

**CP2106**

**Independent Software Development Project**

**Milestone 3 README**

Team: 2428

HelpBuyCan?

Kan Yu Xuan | Low Si Jie

Advisor: Cheng Xianhao

25/07/2021

Proposed Level of Achievement: Apollo 11

**Content Page**

\*You may click on the words to be brought to the respective sections in the document.

| **Section** | **Page Number** |
| --- | --- |
| 1. [**Overview**](#_l2limx4tb8ae) |  |
| 1. [Motivation](#_ctewedz7ug8i) | 2 |
| 1. [Aim](#_xtlmsfxw4pjm) | 2 |
| 1. [User Stories](#_oxx0uhv000fy) | 3 - 4 |
| 1. [**Project Specification**](#_7jhxcbvffix8) |  |
| 1. [Scope](#_aywr1wa323ak) | 5 |
| 1. [Technology Stack](#_bf2m7rcapho1) | 6 |
| 1. [Project Management Tools](#_3v5dbkln9foq) | 6 |
| 1. [Features of Application](#_yjz2g5lml3s6) 2. [Register page](#69rhn5k36vqo) 3. [Login page](#sh014662wcax) 4. [Main page](#nx3py82jx2d8) 5. [Add offer](#glrg7ccyfmm0) 6. [Add request](#99c8lnyt05t1) 7. [List of offers from deliverers](#eqqtfryngzq4) 8. [List of requests from buyers](#j7fzggq496ju) 9. [My Transactions](#ks3dwrkalmi3) 10. [My Chats](#bk987enotb22) 11. [Settings](#79ca4d9q9hi4) 12. [Enhancing User Experience (UX)](#8q48mzr3k1xd) | 7 - 20 |
| 1. [**Bug Report**](#_c2yl6iii8dht) | 21 - 22 |
| 1. [**Software Engineering Practices**](#_1h0d3qnvmk6q) |  |
| 1. [Software Engineering Principles](#_iuljrpw56h7f) | 23 |
| 1. [Improvement from Milestone 2](#_a9oguci8c7xi) | 24 |
| 1. [Testing of Application](#_wdhz1zv3mn5v) 2. [System Testing Plan](#_b8duc1pcd5hu) 3. [User Testing](#_sibdmxzgqbnh) | 25 - 31 |
| 1. [**Problems Encountered & Solutions**](#_l6vr2okvt9xr) | 32 |
| 1. [**Future Plans**](#_hk1cjsolsqut) | 33 |

# Overview

## MOTIVATION

Ever had to travel somewhere far to run a simple errand? Travelling somewhere out of campus is always a hassle, and it may not be worth the time and effort just to carry out a simple task. You could be using the trip more efficiently by completing other tasks, such as helping to buy groceries or necessities for others on campus.

On the other hand, you may need something non-urgently but it is too inconvenient for you to travel out of campus. Existing delivery platforms like Grab, Foodpanda, PandaMart may not have the specific item that you are looking for, or charge high delivery fees that are just not worth the purchase.

## AIM

Help Buy aims to target NUS staff and students living on campus to conveniently purchase items from elsewhere, or to be able to help buy goods for others when they are out. We hope to create a network that fulfils the needs of buyers at the deliverer’s convenience.

To make each trip more worth the time and effort, our mobile application, Help Buy, gives users the option to become ‘deliverers’. As a deliverer, he/ she can provide the service of purchasing items from where he/ she is at for staff or students staying on campus. In return, the deliverer will be paid for the service.

At the same time, Help Buy allows users to put up requests for things that they want to purchase. This group of users will be referred to as ‘buyers’. To skip the hassle of having to personally go to a location far away from campus to purchase an item, buyers can put up requests for things that they want to buy. A deliverer will then accept the request and bring the item back to campus.

## USER STORIES

| **As a** | **I must be able to** | **So that I can** | **Features** |
| --- | --- | --- | --- |
| **Potential user** of the app | Create an account | Start the process of purchasing/delivering items. | Signup/ Login function |
| **Buyer** staying on campus who is interested in purchasing an item | Request for an item to be delivered | Receive the item in the comfort of my own room/ do not need to travel out for it. | Put up job requests with the following details:  - location  - store  - item and estimated price  - delivery time  - price willing to pay extra for |
| View job offers from deliverers in the vicinity | Make my decision on whether to purchase after looking through all the options. | View list of job offers from deliverers |
| **Buyer** who purchased an item | Contact the deliverer | Communicate with the deliverer if I would like to make any additional requests (changes to order, enquiry of products, etc.) and also make arrangements for payment. | Chat function between buyer and deliverer |
| View past transactions | Track the amount spent in the past. | Transaction history |
| **Deliverer** travelling out of campus | Put up a job offer for the location i am travelling to | Make use of my trip more efficiently. | Add job offer  - location  - delivery time  - price willing to get paid |
| Know the list of requests put up by buyers | Decide on which requests to accept/ apply to me the most after looking through all the options. | View list of requests from buyers |
| **Deliverer** who is sending an order | Contact the buyer | Communicate with the buyer if there are any changes in the orders which I need to confirm with the buyer. | Chat function between buyer and deliverer |
| **General user** of the app | Give feedback for other users I interacted with | Give good feedback or report any irresponsible actions. | Feedback feature |
| Edit my username, phone number and password | Have the flexibility of editing my profile to whatever I want. | Edit profile & Change password in Settings |

# Project Specification

## SCOPE

Our mobile application only targets NUS staff and students that stay on campus so as to narrow the scope of this project.

The project will be split into 3 phases: Phase 1, 2 and 3. Phase 1 and 2, which we target to complete by end June and mid July respectively, are the main development phases. They involve the implementation of core features of our mobile application. Phase 3 involves the addition of features that will enrich user experience. We aim to complete this phase by the end of July (excluding extensions).

**Phase 1: Core Features I**

* Login/Sign up feature

Buyer interface:

* Add job requests with the following details:
  + Location
  + Item Description
  + Quantity
  + Estimated Price
  + Delivery date, time, fees

Deliverer Interface:

* Add job offers with the following details:
  + Location
  + Date of Purchase
  + Duration
  + Minimum Fees Request

**Phase 2: Core Features II**

* Chat function between buyer and deliverer (Part 1)
* Transaction history

Buyer interface:

* View list of job offers from deliverers

Deliverer interface:

* View list of requests from buyers

**Phase 3: Additional Features**

* Chat function between buyer and deliverer (Part 2)
* Edit profile feature in Settings
* Email verification
* Feedback & Rating Feature
* **EXTENSIONS** *(to be implemented after Orbital ends)*
  + Real-time update of the status/ location of the items purchased
  + Share location service: Location API
  + Sending alerts (notifications)

A more detailed outline of our project and its phases is included in this [roadmap](https://docs.google.com/spreadsheets/d/1lhZF_DEBzff0vNY_aAo7NKWwgYQf5IloFcBrTc8O1pI/edit?usp=sharing).

## TECHNOLOGY STACK

Our mobile application will be developed for the Android platform using the following technologies:

Programming Language: Java

Toolkit: Android Studio

Database: Firebase

## PROJECT MANAGEMENT TOOLS

* 1. Git & Github
* This is our [GitHub](https://github.com/joeyylow/HelpBuy) repository.
* We made use of Git to store our codes locally before pushing to/ pulling from GitHub remotely to merge our codes.
  + This ensures that we are constantly updated and working on the same page.
  + This also prevents complicated merge conflicts at the end of the project which would be difficult to solve and debug.
* We have 3 working branches (*master*, *sijie*, *yuxuan*)
  + For every feature we would push the codes from our local repository to our own branch in Github.
  + Then we would merge both our branches with the *master* branch and solve any merge conflicts which arise.
  + Once solved, we would pull the codes from the *master* branch to our own branch on GitHub.
  + Lastly, we would do a git pull to pull the codes to our local branch on our computers.
  1. Project Roadmap
* We followed our project roadmap closely to ensure that we are on task and in line with what we planned for for every phase of the project.

1. DevOps: Continuous Integration

* Our team automated the integration of code changes.
* Since the whole project has 2 contributors who work on different parts of the file, it will be difficult to manually merge the codes.
* Therefore, we made use of Github and Android studio to automatically debug our codes before committing the merge. Android Studio will help us review all the codes for errors and warnings, such that the merging will be able to go through smoothly. If there are any merge conflicts detected, the merge and build process will be unable to proceed and we will need to resolve the conflicts individually. Therefore, every branch committed in Github will trigger a test process and build process, and the code will need to be error free before we continue with the development.

## FEATURES OF APPLICATION

This section showcases all the features of HelpBuy.

Descriptions of the user role, user goal and user benefit of each feature are stated. Screenshots are also attached for pictorial aid.

How our features enhance user experience (UX) is elaborated at the end of this section.

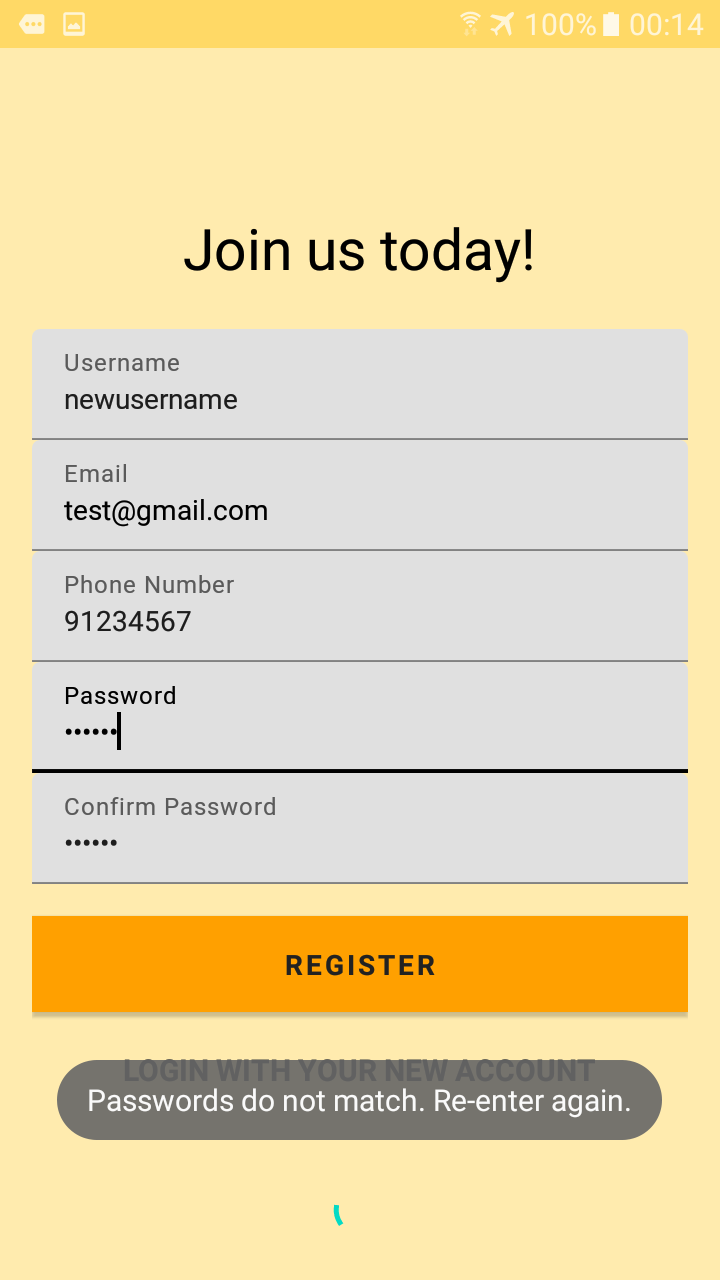
1. Register page

**User Role:** Any member of the public (NUS staff/ student) would be able to register with their credentials.

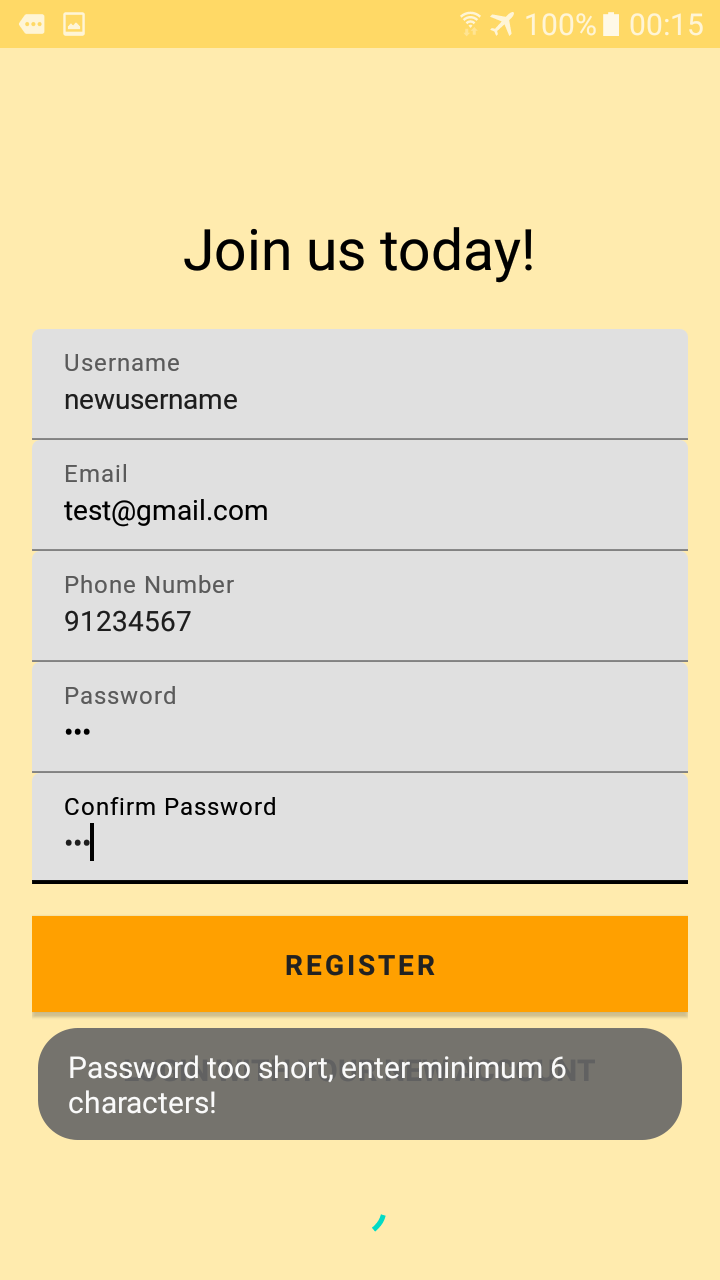
**User Goal:** Every user will have a unique account registered under their name. Each username used by the member must be unique so that their identity can be verified. In addition, there are other criteria for a user to be able to register successfully, such as having a secure password of longer than 6 characters, having a valid email address and requiring them to re-enter their password.

**User Benefits:** After registration, users will be prompted to verify their email before logging in. This is to enhance the security of our app and to ensure the safety of all users on the platform. Users can use their registered credentials to login for subsequent accesses to the application.

