Software Design Document
Of
CAMPUSWORKS

Submitted To :

Md. Jubair Ibna Mostafa
Assistant Professor
Department of Computer Science and Engineering
Islamic University of Technology

Submitted By :

No	Name	Student ID
1	A Z Hasnain Kabir	200042102
2	Mamunur Rahman	200042116
3	Oishy Fatema Akhand	200042128
4	Shadman Sakib	200042144
5	Mukit Mahdin	200042170

Table of Contents

Table of Contribution

Name	Student ID	Contribution
A Z Hasnain Kabir	**200042102**	**3.1 - Transforming Analysis Class into Design Class** **3.2 - Message Passing**
Mamunur Rahman	**200042116**	**1- Introduction** **2.1 - Context Diagram** **2.2 - Archetype Diagram** **2.3 - Refining Archetype**
Oishy Fatema Akhand	**200042128**	**3.3 - UML Activity Diagram** **3.5 - Deployment Diagram**
Shadman Sakib	**200042144**	**3.4 - State Diagrams**
Mukit Mahdin	**200042170**	**4.1 - User Analysis** **4.2 - Task Analysis** **4.3 - User Interfaces Mockup**

Chapter 1: Introduction

This document outlines the architectural, component-level, and user interface design for the CampusWorks App. The architectural design defines the overarching structure and interactions of the system, while the component-level design delves into the intricate details of individual components and their interactions. The user interface design aims to grasp the users' requirements and preferences to craft an intuitive and effective interface.

1.1 Architectural Design

Architectural design is about defining the structure of both data and program components needed for a computer-based system. It involves determining the architectural style, the properties of the components, and how they interact within the system. This design phase serves as the initial blueprint for building the software.

To develop the architectural design for the CampusWorks, we followed these steps:

a) **Representing the System in Context:** This step involves understanding the broader context in which the system will operate.
b) **Defining Archetypes:** Here, we identify the key patterns or types of components that will be used in the system.
c) **Refining the Architecture into Components:** This step involves breaking down the architecture into specific components and defining their properties and relationships.

1.2 Component Level Design

Component-level design comes after the initial architectural design. By this stage, the overall structure of the software has been established, but the internal details of each component are not yet fully fleshed out.

During component-level design, we define the internal data structures, algorithms, interfaces, and communication mechanisms for each component. This detailed design is represented graphically, in tables, or in text.

Here are the steps we followed for component-level design:

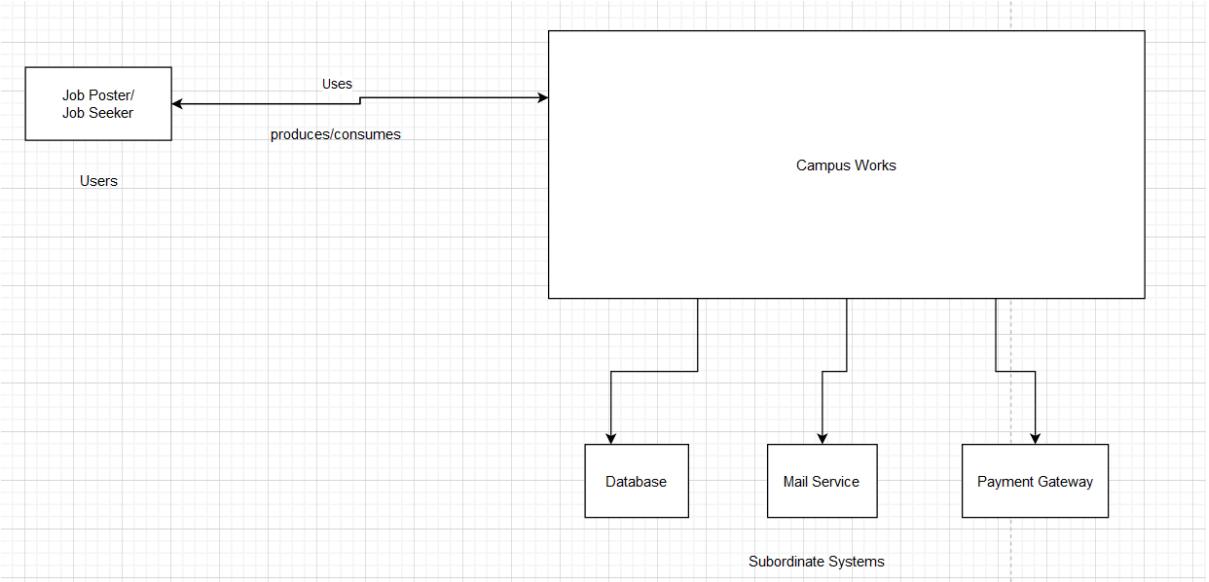
a) **Transforming Analysis Class into Design Class:** Converting the high-level analysis classes into detailed design classes.
b) **Specifying Message Details for Collaboration:** Describing how classes or components will interact by specifying messages.
c) **Describing Processing Flow:** Detailing how each operation within a component will be processed.
d) **Developing Behavioral Representations:** Creating detailed representations of the behavior of each class or component.
e) **Elaborating Deployment Diagrams:** Providing additional implementation details in deployment diagrams.
f) **Refactoring and Considering Alternatives:** Continuously refining the design representations and exploring alternative approaches.

Software Usage Scenario :

The purpose of CampusWorks is to provide a comprehensive platform for the freelance job-seeking and posting needs of the IUT community. The platform aims to facilitate the connection between job seekers and employers by providing them with essential tools and functionalities required for effective communication and collaboration. With its user-friendly interface and a range of features, CampusWorks provides a seamless experience for both job seekers and employers, allowing them to easily find and post jobs that align with their skills and interests. Additionally, CampusWorks aims to promote transparency and accountability in the freelance job market by providing a review and rating system, which allows clients to provide feedback and rate freelancers based on their work. Overall, the purpose of CampusWorks is to create a platform that meets the needs of the IUT community and provides a comprehensive solution for freelancers and employers to connect and find the right jobs.

Chapter 2: Architectural Design

2.1 Context diagram



We have developed a context diagram to model the manner in which the CampusWorks app interacts with entities external to its boundaries.
Systems that interoperate with the target system (the system for which architectural design is to be developed) are represented as

Superordinate systems - those systems that use the target system as part of some higher-level processing scheme. There is no super-ordinate system for CampusWorks.

Subordinate systems - those systems that are used by the target system and provide data or processing that are necessary to complete target system functionality. Mail Service, Payment Gateway, and Database can be considered as subordinate systems for it.

Peer-level systems - those systems that interact on a peer-to-peer basis (i.e., information is either produced or consumed by the peers and the target System). There is no peer system for it.

Actors - entities (people, devices) that interact with the target system by producing or consuming information that is necessary for requisite processing. In this scenario, there is no classification of users as a Job poster can also be a Job seeker.

2.2 Archetype Diagram

Defining Archetypes

An archetype is a class or pattern that represents a core abstraction that is critical to the design of an architecture for the target system. In general, a relatively small set of archetypes are required to design even relatively complex systems. The target system architecture is composed of these archetypes, which represent stable elements of the architecture but may be instantiated many different ways based on the behavior of the system.

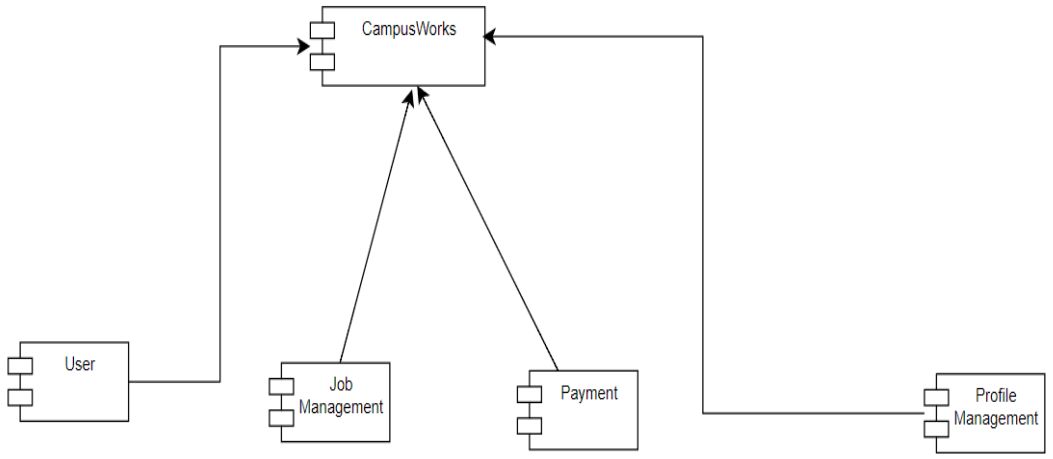
We have derived the archetypes for CampusWorks by examining the analysis classes defined as part of the requirements model. The archetypes are defined as follows:

User: An abstraction that represents a Job poster posting for a job and a Job seeker applying for it.

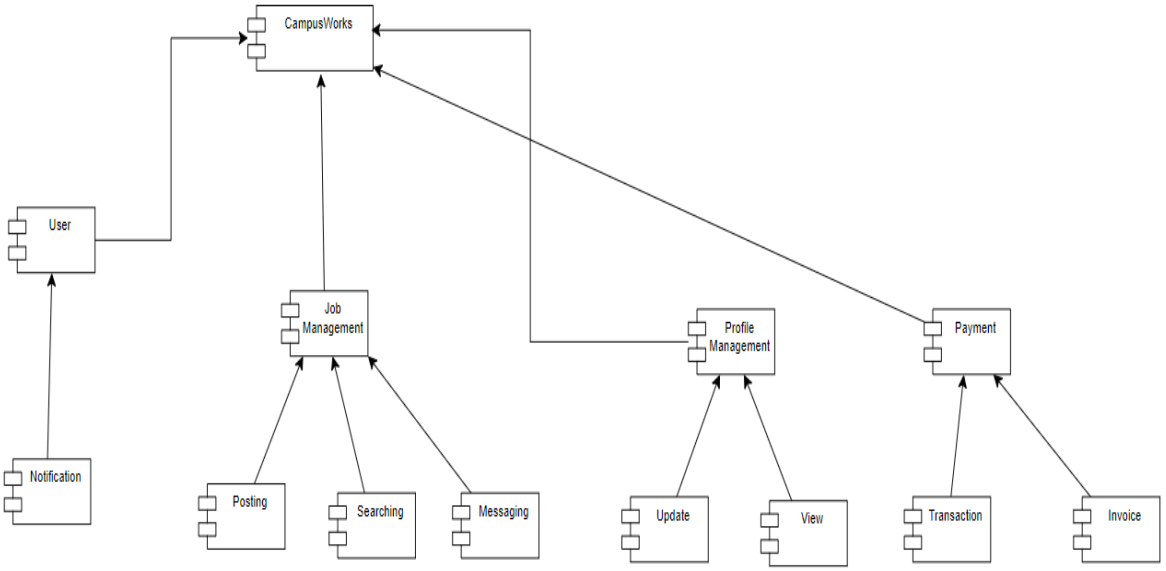
Job Management: An abstraction that encompasses all mechanisms related to the job posting, job searching, and connecting fellow users about more inquiries.

Profile Management: An abstraction that represents all mechanisms for users updating their own profiles so users have a better chance to land a job and also to help with the search algorithm.

Payment: An abstraction that represents all mechanisms for making and storing payment transactions and also sending invoices as confirmation.



2.3 Refining Archetype into Top-level Component



As the software architecture is refined into components, the structure of the system begins to emerge. We have chosen these components by beginning with the classes that were described as part of the requirements Model. These analysis classes represent entities within the application domain that must be addressed within the software architecture. The archetypes are User, Job Management, Profile Management, and Payment. The set of top-level components with their functions are given below:

Notification: This component is refined from the user archetype. This component is involved with sending notifications and initiating messaging services. Users get notified when other users are interested in one of the jobs they posted.

