A Project Report on

Integrating Recommendation System for creative Learning Web Framework

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

Bachelor of Engineering

in

Information Technology

by

Ruchita Raut(19104033)
Kushal Todi(19104047)
Pratik Dhumal(19104031)

Under the Guidance of

Prof. Geetanjali Kalme

Department of Information Technology
NBA Accredited
A.P. Shah Institute of Technology
G.B.Road,Kasarvadavli, Thane(W), Mumbai-400615
UNIVERSITY OF MUMBAI

Academic Year 2022-2023

Approval Sheet

This Project Report entitled ***"Integrating Recommendation System for creative Learning Web Framework"*** Submitted by ***"Pratik Dhumal"(19104031), "Kushal Todi"(19104047), "Ruchita Raut"(19104033)*** is approved for the partial fulfillment of the requirenment for the award of the degree of ***Bachelor of Engineering*** in ***Information Technology*** from ***University of Mumbai***.

Prof. Geetanjali Kalme
Guide

Dr. Kiran Deshpande
Head Department of Information Technology

Place:A.P.Shah Institute of Technology, Thane
Date:

CERTIFICATE

This is to certify that the project entitled ***"Integrating Recommendation System for creative Learning Web Framework"*** submitted by ***"Ruchita Raut" (19104033), "Kushal Todi" (19104047), "Pratik Dhumal" (19104031)*** for the partial fulfillment of the requirement for award of a degree ***Bachelor of Engineering*** in ***Information Technology***,to the University of Mumbai,is a bonafide work carried out during academic year 2022-2023.

Prof. Geetanjali Kalme
Guide

Dr. Kiran Deshpande	Dr. Uttam D.Kolekar
Head Department of Information Technology	Principal

External Examiner(s)

1.

2.

Place:A.P.Shah Institute of Technology, Thane
Date:

Acknowledgement

We have great pleasure in presenting the report on **Integrating Recommendation System for creative Learning Web Framework.** We take this opportunity to express our sincere thanks towards our guide **Prof. Geetanjali Kalme** Department of IT, APSIT thane for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards his constant encouragement, support and guidance through the development of project.

We thank **Dr. Kiran B. Deshpande** Head of Department,IT, APSIT for his encouragement during progress meeting and providing guidelines to write this report.

We thank **Prof. Sonal Jain** BE project co-ordinator, Department of IT, APSIT for being encouraging throughout the course and for guidance.

We also thank the entire staff of APSIT for their invaluable help rendered during the course of this work. We wish to express our deep gratitude towards all our colleagues of APSIT for their encouragement.

Kushal Todi
(19104047)

Ruchita Raut
(19104033)

Pratik Dhumal
(19104031)

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Kushal Todi (19104047)

Ruchita Raut (19104033)

Pratik Dhumal (19104031)

Date:

Abstract

The Internet has an enormous amount and variety of educational resources. As multifaceted educational interest advances, it is becoming increasingly important to support students' course choices. To resolve the issue of sparse and low-quality data in the recommendation systems, we have presented a hybrid approach recommendation system for courses using the KNN i.e. K Nearest Neighbors algorithm. In order to assist students in making decisions, these systems consider the preferences and interests of each student, ratings of each course, and then suggest courses based on how similar they are. Furthermore, both the user behavior and the text vector record are used to improve the calculation of course similarity. This paper also discusses the design of an architecture for online education recommender systems, with a focus on promoting online creative learning courses and web-based creative learning material. Through data analysis, a comparison between traditional recommendation system algorithms will be discussed in further sections. The system employs hybrid recommendation approaches from an algorithmic standpoint. On creative learning sites, our designed system will be capable to provide effective recommendations to learners.

Contents

List of Figures

List of Tables

List of Abbreviations

KNN:	K-Nearest Neighbors Algorithm
CF:	Collaborative Filtering
CBF:	Content–Based Filtering
DB:	Database
ML::	Machine Learning
GUI:	Graphical User Interface

Chapter 1

Introduction

Creative courses-based recommendation systems are a tool designed to assist users find and choose the classes that are most appropriate for their needs and interests. Online learning platforms have gained popularity in recent years and have made a variety of courses available. It might be difficult for consumers to locate courses that match their learning objectives and tastes with so many selections accessible.

As big data has grown, data analysis has become more important for improving online learning tools.In simple terms, it is an algorithm that suggests useful products to users. Due to the recent sharp growth in the number of courses offered within the context of smart education, the problem of choosing the appropriate courses now plays a significant part in the contemporary educational process. Through efficient analysis and processing of the available data, accurate course recommendation services are offered to online education platforms and users, and high-quality courses are made available so that more students can engage in learning at any time and anywhere and that their capacity for independent learning is increased, acknowledging the vital significance of top-notch online education.

Scalability is one of the major problem when dealing real-world datasets for any recommendation system. As user-item interacts, ratings and reviews generate a significant amount of changing data. Also, due to incomplete course descriptions, it may be difficult to completely comprehend the relative relevance of several courses. Certain classes in that subject may be more relevant than others that are not immediately evident to the student, resulting in the student making poor judgements once more, when students desire to pick courses closely related to their particular career path. Systems or engines that generate lists of recommendations are known as recommendation algorithms. The primary algorithm of the recommendation system compares and evaluates the affinity between two items: the user-selected item and the rest of the catalog items. This fundamental mathematical metric is called "similarity".

The user is shown the "You may also enjoy" list of products that have high similarity values to the ones they choose. Systems for making recommendations employ a variety of comparable metrics. Two common but established methods of recommendation algorithms are content-based recommendation and collaborative filtering recommendation. Many personalized suggestions employ the collaborative filtering recommendation method, which has no structural limitations for recommendation items.

The method proposed for course recommendation is K-Nearest Neighbour which will overcome the problems that the traditional recommendation methods lacks. The KNN algorithm is used to search the nearest neighbors to a given student based on the similarity of their course preferences. The recommendations are generated by considering the courses that are most frequently taken by the k value i.e. nearest neighbors. It works by searching for the closest neighbors to target the user based on their prior course ratings or behavior and making recommendations in accordance with their preferences. The algorithm considers the preferences of multiple users and overcomes more diverse and personalized recommendations. The KNN algorithm has been evidenced to be effective in generating high-quality recommendations. It is also a flexible method that can be easily combined with other techniques, such as content-based filtering, to enhance the efficiency of the recommendation system.

1.1 Objectives

- To build and provide a user-friendly interface for web-based application using ReactJS.
- To build a collaborative recommendation system on the user's first enrollment(sign up) category selection inputs using the KNN algorithm along with cosine similarity metrics.
- To build a hybrid recommendation system on the basis of the user's past data i.e category selection, courses, and ratings using the KNN algorithm along with cosine similarity metrics.
- To filter recommended/searched courses on the basis of ranking attributes.

Chapter 2

Literature Review

In this literature survey, the existing literature on hybrid course recommendation systems will be evaluated, emphasizing on the methodologies and techniques used in systems, the kind of data analyzed, and the productivity of the recommendations.

According to the paper [2], the author proposed here a course-based recommendation system that was implemented with the concept of a hybrid recommender system. There are benefits and drawbacks to each recommendation method, including collaborative filtering method and content-based filtering method. So, to level up the shot comes of the filtering models, the author combined both models to generate a hybrid recommender system for the Course Recommendation in the Social Learning network. Here, the Lasso algorithm was proposed for content-based filtering while the KNN algorithm was used for collaborative filtering. In order to amalgamate the results from both filtering techniques, a weighted average method was employed to generate a hybrid result from both filtering models.

In this paper [4], Zheng Chen proposed a course-based recommendation system using a collaborative filtering algorithm to help students' decisions. In that system, the improved cosine similarity metric was used, on the basis of the history of students' course records, and better accuracy was obtained in the recommendation process, which aligns with the needs of users. This paper propounds a collaborative filtering optimization algorithm based on text similarity calculation. Experimental findings demonstrate the model's reliability. Compared with collaborative filtering recommendations based on users and items, the accuracy of the recommendation results is greatly improved.

The work proposed by the author in this paper [1], focuses on building a content-based filtering recommendation system that provides numerous elective courses recommendation in flavor to the student's academic history. They proposed recommendation systems' results that are based on pre-processed words from courses recorded by the user. The weighted score of cosine similarity between both courses and the user profiles is assessed. They adopted two methods, a questionnaire approach, and a validation method to examine the suggested recommendation system. The validation method was used to determine the average accuracy and the questionnaire approach was used for the assessment of the system's performance. They got an average of 64% accuracy for the recommendation system.

