Tribhuvan University

Academia International College

Final Year Project Report
On
Job Recommendation System
[CSC 412]

Under the supervision of

"Mr. Ananda Adhikari"

Submitted by

Anusha Shrestha (T.U. Exam Roll No. 29006/078)

Shreya Bhattarai (T.U. Exam Roll No. 29029/078)

Sunand Shrestha (T.U. Exam Roll No. 29033/078)

Submitted to

Department of Computer Science and Information Technology

Academia International College

Institute of Science and Technology

Tribhuvan University

September 2025

Tribhuvan University

Academia International College

Final Year Project Report

On

"Job Recommendation System"

[CSC 412]

A final year project submitted in partial fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Information Technology awarded by Tribhuvan University

Submitted by

Anusha Shrestha (T.U. Exam Roll No. 29006/078)

Shreya Bhattarai (T.U. Exam Roll No. 29029/078)

Sunand Shrestha (T.U. Exam Roll No. 29033/078)

Submitted to

Department of Computer Science and Information Technology

Academia International College

Institute of Science and Technology

Tribhuvan University

September 2025

Tribhuvan University

Institute of Science and Technology

Academia International College

Department of Computer Science and Information Technology

Email: mail@academiacollege.edu.np

Supervisor's Recommendation

I hereby recommend that the project work report prepared under my supervision by Ms. Anusha Shrestha (29006/078), Ms. Shreya Bhattarai (29029/078) and Mr. Sunand Shrestha (29033/078) entitled "Job Recommendation System" be accepted as fulfilling in partial requirements for the degree of Bachelor of Science in Computer Science and Information Technology. In my best knowledge, this is an original work in Computer Science and Information Technology.

……..

Mr. Ananda Adhikari

Project Supervisor

Department of Computer Science and Information Technology

Academia International College

Tribhuvan University

Department of Computer Science and Information Technology

Academia International College

Certificate of Approval

This is to certify that this project prepared by Ms. Anusha Shrestha, Ms. Shreya Bhattarai and Mr. Sunand Shrestha entitled "Job Recommendation System" in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Information Technology has been well studied. In our opinion, it is satisfactory in the scope and quality as a project for the required degree.

…….. Mr. Ananda Adhikari Project Supervisor Department of Computer Science and IT Academia International College	…….. Mr. Bishwas Mathema HOD/Program Coordinator Department of Computer Science and IT Academia International College
…….. Internal Examiner Academia International College	…….. External Examiner Central Department of CSIT Tribhuvan University

Acknowledgement

We utter our utmost gratefulness to Academia International College for giving us this opportunity to work on this project as part of our syllabus. This project has provided us with hands-on experience and skills that extend beyond theoretical classroom learning

Special thanks to our program coordinator Mr. Bishwas Mathema and supervisor Mr. Ananda Adhikari, Academia International College for their consistent guidance, support, and feedback throughout the report's creation. We are beyond thankful to him and our college for providing this excellent opportunity to widen our expanse of knowledge. It helped us a lot to practically imply what we have studied.

We would like to express our sincere gratitude to everybody involved along with us for supporting and helping us a lot in turning this project within the limited time schedule by providing valuable insights and feedback on the report.

Thanking You,

Anusha Shrestha (T.U. Symbol No. 29006/078)

Shreya Bhattrai (T.U. Symbol No. 29029/078)

Sunand Shrestha (T.U. Symbol No. 29033/078)

Abstract

This project “Job Recommendation System” was developed to make it easier for people to find jobs that match their skills and interests. Job hunting today can feel overwhelming, with countless postings that often fail to align with what candidates truly want. Our system aims to simplify this process by filtering and recommending the most relevant opportunities based on a user’s profile. The system functions in a straightforward way. Users sign up, enter their skills, interests, or career preferences, and the platform generates job suggestions that closely match their input. It is built using Python and the Django framework and was tested on a small dataset of job listings. The recommendation process treats both job descriptions and user profiles as text documents. By applying TF-IDF (Term Frequency–Inverse Document Frequency), the text is converted into numerical values that highlight the most important words. These values are then compared using cosine similarity, a method that measures how close two documents are by checking the “angle” between them. A higher similarity score indicates a stronger match, helping users quickly identify suitable opportunities. This system is particularly valuable for students, fresh graduates, or job seekers who may not know where to begin. Instead of spending hours scrolling through job boards, they receive a curated list of jobs tailored to their background. In the future, the system could be enhanced by incorporating larger and continuously updated job databases, supporting resume uploads for automatic profile generation, and even offering career advice or skill-building recommendations. Overall, the project demonstrates how data, text mining, and machine learning can be combined to solve real-world challenges. It highlights the practical use of technology in connecting individuals with the right opportunities and making the job search process more efficient and effective.

Keywords: Job Recommendation System, TF-IDF, Cosine Similarity, Python, Django, Text Mining, Machine Learning, Job Matching

Table of Contents

List of Figures

List of Tables

List of Abbreviations

BERT	Bidirectional Encoder Representation from Transformers
CSS	Cascading Style Sheet
HTML	Hyper Text Markup Language
NLTK	Natural Language Toolkit
SMTP	Simple Mail Transfer Protocol
SQLite	Structured Query Lite
TFIDF	Term Frequency Inverse Document Frequency
VS Code	Visual Studio Code

Chapter 1: Introduction

1.1 Introduction

In the present time, people are mostly focused on seeking employment, whether in their specific field of study or in areas that offer better opportunities. However, the search for a suitable job is not always easy. Many job seekers face challenges such as too much information, irrelevant postings, and limited personalized guidance. With thousands of listings appearing online every day, individuals often struggle to identify the ones that truly match their skills, interests, and long-term career goals.

The Job Recommendation System is designed to simplify this process by connecting individuals with opportunities that genuinely suit their background and career goals. Using Term Frequency-Inverse Document Frequency (TF-IDF) and cosine algorithm, the system intelligently matches user profiles based on skills, experience, and preferences with relevant job openings.

This system is not just about listing jobs it is about recommending the right jobs. By learning from user interactions and leveraging existing datasets, the system provides tailored suggestions that improve with usage. Whether it's a fresh graduate seeking their first role or a mid-career professional looking to switch fields, the JRS aims to offer a smarter, faster, and more personalized way to navigate the job market.

1.2 Problem Statement

Job seekers often face significant challenges in finding suitable job opportunities due to the overwhelming number of job postings available online. Manually searching for jobs is not only time-consuming but also prone to errors, as individuals may overlook relevant opportunities or fail to identify roles that align with their skills and preferences. This inefficiency creates frustration for job seekers and results in missed opportunities for both candidates and employers. The lack of a personalized and efficient job search mechanism necessitates the development of a system that can streamline the process and provide tailored recommendations.

1.3 Objectives

- To develop a content-based job recommendation system using TF-IDF and Cosine Similarity techniques.
- To match users with job opportunities based on their provided skills, experience, and preferences.
- To enhance the job search experience by providing accurate and relevant recommendations.

1.4 Scope and Limitations

1.4.1 Scope

The scopes of this project are:

- It is an online web platform for job seekers looking for personalized opportunities.
- The development of a content-based filtering system that recommends jobs to users based on their input.
- It saves user time for searching and finding the appropriate job, then any other manual method and also provide suitable job for each candidate.

1.4.2 Limitations

- Recommendations depend on the availability and quality of job data.
- It does not incorporate collaborative filtering or learn from user behavior over time.
- System performance may vary depending on the algorithm and dataset.
- The recommendations are solely based on the textual similarity between user input and job descriptions, which may not account for other factors such as company culture or user preferences beyond the provided data.

1.5 Methodology

For the development of the Job Recommendation System, we followed an iterative process that allowed continuous refinement and improvement at every stage. This was achieved by adopting the agile approach throughout the project. This project was divided into several phases, including planning, design, implementation, and testing. During the planning period, the requirements were collected and analyzed to define the system's scope and objectives. The design phase involved creating system architecture, database models, and user interface prototypes. The implementation phase focused on coding and integrating the various modules, while the testing phase ensured the functionality of the system, reliability, and usability. This iterative approach allowed for continuous feedback and improvements throughout the development process.

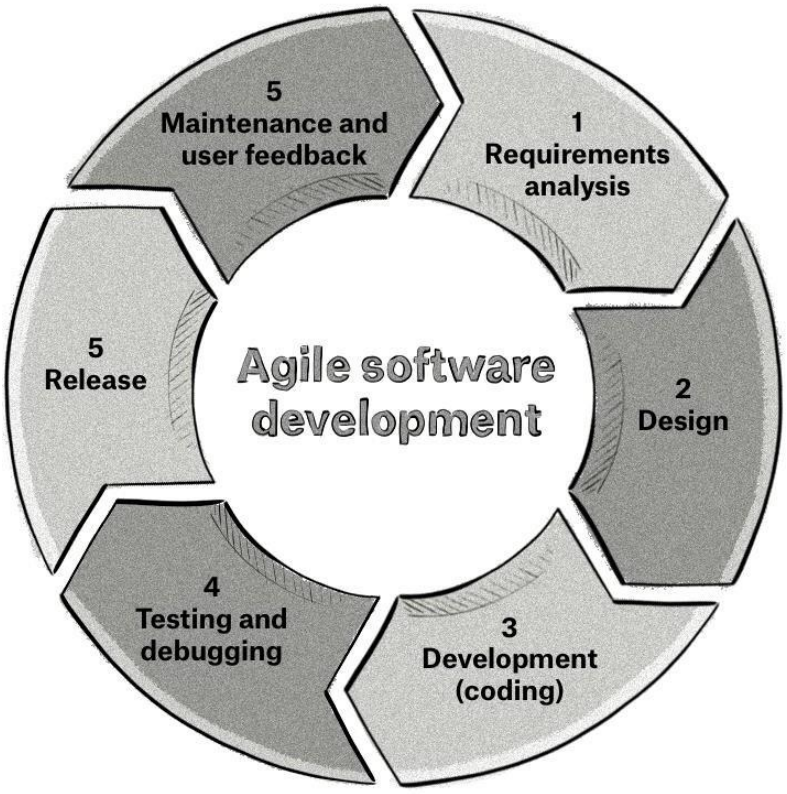

Figure 1.1: Agile Model

1.6 Report Organization

This report is divided into six main chapters:

Chapter 1: Introduction

It is the introductory part of the project which covers the overview, problem, objectives, and methodology of the project.

