Talent Search

Streamlining Recruitment with AI-Powered Candidate Matching

Project Description

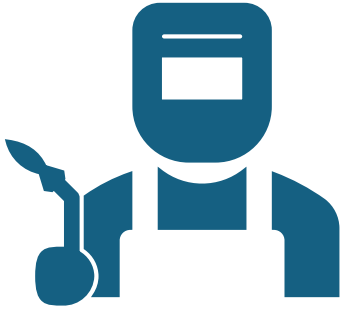
Talent Search is a Java-based application powered by OpenAI. It assists hiring managers in:

Updating job descriptions

Searching for candidates based on specific skills

Matching resumes with job requirements using AI

Talent Search Description



TalentSearch is a Java-based application powered by OpenAI, designed to assist hiring managers in identifying the most suitable candidates for job openings. The application allows hiring managers to log in, update job descriptions, and search for candidates based on specific skills and experience. By leveraging OpenAI's capabilities, **TalentSearch** analyzes job descriptions and matches them with the most qualified candidates, streamlining the recruitment process and ensuring better hiring decisions.

Instead of relying on manual searches and interviews, **TalentSearch** provides an AI-driven approach to talent acquisition, saving time and resources while improving the overall quality of hires. Unlike traditional systems that map keywords to skills, **TalentSearch** uses OpenAI's natural language processing capabilities to understand the nuances of job descriptions and identify the right candidates. This advanced approach enables hiring managers to focus on what matters most which is finding the best talent for their organization.

We are from Group1

- Jayanth Mani (002373810)
- Adharsh Rengarajan (002085750)
- Burra Sai Kalyan (002301631)
- Zuoyu Wang (002342819)
- Xinduo Fan (002059620)

Problem Statement

TRADITIONAL RECRUITMENT METHODS ARE:

TIME-CONSUMING

PRONE TO ERRORS

INEFFICIENT IN MATCHING CANDIDATES TO JOB DESCRIPTIONS

Technology Stack

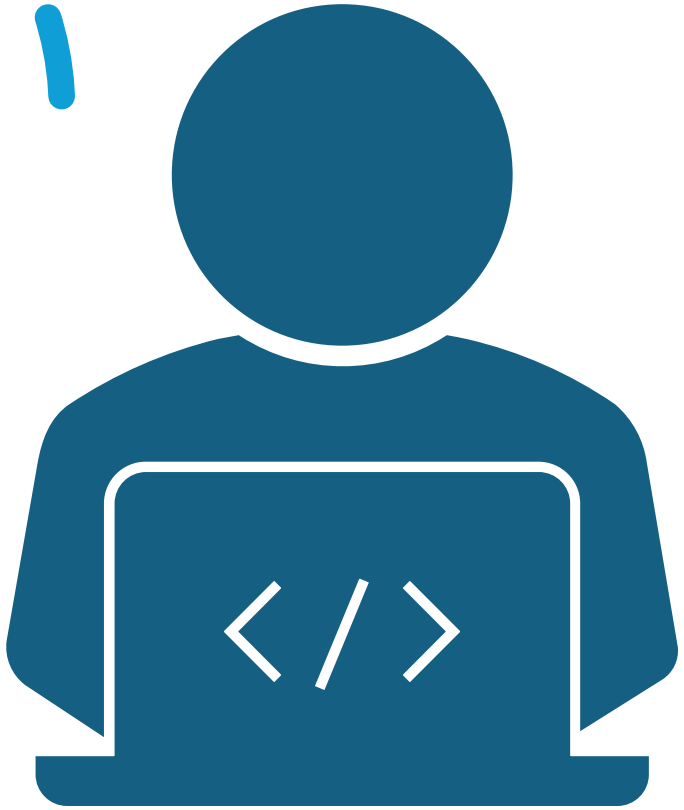
- Front-End: React.js
- Back-End:
- Java: Core language for business logic

