ENHANCING STUDENT ENGAMENT IN ENGINEERING COURSES THROUGH PERSONALIZED BOOK RECOMMENDATIONS

A Capstone Project report submitted

in partial fulfillment of requirement for the award of degree

BACHELOR OF TECHNOLOGY

in

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE

by

ASHWINI CHIDURALA	**(2103A52028)**
SAIPRIYA TAKKARSU	**(2103A52110)**
SHASHANKA BASANI	**(2103A52005)**
MADHUSREE KORUKANTI	**(2103A52090)**
PRANAVI MULASTAM	**(2203A52L02)**

Under the guidance of

Ms Faiza Iram

Associate Professor, School of CS&AI.

SR University, Ananthsagar,Warangal,Telagnana-50637

SR University

Ananthasagar, Warangal.

CERTIFICATE

This is to certify that this project entitled **"**ENHANCHING STUDENT ENGAGEMENT IN ENGINEERING COURSES THROUGH PERSONALIZED BOOK RECOMMENDATIONS" is the bonafied work carried out by **T.Saipriya , CH.Ashwini , B.Shashanka , K.Madhusree , M.Pranavi** as a Capstone Project for the partial fulfillment to award the degree **BACHELOR OF TECHNOLOGY** in **School of Computer Science and Artificial Intelligence** during the academic year 2024-2025 under our guidance and Supervision.

MS FAIZA IRAM

Associate professor,

School of CS&AI, SR University
Ananthasagar, Warangal.

Dr. M.Sheshikala

Professor & Head,

School of CS&AI, SR University
Anathasagar,Warangal

Reviewer-1

Name:

Designation:

Signature:

Reviewer-2

Name:

Designation:

Signature:

ACKNOWLEDGEMENT

We owe an enormous debt of gratitude to our Capstone project guide **MS FAIZA IRAM, Assoc Prof** as well as Head of the School of CS&AI , **Dr. M.Sheshikala, Professor** and Dean of the School of CS&AI, **Dr.Indrajeet Gupta Professor** for guiding us from the beginning through the end of the Capstone Project with their intellectual advices and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us to the right direction.

We express our thanks to project co-ordinators **Mr. Sallauddin Md, Asst. Prof., and R.Ashok Asst. Prof.** for their encouragement and support.

Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

ABSTRACT

The System provides tailored reading recommendations for academic purpose. The project, named "Enhanching Student Engagement in Engineering Courses through Personalized Book Recommendations", covers a wide range of book genres and is intended to benefit students in a number of academic subjects, such as computer science, electronics, artificial intelligence. It makes use of two main approaches: content-based filtering, which proposes books based on a students search history and preferences, and collaborative filtering, which makes book recommendations based on the tastes of students by combining various methods. In addition, the system tackles issues like data sparsity and the cold-start problem, which involves making recommendations for new users. Metrics like as recall and precision are used to evaluate the system's efficacy in making suggestions that correspond with students' academic level

TABLE OF CONTENTS

INTRODUCTION

The project "Enhancing Student Engagement in Engineering Courses through Personalized Book Recommendations" aims to improve learning experiences for engineering students by leveraging personalized content delivery. In today's interdisciplinary academic environment, students from diverse fields such as Computer Science, Electronics, and Artificial Intelligence require tailored resources that align with their academic and personal interests. This system addresses that need by offering book recommendations across a wide variety of genres using advanced filtering techniques. Through the combination of collaborative filtering, which leverages the preferences of similar users, and content-based filtering, which matches resources to individual search histories and preferences, the platform delivers highly personalized recommendations. This approach not only enhances academic engagement but also addresses key challenges such as the cold-start problem for new users and data sparsity. By focusing on precision and relevance, the system aims to significantly contribute to student success and engagement across multiple disciplines. Due to the demanding and occasionally esoteric nature of the subject content, engineering education frequently struggles to keep students interested. Even if they are good at communicating technical knowledge, traditional teaching approaches might not always pique students' attention or accommodate their various learning requirements. Reduced motivation and less than ideal learning results may result from this Personalized learning techniques have become a viable way to deal with this problem. By customizing educational experiences to each student's choices, interests, and needs, personalized learning creates a more stimulating and productive learning environment. Integrating tailored book recommendations into engineering courses is one creative method. Students' varied interests and the regimented curriculum can be reconciled through tailored book choices. Teachers can increase student engagement and promote a deeper comprehension of engineering subjects by recommending reading resources that complement the course objectives and the students' individual interests. In addition to fostering academic progress, this method inspires pupils to love studying for the rest of their lives. The effect of tailored book recommendations on engineering course participation is examined in this paper. We start by going over the body of research on the advantages of individualized learning in higher education

1.1 EXISTING SYSTEM

1. Traditional Recommender Systems

Traditional recommender systems are widely used to suggest books and other resources. These systems generally fall into two categories: collaborative filtering and content-based filtering.

Collaborative Filtering: This method makes recommendations based on the preferences and behaviors of similar users. For example, if a student has read and liked certain books, the system will recommend books that other students with similar preferences have enjoyed. However, collaborative filtering often struggles with the cold-start problem, where it cannot make accurate recommendations for new users due to lack of data.

Content-Based Filtering: This approach recommends books based on the characteristics of the books themselves and the user's past interactions with similar content. For instance, if a student has shown interest in books on machine learning, the system will suggest other books related to this topic. While effective in certain scenarios, content-based filtering can become limited by a narrow focus, potentially missing out on broader interdisciplinary resources.

2. Learning Management Systems (LMS)

Many educational institutions use Learning Management Systems (LMS) like Moodle, Blackboard, or Canvas to deliver course content and manage student interactions. These platforms often include basic recommendation features to suggest additional readings or resources based on the course content. However, these recommendations are generally not personalized to individual student interests and are often limited to the materials available within the LMS.

3. Personalized Learning Platforms

Some educational platforms, like Coursera or edX, offer personalized learning experiences by recommending courses and materials based on a student’s learning history and interests. These platforms use a combination of collaborative and content-based filtering techniques. While effective for online courses, these systems are often not integrated into traditional engineering curricula and may not provide specific book recommendations relevant to the course content.

While there are various methods currently used to recommend books and enhance student engagement in engineering courses, each has its limitations. Traditional recommender systems face challenges like the cold-start problem and limited scope. LMS and personalized learning platforms offer some degree of personalization but may not be fully integrated into the specific requirements of engineering curricula. Library systems and social media platforms provide a wealth of resources but often lack the targeted relevance needed for academic success. Faculty and peer recommendations are valuable but not systematically personalized.

1.1 PROPOSED SYSTEM

The proposed system aims to increase student engagement in engineering courses by providing personalized book recommendations. By leveraging advanced learning algorithms, the system tailors suggestions to meet the unique needs and interests of individual students. The project incorporates two primary learning algorithms: content-based learning and collaborative-based learning.

Overview

1. **Objective**: To enhance student engagement and learning outcomes in engineering courses by recommending books that are most relevant to each student's preferences and academic needs.
2. **Scope**: The system will be integrated into the existing educational platform, providing students with personalized book recommendations based on their interests, past behavior, and peer interactions.

Key Components of the Proposed System

1. **User Data Collection**:
   - **Profile Information**: Basic details such as academic year, courses enrolled, and areas of interest.
   - **Activity Logs**: Data on previously accessed books, reading patterns, course materials engaged with, and other relevant academic activities.
   - **Feedback Mechanism**: Ratings and reviews provided by students on recommended books to continuously refine the recommendation process.
2. **Learning Algorithms**:
   - **Content-Based Learning**: This algorithm focuses on the attributes of books and matches them with the preferences and past behavior of students. It uses features such as book titles, authors, keywords, and descriptions.
   - **Collaborative-Based Learning**: This algorithm makes recommendations based on the preferences of similar users. It identifies patterns and similarities among students' choices to suggest books that peers with similar interests and behaviors have found useful.

3. **Recommendation Engine**:
   - **Hybrid Model**: Combines the strengths of content-based and collaborative-based learning algorithms to provide more accurate and diverse recommendations.
   - **Real-Time Processing**: Continuously updates recommendations as new data is collected, ensuring relevance and timeliness.
4. **User Interface**:
   - **Dashboard**: An intuitive interface where students can view personalized book recommendations, track their reading progress, and provide feedback.
   - **Search and Filter Options**: Allows students to search for specific books and filter recommendations based on various criteria (e.g., topic, author, difficulty level).
5. **Integration with Existing Systems**:
   - **Learning Management System (LMS)**: Seamlessly integrates with the institution's LMS to pull relevant data and provide recommendations within the same environment.
   - **Library Database**: Connects with the institution's library database to ensure availability of recommended books.

Implementation Details

1. **Data Processing**:
   - **Data Collection**: Gather data from various sources, including student profiles, activity logs, and feedback forms.
   - **Data Cleaning and Preprocessing**: Ensure the accuracy and completeness of collected data through validation and preprocessing techniques.
2. **Algorithm Development**:
   - **Content-Based Algorithm**: Develop a model to analyze book attributes and match them with student preferences. This involves techniques like TF-IDF (Term Frequency-Inverse Document Frequency) for keyword extraction and similarity measures.
   - **Collaborative-Based Algorithm**: Implement a collaborative filtering model using techniques like user-based and item-based collaborative filtering to identify patterns in user behavior and preferences.
3. **System Integration**:
   - **APIs and Middleware**: Develop APIs to facilitate communication between the recommendation engine, LMS, and library database.
   - **User Authentication and Authorization**: Ensure secure access to the system through authentication mechanisms and role-based access control.

4. **Testing and Validation**:
   - **Pilot Testing**: Conduct pilot tests with a small group of students to gather initial feedback and refine the system.
   - **Performance Evaluation**: Evaluate the performance of the recommendation engine using metrics such as precision, recall, and user satisfaction.

Benefits

1. **Personalized Learning Experience**: Tailored book recommendations help students find resources that are most relevant to their needs, enhancing their learning experience.
2. **Increased Engagement**: By providing books that align with their interests and academic goals, students are more likely to engage deeply with the material.
3. **Improved Academic Performance**: Access to high-quality and relevant books can support better understanding and retention of course content, potentially leading to improved academic outcomes.
4. **Efficient Resource Utilization**: Helps students make better use of available library resources by guiding them to books that are most beneficial for their studies.

2. LITERATURE SURVEY

Reviewing scholarly articles, business reports, and other pertinent sources would be part of a literature evaluation on personalized book recommendation system. This would help one gain an understanding of the state of the field's advancements, difficulties, and trends.

[1]. Fredricks, J. A., Blumenfeld, P. C., & Paris, A. H. (2004). "School Engagement: Potential of the Concept, State of the Evidence."
This paper explores the concept of student engagement and its various dimensions—behavioral, emotional, and cognitive. The authors emphasize the importance of engagement in achieving academic success and highlight the need for personalized approaches to maintain high levels of engagement. The study provides a foundational understanding of engagement that is crucial for developing personalized educational tools.

[2]. Prince, M. (2004). "Does Active Learning Work? A Review of the Research."
Prince reviews the effectiveness of active learning strategies in engineering education. He finds that active learning significantly improves student engagement and learning outcomes compared to traditional lecture-based methods. This paper underscores the potential benefits of personalized book recommendations as a form of active learning that can cater to individual student interests and enhance engagement.

[3]. Felder, R. M., & Brent, R. (2005). "Understanding Student Differences."
Felder and Brent discuss the diversity of student learning styles and the impact of these differences on educational outcomes. They argue for the necessity of personalized learning approaches to accommodate various learning preferences. This research supports the idea that personalized book recommendations can help address individual differences and improve engagement in engineering courses.

[4]. Ricci, F., Rokach, L., & Shapira, B. (2015). "Recommender Systems Handbook."
This comprehensive handbook provides an overview of different recommender system techniques, including collaborative filtering, content-based filtering, and hybrid methods. The authors discuss the strengths and limitations of each approach, providing valuable insights for developing a hybrid recommendation system tailored to educational contexts.

[5].Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). "Item-based Collaborative Filtering Recommendation Algorithms."
Sarwar et al. explore item-based collaborative filtering, a technique that recommends items based on the similarities between items rather than users. This approach helps mitigate the cold-start problem to some extent. The paper is relevant for designing a recommendation system that can provide accurate book recommendations to engineering students with limited data.

