



## Online Course Recommender System



## Team Members

**Lim Jun Ming**  
**Sarah Elita Shi Yuan Wong**  
**Zhang Yunduo**

A0231523U  
A0231507N  
A0231349H

## CONTENTS

<b>1 EXECUTIVE SUMMARY .....</b>	<b>2</b>
<b>2 INTRODUCTION .....</b>	<b>3</b>
<b>2.1 BACKGROUND .....</b>	<b>3</b>
<b>2.2 OBJECTIVES .....</b>	<b>4</b>
<b>2.3 MINIMUM VIABLE PRODUCT (MVP) .....</b>	<b>4</b>
<b>3 SYSTEM ARCHITECTURE .....</b>	<b>5</b>
<b>4 KNOWLEDGE MODEL.....</b>	<b>6</b>
<b>4.1 OVERVIEW .....</b>	<b>6</b>
<b>4.2 KNOWLEDGE ACQUISITION.....</b>	<b>6</b>
<b>4.3 KNOWLEDGE DISCOVERY .....</b>	<b>8</b>
<b>4.4 KNOWLEDGE REPRESENTATION .....</b>	<b>10</b>
<b>5 RECOMMENDATION REASONING SYSTEM.....</b>	<b>12</b>
<b>5.1 OVERVIEW .....</b>	<b>12</b>
<b>5.2 FEATURE SELECTION.....</b>	<b>13</b>
<b>5.3 FEATURE ENGINEERING &amp; EXTRACTION.....</b>	<b>14</b>
<b>5.4 RECOMMENDATION MODULE.....</b>	<b>15</b>
<b>5.5 OUTPUT .....</b>	<b>17</b>
<b>6 SYSTEM IMPLEMENTATION .....</b>	<b>18</b>
<b>6.1 SYSTEM FRONTEND.....</b>	<b>18</b>
<b>6.2 USER INTERFACE.....</b>	<b>18</b>
<b>6.3 SYSTEM BACKEND .....</b>	<b>22</b>
<b>6.4 SYSTEM INTEGRATION .....</b>	<b>23</b>
<b>6.5 SYSTEM EVALUATION.....</b>	<b>24</b>
<b>7 CONCLUSION .....</b>	<b>26</b>
<b>APPENDIX A – REFERENCES.....</b>	<b>28</b>
<b>APPENDIX B – USER MANUAL GUIDE .....</b>	<b>29</b>
<b>APPENDIX C – PROPOSAL.....</b>	<b>37</b>
<b>APPENDIX D – MAPPING OF SYSTEM FUNCTIONALITIES TO COURSES .....</b>	<b>41</b>
<b>APPENDIX E – INDIVIDUAL MEMBER REPORT.....</b>	<b>42</b>

## 1 EXECUTIVE SUMMARY

---

With digital transformation across the world, the education sector has shown positive initiatives to embrace e-learning or online learning in the past decade. The adoption is further accelerated since Covid-19 outbreak. Schools, universities and institutions are forced to shift their operation online in order to comply with safe distancing measure. Inevitably, this has caused an exploding growth in the online learning market. People are looking for alternatives to bricks and mortars institutions and starting to embrace the convenience of online learning.

Nowadays, there are many online platforms that offers courses in many different topics and languages. With an internet connection, they promote the spirit of learning anything, anytime and anywhere. However, inquisitive learners searching for online courses to learn often find themselves experiencing option fatigue as there are so many options and platforms to choose from. This often leads to a longer search time and effort in order to find the best matching course to start learning.

Our team have designed and built an Online Course Recommender System that aims to be a one-stop-platform that solves this inefficiency. Given a learner's preference, our system is able to perform meaningful cross-platform course recommendations. Learners would no longer need to spend countless hours visiting each of the online course platforms in order to find a best matching course for their needs.

The knowledge base of our system is constructed through data mining from the three major online course platform: Coursera, Edx and Udemy. While building the knowledge model, we utilized tools such Python and web-scraping methods to scrape course data from the platforms. The knowledge base is represented and stored in the form of SQL database. A recommendation reasoning system is developed within our system that is capable of performing cross-platform course recommendations using a customized content-based filtering approach. The frontend user-interface is built based on a web framework using tools such as JavaScript and HTML. To integrate the system frontend and backend including the database and recommendation reasoning system, we utilized the Python Flask app to manage the routes and request across the system.

The first version of our Online Course Recommender System serves as a prototype for proof-of-concept. Besides, we have identified future improvements such as adding more online course platforms into our knowledge base, adding more user functionalities within our system as well as further tuning and improving the recommendation algorithm.

## 2 INTRODUCTION

### 2.1 BACKGROUND

In recent years, there has been a growing demand for online courses. This growth was also further accelerated due to the COVID-19 outbreak. In particular, the global Massive Open Online Course (MOOC) market had an estimated value of USD 6845.4 million in 2020. It is expected to increase to USD 18925.18 million by 2026, with an estimated Compound Annual Growth Rate (CAGR) of 18.13%, from 2021 to 2026 [1]. The number of MOOCs available has also grown steadily with the growth in demand. According to Class Central, it was estimated that from 2012 to 2020 there were a total of 16,300 MOOCs rolled out by 950 universities globally (excluding China), and each year the number of MOOCs rolled out had been increasing [2]. This is shown in the figure below.

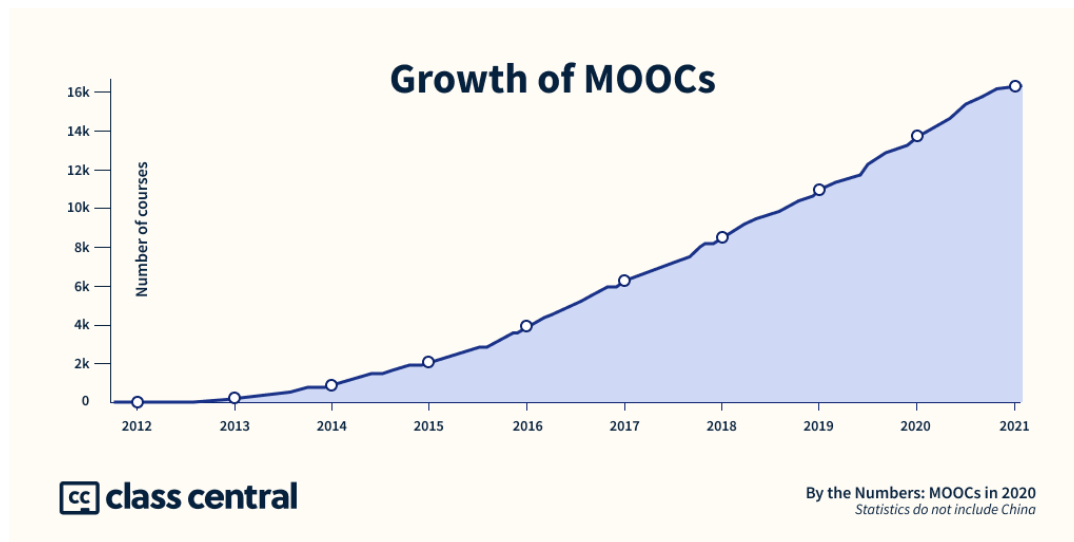


Figure 1. Growth of MOOCs from 2012 to 2021 [2]

With such large numbers of MOOCs available, users may find it difficult to navigate through all of them to find the course that best suits their needs. A good recommendation system will assist users to easily find suitable courses which match their specific requirements.



## 2.2 OBJECTIVES

In this project, we aim to develop a cross-platform recommendation system for online courses, especially MOOCs. The system will consolidate the information of available courses across multiple online course providers and then recommend each user the courses that best matches their needs.

This helps users to increase their search scope and potentially discover courses that they might not have otherwise found. It also saves users time and the hassle of combing through multiple course provider sites to find a course that matches their needs. The system will become an one stop solution for users to search for courses on skills that they would like to learn through e-learning.

## 2.3 MINIMUM VIABLE PRODUCT (MVP)

For our MVP, we have built an online course recommendation system that is based on the structure of an web application. The webapp consists of a few utility features which helps user with searching and managing online courses easily. For course recommendation, a logged-in user will be able to input some text describing the skills or courses they are looking to learn, along with a few other criteria such as the course duration, difficulty and, an option of whether they are willing to pay for the courses. Our system will then perform the recommendation inference within the reasoning system unit based on the user input and return a list of the best matching courses. In addition, historical search results as well as user's personal list of liked courses are store within each user profile that sits within the database.

The knowledge model of our MVP is built based on data scrapped from three selected popular online course platforms, namely Coursera, Edx and Udemy. The recommendation reasoning system are designed to optimize the recommendation of courses retrieved from the knowledge base given a user's query. The frontend user interface is designed to be clean and user friendly so that user can easily manage and search for courses or skills and be able to navigate directly to the course provider to start learning.



### 3 SYSTEM ARCHITECTURE

We have designed the system architecture based on the utilization of various tools. The system frontend, system backend and knowledge models form the foundation blocks of our system architecture. We have built a light weight frontend system which is responsible for user interaction interface using HTML and JavaScript which are wrapped within the Bootstrap frontend web framework. The backbone of our system backend is built using Python and Flask app libraries. The Flask app is responsible for communicating to the various modules with the system architecture. It will manage frontend web page routing and requests as well as interacting with the database and the recommendation reasoning system for query requests.

The recommendation reasoning system is the core cognitive processing module that utilizes various methods such as Natural Language Processing (NLP) and rule-based inference to perform the course recommendations. Whenever a query request is triggered, it retrieves course information from the database and user inputs and preferences from the Flask app to perform the recommendation inference. The SQLite database sits within system backend will interact with the Flask app and the recommendation reasoning system for query requests. Various information such as user profile and account information as well as the full course data knowledge base are stored in the database.

On the other hand, we have the knowledge model which serves to build the course data knowledge base. The web scrapping processes are performed within the knowledge model utilizing Python and web scrapping libraries such as BeautifulSoup and Selenium. We have also utilized Google Collaboratory Notebooks for certain tasks within the knowledge model such as performing knowledge discovery and data pre-processing.

The system architecture is designed with online hosted web application in mind. However, for the scope our MVP, we have limited the application to work and run with local machine only due to the large size of database and the higher cost of hosting the database online as a result. Figure 2 shows an overview of our system architecture design.

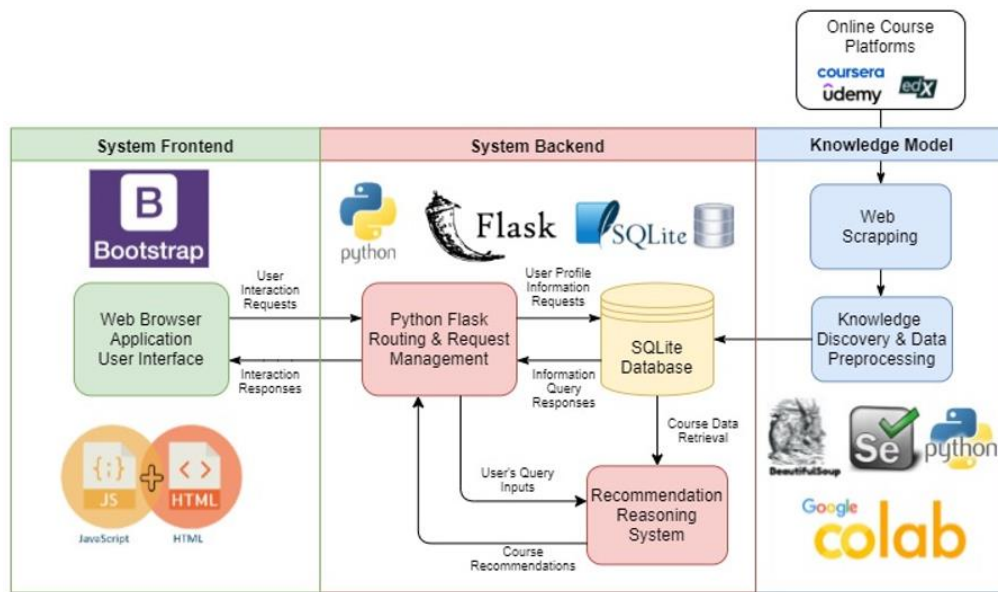


Figure 2. Overview of System Architecture

## 4 KNOWLEDGE MODEL

### 4.1 OVERVIEW

In order to develop the cross-platform recommender system, we have designed a knowledge model that forms the knowledge base where the recommendation reasoning will be conducted on. The knowledge model consists of three key aspects as follows:






Figure 3. Key Aspects of Knowledge Model

They are designed and built in the sequence as above. Each step consists of many important tasks that are performed extensively before moving on to the next. In the following sections, the design and approach for each of the three key aspects will be discussed in detail below.

### 4.2 KNOWLEDGE ACQUISITION

The objective of the MVP is to allow cross-platform course recommendations based on the user's preference or query. In order to achieve that, we have performed knowledge acquisition as the first step in building our knowledge model. The course information will form the core knowledge that the knowledge model will base on.

Nowadays, there are multiple online course platform that offers courses in a wide variety of subjects and topics. To ensure our knowledge model to be built based on a wider scope of course subjects and topics, we have explored various online course platforms and selected three platforms below for our MVP deployment [3-5]. Each of the platforms selected below has their own strengths and weaknesses which will complement each other when they are combined to build our knowledge model.