Chapter 2: Background & Literature Review

The second chapter is the background and literature that provides a background study and review, discussing existing systems and relevant techniques.

Chapter 3: System Analysis

It deals with System Analysis explaining the functional and non- functional requirements.

Chapter 4: System Design

This chapter focuses on system design and presents the design and algorithms used in the system.

Chapter 5: Implementation & Testing

This is all about implementation and testing phases, running our project and performing different tests.

Chapter 6: Conclusion & Future Recommendations

This is the conclusion and future recommendations which summarizes the achievements and suggests future improvements.

Chapter 2: Background Study and Literature Review

2.1 Background Study

In today's digital era, employment has become one of the most pressing concerns for individuals across all fields. While traditional job portals have certainly improved the visibility of vacancies, they still fall short in addressing one key challenge: personalization. Job seekers often find themselves overwhelmed by the sheer volume of listings, many of which do not match their skills, qualifications, or interests. This not only leads to wasted time and effort but can also make the process of finding meaningful work frustrating and inefficient.

To overcome these limitations, the concept of a Job Recommendation System (JRS) has emerged as a valuable solution. A JRS is designed to narrow down job options and suggest opportunities that are highly relevant to the individual. Instead of simply displaying every available vacancy, it intelligently analyzes user information and available job data to deliver more accurate results. These systems often leverage machine learning techniques, allowing them to learn from user interactions, preferences, and feedback to continuously refine their recommendations.

There are two common strategies applied in such systems. Content-based filtering focuses on analyzing user profiles and comparing them with job descriptions to identify the best matches. In contrast, collaborative filtering relies on the behavior and preferences of similar users to generate suggestions, often uncovering opportunities that the individual might not have considered on their own. Both methods contribute in different ways to improving the job search experience.

In our project, we adopted a content-based approach to build the recommendation system. We applied TF-IDF (Term Frequency–Inverse Document Frequency) to give importance to significant terms in both job descriptions and user profiles. To determine the similarity between them, we used cosine similarity, which measures how closely the two sets of information align. By combining these techniques, our system is able to provide more accurate and meaningful recommendations.

2.2 Literature Review

Recommendation systems play a crucial role in guiding users toward items that best match their interests, preferences, and requirements. These systems analyze patterns in user behavior, profile attributes, and item characteristics to deliver meaningful suggestions. Among the different approaches, Content-Based Filtering (CBF) is one of the most widely applied techniques. In CBF, recommendations are generated by comparing the attributes of items (such as descriptions, tags, or metadata) with a user's profile, which is often constructed from their past interactions or explicitly stated preferences. This approach is particularly advantageous when historical interaction data is sparse, such as in situations involving new users or newly added items, where collaborative filtering may struggle due to the "cold start" problem [1].

To improve the precision of such recommendations, text preprocessing is a critical first step. Preprocessing involves cleaning the textual data by removing noise, eliminating common but less informative words (stopwords), and reducing words to their root form through stemming or lemmatization. These steps ensure that the model focuses on meaningful features instead of redundant or irrelevant tokens. Once the text is prepared, techniques like Term Frequency–Inverse Document Frequency (TF-IDF**)** are applied to assign weights to words, highlighting terms that are more distinctive in representing documents or item descriptions. After vectorizing both user profiles and item descriptions, Cosine Similarity is typically employed to measure how closely a particular job description, product detail, or educational resource aligns with the user's interests [1].

Despite its effectiveness, TF-IDF has inherent limitations. It treats words as independent tokens and does not capture the deeper semantic relationships between them. For instance, words like "job" and "employment" may be used interchangeably in descriptions, but a traditional TF-IDF model would not recognize their similarity. To address these shortcomings, modern research in recommendation systems has introduced semantic embeddings, leveraging models such as Word2Vec and FastText [2].

Word2Vec learns word representations by analyzing how words are used within similar contexts, thereby capturing semantic relationships like synonyms or analogies. This means that even if a user profile contains the word "developer," the system can still recognize a strong connection to items described with the term "programmer." FastText extends this

concept further by incorporating subword information. Instead of treating each word as an atomic unit, it breaks words into smaller character-based n-grams, enabling the system to handle rare terms, domain-specific jargon, and even misspellings more effectively [3]. This subword-level analysis makes FastText especially useful in domains where textual descriptions are lengthy and diverse, such as e-learning platforms, product catalogs, or job postings [4].

By mapping words into continuous vector spaces, embedding-based models enrich recommendation systems with the ability to understand language at a deeper level. They allow systems to move beyond strict keyword matching, capturing subtle nuances, contextual meanings, and semantic relationships. This ensures that even if different terminology is used in user profiles and item descriptions, the system can still identify meaningful matches, ultimately producing more accurate and user-relevant recommendations [4].

Building on this foundation, recent advancements in deep learning have integrated word embeddings into neural architectures designed specifically for recommendation tasks. Unlike traditional content-based approaches that separately analyze user and item data, these neural models jointly learn both user preferences and item representations within a single framework. This joint learning enables the system to capture not only semantic understanding of the content but also complex interaction patterns between users and items. Such hybrid architectures have demonstrated the ability to generate highly personalized and adaptive recommendations by continuously refining their understanding of user interests as new interactions occur [5].

In summary, the evolution of content-based filtering in recommendation systems illustrates a clear transition—from keyword-based methods like TF-IDF to embedding-driven and deep learning-enhanced approaches. While classical models provide a solid baseline for recommendation accuracy, the integration of semantic embeddings and neural networks significantly enhances the system's ability to handle rich, diverse, and unstructured text data. This progression has expanded the applicability of recommendation systems across a wide range of domains, making them more intelligent, flexible, and capable of delivering personalized experiences that align closely with user needs.

Chapter 3: System Analysis

3.1 System Analysis

For the Job recommendation system this step was important to ensure the final product would be user friendly. This includes specifying the users, data flow, system functionality, and technical architecture and understanding the databases and relationship among various entities.

3.1.1 Requirement Analysis

Requirement analysis focuses on identifying the needs of users (job seekers) to ensure the system provides efficient experience. The process involved gathering information through domain research and analysis of existing systems to define technical and functional goals. The requirement analysis can be of two different types:

i. Functional Requirements

For Job recommendation system, the main functional requirements are:

- User Registration and Login - Secure registration with strong password policies and email verification for account activation
- Profile Management - Users can update personal information and there is reverification when user changes their email address.
- Job Search and View – Users can look for jobs by category, location or keyword and the job details are shown.
- Job Application - Users can look for jobs by keyword, location or category and the job details are shown.
- Application Tracking - Users can view and manage submitted job applications
- Recommendation System - Suggests similar or relevant jobs on the basis of user behavior and preferences.

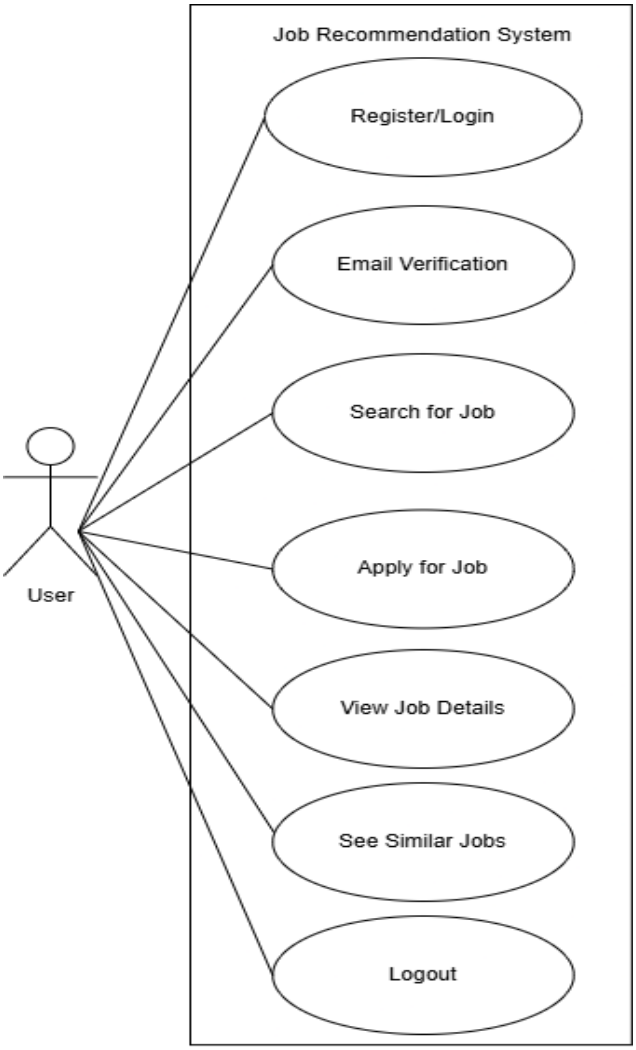

Figure 3.1: Use Case Diagram for user

The process begins with the user registering or logging into the system, followed by email verification to activate the account. Once the account for user is authenticated they can search for their relevant job and apply for that job. User can also view the detailed Information regarding the jobs and can see other similar jobs. Finally, the user can log out of the system once they have completed their activities.

