KASPER – PROGRAMMER DOCUMENTATION

SOFTWARE DEVELOPMENT TEAM:

Project Manager: Tushita Patel

Dev Lead: Kristof Mercier, Dylan Prefontaine

Test Lead: Jeremy Liau

Build Manager: Christopher Mykota-Reid (ChrisMR)

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

Test Team: Christopher May (ChrisJ), Ryan Tetland

Documentation: Arianne Butler

Contents

[1.0 Introduction 3](#_Toc478933396)

[2.0 System Overview 3](#_Toc478933397)

[2.1 Tools and Frameworks 3](#_Toc478933398)

[2.2 Database Data Structures 3](#_Toc478933399)

[2.3 UI Sequence Flow Diagrams 3](#_Toc478933400)

[3.0 Code Base 4](#_Toc478933401)

[4.0 Installation Instructions 4](#_Toc478933402)

[5.0 Future Programmers 5](#_Toc478933403)

[5.1 Unimplemented Requirements 5](#_Toc478933404)

[5.2 Known Bugs 6](#_Toc478933405)

[5.3 Tested/Untested Areas 6](#_Toc478933406)

[5.4 Tricky Areas 6](#_Toc478933407)

[5.5 Integral Areas 6](#_Toc478933408)

[6.0 Deployment and Maintenance 6](#_Toc478933409)

[6.1 Deployment 6](#_Toc478933410)

[6.2 Maintenance 7](#_Toc478933411)

# Introduction

Kasper home app is a real estate application that facilitates communication between sellers and prospective buyers. Sellers refers both to home owners wishing to sell, and to real-estate agents selling on their behalf. Buyers refers to users seeking to purchase a home, and to real-estate agents looking on their behalf. Sellers can post information and images of their properties (called Listings), and buyers can browse through all published Listings; filtering, liking, and disliking as they please. If a user likes a Listing, it will be saved to their Favourites. If a user dislikes a Listing, it will be removed from their Browse page permanently. Buyers should be able to initiate contact with sellers by sending a message containing their email and phone number. Although applications with similar functionality exist, Kasper home app aims to fill the gap for individuals wishing to buy or sell exclusively, purposely excluding renters and leasers. By dividing the market between buyers and renters/leasers, Kasper home app hopes to provide a direct, easy-to-use link between sellers and buyers.

# System Overview

## 2.1 Tools and Frameworks

The front-end uses Node JS, which supplies the NPM package manager. NPM provides both Cordova and Ionic. Cordova builds the application on the three mobile platforms (iOS, Android, and Windows), as well in the Browser. Ionic extends Cordova and is built on Angular2; a development platform that uses Typescript, SCSS, and HTML.

The back-end uses Google App Engine and Python 2.7 for the API calls. The interface for Google App Engine is provided by Web App. Web App is extended by Web App 2, which provides URI routing and better exception handling. Back-end testing is handled by the unittest module in Python. To create our mock test server, we are using Test Bed from the Google App Engine.

Test frameworks include Node JS and NPM which access Jasmine and Protractor. End-to-end tests are written in Jasmine and run on Protractor. We are employing Gulp to toggle a line in the kasper-config.ts file, so that testers can switch between the real database and the test database.

The Build uses Travis CI. For further information regarding build frameworks, see the Build Report on page \_\_ in the following document:

* Insert link to ID5 Project Doc

## 2.2 Database Data Structures

* 5 diagrams for database data structures (Gaurav) – all back-end related

## 2.3 UI Sequence Flow Diagrams

* Tian

# 3.0 Code Base

Below is a link to the project’s GitHub:

<https://github.com/CMPT371Team1/Project>

* Front and back-end code is located in the src and server-gae folders, respectively

# 4.0 Installation Instructions

Front-end Setup

1. NodeJS can be installed at the following link:

<https://nodejs.org/en/download/>

NodeJS version 6 or higher is required. You can verify your currently installed version number by typing node -v in the command line.

