**SC2006 Software Engineering**

****

****

Lab Group SSP1

Group Name: Depressed Elderly

Lab Supervisor: Liu Yang

|  |  |
| --- | --- |
| Group Members | |
| Name | Matriculation No. |
| Pan Shi Han Johnny | U2121697B |
| Bertranne Lee Zi - Wen | U2120209L |
| Yap Shen Hwei | U2140630E |
| Koh Jia Wei | U2121270D |
| Zhi Guang | U2120504C |
| Chong Yue Hong | U2120181D |

[**1. Introduction**](#_xjrwjkhe2aoj) **4**

[1.1. Purpose](#_hv7ce13h960v) 4

[1.2. Intended Audience](#_bqqnoc44q4s1) 4

[1.3. Product Scope](#_52u1jhyntj3g) 4

[**2. Overall Description**](#_9yktl29it2hc) **5**

[2.1. Product Functions](#_7sxgnahah1yk) 6

[2.2. User Classes & Characteristics](#_ef16ulkj10b6) 7

[2.3. Operating Environment](#_3j8zcssrl60) 7

[2.4. Design & Implementation Constraints](#_iemozsj6bzff) 7

[2.5. User Documentation](#_26drahviikg4) 7

[2.6. Assumptions & Dependencies](#_fi2lu636ph05) 7

[**3. External Hardware Requirements**](#_59jd76vtx79c) **8**

[3.1. User Interface](#_865iry5fuk1q) 8

[3.2. Hardware Interfaces](#_nhvcsn4eb6zh) 22

[3.3. Software Interfaces](#_4yq4i8epudll) 22

[**4. System Requirements**](#_xcg0ll14gas6) **24**

[4.1. Account](#_3dy6vkm) 24

[4.2. Job](#_1t3h5sf) 25

[**5. Other Non-Functional Requirements**](#_vwo1a47dz79y) **27**

[5.1. Performance](#_ug2oi03qu2ts) 27

[5.2. Safety Requirements](#_5b64fghiv86u) 27

[5.3. Security Requirements](#_up6qc6dquau2) 27

[5.4. Data Dictionary](#_kfr1myagv0uz) 28

[**6. Use Case Description**](#_o869e140lkxa) **30**

[Create Account](#_vjs7outuv4e1) 30

[Login](#_coxqb1ksos4k) 32

[Edit Account Information (User)](#_88axh4cnyz63) 33

[View Listing](#_2tsca9i8ywuu) 34

[Update Job Status](#_te6hvbhd6rak) 35

[Rate User](#_pv4izygouthb) 37

[Report Emergency](#_s3itwdlvlgy9) 38

[Report to Customer Service](#_w98ho452luij) 39

[Manage Job](#_lc1jmeiih81q) 40

[Manage Service](#_gdqg3bplqpm2) 42

[Accept Job](#_1w4463ehj24l) 42

[Create Chat Box](#_i9juq8i5s87n) 45

[**Appendix A: Analysis Model**](#_wa4x7opgub8y) **46**

[Use Case Diagram](#_19wnwy3cra9u) 46

[Dialog Map](#_ehpm1folvvmc) 47

[Sequence Diagrams](#_4l493q8huet4) 48

[**Appendix B: Unit Testing**](#_7qxiv2uq1prv) **61**

[Black Box](#_l7jv7h3y5hyt) 61

[White Box](#_4p62ljjkq5ds) 64

[**Appendix C: Final Notes**](#_re7tviqhx8i8) **67**

# 

# Introduction

## Purpose

Our group, Depressed Elderly, aims to create a mobile application, Mahjong Movers, which will enable our users to find help for odd jobs that they might have difficulty doing on their own. Our mobile application will serve as a platform for people who are seeking help and those offering help to meet and exchange their services.

## Intended Audience

This software requirement document is intended for developers, users and testers.

Developers for the Mahjong Mover application can review the documentation to understand and analyse how the features and functions of the application were derived and made through the documentation of the functional and non-functional requirements. Thereby allowing them to have a better understanding of how the application works, helping them to develop better features or functions for the application as well as the general maintenance of the application.

Users will benefit from the documentation as it will give them a greater understanding of the general working of the application. Our users will get a better understanding of the features and functions that our application can offer them, giving them a better user experience.

Lastly, for testers, this document will give them a clearer picture of the unit testing that was already done in certain parts of the application. This will help them understand how unit testing was done so that they can follow a standardised format to ensure all our features are tested in a similar way and held to a similar standard.

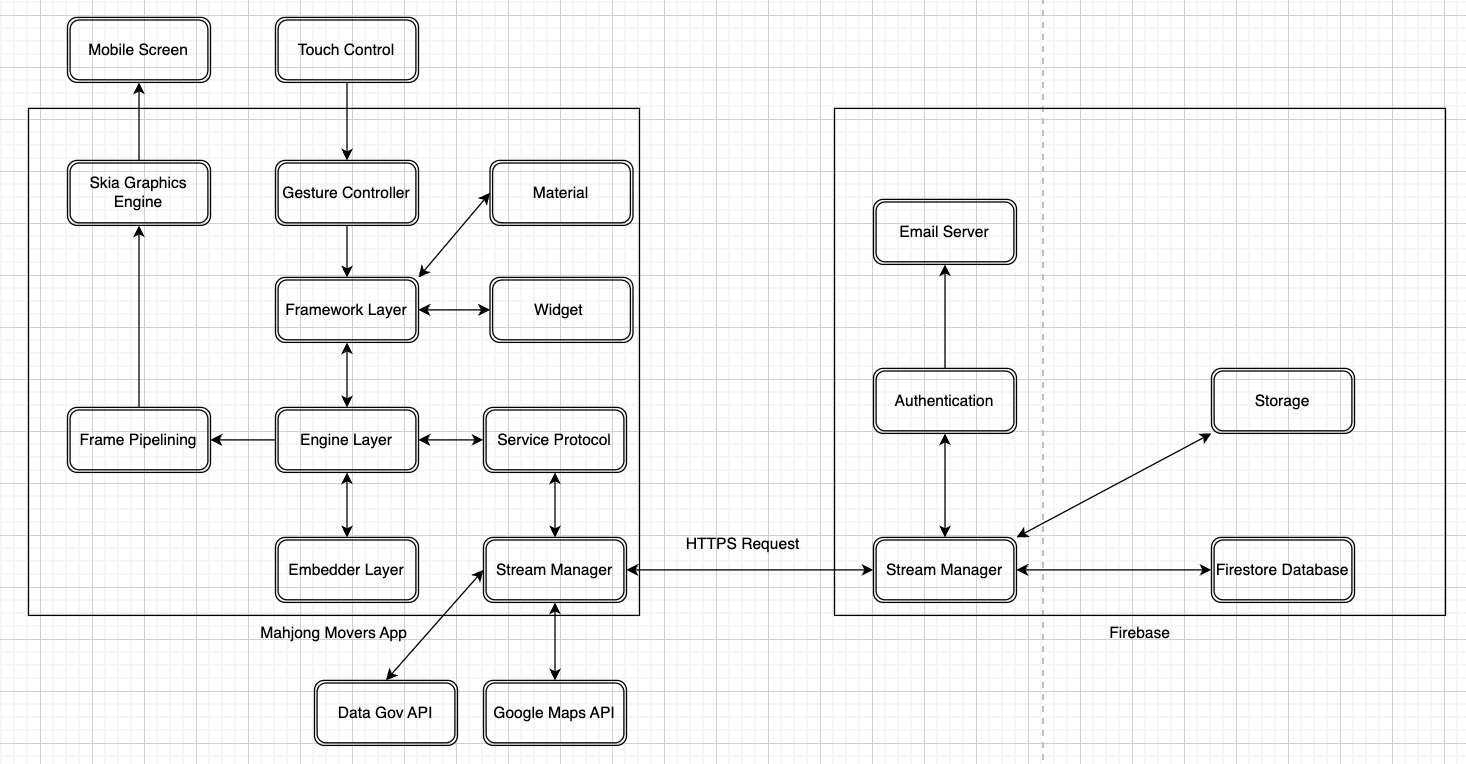
## Product Scope

The Mahjong Movers Application is an easy-to-use mobile application that will allow users to find help for odd jobs that they have. The project will be considered complete when the application has been tested and approved by our internal testing team.

# 

# Overall Description

Mahjong Movers is an application that will be used on Apple devices, specifically on the iPhone. Mahjong Movers originated from wanting to give people a platform to engage others for help for odd jobs that they might have. This application serves as a one-stop service for anyone who is in need of ad hoc assistance to request help whenever they need it.



Shown above is the system architecture for the Mahjong Movers application. Shown on the left square is the Mahjong Movers Application, encompassing the systems in place that help the application run and interact with the other systems we have in place. On the right-hand side are the Firebase systems which assist us in storing user information and transactions that occur throughout the Mahjong Mover application. The top left-hand corner of the diagram shows our inputs and output systems that the application uses, being the touch control screen and mobile screen respectively.

Finally, at the bottom of the screen are the Application Programming Interfaces (APIs) namely; Data Gov and Google Maps API that we interact with to enable the functionality of the system.

## 

## Product Functions

## 

The system must provide the following functions in accordance with the other requirements detailed within this SRS document.

* Registration of new and unique uses of the application
* Logging into said user’s account through a unique identifier for the account and password
* Allowing the user to reset their password if they forgot it
* Customisation of their profile page as well as editing existing account information
* Accepting Jobs
* Creating Jobs
* Reviewing & Rating other users of the Mahjong Mover Application.

## 

## User Classes & Characteristics

For this application, we will be generating 1 user class. Users who are interested in using the Mahjong Mover application to find help for odd jobs or to provide help to those who need assistance. A user can freely move between these two classifications. For example, a user can both post requests for help and accept requests from other users.

## Operating Environment

The Mahjong Mover application will be running on iOS and will be using Firebase to store the data of users and job offers on the application. External APIs such as Google Maps API and Data Gov Singapore will provide supporting information.

## Design & Implementation Constraints

The Mahjong Mover Application currently only supports iOS devices and the application is only in English. However, since Flutter supports cross-platform integration, Mahjong Movers can be easily generated for other platforms such as the Web or Android.

## User Documentation

A simple guide would be provided to the user through the user interface and the features of the application as a part of the application description when it is published to the app store. With a guide provided, it boosts usability from the first interaction with the user as it is clear and easy to understand.

## Assumptions & Dependencies

Users should have an apple device capable of running the Mahjong Movers application.

Users should have an internet connection to use the application.

## 

# External Hardware Requirements

## User Interface

Mahjong Movers is an intuitive iOS-based consumer-to-consumer and business-to-consumer marketplace application. It features seamless login verification, interactive pop-up prompts, and instantaneous messaging. Every page in the application is easy to use and consistent for our target users. If a user were to key in a data field incorrectly, there will be an error message prompting the user that there is an invalid entry and they should retry again. These interactive and user-friendly features improve the user’s experience.

Our Mahjong Movers application is highly interlinked and users can access the various components such as the Register Screen, Login Screen, Home Screen, Job browsing Screen, Messaging interface Screen, Rewards Screen, and Settings Screen. Users are able to traverse through our applications based on their actions and interactions with our application.

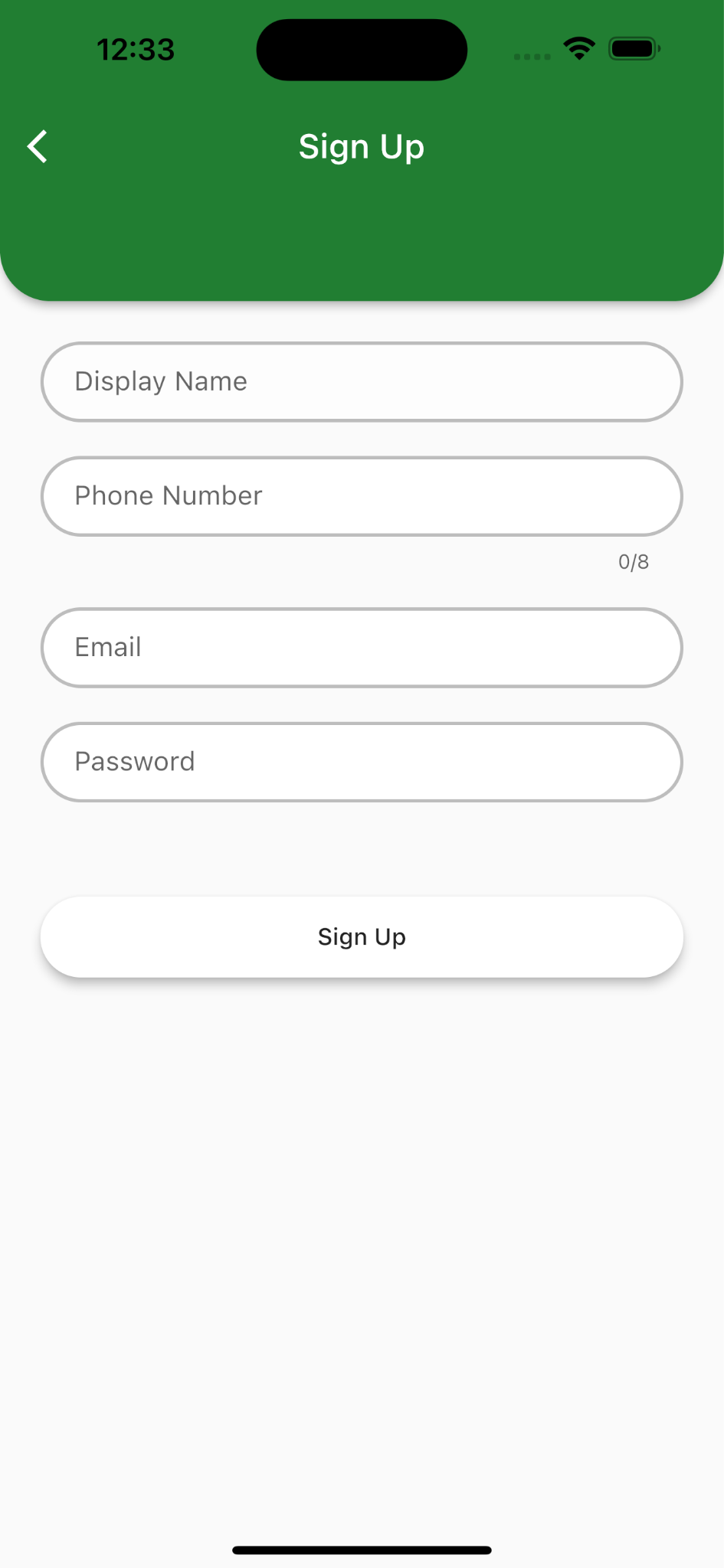
Screen 1: Login Page

The Login Page has three buttons, “Login”, “ Sign Up” and “Forgot Password”. We expect the normal flow of events in our application for users to enter their relevant data into the “Enter your email” and “Enter your password” input fields, and press the “Login” button to be redirected to our main landing page. Alternatively, the user will be redirected to the Register Screen (Screen 2), if the user were a new user. The user interface layout is displayed below.