Posting: This component is refined from the Job Management archetype. This component is involved with how a user can post a job.

Searching: This component is refined from the Job Management archetype and involved with the mechanism and algorithm of how users can search for desired jobs.

Messaging: This component is refined from the Job Management archetype and it is involved with connecting users if they have any inquiries about the jobs.

Update: This component is refined from the Profile Management archetype and it is involved with personalizing a user's profile information.

Viewing: This component is refined from the Profile Management archetype and it is involved with viewing more information about their employer or a job poster.

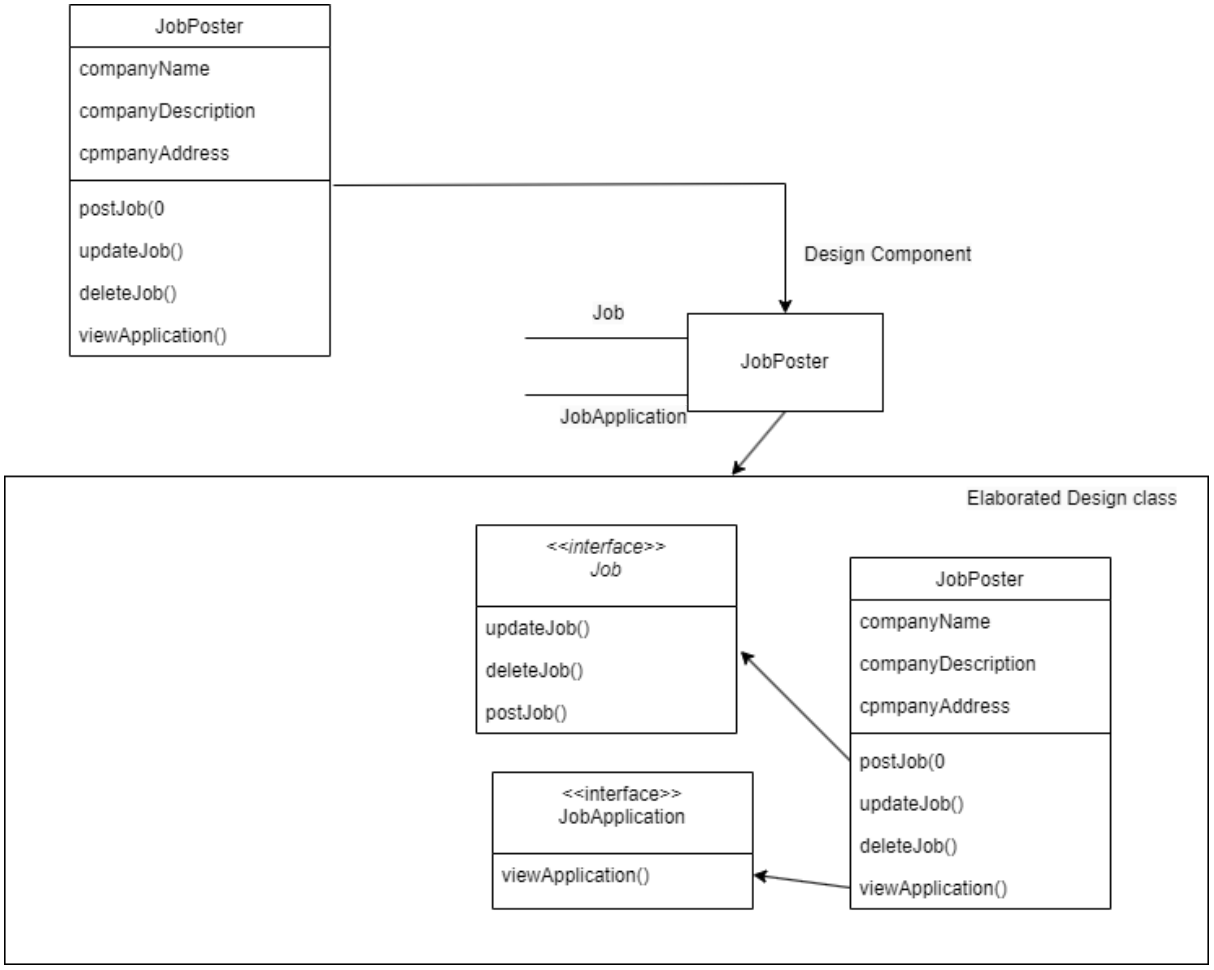
Transaction: This component is refined from the Payment archetype and it is involved with making transactions and storing them in the database.

Invoices: This component is refined from the Payment archetype and it is involved with sending emails for payment confirmation

