Cairo University

Faculty of Computers and Artificial Intelligence

Intelligent Recruiter System

Graduation Project
Academic Year 2021-2022

The Graduation Project Submitted to
The Faculty of Computers and Artificial Intelligence,
Department of Software Engineering
Cairo University
In Partial Fulfillment of the Requirements
for the Bachelor Degree

Presented by

Mazen Fayez 20186021 Hania Osama 20186040

Ibrahim Mustafa 20186037 Heba Hassan 20186048

Supervised by:

Dr Abeer El korany

TA: Ahmed Samir

Table of Contents

List of Figures:

List of tables:

List of Abbreviations:

Abbreviation	Meaning
API	Application Programming Interface
CSS	Cascading Style Sheets
GUI	Graphical User Interface
ERD	Entity Relationship Diagram
App	Application
HTML	Hypertext Markup Language
IOT	Internet Of Things
TF-IDF	Term Frequency-Inverse Document Frequency
TF	Term Frequency
IDF	Inverse Document Frequency

Chapter1: Introduction

C

1.1Motivation:

Intelligent recruiter system is a student platform where students from different fields can find the best questions and answers, interact with them, and participate in weekly competitions. As students we have felt the struggle when trying to apply for jobs/internships, so we have made it fundamental in our website to make it easier for students to be qualified when it comes to recruitment. and to send recommendation notifications to students with ideas about what courses to take according to the real-time market. It will include skills recommendations and jobs recommendations which will include job opportunities that are available to students that are a fit-match for it. Moreover, to make the recruiting selection process in favor to those who rightly deserve it.

Our website facilities the process for students in searching for jobs and to make them receive recommendations regarding the real-time market needs based on their dynamic profiles.

1.2 Problem Definition

Firstly, since we have faced this issue ourselves during our college years, when in need of immediate help or answers to specific questions, our first resort would be searching for it on the internet, and not all websites provide us free access to their information or data.

Secondly, students may not be able to show their full potential unless their behavior and activity has been fully viewed. Most websites focus mainly on the static profile of the student, not on how the student has improved throughout the years or his dynamic behavior.

Thirdly, as students, we have encountered multiple struggles when it came to understanding real-time market needs and what courses or skills to focus on to be more qualified and valued in the work industry, which have led to a gap between students and the real-time market needs. Moreover, we have still not gained skills that will be appreciated in the eyes of the recruiters.

Lastly, recruiters don’t know who is the most qualified for internships or who is the most suitable for the job, especially when it comes to fresh graduates or students who have never been employed before. Some students may have been highly interactive during their college years but not awarded or certified for it.

1.3 Project Objective (Suggested Solution)

The main area of our project is developing a helpful platform for students, or fresh graduates to truly shows their abilities and their real progress throughout their college years.

So, we are intending to build a system that make the student’s profile to be dynamically updated according to his behavior and interactions with other students.

Moreover, badges will be given based on the student’s behavior throughout the activity of the account regarding the questions or the weekly competitions, these badges will be one of the main considerations when it comes to recruiters and their interviews.

Furthermore, due to the importance of internships/jobs to students, which provides them with experience by dealing with real-life projects, in our website it is easier for students to find jobs that match his qualifications. As well, our website will recommend to the recruiters with students with high specific skills regarding the recruiter’s specification.

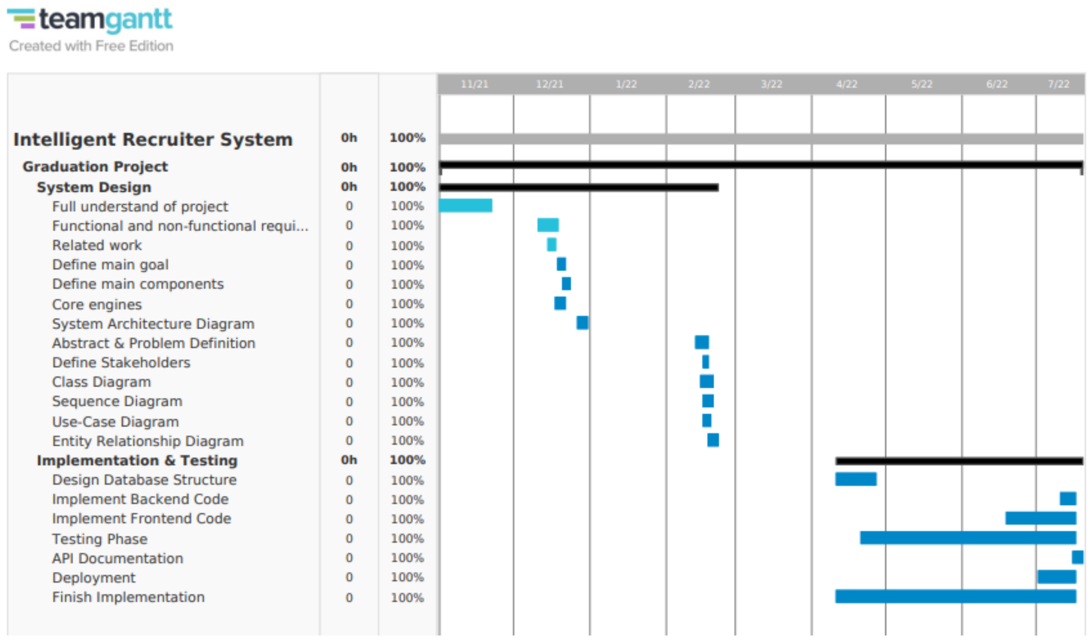
Therefore, Intelligent recruiter system is not only a questions and answers software or a recruitment platform, but both of them combined.

A platform that combines several fields together for students. Combine a question answering platform with a recruitment platform. Develop a platform that consists of three parts, a search engine, a recommendation, and broker engine. A dynamic profile is extracted based on Student’s behavior.

1.4 Gantt chart of project time plan:

TasksID	Tasks	Start Date	End Date	Duration
1	Full understand of project	21/11/2021	11/12/2021	Completed in 3 weeks
2	Functional and non-functional requirements	11/12/2021	18/12/2021	Completed in 1 weeks
3	related work	15/12/2021	19/12/2021	Completed in 4 days
4	Define Main Goal	11/12/2021	31/12/2021	Completed in 20 days
5	Define Main Components	11/12/2021	31/12/2021	Completed in 20 days
6	Core Engines	18/12/2021	31/12/2021	Completed in 13 days
7	System Architecture Diagram	1/1/2022	5/1/2022	Completed in 4 days
8	Abstract & Problem definition	15/2/2022	17/2/2022	Completed in 2 days
9	Define Stakeholders	15/2/2022	16/2/2022	Completed in 1 days
10	Class Diagrams	15/2/2022	16/2/2022	Completed in 1 days
11	Sequence Diagram	16/2/2022	18/2/2022	Completed in 2 days
12	Use-case Diagram	16/2/2022	18/2/2022	Completed in 2 days
13	Entity Relationship Diagram	18/2/2022	19/2/2022	Completed in 1 days
14	Implementation & Testing	6/7/2022	12/7/2022	Completed in 6 days
15	Design Database Structure	1/7/2022	5/7/2022	Completed in 4 days
16	Implement Backend code	1/7/2022	6/7/2022	Completed in 5 days
17	Implement Frontend code	1/7/2022	6/7/2022	Completed in 5 days
18	Testing phase	8/7/2022	13/7/2022	Completed in 6 days
19	API Documentation	25/6/2022	3/7/2022	Completed in 8 days
20	Deployment	1/7/2022	6/7/2022	Completed in 5 days
21	Finish Implementation	6/7/2022	12/7/2022	Completed in 5 days

Table 1: Gant Chart



Intelligent Recruiter System	**0h**	**100%**
Graduation Project	**0h**	**100%**
System Design	**0h**	**100%**
Full understand of project	0	100%
Functional and non-functional requi...	0	100%
Related work	0	100%
Define main goal	0	100%
Define main components	0	100%
Core engines	0	100%
System Architecture Diagram	0	100%
Abstract & Problem Definition	0	100%
Define Stakeholders	0	100%
Class Diagram	0	100%
Sequence Diagram	0	100%
Use-Case Diagram	0	100%
Entity Relationship Diagram	0	100%
Implementation & Testing	**0h**	**100%**
Design Database Structure	0	100%
Implement Backend Code	0	100%
Implement Frontend Code	0	100%
Testing Phase	0	100%
API Documentation	0	100%
Deployment	0	100%
Finish Implementation	0	100%

Figure 1:1.4. Gant Chart

1.5 Project development methodology:

The development methodology followed in this project is the agile methodology principles, as we manage our project by breaking it up into several phases and continuous improvement at every stage, and the client-server architecture.

Languages Used:

Client-side: HTML, CSS, JavaScript, ReactJs.

Server-side: NodeJS, Python to build the model and the main engines.

Database: MongoDB.

1.6 The used tools in the project (SW and HW):

Libraries:

- TensorFlow used in developing and training our model.

- Keras is an open-source software library.

- Sklearn: for machine learning in Python.

- Nltk library for preprocessing.

- Numpy.

- Pandas for the Python programming language.

- React is a JavaScript library for front-end in implementation.

- Express is Nodejs library for server-side API implementation.

- Flask API is used for exposing our functionality as web services

Software Used:

- Visual Studio Code used for python code.
- Postman to test the API’s.
- Microsoft excels.

Hardware:

windows laptops, desktops

1.7 Report Organization (summary of the rest of the report)

- In chapter 2: we are going to talk about the related work of our project and define the main difference between them.
- In chapter 3: We will talk about the system analysis the project specification (discuss the functional and non-functional requirements) and the use cases we have and represent them in a use case diagram.
- In chapter 4: comprises a design review which aims to give the reader requisite background knowledge for understanding the project through a group of structural diagrams (system's object, components, actors and operations) and show the relations between them:
  - I. System Architecture Diagram and discuss each component briefly
  - II. System Component Diagram
  - III. System Class Diagrams
  - IV. Sequence Diagrams
  - V. Project Entity Relation Diagram
  - VI. System GUI Design
- In chapter 5: lastly, we will go through the implementation of our system, testing process in detail showing multiple scenarios that are fully explained with test cases.

Chapter 2: Related Work

Features	Our system	hacker rank	Quora	stack overflow	algo expert	LinkedIn
Student can ask a question.	T	F	T	T	F	F
Student can search for questions.	T	T	T	T	T	F
Student can edit question and answers.	T	F	T	T	F	F
Student can participate in Weekly competition.	T	T	F	F	T	F
Students can get a badge.	T	T	F	T	T	F
Student can search for jobs.	T	F	F	F	F	T
Student can to choose a specific topic interested in.	T	T	T	F	F	T
Students don't have limited search trial	T	T	T	T	T	F
Interactions. (Ex: likes...)	T	F	T	T	F	T
Courses recommended to the student.	T	F	F	F	T	T
Feed page for student.	T	F	T	F	F	T
Recruiter can contact student.	T	F	F	T	F	T
Recruiter can view students' profiles including their answers.	T	F	F	T	F	T
Recruiters can search for a specific student.	T	F	F	F	F	T
Recruiters can search for applicable students with specific skills.	T	F	F	F	F	T
Help.	T	T	T	T	T	T
Get notification.	T	T	T	T	T	T
Career tips section.	T	T	F	F	T	F
Broker engine.	T	F	F	F	F	T
Search engine.	T	T	T	T	T	T
Recommendation engine.	T	F	F	F	T	F

Table 2: Related work

2.1 Stack Overflow:

One of the best Question & Answer websites nowadays.

Stack Overflow is a question-and-answer website for professional and enthusiast programmers. It is the flagship site of the Stack Exchange Network. It features questions and answers on a wide range of topics in computer programming. It was created to be a more open alternative to earlier question and answer websites such as Experts-Exchange. Stack Overflow was sold to Prosus, a Netherlands-based consumer internet conglomerate, on 2 June 2021 for $1.8 billion.

The website serves as a platform for users to ask and answer questions, and, through membership and active participation, to vote questions and answers up or down in a similar way to Reddit and edit questions and answers in a fashion similar to Wikipedia. Users of Stack Overflow can earn reputation points and badges, for example, a person is awarded 10 reputation points for receiving an "up" vote on a question or an answer to a question, and can receive badges for their valued contributions, which represents a gamification of the traditional Q&A website. Users unlock new privileges with an increase in reputation like the ability to vote, comment, and even edit other people's posts.[1]

Our system differs from this website because Stack Overflow only accepts questions about programming that are highly focused on a specific problem not on all fields. Questions of a broader nature—or those inviting answers that are inherently a matter of opinion—are usually rejected by the site's users and marked as closed. The sister site softwareengineering.stackexchange.com is intended to be a venue for broader queries, e.g., general questions about software development. [2]

Searching by jobs is not provided by Stack Overflow.

Moreover, our website introduces a recruitment system which is provided mainly by a Broker engine. Our main goal is to help students find jobs easily, Helping the recruiters to find employees is a really hard process since recruiters need to know how the students think and how much they are helpful for other people in order to employ the student and choose him for a specific job. This can be done through the dynamic behavior obtained by our system using the student's questions and answers.

Furthermore, Stack Overflow has no recommendation engine for its users.

This is a main component of our website since it helps students with improving their skills regarding a specific topic based on their skills and other similar students who match the market needs. This engine will help the students to be aware of the most required skills in the real-time market and recommend them with jobs according to their gained skills.

Finally, the concept of competitions is not available in Stack Overflow, we think it is a great way to encourage students to learn more in different topics in their field, as the competition winners will get badges which will help when being ranked to the recruiters using the broker engine.

2.2 HackerRank:

HackerRank is a tech company that focuses on competitive programming challenges for both consumers and businesses. Developers compete by trying to program according to provided specifications. HackerRank's programming challenges can be solved in a variety of programming languages (including Java, C++, PHP, Python, SQL, JavaScript) and span multiple computer science domains.[3]

On the consumer side, when a programmer submits a solution to a programming challenge, their submission is scored on the accuracy of their output. Programmers are then ranked globally on the HackerRank leaderboard and earn badges based on their accomplishments to drive competition among users. In addition to individual coding challenges, HackerRank also hosts contests (often referred to by HackerRank as "CodeSprints") where users compete on the same programming challenges during a set period of time and are then ranked at the conclusion of the event. HackerRank is part of the growing gamification trend within competitive computer programming[4] and the consumer-side of their website is free for coders to use.

HackerRank is a technology hiring platform that is the standard for assessing developer skills for over 2,800+ companies around the world. By enabling tech recruiters and hiring managers to objectively evaluate talent at every stage of the recruiting process, HackerRank helps companies hire skilled developers and innovate faster.

The journey to HackerRank first began in July 2009, when computer science graduates Vivek and Hari worked at Amazon and IBM (respectively) for about a year in Bangalore. Both were on interview panels, inundated with endless hours of resume reviews, phone screens, and onsite interviews. Unqualified people were making it to the onsite, while some of their most capable, hardworking friends were passed on because of their GPA or lack of a prestigious degree.

While HackerRank is great and helpful in the recruitment process by encouraging students and competitors to compete with each other. Students are ranked based on competitions, code sprints, and tasks, but they won’t learn anything new since it focuses mainly on recruitment. Our system provides interaction between students, which is something not available by this website.

They interact with each other by asking and answering each other’s questions. We encourage students to helps each other, we are learning more and more every day and that makes us more productive and creative.

HackerRank’s ranking strategy differs from our system because they do not look for the student behavior, they look for the competitions and code sprints results, and rank based on them. In our System, we rank the students based on their interactions too other than the competitions, since we are looking forward to building a more collaborative community.

2.3 LinkedIn:

LinkedIn is a social networking site designed specifically for the business community. The goal of the site is to allow registered members to establish and document networks of people they know and trust professionally.

A LinkedIn member’s profile page, which emphasizes skills, employment history and education, has professional network news feeds and a limited number of customizable modules. Basic membership for LinkedIn is free. Network members are called “connections.” Unlike other free social networking sites like Facebook or Twitter, LinkedIn requires connections to have a pre-existing relationship.

