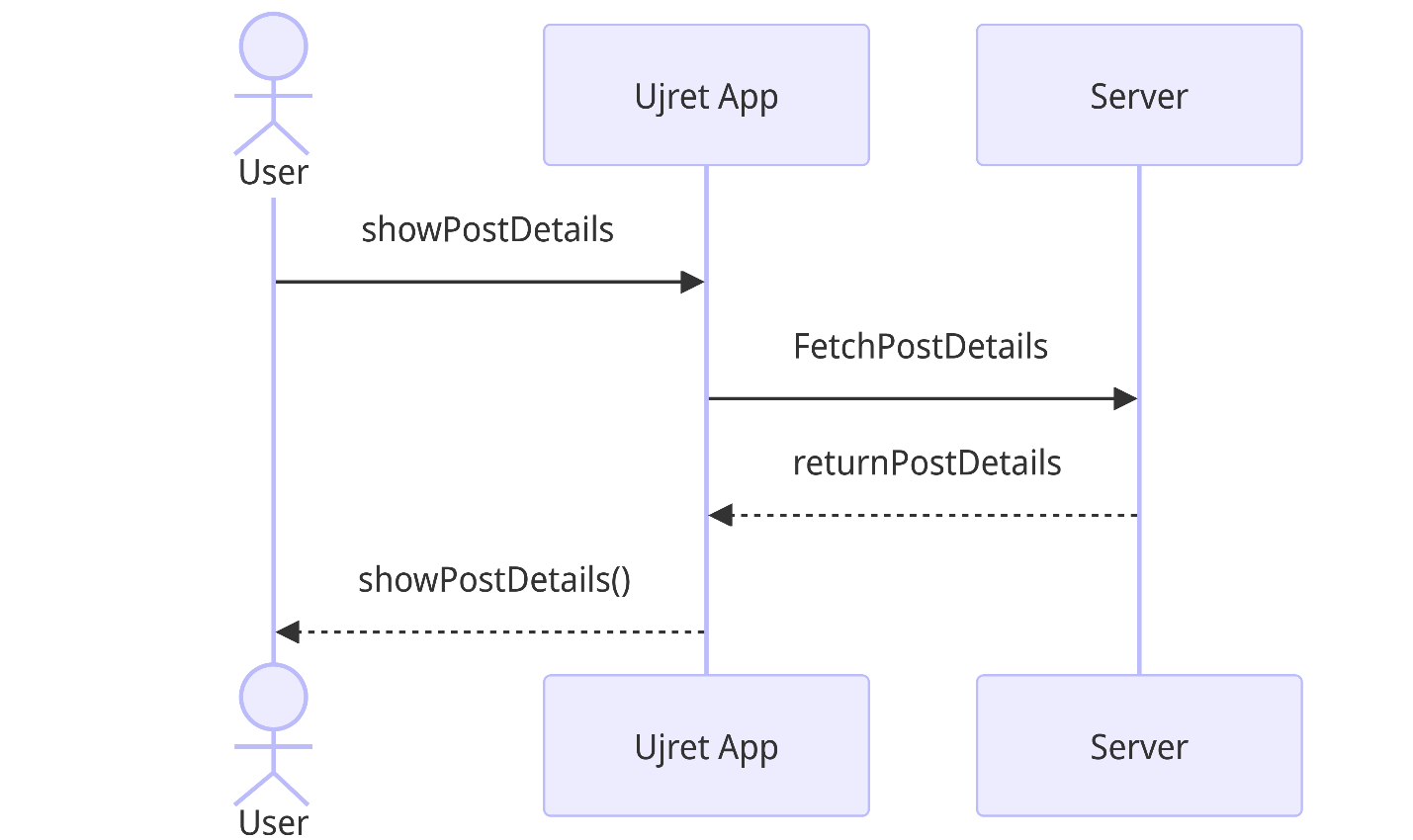
**Tool Renting:**

**Create a post for renting tools:**

**Search tools by category for renting:**



**Show renting tools post details:**



# Software Development Methodology and Plan

Brief introduction of this chapter in a paragraph highlighting the content

## Software Process Selection

Discuss a pros and cons of waterfall and agile (scrum) processes in your own words.

Give proper justification of the software development process that you have used for your project.

**Waterfall Process**

**Pros:**

* Structured Approach: The Waterfall model is linear and sequential, providing a clear structure. Each phase must be completed before the next begins, which makes it straightforward to understand and manage.
* Documentation Focused: It requires extensive documentation upfront, which can be helpful for new team members and for maintaining a clear record of the project's progress and scope.
* Predictability: With its rigid structure, the Waterfall model can offer more predictability in terms of timeline and budget, as long as the scope remains unchanged.

Cons:

* Inflexibility: Once a phase has been completed, it is difficult and often costly to go back and make changes. This lack of flexibility can be problematic if project requirements evolve or are not clearly understood from the beginning.
* Delayed Testing: Testing is left until after the completion of all preceding stages, which can lead to the discovery of issues late in the development process, making them more expensive and time-consuming to address.
* User Feedback: There is little room for client or user feedback during the development process. The final product may not meet user needs if these have changed or were not adequately captured during the requirements phase.

**Agile (Scrum) Process**

Pros:

* Flexibility and Adaptability: Agile methodologies, like Scrum, embrace change. They allow for frequent reassessment of completed work and adaptation of plans, which is ideal for projects with evolving requirements.
* Client Engagement: Regular reviews and iterations invite constant client feedback, ensuring the final product is more likely to meet the client’s needs.
* Early and Continuous Testing: Testing is integrated from the beginning, allowing teams to identify and fix issues early on, reducing the risk of major bugs or project delays.

**Cons:**

* Less Predictability: Agile methods can be less predictable in terms of budget and timeline because the scope can evolve during the project.
* Documentation Can Be Overlooked: The focus on working software over comprehensive documentation can sometimes result in a lack of documentation, which can pose challenges for new team members or when handing over the project.
* Requires Experienced Team Members: To effectively implement Agile methodologies, team members often need to be experienced and self-motivated. Newer teams may struggle without clear hierarchies or well-defined processes.

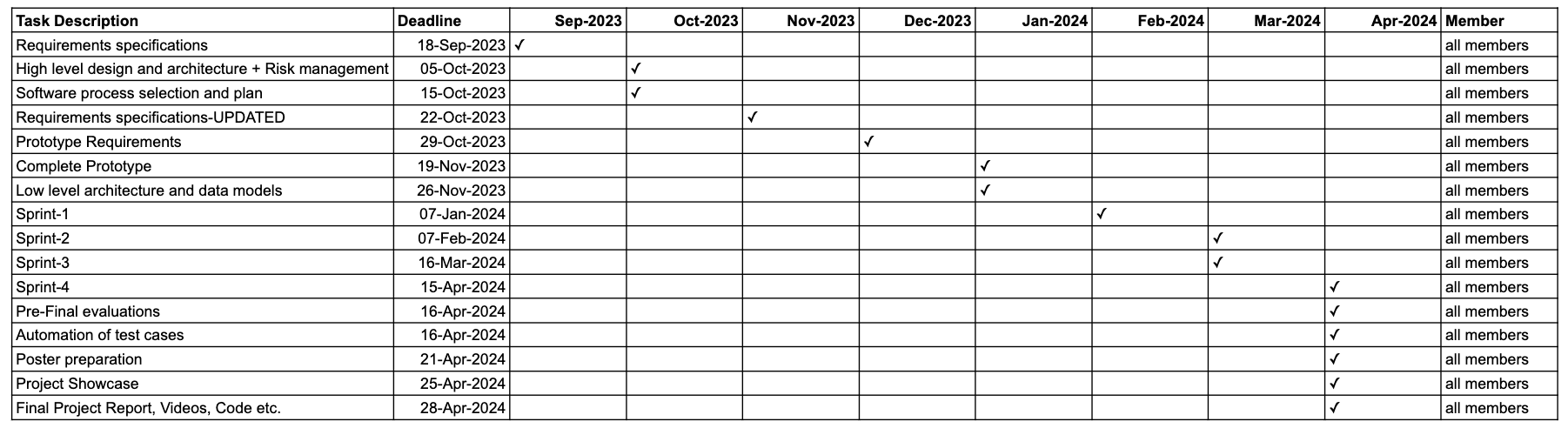
**Justification for the Development Process Used**

For the Ujret Mobile App, the Agile (Scrum) development process was employed due to its strong alignment with the project’s needs:

* Evolving Requirements: The app's development involved evolving requirements, with initial plans adapting from a handyman and carpooling focus to a handyman and renting focus. Agile’s iterative nature allowed for such pivots without derailing the project.
* Team Size (5 Members): With a team size of five members (2 frontend developers, 2 backend developers, and 1 UI designer), Agile is well-suited. It encourages close collaboration and active communication within small teams, ensuring that each team member's contributions are acknowledged and integrated effectively.
* Developers' Experience/Skills**:** As students, our team members may have varying levels of experience, making Agile's adaptive and iterative nature particularly valuable. It allows team members to learn and adapt as the project progresses.
* Rate of Requirements Change**:** Given the dynamic nature of app development and the potential for evolving user requirements, Agile's ability to accommodate changing requirements aligns well with our project context.
* Pressure to Develop Early Releases**:** Agile promotes incremental releases, which means that we can deliver valuable increments of the app early, allowing us to meet the need for early releases while ensuring that each release is functional and tested.
* Availability of Reusable Components**:** Agile encourages the use of reusable components and code, which can be beneficial for our project's efficiency, especially in handyman services module of our app which has many overlapping features.
* User-Centered Design: Agile methodologies facilitated ongoing feedback from users, which was crucial for an app designed around user interaction and satisfaction. This allowed the app to be more responsive to user needs and preferences.
* Complexity and Uncertainty: The project’s complexity, with its need for real-time features such as bidding and direct communication, benefited from Agile’s iterative approach and frequent testing, ensuring that features were refined through continuous integration and feedback loops.

In conclusion, the choice of Agile (Scrum) for the Ujret Mobile App project was driven by its need for flexibility, its focus on user engagement, and the complex nature of its features. This approach allowed for a dynamic and responsive development process, resulting in a product that could adapt to changing requirements and user feedback.

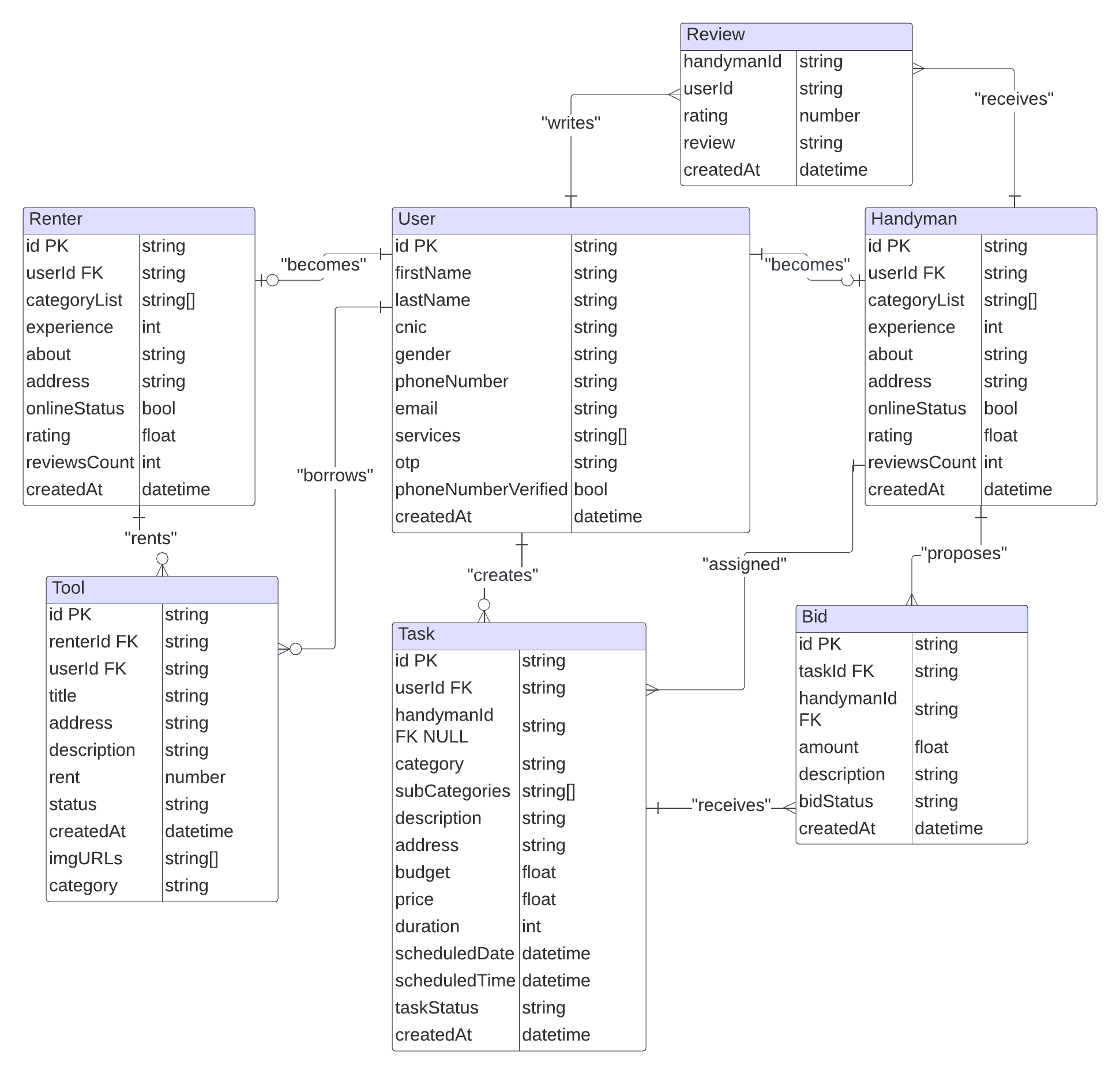
## Gantt Chart



# Database Design and Web Services

Brief introduction of this chapter in a paragraph

## Database Design

Database Entity-Relationship Diagram:

## API Specification

Current API Integrations

1. Google Maps API
   * Purpose: This API is used to facilitate location-based services within the app. It allows users to pinpoint addresses for the handyman services, enabling service providers to locate job sites accurately. Additionally, it assists item lenders and renters in specifying and locating exchange locations.

