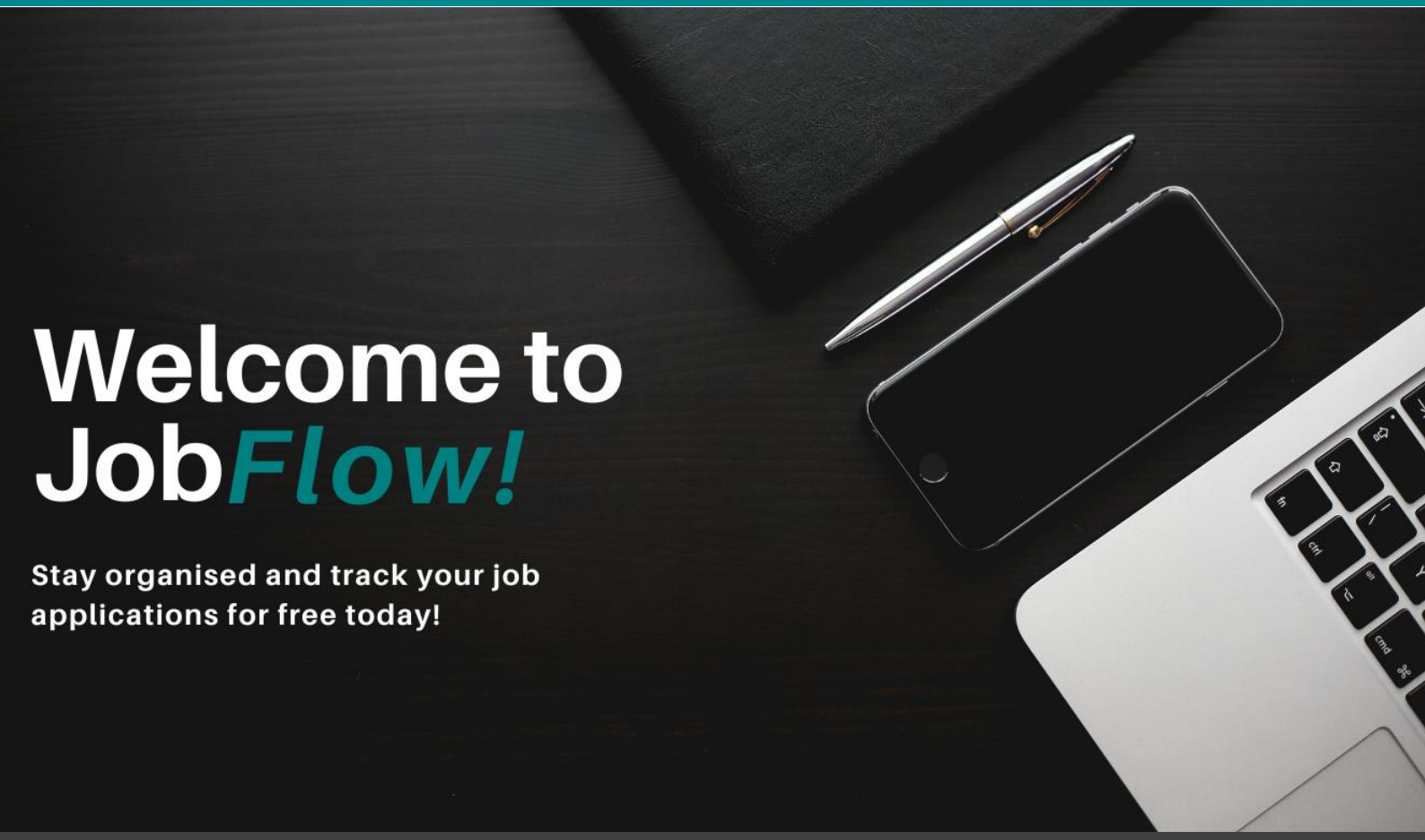
May 2023 | CFG Fullstack-2 Cohort, Group #1



Job Applications Tracker Project Report

Prepared by:

- Joy Omodiale
- Irina Seveliova
- Efuah Faler
- Olivia Onu
- Katarzyna Kaczmarek

INTRODUCTION

JobFlow is an innovative web application that aims to revolutionize the job search process by providing a comprehensive and user-friendly platform for job seekers. The project was undertaken with the primary objective of simplifying the job application process and helping individuals stay organized and efficient throughout their job search journey.

The main aim of JobFlow is to streamline the application tracking process, allowing users to manage and monitor their job applications from start to finish. By centralizing all applications in one place, JobFlow eliminates the hassle of juggling multiple platforms and provides a convenient solution to track the progress of each application.

BACKGROUND

JobFlow operates based on a user-friendly interface that allows job seekers to easily enter and manage their applications. The system incorporates a range of features to ensure a smooth and efficient job search process. Users can create a profile, add job applications, and track the progress of each application from submission to interview and final decision.

