TalentFront - Milestone 4

(Team 5)

SW Engineering CSC 648/848 Spring 2021 Section 02

Beta Launch, QA and Usability Testing and Final Commitment for Product Features (P1 list)

April 20, 2021

Austin Wimberley (awimberley@mail.sfsu.edu) - Team Lead
Battulga Tsogtgerel
Ezra Player
Utkrisht Sharma
Alfonso Duarte-Sarabia

History Table:

Date	Action
March 20, 2021	Submitted for review
May 10, 2021	Date revised/finalized

1) Product summary

TalentFront is a talent pipeline for SFSU students. Recent graduates or current students will be able to look and apply for jobs and employers will be able to job post and search for qualified talent. The idea of TalentFront is to make the talent pipeline easier for employers and make it easier for students to job hunt. Whether it be talent, professors, or employers, all will be able to create an account and share their information. Students will be able to upload resumes and demonstrate what they bring to the table for employers. Professors will also be a fundamental part of TalentFront. Professors will be able to review student's resumes and be able to suggest job posts to students. Employers will consist of either talent acquisition, non-profit organizations, or employee research groups. Employers will essentially market themselves through their job posts but will try to find talent that meets their demands. One of ways TalentFront could market themselves would be to promote our product as much as possible. We can use social media platforms like Twitter or through the official SFSU Discord or Slack channels. Discord and Slack are great since they are primarily used by SFSU students and recent graduates so a product like TalentFront could definitely be beneficial to them. Promotion through social media is always going to be a great way for us Students and graduates use LinkedIn to look for jobs but the difference is that TalentFront is tailored specifically for SFSU students and we have the extra ability for teachers to rate students and suggest job postings to students. Having professors with those functionalities really adds a unique type of product. The relationship between professors and students becomes stronger since a professor could help a student more easily get a job TalentFront offers the functionalities of similar competitors like we mentioned with LinkedIn but tailored to a specific audience which in many ways is better and easier when trying to find a job.

Here are our major committed functions (Final P1 functions):

Talent

1) Sign up - Create account.
2) Returning users should be able to log into previously created accounts.
3) After logging in/registering session information should be stored in the client, so that state/information is persisted to their profile.
4) Talent should be requested to enter standard information: name, school, major, and highest degree completed or expected graduation date.

Professors

1) Professors should be able to create specialized accounts which will be verified by using a correct email's domain.
2) Professors will be able to rate students in a scale from 1-5 fashion, being 5 being the highest ratings that implies knowledgeable, responsible, teamwork, leadership, committed to success, etc. and enter recommendations.

Employers

1) Create job postings for their company.
2) Employers should be able to enter their company profile information: Product/Services they offer, company mission, number of employees, photos of office, office locations, brief introduction of company.
3) Search through talent for prospects with appropriate qualifications.

Not Registered Users

1) Users who don't want to register, should still be able to view talent and job listings.

2) Usability test plan

Test Objectives

One of the major functions that we need to nail in order to be successful is to provide an easy platform for users to fill out a profile on. This profile should consist roughly of the same items that one would put on their resume. The profile should be in a standardized form so that it will be easily searchable on the platform by recruiters looking to fill open positions. This is an important functionality to get correct because it serves as the talent's profile or resume on which they will be reviewed. We want to test that user's intuitively understand how to create a profile on our platform. If they run into trouble creating a strong profile they may be turned away from using our platform.

Each of the different parts of a user's profile should be easily editable. The user's first and last names should be gathered on registration. After that they should be able to easily add and remove skills. The user should also be able to add relevant job experience. This will include their employer's name, job title, start and end dates and description of their responsibilities in the role. Lastly, the user should also be able to add education data, including their school name, major, graduation date, and degree type. Each of these functions is a part of the user's profile creation and should be intuitive.

