# **WithU**

# **Requirement and Specification Document**

### ***2017*-*09*-*29*, version 1.0**

**Team:** Sam Patterson, Joshua Chaimson, Pritesh Kalantri, Will Roberts, Mingren Shen, Shantanu Singhal, Jason Waner

\*Revisions are marked as highlighted\*

## **Project Abstract**

For this project, our team has decided to develop an Android application that will substitute SAFEwalk. The overall goal for this project is to simplify the process of requesting someone to walk with. The current process to request walkers from SAFEwalk is outdated and requires you to send information over text message or a phone call. This slow process has turned some users away from the SAFEwalk program. Initially, this application will branch off into two separate functionalities, one for the users and one for the walkers. On the user side, the user will be able to create an account, locate where they are and their destination, and then simply request a walker. On the walker side, any person that will have a walker account would be notified of the request, accept or decline the request, and communicate between other walkers or clients.

We will be using Android Studio as our IDE, and other resources including Firebase, and the Google Maps API. We will build the application using Java.

## **Document Revision History**

Rev. 1.0 <2017-09-29>: initial version

Rev. 2.0 <2017-11-01>: iteration 1 revision

Rev.3.0 <2017-27-2017>: iteration 2 revision

[**WithU**](#_cg052c6duci) **1**

[**Requirement and Specification Document**](#_u3jxivs57yqj) **1**

[2017-09-29, version 1.0](#_8a4lt8w1i4hz) 1

[Project Abstract](#_ff1pynci250) 1

[Document Revision History](#_uezqih1et91l) 1

[General](#_r2w9znl3nf6i) 3

[Customer](#_txrjmubsfmtx) 3

[Competitive Landscape](#_s652xpbzr5hg) 3

[User Requirements](#_eq47clfh3a3k) 4

[Use Cases](#_1i8bgi1ptkdl) 6

[Security Requirements](#_wthw0hgrwuot) 20

[System Requirements](#_gik1v862o8a0) 20

[Back-end](#_kyo7a2vjmvzd) 20

[Front-end](#_3yzad14ckz3v) 20

[Version control](#_y3mk5vnm6syp) 20

[Specification](#_yhoumybjhj6n) 21

## **General**

This project’s scope is definitely big enough as there are many features we are incorporating in the application itself as well as the majority of the team members will be learning a software they have never previously used. This learning curve along with the amount of features we wish to implement justify the project’s scope being big enough for a team of seven people while at the same time not being too large where we would be unable to complete the application within one semester. We do believe that all of the parts of this application should be able to implement within the allotted timeframe. That being said we do have several other key features in mind that may get implemented as time allows but as of now we tried to focus on the core features to ensure not failing to implement features listed here in the end.

## 

## **Customer**

The customer for our product is anyone who uses the SAFEwalk system, but we are primarily focusing on females. Our main point of contact and our client for this application is the Pi Beta Phi sorority. Having this sorority as our contact will give us insight about how a system like SAFEwalk works and how we can improve our product for future use. Communicating with our client early on in the process is going to help us figure out what requirements we need to meet to make a successful application that will improve on the SAFEwalk process. Some improvements our client wants to make to the original SAFEwalk process are the ability to select where there destination is on a map, simplify the process of requesting a pickup, keep the process as safe as possible, and have the ability to see a map with the real time location of the walkers.

## **Competitive Landscape**

WithU is an application that will provide safe transportation for users, but it is not the only application on the market that provides a safe way to get from point A to point B. Some categories that our application can fit into are transportation and safety applications. The biggest competitors in the transportation category are Uber, Lyft, Cabify, and Gett. The difference between our application and these applications is that these transportation application deal with picking up users in a car. The usage of motor vehicle transportation will create competition because being picked up in a car appeals to a lot of users. Weather and speed of transportation are the main reasons why users would use these applications over WithU. Applications Uber and Lyft have a big competitive advantage over WithU because of how much money they are able to generate. They are then able to create large marketing campaigns to promote their app, and gain a larger market share. Even though they have a multiple advantages over WIthU, their biggest weakness is that they cost the user money while WithU is an application that deals volunteers. This allows the user to use the application for free without having to input any payment methods.

