SW Engineering CSC648/848

EzJobs

Section 01, Team 01

Team members

Name	Role
Rishita Meharishi	Team Leader & Frontend
Luai Almaznai	Frontend Lead
L Chow	Backend Lead
Zaw Win Tun	Scrum Master & Frontend
Yee Yang	Github Master & Fullstack

Milestone 4

11/18/2024

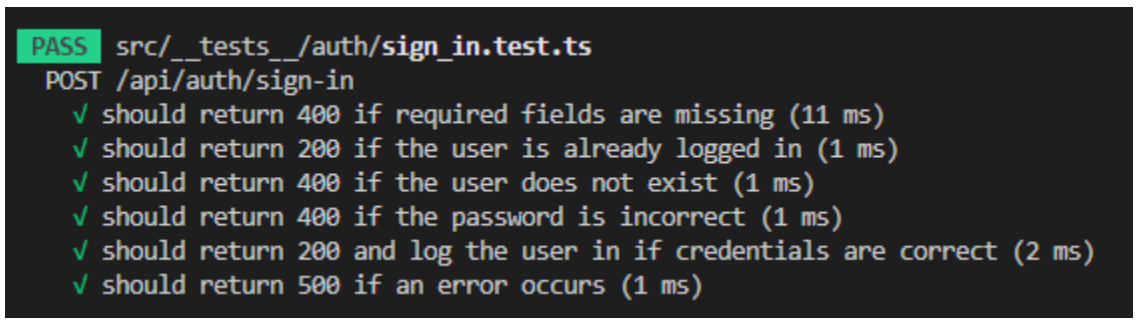
History Table

Revision #	Date	Description
1	11/18/2024	Initial document for milestone 4

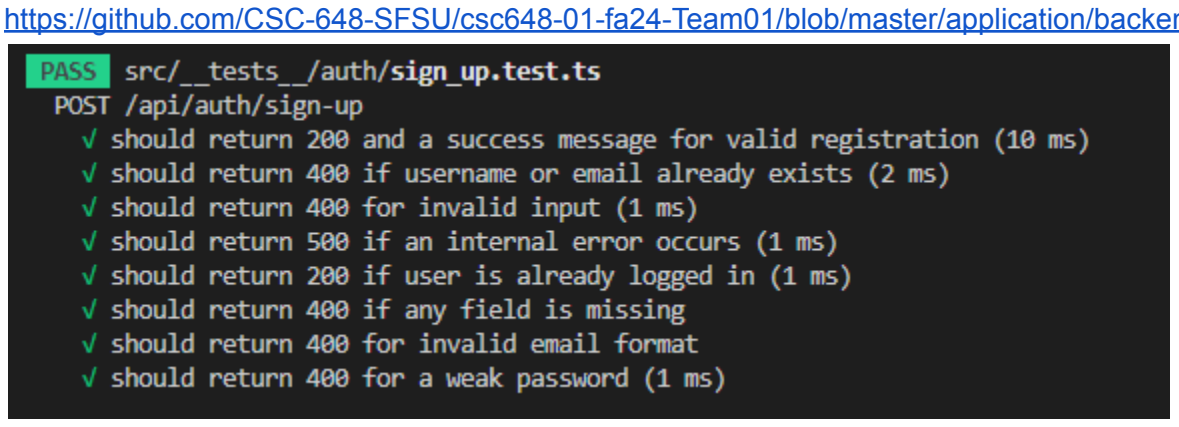
1. QA testing

I. Unit Test

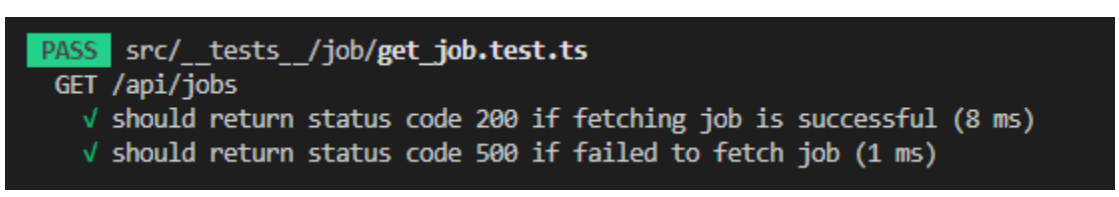
(1). Sign in

- Description: This unit test suite validates the behavior of the `POST /api/auth/sign-in` API endpoint, ensuring it correctly handles various scenarios during user login, including missing fields, already logged-in users, non-existent users, incorrect passwords, successful logins, and unexpected server errors.
- GitHub URL: https://github.com/CSC-648-SFSU/csc648-01-fa24-Team01/blob/master/application/backend/src/__tests__/auth/sign_in.test.ts
- 
```
PASS src/__tests__/auth/sign_in.test.ts
  POST /api/auth/sign-in
    √ should return 400 if required fields are missing (11 ms)
    √ should return 200 if the user is already logged in (1 ms)
    √ should return 400 if the user does not exist (1 ms)
    √ should return 400 if the password is incorrect (1 ms)
    √ should return 200 and log the user in if credentials are correct (2 ms)
    √ should return 500 if an error occurs (1 ms)
```

