Visual analytics for job candidates

HESAM MASHHADI MOHAMMAD

Supervisor: Prof. VALENTINA GATTESCHI

Master's Degree in Computer Engineering

Master's Degree Thesis

POLITECNICO DI TORINO

April 2023

Abstract

The hiring process could be complicated and result in wrong decisions if only the raw data is available to the people responsible for assessing potential new employees. A more efficient and organized way for hiring managers to evaluate job applicants is using data visualization tools to improve the hiring process experience, leading to more reliable results. This work uses web application technologies and data visualization tools to present job applicants' information visually and interactively. Web applications allow consumers to access and interact with services and information on the internet via a web browser, which has become increasingly crucial in today's digital landscape. Aside from being simple to access from anywhere with an internet connection, they also have fewer infrastructure costs than traditional desktop apps and better scalability to handle big user bases. The system's design uses several visualization techniques, including graphs and charts, to display information on applicants' skills, work experience, education, and other factors. A new feature in this application lets job applicants be compared with the team members already working in the company. This way, the hiring manager can determine how each candidate will fit into the team. Using this web application has the advantages of decreasing the time and effort needed to search through resumes and improving the effectiveness and efficiency of the hiring process. This project uses the latest web application technologies, including Reactjs, Expressjs, and REST API.

Keywords – Data Visualization, Hiring process, web application, job applicants

Contents

1 Introduction **1**

1.1 Context 1

1.2 Goal of this work 2

1.3 Document Structure 2

2 Background **3**

3 Realized System **9**

3.1 Architecture 9

3.1.1 Back-end 11

3.1.1.1 Dataset 15

3.1.1.2 Conclusion Back-end 17

3.1.2 Front-end 19

3.1.2.1 Design Criteria 27

3.1.2.2 Data Visualization 30

3.2 How the application works 31

4 User Interface **52**

5 Evaluation **57**

6 Conclusion **59**

References **61**

List of Figures

3.1 Overview of the system's architecture 9
3.2 Database Entity Relation Diagram 31
3.3 Overall ranking of candidates 33
3.4 Team comparison table 41
3.5 Overall ranking of the candidates 46
3.6 Candidates skills bar chart 46
3.7 Team comparison pie chart 47
3.8 Candidates skills bubble chart 48
3.9 Candidates languages bubble chart 49
3.10 Candidates experience to overall score scatter plot 50
3.11 Candidates experience to skills scatter plot 51
4.1 list of all positions in the system 52
4.2 Adding a new position 53
4.3 Adding a new team member 53
4.4 Adding a new team 54
4.5 Position's information page 55
4.6 Comparing candidates window 55
4.7 Candidate's details window 56
4.8 Team comparison table 56
5.1 Time needed to complete tasks 57
5.2 Results of questionnaire 58

1 Introduction

1.1 Context

One of the biggest problems many firms confront is hiring employees. Shortlisting applicants from a prominent prospective candidate is one of the crucial stages of recruitment, but it may be a time-consuming and challenging task in significant corporations.

Tens of thousands of job applications are frequently submitted monthly to large firms, and manually analyzing such huge applicant volumes is time-consuming, costly, and prone to mistakes. Manual screening could pick a select group of candidates for an interview from those who apply initially and fit the job requirements. missing out on possibly better applicants, and This is because it uses a sequential selection approach. An inadequate pool of candidates is produced by this powerful ad hoc method (1). Organizations typically manage such tasks using an Applicant Tracking System (ATS), which manages a candidate application from its submission to the final hiring decision. However, such systems facilitate some aspects of the hiring process.

However, the usually used approaches may lead to non-accurate results. Because they only rely on the information provided in the job description and the job candidates' resumes. Although this approach could seem logical, it is not complete without considering external factors.

For example, there are some external factors such as whether the job candidate graduated from a top university and, if yes, how this could affect the final result of its evaluation.

Also, another aspect of the hiring process that is not considered in the other works is taking into account that the possible employee, in most cases, will work in a team. Because work teams, in the opinion of many firms, are an efficient way to address employee motivation difficulties and accomplish performance targets (2), it is essential to understand better how to fit a job candidate in the related team in the company.

1.2 Goal of this work

The aim of this work is trying to cover and consider what has not been considered by other works to beside providing better analytics also give the user visualization of the data in a way which helps him to make an accurate decision. In this system, for example, there are teams containing team members for each job position. By comparing each job candidate with the whole team members, valuable data will be produced to help the final user decide who is more eligible for that position.

1.3 Document Structure

The remaining sections of the paper are structured as follows.

- Section2 : Background
- Section3: Realized System
- Section 4: Evaluation
- Section 5 : Conclusion

2 Background

Data visualization is essential to computer science and has been used in different fields and applications.

Every day, organizations produce a considerable amount of data. Massive volumes of data need to be presented in a way that is intelligible and accessible. As a result, there is now far more data available online. Users find it challenging to visualize, study, and utilize this vast data. For scientific research to be successful, data visualization is essential. Computers can now process large volumes of data. The design, creation, and use of computer-generated graphics to display data are all aspects of data visualization. Decision-makers can see analytics in a visual format as a result, which makes it simple for them to interpret the data. It aids in pattern recognition, information comprehension, and opinion formation (3).

Information visualization and scientific visualization are other terms used to describe data visualization. In order to make messages or information survive throughout time, humans have always used visualization techniques. It can depict things that cannot be felt, tasted graphically, or smelled (4).

Data visualization is the technique of using a computer to portray information visually. In contrast to static data visualization, interactive data visualization allows users to select the format in which data is shown. Visualization techniques containing (3) :

- Line graph: This depicts how many things are related. It can be applied to contrast changes over time.

- Bar chart: This is employed to compare the amounts of several categories.
- Scatter plot: This two-dimensional plot illustrates how two items can vary.
- Pie chart: The components of a whole are compared using this.

As a result, graphs and charts can be presented in various ways, including bar charts, pie charts, and line graphs. Knowing the graph or chart to utilize data is crucial. Data visualization uses computer visuals to demonstrate patterns, trends, and connections between various data elements.

Data visualization techniques are common approaches used in previous studies(5; 6; 7). (6) used a semantic thesaurus to depict the key similarities and differences among qualifications visually. Without reading all the textual information connected to them, users might rapidly learn the critical differences between one training course and the others by using such a depiction. They presented a semantically based graphical depiction of similarities and differences between qualifications. In order to convey the relevance of the main concepts of qualifications, they were organized into clusters based on their semantic similarity and represented by circles of varying diameters. Without reading all the information linked to them, consumers could immediately understand the critical differences between one training course and the others by using this representation. A few users tested the suggested remedy after being integrated into a Web platform created as part of the TAMTAM "Exploiting the TIPTOE platform by transferring ECVET and EQF semantic tools in a Multisectoral perspective" project.

As mentioned above, one of the techniques in data visualization is a Bar chart. Bar charts have been utilized for visualization because of their capacity to swiftly express quantitative data and important values (8).

In a study by (9), a system for evaluating learners' resumes, stated in terms of Learning Outcomes (LOs) and their matching with available job offers, was presented. This system was aimed at learners but may also be utilized by recruiters. The suggested remedy used semantics to carry out intelligent automatic processing of documents in natural language. A software system could go beyond a simple keyword comparison using semantics since it could comprehend relationships between concepts. The proposed solution computed a numeric value summarizing positive and negative gaps concerning available job offers, considered the level of each LO (explicitly specified or automatically extracted from textual descriptions), and displayed the comparison's findings in a bar chart-based view. Because reasoning from aggregate results might occasionally lead to false conclusions, it was decided to offer the users a numeric result function of the semantic similarity of the LOs and their levels instead.

In an interesting study by (10), tag clouds were used to compare qualifications, resumes, and job profiles. The LO-MATCH platform, a web-based tool used as part of the MATCH "Informal and Non-Formal Competencies Matching Device for Migrant Employability and Active Citizenship" project, was introduced. Due to the use of non-shared vocabularies, the platform relied on semantic technologies to address heterogeneity issues in the descriptions of qualifications/résumés and labor market needs. Additionally, it used a tag cloud-based visualization technique to quickly illustrate factors to be considered during the mobility and job search phases.

In particular, tag cloud properties like font size and distance from the cloud's center were used to give a quick overview of a given qualification's critical features about a particular learner's needs as well as to highlight necessary job seeker attitudes toward a given job offer (from both the job seekers and the employer's points of view).

In a study by (11), An approach for selecting effective visualization methods for dynamic graph visualization scenarios that visualization specialists can use is suggested. It provided a method for balancing application requirements with the properties of dynamic graph presentation. Its objective was to help experts choose the most effective visualization technique. Both the application requirements and the visualization approaches are described in the methodology for presenting dynamic graphs as profiles that address critical aesthetic criteria. The application profile was made using the task and graph features. The most appropriate visualization technique was probably the one whose profile matches the desired application profile the closest. It provided a mechanism for gathering illustrative visualization profiles for examples of dynamic graph visualization approaches in a case study.

One of the beneficial uses of data visualization is in the field of hiring and recruitment. Some studies (7; 12) have used data visualization methods to facilitate hiring.

An article by (7) to mine and analyze resume data suggests using the visual analytics system ResumeVis. First, a text-mining strategy for extracting semantic data is provided. After that, a collection of visualizations is created to portray the semantic data from various angles. The following activities can be completed through interactive investigation on ResumeVis carried out by subject-matter experts: tracking individual career evolution, mining latent social relationships among people, and holding the complete picture of huge resumes' joint mobility.

Another study by (12) introduces Selection Central, a candidate profile visualization and ranking system that offers the user (for example, the hiring manager) two novel features. First, it provided an interface for visually comparing various aspects of candidates' profiles; second, a candidate profile ranking method and interface captured the hiring manager's perspective by letting her adjust the ranking using relevance feedback. Beyond the system-provided candidate ranking, the first feature enabled the user to shortlist candidates by rapidly digging deeper into the candidate profile. The second feature, on the other hand, enabled the user to modify the ranking according to her preferences to get around the effects of vague or ambiguous job descriptions. To produce the best appropriate rating, it let the user add her prejudices or preferences in making decisions. The article offered a thorough usability analysis to gauge Selection Central's effectiveness.

An article by (13) shortlisted candidates for job positions with the help of a system called "PROSPECT". This system tried to extract structured data from resumes and job descriptions written in natural languages, such as skills, experience, and information about education. However, extracting data from this kind of document was the main challenge this project faced.

From the previous works mentioned in this section, it could be understanded that data visualization is an essential tool in various use cases and provides a better understanding of extensive data. Although the works mentioned above offer intriguing approaches to visualizing data and comparing individual qualifications, to the best of our knowledge, they do not compare the competencies of the job seekers with the members of the teams within the organization. Contrarily, the present study has the advantage of using various data visualization techniques, including tables, bubble charts, scatter plots, bar charts, and pie charts, at once. Another advantage of this approach is that it allows users to compare job candidates' qualifications with those currently employed by the organization and in various teams.

3 Realized System

3.1 Architecture

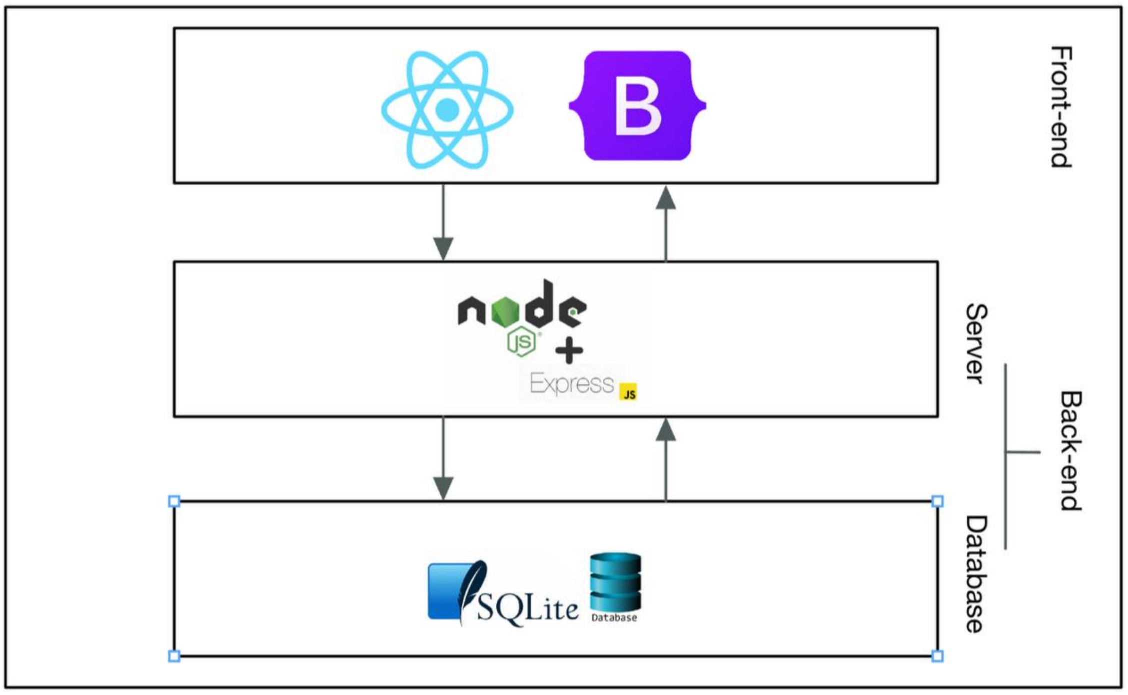


Figure 3.1: Overview of the system's architecture

The Architecture of the "Resume Visualization" web application combines a front-end and a back-end. The choice of the web application for this project is based on the fact that nowadays, almost everywhere, people have access to the internet using different devices.