These transportation applications also set the standard for a convenient and easy way to show your location and request a pick up. They did this by creating a clean and simple user interface. This is an aspect we want to implement in our application. We want the process of requesting a walker to be second nature for the user. Another feature we will need to implement is to be able to show where the walkers are in real time. This feature shown in ridesharing applications gives users the peace of mind because it shows the user that their request worked. Not only do these transportation applications provide competition, they also represent standards we will have to meet and exceed if we want our application to compete in this crowded marketplace.

Another area our application is associated with is safety. One of the more popular safety applications is called Safetrek. Safetrek is an application for users who do not feel safe and want a fast and efficient way for the authorities to be contacted. It does this by making the user keep their finger on the screen until they have made it to their desired location. If the user doesn’t feel safe, they would lift their finger and in ten seconds the authorities would be notified and sent to the user location. Their biggest weaknesses are that a user has to pay a subscription to use the service, and it doesn’t allow for the user to request another person to walk with. Using Safetrek as inspiration we want WithU to be able to request/call the authorities to the user/walker location in a quick and efficient matter.

## **User Requirements**

There will be a total of two different users for the application; one being the walkers and the other being a user requesting a walker.

**1. User opens the WithU application**

1. There is two separate buttons the user can press, either register or sign-in (if they are a previous user)

**2. Sign Up Page**

1. Fields:
   1. First Name
   2. Last Name
   3. Gender
   4. Email
   5. Phone Number
   6. Password
   7. Re-enter Password
2. Buttons:
   1. Register: Finalizes account with the provided information

**3. Login Page**

1. Fields:
   1. E-Mail
   2. Password
   3. Remember Me - Checkbox
2. Buttons:
   1. Login: Searches database for user’s information
   2. Forgot Password
   3. Register for Account: Brings user to sign up page

**4. Password Recovery Page**

1. Fields:
   1. Email
2. Buttons:
   1. Submit

**5. Home Page (Client Side)**

1. Map displaying the user’s location
2. Buttons:
   1. Request Walk: Sends request to walkers
      1. Submit: Finalizes request to walker
   2. “Where to” Search Bar
      1. Allows the user to be able to select their destination
   3. Pull out side window:
      1. Your trip history: Displays the walking history by dates
      2. Switch to Walker: Switches from client profile to walker profile
      3. Help: Simple tutorial of how to use the application
      4. Settings: Account information that the user can update
      5. Report: Allows user to report a walker
      6. Emergency: Directs user to phone application to dial 911
      7. Logout: Logout of the application and close it
      8. Terms of Use : clauses that embody the rules, requirements, restrictions and limitations that a walker must agree to in order to use our mobile app.

**5a.Waiting Screen (Client Side)**

1. Map displaying where the current walkers are
2. Notification Bar on top of the map
   1. Gives the names of both walkers, and their ages
   2. Phone button: for quick access to call the walkers
3. Let’s Go Button
   1. Dismiss: close the window
   2. Call: calls the walker
   3. Navigate: takes you to google maps for turn by turn naivgation
4. Cancel Button
   1. Terminate the walk with a yes or no selection

**6. Home Page (Walker Side)**

1. Map displaying the user’s location as well as nearby walker locations
2. Buttons:
   1. Destination: allows walker to enter desired destination
   2. Emergency: contacts 911
   3. Find Partner
      1. Two main options to pick but always paired with opposite sex
         1. Partner chosen based off closest unpaired walker
         2. Walker can select partner from list of unpaired walkers
   4. Pull out side window:
      1. Your trip history: Displays the walking history by dates
      2. Unpair: Allows walker to unpair with current partner
      3. Switch to User: Switches from walker profile to client profile
      4. Help: Simple tutorial of how to use the application
      5. Settings: Account information that the walker can update
      6. Emergency: Directs walker to the phone applications to dial 911
      7. Logout: Logout of the application and close it
      8. Terms of Use : clauses that embody the rules, requirements, restrictions and limitations that a walker must agree to in order to use our mobile app.

