Department of Computer Science and Engineering

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Autonomous)

Bachupally, Kukatpally, Hyderabad, Telangana, India,500090

2024-2025

A Real Time Research Project/ Societal Related Project Report On

COURSE RECOMMENDER SYSTEM

Submitted in partial fulfillment of the requirements for the award of the

Bachelor of Technology

In

Department of Computer Science and Engineering

By

M. Sai Harshith	23241A05N8
K. Dhanesh	23241A05N2
G. Bhanush	23241A05M3
G. Garuda Sai Vamshi	23241A05M2

Under the Esteemed guidance of

B. Sindhuja

Assistant Professor

GOKARAJU RANGARAJU

INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Autonomous)

CERTIFICATE

This is to certify that the Real Time Research Project/ Societal Related Project entitled "**Course Recommender System**" is submitted by **M. Sai Harshith (23241A05N8), K. Dhanesh (23241A05N2), G. Bhanush (23241A05M3), G. Garuda Sai Vamshi (23241A05M2)** in partial fulfillment of the award of a degree in BACHELOR OF TECHNOLOGY in Computer Science and Engineering during the academic year **2024-2025**.

INTERNAL GUIDE	HEAD OF THE DEPARTMENT
B. Sindhuja	**Dr.B.SANKARA BABU**
Assistant Professor	**Professor**

ACKNOWLEDGEMENT

Many people helped us directly and indirectly to complete our project successfully. We would like to take this opportunity to thank one and all. First, we wish to express our deep gratitude to our internal guide **B. Sindhuja, Assistant Professor**, for his/her support in the completion of our project report. We express our thanks to RTRP Coordinator Dr. **G.R.Sakthidharan**. We thank our Section **Faculty coordinators Ms. R.Soujanya and Mr. K. Srikanth** for their encouragement, coordination and improvement given until our completion of our project. We are extend our thanks to our HOD **Dr.B.SankaraBabu**, for providing resources. We are submitting our grateful thanks to our Principal **Dr. J. Praveen** for providing the facilities to complete our Real Time Research Project/ Societal Related Project. We would like to thank all our faculty and friends for their help and constructive criticism during the project completion phase. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

M. Sai Harshith	**23241A05N8**
K. Dhanesh	**23241A05N2**
G. Bhanush	**23241A05M3**
G. Garuda Sai Vamshi	**23241A05M2**

DECLARATION

We hereby declare that the Real Time Research Project/ Societal Related Project entitled "**Course Recommender System** " is the work done during the period from **2024-2025** and is submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering from **Gokaraju Rangaraju Institute of Engineering and Technology(Autonomous), Hyderabad)**. The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

M. Sai Harshith	**(23241A05N8)**
K. Dhanesh	**(23241A05N2)**
G. Bhanush	**(23241A05M3)**
G. Garuda Sai Vamshi	**(23241A05M2)**

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

In the era of digital education, selecting the right course from a vast array of options can be overwhelming for learners. This paper presents a Course Recommendation System leveraging Natural Language Processing (NLP) to provide personalized course suggestions based on user preferences, skills, and interests. The system processes user input, including textual descriptions of interests, prior knowledge, and career goals, using NLP techniques such as text embedding, keyword extraction, and sentiment analysis. It then matches the extracted features with a database of courses to generate ranked recommendations. Machine learning models such as TF-IDF enhance the accuracy of semantic matching between user queries and course descriptions. The system aims to improve user engagement and learning outcomes by providing context-aware, relevant, and personalized recommendations. Experimental evaluations demonstrate the effectiveness of NLP-driven approaches in refining course selection, ultimately aiding learners in making informed decisions.

Keywords: Course Recommendation, NLP, Machine Learning, Personalization, Semantic Matching".

CHAPTER 1
INTRODUCTION

With the rapid growth of online learning platforms, the availability of educational courses has expanded significantly. While this offers learners a wide range of choices, it also creates a challenge in selecting the most relevant courses that align with their interests, skills, and career goals. Traditional recommendation systems often rely on user ratings, collaborative filtering, or predefined categories, which may not effectively capture a user's specific learning needs.

To address this challenge, Natural Language Processing (NLP) has emerged as a powerful tool for understanding user preferences through textual input. By analyzing descriptions of user interests, prior knowledge, and career aspirations, an NLP-based Course Recommendation System can generate personalized suggestions tailored to individual learning objectives. Techniques such as keyword extraction, sentiment analysis, word embeddings, and semantic similarity measurement enable the system to match user queries with course descriptions in a meaningful way.

This research focuses on developing an NLP-driven course recommendation system that enhances the learning experience by providing accurate, context-aware recommendations. The proposed system leverages machine learning and deep learning models to process and understand natural language input, improving the relevance of suggested courses. By integrating text analysis, semantic matching, and ranking algorithms, the system aims to bridge the gap between learners and the vast pool of educational resources, ultimately improving user engagement and learning outcomes.

