KASPER – ID3 PROJECT DOCUMENTATION

SOFTWARE DEVELOPMENT TEAM:

Project Manager: Tushita Patel

Dev Lead: Kristof Mercier/Dylan Prefontaine

Test Lead: Jeremy Liau

Build Manager: Christopher Mykota-Reid

Developers: Gaurav Arora, Haotian Ma, Melody (Tian) Zhao

Test Team: Christopher May, Ryan Tetland

Documentation: Arianne Butler

Contents

[1.0 Requirements Document ID3 2](#_Toc476332814)

[1.1 Front-end Requirements 3](#_Toc476332815)

[1.2 Back-end Requirements 3](#_Toc476332816)

[1.2 Mini Milestones for ID3 4](#_Toc476332817)

[2.0 Time Estimations 5](#_Toc476332818)

[3.0 Design 6](#_Toc476332819)

[3.1 API Document 6](#_Toc476332820)

[Updated Data Structures for ID3 7](#_Toc476332821)

[3.3 Development Unit Tests 8](#_Toc476332822)

[4.0 Testing Document 8](#_Toc476332823)

[4.4 State Transition Diagrams ID3 (to be updated) 9](#_Toc476332824)

[5.0 Coding Style Guide 10](#_Toc476332825)

[6.0 Build Report 10](#_Toc476332826)

[7.0 Upcoming Requirements 12](#_Toc476332827)

[7.1 ID4 Requirements 12](#_Toc476332828)

[7.2 Future Requirements 12](#_Toc476332829)

[8.0 Triage Meeting ID3 13](#_Toc476332830)

# Requirements Document ID3

## 1.1 Front-end Requirements

Priority (Not Finished in ID2)

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

Front-end Requirements Changes

This section highlights the changes to our UI since ID2. These changes were discussed and agreed upon during both the current and previous ID’s, and have been implemented during ID3.

Finalized changes to the UI for ID3 can be found at the following link:

Insert new link and description from Kristof

## 1.2 Back-end Requirements

Priority (Not Finished in ID2)

* Email verification (not done)
* Forgot password (not done)
* Change password
* Get all Listings for Browse page (removed)
* Get filtered Listings
* Edit Listings
* Like/Dislike
* Edit Account
* Get Favourites Listings

Back-end 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

Back-end Requirements Changes

This section highlights the changes to our back-end since ID2. These changes were discussed and agreed upon during both the current and previous ID’s, and have been implemented during ID3.

* Remove cursor and lastListing from the getListings API call
* Allow optional parameters to be passed to the getListings API call
  + Filter data structure
    - If the filter is not given, a default filter will be applied
  + "included fields"
    - Tells the API which fields of the selected listings should be returned.
    - Include a "featureImage" option which will return the single featured image as opposed to all of the images
  + "limit"
    - The number of listings to be returned
    - By default, only returns id's
* EditListing call must update modified date and return it in the response JSON
* signInWithToken and signIn need to update the "lastSeen" field in the user table.
* The server must be updated if a seller’s listing is modified on the device, and the device must be updated when a favourited Listing is changed on the server. Check for validity must be done on front and back end. If the back-end is passed an invalid listing, it should not be accepted, and the device’s Listing entry should be reverted by force. This will only happen with API misuse, so this behaviour is acceptable.
* A user specific rate limit should be enforced to ensure the API is being used correctly, and to defend against brute force password cracking. There is a default overall rate limit, but we should look into implementing a user specific one as well.
* Add a boolean “verified” field in the user table to indicate if the user has verified their email. This must be updated when they change their email, or when they click an activation link.
* "signInWithToken" should return everything that "signIn" returns.

## 1.2 Mini Milestones for ID3

**Development**:

* Finish all new requirements for front and back-end on Trello boards
* Have the server up on a USASK machine ✓
* Have the front-end communicating with the back-end (server)
  + This has been achieved with Sign in and Sign up only ✓
* Make two test accounts, and dummy Listings ✓