[6]. Pazzani, M. J., & Billsus, D. (2007). "Content-Based Recommendation Systems."
This paper delves into content-based recommendation systems, which suggest items based on the content characteristics and user profiles. The authors highlight the advantages of content-based systems in providing personalized recommendations and discuss potential limitations. The findings are instrumental in designing a system that leverages content-based filtering to recommend relevant books.

[7]. Burke, R. (2002). "Hybrid Recommender Systems: Survey and Experiments."
Burke surveys various hybrid recommender systems that combine collaborative and content-based filtering techniques. The paper provides a detailed analysis of different hybrid models and their performance, offering guidance on how to create a robust recommendation system for educational purposes.

[8].Drachsler, H., Hummel, H. G. K., & Koper, R. (2008). "Personal recommender systems for learners in lifelong learning networks: the requirements, techniques and model."
Drachsler et al. discuss the design and implementation of personal recommender systems in educational settings. They emphasize the importance of personalized recommendations in enhancing learning experiences and outcomes. This paper provides a theoretical framework and practical insights for developing a recommendation system tailored to engineering students.

[9]. Chen, G., de Nart, D., & Bouchet, F. (2017). "Personalized Recommendation in Educational Contexts: A Review and Future Perspectives."
Chen et al. review various personalized recommendation approaches in educational contexts and their impact on student engagement and learning outcomes. The authors identify key challenges and opportunities in implementing personalized recommendation systems in education, offering valuable perspectives for the

current project

[10]. Macedo, J. A., Cardoso, A., & Lam, R. (2016). "Personalized Learning in Higher Education Using Recommendation Systems."

Macedo et al. explore the application of recommendation systems in higher education to personalize learning experiences. They discuss case studies and experimental results that demonstrate the effectiveness of personalized recommendations in improving student engagement and academic performance. This paper provides practical examples and evidence supporting the use of personalized book recommendations in engineering courses.

3. DESIGN

2.1 REQUIREMENT SPECIFICATION

Software requirements

Programming Language: Because of its large machine learning library and user-friendliness,we are using Python for project code development.

Development Environment: The main environment for development will be Google Colab. It provides free access to TPUs and GPUs, which can greatly accelerate machine learning activities including model training.

Machine Learning Libraries: To create and train machine learning models, such as Artificial Neural Networks (ANN) for illness prediction, use libraries such as scikit-learn, TensorFlow, and Keras.

Version Control: To facilitate smooth communication, track changes, and preserve the integrity of the codebase, GitHub will be utilized for version control.

Documentation: Detailed documentation, such as code explanations, API references, and installation instructions, can be created using word document and power point .

Hardware requirements

CPU: For effective computing, a laptop with a contemporary multicoreprocessor. For the best performance, look for AMD Ryzen series or Intel Core i5 or i7 processors.
RAM: To properly manage data processing and model training, using 8GB to 16GB of RAM. When working with larger datasets and complicated models, more RAM could be useful.
Storage: For faster data access and improved overall performance, ideally choosed a laptop with solid-state drive (SSD) storage. It is advised to have 256GB or more of storage space in order to hold project files and datasets.
GPU: Having a dedicated GPU on the laptop can be helpful for local development and testing even though Google Colab will enable access to GPUs for model training. GPUs from NVIDIA, like the GeForce

2.2 UML DIAGRAMS

Unified Modeling Language (UML) is a standardized modeling language used to visualize, specify, construct, and document the artifacts of a software system. UML provides a set of graphical notation techniques to create visual models of object-oriented software systems.

1.USE CASE: A Use Case Diagram is a type of behavioral diagram defined by the Unified Modeling Language (UML) that represents the functional requirements of a system and its interaction with external entities, known as actors. It provides a high-level view of the system's functionality and the various ways users (actors) can interact with it.

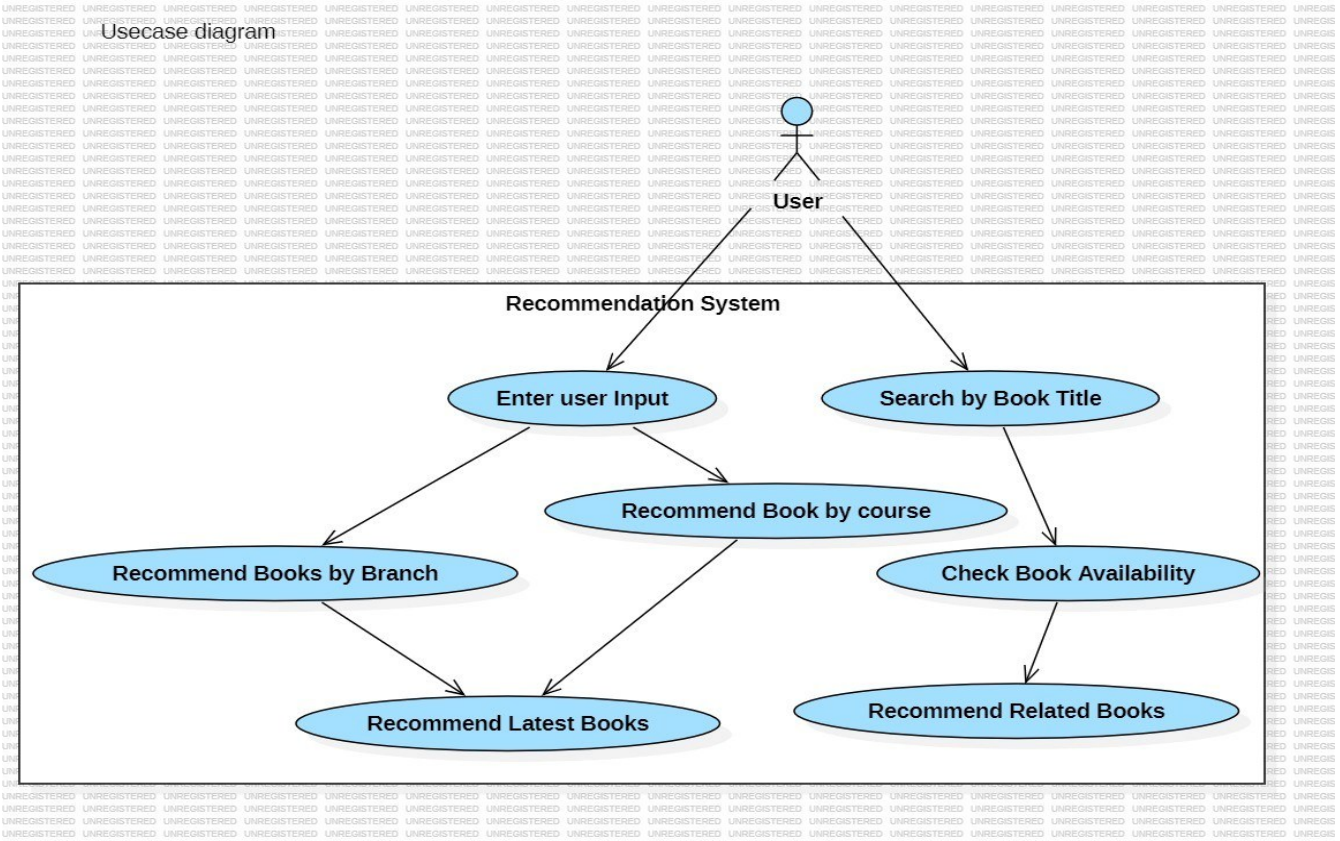

FIG 3.2.1 USE CASE DIAGRAM

2. STATE CHART : A State Chart Diagram, also known as a State Machine Diagram, is a type of behavioral diagram in UML that depicts the dynamic behavior of an object by showing its states and the transitions between those states. It is particularly useful for modeling the lifecycle of an object and illustrating how it responds to events.

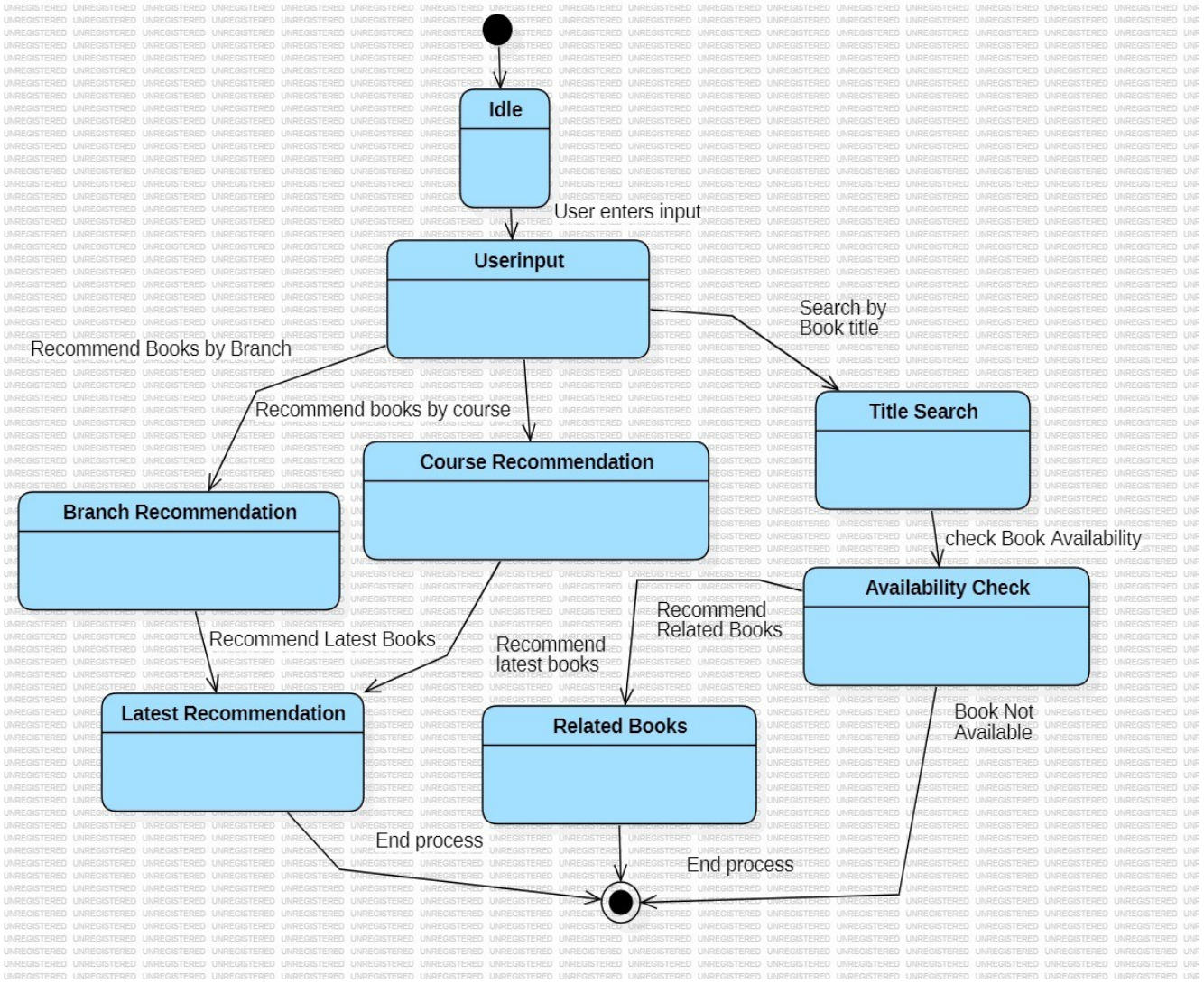

FIG 3.2.2 STATE CHART DIAGRAM

3. SEQUENCE DIAGRAM : A Sequence Diagram is a type of interaction diagram in UML that shows how objects interact in a particular scenario of a use case by emphasizing the sequence of messages exchanged between them. Sequence diagrams provide a visual representation of the chronological order of messages and are particularly useful for detailing the interactions and behavior of a system over time.