Additionally, JobFlow allows users to add notes and comments to each application, enabling them to keep track of their interactions, interview details, and any additional information relevant to the application process.

To ensure an intuitive and user-friendly experience, our team locked in specific requirements for the system. These requirements include a simple and intuitive user interface, efficient data storage and retrieval, seamless integration with popular job search platforms via the JSearch API, and robust security measures to protect user data and privacy using Firebase authentication tool.

By implementing these requirements, JobFlow aims to provide job seekers with a reliable and efficient tool to manage their job applications.

SPECIFICATION AND DESIGN

To ensure the successful development and implementation of JobFlow, a comprehensive set of technical and non-technical requirements were identified and addressed.

Technical Requirements

Frontend Development: The application required a responsive and user-friendly frontend interface. The team leveraged technologies such as React, JavaScript and CSS to create an intuitive and visually appealing user interface.

Backend Development: The backend development focused on utilizing Node.js, an efficient and scalable runtime environment, to handle server-side logic and API integration.

Database Management: The team implemented Sequelize, to manage and interact with the MySQL database. This ensured efficient data storage, retrieval, and management within the application.

API Integration: We integrated the JSearch API, enabling users to search for job opportunities directly within the application. The team worked on effectively connecting the API, allowing users to access real-time job data seamlessly.

Security Measures: Our Team prioritized the protection of user data and privacy by implementing robust security measures. To achieve this, secure authentication and authorization protocols were implemented using Firebase. By adhering to industry best practices, JobFlow ensured the confidentiality and integrity of user information, safeguarding it from unauthorized access or misuse.

Non-Technical Requirements

User-Friendly Interface: The platform aimed to provide a user-friendly and intuitive experience for job seekers. The design and layout were carefully crafted to make navigation and usage straightforward, regardless of the user's technical expertise. We aligned ourselves as closely to Nielsen's heuristic principles when designing the user interface. And we chose a professional, yet engaging colour palette to appeal to our target users.

Ease of Application Management: We aimed to simplify the job application process by allowing users to easily add, track, and manage their applications. The interface and features were designed to enable effortless application management and organization.

Seamless Job Search Integration: The integration of the JSearch API allowed users to search for job opportunities without leaving the JobFlow application. This streamlined the job search process, providing a seamless experience for users.

Performance and Responsiveness: Our team aimed to deliver a highly performant application. Pages and functionalities were optimized to ensure minimal loading times and smooth interactions, enhancing the overall user experience.

Scalability: The application was designed with scalability in mind to accommodate potential growth and increased user demand. The architecture and technology choices allowed for efficient scaling of the application as the user base expanded.

The combination of a robust tech stack, intuitive user interface, and seamless API integration contributed to an efficient and user-friendly platform for people searching for new employment opportunities.

Design and Architecture

The JobFlow application follows a client-server architecture with a frontend, backend, and external API integrations.

User/Client: These are the users of the application, typically accessing it through the web browser.

Frontend (React): The frontend layer of the application built using the React library. It handles user interactions and sends HTTP requests to the APIs.

