KASPER – ID2 PROJECT DOCUMENTATION

SOFTWARE DEVELOPMENT TEAM:

Project Manager: Tushita Patel

Dev Lead: Dylan Prefontaine

Test 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

Documentation: Arianne Butler

Contents

[1.0 Requirements Document 3](#_Toc475058733)

[1.1 Requirements ID2: 3](#_Toc475058734)

[1.2 Mini Milestones for ID2 5](#_Toc475058735)

[2.0 Design 6](#_Toc475058736)

[2.1 API Document 6](#_Toc475058737)

[2.2 Updated Data Structures for ID2 6](#_Toc475058738)

[2.3 Spike Prototype 6](#_Toc475058739)

[3.0 Time Estimations 7](#_Toc475058740)

[4.0 Testing Document 8](#_Toc475058741)

[4.1 Test Report 8](#_Toc475058742)

[4.2 Defect Report 8](#_Toc475058743)

[4.3 Test Matrix 8](#_Toc475058744)

[4.4 State Transition Diagrams 8](#_Toc475058745)

[5.0 Coding Style Guide 8](#_Toc475058746)

[6.0 Build Report 9](#_Toc475058747)

[7.0 Upcoming Requirements 10](#_Toc475058748)

[7.1 ID3 Priority Requirements: 10](#_Toc475058749)

[7.2 Requirements Changes 11](#_Toc475058750)

[7.3 Future Requirements 14](#_Toc475058751)

[8.0 Triage Meeting: 14](#_Toc475058752)

# Requirements Document

## 1.1 Requirements ID2:

Front End Requirements:

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

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)
  + Input:
    - City/Town
    - Address
    - Category (house, condo, building, etc.)
    - Square Feet
    - Price
    - Number of bedrooms
    - Number of bathrooms
* Remove Listings

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. It includes creating new Listings, getting filtered Listings, and edit existing Listings. 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

**Update:**

This ID our team did not expect to finish all back-end requirements as laid out in the above description. The following tasks were not finished for ID2 and will be pushed forward and prioritized in ID3:

* Email verification
* Forgot password
* Reset password
* Get all Listings for Browse page
* Get filtered listings
* Edit listings
* Like/Dislike
* Edit Account
* Get Favourites Listings

## 1.2 Mini Milestones for ID2

**Development**:

* UI Diagrams containing changes to be implemented for ID2 ✔
* API Document ✔
* Style Guide ✔
* Set up the Back End ✔
* Set up Unit Testing for Back End ✔
* Set up Unit Testing for Front End ✔
* Set up the Back End Development Environment ✔
* Spike Prototype Document ✔

**Testing**:

* Fix Protractor issues with sending keys, browsing tabs, etc. ✔
* Fix Karma Problems:
  + Importing issues ✔
  + Type Error, Reflect.getMetaData issue
* Test plan for future ID's ✔
* Manual testing ✔
* Update End-To-End tests, test matrix, and defect report
* Use case and state transition diagram updates ✔
* Teach test team to use tools ✔

