KASPER – ID1 PROJECT DOCUMENTATION

SOFTWARE DEVELOPMENT TEAM:

Project Manager: Tushita Patel

Developing Lead: Dylan Prefontaine

Testing Lead: Jeremy Liau

Build Manager: Christopher Mykota-Reid

Developers: Gaurav Arora, Arianne Butler, Haotian Ma, Kristof Mercier, Melody Zhao

Test Team: Christopher May, Ryan Tetland

Contents

[1.0 Requirements Document 3](#_Toc473898813)

[1.1 Requirements ID-1: 3](#_Toc473898814)

[1.2 Platform Description (iOS, Android, and Web Browser) 4](#_Toc473898815)

[1.3 GUI Mock Ups: 5](#_Toc473898816)

[1.4 Use Cases: 7](#_Toc473898817)

[1.5 Sequence Flow Diagrams: 11](#_Toc473898818)

[1.6 Mini Milestones for ID1 13](#_Toc473898819)

[2.0 Design 14](#_Toc473898820)

[3.0 Testing Plan 14](#_Toc473898821)

[4.0 Test Report 14](#_Toc473898822)

[5.0 Coding Style Guide 14](#_Toc473898823)

[6.0 Build Report 15](#_Toc473898824)

[7.0 Defect Report 15](#_Toc473898825)

[8.0 Upcoming ID-2 Requirements 15](#_Toc473898826)

[8.1 User Requirements: 15](#_Toc473898827)

[8.2 Back-end System Requirements 17](#_Toc473898828)

[9.0 Future Requirements (Nice to Haves) 17](#_Toc473898829)

# 1.0 Requirements Document

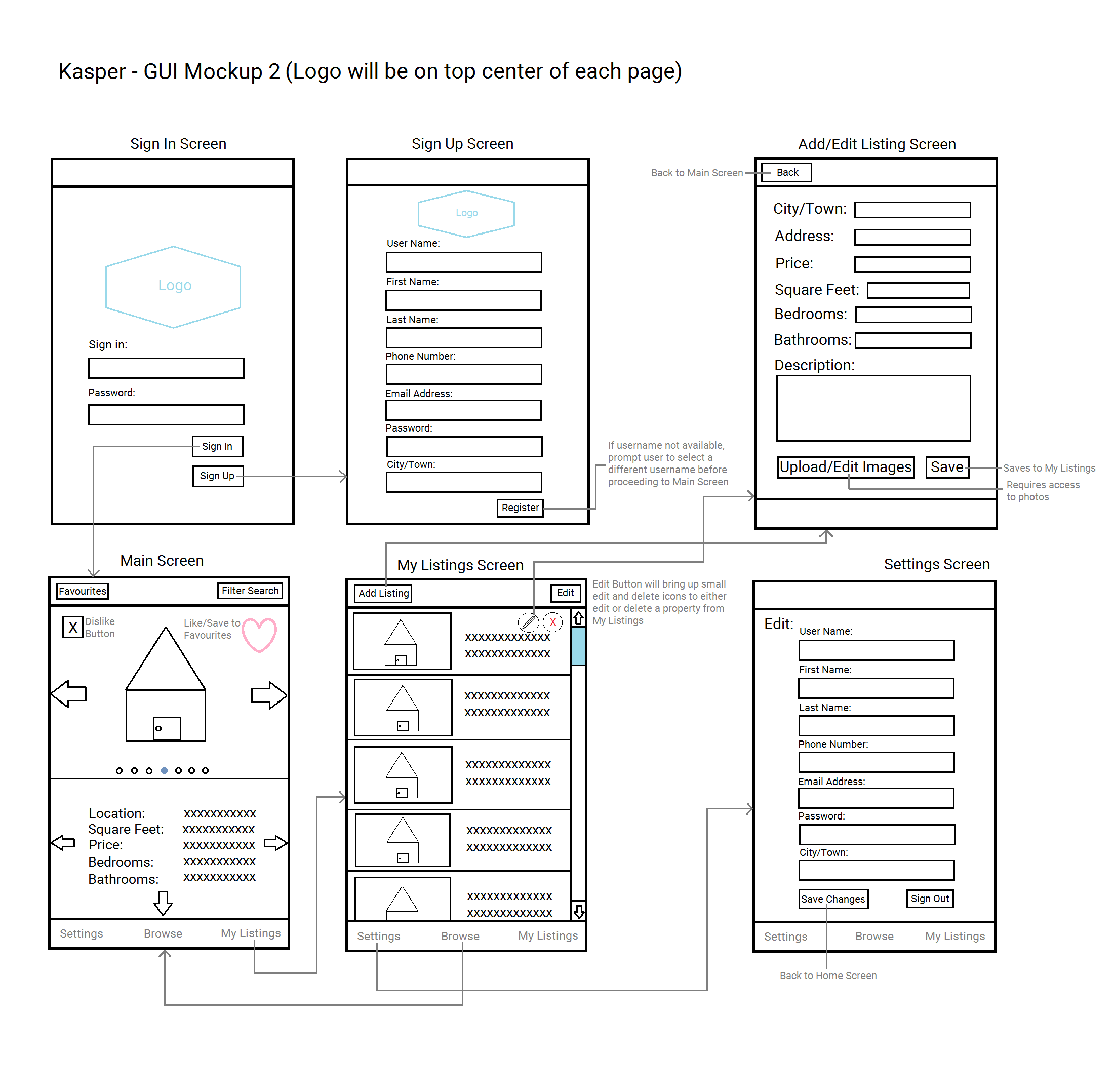
## 1.1 Requirements ID-1:

* Basic User Interface
  + Buttons
  + Text boxes
  + Range sliders
  + Menus
  + Links between screens
  + Image swipe feature
* Generate Dummy Data Structures
  + User Structure:
    - Integer 🡪 ID
    - String 🡪First Name
    - String 🡪 Last Name
    - String 🡪 Email Address
    - String 🡪 Password
    - String 🡪 Salt
    - String 🡪 PhoneNum (x3)
    - The ID 2 stuff looks good
    - Future stuff looks good as well
    - Location 🡪 Location
    - DateTime 🡪 Registered
    - DateTime 🡪 LastSeen
  + Listing Structure Fields:
    - Integer 🡪 ListingID
    - Integer 🡪 ListerID
    - Float 🡪 Price
    - Integer 🡪 SquareFeet
    - Integer 🡪 Bedrooms
    - Float 🡪 Bathrooms
    - Location 🡪 Location
    - String 🡪 Description
    - Boolean 🡪 isPublished
    - DateTime 🡪 dateCreated
    - DateTime 🡪 lastModified
    - Array 🡪 Thumbnail Images
    - Array 🡪 Images
  + Location Structure (to be determined):
    - Province 🡪 Province
    - String 🡪 City
    - String 🡪 Address
    - String 🡪 Postal Code
    - Double 🡪 Latitude
    - Double 🡪 Longitude
  + Province Structure:
    - String 🡪 Name
    - Char[2] 🡪 Abbrev