In this paper [3], the author has proposed the Similarity between clients that are discovered utilizing the ’pearson relationship’ and ’best n-neighbors’ methodology utilized to choose the closest neighbors to the objective understudy structure which is the anticipated course score of the objective.

In [8], the author proposed work aiming at the problems of the rapid increase of learning platforms and course resources, and the poor quality of course recommendation due to the dispersion of online resources, the attention mechanism and deep learning are integrated into the course recommendation problem, and a deep collaborative online learning resources recommendation model focusing on attention is proposed to model the relationship between high-level course sets, which improves the recommendation performance. Although some scholars have improved the collaborative filtering model based on implicit behavior, there is still room for further improvement.

In this paper [6], a new learning resources recommendation system model is proposed, including a course recommendation sub-module based on statistics and a personalized course recommendation sub-module based on professional training requirements.

In paper [9], Raghad Obeidat presented a collaborative recommender system that suggests online courses for students based on the similarities of history students’ courses. The proposed approach uses data mining techniques to identify patterns among the courses.

In the paper [10], Vishal Garg has proposed an effective and efficient Course Recommendation System using Machine Learning. On e-learning platforms, the suggested system will be capable to make useful recommendations to users. The predictions have been generated depending on the ratings of other similar users. The system has been trained to minimize the error using the idea of Gradient Descent.

In reference paper [7], the improved collaborative-filtering algorithm based on a hybrid approach is adopted. By introducing the progressive forgetting curve based on a change in the timeliness of the user interest are better solved. It stated the disadvantages of traditional collaborative filtering algorithm, like it has low efficiency, weak adaptability and novelty rejection.

In the paper [11], a hybrid approach is considered between context-based filtering as well as collaborative filtering to build the system. They declared that this approach overcomes the drawbacks of every individual algorithm and improves the efficiency of the system. Techniques like Clustering, Similarity and Classification are used to get better recommendations thus increasing precision and accuracy.

According to the paper [12], the author focused that a study be conducted on the creation and development of a recommender system for graduate admission seekers that can help students choose graduate programs that match their entire educational profiles. Using Academics data of top performer students who already had the opportunity to study abroad, the system improved a method to transform relational databases for students of all kinds of associated information into universal database format. Using N-related students, the K-nearest Neighbor method was used to identify the top N test takers and advise the top K universities.

Chapter 3

Project Design

3.1 Existing System

Existing research on recommendation systems has focused on addressing the sparsity and cold start problems. To solve these challenges, methods based on fuzzy reasoning and genetic algorithms have been used. Fuzzy reasoning allows for reasoning with uncertain or imprecise information, while genetic algorithms generate diverse and novel recommendations. However, these methods often ignore the fuzziness of users' choices and assume preferences can be accurately represented by numerical ratings or binary indicators. Some of the existing recommendation systems for course selection also typically rely on either content-based or collaborative filtering methods. Content-based methods rely on the similarity of course features, while collaborative filtering uses the preferences of other users to recommend courses. However, these methods have limitations, such as the inability to handle new courses or new users (cold start problem) and the sparsity of user data.

3.2 Proposed System

We propose a hybrid recommendation system that combines collaborative and content-based filtering methods to suggest courses to students. The system identifies similar students based on their course history, ratings and understanding of domain topics and analyzes their course selections to recommend relevant courses. The system also uses information filtering techniques to refine the data for better recommendations.
Along with the hybrid system, we have developed a collaborative filtering approach to identify the initial interests of users without their past data. By combining these two approaches, we can provide more personalized and relevant recommendations for students' course selections, aligned with their career goals. Overall, our proposed system aims to provide an efficient and effective way to suggest courses to students.

3.2.1 Data Collection

The data that will be utilised to train the model is the primary and an essential need for using any recommendation system. Therefore, data collection is the most initial and a crucial step for building any recommendation engine. A google form was created with fields like Name, Course Categories for Creative learning the Platform, ratings, email id and phone number. The form was circulated in our college in all the departments to collect large amount of data. The ratings are taken as a basis for implementing Collaborative filtering in our hybrid recommendation system part.

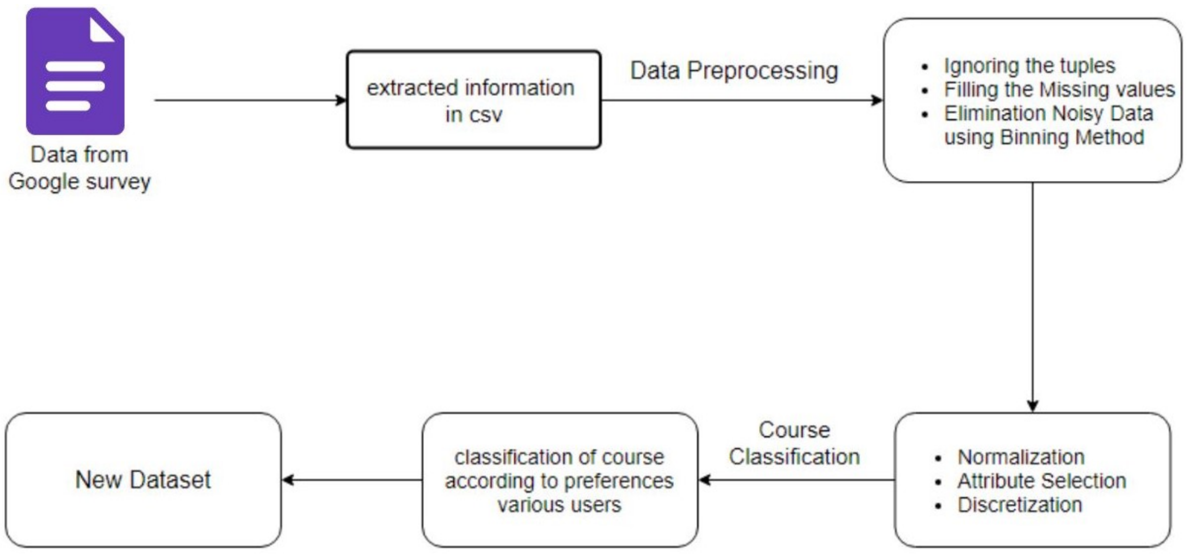

Data from Google survey
extracted information in csv
Data Preprocessing
• Ignoring the tuples
• Filling the Missing values
• Elimination Noisy Data using Binning Method
• Normalization
• Attribute Selection
• Discretization
Course Classification
classification of course according to preferences various users
New Dataset

Figure 3.1 - Overview of Creating Course Dataset

3.2.2 Data Pre-Processing

The data collected was not shaped, which contain missing values, unwanted columns and so forth. It is important to pre-process the information before further handling. In Data Cleaning, undesirable information (traits, tuples not required) are expelled, missing information are filled by supplanting it with rough worth or normal of the considerable number of data, etc. Converting the timestamps into a standardized date format Normalizing the course categories name data collected (e.g., convert them to lowercase and remove any redundant whitespaces).

3.2.3 Proposed Model

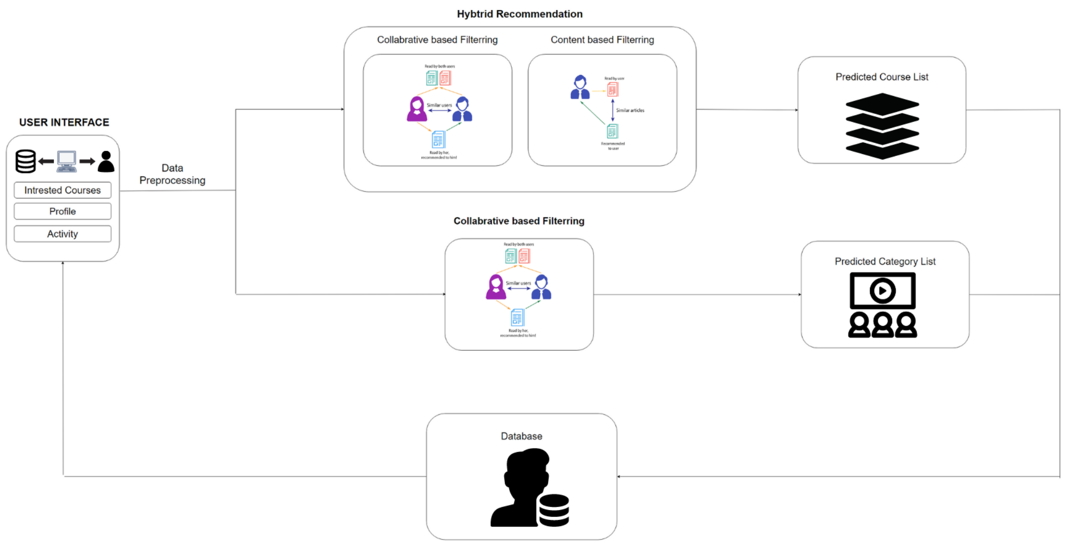


Figure 3.2 - Architecture Diagram for complete Recommendation System

In our first model, Content-based filtering part, in our hybrid course recommendation system depicts each course as a vector of its features. For instance, YouTube suggests videos depending on the ones you've watched. It ascertains the cosine similarity between each course's vector and the user's previous course preference vectors. To recommend to the user, pick the K courses having the highest cosine similarity.
The cosine similarity between the two vectors a and b can be defined as:

$$cosine\_similarity(a, b) = \frac{a \cdot b}{||a|| \cdot ||b||}$$

where '.' represents the dot product, and $||a||$ and $||b||$ represent the Euclidean norm of vectors a and b, respectively.