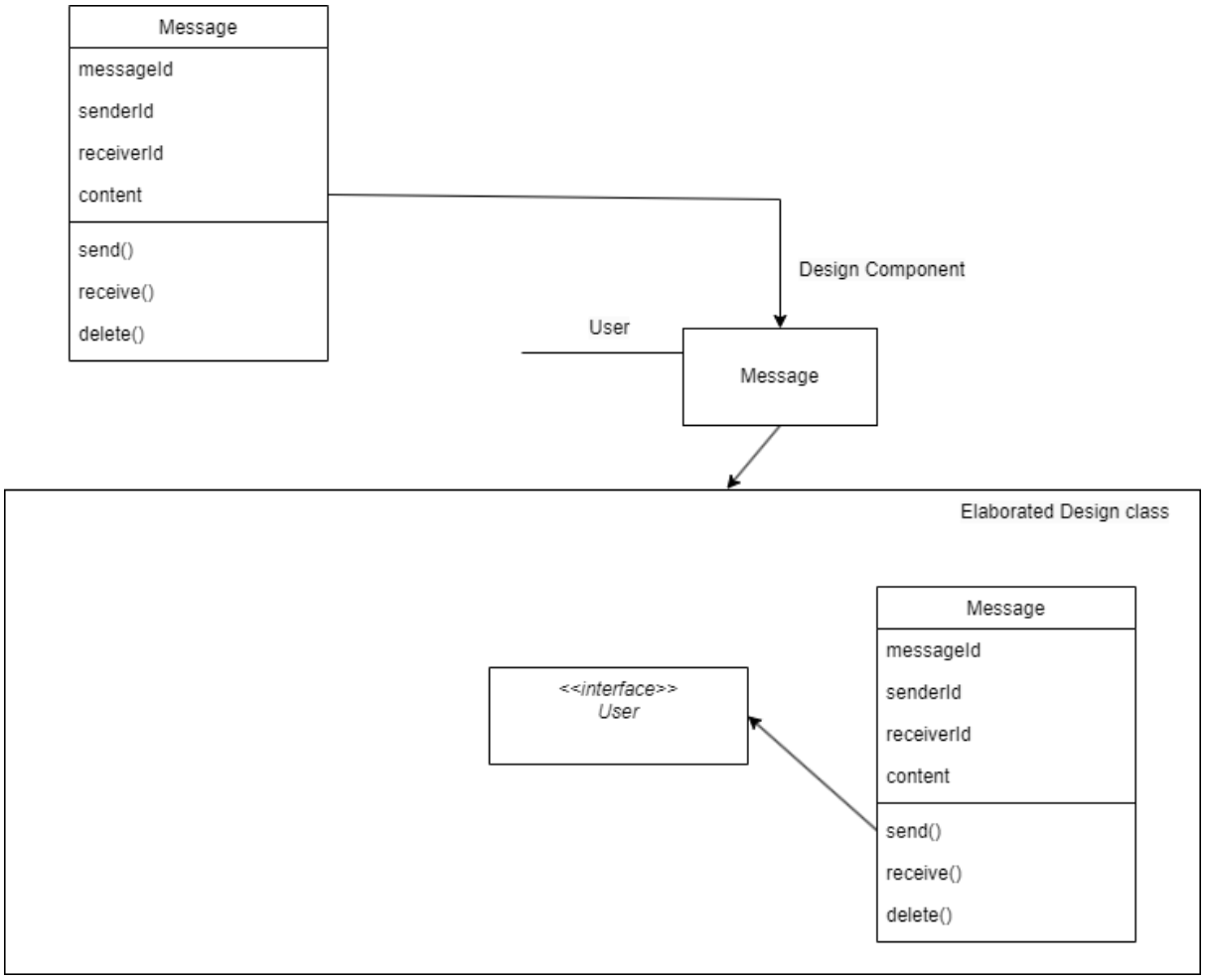
Chapter 3: Component-Level Design

3.1 Transforming Analysis Class to Design Class

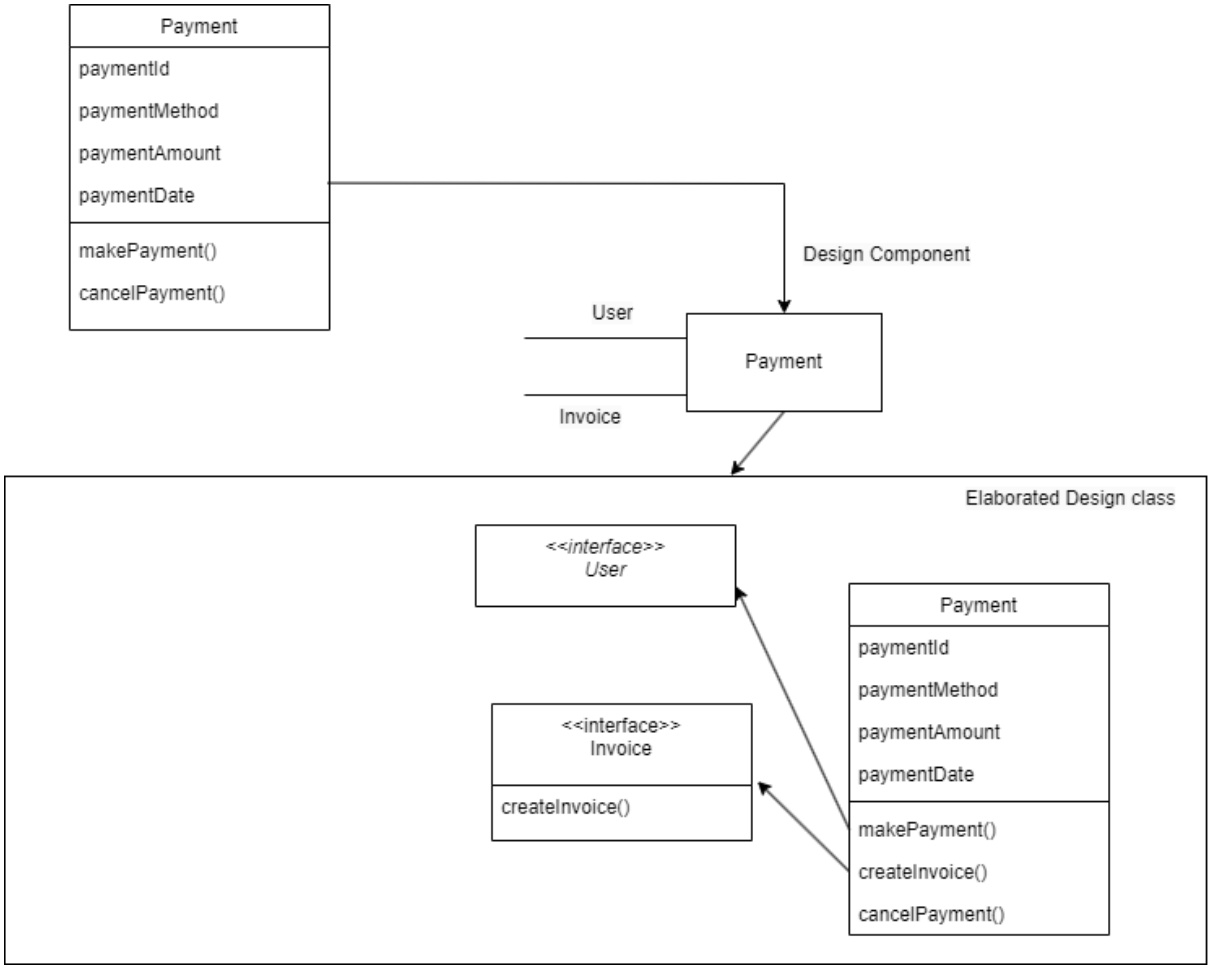
3.1.1 Job Poster



3.1.2 Messaging



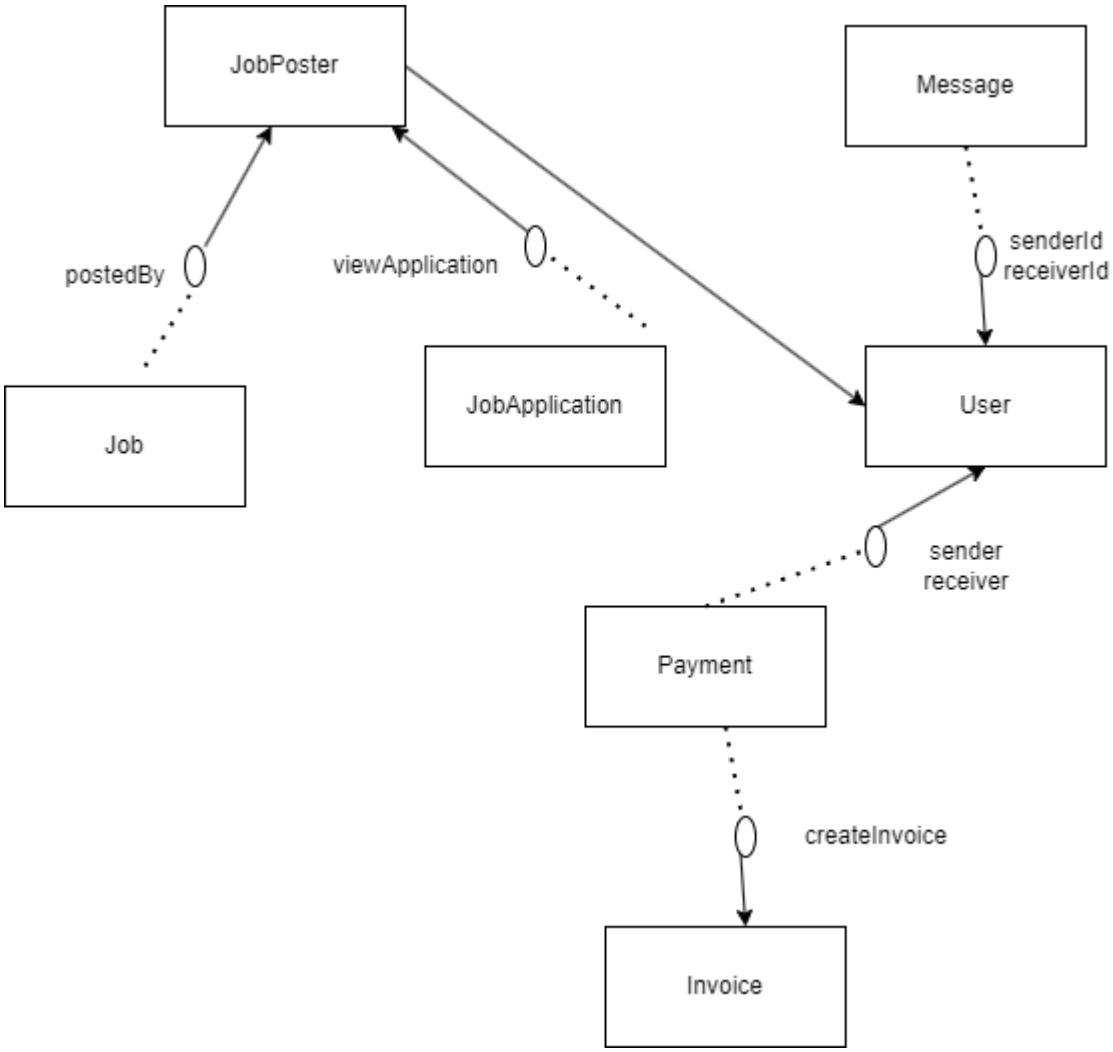
3.1.3 Payment


Payment
paymentId
paymentMethod
paymentAmount
paymentDate
makePayment()
cancelPayment()
Design Component
User
Payment
Invoice
Elaborated Design class
<<interface>>
User
Payment
paymentId
paymentMethod
paymentAmount
paymentDate
makePayment()
createInvoice()
cancelPayment()
<<interface>>
Invoice
createInvoice()

3.2 Message Passing

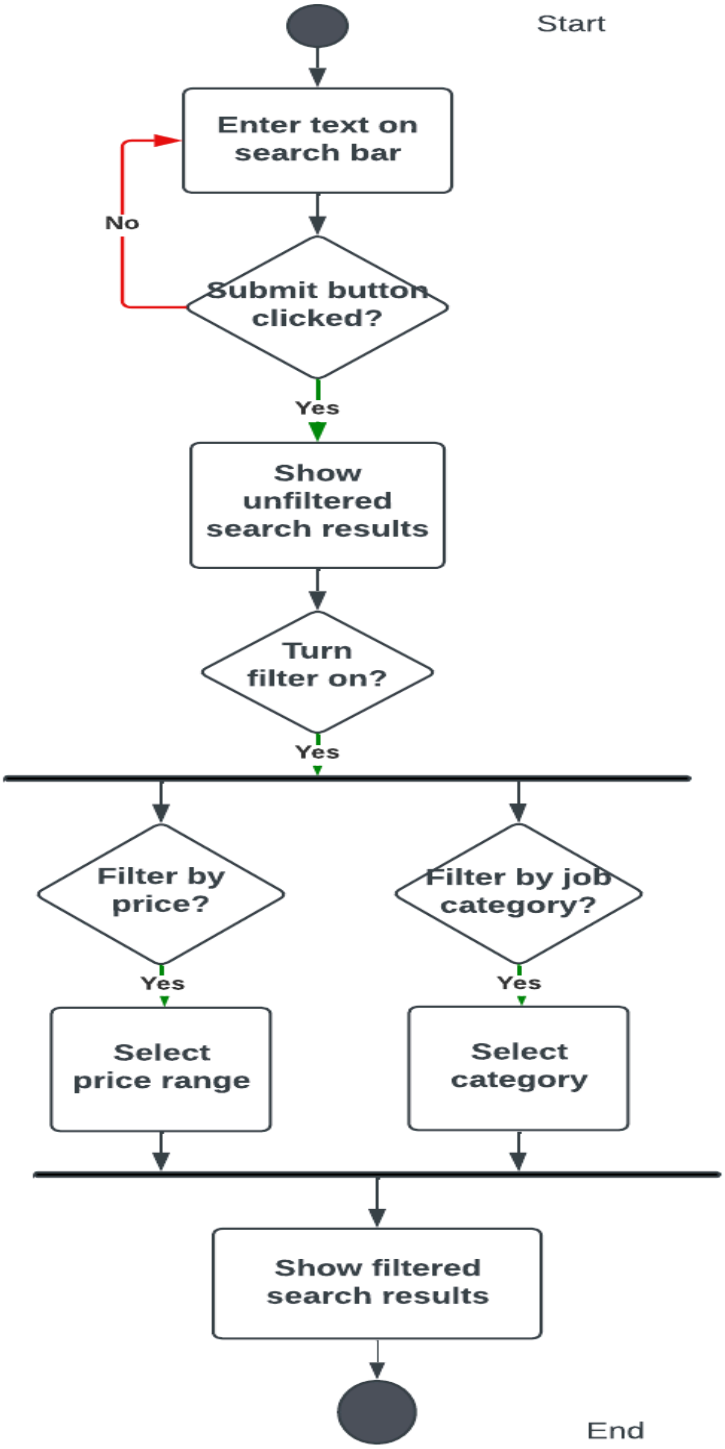
We used a collaboration diagram to show how analysis classes collaborate with one another. As our component-level design proceeded, it was useful to show the details of these collaborations by specifying the structure of messages that are passed between objects within a system. This design activity is used as a precursor to the specification of interfaces that show how components within the system communicate and collaborate.

The figure below illustrates a simple collaboration diagram for the CampusWorks app.