  -->Spring Boot: REST-based scalable architecture

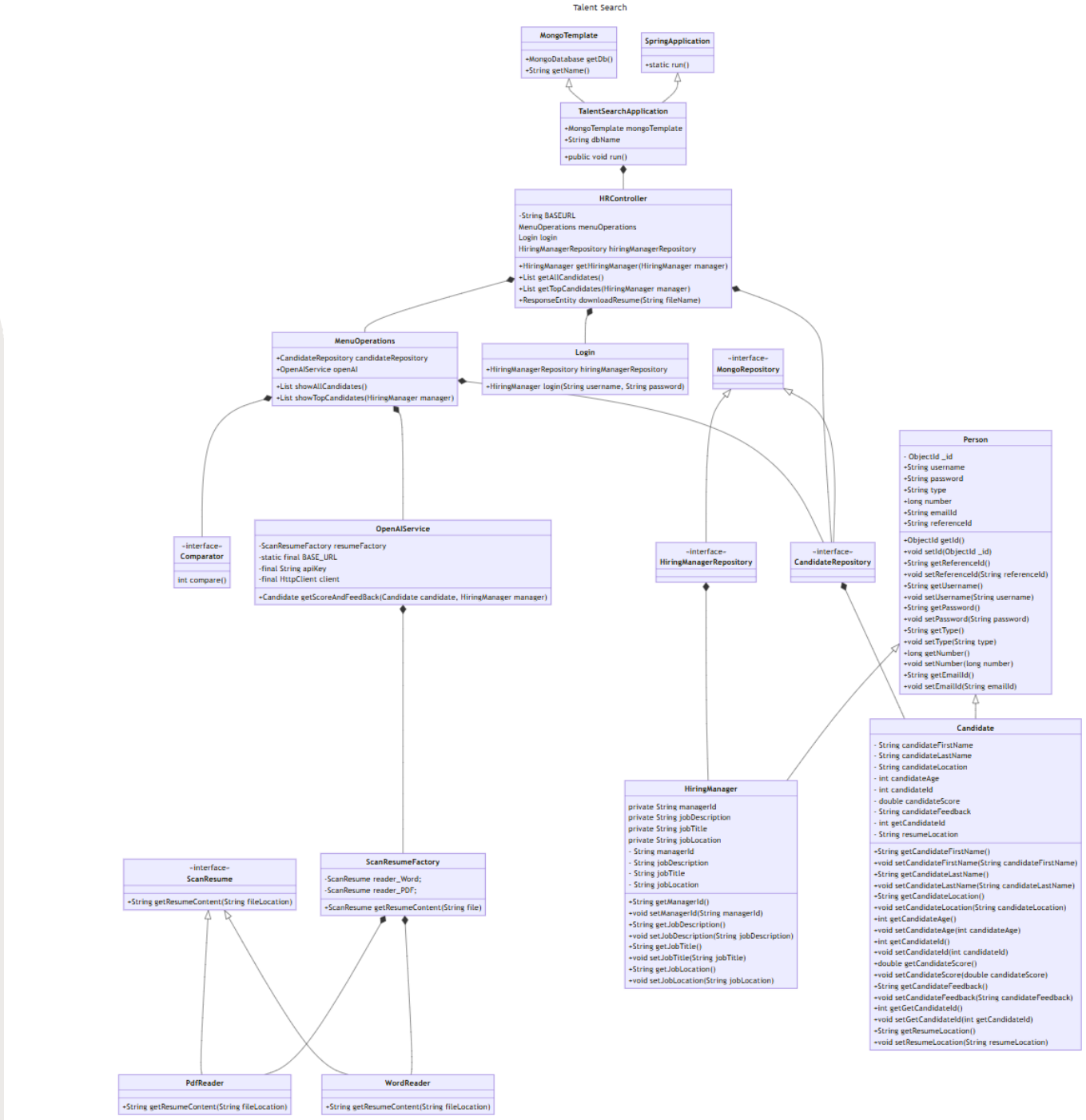
  -->OpenAI API Integration: NLP for resume analysis
- Database: MongoDB

OOD Concept & Design Pattern

- Encapsulation: Secure data handling through classes like User.
- Inheritance: HiringManager and Candidate inherit from Person.
- Polymorphism: Method overriding for database operations.
- Abstraction: Abstract classes and interfaces streamline features.
- Composition: Classes use other classes for tasks like scanning resumes.
- Classes & Objects: Represent entities like HiringManager and Candidate.
- Association: Inter-class relationships for functionality
- Factory Model: Used a factory model to generate objects for scanning resumes in .docx and .pdf formats
- Exception Handling: Used Exception Handling techniques to handle run – time errors
- Generics: Used Generics to create repository using MongoDB Repository



System Architecture Diagram



Milestone 1: The Foundation - Documentation and Planning

- Our journey began with laying a solid foundation for the **project. Milestone 1 was all about documentation.**
- **This was the crucial phase where we defined the project's goals, objectives, and the roadmap ahead. It allowed us to establish a shared vision and ensured that all team members were aligned on the core principles, technologies, and functionalities.**
- By the end of this milestone, we had a comprehensive plan in **place that guided us through the development process, helping to clarify each component of the system and their role in the overall architecture.**

Milestone 2: Building the Core System

With the groundwork set, we moved into **Milestone 2**, which focused on building the essential components that formed the core of our system. This was a critical phase where we brought the project to life by implementing various features.