With basic membership, a member can only establish connections with someone he has worked with, knows professionally (online or offline) or has gone to school with. Connections up to three degrees away (see six degrees of separation) are seen as part of the member's network, but the member is not allowed to contact them through LinkedIn without an introduction. Premium subscriptions can be purchased to provide members with better access to contacts in the LinkedIn database

This system is highly growing up, more functionalities are added every couple of weeks.

The recommendation engine in LinkedIn is similar to ours in one way, but we provide skills recommendations too to help students improve their skills in the most effective way based on the real-time market needs. Moreover, there are no weekly competitions in LinkedIn.

Although the search engine in LinkedIn is very useful and helpful for all users, but the basic and important questions you need help with while you are learning will not be available, but if you are looking for research or videos to understand a specific topic it will provide the help needed.

Since there are no questions and answers for the users, the recruiters can’t actually know how the user’s thinking or how creative he is. Therefore, recruitment is based on only the static skills entered by the user. We can neither confirm nor deny that it is a good method to help with recruitment. But we think that if the recruiter has access to the user questions and answers, exploring them will gave a rate for the user in the recruiter’s point of view.

Chapter 3: System Analysis

3.1 Project specification

3.1.1. Functional requirement

- ✓ Student will register and then login with his registered account after entering all required data and the password must to be unique.
- ✓ Student will be able to choose a specific topic interested in.
- ✓ Student will be able to update his profile when needed (courses finished, certificates, Internships),
- ✓ Student will be able to add, edit, or delete questions/answers.
- ✓ Student can participate in a weekly competition provided by our website.
- ✓ Student can get a badge by reaching specific number of points or by winning in the weekly competitions.
- ✓ Student will have a profile of his own questions, points and badges if exists.
- ✓ Students with specific badges will receive notifications regarding job internships.
- ✓ Student can earn points by effective interacting on another student's questions.
- ✓ Student will receive help provided by our team for support and give us feedback
- ✓ Student who interacts with an answer will get a notification when the question or answer is updated.
- ✓ Career tips section that will include tips and advices on interviews resume and cover letters and career management.
- ✓ Student can also search for jobs by adding the specifications needed such as the skills, titles, and company if needed.
- ✓ Student will be able to interact to another student question(like-answer)
- ✓ Recruiters will register, login and then start entering their specifications that they are searching upon and will receive the list of students returned by our broker engines after matching the specifications entered with the students of the website.
- ✓ Recruiter can contact students by their e-mail in their profiles.
- ✓ Recruiter will choose a topic after log in.
- ✓ Recruiter can search for students by Tags or filters and then a broker start matching such specifications with applicable student profiles.
- ✓ Recruiter will have another feed page different from other students' pages (it will contain a list of students with their badges ascendingly by their points).
- ✓ Recruiter can view student profile and view his questions and answers.
- ✓ Our website will adapt itself easily depending on the type of student logging in. either a recruiter by showing it a completely different UI than a student, vice versa.

3.1.2. Non-functional requirement

- ✓ **Performance:**

Reasonable response time.

- ✓ **Scalability:**

System will perform well under an increased or expanding workload or scope.

- ✓ **Reliability:**

System consistently performs the specified functions without failure.

- ✓ **Security and privacy:**

Using a secure database (MongoDB) to store the student's information to prevent unauthorized access.

- ✓ **Usability:**

Student-friendly as it is simple and straightforward and easy to use and learn.

- ✓ **Availability:**

System is able to function during normal operating times.

- ✓ **Portability:**

Website will be available on all browsers and operating systems.

3.1.3. Stakeholders:

- ➢ Students
- ➢ Recruiters
- ➢ Admins

3.2. Use case Diagrams

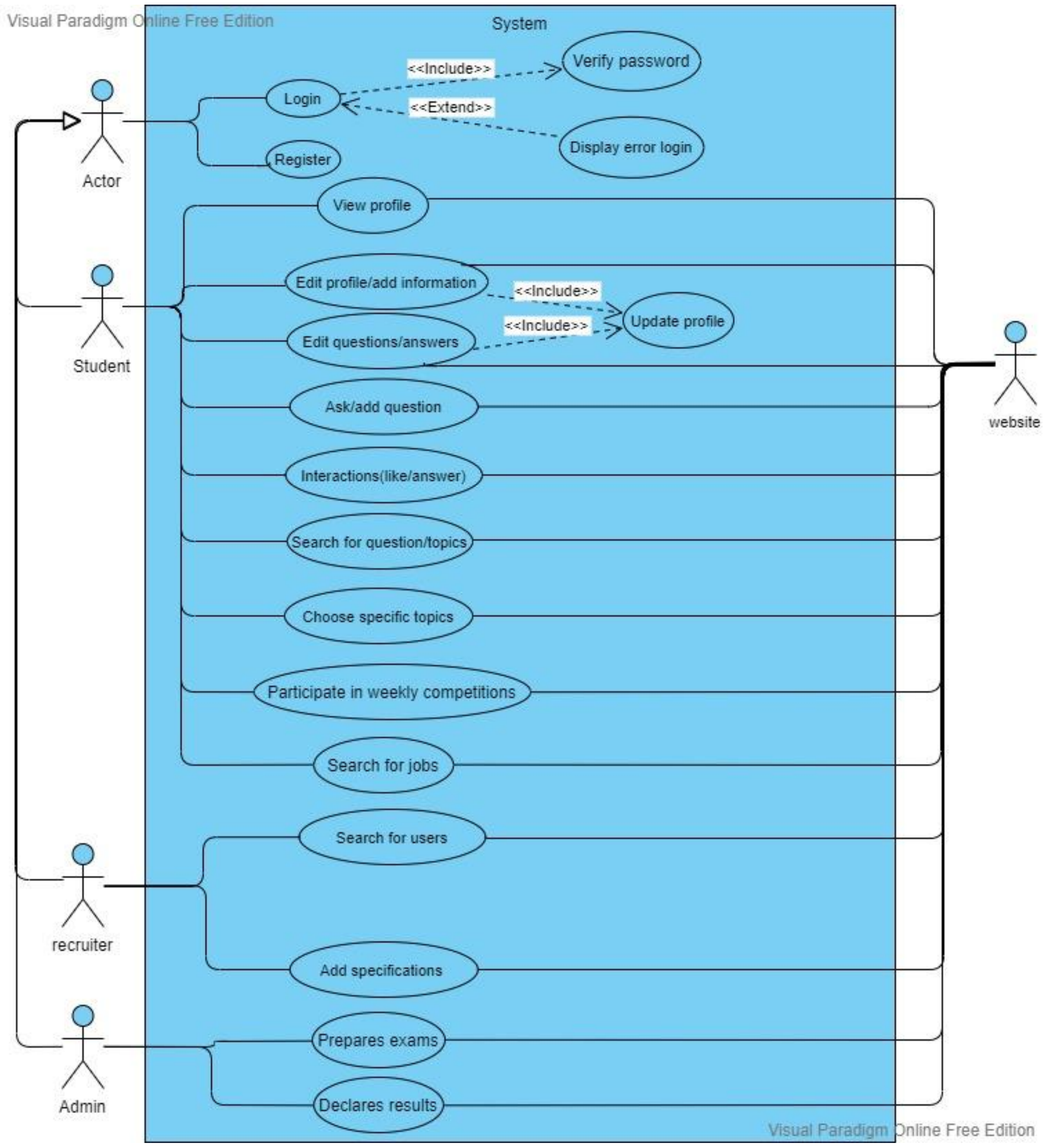


Figure 2: 3.2.Use Case Diagram

Chapter 4: System Design

4.1 System Architecture design

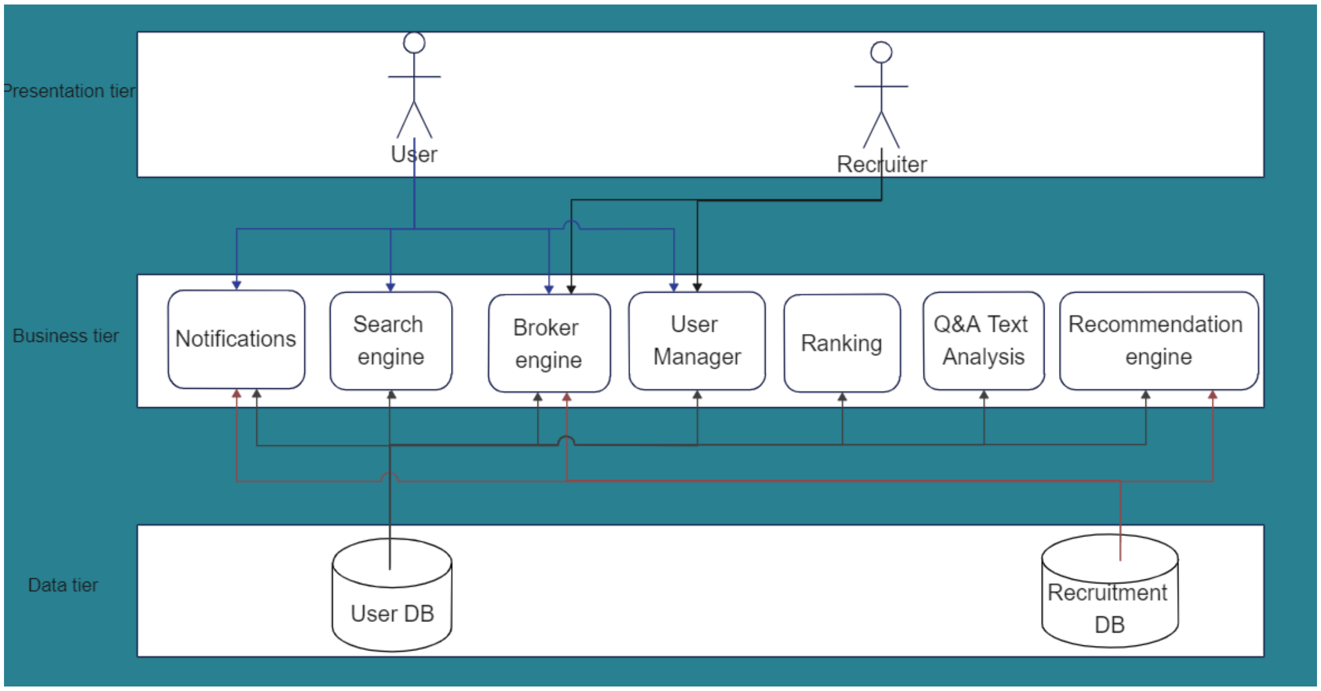


Figure 3: System architecture system

Broker engine:

- Matches what the recruiter is looking for from the available students.

Search engine:

- Student can use it to search a specific tag or keyword regarding the questions.
- Students can search for jobs by titles, skills, or company.
- Recruiter can also use it to search for specific characteristics or values in a student.

Recommendation engine:

- Recommend the students with missing qualifications or skills that if gained will match the real-time market needs.
- Recommend the recruiters with students with high specific skills regarding the recruiter's specification.

Student Manager:

I. Static:

- All information the student enters when creating the account.
- Includes personal information and skills or certificates achieved before making the account.

II. Dynamic:

- Badges earned by students are added to his profile.
- Skills and information gathered by the behavior of the student throughout the activity of the account.

Ranking:

- The list of students outputted from the broker engine will be ranked according to the specifications entered by the recruiter. (Highest match to Lowest match)
- The list of jobs outputted to a student after making a search for a job with specific skills, titles will be ranked according to such specifications. (Highest match to Lowest match)

Notifications:

- Students will receive notifications when getting job offers.
- Students will receive notifications when another student interacts with their question or answer (an answer or a like has been added).
- Students will receive notifications when recommendations are sent to them.
- Recruiters will receive notifications when a match is made.

Q&A Text Analysis:

Text analysis will be performed on questions and answers so that they are classified into their related topics.

4.3 System Component Diagram

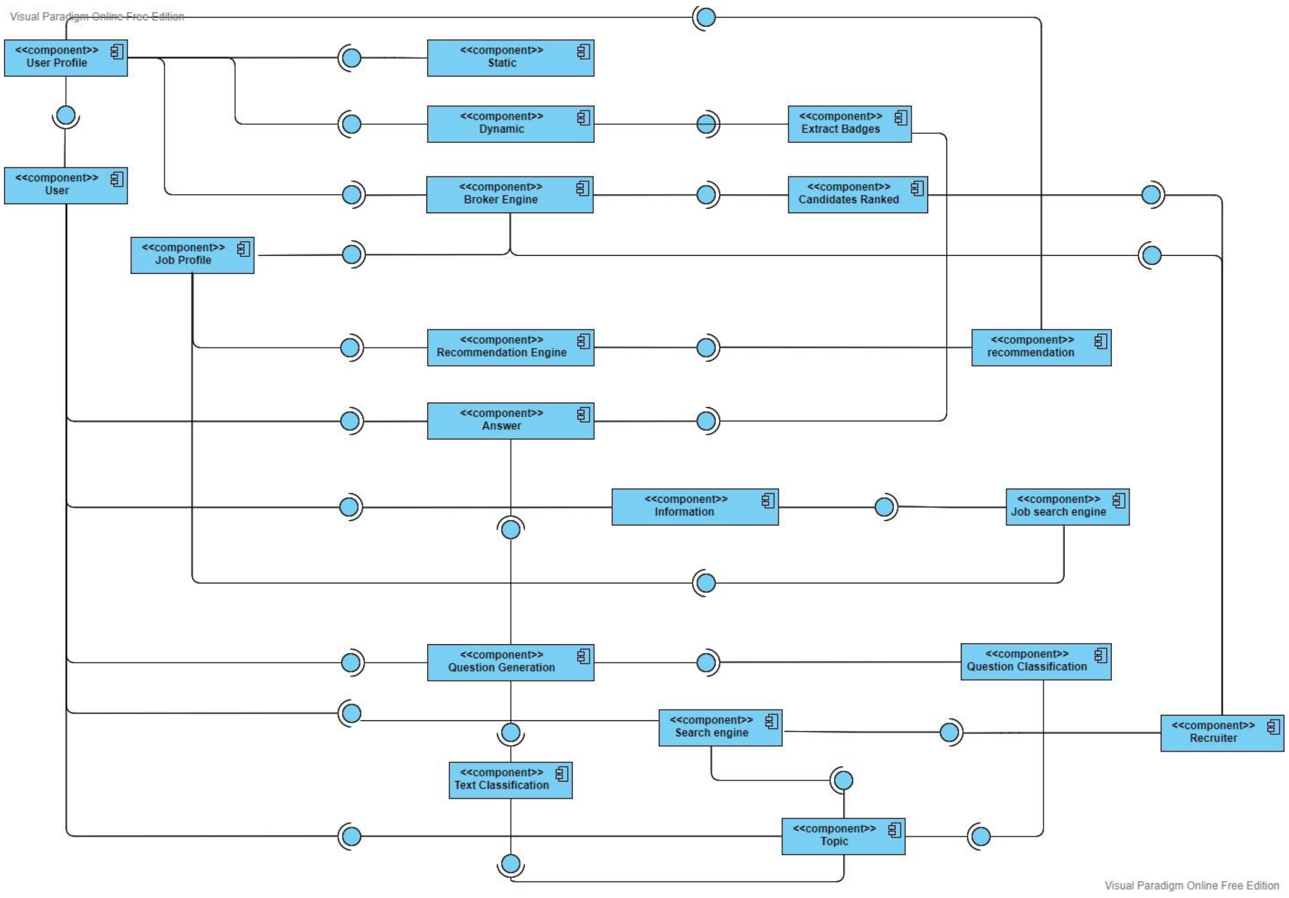

Visual Paradigm Online Free Edition
<<component>> User Profile
<<component>> Static
<<component>> Dynamic
<<component>> Extract Badges
<<component>> User
<<component>> Broker Engine
<<component>> Candidates Ranked
<<component>> Job Profile
<<component>> Recommendation Engine
<<component>> recommendation
<<component>> Answer
<<component>> Information
<<component>> Job search engine
<<component>> Question Generation
<<component>> Question Classification
<<component>> Search engine
<<component>> Recruiter
<<component>> Text Classification
<<component>> Topic
Visual Paradigm Online Free Edition