A web application designed responsively is not limited to a specific device. It is accessible wherever there is an internet connection. To develop this web application, there were different combinations of technologies available.

After doing some research and based on the previous developing experiences, a set of technology was selected.

The leading technologies for this project are Reactjs + Bootstrap for developing the client side and expressjs and SQLite for the server side. The application's front-end is built using ReactJS, Bootstrap, and D3JS JavaScript libraries for creating dynamic and interactive user interfaces.

On the back-end side, the application is powered by Expressjs and SQLite. Expressjs is a popular framework for Node.js, used for building web applications. SQLite is a lightweight, file-based database system that stores and retrieves data.

The Rest API communicates between the front-end and back-end, allowing data exchange in a standardized format. The API acts as an intermediary between the two, allowing the front-end to request data from the back-end, and vice versa, in a structured and organized manner.

The combination of these technologies provides a robust and scalable architecture for the web application. ReactJS and Bootstrap in the front-end provide a rich user experience, while using Expressjs and SQLite in the back-end allows for efficient data management and processing.

The architecture of this web application also allows for easy maintenance and scalability. Separating the front-end and back-end into distinct components means that updates and changes to one side do not affect the other, allowing for smoother and quicker development cycles. Additionally, using Rest API enables easy integration of new features and functionality.

In conclusion, the architecture of this web application is a well-designed and efficient combination of front-end and back-end technologies, providing a robust and scalable solution for web development. ReactJS, Bootstrap, D3JS, Expressjs, SQLite, and Rest API result in a user-friendly and efficient web application.

3.1.1 Back-end

Working on server-side software, or what the end user of a website cannot see, is known as back-end development. Back-end developers ensure the website functions correctly by concentrating on databases, back-end logic, application programming interfaces (APIs), architecture, and servers.
They employ code to assist browsers in interacting with databases and storing, comprehending, and erasing data.
This application's server-side uses Expressjs, SQLite, and REST API as the primary technologies. The server receives and responds to client requests through the HTTP request, which follows the RESTful principles, and the server fetches data from SQLite, the database management system, and sends them to the client.
Open-source server environment Node.js is cross-platform and works with various operating systems, including Windows, Linux, Unix, macOS, and more. JavaScript code can be executed outside of a web browser using Node.js, a back-end runtime environment that uses the V8 JavaScript Engine.
Developers can use JavaScript to create server-side scripts and command-line tools using Node.js. The server-side script functionality creates the dynamic web page content before the page is sent to the user's web browser.
As a result, Node.js represents a "JavaScript everywhere" paradigm,[6] bringing web

application development under one programming language rather than segregating it among server-side and client-side scripting languages.

Expressjs is used to implement the application's server. Expressjs is a Node.js web application framework that offers tools for creating and managing HTTP requests. This framework was selected because of its widespread use, simplicity of use, solid structure, ability to communicate with many databases, and ability to use Rest API to transfer data between the system's front-end and back-end. This framework enables the construction of various endpoints or functions that handle routes.
SQLite has been selected as the project's database. An open-source relational database management system is called SQLite. Data for the program is stored, managed, and retrieved using it.
Using a library like sqlite3, the Expressjs application can communicate with the SQLite database by running SQL queries to insert, edit, or retrieve data as needed. On this project's database, ten tables store all the data used in this project. There are also relations between some of the tables in this database. For example, the table which is called "applications" has one primary key, "ID," and two foreign keys, "candidateId" and "positionId" which are used to relate each row of this table to a specific candidate in the "candidates" and a specific position in the "positions" tables.
In each of these rows, foreign keys in the "application" table are the primary key in their table. These relations between the tables are beneficial when a query needs to join two or more tables to fetch the requested data.

A web service can be built using the REST API architectural style, which adheres to a set of rules and uses HTTP methods (such as GET, POST, PUT, and DELETE) to perform activities on resources indicated by URIs.
By establishing endpoints that correspond to the CRUD activities (Create, Read, Update, Delete) for a given resource, the Expressjs application can implement REST API.

Expressjs gets a request from a client, matches it to the correct endpoint, and then invokes the relevant route handler. In order to provide the client with the proper answer, the route handler can use the information from the request and communicate with the SQLite database. Data in a format like JSON should be provided as part of the answer.

It is an example of an API REST request handling in the expressjs:

```
app.get("/API/teams/:teamId", async (req, res) => {
    const teamId = req.params.teamId
    try {
      const teamMembers = await dao.getTeamMembers(teamId);
      res.status(200).json({ teamMembers });
    } catch (err) {
      console.log(err);
      res.status(400).end();
    }
});
```

Listing 1: Async REST API calling in expressjs

This request is used to return the team members of a team. As each team is identified with a unique id, in the body of the request, there is a field "teamId" which the front-end should define. This teamId is passed to the proper function "getTeamMembers()"; based on that, the query will be sent to the database and returns the team members whose teamId is equal to the value passed to the function. After having the data ready as requested, it will be stored in a JavaScript object. For each kind of request, there is a proper JavaScript object already defined on the system, which improves the integrity between the back-end and front-end of the project, as the front-end knows what kind of object it expects to receive.

When the request's status is equal to 200(the server runs the request successfully), the data will be sent to the front-end side of the application in JSON format. JSON is a free and open-source file format and data exchange format that employs text humans can read to store and send data objects made up of arrays and attribute-value pairs. As shown in this example, the type of request is "GET".

While the server is running, it constantly listens for new requests from the other side of the application. After receiving and processing them, respond to those requests with the proper results. The functioning of the server in this matter has been tested during the development phase using a tool which is called POSTMAN API client. In this way, we could be sure that the APIs are working fine and returning the correct results before developing and calling these APIs from the front-end.

3.1.1.1 Dataset

One of the challenges during this project was finding a reliable dataset. To test the system and check its functionality, we needed a proper dataset with specific characteristics that fulfill the needs of our system.

There are not a lot of public datasets about resumes because companies usually keep this kind of information internal to themselves and tend not to make them public. To have reliable results, a dataset with this information is required:

- Skills set of the applicant
- Years of experience of the applicant
- Field of work of the applicant specific on the resume
- The university that they graduated from
- Languages they speak
- Preferably link to their social pages(medium, GitHub, LinkedIn,. . .)

Some data sets were reviewed and tested for this project; each could have advantages and disadvantages.

The first dataset considered was a dataset of resumes categorized into different fields. Some of the drawbacks of this dataset were a wide variety of areas. Since each of the resumes provided by this dataset had to be cleaned and prepared for use by our system, there was not enough knowledge to properly filter the data, and if it were done, it would take much time.

Also, the data provided by this dataset was related to about 10 – 30 years ago, and

in the evolving environment of the job market, they could not provide precise results. The second dataset also included 2296 resumes from different sectors.
The problem with this dataset is that the information on the resumes is insufficient as there was no specific set of skills for most applicants, and they included only general information. Furthermore, the second disadvantage of this data set was being outdated for our usage.

After searching for a suitable dataset, a dataset was found that could provide most of the required information. The project needed more information about the candidate, such as their workplace name, age, gender, and email address. Nevertheless, it was figured out that this information is usually not publicly available. Hence, it was decided to make some changes to the project to receive the best of the available data we have about each candidate.

The data set used in this project was suitable for our use case because it was :

- Categorized in a different field (mostly related to Computer Science)
- Skills set of each candidate was available
- Years of experience of each candidate could be calculated in most cases
- The university of their graduation was specified
- The data was recent and similar to the most recent job position's requirement
- In some cases, link to their social pages was provided

- The languages they speak could be specified

The information provided on this dataset needed to be cleaned and organized for this thesis's purposes. To achieve this goal, a section of this dataset related to the resumes in the field of computer science was chosen. One of the time-consuming parts of this project was extracting useful data in a proper format from this data set, which had to be done manually.

For example, in this project, the years of experience of each candidate were needed, but this kind of information was not provided explicitly in this data set. For some candidates, there was a list of past experiences, including starting and ending years. For each candidate, the total years of experience must be calculated manually. However, in some cases, the year of experience of each candidate had to be estimated from the number of years mentioned for each skill in the candidate's resume.

After data cleaning and getting the required information from this data set, this information had to be added to the database. For this thesis, SQLite was used as the database system. All the extracted information was added to the related table on the database. The number of tables and their relation were decided by analyzing the system's requirements and structure.

3.1.1.2 Conclusion Back-end

A set of possible combinations of different technologies was considered to implement the server side of this application. However, by looking at the advantages and disadvantages of other technologies, it was figured out that because of some reasons such as: being well-documented, having a sizeable supportive community, providing a robust structure, and all working with one programming language, this set of technologies and frameworks are a better fit to our use case.

In conclusion, Expressjs offers a framework for processing client requests, SQLite offers a system for data storage and management, and the REST API offers a standardized mechanism for the client and server to communicate, all of which combine to produce a full-stack web application.

3.1.2 Front-end

Front-end web development creates a website's graphical user interface using HTML, CSS, and JavaScript so visitors can view and interact with it.

ReactJS, Bootstrap, D3JS, and REST API were used to build the front-end of the "Resume Visualization" online application, which can assist users in various ways.

ReactJS offers a straightforward and adaptable framework for constructing user interfaces, while Bootstrap offers a responsive design that makes creating attractive and practical web pages simple.

With the help of the powerful data visualization toolkit D3JS, users may build interactive graphs, charts, and other visualizations. The application can connect to external data sources over REST API and retrieve data uniformly.

The foundation of any website creation process is HyperText Markup Language (HTML), without which a web page would not exist. Hypertext refers to writing that contains links, also known as hyperlinks.

A user will be taken to another web page when they click on a word or phrase that has a hyperlink. Using a markup language, text can be represented as graphics, tables, links, and other things

. The HTML code outlines the general layout of the website. Tim Berners-Lee created HTML. The W3C announced a recommendation for HTML5 on October 28, 2014, making it the most recent version of HTML. This version includes new and effective methods for handling components like audio and video files.

CSS, which manages the presentation part of the site, gives the website its distinct appearance. It keeps track of style sheets triggered by other inputs, such as the screen size and resolution of the device, and sits on top of other style rules. CSS can be added externally or internally as an HTML tag embedding.

Different styles can be created using Cascading Style Sheets or CSS. Any element can be changed, including the padding, margin, font-color, border, and shadow.

Classes, which compile rules about things that may be identified by their class, kind of HTML component, or identity, are the fundamental units of CSS. CSS can apply these criteria repeatedly to keep visually related objects with semantic relationships consistent.

As in this project, Reactjs is used as the main framework for developing the web application's front-end; some CSS rules should be used differently. For example use of CSS classes in Reactjs is like this:

```
function Navigation() {
  return (
    <>
      <div className="navbar">
        <ul>
          <Link to="/">
            <li className="active">Home</li>
          </Link>
        </ul>
      </div>
      <Outlet />
    </>
  );
}
```

Listing 2: Css classes usage in Reactjs

It is part of the code which is used in the Navigation component. As shown, the div uses a custom CSS class with the keyword of "className" which is a standard way of using CSS classes in Reactjs.

A static HTML page can be turned into a dynamic interface using JavaScript, an event-based imperative programming language (as opposed to HTML's declarative language model).

The Document Object Model (DOM), made available by the HTML standard, can be used by JavaScript code to modify a web page in response to events like user input. React is a front-end JavaScript library for creating user interfaces based on UI components that are free and open-source. React is often referred to as React.js or ReactJS. A community of independent developers and businesses manages it, including Meta (previously Facebook).

When creating single-page, mobile, or server-rendered applications with frameworks like Next.js, React can be the foundation. React mainly focuses on state management and presents that state to the DOM, necessitating other libraries for routing and specific client-side functionality when developing React apps.

JSX is a JavaScript extension for user interface design. The XML-based, extensible language JSX can handle HTML syntax with a few tweaks. User interfaces can be created using React Elements, which can be generated using JSX.

The use of JSX in this project lets to use HTML tags combined with some functionality from JavaScript. This feature provides many opportunities for developing a more dynamic web application which also needs less coding and leads to cleaner and more understandable code. The below lines are a use case of JSX in the project.

```
<Button
          variant="primary"
          style={{ position: "absolute", right: "200px" }}
          onClick={doComparison}
          disabled={comparisonList.length === 2 ? false : true}
        >
          {comparisonList.length === 2
            ? "Compare candidates"
            : "Add 2 candidates to compare"}
</Button>
```

Listing 3: using JSX for creating a Button

It is a use case in which the Button component from the Bootstrap framework is used inside the code. This component is a regular HTML tag in this section and can have some attributes. Bootstrap defines the "variant" attribute to give the button some specifications, such as its color.

The "style" is used to give the component some CSS styles which should be written in a JavaScript object. The "onClick," which is the Reactjs version of the "on click," is used to define the system's behavior when this button is clicked. After clicking on the button, a function called "comparison" is called.

Furthermore, for the last attribute, "disabled," one of the most valuable features of Reactjs is used. For providing value to the "disabled" attribute, a "ternary condition" is used. A ternary condition is a short form of an "If-else" statement that allows returning different values if the condition is met. In this example, if the length of "comparisons" is equal to two, "false" will be returned, and for any other values, "true" will be returned.

As explained, using JSX enables the developer to combine HTML, JavaScript, and CSS code to have a more robust and cleaner code that is also easier to maintain.