**Features Utilized:**

* + Geocoding: Converts addresses into geographic coordinates.
  + Maps and Views: Displays interactive maps for users to visualize locations.
  + Places: Identifies nearby places of interest to enhance user experience when choosing job or exchange locations.

1. WhatsApp API:

* Purpose: The WhatsApp Business API enables direct communication between users within the app. This feature allows users to quickly transition from Ujret’s platform to WhatsApp for more detailed and personalized discussions regarding services or rentals.

**Features Utilized:**

* + Click to Chat: Creates a seamless link for users to initiate a chat with another user’s WhatsApp account.
  + Message Templates: Pre-defined message formats that can be used for frequent communication scenarios.

Planned API Integration

1. **Tasdeeq.com API**

* Purpose (Planned): In the future, Ujret plans to integrate with Tasdeeq.com for robust user verification. This integration will ensure that users who provide services or rent items are verified, thereby increasing trust and security on the platform.

**Features to be Utilized:**

* + CNIC Verification: Confirms the identity of users by cross-referencing their provided CNIC (Computerized National Identity Card) numbers with the Tasdeeq database.
  + Background Checks: Optionally, further background checks may be conducted to ensure the reliability of service providers and lenders.

**API Management Considerations**

1. Security: Ensuring that all API calls are made over secure channels (HTTPS) and that sensitive data is encrypted.
2. Rate Limiting: Managing the number of requests made to these APIs to stay within usage quotas and avoid additional charges.
3. Error Handling: Implementing robust error handling to manage scenarios when the API is unavailable or returns an error response.
4. Data Privacy Compliance: Ensuring that the usage of APIs, especially those handling personal identification information, is compliant with data privacy laws and regulations.

This API Specification serves as a fundamental component of the Ujret Mobile App’s infrastructure, supporting essential features that contribute to the app’s functionality and user experience. As the platform evolves, the integration with additional APIs like Tasdeeq.com will further enhance the app’s capabilities and the value it offers to its users.

# System User Interface

The System User Interface chapter offers a comprehensive guide to the Ujret Mobile App's user interface, detailing the functionalities available to end-users within a visually intuitive and interactive environment. It focuses on the ease with which users can navigate through the Handyman and Renting modules, highlighting the thoughtful design decisions that contribute to a seamless user experience. Accompanied by supporting screenshots, this section will walk users through the various aspects of the application—from the simplicity of signing up as a service provider or seeker, to the convenience of posting and renting items. The interface is designed to be not only aesthetically pleasing but also accessible and efficient, ensuring users of all tech proficiencies can navigate the app confidently and complete their tasks with ease.

This sub-section should explain the functionality of your application to the end user with supporting screenshots.

**1.** **Onboarding Flow**

1. Splash Screen:

The first screen is the splash screen, which is the initial visual the users see when they open the Ujret Mobile App. It displays the app's logo and name, setting a professional tone and reinforcing brand recognition. This screen usually lasts a few seconds, giving the app time to load the main content.



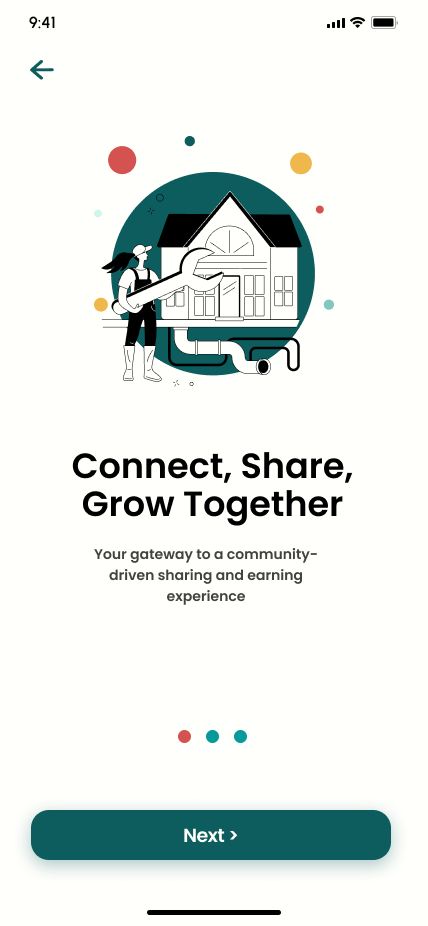
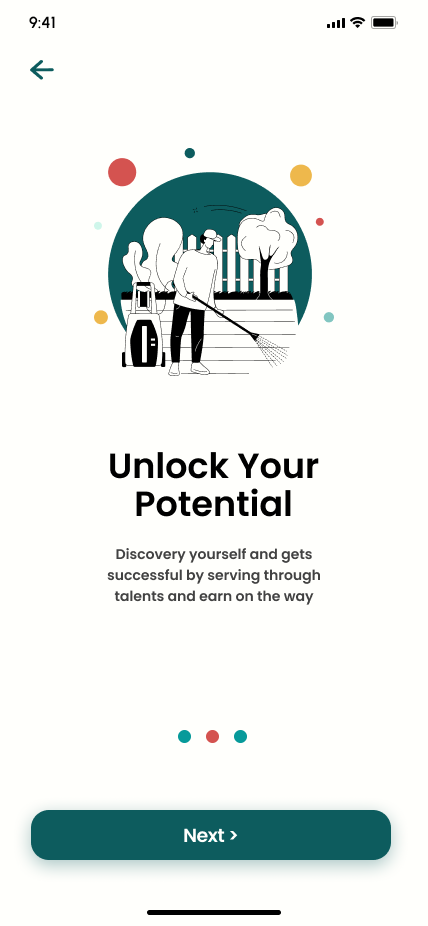
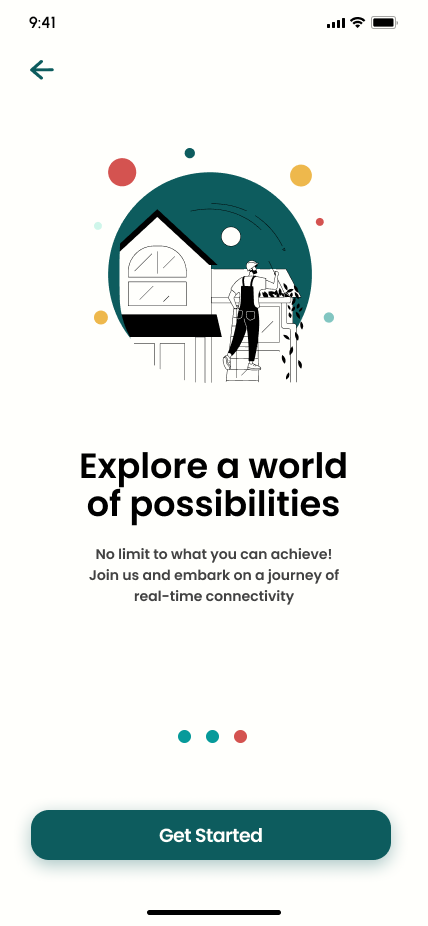
**2.** **Welcome Onboarding Screens:**

These screens are part of the onboarding process, designed to give users a brief overview of the app's purpose and benefits. Each screen showcases a different aspect:

The first onboarding screen (2.png) introduces the app's community-driven approach with an inviting tagline, "Connect, Share, Grow Together."

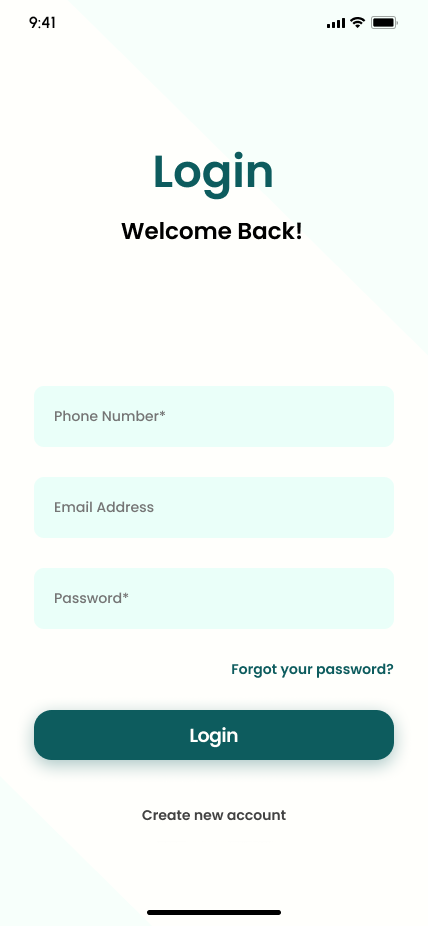
The second (3.png) encourages users to unlock their potential, hinting at the personal growth opportunities available through the app.

The third (4.png) invites users to "Explore a world of possibilities," suggesting a journey of discovery and connection within the platform. The "Get Started" button leads users to either log in or sign up.

3. **Login Screen (5.png)**

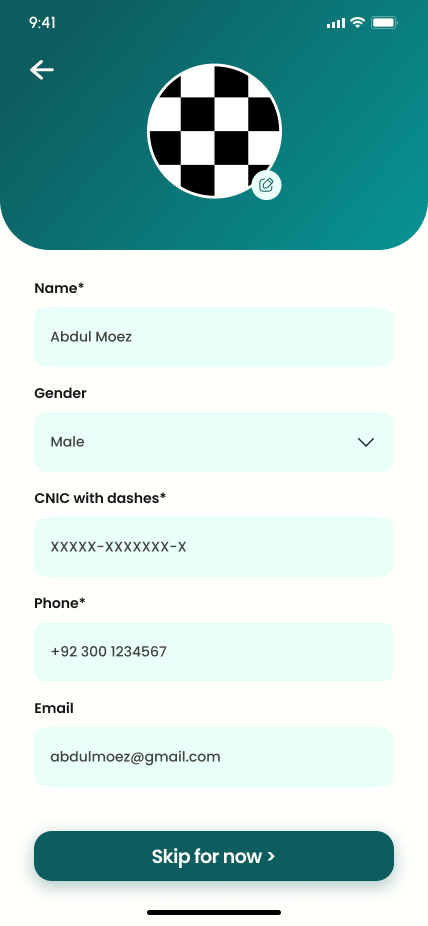
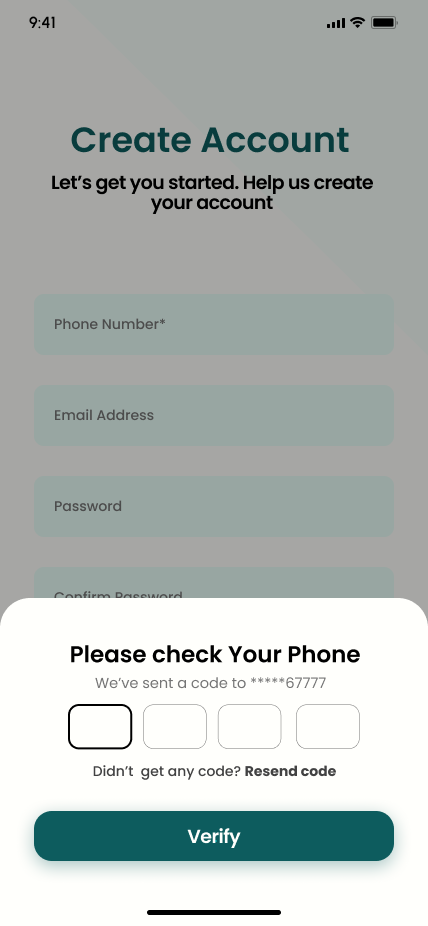
Here, returning users can enter their credentials, including phone number and password, to access their accounts. The screen also offers options for users who have forgotten their passwords or wish to create a new account, ensuring a smooth navigation path for different user needs.



**4.** **Account Creation Screens**

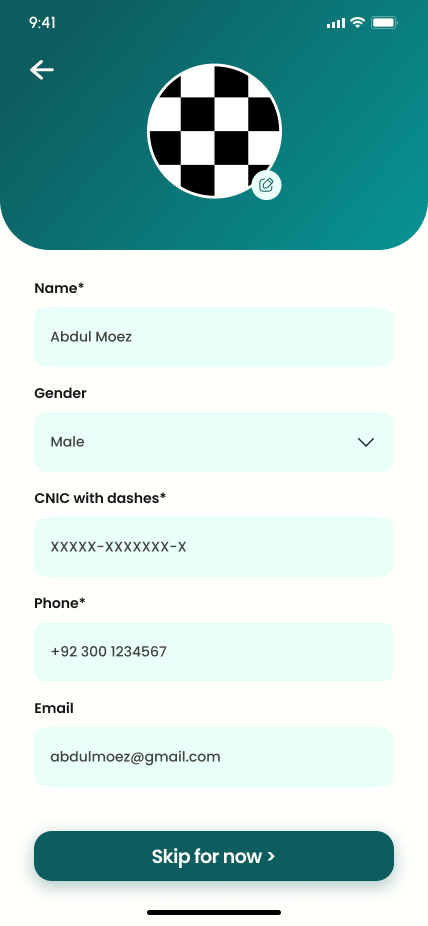
The first account creation screen prompts new users to provide essential details such as phone number and email, facilitating a personalized and secure account setup.

The following screen displays a confirmation step where users are required to enter a verification code sent to their phone, adding an extra layer of security to the account creation process.

