Software Requirements Specification

for

Housing4U

Version 1.0

Prepared by

Group Name: Team Vibrant

Austin Alameda	4600193	AustinAlameda@gmail
Trae Washburn	5138185	Traedog333@gmail
Site Mao	4820429	Site.mao1992@gmail
Jason Horsley	4678918	Jason.h42@gmail
Alvin Tan	5244868	Alvin9845@gmail

Instructor: Chandra Krintz

Course: CS 189A Capstone

Lab Section: Wednesday 6PM

Teaching Assistant: Geoffrey Douglas

Date: February 9, 2014

Contents

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Site Mao, Jason Horsley, Austin Alameda, Trae Washburn, Alvin Tan	Initial version	02/09/14
1.1	Site Mao, Jason Horsley, Austin Alameda, Trae Washburn, Alvin Tan	Added user stories	02/15/14
1.2	Site Mao, Jason Horsley, Austin Alameda, Trae Washburn, Alvin Tan	Updated UI elements and user stories	2/27/14

1 Introduction

Housing4U will be a website application that allows users to quickly search for vacant housing options in your area. It will have different features and tools to help the renter find the best listing possible for their preferences.
There are several services in place, which currently lack useful features, cohesion, and user base. Our app would encompass all of the different properties in your area and would also allow for easy comparison of prices as well as features similar to WalkScore, which gives the distance to restaurants, school districts and many other important businesses.

1.1 Document Purpose

The purpose of this document is to outline the requirements for the web app “Housing4u”. Housing4u will be built with the Ruby on Rails framework and use a PostgreSQL database. It will be accessible by any standard browser and will allow housing units to be viewed by users

1.2 Product Scope

The software is designed to connect people who are looking for a place to rent to houseowners who are looking to lease their property. Currently there are very limited options for finding a place to rent and we want to provide a platform for this. The benefits are that since this will be a dedicated service, we will be able to provide many additional functionalities such as matching with other strangers interested in a property, and linking it to technologies such as Google Maps to further streamline the process.

1.3 Intended Audience and Document Overview

This document is primary technical and designated for project managers, developers and testers. Since our application will ultimately be client-facing and be very easy and intuitive to use, this SRS will be for developers who are interested in adding additional functionality and for people who are interested in improving it.

1.4 Definitions, Acronyms and Abbreviations

- API (Application Programming Interface) - A generalized interface used to allow Housing4U to interact with external software.
- Bootstrap - A front-end framework used to simplify user interface design efforts.
- CAPTCHA (Completed Automated Program to Tell Computers and Humans Apart) - Image-based authentication used to verify a human user and minimize spam.
- Front-end - The aspect of the application that the user interacts with.
- Back-end - The aspect of the application the user does not interact with. It iis developed and maintained by development professionals.
- Gem - An external, packaged, 3rd-party tool used for back-end design.
- Heroku - An external, web-based platform used to deploy Housing4U on a live web server.
- HTTPS - encrypted communication protocol used to keep user sessions secure
- PostgreSQL - A back-end database tool used to store information.
- Ruby - The programming language Housing4U utilizes.

- Rails - The web-based framework Housing4U utilizes, built on top of Ruby.
- UI (User interface) - Front-facing aspect of the Housing4U application.

1.5 Document Conventions

This document uses Arial font throughout the entire document. Section titles use bold, size 18 font. Subsections use bold, size 14 font. Sub-subsections use bold size 12 font. All other text uses size 11 font.

1.6 References and Acknowledgments

- Bootstrap:
  - http://www.getbootstrap.com/
- jQuery
  - http://www.jquery.com
- Heroku:
  - https://www.heroku.com/
- PostgreSQL:
  - http://www.postgresql.org/
- Rails:
  - http://rubyonrails.org/
- Ruby:
  - https://www.ruby-lang.org/en/

2 Overall Description

2.1 Product Perspective

This project is intended to replace Craigslist, the product that (at the time of writing) currently dominates the rental ecommerce market. The project is built from scratch, and thus there is no initial foundation to build our system upon. The system will consist of a backend and a frontend. The backend will handle the storage of rental listing and user information data, as well as the querying and delivery of this data back to the frontend to be displayed. The frontend will allow a user to interact directly with a webpage using our Bootstrap UI, which will allow a user to specify a particular filter and then send a query to the backend to retrieve data.