## **Use Cases**

|  |  |
| --- | --- |
| Name | Sign Up **(must have)** |
| Actor | User/Walker |
| Triggers | Selects ‘Register for Account’ button |
| Events | User enters in the required fields   1. First Name 2. Last Name 3. Gender 4. Email 5. Phone Number 6. Username 7. Password 8. Re-enter Password |
| Exit Condition | User enters an email in the correct format, and enters the password that meets the applications specifications.  The user then selects the ‘Register’ Button |
| Post- Conditions | The client is then sent to the WithU ‘Home Page(Client Side)’ |
| Acceptance Test | The client should be able to register for a new account and be taken to the application ‘Home Page(Client Side)’ |

|  |  |
| --- | --- |
| Name | Log In **(must have)** |
| Actor | User/Walker |
| Triggers | Open the WithU application |
| Events | User enters in the required fields   1. Email 2. Password |
| Exit Condition | User enters an email in the correct format, and enters the password that meets the applications specifications.  The user then selects the ‘Login’ button |
| Post- Conditions | The client is then sent to the WithU ‘Home Page(Client Side)’ |
| Acceptance Test | The client should be able to sign in using their information and be taken to the application ‘Home Page(Client Side)’ |

|  |  |
| --- | --- |
| Name | Logout **(useful)** |
| Actor | User/Walker |
| Triggers | Pull out the side window and select “Account Info” |
| Events | The user is then taken to the ‘Account Info’ page |
| Exit Condition | User clicks on the ‘Logout’ Button |
| Post- Conditions | The user is signed out and taken to the ‘Login Page’ |
| Acceptance Test | The user should be able to logout of the application and then taken to the ‘Login Page’ |

|  |  |
| --- | --- |
| Name | Switch from either User to Walker or Walker to User **(must have)** |
| Actor | User/Walker |
| Triggers | Pull out the side window and select “Switch to Walker” or “Switch to User” |
| Events | The user will click on the ‘switch’ to move from User to Walker or Walker to User |
| Exit Condition | The user then selects ‘yes’ to confirm they want to switch |
| Post- Conditions | The user is either taken to the User ‘Home Page’ or the walker ‘Home Page’ |
| Acceptance Test | The user should be able to switch from the user side of the application to the walker side of the application. |

|  |  |
| --- | --- |
| Name | Request Walk **(must have)** |
| Actor | User |
| Triggers | User clicks on the “Where to” search bar |
| Events | User Selects destination or search’s their destination using the google places API. |
| Exit Condition | User clicks the Request button |
| Post- Conditions | The user is shown a “Requesting..” screen until walkers are found/accept |
| Acceptance Test | The user should be able to request a walk and be assigned walkers. |

|  |  |
| --- | --- |
| Name | Accept Walk Request **(must have)** |
| Actor | Walker |
| Triggers | User clicks on the “Accept” Button |
| Events | Walker is assigned to that walk and given directions to user’s location |
| Exit Condition | User clicks the Accept Walk Button |
| Post- Conditions | Directions to requestor seen by walker while profile of Walker sent to user requesting as well as ETA (Estimated Time of Arrival) Walker marked as on walk and no longer considered for other walks until walk completed. |
| Acceptance Test | The walker should be able to accept a request walk and given directions to user in need. |

|  |  |
| --- | --- |
| Name | Decline Walk Request **(must have)** |
| Actor | Walker |
| Triggers | User clicks on the Decline Walk Button |
| Events | Walker is no longer considered for that walk and a new nearest walker is notified |
| Exit Condition | User clicks the Decline Walk Button |
| Post- Conditions | Walker taken back to the map on main page |
| Acceptance Test | The walker should be able to decline a requested walk and taken to main map. |

|  |  |
| --- | --- |
| Name | Emergency Button **(must have)** |
| Actor | User/Walker |
| Triggers | User clicks on the “Emergency” Button |
| Events | Confirmation popup is displayed and if yes 911 is called |
| Exit Condition | User clicks to end call or clicks no |
| Post- Conditions | User is returned to their respective main page |
| Acceptance Test | The user/walker is able to call 911 upon pressing the emergency button and able to back out of it if emergency button is accidentally pressed. |