Test Background and Setup

System Setup

In order for us to test the ability for the user to edit their profile then we can do it locally in our machines. We can go to our terminal and open up the project. For our front end development we are using *react* which is a JS library for our UI components and *react-redux* which is a JS library to manage our application state. We just need to make sure we have *node* and *yarn* installed in our computer. From there we can use yarn, which is a project and package manager. With the command *'yarn install'* we install all the dependencies and scripts that we need for our project. Then we run *'yarn start'* to build and run our react scripts. This should take us to localhost where we will be able to see our application. For the back end we are using freya and gradle alongside MySQL to manage our database. We need to install docker on our computer. Then we run *docker-compose pull* and *docker-compose up* in our docker-compose directory of the project. With docker we have a container of the database so there is no need to download mysql on the computer. We could have MySQL workbench or any other GUI for database management to check elements in the database. Now we can begin with testing.

Starting Point

The starting point when testing will be from the registration page. We will assume that the user will make a new account and enter important information. When they register, we will have important information such their full name, degree, graduation date, etc. This is the type of information that will be useful for the user to potentially later edit or maybe add some more missing information that they will want to add. We will make sure that after we register the user that all that information is properly stored in the database. Once the account has been made the user will be redirected to their respective profile page.

Who are the intended users

The intended users are either the talent, professors, or employers. Depending on the type of the user the profile page may be different. For example, when the user type is talent then the profile page will have a section where they can put their degree and another section where they can put their skills. And for when the user type is an employer then there will be a section of the profile page where they can put all the information about the company, address, and website.

URL of the system to be tested and what is to be tested

The URL will be localhost:3002/profile. Let's assume we are testing the user type talent, then there is an edit profile button. This should redirect us to a place where we can edit our account info. You can change your email, password, about me page, and skills. You

should be able to save your information and redirect back to the profile page with your updated information.

Usability Task Description

1. Register account - Enter information such as email, name, type of user, etc.
2. Once redirected to profile page, press edit profile button
3. You should be able to edit the about me information, and edit your skills
4. You should be able to add the job experiences and education experiences
5. Click save
6. See updated on the profile page

Lickert Subjective Tests

1. It was easy to register an account:

__ Strongly disagree __ Disagree __ Neither agree or disagree __ Agree __ Strongly agree

2. It was intuitive to add job experience:

__ Strongly disagree __ Disagree __ Neither agree or disagree __ Agree __ Strongly agree

3. It was intuitive to add school information:

__ Strongly disagree __ Disagree __ Neither agree or disagree __ Agree __ Strongly agree

4. It was easy to edit profile page
__ Strongly disagree __ Disagree __ Neither agree or disagree __ Agree __ Strongly agree

3) QA Test Plan

Test objectives: - what is being tested

We are testing the platform that a user will be filling out their information on. When a user registers, they should be able to fill out their account information and have it be saved to the database. When they edit their profile, they should be able to make changes to their account and resume points, such as adding and deleting resume points.

HW and SW setup (including URL): - Feature to be tested
Inputting user profile attributes upon registration.
Editing attributes of a user profile.
Url: lochalhost:3002/profile

QA Test plan: - table format

test #	test title	test description	test input	expected correct output	test results
1	Test filling out user information when registering.	See if inputting information is sent to server and saved.	Email: “john123@gmail.com” first-name:”John” last-name:” Smith” Password:”secretpassword” Confirm-password: “secretpassword” user-Type:” Current Student”	Success!	PASS
2	Test adding a resume point on the user profile edit page.	See if adding a resume point will be saved to the database.	Job-title:”Software Engineer” Company Name: “Bebo” Started-date: “May 2018” End-Date” January 2019” Description: “Worked to develop the frontend.”	Resume point remains on users page after reload.	PASS
3	Test removing a resume point on the	See if the resume point is deleted from	[press delete on resume point]	Resume point is deleted on users page	PASS

	user profile edit page.	the database.		after reload.	

4) Code Review

The Code style that we are using for the backend is defined by the spotless plugin. Spotless is a code formatter that has a standard set of rules that code must have in order for it to pass the build pipeline. By running `./gradlew spotlessApply` it automatically formats all of the code in the entire project to the rules set by spotless. For example one of the rules is no wildcard imports, so the build process will fail if you have one.