- **Login and Authentication:**
  - We kicked off the development by creating the login system, using the Login class. The authentication allowed hiring managers to securely log in, a key functionality for ensuring that the right people had access to the platform.
- **Database Integration:**
  - Next, we focused on establishing database connectivity. This allowed the system to interact with the database, fetching candidates, updating job descriptions, and retrieving necessary data. We built the foundation for storing and retrieving information efficiently.
- **Candidate Management:**
  - We created the Candidate class to manage and store profiles, which was a major milestone in ensuring that the system could handle candidate data (such as scores and feedback) efficiently.
- **Job Description Management:**
  - In parallel, we implemented the MenuOperations class to handle job descriptions, enabling hiring managers to easily update and retrieve job postings. This was an important step toward ensuring that job details could be managed effortlessly.
- **Resume Scanning and Scoring:**
  - The heart of the recruitment process was implemented next—resume scanning. We integrated the ScanResume class and leveraged OpenAI's powerful API to score resumes and generate feedback, which helped to evaluate candidates effectively based on their resumes.
- **Spring Boot Program:**
  - We developed the TalentSearchApplication class as the central entry point for running the application. This was crucial to initialize and orchestrate the system's components.
- **Display Features:**
  - Finally, we implemented the menu operation features to present candidate information in a user-friendly format, making the system easy to navigate and interact with.

- **Milestone 3: Final Integration and Team Contributions**
- As we entered **Milestone 3**, it was time to bring everything together. This phase was all about **integration**, ensuring that all components worked seamlessly and collaborated to deliver the final product. Each team member played a crucial role in this phase, contributing their expertise to make the project functional and efficient.
- **Jayanth Mani (Frontend-Backend Integration Specialist):**
  - Jayanth took charge of integrating the frontend and backend systems, ensuring that the user interface could interact with the backend seamlessly. He also focused on validating frontend inputs and conducted extensive testing to make sure the two layers of the application communicated in real-time.
- **Adharsh Rengarajan and Xindo Fan (AI Integration & Testing Team):**
  - Adharsh and Xindo worked on enhancing the integration with OpenAI's API, handling edge cases, and refining resume analysis algorithms. They also focused on testing the feedback generation system to ensure that the AI provided accurate results. Their work built upon earlier integrations, improving the overall accuracy of candidate evaluation.
- **Zuoyu Wang and Sai Kalyan (Database & Security Team):**
  - Zuoyu and Sai were focused on securing the database connections and ensuring that all sensitive data (like login credentials) were safely handled. They also added runtime checks to validate API responses and conducted thorough testing of the authentication system to ensure security and reliability.
- **Shared Responsibilities:**
  - As the project came together, all team members participated in system testing, ensuring that every feature was working as expected. We collaborated closely during the integration phases, conducting cross-functional code reviews and ensuring that the project was documented comprehensively.