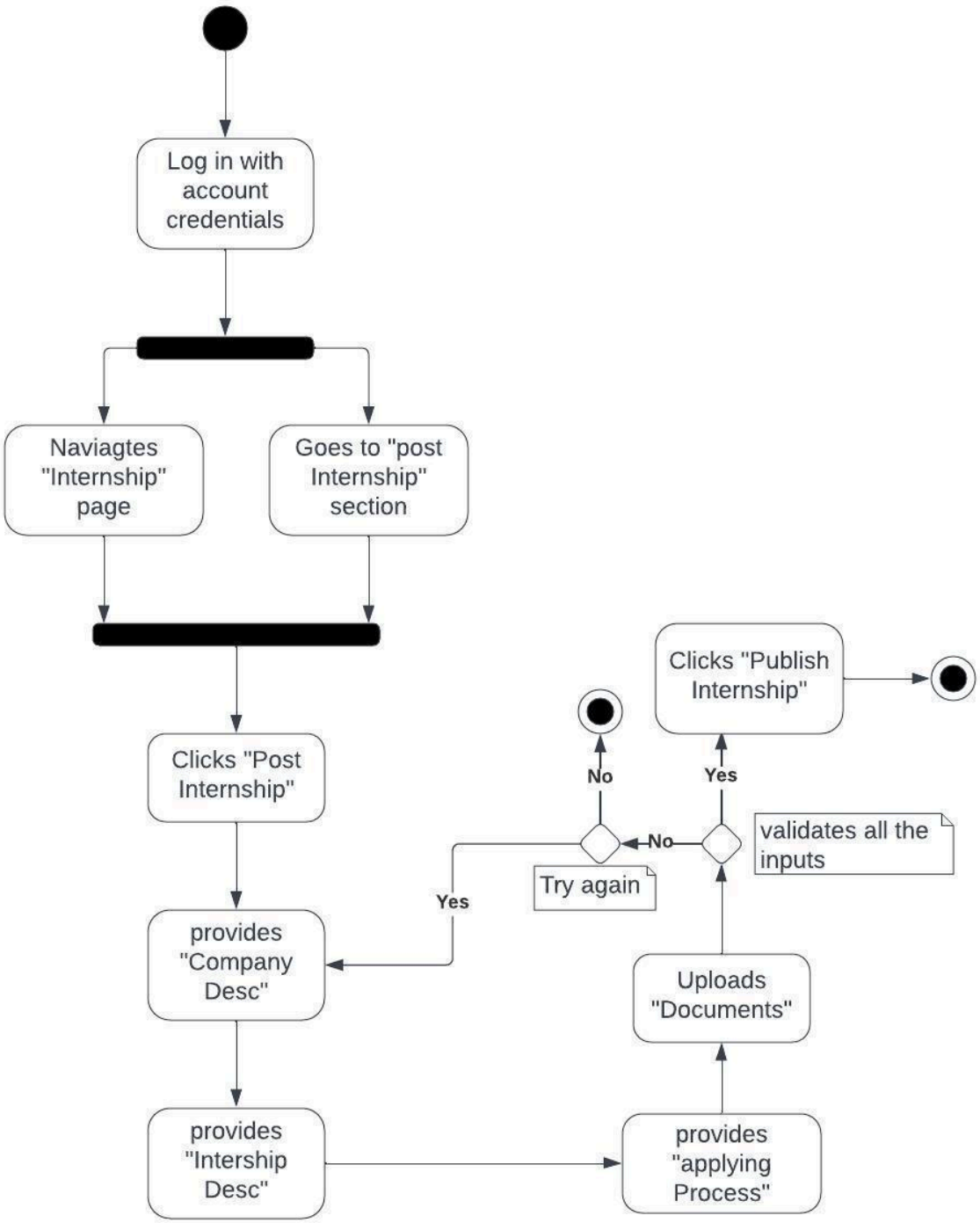


3.3 UML Activity Diagrams

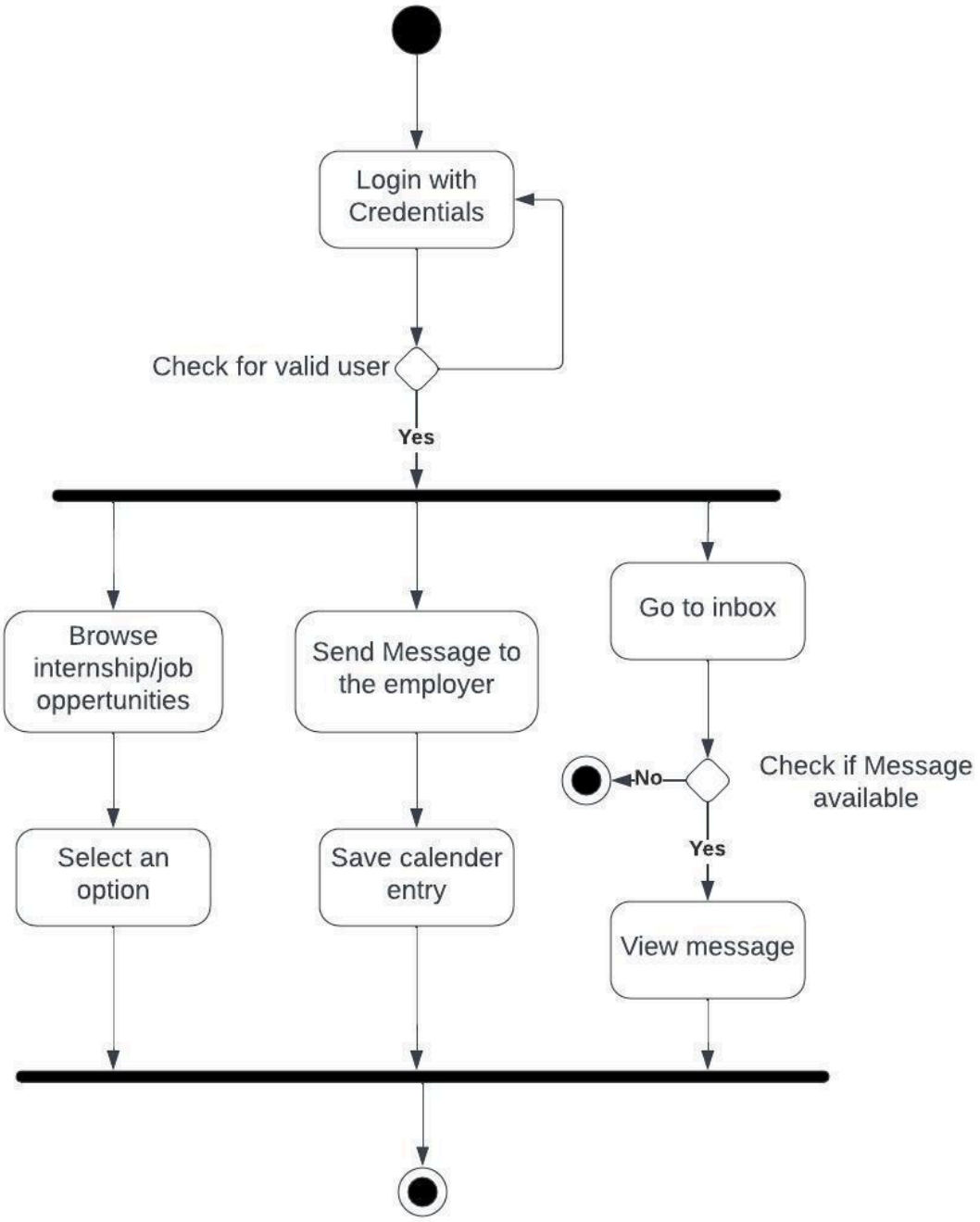
3.3.1 UML Activity Diagram for searchJob()



