

**Tribhuvan University**

**Faculty of Humanities and Social Sciences**

A PROJECT REPORT ON

**Course Recommender System**

**Submitted to**

**Department of Computer Application**

**Patan Multiple Campus**

**Patan Dhoka, Lalitpur**

***In partial fulfillment of the requirements for the Bachelors in Computer Application***

**Submitted by**

John Doe

BCA 6th Semester

Registration No.: 6-2-22-887-2019

Under the Supervision of

**John Doe**



**Tribhuvan University**

**Faculty of Humanities and Social Sciences**

**Patan Multiple Campus**

Patan Dhoka, Lalitpur

Bachelor in Computer Applications (BCA)

# SUPERVISOR’S RECOMMENDATION

I hereby recommend that this project prepared under my supervision by **John Doe** entitled “**Course Recommender System”** in the Partial Fulfillment of requirement for the degree of Bachelor in Computer Application is recommended for that final evaluation.

John Doe

Project Supervisor

BCA Department

Patan Multiple Campus

# ABSTRACT

The Course Recommendation System is a web-based platform designed to help users discover relevant courses based on their search behavior, viewed courses, and trending topics. It utilizes content-based filtering by analyzing course titles and descriptions and collaborative filtering by tracking user interactions to recommend courses frequently viewed together. Logged-in users receive personalized recommendations based on their most searched keywords and frequently viewed courses, while guest users can explore featured and trending courses. The system dynamically updates trending searches and enhances user experience by reducing search time and improving content discovery through an intelligent recommendation engine.

**Keywords**: Content Based Algorithm, Trending Searches, Personalized Learning



**Tribhuvan University**

**Faculty of Humanities and Social Sciences**

**Patan Multiple Campus**

Patan Dhoka, Lalitpur

Bachelor in Computer Applications (BCA)

# LETTER OF APPROVAL

This is certify that this project prepared by **John doe** entitled “**Course Recommender System”** in the Partial Fulfillment of requirement for the degree of Bachelor in Computer Application has been evaluated. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

John doe

Supervisor

BCA Department

Patan Multiple Campus

John doe

Program Coordinator

Patan Multiple Campus

Patan Dhoka, Lalitpur

# 

# ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to our supervisor **Mr. John Doe** who gave me the golden opportunity to do this wonderful project on the topic of **Course Recommender System**, which also helped us in doing a lot of research.

I would like to express my special thanks of gratitude to our Campus Chief Dr**. john doe** who gave me permission for doing this Project.

I would like to express my special gratitude and thanks to our BCA Program Coordinator **Mr. John Doe** for his support and help for our personnel development and mainly for the completion of this Project.

I thank Patan Multiple Campus a lot for their direction and continuous supervision, as well as for providing the knowledge that I required to understand the project and for assisting with seeing it through to its completion..

I would also like to thank my friends who helped me a lot in finalizing this project within the limited time frame.

Yours sincerely,

John doe

**TABLE OF CONTENTS**

[SUPERVISOR’S RECOMMENDATION ii](#_Toc175251926)

[ABSTRACT iii](#_Toc175251927)

[LETTER OF APPROVAL iv](#_Toc175251928)

[ACKNOWLEDGEMENT v](#_Toc175251929)

[CHAPTER 1: INTRODUCTION 1](#_Toc175251930)

[1.1 Introduction 1](#_Toc175251931)

[1.2.Problem Statement 1](#_Toc175251932)

[1.3.Objectives 2](#_Toc175251933)

[1.4 Scope and Limitation 2](#_Toc175251934)

[1.5 Development Methodology 2](#_Toc175251935)

[1.6 Report Organization 3](#_Toc175251936)

[CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW 4](#_Toc175251937)

[2.1 Background Study 4](#_Toc175251938)

[2.1.1 Fundamental Theories and General Concepts 4](#_Toc175251939)

[2.2 Literature Review 5](#_Toc175251940)

[CHAPTER 3: SYSTEM ANALYSIS AND DESIGN 6](#_Toc175251941)

[3.1 System Analysis 6](#_Toc175251942)

[3.1.1 Requirement Analysis 6](#_Toc175251943)

[3.1.2 Feasibility Analysis 8](#_Toc175251944)

[3.1.3 Data modelling: ER Diagram 9](#_Toc175251945)

[3.1.4 Process Modelling: DFD 9](#_Toc175251946)

[3.2 System Design 12](#_Toc175251947)

[3.2.1 Architectural design 12](#_Toc175251948)

[3.2.2 Database Schema design 13](#_Toc175251949)

[3.2.3 Interface design (UI/UX) 13](#_Toc175251950)

[3.2.4 Physical DFD 16](#_Toc175251951)

[3.3 Algorithm details 16](#_Toc175251952)

[CHAPTER 4: IMPLEMENTATION AND TESTING 19](#_Toc175251953)

[4.1 Implementation 19](#_Toc175251954)

[4.1.1Tools Used: 19](#_Toc175251955)

[4.1.2 Implementation details of modules 19](#_Toc175251956)

[5. APPENDICES 20](#_Toc175251957)

[6. REFRENCES 20](#_Toc175251958)

**LIST OF ABBREVIATIONS**

CSS Cascading Style sheets

DFD Data Flow Diagram

ERD Entity Relationship diagram

HTML Hypertext Markup Language

JS JavaScript

UI User Interface

UX User Experience

**LIST OF FIGURES**

Figure 1.1 Waterfall Model 3

Figure 3.1 Use case diagram 7

Figure 3.2 ER-Diagram 9

Figure 3.3 Level 0 DFD 10

Figure 3.4. Level 1 DFD 11

Figure 3.5 Architectural design 12

Figure 3.6 Database Schema design 13

Figure 3.8 Interface design (UI/UX) 13

Figure 3.9 Physical DFD 16

Figure 5.1 Appendices 20

# CHAPTER 1: INTRODUCTION

## Introduction

**Course Recommendation System** is designed to make finding the right courses easier for users. When someone selects a course to view its details, the system automatically suggests **similar courses**, helping them explore more options without extra effort. This way, users can quickly discover courses that align with their interests without having to browse endlessly.