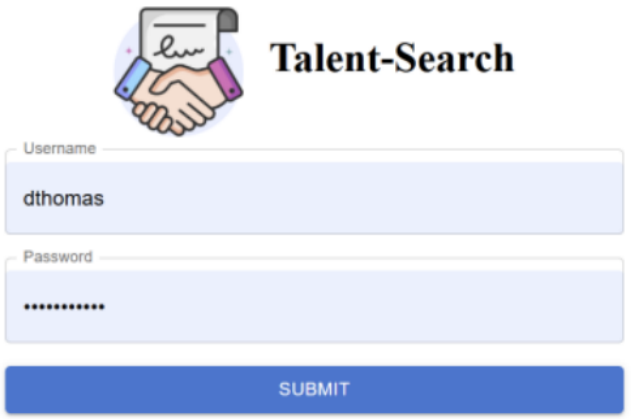
Future Scope of Talent Search Project

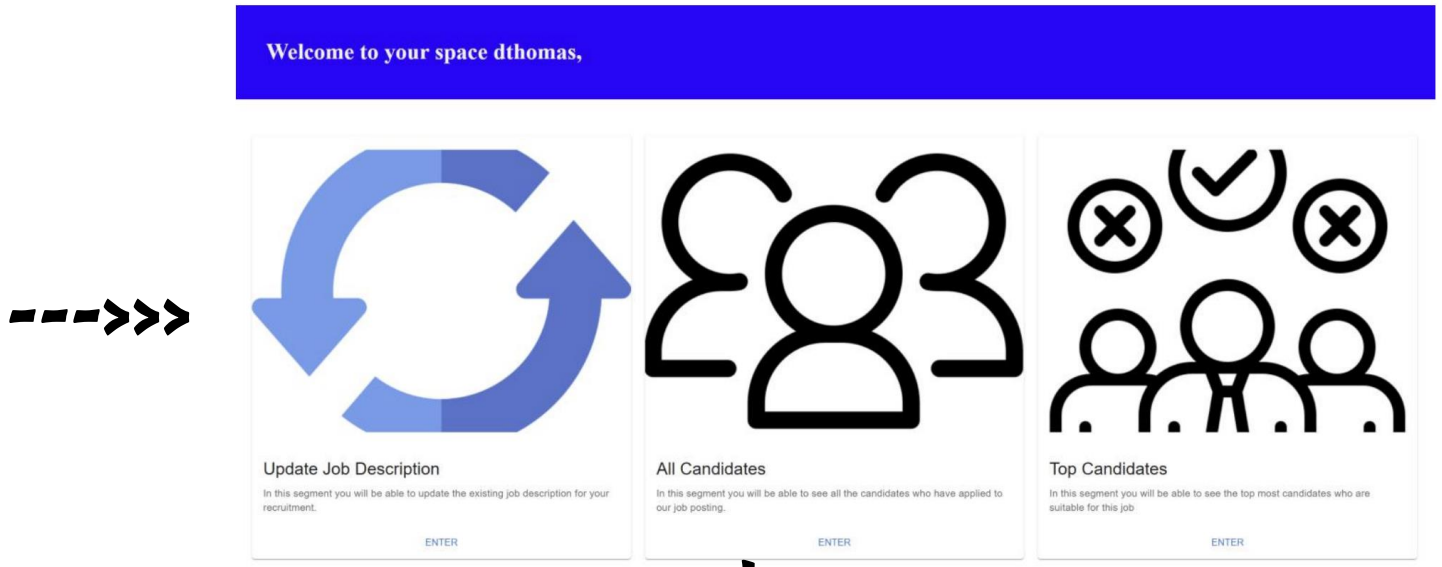
1. **Global Talent Search:**
   - Expand the platform to support multiple languages for global reach.
   - Incorporate region-specific hiring norms and practices.
2. **Candidate Analytics:**
   - Introduce analytics dashboards for hiring managers to visualize candidate trends, scores, and feedback.
   - Insights into hiring success rates and market demand for specific skills.
3. **Mobile App Development:**
   - Launch a mobile application for on-the-go access to job descriptions, candidate profiles, and recruitment updates.

