

# ONLINE COURSE PERSONALIZED RECOMMENDATION SYSTEM

61431-FCS BSc (HONS) COMPUTING

19086235D  
Apr 7, 2023

Project Supervisor: Prof. LI Qing

## Abstract

The selection of courses during undergraduate study is a critical aspect in obtaining relevant knowledge related to future professions, such as software developer, website application developer, while also balancing the course workload to maintain a satisfactory Grade Point Average (GPA). However, the process of gaining an understanding of a course can be difficult, hence the introduction of a personalized recommendation system. The objective is to develop a Hybrid Recommendation System (RS) that suggests courses based on user preferences, which addresses the issue of information overload (IO), and reducing the steps required to understand the course workloads, learning styles, etc.

## Acknowledgement

I would like to express my gratitude towards Prof. LI Qing, for supervising in which sections needed working and help me understand on what to write for this Capstone Project. Alongside that, he gave me pointers on how to gather relevant data for this Capstone Project.

I would like to extend my gratitude to those who have provided me with relevant data. Their assistance has been helpful in completing this Capstone Project.

I would like to thank friends and family who have supported me during the Capstone Project in giving relevant advice to me.

I would like to thank Dr. FUNG Walter for giving out seminars on general guidelines regarding what needs to be done for students undertaking Capstone Project as the culmination of their studies.

## Table of Contents

<b>Abstract .....</b>	<b>2</b>
<b>Acknowledgement .....</b>	<b>3</b>
<b>Table of Contents .....</b>	<b>4</b>
<b>List of Figures .....</b>	<b>6</b>
<b>List of Tables .....</b>	<b>6</b>
<b>Abbreviations.....</b>	<b>7</b>
<b>Chapter 1. Introduction .....</b>	<b>8</b>
1.1. Background and Motivation.....	8
1.2. Benefits of using Recommender System.....	9
1.3. Project Objectives .....	10
<b>Chapter 2. Literature Review .....</b>	<b>11</b>
2.1. Content-Based Filtering (CBF) .....	11
2.2. Collaborative Filtering (CF).....	12
2.3. Hybrid Recommender System .....	13
2.4. Why Hybrid Recommender System .....	14
<b>Chapter 3. Methodology.....</b>	<b>15</b>
3.1. Data Collection .....	15
3.1.1. Content-Based Filtering Data Collection .....	15
3.1.2. Collaborative Filtering Data Collection.....	15
3.2. Database Design.....	16
3.3. Recommender System .....	16
3.3.1. Content-Based Filtering .....	16
3.3.2. Collaborative Filtering .....	17
3.4. Front-end Development.....	17
3.4.1. Website Pages .....	17
3.4.2. Login/Register Feature .....	18
3.4.3. Localization .....	18
3.4.4. Why Website Application .....	19

<b>3.5. Back-end Development.....</b>	<b>19</b>
3.5.1. Database Integration .....	19
3.5.2. Recommender System Integration.....	19
<b>Chapter 4. Implementation.....</b>	<b>20</b>
<b>4.1. Data Collection.....</b>	<b>20</b>
4.1.1. Content-Based Filtering .....	20
4.1.2. Collaborative Filtering .....	21
<b>4.2. Database.....</b>	<b>23</b>
<b>4.3. Recommender System .....</b>	<b>23</b>
4.3.1. Content-Based .....	23
4.3.2. Collaborative Recommender .....	23
<b>4.4. Website.....</b>	<b>24</b>
<b>4.5. API Server .....</b>	<b>27</b>
<b>4.6. Deployment .....</b>	<b>28</b>
<b>4.7. Project Schedule.....</b>	<b>28</b>
<b>Chapter 5. Preliminary Evaluation .....</b>	<b>31</b>
5.1. Experimental / Survey Results .....	31
5.2. Small Data Size .....	31
5.3. Collaborative Recommendation Model Storage Alternative .....	32
5.4. Localization .....	33
<b>Chapter 6. Conclusion .....</b>	<b>34</b>
<b>References .....</b>	<b>35</b>

## List of Figures

Figure 1. CBF recommender system example .....	11
Figure 2. CF recommender system example.....	12
Figure 3. ER Diagram of the database .....	16
Figure 4. PolyU COMP Subject Offerings .....	20
Figure 5. Survey website application .....	21
Figure 6. Rating modal.....	22
Figure 7. Table schema differences .....	22
Figure 8. Screenshot of the search bar component. ....	25
Figure 9. Screenshot of searching courses. ....	25
Figure 10. Screenshot of page "course/COMP3211".....	26
Figure 11. Screenshot of profile page. ....	27
Figure 12. Demographic of people filling in the survey .....	32

## List of Tables

Table 1. Front-end website pages. ....	17
Table 2. Project schedule. ....	28

## Abbreviations

<b>API</b>	Application Programming Interface
<b>UI</b>	User Interface
<b>UX</b>	User Experience
<b>CBF</b>	Content-Based Filtering
<b>CF</b>	Collaborative Filtering
<b>RS</b>	Recommender System
<b>IO</b>	Information Overload
<b>TF-IDF</b>	Term Frequency – Inverse Document Frequency
<b>NLP</b>	Natural Language Processing
<b>MAE</b>	Mean Absolute Error
<b>VM</b>	Virtual Machine
<b>ORM</b>	Object Relational Mapper

## Chapter 1. Introduction

The following chapter introduces the background, motivation as well as objectives of the project, followed by the outline of the remaining part of the report.

### 1.1. Background and Motivation

Registering elective courses for undergraduate students is one of the most important processes in their study. Each student has different aspirations (game developer, software engineer, fintech specialist, data scientist, etc.). To achieve their aspirations, they must take relevant courses to gain required knowledge to excel in their respective fields. Neglecting this process may cause students to spend their study credits on irrelevant courses and may affect Grade Point Average (GPA) negatively due to low performance caused by lacking course satisfaction. Hence students need to plan this process carefully and they need to consider the aspects of each course (is the course relevant to students' study plan, difficulty, workload, teacher's teaching style, is the material up to date, etc.). Students who care about their future career prospects and maintaining higher grades have several steps that can be taken to help them conclude which course to register for. However, there are some pitfalls within these options, so the idea of each university having their own Recommendation System (RS) would be undeniably beneficial to both students and universities by recommending courses relevant to student's study plan (recommend game development-related classes and its pre-requisites for aspiring game developers) and university sees students with higher grades.

The current situation and problems of known steps to the students' traditional decision-making process will be introduced as follows. Then the benefits of having a RS in higher-education institutes will be described. Continued with descriptions of several known techniques used in RS. Followed by the reason why Hybrid RS is picked for this project. After that, methodologies that will be taken for this project is shown and continued with the current progress of the project and wrapped up with the conclusion chapter.

