



The Egyptian E-Learning University (EELU)

Faculty of Information Technology

Graduation Project

2023-2024

Job Fit Analyzer

Supervisors

Prof. Dr. Khaled T. Wassif

Eng. Engy Emad Beshara Salib

Team Members

Mohamed Khaled Mohamed Moghraby

Ahmed Mohamed Tawab

Makarious Ayman Shafek

Mohamed Montser Mohamed

Sandy Nashaat Wadea

Mohammad Atef Nasr Labib

Ziad Mamdouh Ahmed

DECLARATION

We hereby certify that this report, which I now submit for assessment on the program of study leading to the award of Bachelor of Science in Information Technology Bachelor, is all my own work and contains no plagiarism. By submitting this report, I agree to the following terms:

Any text, diagrams or other material copied from other sources (including, but not limited to, books, journals, and the internet) have been clearly acknowledged and cited followed by the reference number used; either in the text or in a footnote/endnote. The details of the used references that are listed at the end of the report are confirming to the referencing style dictated by the final year project template and are, to my knowledge, accurate and complete.

I have read the sections on referencing and plagiarism in the final year project template. I understand that plagiarism can lead to a reduced or fail grade, in serious cases, for the Graduation Project course.

Student Name: Mohamed Khaled Moghraby

ID: 2000912

Signed:

Date: 24/6/2024

Student Name: Sandy Nashaat Wadea

ID: 2001834

Signed:

Date: 24/6/2024

Student Name: Mohamed Montser Mohamed

ID: 2001861

Signed:

Date: 24/6/2024

Student Name: Mohammad Atef Nasr Labib

ID: 2000643

Signed:

Date: 24/6/2024

Student Name: Ahmed Mohamed Tawab

ID: 2001750

Signed:

Date: 24/6/2024

Student Name: Makarious Ayman Shafek

ID: 2001911

Signed:

Date: 24/6/2024

Student Name: Ziad Mamdouh Ahmed

ID: 2000251

Signed:

Date: 24/6/2024

ACKNOWLEDGMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

We highly indebted to (Doctor Khaled Wasif) for the guidance and constant supervision as well as for providing necessary information regarding the project & also for the support in completing the project.

Also We indebted to (Eng. Engy Emad) for the guidance and constant supervision as well as for providing necessary information regarding the project & also for the support in completing the project.

I would like to express my gratitude to wards my parents & member of (The Egyptian E-Learning University) for the kind co-operation and encouragement which help us in completion of this project.

I would like to express my special gratitude and thanks to supervisor`s persons for giving me such attention and time.

My thanks and appreciations also go to my colleagues in developing the project and people who have willingly help me out with their abilities.

Table of Contents

Table of Figures	10
Abstract	11
Objective.....	12
System Requirements	14
Functional Requirements	14
Non-Functional Requirements	16
Similar Systems.....	17
Local System	17
International System	18
System Architecture	19
System Overview	19
Architecture diagram	20
Use Case Diagram	21
Sequence Diagram.....	22
Database Design	23
Server-side Architecture	26
Main Entities in Prisma.....	27
Client-side Architecture	30
AJAX	30
Fetching data from the server	30
React	31
Benefits of choosing React Technology	32
Next.js	32
Benefits of choosing Next.js Technology:.....	33
Material UI (MUI)	33
API Integration in React	33
Authentication.....	34
NextJs Authentication	36
Technology Stack.....	37
AI Integration.....	39

Key Highlights	39
Machine Learning Algorithms	40
Supervised Learning	40
Unsupervised Learning.....	42
Reinforcement Learning.....	43
Types of Recommendations systems	44
Content-Based Recommendations	44
Collaborative Filtering.....	45
Hybrid Recommendations	46
Dataset.....	48
AI Endpoint	56
Flask Overview	56
UI/UX Design	58
UI Design.....	58
UX Design.....	58
Preview	59
Conclusion	66
Future Work.....	67
Implementation.....	67
References	68
Appendix.....	69

Table of Figures

Figure 1 Wuzzuf.com	161
Figure 2 Indeed.com	17
Figure 3 System Design	14
Figure 4 Usecase Diagram	15
Figure 5 Sequence Diagram	16
Figure 6 Database Schema Diagram	19
Figure 7 Server-side Rendering	20
Figure 8 User Model	21
Figure 9 Account Model	21
Figure 10 Job Model	22
Figure 11 Application & Company Models	22
Figure 12 Profile	23
Figure 13 Experience & Education	23
Figure 14 Data fetching	24
Figure 15 React Diagram (1)	25
Figure 16 React Diagram (2)	26
Figure 17 JWT Auth	29
Figure 18 Prisma Rest	32
Figure 19 Second Dataset	44
Figure 20 Accuracy Score Third Dataset	45
Figure 21 Third Dataset	45
Figure 22 Current Dataset	46
Figure 23 Accuracy Score Current Dataset	48
Figure 24 Updated Accuracy Scores Current Dataset	48
Figure 25 Output Current Dataset	49
Figure 26 Home page	53
Figure 27 Sign up page	54
Figure 28 Log in Page	54
Figure 29 Profile page	55
Figure 30 Jobs Page	57
Figure 31 Admin Dashboard	58
Figure 32 Recruiter Dashboard	59
Figure 33 Application page	59

Abstract

This project addresses the current complexity and inefficiency of job matching in today's labour market, leading to job dissatisfaction, high turnover, and resource-intensive hiring processes for both candidates and employers.

Our solution is the development of a Job Fit Analyzer, powered by AI models and advanced software. This innovative solution aims to streamline and optimize the job matching process by utilizing state-of-the-art artificial intelligence techniques.

The Job Fit Analyzer is designed as a web application, functioning as a comprehensive job portal.

This web-based platform will not only enhance the efficiency of job matching but also provide a user-friendly experience for both job seekers and employers, revolutionizing the way individuals find suitable employment opportunities and organizations identify qualified candidates.

Objective

The overarching objective of this project is to conceptualize, design, and implement an advanced job portal system that transcends traditional job search platforms. Our aim is to create a dynamic ecosystem where job seekers and recruiters seamlessly converge, fostering a symbiotic relationship.

For job seekers, the project endeavors to deliver a user-friendly interface empowering them to effortlessly explore, apply for, and manage job opportunities. The system's sophistication lies in its ability to provide personalized job recommendations through an innovative AI-powered engine, ensuring that job seekers encounter opportunities tailored to their skills and preferences.

Recruiters, on the other hand, will benefit from a robust set of tools that enable them to efficiently post, manage, and evaluate job listings. The system seeks to streamline the recruitment process, offering recruiters an intuitive platform to engage with potential candidates effectively.

Throughout the project lifecycle, a key focus will be on implementing cutting-edge technologies and methodologies. This includes the incorporation of a notification system to keep users informed of relevant updates and events. Additionally, we aspire to enhance caching mechanisms using Redis, optimizing the system's performance and responsiveness.

By the conclusion of this project, our vision is to deliver a comprehensive job portal solution characterized by its security, scalability, and user-centric design. We anticipate that the implemented features and innovations will not only meet but exceed the expectations of both job seekers and recruiters, elevating the overall experience within the realm of job searching and hiring.

System Requirements

Functional Requirements

User Types

Admin:

- Full CRUD operations on Applications, Users, Jobs, etc.

Job Seeker:

- Apply for a job.
- View/Update Profile.
- Create and edit profiles with relevant information.
- View Applications.
- Upload CV/Resume.
- Verify Email and Phone number.

Recruiter:

- Post a Job.
- Manage job status and details.
- View Applicants for the job.
- Download applicant's CV.
- Accept or reject an application.

Common Login Functionality for All Users

1. Validation during login.
2. Display appropriate error messages.

Job Search and Filters

Implement job search functionality with filters for:

- Location.
- Industry.
- Job type.
- And more.

AI-Powered Job Recommendation and Search

- The AI system should:
 - Analyze user preferences.
 - Evaluate job listings.
 - Provide personalized job recommendations.

Feedback and Ratings

- Allow both Recruiters and Job Seekers to:
 - Provide feedback.
 - Rate the recruitment process.

Non-Functional Requirements

Performance:

Response Time:

Ensure quick response times for actions like job searches and profile updates.

Scalability:

Make sure the system can handle more users as needed.

Security:

Authentication:

Keep user logins secure using standard protocols.

Authorization:

Allow users access based on their roles.

Data Encryption:

Protect sensitive information with encryption.

Usability:

User Interface Consistency:

Maintain a consistent and user-friendly interface.

Accessibility:

Ensure the system is accessible to users with disabilities.

Integration:

Third-Party Integration:

Seamlessly connect with external services.

Compatibility:

Ensure compatibility with popular browsers and operating systems.

Similar Systems

Local System

Wuzzuf.com (An Egyptian Job Portal Website)

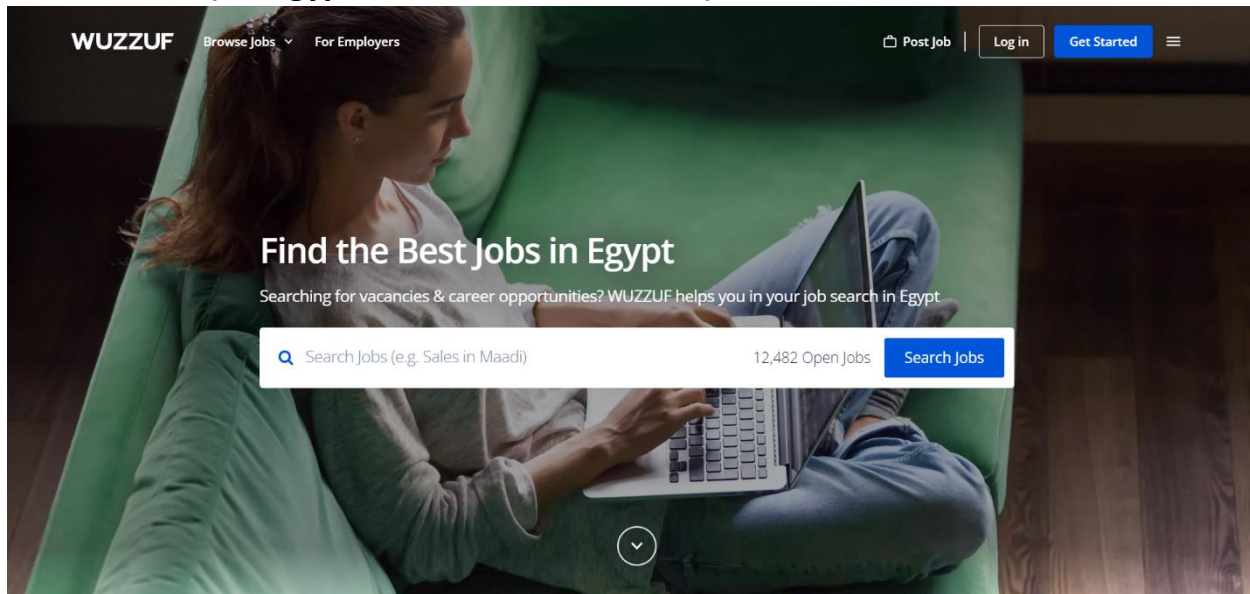


Figure 1 Wuzzuf

Advantages:

- User friendly UI
- Comprehensive Job Listings
- Employer Tools
- Local Insights

● Disadvantages:

- Too reliant on skills
- Security and Data Protection
- Language Barrier – *The website is supposed to be in Arabic, but it displays in English.*
- Machine Learning Integration

International System

Indeed.com

The screenshot shows the Indeed.com homepage. At the top, there is a navigation bar with the Indeed logo, links for Home and Company reviews, and icons for chat, notifications, and user profile. Below the navigation bar is a search bar with two input fields: "Job title, keywords, or company" and "City, state, zip code, or 'remote'", followed by a "Find jobs" button. Below the search bar are two tabs: "Job feed" (selected) and "Recent searches". In the center of the page, there is a message: "We're working on your personalized job feed. In the meantime, run a search to find your next job." with a "Find jobs" button. At the bottom, there is a footer with links for Browse Jobs, Browse Companies, Countries, About, Help Center, ESG at Indeed, © 2024 Indeed, Accessibility at Indeed, Privacy Center and Ad Choices, and Terms.

Figure 2 Indeed

Advantages:

- Comprehensive Job Listings
- User-Friendly Interface
- Powerful Search Functionality
- Resume Upload
- Employer Tools
- Free to use
- **Disadvantages:**
 - Spam and Scams
 - Limited Customization for Applications
 - Search Algorithm Limitations
 - Employer Bias

System Architecture

System Overview

Three-Tier Architecture:

Our system architecture is organized into three distinct tiers, each serving specific functions:

- **Presentation Tier:**
 - This tier is responsible for the user interface and interaction. It encompasses the web-based front end where users interact with the job portal's features and functionalities.
- **Application Tier:**
 - The application tier houses the core logic and functionality of the job portal. It manages processes such as job recommendations, user authentication, AI integration, and communication between the front end and the back end.
- **Data Tier:**
 - The data tier handles the storage and retrieval of information. It includes the database system that stores user profiles, job listings, and related data, ensuring data integrity and accessibility.

Architecture diagram

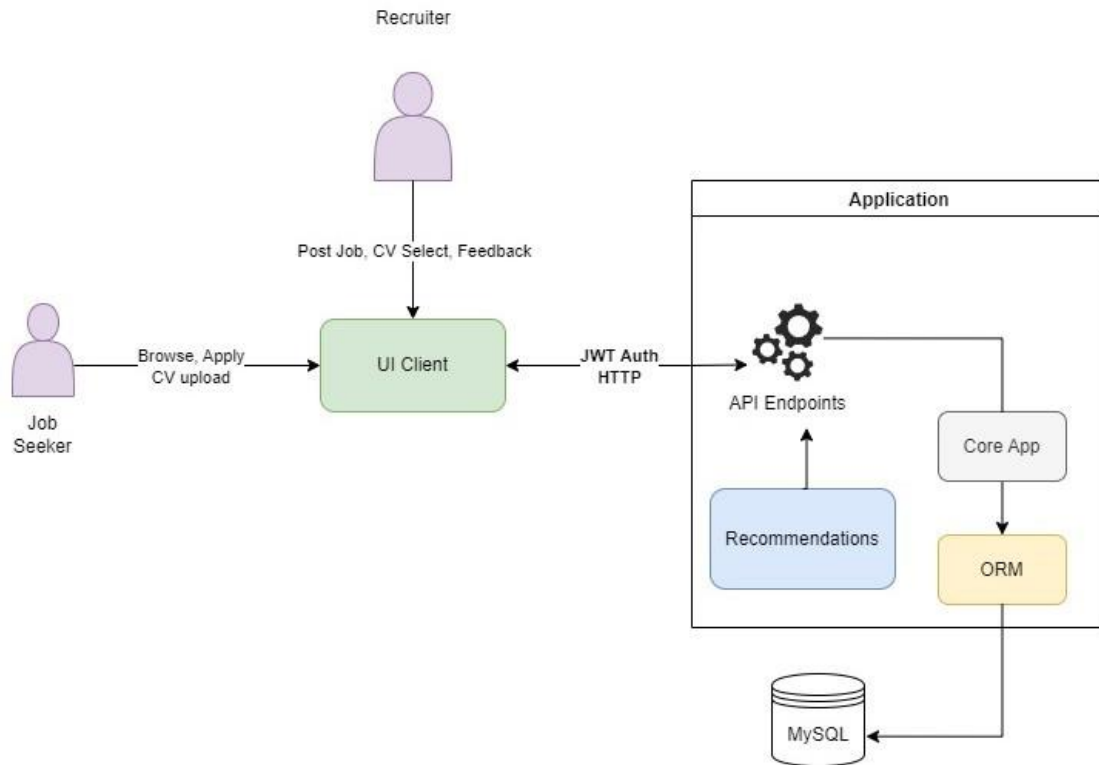


Figure 3 System Design

The diagram explains how the system works together. Job seekers and recruiters use the UI Client to interact with API endpoints. These endpoints then use the core app to deal with the recommendation engine and the ORM to handle interactions with the database.