5. **Profile Information Screen**

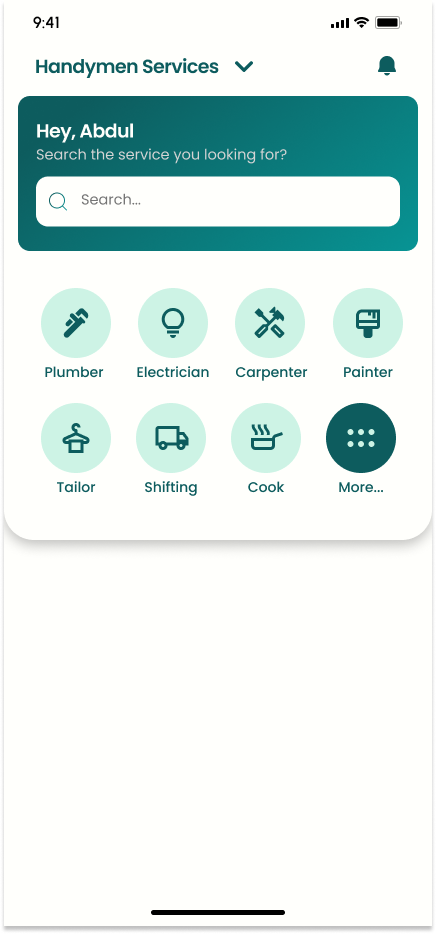
Once the account is created, users are prompted to fill out their profile information. This screen collects data like name, gender, CNIC (a form of national ID in some countries), and contact details, which are vital for building trust and transparency between users who will engage in service exchange or rentals.



**2.** **Task Seeker Side Flow**

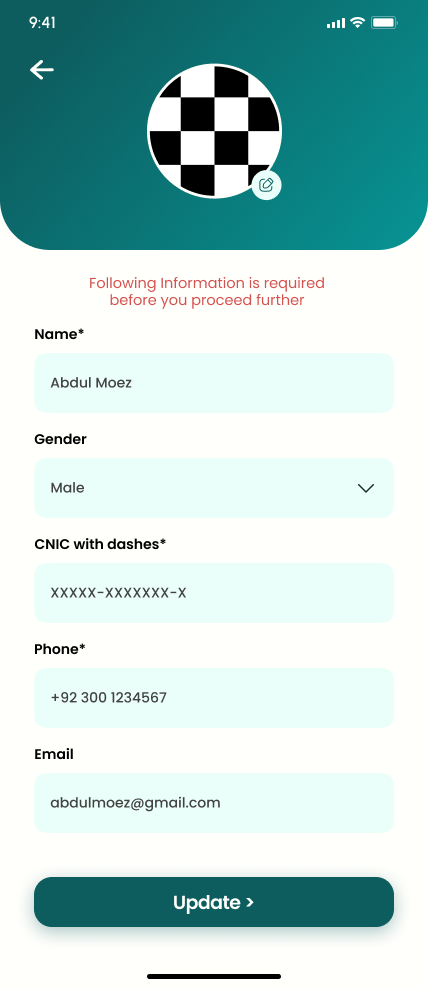
**2.1** **Handyman Home Screen (Handymen\_home\_screen.png)**

The home screen warmly greets the user by name, providing a personalized touch. It features a search bar at the top for quick access to specific services and showcases a grid of service categories like Plumber, Electrician, and Carpenter. This design enables users to easily navigate to the services they require.



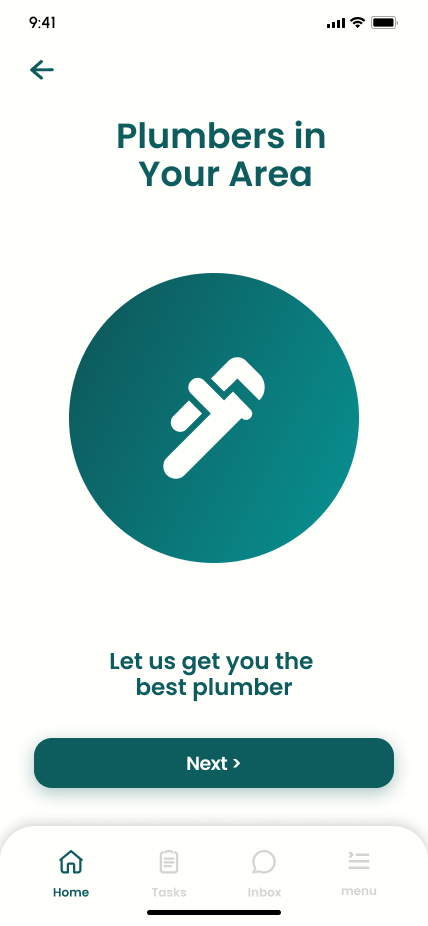
**2.2** **Registration - More Information (Register\_moreInfo.png)**

In this screen, users who are signing up are prompted to provide additional details like their name, gender, national ID (CNIC), phone number, and email. This information is crucial for creating a trusted profile within the community and ensuring the authenticity of users.



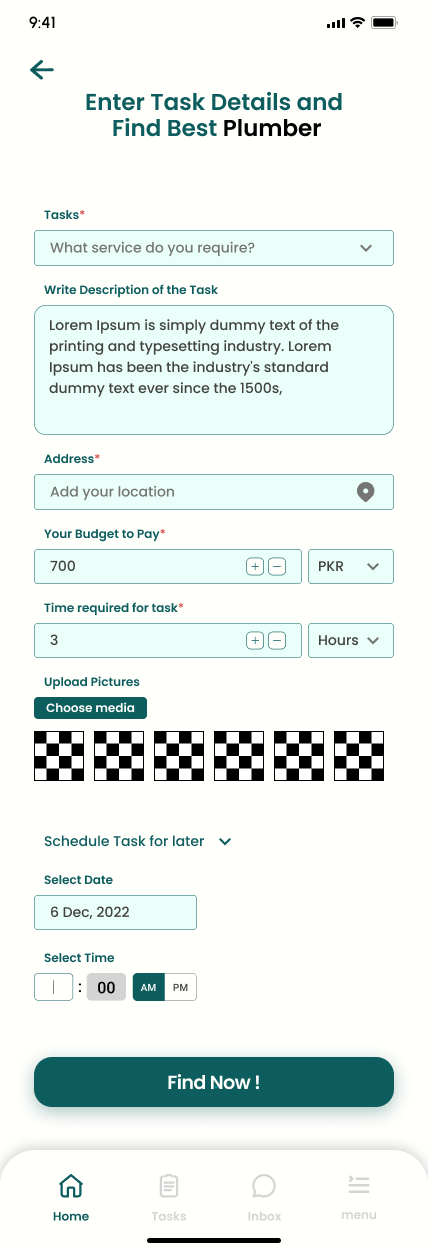
**2.3** **Service Category Selection (Service\_screen.png)**

When users select a specific service category, they're taken to a screen that reaffirms their choice ("Plumbers in Your Area") and encourages them to proceed ("Let us get you the best plumber"). The "Next" button guides them forward in the task posting process.



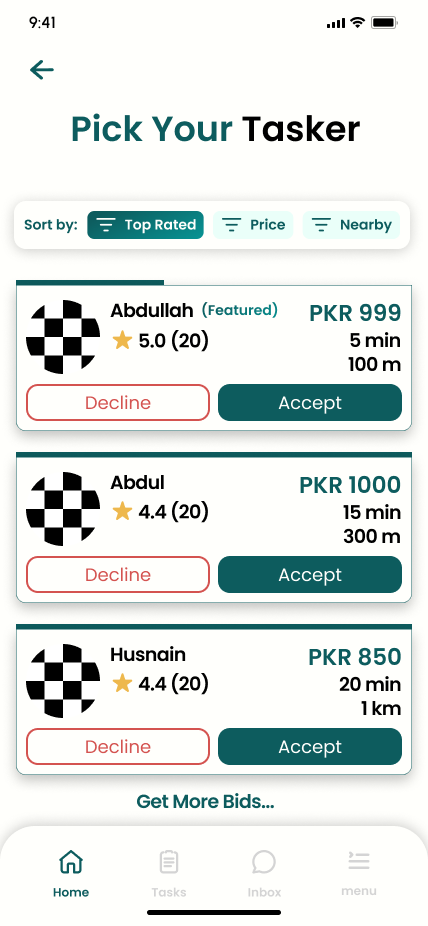
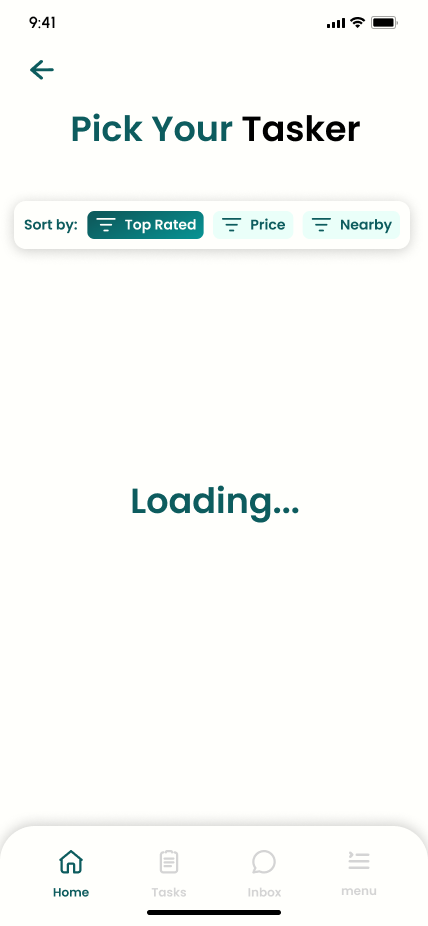
**2.4** **Task Details Submission (task\_details\_screen.png)**

This interface allows users to enter the specifics of the service they need, including a detailed description, location, budget, and the time required for the task. There’s also an option to upload pictures related to the task, providing service providers with clear information and expectations.



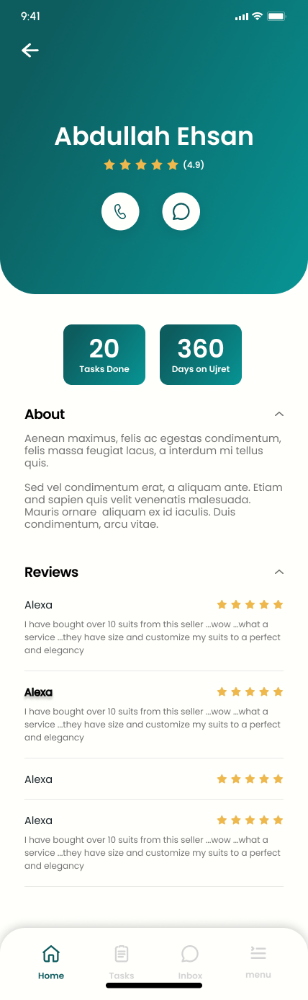
**2.5** **Task Offers from Service Providers**

After the task details are submitted, service seekers are presented with bids from available service providers. Each offer includes the provider's name, rating, proposed price, response time, and distance from the user's location, along with options to accept or decline the bid.

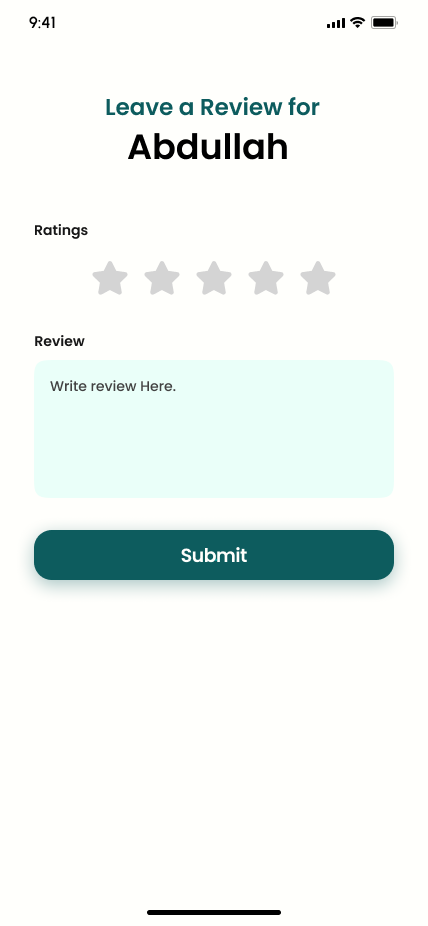
**2.6** **Service Provider Profile (tasker\_profile\_view\_screen.png)**

Before making a decision, service seekers can view detailed profiles of each provider, including their ratings, number of tasks completed, and user reviews. This screen helps in making an informed choice based on the provider's history and reputation.



**2.7** **Service Completion and Review (user-Add\_review.png)**

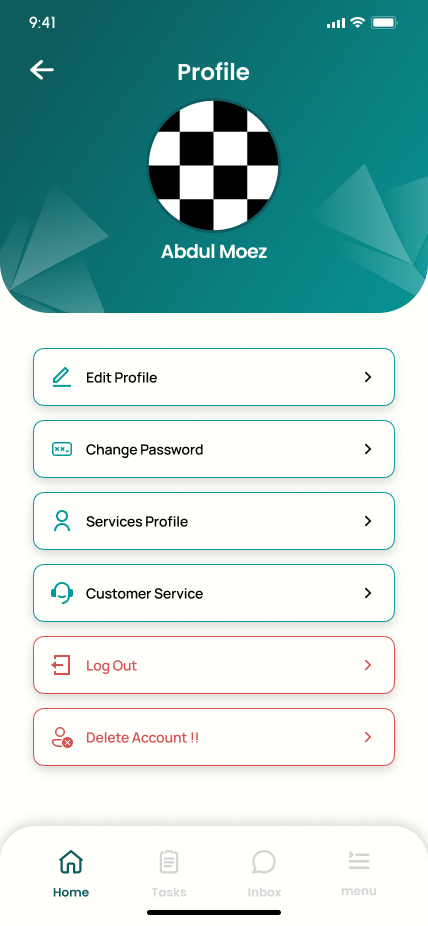
After the service is completed, users have the option to leave a review for the service provider. They can rate their experience and write a detailed review, contributing to the community-driven nature of the app by sharing their satisfaction level and feedback.