When deciding on courses to register, Students may take several steps in considering whether an elective course is beneficial or not. They may ask their peers or academic advisors for advice on which courses to take based on their aspirations. However, there are several drawbacks to this approach as every student has different academic prowess and interests which could lead to unsatisfactory course experience based on the wrong information. Additionally, students who don't



have acquaintances in the study field they are taking, especially international freshmen would have a hard time getting advice from their peers alongside the fact that the pandemic has made it harder for students to connect with their peers.

Students are advised by academic advisors to read the syllabus of each course to have a general idea of the course's workload, etc. Despite that, this solution doesn't provide insight on each lecturer's teaching style. Chetty et al [1] claimed that styles of teaching affects student's learning motivation which contributes to students' academic performance. With that in mind, conflicting learning and teaching style may result in lower GPA. Furthermore, Schmitt et al [2] argued that students lacking the ability to seek for information are prone to experience Information Overload (IO), where the students' mental capacity can't process large amount of information when concluding whether a course is worth taking, hindering their decision-making process. This phenomenon happens when students gather data from several sources (peer advice, syllabus reading, etc.). This phenomenon happens when students synthesize information from several sources, reflect on each impact generated by different courses, and to conclude the most beneficial decision [3].

Some universities implemented a "trial class" system, where students can take a course during the beginning of each semester so students can have a first-hand experience of teacher's teaching style, course contents, etc. and drop the course if students' expectations are not met. However, this approach is impractical due to limitations in actions students can do to adjust their courses, leaving students unable to freely try out different electives.

Lastly, students are encouraged to use a combination of these three approaches to understand each courses' learning and teaching aspects. Using this method offers a significant amount of course insight as each approach is used to cover up the other approaches' weaknesses. Nevertheless, this process is time-consuming and is prone to cause IO to the students, highlighting the need of recommender system.

## 1.2. Benefits of using Recommender System

Personalized recommendation services provided by RS reduces the issue of IO by recommending relevant course in regards to the students' preferences [4]. With that in mind, RS may recommend courses relevant to their chosen interests and students don't have to spend too much time on thinking and get on with their daily tasks. Furthermore, as user rates courses, lecturers teaching courses that have low rating number are encouraged to improve their course (modifying teaching

style, difficulty, workload, material relevance, etc.) so that students can enjoy the course and their university life.

### 1.3. Project Objectives

There are 4 tasks in this project. For task 1, it is imperative to create a website that can gather user course interaction data (explicit data) and store them into a relational database for further processing. Moreover, this website should be able to send request and receive responses from the API server. Task 2 is to create an API server that can receive requests and send responses to the front-end website application. Task 3 is to create a hybrid RS consisting of Content-Based Filtering (CBF) RS and Collaborative Filtering (CF) RS. Lastly, task 4 is to integrate hybrid RS into the API server itself, completing the whole project's proposed functionalities.

## Chapter 2. Literature Review

This chapter presents the conducted literature review for this project. It highlights several types of known personalized recommendation systems such as content-based filtering, collaborative filtering, and hybrid system. Alongside that, strengths and weaknesses associated with these systems are briefly explained. Lastly, an explanation containing why hybrid recommendation system is chosen will be given below.

### 2.1. Content-Based Filtering (CBF)

Wang et al [5] described this type of RS as a system that compares data regarding users searched and rated item. Concluding from the statement before, only the user data is used. Recommendations made by CBF RS relies heavily on the item's features, or courses' features (objectives, intended learning outcomes of the course) in this case to provide similar courses. To illustrate this, a figure explaining how CBF RS works is shown in Figure 1. below.

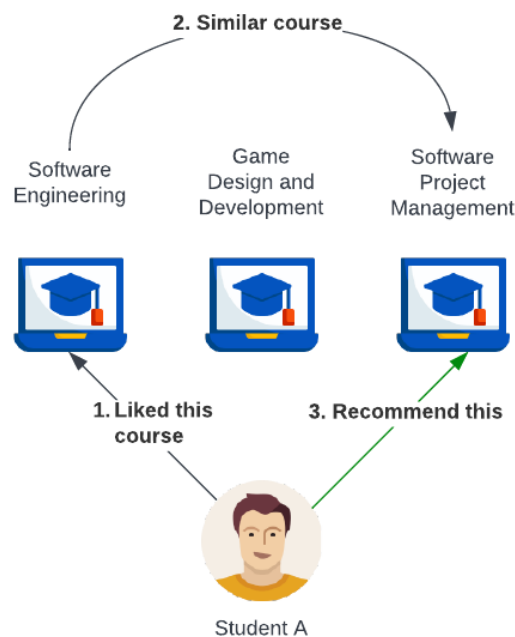


Figure 1. CBF recommender system example

Referring Figure 2.1. above, suppose student A took a Software Engineering course and rated the course positively, they will be recommended to take the Software Project Management course for the next semester due to a degree of similarity in the course's contents. One approach that can be used to determine the similarity of items is by using cosine distance approach. This approach has

several steps needed which are, gathering data that describes the item, cleaning the data from stop words (a list of words that are frequent in any language which can be considered insignificant such as prepositions, pronouns, etc.), process the cleaned data into vectors using Term Frequency – Inverse Document Frequency (TF-IDF) technique which will then be used to find the cosine distance. With this approach, the resulting values will be ranging from 0 to 1, where the higher the value, the likelier an item is correlated to the item in question and vice versa. A strength from this type of RS is that there is no need for other users’ data to start recommending personalized courses to the original user [6], [7]. Moreover, CBF RS provides transparency by explaining how the RS works [7]. On the other hand, this system’s weaknesses are content overspecialization, where users only receives recommendations in regards to defined items in their user profiles and this type of RS’ effectiveness relies heavily on available descriptive metadata [8]. Moreover, the effectiveness of this RS is limited by the person’s knowledge when attributing descriptions to an item. However, this issue is not a case as this project uses course subject description forms provided by PolyU.

## 2.2. Collaborative Filtering (CF)

This type of RS generates recommendations by grouping users together and recommend them with items that have been rated by other users within the same group. Figure 2. below illustrates this type of RS in general.