The frontend of Housing4U will be a web application that displays formatted data to the end user. The web application will be built using Ruby on Rails, and will run on a Heroku platform for web services that interfaces with a PostreSQL server to retrieve and store housing data.

The backend of Housing4U will be a PostgreSQL server that will store data used by the frontend. The backend will never be accessed directly, but instead manipulated through use of the frontend, which will either query the database for filtered information, or insert new listing data into the database.

2.2 Product Functionality

- User can rapidly cycle through listing pictures
- User can login
- User can browse vacant apartments
- User can search/filter
- User can register as a renter or seller
- User can compare listing
- User can bookmark listings
- User can claim interested in listing
- User can rate listings

2.3 Users and Characteristics

We expect three different types of users for this project: Landlords, Potential Renters, and Current Renters.

Landlords are considered the most important users because they are the ones who post listings and provide the site with more content, viewers, and therefore monetization. Landlords will be identified by users who post vacant units for rent and upload different information about those units such as photos, price, and location.

Potential Renters are the second most important users of our site. They are also going to be the majority of site views because the web app is mainly focused on delivering content to them. Most potential users will simply browse listings until they find a vacant unit they are interested in, at which point they can contact the landlord for further details and negotiation.

Current Renters are the last type of viewers of our site. They may already be renting a unit and could be posting to acquire new roommates check for other potential listings.

2.4 Operating Environment

This is a web application and the various details about the operating environment is unimportant and hidden from the end user. End users will use the application from a modern internet browser such as Firefox, Chrome, IE10+.

2.5 Design and Implementation Constraints

- Limited server resources
- Limited bandwidth on Heroku
- Limited base housing data
- Browser compatibility
- API rate-limiting

2.6 User Documentation

The only help manuals required for our application can be accessed via a general “Help” interface that will explain many of the features of our website and how to use them. Our site will be fairly self-explanatory and easy to use, just as “craigslist.org” does not come with a user manual.

2.7 Assumptions and Dependencies

Assumptions to properly use the application include:

- Internet connection with enough bandwidth to fully render various displays
- A modern browser with up to date technologies, such as a modern javascript engine

Our external dependencies include:

- Google Maps, in order to plot units on a map
- Bootstrap, a front-end framework used to simplify user-interface development efforts
- jQuery, a Javascript framework used to simplify front-end development

