URCAREER: AN AI ASSISTED CAREER GUIDANCE WEB APPLICATION

A PROJECT REPORT

Submitted by

BHARATHEESHWAR S (2116210701041)
GOKULA KRISHNA H (2116210701062)

in partial fulfillment for the course

GE19612 - PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND ENTREPRENEURSHIP

for the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE
RAJALAKSHMI NAGAR
THANDALAM
CHENNAI – 602 105

MAY 2024

RAJALAKSHMI ENGINEERING COLLEGE

CHENNAI - 602105

BONAFIDE CERTIFICATE

Certified that this Thesis titled **"URCAREER: AN AI ASSISTED CAREER GUIDANCE WEB APPLICATION**" is the bonafide work of "**BHARATHEESHWAR S (2116210701041) , GOKULA KRISHNA H (2116210701062)"** who carried out the project work for the course GE19612-Professional Readiness for Innovation, Employability and Entrepreneurship under my supervision.

Dr.T.Kumaragurubaran.,M.Tech.,Ph.D
PROJECT COORDINATOR
Assistant Professor (SG)
Department of
Computer Science and Engineering
Rajalakshmi Engineering College
Rajalakshmi Nagar
Thandalam
Chennai - 602105

Submitted to Project and Viva Voce Examination for the course GE19612-Professional Readiness for Innovation, Employability and Entrepreneurship held on _____.

ABSTRACT

The "urcareer" project aims to revolutionize career guidance and job recommendation by leveraging advanced technology. It introduces a comprehensive web application that assists individuals in finding suitable career paths and job opportunities based on their skills and experiences. Through the seamless integration of cutting-edge technologies such as natural language processing (NLP), machine learning (ML), and cloud-based APIs, "urcareer" offers an innovative solution to the challenges of traditional career counseling methods.The core functionality of the application begins with users uploading their resumes, which are then analyzed using NLP techniques to extract relevant keywords and skills. These keywords are assigned weightage based on their importance, providing a more accurate representation of the user's expertise. Subsequently, the system generates personalized assessment questions using OpenAI's language model, tailored to evaluate the user's proficiency in various domains.Upon completing the assessment, users receive detailed feedback on their performance, including scores and recommendations for further career development. Leveraging MongoDB for efficient data storage and retrieval, "urcareer" ensures seamless access to user profiles and assessment results. The application also incorporates responsive web design principles, allowing users to access it across different devices.With its user-centric approach and advanced technological capabilities, "urcareer" offers a transformative solution to career guidance, empowering individuals to make informed decisions about their professional journey.

ACKNOWLEDGEMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavour to put forth this report. Our sincere thanks to our Chairman **Thiru. S.Meganathan, B.E., F.I.E.,** our Vice Chairman **Mr. M.Abhay Shankar, B.E., M.S.,** and our respected Chairperson **Dr. (Mrs.) Thangam Meganathan, M.A., M.Phil., Ph.D.,** for providing us with the requisite infrastructure and sincere endeavouring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N.Murugesan, M.E., Ph.D.,** our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. P.Kumar, M.E., Ph.D.,** Professor and Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work. We express our gratitude to our project coordinator **Dr.T.Kumaragurubaran, M.Tech.,Ph.D.,** for his encouragement and guiding us throughout the project towards successful completion of this project and to our parents, friends, all faculty members and supporting staff for their direct and indirect involvement in successful completion of the project for their encouragement and support.

BHARATHEESHWAR S (2116210701041)

GOKULA KRISHNA H (2116210701062)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	ACRONYM
HTML	Hyper-Text Markup Language
CSS	Cascading Style Sheets
API	Application Programming Interface
URL	Uniform Resource Locator
UI	User Interface
GPT	Generative Pre-trained Transformers
AI	Artificial Intelligence
NLP	Natural Language Processing
ML	Machine Learning
DB	Database
IDE	Integrated Development

CHAPTER 1
INTRODUCTION

1.1 PROBLEM STATEMENT

The conventional approach to career guidance often falls short in providing tailored recommendations and personalized insights, leaving individuals feeling lost amidst the vast array of career options. Moreover, the job market is constantly evolving, making it challenging for individuals to navigate and identify suitable employment opportunities that align with their skills and aspirations. This lack of personalized guidance results in frustration, indecision, and underutilization of talent, leading to suboptimal career outcomes.Existing career counseling methods rely heavily on manual assessments and generic advice, which are time-consuming, subjective, and often outdated. Furthermore, the rapid advancements in technology and the emergence of new industries require a more dynamic and data-driven approach to career planning.

There is a pressing need for a modernized career guidance system that harnesses the power of technology to offer personalized recommendations and actionable insights tailored to individual skills, interests, and career goals. Such a system should leverage advanced techniques like natural language processing (NLP) and machine learning (ML) to analyze resumes, assess skill levels, and match users with relevant job opportunities in real-time. Additionally, integrating external APIs, such as OpenAI, can enhance the system's capabilities by generating customized assessments and providing valuable feedback to users.By addressing these challenges and leveraging technology to create a more efficient and effective career guidance solution, we can empower individuals to make informed decisions about their professional development, optimize their career trajectories, and ultimately contribute to a more productive and fulfilling workforce.

1.2. AIM AND OBJECTIVES:

Aim:

The aim of the "urcareer" project is to develop an innovative career guidance application that utilizes advanced technology to match individuals with suitable job opportunities, thereby fostering efficient employment and career development.

Objectives:

1. Personalized Career Guidance: Develop a system that analyzes user resumes using natural language processing (NLP) techniques to extract relevant skills and experiences, providing personalized career guidance tailored to individual strengths and aspirations.

2. Skill Assessment: Implement a module that generates customized assessment questions based on the user's skill set, leveraging machine learning algorithms and external APIs such as OpenAI to evaluate proficiency levels accurately.

3. Data-driven Recommendations: Utilize MongoDB to store and retrieve user data, enabling the system to generate data-driven job recommendations aligned with the user's skill set, experience, and career interests.

4. User Experience Optimization: Design an intuitive and user-friendly interface with responsive web design principles to ensure seamless accessibility across various devices, enhancing the user experience and usability of the application.

5. Continuous Improvement: Implement mechanisms for feedback collection and analysis to continuously improve the accuracy and relevance of career guidance and job recommendations provided by the system.

1.3 EXISTING SYSTEM

The traditional approach to career guidance primarily involves human counselors providing advice and recommendations based on standardized assessments and subjective evaluations. In this system, individuals seeking career guidance typically schedule appointments with career counselors or utilize online platforms to access generic career advice and resources.

Manual Assessments: Career counselors often rely on manual assessments, interviews, and questionnaires to gather information about an individual's skills, interests, and career aspirations. These assessments are time-consuming, subjective, and may not always accurately reflect an individual's capabilities or preferences.

Generic Recommendations: The advice provided by career counselors is often generic and may not take into account the specific skills, experiences, or aspirations of the individual. This one-size-fits-all approach can lead to recommendations that are not aligned with the individual's career goals or potential.

Limited Resources: Career guidance resources, both online and offline, are often limited in scope and may not cover the full range of career options available. Individuals may struggle to find relevant information or resources tailored to their specific needs and interests.

Lack of Real-time Feedback: The existing system lacks real-time feedback mechanisms to assess an individual's progress or provide immediate insights into potential career opportunities. This lack of feedback can hinder the individual's ability to make informed decisions about their career path.

1.4 PROPOSED SYSTEM

The proposed solution, "urcareer", is an innovative web application that revolutionizes career guidance and job recommendation by leveraging advanced technology and data-driven insights. The system offers a comprehensive platform where individuals can receive personalized career advice, skill assessments, and tailored job recommendations based on their unique strengths, experiences, and aspirations.

Automated Resume Analysis: "urcareer" utilizes natural language processing (NLP) techniques to analyze user resumes and extract relevant keywords, skills, and experiences. This automated process eliminates the need for manual assessments and ensures accurate and consistent data extraction.

Skill Assessment and Proficiency Evaluation: The system generates customized assessment questions using machine learning algorithms and external APIs such as OpenAI. These assessments evaluate the user's proficiency levels in various domains and provide detailed feedback on strengths and areas for improvement.

Personalized Job Recommendations: Leveraging MongoDB for efficient data storage and retrieval, "urcareer" matches users with relevant job opportunities based on their skill set, experience, and career interests. The system utilizes a data-driven approach to generate personalized job recommendations that align with the user's career goals and preferences.

User-Friendly Interface: The application features an intuitive and user-friendly interface with responsive web design principles, ensuring seamless accessibility across different devices. Users can easily navigate the platform, access their profile and assessment results, and explore job recommendations with ease.