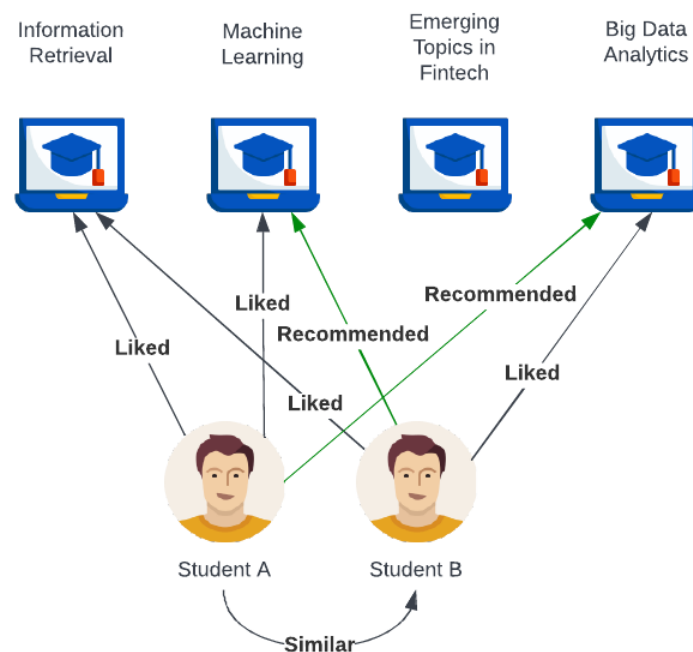


Figure 2. CF recommender system example

Referring to Figure 2.2. above, suppose that subjects liked by student A and student B overlaps, CF RS draws a relation that these students have similar preferences (e.g., both student wants to be a machine learning engineer in the future) and thus, predicts that a course liked by student A will be liked by student B and vice versa. There are two main implementations as to how CF RS comes with these predictions, and they can be categorized into memory-based and model-based collaborative systems which will be explained briefly. For memory-based technique, it can be further divided into user-based and item-based collaborative filtering. User-based and item-based CF relies on grouping either users or items together to generate recommendations. On the other hand, model-based technique uses several algorithms such as matrix factorization (SVD, ALS, PLSA), clustering, etc. [8]. Such system is commonly used to recommend items that is difficult to describe with metadata such as video, music, etc. [8]. It is inferred Collaborative Filtering system combats the issue of Content-Based Filtering System's content overspecialization and is not limited by a person's domain knowledge when attributing any item. Despite that, this approach brings up the cold-start problem, where such systems require large datasets on users interaction with the offered items first before it can generate accurate recommendations [7]. In conclusion, deploying Collaborative RS is impractical for businesses with low number of recurring users because gathering the large dataset requires a long time. Furthermore, using such technique costs a lot of time due to computation growing linearly with number of users and items.

### 2.3. Hybrid Recommender System

Hybrid RS, as it names suggests, this system uses a combination of CBF and CF RS to provide recommendations with better accuracy than RS using only one filtering implementation [6], [7]. This hybrid technique offsets each technique's weaknesses by implementing the other's strengths. For example, it is known that CF method suffers from cold-start problem and in this case, recommendations generated by CBF RS can be used while gathering relevant dataset regarding user's interaction with certain items. When the system has enough data to build a reliable user profile, CF system will be implemented to complement the over-specialization issue brought up by CBF system. Through combining both types of RS, users are given recommendations that are is not restricted to an item and accurate. However, the time needed to develop hybrid systems is a lot due to more than one RS that needs to be built.

## 2.4. Why Hybrid Recommender System

To reiterate, implementing a hybrid RS yields the most accuracy. The system can gather interaction data between a user and a course while recommending courses provided by Content-Based RS using subject description form. When enough data is gathered, recommendations generated by Collaborative RS will be given alongside the former RS. This approach will be used for the project to provide diverse recommendations. For Content-Based RS, TF-IDF and cosine similarity method will be used to generate whether a course is correlated or not to another course. For Collaborative RS, a model-based approach using a dimensionality reduction technique named Alternating Least Squares (ALS) will be used.

## Chapter 3. Methodology

In this section, there are several important parts which spans through Data collection, Database design, hybrid RS development, Front-end/website application development, and Back-end/API server development to implement a website app that can recommend relevant courses. The following section will explain how each parts plays a role in this project.

### 3.1. Data Collection

For this project, there are two types of recommendation that is offered which is Content-Based and Collaborative recommendations. Each RS requires different type of data gathered and the following sub-sections will explain the steps needed to gather the respective data.

#### 3.1.1. Content-Based Filtering Data Collection

For CBF RS to start generating recommendations, PolyU course data must be gathered. Fortunately, these course data are available online as subject description forms. For example, computing subject description form's important features are objectives, intended learning outcomes, subject synopsis/indicative syllabus. However, it is important to note that different department presents descriptive data differently. Looking at Chinese Bilingual Studies (CBS) department's subject description forms, there is only two important features which means these features must be merged into one feature. Moreover, it is possible that some departments do not offer subject description forms online. For now, only computing department subject description form will be used. After gathering enough description data, the features within the data are merged as one which will then be inserted into the database in section 3.2.

#### 3.1.2. Collaborative Filtering Data Collection

As stated above, this type of recommendation will only be given when enough user data is gathered. This kind of recommendations requires gathering several user's existing data such as courses that have been taken and ratings given by the user. As such, a survey will be done to gather final year student's past courses and their ratings towards each taken course. After basic functionalities of the front-end website have been implemented, a course rating feature will be added for data gathering uses.

## 3.2. Database Design

For this section, several essential tables like user, course, and rating are needed for this project to achieve the personalized recommendation feature. However, it is important to note that the database structure might change during this project's development lifecycle. an ER diagram shown in Figure 3. below helps reflect each table's relation to other tables.

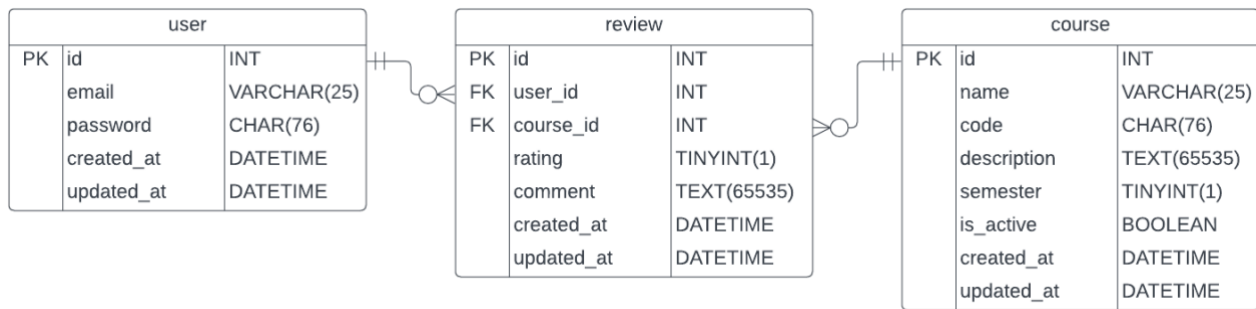


Figure 3. ER Diagram of the database

For usage with Content-Based RS, only the “**course**” table is used, specifically the description column. On the other hand, the “**review**” table containing data about user-course interaction is used for Collaborative RS. As of the writing of this report, the “**comment**” column will not be used, but the gathered on this column could be useful for further improvements or iterations of this recommender system.

## 3.3. Recommender System

This hybrid RS will be built using Python programming language, which allows the leverage of several existing Python libraries such as “**pandas**”, “**scikit-learn**”, “**pyspark**”, and “**numpy**” to generate recommendations from relevant data. The following sub-sections will explain the steps needed to build both RS.

### 3.3.1. Content-Based Filtering

Taking the merged description data from the database, the data needs to be pre-processed before further processing. The data usually needs to be cleaned as there is a possibility of redundant data and to improve data quality [9]. To begin, the extracted text that had been stored before is cleaned from stops words, which are a list of frequently used words that do not carry significant meaning in the context of the text (e.g., “to”, “the”, “as”, etc.) resulting in clean text that is easier to process.



