DEVELOPMENT OF A PERSONALIZED YOUTUBE-BASED SKILL-PATH RECOMMENDER SYSTEM USING MACHINE LEARNING

BY

CHINONYELUM EMMANUELA OBIEZE
GOU/U22/CSC/823

DEPARTMENT OF COMPUTER SCIENCE
FACULTY OF COMPUTING AND INFORMATION SCIENCE
GODFREY OKOYE UNIVERSITY, ENUGU STATE

IN PARTIAL FULFILLMENT OF THE AWARD OF BACHELOR OF SCIENCE (B.SC), DEGREE IN COMPUTER SCIENCE

SUPERVISOR: DR. FRANKLIN OKEBANAMA

JULY 2026

APPROVAL PAGE

This research titled "DEVELOPMENT OF A PERSONALIZED YOUTUBE-BASED SKILL-PATH RECOMMENDER SYSTEM USING MACHINE LEARNING," has been assessed and approved by the Department of Computer Science Committees in the Faculty of Computing and Information Science, Godfrey Okoye University, Enugu, Nigeria.

Supervisor	Signature	Date
Dr. Frank Okebanama	.....	.....
External Examiner	Signature	Date
	.....	.....
HoD, Department of Computer Science	Signature	Date
Dr. Frank Okebanama	.....	.....
Dean, Faculty of Computing & Information Technology	Signature	Date
Prof. I. A. Ajah	.....	.....

CERTIFICATE

I certify that I completed the research included therein and that it was primarily based on my observation and investigation. Consequently, I will fully accept the University's decision if I am found to have cheated on the assignment.

STUDENT'S NAME
CHINONYELUM OBIEZE

REG. NUMBER
GOU/U22/CSC/823

DEDICATION

This project is dedicated to God almighty for His mercies and grace towards me and to my elder brother Mr. Ikenna Paschal Obieze.

ACKNOWLEDGMENTS

I wish to acknowledge the Head of Department Dr. Franklin Okebanama who doubles as my supervisor for his unwavering support and guidance towards completing this project and to Mr. Camillus Njoku for his guidance.

I also wish to acknowledge my mother: Mrs. Nkiruka Obieze; my siblings: Chiamaka, Chinaza, Ikenna, Chinwendu, Arinze, Uchenna, and Sopuluchi; my niece and nephews: Somtochukwu Obi, Obinna Obi, Chimdiebube Obi, Daberechi Obi, and Chimdiebube Obieze.

Abstract

The rapid growth of YouTube as a learning platform has improved access to educational content but the lack of a pedagogically-structured path for learners introduces a very big challenge. YouTube's default recommendation system is primarily engagement-driven, resulting in fragmented learning experiences as it only recommends video content based on high user engagement. This study presents YouLearn: a personalized YouTube-based skill-path recommender system designed to generate structured learning pathways aligned with user goals. The system employs a hybrid approach combining content-based filtering, a Logistic Regression-based skill difficulty classifier model, a collaborative filtering model using Singular Value Decomposition (SVD) and a concept dependency graph that sequences YouTube videos in three categories of "beginner", "intermediate", and "advanced." Video data was collected through the YouTube Data API v3 and preprocessed using Text Frequency-Inverse Document Frequency (TF-IDF) techniques. Due to the absence of YouTube real user interaction data, a Coursera dataset containing Coursera_courses.csv and Coursera_eviews.csv was used as a proxy for training the personalization model. Evaluation using F1-score, RMSE, MAE, Precision@K, and Recall@K demonstrates the system's effectiveness in delivering structured and relevant recommendations for efficient learning.

Table of Contents

List of Figures

Chapter One
Introduction

1.0 Background to the Study

Technology has played a crucial role in the evolution of education. Education, which initially began as oral teachings, has quickly transformed from the use of writing tools like pen and chalk to audio-visual media like film, television and radio, to teaching machines and currently, interactive digital learning environments.

Since the advent of personal computers in the late 1800s and early 1900s, distance learning has become more and more accepted. Distance learning, a form of education in which the main elements include physical separation of teachers and students during instruction and the use of various technologies to facilitate student-teacher and student-student communication (Berg et al, 2025). The first instance of distance learning was in the 19th century when correspondence courses, a form of distance learning where students received printed materials and sent back assignments through post.

Technological innovations such as artificial intelligence, learning analytics, and digital learning systems have significantly transformed and enhanced modern learning environments (Cheung et al., 2021). The development of Massive Open Online Course (MOOC) systems like Coursera, MIT OpenCourseWare, interactive learning environments like Codecademy, search engines like Google, Bing, and YouTube, has significantly imparted online learning and enabled global access to structured education online.

Today, learning anything is possible online, from digital skills to culinary skills to actual degrees in different fields ranging from health to information and communication technology to engineering. But the cost of online learning has significantly increased, ranging from a few hundreds to hundreds of thousands of naira. This significant increase has locked underprivileged learners out of the digital classroom, making online learning inaccessible to them. But, one platform stood out and maintained a service-free charge, though utilizing advert placements as a means of income generation. That platform is YouTube.

YouTube is a free video search engine owned by Google. According to Ramatika et al, 2021, Youtube can be used as a learning medium that helps students learn independently

YouTube is open to creators irrespective of background to create content across diverse topics including but not limited to crafts, cuisine, fashion, education, etc. The free accessibility of YouTube to all creators introduced multiple complexities. The first of which is that it introduces information overload. On average, YouTube records over 20 million video uploads every single day (YouTube Press, 2026). These videos do not follow a pedagogically-structured learning path, making it difficult for learners to follow. Without a structured path to follow, learners jump from one video to another, learning the same thing or learning in a way that leaves gaps in their memory.

The second complexity is finding content in your exact field using your preferred tool or industry. For instance, while trying to learn data analysis using Power BI and MySQL, YouTube might return videos for Data Analysis using Looker Studio or PostgreSQL without paying attention to user-specific tools.

Without a system that can guide learners towards the most relevant content for their industry, skill, and project, they spend unnecessary time searching for relevant videos and become overwhelmed from information overload leading to reduced learning efficiency.

In response to these challenges, there is a need for a personalized, YouTube-based skill-path recommender system that can predict videos personalized based on users' activities; classify videos into three major categories of beginner, intermediate, and advanced; sequence videos into topics and finally recommend exact match recommenders. Integrating such a system within YouTube or other e-learning platforms can significantly enhance learner engagement and improve learning outcomes.

This study focuses on the development of a personalized skill learning and resource recommender system intended to support learners in identifying the most suitable resources on YouTube for their study.

1.1 Statement of the Problem

The problems this system wishes to solve are:

a. Lack of Structured Learning on YouTube: YouTube contains thousands of educational videos that are not arranged in a pedagogical sequence forcing learners to consume content randomly, resulting in knowledge gaps and poor skill progression. (Joy et al, 2021)
b. Difficulty in Identifying High-Quality and Relevant Videos: Learners are often overwhelmed manually filtering through millions of videos searching for the right video that fits their industry, skill, tools, and knowledge level. The system addresses this by automatically evaluating, ranking, and selecting the most suitable videos for each stage of learning.
c. No Personalization Based on Learner Intent: YouTube does not distinguish why a learner wants to learn a skill leading to mismatched recommendations. The system will introduce intent-aware personalization to match the learning path to the user's specific goal.
d. Information Overload for Beginners: The system will simplify learning by giving beginners a clear starting point and guiding them step-by-step.
e. Inconsistent Learning Quality Across Channels: Different channels teach differently and at different depths. A learner may unknowingly mix beginner content with advanced content, leading to confusion. The system normalizes this by curating content to ensure pedagogical progression and steady difficulty levels.
f. Time Wasted Searching Instead of Learning: A significant amount of learning time is lost searching, curating, comparing and evaluating videos. The system reduces this friction by automating the discovery and curation process, enabling learners to focus on actual learning.
g. Absence of Guided Skill Mastery: YouTube cannot assess how much a learner has progressed or what they should learn next. The system fills this gap through progress tracking and adaptive recommendations.

1.2 Aim of the Study

The aim of this study is to develop a personalized YouTube-based skill-path recommender system using machine learning.

1.3 Objectives of the Study

To achieve the stated aim, the following specific objectives will be pursued:

1. Identify the challenges associated with learning a new skill on YouTube.
2. Design a smart approach to YouTube personalized learning.
3. Develop the approach into real models that personalizes learning systems.
4. Test and evaluate the model to determine its accuracy.

1.4 Significance of the Study

The significance of the study which is about the development of a personalized YouTube-based skill-path recommender system, will have the following benefits:

a. Personalized Recommendations: This system will be aware of the user's preferences about the channels they like. The system will curate videos to match these user preferences.
b. Classified Recommendations: The recommended videos will be classified into three broad categories of "Beginner", "Intermediate", and "Advanced".
c. Sequenced Recommendations: This system will be able to sequence recommended videos into topics that are structured in a pedagogically-structured manner
d. Time and Effort Efficiency: This system will save the time learners use to manually filter and search for videos by providing a playlist that prioritizes high-quality videos with useful content.
e. Enhanced Learning Effectiveness: The system will minimize cognitive overload and promote deep understanding and mastery because learning follows a sequential order.
f. Progress Tracking: Learners can track watched videos and set clear milestones towards achieving mastery in a skill.

g. Research Tool: This system can also serve as a research tool for students and developers who are interested in recommender systems, Massive Open Online Course (MOOC) System and YouTube.

1.5 Scope of the Study

The study covers the design and implementation of YouLearn: a personalized YouTube-based skill-path recommender system for YouTube. It covers the processes of user registration and authentication, interface design, model development and model evaluation. The system only offers integration with YouTube and not other MOOC platforms. This study is also limited to the video learning materials available on YouTube and the recommendations are limited to keywords placed in the video title.

1.6 Limitations of the Study

1. YouTube Data API v3: YouTube Data API v3 will be used to collect YouTube videos in real-time from YouTube. This forms the limitations:
    a. Data Access Limitation: YouTube Data API v3 provides access to video and channel metadata like channel name, channel id, etc. But, the API limits how much data one can access, especially detailed analytics such as watch time or audience retention. This reduces the system's ability to accurately judge video quality and build highly optimized learning paths. As a result, users may receive learning sequences that are structured but not perfectly tailored to their preferences or skill levels. The constraints affect system accuracy, user experience, and developer workload.
    b. Limited Quotas: While free to use, YouTube Data API v3 relies on quotas. The default daily quota available is 10000 units. Actions like "Create WatchList," "Add/Remove Video To Watch List," will consume about 50 quotas each. If a user creates one watchlist and adds about 20 videos, that will be a total of 1050 units/day. This means that only about 9 - 10 people can use this system in a day. (YouTube Blog, 2024)

2. Lack of Real User Data: The proposed system is a research-level project and not for commercial purposes. This relies on the testing of the personalization model on metrics only.
3. Data Integrity: The accuracy of the model to predict content depends on the integrity of a video's metadata. If a video's metadata falsely represents the content of the video, there will be no way for the system to further ascertain if the video content matches the user's intent.
4. Skills Domain: The current system will only be limited to twelve specific skill domains of Python, JavaScript, web development, data science, data analysis, machine learning, artificial intelligence, cloud computing, cybersecurity, UI/UX, and networking.

1.7 Operational Definition of Terms

1. Collaborative Filtering: Collaborative filtering predicts user interests based on similarities between users or items using historical interaction data (Muniraja et al, 2026)
2. Content-Based Filtering: Content-based filtering recommends items similar to those a user previously liked by analyzing item features and characteristics (Ko et al, 2022)
3. Difficulty Level: Difficulty level refers to the degree of complexity or challenge associated with a learning task, instructional material, or educational activity. (Khalifeh et al, 2026)
4. Intent-Aware Recommendation: Intent-aware recommendation systems incorporate user intent, goals, or situational context into the recommendation process to improve personalization relevance. (Alberti et al, 2026)
5. Knowledge Profile: A knowledge profile is a structured representation of a learner's knowledge, competencies, preferences, or learning characteristics used for personalization and recommendation. (Kowalczyk et al, 2021)
6. Learning Intent: Learning intent refers to the learner's purpose, objective, or goal for engaging in a learning activity or educational resource. (Tetzlaff et al, 2021)
7. Learning Materials: Learning materials are instructional resources or educational content used to support teaching and learning processes. (Bernacki et al, 2021)

8. Learning Path: A learning path is a structured sequence of learning activities or materials organized to help learners progressively achieve educational goals or competencies. (Khalifeh et al, 2026)
9. MOOC: A MOOC (Massive Open Online Course) is an online course designed for large-scale participation and open access through the internet. Massive Open Online Courses (MOOCs) are designed for open and large-scale online learning. (Zawacki-Ritcher et al, 2021)
10. Personalized Learning: Personalized learning is the adaptation of instruction and learning experiences to meet the needs, preferences, goals, and characteristics of individual learners. It is the systematic adaptation of instruction to individual learners. (Tetzlaff et al, 2021)
11. Recommender System: A recommender system is an information filtering system that predicts user preferences and provides personalized suggestions or recommendations. Recommender systems provide personalized service support to users by learning their previous behaviors and predicting their current preferences (Zhang et al, 2021)
12. Skill Path: A skill path refers to an organized progression of learning activities designed to develop specific competencies or mastery levels in a domain. (Tetzlaff et al, 2021)
13. YouTube Video Metadata: YouTube video metadata refers to descriptive information associated with videos such as titles, descriptions, tags, captions, views, categories, and engagement statistics. (Kohler et al, 2021)

Chapter Two
Literature Review

2.0 Introduction

This chapter aims to explore personalized recommender systems, as used in YouTube and e-learning systems. This chapter is divided into two major parts: theoretical background and literature review. Literature review is composed of summarized academic and industry related publications while theoretical background provides details on the concepts involved in the study. A summary highlights key findings and technology, setting the stage for system analysis and design.

