Easy-Hiring

Team Member: Yongrui Lin, Xiaojing Ji

Motivation

College students who are actively looking for internships or full-time positions may be familiar with on-campus career fairs and joke that they need to be mentally prepared for handling the massive recruiting information flood. Center for Career Discovery and Development at Georgia Institute of Technology has announced that there are over 400 companies and 5000 students attending a two-day career fair event annually[1]. Considering a large number of students and recruiters attend the event, it is an exhausting experience for all participants because of extremely long waiting queues and a massive amount of information. Applicants may stay in lines for hours with their resume hardcopies to reach out companies even without knowing whether themselves are a good fit for the positions. At the same time, going through thousands of resumes within a few hours to select best candidates is an overwhelming experience for recruiters as well. We realize that job fair events may need a more dynamic solution for supporting applicants and recruiters to filter out irrelevant options quickly. Time limitations become a considerable issue for improving the event efficiency. However, there barely has services specifically designed for the career fair event, which is considered as one major component for recruiting. Hence, it definitely has room for further improvements. Here we attempt to develop a mobile application which assists job hunters to search for matched positions on the career fairs, as well as helping companies select qualified candidates based on resume contents and required skills.

Related Work and Tools

- One key feature of recruiting services is providing a platform for information integration and job-related social networking. Several popular platforms in this field have successfully integrated company information and user profiles at some level. As an employment-related search engine, Indeed[2] lists jobs that meet searching conditions. LinkedIn can compare profile contents to some posted positions and acknowledge users with recommended jobs. Glassdoor collects anonymous company reviews and offers online job application access[3]. Job-related services provided by those websites significantly improve the efficiency of job hunting experience as users can utilize integrated data to find best matches.

- **React Native**: we initially decided to use React Native as the platform for this mobile application. The actual application is built on an updated version called Create React Native. Create React Native removes most mobile OS dependency packages existed in previous React Native versions and can be ejected to traditional React Native. React Native was originally released in 2015 and has started gaining popularity ever since then[4]. It shares the same design as React and supports UI building blocks for iOS and Android deployment. JavaScript package manager npm keeps many public API packages to accommodate React Native, which simplifies implementation of certain features on this application.

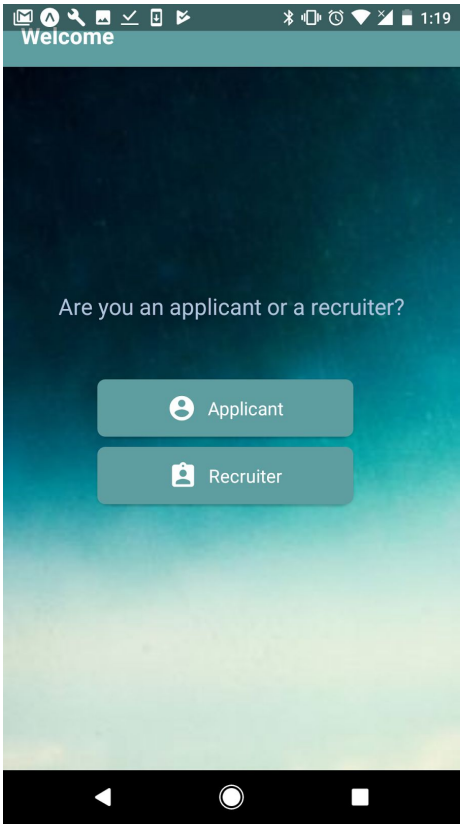
- **Expo**: a set of tools to support tools, libraries and services for building React Native apps across Android and iOS devices[5]. It enables debugging on both Emulators or

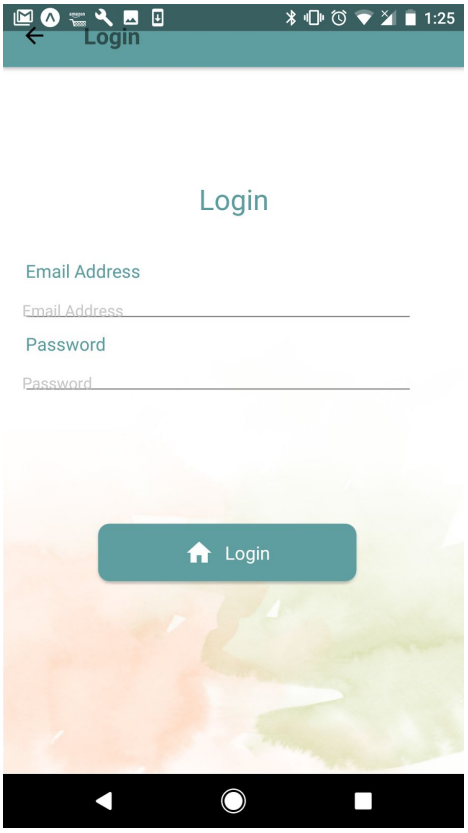
actual devices. Testing on a mobile device simply requires scanning the generated QR code of a particular application from Expo mobile app.