Subsequently, the cleaned text is converted into vectors that can be plotted. The vector values are then used to calculate course similarity using the cosine similarity method.

### 3.3.2. Collaborative Filtering

In this Collaborative RS, library “**PySpark**” is used to implement ALS algorithm onto the user-course interaction data sourced from the web application and survey. As fitting the model takes a long time, it is impractical to fit the model on each request because users will have to wait a long time before getting any recommendations. To address this issue, model training will be scheduled on a timed interval and the resulting model will be saved into the local filesystem of the API server for subsequent use to complete user requests.

## 3.4. Front-end Development

This proposed website will be the primary interface for users interact with the RS, where relevant course data are sent from the back end / API server to be visualized to the website in a manner to prevent IO. The following sections consists of website pages, login/register feature, localization, and why website is chosen.

### 3.4.1. Website Pages

The website will have several pages with its purpose briefly listed in Table 1. below. Some of the pages are only available when the user has logged in to the website.

*Table 1. Front-end website pages.*

Page Name	Purpose
Home	Serves as the first page users go into. Includes a section to introduce the user what this website is about.
Courses	Show a list of courses that the user is interested in. In this page, the user can search for certain courses using a provided search bar from the website.
Course info	Show course’s information from the gathered subject description form and list other courses that is like this course through Content-Based Filtering. Furthermore, user can add the course into a list of courses they had taken and give a rating towards the course for Collaborative recommendations.

Profile	Show courses that have been taken and their ratings by the user, where they can modify the ratings and remove a course from the taken course list if they had made mistakes.
Documents	Contains several capstone project submissions ranging from project outline to final report.

Inferring from Table 3.1. above, the website will be able to query the API server and visualize the recommended course data from the API server's response to help the students' decision-making process. Furthermore, the website allows Create, Read, Update, Delete (CRUD) operations on the back-end server to ensure data sent to the users are up to date.

### 3.4.2. Login/Register Feature

A feature that hasn't been mentioned is the login/register feature as these will be shown in a modal pop-up. It is crucial to have a user account feature for this project because users will be able to rate courses they have taken, and these data will be taken into consideration for collaborative recommendations. To implement login/register feature, more research needs to be done on how it can be implemented due to security reasons. There are several available security methods that can be utilized for this project, but I will choose JSON Web Token (JWT) approach as it is quick and efficient to use for user authentication with the API server. Additionally, if a user who haven't logged into their account tries to see any course information, they will only be provided with recommendations generated by CBF method because the system requires the user to be logged into the website application to provide recommendations provided by CF method.

### 3.4.3. Localization

An additional feature added to this website is to change the language of the website as PolyU's student community is diverse. Currently, there are several languages provided with the help of both DeepL, a neural machine translation service, and my native language knowledge of Indonesian such as English, Chinese (Simplified), Indonesian, Japanese. However, important course information will not be translated (e.g., course name, objectives, intended learning objectives, and subject synopsis/indicative syllabus) to prevent mistranslations of the course description. Additionally, I might plan on asking for the help of other students to help with checking the correctness of translations generated by DeepL.

#### 3.4.4. Why Website Application

Two main factors are considered before choosing a website application approach. A study conducted found that wireless traffic had seen an 73% increase within the span of two years from 2007 to 2009 [10], which signifies the importance of providing a website application that can be accessed by both smartphones and desktops over developing a mobile or desktop application that is not fully accessible by other devices and vice versa. Secondly, I'm familiar with website application development process using NextJS, a front-end JavaScript framework built on top of ReactJS that provides many out-of-the-box features such as hybrid static & server-side rendering, smart bundling, route prefetching, image optimization and thus improving the developer experience without unnecessary configurations. Because of this, development with NextJS will consume less time over using mobile or desktop application development technologies such as React Native or ElectronJS due to unfamiliarity.

### 3.5. Back-end Development

Django, a Python web framework will be used for this project to provide personalized recommendations while providing basic CRUD operations. Django is chosen as the Recommender System is programmed in Python. It is planned to implement the Recommender System inside the Django API server. This API server will be able to receive requests and send response to the front-end website. The following sub-section discusses about database and RS integration.

#### 3.5.1. Database Integration

To fully integrate the database into the API server, there needs several endpoints that does CRUD operations. More research needs to be done on which database to use, whether NoSQL or SQL database is used. However, it is likely that SQL database is used to visualize the relations of the data between each table.

#### 3.5.2. Recommender System Integration

Integrating the hybrid RS into the API server can be done last when basic API functionalities is finished. This is due to both Recommender System, and API server programmed in Python language. However, more research needs to be done on scheduling model training jobs for the Django application.

## Chapter 4. Implementation

This chapter follows the process in building the hybrid recommender system. For ease of understanding, this chapter is categorized similarly as the last chapter such as Data collection, Database, RS, Front-end/website application development, and Back-end/API server implementations followed by the deployment process.

### 4.1. Data Collection

#### 4.1.1. Content-Based Filtering

To collect data for this RS, the subject description forms of the PolyU Computing department were used. A Python web scraper was created to gather data on all the Computing courses that are currently being offered. This web scraper sends a request to the [website](#) shown in Figure 4. below.

### 2022/23 Academic Year

(The lists below are subject to review and changes which the Department can decide to make from time to time.)