3 Specific Requirements

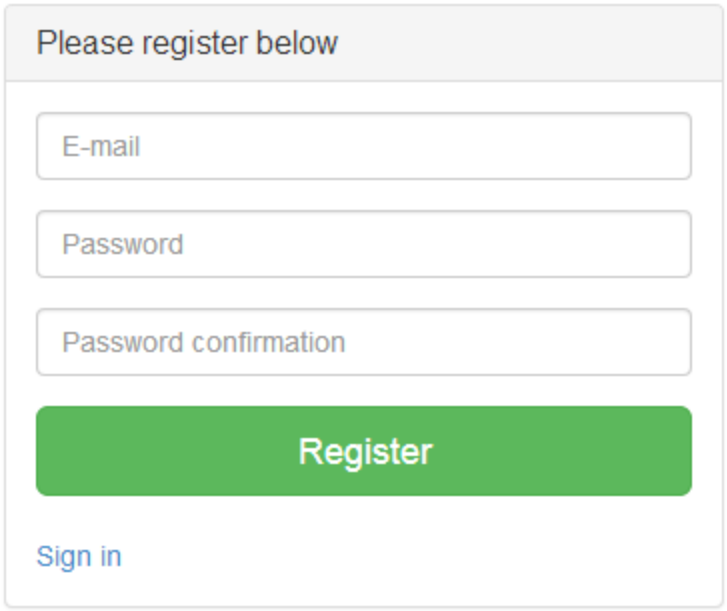
3.1 External Interface Requirements

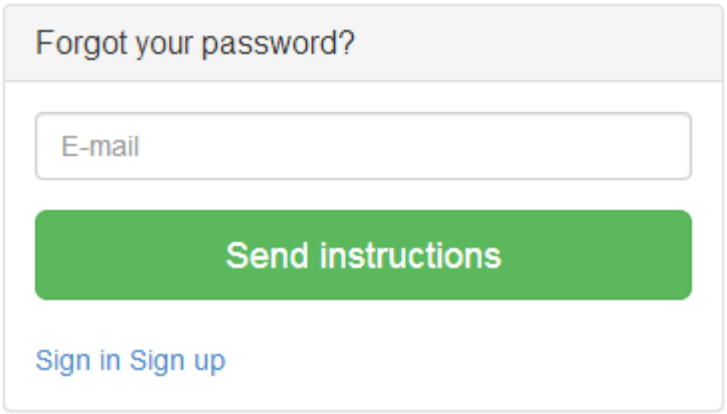
3.1.1 User Interfaces

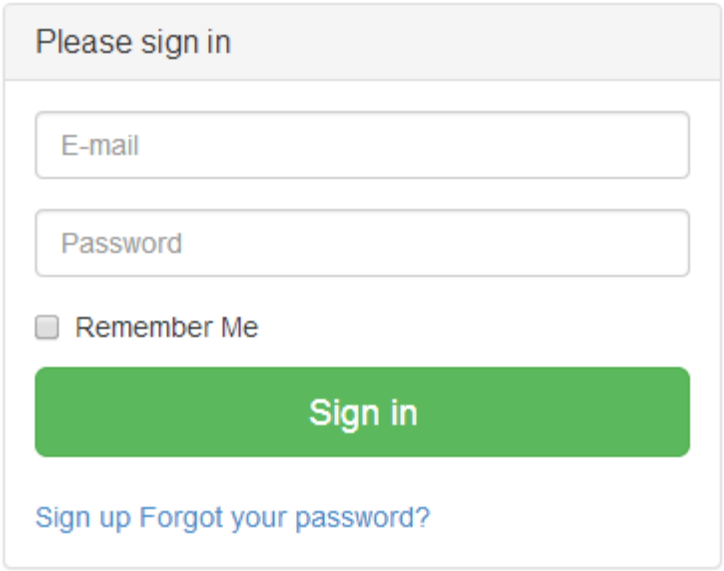
Bootstrap offers a standardized UI framework that maintains a consistent set of user interface tools that Housing4U will utilize. All front-facing interfaces will follow Bootstrap design guidelines.

Every page will include a standard header, including the Housing4U logo and important links. Additionally, each page will contain a general footer that includes other important links.

Some preliminary user authentication dialogs are shown below:

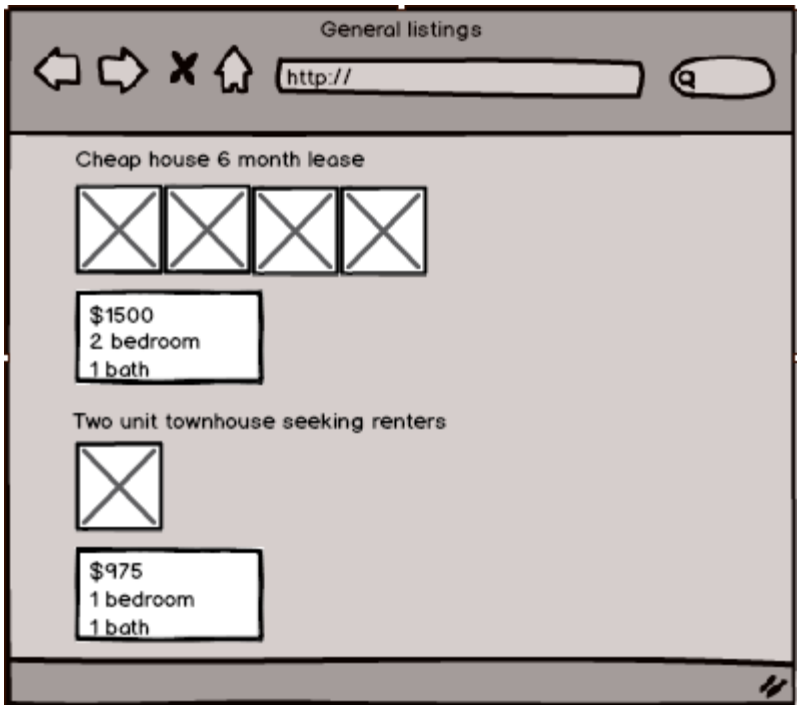


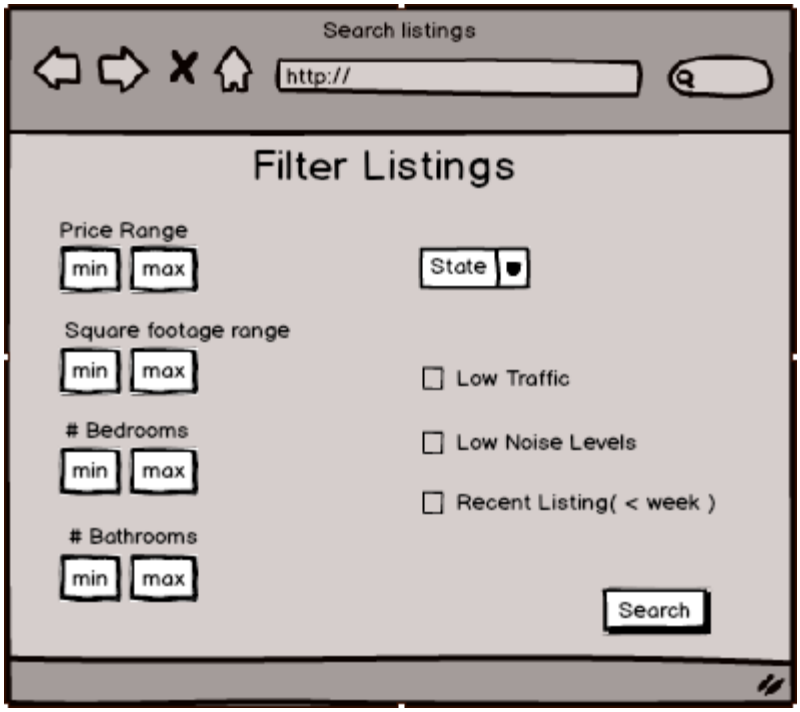





Create a listing
http://
Create New Listing
Rent
Address
Security Deposit
City
Application fee
State
Square footage
Amenities
Bedrooms
Bathrooms
Submit


Modify a listing
http://
Edit Listing
Rent: $950
6777 Del Playa Drive
Security Deposit: $250
Goleta
Application fee: $40
CA
Square footage: 1400
Amenities
Bedrooms: 1.5
Comes with new jacuzzi, floors recently refurbished.
Bathrooms: 1
Delete listing
Submit Changes


General listings
http://
Cheap house 6 month lease
$1500
2 bedroom
1 bath
Two unit townhouse seeking renters
$975
1 bedroom
1 bath


Search listings
http://
Filter Listings
Price Range
min
max
State
Square footage range
min
max
Low Traffic
Bedrooms
min
max
Low Noise Levels
Recent Listing(< week)
Bathrooms
min
max
Search

3.1.2 Hardware Interfaces

Housing4U does not use any hardware interfaces.

3.1.3 Software Interfaces

Our product will be run on a Heroku platform in the cloud, which essentially receives our code from a command line call referencing our GitHub repo, compiles our code and hosts it in an isolated Unix virtual machine. This hooks into our database, and then deploys it on our domain. Afterward, the only messages passed would be GET and POST requests from the user, as well as calls to add, remove, or select data from our database.

3.1.4 Communications Interfaces

Our site will be secured via an HTTPS connection to prevent session hi-jacking. We will use CAPTCHA images to ensure all communication originates from a human, and thus prevent spam data (to a certain extent). Furthermore, sensitive data will be stored in the database using industry standard encryption. Emails sent to restore a password change will contain a randomized temporary password (not in plaintext) so as to prevent identity theft.

3.2 Functional Requirements

Search/Filter: The search and filter features will allow users to view a list of vacant units based on certain criteria they define. For example, a user may want to search on all units that are available within a certain distance of Santa Barbara and under $1000 dollars a month for rent.

Localization: This feature will allow users to search by different locations in the United States. It will instantly find the users' location through Google and show them different listings in their area. Users will also have the option of searching whichever location they choose.

Geographical Distances: The Geographical Distancing feature will show users the distances to certain high priority sites, such as school districts, parks, malls, and restaurants.

