KASPER – ID2 PROJECT DOCUMENTATION

SOFTWARE DEVELOPMENT TEAM:

Project Manager: Tushita Patel

Dev Lead: Kristof Mercier

Test Lead: Jeremy Liau

Build Manager: Christopher Mykota-Reid

Developers: Gaurav Arora, Haotian Ma, Dylan Prefontaine, Melody Zhao

Test Team: Christopher May, Ryan Tetland

Documentation: Arianne Butler

Contents

[1.0 Requirements Document 3](#_Toc475133506)

[1.1 Requirements ID2 3](#_Toc475133507)

[1.2 Mini Milestones for ID2 5](#_Toc475133508)

[2.0 Time Estimations 6](#_Toc475133509)

[3.0 Design 7](#_Toc475133510)

[3.1 API Document 7](#_Toc475133511)

[3.2 Updated Data Structures for ID2 7](#_Toc475133512)

[3.3 Development Unit Test Plan 7](#_Toc475133513)

[4.0 Testing Document 8](#_Toc475133514)

[4.1 Test Plan 8](#_Toc475133515)

[4.2 Test Report 10](#_Toc475133516)

[4.3 Defect Report 10](#_Toc475133517)

[4.4 State Transition Diagrams ID2 11](#_Toc475133518)

[4.5 Critical Use Cases 12](#_Toc475133519)

[5.0 Coding Style Guide 14](#_Toc475133520)

[6.0 Build Report 15](#_Toc475133521)

[7.0 Upcoming Requirements 16](#_Toc475133522)

[7.1 ID3 Priority Requirements 16](#_Toc475133523)

[7.2 Requirements Changes 17](#_Toc475133524)

[7.3 Future Requirements 20](#_Toc475133525)

[8.0 Triage Meeting ID2 21](#_Toc475133526)

# 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**:

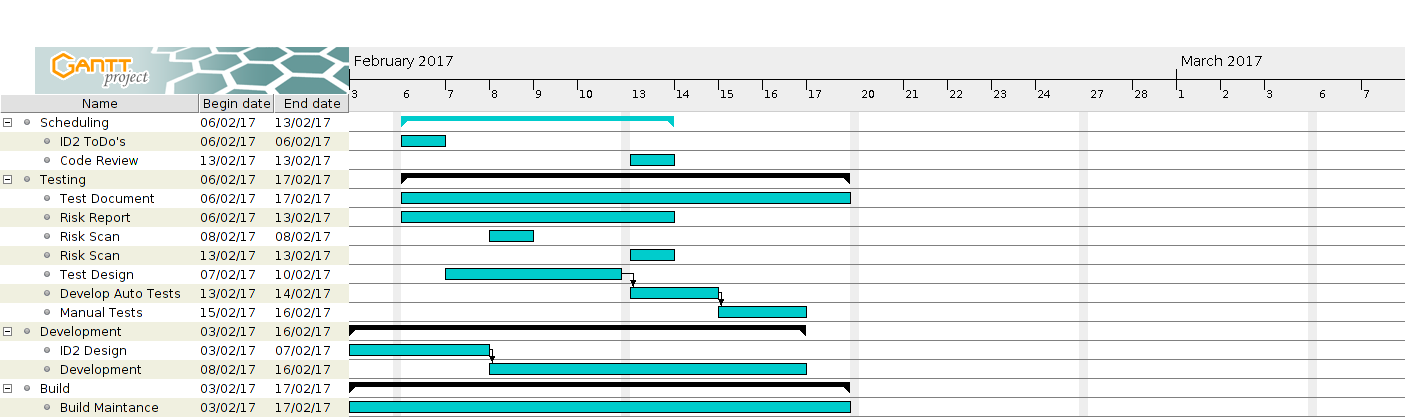
* Fix Protractor issues with sending keys, browsing tabs, etc. ✔
* Test plan for future ID's ✔
* Manual testing ✔
* Update End-To-End tests, test matrix, and defect report ✔
* Use case and state transition diagram updates ✔
* Teach test team to use tools ✔