- **Firebase Real-time Database**: a NoSQL cloud database sponsored by Google[6]. It synchronizes data in real-time and stores data in JSON format. Firebase displays database data in a hierarchical view so that developers understand internal data structure in a more straightforward approach. The data synchronization feature ensures instantaneous updates in case clients have made any changes. Large files like PDF and images can go into Firebase storage section.

Platform and Implementation Detail

- **Application Overview:** We used react-native to build a cross-platform mobile app which supports deployment on both Android and iOS operating system. This single application divides into two modes: applicant and recruiter/company. The homepage requests a user to select as an applicant or a recruiter and navigates to different views which are corresponding to different user modes. After the authentication, a recruiter can scan QR code, post new job descriptions and download resume of candidates for further consideration. From an applicant's view, the profile contains QR code for the resume, professional skills, and a list of matched jobs in the system pool. Noticing the career fair event usually has many companies, we also provide a map to display/search locations of attended companies.

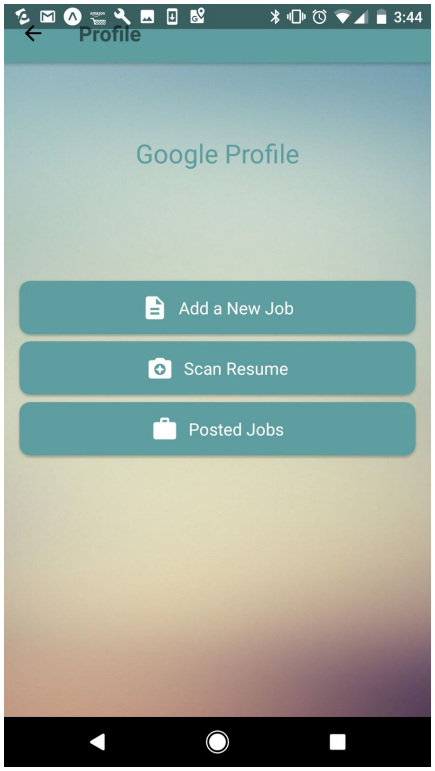




- **System Architecture**:

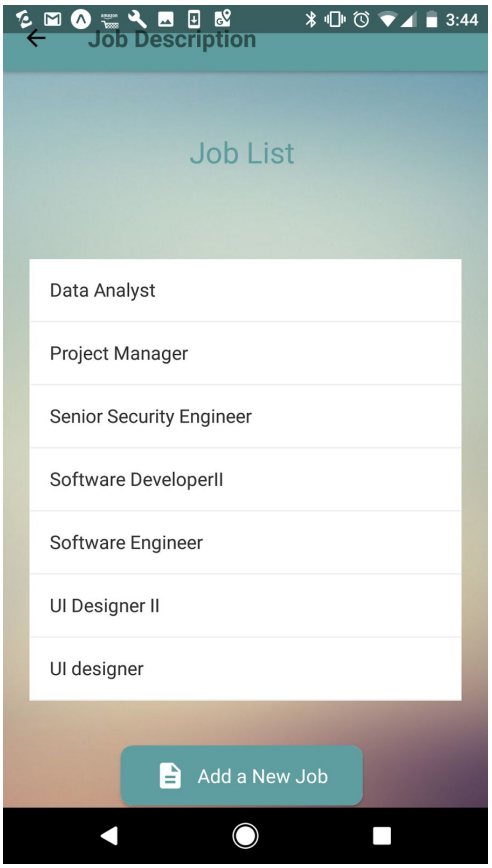
We select React Native as the platform to integrate with Firebase database. This cloud-hosted database stores user profiles, PDF documents, company information, career fair map, internal relations among companies, jobs and candidates, as well as other relevant data.When launching the application, it initializes a connection between the client and the cloud database.

- **Recruiter:** The following figure demonstrates the main page from a recruiter's perspective. To keep the application concise and clean, it only contains three major features: post a new position, QR code reader to grab resume, and checkout posted jobs.