In the collaborative filtering part, it represents each user as a vector of their past course enrolments or ratings. For instance, Amazon displays suggestions based on the wishlist of the user. Between the user's vector and the vectors of other users who have taken comparable courses, the cosine similarity score is calculated. KNN is used to identify the K users who are the most similar to the present user and to suggest courses that those users have taken or highly rated.
In our second model, i.e. Collaborative recommendation system is built to get the initial interests of the user based on the course categories. This is done when there is no past data of the user available.

3.2.4 System Diagrams

- Use Case Diagram

The use case diagram below shows different actors as register user and new user. The relation between the actors and what they do with the system. The recommended courses will be evaluated on the basis of the category preferences and ratings of the user. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements.

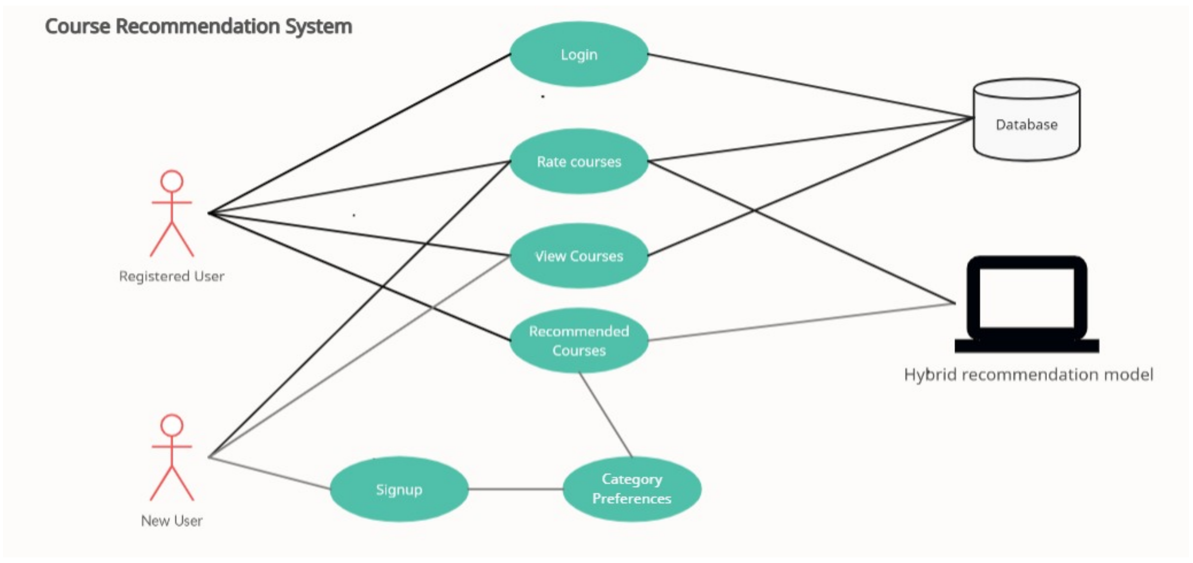


Figure 3.3 - Use Case Diagram

- Activity Diagram

The below activity diagram shows the flow of the system operations. When the user signs up as a new user he selects course categories in which he is interested which gets updated in the database and gives personalized recommendations to the user. When the user logs in to the system he can rate the courses which will again update the ratings database and will recommend the trending courses. And all these recommendations will be displayed on the user interface.

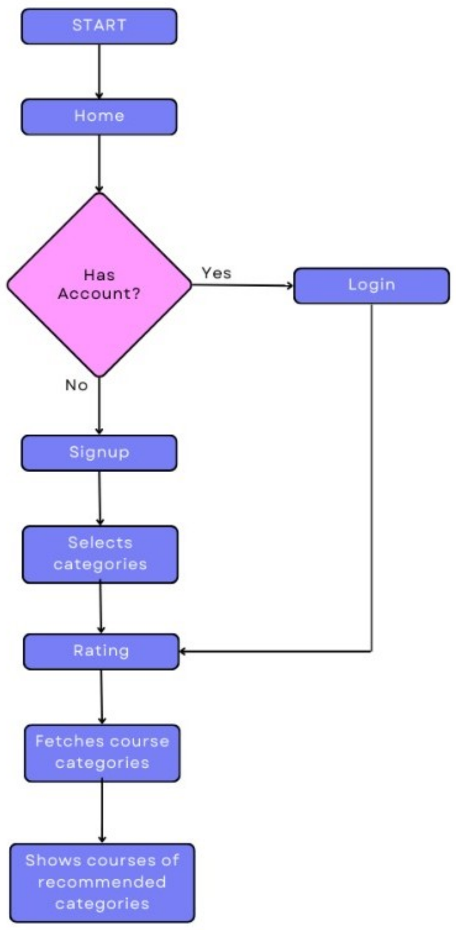

START
Home
Has Account?
Yes
Login
No
Signup
Selects categories
Rating
Fetches course categories
Shows courses of recommended categories

Figure 3.4 - Activity Diagram

Chapter 4

Project Implementation

4.1 Implementation

First, the User/Student will have to first Sign In/Sign Up. After this only user will get to access the courses website. As soon as, the student logs in to the website a pop-up will show up, listing the categories of creative courses and the user has to select either one or multiple categories he/she is interested in and then to proceed he/she should click on "Save Changes".