In the frontend, we decided to use reactHooks such as useEffect and useState to manage individual components throughout its life cycle. If other parts of the application control the state of the individual component that state should be managed by redux and appropriate slices should be created to manipulate state as well as keep it in sync. All API calls would be created in an individual wrapper function and that wrapper function will be used in the parts of application. This helps us to be consistent in terms of communicating with the backend.

Job creating in Profile page feature #32

Edit | Open with

Open | theTulga wants to merge 1 commit into `master` from `tulga`

Conversation 5 | Commits 1 | Checks 0 | Files changed 5 | +469 −55

theTulga commented 8 minutes ago

No description provided.

sync 86b4b8b

theTulga requested a review from **pachecosf** 8 minutes ago

pachecosf reviewed 5 minutes ago — View changes

`application/fe/my-app/src/screens/profile/profile.v2.js`

```
413 +   return (
414 +     <div>
415 +       <AddNewExperience show={show} handleClose={handleClose} />
416 +       <Navbar />
```

pachecosf 5 minutes ago

What do you think about making Navbar global and putting it on the div returned by App.js ? Also feel like this is where we should put the footer as well, once we get around to making it that is

Reply...

Resolve conversation

`application/fe/my-app/src/screens/profile/profile.v2.js`

```
12 + import Cookies from "universal-cookie";
13 +
14 + const cookies = new Cookies();
15 + const BASE_URL = "http://localhost";
```

pachecosf 5 minutes ago

please review your logic surrounding BASE_URL. It doesn't make much sense your setting the top level var to localhost which your using for image url. Then on the submit function your creating a new URL for window.origin

Reply...

Resolve conversation

`application/fe/my-app/src/screens/profile/profile.v2.js`

```
149 +   }, []);
150 +
151 +   const handleSubmit = (file) => {
152 +     const BASE_URL = window.origin;
```

pachecosf 5 minutes ago

another BASE_URL here

Reply...

Resolve conversation

Reviewers
pachecosf
Still in progress? Convert to draft

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

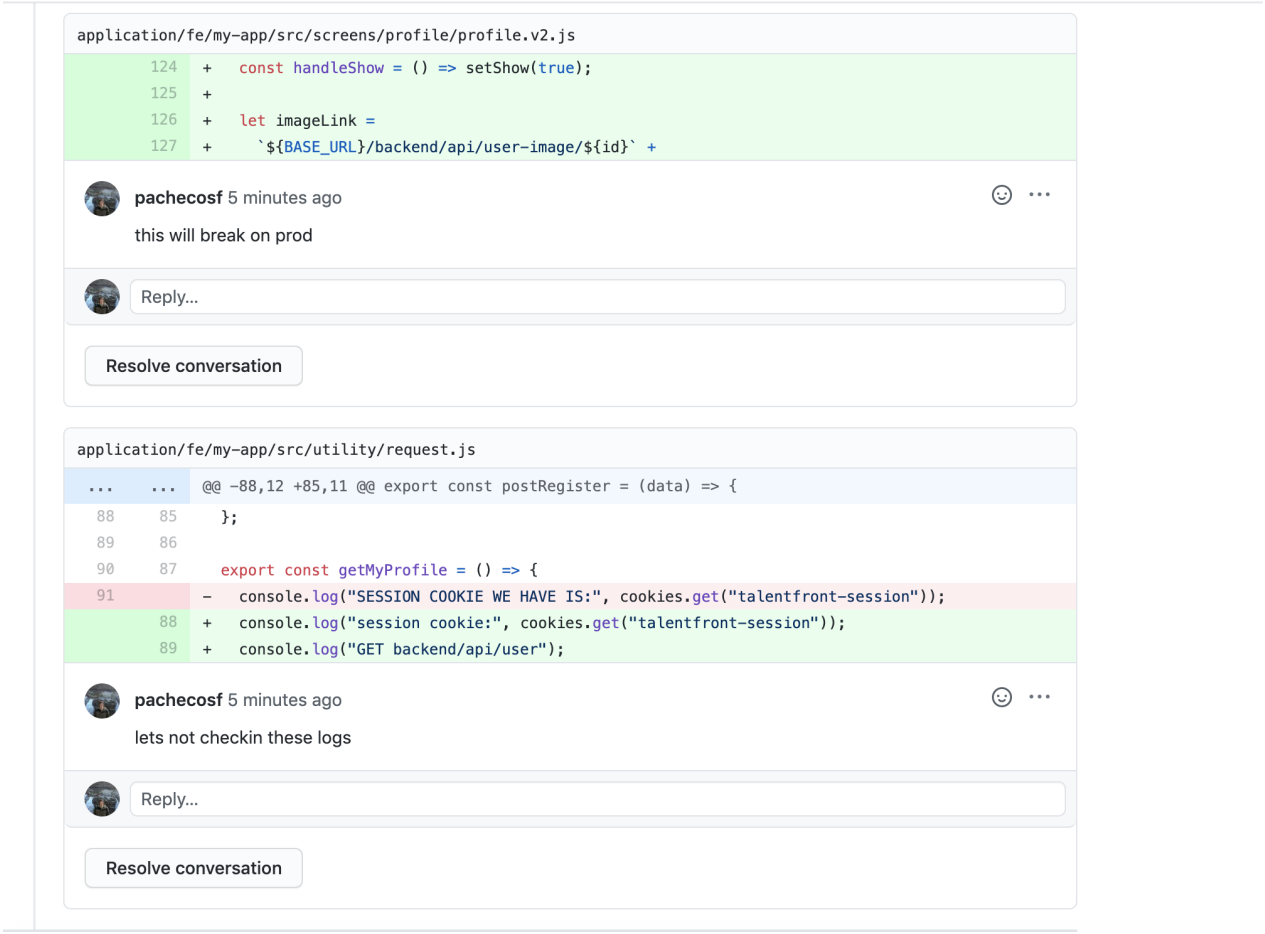
Linked issues
Successfully merging this pull request may close these issues.
None yet