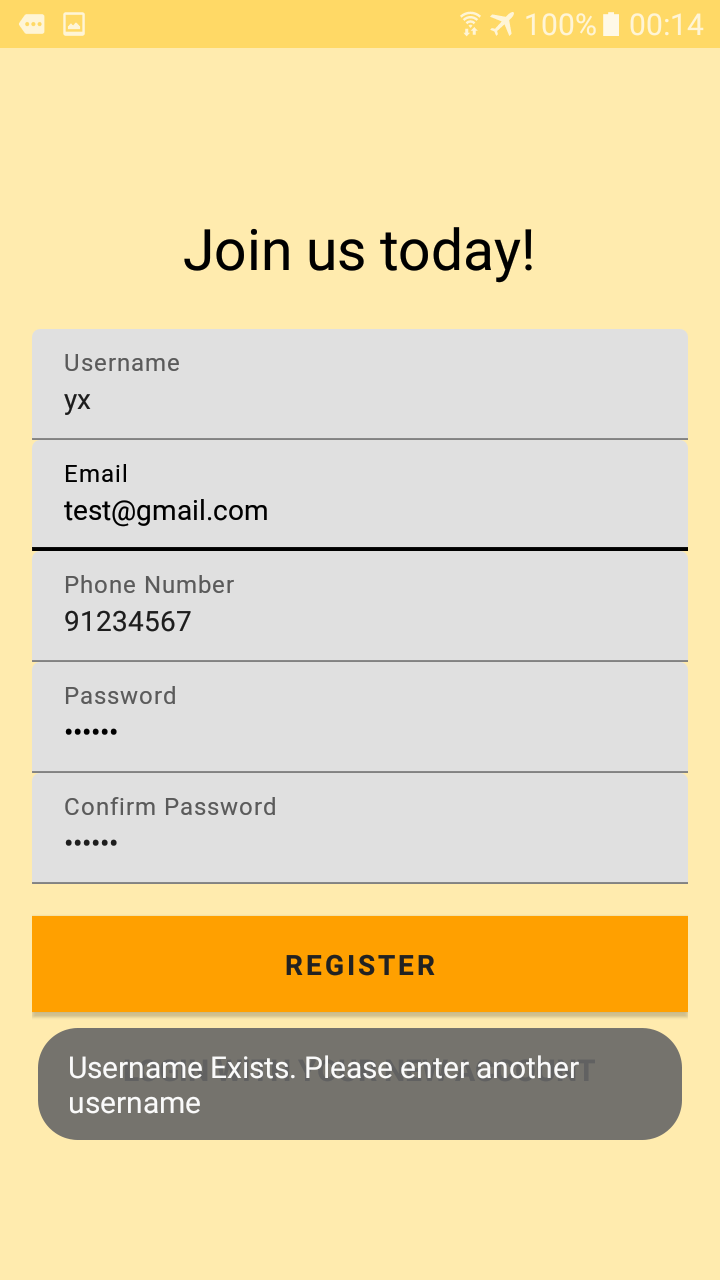
Scenario 1: Passwords do not match



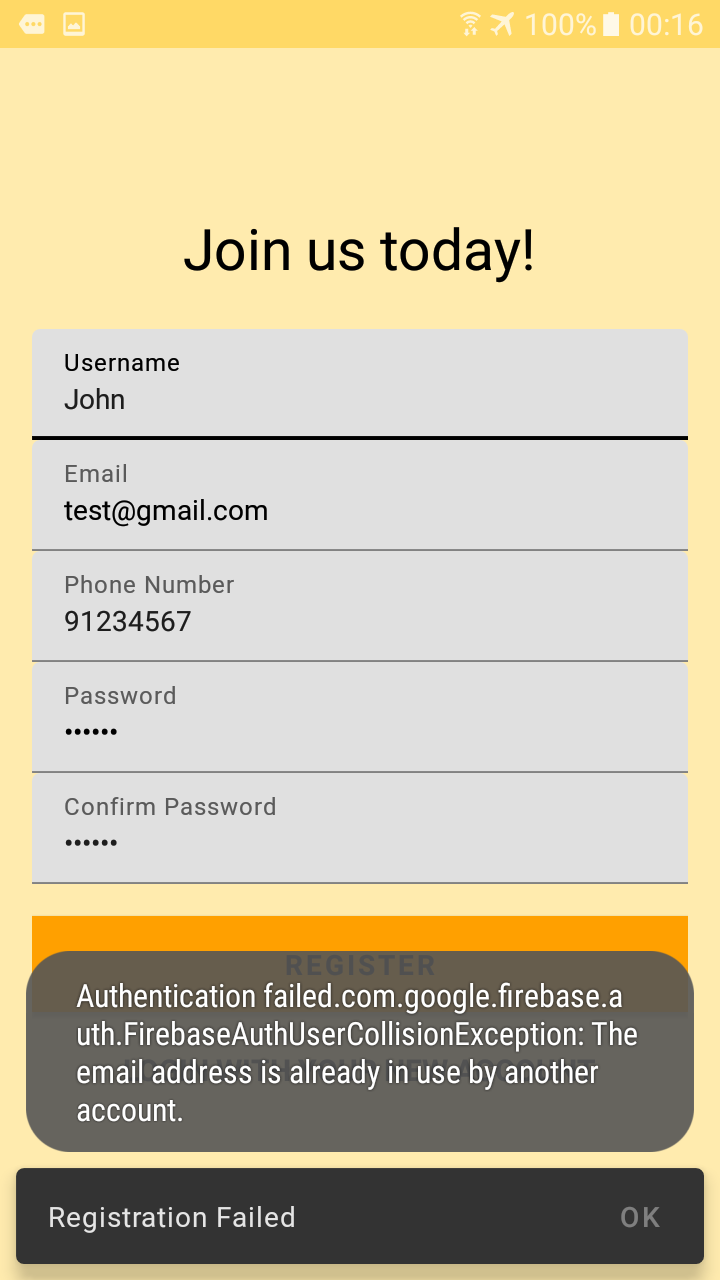
Scenario 2: Password length is less than 6 characters



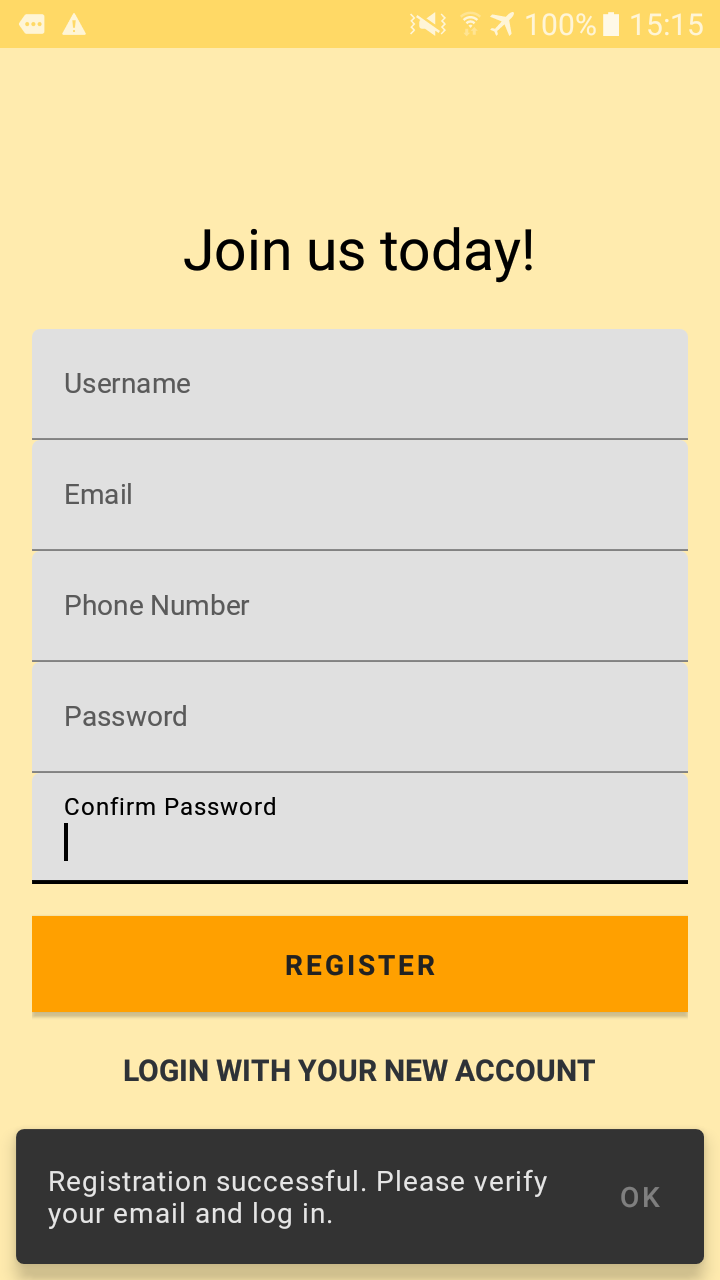
Scenario 3: Username entered already exists in database



Scenario 4: Email address has already been registered



Scenario 5: Successful registration

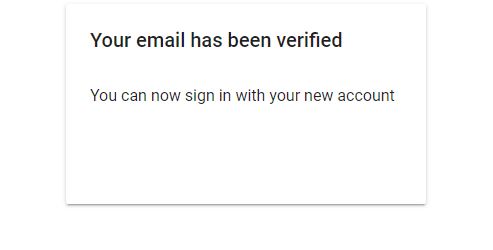
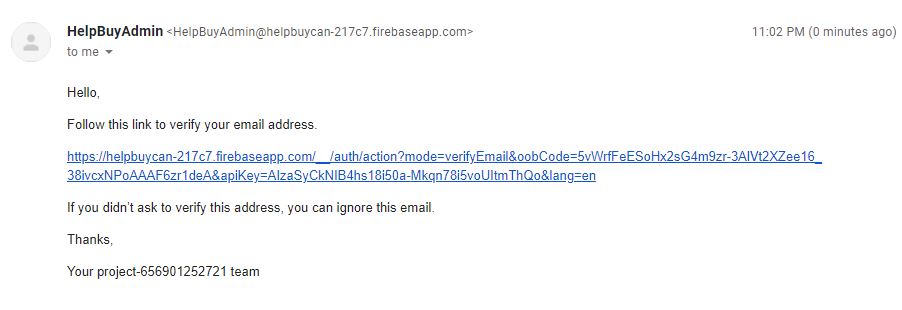
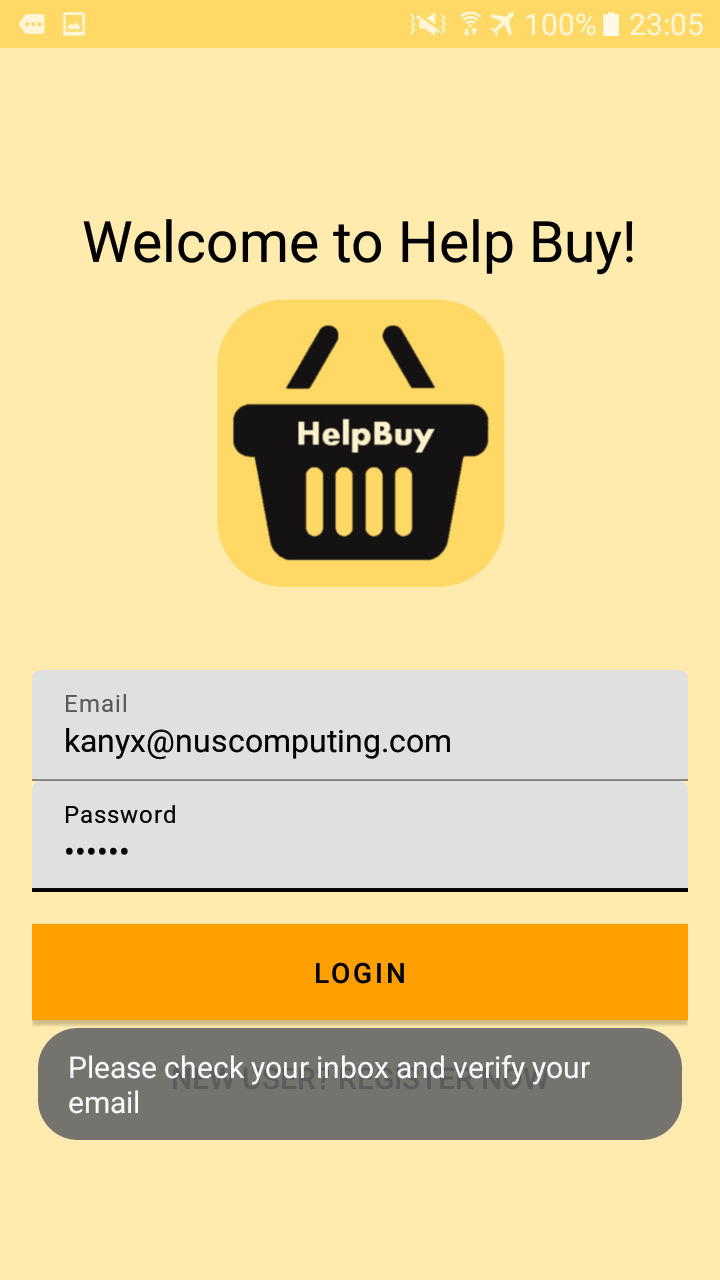
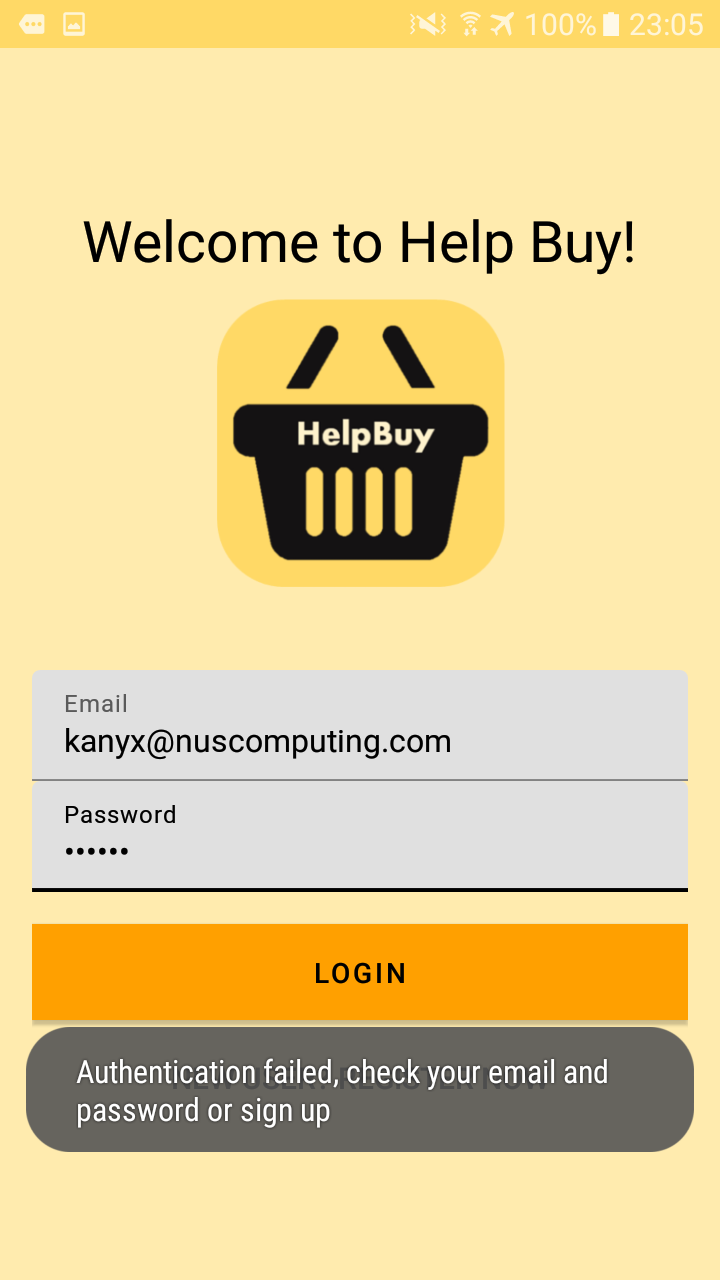


1. Login page

**User Role:** Any member who already registered on the platform and verified their email will be able to login with their credentials in this page.