The system also keeps track of **what users search for and which courses they view**, storing this information in a database. This data is then used to create a **Trending Searches** section, showing popular search topics among users. By doing this, the system not only recommends similar courses but also helps users stay updated with what’s trending, making the experience more engaging and relevant.

Another key feature is **recommendations based on search keywords**. When a user searches for a course, the system looks at the words they used and suggests other related courses. This makes it easier for users to find exactly what they’re looking for, without having to refine their searches multiple times. With these smart recommendations, the **Course Recommendation System** simplifies the entire process, making course selection smoother and more personalized.

## 1.2.Problem Statement

In today's digital learning environment, students and professionals often struggle to find the right courses that match their interests and learning goals. With thousands of courses available, manually searching and selecting the most relevant ones can be time-consuming and overwhelming. Traditional search methods rely heavily on user input, but they lack personalized recommendations based on user behavior and preferences.

Course Recommendation System addresses this challenge by automating course discovery through a smart recommendation engine. It analyzes user interactions, search keywords, and course views to suggest similar courses tailored to individual needs. Additionally, a Trending Searches section helps users stay informed about popular topics, improving engagement and decision-making. This system enhances the learning experience by reducing search effort, providing personalized recommendations, and improving course discoverability.

## 1.3.Objectives

The main objectives of Used Car Price Prediction System are as below:

* To recommend similar courses using Content-Based Filtering
* To provide users with relevant course recommendations based on their selected courses and search history
* To minimize the time users spend searching the courses by automating recommendations and making course exploration more efficient.

## 1.4 Scope and Limitation

**i. Scope:**

The Course Recommendation System is designed to enhance the learning experience by providing personalized course suggestions based on user interactions. The system caters to both authenticated and non-authenticated users, allowing anyone to search for courses while storing search keywords and user views only for logged-in users.

The recommendation engine works in two keyways: when a user views a course, similar courses are suggested, and when a user searches for a course, recommendations are generated based on their search keywords. Additionally, the system features a Trending Searches section that highlights popular search terms, helping users discover trending courses.

The system is scalable and can be integrated with various e-learning platforms, educational institutions, or online course marketplaces. Future improvements may include advanced filtering options, AI-driven recommendation enhancements, and integration with user profiles for more personalized suggestions. Overall, the system aims to make course selection easier, more intuitive, and tailored to user preferences.

**ii. Limitations:**

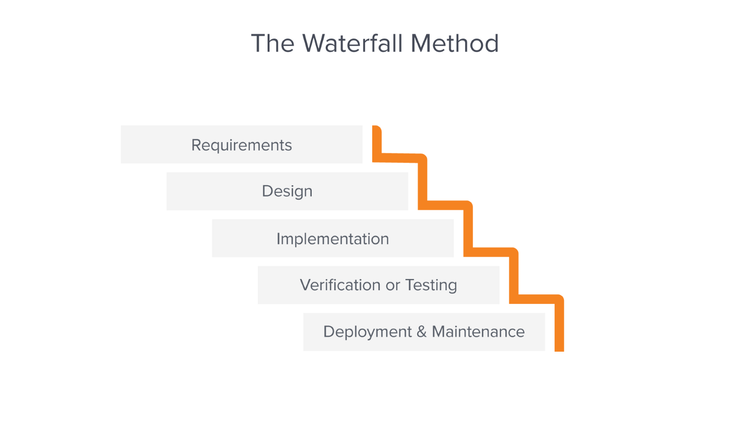
One limitation of the Course Recommendation System is that its accuracy depends on user interactions. New users, who haven’t viewed any courses or searched for specific topics, may receive generic recommendations instead of highly personalized ones. Additionally, since the system relies on stored search keywords and course views, the quality of recommendations is tied to the availability of user data. If there isn't enough data, the suggestions may not be as relevant. The Trending Searches feature also has its limitations, as it highlights popular topics based on search frequency, which could create a bias toward widely searched courses while overlooking niche or specialized subjects.

Another challenge is that the system does not analyze course content directly, meaning recommendations are based only on search behavior and course selection history rather than the actual topics covered in a course. As the number of users and courses grows, scalability may become an issue, requiring efficient filtering and optimization to maintain system performance. Additionally, the system lacks a direct feedback mechanism, such as allowing users to like, dislike, or rate recommended courses, which could improve future suggestions. To ensure that recommendations remain useful and relevant, the system also requires continuous data updates to reflect evolving user preferences and trends.

.

## 1.5 Development Methodology

The process flow for Course Recommendation System includes analysis of the requirements, design, implementation, testing, and maintenance. During the requirement analysis process, every functional and non-functional requirement is examined and the system is then developed to meet the requirements. The system is integrated and tested after the design phase is followed by the coding and development phase. The system is installed if the testing is successful; if not, some maintenance is carried out prior to the system being used.



**Figure 1.1 Waterfall Model**

## 1.6 Report Organization

The purpose of this report is to provide a comprehensive overview of the Course Recommendation System project. It begins with **Chapter 1:** Introduction, which presents the background of the project, the problems it aims to solve, its objectives, scope, and limitations. This chapter also outlines the development methodology used and explains the structure of the report.

**Chapter 2:** Background Study and Literature Review explores key theories, concepts, and terminology related to the project. It also examines similar systems and reviews previous research in the field, providing a foundation for understanding the project’s significance.

**Chapter 3:** System Analysis and Design focuses on the analysis and design of the system. It details the requirements, data structure, process flow, and overall system architecture, including database design and user interface development.

Finally, **Chapter 4:** Implementation and Testing describes how the system was developed and tested. It covers the tools, platforms, and technologies used, along with the deployment setup and testing methodologies to ensure system reliability and efficiency.

# CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW

## 2.1 Background Study

With the rapid growth of online education and e-learning platforms, learners often struggle to find the most relevant courses tailored to their needs. Traditional search methods rely heavily on user input, requiring individuals to manually browse through large course catalogs, which can be time-consuming and inefficient. Additionally, generic recommendation systems often fail to provide personalized suggestions, leading to frustration and poor user engagement.

To address these challenges, course recommendation systems have become an essential tool in e-learning platforms, leveraging data-driven insights and user behavior tracking to provide better course suggestions. Various recommendation techniques, such as collaborative filtering, content-based filtering, and hybrid models, have been used in different academic and professional learning platforms. However, many of these systems focus primarily on user ratings or historical enrollments rather than real-time search behavior and browsing history.