## 

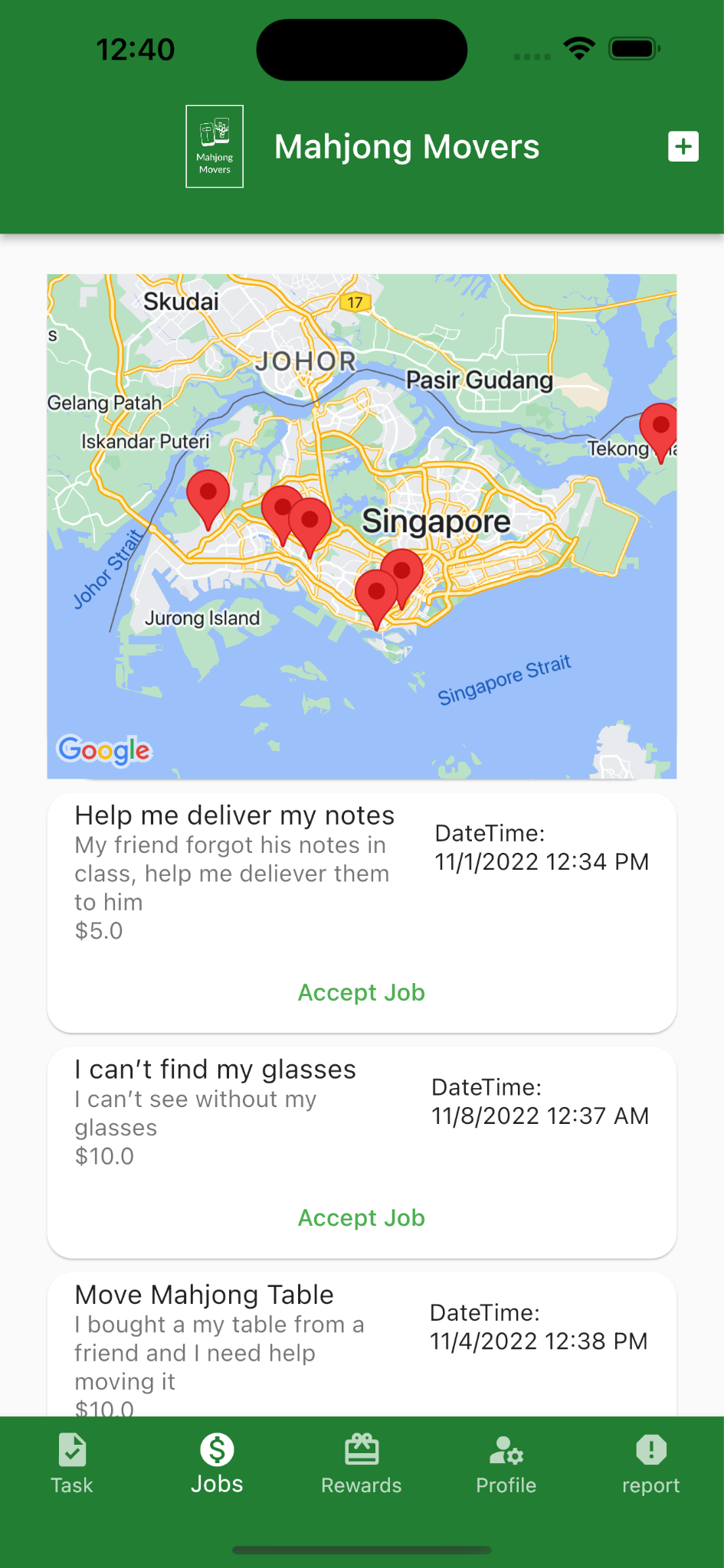
Screen 2: Registration

The Registration Screen has input fields for Display Name, Phone Number, Email, and Password. The user is required to enter this information in order for our application to register the user into our database. After the user inputs the valid and required data, the user can click on the “Sign Up” button. After successful verification, the application will redirect the user to our main landing page (Screen 3) The user interface layout is displayed below.



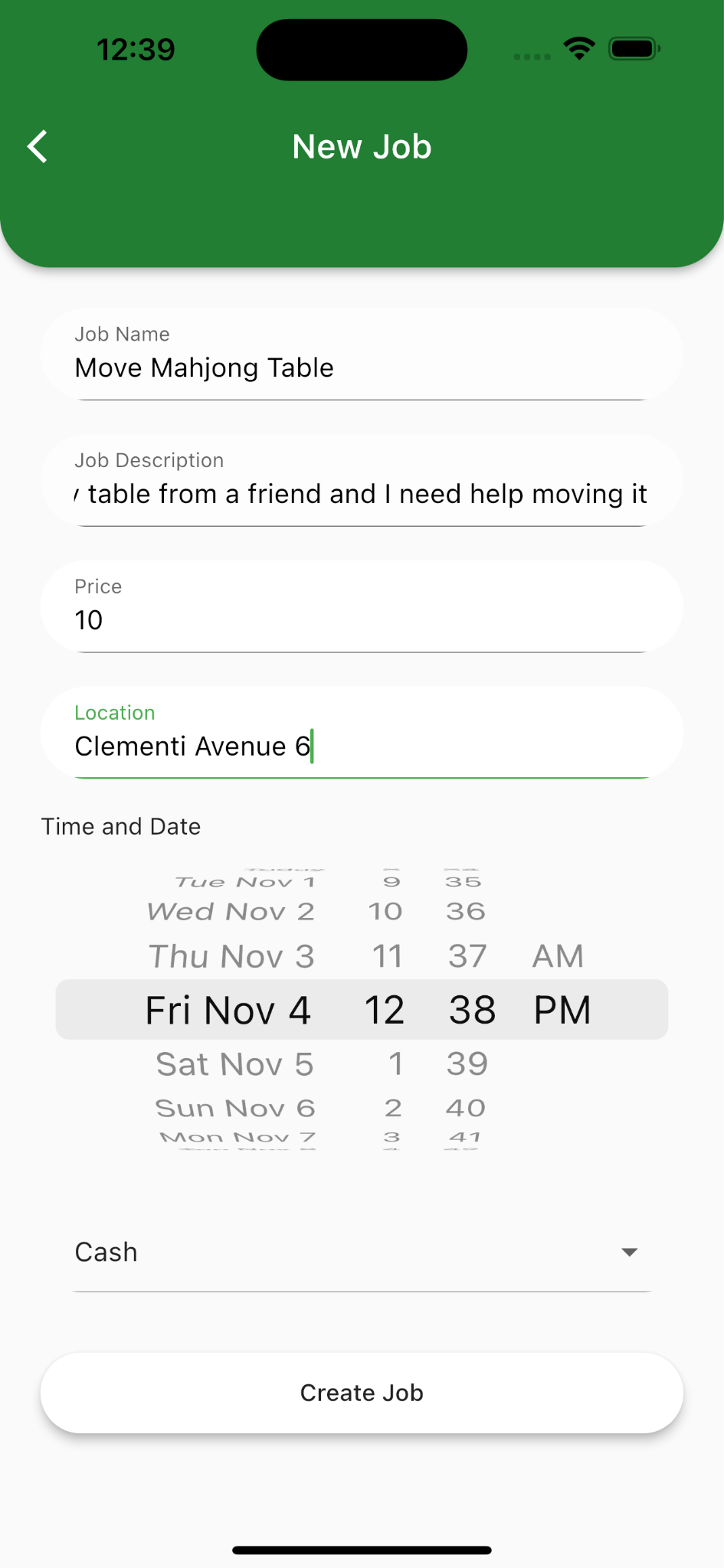
Screen 3: Main Landing Page

The Main Landing Page has interactive buttons in the navigation bar such as “Task”, “Rewards”, “Jobs”, “Profile” and “Report”. The “Jobs” page is an alias for the Main Landing Page and if the user were to press the “Jobs” button, while on the Main Landing Page, the Main Landing Page would simply refresh the available jobs. The other buttons on the navigation page, when pressed will redirect the user to the other pages. The user interface layout is shown below.



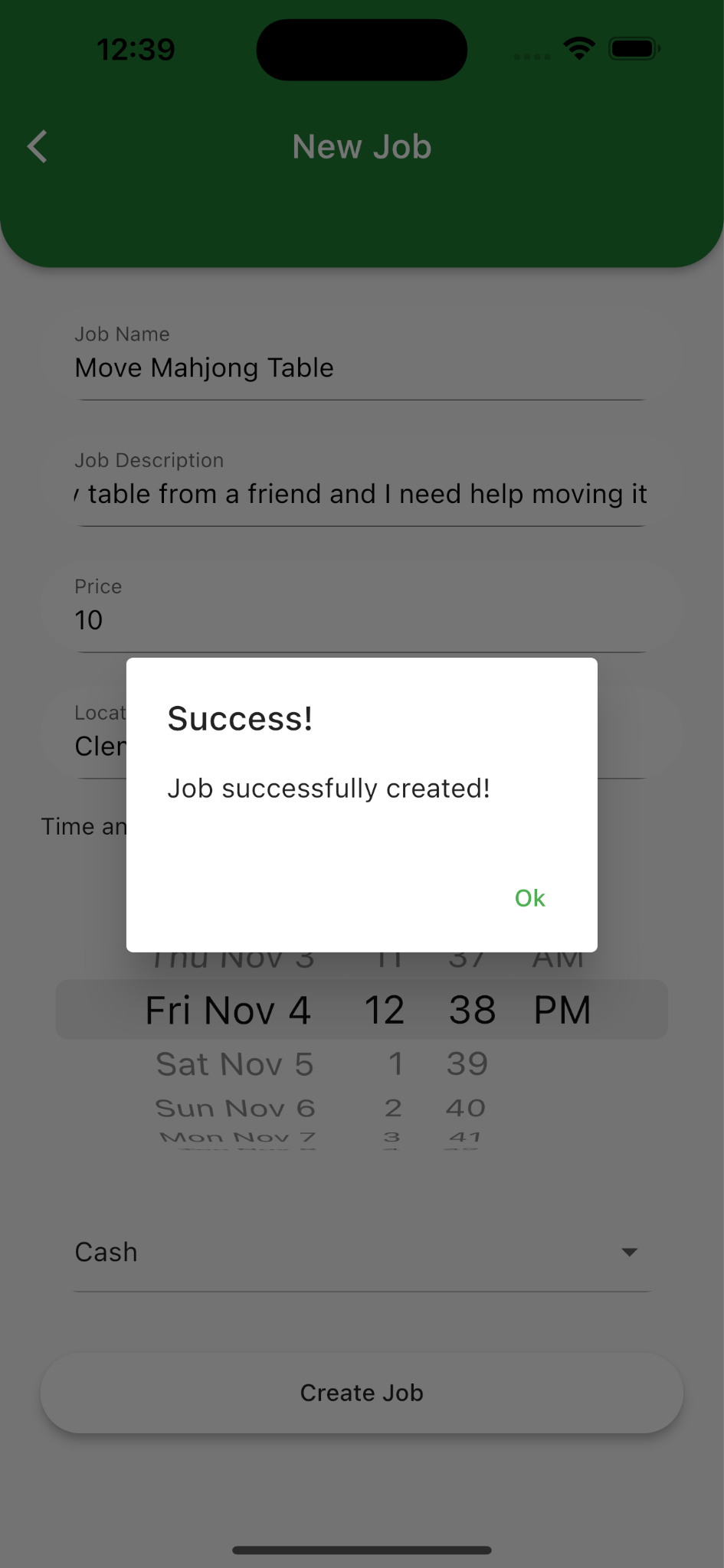
Screen 4:

The new job page is provided to the user to post jobs, the user has to enter and select corresponding job details such as the job name, job description, price, location, desired time, and payment method in order to successfully create a job. The user interface layout is shown below.



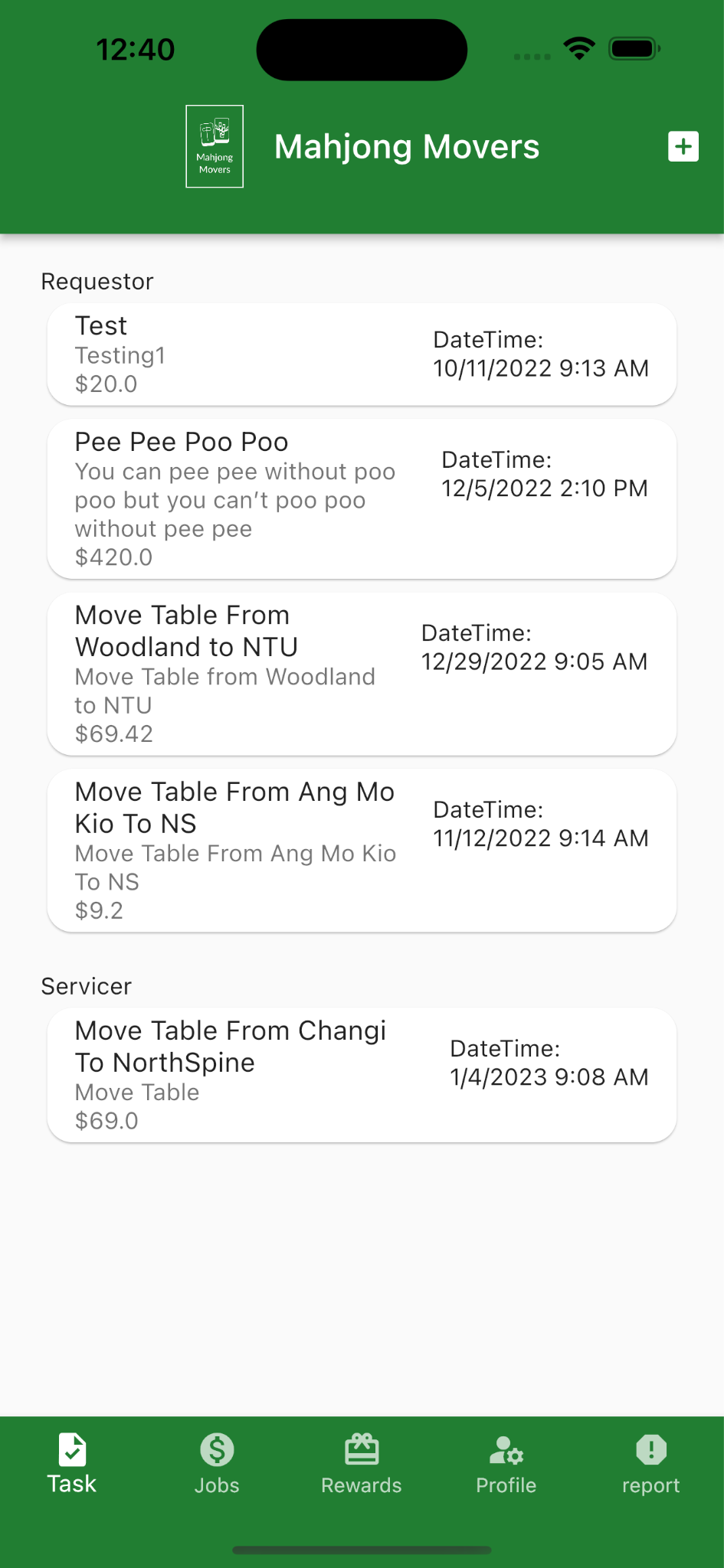
Screen 5:

After the user inputs all the details and presses the create job button, a notification window displaying “success” will pop up for the user to confirm that the creation of the job was successful as shown below. After the user’s confirmation by pressing the ok button, he will be redirected to our main landing page and register information about the job into our database. The user interface layout is shown below.