**Documentation**:

* Compile documents ✔
* Edit documents ✔

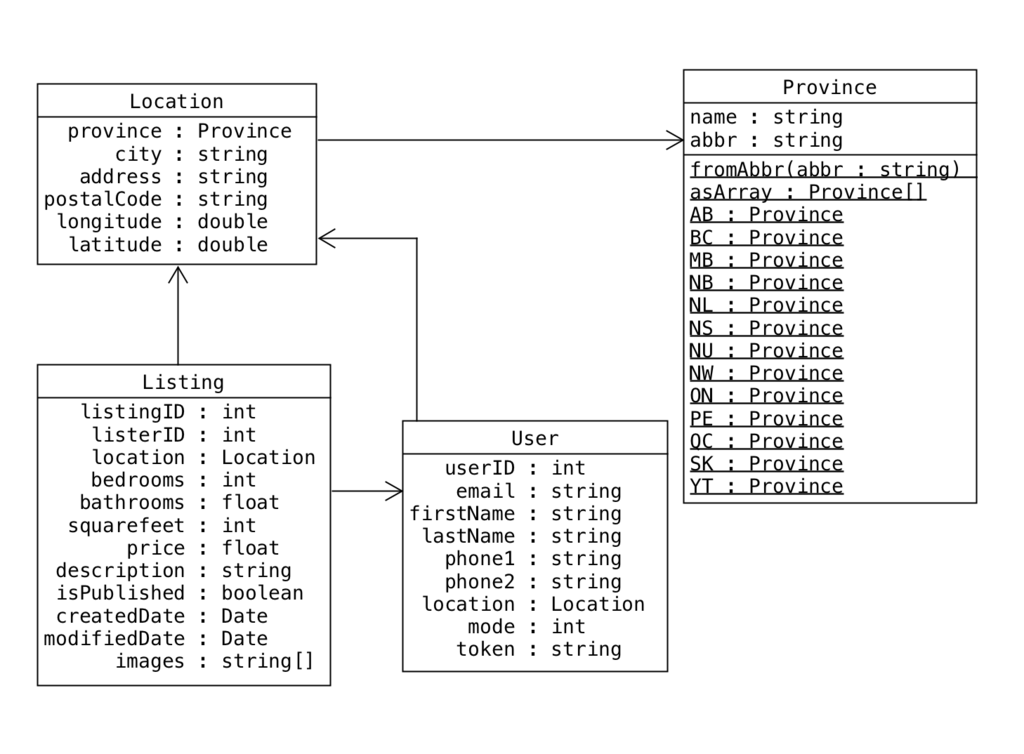
# Design

## 2.1 API Document

The API document outlines the client server communication of our system. Our project will contain roughly twelve API calls (or calls to the server), three of which are currently implemented. API calls are robust and complex call to the server that send a JSON with relevant information. The server receives, parses, and processes the JSON using the database. When finished, it will reply to the sender with a token containing the reply status and a JSON containing the requested information. The following link contains our API document and details on the various calls to our database:

[https://docs.google.com/document/d/1N4jt1\_PgxPhXwdc1TcT7TBjFNOZqYO5L10ha3bpO5M8/edit#heading=h.1sskatsa28we](https://docs.google.com/document/d/1N4jt1_PgxPhXwdc1TcT7TBjFNOZqYO5L10ha3bpO5M8/edit%23heading=h.1sskatsa28we)