CHAPTER 2
SYSTEM REQUIREMENTS

The following are the requirements of the project.

2.1 Software Requirements:

a. Programming Languages Used: Python .

b. Frameworks & Libraries:

- Flask – for building the web interface (backend + frontend)
- scikit-learn – for TF-IDF vectorization and cosine similarity
- pandas – for data handling and preprocessing
- numpy – for numerical operations
- jQuery – for handling real-time suggestion display in the UI

c. Web Technologies Used:

- HTML5, CSS3 (Bootstrap for styling)
- JavaScript (for live search/autocomplete)

2.2 Hardware Requirements:

- Processor Intel Ultra core 7
- RAM 8GB
- Storage At least 1 GB free space (for dataset, virtual environment, libraries)
- System Type 64-bit OS (Windows)
- Internet Required for installing dependencies and accessing external APIs or course data

2.3 Dataset:

The dataset used in this project is sourced from publicly available online course repositories. It contains information on thousands of courses across various domains. Each record typically includes:

- Course Title: Name of the course.
- Course Description: Summary of course contents.
- Category: Subject domain (e.g., Data Science, Business, IT).
- URL: Direct link to the course.

This structured data enables natural language processing and filtering based on relevance.

CHAPTER 3
LITERATURE SURVEY

Several studies and systems have previously tackled course recommendation:

3.1 Content-Based Filtering

Content-Based Filtering (CBF) is one of the earliest and most widely used recommendation approaches. It relies entirely on the attributes of items (courses) rather than interactions between users. The main idea is to recommend items that are similar to those the user liked in the past, or in our case, match the keywords or topics specified by the user.

Key Concepts:

- Feature Representation: Courses are represented using feature vectors derived from course metadata (title, description, category, difficulty, etc.).
- Textual Features: Common techniques include TF-IDF, Word Embeddings to represent course descriptions.
- Similarity Metrics: The most common measure used is cosine similarity, which compares two vectors by the cosine of the angle between them.

3.2 Collaborative Filtering

Collaborative Filtering (CF) is a user-driven approach. Instead of using content features, it recommends items based on user behavior, such as ratings, enrollments, clicks, or watch time. It identifies similar users and suggests items that users with similar tastes have interacted with.

Types of Collaborative Filtering:

- User-Based CF: Recommends items that similar users liked.
- Item-Based CF: Recommends items similar to those the user previously liked.

Matrix Factorization Techniques: Such as SVD (Singular Value Decomposition), which decompose the user-item interaction matrix into latent factors.

3.3 Hybrid Models

Hybrid recommender systems combine the strengths of content-based and collaborative filtering methods to overcome the limitations of each.

Types of Hybridization:

- Weighted Hybrid: Combines scores from both methods with different weights.
- Switching Hybrid: Chooses one method dynamically depending on the context (e.g., use content-based for new users).

Feature Augmentation: Uses collaborative filtering outputs as additional features in content-based systems or vice versa.

3.4 NLP in Recommendation

Natural Language Processing (NLP) plays a critical role in modern content-based recommendation systems. With most course metadata being textual (titles, descriptions, reviews), NLP techniques allow systems to understand and compare natural language efficiently.

Common NLP Techniques Used:

- TF-IDF Vectorization (TfidfVectorizer from sklearn) : Converts course titles + descriptions into feature vectors.
  Captures the importance of words while reducing noise from common terms.
- Stop Word Removal : Automatically done by setting stop_words='english' in TfidfVectorizer.
- Text Normalization : Includes lowercasing and tokenization (done internally by TfidfVectorizer).
- Cosine Similarity : Measures similarity between the user input and course descriptions to find the most relevant matches.

CHAPTER 4
PROPOSED MODEL, MODULES DESCRIPTION

Proposed Model

The system applies TF-IDF (Term Frequency-Inverse Document Frequency) for feature extraction and cosine similarity for course recommendation.

a. Workflow Overview

1. **User Input:** The user types a keyword or phrase.
2. **Text Vectorization:** The system vectorizes course descriptions using TF-IDF.
3. **Similarity Calculation:** Computes similarity between user input and course descriptions.
4. **Result Display:** Returns top N courses based on similarity score.

b. Why Content-Based Filtering?

1. Handles cold start (new users) effectively.
2. Works well without interaction data.
3. Explains recommendations based on matched keywords.

c. Algorithms Used

1. TF-IDF Vectorizer (sklearn.feature_extraction.text)
2. Cosine Similarity (sklearn.metrics.pairwise)
3. Text Preprocessing: nltk, re, string libraries

d. Evaluation Metrics

1. Though content-based filtering is less suited to traditional recommender metrics (like RMSE), we assess:
2. Precision@K
3. User Satisfaction (Survey-based)
4. Semantic Relevance

Module Description

To build a scalable and modular system, the architecture is divided into the following modules:

Table 1. Module Description

Module	Name	Description
1	Data Ingestion	Load and validate data from CSV.
2	Text Preprocessing	Clean and tokenize text.
3	Vectorization	Convert text to TF-IDF vectors.
4	Similarity Engine	Calculate similarity using cosine metric.
5	Filtering & Ranking	Apply optional filters and sort results.
6	Recommendation Engine	Extract and return top-N results.
7	Web Interface	Accept input and display courses via Flask.

