|  |  | | | | |
| --- | --- | --- | --- | --- | --- |
|  | |  | | | |
| Project: | | Mobile Mech | | | |
| Team No.: | | Team #4 | | | |
| Class: | | CSE 3310; Fall 2024 | | | |
| Module: | | Final Term Project | | | |
| Deliverable: | | Final Term Project Document | | | |
| **Version:** | | | **[1.0]** | **Date:** | **[12/3/2024]** |

Team members are:

Sujana Kabir

Margaret Roche

Patrick Williams

Jonathan Hor

**Revision History**

| ***Version number*** | ***Date*** | ***Originator*** | ***Reason for change*** | ***High-level description of changes*** |
| --- | --- | --- | --- | --- |
| 1.0 | 12/3//2024 | Team 4 | Initial draft |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

TABLE OF CONTENTS

[**1. READ ME**](#_heading=h.2et92p0) 3

**2. SRA DOCUMENT**  4

**3. TEST PLAN DOCUMENT** 51

**4. USER MANUAL DOCUMENT** 68

[**5. SELECTED SOURCE CODE**](#_heading=h.4d34og8)83

READ ME

Team Members

Jonathan Hor

Sujana Kang

Patrick Williams

Margaret Roche

Student Emails

Jonathan Hor - jxh1278@mavs.uta.edu

Sujana Kang - sxk4940@mavs.uta.edu

Patrick Williams - pxw0029@mavs.uta.edu

Margaret Roche - mlr9944@mavs.uta.edu

Contact Phone Number

Patrick Williams - 812-530-7417

Margaret Roche - 817-703-3960

Exceptions

* Due to the complexity of altering data live in the app, then saving the data to firebase over the old data, profile editing does not work.
* Because of the complexity of coding it, the messaging feature does not function. We wrote code several times, attempting to get it working, and did not succeed.
* As a result of messaging not working; reporting, payment, and calendar have unfinished aspects. All 3 were intended to be implemented within the messaging system.

Valid Login

*Customer Profile*

Username - user1@test.com

Password - usertest1!

*Mechanic Profile*

Username - mech1@test.com

Password - mechtest1!

|  |  | | | |
| --- | --- | --- | --- | --- |
|  | |  | | |
| Project: | | Mobile Mech (MM) | | |
| Team No.: | | 4 | | |
| Class: | | CSE 3310; Fall 2024 | | |
| Module: | | System Requirements Analysis (SRA) | | |
| Deliverable: | | SRA Document | | |
| **Version:** | | **[1.0]** | **Date:** | **[10/31/2024]** |

Contributors:

Sujana Kabir

Margaret Roche

Patrick Williams

Jonathan Hor

**Revision History**

| ***Version number*** | ***Date*** | ***Originator*** | ***Reason for change*** | ***High-level description of changes*** |
| --- | --- | --- | --- | --- |
| 1.0 | 10/31/2024 | Team#4 | Initial draft | Added all requirements for features |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

TABLE OF CONTENTS

[**1. Introduction and Project Overview 4**](#_heading=h.3znysh7)

[**2. Objectives 5**](#_heading=h.2et92p0)

[2.1 BUSINESS Objectives 5](#_heading=h.tyjcwt)

[2.2 SYSTEM Objectives 7](#_heading=h.3dy6vkm)

[**3. Project Context Diagram 8**](#_heading=h.1t3h5sf)

[**4. Systems Requirements 9**](#_heading=h.4d34og8)

[4.1 “Registration and Login” Requirements 9](#_heading=h.2s8eyo1)

[4.2 “User Profile” Requirements 12](#_heading=h.17dp8vu)

[4.3 “Communication” Requirements 14](#_heading=h.3rdcrjn)

[4.4 “Mechanic Profile” Requirements 18](#_heading=h.26in1rg)

[4.5 “Reporting” Requirements 20](#_heading=h.lnxbz9)

[4.6 “Payment” Requirements 21](#_heading=h.35nkun2)

[4.7 “Appointment” Requirements 24](#_heading=h.1ksv4uv)

[4.8 “Search” Requirements 25](#_heading=h.44sinio)

[**5. Software Processes and Infrastructure 27**](#_heading=h.z337ya)

[5.1 Hardware and Infrastructure 27](#_heading=h.3j2qqm3)

[5.2 UML Diagrams 28](#_heading=h.1y810tw)

[5.2.1 UML Diagram: “System” 28](#_heading=h.lwngx53wqh4e)

[5.2.2 UML Diagram: “Registration and Sign-in” 29](#_heading=h.578owaxa7p5e)

[5.2.3 UML Diagram: “User Profile” 30](#_heading=h.l5su4k8972n3)

[5.2.4 UML Diagram: “Communication” 31](#_heading=h.h2iqve1vhyyc)

[5.2.5 UML Diagram: “Mechanic Profile” 32](#_heading=h.rj7maqhuq7do)

[5.2.6 UML Diagram: “Reporting” 33](#_heading=h.xnr1zxyyr7i)

[5.2.7 UML Diagram: “Payment” 34](#_heading=h.7snodb4lskre)

[5.2.8 UML Diagram: “Appointment” 35](#_heading=h.qn8duq939pvw)

[5.2.9 UML Diagram: “Search” 36](#_heading=h.ogg9sogc65gm)

[5.3 Conceptual Data Model 37](#_heading=h.4i7ojhp)

[5.4 Screen Shots 38](#_heading=h.2xcytpi)

[5.4.1 Screen “Start Screen” 38](#_heading=h.lhof2wkmlx19)

[5.4.2 Screen “User Registration/Login Screen” 39](#_heading=h.xpsgg2yfczet)

[5.4.3 Screen “Mechanic Registration/Login Screen” 40](#_heading=h.t3k1abeqjkwe)

[5.4.4 Screen “User Registration Screen” 41](#_heading=h.e6l1xqxozyxm)

[5.4.5 Screen “User Registration Info Collection Screen” 42](#_heading=h.fp58zjqblj65)

[5.4.6 Screen “User Login Screen” 43](#_heading=h.m4j7l3wo4bro)

[5.4.7 Screen “Mechanic Registration Info Collection Screen” 44](#_heading=h.mly2olx8xrbu)

[5.5 Test Plan 44](#_heading=h.1ci93xb)

[**6. Assumptions and Constraints 45**](#_heading=h.3whwml4)

[6.1 ASSUMPTIONS 45](#_heading=h.2bn6wsx)

[6.2 CONSTRAINTS 45](#_heading=h.qsh70q)

[6.3 Out-of-Scope Material 45](#_heading=h.3as4poj)

[**7. Delivery and Schedule 46**](#_heading=h.1pxezwc)

[**8. Stakeholder Approval Form 47**](#_heading=h.49x2ik5)

[**Appendix: 48**](https://docs.google.com/document/d/1iMuyhsYjbGwrR7T57WDA45VjU6IdpbSa/edit#heading=h.2p2csry)

# 1. Introduction and Project Overview

The Mobile Mech App is an innovative Android application designed to connect users with professional mechanics, offering a convenient and efficient way to access automotive repair and maintenance services. Inspired by healthcare apps like ZocDoc or Teledoc that provide "on-demand" services, the Mobile Mech App aims to bring similar convenience to the automotive repair industry.

The android project will consist of 8 different components: Registration and Sign in, User Profile, Communication, Mechanic Profile, Reporting, Payment, Appointment, Search. When the user logs in to the application, they will be prompted to register or sign in. If they are an existing user, the user will simply sign in using their email and password. Otherwise the user will have to register. There will be both user and mechanic profiles so that both parties can have a bit of an overview. There will be ways of communication for the user with the mechanic. In the app, there will be an option to transact money from the user to the mechanic. Similar to ride apps, there will be reviews available of the mechanic so that the user can choose whom to contact best.

# 2. Objectives

## 2.1 BUSINESS Objectives

The following is a list of business objectives:

**Objective 1**:Registration and Login: Users and Mechanics must register before accessing the system. The following information is required for user registration:

* Email Address
* Password
* First name
* Last name
* Area of Residence

The following information is required for mechanic registration:

* Email address
* Password
* First name
* Last name
* Service area
* Specialties/Skills

Once registered, the user must login to access the system.

**Objective 2**:User Profile: Users will be able to view and modify their profile as needed, such as their information and area of residence to ensure applicable mechanics are recommended to them. The user profiles will be able to be viewed by mechanics once communication has been started by the user.

**Objective 3**: Mechanic Profile: Mechanics will be able to view and modify their profile as needed, such as their information, service area, and specialties/skills that they offer. The mechanic profiles will be able to be viewed by users who are deciding who they want to ask to perform some service. The mechanic profiles will have reviews associated with them in order to better educate users on the quality of work that the mechanic typically delivers.

**Objective 4**: Communication: Users and mechanics will use the communication feature in order to communicate with one another. Acts as a messaging service and is the starting point of creating appointments, payments to and from, and other features.

**Objective 5**: Reporting: Users and mechanics will be able to report inappropriate things that they see, whether in other’s individual profiles, messages sent in the communication feature, or actions witnessed in person. The reporting feature shall be present in multiple areas of the app, including in the communication section and the profiles section.

**Objective 6**: Payment: Mechanics will be able to request payment for their services, and users will be able to make those payments. Payments will originate in the communication section of the app and be handled by the system.

**Objective 7**: Appointments: Users will be able to request appointments, view set appointments, view availability of mechanics, and remove existing appointments in this section of the app. The appointments should be shown to the user in an aesthetically pleasing nature that makes it easy to see when appointments are and what availability of mechanics are.

**Objective 8**: Search: Users will be able to search for mechanics corresponding to their requested services. The results of the search should include the mechanic’s name, reviews, service, service area, and have a button to view availability.

## 2.2 SYSTEM Objectives

The following is a list of system objectives:

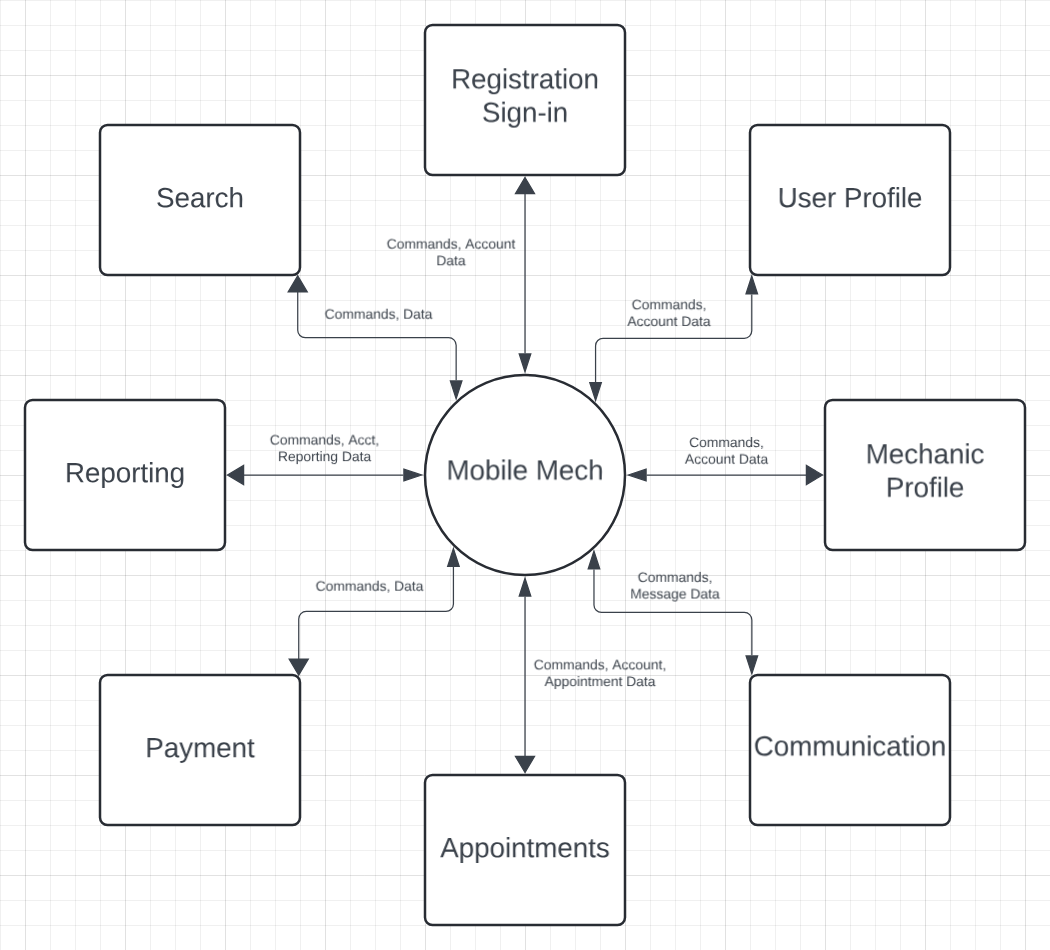
**Objective 1**: The system will be an Android-based Mobile system

**Objective 2**: ReactNative will be utilized to develop

**Objective 3**: The system will offer a user-friendly interface to enhance user experience

**Objective 4**: The system will be able to add new features in the future based on user preference and industry demand.

# 3. Project Context Diagram



# 4. Systems Requirements

{**Use the designated Requirements Form and insert all forms below in its related section.**}

## 4.1 “Registration and Login” Requirements

| **Requirement Title:** | Registration and Login |
| --- | --- |
| **Sequence No:** | 001 |
| **Short Description:** | Registering new users |
| **Description:** | Users and mechanics must register accordingly before accessing the application. For regular consumers, the following information will be collected:   * Name {First, Last name} * Email address * Password * Area   For mechanics, the following information will be collected:   * Name {First, Last name} * Email Address * Password * Specialities (skills) * Service Area   Users can press:   * “I’m a mechanic” button * “I’m a user” button * Submit * Cancel * Exit screen |
| **Pre-Conditions:** | * App must be loaded in order to process the request * Duplicate registration (matching emails) is not allowed |
| **Post Conditions:** | * Account created successfully, information saved |
| **Other Attributes:** | * None |

| **Requirement Title:** | Registration and Login |
| --- | --- |
| **Sequence No:** | 002 |
| **Short Description:** | Logging in existing users |
| **Description:** | Users must be authenticated before accessing the app. The user must enter:   * Email address * Password   The user can press:   * Submit * Forgot password * Cancel * Exit screen |
| **Pre-Conditions:** | * App must be loaded in order to make requests * User must already be registered |
| **Post Conditions:** | * User can access the app from the perspective of their role (Mechanic // User) |
| **Other Attributes:** | * None |

| **Requirement Title:** | Registration and Login |
| --- | --- |
| **Sequence No:** | 003 |
| **Short Description:** | Forgot Password |
| **Description:** | Users can request to reset their password. The user must enter:   * Email address   An email will be sent to the user’s email address. Once the user clicks on the link in the email, the user must enter:   * New password   The user can press:   * Submit * Cancel * Exit screen |
| **Pre-Conditions:** | * App must be loaded in order to make requests * Email must exist and be associated with an account |
| **Post Conditions:** | * Password changed successfully, new password set |
| **Other Attributes:** | * None |

## 

## 4.2 “User Profile” Requirements

| **Requirement Title:** | User Profile |
| --- | --- |
| **Sequence No:** | 001 |
| **Short Description:** | Profile of customers searching for Mechanics |
| **Description:** | Users can customize and view a profile of themselves. The default viewing of a profile is static. It displays:   * User name * User location (approximate - by city) * User profile picture   From the profile, the user can choose to:   * Change user name * Change user location * Change user profile picture * Change user password |
| **Pre-Conditions:** | * App must be loaded in order to make requests * Email must exist and be associated with an account * Password must be set |
| **Post Conditions:** | * Desired information was successfully updated and is displayed accordingly |
| **Other Attributes:** | None |

| **Requirement Title:** | User Profile |
| --- | --- |
| **Sequence No:** | 002 |
| **Short Description:** | Altering profile of customers searching for Mechanics |
| **Description:** | Users can customize and view a profile of themselves. Users must be authenticated before altering their profile. The user must enter:   * Password   After authenticating, users can alter:   * User name * User location * User profile picture * User password |
| **Pre-Conditions:** | * App must be loaded in order to make requests * Email must exist and be associated with an account * Password must be set |
| **Post Conditions:** | * User profile information altered successfully |
| **Other Attributes:** | None |

## 

## 4.3 “Communication” Requirements

| **Requirement Title:** | Communication |
| --- | --- |
| **Sequence No:** | 001 |
| **Short Description:** | View Inbox and Retrieve Messages |
| **Description:** | When a user selects Messages it directs them to the Inbox where all previous and new messages are retrieved and displayed.  While in the Inbox, users can select among the following:   * Send Message * Mute Chat * Request Payment * Send Payment |
| **Pre-Conditions:** | * App must be loaded in order to process the request * User must have an existing account |
| **Post Conditions:** | * Inbox information is updated accordingly if a message is new or viewed |
| **Other Attributes:** | None |

## 

| **Requirement Title:** | Communication |
| --- | --- |
| **Sequence No:** | 002 |
| **Short Description:** | Send Message to another user |
| **Description:** | User selects Compose Message to reply or create a new message. A text box is displayed for user input.  After the user is satisfied with their input, the user selects Send Message to initiate or reply to a conversation. |
| **Pre-Conditions:** | * App must be loaded in order to process the request * User must have an existing account |
| **Post Conditions:** | * Inbox information is updated accordingly * User receives confirmation that their message was successfully sent |
| **Other Attributes:** | * Non-mechanic users initiate conversations |

## 

| **Requirement Title:** | Communication |
| --- | --- |
| **Sequence No:** | 003 |
| **Short Description:** | Request Payment via message |
| **Description:** | After a user and Mechanic conversates, the Mechanic is able to Request Payment.  The Mechanic requests payment via messaging and the user receives a message that directs them to Payment. |
| **Pre-Conditions:** | * App must be loaded in order to process the request * User must have an existing account |
| **Post Conditions:** | * Inbox information is updated accordingly |
| **Other Attributes:** | * Mechanic users can only request payment |

| **Requirement Title:** | Communication |
| --- | --- |
| **Sequence No:** | 004 |
| **Short Description:** | Send Payment via message |
| **Description:** | After the user opens and views the invoice from the Mechanic, the user is directed to Payment. |
| **Pre-Conditions:** | * App must be loaded in order to process the request * User must have an existing account |
| **Post Conditions:** | * Inbox information is updated accordingly * User is directed to Payment |
| **Other Attributes:** | None |

## 

## 4.4 “Mechanic Profile” Requirements

| **Requirement Title:** | Mechanic Profile |
| --- | --- |
| **Sequence No:** | 001 |
| **Short Description:** | Profile of Mechanics searching for customers |
| **Description:** | Mechanics can customize and view a profile of themselves. The default viewing of a profile is static. It displays:   * Mechanic name * Mechanic rating * Mechanic location (approximate - by city) * Mechanic profile picture * Mechanic price range * Mechanic services * Mechanic reviews   From the profile, the mechanic can choose to:   * Change Mechanic name * Change Mechanic location * Change user profile picture * Change mechanic price range * Change mechanic services * Change user password |
| **Pre-Conditions:** | * App must be loaded in order to make requests * Email must exist and be associated with an account * Password must be set |
| **Post Conditions:** | * Desired information was successfully updated and is displayed accordingly |
| **Other Attributes:** | None |

| **Requirement Title:** | Mechanic Profile |
| --- | --- |
| **Sequence No:** | 002 |
| **Short Description:** | Profile of Mechanics searching for customers |
| **Description:** | Mechanics can customize and view a profile of themselves. Mechanics must be authenticated before altering their profile. The Mechanic must enter:   * Password   After authenticating, Mechanics can alter:   * Mechanic name * Mechanic location (approximate - by city) * Mechanic profile picture * Mechanic price range * Mechanic services * Mechanic password   Mechanics cannot alter their ratings and reviews, which are permanently affixed to the profile unless removed by the app developers. |
| **Pre-Conditions:** | * App must be loaded in order to make requests * Email must exist and be associated with an account * Password must be set |
| **Post Conditions:** | * Desired information was successfully updated and is displayed accordingly |
| **Other Attributes:** | None |

