SWEseek - Job Searching and Tracking

CPSC 471 Final Report

GROUP 45

Nicholas Kyle Knapton - 30098353
Zeeshan Salim Chougle - 30094417
Mohamed Numan - 30086940

Abstract

SWEseek is a job search platform targeting job seekers and employers in the tech field. Searching for jobs in the tech field poses interesting challenges in regards to applications and keeping track of jobs. SWEseek provides users the ability to apply for jobs, track the job and status of application, explore salaries reported by others, and learn how to beat the so-called technical interview. All these features make SWEseek a compelling option for tech workers and students looking to start their job search process.

Throughout this paper we will go into an introduction to why SWEseek is a useful application, some of the design choices we made in regards to the database, API, queries, and user interface. And finally a user guide demonstrating how the users of the system can go about using SWEseek.

For information on running the source code of the application, please see the end of this document.

Introduction

Problem:

Computer science students looking for internships and early career software engineers all face similar problems. In our experience and from other students and software engineer's experiences shared online, applications to medium to large software companies is a numbers game. Not only this but also when getting interviews from companies in the tech space we noticed they all give the "technical interview", which even has a book (Cracking the coding interview) dedicated to it. From this there is a need for an application that could aid in the unique job search process faced by students and software engineers. Software engineers needed a way they could track their application process from start to finish and a place where they had the necessary resources to learn what they needed to know to pass their technical interviews. This solution would also be helpful if it could combine information from other resources as well, such as pay information and job postings that sites like LinkedIn, Indeed and other job websites provide.

Solution:

Our solution to this problem is SWEseek. SWEseek allows software engineers early in their career and students find jobs, track applications, know their worth, and learn interviewing skills. We have developed a unique tracking list where users can see jobs they have applied to, jobs they are interviewing for, jobs they have offers for, and jobs they have been rejected from. We have also built functionality to allow users to discover job postings as well as learn these interview questions they may be asked. Finally we have built a way for users to even apply for jobs that companies have posted on the site, and allow the companies to sign in and access the resumes and cover letters of applicants.

To summarize, SWEseek combines the ability for job seekers to apply for jobs, track application status, discover information about salaries and access learning resources about the technical interview all in on places. It also gives employers a dashboard to allow them to post and manage jobs, as well as see applicants allowing them to hire faster. This solution is sure to ease the entire job search process for both the job seeker and employer.

<u>Project Design</u>

Users and Transactions

Two types of users that in conjunction make SWEseek a great platform are the Employer and the Job seeker.

Functionalities of Job seeker:

1. Sign up
2. Log in
3. Add and retrieve their own tracking lists
4. Change job status in their tracking list
5. Add and remove jobs to and from tracking lists
6. View salary information of companies
7. Add salary information about companies
8. Save and access learning resources
9. Retrieve company reviews
10. Add, remove and access documents
11. Apply to jobs

Functionalities of Employer:

1. Sign up
2. Log in
3. Post and delete jobs to and from the platform
4. Retrieve document of Job seekers who applied to the job posting

API functionality

The API can handle all these functionalities alone by sending requests to the backend manually using postman. The complete functionality of the API is as follows, examples and complete requests can be found in the API documentation section:

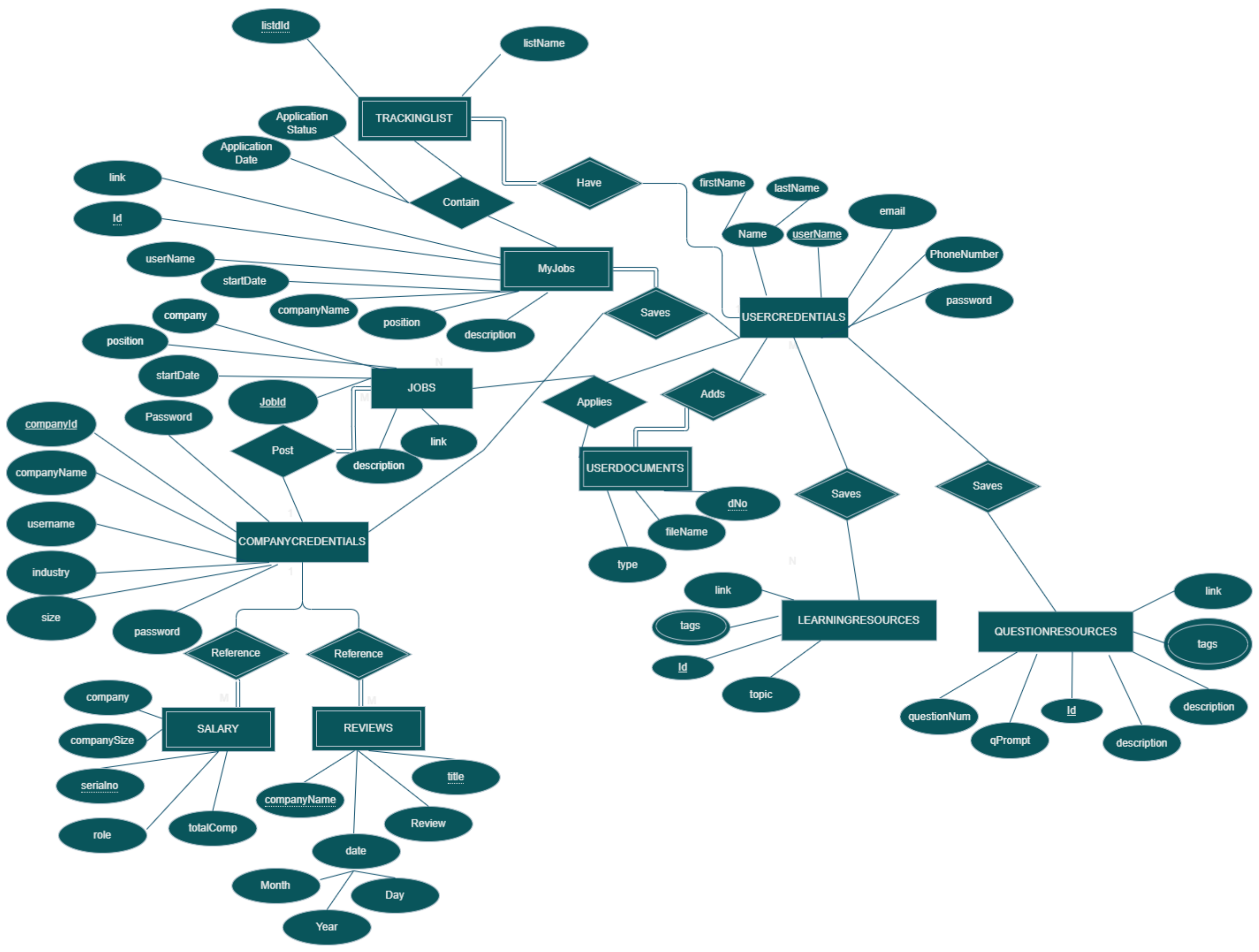
- User signup
- User Login
- Retrieve all TrackingLists of a user
- Retrieve specific TrackingList of a user

- Update the application status of a job in a TrackingList of a user
- Add a new TrackingList for a user
- Add a job to a TrackingList of a user
- Remove a job from a TrackingList of a user
- Post a new Job as a company
- Get every job posting that is posted
- Get summarized Salary info of popular (faang) companies
- Get summarizes Salary info of all companies
- Add a salary report for a specific company
- Get all learning resources available
- Get all practice questions available
- Get all company reviews
- Add documents to a users account (resume + cover letter)
- Get the names of all a users documents
- Retrieve a specific document a user has uploaded
- Get all usernames and documents that have applied to a specific job
- Apply to a job
- Retrieve a users saved learning resources
- Add a new learning resource to a users account
- Remove a learning resource from a users account
- Retrieve a users saved practice questions
- Add a new practice questions to a users account
- Remove a practice questions from a users account
- Remove a posted job
- Edit a posted job
- Get all a companies posted jobs
- Remove a document from a users account
- Sign up a company
- Login as a company

General Application Design

Our application relies on the frontend speaking to the backend (through requests) and the backend retrieving, adding, modifying, and deleting things from the database based on the request and sending required information back to the frontend as required. This is necessary to protect the database from potentially malicious users, and restrict their access to only the allowed endpoints. It also allows the application to scale well. We can host a server with the backend on it and host the site online which can make requests to the server. This would allow multiple people to use the API service through multiple frontends at the same time. If we ever ran into speed issues we could also look at horizontal scaling and add servers throughout our target users regions. So by using this design of sending requests to a backend server which updates the database we can create a “funnel” of data where multiple frontends can have data flowing into a single backend server and a single database instance.

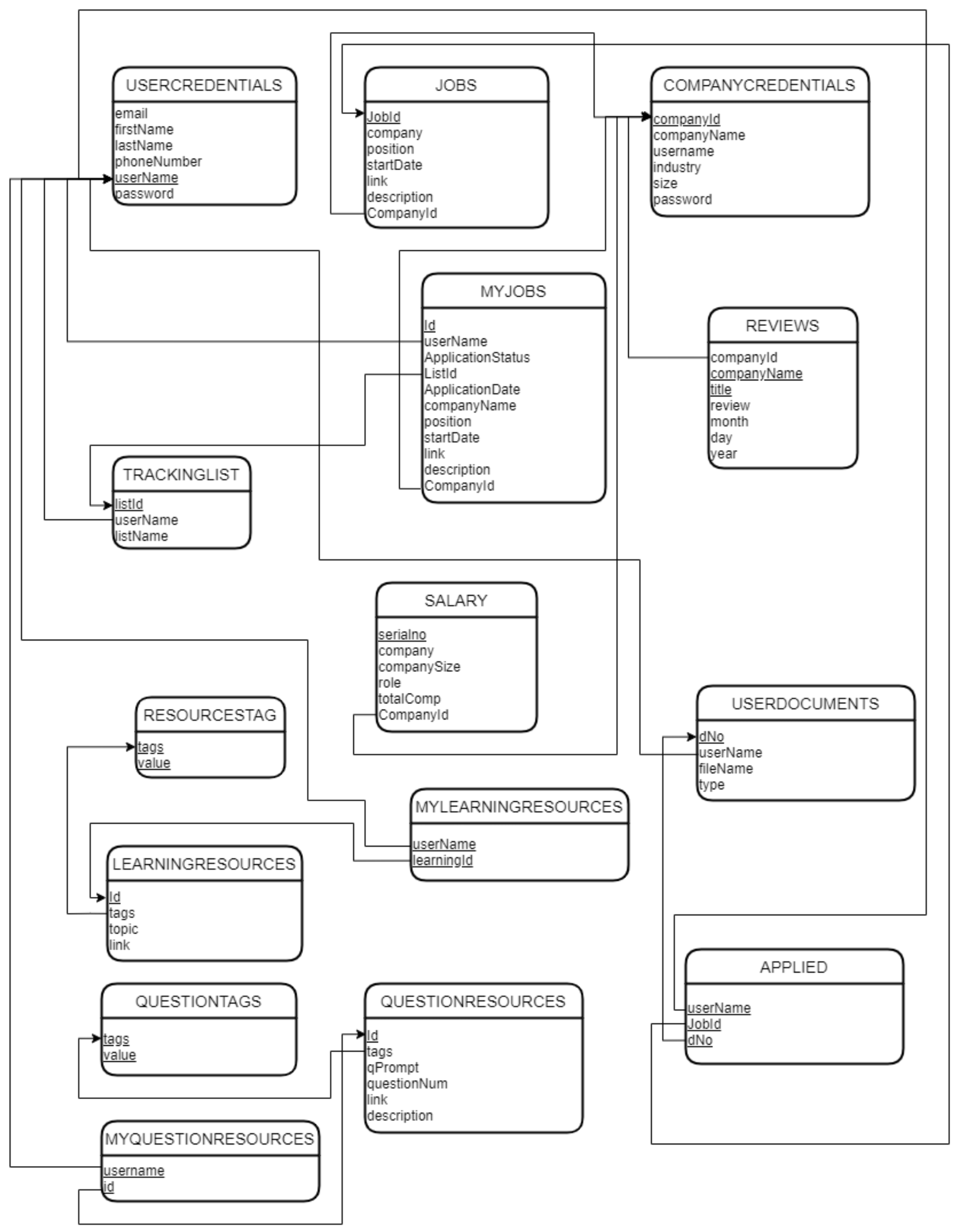
ER Diagram



Implementation:

The Relational model diagram strictly follows the algorithm for converting the entity relationship diagram to the relational schema diagram. Idea and implementation of SWEseek follows the Relational Model. The change in the implementation with respect to the Model is the Table names. Adding to this no significant and unusual change was made during the implementation.