4.1 Modules

4.1.1 Data Ingestion

- Implemented using pandas.
- CSV parsed and null entries dropped.
- Data normalized to lowercase for consistency.

4.1.2 Text Preprocessing

- Removal of stop words, punctuation, and HTML tags.
- Tokenization and lemmatization using nltk.
- Creation of a clean corpus for model input.

4.1.3 Feature Extraction

- TF-IDF vectorizer fit on course descriptions.
- Resulting sparse matrix stored for efficient similarity checks.

4.1.4 Similarity Engine

- Transforms user input into TF-IDF vector.
- Computes similarity scores with the corpus using cosine_similarity.
- Optimized using sparse matrix algebra.

4.1.5 Filtering and Ranking

- Supports optional filters (e.g., difficulty, rating).
- Results are sorted by descending similarity score.

4.1.6 Web Interface

- Built using Flask.
- Frontend includes:
  - Input textbox
  - Search button
  - Dynamic display of top N course cards

CHAPTER 5
IMPLEMENTATION, EXPERIMENTAL RESULTS

5.1 Implementation

5.1.2 Front End

Style

```
body {
   margin: 0;
   padding-top: 50px;
   font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-image: url('https://images.unsplash.com/photo-1584697964404-cd6c853be176?
auto=format&fit=crop&w=1950&q=80');
   background-size: cover;
   background-repeat: no-repeat;
   background-position: center;
   background-attachment: fixed;
   color: #fff;
   min-height: 100vh;
  }
.container {
 max-width: 800px;
 background-color: rgba(0, 0, 0, 0.85);
 padding: 30px;
 border-radius: 10px;
 margin: auto;
 }
 h1, h2, h4 {
 color: #ffc107;
 }
 .form-control {
 background-color: #fff;
 color: #000;
 }
 .suggestion-item {
 cursor: pointer;
 background-color: #fff;
 color: #000;
 }
```

```
.list-group-item {
 background-color: #444;
 border-color: #666;
 color: #fff;
 }

 a {
 color: #17a2b8;
 }
```

INDEX.HTML}

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Course Recommender System</title>
  <!-- Bootstrap CSS (optional) -->
  <link
    rel="stylesheet"
    href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
  >
  <!-- Your Custom Styles -->
  <style>
    /* 1) Make html/body span full height */
    html, body {
      height: 100%;
      margin: 0;
    }
    /* 2) Page background image—no fallback color */
    body {
      background:
        linear-gradient(rgba(0,0,0,0.6), rgba(0,0,0,0.6)),
                    url('https://images.unsplash.com/photo-1530919048397-f5a2b3efecb9?ixlib=rb-
4.0.3&auto=format&fit=crop&w=1080&q=80')
        no-repeat center center fixed;
      background-size: cover;
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      color: #fff;
      padding-top: 50px;
    }
```

```
/* 3) Dark, semi-transparent content container */
  .container {
    max-width: 800px;
    margin: auto;
    background-color: rgba(0, 0, 0, 0.85);
    padding: 30px;
    border-radius: 10px;
  }

  /* 4) Headings and link colors */
  h1, h2, h4 {
    color: #ffc107;
    text-align: center;
    margin-bottom: 1rem;
  }
  a {
    color: #17a2b8;
  }
  a:hover {
    text-decoration: underline;
  }

  /* 5) Form controls */
  .form-control {
    width: 100%;
    padding: 12px;
    margin: 12px 0;
    border: none;
    border-radius: 6px;
    font-size: 16px;
    background-color: #fff;
    color: #000;
    box-shadow: inset 0 1px 3px rgba(0,0,0,0.1);
  }
  button.btn {
    width: 100%;
    padding: 12px;
    margin-top: 10px;
    background-color: #ffc107;
    color: #000;
```

```
 font-weight: bold;
 border: none;
 border-radius: 6px;
 cursor: pointer;
 transition: background-color 0.3s ease;
 }
 button.btn:hover {
 background-color: #e0a800;
 }
/* 6) Suggestions dropdown */
   #suggestions {
     position: absolute;
     top: 100%;
     left: 0;
     right: 0;
     z-index: 1000;
     border-radius: 6px;
     overflow: hidden;
     box-shadow: 0 4px 10px rgba(0,0,0,0.2);
   }
   .suggestion-item {
     padding: 10px;
     background-color: #fff;
     color: #000;
     border-bottom: 1px solid #ddd;
     cursor: pointer;
     transition: background-color 0.2s ease;
   }
   .suggestion-item:hover {
     background-color: #f0f0f0;
   }
/* 7) Recommendation list items */
   .list-group-item {
     background-color: #444;
     border-color: #666;
     color: #fff;
     margin-bottom: 0.5rem;
     border-radius: 6px;
     padding: 15px;
```

