Requirement Specification Document

1. **Introduction**
   1. **Background**: "Referral Finder" is a web application designed to facilitate the process of students in the campus connecting to industry referrals. In essence, the platform allows alumni to create posts containing referral information, which students from the school can subsequently apply to. The aim of this product is to establish a seamless link between students seeking professional referrals and alumni who can provide valuable industry connections.
   2. **Version**: 1.0.0
   3. **Notation**:
      1. **Student**: for those who use student email to register. They cannot post new jobs but only apply for existing jobs.
      2. **Alumni**: for those who use company email to register. They cannot apply for jobs but only post new jobs and review applications.
      3. **User**: Both Student and Alumni. Anyone who registers for this web application.
      4. **Administrator: use Django admin class**
2. **System Design**:
   1. **Overall**: The design of "Referral Finder" involves creating a user-friendly and efficient platform that bridges the gap between students and industry referrals. The system comprises the following key components and functionalities:
      1. **User Registration and Authentication**
         1. Student must use school email (.edu) to register.
         2. Alumni must use to company email to register.
         3. Administrator must **review** the job posts:
            1. ‘under-review’
            2. ‘pass’
            3. ‘fail’
      2. **Job Post Management**
         1. Alumni can submit a job post.
         2. Then the post will be displayed in the main page.
         3. Job Post has two statuses:
            1. Accept: open for application
            2. Closed: no longer open for applications
      3. **Job Search and Filtering** 
         1. User can search for jobs by entering key words.
         2. User can choose to display the posts by time & **number of applications**.
      4. **Application Management**
         1. **Applying:**
            1. Student can apply for the job.
            2. Student can change the application if it is not being reviewed and within 1 day.
            3. Student can view all jobs that they have applied. The status of the job will be shown: In progress, Selected, Not-Moving-Forward.
         2. **Reviewing:** 
            1. Alumni can review the applications.
            2. Alumni will update the status of the applications: either Selected, Not-Moving-Forward. (Default: In progress)
      5. **Favorite Job Management**
         1. Student can mark some jobs as favorite.
         2. Student can delete those jobs from the favorite category.
      6. **Job Post Status Management**
         1. Job Post can have two statuses:
            1. Accept application. (Student can apply for this job and the job is visible to everyone)
            2. Closed. (This job is only visible to the alumni who posts the job. Then the alumni can review this job)
      7. **Push Notification:**
         1. Use FastAPI websockets to enable in-time push notification feature.
         2. Notifications pushed to every connected user. Frontend filtering & rendering
      8. **Async Email Tasks:**
         1. Whenever a new user registered/ an application submitted / application status updated / application edited/, an email will be sent to the corresponding user asynchronously via RabbitMQ & Celery.

Functionality (System Functional Architecture Diagram):  
aA diagram of a referral

Description automatically generated

* 1. Running Environment
     1. Frontend: React / MUI
     2. Backend: Django / FastAPI
     3. Database: PostgreSQL & Redis

1. **System Feature**: System features describe the main functions and characteristics provided by the "Referral Finder" application. Here are the descriptions of the system features:
   1. User Registration and Authentication
      1. Students must register using their school email (.edu) to ensure the authenticity and accuracy of user identity.
      2. Student also need to provide their major, location, school, year in school, sponsorship-situation, and contact-info in the registration.
      3. Alumni must register using their company email to verify their affiliation.
   2. Job Posts Management
      1. Alumni can submit job posts to share industry referrals with students. Each post requires Job Name, company, location, targeted-major, job description, job requirement, sponsorship info, and Job Question (individualized).
      2. Submitted job posts will be displayed on the main page for students to browse and apply.
   3. Job Search and Filtering
      1. Users can search for specific job posts by entering keywords. The result displayed will be the specific jobs satisfying the requirement.
      2. There is also a **button** on frontend that also users to choose to sort job posts by time posted. By default, The Job posted is placed by alphabetically order of the job name.
   4. Student Application Management
      1. Students can apply for job posts by submitting their resume (pdf), LinkedIn URL, and responses to the Job Question.
      2. Students can withdraw their applications if not yet reviewed and within one day.
   5. Favorite Job Management
      1. Students can mark specific job posts as “favorite”. The favorite jobs will be displayed in the ‘xxx/favorite’ page.
      2. Students can remove job posts from their favorites list if they are no longer interested.

These system features collectively constitute the core functionalities of the "Referral Finder" application. Through these features, students can browse and apply for job referrals provided by alumni, while alumni can post job openings and review applications, facilitating effective interaction between alumni and students in terms of professional referrals. Additionally, the system offers user authentication, search, sorting, and favoriting functionalities to enhance user experience and efficiency.

1. Specific User Case for two types of Users
   1. Student User Flow:
      1. **Sign up** with School Email. The system requires unique school email. Otherwise, prompts user to re-enter this field.
      2. **Login** with School Email
      3. **Modify** the basic information (year in school, location, contact-info, major) in the personal profile page.
      4. **Apply for a job** by uploading the Resume and LinkedIn URL.
      5. **Recall an application:**  if the application is not being reviewed/ within a day, it can be recalled. (And allows re-submission later.)
      6. **Marked a job as favorite**: by clicking the button “Save to favorite” in the frontend.
      7. **Browse existing jobs**: see the list of jobs available jobs displayed in the main page.
      8. **Search for available jobs:** type the keyword that user want to search for (i.e software engineer), then the system will display the matched roles.
      9. **Filter the available jobs according to time:** click button ‘Filter by Time’ to display the job in ascending/descending time order.
   2. Alumni User Flow:
      1. **Sign** **up** with Company Email. The system requires unique company email. Otherwise, prompts user to re-enter this field.
      2. **Login** with Company Email with password.
      3. **Modify** the basic information (company, location, contact-info) in the personal profile page.
      4. **Browse existing jobs**: see the list of jobs available jobs displayed in the main page. (Note: they can only view jobs but not be able to apply)
      5. **Post a new job**: enter Job Name, company, location, targeted-major, job description, job requirement, sponsorship info, and Job Question (individualized) to post a new job.
      6. **View/Accept Application**: view the application in the frontend. If the role matched, then alumni can click on ‘Selected.’ This automatically sends a user an email. Otherwise, alumni can click on “Not-Moving-Forward” to reject the rest of the application
   3. Unregistered Users:
      1. They cannot login to the system. But only see the landing page
2. Non-functionality Requirement:
   1. Performance Requirement:
      1. Response time: as fast as possible -- 1s – 2s for each request.
      2. Allow 100 – 200 users to use simultaneously.
   2. Safety Requirement:
      1. According to different roles, the system assigns different permission to them.
      2. Only the registers users can enter the system.
      3. Student could only apply for the job, while alumni could only post jobs.