**User Goal:** With the correct email address and password, they would be able to enter the app. They must also verify their email with the link sent to their email address, without it they will not be able to access their account.

**User Benefits:** The login page also contains the register button such that new users (members of the public who have not used the platform before) can navigate to the register page to create a new account. The email authentication serves as a tool to ensure every user in the database is a legitimate user (not a bot).

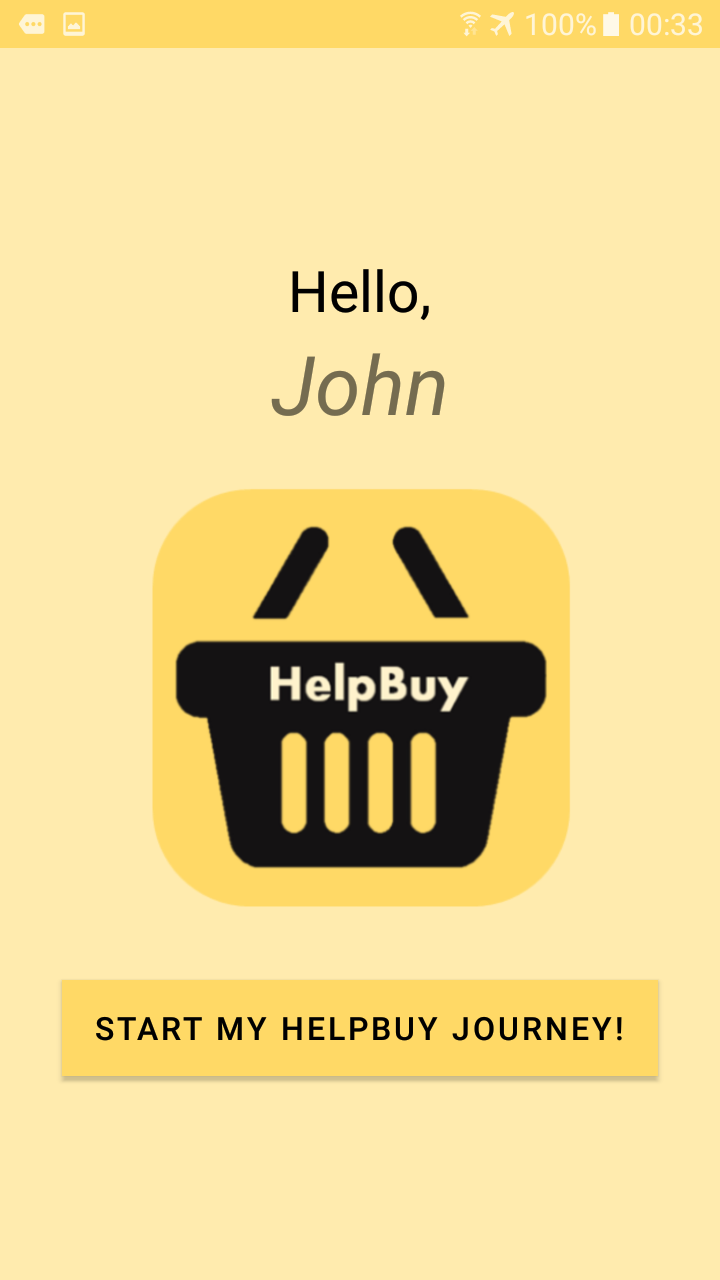


1. Main page

**User Role:** Users who have successfully logged in will be able to enter this main page.

**User Goal:** The main page will show the name of the user, for users to confirm that they have successfully logged in.

**User Benefits:** This creates a confirmation for users to let them know they are in the app and have the option to enter the features.

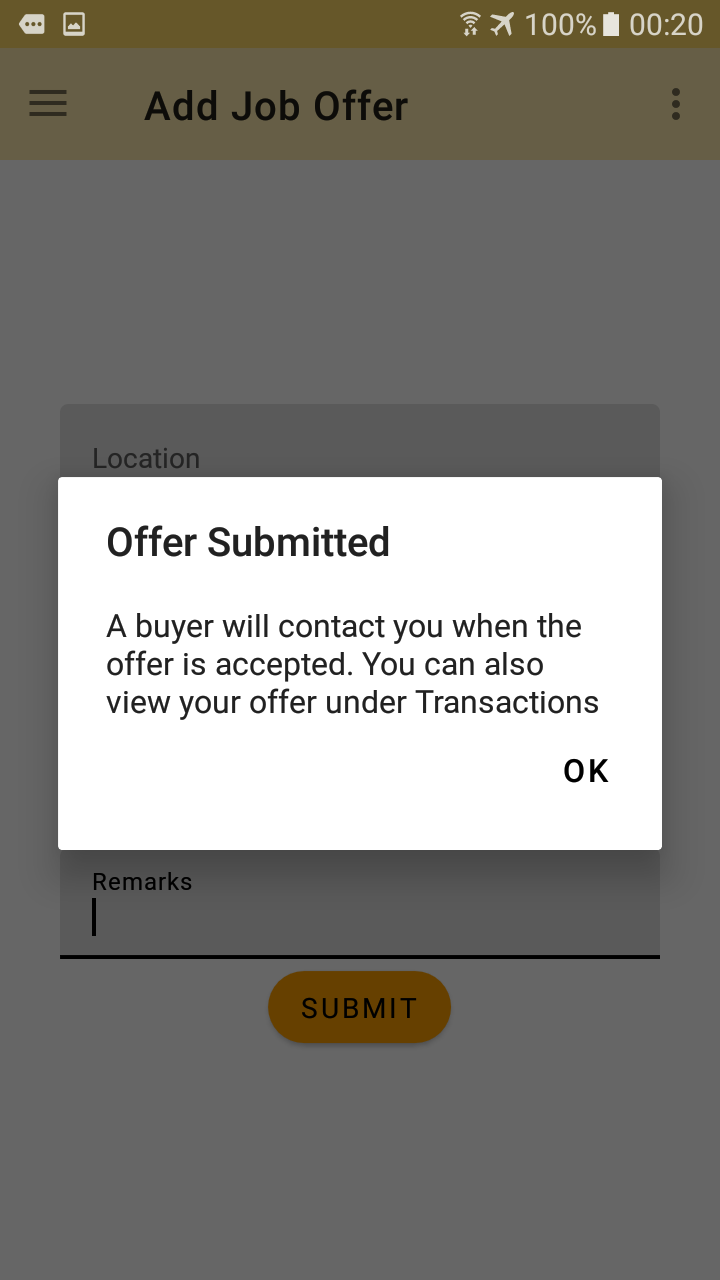
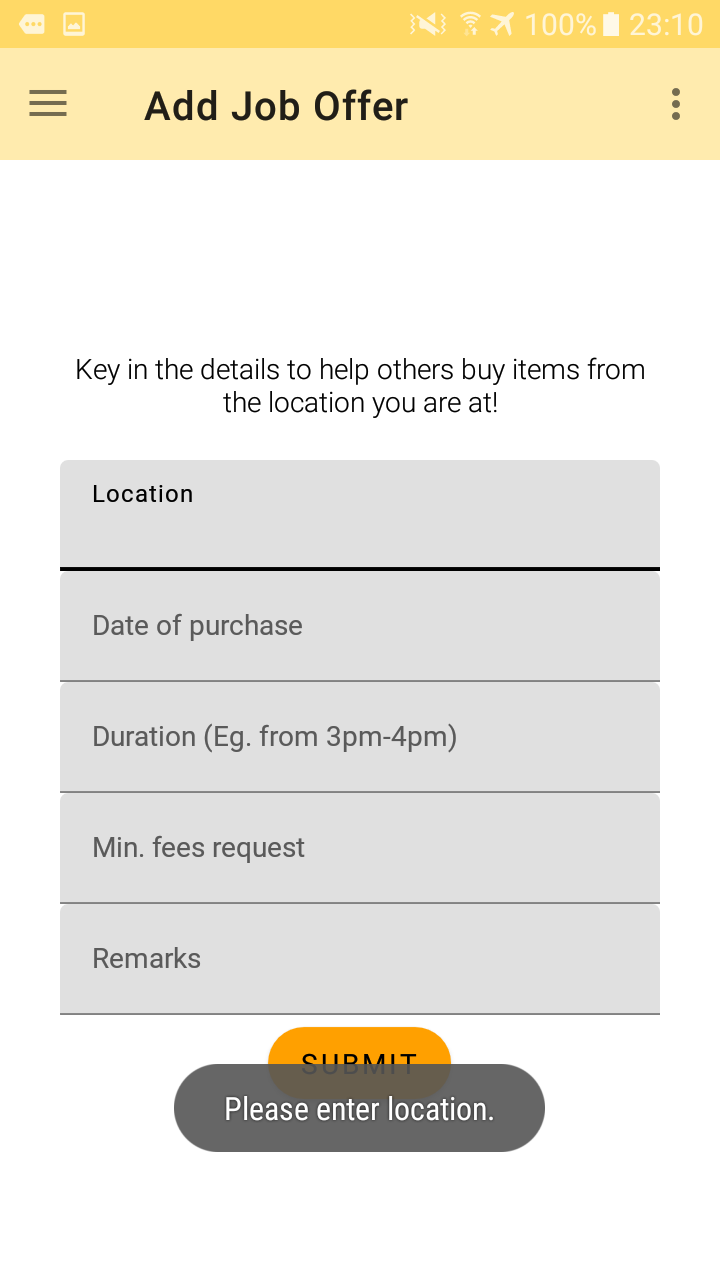
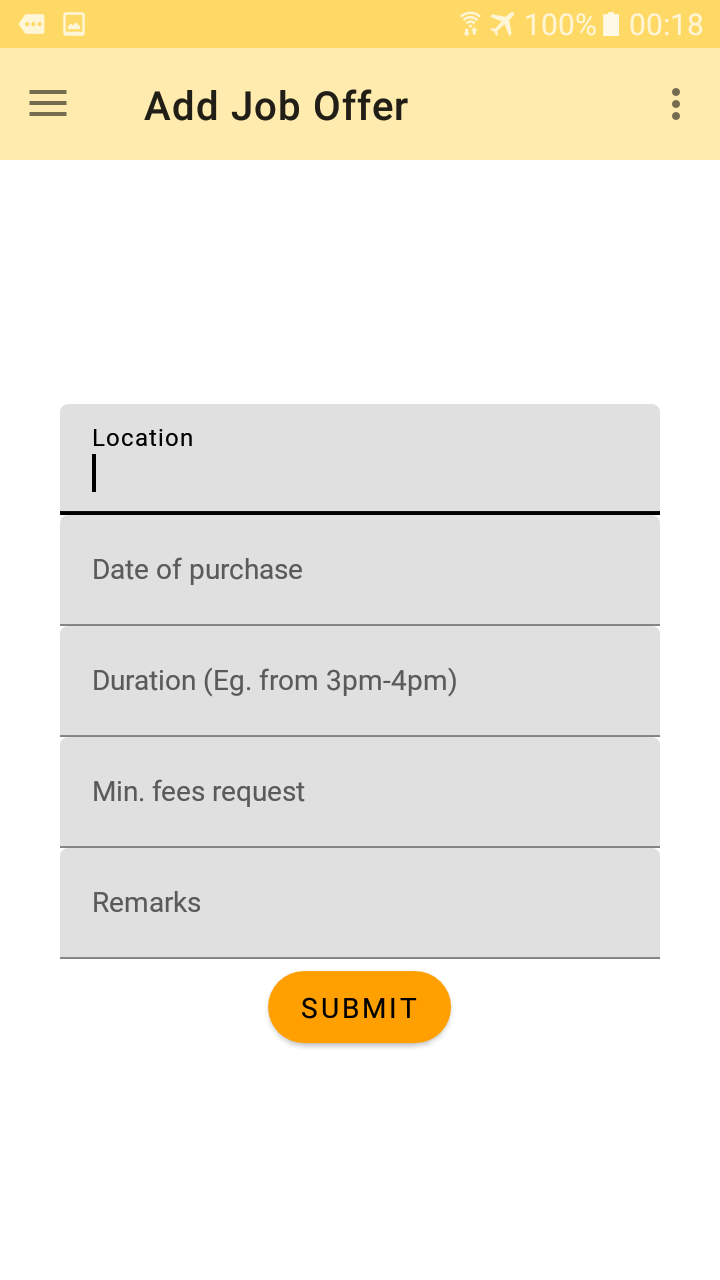


1. Add offer

**User Role:** As a deliverer, if i am travelling to a location which i can help to buy items, I would be able to add a job offer with the relevant fields such as location, date, duration I will be at the location for, delivery fees i am requesting and additional remarks. Some fields are mandatory as they give the buyers enough information to accept the offer.

**User Goal:** Clicking the submit button will automatically add a job offer into the database and it can be viewed by all deliverers from the List of Offers.The dialog box gives proper instructions as how to navigate to the offer to check if it has been accepted.

**User Benefits:** This will add the job offer to the database such that other potential buyers will be able to accept it.

