# Sahelly / سهلّي

**Fixing It For You @ Home**

**Software Engineering Project**

**Presented and Documented By**

**Youssef Ashraf Ali**

**Zeiad Nouby**

**Shaheer**

# Index

# Project Overview

Sahelly stands as a pioneering solution, poised to redefine the landscape of how customers access essential home services. In this era of digital convenience, envision a world where you can effortlessly request a diverse array of services encompassing cleaning, carpentry, electrical work, maid services, paint jobs, and much more, all with the simple touch of your smartphone. In a manner reminiscent of the ease with which one orders a favorite meal from a restaurant via platforms like Talabat, Sahelly seamlessly extends this simplicity to the realm of home services.

Have a wobbly cabinet or a creaky door that demands the expertise of a skilled woodworker? Rest assured, Sahelly has you covered. Are your wall plugs behaving erratically, leaving you perplexed? Order the services of a seasoned electrician to diagnose and resolve the issue. Or perhaps, after a long and taxing day, you yearn to return to a home that is not only clean but also exudes an aura of meticulous organization. A request for maid services can be made at your utmost convenience.

The possibilities within Sahelly are boundless, rendering it your paramount solution for a comprehensive spectrum of home service needs. It is akin to having a dedicated team of experts at your beck and call, perpetually prepared to surmount any household challenge. Thus, we cordially invite you to embrace a realm of hassle-free, on-demand solutions with Sahelly.

Sahelly transcends the notion of a mere service provider; it is a deliverer of unparalleled convenience, imbuing households with an enduring sense of serenity and the luxury of time. It is your clandestine ally in the quest for a meticulously maintained and stress-free home. Whether you find yourself in need of prompt solutions, transformative renovations, or simply a respite, Sahelly stands resolute as your trusted confidant in the art of gracious living.

# Feasibility Study

## Economic Feasibility:

Given the project duration of 3 months, a capital of EGP 100,000 will be allocated. The money will be spent as follows:

**1. Personnel Costs (Salaries for Developers):**

- EGP 70,000 for the Flutter mobile app development team.
- Hourly cost: EGP 600
- Assuming 4 hours per week times 3 developers for 2.5 months = 120 hours.

**2. Hardware and Software Expenses:**

IDEs, version control systems, subscriptions, integrations and legal fees: EGP10,000

**3. Infrastructure Costs:**

A total of EGP 10,000 is allocated for the server fees and hosting costs (Firebase, Apple Developer account, Google Developer account)

**4. Marketing:**

A total of EGP 10,000 is allocated for online marketing for the product to drive app downloads.

**Revenue Streams:**

Positive Aspect: Sahelly can generate revenue through service fees, subscription models, or commissions from service providers, creating multiple income streams.
Concern: It may take time to build a substantial user base, and thus, revenue may initially be limited.

## Technical Aspects:

**Programming Language and Framework:** The project will be developed using Dart and the Flutter framework, which allows for cross-platform development for both IOS and Android. This choice simplifies development by using a single language for both platforms and provides flexibility for potential expansion to web applications in the future.

**Payment Gateway:** The implementation of a payment gateway is essential. Without it, the application might initially be limited to cash transactions until a suitable payment solution is integrated.

## Operational Considerations:

**Learning Curve**: As the team is starting from scratch with Flutter, there will be a learning curve to overcome. Learning resources will primarily be online content and self-study. This learning period should be factored into the project timeline.

**Development Team Size:** With only three team members, developing the application within a tight timeframe might be challenging, considering the learning curve and the complexity of the project.

**User-Friendliness:** Ensuring that users can easily understand and use the application is a crucial operational aspect. User experience and interface design should be a priority to make the application accessible to a wide audience.

## Schedule Feasibility:

**Project Timeline:**

Start Date: This project is set to submit its first milestone on the 10th of October 2023, which will mark the beginning of this project.

End Date: Project is set to be completed by the end of this educational semester, which is set to be in January 2024.

**Team Availability:**

The team members are available almost everyday and they're working around 3-4 hours weekly on this project every Sunday and Tuesday. We also have two scrum meetings, Youssef being the scrum master:

Obligatory Scrum Meeting: 12:30PM Tuesday GreenWall

Optional Scrum Meeting: Friday 8PM At Home

**Equipment and Software:**

We are mainly developing this application on mobile phones, so we're using operating systems that are running windows to develop the android application and we're using a macbook to develop the same application but for apple devices.

We're using VSCode to code flutter and we're also using Dart as a platform.

**Learning Period:**

Learning period will take around a month, and it will take the team a full week after learning Flutter to adapt to it.familiarize themselves with new technologies,

Milestones: Define specific project milestones that signify significant achievements or the completion of key project phases. Provide dates or timeframes for achieving these milestones, offering clear progress markers throughout the project.

**Contingency Period:** Dedicating a portion of the schedule to contingency time, allowing for unexpected issues, delays, or scope changes that may arise during the project.

**Purpose of Contingency:** contingency time is intended to ensure project flexibility and adaptability, minimizing the impact of unforeseen challenges.

# System Architecture:

**Technology Stack:**

We're developing this application on Flutter and we're going to use VSCode as an IDE to help us develop this application. As mentioned earlier, we're going to use two operating systems to help us fully implement this application on both IOS and Android platforms: MacOS and Windows.

**Scalability**:

Flutter is really good when it comes to Workload handling, which is basically if the system can effectively manage an increased number of users or a higher volume of transactions without becoming sluggish or unstable. And this language is also really optimized and efficient and it has lots of capabilities. Flutter can use frameworks like firebase, which is really good for data handling and storage.\

**Integration:**

We'll try to implement two types of integration:

Data Integration: It involves combining data from different sources and formats into a cohesive and accessible dataset. It's crucial for business intelligence, reporting, and decision-making.

Application Integration: Application integration connects various software applications to enable them to share data and functionalities. This integration can streamline business processes and improve productivity.

The benefits of integration include improving efficiency because it reduces manual data entry. Moreover, enhanced data accuracy, because it minimizes errors by duplicate data entry or inconsistencies. Furthermore, real-time access enables quicker decision-making and more responsive customer service.

**Security:**

Security evaluation within system architecture is a critical process aimed at assessing the measures in place to protect user data and transactions. In an increasingly digital world, safeguarding sensitive information is paramount. Here's how security is evaluated:

**Data Encryption:** The evaluation assesses the use of encryption mechanisms to secure data both in transit and at rest. It verifies that strong encryption protocols are used to prevent data breaches.

**Access Control:** This involves assessing who has access to sensitive data and ensuring that access is restricted to authorized personnel only. It verifies the presence of robust authentication and authorization mechanisms.

**Network Security:** The evaluation examines network configurations, firewalls, intrusion detection systems, and other security measures to safeguard against external threats, such as cyberattacks and data interception

# Delivery Schedule

Iteration 1: Weeks 1-2

Milestone 1: Project Kickoff (**Week 1**)

Deliverable 1: Project Plan

Deliverable 2: Team Roles and Responsibilities

Milestone 2: Requirement Gathering (**Week 2**)

Deliverable 3: Detailed Requirements Document

Iteration 2: Weeks 3-4

Milestone 3: System Design (**Week 4**)

Deliverable 4: High-Level System Architecture

Deliverable 5: Design Specifications

Iteration 3: Weeks 5-6

Milestone 4: Development Phase 1 (**Week 6**)

Deliverable 6: Initial Codebase

Deliverable 7: Unit Testing Reports

Iteration 4: Weeks 7-8

Milestone 5: Development Phase 2 (**Week 8**)

Deliverable 8: Feature Implementation

Deliverable 9: Updated Codebase

Iteration 5: Weeks 9-10

Milestone 6: Quality Assurance (**Week 10**)

Deliverable 10: Test Plan

Deliverable 11: Test Cases

Iteration 6: Weeks 11-12

Milestone 7: Integration and Testing (**Week 12**)

Deliverable 12: Integrated System

Deliverable 13: Test Reports

Iteration 7: Weeks 13-14

Milestone 8: User Acceptance Testing (**Week 14**)

Deliverable 14: UAT Plan

Deliverable 15: UAT Test Cases

Iteration 8: Weeks 15-16

Milestone 9: Beta Release (**Week 16**)

Deliverable 16: Beta Release Version

Iteration 9: Weeks 17-18

Milestone 10: Client Review (**Week 18**)

Deliverable 17: Review Feedback

Iteration 10: Weeks 19-20

Milestone 11: Final Adjustments (**Week 20**)

Deliverable 18: Final Release Version

Iteration 11: Weeks 21-22

Milestone 12: Deployment (**Week 22**)

Deliverable 19: Deployment Plan

Deliverable 20: Deployment Reports

Iteration 12: Weeks 23-24

Milestone 13: Post-Deployment Review (**Week 24**)

Deliverable 21: Post-Deployment Review Report

Deliverable 22: Lessons Learned

# User Requirements

**User Registration and Profiles (Functional):**
Users should be able to create accounts, with options for social media login.
Users should have personal profiles to manage their preferences and history.