Figure 4: System Component diagram

4.3 System Class Diagrams

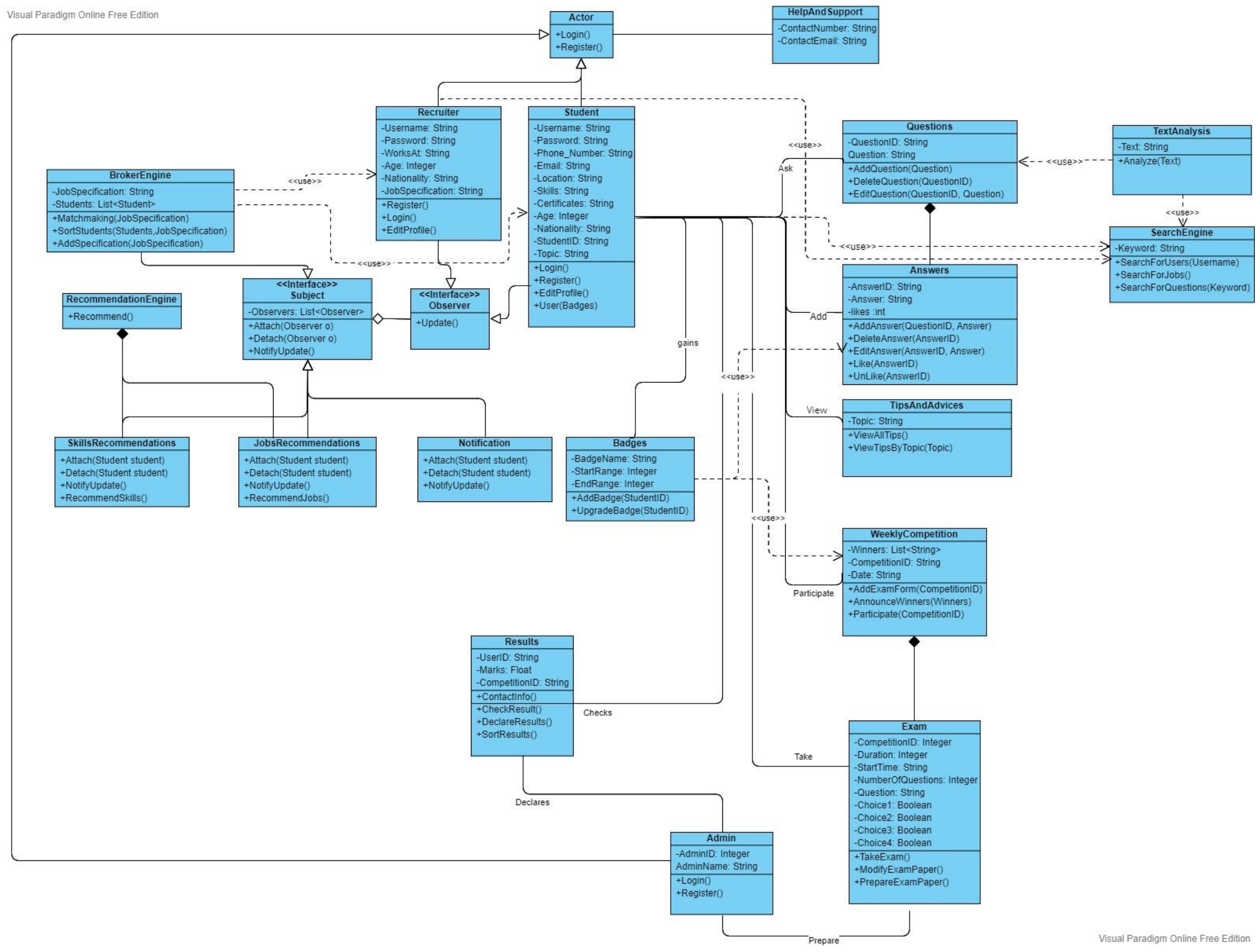


Figure 5: System Class Diagram

4.4 Sequence Diagrams

4.4.1. Sequence 1:

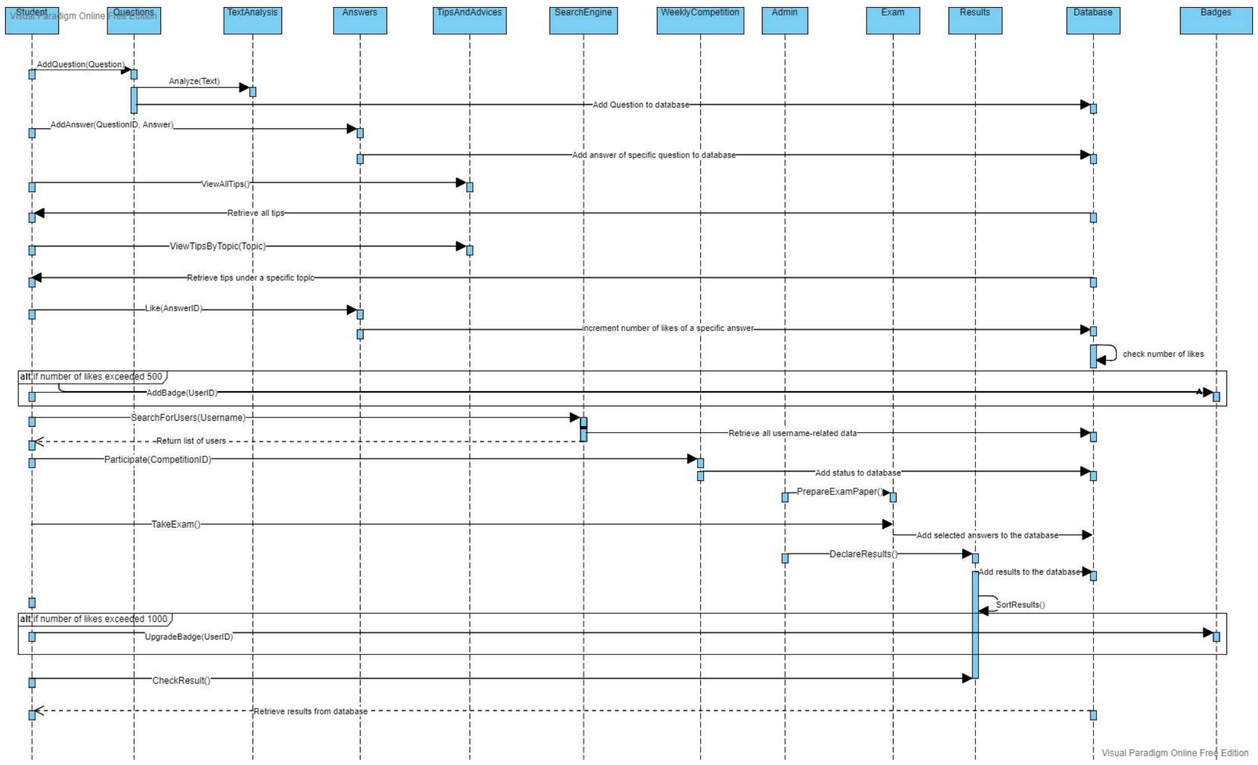

Student
Questions
TextAnalysis
Answers
TipsAndAdvices
SearchEngine
WeeklyCompetition
Admin
Exam
Results
Database
Badges
Visual Paradigm Online Free Edition
AddQuestion(Question)
Analyze(Text)
Add Question to database
AddAnswer(QuestionID, Answer)
Add answer of specific question to database
ViewAllTips()
Retrieve all tips
ViewTipsByTopic(Topic)
Retrieve tips under a specific topic
Like(AnswerID)
increment number of likes of a specific answer
check number of likes
alt if number of likes exceeded 500
AddBadge(UserID)
SearchForUsers(Username)
Retrieve all username-related data
Return list of users
Participate(CompetitionID)
Add status to database
PrepareExamPaper()
TakeExam()
Add selected answers to the database
DeclareResults()
Add results to the database
SortResults()
alt if number of likes exceeded 1000
UpgradeBadge(UserID)
CheckResult()
Retrieve results from database
Visual Paradigm Online Free Edition

Figure 6: System Sequence Diagram 1.

4.4.2. Sequence 2:

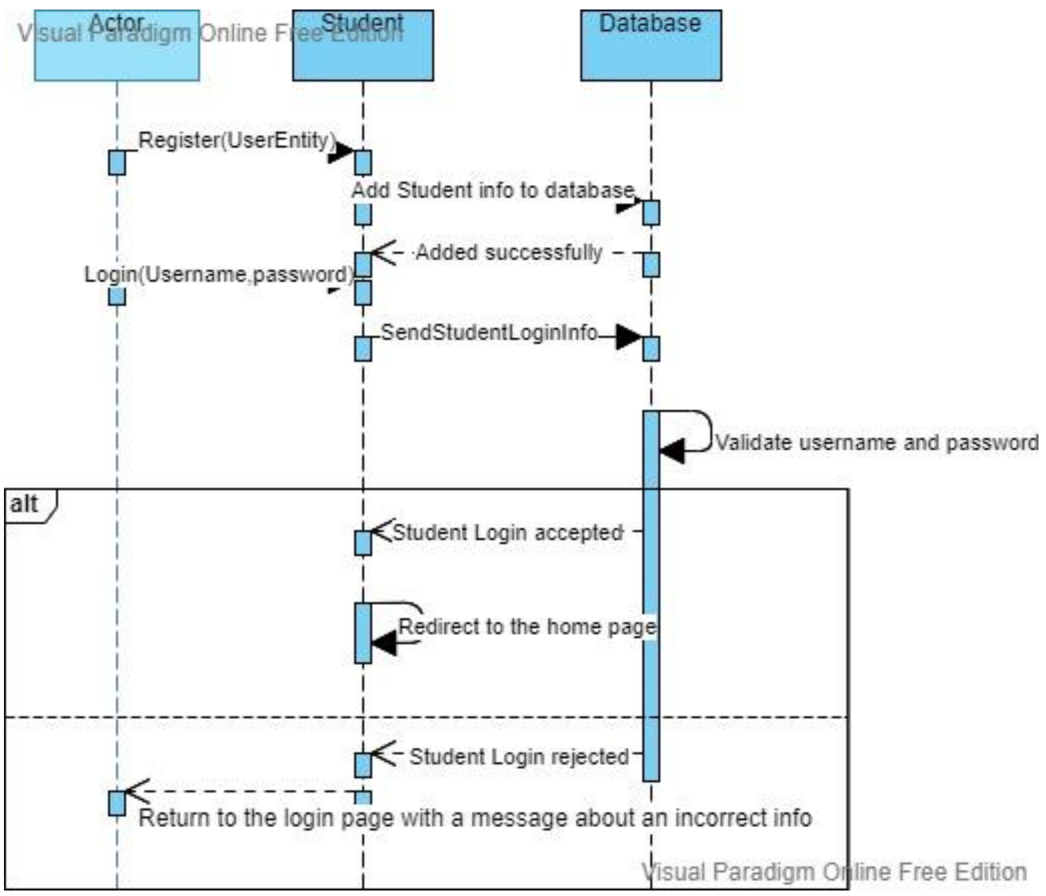

Actor
Student
Database
Visual Paradigm Online Free Edition
Register(UserEntity)
Add Student info to database
Added successfully
Login(Username,password)
SendStudentLoginInfo
Validate username and password
alt
Student Login accepted
Redirect to the home page
Student Login rejected
Return to the login page with a message about an incorrect info
Visual Paradigm Online Free Edition

Figure 7: Sequence Diagram 2

4.4.3. Sequence 3:

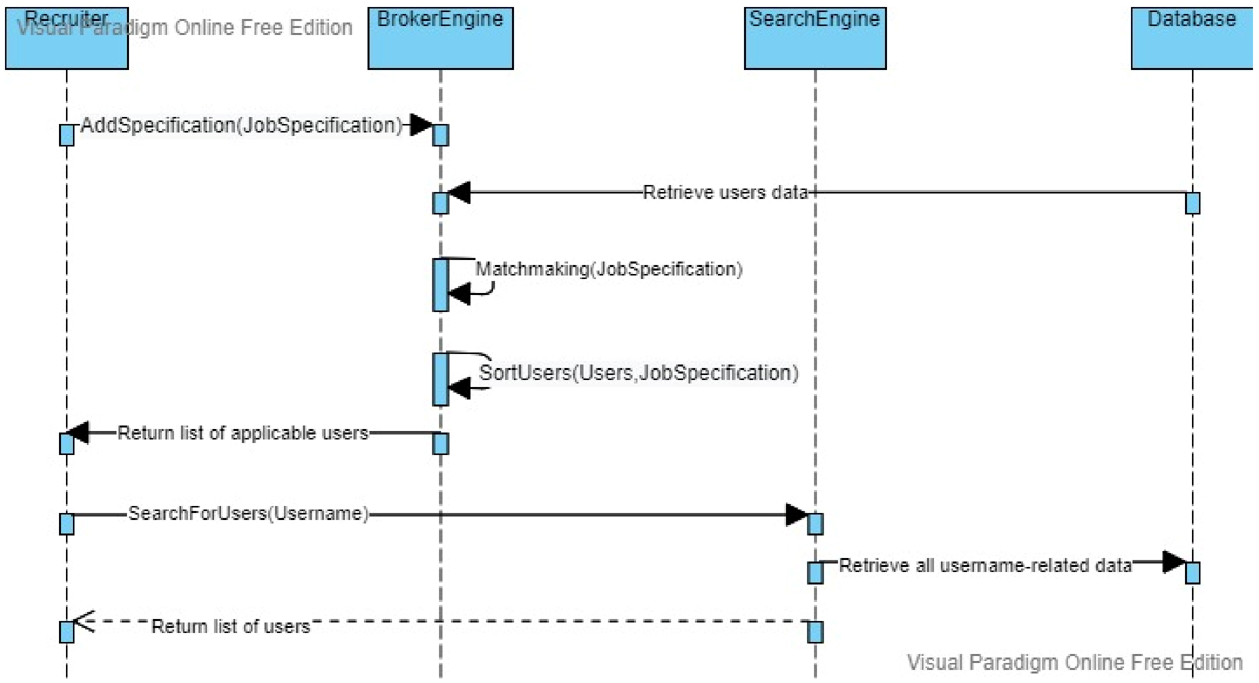

Recruiter
BrokerEngine
SearchEngine
Database
AddSpecification(JobSpecification)
Retrieve users data
Matchmaking(JobSpecification)
SortUsers(Users,JobSpecification)
Return list of applicable users
SearchForUsers(Username)
Retrieve all username-related data
Return list of users
Visual Paradigm Online Free Edition
Visual Paradigm Online Free Edition

Figure 8: System Sequence Diagram 3.

