**Final Report**

Group 4: JobPort

Members: 1. Krunal Priyadarshi.

2. Smit Hirpara.

3. Pratikkumar Bhingaradiya.

Live URL: <https://job-port-eight.vercel.app/login>

**Purpose of the website**

* This job portal is designed to connect job seekers with employers efficiently. Job seekers can find jobs that match their skills and experience, while employers can find candidates for their job openings. It aims to streamline the recruitment process for both parties.

**Problem trying to solve**

* Traditional job search methods can be time-consuming and inefficient for both job seekers and employers. This portal provides a centralized platform where users can search and apply for jobs easily, while employers can manage applications and reach a larger pool of candidates.

**Learnings during development**

* During development, we learned a lot about creating user-friendly interfaces and efficient backend systems. I also improved my understanding of databases, API integrations, and best practices in web development.

**Problems faced and solutions**

Front-End Developer (Pratik): **Responsive Design Challenges**

* **Problem**: I struggled with ensuring that the design and layout of the recruiter dashboard and job listing pages were responsive across different devices and screen sizes. The challenge was managing complex layouts and maintaining consistency in styling.
* **Solution**: I resolved this by using Bootstrap's responsive grid system effectively, and making use of media queries to adjust styles for different screen sizes. I also conducted extensive testing on various devices and browsers to ensure the layout appeared as expected.

Back-End Developer (Smit and Krunal): **Database Query Optimization**

* **Problem**: We confronted challenges in optimizing database queries within the context of GraphQL to effectively handle substantial amounts of job and application data. The performance of certain GraphQL queries was suboptimal due to inefficient database operations.
* **Solution**: To address this issue, we prioritized optimizing database queries by employing appropriate indexing and minimizing unnecessary data retrieval, all within the GraphQL schema. Additionally, we introduced caching mechanisms to alleviate the database load, thereby enhancing the overall performance of GraphQL API endpoints.

All Team Member**: Integration Testing and Debugging**

* **Problem**: Difficulties with ensuring smooth integration between the front-end and back-end components, leading to unexpected bugs and issues in the application.
* **Solution**: Focused on thorough integration testing to identify and fix issues early. Debugged problems by examining network requests and console logs. Also, used stack overflow (**question-and-answer website for computer programmers**) to pinpoint and resolve any inconsistencies between the front-end and back-end.

**What to improve**

* We would like to improve the speed and performance of the website by further optimizing the backend and frontend code. Additionally, adding more advanced filtering and search options for job seekers would enhance the user experience.

**Project's competitiveness**

1. **User-Friendly Interface**: The project uses Bootstrap for responsive design, ensuring a consistent and user-friendly interface across different devices and screen sizes. This enhances user experience, making it easier for job seekers and recruiters to navigate and use the platform effectively.
2. **Simplified Recruiter Dashboard**: The recruiter dashboard is designed to be straightforward and efficient, allowing recruiters to manage job posts, applicants, and insights quickly. This simplicity appeals to users who want to perform tasks without any hassle.
3. **Clear Calls to Action**: The code include clearly defined actions for recruiters, such as creating a new job, managing applicants, and viewing insights. This focus on guiding the user to key actions helps streamline the user experience.
4. **Well-Structured Components**: The project demonstrates good separation of concerns with components like **Job Seeker** and **Job Posts**, which handle specific functionalities. This modularity contributes to maintainability and ease of updates, giving the project an edge in the long run.
5. **GraphQL Usage**: The use of GraphQL for data fetching provides flexibility in querying data and reduces over-fetching. This approach allows the project to perform more efficiently and provide a better user experience.
6. **Future Extensibility**: The structure and approach in the codebase lay a solid foundation for future enhancements and features. By adhering to modern development practices and using established technologies like React and Bootstrap, the project is set up for potential future growth and additions.