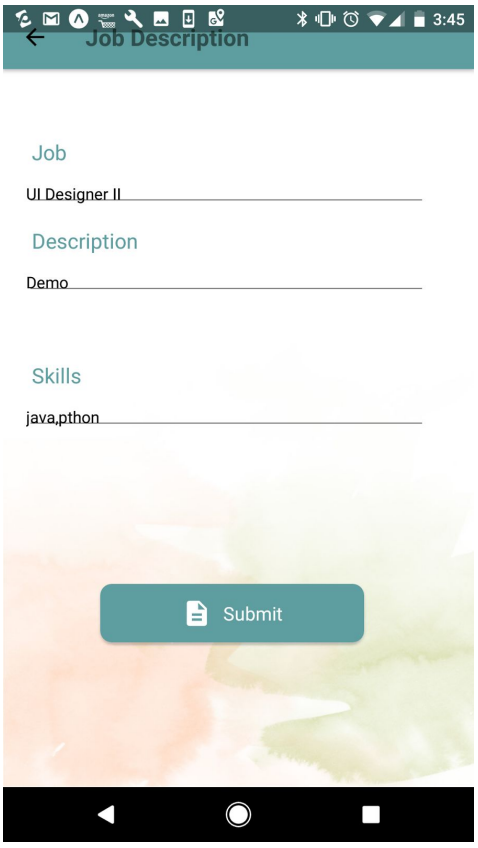
Scan Resume: We attempt to eliminate hard copies of resume on career fair and realize the importance of face-to-face communications on the event, so we enable the feature of scanning QR code to receive resume. In a sense, the recruiter can open the camera to scan QR code attached to an applicant's profile. Implementing QR code scanning is challenging as the application is built on Create React Native, which does not support linking some customized react native packages including react-native-camera. To complete this task, we have done some hacky work to get around with package linking issues. It first requests permission for accessing the device camera. After capturing the given QR code, it asks the user for allowing redirection to the decoded link. Behaviors of accessing resume files stored on Firebase storage

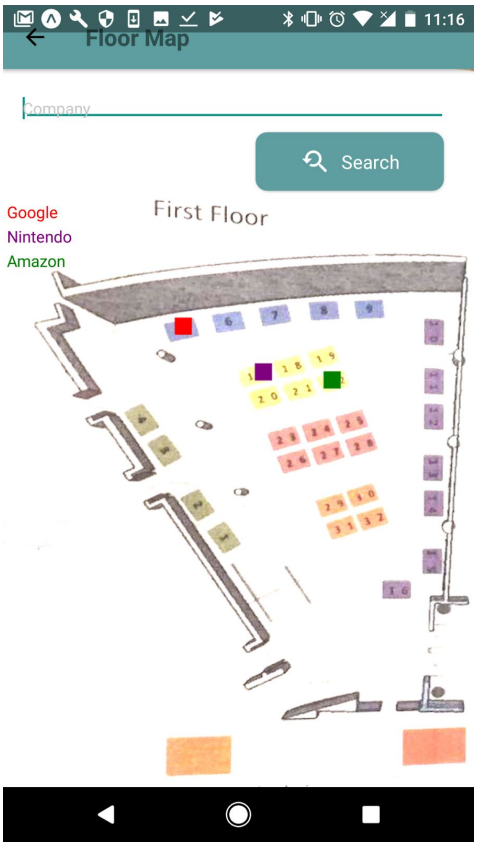
depend on devices. Some Android devices can directly load PDF documents while others ask to download files before hands.



Check Out Applied Candidates: Recruiters definitely need to collect candidates professional information from the data pool. Our implementation augments applicant information to each job entry they have applied so that recruiting data is more organized. Recruiters view candidate resume grouped by posted job positions. For example, they can check out candidate resume files just for a specific software engineer position. Keep in mind that current implementation completes filtering on the applicant side, so a hidden assumption is all candidates for a certain job showed up to the recruiters should at least pass the filtering condition.

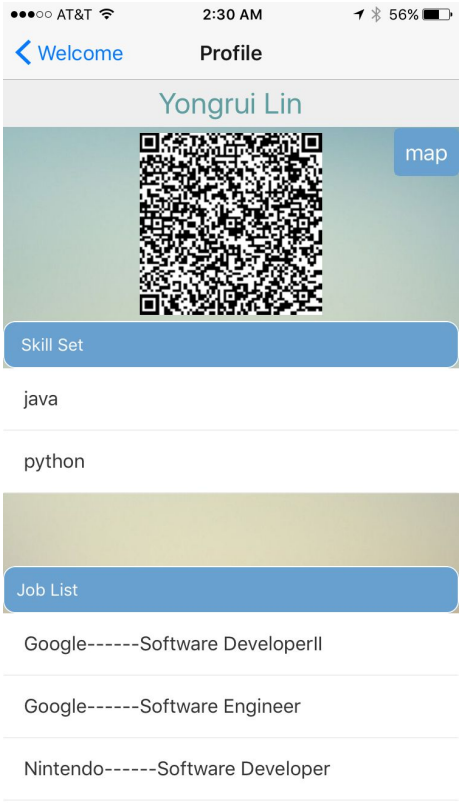
Add a New Job: A company is able to post new open positions by filling out job titles, detailed descriptions, and required skills. Once a valid post has been submitted, the app triggers updates and forwards entered data to Firebase database. It inserts this new job under that specific company data collection in a hierarchical manner. Because data is stored in JSON format, traversing through the data hierarchy is manageable by picking correct child node via the corresponding name on each level. However, this implicitly brings another interesting challenge to the implementation. In order to insert new entry under correct company profile, the application needs to parse related information from previous pages to the current page for retrieving data. Such action requires parameters parsing among pages in the navigation. We apply navigation package to record and parse data from one page to the other. There is an alternative solution as data export, but this is not as dynamic as navigation and won't be sensitive to any value updates. The following figure shows the view of posting a new job.