Relational Model



DBMS Choice

The DBMS selected for the implementation of SWEseek is the MySQL Relational database. The purpose of selecting this database was the ease of categorizing data, Flexibility with table connections and exceptionally high performance. SWEseek database consists of 16 tables. SQL queries mentioned below are in the syntax required to run on the MySQL shell; they have been converted to python format to run on Flask framework.

SQL Statements

The SQL queries corresponding to the specified endpoints:

Endpoint 1:
```
INSERT INTO USERCREDENTIALS(email,firstName,lastName,phoneNumber,username,password)
VALUES('jibran6@gmail.com','jibran','khan','2832389801','jbrn','jibn2193');
```

Endpoint 2:
```
Select * FROM USERCREDENTIALS WHERE username=’num4n’ AND password=’Numan1234’;
```

Endpoint 3:
```
SELECT T.ListId, T.ListName
FROM TRACKINGLIST AS T
WHERE T.userName='num4n';

SELECT
M.Id,M.companyName,M.Position,M.link,M.applicationDate,M.applicationStatus,M.startDat
e,M.description
FROM TRACKINGLIST AS T ,MYJOBS AS M
WHERE M.ListId = T.listId;
```

Endpoint 4:
```
SELECT T.ListId, T.ListName
FROM TRACKINGLIST AS T
WHERE T.userName='num4n' AND T.listName = 'Back-End';

SELECT
M.Id,M.companyName,M.Position,M.link,M.applicationDate,M.applicationStatus,M.startDat
e,M.description
FROM TRACKINGLIST AS T ,MYJOBS AS M
WHERE T.listId=1 AND M.ListId = T.listId;
```

Endpoint 4.1:
```
UPDATE TRACKINGLIST AS T, MYJOBS AS M
SET applicationStatus = 'Offer Recieved'
WHERE listName = 'N/A'AND listId='N/A'AND T.Id =3 AND T.Id=M.Id;
```

Endpoint 5:

```
INSERT INTO TRACKINGLIST (username, listName)
VALUES ('num4n', 'DataScience');
```

Endpoint 6:

```
INSERT INTO MYJOBS
(JobId,userName,ApplicationStatus,ListId,ApplicationDate,companyName,position,startDa
te,CompanyId,link,description)
VALUES (1,'nick_knapton', 'Offer Received', 1
,'01-01-2022','Google','Back-End','01-01-2021','002','amazoncareers.ca','N/A')
```

Endpoint 7:

```
DELETE FROM MYJOBS WHERE listId=%s AND myJobId=%s AND userName=%s
```

Endpoint 8:

```
INSERT INTO JOBS (CompanyId, company, position, StartDate, link, Description)
VALUES ('003','IBM','Front-End','01-01-2022','ibmrecruitment.ca','N/A')

SELECT JobId FROM JOBS WHERE company = 'IBM' AND position = 'Front-End';
```

Endpoint 9:

```
SELECT J.companyId,position,size,industry,link,description FROM JOBS AS J,
COMPANYCREDENTIALS AS C WHERE J.companyId = C.companyId;
```

Endpoint 10:

```
SELECT company,companySize,industry
FROM SALARY AS S, COMPANYCREDENTIALS AS C WHERE S.companyId = C.companyId;

SELECT company,role,totalComp
FROM SALARY AS S, COMPANYCREDENTIALS AS C WHERE S.companyId = C.companyId
```

Endpoint 11:

```
SELECT company,companySize,industry
FROM SALARY AS S, COMPANYCREDENTIALS AS C WHERE S.companyId = C.companyId;

SELECT company,role,avg(totalComp)
FROM SALARY AS S, COMPANYCREDENTIALS AS C
WHERE S.companyId = C.companyId
GROUP BY company;
```

Endpoint 12:

```
INSERT INTO SALARY (company, companySize, role, totalComp)
VALUES ('Google', 'Large', 'Front-End', 119000);
```

Endpoint 13:

```
SELECT id,tags,topic,link
FROM LEARNINGRESOURCES;
SELECT tags,value
FROM RESOURCESTAG
```

```
WHERE tags=1;
Endpoint 14:
SELECT M.username,L.tags,L.topic,L.link
FROM LEARNINGRESOURCES AS L, MYLEARNINGRESOURCES AS M
WHERE M.username = 'zeeshansalim' AND M.learningId = L.tag
SELECT value
FROM RESOURCESTAG
WHERE tags=3;
```

Endpoint 15:

```
SELECT * FROM REVIEWS WHERE companyName ='Google';
```

Endpoint 16:

```
INSERT INTO USERDOCUMENTS (username,fileName,file,type)
VALUES('zeeshansalim','file 6','path6','pdf');
```

Endpoint 17.1:

```
SELECT fileName,dNo,file,type FROM USERDOCUMENTS WHERE userName ='num4n' AND
fileName='file1';
```

Endpoint 17.2:

```
SELECT U.file,U.type
FROM APPLIED AS A, USERDOCUMENTS AS U
WHERE A.username = 'nick_knapton' AND  JobId = 2 AND U.dNo = A.dNo
```

Endpoint 18:

```
INSERT INTO APPLIED(username,JobId)
VALUES ('zeeshansalim','5','2');
```

Endpoint 19:

```
SELECT M.username,L.tags,L.topic,L.link
FROM LEARNINGRESOURCES AS L, MYLEARNINGRESOURCES AS M
WHERE M.username = 'zeeshansalim' AND M.learningId = L.tag
```

Endpoint 19.1:

```
INSERT INTO LEARNINGRESOURCES (tags,topic,link)
VALUES (4,'coding tutorials','hackthechallenge.com');

SELECT Id FROM LEARNINGRESOURCES WHERE tags=4 AND topic ='coding tutorials';

INSERT INTO MYLEARNINGRESOURCES (username,learningId)
VALUES ('zeeshansalim', 7);

INSERT INTO RESOURCESTAG (tags,value)
VALUES(4,'#easycode');
```

Endpoint 19.2:

```
DELETE FROM MYLEARNINGRESOURCES WHERE username ='num4n' AND learningId =2;
```

Endpoint 20:
```
SELECT M.id,Q.tags,Q.qPrompt,Q.questionNum,Q.link,Q.description
FROM MYQUESTIONRESOURCES AS M, QUESTIONRESOURCES AS Q
WHERE username = 'num4n' AND M.id = Q.Id;
SELECT value
FROM QUESTIONTAGS
WHERE tags=3;
```

Endpoint 20.1:
```
INSERT INTO MYQUESTIONRESOURCES(username,id)
VALUES ('num4n', id);
```

Endpoint 20.2:
```
DELETE FROM MYQUESTIONRESOURCES WHERE username='num4n' AND id=1
```

Endpoint 21.1:
```
SELECT CompanyId FROM COMPANYCREDENTIALS WHERE companyName = 'Blizzard';
INSERT INTO JOBS (CompanyId, company, position, StartDate, link, Description)
VALUES ('006','Blizzard', 'Back-End','11-02-2020','blizzardhiring.ca','N/A');
```

Endpoint 21.2:
```
SELECT JobId FROM JOBS WHERE company ='Amazon' AND position ='Back-End';
DELETE FROM JOBS WHERE JobId=1;
```

Endpoint 21.3:
```
UPDATE JOBS
SET startdate = '09-09-2029',description='very long', link = 'newmicro.com'
WHERE company = 'Microsoft'AND JobId=7 AND position='Full-Stack';
```

Endpoint 22:
```
SELECT JobId,position,startDate,link,description
FROM JOBS
WHERE company ='Microsoft';
```

EndPoint 23:
```
DELETE FROM APPLIED WHERE userName='num4n' AND dNo=1;
DELETE FROM USERDOCUMENTS WHERE userName='num4n' AND dNo=1;
```

Endpoint 24:
```
INSERT INTO COMPANYCREDENTIALS(companyName, username, industry, size, password)
VALUES ('DoorDash','DDSH','food delivery','doordashjobs.ca','HDWO2347');
```

Front-End Implementation

Firstly we chose to use Reactjs for the frontend as it allowed us to quickly build up pages that could react to changes in data seamlessly. React also supported all the libraries we needed to use like Axios, React-Bootstrap, and react-beautiful-dnd. We used Axios to make POST, GET, PUT, and DELETE requests to the backend, React-Bootstrap to quickly build up pages, and react-beautiful-dnd to allow us to have moveable cards and columns in the tracking page.

Back-End Implementation

Next we chose to use the Python Flask framework for our backend as it allowed simple building of API endpoints that just worked. We then used flask_mysqldb to aid in querying the database, and a few build in packages to aid in storing and retrieving files.