## 

## 4.5 “Reporting” Requirements

| **Requirement Title:** | Reporting |
| --- | --- |
| **Sequence No:** | 001 |
| **Short Description:** | Report messages or users |
| **Description:** | The user may report a message or profile if either one violates the rules. A report may be filed by:   * An existing message in the inbox * Reporting a user or mechanic profile   A text box will appear first so the user may input why they are filing a report.   * There are options throughout the process to return home in case the reportee changes their mind   After the user submits the report they have an option to block the account. |
| **Pre-Conditions:** | * App must be loaded in order to process the request * Reportee must have an existing account |
| **Post Conditions:** | * Information is updated after the user submits a report * Reportee no longer sees reported account if they selected to block user |
| **Other Attributes:** | None |

## 

## 4.6 “Payment” Requirements

| **Requirement Title:** | Payment |
| --- | --- |
| **Sequence No:** | 001 |
| **Short Description:** | Selecting payment method |
| **Description:** | The user can initiate a payment method by selecting one of the options below:   * Credit/Debit Card payment * Cash payment   This allows the user to pay for the services they have already requested  The user can press:   * Cancel * Exit screen |
| **Pre-Conditions:** | * App must be loaded in order to process the request * User must have completed the service selection process * User must have a valid payment option available |
| **Post Conditions:** | * If user selects a credit card, they will be given an option to enter their card details * If user selects cash method, they will be prompted to provide cash |
| **Other Attributes:** | * The system should handle the process to switch payment method |

| **Requirement Title:** | Payment |
| --- | --- |
| **Sequence No:** | 002 |
| **Short Description:** | Credit Card payment method |
| **Description:** | The user selects the credit card payment method and proceeds to enter the required details:   * Card number * CVV * Name & Address   This allows the user to pay for the services they have already requested by credit card  The user can press:   * Cancel * Exit screen * Switch payment method |
| **Pre-Conditions:** | * App must be loaded in order to process the request * User must have selected the credit card payment option * User must have a valid credit card |
| **Post Conditions:** | * If a user enters a valid credit card, the payment is authorized and a confirmation email will be sent. * If payment fails, user is prompted to select again |
| **Other Attributes:** | * The system should handle the process to re-enter credit card information or switch/cancel |

| **Requirement Title:** | Payment |
| --- | --- |
| **Sequence No:** | 003 |
| **Short Description:** | Cash payment method |
| **Description:** | The user selects the cash payment method and proceeds to enter the required details:   * Amount of transaction * Name & Address   This allows the user to pay for the services they have already requested by cash  The user can press:   * Cancel * Exit screen * Switch payment method |
| **Pre-Conditions:** | * App must be loaded in order to process the request * User must have selected the cash payment option * User must have a sufficient cash |
| **Post Conditions:** | * The mechanic is notified for a pending cash payment * If payment fails, user is prompted to select again |
| **Other Attributes:** | * The system should handle the tracking of cash payments or switch/cancel |

## 4.7 “Appointment” Requirements

| **Requirement Title:** | Appointment |
| --- | --- |
| **Sequence No:** | 001 |
| **Short Description:** | Setting up appointment between user and mechanic |
| **Description:** | The user will be able to do the following of these:   * Selecting the mechanic * editing/ canceling the appointment * Add to calendar for setting up a reminder   The system should make sure that all the appointments are properly set by the users and notified to the mechanics so that both parties are able to manage their appointments effectively.  The user can press:   * Confirm appointment * Cancel appointment * Exit screen |
| **Pre-Conditions:** | * App must be loaded in order to process the request * User and Mechanic both must be registered and logged in the app * Both the availability of the user and mechanic must be matching |
| **Post Conditions:** | * Appointment status updated to the system * Added to the calender of both user and mechanic * Notification about the appointment is sent to both user and mechanic |
| **Other Attributes:** | * The system should ensure that all the changes made into the existing appointment should be alerted to the mechanic and user |

## 

## 4.8 “Search” Requirements

| **Requirement Title:** | Search |
| --- | --- |
| **Sequence No:** | 001 |
| **Short Description:** | Enables the user to search for services |
| **Description:** | The user will be able to search for services, for example, “oil changes.” The user must enter:   * Requested services in search box   The user can press:   * Search button * Cancel * Exit screen |
| **Pre-Conditions:** | * The app and data must be properly loaded so that the request can be processed * User is logged in |
| **Post Conditions:** | * Query made by system towards database |
| **Other Attributes:** | * None |

| **Requirement Title:** | Search |
| --- | --- |
| **Sequence No:** | 002 |
| **Short Description:** | System queries the database, results returned |
| **Description:** | The system will take the request from Search sequence number 001(above) and query the database, and then deliver the results to the user.  The user will be able to scroll and press:   * View mechanic profile * Schedule with mechanic * Return to search * Cancel * Exit screen |
| **Pre-Conditions:** | * The app and data must be properly loaded so that the request can be processed * User is logged in |
| **Post Conditions:** | * Results delivered by system |
| **Other Attributes:** | * System should be able to return results not directly like the search input to account for user mistyping/spelling something incorrectly. |

# 5. Software Processes and Infrastructure

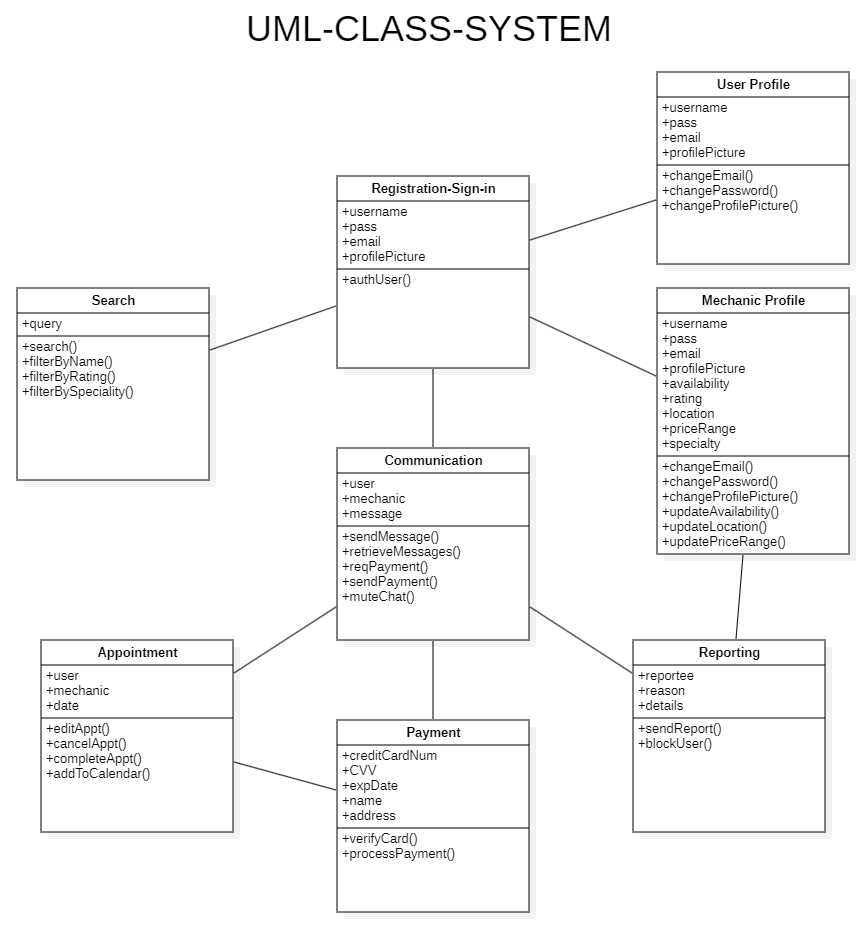
## 5.1 Hardware and Infrastructure

For building the application, we are using version 0.74.5 of the react-native framework with version 51.0.28 of expo that acts as a set of tools on top of react native. It also helps to run, debug, and test the app. These tools allow us to build native applications for both Android and iOS devices. With that, comes with all of the relevant dependencies. For the back end services of the application, such as user authentication or querying, we are using Google Firebase, which is implemented in our application through their API (version 10.14.1.)

## 

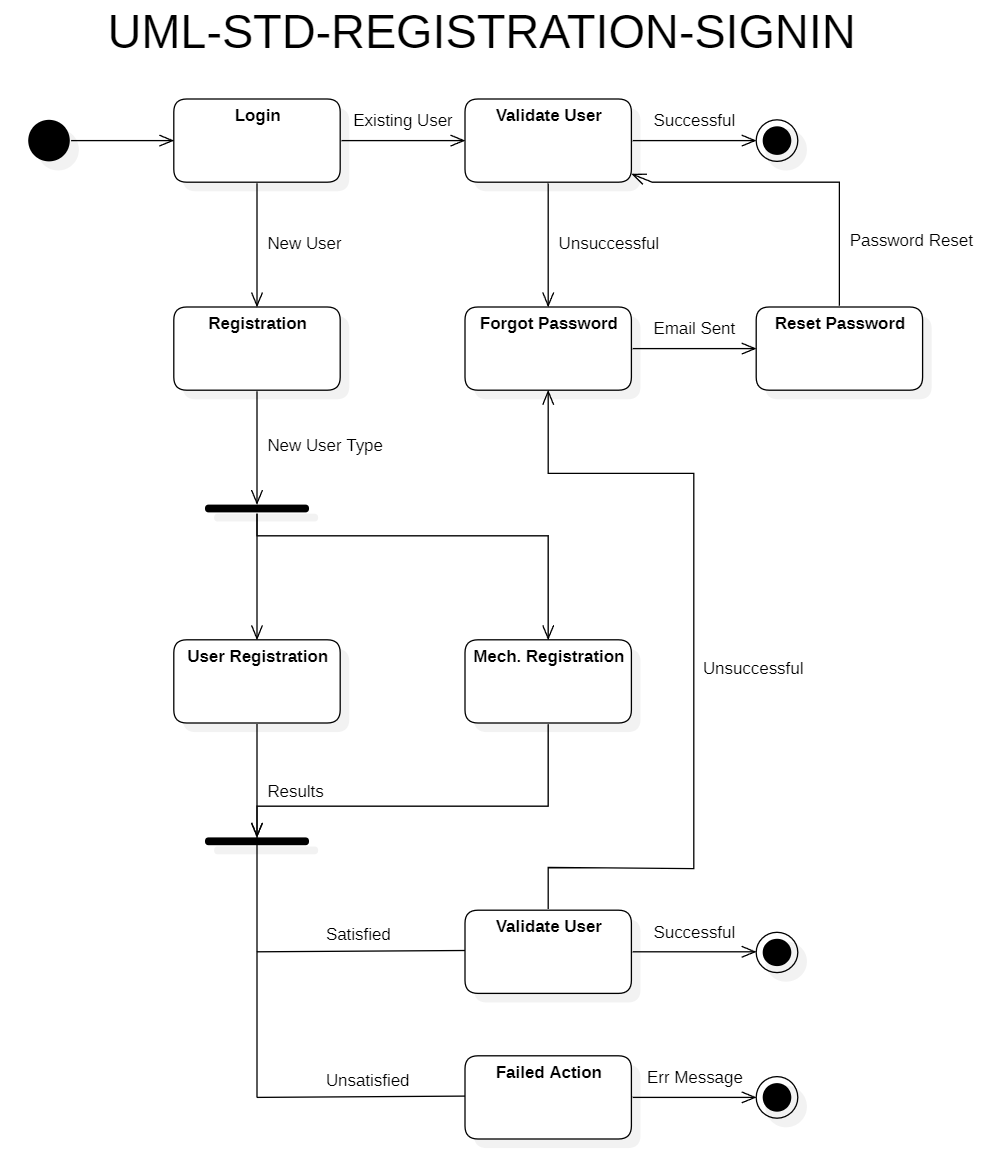
## 5.2 UML Diagrams

### 5.2.1 UML Diagram: “System”

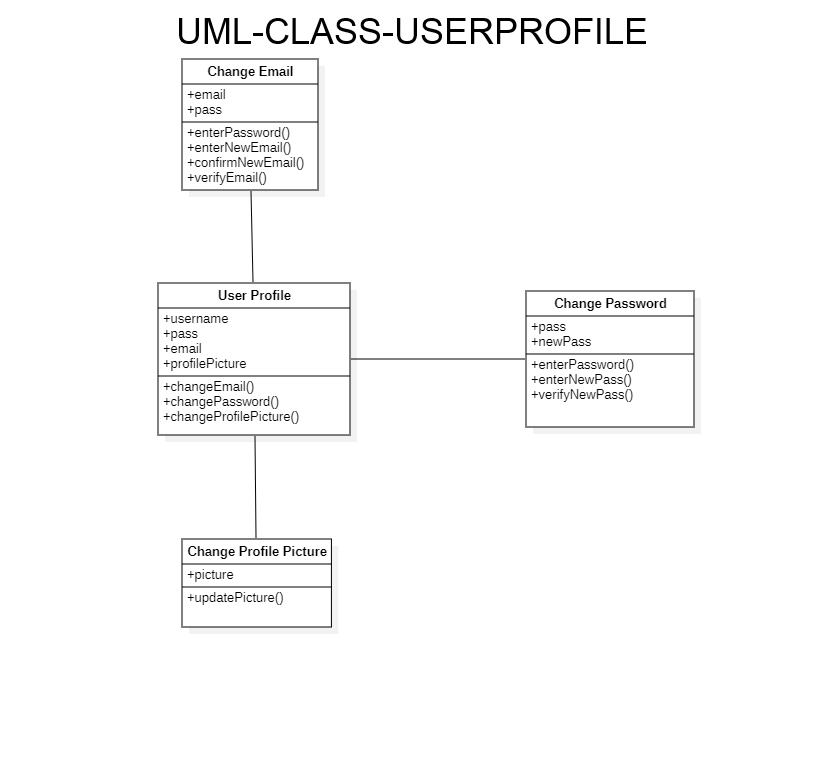


### 

### 5.2.2 UML Diagram: “Registration and Sign-in”

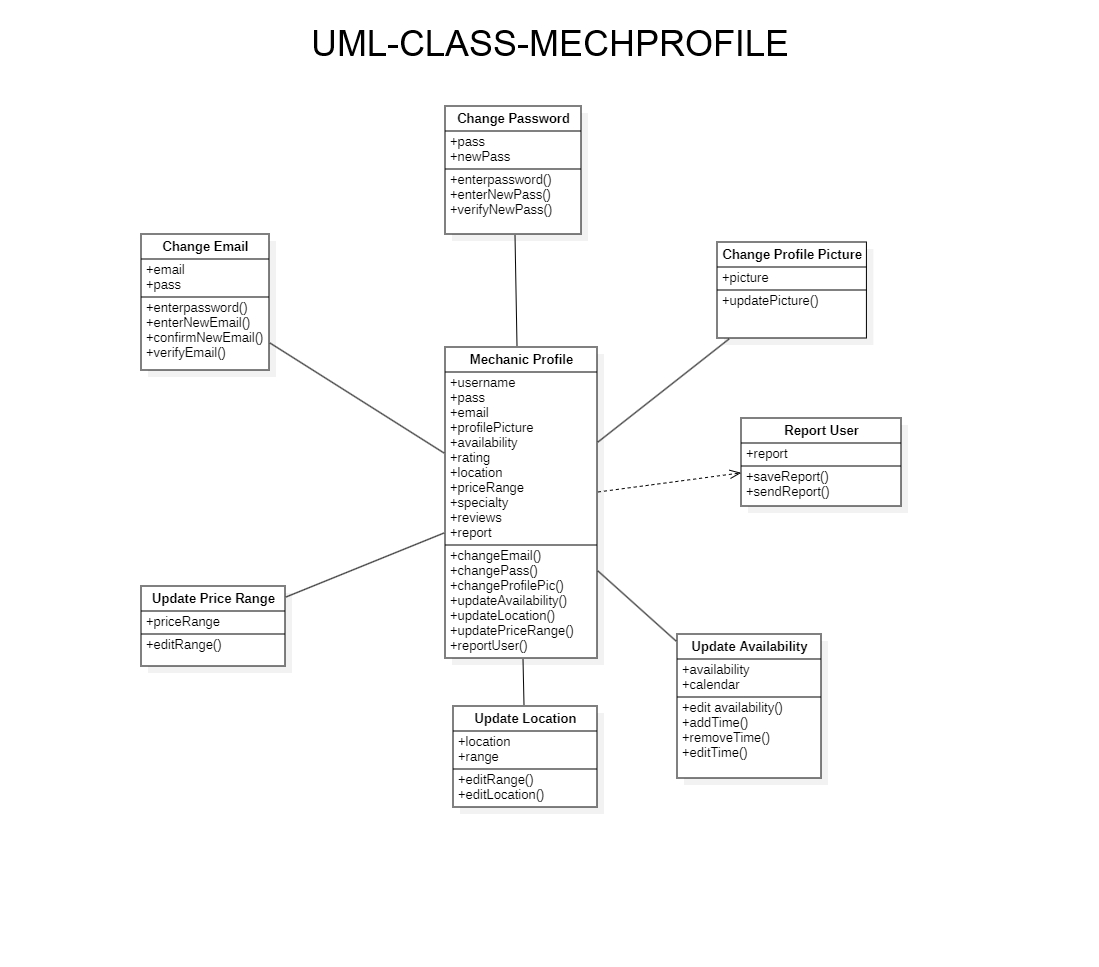


### 5.2.3 UML Diagram: “User Profile”

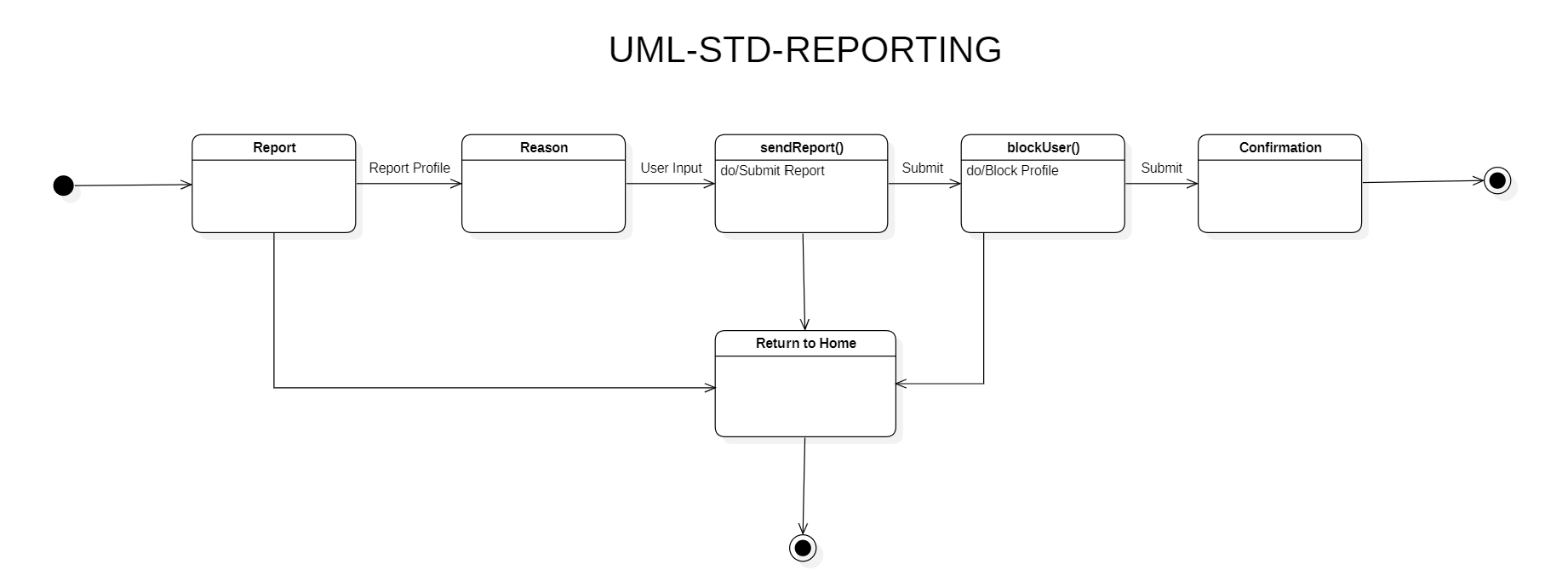


### 5.2.4 UML Diagram: “Communication”

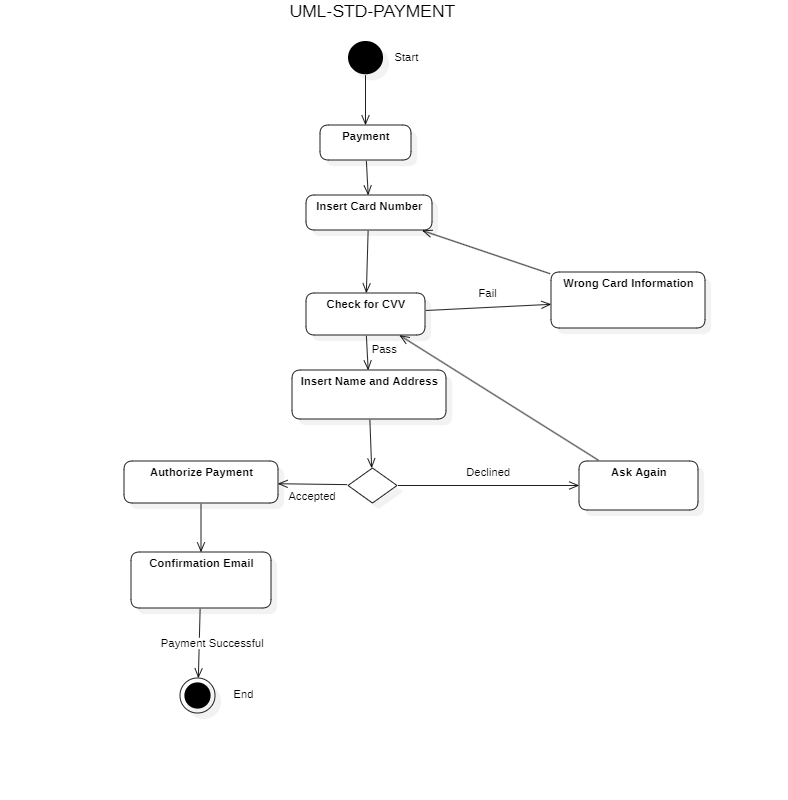
### 5.2.5 UML Diagram: “Mechanic Profile”