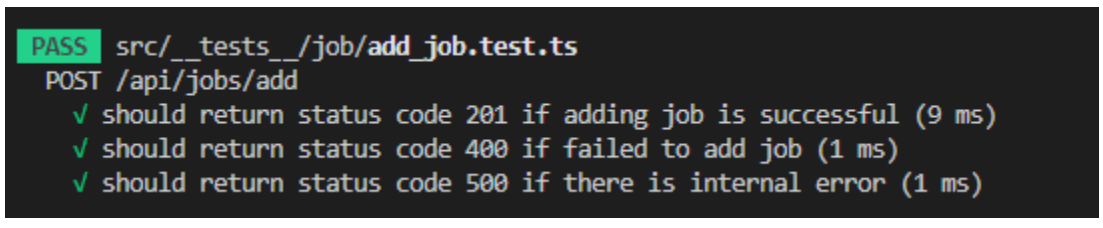
(2). Sign up

- Description: The unit tests validate the functionality of the `POST /api/auth/sign-up` endpoint by checking various scenarios, including successful user registration, handling of invalid input (e.g., missing fields, weak passwords, invalid email formats), pre-existing username or email, internal server errors, and cases where a user is already logged in.
- GitHub URL: https://github.com/CSC-648-SFSU/csc648-01-fa24-Team01/blob/master/application/backend/src/__tests__/auth/sign_up.test.ts
- 
```
PASS src/__tests__/auth/sign_up.test.ts
  POST /api/auth/sign-up
    √ should return 200 and a success message for valid registration (10 ms)
    √ should return 400 if username or email already exists (2 ms)
    √ should return 400 for invalid input (1 ms)
    √ should return 500 if an internal error occurs (1 ms)
    √ should return 200 if user is already logged in (1 ms)
    √ should return 400 if any field is missing
    √ should return 400 for invalid email format
    √ should return 400 for a weak password (1 ms)
```

(3). Get job

- Description:This unit test verifies that the API endpoint `GET /api/jobs` correctly fetches job data for a user from the database and handles both successful and error scenarios, returning the appropriate HTTP responses.
- GitHub URL: https://github.com/CSC-648-SFSU/csc648-01-fa24-Team01/blob/master/application/backend/src/__tests__/job/get_job.test.ts
- 
```
PASS src/__tests__/job/get_job.test.ts
  GET /api/jobs
    √ should return status code 200 if fetching job is successful (8 ms)
    √ should return status code 500 if failed to fetch job (1 ms)
```

(4). Add job

- Description: This unit test verifies that the API endpoint `POST /api/jobs/add` correctly handles successful job additions, failure responses, and error scenarios, ensuring proper interaction with the database and appropriate HTTP responses.
- GitHub URL: https://github.com/CSC-648-SFSU/csc648-01-fa24-Team01/blob/master/application/backend/src/__tests__/job/add_job.test.ts
- 
```
PASS src/__tests__/job/add_job.test.ts
  POST /api/jobs/add
    √ should return status code 201 if adding job is successful (9 ms)
    √ should return status code 400 if failed to add job (1 ms)
    √ should return status code 500 if there is internal error (1 ms)
```

(5). Get contact

- Description: This unit test verifies that the API endpoint `GET /api/contacts/search` correctly handles successfully in several scenarios, such as giving back all contacts when no searchParam is given, when contacts are searched, and when contacts aren't found, among a few other scenarios.
- GitHub URL: https://github.com/CSC-648-SFSU/csc648-01-fa24-Team01/blob/master/application/backend/src/__tests__/contact/get_contact.test.ts
- ```
  router.get("/api/contacts/search", checkAuth, Controller.Contacts.getContact);
  ```
- ```
  PASS src/__tests__/contact/get_contact.test.ts (12.786 s)
    GET /api/contacts
      √ should return status code 200 and all contacts if no searchParam is provided (18 ms)
      √ should return status code 200 and filtered contacts if searchParam is provided (2 ms)
      √ should return status code 404 if no contacts are found (2 ms)
      √ should return status code 500 if there is an internal server error (4 ms)
      √ should handle missing searchParam gracefully (2 ms)
      √ should log the received searchParam (2 ms)
      √ should log the fetched contacts (3 ms)
      √ should log the error message on failure (2 ms)
      √ should handle unknown error gracefully (3 ms)
      √ should handle empty contacts array (3 ms)
  ```

