San Francisco, CA 94123
(415) 724-9116

Milestone 4

MEET OUR AWESOME TEAM 5 MEMBERS

Jeffrey Fulmer Gardner
Team Lead / Github Master/ jfulmergardner@mail.sfsu.edu

Adele Wu
Front-end Lead

Eddy Yun
Back-end Lead

Jeffrey Friedrich
Database Lead

Jose Andres Quinteros
Team member of Front-end/Document Master

Kris Byington
Team member of Database Team

CSC 648/848 Software Engineering - Fall 2021

November 2th 2021

Use Cases, High-level Requirements and Architecture

Henry Villar, CEO & CTO, SFSU

jvillar@mail.sfsu.edu

Acknowledge

Content based on class material created by Dr. D. Petkovic & Anthony Souza

Team05 - www.RoomM8.net

HISTORY TABLE

MILESTONES	REVISION DATE	NOTES / FEEDBACK
Milestone 1 draft	September 21, 2021	No revision. High Level Requirements & Specifications, Use Cases, and Architecture.
Milestone 2 draft	October 1, 2021	More Detailed Requirements, Specs, Architecture, UI, Mock-Ups and Vertical SW prototype.
Milestone 3 draft	October 26, 2021	Review of Functionality, UI, SW and planning for final product delivery.
Milestone 4 draft	November 9, 2021	Beta Launch, QA and Usability Testing and Final Commitment for Product Features (P1 list). Oral Presentation (Seniors)
Milestone 5 draft	December 7, 2021	Final Demo and Documents for Grading

TABLE OF CONTENTS

1) Product summary (e.g. how would you market and sell your product – about 1⁄2 page)

Name of the product: *Room M8*
Room M8 is an application that helps people find roommates by offering a place for people without rooms to connect with people with rooms.

Functions:

Guest Users

- Guest users are able to access the landing page.
- Guest users are able to search for rooms.
- Guest users are able to search for roommates.
- Guest users are able to register and create accounts.
- Guest users shall not be able to utilize premium features (posting + commenting)

Registered Users

- Registered users are able to access the landing page.
- Registered users are able to access the about page.
- Registered users are able to access the login page.
- Registered users are able to access the landing page.
- Registered users are able to access the about page.
- Registered users are able to access the login page.
- Registered users are able to search for rooms.
- Registered users are able to search for roommates.
- Registered users are able to post a vacant room.
- Registered users are able to search around their preferred location.
- Registered users are able to set their room preferences such as: majors, school, pets, smoking, language, similar interests, and/or hobbies.
- Registered users are able to filter roommates by majors, school, pets, smoking, language, similar interests, major, and/or hobbies.

Admins

- Admins are able to delete or ban any registered users.
- Admins are able to give and/or remove permissions and other users that are not admins.

Our unique feature includes our expansive filter options and searches are divided between users and rooms.

URL: roomm8.net or http://3.13.140.74/

2) Usability test plan

Test Objectives

- Measure overall friction of the roomm8.net
- Find heavy friction points in site flow

Tests

- Click map each task associated with search
- User task evaluation with likert scale questionnaire

Click Map

Users

Usability testing programmer

Methodology

The usability testing programmer will perform a search for a house on silver street and record each click per page change. The test environment will be either of the browsers fully supported by roomm8.net. There will be no test monitor provided. The data will be stored in a table and appended to the end of this description.

Task	Clicks to completion	Browser
Find a room with a private room	3	Firefox
Find a male roommate	3	Firefox

User Task evaluation

Methodology

Users will be seated alone in a cubicle. Users will be given a personal computer or a mobile device to reach the website. It will have either Firefox or Chrome installed. They will be instructed to go to roomm8.net and find a place that either they or someone they know would like to live in. Each user will be accompanied by a test monitor. The test monitor will give all instructions to the Users. The test Monitor will only give instructions at the start of the test. Once the search test is completed the test monitor will instruct the User to fill out the likert test.

Users

This should be focus group large random type test

Tasks:

- Go to roomm8.net and find a room that looks like one you would like to live in
    - The User will use either Firefox or Chrome
    - They will be given a url www.roomm8.net
    - They will be given free reign on the site
    - Encouraged to ask questions or make comments on their experience
- Fill out likert scale questionnaire

Test Monitor

- Will not guide the User in any way after initial instructions
- Will not give a directive answer to the user
- Will log the start and end times of the test
- Will take notes of every question or comment the User has
- Will identify when a Users question is answered by the site design

Evaluation

Answers to the Likert Scale Questions will be the primary sources for evaluation. In addition, test users will be timed but not informed of it in order to see if extra time on the site correlates with positive or negative Likert Scale answers.

Likert Test Questions Format

A question about the user experience when using the product

- ☐ Strongly Agree
- ☐ Somewhat Agree
- ☐ Disagree
- ☐ Somewhat disagree
- ☐ Strongly disagree

1. It is clear what the product is
2. The website was easy to scan/skim for information
3. You feel confident that you will find what you are looking for within the website
4. Search is easy to use
5. You understand the types of information collected and used by the website
6. It is likely you will use or recommend this website

3) QA test plan - max 2 pages

Test Objectives:

- To evaluate the search post functionality of the site and determine what information should be added.

HW and SW setup (including URL):

Windows10 PC - Firefox,Chrome URL: http://3.13.140.74/browse-room

Feature to be tested:

- Search post
- Filter post
- Click Map

QA Test plan:

FIREFOX

Test #	Test Title	Test Description	Test Input	Expected Correct Output	Test Results (PASS/ FAIL)
1	Area	Type in an area; return info relevant to that area	Enter '2701 Green St' into search bar; Press magnifying glass.	Map focuses on 2701 Green St. San Francisco; Show apartments from there	PASS

2	Check min/max	Type in a max or min; check that apartments are filtered	Enter max range: ‘2000’; Press ‘Filter’	All visible apartments cost no more than 2000. No visible apartments are for more than 2000.	PASS
3	Check private/shared room	Select private/shared room; check that the results are limited appropriately	Select ‘Private Room’; Press ‘Filter’	All visible apartments are for a private room. No visible apartments are for shared room.	FAIL/ Missing info, fix incoming

GOOGLE CHROME

Test #	Test Title	Test Description	Test Input	Expected Correct Output	Test Results (PASS/FAIL)
1	Area	Type in an area; return info relevant to that area	Enter ‘2701 Green St’ into search bar; Press magnifying glass.	Map focuses on 2701 Green St. San Francisco; Show apartments from there	PASS
2	Check min/ max	Type in a max or min; check that apartments are filtered	Enter max range: ‘2000’; Press ‘Filter’	All visible apartments cost no more than 2000. No visible apartments are for more than 2000.	PASS
3	Check private /shared room	Select private/shared room; check that the results are limited appropriately	Select ‘Private Room’; Press ‘Filter’	All visible apartments are for a private room. No visible apartments are for shared room.	FAIL/ Missing info, fix incoming

4) Code Review:

a) By this time you should have chosen a coding style. In the report say what coding style you chose.

browse-room.hbs

```
<div class="search ">
     <input type="text" id="search-text" placeholder="search"
class="w-11/12"></input>
     <button id="search-button">
         <img src="/images/search_icon.png" width="20px">
     </button>
</div>
```

This code is nice and concise. There is not as much room to be creative with html tags expect for the indentation and I like the way that this one is laid out and each new tier of information has its own layer.

search-post.js

```
const searchButton = document.getElementById("search-button");
/**
 * GoogleMap Object to generate all the markers
 */
const gmap = new GoogleMap("map");

if (searchButton) {
  searchButton.onclick = executeSearch;
}
/**
 * Generates all the necessary innerHTML from the results of a query
 * and googlemap object to generate the markers based on their lat lng.
 *
 * @function executeSearch
 */
async function executeSearch() {
  let searchTerm = document.getElementById("search-text");
  // if the users doesn't search for anything redirect back to the same
room
  if (!searchTerm) {
    location.replace("/browse-room");
    return;
  }
  // if the users is looking for a users than we know that the username
starts with an alphabet
  // else will be an address
  let isUser = /[a-zA-Z]/.test(searchTerm.value.charAt(0));
  if (isUser) {
    let mainContent = document.getElementById("room_results");
    let searchURL = `post/search?search=${searchTerm.value}`;
    let response = await axios.get(searchURL);
    if (!response) {
      location.replace("/browse-room");
      return;
    }
    let newMainContentHTML = "";
    response.data.results.forEach((post) => {
      newMainContentHTML += createPost(post);
      gmap.pinpointLocation(post.address);
    });
    mainContent.innerHTML = newMainContentHTML;
  } else {
    gmap.pinpointLocation(searchTerm.value);
  }
}

/**
 * As the function name suggest, this is the blueprint for each individual
card
 * based on the n number of results.
 *
 * @function createPost
 * @param post
 */
function createPost(post) {
  // edited this output stream with the proper div location to wrap the
button
  return `
    <div id="${post.post_id}" class="item card" style="height:500px">
      <img class="cardImage  max-w-screen-lg mx-auto"
src="images/uploads/posts/${post.thumbnail}" id="${post.post_id}"
alt="room" />
      <div class="cardBody break-words">
```

```
        <p class="cardTitle font-bold text-lg">${post.title}</p>
        <p class="cardAddress">${post.address}</p>
        <p class="cardDescription">${post.description}</p>
        <button class="postButton">
        <a href="/post/${post.post_id}">
        Check Post
        </a>
       </button>
      </div>
    </div>
    `;
}
```

Something that I have found out recently is, because this JavaScript file is loaded in the Dom you can construct different HTML elements by interacting with the Dom’s functions rather than just inserting html code. This change in technique is good for performance, besides that this code is very well formatted.

user.js

```
router.get("/search", async (req, res, next) => {
  try {
    let searchTerm = req.query.search;
    if (!searchTerm) {
      res.send({
        results: [],
      });
    } else {
      let results = await User.search(searchTerm);
      if (results.length) {
        res.send({
          results: results,
        });
      } else {
        let results = await User.getTenMostRecent(10);
        res.send({
          results: results,
        });
      }
    }
  } catch (err) {
    next(err);
  }
});
```

The chosen code style is to use aberrations and other JS features in order to create tight and compact code, and adherence to the coding style is very good. There is not a lot of white space, and it is very compact. compact meaning the code both does not take up a lot of vertical or horizontal space. It also removes a lot of redundant statements and uses method chaining, when necessary, in order to keep code bloat down. Overall, this gets a 100% adherence to the chosen code style.

Users.js

```
User.search = async (searchTerm) => {
  let baseSQL =
    "SELECT user_id, first_name, last_name, gender, dob, occupation,
fields, school, email, username, photopath, description, CONCAT(' ',
username, first_name, last_name, occupation) AS haystack FROM users HAVING
haystack like ?;";
  searchTerm = "%" + searchTerm + "%";
  return await db
    .execute(baseSQL, [searchTerm])
    .then(([results, fields]) => {
      return results;
    })
    .catch((err) => Promise.reject(err));
};
```

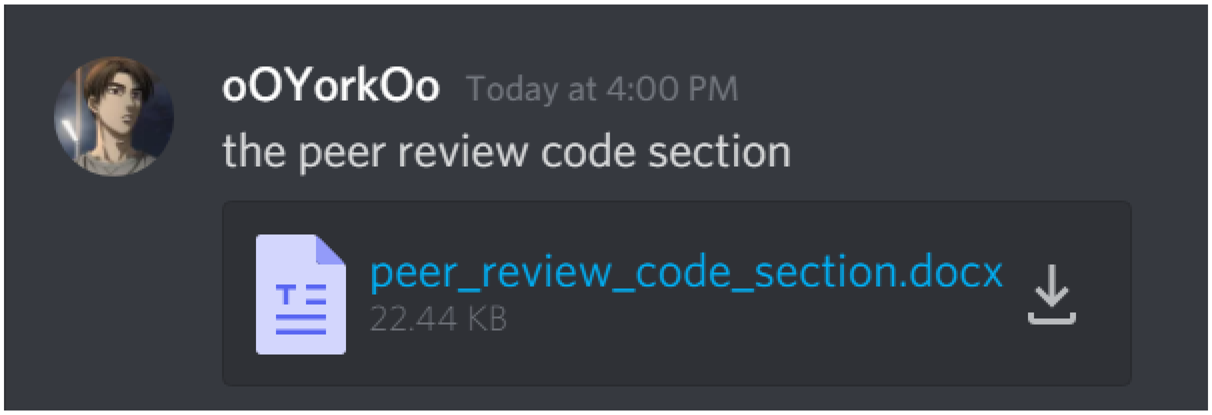
```
User.getTenMostRecent = async (numberOfPosts) => {
  let baseSQL =
    "SELECT user_id, first_name, last_name, gender, dob, occupation,
fields, school, email, username, photopath, description FROM users ORDER
BY created DESC LIMIT " +
    numberOfPosts +
    ";";
  return db
    .execute(baseSQL, [numberOfPosts])
    .then(([results, fields]) => {
      return results;
    })
    .catch((err) => Promise.reject(err));
};
```

In my personal opinion I don’t particularly care too much for the hyper abbreviations of code and the different shortcuts JavaScript provides. I prefer verbose code that tells the reader what it is you are doing. I understand that this is how it’s done in industry, but I don’t think that because it’s done in the industry that it means it’s the end all be all of what good code readable code could be. So many conners are cut while working in the industry and I think that one of those things is hyper readable code. If you look at obfuscated code (code that is meant to be difficult to read on purpose) it has a lot of similarities to this hyper compact style. I would think the goal of writing readable code would stray far away from code that looks obfuscated. I understand that I am in the minority here, but I will honestly state how I feel about this style of code. This however I just my opinion against this style and not against your talent in implementing this style. My previous concerns about hyper compact code aside this is another good example of code that follows its style to the t. An example of good code from this section is how the promise chaining is set up. The code is in nice vertical rows making it easy predict where logic beings and ends for different promises. Another good example of your code overall is the html codes perfect indentation.

b) Chose the code (substantial portion of it) related to the feature you used for QA and usability test. You need to submit an example of the code under review (or part of it – 2 pages or so MAX) for this function to be peer reviewed, and document this as follows:
1. One team member should submit code to other team member(s) for peer review.
2. Peer review should be performed by other group member(s) (1 review is OK).
3. Peer review is to be done by e-mail and comments are to be included in the code
4. Submit the e-mail containing or screen shot of the peer review and commented code and e-mail communication related to this in your Milestone 4 document.

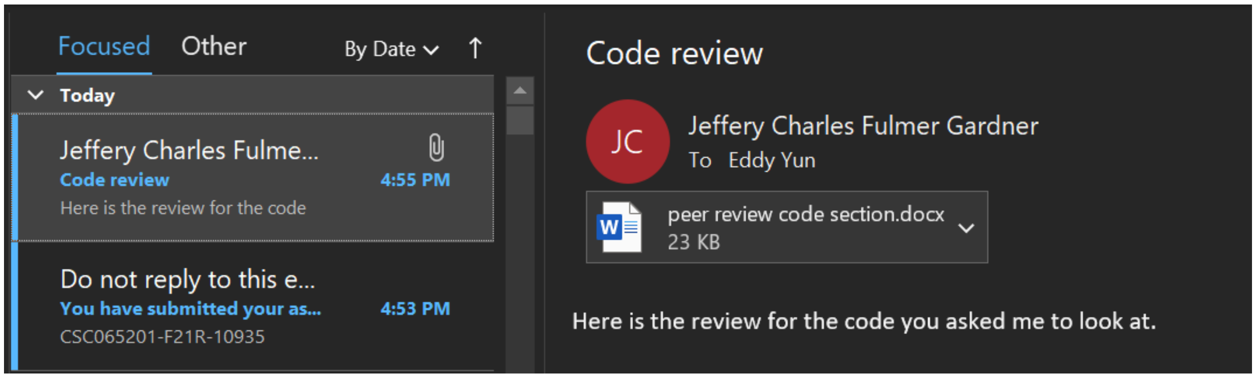
Important: It is critical that code reviews are friendly and helpful, intended to help and education, and not to criticize. It is strongly suggested that you use peer review in the development of the whole system. Reviewers should also check for at least minimal code header and in-line comments and flag this as a problem if this is not adequate

Note: peer review must include checking for basic header and in-line comments


oOYorkOo Today at 4:00 PM

the peer review code section

peer_review_code_section.docx
22.44 KB



Jeffery Charles Fulme...
Code review 4:55 PM
Here is the review for the code

Do not reply to this e...
You have submitted your as... 4:53 PM
CSC065201-F21R-10935

Code review

Jeffery Charles Fulmer Gardner
To Eddy Yun

peer review code section.docx
23 KB

Here is the review for the code you asked me to look at.

5) * Self-check on best practices for security*** (1⁄2 page)**

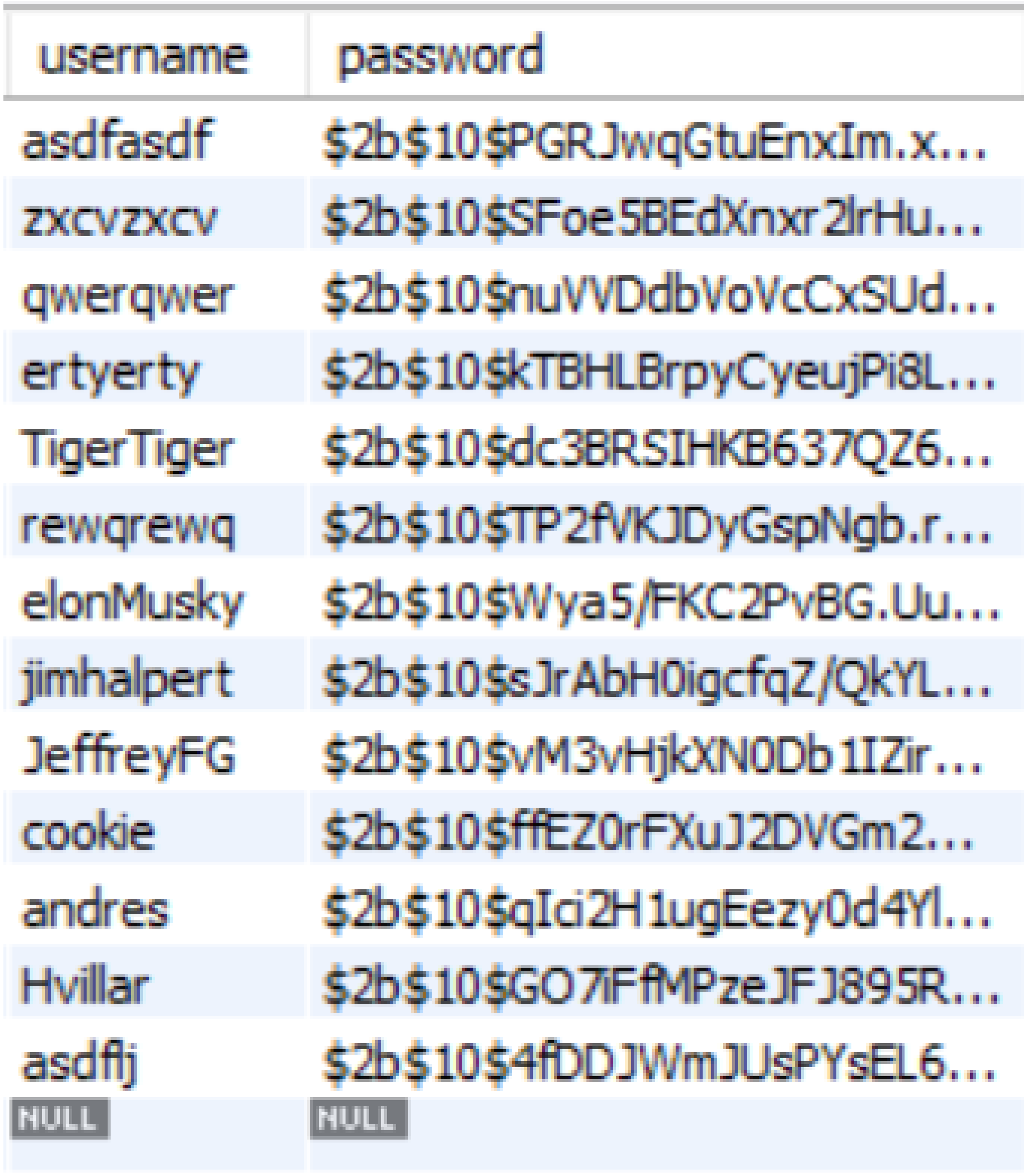
- List major assets you are protecting
    - User information
        - Username
        - Email
        - Password

- Say how you are protecting each asset (1-2 lines of text per each)
    - Username is protected through a simple database query that will check to see if there are any username that exist in our database. Emails are also checked in this matter.

    - Email is protected through an express middleware called express-validator which is to check if the user has entered a correct email which will sanitize the email as well. There are other functions that we could easily apply to it such as normalizeEmail(), .not(), isEmpty(), .trim() and so forth.

- With password, we leveraged the node library called bcrypt that uses a blowfish cipher that salts our password. The value in the second parameter of the hash function call is the number of salts applied to our password and then stored into our database.

- Confirm that you encrypt password in the database



username	password
asdfasdf	$2b$10$PGRJwqGtuEnxIm.x...
zxcvzxcv	$2b$10$SFoe5BEdXnxr2lrHu...
qwerqwer	$2b$10$nuVVDdbVoVcCxSUd...
ertyerty	$2b$10$kTBHLBrpyCyeujPi8L...
TigerTiger	$2b$10$dc3BRSIHKB637QZ6...
rewqrewq	$2b$10$TP2fVKJDyGspNgb.r...
elonMusky	$2b$10$Wya5/FKC2PvBG.Uu...
jimhalpert	$2b$10$sJrAbH0igcfqZ/QkYL...
JeffreyFG	$2b$10$vM3vHjkXN0Db1IZir...
cookie	$2b$10$ffEZ0rFXuJ2DVGm2...
andres	$2b$10$qIci2H1ugEezy0d4Yl...
Hvillar	$2b$10$GO7iFfMPzeJFJ895R...
asdflj	$2b$10$4fDDJWmJUsPYsEL6...
NULL	NULL

As you can see, each and every password will be encrypted using bcrypt

- Confirm Input data validation (list what is being validated and what code you used) -- we request you validate search bar input for up to 40 alphanumeric characters;

6) Self-check: Adherence to original Non-functional specs – performed by team leads(TODO…)

Copy all original non-functional specs as in a high level application document published at the very beginning of the class. Then for each say either: DONE if it is done; ON TRACK if it is in the process of being done and you are sure it will be completed on time; or ISSUE meaning you have some problems and then explain it.

Note: *you must adhere to all original non-functional specs as published in the original high level specification document. Failure to do so may cause reduced SE Product grade.*

List of Non-Functional Requirements (Copy from Milestone One)

Non-Functional Requirement	Complection Status
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 this class, some may be chosen by the student team but all tools and servers have to be approved by class CTO).	DONE
Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers.	DONE
Selected application functions must render well on mobile devices.	DONE
Data shall be stored in the team's chosen database technology on the team's deployment server.	DONE
No more than 100 concurrent users shall be accessing the application at any time.	
Privacy of users shall be protected, and all privacy policies will be appropriately communicated to the users.	
The language used shall be English.	DONE
Application shall be very easy to use and intuitive.	DONE
Google maps and analytics shall be added.	DONE

No e-mail clients shall be allowed. You shall use webmail.	
Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI.	DONE
Site security: basic best practices shall be applied (as covered in the class).	DONE
Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development.	DONE
The website shall prominently display the following exact text on all pages “SFSU Software Engineering Project CSC 648-848, Fall 2021. For Demonstration Only” at the top of the WWW page.” (Important so not to confuse this with a real application).	DONE