|  |  |
| --- | --- |
| Name | Rate Walkers **(Useful)** |
| Actor | User |
| Triggers | Once walker has finished the walk (Clicked “Yes” on the Finish Walk Notification) |
| Events | Rating system shows on the screen, allowing the user to rate their walk experience |
| Exit Condition | User rates the walker or chooses to decline to rate |
| Post- Conditions | User taken back to main page, rating is updated for the walker |
| Acceptance Test | The user should be able to rate a walker once at their destination and walker should have rating updated accordingly |

|  |  |
| --- | --- |
| Name | Previous Walks **(Useful)** |
| Actor | Walker |
| Triggers | User clicks on Previous Walk button in the side pull out window |
| Events | Page is brought up showing the the walker history for the user |
| Exit Condition | User navigates to a different part of the app |
| Post- Conditions | User is taken to the different part of app they selected |
| Acceptance Test | A user registered as a walker should be able to view their walk history when they select Previous Walks from pull out side window. |

## 

|  |  |
| --- | --- |
| Name | Request Partner **(Must Have)** |
| Actor | Walker |
| Triggers | When a walker presses the “Find Partner” button. |
| Events | The walker will be paired with another walker of the opposite sex who is within close proximity to them. |
| Exit Condition | The walker is paired with another walker, or they log out of the app. |
| Post- Conditions | Once the walker has a partner, the walker main screen will display. |
| Acceptance Test | A walker will be able to click ‘Find Partner” and be matched with another walker. |

|  |  |
| --- | --- |
| Name | Finish Walk Notification **(Must Have)** |
| Actor | Walker |
| Triggers | Once walker is detected to be at the destination |
| Events | Popup is displayed asking if the walker is at the destination. |
| Exit Condition | Walker selects either “Yes” or “No” |
| Post- Conditions | **Yes**: The walker is taken back to the walker main page.  **No:** The walker must manually navigate to the “End Walk” button in the menu. |
| Acceptance Test | The walker should be prompted to end their walk when they are within a few yards of the destination created by the user. |

|  |  |
| --- | --- |
| Name | Start Walk Notification **(Must Have)** |
| Actor | Walker |
| Triggers | When the walkers arrive near the user’s location. |
| Events | A push notification appears on the walkers’ screens asking if they have located the user. |
| Exit Condition | The walker selects “Yes” or “No” |
| Post- Conditions | **Yes:** The walk begins. The path to the destination centers itself in the screen for the walkers and the user.  **No:** The notification disappears, and the walkers are able to manually confirm the location of the user. |
| Acceptance Test |  |

|  |  |
| --- | --- |
| Name | End Walk **(Must Have)** |
| Actor | Walker |
| Triggers | The walker presses the “end walk” button. |
| Events | A confirmation option is displayed. |
| Exit Condition | The walker selects “Yes” or “No” |
| Post- Conditions | **Yes**: The walk is cancelled. The walker is returned to the walker main screen, and the user receives a notification that the walk was cancelled and is returned to the user main screen.  **No**: The confirmation modal disappears and the walk continues. |
| Acceptance Test | If the walker cancels a walk, the user should receive a notification and both walker and user should be returned to their respective main screens. |

|  |  |
| --- | --- |
| Name | Out of Bounds Request (**Useful**) |
| Actor | User |
| Triggers | The user submits a walk request, but the user is either located outside of the allowed zone, or their destination is outside of the allowed zone. |
| Events | A popup screen displays telling the user that their request is out of bounds. |
| Exit Condition | The user closes out of the popup. |
| Post- Conditions | The user can submit a new request if their destination was out of bounds, or they must move into the allowed zone if they are able to. |
| Acceptance Test | A user who is outside of the allowed zone or submits a destination outside of the allowed zone should receive a message informing them that their request was not processed. |