4.5 Project ERD

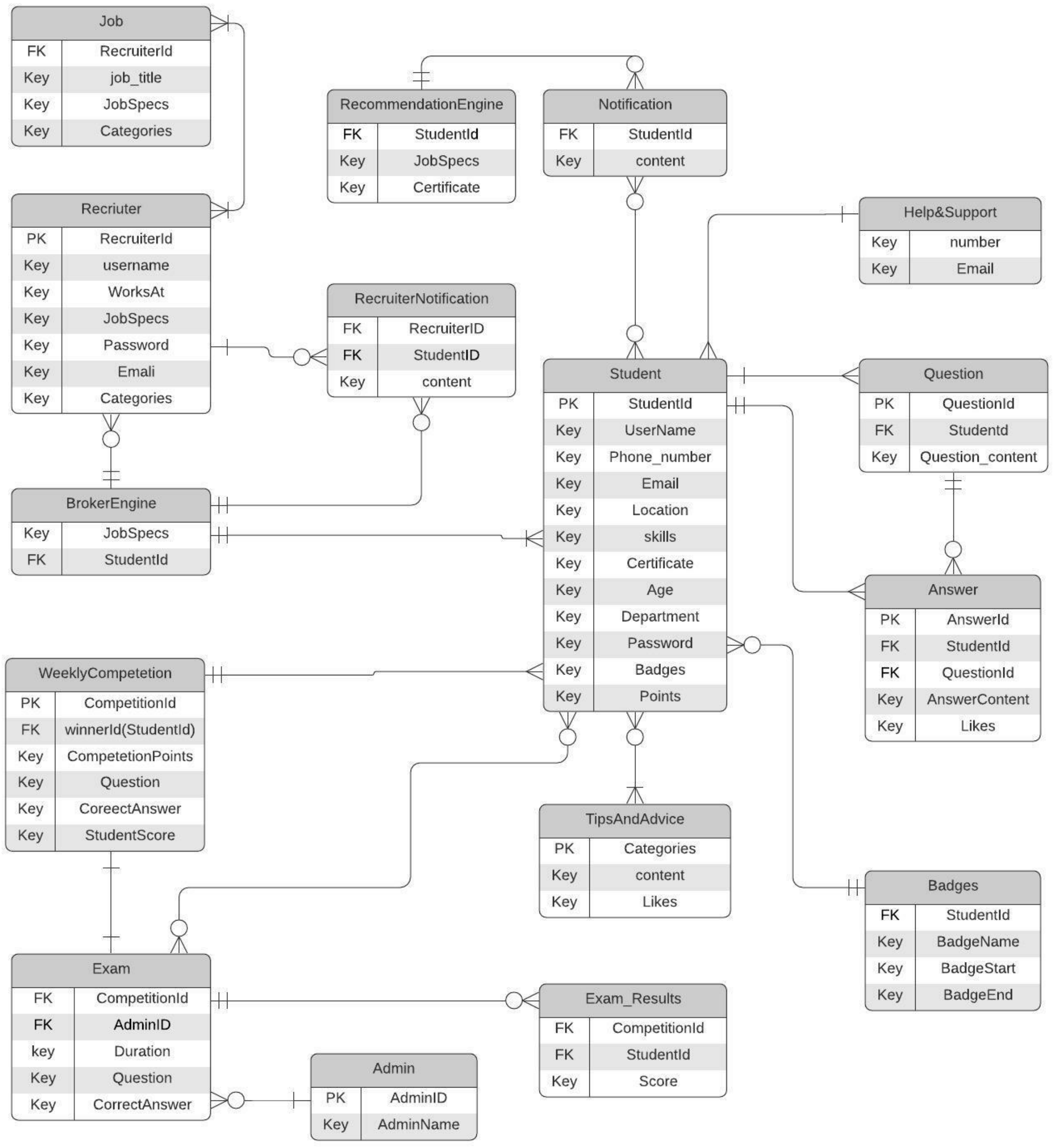
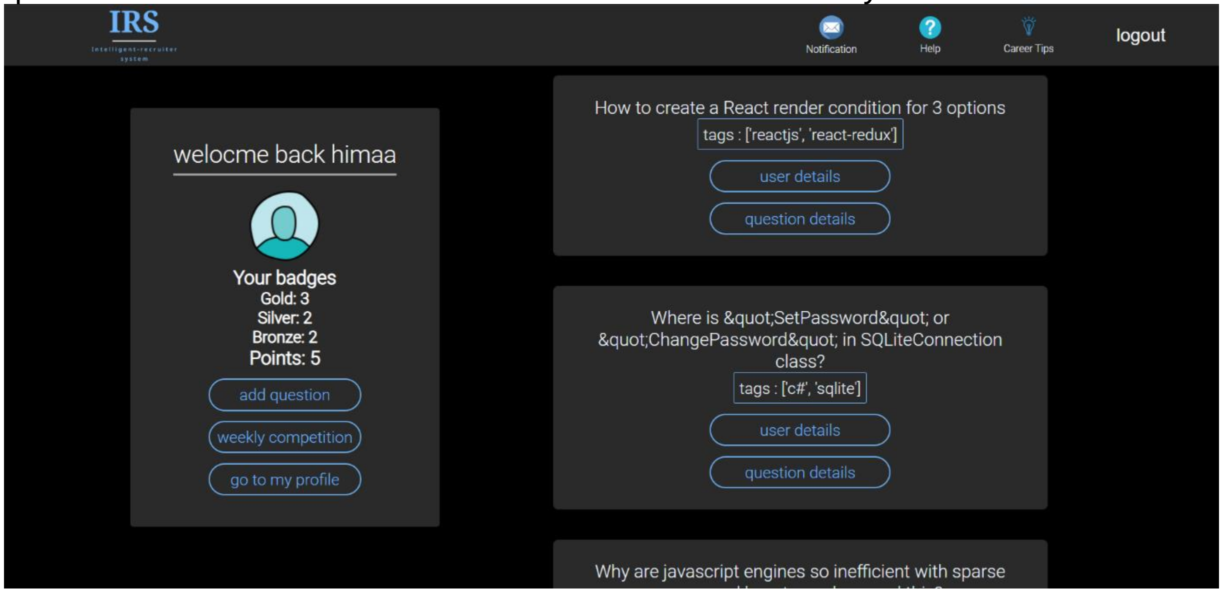


Figure 9: Entity relation diagram

4.6 System GUI Design

Home Page:

for Student: when a student logs in, he will be able to see his feed page of the asked questions and will be able to see a shortcut of his profile which links to his profile page, weekly competition, and add question on the left side. On the other side we have questions that have been asked by other students.



welocme back himaa

Your badges
Gold: 3
Silver: 2
Bronze: 2
Points: 5

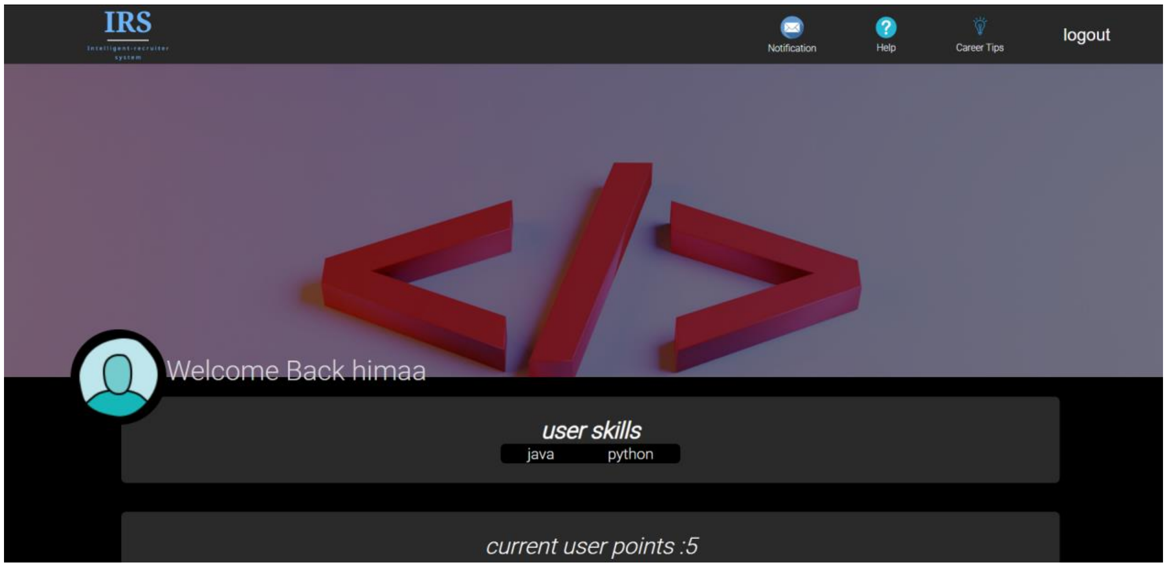
How to create a React render condition for 3 options
tags : ['reactjs', 'react-redux']

Where is &quot;SetPassword&quot; or &quot;ChangePassword&quot; in SQLiteConnection class?
tags : ['c#', 'sqlite']

Why are javascript engines so inefficient with sparse

Profile Page:

contains student’s skills, his badges with the points gained, and questions that have been asked by him.



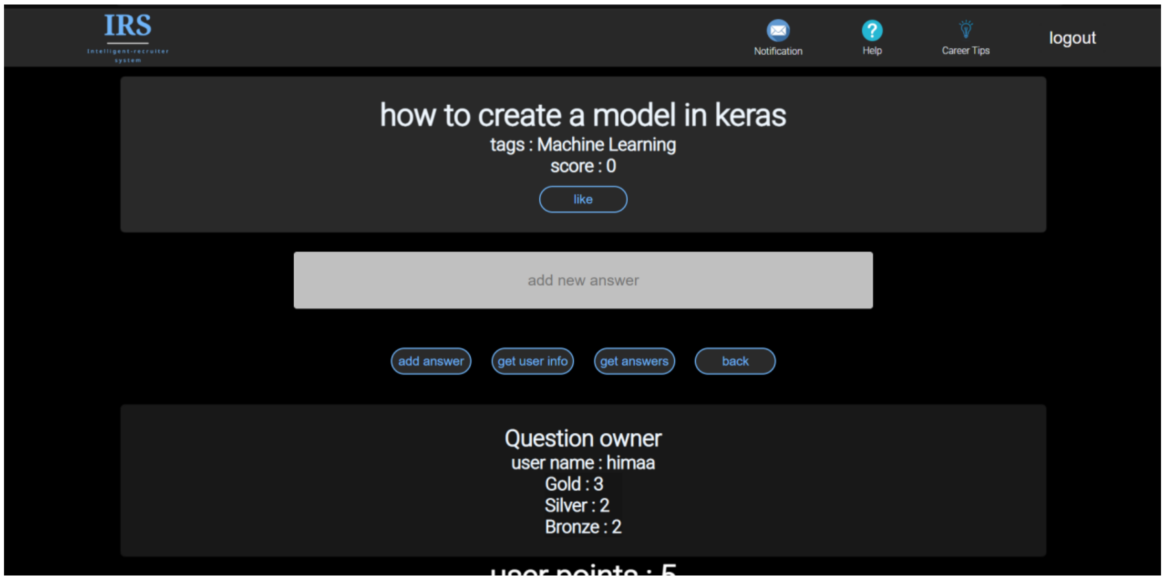
Welcome Back himaa

user skills
java python

current user points :5

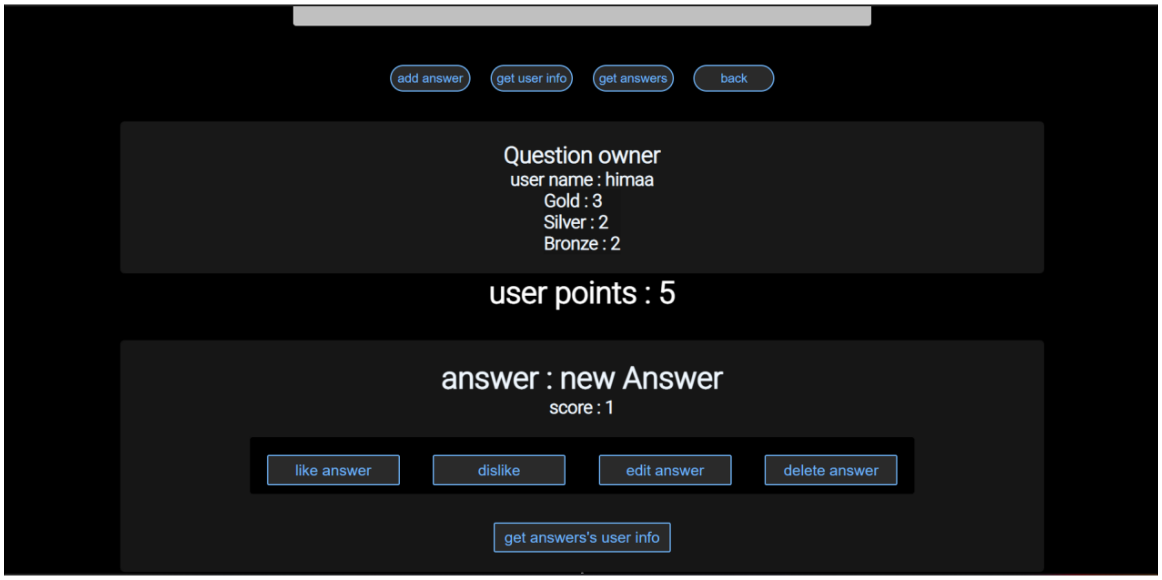
Question and Answer Page:

In this page, the student can find a question and the corresponding answers and can add like, dislike, edit, and delete answers but only if he is the answer's owner.



how to create a model in keras
tags : Machine Learning
score : 0

Question owner
user name : himaa
Gold : 3
Silver : 2
Bronze : 2



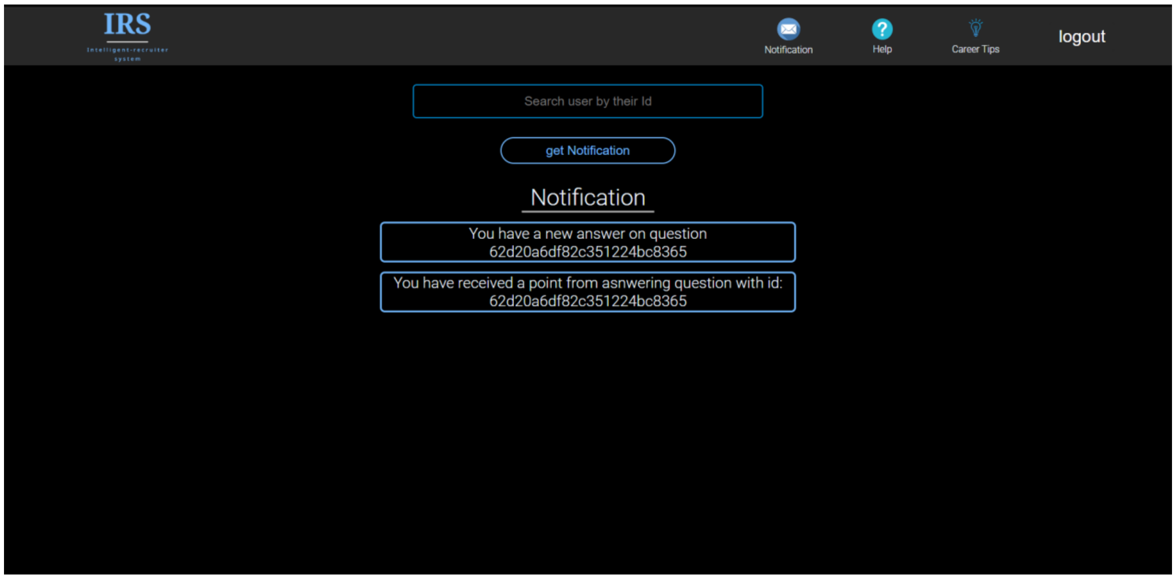
Question owner
user name : himaa
Gold : 3
Silver : 2
Bronze : 2

user points : 5

answer : new Answer
score : 1

Notifications page:

In this page, the student can view his notifications and can also search for the question by its notified id, student receives a notification when he gets a like on an answer, or an answer is added to a question that he owns.