(6). Add contact

- Description: This unit test verifies that the API endpoint `POST /api/contacts/add` correctly handles successfully in several scenarios, such as when adding a new contact and various errors related to it.
- GitHub URL: https://github.com/CSC-648-SFSU/csc648-01-fa24-Team01/blob/master/application/backend/src/__tests__/contact/add_contact.test.ts
- ```
  router.post("/api/contacts/add", checkAuth, Controller.Contacts.addContact);
  ```
- ```
  PASS src/__tests__/contact/add_contact.test.ts (13.032 s)
    addContact controller
      √ should add a new contact and return 201 status (17 ms)
      √ should return 400 status if required fields are missing (2 ms)
      √ should return 500 status if there is an error adding the contact (3 ms)
      √ should handle unexpected errors gracefully (2 ms)
      √ should return 400 status if session user is not available (2 ms)
      √ should return 400 status if session user is undefined (3 ms)
      √ should return 400 status if session is null (7 ms)
      √ should return 500 status with unknown error message if createContact throws a non-Error object (2 ms)
  ```

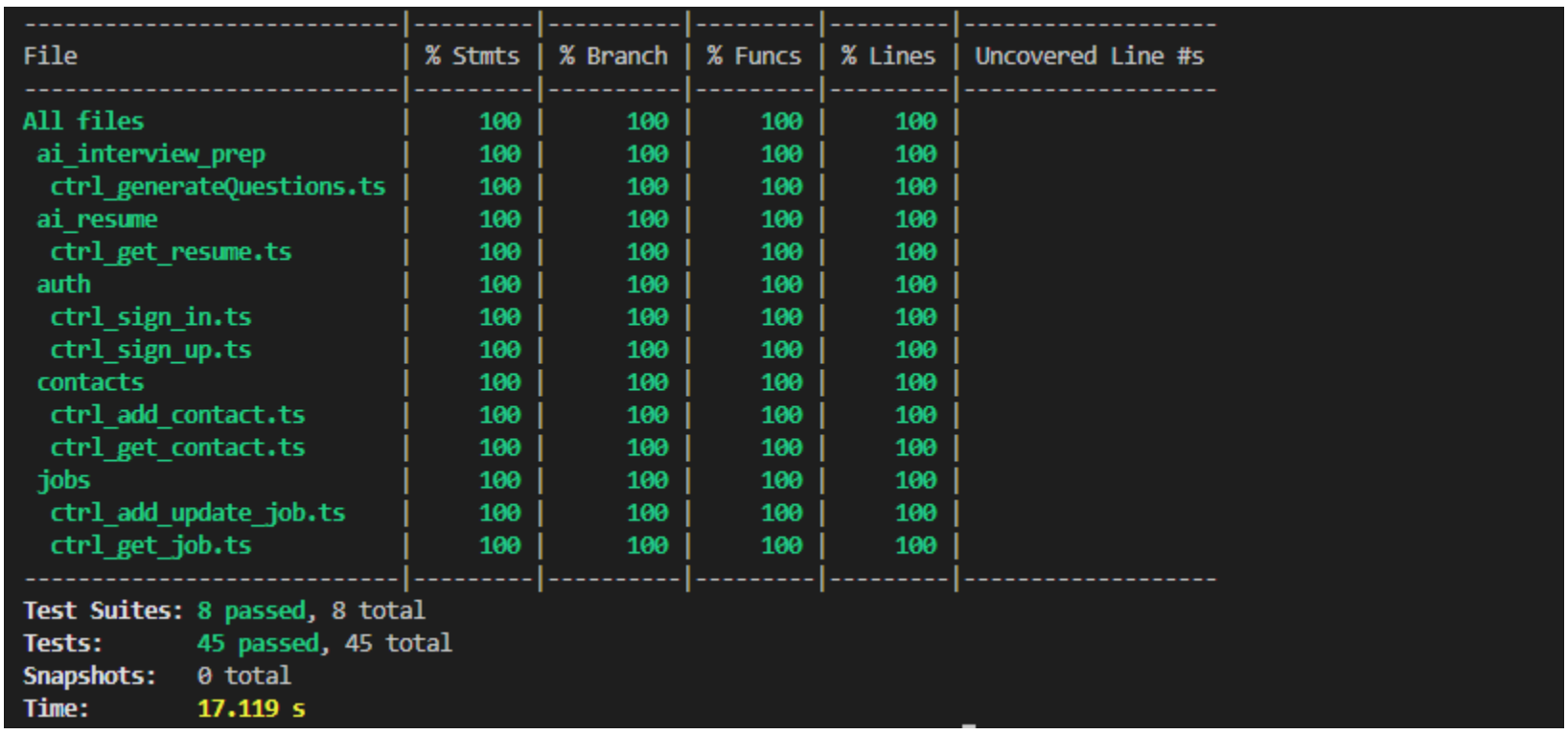
(7). Get chat response

- Description: This unit test verifies that the API endpoint `GET /api/chatbot` correctly handles successfully when responses are received or not from the OpenAI API, among a few other cases.
- GitHub URL: https://github.com/CSC-648-SFSU/csc648-01-fa24-Team01/blob/master/application/backend/src/__tests__/ai_chat/get_chat_response.test.ts
- ```
  router.post("/api/chatbot", getChatbotResponse);
  ```
- ```
  PASS src/__tests__/ai_chat/get_chat_response.test.ts (13.117 s)
    getChatbotResponse
      √ should return 400 if no message is provided (30 ms)
      √ should return 200 with chatbot response (2 ms)
      √ should return 500 if OpenAI API response error occurs (8 ms)
      √ should return 500 if no response received from OpenAI API (5 ms)
      √ should return 500 if an unknown error occurs (3 ms)
  ```

(8). Get resume data

- Description:This unit test validates the `GET /api/ai-resume` api endpoint by checking if it correctly handles cases where the user's resume input data does not exist, exists with specific data, or encounters a database error, ensuring appropriate HTTP status codes and JSON responses are returned.
- GitHub URL: https://github.com/CSC-648-SFSU/csc648-01-fa24-Team01/blob/master/application/backend/src/__tests__/ai_resume/get_resume_data.test.ts
- ```
  PASS src/__tests__/ai_resume/get_resume_data.test.ts
    GET /api/ai-resume
      √ should return status code 200 with hasInput false if user input does not exist (8 ms)
      √ should return status code 200 with hasInput true and resumeData if user input exists
      √ should return status code 500 if an error occurs during database access (1 ms)
  ```
  ```

Coverage Report