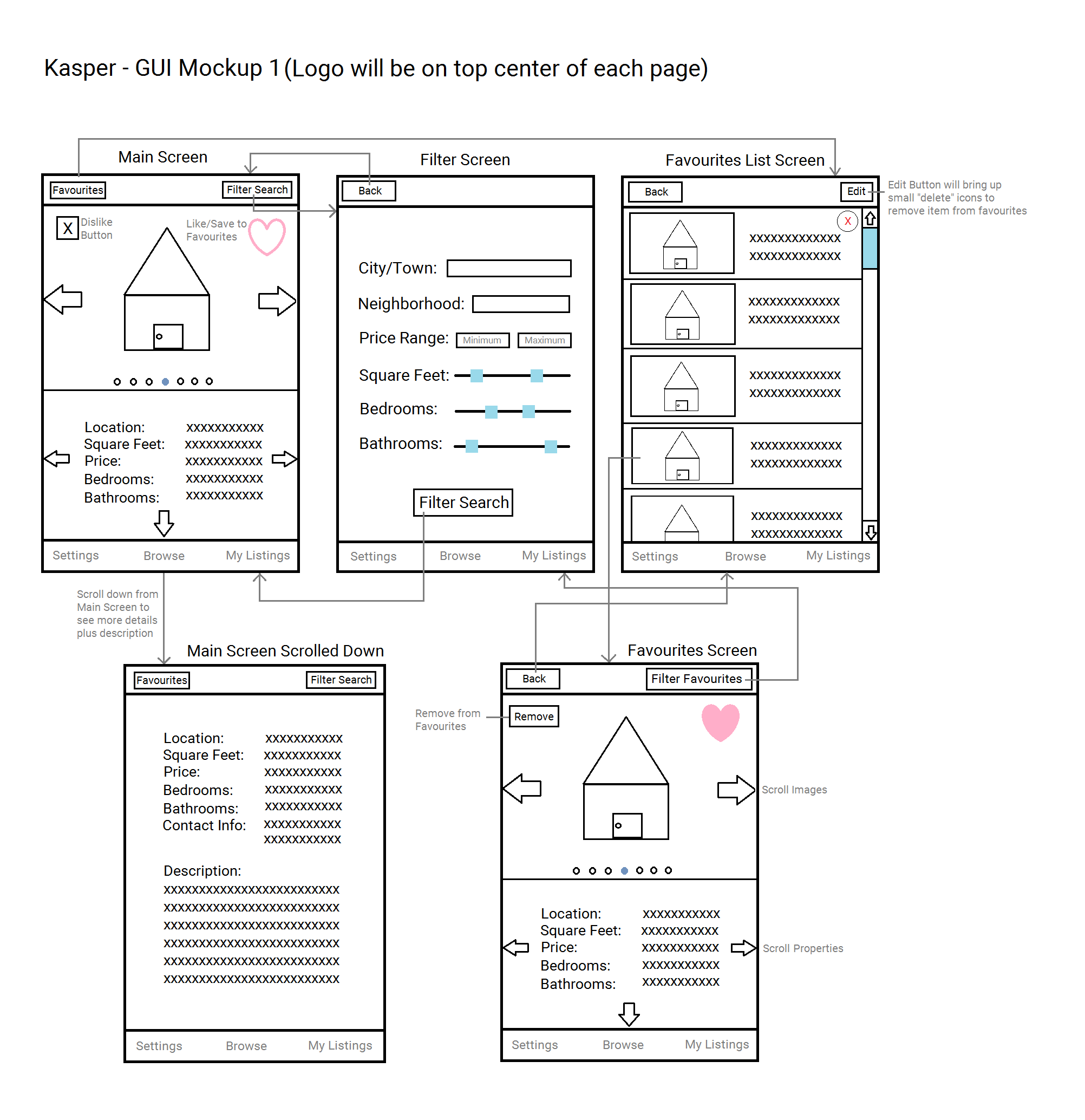
## 1.2 Platform Description (iOS, Android, and Web Browser)

* Insert Platform Description by Tushita

## 1.3 GUI Mock Ups:

GUI Mockup showing Sign In, Sign Up, Main Screen, Listings, Add/Edit Listings, and Settings

The above diagram shows the flow of events when a user chooses to either Sign in or Sign up. When a user signs in or signs up successfully, they will be taken to the Main Screen, where they can browse through an unfiltered set of property Listings. Users who wish to post their own Listings may do so by choosing the My Listings option from the bottom tab bar. Clicking on My Listings will take the user to the My Listings screen, where they can Edit, Remove, and Add New Listings. Choosing the Add Listing button from the top tab bar will navigate to the Add/Edit Listing screen, where the user can set up their new Listing, add pictures, and provide a description. Lastly, users may edit their personal profile in the Settings screen.