**Documentation**:

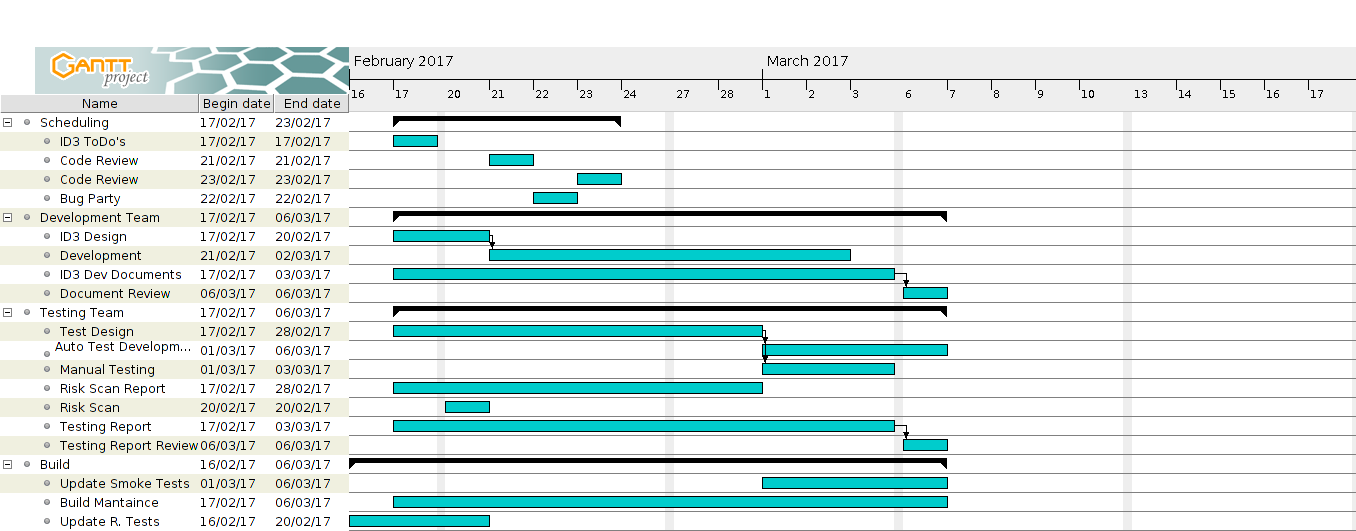
* Compile documents ✔
* Edit documents ✔

# 2.0 Time Estimations

Gantt Diagram Time Estimation ID2:



Gantt Diagram Time Estimation ID3:



In ID3, we will be adding time estimations for each assigned task. Trello tasks will be updated to contain time duration labels. These labels should be filled in by developers.