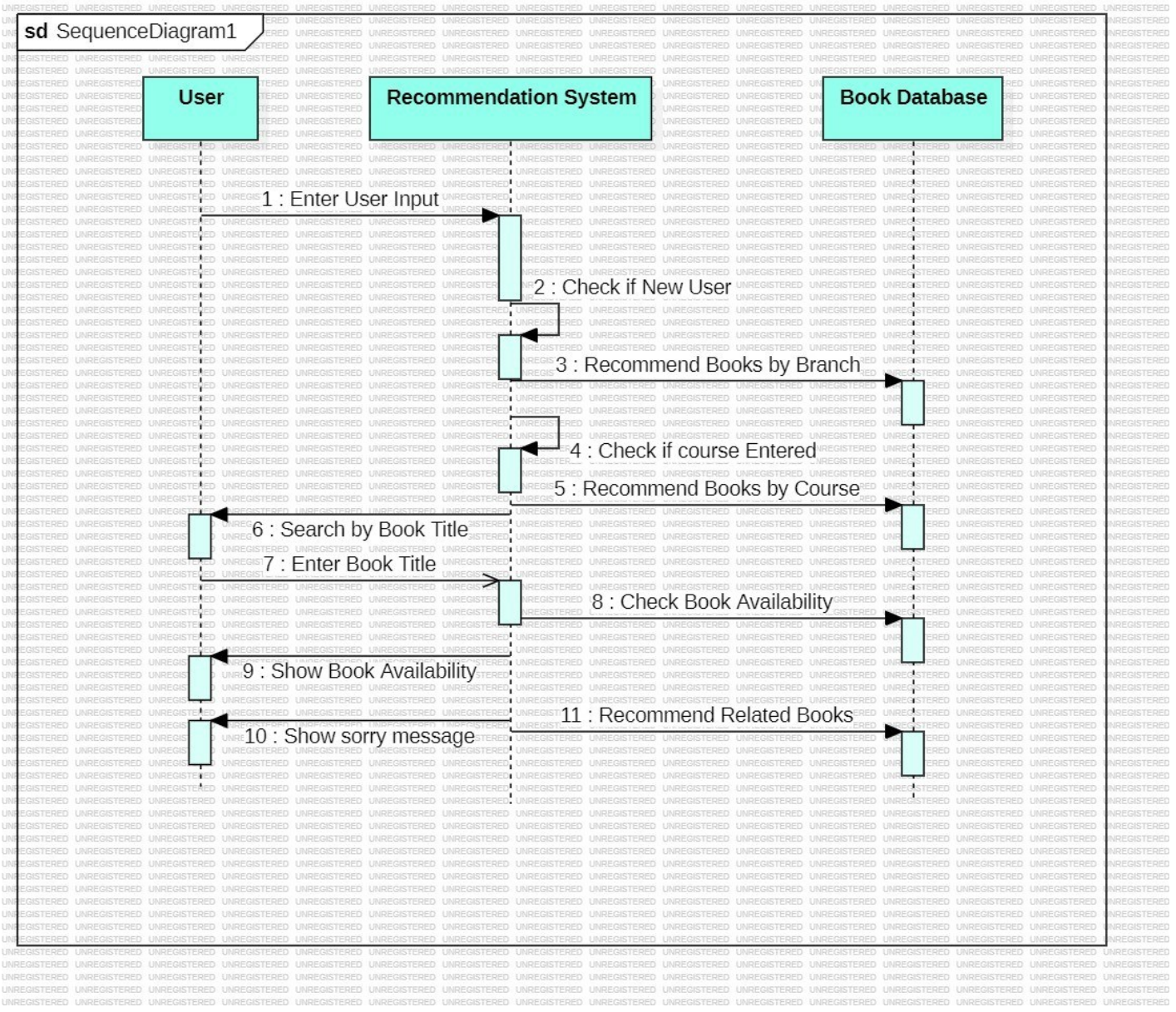

FIG 3.2.3 SEQUENCE DIAGRAM

4. DEPLOYMENT DIAGRAM : A Deployment Diagram is a type of structural diagram in UML that shows the physical arrangement of hardware and software components in a system. It illustrates how software components (artifacts) are deployed on hardware components (nodes) and how these nodes are connected. Deployment diagrams are particularly useful for visualizing the physical distribution of a system and for planning its infrastructure**.**

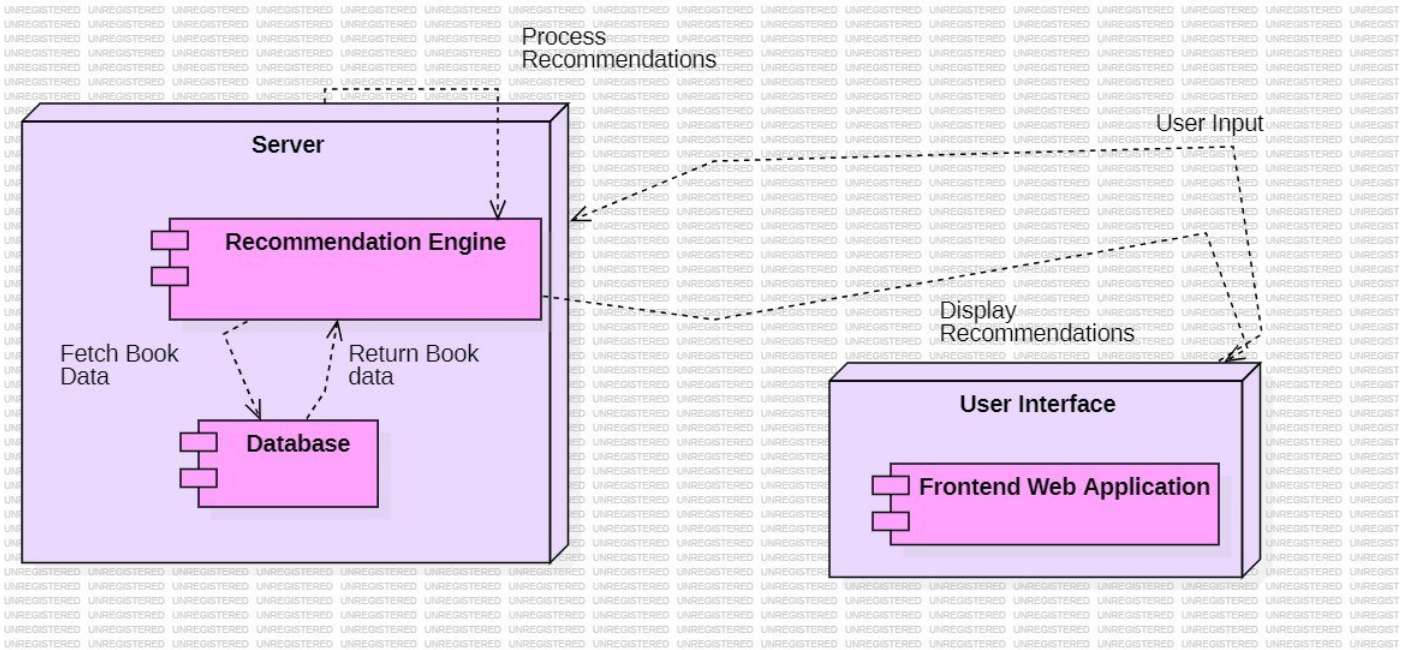

FIG 3.2.4 DEPLOYMENT DIAGRAM

5.CLASS DIAGRAM : A Class Diagram is a type of structural diagram in UML that describes the structure of a system by showing its classes, their attributes, methods (operations), and the relationships among the objects. Class diagrams are essential for modeling the static view of an application and are commonly used to represent the blueprint of an object-oriented system.

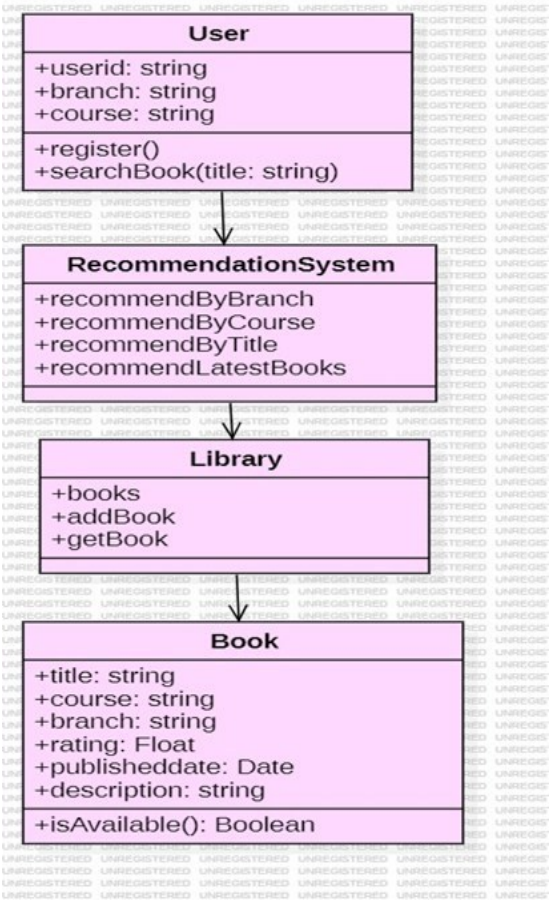

FIG 3.2.5 CLASS DIAGRAM

6.ACTIVITY DIAGRAM : An Activity Diagram is a type of behavioral diagram in UML that represents the workflow of a system or a process. It shows the sequence of activities, actions, and decisions that occur within a process, making it useful for modeling the dynamic aspects of a system. Activity diagrams provide a high-level view of the business logic and are particularly useful for describing complex processes and parallel activities.