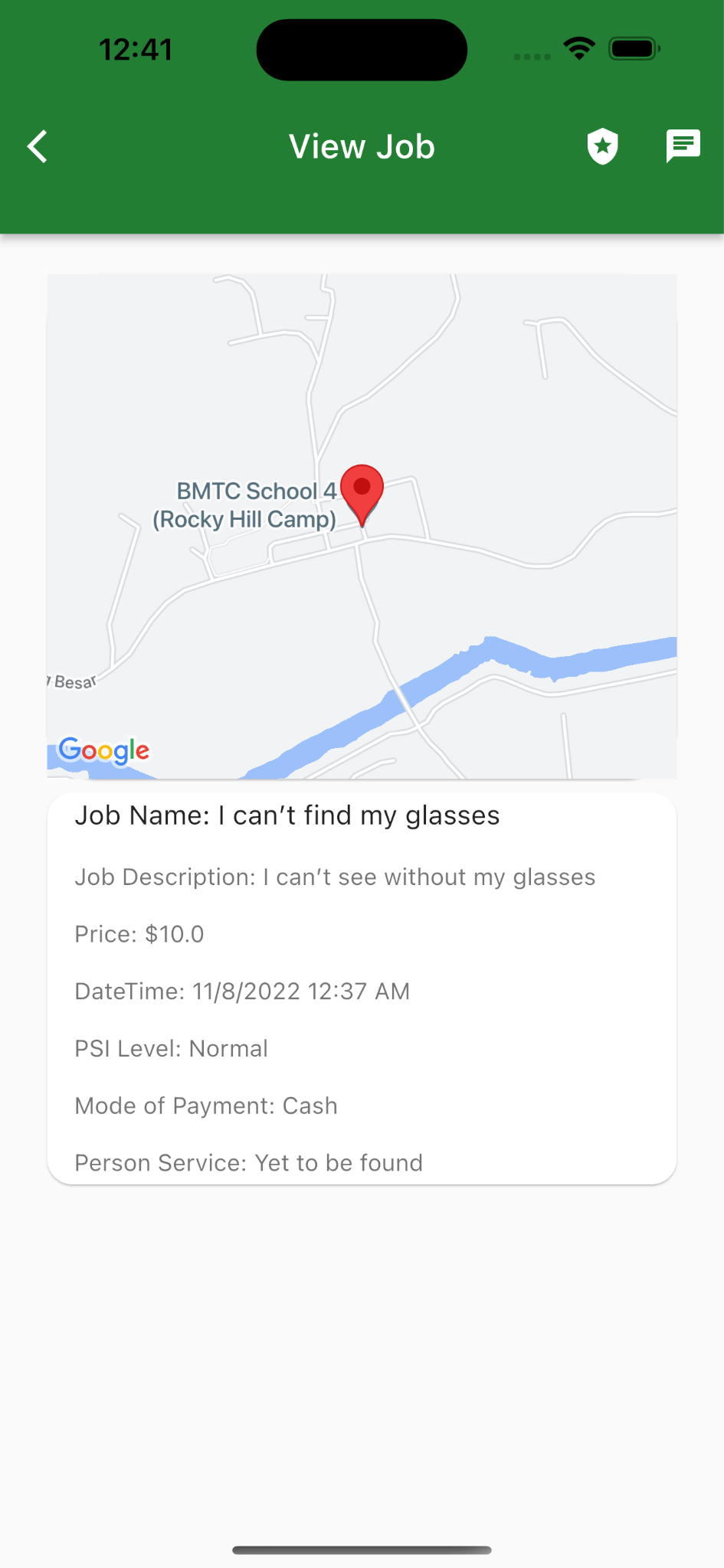
Screen 6:

This tasking page is provided so that users can keep track of the jobs that have been posted by the user (under the requestor section) or the jobs that have been accepted (under the sevicer section) thus far. More details and information will be displayed to the user after clicking on the corresponding job card. The user interface layout is shown below.



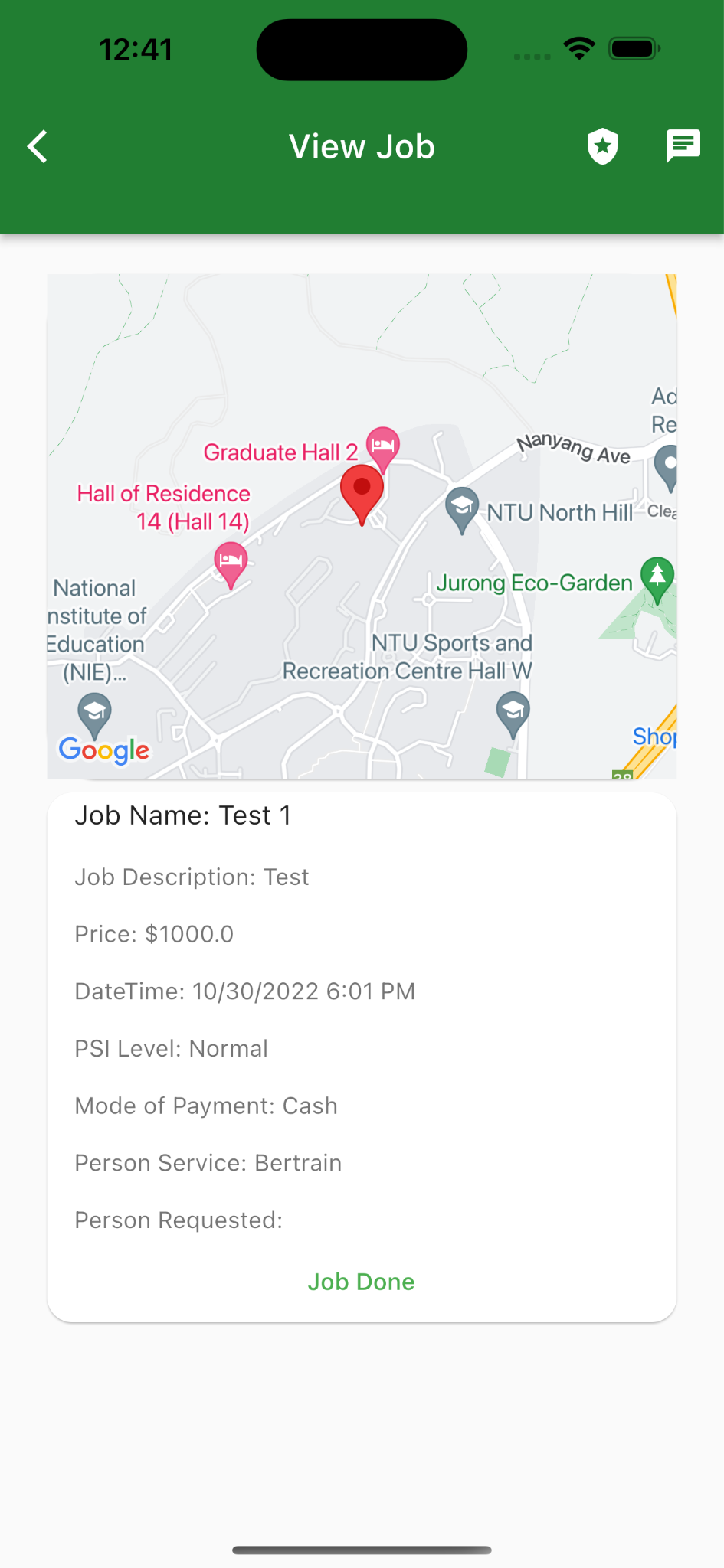
Screen 7:

The “View Job” page is shown when the user clicks on the job, under the requestor’s section. Through reverse geocoding to obtain the latitudinal and longitudinal positions by calling the Google Map API, the map displayed indicates the location of the job posted by the user. Furthermore, additional job details such as the job description, name, salary, DateTime, PSI level, mode of payment, name of the servicer, and name of the requester will also be shown on the page. The user interface layout is shown below.



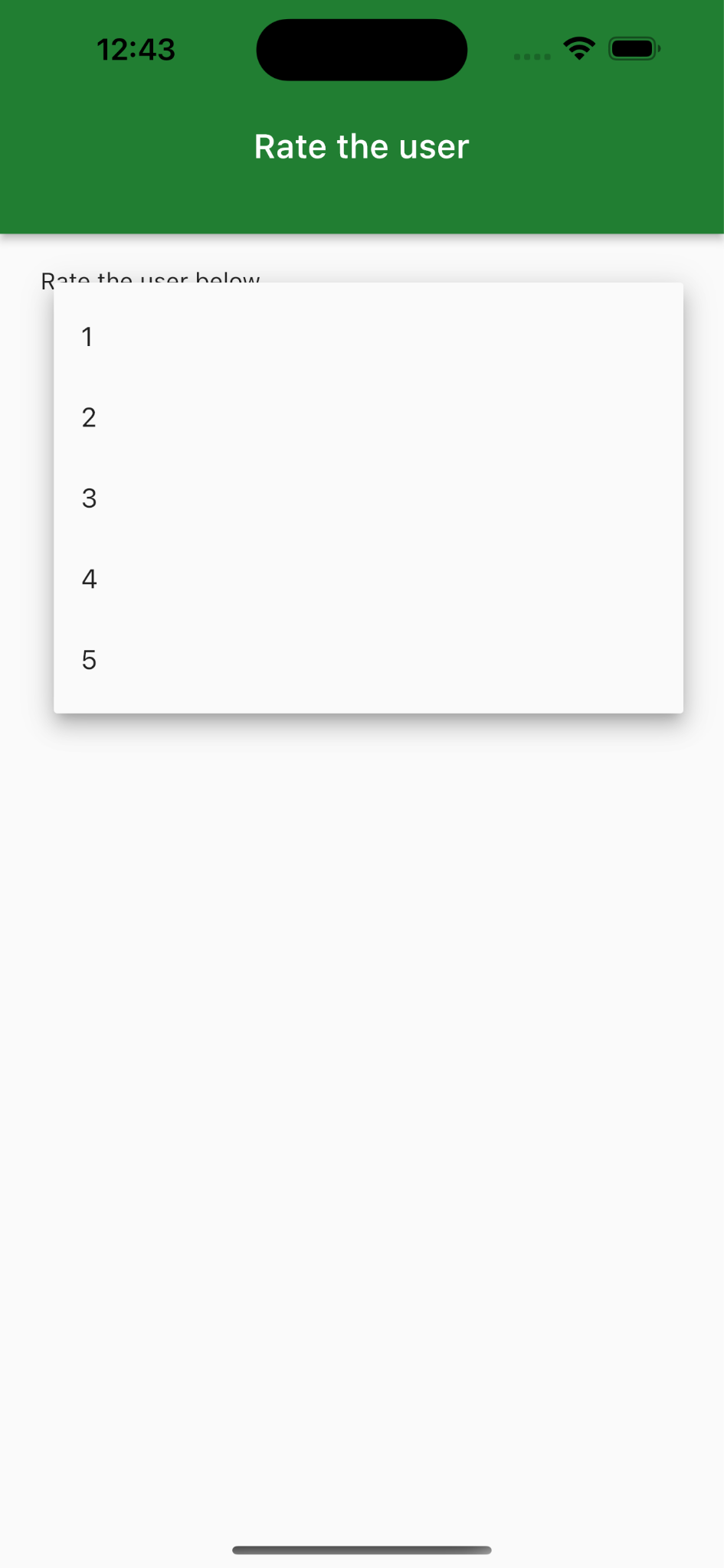
Screen 8: View job page

The view job page shows the map indicating the location of the job posted by the user. Other job details such as the job description, name, salary, DateTime, PSI level, mode of payment, name of the servicer, and name of the requester will also be shown on the page. The user interface layout is shown below.



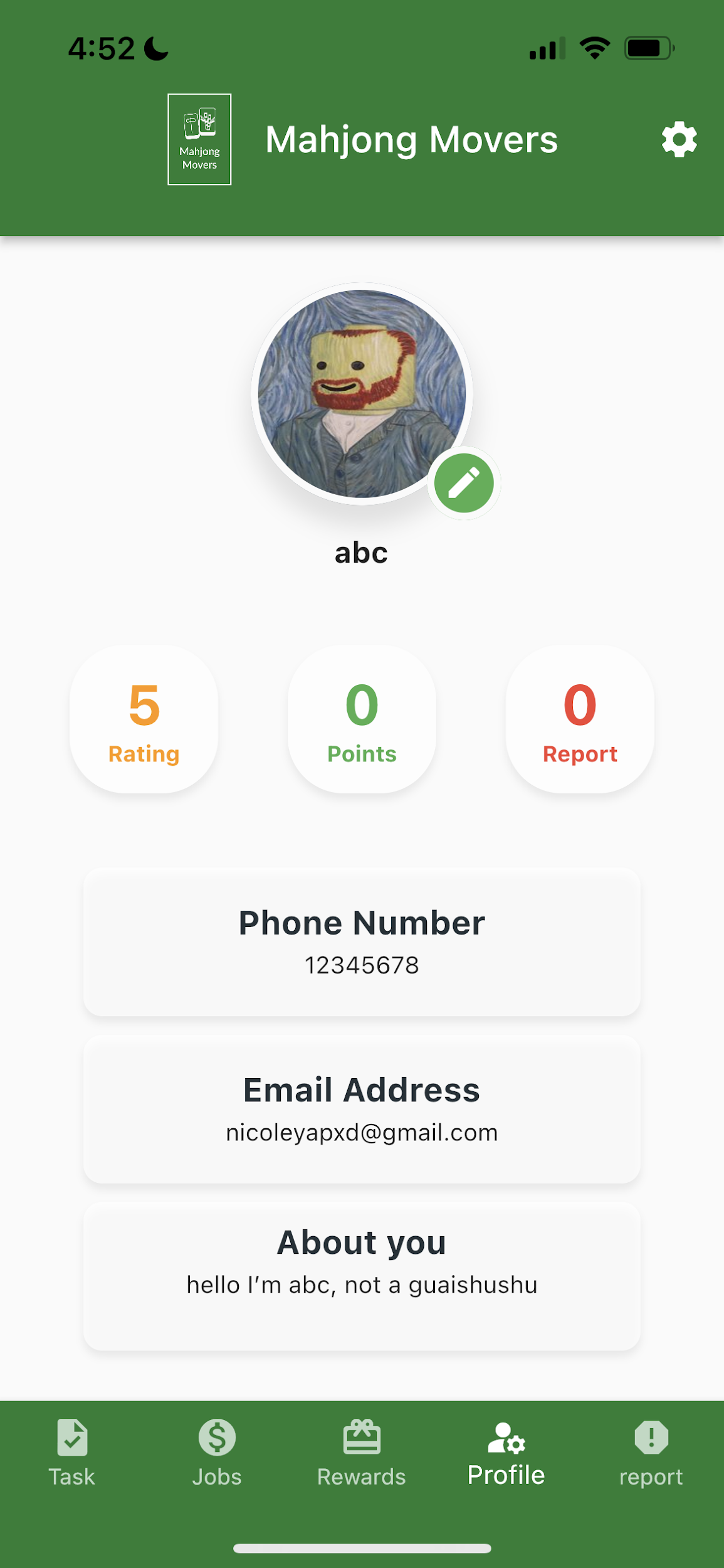
Screen 9:

The “rate the user” section is a feature provided to the user to rate the requester or the servicer after the job has been successfully completed. The rating ranges from 1-5 and the user rating will be provided after the user has submitted his rating. He will then be automatically redirected back to the main landing page. The user interface layout is shown below.



