**Individual Test 2**

**Name: Le Tran Trong Hung Student number: 3805504**

For this test, I used a variety of technologies to build the job ATM. Firstly, for the two main techs, I chose to use Spring for the back end/server, and ReactJS for the front-end/client. These are two of the most popular frameworks for making applications on the market right now. For authentication, I used JWT, otherwise known as JSON Web Token as it is a proposed Internet standard for creating data with optional signature and/or optional encryption whose payload holds JSON that asserts some number of claims. The tokens are signed either using a private secret or a public/private key. I used PostgreSQL for storing information (act as a database), due to it being a free and open-source relational database management system emphasizing extensibility and SQL compliance. In optimizing the app, I employed the use of Redux, a predictable state container for JavaScript apps. It helps you write applications that behave consistently, run in different environments (client, server, and native), and are easy to test. As for the UI design, I implemented the online library Bootstrap 4.

I have finished adding in the CRUD services in the backend, as well as authentication using JWT. As for the front end, I successfully designed and built the needed interfaces for using the build services of the back-end (CRUD), along with the signup and login page. The UI was also designed to be somewhat presentable. Three types of account can be created in the app: Admin, Employer and Employee. Admins can view jobs, update them and delete them. Employers can post jobs, view them and update them. Employees can only view the jobs in the data base. As for the rest of the functions, I couldn’t integrate them into the app due to the time constraint. The same goes for deploying them online.

To start using the app, you will have to first go into the root folder of the back end (server) and change the following variables to fit your system:

Text

Description automatically generated

Then use the following command: **mvn spring-boot:run**

After the backend is finished setting up, connect to the newly created database ‘testdb’ and run the following scripts:

**INSERT INTO roles(name) VALUES('ROLE\_ADMIN');**

**INSERT INTO roles(name) VALUES('ROLE\_EMPLOYER');**

**INSERT INTO roles(name) VALUES('ROLE\_EMPLOYEE');**

This is because need to add some rows into **roles** table before assigning any role to User. Then you can go into the root folder of the front end (client) and run the following command in the terminal: **npm start**

You can then go to your designated localhost port and use the app to create accounts. However, every account create this way will only have the role of Employee. To create Admin and Employer account, you will have to use the signup method in programs like Postman:

Graphical user interface, text, application, email

Description automatically generated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Item | Questions | Marks | Completed and work well | Completed but not work (have some code but not able to make it work) | Skip (Not done, not writing any code) |
|  | **Backend** |  |  |  |  |
| a | (4 marks) Build a backend with necessary entities, models for **employee, employer, jobs, job categories (specialization), users**. | 4 | X |  |  |
| b | (2 marks) Build services (Service classes) to manage (CRUD, search, filter, pagination) these entities | 2 | X |  |  |
| c | (2 marks) Build REST API (Controller classes) to do CRUD on these entities with filter, search, and pagination and other required APIs that you think necessary to support the frontend. | 2 | X |  |  |
| d | **Frontend**  Build a frontend ReactJS application (web-based system) to provide functionality as described above. It should include pages to show the system on a Job ATM kiosk using a touch screen and pages for admin to moderate the system and pages for employee/employer to see it at home (not at the kios). |  | X |  |  |
|  | (3 marks) Signup/Signin. You can use a dummy sample PIN code for login if you couldn’t manage to setup sending PIN via SMS | 3 | X |  |  |
|  | (2 marks) Allow sending PIN code to users via SMS. You can use AWS Message or consider the following services: <https://www.infobip.com/>, <https://speedsms.vn/>, <https://esms.vn/> | 2 |  |  | X |
|  | (3 marks) Employer posting a job, view and edit their posted jos | 3 | X |  |  |
|  | (5 marks) Employee search, view, and apply for a job, view your applied jobs. | 5 |  | X |  |
|  | (2 marks) Allow employees to print a receipt with QR code that contains a URL to the job that they applied for. You can use this tool <https://www.qr-code-generator.com/> to generate QR code for a URL. | 2 |  |  | X |
|  | (2 marks) Admin to view and manage all employees, employer, jobs | 2 |  | X |  |
|  | (3 marks) Admin to view statistic reports: number of jobs by job category, number of job posted each day (start date - end date), number of jobs applied per days (start date - end date) | 3 |  |  | X |
|  | **Kafka/redis** |  |  |  |  |
| e | (3 marks) Use kafka to receive and process all messages for add new and update, except delete. | 3 |  |  | X |
| f | (2 marks) Use redis for all get requests. | 2 |  |  | X |
|  | **Deployment and readme** |  |  |  |  |
| g | (4 marks) Deploy the full system to cloud services and provide URL for testing (include it in the readme). | 4 |  |  | X |
| h | (3 marks) Write a short report (readme.doc, docx, pdf) 3 pages max to explain clearly your approach, justify architecture and design you have chosen, what functionalities you have completed, what you have not completed, any innovation in your design. | 3 | X |  |  |