2.1 Theoretical Background

2.1.1 Recommender Systems

The simplest definition of recommender systems is that they are programs attempting to recommend the best items to particular users, with the user's interest in the item being predicted using the data on the items, the users, and the relations between them (Alfaifi, et al 2024). These systems play a pivotal role in digital platforms by reducing information overload and increasing learner engagement. It changes the way businesses communicate with users and strengthens the interactivity between them (Xing Xie et al, 2022).

Types of Personalized Recommender System

1. Collaborative Filtering: Collaborative filtering is a widely used approach that makes recommendations based on similarities between users or items (Kaoutar Errakha et al, 2025). It is based on the principle that users with similar tastes in the past will continue to have similar preferences in the future. It leverages user-item interactions like ratings, clicks, purchases, etc., to make recommendations. Collaborative filtering does not require knowledge about the items themselves, only user interactions. Types of Collaborative Filtering include:
    a. User-Based Collaborative Filtering: This type of collaborative filtering finds users with similar preferences to the target user and recommends items liked by those similar users. A major advantage is that it is simple and intuitive while a core

disadvantage is that it struggles with scalability when the user base is large. An example is the belief that users who liked X will also like Y.

b. Item-Based Collaborative Filtering: This finds items similar to items the target user has liked or interacted with. An advantage over the user-based collaborative filtering is that it is more stable and especially good for large user bases but it still suffers from sparsity issues.

c. Model-Based Collaborative Filtering: This uses machine learning algorithms to learn latent features from the user-item matrix. Common methods include matrix factorization, deep learning models, and neural collaborative filtering. This is better than the other types of collaborative filtering because it handles sparsity better and can capture complex patterns but it is computationally expensive because it requires more data.

2. Content-Based Filtering: Content-based filtering is an information retrieval method that uses item features to select and return items relevant to a user's query. This recommends items with attributes similar to a user's past preferences. This method often takes features of other items in which a user expresses interest into account. It focuses on the similarity between items' attributes and what the user has liked in the past. Types of Content-Based Filtering include:

   a. Keyword/Feature Matching: Here, items are represented by keywords or attributes. Example: If a user likes action movies, recommend other action movies.

   b. TF-IDF / Vectorization Approaches converts item features into vectors like text features and tags and compute similarity between items and user profile

   c. Machine Learning Approaches: This uses supervised learning to predict user ratings/preferences based on item features. This works well for niche or new items and interpretable recommendations but it cannot recommend items outside the user's existing interest and requires detailed item features.

3. Hybrid Models: This combines multiple approaches to balance strengths and weaknesses. Hybrid recommender systems combine collaborative filtering and content-based filtering to leverage the strengths of both methods and overcome their weaknesses. It utilizes methods such as weighted combination of collaborative filtering and content-based

filtering results. In addition, it alternates the use of collaborative and content-based filtering depending on the availability of user and item information. One strength that hybrid recommender systems have over individual recommendation approaches is their ability to overcome the problem of cold start.

2.1.2 Personalized Learning Path Recommender Systems

A personalized learning path system expands the concept of recommendation by organizing the sequence of content in such a way that takes into consideration the context-awareness needed for the development of a particular skill set or learning process. Learning paths are created in accordance with the cognitive level of the learner, the preferred style of learning of the learner, and past performance of the learner. These systems normally combine association rule approach (for the discovery of sequence of knowledge points) with swarm intelligence or any other optimization technique.

2.1.3 Classification Systems

Classification systems refer to machine learning models that classify data into predefined classes or categories using patterns discovered during the model training stage. The technique works through supervised learning and is usually applied in spam identification, sentiment classification, fraud detection, and education systems. In regards to YouLearn, the classification algorithm will classify users or content in terms of levels like beginner, intermediate, and advance.

2.1.4 Sequencing Systems

Sequencing systems are computational models used for ordering concepts, events, or learning tasks in a logical manner in accordance with dependencies and relations. These models are extensively utilized in education-based recommendation systems in the creation of learning pathways for users. Sequencing models facilitate the learning process through gradual transition from simple concepts to more complex ones. Some popular sequencing models are dependency graphs, knowledge graphs, Markov models, and recurrent neural networks.

2.1.5 YouTube

YouTube is a popular video-sharing social networking site, which was created in 2005 by Chad Hurley, Jawed Karim, and Steve Chen. It was acquired by Google in 2006, and it has become the second-most-visited website in the world with billions of visitors every month who watch millions of videos on a daily basis. YouTube earns money from advertisements, subscriptions, and collaborations with creators. As YouTube grew, the scope of the website has expanded to include mobile applications, live streams, games, and various types of videos such as music videos, movies, and vlogs. Although it has greatly influenced global culture and digital entertainment, it has also faced criticism over misinformation, copyright issues, privacy concerns, censorship, and child safety.

2.2 Review of Related Literature

Ramadhan, et al (2021). In this research, the problem of information overload in online learning spaces where students have difficulty finding information that corresponds to their course syllabuses is addressed. The goal of this research is to develop a recommendation system that focuses on matching video information specifically to the course syllabus instead of general topic similarity. The researchers used content based filtering exclusively by measuring cosine similarity between the course/syllabus and YouTube video description through title and description collected using YouTube API. The four recommendation metrics used for evaluation are: relevance, novelty, serendipity, and recommendation diversity. The average score obtained is 81.13%.

Xu, et al (2021). This paper addresses one critical drawback of conventional collaborative filtering algorithms employed in educational recommendation engines, which is the inability of the algorithm to consider semantic connections between recommended courses. As a result, recommendations from the algorithm end up being statistically feasible but not educationally meaningful. In this research, the aim is to design an engine that incorporates knowledge graph embeddings and collaborative filtering in order to ensure that both semantic connections between courses and user behavior help influence the recommendation process. First, knowledge graph embeddings are used to embed the semantic information about courses in a lower dimensional

space where connections such as dependencies and similarity of topics are captured. The evaluation metrics include precision, recall, and F1-score.

Troussas, et al (2023). The core issue that this research deals with is that traditional recommender systems are unable to produce educationally coherent learning paths, because most methods suggest single items without considering the prerequisite relationships among educational items. In the suggested recommender system, the authors build a knowledge graph, in which the nodes are learning activities, and the edges correspond to their prerequisite or semantic relationships. A graph traversal and a path scoring algorithms are used for suggesting learning paths to learners, depending on their current knowledge level, estimated based on the history of interactions. Results show that path-based recommendations outperform item-level recommendations on both recommendation accuracy and learner progression metrics.

Zhang, et al (2023). The work was inspired by the understanding that modern e-learning systems generate huge amounts of learner interactions that cannot be represented accurately with existing recommendation techniques. The objective of the study is to propose a personalized learning path recommendation system based on knowledge graphs and graph convolutional networks that will enable them to capture both domain and learner structure at once. The effectiveness of the proposed system is demonstrated with the help of e-learning interaction datasets through popular recommendation evaluation metrics such as precision, recall, and NDCG.

Ma, et al (2023). This research study was driven by the observation that e-learning recommendation systems generally ignore the significant variations in student's learning styles while making recommendations. The researchers seek to create a more personalized recommendation system that takes into account the different learning styles of different learners. The researchers utilized a combination of knowledge graph-based content models and clustering models based on learning styles. The performance of the study is evaluated through precision, recall, and overall recommendation accuracy and proves to be better than just pure knowledge graph-based models and standard collaborative filtering models. The results confirm that accounting for learning style diversity significantly improves recommendation relevance.

Gu, R. (2025). The researcher intends to create an adaptive learning path recommendation system by integrating the Graph Attention Network (GAT) and Deep Reinforcement Learning (DRL). The graph attention network will be used on the resource dependency graph for the generation of context-dependent embeddings of the content to be fed into a deep Q-network to learn an optimal recommendation policy that maximizes cumulative learning performance rewards in simulated and real interactions within e-learning platforms. This is evaluated using the metrics of both learning gain and recommendation accuracy, where the proposed model outperformed DRL-only, GAT-only, and static collaborative filtering benchmarks.

Yuan, et al (2025). The purpose of this research is to devise a dual attention network that will enable modeling of temporal and frequency-domain characteristics, thereby obtaining more complete representations of user behavior for making course recommendations. The Fast Fourier Transform is used to obtain frequency-domain features from the user interaction sequences and fuse them with conventional self-attention temporal features using a dual-domain fusion approach. The algorithm is tested on MOOC interaction datasets including XuetangX and Coursera using NDCG@10 and Hit@10 metrics. This paper is able to prove the significance of frequency-domain analysis beyond time-domain analysis.

Zheng, et al (2024). The authors aim to design a comprehensive personalized learning pathway recommendation framework that jointly addresses learner knowledge state estimation, prerequisite-aware content sequencing, and adaptive difficulty progression within a single model. The framework employs knowledge tracing to maintain dynamic estimates of learner mastery, a prerequisite graph to constrain path feasibility, and a ranking model to select and sequence content from a large resource pool. The system is evaluated on multiple e-learning platform datasets using both recommendation accuracy metrics (precision, NDCG) and learning outcome metrics (assessed knowledge gain), demonstrating improvements over single-component baseline systems across both metric types. The study successfully achieves its objective of demonstrating that holistic, integrated pathway recommendation substantially outperforms piecemeal approaches.

Yun, et al. (2024). In this paper, the goal of the researchers is to build an offline Reinforcement Learning (RL) algorithm which can learn to recommend the best possible learning paths based on the previously collected logs without conducting any live experiments. The authors propose to use a doubly constrained offline RL which involves imposing two constraints at once: the distributional one which ensures that the learnt policy will not be much different from the one which collected the data, and the safety constraint which guarantees that the recommended paths will never drop below certain educational quality. Evaluation of the algorithm was conducted on real-world educational platform data using such metrics as learning gain and long-term engagement. It was proven that the doubly constrained offline RL performed way better than the traditional offline RL, conservative Q-learning baselines and supervised sequence models especially in terms of safety and generalization from the historical data.

Zhang, et al (2024). The authors aim to combine process mining with deep knowledge tracing to generate personalized learning path recommendations that reflect both individual knowledge states and observed effective procedural patterns. Process mining is applied to historical learner interaction logs to identify common learning process templates across high-performing learners; deep knowledge tracing (implemented via transformer-based architectures) dynamically estimates each learner's current knowledge state. The two components are integrated, the knowledge state gates which process templates are appropriate, and the template constrains the sequencing of recommended resources. Evaluation on educational platform datasets using both recommendation accuracy (NDCG, recall) and learning outcome (assessment score improvement) metrics shows the integrated approach outperforms knowledge tracing-only and process mining-only baselines.

Li, et al (2024). The authors' aim is to design a multitask learning framework that jointly optimises knowledge tracing accuracy and recommendation quality, enabling each task to mutually reinforce the other through shared representations. The model employs a transformer-based KT backbone that estimates learner mastery of individual knowledge concepts, while a co-trained recommendation head uses these dynamic mastery estimates to

select and rank next exercises. The model is evaluated on multiple educational datasets using AUC for the KT task and NDCG for the recommendation task, with consistent improvements over single-task baselines on both objectives.

Lin, et al (2023). This paper attempts to address the problem of accurate watch time prediction in short video recommendation platforms, based on the assumption that watch time rather than click-through rate or like count is the real indicator of users' interest and satisfaction with videos. The proposed method named as the Tree-based Progressive Regression Model (TPM) represents a tree-based approach where the watch time prediction is solved in several layers of subtasks, progressively improving the prediction results and fixing mistakes made by the previous layer. The model was trained on large-scale video interaction logs from industrial use cases and validated with watch time prediction metrics such as mean absolute error and ranking consistency.

Kaladiya, et al (2025). This article attempts to take advantage of Natural Language Processing (NLP)-based sentiment analysis of comments on YouTube videos to improve educational video recommendations, as it is argued that the sentiment of the comments serves as an exact indicator of education quality that cannot be determined by metadata characteristics. In order to make use of the sentiment information, the system collects comments from educational YouTube videos by using the YouTube Data API, applies sentiment classification to evaluate the percentage of positive, negative and neutral sentiments expressed by learners and uses this sentiment score in a content-based filtering recommendation system. Sentiment analysis techniques are compared using user satisfaction surveys and precision measures, where sentiment inclusion is proven better than metadata-based recommendation.

Chanaa, et al (2024). In this study, the authors seek to develop a course recommendation system based on prerequisites which builds concept prerequisite relationships and uses such a relationship model for constraining the recommendation process to achieve pedagogically-structured learning sequences. To build the concept prerequisite graph, the authors used expert knowledge about concept prerequisite relationships as well as mined information

from co-occurrence of concepts in learner interaction logs. In addition, there is a metadata matching module that compares the learner profiles with the course metadata and identifies appropriate courses that do not exceed the knowledge base of the learners. Evaluation of the system was performed using precision, NDCG with results showing that prerequisite-aware recommendation substantially improves both recommendation relevance and learning sequence coherence compared to popularity-based and standard collaborative filtering baselines.

