SW Engineering CSC648/848 Spring 2021

648-02 | Team 02 | ProSpector

Cameron Cirini (Team Lead) ccirini@mail.sfsu.edu,
Franklin Arevalo (Frontend Lead), Faisa Jama (GitHub Master),
Tony Cao (Backend Lead), Zhuojun He (Backend Support)

“Milestone 4”

04/20/21

Date Submitted	Date Revised
04/20/21	05/01/21

Product Summary

Name of the product: ProSpector

List of committed functions:

ReqID	Function
F1	**All Users** shall be able to **sign in**.
F2	**All Users** shall be able to **sign up**.
F3	**Students** shall be able to **update** their profile information.
F4	**Students** shall **receive alerts** to get ready for interviews.
F5	**Professors** shall have the ability to **rate students** on a scale from 1-5.
F6	**Professors** shall have the ability to **enter recommendations** for **Students.**
F7	**Recruiters** shall have the ability to **register** to the portal.
F8	**Recruiters** shall have the ability to filter candidates with **advanced search**.
F9	**Recruiters** shall have the ability to **view and download resumes**.
F10	**Recruiters** shall **receive alerts** for all profiles that match saved searches
F11	All **Users** shall have the ability to **delete their profile**.
F12	**Advanced search** shall be done on major and demographic information.

What is unique about your product?

Prospector is unique in that it allows recruiters to sift through a pool of candidates easily by providing them access to the demographic information they want and need. The hiring process can be tedious for both parties, so our platform aims to limit rejection, as well as the time to fill positions, by streamlining the recruiting process. Recruiters on our platform are given the tools they need to stay up to date on new candidates Prospector that meet their hiring needs.

URL to your product: http://54.70.249.83/

Usability Test Plan

Test Objectives:
The search function will be our main object for this usability test plan. The target audiences will be the recruiters who register and use our app for searching the talented students in our school. We will send out an email that contains a survey to the recruiter users to investigate the satisfaction of using our website. The goal of our usability test is to develop a synchronous measurable data to evaluate the users experience of using our app, which can give us direction on which field we should focus on to improve our app in the future. From this test we hope to gain feedback on the ways in which we can improve the search functionality of our app. We want to discover if users find the search intuitive, and if our search yields the results they are expecting. Search is a key aspect of our app, as it is at the core of what our product offers to differentiate itself from the competition.

Test Background and Setup:
Again, the recruiters who use our app to search for talented students are the intended users for this test plan. System setup and the starting point is after beta launched, we will send a survey to the user by email asking for their user experience. The IP address: http://54.70.249.83/search is the URL of the system to be tested. Effectiveness, efficiency, and satisfaction are going to be measured in this test. These attributes will be evaluated on the scale of 1 through 5. The results from the users will be stored back in another database table for further analysis. Here are some sample likert subjective questions on the survey.

Please use the scale from 1 to 5 to rate your experience about using the talent search page on http://54.70.249.83/search/default/text/default. For example, 1 means strongly disagree, 2 means disagree, 3 means average, 4 means agree, 5 means strongly agree.

1. You can search the talent student by the major filter. (Task 1)
2. You can search the talent student by the gender filter. (Task 2)
3. You can search the talent student by the other options in the dropdown menu. (Task 3)
4. The dropdown option filter is working properly. (satisfaction)
5. The search engine gives you proper information that is satisfactory. (Effectiveness)
6. The talents' profile you found matches with the input you search. (Effectiveness)
7. The search engine gives you results fast. (Efficiency)
8. There is no lagging when using the search app. (Efficiency)
9. There are no unexpected results or bugs when using the search page. (satisfaction)
10. The UI such as the input text field, buttons, dropdown menu are clear and understandable. (satisfaction)
11. The app gives you enough information to guide you how to use it. (satisfaction)
12. The overall experience of using our app is good and you will recommend it to the other people. (satisfaction)

Usability Task Description:

Here are some of the task plans in the usability test:

1. Search talent students by major
2. Search talent students by gender
3. Search talent students by the other options in the dropdown menu

In our survey questions above, Question 1 through 4 is used to evaluate the completeness and satisfaction of the filter in the search engine. Intent users just need to rate these questions, and it will give us feedback on how well we are doing in these particular areas in the search page.

In addition, product effectiveness is defined as “accuracy and completeness of user goal achievement”. In our survey, there are a few questions such as Q5 and Q6 which consider whether the search result is proper and helpful or not. Intent users will be asked to give a score from 1 to 5 to rate the effectiveness of our app.

And efficiency is defined as "resources spent by the user in order to ensure accurate and complete achievement of the goals". To apply this theory on our app, efficiency is about reaction time related to the software. As usual, recruiter users will be given questions such as Q7 and Q8 to rate the efficiency of our app. The input users type in, the response of the search engine should react fast. We don’t want any lag or no response finally.