```
----------------------------|---------|----------|---------|---------|-------------------
File                        | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------------------------|---------|----------|---------|---------|-------------------
All files                   |     100 |      100 |     100 |     100 |
 ai_interview_prep          |     100 |      100 |     100 |     100 |
  ctrl_generateQuestions.ts |     100 |      100 |     100 |     100 |
 ai_resume                  |     100 |      100 |     100 |     100 |
  ctrl_get_resume.ts        |     100 |      100 |     100 |     100 |
 auth                       |     100 |      100 |     100 |     100 |
  ctrl_sign_in.ts           |     100 |      100 |     100 |     100 |
  ctrl_sign_up.ts           |     100 |      100 |     100 |     100 |
 contacts                   |     100 |      100 |     100 |     100 |
  ctrl_add_contact.ts       |     100 |      100 |     100 |     100 |
  ctrl_get_contact.ts       |     100 |      100 |     100 |     100 |
 jobs                       |     100 |      100 |     100 |     100 |
  ctrl_add_update_job.ts    |     100 |      100 |     100 |     100 |
  ctrl_get_job.ts           |     100 |      100 |     100 |     100 |
----------------------------|---------|----------|---------|---------|-------------------
Test Suites: 8 passed, 8 total
Tests:       45 passed, 45 total
Snapshots:   0 total
Time:        17.119 s
```

The screenshot shows a test coverage report for the 8 unit tests above. Each file is listed along with its test coverage in the following categories:

- **% Stmts (Statement Coverage)**: This represents the percentage of executed statements in the file. All files have 100% statement coverage, meaning all statements in these files are tested.

- % Branch: This indicates the percentage of branches (e.g., if-else conditions) covered by tests. All files have 100% branch coverage, meaning all possible branches are tested.

- **% Funcs (Functional Coverage):** This shows the percentage of functions that have been called in the tests. Every file in the report has 100% function coverage, meaning all functions are tested.
- % Lines: This represents the percentage of lines of code executed during tests. All files have 100% line coverage, meaning every line of code has been executed in the tests.

- Uncovered Line #s: This shows the line numbers of code that is not covered by tests. In this case, all files have 100% coverage, so no uncovered lines are listed.

Continuous Integration (CI) Workflow

Our CSC648 project uses an automated Continuous Integration (CI) workflow powered by **GitHub Actions** to ensure code quality, reliability, and consistency across the frontend and backend.

https://github.com/CSC-648-SFSU/csc648-01-fa24-Team01/blob/master/.github/workflows/ci.yml

Workflow Overview

1) The CI workflow is triggered when a PR is merged into the `master` branch.
2) The workflow executes separate jobs for the frontend and backend.
3) If all checks pass (formatting, testing, and building), the PR is considered safe for merging.
4) If a `package-lock.json` is updated during the workflow, it will automatically commit the changes back to the PR branch with `[ci skip]` to avoid redundant CI runs.

The following checks and operations are performed:

Frontend

1) **Dependency Installation**: Ensures all required dependencies are installed using `npm ci`.
2) **Code Formatting**: Automatically formats any modified code using Prettier.
3) **Build Verification**: Builds the frontend using Vite to confirm there are no build errors.
4) **Lockfile Sync**: Ensures `package-lock.json` is in sync with `package.json`. If not, it automatically updates and commits the changes back to the PR branch.

Backend

1) **Dependency Installation**: Ensures all required dependencies are installed using `npm ci`.
2) **Code Formatting**: Automatically formats any modified code using Prettier.
3) **Testing**: Executes all unit and integration tests with Jest to ensure no tests fail.
4) **Build Verification**: Builds the backend code to confirm there are no errors.
5) **Lockfile Sync**: Ensures `package-lock.json` is in sync with `package.json`. If not, it automatically updates and commits the changes back to the PR branch.

II. Integration Test

Google Sheet Link:

CSC648 FA24 Section01 Team01 - Testing

(1). sign up

Test Case ID	AUTH_01	**Test Case Description**	Test the sign up functionality		
Created By	Yee	**Reviewed By**	L	**Version**	1.0
QA Tester's Log	Initial test				
Tester's Name	Yee	**Date Tested**	11/27/2024	**Test Case (Pass/Fail/Not)**	Pass

S #	Prerequisites
1	web browsers
2	https://ezjobs.onrender.com is live
3	username is not taken
4	email is not taken

S #	Test Data
1	username: test00 (if username is taken, change the number)