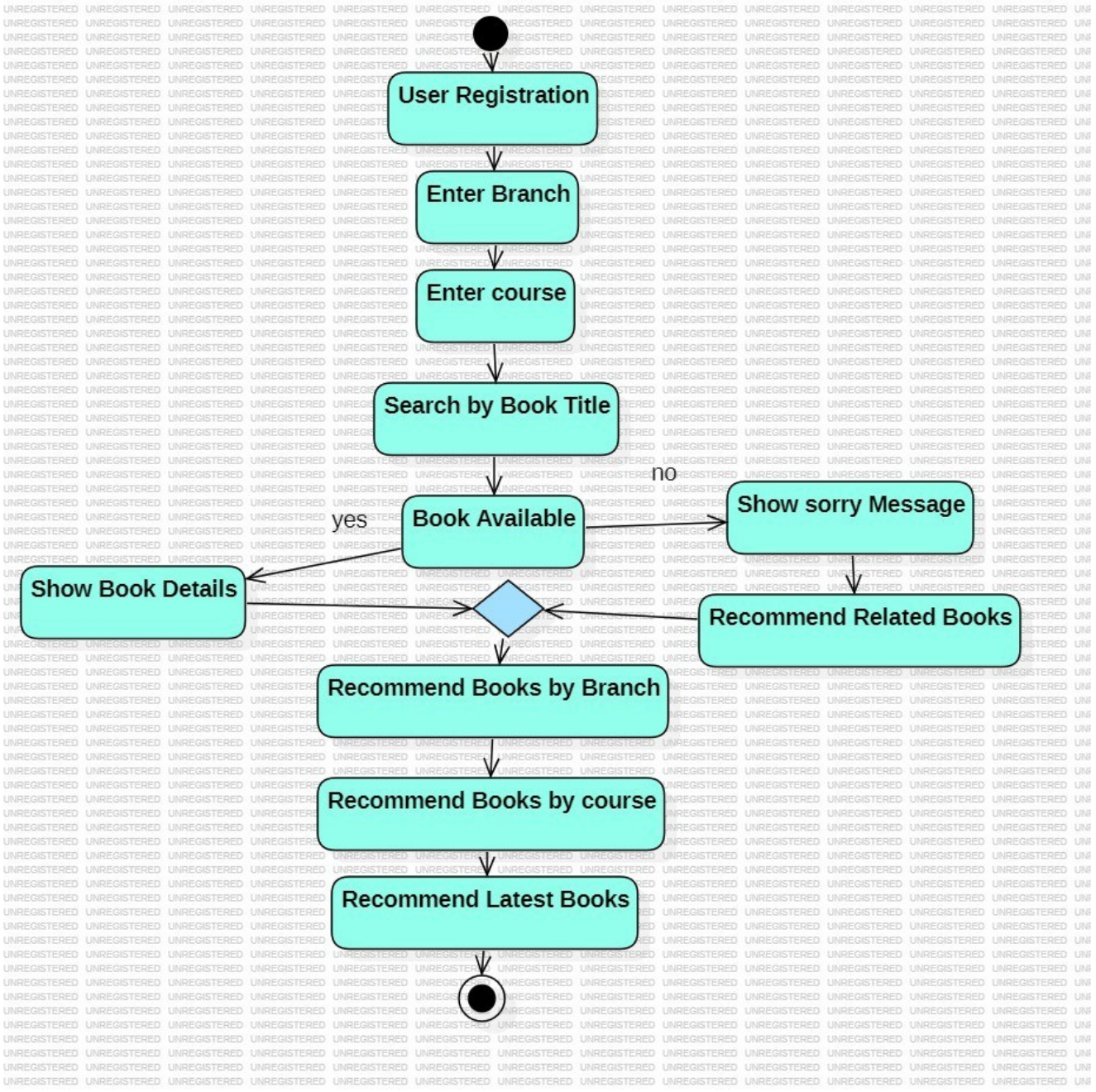

FIG 3.2.6 ACTIVITY DIAGRAM

3. IMPLEMENTATION

3.1 MODEL ARCHITECTURE

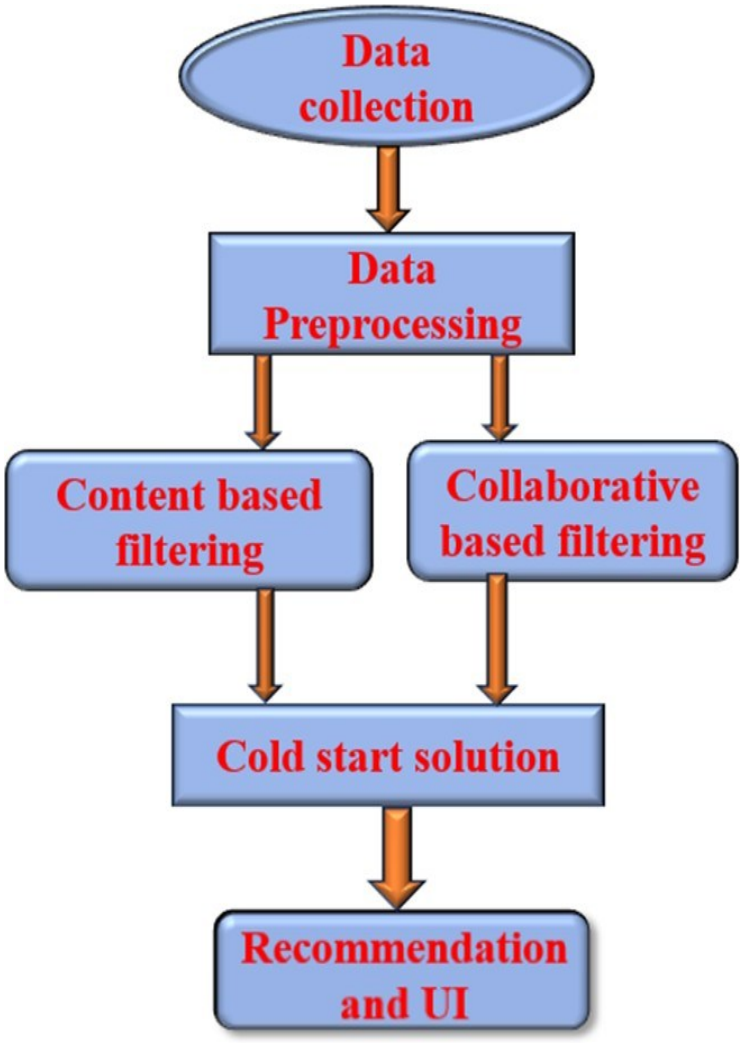

Fig:4.1.1 Model architecture

3.2 MODEL IMPLEMENTATION

This project involves creating a personalized book recommendation system for engineering courses using content-based learning and collaborative-based learning algorithms.

Phase 1: Problem Understanding and Requirement Analysis

1. **Define Objectives**:
   - Enhance student engagement in engineering courses through personalized book recommendations.
   - Use content-based and collaborative-based algorithms for recommending relevant books to students.
2. **Gather Requirements**:
   - Functional Requirements:
     - Personalized book recommendation engine.
     - Integration with the existing Learning Management System (LMS).
     - Feedback and rating system for books.
     - User interface for students to view and interact with recommendations.
   - Non-functional Requirements:
     - Scalability for handling large numbers of users.
     - Data security and privacy for student information.
3. **Identify Data Sources**:
   - Student profiles (demographic data, course enrollment).
   - Historical interaction data (books viewed, books read, ratings given).
   - Engineering course content (topics, course materials, book metadata).
   - Library database for book availability.

Phase 2: Data Collection and Preprocessing

1. **Data Collection**:
   - **Student Data**: Collect data on students such as courses enrolled, academic year, interests, reading history, and feedback from the LMS.
   - **Book Data**: Gather metadata for engineering books, including titles, authors, descriptions, and categories.
   - **Interaction Data**: Collect data on students' previous book interactions, including ratings, browsing history, and the time spent on books.

2. **Data Preprocessing**:
- **Handling Missing Values**: Impute or remove missing data (e.g., incomplete student profiles or missing book ratings).
- **Normalization and Standardization**: Normalize numerical attributes (e.g., book ratings) and standardize where necessary to ensure uniform data representation.
- **Categorical Encoding**: Convert categorical data (e.g., book genres or student interests) into numerical format using techniques like one-hot encoding or label encoding.
- **Data Integration**: Merge data from various sources (LMS, library database, feedback system) into a single, unified dataset.
3. **Data Cleaning**:
- Remove duplicates from the data (e.g., duplicate book entries or duplicate student profiles).
- Correct any inaccuracies or inconsistencies in the dataset (e.g., incorrect book metadata or student details).

Phase 3: Algorithm Development

1. **Content-Based Learning**:
- **Book Feature Extraction**: Extract relevant features from books such as titles, authors, categories, and keywords.
- **Similarity Calculation**: Implement a method (e.g., cosine similarity) to compare books based on their features and match them with students' past preferences.
- **Recommendation Generation**: Develop a recommendation engine that suggests books based on the similarity between the student's previous interactions and the available book features.
2. **Collaborative-Based Learning**:
- **User-Item Matrix**: Construct a user-item matrix where rows represent students and columns represent books. The matrix values are the ratings or interactions a student has with a book.
- **Collaborative Filtering**: Implement collaborative filtering using techniques like **user-based** (find similar users) or **item-based** (find similar books) filtering. This method recommends books based on similar preferences or behaviors of other students.
- **Hybrid Model**: Combine content-based and collaborative-based approaches to create a hybrid model that leverages both content similarity and peer preferences for more accurate recommendations.
3. **Testing and Tuning**:
- **Evaluation Metrics**: Use evaluation metrics like **precision**, **recall**, **F1-score**, and **root mean squared error (RMSE)** to assess the performance of the recommendation system.
- **Algorithm Tuning**: Fine-tune parameters (e.g., similarity threshold, collaborative filtering settings) to

improve the accuracy and relevance of the recommendations.

Phase 4: System Development and Integration

Frontend Development: Create a user interface (UI) that allows students to view their book recommendations, rate books, and interact with the system. This can be built using web technologies (e.g., HTML,css)

User Interface:

- **Recommendation Dashboard**: Develop a dashboard where students can see a list of recommended books based on their preferences and academic needs.
- **Search and Filter**: Provide search and filter options for students to explore books by category, author, or topic.
- **Feedback Mechanism**: Allow students to rate or review books to further refine future recommendations.

Phase 5: Testing and Validation

1. **User Testing**:
   - Conduct pilot tests with a small group of students to collect feedback on the recommendations and the user interface.
   - Gather insights on the system's usability, accuracy of book suggestions, and overall student engagement.
2. **Performance Testing**:
   - Test the system for scalability, ensuring that it can handle a large number of students and recommendations efficiently.
   - Validate the performance of the recommendation algorithms using the evaluation metrics.
3. **Bug Fixing and Refinements**:
   - Address any bugs, errors, or performance issues identified during testing.
   - Improve the recommendation logic based on user feedback and testing results.

Phase 6: Deployment and Maintenance

1. **Deployment**:
   - Deploy the recommendation system on the live educational platform or in a cloud environment (e.g., AWS, Azure) to ensure high availability and scalability.
   - Integrate the system into the student portal or LMS for easy access.
2. **Monitoring and Support**:

- Set up continuous monitoring to track the system's performance and user interactions.
- Provide ongoing support to address any issues or updates needed to the system.

3. **Feedback Loop**:
   - Continuously collect user feedback to refine and update the recommendation algorithms.
   - Update the book database regularly to ensure the system provides fresh and relevant book recommendations.