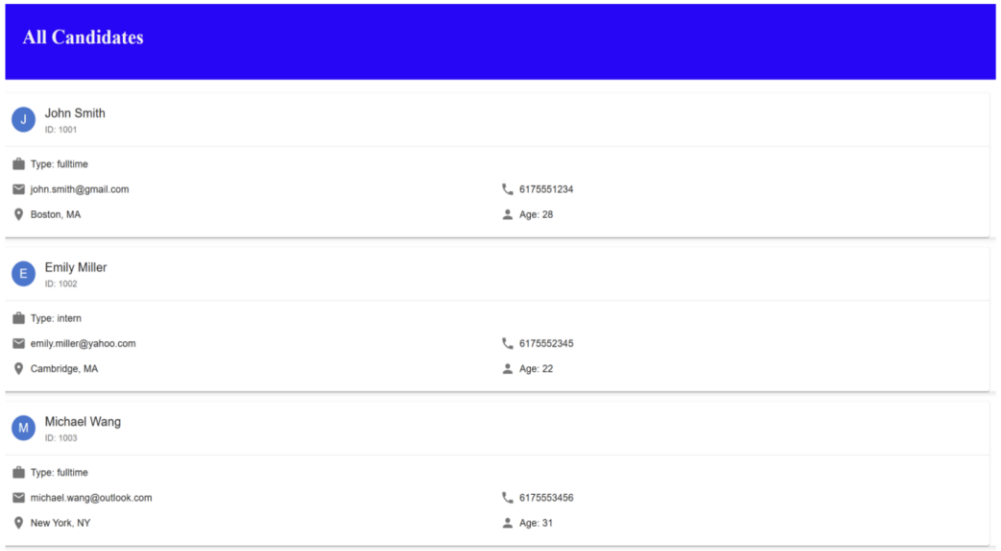
Takeaways

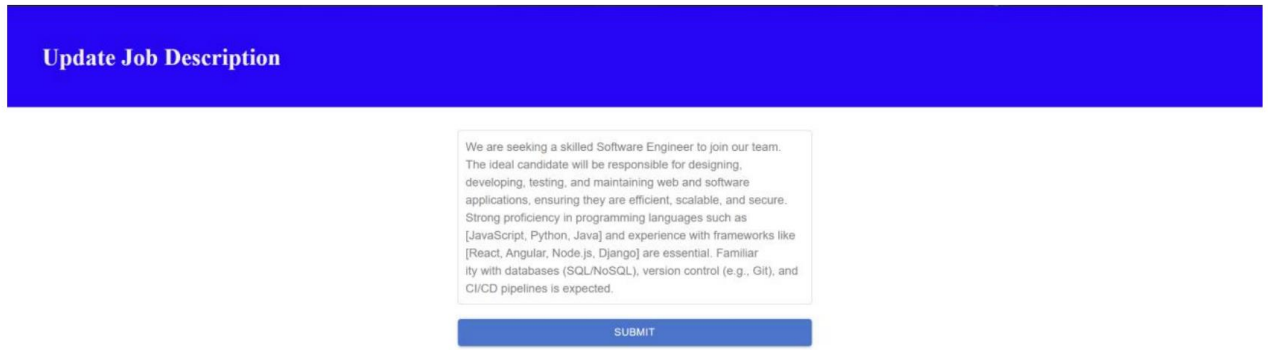
- **Encapsulation**: Bundles data and methods within a class, restricting access to sensitive data using getter and setter methods (e.g., for passwords).
- **Inheritance**: Allows a subclass to inherit properties and behaviors from a superclass, promoting code reuse and reducing redundancy.
- **Polymorphism**: Enables methods to perform different tasks based on the calling object, with method overriding and interfaces offering different implementations.
- **Abstraction**: Hides implementation details and presents only essential features, typically using abstract classes and interfaces.
- **Composition**: Combines objects of different classes to achieve complex functionality (e.g., Home class uses HiringManager and Candidate objects).
- **Classes and Objects**: Classes define the structure and behavior of objects, which interact to form a modular system.
- **Association**: Defines relationships between classes, allowing them to work together.
- **SOLID pattern**: Principles ensuring scalable, maintainable, and robust object-oriented software design.
- **Factory Model**: Design pattern creating objects without specifying their concreted classes explicitly

Demo Front-end









Demo Back-end from console and Database

- **Manager Loggin In (Console)**

```
Entered[HiringManager [managerId=HM001, jobDescription=Looking for a Software Engineer with strong expertise in Java, Spring Boot, and microservices. Experience with Docke
Logged in manager ID: 65730d8b70e68c2d45a1b1a1
```

- **Scrapping Resume contents (Console Output)**

```
Name: John Smith
Phone: (123) 456-7890
Email: john.smith@example.com
LinkedIn: linkedin.com/in/jo
Emily
Full Stack Developer (MERN Stack)
Sarah Johnson sarah.j.dev@email.com | (206) 555-7890 | Seattle, WA
Working
Michael
Name: James Williams
Phone: (456) 789-0123
Email: james.williams@example.com
LinkedIn: linkedin.c
```

Before jobDescription Updation (Database)


```
_id: ObjectId('65730d8b70e68c2d45a1b1a1')
username : "sarah.mitchell"
password : "securePass123!"
type : "HIRING_MANAGER"
number : 9876543210
emailId : "sarah.mitchell@techcorp.com"
managerId : "HM001"
jobDescription : "Looking for a Software Engineer with strong expertise in Java, Spring …"
jobTitle : "Senior Software Engineer - Backend"
jobLocation : "San Francisco, CA"
```


After Job Descriptipn updation (Database)


```
_id: ObjectId('65730d8b70e68c2d45a1b1a1')
managerId : "HM001"
jobDescription : "Updating Sarah's Job Description From React"
jobTitle : "Senior Software Engineer - Backend"
jobLocation : "San Francisco, CA"
username : "sarah.mitchell"
password : "securePass123!"
type : "HIRING_MANAGER"
number : 9876543210
emailId : "sarah.mitchell@techcorp.com"
_class : "edu.neu.csye6200.models.HiringManager"
```