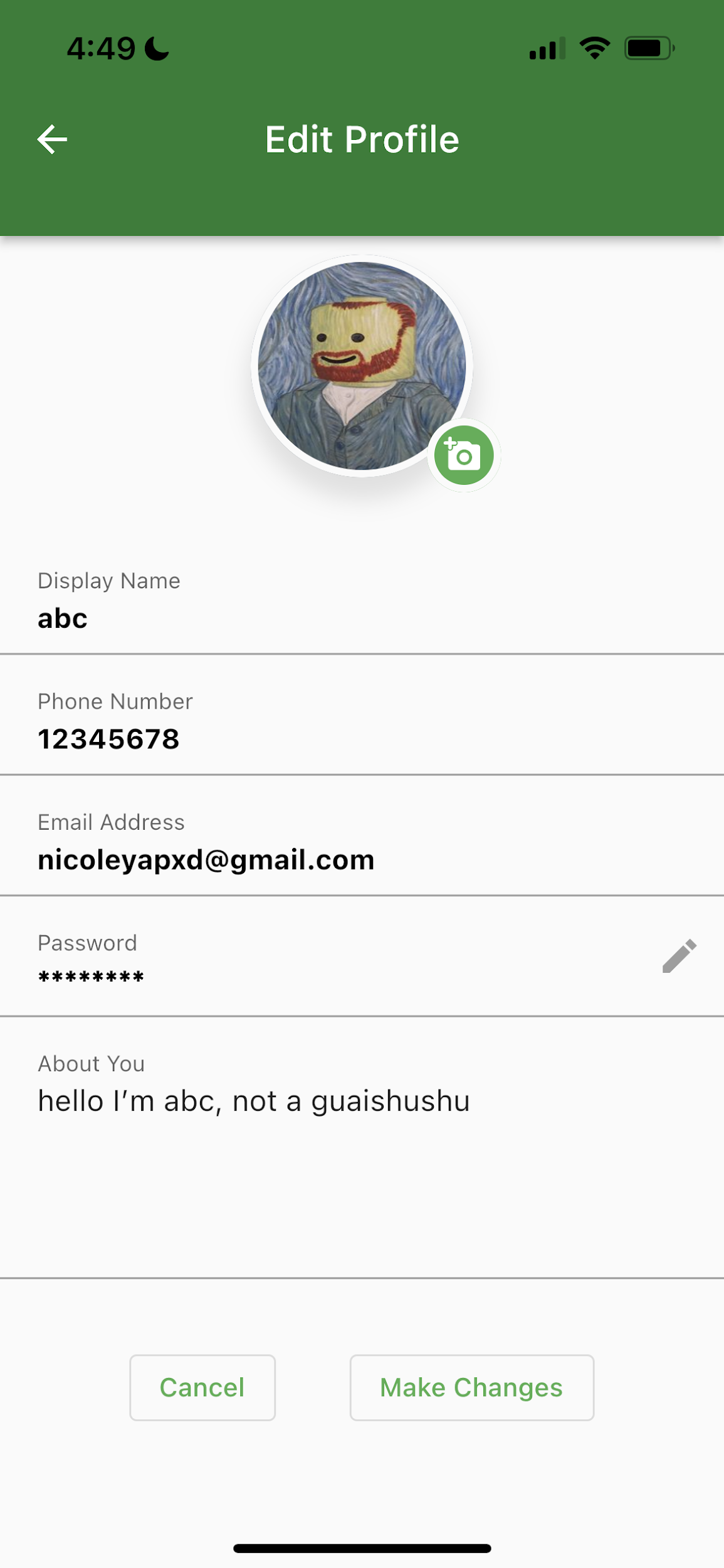
Screen 10: Profile page

The profile page will show the user information, overall rating, total points, and the number of reports submitted. The pencil icon on the bottom right corner of the profile picture allows the user to edit profile information. The user interface layout is shown below.



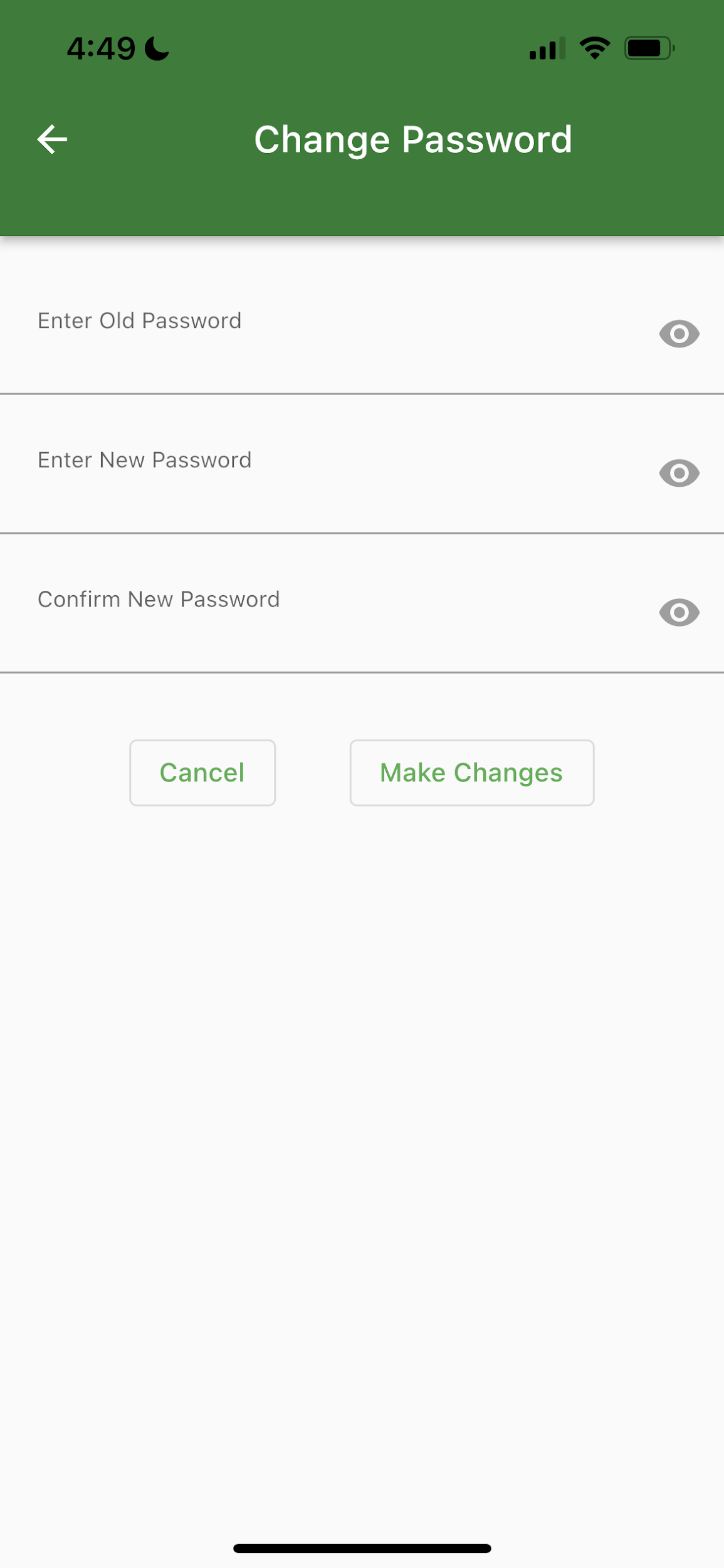
Screen 11: Edit profile page

Edit profile page will show the user’s current information. Users can change any information on the page by simply editing the fields. The ‘make changes’ button will update the user's details in our database while the ‘cancel’ button will clear all the inputs and redirect the user to the main landing page. The user interface layout is shown below.



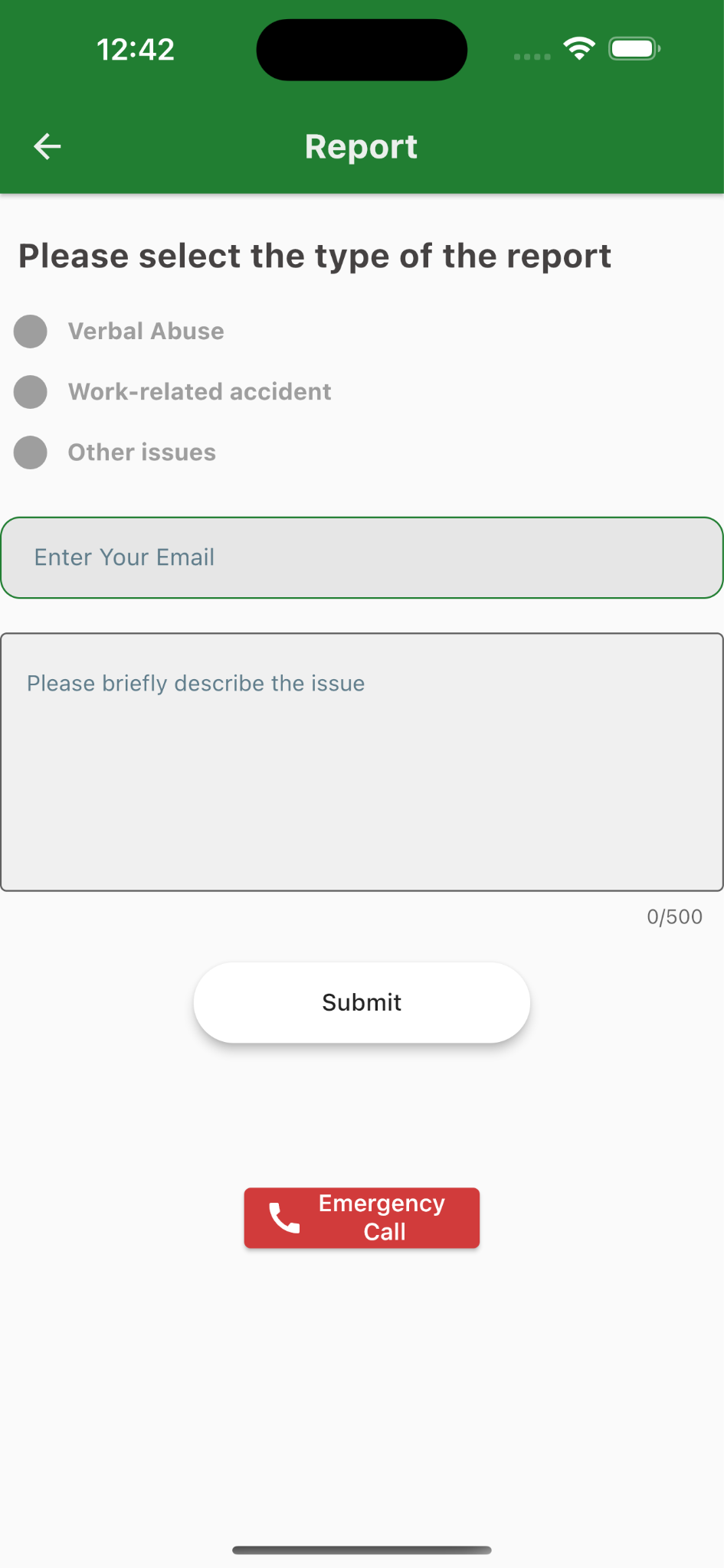
Screen 12: Change password page

This page will prompt users to enter the old password and the new password twice. The ‘make changes’ button will change the user's password in the database while the ‘cancel’ button will clear all the inputs and redirect the user to the main landing page. The user interface layout is shown below.



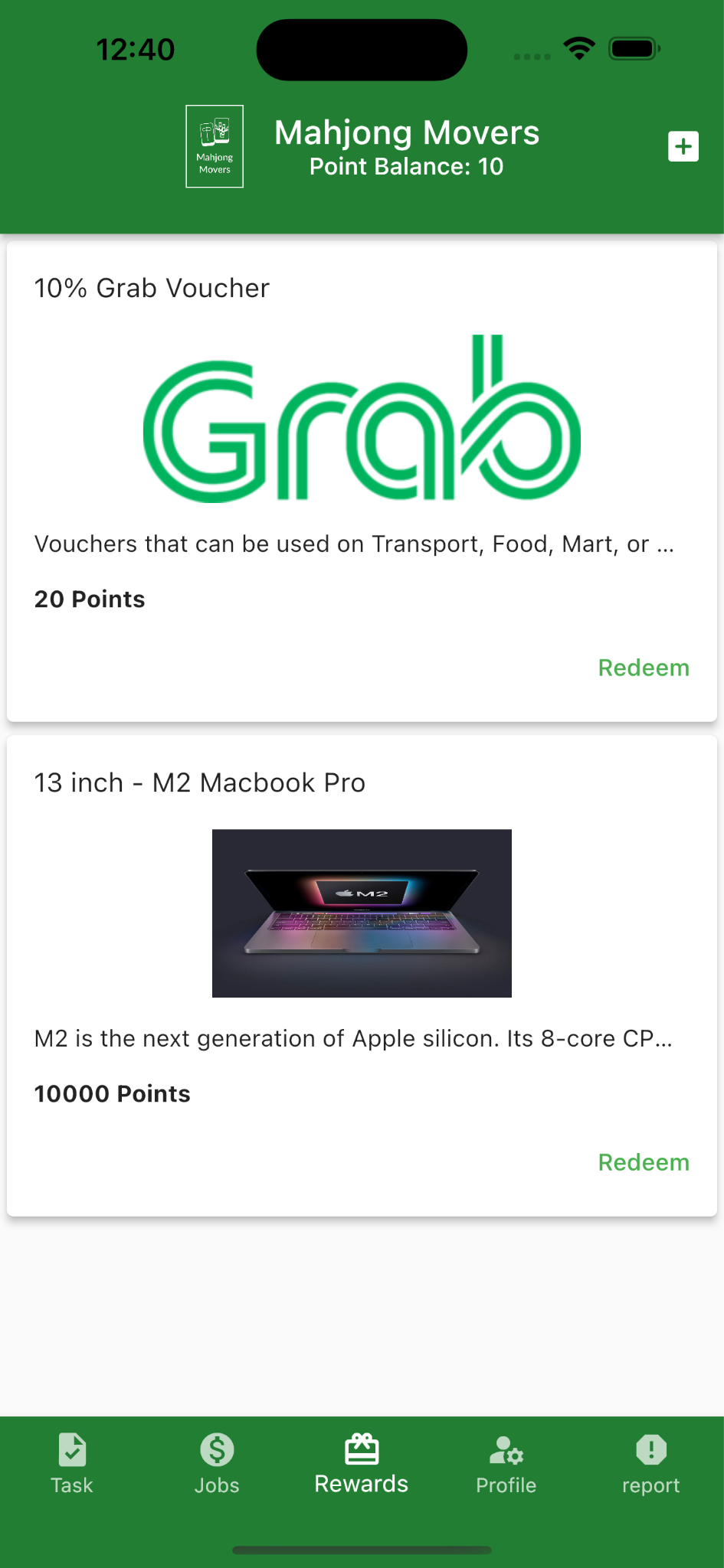
Screen 13: Report page

The “Report” page will prompt users to select the type of report and to input the email address as well as the report’s description. Users can click on the ‘Submit’ button to confirm once they have completed it. There is also an ‘Emergency call’ button for users to call the police immediately. The user interface layout is shown below.