2	password: SFSUcsc648
3	email: test00@email.com (if email is taken, change the number)
4	

Test Scenario	Verify that a new user can successfully sign up

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://ezjobs.onrender.com/sign-up	The Sign Up page loads successfully	As Expected	Pass
2	Fill in input field with test data	Form fields are populated without any errors	As Expected	Pass
3	Click the SIGN UP button to Submit the sign up form	Sign up successful prompt appears, and redirect to sign in page	As Expected	Pass
4	Repeat the sign up steps above with same test data	Sign up failed due to username is taken, a prompt appears and states that username or email is invalid	As Expected	Pass

(2). sign in

Test Case ID	AUTH_02	**Test Case Description**	Test the sign in functionality		
Created By	Yee	**Reviewed By**	L	**Version**	1.0
QA Tester's Log	Initial test				
Tester's Name	Yee	**Date Tested**	11/27/2024	**Test Case (Pass/Fail/Not)**	Pass

S #	Prerequisites
1	web browsers
2	https://ezjobs.onrender.com/ is live
3	
4	

S #	Test Data
1	username: test
2	password: SFSUcsc648
3	
4	

Test Scenario	Verify that an existing user can successfully sign in

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://ezjobs.onrender.com/	Page load without any errors	As Expected	Pass
2	Fill in input field with test data	Form fields are populated without any errors	As Expected	Pass
3	Click the SIGN IN button to submit the sign in form	Sign in successful, and redirect the user to	As Expected	Pass

		dashboard page		
4	Navigate to https://ezjobs.onrender.com/ again	Prompt appears and states that user is already signed in, then redirect the user to dashboard page	As Expected	Pass

(3). sign out

Test Case ID	AUTH_03	**Test Case Description**	Test the sign out functionality		
Created By	Yee	**Reviewed By**	L	**Version**	1.0
QA Tester's Log	Initial test				
Tester's Name	Yee	**Date Tested**	11/27/2024	**Test Case (Pass/Fail/Not)**	Pass

S #	Prerequisites
1	web browsers
2	signed in to https://ezjobs.onrender.com/ with test data
3	
4	

S #	Test Data
1	username: test
2	password: SFSUcsc648
3	
4	

Test Scenario	Verify that a signed-in user is successfully signed out, and the session is terminated properly.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://ezjobs.onrender.com/app	The homepage is displayed with a button to sign out in top-right corner	As expected	Pass
2	Click the sign out button	User is sign out and redirect to sign in page	As expected	Pass
3	Navigate to https://ezjobs.onrender.com/app again without signing in	A prompt appears and states that authentication error occurred, then the user is redirected to sign in page	As expected	Pass

(4). add job

Test Case ID	J_01	**Test Case Description**	Test the add job functionality in the job board page		
Created By	Yee	**Reviewed By**	L	**Version**	1.0
QA Tester's Log	Initial test				
Tester's Name	Yee	**Date Tested**	11/26/2024	**Test Case (Pass/Fail/Not)**	Pass

S #	Prerequisites
1	web browsers
2	signed in to https://ezjobs.onrender.com/ with test data
3	
4	

S #	Test Data
1	username: test
2	password: SFSUcsc648
3	
4	

Test Scenario	User wants to add a new job application

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://ezjobs.onrender.com/app	Page loads without any issue	As Expected	Pass

	/			
2	Click add job button	Add job form appears	As Expected	Pass
3	Fill the input field with "Test J_01"	Fill input fields without any issue	As Expected	Pass
4	Click save button in the add job form	Add job form closed	As Expected	Pass
5	Find newly added job in the job list	New job added	As Expected	Pass

(5). remove job

Test Case ID	J_02	**Test Case Description**	Test the remove job functionality in the job board page		
Created By	Yee	**Reviewed By**	L	**Version**	1.0
QA Tester's Log	Initial test				
Tester's Name	Yee	**Date Tested**	11/27/2024	**Test Case (Pass/Fail/Not)**	Pass

S #	Prerequisites
1	web browsers
2	signed in to https://ezjobs.onrender.com/ with test data
3	follow test case J_01 to add new job
4	

S #	Test Data
1	username: test
2	password: SFSUcsc648
3	
4	

Test Scenario Verify that user can delete a job application

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://ezjobs.onrender.com/app/	Page loads without any issue	As Expected	Pass
2	Locate Test Job and click DELETE button	Test Job is removed from job list after button clicked	As Expected	Pass
3	Refresh current page	Job list data remains same	As Expected	Pass

(6). get job

Test Case ID	J_03	**Test Case Description**	Test the fetch & display job functionality in the job board page		
Created By	Yee	**Reviewed By**	L	**Version**	1.0
QA Tester's Log	Initial Test				
Tester's Name	Yee	**Date Tested**	12/1/2024	**Test Case (Pass/Fail/Not)**	Pass

S #	Prerequisites
1	web browsers
2	signed in to https://ezjobs.onrender.com/ with test data