Final Report & Further Analysis:
All of the survey feedback will be sent back to the server for data analysis. An evaluation table will be constructed in our database. There is a sample table below. Each questions’ feedback will update the accumulated rating in the evaluation table. So, the instant results can give us real time feedback on any new change we make in the search page. Our target is to achieve the score of 4 which means agree on the questions. Any accumulated rate below 4 will be the areas we need to improve.

Search Function Evaluation Table:

Question Number	Description	Number of answers	Accumulate Rating
1	Search by major	40	4.2

QA Test Plan

Test Objectives
We will be testing the search function for the recruiter again for the QA test. The search function is held on the recruiter’s home page. The recruiter has the option to search the full list of students or filter by the different inclusivity categories. We will be testing the accuracy in the search functionality and the filters this time.

These accuracy tests are important, as they in tandem with our usability tests can be used to enhance the user experience by providing a product that works as both the developers and end-users intend. By checking each result and comparing it with our design expectations we will be able to ensure that the implementation was accurate.

After completing this QA test, we expect the search page to be fully functionable. The target users are able to search qualified students by typing what they need in the text field. A corresponding result page will be displayed under the search bar. Any invalidate and irrelative search term should generate empty results. Other than that, recruiters should be able to use the options in the dropdown menu to filter the results. We expect the user can perform the search by major, gender, and all the other demographic options in our dropdown menu.

HW and SW setup & URL:

The target testers are not only the recruiter users this time, but also can be the development team. The search app is at http://54.70.249.83/search/default/text/default. All the Inspectors will be invited together to join the QA testing event in the office a couple of times before the beta launching. Each testing task only contains boolean results. It can either pass or fail. Our goal is passing all the tests in the QA test plan list. Any tests where it fails are the area we need to fix. The input text field and the options in the pull-down menu will be the feature to be tested. Here are some features we plan to test.

- *Search all (default default)*. Without inputting anything, the users only need to hit the 'search' button, the search engine will display all talented students in our database. Expected result contains: Syble Hamill, Dawson Macejkovic, Heather Ledner, Roslyn McGlynn, Monica Conroy, Aubree Altenwerth, Zakary Mertz, Noe Luettgen, Vergie Grant and Thad Cormier.
- *Search category and text (const const).* For example, use 'computer science' as the input text, choose 'major' on the filter. The expected result is Aubree Altenwerth; Entry 'male' as the input text, choose 'gender' on the filter. The expected result is Dawson Macejkovic, Roslyn McGlynn, Aubree Altenwerth and Noe Luettgen.
- *Search category no text (default const) (error no search).* The users choose an option in the filter without entering any search text. The expected result will be an error message popping up.
- *Search default category just text (const default) (empty result).* The users enter a search text without choosing an option in the dropdown menu. The expected result is empty. Nothing will be displayed.
- *Search validation by using an illegal string. (empty result).* The users enter an invalidated string such as "aaa!^aa000134AA", and choose any option in the filter. The expected result is also empty.

QA Test Plan Table:

Please refer to the expected correct outputs on the list above for more detailed results, as this table contains summarized information due to space constraints. Additionally, these tests have been applied on Chrome, Edge and Firefox all yielding the same results.

Test#	Test Title	Description	Test Input	Expected Correct Output	Test Results
1	Display all	Search all	Hit ‘search’ button without input anything	Get all students in our database	pass
2	Search by filter	Search by major	Enter “computer science”, choose “major” in the dropdown menu	Get at least one result, and the major of all the results are computer science	pass
3	Search by filter	Search by gender	Enter “male”, choose “gender” in the dropdown menu	Get at least one result, and all students are male.	pass
4	Input validation	Search category no text	Choose “major” in the dropdown menu without enter any search text	Error message pops up	pass
5	Input validation	Search default category just text	Enter “computer science” in the textbox, without touch any options in the filter	The result is empty	pass
6	Input validation	Use an invalid string for search	“aaa!^aa000134AA” as input, choose “major” in the filter	The result is empty	pass

Code Review:

Jose Franklin Arevalo Ruiz April 16, 2021 at 12:12 PM

4) Code Review Details

To: Faisa Jama, Cc: Cameron Matthew Cirini, Zhuojun He, Tony Trung Cao

Hi Team,

Our chosen coding style is 2 tabs for indentations and camelcase for our variables and functions.

Here is the handleClick function which handles the search button click. I am using a const called active which is set to true to display the search and if and else statement to make search the search bar is not empty when selected value is filtered. Can you please look over it and provide feedback!

```
function handleClick(e) {
    // if statement to check if dropdown is selcted but search is empty

    e.preventDefault();
    console.log('The button was clicked.');
    setActive(true);

    if (selectedValue != "default" && searchValue == "default"){
        alert("Search bar is empty!");
    } else {
        axios.get(`${API_BASE}/search/${selectedValue}/text/${searchValue}`)
        .then(res => {
            setStudents(res.data);
        })
        .catch(err => {
            console.log(err);
        })
    }
}
```

Thank you,
Franklin

Team 2 - Member

Faisa Jama Yesterday at 9:15 PM

Re: 4) Code Review Details

To: Jose Franklin Arevalo Ruiz, Cc: Cameron Matthew Cirini, Zhuojun He, Tony Trung Cao

Hey Franklin,

Two tabs for indentations and camelcase for the variables and functions is a great style you have chosen. It is neat, and easy on the eyes of your fellow teammates to follow.