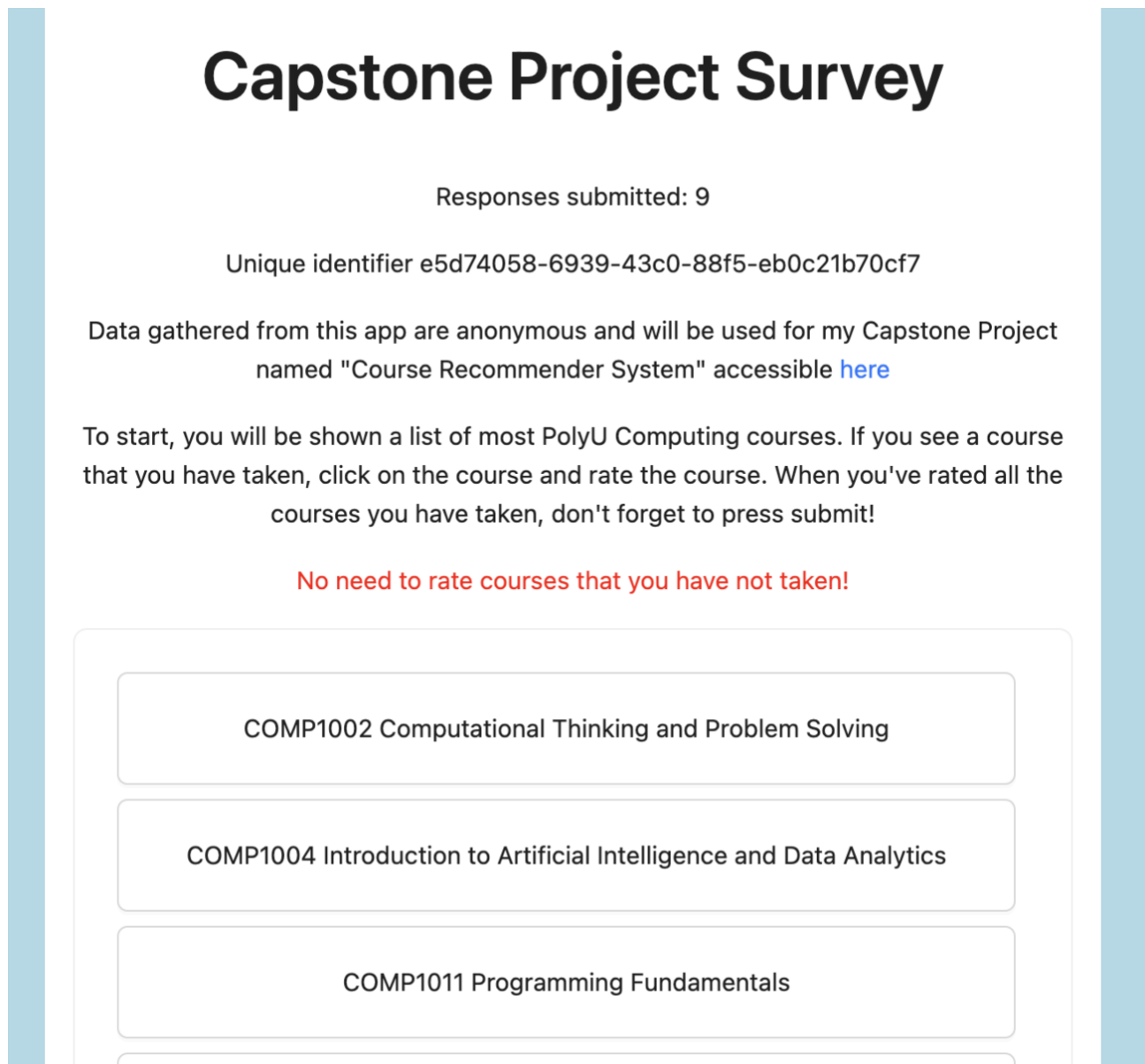
Subject Code	Subject	Semester 1	Semester 2	Semester 3 (Summer Term)
COMP1002	Computational Thinking and Problem Solving	Yes		
COMP1004	Introduction to Artificial Intelligence and Data Analytics	Yes	Yes	
COMP1011	Programming Fundamentals		Yes	
COMP1012	Programming Fundamentals and Applications	Yes		

Figure 4. PolyU COMP Subject Offerings

The resulting HTML file is parsed using the “**BeautifulSoup**” library, which returns an array of offered computing courses. The Python web scraper then makes another request to download all the PDFs provided by PolyU. After that, the “**PyPDF2**” library is used to extract all the text inside the PDFs. After that, the extracted text had to be cleaned using “**nltk**” library as the text had hidden Unicode characters. Once all the text data has been processed, it is stored in JSON and CSV format that is usable for the back-end server development.

#### 4.1.2. Collaborative Filtering

For Collaborative RS, a survey is needs to be conducted to gather the necessary data for further use by the RS that will be built. As the survey requires the user rate all the courses that they have taken and rate them, a custom web application is most suitable to gather the data efficiently. React was used for this [web application](#). Figure 5. below shows the website's user interface.



The screenshot displays a web application titled "Capstone Project Survey". It features a light blue header and footer. The main content area is white and contains the following elements:

- Title:** "Capstone Project Survey" in a large, bold, black font.
- Responses:** "Responses submitted: 9" in a smaller black font.
- Identifier:** "Unique identifier e5d74058-6939-43c0-88f5-eb0c21b70cf7" in a smaller black font.
- Data Notice:** "Data gathered from this app are anonymous and will be used for my Capstone Project named 'Course Recommender System' accessible [here](#)" in a smaller black font.
- Instructions:** "To start, you will be shown a list of most PolyU Computing courses. If you see a course that you have taken, click on the course and rate the course. When you've rated all the courses you have taken, don't forget to press submit!" in a smaller black font.
- Warning:** "No need to rate courses that you have not taken!" in a red font.
- Course List:** A list of three courses, each in a white box with a thin grey border:
  - COMP1002 Computational Thinking and Problem Solving
  - COMP1004 Introduction to Artificial Intelligence and Data Analytics
  - COMP1011 Programming Fundamentals

*Figure 5. Survey website application*

Upon clicking any of the course's name, a modal with a rating component appears so users can rate a course quickly shown in Figure 6. below.

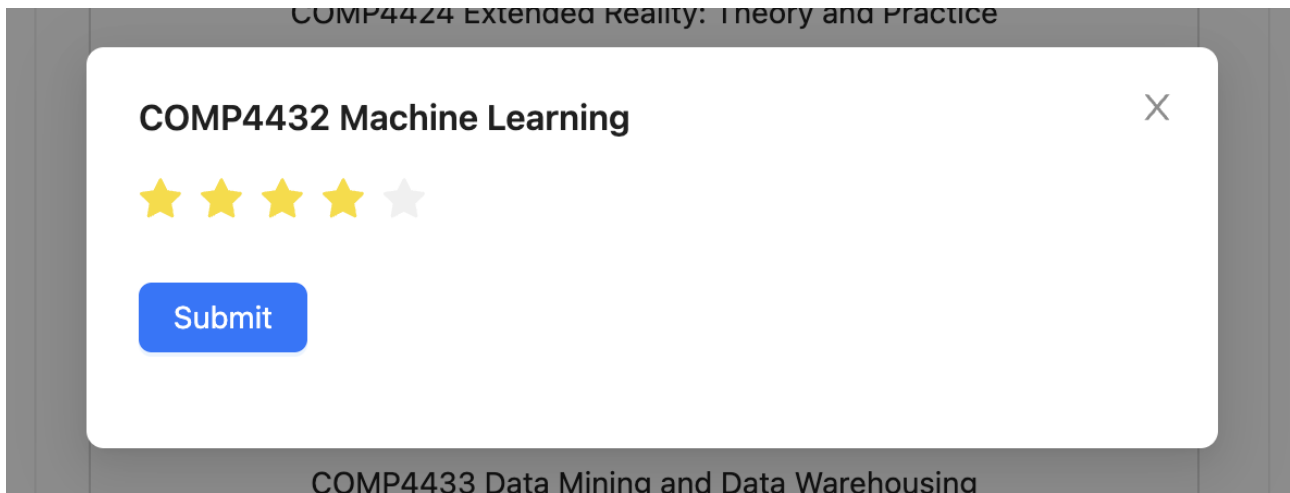


Figure 6. Rating modal

After user rates all the courses that they have taken, they will be prompted with a confirmation before the data is sent into a new table named “**survey**” inside the database. The reasoning behind putting the survey data into a different table than the current table is that the data gathered in this survey application is anonymous, hence the id of the “**survey**” is filled with Universally Unique Identifier (UUID). Below is a comparison of the “**survey**” and “**review**” table schema.