4. TESTING

4.1 TEST CASES

```
pip install pandas scikit-learn numpy surprise

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

# Load dataset
df = pd.read_csv('/content/cleaned_combined_books_dataset.csv')

# Display first few rows of the dataset
print(df.head())

# Fill missing values if any
df.fillna('', inplace=True)

print("Feature names in the dataset:")
print(df.columns)

print("Shape of the dataset (rows, columns):")
print(df.shape)

print("\nData types of features:")
print(df.dtypes)

print("Null values in each feature:")
print(df.isnull().sum())
```

```
num_duplicates = df.duplicated().sum()
print(f"\nNumber of duplicate rows: {num_duplicates}")

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Set a style for seaborn
sns.set(style="whitegrid")

# 1. Bar Plot for Categorical Features (e.g., 'Branch')
plt.figure(figsize=(10, 6))
sns.countplot(x='Branch', data=df, order=df['Branch'].value_counts().index)
plt.title('Distribution of Books by Branch')
plt.xticks(rotation=45)
plt.show()

# 2. Histogram for Numerical Features (e.g., 'Rating')
plt.figure(figsize=(10, 6))
sns.histplot(df['Rating'], bins=20, kde=True)
plt.title('Distribution of Ratings')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.show()

# 3. Box Plot for Numerical Feature (e.g., 'Rating')
plt.figure(figsize=(8, 6))
sns.boxplot(x='Rating', data=df)
plt.title('Box Plot of Ratings')
plt.show()

# Count the number of books for each branch
branch_counts = df['Branch'].value_counts()
```

```
# Create a pie chart
plt.figure(figsize=(8, 8))
plt.pie(branch_counts, labels=branch_counts.index, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of Books by Branch')
plt.axis('equal') # Equal aspect ratio ensures that pie chart is a circle.
plt.show()

# Box plot for Ratings by Branch
plt.figure(figsize=(10, 6))
sns.boxplot(x='Branch', y='Rating', data=df)
plt.title('Rating Distribution by Branch')
plt.xlabel('Branch')
plt.ylabel('Rating')
plt.show()

from IPython.display import display, HTML

# Define your style
style = """
<style>
  body {
    font-family: Arial, sans-serif;
    background-color: #f5f5f5;
    color: #333;
  }
  h1 {
    color: #0066cc;
  }
  /* Add more styling as needed */
</style>
"""
```

```
# Display the style
display(HTML(style))

def login_screen():
    # Your login screen code here
    pass



import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
from datetime import datetime
from google.colab import files
import ipywidgets as widgets
from IPython.display import display, HTML, clear_output
import sys

# Custom styling
"""style =
<style>
.custom-header {
    background-color: #2b5797;
    color: white;
    padding: 20px;
    border-radius: 10px;
    margin: 10px 0;
    text-align: center;
    font-family: 'Arial', sans-serif;
}
.welcome-message {
    background-color: #4CAF50;
    color: white;
```

```
    padding: 15px;
    border-radius: 8px;
    margin: 10px 0;
    text-align: center;
}
.result-container {
    background-color: #f8f9fa;
    padding: 15px;
    border-radius:  8px;
    border: 1px solid #dee2e6;
    margin: 10px 0;
}
.book-card {
    background-color: white;
    padding: 10px;
    margin: 5px 0;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}
</style>"""

display(HTML(style))

def login_screen():
    clear_output(wait=True)
    display(HTML("<div class='custom-header'><h1>       Engineering Book Recommendation System</h1></div>"))


    username_input = widgets.Text(
        value='',
        placeholder='Enter your name',
        description='Username:',
        style={'description_width': 'initial'},
```

```
        layout=widgets.Layout(width='50%')

    )

    login_button = widgets.Button(
        description='Login',
        button_style='success',
        layout=widgets.Layout(width='20%')
    )

    def on_login_click(b):
        if username_input.value.strip():
            initialize_system(username_input.value)

    login_button.on_click(on_login_click)
    display(username_input, login_button)

def initialize_system(username):
    clear_output(wait=True)

    # Display welcome header
    welcome_html = f"""
        <div class='custom-header'>
            <h1>      Smart Book Recommendation System</h1>
        </div>

        <div class='welcome-message'>
            <h2>Welcome, {username}! 📚</h2>
```

```
        <p>What would you like to explore today?</p>
    </div>
"""
display(HTML(welcome_html))


# Load the dataset
df = pd.read_csv("/content/cleaned_combined_books_dataset.csv")

# Preprocess the data
df['text'] = df['Book Title'] + ' ' + df['Course'] + ' ' + df['Branch'] + ' ' + df['Description']
df['text'] = df['text'].fillna('')
df['Published Date'] = pd.to_datetime(df['Published Date'])

def get_recommendations(input_text, input_type, df, top_n=5):
    tfidf = TfidfVectorizer(stop_words='english')
    tfidf_matrix = tfidf.fit_transform(df['text'])

    input_tfidf = tfidf.transform([input_text])
    cosine_sim = linear_kernel(input_tfidf, tfidf_matrix)

    sim_scores = list(enumerate(cosine_sim[0]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    top_indices = [i[0] for i in sim_scores[:top_n]]
    return df.iloc[top_indices]

def display_results(title, results):
    result_html = f"<div class='result-container'><h3>{title}</h3>"

    for _, row in results.iterrows():
        result_html += f"""
            <div class='book-card'>
```

```
<h4>{row['Book Title']}</h4>
<p><strong>Course:</strong> {row['Course']}</p>
<p><strong>Branch:</strong> {row['Branch']}</p>
<p><strong>Rating:</strong> {"    " * int(row['Rating'])}</p>
<p><strong>Published:</strong> {str(row['Published Date']).split()[0]}</p>
```

```
            </div>
        """

    result_html += "</div>"
    display(HTML(result_html))

def recommend_by_branch(branch):
    clear_output(wait=True)
    display_header()
    recommendations = get_recommendations(branch, 'Branch', df)
    display_results(f"📚 Recommended Books for {branch}:", recommendations)
    display_navigation()


def recommend_by_course(course):
    clear_output(wait=True)
    display_header()
    recommendations = get_recommendations(course, 'Course', df)
    display_results(f"📚 Recommended Books for {course}:", recommendations)
    display_navigation()


def search_by_title(title):
    clear_output(wait=True)
    display_header()
    recommendations = get_recommendations(title, 'Book Title', df)
    if recommendations.empty:
        recommendations = df.sort_values('Rating', ascending=False).head(5)
    display_results(f"🔍 Search Results for '{title}':", recommendations)
    display_navigation()

def get_recent_books():
    clear_output(wait=True)
    display_header()
```

```
        current_year = datetime.now().year
        description='Branch:',
        recent_books = df[df['Published Date'].dt.year >= (current_year - 2)].sort_values('Published Date',
ascending=False).head(10)
        display_results("📚 Recent Books:", recent_books)
        display_navigation()


    def exit_system(b):
        clear_output(wait=True)
        display(HTML("""
            <div class='custom-header'>
                <h2>📚 Thank you for using the Book Recommendation System!</h2>
                <p>Have a great day!</p>
            </div>
        """))

    def display_header():
        header_html = f"""
            <div class='custom-header'>
                <h1>      Engineering Book Recommendation System</h1>
                <p>Welcome, {username}!</p>
            </div>
        """
        display(HTML(header_html))

    def display_navigation():
        display(widgets.HBox([branch_dropdown, branch_button]))
        display(widgets.HBox([course_input, course_button]))
        display(widgets.HBox([title_input, title_button]))
        display(widgets.HBox([recent_button, exit_button]))

    # Create styled widgets
    branch_dropdown = widgets.Dropdown(
        options=df['Branch'].unique(),
```

```
    style={'description_width': 'initial'},
    layout=widgets.Layout(width='50%')
)

course_input = widgets.Text(
    value='',
    placeholder='Enter course name',
    description='Course:',
    style={'description_width': 'initial'},
    layout=widgets.Layout(width='50%')
)

title_input = widgets.Text(
    value='',
    placeholder='Enter book title',
    description='Title:',
    style={'description_width': 'initial'},
    layout=widgets.Layout(width='50%')
)

# Create styled buttons
branch_button = widgets.Button(
    description='🔍 Search by Branch',
    button_style='info',
    layout=widgets.Layout(width='20%')
)

course_button = widgets.Button(
    description='🔍 Search by Course',
    button_style='info',
    layout=widgets.Layout(width='20%')
)
```

```
    title_button = widgets.Button(
        description=' Search by Title',
        button_style='info',
        layout=widgets.Layout(width='20%')
    )

    recent_button = widgets.Button(
        description='      Recent Books',
        button_style='warning',
        layout=widgets.Layout(width='20%')
    )

    exit_button = widgets.Button(
        description='      Exit',
        button_style='danger',
        layout=widgets.Layout(width='20%')
    )

    # Set up button click events
    branch_button.on_click(lambda _: recommend_by_branch(branch_dropdown.value))
    course_button.on_click(lambda _: recommend_by_course(course_input.value))
    title_button.on_click(lambda _: search_by_title(title_input.value))
    recent_button.on_click(lambda _: get_recent_books())
    exit_button.on_click(exit_system)

    # Display initial layout
    display_navigation()

# Start the system
login_screen()
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
from sklearn.metrics.pairwise import linear_kernel
from datetime import datetime
from IPython.display import display, HTML, clear_output
import ipywidgets as widgets

# Load dataset
df = pd.read_csv("/content/cleaned_combined_books_dataset.csv")

# Preprocess the data
df['text'] = df['Book Title'] + ' ' + df['Course'] + ' ' + df['Branch'] + ' ' + df['Description']
df['text'] = df['text'].fillna('')
df['Published Date'] = pd.to_datetime(df['Published Date'])

def get_recommendations(input_text, df, top_n=5):
    tfidf = TfidfVectorizer(stop_words='english')
    tfidf_matrix = tfidf.fit_transform(df['text'])
    input_tfidf = tfidf.transform([input_text])
    cosine_sim = linear_kernel(input_tfidf, tfidf_matrix)
    sim_scores = list(enumerate(cosine_sim[0]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    top_indices = [i[0] for i in sim_scores[:top_n]]
    return df.iloc[top_indices]

def precision_at_k(recommended_df, k=5, rating_threshold=4.0):
    relevant_items = recommended_df[recommended_df['Rating'] >= rating_threshold]
    precision = len(relevant_items) / min(k, len(recommended_df))
    return precision

def evaluate_recommendation_system(df, input_texts, k=5):
    precisions = []
    for text in input_texts:
        recommended_df = get_recommendations(text, df, top_n=k)
        precision = precision_at_k(recommended_df, k)
```

```
        precisions.append(precision)
    avg_precision = np.mean(precisions)
    print(f"Average Precision@{k}: {avg_precision:.2f}")

# Example: Evaluating the recommendation system with sample input texts
input_texts = ['Math', 'Computer Science', 'Data Science', 'Mechanical Engineering']
evaluate_recommendation_system(df, input_texts)

import matplotlib.pyplot as plt

# Function to plot Precision@K for each input text and the average precision
def plot_precision(input_texts, precisions, k=5):
    plt.figure(figsize=(10, 6))
    plt.bar(input_texts, precisions, color='skyblue')
    plt.axhline(y=np.mean(precisions), color='r', linestyle='--', label=f'Average Precision@{k}')
    plt.xlabel('Input Text')
    plt.ylabel(f'Precision@{k}')
    plt.title(f'Precision@{k} for each input text')
    plt.legend()
    plt.show()

# Modified evaluation function to include plotting
def evaluate_and_plot(df, input_texts, k=5):
    precisions = []
    for text in input_texts:
        recommended_df = get_recommendations(text, df, top_n=k)
        precision = precision_at_k(recommended_df, k)
        precisions.append(precision)
    avg_precision = np.mean(precisions)

    # Displaying Precision@K values
    for idx, text in enumerate(input_texts):
        print(f"Precision@{k} for '{text}': {precisions[idx]:.2f}")
```

```
    print(f"Average Precision@{k}: {avg_precision:.2f}")

    # Plotting Precision@K for each input text
    plot_precision(input_texts, precisions, k)

# Example: Evaluating and plotting
evaluate_and_plot(df, input_texts)
from sklearn.metrics import precision_score, recall_score, f1_score

# Calculate Precision, Recall, and F1-score at K
def precision_at_k(recommended_df, k=5, rating_threshold=4.0):
    relevant_items = recommended_df[recommended_df['Rating'] >= rating_threshold]
    precision = len(relevant_items) / min(k, len(recommended_df))
    return precision

def recall_at_k(recommended_df, k=5, rating_threshold=4.0):
    relevant_items = recommended_df[recommended_df['Rating'] >= rating_threshold]
    recall = len(relevant_items) / sum(recommended_df['Rating'] >= rating_threshold)
    return recall

def f1_at_k(precision, recall):
    if precision + recall == 0:
        return 0
    return 2 * (precision * recall) / (precision + recall)

# Function to evaluate and plot metrics (Precision, Recall, F1-score)
def evaluate_and_plot(df, input_texts, k=5):
    precisions = []
    recalls = []
    f1_scores = []

    for text in input_texts:
        recommended_df = get_recommendations(text, df, top_n=k)
```

```
    precision = precision_at_k(recommended_df, k)
    recall = recall_at_k(recommended_df, k)
    f1 = f1_at_k(precision, recall)

    precisions.append(precision)
    recalls.append(recall)
    f1_scores.append(f1)

avg_precision = np.mean(precisions)
avg_recall =  np.mean(recalls)
avg_f1 = np.mean(f1_scores)

# Displaying individual metrics for each input text
for idx, text in enumerate(input_texts):
    print(f"Metrics for '{text}':")
    print(f" Precision@{k}: {precisions[idx]:.2f}")
    print(f"  Recall@{k}: {recalls[idx]:.2f}")
    print(f" F1-Score@{k}: {f1_scores[idx]:.2f}")
    print()

# Display average metrics
print(f"Average Precision@{k}: {avg_precision:.2f}")
print(f"Average Recall@{k}: {avg_recall:.2f}")
print(f"Average F1-Score@{k}: {avg_f1:.2f}")

# Plotting the metrics
metrics_df = pd.DataFrame({
    'Input Text': input_texts,
    'Precision': precisions,
    'Recall': recalls,
    'F1-Score': f1_scores
})
metrics_df.set_index('Input Text', inplace=True)
```

```
    # Plotting the metrics for each input text
    metrics_df.plot(kind='bar', figsize=(10, 6))
    plt.title(f'Metrics (Precision, Recall, F1-Score) for each input text')
    plt.ylabel('Metric Value')
    plt.xlabel('Input  Text')
    plt.xticks(rotation=45)
    plt.show()

# Example: Evaluating and plotting the metrics
evaluate_and_plot(df, input_texts)
```