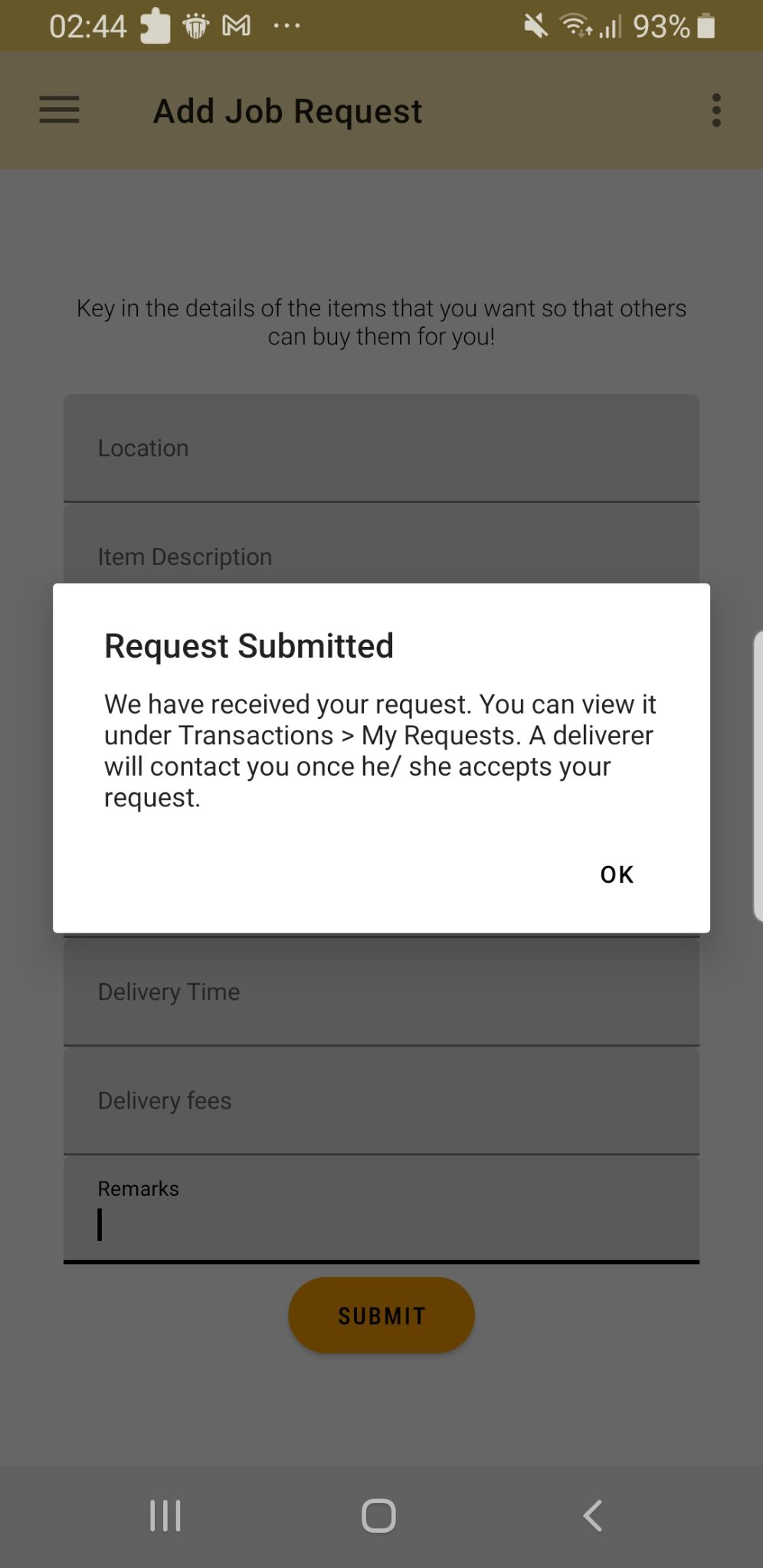
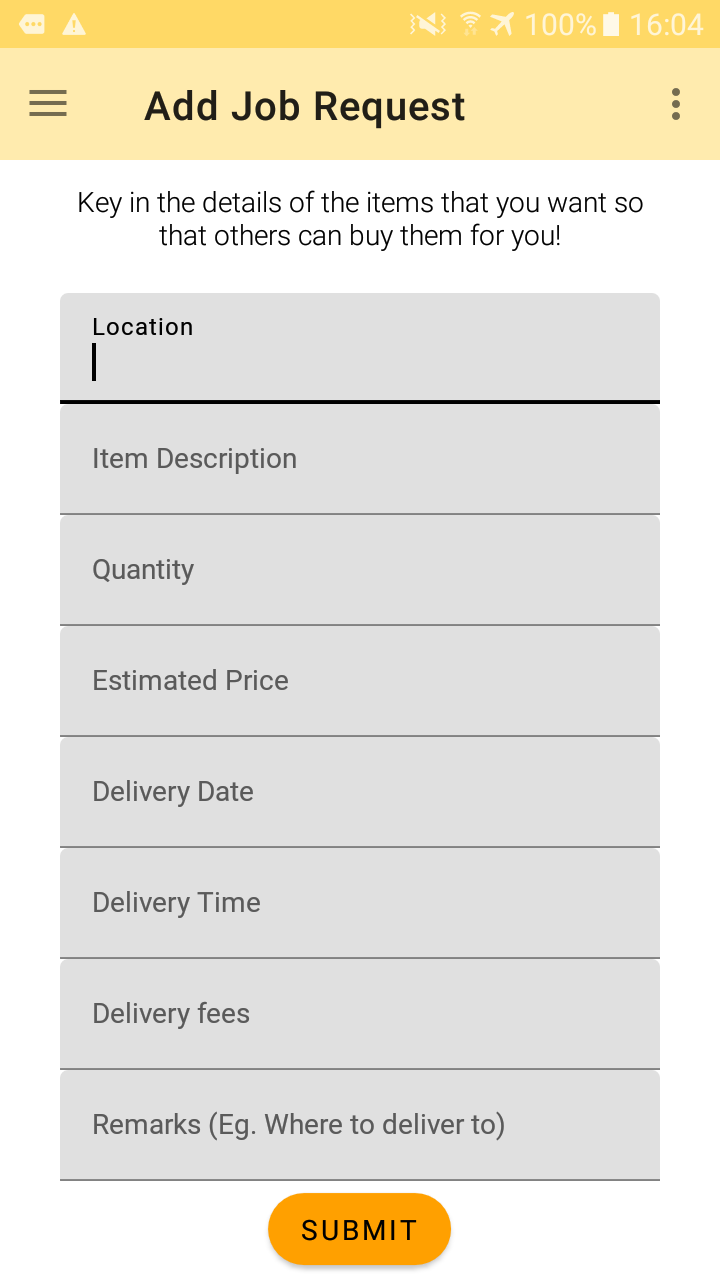


1. Add request

**User Role:** As a buyer, if I wish to get someone else to help me purchase an item, I would be able to add a job request with the relevant fields as shown below. Some fields are mandatory as they are necessary to provide deliverers with sufficient information to accept the request.

**User Goal:** Clicking the submit button will automatically add a job request into the database and it can be viewed by all deliverers from the List of Requests. The dialog box gives proper instructions as to how to navigate to the request to check if it has been accepted.

**User Benefits:** This will add the job request to the database such that other potential deliverers will be able to accept it.

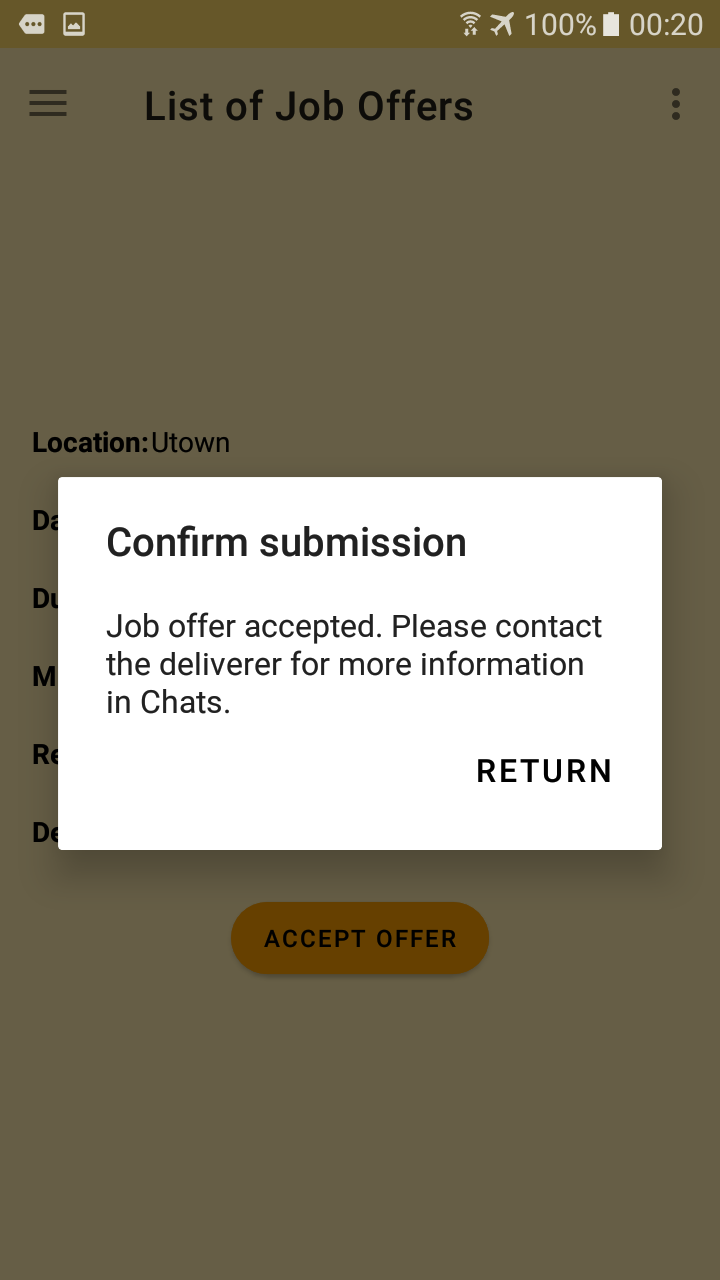
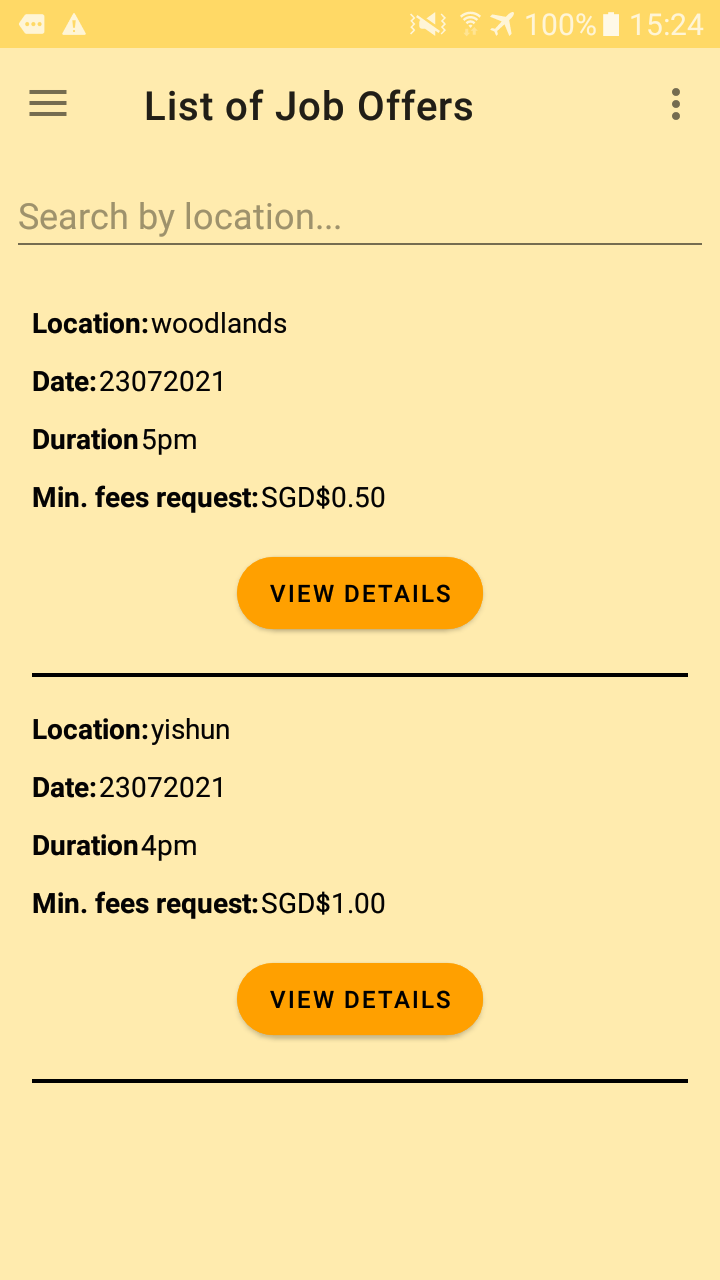


1. List of offers from deliverers

**User Role:** As a buyer, the List of Offers contains all the job offers (that are not yet accepted) in the database.

**User Goal:** Once the ‘View Details’ button is pressed, there will be a page showing the details of the offer and the user to contact if they need more information. By clicking on the ‘Accept Offer’ button, they will be prompted to contact the deliverer and return to the previous list where the offer will be gone.

**User Benefits:** A buyer will be able to find locations where others are at and request for an item without having to personally travel to get the item. The buyer can also conveniently search for an offer based on specific locations.