GUI Mockup showing Main Screen (with scroll down), Filter Search, Favourites List, and Favourites ****

The two large sideways arrows on the Main Screen indicate the ability to scroll through the provided images, while the two smaller sideways arrows below indicate the ability to swipe on to a new Listing. The small arrow on the bottom shows the ability to swipe down for more information as shown. Users will have the option to filter a search and be shown only those Listings which match their filter criteria. Users can tap the heart button to save a Listing to Favourites, and all Listings saved to Favourites can be removed by choosing the Edit button on the Favourites List Screen. To view the Listing of an item saved to Favourites, the user must click on the desired Listing in the Favourites List screen, which will bring up their selected Listing in the Favourites Screen. Users may also remove a Listing from Favourites by hitting remove on the Favourites screen.

## 1.4 Use Cases:

Sign-in:

The user must enter the correct e-mail and password to enter the app. They can also choose the option to register.

* Actor(s):
  + User, Database
* Precondition(s):
  + app is open
* Basic Flow:
  1. User enters Username and password
  2. User clicks Sign-in
  3. User’s info is verified with the Database
  4. If password is correct the User is sent to Main Screen
* Alternate Flow:

4\*. Username/password is incorrect and User re-enters

* Post Condition(s):
  + The app is on the Main Screen

Sign-up:

The user must enter their information to create an account. They also have the option of going to the Sign-in page.

* Actor(s):
  + User, Database
* Precondition(s):
  + App is open
* Basic Flow:
  1. User selects sign up button
  2. User enters information and selects register
  3. The User’s information is checked if correct and added to the Database
  4. User signs in
* Alternate Flow:

4\*. There is an issue with the info and User is prompted to fix issue

5\*. User fixes info and submits info

* Post Condition(s):
  + User has an account
  + The app is on the Main Screen

Filter:

The user will click on the Filter Button to enter a specialized Listing search. They must choose which constraints to use, and the Listings will be updated accordingly.

* Actor(s):
  + User, Database
* Precondition(s):
  + User is either on the Main Screen or Favourites
* Basic Flow:
  1. User selects Filter Button
  2. User selects specific filters and clicks filter search button
  3. The database is called and the list of viewable Listings is populated.
* Alternate Flow:

2\*. User selects back and returns to Main Screen

* Post Condition(s):
  + Entries change based on filter input

Swipe through Listings:

The user will swipe right on images to view the next image for the current Listing. To view the next Listing the user will swipe the text portion of the screen. Swiping right and left will move forward and backward through the Listings respectively.

* Actor(s):
  + User, Database
* Precondition(s):
  + User is on the Main Screen
  + Listings Buffer contains five Listings loaded from the Database (relevant to design in ID-2)
* Basic Flow:
  + User swipes Left or Right
  + Next Listing is loaded from the Database
* Alternate Flow: none
* Post Condition(s):
  + Next Listing is visible on the Main Screen

Disliking or Add Listing to Favourites:

While navigating the main page, the user can indicate if they would like to not see a property again, or they can add one to their favourites list.

* Actor(s):
  + User, Database
* Precondition(s):
  + User is on the Main Screen
* Basic Flow:
  1. User selects Favourite Button on a Listing
  2. The Listing is added to user Favourites

Alternate Flow:

1.\*User selects or deselects the dislike button, changing the “disliked” field on a Listing.

* Post Condition(s):
  + Listing has either been added to favourites or the “dislike” field has been altered.

Change Settings:

To change their personal settings, the user must navigate to the Settings. They can make the necessary changes and then save them, or they can log out.

* Actor(s):
  + User, Database
* Precondition(s):
  + User is on the Main Screen
* Basic Flow:
  1. User selects Settings Button
  2. User edits Settings and clicks the Save Changes button
  3. The User’s new info is confirmed and updated in the Database
* Alternate Flow:

2\*. User clicks Sign-out button and is sent to the Sign-in Screen

* Post Condition(s):
  + Users info is successfully updated

Add Listing:

To add a Listing, a user must upload their information including a description of the house and images. They can then choose to publish a Listing once they have finished creating its profile.

* Actor(s):
  + User, Photo Directory, Camera, Database
* Precondition(s):
  + User is on the Main Screen
* Basic Flow:
  1. User selects the Add Listing Button
  2. User fills in the Listing information and may upload photos
  3. User clicks the Save Button
* Alternate Flow: none
* Post Condition(s):
  + Listing is added to My Listings and the Data Base and is made available for other users to see

Edit Listing:

While looking through their Listings, a user can choose to edit the current Listing. At this point they can then choose to publish or un-publish a Listing, or alter the Listing’s images or information.