CHAPTER 2
LITERATURE REVIEW

The field of career guidance and job recommendation has evolved significantly in recent years, driven by advancements in technology and a growing demand for personalized and data-driven solutions. Historically, career counseling has relied on traditional models such as trait-factor theory and Holland's theory of vocational choice. These models focus on matching individuals with careers based on personality traits, interests, and aptitudes. While these approaches have provided valuable insights into career decision-making processes, they often lack the flexibility and customization needed to address the diverse needs of today's workforce.Recent research has highlighted the potential of technology to enhance career guidance services. Tools such as online career assessments, virtual career fairs, and AI-powered chatbots have emerged as effective means of delivering personalized career advice and resources to individuals. For example, studies by Savickas (2019) and Sampson et al. (2020) emphasize the importance of integrating technology into career counseling to reach a wider audience and provide scalable solutions.The use of natural language processing (NLP) techniques for resume analysis has gained traction in the field of human resources and career guidance. Research by Gugnani et al. (2018) demonstrates the effectiveness of NLP in extracting key information from resumes, such as skills, experiences, and qualifications. By automating the resume screening process, NLP algorithms can save time and resources for both job seekers and employers.Machine learning (ML) algorithms offer promising opportunities for assessing individual skills and competencies in a data-driven manner. Studies by AlZoubi et al. (2021) and Sina et al. (2020) explore the use of ML models for predicting job performance and identifying skill gaps.

Personalized job recommendation systems leverage user data and machine learning algorithms to match individuals with relevant job opportunities. Research by Li et al. (2019) and Zhou et al. (2021) demonstrates the effectiveness of recommendation algorithms in improving job search outcomes and user satisfaction. These systems analyze factors such as skills, experience, and career preferences to provide tailored job recommendations that align with the individual's needs and interests.While technology has transformed the field of career guidance, several challenges remain. Issues such as data privacy, algorithmic bias, and the digital divide must be addressed to ensure equitable access to career resources and opportunities. Future research should focus on developing ethically sound and inclusive technologies that empower individuals from diverse backgrounds to make informed decisions about their careers.As technology plays an increasingly prominent role in career guidance, it is crucial to address ethical considerations to ensure fair and equitable outcomes for all users. Research by Coetzee and Ferreira (2019) highlights the importance of transparency, accountability, and privacy protection in the design and implementation of career guidance technologies. Ethical guidelines should be established to govern the collection, use, and storage of user data, particularly sensitive information such as resumes and personal preferences. Additionally, measures to mitigate algorithmic bias and ensure algorithmic fairness must be implemented to prevent discriminatory outcomes in job recommendations and skill assessments. Looking ahead, several emerging trends and technologies have the potential to shape the future of career guidance and job recommendation. Augmented reality (AR) and virtual reality (VR) applications offer immersive experiences for career exploration and skill development, allowing users to simulate real-world work environments and tasks. Blockchain technology holds promise for creating decentralized career credentialing systems, enabling individuals to securely store and share their qualifications and achievements.

CHAPTER 3
HARDWARE AND SOFTWARE REQUIREMENTS

- Desktop/Laptop Computers:
    - Processor: Intel Core i3 or equivalent
    - RAM: 4GB or higher
    - Storage: 128GB SSD or higher
    - Display: 15-inch monitor or larger
- Operating Systems:
    - Database Server: MongoDB 4.4 or later
    - Client Devices: Windows 10, macOS 10.15, iOS 12.0, Android 7.0, or later
- Development Tools:
    - Integrated Development Environment (IDE): Visual Studio Code, PyCharm
    - Version Control: Git, GitHub
    - Package Manager: pip (Python)
- Backend Technologies:
    - Programming Language: Python 3.8 or later
    - Web Framework: Flask 1.1.2 or later
    - Database Connectivity: pymongo 3.11.3 or later
    - Natural Language Processing: NLTK 3.5 or later, BeautifulSoup 4.9.3 or later
    - Machine Learning: OpenAI API
- Frontend Technologies:
    - HTML5, CSS3, JavaScript
    - Frontend Framework: Bootstrap 4.5.2 or later
- Database:
    - MongoDB 4.4 or later

CHAPTER 4
SYSTEM DESIGN

4.1 ARCHITECTURE DIAGRAM

An architecture diagram is a visual representation or blueprint that illustrates the high-level structure of a system, application, or a specific component within a system.

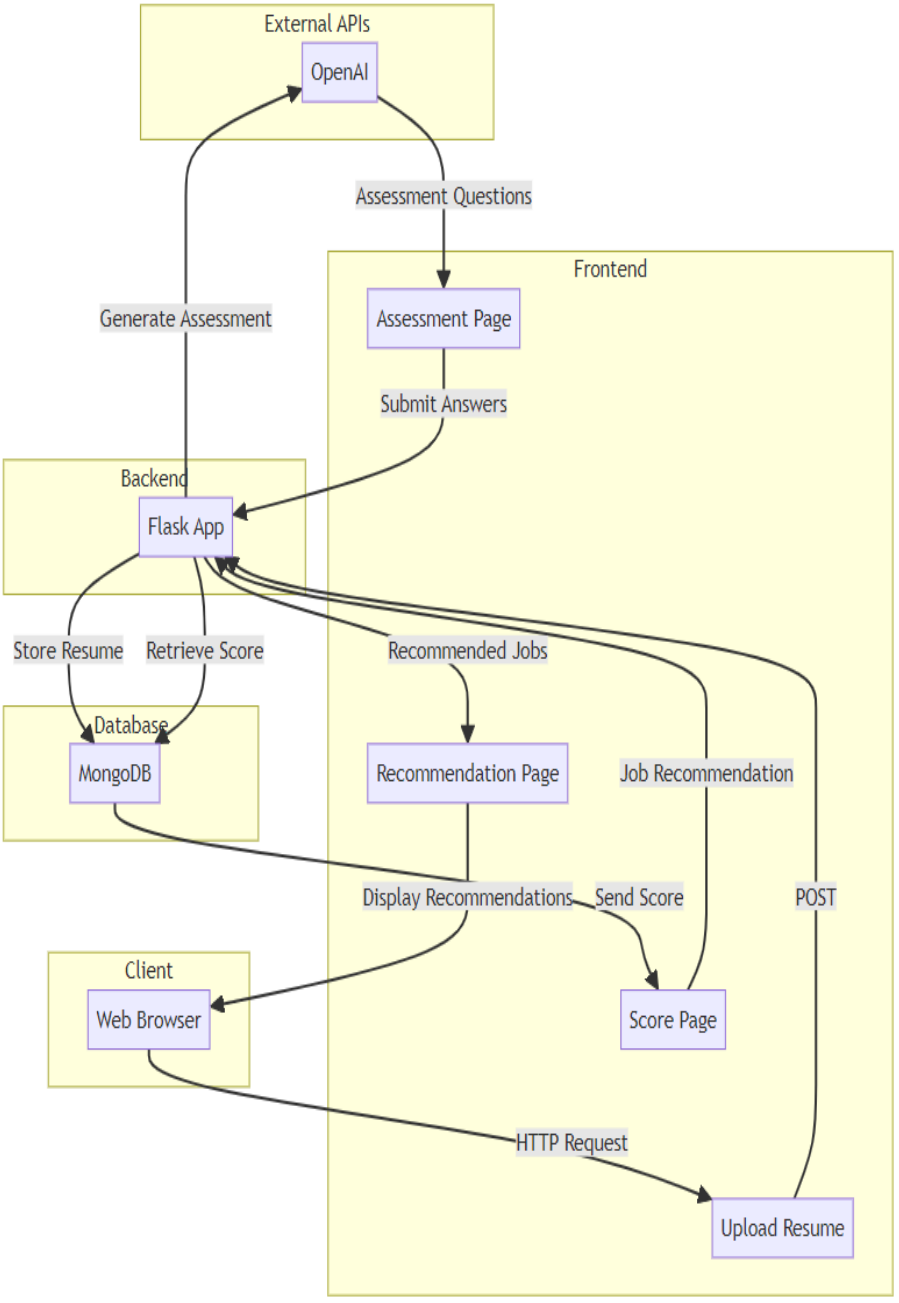


FIG. 4.1 ARCHITECTURE DIAGRAM

4.2 SYSTEM FLOW DIAGRAM

A Flow-Chart represents the flow of the application.It is a representation of a start to end of the application.