API Documentation

Our professional API documentation using the postman tool can be found here: https://documenter.getpostman.com/view/14606338/UVRAJ7Cc

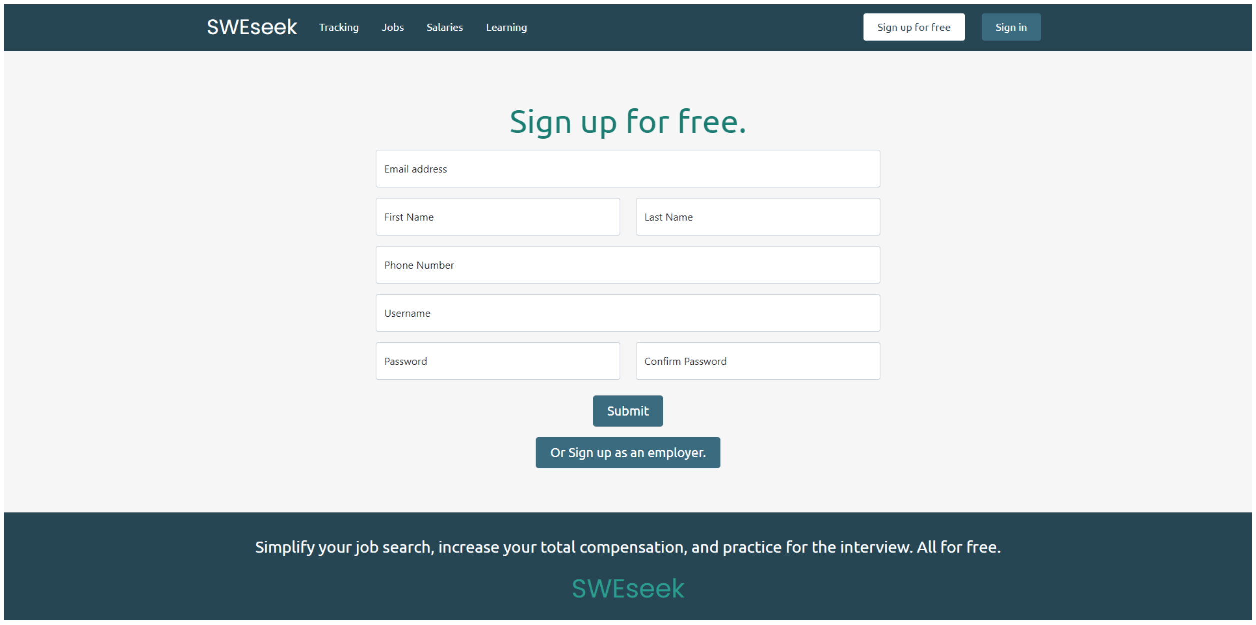
Note: There are three endpoints which were shown in our endpoints document but not present in the API documentation which are: addUserDocument, getUserDocuments and removeUserDocuments. This is because these endpoints require file type inputs (transfer of files over server) which we were unable to provide as an input in json format on the postman application. **However, these endpoints are perfectly functional in our code and their functionalities have been demonstrated in the presentation**.

User Guide

SignUp Page

To use the application you must sign up. All features other than viewing the landing page require a login so this is the first step.

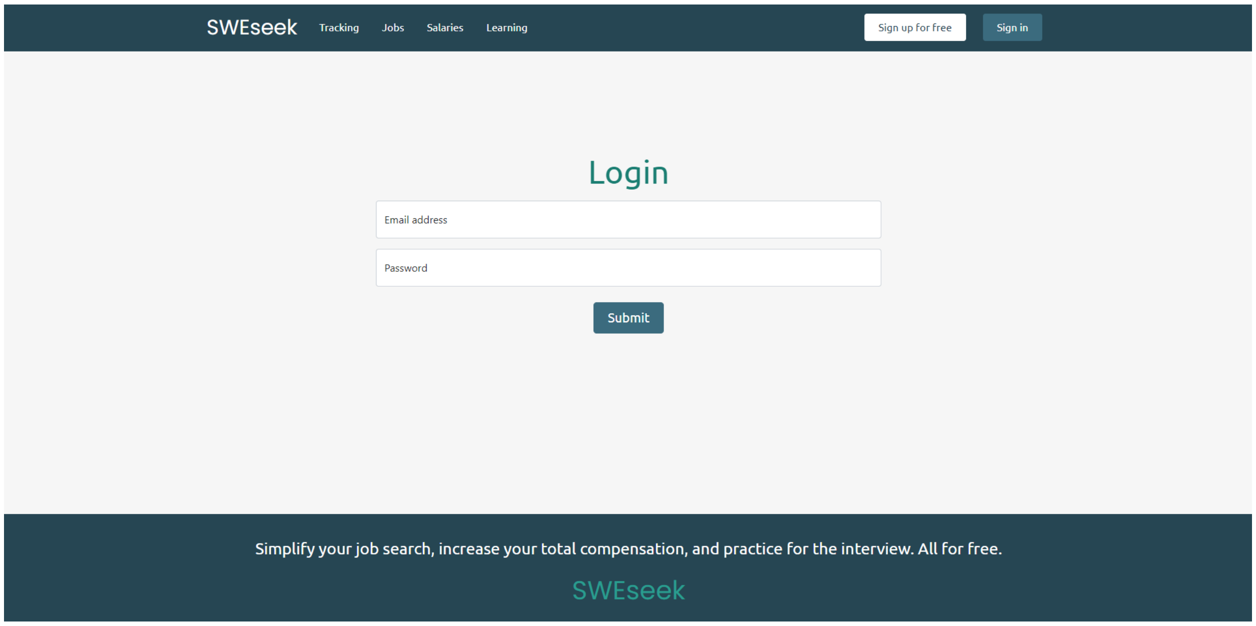
First navigate to the sign up page, you will see the following:



On this page you can either fill in the information and click “Submit” to sign up as a regular user (student/job seeker) or click on “Or Sign up as an employer.” which will take you to the employer sign up covered later in this guide.

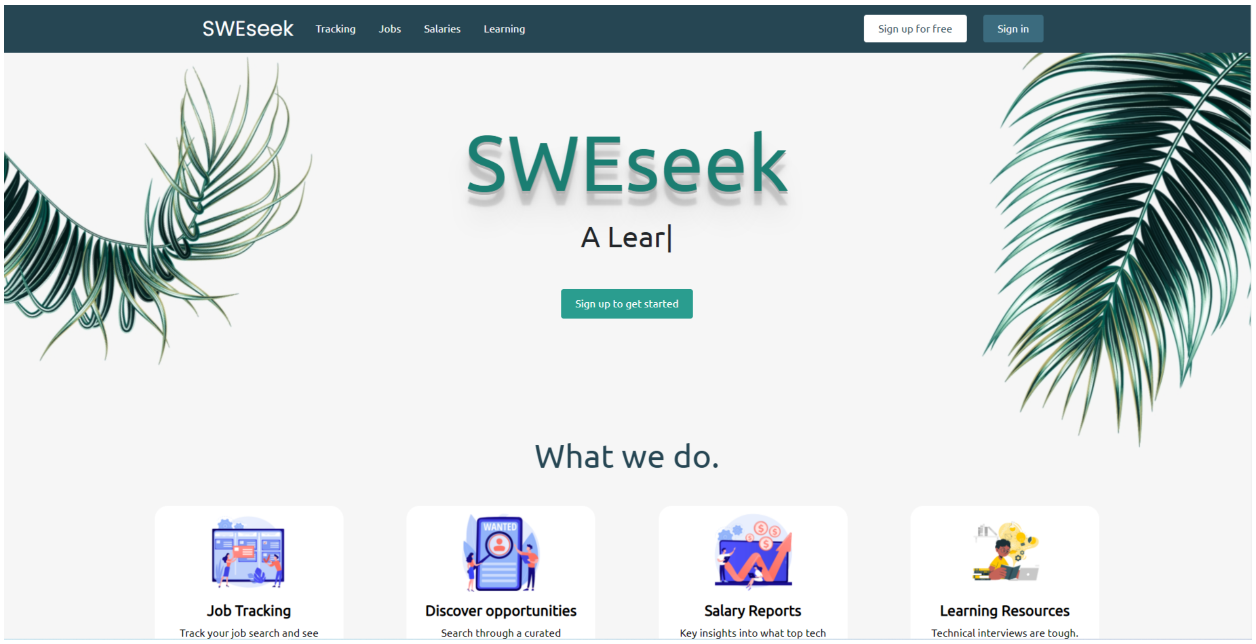
Login Page

If a user is already signed up they can simply go to the “Sign in” page and fill out the form to login. It should look something like below.


SWEseek
Tracking
Jobs
Salaries
Learning
Sign up for free
Sign in
Login
Email address
Password
Submit
Simplify your job search, increase your total compensation, and practice for the interview. All for free.
SWEseek

Landing Page

The landing page doesn't serve much functionality other than being a landing for new users, describing what SWEseek does.



What we do.

Job Tracking
Track your job search and see

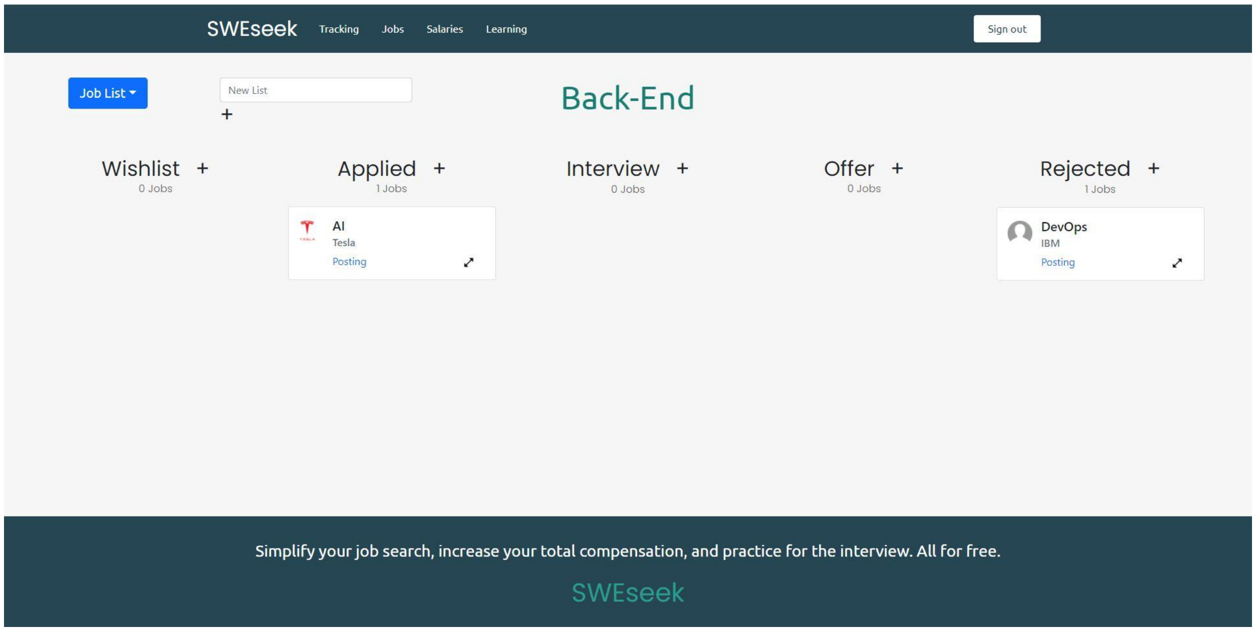
Discover opportunities
Search through a curated

Salary Reports
Key insights into what top tech

Learning Resources
Technical interviews are tough.

Tracking Page

The tracking page allows users to track jobs they have applied for. The general look of the tracking page is as follows.