Screen 14: Rewards Page

The rewards page shows the user's point balance and a list of rewards, to give an incentive for users to use our application. The rewards are shown by a title, picture, and description. There is a “Redeem” button for users to redeem the reward. The user interface layout is shown below.



## 

## Hardware Interfaces

Hardware interfaces depend on the iPhone’s multi-touch capacitive Touch Screen. The touch screen hardware has a grid array of touchpoints that relay touch information to the processor. This enables the device to support a wide array of gesture supports such as swiping and scrolling. This interface allows the user to perform most of the functions required in the application.

Some other interfaces that could assist in the application include, power button, volume control, and camera. The camera can enable the user to take a picture of the profile picture.

## 

## Software Interfaces

The application is built using Flutter and Firebase. Flutter is a library by Google that allows the user to code in DART programming language and generate the application for different operating systems. Firebase supports the backend for the application, which includes managing of user’s credentials, job status, etc. Our application also relies on external APIs.

For Flutter, we have the following third-party dependencies, together with their version number.

|  |  |  |
| --- | --- | --- |
| **Library** | **Version** | **Description** |
| cupertino\_icons | 1.0.2 | iOS Specific Icons |
| firebase\_core | 1.22.0 | Library to interact with Firebase |
| firebase\_auth | 3.8.0 | Authenticate the user with Firebase |
| cloud\_firestore | 3.4.7 | Interact with the Firestore |
| google\_maps\_flutter | 2.2.0 | Google Maps API and Display the Google Maps |
| flutter\_launcher\_icons | 0.10.0 | Set the launch icons |
| http | 0.13.5 | Used to send HTTP Request and Receive the reply |
| url\_launcher | 6.1.6 | Library to launch other applications or website links |
| firebase\_storage | 10.3.10 | Used to interact with the Firebase Storage |
| image\_picker | 0.8.5+3 | Allows the applications to choose an image from the device |
| flutter\_phone\_direct\_caller | 2.1.1 | Enables the calling function |
| file\_picker | 5.2.0+1 | Allows the applications to choose a file |
| flutter\_chat\_ui | 1.6.5 | Basic Chat UI |

For Firebase, we have used the following services:

|  |  |
| --- | --- |
| **Firebase Services** | **Description** |
| Firebase Auth | Used for authenticating the user, together with common features such as setting password, changing emails, providing UID |
| Firebase Firestore | Used for storing profile pictures of the users |
| Firebase Storage | NoSQL DB that allows the users to store their current transaction, chat history, current points, etc. |

For the external APIs, we have the following:

|  |  |
| --- | --- |
| **API Name** | **Description** |
| Google Maps API | Used the Google Maps API to display the locations of the job description, reverse geolocation to get the longitude and latitude of the location. |
| Data Gov | Used the Singapore Government API to get the PSI reading. When the readings get bad, the user can wear the appropriate protective gear to do the job. |

## 

# System Requirements

## Account

* + 1. Users must be able to register for a new account.
       1. System must create a new account for the user once all requirements are fulfilled.
          1. System must prompt the user to change the username if it is taken.
          2. System must prompt the user to change the passwords if the password didn’t adhere to requirements: at least 1 special character, letters in mixed cases, at least 8 characters.
          3. System must prompt the user to change the phone number if there is a phone number linked to an existing account.
          4. System must prompt the user to change the email address if there is an email address linked to an existing account.
       2. System must send a confirmation email to the user when a new account is successfully created.
    2. Users must be able to login to the account with username and password.
       1. After the user submits details, the system must validate the user’s username and password.
       2. System shall show the main landing page.
       3. Users must be able to view their profile after clicking the profile tab.
       4. Users must be able to modify their profile if they want
          1. Any changes made by users must fulfilled the requirements according to 1.1.1
          2. System must be able to update the profile picture after edit the profile picture
          3. System must be able to update the email after the email has been changed by the user.
          4. System must be able to update the bio after the bio has been changed by the user.
          5. System must be able to update the phone number after the phone number has been changed by the user.
    3. Users must be able to reset the password if they had forgotten their passwords.
       1. System must send an email to users to reset the password.
       2. System must prompt the user to change the password if the new password didn’t adhere to requirements: at least 1 special character, letters in mixed cases, at least 8 characters.
       3. System must prompt the user to change the password if the new password is used before.

## Job

* + 1. Requesters must be able to create job requests.
       1. System must ensure the job request has a job name.
       2. System must ensure the job request has a job description.
       3. System must ensure the job request has a location specified.
       4. System must ensure the job request has date and time specified.
       5. System must ensure the job request has a salary specified.
       6. System must ensure the requester specifies the payment method.
       7. System must add the job request to the existing job listing after the job request is successfully created.
    2. Requesters must be able to edit job requests if the job request hasn’t been accepted.
       1. System must update the changes to the job.
    3. Requesters must be able to delete job requests
       1. System must remove the job request from the job listing.
       2. System must penalise requesters after the servicers have accepted for 3 minutes
    4. Servicers must be able to filter through the available jobs.
       1. System shall allow servicers to filter by keywords, salary, and location.
    5. Servicers must be able to accept job requests.
       1. System must create a job contract with all details on it.
       2. System must add the job to the servicer’s “Job List”.
          1. System must set the job status to active.
       3. System must remove the job request from the job listing.
       4. System must ensure both servicers and requesters are able to contact each other after the job contract is created.
       5. System must allow both servicers and requesters to contact through the app.
    6. Servicers must be able to retract services
       1. System must add the job request to the job listing.
       2. System must penalise servicers after the servicers have accepted the job for 3 minutes.
    7. Requesters must be able to report if any emergency occurred.
       1. System must immediately alert customer service.
    8. Requesters must be able to end the job after it is completed.
       1. Servicers must be able to check the “I have received payment”.
       2. System must change the job status to history.
       3. Requesters must be able to rate and review the servicers.
          1. System must update any comments and ratings in the servicer’s profile.
       4. Servicers must be able to rate and review the requesters.
          1. System must update any comments and ratings in the requester’s profile.

# 

# Other Non-Functional Requirements

## Performance

* + 1. The application shall take within 3 seconds to load in
    2. The sign up and sign in page shall take within 1 second to load.
    3. The validation email for sign up shall take within 1 min to arrive.
    4. Resetting password shall take 1 second to update in the system.
    5. Each function module shall take within 1 seconds to load in.
    6. System must add the new job request to the job listing in 1 minute.
    7. Details of job shall take within 2 seconds to load.
    8. The application shall take within 3 seconds to exit.
    9. All reminding messages shall take within 0.5 second to load in.
    10. Accepting and posting jobs shall take within 1 second to be updated in the system.
    11. Map must be displayed within 1 second.
    12. Information about jobs must be displayed within 1 second.

## Safety Requirements

* + 1. Users shall add friends via this application and chat.
    2. Team shall maintain the system
    3. Team shall test and debug the application beforehand.

## Security Requirements

* + 1. If the user enters in the wrong password more than 3 times, the account will be locked
    2. Users must log in their account to post or accept jobs.
    3. The system shall handle exceptions.
       1. Reminding message shall display when the user enters two different passwords when setting or resetting the password.
       2. Reminding message shall display when the user inputs the wrong verification code.
    4. The application must not leak user information.
       1. User information must not be seen or used by others.
       2. User information must be deleted if the user account is deleted.

## Data Dictionary

|  |  |
| --- | --- |
| Term | Definition |
| User | A person using the application. |
| New User | A person who has yet to register on the application. They can only view the log in page of the application. |
| Existing User | A person who has a registered account and is using the services provided by the application. |
| Customer Service | The department of the company that deals with complaints, reports, and concerns of the users of the application. |
| Account | The profile of the user in our system which contains details (Name, Username, Email, Phone Number, Rating) associated with them. |
| Display Name | A name that belongs to the user which is shown to other user of our application |
| Password | A string of characters (letters, numbers, special symbols) that allow the user to access their account |
| Job | An instance of an ad hoc request created by a requester which details such as location, time, description, number of people needed and salary.  Jobs are completed by the servicer. |
| Contract | An agreement which is created when a requester has accepted the help from a Servicer(s). |
| Requester | A user who creates a job request in the system. When the user is in the “Post Job” page, he/ she is in “Requester Mode”/ is a “Requester”. |
| Servicer | A user who does the job created by a requester. When the user is in the “Job List” page, he/ she is in “Servicer Mode”/ is a “Servicer”. |
| Rating | A numerical way to indicate how well a Servicer or Requester worked when the job has been completed. They will be rated on a 1 - 5 system (1 = worst, 5 = best) |
| Points | Numeric value assigned to every account. It is increased when a contract has been fulfilled. It is decreased when a penalty has been imposed on them. For a servicer, the amount of points they have will affect how likely they are to be seen by a requester. The points act as the determining factor of which servicer to be assigned to a job in the case of competition. |
| Report Count | A variable shown on the user’s profile that indicates the number of reports someone had made on the user. |
| Reporter | A user who is involved in a contract that reports the counter party involved in the same contract. |
| Review | A short description which allows both users to talk about their experience doing the job. |
| Main Landing Page | The first page that the user sees after he logs into the application. It displays the main information that the user needs to know such as a map which displays all the requesters looking for a servicer at the current moment. |
| Job Status | Created: Job has just been created by Requester  Assigned: Job has an assigned Servicer  Pending Confirmation: Servicer confirms that he/ she has finished job, but pending confirmation from Requester.  Confirmed: Requester agrees and confirms that job has been finished. This also implies that the job has been completed.  Deleting: Requester deletes job, but job is not yet deleted in database. This is required to ensure a checking of a state to send a notification to the Servicer.  Retracting: Servicer retracts from service. Notification will be sent to the Requester in this state.    Deleted: Job completely deleted from database. |

## 

# Use Case Description

Key:

Frequency 1-5, 1 = Low 5 = Very Frequent

Priority 1-5, 1 = Highest Priority 5 = Lowest Priority

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **USER\_UC\_1** | | |
| Use Case Name: | **Create Account** | | |
| Created By: | **Shen Hwei (Nicole)** | Last Updated By: | **Shen Hwei (Nicole)** |
| Date Created: | **02.09.2022** | Date Last Updated: | **13.09.2022** |

|  |  |
| --- | --- |
| Actor: | **User** |
| Description: | **Create Account allows new users to create an account in our system database. To create an account, users must input Display Name, Email, Phone Number, Password.** |
| Preconditions: | 1. **Phone number must not link to an existing account.** 2. **Email must not link to an existing account.** 3. **Passwords must adhere to requirements: at least 1 special character, letters in mixed cases, alphanumeric and has at least 8 characters.** |
| Postconditions: | 1. **System database holds new user’s account information.** |
| Priority: | **1** |
| Frequency of Use: | **5** |
| Flow of Events: | 1. **System displays empty form fields for Display Name, Email, Phone Number and Password.** 2. **User inputs required information: Display Name, Email, Phone Number, Password.** 3. **User submits form.** 4. **System validates the information.** 5. **System creates the account in database.** 6. **System updates account information in database.** 7. **System sends confirmation link to user’s email account.** 8. **User clicks onto confirmation link.** 9. **System displays main landing page.** |
| Alternative Flows: | **AF-S4-a. User account already exists: Phone number/ email taken.**   1. **System displays “User account already exists”** 2. **System prompts user to create account or log in.** 3. **If create account, then continue from main flow step 1.** 4. **If log in, then system directs to log in page to continue from Login (USER\_UC\_2)**   **AF-S4-b. Password does not meet requirements.**   1. **System displays “Please input a password of at least 1 special character, mixed case and at least 8 characters long” message.** 2. **Continue from main flow step 1.**   **AF-S4-c. Phone number is not a valid number.**   1. **System displays prompt message to user.** |
| Exceptions: |  |
| Includes: | **Update Account Information (S4)** |
| Extends: |  |
| Special Requirements: | 1. User must include phone number and email in the creation of account. 2. User must verify account through email confirmation link to become a User of the application. |
| Assumptions: |  |
| Notes and Issues: | [Updates 13.09.2022 - Shen Hwei]   * Remove username from flow. * Add confirmation link into normal flow. * Add special requirements. * Add Assumptions to elaborate on Special Requirements terms.   [Updates 26.10.2022 - Shen Hwei]   * Updated Phone number is not a valid number |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **USER\_UC\_2** | | |
| Use Case Name: | **Login** | | |
| Created By: | **Shen Hwei (Nicole)** | Last Updated By: | **Shen Hwei (Nicole)** |
| Date Created: | **02.09.2022** | Date Last Updated: | **13.09.2022** |

|  |  |
| --- | --- |
| Actor: | **User** |
| Description: | **Login allows users to enter the app to use features catered for registered users of the app. To login to the system, the user must input a unique email and password.** |
| Preconditions: | **User must have an existing account in the application database** |
| Postconditions: | **System displays the main landing page** |
| Priority: |  |
| Frequency of Use: | **5** |
| Flow of Events: | 1. **System displays login page with empty fields.** 2. **User enters email and password on login form.** 3. **User submits details.** 4. **System validates the user’s email and password.** 5. **System shows the main landing page.** |
| Alternative Flows: | **AF-S2-a. Invalid Email**   1. **System cannot find the email in the database.** 2. **System displays an error message.**   **AF-S2-b. Invalid Password**   1. **Password does not match for unique email.** 2. **System displays an error message.** 3. **System prompts user to input email and password.** 4. **Use case resumes at main flow step 1.**   **AF-S2-c. Missing Fields**   1. **System displays an error message.** 2. **System prompts user to input into missing fields.** 3. **Use case resumes at main flow step 2.** |
| Exceptions: |  |
| Includes: | **Validate Account, Verify Account** |
| Extends: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: | [Updates 13.09.2022 - Shen Hwei]   * Update AF-S2-c. To “Missing Fields” instead of “Missing Username/ Password” * Remove Username and Add Email instead.   [Updates 26.10.2022 - Shen Hwei]   * Remove AF to the number of sign-ins allowed, due to limitation of firebase. |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **USER\_UC\_3** | | |
| Use Case Name: | **Edit Account Information (User)** | | |
| Created By: | **Shen Hwei (Nicole)** | Last Updated By: | **Shen Hwei (Nicole)** |
| Date Created: | **02.09.2022** | Date Last Updated: | **03.10.2022** |

|  |  |
| --- | --- |
| Actor: | **Existing User** |
| Description: | **Edit Account Information allows user to edit account information such as Display Name, Email, Phone Number, Profile Picture, Password.** |
| Preconditions: | **User must have an existing account in the application database** |
| Postconditions: | **System displays the profile page with updated details.** |
| Priority: |  |
| Frequency of Use: | **5** |
| Flow of Events: | 1. **User clicks on “edit account” button on the interface.** 2. **User selects a field to update.** 3. **User inputs new updates on selected fields.** 4. **System receives new changes.** 5. **System updates database. (Update Account Information)** 6. **System updates user interface to reflect the changes in database.** |
| Alternative Flows: | **AF-S1-a. Change Password**   1. **User inputs Old Password** 2. **User inputs New Password** 3. **User inputs Confirm New Password** 4. **System validates input fields** 5. **If validation successful, system verifies user and old password.** 6. **If verification successful, system updates user’s password to new password.** |
| Exceptions: |  |
| Includes: |  |
| Extends: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: | [Updates 14/09/2022]   * Change description, further discussion about description needs to be made.   [Updates 18/09/2022]   * Changed to split into two use cases: “Edit Account Information” (for user initiated) and “Update Account Information” (system’s job) * Updated Flow. |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **USER\_UC\_4** | | |
| Use Case Name: | **View Listing** | | |
| Created By: | **Shen Hwei** | Last Updated By: | **05.09.2022** |
| Date Created: | **05.09.2022** | Date Last Updated: | **05.09.2022** |