```
}
 </style>
</head>
<body>
  <div class="container">
    <h1>Find Courses You'll Love</h1>
    <!-- Search Form -->
    <form method="post" action="/">
      <div class="form-group position-relative">
        <input
          type="text"
          class="form-control"
          id="query"
          name="query"
          placeholder="e.g. I want to learn data science"
          autocomplete="off"
          required
        >
        <ul id="suggestions" class="list-group" style="display: none;"></ul>
      </div>
      <button type="submit" class="btn">Search</button>
    </form>
    <!-- Recommendations Section -->
    {% if recommendations %}
      <hr class="bg-secondary">
      <h2 class="mt-4">Recommended Courses:</h2>
      <ul class="list-group">
        {% for course in recommendations %}
          <li class="list-group-item">
            <h4>{{ course.course_title }}</h4>
            <p>
              <a href="{{ course.course_url }}" target="_blank">
                {{ course.course_url }}
              </a>
            </p>
          </li>
        {% endfor %}
 </ul>
 {% elif recommendations is defined and recommendations|length == 0 %}
 <p class="text-center text-warning mt-4">
```

```
No recommendations found. Please try a different query.
 </p>
 {% endif %}
 </div>
  <!-- jQuery for AJAX suggestions -->
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
  <script>
    $(function() {
      const $suggestions = $('#suggestions');
      $('#query').on('keyup', function() {
        const q = $(this).val().trim();
        if (q.length > 1) {
          $.getJSON('/suggest', { q }, function(data) {
            if (data.length) {
              $suggestions.html(
                data.map(item =>
                  <li class="list-group-item suggestion-item">${item}</li>
                ).join('')
              ).show();
            } else {
              $suggestions.hide();
            }
          });
        } else {
          $suggestions.hide();
        }
      });
      $(document).on('click', '.suggestion-item', function() {
        $('#query').val($(this).text());
        $suggestions.hide();
      });
      $(document).click(function(e) {
        if (!$(e.target).closest('.position-relative').length) {
          $suggestions.hide();
        }
      });
    });
  </script>
</body>
</html>
```

5.1.3 Backend

Recommender code

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import pandas as pd

def get_recommendations(user_input, courses_df):
   # Combine course title and description into one text string.
         courses_df['combined'] = courses_df['course_title'].fillna('') + ' ' +
courses_df['course_description'].fillna('')
   documents = courses_df['combined']

   # TF-IDF vectorization
   vectorizer = TfidfVectorizer(stop_words="english")
   tfidf_matrix = vectorizer.fit_transform(documents)

   # Vectorize the user input and compute cosine similarity
   query_vec = vectorizer.transform([user_input])
   similarities = cosine_similarity(query_vec, tfidf_matrix).flatten()

   top_indices = similarities.argsort()[-5:][::-1]
   return courses_df.iloc[top_indices][['course_title', 'course_url']].to_dict(orient='records')
```

app.py

```
from flask import Flask, render_template, request, jsonify
import pandas as pd
from model.recommender import get_recommendations

app = Flask(_name_)

# Load dataset (make sure the CSV has been processed with normalized columns)
courses_df = pd.read_csv("data/courses.csv")
# (Optional) Normalize column names if needed:
courses_df.columns = [col.strip().lower() for col in courses_df.columns]
@app.route("/", methods=["GET", "POST"])
def index():
   recommendations = []
   if request.method == "POST":
      user_input = request.form["query"]
      recommendations = get_recommendations(user_input, courses_df)
```

```
    return render_template("index.html", recommendations=recommendations)

# Suggestion endpoint for autocomplete
@app.route("/suggest", methods=["GET"])
def suggest():
    query = request.args.get("q", "").strip().lower()
    suggestions = []
    if query:
        # Here we use a simple substring match on the course title.
        # You can refine this NLP technique if needed.
        matched = courses_df[courses_df['course_title'].str.lower().str.contains(query, na=False)]
        suggestions = matched['course_title'].drop_duplicates().head(5).tolist()
    return jsonify(suggestions)

if _name_ == "_main_":
    app.run(debug=True)
```