Use Case Diagram

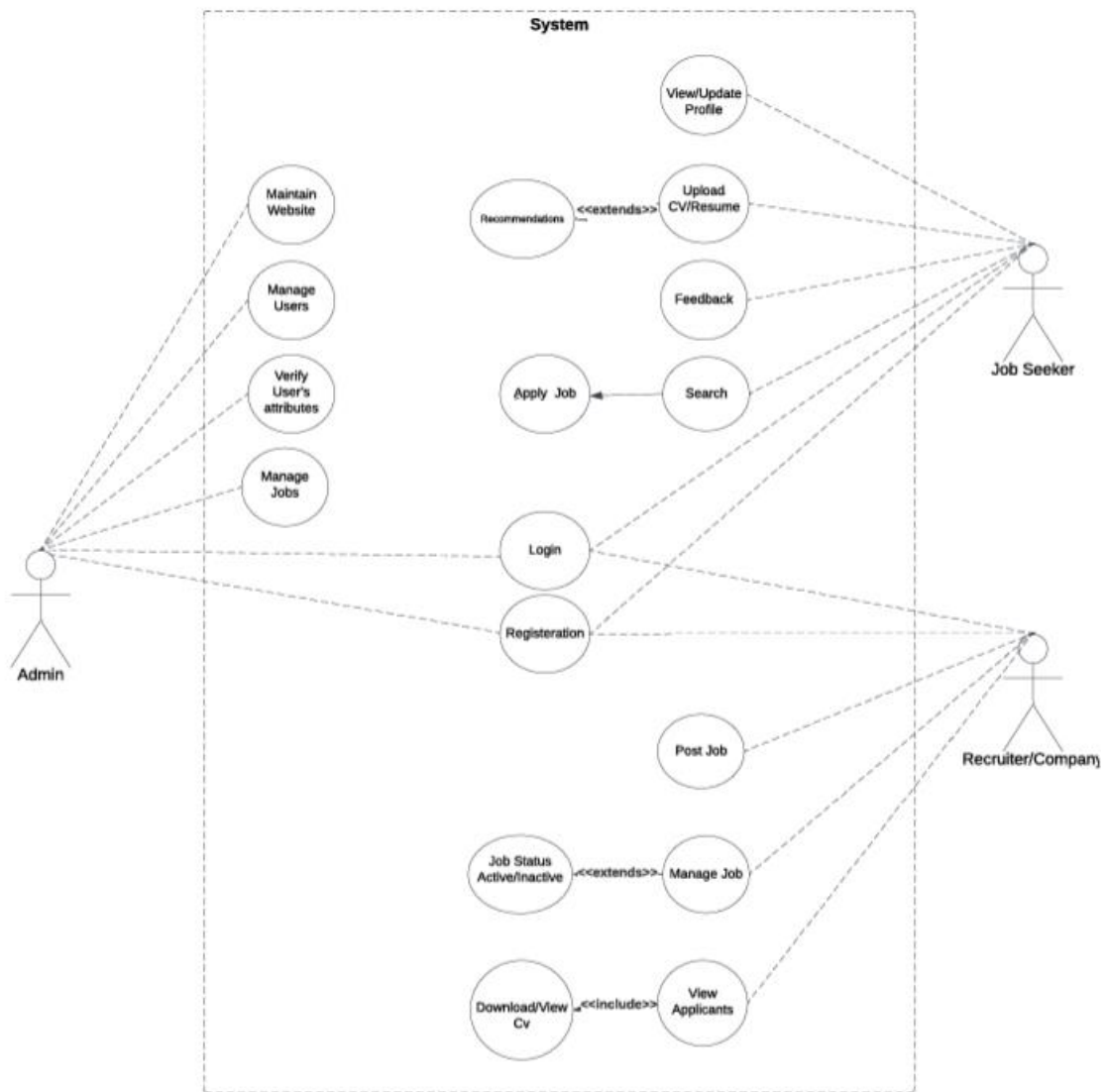


Figure 4 Use Case Diagram

Sequence Diagram

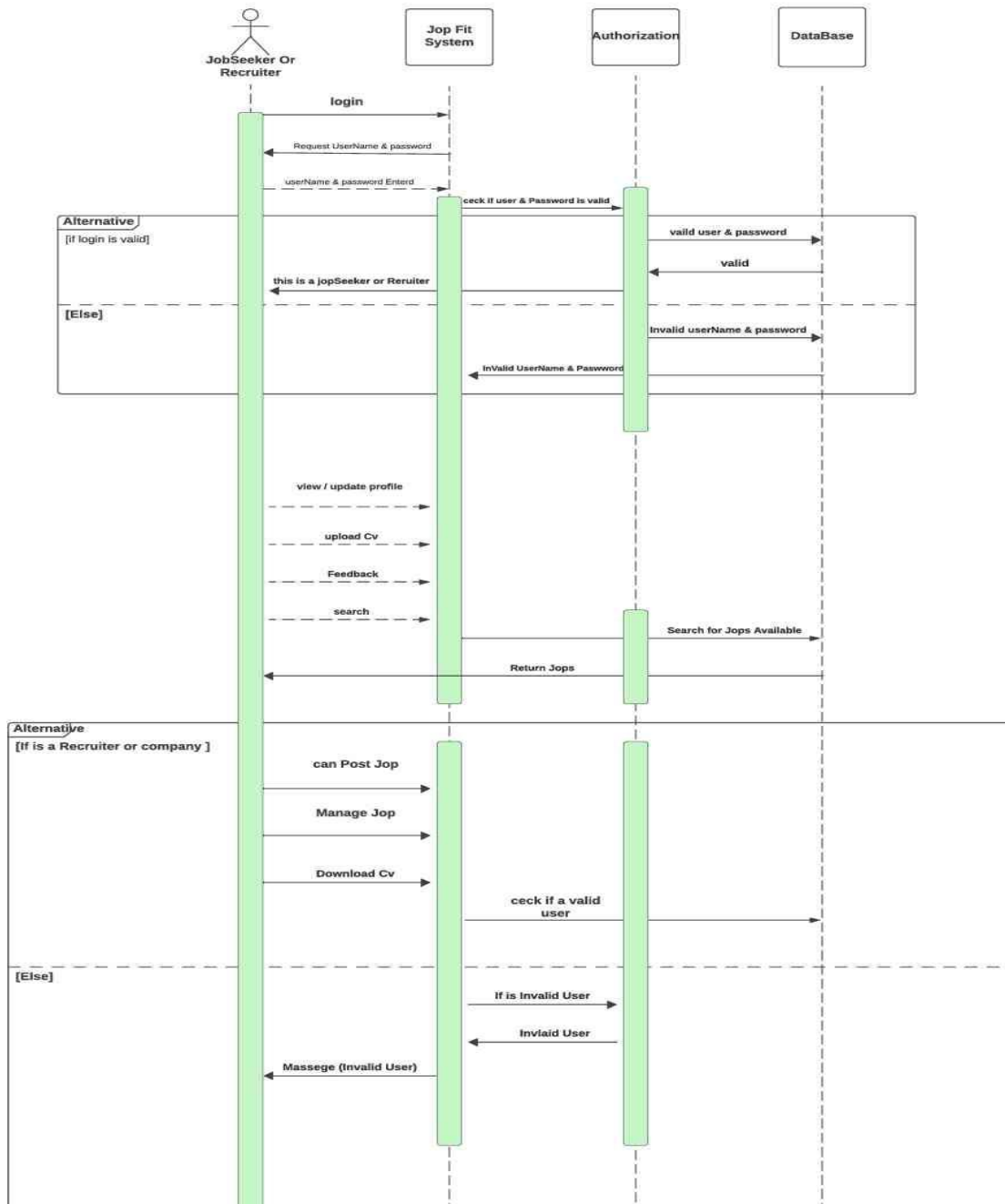


Figure 5 Sequence Diagram

Database Design

In the context of our project, the database plays a pivotal role in ensuring the efficiency, reliability, and seamless functionality of the job portal system. The importance of the database can be underscored through several key aspects:

1. Data Storage and Retrieval:

- The database serves as the central repository for storing vast amounts of crucial data, including user profiles, job listings, applications, and system configurations. This structured storage facilitates efficient retrieval, enabling quick access to relevant information.

2. Data Integrity and Consistency:

- A relational database, such as MySQL, provides a framework for enforcing data integrity through constraints and relationships. This ensures that data remains accurate and consistent throughout the system, preventing discrepancies that could compromise the reliability of the job portal.

3. Scalability:

- As the job portal system grows and manages an increasing volume of users, a well-designed database allows for seamless scalability. MySQL, being a relational database, offers robust scalability features, enabling the system to manage larger datasets and a growing user base.

4. Query Performance:

- The relational model of MySQL allows for optimized query performance. This is crucial for quickly retrieving and displaying relevant job listings, user profiles, and other data, contributing to a responsive and user-friendly experience.

5. Data Relationships:

- The relational database model excels at managing complex relationships between different entities in the system. For example, establishing relationships between users and their applications or connecting job listings with specific recruiters. This relational structure enhances the overall functionality of the job portal.

6. ACID Compliance:

- MySQL ensures ACID (Atomicity, Consistency, Isolation, Durability) compliance, providing a robust foundation for transactional operations. This is essential for maintaining the integrity of the database, especially during critical operations such as user registrations, job applications, and financial transactions.

7. Security:

- Database security is paramount, especially when dealing with sensitive user information. MySQL offers robust security features, including user authentication, access control, and encryption, safeguarding the confidentiality and integrity of the data stored in the system.

In conclusion, the choice of MySQL as a relational database for our job portal project is instrumental in achieving a well-structured, scalable, and high-performance system. The database not only acts as a reliable storage solution but also plays a crucial role in ensuring data integrity, security, and optimal query performance, contributing significantly to the overall success and functionality of the job portal.



Figure 6 Database Schema Diagram

In the ERD diagram, there are three main tables: User, Jobs, and Application. The User table is linked to other tables like user_profile, covering details such as jobs_skills, experience, education, languages, and more.

Server-side Architecture

It's split into three parts:

1. Core Functions:

- This part manages the essential features and tools the system needs, including the recommendation engine.

2. Recommendation Engine Area:

- Inside the Core Functions, there's a specific area for the recommendation engine. This part makes it easy to add and use model scripts.

3. API Endpoints:

- Dependent on the Core Functions, API Endpoints handle external interactions. They're like the system's communication hub, offering functions to users and other services.

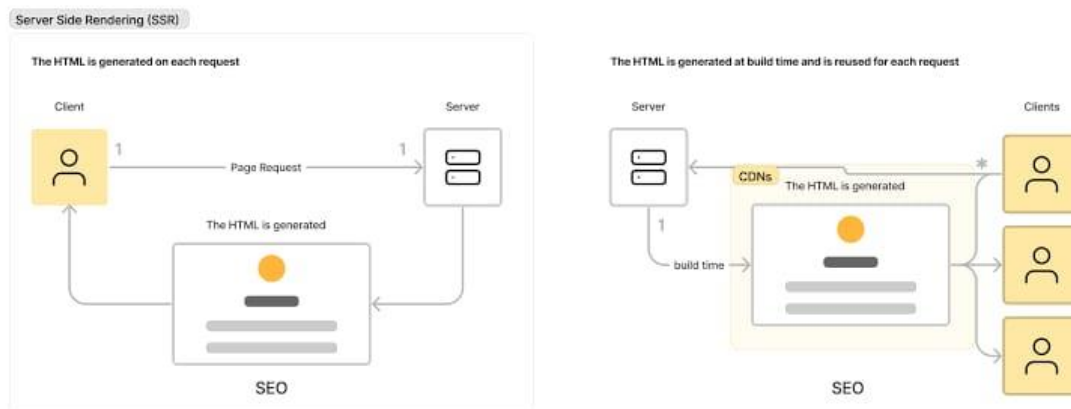


Figure 7 Server-side Rendering

Main Entities in Prisma

```
model User {  
  id          String      @id @default(cuid())  
  name        String?  
  email       String?     @unique  
  emailVerified DateTime?  
  image       String?  
  password    String?  
  phone       String?  
  banned      Boolean      @default(false)  
  role        Role         @default(STANDARD)  
  accounts    Account[]  
  profile     Profile?  
  jobs        Job[]  
  applications Application[]  
  companies   Company[]  
}
```

Figure 8 User Model

```
model Account {  
  id          Int         @id @default(autoincrement())  
  userId      String  
  type        String  
  provider    String  
  providerAccountId String  
  refresh_token String? @db.Text  
  access_token String? @db.Text  
  expires_at  Int?  
  token_type  String?  
  scope       String?  
  id_token    String? @db.Text  
  session_state String?  
  
  user User @relation(fields: [userId], references: [id], onDelete: Cascade)  
  
  @@unique([provider, providerAccountId])  
}
```

Figure 9 Account Model

```

model Job {
  id          Int          @id @default(autoincrement())
  title       String
  description  String
  location    String
  createdAt   DateTime     @default(now())
  updatedAt   DateTime     @updatedAt
  user        User         @relation(fields: [userId], references: [id], onDelete: Cascade)
  userId      String
  type        JOB_TYPE     @default(FullTime)
  status      JOB_STATUS   @default(Active)
  applicants  Application[]
  company     Company      @relation(fields: [companyId], references: [id], onDelete: Cascade)
  companyId   Int
}