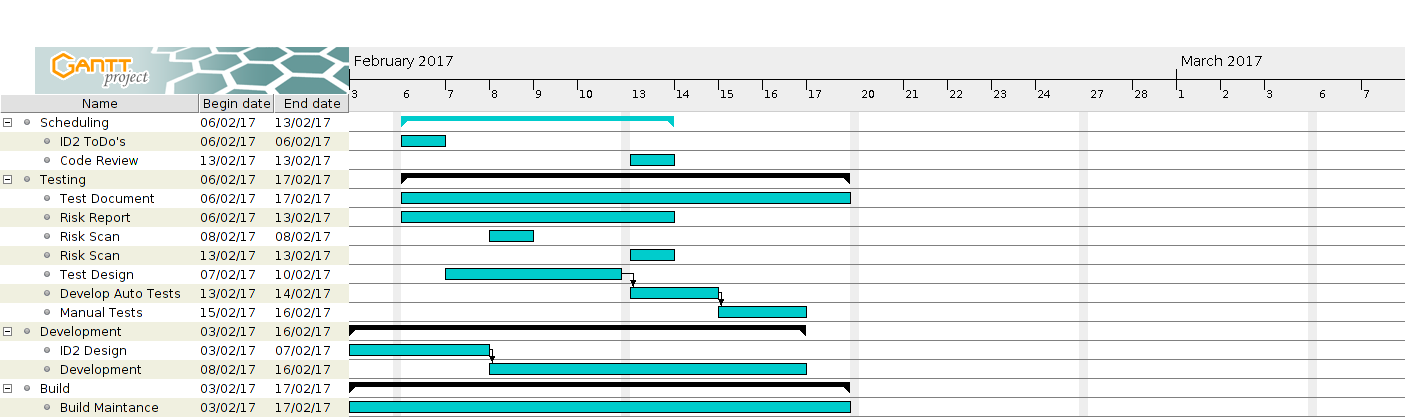
## 2.2 Updated Data Structures for ID2



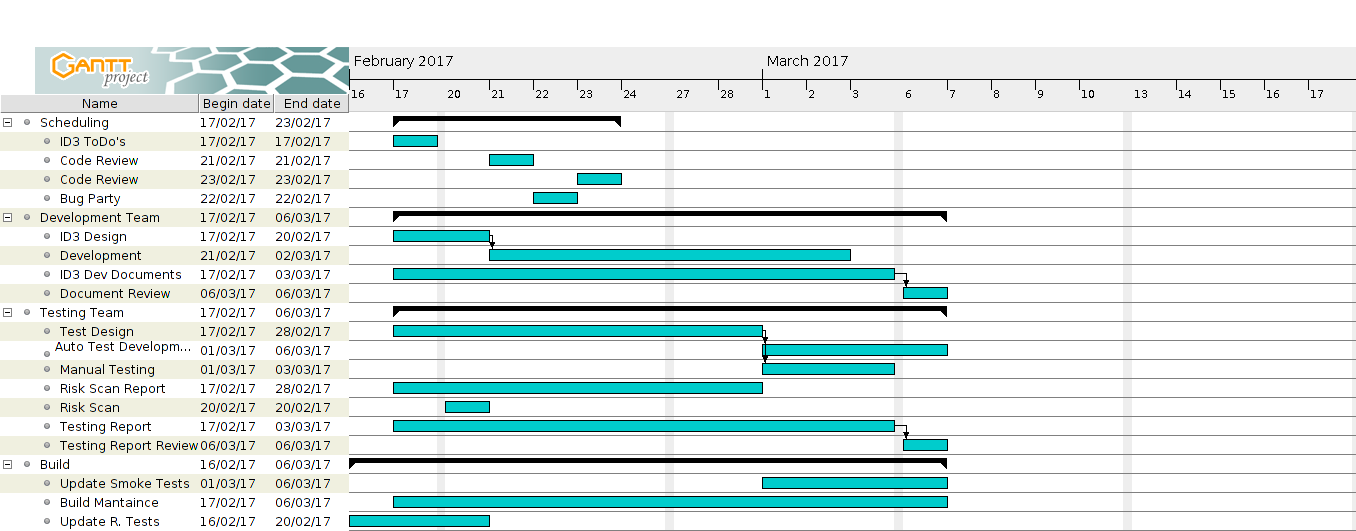
## 2.3 Spike Prototype

# 3.0 Time Estimations

Gantt Diagram Time Estimation ID2:



Gantt Diagram Time Estimation ID3:



# 4.0 Testing Document

## 4.1 Test Report

- Jeremy and test team

## 4.2 Defect Report

- Jeremy and test team

## 4.3 Test Matrix

The Testing Matrix can be found at the following link:

<https://github.com/CMPT371Team1/Documentation/blob/master/ID1-Documentation/TestingDocuments/testMatrix.xlsx>

## 4.4 State Transition Diagrams

- Jeremy and test team

# 5.0 Coding Style Guide

The development team has put together a set of guidelines to serve our purposes. These guidelines highlight the salient features of the coding style to be followed by developers. Useful examples are provided for quick referencing.

For front-end development using Ionic, the coding guide and sample can be found at the following wiki page:

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

For back-end development using Python, the coding guide and sample can be found at the following wiki page:

<https://github.com/CMPT371Team1/Project/wiki/Coding-Style-Guide-(Python)>

# 6.0 Build Report

Smoke Test Status:

The smoke test is currently not in place due the relatively small amount of implemented code in ID’s 1 and 2. The build manager is currently working to incorporate Protractor, an automated UI testing tool. Research is being conducted on Protractor installation and the Travis-CI script.