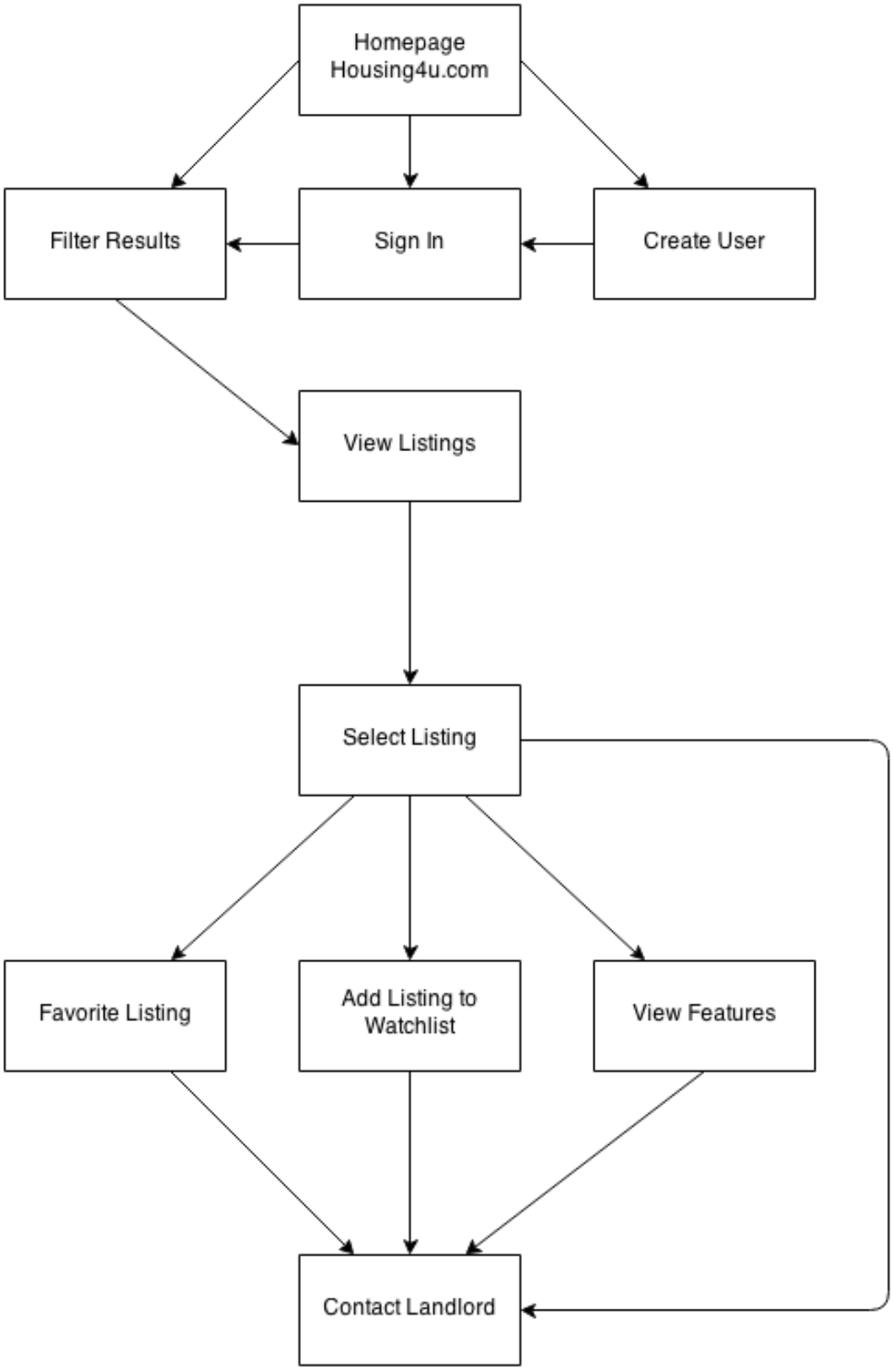
Traffic Proximity: This feature will allow users to see the different traffic patterns around them, which could give them ideas of how long it will take to commute to work or even a friend's house. This feature will be able to show traffic for different times and areas. We can draw this information by using traffic information from Google Maps.

Price Comparison: Comparing prices of different properties is one of the most important features for potential buyers. Our price comparison tool will be useful, allowing for price comparisons of many different houses in many different locations.

Noise Level: The idea behind this feature would be to give a value to the user of the average noise level in the area. For example the property might be located near a freeway or even in a college town that parties a lot, such as Isla Vista.