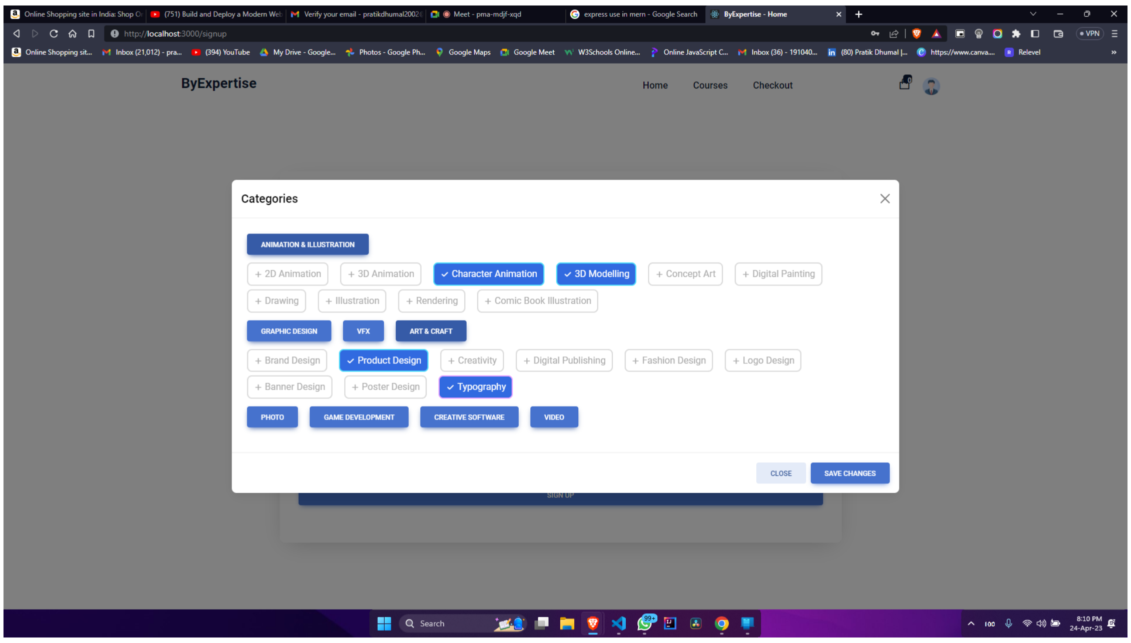
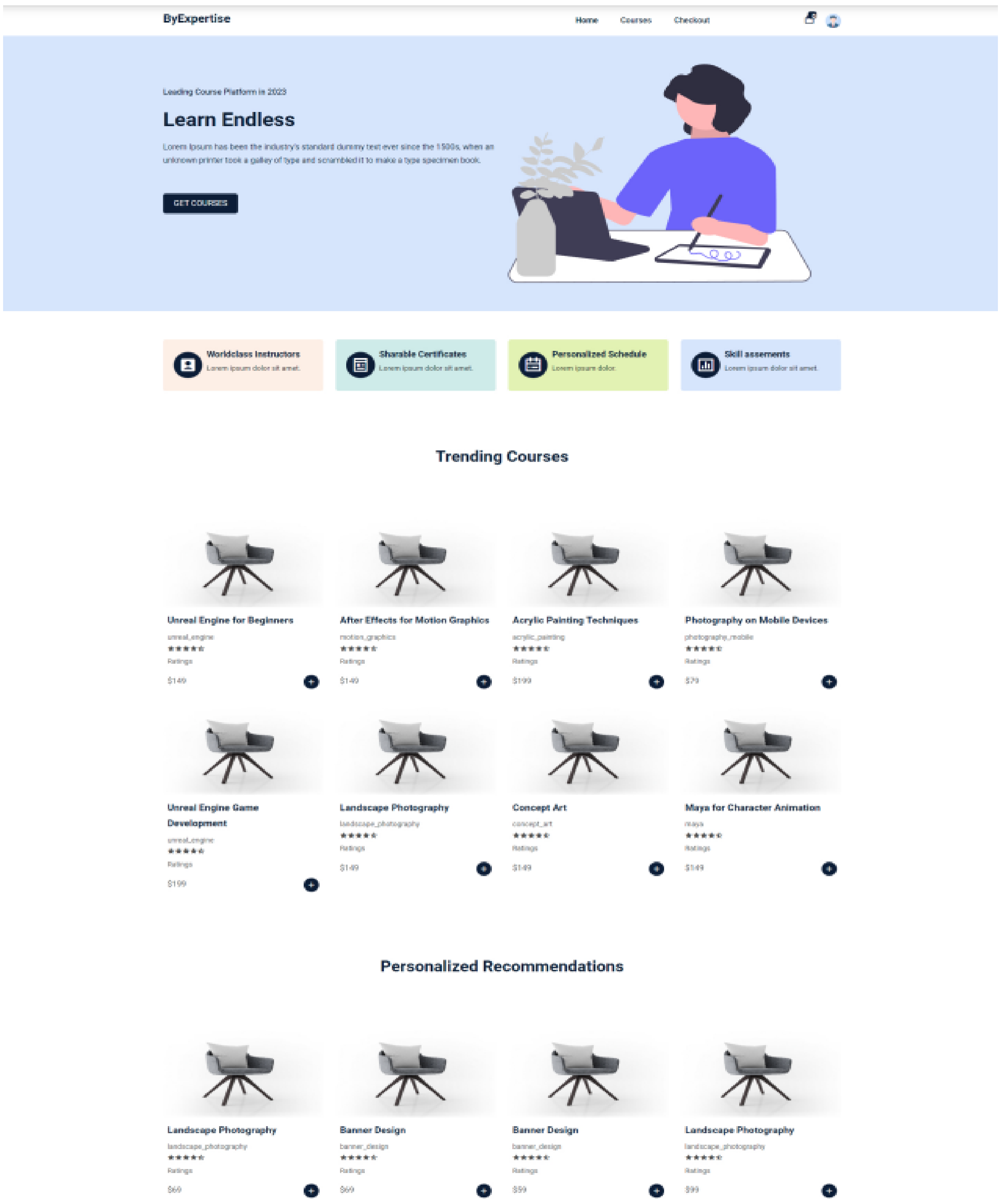


Figure 4.1 - User Input for initial course category selection

Based on the categories selected by the user, the courses will be recommended to the user for the first-time login process. If the same user login afterwards, the pop-up window will not show-up, pop-up will prompt only for the new users. Based on the user interests it will then recommend courses. If the user log in again, based on the past course selection and ratings the user will get the new recommended courses.

In Figure 4.2 the Trending courses will show the courses recommended based on the ratings of users and the personalized Recommendations will show the courses the for the categories selected by the users.


ByExpertise

Home Courses Checkout

Leading Course Platform in 2023

Learn Endless

Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

GET COURSES

Worldclass Instructors
Lorem ipsum dolor sit amet.

Sharable Certificates
Lorem ipsum dolor sit amet.

Personalized Schedule
Lorem ipsum dolor.

Skill assements
Lorem ipsum dolor sit amet.

Trending Courses

Unreal Engine for Beginners
unreal_engine
★★★★★
Ratings
$149

After Effects for Motion Graphics
motion_graphics
★★★★★
Ratings
$149

Acrylic Painting Techniques
acrylic_painting
★★★★★
Ratings
$199

Photography on Mobile Devices
photography_mobile
★★★★★
Ratings
$79

Unreal Engine Game Development
unreal_engine
★★★★★
Ratings
$199

Landscape Photography
landscape_photography
★★★★★
Ratings
$149

Concept Art
concept_art
★★★★★
Ratings
$149

Maya for Character Animation
maya
★★★★★
Ratings
$149

Personalized Recommendations

Landscape Photography
landscape_photography
★★★★★
Ratings
$69

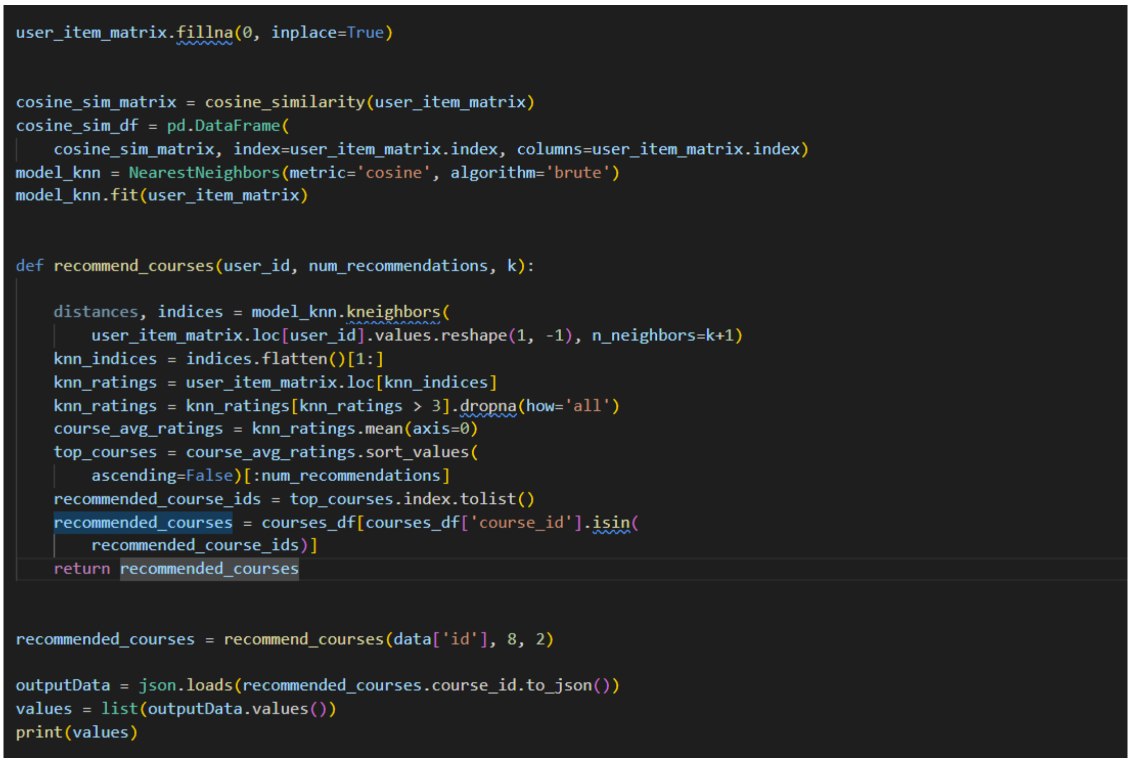
Banner Design
banner_design
★★★★★
Ratings
$69