4.2 TEST RESULTS

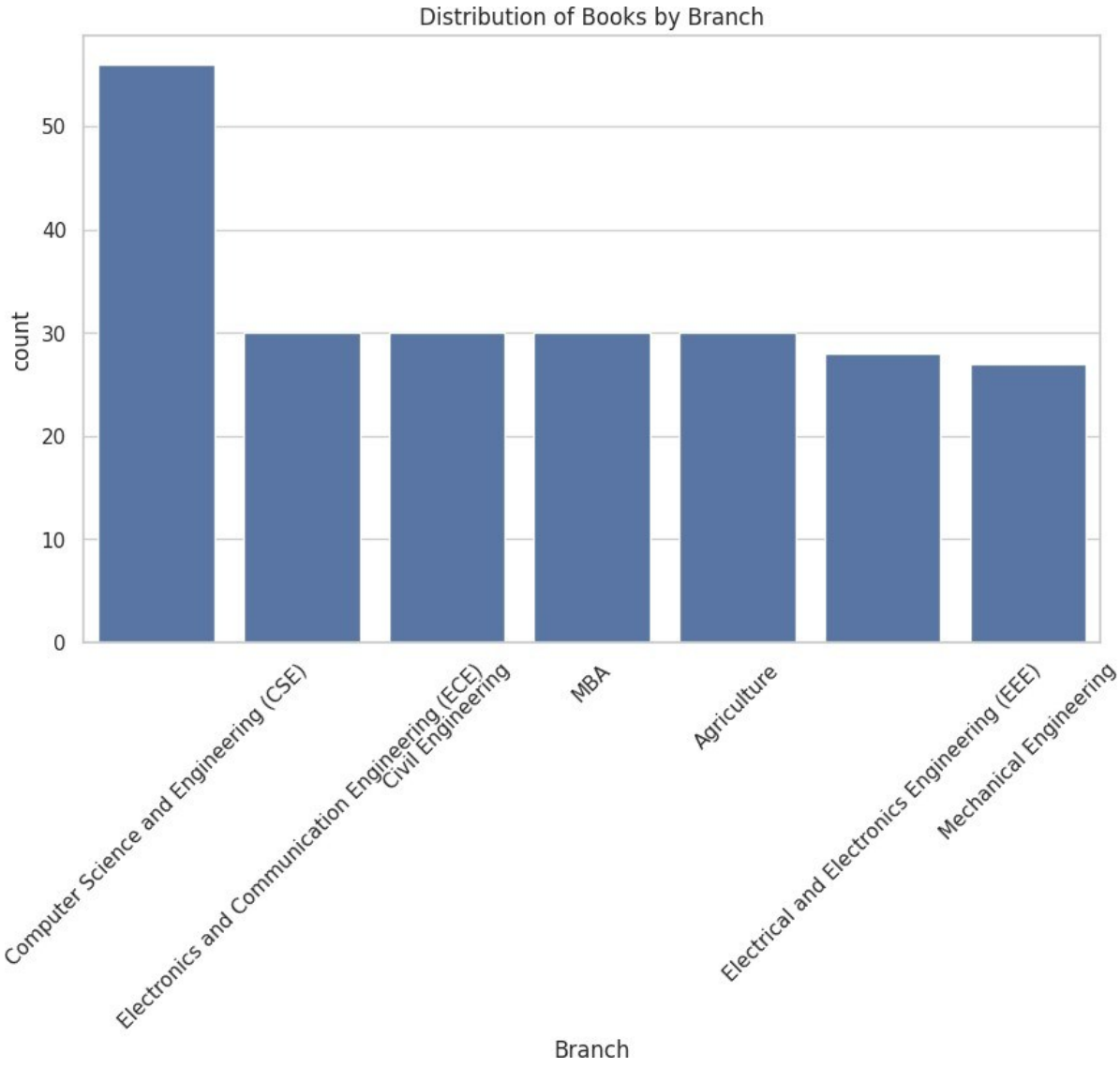

Fig 5.2.1 DISTRIBUTION OF BOOKS BY BRANCH

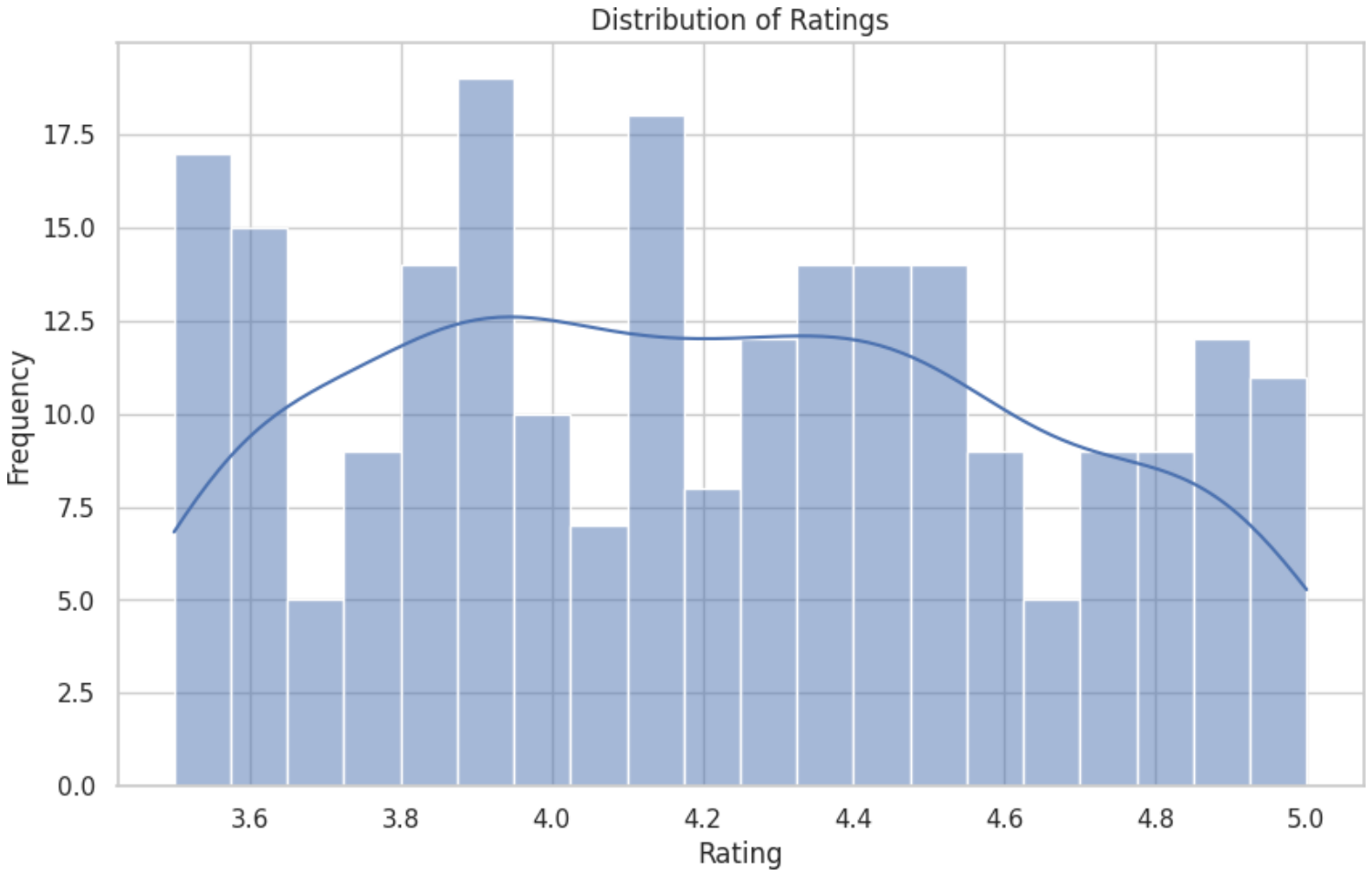

FIG 5.2.2 DISTRIBUTION OF RATINGS

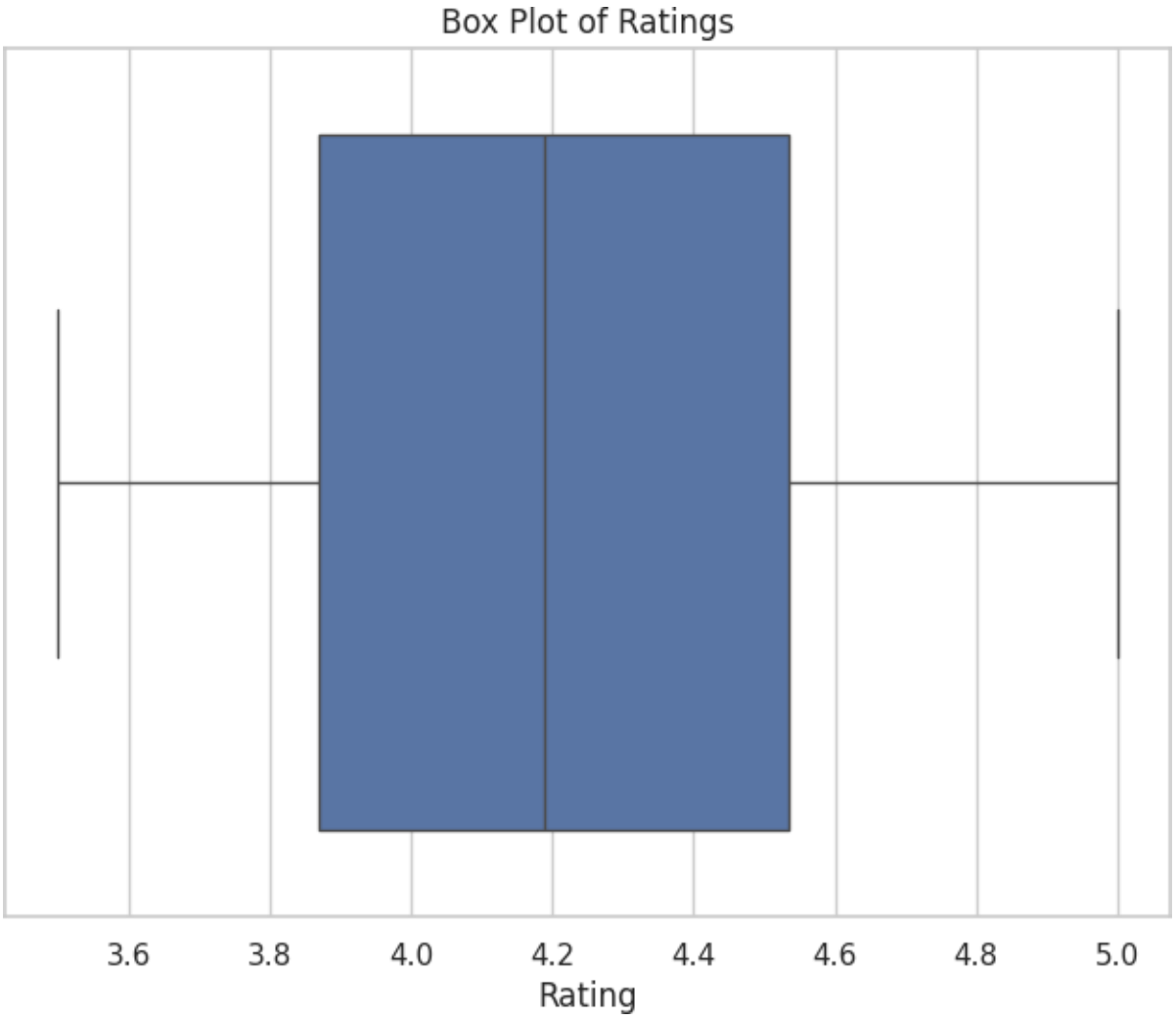

FIG 5.2.3 BOXPLOT OF RATINGS

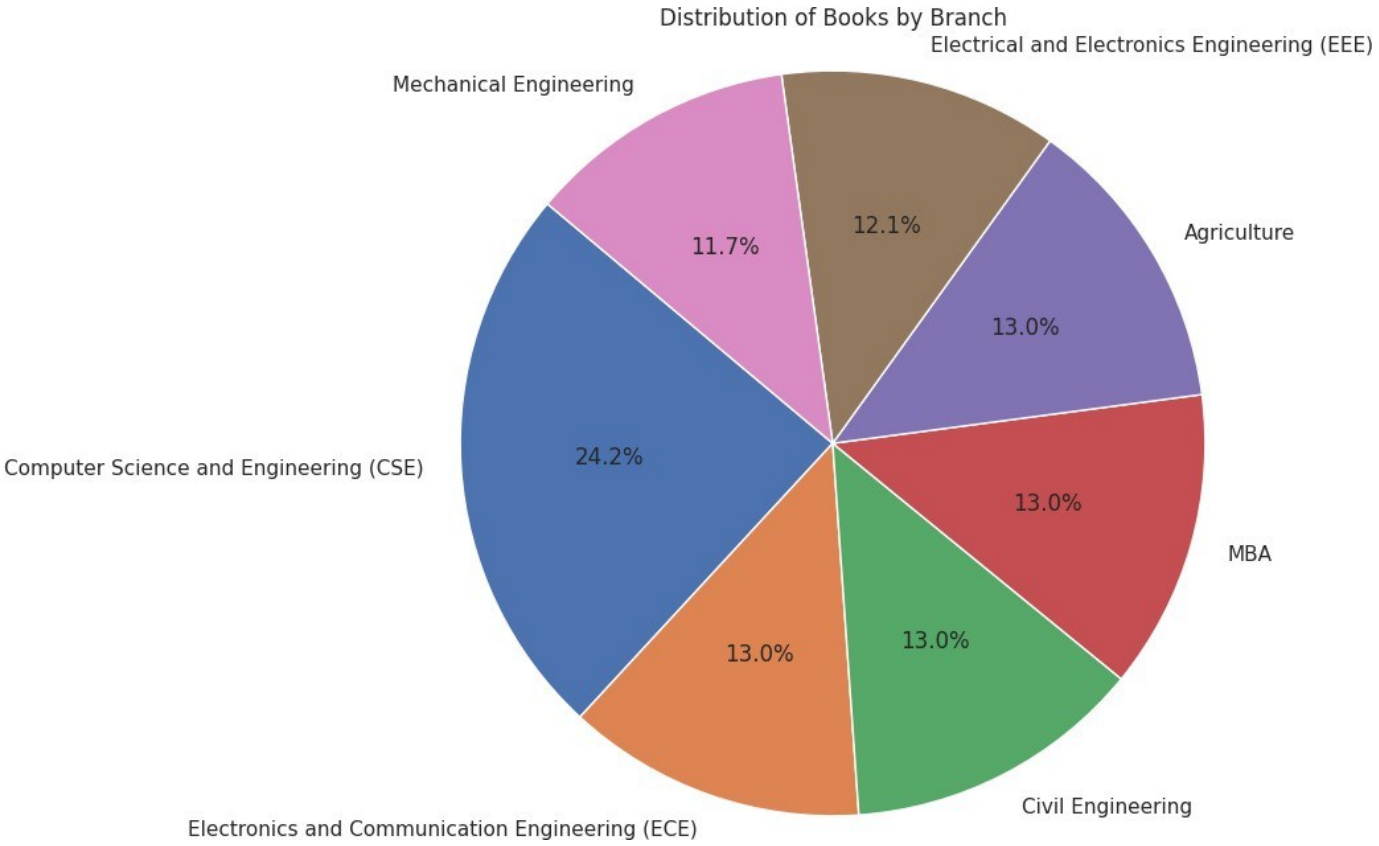

FIG 5.2.4 DISTRIBUTION OF BOOKS BY BRANCH

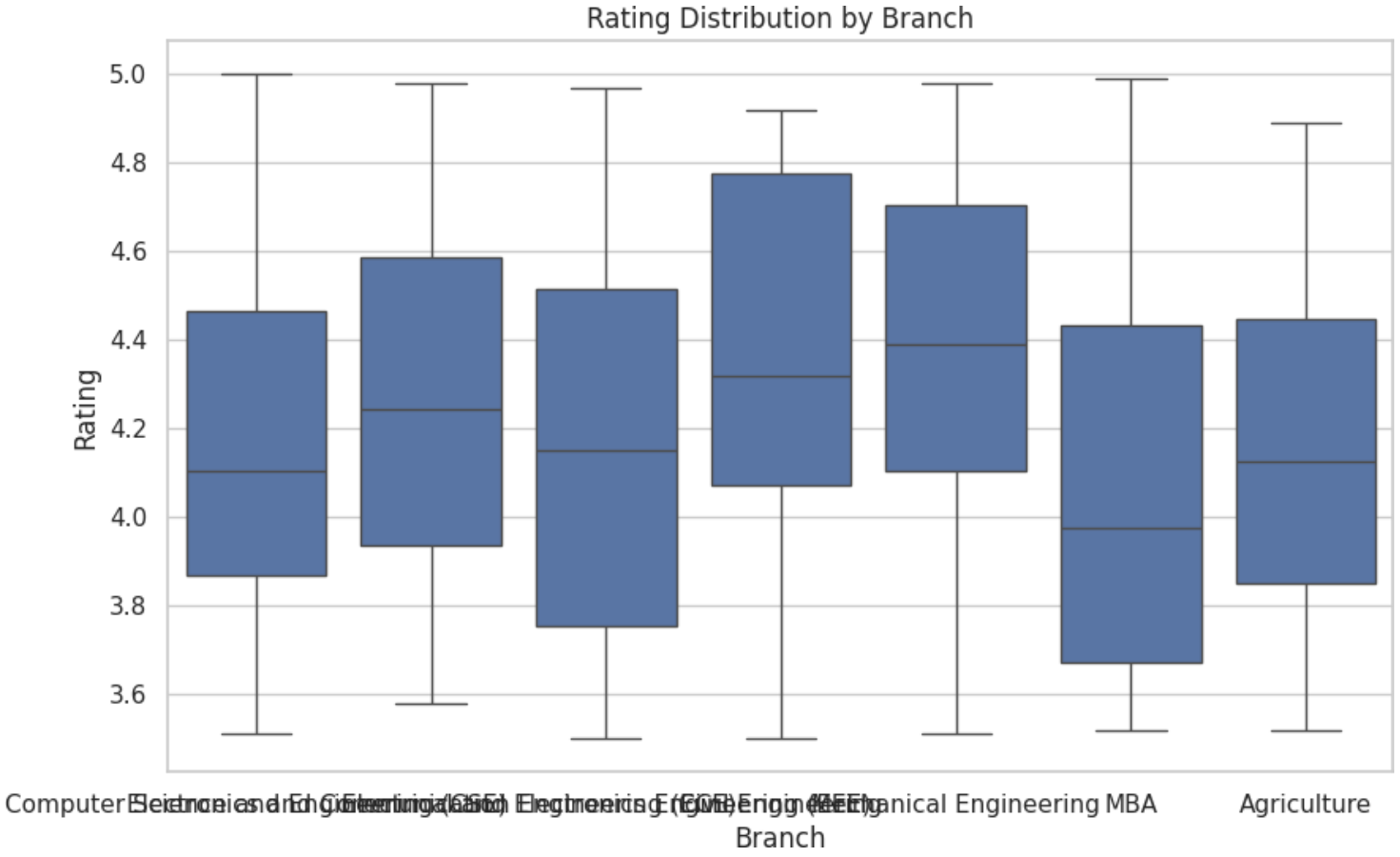

FIG 5.2.5 RATING DISTRIBUTION BY BRANCH

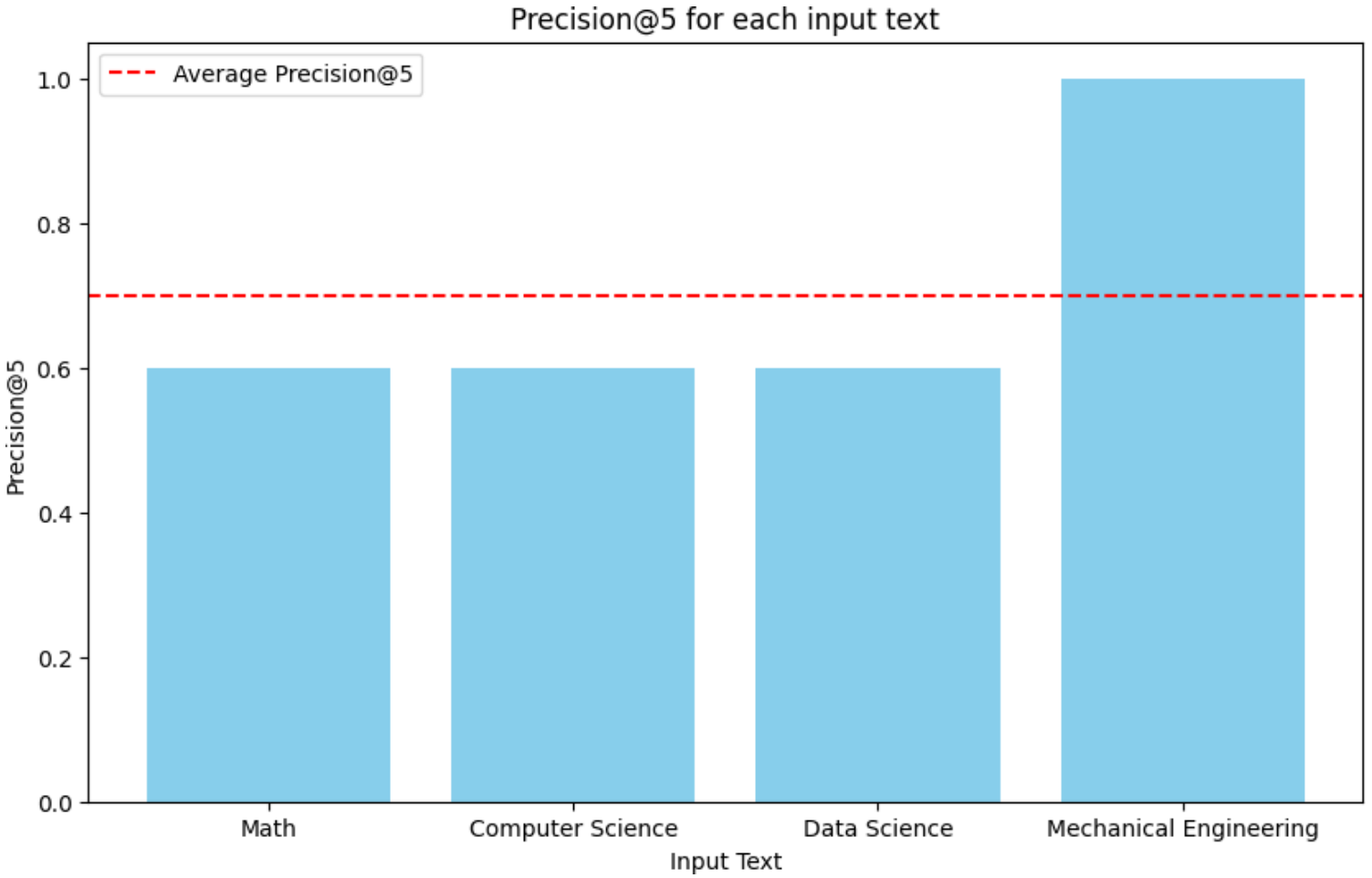

FIG 5.2.6 PRECISION FOR EACH INPUT TEXT

RESULTS

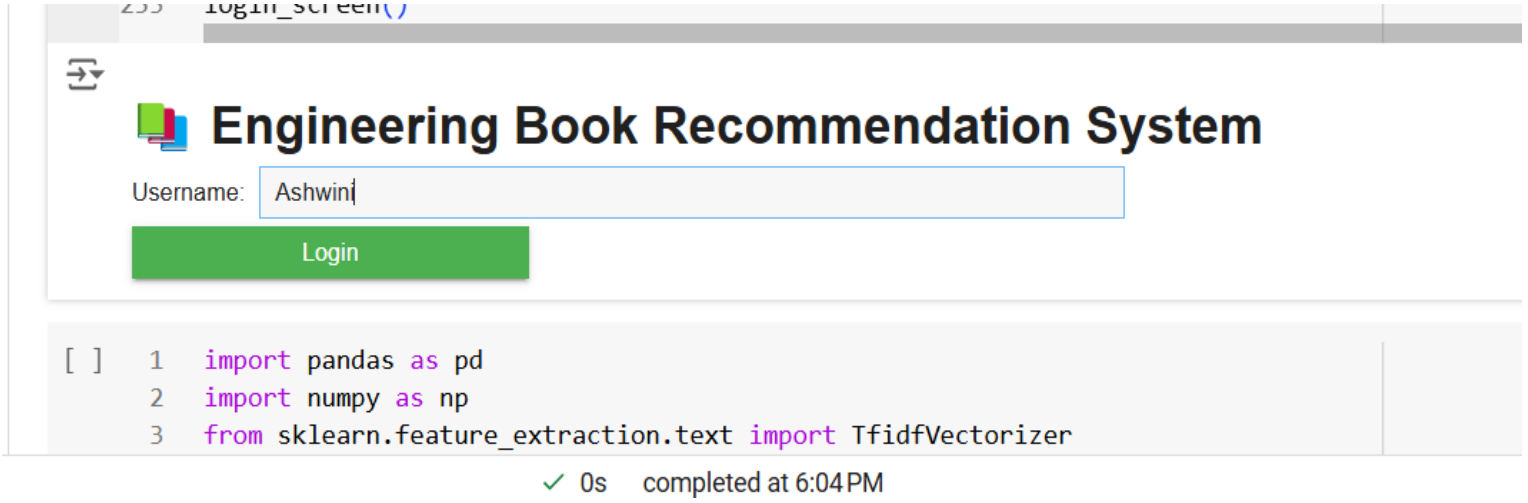

FIG 6.1

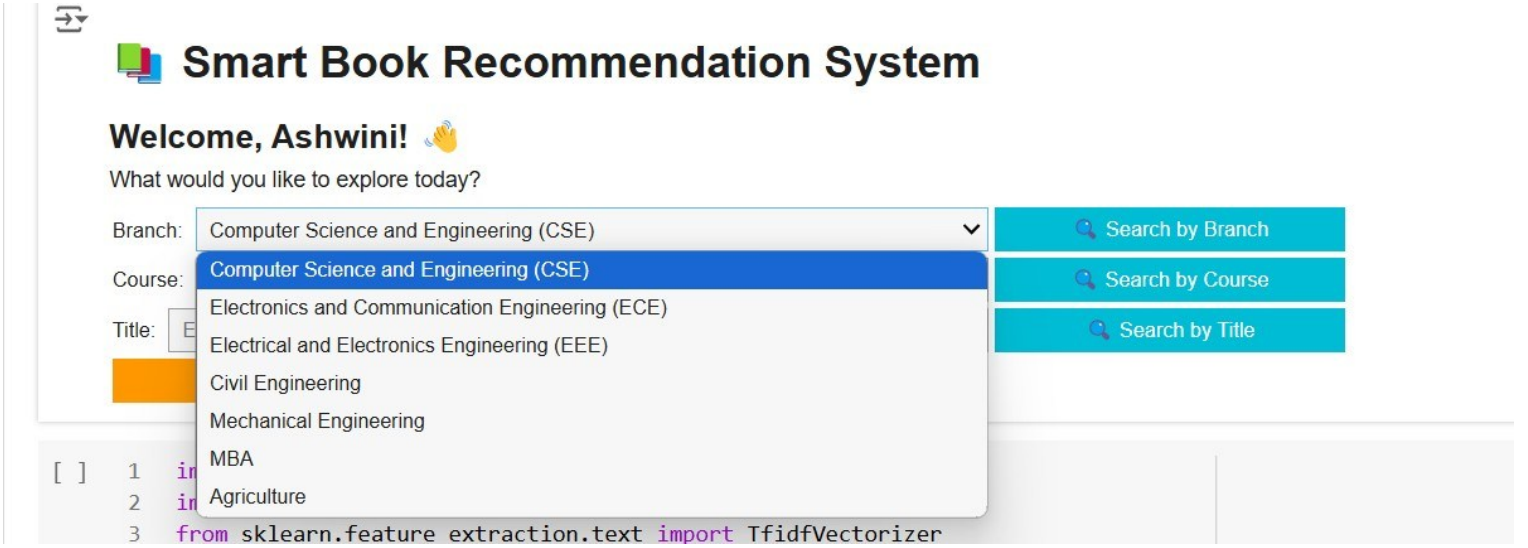

FIG 6.2

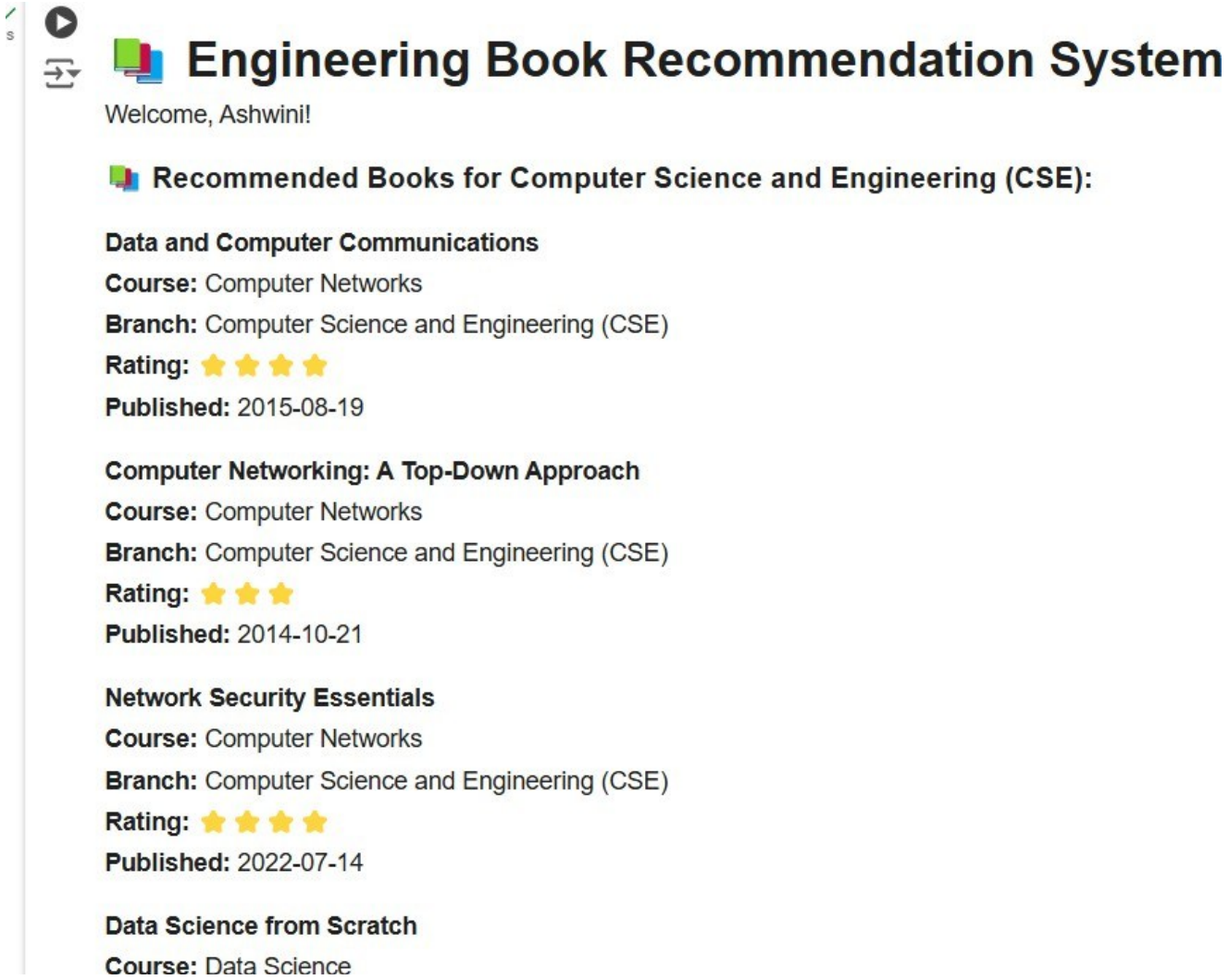
Engineering Book Recommendation System
Welcome, Ashwini!
Recommended Books for Computer Science and Engineering (CSE):
Data and Computer Communications
Course: Computer Networks
Branch: Computer Science and Engineering (CSE)
Rating:
Published: 2015-08-19
Computer Networking: A Top-Down Approach
Course: Computer Networks
Branch: Computer Science and Engineering (CSE)
Rating:
Published: 2014-10-21
Network Security Essentials
Course: Computer Networks
Branch: Computer Science and Engineering (CSE)
Rating:
Published: 2022-07-14
Data Science from Scratch
Course: Data Science

FIG 6.3

Published: 2021-11-04

Data Structures and Algorithms in Java
Course: Data Structures
Branch: Computer Science and Engineering (CSE)
Rating: ⭐ ⭐ ⭐ ⭐
Published: 2022-09-18

Data Science from Scratch
Course: Data Science
Branch: Computer Science and Engineering (CSE)
Rating: ⭐ ⭐ ⭐ ⭐
Published: 2021-04-14

Practical Statistics for Data Scientists
Course: Data Science
Branch: Computer Science and Engineering (CSE)
Rating: ⭐ ⭐ ⭐ ⭐
Published: 2021-05-26

FIG 6.4

```
248
249     # Display initial layout
250     display_navigation()
251
252 # Start the system
253 login_screen()
```

👋 Thank you for using the Book Recommendation System!
Have a great day!

FIG 6.5

input	precision	recall	F1-Score
Math	0.60	1.00	0.75
Computer Science	0.60	1.00	0.75
Data Science	0.60	1.00	0.75
Mechanical Engineering	1.00	1.00	1.00

FIG 6.6 Table: comparison of different metrices using sample input

Average Precision: 0.70
Average Recall: 1.00
Average F1-Score: 0.81

6. CONCLUSION

