SW Engineering CSC648/848 Spring 2019

SFSURent

Team 11

Cory Lewis(Team Lead, clewis9@mail.sfsu.edu)
Xinyu Zou(GitHub Master)
Soheil Ansari(Back End Lead)
Junwei Liang
Poorva Rathi
David Dropping(Front End Lead)
Chintan Sanjay Puri

Milestone 4

History Table:
4/30/19
5/19/19

1) Product Summary

SFSURent

SFSURent prioritizes a top-notch user experience geared for students attending San Francisco State University. In addition to the critical features provided by the other competitive systems such as filtering and searching, SFSURent will provide additional unique features for SFSU students. There are many websites that offer housing options, but they are too expensive, far from school, and don't cater to San Francisco State students specifically. Craigslist, Roomster, and Zillow are all common sources for housing but offer minimal opportunity for finding the optimal place for a typical SFSU student. Our platform does not charge to contact landlords and we have an administration system in place that removes postings that might contain inappropriate content or does not meet our website guidelines.

Home Page

- Listings are organized well on a home page. User can see basic listing details such as, price, distance, number of bedrooms, and number of bathrooms on the details page for the listing. If a user is interested in a listing, they can contact the landlord from the home page or the details page.
- Search is available on our home page. User can search listings by listing types.
- After search results display on home page, user can also filter the search results by price, distance to SFSU, number of bedrooms, and number of bathroom.
- Create posting and view dashboard links are available for registered users.
- Login/Sign up link is also available on our home page.

Listing Detail Page

- All the listing details will be displayed on this page, including pictures of the listing, map of the listing, all the features, price, distance to SFSU, number of bedrooms, and number of bathrooms.

Create Post Page

- Registered users can simply type in information about their listings in a post listing form.

Dashboard Page

- Registered users can see their own postings and view message from potential tenants on dashboard page.
- If a posting is expired or unavailable, registered users are able to delete the posting.
- Admin can see all the postings. After registered users submit the post listing form, admin will approve or disapprove the postings before they go live.
- Admin can also delete any postings and block accounts.

Login/Sign Up Page

- If registered users forget their password, they are able use forgot password to recover login.

Our product mainly focuses on helping SFSU students to find the perfect living circumstances. We help incoming SFSU students find housing easily by designing our filtering to reference SFSU in terms of travel distance, and price. Landlords looking to rent their property quickly can find a market with our site because each semester SFSU has many incoming students eagerly searching for housing options. Additionally, landlords can list their property with ease due to our Google Maps API integration for address auto fill. Our website makes every effort to be secure and keep our users passwords encrypted so even our administrators cannot view this data.

SFSU Rent Site: http://ec2-18-144-46-90.us-west-1.compute.amazonaws.com/listing/

2) Usability Test Plan

- Test objectives

The feature being tested in this usability test is the “Search” feature. Search is an integral component of the website as the intended users i.e. SFSU students need this feature in order to efficiently find the listing that they are looking for.

- Test description
    - System setup: Single room setup with observer/monitor close to the evaluator.
    - Starting Point: User clicks supplied URL and is started at the Home Page where available listings can be viewed.
    - Intended users: Since the website is designed to contain housing listings around the SFSU campus, the intended users for this test are **SFSU students with <u>average</u> computing skills**.
    - URL: http://ec2-18-144-46-90.us-west-1.compute.amazonaws.com/listing/
    - What is to be measured:
      For this test, user satisfaction evaluation is the primary metric that is going to be measured.
      users will be asked to perform the task and then a Likert questionnaire will be used where user answers/rates how they “feel” about usability.

- Usability Task description:
  Open the website
  Search for a listing with one room.
  Search for a listing that is a house
  Search for all apartments
  Search for all rooms
  Search for all bungalows
  Search for a quiet apartment.

Questionnaire :

		Strongly Disagree	Disagree	Neither agree or disagree	Agree	Strongly Agree
	Questions	**1**	**2**	**3**	**4**	**5**
1	I found searching for listings easy to use.					
2	I found searching for a particular listing (apartment,bungalow,room) simple.					
3	I found the website easy to navigate.					