survey		
PK	id	INT
	session	VARCHAR(76)
	code	VARCHAR(76)
	rating	TINYINT(1)
	created_at	DATETIME

review		
PK	id	INT
FK	user_id	INT
FK	course_id	INT
	rating	TINYINT(1)
	comment	TEXT(65535)
	created_at	DATETIME
	updated_at	DATETIME

Figure 7. Table schema differences

It is important to note that since the “**session**” column is of string type, and the “**user\_id**” column is of integer type, the data inside the “**survey**” needs to be processed into integer type. Furthermore, the “**code**” column contains the course code (e.g., COMP3235) while the “**course\_id**” column contains the course id (e.g., 23) regarding the “**course**” table, this means that the mismatching data needs to be converted into the same type and saved in CSV file for ease of use by the Recommender System later in this project.

## 4.2. Database

For the database, supabase is used as it provides relatively easy-to-use API to perform CRUD operations from the survey application and API server. Another factor why supabase is chosen is they provide free tier. Lastly, supabase uses PostgreSQL, a relational SQL database that can show relations of data between each table and allow easier development experience at the cost of being difficult to scale when the data is large. However, scalability will not be an issue for this project as the data required for this project doesn't require the use of NoSQL database yet.

## 4.3. Recommender System

This chapter is split into two subchapters for each recommender system (CF and CBF) in the hybrid recommender system following on the technical processes needed to complete the project.

### 4.3.1. Content-Based

Generating personalized course recommendations based on course descriptions involves several steps, which are described in the following sentences. A TF-IDF matrix is created using **"TfidfVectorizer"** from **"scikit-learn"** with the subject description text sourced from the stored CSV data from before. The resulting TF-IDF matrix is then used to create a cosine similarity matrix with **"cosine\_similarity()"** provided by **"scikit-learn"**. Finally, a function is created that takes the course code as input and returns a list of recommended course codes based on the cosine similarity scores of the inputted course code.

### 4.3.2. Collaborative Recommender

For this recommender system, it is split into saving and loading the model for later use. The first paragraph will explain the saving process and followed by the loading process.

Firstly, course reviews gathered from the web and survey application are concatenated together into a **"pyspark"** data frame. After that, data frame is split into training and test data with the ratio of 8:2. The cross validation is done with 3-fold parameter and the training data is fitted into the validator resulting in a **"pyspark"** model that will be saved into the **"server\_data/model"** folder.

**"ALSModel"** class provided by **"pyspark"** allows loading of saved model within the folder. Once the loading is done, the model is used to generate recommendation for all users resulting in a data frame that needs to be filtered according to a user id to obtain the personalized recommendations. It

is important to note that the resulting recommendations contain courses that the user have taken in the past, so filtering out the data is important to not re-recommend the same course resulting in the final recommendations that can be further processed in the API server.

## 4.4. Website

This section discusses the libraries chosen for this website application, followed by the notable pages within this front-end web application.

The website app is built using Next.js. The framework uses file-based routing which means that JavaScript files inside the “**pages**” folder determine the routing of the app. For UI, “**antd**” was used as they had pre-built production grade UI components that can be utilized to create good looking UI for the web application. Alongside that, “**styled-components**” library was also used to customize some of the premade UI components provided by “**antd**”. For user authentication feature, “**@reduxjs/toolkit**” was used as it features caching responses sent from the API server to improve the UX and reduce redundant requests to the API server. Localization was initially made using the “**next-translate**” library, however due to implications such as translating the course subject description would cause the Content-Based RS to produce different results etc. this feature was left alone.

Out of all the website pages, the notable ones are the “**/courses**” “**/course/[code]**”, and “**/profile**” pages and more will be explained why.

Within the “**/courses**” page, a search bar with autocomplete capability is included so users can search for a specific course shown in the Figure 8. below.



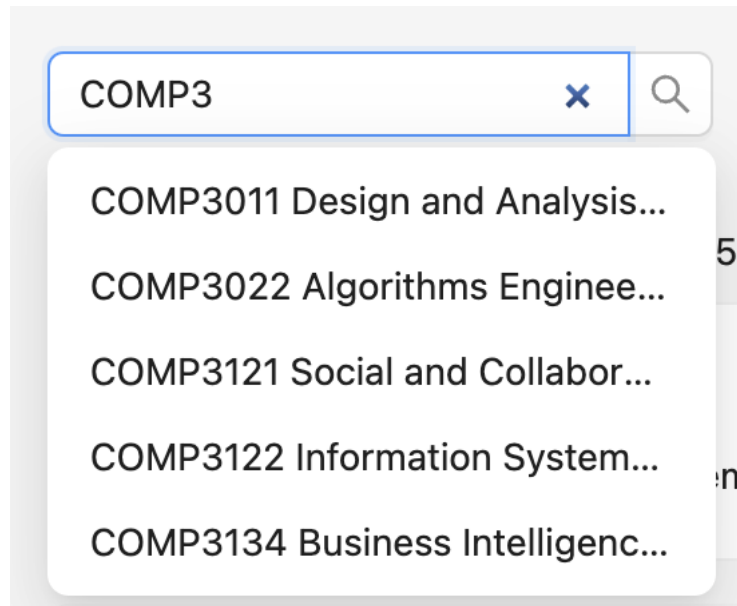


Figure 8. Screenshot of the search bar component.

Users can also search for courses with incomplete course name if they can't remember the course code shown in Figure 9. below.

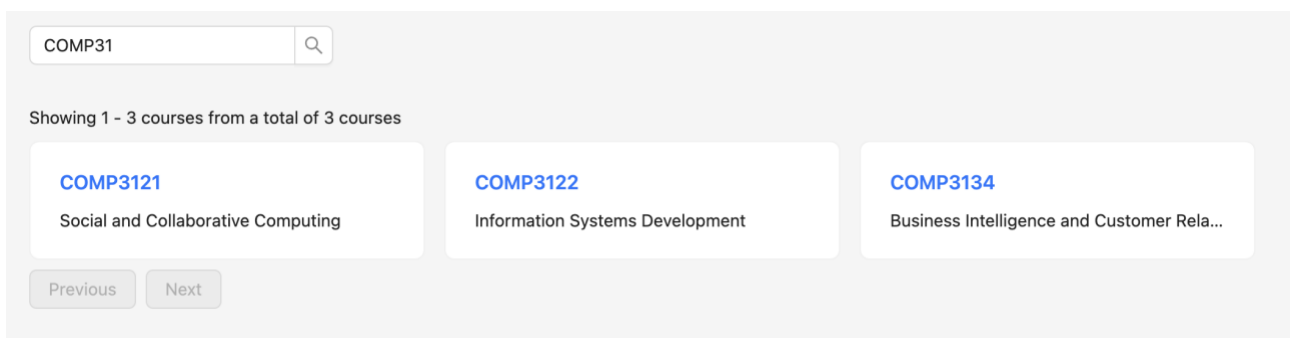


Figure 9. Screenshot of searching courses.