Banner Design
banner_design
★★★★★
Ratings
$59

Landscape Photography
landscape_photography
★★★★★
Ratings
$99

Figure 4.2 - Display of Recommended Courses

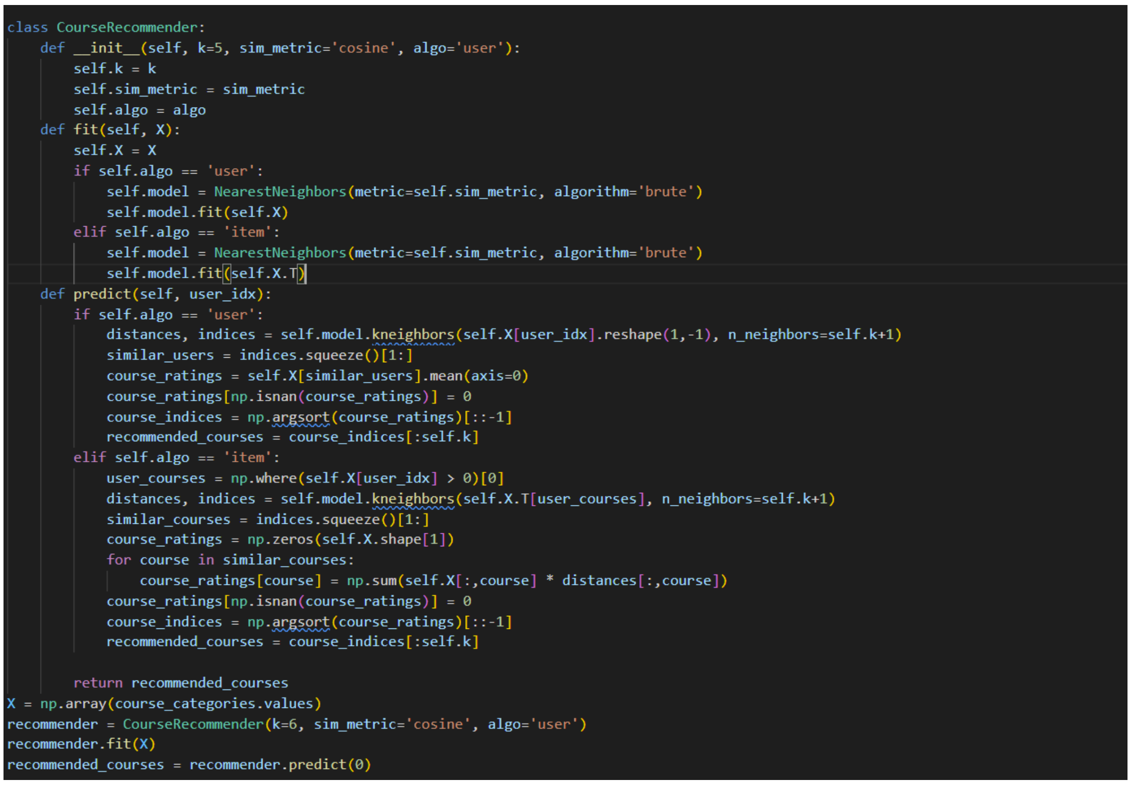
When the user clicks on sort/filter button a drop menu will appear including the categories and the by clicking in any of the categories courses will be shown based on the ranking to select the most preferred course.



```python
user_item_matrix.fillna(0, inplace=True)


cosine_sim_matrix = cosine_similarity(user_item_matrix)
cosine_sim_df = pd.DataFrame(
    cosine_sim_matrix, index=user_item_matrix.index, columns=user_item_matrix.index)
model_knn = NearestNeighbors(metric='cosine', algorithm='brute')
model_knn.fit(user_item_matrix)


def recommend_courses(user_id, num_recommendations, k):

    distances, indices = model_knn.kneighbors(
        user_item_matrix.loc[user_id].values.reshape(1, -1), n_neighbors=k+1)
    knn_indices = indices.flatten()[1:]
    knn_ratings = user_item_matrix.loc[knn_indices]
    knn_ratings = knn_ratings[knn_ratings > 3].dropna(how='all')
    course_avg_ratings = knn_ratings.mean(axis=0)
    top_courses = course_avg_ratings.sort_values(
        ascending=False)[:num_recommendations]
    recommended_course_ids = top_courses.index.tolist()
    recommended_courses = courses_df[courses_df['course_id'].isin(
        recommended_course_ids)]
    return recommended_courses


recommended_courses = recommend_courses(data['id'], 8, 2)

outputData = json.loads(recommended_courses.course_id.to_json())
values = list(outputData.values())
print(values)
```

Figure 4.3 - Code For Hybrid Recommendation System

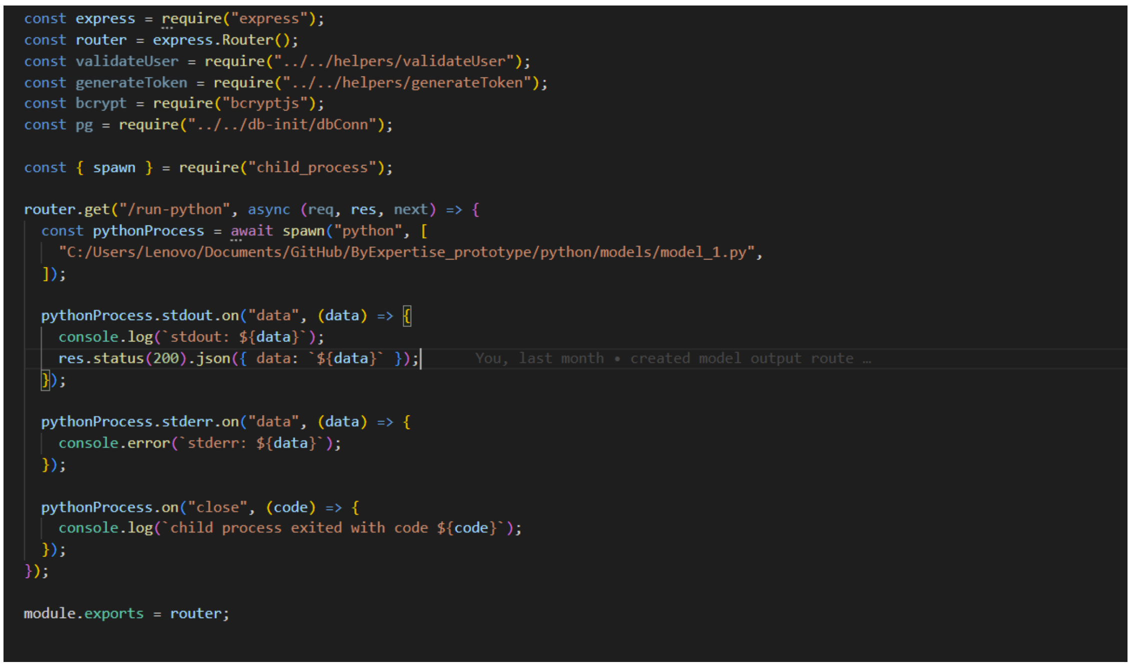
This is a ML model that connects to a PostgreSQL database, retrieves user ratings and course data, and uses the data to create a recommendation system based on the K-Nearest Neighbors algorithm. The system recommends courses to a given user based on their similarity to other users who have rated courses highly. The script returns a list of recommended course IDs in JSON format.



```python
class CourseRecommender:
    def __init__(self, k=5, sim_metric='cosine', algo='user'):
        self.k = k
        self.sim_metric = sim_metric
        self.algo = algo
    def fit(self, X):
        self.X = X
        if self.algo == 'user':
            self.model = NearestNeighbors(metric=self.sim_metric, algorithm='brute')
            self.model.fit(self.X)
        elif self.algo == 'item':
            self.model = NearestNeighbors(metric=self.sim_metric, algorithm='brute')
            self.model.fit(self.X.T)
    def predict(self, user_idx):
        if self.algo == 'user':
            distances, indices = self.model.kneighbors(self.X[user_idx].reshape(1,-1), n_neighbors=self.k+1)
            similar_users = indices.squeeze()[1:]
            course_ratings = self.X[similar_users].mean(axis=0)
            course_ratings[np.isnan(course_ratings)] = 0
            course_indices = np.argsort(course_ratings)[::-1]
            recommended_courses = course_indices[:self.k]
        elif self.algo == 'item':
            user_courses = np.where(self.X[user_idx] > 0)[0]
            distances, indices = self.model.kneighbors(self.X.T[user_courses], n_neighbors=self.k+1)
            similar_courses = indices.squeeze()[1:]
            course_ratings = np.zeros(self.X.shape[1])
            for course in similar_courses:
                course_ratings[course] = np.sum(self.X[:,course] * distances[:,course])
            course_ratings[np.isnan(course_ratings)] = 0
            course_indices = np.argsort(course_ratings)[::-1]
            recommended_courses = course_indices[:self.k]

        return recommended_courses
X = np.array(course_categories.values)
recommender = CourseRecommender(k=6, sim_metric='cosine', algo='user')
recommender.fit(X)
recommended_courses = recommender.predict(0)
```