|  |  |
| --- | --- |
| Actor: | **User** |
| Description: | **View Listing allows only existing users, to view the ad-hoc jobs available.** |
| Preconditions: | **User must be on main landing page.** |
| Postconditions: |  |
| Priority: | **1** |
| Frequency of Use: | **5** |
| Flow of Events: | 1. **System displays the main landing page of job listing to user.** 2. **User interacts with the main landing page.** 3. **If user chooses to accept jobs, use case continues from Accept Job (S1).** 4. **If user chooses to request jobs, use case continues from Post Job (S1).** |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: |  |
| Extends: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: | [Updates 13.09.2022 - Shen Hwei]   * **Remove condition of “any users, regardless of whether logged in or not can view listing”. Only existing users can view listing.** * **Add AF-21-b** |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **USER\_UC\_5** | | |
| Use Case Name: | **Update Job Status** | | |
| Created By: | **Koh Jia Wei** | Last Updated By: | **Shen Hwei** |
| Date Created: | **05.09.2022** | Date Last Updated: | **26.10.2022** |

|  |  |
| --- | --- |
| Actor: | **User** |
| Description: | **At different stages of a contract between a Servicer and a Requester, the state of the job is different.**  **Refer to data dictionary on Job Status for a more detailed explanation of the different statuses.**  **Update Job Status is an explicit use case for the Requester and Servicer to manually input the state of the jobs:**  **After servicer is done with the job, he/she can update from Assigned -> Pending Confirmation.**  **After the requester agrees that the job is done by servicer, he/she updates from the state**  **Pending Confirmation -> Confirmed.** |
| Preconditions: | 1. **There is an existing contract between the requester and servicer.** |
| Postconditions: | **The state of the job will be changed.** |
| Priority: | **1** |
| Frequency of Use: | **5** |
| Flow of Events: | 1. **The servicer updates the status of the job to “Servicing”, and this is reflected in the system for all to see.** 2. **The job is successfully done by the servicer.** 3. **The servicer updates that the job is done, the status of job is now “Awaiting confirmation”, and this is reflected in the system.** 4. **The requester is satisfied with the job and the status is now updated to “Confirmed”** 5. **Once confirmed, the servicer and the requester can rate each other.** 6. **If user chooses to rate, the system directs the user to the rating page. Use case continues from Rate User.** |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: |  |
| Extends: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **USER\_UC\_5** | | |
| Use Case Name: | **Rate User** | | |
| Created By: | **Shen Hwei** | Last Updated By: | **05.09.2022** |
| Date Created: | **05.09.2022** | Date Last Updated: | **13.09.2022** |

|  |  |
| --- | --- |
| Actor: | **User** |
| Description: | **Rate User allows user from either end (Requester/ Servicer) to rate the opposite party after job has been “Completed”.** |
| Preconditions: | 1. **Job/ Service must be completed and accepted by both the Requester and the Servicer.**   **Here, accepted means that**   1. **The Servicer has updated the job status from assigned -> awaiting confirmation.** 2. **The Requester has updated the job status from awaiting confirmation -> confirm.** |
| Postconditions: | 1. **Average Rating is updated in the account database.** 2. **System displays average rating on user profile.** |
| Priority: |  |
| Frequency of Use: |  |
| Flow of Events: | 1. **System displays rate user page.** 2. **User performs rating.** 3. **System records the rating for the user for the specified Job/Service.** 4. **System calculates the average rating for user.** 5. **System updates account information.** 6. **Both User gets points based on the rate score.** |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: | **Update Account Information (S5)** |
| Extends: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: | **[Updates 13.09.2022 - Shen Hwei]**   * **Refine Flow** |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **USER\_UC\_6** | | |
| Use Case Name: | **Report Emergency** | | |
| Created By: | **Shen Hwei** | Last Updated By: | **05.09.2022** |
| Date Created: | **05.09.2022** | Date Last Updated: | **05.09.2022** |

|  |  |
| --- | --- |
| Actor: | **User** |
| Description: | **Report Emergency allows users who are participants of a contract to contact Police directly through the application.** |
| Preconditions: | 1. **Reporters must be users who are participants of a contract.** |
| Postconditions: | 1. **Case escalates to local police.** |
| Priority: | **3** |
| Frequency of Use: | **1** |
| Flow of Events: | 1. **User clicks onto a “Report Emergency” interface on the application.** 2. **System initiates a call to the police authority immediately.** |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: | **Report to Customer Service** |
| Extends: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: | [Updates 13.09.2022 - Shen Hwei]   * **Polish use case.**   **[Updates 18.09.2022 - Shen Hwei]**   * **Updated Includes** |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **USER\_UC\_7** | | |
| Use Case Name: | **Report to Customer Service** | | |
| Created By: | **Shen Hwei** | Last Updated By: | **Bert** |
| Date Created: | **05.09.2022** | Date Last Updated: | **19.09.2022** |

|  |  |
| --- | --- |
| Actor: | **User(Reporter), User(Reported), Customer Service (Secondary)** |
| Description: | **Report To Customer Service allows users (Reporter) to contact and report a user (Reported) to customer service if the reported is not behaving in a satisfactory way.** |
| Preconditions: |  |
| Postconditions: | 1. **System updates the report to the database.** |
| Priority: | **3** |
| Frequency of Use: | **2** |
| Flow of Events: | 1. **Reporter clicks onto a “Report” UI.** 2. **System directs user to a report form.** 3. **Reporter fills out form and submits** 4. **System shows a submission completed message.** 5. **System updates the database.** |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: |  |
| Extends: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: | **[Updates 13.09.2022 - Shen Hwei]**   * **Polish use case.**   **[Update 19.09.2022 - Bertranne Lee]**   * **Reduced ambiguity and clarified use case**   **[Update 26.10.2022 - Shen Hwei]**   * **Updated the flow.** |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **REQUESTER\_UC\_1** | | |
| Use Case Name: | **Manage Job** | | |
| Created By: | **Shen Hwei** | Last Updated By: | **Shen Hwei** |
| Date Created: | **05.09.2022** | Date Last Updated: | **14.09.2022** |

|  |  |
| --- | --- |
| Actor: | **Requester** |
| Description: | **Encapsulates the following functions**   1. **Post Job** 2. **Delete Job** 3. **Update Status** 4. **Edit Job** 5. **Rate User** |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **REQUESTER\_UC\_2** | | |
| Use Case Name: | **Post Job** | | |
| Created By: | **Shen Hwei (Nicole)** | Last Updated By: | **Shen Hwei (Nicole)** |
| Date Created: | **02.09.2022** | Date Last Updated: | **14.09.2022** |

|  |  |
| --- | --- |
| Actor: | **User (Requester), User (Servicer)** |
| Description: | **When a user is in “Requester” mode, they can post ad-hoc jobs for Servicers to take up. To post a job, the Requester will have to input the location, title of job, description of job, date and time of job and salary. Requester must also have a verified phone number linked to the Requester’s account.** |
| Preconditions: | 1. **A job is created, the job has a status “Created”.** |
| Postconditions: |  |
| Priority: |  |
| Frequency of Use: | **5** |
| low of Events: | 1. **Requester fills up job information: name of job, job description, price of job, location, time and date of job, mode of payment.** 2. **System displays the job onto the job listing page.** |
| Alternative Flows: |  |
| Exceptions: | E-S1-a. Requester cancels job before submission   1. System deletes any saved information on job form about job. 2. System directs user to Main Landing Page.   E-S4-a. Requester deletes job.   1. System interrupts recording of job: 2. If job already added to database    1. Delete from database. 3. If job is not in database yet:    1. Stop the post request to database.    2. Delete post request to database. 4. System directs user to Main Landing Page. |
| Includes: | **Validate Account, Verify Account, Schedule Job** |
| Extends: |  |
| Special Requirements: |  |
| Assumptions: | **The requester will not be able to see the job he posted on the “Jobs” page since he isn’t the servicer and therefore cannot accept his own job.** |
| Notes and Issues: | **[Updates 14/09/2022 - Shen Hwei]**   * **Removed Alt Flow and put into exceptions.** * **Updated Exceptions.** |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **SERVICER\_UC\_1** | | |
| Use Case Name: | **Manage Service** | | |
| Created By: | **Jia Wei** | Last Updated By: | **05.09.2022** |
| Date Created: | **05.09.2022** | Date Last Updated: | **05.09.2022** |

|  |  |
| --- | --- |
| Actor: | **User** |
| Description: | **Encapsulates the following functions**   1. **Accept Job** 2. **Retract Service** 3. **Update Status** 4. **Rate User** 5. **Create Chatbox** |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **SERVICER\_UC\_1** | | |
| Use Case Name: | **Accept Job** | | |
| Created By: | **Shen Hwei (Nicole)** | Last Updated By: | **Shen Hwei (Nicole)** |
| Date Created: | **02.09.2022** | Date Last Updated: | **13.09.2022** |

|  |  |
| --- | --- |
| Actor: | **User (Servicer), User (Requester)** |
| Description: | **When a user is on Servicer mode, they can browse the listing of ad hoc jobs to accept jobs that they wish to take up. Upon accepting the job, the servicer (primary actor) will receive details of the requester (secondary actor). The servicer will only receive pseudo information of the requester. For example, the name is not displayed as the requester’s full name; phone number will be hidden, any communication between the servicer and requester must be done through the application.** |
| Preconditions: | 1. **User must have GPS location enabled. (Limitation)** 2. **User must be in “Servicer” mode.** |
| Postconditions: | 1. **System creates a chat box between the Servicer and Requester.** |
| Priority: |  |
| Frequency of Use: | **5** |
| Flow of Events: | 1. **System checks for preconditions.** 2. **System creates a contract between the servicer and the requester to indicate agreement.** 3. **System creates a chatbox between Servicer and Requester.** 4. **System updates the job status to Assigned.** 5. **System displays job information to Servicer.** 6. **System creates a chat box with call functions between the servicer and requester.** |
| Alternative Flows: |  |
| Exceptions: |  |
| Includes: |  |
| Extends: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: | **[Updates 13.09.2022 - Shen Hwei]**   * **Remove Phone Number Verification and User not Logged in Alt. flows as accepting a job implies existing user which implies phone number already verified upon create account.** |

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | **SERVICER\_UC\_3** | | |
| Use Case Name: | **Create Chat Box** | | |
| Created By: | **Shen Hwei** | Last Updated By: | **05.09.2022** |
| Date Created: | **05.09.2022** | Date Last Updated: | **13.09.2022** |

|  |  |
| --- | --- |
| Actor: | **Servicer** |
| Description: | **Create Chat Box allows the servicer and the requester to communicate with each other. The chat box will be created immediately after the servicer accepts the job. Chat Box also has a call function for users to call one another.** |
| Preconditions: |  |
| Postconditions: | 1. **System directs the case to the Customer Service department.** |
| Priority: | **3** |
| Frequency of Use: | **1** |
| Flow of Events: | 1. **Servicer accepts job.** 2. **System creates chat box under “chats”** 3. **System gives chatbox access to the servicer and requester.** 4. **System updates interface** 5. **System notifies Requester and Servicer of new chat box created** |
| Alternative Flows: | **AF-S1-b. Requester Deletes Job**   1. **Flow continues from REQUESTER\_UC\_3 (Delete Job) S2.** 2. **System deletes the chat box from database.** 3. **System removes the chat box from interface.** |
| Exceptions: |  |
| Includes: |  |
| Extends: |  |
| Special Requirements: |  |
| Assumptions: |  |
| Notes and Issues: | [Updates 13.09.2022 - Shen Hwei]   * **Created** |