A React component is the primary building block of a React application. It uses JSX and React components to build its user interface. A React component is fundamentally either a JavaScript class (which extends the React. component class) or a JavaScript function.

A React component has all its characteristics, state management, life cycle, and event handler. React components can be used to do both simple and complicated reasoning. The following are just a few ways that React architecture helps with web development:

- React contributes to developing component-based software architecture, facilitating code reuse and maintenance.
- It permits the maintenance of a global state variable using state management tools like Redux.
- Since the React design allows for component building, more code may be added more effectively as the project grows.
- Because the React architecture is component-based, unit testing is more effortless.

In this project, using Reactjs components leads to a better structuring of the application and easier data management. Reactjs components are described below by an example from the project.

```
import Container from "react-bootstrap/Container";
import Row from "react-bootstrap/Row";
import API from "../API/API";

import { useEffect, useState } from "react";
import PositionsList from "./PositionsList";

function Home() {
  const [positions, setPositions] = useState([]);
  useEffect(() => {
    API.getPositions()
      .then((positionsInfo) => {
        setPositions(positionsInfo);
      })
      .catch((err) => console.log(err));
  }, []);
  return (
    <>
      <Container fluid>
        <Row>
          <PositionsList
            positions={positions}
          ></PositionsList>
        </Row>
      </Container>
    </>
  );
}

export default Home;
```

Listing 4: Function Component in Reactjs

it is the code related to the Home component of the project. This component is the first page of the application, which shows a list of available job positions. Some components, functions, and Reactjs features are imported on the first five rows of this component. Reactjs components are of two types: Class components and Function components.

Class components are not used widely anymore because of their complexity. It is a function component that provides features to let the user handle the data better. The most important feature of function components is the Hooks. Hooks are some built-in functions of Reactjs which let the user manage the code in different lifecycles. The most used hooks are "useState" and "useEffect".

Reactjs' useState hook enables the user to include a state variable in the components. Components will be updated whenever a state is updated, or its data is changed. Another used hook in this component is "useEffect". useEffect is a Reactjs built-in function that will be run whenever its dependency changes or only in some lifecycles of the component.

For example, in this component, the useEffect is used without any dependency, meaning it only runs the first time the component is mounting. These hooks useState and useEffect, are used in this way.

One state in this component is " positions", and one function to update this state is called "setPositions". In the useEffect hook, a function from the API module is called to retrieve all positions from the database. After receiving the positions' data, the setPositions function is called.

A list of the positions fetched using the API is passed to this function, and new values will be set to the related state, which is positions. This useEffect only runs the first time the component renders, as the data related to the positions will not change inside the component. Finally, the data stored in the state is passed to the PositionList component as a prop

3.1.2.1 Design Criteria

This project aimed to have a reliable and clean visual design that provides information to the user in an understandable way. In this project, a large amount of data and information is shown to the user, such as different charts, tables, and windows.

Hence Having a visual design that can show all this information to the user and prevent him from becoming confused is essential. To approach this goal, the visual design should follow some specific rules in the field of user experience in the user interface.

The user experience (UX) describes how a person engages with and utilizes a system, product, or service. It encompasses how useful, simple, and effective something is in someone's eyes. There are some examples of UI and UX that a proper design must follow. For example, the distance between elements, the variety of colors, and many other factors. Several tools and libraries provide a framework and components that make designing a web page easier.

Bootstrap is selected as the framework for this project to design the whole application. Because of its familiar design for the users, it is used in many different projects (19.2 percent of websites utilize Bootstrap, according to W3Techs.), and because it improves the whole user interface and user experience.

Bootstrap is A free, open-source CSS framework designed for front-end web development that prioritizes mobile responsiveness. It includes design templates for typography, forms, buttons, navigation, and other interface elements in HTML, CSS, and (optionally) JavaScript.

An HTML, CSS, and JS package called Bootstrap makes it easier to create educational web pages than online applications. Applying Bootstrap's color, size, font, and layout selections to that project is the primary goal of adding it to a web project. Therefore, the main determinant is whether the responsible developers like those options.

All HTML components have basic style declarations once Bootstrap is introduced to a project. As a result, texts, tables, and form components appear consistently in all web browsers. In order to further modify the look of their content, developers can use the CSS classes defined in Bootstrap. For instance, Bootstrap offers tables in light and dark colors, page headings, pull more noticeable quotes, and content that is highlighted.

The layout components of Bootstrap are the most noticeable since they have an impact on the entire web page. Every other element on the page is contained in the fundamental layout element, which is referred to as the "Container." A fixed-width container or a fluid-width container are the two options for developers. The former employs one of the five predetermined fixed widths based on the size of the screen displaying the page, whereas the latter always fills the width with the web page:

- Smaller than 576 pixels
- 576–768 pixels
- 768–992 pixels
- 992–1200 pixels
- Larger than 1200 pixels

Following the placement of a container, other Bootstrap layout elements build a CSS Flexbox layout by specifying rows and columns. One CSS file, three JavaScript files, and a precompiled version of Bootstrap are readily available for use in any project. However, developers can apply additional customization and size optimizations using Bootstrap in its raw form. The developer can delete unnecessary components, apply a theme, and alter the uncompiled Sass files because this raw form is modular.

3.1.2.2 Data Visualization

The main goal of this project is to visualize data for the user to help him better understand them. One of the approaches for visualizing data is using charts and plots as they graphically present data, making it easier to understand the relationship between them. One of the most used tools for creating charts in web applications using JavaScript is D3.js.

A JavaScript library called D3.js (often called D3, short for Data-Driven Documents) is used to create dynamic, interactive data visualizations in web browsers. It adheres to the Cascading Style Sheets (CSS), HTML5, and Scalable Vector Graphics (SVG) standards. It is the framework that replaces the earlier Protovis one.

Version 2.0.0 was launched in August 2011, and its development was recorded that year. D3 was transformed from a single library into a collection of smaller, modular libraries that can be used independently with version 4.0.0 in June 2016.

To choose items, create SVG objects, style them, or add transitions, dynamic effects, or tooltips, the D3.js package uses pre-built functions. Additionally, CSS can be used to style these objects. D3.js routines can build text/graphic charts and diagrams from large datasets tied to SVG objects. The data can be in various forms, including JSON, CSV, or geoJSON, although JavaScript utilities can be built to read other data formats if necessary.

D3.js array operations are designed to supplement JavaScript's array support, which includes iteration methods like filter, every, forEach, map, some, reduce, and reduceRight, as well as mutator methods like sort, reverse, splice, shift, and unshift.

3.2 How the application works

A list of job positions is stored in a table named positions in the database. Each position includes data such as required skills, minimum experience, the related team to this position, and other information. There is another table in the database called "candidates," which includes data about each candidate.

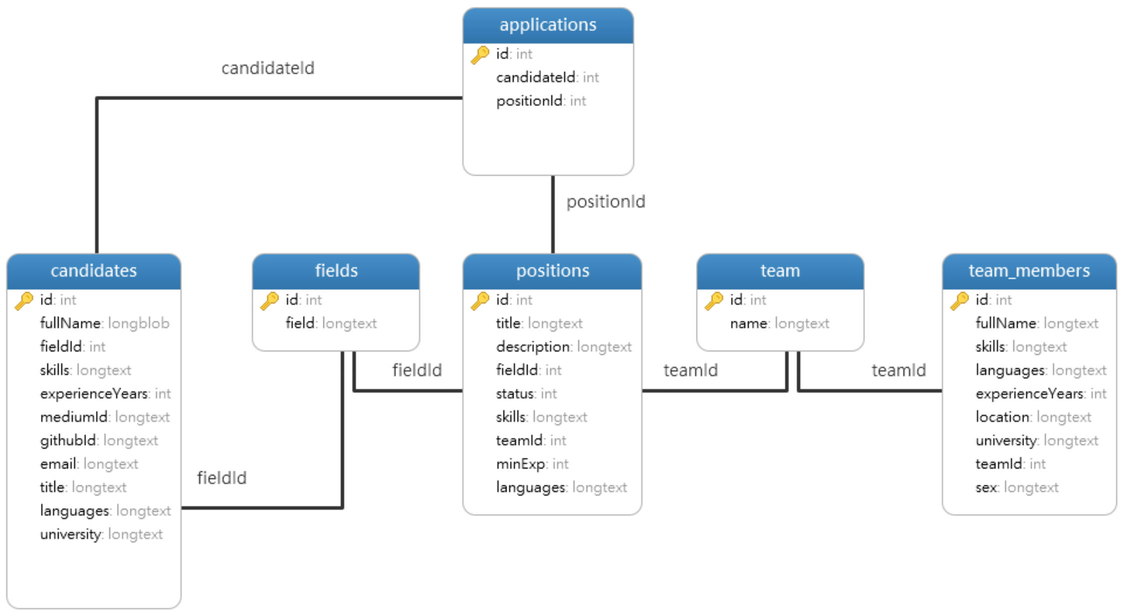


Figure 3.2: Database Entity Relation Diagram

On the other hand, a table called applications works as a bridge between the positions table and the candidates' table—each row in the applications table shows which candidates have applied for which position. On the main page of the application, there is a list of all the positions in the system, This list of positions is fetched from the server using a Rest API, which receives the latest positions available on the database.

Some information related to that position is shown to the user. This information includes the number of applicants for this position, the status (if it is active or closed), required skills, minimum experience, and the position description.

On this page, some functionalities are provided for the user, such as creating a new position, creating a new team, and adding new team members to each team. As each position is connected to a team in this system and each team includes some team members, these features help the user manage the positions better.

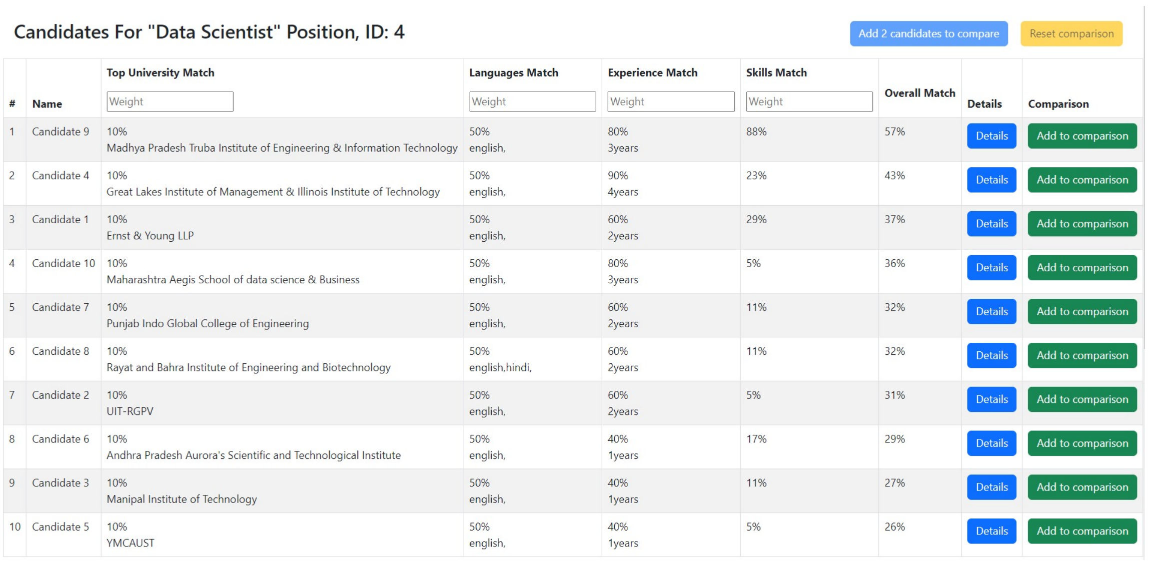
First, the user needs to add a new team to the system. Adding a new team is easy as the user only needs to click the related button "Add a new team". On the new window shown, provide the team's name. after clicking on the "Submit" button, the provided information by the user will be stored on the database on the "team" table.

After adding a new team user needs to add a new position. In this case, the user can click the related button "Add a new position." After clicking on this button, a new window will be shown to the user. In this window, there is a form that consists of some fields. These fields are information about the new position that the user is adding. After clicking on the "Submit" button, the provided information by the user will be stored in the database on the "positions" table.

Also, the user can add some members to each team he has created already. This feature will be helpful when each candidate's information is compared with the team members. The user must click the "Add a new team member" button to add team members. After clicking on this button, a new window will be shown to the user. In this new window, there is a form that lets the user put the information about the team member. After clicking on the "Submit" button, the provided information by the user will be stored in the database on the "team$_memebrs" table.$

The user can click on the "details" button and see a list of all the candidates who have applied for this position.

The new page contains detailed information about all the applicants for this job position. The page could be separated into three parts: a table of candidates ranked by their score, a table of candidates ranked based on their Match to the team members of the company, and some charts and plots for visualizing the candidates' information. The first table includes some information about the candidates, In each row, there are nine columns.