Figure 4.4 - Code For Collaborative Recommendation System

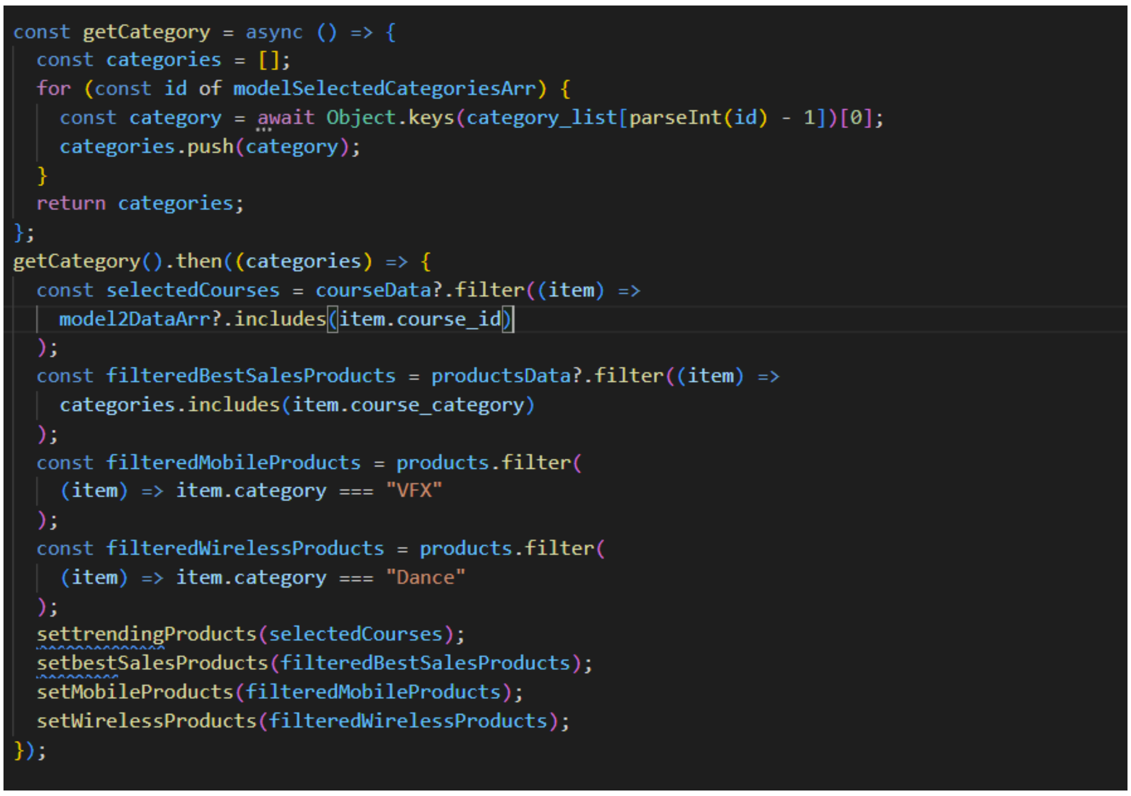
This ML model code retrieves data from a PostgreSQL database and uses it to generate course recommendations using collaborative filtering. It creates a Course Recommender class with parameters for the number of neighbors to consider, similarity metric, and algorithm type, and then generates recommendations for a specific user based on their course ratings.



```
const express = require("express");
const router = express.Router();
const validateUser = require("../../helpers/validateUser");
const generateToken = require("../../helpers/generateToken");
const bcrypt = require("bcryptjs");
const pg = require("../../db-init/dbConn");

const { spawn } = require("child_process");

router.get("/run-python", async (req, res, next) => {
  const pythonProcess = await spawn("python", [
    "C:/Users/Lenovo/Documents/GitHub/ByExpertise_prototype/python/models/model_1.py",
  ]);

  pythonProcess.stdout.on("data", (data) => {
    console.log(`stdout: ${data}`);
    res.status(200).json({ data: `${data}` });      You, last month • created model output route …
  });

  pythonProcess.stderr.on("data", (data) => {
    console.error(`stderr: ${data}`);
  });

  pythonProcess.on("close", (code) => {
    console.log(`child process exited with code ${code}`);
  });
});

module.exports = router;
```

Figure 4.5 - Code for API to pass data through the ML Model

The code defines a router and a GET endpoint named "run-python". When the endpoint is called, the code creates a child process to run a Python script using the child process module. The stdout and stderr of the child process are captured and logged to the console. The script also imports and uses other modules such as express-validator for validation and jwt for generating tokens. Overall, the code seems to be an example of how to use child processes in Node.js to run external scripts.



```
const getCategory = async () => {
  const categories = [];
  for (const id of modelSelectedCategoriesArr) {
    const category = await Object.keys(category_list[parseInt(id) - 1])[0];
    categories.push(category);
  }
  return categories;
};
getCategory().then((categories) => {
  const selectedCourses = courseData?.filter((item) =>
    model2DataArr?.includes(item.course_id)
  );
  const filteredBestSalesProducts = productsData?.filter((item) =>
    categories.includes(item.course_category)
  );
  const filteredMobileProducts = products.filter(
    (item) => item.category === "VFX"
  );
  const filteredWirelessProducts = products.filter(
    (item) => item.category === "Dance"
  );
  settrendingProducts(selectedCourses);
  setbestSalesProducts(filteredBestSalesProducts);
  setMobileProducts(filteredMobileProducts);
  setWirelessProducts(filteredWirelessProducts);
});
```

Figure 4.6 - Code for Displaying the ML Model output to UI

This code defines a function to get course categories, filters and sets various product lists based on those categories and selected courses. The filtered product lists are then set as state variables.

4.2 Technology Stack

1. Python (Machine Learning)
    - To create the Hybrid Recommendation model.
    - To create the Collaborative Recommendation model.
2. React JS
    - To create a user friendly and interactive user interface.
3. Node JS
    - To create APIs for interaction between the ML models and the user interface.
4. Express
    - Used to establish a connection between the APIs and the user interface.
5. JWT
    - Used to generate JWT token to keep the user data secure.
6. PostgreSQL
    - To store all data user data, course data, login credentials, etc.
7. Postman
    - To test working of APIs.
8. VS Code
    - Used to write the codes and scripts required for developing the user interface and the recommendation models.
    - Codes written in Python and ReactJS.

Chapter 5

Testing

Testing is an organized summary of testing objectives, activities, and results. It is created and used to help stakeholders (product manager, analysts, testing team, and developers) understand product quality and decide whether a product, feature, or a defect resolution is on track for release. Test documentation includes all files that contain information on the testing team's strategy, progress, metrics, and achieved results. The combination of all available data serves to measure the testing effort, control test coverage, and track future project requirements.

5.1 Functional Testing

There are multiple ways to ensure software meets the requirements as specified. When software developers make changes to the source code, they must run tests to ensure proper functionality. One of the best ways to do this is through functional testing. Functional testing is a type of software testing that verifies the accuracy and correctness of a system's functional requirements or use cases. It validates whether the system functions as expected based on user input, configuration settings, and system data. Functional tests are black-box testing strategies, meaning they are typically carried out by testers who have no knowledge of the codebase but are familiar with how it should work from a user's perspective. This type of testing is often used to validate the functionality of an application before releasing it to the production team.

5.1.1 Unit Testing

Unit testing is the first level of testing, which is typically performed by the developers themselves. Because flaws will be detected earlier in the testing process and will take less time to fix than if they were discovered later, debugging will be easier as a result of unit testing. It helped us understand the desired output of each module, which we had broken down into separate units, and classifying the cry categories on the basis of algorithm that we have used.