### 5.2.6 UML Diagram: “Reporting”



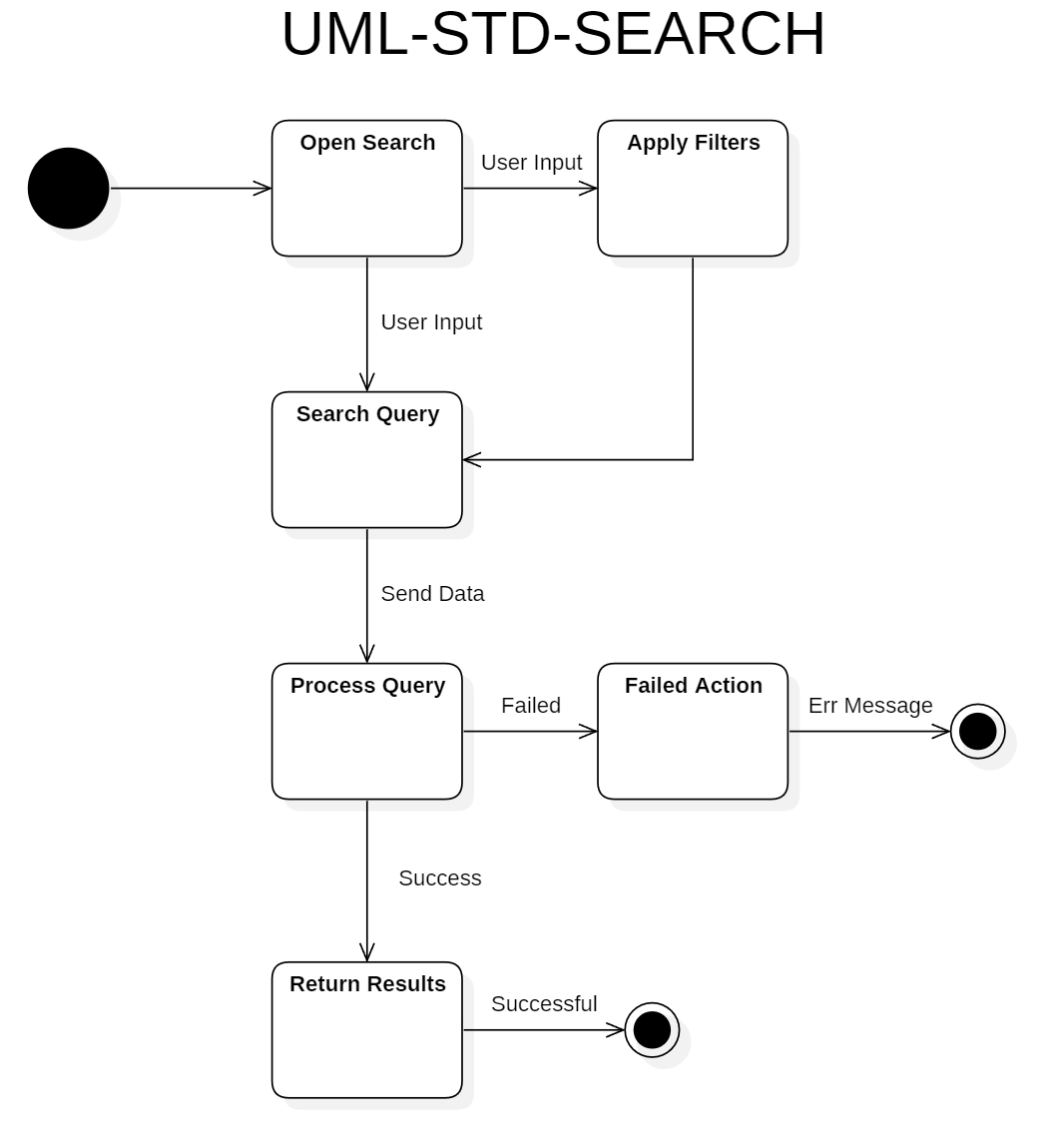
### 5.2.7 UML Diagram: “Payment”



### 5.2.8 UML Diagram: “Appointment”

## 

### 5.2.9 UML Diagram: “Search”



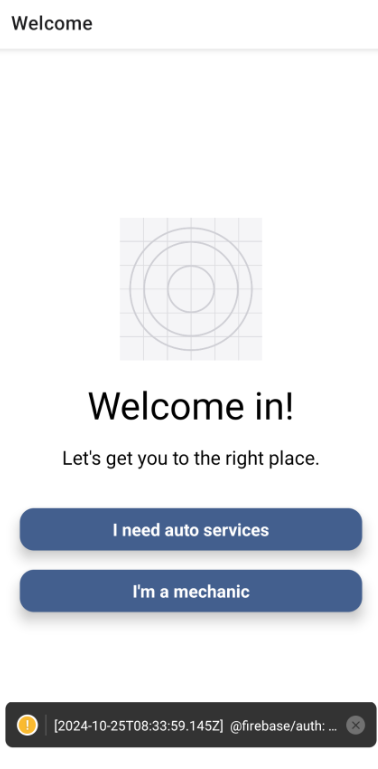
## 5.3 Conceptual Data Model

We will be using Firebase and its Firestore functionality for all of our database needs. We will be able to access the databases through Firebase’s API, and it has methods for maintaining security, performing queries and returning results, and asynchronously adding entries into the database, among other features.