Candidates For "Data Scientist" Position, ID: 4
Add 2 candidates to compare
Reset comparison

#	Name	Top University Match	Languages Match	Experience Match	Skills Match	Overall Match	Details	Comparison
		Weight	Weight	Weight	Weight			
1	Candidate 9	10% Madhya Pradesh Truba Institute of Engineering & Information Technology	50% english,	80% 3years	88%	57%	Details	Add to comparison
2	Candidate 4	10% Great Lakes Institute of Management & Illinois Institute of Technology	50% english,	90% 4years	23%	43%	Details	Add to comparison
3	Candidate 1	10% Ernst & Young LLP	50% english,	60% 2years	29%	37%	Details	Add to comparison
4	Candidate 10	10% Maharashtra Aegis School of data science & Business	50% english,	80% 3years	5%	36%	Details	Add to comparison
5	Candidate 7	10% Punjab Indo Global College of Engineering	50% english,	60% 2years	11%	32%	Details	Add to comparison
6	Candidate 8	10% Rayat and Bahra Institute of Engineering and Biotechnology	50% english,hindi,	60% 2years	11%	32%	Details	Add to comparison
7	Candidate 2	10% UIT-RGPV	50% english,	60% 2years	5%	31%	Details	Add to comparison
8	Candidate 6	10% Andhra Pradesh Aurora's Scientific and Technological Institute	50% english,	40% 1years	17%	29%	Details	Add to comparison
9	Candidate 3	10% Manipal Institute of Technology	50% english,	40% 1years	11%	27%	Details	Add to comparison
10	Candidate 5	10% YMCAUST	50% english,	40% 1years	5%	26%	Details	Add to comparison

Figure 3.3: Overall ranking of candidates

- In the first and second columns are row numbers and names of the candidates.
- 3rd column is called "Top University Match." As for companies in different countries, it could be important to know from which university the applicant graduated. Sometimes, they give priority to the people coming from some specific universities this field is provided. In this field, besides showing the name of the university from which the candidate graduated, we also check

on the university. There is a list of 400 top universities around the world provided in the system. The name of the candidate's university is compared with this list, and if there is a match between them, a given score will be given to the candidate. It could be helpful for job positions and companies looking to hire people who graduated from top universities or the university local to the company. There is a number in percentage and the name of each candidate's university. This number could vary between 10 to 100. There is a formula for calculating this number. The application compares the university which the candidate has graduated from with a list of 400 top universities and gives different scores based on the results of this comparison:

  - 100 points if the university is among the top 50 universities on the list.
  - 90 points if the university is ranked between 50 and 100 on the list.
  - 80 points if the university ranks 100 and 150 on the list.
  - 70 points if the university ranks 150 and 200 on the list.
  - 60 points if the university ranks 200 and 250 on the list.
  - 50 points if the university ranks 250 and 300 on the list.
  - 40 points if the university is ranked between 300 and 350 on the list.
  - 30 points if the university ranks 350 and 400 on the list.

- 4th column in the first table is called "Language Match. There are preferred languages for each position in this application. As different companies in an international environment have some preferences on the number of languages the applicant can speak, it is essential to provide this information to the user. For example, a company working with clients from foreign countries needs a person familiar with specific languages, which could be an advantage for the applicant. In this field, the preferred languages of the position are compared with the candidates' languages and based on that, a point is given to each candidate. The formula to calculate this number is as below for each language match between the candidate and the job position:

Language point = (candidate's languages)*(100/number of preferred languages)

For example, if there are three preferred languages for this position and the candidate only knows two of those languages, the language point would be :

Language point = 2*(100/3) => 66.66

- 5th row in the table is called "Experience Match, " a number is shown as a percentage. As mentioned, each position has a "minimum experience" field that offers the minimum experience required for this job position. Different job positions need different levels of experience, and the company needs to hire a person who best fits that position. In this field, the experience years of the applicant are compared to the minimum required experience of the position. Based on a formula explained below, a score is given to the candidate. A higher number means a better match.
  - 100 points if the candidate has two or more years of experience than the required experience years for the position.

  - 90 points if the candidate has one year of experience more than the required experience years for the position.
  - 80 points if the candidate has years of experience equal to the experience years necessary for the position.
  - 60 points if the candidate has one year of experience less than the required experience years for the position.
  - 40 points if the candidate has two years of experience less than the necessary experience years for the position.
  - 10 points if the candidate has three or more years of experience less than the required experience years for the position.

- The 6th column shows the "Skills Match," a percentage number. As mentioned earlier, each position requires some skills, and each candidate has a set of skills. The application compares two candidates' skill sets and the position with each other. Based on the result of this comparison, a specific amount of points are given to the candidate. A higher score means the candidate has more skills than the position's required skills. This number is calculated based on the below formula:

  Skills point = (candidate's matched skills)*(100/number of required skills)

  For example, if the position requires four skills and the candidate has 3 of them in his skill set, the Skills point would be calculated as below:

Skills point = 3 * (100/4) => 75

The final number is shown as a percentage that indicates the Match between the candidate's skills and the position's required skills.

- 7th column is called "Overall Match." In this column, a number is shown. This number is the average of all the last columns. It shows the overall Match of the candidate with the position based on the different factors present in the table: University Match, Experience Math, Languages Match, and Skills Match.

  As it is visible in the table, each column in this table has an input field that is called "weight." The "weight" is one of the features that could be useful for the user as the user can give different weights to each of the columns that affect the final results of the other comparison.

  When calculating the average of these different factors to get the value, By default, the overall Match's weight of each factor is equal to one. However, as different users for different positions might have other priorities, there is this possibility to change the importance of each element to recalculate the score of the candidates. Changing the weight for each factor affects the final results, as it is shown in the formula below:

```
    overallMatch += (uniMatch / 4) * weight.uni;
    overallMatch += (expMatch / 4) * weight.exp;
    overallMatch += (skillsMatch / 4) * weight.skills;
    overallMatch += (langMatch / 4) * weight.lang;

    overallMatch =
      overallMatch /
      ((parseInt(weight.uni) +
        parseInt(weight.exp) +
        parseInt(weight.skills) +
        parseInt(weight.lang)) /
        4);
```

Listing 5: The code to calculate overallMatch compared to the position's info

As seen from the code's screenshot, each score calculated before is divided by 4, which is the number of factors included in calculating the average(overall Match). The result will be multiplied by the weight of each element which the user could modify.

After that, the result will be divided by the sum of all weight values of different factors divided by 4. This way, it could be sure that the final result would be a number below 100 and could be shown as a percentage. Whenever the user changes each element's weight in the table, the calculations will be rerun, and the new results will be shown to the user in real time.

The order of candidates shown in the table could be changed. By default, the candidates are ordered by their "overall Match" score in an Ascending mode, which means whoever has more "overall match" score is placed higher in the table.

After modifying the user's weight of each table column, the table will be sorted again to ensure that it is shown the correct result. Also, the user can change the order of the table to see different results. By clicking on each column, the table will be automatically sorted based on the outcome of that specific column.

For example, if the user clicks on the "University Match" column, the table will be sorted so that the candidates with a higher score of "University Match" are placed at the top. This feature helps the user to sort the table based on his preferences and gives him the ability to get the most out of the data which is shown to him. as for each position, some of the elements might be more important than others, and this feature is provided to answer to this need of the user.

- In the 8th column, there is a button called "Details." This button is supposed to the information related to each candidate in a pop-up window. A new window will appear on the screen by clicking on this button. There are details about the selected candidate on a table in this window, such as his skills, languages, overall score, github id, and link to his Medium page.

  On the skills and languages row, there is a feature that distinguishes the items which match the job position requirements. For example, all the skills and languages of the candidate, by default, are shown in a blue pill. Nevertheless, the items that match the position's requirements are shown in a green pill.

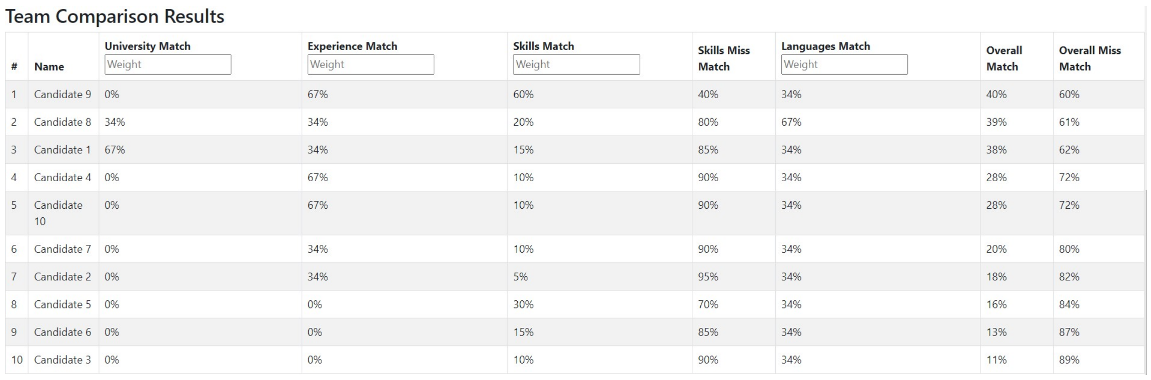
  In this way, we provided a visualized way for the user by a quick scan of the information displayed on the screen to understand which skills and languages of the candidate match the job position requirement. Also, two rows show the GitHub id of the candidate and are linked to the candidate's GitHub profile if the applicant's GitHub id is available.

  Moreover, the other row shows the Medium username, which is connected to the URL of his Medium page if available. By providing this information, the user can assess all the details of the applicant quickly and easily. As for many IT and Computer Science positions, this information could demonstrate the candidate's real-world skills.

- In the 9th column, a button is provided for the user to add candidates to a comparing table. Each comparison could be made between two different candidates. After adding two candidates to the comparison, the related button on the top of the page will be enabled. Furthermore, let the user see the information about the two selected candidates beside each other, which helps him to have a better idea of which candidate could be a better fit for this position.

  In the comparison window, which will be appeared after clicking on the "Compare Candidates" button, the information of two candidates will be shown in the same format as the details window and let the user see the result cleanly and efficiently.

That was all functionality and details about the first section of this page, which was about scoring and sorting the candidates and comparing them with each other. In the next section of the page, there is a table with different usage. In a real-world situation, each person a company hires is likely to work in a team, and we can conclude that each published job position is related to a team inside the company.



Team Comparison Results

#	Name	University Match Weight	Experience Match Weight	Skills Match Weight	Skills Miss Match	Languages Match Weight	Overall Match	Overall Miss Match
1	Candidate 9	0%	67%	60%	40%	34%	40%	60%
2	Candidate 8	34%	34%	20%	80%	67%	39%	61%
3	Candidate 1	67%	34%	15%	85%	34%	38%	62%
4	Candidate 4	0%	67%	10%	90%	34%	28%	72%
5	Candidate 10	0%	67%	10%	90%	34%	28%	72%
6	Candidate 7	0%	34%	10%	90%	34%	20%	80%
7	Candidate 2	0%	34%	5%	95%	34%	18%	82%
8	Candidate 5	0%	0%	30%	70%	34%	16%	84%
9	Candidate 6	0%	0%	15%	85%	34%	13%	87%
10	Candidate 3	0%	0%	10%	90%	34%	11%	89%

Figure 3.4: Team comparison table

The idea was to let the hiring people compare the job applicants with their team members already working in the company. It could be said that each job position and each company has its priorities. There are cases in which the new person who is adding to the team should have the same set of skills and experience as the other team members, and there are cases that the company needs someone who has a different set of skills and knowledge. Imagine hiring a new intern to work with a team of seniors.

The candidate is not supposed to have the same years of experience as the team members. So to answer different needs and respond to different possible situations, in this section, we tried to provide the information so that the user can have a better choice and view of all candidates. For this reason, for each job position in the system, there is a team related to that, and each team has some team members whose information is stored in the database in case we need to compare them with the job applicants.

Each column in this table has an input field that lets the user give them weight. As different users and positions might have other priorities, this feature enables them to provide each factor of the comparison differently, leading to different results and helping the user make a better decision. Each field of this table is explained here.

- The first and second columns show the candidate's row number and name.
- 2nd column is called "University Match." In this column, the candidate's university is compared with all team members. to see if they graduated from the same university. If they graduate from the same university, some points will be added to the University Match of the candidate. This number is calculated below for each Match between the candidate's university and each team member's university:

  University Match = (number of matched university) * (100/ number of team members) The final result will be shown as a percentage.

- 3rd column is called "Experience Match." In this column, the candidate's years of experience are compared with all the team members' years of experience. If the team member's years of experience are equal to or less than the candidate's years of experience, some points will be added to the Experience Match. The formula which is used to calculate this number is shown below:

  Experience Match = (number of matched experience) * (100 / number of team members)

- 4th column shows the "Skills Match." It is a number that indicates the Match between the candidate's skills and all the team member's skills. To calculate this number, in the first step, all the skills of team members are added to an array. Then each candidate's skill is compared with this array of skills, and for each Match, some points will be added to the Skills Match. The Skills Match is calculated as below:

> Skills Match = (number of matched skills) * (100 / number of team members)

- 5th column, "Skills Miss Match" shows the exact opposite of the Skills Match value. In this column, we decided to show how different the skills set of the candidates are compared to the skills set of the team members. It could be helpful for the user to decide if the selected person for this position needs to have the same skills as his team members or if he needs to bring new insights and a bunch of skills to the team.

- The 6th column shows the "Languages Match" In this column, the languages of the candidate will be compared with the languages of all team members to see how similar their languages are and if the new team member can communicate with each other appropriately.
  This kind of information and insights could be helpful for companies that hire people from different nationalities and must be sure that the new people joining the company can communicate effectively with the other people in the teams and company.
  To calculate "Languages Match" score, all team members' languages are first added to an array, and then the candidate's languages are compared to them one by one. The formula to calculate this number is :

  > Languages Match = (number of matched languages) * (100 / number of team members)

- 7th field on the table shows the Overall Match. This number shows how matched this candidate is to the other team members he might work with. After calculating all the different comparison fields, it is time to show the user the candidate's overall score in this comparison. To calculate this number, we need to consider the weight of each field that could be modified by the user and is one by default. The formula to calculate this number is as bellow:

```
overallMatch += (uniMatch / 4) * teamWeight.uni;
    overallMatch += (expMatch / 4) * teamWeight.exp;
    overallMatch += (skillsMatch / 4) * teamWeight.skills;
    overallMatch += (langMatch / 4) * teamWeight.lang;
    overallMatch =
      overallMatch /
      ((parseInt(teamWeight.uni) +
        parseInt(teamWeight.exp) +
        parseInt(teamWeight.skills) +
        parseInt(teamWeight.lang)) /
        4);
```

Listing 6: The code to calculatie overallMatch compared to team members

The overall Match is the average of all the other elements, calculated before, then each element's weight is multiplied by the result to apply the user choice.