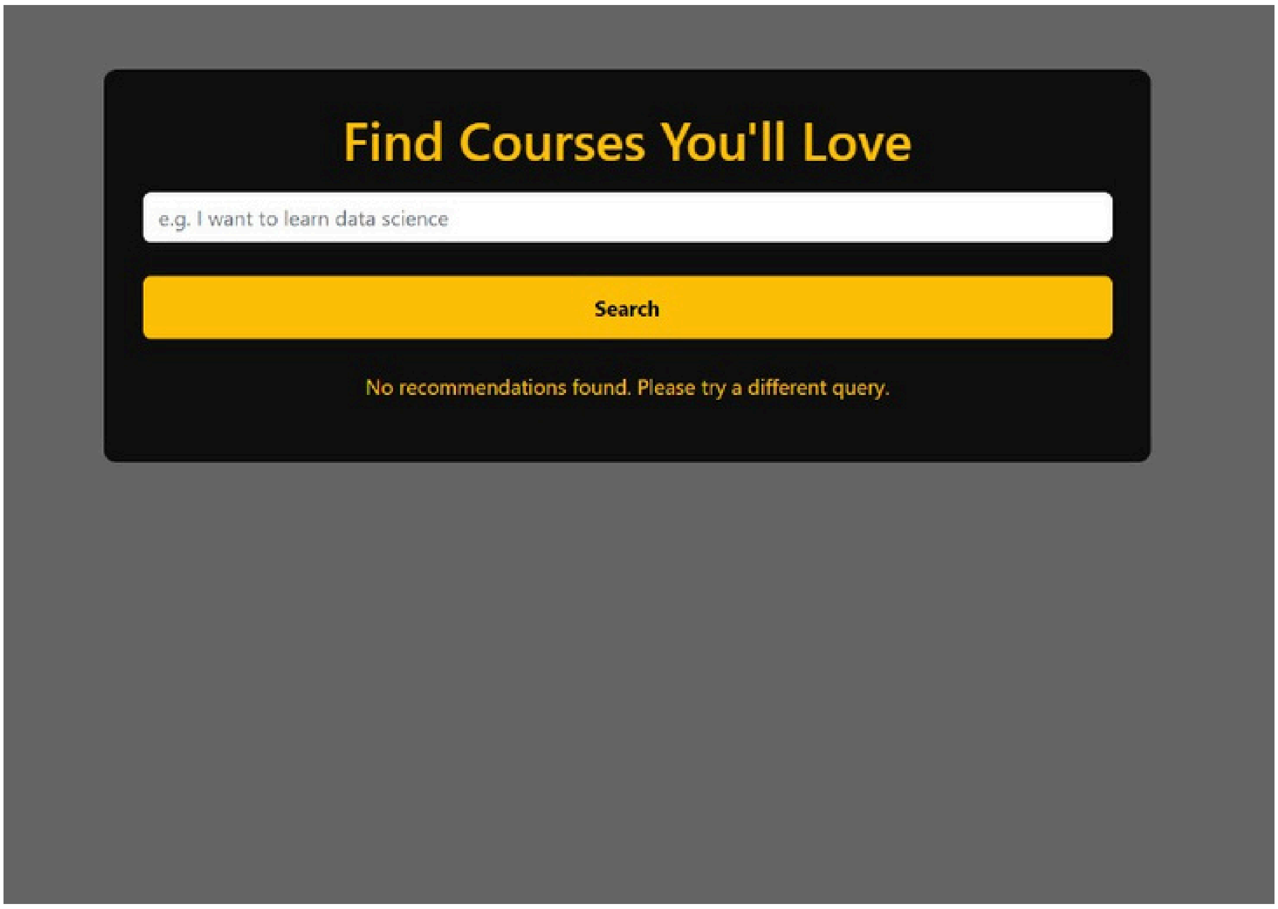
Download courses

```
import os
from kaggle.api.kaggle_api_extended import KaggleApi
import zipfile
import pandas as pd
# Set up your Kaggle API credentials (optional if you already have kaggle.json)
os.environ['KAGGLE_USERNAME'] = "koludhanesh"  # Replace with your username
os.environ['KAGGLE_KEY'] = "b3062e26d0a0931e7eea409750dc88c2"  # Replace with your
key

# Initialize the Kaggle API
api = KaggleApi()
api.authenticate()

# Define the dataset path (valid dataset)
dataset = "tianyimasf/coursera-course-dataset"  # Valid dataset path

# Download and unzip the dataset
try:
    api.dataset_download_files(dataset, path="data", unzip=True)
    print("✅ Dataset downloaded and extracted successfully.")
except Exception as e:
    print(f"❌ Error occurred while downloading dataset: {e}")
```

```
# Load and preview the dataset
try:
    df = pd.read_csv("data/coursera_courses.csv")
    print(df.head())  # Show first 5 records to verify data
except Exception as e:
    print(f"❌ Error occurred while loading the dataset: {e}")

# Optional: Save the cleaned version of the dataset (if needed)
try:
    df.to_csv("data/courses.csv", index=False)
    print("✅ Dataset saved as data/courses.csv")
except Exception as e:
    print(f"❌ Error occurred while saving the dataset: {e}")
```

5.2 Experimental Results

Home Page



No recommendations found. Please try a different query.