Comments :	

3) QA test plan (2.5 pages MAX)

The Item being tested is the search function. The test objective is to test the accuracy of the %like search. The device the test will be performed on is a Dell XPS laptop. The browsers that we will test our application on are Google Chrome and Mozilla Firefox. The expected time of this test will be less than 10 minutes.

- **Input**: enter “SFSU”
- **Output**: Check you get 1 result, containing the string “one” in the name field

- **Input**: enter filter for $1000-$1500
- **Output**: Check you get 2 results, both within the price range of $1000-$1500

- **Input**: enter “student”
- **Output**: Check you get 7 results, all with the string “title” in the name field

Test Number	Title	Description	Input in search	Expected Output	Result Pass/Fail
1	Search field test	Test % like in search for name field by entering a single word and comparing the number of responses to the number of expected responses from the database	“SFSU”	Get 5 results, containing the word “sfsu” in the title field	Chrome: Pass Firefox: Pass
2	Search field test	Test % like in search for name field by entering a single word and comparing the number of responses to the number of expected responses from	“1 bedroom apartment”	Get 5 result, containing the word “1 bedroom apartment” in the title field	Chrome: Pass Firefox: Pass

		the database			
3	Search field test	Test % like in search for name field by entering a single word and comparing the number of responses to the number of expected responses from the database	“student”	Get 2 results, all containing the word “student” in the title field	Chrome: Pass Firefox: Pass

4) Code Review

- Coding Style
    - GitHub Commits: (Add|Remove|Modify|Fix) what you did
    - Header Comments:
      /*
      Author:
      Date:
      Description:
      */
    - Block Comments: // notes, Name, Date Modified
    - Lint used for code formatting

- Code review is done by the two team members where the emails are exchanged for reviewing the coding style on a block of code (input validation). Below are the screenshots of the emails and the code.

    - Email was sent by one member to the team lead to verify the coding style.

From: Soheil Ansari
Sent: Friday, May 1, 2019 12:28 PM
To: Cory Jameson Lewis <clewis9@mail.sfsu.edu>
Subject: Code Review

Dear Cory,

Please review the following commit regarding user input validation for coding style:

https://github.com/CSC-648-SFSU/csc648-sp19-team11/blob/18b8d09a1f7edf26fdb473340b82a5feb7be7d62/src/validators/user.js

Looking forward to hear your feedback.

Best Regards,

Soheil **A**nsari
ID 917951156
M (202) 644-1018
IN linkedin.com/in/soheilansari

    - Code under review:

```
check('name', 'Name must be only characters and spaces.').exists().matches(/^[a-z ]+$/i),
check('email', 'Invalid email').exists().isEmail(),
check('password', 'Password must be between 6 and 18 characters.').exists().isLength({ min: 6, max: 18 }),
```

Email replied by the team lead to encourage the hard work and improve the coding style.

Sent: Friday, May 3, 2019 3:34 PM
To: Soheil Ansari <sansari1@mail.sfsu.edu>
Subject: RE: Code Review

Hello Soheil,

Thank you for your contribution to the branch. Your code works well.

Some small formatting changes:

- Please add length for name field
- Please change the message so it is more clear to the user what they are mistyping
- Please break the long lines so they are easier to read

Best,

Cory Lewis

- Code after the review:

```
14  +   check('name',
15  +     'Name must be only characters and spaces and be between 2 and 15 characters long.')
16  +     .exists().matches(/^[a-z ]+$/i).isLength({ min: 2, max: 15 }),
17  +   check('email',
18  +     'Invalid email address.')
19  +     .exists().isEmail(),
20  +   check('password',
21  +     'Password must be between 6 and 18 characters.')
22  +     .exists().isLength({ min: 6, max: 18 }),
```

5) Self-check on best practices for security

Security and privacy are among the hot topics in the media these days. There have been several scandals this year alone which revealed some major companies were not following basic privacy and security measures which could cause a data breaches and affect the privacy of millions of users. Therefore, we have taken the security and privacy of users into account from the first day of designing our application and implemented measure to cover the most known attacks.

The following lists the major asset that we are protecting in our application:

- User's login information namely their email and password.
- User's personal information such as location and phone number.
- User's listing ownership (not revealing information on landlord on listing page).
- Messages between renters and landlords.
- Search limited to 40 alphanumeric characters

The first measure that we took is to keep the user password encrypted in the database to the website administrators and engineers would not have access to plain text passwords. Moreover, that would avoid the passwords to be revealed in case that the database is compromised. For this purpose, we took advantage of bcrypt Node.js package which is the most used library for password encryption. The following figure shows the sample encrypted password saved in our database:

id	name	email	password
1	admin	admin.1@gmail.com	$2b$10$.7z4MKeIhF5uL69UYXJ8beZlhUJqsWcoHvnHyr/bObudxDBgSor5i
2	user1	user.1@gmail.com	$2b$10$I/11k8nYq7zDJxD1g3mbme20uNM0A/FXoVj/O3oXvXKXG7kS7EltC
3	user2	user.2@gmail.com	$2b$10$KxR9tI/uj6gACmrLRXRJl.aNZIC9RMijBc5f0dptbJDIyIsDMrjfO
4	user3	user.3@gmail.com	$2b$10$na4FGv27DHBe8SLPZB26gu7hAOF2vg7Oij7UQMqoS.COHjvnCiCBm
5	user4	user.4@gmail.com	$2b$10$dXLX8ZFLz56XBBR2Q7qJJ.z7viFIPmSKKZXKTfKTnl26K13IsJxlO
6	user5	user.5@gmail.com	$2b$10$KyXp3TytQ.IJSKiOi.10GOJ7J5eVmqkjOK2IAwxlbkdjfnWglZ8Su

Another important consideration is input validation to prevent a variety of attacks such as SQL injection and Cross-site scripting (XSS). Moreover, it will normalize the content of the website and make it more user-friendly and predictable. To do this, we have taken advantage of the Express Validator library which is the industry standard for input validation in Node JS. More importantly, this library uses Validator.js which provides standard input validation methods. For example, it has methods to detect if an email address is in a valid format. To organize the project, we have declared all the validation rules in separate packages in respect to their class. The following code snippet shows our validation rules for creating a new user:

```
module.exports.validateCreateUser = () => [
  check('name',
    'Name must be only characters and spaces and be between 2 and 15 characters long.')
    .exists().matches(/^[a-z ]+$/i).isLength({ min: 2, max: 15 }),
  check('email',
    'Invalid email address.')
    .exists().isEmail(),
  check('password',
    'Password must be between 6 and 18 characters.')
    .exists().isLength({ min: 6, max: 18 }),
  check('password_confirm').custom((value, { req }) => {
    if (value !== req.body.password_confirm) {
      throw new Error('Password confirmation does not match password');
    }
  }),
];
```

Finally, to protect our web server and database. We have used AWS firewall feature which only allows specific IP address to connect to specific ports. Currently, the only port that is open to public traffic is port 80 which is used for http requests. Our MySQL database is listening on port 3306 which is protected by the firewall to only accept connections from within the server. Images on the server are stored as type BLOB so we can keep all assets locked under the same ruleset as all the other fields in the database to prevent any data breach.

5) Self-check: Adherence to original Non-functional specs

1.1. Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0 (some may be provided in the class, some may be chosen by the student team, but all tools and servers must be approved by class CTO). - DONE

1.2. Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers - DONE

1.3. Selected application functions must render well on mobile devices - ON TRACK

1.4. Data shall be stored in the team's chosen database technology on the team's deployment server. - DONE

1.5. No more than 50 concurrent users shall be accessing the application at any time - DONE

1.6. Privacy of users shall be protected, and all privacy policies will be appropriately communicated to the users. - DONE

1.7. The language used shall be English. - DONE

1.8. Application shall be very easy to use and intuitive. - ON TRACK

1.9. Google analytics shall be added - DONE

1.10. No email clients shall be allowed - DONE

1.11. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated. - DONE

1.12. Site security: basic best practices shall be applied (as covered in the class) - DONE

1.13. Before posted live, all content (e.g. apartment listings and images) must be approved by site administrator - Done

1.14. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development - DONE

1.15. The website shall prominently display the following exact text on all pages *"SFSU Software Engineering Project CSC 648-848, Spring 2019. For Demonstration Only"* at the top of the WWW page. (Important to not confuse this with a real application). - DONE