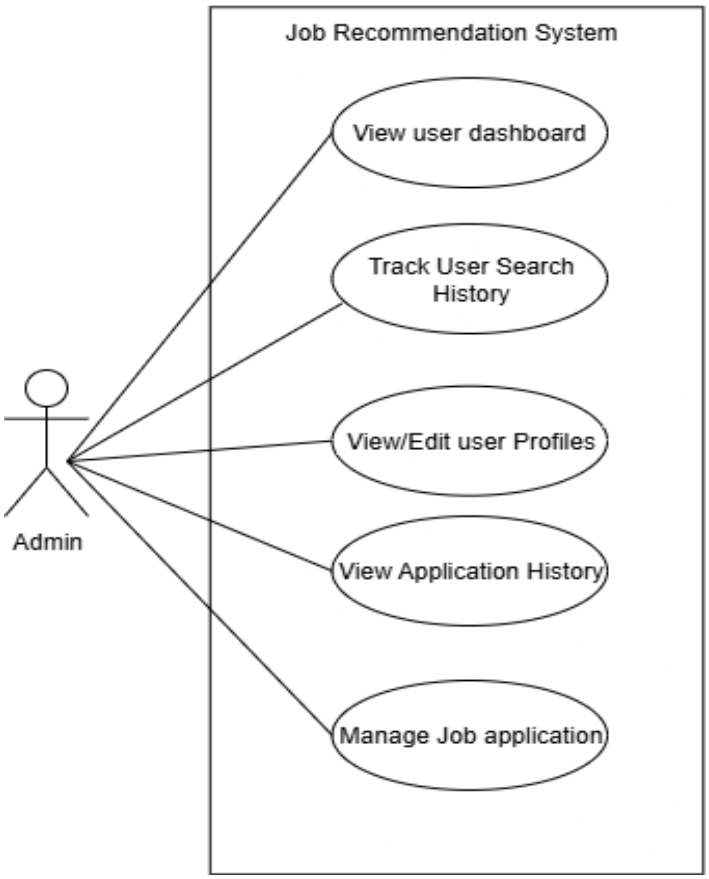

Figure 3.2: Use Case Diagram for Admin

The admin can view the dashboard, track users job search history, and view or edit their profile information. They can also access their application history and manage personalized job recommendations based on their activity and preferences.

ii. Non-Functional Requirements

For Job recommendation system, the non-functional requirements are:

- Security – User data and passwords are safely stored, and email verification ensures only real users get access.
- Usability – The system has a clean, user-friendly interface that works smoothly on all devices.
- Responsive – Pages load quickly, and user actions are handled without delay.
- Reliability – The system is stable and works without crashing or losing user data.
- Maintainability – It's built in a way that can handle more users and job listings in the future.

3.1.2 Feasibility Analysis

i. Technical Feasibility

The project complies with current technology, including hardware and software, and is technically feasible. The project is a web-based system that is made using Django with Python for the backend, JavaScript, html and CSS3, Bootstrap5 for the frontend framework, and dB SQLite for the database. The project also uses Machine Learning with TF-IDF Vectorization, Cosine Similarity along with SMTP (Gmail integration).

ii. Operational feasibility

The job recommendation system is operationally feasible as it provides easy user interface to create user profile, upload resume, search for jobs and receive job recommendation on the basis of search with less effort. Administrators are able to keep an eye on system performance and effectively manage job postings. Additionally, the integration of email notifications for account confirmation enables smooth communication between the system and users. Overall, the system is expected to operate effectively within the existing organizational processes, which enhances productivity and user satisfaction.

iii. Economic feasibility

This project is budget-friendly because it uses free tools like Django and SQLite, so there's no need to spend money on expensive software. It can run on a basic server, which keeps hosting costs low. Since most of the work is done during development without needing any kind of special equipment, it's affordable to build and maintain. Overall, it's a smart and cost-effective solution for helping people find jobs without a big financial investment.

iii. Schedule feasibility

The project was completed within the given timeframe by taking into consideration of all required tasks, such as planning, implementation, programming, and testing. With a well plan timeline, the project was divided into manageable parts which ensured the progress was made efficiently and completed within the given schedule. The following Gantt chart shows the timetable set for the development of the system.

Table 3.1: Table for Activity Schedule

Task	Start Date	End Date	Duration (Days)
Requirement Analysis & Planning	5/5/2025	5/12/2025	7
Learn Basics and set up Project	5/13/2025	5/26/2025	13
Build user auth and UI templates	5/27/2025	6/09/2025	13
Build Job model and forms	6/10/2025	6/25/2025	15
Build Recommendation Script	6/16/2025	7/13/2025	17
Job Search & Filtering Module	7/14/2025	7/26/2025	12
Email Notification Setup	7/27/2025	8/7/2025	11
Testing, Parsing and Debugging	5/26/2025	8/8/2025	74
Documentation	5/6/2025	9/2/2025	113