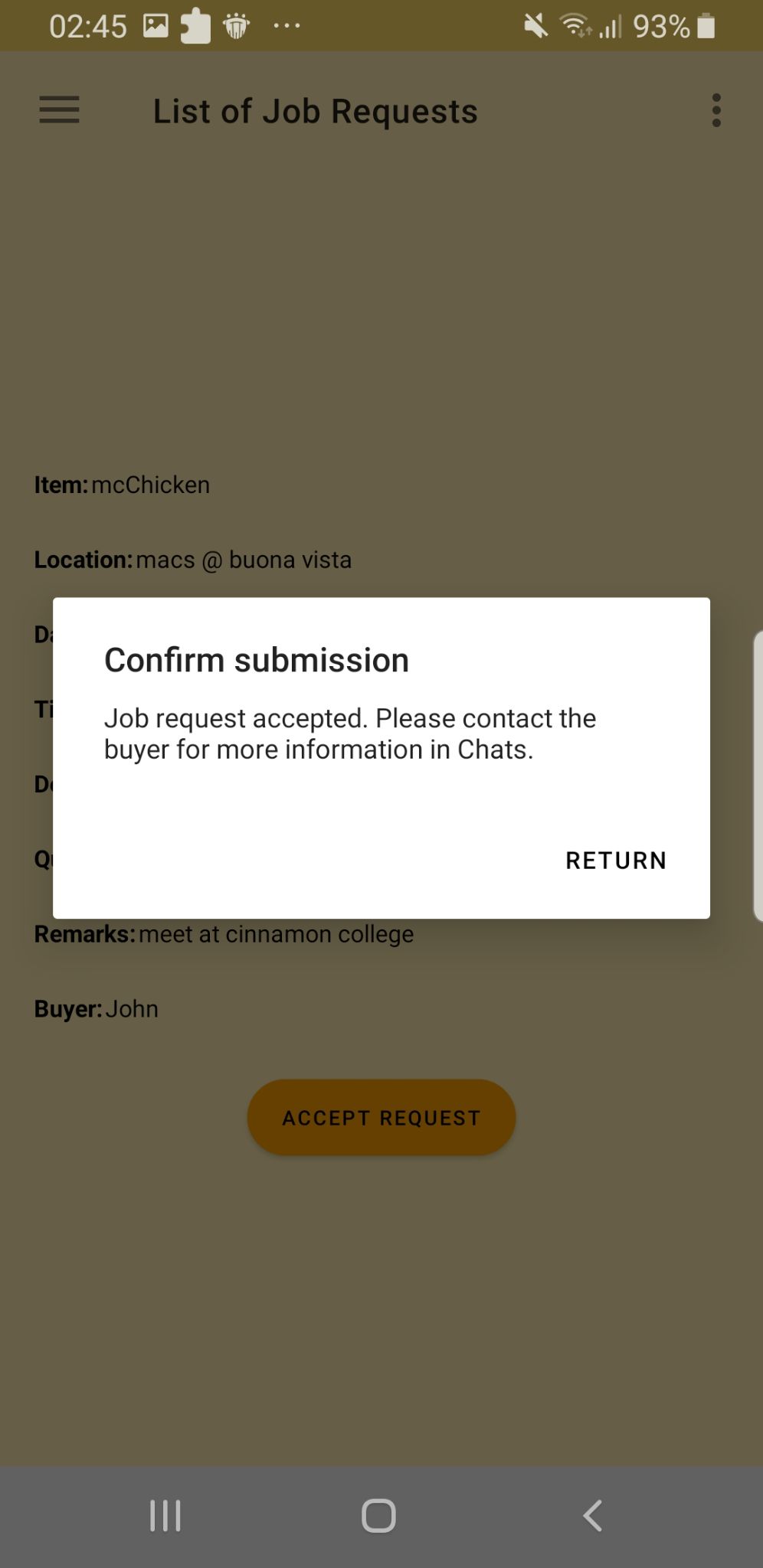
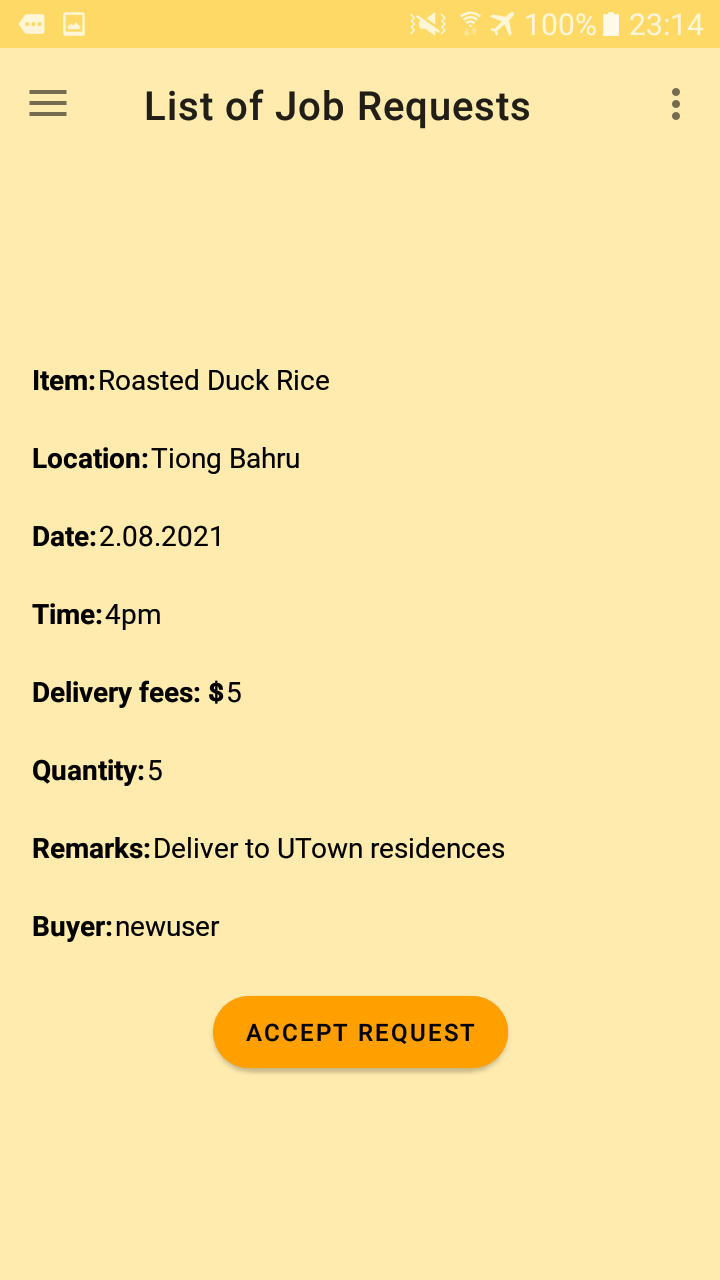
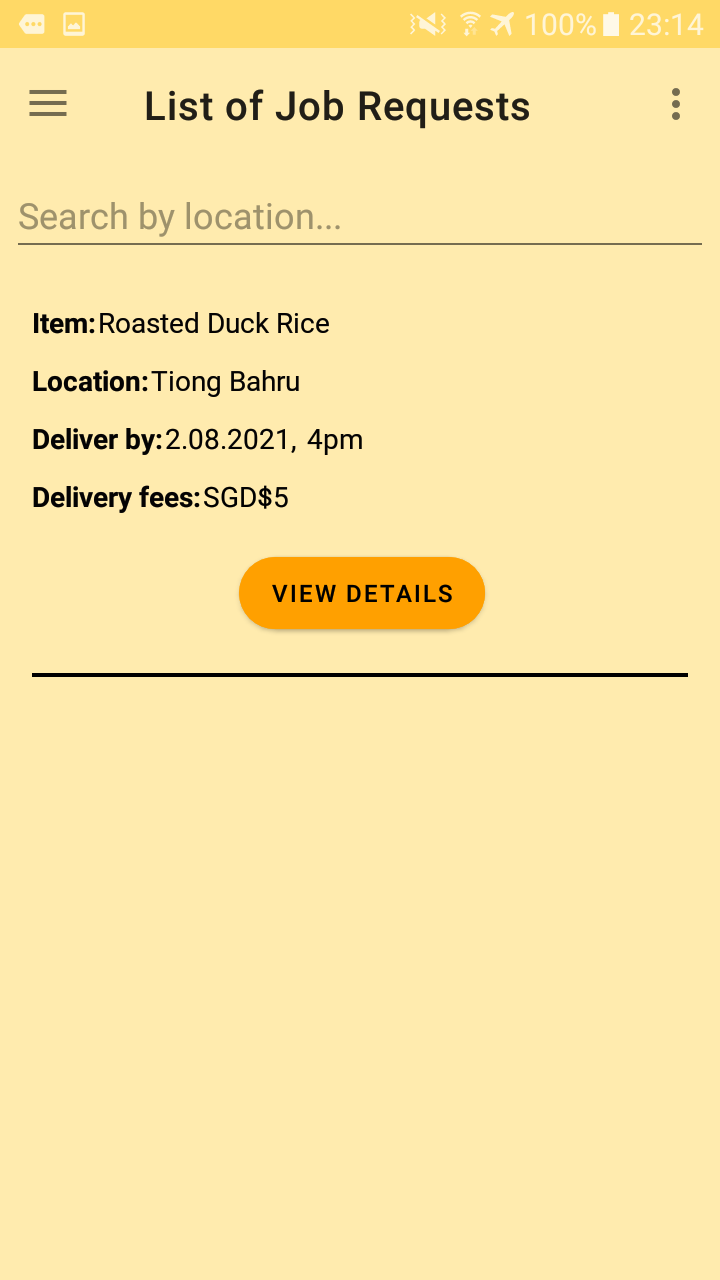


1. List of requests from buyers

**User Role:**  As a deliverer, the List of Requests contains all the job requests (that are not yet accepted) in the database.

**User Goal:** Once the ‘View Details’ button is pressed, there will be a page showing the details of the request and the user to contact if they need more information. By clicking on the ‘Accept Request’ button, they will be prompted to contact the buyer and return to the previous list where the request will be gone.

**User Benefits:** A deliverer will be able to find potential buyers who require him/ her to purchase items for them. The deliverer can also conveniently search for a request based on specific locations.

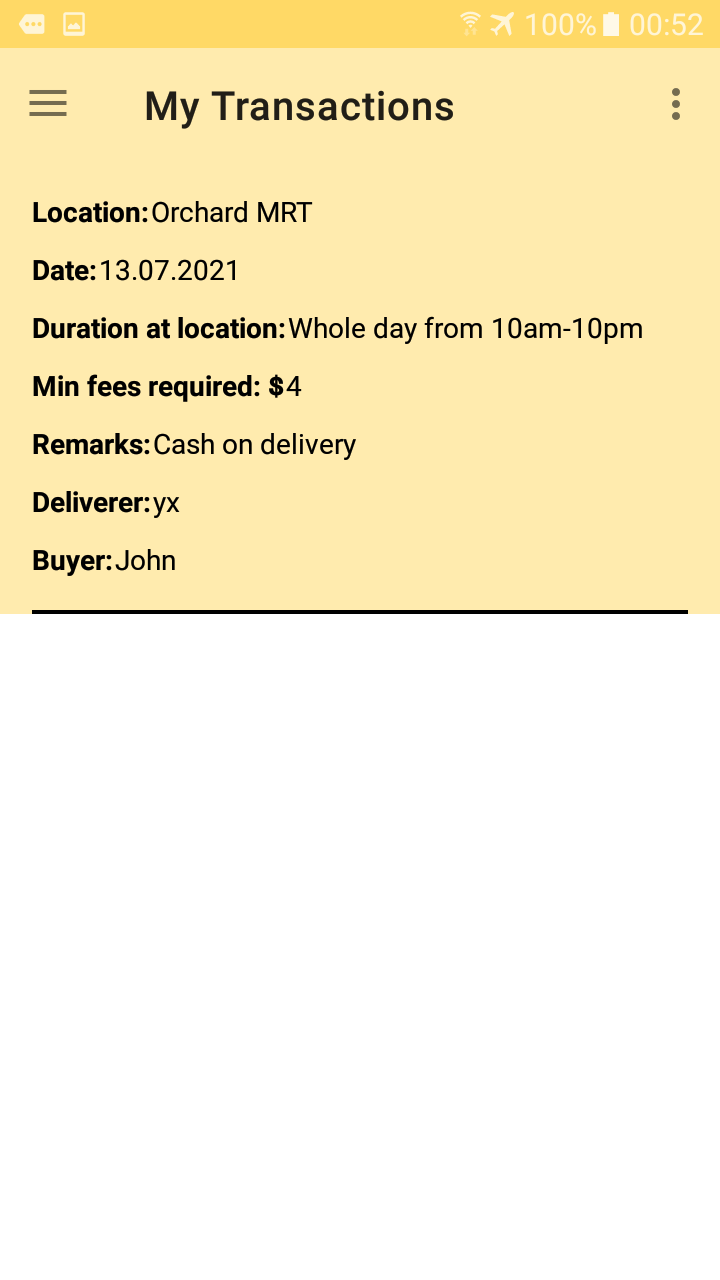
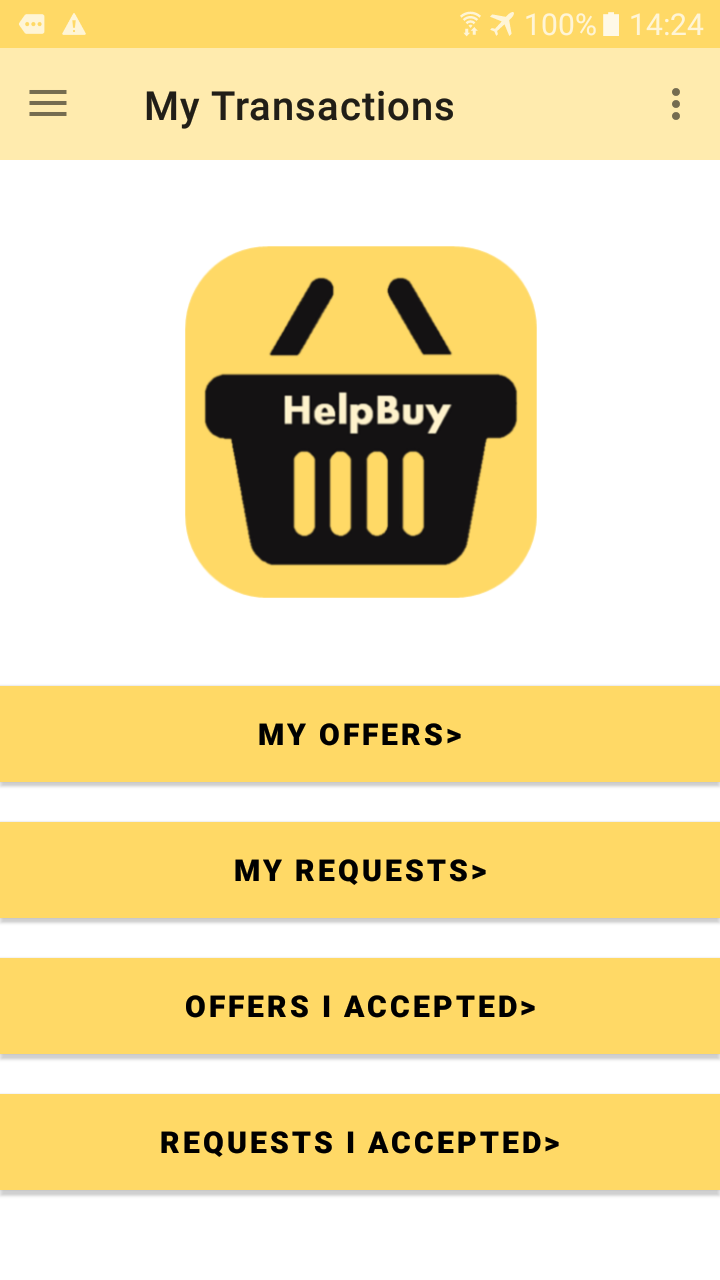


1. My Transactions

**User Role:** The transaction history feature will allow users to check their previous job offers, job requests, as well as those they had accepted from others.

**User Goal:** With the username of the user displayed, buyers/ deliverers would then be able to contact the user through the chat function should they need further information.