Mohd Zamani, (2025). The rationale behind this research work arises from the difficulty encountered by students in computer science in finding appropriate educational videos based on their learning styles and needs amidst the huge amount of online content. This research seeks to design and analyze the performance of a recommendation system based on a content-based filtering model. This system will employ a content-based filtering approach which calculates the degree of similarity between students' preference profile and video metadata, sorting out the videos according to their relevance to every learner. Various accuracy measures are used in the evaluation of the performance of the system; Precision is 1.00, Mean Absolute Error (MAE) is 0.49 and Mean Squared Error (MSE) is 0.26.

Carvalho, et al (2024). The background for the paper is in the rapid growth of YouTube as a informal source of education and the difficulty involved in identifying educational videos among the large number of uploads, where the majority of them are entertainment videos. The intention of the paper is to create a machine learning pipeline where videos on YouTube will be firstly classified as educational or not, and then based on the result, will be used as better input in the recommendation system. Various machine learning classifiers are examined for the classification task, and the hyperparameter tuning process is done through Grid Search. It was found that machine learning content quality filtering improves significantly the pool of educational recommendations and the classification is best done using CNN based classifiers. The goals of the study were accomplished.

Yun, et al (2025). The goal of the authors in this paper is to develop an integrated knowledge graph and deep reinforcement learning framework for modeling the structure and dynamics of knowledge acquisition in the English language and producing customized learning trails based on them. In the knowledge graph, prerequisites and semantics between learning items and resources are captured. Simulated and actual datasets of learner interactions are used to evaluate this system by using learning gain and recommendation accuracy as performance indicators.

Kim, et al (2022). This study addresses the challenge of helping learners efficiently locate relevant educational content on YouTube. The aim is to design a content-feature-based recommendation system tailored to YouTube educational videos, implemented as a web application. The system extracts content features from video metadata retrieved in real time through the YouTube Data API. Similarity between learner queries and video content is computed using a cosine similarity method, producing ranked lists of relevant educational videos. Evaluation was conducted through user satisfaction surveys assessing two dimensions yielding scores of 85.36% and 87.80% respectively. The study confirms that content-based feature extraction outperforms pure viewing-history methods for educational purposes.

Table 2.1 Summary of Related Literary Works

Authors (Year)	Technique	Contribution	Limitations
Ramadhan & Musdholifah (2021)	Content-based filtering, cosine similarity, syllabus-aligned YouTube API retrieval	Syllabus-anchored YouTube video recommendation with 81.13% average quality score	No user history; no sequential/skill-path component; small-scale evaluation
Xu et al. (2021)	Knowledge graph (KG) embedding & collaborative filtering (CF) fusion	Fusing semantic KG embeddings with CF improves precision, recall, F1 for course recommendation	Static knowledge graph; no sequential modelling; single platform evaluation

Troussas & Krouska (2023)	Knowledge graph traversal, path scoring, prerequisite modelling	Path-based KG recommendation generates ordered learning sequences respecting prerequisites	Relatively small dataset; no deep learning component; manual graph construction
Zhang et al. (2023)	Knowledge graph & graph convolutional network (GCN), learner embedding propagation	GCN-enriched KG generates coherent prerequisite-aware personalised learning paths	Static graph structure; computationally intensive; single domain evaluation
Ma et al. (2023)	Knowledge graph & learning style clustering, cosine similarity ranking	Learner style segmentation combined with KG improves recommendation relevance	Style classification may not generalise; no temporal/sequential modelling
Gu (2025)	Graph attention network (GAT) & deep Q-network reinforcement learning	Adaptive real-time learning path updates via GAT-DRL; outperforms static and DRL-only baselines	High computational cost; requires sufficient interaction data for RL training
Yuan et al. (2025)	FFT frequency-domain analysis & self-attention, hybrid dual-domain fusion	Frequency-domain modelling captures stable long-term preferences missing from temporal-only models; & 10% NDCG	Increased model complexity; FFT sensitivity to sequence length; MOOC-centric evaluation
Zheng et al. (2024)	Unified framework: KT & prerequisite	Integrated pathway system outperforms	High system complexity; component

	graph & ranking model, multi-metric evaluation	component-only approaches; especially effective for intermediate learners	interdependence makes tuning difficult; single-domain evaluation
Yun et al. (2024)	Doubly constrained offline RL, distributional & safety constraints	Safe offline RL for learning path recommendation without live experimentation; outperforms online RL baselines	Performance bounded by offline data quality; conservative constraints may limit exploration
Zhang et al. (2024)	Process mining (pattern extraction) & deep knowledge tracing (transformer)	Process-type path recommendation aligns with effective learning procedures; outperforms single-component systems	Process templates may not generalise across domains; requires high-quality historical logs
Li et al. (2024)	Multitask learning: transformer KT & recommendation head (MLKT4Rec)	Joint KT-recommendation training mutually improves both tasks; superior to single-task models	Requires labelled responses (correct/incorrect) alongside interaction data; evaluated on exercise domains
Lin et al. (2023)	Tree-based progressive regression model (TPM), watch-time prediction (KDD '23)	Hierarchical regression improves watch-time prediction; positive real-world platform deployment results	Watch-time prediction focus; does not address skill sequencing; short-video platform context

Kaladiya & Patel (2025)	Sentiment analysis (transformer classifiers) & content-based filtering, YouTube API	Comment sentiment augments educational video quality filtering; improves perceived recommendation quality	Comment quality and volume vary; sentiment may not reflect pedagogical effectiveness; limited scale
Chanaa & El Faddouli (2024)	Prerequisite graph (expert and mined) & metadata matching, pedagogical coherence metric	Prerequisites-based recommendation ensures pedagogically valid sequences; novel coherence evaluation	Prerequisite graph construction requires expert input; scalability to large video libraries unproven
Mohd Zamani (2025)	Content-based filtering, cosine similarity, user preference profiles	High-precision (1.00) educational video recommendation for CS students; validated prototype	Thesis-level scale; no sequential modelling; no personalisation beyond profile matching
Carvalho et al. (2024)	Random Forest, Neural Networks, CNN (with Grid Search), educational video classification & recommendation	ML-based classification filters non-educational YouTube content; CNN achieves highest classification accuracy	Classification errors propagate to recommendation; no sequential/skill-path component
Yun et al. (2025)	Knowledge graph & exponential forgetting mechanism & deep RL policy (Scientific Reports)	KG-DRL with forgetting model generates adaptive personalised English learning paths; validated on real learner data	Domain-specific (English language); forgetting model requires calibration; RL sample efficiency

Kim & Kim (2022)	Content-based filtering, cosine similarity, YouTube API	YouTube-specific educational video recommender using content features; 85–87% user satisfaction	No personalisation model; relies on static metadata; no learning path sequencing

2.4 Research Gaps

1. Lack of a YouTube-Specific Learning Path Sequencer System: While there are many YouTube-specific recommender systems, there is none that combines the key tasks of personalization, classification, sequencing and recommendation in one system.
2. Low Focus on Digital Skills and Skill-Paths: The studies aimed at building recommender systems for educational content covering subject-specific domains with none focusing on digital skills and skill paths.
3. Skill-Path Progression: The recommenders reviewed focused more on finding a suitable output to an input without taking into account skill progression.

Chapter Three
System Analysis and Design

3.0 Introduction

This chapter presents the system analysis and design for the development of YouLearn: A Personalized YouTube-Based Skill-Path Recommender System. This system looks into the existing YouTube default recommender system, while constructively evaluating and criticizing this system. It further discusses methodologies used in the development of YouLearn, models it in readiness for system implementation

3.1 System Analysis

3.1.1 Analysis of the Existing System

The system analyzed is YouTube's Default Recommendation System. YouTube's default recommender system was introduced initially in 2006 following Google's acquisition of YouTube. The system has undergone several improvements since then with major algorithmic changes in 2012 and 2016 when watch-time optimization and deep learning techniques were introduced respectively. Despite these major advancements, the system still remains primarily engagement-driven and does not support structured skill learning or educational progression.

3.1.1.1 YouTube

The existing YouTube system supports video upload, processing, metadata generation, and multimedia delivery across distributed networks. Each uploaded video is associated with metadata such as titles, descriptions, tags, captions, and categories, which enable efficient indexing and content retrieval. In addition, YouTube continuously captures user interaction data including views, watch time, likes, comments, and subscriptions. The platform also employs analytics and logging systems to process large volumes of user and content data in real time.

For the proposed system, YouTube serves as the primary source of skill-related dataset for model training and educational video content that will be returned. The existing YouTube platform does not inherently provide guided learning progression, skill-based sequencing, or personalized

educational pathways. Consequently, the proposed system extends the capabilities of YouTube by introducing structured learning recommendations that align video content with users' skill levels, learning goals, and progression requirements.

3.1.1.2 YouTube Default Recommendation Algorithm

YouTube's recommender system was built on the simple principle of helping people find the videos they want to watch and the videos that will give them value (YouTube Blog, 2021). You can find recommendations at work in two major pages:

- The YouTube homepage which displays a mixture of personalized recommendations, subscriptions, advertisements, and the latest videos from subscribers.
- The second page is the up-next panel that appears when you are watching a video and suggests additional content based on what you are currently watching or similar videos from the channel.

The system was built on the intuition that everyone has unique viewing habits. It first compares these viewing habits with those that are similar to the particular user and uses that information to suggest content and sends a number of signals to each other to help inform the system about what you find satisfying. These signals include clicks, watchtime, survey responses, sharing, likes and dislikes referred to as data. The key definitions of the different data that feeds the system are:

- Clicks: Clicking on a video provides a strong indication that you will watch it and find it satisfying.
- Watchtime: This is the length of time you have watched a video. It provides personalized signals to the system about what you most likely want to watch.
- Survey response: Asking viewers if they are really satisfied with the video they just watched. The value is measured by asking you to rate a video from 1 to five stars. The results of this survey are fed into a machine learning model and used to train an algorithm that predicts potential survey responses for everyone else. The accuracy of these predictions is tested by asking the model to predict already known survey answers.

- Engagements: On average, viewers are satisfied with videos they like or share and dissatisfied with videos they dislike. The system uses this information to recommend videos with a high number of engagements.

This system has the following strengths:

- Ability to recommend a wide range of content in a particular topic, say data and analytics.
- Ability to make customized suggestions based on user's interactions with channels.

3.1.3 Limitation Of Existing System

- Over-Reliance On Popularity Metrics: Personalized recommendation returned by the system is about previous watch history, not a goal the learner is working towards.
- Poor Recommendations For New Users: YouTube recommender system uses a collaborative filtering recommendation algorithm causing cold-start problems.
- Increases Learning Inefficiency: The recommender system recommends multiple videos that match your search term without structuring it in the manner a learner will learn it.
- Lack Of Structured Paths: The recommender system is only interested in generating videos on a particular topic with no attention paid to pedagogy.
- Lack Of Intent-Awareness:The recommender systems are not aware of the goals that the learner wishes to achieve instead, they are only aware of the context, that is, the specific search term the learner has entered.
- No Learning Outcome Tracking: The system cannot track watched videos and might still recommend videos that have been previously watched.

3.1.4 Analysis of the Proposed System

The proposed system is a highly innovative technology that combines machine learning and YouTube's data API to improve and structure recommendations in a way that makes learning on the said channel more pedagogically structured.

Key Improvements

- Intent Awareness: This system will be aware of the intentions of the user like skill, industry, tools, and skill level (beginner, intermediate and advanced).
- Presence of Structured Paths: The system will generate a watchlist of pedagogically-structured content that aligns with users' set goals.
- Progress Tracking: The system will be able to track learning progress by ticking watched videos and updating the progress bar.
- A User Interface For Interacting With Users: Unlike YouTube recommender system that aligns on YouTube's search bar to recommend items, this system has its own standalone interface that interacts with users.

3.2 System Requirements Specifications (SRS)

The recommender system requires an interface that will interact between the user, the machine learning algorithms, and YouTube. When a user enters an input, the system will be able to use its hybrid machine learning algorithm to determine which content type to recommend by matching keywords in tools, skills, skill level, and industry to videos on YouTube, and generate a watchlist for the learner by pulling videos via the YouTube Data API v3.

3.2.1 Functional Requirements

- Register and Authenticate Users: The system will be able to register and authenticate users using the Google account attached to the desired YouTube channel and link the Google account to an existing YouTube account associated with the Google account.
- Capture learner's data: Learner's data like skill level, industry, tools, skill will be asked from the user via the interface.
- Fetch educational videos using YouTube APIs: Through YouTube's API, the system will create a watch list for the learner based on the content gathered.
- The system should be able to build a watchlist of structured content based on learner input.
- The system should be able to track watched videos and automatically update learning progress.
- The system should allow users to rate or provide feedback on recommended videos.

- The system should be able to store user data and recommendation history.

3.2.2 Non-Functional Requirements

- Scalability: The architecture must be able to support the estimated 2.7 billion people (Global Media Insight, January 2026) who already use YouTube. This creates a need to have an infrastructure that has the capability to scale horizontally, cloud auto-scale, and have capabilities to manage resources during unexpected high traffic.
- Real-Time System: This system must be able to stay up-to-date with freshly-released videos by searching for videos through the YouTube Data v3 API.
- Usability: The user interface will imitate YouTube's interface by using the exact design scheme that YouTube uses on both light and dark schemes. It will also maintain a consistent design scheme throughout the entire application.
- Reliability: The system will have minimal system downtime.
- Proper Error Handling: The system should be able to handle errors by using interactive widgets to explain what went wrong and why it went wrong.
- Maintainability: The system will use a modular architecture that is reusable with proper commenting and documentation to explain each step of the system.