**Service Search and Discovery (Functional):**
A user-friendly interface for searching and discovering various home services.
Filters to refine service search by location, type, and price.

**Detailed Service Listings (Functional):**
Comprehensive service descriptions, including pricing, availability, and service provider details. High-quality images and customer reviews for each service.

**Booking and Scheduling (Functional):**
An easy booking process with options for scheduling services at preferred times.
Notifications and reminders for booked services.

**User-Service Provider Communication (Functional):**
A messaging system for users to communicate with service providers.
In-app notifications and chat support for inquiries.

**Real-Time Tracking (Functional):**
Live tracking of service provider's location and estimated arrival time.
Geo-fencing to confirm service initiation and completion.

**User Ratings and Reviews (Non-Functional):**
Users should be able to rate and review service providers.
Transparent review system to build trust.

**Complaints and Dispute Resolution (Non-Functional):**
An efficient system for users to report issues or disputes with service quality.
A process for resolving complaints and providing refunds if necessary.

**Promotions and Loyalty Programs (Non-Functional):**
Offers, discounts, and loyalty programs to reward regular users.
Notification of promotions and discounts.

**Feedback and Improvement (Non-Functional):**
Channels for users to provide feedback and suggestions. Continuous improvement of the app based on user input.

**24/7 Customer Support (Non-Functional):**
Access to customer support through various channels (chat, email, phone). Quick response to user queries and issues.

**Accessibility and Usability (Non-Functional):**
An accessible app with features for users with disabilities. User-friendly design and navigation.

**Secure Payment Gateway (Functional):**
Integration with secure payment gateways for online payments.
Options for various payment methods, including credit cards, digital wallets, and cash on delivery.

**Service Provider Verification (Functional):**
Background checks and verification of service providers for user safety.
Display of provider ratings and reviews.

**Service Categories (Non-Functional):**
A wide range of service categories, such as cleaning, carpentry, electrical work, and more. Regular updates and additions to the service categories.

**Multi-Language Support (Non-Functional):**
Support for multiple languages is considered (besides Arabic) to cater to a diverse user base.

**Security and Data Privacy (Non-Functional):**
Strong security measures to protect user data and payment information. Clear privacy policies and user data management options.

# System Requirements

**User Registration and Profiles:**

- The system shall allow users to register for an account using an email address and password combination.
- The system shall offer social media integration for account creation and login, supporting platforms such as Facebook, Google, and Twitter.
- The system shall provide personal profile management where users can set and modify their preferences and view their service usage history.

**Service Search and Discovery:**

- The system shall provide a search interface that allows users to input search criteria for home services.
- The system shall include filters that allow users to refine search results by location, service type, and price range.
- The search results shall be displayed to the user in a list or grid format, sortable by various criteria.

**Detailed Service Listings:**

- The system shall display a detailed description for each service, including pricing, availability, and provider details.
- Each service listing shall include high-resolution images and an option for users to view and read customer reviews.

**Booking and Scheduling:**

- The system shall provide an interface for booking services, with the ability to choose preferred service times.
- The system shall send automatic notifications and reminders to users about their upcoming booked services.

**User-Service Provider Communication:**

- The system shall include a messaging feature for users to send and receive messages with service providers.
- The system shall provide real-time in-app notifications and chat support for user inquiries.

**Real-Time Tracking:**

- The system shall offer real-time tracking of service provider location and provide users with an estimated time of arrival.
- The system shall use geo-fencing technology to confirm the initiation and completion of services at the user's location.

**User Ratings and Reviews:**

- The system shall allow users to leave ratings and reviews for service providers after a service is completed.
- The system shall display reviews and ratings in a manner that ensures transparency and trustworthiness.

**Complaints and Dispute Resolution:**

- The system shall provide a feature for users to report issues or disputes with services.
- The system shall have a defined process for handling complaints, which may include refund issuance where applicable.

**Promotions and Loyalty Programs:**

- The system shall allow the administration to create and manage promotions, discounts, and loyalty programs.
- The system shall notify users of new promotions and loyalty program benefits based on their usage patterns and profile preferences.

**Feedback and Improvement:**

- The system shall include a feedback mechanism for users to submit suggestions and comments.
- The system shall support the collection and analysis of user feedback for continuous app improvement.

**24/7 Customer Support:**

- The system shall provide users with access to customer support via chat, email, and telephone.
- The system shall ensure quick response times, aiming for issue resolution within defined service level agreements (SLAs).

**Secure Payment Gateway:**

- The system shall integrate with at least one secure online payment gateway that supports encryption and secure data transmission.
- The system shall offer multiple payment options, including major credit cards, popular digital wallets, and the option for cash on delivery where applicable
- The system shall comply with relevant standards and regulations for financial transactions, such as PCI DSS.

**Service Provider Verification:**

- The system shall have a process for conducting background checks on service providers before onboarding them onto the platform.
- The system shall display verified service provider profiles with ratings and reviews from previous users.

**Service Categories:**

- The system shall categorize services into cleaning, carpentry, electrical work, and others as defined by the business.
- The system shall provide an administrative function for adding, updating, or removing service categories as needed.

**Multi-Language Support:**

- The system shall support content localization for multiple languages, including but not limited to Arabic, with the ability to switch languages within the app.
- The system shall provide a mechanism for maintaining and updating language-specific content.

**Security and Data Privacy:**

- The system shall implement robust security protocols for data transmission, storage, and handling of user information and payment details.
- The system shall provide users with clear privacy policies, data usage consents, and the ability to manage their data privacy settings.
- The system shall undergo regular security audits and updates to ensure ongoing protection against new vulnerabilities.

# Tasks and Dependency Chart

A - Research and Design (7 days): **HIGH PRIORITY**
This phase involves in-depth research and planning, where you will outline the project's scope, objectives, and design the application's architecture. It includes creating wireframes, defining features, and setting the project's overall direction.
Acceptance Tests:

- User research completed, and design requirements documented.
- Project design plan created and approved by the team.

B - Identify Target Market (3 days - Dependant on A): **HIGH PRIORITY**
Building on the research from task A, you will identify your target market for the application. This task includes analyzing user demographics, preferences, and needs, ensuring your app aligns with the intended audience.
Acceptance Tests:

- Target market analysis report prepared.
- Marketing strategy outlined and reviewed.

C - Competitor Research (4 days - Dependant on A): **MODERATE PRIORITY**
In parallel with task B, you'll conduct competitive analysis to understand the strengths and weaknesses of similar applications in the market. This data will help you position your app effectively.
Acceptance Tests:

- Competitor analysis report compiled.
- Competitive advantages identified.

D - Gathering Functional and Non-functional Requirements (5 days - Dependant on B, C): **HIGH PRIORITY**
Based on the information gathered in tasks B and C, you'll document the specific functional and non-functional requirements your app needs to meet. Functional

requirements outline features and capabilities, while non-functional requirements address performance, security, and scalability.

Acceptance Tests:

- Detailed requirements document created and validated.
- Non-functional requirements defined and approved.

E - Plan the Development Team (2 days - Dependant on D): **HIGH PRIORITY**

With the requirements in place, this task involves organizing the development team, assigning roles, and creating a development plan. You'll ensure you have the right skillsets to execute the project.

Acceptance Tests:

- Team roles and responsibilities defined.
- Development team plan reviewed and accepted.

F - App Development (20 days - Dependant on E, D): **HIGH PRIORITY**

This is the core development phase, where your team will start building the application according to the requirements outlined in task D. It includes coding, testing, and iterating to achieve the desired features.

**UI/UX Design (3 days):**

- Create wireframes and user interface designs.
- Design the user experience and app navigation.

**Frontend Development (4 days):**

- Develop the frontend of the application using Flutter.
- Implement user interfaces and app screens.

**Backend Integration (3 days):**

- Integrate the frontend with the backend systems.
- Establish connections to the server and databases.

**Functionality Implementation (3 days):**

- Develop the core functionalities of the application.
- Implement features like user registration, login, and service requests.

**Testing and Debugging (3 days):**

- Conduct thorough testing of the developed features.
- Identify and fix any bugs or issues in the application.

**User Authentication (2 days):**

- Implement secure user authentication and authorization.
- Ensure user data protection.

**Data Management (2 days):**

- Develop data storage and retrieval mechanisms.
- Implement database operations and data synchronization.

**Quality Assurance (2 days):**

- Perform quality checks to ensure the application meets requirements.
- Verify performance, usability, and adherence to design.

Acceptance Tests:

- Software developed and tested.
- Code review completed and approved.

G - Setting up Database (7 days - Dependant on F): **HIGH PRIORITY**

As part of the infrastructure setup, you'll establish the database systems required for the application, ensuring data storage and retrieval functions smoothly.

Acceptance Tests:

- Database configured and operational.
- Data storage and retrieval tested successfully.

H - Setting up API (4 days - Dependant on F): **MODERATE PRIORITY**
In parallel with task G, you'll configure the application's API (Application Programming Interface) to enable communication between different components of the app and external services.
Acceptance Tests:

- API endpoints defined and tested.
- Data exchange between the app and database validated.