3	
4	

S #	Test Data
1	username: test
2	password: SFSUcsc648
3	
4	

Test Scenario Verify that job application data is fetched from the backend and displayed correctly in the frontend.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://ezjobs.onrender.com/app/	Page loads without any issue	As Expected	Pass

2	Verify that the job application data is rendered in the correct format on the page	Job application data is displayed correctly in the specified UI components	As Expected	Pass
3	Refresh current page	Job list data remains same	As Expected	Pass

(7). job charts

Test Case ID	J_C_01	**Test Case Description**	Test the fetch & display job stats charts functionality in the job board page		
Created By	Yee	**Reviewed By**	L	**Version**	1.0
QA Tester's Log	Initial test				
Tester's Name	Yee	**Date Tested**	12/1/2024	**Test Case (Pass/Fail/Not)**	Pass

S #	Prerequisites
1	web browsers
2	signed in to https://ezjobs.onrender.com/ with test data
3	
4	

S #	Test Data
1	username: test
2	password: SFSUcsc648
3	
4	

Test Scenario	Verify that job chart data is fetched from the backend and displayed correctly in the frontend.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://ezjobs.onrender.com/app/stats-heatmap	Chart in page loads without any issue	As Expected	Pass
2	Navigate to https://ezjobs.onrender.com/app/stats-heatmap	Chart in page loads without any issue	As Expected	Pass
3	Navigate to https://ezjobs.onrender.com/app/stats-donut	Chart in page loads without any issue	As Expected	Pass
4	Navigate to https://ezjobs.onrender.com/app/stats-sankey	Chart in page loads without any issue	As Expected	Pass

(8). AI resume

Test Case ID	AI_R_01	**Test Case Description**	Test the generate resume functionality in the ai resume page		
Created By	Yee	**Reviewed By**	L	**Version**	1.0
QA Tester's Log	Initial test				
Tester's Name	Yee	**Date Tested**	12/01/2024	**Test Case (Pass/Fail/Not)**	Pass

S #	Prerequisites
1	web browsers
2	signed in to https://ezjobs.onrender.com/ with test data
3	
4	

S #	Test Data
1	username: test
2	password: SFSUcsc648
3	
4	

Test Scenario	Verify that a signed-in user can successfully save resume input & delete resume data

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to https://ezjobs.onrender.com/app	Page loads without any issues	As expected	Pass

	/ai-resume			
2	Click SAVE green button	A prompt appears and notifies user that resume data saved	As expected	Pass
3	Click BUILD blue button	After 5-10 seconds, a prompt appears and notifies user that AI resume generated	As expected	Pass
4	Click VIEW RESUME grey button	Open resume page in new window and prompt user to download as a pdf	As expected	Pass
5	Click DELETE red button	A prompt appears and notifies the user that resume data has been deleted. All input field cleared, and only save button enabled	As expected	Pass

(9). add contact

Test Case ID	CT_01	**Test Case Description**	Test the functionality to add contact(s).		
Created By	L	**Reviewed By**	Yee	**Version**	1.0
QA Tester's Log	Initial Test				
Tester's Name	L	**Date Tested**	12/01/24	**Test Case (Pass/Fail/Not)**	Pass

S #	Prerequisites
1	Have a browser (firefox, chrome, etc.)
2	Signed into website: https://ezjobs.onrender.com/
3	
4	

S #	Test Data
1	Name: Bob
2	Position: HR
3	Company: BobsCompany
4	Email: Bob@email.com
5	Phone Number: 12345678910

S # cont.	Test Data Cont.
6	Contact Notes: Hello, I'm Bob

Test Scenario	Once signed onto the website, the user should be able to navigate to the Contacts tab and add contact(s) using the "Add Contact" button. The newly added contacts(s) should then show on the contact page afterwards.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	https://ezjobs.onrender.com/app/contacts	Page load without any errors	As Expected	Pass
2	Click on "Add Contact Button"	Displays a popup for form field for user input	As Expected	Pass
3	Click on "Add Contact Button" again	Popup should go away afterwards	As Expected	Pass
4	Wait until popup goes away	Now, the newly added contacts should appear at the bottom of the contacts	As Expected	Pass
5				

(10). remove contact

Test Case ID	CT_02	**Test Case Description**	Test the functionality to remove contact(s).		
Created By	L	**Reviewed By**	Yee	**Version**	1.0
QA Tester's Log	Initial Test				
Tester's Name	L	**Date Tested**	12/01/24	**Test Case (Pass/Fail/Not)**	Pass

S #	Prerequisites
1	Have a browser (firefox, chrome, etc.)
2	Signed into website: https://ezjobs.onrender.com/
3	
4	

S #	Test Data
1	Same as above for testing add contact
2	
3	
4	

Test Scenario	Once signed onto the website, the user should be able to navigate to the Contacts tab and be able to remove contacts by clicking on the "Remove Contact" button. These removed contacts should then disappear fron the contact page afterwards.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	https://ezjobs.onrender.com/app/contacts	Page load without any errors	As Expected	Pass