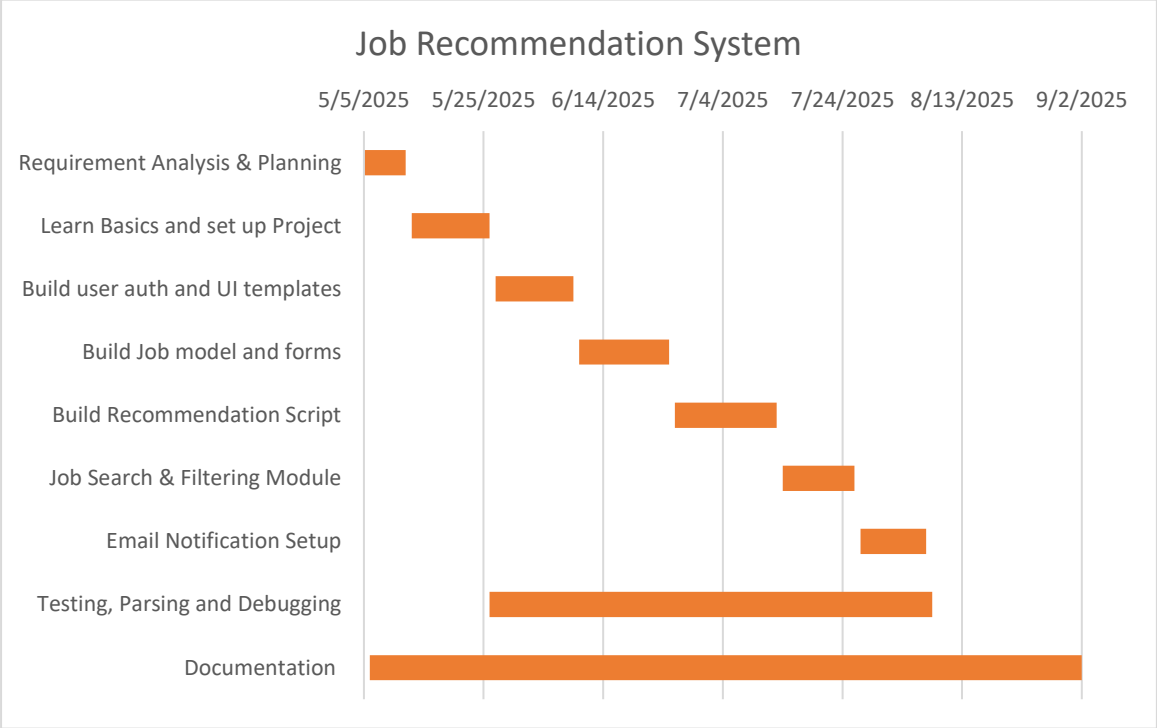

Figure 3.3: Gantt Chart

3.1.3 Object Modeling using Class Diagram

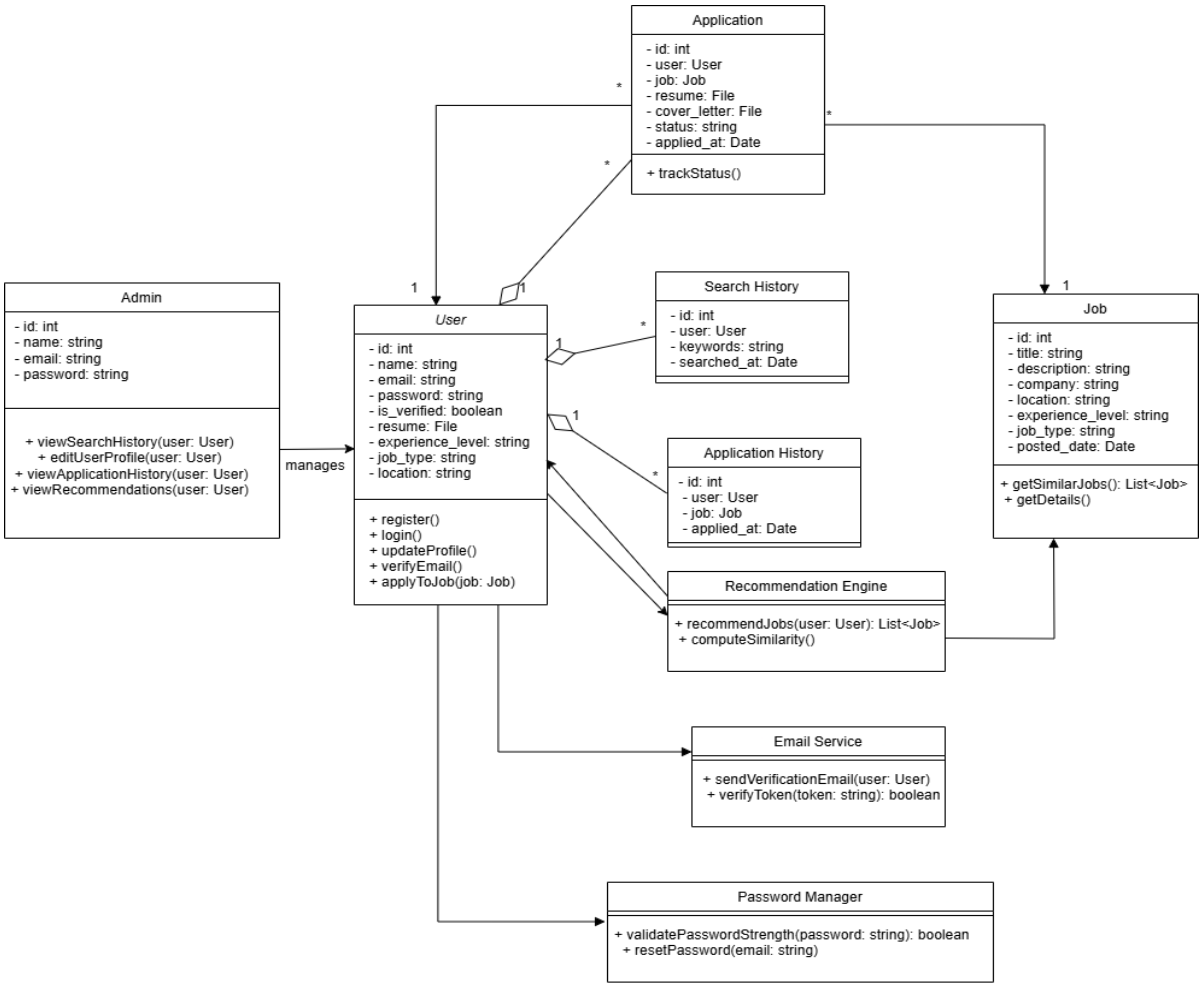

Figure 3.4: Class Diagram for Job recommendation system

This class diagram shows different classes of a Job Recommendation System and the relationship between them. At the center is the user, who can search and apply for jobs. Their activity is tracked through Search History and Application History, helping the system understand their interests. Based on this, the Recommendation Engine suggests jobs that match their preferences. The admin manages the system, while Job and Application handle job listings and applications. Supporting services like Email Service send notifications, and the Password Manager helps keep accounts secure. Altogether, these parts make the system helpful, personalized, and safe for users.

3.1.4 Dynamic modelling using sequence diagram

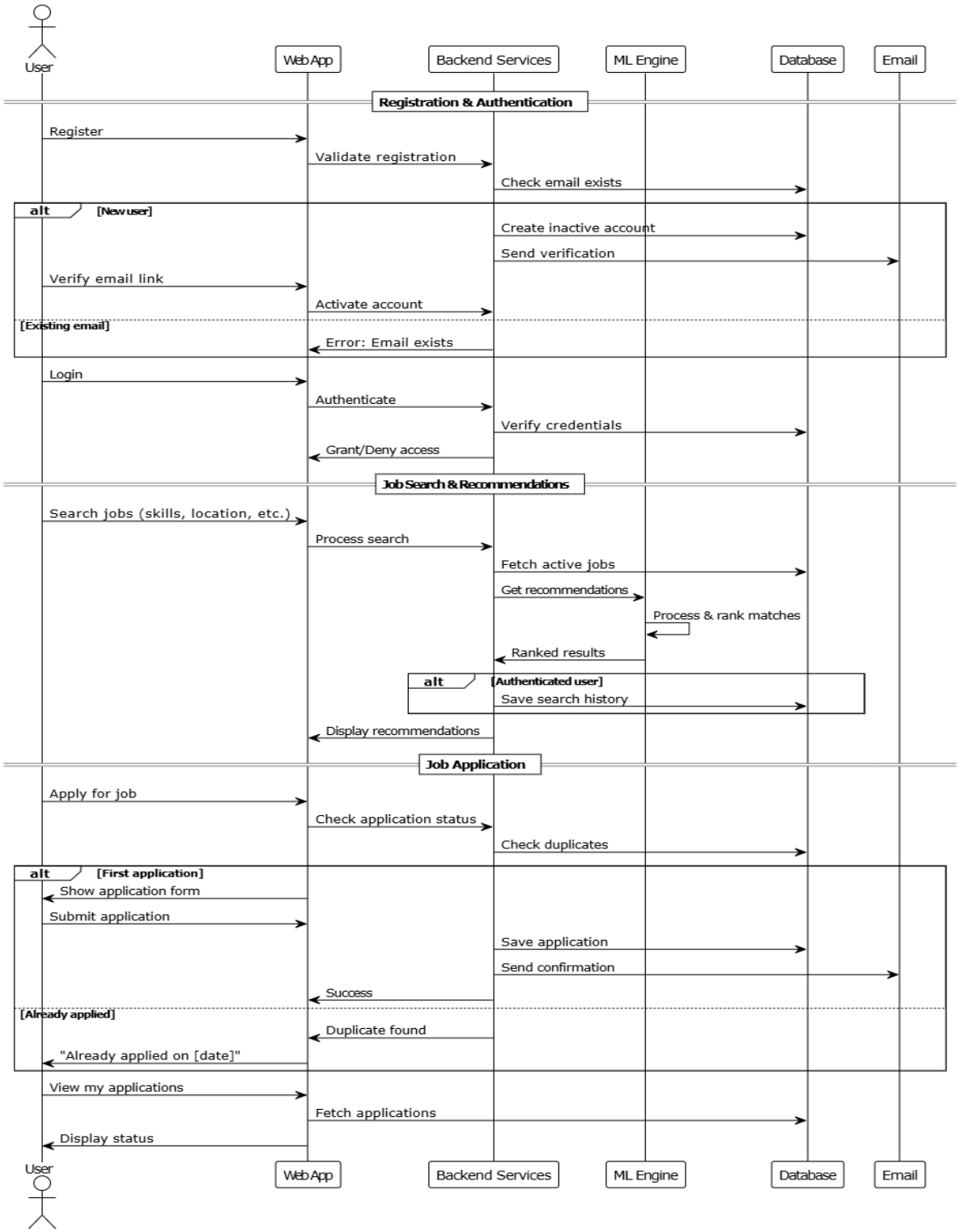

Figure 3.5: Sequence Diagram for Job recommendation system

This sequence diagram graphically shows the flow of processes that occurs between the system and its components. The process begins with a user registering by providing their name, email, and password and upon successful verification it activates the account. Once registered, the user can view job details and apply for a job by submitting a resume and cover letter. After applying, the system retrieves the user's application history and will provide job based on their profile. The interaction involves multiple components like the User-Controller, Email-Service, and Recommendation-Engine.

3.1.5 Process modelling using activity diagram

The following activity diagram illustrates the user journey from registration to job application.

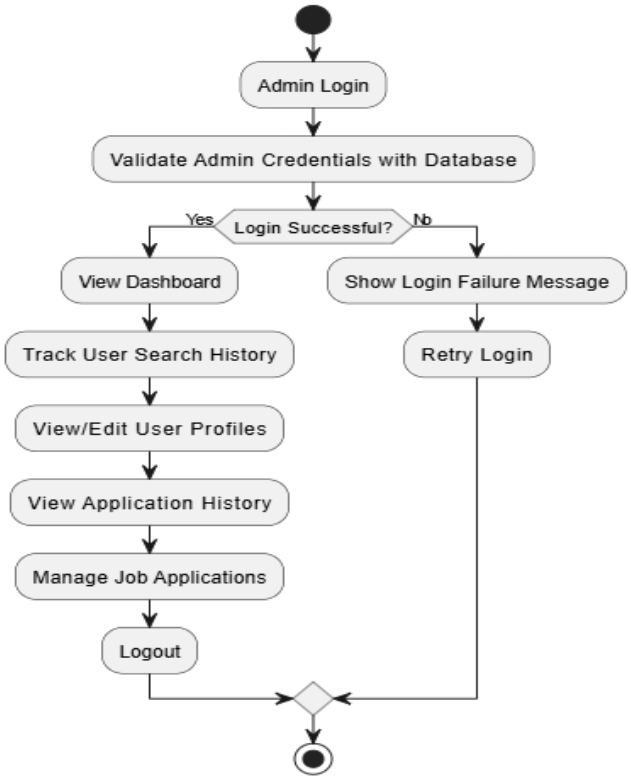

Figure 3.6: Activity Diagram for admin of Job recommendation system

The admin activity diagram shows the workflow for an administrator which manages the job recommendation system. It starts with the admin logging in to track users search history, viewing and editing user profile, checking application history, and managing personalized job recommendations.