Fig. 5.2.1 Home Page

While Searching

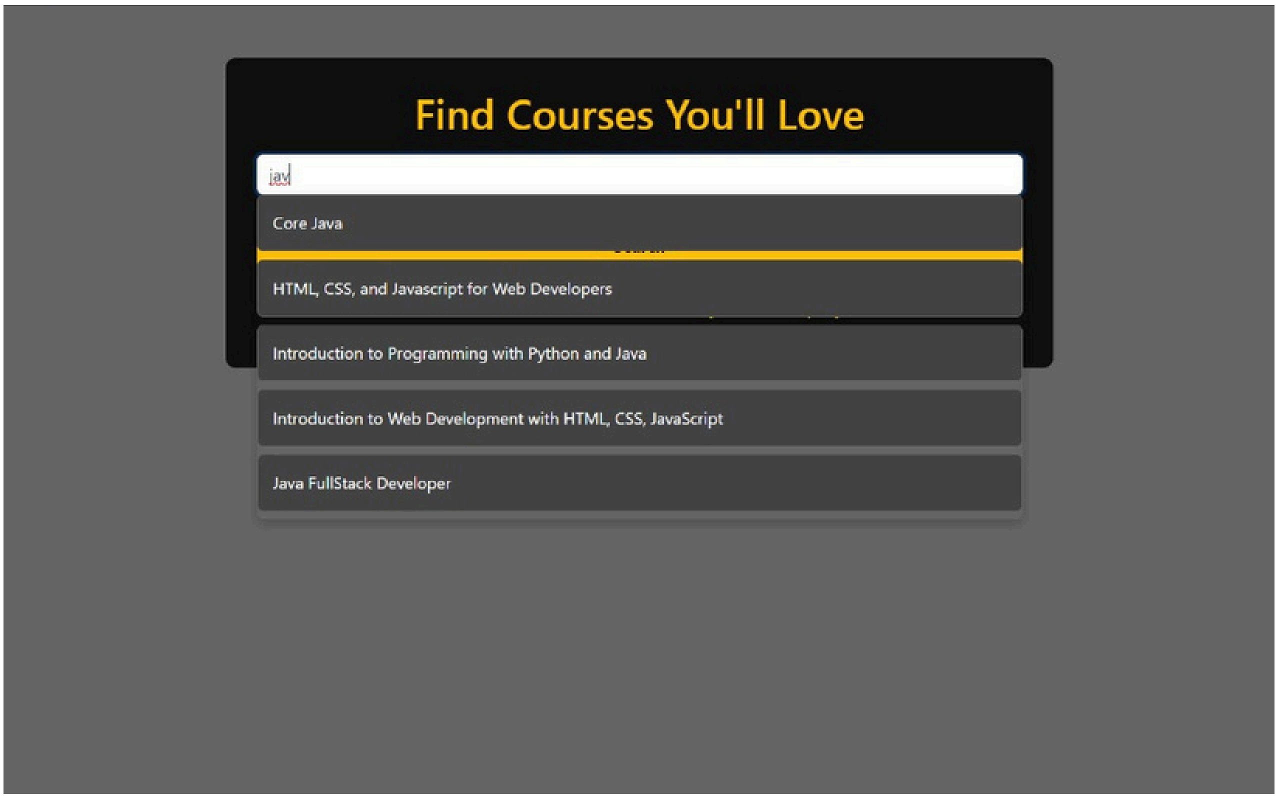
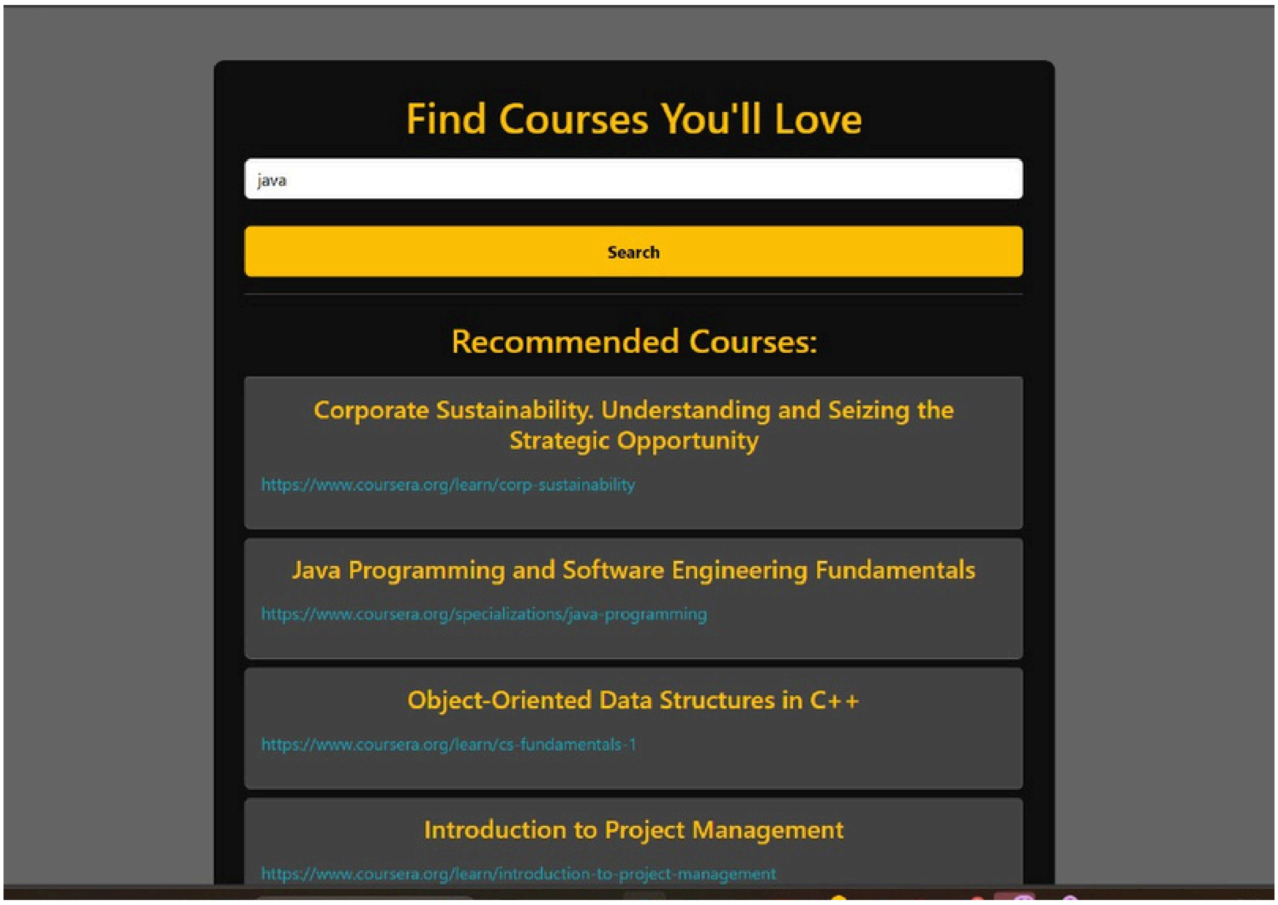