2	Look for a contact to delete and click on "Remove Contact"	The contact should be deleted afterwards	As Expected	Pass
3				
4				
5				

(11). get contact

Test Case ID	CT_03	**Test Case Description**	Test the functionality to get contact(s) using a search bar.		
Created By	L	**Reviewed By**	Yee	**Version**	1.0
QA Tester's Log	Initial Test				
Tester's Name	L	**Date Tested**	12/01/24	**Test Case (Pass/Fail/Not)**	Pass

S #	Prerequisites
1	Have a browser (firefox, chrome, etc.)
2	Signed into website: https://ezjobs.onrender.com/
3	
4	

S #	Test Data
1	Using the 10 mock data already in our database.
2	
3	
4	

Test Scenario	Once signed onto the website, the user should be able to navigate to the Contacts tab and be able to search for contacts with a search bar. These contacts should be able be searched by their name, company, position and email. The appropriate contact(s) should appear once searched in the search bar.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	https://ezjobs.onrender.com/app/contacts	Page load without any errors	As Expected	Pass
2	On search bar, type Jane Doe	Only the contact with this name will appear	As Expected	Pass
3	On search bar, type CEO	Only the contact with this position will appear	As Expected	Pass
4	On search bar, type Company I	Only the contact with this company will appear	As Expected	Pass

5	On search bar, type james.blue@companyj.com	Only the contact with this email will appear	As Expected	Pass

(12). AI chat

Test Case ID	AI_C_01	**Test Case Description**	Test the functionality to use the OpenAI chatbot.		
Created By	L	**Reviewed By**	Yee	**Version**	1.0
QA Tester's Log	Initial Test				
Tester's Name	L	**Date Tested**	12/01/24	**Test Case (Pass/Fail/Not)**	Pass

S #	Prerequisites
1	Have a browser (firefox, chrome, etc.)
2	Signed into website: https://ezjobs.onrender.com/
3	
4	

S #	Test Data
1	User Input: Hello
2	User Input: Can you give me some interview questions for a Cashier?
3	User Input: Can you give me more questions?
4	

Test Scenario	Once signed onto the website, the user should be able to navigate to the Interview Prep tab and be able to interact with the chatbot. The chatbot shall respond accordingly to the user's input(s).

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	https://ezjobs.onrender.com/app/ai-interview	Page load without any errors	As Expected	Pass
2	Enter "Hello" as user input	Get a response back like "Hello, how may I help you?"	As Expected	Pass
3	Enter "Can you give me some interview questions for a Cashier?" as user input	Get response back with some interview questions related to a Cashier.	As Expected	Pass
4	Enter "May I get more questions?"	Get response back with more interview questions related to Cashier.	As Expected	Pass

Coverage Analysis

A total of **12 P1 features** were tested, representing the primary functionalities of the system. Each feature passed all integration test cases, meeting the expected results without deviations.

- **Total Features Tested:** 12
- **Features Passed:** 12
- **Coverage:** 100%
- **Tested Features:** Sign In, Sign Up, Sign Out, Add Job, Remove Job, Get Job, Job Charts, AI Resume, Add Contact, Remove Contact, Get Contact, AI Chat

The test coverage provides a comprehensive evaluation of the application's core functionalities. By testing features across user authentication, job management, contact management, analytics, and AI functionalities, the following conclusions can be drawn:

- **Core Functionalities Tested:** All primary user workflows have been validated.
- **User Journey:** Each step of the user journey (e.g., signing in, managing jobs/contacts, leveraging AI features) is confirmed to function as intended.
- **System Robustness:** The system demonstrates a high degree of reliability, as no test cases failed during the integration testing phase.
- **Feature Completeness:** All key features identified in the project scope have been covered, ensuring no significant gaps in functionality.
- The **100% coverage** achieved in testing suggests the system is ready for production deployment.

2. Coding practices

I. Coding style

Our group project follows the Google TypeScript Style Guide with some customized coding style enforcement handled through **Prettier** with formatting on save and **ESLint**. The ESLint configuration uses recommended TypeScript & React settings from plugins. The Prettier configuration includes customizations such as always using parentheses for arrow functions, enabling bracket spacing, sorting imports and JSON, and maintaining specific formatting for SQL files. The configuration also sets a print width of 100, uses double quotes, includes trailing commas in ES5 contexts, and specifies a tab width of 2 spaces.

