CSC 510 SOFTWARE ENGINEERING PROJECT REPORT

APPLICATION TRACKING SYSTEM

SOFTWARE CODE DOCUMENTATION

TEAM MEMBERS

Setu Kumar Basak (sbasak4)

Conor Thomason (cjthom24)

Keertana Vellanki (kvellan)

Muntasir Hoq (mhoq)

Matthew Sohacki (mjsohack)

INTRODUCTION

The process of applying for jobs and internships is not a cakewalk. Managing job applications is a time-consuming process. Due to the referrals and deadlines, the entire procedure can be stressful. Our application allows you to track and manage your job application process, as well as regulate it, without the use of cumbersome excel spreadsheets.

Our application keeps track of the jobs you've added to your wish list. It also keeps track of the companies you've already applied to and keeps a list of any rejections. Rather than having the user browse each company's website for potential prospects, our application allows the applicant to search for them directly using basic keywords. Any prospective work offers can then be added to the applicant's wishlist.

This report offers users a broad overview of the project, allowing individuals to understand it as open-source software and add improvements. The report also helps developers in comprehending the code and serves as a starting point for the project.

POTENTIAL USERS

1. Students who are applying for internships and full-time opportunities, probably for the first time
2. People, currently working, but interested in taking up a new job
3. People who went through a career change and are now looking for opportunities in a new area

ARCHITECTURE

The following technologies were used to complete the development, and it is recommended that the next group of developers who take on this project have these technologies installed and running before proceeding:

1. FrontEnd:
   a. React
   b. Node.Js
   c. CSS
2. Backend:
   a. Python, Flask
   b. Pytest (Test Framework)
3. Database:
   a. MongoDB

EXPLANATION:

Currently, we have four fundamental steps in our project:

1. The position for which you have applied
2. The job you want to apply for, without a referral
3. The job at which you have faced rejection, and
4. The job you're waiting for a referral.

Any details in any table can be modified at any time during the process.

CODE FUNCTIONALITIES:

Backend:

App.py: This is the file that generates the Flask server - upon calling “flask run” within this directory, this is the file that will be called.

1. def search(): HttpGet endpoint for getting search results on keyword e.g. job title, company name etc.
    a. Route: /search
    b. Input:
        i. keywords: A plain text input.
    c. Output:
        i. records: Returns a Json of searched results.

2. def get_data(): HttpGet endpoint for fetching all the saved job data.
    a. Route: /application
    b. Output:
        i. apps_json: Return a Json of all the job information.

3. def add_application(): HttpPost endpoint for saving new application.
    a. Route: /application
    b. Input:
        i. application: Job details
    c. Output:
        i. application: Return the Json of the newly saved application.

4. def update_application(): HttpPut endpoint for updating an existing application.
    a. Route: /application
    b. Input:
        i. application: Updated job details

   c. Output:
      i. application: Return the Json of the updated application.

5. def delete_application(): HttpDelete endpoint for deleting an existing application.
   a. Route: /application
   b. Input:
      i. application: Job details which will be deleted.
   c. Output:
      i. application: Return the Json of the deleted application.

TESTING

Backend:

For functional testing of the backend of our Flask application and database, we have a separate file named test_app.py in the backend directory and included pytest framework.

Test 1 (test_alive): In this test, we tested the GET method of our Flask client and tested whether the Flask application is live or not, by checking the string output of the client app.

Test 2 (test_search): In this test, we tested the search method of our application by asserting a mock test search and checking a fixed output.

Test 3 (test_get_data): In this test, we tested the GET method of the Flask client to get all the entries of the database and checked the database entries by mocking the output.

Test 4 (test_add_application): In this test, we tested the POST method of the Flask client to add a new entry to the database and checked the added data entry by mocking the output.

Test 5 (test_update_application): In this test, we tested the PUT method of the Flask client to update any entry of the database and checked the updated data entry by mocking the output.

Test 6 (test_delete_application): In this test, we tested the DELETE method of the Flask client to delete any entry from the database and checked the deleted data entry by mocking the output.

Test 7 (test_get_new_id): In this test, we tested the getting_new_id function and checked the output is returning the correct next new id for a new application added to the database by mocking the output of MongoEngine objects function.

Test 8 (test_alive_status_code): In this test, we tested the GET method of our Flask client and tested whether the Flask application is live or not, by checking the status code of the client app.

FUTURE SCOPE

1. Include deadline reminders for the application and interview.
2. Add a feature that allows users to attach these reminders to their Google calendar.
3. Incorporate notifications for upcoming deadlines.
4. Add a storage option for resumes and cover letters so they can be saved for future use.
5. Include a direct link to the company's application website when the wishlist item is clicked.
6. Include a link to the university’s career fair page.
7. Direct connection to Linkedin, allowing for the addition of job opportunities to the wishlist.
8. Improve keyword search to improve specifications such as pay range, employment location, and so on.
9. An option to maintain separate profiles for job tracking.