```

Figure 10 Job Model

```

model Application {
  id          Int          @id @default(autoincrement())
  jobId       Int
  userId      String
  status      Status       @default(PENDING)
  resume      String
  coverLetter  String?
  createdAt   DateTime     @default(now())
  updatedAt   DateTime     @updatedAt
  user        User         @relation(fields: [userId], references: [id])
  job         Job          @relation(fields: [jobId], references: [id])
}

model Company {
  id          Int          @id @default(autoincrement())
  name        String
  location    String?
  website     String?
  image       String?
  description  String?
  founded     DateTime?
  createdAt   DateTime     @default(now())
  updatedAt   DateTime     @updatedAt
  user        User?        @relation(fields: [recruiterId], references: [id])
  recruiterId String?
  jobs        Job[]
  Experience  Experience[]
}

```

Figure 11 Application & Company Models


```

model Profile {
  id          Int          @id @default(autoincrement())
  userId      String
  bio         String?
  location    String?
  website     String?
  resume      String?
  createdAt   DateTime     @default(now())
  updatedAt   DateTime     @updatedAt
  skills      Skill[]
  languages   Language[]
  experiences  Experience[]
  educations  Education[]
  user        User         @relation(fields: [userId], references: [id], onDelete: Cascade)

  @@unique([userId])
}

```

Figure 12 Profile

```

model Experience {
  id          Int          @id @default(autoincrement())
  title       String
  company     Company?    @relation(fields: [companyId], references: [id], onDelete: SetNull)
  companyId   Int?
  location    String
  from        DateTime
  to          DateTime
  type        JOB_TYPE?   @default(FullTime)
  current     Boolean
  description  String
  profile     Profile      @relation(fields: [profileId], references: [id], onDelete: Cascade)
  profileId   Int
}

model Education {
  id          Int          @id @default(autoincrement())
  school      String
  degree      String
  field       String
  from        DateTime
  to          DateTime
  current     Boolean
  description  String
  gpa         Float?
  profile     Profile      @relation(fields: [profileId], references: [id])
  profileId   Int
}

```

Figure 13 Experience & Education

Client-side Architecture

AJAX

Is a technology that allows web developers to create dynamic, interactive websites without having to reload the page each time a user interacts with the website.

An API (Application Programming Interface) is a set of rules and protocols that allow two different applications or services to communicate with each other.

AJAX relies on an API to make requests from the server and receive data from it.

The API provides a way for the AJAX code to make requests to the server and receive data back in format that can be used by the client-side code. This allows for dynamic, interactive websites without having to reload the page each time a user interacts with it.

Fetching data from the server

Another very common task in modern websites and applications is retrieving individual data items from the server to update sections of a webpage without having to load an entire new page.

Instead of the traditional model, many websites use JavaScript APIs to request data from the server and update the page content without a page load. So, when the user searches for a new product, the browser only requests the data which is needed to update the page.

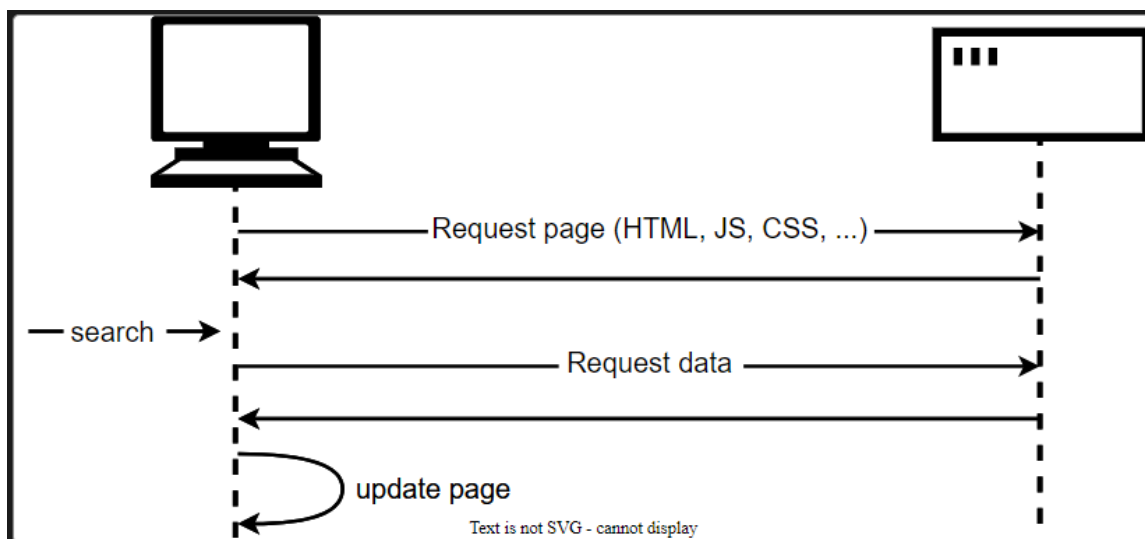


Figure 14 Data fetching

The main API here is the Fetch API. This enables JavaScript running in a page to make an HTTP request to a server to retrieve specific resources. When the server provides them, the JavaScript can use the data to update the page, typically by using DOM manipulation APIs. The data requested is often JSON, which is a good format for transferring structured data, but can also be HTML or just text.

React

React is a JavaScript Library for building interactive user interfaces.

It's widely used for building single-page applications (SPAs) due to its component-based architecture and efficient rendering using a virtual DOM.

By user interfaces (UI), we mean the elements that users see and interact with on-screen.

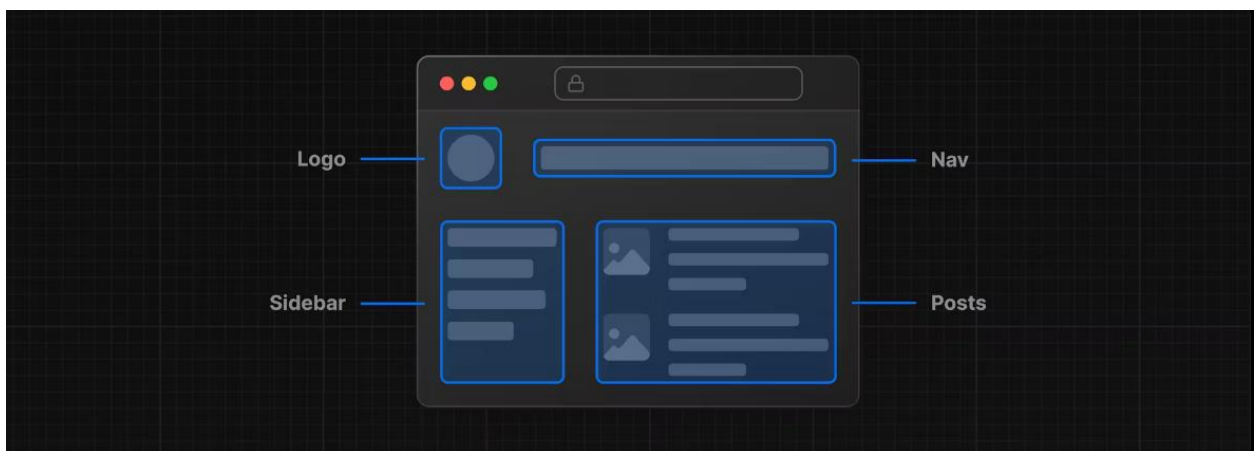


Figure 15 React Diagram (1)

By library, we mean React provides helpful functions (APIs) to build UI

Benefits of choosing React Technology

Component-Based Architecture:

Encourages the development of reusable and modular components, making code more maintainable and scalable.

Virtual DOM:

Enhances performance by efficiently updating only parts of the DOM that need changes, reducing the number of direct manipulations to the actual DOM.

Next.js

Next.js is a React framework that gives you building blocks to create web applications.

By framework, we mean Next.js handles the tooling and configuration needed for React, and provides additional structure, features, and optimizations for your application.

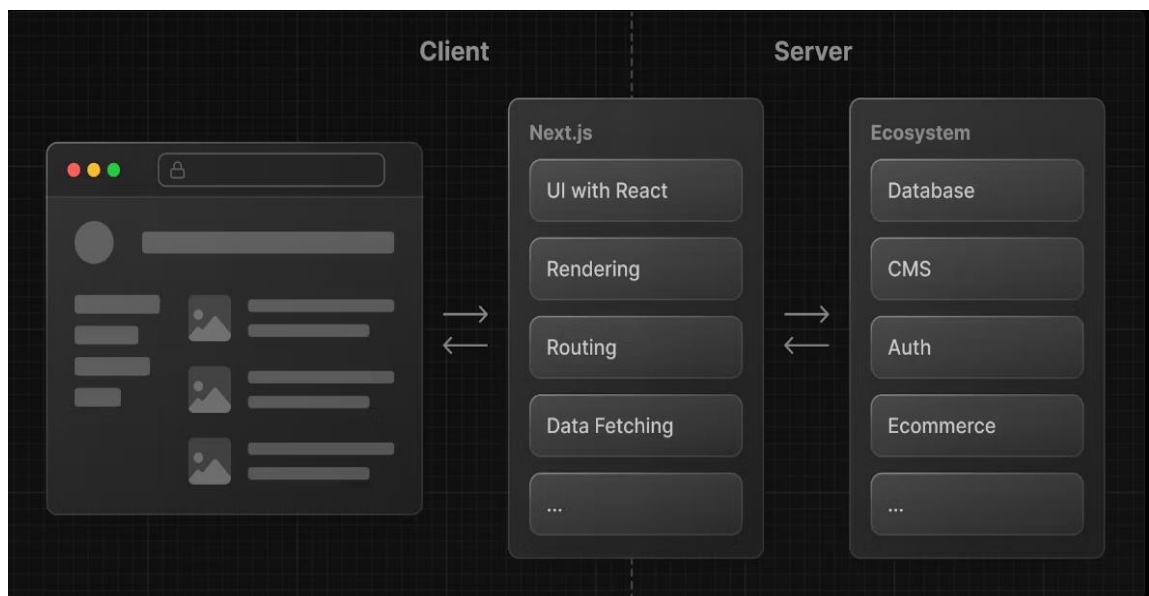


Figure 16 React Diagram (2)

Benefits of choosing Next.js Technology:

Server-Side Rendering

Improves initial load times and SEO by rendering pages on the server before sending them to the client.

Static Site Generation

Generates static HTML pages at build time, which are served instantly to users, enhancing performance and scalability.

API Routes

Provides a way to create backend endpoints within the Next.js app, eliminating the need for a separate backend server for small to medium-sized applications

Material UI (MUI)

Material-UI (MUI) is a popular React component library that implements Google's Material Design.

It provides a wide range of pre-built components that follow Material Design principles, which can significantly speed up the development process and ensure a consistent, aesthetically pleasing user interface

API Integration in React

Integrating an API in React.js involves making HTTP requests to the API endpoints and handling the data received in your React components.

There are various ways to achieve this, but the most common approach is to use the 'fetch' API or a library like Axios to handle the HTTP requests.

Authentication

Workflow:

1- User Authentication (Login):

The user enters their credentials (username and password) into the UI (e.g., a login form).

The UI client sends a POST request to the server's authentication endpoint, typically /login, with the user credentials.

2- Token Generation on the Server:

Upon successful authentication, the server generates a JWT containing claims such as the user's ID, roles, and expiration time.

The server signs the JWT using a secret key to create the token.

3- Token Transmission to the UI Client:

The server sends the JWT back to the UI client as part of the response, usually in the JSON payload.

The UI client stores the JWT securely, often in local storage or a secure HTTP cookie.

4- Subsequent Requests from the UI Client:

When the user performs actions or accesses protected resources, the UI client includes the JWT in the Authorization header of HTTP requests.

5- Server-Side Token Verification:

The server receives the JWT in the Authorization header.

It verifies the token's integrity by checking the signature using the shared secret key.

If the signature is valid, the server extracts claims from the payload.

6- User Authorization:

The server uses the extracted claims to authorize the user and determine their access rights.

For example, checking if the user has the required roles or permissions to perform the requested action.

7- Token Expiration and Refresh:

The server checks the expiration time (exp) of the JWT to ensure it's still valid.

If the token is expired, the UI client needs to refresh it by re-authenticating the user, typically using a refresh token or by prompting the user to log in again.

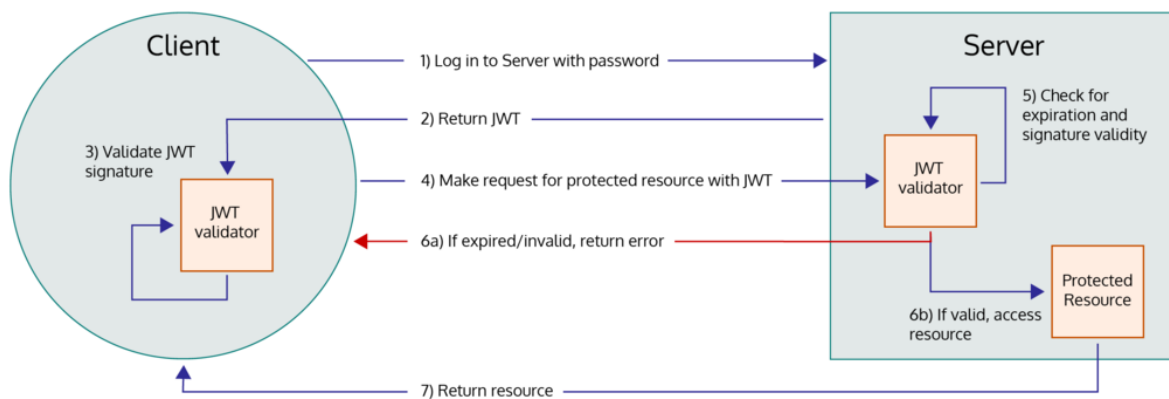


Figure 17 JWT Auth

NextJs Authentication

Authentication Strategies

Modern web applications commonly use several authentication strategies:

1. **OAuth/OpenID Connect (OIDC):** Enable third-party access without sharing user credentials. Ideal for social media logins and Single Sign-On (SSO) solutions. They add an identity layer with OpenID Connect.
2. **Credentials-based login (Email + Password):** A standard choice for web applications, where users log in with an email and password. Familiar and easy to implement, it requires robust security measures against threats like phishing.
3. **Passwordless/Token-based authentication:** Use email magic links or SMS one-time codes for secure, password-free access. Popular for its convenience and enhanced security, this method helps reduce password fatigue. Its limitation is the dependency on the user's email or phone availability.
4. **Passkeys/WebAuthn:** Use cryptographic credentials unique to each site, offering high security against phishing. Secure but new, this strategy can be difficult to implement.