Back-End

Wishlist + 0 Jobs

Applied + 1 Jobs

AI
Tesla
Posting

Interview + 0 Jobs

Offer + 0 Jobs

Rejected + 1 Jobs

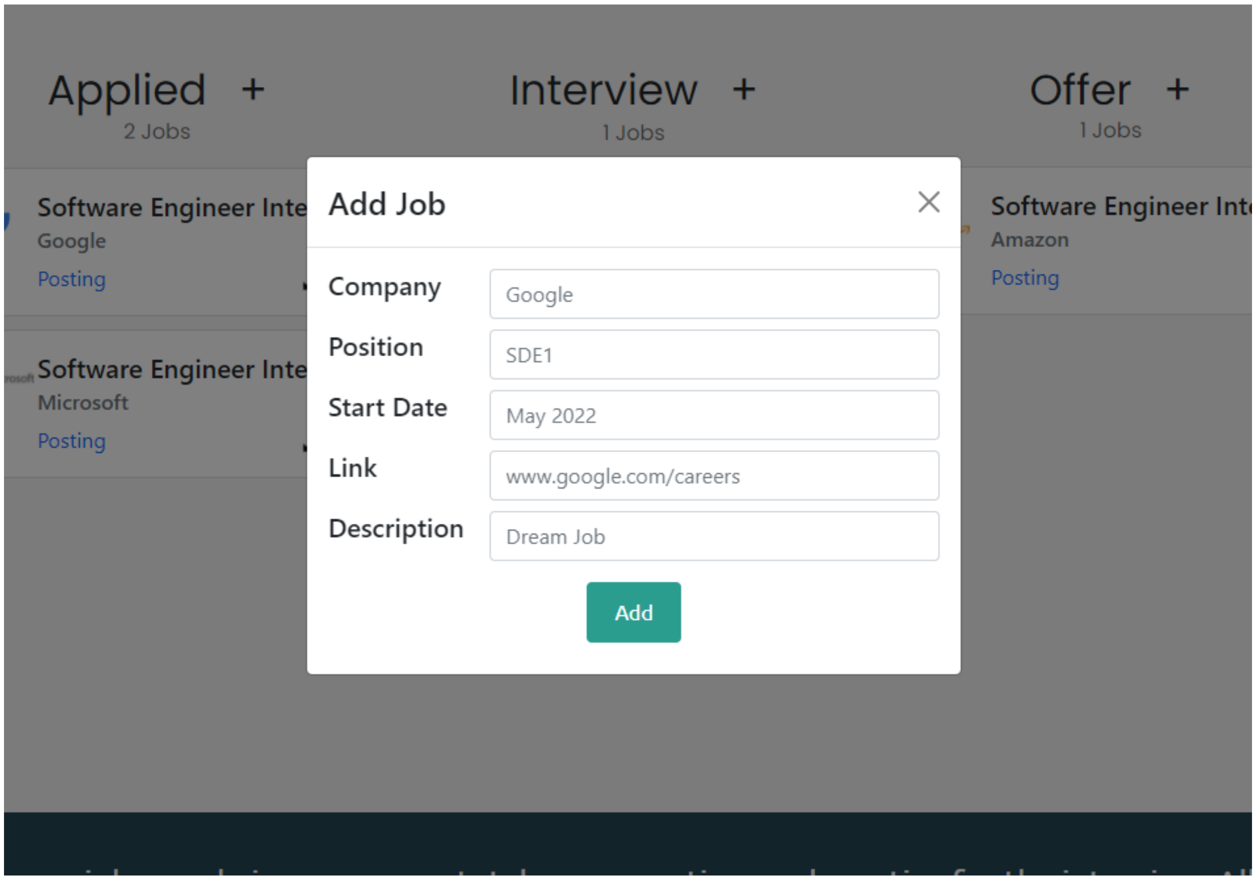
DevOps
IBM
Posting

Simplify your job search, increase your total compensation, and practice for the interview. All for free.

SWEseek

To move a job to a different “state” (Wishlist, Applied, Interview, Offer, Rejected), simply drag and drop the card.

To add a new job click the “plus” icon on any of the job states and fill out the form shown below.



Applied + 2 Jobs

Interview + 1 Jobs

Offer + 1 Jobs

Software Engineer Inte
Google
Posting

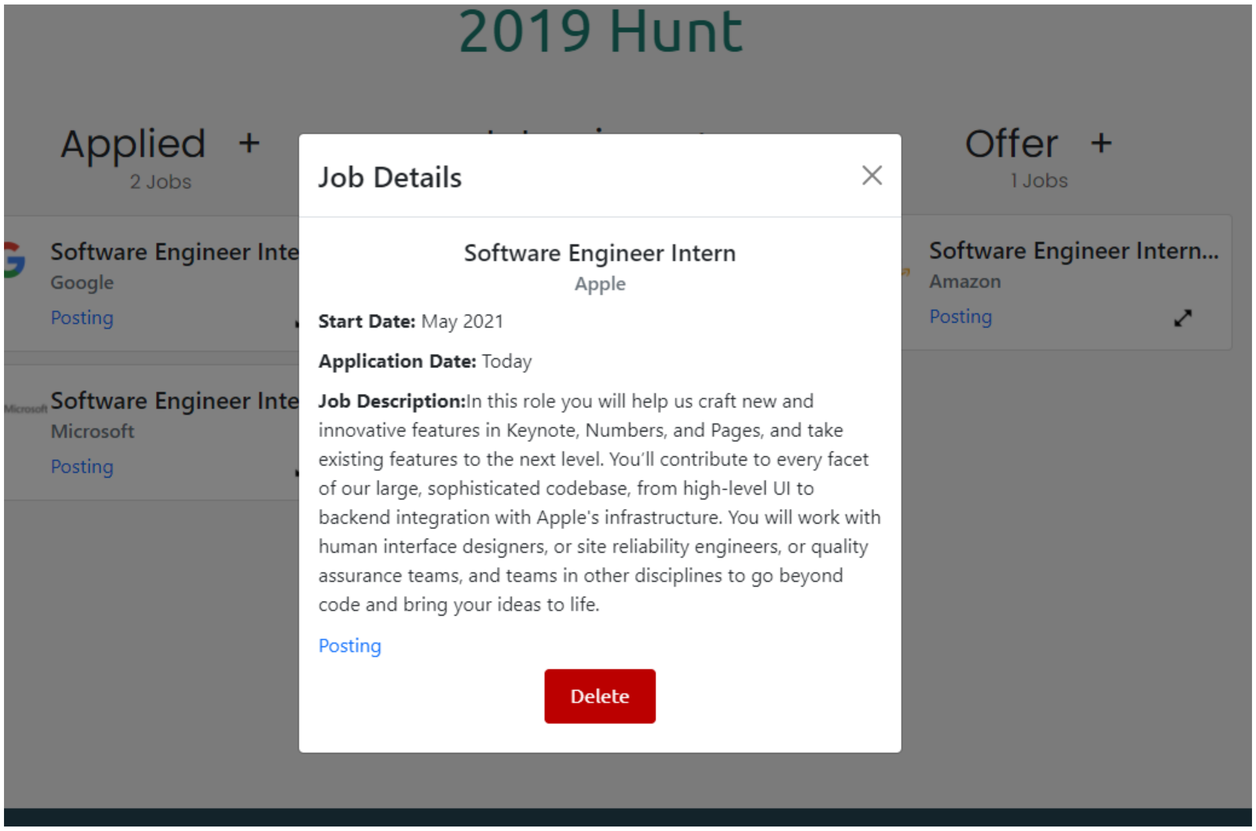
Software Engineer Inte
Microsoft
Posting

Software Engineer Int
Amazon
Posting

Add Job

Company	Google
Position	SDE1
Start Date	May 2022
Link	www.google.com/careers
Description	Dream Job

To see details of a specific job, click the “expand” icon and a popup shows, as shown below.



2019 Hunt

Applied + 2 Jobs

Software Engineer Inte
Google
Posting

Software Engineer Inte
Microsoft
Posting

Offer + 1 Jobs

Software Engineer Intern...
Amazon
Posting

Job Details

Software Engineer Intern
Apple

Start Date: May 2021

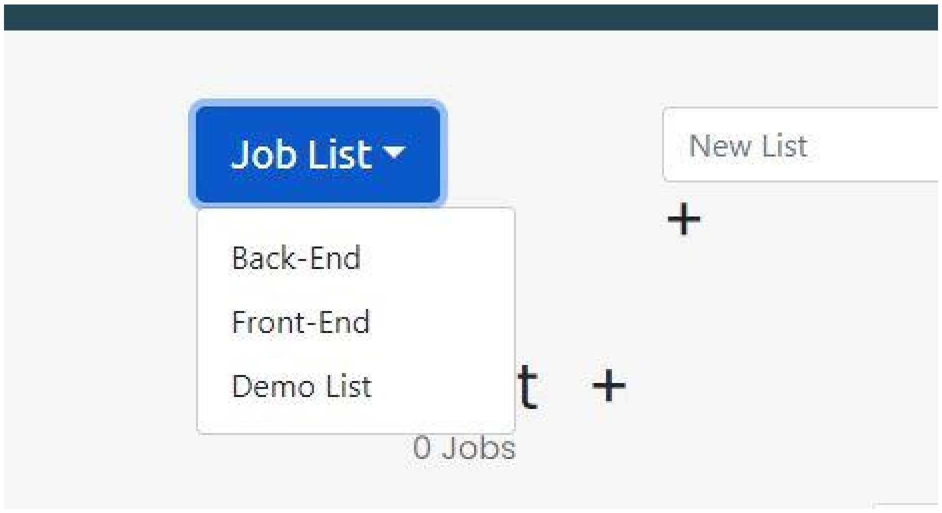
Application Date: Today

Job Description: In this role you will help us craft new and innovative features in Keynote, Numbers, and Pages, and take existing features to the next level. You'll contribute to every facet of our large, sophisticated codebase, from high-level UI to backend integration with Apple's infrastructure. You will work with human interface designers, or site reliability engineers, or quality assurance teams, and teams in other disciplines to go beyond code and bring your ideas to life.

Posting

Delete

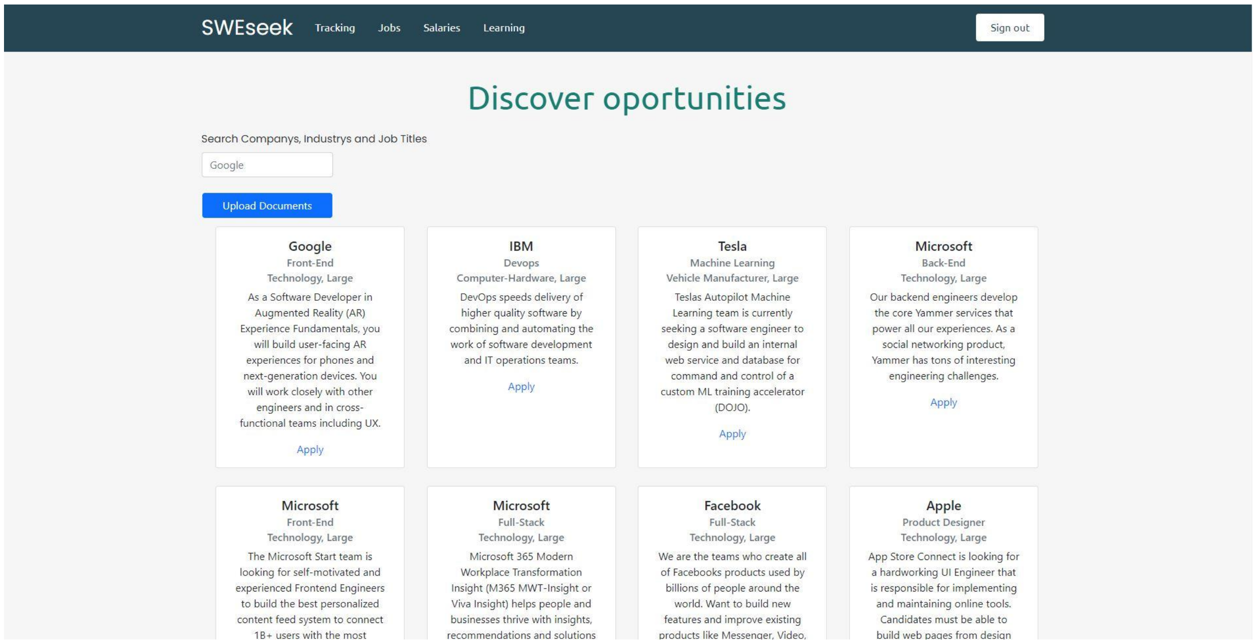
To switch the list your viewing, simply click the "Job List" dropdown menu and select the list you would like to see. Should look something like below.