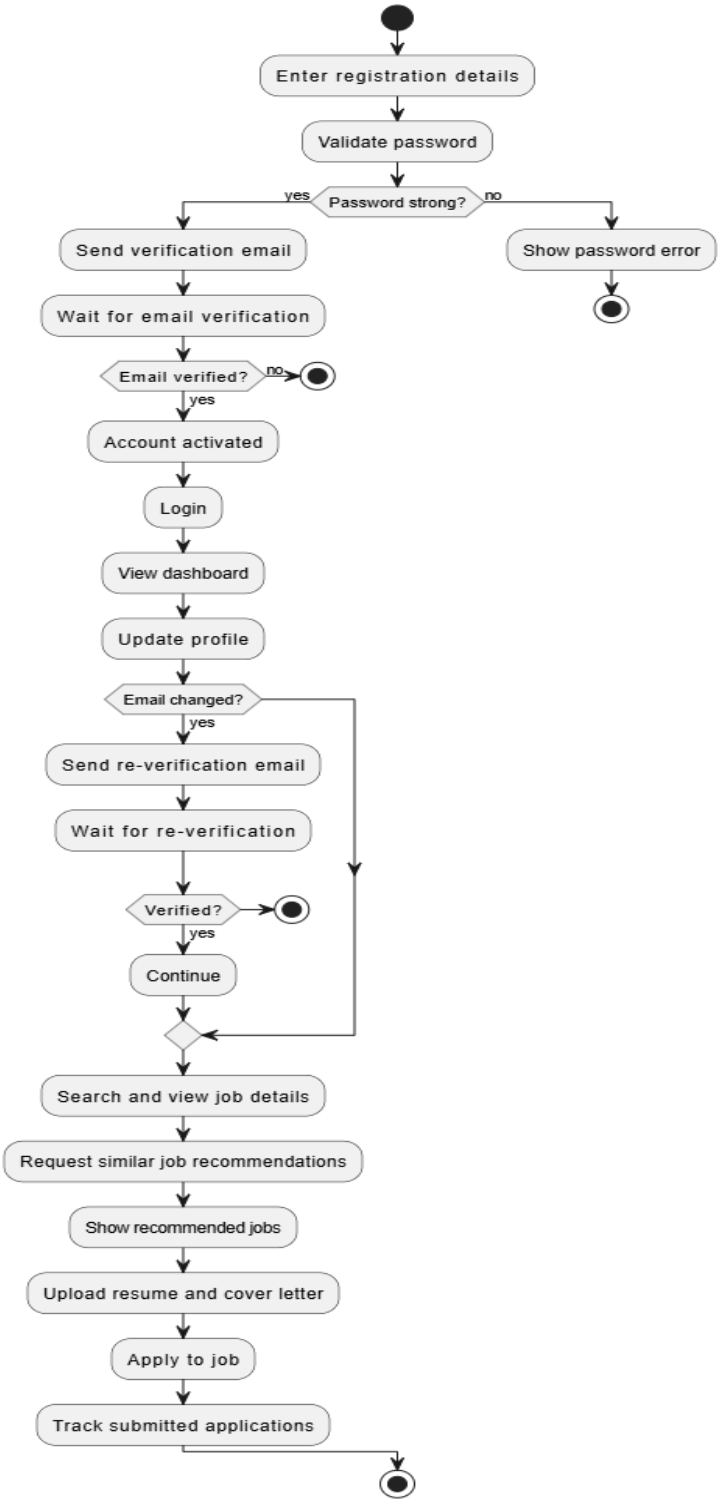

Figure 3.7: Activity Diagram for user of Job recommendation system

Chapter 4: System Design

4.1 Design

4.1.1 Refinement of Class, Object and Activity Diagram

As discussed in the analysis chapter, the system design also follows an object-oriented approach. Since, this is a simple project with not much complexity, the class diagrams, object diagrams and activity diagrams designed earlier are already refined enough to be applicable within the project.

4.1.2 Component Diagram

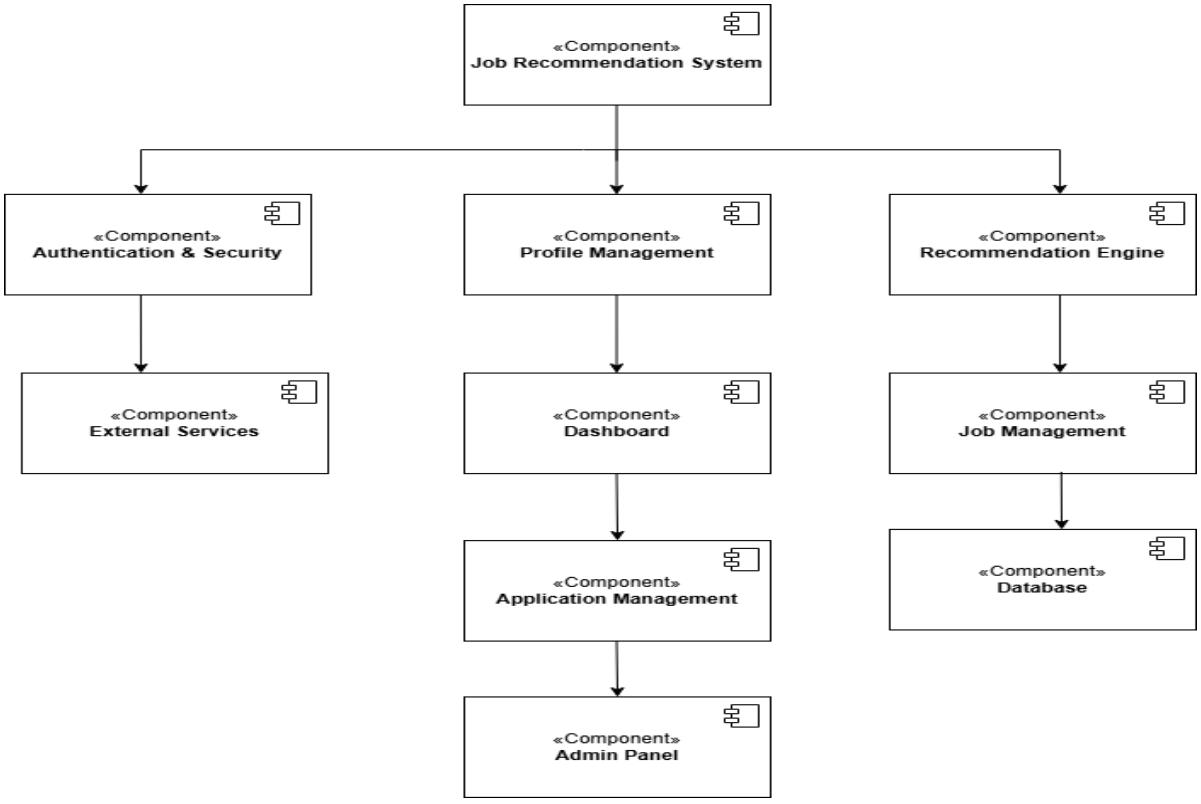

Figure 4.1: Component Diagram

This component diagram illustrates the diagrammatic representation of how different components are interlinked with each other to function as a whole recommendation system. There are various components like database, admin panel, dashboard in this system which shows how they interact with each other.

4.2 Algorithm Details

4.2.1 TF-IDF Algorithm

TF-IDF stands for Term Frequency–Inverse Document Frequency. The Job Recommendation System leverages the TF-IDF (Term Frequency-Inverse Document Frequency) technique to effectively analyze and match user inputs with relevant job postings [6].TF-IDF is a widely used text mining method that evaluates the importance of words in a document relative to a collection of documents [7]. In the context of system, each job posting—comprising the job title, description, and requirements is treated as a document [7]. The system preprocesses these texts and transforms them into numerical vectors that reflect the significance of each term within the entire job database [8].

By converting both user input (such as skills, experience, or preferences) and job postings into these TF-IDF vectors, the system can measure the textual similarity between them using cosine similarity [7]. This similarity score quantifies how closely a job aligns with the user's profile or query. Jobs with higher similarity scores are considered more relevant and are recommended to the user.

$$\mathrm{tf}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}},$$

$$\mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

Then,

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \cdot \mathrm{idf}(t,D)$$

Algorithm for TF-IDF Vectorization

- Text Combination: For each job posting, combine key textual fields (title, description, requirements) into a single string to represent the job comprehensively.
- Vectorization: Use the TF-IDF vectorizer to convert the cleaned texts into numerical vectors reflecting term importance across the dataset.
- Similarity Computation: Compute cosine similarity between the user's input vector and each job vector to measure their closeness in terms of content [9].
- Filtering and Ranking: Filter out jobs with very low similarity scores to improve recommendation quality and sort the remaining jobs by their similarity scores in descending order.
- Recommendation: Select and return the top N jobs as recommendations to the user.