# 3.0 Design

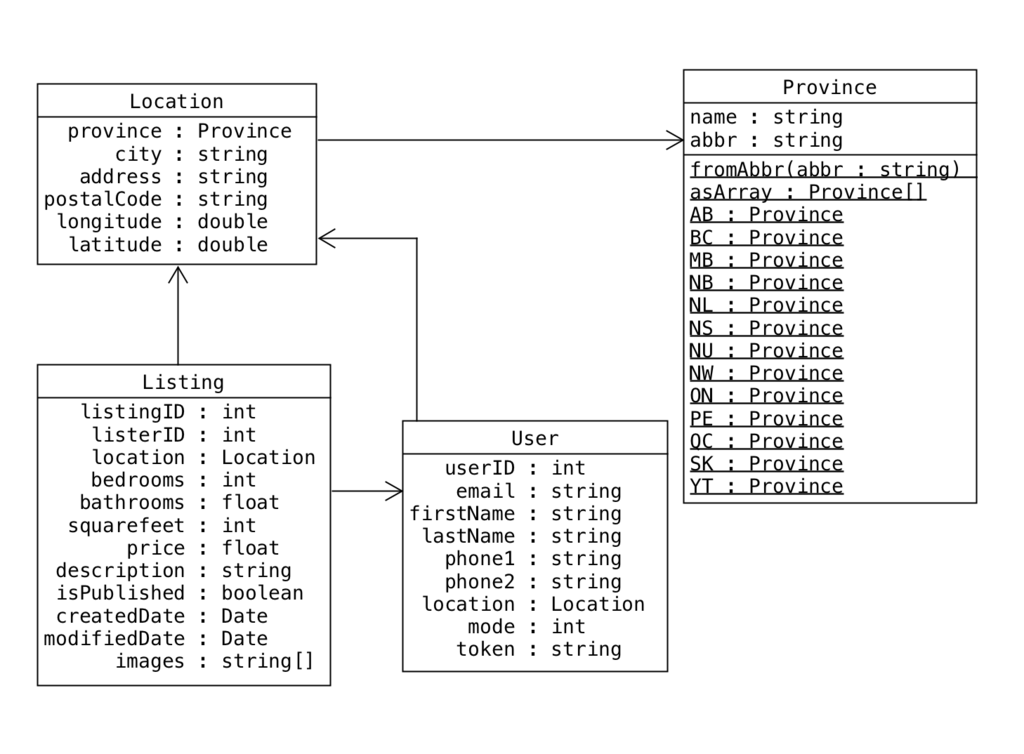
## 3.1 API Document

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 calls to the server. These calls send a JSON body with relevant data. 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 body containing the requested information. The following link contains our API document and details on the various calls to our database.

Note: We are currently experiencing difficulty with this link. Please copy paste it into your web browser.

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

## 3.2 Updated Data Structures for ID2



## 3.3 Development Unit Test Plan

The procedure to be followed by all developers for writing unit tests can be found at the following link:

<https://github.com/CMPT371Team1/Project/wiki/Testing-Plan>

# 4.0 Testing Document

## 4.1 Test Plan

4.1.1 Test Team Responsibilities

* Strong communication of what has been tested, what we are planning to test, etc. so that no two testers are working on the same test suite
* Creating test cases and manually testing them
* Automating the test cases created
* Updating the test matrix and defect report

4.1.2 Test Strategy

**Critical Items:** User Registration, Browse/My Listings/My Favourites Filters, Settings, Add Listing screens.

**Other Items:**

* Manually check that all screens are behaving as intended and that images are showing properly
* Check for any graphical issues or unintuitive features
* Make sure navigation is working as intended and check for performance issues

**Specifics (in the form of use-cases):**

* User input fields, specifically phone number fields, price fields, and filter pages:
  + Allowable keystrokes (no characters allowed when entering phone numbers, prices, and filter ranges)
  + Disallow negative input when specifying prices, and disallow the min/max price in filter to go below 0
* Ensure province filter works as intended
* When signing out after successful login, ensure user info is saved for next use
* Ensure that changes to user settings are saved and the relevant information for personal listings have been changed as a result
* Ensure user password change is successful

**Check for invalid inputs in the forms:**

* Disallow symbols to be entered in certain fields (bedroom/bathroom fields)
* Allowable symbols: “-” for phone number, “@”, “\_”, “-”, and “.” for email address. Other symbols are allowable in email addresses, but most email providers do not allow these symbols on email creation
* If no character limit is set for the description, ensure that it goes through stress testing. Example: Try to break the system by entering large amount of characters.
* As a QOL change, should look into making separate description input box (rather than the one we have now)

**End-to-end testing:**

Using Protractor with Jasmine framework. Browser testing using Chrome.

**Automatic unit testing:**

No longer using Karma with Jasmine framework.

4.1.3 Training Requirements

All testers must gain familiarity with the testing tools, specifically Protractor and Istanbul. Protractor is used for end-to-end test automation, and Istanbul is used to check test coverage. Testers will write JavaScript tests for the test configuration file to use. A meeting for the test team on February 9th will go over Protractor specifically. Istanbul is a simple command line test coverage tool that outputs test coverage and requires no prior knowledge. Karma is no longer being used as our unit test runner.