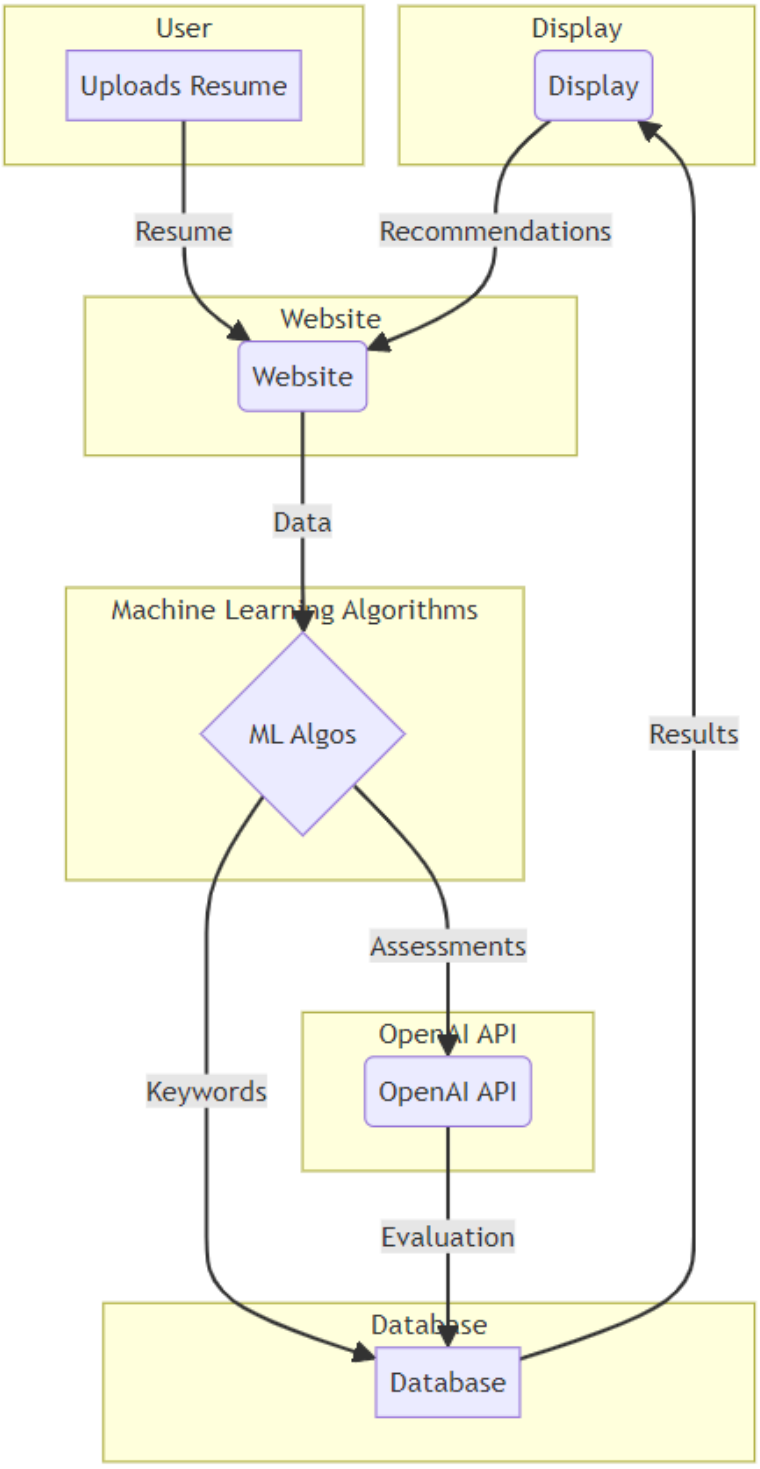


FIG. 4.2 SYSTEM FLOW DIAGRAM

4.3 SEQUENCE DIAGRAM

A sequence diagram is a type of interaction diagram in the Unified Modeling Language (UML) that illustrates the interactions between objects or components within a system in a chronological order. It provides a dynamic view of the system's behavior by depicting the sequence of messages exchanged between different entities over time.

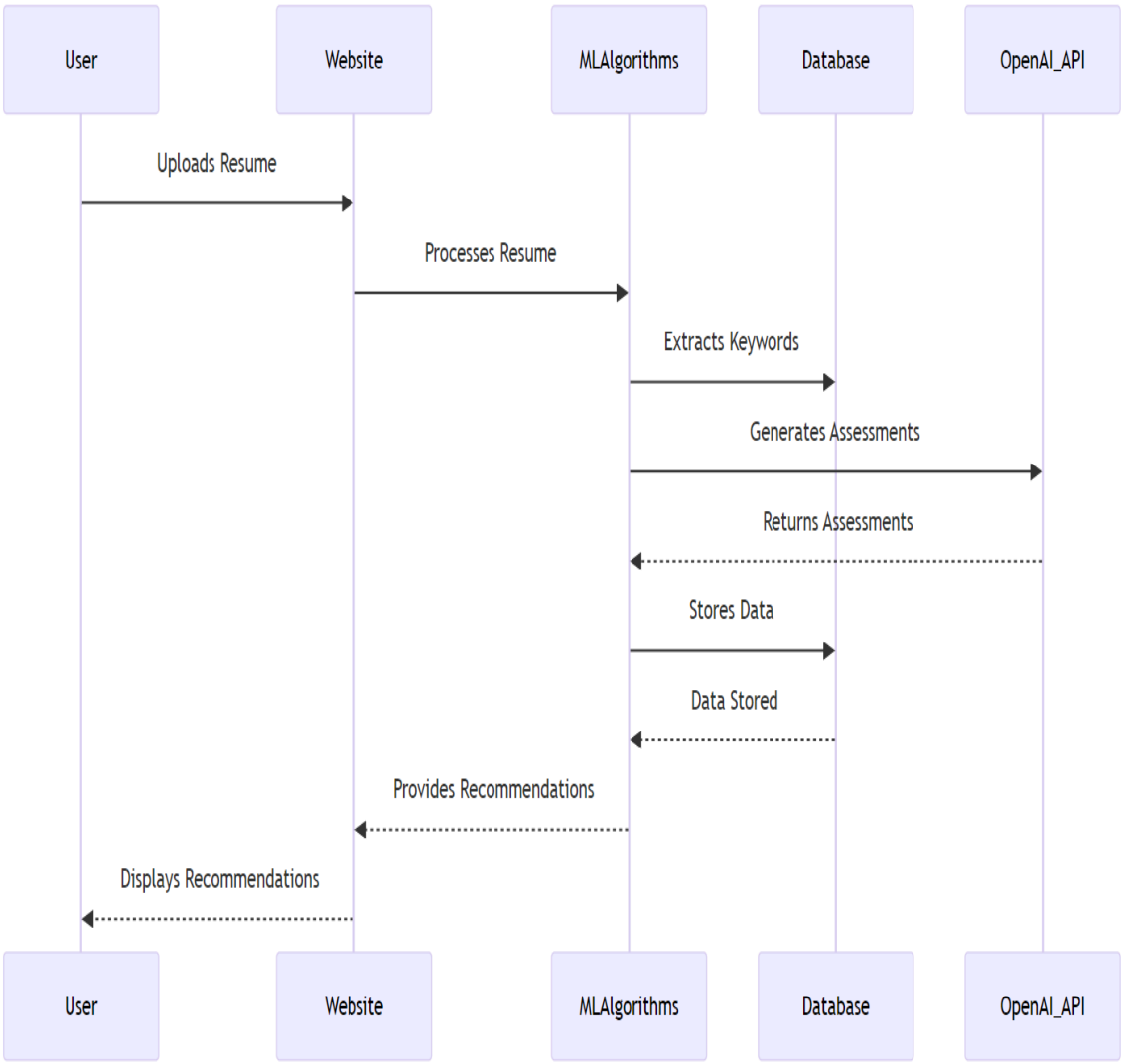


FIG. 4.3 SEQUENCE DIAGRAM

4.4 ENTITY-RELATIONSHIP DIAGRAM

An Entity-Relationship (ER) diagram is a graphical representation of the data model of a system or a business process.