## 

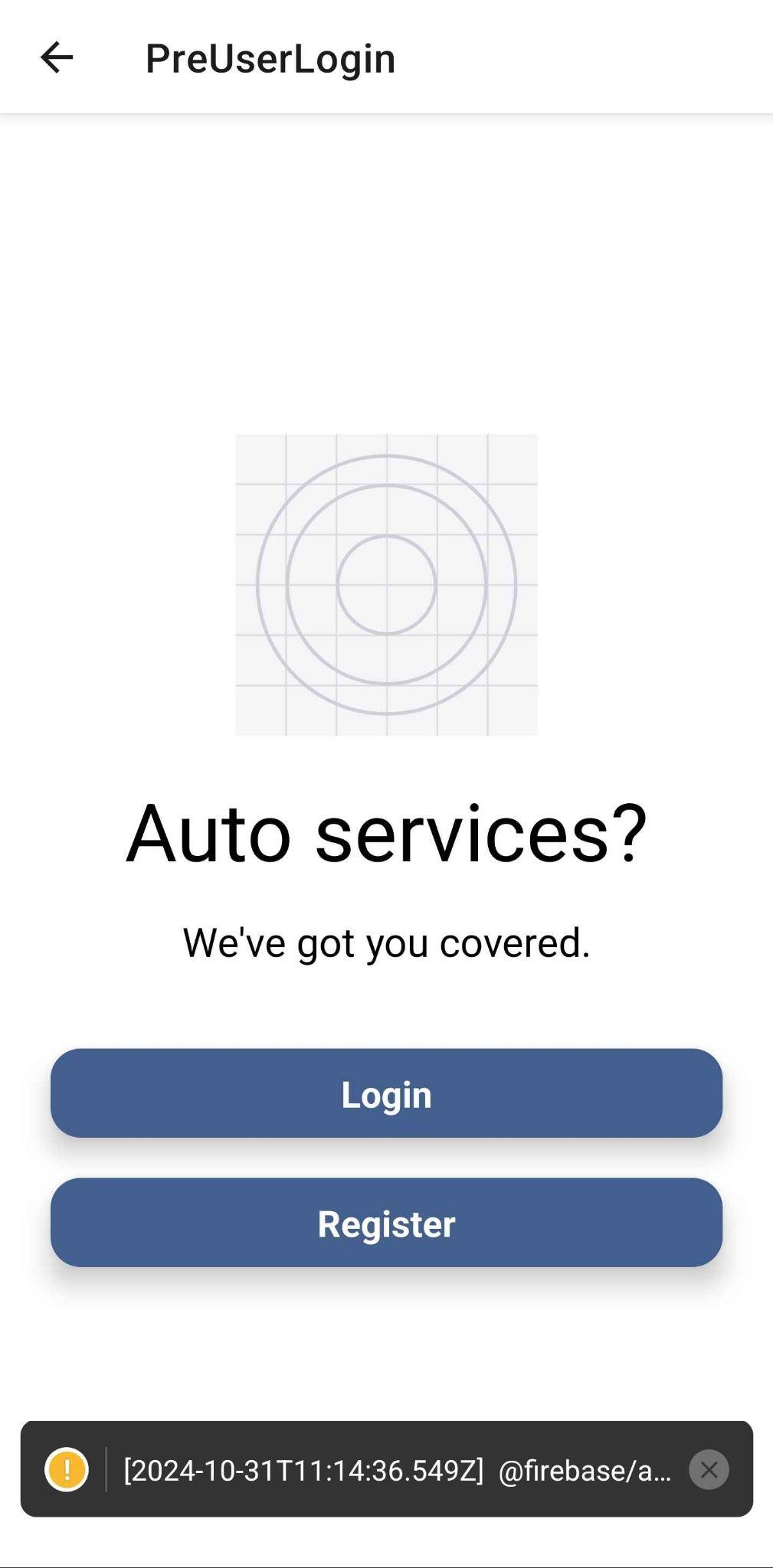
## 5.4 Screen Shots

### 5.4.1 Screen “Start Screen”



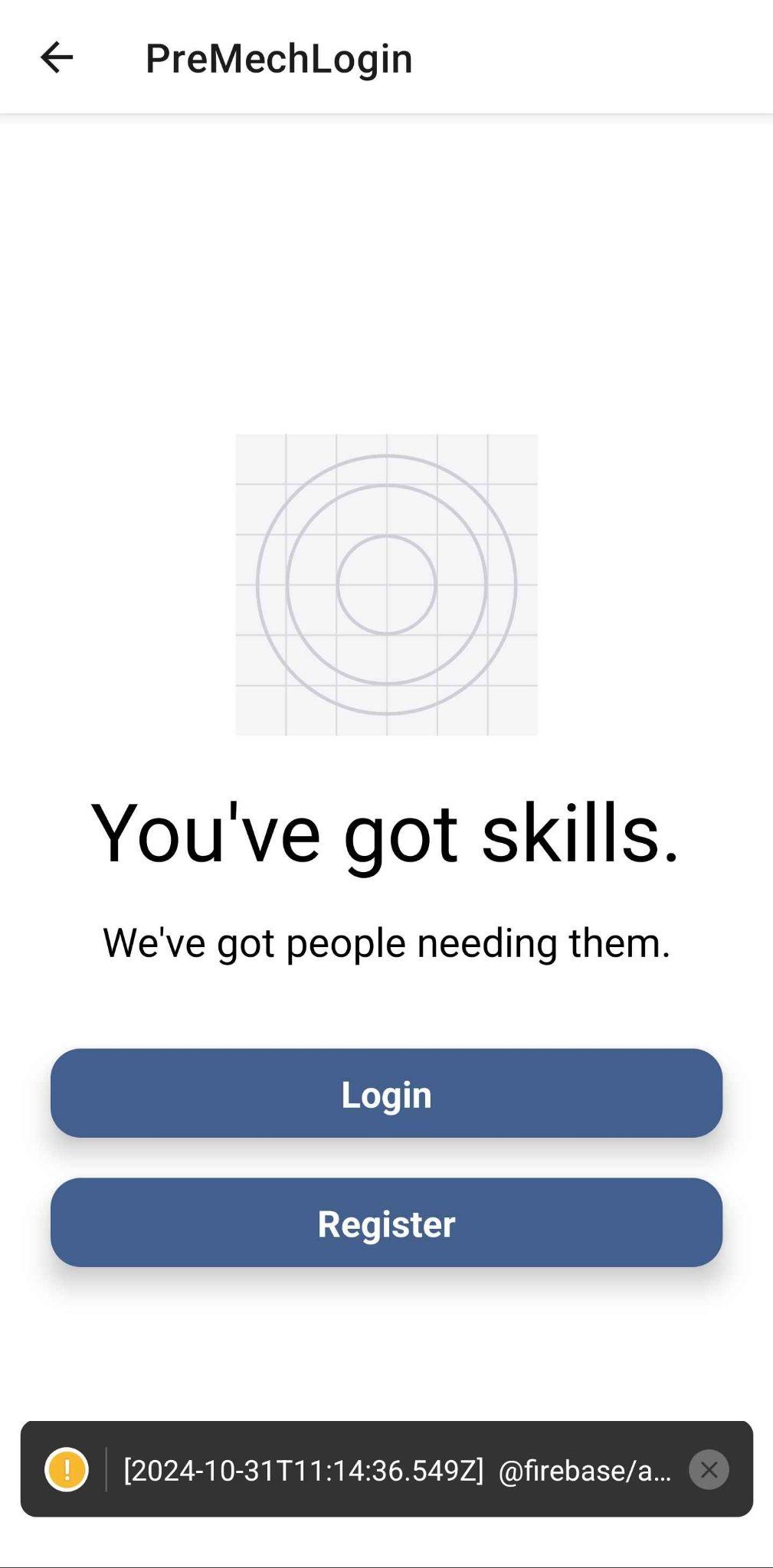
### 

### 5.4.2 Screen “User Registration/Login Screen”



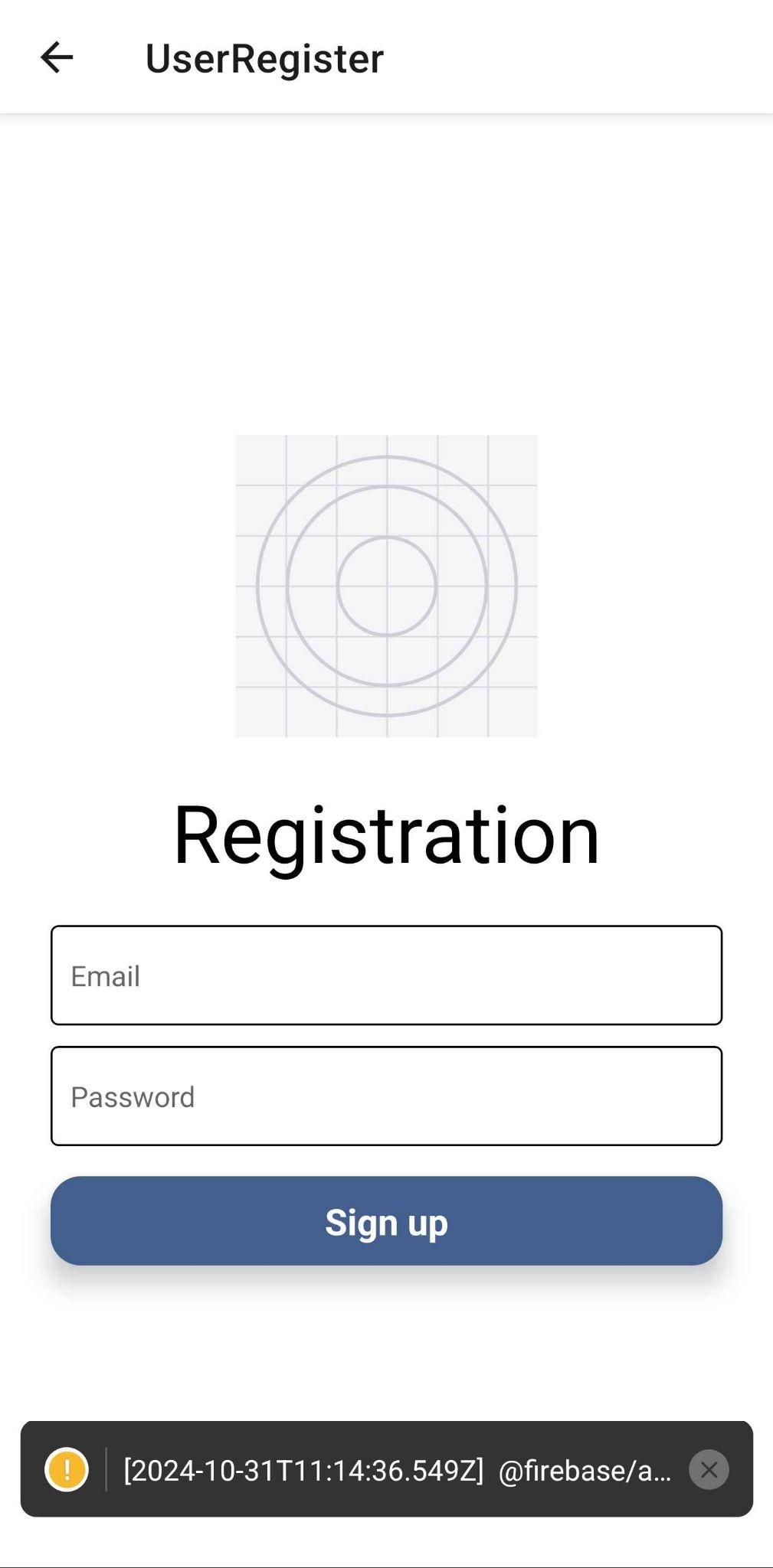
### 

### 5.4.3 Screen “Mechanic Registration/Login Screen”



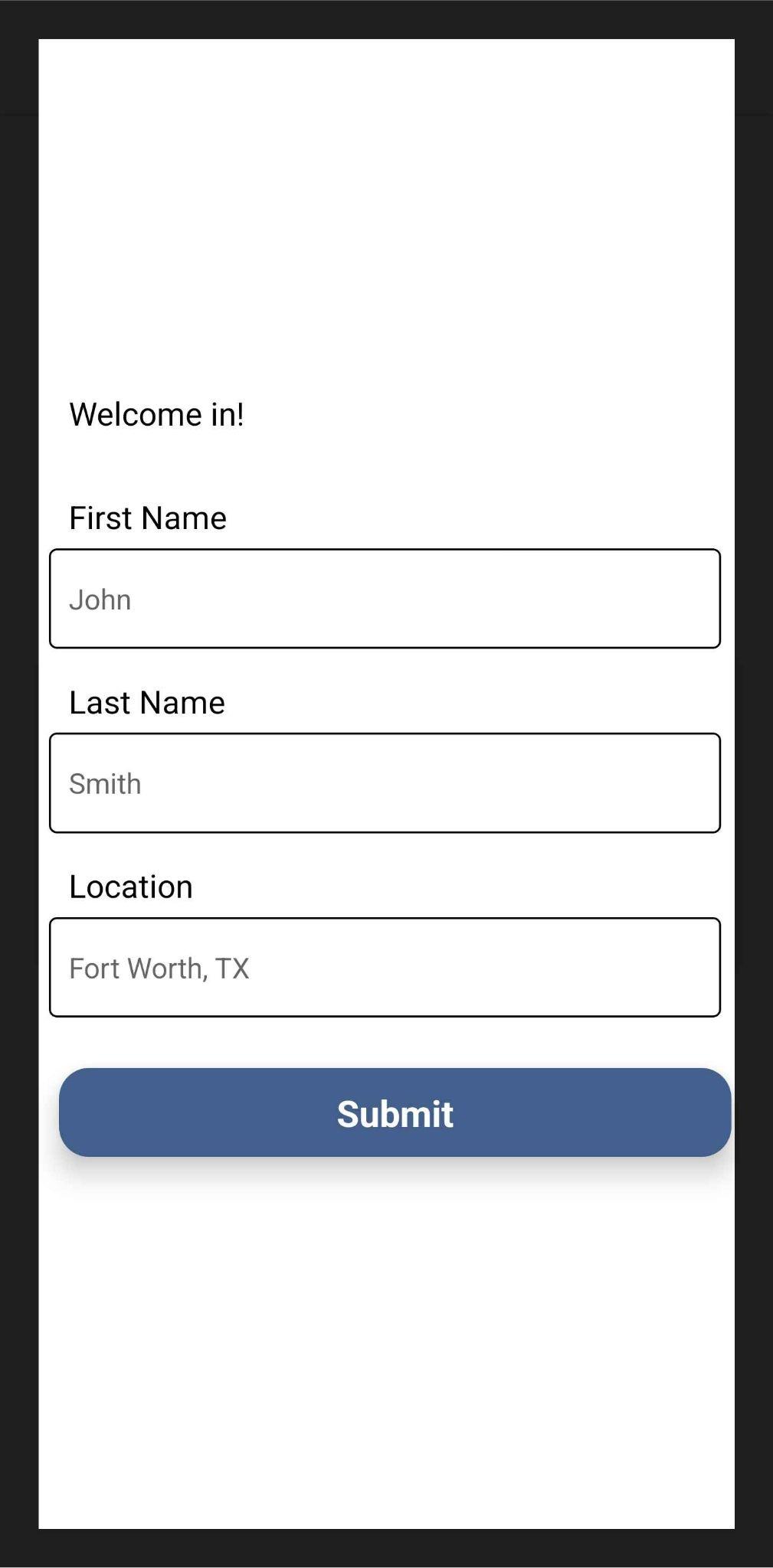
### 

### 5.4.4 Screen “User Registration Screen”



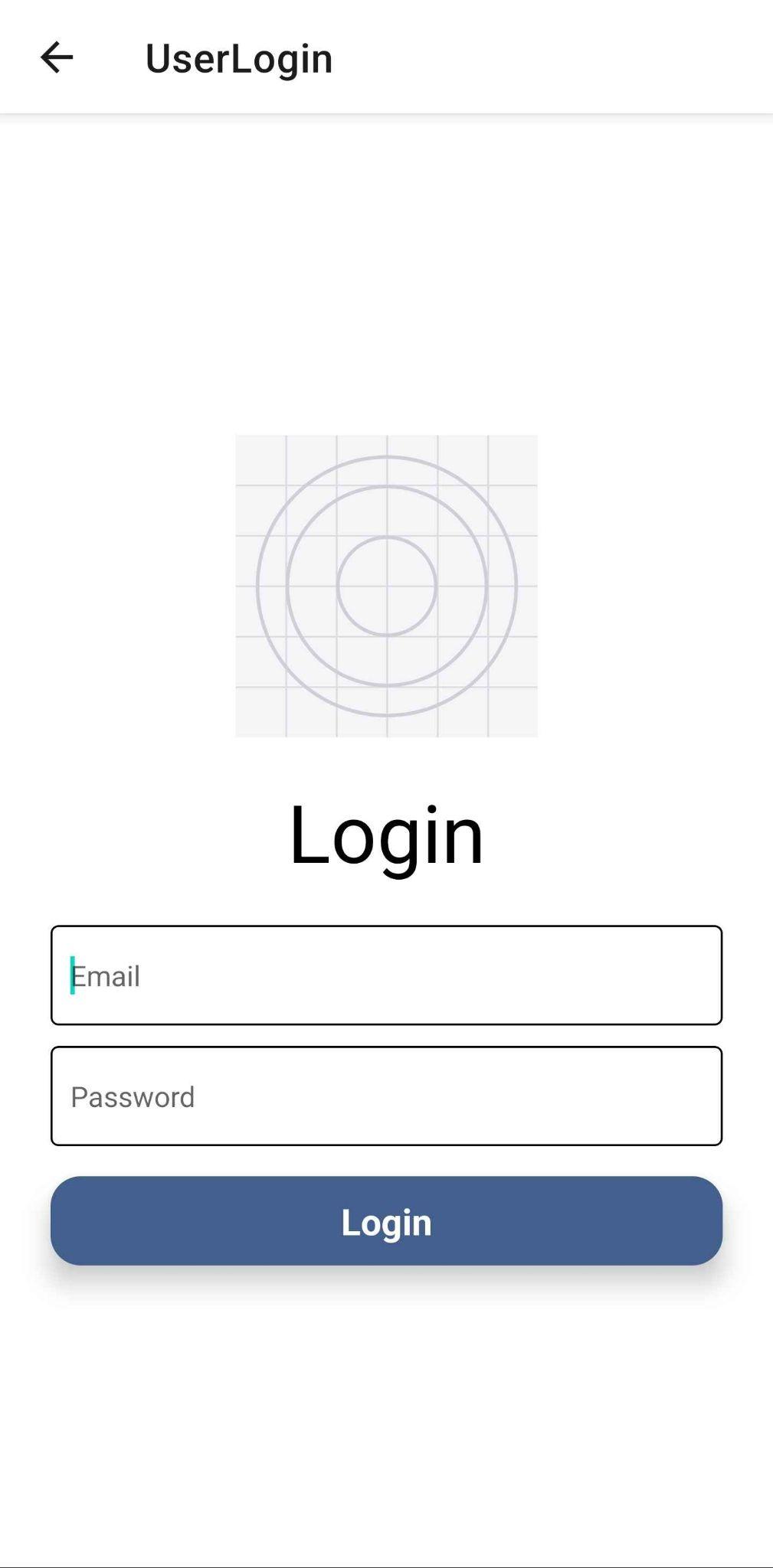
### 

### 5.4.5 Screen “User Registration Info Collection Screen”



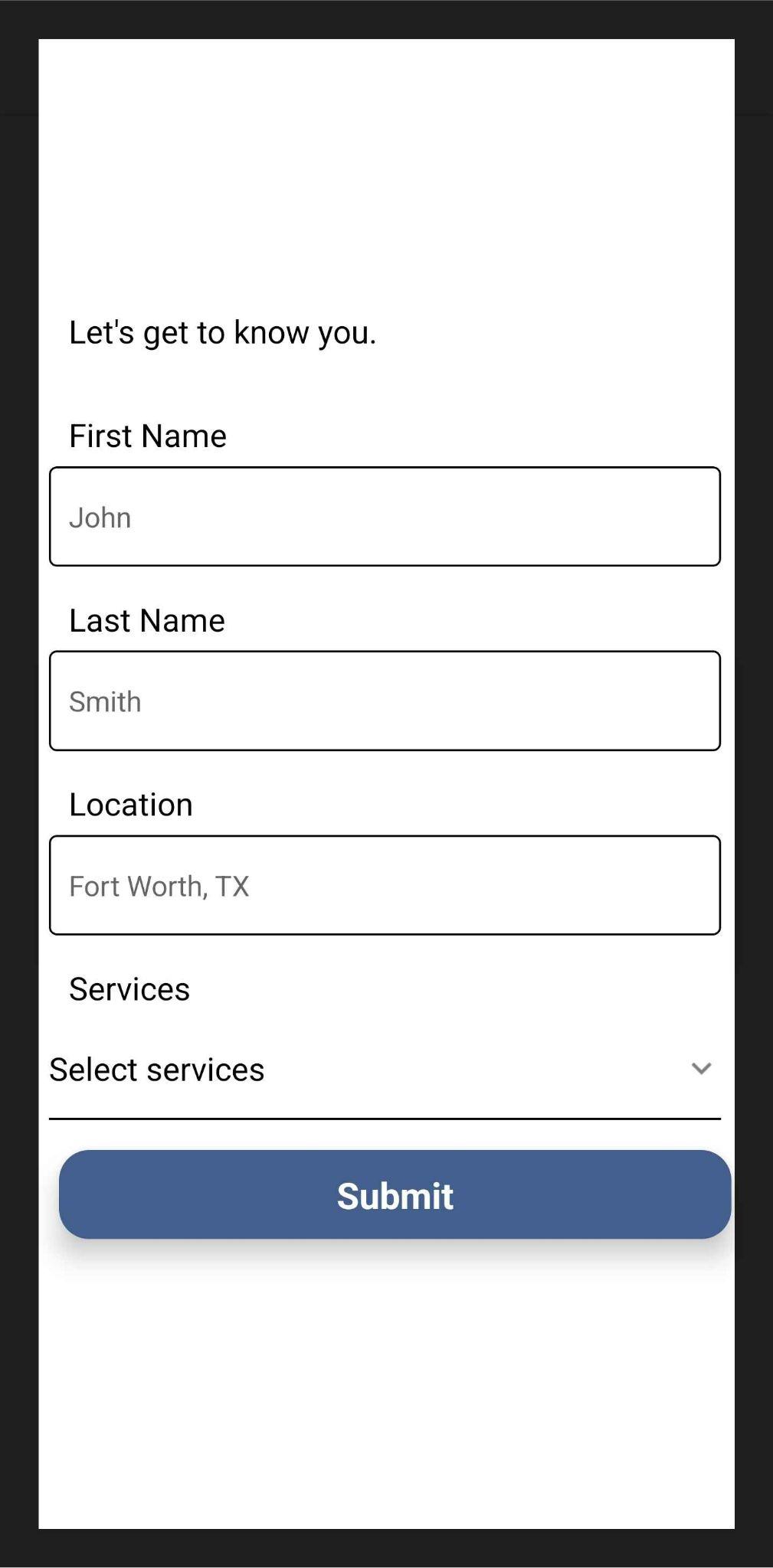
### 

### 5.4.6 Screen “User Login Screen”



### 

### 5.4.7 Screen “Mechanic Registration Info Collection Screen”



## 5.5 Test Plan

A test plan will be provided at a later stage of the project.

# 6. Assumptions and Constraints

## 6.1 ASSUMPTIONS

The following is a list of assumptions:

* Assume all users over the age of 18
* Ignore all financial and tax reporting details
* Code is not written with high-standard security protocols in mind
* Scalability is not a necessary consideration
* Code is not written with high-standard optimization in mind

## 6.2 CONSTRAINTS

The following is a list of constraints:

* The team lacks Android development skills
* Schedule very aggressive
* Limited experience with UI/UX
* Relying on third party software
* Development team is new to working together

## 6.3 Out-of-Scope Material

The following is a list of “out of scope” material:

* Post Project maintenance is not covered
* Proper security feature implementation
* Cross-platform compatibility
* Design of proper and thorough accessibility features
* External system integration

# 7. Delivery and Schedule

{List all tasks/milestones from the start of the project to the end with specific dates for both Anticipated Start & End Dates, Remove this guideline}

| Task/Milestone Description | Anticipated Start Date | Anticipated End Date | Status  {Complete, In Progress, To Be Completed-TBC} | Comments  (Deliverable plus the team member working on the listed item) |
| --- | --- | --- | --- | --- |
| Prepare UML diagrams | 9/10/2024 | 9/26/2024 | Complete | Deliverable 🡺 UML document (TEAM) |
| Create an official SRA document | 9/27/2024 | 10/31/2024 | In Progress | Deliverable will be the SRA document. All stakeholders agree on the content of the SRA by signing in section 8. |
| Registration and Sign-in | 10/11/24 | 10/17/24 | Complete | Patrick Williams |
| User profile | 10/31/2024 | 11/18/24 | TBC | Margaret Roche |
| Communication | 10/31/2024 | 11/18/24 | TBC | Jonathan Hor |
| Mechanic Profile | 10/31/2024 | 11/18/24 | TBC | Margaret Roche |
| Reporting | 10/31/2024 | 11/18/24 | TBC | Jonathan Hor |
| Payment | 10/20/24 | 11/15/24 | TBC | Sujana Kabir |
| Appointment | 10/25/24 | 11/18/24 | TBC | Sujana Kabir |
| Search | 10/25/24 | 11/10/24 | TBC | Patrick Williams |
| Test Plan Delivery | 11/1/2024 | 11/14/2024 | TBC | The Team Deliverable will be the Test plan document. |
| External Documentation (i.e., User Manual) | 11/26/2024 | 12/1/2024 | TBC | The team Deliverable will be a User Manual. |
| Final Milestone: Project Delivery and Team Presentation | 11/26/2024 | 12/3/2024 | TBC | The Team Deliverable will be the final project binder plus the product demo |

# 8. Stakeholder Approval Form

| Stakeholder Name | Stakeholder Role | Stakeholder Comments | Stakeholder Approval Signature and Date |
| --- | --- | --- | --- |
| Bahram Khalili | Client |  |  |
| Tuan Dang | Client Project Manager |  |  |
| Margaret Roche | Developer |  | Margaret Roche  10/30/2024 |
| Patrick Williams | Developer |  | Patrick Williams  10/30/2024 |
| Sujana Kabir | Developer |  | Sujana Kabir  10/30/2024 |
| Jonathan Hor | Developer |  | Jonathan Hor  10/30/2024 |

|  |  | | | | |
| --- | --- | --- | --- | --- | --- |
|  | |  | | | |
| Project: | | Mobile Mech | | | |
| Team No.: | | Team#4 | | | |
| Class: | | CSE 3310; Fall 2024 | | | |
| Module: | | Test Plan | | | |
| Deliverable: | | Test Plan Document | | | |
| **Version:** | | | **[1.0]** | **Date:** | **[11/14/2024]** |

Team members are:

Sujana Kabir

Margaret Roche

Patrick Williams

Jonathan Hor

**Revision History**

| ***Version number*** | ***Date*** | ***Originator*** | ***Reason for change*** | ***High-level description of changes*** |
| --- | --- | --- | --- | --- |
| 1.0 | 11/14/2024 | Team 4 | Initial draft |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

TABLE OF CONTENTS

[**1. INTRODUCTION AND PLAN OF APPROACH 3**](#_heading=h.2et92p0)

[**2. TEST CASES: “REGISTRATION AND LOGIN” 4**](#_heading=h.tyjcwt)

[**3. TEST CASES: “USER PROFILE” 6**](#_heading=h.ajnb7f6srkai)

[**4. TEST CASES: “COMMUNICATION” 8**](#_heading=h.jpd286r7w0w)

[**5. TEST CASES: “MECHANIC PROFILE” 9**](#_heading=h.dqfpumqbpxl5)

[**6. TEST CASES: “REPORTING” 11**](#_heading=h.2s8eyo1)

[**7. TEST CASES: “PAYMENT” 12**](#_heading=h.17dp8vu)

[**8. TEST CASES: “APPOINTMENT” 14**](https://docs.google.com/document/d/1cLkJi2HIzLtdTlFBoMvuakDxoZNoqiG0/edit#heading=h.3rdcrjn)

**9**[**. TEST CASES: “SEARCH” 15**](https://docs.google.com/document/d/1cLkJi2HIzLtdTlFBoMvuakDxoZNoqiG0/edit#heading=h.3rdcrjn)

# 1. Introduction and Plan of Approach

The Mobile Mech App is an innovative Android application designed to connect users with professional mechanics, offering a convenient and efficient way to access automotive repair and maintenance services. Inspired by healthcare apps like ZocDoc or Teledoc that provide "on-demand" services, the Mobile Mech App aims to bring similar convenience to the automotive repair industry.

The android project will consist of 8 different components: Registration and Sign in, User Profile, Communication, Mechanic Profile, Reporting, Payment, Appointment, Search. When the user logs in to the application, they will be prompted to register or sign in. If they are an existing user, the user will simply sign in using their email and password. Otherwise the user will have to register. There will be both user and mechanic profiles so that both parties can have a bit of an overview. There will be ways of communication for the user with the mechanic. In the app, there will be an option to transact money from the user to the mechanic. Similar to ride apps, there will be reviews available of the mechanic so that the user can choose whom to contact best.

# 2. Test Cases: “[REGISTRATION AND LOGIN](#_heading=h.tyjcwt)”

**Project Name:** Mobile Mech

**Test Case Name:** Registration and Login

**Test Case Id**: CSE3310/Fall 2024/Team4/Registration-and-Login

| **Test Case No.** | **Test Case Description** | **Expected results** | **Outcome**  **Pass, Fail, Other (comments)** |
| --- | --- | --- | --- |
| TC1 | Enter the login screen and press “Register new user” | The system should display and collect the information below and establish a new user ID and Password:   * First & Last Name * Email address * User ID (at least 6 characters, may not include any wild characters * Password (at least 8 charters, must include a number and wild character) * Answer to security question “In what city were you born?”, i.e., for password reset |  |
| TC2 | Enter the login screen and try to register the same ID once again | The system should provide an error message that “user already exists” |  |
| TC3 | Tab into the User and password fields and enter a valid user ID/password (a test ID should be created and provided to testers) | System should let you in |  |
| TC4 | Tab into the User and password fields and enter an invalid user ID/password (tester: please make up any user/password) | System should not accept and prevent you from entry |  |
| TC5 | Enter a valid user ID (use the valid ID from TC1) and press “Forgot Password” | The system should ask for your e-mail address and ask for the answer to a security question, if the requested information is provided, a temporary password should be created, and the system should encourage the user to change this randomly generated password |  |

# 3. Test Cases: “User Profile”

**Project Name:** Mobile Mech

**Test Case Name: User Profile**

**Test Case Id**: CSE3310/Fall2024/Team4/User Profile

| **Test Case No.** | **Test Case Description** | **Expected results** | **Outcome**  **Pass, Fail, Other (comments)** |
| --- | --- | --- | --- |
| TC1 | Updating user profile picture from the default profile picture. Image should fit the allowed size (notify user picture is too big/small if not within allowed sizes). | Users should navigate to the user profile page and select to edit the profile. From there, they can trade the default profile picture for a profile picture they have uploaded from their device. Image should have the same rotation as the uploaded photo. Users should receive an error if the image is the wrong size. |  |
| TC2 | Updating user profile picture from the current profile picture. Image should fit the allowed size (notify the user the picture is too big/small if not within allowed sizes). | Users should navigate to the user profile page and select to edit the profile. From there, they can trade the current profile picture for a profile picture they have uploaded from their device. Image should have the same rotation as the uploaded photo. Users should receive an error if the image is the wrong size. |  |
| TC3 | Alter the user name from the name entered when creating their account. The user should be able to open the profile page, choose to edit their profile, and enter and save a new user name. | Saving the user name should cause the page to refresh and display the new name. |  |
| TC4 | Updating user location. The app detects user location by default, but users may choose to edit their profile and change their current location for another one. | Update what user location is displayed. |  |
| TC5 | Update password. Like all other user specific details, users can navigate to the profile page and update their profile. | Update the used password used to access the app. |  |

# 4. Test Cases: “Communication”

**Project Name:** Mobile Mech

**Test Case Name: Communication**

**Test Case Id**: CSE3310/Fall2024/Team4/Communication

| **Test Case No.** | **Test Case Description** | **Expected results** | **Outcome**  **Pass, Fail, Other (comments)** |
| --- | --- | --- | --- |
| TC1 | User selects Messages tab | The user will be prompted by three options: “View Messages”, “Compose Message”, “Send Payment”, or “Request Payment” if the user is a Mechanic |  |
| TC2 | View Messages | If selected the user should be able to view all and select messages in their inbox |  |
| TC3 | Compose Message | Users will be prompted with a text box to enter their text and enter the mechanic’s name. After their input is complete, the user clicks “Send”. |  |
| TC4 | Confirmation Message Sent | After user clicks, “Send”, user will receive a confirmation screen that message was successfully sent |  |
| TC5 | Send Payment | After a Mechanic user has requested payment to a user, the user will be able to select “Send Payment” and will be redirected to “Payment” to complete the transaction |  |
| TC6 | Request Payment | After a Mechanic user engages with a potential customer, they are able to select Request Payment to send an invoice to a user |  |