Our Course Recommendation System is designed to improve this process by analyzing user interactions, including course views and search keywords, to provide more relevant course recommendations. Unlike traditional recommendation models, this system dynamically adapts to search trends and user interests, ensuring that learners discover courses that align with their specific needs. Additionally, the Trending Searches feature enhances user engagement by displaying popular search topics, allowing users to explore trending courses effortlessly. This approach creates a more intuitive, personalized, and efficient course selection experience, making learning more accessible and streamlined.

### 2.1.1 Fundamental Theories and General Concepts

Recommendation Systems: A recommendation system is an intelligent filtering technique used to suggest relevant items to users based on their interests, behaviors, or interactions. It helps improve user experience by reducing search effort and providing personalized suggestions. An instance of this is content-based filtering, where the system matches search keywords and course attributes to recommend courses that align with the user’s interests. This ensures that recommendations are relevant and tailored to individual preferences.

Search Trend Analysis: Search trend analysis involves tracking and analyzing frequently searched keywords to identify popular topics or trends over time. It helps users discover high-demand content based on what others are searching for. An instance of this is keyword frequency tracking, where the system counts and ranks the most commonly searched terms to generate trending course recommendations dynamically.

Database Management: Database management involves storing, organizing, and retrieving structured data efficiently to support system functionality. A well-structured database ensures fast and accurate data retrieval for recommendation processes. An instance of this is SQL-based database querying, where the system fetches relevant courses based on user search terms and historical interactions to generate meaningful recommendations.

## 2.2 Literature Review

The Course Recommendation System is based on various research studies and methodologies in recommendation systems, search trend analysis, and user behavior tracking. Several academic and industry-driven projects have explored different approaches to enhance course recommendations, ranging from collaborative filtering to content-based filtering and hybrid models. This literature review examines existing research and methodologies that have influenced the development of our system.

Recommendation systems play a crucial role in e-learning platforms by assisting users in finding courses that match their interests and learning needs. Studies have shown that traditional search-based discovery methods often overwhelm learners due to the vast number of available courses [1]. Many e-learning platforms, such as Coursera and Udemy, implement machine learning-driven recommendations, but these systems require large datasets and computational resources to function effectively [2]. In contrast, our system provides a simpler approach by recommending courses based on search keywords and viewed courses, ensuring quick and efficient suggestions without complex AI models.

Recommendation models are broadly classified into content-based filtering and collaborative filtering [3]. Content-based filtering suggests items based on the attributes of previously interacted content, while collaborative filtering makes recommendations based on the preferences of similar users. Research highlights that collaborative filtering works well for platforms with extensive user data, but it struggles with cold-start problems, where new users receive poor recommendations due to a lack of prior data. Our Course Recommendation System primarily uses content-based filtering, where recommendations are generated by matching search keywords and course attributes, ensuring that even new users receive relevant suggestions.

Several studies emphasize the importance of user behavior tracking in personalizing recommendations. User interaction data, such as search history and page views, can significantly improve recommendation accuracy [4]. Our system stores user search keywords and course views, allowing it to refine recommendations based on past user behavior. Unlike systems that require explicit user feedback (such as ratings or likes), this approach ensures automatic personalization without requiring additional user input.

Existing research on recommendation systems, search trend analysis, and user behavior tracking has significantly influenced the design of our Course Recommendation System. By adopting a content-based filtering approach, integrating trending search analysis, and leveraging user interaction data, our system ensures efficient, personalized, and relevant course recommendations [5]. Unlike AI-driven models that require large datasets, our system offers a lightweight and effective alternative that improves the course discovery experience for users.

# CHAPTER 3: SYSTEM ANALYSIS AND DESIGN

## 3.1 System Analysis

3.1.1 Requirement Analysis:

The requirements are to be collected prior to beginning projects’ development life cycle. Both functional and non-functional requirements of the system have been researched in order to build and create it.

#### Functional Requirements

**User Management**

* The system allows users to search for courses without authentication
* Logged-in users can create an account, log in, and log out securely.
* The system stores search keywords and course views only for logged-in users.

**Course Search and Viewing**

* Users can search for courses using keywords.
* The system retrieves and displays a list of relevant courses based on the search input.
* Users can click on a course to view detailed information, including the title, description, category, and instructor details.

**Course Recommendation**

* When a user views a course, the system suggests similar courses based on course attributes.
* When a user searches for a course, the system provides recommendations based on the search keywords.
* The system dynamically updates recommendations based on user interactions.

**Use Case Diagram**

The Course Recommendation System is designed to enhance course discovery by providing personalized recommendations based on user interactions. The system allows both authenticated and non-authenticated users to search for courses, with logged-in users benefiting from personalized recommendations based on search history and course views. Additionally, a Trending Searches section highlights frequently searched keywords, helping users explore popular courses.

A diagram of a course

AI-generated content may be incorrect.

**Figure 3. 1 Use case diagram**

#### ii. Non-Functional Requirement

In addition to functional requirements, the Course Recommendation System must meet several non-functional requirements to ensure reliability, performance, security, and usability. These define how the system should operate rather than what it should do.

**Performance Requirements:**

* The system must provide real-time course recommendations with minimal response time.
* Search results and recommendations should load within 2 seconds for an optimal user experience.
* The system should be able to handle multiple concurrent users without significant performance degradation.

**Scalability:**

* The system should be scalable to accommodate an increasing number of courses and users.
* The database should efficiently store and manage large volumes of search history and user interactions.
* The recommendation algorithm should adapt to growing data volumes without a decline in efficiency.

**Security:**

* User authentication and session management should be secured using encryption (for logged-in users).
* Search keywords and course views should be stored securely to maintain data integrity.
* The system must prevent unauthorized access to modify course data or user interactions.

**Usability:**

* The user interface should be simple, intuitive, and easy to navigate.
* The recommendation system should provide clear course suggestions without requiring complex user actions.
* The Trending Searches section should be visible and accessible for easy course discovery.

**Reliability and Availability:**

* The system should be available 24/7, ensuring uninterrupted access to course recommendations.
* In case of failures, the system should recover automatically or notify administrators.