The handleClick function looks good. The function is written in way which is easy to understand what it being implemented. The user can search without filtering, but must have some text in the search bar when a filter is chosen. Even though this function is quite self-explanatory, I would suggest adding single-line comments for better understanding in the future.

This is great work!

Best regards,
Faisa Jama
Team 2 - GitHub Master

Hey Franklin,

Thanks for submitting this, the logic looks sound and is easy to follow aside from a few concerns regarding the styling:

1. Spacing and font should be consistent to each other and themselves
    a. Specifically, spacing and font family change within the second comment

```
//if  dropdown is selected with input in search
```

2. Comments should be aligned with the code blocks they refer to or inline when applicable

```
        // if dropdown is selected and search is empty
alert("Search bar is empty!");
```

3. Avoid comments that may be redundant to conditional statements in the code itself

```
if (selectedValue != "default" && searchValue == "default"){
          // if dropdown is selected and search is empty
    alert("Search bar is empty!");
        } else {
    //if  dropdown is selected with input in search
```

4. Nested blocks should be aligned to be more readable.
    a. Specifically, the axios.get() should be aligned so that it is more clearly within the else block

```
    } else {
//if  dropdown is selected with input in search
axios.get(`${API_BASE}/search/${selectedValue}/text/${searchValue}`)
        .then(res => {
            setStudents(res.data);
        })
        .catch(err => {
            console.log(err);
        })
    }
```

Best,
Cameron Cirini
Team 2 - Team Lead

Self-check on best practices for security:

1. **List of Assets protecting**
    a. User password
    b. User data and information in database
    c. GET/POST routes to backend server
    d. Assets from file system

2. **How to protect**
    a. We will use a JavaScript module named bcrypt to encrypt a user's password when they sign up before inserting into the database. When a user signs in with their original password, bcrypt to decrypt and hash to password to match with an encrypted password that is in the database.
    b. We will protect the users' data only allow queries to databases if a user is signed into their respected account.
    c. Any request to the backend server, whether it be a simple request to a backend route or query request to database will require users to be logged into their account or register for an account.
    d. The file system for resumes is linked to the database therefore, requests to access the assets file system will require users to be logged into their account or register for an account. Furthermore, we will validate users when they request/query a resume.

3. **Confirm encrypting password**
    a. Confirming that user password will be encrypted before inserting into the databases. USING: bcrypt

4. **Confirm input validation**
    a. At registration, the frontend will validate input fields like email format and password requirement, then the backend will do another validation check along with a password encryption before inserting into the database.
    b. At sign in, the backend will do a validation check to confirm that the user exists before returning the user's data to the frontend.

Adherence to original Non-functional specs:

NF1	**Application** shall be developed.	Application shall be developed, tested and deployed using tools and servers approved by Class CTO and as agreed in M0	DONE
NF2	**Application** shall work on desktop/laptop.	Application shall be optimized for standard desktop/laptop browsers e.g. must render correctly on the two latest versions of two major browsers	DONE
NF3	**Application** must render well on mobile	Selected application functions must render well on mobile devices	DONE
NF4	**Data** shall be stored in database	Data shall be stored in the team's chosen database technology on the team's deployment server.	DONE
NF5	**Application** shall handle loads of under 100 concurrent users	No more than 100 concurrent users shall be accessing the application at any time	DONE
NF6	**Application** shall protect the privacy of the users	Privacy of users shall be protected, and all privacy policies will be appropriately communicated to the users.	ON TRACK
NF7	**Application** shall be displayed in english	The language used shall be English.	DONE
NF8	**Application** shall be used by anyone	Application shall be very easy to use and intuitive.	DONE
NF9	**Google maps and analytics** shall be added	Google maps and analytics shall be added	ON TRACK
NF10	**No email clients** shall be allowed	No email clients shall be allowed. You shall use webmail.	DONE

NF11	**No Pay Functionality** shall be implemented	Pay functionality, if any (e.g. paying for goods and services) shall not be implemented nor simulated in UI.	DONE
NF12	**Site** shall be secured.	Site security: basic best practices shall be applied (as covered in the class)	ON TRACK
NF13	During **Application Development**, modern SE processes and practices shall be applied	Modern SE processes and practices shall be used as specified in the class, including collaborative and continuous SW development	DONE
NF14	**Site** shall display “SFSU Software Engineering Project CSC 648-848, Spring 2021”.	The website shall prominently display the following exact text on all pages "SFSU Software Engineering Project CSC 648-848, Spring 2021. For Demonstration Only” at the top of the WWW page. (Important so not to confuse this with a real application)	DONE