**3.** **Profile Management Flow**

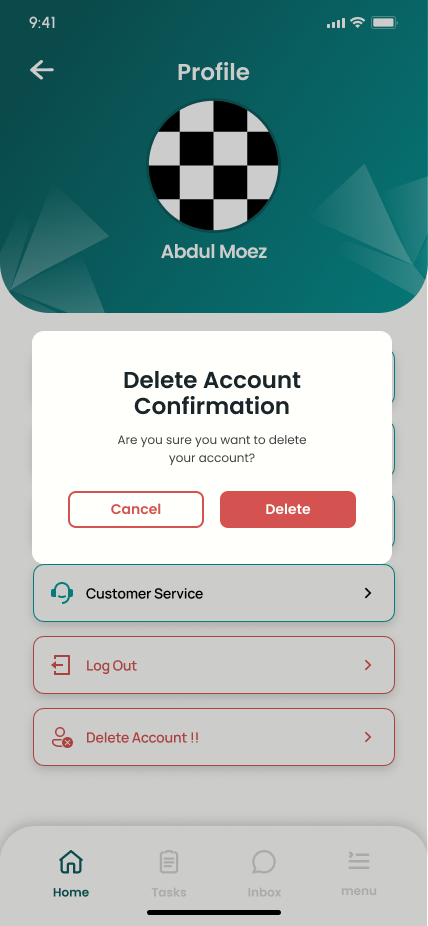
**3.1**  **Profile Management Main Screen**

This screen serves as the central hub for users to manage their account settings. It includes options to edit the profile, change the password, and access customer service. The profile picture, username, and navigational cues for additional actions like logging out or deleting the account are prominently displayed, reinforcing the app's commitment to user autonomy and privacy.



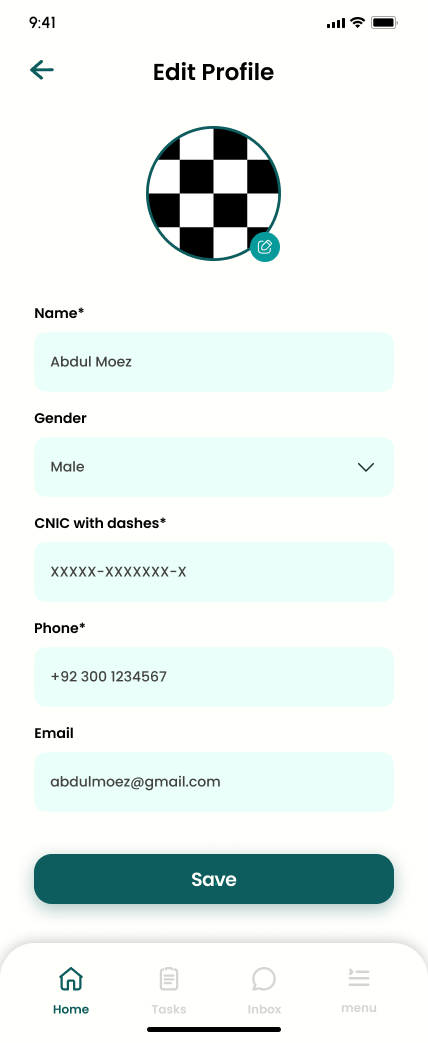
**3.2** **Delete Account Confirmation (2.png)**

When a user chooses to delete their account, a confirmation dialog box appears, asking them to confirm this irreversible action. This ensures that account deletion is intentional and prevents accidental loss of user data.



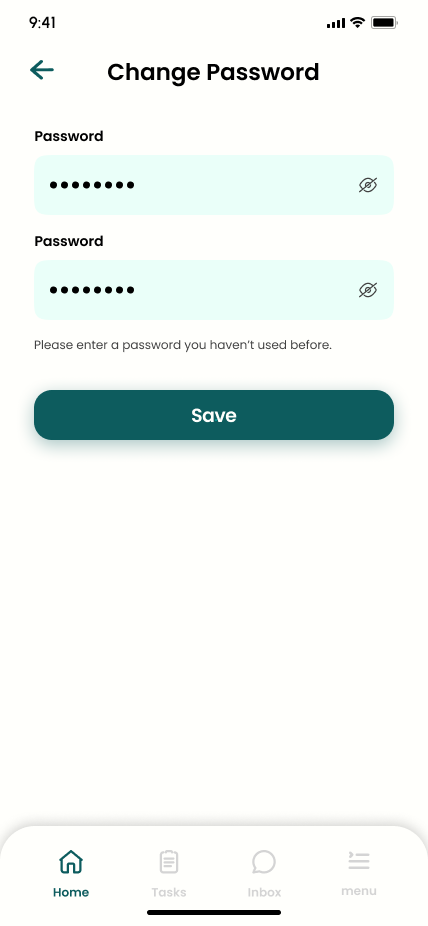
**3.3** **Profile Information**

The app allows users to edit their profile information, including name, gender, national ID, phone number, and email. This functionality is crucial for keeping user information up-to-date and ensures that the platform’s service is personalized and secure.



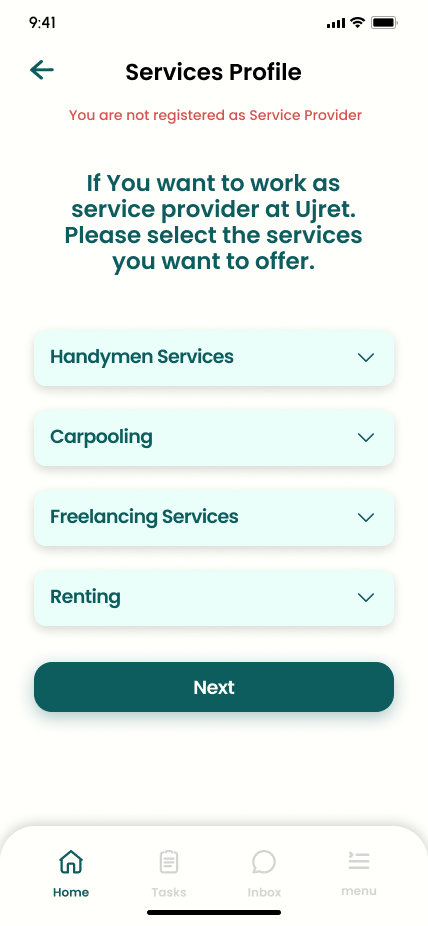
**3.4**  **Change Password Screen (5.png)**

Security is a top priority, and the Change Password feature enables users to update their password, adding an extra layer of security to their accounts.



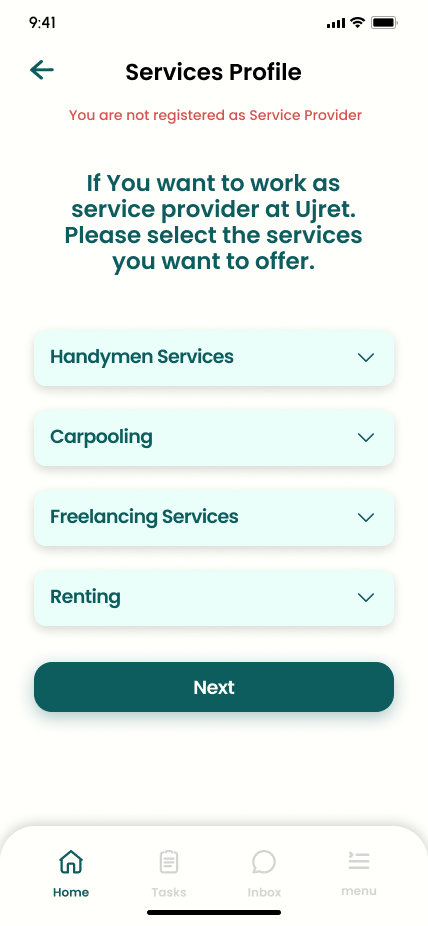
**3.5**  **Service Profile Selection (6.png)**

Users who wish to provide services can select their areas of expertise in this screen. This flexibility allows for a personalized service offering, catering to the diverse needs of the app's user base.



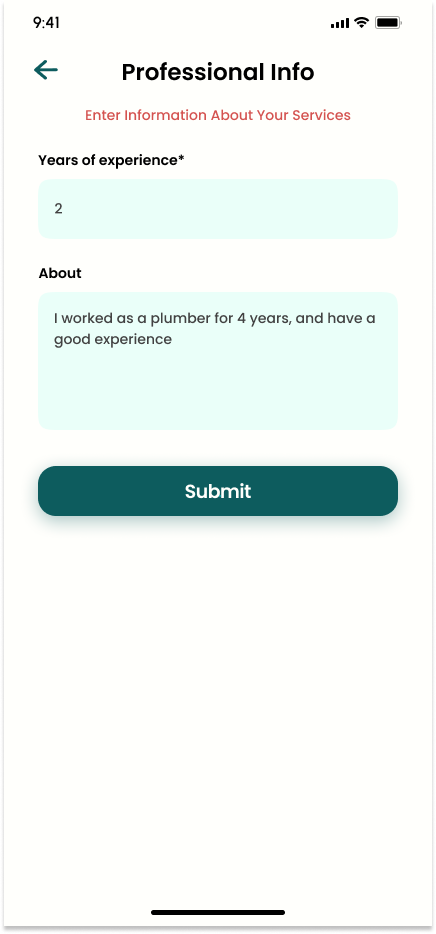
**3.6**  **Personal Information Confirmation (7.png)**

In this screen, users can confirm or correct their personal information, including their name and national ID, which is crucial for authenticity and trust on the platform.



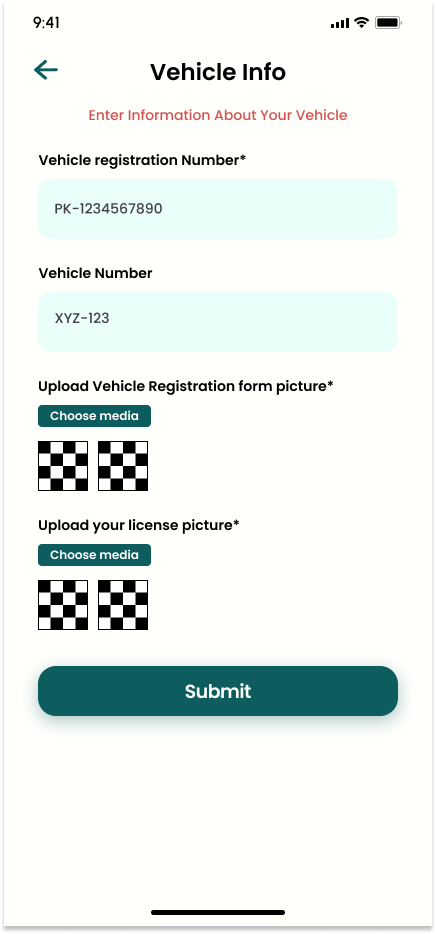
**3.7** **Professional Information Input (8.png)**

Service providers can add professional information, such as their years of experience and a brief bio, which helps them to present their qualifications to potential clients.



**3.8**  **Vehicle Information for Service Providers (9.png)**

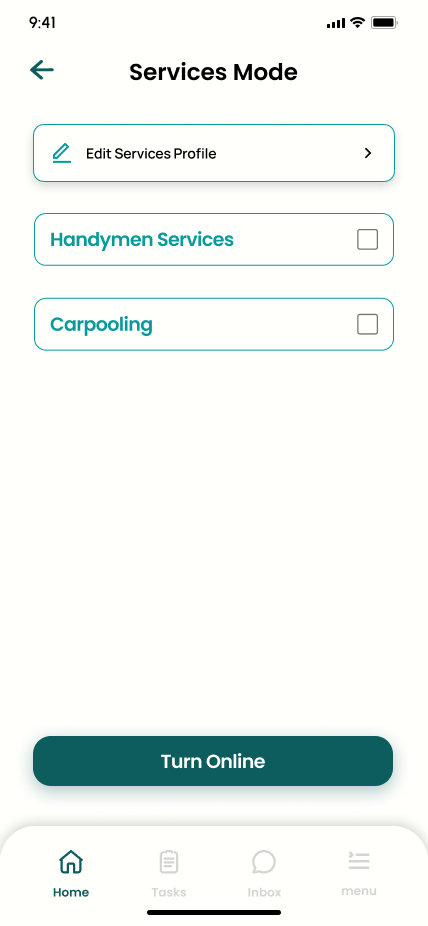
For services that require transportation, this screen allows users to enter details about their vehicle, ensuring that all information is transparent and verifiable.



**4.** **Task Provider Side Flow**

**4.1** **Service Selection Screen (1.png)**

The task provider starts their journey by selecting which services they wish to offer from a list of options such as Handyman Services, Carpooling, Freelancing, and Renting.

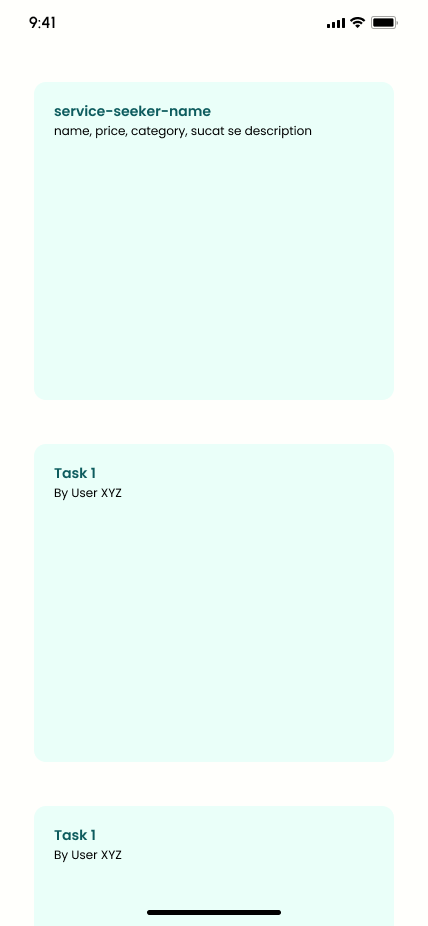


**4.2**  **Service Category Selection:**

Once the main service is selected, the provider can narrow down their offering by choosing specific sub-categories, enabling them to target their expertise effectively.