|  |  |
| --- | --- |
| Name | Pull Out Side Window (**Must Have)** |
| Actor | User/Walker |
| Triggers | User clicks on pull out side window button |
| Events | Side menu is brought up |
| Exit Condition | User navigates to a different part of the application |
| Post- Conditions | User is taken to the part of application the user selected |
| Acceptance Test | User should be able to click on the pull out side window button and the new sub-menu should appear |

|  |  |
| --- | --- |
| Name | Account Info (**Must Have)** |
| Actor | User/Walker |
| Triggers | User clicks on the Account Info button on the side window pull out |
| Events | User is taken to page displaying their account information |
| Exit Condition | User navigates to a different part of the application |
| Post- Conditions | User taken to the area of the application they chose |
| Acceptance Test | User should be able to click on the Account info button under the pull out side window and be able to pull up information on the account they are registered with |

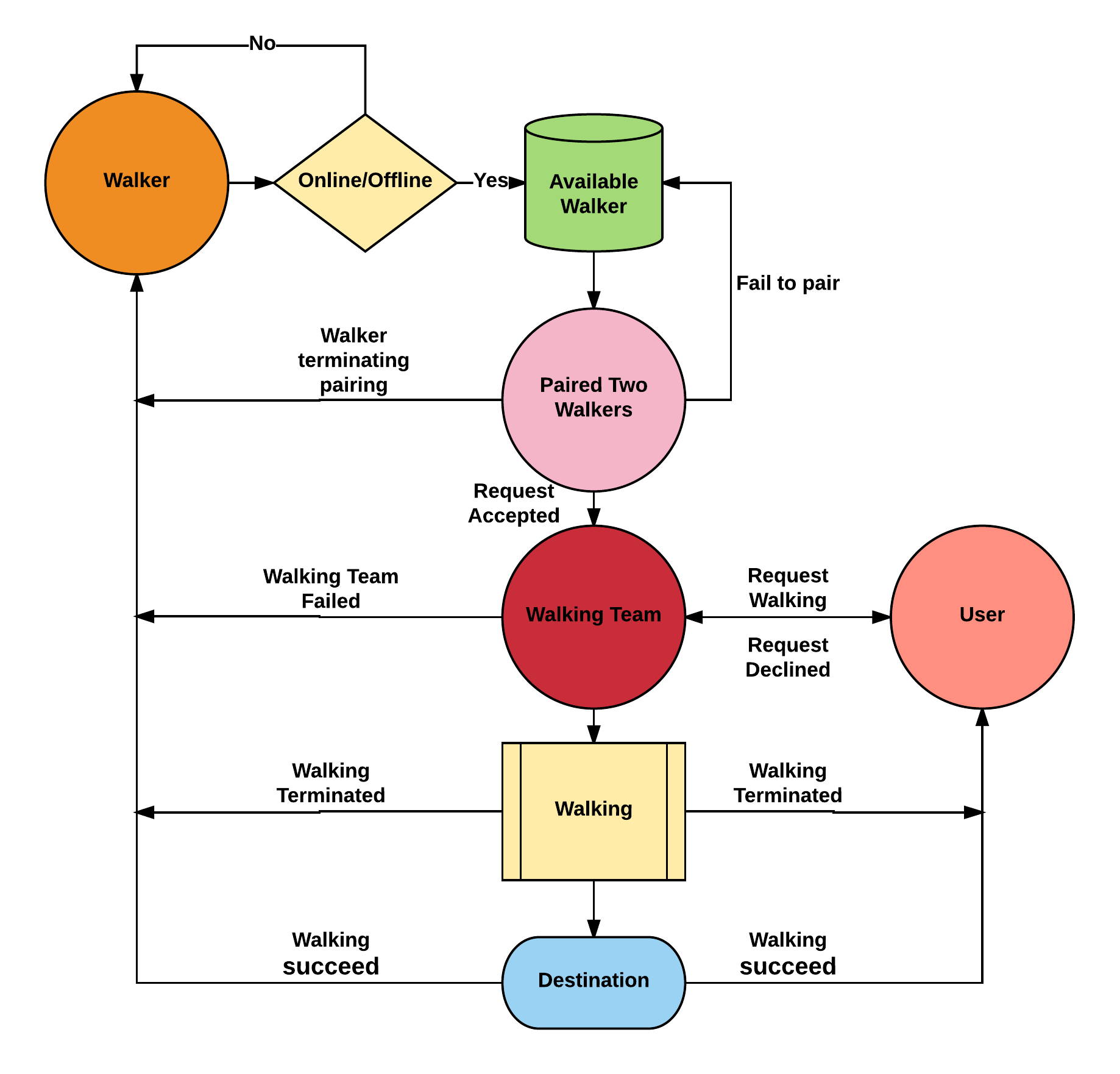
|  |  |
| --- | --- |
| Name | Report User / Walker (**Useful)** |
| Actor | Client |
| Triggers | The user/walker clicks the “Report” button located in the side menu. |
| Events | Popup to submit and file a complaint against walker comes up |
| Exit Condition | User completes the form or cancels |
| Post- Conditions | User is taken back to the main map page |
| Acceptance Test | User should be able to use the report button during or after a walk and walker will be marked down for it. |