3.3.2 UML Activity Diagram for postInternship()

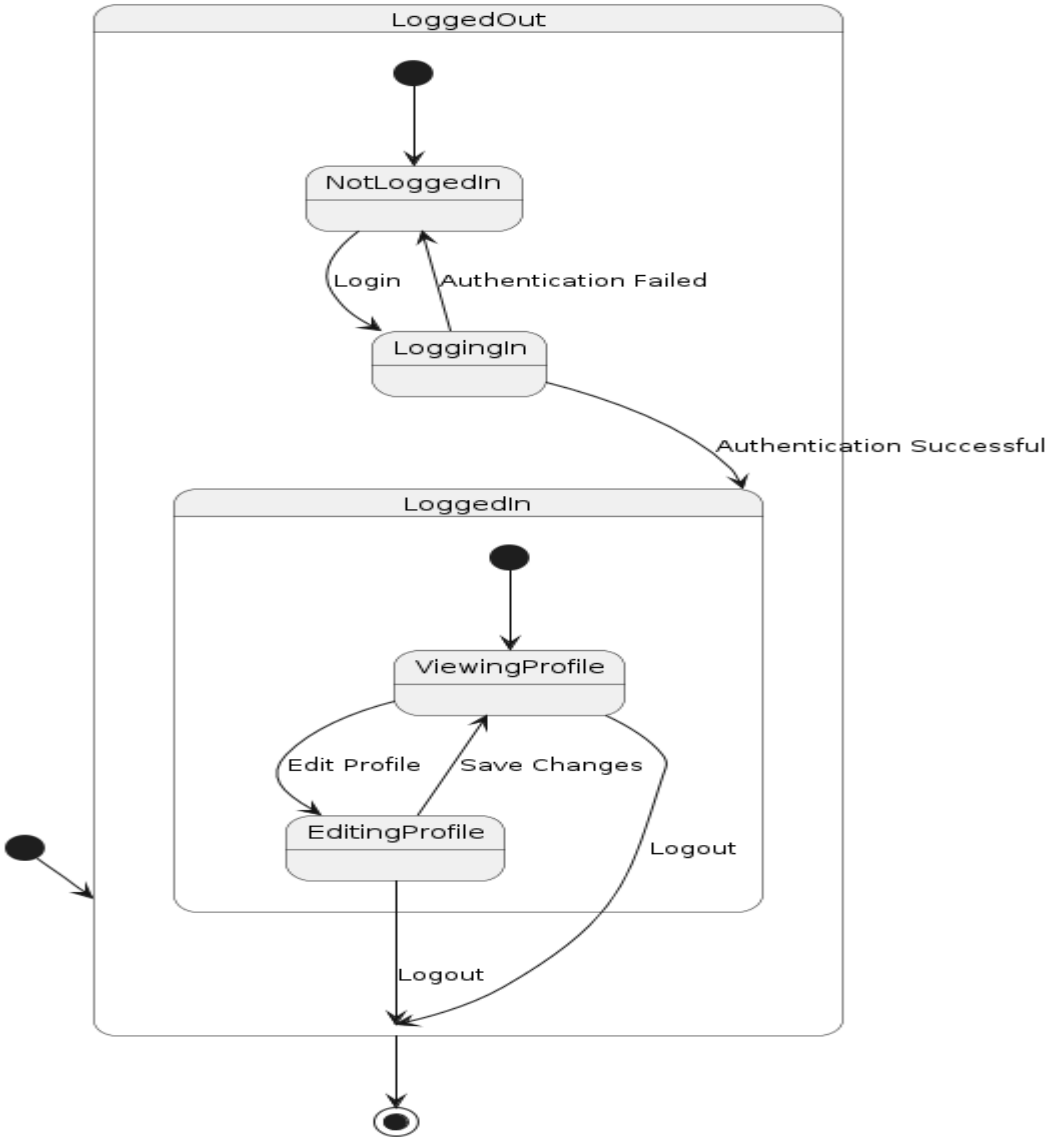


3.3.3 UML Activity Diagram for messageEmployer()

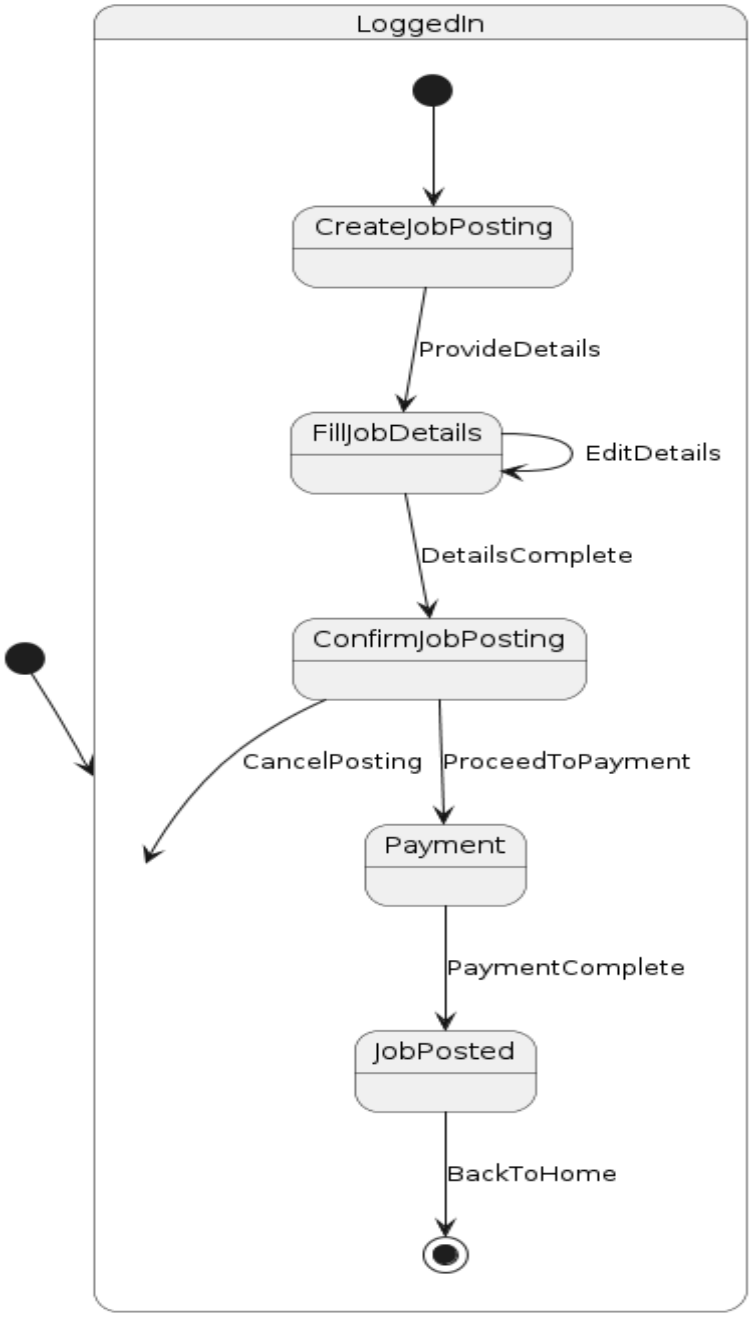


3.4 State Diagrams

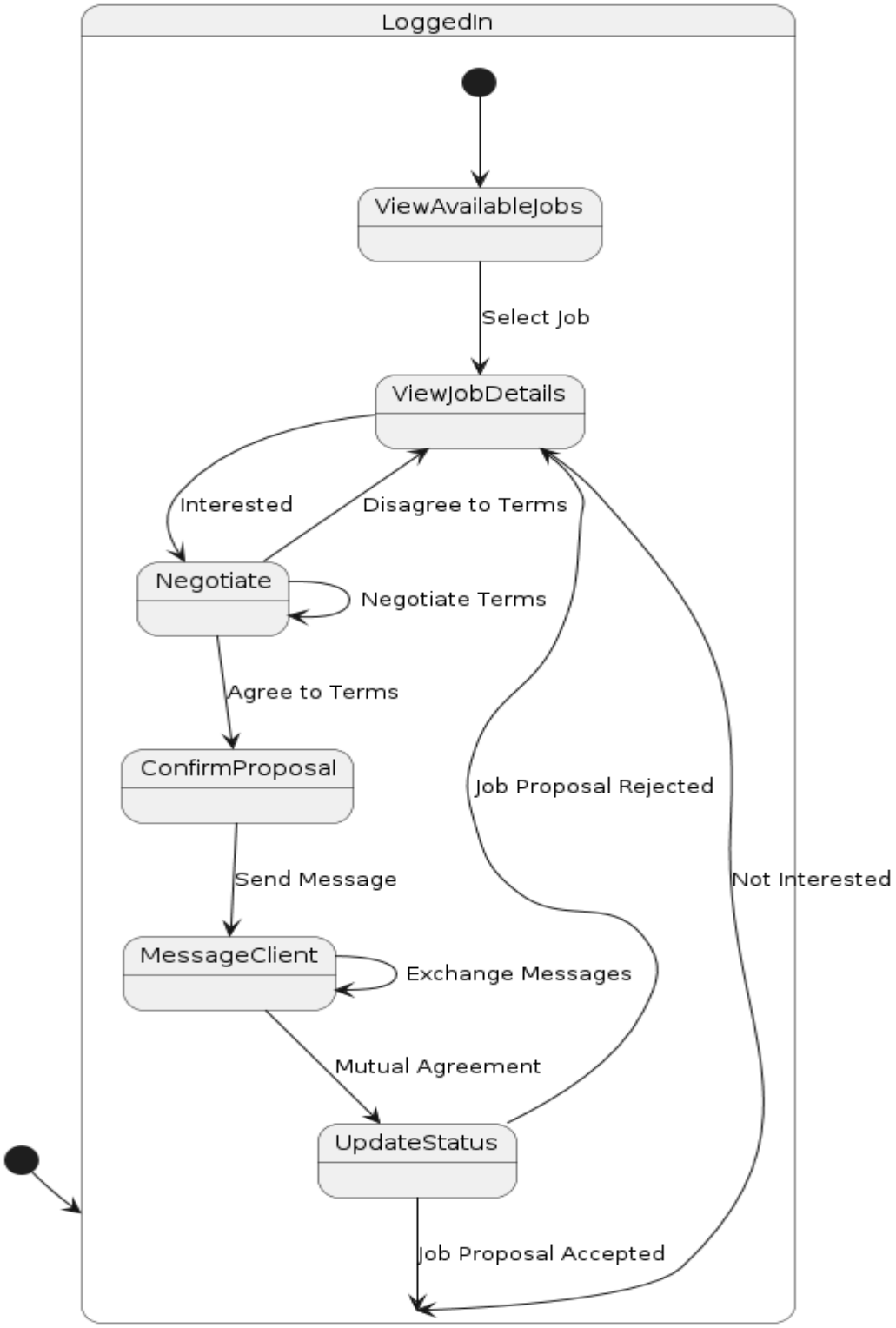
3.4.1 User Authentication State Diagram



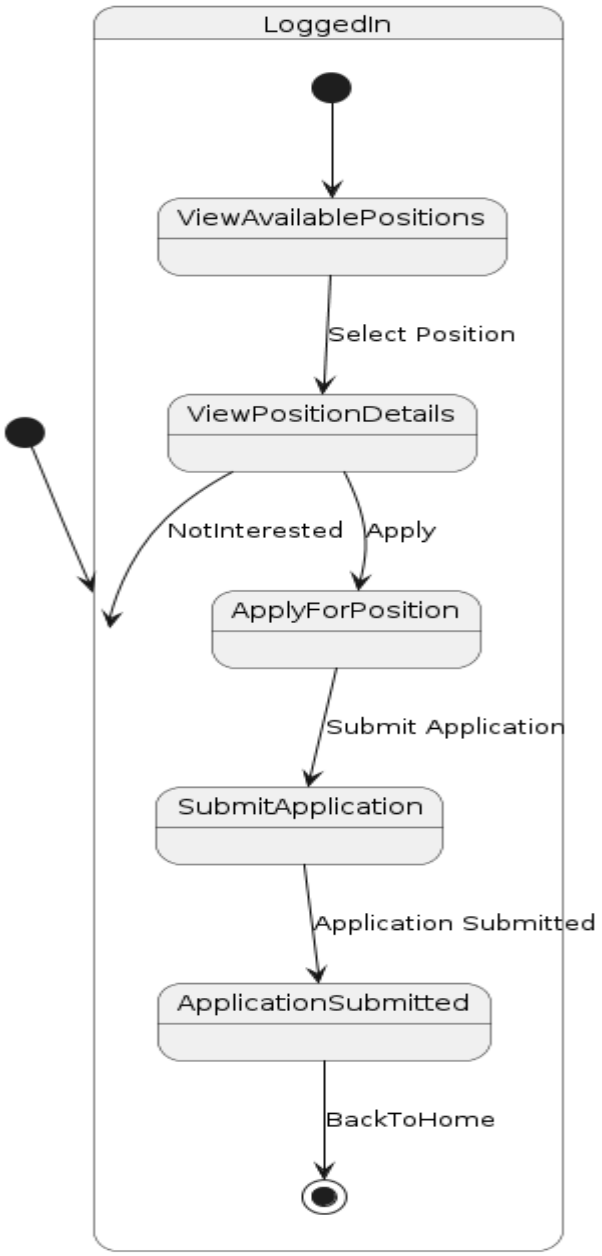
3.4.2 Job Posting State Diagram



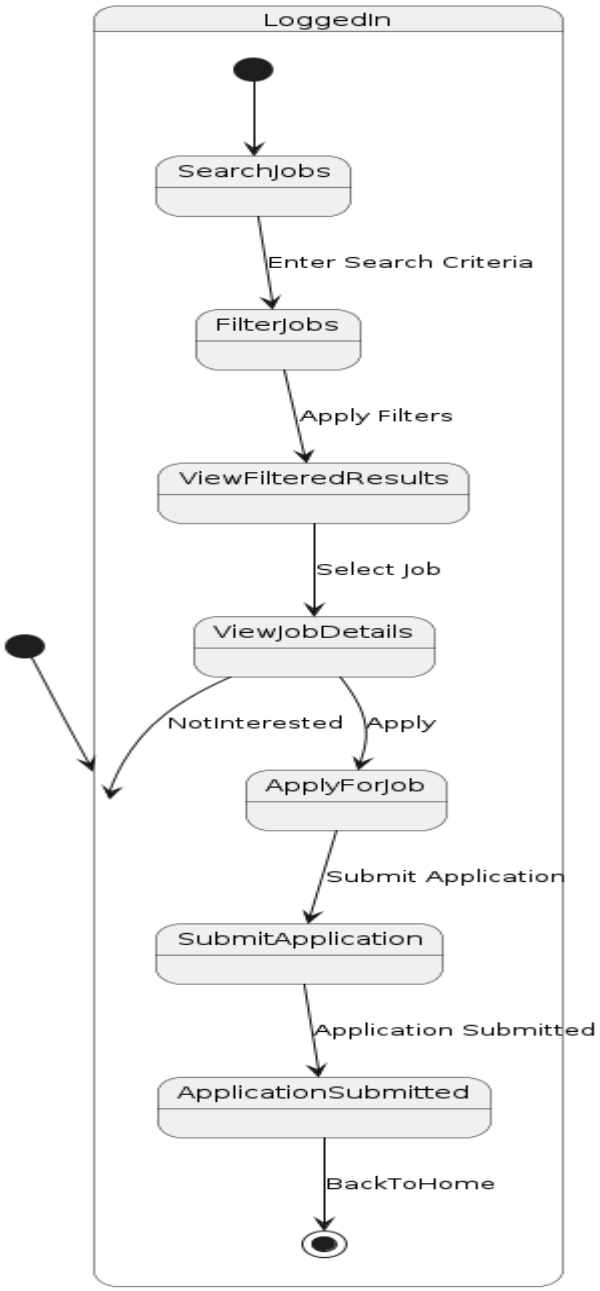
3.4.3 Messaging and Job Proposal State Diagram



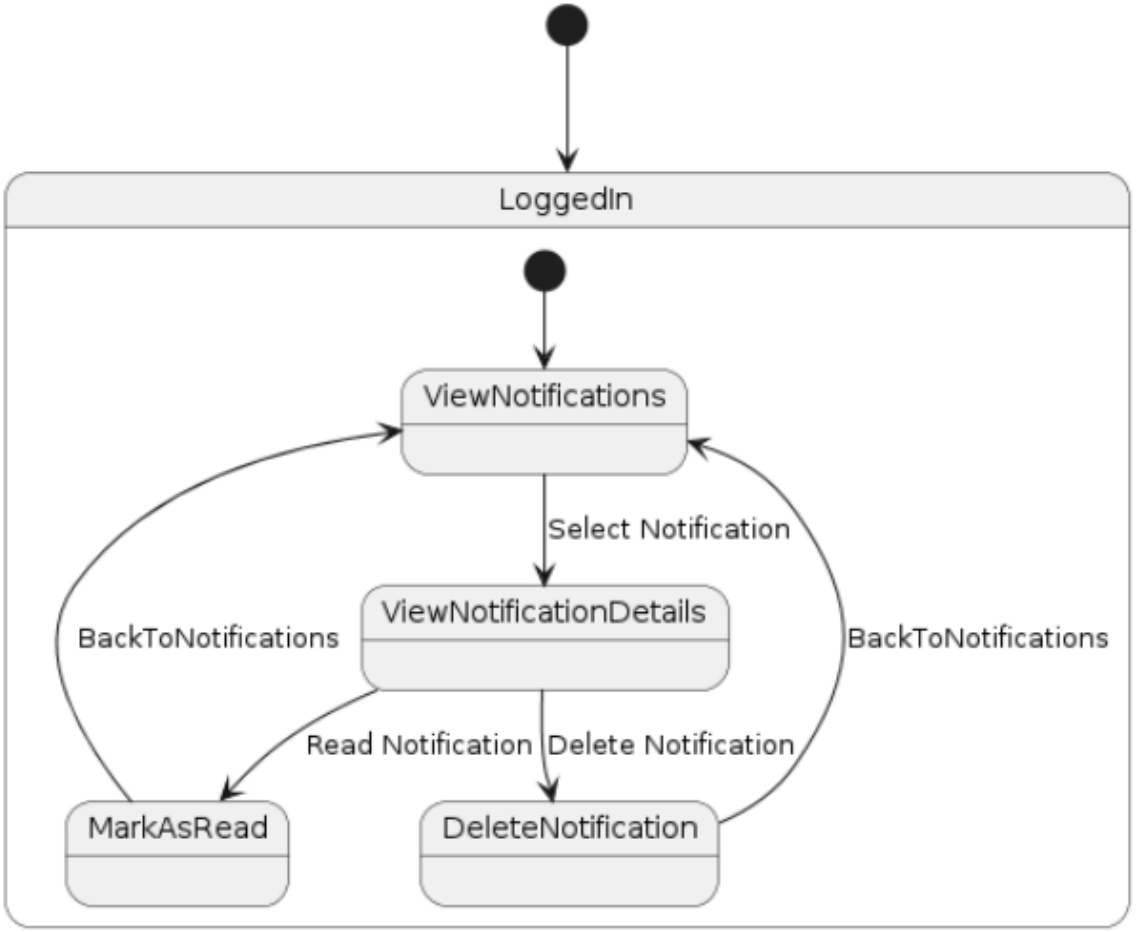
3.4.4 Job and Internship Application State Diagram



3.4.5 Search State Diagram



3.4.6 Notification State Diagram



3.5 Deployment Diagram

A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them. Deployment diagrams are typically used to visualize the physical hardware and software of a system. Using it we can understand how the system will be physically deployed on the hardware.