### 3.1.2 Feasibility Analysis

#### i. Technical Feasibility Analysis

The Course Recommendation System is technically feasible as it can be developed using widely available technologies like Python (Flask/Django) or JavaScript (Node.js/React) for backend and frontend development. A relational database such as MySQL or PostgreSQL will store user interactions, course data, and search keywords. The recommendation engine will rely on content-based filtering to match user searches with relevant courses. Cloud-based platforms like AWS, Google Cloud, or Heroku will ensure smooth hosting and scalability, making the system adaptable for growing user traffic.

#### ii. Operational Feasibility Analysis

The system is designed to be user-friendly and intuitive, ensuring that users can search for courses effortlessly and receive personal recommendations. The Trending Searches feature enhances engagement by highlighting popular course topics, helping users explore relevant content. Since the system operates with minimal manual intervention, it requires low maintenance, making it easy to implement in various educational and professional learning environments. Overall, the system aligns well with user needs and expectations.

#### iii) Economic Feasibility Analysis

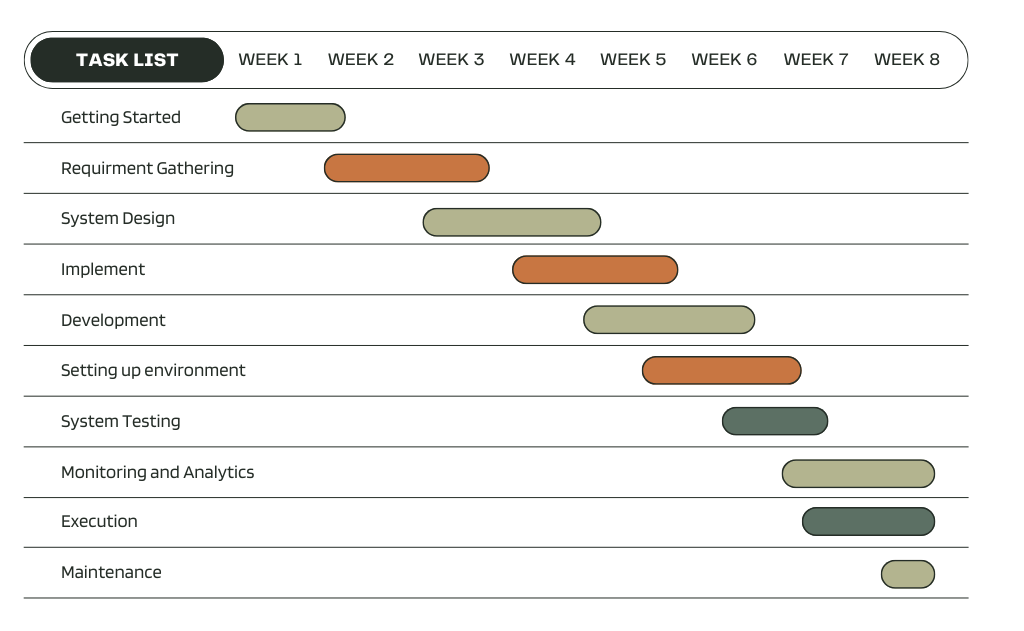
The system is cost-effective, as it utilizes open-source technologies such as Python, MySQL, and JavaScript, reducing software expenses. Cloud-based hosting services will require monthly operational costs, but these can be managed efficiently with scalable solutions. While initial development costs may include expenses for developers, UI/UX designers, and database management, the system has monetization potential through partnerships with e-learning platforms and premium recommendation features. The benefits of automated course discovery and enhanced user engagement outweigh the costs, making the system economically viable.

**iv) Schedule Feasibility**

The system is completed within scheduled time and do not exceed the scheduled time.

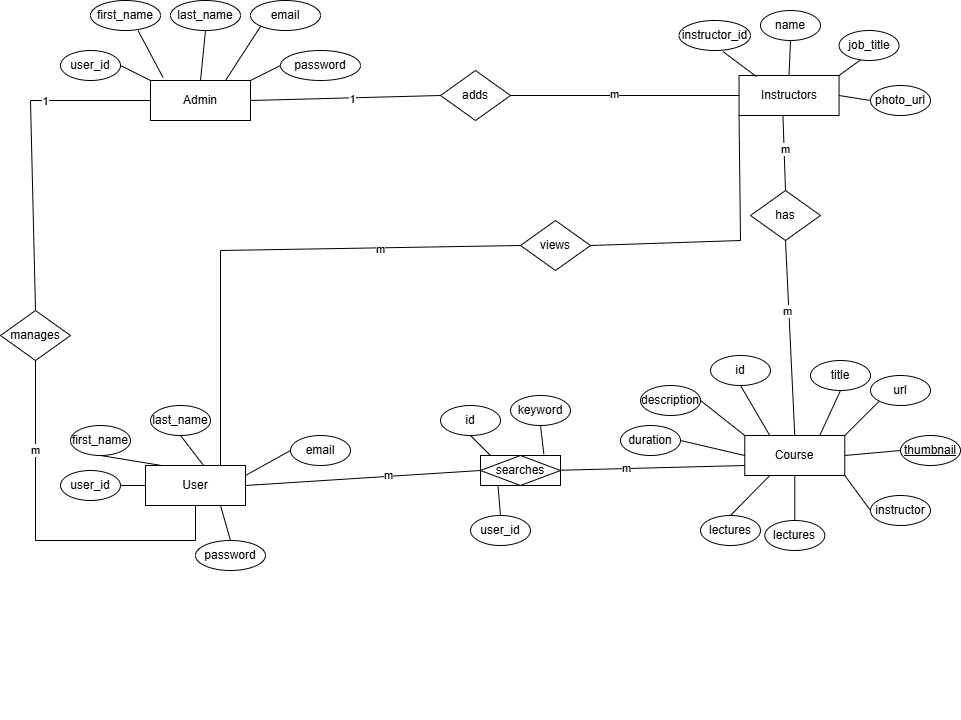
**Table 3.1: Gantt chart Table for Room Rental System**

|  |  |
| --- | --- |
| **Task Name** | **Duration** |
| Getting Started | 1 week |
| Requirement Gathering | 1 week |
| System Design | 2 week |
| Implement | 2 week |
| Development | 3 week |
| Setting up environment | 4 week |
| System Testing | 5 week |
| Monitoring and Analytics | 6 week |
| Execution | 7 week |
| Maintenance | 8 week |