**User Benefits:** This would allow users to track the information of their purchases if needed.

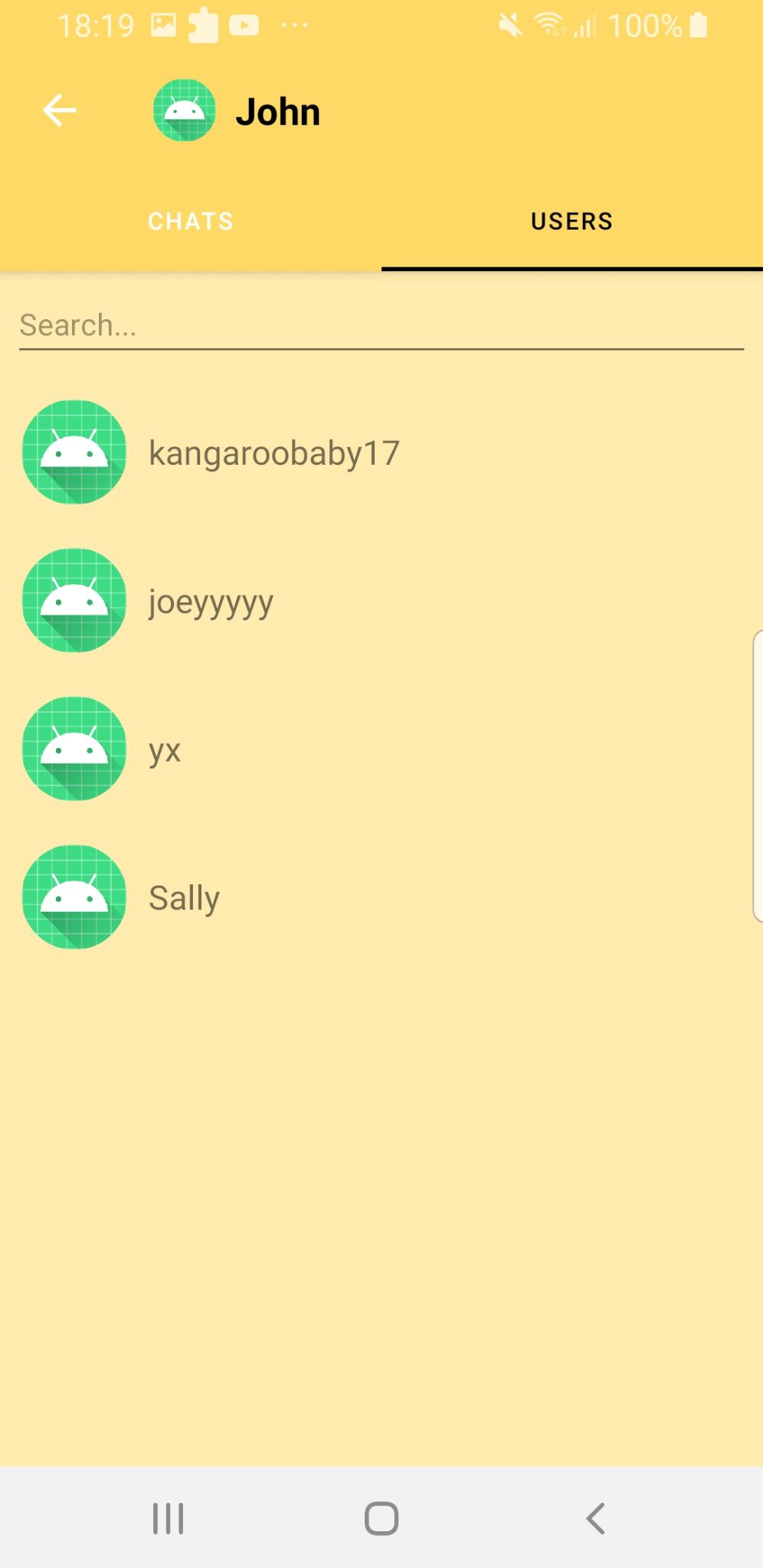
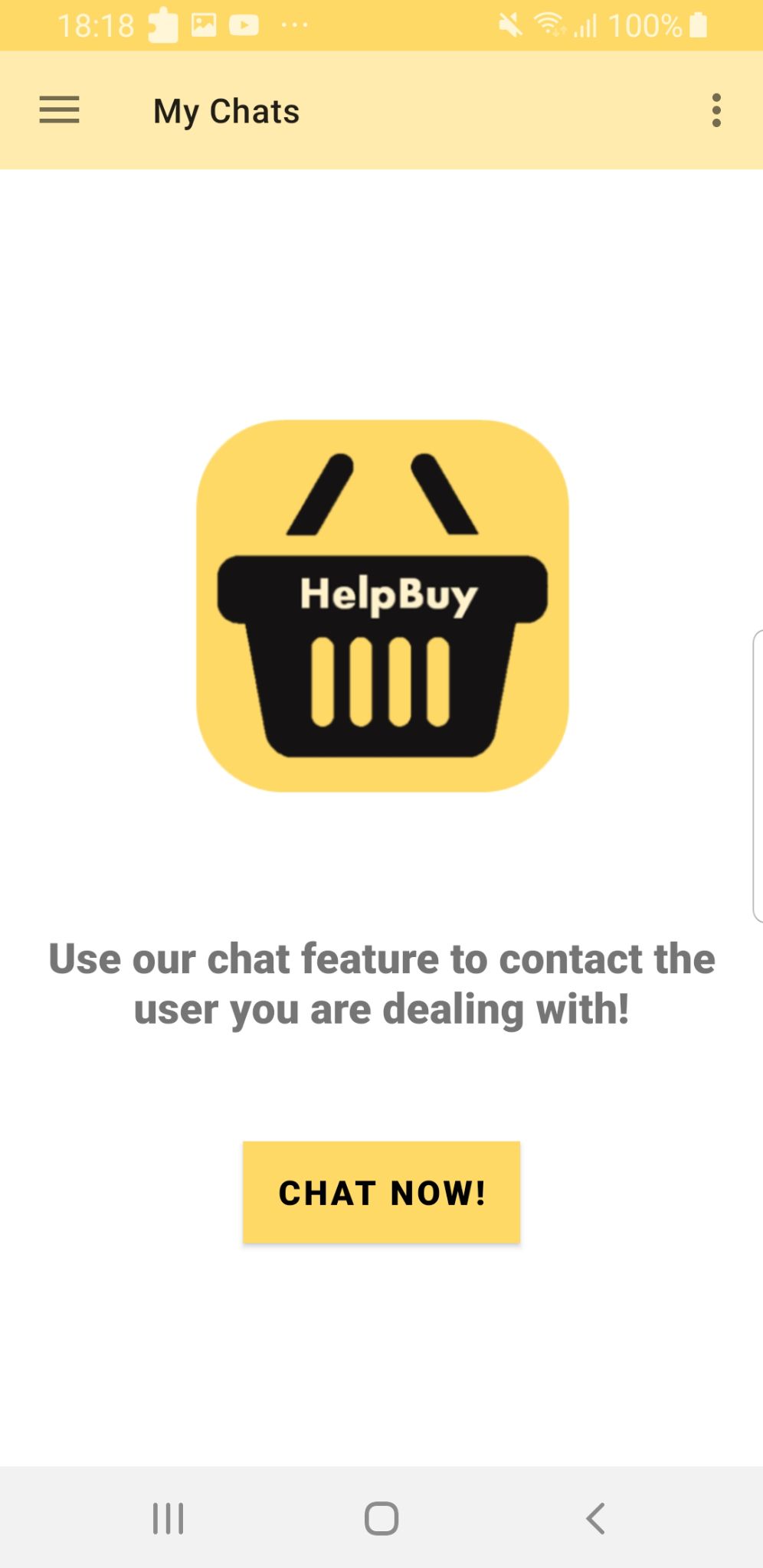


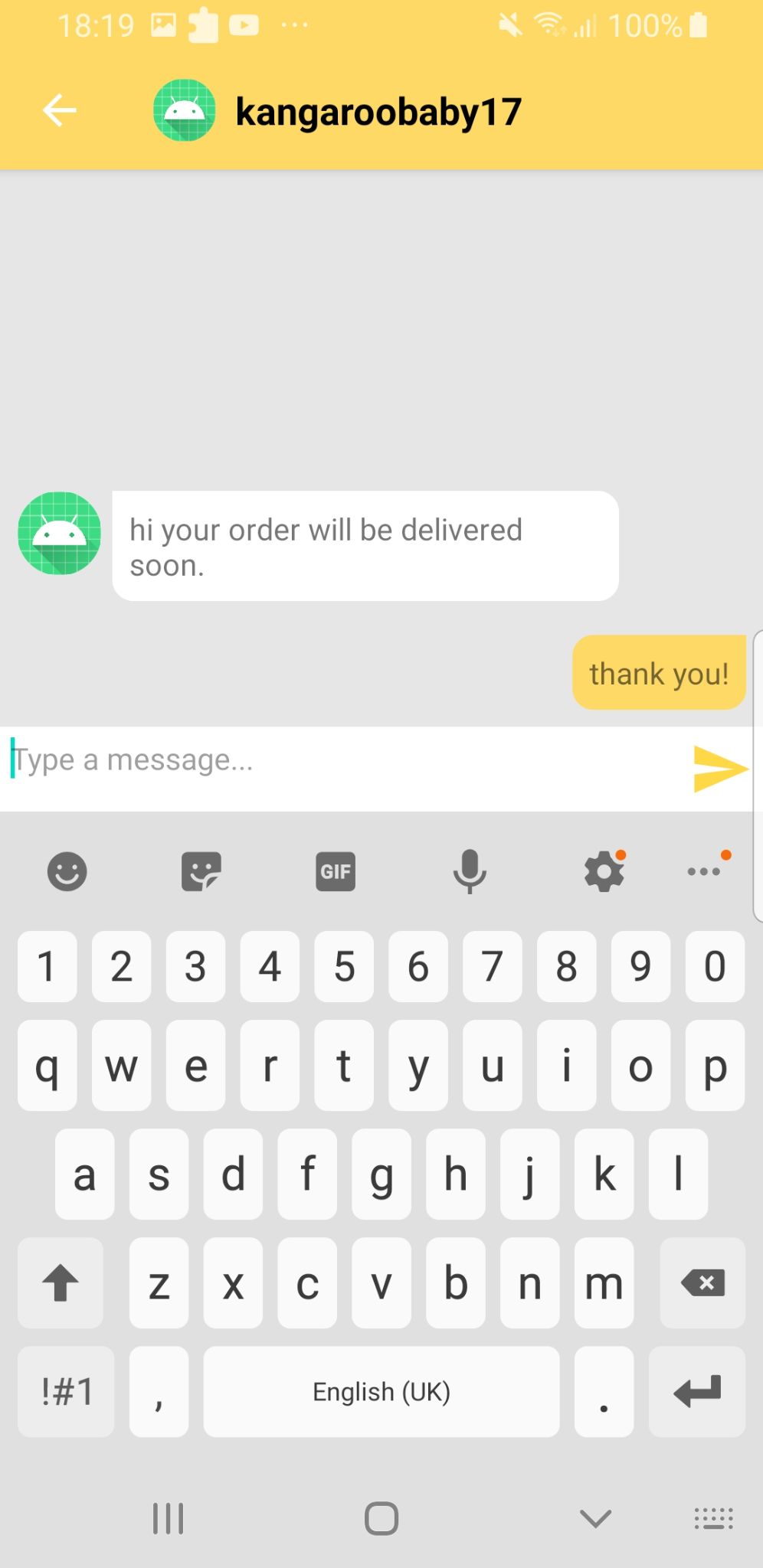
1. My Chats

**User role**: Both the buyer and deliverer will be able to use this feature to communicate with each other.

**User goal**: Users can use the chat function to communicate with the buyer/ deliverer if they need to make any clarifications with regards to their orders.

**Benefits**: This conveniences the users and lowers the chances of making mistakes in the orders.



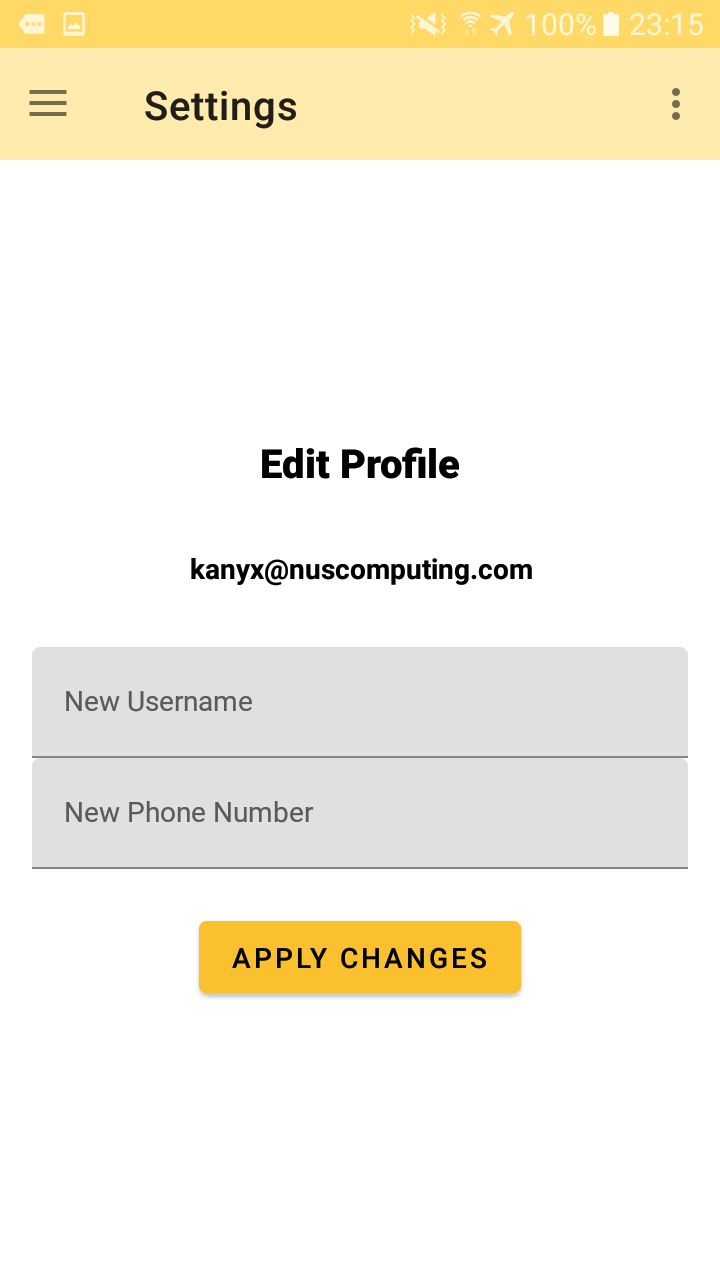
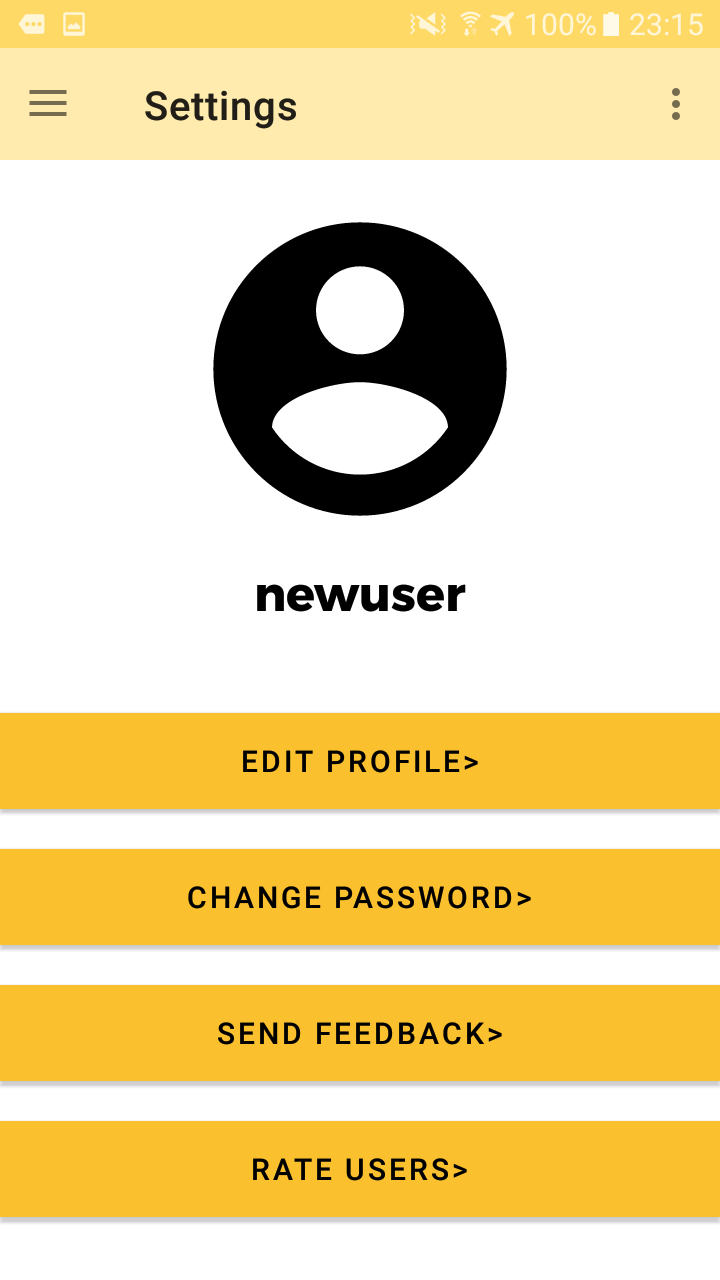