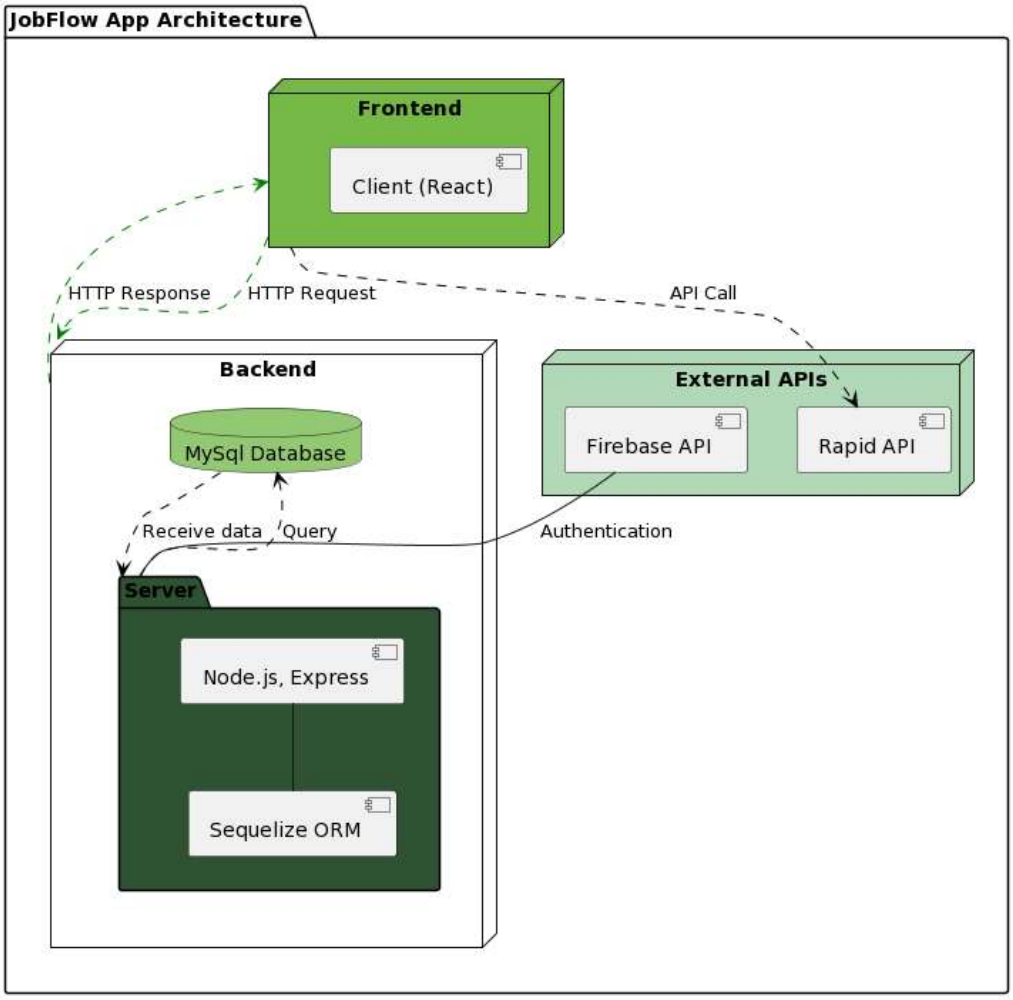
Backend (Node.js, Express): The backend layer of the application built using Node.js and Express. It receives and processes HTTP requests from the frontend and communicates with Firebase and the database.

MySQL Database/Sequelize ORM/ Cloud hosting: The relational database management system used, coupled with Sequelize as the Object-Relational Mapping (ORM) tool, to store and retrieve job data and user information. This combination enabled efficient data management, seamless mapping, and simplified database operations. For user convenience and to ensure the data integrity, main database is hosted on a cloud service provided by Clever Cloud. A local database configuration is also available, which is useful while running backend unit tests as they reset the database and remove the existing data.

External APIs: Integration with external APIs enhances the functionality of the JobFlow application.

- Rapid API: An external API (JSearch) used to retrieve job listings based on user search criteria.
- Firebase API: Integration with Firebase for user management and authentication features.

The flow of the architecture starts with the User Client, which interacts with the Frontend. The Frontend sends HTTP requests to the Backend APIs, which handle the user requests, interact with the MySQL Database for data storage and retrieval, and communicate with the External APIs for additional functionality.



IMPLEMENTATION AND EXECUTION

SWOT Analysis

During one of our earliest meetings, we performed a SWOT analysis to identify each team members individual strengths and we discussed if people had any preferences over how tasks were assigned.

In doing so, we were able to distribute the team's responsibilities as follows:

- **Olivia** took charge of critical frontend development tasks, including designing components, color scheme, layouts, and search functionality integration. She also improved the user experience by designing navigation, counter, and footer sections. Additionally, Olivia crafted compelling copy for the landing, about, and how-it-works pages.
- **Efuah** was integral to key parts of frontend and backend development. On the frontend, she built the Jobs Table, Modal Form, and Contact Us form, connecting them to the API. On the backend, Efuah set up Sequelize connections, added models and routes, integrated Firebase Middleware, and implemented unit testing for routes.
- **Joy** led key parts of frontend development, creating the signup page and job-details page. She contributed to project ideation, established file structure, addressed bugs, and created test cases. Joy also set up the server-side infrastructure, preparing it for future API calls.

- **Kasia** was pivotal in implementing Firebase authentication, assisted with backend development, and integrated Sequelize for MySQL tables. She developed code for user communication management and implemented frontend unit tests for improved reliability.
- **Irina** played a key role throughout the project, creating prototypes, wireframes, and finalizing the design. She developed the Login page, Registration process from backend side and set up remote database configuration. Irina also prepared unit tests, collaborated with the team, and assisted with code reviews and issue resolution.

In summary, each team member played a vital role in various aspects of the JobFlow application, showcasing their expertise and contributing to its successful completion.

Tools and Libraries

To develop JobFlow we utilized the following tools and libraries:

- Draw.io;
- React, CSS, FontAwesome, react-router-dom;
- Node.js, Firebase, Axios, Express;
- MySQL, Sequelize, Clever Cloud hosting;
- JSearch API;
- Postman, Git, GitHub, VSCode, MySQL Workbench.