3.3 System Design Methodology

The design of YouLearn is based on the Structured System Design Methodology (SSDM). This methodology uses Data Flow Diagrams (DFDs) and Entity Relationship Diagrams (ERD) to model the data flow and data storage to provide scalability and efficiency respectively.

3.3.1 Justification For The Methodology

Structured System Design Methodology was adopted for the development of YouLearn because:

1. The system has a clear requirement definition with almost no change made along the development process.
2. It allows for process decomposition, data flow analysis and database structure modeling through Data Flow Diagrams (DFDs) and Entity Relationship Diagrams (ERD).

3. The methodology follows a top-down structured approach allowing the system to be broken into smaller manageable subsystems making it easier to analyze how data moves through the system, identify external entities, define processes and specify data stores before implementation begins.

3.3.2 Machine Learning Methodology

The Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology was adopted as the guiding framework for the development of the YouLearn machine learning service layer. The methodology was selected for this study because the YouLearn platform involves multiple machine learning processes including dataset collection, preprocessing, feature engineering, recommendation modelling, classification, evaluation, and deployment. The methodology also supports systematic handling of heterogeneous datasets such as the youtube_api_raw_dataset and the Coursera proxy interaction datasets used in this research. The six phases of CRISP-DM guided the implementation and evaluation of the content-based recommender system, skill difficulty classifier, and SVD personalization model developed for YouLearn.

Table 3.1 CRISP-DM Phase vs YouLearn Activity

CRISP-DM Phase	**Activity in YouLearn**
Business Understanding	Define objectives for YouLearn
Data Understanding	Collect and analyze YouTube API dataset and Coursera proxy datasets, examine class distributions and interaction patterns
Data Preparation	Clean text, remove duplicates, infer labels, normalize ratings, engineer features, and apply TF-IDF vectorization

Modeling	Train content-based recommender, skill difficulty classifier, personalization models and build a Concept Dependency Graph (CDG)
Evaluation	Evaluate models using Precision@5, Precision@10, Accuracy, Recall, F1-Score, AUC, RMSE, and MAE
Deployment	Integrate trained models into the FastAPI-based machine learning service layer

3.3.3 Method of Data Collection

Two major datasets were used in the development of YouLearn. A primary dataset collected from YouTube across twelve domains and a proxy domain-matched dataset downloaded from Kaggle.

3.3.3.1 Primary Dataset Method of Collection

The primary dataset called youtube_api_raw_dataset.csv was collected by submitting twelve targeted search queries to the YouTube Data API v3, each corresponding to one of the skill domains supported by YouLearn. The twelve domains are: Python, JavaScript, web development, data science, data analysis, machine learning, artificial intelligence, cloud computing, cybersecurity, UI/UX, and networking. For each query, the API returned the top-ranked 50 video results, from which the following metadata fields were extracted and stored: video_id, title, description, channel_title, published_at, tags, view_count, like_count, and comment_count. The resulting dataset was a representative cross-section of YouTube's most ranked educational content across the twelve target skill domains.

3.3.3.2 Primary Dataset Preprocessing Steps

During the data collection stage, an end-to-end data pipeline was set up to check for existing data in the youtube_api_raw_dataset.csv file, then collect new data using the YouTube Data API v3 key. The cleaning started by filling empty cells in tags, description and title with an empty string (“”), dropped duplicate rows using the video_id column as a target column, and merged title, tag,

and description to ensure a feature rich description before saving the newly collected cleaned dataset in the same file youtube_api_raw_dataset.csv.

To prepare the collected data in readiness for model training, the researcher used the clean_text() function which processed each video's combined title, description, and tags field by converting all text to lowercase; removing punctuation and special characters using regular expression substitution; collapsing multiple whitespace characters to a single space; and stripping English stop words using NLTK's stop word corpus. Single-character tokens were additionally removed as they carry no semantic information.

The raw YouTube dataset contains no explicit difficulty labels, therefore to train a skill classifying model, the infer_difficulty() function was used to label each of the videos into one of three labels of beginner, intermediate or advanced using a weighted keyword scoring system. The keywords used for each label was:

- BEGINNER_KEYWORDS (e.g., introduction, for beginners, crash course, getting started, from scratch),
- INTERMEDIATE_KEYWORDS (e.g., practical, project, build, hands-on)
- ADVANCED_KEYWORDS (e.g., advanced, deep dive, masterclass, optimization, production).

Analogous functions (infer_skill() and infer_industry()) assigned each video to one of the twelve skill categories and nine industry categories respectively. Nine industry categories each with keywords were set across technology, healthcare, finance, education, marketing, data, security, design, cloud and a general set for industries not specified. An engagement score was computed for each video as a weighted log-normalised combination of interaction metrics:

engagement_score = 0.50 × log(1 + view_count) + 0.35 × log(1 + like_count) + 0.15 × log(1 + comment_count).

Logarithmic scaling was applied to prevent outlier videos with extreme view counts from dominating the ranking signal. This score is used as a secondary ranking criterion within concept stages during sequencing.

For the content-based recommender, an enriched text representation was constructed for each video by appending its inferred skill, industry, and difficulty labels to the cleaned content_text field. This enriched representation was then vectorised using TF-IDF with a 1–2 gram range and

30,000 vocabulary features, with sublinear term frequency scaling applied. For the skill classifier, a separate TF-IDF vectoriser with a 1–3 gram range and 20,000 features was fitted on the content_text field alone. Fig 3.1 below details the primary dataset distribution of all three keyword classification.

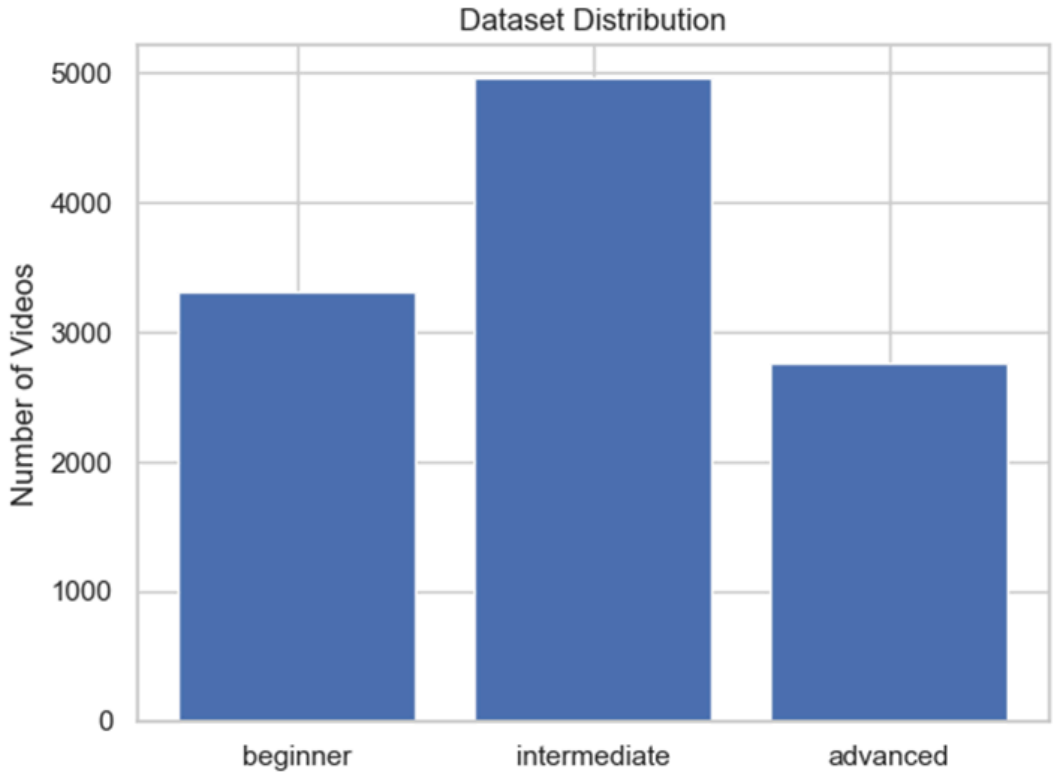

Fig 3.1: A class distribution chart of the keyword labels

3.3.3.3 Secondary Dataset Method of Collection

Since no publicly available dataset of YouTube user-interaction data exists (YouTube has never released user data due to its user data protection policy) and the current project has no real users, a domain-matched proxy dataset was used to pre-train the Singular Vector Decomposition (SVD) personalisation model. The Coursera_course.csv and the Coursera_reviews.csv dataset, containing approximately 1.45 million learner ratings on online technology courses, was selected as the proxy dataset. This dataset was chosen because it represents learner preferences for online educational content, which has the same domain intent, skill keywords, and learning-oriented user behaviour as YouLearn. The companion coursera_courses.csv file was used to filter ratings

to technology-relevant courses only, retaining subjects that matched the YouLearn supported skill domain. Fig 3.2 below is a class distribution of skills across all 12 supported domains

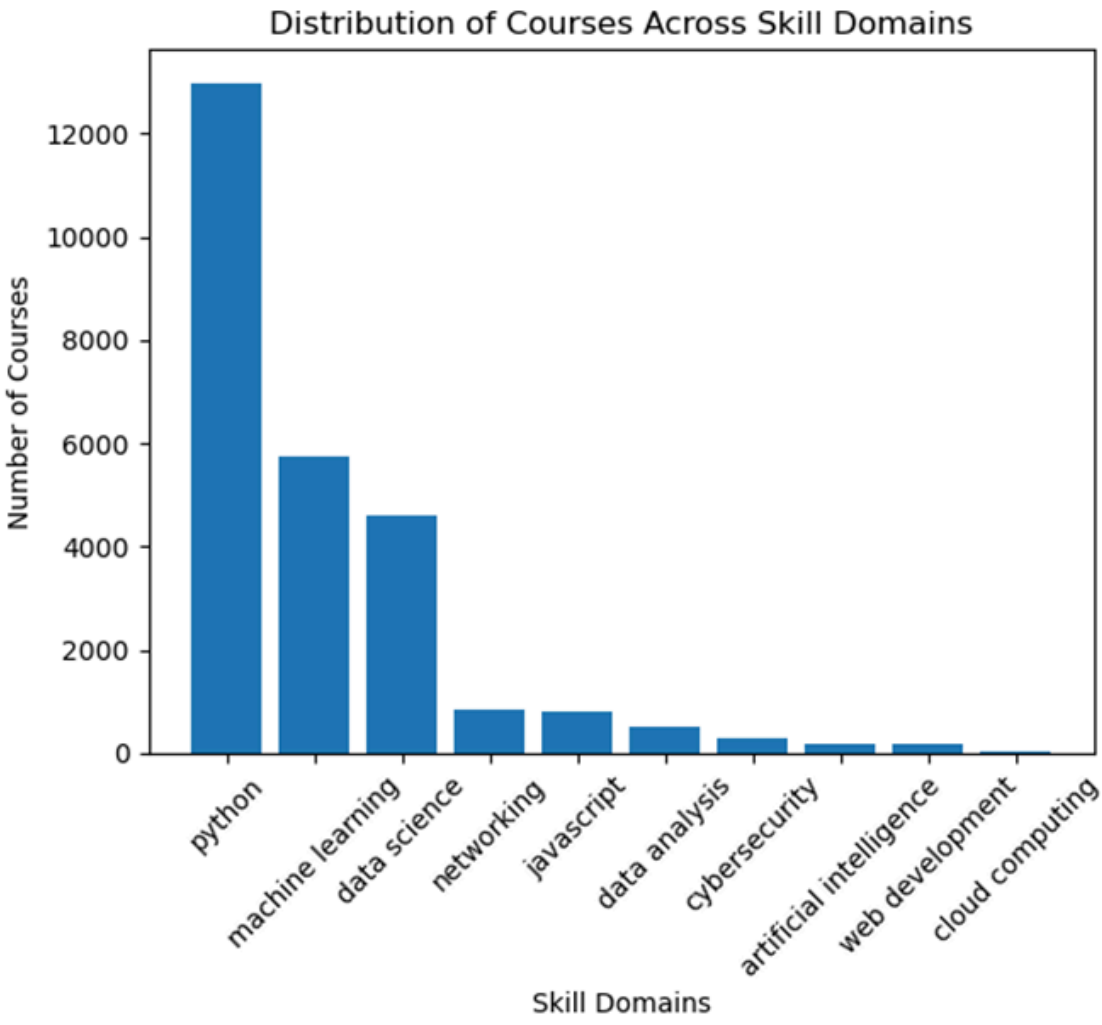

Fig 3.2 Coursera Dataset Class Distribution

3.3.3.4 Secondary Dataset Preprocessing

The Coursera datasets which consists of courses and reviews, were subjected to data cleaning by eliminating duplicates as well as entries that contained missing critical attributes such as user ID, course ID, and rating to ensure data integrity. Additionally, the textual attributes like course titles, skills, and reviews, were normalized by utilizing the same clean_text() function used for cleaning data used on the main dataset, to ensure consistency in lowercasing, removing punctuations, and removing unnecessary whitespaces. The skills attribute was further cleaned

through tokenization and standardization to create consistent skills representation. After that, the datasets were merged on the course_id attribute to create a unified user-item interaction dataset. Afterward, feature engineering was done by combining course titles and skills in order to create better representation that enhances semantic relevance. The dataset was filtered by using the target queries found in the youtube_api_raw_dataset.csv in order to include only domain-specific courses relevant to the objectives of the system. Users and items with less than five interactions per user were eliminated to reduce sparsity. Finally, the processed data was structured into a standardized user–item–rating format, making it suitable for training and evaluation within a recommendation framework.