Build Status:

The builds for iOS and android are running and simulating correctly. The Linux build is currently building, then stalling and waiting for input. We intend to have the Linux build simulating before the end of ID2. Server is being built. Time for build operations was much lower (do you mean higher?) than expected, so the team was unable to finish all that was planned.

SDK’s, Packages, and Tools:

All SDKs, packages, and tools employed in our build, as well as their version number, are subject to change. These frameworks are still in question due to lack of experience. These decisions will be made final once the build manger has a firm understanding of all automated testing, deploying, server builds, and system builds.

Current list of SDK’s, Packages, and Tools:

* Cordova CLI: 6.5.0
* Ionic Framework Version: 2.0.0-rc.5
* Ionic CLI Version: 2.2.1
* Ionic App Lib Version: 2.2.0
* Ionic App Scripts Version: 1.0.0
* npm: 3.10.10
* jdk: 1.8.0\_121
* nvm: 0.32.0
* node: 6.9.4
* packages listed in package.json
* plus ~400 other Ionic dependency packages
* Android:
  + SDK Platform Android 7.1.1, API 25, revision 3
  + Android SDK Tools, revision 25.2.5
  + Android SDK Build-tools, revision 25.0.1
  + Android SDK Platform-tools, revision 25.0.3
  + Google Repository, revision 42
  + Android Support Repository, revision 42
* iOS:
  + OS: OS X El Capitan
  + Xcode version: Xcode 7.3.1 Build version 7D1014
* Server:
  + Python 2.7
  + Google Cloud sdk v143.0.1
  + Python Extension for google cloud v1.9.50
  + Python Extension (Extra Libs) v1.9.49

Corodova 6.5.0 requires both jdk 1.8 (or higher), npm v2.2.1, and node v4.0.0. We will be using the most recent version of node and npm to reduce version conflicts. We are using Google Cloud as required for the Google App engine, because our servers are built on this platform. The server uses Python 2.7 and the most recent version of Google Cloud SDK. The Python extensions are required for Google Cloud SDK, because our server is written in Python. All developers and testers are set up with the latest versions of all tools. We are using Xcode 7.3.1 for the time being. The Ionic dependencies are extensive, and can be viewed in further detail at the following link:

[https://ionicframework.com/docs/](https://ionicframework.com/docs/%20)

You can find the releases (for what?) at the following link:

<https://github.com/CMPT371Team1/Project/tree/develop/releases>

# 7.0 Upcoming Requirements

## 7.1 ID3 Priority Requirements:

Front-end Requirements (Not finished during ID2):

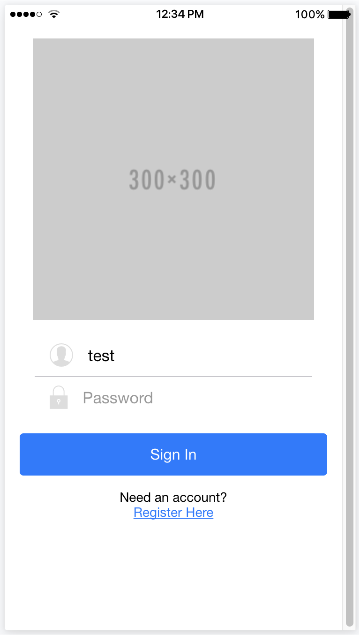
* Receive notifications regarding Favourites
* Price changes
* Listing removed/edited
* Seller Upload images

ID3 Back-end Requirements (Not finished for ID2):

* Email verification
* Forgot password
* Reset password
* Get all Listings for Browse page
* Get filtered listings
* Edit listings
* Like/Dislike
* Edit Account
* Get Favourites Listings