* Actor(s):
  + User, Photo Directory, Camera, Database
* Precondition(s):
  + User is on the Main Screen and has at least one Listing in My Listings
* Basic Flow:
  1. User selects Listings button
  2. User selects Edit from the Listings Screen
  3. User submits the edited info
  4. The new Listing info is verified and then updated in the entries Database
* Alternate Flow:

3\*. User toggles the publish button

* Post Condition(s):
  + Listing modified in My Listings and the Database

Viewing and Removing Listing from Favourites:

The user can view their Favourites List by navigating to the Favourites List page.

* Actor(s):
* Precondition(s):
  + User is on the Main Screen
* Basic Flow:

1. User selects the Favourites button
2. Scrolls through their favourited properties

* Alternate Flow:

1. The user swipes left on a property and presses the delete button.

* Post Condition(s):
  + Listings are removed from a User’s favourites if applicable.

Selecting an Item from Favourites Listing:

The user may change the Main View to reflect the contents of their Favourites List. At this point the user may proceed in a similar way to the Main Page, except that Listings can now only be removed from Favourites, and not added.

* Actor(s):
  + User, Database
* Precondition(s):
  + User is on the Main Screen
* Basic Flow:
  + User selects a Listing from the Favourites List
* Alternate Flow:

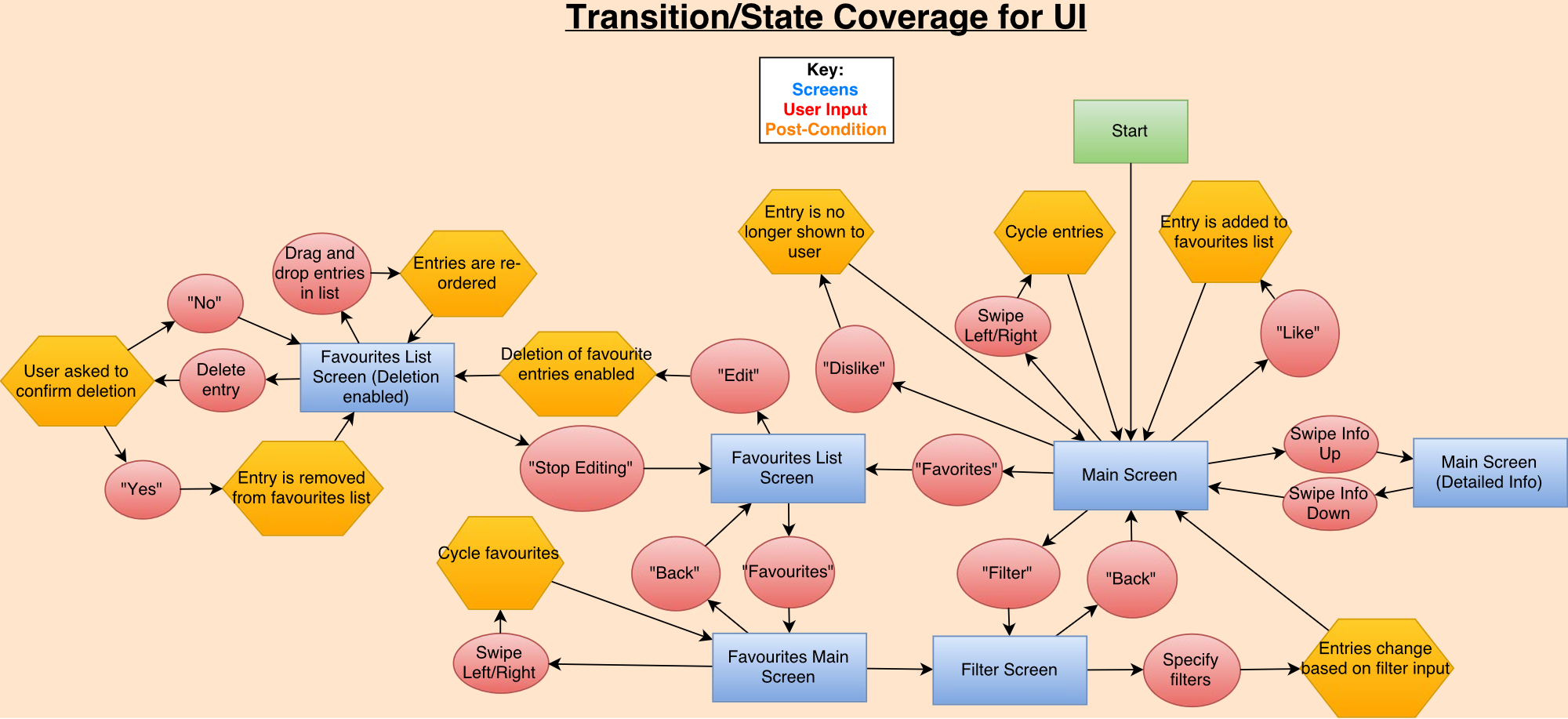
1. User selects the Edit Button

2. User selects the Remove Button and confirms the action.

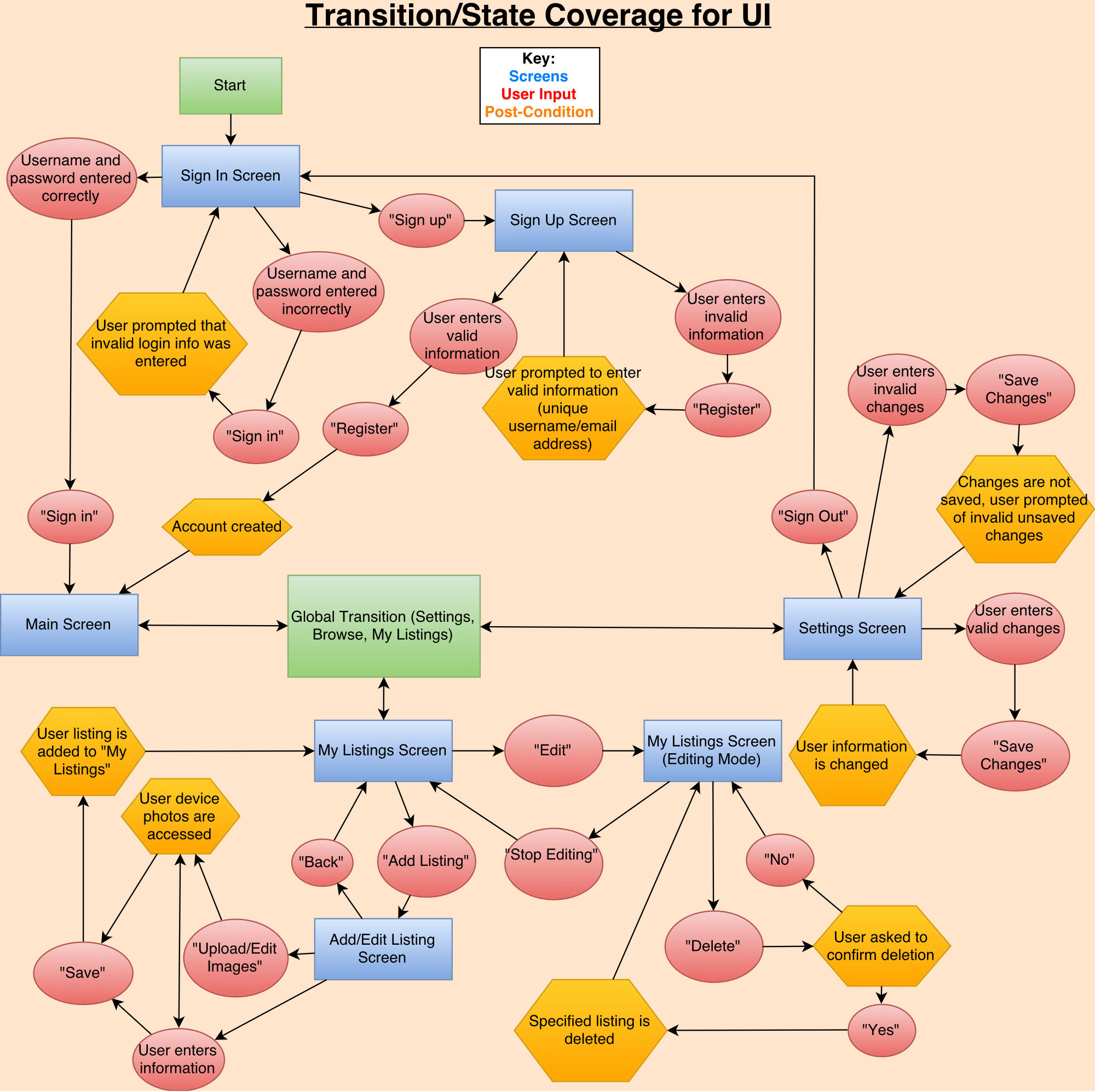
Post Condition(s):