In addition, we used a variety of tools to effectively communicate with one another, making the development process more efficient:

- Slack;
- Jira;
- Google Docs;
- Google Meet.

Implementation Process & Challenges

During the implementation of our backend, we initially encountered problems using Sequelize to create tables in the MySQL database. The Sequelize documentation was quite ambiguous in parts and so we had to regroup and consider how to approach the problem. During a team meeting, we shared screens and tried to troubleshoot the issues. And after some research, we later agreed to try a different approach at connecting to the database and setting up tables which proved to be successful.

Another challenge we faced in the backend was establishing a connection to Firebase, which we had used for user authentication. Since authentication was such an integral aspect of security, it was very important for us to ensure that this was fully integrated. After extensive reading through the Firebase documentation, it was made clearer to us on how to approach this. We set up Firebase middleware in the backend and we were able to achieve our desired results.

Agile Development

Our iterative approach was the cornerstone of our development process, allowing us to continuously refine our work. Jira served as our project management tool, enabling us to plan and manage our sprints effectively.

Refactoring: As we progressed through the iterations, we actively refactored our codebase to optimize performance, enhance readability, and adhere to coding best practices.

Code Review: To ensure code integrity and adherence to best practices, we implemented a balanced code review process. Leveraging **Git** and **GitHub**, we followed a structured workflow that involved creating feature branches, committing code regularly, and utilizing pull requests.

Feedback: Real-time communication was vital to our project, and we relied on **Slack** to facilitate it. Slack became our virtual workspace, allowing us to exchange ideas and address any project-related queries instantly. In addition to that

we had weekly meetings in **Google Meet** to track the progress of the development, share the insights and agree on next steps.

These elements aided the team to iterate on our work, maintain code quality, and collaborate effectively, ultimately resulting in the successful delivery of our project.

TESTING AND EVALUATION

Testing Strategy

Our testing strategy combined automated testing by creating **unit tests** and **manual testing** by carefully going through the workflows step by step, the same way a customer would while using our application.

For our project, we decided to use unit tests that involved isolating parts of the code and testing them independently to ensure they function as intended. The primary tool used to facilitate testing was Jest, a popular JavaScript testing framework.

In the frontend, our testing involved checking that the components of our website were displayed properly and that the important features worked well when users interacted with them, such as clicking buttons or entering information in the form. This helped ensure that users have a smooth and satisfying experience on our website.

In the backend, we decided to write a test file per Express router and test the routes as broadly as possible. This meant that each test run would reset the database state to ensure tests were run consistently. For the Jobs routes, we created a mock middleware to set up some properties on the request. This was to imitate the Firebase middleware used in the real application since Firebase as a third-party service could not be unit-tested predictably.

As part of manual testing, we executed hundreds of tests while developing the application to ensure proper user experience, used Postman and MySQL Workbench regularly to confirm that routes in our backend are working as expected and that the database is updated as intended.

Functional and User Testing

Functional and user testing covered the main scenarios for the end user:

1. Open home page, navigate to other pages.
2. On Sign Up page, fill in the fields and create a new account.
3. On Log In page, provide existing username and password and move to My Applications page.
4. On My Applications page, create a new application by clicking “Add a Job” button, fill in the fields, save and make sure the created application is visible in the list of applications.
5. Update/delete the existing application by using Edit and Delete buttons.
6. Click on existing application to open a detailed view.
7. Log out and navigate to Log In page.
8. On Contact Us page, fill in the form and receive a confirmation the message was sent.
9. On Search for Jobs page, input a job title & a location and get a list of jobs in result.

System Limitations

As we were limited in time and recourses to deliver a working solution, our application includes certain constraints.

From a user perspective:

- Application is compatible with Windows 10 and higher, MacOS, web browsers Chrome, Edge, Safari.
- Mobile version of the application has not been tested and is out of scope for this project.

- While using Search for Jobs tool, it may take some time to receive the result of the search. The performance of this feature depends on the external service provider, and we don't have any influence on that.

From a developer perspective:

- Clever Cloud database hosting service allows up to 5 connections to the database at a time.
- To use the local database, it's required to configure it first on the local machine, more details are in the README file.
- Backend unit tests reset the database and delete the data, it's recommended to switch from remote to a local db connection before running unit tests.
- JSearch API is an external service that has limited number of queries included to a free account. If you don't receive any results when performing a search, please contact one of the team members and we'll provide you a fresh API key with 150 queries available.

CONCLUSION

Ultimately, with JobFlow, we have delivered what we have set out to. We have made the process of tracking job applications for job seekers much easier and more streamlined. By also integrating a job search page into our website, we have built an application that our customers could depend on throughout the entire lifecycle of their search process.

We are very proud of what we have achieved with JobFlow so far. It is certainly something that we would all personally use and believe is an excellent minimum viable product that could be rolled out to users in the very near future.