- The 8th field on this table shows the "Overall Miss Match." This field is the exact opposite of the Overall Match Value. This field is for cases where the user needs to see how different the candidate is compared to the other team members in all functional aspects.

As mentioned before, sometimes a company needs to add a new person to a team who has a different skill set, experience, education, and language to fulfill this kind of need in this field is provided. As explained above, we provided user information about comparing candidates to team members in this section. As in most cases, the selected candidate will work in a team. It is helpful for the user to understand better how this possible team member will fit into the team.

The first part of the page was about two tables showing the results of different comparisons between candidates and the position requirements. The third section is a place to visualize the data we have crafted from each candidate to help the user explore all available information and decide better.

This section provides the user with seven different plots and diagrams, each showing another part of the data.

- 1st plot here is a bar chart. This bar chart shows the overall ranking of the candidates sorted from higher to lower based on each candidate's Overall Match score, which has been calculated before. This plot is provided to give the user a quick and easy way to review all the candidates in a chart and better understand the job applicants' situation.

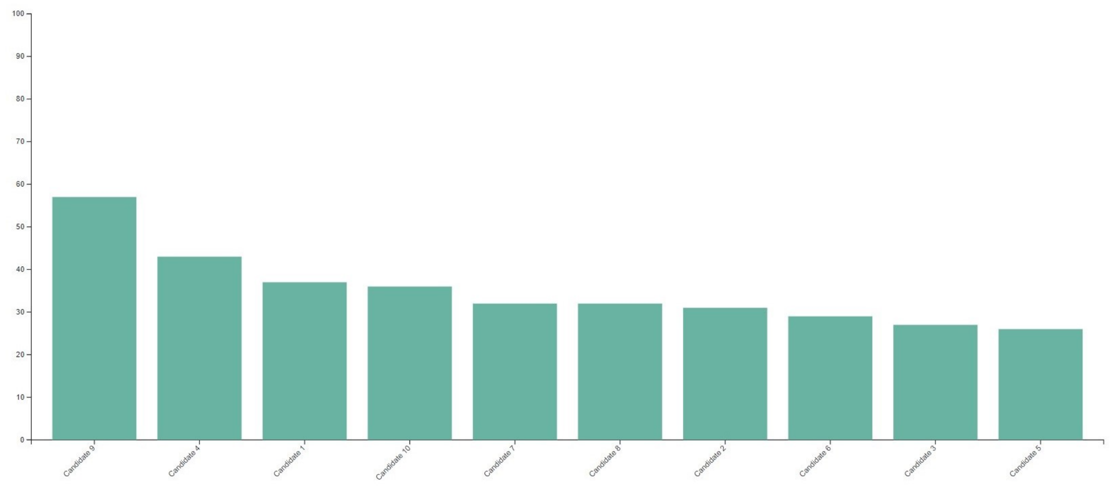


Figure 3.5: Overall ranking of the candidates

- 2nd plot is a bar chart. This bar chart shows all the candidates' skills in one place. This chart shows how many candidates have a specific skill, giving the user a better view of all applications for this job. This chart is sorted, and the higher values are shown on the left side of the chart.

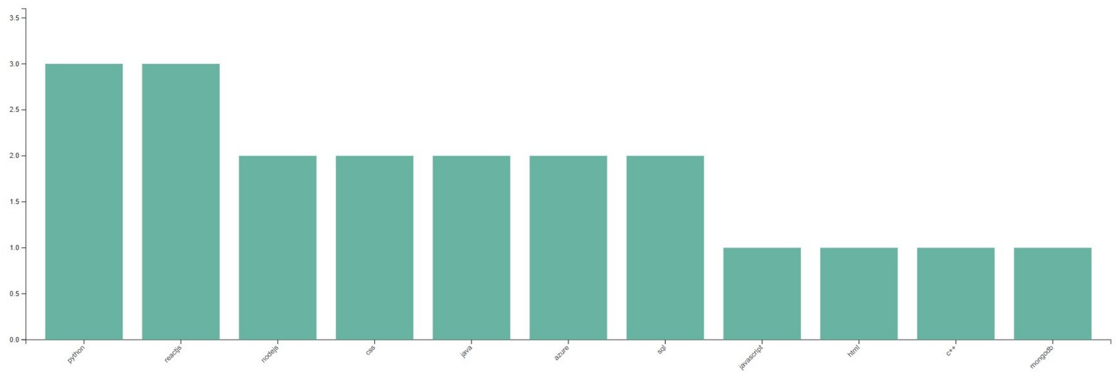


Figure 3.6: Candidates skills bar chart

- 3rd plot is a pie chart. This pie chart shows the data related to the team comparison. In this pie chart, the user can see each candidate's name and their relative overall match score compared to the other team members. The data plotting this pie chart is sorted from the highest to the lowest. As a result, the candidate with the highest score is shown in the first part of the pie chart. Also, another pie chart in this section shows the overall miss match value of each candidate compared to the team members. This way, the user has a complete view of the candidate's situation compared to the team members.

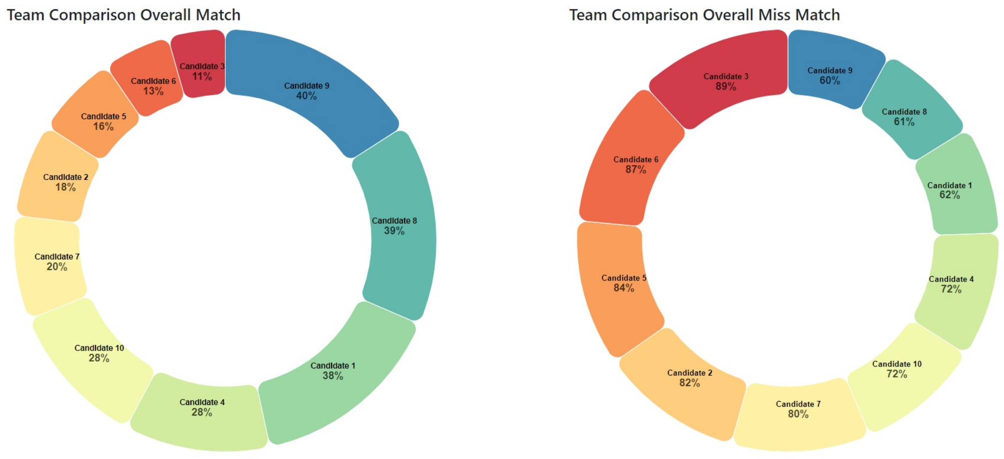


Figure 3.7: Team comparison pie chart

- 4th plot is a bubble chart. This bubble chart shows the skills of all the candidates in a way that the skills most repeated in the different applicants' skills are shown in larger bubbles with the number of candidates who have this skill in their skill set. This bubble chart provides information to the user graphically and interestingly, which is also more understandable.

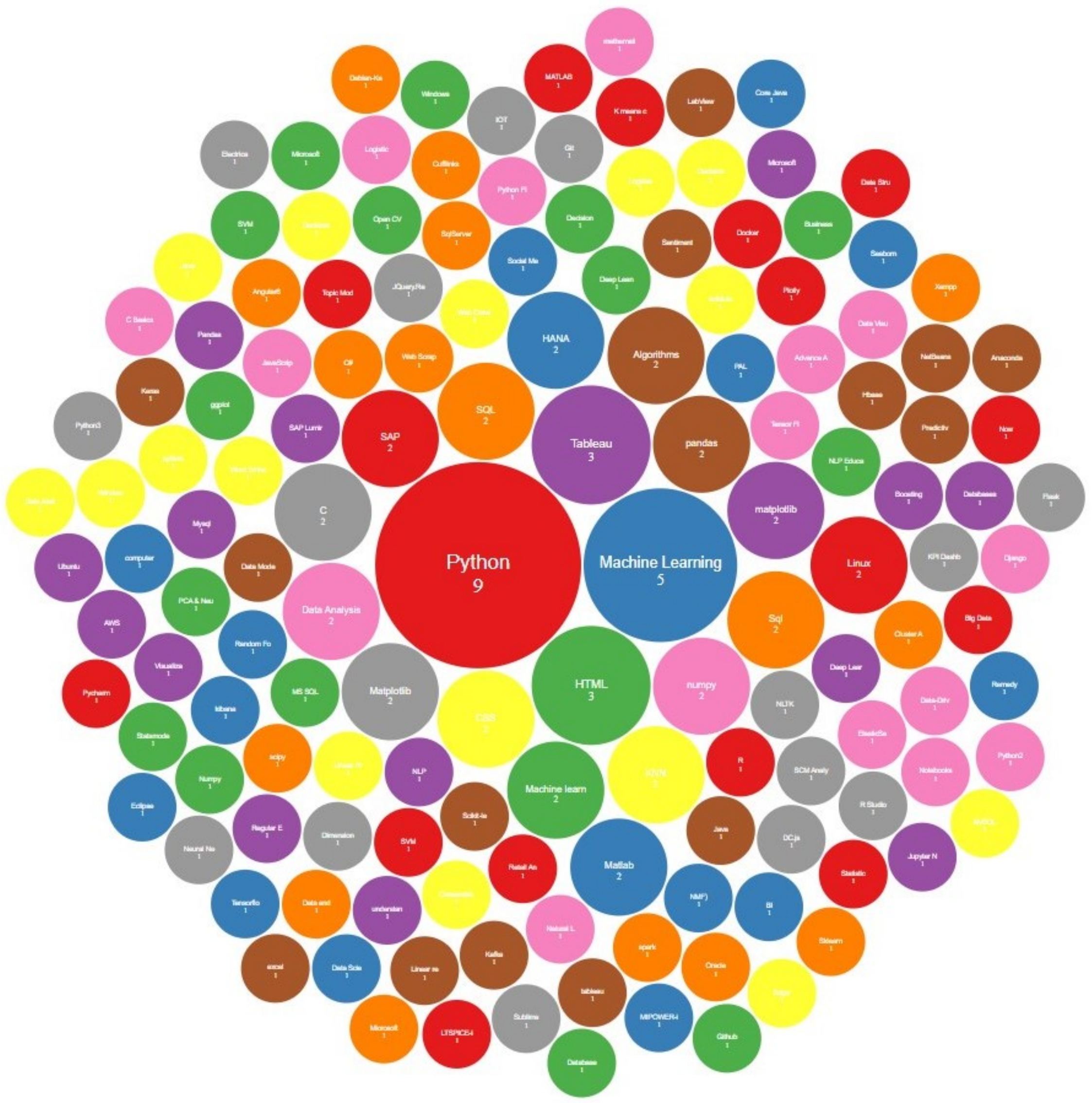


Figure 3.8: Candidates skills bubble chart

- 5th plot is a bubble chart. This bubble chart shows the languages of all the candidates in a way that the languages most repeated in the different candidates' languages are shown in larger bubbles with the number of candidates who have this language in their set of languages. This bubble chart provides information to the user graphically and interestingly, which is also more understandable.

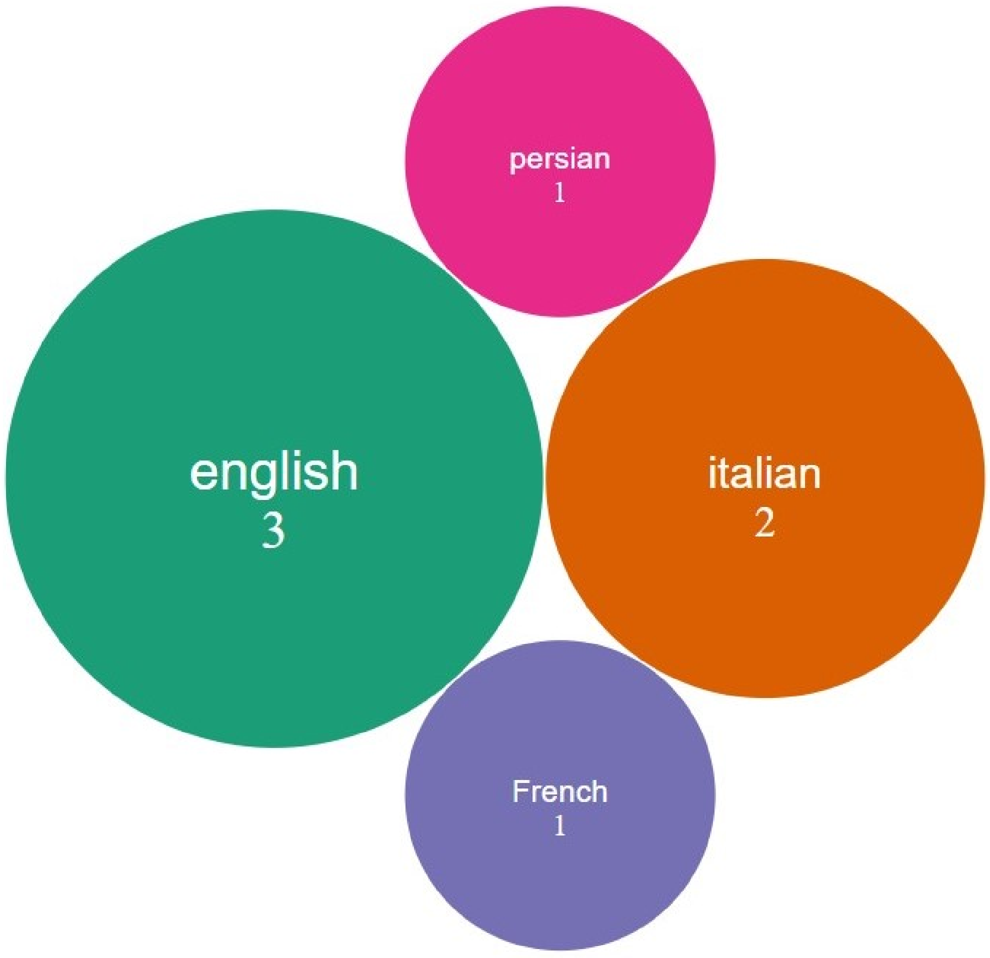


Figure 3.9: Candidates languages bubble chart

- 6th plot is a scatter plot. This plot provides information about the relationship between the experience years of the candidate and each candidate's overall score. This plot shows some important results to the user. For example, the user can see how years of experience affect the overall score of all candidates for this position.