```
{
  "arrowParens": "always",
  "bracketSpacing": true,
  "embeddedSqlTags": ["SQL"],
  "importOrder": ["^[./]"],
  "importOrderSeparation": true,
  "importOrderSortSpecifiers": true,
  "keywordCase": "upper",
  "language": "postgresql",
  "plugins": [
    "prettier-plugin-sort-json",
    "@trivago/prettier-plugin-sort-imports",
    "prettier-plugin-sql",
    "prettier-plugin-embed",
    "prettier-plugin-ejs"
  ],
  "printWidth": 100,
  "semi": true,
  "singleQuote": false,
  "tabWidth": 2,
  "trailingComma": "es5"
}
```

Source files demonstration

1) P1 feature - update AI resume
   https://github.com/CSC-648-SFSU/csc648-01-fa24-Team01/blob/master/application/backend/src/controllers/ai_resume/ctrl_add_update_resume.ts
2) P1 feature - generate AI chat questions
   https://github.com/CSC-648-SFSU/csc648-01-fa24-Team01/blob/master/application/backend/src/controllers/ai_interview_prep/ctrl_generateQuestions.ts
3) P1 feature - sign in
   https://github.com/CSC-648-SFSU/csc648-01-fa24-Team01/blob/master/application/backend/src/controllers/auth/ctrl_sign_in.ts
4) P1 feature - get contact
   https://github.com/CSC-648-SFSU/csc648-01-fa24-Team01/blob/master/application/backend/src/controllers/contacts/ctrl_get_contact.ts
5) P1 feature - get job
   https://github.com/CSC-648-SFSU/csc648-01-fa24-Team01/blob/master/application/backend/src/controllers/jobs/ctrl_get_job.ts

The selected 5 P1 features source files above demonstrates adherence to the group's coding style as follows:

1) **Google TypeScript Style Guide Compliance**: The code is structured with clear function definitions, type annotations (`Request`, `Response`), and proper error handling, aligning with the principles of clean and maintainable TypeScript code.

2) **Prettier Formatting**:
    a) **Arrow Function Parentheses**: The arrow function includes parentheses around its single parameter (`req`), following the `"arrowParens": "always"` rule.
    b) **Import Order**: Imports are sorted logically and separated by blank lines, adhering to the configured `importOrder` and `importOrderSeparation`.
    c) **Consistent Spacing**: The use of spaces, brackets, and commas is consistent with the `bracketSpacing` and `trailingComma` rules.
3) **Readable SQL Integration**: While not directly visible here, any SQL-related functionality would benefit from the `prettier-plugin-sql` configuration.
4) **Error Handling and Logging**: The use of proper status codes, clear error messages, and console logging ensures clarity and debuggability in line with best practices.
5) **Tab Width and Line Length**: The code uses 2-space indentation (`tabWidth: 2`) and adheres to the maximum line length of 100 (`printWidth: 100`).

II. Document Generation

This project uses **Swagger** to generate and host API documentation directly from comments in the source code.

- API documentation link:
  https://ezjobs-server.onrender.com/api-docs/
- Comment source files:
  https://github.com/CSC-648-SFSU/csc648-01-fa24-Team01/blob/master/application/backend/src/routes/api_docs.ts

The documentation is built using the following tools:

- **swagger-jsdoc:** Parses JSDoc-style comments in the code to generate the OpenAPI specification.
- **swagger-ui-express:** Serves the generated API documentation on a dedicated endpoint for easy access.

The following API endpoints of P1 features are selected to generate documentation: `Sign In, Sign Up, Add Job, Get Job, AI Resume, Add Contact, Get Contact, AI Chat`