Notifications Customize
Unsubscribe

review was requested.

2 participants

Lock conversation



5) Best Practices for Security

The first major asset we are protecting is passwords. We use a standard salt and hash verification for saving passwords in the database. So in the database there are two columns for the password logic, the first being a salt the second being the hash of the salt and password. When a user tries to login we pull up the authentication record, grab the salt, add it to the password, hash it and verify that the hashes match. By using this approach we protect against rainbow table attacks.

The second asset we want to protect against is changing information that doesn't belong to the user. We have a simple design where each record of information has an owner, and only the owner has permissions to edit the information. In order to edit any information a user must be in a logged in state. To maintain session information we use a cookie with a user ID and session UUID for verification. We then do a backend look up in the session store to verify that this user has a matching session record, and once verified that user is allowed to edit any information associated with that session's user ID.

6) Adherence to original Non-functional Specifications

List of non-functional requirements

- Passwords should be protected and not stored in the database in plain text. They should only be passed to the backend in secured POST messages, and saved/checked using salt and hashing standards.
    - DONE
- There must be a documented term of service that clearly defines how the user's personally identifiable information (PII) is stored and shared.
    - ON TRACK
- Everything should be in English
    - DONE
- The site must make use of Google Maps and Google Analytics.
    - ISSUE
    - Integration with google's API is not on our P1 requirements and it will take a good amount of precious development hours in order to implement. Our goals are to complete all P1 requirements, the integration with Google Maps is our next top priority but I can not at this time guarantee that we will be able to get there.
- Application must be optimized for desktop/laptop and must render correctly on the latest version of two major browsers.
    - DONE
- Application UI shall be easy to use and intuitive.
    - ON TRACK
- Selected application functions must render well on mobile devices
    - ON TRACK
- Application shall be developed, tested and deployed using tools and servers approved by class CTO and as agreed in Milestone 0.
    - DONE
- All data should be stored in the database on the team's deployment server.
    - DONE
- No more than 100 concurrent users shall be accessing the application at any time
    - ON TRACK
- 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 Software Engineering processes and practices covered in the class will be used, including collaborative and continuous development/deployment.
    - ON TRACK
- 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 bottom of the WWW page. (Important so not to confuse this with a real application).
    - ON TRACK