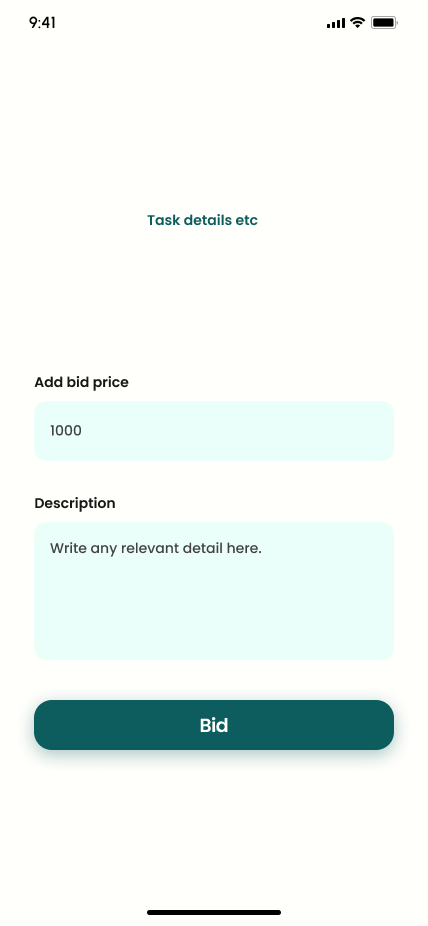
**4.3** **View Available Tasks (3.png)**

The provider is then presented with a list of available tasks within their selected category. This screen shows each task's details, such as the service seeker’s name, the price offered, and the task description, allowing them to choose which ones they’re best suited for.



**4.4** **Task Bidding (4.png)**

After selecting a task, the provider can place a bid on it. They enter their proposed price and any relevant details or terms into the bid submission form, which the service seeker will then review.

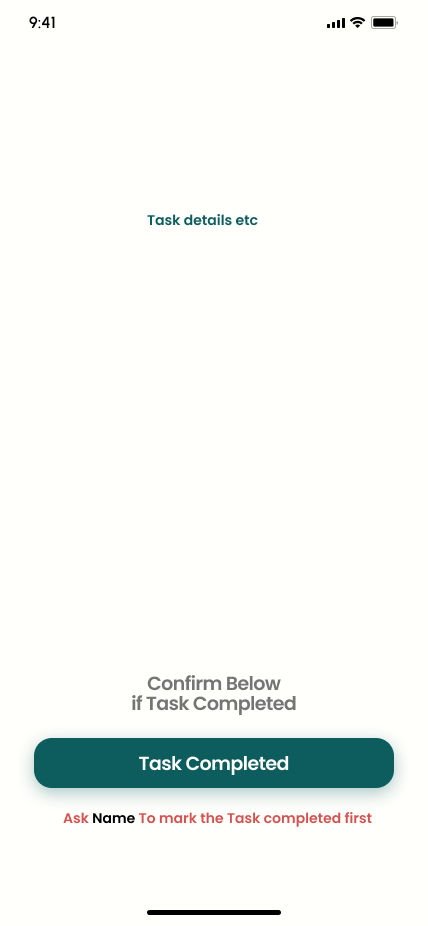


**4.5** **Bid Status Confirmation (5.png & 6.png)**

Post-bid submission, the provider can check the status of their bid—whether it’s been accepted or declined by the service seeker. This immediate feedback allows them to adjust their offers or look for other tasks as needed.

 4.5 **Task Completion Confirmation**

Once their bid is accepted, and the task is completed, the provider uses this screen to confirm task completion. This step may involve a final check or additional details to ensure that all aspects of the job are accounted for before it's officially marked as complete.



**Renting Module Flow**

* **Module Selection Screen:**

Users start by choosing the Renting module from the main dashboard, indicating their intention to either lend or rent items.

* **Role Selection Screen:**

Users specify their role: either as an 'Item Lender' to list items for rent or as an 'Item Renter' to browse available listings.

* **Available Items Listing Screen (for Renters):**

Item Renters are presented with a catalog of available items, which can be sorted or filtered by various criteria such as category, proximity, or price.Each listing includes a photo, brief description, rental price, and lender rating.

* **Item Detail Submission Form (for Lenders):**

Item Lenders fill out a form to list their item, including details such as the item's name, category, description, rental price, and photos.Lenders can also specify rental terms and availability.

* **Item Listing Confirmation (for Lenders):**

Once the form is submitted, lenders receive a confirmation that their item is listed, along with information about how to manage their listings.

* **Direct Connection via WhatsApp:**

Both Renters and Lenders can connect through a direct WhatsApp link for further communication regarding item availability, rental conditions, and transaction arrangements.

* **Rating and Review Screen (Post-Rental):**

After the rental period ends, both parties are prompted to rate and review each other.

This screen will include a star-rating system and a text box for a written review to ensure feedback is comprehensive and helpful to other users.

# Project Security

The Project Security chapter is dedicated to detailing the robust security measures and protocols integrated within the Ujret Mobile App to safeguard user data and interactions. This section highlights the strategic approaches taken to mitigate risks, protect privacy, and ensure the integrity and reliability of the platform. It covers everything from data encryption and secure authentication processes to compliance with international security standards and regular security audits. The chapter aims to provide transparency into the security infrastructure of the app, offering users and stakeholders peace of mind and reinforcing the app's commitment to creating a secure and trustworthy digital environment. Through this chapter, readers will gain insight into the layers of security that make the Ujret Mobile App a safe choice for users engaging in handyman and renting services.

**Identifying Security Threats**

1. Unauthorized Access:

* + Vulnerability: User accounts are a primary target for unauthorized access attempts, particularly through weak authentication mechanisms. Given the platform's functionality, access to user profiles, financial details, and private communications are highly sensitive.
  + Attack Vectors: Phishing, brute force attacks, and exploitation of software vulnerabilities.

**2. Data Interception and Theft:**

* Vulnerability: Data in transit can be intercepted, and sensitive information like personal identification numbers, contact details, or financial data can be stolen.
* Attack Vectors: Man-in-the-middle attacks, spyware/malware, unsecured Wi-Fi networks.

**3. Data Leakage:**

* Vulnerability: Inadequate data storage and disposal practices can lead to unintentional data leaks, including user data and proprietary business information.
* Attack Vectors: Misconfigured databases, improper data disposal methods, accidental leaks via employee error.

**4. Service Disruption (DoS/DDoS Attacks):**

* Vulnerability: The platform's availability to users can be threatened by denial of service attacks, disrupting the service and eroding trust.
* Attack Vectors: DDoS attacks exploiting network vulnerabilities or application flaws.

**5. Code Injection:**

* Vulnerability: The application’s input fields could be vulnerable to code injection attacks, compromising the underlying databases and systems.
* Attack Vectors: SQL injection, Cross-site Scripting (XSS), Command injection.

**6. Repudiation:**

* Vulnerability: Disputes over the authenticity of transactions or communications can arise due to the lack of proper tracking and logging.
* Attack Vectors: Forged requests, denial of actions without proper logging mechanisms.

**7. Malware and Ransomware:**

* Vulnerability: Users' devices could become infected with malware or ransomware through the platform, particularly if they are tricked into downloading malicious content.
* Attack Vectors: Infected files or links distributed via the app, exploitation of software vulnerabilities.

**Potential Losses**

For each identified risk, potential losses include:

* Financial Loss: Unauthorized access or theft can lead to direct financial loss through fraud or theft of funds.
* Reputation Damage: Data breaches can severely damage the app’s reputation, leading to a loss of user trust and subsequent decline in user base.
* Operational Disruption: Service disruption attacks can halt operations, leading to a loss of service availability and business continuity issues.
* Legal Consequences: Failure to protect user data can result in litigation, fines, and penalties from regulatory bodies.
* Intellectual Property Theft: Code injection and malware can lead to the theft of proprietary algorithms and business logic.
* Loss of Integrity: Repudiation risks can undermine the integrity of transactions, eroding trust in the platform's reliability.

**Security Controls**

**Protective Controls:**

* Multi-Factor Authentication: Protects against unauthorized access by requiring multiple forms of user verification.
* Data Encryption: Encrypts data both at rest and in transit to prevent interception and theft.
* Input Validation: Ensures that all input is validated to protect against code injection attacks.
* Regular Software Updates: Keeps the system secure by patching known vulnerabilities.

**Detective Controls:**

* Audit Logs: Tracks user activities to detect unauthorized actions and support non-repudiation.
* Anomaly Detection Systems: Identifies patterns that may indicate a security breach.

**Responsive Controls:**

* Intrusion Detection Systems: Identifies and responds to unauthorized access or attacks in real-time.
* Real-Time Alerts: Notifies administrators of suspicious activities, allowing for immediate action.

**Recovery Controls:**

* Data Backups: Regular backups ensure that data can be recovered following a loss event.
* Disaster Recovery Plan: Outlines steps to restore services and data after a major security event.

**Static and Dynamic Security Scanning Tools**

**Static Application Security Testing (SAST):**

* Tool Selection: SonarQube is a comprehensive tool that supports multiple languages and can be integrated into the development pipeline for continuous inspection of code quality and vulnerabilities.

**Dynamic Application Security Testing (DAST):**

* Tool Selection: OWASP ZAP is an open-source option that provides automated scanning of web applications to find security vulnerabilities during the testing phase.

Both tools are selected for their compatibility with a wide range of programming languages and their community support, ensuring they stay up-to-date with the latest vulnerability databases and security practices. They will be integral in the development lifecycle to continuously assess and ensure the security of the Ujret Mobile App.

| **Sr#** | **Security Risks** | **Potential Losses** | **Controls** |
| --- | --- | --- | --- |
| 1 | **Injection flaws**, such as SQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. This can trick the interpreter into executing unintended commands, leading to data loss or exposure. | Unauthorized data access, data corruption, financial losses from fraudulent transactions, and tarnished reputation. | 1. Validate and sanitize all user inputs to ensure they do not contain malicious code or special characters. 2. Employ web application firewalls (WAFs) to detect and block injection attacks. WAFs can help filter out malicious input before it reaches your application. |
| 2 | **Broken Authentication:**  If authentication mechanisms aren't implemented correctly, attackers can impersonate other users, leading to unauthorized access. | Data breaches, unauthorized actions on behalf of users, and financial losses. | Implement multi-factor authentication.  Ensure session timeouts and automatic logouts.  Encrypt all passwords using strong cryptographic algorithms. |
| 3 | **Cross-Site Scripting (XSS):**  XSS flaws occur when an application includes untrusted data on a web page without proper validation, allowing attackers to execute malicious scripts in a user's browser. | Session hijacking, defacement of websites, distribution of malware. | Validate, sanitize, and escape all user inputs.  Implement content security policies to prevent unauthorized script execution. |

# Risk Management

## Potential Risks and Mitigation Strategies

| **Sr.** | **Risk Description** | **Mitigation Strategy** |
| --- | --- | --- |
|  | Unauthorized Account Access | Implement multi-factor authentication and regular password audits to verify the strength of user passwords. |
|  | Data Interception During Transmission | Utilize end-to-end encryption protocols such as TLS for all data in transit. |
|  | Injection Attacks (SQL, XSS, etc.) | Conduct thorough input validation, use prepared statements for database access, and implement content security policies. |
|  | Denial of Service (DoS) Attacks | Employ a web application firewall (WAF), rate limiting, and scalable infrastructure to absorb traffic spikes. |
|  | Insecure APIs | Enforce strict access control and rate limiting, and perform regular security audits of all API endpoints. |
|  | Data Leakage from Storage | Encrypt all sensitive data at rest, employ database access controls, and use secure data disposal practices. |
|  | Insider Threats | Limit privileges through role-based access control systems and monitor activity logs for unusual actions. |
|  | Phishing Attacks | Educate users about the dangers of phishing, use domain authentication methods, and implement email filtering solutions. |
|  | Code Vulnerabilities | Adopt a secure coding standard, perform regular static and dynamic code analysis, and keep all third-party libraries updated. |
|  | Misconfiguration of Cloud Services | Automate cloud configurations with Infrastructure as Code (IaC) to ensure consistency and employ regular configuration audits. |

# Testing and Evaluation

**Testing and Evaluation: Frontend Testing Strategy**

Overview

Our manual testing strategy for the Ujret Mobile App's frontend was meticulously planned to ensure the application's user interface and overall user experience met our high standards for quality, functionality, and performance. Manual testing allowed our team to adopt the perspective of an end-user and interact with the application as our customers would, providing invaluable insights that automated tests might overlook.

**Testing Strategy**

* Functional Testing:

The core of our manual testing involved rigorous checks of all functional elements of the app. Each button, link, form, and interactive element was tested for proper operation.

* **Usability Testing:**

A diverse group of users, including those with varying levels of technical expertise, was invited to interact with the app. Their feedback was critical in assessing the app’s navigability, understandability, and overall user satisfaction.

**Interface Testing:**

* We paid close attention to the interface elements such as menus, buttons, and icons. We ensured that they were not only aesthetically pleasing but also accessible and responsive across different devices and screen sizes.

**Consistency Testing:**

* Consistent behavior across various sections of the app was tested, ensuring that themes, fonts, color schemes, and response messages remained uniform throughout.

**Error Handling Testing:**

* Scenarios were simulated to check the app’s ability to handle errors gracefully. This included testing with invalid input, unexpected user behavior, and network interruptions.

**Compatibility Testing:**

The application was tested across multiple devices and operating system versions to ensure compatibility and identify any device-specific issues.

**Sample Test Cases**

Here are examples of some of the test cases that we developed for manual frontend testing:

**Test Case: User Registration Process**

* Description: Verify that new users can successfully register.
* Steps: Navigate to the sign-up page, fill out the registration form, submit, and confirm account creation.
* Expected Result: The user should receive a confirmation notification and be directed to the login screen.