The ratings column which was within a 1–5 scale was normalized to a 0–1 range using min–max scaling to ensure consistency with the implicit feedback representation (likes = 1, dislikes = 0). That is to say that 3-5 was normalized as 1 (like) while 1-2 was normalized as 0 (dislike). Fig 3.3 below details the rating class balance distribution between likes and dislikes.

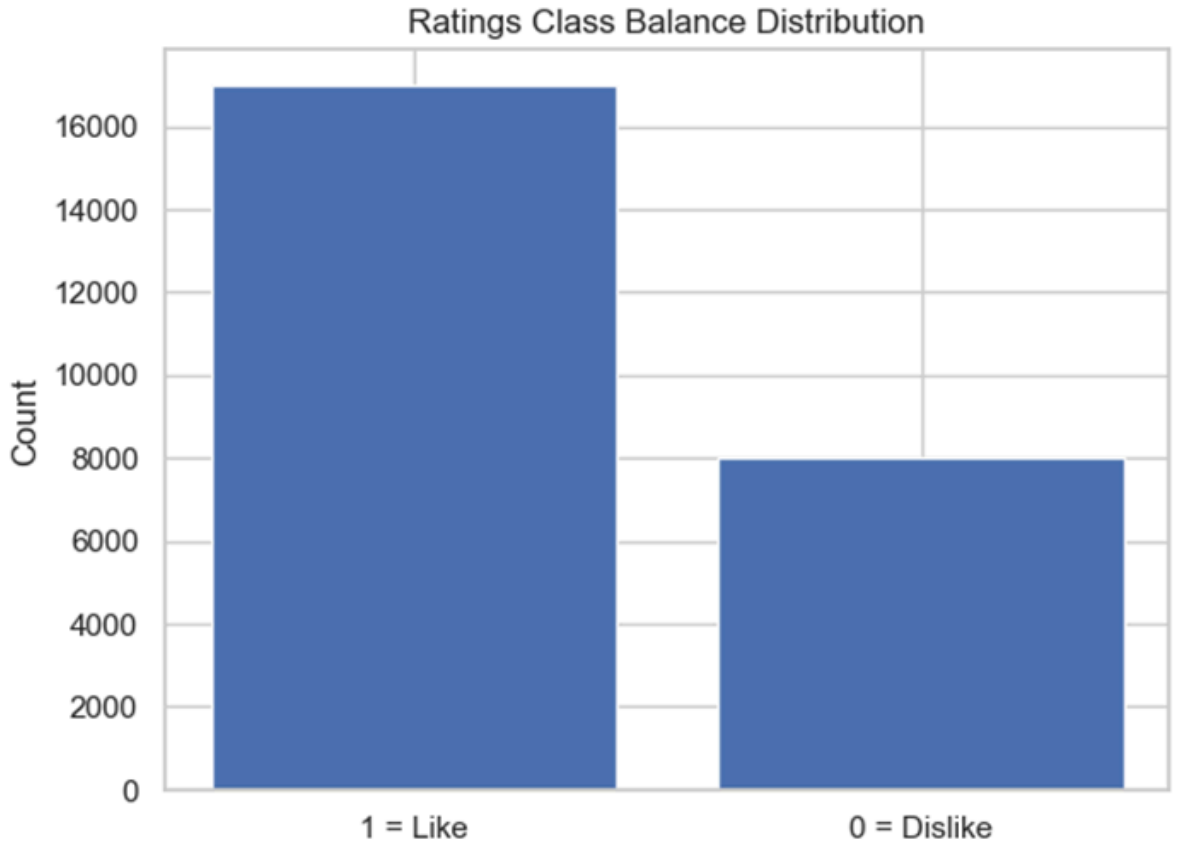

Fig 3.3 Class Distribution of ratings in the Coursera dataset.

3.4 System Modeling Using Structured System Design Tools

YouLearn, which is developed using Structured System Design Methodology (SSDM) methodology, provides a systematic approach for modeling system processes and data through standardized diagrammatic tools such as Use Case Diagrams, Data Flow Diagram (DFDs) and Entity Relationship Diagram (ERDs).

3.4.1 Use Case Diagram

This diagram depicts the relationships between the system users (actors) and the functionalities delivered by the system. It highlights user tasks such as user registration and authentication, data input, watchlist generation, and system interaction between the user and external entities.

Components of a Use Case Diagrams

1. Use Cases: These represent the specific actions or functionalities provided by the system, delivering value to the user. For the recommender systems, the identified use cases are:
    a. Signup with Google: In order to match the recommender system to learner's YouTube, the user will have to login using his Google account. Upon login, the system automatically finds the YouTube channel attached to the Google account
    b. Link YouTube channel: After signup, the learner will be prompted to confirm if the YouTube channel displayed (the one associated with the Google account used in signing up) is the correct one. This is especially useful in the case of Google accounts with multiple channels.
    c. Complete Profile Setup: The user will be prompted to set up their profile to add a profile picture, add a username, and set and confirm password for future login with password.
    d. Create Watchlist: After login, the learner navigates to the dashboard where he can click a "Create Watchlist" button to start creating new skill paths.
    e. Input: The learner will input details about the skill he wishes to learn. These details include watchlist name, skill name, industry, tools, and skill level.
    f. Generate watchlist: Hitting this button will create a watchlist filled with videos according to your input.

g. View Watchlist: On the dashboard, the user can click a specific watchlist card to view details about the watchlist like the number of videos, etc.
h. Edit watchlist: The edit button will help edit skill path details like skill name, industry of choice, etc. This action modifies the watchlist according to the new information.
i. Add/Remove video: In a specific watchlist, users can delete videos they don't like and also manually add the link to a YouTube video into their watchlist.
j. Save changes: After editing a watchlist, the user has to click on save changes to be able to update the new preferences.
k. Rate video: Users can decide to rate videos in a watchlist by giving it a "like" or a "dislike".
l. Mark as complete: Once the link to a video has been clicked, the system will mark that particular video as complete.

2. Actors:
    a. Users: External entities (e.g., learners) interacting with the system to generate watch lists.
3. Associations:
    a. Users interact with the system via text and buttons for actions like creating skill paths, generating watchlists, etc. The interactions are depicted as lines connecting actors to use cases and use cases to use cases to show flow.
4. System Boundary
    a. The system boundary defines the scope of functionalities, focusing on recommendations through the learner's interactions with the system.

Visualizing the Diagram

This diagram illustrated in Fig 3.4 offers a high-level overview of user engagement with the recommender system, emphasizing efficient skill path generation, progress tracking and further manual recommendation.

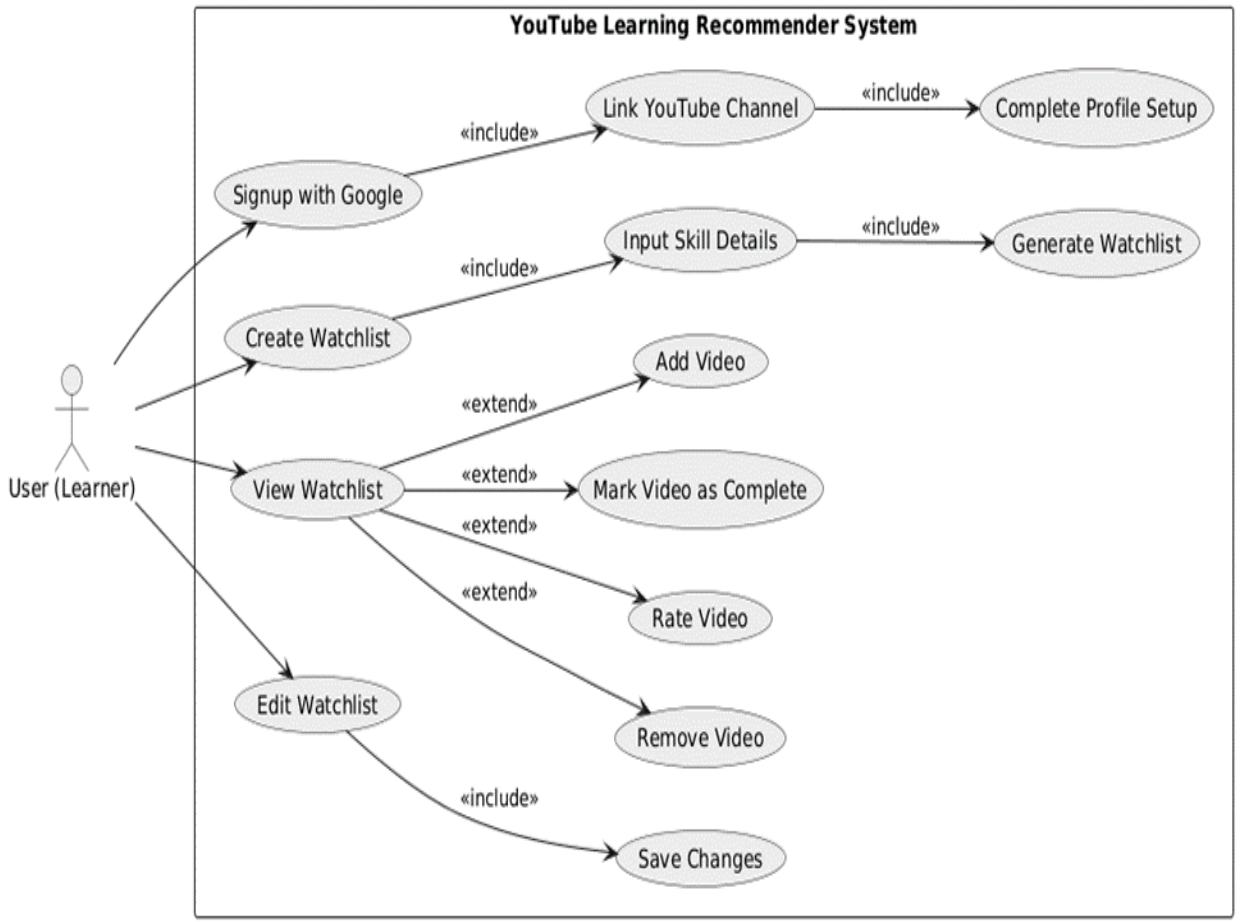

Figure 3.4: Use Case Diagram

3.4.2 Context (Level 0) Data Flow diagram

Level 0 Data Flow Diagram illustrated in Fig 3.5 offers an overview of the entire system as one process interacting with several external entities. The central process which is the YouLearn System receives input like information about registration, preferences on learning, and ratings on the suggested content from the user. It interacts with other external services such as Google Authentication and YouTube Data API v3. Google Authentication performs the authentication of the user and generates the access token, while YouTube API returns educational videos from YouTube. This process offers output to the user in the form of skill paths and video recommendations.

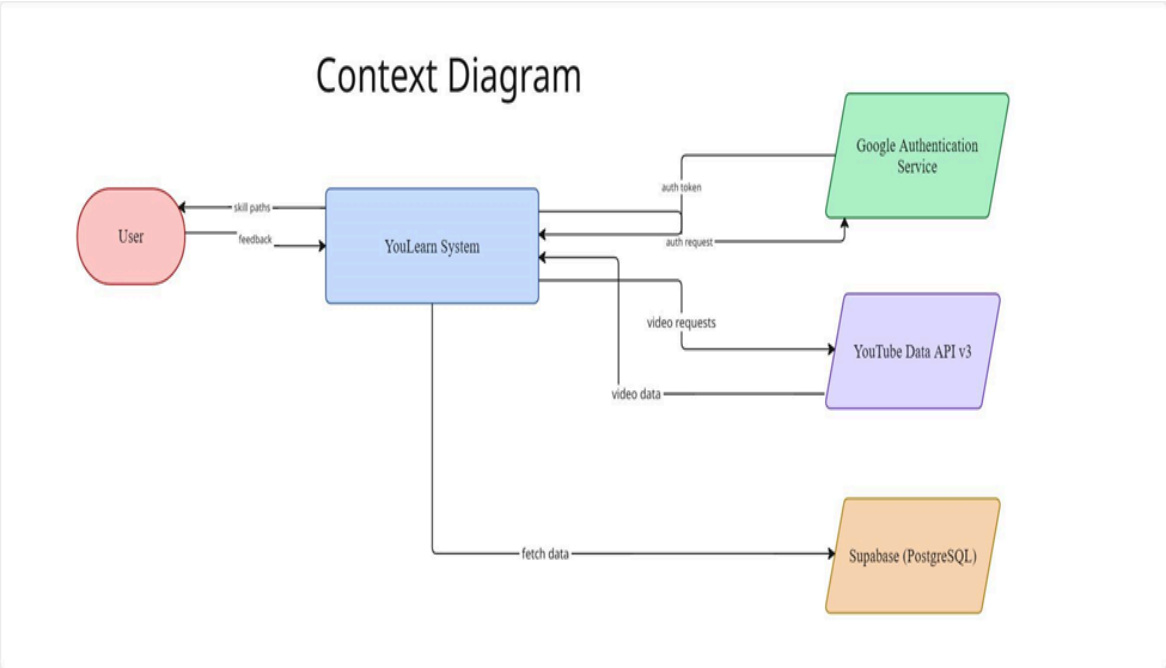

Fig 3.5 Context Diagram

3.4.3 Level 1 Data Flow Diagram

The Level 1 Data Flow Diagram detailed in Fig 3.6 decomposes the system into five major processes of:

1. The Auth and Profile process handles user login through Google Authentication and manages user profile data stored in the Profiles database.
2. The Watchlist Creation process captures user learning preferences such as skill, level, and industry, and stores them in the Watchlists database.
3. The Recommendation Engine is the core intelligence of the system. It receives user profile data and learning preferences, analyzes them, and sends a query to the YouTube Data API v3 to generate structured skill paths and query parameters for video retrieval.
4. The Video Generation process fetches the data sent from the YouTube Data API v3 and stores retrieved video data in the Videos database.
5. The Ratings process communicates whether a user likes or dislike a video using 0 and 1, 0 for dislike and 1 for like.
6. The Watchlist Progress process tracks the learner's learning progress using a percentile function.