Finally to create a new list, simply fill in the box at the top lift with the list name and click the "plus" icon below it.

Jobs Page

Upon navigating to the Jobs page you will see something like what's below.



SWEseek Tracking Jobs Salaries Learning Sign out

Discover oportunities

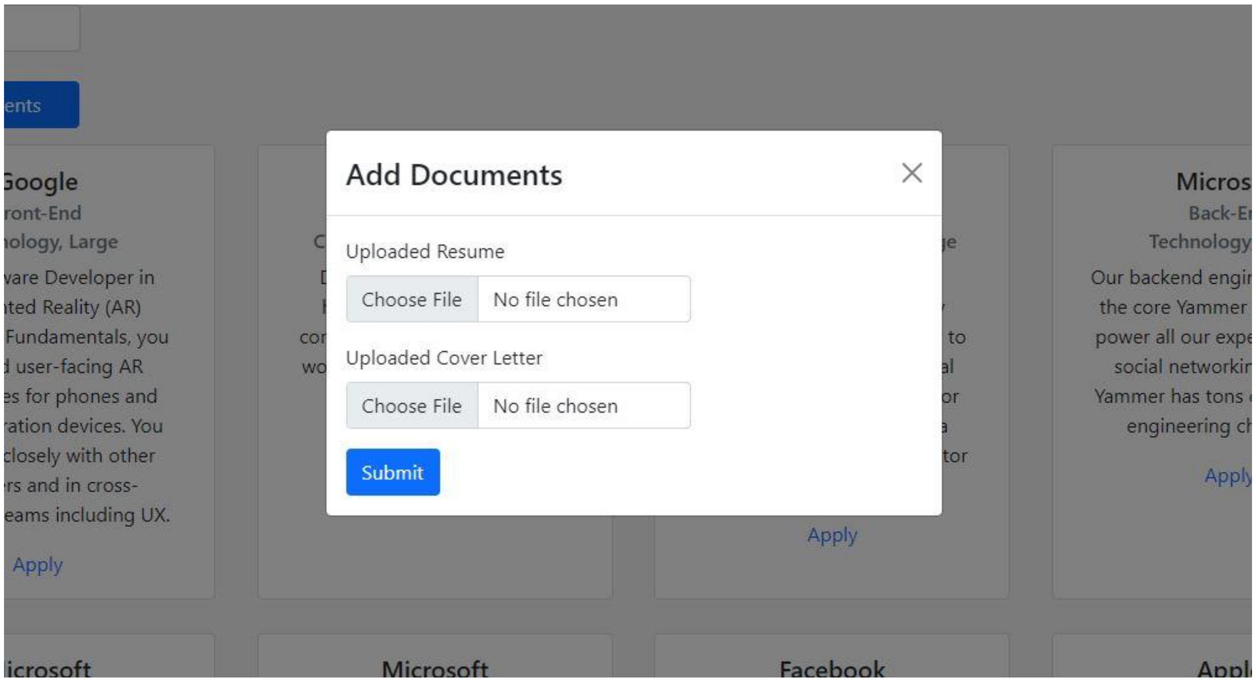
Search Companys, Industrys and Job Titles

Google

Upload Documents

Google Front-End Technology, Large As a Software Developer in Augmented Reality (AR) Experience Fundamentals, you will build user-facing AR experiences for phones and next-generation devices. You will work closely with other engineers and in cross-functional teams including UX. Apply	**IBM** Devops Computer-Hardware, Large DevOps speeds delivery of higher quality software by combining and automating the work of software development and IT operations teams. Apply	**Tesla** Machine Learning Vehicle Manufacturer, Large Teslas Autopilot Machine Learning team is currently seeking a software engineer to design and build an internal web service and database for command and control of a custom ML training accelerator (DOJO). Apply	**Microsoft** Back-End Technology, Large Our backend engineers develop the core Yammer services that power all our experiences. As a social networking product, Yammer has tons of interesting engineering challenges. Apply
Microsoft Front-End Technology, Large The Microsoft Start team is looking for self-motivated and experienced Frontend Engineers to build the best personalized content feed system to connect 1B+ users with the most	**Microsoft** Full-Stack Technology, Large Microsoft 365 Modern Workplace Transformation Insight (M365 MWT-Insight or Viva Insight) helps people and businesses thrive with insights, recommendations and solutions	**Facebook** Full-Stack Technology, Large We are the teams who create all of Facebooks products used by billions of people around the world. Want to build new features and improve existing products like Messenger, Video,	**Apple** Product Designer Technology, Large App Store Connect is looking for a hardworking UI Engineer that is responsible for implementing and maintaining online tools. Candidates must be able to build web pages from design

Prior to applying to any jobs, a user must upload at least 1 resume. To do this, simply click the "Upload Documents" button and fill out the form shown below.



Add Documents

Uploaded Resume

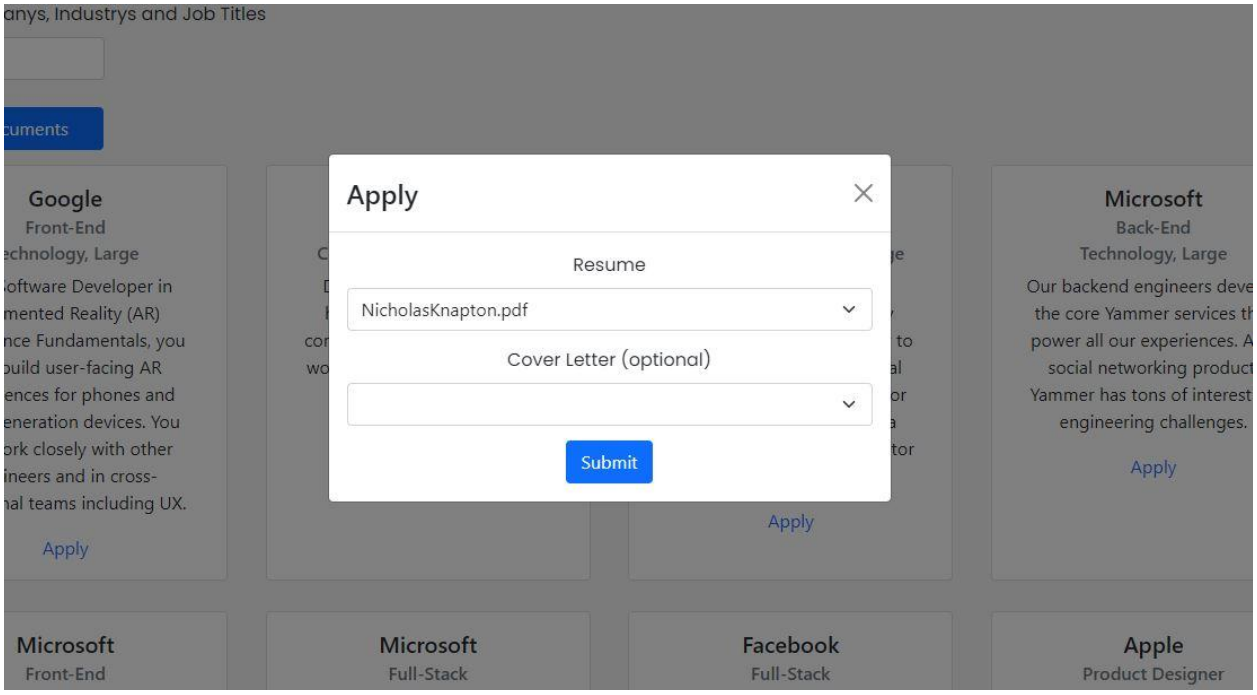
Choose File No file chosen

Uploaded Cover Letter

Choose File No file chosen

Submit

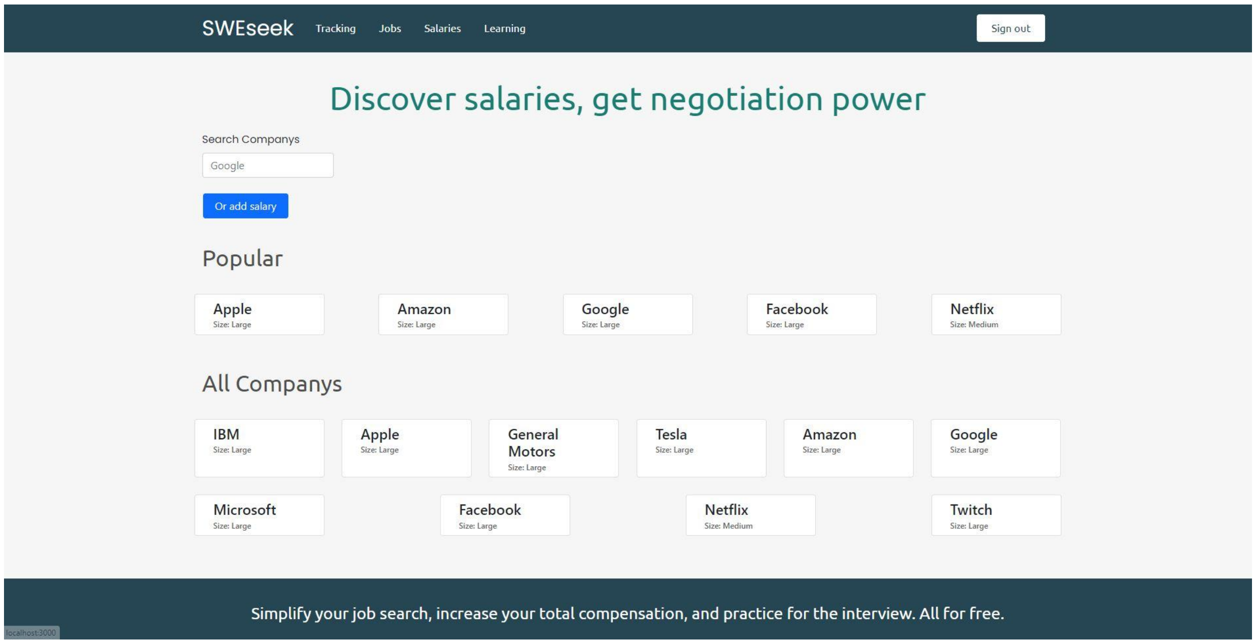
After uploading at least 1 resume, you can now click "Apply" on any job and fill out the form shown below to apply for the job as shown below.