Platform	Strengths	Weaknesses
	Courses are designed in collaboration with leading companies and universities	Course offerings are more focused on a few selections of topics
	Courses are designed and conducted by leading education institutions	Smaller catalog of courses offered
	Very large catalog of courses and offerings are in wide variety of topics	Most of the course offerings are paid courses conducted by private institutions

As all the course information are available online and open to public, we have performed the data mining approach – web scrapping to extract the course information. We have utilized Python as well as a selection of accompanying libraries such as BeautifulSoup and Selenium for conducting the web scrapping on the three online course platforms.

The web scrapping process is performed separately due to the different complexity of web page design and structure across the three platforms. The web scrapping tool coded with Python are run separately in order to catered to the different html design and structure of the online course platforms.



Below are some examples of the landing and course webpage design across the three platforms.

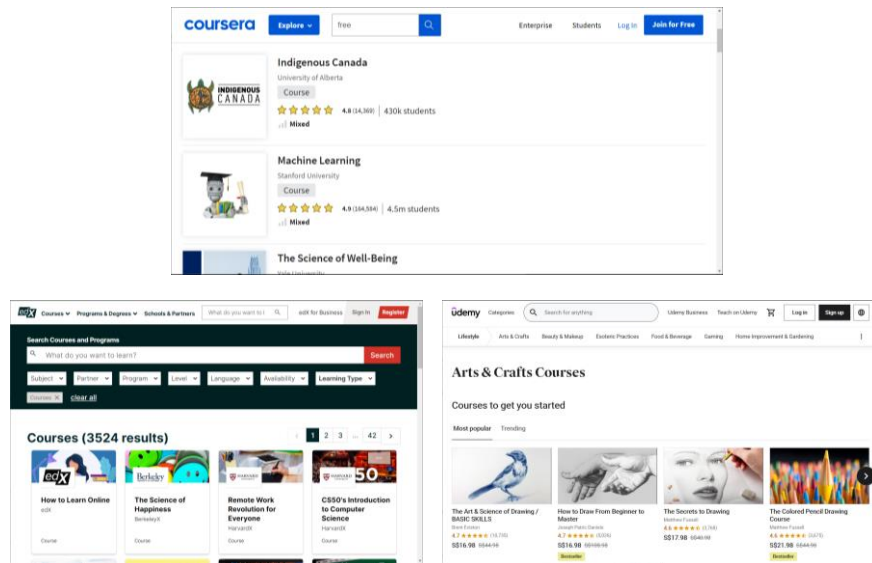


Figure 4. Webpage Interface of Online Course Platforms (Coursera, Edx, Udemy)

While the web scrapping process can be performed routinely for knowledge base refresh and updates, the context of our MVP is limited to the snapshot of data scrapped in *August 2021*. One common challenge in the web scrapping process is that routine maintenance of the scrapping tool is crucial if a continuous web scrapping process is required. This is due to the owners of the webpages or platforms will push updates, changes or even redesign their webpage interface and structure routinely in order to promote or improve the visual appeal of their webpage to visitors.

Based on the webpage exploration conducted on each platform, we observed that the type of course information available across the three platforms can be very different. The course data extracted from each of the platform are stored temporary in the comma separated values (CSV) table format before they are combined and used in the later stages of knowledge modelling. Below is a summary of the type of course information that are extracted from each of the online course platforms.

Platforms	coursera	edx	udemy
Title	✓	✓	✓
Category	✓	✓	✓
Course URL	✓	✓	✓
Course Provider	✓	✓	✓
Short Descriptions		✓	✓
Long Descriptions	✓	✓	
Course Difficulty	✓	✓	✓
Course Duration	✓	✓	✓
Course Language	✓	✓	✓
Free Option	✓	✓	✓
Certification Cost		✓	✓
Course Ratings	✓		✓
Number of Enrolment	✓	✓	✓
Instructors	✓	✓	
Requirements	✓		



### 4.3 KNOWLEDGE DISCOVERY

With the raw course information extracted, we performed knowledge discovery and merged the datasets from the three online course platforms. In the process of knowledge discovery, we studied the data types and content of each data field to determine the importance and relevance of the data extracted from web scrapping.

We observed that the data extracted for each of the platforms has various issues which requires attention. Some examples of issues are missing data, inconsistent format of data values across the platforms and data values are in different unit of measurement standard. Various data cleaning procedures such as removing or replacing empty data fields with zero values and aligning the data type across the table are conducted to ensure data consistency.

Based on the list of raw data fields extracted, some of them are not useful for building the knowledge model as they do not serve any value for the recommendation system. Two such data fields that are found in the raw data fields are the course instructors and course requirement. They are identified as not useful in the recommendation system as generally they have less impact on the take up rate or enrolment rate of online courses. Hence, they are discarded from the dataset during the knowledge discovery process.

To build an efficient knowledge model, adjustment and transformation must be applied on some data such that the efficient and useful knowledge representation can be extracted.

Below is a summary of three adjustments and transformations performed:

Adjustment & Transformations	Motivation
<b>Course duration are encoded into 3 types – Short, Medium and Long duration</b>	This is to ensure that the course duration data is on the same measurement across the platforms. Three categories of duration are used for simpler groupings instead of the actual number of hours.
<b>Course difficulty are encoded into 3 types – Introductory, Intermediate and Advanced</b>	This is to ensure that the course difficulty level is on the same measurement across the platforms. Three categories of difficulty level are used to represent the common three tiers of course difficulty.
<b>Derivation of popularity index</b>	As rating information is not available in some platform, we calculate the popularity index which is derived using combination of number of enrolment and ratings data. This ensure that the popularity of each course can be compared based on the common popularity index.
<b>English language courses</b>	For the context of our MVP, we limit the course recommendation to only courses conducted in English.

With the data cleaning, adjustment and processing completed, the final dataset is formed by combining the processed dataset into one full complete dataset. Below is a list of data fields and corresponding type properties in the complete dataset:

Data Fields	Data Type
Title	String
URL	Hyperlink Address
Categories	String
Short Description	String
Long Description	String
Difficulty	Categorical {1: Introductory, 2: Intermediate, 3: Advanced}
Duration	Categorical {1: Short, 2: Medium, 3: Long}
Free Option	Binary {1: Free, 2: Not Free}
Number of Enrolment	Integer
Rating	Float
Certification Cost	String
Language	String
Subtitles	String
Platform	Categorical {0: Edx, 1: Udemy, 2: Coursera}
Provider	String
Course Image Holder	Hyperlink Address

Below is an illustration of the knowledge discovery process:

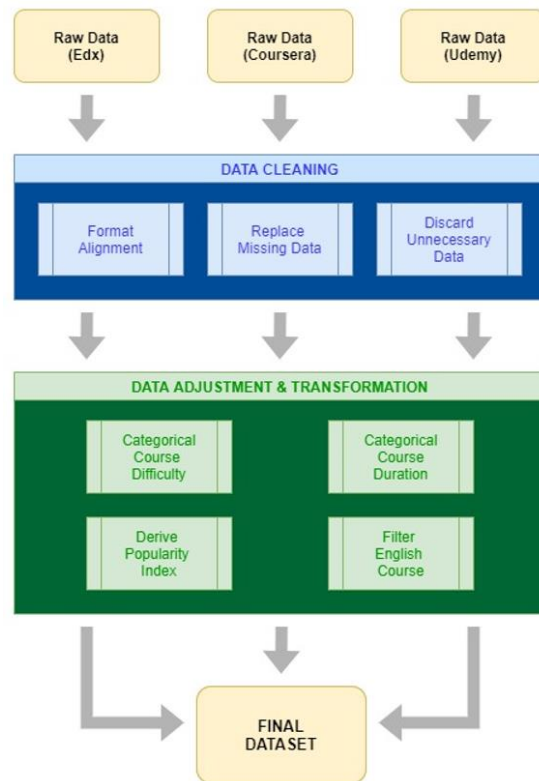


Figure 5. Knowledge Discovery on Raw Course Dataset

## 4.4 KNOWLEDGE REPRESENTATION

### ***Database***

The pre-processed data from the knowledge discovery step then form the knowledge base for the courses information and are stored in a database created using SQLite. Besides the course information, the database as act as a central knowledge base for other information related to user profile and interactions such as user login information, user query histories as well as courses marked and noted by users within the user's account. A total of five tables are stored inside the database which are connected to each other in order to function as a core knowledge base for the system.

### ***Table - course***

The *course* table stores the courses information of all three platforms which are built from the pre-processed data from the knowledge discovery step. A total of 18 fields are structured within the *course* table. It includes some important information about each course such course title, course URL, description, categorical difficulty, categorical duration, language, popularity index, etc. The *courseID* field is unique and act as a primary key which allow other tables to reference and connect to.

### ***Table - user***

The *user* table stores all the user authentication data such as *userID*, name, username, password. The *userID* field is unique and also act as a primary key which allow other tables to reference and connect to. New entry will be inserted to the table whenever a new user account is created within the system.

### ***Table - query***

The *query* table contains details of the search history of all users. It stores individual search inputs according to the *userID* field and the user *query\_count* field. For each query, the topic, duration, difficulty, and free course availability option preferences are stored in fields *query\_text*, *query\_duration*, *query\_difficulty* and *query\_free\_option* respectively for future reference. There is also a data field *query\_time* which stores the timestamp of the query and is filled in by default whenever a query is performed by the user.

### ***Table - recommendation***

The *recommendation* table stores all the past recommendations made for each of the query performed by all users. The recommendations are stored according to the *userID* field and the *query\_count* field. For example, for a particular *userID* and *query\_count*, there will be multiple rows containing a recommendation ranking and the corresponding *courseID*. Multiple rows of new entry will be inserted here whenever a query is performed by a user.

### ***Table - favourite***

The *favourite* table stores the all the courses liked and marked for all users. Each *userID* can have multiple rows, each with a *courseID* for a user's liked course. This table serves as a storage for user liked courses so that it allows users to build their own list of courses which they like or would like to mark down for future review within our system.

Below is an overview of the knowledge base structure stored using an SQLite database system.

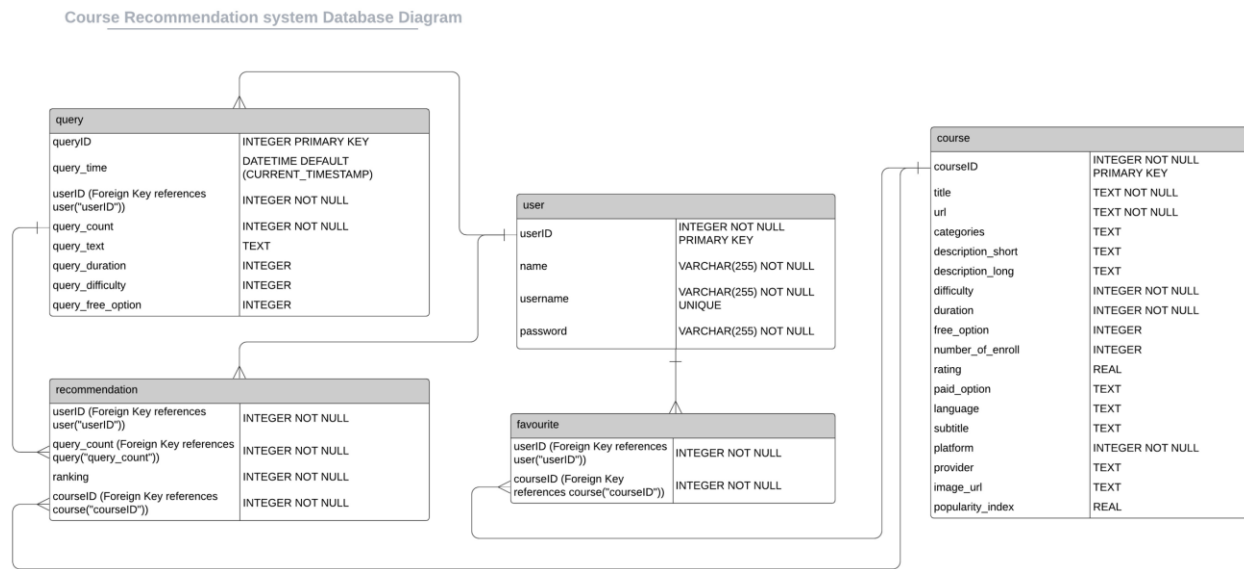


Figure 6. Knowledge Base – SQLite Database Structure

### User Inputs

Another type of knowledge representation is the set of user inputs information as variables that is linked and connect through both the frontend and the backend systems. The user inputs information is not only stored in the database table, *query* but is also transmitted as python variables from the Flask app directly to the recommendation reasoning system where recommendation inference can be performed. The user inputs information includes:

Input Fields	Data Type	Description
Skills or Topics	STRING	A short text description about the skills or course user looking for
Course Duration Preference	INTEGER	User preferred course duration
Course Difficulty Preference	INTEGER	User preferred course difficulty
Course Free Option Preference	INTEGER	Whether user are open to paid courses

The specifications of user interface for the user inputs are discussed in the frontend system design Section 6.1 and the utilization of the user inputs for recommendation inference are discussed in the next section on recommendation reasoning system.