Moving onto the “/course/[code]” page, this page shows the details of each course using the course code, such as going to the “/course/COMP3211” page would result in the following Figure 10. below.

# COMP3211 Software Engineering

Take course

## Reviews

★★★★★ nicb112  
forgot to add message

## Semester Offerings

Semester 1	Semester 2	Summer Term
Available	Unavailable	Unavailable

### Pre-requisites

COMP2011

## Similar Courses

[COMP3235 Software Project Management](#)  
[COMP2021 Object-Oriented Programming](#)  
[COMP4123 Business Process and Workflow Management](#)

Figure 10. Screenshot of page "course/COMP3211".

As seen in Figure 10. above, this page shows all the reviews given by students, the semester availability, course pre-requisites and similar courses generated by Content-Based RS. The “Take course” button allows the user to review the course in the “/profile” page.

Lastly, the “/profile” page shows recommendations generated by Collaborative RS and courses that the user have taken shown in Figure 11. below.

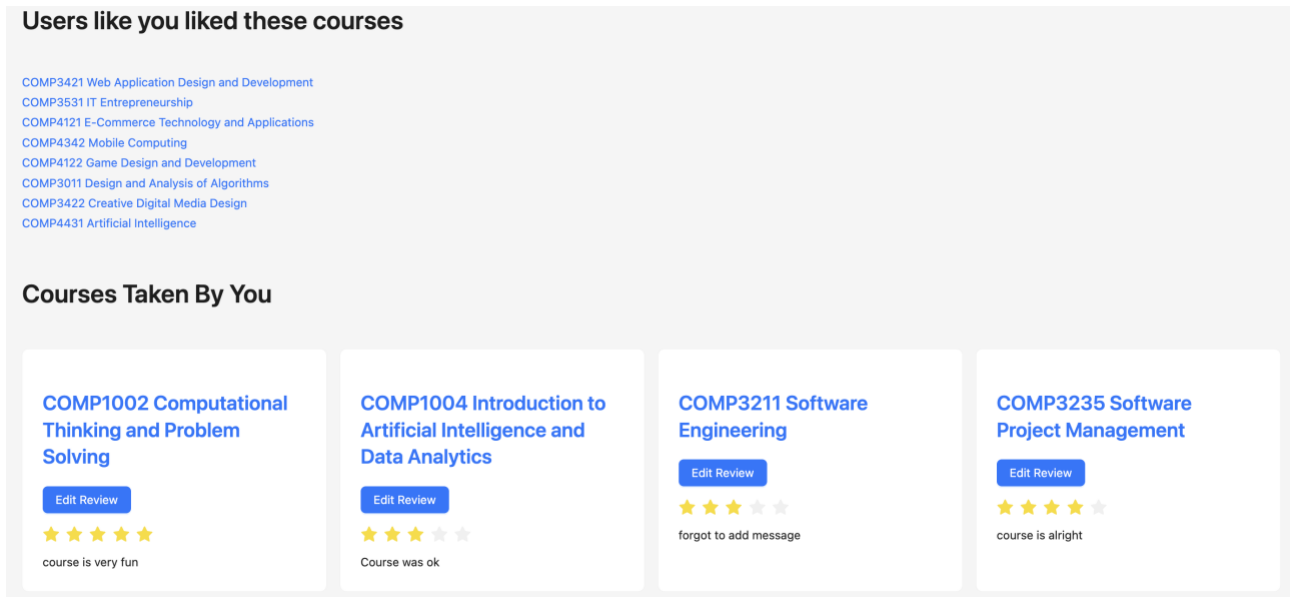


Figure 11. Screenshot of profile page.

Within the first section, as loading the trained model in the API server takes time, caching the first response containing course recommendations from Collaborative RS using “@reduxjs/toolkit” helps improve the website UX and reduce load on the API server. In the “Courses Taken By You” section, users are able to add/edit reviews and the data will be sent to the database for Collaborative RS model training.

## 4.5. API Server

This section is split into two parts, namely allowing CRUD operations onto the database, and integrating both Collaborative and Content-Based RS into the API server.

Fortunately, Django provides built-in Object Relational Mapper (ORM) that allows creating Models with relations and constraints without having to write pure SQL statements. Alongside that, Django provides script that loads the stored course data in JSON format into the database. To send responses in a readable format by the front-end web application, the response would be sent in JSON format. With the usage of “**djangorestframework**” library, it was a relatively easy process to allow the front-end website to make CRUD request to the database. For security, when users login to their account, they are given JWT that will be asked for subsequent request on protected routes to ensure that the data is only for the owner to access.

Once finished, integrating the Collaborative and Content-Based RS is the next step. Integrating each system requires changing the code to source the stored data (from reading CSV data to reading data

from the database). Alongside that, the results generated by both RS needs to be converted into a JSON response that is usable for the front-end website application. Lastly, “**apscheduler**” was used to schedule the task of training Collaborative RS model on a 30-minutes interval. The resulting Collaborative model are saved into “**server\_data/model**” folder, allowing for loading the model in future request from the front-end web application for recommendations.

## 4.6. Deployment

Initially, Platform as a Service (PaaS) provider Vercel was chosen to deploy the survey application, front-end website application and API server. This is due to the simple deployment steps needed to get a fully accessible website on the internet and I was most familiar with this approach. However, as the “**pyspark**” library was added to the API server, the API server couldn’t be deployed on Vercel due to the application size exceeding the size limit (250MB). Because of this, another approach is to use Infrastructure as a Service (IaaS) providers like AWS, DigitalOcean. In the end, DigitalOcean was chosen because it offers the lowest cost out of all the other IaaS providers. After purchasing a Virtual Machine (VM), initial setups were done to prepare the VM for hosting Dockerized (containerized) applications on to the internet. Caddy, a web server that is an alternative to NGINX is used to handle incoming requests towards the Caddy web server to the front-end application or the API server. Finally, domain names were set so that the Recommender System is accessible at this [link](#).

## 4.7. Project Schedule

Table 2. Project schedule.