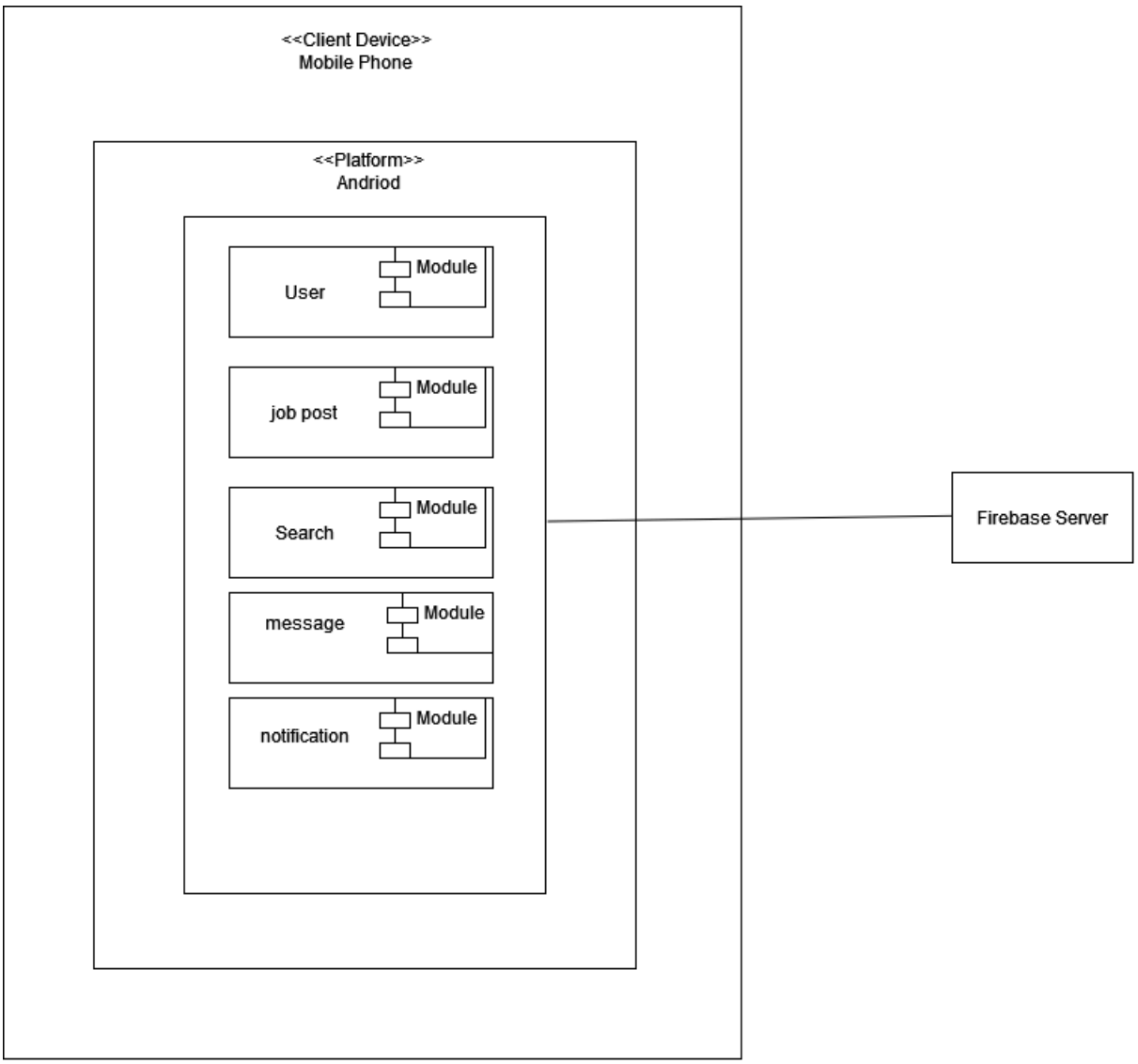
Deployment diagrams help model the hardware topology of a system compared to other UML diagram types which mostly outline the logical components of a system. Node is a computational resource upon which artifacts are deployed for execution. A node is a physical thing that can execute one or more artifacts. A node may vary in its size depending on the size of the project.

Node is an essential UML element that describes the execution of code and the communication between various entities of a system. It is denoted by a 3D box with the node name written inside of it. Nodes help to convey the hardware that is used to deploy the software. An association between nodes represents a communication path from which information is exchanged in any direction.

Here is the deployment diagram for the CampusWorks app.

Here,
<<Client Device>> Mobile Phone is a node. It represents a physical machine where the app runs.
<<Platform>> Android is also a node that represents the platform in which the app is going to be executed.
Node: <<Platform>> Android is nested into Node: <<Client Device>> Mobile Phone.

The 4 components are nested in the <<Client Device>> Mobile Phone node.
-user
-job post
-search
-message
-notification


<<Client Device>>
Mobile Phone
<<Platform>>
Andriod
User
Module
job post
Module
Search
Module
message
Module
notification
Module
Firebase Server

Chapter 4: User Interface Design

4.1 User Analysis

Users of CampusWorks can be anyone affiliated with the Islamic University of Technology (IUT), including:

- **Current Students:** Undergraduate and postgraduate students who are actively pursuing their studies at IUT.

- **Alumni:** Graduates of IUT who have entered their workplace and may be seeking freelance opportunities or internships to enhance their careers. Also they maybe offering opportunities to their juniors

- **Faculty Members:** Professors, lecturers, and other academic staff who may wish to offer mentorship, freelance opportunities, or internships to students and alumni.

- **Administrative Staff:** Non-academic staff members who may use the platform to post job openings or internships relevant to the IUT community.

User Demographics and Characteristics :

- **Age Range:** The users' age ranges between 20 and 60, with the majority likely falling between 21 and 50. This includes both young adults in their early career stages and more experienced professionals.

- **Gender:** The platform is intended for all genders, and no single gender is expected to dominate the user base.

- **Primary Language:** English is the primary spoken and written language among users. Therefore, all platform communications, instructions, and interfaces are in English.

User Education and Expertise

- **Education Level:** Users are expected to have at least some tertiary education, given that they are associated with IUT. This includes undergraduate, graduate, and postgraduate education levels.
- **Subject Matter Expertise:** Users are generally experts or possess a strong understanding of their respective fields of study or work, given their affiliation with a technical university.

- **Technology Proficiency:** While users are likely comfortable with using technology and digital platforms, they may not necessarily be experts in the underlying technologies that power the platform. However, they should be capable of navigating and using the platform's features with ease.

User Needs and Motivations

- **Current Students:**
    - Seeking internships and freelance jobs to gain practical experience.
    - Networking with alumni and faculty for career guidance and mentorship.
    - Looking for opportunities to apply their academic knowledge in real-world scenarios.

- **Alumni:**
    - Seeking freelance work or internships to pivot careers or gain additional experience.
    - Offering freelance jobs or internships to current students to give back to the community.
    - Networking with peers and current students for professional growth.

- **Faculty Members:**
    - Providing mentorship and guidance to students and alumni.
    - Offering internships or project opportunities related to their academic research or departmental needs.
    - Networking with alumni for potential collaboration on research or professional projects.

- **Administrative Staff:**
    - Posting job openings, internships, and other opportunities relevant to the IUT community.
    - Facilitating connections between students, alumni, and faculty members.
    - Managing and overseeing the platform to ensure it meets the needs of all users.

User Interaction with the Platform

- User wants the platform to be user-friendly with intuitive navigation and clear instructions. Users are expected to understand and follow given instructions easily.

- Users have basic knowledge of how to communicate with others through the platform, whether it be messaging, posting job listings, or applying for opportunities

- Users are capable of learning new features and functionalities through written materials, tutorials, or help sections provided on the platform.

4.2 Task Analysis

Task : User Registration

Subtasks :

- User enters their Full Name in the provided input field.
- User enters their Email address in the provided input field.
- User sets their Password in the provided input field.
- User selects their Department from a dropdown list.
- User selects their Batch from a dropdown list.
- User selects their User Category from a dropdown list.
- User clicks the Register button to submit their information.
- System verifies the validity of the entered email address.
- System verifies the entered password meets security criteria.
- If all information is valid, the system creates the account and logs the user in.
- System sends a confirmation email to the user's provided email address for verification purposes.
- If the user is already registered, they can click the Sign In link to proceed to the login page.

Task: User Login

Subtasks:

- User navigates to the Login screen.
- User enters their Email in the provided input field.
- User enters their Password in the provided input field.
- User clicks the Sign In button to submit their login information.
- System verifies the entered email and password against the stored user credentials.
- If the email or password is incorrect, the system displays an error message prompting the user to try again.
- If the credentials are correct, the system logs the user in.
- System redirects the user to the main landing page of the application.
- If the user forgets their password, they can click the Forgot Password link to reset their password via email.

Task: Send Job Request

Subtasks:

- User finds a job listing they are interested in.
- User clicks the Interested button.
- System logs the user's interest and increments the interest counter.
- User clicks the Message button to contact the job poster.
- System redirects the user to a messaging interface.
- User composes a message to the job poster.
- User clicks the Send button to submit the message.
- System delivers the message to the job poster.

Task: Search For Job

Subtasks:

- User opens the application.
- User navigates to the Search for Freelance Jobs section.
- User enters relevant keywords into the search bar.
- System displays a list of job listings matching the search criteria.
- User reviews the list of job listings.
- User clicks on a job listing to view more details.

Task: Message Job Poster

Subtasks:

- User navigates to a specific job listing.
- User clicks the Message button associated with the job poster's profile.
- System redirects the user to the messaging interface.
- User composes a message to the job poster.
- User clicks the Send button to submit the message.
- System delivers the message to the job poster.

Task: Post Job

Subtasks:

- User navigates to the Post Job section of the application.
- User enters the Job Title in the provided input field.
- User provides a detailed Job Description.
- User sets the Requirements for the job.
- User specifies the Duration and Budget for the job.
- User clicks the Post Job button to submit the job listing.
- System verifies the entered information.
- If all information is valid, the system publishes the job listing.

