



Milestone 2

Software Design Document

Team 1

Nora Schinkel, Team Lead | ncschinkel@gmail.com

David Chau, Technical Lead

Paul Derugin

Imran Irfan

Yansen Liu

Steven Nguyen

Submitted to—

Dr. Dragutin Petkovic, Marc Sosnick
Department of Computer Science, San Francisco State University

Document Version: 1.0

Last Modified: 2016 October 26

Contents

1. Use Cases	2
1.1 Renter, Unregistered User	2
1.2 Renter, Registered Student	2
1.3 Owner, Registered Student	2
1.4 Owner, Registered User	2
1.5 Administrator	2
2. Data Definitions	3
3. Functional Requirements	4
Priority 1	4
All Users	4
Registered Users	4
Administrators	4
Priority 2	4
All Users	4
Registered Users	4
Site	4
Priority 3	5
Site	5
4. Non-Functional Requirements	6
5. UI Mockups and Storyboards	7
5.1. Storyboard 1 – Renter, Unregistered User	7
5.2. Storyboard 2 – Renter, Registered Student	9
5.3. Storyboard 3 – Owner, Unregistered User	11
6. High-Level System Architecture	13
Web Stack – LAMP	13
Deployment – Amazon Cloud	13
Dynamic UI – JavaScript/jQuery	13
Database – MySQL	13
UI – Bootstrap	13
MVC Framework – Mini Framework	13
Google API – Google Maps	13

Search Architecture	13
Database Layout.....	14
7. High Level UML Diagrams.....	15
UML class diagram	15
UML Deployment Diagram	15
8. High Level API's.....	16
Users	16
Listings	16
UserImages.....	17
FavoritesListings.....	17
Addresses	17
ListingDetails.....	17
ListingImages	17
9. Key Risks	18
10. Team	19

1. Use Cases

1.1 Renter, Unregistered User

Edward is currently attending SFSU and wants to find an apartment to rent. He is able to browse **apartment listings** on David's List without registering for an account using the search bar at the very top of the page. He has access to the same information for each **apartment listing** as a **registered user** except for the contact information of the **owner**. Edward finds an apartment he is interested in renting. When he tries to contact the owner of the apartment, he is prompted to register, and must do so before being able to proceed.

1.2 Renter, Registered Student

Allen is an incoming freshman to SFSU and needs to find a place to live for his first semester. He has browsed for an apartment that he is interested in renting on our site using a variety of search filters such as price, number of rooms, and distance from campus. In order to rent the apartment, Allen must register for an account on our website. After registering for an account, he is able to communicate with the apartment **owner** to arrange a time to see the apartment as well as to come to an agreement on the terms of the lease. Allen now has a comfortable room for his first year of study at SFSU.

1.3 Owner, Registered Student

Brenda is starting her senior year at SFSU and no longer needs her apartment because she has found other living arrangements. Brenda has registered an account on our website with her SFSU email and has a "**Verified Student**" status on her profile. She is able to list her apartment on the website by clicking on the "Rent out" button to go to the page to create an **apartment listing**, she will be prompted with a form where she has to provide the **address** of the apartment, **number of bedrooms**, **number of bathrooms**, the **price** she is willing to rent for, and any other details that she thinks another fellow student **renter** should know. Once her listing is posted, she will receive email notifications if another student is interested in her apartment, and will be able to communicate with the other student to make any necessary arrangements.

1.4 Owner, Registered User

Candice is an elderly woman who has several properties that she would like to rent to SFSU students. She is able to navigate to the page to post her **apartment listings** using the "Rent out" button. Once she has posted multiple **apartment listings**, she is able to edit or delete her listings by navigating to her personal account page where all of her listings will be.

1.5 Administrator

Francine has been hired to keep David's List clean and friendly for students. She is able to edit and remove any **apartment listing** that she deems inappropriate for the website. She will be able to issue warnings to **registered users** if they violate the website's Terms and Conditions and ban **registered users** for repeat offences. She accomplishes this using the MySQL Workbench on the backend.

2. Data Definitions

1. **Registered User** – User with an account; can be renter or owner.
 - a. **Email** – Email address associated with account.
 - b. **Phone** – Optional phone number associated with account.
 - c. **Bio** – An optional user description.
 - d. **Image** – An optional image.
2. **Guest User** – User without an account; can browse site for apartments; cannot rent or rent out.
3. **Renter** – Registered User seeking apartment.
4. **Owner** – Registered User posting apartment.
5. **Listing** – An apartment/room/share for rent.
 - a. **Number of Bedrooms**
 - b. **Number of Bathrooms**
 - c. **Price**
 - d. **Internet** – Capability, not whether or not it is provided.
 - e. **Pet Policy**
 - f. **Elevator Access**
 - g. **Furnishing**
 - h. **Air Conditioning**
 - i. **Address**
 - j. **Description**
 - k. **Image**
6. **Administrator (Admin)** – User with special privileges; has the ability to remove listings from the site, issue warnings and bans from the site, and generally enforce the Terms and Conditions of the site.

3. Functional Requirements

Priority 1

All Users

- Guest users shall be able to browse through the site, but not have access to post items or contact registered users.
- All users shall be able to create accounts.
- All users shall be presented with the site's Terms and Conditions and Privacy Policy.
- All users shall be able to search for apartments.
- All users shall be able to filter the search based on the following listing types: (a) whole apartment; (b) private room; (c) shared room
- All users shall be able to filter the search based on number of rooms.
- All users shall be able to filter the search based on price.

Registered Users

- Registered users shall be able to login.
- Registered users shall be able to reset their password should they forget it.
- Registered users shall be able to post listings.
- In creating a listing, a registered user shall provide at least one picture of the apartment or room to be rented, the price, and the address.
- Registered users shall be able to edit their listings.
- Registered users shall be able to remove their listings.

Administrators

- Admins shall be able to delete accounts.
- Admins shall be able to delete postings.

Priority 2

All Users

- All users shall be able to filter the search based on distance from the SFSU campus.

Registered Users

- Registered users shall be able to obtain Verified Student status based on their SFSU email.
- Registered users shall be able to create a biography for their profile.
- Registered users shall be able to save listings to their Favorites.
- Registered users shall be able to find roommates.

Site

- The site shall show recently created listings.

- The site shall use Google Maps integration to display the approximate location of the apartment.
- The site shall show a rating of a registered user, based on votes from other users.

Priority 3

Site

- The site shall provide directions between SFSU and the apartment and give the time it takes to travel from the apartment to SFSU.
- The site shall display nearby restaurants on Google Maps.

4. Non-Functional Requirements

1. Application shall be developed using class provided LAMP stack.
2. Application shall be developed using pre-approved set of SW development and collaborative tools provided in the class. Any other tools or frameworks shall be explicitly approved by Marc Sosnick on a case by case basis.
3. Application shall be hosted and deployed on Amazon Web Services as specified in the class.
4. Application shall be optimized for standard desktop/laptop browsers, and shall render correctly on the two latest versions of all major browsers: Mozilla, Safari, Chrome. It shall degrade nicely for different sized windows using class approved programming technology and frameworks so it can be adequately rendered on mobile devices.
5. The site shall have a response time of less than 3 seconds.
6. Data shall be stored in the MySQL database on the class server in the team's account.
7. Application shall be served from the team's account.
8. The site shall accommodate up to 50 concurrent users accessing the application at any time.
9. The site shall be governed by a set of Terms and Conditions, which shall include, but not be limited to (1) a statement limiting David's List's liability; (2) a notice of copyright; (3) a Privacy Policy.
10. Privacy of users shall be protected and a Privacy Policy shall be effectively communicated to the users. The Privacy Policy shall disclose what user data is collected and how it is used.
11. The language used shall be English.
12. Application shall be very easy to use and intuitive. No prior training shall be required to use the website.
13. Google analytics shall be added for major site functions.
14. Messaging between users shall be done only by class approved methods to avoid issues of security with e-mail services.
15. Pay functionality (how to pay for goods and services) shall be simulated with proper UI, no backend.
16. Site security: basic best practices shall be applied (as covered in the class). In addition, the site shall salt and hash user passwords and prevent basic SQL injection.
17. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development, and only the tools and practices approved by instructors.
18. The website shall prominently display the following text on all pages "SFSU/FAU/Fulda Software Engineering Project, Fall 2016. For Demonstration Only". (Important so as to not confuse this with a real application).

5. UI Mockups and Storyboards

5.1. Storyboard 1 – Renter, Unregistered User

Edward enters his search keywords in the search bar and presses 'Browse'. On the browse page he can refine his search using the filters. Once he finds the right apartment he can open the details in a new page.

HOME PAGE

The Home Page mockup includes a header with a 'login/register' link, a search bar with a dropdown arrow, and a 'BROWSE' button. Below the header is a navigation bar with 'HOME | RENT OUT | ABOUT'. The main content area features a 'WELCOME' section with a link to 'ABOUT THE WEBSITE'. Below this is a 'FEATURED LISTINGS' section containing two identical listing cards. Each card displays a house icon, a 'LISTING NAME' field, an 'INFO' link, a 'PRICE' field, and a 'RENT' button. At the bottom of the page is a footer with 'PRIVACY | CONTACT'.

BROWSE

The Browse page mockup features a header with a search bar and a 'BROWSE' button, and a navigation bar with 'HOME | RENT OUT | ABOUT'. The main content area is divided into three columns. The left column is a 'FILTER' sidebar with a 'FILTERS' section. The middle column, titled 'FOUND # LISTINGS', displays a list of four apartment listings. Each listing card includes a house icon, a 'LISTING NAME' field, an 'INFO' link, a 'PRICE' field, and a 'RENT' button. The first listing also has a 'more info' link. The right column is a 'MAP' section. At the bottom of the listings is a pagination bar with '< # / TOTAL >'. The 'RENT' button on the first listing is highlighted with a red border.

When Edward presses the 'Rent' button, he is prompted to login or register. After registering he can contact the owner.

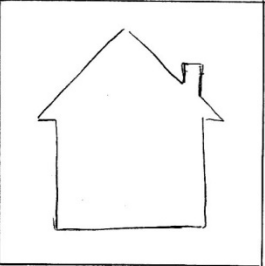
LISTING DETAILS

[login](#) | [register](#)

LOGO

HOME | RENT OUT | ABOUT

DETAILS OF LISTING NAME



MAP

OWNER
INFO

LISTING NAME PRICE
INFO

★

FOOTER

REGISTER

[login](#) | [register](#)

LOGO

HOME | RENT OUT | ABOUT

REGISTER

ACCOUNT
BIO
PICTURE

NAME
EMAIL

☐ I AGREE ...

FOOTER

5.2. Storyboard 2 – Renter, Registered Student

Allen puts his keyword in the search bar and presses 'Browse'. He can find the apartment he needs by using filter. Then he registers for his personal account.

HOME PAGE

login/register

LOGO ▾ BROWSE

HOME | RENT OUT | ABOUT

WELCOME

ABOUT THE WEBSITE

FEATURED LISTINGS

	LISTING NAME	PRICE
	INFO	RENT

	LISTING NAME	PRICE
	INFO	RENT

PRIVACY | CONTACT

BROWSE

LOGO ▾ BROWSE

HOME | RENT OUT | ABOUT

FILTER

FOUND # LISTINGS

	LISTING NAME	PRICE
	INFO	RENT

		PRICE
		RENT

--	--	--

--	--	--


< # / TOTAL >

MAP

After the registration, he can contact the owners.

REGISTER

LOGO



BROWSE

HOME | RENT OUT | ABOUT

REGISTER

ACCOUNT

BIO

PICTURE

NAME

EMAIL

☐ I AGREE ...

CANCEL

NEXT

FOOTER

5.3. Storyboard 3 – Owner, Unregistered User

Candice goes to the homepage and presses “Rent Out.” She is prompted to login or register. She chooses to register and is redirected to the registration page.

HOME PAGE

login | register

LOGO [dropdown] [input] BROWSE

HOME | RENT OUT | ABOUT

WELCOME

ABOUT THE WEBSITE

FEATURED LISTINGS

	LISTING NAME	PRICE
	INFO	RENT

PRIVACY | CONTACT

REGISTER

login | register

LOGO [dropdown] [input] BROWSE

HOME | RENT OUT | ABOUT

REGISTER

ACCOUNT
BIO
PICTURE

NAME
EMAIL

☐ I AGREE...

CANCEL NEXT

FOOTER

Ater registering she is directed to the Rent Out page. Here she fills in all the details about the room. When she is finished, she can review all her posted listings on her profile page.

CREATE LISTING

username | profile

LOGO BROWSE

HOME | RENT OUT | ABOUT

RENT OUT

LISTING NAME* ?

PRICE*

...

PICTURES

DETAILS

☐ I AGREE...

CANCEL POST

FOOTER

MY LISTINGS

username | profile

LOGO BROWSE

HOME | RENT OUT | ABOUT


USERNAME > MY LISTINGS

OVERVIEW


YOU HAVE # LISTINGS

FAVORITES

MY LISTINGS

 LISTING NAME EDIT

INFO



FOOTER

6. High-Level System Architecture

Web Stack – LAMP

- Linux Operating System
- Apache Web Server
- MySQL Database
- PHP programming/scripting language

Deployment – Amazon Cloud

- Great reliable service

Dynamic UI – JavaScript/jQuery

- Interactive front end
- Dynamic coding scripts

Database – MySQL

- Industry standard for database
- Used in L.A.M.P software bundle

UI – Bootstrap

- Supports popular browsers like Chrome, Firefox, and Internet Explorer
- Built-in scaling
- Minimum HTML needed
- Contains HTML, CSS, and JavaScript based templates

MVC Framework – Mini Framework

- Minimalistic MVC framework in PHP
- Low learning curve

Google API – Google Maps

- Allows users to determine how far the apartment is from the school
- Allows users to find directions from school to apartment

Search Architecture

- Percentagewise comparison, simple and quick
- Bigram comparison, priority 2
- Allows users to search for zip codes, addresses and cities.

Database Layout

USER
UserID
EmailAddress
StudentID
Password
Phone
Bio
Verified

LISTING
ListingID
UserID
Address
Price
Type
Status

LISTING DETAILS
ListingID
NumberOfBedrooms
NumberOfBathrooms
Internet
PetPolicy
ElevatorAccess
Furnishing
AirConditioning
Description

FAVORITES
UserID
ListingID

USER IMAGES
UserID
Image
ImageThumbnail

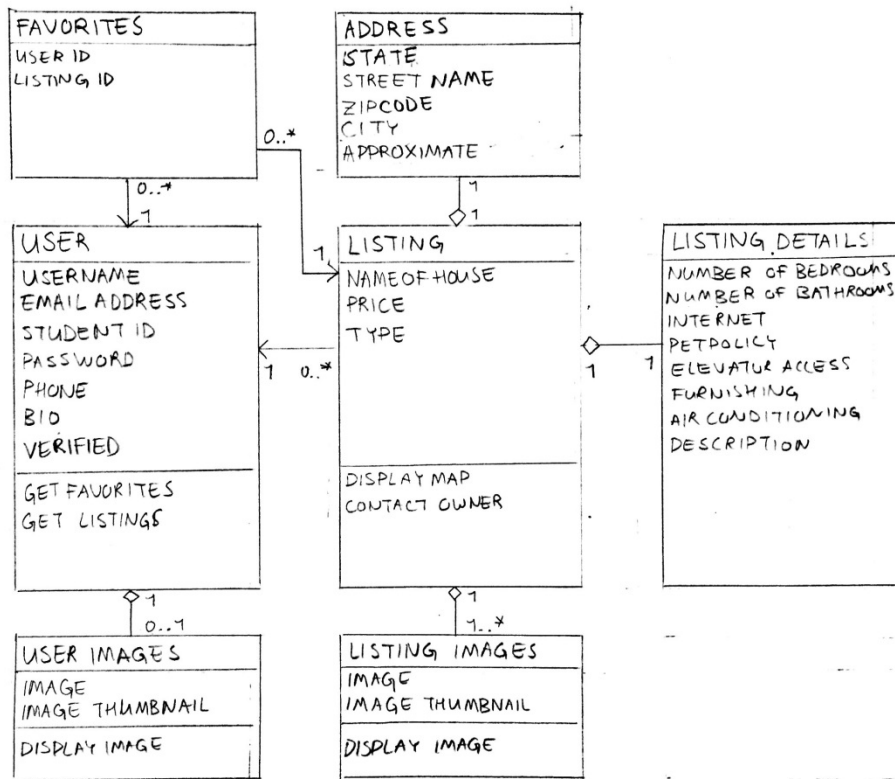
LISTING IMAGES
ListingID
Image
ImageThumbnail

ADDRESS
ListingID
StreetName
City
Zipcode
State
Approximate

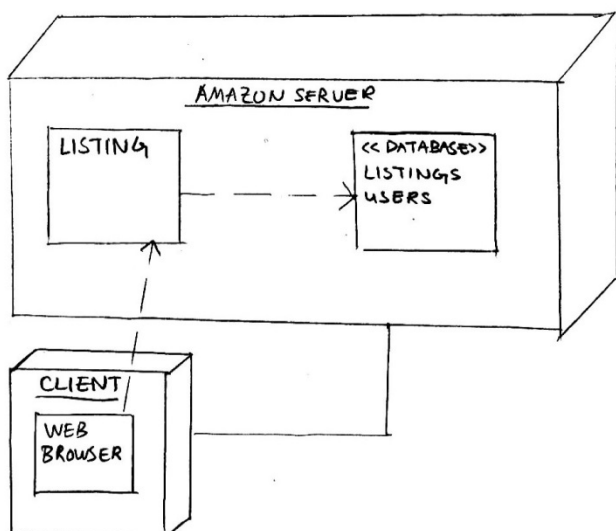
The images and image thumbnails will be saved in the database as BLOBs. The BLOB sizes are 65535 bytes for the thumbnail and 4294967295 bytes for the full image.

7. High Level UML Diagrams

UML class diagram



UML Deployment Diagram



8. High Level API's

Users

Users Controller	
/users/getuser/{userid}	This function returns a user's profile page. Parameter is the integer id of the user. Returns an HTML page.
/users/deleteuser/{userid}	Function to delete a user. Parameter is the integer id of the user
/users/edituser/{userid}	Function to edit a user's profile. Parameter is the integer id of user. External information is JSON encoded data which contains what part of the profile to change.
/users/newuser/	Function to create a new user. External information is JSON encoded data which contains new user information. Returns HTML to user's page
/users/login/	Function to login as a user. External information is JSON encoded data which contains login data, such as email/username and password

Listings

Listings Controller	
/listings/getlisting/{listingid}	Function to return a listing's page. Parameter is the id of the listing. Returns HTML
/listings/deletelisting/{listingid}	Function to delete a listing's page. Parameter is the listing id.
/listings/editlisting/{listingid}	Function to edit a listing. Parameter is the listing id. External information is JSON encoded data which contains part of listing data to change
/listings/newlisting	Function to create a new listing. External information is JSON encoded data which contains new listing data

UserImages

/userimages/getuserimages/{userid}	Get all user associated images, given user id
/userimages/deleteuserimages/{userid}	Delete user associated images, given user id
/userimages/uploadimage/{userid}	Uploads a new user image to user, given id

FavoritesListings

/favoritelistings/addfavorites/{listingid}	Adds listing to list of favorites
/favoritelistings/deletefavorites/{listingid}	Deletes listing from list of favorites

Addresses

/addresses/createaddress/{listingid}	Creates new addresses related to listing id
/addresses/searchbyaddress/	Searches addresses based on JSON data
/addresses/deleteaddress/{listingid}	Deletes addresses related to listing id
/addresses/updateaddress/{listingid}	Updates address related to listing id

ListingDetails

/listingdetails/getdetails/{listingid}	Retrieves listing details based on listing id
/listingdetails/deletedetails/{listingid}	Deletes listing details based on listing id
/listingdetails/editedetails/{listingid}	Updates listing details based on listing id
/listingdetails/createdetails/{listingid}	Creates new listing details associated with listing id

ListingImages

/listingimages/getimages/{listingid}	Gets images based on listing id
/listingimages/deleteimages/{listingid}	Deletes images based on listing id
/listingimages/uploadimages/{listingid}	Uploads new listing images to listing id

9. Key Risks

Risk Type	Risk	Solution
Skills risk	No previous knowledge of BLOBs can cause delay in implementation.	Drop BLOBs if there is too much time spent on implementing them.
Schedule risk	Team members are currently engaged in other projects as well. The time they can spend on this project may be limited.	Provide a planning each week, so each team member can divide their time accordingly.

10. Team

Team Member	Role
Nora Schinkel	Team Lead
David Chau	Technical Lead
Paul Derugin	Back-end Developer
Steven Nguyen	Front-end Developer
Yansen Liu	Front-end Developer/UI/UX designer
Imran Irfan	Back-end Developer