our project engineering students' book recommendation system demonstrates promising results, with an Average Precision of 0.70, Average Recall of 1.00, and an Average F1-Score of 0.81. These metrics indicate that our system effectively retrieves relevant books and performs well in balancing precision and recall This performance suggests that the recommendation system is particularly strong in identifying relevant books for students.

7. FUTURE SCOPE

Future enhancements for this book recommendation system could include integrating additional data sources, such as academic journals, articles, and links to online resources, to provide a broader selection of study materials. Expanding the dataset with more comprehensive book metadata and student feedback can improve personalization and recommendation accuracy. Additionally, incorporating real-time updates to adapt recommendations based on trending topics or new courses could enhance system relevance and usability.

8.BIBLIOGRAPHY

[1] Roy, D., Dutta, M. A systematic review and research perspective on recommender systems. *J Big Data* **9**, 59 (2022). https://doi.org/10.1186/s40537-022-00592-5
[2] Wu, S., Sun, F., Zhang, W., Xie, X., & Cui, B. (2022). Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, *55*(5), 1-37.
[3] Kommineni, M., Alekhya, P., Vyshnavi, T. M., Aparna, V., Swetha, K., & Mounika, V. (2020, January). Machine learning based efficient recommendation system for book selection using user based collaborative filtering algorithm. In *2020 Fourth International Conference on Inventive Systems and Control (ICISC)* (pp. 66-71). IEEE.
[4] Cui, B., & Chen, X. (2009, August). An online book recommendation system based on web service. In *2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery* (Vol. 7, pp. 520-524). IEEE.
[5] Tewari, A. S., Kumar, A., & Barman, A. G. (2014, February). Book recommendation system based on combine features of content based filtering, collaborative filtering and association rule mining. In *2014 IEEE International Advance Computing Conference (IACC)* (pp. 500-503). IEEE.
[6] Sohail, S. S., Siddiqui, J., & Ali, R. (2013, August). Book recommendation system using opinion mining technique. In *2013 international conference on advances in computing, communications and informatics (ICACCI)* (pp. 1609-1614). IEEE.