I - Performance Testing (3 days - Dependant on F, G, H): **HIGH PRIORITY**
This phase focuses on evaluating the application's performance. You'll conduct tests to ensure that the app functions smoothly, responds quickly, and can handle expected user loads.
Acceptance Tests:

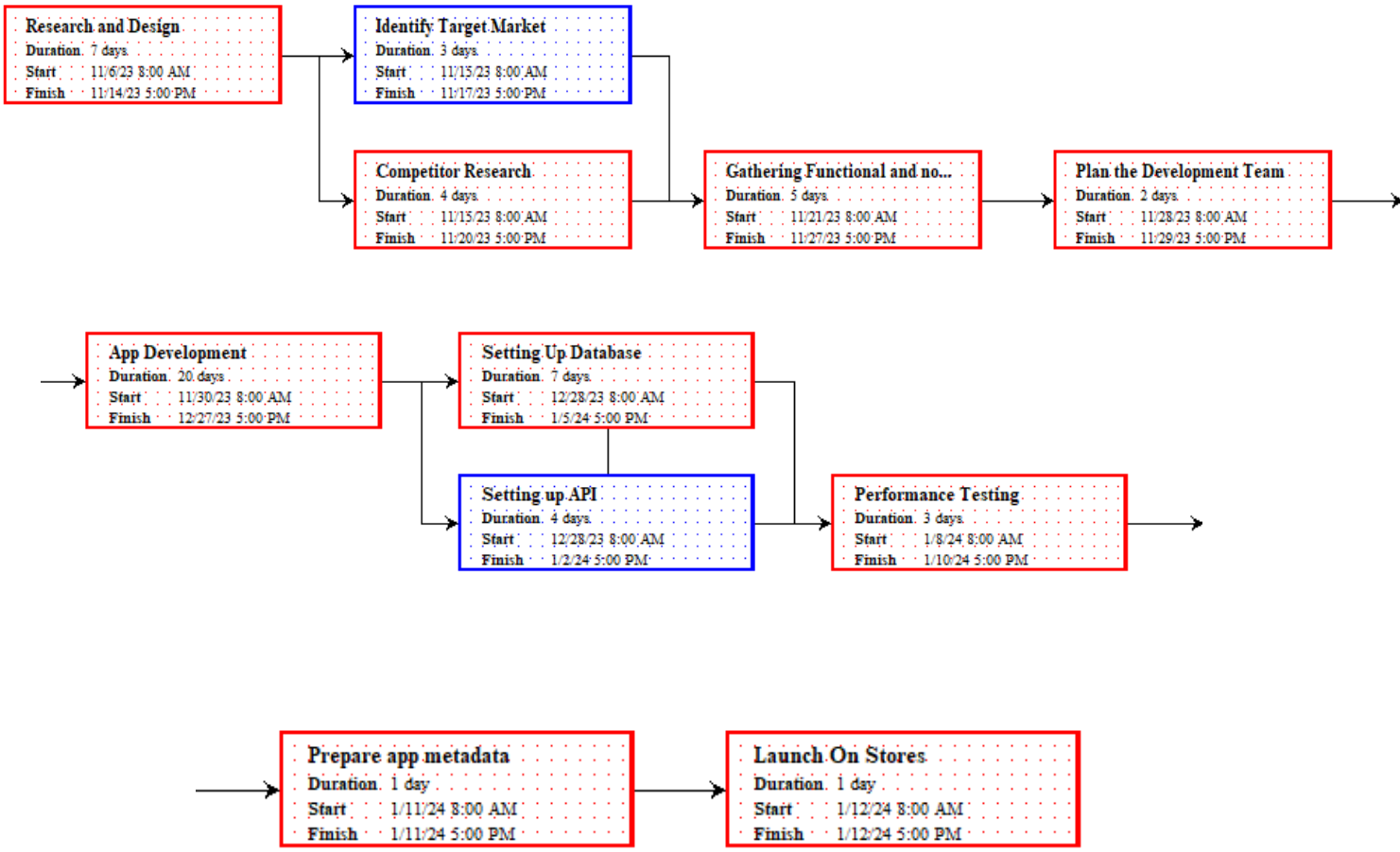
- Performance tests executed and results analyzed.
- App performance meets predefined criteria.

J - Prepare App Metadata (1 day - Dependant on I): **HIGH PRIORITY**
In preparation for launch, you'll create metadata, including app descriptions, screenshots, and other promotional materials that will be used in app stores and marketing efforts.
Acceptance Tests:

- App metadata, including descriptions and images, prepared.
- Metadata reviewed and approved.

K - Launch on Stores (1 day - Dependant on J): **HIGH PRIORITY**
Finally, with all preparations complete, you'll submit your application to the relevant app stores (e.g., Apple App Store, Google Play Store) for public release, making it available for download by users.
Acceptance Tests:

- App successfully published on app stores.
- User access and download tested.




Effort (in person-months) = a * (KLOC) ^ b

Effort = 2.8 * (2.0 ^ 1.1)

Effort ≈ 3.68 person-months

Now, let's calculate the total cost:

Total Cost = (Effort * Duration * Developers * Hours per day * Labour rate) / (Working days per month)

Assuming a project duration of 3 months, 3 developers per task, 8 hours per day, and 20 working days per month:

Total Cost = (3.68 * 3 * 3 * 8 * 20 * 3) / 20

Total Cost ≈ $8,870.40

---

**The Constructive Cost Model (COCOMO) offers several benefits for this project:**

- Effort Estimation: COCOMO provides a structured and well-defined method for estimating the effort required to complete a software project. This is valuable for project planning and resource allocation.
- Cost Estimation: It helps in estimating the total cost of the project, including labour costs. This allows for budget planning and cost control throughout the project's lifecycle.
- Resource Allocation: COCOMO assists in allocating the right number of developers to the project, ensuring that the workload is manageable and that the project can be completed on schedule.

# Technical & Operational Requirements:

**Technical:**

- **Data Security:** Robust security measures must be in place to protect user data, transactions, and sensitive information. This includes encryption, secure authentication, and authorization mechanisms.
- **Performance Optimization:** The app should be optimised for performance to ensure quick response times and smooth user interactions, even during peak usage periods.
- **API Integration:** Integration with external APIs may be required to access additional services or data. The architecture should support seamless API integration.

**Operational**:

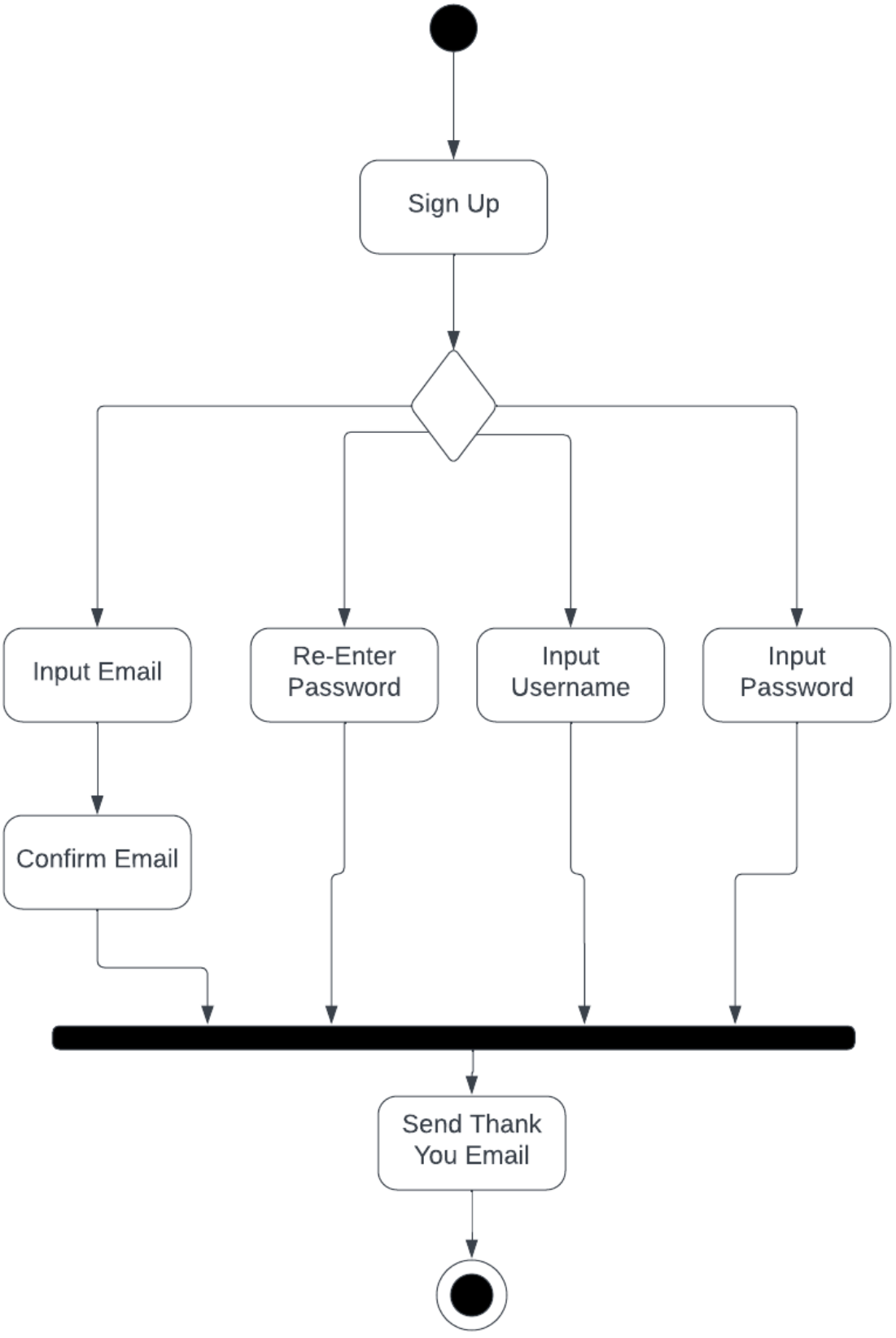
- **User-Friendly Interface:** The application's user interface should be intuitive, user-friendly, and easily navigable to ensure that users can efficiently access and utilize the available services.
- **24/7 Uptime:** The system should aim for a high level of availability, minimizing downtime to provide uninterrupted service to users.
- Customer Support: A support system should be in place to address user inquiries, issues, and complaints promptly.
- **Payment Gateway Integration:** A payment gateway must be implemented to facilitate transactions within the application, ensuring a secure and convenient payment process for users.