|  |  |
| --- | --- |
| Name | Auto center of map/current location **(Useful)** |
| Actor | User/Walker |
| Triggers | The user/walker touches the current location button in the bottom right of the screen |
| Events | The map centers to the user location |
| Exit Condition | There is no exit condition |
| Post- Conditions | The user stays on the main page |
| Acceptance Test | The user should be able to search the map by moving it, and then click on the auto center/current location button to move the user’s map to their location. |

|  |  |
| --- | --- |
| Name | Partner Meetup |
| Actor | Walker |
| Triggers | Two walkers agree to be partners |
| Events | Map is brought up and walkers can select/agree on a place to meet up to begin their partnership |
| Exit Condition | Users arrive at their meetup point |
| Post- Conditions | Both walkers are taken back to the main walker page and wait for walk requests |
| Acceptance Test | Two walkers should be able to select a meetup location upon agreeing to be partners and once both arrive at destination they can start receiving requests |

**User Interface Requirements**

Workflow



So based on workflow we divide the interface into 3 parts: (1) Pairing (2) Meeting (3) Walking. And we will use some mockup figures to show the normal process that both the walker and users will see what when use our app. Here normal means we do not consider some edge cases which have been listed in test case part.

(1) Pairing

Pairing means two walkers are paired together to work together to walk the user home.

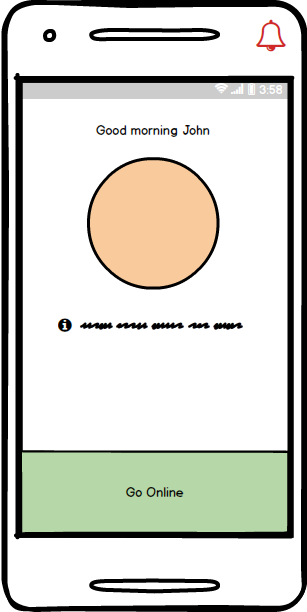
(2) Meeting

Meeting means walker team approach the user to form a walking team.

(3) Walking

Walking is the key part of our app where the users and two walkers form a team to complete the walking.

Then we will illustrate the working interface of our app by a walker and user story. Assuming **a walker whose name is John** and he has decided to be the walker tonight, so he opens his app on his Android phone and he see the following greetings,