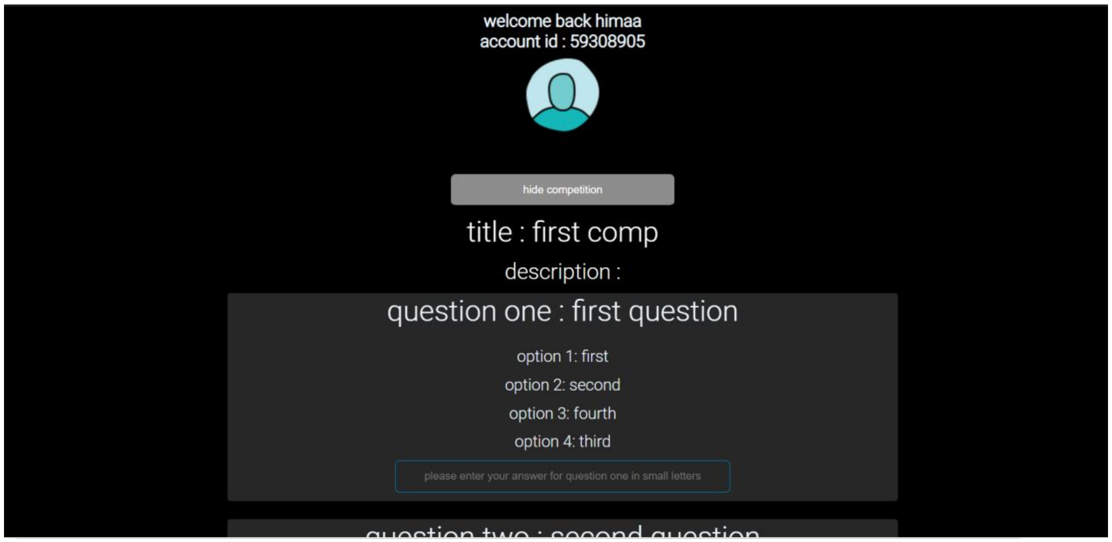
Notification

You have a new answer on question 62d20a6df82c351224bc8365

You have received a point from asnwering question with id: 62d20a6df82c351224bc8365

Competition Page:

students can attend the weekly competition only once, then it will be blocked for them, only admins can add the competition's questions.



welcome back himaa
account id : 59308905

title : first comp

description :

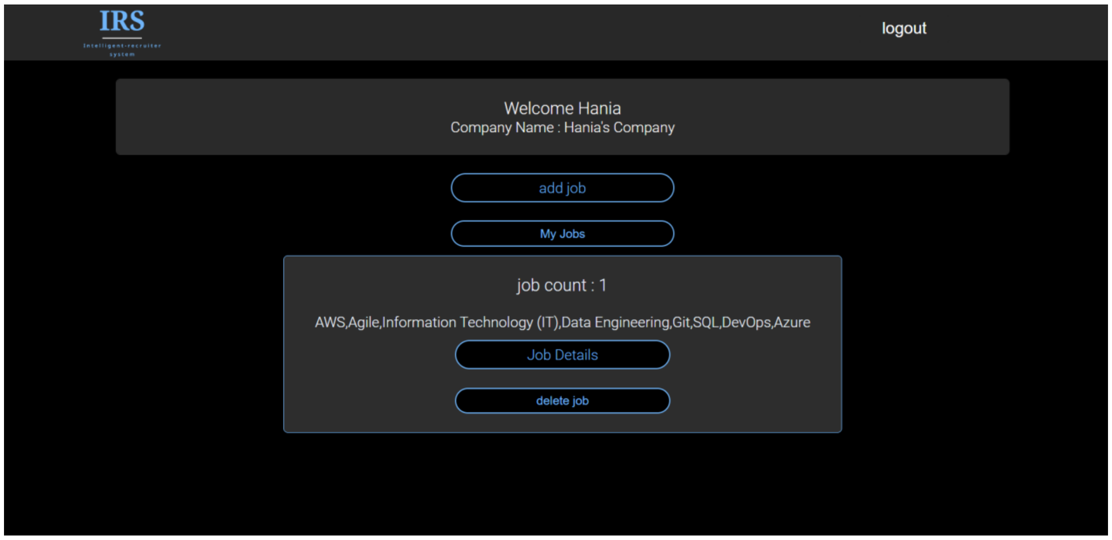
question one : first question

option 1: first
option 2: second
option 3: fourth
option 4: third

question two : second question

Recruiter's Home Page:

Recruiters after Login or register can find this page, they can add a new job, or view a list of his previous job offers, then he can navigate to the job details or delete that job.



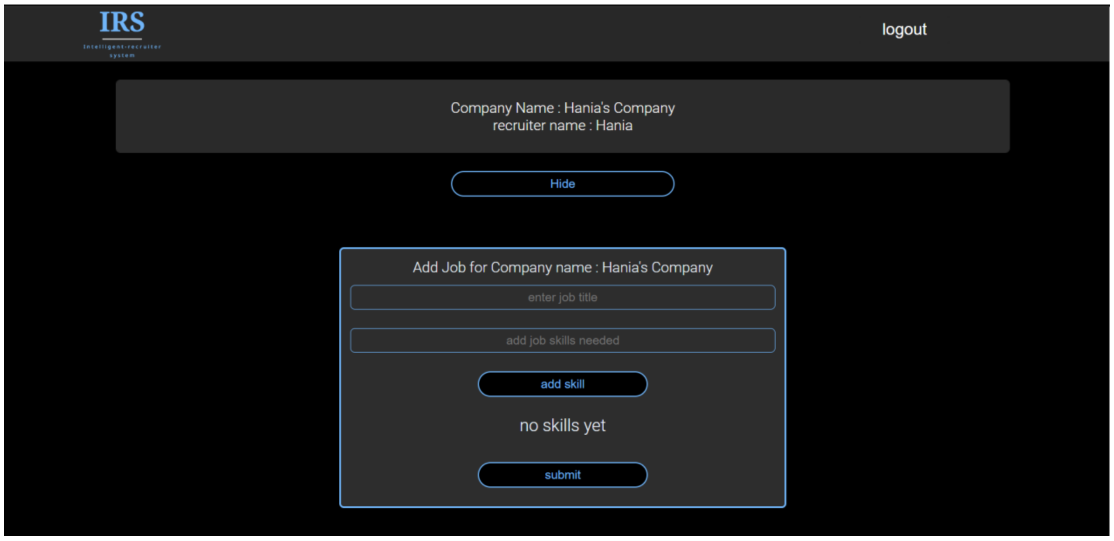
Welcome Hania
Company Name : Hania's Company

job count : 1

AWS,Agile,Information Technology (IT),Data Engineering,Git,SQL,DevOps,Azure

Recruiter adding new Job:

Recruiter can add a new job with a job title and a job description or skills needed for such job.



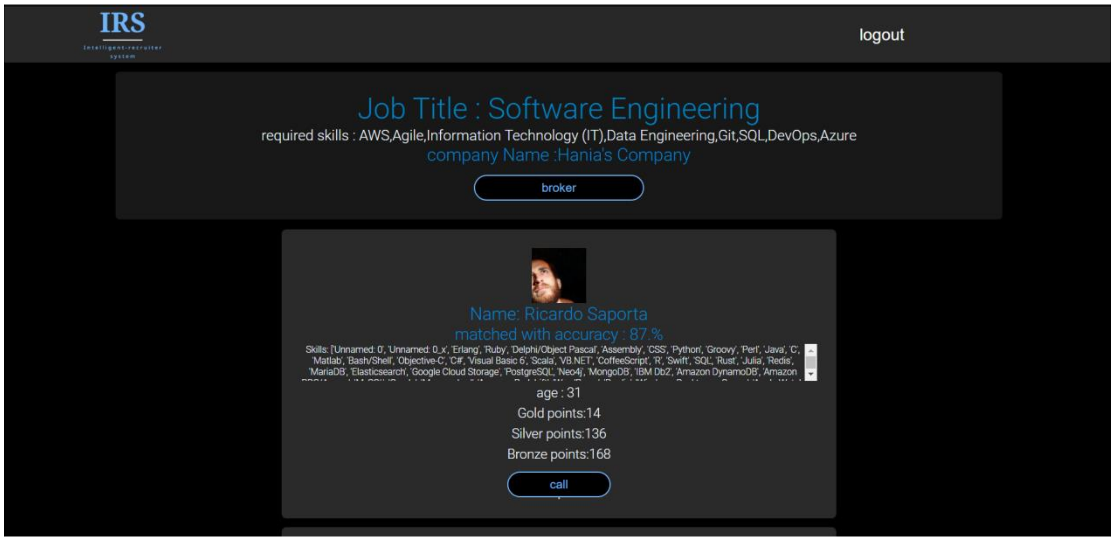
Company Name : Hania's Company
recruiter name : Hania

Add Job for Company name : Hania's Company

no skills yet

Broker Page:

After the recruiter presses on a specific job detail, a list of all students on the system that have matched with his needed job skills will then appear to him, then he can contact any user he chooses.

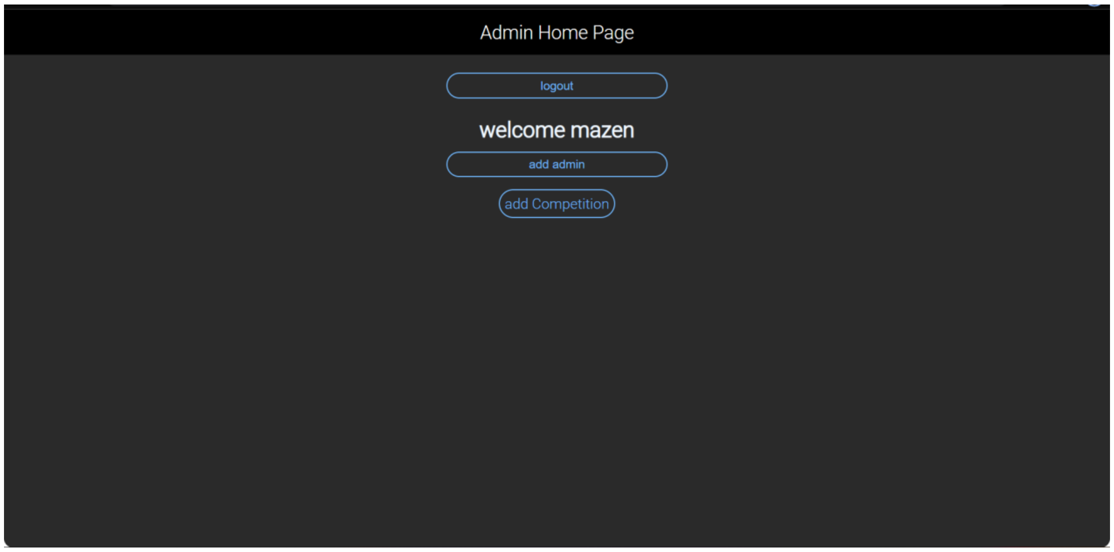


Job Title : Software Engineering
required skills : AWS,Agile,Information Technology (IT),Data Engineering,Git,SQL,DevOps,Azure
company Name :Hania's Company

Name: Ricardo Saporta
matched with accuracy : 87.%
Skills: ['Unnamed: 0', 'Unnamed: 0_x', 'Erlang', 'Ruby', 'Delphi/Object Pascal', 'Assembly', 'CSS', 'Python', 'Groovy', 'Perl', 'Java', 'C', 'Matlab', 'Bash/Shell', 'Objective-C', 'C#', 'Visual Basic 6', 'Scala', 'VB.NET', 'CoffeeScript', 'R', 'Swift', 'SQL', 'Rust', 'Julia', 'Redis', 'MariaDB', 'Elasticsearch', 'Google Cloud Storage', 'PostgreSQL', 'Neo4j', 'MongoDB', 'IBM Db2', 'Amazon DynamoDB', 'Amazon
age : 31
Gold points:14
Silver points:136
Bronze points:168

Admin page:

Only Admins can add other admins and add competitions, and this is their home page.



Chapter5: Implementation and Testing

5.1 Implementation

5.1.1. Pre-Processing

Before passing the sentences to the search engine or the user's skills in our system to the cosine similarity function to calculate it used in the recommendation engine, we have to perform an important step we cannot ignore it, it is pre-processing the incoming text from the user because we are dealing with natural language.

Dealing with natural language it is not easy at all, user may insert a word or a character that wouldn't help us in measuring the similarity, such as: is, at, on, in, the, of, a, etc.

No machines can't understand these. In fact, machines can't understand any text data at all, be it the word "blah" or the word "machine". They only understand numbers. So, over the decades scientists have been researching how to make machines understand our language. And thus, they developed all the Natural Language Processing or NLP Techniques.

Raw text data might contain unwanted or unimportant text due to which our results might not give efficient accuracy and might make it hard to understand and analyze. So, proper pre-processing must be done on raw data. Pre-processing is therefore the most important task in NLP. It helps us remove all the unimportant things from our data and make our data ready for further processing.

Python provide many libraries for text pre-processing, we used one of them in the pre-processing stage, it is Natural Language ToolKit (NLTK).

NLTK is a wonderful open-source Python library that provides modules for classification, tokenization, stemming, tagging, etc.

The pre-processing for the user incoming text for his/her skills is very important to give us an accurate similarity, there are different ways to preprocess a text. Here are some of the approaches that was followed on our pre-processing stag

Lowercasing

Normalizing

Tokenizing

Removing stop words

Lemmatization

- **Lowercasing:**

Lowercasing all our incoming text data, although commonly overlooked, is one of the simplest and most effective form of text preprocessing. It is applicable to most text mining and NLP problems and can help in cases where your dataset is not very large and significantly helps with consistency of expected output.

So, Word with different cases all map to the same lowercase form.

Here is an example where lowercasing is very useful in our system in the search engine. Imagine, a user is searching for a question containing “python”. However, no results were showing up because of the user incoming text was “python” was indexed as “PYTHON”.

- **Normalization:**

Before further processing, text needs to be normalized. Normalization generally refers to a series of related tasks meant to put all text on a level playing field: converting all text to the same case (Upper or lower), removing punctuation, converting numbers to their word equivalents, and so on. Normalization puts all words on equal footing and allows processing to proceed uniformly.

For example:

The user may insert a skill “ “java’s programmer” ”, the computer may not understand that it is mapped to “java programmer”

Normalizing helps us to remove any special characters, because they wouldn’t help us in the measurement of similarity.

- **Tokenization:**

It is a step which splits longer strings of text into smaller pieces, or tokens. Larger chunks of text can be tokenized into sentences, sentences can be tokenized into words, etc. Further processing is generally performed after a piece of text has been appropriately tokenized. Tokenization is also referred to as text segmentation or lexical analysis. Sometimes segmentation is used to refer to the breakdown of a large chunk of text into pieces larger than words (e.g., paragraphs or sentences), while tokenization is reserved for the breakdown process which results exclusively in words. This is important because the meaning of the text could easily be interpreted by analyzing the words present in the text.

- **Removing stop words:**

The words which are generally filtered out before processing a natural language are called stop words. These are actually the most common words in any language (like articles, prepositions, pronouns, conjunctions, etc.) and does not add much information to the text. Examples of a few stop words in English are “the”, “a”, “an”, “so”, “what”. Stop words are available in abundance in any human language. By removing these words, we remove the low-level information from our text in order to give more focus to the important information. In order words, we can say that the removal of such words does not show any negative consequences on the model we train for our task.

one of the main reasons for removing stop words was to decrease the computational time for text mining; it can be regarded as a dimensionality reduction of text data and was commonly-used in search engines to give better results.

NLTK library provided by python was very helpful and great in this stage.

- **Lemmatization**

Lemmatization is the algorithmic process of finding the lemma of a word depending on their meaning. Lemmatization usually refers to the morphological analysis of words, which aims to remove inflectional endings. It helps in returning the base or dictionary form of a word, which is known as the lemma.

The NLTK Lemmatization method is based on WordNet's built-in morph function. We standardize the incoming text through lemmatization. This boils down a word to just the root, which can be useful in minimizing the unique number of words used. This is certainly an optional step, in some cases such as text generation this information may be important - while in others such as classification it may be less so. To lemmatize our tokens, we used the NLTK WordNetLemmatizer.

After illustrating the pre-processing stages, here is an example to summarize them.

"How can I creates a "function" in Java?"

After lowercasing the incoming text data to the search bar, the sentence will be like that,

"how can i creates a "function" in java?"

After Normalize the incoming text data to the search bar, the sentence will be like that,

"how can i creates a function in java"

After Tokenizing the incoming text data to the search bar, the sentence will be like that,

{"how", "can", "i", "creates", "a", "function", "in", "java"}

After removing stop words, the incoming text data to the search bar, the sentence will be like that,

{"creates", "function", "java"}

After lemmatization, the incoming text data to the search bar, the sentence will be like that,

{“create”, “function”, “java”}

Here is our sentence is ready for the next stage which it is “word embedding”

5.1.2. Word Embedding and Vectorization

A word embedding is a learned representation for text where words that have the same meaning have a similar representation.

It is this approach to representing words and documents that may be considered one of the key breakthroughs of deep learning on challenging natural language processing problems.