* + The user is on the Favourites Screen of the Listing

## 1.5 Sequence Flow Diagrams:

****

* Jeremy will replace these and test team will provide brief description

****

## 1.6 Mini Milestones for ID1

**Dev Team:**

* Every developer familiar with Ionic2, CSS, HTML, typed Script and Angular2 ✔
* Each developer has completed at least one App Page
* Unit Testing of GUI’s where possible (not much to test in ID-1)
* Decide on System Architecture ✔
* Finalize Design for the System
* Implement GUI Designs

**Test Team:**

* Visual testing of the GUI’s implemented by Dev Team
* Testing reports complete
* All testers familiar with testing frameworks ✔

**Build:**

* Set up Travis with Ionic
* Verify code style (low priority)
* Set up Django, goDaddy and server

**Documents:**

* Testing Plan Document
* Risk Report
* Requirements Document
* Design Document
* All documents edited and compiled into one fluid document

**Other:**

* Each Developer and Tester has completed a pair programming session
* Get server to host Django
* Learn how to host Website on goDaddy.com
* All group member’s familiar with Trello, GitHub, Slack, and Google Documents ✔
* Finish GUI Design and Mock-ups for ID1 ✔
* Triage meeting
* Meet with Upper Management
* Meet with client weekly ✔
* Every member familiar with the Schedule as given by project manager ✔

# 2.0 Design

* UML Diagrams -> DYLAN
* Architecture Diagram -> GAURAV
* Architecture Description / Defense -> KRISTOF

SELECT \* FROM (table) WHERE (current range\*) ORDER BY (chosen field), id LIMIT 100

\*this range will be stored on the device to resume the search later.

* when you get to 100, call the same sort (sorted by filter) again on the database and skip to the stored cursor position

Architecture is 3-Tiered such that tier 1 is View/Model components, tier 2 is Controller, tier 3 is the server. MVC is used by ionic.

# 3.0 Testing Plan

* Jeremy and Test Team

# 4.0 Test Report

* Jeremy and Test Team

# 5.0 Coding Style Guide

Initially, our team decided to follow the Google JavaScript Guidelines:

<https://google.github.io/styleguide/jsguide.html>

However, because this resource is detailed and extensive, and because the team will also be using TypeScript, we decided to create our own set of guidelines to serve our purpose. The guideline highlights the salient features of the coding style to be followed by all developers, along with useful examples for quick referencing.

The summarized guide and sample can be found on the following wiki page:

<https://github.com/CMPT371Team1/Project/wiki/(Rough)-Coding-Style-Example-(JavaScript)>