**Test Case: Search Functionality**

* Description: Validate the search functionality returns accurate results.
* Steps: Enter a query in the search bar and execute the search.
* Expected Result: Relevant results based on the entered query should be displayed.

**Test Case: Task Posting Flow**

* Description: Confirm that a user can post a task without errors.
* Steps: Log in, navigate to the task posting form, fill out all fields, and submit.
* Expected Result: The task should be posted, and a confirmation message should appear.

**Test Case: Profile Editing**

* Description: Ensure that users can edit and save their profile information.
* Steps: Go to the profile settings, change several fields, save changes.
* Expected Result: Changes are reflected immediately, and persist upon reloading the app.

Automation Tools Used

While our main focus was on manual testing for the frontend, we did incorporate some automation to complement our efforts:

* Selenium: Used for automating web application testing to verify that it works as expected.

Through this combined approach of detailed manual testing and selective automated testing, we aimed to cover all bases - verifying not only the functionality and reliability of the app but also ensuring a high-quality user experience that aligns with our users’ needs and expectations.

**Backend Testing Strategy**

Overview

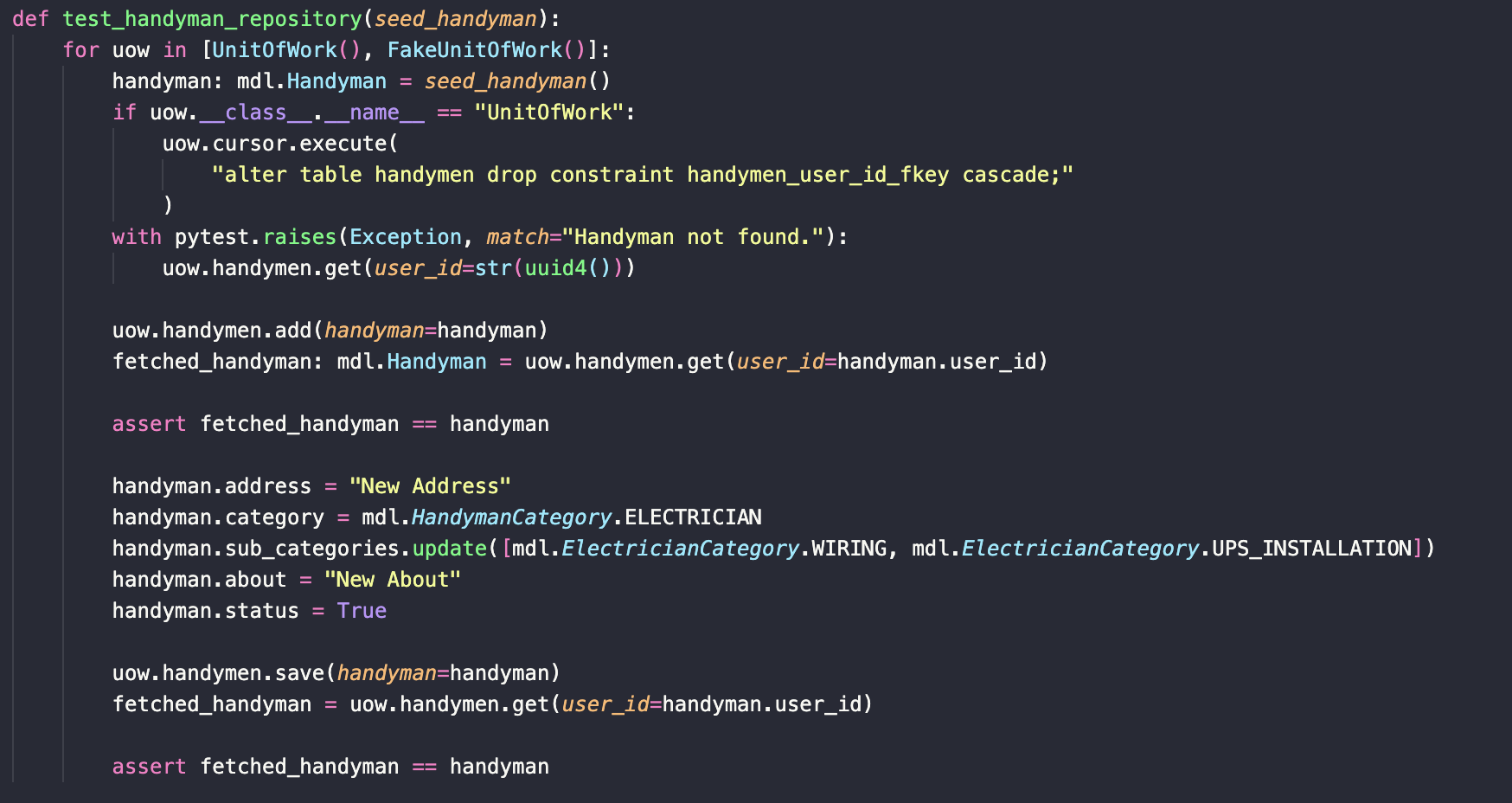
Testing the backend of the Ujret Mobile App required a multifaceted approach to ensure that all systems operated as expected. We undertook thorough testing to validate the integrity, security, and functionality of our APIs which serve as the backbone of our application. Our strategy integrated both automated and manual testing methods to cover a broad spectrum of scenarios and use cases.

**Comprehensive Testing with Pytest**

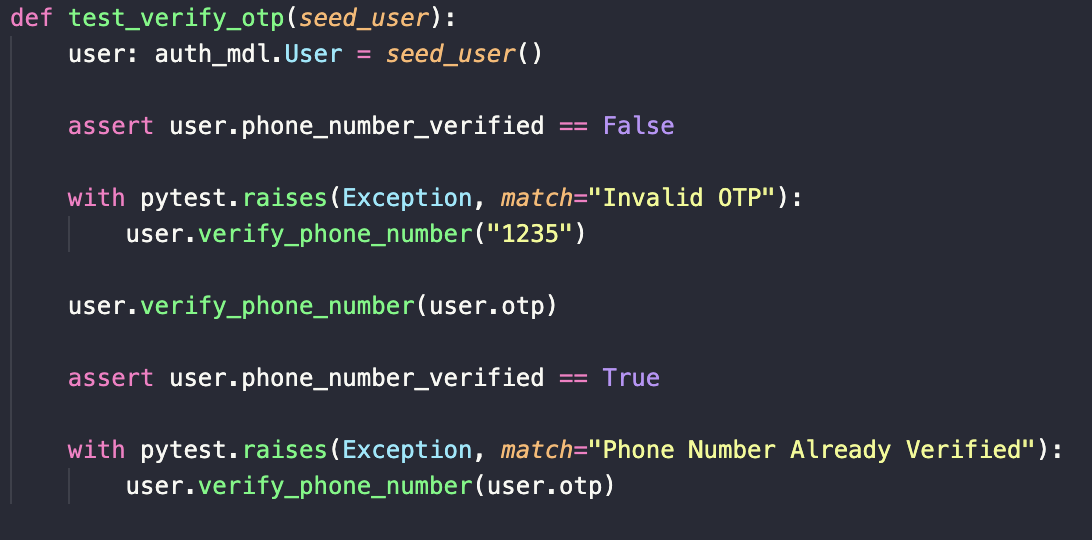
The tests are divided between model tests, repository tests and command level tests. A testing configuration file (conf.py) helps in creating instances of our entities and value objects from our domain model. We use these to perform tests on all our layers.



The add/get/save logic testing has been incorporated in the repository.py files of each model. These tests ensure that database operations on our objects are happening correctly with no errors.



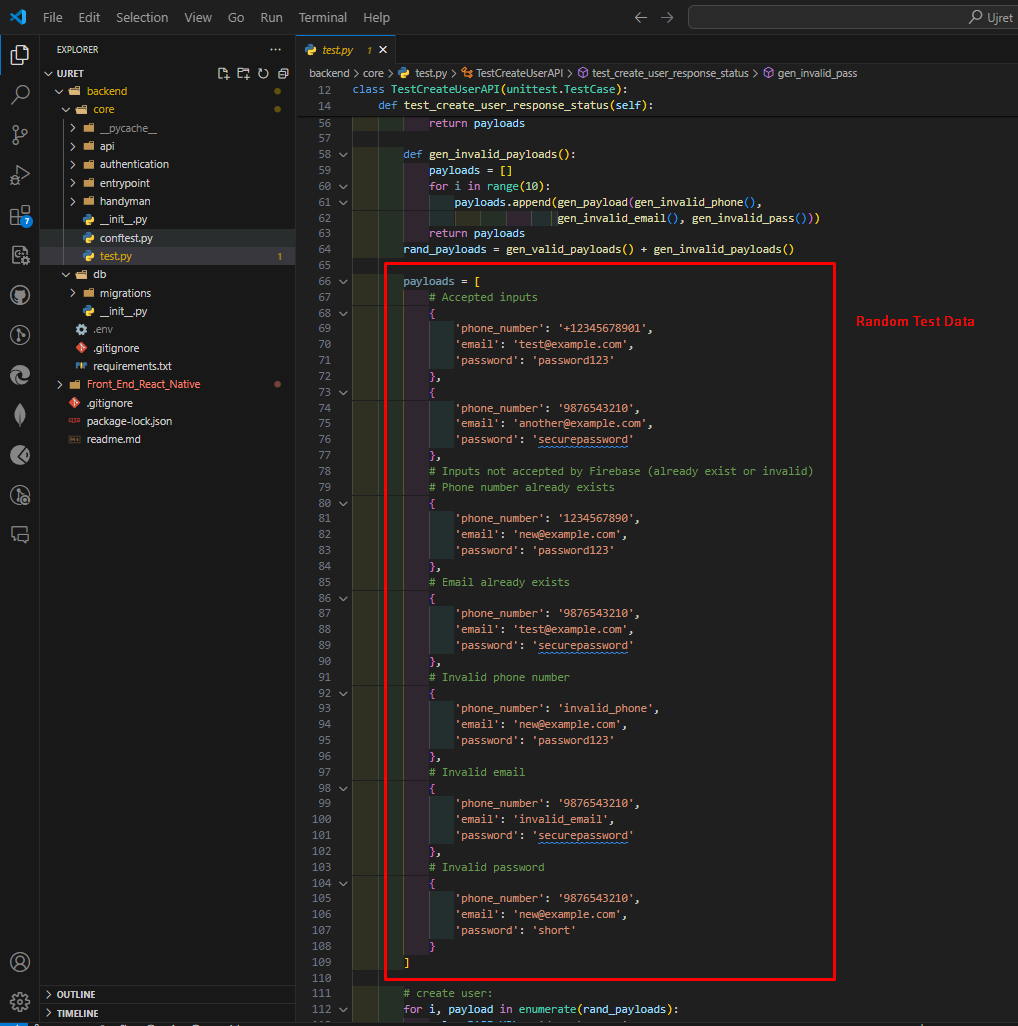
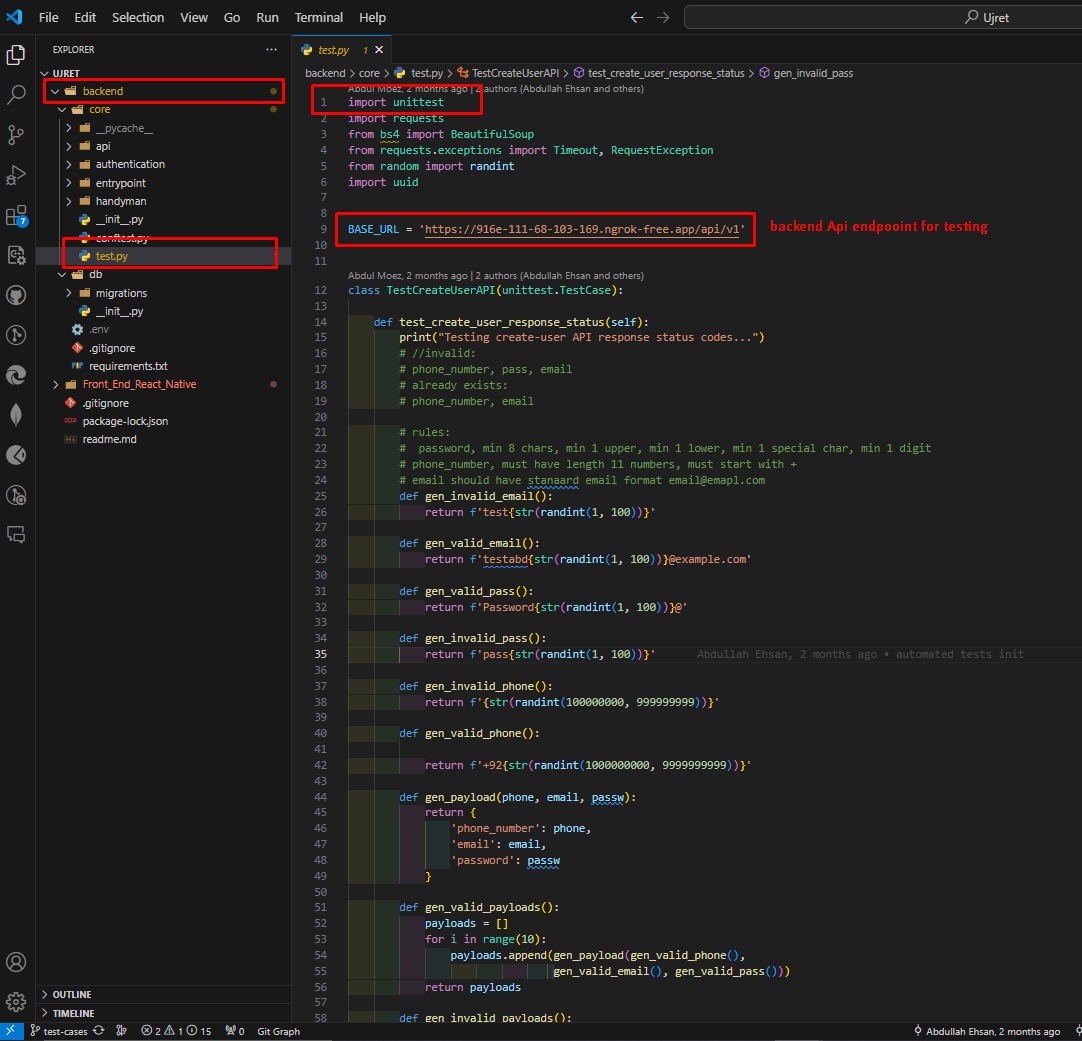
The model tests ensure that the core business logic of our domain is functioning correctly along with expected mutations.

