

# **COMP9727: Recommender Systems - Project Design**

**Project title:** A hybrid recommender system for MOOC platforms

**Name: Tianyu Zhang**

**zID: z5516759**

**Date: 7.5**

## **1. Scope**

### **1.1 Client & Domain**

Users:

Primary: Lifelong learners (age 18-35) needing career-related upskilling

Secondary: Educators discovering popular course topics

Why Recommend this system?

72% of users leave sites due to choice overload, spending an average of 42 minutes browsing before registering for a course, edX 2023 report states. This recommender system aims to reduce decision fatigue.

### **1.2 Interface & Interaction**

Recommendation Presentation:

Number of items: 5 courses per recommendation (top-N recommendation)

Interface type: Web-based responsive design

Mockup Description:

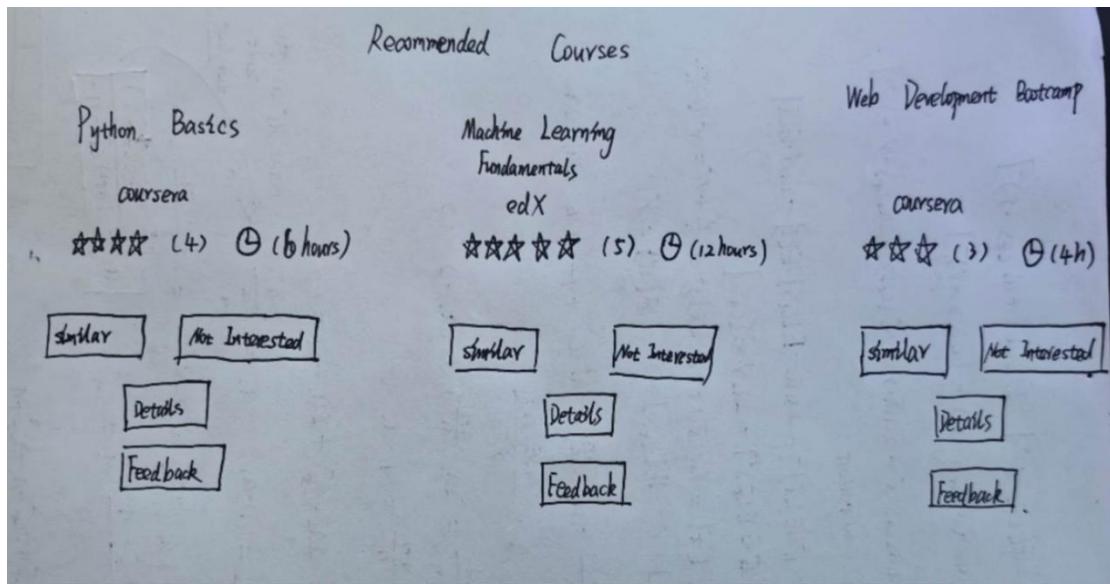


Figure1:MOOC Platform Recommendation Interface Wireframe

#### A.Layout:

Three-column responsive design (left: user profile, center: recommendations, right: course details)

#### B.Core Components:

Recommendation Cards (center): Display 5 courses with titles, and ratings (★)

Feedback Buttons (card footer): Thumbs up/down for explicit ratings

#### C.Interaction Hints:

Hover on cards results in implicit feedback tracking

Clicking "Details" opens right panel with syllabus

New users see highlighted popular courses (cold-start handling)

#### Interaction Flow:

a.User rates 3 courses during onboarding (cold-start mitigation)

b.System refreshes recommendations weekly through:

Explicit feedback (button clicks)

Implicit signals (dwell time, video completion)

### 1.3 Dynamic Updates

New User:Content-based + Popularity(Real-time)

New Course:TF-IDF similarity to existing (Weekly)

Active User:Collaborative filtering(Daily)

### 1.4 Business Model

Revenue Streams:

- a.Affiliate fees (5-10% per enrollment via platform APIs)
- b.Premium analytics for educators (\$99/season)

### 1.5 Timetable

W1:Data/EDA    W2:Cleaning/Features    W3:Content-model    W4:Hybrid

W5:Offline-eval    W6:UI/Study-prep    W7:User-testing    W8:Final-report

## 2. Dataset

### 2.1 Primary Sources

a. HarvardX-MITx Person-Course Dataset (2013 Academic Year):

Scale: 160K enrollments (single academic year)

Key Fields:

'user\_id', 'course\_id', 'final\_grade': For collaborative filtering

'nevents', 'ndays\_act': For implicit feedback

**Citation:**HarvardX-MITx. (2014). *HarvardX-MITx Person-Course Academic Year 2013 De-Identified dataset* [Data set]. Harvard Dataverse.

<https://doi.org/10.7910/DVN/26147>

b. Udemy Courses Dataset (2023 Version):

Scale: 13K courses

Key Fields:

'title', 'description': For content-based filtering

'avg\_rating', 'num\_subscribers': For popularity signals

**Citation:** Kaggle. (2023). Udemy Courses Dataset (2023 Version) [Data file]. Retrieved from <https://www.kaggle.com/datasets/andrewmvd/udemy-courses>

## 2.2 Data Limitations & Solutions

a.No cross-platform user IDs

Impact: Cannot track user behavior across multiple MOOC platforms

Solution: Leverage course content features as a bridge to make cross-platform inferences

b.Sparse rating data (92% of courses have <50 ratings)

Impact: Introduces popularity bias in recommendations

Solution: Apply Bayesian smoothing (add-1 regularization) to balance against the distribution

c.Demographic skew (60% North American users)

Impact: Introduces regional bias in suggestions

Solution: Use stratified sampling across geographic regions to ensure balanced coverage

d. Additional recommendation: Introduce temporal analysis to overcome potential recency bias in course ratings Use cold-start mitigation strategies on new courses with limited interaction data Introduce diversity measures to evaluation to prevent filter bubble effects

## 3. Methods

### 3.1 Hybrid Architecture

Phase 1 (Cold Start):

Content-based using TF-IDF

```
from sklearn.feature_extraction.text import TfidfVectorizer  
  
tfidf = TfidfVectorizer(stop_words='english')  
  
course_vectors = tfidf.fit_transform(course_descriptions)
```

Phase 2 (Warm Start):

Collaborative filtering with Surprise

```
from surprise import KNNBasic  
  
sim_options = {'name': 'cosine', 'user_based': False}  
  
algo = KNNBasic(sim_options=sim_options)
```

## 3.2 Technical Stack

Backend: Python (Flask)

Libraries: scikit-learn, Surprise, NLTK

Infrastructure: AWS EC2 (batch) + Lambda (real-time)

## 4. Evaluation

### 4.1 Offline Metrics

Metric	Target	Tool
Precision@5	$TP/(TP+FP) \geq 0.5$	Sklearn.metrics.precision_score
Coverage	$recs/total\_courses \geq 60\%$	Custom coverage () function
Novelty	$-\sum p(i)\log p(i) \geq 0.7$	Surprise.prediction_algorithms

### 4.2 User Study

Participants: 20 users(15students+5workers)

Method: 3 recommendation cycles + short survey

Success Criteria:

- A.  $\geq 80\%$  satisfaction
- B.  $\leq 15\%$  negative feedback

### **4.3 Computation requirement**

Hardware:

Minimum: 4-core CPU, 8GB RAM

Recommended: 8-core CPU, 16GB RAM (for more rapid model training)

Runtime Requirements:

Daily model updates: ~2 hours run time

Memory: 4GB re for full dataset processing