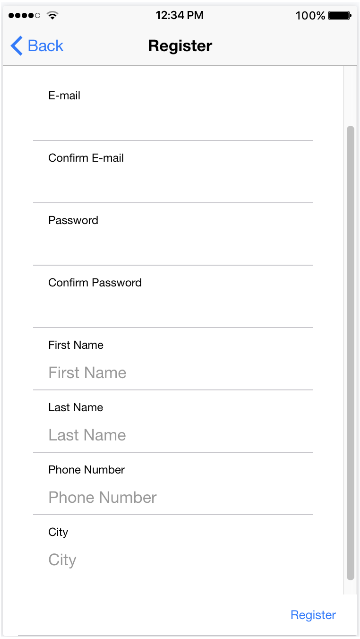
## 7.2 Requirements Changes

This section highlights the upcoming changes to our UI since ID1. These changes were discussed and agreed upon during Dylan’s formal code review, and will be implemented in ID3. For each screen that will undergo change, a screenshot of it’s original state, along with a point form description of the changes is shown.



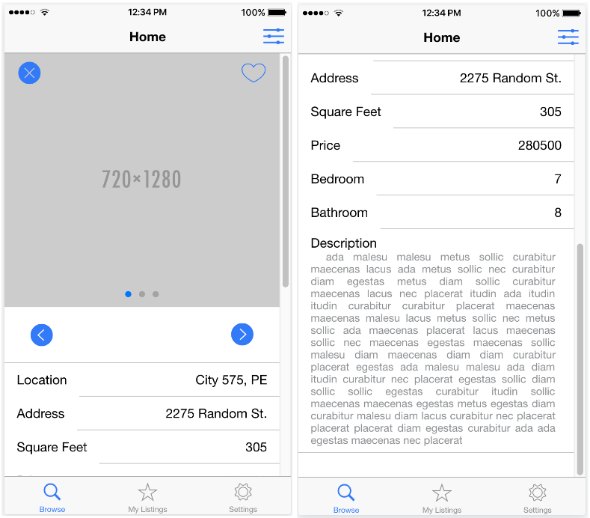
**Sign in Screen:**

* The Sign in Screen will no longer be the entry point of the app, and the new Browse Screen will take its place. The old Browse Screen has been renamed to “Details”
* Add Kasper logo
* Add Sign in/up using Facebook as per client request
* Style the app

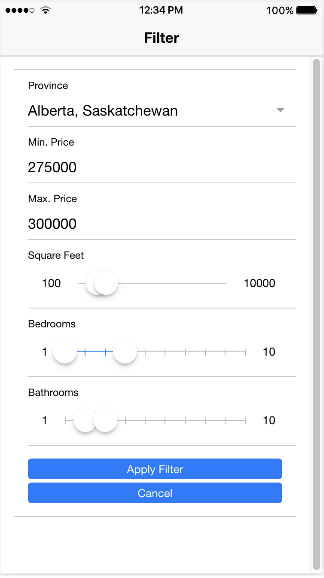


**Sign up Screen**:

* Ask for information in categorized steps, instead of all at once
* Indicate which fields are required \*
* Remove e-mail confirmation
* Send confirmation link
* Require email when requesting contact information or publishing new Listings
* Add pills to toggle buy/sell mode

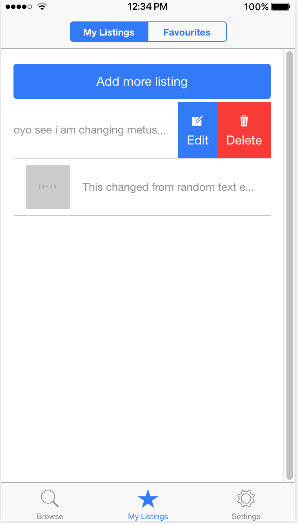
**Browse Screen**:

* Rename Browse to Detail, because of the new Browse page
* Add contact capability
* Add map capability
* Add Edit button, if the Listing belongs to the user
* Replace Filter button
* Improve Dislike button
* Remove star and replace with heart for consistency
  + - Remove Like/Dislike buttons according to whether their profile is in buy or sell mode, and whether it is the users own listing
    - Improve design and add icons (i.e. bed/bath icons)