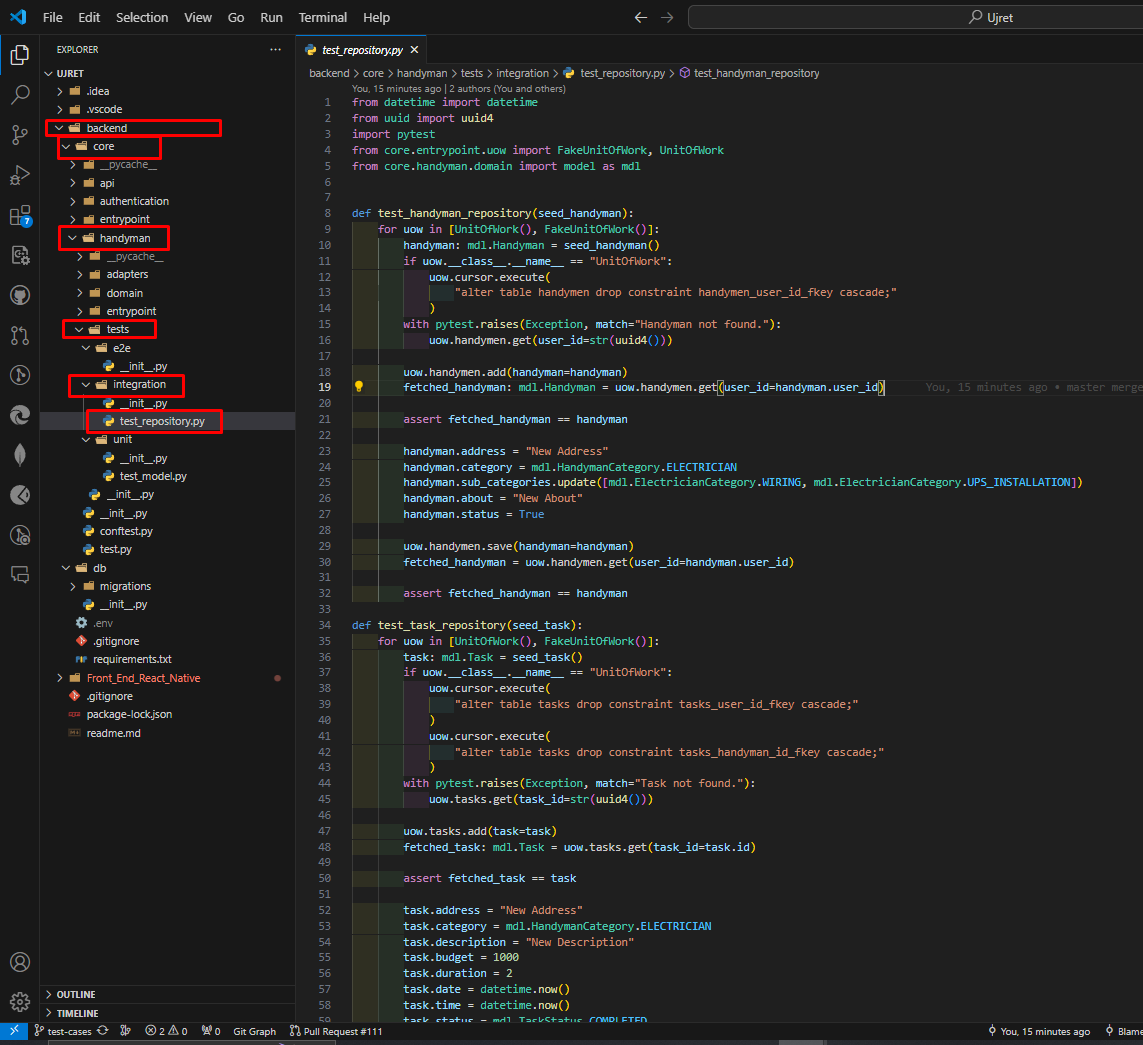
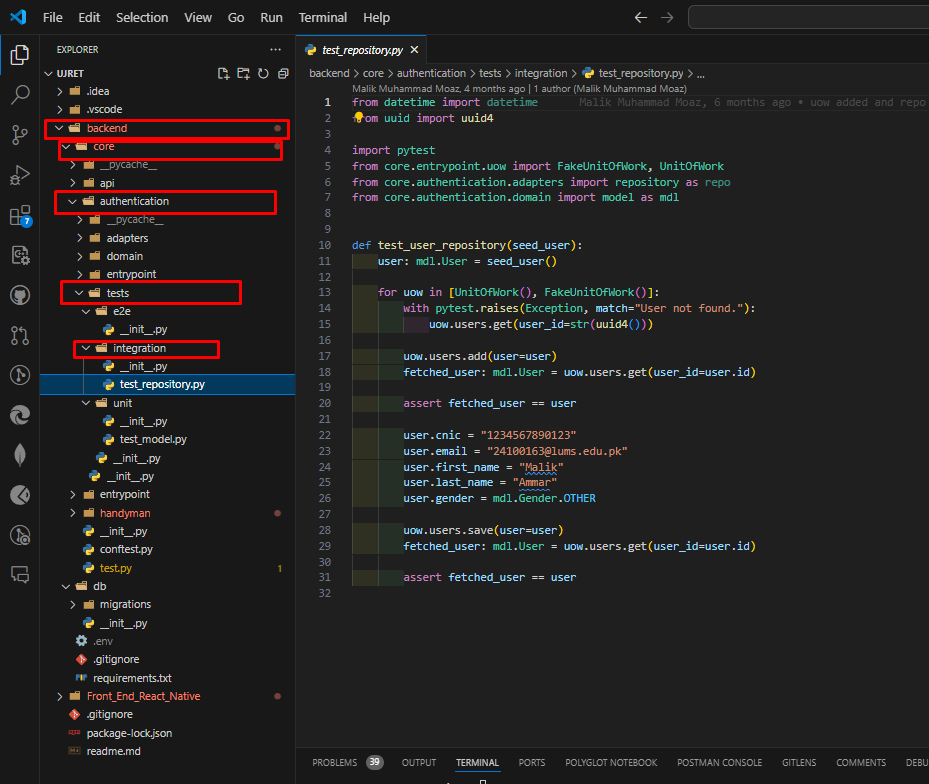


Unit Testing with Python unittest

We utilized Python's unittest library to construct and run our suite of automated test cases. This allowed us to validate individual units of code in isolation, ensuring that each component behaved correctly under various conditions.

* Test Setup: We configured a test environment mirroring our production settings, which included setting up a BASE\_URL pointing to our test API endpoint.
* Test Execution: By executing the test.py file while the backend was running, we could perform real-time testing against our API endpoints.
* Test Cases: These ranged from user creation, data validation, error handling, to task creation and management.
* Assertions: We wrote assertions to check the responses of API calls against expected outcomes. This included status codes, returned data, and error messages.
* Coverage: We aimed for high coverage across our models, views, and utility functions, running tests for typical, edge, and failure cases to simulate a range of user interactions.

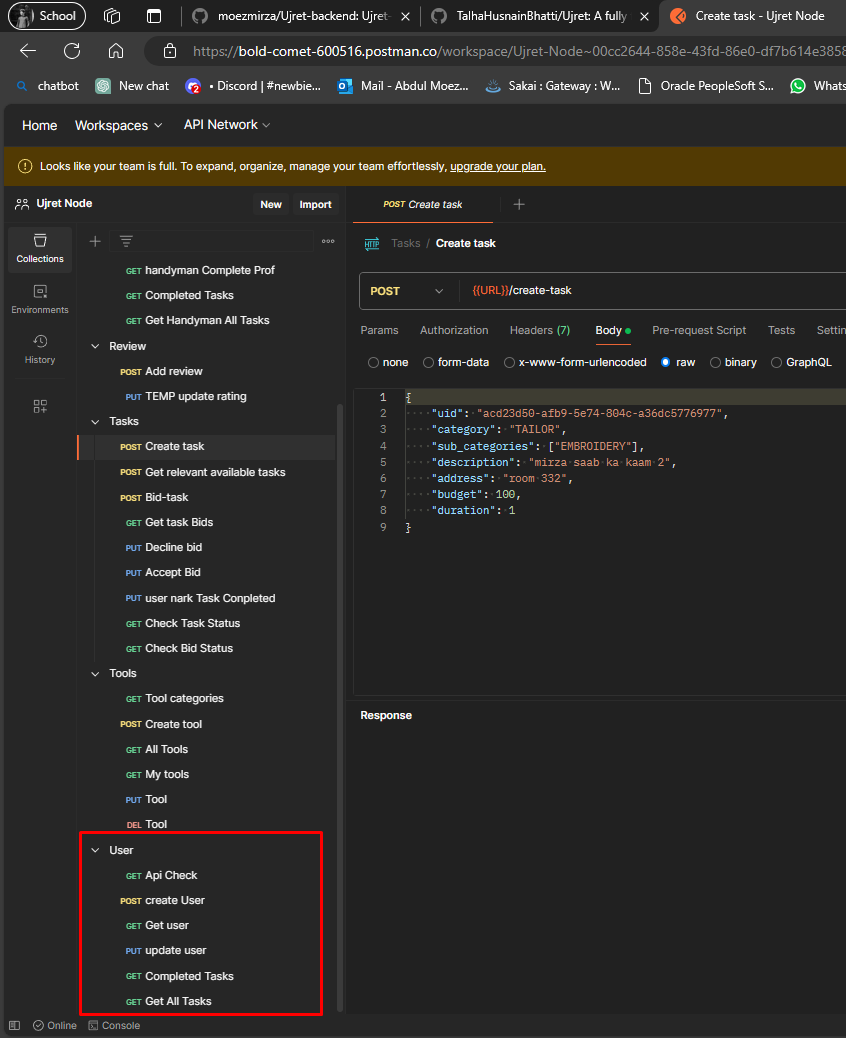
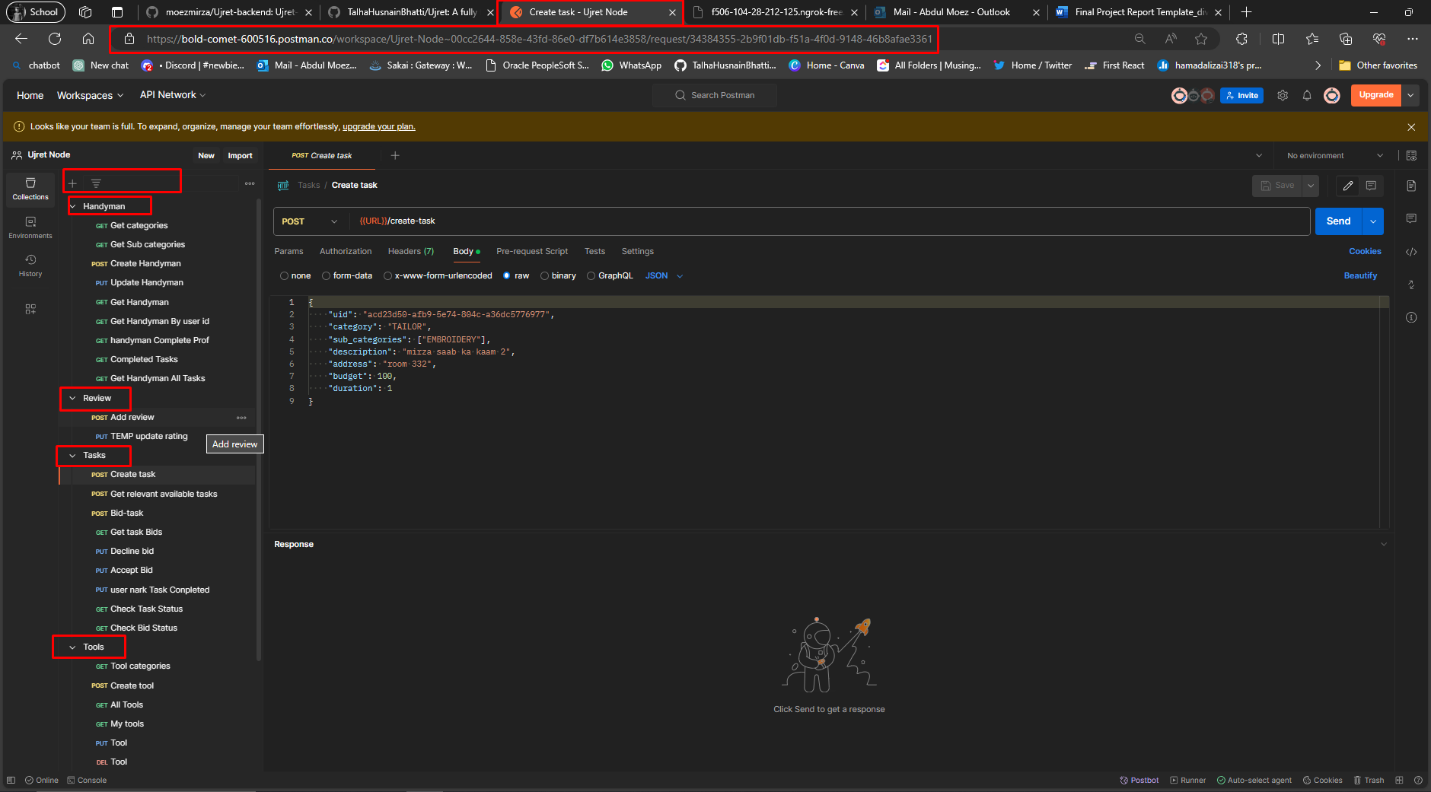




**API Endpoint Testing with Postman**

For manual testing, we leveraged Postman for its robust interface and user-friendly features that allowed us to send requests to our Node.js backend and examine the responses.