Date	Milestone	Status
October 2022	<b>Deliverables of Project Outline:</b> <ul style="list-style-type: none"> <li>- Detailed Project Plan</li> </ul> <b>Research:</b> <ul style="list-style-type: none"> <li>- Currently known recommendation techniques</li> <li>- Technologies that can be used for this project (React, Django)</li> </ul>	Done
November 2022	<b>Implementation:</b> <ul style="list-style-type: none"> <li>- Prototype user interface of the application</li> </ul>	Done
December 2022	<b>Research:</b> <ul style="list-style-type: none"> <li>- Do in-depth analysis of other works</li> </ul>	Done

	<b>Implementation:</b> <ul style="list-style-type: none"> <li>- Create a prototype front-end website for displaying courses</li> <li>- Finalize which recommendation techniques to use</li> <li>- Gather relevant data about PolyU computing courses for recommendation</li> <li>- Design DB relations</li> </ul>	
January 2023	<b>Deliverables of Interim Assessment:</b> <ul style="list-style-type: none"> <li>- Interim report</li> <li>- Presentation video and PowerPoint file</li> <li>- Meeting notes with supervisor</li> </ul> <b>Research:</b> <ul style="list-style-type: none"> <li>- Connecting the front-end website with the API server</li> <li>- User login and registration</li> </ul> <b>Implementation:</b> <ul style="list-style-type: none"> <li>- API server with basic functionalities</li> <li>- DB implementation and integration to the API server</li> <li>- Implement user login and registration functions to both website and API server</li> <li>- Input all gathered data about PolyU courses to the DB</li> </ul>	Done
February 2023	<b>Implementation:</b> <ul style="list-style-type: none"> <li>- Add recommendations using CBF to the API server based on each course' overview</li> <li>- Continue work on the website to gather user data for CF recommendations</li> <li>- Offer CF recommendations when enough user data is gathered during deployment</li> </ul>	Done
March 2023	<b>Implementation:</b> <ul style="list-style-type: none"> <li>- Testing app by asking a sample of computing students to try the application</li> <li>- Optimize both website and API server if needed</li> <li>- Bug fixes if needed</li> </ul>	Done
April 2023	<b>Deliverables of Final Report:</b> <ul style="list-style-type: none"> <li>- Final Report</li> <li>- Final Presentation with project demonstration</li> </ul>	Done



## Chapter 5. Preliminary Evaluation

In this chapter, experimental results and several shortcomings of this project that will be pointed out to describe how these parts can be improved for future iterations of this project.

### 5.1. Experimental / Survey Results

After creating the CF System, a placeholder data with bias towards a certain subject was inserted to the RS. It was quickly found that the system had also generated recommendations with bias towards this subject as expected. The result meant that the system is working and may adapt recommendations to new data from the front-end application.

While gathering the survey data, it was found that whenever a new unique survey data was added to the “**survey**” table for the CF System to use, the recommendations change which concluded that the CF System worked as expected.

On the other hand, loading generated CF recommendations on first request takes around 1 to 2 seconds to complete which could affect the User Experience as no one likes to wait for slow web applications. It is important to note on subsequent requests that it takes less than 1 second to load the CF recommendations again.

### 5.2. Small Data Size

It is important to point out that the number of data generated by the user is small which is an issue. As of this writing, there has been 10 surveys submitted to the survey application, which has a possibility of making the recommender system giving biased results. Figure 12. is shown below to illustrate the data distribution.



*Figure 12. Demographic of people filling in the survey*

With 10 respondents, there were 164 rows of course reviews recorded. Alongside the small number of respondents, these rows in the database were also filtered to remove computing compulsory courses as recommending compulsory course is counter-intuitive towards computing students. However, this would also make the data submitted by the accounting student in PolyU irrelevant as they had mostly taken compulsory computing courses.

To address this issue, future iterations of this system may use PolyU computing students' course records to generate more reliable and accurate recommendations. Furthermore, it could be possible to omit the process to remove compulsory courses so that students from other departments may benefit from the recommendations. In the meantime, a combination of test data with around 1300 rows of test response and responses from the survey are used together to provide recommendations.

### 5.3. Collaborative Recommendation Model Storage Alternative

To solve the performance issue mentioned in section 5.1, an alternative to storing the generated recommendations over storing the trained model in the server's filesystem is to store the outputted recommendations in the database as string data type. This allows faster retrieval of data and decreases the user waiting time albeit it is unclear how much more effective it is over the method that was used in this project. However, this method is limited by scalability issues as when the user



base grows large, huge number of data insertions to the database could affect the performance of the whole system, therefore more research needs to be done on this to see which approach is better.

## 5.4. Localization

It has been mentioned that localizing the website application and course subject description will be done as PolyU is a diverse community stemming from different cultural background and languages. However, there is always a risk of losing the original meaning of the course description when it is localized into another language, ultimately resulting in unexpected recommendations. It is concluded that localization may not be a necessary feature for this application, hence the localization feature is incomplete in the final build.

## Chapter 6. Conclusion

In view of the difficulties of deciding which university courses to take, it is imperative to help reduce Information Overload issue by providing a web-based hybrid RS that offers personalized course recommendations. This project aims to remove the steps needed for students (e.g., reading course syllabus, trying out trial classes, etc.). to conclude whether a course is suitable for them. Alongside that, this project serves to provide a building block for universities to develop a RS as this project was limited by the relatively small number of respondents. This project is accessible at this [link](#)(The project might be disabled after April 2023 due to me having to pay more than US\$10 to rent the virtual machine from DigitalOcean as the scheduled model training task that runs every 30 minutes requires high bandwidth). GitHub repository is available at this [link](#).

## References

- [1] N. D. S. Chetty *et al.*, “Learning Styles and Teaching Styles Determine Students’ Academic Performances.,” *Int. J. Eval. Res. Educ.*, vol. 8, no. 4, pp. 610–615, 2019.
- [2] J. B. Schmitt, C. A. Debbelt, and F. M. Schneider, “Too much information? Predictors of information overload in the context of online news exposure,” *Inf. Commun. Soc.*, vol. 21, no. 8, pp. 1151–1167, 2018.
- [3] D. Dean and C. Webb, “Recovering from information overload,” 2011.
- [4] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, “Recommender system application developments: A survey,” *Decis. Support Syst.*, vol. 74, pp. 12–32, Jun. 2015, doi: 10.1016/j.dss.2015.03.008.
- [5] Y.-Y. Wang, A. Luse, A. M. Townsend, and B. E. Mennecke, “Understanding the moderating roles of types of recommender systems and products on customer behavioral intention to use recommender systems,” *Inf. Syst. E-Bus. Manag.*, vol. 13, no. 4, pp. 769–799, 2015.
- [6] D. Das, L. Sahoo, and S. Datta, “A survey on recommendation system,” *Int. J. Comput. Appl.*, vol. 160, no. 7, 2017.
- [7] P. B. Thorat, R. M. Goudar, and S. Barve, “Survey on collaborative filtering, content-based filtering and hybrid recommendation system,” *Int. J. Comput. Appl.*, vol. 110, no. 4, pp. 31–36, 2015.
- [8] F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, “Recommendation systems: Principles, methods and evaluation,” *Egypt. Inform. J.*, vol. 16, no. 3, pp. 261–273, Nov. 2015, doi: 10.1016/j.eij.2015.06.005.
- [9] E. Rahm and H. H. Do, “Data cleaning: Problems and current approaches,” *IEEE Data Eng Bull*, vol. 23, no. 4, pp. 3–13, 2000.
- [10] J. Horrigan, *Wireless internet use*. Pew Internet & American Life Project Washington, DC, 2009.