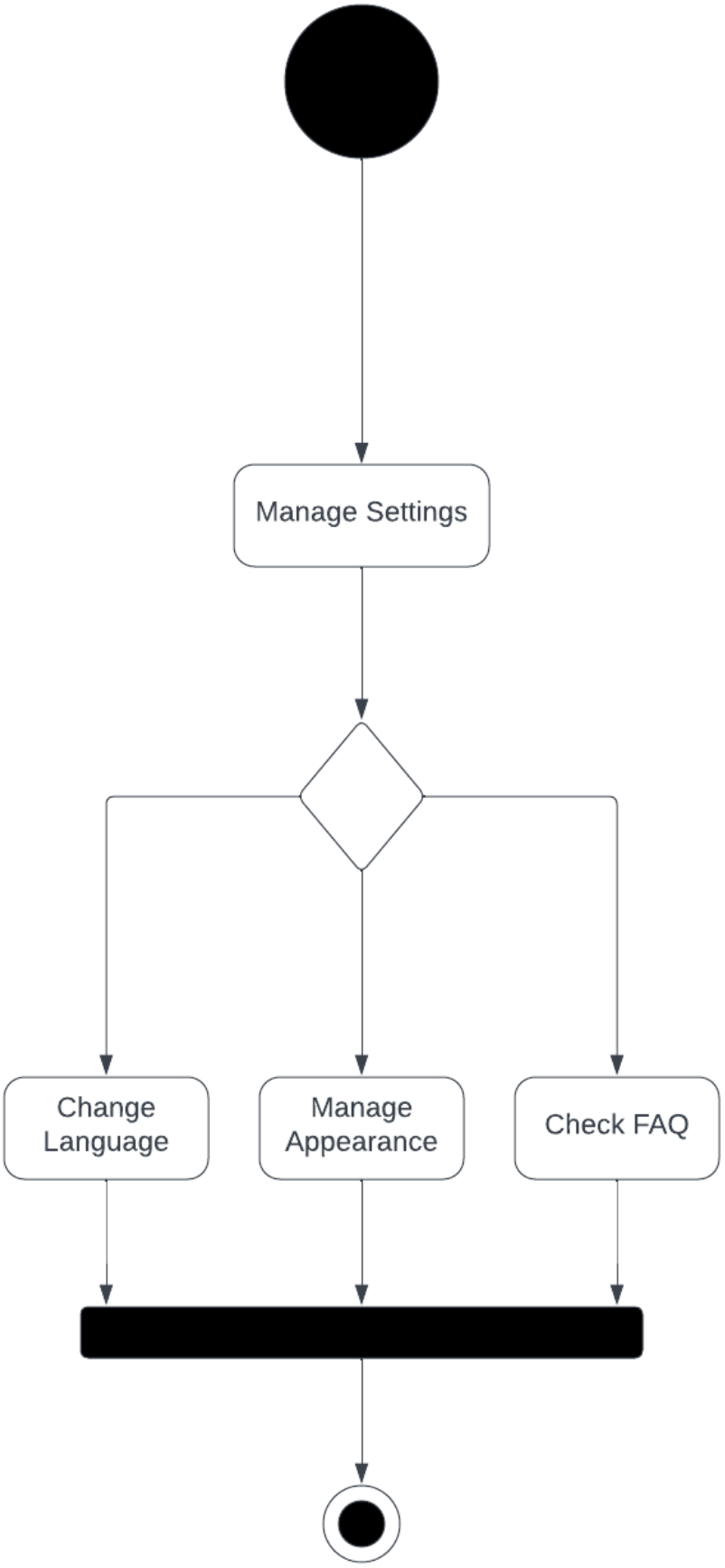
# RMMM Plan

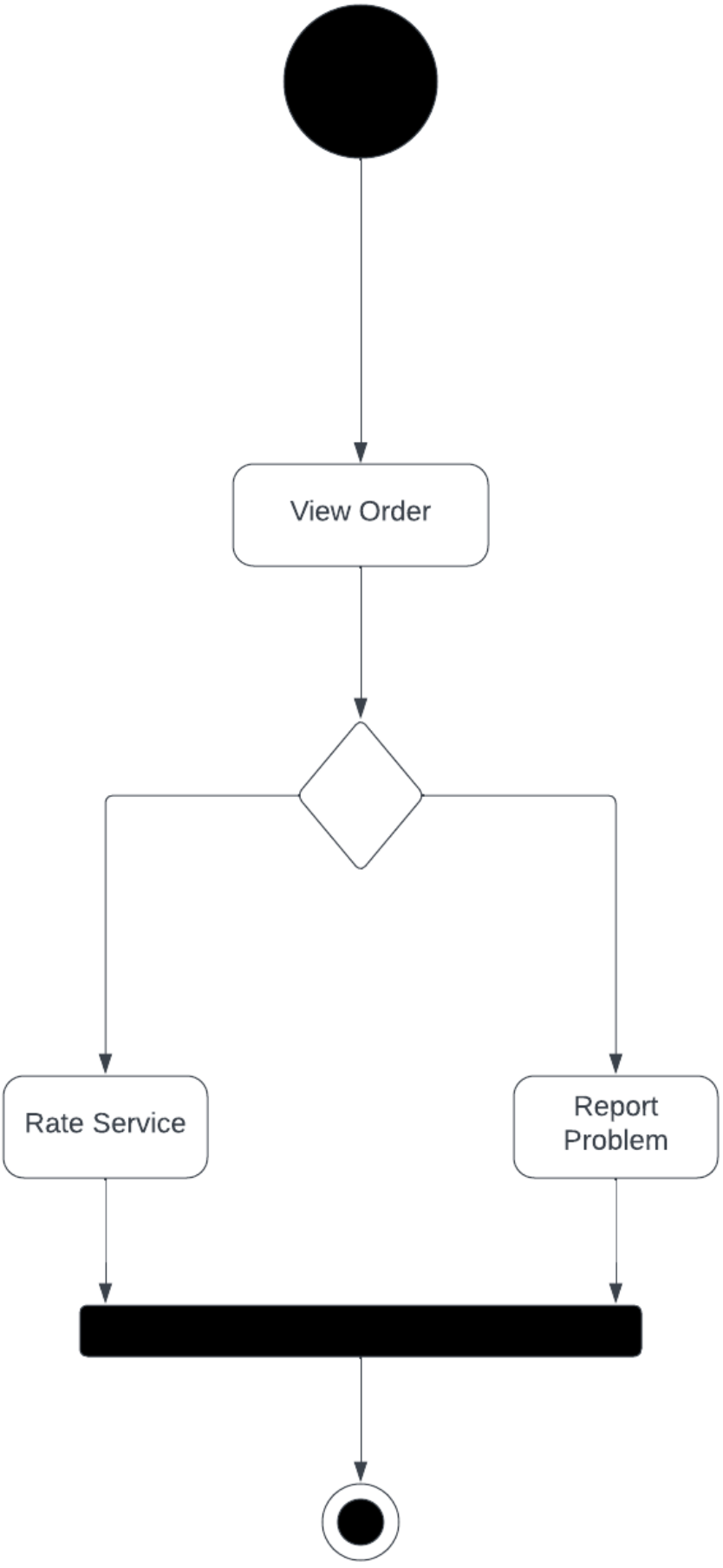
| Risk Id | Name | Description | Category | Priority | Probability | Impact | Triggers | Response |
|---|---|---|---|---|---|---|---|---|
| 1 | Technology Risks | Unforeseen technical challenges during development | Product | High | Medium | Serious | Complex Feature Integration | - Maintain Clear Communication<br>- Conduct Regular Tech Meetings |
| 2 | Resource Shortage | Insufficient Resources (Time, Capital) | Project | Medium | Medium | Serious | Delays | - Optimise Resource<br>- Allocate more resources<br>- Crowdfunding |
| 3 | Security Threats | Data Breaches, Hacking Attempts | Business Product | High | Low | Catastro-phic | - Seeking access to data for scraping or DDoS or to sell the data | - Penetration tests to ensure security<br>- Pushing users to adhere to security practices, 2FA and password managers. |
| 4 | User Adoption | Low Understanding of the app | Product | Medium | Low | Tolerable | Poor Design and Training | Tutorial videos explaining how the services works |
| 5 | Change in Requireme nts | Frequent changes in project requirements | Project | High | Medium | Serious | Evolving Customer Needs | A/B testing new features and the ones that show success are pushed to the mainstream |
| 6 | Integration Issues | Problems in integrating third party services | Product | High | Medium | High | Inconsistent APIs | - Seek support from providers.<br>- Seek other providers |