# 

# Appendix A: Analysis Model

## Use Case Diagram

## 

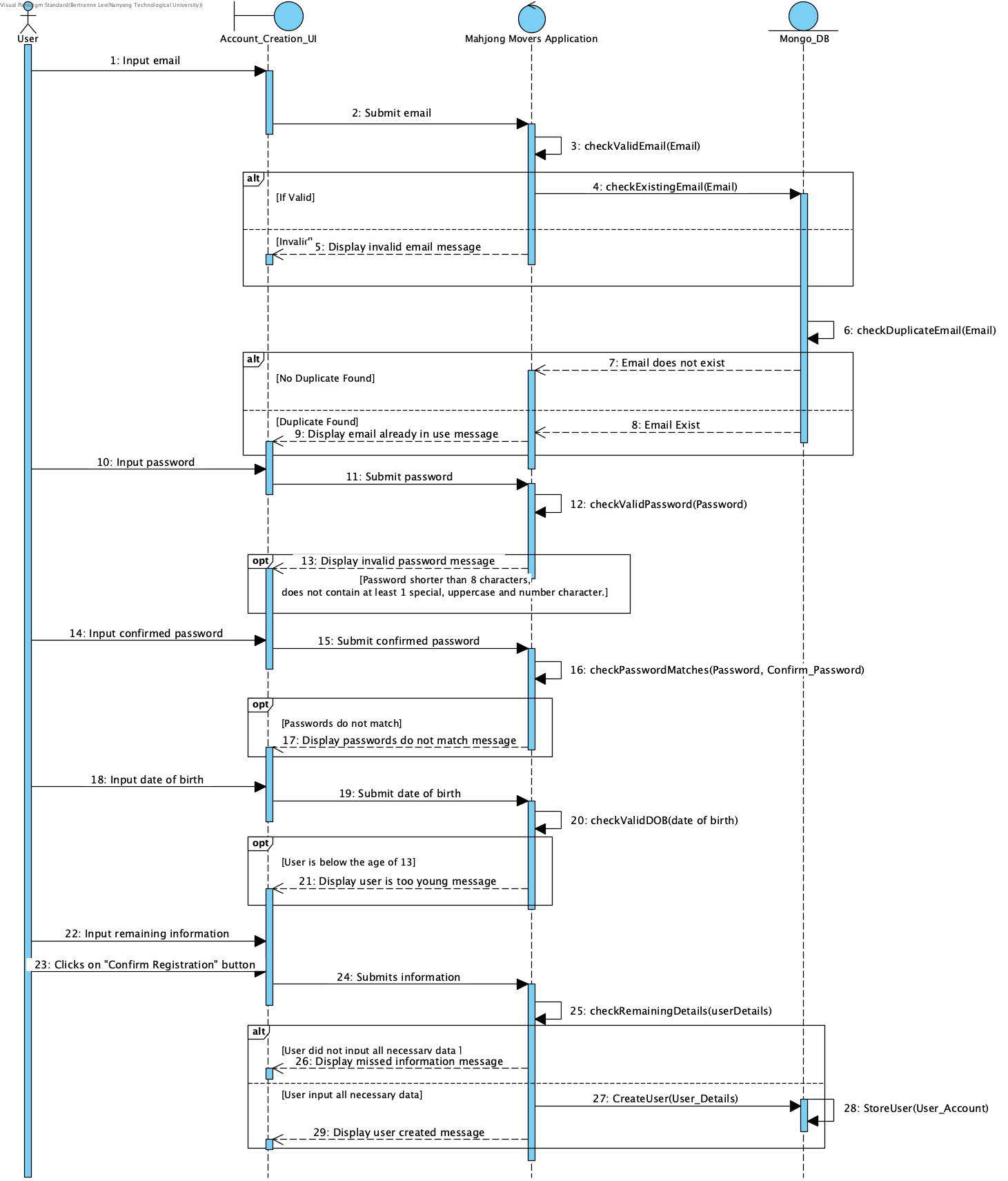
## Dialog Map



## 

## Sequence Diagrams

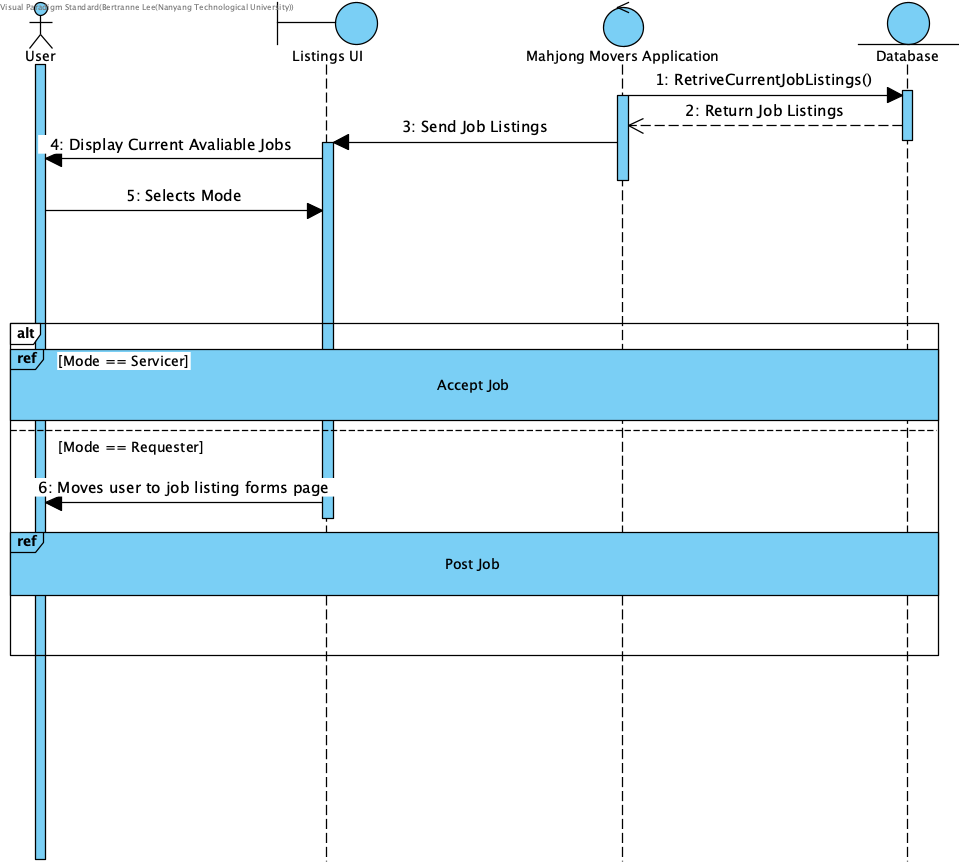
Create Account



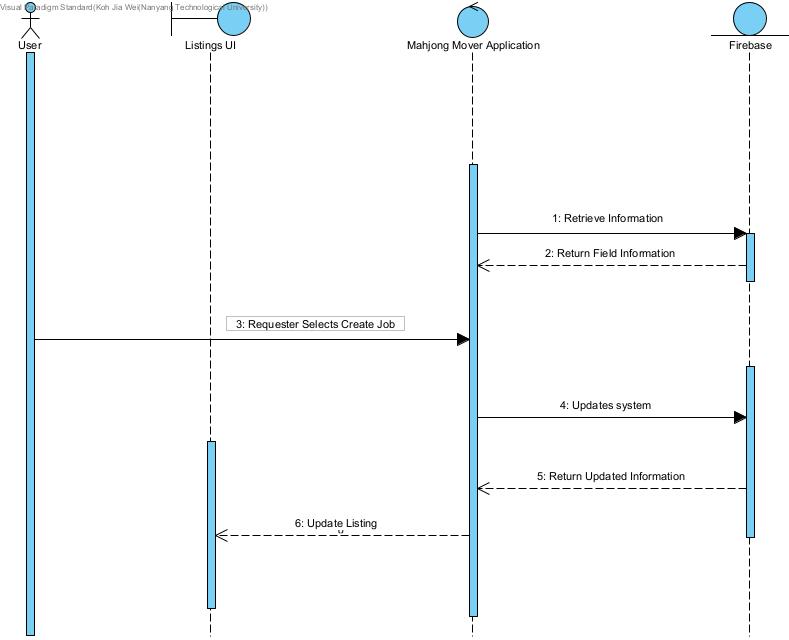
Login



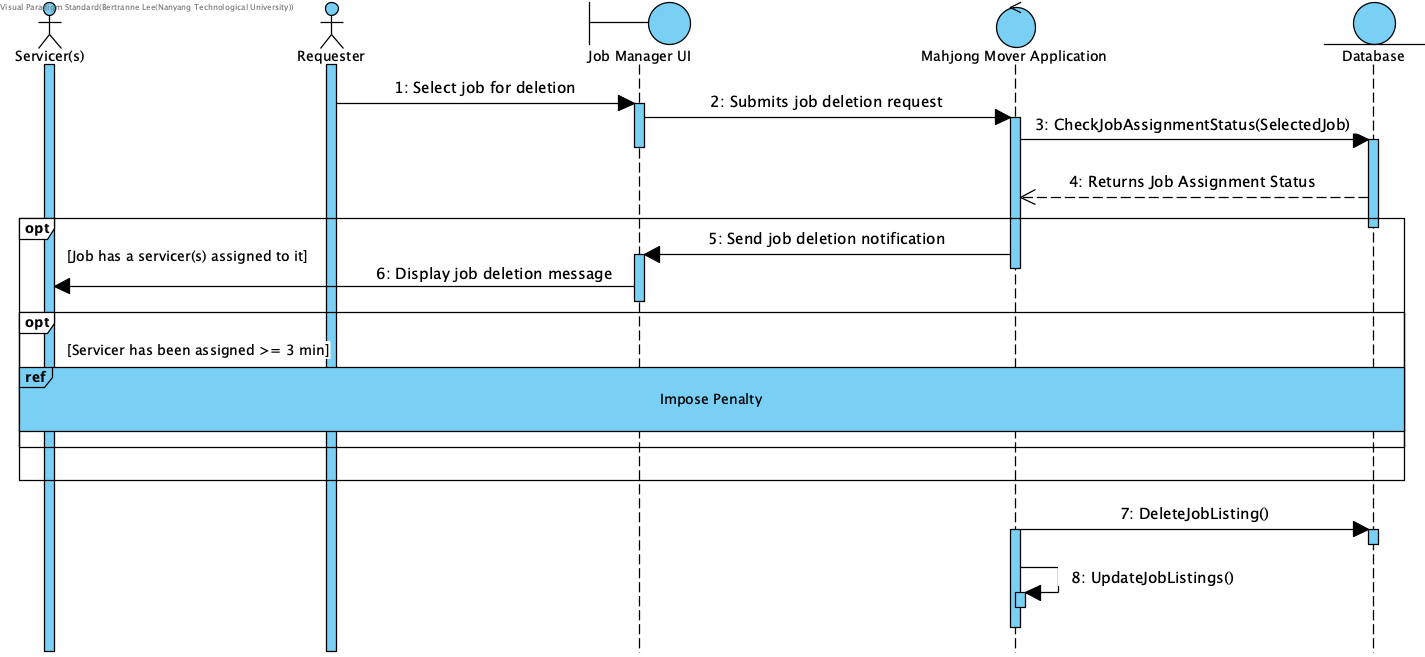
View Listing



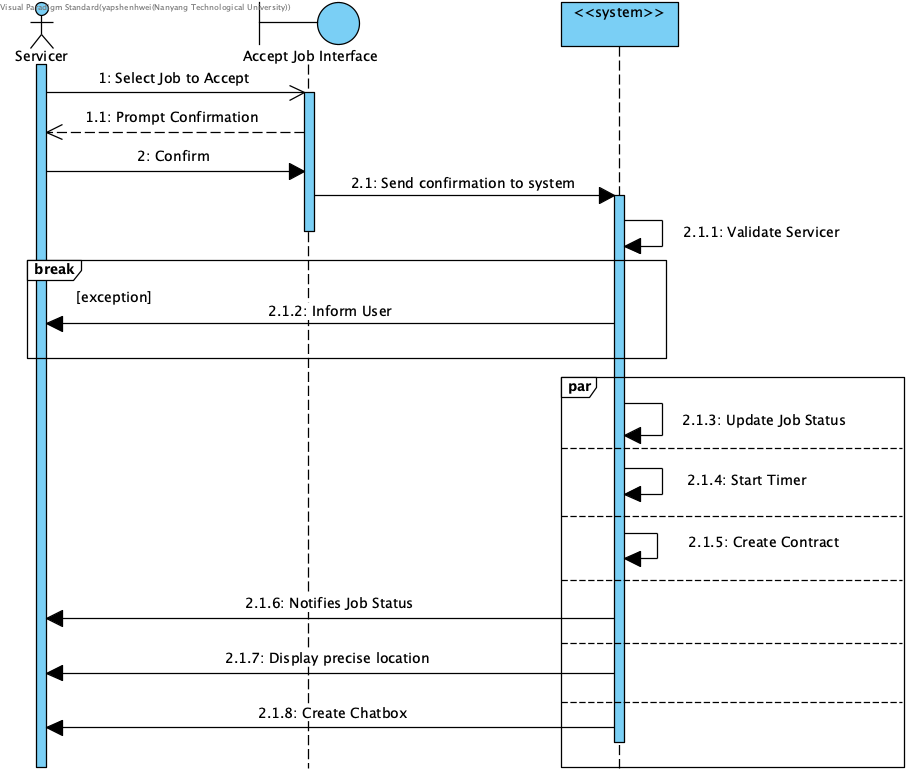
Create Job



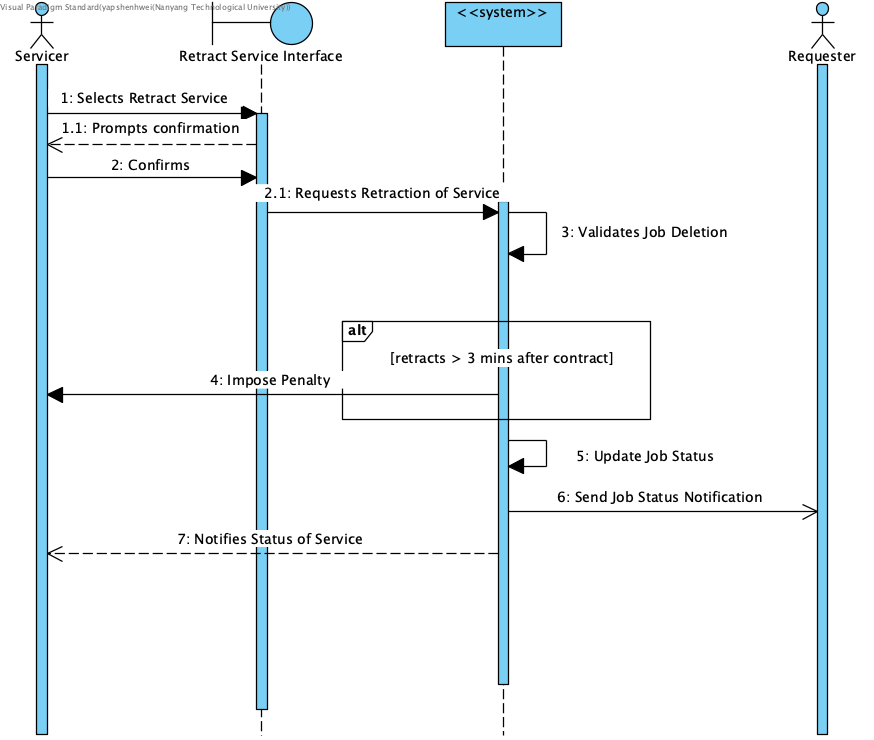
Delete Job



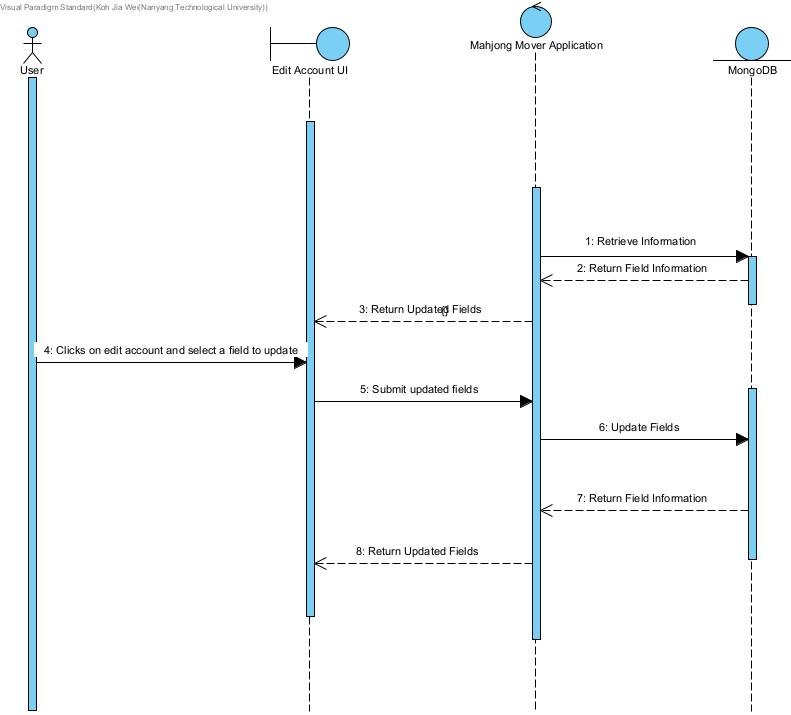
Accept Job



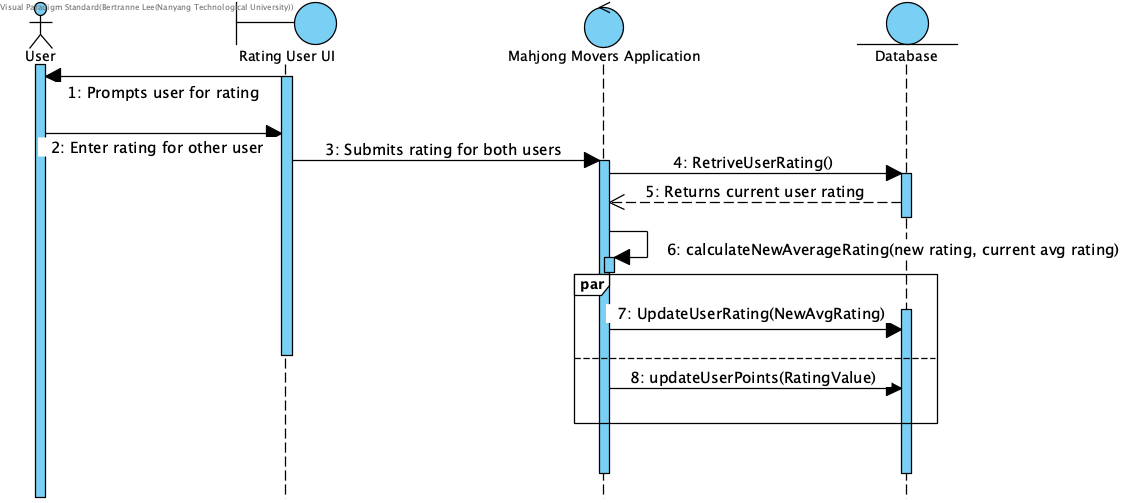
Retract Service



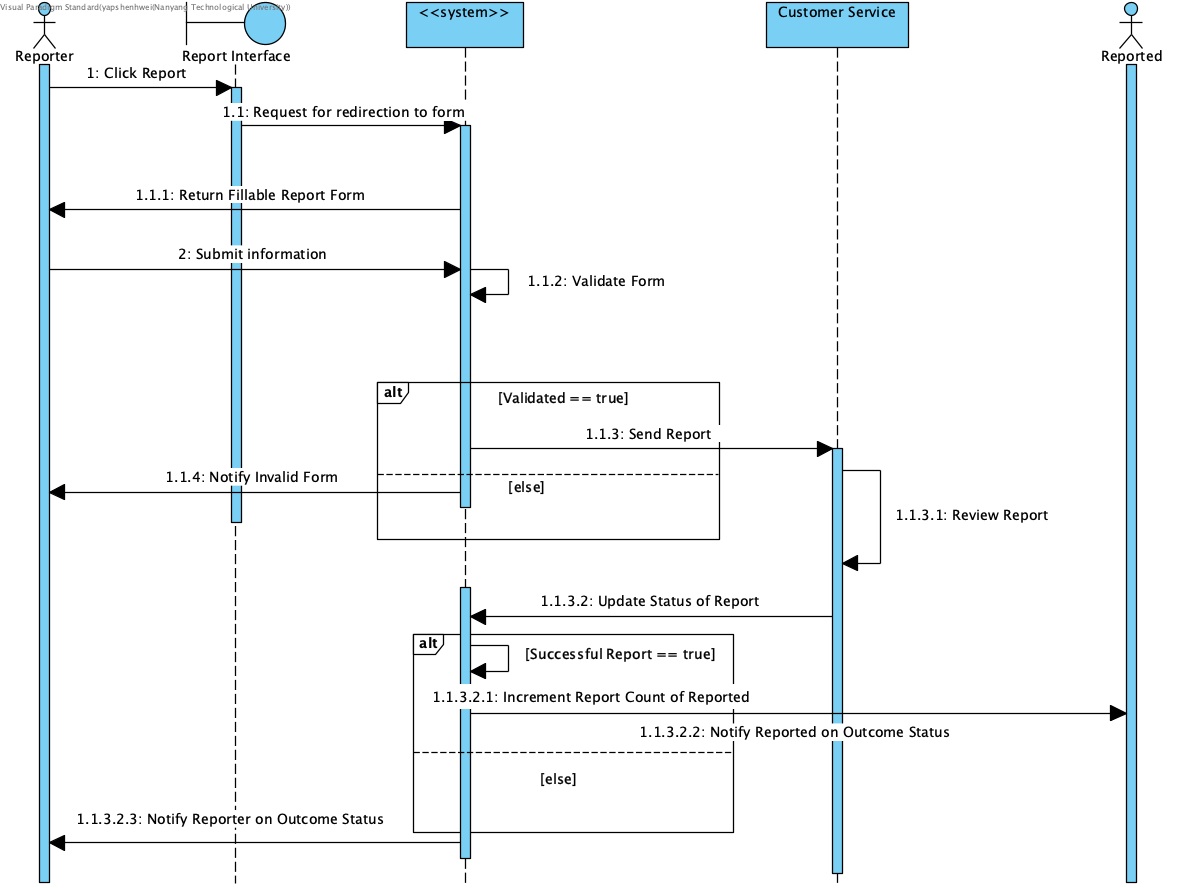
Edit Account Information



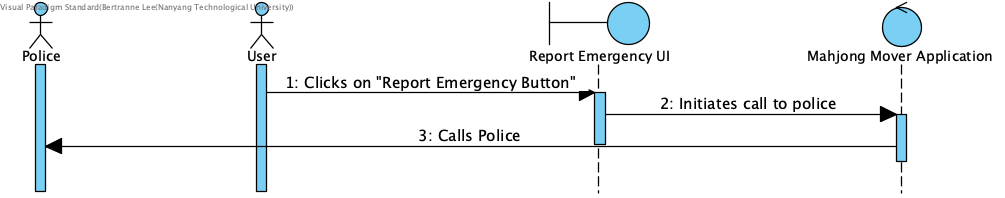
Rate User



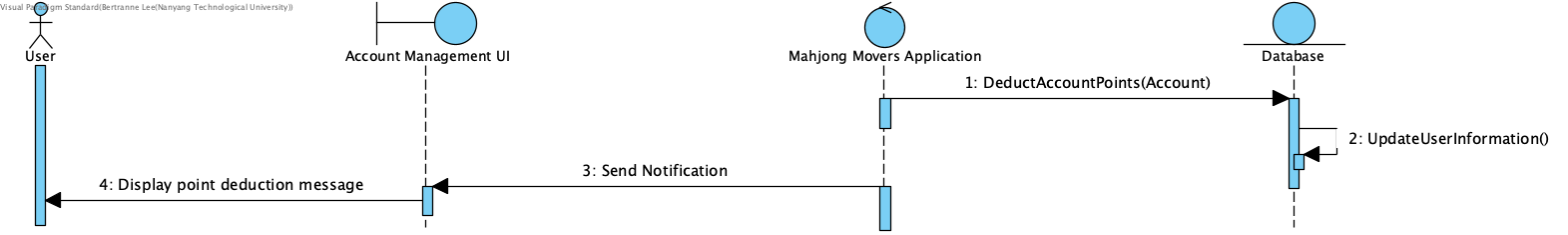
Report To Customer Service



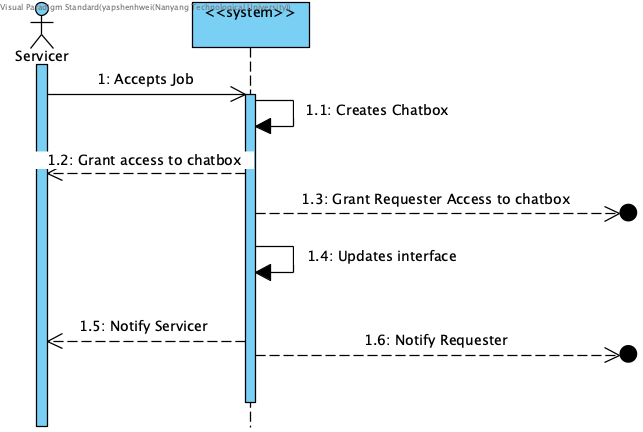
Report Emergency



Impose Penalty



Create Chatbox



# 

# Appendix B: Unit Testing

The two main types of testing done on the Mahjong Mover application is Black Box and White Box Testing. It is done to ensure that the features and functionality of our application do not contain any errors or bugs that may result in incorrect or unexpected outputs. For unit testing, we are only doing specific test cases for black and white box testing, as we feel these are the most important and fundamental functions of our application. These are: Creating Account, Logging In and Create Job for Black Box. While for white box, we will be testing Creating Account and Create Job.

## Black Box

Black Box testing is used to test equivalences classes and boundary value testing.

Test Case 1: Creating Account

Checking if all the user details that a user inputs is valid before creating and inserting their account into our database.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Display Name | Phone Number | Email | Password | Expected Output | Actual Output (Is Log == Oracle?) |
| No Entry | 91507770 | bertrainlee@gmail.com | Password1 | Display Name is Required | Pass |
| Bertrain | No Entry | bertrainlee@gmail.com | Password1 | Phone Number is Required | Pass |
| Bertrain | 91507770 | No Entry | Password1 | Email is required | Pass |
| Bertrain | 91507770 | [bertrainlee@gmail.com](mailto:bertrainlee@gmail.com) (Assuming it was already in the system) | Password1 | The account already exist for that email. | Pass |
| Bertrain | 91507770 | bertrainlee@gmail.com | No Entry | Password is required | Pass |
| Bertrain | 91507770 | bertrainlee@gmail.com | Password1 | Account has been created, please verify your email. | Pass |

Test Case 2: Logging In

Checking if a user’s credentials are valid at the time of login. For this test case, let us assume that their email is [bertrainlee@gmail.com](mailto:bertrainlee@gmail.com) and password is “Password1”

|  |  |  |  |
| --- | --- | --- | --- |
| Email | Password | Expected Output | Actual Output (Is Log == Oracle?) |
| No entry | Password1 | Wrong Email/ Password | Pass |
| bertrain | Password1 | Wrong Email/ Password | Pass |
| bertrain@g.com | Password1 | Wrong Email/ Password | Pass |
| [bertrainlee@gmail.com](mailto:bertrainlee@gmail.com) (Assuming this email has not been used to register for an account) | Password1 | Wrong Email/ Password | Pass |