4.1.4 Documentation

**Test Matrix:**

The X-axis represents what is being tested, and the Y-axis represents the requirements

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Sign-In** | **Sign-Up** | **Browse Listings Filter** | **Add New Listing** | **My Profile** | **Browse** | **My Favourites** |
| **Add a listing in ‘Seller Mode’** |  |  |  |  |  |  |  |
| **Registering a new user** |  |  |  |  |  |  |  |
| **Browse in ‘Buyer Mode’** |  |  |  |  |  |  |  |
| **Edit account information** |  |  |  |  |  |  |  |

**Defect Report:**

|  |  |
| --- | --- |
| **Bug:** |  |
| **Reproducibility:** |  |
| **Steps taken:** |  |
| **Variations:** |  |
| **Screenshot:** |  |

Testers must follow the guidelines below when producing a bug report:

* + List the bug
  + Is the bug reproducible? To what certainty?
  + List the steps to re-create the bug (input data, prior usage of application, etc.)
  + Note any variations that may occur while attempting to recreate. A screenshot can be taken.

4.1.5 Schedule

**ID1:**

Most testing will be done manually on the GUI. The test team will begin creation of automated tests with Protractor. All data should be recorded in the test matrix and all defects should have an accompanying defect report.

**ID2:**

The Test Team will integrate automated end-to-end tests with TravisCI, update the defect report, update flow charts to better reflect requirements changes/current state of the system, and re-create, simplify, and update the test matrix.

**ID3 and Future ID’s:**

Organize a bug party

Continue updating the test matrix, defect report, and flow charts

Compatibility testing for iOS and Android

Performance testing

## 4.2 Test Report

The following summarizes various observations and measures taken by the Test Team in ID2:

* Password on Sign-in and Change Password pages clears on re-entry. This feels awkward to use and is uncommon
* The app should disallow negative values when filtering by price
* The app should prevent the user from entering characters and symbols in the phone number field
* "Add more listing" on the My Listings page should be named something else
* A confirmation check should be added if the user tries to leave the Add Listing page without saving
* Separators should be used on large numbers for improved readability (ex. 1 300 500, or 1,300,500)

## 4.3 Defect Report

|  |  |
| --- | --- |
| **Bug:** | Broken Icons |
| **Reproducibility:** | One-time occurrence |
| **Steps taken:** | 1. “test” entered into E-mail field on Sign-in page 2. “Sign-in” button selected. |
| **Variations:** | Icons reappeared in every attempt to recreate |
| **Screenshot:** | None taken |

## 4.4 State Transition Diagrams ID2

C:\Users\Arianne\Desktop\CMPT371_TestTeamDocs_ID2\KasperUIFlowDiag1_ID2.png

C:\Users\Arianne\Desktop\CMPT371_TestTeamDocs_ID2\KasperUIFlowDiag2_ID2.png

## 4.5 Critical Use Cases

The following critical use cases outline the low level details of the most important features of our application.

|  |  |
| --- | --- |
| **Use Case Component** | **Description** |
| Use Case Name | ID2: Add Trailer Listing in Saskatchewan |
| Primary Actor | Seller |
| Pre-conditions | User is in seller mode  On “My Listings” page |
| Success End Conditions | Listing is added to browse page |
| Failed End Conditions | Listing is not added to browse page |
| Main Success Scenario  A: Actor  S: System | 1. A: Selects “Add more listing” button 2. A: Selects “Saskatchewan” 3. A: Inputs “Saskatoon” in City/Town field 4. A: Inputs “123 First Street East” in Address field 5. A: Inputs “FLS 9U8” in Postal Code field 6. A: Inputs “130000” in Price field 7. A: Inputs “600” in Sq. Feet field 8. A: Inputs “5” in Bed Rooms field 9. A: Inputs “2” in Bath Rooms field 10. A: Inputs “Nice trailer with a little dust. Fixer upper.” 11. A: Adds 1 Image to images 12. A: Selects “Publish” 13. S: Adds the listing to database |
| Extensions | I.-XIII. A: Saves information prematurely  XI. A: Does not input an image |
| Priority | Critical |
| Response Time | < 1 second |
| Notes | Disallow < 0 in Price, Bedroom, Bathroom, Sq. Ft fields. |