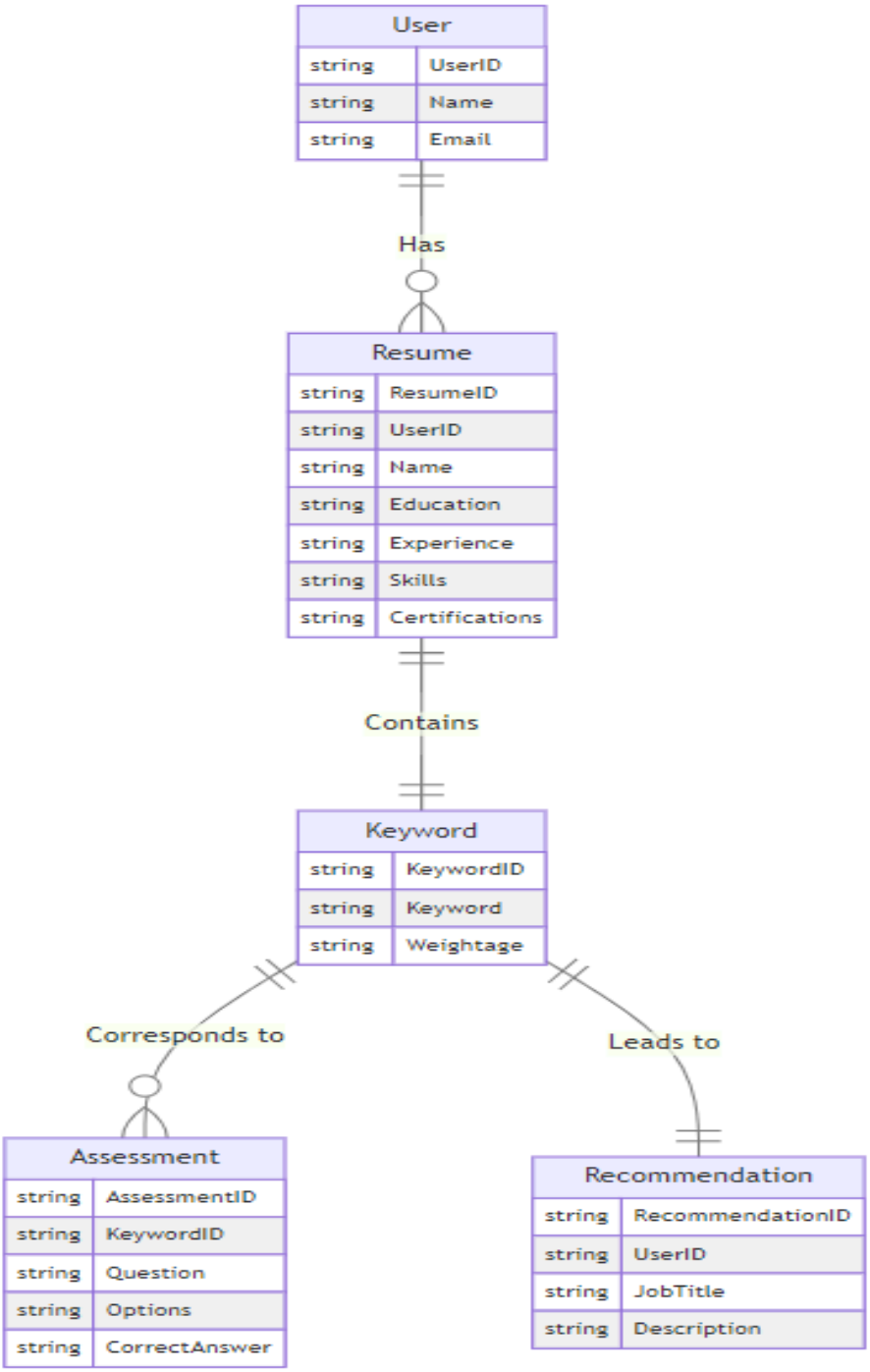


FIG. 4.4 ER DIAGRAM

CHAPTER 5
PROJECT DESCRIPTION

5.1 MODULES

1. User Authentication:

The user authentication module is responsible for verifying user identities during login and registration processes. It employs cryptographic techniques to securely hash and salt passwords, protecting user credentials against unauthorized access and data breaches. Robust authentication protocols, implemented using technologies like Flask and JWT (JSON Web Tokens), ensure the confidentiality and integrity of user authentication.

2. Profile Management:

The profile management module enables users to create and manage their profiles within the application. It provides functionalities for users to update, retrieve, and delete their profile information securely. Backend APIs handle communication between the frontend interface and the database, ensuring efficient retrieval and manipulation of user data.

3. Resume Upload:

The resume upload module facilitates the seamless uploading of resumes by users. It utilizes Flask APIs to handle file uploads, ensuring efficient transmission of resume data to the backend server. Uploaded resumes undergo preprocessing to extract textual content, which is then analyzed using natural language processing (NLP) techniques. Technologies like BeautifulSoup and NLTK are leveraged to parse and analyze resume content, extracting relevant keywords and skills for further processing.

4. Resume Analysis:

In the resume analysis module, advanced NLP algorithms are employed to extract valuable insights from user-uploaded resumes. Extracted keywords and skills are weighted based on their frequency and relevance to the user's career objectives. These weighted keywords are stored in a MongoDB database for subsequent analysis and recommendation generation. The backend utilizes Flask APIs to handle resume processing and database interactions, ensuring efficient and scalable analysis of resume data.

5. Skill Assessment:

The skill assessment module enables users to assess their proficiency levels in various domains relevant to their career aspirations. Leveraging insights from resume analysis, the backend dynamically generates personalized assessment questions tailored to the user's skills and experiences. Flask APIs handle assessment requests and process user responses, calculating proficiency scores based on predefined criteria. Adaptive algorithms adjust assessment difficulty levels based on user performance, providing a customized evaluation experience.

6. Proficiency Evaluation:

The proficiency evaluation module evaluates user responses from skill assessments to determine proficiency levels in different domains. User scores and assessment results are stored in the MongoDB database for future reference and analysis. Advanced algorithms adapt assessment difficulty levels based on user performance, ensuring accurate proficiency evaluation. Technologies like Flask and MongoDB facilitate efficient handling of assessment data and scoring mechanisms, enhancing the accuracy and reliability of proficiency evaluations.

7. Job Recommendation:

The job recommendation module utilizes machine learning algorithms to provide personalized job recommendations aligned with users' skills and career objectives. Analyzing user profiles and resume data, the backend identifies relevant job opportunities from a vast database of job listings. Recommendation algorithms such as content-based filtering is used to consider user preferences, historical job interactions, and industry trends to generate tailored job suggestions. These recommendations are presented to users through the frontend interface, enabling them to explore detailed job descriptions and apply for positions directly.

8. Job Matching:

The job matching module evaluates user qualifications against job requirements to ensure optimal alignment between users and job opportunities. Leveraging recommendation algorithms and user profile data, the backend assesses the suitability of job listings for individual users. Job matching algorithms consider factors such as skills, experience, location, and industry preferences to identify the most relevant job matches. Seamless integration with frontend components and backend services enables users to make informed decisions about their career paths, enhancing user engagement and satisfaction.

CHAPTER 6

SAMPLE CODING

urcareerapp.py

```
from flask import Flask, redirect, render_template, request, jsonify, url_for
from flask_pymongo import PyMongo
import openai
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import fitz
from docx import Document  # python-docx for DOCX files
import os

app = Flask(__name__, template_folder='templates', static_folder='static')
app.config["MONGO_URI"] = "mongodb://localhost:27017/urcareer"
app.config['UPLOAD_FOLDER'] = 'uploads'  # Folder to store uploaded files
mongo = PyMongo(app)
collection = mongo.db.resume

nltk.download('punkt')
nltk.download('stopwords')

stop_words = set(stopwords.words('english'))

openai.api_key =
'sk-proj-RLexVVZONbI85fp3CYqkT3BlbkFJ6VpPhCLy2dHmtqmGuLVp'  #
Replace with your OpenAI API key

# Function to extract text from PDF files
```

```
def extract_text_from_pdf(file_path):
    text = ""
    try:
        with fitz.open(file_path) as doc:
            for page in doc:
                text += page.get_text()
    except Exception as e:
            app.logger.error('An error occurred while extracting text from PDF: %s', str(e))
    return text

# Function to extract text from DOCX files
def extract_text_from_docx(file_path):
    text = ""
    try:
        doc = Document(file_path)
        for para in doc.paragraphs:
            text += para.text
    except Exception as e:
        app.logger.error('An error occurred while extracting text from DOCX: %s', str(e))
    return text

# Function to extract keywords from the resume content
def extract_keywords(resume_content):
    words = word_tokenize(resume_content)
        keywords = [word.lower() for word in words if word.isalnum() and word.lower() not in stop_words]
    word_freq = nltk.FreqDist(keywords)
```

```
    return word_freq

# Function to assign weightage to keywords
def assign_weightage(keywords):
        weighted_keywords = {keyword: freq * 10 for keyword, freq in keywords.items()}
    return weighted_keywords

# Function to generate MCQs using OpenAI
def generate_mcq(keyword):
      prompt = f"Generate a multiple-choice question (MCQ) for the technical keyword '{keyword}' with 4 options."
    response = openai.Completion.create(
        engine="text-davinci-003",
        prompt=prompt,
        max_tokens=150
    )
    return response.choices[0].text.strip()

# Function to evaluate the assessment answers
def evaluate_answers(user_answers, correct_answers):
    score = 0
    for user_answer, correct_answer in zip(user_answers, correct_answers):
        if user_answer == correct_answer:
            score += 1
    return score

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

```
try:
    if 'file' not in request.files:
        return jsonify({'error': 'No file part'}), 400
    file = request.files['file']
    if file.filename == '':
        return jsonify({'error': 'No selected file'}), 400
    if file:
        filename = file.filename
        file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(file_path)
        if filename.endswith('.pdf'):
            resume_content = extract_text_from_pdf(file_path)
        elif filename.endswith('.docx'):
            resume_content = extract_text_from_docx(file_path)
        else:
            return jsonify({'error': 'Unsupported file format'}), 400

        # Extract keywords from resume content
        keywords = extract_keywords(resume_content)
        # Assign weightage to keywords
        weighted_keywords = assign_weightage(keywords)
        # Store keywords with weightage in MongoDB
        mongo.db.keywords.insert_one({'keywords': weighted_keywords})

        return redirect(url_for('generate_assessment')), 302
except Exception as e:
    app.logger.error('An error occurred while uploading resume: %s', str(e))
    return jsonify({'error': 'An error occurred while uploading resume'}), 500
```

```
@app.route('/generate_assessment', methods=['GET','POST'])
def generate_assessment():
    try:
        # Fetch the highest-weighted keywords from MongoDB
        keywords_data = mongo.db.keywords.find_one(sort=[("_id", -1)])
        if not keywords_data:
            return jsonify({'error': 'No keywords found in database'}), 400

        keywords = keywords_data['keywords']
            sorted_keywords = sorted(keywords.items(), key=lambda item: item[1], reverse=True)
            top_keywords = [kw for kw, weight in sorted_keywords[:5]]  # Taking top 5 keywords

        questions = []
        for keyword in top_keywords:
            question = generate_mcq(keyword)
            questions.append(question)

        return render_template('generate_assessment.html', questions=questions)
    except Exception as e:
            app.logger.error('An error occurred while generating assessment: %s', str(e))
            return jsonify({'error': 'An error occurred while generating assessment'}), 500