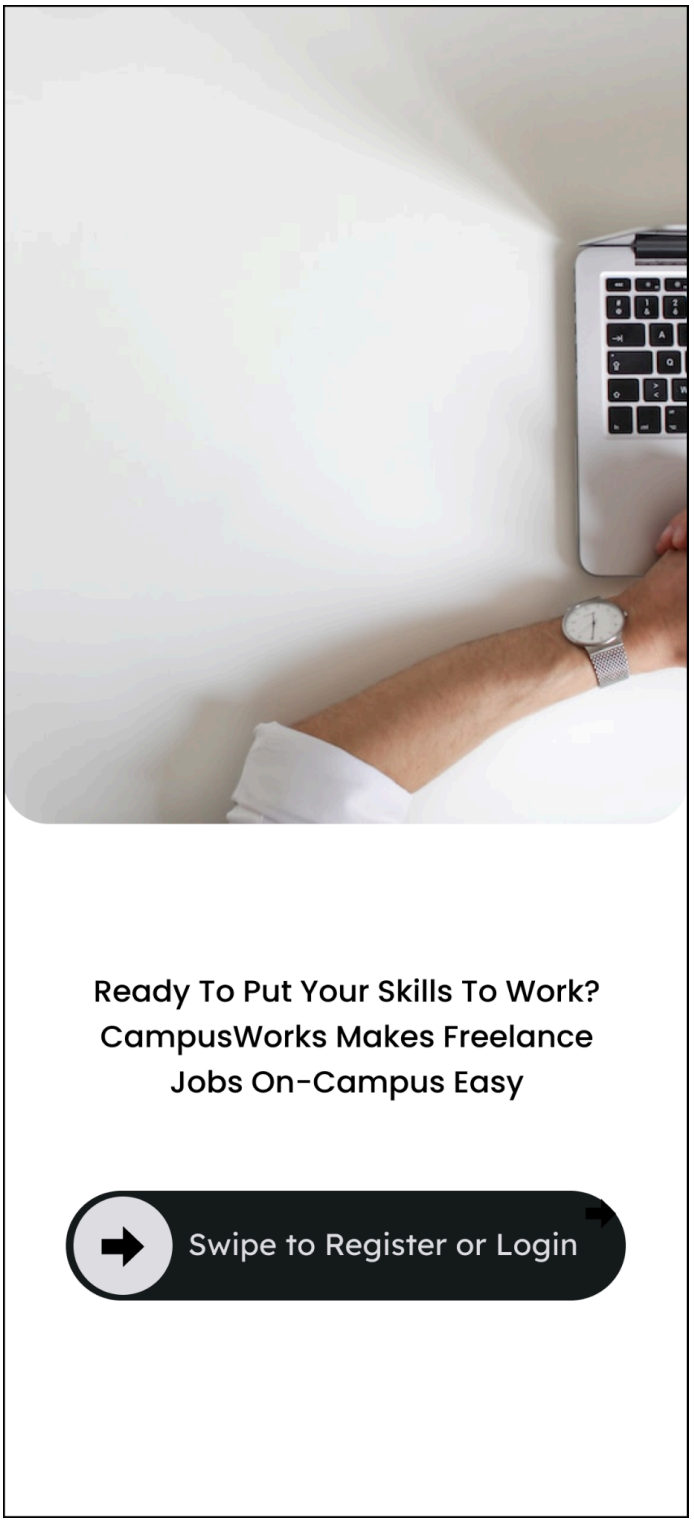
Task: Make Payment

Subtasks:

- User navigates to the Payment section.
- User enters their Payment Account No in the provided input field.
- User enters the Receiver Account No in the provided input field.
- User enters the CampusWorks Email associated with the transaction.
- User enters the Job ID related to the payment.
- User enters the Amount to be paid.
- User clicks the Make Payment button to submit the payment.
- System verifies the entered payment information.
- If all information is valid, the system processes the payment.