Word embeddings are in fact a class of techniques where individual words are represented as real-valued vectors in a predefined vector space. Each word is mapped to one vector and the vector values are learned in a way that resembles a neural network, and hence the technique is often lumped into the field of deep learning. Key to the approach is the idea of using a dense distributed representation for each word. Each word is represented by a real-valued vector, often tens or hundreds of dimensions. This is contrasted to the thousands, or millions of dimensions required for sparse word representations, such as a one-hot encoding. words that are closer in the vector space are expected to be similar in meaning

- **TF-IDF:**

TF-IDF (term frequency-inverse document frequency) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents.

This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents.

While machine learning algorithms traditionally work better with numbers, TF-IDF algorithms help them dealing with words by allocating them a numerical value or vector. This has been revolutionary for machine learning, especially in fields related to NLP such as text analysis.

In text analysis with machine learning, TF-IDF algorithms help sort data into categories, as well as extract keywords. This means that simple, monotonous tasks, like tagging.

Variations of the tf–idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query. tf–idf can be successfully used for stop-words filtering in various subject fields, including text summarization and classification.

a word vector represents a document as a list of numbers, with one for each possible word of the corpus. Vectorizing a document is taking the text and creating

one of these vectors, and the numbers of the vectors somehow represent the content of the text. TF-IDF enables us to give us a way to associate each word in a document with a number that represents how relevant each word is in that document. Then, documents with similar, relevant words will have similar vectors, which is what we are looking for in a machine learning algorithm.

TF-IDF for a word in a document is calculated by multiplying two different metrics:

The term frequency (TF) of a word in a document. There are several ways of calculating this frequency, with the simplest being a raw count of instances a word appears in a document. Then, there are ways to adjust the frequency, by length of a document, or by the raw frequency of the most frequent word in a document.

As shown below:

tf(t) = No. of times term ‘t’ occurs in a document

tf(t) = (No. of times term ‘t’ occurs in a document) / (No. Of terms in a document)

tf(t) = (No. of times term ‘t’ occurs in a document) / (Frequency of most common term in a document)

Sklearn library which was used in our system uses the first one i:e No. Of times a term ‘t’ appears in a document.

The inverse document frequency (IDF) of the word across a set of documents. This means, how common or rare a word is in the entire document set. The closer it is to 0, the more common a word is. This metric can be calculated by taking the total number of documents, dividing it by the number of documents that contain a word, and calculating the logarithm.

The reason we need IDF is to help correct for words like “of”, “as”, “the”, etc. since they appear frequently in an English corpus. Thus, by taking inverse

document frequency, we can minimize the weighting of frequent terms while making infrequent terms have a higher impact.

IDFs can also be pulled from either a background corpus, which corrects for sampling bias, or the dataset being used in the experiment at hand.

So, if the word is very common and appears in many documents, this number will approach 0. Otherwise, it will approach 1.

Multiplying these two numbers results in the TF-IDF score of a word in a document. The higher the score, the more relevant that word is in that particular document.

To put it in more formal mathematical terms, the TF-IDF score for the word t in the document d from the document set D is calculated as follows:

$$tf\,idf(t, d, D) = tf(t, d) \,.\, idf(t, D)$$

Where:

$$tf(t, d) = \log(1 + freq(t, d))$$

$$idf(t, D) = \log\left(\frac{N}{count(d \in D : t \in d)}\right)$$

Once you've transformed words into numbers, in a way that machine can understand it, our data is ready to be passed to the recommendation engine and calculate the cosine similarity using the vectors or to be passed to the text classification model.

5.1.3. Cosine Similarity

We have used the Cosine similarity to measure the similarities between the user's skills in order to implement the job Recommendation Engine.

Also, this measurement was used to build an Automated Recruitment system.

Cosine similarity is a measure of similarity between two sequences of numbers. For defining it, the sequences are viewed as vectors in an inner product space, and the cosine similarity is defined as the cosine of the angle between them, that is, the dot product of the vectors divided by the product of their lengths. It follows that the cosine similarity does not depend on the magnitudes of the vectors, but only on their angle. The cosine similarity always belongs to the interval.[−1,1].

For example, two proportional vectors have a cosine similarity of 1, two orthogonal vectors have a similarity of 0, and two opposite vectors have a similarity of -1. The cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0,1].

For example, in information retrieval and text mining, each word is assigned a different coordinate and a document is represented by the vector of the numbers of occurrences of each word in the document. Cosine similarity then gives a useful measure of how similar two documents are likely to be, in terms of their subject matter, and independently of the length of the documents.

the closer the documents are by angle, the higher is the Cosine Similarity (Cos theta).

$$Cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \ \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2} \sqrt{\sum_1^n b_i^2}}$$

where, $\vec{a} \cdot \vec{b} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ is the dot product of the two vectors.

where,

- a . b = product (dot) of the vectors 'a' and 'b'.
- ||a|| and ||b|| = length of the two vectors 'a' and 'b'.
- ||a|| * ||b|| = cross product of the two vectors 'a' and 'b'.

As we include more words from the document, it's harder to visualize a higher dimensional space. But you can directly compute the cosine similarity using this math formula. Enough with the theory.

The cosine similarity between two vectors is measured in 'θ'.

Simply:

If θ = 0°, the 'x' and 'y' vectors overlap, thus proving they are similar.

If θ = 90°, the 'x' and 'y' vectors are dissimilar.

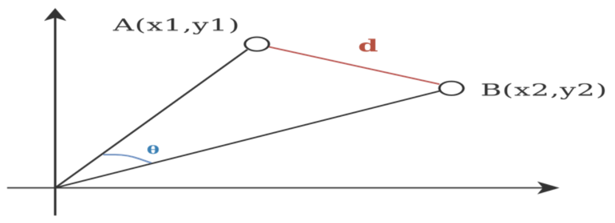


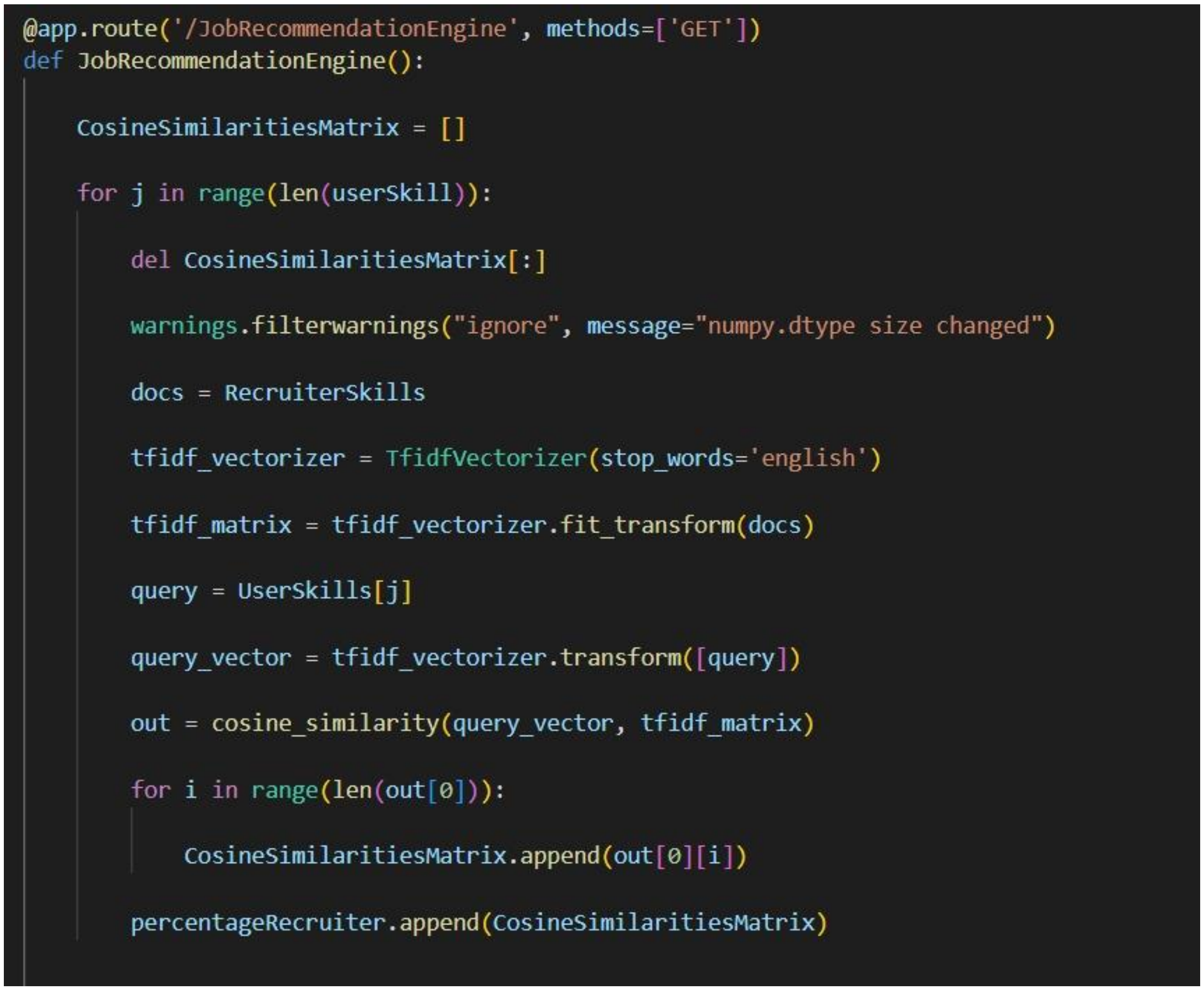
Figure 10: Euclidean vs. Cosine Distance

This is a visual representation of Euclidean distance (d) and cosine similarity (θ). While cosine looks at the angle between vectors (thus not taking into regard their weight or magnitude), Euclidean distance is similar to using a ruler to actually measure the distance.

The cosine similarity is beneficial because even if the two similar data objects are far apart by the Euclidean distance because of the size, they could still have a smaller angle between them. Smaller the angle, higher the similarity.

When plotted on a multi-dimensional space, the cosine similarity captures the orientation (the angle) of the data objects and not the magnitude.

Implementation Code Snippet:



```
@app.route('/JobRecommendationEngine', methods=['GET'])
def JobRecommendationEngine():

    CosineSimilaritiesMatrix = []

    for j in range(len(userSkill)):

        del CosineSimilaritiesMatrix[:]

        warnings.filterwarnings("ignore", message="numpy.dtype size changed")

        docs = RecruiterSkills

        tfidf_vectorizer = TfidfVectorizer(stop_words='english')

        tfidf_matrix = tfidf_vectorizer.fit_transform(docs)

        query = UserSkills[j]

        query_vector = tfidf_vectorizer.transform([query])

        out = cosine_similarity(query_vector, tfidf_matrix)

        for i in range(len(out[0])):

            CosineSimilaritiesMatrix.append(out[0][i])

        percentageRecruiter.append(CosineSimilaritiesMatrix)
```

5.1.4. Search Engine

A search engine is one of our main components and one of our most important ones right after the text classification. It is mainly used to help our users find whatever they are looking for with ease. Our search engine’s main functionalities are as follows:

- Recruiter searches for students by user id.
- Recruiter searches for students by specific skills.
- Student searches for jobs by job title.
- Student searches for jobs by company name.
- Student searches for jobs by specific skills.
- Student searches for questions by question titles.

In order to increase the accuracy of the returned outputs to our users, we have decided to use the cosine similarity method to measure the similarity between the text entered and the skills in our database.

5.1.5. Recommendation Engine

Not all of us is aware about the market needs, we all study and work hard, but not all of us are employed. We implemented the recommendation engine to help us avoid this situation.

This engine is implemented using several techniques, such as pre-processing, word embedding and cosine similarity.

The recommendation engine in our system provides the students awareness towards the market needs and reduces the gap between them and employment.

Our recommendation engine in the system is split into two categories:

5.1.5.1. Job recommendation engine

The main purpose of this engine is recommending jobs for students.

This is done by going through the student’s skills, checking the similarity between the student’s skills and the job specifications added by the recruiters to our system.

After calculating the similarities for each user and other job specification, we receive a matrix which represents the similarity between them.

After choosing a threshold which equals to 0.8 in our case, we are going to recommend the jobs with the similarities that match or are higher than the threshold to the student.

This engine is used to help students find jobs. Not just searching for any job and applying for it, but also recommending the jobs that fits the most with the student skills, to get them closer into being accepted.

5.1.5.2. Skills recommendation engine

The skills recommendation engine is used for reducing the gap between the student and the market needs, by going through their skills and calculating the similarity between all the users based on their skills, students with high match rates (with similarities equal or higher than the threshold of 0.7), are being considered in the next step.

The next step is looking for those students with high similarities and looking for the students who are employed and others who are not.

If a user is employed and matches with another user with a similarity that equals or is over 70%, we are going to look for the differences between them.

That is because we consider that these differences are the causes of one person being employed and the other is not.

Capturing the similarities between users means also capturing the field and category they're interested in. So, the skills which will be recommended for the students are very similar to their roadmap. This engine helps students to get employed if they follow the recommendations.

Improving the student's skills based on the similarity with others makes them much closer to match the job specifications and the market needs. It also makes them aware of the new technologies in their fields to learn it or improve their skills on others.

5.1.6. Broker Engine

In order that we are going to facilitate the recruitment process and to make the recruitment process much easier and effective, a broker engine was implemented for the recruitment process.

It helps the recruiters to find the student which has been matched with the recruiter's job specifications. The recruiter does not have to carry about searching for the most matched user with his job specifications. Our broker engine is responsible for this process.

Searching in thousands or millions of students that match with a recruiter job specification is a really hard way and not usable for the recruiter.

The process of matchmaking is very important and must be performed in the most efficient way to bring out the most helpful results for the recruiter.

Matchmaking is done by calculating the similarity between the job specifications added by the recruiters with each user's skills.

After the broker engine job ended the recruiter will get the most students matched with a specific job, recruiters can easily choose a student and contact him. The recruitment process become much effective.

5.1.7. Text Classification

Text classification also known as text tagging or text categorization is the process of categorizing text into organized groups. By using Natural Language Processing (NLP), text classifiers can automatically analyze text and then assign a set of pre-defined tags or categories based on its content.

We have used the text classification in Topic Detection for the incoming questions from students by assigning a label (tag) for it. This helps students in searching on our website.

It is not that easy to deal with natural language, because students may enter meaningless data or data that may not help in the machine learning process and decrease our model's performance.

Therefore, we followed the pre-processing steps on our data, vectorized it using the TFIDF technique, and then passed it to the keras model to train and tested it.

This is the most important and complex component in our system. This component's main goal is to predict the topic of every question entered into our system in order to create its tag. We have achieved this by training a machine learning model using keras, the question titles and their tags are used as the dataset of our model. They are then trained using a deep neural network model.

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

Keras is:

- Simple, but not simplistic. Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter.
- Flexible, Keras adopts the principle of progressive disclosure of complexity: simple workflows should be quick and easy, while arbitrarily advanced workflows should be possible via a clear path that builds upon what you've already learned.
- Powerful, Keras provides industry-strength performance and scalability: it is used by organizations and companies including NASA, YouTube, and Waymo.