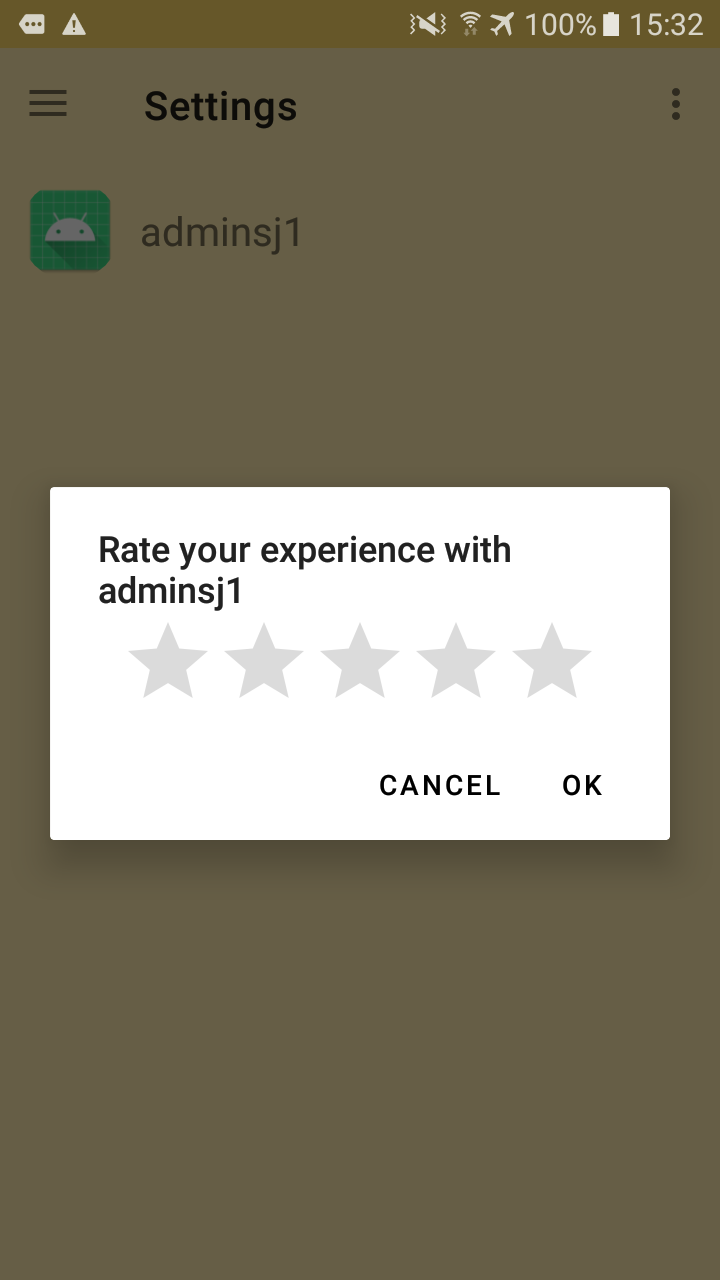
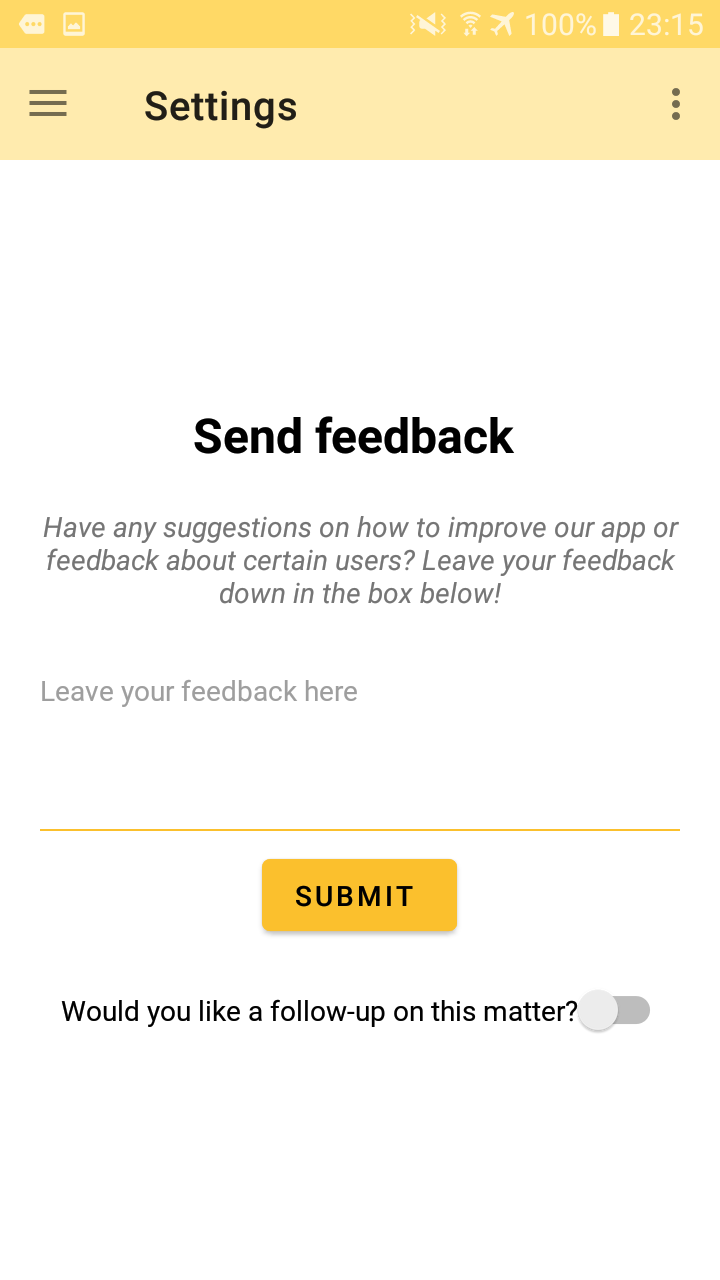
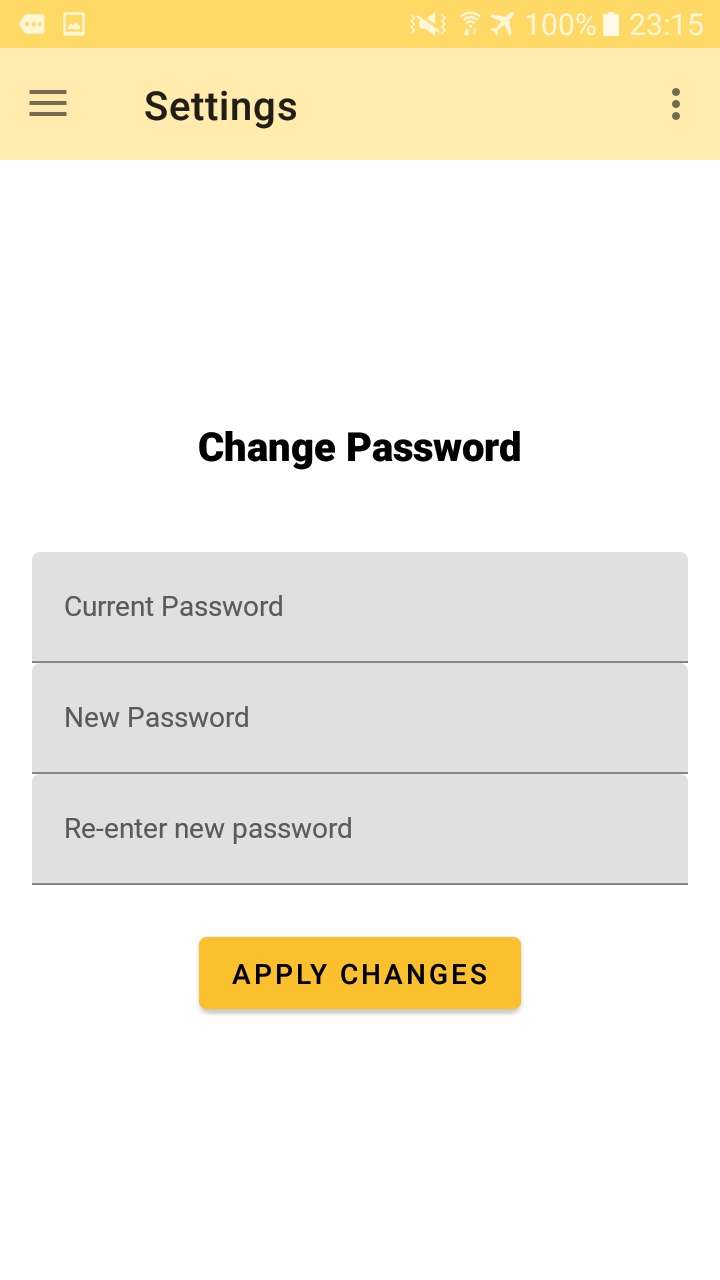
1. Settings

**User role**: Users who wish to make amendments to their profile or send feedback to rate their experience on our platform through the Settings feature.

**User goal**: Users are able to change their password, username as well as phone number recorded in the system. They will also be able to rate the users they have interacted with, either through accepting their request/offer or the other way around to increase user credibility. They will not be able to rate any other user on the platform. Each rating recorded will also override the previous rating they gave for the same user. There will be a feedback system such that users can provide feedback to the administrator if they encounter any problems.

**Benefits**: By giving users an opportunity to define their user experience, this would improve satisfaction and give developers a chance to improve on the features. The edit profile section will also allow users to determine their own user journey.





1. \*\*Enhancing User Experience (UX)

We included the following details in our features to provide a pleasant user experience.

1. For the layout .xml files, instead of having restricted layouts, we added constraint layout. This ensures that across different phone models and screens, the images and words of our application will be scaled accordingly and not be cut off.
2. A consistent colour scheme is used throughout the entire application interface.
3. To make it easier for users to key in details (in add request & offer, etc.), we created a scrollable view.
4. User guides (description of what the feature is for/ how to use the feature) are shown on most screens.
5. Error/ success/ pop-up messages are shown after every submit action to tell users the error/ what to do next.
6. If an action is error-free, the fields which are initially filled with the details which users keyed in will be cleared to indicate a successful action.

# Bug Report

Listed below are the bugs which we encountered. Most of the bugs have been debugged (except for one). As a result, we are unable to reproduce the bugs for those resolved as the error would have been debugged in the backend.

The severity of the bug are rated from 1 to 3 asterisks, with \*\*\* being the most severe.

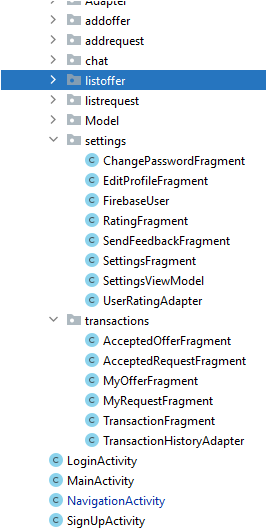
| **Feature** | **Bug** | **Expected behavior** | **How we solved** |
| --- | --- | --- | --- |
| **Chat** | Username is not shown.  *Severity: \*\*\** | When using the search function in chat, the username of the searched user cannot be displayed. An empty box with the default display picture is shown instead. | For every QueryDocumentSnapshot, we need to *getString(“Username”)* the username of the user and then *setUsername()* the user’s username to be that. |
| An empty search bar displays a textbox with no username shown.  *Severity: \*\*\** | After searching and clearing the text in the search bar, instead of showing all the users of the application, the recycler view will show a blank user textbox. | Add *!user.getUsername().equals(“”)* into the if clause under searchUser method. |
| Unable to chat with multiple users.  *Severity: \*\*\** | User A chats with User B and C. Once User B and/ or C replies to User A, the chat feature would crash every time User A tries to enter the *Chat activity*. | A new *Chatlist* class is added. Queries of the user IDs were made from this *Chatlist* class instead of the *Chat* class where it also stored the messages sent among users. |
| The order of the messages sent is jumbled up.  *Severity: \*\*\** | Messages sent before midnight will be deemed as sent later than messages sent after midnight because the date is not taken into account. Eg. Messages sent at 2350 will appear later than messages sent at 0007. | Added date to the ‘*aTime*’ field so that the messages will be ordered according to the date first, then the 24HR time stamp. |
| Chat does not show up immediately  *Severity: \** | When User A selects User B from the *Users* tab and chats with him, the chat does not immediately appear in the *Chats* tab even though it should. For User B to be displayed, User A has to exit the *Chat activity* and enter again. | Unable to resolve. However, it is not a major issue as the user can exit the page and enter again for the *Chats* tab to be refreshed and display the users that the account holder has an active chat with. |
| **Settings** | Clash of usernames  *Severity: \*\** | Users can have the same username in different casings. Eg. ‘John’ and ‘john’ can coexist which makes searching in the chat difficult. | During sign-up, we added one more restriction such that usernames (no matter in what casing) cannot be duplicated. This is done by comparing *name.toLowerCase()* instead of *name*. |
| Need to fill in both fields in editing profile  Severity: \* | If a user wants to edit their profile, they will need to key in both a phone number and a username for the change to be successful. However, they cannot only edit their phone number/username at once. | If-else conditions have been introduced such that the change can still be made even if the other textview is empty. |
| **List of requests and offers** | Text is cut off  *Severity: \*\** | If a user types a long description for their item/ remarks, the screen will cut the text off. | Layout changed to constraint layout such that all the words will be constrained in the same screen. |
| **Layout** | Orientation of the application  *Severity: \*\** | When orientated to horizontal layout, the buttons overlap with each other and the app crashes. | Changed the whole app to only portrait orientation. This is because if the device is turned horizontally, there is a need to create a different layout for that. |

# Software Engineering Practices

## SOFTWARE ENGINEERING PRINCIPLES

* 1. Separation of Concern

While working on the features in our code, we followed the separation of concern principle where the data structure is separated as much as possible to achieve different functions. As seen from the java classes named after different features, it shows that they are all responsible for their own component and one class does not affect the other. Since each file addresses a different concern, this helps us to debug codes easily by identifying the file with the problem and working on that one file instead of having to search through every code. Furthermore, it helps us to work on features more efficiently as we could easily create the basic function for every part first before slowly upgrading and improving on them along the way.