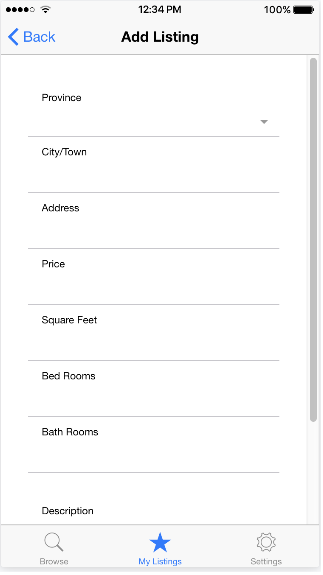
**Filter Screen**:

* Make upper-bound price optional
* Make upper-bound bedrooms/bathrooms optional
* Improve price selector by displaying in K’s or M’s instead of 0’s



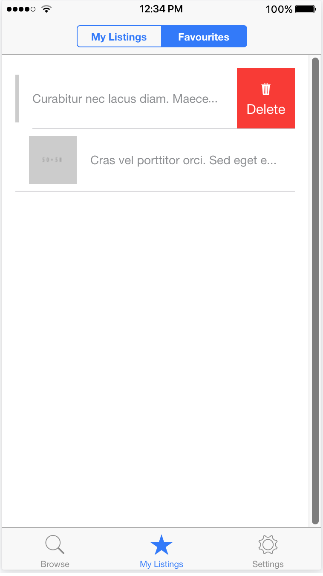
**My Listings Screen**:

* Switch from My Listings and Favourites segments to individual tabs accessible based on which mode the user is in
* Add button on the left navigation bar
* Edit button to the right navigation bar



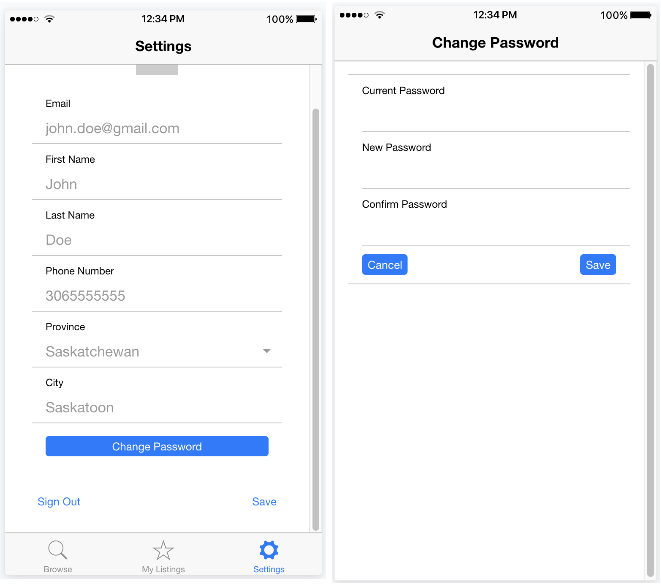
**Add Listings Screen:**

* Require at least one picture to publish a new Listing
* Fix description box
* Save without Publishing button
* Add Publish button
* Add Un-publish button
* Display images
* Optional ordering of images (nice to have)
* Specify contact preferences (i.e. phone or email)



**Favourites Screen**:

* This should look more like the new Browse page



**Settings and Password Screen**:

* Implement Buy and Sell mode
* Toggle the following notifications:
  + - * + Unpublished
        + Favorited
        + Price Change
* Divide Settings and Profile, and rename Profile portion to My Account

GUI Mock-ups of the above UI changes can be found at the following link:

<https://github.com/CMPT371Team1/Documentation/blob/master/ID2-Documentation/OtherDocuments/ID3-UI-Mockups.pdf>

## 7.3 Future Requirements

These section outlines requirements identified for upcoming ID’s, some of which will take priority over others, and some of which we may not have time to implement this term.

* Book a viewing 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

# 8.0 Triage Meeting: