Milestone 04 – 04/19/2022

SW Engineering CSC648 Spring 2022 Section 02

Team 06

Application Name: "GitJob"

Team: Allison (Team Lead), TJ (Github Master), Joshua (Front End Lead), Sedric (Front End Team Member), Thien (Back End Lead)

Email: aadad@mail.sfsu.edu

History Table:

Milestone	Date Submitted	Revision Date
M1	**02/22/22**	**3/06/22**
M2	**03/08/22**	**3/26/22**
M3	**04/05/22**	**4/05/22**
M4	**04/19/22**	

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

List of Finalized Functional (P1) Requirements:

1. Employers (companies, recruiters) shall be able to post job listings within these 9 fields: Artificial Intelligence and Machine Learning, Robotic Process Automation (RPA), Edge Computing, Quantum Computing, Virtual Reality and Augmented Reality, Blockchain, Internet of Things (IoT), 5G, and Cyber Security. Students shall be able to search for jobs within these fields.
2. Users shall be able to search and filter through job posts, through filters such as location, field (tech area), job position/title, skills.
3. Employers shall be able to post (and delete) job listings that must contain a job title, description, location, salary, and skills.
4. Registration form: required for users to register. Contains name, email and password. Stored in the database.
5. Users will have a "Saved Jobs" folder that allows them to view the jobs they have bookmarked/saved from searching.
6. Users shall be able to explore the site without creating an account, but must create an account if they want to apply to jobs, post a job listing, or save jobs.
7. Once logged in users shall have sections on the navigation bar for jobs, profile, careers, and post jobs (only if they are employers).
8. The application has a section specifically for searching through job posts with filters, under the Jobs tab in the navigation bar.

9. Students who are registered shall be notified about job postings (by admin) related to them based on the students skills, or interest in the 9 fields.
10. Admin shall have capabilities to delete job postings.
11. Employer/Company shall be able to delete a post after it is posted.
12. Application shall not allow messaging between users.
13. Students shall be able to see the job listings they applied to under a section called 'Applications'.
14. System database shall store job listings, account information (employers and students).

Our product (Gitjob) differs from others as it has a save-job feature, which allows users to compile a list of job postings in which you are interested and might eventually apply to, as well as searches for jobs in 9 specific technology fields, and is user friendly.

Link to product: http://13.52.76.208

2) Usability test plan – max 2 pages

SEARCH JOBS FEATURE

Write a usability test plan for this selected function. Please consult class material on developing usability test plans and questionnaires. This test plan is to contain separate sub-sections titled as below:

- Test objectives:

- The Search Jobs Feature is being tested because it is the main feature of the ‘Gitjob’ application.
- The search filters, which are part of the Search Jobs Feature, will be tested so students can sort jobs based on location, skill, and field.
- The Search Job Feature is being tested to verify the correct results are being printed out, and the SQL connection is working properly.
- The Search Job Feature is being tested to verify the results displayed in an clear and easy-to-understand manner.

- **Test background and setup -** separate paragraphs each covering: System setup, starting point, who are the intended users, URL of the system to be tested and what is to be measured (for M4 focus only on user satisfaction evaluation e.g Likert tests). Up to 1 page

System setup:

The user just has to open up a web browser (preferably Google Chrome). Once they are on the website they can go to http://13.52.76.208 , and here they will see the Gitjob app and the first page they will be directed to is the ‘Jobs’ page, where the user will see a search bar with filters.

Starting Point:

User will go to the URL: http://13.52.76.208 , then they will see the side navigation bar and click on ‘Jobs’. Then the user can type in anything to search for jobs, or can just press ‘search’ for all

jobs to appear. The user can also use filters to find jobs. The user will be able to view the job results on the same ‘Jobs’ page and be able to interact with the job post (either save or apply).

To use this feature the user doesn't have to be logged in. In order to save or apply to a job, the user must register or be logged in. This feature is the first thing that appears when a user enters the ‘Gitjob’ website. Users are able to access this Search Bar Feature from the navigation bar which is on the left side of the website, under the ‘Jobs’ tab.

Intended Users:

The intended users are meant to be the employers and companies with higher than average computing skills, who are familiar with job-searching platforms. Students will mainly use this feature to search jobs based on the filters (skill, location, field), and will be able to save these jobs so they can apply for them in the future. Companies/Employers will also use this feature to verify the jobs they have posted have been successfully published. Companies/Employers will be able to search the job the same way as students and have access to filters as well.

URL: http://13.52.76.208

-- **Usability Task description**: Write them in a separate paragraph in the format of instructions for the usability tester what to do e.g. describe the task testers do before filling out the Likert questionnaire and do the assessment

- Say how would you measure effectiveness:

By whether the user is able to visibly see search results load on the same page after pressing the ‘search’ button.

By whether the user is able to successfully attain different search results by changing filters.

By whether the user is able to clearly read and understand the search results loaded.

- Say how you would measure efficiency (in 3 lines of text):

By how fast the search results are able to appear once the user is able to select filters and press search.

By how easy it is for users to select and unselect filters and search.

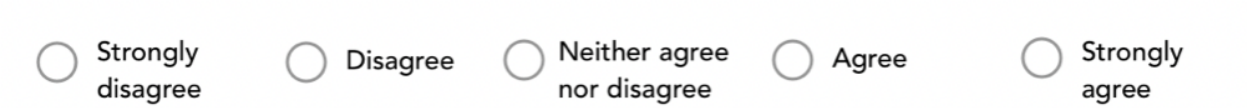
By how many clicks it takes for the user to load results for a desired search.

- **Lickert subjective test:** Provide 3 Lickert scale questions getting user satisfaction after the above task has been performed, in a proper format as it is to be used by reviewer (check class slides) – 3/4 page

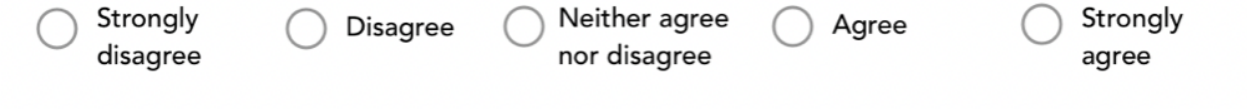
The search filters are easy to use:

Strongly disagree	Disagree	Neither agree nor disagree	Agree	Strongly agree

The search results displayed are easy to understand and read:



Searching for a job took too long or too many clicks:



3) QA test plan - max 2 pages

Test objectives: Search Bar Feature

- **HW and SW setup** (http://13.52.76.208): we are using AWS as the server, VS Code to store our (Javascript and React) code, and using MYSQLWorkbench to manage our SQL databases.
- After analyzing the usability tests our team will also be creating a QA test plan. We will be testing our search feature in order to ensure that our algorithm is working efficiently. We have three test cases. First one checks that the search function is working effectively. The second case checks that our first filter is functional. Lastly, the third case examines if our second filter is functional.

Test #: 1

Test Title : Search Function Test
Test Description: Our first test is going to consist of checking if the function is working. Type a job and the list of the database should be showing the similar job posts.
Test Input: “Name of Job”
Expected Correct Output : A database with the similar jobs will be printed out. First Name of Recruiter : Last Name of Recruiter : Job Title: Pay: Job Description: Company: Phone Number: Jobs: Location: Skills:
Test Results : pass

Test # : 2
Test Title : Second Search Function Test using one of the filters
Test Description : This second test will be testing the search filters that are included. At times people will not have a specific job. So they will be able to search for nothing and simply filter with the location or skills filter.

Test Input : Nothing in the search engine. The Location checkbox should have “California” clicked
Expected Correct Output : The result should print any jobs from the database that have their location set in “California”
Test Results : pass

Test # : 3
Test Title : Third Search Function Test using the second search “Skills” filter
Test Description :This third test will be testing the search “Skills” filters
Test Input : Nothing in the search function. The skills filter should have “C++” clicked.
Expected Correct Output : The expected result should print/ show any jobs from the database that have “C++” as the skill.
Test Results : pass

4) Code Review:

a) OOP, client-server structure, indentation format created by VS Code, camelCase variables, filenames for CSS files are lowercase, filenames for JS will be all lowercase.

b) Search Bar Feature inside home.jsx:

home.jsx | JobPosting.js | Login.js

application > my-app > frontend > src > pages > home.jsx > Home

```jsx
import axios from 'axios';
import React from 'react';
import { useState, useEffect } from 'react';
import { Link } from 'react-router-dom';
import { SidebarData } from '../components/SidebarData';
import "../css/home.css";
import "../css/form.css";
import NavBar from '../components/Navbar';

const Home = () => {


    const [title, setTitle] = useState(""); //state variable for job title
    const [field, setField] = useState(""); //state variable for job field
    const [resData, setResData] = useState([]);

    const { baseUrl } = require("../config/config.js"); // retrieves site url where POST request is sent

    // event handler "handleSubmit" handles 'submit' button event
    const handleSubmit = (event) => {
        event.preventDefault();
        axios.post(baseUrl + ":3001/search", { title, field }) //created POST request to send data to URL
            .then(res => {
                console.log(res.data.result);
                setResData(res.data.result); // sets value

            });
    };

    //generates search results
    useEffect(() => {
    }, []);



    return (
        <div>
```

home.jsx | JobPosting.js | Login.js

application > my-app > frontend > src > pages > home.jsx > Home

```jsx

    return (
        <div>
            <NavBar>
            </NavBar>

            <div className="container">

                {/* side navbar */}
                <div className="sidenav-home">
                    <div className='nav-text'>


                        {SidebarData.map((item, index) => {
                            return (
                                <li key={index} className={item.cName}>
                                    <Link to={item.path}>
                                        {item.icon}
                                        <span>{item.title}</span>
                                    </Link>
                                </li>
                            );
                        })}
                    </div>
                </div>

                <div class="center-scroll">
                    <div>
                        <div className="search-container">
                            <form onSubmit={handleSubmit}>
                                <input className="input-search" type="text" placeholder="Search" onChange={e => setTitle(e.target.value)} />

                                {/* field drown down */}
                                <select id="field" name="field" onChange={e => setField(e.target.value)}>
                                    <option value="">Job Field</option>
                                    <option value="Artificial Intelligence and Machine Learning">AI and ML</option>
```

home.jsx | JobPosting.js | Login.js

application > my-app > frontend > src > pages > home.jsx > Home

```
                                </select>

                                <button type="submit">Search<i className="search" /></button>
                            </form>
                        </div>
                    </div>


                    {/* map function that loads results with the same format */}
                    {resData.map(post => (
                        <div className='search-results-container'>
                            <div>Job Name: {post["job name"]}</div>
                            <div>Job Field: {post["job field"]}</div>
                            <div>Date Posted: {post["date posted"]}</div>
                            <div>Job Description: {post["job desc."]}</div>
                            <div>Job Salary: {post["job salary"]}</div>
                            <br />
                        </div>
                    ))}

                </div>
            </div>
        </div>


    );
};

export default Home;
```

home.jsx | JobPosting.js | Login.js

application > my-app > frontend > src > pages > home.jsx > Home

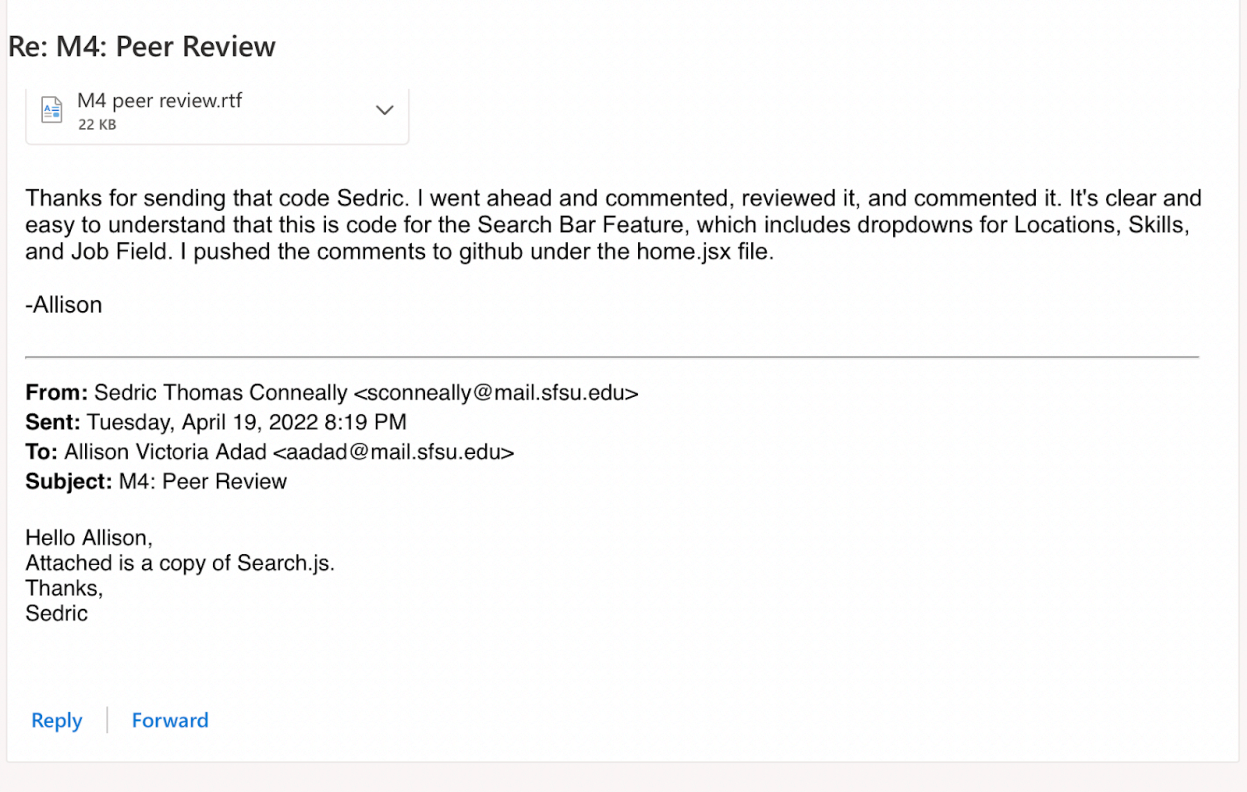
```
                                <option value="">Job Field</option>
                                <option value="Artificial Intelligence and Machine Learning">AI and ML</option>
                                <option value="Robotic Process Automation">Robotic Process Automation</option>
                                <option value="Edge Computing">Edge Computing</option>
                                <option value="Quantum Computing">Quantum Computing</option>
                                <option value="Virtual Reality and Augmented Reality">Virtual Reality and Augmented Reality</option>
                                <option value="Blockchain">Blockchain</option>
                                <option value="Internet of Things">Internet of Things</option>
                                <option value="5G">5G</option>
                                <option value="Cyber Security">Cyber Security</option>
                            </select>

                            {/* location drown down */}
                            <select id="locations" name="field" onChange={e => setField(e.target.value)}>
                                <option value="">Location</option>
                                <option value="California">California</option>
                                <option value="Seattle">Seattle</option>
                                <option value="Texas">Texas</option>
                                <option value="New York">New York</option>
                                <option value="Florida">Texas</option>
                                <option value="Virginia">Virginia</option>
                                <option value="Washington State">Washington State</option>
                            </select>

                            {/* skills drown down */}
                            <select id="skills" name="field" onChange={e => setField(e.target.value)}>
                                <option value="">Skill</option>
                                <option value="C++">C++</option>
                                <option value="Java">Java</option>
                                <option value="Excel">Excel</option>
                                <option value="Python">Python</option>
                                <option value="JavaScript">Java Script</option>
                            </select>

                            <button type="submit">Search<i className="search" /></button>
                        </form>
```

Email:



Re: M4: Peer Review

M4 peer review.rtf
22 KB

Thanks for sending that code Sedric. I went ahead and commented, reviewed it, and commented it. It's clear and easy to understand that this is code for the Search Bar Feature, which includes dropdowns for Locations, Skills, and Job Field. I pushed the comments to github under the home.jsx file.

-Allison

From: Sedric Thomas Conneally <sconneally@mail.sfsu.edu>
Sent: Tuesday, April 19, 2022 8:19 PM
To: Allison Victoria Adad <aadad@mail.sfsu.edu>
Subject: M4: Peer Review

Hello Allison,
Attached is a copy of Search.js.
Thanks,
Sedric

Reply | Forward

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

The assets we are protecting are passwords inside the SQL database. For the password we are encrypting it with bcrypt which uses a salt. For the database we are protecting it from SQL Injection by escaping the query if it detects a certain character to see if it might be a SQL injection. Right now we have a validation feature to check if the passwords match each other, another form of validation we have is the username and password input requirements that must be greater than 0 and at max 30 characters. We also made sure that the emails stored in the SQL

database ‘accounts’ table are unique so users cannot make multiple accounts with the same email.

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

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.

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 have to be approved by class CTO). ON TRACK
2. Application shall be optimized for standard desktop/laptop browsers e.g., must render correctly on the two latest versions of two major browsers ON TRACK
3. Selected application functions must render well on mobile devices ON TRACK
4. Data shall be stored in the team’s chosen database technology on the team’s deployment server. ON TRACK
5. Privacy of users shall be protected, and all privacy policies will be appropriately communicated to the users. ON TRACK
6. The language used shall be English. ON TRACK
7. Application shall be very easy to use and intuitive. ON TRACK
8. Google maps and analytics shall be added ON TRACK

9. No e-mail clients shall be allowed. You shall use webmail. ON TRACK
10. Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI. ON TRACK
11. Site security: basic best practices shall be applied (as covered in the class) ON TRACK
12. Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development ON TRACK
13. The website shall prominently display the following exact text on all pages *"SFSU Software Engineering Project CSC 648-848, Spring 2022. For Demonstration Only"* at the top of the WWW page. (Important so not to confuse this with a real application). ON TRACK