4.3 User Interfaces Mockup

Onboarding Screen - 1



Onboarding Screen - 2

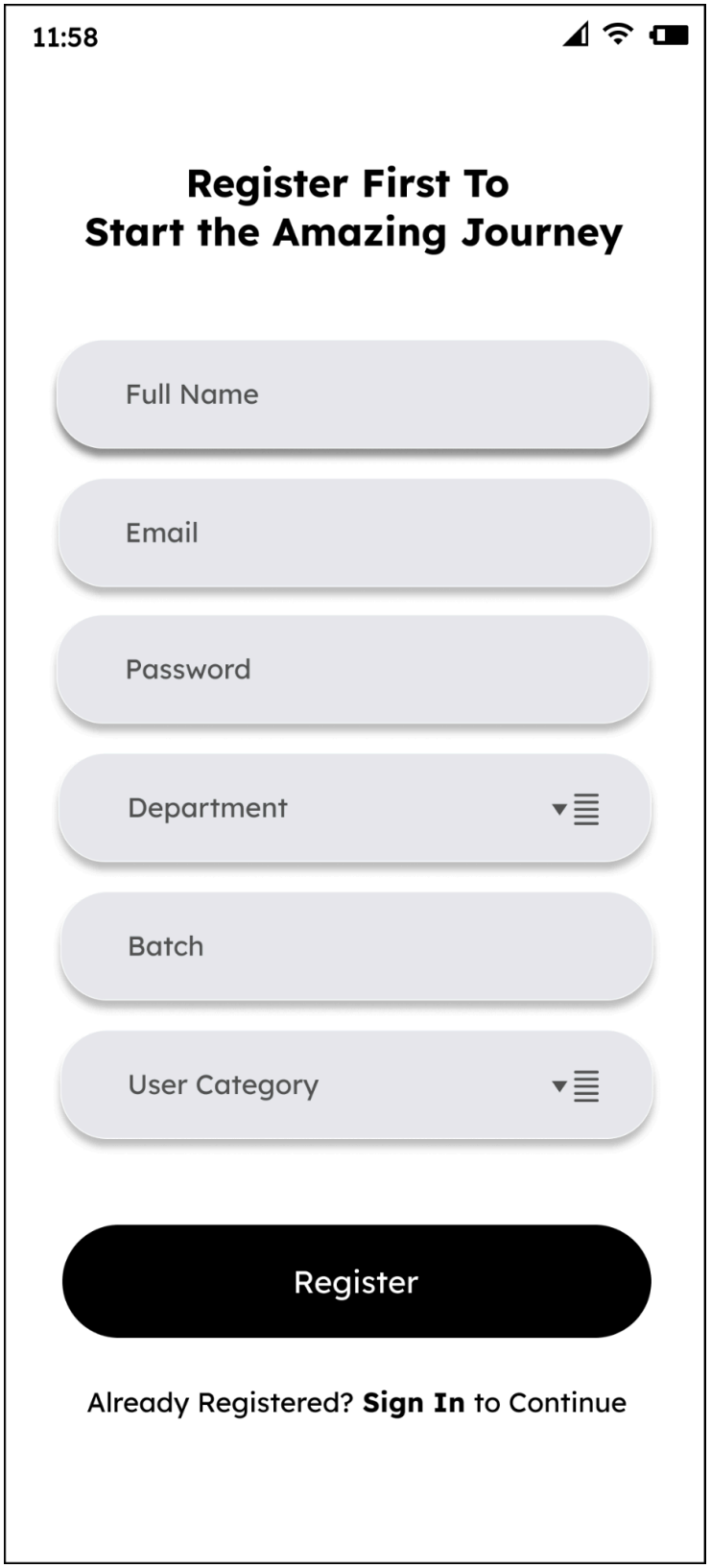
Welcome To

CampusWorks

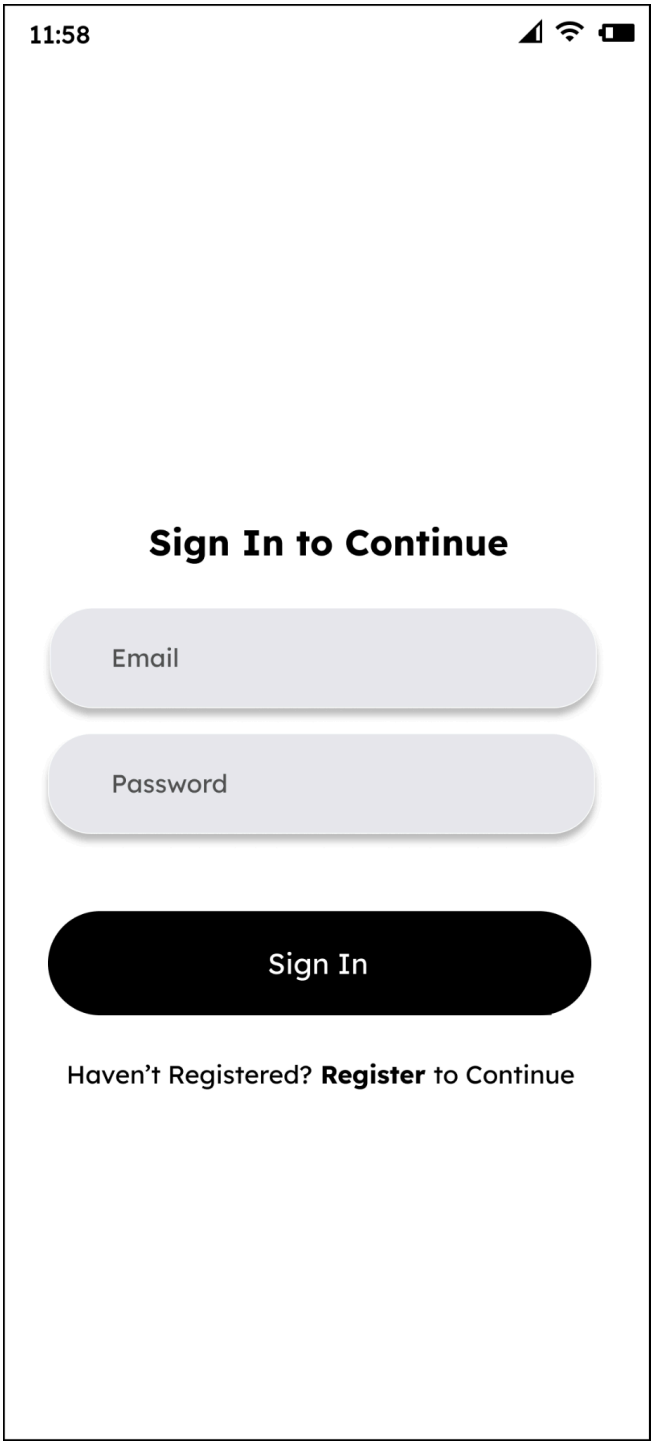
Log In

Register

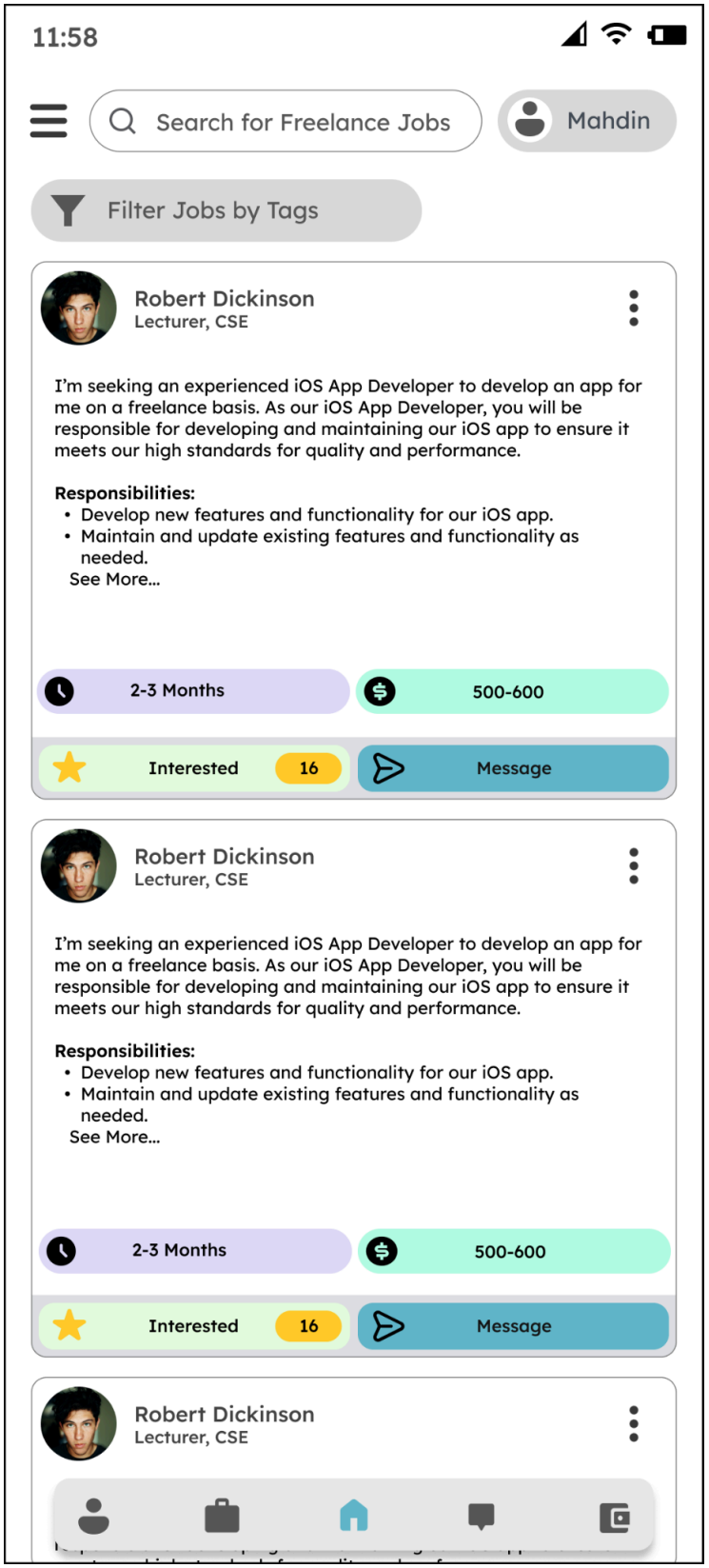
Sign Up Screen



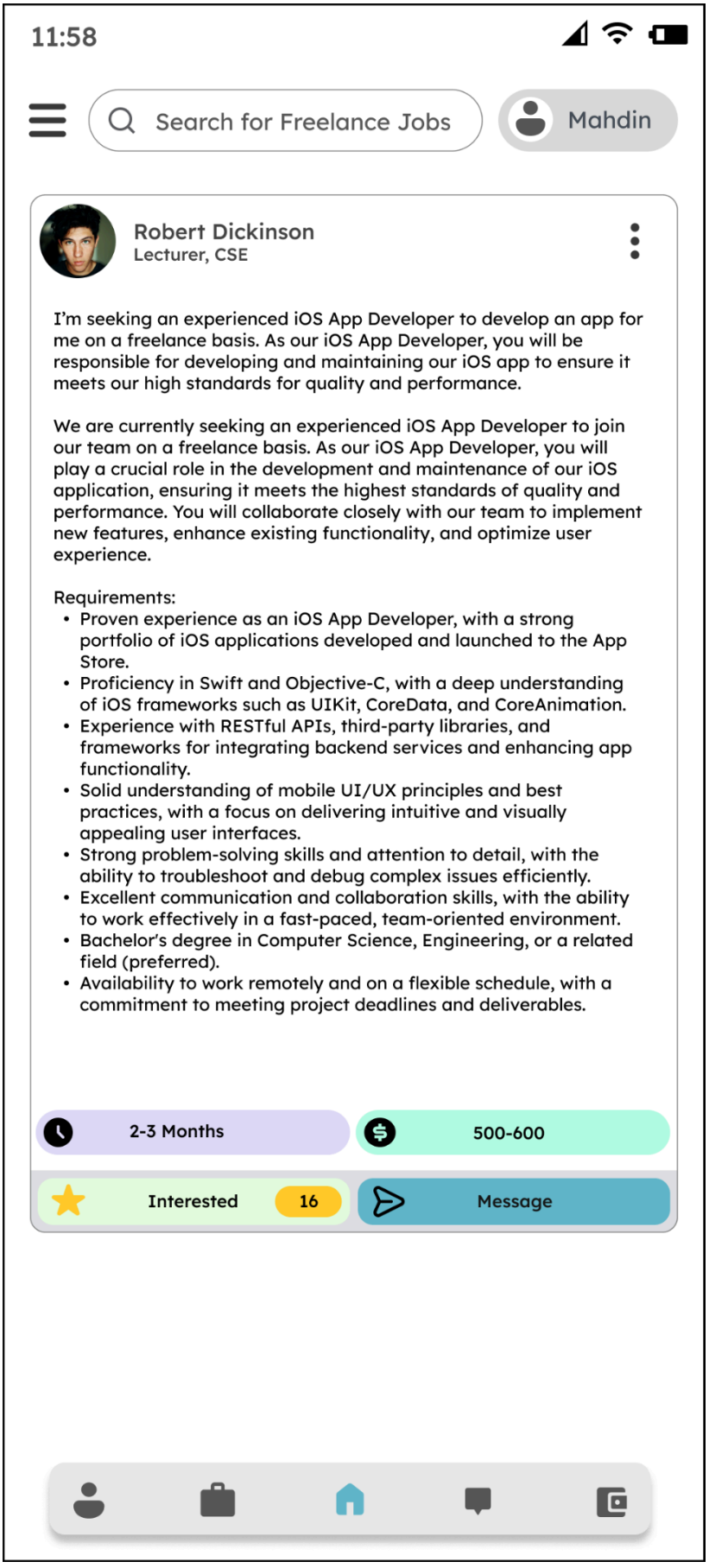
Log In Screen



Landing Page



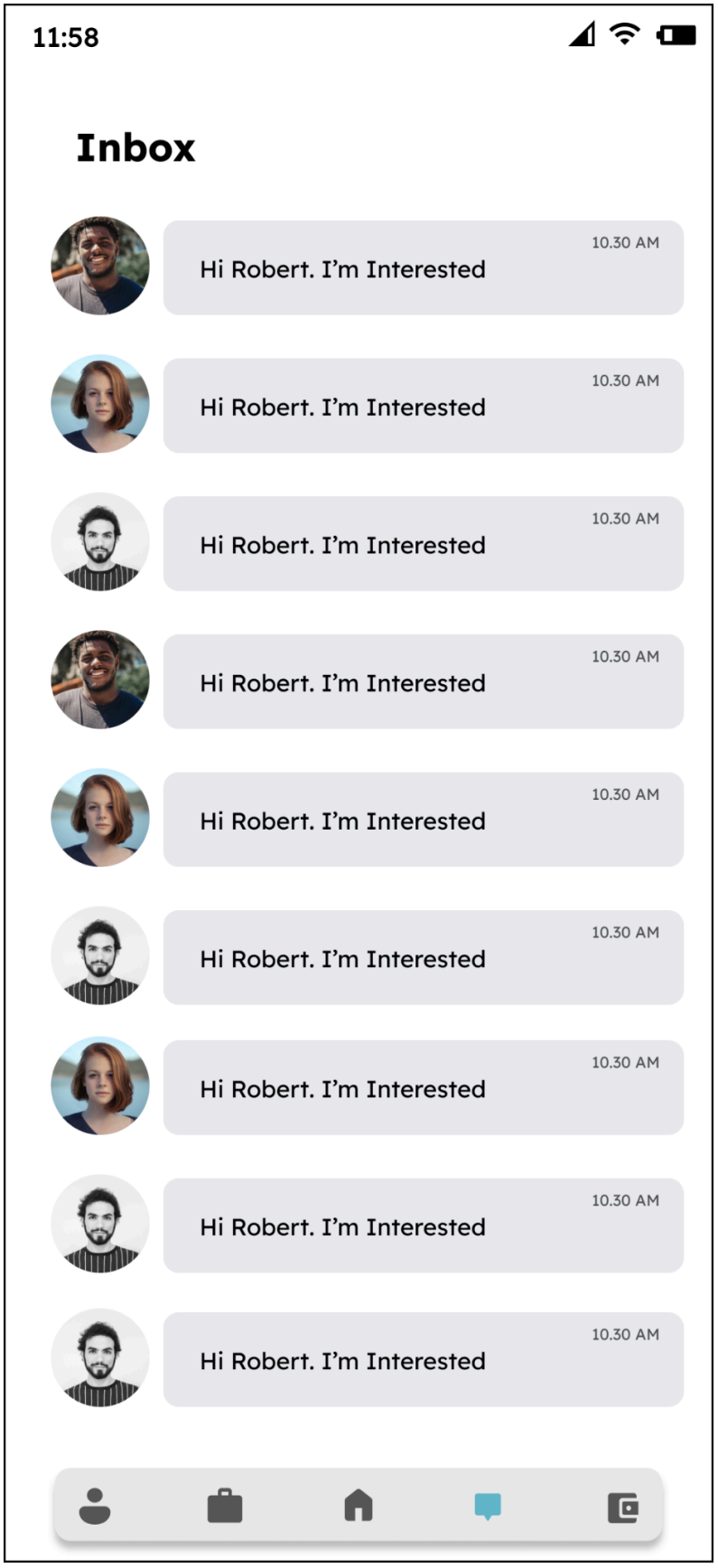
Detailed Job Screen



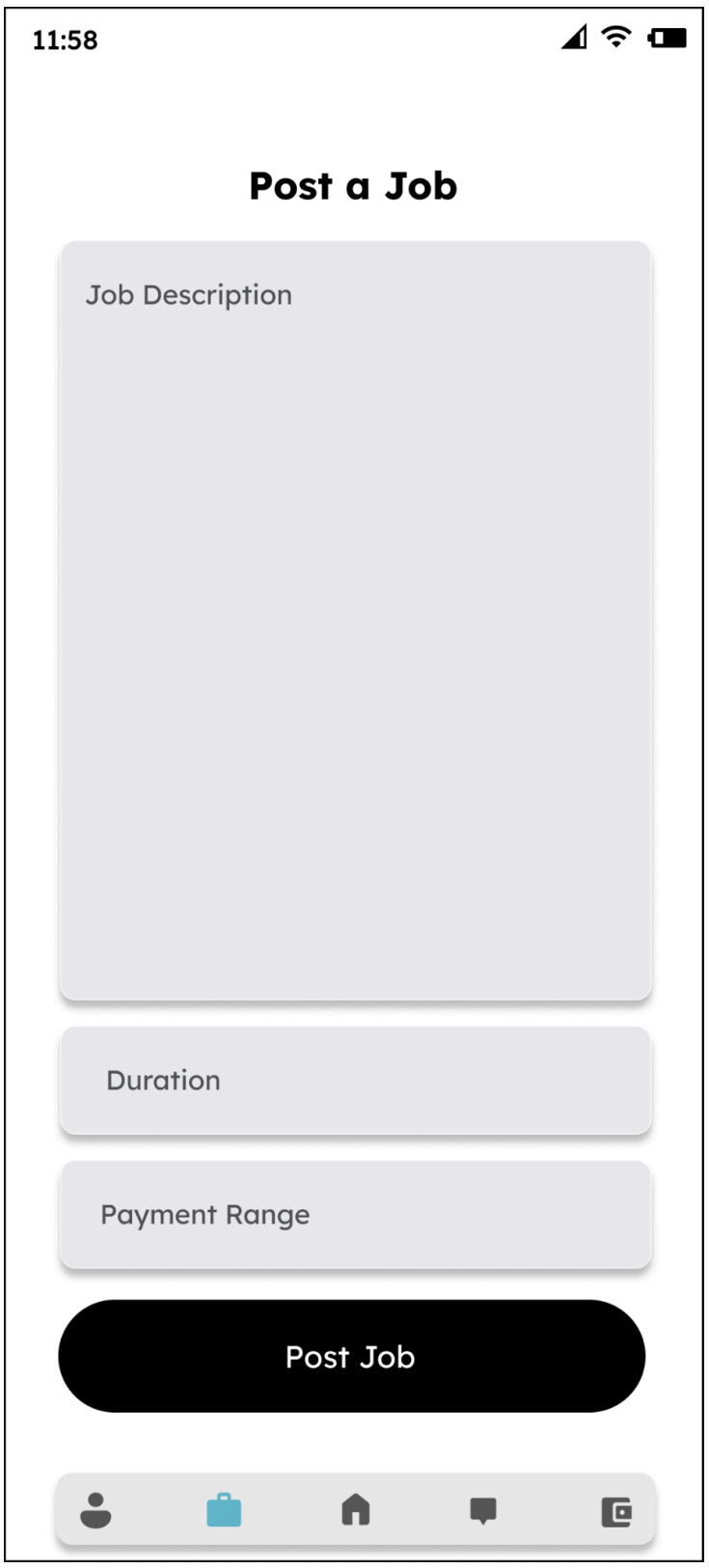
Chat Screen


11:58
Message Robert
Hi, Robert . I'm Interested
Hi, Good to Hear
Hi, Robert . I'm Interested
Hi, Good to Hear
Hi, Robert . I'm Interested
Hi, Good to Hear
Hi, Robert . I'm Interested
Hi, Good to Hear
Write your message

Inbox Screen



Post Job Screen



Payment Screen