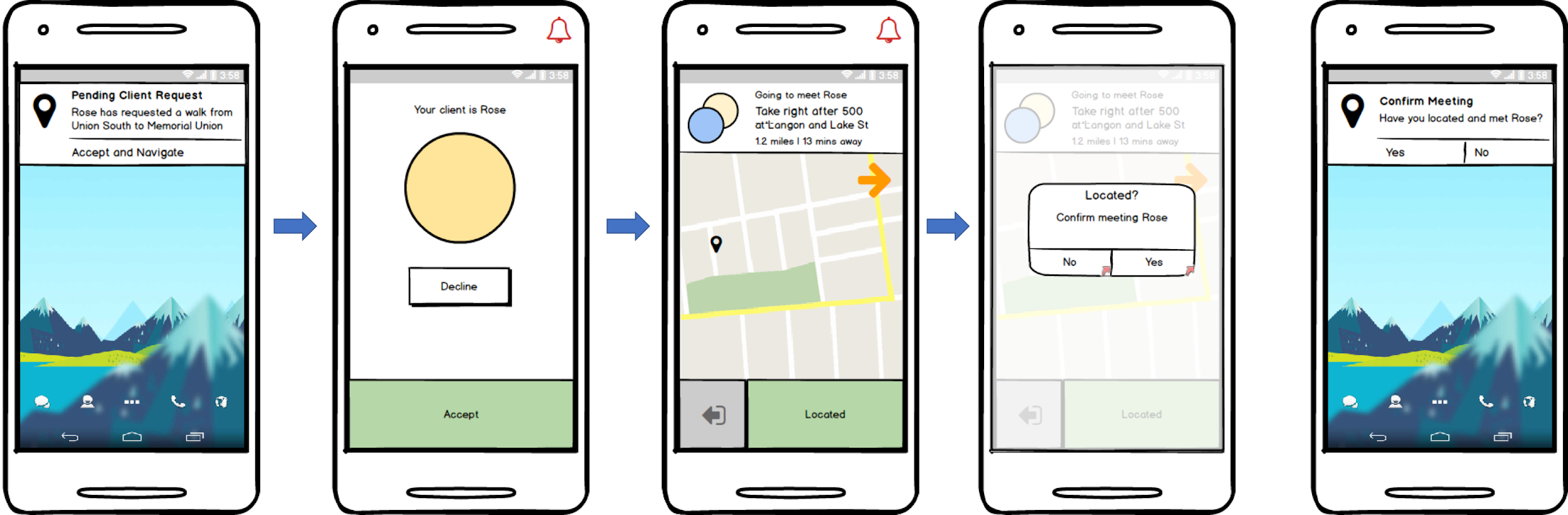


Of course, John is very happy to choose the go online button which will make him be available in our walker database then based his location we will pair him with another walker, typically another walker is in the opposite sex to make both a female and male walker in a walking team for safety concerns and make our user feel comfort. So in John’s case, let’s assume he is paired with **another walker Jane**.