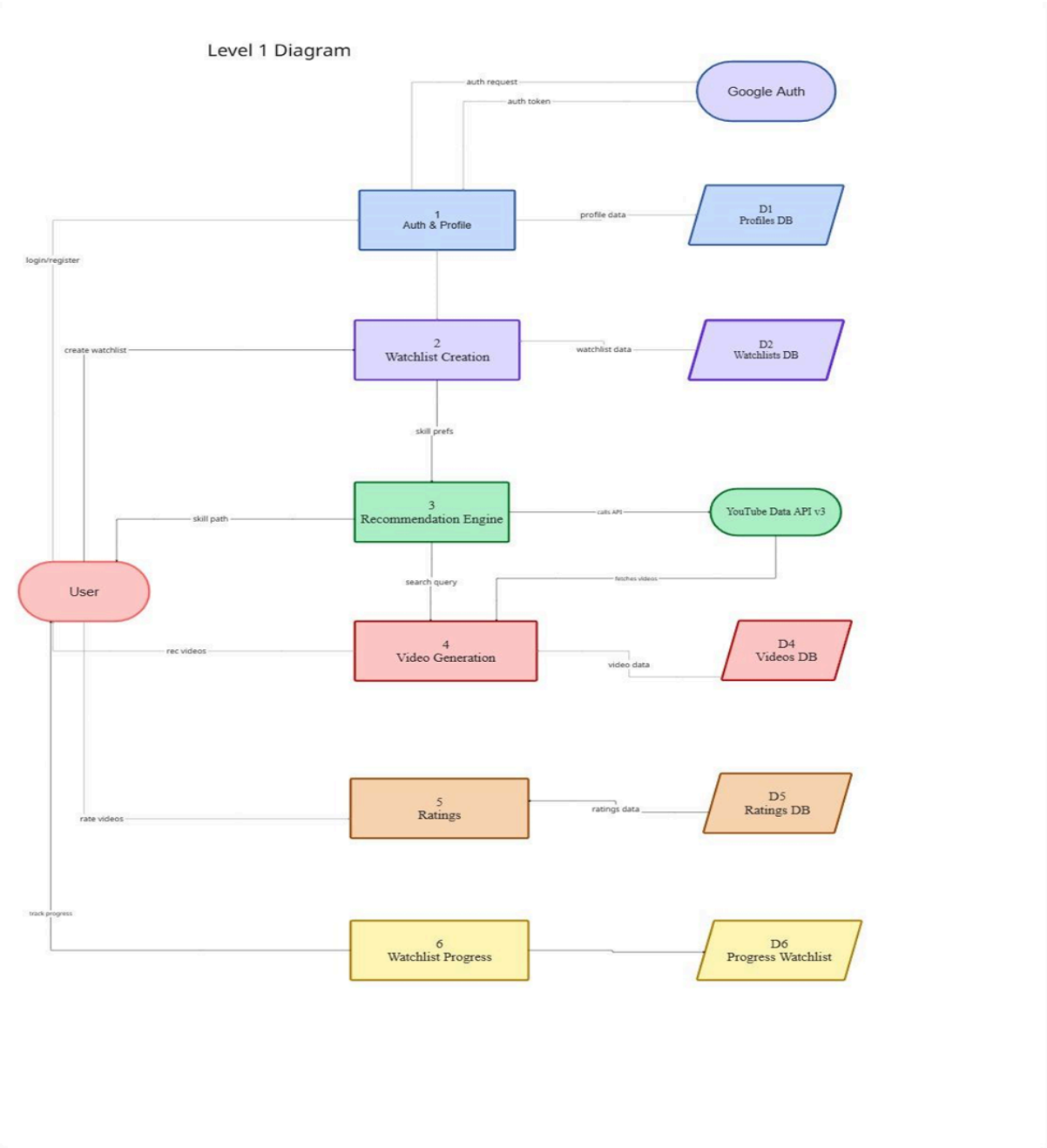

Fig 3.6 Level 1 Diagram

3.5 System Design

3.5.1 Input Design

The recommender system accepts input in the following forms:

1. Text: The recommender system accepts text directly from the user via the presentation layer, then validates and processes the word to build a feature representation. It will accept the following data in the following categories
    a. User registration: It will accept user registration using a Google account attached to the user's YouTube account. The YouTube user's metadata will like username and profile picture will be imported into the profile module
    b. Profile creation: The user will be able to edit their existing YouTube username and avatar, the user can also add a password that will be used for subsequent sign-ins.
    c. Watchlist creation: The user will provide information about the watchlist they want to create such as the watchlist name, skill name, skill level, number of videos, industry, and tools
2. Icon-based input: The user can rate a video using a "thumbs up" or a "thumbs down" icon representing like and dislike respectively. There also exists an eye icon that represents if a video has been watched or not.
3. Links: Users can manually insert YouTube video into a watchlist through the "Add Video by URL" section. When the YouTube video url is inserted, the system automatically fetches and inserts the video into the watchlist.

3.5.2 Output Design

The recommender system satisfies user input by outputting the following in response to each of the input formats.

1. Text: After a learner enters user preferences through text, the system outputs a structured list of videos depending on the number (ranging from 5 to 20) specified by the user
2. Icon: When either of the like or dislike icons are clicked, the button will turn to a red color while the eye icon that represents if a system has been watched or not will be crossed.
3. Link: When a YouTube video link is inserted to the "Add Video by URL" section, the system fetches basic information about the video like the video title, channel name, watch count, etc. and displays it to the user

3.5.3 Database Design

The database was modeled using the Entity Relationship Diagram (ERD) depicted in Fig 3.7 ERD represents the logical structure of the database used in the YouLearn System. It consists of five core entities of:

1. The Profiles entities contain user-specific data including user id, full name, email, profile completion status, avatar URL, and date of account creation. This entity form the basis of the system since all other entities can be traced back to this entity either directly or indirectly.
2. The Watchlists entities are entities holding user-created learning paths. Each of the watchlists is tied to one user and has attributes such as skill name, skill level, and industry to enable recommendation generation in accordance with the goals of the learner.
3. Videos are entities containing data about educational videos obtained from YouTube Data API. These videos have attributes such as title, channel name, duration, number of views, and progression-related attributes such as path stage and position.
4. The Ratings entity stores the feedback from users on the generated videos. There is a relationship between users and videos. Each user can provide ratings to many videos and is essential for improving recommendation accuracy.
5. Watchlist Progress: This entity shows the progress of the learner in a certain watchlist using a percentile function.

The relationships between the entities are:

- A Profile can have zero or many Watchlists (1:N)
- A Watchlist can contain zero or many Videos (1:N)
- A Profile can give zero or many Ratings (1:N)
- A Video can receive zero or many Ratings (1:N)

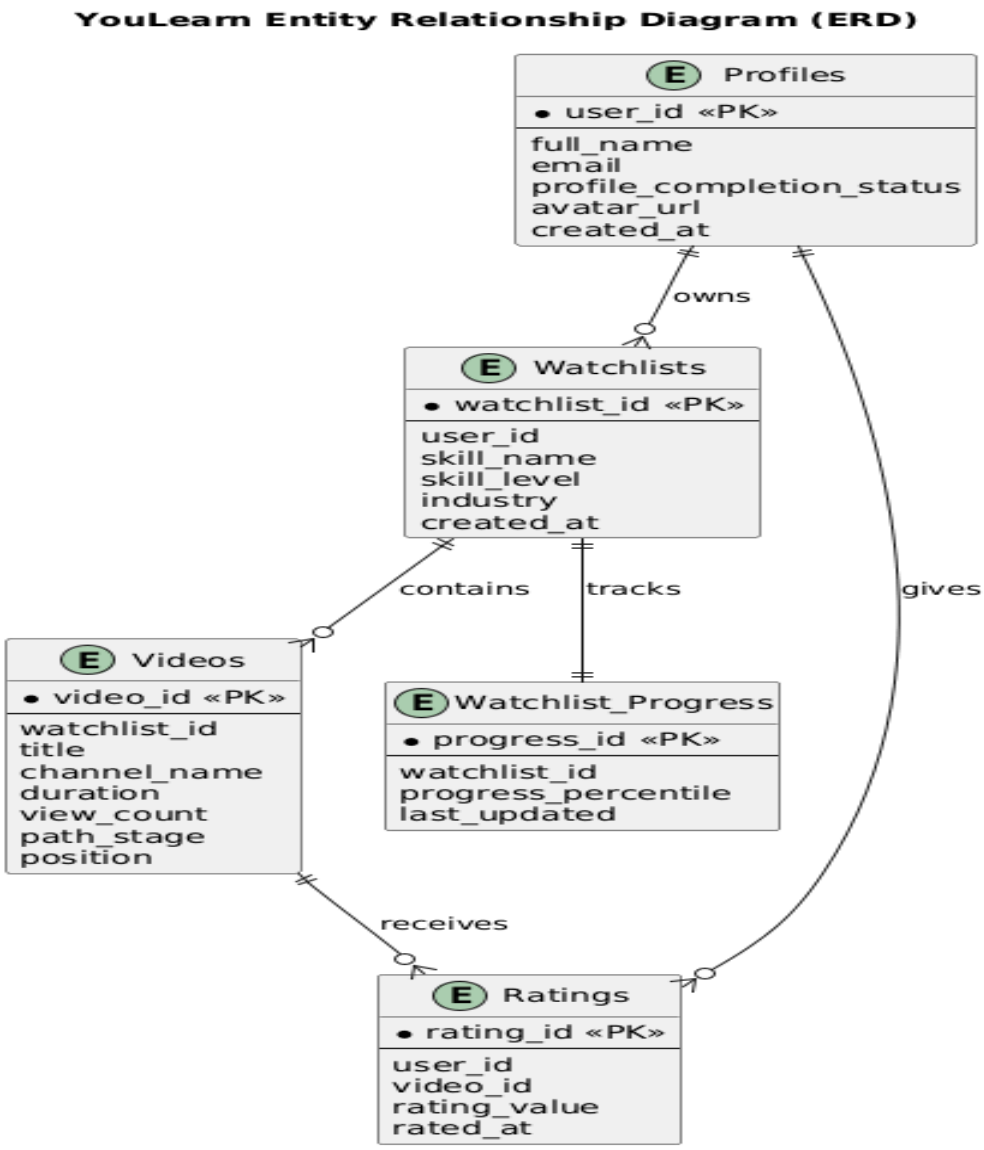

Fig 3.7 Entity Relationship Diagram

3.5.4 System Architecture Design

1. This system employs a simple four-tier architecture to guarantee separation of concerns as illustrated in Fig 3.8.
2. Presentation Layer: The presentation layer acts as the user interface layer for authentication, capturing of learner's learning preferences, watchlist management, and displaying recommended videos. It will interact with the backend through REST APIs and will facilitate all user interactions in an interactive way.

3. Application Layer: The application layer acts as the central layer of the entire application system. It will handle API calls from the presentation layer, user sessions, and routing of data between the database and the machine learning service. It will also provide secure connections to external systems and application logic enforcement.
4. Data Layer: The data layer will be built using PostgreSQL hosted on Supabase and will store the structured data on four main tables that include profiles, watchlists, videos, watchlist progress, and ratings. This tier ensures persistence of data, data integrity, and historical data which will be consumed by the ML engine to improve recommendations.
5. External API Integration (YouTube Data API v3): This system is integrated with YouTube Data API v3 to get educational video metadata including the title of the video, the channel, view count, and the thumbnails.
6. Machine Learning Layer: This layer consists of four sub-layers:
    a. Recommendation Layer: The recommendation layer is the intelligence core of the system. It will be capable of processing user inputs, It will also formulate queries for the YouTube Data API v3 and rank returned videos based on relevance to the user's learning goals.
    b. Skill Classifier Layer: This layer will classify videos according to three levels of "Beginner", "Intermediate," and "Advanced" using keywords like: basics, intro, introduction, beginner for beginners, intermediate, project, build, in depth for intermediate and advanced, expert for advanced.
    c. Personalization Layer: This layer will process learning history, user preferences, like/dislike categories, to make video recommendations.
    d. Concept Dependency Graph: This layer sequences the skills returned into the order they should follow.