# 5. Test Cases: “Mechanic Profile”

**Project Name:** Mobile Mech

**Test Case Name:**  Mechanic Profile

**Test Case Id**: CSE3310/Fall 2024/Team4/Mechanic-Profile

| **Test Case No.** | **Test Case Description** | **Expected results** | **Outcome**  **Pass, Fail, Other (comments)** |
| --- | --- | --- | --- |
| TC1 | Updating mechanic profile picture from the default profile picture. Image should fit the allowed size (notify user picture is too big/small if not within allowed sizes). | Mechanics should navigate to the mechanic profile page and select to edit the profile. From there, they can trade the default profile picture for a profile picture they have uploaded from their device. Image should have the same rotation as the uploaded photo. Mechanics should receive an error if the image is the wrong size. |  |
| TC2 | Updating mechanic profile picture from the current profile picture. Image should fit the allowed size (notify the user the picture is too big/small if not within allowed sizes). | Mechanics should navigate to the mechanic profile page and select to edit the profile. From there, they can trade the current profile picture for a profile picture they have uploaded from their device. Image should have the same rotation as the uploaded photo. Mechanicss should receive an error if the image is the wrong size. |  |
| TC3 | Alter the mechanic name from the name entered when creating their account. The mechanic should be able to open the profile page, choose to edit their profile, and enter and save a new user name. | Saving the mechanic name should cause the page to refresh and display the new name. Reviews, ratings, and other mechanic statistics should remain unchanged. |  |
| TC4 | Updating mechanic location. The app detects mechanic location by default, but mechanics may choose to edit their profile and change their current location for another one. | Update what mechanic location is displayed. |  |
| TC5 | Update password. Like all other mechanic specific details, users can navigate to the profile page and update their profile password. | Update the mechanic password used to access the app. |  |
| TC6 | Each mechanic can set, if they choose to, a price range for their services. There is no price range displayed by default.They can both set and alter the price range by navigating to the profile page and choosing to edit it. | If mechanics edit their price range, said price range should update and display in the app. Range should not display outside of the permitted minimum and maximum values. |  |
| TC7 | Each mechanic has a section to list which services in the auto industry their traveling service provides. They can update this list any time by navigating to the profile page and choosing to edit it. | Profile page should update services based on what was removed and added. |  |
| TC8 | Review added to mechanic profile. Mechanics cannot alter the review left by user profiles. However, the profile page should automatically update and display any reviews and ratings added by user profiles in the app. | Mechanic profile overall rating and reviews update based on reviews added by users. |  |

# 6. Test Cases: “Reporting”

**Project Name:** Mobile Mech

**Test Case Name: Reporting**

**Test Case Id**: CSE3310/Fall 2024/Team4/Reporting

## 

| **Test Case No.** | **Test Case Description** | **Expected results** | **Outcome**  **Pass, Fail, Other (comments)** |
| --- | --- | --- | --- |
| TC1 | User should be able to report message(s) that break the ToS | The user should be able to select messages in a conversation and be able to file a report |  |
| TC2 | Display Text Box prompt | After user selects report, the user should be prompted with a text box to enter their reason for reporting |  |
| TC3 | Return Home | Throughout the entire process, the user will be able to click a button to return home if they decide to change their mind |  |
| TC4 | Submit Report | After user finishes entering their text, they should be able to click a button that says submit report |  |
| TC5 | Confirmation | After report is submitted, user should get a confirmation screen and user returns back to messages |  |

# 7. Test Cases: “Payment”

**Project Name:** Mobile Mech

**Test Case Name: Payment**

**Test Case Id**: CSE3310/Fall 2024/Team4/Payment

## 

| **Test Case No.** | **Test Case Description** | **Expected results** | **Outcome**  **Pass, Fail, Other (comments)** |
| --- | --- | --- | --- |
| TC1 | Enter the payment page and be prompted to choose between two payment methods: Credit card or cash on payment | There will be given the choices of choosing the payment method. Based on that, the screen should change and go to the desired payment method page. There may be an option to go back and change your selection. |  |
| TC2 | For credit card, it prompts the user to accept necessary information | The user is prompted to add personal information and card details like-   * Enter the name of the card holder * Enter address and zip code * Enter CVV of the card |  |
| TC3 | For cash on payment, it prompts the user to accept necessary information | The user is prompted to enter the name of the payer and the amount to be paid. A notification of pending payment will be sent to both user and mechanics and will be confirmed by the mechanic when the payment is done in person. |  |
| TC4 | User payment successfully authorized | The app should display ‘payment accepted’ and send a notification to both user and mechanic that the payment was processed in the app. An online receipt will be sent for successful payment to both parties as a confirmation message. |  |
| TC5 | User payment failed | The app should display a ‘payment failed’ message. Then the user will be prompted to-   * change payment method * enter new payment information * correct the existing information where it is marked in red. For eg. CVV/ Name of card holder |  |

# 8. Test Cases: “Appointment”

**Project Name:** Mobile Mech

**Test Case Name: Appointment**

**Test Case Id**: CSE3310/Fall 2024/Team4/Appointment

## 

| **Test Case No.** | **Test Case Description** | **Expected results** | **Outcome**  **Pass, Fail, Other (comments)** |
| --- | --- | --- | --- |
| TC1 | Enter the mechanic profile and press “Make an appointment” | The system should display and collect the information below to confirm a new appointment:   * First & Last Name * Email address * Meeting Location * Appointment Date & Time |  |
| TC2 | User tries to edit an existing appointment | The system should display and collect the information below to edit an existing appointment:   * First & Last Name * Email address * New Meeting Location * New Date & Time * Reason for change |  |
| TC3 | User wants to cancel the appointment | The user has to provide a reason in the text box provided. The appointment will be canceled and a confirmation email will be sent to both user and mechanic. |  |
| TC4 | Confirm the appointment | The user and mechanic will both receive a confirmation email. |  |
| TC5 | Add to calendar | The appointment is added to calendar and a reminder is set. |  |

# 9. Test Cases: “Search”

**Project Name:** Mobile Mech

**Test Case Name: Search**

**Test Case Id**: CSE3310/Fall 2024/Team4/Search

## 

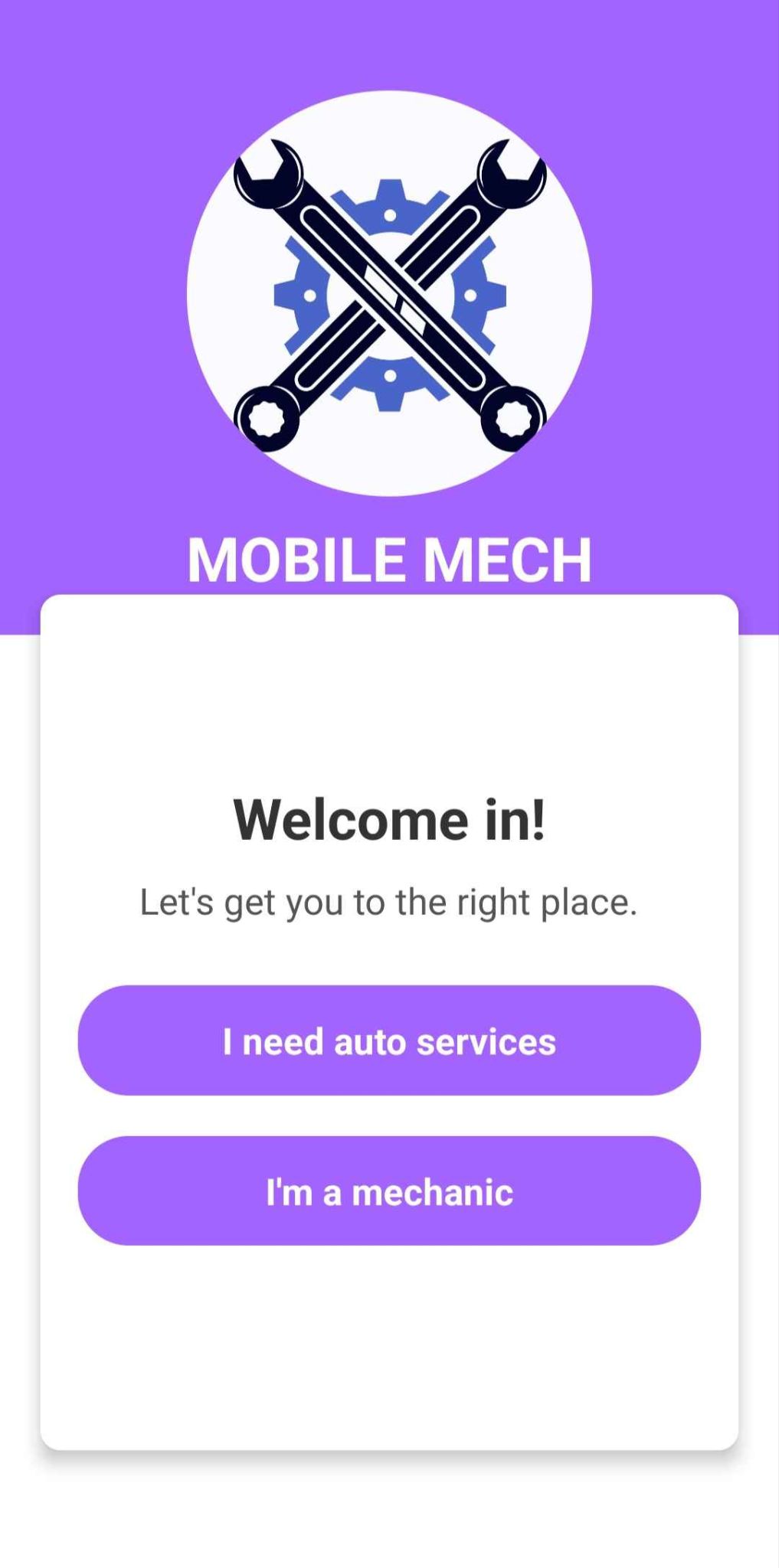
| **Test Case No.** | **Test Case Description** | **Expected results** | **Outcome**  **Pass, Fail, Other (comments)** |
| --- | --- | --- | --- |
| TC1 | Enables the user to search for services. Test case is to search partial words, like if enter is hit too early by accident. | For snippets of words such as “oi” for “oil change, “brake” for “brake check”, or “wipe” for “wiper fluid” the app should search for the best match from listed services. Mechanics who offer the corresponding best match service will be displayed from best rated to worst rated by default. |  |
| TC2 | Search with complete phrases such as “oil change”, “brake check”, and “wiper fluid”. | Mechanics who offer the corresponding best match service will be displayed from best rated to worst rated by default. |  |
| TC3 | Test extremely long search entries, such as “oil change and brake check for lexus RX350” | Search should accept any string of words, up to a certain length, parse into tokens via the white spaces, and create best matches based on each token. In this case, search would provide matches for tokens oil, change, brake, check. Car makes and models are not part of services. |  |
| TC4 | Try to enter a string of words over the character limit | The search bar will not accept a string of characters which extends past the character limit. |  |
| TC5 | Enter one long string of words, with no white spaces to parse, such as “brakecheckandoilchange | With no spaces to indicate tokens, this will be treated as one long token, and there are likely to be no matches. If no matches is the case, app should display something like “No matches found, enter a new search” |  |

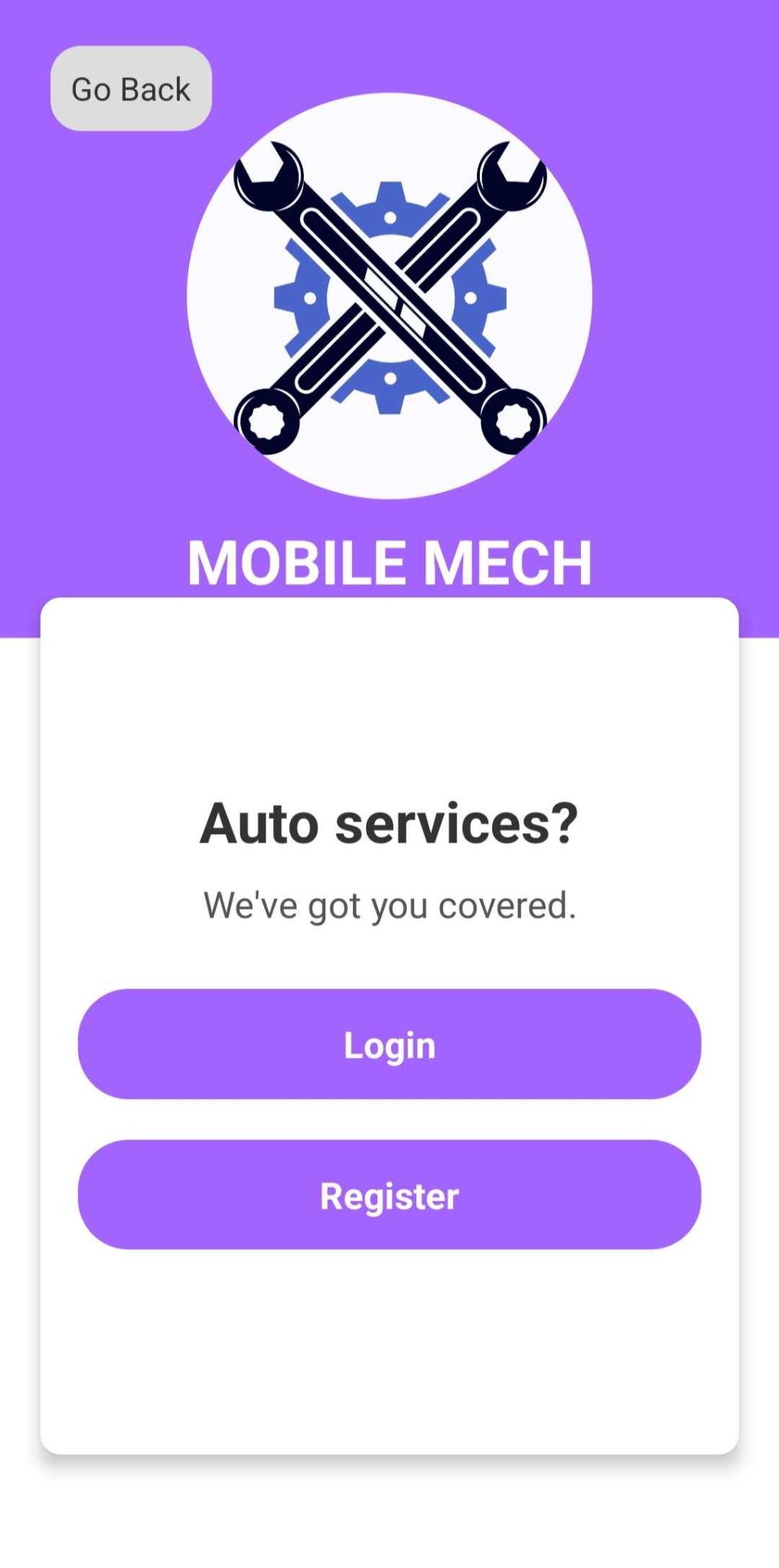
USER MANUAL DOCUMENT

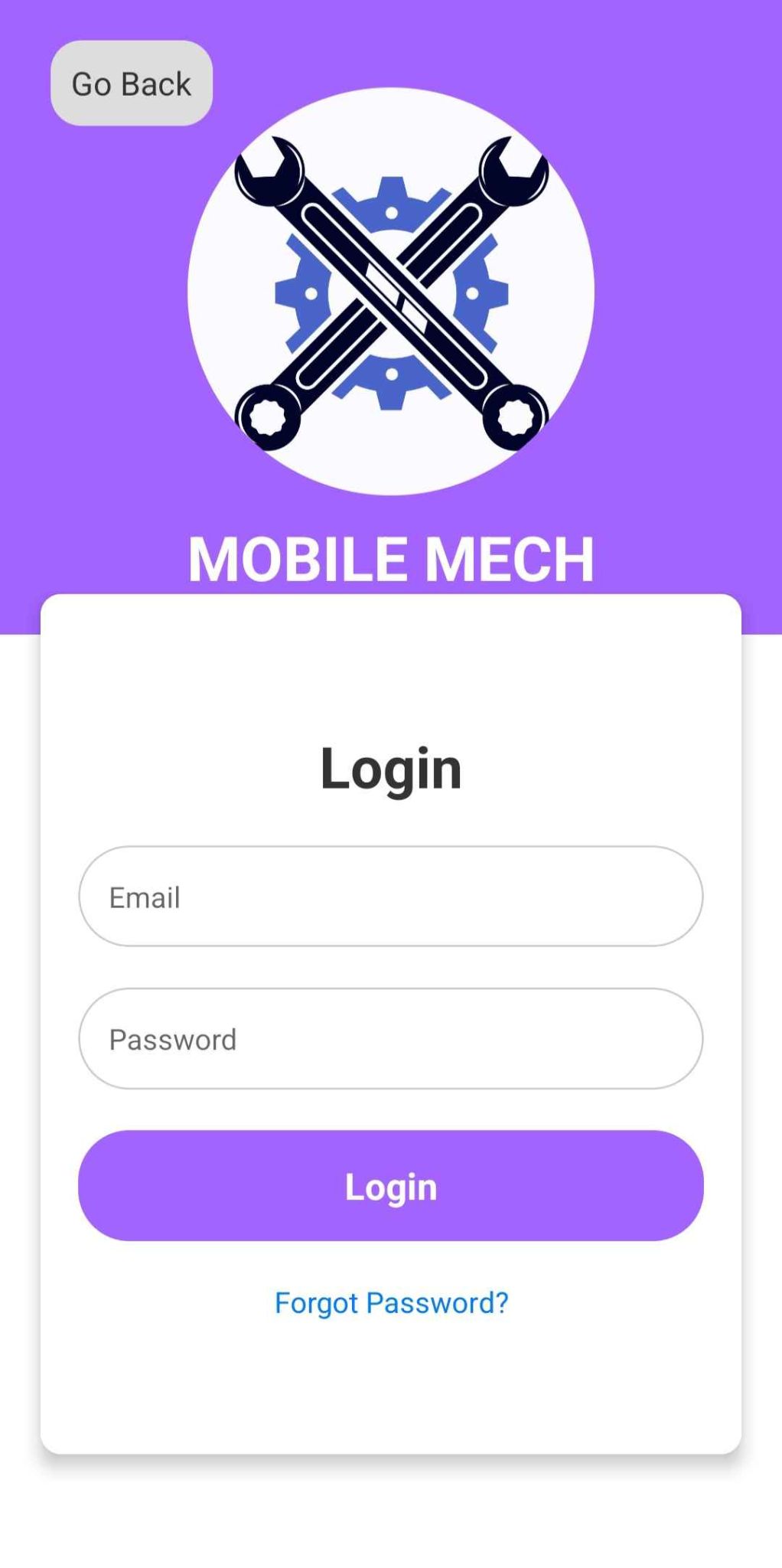
Page Index

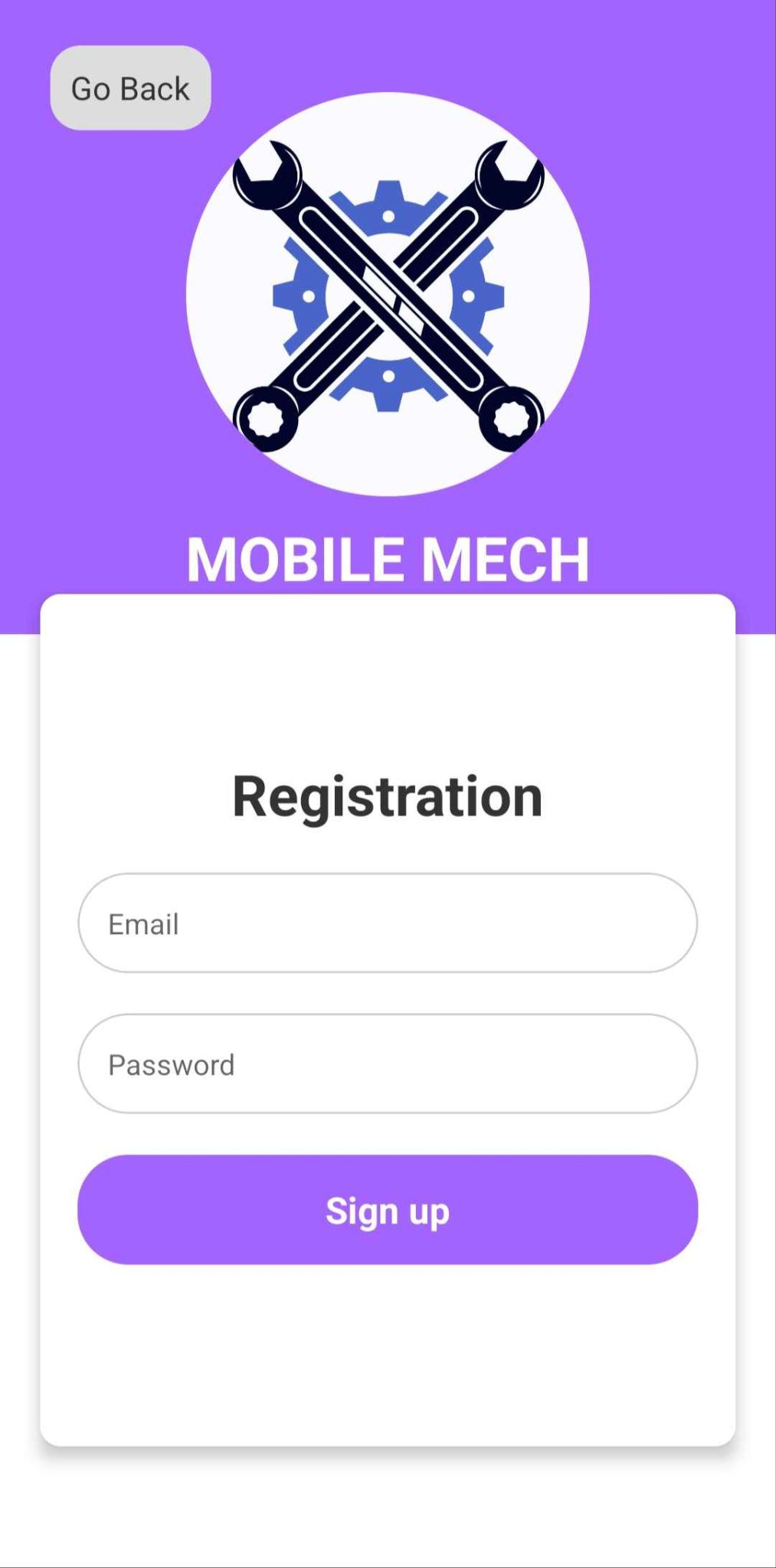
1. Welcome
2. User Login/Register
3. User Login
4. User Register
5. User Search
6. User Messaging
7. User Profile
8. Mechanic Login/Register
9. Mechanic Login
10. Mechanic Register
11. Mechanic appointments
12. Mechanic Messaging
13. Mechanic Profile