**Figure 3.1.3.1: Gantt chart for Course Recommender System**

### 3.1.3 Data modelling: ER Diagram



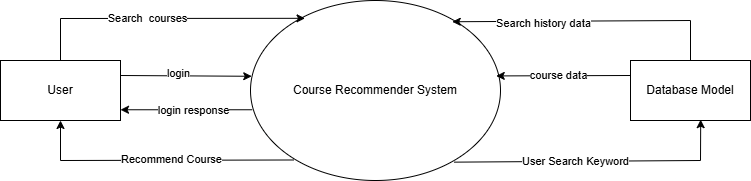
**Figure 3. 2 ER-Diagram**

### 3.1.4 Process Modelling: DFD

Data Flow Diagram consists of two levels of DFD which are context diagram and level 1 DFD. This is used to make the system.

**Level 0: Context DFD**

In Level 0 DFD, login request, prediction, registration is input for LLPS and user request for login and register and admin request for login. User will perform prediction by giving in some specification.



**Figure 3.3 Level 0 DFD**

**Level 1 DFD:**

In level 1 DFD there are three processes where login is responsible for login of admin and users. Prediction is responsible for user prediction. Registration is responsible for user registration. There are three entities Admin, user and system and two data stores which are users and predictions.

**Figure : 3.4 Level 1 DFD**

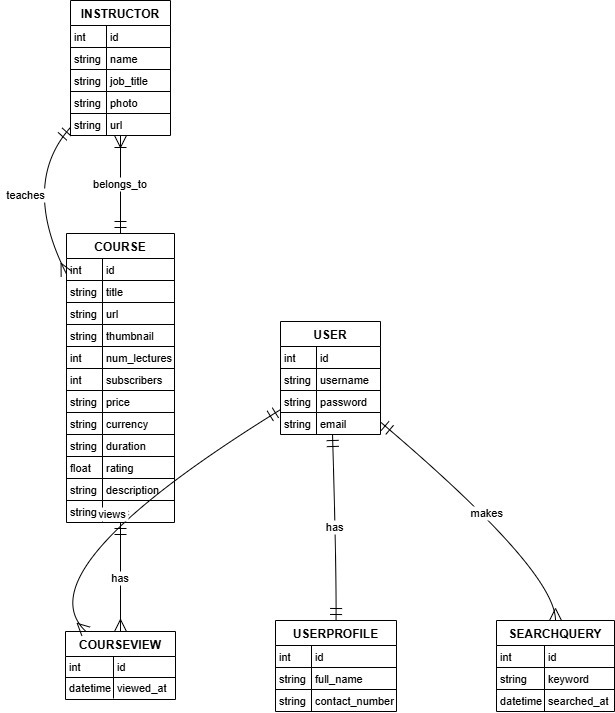
## 3.2 System Design

System design involves defining the architecture, components, and interfaces of a system to meet specified requirements. It includes the planning of both hardware and software elements to ensure they work together efficiently and effectively.

### 3.2.1 Architectural design

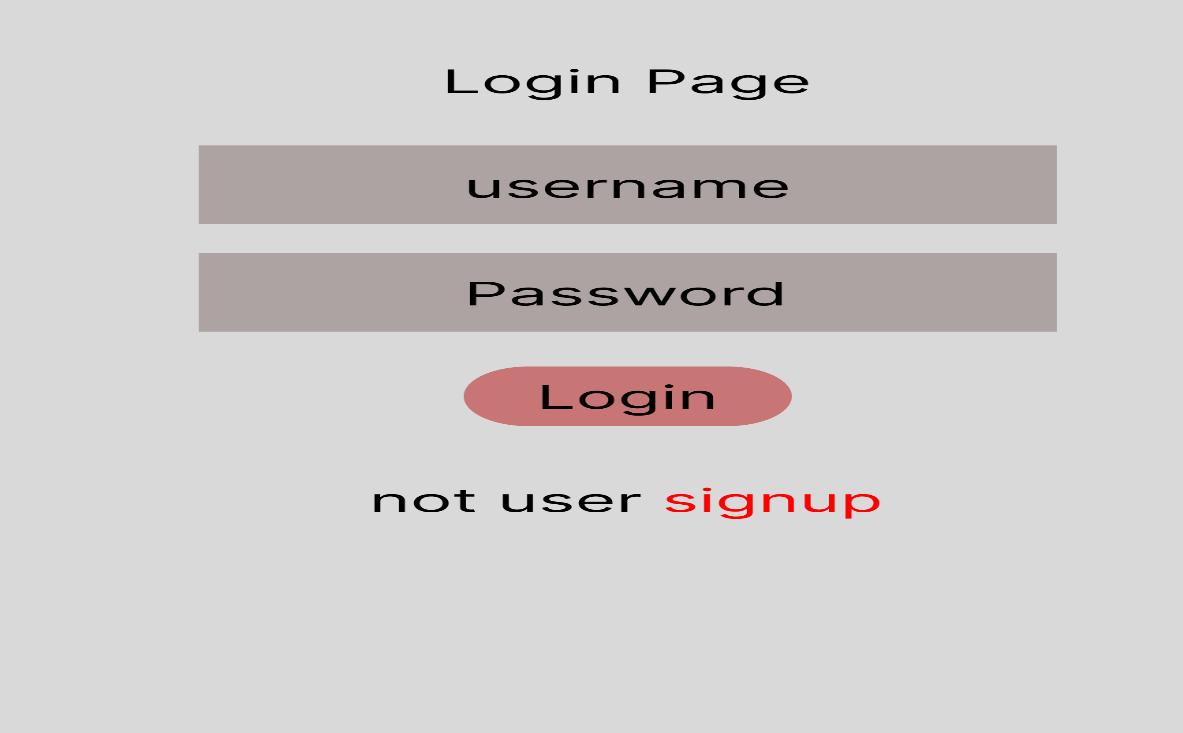
The Course Recommendation System follows a modular and scalable architecture based on the Model-View-Controller (MVC) pattern, commonly used in Django applications. It consists of four primary layers:

* **Presentation Layer (Frontend - UI/UX):** The presentation layer is responsible for delivering an interactive and user-friendly interface for the Course Recommendation System. It includes dynamic elements such as search functionality, course listings, trending searches, and personalized recommendations. Built with HTML, TailwindCSS, and JavaScript (Swiper.js for sliders), the frontend ensures a modern and responsive design. Using Django’s templating engine (Jinja2), the system dynamically renders course details, recommendations, and user-specific content. The interface is designed to be intuitive, enabling users to search, explore, and interact with recommended courses efficiently.
* **Application Layer (Django Views & Business Logic):** The application layer serves as the backend logic of the system, handling user authentication, search queries, course retrieval, and recommendation generation. It is implemented using Django’s Model-View-Template (MVT) architecture, where views process user requests and render dynamic pages. This layer includes functions such as search\_view(), home\_view(), and course\_detail\_view(), which fetch and display relevant data. Business logic like user-based and item-based recommendations is processed within the views, ensuring that the system dynamically adapts to user interactions.
* **Data Layer (Database Models & Storage):** The data layer manages structured storage for users, courses, instructors, search queries, and course views. Implemented using Django ORM with PostgreSQL or SQLite, it efficiently stores and retrieves data while maintaining relational integrity. The key models include Course, Instructor, CourseView, and SearchQuery, ensuring that the system tracks user interactions, trending searches, and course details. By storing search history and viewed courses, the database plays a crucial role in powering personalized recommendations.
* **Recommendation Engine (AI-Based Course Recommendations):** The recommendation engine forms the core of the system, leveraging machine learning and filtering techniques to provide personalized course suggestions. It employs User-Based Filtering (search history analysis), Item-Based Filtering (co-viewed courses), and Content-Based Filtering (topic similarity) to generate relevant recommendations. Functions like get\_user\_based\_recommendations() and get\_similar\_courses() analyze stored data to suggest courses most relevant to the user's interests. Additionally, the system identifies trending searches and frequently viewed topics, making it an intelligent and adaptive learning platform.3.2.2 Database Schema design



