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 2](#_Toc474973829)

[1.1 Requirements ID2: 2](#_Toc474973830)

[1.2 Mini Milestones for ID2 5](#_Toc474973831)

[1.3 Use Cases: 6](#_Toc474973832)

[2.0 Design 6](#_Toc474973833)

[3.0 Testing Plan 7](#_Toc474973834)

[4.0 Test Report 8](#_Toc474973835)

[5.0 Coding Style Guide 8](#_Toc474973836)

[6.0 Build Report 8](#_Toc474973837)

[7.0 Defect Report 9](#_Toc474973838)

[8.0 Upcoming Requirements 9](#_Toc474973839)

[8.1 ID3 User Requirements: 9](#_Toc474973840)

[8.2 Future Requirements (Nice to Haves) 9](#_Toc474973841)

[9.0 Triage Meeting: 10](#_Toc474973842)

# 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
* 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

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.

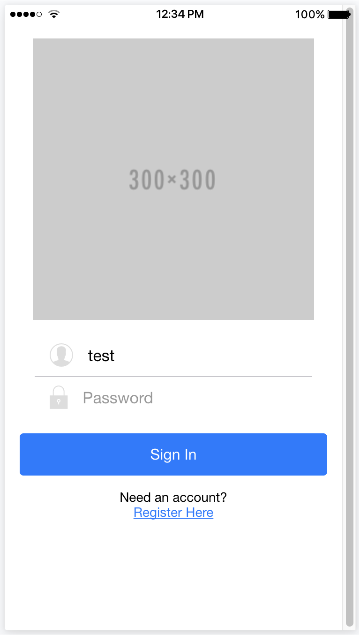
**Software:**

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

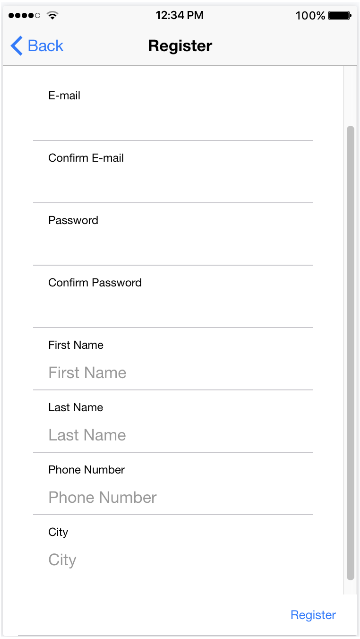
Requirements Changes

This section highlights the upcoming changes to our UI since ID1. These changes were discussed and agreed upon during Dylan’s formal code review. 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.



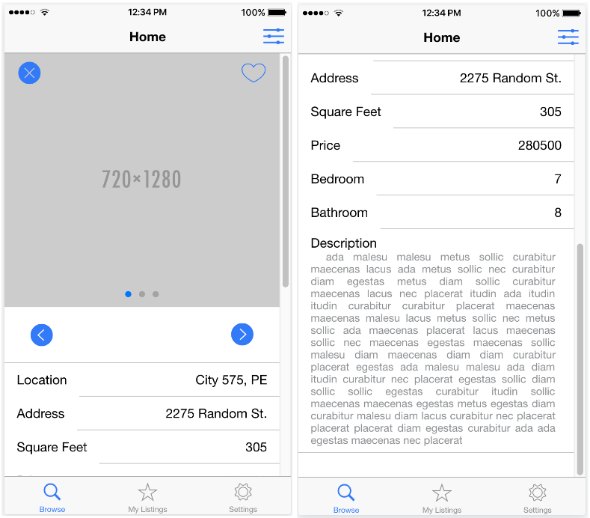
**Sign in Screen: 🡪** check for name changes

* The Sign in Screen will no longer be the entry point of the app, and the Browse Screen (formerly known as the Home Screen) will take its place
* Get legit icon 🡪 more detail
* 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 (ability to contact or ability to add contact?)
* Add map capability
* Add Edit button, if the Listing belongs to the user
* Replace Filter button
* Improve Dislike button
* Remove star (the My Listings start? To be replaced with what?)
  + - 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)

## 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 the Back End Development Environment

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:

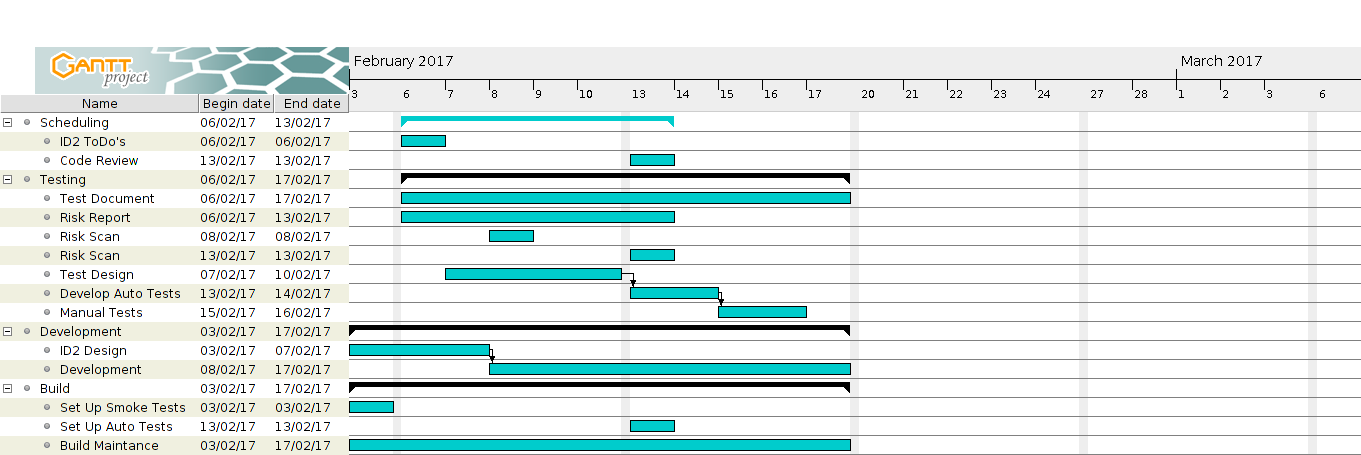
* Compile documents
* Edit documents

## 1.3 Use Cases:

# 2.0 Design

* Changed data structures (new UML diagram)
* API Doc from Gaurav – post a link
* 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)

# 3.0 Testing Plan

Gantt Diagram ID-2:

Test Matrix:

The Testing Matrix (insert definition) can be found at the following link:

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

# 4.0 Test Report

# 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/Documentation/blob/master/ID1-Documentation/OtherDocuments/CodingStyleGuide>

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/(Rough)-Coding-Style-Example-(JavaScript)>

# 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-1. 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 ID-1. Server builds are currently not in effect, because the team will begin back-end requirements in ID-2. All builds are currently slow, with build time reduction a priority for ID-2.

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
* 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

Corodova 6.5.0 requires both jdk 1.8 (or higher), and 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. All developers and testers have been 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)

# 7.0 Defect Report

# 8.0 Upcoming Requirements

## 8.1 ID3 User Requirements:

## 8.2 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

# 9.0 Triage Meeting:

**TO DO:**

Future Requirements:

* UI doc
  + Changes/updates to the original GUI mock-ups. Combine this with Dylan’s code review and formalize
  + UI stuff 🡪 restructured some things to fit better with our architecture for server interactions
    - Restructured things so there wasn’t duplicate code
    - Some minor requirements changes
* Dylan’s code review –notes from Ryan and Dylan – use this to show changes
* API doc
  + Outlines client server communication. 13 (10 currently) API calls (calls to the server). 13 server functions that you can call. Robust and complex function calls, send server a JSON, (for sign in user, send server a JSON with PW and email and server will receive request and take info and proves it, go to database get data and send reply back to client (called token – has status and JSON))
  + A set of routines, protocols, and tools for building software
* ~~Style Guide~~ 
  + ~~Get from Kristof~~

Talk to Gaurav and Tian about backend:

* Got the back end set up 🡪 ask Tian and Gaurav
* Got the environment for back end set up
  + PyCharm, got the plug-ins
  + GAURAV OR TIAN 🡪 Google App Engine 🡪 Web app engine (framework) that does stuff for our system like password security

Finalize all mini milestones with leads and Tushita

Find out who is doing the UML diagrams for ID2 Design

Ask Jeremy to define test matrix