**Testing**:

**Build**:

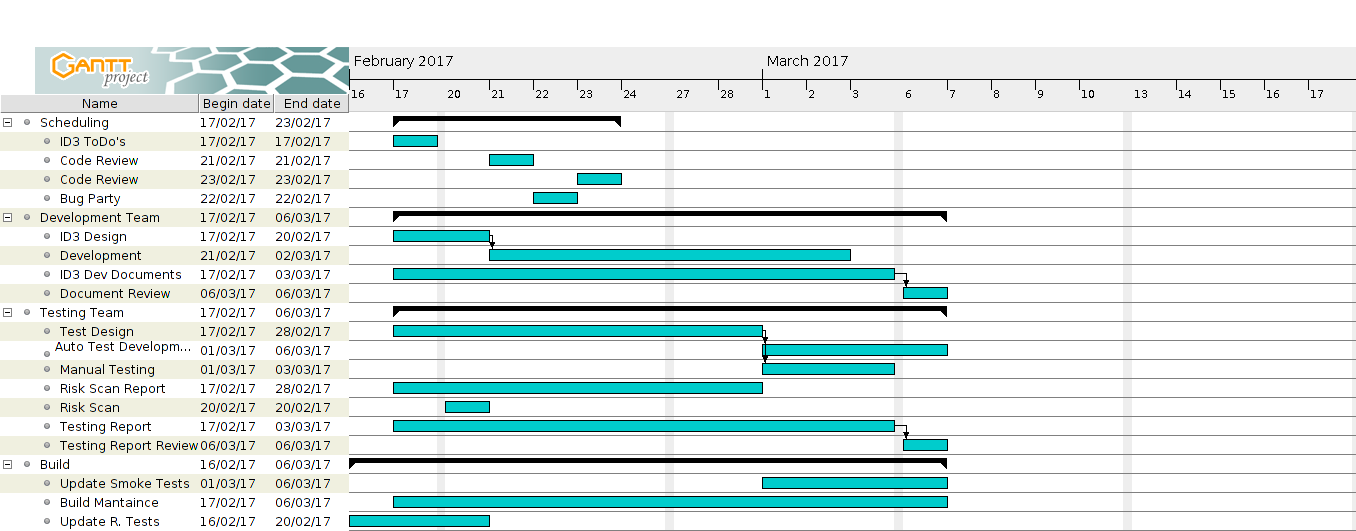
* Implement front-end smoke tests ✓
* Implement back-end smoke tests ✓
* Set up multi-stage build ✓

**Documentation**:

* Organize, compile, and edit all documentation ✓

# 2.0 Time Estimations

Gantt Diagram Time Estimation ID3:



To see time estimations on all individual tasks, please follow the link in Process Documentation, section 2.0 Activity Log, and click on the bottom tab labelled “Individual Activity Log”.

For ID3, development time estimations have been added for each assigned task on Trello. Trello cards now contain a coloured label indicating the approximate estimation of each task. These estimates were approximated by the dev team.

The following image shows the colour coded time estimation scale with which each trello card will be rated.