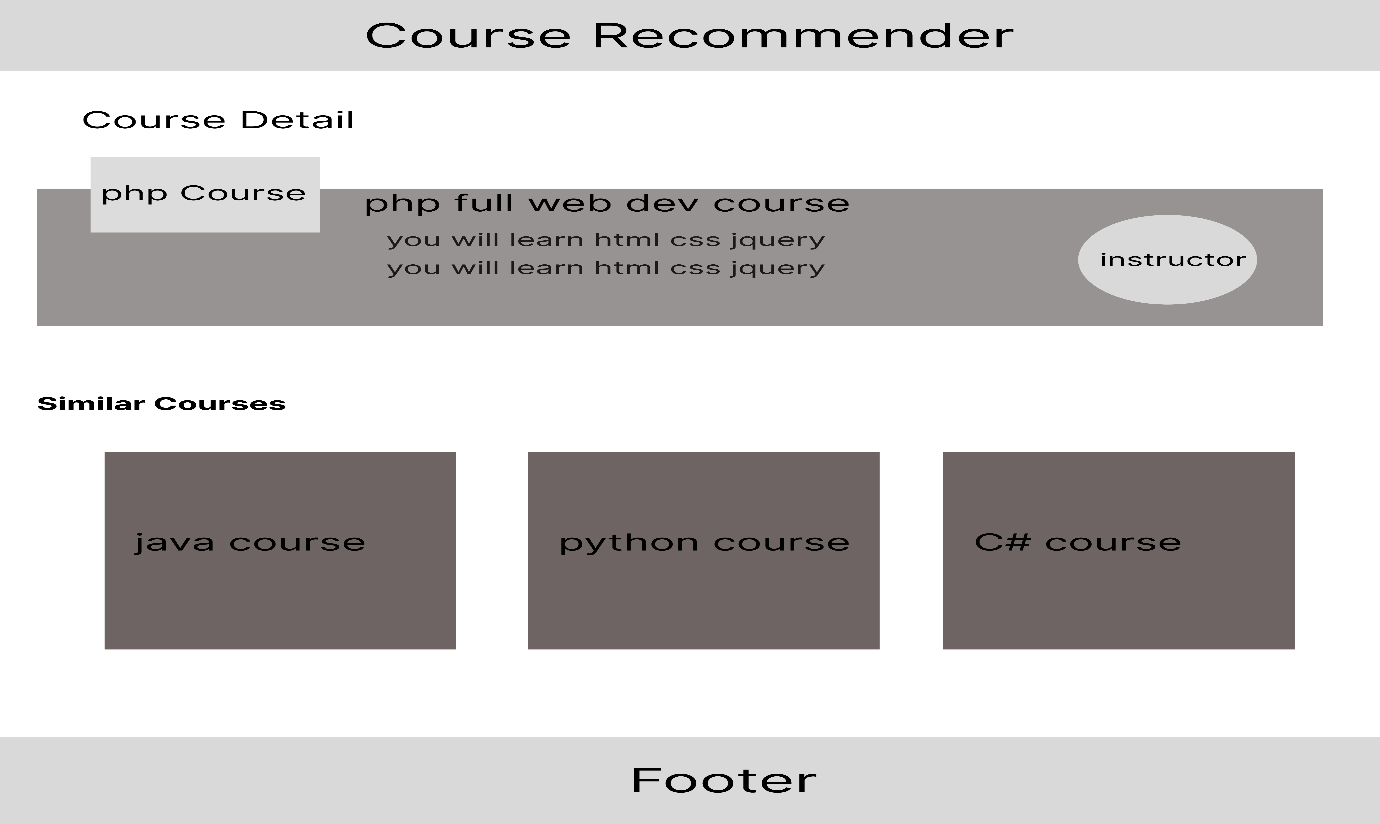
**Figure 3. 6 Database Schema design**

### 3.2.3 Interface design (UI/UX)



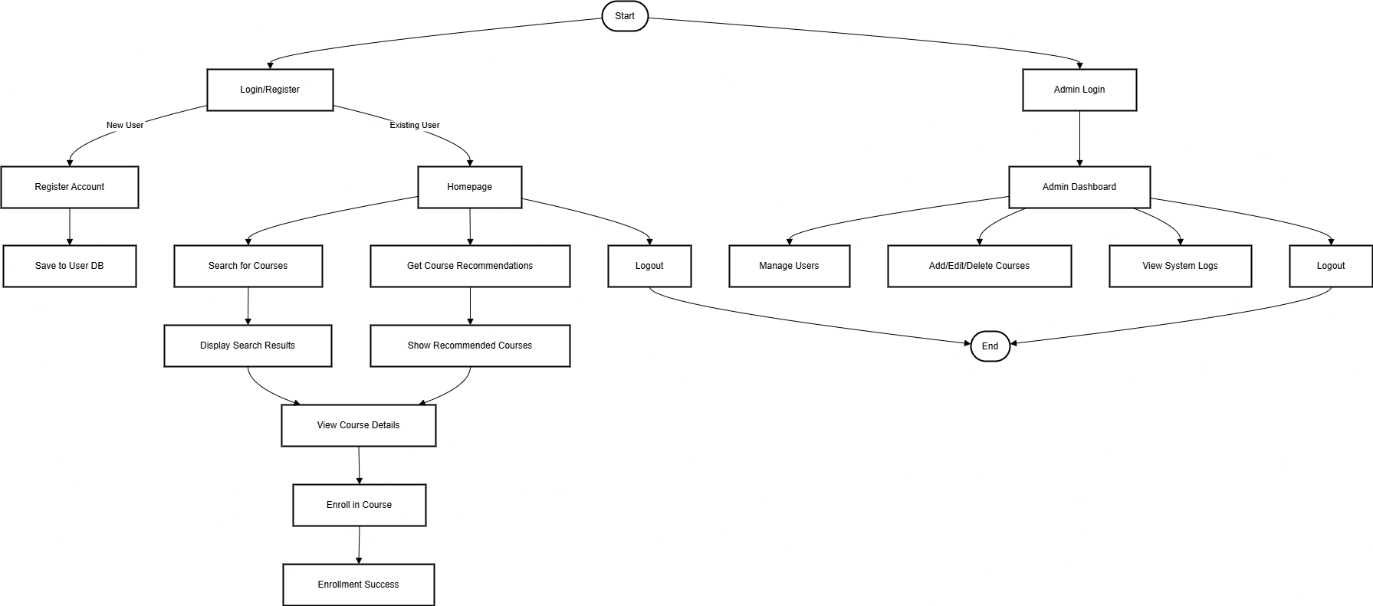
A screenshot of a phone

AI-generated content may be incorrect.



**Figure 3. 7 Interface design (UI/UX)**

### 3.2.4 Physical DFD

****

**Figure 3. 3 Physical DFD**

## 3.3 Algorithm details

Content Based Filtering

Content-based filtering is a recommendation algorithm that suggests items to users based on the characteristics of items they have previously interacted with. It relies on analyzing item features and creating a user profile based on past behavior, such as ratings, clicks, or purchases. Each item is represented using attributes, and the system compares these attributes to recommend similar items. For example, in an online course recommendation system, if a user has previously taken a Python programming course, the algorithm might recommend courses related to data science, machine learning, or web development using Python. By analyzing the textual descriptions, categories, and other metadata of items, content-based filtering ensures that users receive recommendations tailored to their specific interests.

One of the core techniques used in content-based filtering is similarity measurement, where the system calculates how closely related two items are based on their attributes. Common similarity metrics include cosine similarity, which measures the angle between item feature vectors, and TF-IDF (Term Frequency-Inverse Document Frequency), which helps process textual descriptions by identifying important keywords. The recommendation process involves creating a user preference profile by aggregating the features of items they have engaged with. When a user interacts with a new item, the system updates their profile and refines future recommendations, making the algorithm more accurate over time. Unlike collaborative filtering, which depends on other users' preferences, content-based filtering focuses solely on individual user behavior and item attributes.

**Pseudo Code**

**User-Based Recommendation (Search History-Based)**

Function get\_user\_based\_recommendations(user, num\_recommendations):

// Step 1: Retrieve user's most searched keywords

user\_searches = Get search queries from database where user = user

Count occurrences of each keyword and sort in descending order

// Step 2: Check if user has search history

If user\_searches is empty:

Return None // No recommendations available

// Step 3: Get top searched keywords

top\_keywords = Extract top 2-3 keywords from user\_searches

// Step 4: Fetch courses related to the keywords

recommended\_courses = Get courses where title OR description contains top\_keywords

Sort courses by rating

Select num\_recommendations courses randomly

// Step 5: Return the recommended courses

Return recommended\_courses

**Item-Based Recommendation (Course View-Based)**

Function get\_item\_based\_recommendations(user, num\_recommendations):

// Step 1: Identify courses most frequently viewed by the user

viewed\_courses = Get courses from CourseView table where user = user

Count occurrences of each course and sort in descending order

// Step 2: Find the most viewed course by the user

If viewed\_courses is empty:

Return None // No recommendations available

most\_viewed\_course = Select the top viewed course from viewed\_courses

// Step 3: Fetch similar courses

similar\_courses = Get courses that were viewed by users who also viewed most\_viewed\_course

Rank courses by co-view count

Select top num\_recommendations courses

// Step 4: Return the recommended courses

Return similar\_courses

**Trending Search-Based Recommendation**

Function get\_trending\_searches\_with\_courses(limit, course\_limit):

// Step 1: Identify top trending search keywords

trending\_keywords = Get all search queries

Count occurrences of each keyword and sort in descending order

Select top limit keywords

trending\_data = Empty List

// Step 2: Fetch courses related to trending keywords

For each keyword in trending\_keywords:

related\_courses = Get courses where title OR description contains keyword

Sort courses by rating

Select top course\_limit courses

// Step 3: Store keyword and related courses

trending\_data.append({ "keyword": keyword, "courses": related\_courses })

// Step 4: Return trending searches with courses

Return trending\_data

# CHAPTER 4: IMPLEMENTATION AND TESTING

## 4.1 Implementation

### 4.1.1Tools Used:

* **CASE Tools:** Figma, Canva and Draw.io were used for creating and managing deployment diagrams. These tools provide robust features for designing and visualizing system architectures.
* **Programming Languages:** The application is built using Python for the backend with Flask for handling web requests and managing server-side logic, HTML and CSS for the frontend, and JavaScript for dynamic functionalities. These languages are essential for developing and integrating various system components.
* **Database Platforms:** SQLite is used as the database platform. It is a lightweight, file-based database that supports SQL queries and efficiently handles data storage needs.
* **Dataset:** The data used for the application was sourced from Udemy, a well-known platform for studying.