| bertrainlee@gmail.com | No Entry | Wrong Email/ Password | Pass |
| bertrainlee@gmail.com | ksladfjasidofj0iew123 | Wrong Email/ Password | Pass |
| bertrainlee@gmail.com | Password1 | User is directed to the landing page of the Mahjong Mover Application | Pass |

Test Case 3: Create Job

Checking if a user has entered valid data when they are creating their job posting.

Entries such as Time and Date and Payment type do not need to be validated as they have fixed input values.

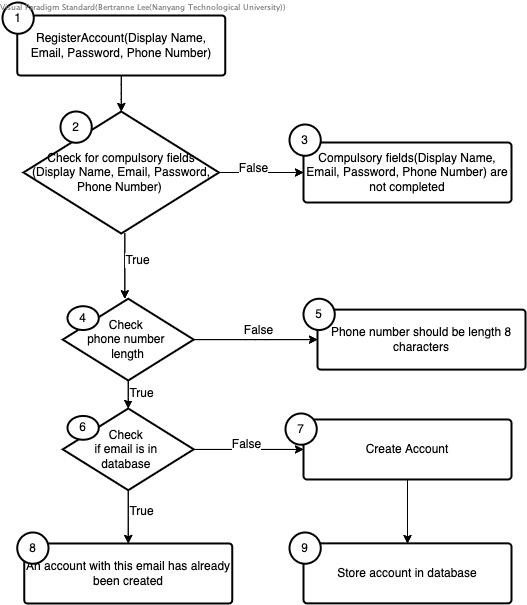
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Job Name | Job Description | Price | Location | Expected Output | Actual Output (Is Log == Oracle?) |
| No Entry | My mahjong table is extremely heavy and I can’t carry it by myself | 10 | NTU Hall 10 | Job Name is required! | Pass |
| Please help me carry my mahjong table from hall 10 to SCSE block | No Entry | 10 | NTU Hall 10 | Job Description is required | Pass |
| Please help me carry my mahjong table from hall 10 to SCSE block | My mahjong table is extremely heavy and I can’t carry it by myself | No Entry | NTU Hall 10 | Price is required | Pass |
| Please help me carry my mahjong table from hall 10 to SCSE block | My mahjong table is extremely heavy and I can’t carry it by myself | 10 | No Entry | Location is required | Pass |
| Please help me carry my mahjong table from hall 10 to SCSE block | My mahjong table is extremely heavy and I can’t carry it by myself | 10 | NTU Hall 10 | Job successfully created | Pass |

## 

## White Box

White box testing was done to ensure that the flow of our application is correct as well as to check for alternative flows in our application.

Test Case 1: Creating Account



Test Cases:

1. User creates an account with all the compulsory fields entered, a phone number input of 8 characters and their email has not been used to create an account.
2. User left one of the compulsory fields (Display Name, Email, Password, Phone Number) empty.
3. User inputs all the compulsory fields, but phone number is invalid (shorter than 8 characters)
4. User inputs all compulsory fields, phone number is 8 characters long but their email has been used to register an account.

Execution Paths:

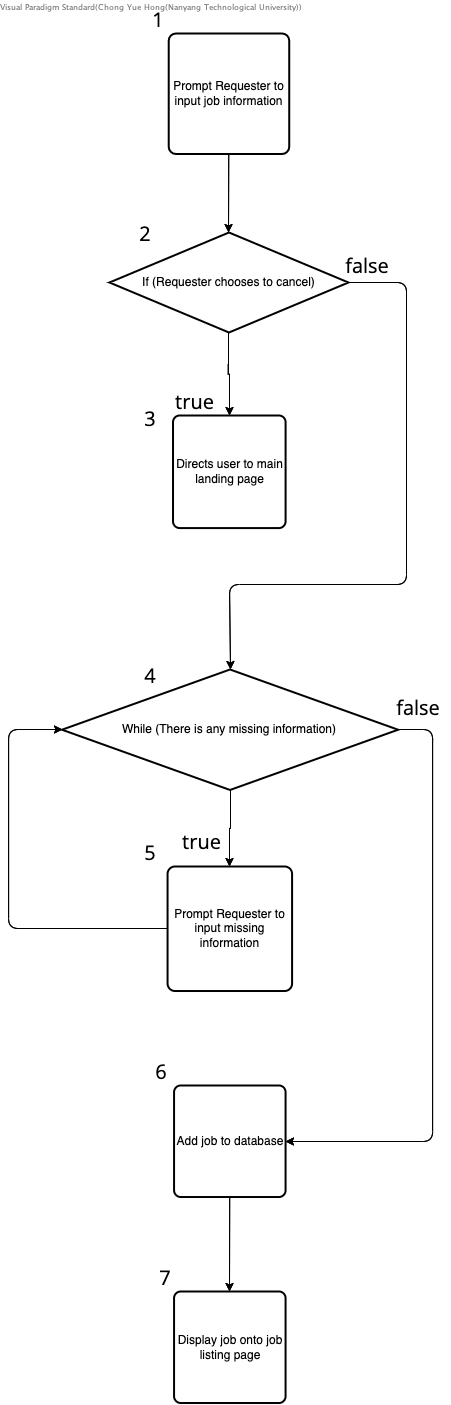
1. 1, 2, 4, 6, 7, 9
2. 1, 2, 3
3. 1, 2, 4, 5
4. 1, 2, 6, 8

Cyclomatic Complexity:

[Edges - Nodes] + 2 = [8 - 6] + 2 = 4

[Conditions] + 1 = 3 + 1 = 4

Test Case 2: Posting Job



Test Cases:

1. Requester tries to post job
2. The requester chooses to cancel the posting
3. The requester clicks ‘submit’ without inputting all information

Execution Paths:

1. 1, 2, 4, 6, 7
2. 1, 2, 3
3. 1, 2, 4, 5, 4, 6, 7

Cyclomatic Complexity:

[Edges - Nodes] + 2 = [7 - 7] + 2 = 2

[Conditions] + 1 = 2 + 1 = 3

# Appendix C: Final Notes

For clearer images of our various diagrams as well as the source code of the Mahjong Movers Application (stored in the “mahjong\_movers” folder). Please refer to our Github (as seen at the bottom of the page) or the SVN repository.

Our demonstration video can be found at: <https://www.youtube.com/watch?v=qQqhNCfb_UQ>