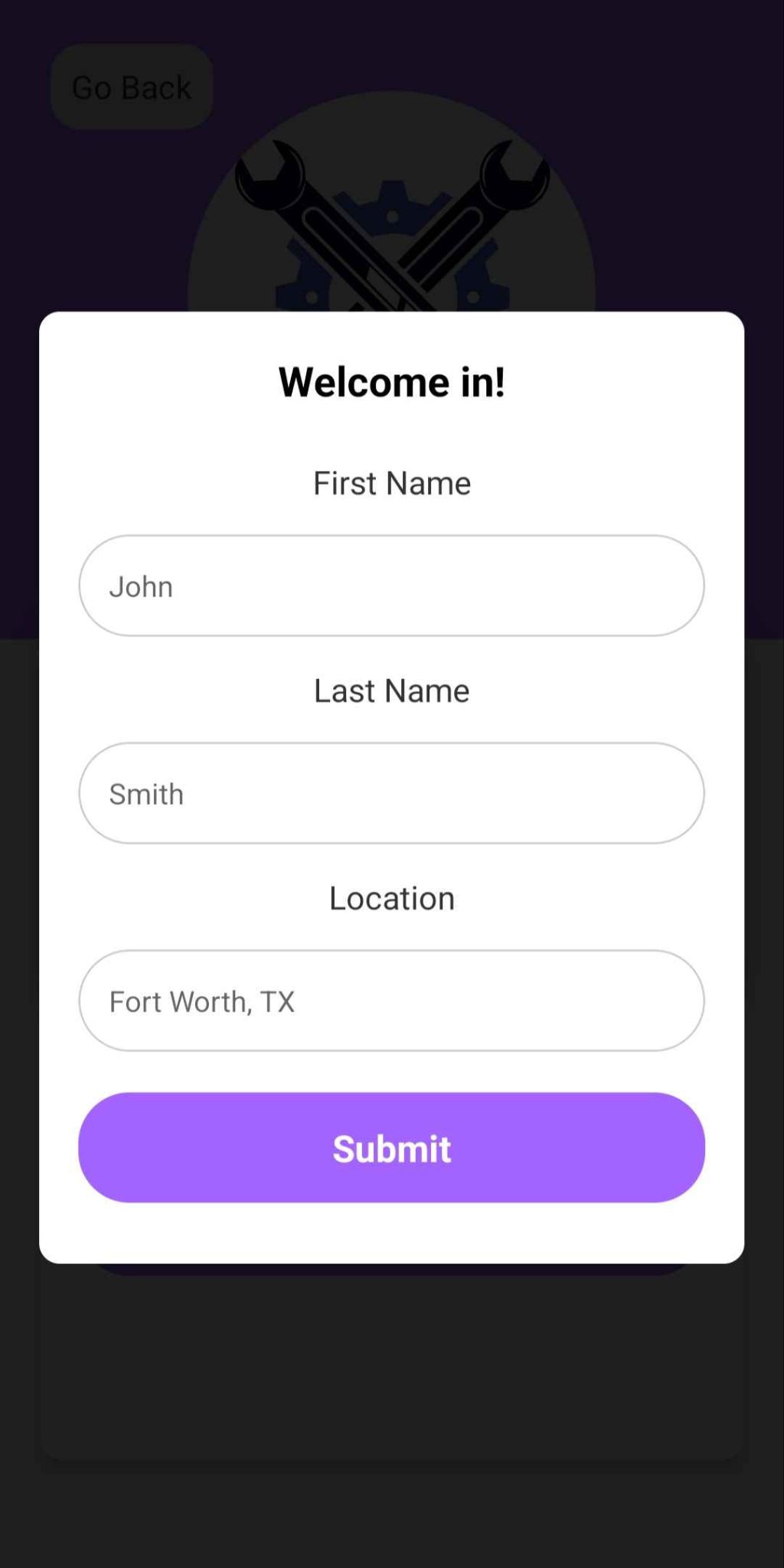
Welcome



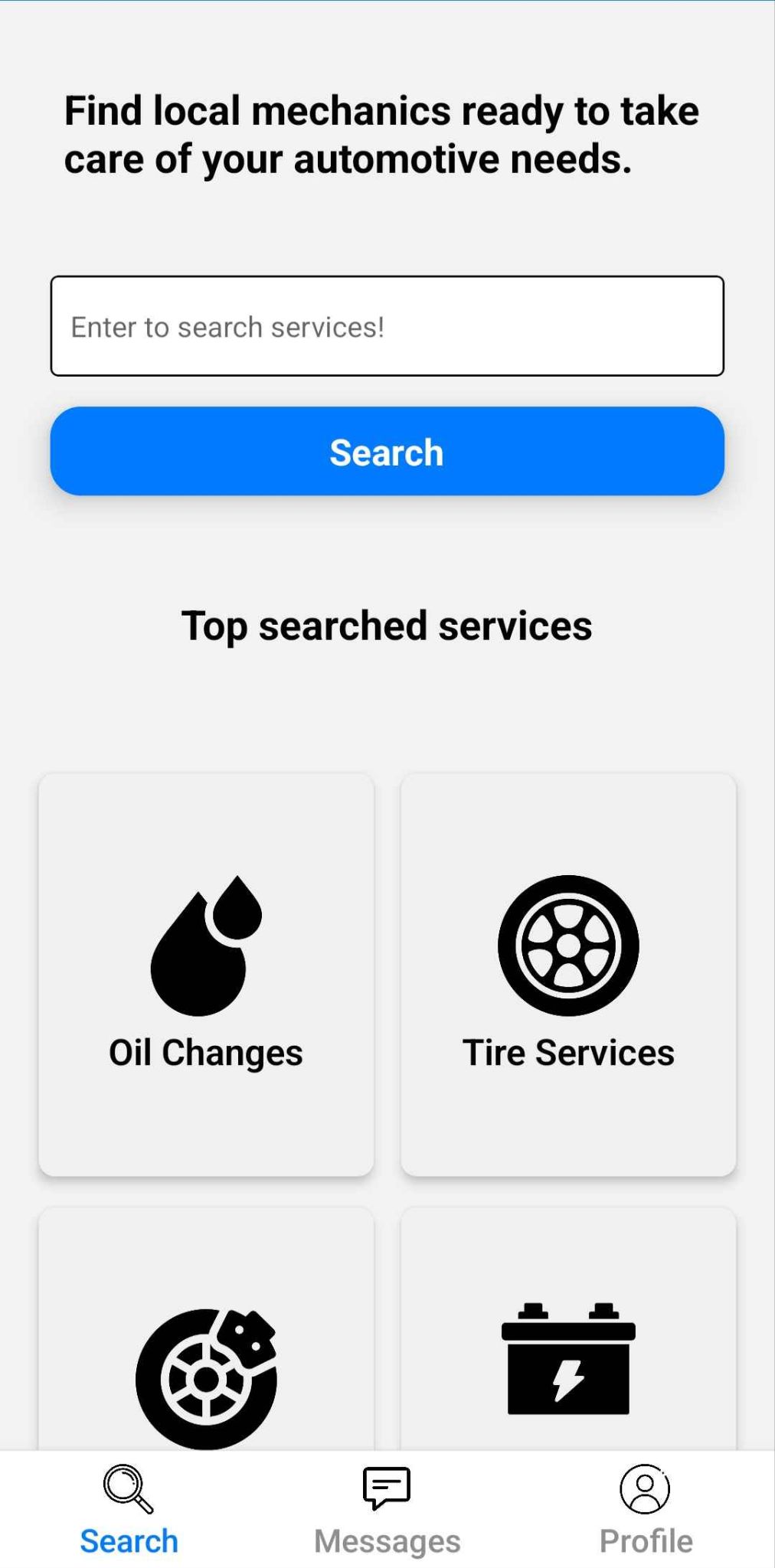
User Login/Register

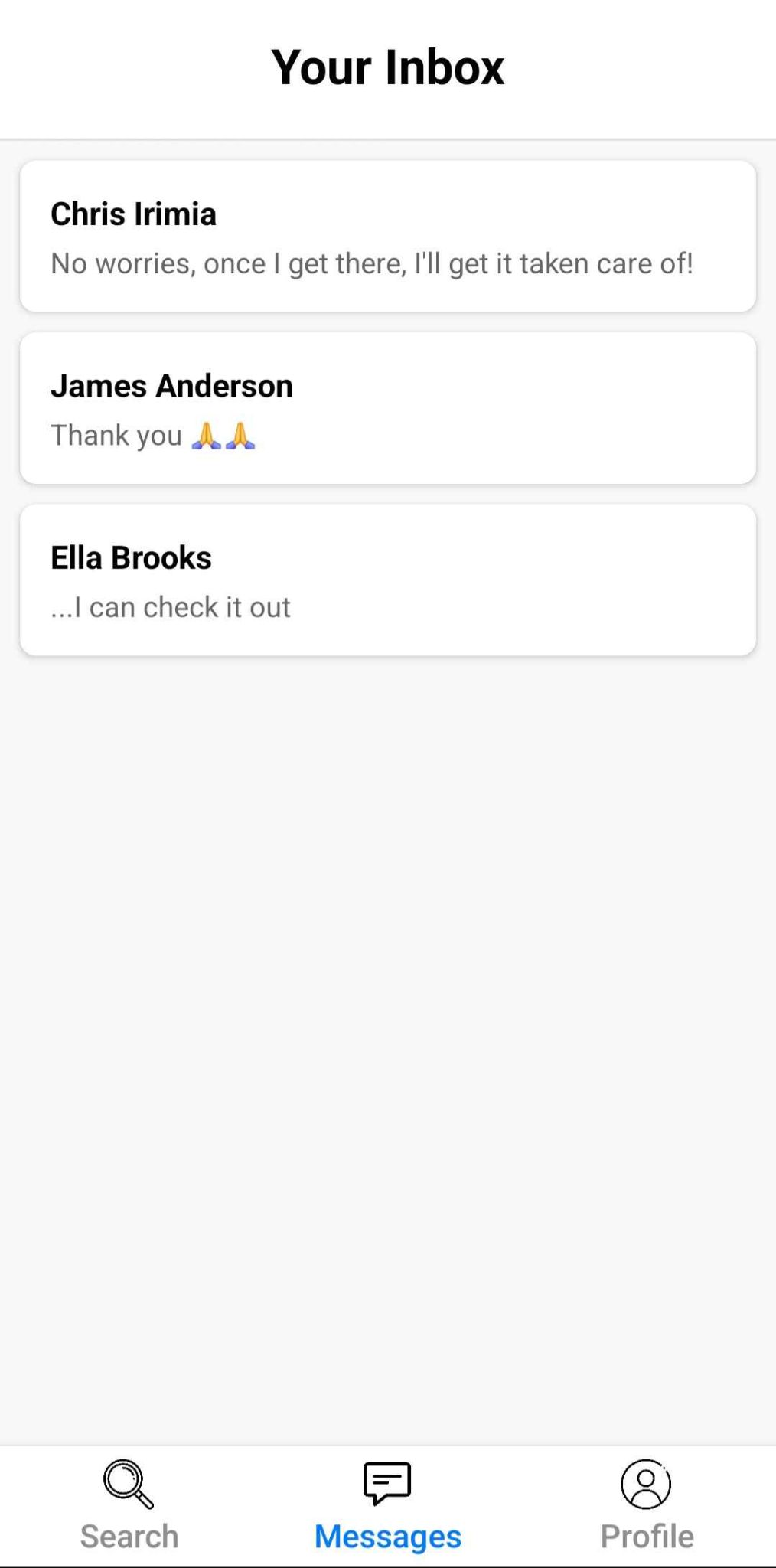
User Login

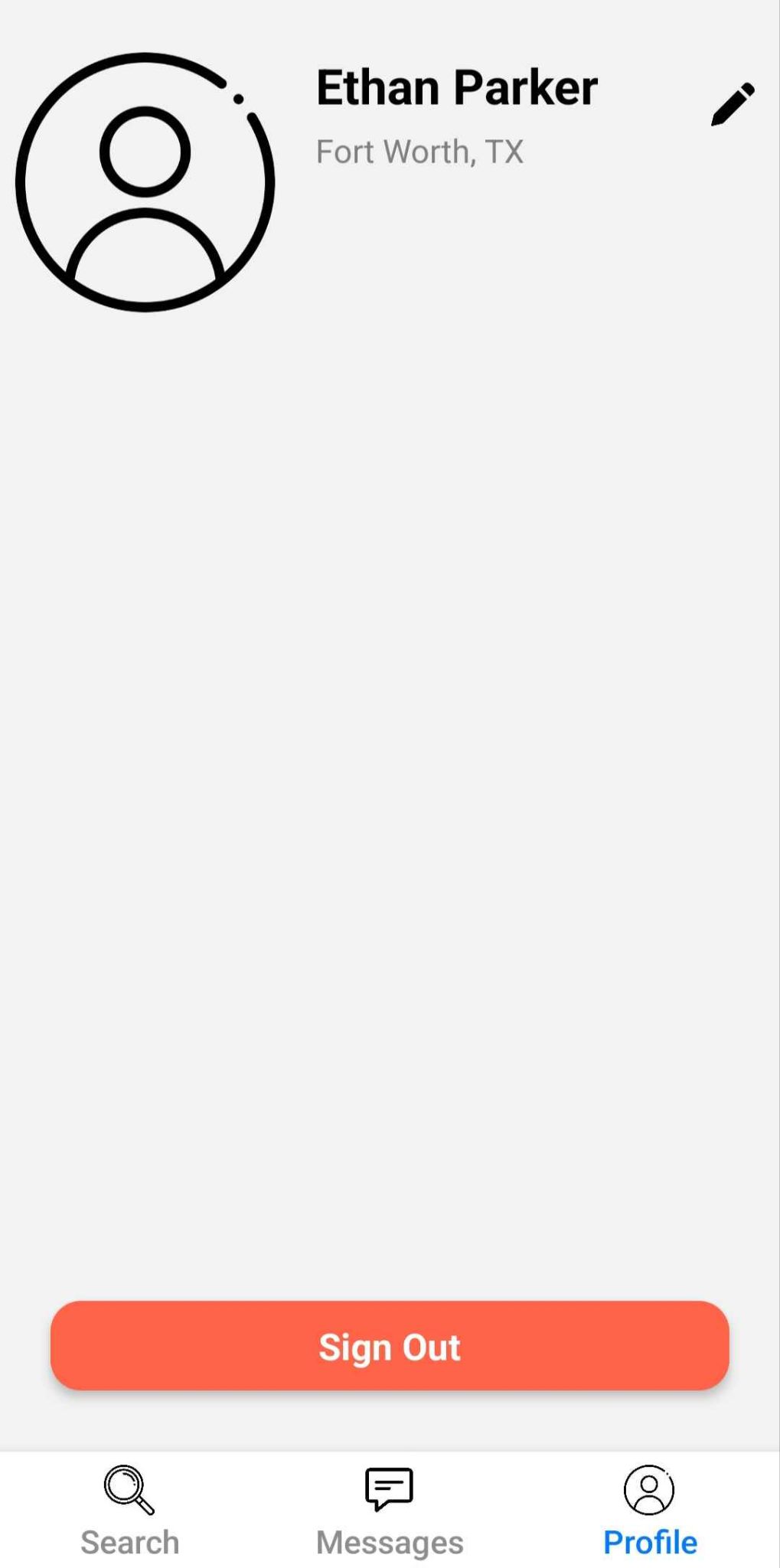
User Register

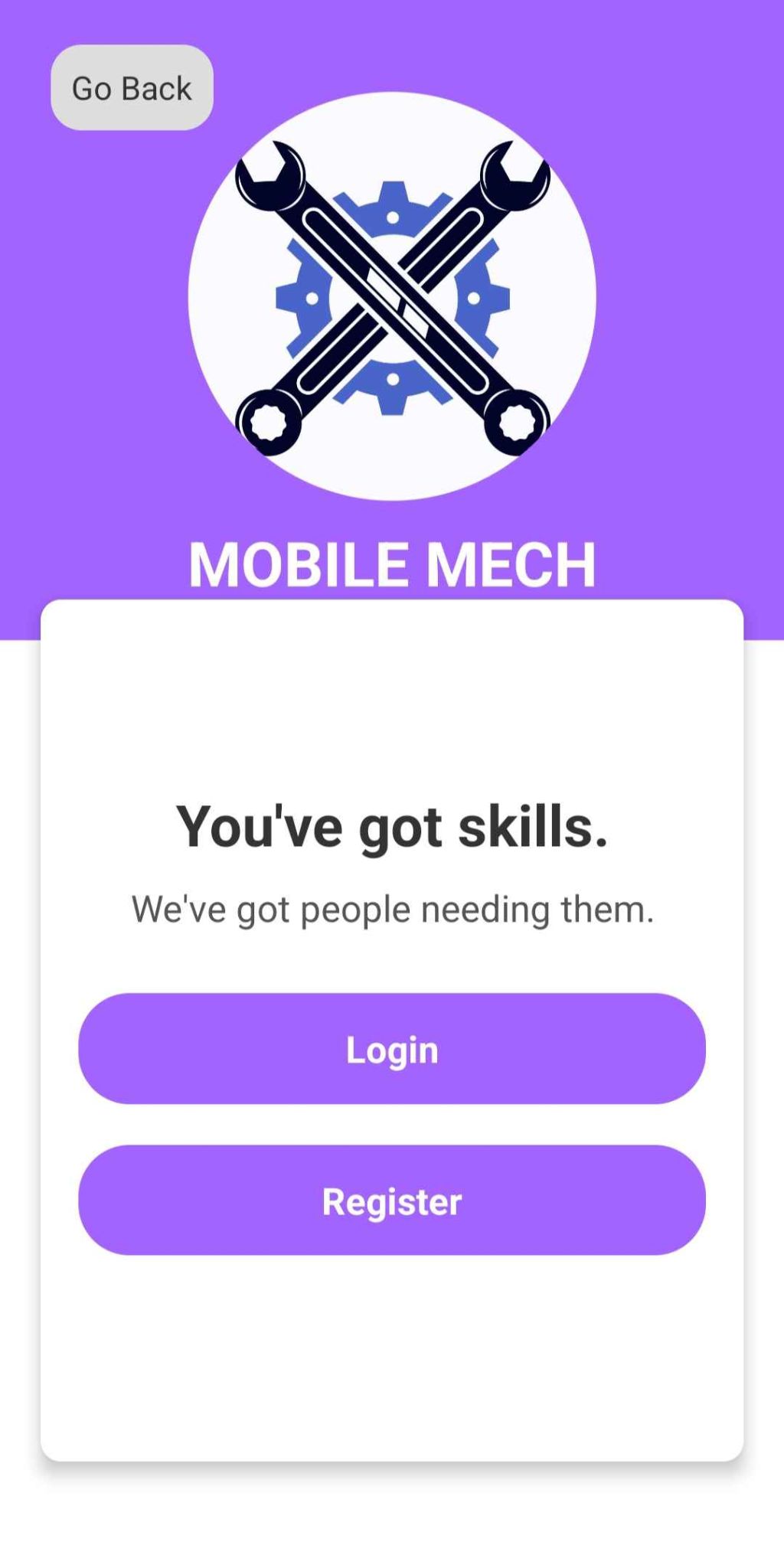


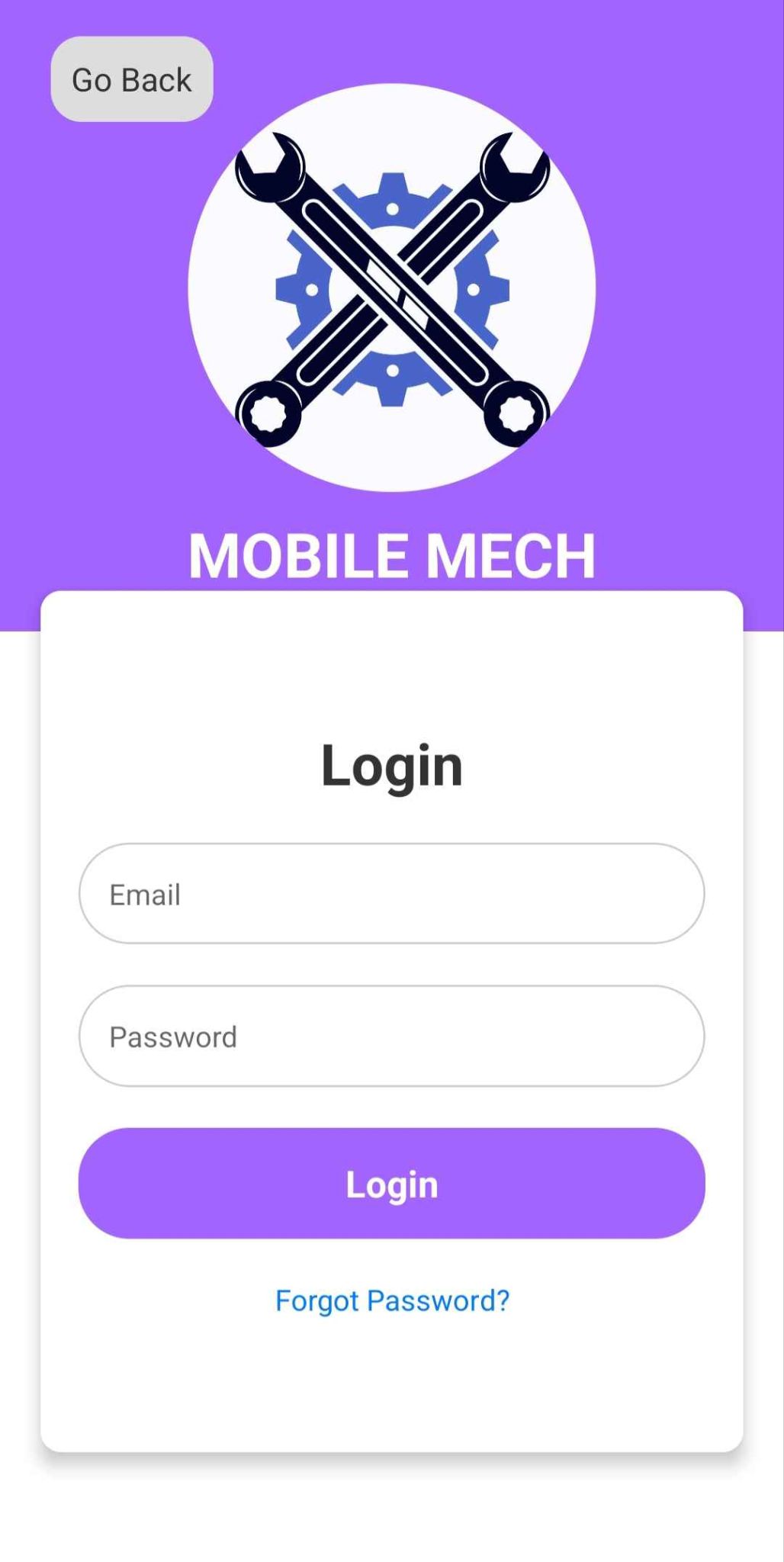
User Search

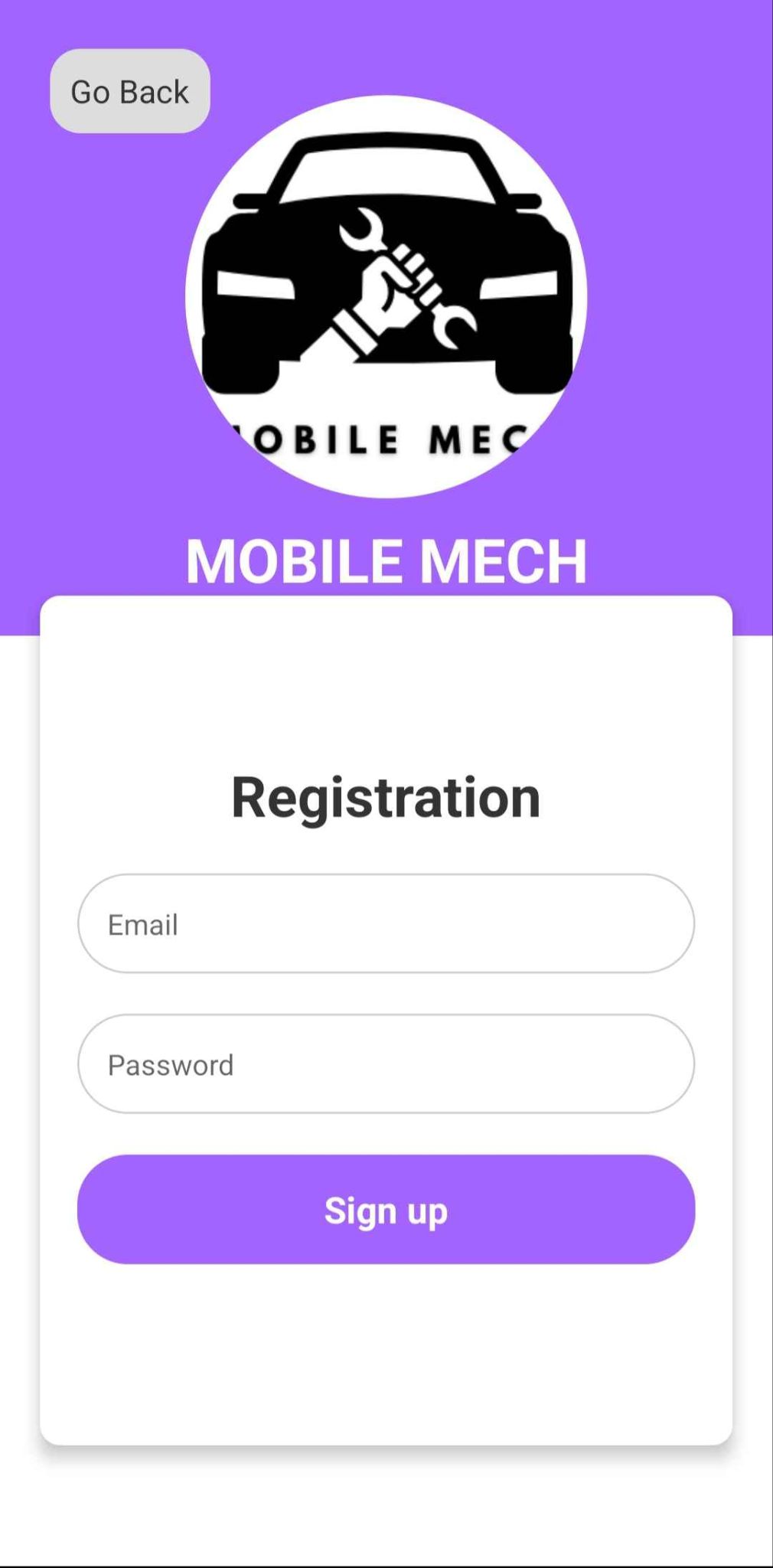


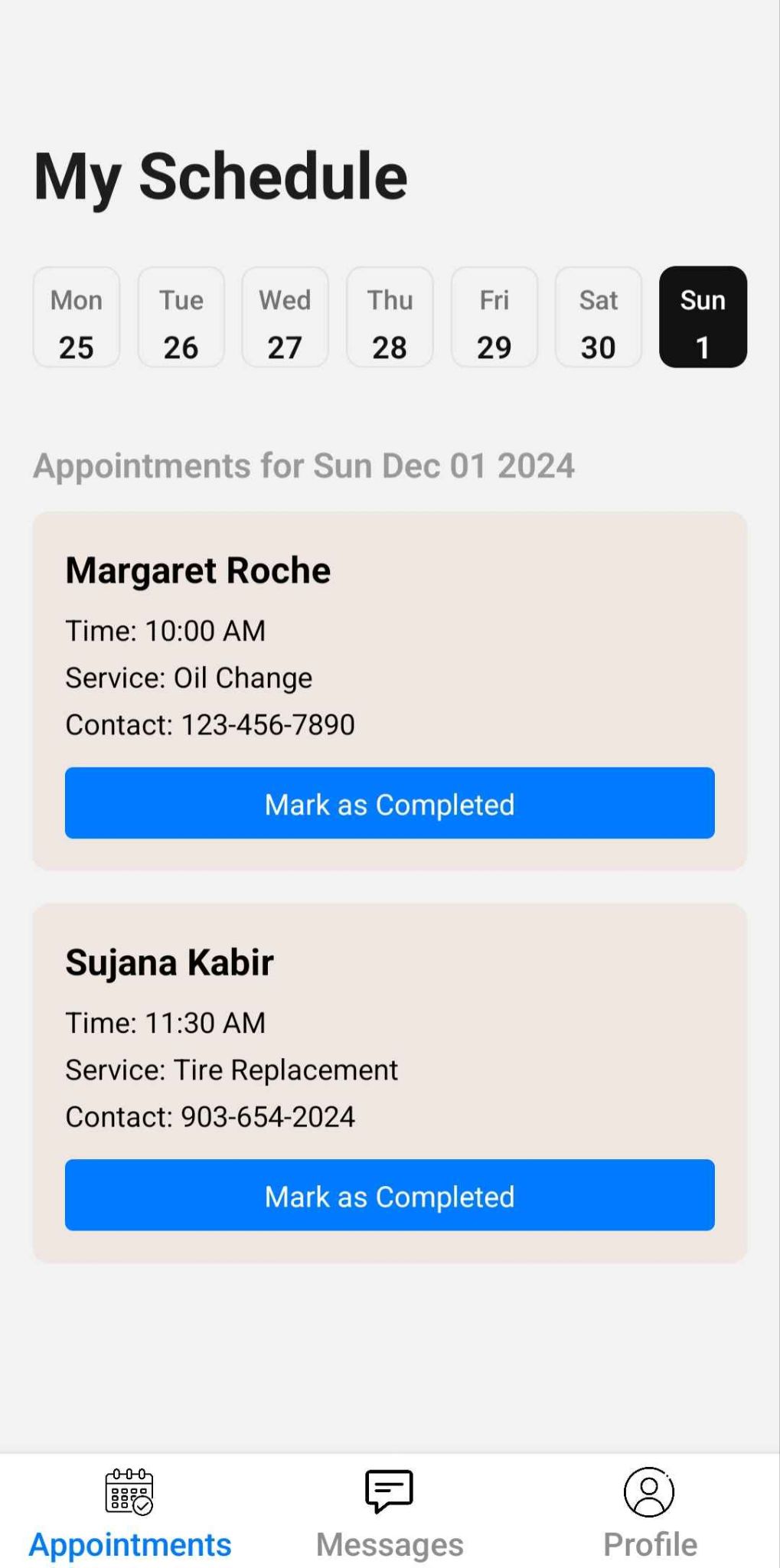
User Messaging

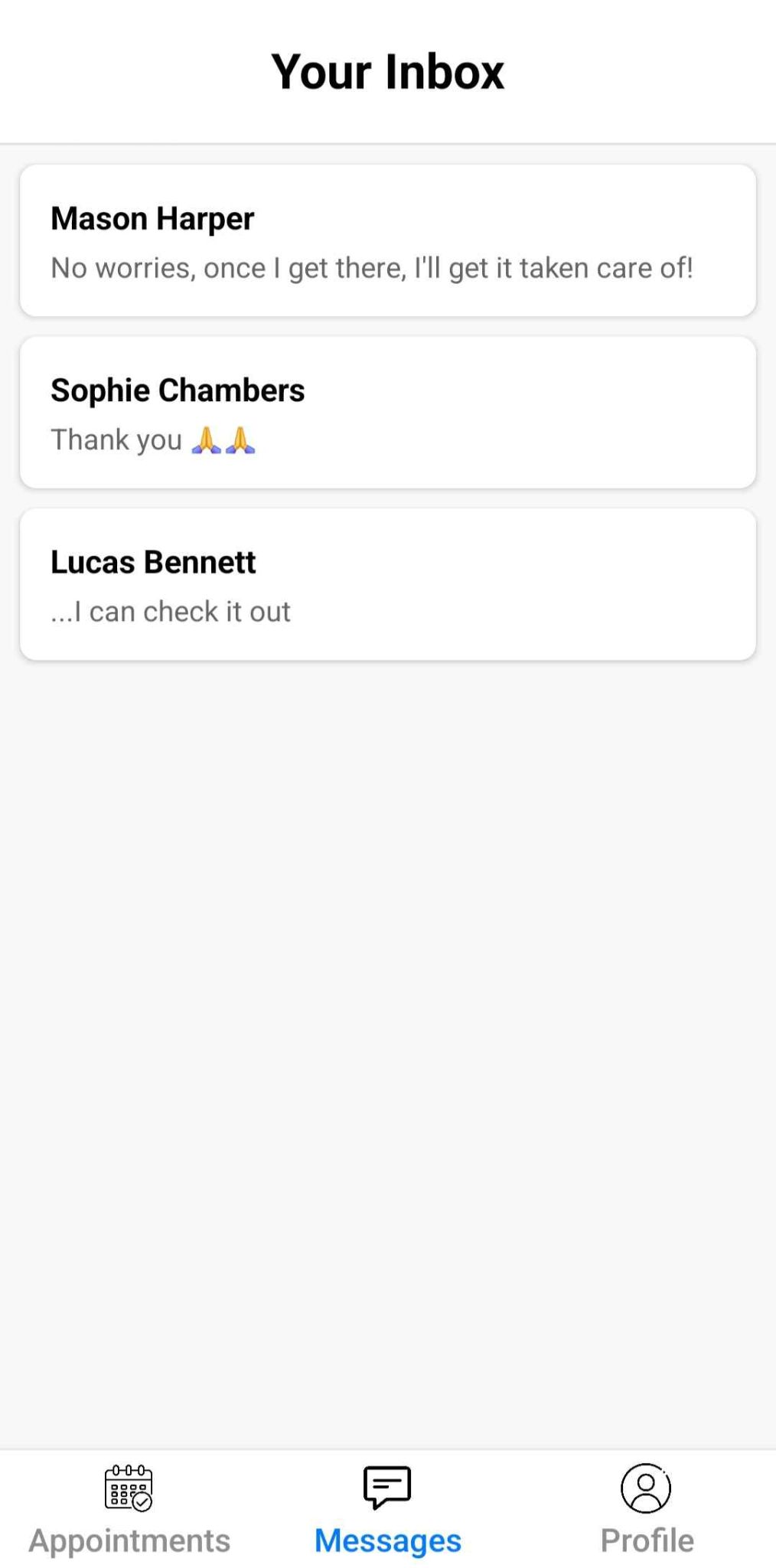
User Profile

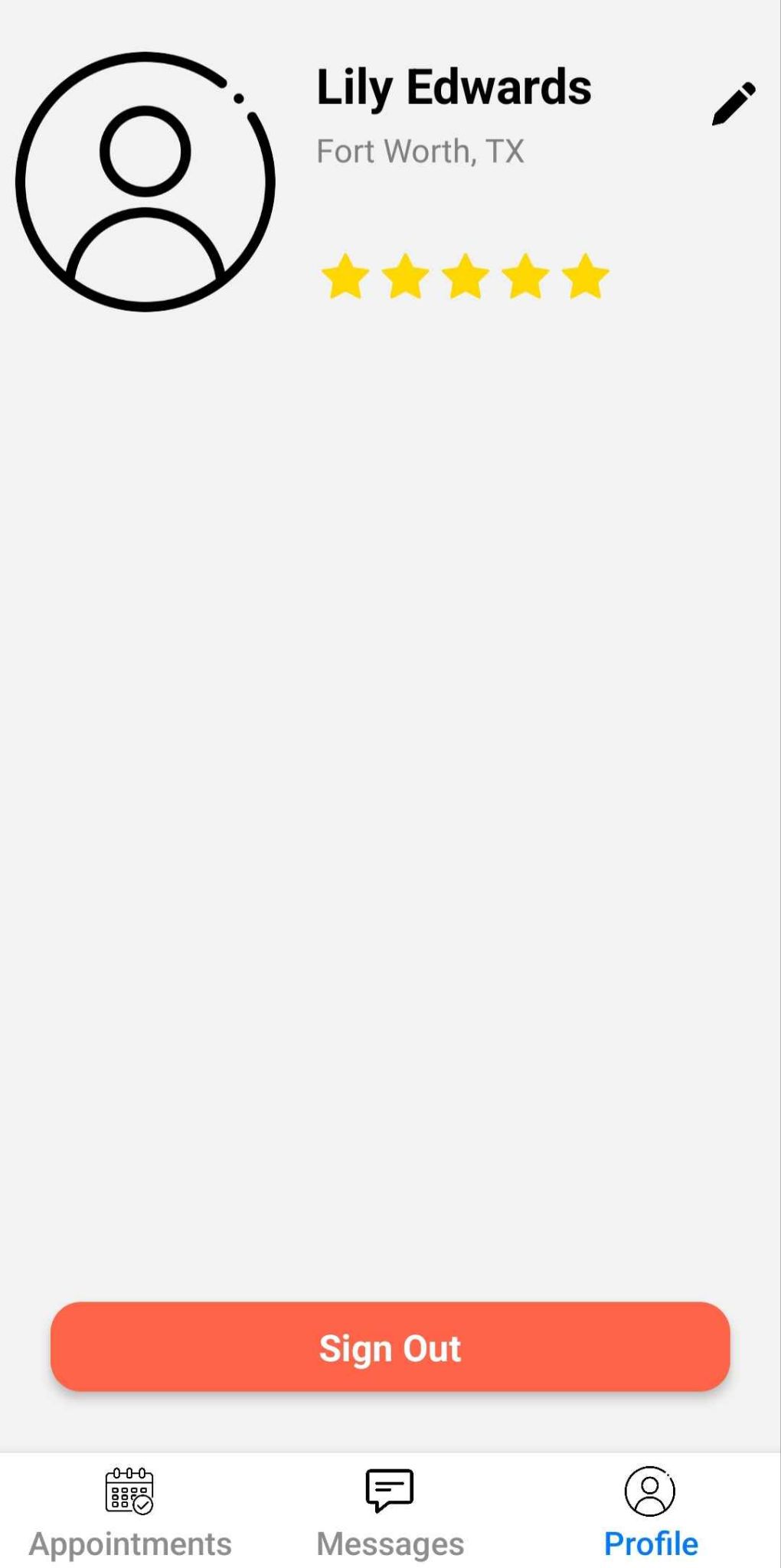
Mechanic Login/Register

Mechanic Login

Mechanic Register

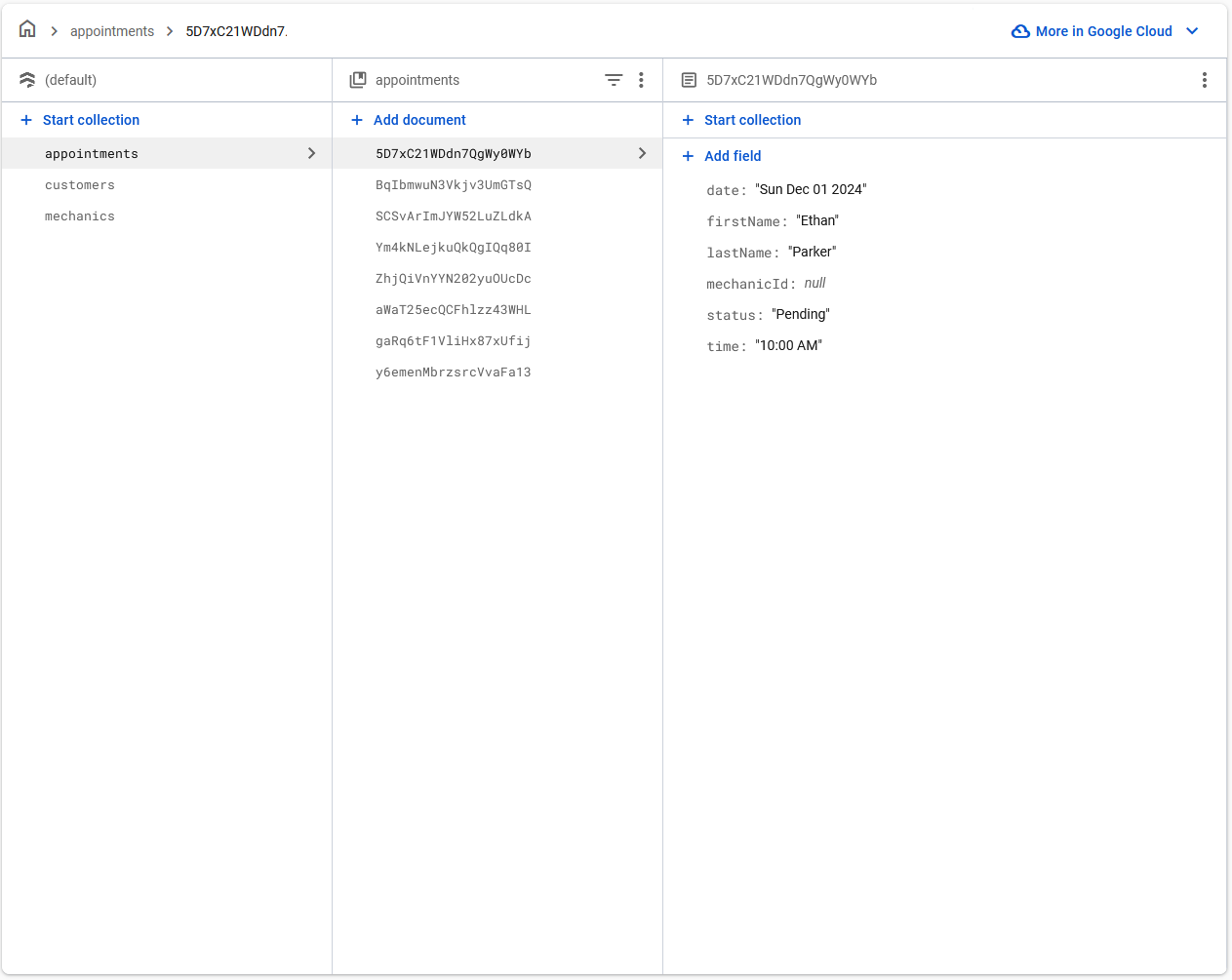
Mechanic appointments

Mechanic Messaging

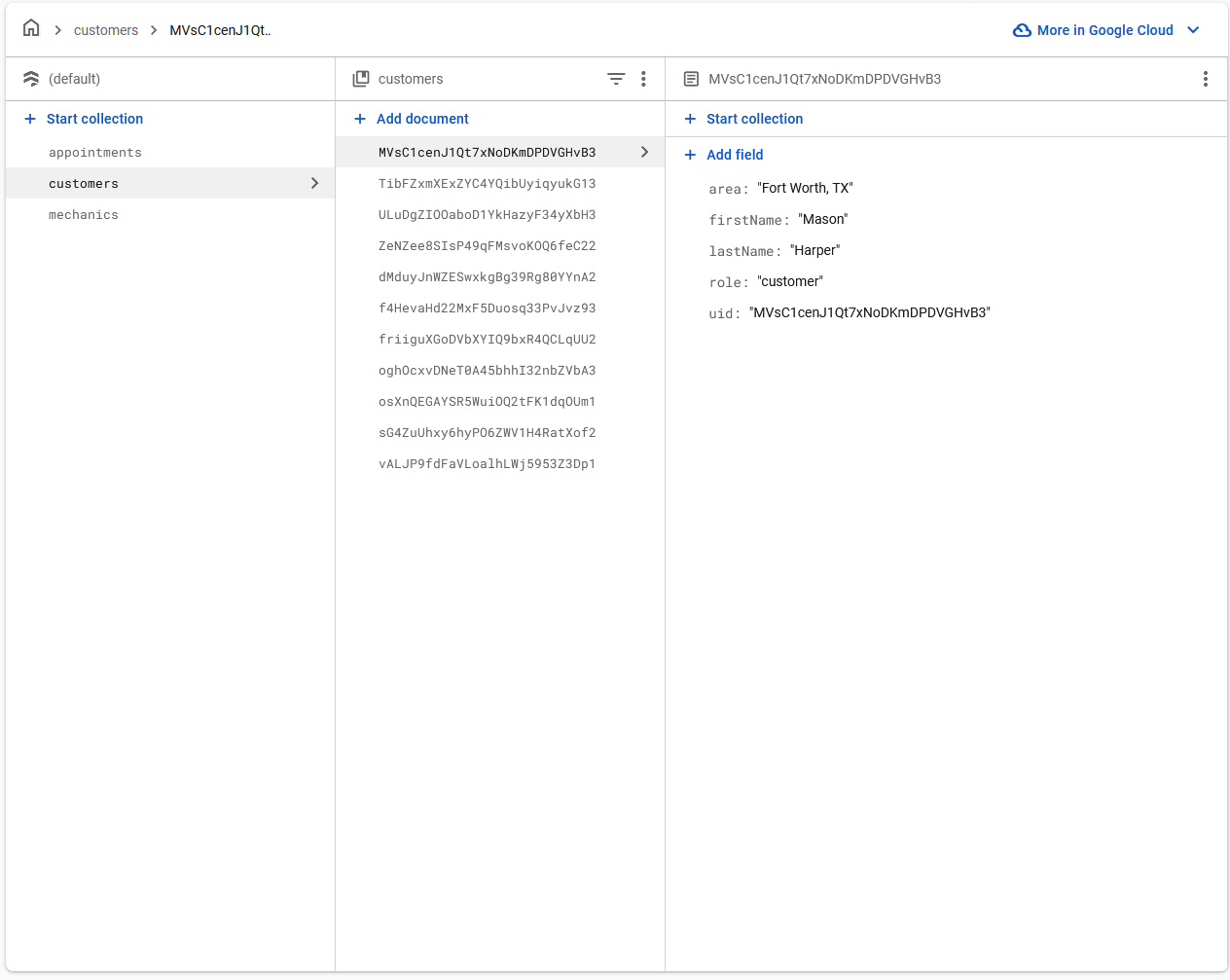
Mechanic Profile

SELECTED SOURCE CODE

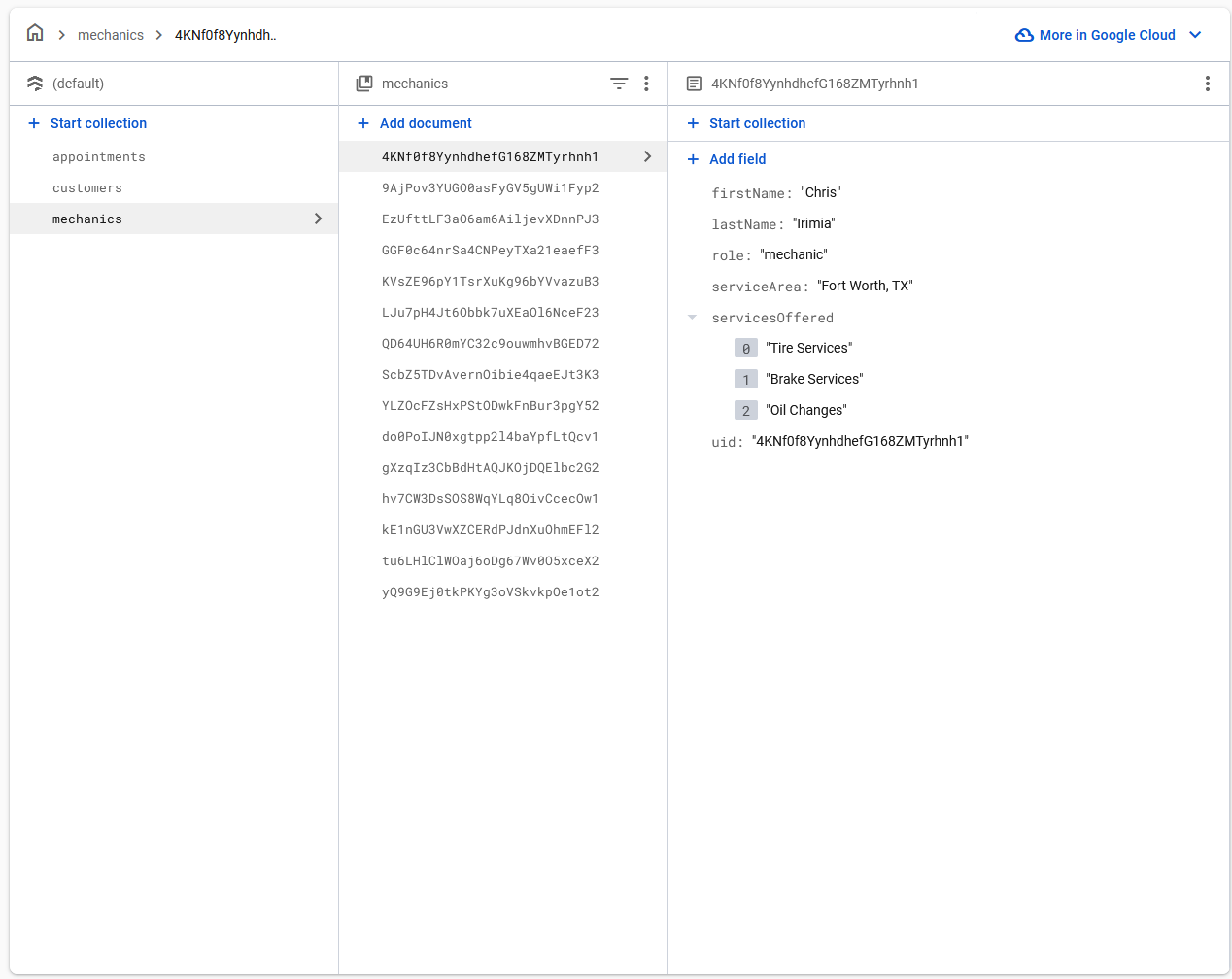
Firebase Appointment Data Configuration



Firebase Customer Data Configuration



Firebase Mechanic Data Configuration



FireBase Configuration

import { initializeApp } from "firebase/app";

import { getAnalytics } from "firebase/analytics";

import { getAuth } from "firebase/auth";

import { collection, getFirestore } from "firebase/firestore";

// https://firebase.google.com/docs/web/setup#available-libraries

// Firebase configuration

const firebaseConfig = {

apiKey: "AIzaSyCayKf5ZtcOu8cSFRU85jAeEU9h6MoXxEo",

authDomain: "mobilemech-cse3310.firebaseapp.com",

projectId: "mobilemech-cse3310",

storageBucket: "mobilemech-cse3310.appspot.com",

messagingSenderId: "912072646854",

appId: "1:912072646854:web:068b2257861ce5765bf7c7",

measurementId: "G-92NYP3J7T3"

};

// Initialize Firebase

export const FIREBASE\_APP = initializeApp(firebaseConfig);

export const FIREBASE\_AUTH = getAuth(FIREBASE\_APP);

export const FIREBASE\_DB = getFirestore(FIREBASE\_APP);

export const mechsCollection = collection(FIREBASE\_DB, 'mechanics');

export const customersCollection = collection(FIREBASE\_DB, 'customers');

export const appointmentsCollection = collection(FIREBASE\_DB, 'appointments');

Appointment Booking

import React, { useState, useRef, useEffect } from "react";

import {

StyleSheet,

Dimensions,

SafeAreaView,

View,

Text,

TouchableOpacity,

TouchableWithoutFeedback,

FlatList,

} from "react-native";

import { useNavigation, useRoute } from '@react-navigation/native';

import { FIREBASE\_AUTH, FIREBASE\_DB } from "../../../FirebaseConfig";

import { doc, getDoc, addDoc, collection } from "firebase/firestore";

import { styles } from "../mechstyle";

const { width } = Dimensions.get("window");

export default function Appointments() {

const navigation = useNavigation();

const route = useRoute();

const mechanicUid = route.params?.mechanicUid || null;

const [value, setValue] = useState(new Date());

const [week, setWeek] = useState(0);

const [selectedTime, setSelectedTime] = useState(null);