4.2.2 Cosine Similarity

Cosine Similarity is a metric used to measure how similar two non-zero vectors are, regardless of their magnitude. It calculates the cosine of the angle between two vectors in a multi-dimensional space [9]. In the context of a Job Recommendation System, cosine similarity is used to compare the TF-IDF vector of the user's input (e.g., resume, skills) with the TF-IDF vectors of job descriptions [6]. A higher cosine similarity value (closer to 1) indicates that the texts are more similar, making it a suitable metric for ranking job recommendations based on textual relevance [9]. The cosine similarity between two vectors, A and B, is calculated using the following formula:

$$\text{cosine similarity } (A, B) = A \cdot B / ||A|| \cdot |B||$$

Here, $A \cdot B$ represents the dot product of the vectors, and $\|A\|$ and $\|B\|$ represent the Euclidean norms of the vectors.

Chapter 5: Implementation and Testing

5.1 Implementation

5.1.1 Tools Used

To guarantee efficiency, scalability, and maintainability, a variety of current techniques and technologies were used in the development of the Job Recommendation System. Below is a list of the primary tools and technologies used:

- Programming Language: Python 3.12
- Web Framework: Django 5.2.2
- Frontend Technologies: HTML5, CSS3, Bootstrap 5, JavaScript
- Database: SQLite (for development purposes)
- Machine Learning Libraries: scikit-learn, NLTK
- Email Integration: SMTP (Gmail)
- Version Control: Git and GitHub
- Development Environment: Visual Studio Code

These tools were chosen for their robustness, ease of use, and compatibility with the project requirements.

5.1.2 Implementation Details of Modules

The system was divided into several modules, each responsible for a specific functionality. Below is a detailed explanation of the key modules:

1. Authentication Module

- Description: This module handles user registration, login, logout, and email verification.
- Implementation: Django's built-in authentication system was extended to include email verification using token-based activation links. Password validation and secure session management were also implemented.

2. Job Recommendation Engine

- Description: This is the core module responsible for recommending jobs based on user input.
- Implementation: The system uses TF-IDF (Term Frequency-Inverse Document Frequency) for text vectorization and Cosine Similarity to calculate the relevance of jobs to the user's input. Pre-processing steps such as tokenization, stopword removal, and lowercasing were applied to improve accuracy.

3. Job Management Module

- Description: This module manages job postings, including their creation, storage, and retrieval.
- Implementation: A Job model was created in Django to store job details such as title, description, company, location, and requirements. The admin panel was customized to allow easy management of job postings.

4. User Dashboard Module

- Description: Provides users with a personalized dashboard to view their search history, manage applications, and edit their profiles.
- Implementation: The dashboard dynamically fetches data from the database, including recent searches and application statuses. Django templates and Bootstrap were used to create a responsive and user-friendly interface.

5. Contact Module

- Description: Allows users to send inquiries or feedback to the system administrators.
- Implementation: A simple form was created using Django Forms, and messages are stored in the database for review by administrators.

6. Search and Filter Module

- Description: Enables users to search for jobs based on keywords, experience level, job type, and location.
- Implementation: The search functionality was implemented using Django QuerySets, and filters were applied to narrow down results based on user preferences.

5.2 Testing

Testing was performed manually by the developer to ensure the system's functionality and reliability. The following steps were taken during the testing phase:

1. **Unit Testing:**

- Each module was tested individually to verify its functionality.
- For example, the authentication module was tested for successful registration, login, and email verification.

2. **Integration Testing:**

- Modules were tested together to ensure seamless interaction.
- For instance, the job recommendation engine was tested with the search and filter module to verify accurate results.

3. **User Interface Testing:**

- The frontend was tested for responsiveness and usability on different devices and screen sizes.
- Forms were tested for proper validation and error handling.

5.2.1 Test Case for Unit Testing

Table 5.1: Test Cases for Registration/Login

Test Case Id	Test Case Description	Steps to Perform	Expected Outcome	Actual Outcome	Status
UT-01	Test user registration functionality	1. Navigate to the registration page. 2. Fill in valid details. 3. Submit the form.	User is registered successfully and redirected to the login page.	User is registered successfully and redirected to the login page.	Pass
UT-02	Test login functionality	1. Navigate to the login page. 2. Enter valid credentials. 3. Submit the form.	User is logged in and redirected to the dashboard.	User is logged in and redirected to the dashboard.	Pass
UT-03	Test invalid login attempt	1. Navigate to the login page. 2. Enter invalid credentials. 3. Submit the form.	Error message is displayed: "Invalid username or password."	Error message is displayed: "Invalid username or password."	Pass
UT-04	Test email verification	1. Register a new account. 2. Check the email inbox. 3. Click the verification link.	Account is activated, and user can log in.	Account is activated, and user can log in.	Pass

Table 5.2: Test Cases for Job Recommendation Model

Test Case Id	Test Case Description	Steps to Perform	Expected Outcome	Actual Outcome	Status
UT-05	Test job recommendation accuracy	1. Enter a job description in the search form. 2. Submit the form.	Relevant job recommendations are displayed.	Relevant job recommendations are displayed.	Pass
UT-06	Test filtering by location	1. Enter a job description. 2. Select a location filter. 3. Submit the form.	Jobs matching the location filter are displayed.	Jobs matching the location filter are displayed.	Pass
UT-07	Test filtering by experience level	1. Enter a job description. 2. Select an experience level filter. 3. Submit the form.	Jobs matching the experience level are displayed.	Jobs matching the experience level are displayed.	Pass
UT-08	Test empty search input	1. Leave the search form empty. 2. Submit the form.	Error message is displayed: "Please enter a job description."	Error message is displayed: "Please enter a job description."	Pass

5.2.2 Test Case for Integration Testing

Table 5.3: Test Cases for Integration Testing

Test Case Id	Test Case Description	Steps to Perform	Expected Outcome	Actual Outcome	Status
IT-01	Test integration of registration and login	1. Register a new account. 2. Log in with the registered credentials.	User is able to log in after registration.	User is able to log in after registration.	Pass
IT-02	Test integration of job search and recommendation engine	1. Enter a job description. 2. Submit the form. 3. View the recommendations.	Job recommendations are displayed based on the input.	Job recommendations are displayed based on the input.	Pass
IT-03	Test integration of dashboard and search history	1. Perform a job search. 2. Navigate to the dashboard. 3. View search history.	Search history displays the recent job searches.	Search history displays the recent job searches.	Pass

5.2.3 Test Case for System Testing

Table 5.4: Test Cases for Responsiveness of System

Test Case Id	Test Case Description	Steps to Perform	Expected Outcome	Actual Outcome	Status
ST-01	Test responsiveness on desktop	1. Open the application on a desktop browser. 2. Navigate through all pages.	Pages are displayed correctly without layout issues.	Pages are displayed correctly without layout issues.	Pass
ST-02	Test responsiveness on tablet	1. Open the application on a tablet browser. 2. Navigate through all pages.	Pages are displayed correctly without layout issues.	Pages are displayed correctly without layout issues.	Pass
ST-03	Test responsiveness on mobile	1. Open the application on a mobile browser. 2. Navigate through all pages.	Pages are displayed correctly without layout issues.	Pages are displayed correctly without layout issues.	Pass

Table 5.5: Test Cases for Usability Testing

Test Case Id	Test Case Description	Steps to Perform	Expected Outcome	Actual Outcome	Status
ST-04	Test ease of navigation	1. Navigate through the application. 2. Access all major features.	Users can easily navigate and access features.	Users can easily navigate and access features.	Pass
ST-05	Test form validation	1. Submit forms with invalid or incomplete data. 2. Observe error messages.	Appropriate error messages are displayed.	Appropriate error messages are displayed.	Pass
ST-06	Test user feedback	1. Perform actions like login, search, and apply. 2. Observe feedback messages.	Feedback messages are displayed for all actions.	Feedback messages are displayed for all actions.	Pass

5.3 Result Analysis

The Job Recommendation System was found to be effective in recommending relevant jobs based on user input. Key observations include:

- User Experience: The interface was intuitive and easy to navigate, with users appreciating the clean design and responsiveness.
- Performance: The recommendation engine processed queries within 1-2 seconds, even with a large dataset.
- Feedback: Users provided positive feedback on the system's ability to filter jobs by location, experience level, and job type.

Chapter 6: Conclusion and Future Recommendations

6.1 Conclusion

The Job Recommendation System successfully achieved its objectives of providing personalized job recommendations to users. The system utilized machine learning techniques to analyze user input and match it with relevant job postings. Key features such as authentication, job search, and user dashboards were implemented effectively, resulting in a functional and user-friendly application. This project demonstrated the potential of combining machine learning with web development to solve real-world problems.

6.2 Future Recommendations

While the system is functional, there are several areas for improvement and expansion:

1. Integration of Advanced AI Models: Implement deep learning models such as BERT for better natural language understanding and more accurate recommendations.

2. Improved User Interface: Enhance the UI with modern design frameworks and animations to improve user engagement.

3. Expansion of Job Categories: Include more diverse job categories and industries to cater to a broader audience.

4. Mobile Application: Develop a mobile app to make the system accessible on smartphones and tablets.

5. Real-Time Notifications: Add features to notify users of new job postings or application updates via email or push notifications.