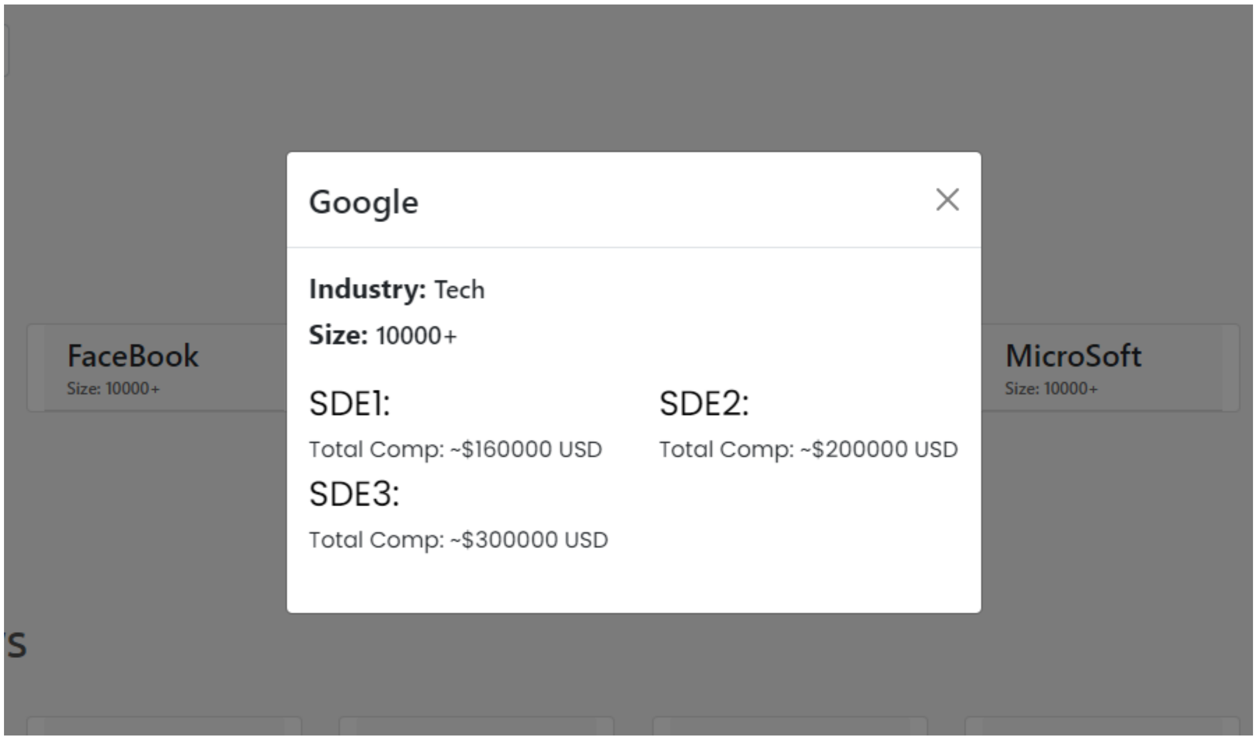
Finally users can search for companies, industries, and job titles using the search bar at the top left corner.

Salaries Page

The salaries page has a list of popular companies as well as all companies in the database. Upon navigating to the page you should see something like below.



To see the salaries offered at a company, simply click on the desired company to see something like below.



Google

Industry: Tech

Size: 10000+

SDE1:

Total Comp: ~$160000 USD

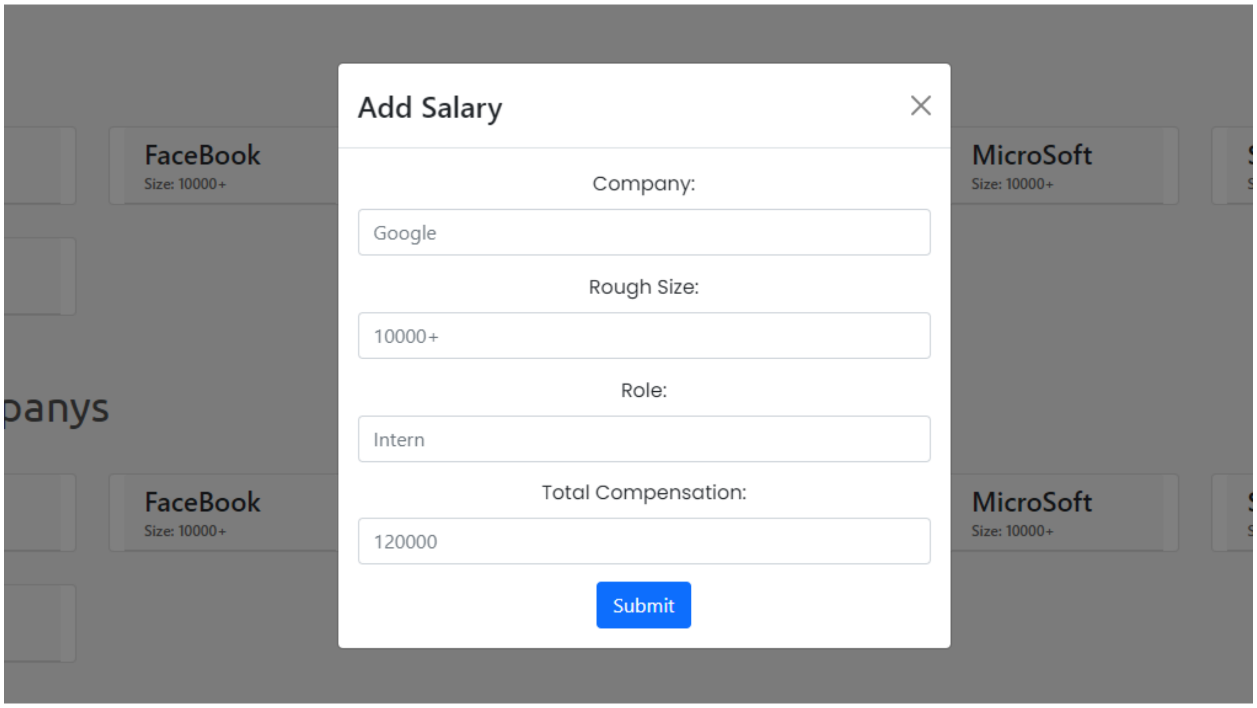
SDE2:

Total Comp: ~$200000 USD

SDE3:

Total Comp: ~$300000 USD

To add a salary of your own, users can click on the “Or add salary” button and fill out the form below.



Add Salary

Company: Google

Rough Size: 10000+

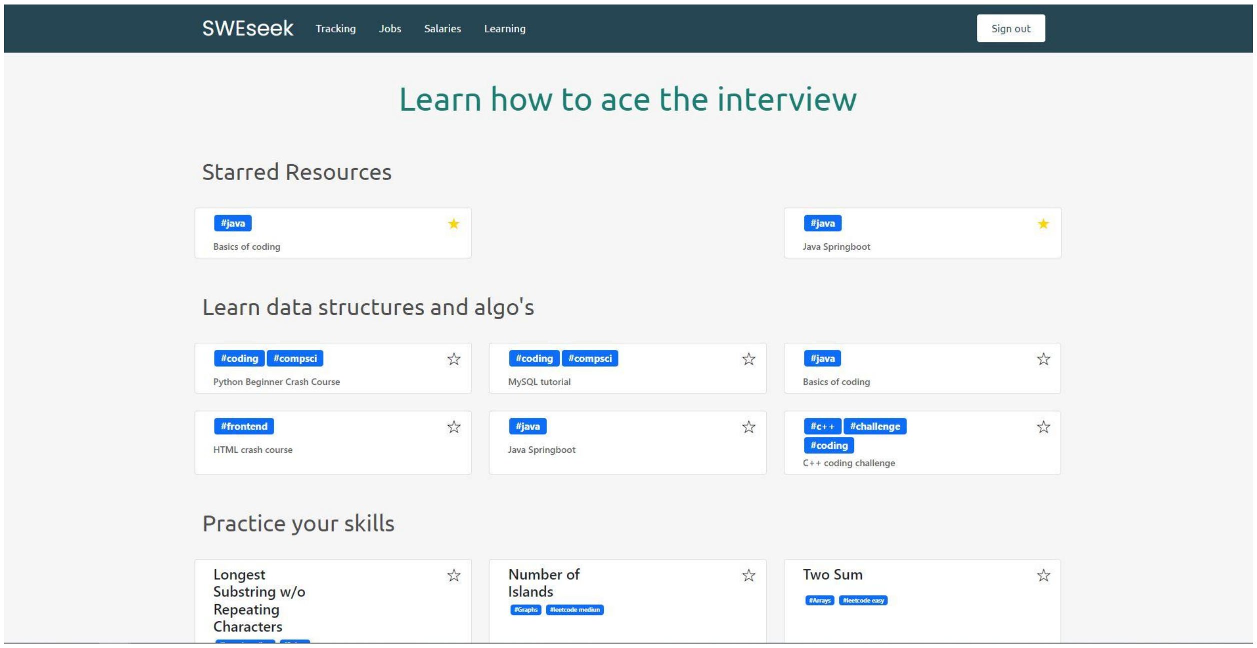
Role: Intern

Total Compensation: 120000

Finally users can also search this page for companies using the search bar at the top left of the page.

Learning Page

The learning page provides learning resources as well as practice questions. Upon navigating to this page you should come across the following.