const [userInfo, setUserInfo] = useState({ firstName: "", lastName: "" });

const [dailyAvailableSlots, setDailyAvailableSlots] = useState({});

const flatListRef = useRef(null);

// Fixed time slots from 9 AM to 5 PM with 1-hour intervals

const generateTimeSlots = () => {

const slots = [];

for (let hour = 9; hour < 17; hour++) {

const formattedHour = hour < 12 ? `${hour}:00 AM` : `${hour - 12}:00 PM`;

slots.push(formattedHour);

}

return slots;

};

useEffect(() => {

initializeSlotsForWeek();

fetchUserInfo();

}, []);

const initializeSlotsForWeek = () => {

const slots = generateTimeSlots();

const newSlots = {};

// Initialize slots for the next 7 days

for (let i = 0; i < 7; i++) {

const date = addDays(new Date(), i);

newSlots[date.toDateString()] = [...slots];

}

setDailyAvailableSlots(newSlots);

};

const fetchUserInfo = async () => {

const user = FIREBASE\_AUTH.currentUser;

if (!user) {

console.error("User not logged in");

return;

}

try {

const userDoc = doc(FIREBASE\_DB, "customers", user.uid);

const docSnap = await getDoc(userDoc);

if (docSnap.exists()) {

const { firstName, lastName } = docSnap.data();

setUserInfo({ firstName, lastName });

} else {

console.error("User document does not exist");

}

} catch (error) {

console.error("Error fetching user info:", error);

}

};

const handleBookAppointment = async () => {

if (!selectedTime) {

alert("Please select a time slot.");

return;

}

const user = FIREBASE\_AUTH.currentUser;

if (!user) {

alert("You must be logged in to book an appointment.");

return;

}

try {

const appointmentsCollection = collection(FIREBASE\_DB, "appointments");

const appointment = {

firstName: userInfo.firstName || "",

lastName: userInfo.lastName || "",

date: value.toDateString(),

time: selectedTime,

mechanicId: mechanicUid, // Use mechanicUid from route params

status: "Pending",

};

await addDoc(appointmentsCollection, appointment);

setDailyAvailableSlots((prevSlots) => {

const updatedSlots = { ...prevSlots };

updatedSlots[value.toDateString()] = updatedSlots[

value.toDateString()

].filter((slot) => slot !== selectedTime);

return updatedSlots;

});

setSelectedTime(null);

navigation.navigate("Payment");

} catch (error) {

console.error("Error booking appointment:", error);

}

};

const getStartOfWeek = (date) => {

const day = date.getDay();

const diff = date.getDate() - day + (day === 0 ? -6 : 1);

return new Date(date.setDate(diff));

};

const addDays = (date, days) => {

const newDate = new Date(date);

newDate.setDate(newDate.getDate() + days);

return newDate;

};

const formatDate = (date, format) => {

const options =

format === "weekday"

? { weekday: "short" }

: { day: "numeric" };

return new Intl.DateTimeFormat("en-US", options).format(date);

};

const weeks = React.useMemo(() => {