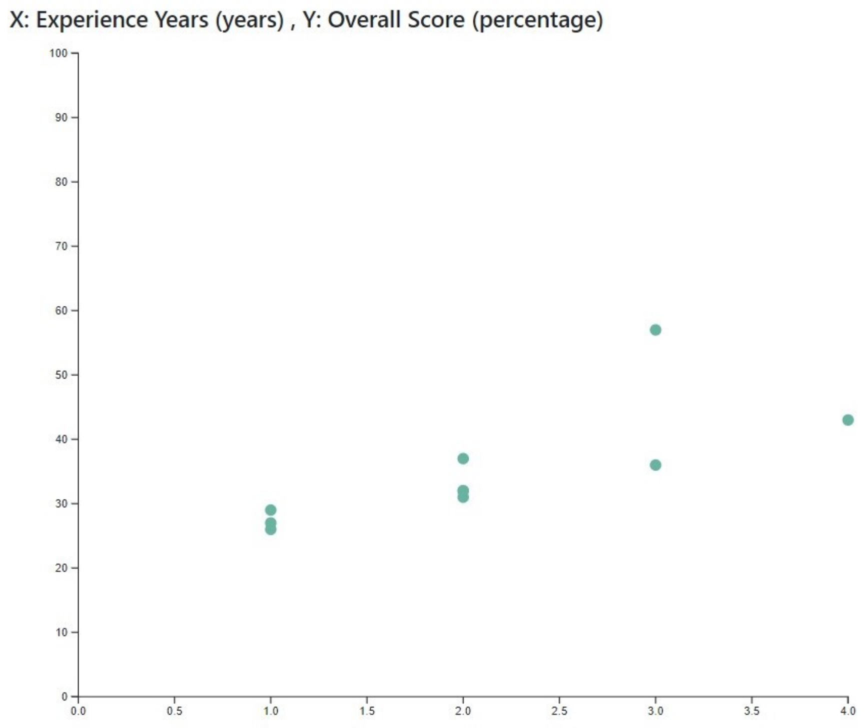


Figure 3.10: Candidates experience to overall score scatter plot

- 7th plot is a scatter plot. This scatter plot shows information about the relationship between the experience years and the skills match score of each candidate. Using this plot, the user can understand how a candidate's years of experience could be related to his skills Match score and the overall trend among all the applicants for this job position.

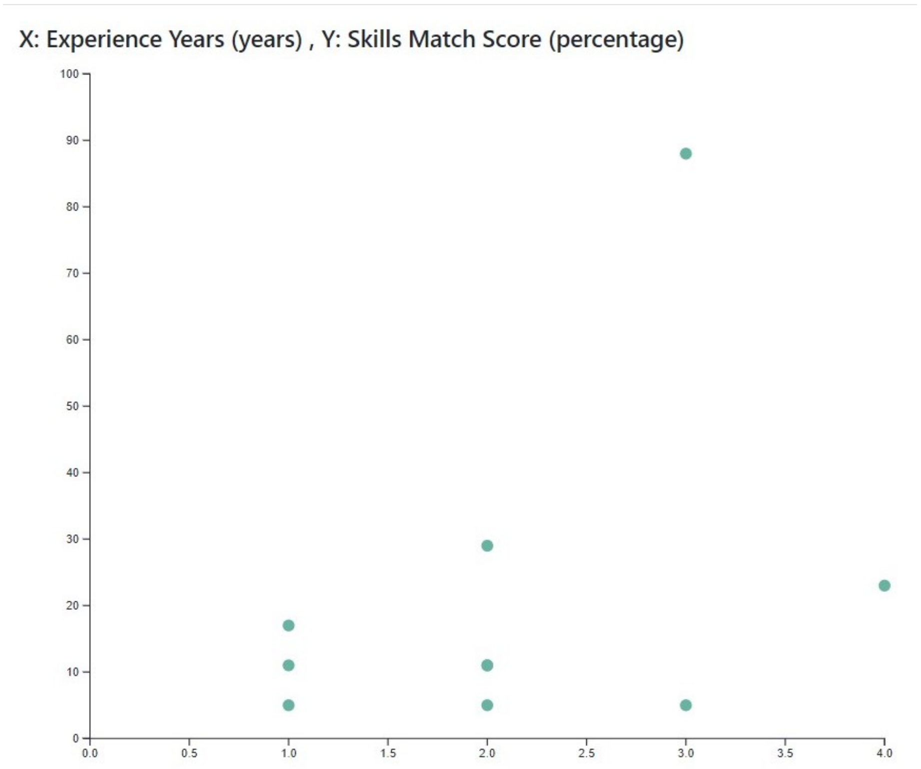
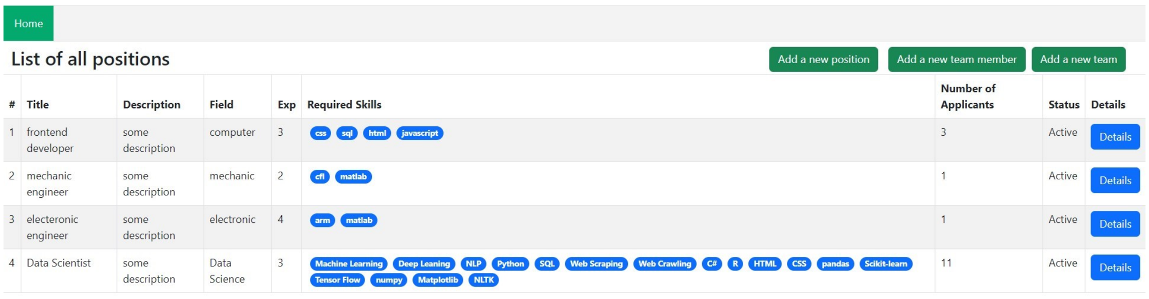


Figure 3.11: Candidates experience to skills scatter plot

As the above lines explain, some plots and charts are provided for the user in this section. Each focuses on different aspects of the available data about the candidates, as data visualization is one of the essential tools when working with different kinds of data. Plots could be helpful for the user to understand better the situation of all the candidates for each job position and make them able to select the person who has the most requirements for the job position.

4 User Interface

In designing different sections of this application, the main focus has always been on ease of use and straightforward design. Different sections of the application are designed so that most of the information the user needs will be shown at a glance. The first page of the application contains a list of all the job positions available on the system. When this page loads, an API is called and a request is sent to the server. The server fetches the requested data, which is a list of positions and their related information, and returns it to the client. A list of all positions with their details will be shown in a table to the user.

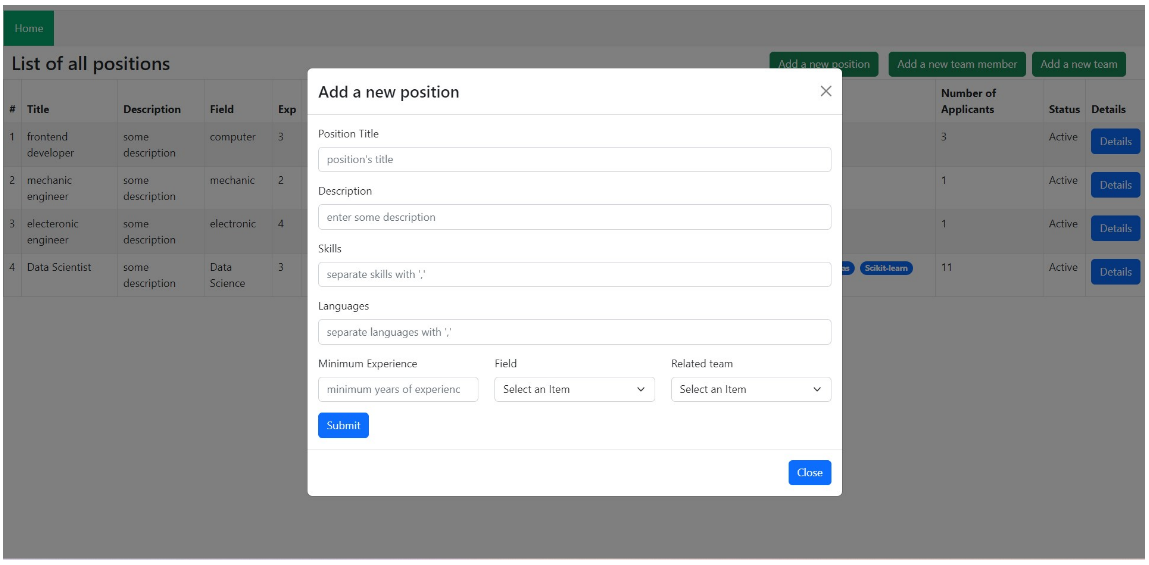


List of all positions

#	Title	Description	Field	Exp	Required Skills	Number of Applicants	Status	Details
1	frontend developer	some description	computer	3	css sql html javascript	3	Active	Details
2	mechanic engineer	some description	mechanic	2	cfl matlab	1	Active	Details
3	electeronic engineer	some description	electronic	4	arm matlab	1	Active	Details
4	Data Scientist	some description	Data Science	3	Machine Learning Deep Leaning NLP Python SQL Web Scraping Web Crawling C# R HTML CSS pandas Scikit-learn Tensor Flow numpy Matplotlib NLTK	11	Active	Details

Figure 4.1: list of all positions in the system

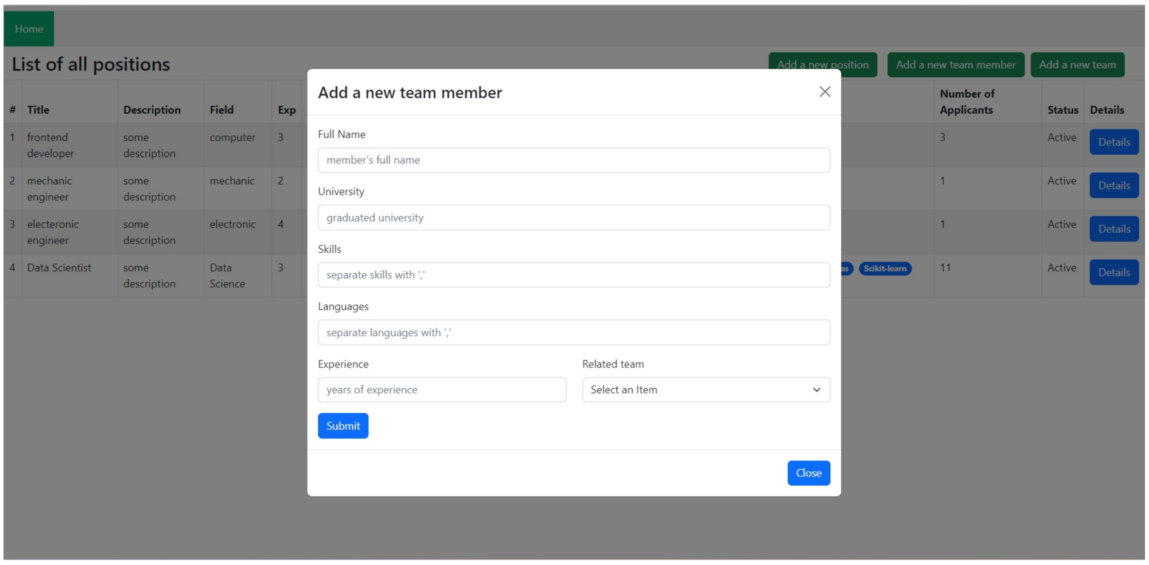
There are also three different buttons on the top of the page. The first button is for adding a new position. By clicking on this button, a new window will be popped out, and the user can create a new position by filling out the form, which also includes setting the "Related Team" and "Field" of the position and clicking on the "Submit" button.



