`

**Tribhuvan University**

**Faculty of Humanities and Social Sciences**

A PROJECT Proposal

ON

**Course Recommender System**

**Submitted to**

**Department of Computer Application**

**Patan Multiple Campus**

**Patandhoka, Lalitpur**

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

**Submitted by**

Rastrika Maharjan

BCA 6th Semester

Roll: 077/50

# 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

Contents

[ABSTRACT ii](#_Toc171231957)

[Introduction 1](#_Toc171231958)

[Problem Statement 1](#_Toc171231959)

[Objective: 1](#_Toc171231960)

[Algorithm Implementation:- 2](#_Toc171231961)

[Methodology 2](#_Toc171231962)

**A.** [Requirement Identification 2](#_Toc171231963)

[A. Functional Requirement:- 3](#_Toc171231964)

[B. Non-Functional Requirement 3](#_Toc171231965)

[C. Data Requirements 4](#_Toc171231966)

[D. Use case diagram 4](#_Toc171231967)

[B. Feasibility Study 5](#_Toc171231968)

[A. Technical Feasibility 5](#_Toc171231969)

[B. Operational Feasibility 5](#_Toc171231970)

[C. Economic Feasibility 5](#_Toc171231971)

[D. High level Design of system 6](#_Toc171231972)

[E. System Development Implementation tools 6](#_Toc171231973)

[F. System Analysis and Design 7](#_Toc171231974)

[Gantt chart 7](#_Toc171231975)

[Expected Outcome 8](#_Toc171231976)

[References: 9](#_Toc171231977)

# 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.

# 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.

# Objective:

The main objectives of Used Course Recommender 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.

# Algorithm Implementation:-

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

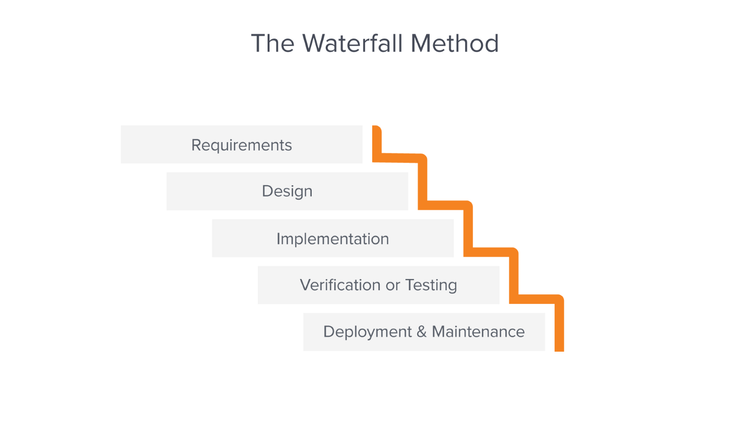
trending\_data.append({ "keyword": keyword, "courses": related\_courses })

// Step 4: Return trending searches with courses

Return trending\_data

## 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**

# Requirement Identification

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

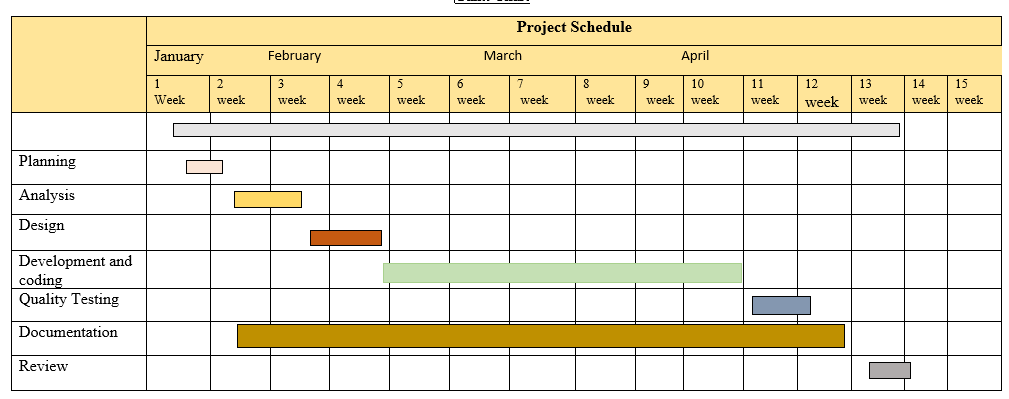
The system is designed to be user-friendly and intuitive, ensuring that users can search for courses effortlessly and receive personal recommendations. The Trending Searches feature enhances engagement by highlighting popular course topics, helping users explore relevant content. Since the system operates 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, SQLite , 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.

# Gantt chart

A Gantt chart is a project management tool assisting in the planning and scheduling tool assisting in the planning and scheduling of projects of all sizes, although they are particularly useful for simplifying complex projects. Projects management timelines and tasks are converted into a horizontal bar chart, showing start and end dates, as well as scheduling and deadlines, including how much of the task is completed per stage and who is the task owner. This is useful to keep tasks on track when there is a large team and multiple stakeholders when the scope changes.



**Fig:3 Gantt Chart**

# Expected Outcome

The expected outcome of this project is a personalized course recommendation system that enhances user experience by suggesting relevant courses based on their search history, viewed courses, and trending topics. The system will intelligently analyze user interactions, utilizing content-based filtering and collaborative filtering techniques to generate recommendations that align with user preferences. Logged-in users will receive tailored recommendations based on their most searched keywords and frequently viewed course topics, while anonymous users will have access to trending courses based on overall platform activity. By integrating machine learning and data-driven insights, the system aims to improve user engagement, facilitate informed decision-making, and optimize course discovery, making it easier for learners to find the most suitable courses

# Conclusion:

The course recommendation system effectively leverages user search behavior, viewing patterns, and trending topics to provide personalized course suggestions. By integrating content-based filtering and collaborative filtering, the system ensures that users receive recommendations tailored to their interests, enhancing their learning experience. The dynamic approach of storing user interactions and analyzing keywords allows the system to continuously refine its recommendations, making course discovery more intuitive and efficient. Additionally, the implementation of trending searches and featured courses ensures that both logged-in and anonymous users benefit from relevant suggestions. This project demonstrates the potential of data-driven recommendation systems in improving user engagement, simplifying course selection, and making e-learning platforms more interactive and user-friendly

# References:

[1] About the e-learning detail <https://onlinelibrary.wiley.com/journal/13652729>

[2] Basic Concept and specific methodology detail <https://elearningindustry.com/>

[3] G. Qadah,"Requirements, design and implementation of an informative website.," 2005.https://www.researchgate.net/publication/Requirements\_design\_and\_imp

[5] "slideshare.ne," https://www.slideshare.net/steve2434/-41010890.

[6] "apm.org.uk," https://www.apm.org.uk/.

[5] "en.wikipedia.org," https://en.wikipedia.org/wiki/SmartDraw.