3.3 Behaviour Requirements

3.3.1 Use Case View



User stories:

- As a user, I can register a new account.
- As a user, I can log in and log out.
- As a user, I can recover my account details.
- As a user, I can view units posted by property owners.
- As a user, I can filter units by geographical distances.
- As a user, I can filter units by noise levels.
- As a user, I can filter units by traffic proximity.
- As a user, I can filter units by price constraints.
- As a user, I can filter units by units that contain images.
- As a user, I can filter units by number of bedrooms.
- As a user, I can filter units by number of bathrooms.
- As a user, I can filter units by square footage.
- As a user, I can filter units by description.
- As a user, I can filter units by location.
- As a user, I can change my password.
- As a user, I can change the color scheme of the website.
- As a user, I can change my e-mail.
- As a user, I can view selected properties on a map.
- As a user, I can make comparisons between different units.
- As a user, I can securely contact individual property owners.
- As a user, I can add properties to a “watch” list.
- As a user, I will be notified if a listing on “watch” list is edited.
- As a user, I will be notified if a listing on “watch” list is deleted.
- As a user, I can choose to be notified by e-mail.
- As a user, I can choose to be notified by text message.
- As a user, I can browse my “watch” list.
- As a property owner, I can add, remove, and update units in the system.
- As a property owner, I can upload multiple images for each unit.
- As a property owner, I can see the amount of interest a unit receives.

4 Other Non-functional Requirements

4.1 Performance Requirements

- All pages of the website should take less than one second to generate
- Minimization of required bandwidth using industry-standard compression

4.2 Safety and Security Requirements

Security requirements in Housing4U include the following:

- Protection of user accounts, including e-mail addresses and other personally-identifiable information
- CAPTCHA-based authentication on registration used to minimize spam and verify human users
- Use of HTTPS in order to prevent user session hi-jacking

It is important to address these security issues in order to prevent a breach of information by any other users of the system.

4.3 Software Quality Attributes

4.3.1 Correctness

Our product will implement a method to verify correctness of rental listings. One way we can do this is with a CAPTCHA image, so as to limit the postings to human users only. This will prevent automated spam bots from posting on our site. Beyond that, me must also verify the validity of listings. We may do this by hooking into the Google Maps API, and cross referencing entered addresses with Google to verify the address is real. Furthermore, we can impose a check to prevent duplicate listings of the same property.

4.3.2 Adaptability

Our product must be adaptable in the sense that it displays properly and functionally to a user no matter what screen size they are using. This is important because today more than 30% of website traffic originates from a mobile device or tablet. We will accomplish this using a responsive CSS design that will downscale our layout according to the screen size of the viewer. This could be a strict downscale in which the content does not change, it simply shrinks, or it could be a full content switch in which larger items are replaced by smaller, phone oriented items (such as shrinking the navigation menu into a dropdown from a button, as opposed to a horizontal list inside a bar).

5 Other Requirements

Appendix A – Data Dictionary

Appendix B - Group Log

CS 189A SCRUM (2-11-14 @ 11:00am):

Completed: Preparation for first sprint presentation in class
Notes: Site team lead will explain what we currently have working and what we plan to have working next sprint. Jason will demo our working mock-up app with its user (database) functionality as well as bootstrap.
Work: Site (team leader) in charge of submitting SRS before deadline

First Sprint Completed:

-Database set up using Heroku
-Integrate Bootstrap into front end
-Get login functionality working
-User sign-up (adds entry into database)
-User Authentication
-Forgot email/password functionality

CS 189A SCRUM (2-10-14 @ 7:00pm):

Completed: Final SRS draft completed and ready for turn intomorrow
Notes: Added final touches and completeness/editing of SRS.
Work: Austin in charge of submitting SRS before deadline in PDF format
-Jason in charge of preparing demo for tomorrow

-Site in charge of preparing presentation for demo tomorrow

CS 189A SCRUM (2-9-14 @ 7:00pm):

Completed: Majority of SRS is done. Need to go over it one more time before submission
Notes: Met up at Site's house to work on SRS.
-We were going to assign each person a section but we thought it would work better if people just grabbed sections they knew how to do.
-If a person did not know something we would talk about it as a group and fill in the question.
-After all of the questions were filled we went over the SRS as a group editing and checking for errors.
-Inserted PowerPoint diagrams into the SRS.