6. Analytics Dashboard for Admins: Provide administrators with insights into user activity, popular job searches, and system performance.

7. Multi-Language Support: Enable the system to support multiple languages to reach a global audience.

References

[1] A. Oktaviantoro and D. Kusuma, "Content-based filtering using cosine similarity algorithm for alternative selection on training programs," *Journal of Soft Computing Exploration*, vol. 4, p. 204–212, 2023.

[2] R. Huang, "Improved content recommendation algorithm integrating semantic information," *Journal of Big Data*, vol. 10, no. 84, 2023.

[3] P. Bojanowski, E. Grave, A. Joulin and T. Mikolov, "Enriching Word Vectors with Subword Information," *Transactions of the Association for Computational Linguistics*, vol. 5, no. 135–146, 2017.

[4] S. Zhang, L. Yao, A. Sun and Y. Tay, "Deep Learning Based Recommender System: A Survey and New Perspectives," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1-38, 2019.

[5] H. Wang, N. Wang and D.-Y. Yeung, "Collaborative Deep Learning for Recommender Systems," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.

[6] M. R. C. B. A. Elsafty, "Document-based Recommender System for Job Postings using Dense Representations," in *Proc. NAACL-HLT 2018 (Industry Papers)*, 2018.

[7] P. R. H. S. C. D. Manning, Introduction to Information Retrieval, Cambridge: Cambridge University Press, 2008.

[8] s.-l. Developers, "TfidfVectorizer — scikit-learn documentation," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html. [Accessed 27 August 2025].

[9] E. Garcia, "Cosine Similarity Tutorial," [Online]. Available: http://itlab.uta.edu/educators/online-courses/9-itlab-tutorials/37-cosine-similarity.