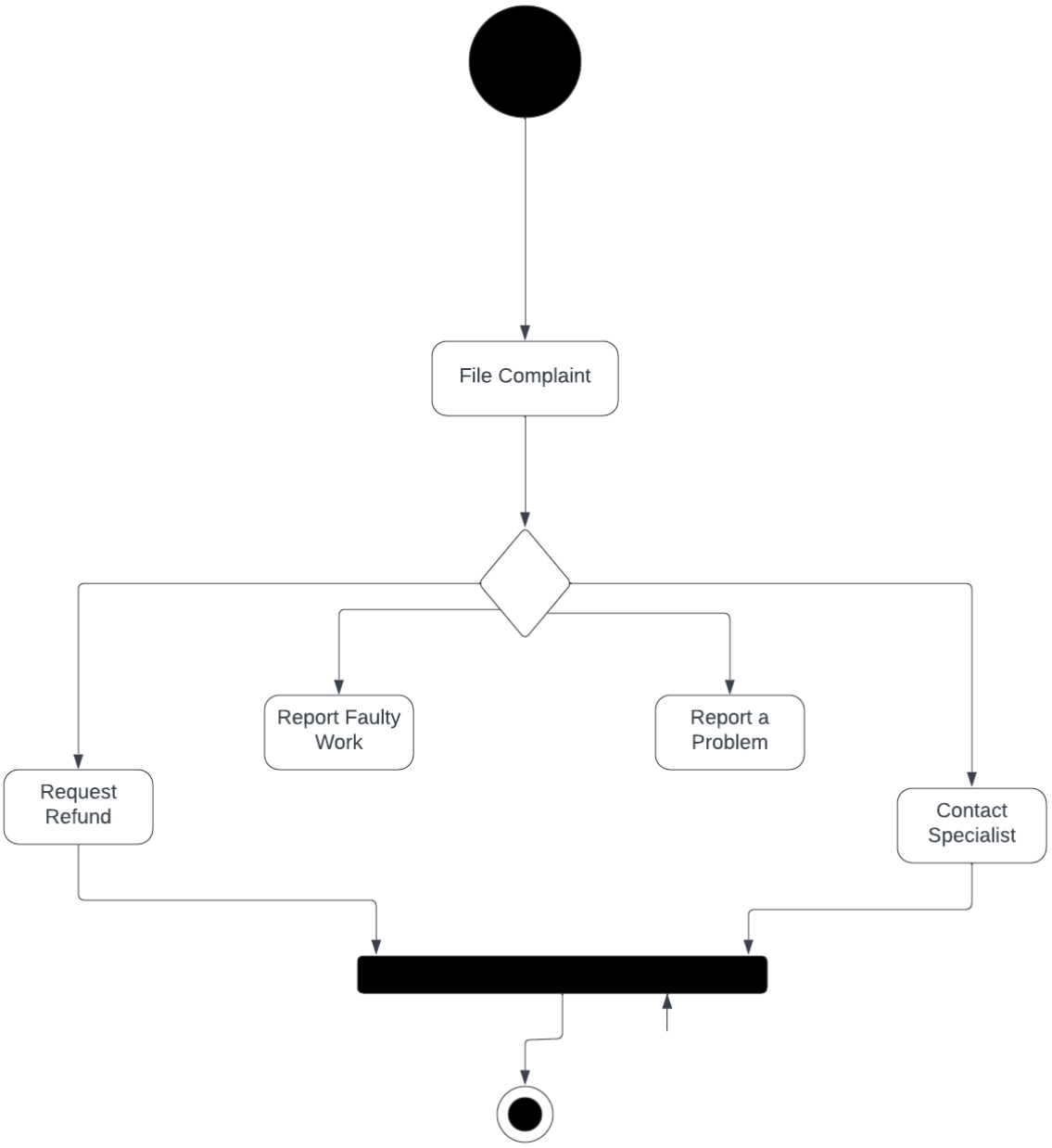
# Use Cases Diagram

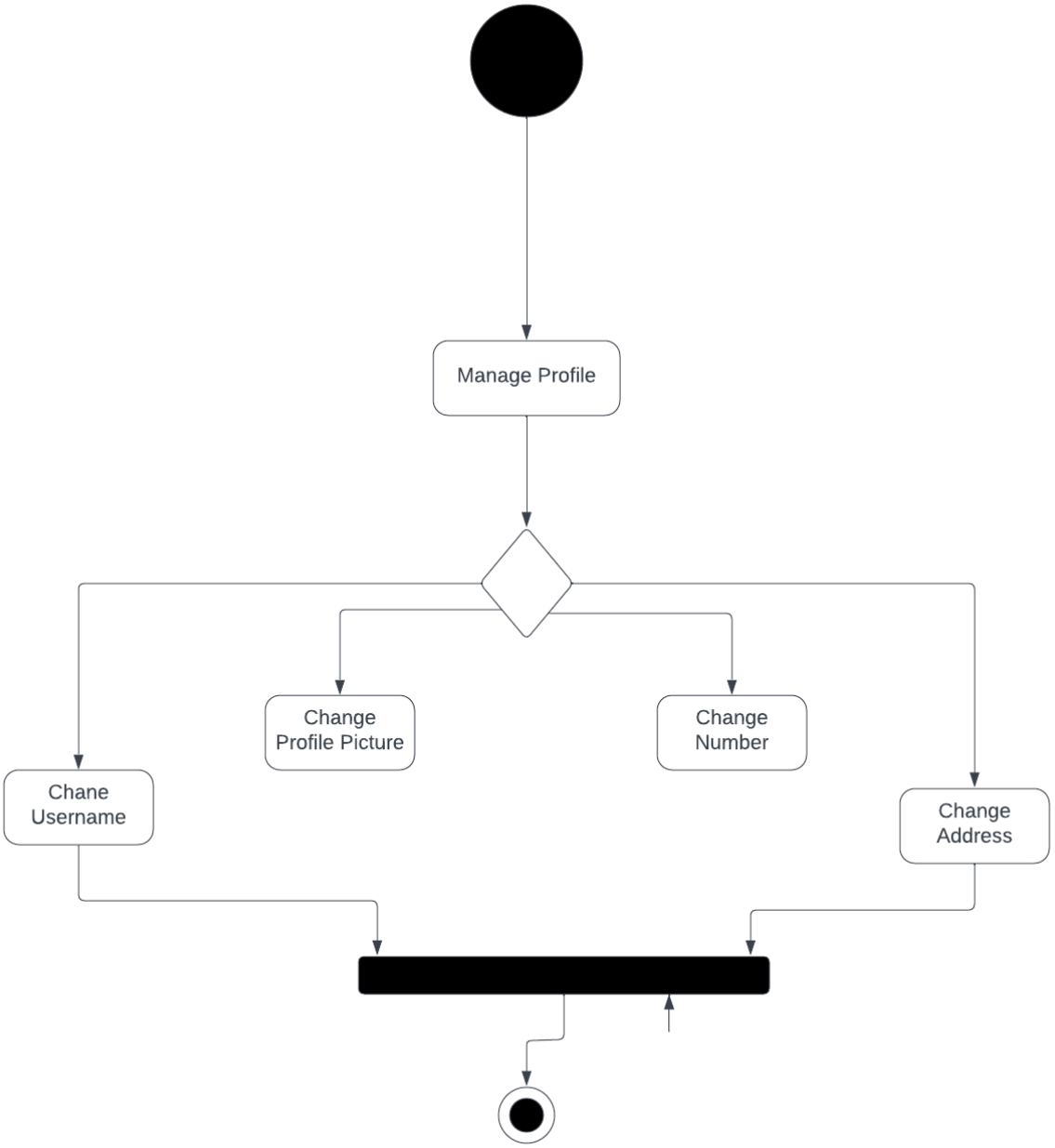
# Use Cases and Scenarios

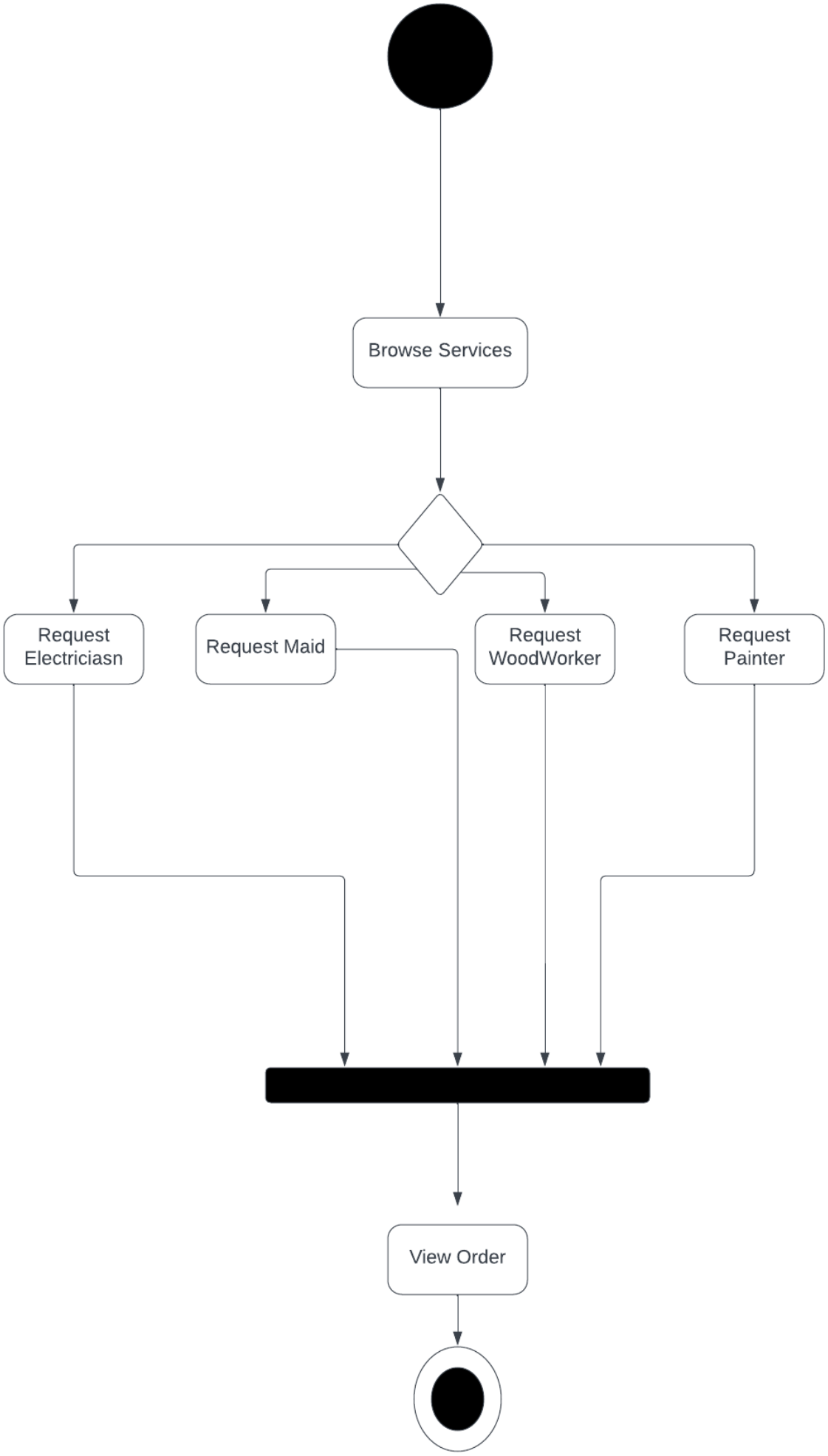


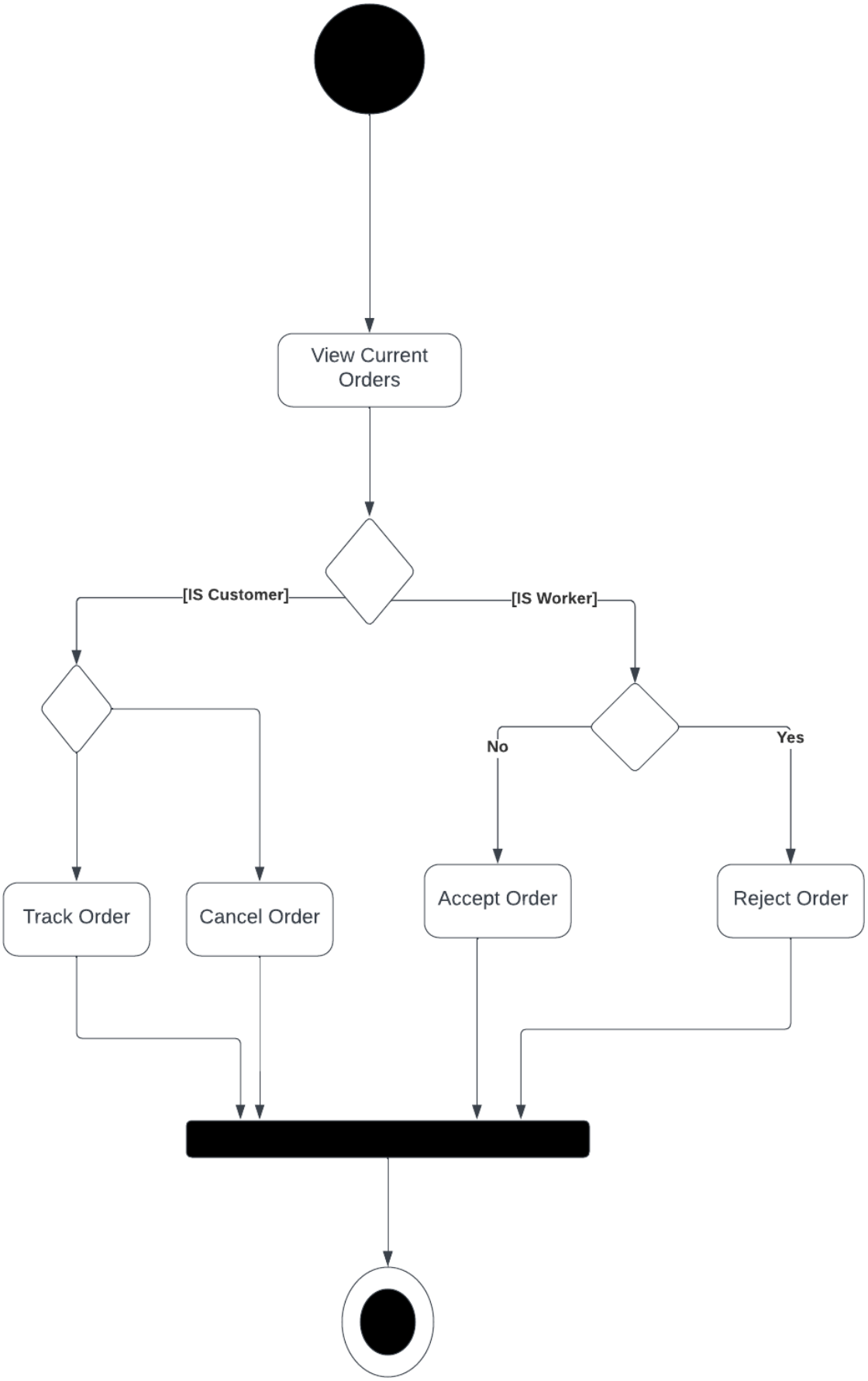

Manage Settings
Change Language
Manage Appearance