* 1. YAGNI (You Aren’t Gonna Need It)

Our code also follows the YAGNI principle, where we do not include codes that we think may be useful for the future. This prevents complexity in our code and saves money and effort. For example, there are many possible extensions in our app such as enabling location services for users to filter their results by the location they are at, or to enable phone number verification for login. However, we are not sure about the API we need to complete the feature as well as if there are billing involved to use such services, therefore we do not waste time on starting on these designs but work on those that are more practical. All the features in the app are necessary for it to function and provide the most basic user experience, therefore additional codes will only go to waste if we design them.

* 1. Avoid Premature Optimisation

This principle suggests that by optimizing your code in the early stages of development, it is easy to waste time and effort by always going back to the code to test it. Therefore, in the earlier stages of development (Phase 1 and 2), we used simple and effective ways to complete features, though it may not be the most optimal. We only started to clean up the code in the last phase of the development, by combining similar if-else conditions, optimising long or unused codes etc. If we had done it earlier in the development process, we would have wasted the time that we could have used to improve on other features.

## IMPROVEMENT FROM MILESTONE 2

Based on the feedback we received from our advisor and other teams in Milestone 2’s peer evaluation, we addressed the key issues and improved on our application.

| **Feedback** | **Follow-up** | **Elaborations** |
| --- | --- | --- |
| “You can view your own job request/ offer in the corresponding offer/ request pages which allows users to accept their own request.” | Incorporated | Users can still see their requests/ offers in the lists, but now they are unable to accept their own requests/ offers. An error message will be shown when users attempt to accept. |
| “Chat crashes on relogging with different accounts on the same device.” | Incorporated | Debugged this error (further elaborations in bug report). |
| “Chat seems to work fine but would like to see chatting being limited to the people interacting via the job contract they have as chatting to people beyond that seems pointless.” | Not incorporated | We think that users should have the freedom to chat with others, even if the other user is not involved in any transaction (yet). This is to allow users to be able to make enquiries about a request/ offer before deciding whether to accept them or not. |
| “You might want a feature that allows the user to sort/search by location or dates.” | Incorporated | Under list of requests and list of offers, users are now able to search by location to find their most suited request/ offer. |
| “Typically for such app, with 2 user groups, you'd want 1 of the groups to be the active user and the other group to be the passive user, rather than both user groups being the passive users.” | Not incorporated | As this feedback involves a fundamental change to the workings of our application, we decided to not make the change given the short timeframe given for Milestone 3. However, we acknowledge that it is better for an application like ours to have a passive and active user instead of both target audiences being active users. Hence we decided to incorporate this change in our future developments for this project (further elaborations in future plans section). |

## TESTING OF APPLICATION

For deployment procedure, we tested the prototype on an Android phone by following the steps below:

1) Setting up your android phone

1. Download Android Studio or IntelliJ on the computer
2. Connect the computer to an Android phone with a USB
3. On the phone, open the Settings app. If your device uses Android v8.0 or higher, select System. Otherwise, proceed to the next step.
4. Scroll to the bottom and select About phone.
5. Scroll to the bottom and tap Build number seven times.
6. Return to the previous screen, scroll to the bottom, and tap Developer options.
7. In the Developer options window, scroll down to find and enable USB debugging
8. On the computer, click on the “play” icon at the top right hand side in Android Studio to launch the application.

2) Linking to the HelpBuy application

1. Clone files from <https://github.com/joeyylow/HelpBuy.git> . Make sure you are in the *master* (default) branch.
2. In Android Studio, go to File>Open>Navigate to the directory where it was downloaded into > Select directory.
3. Ensure that your android phone is connected and the device model is shown on the screen.
   * Note: If you do not have an android phone, you may also choose to run the program using the emulator in-built in android studio. The emulator can be selected from the AVD Manager.
   * Note that the android phone should be connected to the internet
4. Run the project by clicking on the green play button at the top of your window as shown below.



### System Testing Plan

* 1. Self Evaluation
* Throughout the whole development process, we made sure to constantly test the application for every feature we developed. We strove to put ourselves in the shoes of the users and tried our best to evaluate whether our application provides a good user interface and user experience to our users.
* Once we are almost complete with a feature, we would push our codes to our own branch in GitHub so that our partner can test out the feature. **[Unit Testing]**
* We would then ask each other for feedback before working on improving our codes and finalising them.
* Once each feature is finalised, we would push the codes to the *master* branch on GitHub. The master branch would have the codes to all the features we both worked on.
* We then test out the application as a whole to see if the features interact well with each other. **[Integration Testing]**
  1. Role Playing **[Scripted Testing]**
* We conducted scripted testing on our completed system to evaluate our application’s compliance with its expected performance.
* Our mobile application is unique in the way that it allows users to be both a deliverer and a buyer all in a single account.
* This means that we must constantly ensure that we think from the perspectives of both deliverer and buyer.
* However this can be confusing at times during the development phase as we are concurrently developing features for both deliverers and buyers.
* To aid our testing process, both of us each took a role.

Yu Xuan: Deliverer; Si Jie: Buyer

We then role played our respective roles and did a walkthrough of our application from the very first step (registering for an account) to the last step in a transaction (accepting an offer/ request).

* Throughout the testing process, we followed the user stories which we planned out in Milestone 1 and made sure that we followed all the requirements closely.
  + Listed in the table below are some of the test scenarios we used.
  + \*Note that this list is not exhaustive. We conducted many other tests but are unable to list them in this table as those tests did not require users to have text ‘inputs’. As a result we cannot display the test input. Eg. Clicking on buttons to see if they work.

| **Devised Test Cases** | **Test Input** | **Expected Result** |
| --- | --- | --- |
| Valid email verification  Users should only be able to register with a valid email. | sally@gmail.com | Positive |
| sally | Negative |
| sally@fmail.com | Negative |
| Valid password  Users need to key in a text with at least 6 characters.  The text in ‘*Confirm Password*’ must also match with *‘Password’*. | 123456  123456 | Positive |
| 1b3  1b3 | Negative |
| 123456  adifferentpassword | Negative |
| All fields have to be filled up  Users have to fill up all the fields with relevant details when adding a job request or offer. | All fields filled up | Positive |
| Presence of an empty field | Negative |

### User Testing

1. Usability Testing with Potential Users

* As the application’s developers ourselves, we understand that we might overlook certain problems that potential users of our application may face.
* Therefore, after developing the main basic features of our application, we asked our friends to test out our working prototype.
* Among all the friends we approached, a few of them are also part of Orbital. This allows us to not just receive general feedback, but also technical feedback from those who have experiences in mobile/ web development.
* Feedback was asked informally throughout the whole process of our application development.

1. Survey of Potential Users

* We conducted a formal survey with the target group being NUS staff and students (the target audience of HelpBuy).
* This is the [survey form](https://forms.gle/ahMDDqsNXNKLmG9H9) which we used.
* Each question is meant to help us understand the usability of each feature we implemented. Those surveyed can also give additional comments on how to improve the features if they have any.
* Shown below are the survey results which we obtained, along with our analysis of the results.

***Survey Results & Analysis***

| **Profiling** | **What is your profile?** | |
| --- | --- | --- |
| Forms response chart. Question title: What is your profile?. Number of responses: 6 responses. | |
| As the target audience of the HelpBuy application are NUS staff and students, our survey was open to the two groups. However, the results showed that only NUS students were surveyed. Problems which NUS staff may face when using the app may therefore not be known to us. We aim to get NUS staff to test out our app in future for more comprehensive and unbiased feedback. | |
| **Signup Feature** | **Rate your signing up experience.** | **Any problems faced/ areas to improve/ additional comments?** |
|  | * Verification of email was good as it provides an additional layer of security. * Phone number is not verified, can key in any number. |
| We did not enable verification of phone numbers as we already have email verification, and we think that having both verifications will make signing up a hassle for new users. Sending a one-time link via email for verification would be sufficient to ensure accountability and credibility. | |
| **Login Feature** | **Rate your logging in experience.** | **Any problems faced/ areas to improve/ additional comments?** |
|  | NIL |
| Our login interface satisfies all users surveyed and it is good to go. | |
| **Main Page** | **Rate your experience with the main page.** | **Any problems faced/ areas to improve/ additional comments?** |
|  | * Don’t have to show main page if the user is already logged in. |
| We agree that it will be more convenient for users if auto-login is enabled. In that case, users will not need to log in every time they open the app. There is also no need to display the main page if auto-login is enabled. Therefore for future versions of the app, if we implement auto-login, we will remove the main page accordingly. | |
| **Add Offer Feature** | **Rate your adding offer experience.** | **Any problems faced/ areas to improve/ additional comments?** |
|  | * Easy to key in details * Prevent users from adding random addresses which do not exist. |
| To improve user experience when adding locations, we will add a location API in future versions of our application if we intend to scale it. This ensures that the locations added are legitimate. We did not include the location API in this milestone as we are required to pay for its use. | |
| **Add Request Feature** | **Rate your adding request experience.** | **Any problems faced/ areas to improve/ additional comments?** |
|  | * May want to input a calendar to choose the date instead of manually keying in. * Maybe can allow users to add images of the items which they want to purchase. * May need to make it mandatory for users to key in the location they want the item to be delivered to. |
| Adding the calendar feature is a good idea as it standardises the format of the date entered.  We will also allow users to add images to their offers and requests in future as an extension to our application. This would make the offers and requests clearer and more specific.  Currently, the meet-up location is not a required field to be filled up as both parties can easily decide on a location (which is somewhere in NUS) via the chat feature. However, to further convenient users, we may consider adding this field in. Any changes to the initial location suggested by buyers/ deliverers can be negotiated in chats. | |
| **List of Offers Feature** | **Rate your experience with viewing the list of offers.** | **Any problems faced/ areas to improve/ additional comments?** |
|  | * Search function was very useful in helping to find a request/ offer near me. |
| Searching the offers/ requests by location is a newly added feature which we introduced in Milestone 3. It allows users to search for offers/ requests that are near them. | |
| **List of Requests Feature** | **Rate your experience with viewing the list of requests.** | **Any problems faced/ areas to improve/ additional comments?** |
|  | * Display images of the items that the user wants to purchase. |
| In the next version of our application, we will be allowing users to add images to their requests/ offers. | |
| **My Transactions Feature** | **Rate your experience with viewing your transactions.** | **Any problems faced/ areas to improve/ additional comments?** |
|  | * Can include timestamp of when the offers/ requests were created. |
| Adding timestamps would allow users to track when the request/ offer was created. We would add this feature in the future. | |
| **My Chats Feature** | **Rate your experience with using Chats.** | **Any problems faced/ areas to improve/ additional comments?** |
|  | * Chats does not show up immediately under chats tab. * Display the last message sent for each chat. * Perhaps can enable chat notification. |
| Chats not showing up immediately in the *chats* tab is a bug which we are unable to resolve (further elaborated in bug report).  For milestone 3, we only intended to create a basic chat feature which allows users to communicate with each other. For better user experience, we will consider showing the last message that users sent to each other, and also allow notifications to be sent to the users’ phones once they receive a message. | |
| **Settings Feature** | **Rate your experience with using Settings.** | **Any problems faced/ areas to improve/ additional comments?** |
|  | * Profile picture seems to be redundant. * Can allow users to change their profile pictures. * Can change to any unverified mobile number. |
| In future versions, we would be allowing users to change their profile pictures.  Similar to the analysis for signing up, change of phone numbers will only be possible if the phone numbers have been verified to be legitimate. | |

# Problems Encountered & Solutions

| **Problems faced during implementation** | **Solutions** | **Justification on approach** |
| --- | --- | --- |
| **Technical** | | |
| Each of us implemented different features on our own computers at different times of the project. For a comprehensive testing of our whole app, we had to merge all our codes together. This is a very tedious process if the merging was done manually. | We created a repository on GitHub and learnt how to leverage on the collaborative feature of it to aid in the sharing and merging of our codes. | This made it easier to track any changes in codes and merge any time without needing to copy and paste huge chunks of code. Github also tracks previous versions of our codes which makes it easy for us to retrieve them easily when we need our old codes for debugging. |
| We had no prior experience in using Android Studio and Firebase, so we faced many difficulties when trying to figure out how to use these softwares from scratch. | We actively searched for source codes and watched video tutorials. We also read up on the Firebase documents. Based on what we learnt online, we modified and applied them accordingly to fit our project requirements. | By learning the coding logic used in the source codes we found online, we managed to understand Android Studio and the tech stack that we are using. After several weeks, we became more familiar with the softwares and are less reliant on source codes. |
| **App Specific** | | |
| Our app users can be both a buyer and a deliverer which makes navigation difficult. | We each took a role of buyer and deliverer so that it would be easier to test the different functions. | Since one user can take up either 2 roles when using the platform, taking up one role when developing the app can help to debug and simulate the actual process. |

# Future Plans

As an extension to our project, the following features could be implemented to enrich the user’s experience and enhance usability:

1. **Location Services**

* Users will be able to share their location with other users so as to filter their job requests and offers by those that are close to them. A user that is in UTown would be able to view the jobs that are nearer to them at the top of the list.
* Besides using location filtering, we can also introduce Google’s Place autocomplete feature to allow users to enter the location instead of manually keying in. This allows more accuracy in the details and specifies proper places instead of leaving ambiguity. (Eg. Woodlands)

1. **Real-time update of delivery status using location**

* Users will be able to track if their item has arrived automatically without the deliverer/ buyer needing to communicate through chat.
* Similar to delivery platforms like Grab, users will be able to see the journey and estimated time of arrival for their parcels.
* Once the deliverer arrives at the location, an automated message will be sent to the buyer to inform them about the arrival.

1. **Notification sent to phone when there is a message received**

* The current chat feature requires users to enter the chat with another user to view the messages. As an extension, we intend to include phone notifications when a message is sent.

1. **Notification sent when another user accepted my request/ offer**

* Similarly, a notification will also be sent to the user when someone has accepted a request or offer they have put up, so that they would not need to actively enter ‘My Transactions’ to check if it has been done so. When a buyer/ deliverer accepts a job, they will need to go to the chat function to initiate the discussion with the other party, making them the active user in the user journey. The user who put up the offer/request will then be the passive user waiting for someone to accept it, therefore a notification will enhance the user experience.