#	Title	Description	Field	Exp	Number of Applicants	Status	Details
1	frontend developer	some description	computer	3	3	Active	Details
2	mechanic engineer	some description	mechanic	2	1	Active	Details
3	electeronic engineer	some description	electronic	4	1	Active	Details
4	Data Scientist	some description	Data Science	3	11	Active	Details

Figure 4.2: Adding a new position

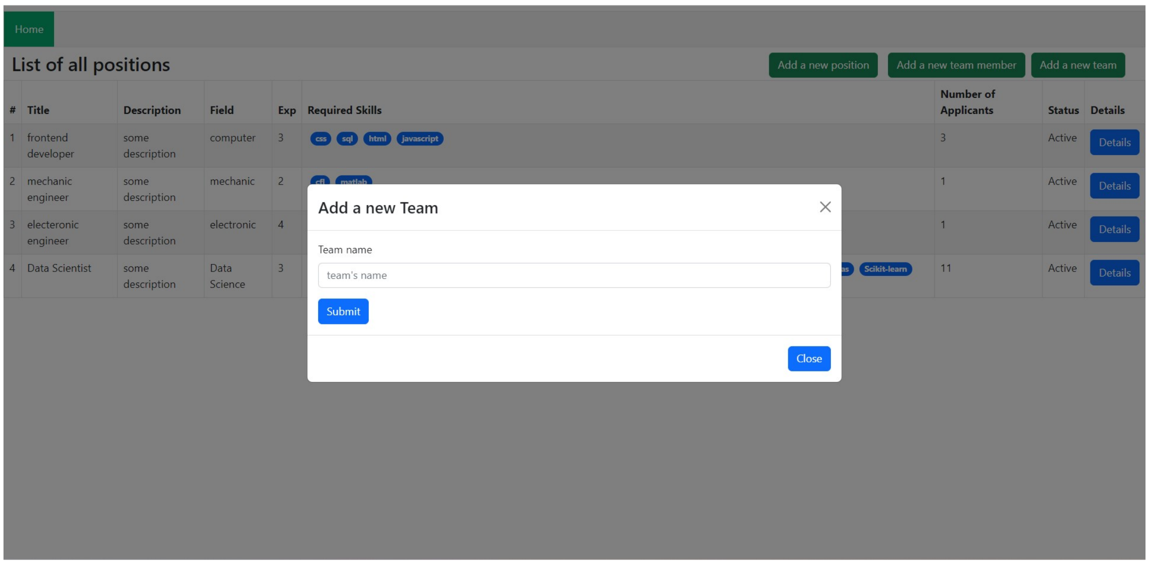
The second button is for adding a new team member. A new window will be popped out by clicking on this button, and the user can add a new team member to one of the existing teams by filling out the form and clicking on the "Submit" button.



#	Title	Description	Field	Exp	Number of Applicants	Status	Details
1	frontend developer	some description	computer	3	3	Active	Details
2	mechanic engineer	some description	mechanic	2	1	Active	Details
3	electeronic engineer	some description	electronic	4	1	Active	Details
4	Data Scientist	some description	Data Science	3	11	Active	Details

Figure 4.3: Adding a new team member

The last button is also for adding a new team to the system. By clicking on this button, the user can provide the name of the new team in a form that is popped out and add this new team.



List of all positions

#	Title	Description	Field	Exp	Required Skills	Number of Applicants	Status	Details
1	frontend developer	some description	computer	3	css sql html javascript	3	Active	Details
2	mechanic engineer	some description	mechanic	2	cfl matlab	1	Active	Details
3	electeronic engineer	some description	electronic	4		1	Active	Details
4	Data Scientist	some description	Data Science	3	as Scikit-learn	11	Active	Details

Add a new Team

Team name

team's name

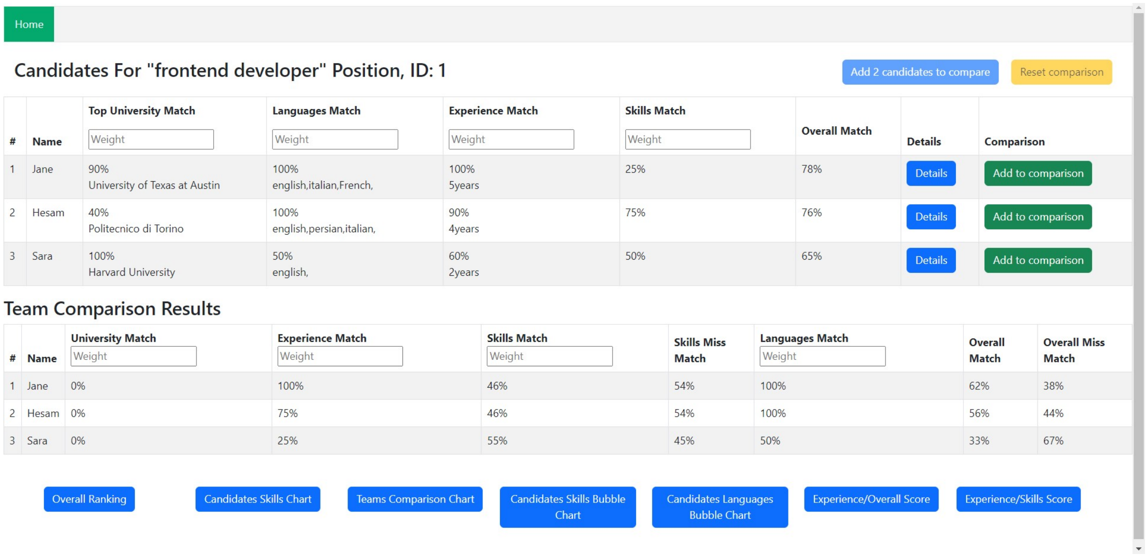
Submit

Close

Figure 4.4: Adding a new team

For saving the above information, three different APIs are called separately. This way, whenever the user creates a new element, the database will be updated immediately, and the data will be consistent throughout the system.

This page shows the position's information, which includes a list of people who have applied for this job and some charts and tables. The first section shows a table of candidates' scores in different fields.



Home

Candidates For "frontend developer" Position, ID: 1

Add 2 candidates to compare | Reset comparison

#	Name	Top University Match (Weight)	Languages Match (Weight)	Experience Match (Weight)	Skills Match (Weight)	Overall Match	Details	Comparison
1	Jane	90% University of Texas at Austin	100% english,italian,French,	100% 5years	25%	78%	Details	Add to comparison
2	Hesam	40% Politecnico di Torino	100% english,persian,italian,	90% 4years	75%	76%	Details	Add to comparison
3	Sara	100% Harvard University	50% english,	60% 2years	50%	65%	Details	Add to comparison

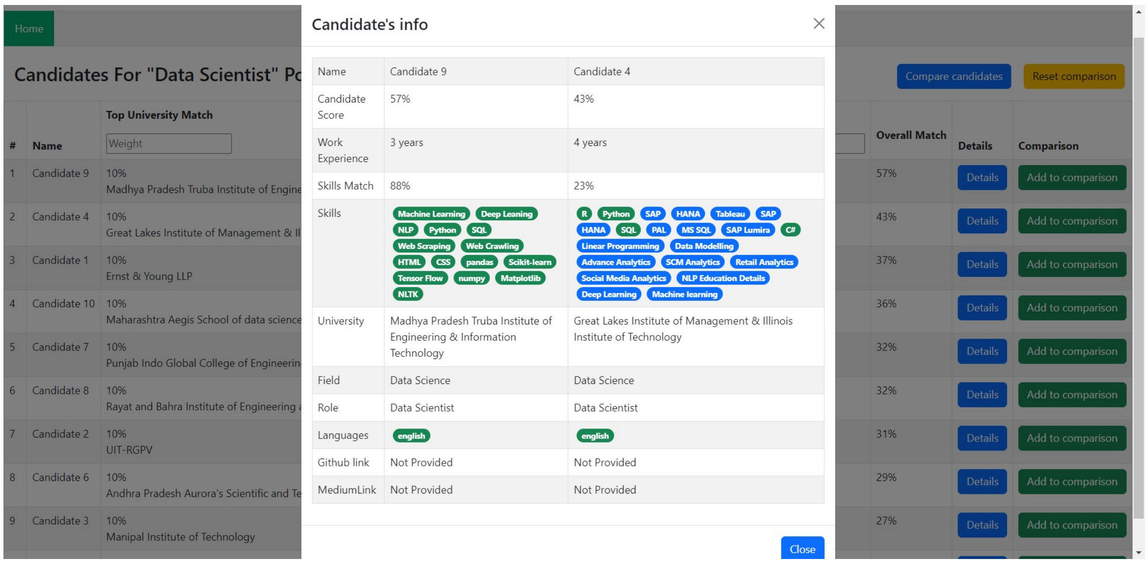
Team Comparison Results

#	Name	University Match (Weight)	Experience Match (Weight)	Skills Match (Weight)	Skills Miss Match	Languages Match (Weight)	Overall Match	Overall Miss Match
1	Jane	0%	100%	46%	54%	100%	62%	38%
2	Hesam	0%	75%	46%	54%	100%	56%	44%
3	Sara	0%	25%	55%	45%	50%	33%	67%

Overall Ranking | Candidates Skills Chart | Teams Comparison Chart | Candidates Skills Bubble Chart | Candidates Languages Bubble Chart | Experience/Overall Score | Experience/Skills Score

Figure 4.5: Position's information page

Also, there is a functionality in this section for comparing two different candidates with each other.



Home

Candidates For "Data Scientist" Pc

Compare candidates | Reset comparison

#	Name	Top University Match (Weight)	Overall Match	Details	Comparison
1	Candidate 9	10% Madhya Pradesh Truba Institute of Engine	57%	Details	Add to comparison
2	Candidate 4	10% Great Lakes Institute of Management & Il	43%	Details	Add to comparison
3	Candidate 1	10% Ernst & Young LLP	37%	Details	Add to comparison
4	Candidate 10	10% Maharashtra Aegis School of data science	36%	Details	Add to comparison
5	Candidate 7	10% Punjab Indo Global College of Engineerin	32%	Details	Add to comparison
6	Candidate 8	10% Rayat and Bahra Institute of Engineering	32%	Details	Add to comparison
7	Candidate 2	10% UIT-RGPV	31%	Details	Add to comparison
8	Candidate 6	10% Andhra Pradesh Aurora's Scientific and Te	29%	Details	Add to comparison
9	Candidate 3	10% Manipal Institute of Technology	27%	Details	Add to comparison

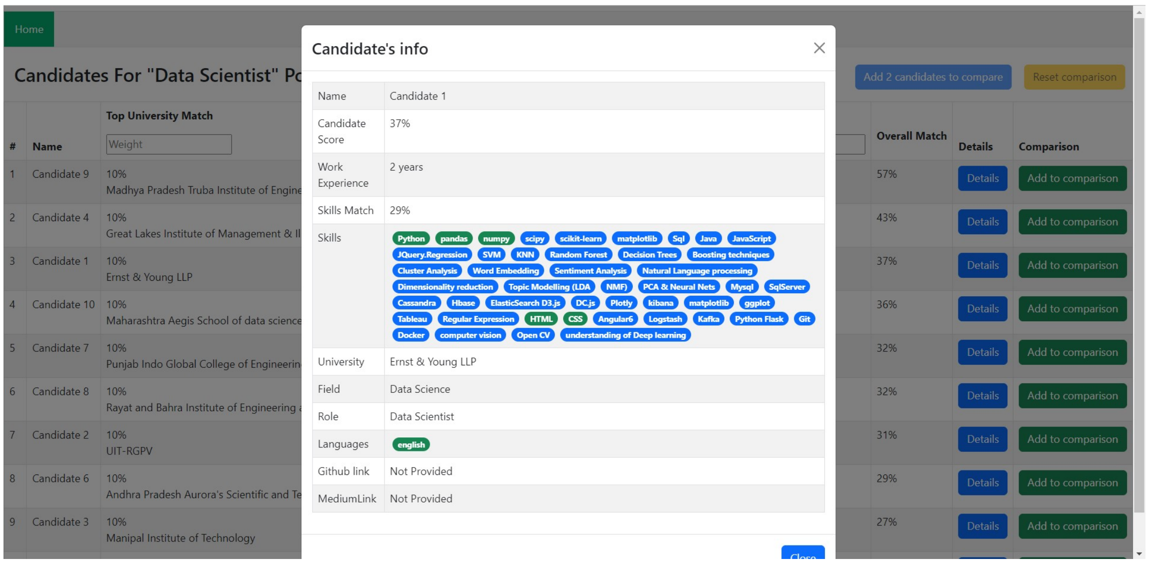
Candidate's info

Name	Candidate 9	Candidate 4
Candidate Score	57%	43%
Work Experience	3 years	4 years
Skills Match	88%	23%
Skills	Machine Learning, Deep Leaning, NLP, Python, SQL, Web Scraping, Web Crawling, HTML, CSS, pandas, Scikit-learn, Tensor Flow, numpy, Matplotlib, NLTK	R, Python, SAP, HANA, Tableau, SAP, HANA, SQL, PAL, MS SQL, SAP Lumira, C#, Linear Programming, Data Modelling, Advance Analytics, SCM Analytics, Retail Analytics, Social Media Analytics, NLP Education Details, Deep Learning, Machine learning
University	Madhya Pradesh Truba Institute of Engineering & Information Technology	Great Lakes Institute of Management & Illinois Institute of Technology
Field	Data Science	Data Science
Role	Data Scientist	Data Scientist
Languages	english	english
Github link	Not Provided	Not Provided
MediumLink	Not Provided	Not Provided

Close

Figure 4.6: Comparing candidates window

The "Details" button in front of each candidate opens a pop-up window. This window shows all the information related to the selected candidate. In the "Skills" and "Languages" fields, functionality is implemented, which makes the items that match the position's description in a green color to distinguish them from those that are not a match and provides better visualization for the user.