Check FAQ


View Order
Rate Service
Report Problem

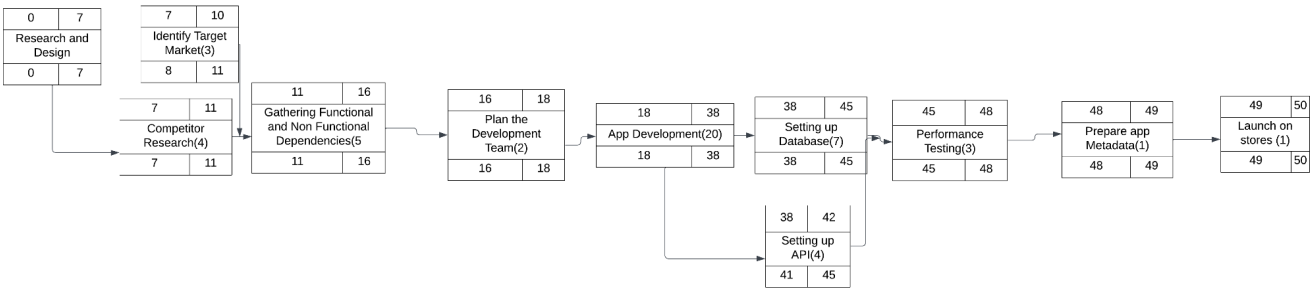

File Complaint
Report Faulty Work
Report a Problem
Request Refund
Contact Specialist


Manage Profile
Change
Profile Picture
Change
Number
Chane
Username
Change
Address