## 5 RECOMMENDATION REASONING SYSTEM

### 5.1 OVERVIEW

Various processes are conducted within the reasoning system where both the user query information and the knowledge base are utilized for course recommendation reasoning. The recommendation module forms the core processing unit of the reasoning system in our MVP application. The reasoning system algorithms and processes are fully developed within the Python ecosystem which utilizes some of the Natural Language Processing (NLP) libraries such as NLTK. Figure 7 below is an overview of the processes and modules work flow design within the reasoning system.

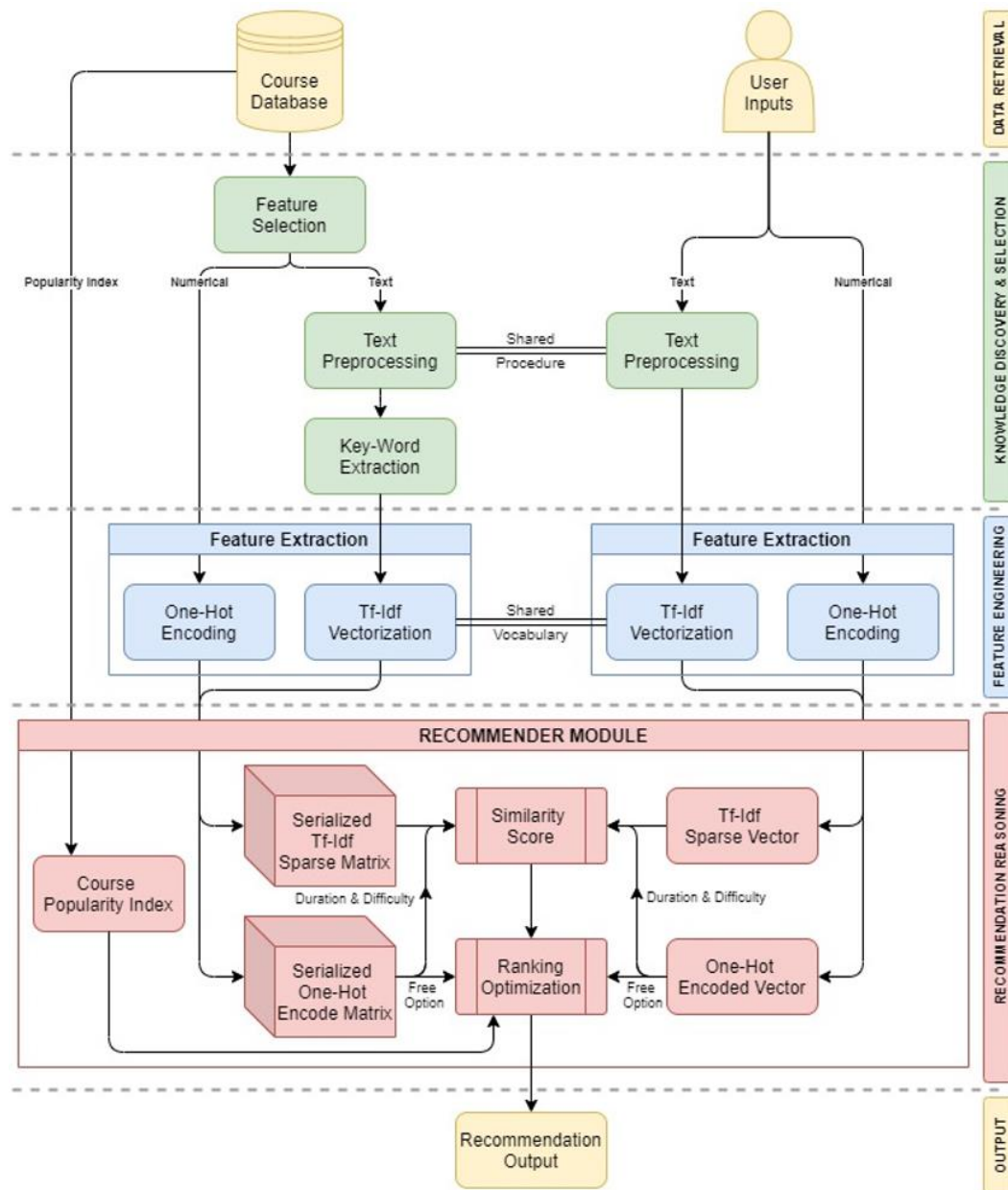


Figure 7. Recommender Reasoning System

## 5.2 FEATURE SELECTION

Based on the database build from the knowledge modelling, we can see that not all data fields are necessary useful in the recommendation reasoning process. In addition, the course data in the database are of variety of data types such as strings, integers and categorical values.

In order to develop the recommendation reasoning system, we have selected some data fields as features and are separated into two groups namely the text-based features and numerical features. The selection of features is designed based on the natural process and the motivation of how courses are usually selected by a human learner. This allow the reasoning process to effectively perform the recommendation reasoning that is catered or similar to the natural process of course selection by a human learner. A summary of selected data fields for the two groups are as follows:

Text-Based Features	Numerical Features
Course Title	Course Duration
Course Categories	Course Difficulty
Course Short Description	Course Free Option Availability
Course Long Description	

### *Text-Based Features*

The text-based features are all string-based data types that are retrieved from the database. Together, they form the descriptive information of each course. They are the key data in the coming stages which allows the matching of user's intent or preference to the course descriptions through NLP processes. Before the text-based features are feed into the NLP processes, we have performed text pre-processing to prepare the data for the next stage. Following are a list of text pre-processing processed on each of the selected text-based features.

Text Preprocessing	Motivation
<b>Removal of Non-ASCII characters</b>	As in the context of this project, we are only working with English language courses, all non-alphabetical and non-numerical characters are removed
<b>Lower Casing</b>	Lower casing all alphabetical characters
<b>Removal of White Spaces</b>	Removal of extra and unnecessary white spaces before or after the text
<b>Removal of Punctuations</b>	Removal of punctuations from the string which will not be useful for recommendation reasoning
<b>Tokenization</b>	Convert the strings into list of tokens
<b>Removal of Stop Words</b>	Removal of stop words from the list of tokens which will not be useful for recommendation reasoning
<b>Lemmatization</b>	Convert the tokens into lemma which is the basic forms of words

The same text pre-processing procedure is also applied on the user's intent inputs which also comes in the form of text. Similar to the courses text-based features, the pre-processed user's intent text inputs will be utilized in the later stages of the reasoning system for recommendation inference.

### ***Numerical Features***

The course duration, difficulty and free option availability retrieved from the database are selected as these are some of the important elements that a learner will look into during the natural course selection process. Similarly, from the previous Section 4.4 on user inputs, the numerical user inputs are retrieved and feed into the reasoning system where they will be utilized in the recommendation module. From here, we can see that the motivation of the design of user numerical inputs actually matches and is based on the selected numerical features from the course database.

## **5.3 FEATURE ENGINEERING & EXTRACTION**

### ***NLP Features***

In this step, we have applied some NLP methods to extract and transform the data into appropriate feature representation that will be useful for the recommendation module.

A quick exploration on the four text-based features would show that they all have different lengths of strings. The length for course title and categories would usually be less than 10 words, while course short and long descriptions can be more than 20 words especially the long descriptions which can come in the form of essay length. If we are to apply recommendation reasoning based on the aggregated pre-processed text of the four data fields, the recommendation simply will not perform well. This is due to the presence of noise in the form of common words which are not very descriptive of the course in the data especially the long description data field. Hence, to improve the recommendation using the four aggregated features, we have further applied an NLP technique, Rapid Automatic Keyword Extraction (RAKE) [6] which is able to extract key words from long text or essay. RAKE is only applied on the short and long description data field as we observed that most of the noise comes from these two data fields. After applying RAKE, we aggregated the resulting four data fields into one list of tokens which is essentially a bag of words.

With the combined tokens in the form of bag of words, they are ready for NLP based feature extraction. We have applied the Term Frequency-Inverse Document Frequency (Tf-Idf) Vectorization on the bag of words. This transforms the data into a sparse feature matrix which captures the normalized word frequency information of the text data. The word frequency information is a useful feature representation that will be utilized in recommendation inference. The sparse feature matrix is further serialized and is stored in Python compact binary format – Pickle and will be ready for use every time a recommendation inference is called.

For the pre-processed user's intent text inputs, we applied the Tf-Idf Vectorization directly to transform the data into a sparse feature vector. The RAKE is not applied on user's intent text inputs as the user input requirements are designed on the assumption that user will only input short sentences or phrases of words in the query forms.

### ***One-Hot Encoding Features***

For the numerical features from both the course data and user inputs, we applied a simple one-hot encoding to convert them into binary matrix and vectors which prepare them for processing in the recommendation module.



## 5.4 RECOMMENDATION MODULE

From figure 7 on recommender reasoning system, we can see that the recommendation module is a two steps processing unit which consists of the similarity score derivation based on the text-based sparse feature matrix and vectors as well as the ranking optimization processing which utilizes the one-hot encoded binary matrix and vectors. Overall, the architecture of the recommendation module is developed based on the content-based filtering recommendation approach with customizations built-in.

### *Similarity Score*

The first process in the recommendation module is the computation of similarity score using cosine similarity function [7]. The cosine similarity of two vectors of length  $n$  is defined as follows:

$$Sim_{cosine}(A, B) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

The similarity score of the user input Tf-Idf vector against each of the course feature vector in the Tf-Idf sparse matrix is computed. Similarly, the similarity score of the user input one-hot encoded vector against each of the course feature vector in the one-hot encoded matrix is computed at this stage. However, only the slice of the one-hot encoded vector and matrix on course duration and course difficulty are concatenated and used here for the similarity score computation. The course free option availability one-hot encoded vector and matrix are not utilized here.

Unlike in the usual content-based filtering procedure, the similarity scores are not immediately used to rank the most similar courses here. The ranking process is performed in the second step of the recommendation module. Besides, based on the experimentation performed, most of the courses with positive Tf-Idf similarity score given a query already form a good list of recommended courses.

### *Ranking Optimization*

The second step in the recommendation module is the ranking optimization process which is a customized ranking procedure that includes some element of rule based ranking and dynamic based ranking. The ranking optimization will utilize the two sets of similarity scores (Tf-Idf vectors and One-Hot encoded duration and difficulty) computed from the first step, the one-hot encoded free option availability vectors as well as the popularity index of each course directly retrieved from the database as inputs.

The ranking optimization consists of 5+1 stages of filters and sorting. These customized ranking and sorting procedures are designed such that the recommendation reasoning would take into account of course similarity and at the same to optimize the search and sorting of highest rated or most popular courses.

The details of the stages in Ranking Optimization Process are as follows.

***Stage 1: Positive Text Based Similarity Score Filtering***

We only include courses that have positive Tf-Idf based similarity score and discard courses that have Tf-Idf based similarity score of 0.

***Stage 2: Dynamic Filtering of Free Option Availability Requirement***

If user input to only show free courses, we will filter out the paid courses. Otherwise, we include both free and paid courses.

***Stage 3: Split Courses into One-Hot based Similarity Score Groupings***

Split the filtered courses from stage 1 and 2 into groupings based on the similarity scores calculated based on the one-hot encoded course duration and difficulty. Due to the course duration and difficulty can only takes in three categories (Short, Medium, Long; Introductory, Intermediate, Advanced), there are only three combination of similarity scores (0, 0.5, 1.0) calculated as a result. A similarity score of 0 here would indicate none of the course duration and difficulty matches the user's preference. A similarity score of 0.5 would indicate either duration and difficulty but not both matches the user's preference. Finally, a similarity score of 1.0 would indicate both duration and difficulty match the user's preference.

***Stage 4: Ranking and Sort by Popularity Index with Each Sub-Group***

Within each sub-group from step 3, courses are ranked and sorted using the popularity index.

***Stage 5: Append Results of Sub-Groups in Descending Order of Similarity Score Groupings***

The ranked and sorted course results from the sub group with one-hot based similarity score of 1.0 will be append the course results from the sub group with one-hot based similarity score of 0.5 and followed by that of similarity score of 0.

***Stage 6: Append Paid Courses if Below Result Count Limit (\*Dynamic)***

An extra stage 6 which is dynamic and will only trigger upon the following conditions are met:

- User's preference to only include free courses
- The total number of free courses result from stage 1 – 5 is less than 20 courses

The algorithm will repeat stage 3 & 4 on the paid courses that were filtered out in stage 2 and append the its sorted results to the output results of the free courses group in stage 5.

Figure 8 below is a summary of the steps within the ranking optimization process.