# 3.0 Design

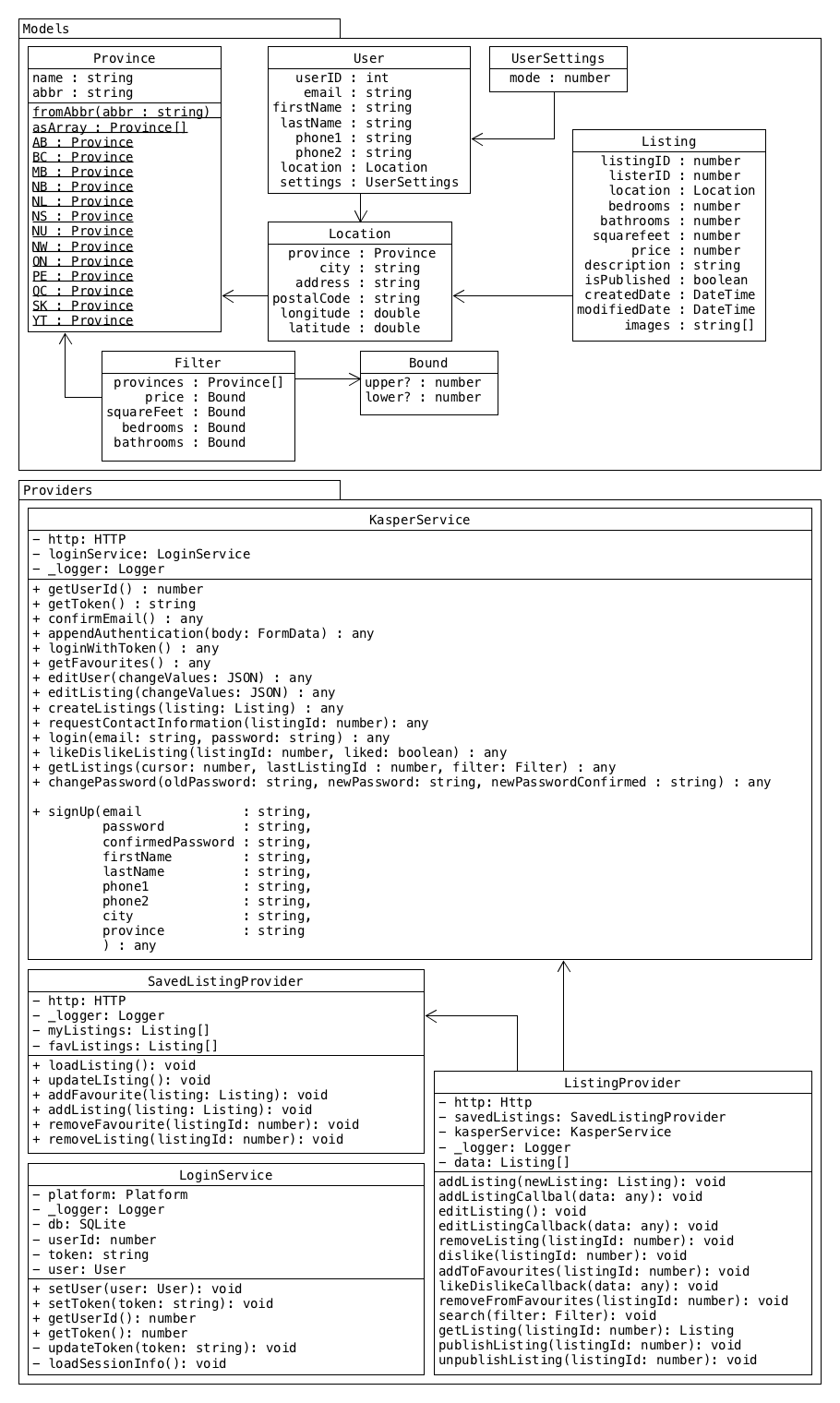
## 3.1 API Document

The API document outlines the client server communication of our system. Our project will contain sixteen API calls (calls to the server), eleven of which are currently implemented, and two of which are in full communication with the server. 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 updated ID3 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)

## Updated Data Structures for ID3



## 3.3 Development Unit Tests

The procedure to be followed by all developers for writing both front and back-end unit tests respectively, can be found at the following links:

Front-end Guide for Unit Testing:

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

Back-end Guide for Unit Testing:

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

# 4.0 Testing Document

## 4.4 State Transition Diagrams ID3 (to be updated)

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

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

# 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:

Front-end tests are now implemented using protractor and are currently failing due to changes in requirements. The back-end smoke tests are implemented and failing due to problems with back-end implementation.

Build Status:

The builds for iOS and Android are running and simulating correctly. Both the Linux build and remote server are currently undergoing smoke tests. The server is now integrated with the front-end for two of our API calls. End-to-end smoke tests are being conducted using Protractor on the Firefox browser. The back-end is being tested with Python scripts. 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 v5.0.0
  + 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 ID4 Requirements

Front-end Requirements

* Form control validation for all remaining pages
* Replace descriptive text with icons
* Fix description box in Add Listings
* Add Google Maps section to Detail page
* Display currently added images
* Get a legitimate Kasper icon
* Finalize method of ordering images
* Make Favourites page the same style as Browse page
* Get front-end fully integrated with back-end
* Finalize and complete UI flow

Priority Back-end Requirements

* Email verification (not done for ID3)
* Forgot password (not done for ID3)
* Confirm email
* Logout
* Contact Seller
* Sign in / Sign up using Facebook
* Delete Listing
* Get the front-end fully integrated with the back-end for all API calls

Other Back-end Requirements

* Thorough review of test cases for all back-end code
* Push notifications if something changes regarding a Listing saved in Favourites (a change in the database triggers a notification)
* Integration with Google Maps

## 7.2 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.

* Send a message to the seller including contact information
* Users can sign up to receive “hot list” notifications
  + Feature Listing (paid for by Sellers)
  + Newly added Listings
  + Price changes on Favourites
  + Based on previous search history
* Sellers receive notifications regarding personal Listings:
  + When a Listing is saved to Favourites
  + When someone requests a viewing
  + When someone sets a price watch
* Super Admin User
  + Log in as Super Admin
  + Add new Listing under any user
  + Edit any Listing
  + Remove any Listing

# 8.0 Triage Meeting ID3

**Date**:

**Start Time**:

**End Time**:

**Location**:

**Members Present**:

**Summary**:

**1) Front-end**

**2) Back-end**

**3) Testing**

**4) Build**