Selecting an authentication strategy should align with your application's specific requirements, user interface considerations, and security objectives.

Implementing Authentication

In this section, we'll explore the process of adding basic email-password authentication to a web application. While this method provides a fundamental level of security, it's worth considering more advanced options like OAuth or passwordless logins for enhanced protection against common security threats. The authentication flow we'll discuss is as follows:

1. The user submits their credentials through a login form.
2. The form sends a request that is handled by an API route.

3. Upon successful verification, the process is completed, indicating the user's successful authentication.
4. If verification is unsuccessful, an error message is shown.

Technology Stack

In constructing our job portal system, we have strategically assembled a technology stack at the forefront of industry trends. Our chosen stack incorporates cutting-edge solutions, emphasizing innovation and optimal performance. The backend is developed with a language known for its modern capabilities, offering a foundation for robust and scalable functionality. The data layer is implemented with a powerful database solution, enabling efficient storage and retrieval of essential information. Real-time communication is facilitated through advanced tools, enhancing user engagement. Our commitment to security and efficiency is further reflected in the incorporation of industry-leading practices, ensuring that our job portal system remains at the forefront of technological advancements in the field.

So far in our technology stack, we rely on:

1. **Front End & Back End**

Next.js:

Description: Next.js is used as a full-stack framework to handle both front-end and back-end functionalities. It allows for server-side rendering and static site generation, providing a dynamic and responsive user experience while managing server-side operations and API routes.

Key Features:

Server-Side Rendering (SSR): Enhances performance and SEO by rendering pages on the server.

Static Site Generation (SSG): Generates static HTML pages at build time.

API Routes: Simplifies the creation of backend API routes within the Next.js application.

Routing: Offers a file-based routing system, making navigation straightforward and efficient.

Full-Stack Capabilities: Combines front-end and backend logic in a single framework, streamlining development and deployment.

2. **AI Integration:**

Python:

Description: Python is employed for creating and integrating machine learning models. These models enhance application functionalities, such as the recommendation engine.

Key Features:

- **Machine Learning Libraries:** Utilizes libraries such as TensorFlow, PyTorch, or scikit-learn for model development.
- **Data Processing:** Handles data preprocessing, feature extraction, and model training.
- **Integration:** Python scripts and models are integrated into the Next.js application, providing advanced AI-driven features.

3. Database

MySQL with Prisma ORM:

- **Description:** MySQL serves as the relational database management system for the application, efficiently storing and retrieving data. Prisma ORM is used to facilitate type-safe database interactions and streamline database management.
- **Key Features:**
 - **Structured Data Storage:** Organizes data into tables with predefined schemas.
 - **Type-Safe Queries:** Prisma provides an intuitive and type-safe query builder for interacting with the database.
 - **Schema Management:** Prisma schema defines the database structure and relationships, making migrations and updates straightforward.
 - **Performance Optimization:** Includes features for optimizing query performance and database access patterns.

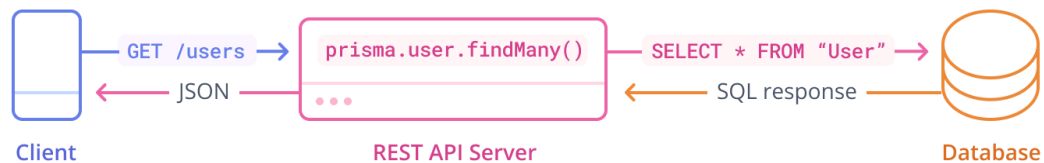


Figure 18 Prisma Rest

AI Integration



Finding the perfect career match can be both challenging and time-consuming. Our recommendation system is here to transform the job-seeking experience by offering intelligent and personalized job recommendations based on the job seeker's CV which includes their skills, experience, preferences, etc.

For employers/recruiters, it ensures efficient candidate matching, resulting in better recruitment outcomes.

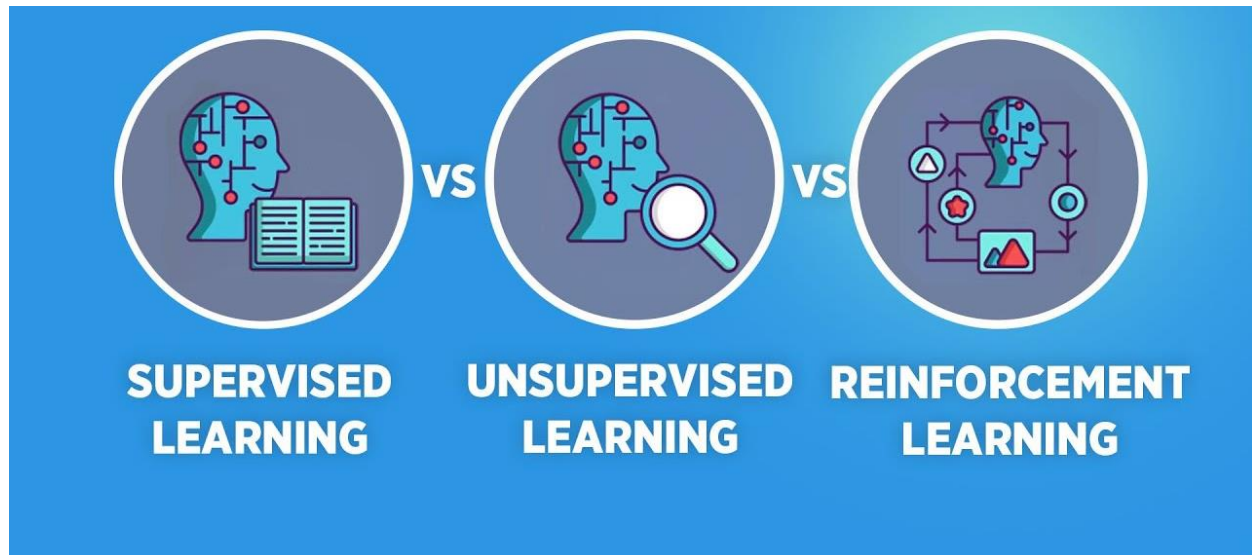
Key Highlights

Smart Candidate Matching: Our recommendation system employs sophisticated algorithms to match candidates with job opportunities that align with their skills, experience, and career aspirations.

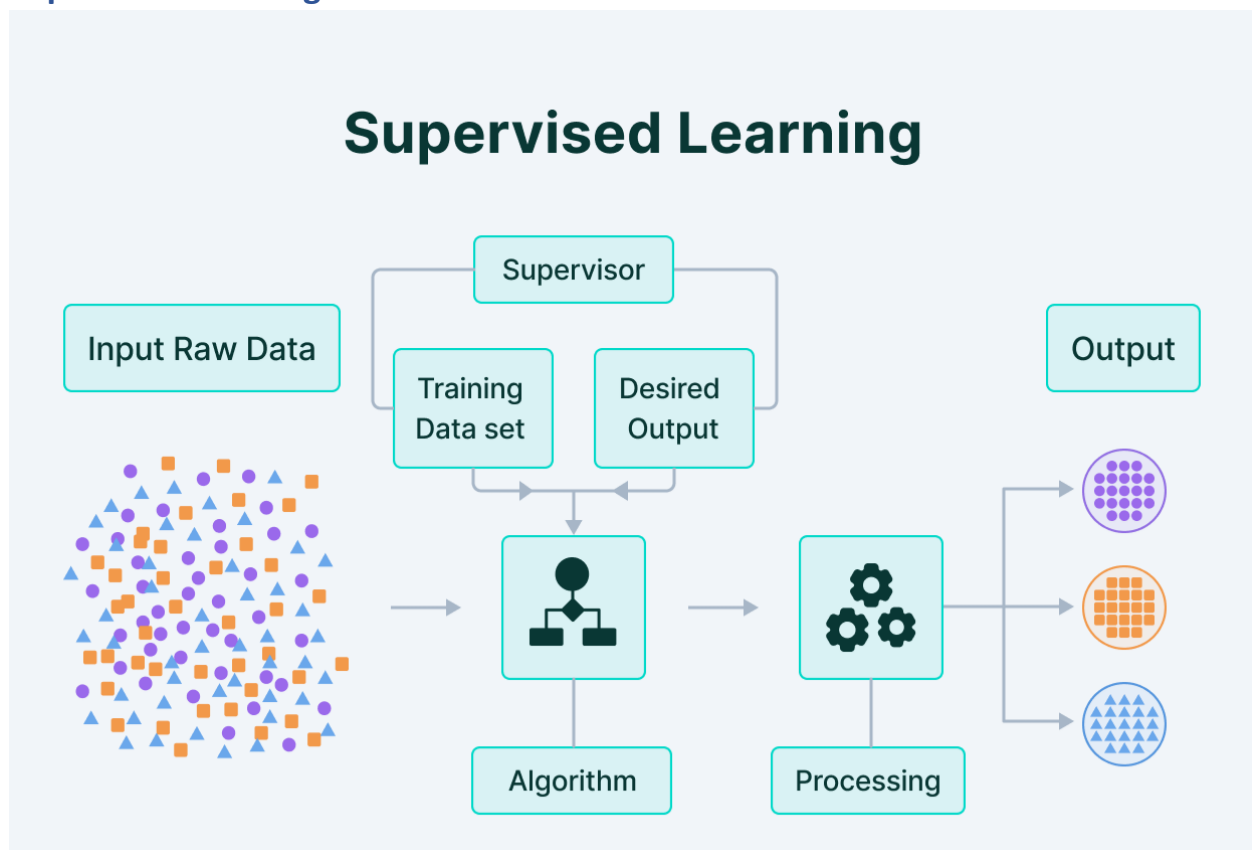
Enhanced Employer Insights: For employers, the system recommends candidates who best fit their job requirements, leading to more efficient and targeted hiring decisions.

User-Centric Experience: Prioritize user satisfaction by delivering a personalized job search experience that adapts to individual preferences and career goals.

Machine Learning Algorithms



Supervised Learning



Supervised learning is a type of machine learning where the algorithm is trained on a **labelled** dataset, which means that each training example is paired with an output label.

These are the technique(s) we implemented on our project using **Supervised Machine Learning**.

1. Support Vector Machines (SVM)

Purpose: Classification and regression tasks.

Description: It finds the hyperplane that best separates the classes in the feature space. For nonlinear problems, it can use kernel functions to transform the data into a higher-dimensional space.

Example: Image classification tasks.

2. Decision Tree

Purpose: Classification and regression tasks.

Description: A tree-like model of decisions where each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label or regression value.

Example: Diagnosing diseases based on symptoms.

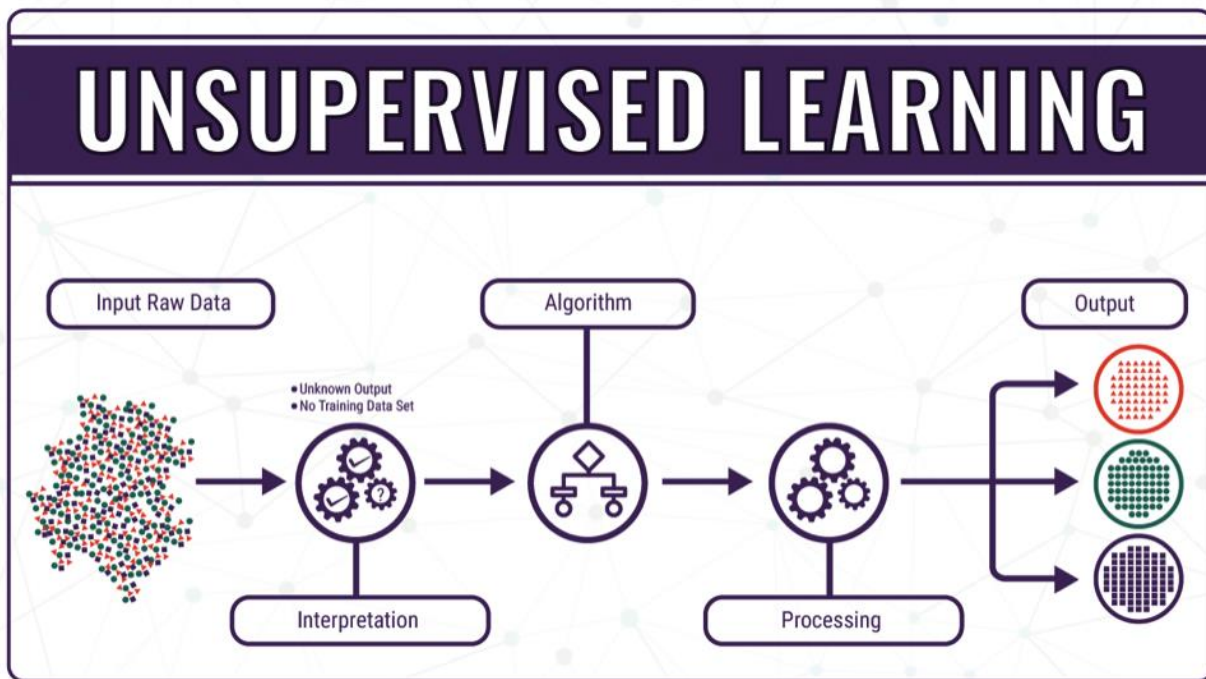
3. Random Forests

Purpose: Classification and regression tasks.

Description: An ensemble method that builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Example: Credit scoring systems.

Unsupervised Learning



Supervised learning is a type of machine learning where the algorithm is trained on an **unlabelled** dataset.

These are the technique(s) we implemented on our project using **Unsupervised Machine Learning**.

K-Means Clustering

Purpose: Partitioning data into distinct groups (clusters).

Description: The algorithm divides the data into k clusters, where each data point belongs to the cluster with the nearest mean, serving as a prototype of the cluster.

Example: Market segmentation to group customers with similar behaviours.