### 4.1.2 Implementation details of modules

Course Management Module:

This module manages courses and instructors and stores relevant course details.

* Models:
  + Course: Stores course details like title, description, instructor, rating, price, etc.
  + Instructor: Stores instructor details like name, photo, and profile link
* Views:
  + course\_detail\_view(request, course\_id): Displays the course details.
  + instructor\_courses\_view(request, instructor\_id): Displays all courses by a specific instructor.
* Templates:
  + course\_detail.html → Displays course details
  + instructor\_courses.html → Lists courses by an instructor

Course Recommendation Module

Provides personalized course recommendations based on user behavior.

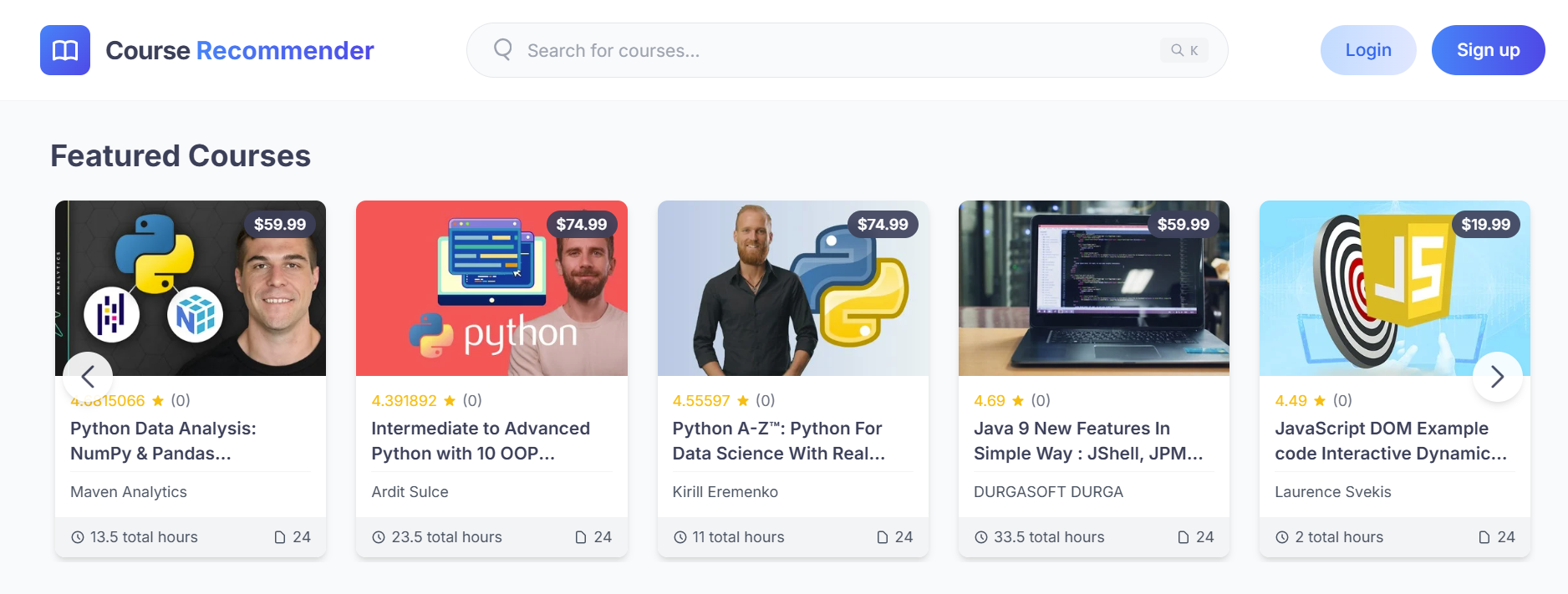
* Models
  + CourseView: Stores which user viewed which course.
* Views:
  + get\_user\_based\_recommendations(user): Suggests courses based on user’s past searches.
  + get\_item\_based\_recommendations(user): Suggests courses based on user’s most viewed course.
  + get\_similar\_courses(course): Suggests courses frequently viewed together.
* Templates:
  + home.html → Displays recommended courses.
  + course\_detail.html → Displays similar courses.

Search and Trending Module

Handles course search functionality and tracks trending searches.

* Models:
  + SearchQuery: Stores search keywords along with the user who searched.
* Views:
  + search\_view(request): Handles search queries and retrieves relevant courses.
  + get\_trending\_searches\_with\_courses(): Identifies top searched keywords and retrieves related courses.
* Templates
  + search\_results.html → Displays search results and trending searches.
* Logic for Trending Searches:
  + Uses SearchQuery table to count occurrences of each keyword.
  + Retrieves top 5 searched keywords and fetches related courses.

# 5. APPENDICES



A screenshot of a phone

AI-generated content may be incorrect.

A blue background with white text

AI-generated content may be incorrect.

A screenshot of a login page

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

# 

# References

|  |  |
| --- | --- |
| [1] | W. &. S. J. Xing, "Personalized learning in online education: The role of recommendation systems," *Journal of Educational Technology,* pp. 45-60, 2021. |
| [2] | H. L. Y. &. W. Z. Tang, "AI-powered recommendations in e-learning platforms: A case study of Coursera," *International Journal of Artificial Intelligence in Education,* pp. 112-130, 2020. |
| [3] | J. &. B. K. Smith, Recommender Systems: Theories, Applications, and Challenges, Academic Press, 2018. |
| [4] | N. I. o. E. Research, "Data-Driven Approaches to Enhancing Student Engagement," Educational Analytics Board, 2020. |
| [5] | I. S. f. L. T. (ISLT), "The Impact of Artificial Intelligence on Course Recommendations," ISLT Research Division, 2021. |
| [6] | R. &. G. M. Patel, Data Science for Personalized Learning, Springer Publications, 2020. |