@app.route('/submit_assessment', methods=['POST'])
def submit_assessment():
    try:
```

```
        user_answers = request.form.getlist('answers')
          correct_answers = request.form.getlist('correct_answers')  # This should
come from the generated questions

        score = evaluate_answers(user_answers, correct_answers)
        return jsonify({'score': score}), 200
    except Exception as e:
            app.logger.error('An error occurred while submitting assessment: %s',
str(e))
         return jsonify({'error': 'An error occurred while submitting assessment'}),
500

@app.route('/view_score', methods=['GET'])
def view_score():
    try:
        # Placeholder for retrieving score from MongoDB
        score = 80  # Placeholder for retrieving score
        return render_template('view_score.html', score=score)
    except Exception as e:
        app.logger.error('An error occurred while retrieving score: %s', str(e))
        return jsonify({'error': 'An error occurred while retrieving score'}), 500

@app.route('/recommend_job', methods=['GET'])
def recommend_job():
    try:
        # Placeholder for job recommendation based on score
                recommendation = "Software Engineer"  # Placeholder for
recommendation
                                return render_template('recommend_job.html',
```

```
recommendation=recommendation)
    except Exception as e:
        app.logger.error('An error occurred while recommending job: %s', str(e))
        return jsonify({'error': 'An error occurred while recommending job'}), 500

@app.route('/')
def index():
    return render_template('welcome.html')

if __name__ == '__main__':
    app.run(debug=True)
```

urcareer_job_recommendation.ipynb

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.preprocessing import MinMaxScaler
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

nltk.download('punkt')
nltk.download('stopwords')
# Collaborative Filtering
interaction_data = {
    'Resume_ID': [0, 0, 1, 1, 2, 2, 3, 3, 4, 4],
    'Job_ID': [0, 2, 1, 3, 2, 4, 3, 0, 4, 1],
    'Interaction': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

```
}

interaction_df = pd.DataFrame(interaction_data)
user_item_matrix = interaction_df.pivot(index='Resume_ID', columns='Job_ID', values='Interaction').fillna(0)

user_similarity = cosine_similarity(user_item_matrix)

def get_collaborative_recommendations(user_id, user_similarity, top_n=3):
    similar_users = list(enumerate(user_similarity[user_id]))
            similar_users = sorted(similar_users, key=lambda x: x[1], reverse=True)[1:top_n+1]
    recommended_jobs = set()
    for user, score in similar_users:
                user_jobs = set(interaction_df[interaction_df['Resume_ID'] == user]['Job_ID'])
        recommended_jobs = recommended_jobs.union(user_jobs)
    return recommended_jobs

collab_recommendations = get_collaborative_recommendations(0, user_similarity)
print("Collaborative Filtering Recommendations for Resume 0:", collab_recommendations)

# Content-Based Filtering
jobs_df['Combined'] = jobs_df['Key Skills'] + ' ' + jobs_df['Role Category'] + ' ' + jobs_df['Functional Area'] + ' ' + jobs_df['Industry'] + ' ' + jobs_df['Job Title']
resumes_df['All_Skills'] = resumes_df['Skills'].apply(lambda x: ' '.join(x))
```

```
tfidf_vectorizer = TfidfVectorizer(stop_words='english')
job_tfidf = tfidf_vectorizer.fit_transform(jobs_df['Combined'])
resume_tfidf = tfidf_vectorizer.transform(resumes_df['All_Skills'])

cosine_similarities = cosine_similarity(resume_tfidf, job_tfidf)