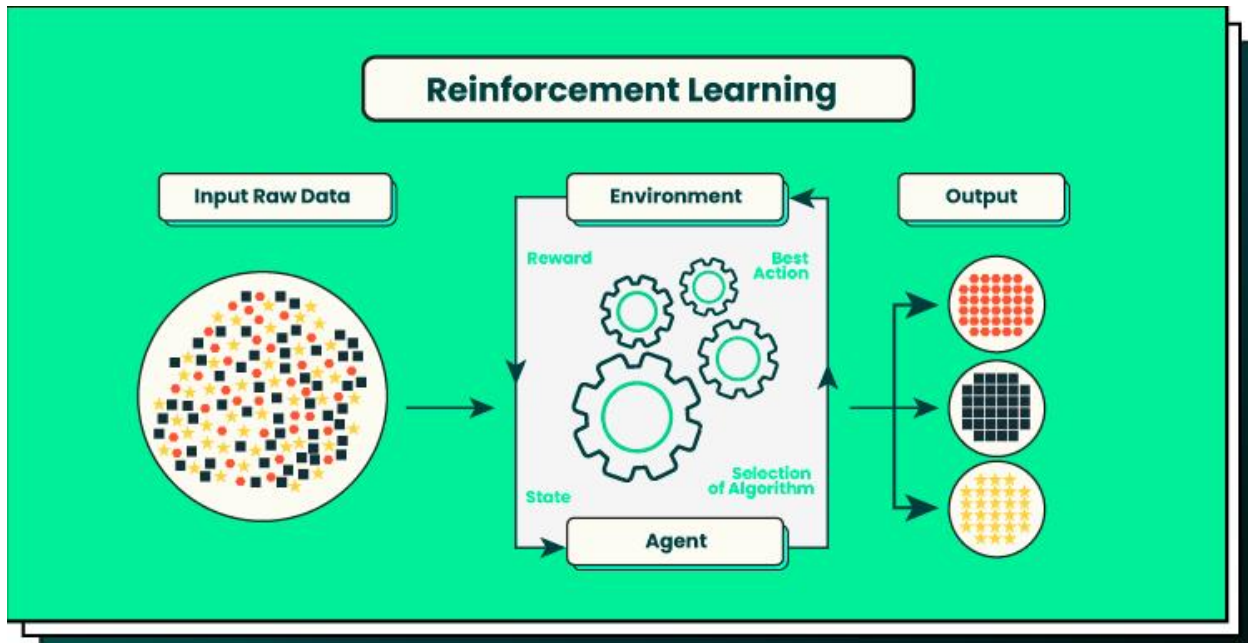
Natural Language Processing (NLP)

Purpose: Is to bridge the communication gap between humans and machines

Description: NLP empowers computers to interpret human language

Example: Allows smartphones to understand human voice commands.

Reinforcement Learning



Reinforcement learning (RL) is a type of machine learning (ML) where an agent learns to make decisions by interacting with an environment, using a trial-and-error approach to maximize cumulative rewards.

We did not implement **RL** in our project due to having better outcomes and results from Supervised and Unsupervised Machine Learning.

1. Q-Learning

Purpose: Learning optimal action-value functions.

Description: It learns an action-value function that gives the expected utility of taking a particular action in a given state and following the optimal policy thereafter.

Example: Training a robot to navigate a maze.

2. Deep Q-Networks (DQN)

Purpose: Extending Q-learning to deep neural networks.

Description: Uses a neural network to approximate the action-value function, allowing for more complex state representations and potentially faster learning.

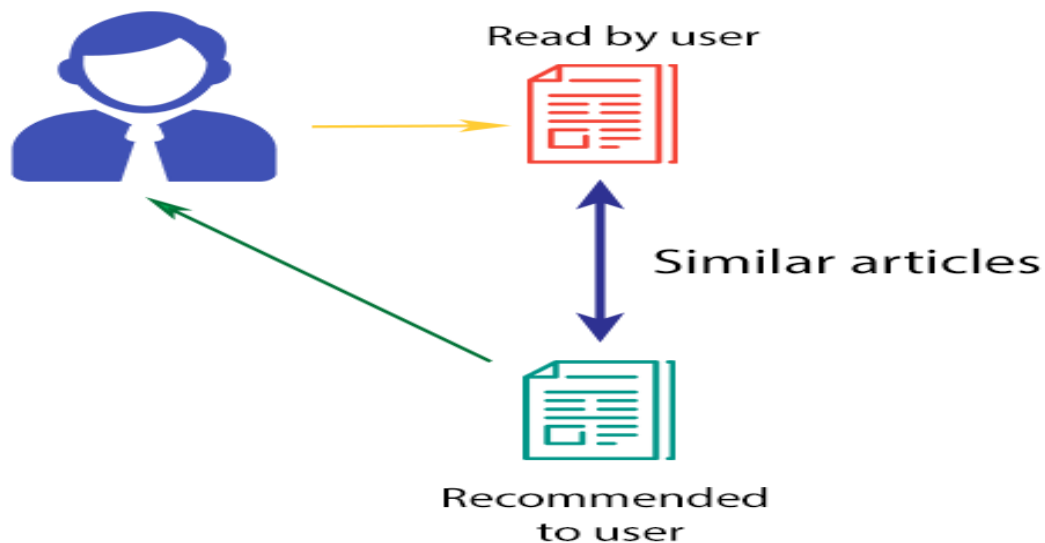
Example: Playing Atari games where the agent directly observes pixel data.

Types of Recommendations systems

In job websites, there are four main ways to recommend jobs. By using these different methods, the suggestions become more diverse and accurate. It's helpful to mix these methods to get the best results for job recommendations.

Content-Based Recommendations

CONTENT-BASED FILTERING



How It Works

In the content-based approach, our recommendation system analyses the inherent features of jobs and job seekers to generate precise recommendations. By focusing on key attributes provided in their CV/Resume such as skills, education, qualifications, and preferences, the system tailors suggestions that align with individual user profiles.

Strengths

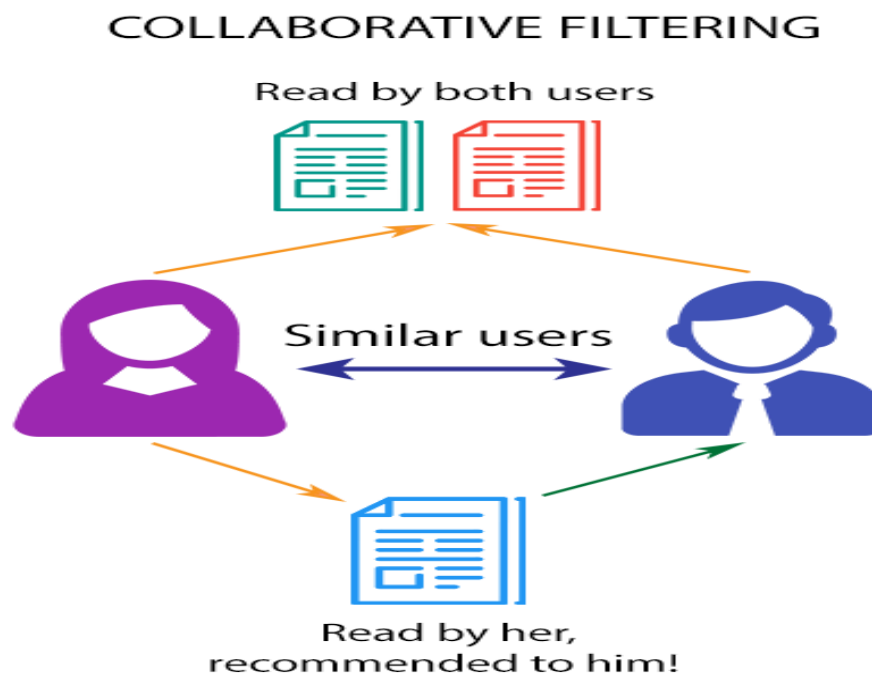
- Personalized to individual user preferences.

- Adaptive. Captures changes in user's interests.
- Recommends unpopular products, for instance if an employer/recruiter is looking for or hiring a Software Engineer that only speaks Spanish, then the system will target Software Engineers that speaks Spanish.
- Items recommended for one user do not depend on other users.

Considerations

- Limited diversity in recommendations.
- Depends heavily on details. Needs good information about items to work well.
- Takes time to adjust to changing user preferences effectively.

Collaborative Filtering



How It Works

Collaborative filtering relies on collective user behaviour to make recommendations. By identifying patterns and similarities among job seekers, the

system suggests jobs that align with the preferences and actions of job seekers with similar profiles.

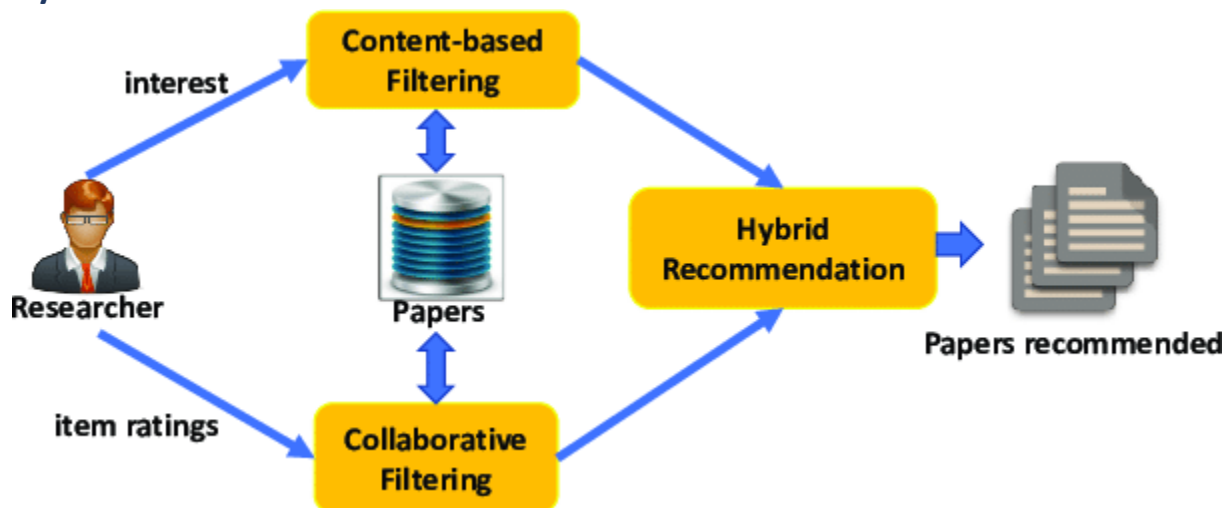
Strengths

- The algorithm understands what you might like through natural preferences by observing others with similar tastes.
- Versatile suggestions where it recommends various jobs.

Considerations

- Faces difficulty in making accurate recommendations for new job seekers with no history of preferences established.
- Data Sparsity can be found due to the lack of sufficient data.

Hybrid Recommendations



How it works

Within our job portal framework, Hybrid Recommendations seamlessly integrate two powerful methods: content-based and collaborative filtering. Think of it as an

experienced job matchmaker. It not only understands your individual job preferences but also learns from the collective choices of other job seekers who share similarities with you.

Strengths

- Offers personalised job suggestions by examining your CV/Resume contents with precision.
- Ability to expand recommendations to various job types, ensuring more complete view to offered jobs by recruiters.
- Also adapts to changes in job seeker's behaviour and preferences.

Considerations

- Implementation can be complex; the combination requires careful setup and management.
- Regular maintenance and adjustments are necessary to maintain performance.

According to our research, choosing the Content-Based Recommendation System for our AI Integration System means we prioritize personalized and accurate job suggestions to job seekers. This system pays attention to job details and contents within the CVs/Resumes, making it great for delivering recommendations that really fit the job seeker's preferences, unlike other algorithms, this algorithm offers a more straightforward experience for job seekers even if you're new to the platform. So, by using the Content-Based Recommendation System, we aim to make your job search experience more personalized and satisfying.

Dataset

First Dataset

In this section of our documentation, we will be explaining the dataset used in our system.

- **Title (Job Title)**

In our first column the Title column is necessary that is going to be provided by the recruiter to display the jobs available relevant to its location to the job seeker in our case this column is needed hence it has a proportional relationship with our sixth column (Skills) we will talk about it more below in our “Skills” sections below.

- **Company**

In our second column it represents the recruiter’s company name that is assigned to the Job provided in the system.

- **Location**

In our third column the Location column provides the job seeker an idea of where the job is going to be located by the recruiter.

- **Type**

In our fourth column the Type column gives an insight to the job seeker whether the job will be Full-time job or Part-time job.

- **Level**

In our fifth column the Level column informs the job seeker what level the job position offers for example, entry level, experienced, manager or work from home.

- **YearsExp**

In our sixth column the YearsExp column is needed to provide an idea to the job seeker how many years of experience is needed to apply for the job position.

- **Country**

In our seventh column the Country column provides the country allocated with the job by the recruiter.

- **Skills**

In our last column the Skills column is yet to be one of the most important if not the most important column in our dataset.

Hence, it's the column we will be applying our algorithm to, in our case it's the recommendation system algorithms where we will be fetching the data and most important contents provided within the CV/Resume to recommend the job seeker the best suited jobs provided in our first column "Title".

Second Dataset

Job Id	Title	Role	Company Profile
20801	Frontend Developer	Software Engineer	Frontend Developer
10340	Software Engineer	Software Engineer	Software Engineer
84225	Software Engineer	Software Engineer	Software Engineer
120906	Software Engineer	Software Engineer	Software Engineer
13944	Software Engineer	Software Engineer	Software Engineer
21106	Software Engineer	Software Engineer	Software Engineer
20510	Software Engineer	Software Engineer	Software Engineer
40550	Software Engineer	Software Engineer	Software Engineer
103543	Software Engineer	Software Engineer	Software Engineer
102069	Software Engineer	Software Engineer	Software Engineer
130320	Software Engineer	Software Engineer	Software Engineer
117285	Software Engineer	Software Engineer	Software Engineer
79071	Software Engineer	Software Engineer	Software Engineer
127900	Software Engineer	Software Engineer	Software Engineer
40128	Software Engineer	Software Engineer	Software Engineer
40100	Software Engineer	Software Engineer	Software Engineer
20518	Software Engineer	Software Engineer	Software Engineer
120630	Software Engineer	Software Engineer	Software Engineer
40190	Software Engineer	Software Engineer	Software Engineer
64351	Software Engineer	Software Engineer	Software Engineer
118541	Software Engineer	Software Engineer	Software Engineer
40721	Software Engineer	Software Engineer	Software Engineer
127385	Software Engineer	Software Engineer	Software Engineer
20840	Software Engineer	Software Engineer	Software Engineer
61318	Software Engineer	Software Engineer	Software Engineer
103370	Software Engineer	Software Engineer	Software Engineer
51353	Software Engineer	Software Engineer	Software Engineer
90970	Software Engineer	Software Engineer	Software Engineer
50930	Software Engineer	Software Engineer	Software Engineer
64185	Software Engineer	Software Engineer	Software Engineer
101834	Software Engineer	Software Engineer	Software Engineer
130992	Software Engineer	Software Engineer	Software Engineer
30844	Software Engineer	Software Engineer	Software Engineer
27725	Software Engineer	Software Engineer	Software Engineer
35270	Software Engineer	Software Engineer	Software Engineer
102060	Software Engineer	Software Engineer	Software Engineer
20884	Software Engineer	Software Engineer	Software Engineer
104700	Software Engineer	Software Engineer	Software Engineer
47107	Software Engineer	Software Engineer	Software Engineer

Figure 19 Second Dataset

As we can clearly see, our second dataset is filled with completed data, but it also led to coding issues, in which the data was so large that the accuracy score was 1.0 (Overfitting) in the Title column.