Appendices

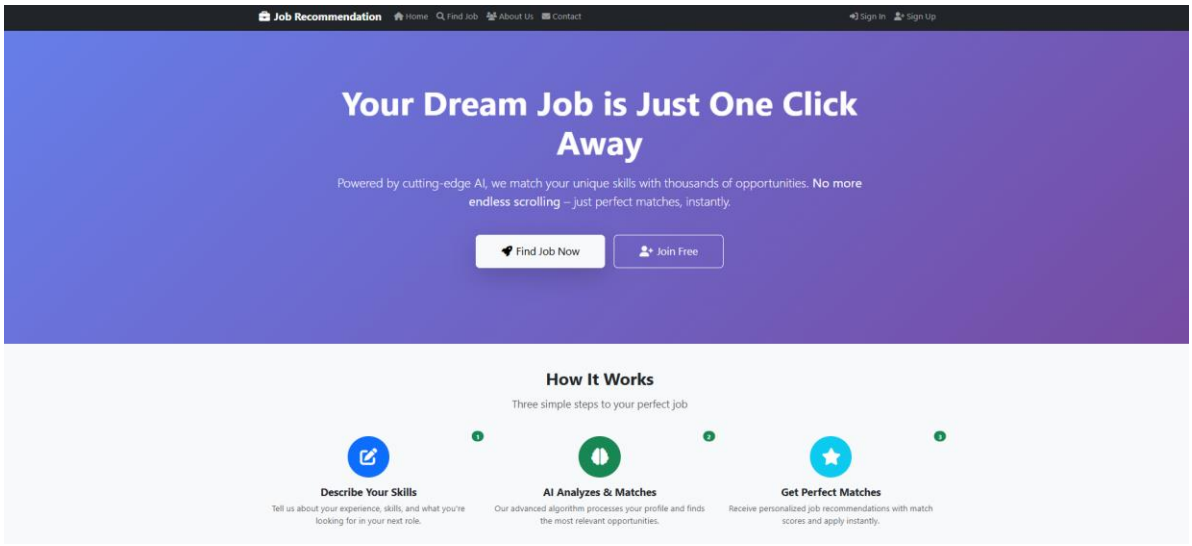

Figure 1: Home Page

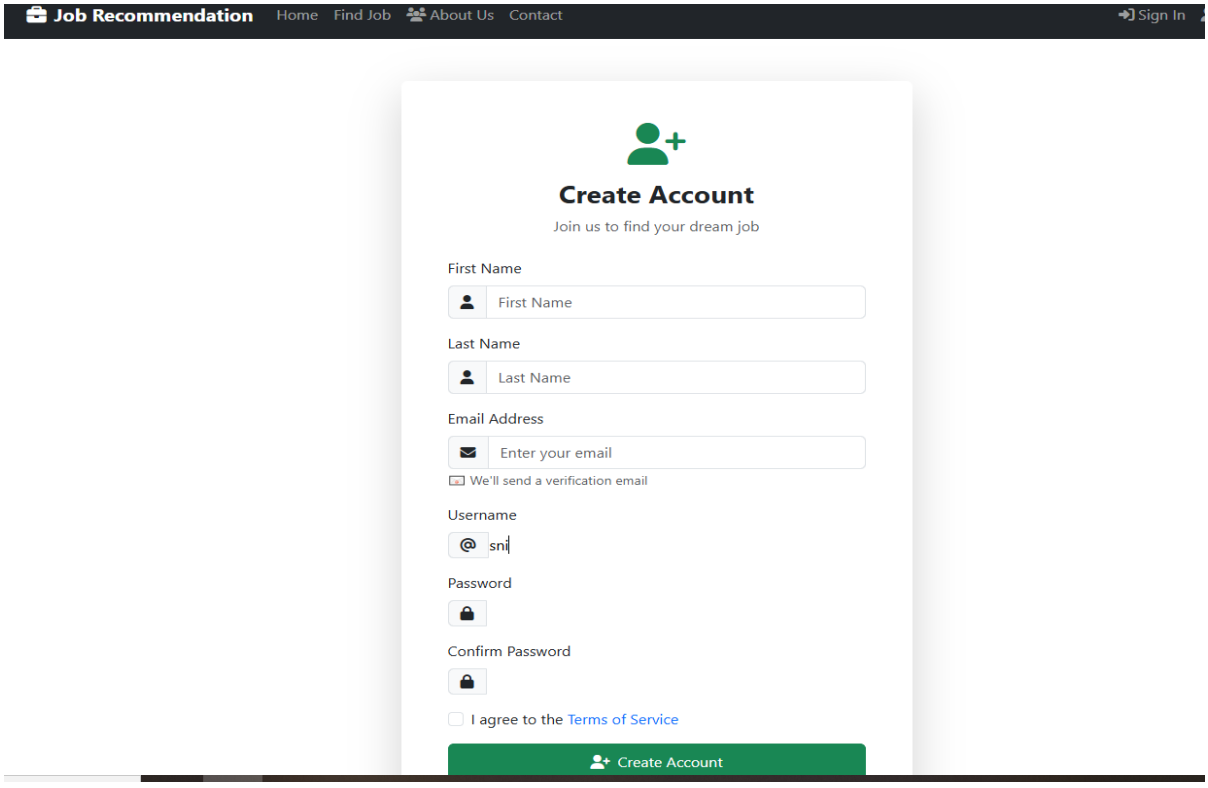

Figure 2: Registration Page

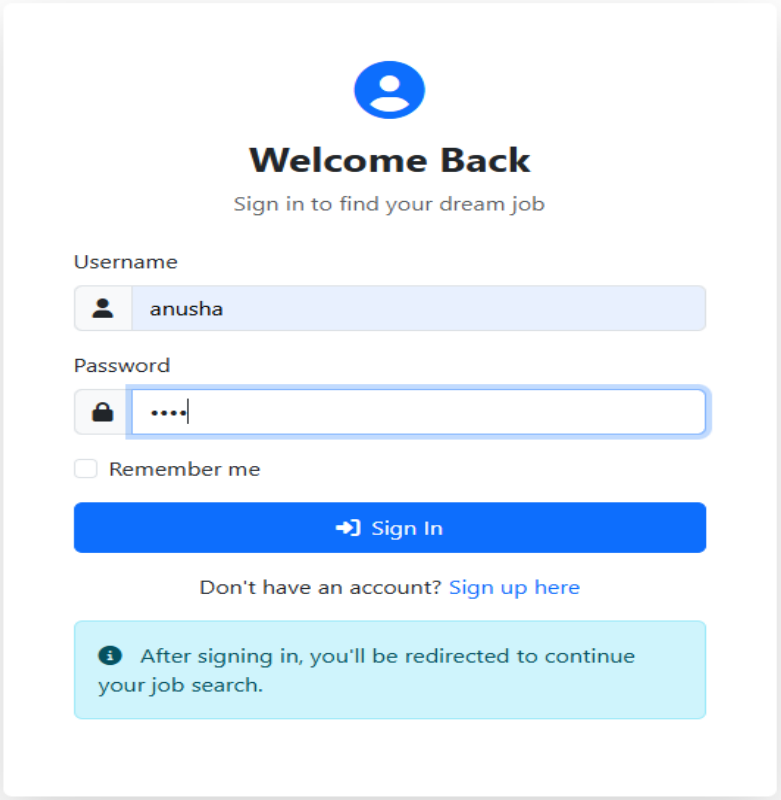

Figure 3: Login Page

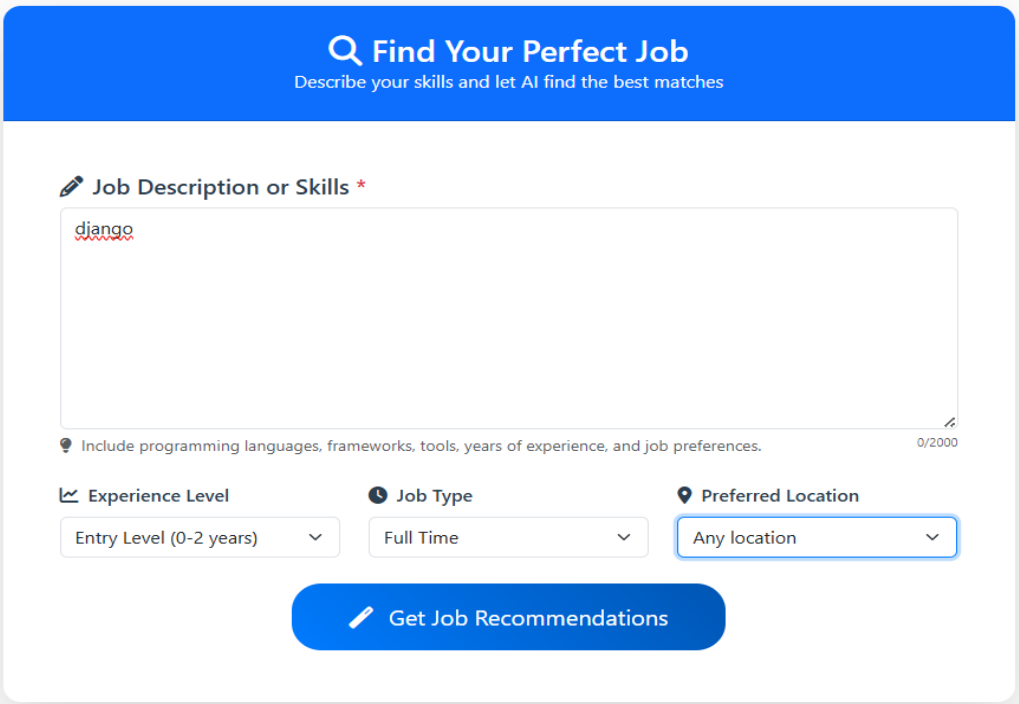

Figure 4: Search for Job

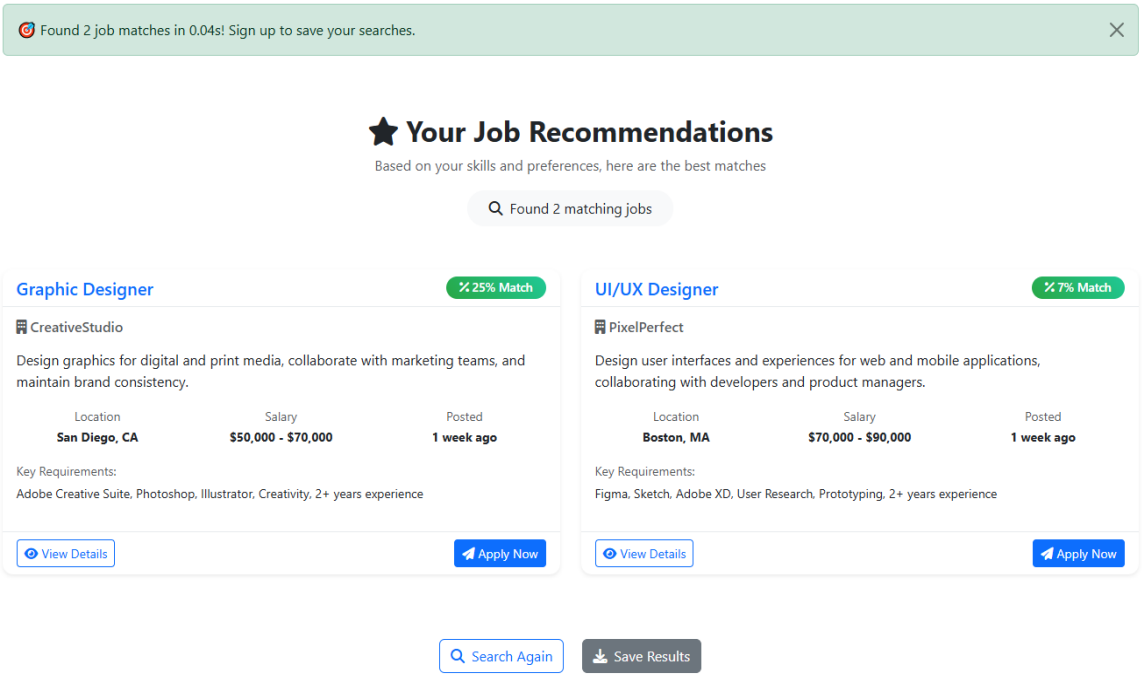

Figure 5: Job Recommendation Page

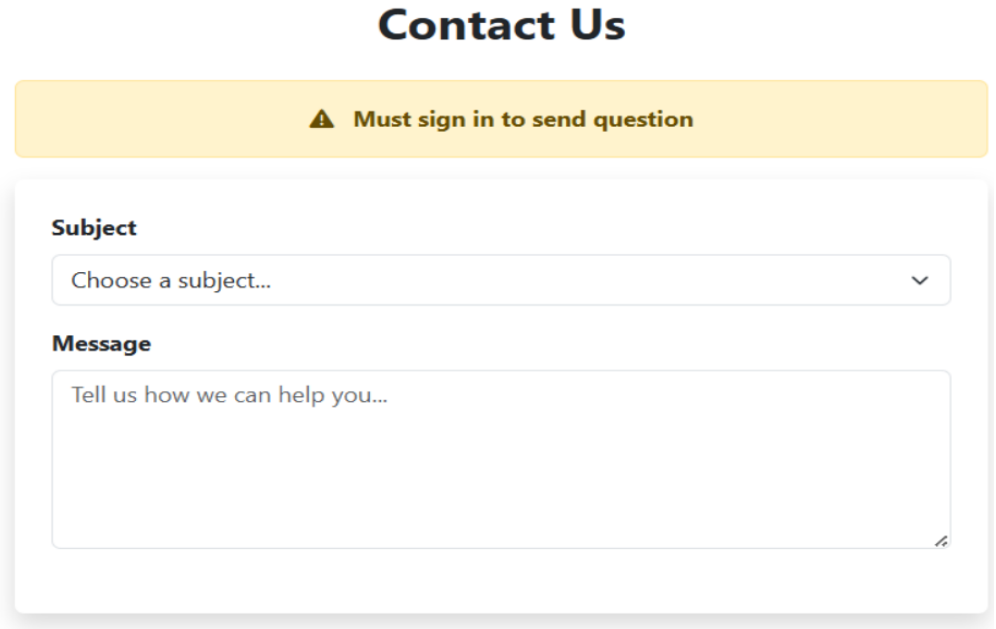

Figure 6: Contact us Page

Algorithm

```
# recommendations/ml_engine.py


import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np


class JobRecommendationEngine:
    def __init__(self):
        self.vectorizer = TfidfVectorizer(
            stop_words='english',
            max_features=5000,
            ngram_range=(1, 2),
            lowercase=True
        )


    def preprocess_text(self, text):
        """Clean and preprocess text"""
        if not text:
            return ""


        # Convert to lowercase and remove special characters
        text = re.sub(r'[^a-zA-Z0-9\s+]', ' ', str(text).lower())
        # Remove extra whitespace
```

```
    text = ' '.join(text.split())
    return text

def get_recommendations(self, user_input, jobs, top_n=5, filters=None):
    """
    Get job recommendations based on user input
    """
    if not jobs.exists():
        print("No jobs available in the database.")
        return []

    # Preprocess user input
    user_input_clean = self.preprocess_text(user_input)
    print(f"Preprocessed User Input: {user_input_clean}")

    # Combine job description and requirements for better matching
    job_texts = []
    job_objects = []

    for job in jobs:
        combined_text = f"{job.title} {job.description} {job.requirements}"
        job_texts.append(self.preprocess_text(combined_text))
        job_objects.append(job)

    if not job_texts:
```

```
            print("No job texts available after preprocessing.")
            return []

        try:
            # Create TF-IDF matrix
            all_texts = [user_input_clean] + job_texts
            tfidf_matrix = self.vectorizer.fit_transform(all_texts)
            print(f"TF-IDF Matrix Shape: {tfidf_matrix.shape}")
            print(f"TF-IDF Matrix (Dense Format):\n{tfidf_matrix.toarray()}")

            # Calculate cosine similarity
            user_vector = tfidf_matrix[0:1]
            job_vectors = tfidf_matrix[1:]
            similarities = cosine_similarity(user_vector, job_vectors).flatten()

            # Get top recommendations
            job_similarity_pairs = list(zip(job_objects, similarities))

            # Filter out jobs with very low similarity (< 0.01)
            job_similarity_pairs = [(job, sim) for job, sim in job_similarity_pairs if sim >
0.01]

            # Sort by similarity score
            job_similarity_pairs.sort(key=lambda x: x[1], reverse=True)

            # Return top N recommendations
```

```
    return job_similarity_pairs[:top_n]


except Exception as e:
    print(f"Error in recommendation engine: {e}")
    return []
```