Find Courses You'll Love
jav
Core Java
HTML, CSS, and Javascript for Web Developers
Introduction to Programming with Python and Java
Introduction to Web Development with HTML, CSS, JavaScript
Java FullStack Developer

Fig. 5.2.2 While Searching

Showing Results



Recommended Courses:

Corporate Sustainability. Understanding and Seizing the Strategic Opportunity

https://www.coursera.org/learn/corp-sustainability

Java Programming and Software Engineering Fundamentals

https://www.coursera.org/specializations/java-programming

Object-Oriented Data Structures in C++

https://www.coursera.org/learn/cs-fundamentals-1

Introduction to Project Management

https://www.coursera.org/learn/introduction-to-project-management

Fig. 5.2.3 Showing Results

CHAPTER 6
CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

The creation of a Course Recommender System is an important milestone in tackling the vast quantity of learning material on online learning platforms. Utilizing content-based filtering and Natural Language Processing (NLP) methods like TF-IDF vectorization and cosine similarity, this system offers learners precise, effective, and relevant course recommendations based on their keywords or interests.

The system works without needing historical user interaction data, which makes it suitable for cold-start scenarios where user data is limited or non-existent. It is also transparent, interpretable, and easily extensible, which makes it a prime candidate for deployment in academic settings as well as commercial e-learning platforms.

The modular structure — data preprocessing, vectorization, similarity scoring, and recommendation display — makes each module upgradable or replaceable independently, providing high scalability and flexibility.

This project has been able to prove that even simple machine learning methods, when used properly in a well-designed dataset, can produce useful and effective results in personalized learning.

6.2 Future Scope

To enhance the system's performance, usability, and adaptability, several improvements can be incorporated in future versions:

Hybrid Recommendation Models
Combine collaborative filtering with content-based methods to personalize recommendations based on user interactions (clicks, enrollments, ratings) and user profiles.

User Profiling & Behavior Tracking
Maintain user profiles with long-term preferences and learning patterns, enabling better personalization and trend analysis.

Deep Learning-Based NLP
Integrate models like BERT, GPT, or SBERT for improved semantic understanding of user queries and course descriptions, resulting in better contextual recommendations.

Feedback Loop Integration
Include real-time feedback mechanisms (like/dislike, rating) to continuously improve recommendations through reinforcement learning or adaptive filtering.

Multilingual Support
Allow course recommendations and inputs in multiple languages to reach a broader audience, especially non-English speakers.

Visual & Video Data Incorporation
Extend the recommendation logic to consider course thumbnails, preview videos, and instructor reviews using computer vision or video summarization techniques.

Cloud Deployment
Deploy the system using cloud services (AWS, GCP, or Azure) to make it accessible globally and handle real-time queries at scale.