# 6.0 Build Report

* Chris MR

# 7.0 Defect Report

# 8.0 Upcoming ID-2 Requirements

## 8.1 User Requirements:

* Generate Dummy Data Structures:
  + UserVotes:
    - Integer 🡪 UserID
    - Integer 🡪 ListingID
    - Boolean 🡪 isLiked
  + Filter:
    - Location 🡪 Location
    - Float 🡪 minPrice
    - Float 🡪 maxPrice
    - Integer 🡪 minBedroom
    - Integer 🡪 maxBedroom
    - Integer 🡪 minFeet
    - Integer 🡪 maxFeet
    - Boolean 🡪 isFavourites
  + BrowseList:
    - Integer 🡪Cursor
    - Integer[100] 🡪 ListingIDs
    - Listings[5] 🡪 curLoaded
  + ListingImages:
    - Integer 🡪 ListingID
    - Image 🡪 Image
    - Integer 🡪 OrderDisplayed
* Generate Mock Classes:
  + ListingProvider:
    - addListing(ListingToJSON: String)
    - editListing(ListingToJSON: String)
    - removeListing(ListingID: Integer)
    - dislike(ListingID: Integer)
    - addToFavourites(ListingID: Integer)
    - removeFromFavourites(ListingID: Integer)
    - search(Filter: Filter) 🡪 Filter can be Null
  + LoginProvider (to be determined)

General User Requirements:

* Log on
* Sign up
* Edit settings and User Info
* View all property Listings
* Listing swipe feature (cursor)

Buyer Specific Requirements:

* Filter Search Based on:
  + City/Town
  + Address
  + Category (house, condo, building, etc.)
  + Square Feet
  + Price range
  + Number of bedrooms
  + Number of bathrooms
* View all Listing Info (including pictures, descriptions, and seller contact info)
* Save Listing to Favourites
* Browse Favourites
* Remove Listing from Favourites
* Receive notifications regarding Favourites
  + Price changes
  + Listing removed/edited

Seller Specific Requirements:

* View personal Listings
* Edit personal Listings (text fields, description, and images)
* Add new Listings (contact info can be automatically added via sign-up info)
  + Upload images
  + Input:
    - City/Town
    - Address
    - Category (house, condo, building, etc.)
    - Square Feet
    - Price
    - Number of bedrooms
    - Number of bathrooms
* Remove Listings

## 8.2 Back-end System Requirements

**System Design:**

The back-end system implementation is separated into two main modules – User accounts and Listings information. The account module handles user Sign-in, Sign-out, Sign-up, email verification, forgotten passwords, and resetting passwords. The Listings module defines a set of data related to a listed property, such as its location, price, description, images etc. To start the back-end, the http server is initialized, which calls all system modules before serving user requests. Thus, it has complete control over all parts of the system, and can decide to close any aspect should an issue arise.

**System Requirements:**

Fundamental aspects of the back-end behaviour can be defined by the following set of requirements:

**Functional Requirements:**

1. The back-end must gather data sent from devices and store it in the database for future reference.
2. User requests must be handled appropriately, and relevant information stored in the database must be sent to the device interface for display.
3. The system should be capable of recovering from failures and crashes whilst maintaining the integrity of any stored data.

**Non-functional Requirements:**

1. The back-end system should be responsive to user requests, so that delays in displaying data are minimized.
2. Data integrity and error correction mechanisms should be implemented so that no erroneous data is stored in the database.
3. The system should send informative error messages to the client about the source of error.
4. The system should provide an appropriate debugging environment, in which new code can be easily integrated, tested, and checked for errors.

**Software:**

The back-end system is implemented in Python and uses several external sources for specific implementations:

1. Google App Engine
2. NoSQL
3. Google Datastore NDB Client Library
4. Webapp2: a lightweight Python web framework

# 9.0 Future Requirements (Nice to Haves)

* Book a viewing feature
* Sign in using Facebook feature
* Set price watch on a given Listing
* Users can sign up to receive “hot list” notifications
  + Feature Listings (paid for by Sellers)
  + Newly added Listings
  + Price changes on Favourites
  + Based on previous search history
* Push notifications if something changes regarding a Listing saved in Favourites (change in database triggers notification)
* Sellers receive notifications regarding personal Listings:
  + When a Listing is saved to Favourites
  + When someone requests a viewing
  + When someone sets a price watch
* Integration with Google Maps
* Super admin User:
  + Log in as Super Admin
  + Add new Listing under any user
  + Edit any Listing
  + Remove any Listing