def get_content_based_recommendations(resume_id, cosine_similarities, top_n=3):
    sim_scores = list(enumerate(cosine_similarities[resume_id]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)[:top_n]
    recommended_jobs = [job[0] for job in sim_scores]
    return recommended_jobs
content_recommendations = get_content_based_recommendations(0, cosine_similarities)
print("Content-Based Filtering Recommendations for Resume 0:", content_recommendations)
```

CHAPTER 7

OUTPUTS AND FINAL RESULTS

urcareer Front-end:

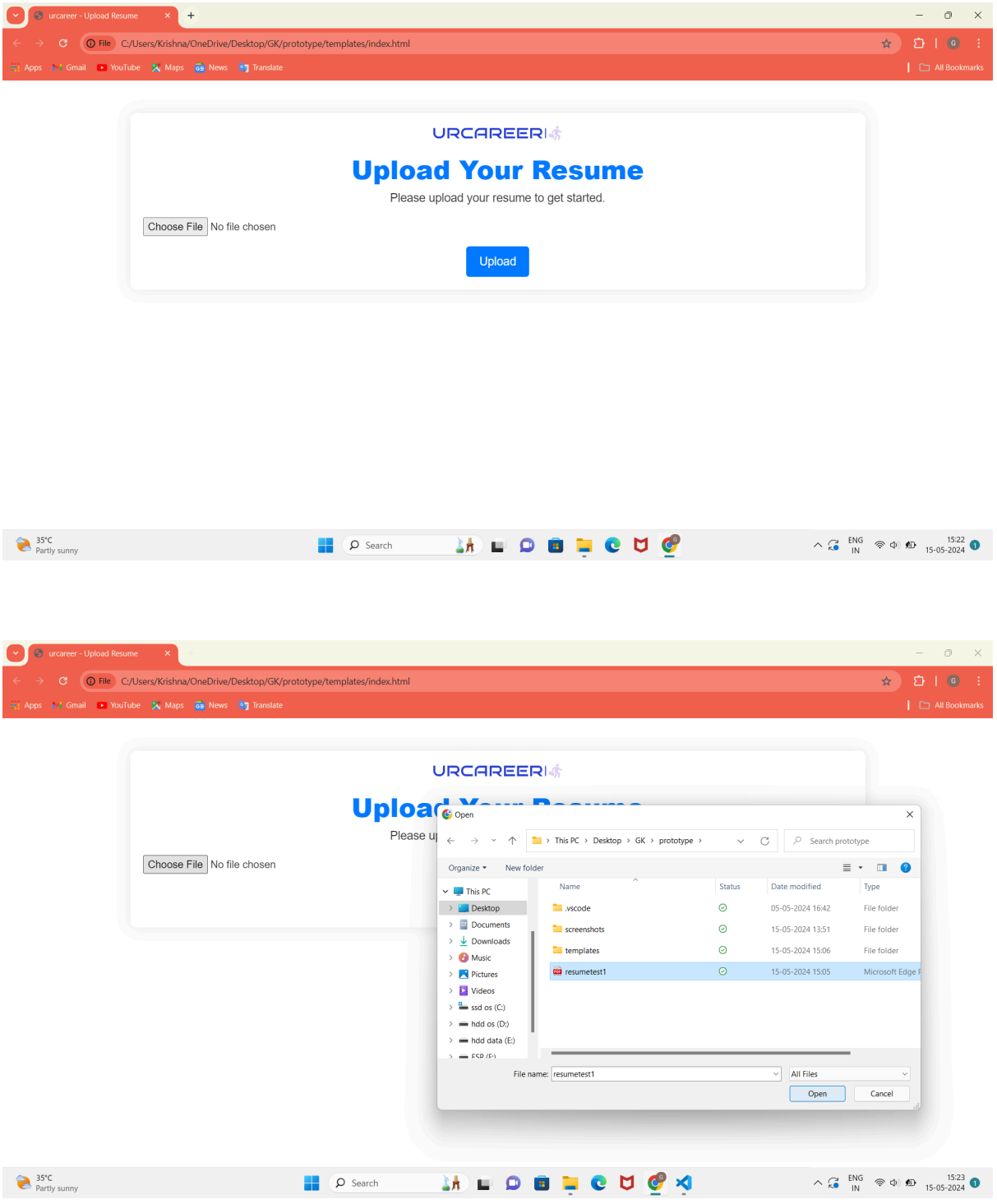


FIG. 5.1 Uploading resume in the urcareer site

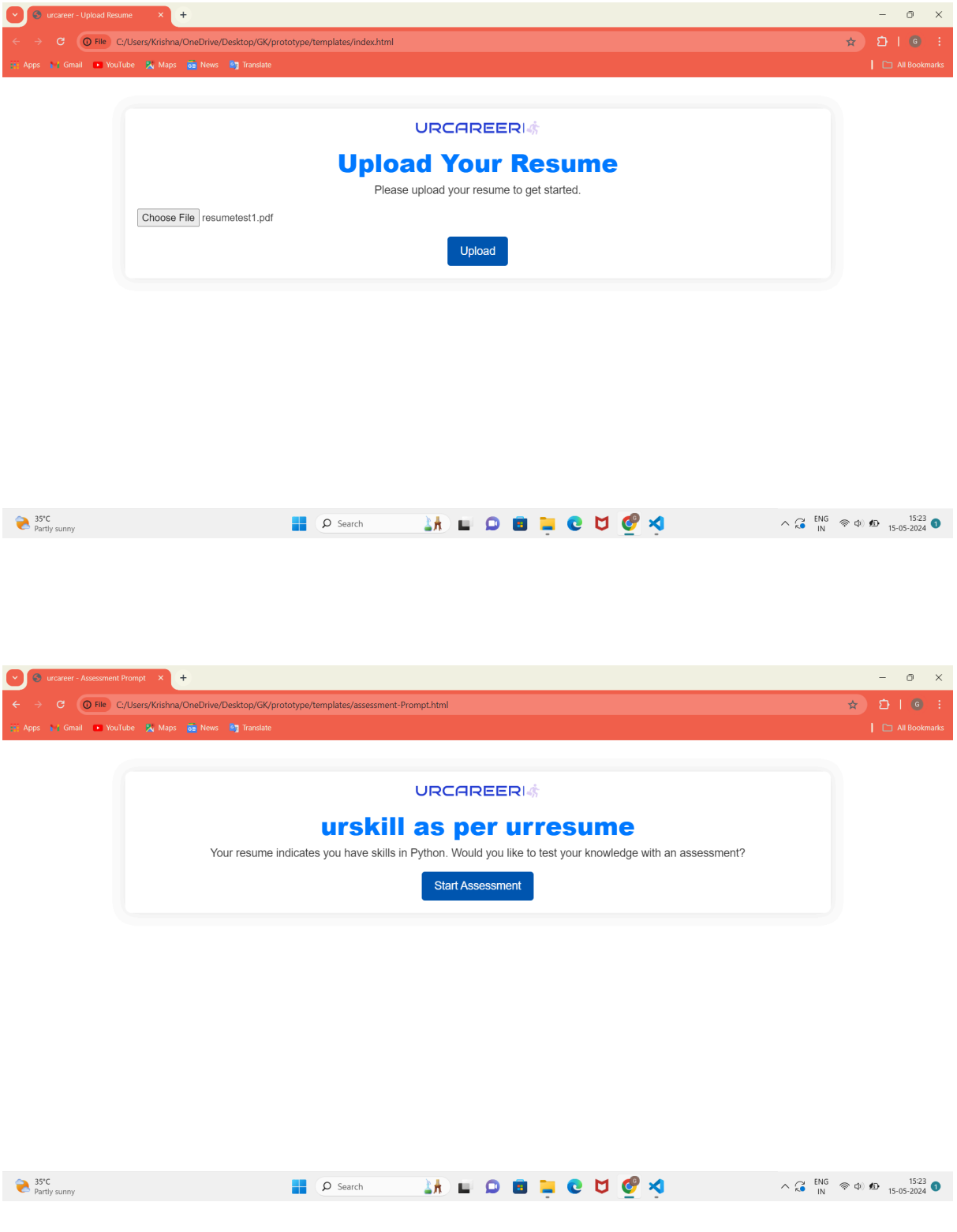
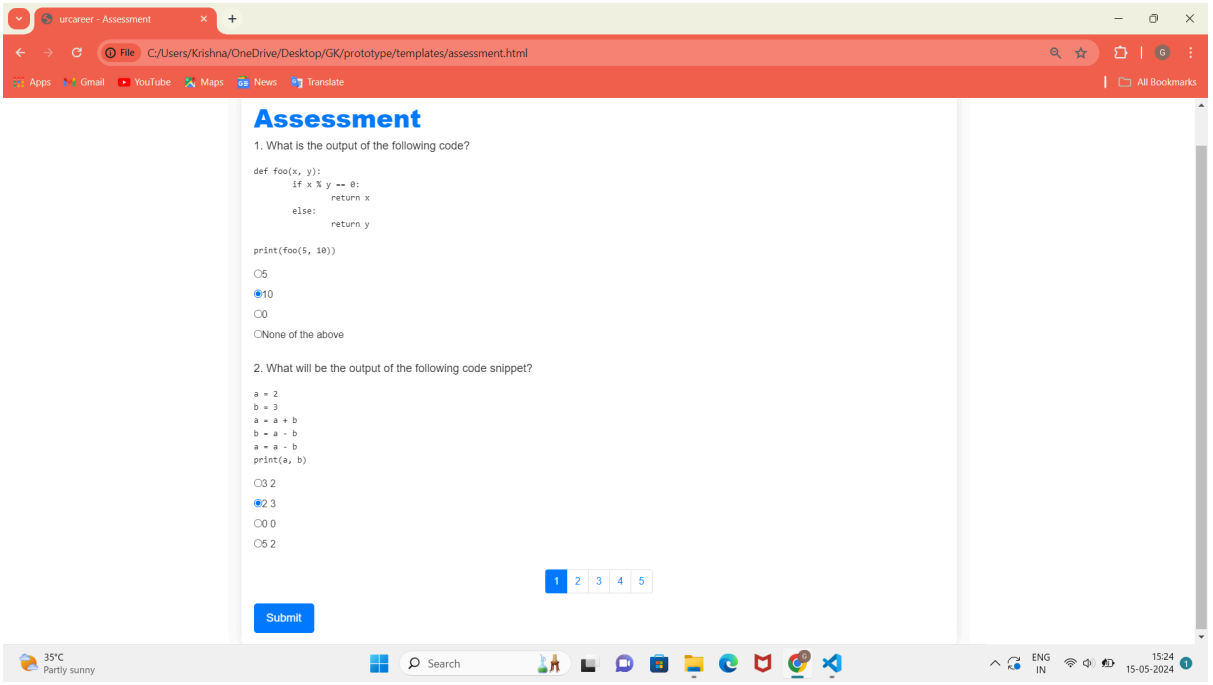


FIG. 5.2 Assessment generated based on Skill



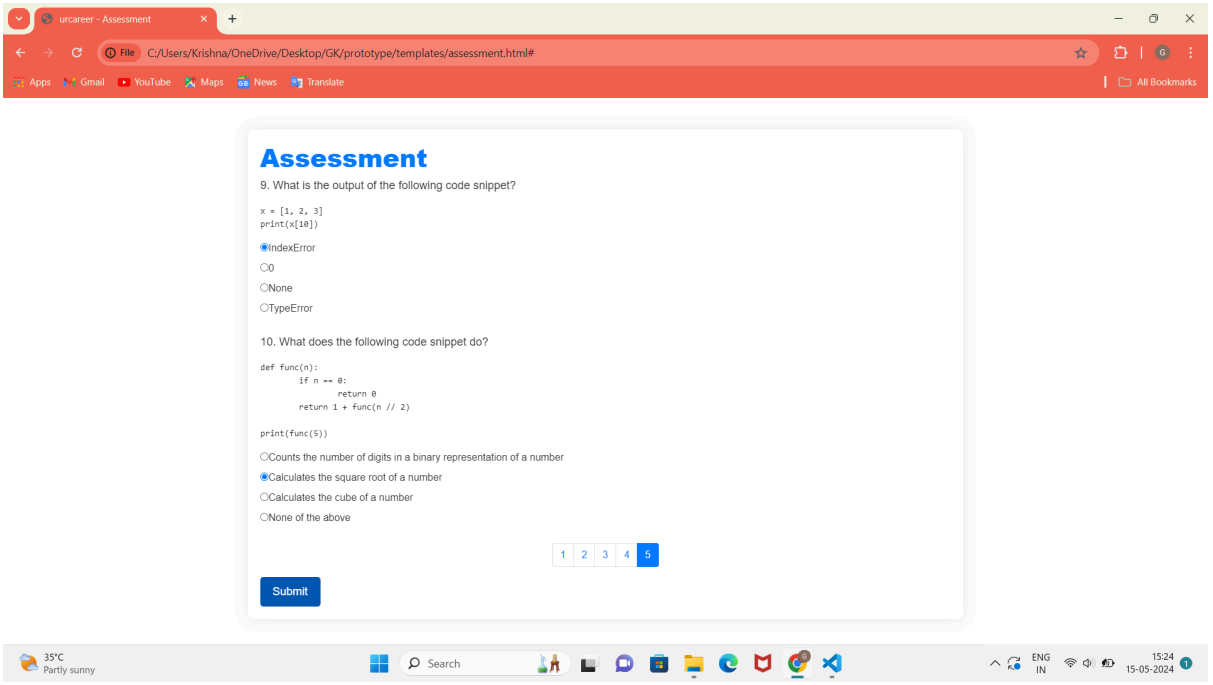
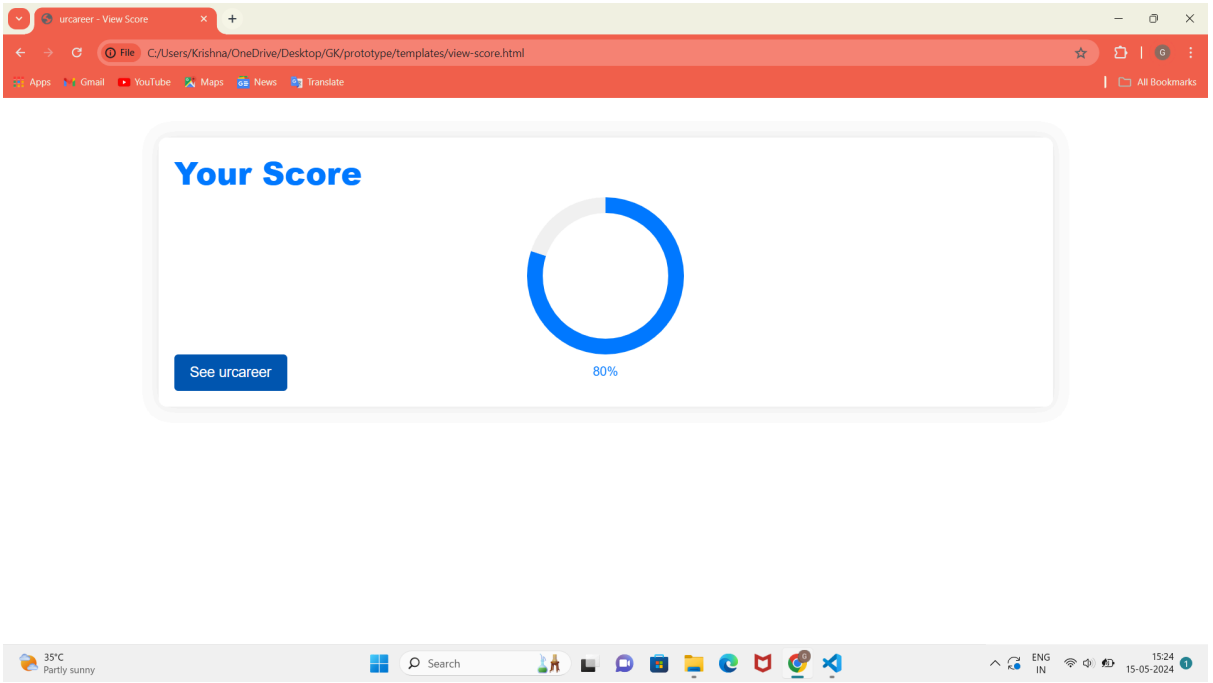
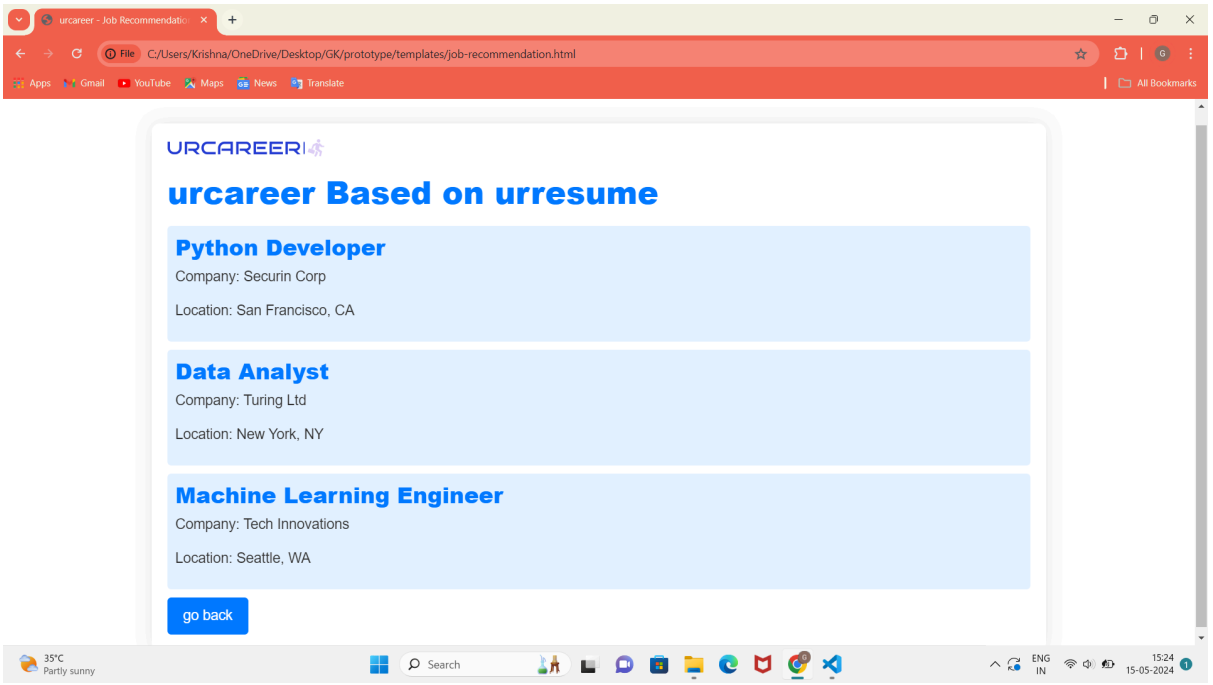


FIG. 5.3 urcareer Assessment Page





URCAREER

FIG. 5.4 View Score and Job Recommendation

urcareer: Back-end

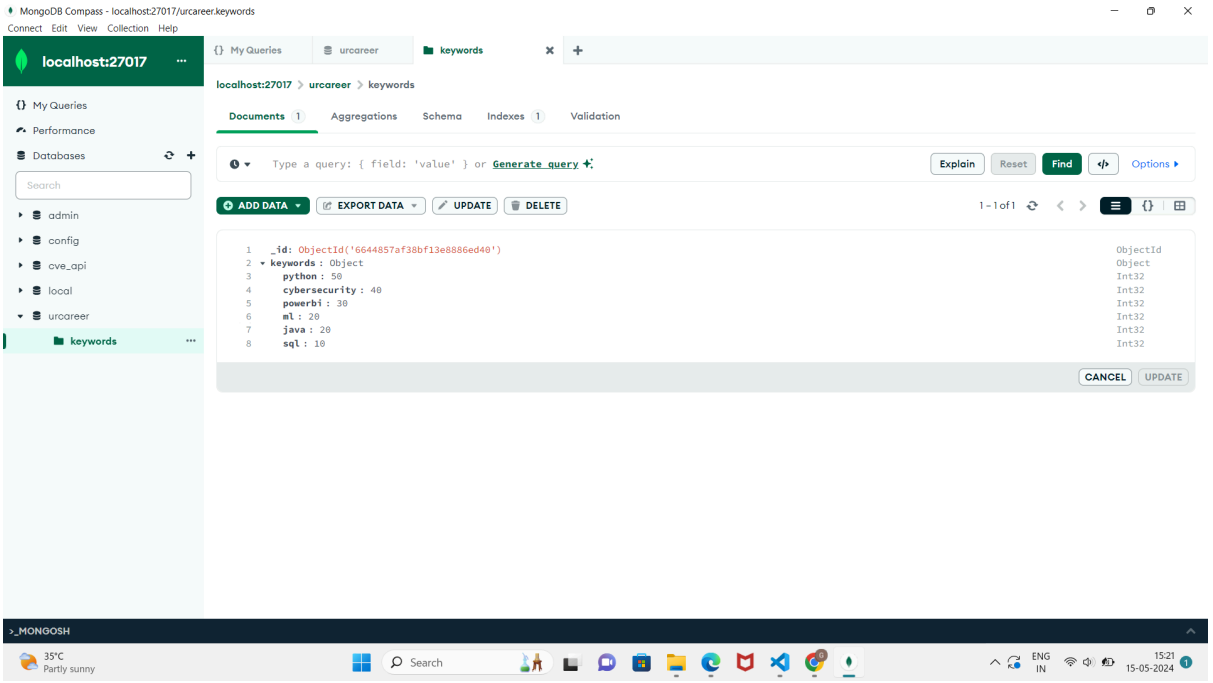


FIG. 5.5 Keywords scraped and stored in DB

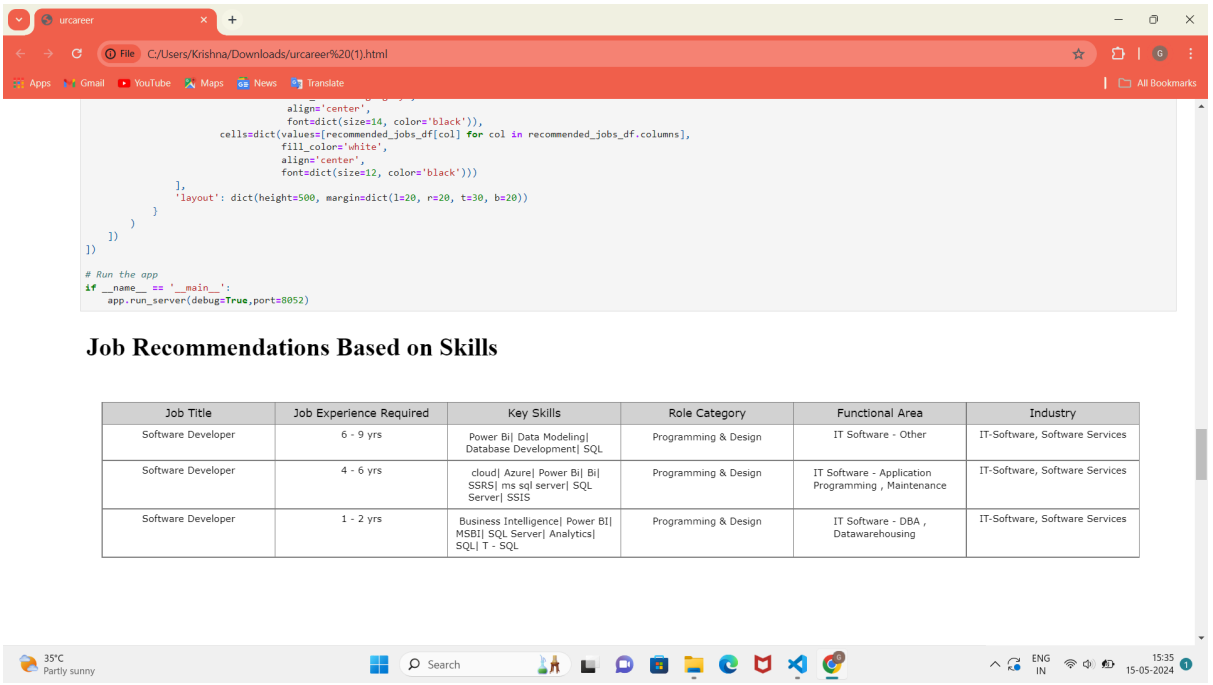
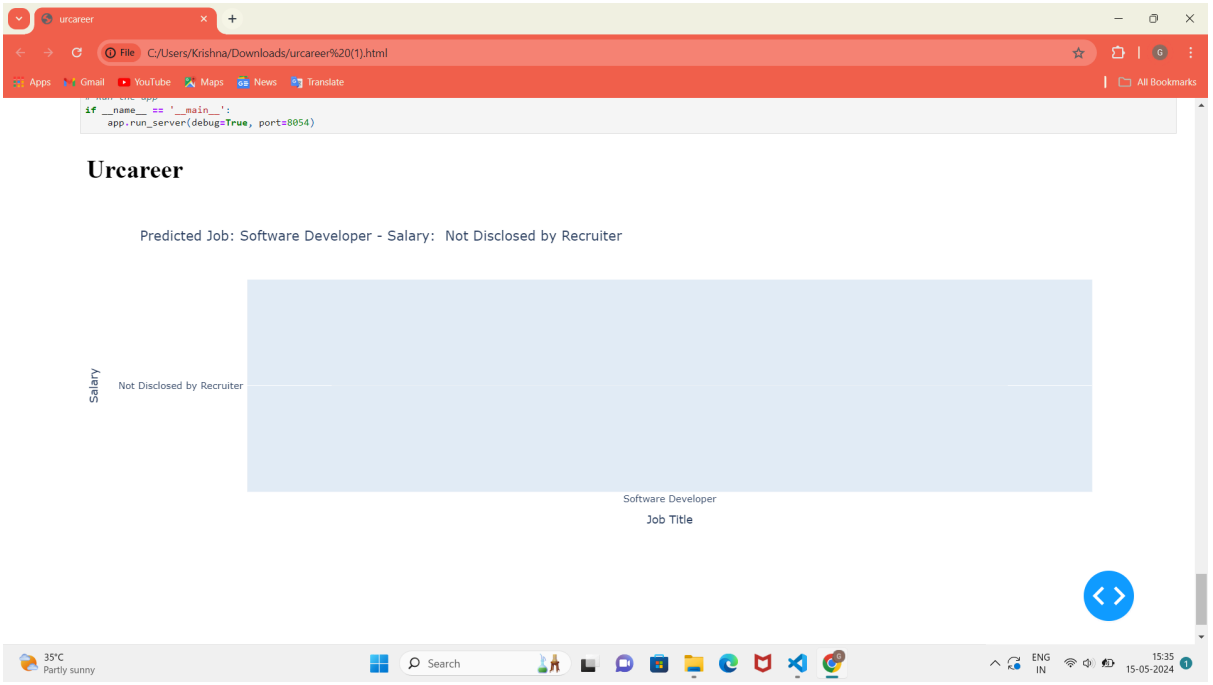


FIG. 5.6 Content-Based Filtering



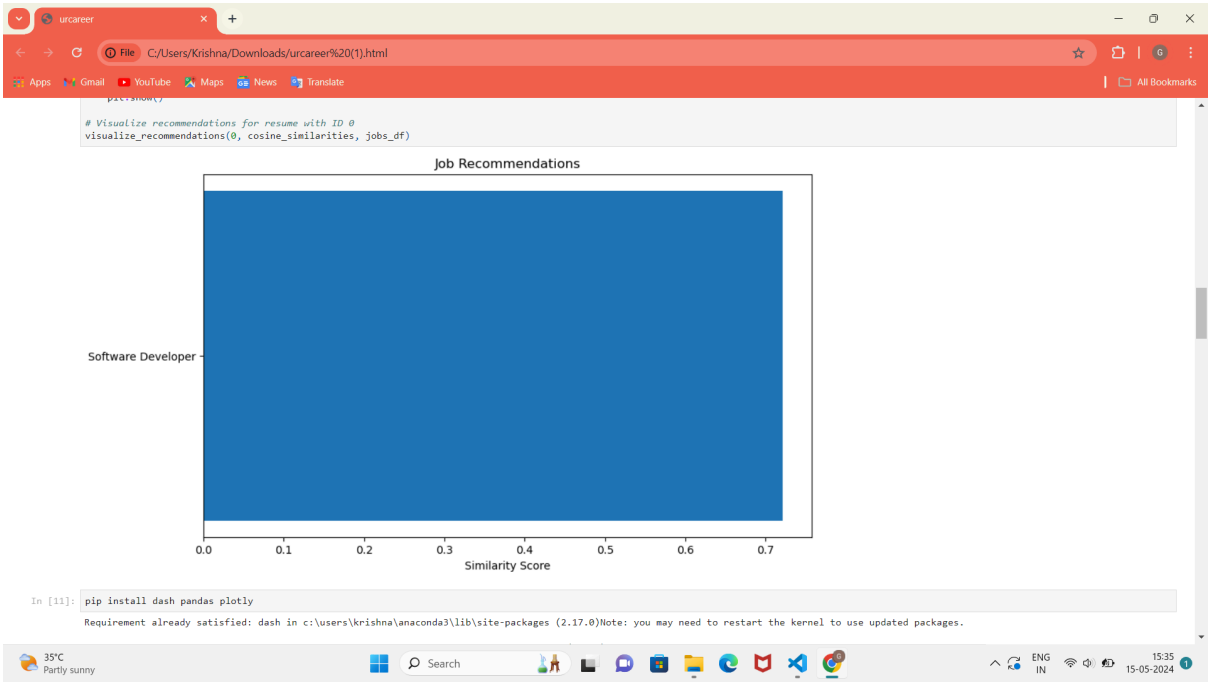


FIG. 5.7 Analytical Predictions of Jobs at urcareer