|  |  |
| --- | --- |
| **Use Case Component** | **Description** |
| Use Case Name | ID2: Registering a new user |
| Primary Actor | Seller or buyer |
| Pre-conditions | Starts on Sign in Screen |
| Success End Conditions | A new user is registered |
| Failed End Conditions | A new user is not registered |
| Main Success Scenario  A: Actor  S: System | 1. A: Selects: "Register here" 2. S: Page switches from Sign-In to Register page 3. A: Inputs “testEmail@email.ca” in E-mail field 4. A: Inputs “testEmail@email.ca” in Confirm E-mail field 5. A: Inputs “testPassword1” in password field 6. A: Inputs “testPassword1” confirm password field 7. A: Inputs "John" into First Name field 8. A: Inputs "Smith" into Last Name field 9. A: Inputs "3065551234" into Phone Number field 10. A: Inputs “Saskatoon” in City field 11. A: Selects Province drop down list 12. S: Lists Province Options 13. A: Selects “Alberta” option 14. A: Selects “OK” button 15. A: selects “Register” button 16. S: Adds the user to the database |
| Extensions | II. - IV. A: Does not input matching fields  II., III. A: Does not input strong enough password  XV. S: User is already registered in database |
| Priority | Critical |
| Response Time | < 1 second |
| Notes | - E-mail provided must follow standard email name structure  - Passwords must include at least one capital letter and at least one number  - Fields are not checked for non-numbers in phone number field |

|  |  |
| --- | --- |
| **Use Case Component** | **Description** |
| Use Case Name | ID2: User Login |
| Primary Actor | Buyer/Seller |
| Pre-conditions | User has already registered an account, and had previously logged out from the system. |
| Success End Conditions | The user logs into the system. |
| Failed End Conditions | The user fails to login, and a prompt is given detailing the failed login. |
| Main Success Scenario  A: Actor  S: System | 1. A: Inputs “test” into e-mail field 2. A: Inputs “password” into password field. 3. A: Selects “Sign In” button 4. S: Attempts to log the user in using input credentials |
| Extensions | I. A: “test” is not entered, and fails to Sign-In |
| Priority | Critical |
| Response Time | 5 seconds or less, depending on how busy the server is. |
| Notes | * Currently, the only way to get past Sign-In screen is to use “test” as the E-mail field, or by going through Register page. |

# 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 front-end smoke tests are in process of being implemented using Protractor. We are currently experiencing difficulty due to a large number of package dependencies, and are working to troubleshoot the problem. The back-end does not currently have smoke tests implemented due to time constraints. The goal for ID3 is to have one branch compile and run smoke tests. If these tests pass, then they should be pushed to another branch for more rigorous testing, including automated unit tests and debugging. As the system evolves and more UI components are added, the smoke test will grow to incorporate these new UI features. We do not expect the back-end smoke test to undergo much change beyond ID2.

Build Status:

The builds for iOS and Android are running and simulating correctly. The Linux build is currently undergoing a smoke test. The Server is under construction and is being hosted locally for now. Its main purpose in ID2 is for testing. End-to-end smoke tests are being run using Protractor on the Firefox browser. Until now, all testing has been done on the browser version of our app; smoke tests are currently not in place for the other platforms.

SDKs, 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 manager has a firm understanding of automated testing, deployment, 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 7D1014Server:
* 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
* End to End Tests
  + Protractor v5.1.1
  + Firefox v47.0.1
  + Selenium v3.1

Corodova 6.5.0 requires both jdk 1.8 (or higher), as well as 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. Google Cloud is required for the Google App engine, the platform our servers are built upon. The server uses Python 2.7, the most recent version of Google Cloud SDK, and Python extensions for Google Cloud SDK. Our end-to-end tests are driven by Protractor, which sends to Ionic through the Selenium server. We are currently testing our system on Firefox v47.0.1, because of compatibility issues with newer versions. All developers and testers have been set up with the latest versions of the required 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)

Releases for our build can be found at the link below:

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

# 7.0 Upcoming Requirements

## 7.1 ID3 Priority Requirements

Front-end Requirements (Not finished during ID2):