5.1.2 Various Testcases

Test Case Name	Test Condition	Test Steps	Test Data	Expected Results	Actual Results
Successful execution of Python script	The Python script exists in the specified file path and executes without errors	Send a GET request to the /run-python endpoint. Verify that the response status code is 200	N/A	The response status code is 200, and the response body contains the output from the Python script as a string in a "data" field	Pass
Python script execution fails due to invalid file path	The file path specified in the code does not exist or is incorrect	Send a GET request to the run-python endpoint. Verify that the response status code is 201	N/A	The response status code is not 200 (e.g. 404 or 500), and the response body may contain an error message or no data at all	Pass
Python script execution fails due to errors in the script	The Python script has syntax errors or other issues that prevent it from executing successfully	Modify the Python script to intentionally introduce syntax errors or other issues. Send a POST request to the /user-data-model endpoint with valid input data in the request body 3. Verify that the response status code is 200 or 500, depending on the error handling in the code	{"user_id": 100}	The response status code is 200 or 500, depending on the error handling in the code, and the response body may contain an error message or no data at all	Pass/Fail
Python script execution with empty input	The Python script executes successfully with empty input data	Send a POST request to the /user-data-model endpoint with an empty request body. Verify that the response status code is 200 and the response body contains the output from the Python script as a string in a "data" field	{}	The response status code is 200, and the response body contains the output from the Python script as a string in a "data" field	Pass
Python script execution with invalid input	The Python script executes successfully with invalid input data	Send a POST request to the /user-data-model endpoint with invalid input data in the request body. Verify that the response status code is 200 and the response body contains the output from the Python script as a string in a "data" field	{"user_id": 100}	The response status code is 200, and the response body contains the output from the Python script as a string in a "data" field	Pass

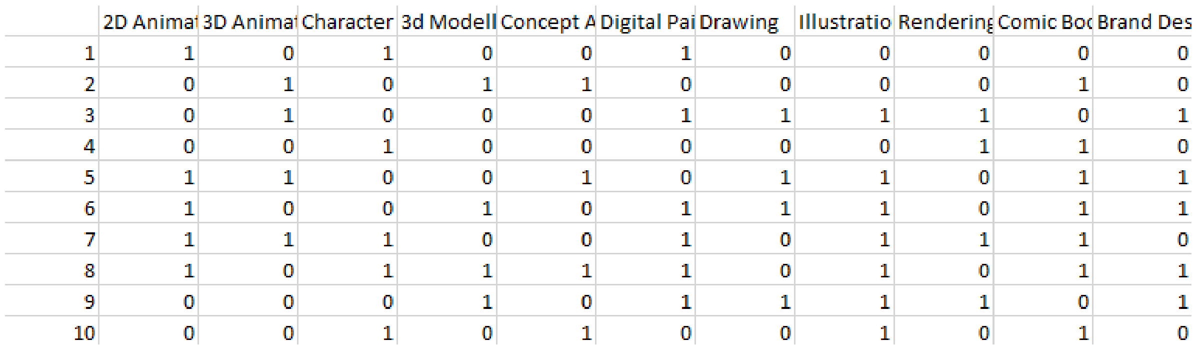
Table 5.1.2 - Testcases for Testing

Chapter 6

Result

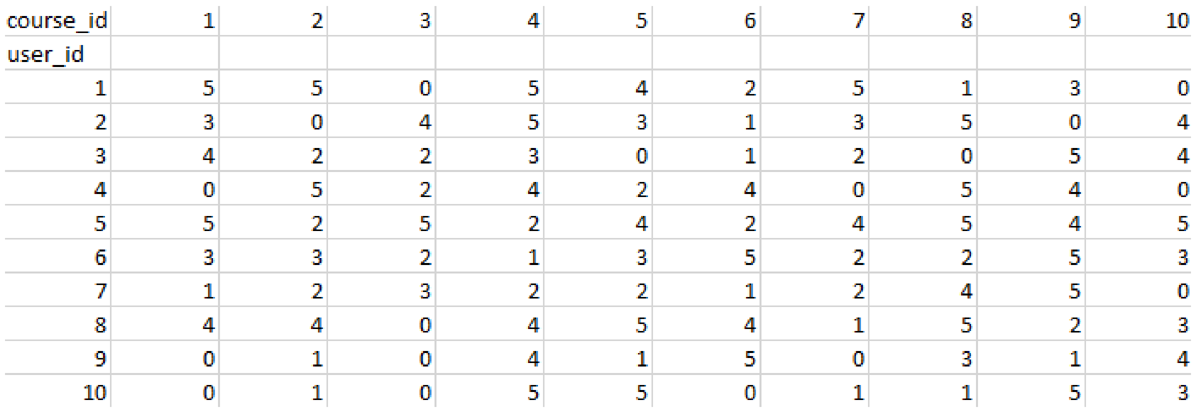
6.1 Datasets

The data collected through the Google form has been divided into three datasets courses, user_category_selection, and rating. Each dataset provides valuable information that helps to understand user behavior and preferences on the platform. The courses dataset contains 200 entries, providing detailed information on the courses available on the platform. This dataset provides details about the courses which includes course category, price, course start and end date, and so on. The user_category_selection dataset contains data on what categories the users have selected, and it has 0 and 1 values. This dataset helps to identify which categories are more popular among users and can be used to improve the user experience by providing more courses on popular categories. The rating dataset contains ratings from 1 to 5, and it is updated as users get added to the platform. This dataset provides essential information on how users perceive the courses and helps in identifying areas of improvement for the platform. The results have helped us to make informed decisions about the courses offered on the platform, and to enhance the user experience by offering courses in popular categories and improving course quality based on user feedback.



	2D Anima	3D Anima	Character	3d Modell	Concept A	Digital Pai	Drawing	Illustratio	Rendering	Comic Boo	Brand Des
1	1	0	1	0	0	1	0	0	0	0	0
2	0	1	0	1	1	0	0	0	0	1	0
3	0	1	0	0	0	1	1	1	1	0	1
4	0	0	1	0	0	0	0	0	1	1	0
5	1	1	0	0	1	0	1	1	0	1	1
6	1	0	0	1	0	1	1	1	0	1	1
7	1	1	1	0	0	1	0	1	1	1	0
8	1	0	1	1	1	1	0	1	0	1	1
9	0	0	0	1	0	1	1	1	1	0	1
10	0	0	1	0	1	0	0	1	0	1	0

Figure 6.1 - Category Selection Data



course_id	1	2	3	4	5	6	7	8	9	10
user_id										
1	5	5	0	5	4	2	5	1	3	0
2	3	0	4	5	3	1	3	5	0	4
3	4	2	2	3	0	1	2	0	5	4
4	0	5	2	4	2	4	0	5	4	0
5	5	2	5	2	4	2	4	5	4	5
6	3	3	2	1	3	5	2	2	5	3
7	1	2	3	2	2	1	2	4	5	0
8	4	4	0	4	5	4	1	5	2	3
9	0	1	0	4	1	5	0	3	1	4
10	0	1	0	5	5	0	1	1	5	3

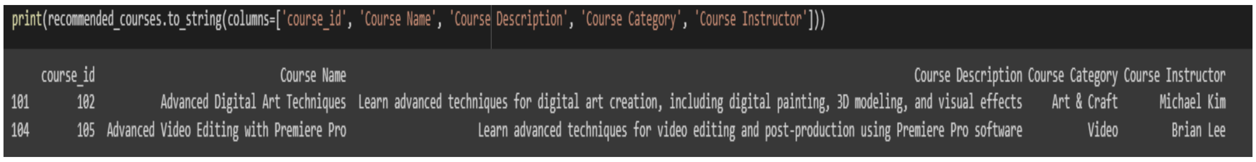
Figure 6.2 - User Rating Data

6.2 Model Results

The designed course recommendation system has shown promising results in improving the accuracy of course recommendations based on user preferences. The system includes two models: a Hybrid Recommendation model and a collaborative filtering model. In the Hybrid Recommendation Model, each user is represented as a vector of their past course enrolments or ratings, and the cosine similarity score is calculated between the user's vector and the vectors of other users who have taken comparable courses. The K users who are most similar to the present user are identified using KNN, and courses that those users have taken or highly rated are recommended to the user. The Collaborative-based filtering model uses a vector representation of each course's features and calculates the cosine similarity between each course's vector and the user's previous course preference vectors. Based on the K courses with the highest cosine similarity, the model recommends courses to the user. Additionally, the designed system includes a category selection pop-up that triggers the second model to determine the user's initial interests based on the course categories. This approach has been shown to be effective in providing course recommendations to users who do not have a history of course enrolments or ratings.

Overall, the course recommendation system has the potential to enhance the user experience on the platform by providing more accurate course recommendations based on user preferences. The datasets and models used in this study provide valuable insights into user behavior and preferences and can be used to improve the platform further.