Candidates For "Data Scientist" Po

Add 2 candidates to compare | Reset comparison

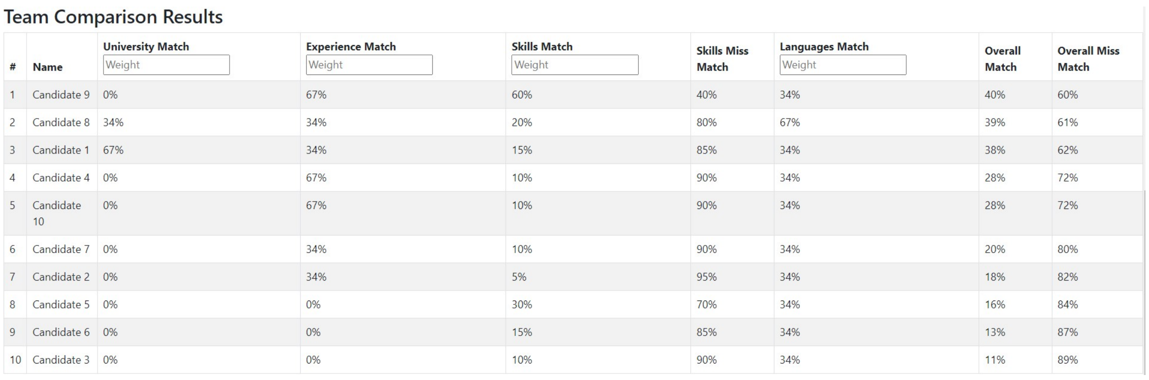
#	Name	Top University Match	Overall Match	Details	Comparison
1	Candidate 9	10% Madhya Pradesh Truba Institute of Engine	57%	Details	Add to comparison
2	Candidate 4	10% Great Lakes Institute of Management & II	43%	Details	Add to comparison
3	Candidate 1	10% Ernst & Young LLP	37%	Details	Add to comparison
4	Candidate 10	10% Maharashtra Aegis School of data science	36%	Details	Add to comparison
5	Candidate 7	10% Punjab Indo Global College of Engineerin	32%	Details	Add to comparison
6	Candidate 8	10% Rayat and Bahra Institute of Engineering	32%	Details	Add to comparison
7	Candidate 2	10% UIT-RGPV	31%	Details	Add to comparison
8	Candidate 6	10% Andhra Pradesh Aurora's Scientific and Te	29%	Details	Add to comparison
9	Candidate 3	10% Manipal Institute of Technology	27%	Details	Add to comparison

Candidate's info

Name	Candidate 1
Candidate Score	37%
Work Experience	2 years
Skills Match	29%
Skills	Python, pandas, numpy, scipy, scikit-learn, matplotlib, Sql, Java, JavaScript, JQuery.Regression, SVM, KNN, Random Forest, Decision Trees, Boosting techniques, Cluster Analysis, Word Embedding, Sentiment Analysis, Natural Language processing, Dimensionality reduction, Topic Modelling (LDA, NMF), PCA & Neural Nets, Mysql, SqlServer, Cassandra, Hbase, ElasticSearch D3.js, DC.js, Plotly, kibana, matplotlib, ggplot, Tableau, Regular Expression, HTML, CSS, Angular6, Logstash, Kafka, Python Flask, Git, Docker, computer vision, Open CV, understanding of Deep learning
University	Ernst & Young LLP
Field	Data Science
Role	Data Scientist
Languages	english
Github link	Not Provided
MediumLink	Not Provided

Figure 4.7: Candidate's details window

The second table shows the results of the comparison between each candidate and related team members of the company (which means people already working in the same team this position is related to).



Team Comparison Results

#	Name	University Match	Experience Match	Skills Match	Skills Miss Match	Languages Match	Overall Match	Overall Miss Match
		Weight	Weight	Weight		Weight		
1	Candidate 9	0%	67%	60%	40%	34%	40%	60%
2	Candidate 8	34%	34%	20%	80%	67%	39%	61%
3	Candidate 1	67%	34%	15%	85%	34%	38%	62%
4	Candidate 4	0%	67%	10%	90%	34%	28%	72%
5	Candidate 10	0%	67%	10%	90%	34%	28%	72%
6	Candidate 7	0%	34%	10%	90%	34%	20%	80%
7	Candidate 2	0%	34%	5%	95%	34%	18%	82%
8	Candidate 5	0%	0%	30%	70%	34%	16%	84%
9	Candidate 6	0%	0%	15%	85%	34%	13%	87%
10	Candidate 3	0%	0%	10%	90%	34%	11%	89%

Figure 4.8: Team comparison table

On the bottom of the page, there are different buttons for showing different charts and bars, which visualize data in a way that is more understandable and interesting for the user and helps the user make a better decision.

5 Evaluation

Six volunteers (1 student, five employed), each with a different educational background (1 with a BSc and 5 with an MSc), were requested to use the system for evaluation reasons.

Everyone was supposed to do a set of tasks defined the same for everyone. After briefly introducing the system and how different functionalities work, each person started to use the system and complete different tasks. To better compare the system's performance, some tasks had to be done both using the system and without the system. All volunteers were monitored while completing each task, and their time was written separately. The task chart shows the time to complete each task with and without the system. The difference between the time taken to complete a task with the system is noticeable.

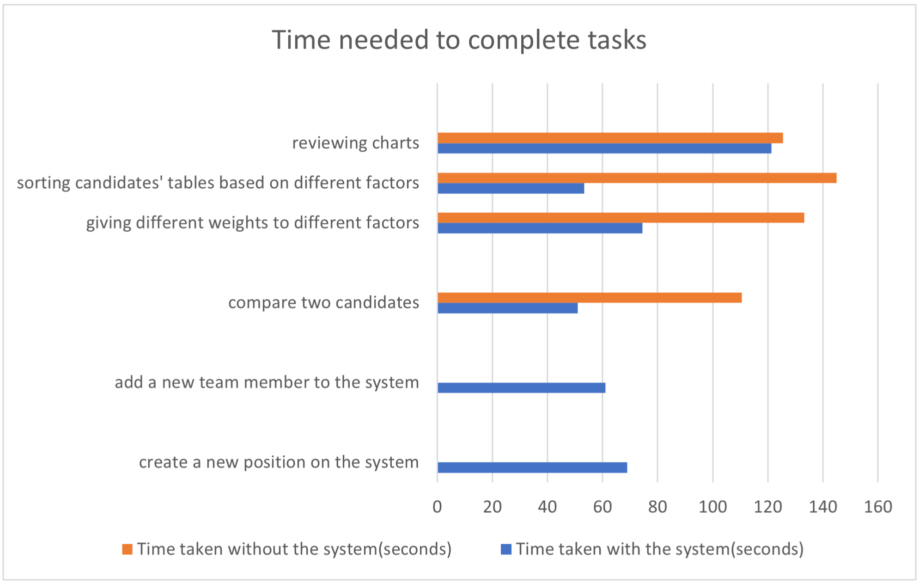


Figure 5.1: Time needed to complete tasks

After each person completed the defined tasks, a questionary was given to them. They were asked to give a score between 1 to 10 to different questions related to the system's functionality. the questionnaire chart, shows the average answers of all volunteers to the questionnaire, it is evident that working with the system has been easy for the users by considering the fact that was the first time they were using the system. Also, it is noticed that most volunteers prefer to use such a system for the hiring process.

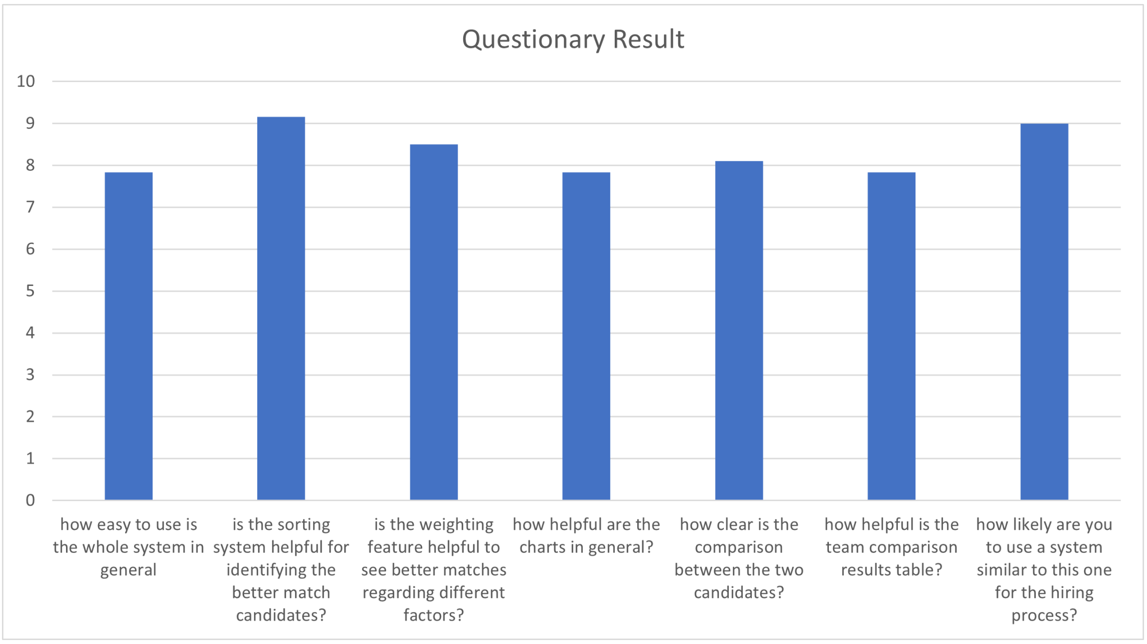


Figure 5.2: Results of questionnaire

6 Conclusion

This paper offers a system for comparing and rating job applicants based on two critical criteria. The first consideration is the job offer's description, which includes details on the needed skills, minimal experience, and, in some cases, the languages needed. The second aspect depends on how well each applicant complements the other team members already employed by the organization.

The suggested tool is based on a data set that contains information about various job candidate credentials, including educational background, skill sets, and years of experience. Recruiters in businesses are the system's primary target users. Using this system, many qualifications of applicants for a particular employment vacancy could be quickly analyzed by companies. Additionally, the end user may read a summary of each candidate's standing concerning other candidates and alter the importance of each relevant component in the comparison results based on the requirements of each job position.

On the one hand, the user is given helpful information about the possible candidate's suitability in the team by comparing each candidate to each team member inside the organization connected to the job post. Additionally, by utilizing various data visualization tools like bar charts, pie charts, bubble charts, and scatter plots, users can get a quick overview of the comparison's key characteristics of interest and get an overall view of all candidates.

The system could benefit from a more extensive data set in future works by considering more aspects of relevance between each job candidate and job positions. On the other hand, work will be done to create a team-building tool the companies may use to create an effective team for each position by considering various applicant characteristics.

References

[1] S. Mehta, R. Pimplikar, A. Singh, L. R. Varshney, and K. Visweswariah, "Efficient multifaceted screening of job applicants," in *Proceedings of the 16th International Conference on Extending Database Technology*, pp. 661–671, 2013.

[2] J. R. Hackman, *The psychology of self-management in organizations.* American Psychological Association, 1986.

[3] M. Sadiku, A. E. Shadare, S. M. Musa, C. M. Akujuobi, and R. Perry, "Data visualization," *International Journal of Engineering Research And Advanced Technology (IJERAT)*, vol. 2, no. 12, pp. 11–16, 2016.

[4] J. V. Sancho, J. C. Domínguez, and B. M. Ochoa, "An approach to the taxonomy of data visualisation," *Revista Latina de Comunicación Social*, no. 69, p. 486, 2014.

[5] S. Yuan, A. Tabard, and W. Mackay, "Streamliner: A general-purpose interactive course-visualization tool," in *2008 IEEE International Symposium on Knowledge Acquisition and Modeling Workshop*, pp. 915–919, IEEE, 2008.

[6] V. Gatteschi, F. Lamberti, and C. Demartini, "A graphical approach for comparing qualifications," in *2014 IEEE Global Engineering Education Conference (EDUCON)*, pp. 361–366, IEEE, 2014.

[7] C. Zhang and H. Wang, "Resumevis: A visual analytics system to discover semantic information in semi-structured resume data," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 1, pp. 1–25, 2018.

[8] W. S. Cleveland and R. McGill, "Graphical perception: Theory, experimentation, and application to the development of graphical methods," *Journal of the American statistical association*, vol. 79, no. 387, pp. 531–554, 1984.

[9] V. Gatteschi, F. Lamberti, G. Paravati, A. Raso, and C. Demartini, "Which learning outcomes should i acquire? a bar chart-based semantic system for visually comparing learners' acquirements with labor market requirements," in *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, pp. 774–779, IEEE, 2016.

[10] V. Gatteschi, F. Lamberti, A. Sanna, and C. Demartini, "Using tag clouds to support the comparison of qualifications, résumés and job profiles," in *2011 9th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, pp. 57–61, IEEE, 2011.

[11] F. Beck, M. Burch, and S. Diehl, "Matching application requirements with dynamic graph visualization profiles," in *2013 17th international conference on information visualisation*, pp. 11–18, IEEE, 2013.

[12] J. Mahapatra, A. Metrewar, A. Tripathi, and P. Dutta, "Selection central: Candidate profile enhancement and feedback based ranking," 10 2014.

[13] A. Singh, C. Rose, K. Visweswariah, V. Chenthamarakshan, and N. Kambhatla, "Prospect: a system for screening candidates for recruitment," in *Proceedings of the 19th ACM international conference on Information and knowledge management*, pp. 659–668, 2010.