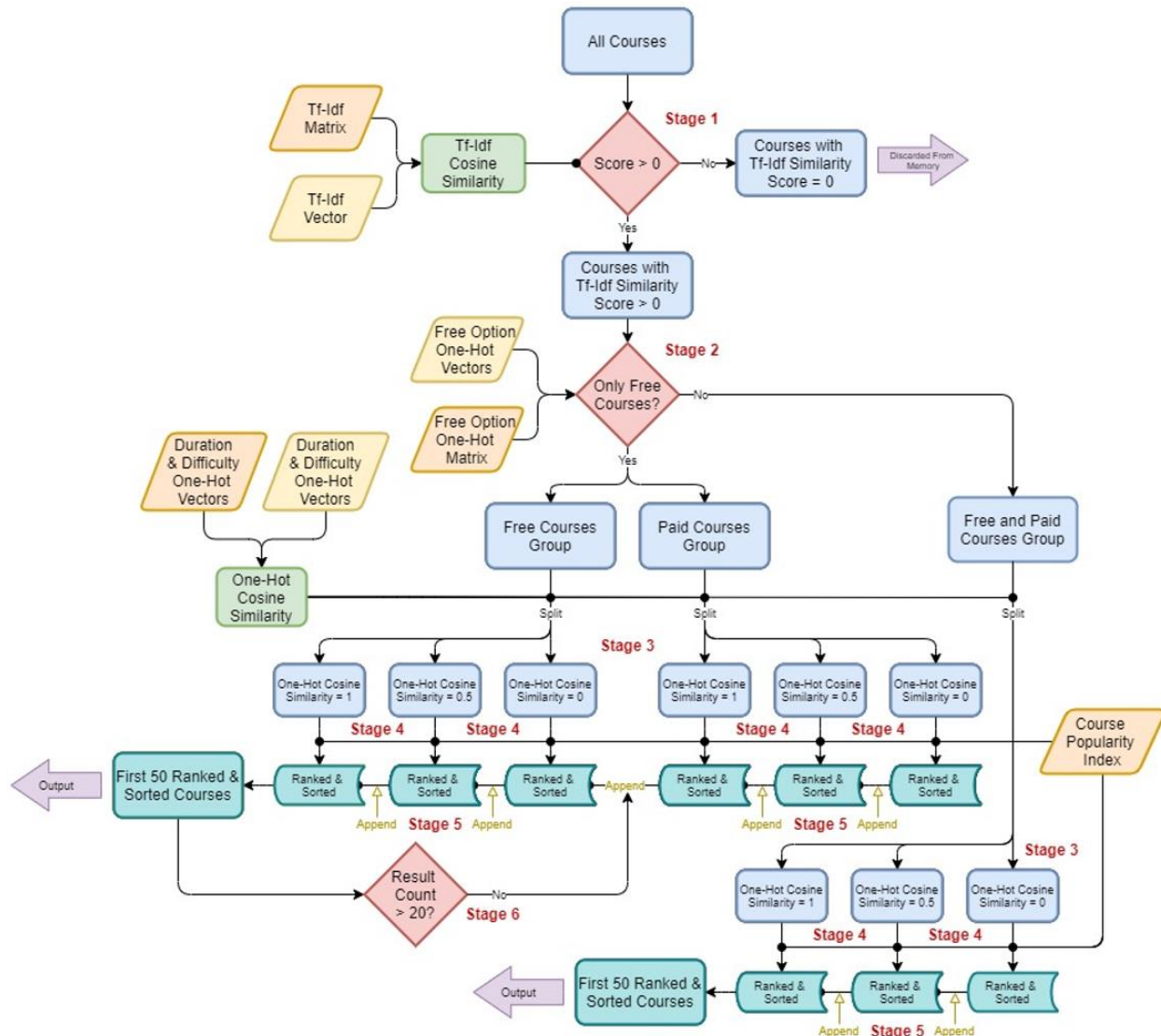


Figure 8. Ranking Optimization

## 5.5 OUTPUT

Finally, with the ranked and sorted courses based on the user's intent text inputs and preference, we take the first 50 courses as the output of the recommendation module. With the output, they are then sent to the frontend module for presenting the result to user.

## 6 SYSTEM IMPLEMENTATION

### 6.1 SYSTEM FRONTEND

As an MVP, the frontend user interface design ought to be simple but practical. The framework for the system frontend is a web application that is rendered based on HTML and JavaScripts. The user interface is rendered in web browser for interaction. The CSS framework and style adopted for our frontend system is the Bootstrap [8] templates which is a widely-use modern webpage styling methods. In addition, some of the on-page interactive functions was built using JavaScripts ajax. The frontend is integrated with the backend request management module which is built based on the python Flask application.

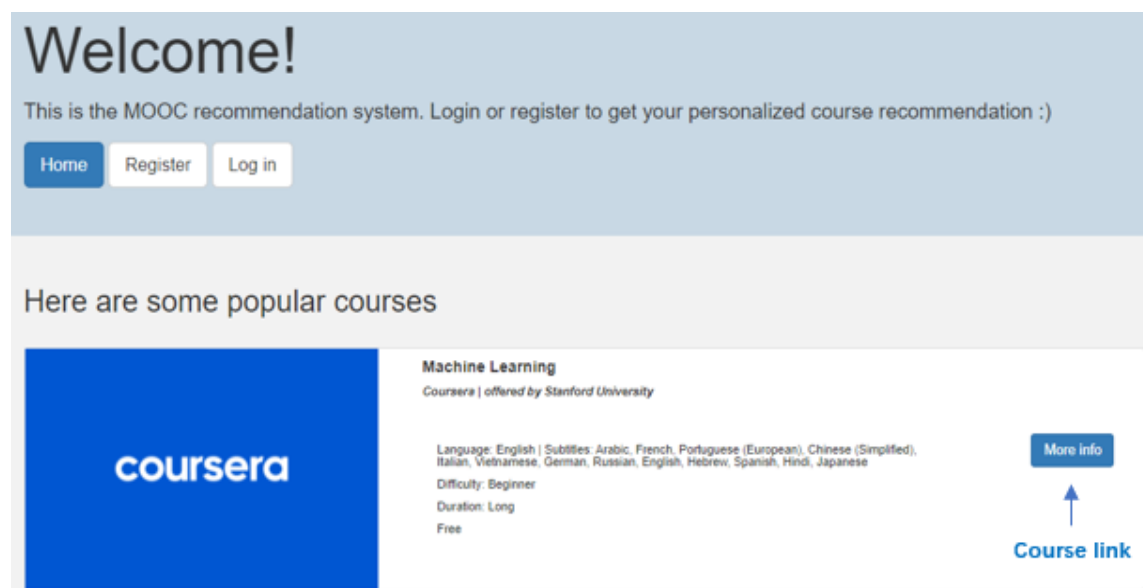


### 6.2 USER INTERFACE

Following are the essential elements of the user interface structure that is build upon the Flask frontend framework.

#### *Home page*

On this page, the popular courses from the database are retrieved and shown here, ordered by their popularity scores. The popularity scores used here is defined by the popularity index data field which are discussed in detail in the knowledge discovery section. On this page, no user account login is required. It is a landing page for all users. For all user interface web pages within the system that has a course information card, including the home page, will have a 'more info' button which will open up a new browser tab or window that redirect the user to the corresponding course webpage.



### User Sign up & Login

In order to record users' past search and favourite course, user registration and authentication functions are needed. The encryption technique used here is bcrypt, to store hash value of the users' password safely. The web application will self-validate on the user inputs before interacting with database, checking whether the inputs are legal and logical. For example, username must be unique, password length must be longer than 8 digits, user existence check and so on.

The image shows two side-by-side web forms for a MOOC recommendation system. Both forms have a light blue header with the text 'Welcome!' and a subtitle 'Sign up for a MOOC recommendation account here' (left) or 'Log in to your MOOC recommendation account here' (right). Below the header are three buttons: 'Home', 'Register', and 'Log in'. The left form is titled 'Sign up' and contains three input fields: 'Name', 'Username', and 'Password'. Below these fields is a 'Signup' button and a link 'Already have an account? Log in here'. The right form is titled 'Login' and contains two input fields: 'Username' and 'Password'. Below these fields is a 'Remember me' checkbox, a 'Login' button, and a link 'Not yet registered? Sign up here'.

### Recommendation Query

This is the page that collects user inputs and preferences on the skills or courses the user would like to learn. Total of four fields are available, text field for describing the skill or course, duration of the course, difficulty level, and user's preference on free or paid courses.

Let us know your preferences below:

What would you like to learn?

Duration\*

- ☐ No preference
- ☐ Short (1 - 10 hours)
- ☐ Medium (10 - 50 hours)
- ☐ Long (>50 hours)

Difficulty level\*

- ☐ Any
- ☐ Beginner
- ☐ Intermediate
- ☐ Advanced

Would you be open to paid courses?\*

- ☐ Yes. Show me both paid and free courses
- ☐ No. Show me free courses only

Get courses

### Recommendation Results


User is redirected from the 'get preference' page once the user inputs are posted to system. This page cannot be accessed without posting user inputs query information. And if you refresh the page, it will retrieve your search preference from the URL parameters, make recommendation, and render the page accordingly. This structure is dynamically adjusted to the changes in database, instead of showing the same result if the user refreshes the result page.

# Welcome testuser!

[Home](#)[Get Personalized Recommendation](#)[My Favourites](#)[History](#)[Logout](#)

Got your preferences!

Here are the recommendations based on your preferences




**Machine Learning**  
*Coursera | offered by Stanford University*

Language: English | Subtitles: Arabic, French, Portuguese (European), Chinese (Simplified), Italian, Vietnamese, German, Russian, English, Hebrew, Spanish, Hindi, Japanese  
Difficulty: Unknown  
Duration: Medium  
Free

☐ [More info](#)

Click to unlike




**CS50's Introduction to Artificial Intelligence with Python**  
*Edx | offered by Harvard University*

Language: English | Subtitles: English  
Difficulty: Unknown  
Duration: Medium  
Free

☐ [More info](#)

Click to like



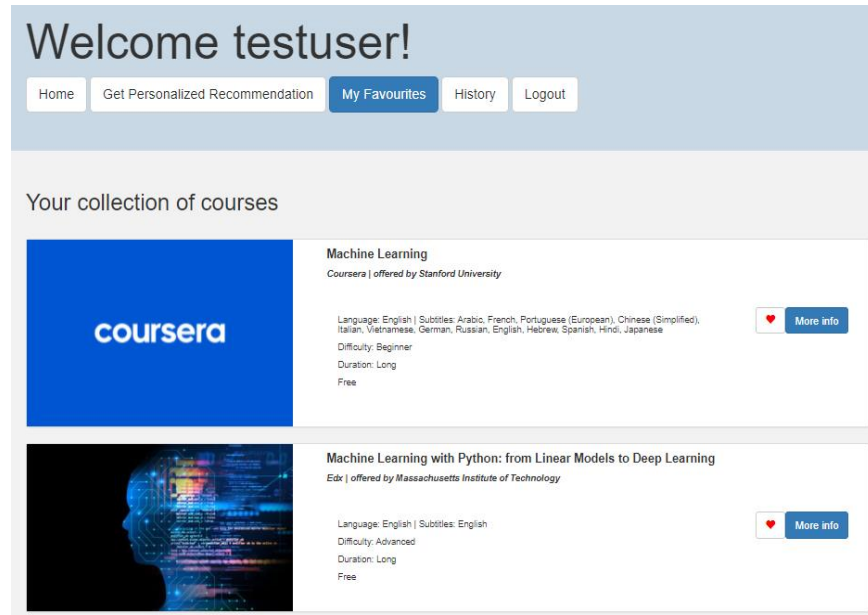
**Machine Learning with Python: from Linear Models to Deep Learning**  
*Edx | offered by Massachusetts Institute of Technology*

Language: English | Subtitles: English  
Difficulty: Intermediate  
Duration: Medium  
Free

☐ [More info](#)

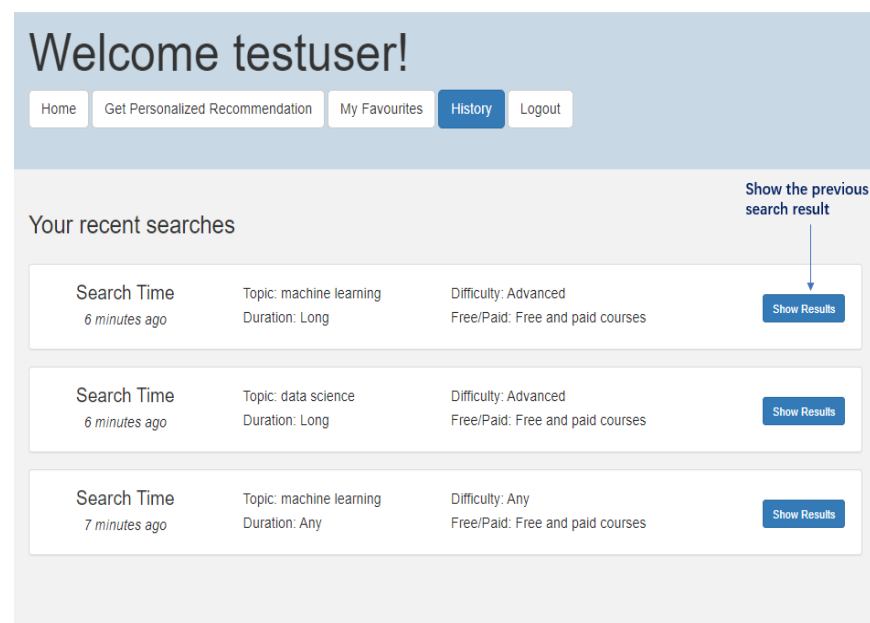
### User Liked Courses

Once the user is logged in, there will be a like button on every course information card. Courses that the user have clicked the like button from the home page or recommendation result page will appear here. As you can see the two courses with the like button clicked on in the previous recommendation result page appear in the “My Favourites” page.