CHAPTER 8
CONCLUSION AND FUTURE ENHANCEMENTS

In conclusion, the career guidance application leverages advanced technologies and robust backend architecture to provide users with personalized career insights, skill assessments, and job recommendations. Through modules such as user authentication, resume analysis, and job matching, the application empowers users to make informed decisions about their career paths. By integrating natural language processing (NLP), machine learning algorithms, and database management systems, the application delivers a seamless and intuitive user experience.

The project's backend utilizes Flask for API development and MongoDB for efficient data storage and retrieval, ensuring scalability and performance. Frontend components are built using HTML, CSS, and JavaScript, providing a visually appealing and responsive user interface. The application's modular architecture facilitates easy maintenance and future enhancements, enabling seamless integration of new features and functionalities.

Future Enhancements:

Integration with Online Learning Platforms: Integrate with online learning platforms to provide personalized learning recommendations and skill development opportunities based on user career goals and assessment results.

Gamification Features: Incorporate gamification elements such as badges, leaderboards, and rewards to incentivize user engagement and motivate users to actively participate in skill assessments and career development activities.

LIST OF REFERENCES

- Harvey & Paul Deitel & Associates, Harvey Deitel and Abbey Deitel, “Internet and World Wide Web - How ToProgram”, Fifth Edition, Pearson Education, 2011.

- Jeffrey C and Jackson, “Web Technologies A Computer Science Perspective”, Pearson Education, 2011.

- S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, Third Edition, 2015.

- Nils J. Nilsson, Artificial Intelligence: A New Synthesis (1 ed.), Morgan-Kaufmann, 1998. ISBN 978- 1558605350.

- Stephen Marsland, “Machine Learning – An Algorithmic Perspective”, Second Edition, Chapman and Hall/CRC Machine Learning and Pattern Recognition Series, 2014.

- Peter Flach, “Machine Learning: The Art and Science of Algorithms that Make Sense of Data‖”, First Edition,Cambridge University Press, 2012.

- Tom M Mitchell, “Machine Learning”, First Edition, McGraw Hill Education, 2013.

- Trevor Hastie, Robert Tibshirani and Jerome Friedman, “The Elements of Statistical Learning (ESL)”, 2nd edition,Springer, 2016. ISBN 978-0387848570.