The workflow of the machine learning layer is depicted in Fig 3.9

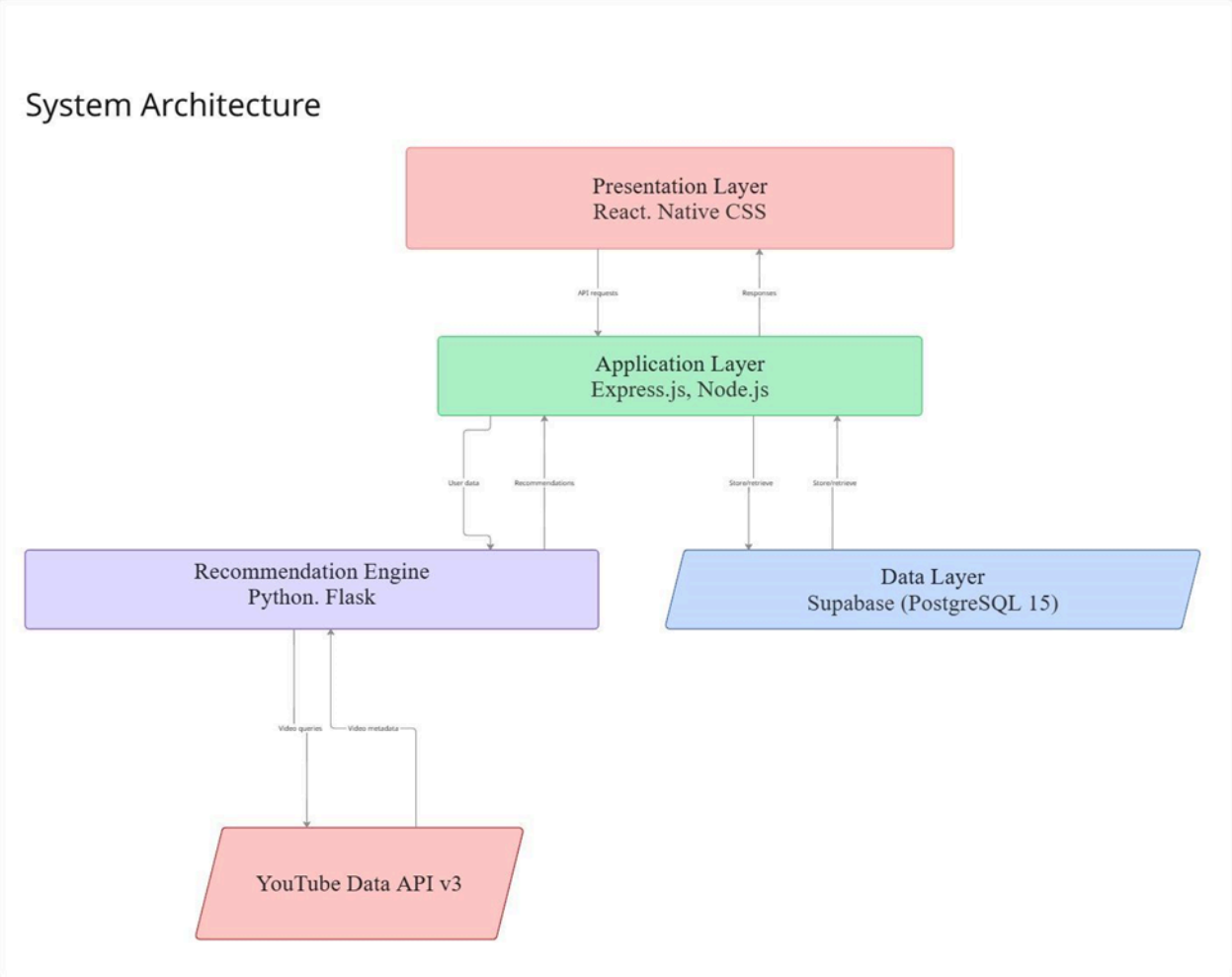

Fig 3.8 System Architecture Overview

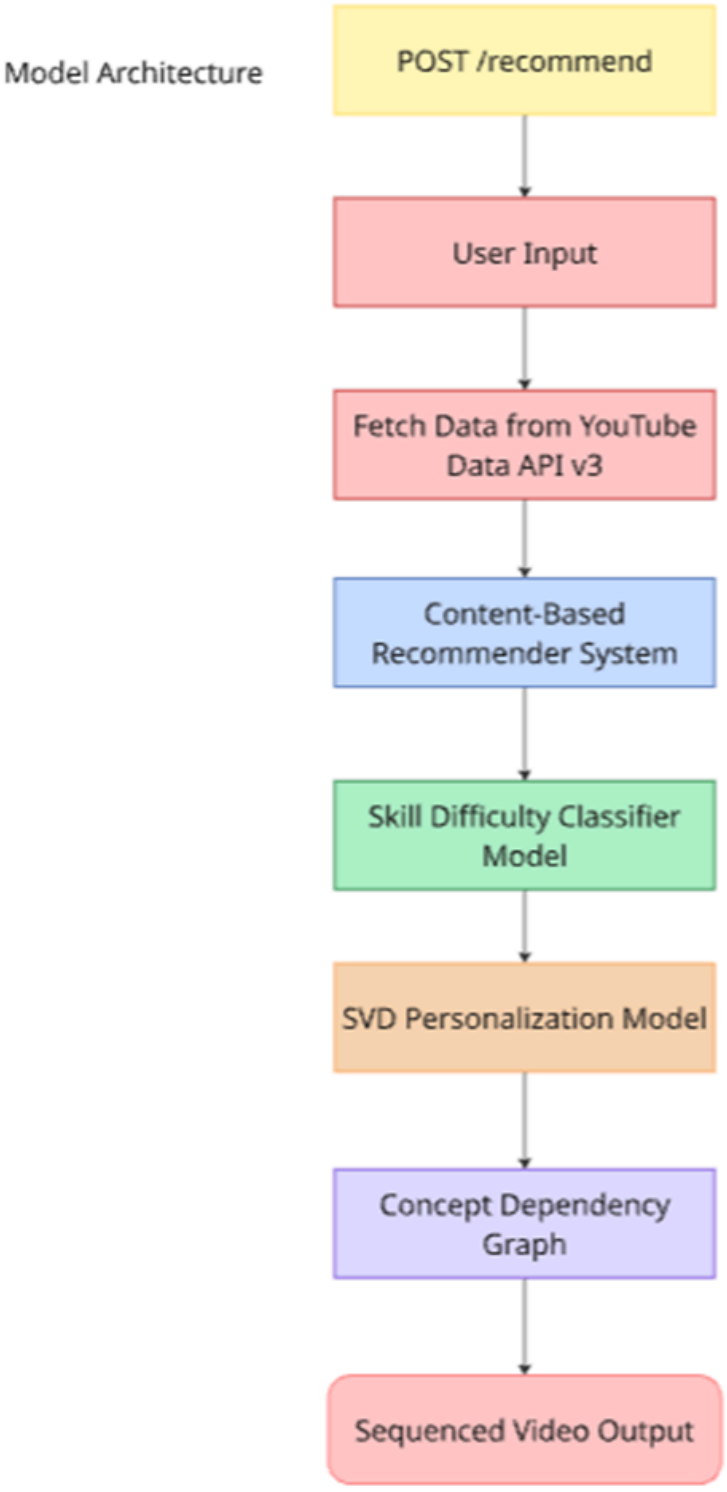

Fig 3.9 Machine Learning Workflow

Chapter Four
System Implementation

4.0 Introduction

This chapter describes the development environment, tools, datasets, preprocessing steps, the machine learning models trained and how the models were evaluated. This chapter demonstrates how the objectives stated in Chapter One were realised in a working application.

4.1 Development Environment and Tools

The YouLearn system was built across three independent service tiers of a React frontend, an Express.js backend, and a Python (FastAPI) machine learning service. Each of the tiers required distinct sets of tools and development environments. Each of these tools were selected under considerations for their ease of use, suitability to task, community support, and compatibility with the project's modular monolith architecture.

4.1.1 Programming Languages and Libraries

Three programming languages were used across the three service tiers.

4.1.1.1 The Machine Learning Tier

Python 3.14 was used as the language of the machine learning service due to its support ecosystem of libraries for text processing, machine learning, web development and testing. The following libraries The scikit-learn library (version 1.4.0) provided the TF-IDF vectoriser and Logistic Regression classifier used in the content-based recommender system and the skill difficulty classifier. The scipy library (version 1.12.0) provided the Singular Value Decomposition (SVD) algorithm (scipy.sparse.linalg.svds) used in the personalisation model. The FastAPI framework was used to expose the machine learning pipeline as a REST API. The NLTK library (version 3.8.1) provided English stop word lists used during text preprocessing. The Google API Python Client library provided the interface to the YouTube Data API v3 for live video retrieval.

4.1.1.2 The Frontend Tier

JavaScript (ECMAScript 2022) was used for both the backend and frontend services. React 18 was used to build the frontend application, with React Router v6 providing client-side routing.

All user interface styling was implemented using vanilla CSS to ensure full design control consistent with the YouTube-inspired visual language of the application.

4.1.1.3 The Backend Tier

JavaScript (ECMAScript 2022) was also used for the backend. Node.js 20 with the Express 4 framework was used to implement the backend REST API. Express was selected for its minimal footprint and strong middleware ecosystem. The Supabase JavaScript SDK was used on the backend for authenticated database operations, and on the frontend for authentication session management.

4.1.1.4 The Data Layer

The database was built with Supabase (PostgreSQL 15) which utilized PostreSQL on cloud. It has five tables of profiles for storing user information, watchlists for storing watchlists created by the user, ratings for storing ratings (0 or 1), watchlist_progress for tracking a watchlist's watch progress and videos that saves the videos collected through the YouTube Data API

Table 4.1 Summary of Used Libraries

Library/Framework	Version	Purpose
FastAPI	0.136.1	A lightweight Python web framework used to build RESTful APIs and handle backend server logic.
Uvicorn	0.47.0	Uvicorn is an Asynchronous Server Gateway Interface (ASGI) that runs FastAPI
google-api-python-client	2.111.0	Provides access to Google APIs, specifically used to interact with the YouTube Data API for video and metadata retrieval.
scikit-learn	1.4.0	A machine learning library used for building, training, and evaluating predictive models. It provided the TF-IDF vectorisation, Logistic Regression model and, cross-validation used at the machine learning layer

pandas	2.1.4	A data manipulation and analysis library used for handling structured datasets such as CSV files.
numpy	1.26.3	Provides support for numerical computations and array operations, forming the foundation for scientific computing.
scipy	1.12.0	Used for advanced mathematical, statistical, and scientific computations. Truncated SVD matrix factorisation for the personalisation model.
joblib	1.3.2	Enables efficient serialization and storage of machine learning models and large data objects.
python-dotenv	1.0.0	Loads environment variables from a .env file for secure configuration management.
requests	2.31.0	A Python HTTP library used for making API calls and handling external requests.
gunicorn	21.2.0	A production-grade WSGI server used to deploy Python web applications.
nltk	3.8.1	A natural language processing library used for text preprocessing and linguistic analysis.
React	19.2.4	A JavaScript library for building dynamic and component-based user interfaces.
React DOM	19.2.4	Provides DOM-specific methods for rendering React components in web applications.
React Router DOM	7.13.2	Handles client-side routing and navigation in React applications.

@supabase/supabase-js	2.101.1	JavaScript client for interacting with Supabase backend services such as authentication and databases.
Vite	8.0.1	A modern frontend build tool that provides fast development and optimized production builds.
@vitejs/plugin-react	6.0.1	Integrates React with Vite for efficient development and hot module replacement.
ESLint	9.39.4	A static code analysis tool used to identify and fix JavaScript code issues.
@eslint/js	9.39.4	Provides base ESLint configurations for JavaScript projects.
eslint-plugin-react-hooks	7.0.1	Enforces best practices for using React Hooks.
eslint-plugin-react-refresh	0.5.2	Ensures compatibility with React Fast Refresh during development.
globals	17.4.0	Provides predefined global variables for JavaScript environments.
@types/react	19.2.14	TypeScript type definitions for React.
@types/react-dom	19.2.3	TypeScript type definitions for React DOM.
Express	4.18.2	A Node.js web framework used for building backend APIs and handling server-side logic.
cors	2.8.5	Middleware that enables Cross-Origin Resource Sharing in Node.js applications.
dotenv	16.3.1	Loads environment variables into Node.js applications for configuration management.

axios	1.6.5	A promise-based HTTP client used for making API requests from the backend.
nodemon	3.0.2	A development tool that automatically restarts the Node.js server upon file changes.

4.1.2 Development Platform and Hardware Requirements

All development was carried out using Visual Studio (VS) Code, Supabase cloud (PostgreSQL 15) and JupyterLab with Git and GitHub used for version control and Google Cloud Console for Google Authentication using OAuth API keys, linking YouTube accounts to YouLearn queries to YouTube and collecting video data into the system.

4.1.2.1 Visual Studio Code (VS Code)

VS Code was selected as the primary development environment because of its multi-language support, integrated terminal, and compatibility with the ESLint and Pylint linting tools used to maintain code quality across the JavaScript and Python codebases respectively. It was used to develop the frontend, backend, and the machine learning layers.

4.1.2.2 JupyterLab

JupyterLab was used to set up a data collection and preprocessing pipeline that runs concurrently, collecting and saving data from YouTube using the YouTube Data API v3 in a csv file called youtube_api_raw_dataset.csv. It was used during the data exploration phase to explore the collected youtube_api_raw_dataset.csv before committing preprocessing logic to the training scripts. It was further used during the preprocessing and visualization phases of both the primary and secondary dataset to clean and visualize the evaluation metrics from the different models trained.

4.1.2.3 Supabase

The database was hosted on Supabase Cloud (PostgreSQL 15), which eliminated the need for a local database server infrastructure.

4.1.2.4 Google Cloud Console

Google Cloud Console was used to get the API keys used in different phases of the project like the Google OAuth client key which was used to setup Google authentication and the YouTube

Data API v3 key was used to link the learner's YouLearn account to their YouTube account. The YouTube Data API v3 key was also used during the data collection phase to collect data and to collect live videos to satisfy user input.