### Search Histories

This page will show past historical searches performed by the user. Every information card contains the text field for describing the skill or course, duration of the course, difficulty level, and user's preference on free or paid courses, and time of that particular search. There will be a button for each historical search that will redirect the user to the result page that lists the recommendation results of that particular search.

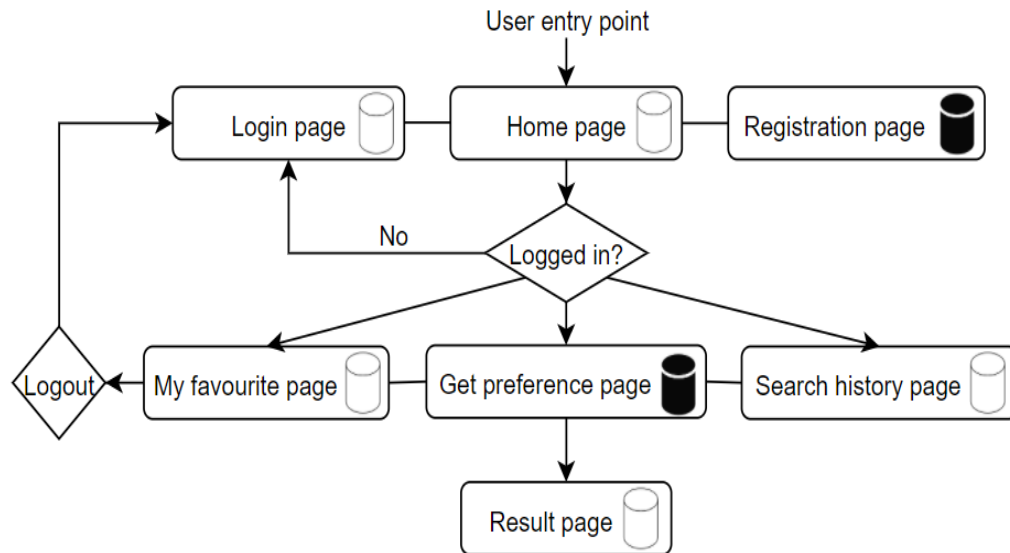




**Logout function.**

Provide an account logout function and return the user back to the home page.

The overall user interface structure of our web application is shown below. The solid-coloured container icons represent major read-write actions to the database, while the hollow container icons represent information retrieval from the database.

**6.3 SYSTEM BACKEND**

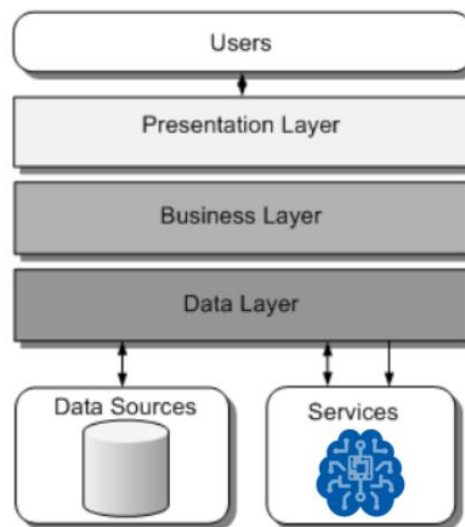
The backbones of our system backend include the Flask app module, the recommendation reasoning system for recommendation inference, and the database as our knowledge base. The Flask app module is responsible for managing the request and routing in between the system frontend and system backend modules. It is a python backend framework which is well adaptive to the frontend. It is a light-weight framework without database abstraction layers or many pre-defined functions and it allows the flexibility to plug-in libraries to create customized features.[9] In addition, we also utilize the Flask-SQLAlchemy library package which is built based on SQLAlchemy for interaction with the SQLite engine to handle the query sessions to our database.



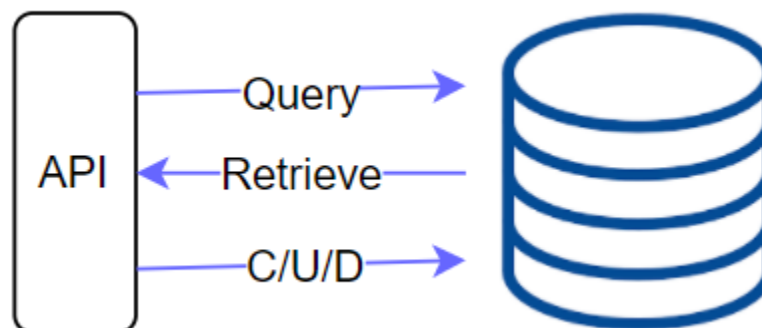
The recommendation reasoning system forms the core engine for the recommendation inference whenever a user send a course recommendation request from the frontend user interface and through the Flask routing. The details of the recommendation reasoning system are discussed in the previous section 5. The last module within the system backend is the knowledge base module which is built in the form of SQLite database. It consists of serval which are interlinked and serve as our knowledge representation. Details of the construction of the database are discussed in previous Section 3.

## 6.4 SYSTEM INTEGRATION

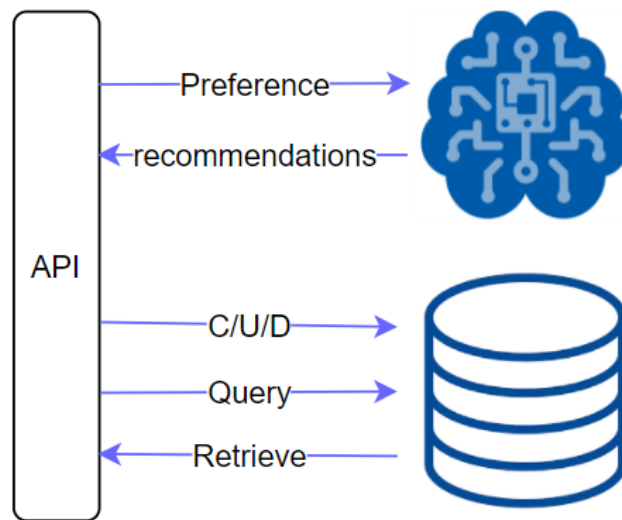
In this section, only how they interact with the previous layer specifically for our project will be introduced.



The database manipulation is famous with the term CRUD, which stands for create, read, update and delete. Majority of routes and request from the frontend into the Flask app are mostly interactive with the database. For example, index page, home page, login page, history page, my favourite page and result page. All the pages render information of the courses, and only retrieve the course information from database. Some routes have to perform create action, like account registration. On the other hand, the favourite page routing does not require interaction with the database, it only perform the create and delete action.



Aside from the majority of the routings and requests that deals with the database user information retrieval, the Flask app also manage the routing for recommendation inference which requires interaction with both the recommendation reasoning system as well as the database. The user inputs are sent to the recommendation reasoning system for recommendation inference and at the same are sent to the database for query search information register. With the list of recommended courses in the form of course index from the recommendation reasoning system, Flask will then redirect the request to the database again to retrieve the course information based on the list of course index. Finally, the course information will then be sent back to the system frontend through the Flask and will be rendered in the result page for presentation.



## 6.5 SYSTEM EVALUATION

### *Recommender system performance*

We evaluate our recommender system using mean average precision at K. We set K at 10 to keep the diversity and test the system and get feedback from our test users (example of 10 queries are evaluate by different people). Here are the details of their preferences and the result.

	Key words	Duration	Difficulty level	Paid/free
<b>Query1</b>	Machine learning	Long	Any	Any
<b>Query2</b>	Guitar fingerstyle	Medium	Beginner	Any
<b>Query3</b>	Swim	Medium	Beginner	Any
<b>Query4</b>	Artificial intelligence	Medium	Beginner	Free
<b>Query5</b>	Psychology	Any	Beginner	Any
<b>Query6</b>	Philosophy	Short	Beginner	Any
<b>Query7</b>	Cloud computing	Long	Intermediate	Any
<b>Query8</b>	Basketball	Short	Beginner	Free
<b>Query9</b>	Cooking	Medium	Intermediate	Any
<b>Query10</b>	Zoo	Any	Beginner	Free

	Course1	C2	C3	C4	C5	C6	C7	C8	C9	C10	p@K
Query1	1	1	0	1	1	1	1	1	0	1	0.8
Query2	1	0	1	0	0	1	1	0	0	1	0.5
Query3	1	0	1	0	1	1	1	1	0	0	0.6
Query4	1	0	1	0	1	0	1	1	0	0	0.5
Query5	1	0	1	1	0	0	1	0	1	0	0.5
Query6	0	0	0	1	1	0	1	0	0	0	0.3
Query7	0	0	1	1	1	1	0	0	0	0	0.4
Query8	1	1	1	0	1	1	1	1	0	1	0.8
Query9	1	0	0	1	0	1	1	1	1	1	0.7
Query10	1										

(\*course recommended is considered relevant by test user set 1, else 0)

**map@K = 0.61**

From the result, we can see that:

- The system achieves relatively high performance at the top 1 recommendation.
- Since our courses database is not considered that huge, the precision was affected by the search key word very much. For example, for philosophy and psychology may have fewer courses, hence, may get worse precision than keywords like python, machine learning. Also, for 'zoo' there is only one course is considered relevant by our system.

#### ***Other performances and feedbacks***

- Regarding the processing time of our application, we can get the recommend result instantly, and the loading time for our application is also considered fast from test users' feedback.
- The UI design is also satisfactory, 9 out of 10 test users feedbacks they will use our application in the future if it can summarize courses from more platforms.

## 7 CONCLUSION

---

We have successfully built a functioning MVP of online course recommender system. Throughout the whole project period, we have a better understanding of the course content and utilize them in our projects. For example, when we were choosing suitable recommendation method for our projects, we listed out what are the feature needed for different kinds of methods, and what is scenario that each method is used in, combining the data we have, considering our limitation, and then select the method that suit our use case. What's more, we can match different modules in our system to the knowledge acquisition, knowledge representation, and how to do knowledge modelling as well as proper reasoning using the recommendation module. The contents in the course are well connected in our mind.

However, we have encountered many challenges building the system. When preparing the data, we learn how to scrape useful information from different website. Considering the web structures are all different, there is no one-shot method to get all information simultaneously. And there are tens of thousands of courses, and it took us plenty of time to do data collection. For the recommendation system, we have tried out many methods in order to get better performance, as well as balancing between the response speed and the performance. When we built the web application, a lot of effort were made to improve the user experience, and also cutting of some unessential display and function for not confusing the user.

Nonetheless, the system produces satisfactory results, and the user feedbacks are good as well. We have done a good practice of project management too, keeping progressing throughout the project.

**Limitations and Future Improvements**

LIMITATIONS	IMPROVEMENTS
<b>Courses still not diverse enough.</b> As a minimal viable product, the data are only collected from three major online course platforms, Udemy, Coursera, and Edx.	<b>Exploring and include more online learning platforms.</b> In order to make the recommendation more diverse, future works can be exploring more platforms such as DataCamp, Udacity, or even MOOC website from other languages.
<b>Source websites' content are dynamically changing,</b> like putting new courses on, putting off courses for special reasons, and modifying course syllables, and different platforms have different website structures, resulting in high difficulty to scrap for this recommender system the courses database is static for now.	<b>Making it dynamic on the course content,</b> including periodically scrapping from the original sites and updating the feature maps such as the Tf-Idf Matrix and one-hot encoding categorical matrix.
<b>The Recommendation Reasoning System is by no means the perfect recommender</b> given a user input. As we can see from the recommendation evaluation, we are achieving only a mean average precision (map@K) of only around 0.6. This is because we are only using handcrafted features, ie: Tf-Idf as feature representation in the recommendation module. The handcrafted features performance depends on the data it was fit on and hence will ultimately affect the performance of the recommendation depending on which data it is trained on.	<b>To explore deep neural network (DNN) based models</b> for the feature extractions and recommendations such as LSTM or transformer-based NLP DNN methods. The DNN based models can learn the more generic feature representations of course title or description better and this will help to improve the recommendation further.
<b>Individual users' data on course rating is not available.</b> Our system only makes recommendations based on user input preferences, and we cannot make collaborate filtering recommendation of these courses for now because the lack of individual user data/ratings. There are individual user comments under each course for some of the platforms but in consideration of the time to scrap these information, content-based recommendation is promoted instead. The course rating in this system is the mean rating score obtained from the website.	<b>Making use of the user comments for every courses.</b> Future work on this aspect can be making use of the individual comments and applying NLP technique to get the individual ratings from the users and making use of CF recommendation in the system. Also, we can the user input preference and past searches or favourite courses to make a personalized recommendation, subsequently rendering a more personalized home page whenever the user login to their account.
<b>The system only host locally.</b> We have explored the online deployment option and finally find that platform like Heroku has a limit for database hosting. We also try to dockerize our application, but the built image is too large and we consider it is tedious for the user to do all the setups for it.	Explore more options for online hosting for our system.