[7] Jomsri, P. (2014, August). Book recommendation system for digital library based on user profiles by using association rule. In *Fourth edition of the International Conference on the Innovative Computing Technology (INTECH 2014)* (pp. 130-134). IEEE.
[8] Kanetkar, S., Nayak, A., Swamy, S., & Bhatia, G. (2014, August). Web-based personalized hybrid book recommendation system. In *2014 International Conference on Advances in Engineering & Technology Research (ICAETR-2014)* (pp. 1-5). IEEE.
[9] Kurmashov, N., Latuta, K., & Nussipbekov, A. (2015, September). Online book recommendation system. In *2015 Twelve International Conference on Electronics Computer and Computation (ICECCO)* (pp. 1-4). IEEE.
[10] Kusumawardhani, N. K., Nasrun, M., & Setianingsih, C. (2019, December). Web Recommended System Library Book Selection Using Item Based Collaborative Filtering Method. In *2019 IEEE International Conference on Engineering, Technology and Education (TALE)* (pp. 1-8). IEEE.
[11] Liang, Y., & Wan, S. (2018, November). The Design and Implementation of Books Recommendation System. In *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)* (pp. 305-308). IEEE.
[12] Sirikayon, C., Thusaranon, P., & Pongtawevirat, P. (2018, May). A collaborative filtering based library book recommendation system. In *2018 5th International Conference on Business and Industrial Research (ICBIR)* (pp. 106-109). IEEE.
[13] Verma, M., & Patnaik, P. K. (2024). An automatic college library book recommendation system using optimized Hidden Markov based weighted fuzzy ranking model. *Engineering Applications of Artificial Intelligence*, *130*, 107664.
[14] Ifada, N., Syachrudin, I., Sophan, M. K., & Wahyuni, S. (2019). Enhancing the performance of

library book recommendation system by employing the probabilistic-keyword model on a collaborative filtering approach. *Procedia Computer Science*, *157*, 345-352.
[15] Sohail, S. S., Siddiqui, J., & Ali, R. (2015). OWA based book recommendation technique. *Procedia Computer Science*, *62*, 126-133.
[16] Li, C., Ishak, I., Ibrahim, H., Zolkepli, M., Sidi, F., & Li, C. (2023). Deep Learning-Based Recommendation System: Systematic Review and Classification. *IEEE Access*.
[17] Nitin Mishra, Neetish Kumar Chandrakar, Saumya Chaturvedi, Kare Prashanthi, 2015, Review of Various Recommendation Techniques for Web Applications, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) ISNCESR – 2015 (Volume 3 – Issue 20),