* Collection Organization: We organized our tests into collections reflecting the structure of our application, such as User, Handyman, and Tasks.
* Environment Variables: To maintain flexibility and ease of configuration, we stored our API endpoint as an environment variable {{URL}} in Postman.
* Manual Test Execution: For each endpoint, we composed requests to cover a variety of scenarios, including successful calls and those that should fail (e.g., submitting invalid data).
* Response Validation: After sending requests, we manually checked the responses to ensure they contained the correct data and that HTTP status codes were appropriate for each situation.



Detailed Example:

User Creation Testing

Objective: To verify the API endpoint's capability to register a new user with valid credentials and to handle errors appropriately.

**Test Case 1: Successful User Creation**

* Description: Send a POST request with a valid payload including name, email, and password.
* Expected Outcome: The API should return a 200 OK status with a success message and user ID.

**Test Case 2: Invalid Email Address**

* Description: Attempt to create a user with an improperly formatted email address.
* Expected Outcome: The API should reject the request and return a 400 Bad Request status with an error message detailing the invalid email format.

**Test Case 3: Duplicate User**

* Description: Submit a new user creation request using an email that already exists in the system.
* Expected Outcome: The API should return a 409 Conflict status with a message indicating the account already exists..

Static Analysis Tools: To complement our testing, we incorporated static analysis tools like ESLint for JavaScript and Pylint for Python to catch potential issues before runtime.

Through this comprehensive testing approach, we aimed to ensure that every aspect of the Ujret backend was scrutinized for potential faults and that any issues were caught early in the development cycle. This commitment to thorough testing underpins the reliability and robustness of the Ujret Mobile App.

# Deployment Guidelines

1. GitHub Repository:

* GitHub Repository Link: <https://github.com/TalhaHusnainBhatti/Ujret>.

2. Continuous Deployment:

* The deployment process is automated using Render. Every code push to the relevant directory triggers an automatic deployment.
* Both our backend services (Flask and NodeJS) are deployed on Render

3. Flask Server Deployment:

* Flask API URL:<https://ujret-python-api.onrender.com/api/v1>
* No authentication information required

4. Node.js Server Deployment:

* Node.js API URL:<https://ujret-backend-node.onrender.com/api/v1>
* No authentication information required

5. Deployment Steps:

* Clone the GitHub repository to your local environment if you haven't already.
* Make necessary changes to the codebase.
* Push the changes to the main branch of the GitHub repository.
* Add repository on render and choose relevant branch for deployment (main in our case)
* Input appropriate build commands (mentioned in readme)
* After every push, render will automatically detect the changes and trigger deployment to the Flask and Node.js servers.
* We monitor the deployment process through the Render logs
* Once the deployment is completed, the updated application will be accessible through the provided URLs.

# Conclusion

In this sub-section, we will provide a summary of the project approach and the key learning outcomes from the development of the Ujret Mobile App. This project was designed to cater to the growing needs in the Consumer-to-Consumer (C2C) market, specifically focusing on handyman services and item rentals. Here is an overview of how we approached the project and the valuable insights gained through its execution:

**Project Approach**

1. **Initial Planning and Scope Definition**:

The project commenced with defining the scope based on identified market needs and potential user demand. Originally planned to include both handyman and carpooling modules, the scope was later refined to focus exclusively on handyman services and a new renting module. This adjustment was made to better align with user demand and market dynamics.

**2. Requirement Gathering:**

Detailed requirement sessions were conducted involving stakeholders and potential users to gather functional and non-functional requirements. This phase was crucial in understanding the precise needs of the users and the technical capabilities required to support those needs effectively.

**3. Design and Prototyping:**

With the requirements in place, the team moved to the design phase, where user interfaces and experiences were crafted. Prototypes were developed using tools like Figma to visualize the interaction flow and receive early feedback from test users.

**4. Development and Implementation:**

The development phase was executed in iterative cycles, adhering to Agile methodologies. This approach allowed for flexibility in handling project adjustments and incorporating feedback efficiently. The technology stack included React Native for frontend development, Flask and Node.js for the backend, and PostgreSQL and MongoDB for databases.

**5. Testing and Quality Assurance:**

Rigorous testing phases included manual and automated tests to ensure the application met all functional requirements and quality standards. Use case testing, unit testing with Python's unittest library, and API testing using Postman were key components of our QA process.

**6. Deployment and Monitoring:**

Deployment involved setting up the application on cloud services with continuous integration/continuous deployment (CI/CD) pipelines to manage releases smoothly. Post-deployment, the application was monitored actively to ensure operational stability and performance.

**7. User Feedback and Iteration:**

After initial deployment, user feedback was solicited actively, and necessary adjustments were made. This ongoing process helped refine user experience and add valuable features in line with user expectations.

**Lessons Learned**

**1. Flexibility in Planning:**

One of the critical lessons learned was the importance of flexibility in the scope and planning stages. The ability to pivot from the initial idea (including the carpooling module) to better cater to market demands was pivotal in delivering a more focused and useful product.

**2. User-Centric Design:**

The project reinforced the importance of a user-centric design approach. Engaging with real users during the design and prototype testing phase helped uncover practical insights that significantly shaped the final product.

**3. Agile Methodology Benefits:**

Employing Agile methodologies facilitated a more responsive development process. This approach helped the team adapt to changes swiftly and enhanced collaboration among cross-functional teams.

**4. Importance of Robust Testing:**

Comprehensive testing strategies were crucial in ensuring the reliability and stability of the application. Automated testing tools streamlined the process, while manual testing ensured the application’s usability and effectiveness.

**5. Continuous Learning and Improvement:**

Post-launch monitoring and continuous improvement based on user feedback proved essential. This iterative process not only helped in refining the application but also in staying relevant to user needs.

**6. Data Security and Privacy:**

Given the nature of the service, focusing on data security and privacy from the beginning was vital. Implementing stringent security measures and ensuring compliance with data protection regulations was a key takeaway.

This project not only resulted in the creation of a functional and valuable application for the C2C market but also provided the team with extensive learning opportunities in dealing with real-world challenges in app development. The experience has prepared the team better for future projects, emphasizing the iterative nature of modern software development and the need for user-driven innovation.

## Challenges

During the development of the Ujret Mobile App, the project team encountered several challenges that spanned both technical and non-technical aspects. Addressing these challenges required strategic problem-solving and adaptive measures to ensure the project's success. Below, I detail some of the key challenges faced and the approaches taken to mitigate these issues:

**Technical Challenges**

**1. Integration Complexity:**

Issue: Integrating multiple technologies such as React Native for the frontend, Flask and Node.js for the backend, and PostgreSQL and MongoDB for data management presented significant challenges. Ensuring seamless interaction between these components while maintaining performance standards was complex.

Solution: The team implemented robust API gateways and standardized data exchange formats like JSON. Extensive interface testing was conducted to ensure that all parts of the application communicated effectively without data loss or delay.

**2. Scalability and Performance:**

Issue: As the application was expected to handle a substantial user base, scalability was a primary concern. The initial deployment showed signs of strain under heavy load, which could lead to performance degradation.

Solution: We employed load balancing techniques and optimized our database queries to enhance performance. Additionally, the infrastructure was moved to scalable cloud services that could dynamically adjust resources based on traffic demand.

**3. Data Security and Privacy:**

Issue: With stringent data protection laws and the sensitive nature of user data, ensuring top-notch security measures was imperative but challenging.

Solution: We implemented multiple layers of security, including SSL/TLS for secure data transmission, encryption for data at rest, and robust authentication mechanisms. Regular security audits and compliance checks were instituted to maintain security standards.

**Non-Technical Challenges**

**1. Requirement Changes:**

Issue: Initial requirements were adjusted several times during the project due to shifting market needs and user feedback, which led to scope creep.

Solution: We adopted an Agile development approach, allowing us to be more adaptive to changes. Regular sprint reviews and stakeholder meetings were organized to align the development with current needs and expectations.

**2. User Adoption:**

Issue: Encouraging users to adopt a new platform over established competitors was challenging, particularly with initial trust-building.

Solution: We focused on unique value propositions, such as superior user experience and innovative features.

**3. Team Coordination:**

Issue: Coordinating a team of developers and designers was difficult due to the busy schedule of everyone and other courses workload .

Solution: We utilized collaboration tools like Slack, Zoom, and Jira to enhance communication. Regular virtual meetings and flexible work schedules helped maintain team cohesion and project momentum.

**4. Budget Constraints:**

Issue: Limited budget resources posed restrictions on certain aspects of development and testing phases.

Solution: Prioritization of project phases based on ROI and impact was key. We also leveraged open-source technologies and cloud-based resources to reduce costs without compromising quality.

Conclusion

Addressing these challenges required a mix of strategic planning, agile methodologies, and a focus on user-centric development. By turning these challenges into opportunities for learning and adaptation, the team delivered a robust and user-friendly platform that stands out in the competitive market of C2C services. Each hurdle reinforced the importance of resilience and innovation in the face of adversity, shaping a stronger and more versatile development team.

## Future

The Ujret Mobile App, poised for expansion, presents a host of opportunities to enhance and extend its functionalities to better serve its user base and enter new markets. Here are detailed approaches for future extensions based on the plans to include new modules like carpooling and freelancing, and improvements in existing features:

**1. Expansion to New Modules**

* **Carpooling Module:**
  + Purpose: To provide a platform for users to share rides, reducing travel costs and environmental impact.
  + Features: Real-time tracking of vehicles, scheduled rides, and automatic fare splitting.
  + Implementation: Integrate with GPS and mapping services to provide efficient routing and match riders with drivers based on routes and timings.

**Freelancing Module:**

* Purpose: To connect freelancers with potential clients for various services such as writing, graphic design, and digital marketing.
* Features: Project bidding, portfolio showcases, client reviews, and secure in-app payment channels.
* Implementation: Develop a robust system for profile management, service delivery tracking, and financial transactions, ensuring both party's interests are safeguarded.

**2. Enhanced Real-Time Interaction**

**Real-Time Communication:**

* Purpose: Improve user interaction within the app by facilitating instant communication.
* Features: In-app messaging, notifications for task updates, and live status sharing.
* Implementation: Use WebSocket for maintaining a persistent connection between the client and server, enabling real-time data transmission without repeated polling.

**3. Promotions and Loyalty Programs**

**Marketing Enhancements:**

* Purpose: To attract new users and retain existing ones by offering value-added services.
* Features: Discount codes, referral bonuses, and loyalty points that can be redeemed for services within the app.
* Implementation: Integrate a marketing automation tool to manage promotions and track user engagement and redemption rates efficiently.

**4. Advanced Payment Integrations**

**Secure Payment Solutions:**

* Purpose: To streamline the payment processes for all modules, making transactions seamless and secure.
* Features: Support for multiple payment gateways, digital wallets, and possibly cryptocurrency transactions.
* Implementation: Partner with established payment processors and implement blockchain technology for secure, transparent transactions.

**5. Artificial Intelligence and Machine Learning**

**Smart Matching Algorithms:**

* Purpose: To enhance the matching of services providers with seekers and carpool drivers with riders using AI.
* Features: Predictive analytics for service demand, automated matching based on preferences and past behaviors, and dynamic pricing models.
* Implementation: Deploy machine learning models that learn from user interactions and continuously improve recommendation and matching accuracy.

**6. Expansion to New Geographical Markets**

**Global Scaling:**

* Purpose: To expand the app’s presence internationally, adapting to different markets and user needs.
* Features: Multi-language support, localization of services, and compliance with local laws and regulations.
* Implementation: Research and develop region-specific features, comply with local data protection laws, and customize marketing strategies to different demographics.

**7. Robust Data Analytics Dashboard**

**Data-Driven Decisions:**

* Purpose: To provide administrators and business analysts with insights into user behavior and market trends.
* Features: Customizable reports, analytics on user engagement, financial metrics, and service quality metrics.
* Implementation: Integrate with a business intelligence tool or develop in-house capabilities for data aggregation and visualization.

By implementing these extensions, Ujret can significantly enhance its market position, user satisfaction, and overall service efficiency. These future developments will not only increase the app's utility but also foster a more connected and efficient user community.

# Review checklist

Before submission of this report, the team must perform an internal review. Each team member will review one or more sections of the deliverable.

| **Chapter/Section Name** | **Who did what** |
| --- | --- |
| Introduction,Objectives and Scope, Development and Planning, system Requirements (Functional + Non-Functional), Tool and Technologies, Use Case Diagram, Class Diagram, Software Process Selection, API specifications, system users' interface, Project Security, Project Risk Management, Testing and Evaluation, Conclusion, Challanges, Future Work | Talha Husnain |
| System Architecture, Architecture Diagram, Justification of the Architecture, Tools and Technologies, Database Design | Abdullah Ehsan |
| Sequence Diagrams | Jahanzaib Khursheed |
|  |  |

| **Chapter/Section Name** | **Reviewer Name(s)** |
| --- | --- |
| Introduction,Objectives and Scope, Development and Planning, system Requirements (Functional + Non-Functional), Tool and Technologies, Use Case Diagram, Class Diagram, Software Process Selection, API specifications, system users interface , Project Security, Project Risk Management, Testing and Evaluation, Architecture diagram, Sequence Diagram, Conclusion, Challanges, Future work | Talha Husnain |
| System Architecture, Architecture Diagram, Justification of the Architecture, Tools and Technologies, Database Design | Abdullah Ehsan |
| Non-functional Requirements  Sequence Diagrams  Software Process Selection | Jahanzaib Khursheed |
| Introduction, system requirement system architecture, Testing and Evaluation | Abdul Moez |

# References