## APPENDIX A – REFERENCES

---

- [1] Mordor Intelligence. (2019). MASSIVE OPEN ONLINE COURSE (MOOC) MARKET – GROWTH, TRENDS, AND FORECAST (2020 – 2025). <https://www.mordorintelligence.com/industry-reports/massive-open-online-course-mooc-market>
- [2] Shah, D. (2020, November 30). By The Numbers: MOOCs in 2020. Class Central. <https://www.classcentral.com/report/mooc-stats-2020/>
- [3] Coursera <https://www.coursera.org/>
- [4] Edx <https://www.edx.org/>
- [5] Udemy <https://www.udemy.com/>
- [6] Rapid Automatic Keyword Extraction, Stuart Rose, Dave Engel, Nick Cramer and Wendy Cowley [https://www.researchgate.net/publication/227988510\\_Automatic\\_Keyword\\_Extraction\\_from\\_Individual\\_Documents](https://www.researchgate.net/publication/227988510_Automatic_Keyword_Extraction_from_Individual_Documents)
- [7] Cosine Similarity [https://en.wikipedia.org/wiki/Cosine\\_similarity](https://en.wikipedia.org/wiki/Cosine_similarity)
- [8] Bootstrap Template <https://getbootstrap.com/>
- [9] En.wikipedia.org. 2021. *Flask (web framework)* - Wikipedia. [https://en.wikipedia.org/wiki/Flask\\_\(web\\_framework\)](https://en.wikipedia.org/wiki/Flask_(web_framework))



## APPENDIX B – USER MANUAL GUIDE

### Installation Guide

As the Online Course Recommender WebApp is mainly written in Python, in order to install and start using the application, Python is required to be installed on the user machine. To avoid the complications and version compatibility issues of Python modules, we recommend the user to create a new python or conda environment and to install and run the WebApp in the new environment.

### Pre-Requisite

Anaconda - <https://www.anaconda.com/>

Below are the installation commands following a Windows OS and Linux Ubuntu 20.04 OS using a command prompt or terminal.

### Installation Procedure

#### Step1: Create New Conda Environment

Windows OS	Linux Ubuntu 20.04
Type in command prompt: conda create --name py39_ocrs python=3.9 -y  <code>&gt; conda create --name py39_ocrs python=3.9 -y</code>	Type in terminal: conda create --name py39_ocrs python=3.9 -y  <code>\$ conda create --name py39_ocrs python=3.9</code>

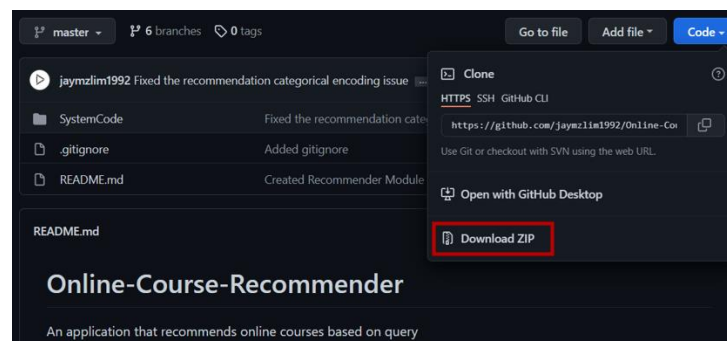
#### Step 2: Activate New Conda Environment

Windows OS	Linux Ubuntu 20.04
Type in command prompt: conda activate py39_ocrs  <code>&gt; conda activate py39_ocrs</code>	Type in terminal: conda activate py39_ocrs  <code>\$ conda activate py39_ocrs</code>

#### Step 3: Download and Clone from Github Repository

Github Repository: <https://github.com/jaymzlim1992/Online-Course-Recommender>

Manual Download:



**Step 4: Change Directory to the Clone Repository and Go into *SystemCode* folder**

Windows OS	Linux Ubuntu 20.04
Type in command prompt: cd <path to repo>\SystemCode <code>&gt; cd \path\to\repo\SystemCode</code>	Type in terminal: cd <path to repo>\SystemCode <code>\$ cd path/to/repo/SystemCode</code>

**Step 5: Install the Required Python Packages**

Windows OS	Linux Ubuntu 20.04
Type in command prompt: pip install -r requirements.txt <code>&gt; pip install -r requirements.txt</code>	Type in terminal: pip install -r requirements.txt <code>\$ pip install -r requirements.txt</code>

**Step 6: Install the NLTK Corpora**

Windows OS	Linux Ubuntu 20.04
Type in command prompt: python nltk_setup.py <code>&gt; python nltk_setup.py</code>	Type in terminal: python nltk_setup.py <code>\$ python nltk_setup.py</code>

**Setup will be completed after step 6!**

### Running the Application

Starting the WebApp is easy, user just need to navigate to the SystemCode folder in the Repository and run with python environment activated:

#### Activate Conda Environment

Windows OS	Linux Ubuntu 20.04
Type in command prompt: conda activate py39_ocrs <code>&gt; conda activate py39_ocrs</code>	Type in terminal: conda activate py39_ocrs <code>\$ conda activate py39_ocrs</code>

#### Change Directory to the Clone Repository and Go into *SystemCode* folder

Windows OS	Linux Ubuntu 20.04
Type in command prompt: cd <path to repo>\SystemCode <code>&gt; cd \path\to\repo\SystemCode</code>	Type in terminal: cd <path to repo>\SystemCode <code>\$ cd path/to/repo/SystemCode</code>

#### Run the Application

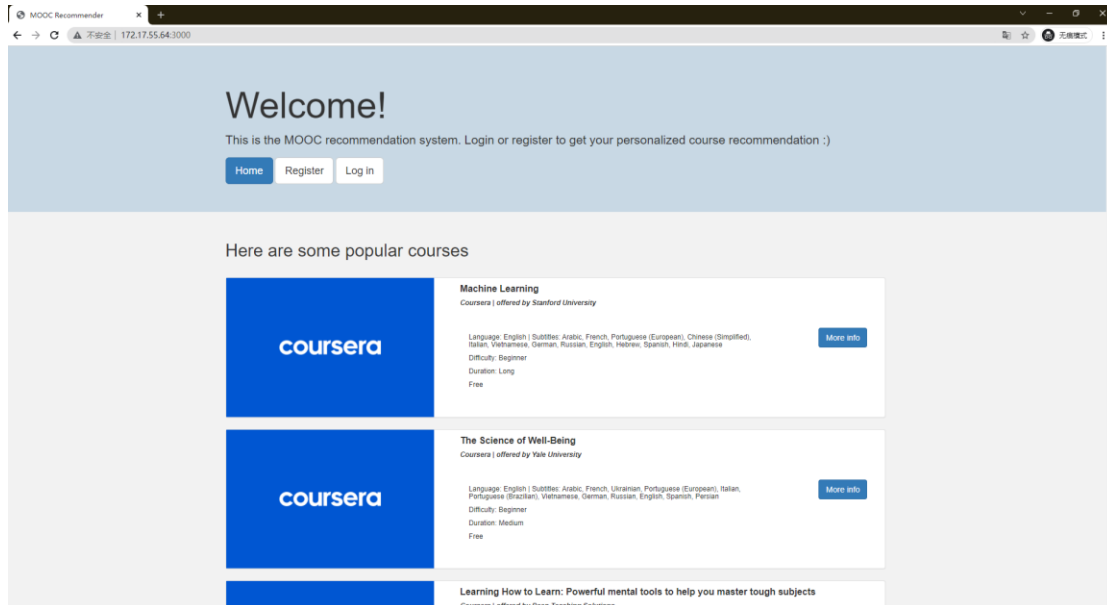
Windows OS	Linux Ubuntu 20.04
Type in command prompt: python run.py <code>&gt; python run.py</code>	Type in terminal: python run.py <code>\$ python run.py</code>

The application will launch in the user's default web browser and will be directed to the Home page.

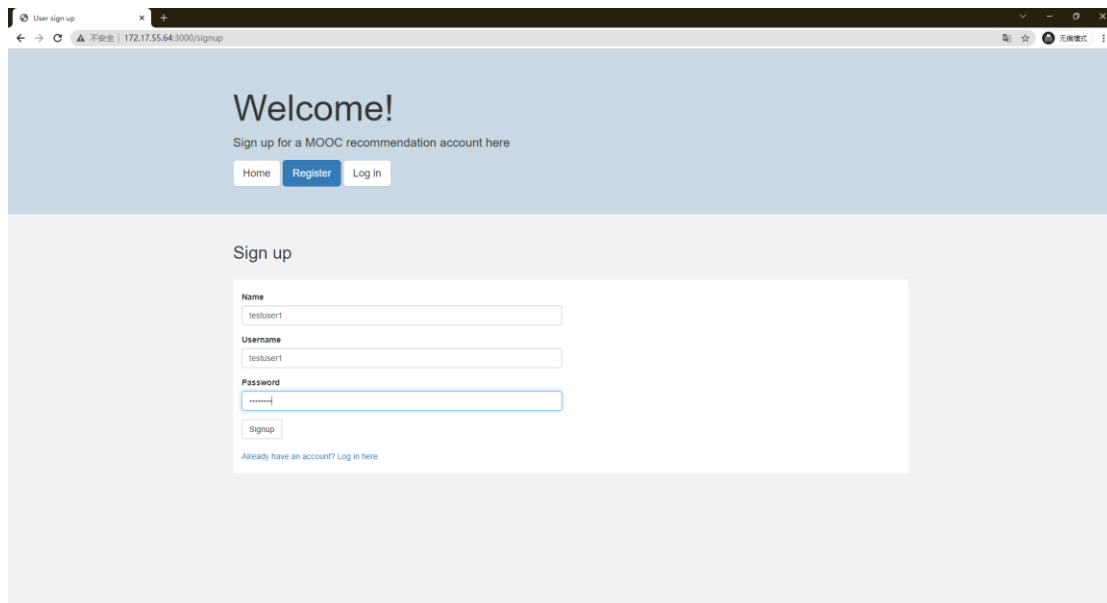
## User Manual

Here is a step-by-step navigation guide for our online course recommender system.

- After you have done all the installation and run the programme, you will be landed at our home page. You can glance through all the top-ranking courses here and if you are interested you can also click on more information of the course and navigate to its own platform's course info page.

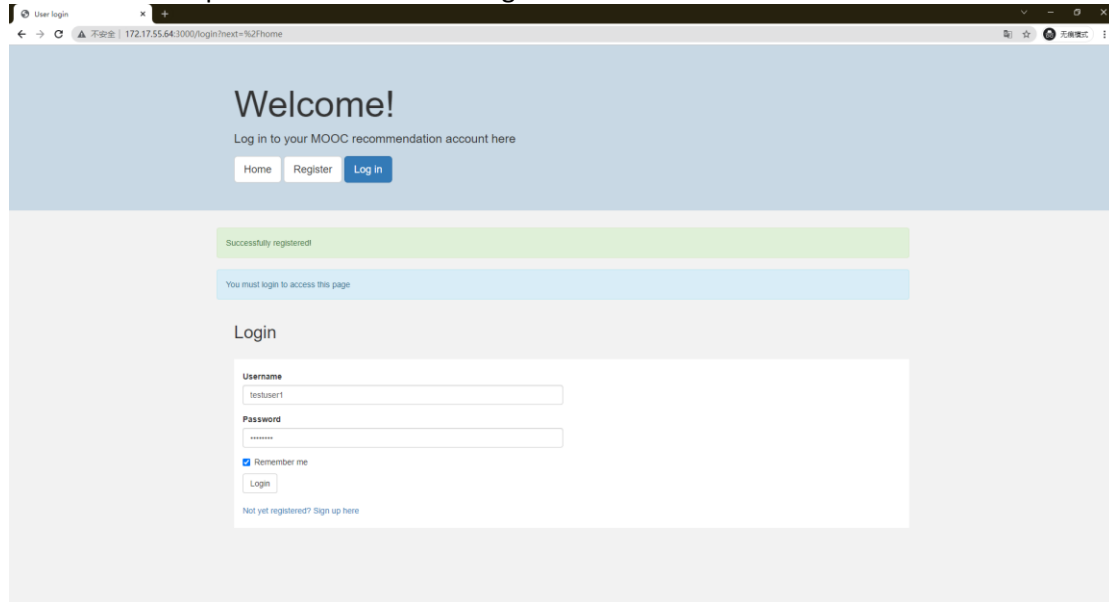


- You need to register for your own account. Click on 'Register' button beside the 'Home' button and you will be landed on the following page. Input your display name in the 'Name' field, as well as your username and set your password. After that, clicked on 'Signup'

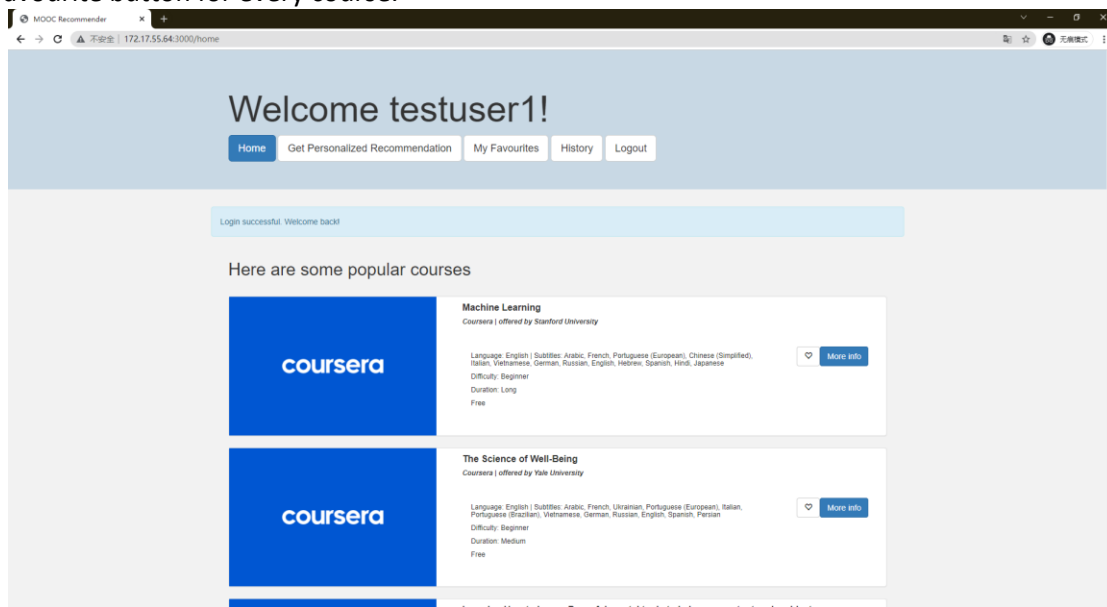


It will perform self-validation on this page too, checking any existence of the same username as your input username. If exist, error will prompt, and the webpage will ask you to choose another username.