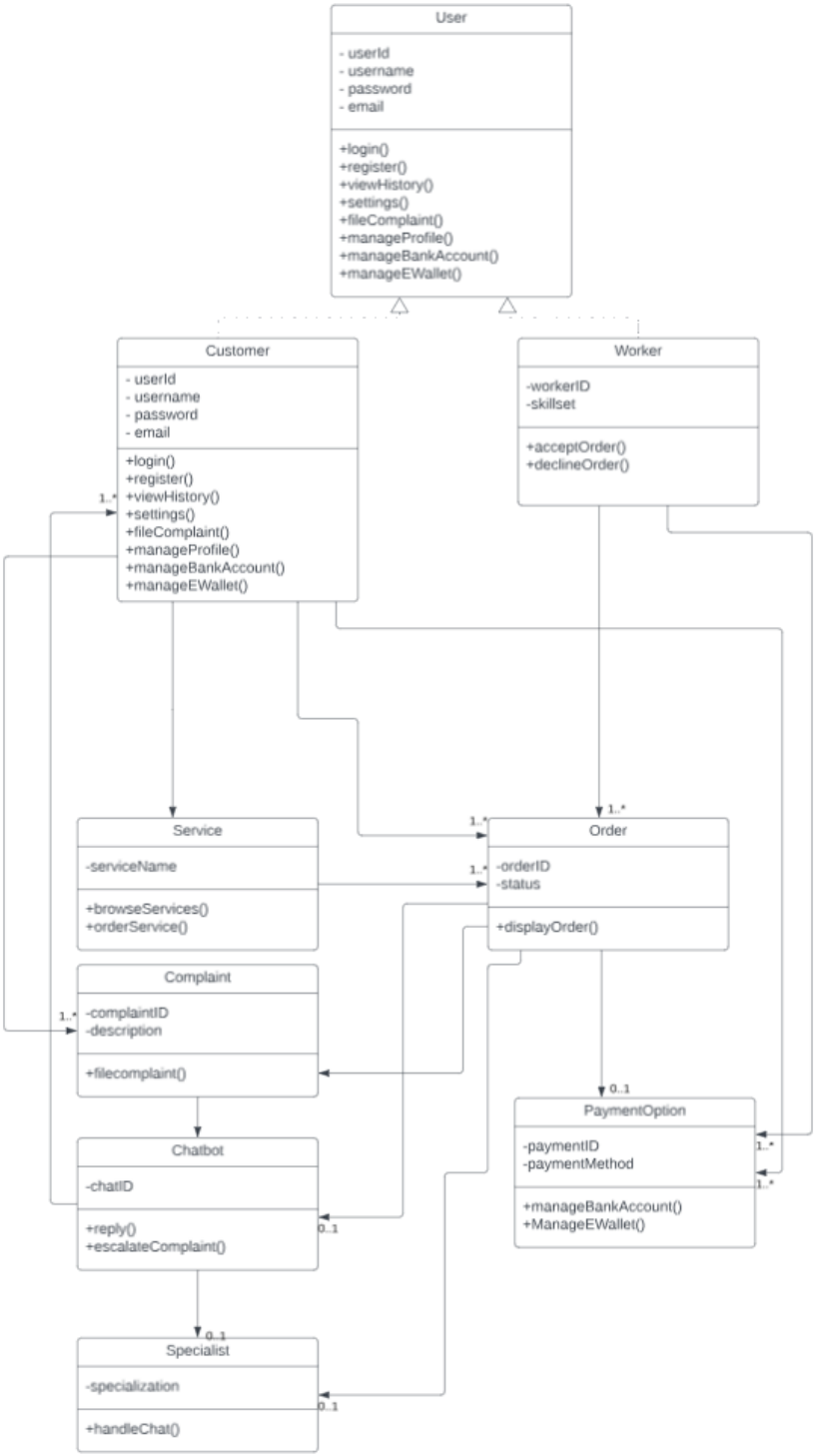

Browse Services
Request Electriciasn
Request Maid
Request WoodWorker
Request Painter
View Order


View Current Orders
[IS Customer]
[IS Worker]
No
Yes
Track Order
Cancel Order
Accept Order
Reject Order

# Task Activity Diagram



# Class Diagram



User to Order: 1..* (One User can place multiple Orders)

Worker to Order: 1..* (One Worker can handle multiple Orders)

Order to Specialist: 0..1 (An Order may involve a Specialist)

Order to ChatBot: 0..1 (An Order may involve a ChatBot for complaint handling)

Order to PaymentOption: 0..1 (An Order involves PaymentOption for payments)

Order to E-WalletService: 0..1 (An Order involves E-WalletService for payments)

ChatBot to Specialist: 0..1 (ChatBot can forward to a Specialist)

ChatBot to User: 1..* (ChatBot assists multiple Users)

Service to Order: 1..* (One Service can be part of multiple Orders)

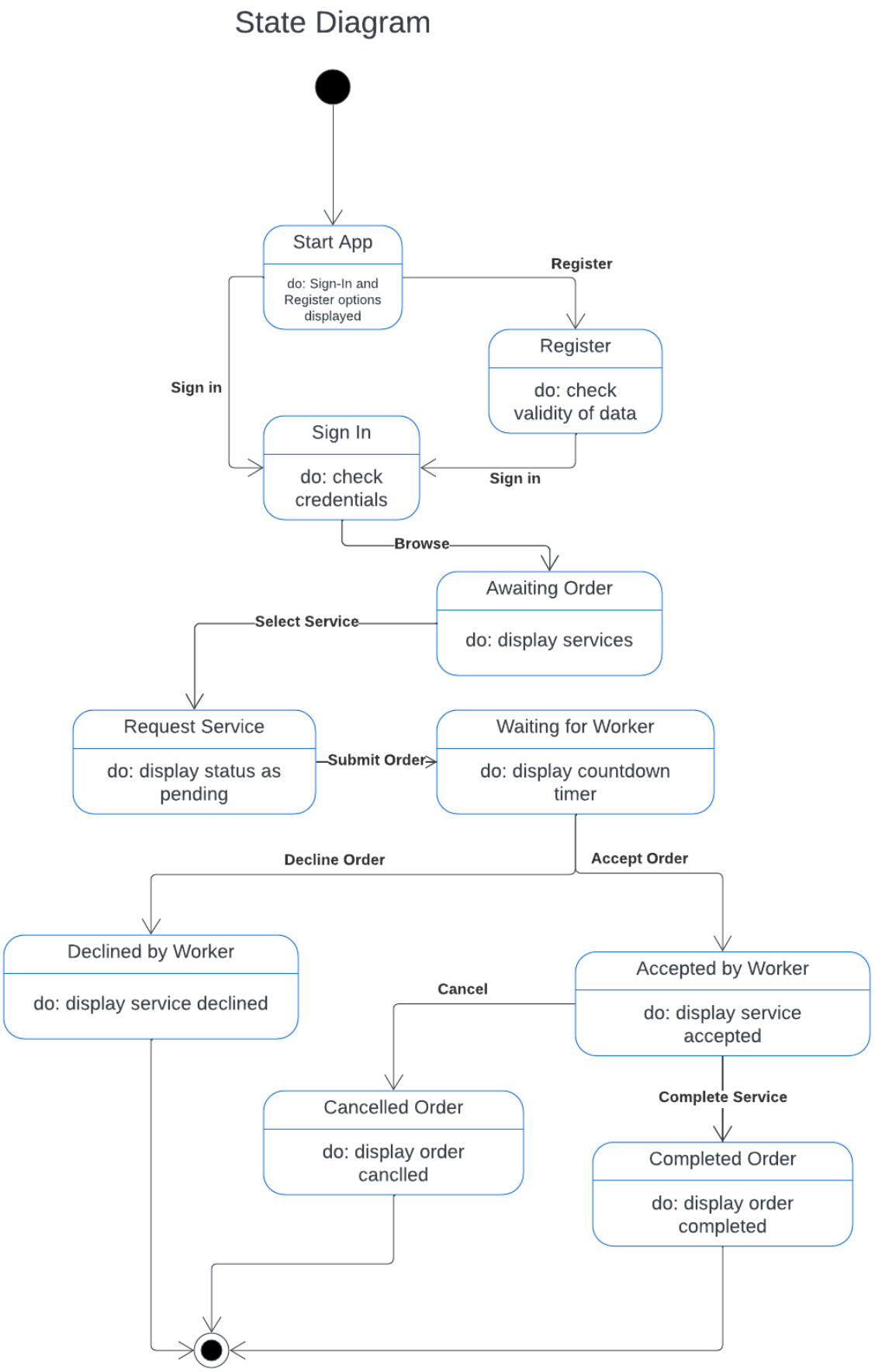
User to Complaint: 1..* (One User can file multiple Complaints)

User to PaymentOption: 1..* (One User can have multiple Payment Options)