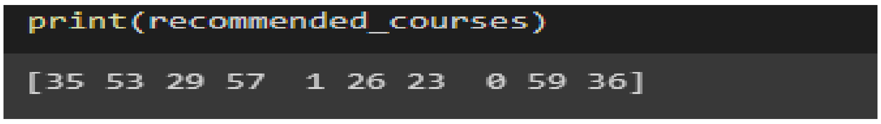
The below images are outputs of the hybrid recommendation ML model and the collaborative recommendation ML model, respectively. The hybrid model gives a table with the recommended Course IDs and some details of that course based on the user's category selection. While the collaborative model provides us with an array of recommended Course IDs based on the user's course ratings. These Course IDs are then mapped to the course details stored in the database and the list of recommended courses are displayed to the user on the user interface.



```
print(recommended_courses.to_string(columns=['course_id', 'Course Name', 'Course Description', 'Course Category', 'Course Instructor']))
```

```
     course_id                             Course Name                                                                          Course Description Course Category Course Instructor
101        102            Advanced Digital Art Techniques  Learn advanced techniques for digital art creation, including digital painting, 3D modeling, and visual effects    Art & Craft       Michael Kim
104        105  Advanced Video Editing with Premiere Pro                    Learn advanced techniques for video editing and post-production using Premiere Pro software          Video          Brian Lee
```

Figure 6.3 - Hybrid Recommendation Model Output



```
print(recommended_courses)
```

```
[35 53 29 57  1 26 23  0 59 36]
```

Figure 6.4 - Collaborative Recommendation Model Output

Chapter 7

Conclusion and Future Scope

7.1 Conclusion

In conclusion, a hybrid approach to course recommendation that uses the k-nearest neighbors (KNN) algorithm can overcome the drawbacks of both Collaborative filtering as well as Content-based filtering techniques. In order to deliver more precise and varied course recommendations that are catered to each user's tastes and needs, this strategy can take advantage of both approaches. Through our review of previous research, we found that many studies have examined the use of hybrid course recommendation systems for personalised learning. These systems have been shown to improve the accuracy of course recommendations and the user experience. Furthermore, the KNN algorithm has been demonstrated to be productive in a wide range of domains, making it ideal for hybrid course recommendation systems. Our developed system, is a hybrid course recommendation system based on the KNN algorithm which is easily customizable and adaptable to various data sources and features. This flexibility makes it an ideal solution for academic institutions, online course providers, and other organizations that want to offer personalized and relevant course recommendations to their users. Overall, a hybrid course recommendation system employed using KNN algorithm can make a significant contribution to the field of personalized learning by improving the quality and effectiveness of online courses.

7.2 Future Scope

In the future, our designed system can be developed in a way where the system could provide a feedback mechanism for students in real-time and based on that feedback the recommendations will be provided. This could help the system become adaptable and enhance the quality of recommendations.

Bibliography

[1] Y. Adilaksa and A. Musdholifah, ”Recommendation System for Elective Courses using Content-based Filtering and Weighted Cosine Similarity,” 2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI), Yogyakarta, Indonesia, 2021, pp. 51-55, doi: 10.1109/ISRITI54043.2021.9702788.

[2] S. G. G, G. M. Bharath, S. R and M. Indumathy, ”Course Recommendation System in Social Learning Network (SLN) Using Hybrid Filtering,” 2021 5th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2021, pp. 1078-1083, doi: 10.1109/ICECA52323.2021.9675992.

[3] Rani, L. P. J., Wise, D. C. J. W., Ajayram, K., Gokul, T., Kirubakaran, B. (2020). Course Recommendation for students using Machine Learning. 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC). doi:10.1109/icesc48915.2020.9156012.

[4] Chen, Zheng; Liu, Xueyue; Shang, Li (2020). [IEEE 2020 International Conference on Big Data and Informatization Education (ICBDIE) - Zhangjiajie, China (2020.4.23-2020.4.25)] 2020 International Conference on Big Data and Informatization Education (ICBDIE) - Improved course recommendation algorithm based on collaborative filtering. , (), 466–469. doi:10.1109/ICBDIE50010.2020.00115

[5] Ajaegbu, C. (2021). An optimized item-based collaborative filtering algorithm. Journal of Ambient Intelligence and Humanized Computing. doi:10.1007/s12652-020-02876-1

[6] Y. Ren, Z. He and T. Han, ”Research on Optimal Design of Online Education Course Recommendation System Based on Hybrid Recommendation Algorithm,” 2021 2nd International Conference on Big Data and Informatization Education (ICBDIE), 2021, pp. 461-464, doi: 10.1109/ICBDIE52740.2021.00111.

[7] Alhijawi Bushra,Al-Naymat Ghazi,Obeid Nadim,Awajan Arafat. Novel predictive model to improve the accuracy of collaborative filtering recommender systems[J]. Information Systems,2021,96.

[8] Zhang Fan, Liu Guoguo. Research on University Ideological and political course recommendation system based on improved collaborative filtering algorithm [J]. Microcomputer application, 2020,36 (09): 1-4

[9] Obeidat, R., Duwairi, R., Al-Aiad, A. (2019). A Collaborative Recommendation System for Online Courses Recommendations. 2019 International Conference on Deep Learning and Machine Learning in Emerging Applications (Deep-ML). doi:10.1109/deep-ml.2019.00018

[10] Garg, V., Tiwari, R. (2016). Hybrid massive open online course (MOOC) recommendation system using machine learning. International Conference on Recent Trends in Engineering, Science Technology - (ICRTEST 2016). doi:10.1049/cp.2016.1479

[11] Geetha et al 2018 J. Phys.: Conf. Ser. 1000 012101 A Hybrid Approach using Collaborative filtering and Content based Filtering for Recommender System.

[12] K. Wakil, B. Akram, N. Kamal and A. Safi, "Web Recommender System for Private Universities' Admission in Iraq:UHD Case Study," International Journal of e-Education, e-Business, e-Management and eLearning, pp. 329-340, 2014

[13] A. S. Tewari, A. Kumar and A. G. Barman, "Book recommendation system based on combine features of content based filtering, collaborative filtering and association rule mining," 2014 IEEE International Advance Computing Conference (IACC), Gurgaon, India, 2014, pp. 500-503, doi: 10.1109/IAdCC.2014.6779375.

[14] H. Simanjuntak, D. Tarigan, I. Sibarani, C. J. Hutapea, R. Lumbantoruan and M. Sigiro, "Weighted Hybrid Recommendation System for Toba Tourism Based on Google Review Data," 2022 IEEE International Conference of Computer Science and Information Technology (ICOSNIKOM), Laguboti, North Sumatra, Indonesia, 2022, pp. 1-8, doi: 10.1109/ICOSNIKOM56551.2022.10034911.

[15] N. Song, Q. Lu and Z. Zhang, "Collaborative Filtering Algorithm Based on Rating Prediction and User Characteristics," 2019 IEEE Intl Conf on Parallel Distributed Processing with Applications, Big Data Cloud Computing, Sustainable Computing Communications, Social Computing Networking (ISPA/BDCloud/SocialCom/SustainCom), Xiamen, China, 2019, pp. 673-678, doi: 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00102.

Appendices

Appendix-I: Python Libraries

1. Download Python from https://www.python.org/downloads/

2. Now download the required libraries using the following commands:

   - pip install numpy
   - pip install pandas
   - pip install json
   - pip install flask
   - pip install sklearn

Appendix-II: Node Libraries

1. Download Node.js from https://nodejs.org/en/download
2. Now download the required libraries using the following commands:
   - npm install axios
   - npm install bcrypt
   - npm install bcryptjs
   - npm install body-parser
   - npm install config
   - npm install cors
   - npm install express
   - npm install joi
   - npm install jsonwebtoken
   - npm install morgan
   - npm install multer
   - npm install pg-promise
   - npm install jest
   - npm install moxios
   - npm install nodemon
   - npm install supertest
   - npm install firebase
   - npm install framer-motion
   - npm install mdb-react-ui-kit
   - npm install react
   - npm install react-confirm-alert
   - npm install react-dom
   - npm install react-redux
   - npm install react-router-dom

- npm install react-scripts
- npm install react-spinners
- npm install react-toastify
- npm install reactive-button
- npm install reactjs-popup
- npm install reactstrap
- npm install remixicon
- npm install web-vitals

Publication

Paper entitled **“Developing Recommendation System for creative Learning Web Framework”** is **“ACCEPTED”** at **“IC3SEA 2023: International Conference on Contemporary Challenges in Science and its Engineering Applications”** by **“Ruchita Raut, Kushal Todi and Pratik Dhumal ”.**