- After you have successfully registered an account, you will be landed/redirect to the login page. Input your username and password and click on Login.



- After you have successfully logged in, you will be redirect to the home page again. There will be a difference compared to the un-logged-in home page, that is if you are logged in, you can see the favourite button for every course.



- Let's begin your personalized recommendation! Click on 'Get personalized Recommendation' button, and you will be landed on the survey page. Input your preferences and keywords, click on the 'get course' button, and you will be landed on the result page.

Personalized course recommender

Welcome testuser1!

Home Get Personalized Recommendation My Favourites History Logout

Let us know your preferences below

What would you like to learn?  
python

Duration\*

☐ No preference  
☐ Short (1 - 10 hours)  
☐ Medium (10 - 50 hours)  
☒ Long (>50 hours)

Difficulty level\*

☒ Any  
☐ Beginner  
☐ Intermediate  
☐ Advanced

Would you be open to paid courses?\*

☒ Yes. Show me both paid and free courses  
☐ No. Show me free courses only

Get courses

- Below is the recommendation result of your input preferences. You can scroll down to view more courses. The courses are ranked based on your preferences, so the most relevant course should be in the top few. You can also favourite the courses that you like for later review.

Recommendation results

Welcome testuser1!

Home Get Personalized Recommendation My Favourites History Logout

Got your preferences!

Here are the recommendations based on your preferences

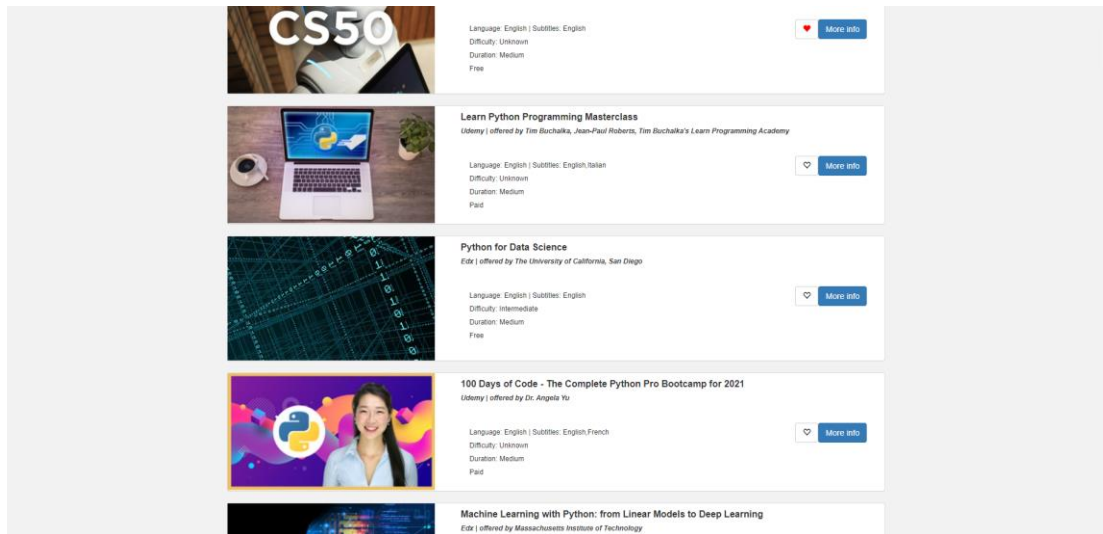
**Introduction to Computer Science and Programming Using Python**  
 Edx | offered by Massachusetts Institute of Technology

Language: English | Subtitles: English  
 Difficulty: Unknown  
 Duration: Medium  
 Free

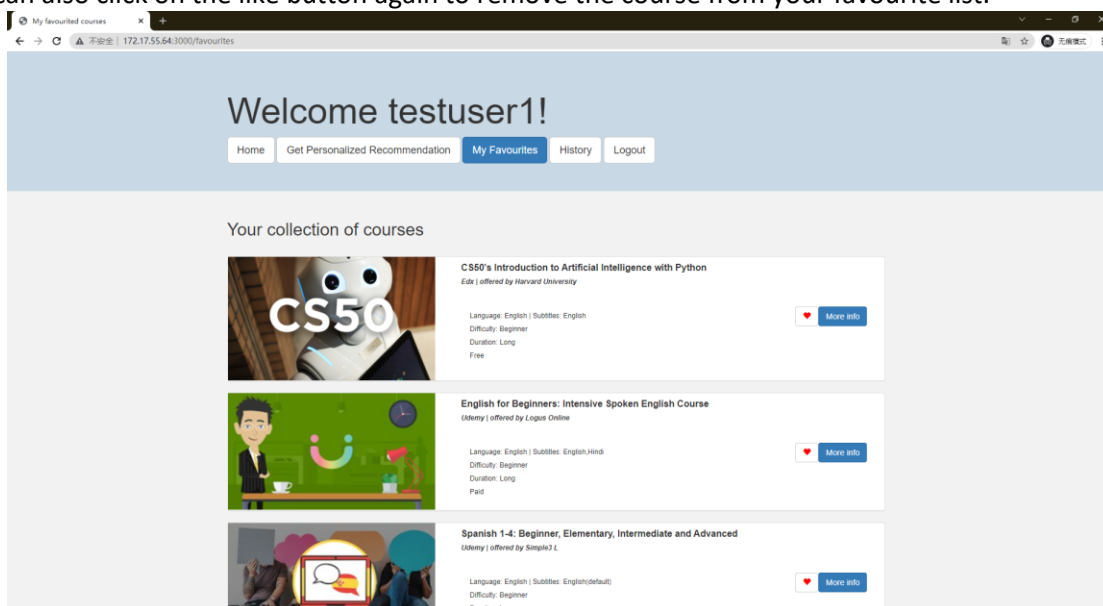
**CS50's Web Programming with Python and JavaScript**  
 Edx | offered by Harvard University

Language: English | Subtitles: English  
 Difficulty: Beginner  
 Duration: Medium  
 Free

**CS50's Introduction to Artificial Intelligence with Python**  
 Edx | offered by Harvard University

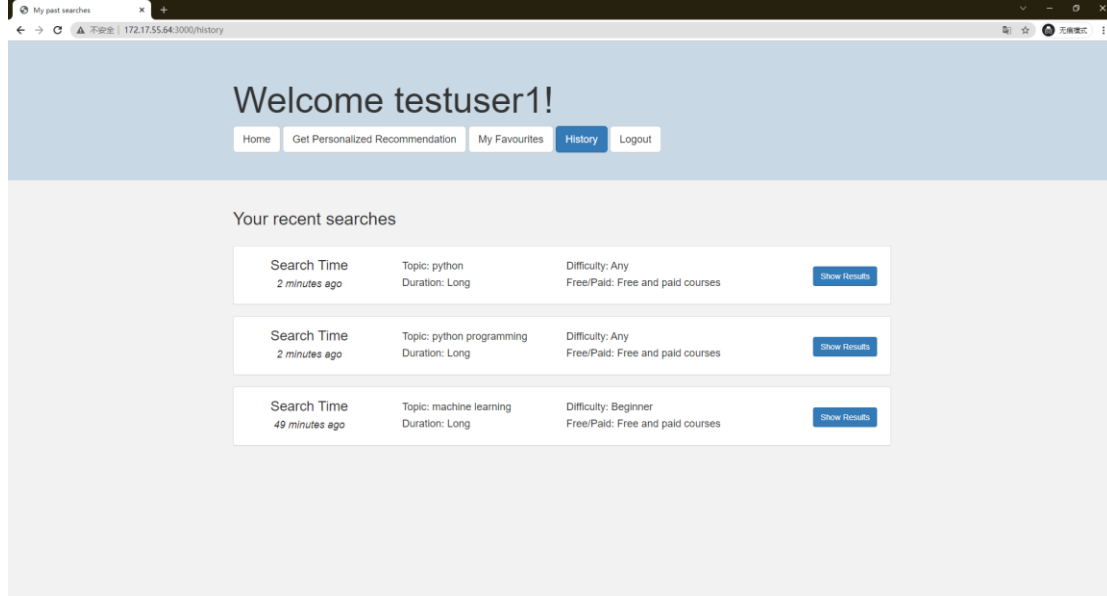


- The 'my favourite' page stores all the courses that you clicked like button for your user account. You can also click on the like button again to remove the course from your favourite list.

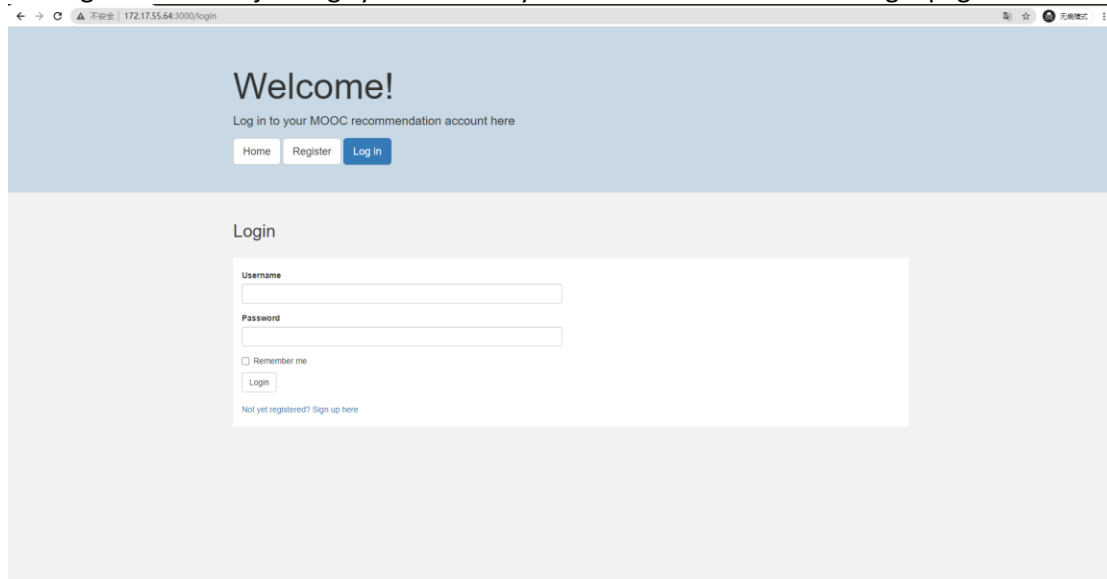




- The 'History' Page contains all the past queries and the query information for this particular user. You can also click on the 'show details' button to see this particular search result.



- The 'Logout' function just logs you out. And you will be redirected to the login page.



## APPENDIX C – PROPOSAL

### GRADUATE CERTIFICATE: Intelligent Reasoning Systems (IRS)

#### PRACTICE MODULE: Project Proposal

<b>Date of proposal:</b>  18 September 2021
<b>Project Title:</b>  Online Course Recommender System
<b>Sponsor/Client:</b> <i>(Name, Address, Telephone No. and Contact Name)</i>  Institute of Systems Science (ISS) at 25 Heng Mui Keng Terrace, Singapore NATIONAL UNIVERSITY OF SINGAPORE (NUS) Contact: Mr. GU ZHAN / Lecturer & Consultant Telephone No.: 65-6516 8021 Email: <a href="mailto:zhan.qu@nus.edu.sg">zhan.qu@nus.edu.sg</a>
<b>Background/Aims/Objectives:</b>  <p>In recent years, there has been a growing demand for online courses. This growth was also further accelerated due to the COVID-19 outbreak. In particular, the global Massive Open Online Course (MOOC) market had an estimated value of USD 6845.4 million in 2020. It is expected to increase to USD 18925.18 million by 2026, with an estimated Compound Annual Growth Rate (CAGR) of 18.13%, from 2021 to 2026 [1]. The number of MOOCs available has also grown steadily with the growth in demand. According to Class Central, it was estimated that from 2012 to 2020 there were a total of 16,300 MOOCs rolled out by 950 universities globally (excluding China), and each year the number of MOOCs rolled out had been increasing [2].</p> <p>With such large numbers of MOOCs available, users may find it difficult to navigate through all of them to find the course that best suits their needs. A good recommendation system will assist users to easily find suitable courses which match their specific requirements.</p> <p>In this project, we aim to develop a cross-platform recommendation system for online courses, especially MOOCs. The system will consolidate the information of available courses across multiple online course providers and then recommend each user the courses that best matches their needs.</p> <p>This helps users to increase their search scope and potentially discover courses that they might not have otherwise found. It also saves users time and the hassle of combing through multiple course provider sites to find a course that matches their needs. The system will become an one stop solution for users to search for courses on skills that they would like to learn through e-learning.</p>

**Requirements Overview:**

- Web-scraping skills
- Data Processing skills
- Database skills
- Knowledge on recommendation systems
- Research ability
- Programming ability
- System integration ability

**Resource Requirements (please list Hardware, Software and any other resources)**

Hardware proposed for consideration:

- Laptop or PC with sufficient RAM, disk space and computing power to install and run the below software

Software proposed for consideration:

- Reasoning systems, e.g. Recommendation
- Pretained machine learning models, e.g. NLP
- Robotic Process Automation, e.g. Selenium, Beautiful Soup
- Development environment, e.g. Anaconda, Python
- Web Development, e.g. Flask, Python
- Database Management System, e.g. SQLite

**Number of Learner Interns required: (Please specify their tasks if possible)**

A team of 3 project members is required. The team will work together in an agile manner and will take up development subtasks at each sprint according to their skills.