Learn how to ace the interview

Starred Resources

#java
Basics of coding

#java
Java Springboot

Learn data structures and algo's

#coding #compsci
Python Beginner Crash Course

#coding #compsci
MySQL tutorial

#java
Basics of coding

#frontend
HTML crash course

#java
Java Springboot

#c++ #challenge #coding
C++ coding challenge

Practice your skills

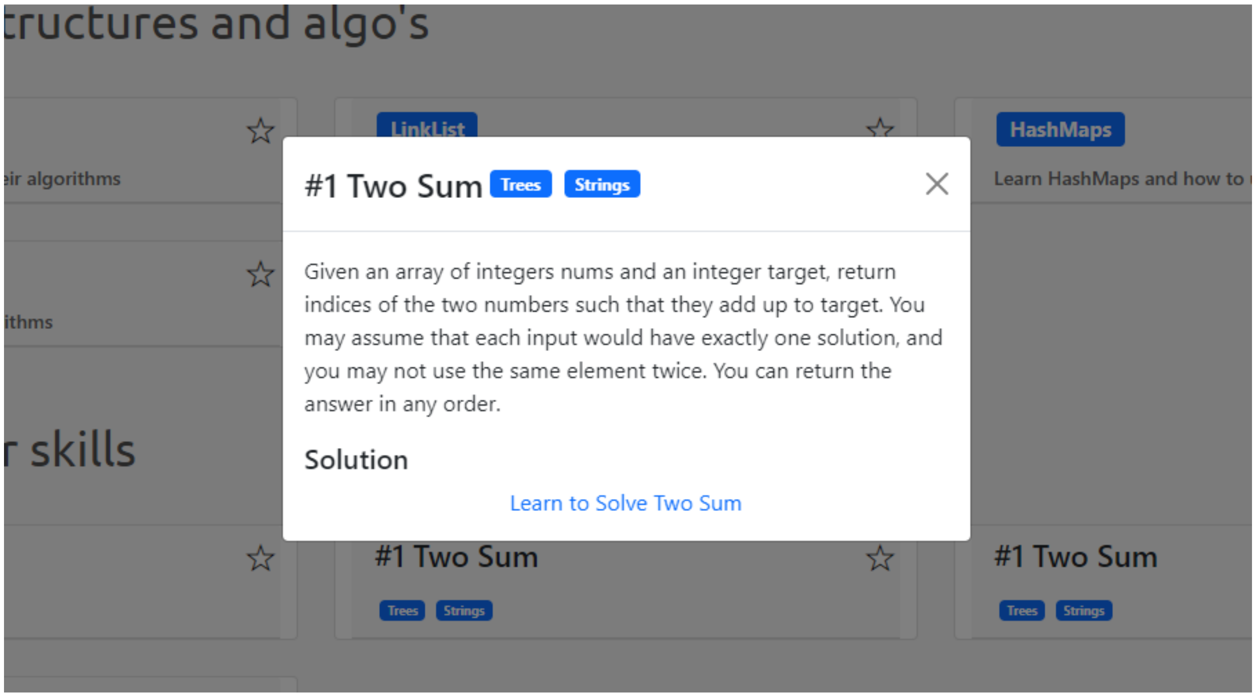
Longest Substring w/o Repeating Characters

Number of Islands
#Graphs #leetcode mediun

Two Sum
#Arrays #leetcode easy

To see either a learning resource simply click on the resource and you will be redirected to the learning resource.

To see a practice question, simply click on the practice question card and you will see a pop up with the question that looks something like below.


tructures and algo's

HashMaps

Learn HashMaps and how to

LinkList

ir algorithms

ithms

r skills

#1 Two Sum Trees Strings

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target. You may assume that each input would have exactly one solution, and you may not use the same element twice. You can return the answer in any order.

Solution

Learn to Solve Two Sum

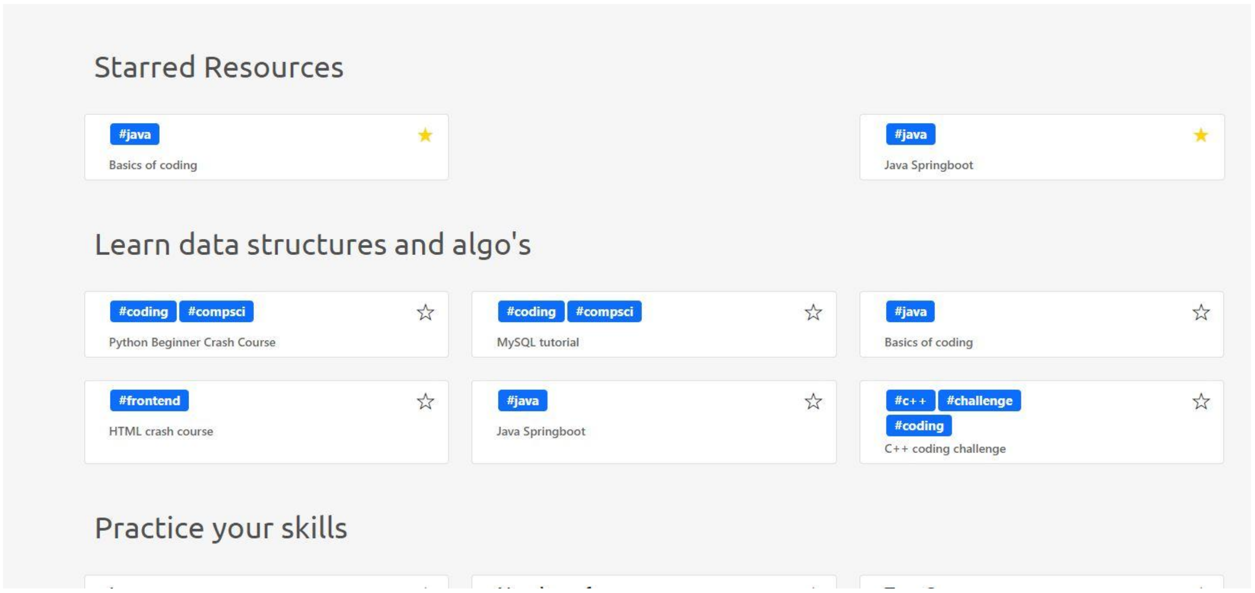
#1 Two Sum

Trees Strings

#1 Two Sum

Trees Strings

Finally, users can star (save) a resource for later by clicking the star icon on the top right of a card. Upon saving a resource, you should see something like below.


Starred Resources

#java
Basics of coding

#java
Java Springboot

Learn data structures and algo's

#coding #compsci
Python Beginner Crash Course

#coding #compsci
MySQL tutorial

#java
Basics of coding

#frontend
HTML crash course

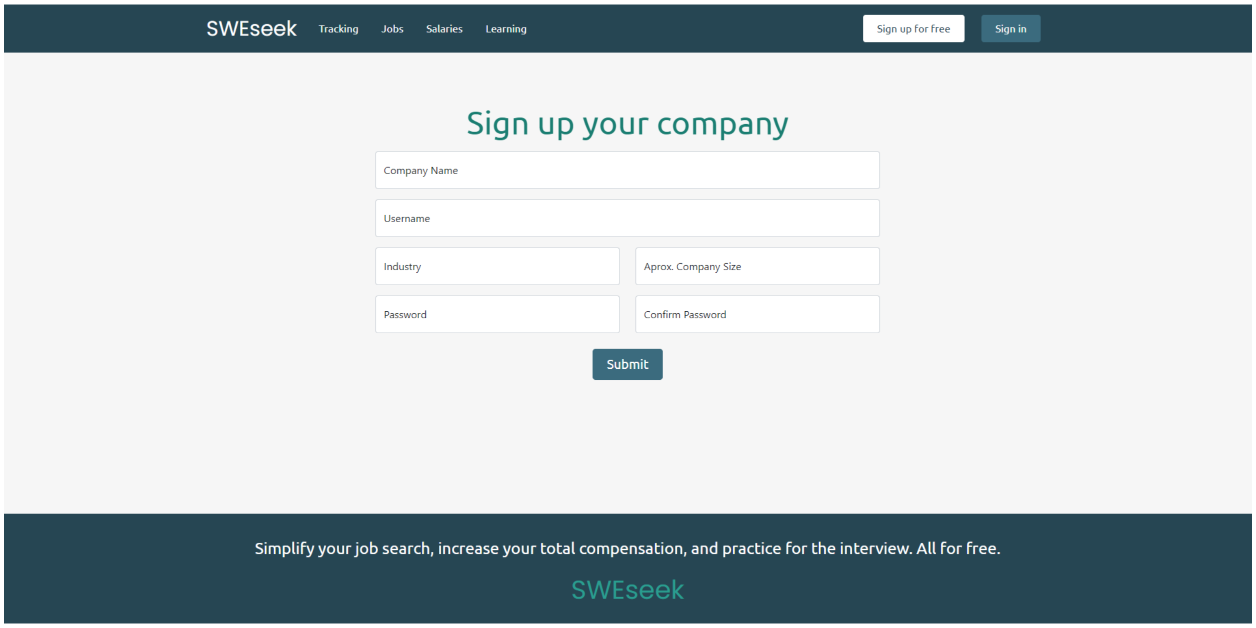
#java
Java Springboot

#c++ #challenge #coding
C++ coding challenge

Practice your skills

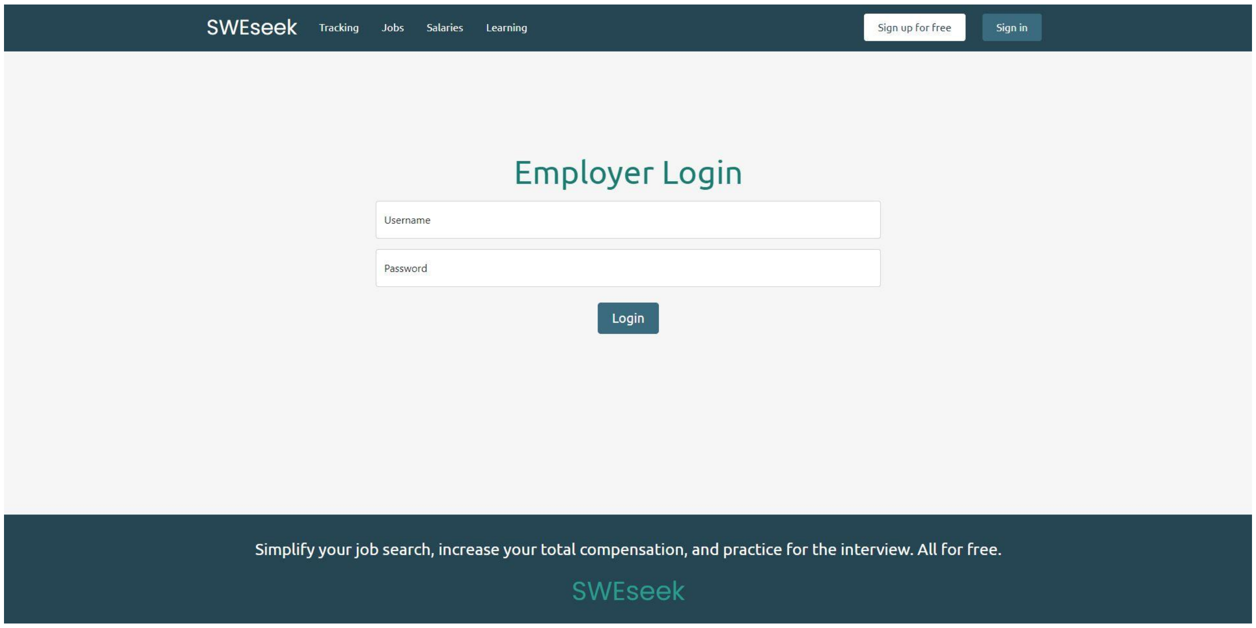
Employer Signup

Employers can sign up by navigating to the employer sign up page (you can reach this by clicking “Or sign up as an employer” on the regular sign up page) and filling out the form presented, it will look something like below.



Employer Sign In

If an employer already has an account, they may sign in with their username and password on the Employer Login page, you can get to this page by going to the login page and clicking “Or login as Employer”. It looks something like this.