- **Applicant:** The applicant's mode has three individual parts. The first part is a QR code which links to the current user's resume. We use "react-native-qrcode" library to generate the QR code. It only needs to input the URL link to implement this feature. The second part is to display the skill set the current applicant. The data of the applicant has already been stored on the Firebase real-time database. What we did is to call Firebase's API to get the structured data. Then display the skill we got on the screen. The third part is showing the jobs that match to the applicant. We also used Firebase's API to get all the job's skills, then compared these skills with the skill we got in the second part. If we find that the skill of the applicant match at least one required skill of this job, we display this job on the screen. Additionally, each entry of the job list is a "TouchableHighlight" type, which can be pressed by the users. The user can press this item to apply for the specific job. Then the backend database will update, adding the current user to the applicants list under that job.

  The following figure demonstrates an applicant's view:



Scope and Limitation

Overall, we completed the performs as outlined in the proposal: On applicants side, it generates the QR code which links to the resumes, provides the job list that matches applicant's skills; On recruiter side, it has function that can scan the QR code to view the applicant's resumes, post jobs with description and required skills. However, there are some minor differences that should be noted.

The major difference is that we did not implement the filter on the recruiter side. Because, once we added the filter on applicant side, we notice that the applicant can only submit their application for the job that the requirements are satisfied. There is no need to create a filter for recruiters.

Additionally, due to the time constraint, we gave up to build the function of uploading pdf format resume on the applicant side. After talked to professor Pu, this function can be replaced by accessing the Linkedin profile. LinkedIn also enables the feature of the professional network.

Finally, our proposal did not state specifically to use what criteria to match the applicant with the job. In our implementation, we use the "skill" as the filter. This may be too one-sided for the recruiters to pick their desired employees.

Conclusion

We used React-Native and Firebase real-time database as backend building a mobile application called Easy-Hiring. Using it, a job applicant can upload their resumes, get a QR code shown on the screen of the device and get the job that matches to skills. The

recruiter can use it to post jobs, get applicant's resume by scanning QR code, as well as review candidates electronic resume if needed. We have deployed this mobile application on android devices(Pixel and Nextbit) and performed testing on an iPhone 6 device as well. Our current implementation leaves lots of room for future work on network connection and job reviews.

Future Work

- **Integrate with Linkedin:**
  Use Linkedin api as the authentication to login to the app. Access to profile data on Linkedin so that recruiter can connect to the applicant right away after they chat to each other on the career fair. On one hand, the applicant no longer need to upload their resume again to our app. On the other hand, it is easier for both applicant and recruiter to expand their networking.
- **Integrate with Glassdoor:**
  The current application lacks of information of the company itself. The applicant would like to be exposed more information of the position they apply and more information about the company, not only their company's culture but also the interview process, the average salary and the reviews by the employees. We can achieve this by using the Glassdoor's data.
- **Networking:**
  Take a long view, we can build our app as not just a job hunting tool. We can build it with a career based social media. Different from Linkedin, it is more focus on job interviewing, letting users can get together to prepare for a specific interview if they apply to the same position.

Reference

1. "All Majors Career Fair." *C2D2 | Georgia Institute of Technology | Atlanta, GA*, Georgia Institute of Technology, career.gatech.edu/all-majors-career-fair.
2. Job Search | Indeed." Jobs, www.indeed.com/.
3. "Glassdoor Job Search | Find the Job That Fits Your Life." Glassdoor, www.glassdoor.com/index.htm.
4. "A Framework for Building Native Apps Using React." React Native, Facebook, facebook.github.io/react-native/.
5. "Quick Start." Quick Start | Expo latest documentation, Expo, docs.expo.io/versions/latest/index.html.
6. "Firebase Realtime Database | Firebase." Google, Google, firebase.google.com/docs/database/.