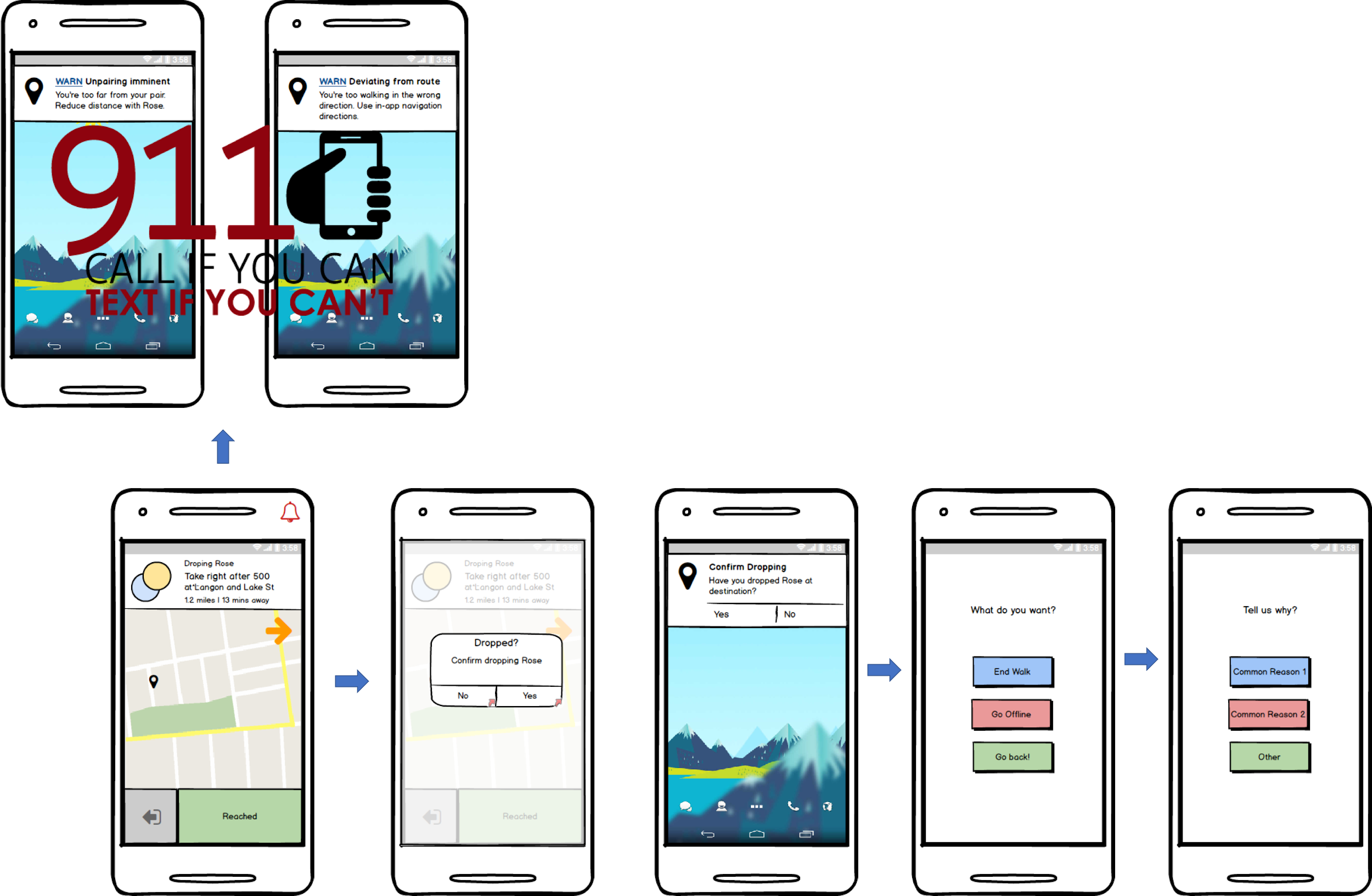


There will also be cases that the pairing is not successful or another walker won’t accept the pairing which is indicated in the figure. And if both walkers agree to team together, then our app will guide them to meet with each other. So one walker will approach another with the direction shown in map and on another user’s phone there will be a waiting screen. After they meet each, they will be marked available to accept user request in our database.

Then if we assume **a client named Rose** requests a walking service, then the walker and user needs to meet with each other to form a team. Below the figure to show this process.



Then our walking team can help the user to reach their destination. During the walk, there maybe some unexpected cases like deviation from planned route or an emergency situation, so we give both the user and walkers the option to call 911 or show their progress of route to one of their friends. And after both the user and walker confirm that the walking is complete, the walker can choose whether he or she wants a next walk or just simply go offline. If they like, we also wants to hear advice and suggestions from them.



## 

## **Security Requirements**

As of now there are no real potential security issues for our application. A user’s login/password information will be stored, however, Firebase has its own authentication system so that should handle potential threats. Outside of the login information, the only other information that will be stored is generic profile information such as name/age/etc but nothing of major concern privacy-wise. Firebase has a built in authentication.

## **System Requirements**

### Back-end

* Firebase
* Google Map API
* Google Places API

### Front-end

* User Interface
* Walker Interface

Phone Requirements

* Android 4.0 OS (ice cream sandwich) or higher
* GPS
* Data Connection
  + Need a cellular plan to access data or constant wifi
* At least 1GB of RAM
* At least 100 MBs of internal storage
* At least a speed of 550 MHZ for the processor
* Touch screen

### Version control

* Github repository to manage code base
* Application code is split into four categories
  + Backend - Firebase
  + Middleware - Google Maps API,Google Places API, and Android Studio
  + Front-end and UI (Client Side)
  + Front-end and UI (Walker Side)
* Google drive to manage documentation

## 

## **Specification**

**Client Side**

****

\* indicates function will return to Home Page

**Walker Side**

****

\* indicates function will return to Home Page