1. Install Cordova and Ionic dependencies by typing npm install -g ionic cordova into the command line.  
   If you are not running your command prompt with administrator privilege, you may need to use sudo npm install -g ionic cordova
2. Change current working directory to project folder
3. run npm install
4. run ionic state restore
5. Run the front-end code using ionic serve

Back-end Setup

1. In the command prompt, run curl https://sdk.cloud.google.com | bash
2. Run gcloud components install app-engine-python
3. Run gcloud components install app-engine-python-extras
4. Python 2.7 or higher is required, and pip must be for version 2.7. To verify your currently installed versions of Python and pip, run python -v and pip -v, respectively
5. Run sudo pip install webapp2
6. Run sudo pip install webob
7. Run sudo pip2 install pyyaml
8. Either modify the PYTHONPATH variable (to what???) OR in PyCharm:
   1. Preferences
   2. Interpreter Project structure
   3. Add content root locate where google\_appengine was installed
9. To run the server, run the command Open Terminal in pyCharm dev\_appserver.py app.yaml (???)

Testing Set-up

* Jeremy:
  + How to set up test environment and integrate it with the code base

# 5.0 Future Programmers

## 5.1 Unimplemented Requirements

* Sign in / Sign up using Facebook
  + This feature is incomplete in the front-end, but complete in the back-end.
* Messaging Functionality­
  + This feature is incomplete in the front-end, but complete in the back-end. It includes functionality for sending initial messages to sellers, marking read messages, and deleting messages. Once implemented, it will need to be integrated with the back-end and tested.
* Edit listing
  + This feature is incomplete in the front-end, but complete in the back-end
* Push notifications
  + The app should be able to track changes in the database and trigger notifications to interested buyers
  + Sellers should receive a notification if a prospective buyer sends a message regarding one of their Listings
  + Buyers can elect receive “hot list” notifications for one or more of the following:
    - A Seller publishes a Feature Listing
    - A new Listing is added that fits some predetermined criteria (i.e. neighborhood, price range, seller, etc.)
    - Price change on a Favourited Listing
    - A Listing is published that contains similarities to past searches
  + Sellers can elect to receive notifications regarding their personal Listings:
    - A Listing has been Favourited by a user
    - A buyer sets a price watch on a Listing
* Integration with Google Maps
  + This feature should show Listings with regards to their physical location on a map. The Google Maps functionality should be accessible from the Detail page.
* Forgot password
  + This feature should allow a user to request an email or text message containing reset information
* Re-order a Listing’s images.
  + This feature should allow a seller to select a primary image to be displayed on the browse page, and to re-order a Listing’s images.
* Super Admin User
  + A Super Admin User can access the following additional functionality:
    - Publish a Listing under any user
    - Edit any Listing
    - Remove any Listing
* API Versioning
* Edit User and Delete listing may be a little messed up. Check up on this nearer to deadline.

## 5.2 Known Bugs

* Jeremy: Defect report

## 5.3 Tested/Untested Areas

* Jeremy
  + No iOS testing
  + Tested on Firefox and Chrome
  + Tested on android and browser
    - Sign in with Facebook functionality has unit tests but needs integration and end-to-end testing
  + See defect report for buggy areas that are not fully tested because of said bugs
  + Do test hooks from server side???
  + Different types of testing implemented

## 5.4 Tricky Areas

* Get Listings using Filter Functionality:
  + It is not possible to query multiple variables at once with Google App Engine, so each variable needs to be queried separately. The intersection of the queries yields the correct results.

## 5.5 Integral Areas

* The check\_required\_valid function (provide the file name???)
  + This function is used by every API call and checks the JSON for correctness
* Get user by ID/password/token (provide file name???)
  + User.get\_by\_id: returns a User object
  + self.auth.get\_user\_by\_password: returns a dictionary of the user's information
  + User.get\_by\_auth\_token: returns a list where the User object is in the [0] element
* Initialize\_DB\_Testers (provide file name???)
  + This function reinitializes the test data base by deleting all data and creating dummy data for testing

# 6.0 Deployment and Maintenance

## 6.1 Deployment

* Gaurav

<https://cloud.google.com/appengine/docs/standard/python/getting-started/deploying-the-application>

<https://cloud.google.com/appengine/docs/standard/python/tools/uploadinganapp>

## 6.2 Maintenance

* Chris MR, Gaurav, Jeremy???
* Systems evolve after deployment
  + Continued improvements
  + Bug fixes