Also, we calculated the Location column accuracy score and it was below 0.05 (Underfitting), so this dataset was not ideal.

Figure 20 Accuracy Score Third Dataset

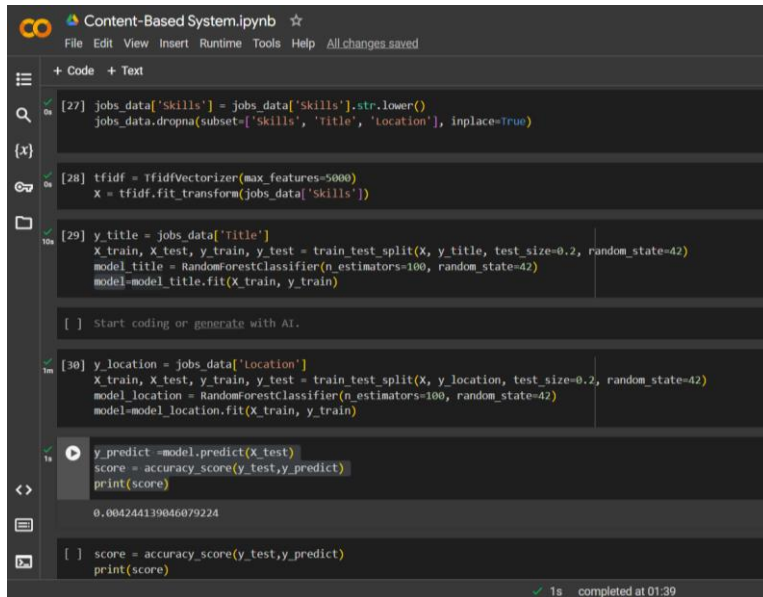
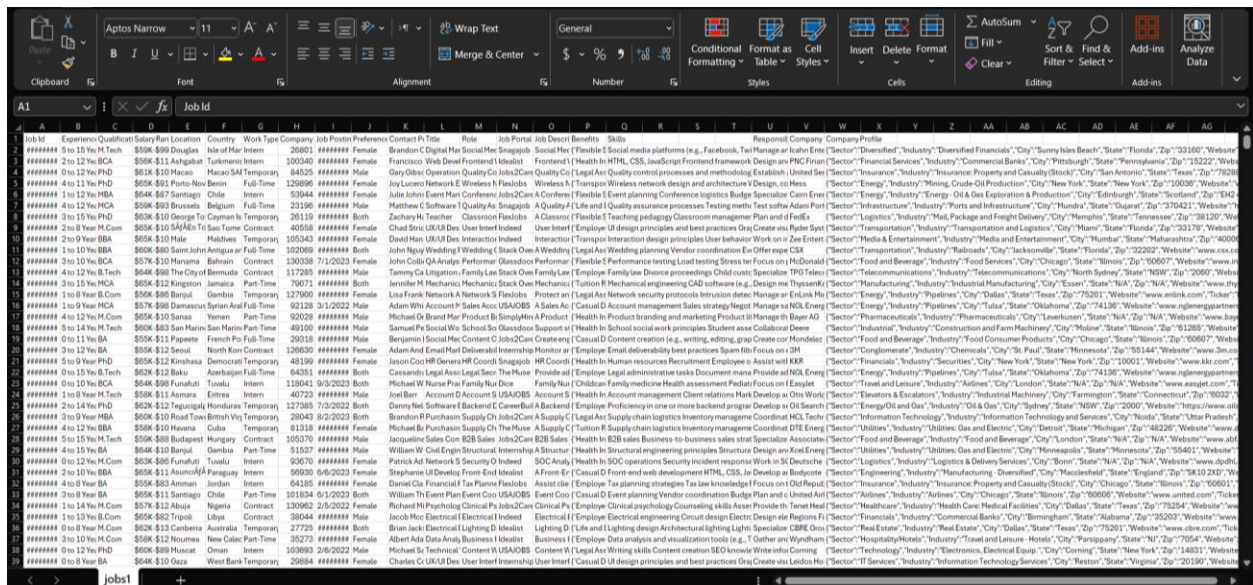


Figure 21 Third Dataset



Here also, our dataset is not ideal due to the very low accuracy score described in the picture (0.004) it was underfitted for our results so, we are going to move on to our final and best fitted dataset.

Current Dataset

Figure 22 Current Dataset

Job Title	Job Experience	Key Skills	Role Category	Location	Functional	Industry	Role	Salary
Digital Media Planner	5 - 10 yrs	Media Planning	Advertising	Mumbai	Marketing	Advertising	Media Planner	3855
Online Bidding Executive	2 - 5 yrs	pre sales	Retail Sales	Pune, Pune	Sales , Retail	IT-Software	Sales Executive	2639
Trainee Research/ Researcher	0 - 1 yrs	Computer	R&D	Gurgaon	Engineering	Recruitment	R&D Executive	2156
Technical Support	0 - 5 yrs	Technical	Admin/Management	Mumbai	IT Software	IT-Software	Technical Support	4059
Software Test Engineer	2 - 5 yrs	manual testing	Programming	Hyderabad	IT Software	IT-Software	Testing Engineer	5347
Opening For Adobe After Effects	5 - 7 yrs	adobe experience	Programming	Pune	IT Software	IT-Software	System Administrator	4838
Sales- Fresher-for Lead Generation	0 - 0 yrs	channel partner	Retail Sales	Bengaluru	Sales , Retail	Real Estate	Sales Executive	2585
Opportunity For Azure	9 - 14 yrs	TFS Azure	Programming	Hyderabad	IT Software	IT-Software	Technical Support	3914
BDE- PUNE	2 - 7 yrs	Bde	Institutional	Pune	Sales , Retail	IT-Software	Sales Executive	2111
Technical Support/ Project Manager	1 - 5 yrs	technical support	Voice	Bengaluru	ITES , BPO	IT-Software	Associate/Manager	2359
Executive Assistant To	5 - 10 yrs	secretary	Corporate	Mumbai	Strategy , Marketing	Courier, Travel	Corporate	5569
SEO Executive	1 - 6 yrs	website	Admin/Management	Noida, Delhi	IT Software	IT-Software	Webmaster	2167
Workflow Coordinator	2 - 7 yrs	operations	Operations	Delhi, Delhi	ITES , BPO	BPO, Call Center	Process Flow	2063
Oracle IDAM	3 - 7 Years	Oracle IDAM	Programming	Bengaluru	IT Software	IT-Software	Software Developer	3239
Looking Facebook /social media	2 - 4 yrs	digital marketing	Online/Digital	Gurgaon	Marketing	Advertising	Social Media	6979
Wanted Engineering	0 - 1 yrs	instrumentation	engineering	Chennai	Other	Other	Other	1513
Tooling & Sampling	2 - 7 yrs	Help Desk	Engineering	Faridabad	Engineering	Industrial	Senior Designer	6622
Account Manager	3 - 7 yrs	Report Generation	Accounts	Mumbai	Accounts , IT-Software	Accounts	Manager	5324
Magento Developer	2 - 7 yrs	Copyright	Programming	Ahmedabad	IT Software	Textiles, Garments	Software Developer	6891
Looking For Trained Fresher	0 - 3 yrs	C# MS Dynamics	Programming	Hyderabad	IT Software	IT-Software	Software Developer	6493
Job Openings Kotak Life	3 - 7 yrs	sales manager	Retail/Personal	Bhilai/Bhilai	Financial Services	Insurance	Sales Officer	5623
Business Developer	0 - 4 yrs	lead generation	Retail Sales	Delhi NCR	Sales , Retail	IT-Software	Sales/Business	1564
Business Developer	2 - 5 yrs	contract	Retail Sales	Delhi, Delhi	Sales , Retail	IT-Software	Sales/Business	5439
QA Executive	2 - 5 yrs	QA Executive	QA/Testing	Hyderabad	IT Software	Pharma, Biotech	Quality Assurance	5434
Back End Java Developer	6 - 8 yrs	Java EE JSP	Programming	Bengaluru	IT Software	IT-Software	Software Developer	4114
Product Engineer	4 - 9 yrs	Conceptual	Programming	Bengaluru	IT Software	IT-Software	Software Developer	4897
Opening For After Effects	2 - 5 yrs	After Effects	Programming	Chennai	IT Software	Animation	Graphic Artist	4626

Job Title

In our first column the Job Title column is necessary that is going to be provided by the recruiter to display the jobs available to the job seeker in our case this column is needed hence it has a proportional relationship with our sixth column (Key Skills) we will talk about it more below in our “Key skills” sections below

Job Experience

In our second column the Job Experience Required column is also a column that gives the job seeker the required job experience to apply for a certain job provided by the recruiter.

Key Skills

In our third column the Key Skills column is yet to be one of the most important if not the most important column in our dataset.

Hence, it's the column we will be applying our algorithm to, in our case it's the recommendation system algorithms where we will be fetching the data and most important

contents provided within the CV/Resume to recommend the job seeker the best suited jobs provided in our first column “Job Title”.

Role Category

In our fourth column the Role Category column is needed to categorize and classify each job that is provided by the recruiter, so the job seeker could understand the kind of roles available within the job.

Location

In our fifth column the Location column provides the job seeker an idea of where the job is going to be located by the recruiter.

Functional Area

In our sixth column the Functional Area column provides the job seeker an idea of which fields of profession categories their provided CV/Resume can be applied to, it's also related to our fourth column in the dataset “Role Category” in which it collects the categories that the CV/Resume can fit into.

Industry

In our seventh column the industry column refers to the sector or field in which the recruiter's organisation/company operates.

Roles

In our second to last and last column the Role column provides a more specific information than our seventh column “Role Category” to the job seeker,

It specifies the exact job titles or specific roles within the broader category provided by the recruiter, basically it gives the job seeker an idea of what exact role they will work in the organisation.

Job Salary (sal)

In our last column the Job Salary column is a column that gives the job seeker an idea of the salary to expect within the job provided by the recruiter the job salary can be provided or not disclosed by the recruiter.

```

Content-Based System.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[122] y_pred_title = model_title.predict(X_test_title)
accuracy_title = accuracy_score(y_test_title, y_pred_title)
print("Accuracy Score for Job Titles:", accuracy_title)

Accuracy Score for Job Titles: 0.8695652173913043

[117] y_location = jobs_data['location']
X_train_location, X_test_location, y_train_location, y_test_location = train_test_split(X, y_location, test_size=0.2, random_state=42)

[118] y_train_location = y_train_location.fillna('Unknown')
y_test_location = y_test_location.fillna('Unknown')

model_location = RandomForestClassifier(random_state=42)
model_location.fit(X_train_location, y_train_location)
y_pred_location = model_location.predict(X_test_location)
accuracy_location = accuracy_score(y_test_location, y_pred_location)
print("Accuracy Score for Job Locations:", accuracy_location)

Accuracy Score for Job Locations: 0.7448979591836735

[156] def predict_job_title_and_location(skills, top_n=5, relevance_threshold=0.00):
    skills_transformed = tfidf.transform([skills.lower()])

    probs_title = model_title.predict_proba(skills_transformed)[0]
    top_n_indices_title = np.argsort(probs_title)[-top_n:]
    top_n_probs_title = probs_title[top_n_indices_title]
    relevant_titles = [model_title.classes_[i] for i, prob in zip(top_n_indices_title, top_n_probs_title) if prob >= relevance_threshold]

```

Figure 23 Accuracy Score Current Dataset

As we can see, this is the best fitted dataset between other datasets mentioned, the dataset is clean with enough information, and by default it was already showing better accuracy score, but we will fix it to make it better in the next pictures.

```

[61] jobs_data.dropna(subset=['Job Title'], inplace=True)

[62] tfidf = TfidfVectorizer()
X = tfidf.fit_transform(jobs_data['Key Skills'].astype(str))

[75] y_title = jobs_data['Job Title']
X_train_title, X_test_title, y_train_title, y_test_title = train_test_split(X, y_title, test_size=0.22, random_state=40)

[76] y_train_title = y_train_title.fillna('Unknown')
y_test_title = y_test_title.fillna('Unknown')

model_title = RandomForestClassifier(random_state=40)
model_title.fit(X_train_title, y_train_title)
y_pred_title = model_title.predict(X_test_title)
accuracy_title = accuracy_score(y_test_title, y_pred_title)
print("Accuracy Score for Job Titles:", accuracy_title)

Accuracy Score for Job Titles: 0.9335576114381834

[66] y_location = jobs_data['Location']
X_train_location, X_test_location, y_train_location, y_test_location = train_test_split(X, y_location, test_size=0.2, random_state=42)

[67] y_train_location = y_train_location.fillna('Unknown')
y_test_location = y_test_location.fillna('Unknown')

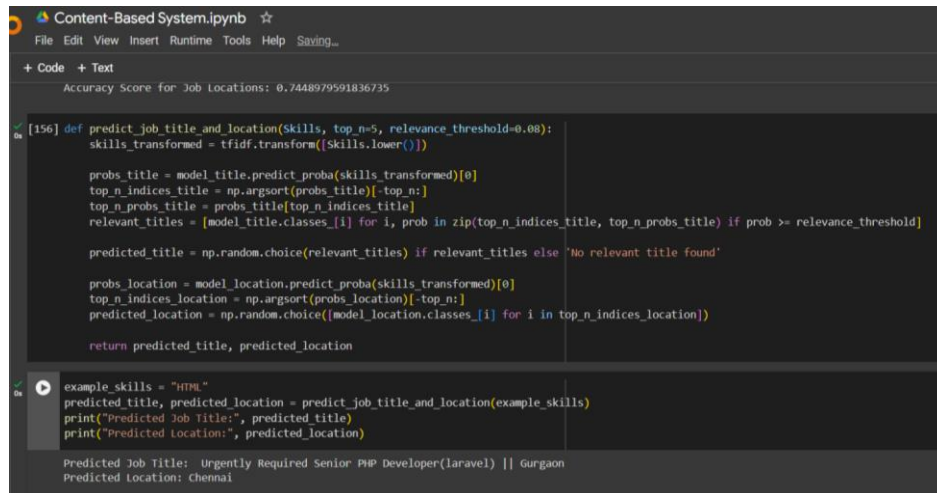
[68] model_location = RandomForestClassifier(random_state=42)
model_location.fit(X_train_location, y_train_location)
y_pred_location = model_location.predict(X_test_location)
accuracy_location = accuracy_score(y_test_location, y_pred_location)
print("Accuracy Score for Job Locations:", accuracy_location)

Accuracy Score for Job Locations: 0.9574468085106383

```

Figure 24 Updated Accuracy Scores Current Dataset

As we can see, after some coding adjustments we acquired a very good accuracy score, 0.93+ for both Job Titles and Job Locations columns, therefore we chose this dataset to be our best fitted dataset.



```
Content-Based System.ipynb ☆
File Edit View Insert Runtime Tools Help Saving...

+ Code + Text
Accuracy Score for Job Locations: 0.7448979591836735

[156] def predict_job_title_and_location(skills, top_n=5, relevance_threshold=0.00):
    skills_transformed = tfidf.transform([skills.lower()])

    probs_title = model_title.predict_proba(skills_transformed)[0]
    top_n_indices_title = np.argsort(probs_title)[-top_n:]
    top_n_probs_title = probs_title[top_n_indices_title]
    relevant_titles = [model_title.classes_[i] for i, prob in zip(top_n_indices_title, top_n_probs_title) if prob >= relevance_threshold]

    predicted_title = np.random.choice(relevant_titles) if relevant_titles else 'No relevant title found'

    probs_location = model_location.predict_proba(skills_transformed)[0]
    top_n_indices_location = np.argsort(probs_location)[-top_n:]
    predicted_location = np.random.choice([model_location.classes_[i] for i in top_n_indices_location])

    return predicted_title, predicted_location

example_skills = "HTML"
predicted_title, predicted_location = predict_job_title_and_location(example_skills)
print("Predicted Job title:", predicted_title)
print("Predicted location:", predicted_location)

Predicted Job Title: Urgently Required Senior PHP Developer(laravel) || Gurgaon
Predicted location: Chennai
```

Figure 25 Output Current Dataset

Due to our coding adjustments, the model can predict best fitted outputs which are the job title and location, with much more efficiency.