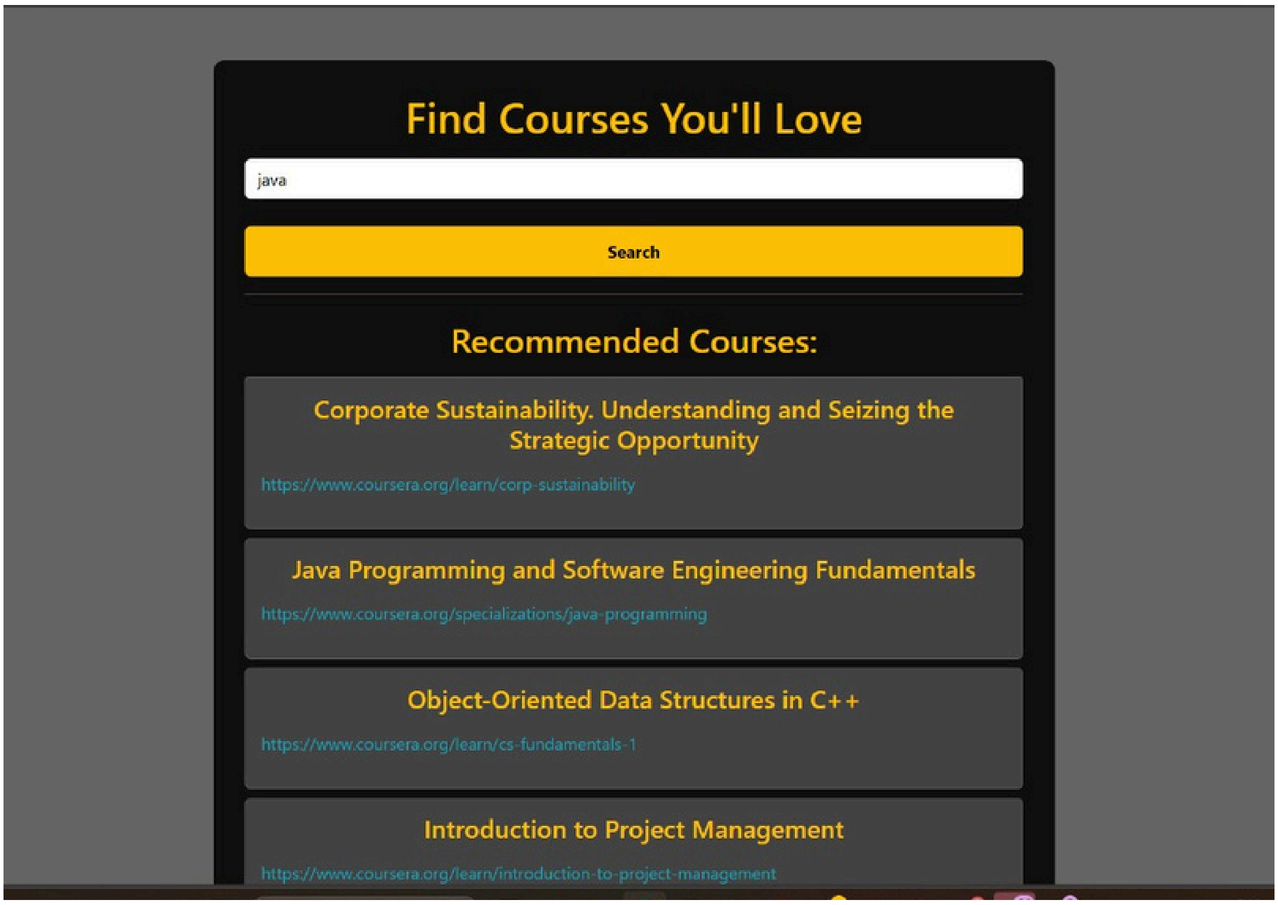
Mobile App Integration
Create mobile app versions for Android and iOS to make recommendations more accessible and to leverage native device features (e.g., notifications, voice input).

CHAPTER 7
REFERENCES

- https://scikit-learn.org
- Flask Documentation
- Research papers on Course Recommender Systems
- MySQL Official Docs

CHAPTER 8
APPENDIX

8.1 Snapshot of the Result



Find Courses You'll Love

java

Search

Recommended Courses:

Corporate Sustainability. Understanding and Seizing the Strategic Opportunity

https://www.coursera.org/learn/corp-sustainability

Java Programming and Software Engineering Fundamentals

https://www.coursera.org/specializations/java-programming

Object-Oriented Data Structures in C++

https://www.coursera.org/learn/cs-fundamentals-1

Introduction to Project Management

https://www.coursera.org/learn/introduction-to-project-management

Fig. 8.1.1 Snapshot of the result

8.2 Software Installation /Dependencies

To run this project, the following tools and libraries were installed:

- Python 3.10+
- Flask (pip install flask)
- Scikit-learn (pip install scikit-learn)
- Pandas (pip install pandas)
- NumPy (pip install numpy)

3

8.3 Pseudo Code

```
1. Load dataset (CSV)
2. Combine course title and description
3. Use TF-IDF Vectorizer to convert text to numerical features
4. Accept user query and vectorize it
5. Compute cosine similarity between query vector and course vectors
6. Return top 5 most similar courses
```