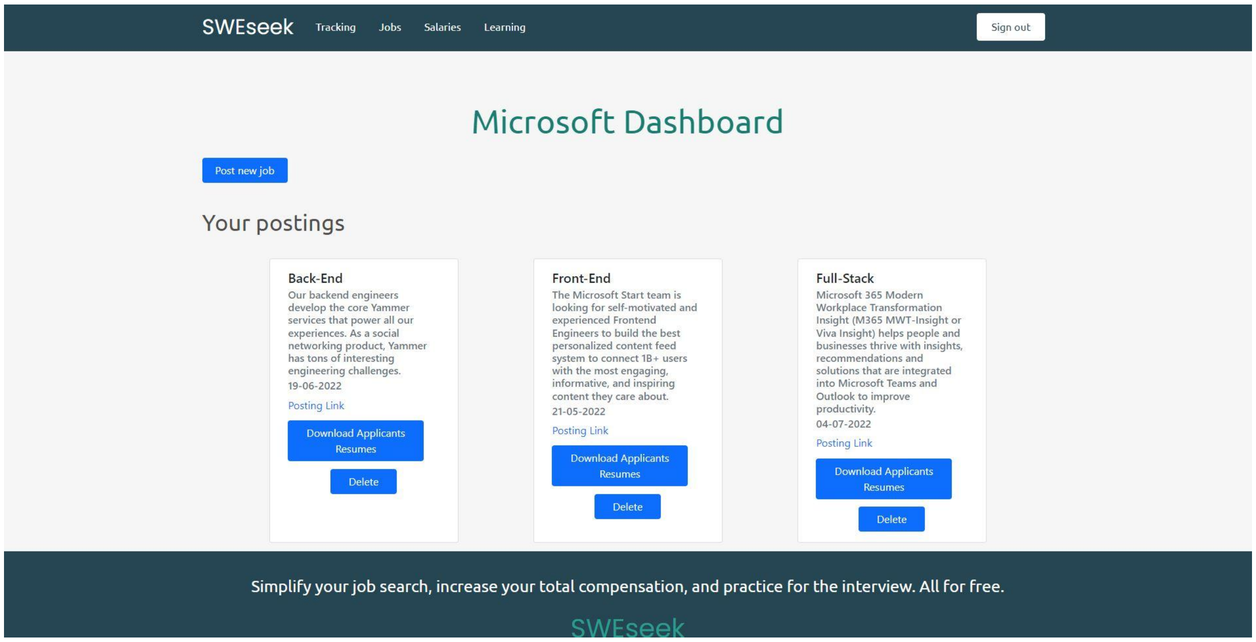


Employer Login

Simplify your job search, increase your total compensation, and practice for the interview. All for free.

Employer Dashboard

When an employer is signed in, they are able to view their dashboard. This is where they can see all the jobs they have posted, post new jobs and see applicants. It will look something like below.



Microsoft Dashboard

Your postings

Back-End
Our backend engineers develop the core Yammer services that power all our experiences. As a social networking product, Yammer has tons of interesting engineering challenges.
19-06-2022
Posting Link

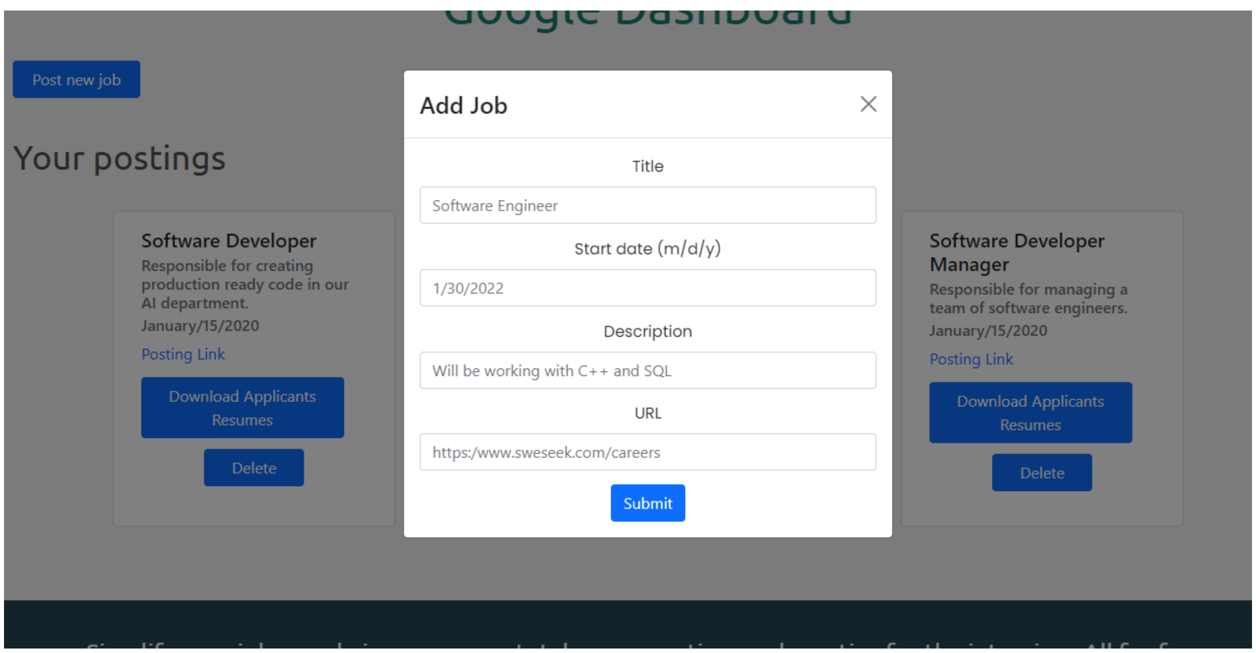
Front-End
The Microsoft Start team is looking for self-motivated and experienced Frontend Engineers to build the best personalized content feed system to connect 1B+ users with the most engaging, informative, and inspiring content they care about.
21-05-2022
Posting Link

Full-Stack
Microsoft 365 Modern Workplace Transformation Insight (M365 MWT-Insight or Viva Insight) helps people and businesses thrive with insights, recommendations and solutions that are integrated into Microsoft Teams and Outlook to improve productivity.
04-07-2022
Posting Link

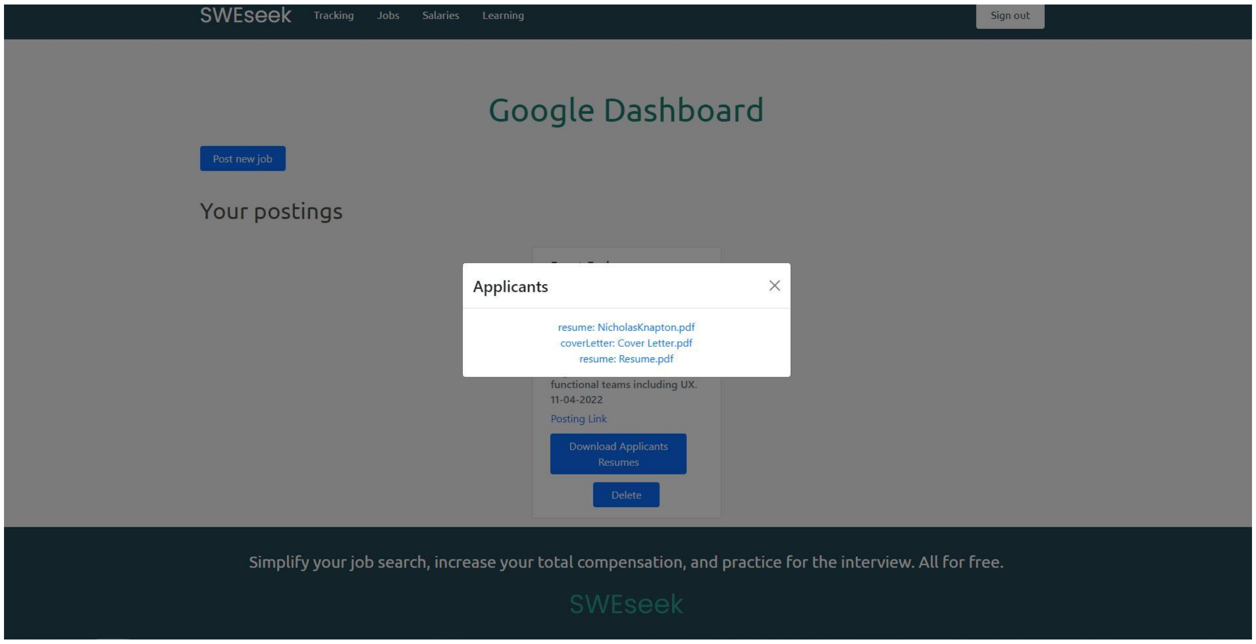
Download Applicants Resumes
Delete
Download Applicants Resumes
Delete
Download Applicants Resumes
Delete

Simplify your job search, increase your total compensation, and practice for the interview. All for free.

To post a new job, companies can click the “Post new job” button and fill out the form shown below to post it.


Post new job
Your postings
Software Developer
Responsible for creating production ready code in our AI department.
January/15/2020
Posting Link
Download Applicants Resumes
Delete
Add Job
Title
Software Engineer
Start date (m/d/y)
1/30/2022
Description
Will be working with C++ and SQL
URL
https:/www.sweseek.com/careers
Submit
Software Developer Manager
Responsible for managing a team of software engineers.
January/15/2020
Posting Link
Download Applicants Resumes
Delete

To get the list of applicants and their resumes/cover letters for a job simply click Download Applicants Resumes, and Select one of the applications to download it. There is an example below.


SWEseek
Tracking
Jobs
Salaries
Learning
Sign out
Google Dashboard
Post new job
Your postings
Applicants
resume: NicholasKnapton.pdf
coverLetter: Cover Letter.pdf
resume: Resume.pdf
functional teams including UX.
11-04-2022
Posting Link
Download Applicants Resumes
Delete
Simplify your job search, increase your total compensation, and practice for the interview. All for free.
SWEseek

A company may also delete a posting by clicking on the “Delete” button.

Running SWEseek Source Code

SWEseek is built with React, Flask and MySQL. The prerequisites to running the application are as follows:

- Nodejs
- Npm (should come with Nodejs)
- Python3

After this clone the source code into desired location and follow the steps below:

1) Open a terminal window and navigate to the "frontend" folder. This can be done by doing "cd frontend" from the project main directory.
2) Run "npm install", this will install all required packages for the frontend of the project.
3) Run "npm start", this will start a development server and should automatically open a tab in your browser at "http://localhost:3000", if nothing opens but the server is running, you should be able to manually navigate there and see the project.
4) Open MySQL workbench and run the "SWESEEK.sql" file located in the "SweSeek Database" folder. This should create a new database called "sweseek" and populate it with some example data.
5) Open the "app.py" file in the "backendPython" folder and on line 13, enter your MySQL connection password.
6) In the "app.py" file change line 16 to correspond to the directory the project is stored in + "/backendPython/resumeStorage" this is to ensure the resumes and cover letters users submit get placed in the right directory. An example is there already.
7) Open a new terminal (Do not close the frontend terminal) and navigate to the "backendPython" folder. This can be done by doing "cd backendPython" from the project directory.
8) Run "pip install" on the following:
    - "pip install flask"
    - "pip install flask_mysqldb"
    - "pip install flask_cors"

9) Run "py ./app.py", the development server should start up.
10) If all worked correctly you should now be able to use the frontend and requests will be processed by the backend which will update the database accordingly.