The Final Dataset's data was gathered from multiple references mentioned below in **References**.

AI Endpoint

Our Flask code is provided below in the following **Appendix** section in the document.

Flask Overview

Flask is a lightweight and flexible web framework for Python that allows developers to create web applications quickly and easily. It is designed to be simple and unopinionated, giving developers the freedom to structure their projects as they see fit. Flask provides essential tools and libraries to build web applications, including URL routing, request handling, and templating.

Why Use Flask?

- **Simplicity:** Flask is minimalistic and easy to set up, making it ideal for small to medium-sized applications.
- **Flexibility:** It doesn't impose a specific way of doing things, allowing developers to use the libraries and tools they prefer.
- **Extensibility:** Flask can be extended with numerous extensions to add functionality, such as database integration, form handling, and authentication.
- **Community:** It has a large and active community, providing plenty of resources, tutorials, and third-party extensions.

Explanation of Using One Endpoint

A single endpoint is utilized to serve a specific function within the application.

Endpoint: /predict

- **Purpose:** This endpoint is designed to predict the job title and location based on the skills provided in the request.
- **Method:** It uses the POST method, which is suitable for requests that involve sending data to the server to be processed.
- **Request Handling:** The endpoint expects a JSON object containing the user's skills and optionally, the parameters `top_n` and `relevance_threshold`.
- **Data Processing:** The endpoint transforms the input skills using a pre-trained TF-IDF vectorizer, then uses two Random Forest models to predict job titles and locations.
- **Response:** It returns a JSON response with the predicted job title and location.

Using a single endpoint in this context is beneficial for several reasons:

1. **Simplicity:** Having a single endpoint keeps the application straightforward and easy to understand, especially for developers who are maintaining or extending the codebase.
2. **Focus:** This endpoint focuses on a single responsibility—predicting job-related information based on skills—making it easier to manage and debug.
3. **Scalability:** If additional functionalities are needed in the future, new endpoints can be added without affecting the existing one. For instance, separate endpoints could be created for adding new jobs, updating job information, or retrieving job details.
4. **Efficiency:** Handling both job title and location predictions in one request reduces the need for multiple HTTP requests from the client, improving efficiency and reducing latency.

UI/UX Design

UI and UX design are critical components of product development, especially in digital environments like websites, mobile applications, and software. They are interconnected disciplines that focus on different aspects of the user's interaction with a product.

UI Design

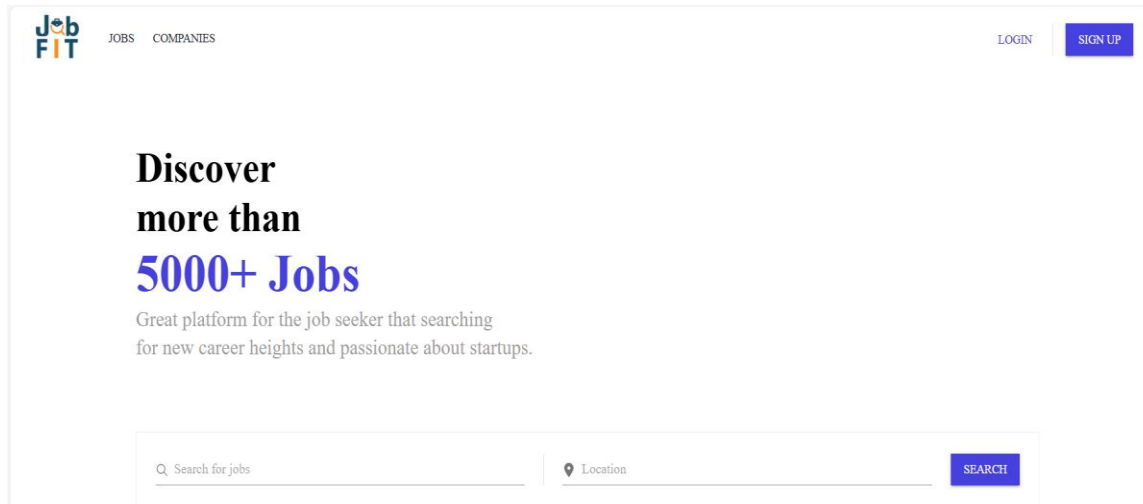
The User Interface (UI) section of our documentation delves into the visual and interactive elements that define the user experience within our job portal system. This segment provides a comprehensive overview of the design, layout, and functionality of the portal's interface. From the landing page to individual job listings and user profiles, this section explores how users will interact with the system, emphasizing a user-centric approach. By focusing on intuitive navigation, responsive design, and aesthetic appeal, we aim to create a seamless and engaging experience for both job seekers and recruiters. This UI section is structured to guide developers and stakeholders through the various components, ensuring a thorough understanding of the visual aspects that contribute to the overall success of our job portal.

UX Design

UX design focuses on the overall experience of the user as they interact with the product. It aims to create products that are not only functional but also enjoyable to use.

Preview

Figure 26 Home Page



A home page is the main web page that a visitor will view when they navigate to a website via a search engine, and it may also function as a landing page to attract visitors. In some cases, the home page is a site directory, particularly when a website has multiple home pages.

Our home page highlights are simplicity and straight-forward to aid the user that interacts with our home page, the user can navigate easily without being confused because everything is stated on the page and functions as labelled.

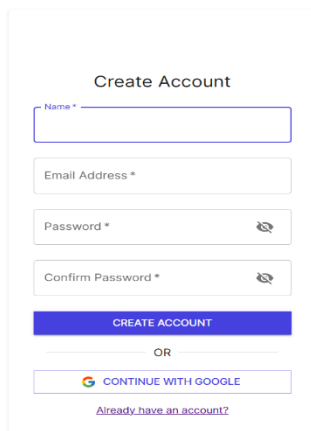
As shown in **Figure 13 Home page**, we included a search bar where the user can search for their desired job, if the user wants to search for a job generally without specifying their location, they can search for the desired job without adding location and then our page will navigate to all available jobs with multiple locations, and if they want to specify the location, then they can do so and our page will navigate and result the jobs with the location entered.

Also, the user can navigate directly to the **JOBS** tab to view all possible jobs listed in our program in general.

As well as the **COMPANIES** tab to view the companies and their job listing and to gain more information about them.

We also added **LOGIN** and **SIGN UP** for the user to create their accounts or log into their existing account.

Figure 27 Sign Up

A sign-up form titled "Create Account". It contains four input fields: "Name *", "Email Address *", "Password *", and "Confirm Password *". Each password field has a small eye icon to the right for toggling visibility. Below the fields is a blue "CREATE ACCOUNT" button. Underneath the button is the text "OR" and a button labeled "CONTINUE WITH GOOGLE" with the Google logo. At the bottom, there is a link that says "Already have an account?".

Here is our **Sign up** page where the user can create their account and register in our database.

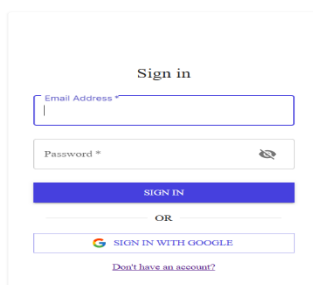
As we can see, we went for a simple design the gives the user a straightforward approach, where the user can include their name, email address and password with a confirmation in order to ensure integrity.

We also provided **Show password** button in **Password & Confirm Password** fields in order to aid and help the user to ensure they entered their password correctly.

If the user does not want to include all of their information like we stated above, the user can create their account using their **Google Email** for an easier approach.

If the user already has an account registered in our system, we included at the very bottom of the UI **`Already have an account?`** button in order to navigate them to the **Log in** page.

Figure 28 Log in page

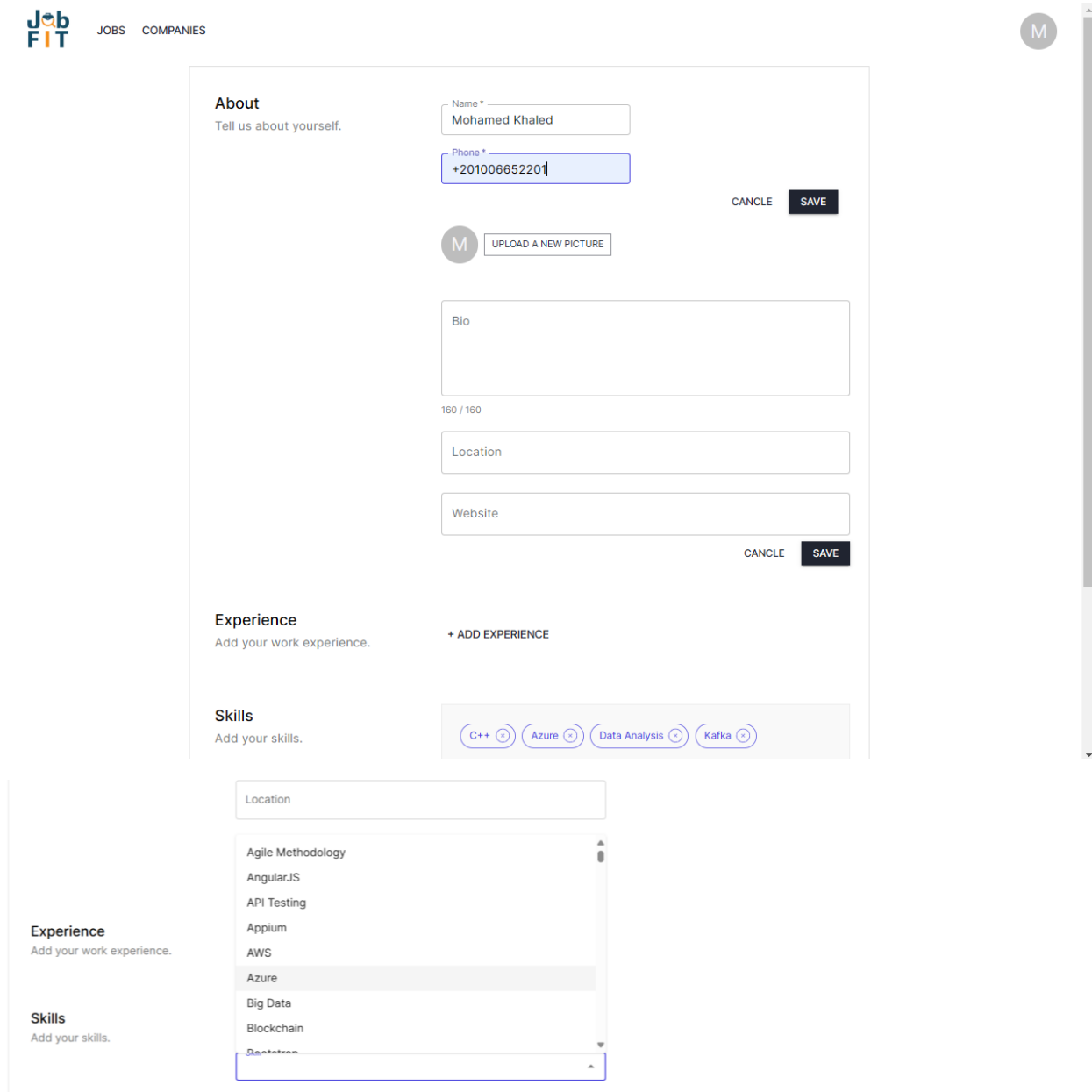
A log-in form titled "Sign in". It contains two input fields: "Email Address" and "Password *". The password field has a small eye icon to the right for toggling visibility. Below the fields is a blue "SIGN IN" button. Underneath the button is the text "OR" and a button labeled "SIGN IN WITH GOOGLE" with the Google logo. At the bottom, there is a link that says "Don't have an account?".

Here is our **Log in** page, we also went for a minimal simplistic approach to make it easy on the user to log into their account, we also provided the same approach for the **Show password** button to ensure the user that they entered their password correctly.

As stated above, we also provided logging in with **Google** so they can log into our system directly using **Google** account that is registered in our system.

If the user did not have an account, **'Don't have an account?'** button comes in handy and navigates them back to **Sign up** page.

Figure 29 Profile page



The screenshot displays the 'Job FIT' profile page. At the top left is the 'Job FIT' logo with 'JOBS' and 'COMPANIES' links. A user profile icon with the letter 'M' is in the top right. The main content area is divided into three sections: 'About', 'Experience', and 'Skills'. The 'About' section includes a 'Name' field with the value 'Mohamed Khaled', a 'Phone' field with '+201006652201', a 'Bio' field with a character count of '160 / 160', a 'Location' field, and a 'Website' field. Each of these fields has a 'CANCEL' button and a 'SAVE' button. Below the 'About' section is the 'Experience' section with a '+ ADD EXPERIENCE' button. The 'Skills' section shows a list of skills: C++, Azure, Data Analysis, and Kafka. A dropdown menu is open below the 'Skills' section, showing a list of skills including Agile Methodology, AngularJS, API Testing, Appium, AWS, Azure (highlighted), Big Data, Blockchain, and Docker.