CS 189A SCRUM (2-8-14 @ 7:00pm):

Completed: Tutorial up to Chapter 5 completed!!!(Site, Trae, Austin, Jason, Alvin)
Notes: Tomorrow we will meet up at Site's house at 2pm to work on the SRS together.

CS 189A SCRUM (2-7-14 @ 7:00pm):

Notes: Everyone is finishing up the tutorial up to chapter 5 and we are using all of our sample apps as a reference for our real project.

CS 189A SCRUM (2-6-14 @ 7:00pm):

Completed: Chapter 4 of the tutorial (Austin, Trae, Jason, Site)
Note: Work on chapter 5 of the tutorial setting up git and using test driven development (continued)
-Deciding to stop tutorial after completing chapter 5 because that is all we need to know for now. We then can start working more on our web app.

CS 189A SCRUM (2-5-14 @ 7:00pm):

Notes: Met up with Andrew (mentor) to demo our working mock-up web app with databases and users gems attached.
-Gave Andrew access to our git-hub to look at code and help with bugs/errors.
-Cancan for different roles in the system (for system admin access etc.)
-Next thing to do is xml parsing. Andrews recommends Nokogiri (Jason)
-Suggestion to use simpleform bootstrap
-Suggestion to do attachments in S3 called paperclip
Note: Work on chapter 4 of the tutorial setting up git and using test driven development (continued)

Also note that we decided to meet with Andrew every other week now because one week was too much.

CS 189A SCRUM (2-4-14 @ 7:00pm):

Completed: Chapter 3 of the tutorial (Austin, Trae, Jason)
Note: Work on chapter 4 of the tutorial setting up git and using test driven development (continued)

CS 189A SCRUM (2-3-14 @ 7:00pm):
Completed: Chapter 3 of the tutorial (Austin, Site, Trae, Jason)
Notes: Work on chapter 4 of the tutorial setting up git and using test driven development

---

CS 189A SCRUM (2-2-14 @ 7:00pm):

Completed: Chapter 2 of the tutorial (Austin, Site, Trae, Jason, Alvin)
Completed: Mock-up Housing4U app with databases set up (Site, Jason)
Notes: Working on chapter 3 of the tutorial to set up web server with databases and heroku.

CS 189A SCRUM (2-1-14 @ 7:00pm):

Notes: Working on chapter 2 of the tutorial to set up web server (Continued)
Notes: Site and Jason to complete database and simple web app set up tomorrow, Sunday.
-Site and Jason to spend a few hours at Site's house tomorrowworking on the web app.

CS 189A SCRUM (1-31-14 @ 7:00pm):

Completed: Chapter 1 of the tutorial (Austin, Site, Trae, Jason, Alvin)
Notes: Working on chapter 2 of the tutorial to set up web server

---

CS 189A SCRUM (1-30-14 @ 7:00pm):

Completed: Chapter 1 of the tutorial (Austin, Site)
Notes: Working on chapter 1 of the tutorial to download and set up environment (Continued)
(If person not specified assume entire group working on task)

CS 189A SCRUM (1-29-14 @ 7:00pm):

Notes: Working on chapter 1 of the tutorial to download and set up environment (Continued)

CS 189A SCRUM (1-28-14 @ 7:00pm):

Notes: Working on chapter 1 of the tutorial to download and set up environment

---

CS 189A SCRUM (1-27-14 @ 7:00pm):

Completed:
-We have set up IDE's and a basic web server.
-Github set up

First Sprint:
-Parse the XML file -> shove everything into a database
-Display data on the web page
-Get login functionality working
-User Authentication

Minimum viable product:
1.) Set up the database (with XML data)
2.) Searching for vacant units
3.) User authentication
4.) Add listings/remove listing/edit listing
5.) Contact us
6.) Sleek front end website
7.) Google api integration (pass arguments to the google api)
8.) User reviews
9.) Comparison feature (side by side)

Tasks:
-Learn pivotal tracker (site)
Finish first 3 chapters of rails tutorial (everyone)

CS 189A TA Meeting:

Notes:
-Next week be able to have individual tasks split up between teammates
-Advice going in mentor meetings: Make sure to write down specific questions that you come up with before you meet (design decisions, discussion, etc)

CS 189A SCRUM (1-22-14 @ 7:00pm) *Mentor Meet:

Notes:
-The XML file will tell you everything you want to know about a vacant unit.
We need to parse the xml in ruby. The XML has all the data they have on rentable available units.
-Andrew recommends nokogiri.org for parsing it.
-Test Driven Development: If we are interested in learning the industry way to do things.
-Tutorials for Rails: Michael hartl
One sentence describing our product: A better online marketplace for apartments
-Divise for ruby gems
-Sunspot solr-powered search for ruby objects
-Jetstrap. For if you really don't want to write css, it is a bootstrap generation tool
-We want to use rubymine.