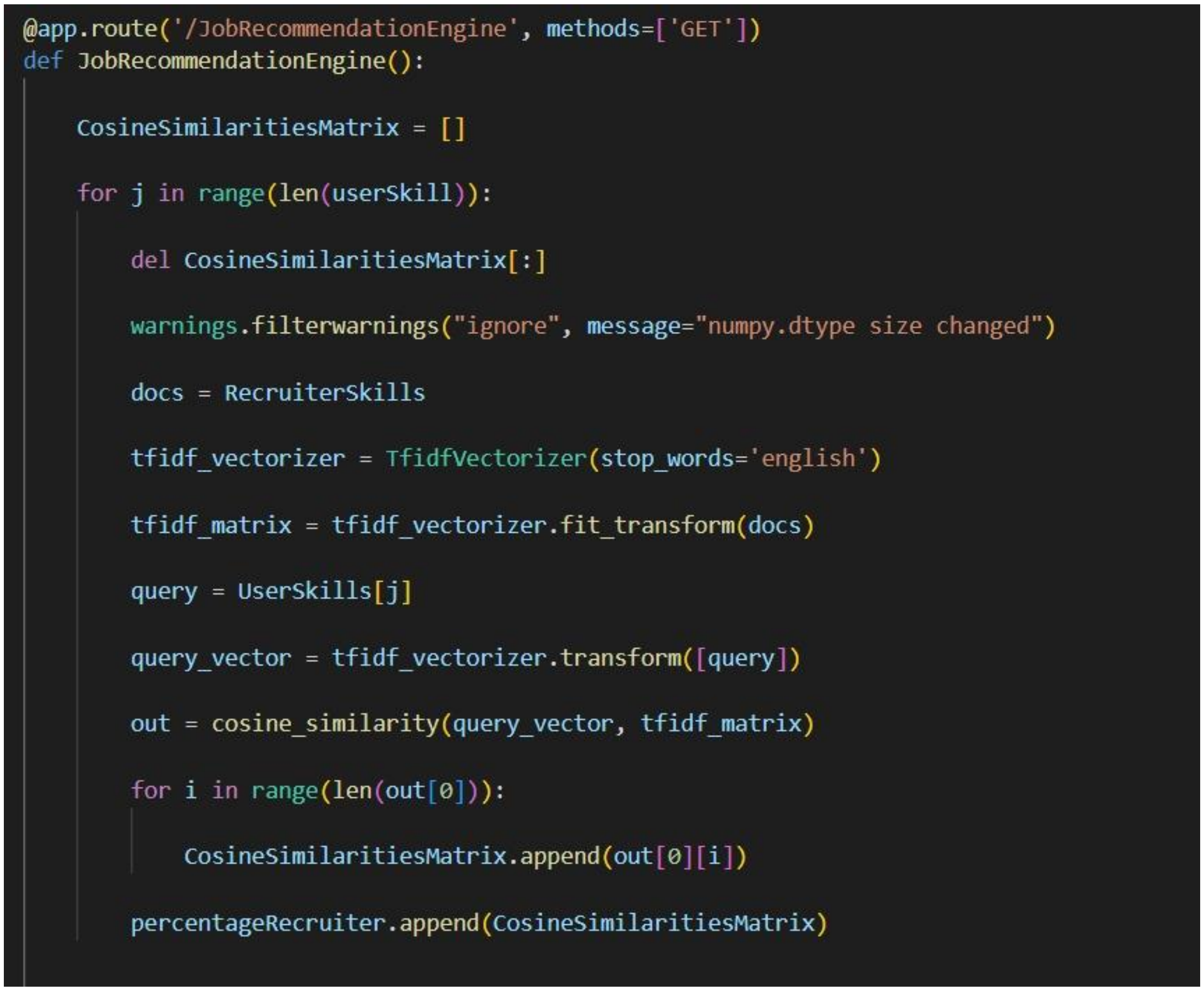
We all know that the first problem anybody faces when trying to train a machine learning model is the data. Our data was gathered by using Stack Exchange api, by fetching the data with specific tags. After the data was gathered, we faced a new problem, which was the imbalanced data. After removing duplicates, collecting more data, and handling this problem.

We are going to perform the preprocessing steps in our data such as lower casing, Normalizing, tokenizing, removing stop words and lemmatizing the data.

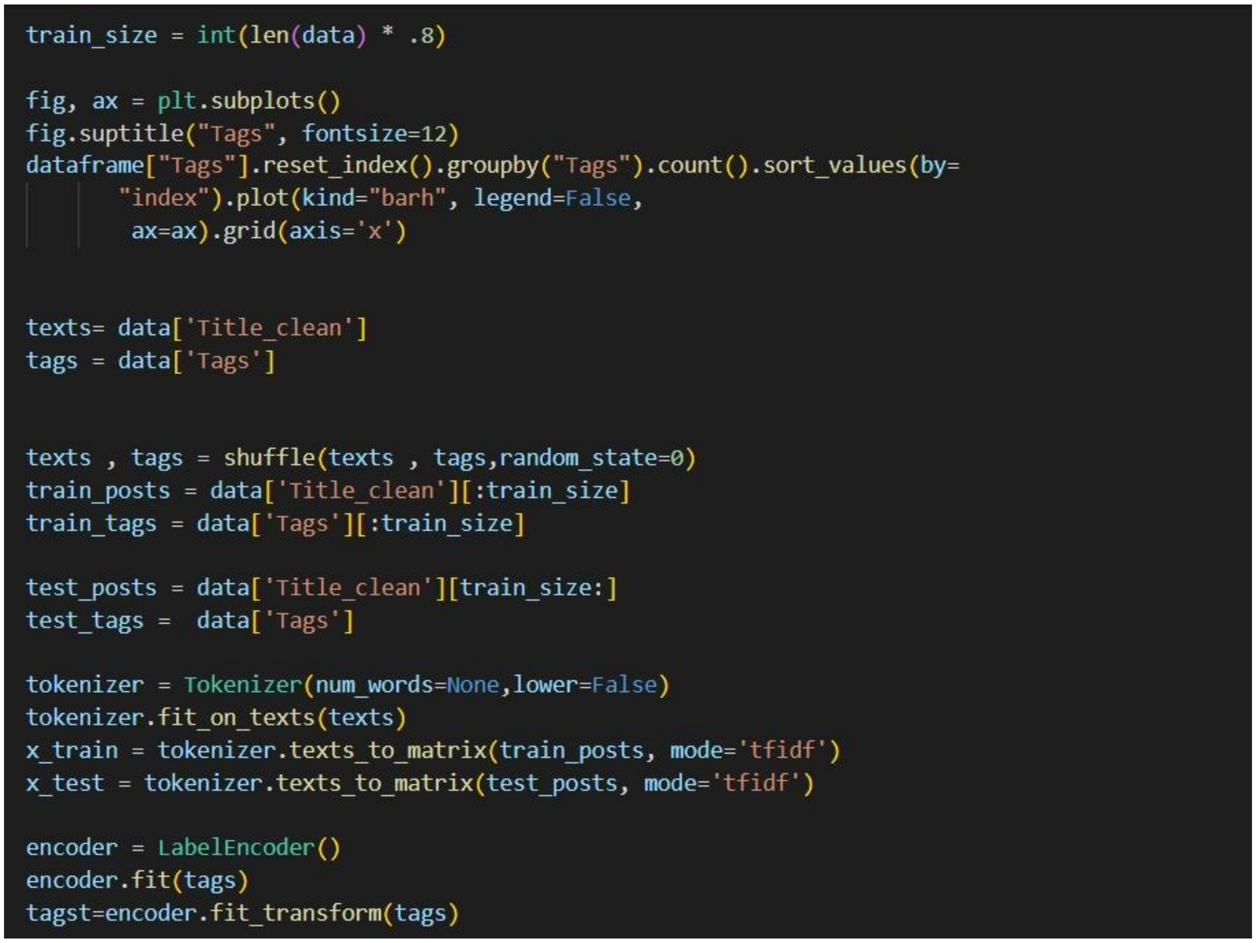
Since the machine can’t understand natural language and it only understands numbers, we are going to follow the next step.

The next step performed is vectorizing the data using TFIDF technique and then passing the values to the model to train.

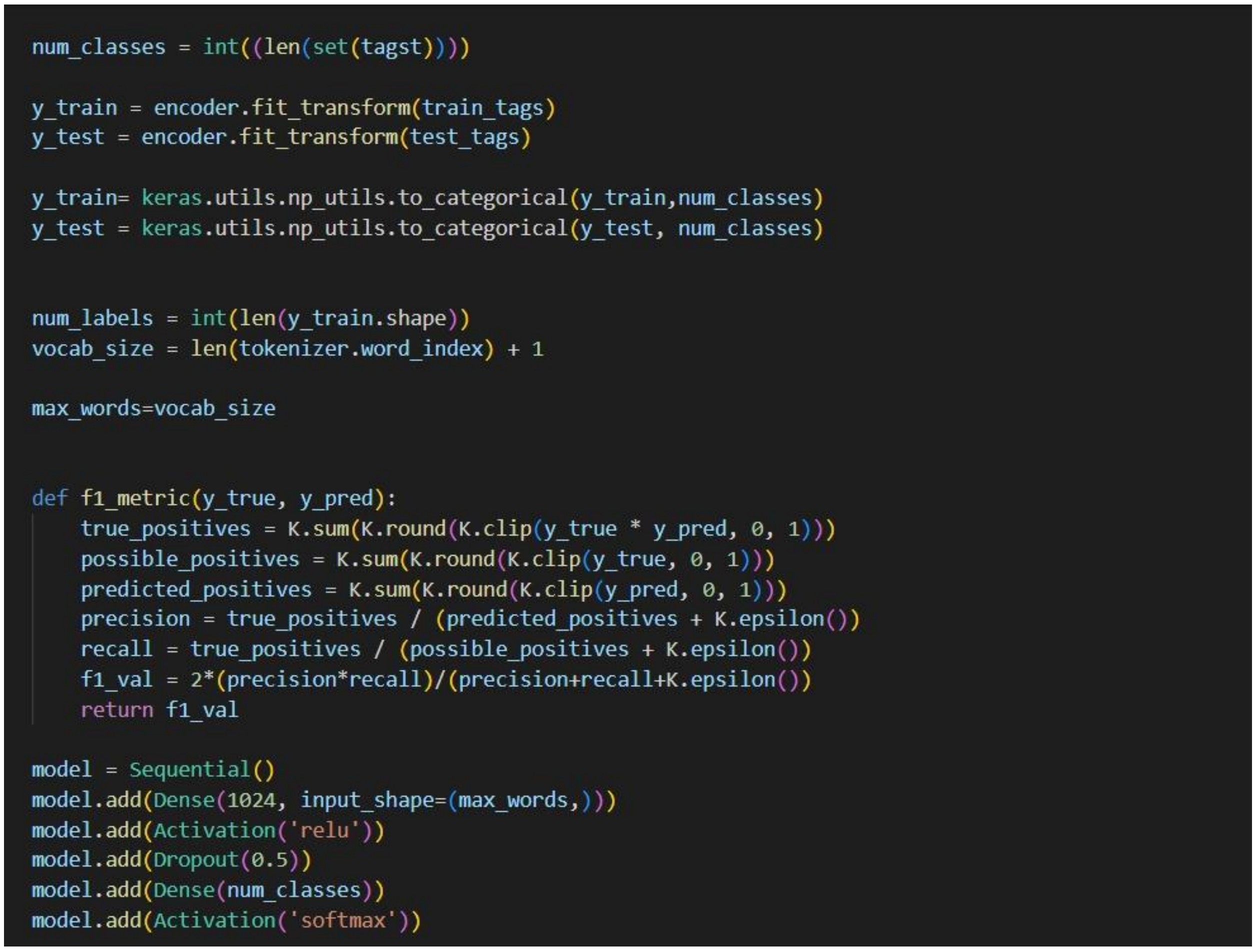
Implementation Code Snippets:



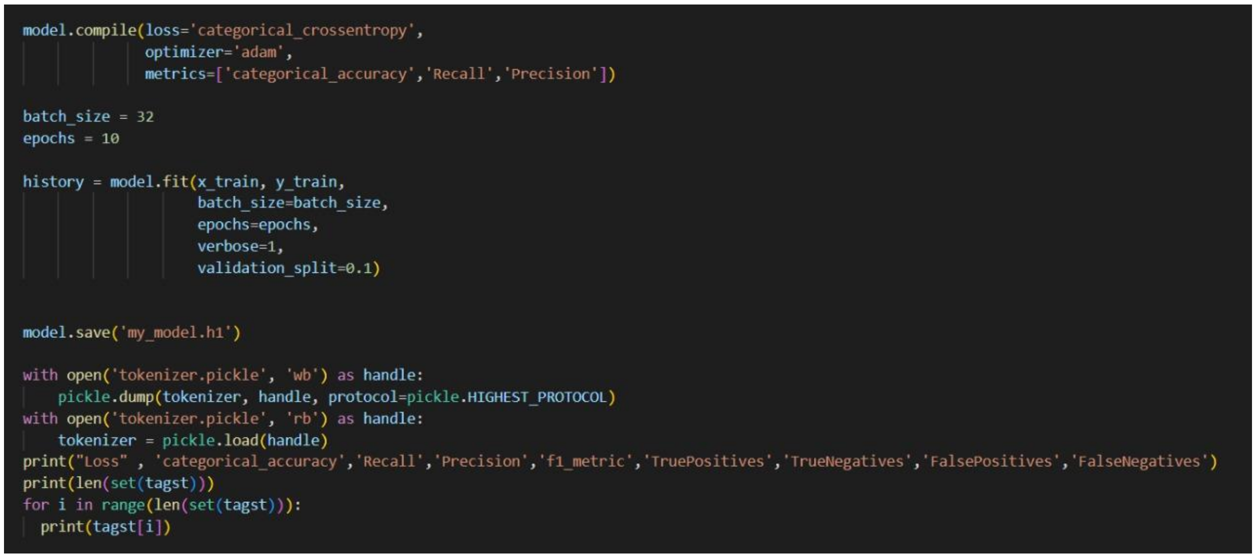
```python
@app.route('/JobRecommendationEngine', methods=['GET'])
def JobRecommendationEngine():

    CosineSimilaritiesMatrix = []

    for j in range(len(userSkill)):

        del CosineSimilaritiesMatrix[:]

        warnings.filterwarnings("ignore", message="numpy.dtype size changed")

        docs = RecruiterSkills

        tfidf_vectorizer = TfidfVectorizer(stop_words='english')

        tfidf_matrix = tfidf_vectorizer.fit_transform(docs)

        query = UserSkills[j]

        query_vector = tfidf_vectorizer.transform([query])

        out = cosine_similarity(query_vector, tfidf_matrix)

        for i in range(len(out[0])):

            CosineSimilaritiesMatrix.append(out[0][i])

        percentageRecruiter.append(CosineSimilaritiesMatrix)
```