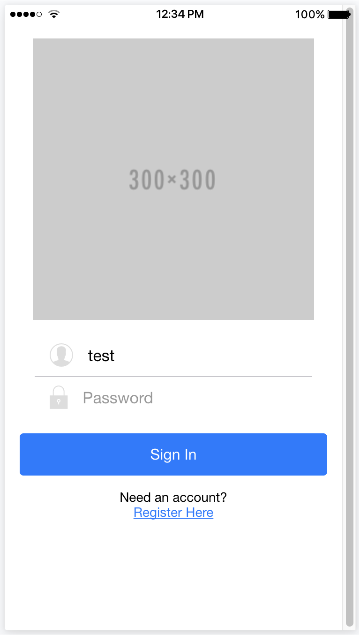
* Receive notifications regarding Favourites
* Price changes
* Listing removed/edited
* Seller Upload images

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

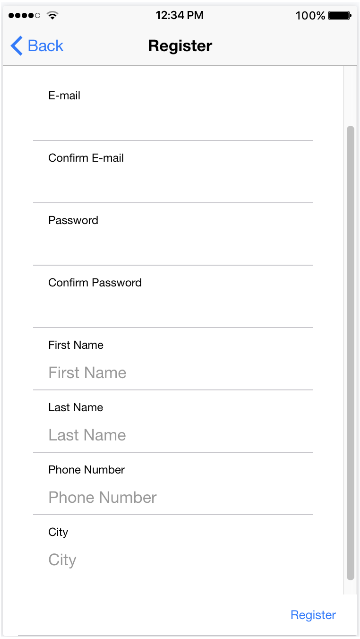
## 7.2 Requirements Changes

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.



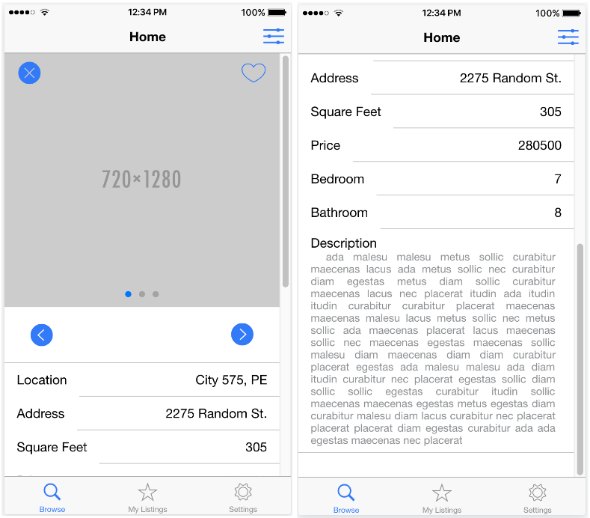
**Sign in Screen:**

* 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

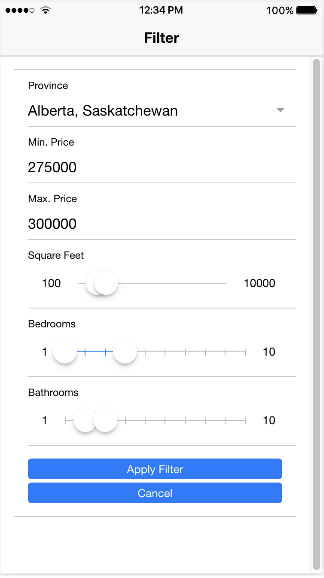


**Sign up Screen**:

* 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

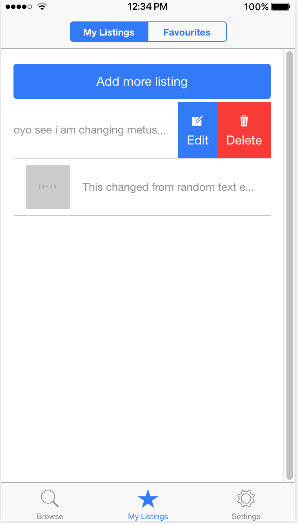
**Browse Screen**:

* 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)



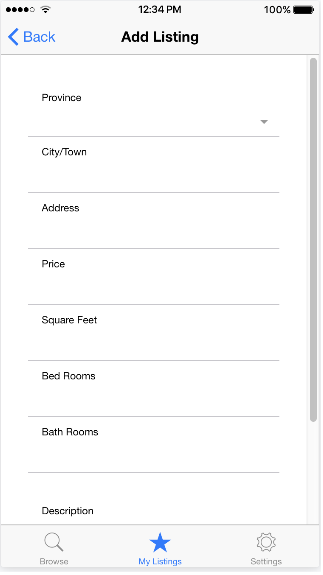
**Filter Screen**:

* 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



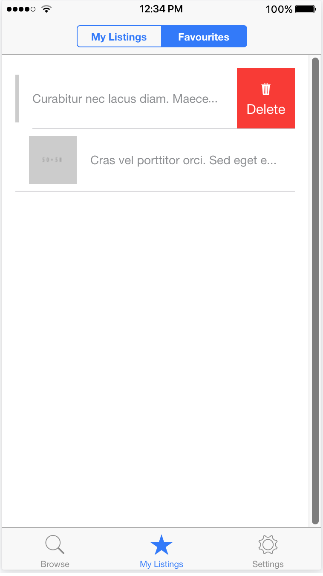
**My Listings Screen**:

* 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



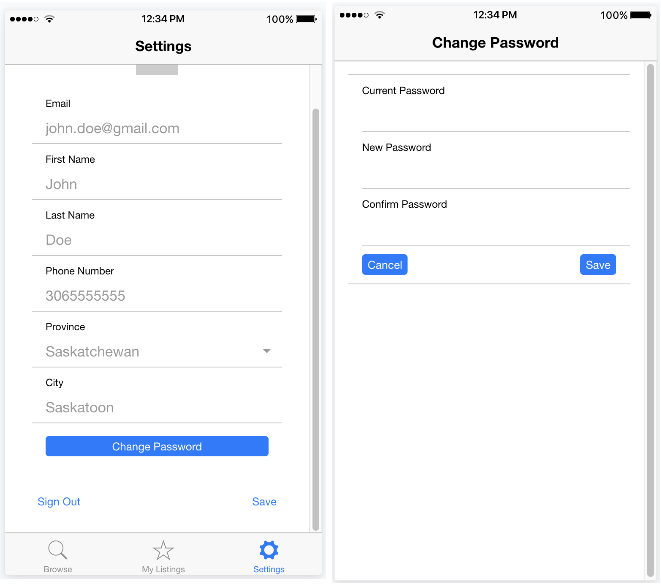
**Add Listings Screen:**

* 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

This section outlines requirements identified for upcoming ID’s, some of which will take priority over others, and some of which may not be implemented 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 ID2

**Date**: Feb 17, 2017

**Start Time**: 1:30pm

**End Time**: 2:30pm

**Location**: Spinks320

**Members Present**: Tushita Patel (Project Lead), Kristof Mercier and Dylan Prefontaine (Dev Leads), Jeremy Liau (Test Lead) and Chris Mykota-Reid (Build Manager)

**Summary**:

**1) Front-end**

The development team was unable to meet requirements because of the dependency on the back-end documentation. This is an identified risk for this ID. Instead, the front-end developers worked on the following tasks:

* Added provinces to register screen
* Researched saving information to the device and built a spike prototype as part of this research
* Other minor changes:
  + Changed tabs
  + Set goals for ID3 – back-end to front-end and back-end to back-end
* Integrated Sign-up, Sign-in and Add Listing with the back-end

**2) Back-end**

The following tasks were finished for ID2:

* Data structures
* Unit testing setup
* Ran unit tests
* WebApp interface
* Implemented Sign-in, Sign-up, Add Listing
* Three API calls implemented, tested and documented
* Eight more API calls documented but not implemented
* Finished API, Style and Unit Test documentation

The following tasks were not finished for ID2:

* Implement the aforementioned eight API calls
* Addition of more API calls

**3) Testing**

The following tests were done for ID2:

* Manual testing,
* Test matrix update
* Update path coverage diagrams
* Testing plan documentation
* End-to-end testing
* Critical use cases
* Testing results and defect report

Unit testing could not be carried out because Karma (the program which runs our unit tests) is not set up. To mitigate this loss, the developers built a separate way of unit testing for this ID.

**4) Build**

* The server build is fully implemented and running
* The smoke test is still in the early processes of being implemented. We aim to have it finished by the end of this weekend (February 19th), or by mid-ID3 (February 24th) at the latest.