Here the user can manage and edit that is needed in their profile.

In **About** section the user is required to provide their names and their phone number, if their satisfied with just providing those two or continue later, they can either cancel or save the process by clicking on the **Cancel/Save** Buttons.

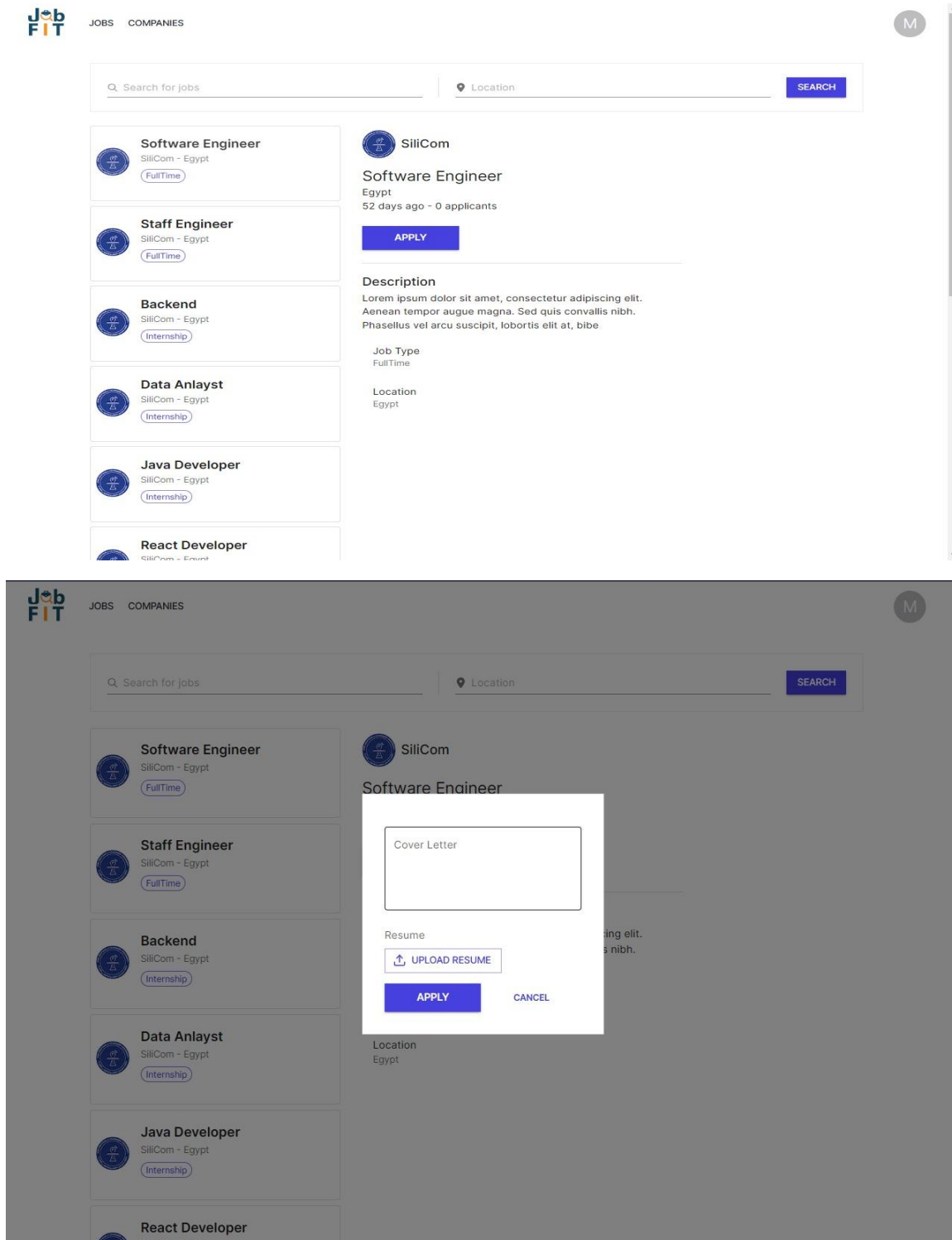
The user can upload their pictures for their profile, tell more about themselves in the **Bio** text box, provide their location as well as their website if they got one.

In the **Experience** section, the user can list all their past work experiences to improve their **CV/Resume**.

The **Skill** section allows the user to add their personal skills through either clicking the provided skills suggestions that we included on our system, or through a drop down box that has lists of experiences they can add, if the skills they got are not in our system then they can simply just write it down in our text box.

The **Page layout** is based on our home page where we provide the **Companies** and **Jobs** tab to navigate to their pages, they also can navigate back to the home page by simply clicking on our **JobFit** logo.

Figure 30 Jobs Page



The **Job** page provided everything for the about the jobs for the **Jobseeker**,

Where all jobs are listed on the left side of our panel, in which the jobs are previewed briefly below each other providing the job name, location of the job, company that listed the job, type in which whether the job is full-time, part-time, internship, etc.

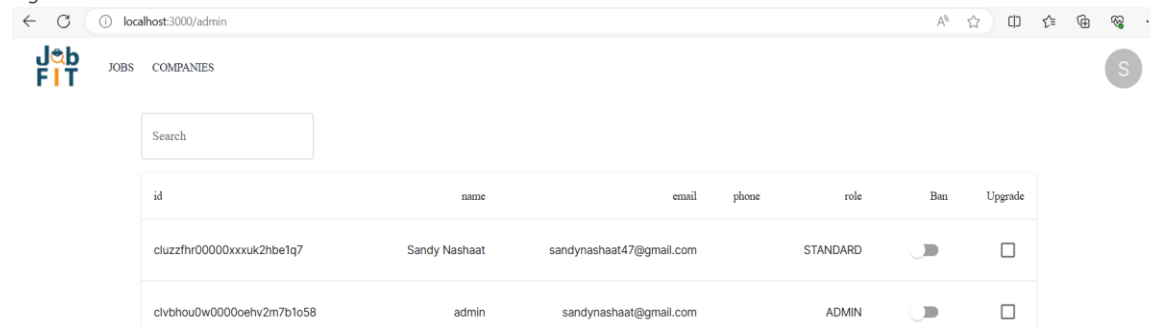
The user can also search for jobs to filter the jobs provided based on the jobs they searched for, or search by location to filter the jobs provided based on the location provided.

When clicked on the desired job, the right panel appears stating every detail that is within the job and its description, which helps the user to learn more about the job, and then we provided an **Apply** button to help the user in applying for the job if they are satisfied with the job.

Upon clicking on the **Apply button** the user can provide their cover letter in the cover letter text box and upload their **resume/cv** in whatever format like .pdf, .doc, etc.

If the user wishes to confirm their application they can click on the apply button or cancel the application if they are not satisfied.

Figure 31 Admin Dashboard



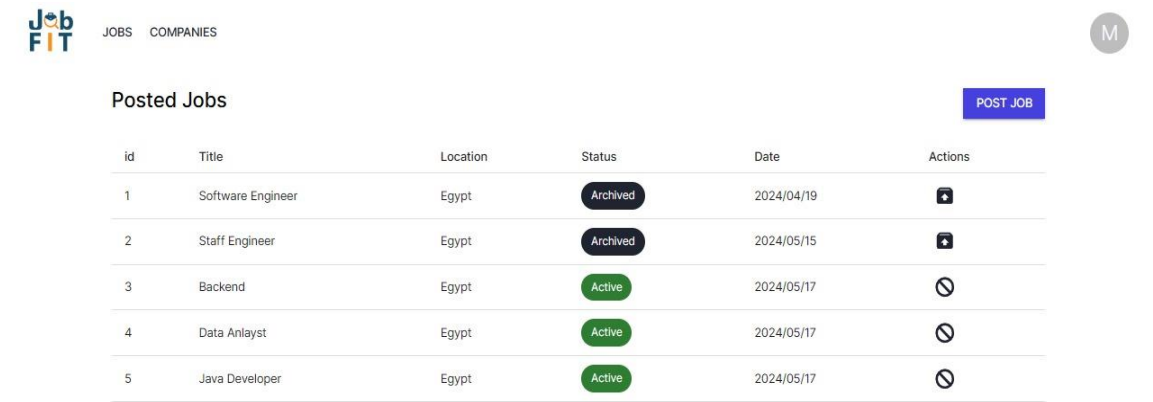
An **Admin Dashboard** is a single screen that includes all crucial information. In contrast, an admin or control panel allows for certain actions. It enables administrators to conduct efficient management, control the system's performance, and implement the required changes.

Through this page the **Admin** can:

Manage every administrative task provided in the system like banning, upgrading etc. Through interactive buttons provided.

Our dashboard helps the **Admin** to navigate easily by searching or navigating through every task.

Figure 32 Recruiter Dashboard

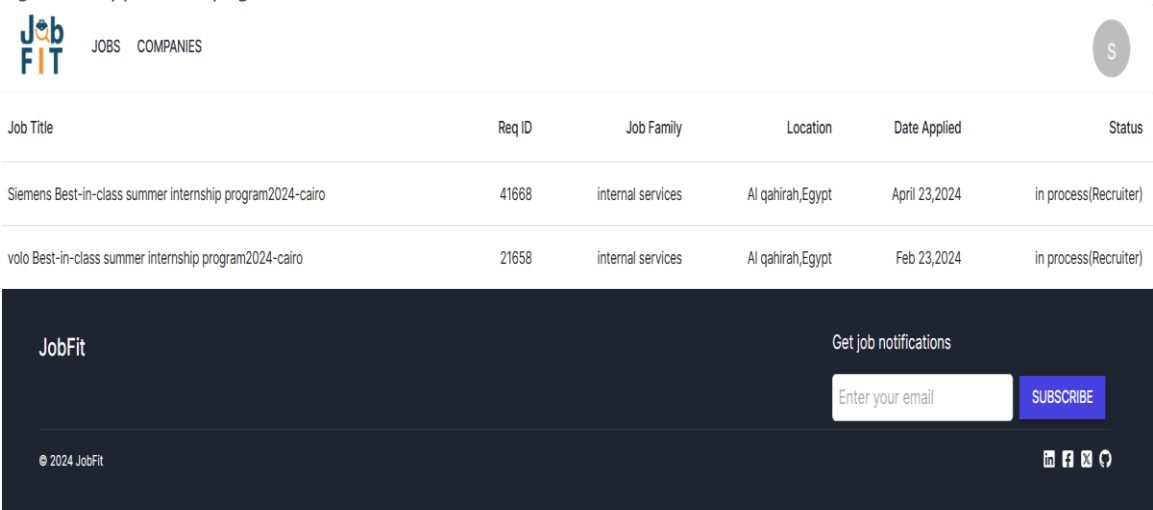


The **recruiter** is responsible for attracting, hiring, and onboarding new employees while ensuring compliance with labor laws.

They work with management and employees.

Our **Admin Dashboard** aids the **recruiter** to their responsibilities by having an easy interface, and buttons which helps them to **post job** and view them thoroughly.

Figure 33 Application page



This page shows the **Jobseeker** the jobs they applied for, the date of application, and the status of whether they got accepted or rejected.

Conclusion

The development of our advanced Job Portal System aims to revolutionize the job market by addressing its complexities and inefficiencies, offering enhanced experiences for both job seekers and recruiters. Through the integration of cutting-edge technologies and innovative strategies, our system provides a robust solution to streamline job matching processes and elevate overall user satisfaction.

Key Highlights of the Project:

- **Innovative Job Fit Analyzer:** Powered by AI and advanced software, our Job Fit Analyzer sets new standards in job matching with personalized recommendations and efficient algorithms.
- **Comprehensive Functionality:** The system caters comprehensively to diverse user needs, featuring sophisticated tools for job seekers (profile management, AI-driven job recommendations) and recruiters (job posting, applicant management).
- **Secure and Scalable Architecture:** Built on a secure three-tier architecture, integrating Next.js for frontend and backend capabilities, MySQL with Prisma ORM for robust data management, and Flask for AI model deployment. This architecture ensures scalability, performance optimization, and data security.
- **User-Centric Design:** Prioritizing user experience, the system employs Material-UI for aesthetic consistency, AJAX for dynamic interactions, and Next.js for server-side rendering and API handling.
- **Authentication and Security:** Utilizing JWT-based authentication for secure user access and authorization, ensuring the integrity and protection of user data.
- **AI Integration:** Incorporating machine learning models such as SVM and Decision Trees via Flask endpoints for intelligent job recommendation systems. This enables precise candidate matching and enhances recruitment outcomes.
- **Recommendation Systems:** Leveraging Content-Based Recommendation Systems to deliver personalized job suggestions based on user CV/resume content, ensuring relevance and user satisfaction.

Future Work

Implementation

Long-Term Vision

Our long-term goal is to transition from a semester-based project to a real-world application, continually improving our artificial intelligence, frontend, backend, and feature set. This evolution will involve:

- **AI Enhancements:** Advancing our AI capabilities to provide more intelligent and adaptive solutions.
- **Frontend Improvements:** Elevating the user experience through a more intuitive and responsive interface.
- **Backend Development:** Strengthening our backend infrastructure to support scalability, security, and high performance.
- **Feature Expansion:** Continuously adding new features and functionalities to meet user needs and industry standards.

By focusing on these areas, we aim to deliver a cutting-edge, real-world application that addresses the evolving demands of our users and stakeholders.

References

[1]

“Google Job Skills,” *www.kaggle.com*.

<https://www.kaggle.com/datasets/niyamatalmass/google-job-skills> (accessed Jun. 10, 2024).

NOTE ((OUR DATASET WAS INSPIRED AND IMPLEMENTED ON GOOGLE DATASET REFERENCE PROVIDED))

[2]

“Designing a Database for an Online Job Portal,” *Vertabelo Data Modeler*, Nov. 15, 2016. <https://vertabelo.com/blog/designing-a-database-for-an-online-job-portal/>

[3]

Z. Pajorska, “Guide to Recommendation System: Types, Selection Criteria, How to Build One,” *Stratoflow*, Oct. 19, 2023. <https://stratoflow.com/guide-to-recommendation-system/>

[4]

“What is the relationship between AJAX and an API?,” *Quora*.(accessed Jun. 10, 2024). <https://www.quora.com/What-is-the-relationship-between-AJAX-and-an-API>

[5]

“Client-side web APIs,” *MDN Web Docs*, Jun. 19, 2018.

[https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side web APIs](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs)

[6]

“Jobs in Egypt - WUZZUF | وظائف.كوم,” *WUZZUF*.

<https://wuzzuf.net/jobs/egypt>

[7]

indeed, “Job Search | Indeed,” *www.indeed.com*, 2022.

<https://indeed.com>

Appendix

[Flask integration.](#)