const startOfCurrentWeek = getStartOfWeek(

new Date(new Date().setDate(new Date().getDate() + week \* 7))

);

return [-1, 0, 1].map((adj) => {

return Array.from({ length: 7 }).map((\_, index) => {

const date = addDays(startOfCurrentWeek, adj \* 7 + index);

return {

weekday: formatDate(date, "weekday"),

date,

};

});

});

}, [week]);

const handleScrollEnd = (event) => {

const newIndex = Math.round(event.nativeEvent.contentOffset.x / width);

if (newIndex === 1) return;

const newWeek = week + (newIndex - 1);

setWeek(newWeek);

setValue(addDays(value, (newIndex - 1) \* 7));

setTimeout(() => {

flatListRef.current.scrollToIndex({ index: 1, animated: false });

}, 100);

};

const availableSlots =

dailyAvailableSlots[value.toDateString()] || generateTimeSlots();

return (

<SafeAreaView style={styles.container}>

<TouchableOpacity

style={styles.goBackButton}

onPress={() => navigation.goBack()}

>

<Text style={styles.goBackText}>Go Back</Text>

</TouchableOpacity>

<View style={styles.picker}>

<FlatList

ref={flatListRef}

data={weeks}

horizontal

showsHorizontalScrollIndicator={false}

pagingEnabled

initialScrollIndex={1}

onScrollEndDrag={handleScrollEnd}

keyExtractor={(item, index) => index.toString()}

getItemLayout={(\_, index) => ({

length: width,

offset: width \* index,

index,

})}

renderItem={({ item: dates }) => (

<View style={styles.itemRow}>

{dates.map((item, dateIndex) => {

const isActive =

value.toDateString() === item.date.toDateString();

return (

<TouchableWithoutFeedback

key={dateIndex}

onPress={() => setValue(item.date)}

>

<View

style={[

styles.item,

isActive && {

backgroundColor: "#a363ff",

borderColor: "#a363ff",

},

]}

>

<Text

style={[

styles.itemWeekday,

isActive && { color: "#fff" },

]}

>

{item.weekday}

</Text>

<Text

style={[

styles.itemDate,

isActive && { color: "#fff" },

]}

>

{item.date.getDate()}

</Text>

</View>

</TouchableWithoutFeedback>

);

})}

</View>

)}

/>

</View>

<View style={{ flex: 1, paddingHorizontal: 16, paddingVertical: 24 }}>

<Text style={styles.subtitle}>{value.toDateString()}</Text>

<View>

{availableSlots.length > 0 ? (

availableSlots.map((slot, index) => (

<TouchableOpacity

key={index}

onPress={() => setSelectedTime(slot)}

style={[

styles.slotCard,

selectedTime === slot && { backgroundColor: "#a363ff" },

]}

>

<Text

style={[

styles.slotText,

selectedTime === slot && { color: "#fff" },

]}

>

{slot}

</Text>

</TouchableOpacity>

))

) : (

<Text style={{ textAlign: "center", color: "#999" }}>

No available time slots.

</Text>

)}

</View>

</View>

<View style={styles.footer}>

<TouchableOpacity onPress={handleBookAppointment}>

<View style={styles.btn}>

<Text style={styles.btnText}>Book Appointment</Text>

</View>

</TouchableOpacity>

</View>

</SafeAreaView>

);

}