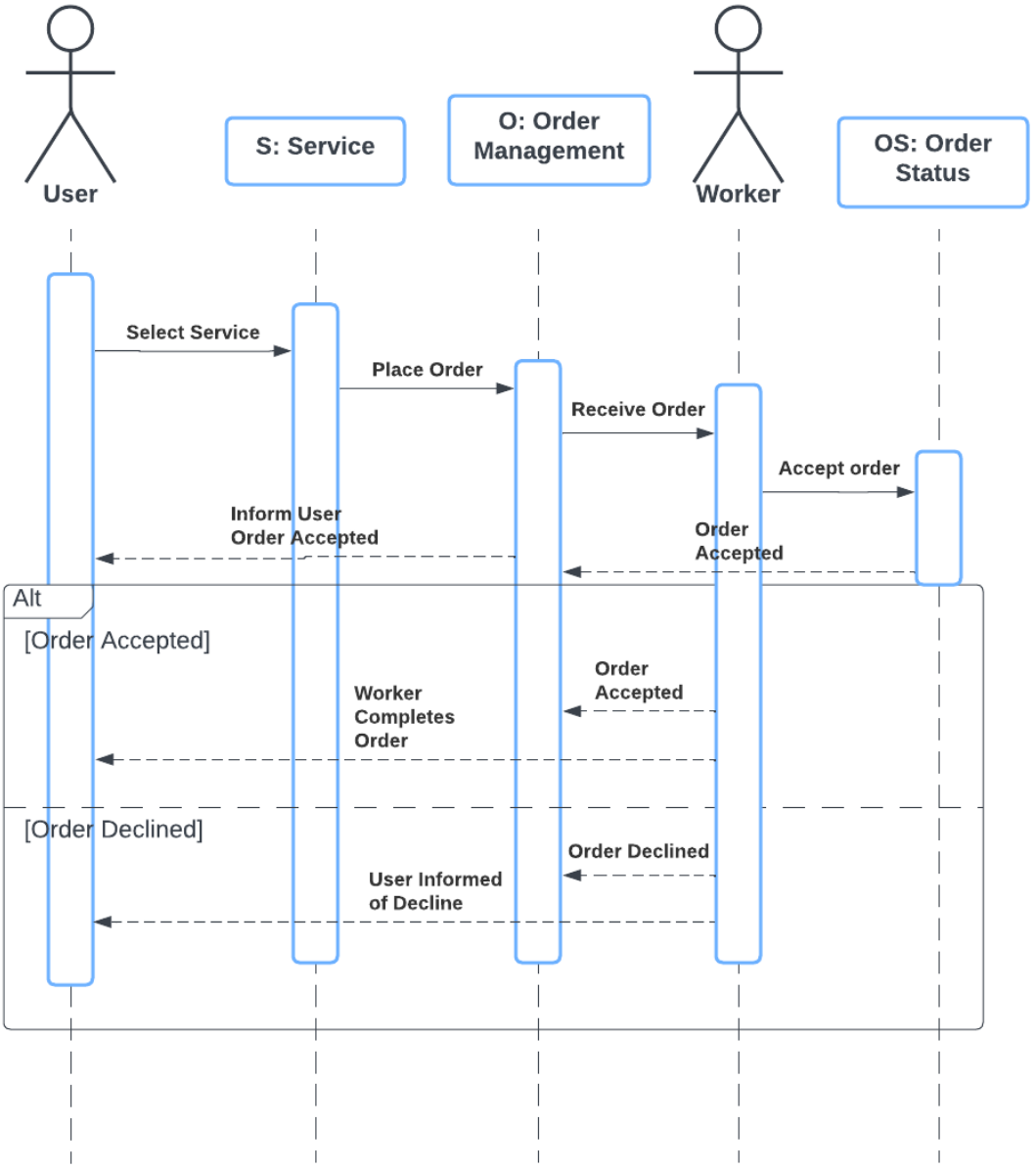
Worker to PaymentOption: 1..* (One Worker can have multiple Payment Options)

## States and Events

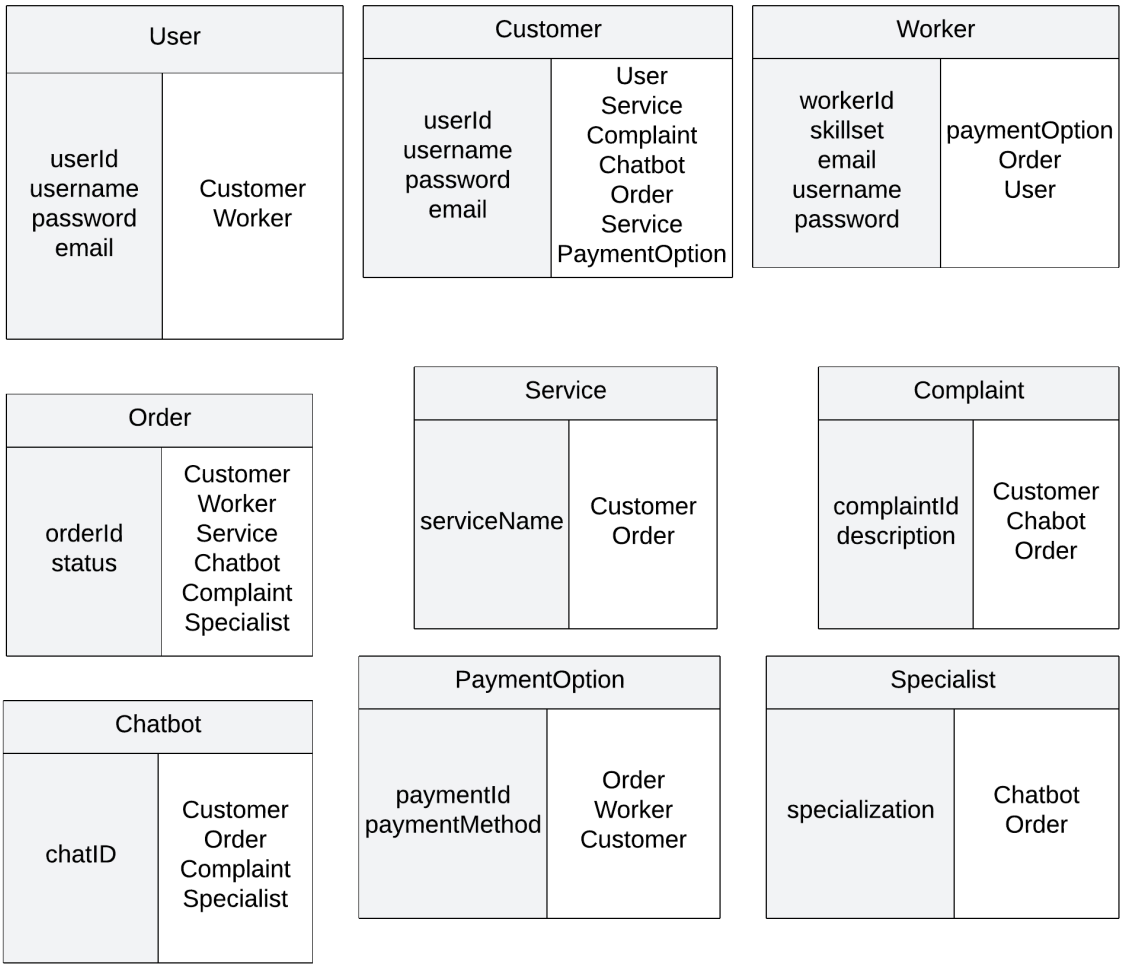
| State | Description |
|---|---|
| **Start App** | The user opens the app. |
| **Register** | The user gets to register an account. System checks if data is valid. |
| **Sign-In** | The user gets to register an account. System checks if data is valid. |
| **Awaiting Order** | The user has logged in and is browsing services |
| **Request Service (Order Placed)** | User requests service. System shows service request status as pending. |
| **Waiting for Worker** | Pending a worker to accept the service. Request shows pending |
| **Accepted by worker** | Service is accepted. A worker accepted the service request. |
| **Declined by worker** | Service is declined. A worker declined the service request. |
| **Completed order** | Service is completed. A worker completed the service request |
| **Cancelled order** | Service is cancelled. User cancelled service. |


State Diagram
Start App
do: Sign-In and Register options displayed
Register
Register
do: check validity of data
Sign in
Sign In
do: check credentials
Sign in
Browse
Awaiting Order
do: display services
Select Service
Request Service
do: display status as pending
Submit Order
Waiting for Worker
do: display countdown timer
Decline Order
Accept Order
Declined by Worker
do: display service declined
Accepted by Worker
do: display service accepted
Cancel
Cancelled Order
do: display order canclled
Complete Service
Completed Order
do: display order completed

Sequence Diagram



# CRC Model


User
userId
username
password
email
Customer
Worker
Customer
userId
username
password
email
User
Service
Complaint
Chatbot
Order
Service
PaymentOption
Worker
workerId
skillset
email
username
password
paymentOption
Order
User
Order
orderId
status
Customer
Worker
Service
Chatbot
Complaint
Specialist
Service
serviceName
Customer
Order
Complaint
complaintId
description
Customer
Chabot
Order
Chatbot
chatID
Customer
Order
Complaint
Specialist
PaymentOption
paymentId
paymentMethod
Order
Worker
Customer
Specialist
specialization
Chatbot
Order