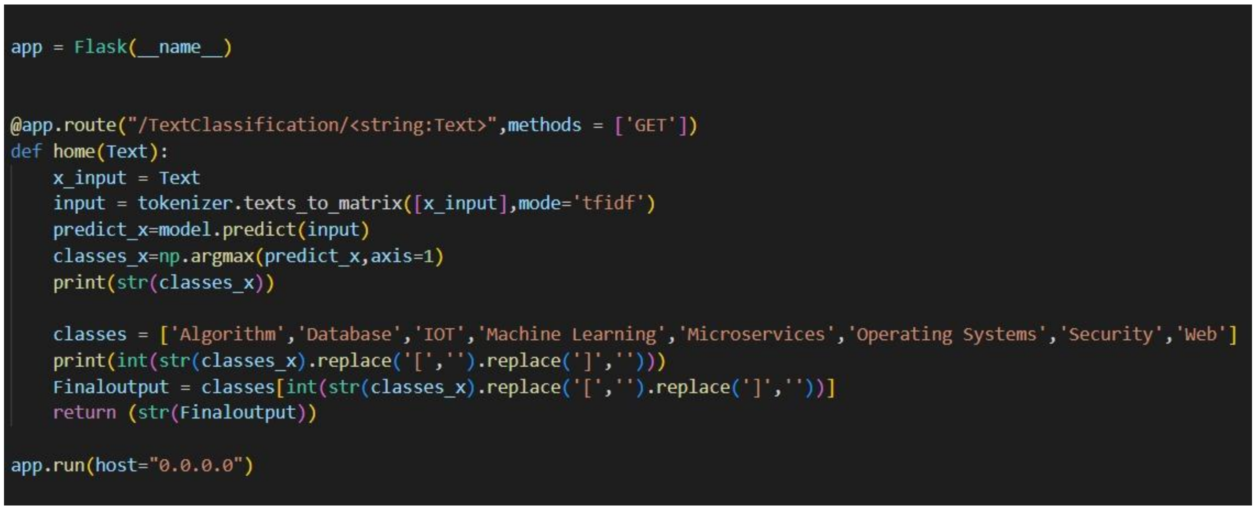
```
train_size = int(len(data) * .8)

fig, ax = plt.subplots()
fig.suptitle("Tags", fontsize=12)
dataframe["Tags"].reset_index().groupby("Tags").count().sort_values(by=
        "index").plot(kind="barh", legend=False,
         ax=ax).grid(axis='x')


texts= data['Title_clean']
tags = data['Tags']


texts , tags = shuffle(texts , tags,random_state=0)
train_posts = data['Title_clean'][:train_size]
train_tags = data['Tags'][:train_size]

test_posts = data['Title_clean'][train_size:]
test_tags =  data['Tags']

tokenizer = Tokenizer(num_words=None,lower=False)
tokenizer.fit_on_texts(texts)
x_train = tokenizer.texts_to_matrix(train_posts, mode='tfidf')
x_test = tokenizer.texts_to_matrix(test_posts, mode='tfidf')

encoder = LabelEncoder()
encoder.fit(tags)
tagst=encoder.fit_transform(tags)
```



```
num_classes = int((len(set(tagst))))

y_train = encoder.fit_transform(train_tags)
y_test = encoder.fit_transform(test_tags)

y_train= keras.utils.np_utils.to_categorical(y_train,num_classes)
y_test = keras.utils.np_utils.to_categorical(y_test, num_classes)


num_labels = int(len(y_train.shape))
vocab_size = len(tokenizer.word_index) + 1

max_words=vocab_size


def f1_metric(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    recall = true_positives / (possible_positives + K.epsilon())
    f1_val = 2*(precision*recall)/(precision+recall+K.epsilon())
    return f1_val

model = Sequential()
model.add(Dense(1024, input_shape=(max_words,)))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes))
model.add(Activation('softmax'))
```



```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['categorical_accuracy','Recall','Precision'])

batch_size = 32
epochs = 10

history = model.fit(x_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_split=0.1)


model.save('my_model.h1')

with open('tokenizer.pickle', 'wb') as handle:
    pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST_PROTOCOL)
with open('tokenizer.pickle', 'rb') as handle:
    tokenizer = pickle.load(handle)
print("Loss" , 'categorical_accuracy','Recall','Precision','f1_metric','TruePositives','TrueNegatives','FalsePositives','FalseNegatives')
print(len(set(tagst)))
for i in range(len(set(tagst))):
  print(tagst[i])
```



```
app = Flask(__name__)


@app.route("/TextClassification/<string:Text>",methods = ['GET'])
def home(Text):
    x_input = Text
    input = tokenizer.texts_to_matrix([x_input],mode='tfidf')
    predict_x=model.predict(input)
    classes_x=np.argmax(predict_x,axis=1)
    print(str(classes_x))

    classes = ['Algorithm','Database','IOT','Machine Learning','Microservices','Operating Systems','Security','Web']
    print(int(str(classes_x).replace('[','').replace(']','')))
    Finaloutput = classes[int(str(classes_x).replace('[','').replace(']',''))]
    return (str(Finaloutput))

app.run(host="0.0.0.0")
```

5.1.8. Ranking

Our broker engine, and our job skills search engine are both implemented by measuring the similarity between the text inputted and the skills in our database. For our broker engine, we have decided on a threshold of 50% which was the best according to our data. For our job skills search engine, we have decided to go with a threshold of 70% since it was the most suitable. After that step was done, we started sorting the similarities we had, and retrieve the ids of such skills that produced such a similarity. In our broker Engine our data was then ordered according to the students' badges, if 2 users had the same number of gold badges, we start ranking them by their silver badges, if the same issue occurs again we start ordering them according to their bronze badges, and if those badges are the same too, then we output both users in any order. In our search engine we order the recruiter ids according to the similarities and output them after the sorting function has been applied on them.

5.1.9. Dynamic behavior:

Capturing the behavior of the student is very important to encourage him. This strategy is used in our system to encourage students to interact with each other in order to explore the world's knowledge.

Our community must be helpful and effective, so the idea of tracking the dynamic behavior of the student is a must to achieve this goal.

This is done by the interacting with other students and help them to get the information they needed or solve their problems.

Not only is the student helping others to solve their problems and share his knowledge globally, but also the student will have a positive effect on that. Student will have a higher rank on our website because of the points gained by the effective interaction with other student's questions, which will result in the badges being added when a specific criterion is met in the system.

Students will also gain badges when they join competitions and do well on them.

These competitions help students to learn more and improve their rank in the system.

Students needs to improve his rank since the rank and badges are used in ranking the results outputted for the recruiter while searching for students with specific skills.

This makes their luck for getting the job better than others.

5.2 Testing:

5.2.1 System testing:

5.2.1.1. Login Test Case (Student):

No.	Step	Expected Results	Actual Results	Status
1	Student navigates to https//example.	Website is displayed	As Expected	Passed
2	Enter valid username and password	Username should appear in the username text box, password must appear masked.	As Expected	Passed
3	Student clicks on “Login” button	The button must be clickable, Student Logs in successfully and the feed page of the student appears	As Expected	Passed

Table 3: Test case1

5.2.1.2. Student Logout Test Case (Student):

No.	Step	Expected Results	Actual Results	Status
1	Student Clicks on the “Logout” button	The button is clickable and the student logs out of the website and returns to the login page	As Expected	Passed
2	Student clicks on the “back” button in the browser	Student does not return to his logged in account, Student must login again	As Expected	Passed

Table 4: Test case 2

5.2.1.3. Keywords Search Engine Test Case (Student):

No.	Step	Expected Results	Actual Results	Status
1	Student clicks on the “Search bar” button	The search bar is clickable	As Expected	Passed
2	Student enters keyword to search for it	Keyword appears in the search bar	As Expected	Passed
3	Student clicks on “Search” button	Questions retrieved are displayed to the student	As Expected	Passed

Table 5: Test case 3

5.2.1.4. Adding A Question Test Case (Student):

No.	Step	Expected Results	Actual Results	Status
1	Student clicks on the “Add question” button	The button is clickable	As Expected	Passed
2	Student enters the question they want to add	Text entered appears to the student in the text box	As Expected	Passed
3	Student clicks on “get tag” button	Button must be clickable and the tag appears to the student	As Expected	Passed
4	Student clicks on “add tag”	Tag is added to the question successfully	As Expected	passed
5	Student clicks on “post question”	Question is added successfully	As Expected	passed

Table 6: Test case 4

5.2.1.5. Competition Test Case (Student):

No.	Step	Expected Results	Actual Results	Status
1	Clicking on the “competitions” button	The available competition is displayed to the student.	As Expected	Passed
2	Student click on “join competition”	Student successfully joined the competition and the questions appear with its options	As Expected	Passed
3	Student chooses all the correct choices for the questions	The bullet points chosen by the student are flagged	As Expected	Passed
4	Student click submit	Student’s grade is displayed and the badge is added to the student’s profile	As Expected	Passed

Table 7: Test case 5

5.2.1.6. Answer Likes (Student):

No.	Step	Expected Results	Actual Results	Status
1	Student clicks on “like” for a specific answer	Button is clickable, a point is added to the owner of the answer and a notification is sent.	As Expected	Passed

Table 8: Test case 6

5.2.1.7. Student Skills Search Engine (Recruiter):

No.	Step	Expected Results	Actual Results	Status
1	Recruiter clicks on "search for students"	Search bar is clickable	As Expected	Passed
2	Recruiter enters the skills he/she wants to search for and presses the "Search" button	List of students displayed for the recruiter based on the skills entered	As Expected	Passed

Table 9: Test case 7

5.2.1.8. Add Answer Test Case (Student):

No.	Step	Expected Results	Actual Results	Status
1	Student clicks on "add answer" button for a specific question in his feed page	Student is able to write his answer in the text box.	As Expected	Passed
2	Student clicks on "add"	The answer is added to the question successfully and displayed for the students	As Expected	Passed

Table 10: Test case 8

5.2.1.9. Profile Page (Student):

No.	Step	Expected Results	Actual Results	Status
1	Student navigates to his profile page	The profile page opens successfully	As Expected	Passed
2	Student clicks on “add skill” and enters the skill to be added	Button is clickable and the student is able to see his text	As Expected	Passed
3	Student clicks on “add”	Button is clickable and the student is able to see his text and the profile is updated	As Expected	Passed

Table 11: Test case 9

5.2.1.10.Help Test Case (Student):

No.	Step	Expected Results	Actual Results	Status
1	Student click “help” button.	Student will be routed to help page,	As Expected	Passed
2	Student fill the help form with data.	Student can see the data entered in the text box	As Expected	Passed
3	Student click on “Submit” button	Button is clickable and a message displayed for the Student “Thank you, we are going to help you as soon as possible”	As Expected	Passed

Table 12: Test case 10

5.2.1.11. Register(Recruiter):

No.	Step	Expected Results	Actual Results	Status
1	Recruiter navigates to https//example	Website is displayed	As Expected	Passed
2	Recruiter Clicks on “Create an account” hyperlink	Recruiter is routed to the registration page.	As Expected	Passed
3	Recruiter enters a valid e-mail and fills the required fields except the password.	Recruiter information displayed in the text boxes.	As Expected	Passed
4	Recruiter clicks on “Register” button	A message is displayed to the recruiter “please fill all required fields.”	As Expected	Passed

Table 13: Test case 11

5.2.1.12. Broker Engine (Recruiter):

No.	Step	Expected Results	Actual Results	Status
1	Recruiter navigates to the "home page"	Website is displayed	As Expected	Passed
2	Recruiter clicks on the "My Jobs" button	Recruiter's jobs display for the recruiter	As Expected	Passed
3	Recruiter chooses a specific job by clicking on the "Job Details" button	The details of the job which was chosen is then displayed to the recruiter after routing the recruiter to another page	As Expected	Passed
4	Recruiter clicks on the "Broker" button	A list of matching students displayed for the recruiter.	As Expected	Passed

Table 14 Test case 12

5.2.2 Text Classification testing:

- **Input: how to create a binary search tree?**
  - Predicted Result: Algorithm
  - Actual Result: Algorithm

- **Input: how to increase my algorithm performance?**
  - Predicted Output: Algorithm
  - Actual Output: Algorithm

- **Input: how to select multiple tuples in a table?**
  - Predicted Output: Database
  - Actual Output: Database

- **Input: how to create a query to Oracle db?**
  - Predicted Output: Database
  - Actual Output: Database

- **Input: how to implement apcache ranger for presto?**
  - Predicted Output: IOT
  - Actual Output: IOT

- **Input: how can I connect an iot to wifi**
  - Predicted Output: IOT
  - Actual Output: IOT

- **Input: how to implement red black tree?**
  - Predicted Output: IOT
  - Actual Output: Algorithm

- **Input: What is the best activation function in the output layer machine learning?**
  - Predicted Output: machine learning
  - Actual Output: machine learning

- **Input: How to avoid overfitting in my model machine learning?**
  - Predicted Output: machine learning
  - Actual Output: machine learning

- **Input: Can I use different database relation in different microservices?**
  - Predicted Output: Microservices
  - Actual Output: Microservices

- **Input: How to use API gateway?**
  - Predicted Output: Microservices
  - Actual Output: Microservices

- **Input: how does the kernel work?**
  - Predicted Output: operating system
  - Actual Output: operating system

- **Input: what are the advantages of virtual memory?**
  - Predicted Output: operating system
  - Actual Output: operating system

- **Input: how to implement two-way authentication?**
  - Predicted Output: security
  - Actual Output: security

- **Input: how to exchange a public key?**
  - Predicted Output: security
  - Actual Output: security

- **Input: how to add dropdown list in my page with hoover?**
  - Predicted Output: web
  - Actual Output: web

- **Input: how to create a homepage in my website?**
  - Predicted Output: web
  - Actual Output: web

- **Input: how to download mongo compass DB?**
  - Predicted Output: Web
  - Actual Output: Database

Graduation Poster

Intelligent Recruiter System

Mazen, Ibrahim, Hania, and Heba
Supervised By: Dr.Abeer El Korany

IRS
Intelligent-recruiter system

Abstract

Since there is no platform that includes all different fields, or a website for students, or fresh graduates that truly shows their abilities and their real progress throughout their college years. As students we have felt this struggle when trying to apply for jobs/internships, or when we needed questions answered regarding specific topics.

Our main goal is to make a platform where students from different fields can find the best questions and answers, interact with them, and participate in weekly competitions. The student's profile will be dynamically updated according to his behavior and interactions with other students.

Moreover, badges will be given based on the student's behavior throughout the activity of the account regarding the questions or the weekly competitions, these badges will be one of the main considerations when it comes to recruiters and their interviews.

Furthermore, due to the importance of internships/jobs to students, which provides them with experience by dealing with real-life projects, we have made it fundamental in our website to make it easier for students to be qualified when it comes to recruitment. Our students will be able to search for jobs and receive recommendations regarding the real-time market needs based on their dynamic profiles

Therefore, it is not only a Q&A software or a recruitment platform, but both of them combined.

.

Introduction

Our website facilitates the process for searching for jobs and receiving recommendations regarding the real-time market needs based on their dynamic profiles.

A platform that contributes in making learning easier, preparing students and fresh graduates for the work environment .

Reduces the gap between the students and the real market needs. And to Make recruitment process more efficient.

Our project's main technologies are text analysis, a recommendation engine, a broker engine, and a search engine.

Methods

The development methodology followed in this project is the agile methodology principles, as we manage our project by breaking it up into several phases and continuous improvement at every stage, and the client-server architecture.

Languages Used:

Client-side: HTML, CSS, JavaScript, ReactJs.

Server-side: NodeJS. Python to build the model and the main engines.

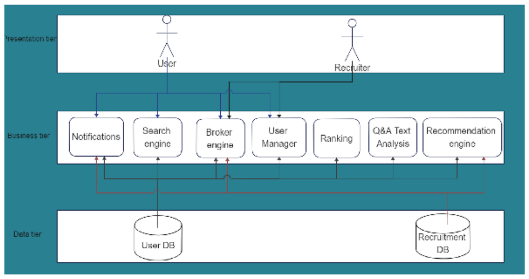
Database: MongoDB.

Software Used:

- Visual Studio Code used for python code.
- Postman to test the API's.
- Google Chrome-browser.
- Microsoft excel.

Libraries:

- TensorFlow used in developing and training our model.
- Keras.
- Sklearn: for machine learning in Python.
- Nltk library for preprocessing .
- Numpy.
- Pandas for the Python programming language.
- React is a JavaScript library for front-end in implementation.
- Express is Nodejs library for server-side API implementation.
- Flask API is used for exposing our functionality as web services



Primarily Design

The website uses text analysis to analyze questions and extract the keywords used which will contribute in the classification of it into a specific topic, while the recommendation engine is used to send recommendation notifications to students with ideas about what courses to take according to the real-time market. It will include skills recommendations and jobs recommendations which will include job opportunities that are available to students that are a fit-match for it. Moreover, the broker engine is used for the matchmaking between the recruiter's specifications and the students' skills and qualifications. Lastly, the search engine is used by students to search for jobs and for a specific topic-related question or by students to search for jobs with specific skills, or title taken as input. The list of jobs outputted will be ranked from highest match to lowest match.

Conclusion

To conclude, The goal of our system is to spread and expand knowledge. Although not all knowledge can be recorded in writing, a significant portion of it can be. In order to better understand one another, we want to bring together those who have information and those who need it. We also want to provide everyone with the ability to share their knowledge with the rest of the world without restrictions and support an efficient recruitment process.

We look forward to the improvement of students and raising their capabilities in all fields. in addition to making their inclusion in the labor market easier.

contact us: IntelligentRecruiter@gmail.com

References:

1. "List of privileges". Archived from the original on 14 February 2020. Retrieved 22 November 2017

2. "How do I ask a good question?". Retrieved 26 April 2021.

3. HackerRank (8 January 2018). "Computer science tracks supported by HackerRank".

4. "HackerRank 'gamifies' technical job recruiting for game companies (exclusive) - GamesBeat - Games - by Dean Takahashi". VentureBeat.