Tasks (which are defined during requirement gathering and analysis) will be broken down into subtasks and completed over sprints.

**Methods and Standards:**

Procedures	Objective	Key Activities
<b>Requirement Gathering and Analysis</b>	The team should meet up to scope the details of project and ensure the achievement of business objectives. Regular meetings should also be held to ensure continuous progress and to deliver the solution in an agile manner.	<ol style="list-style-type: none"> <li>1. Gather &amp; Analyze Requirements</li> <li>2. Define internal and External Design</li> <li>3. Prioritize &amp; Consolidate Requirements</li> <li>4. Establish Functional Baseline</li> </ol>
<b>Technical Construction</b>	To develop the source code in accordance to the design.	<ol style="list-style-type: none"> <li>1. Setup Development Environment</li> <li>2. Understand the System Context, Design</li> </ol>

	<ul style="list-style-type: none"> <li>To perform unit testing to ensure the quality before the components are integrated as a whole project</li> </ul>	<ol style="list-style-type: none"> <li>Perform Coding</li> <li>Conduct Unit Testing</li> </ol>
<b>Integration Testing and acceptance testing</b>	To ensure interface compatibility and confirm that the integrated system hardware and system software meets requirements and is ready for acceptance testing.	<ol style="list-style-type: none"> <li>Prepare System Test Specifications</li> <li>Prepare for Test Execution</li> <li>Conduct System Integration Testing</li> <li>Evaluate Testing</li> <li>Establish Product Baseline</li> </ol>
<b>Acceptance Testing</b>	<p>To perform evaluation of the system to ensure that the system meets the requirements.</p> <p>Since this is a self-initiated project, team members will find target users as the 'Customers' to conduct acceptance testing.</p>	<ol style="list-style-type: none"> <li>Plan for Acceptance Testing</li> <li>Conduct Training for Acceptance Testing</li> <li>Prepare for Acceptance Test Execution</li> <li>Customer Evaluates Testing</li> <li>Obtain Customer Acceptance Sign-off</li> </ol>
<b>Delivery</b>	To package the system into a runnable program and upload into Github	<ol style="list-style-type: none"> <li>Software must be packed by following ISS's standard</li> <li>Deployment guideline must be provided in ISS production (ISS standalone server) format</li> </ol>

## Team Formation &amp; Registration

Team Name: <b>Recommender System Team</b>
Project Title (repeated): <b>Online Course Recommender System</b>
System Name (if decided): <b>Online Course Recommender System</b>
Team Member 1 Name: <b>Lim Jun Ming</b>
Team Member 1 Matriculation Number: <b>A0231523U</b>
Team Member 1 Contact (Mobile/Email): <b>Mobile: 8318 9918</b> <b>Email: <a href="mailto:e0703555@u.nus.edu">e0703555@u.nus.edu</a> / <a href="mailto:jmlim1992@gmail.com">jmlim1992@gmail.com</a></b>
Team Member 2 Name: <b>Sarah Elita Shi Yuan Wong</b>
Team Member 2 Matriculation Number: <b>A0231507N</b>
Team Member 2 Contact (Mobile/Email): <b>Mobile: 8877 8955</b> <b>Email: <a href="mailto:e0703539@u.nus.edu">e0703539@u.nus.edu</a> / <a href="mailto:Sarahwongsy09@gmail.com">Sarahwongsy09@gmail.com</a></b>
Team Member 3 Name: <b>Zhang Yunduo</b>
Team Member 3 Matriculation Number: <b>A0231349H</b>
Team Member 3 Contact (Mobile/Email): <b>Mobile: 98916368</b> <b>Email: <a href="mailto:e0703381@u.nus.edu">e0703381@u.nus.edu</a> / <a href="mailto:zhayunduo@hotmail.com">zhayunduo@hotmail.com</a></b>

## APPENDIX D – MAPPING OF SYSTEM FUNCTIONALITIES TO COURSES

---

Modular Courses	System Functionalities / Techniques
<b>Machine Reasoning (MR)</b>	Knowledge Modelling Rule Based System
<b>Reasoning System (RS)</b>	Content-Based Filtering Recommendations
<b>Cognitive System (CGS)</b>	Text Preprocessing NLP Features – TfIdf

## APPENDIX E – INDIVIDUAL MEMBER REPORT

Team Member 1	Lim Jun Ming
Matriculation ID	A0231523U

### *Personal Contribution & Learning Journey*

I was heavily involved in the idea generation and structuring of our project. I came up with the initial idea for this project on Online Course Recommender System. I helped the team in creating the initial system architecture for our project. After some market research conducted by everyone in the team, we selected the three online course platforms to build our knowledge model. I am involved with the data mining and data cleaning and preparation of the course information from Edx.com. After we have the course data ready and aligned in consistent format, I performed the knowledge discovery and selected useful information and prepared them for knowledge base construction.

I was tasked to design and build the recommendation reasoning system. This was an interesting learning journey as I explored various methods and architecture of building a recommendation system. At the same time, I would need to ensure that the methods and design used is applicable and effective for our business problem. I started with simple feature extraction methods and recommendation algorithm. However, the initial version of the recommendation system was not giving any good course recommendation. After a few iterations of testing and adjustments, I came out with the idea of the customized content-based filtering algorithm where there is a customized ranking optimization step within the module. Working together with other processes within the recommendation system, it is able to infer a list of meaningful course recommendations.

After the final recommendation algorithm was selected, I built and integrated the recommendation reasoning system to the Flask app and system backend. I have also helped to review the system frontend and made adjustment to improve the flow and presentation of the user interface such as the webpage colors, buttons and presentation consistency across the frontend design. Below is a summary of the tasks that I was involved in this project:

#### Pre-development:

- Project ideation
- Proposed system architecture
- Research on feasibility of system design

#### Design and Development:

- Knowledge acquisition – Edx.com web-scraping and data preparation
- Knowledge discovery – Feature selection and data pre-processing
- Recommendation Reasoning System – Design and development of the reasoning system as well as exploring various methods and approaches for recommendation algorithms.
- System frontend – Review and made adjustment to user interface to improve flow and presentation.

#### Post-development:

- Project report writing
- Technical video creation and explanation

### ***Learning Outcome***

Through this project, I have learnt so many things about development of an intelligent system especially on the recommendation reasoning system design. I have learnt how to perform web-scraping on websites and understood the challenges of data mining due to the dynamic environment of the internet webpage ecosystem. I have also learnt some knowledge discovery methods in selecting and extracting useful information which can then later be used to build useful knowledge base. In addition, I have learnt the process of integrating various module within a system such as connecting the Flask app to the database, frontend as well as request and response routing for recommendation inference with the reasoning system. I had great time understanding and working with my teammates on system integration.

As I was tasked for designing and developing the recommendation reasoning system, I had the opportunity to learn and experiment different design of the reasoning system. This includes selecting useful text pre-processing for the text-based data, learning and applying NLP keyword extraction method RAKE as well as NLP feature extraction using Tf-Idf vectorization. On recommendation algorithm, I had explored and tested various methods and finally came up with the idea of modifying the content-based filtering recommendation algorithm to include a customized ranking optimization method. This had successfully improved the course recommendations of our system.

### ***Knowledge and Skill Application***

I believe that the knowledge and skill I have learnt from this project will be very useful for any system design problems in the future. I have learnt how to select and apply the knowledge modelling skills correctly given a problem statement and system design objective. I can adopt and scale a similar system design and structure for a web-based application. I can also apply the web scrapping skills to any other project that requires data mining from the internet webpage. I would also know how to perform system integration between database, system frontend and system backend which are the core backbones of any system design.

The knowledge on recommendation reasoning system that I have learnt will definitely be useful in designing any other recommendation system in the future. I understand the general process flow of a recommendation system and know how to modify them in order to adapt to the business problem at hand. In addition, the NLP feature extraction and processing methods that I have learnt will also help to strengthen my skills and techniques in NLP related tasks.



Team Member 2	Sarah Elita Shi Yuan Wong
Matriculation ID	A0231507N

***Personal contribution to group project***

During ideation stage, my groupmates and I brainstormed as a group and came up with some ideas for recommendation systems. After much discussion and consideration of the practice module requirements, we decided to make a recommendation system for online courses. We started with drawing the overall architecture to determine the individual components that would be required in the final working system.

During development stage, we started with web-scraping. Each member did the web-scraping for 1 online course platform, since we figured that every website structure would be different. So, I did the web-scraping for Coursera. This step took us some time to complete as it was our first time doing web scraping. After completing the web-scraping, we split up the tasks to have faster progress. I looked into Streamlit as a potential application framework but it did not seem to be suitable for this project, so we later decided to go with Flask web application framework instead. I mainly worked on building the web application frontend together with my teammate. I also worked on the system database design, and data cleaning and transformation for Coursera. We also had online meetings around once a week to share our progress, difficulties and to decide on the finer details of the components we were working on individually.

During the report-writing stage, we all worked on writing the report and the two videos by dividing and combining our work.

***What learnt is most useful for you***

Through this project, I have gained valuable hands-on experience with web scraping, database design and using Flask framework for web applications. Through our discussions and knowledge sharing, I have also gained a deeper understanding of how to design a recommender system and the tools and techniques that may be required to implement it.

Personally, I think that web-scraping and database implementation will be the most useful for me in the future as there are many potential applications for those skills.

***How you can apply the knowledge and skills in other situations or your workplaces***

When implementing a new intelligent reasoning system, a database is usually required to store the knowledge base. In this project, we have designed and implemented an SQLite relational database to store our system's knowledge base. In the system, have also used various SQL commands for database operations. This has given me more confidence to handle databases in future, whether it is in the workplace or in other projects. I believe that relational databases are quite commonly used, so this skill will definitely come in handy.

On the other hand, web-scraping is a useful technique for data mining (knowledge discovery). As an automation of knowledge discovery, it has many potential uses, such as price monitoring and news monitoring.

Team Member 3	Zhang Yunduo
Matriculation ID	A0231349H

***Personal Contribution***

I have involved in all parts except for the recommendation algorithm coding, including idea generation, Udemy courses information scrapping, data cleaning and preparation, database building, flask web application building. Nonetheless I have involved in the discussion of what recommendation method we should use, and help debugging in the final system. I put most effort Udemy courses data scrapping and flask web application building. There are over 70000 courses on the Udemy platform which is most of our system's courses come from. It does take me a lot of time to scrap and clean and prepare the useful data for our system. As a green hand of web application building, I learned flask framework along the journey of our project. This Includes how to make the favorite function work, how to encrypt the use password and register/login interface and interact with database properly. Not to mention the documentation of the project, such as report writing, presentation video making, as well as our business video making. For report and technical video, I am responsible for the part that I have done for the project.

***Learning outcome***

I have learned a lot throughout the whole project process. Such as web application design and building, web scrapping as well as the raw web data cleaning and preparation, project management, and strengthen my Python programming skills. I have a clearer understanding of our Intelligent Reasoning System course content as well. For example, I can map all the modules in our own intelligent recommendation system to the concepts that we learn in the class such as knowledge representation, knowledge extraction, knowledge modeling and so on. I am not aware of such structure of all the intelligent system around us until I learn and build an intelligent system myself. It also a great experience to work with my teammates Jun Ming and Sarah whom I learn a lot from.

***Knowledge and Skill Application***

Specifically, from the recommendation module I can apply the NLP techniques to future text processing tasks, the content base recommendation I can apply it future recommendation tasks. For the web application building, I can apply it to any system that I want to build a smart user interface to showcase my system to the user and the web scrapping part it strengthens my data analytic skills.

In the future, especially the coming internship, I want to apply what I have learnt from this course to build intelligent reasoning system as well as deploy these systems. From my perspective, other certificates teach me how to utilize the machine learning or AI techniques in specific area, while the intelligent reasoning system certificate teaches me what is an intelligent system and how to build one and linking my knowledge from all the other certificates. I would like to say that we should think like a machine to build an intelligent system but it serves like a human.