Team Number: 34
Team Members: Abhiroop Goel,Aryan Ghorpade,Chloe Tran,David Sutherland,Dylan Kneidel, Ibrahim Sufi
Project Name: AI Job search
Synopsis: A web app that allows the user to input a series of skills that they and then returns jobs that matches those skills

Architecture Description:

The project has 4 main components. First there is the web scrapper layer. There are two web scrapes that continuously go to job site listing websites such as linkedin, indeed, glassdoor, etc. and collect lists of jobs including the job description, a link to the job and the role detail. The next layer is the database layer. These jobs are all entered into a sqlite lite database which stores all the information about the jobs as well as the skills corresponding to those jobs. The database is laid out as follows. There is one file called jobs.db this file follows the sql lite schema layout so it functions as a sql lite database. Inside the file there are 3 sql tables. First there is the jobs table that has 3 columns: the url which is a string, the description which is text, and the title which is also a string. Finally each job has an id. Next there is a skills table that has a list of skills each skill has a name and an id. Finally, because there is a many association between jobs and the skills, each job has many skills and each skill has many jobs that it can correspond to. Because of this we need a third table that is called job_skills this table has a job_id and a skill_id in each row which represent the fact that a given job corresponds to a certain skill. Overall these tables provide an interface written in python using the sqlite library that allows fetching all the jobs that match a skill and all the skills that match a job. Additionally there is the ability to add a new skill or a new job to the database.

The third component of the project is the flask app. This app connects to the database and shows in the web app the jobs corresponding to the user's imputed skills. Finally there is a frontend written in html and css that shows a form the user can submit to see the jobs that match their skills and then there is another page that the flask app renders that shows a list of all the pages. This page is dynamic since it will show new jobs if the web scrapper finds new jobs there is the ability to show those additionally on the web page.

This additional functionality is accomplished via using a web socket. On the html page that shows the list of all jobs there is a javascript that creates a socket connection that connections to another route on the flask api this route continuously returns new jobs in a json format that are then turned into html by the js that renders on the page in the same format as the rest of the jobs. One optional dependency in the project is using react to make the frontend end more interactive this will allow the user to temporarily hide jobs that they are not interested in as well as preview jobs they might want to apply to.

UML Diagrams are on the github repo.