Tasks:
Get pivotal tracker private (site)
Apply to rubymine with school email (everyone).
Finish learning rails (everyone)

Next Meeting: (Wednesday, then Thursdays)
-Next Week (1-29-14): Next week we can do an exercise that would take an hour and go over what the minimal set of features is. We want to do this prior to the SRS and then we can write the SRS

for the minimum amount of features. And then when we get the account for pivotal tracker we can do estimation and prioritize the different tasks. It's just a rough estimate of difficulty.

Week after (2-5-14): We complete the SRS

Week after that (2-12-14): Pivotal Tracker – delegate and prioritize tasks. Estimate time to complete tasks. Eventually calculate velocity.

CS 189A SCRUM (1-21-14 @ 7:00pm):

Notes: Finishing learning ruby on rails. Meeting to start developing a blank rails application.

To do: Create a blank rails application to be accessed on the web that says hello world.

CS 189A SCRUM (1-19-14 @ 7:00pm):

Notes: Continuing to learn Ruby on Rails

CS 189A SCRUM (1-16-14 @ 7:00pm):

Notes: Team is learning programming language Ruby. Each member is spending time looking at tutorials.

To do: Once Ruby is understood, learn the rails framework.

CS 189A SCRUM (1-15-14 @ 7:00pm):
-Meeting with mentor to speak about the vision and design details of the project
-Location: AppFolio facility

Notes:
Define the initial project milestone: specification, design, and prototyping:
Have a blank rails app running somewhere, download Ruby download rails and have it running on somewhere of your choosing.

How do you plan to articulate and design a solution?
We plan to design our solution by holding daily scrum meetings devoted to discussing and analyzing the best way to solve the issue at hand. We will do this by contacting different persons of knowledge given to us by our mentor who can point us to different design solutions to this problem.

Technologies: Ruby on Rails*, Huroku (free), GitHub for source control, PosGRES (locally)
Process model:

Follow huroku tutorial to get your app to get pushed up

Set up a linux VM because running rails on unix is really easy. Virtual image development. Alternatively you could have an instance that you ssh into, or also you could work in the lab.

Rails tutorial starts you out on SQLLite

What features are lacking in craigslist?

…

-You would have a web app with a bunch of vacant units, it would be great to see them on an app.
-You can talk to potential users to see what they would want. When you are looking for a place to live what do you care about? Proximity to campus, price, traffic, distance to restaurants, WALKSCORE*, school districts.
Do we want to build the perfect solution for Isla Vista, or a better craigslist for everyone
-A solution where you could find all places where the traffic midpoint between two places.
-Isla Vista is feasible but maybe not winnable.

They can offer Data such as vacant units, but we could add things like allowing people to search by walkscore. We could build an app where somewhere could search by proximity to parks, schools etc.

Padmapper
Google maps
Walkscore

You can crawl webpages to get information and parse for the info we need.
To Do: Give access to GitHub.
Give access to google doc

Place to learn ruby: try learning a demo app. Ruby.railstutorial.org/chapters
***https://Rentals.appfolio.com/mits.xml gives you the data xml
“xml parser ruby” google search
The use servers and not the cloud for their different data servers.
Mits standard dtd, their data is nationwide

To Do:
-Email mentor with time to meet next week
-First sprint: Create a simple blank ruby on rails app and host it online
-Become familiar with ruby on rails by next friday

CS 189A SCRUM (1-14-14 @ 10:15pm):
-Meeting to set up first Google Hangout and address times to meet with mentor.
-Discuss team vision.

Notes:
-Plan to meet mentor at 7:30pm tomorrow (1-15-14).
-Site designated as team leader, Trae designated as team scribe.

To Do:
-Email Mentor with plans to meet in person asking for a good place to meet (Site).