4.1.2.5 Git and GitHub

Git and GitHub played a major role in developing the YouLearn system due to their ability to provide version control, code management, and backup during the development of the system. Due to the use of Structured Design approach in developing the system, the project was divided into several independent modules comprising of a React Frontend, and Express backend, a Supabase database, and FastAPI machine learning service layer. Git assisted in managing changes made to the source code systematically, whereas GitHub helped to manage the repository of the project.

4.2 Model Training

Three models were trained and developed for YouLearn in total. Each is described below with its algorithm, training parameters, theoretical justification, and metrics. They are:

1. A content-based recommender model using TF-IDF (Term Frequency-Inverse Document Frequency) technique.
2. Skill difficulty classifier model using TF-IDF technique and a Logistic regression model
3. A personalization model that uses Singular Vector Decomposition (SVD)
4. A Concept dependency graph

4.2.1 Content-Based Recommender

The content-based recommender starts with preprocessing all videos on YouTube in the form of numerical representation of the titles, descriptions, and tags through the TF-IDF (Term Frequency-Inverse Document Frequency) approach. As a common NLP approach, the TF-IDF technique helps the system measure the weight of each word within a video based on the whole set of documents. It uses 1-2 grams (words and couples of words) for a better representation of individual words and word pairs.

The recommender does not base on textual similarity improvement only. Instead, it calculates the final score with 70% of semantic relevance (similarity to a query) and 30% of normalized

engagement metrics (likes/views). To filter out videos that do not have the corresponding keywords in the title, description, or tags, the topic relevance filter is utilized, employing the skill aliases dictionary. This way, the system avoids selecting generic keywords such as “tutorial” when looking for a relevant one.

4.2.2 Skill Difficulty Classifier

The skill difficulty classifier is designed to work by analyzing the title, description, and tags from the combined video to estimate the degree of difficulty. In other words, the classifier learns to classify videos into one of the three categories of beginner, intermediate, and advanced automatically.

This model is based on a two-stage pipeline. The first stage is the application of the TF-IDF (Term Frequency–Inverse Document Frequency) vectorizer that transforms the text data into the feature vector form. This approach allows detecting important features like “getting started” (for beginners) and “deep dive” (for advanced). These features are then fed to the Logistic Regression classifier model that outputs the probability for the video being assigned to one of the three classes. Logistic Regression is the optimal choice for this task due to the ability to output probabilities for all classes. It means that boundaries between these categories are unclear.

To make the model fair and accurate, class imbalance is considered in the data set. There are many more intermediate-level videos than any other category, so class_weight='balanced' is used.

In order to achieve a proper balance between performance and efficiency, the vocabulary size was kept small enough – only 20,000 features were used in order to avoid high memory consumption. In addition to that, the sublinear scaling factor was used in order to mitigate the effect of overly common words, the default L2 regularization strategy was used in order to avoid overfitting, and the L-BFGS (Limited-memory Bryoden-Fletcher-Goldfarb-Shanno) solver was used for optimization of the model in the multiclass setting. Generally, this classifier allows the system to recommend videos to a user based on his or her skill level. Below is a table summary of the key training parameters.

Table 4.2: Skill difficulty classifier training parameters

Parameter	Value	Justification
ngram_range	(1, 3)	Captures multi-word difficulty phrases (e.g., 'deep dive')
max_features	20,000	Balances vocabulary coverage against memory usage
sublinear_tf	True	Reduces dominance of high-frequency common words
Regularisation C	1.0	Standard L2 regularisation, prevents overfitting
solver	Limited-memory Bryoden-Fletcher-Goldfarb-Shanno (L-BGFS)	Efficient convergence for multinomial classification
multi_class	multinomial	Full probability distribution across all three classes
class_weight	balanced	Corrects for label imbalance in training data

4.2.3 Singular Value Decomposition (SVD) Personalisation Model

The Personalization algorithm is a Collaborative Filtering algorithm which uses the concept of Singular Value Decomposition where the algorithm learns the patterns from user-item interactions (users and course/video in this case). The approximation of the original rating matrix takes place through decomposition of the matrix into three matrices: the first represents user factors, the second one represents item factors, and the third one represents the weight of the factors. Instead of considering the whole matrix, the truncated matrix is considered by taking into consideration the most significant factors only.

Bias terms are added to the algorithm to consider some tendencies such as some users having a consistent high rating while some other items have an increased rating in general for more accurate predictions. The final rating for user-item interaction consists of global average rating, user bias, item bias, and user-item latent interaction.

The model was implemented using a truncated SVD approach, which keeps only 50 latent factors, a value commonly used for medium-sized datasets as it balances performance and computational cost. The key parameters are:

Table 4.3: SVD personalisation model training parameters

Parameter	**Value**	**Justification**
n_factors (k)	50	Standard for 100K-scale datasets (Koren et al., 2009)
Bias terms	User + item bias	Full SVD formulation; reduces RMSE vs. basic SVD
Rating scale (proxy)	1.0 – 5.0	Coursera ratings; thumbs -1→1.0, +1→5.0
Cross-validation	5-fold KFold	Reports mean RMSE and MAE across folds
Validity criterion	RMSE < global mean baseline	Minimum validity

4.2.4 Concept Dependency Graph

A rule-based Concept Dependency Graph (CDG) structurally arranges the recommended content obtained from the content-based filtering model. Unlike the other components described above, this part is just a deterministic knowledge-based system that works at run-time, not a machine learning model.

For every skill supported by the platform (for instance, Python, web development, data science, etc.), there is a defined sequence of several learning stages starting with the basic concepts (such as setting up the working environment) and finishing with advanced stages (such as libraries or deployment for a particular language). In addition, every stage has a pre-defined set of primary keywords and a secondary set of keywords that allows finding the most relevant learning stage based on the title and description of the video.

Once the set of videos is delivered from the recommendation pipeline, the sequencer assigns every video to a particular learning stage depending on the number of matches with the keyword sets, and then videos are fine-tuned according to the skill level of the user (beginner, intermediate, advanced).

The CDG acts as the final structuring layer, transforming individually relevant recommendations into an ordered educational pathway. While the other models focus on what to recommend, the CDG determines how the content should be structured to ensure learning is structured pedagogically, thereby bridging the gap between recommendation and curriculum design.

4.3 System Implementation and Modules

YouLearn is implemented as a three-tier web application with a loosely coupled machine learning microservice with five major functional modules. The five functional modules of the system are described below.

4.3.1 Authentication and Profile Management Module

The authentication module is implemented using Supabase Auth, which manages user registration, login, session handling, and linking user's YouTube account to their Google account through secure OAuth and email/password authentication. During signup, users authenticate via Google OAuth and during login, users can choose between Google OAuth or email/password (if they added one in their profile), after which Supabase automatically creates a session containing the authenticated user's information, including their email and unique identifier. Route protection is implemented using a custom ProtectedRoute component, which checks whether a valid session exists and if no session is found, the user is redirected to the login page, ensuring that only authenticated users can access secured parts of the application.

4.3.2 Watchlist Management Module

The watchlist management module handles tasks consisting of creating, viewing, updating, and deleting of the user's watchlists. When a user submits the Create Watchlist form that includes parameters such as skill name, skill level (beginner, intermediate, or advanced), tool and industry, the request will be processed by the createWatchlist controller in a process that consists of three steps in total. First, a new row will be created in the watchlists table. Secondly, the createWatchlist controller will make a call to the ML service /recommend endpoint, supplying the information about the user ID and skill parameters. The /recommend endpoint will return an ordered list of video recommendations. Lastly, the received video metadata will be inserted into the videos table. The watchlist_progress database view computes progress percentages server-side, so the Dashboard receives aggregated card data in a single query.

4.3.3 Recommendation Engine Module

The recommendation engine is the core intellectual contribution of the YouLearn system. It is implemented as a four-stage pipeline in the Python ML service. Stage 1 retrieves videos through a live YouTube Data API v3 query. This stage applies the topic relevance filter, rejecting any video whose title and description do not contain at least one keyword for the requested skill. Stage 2 applies personalisation based on input (if any) to the retrieved content. Stage 3 applies the Concept Dependency Graph sequencer to assign each video to its position in the ordered learning curriculum. Stage 4 trims the result to the requested number of videos and returns the list to the backend which returns it to the frontend.

4.3.4 Rating and Feedback Module

Likes and dislikes from user interactions will be used to re-train the personalisation model with real user preference data, making the system progressively more personalised as usage grows.

4.3.5 Curriculum Display Module

This module is responsible for displaying sequenced videos from the recommendation engine to the user interface.. Videos can also be added manually via the Edit Watchlist page.

4.4 Model Evaluation

Four sets of evaluation metrics were used to assess the YouLearn system, one for each machine learning model and the Concept Dependency Graph used in the development of YouLearn.

4.4.1 Content-Based Recommendation Model

4.4.1.1 Recommender Performance Metrics

The recommender achieved a precision value of 0.93, indicating that most of the videos recommended to users were highly relevant to their interests and learning preferences. A recall value of 0.90 shows that the recommender system was able to successfully identify and recommend a large proportion of relevant educational videos available to users. The F1-score of 0.91 demonstrates a strong balance between precision and recall, indicating that the system performs effectively in both recommending relevant content and ensuring sufficient content coverage. These high metric values collectively indicate that the recommender system is capable of delivering accurately personalized, and user-centered recommendations. The metrics are visualizd in Fig 4.1

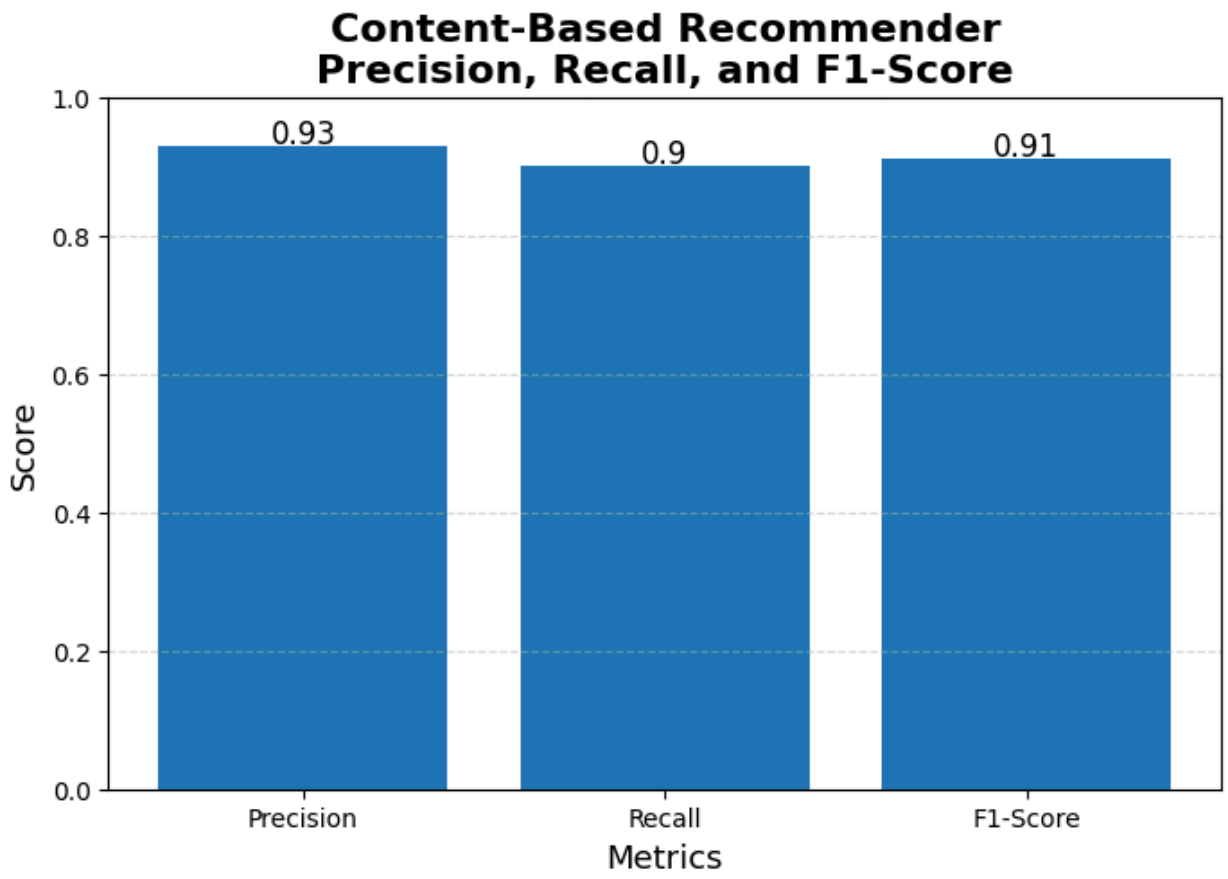

Fig 4.1: Recommender Performance Metric

The recommender model gave an RMSE value of 0.41 and an MAE value of 0.29 as illustrated in Fig 4.2, proving that the recommender system has a high prediction accuracy. The low RMSE value implies that the recommender system produces very few errors in the predictions of user preferences and video relevancy. Likewise, the low MAE value means that the error of the predicted user interactions compared to the real interaction is low and consistent. Together, these metrics demonstrate that the recommender model can reliably predict user interests and provide accurate video recommendations with minimal deviation from actual user behavior.

4.4.1.2 Recommender Error Metrics

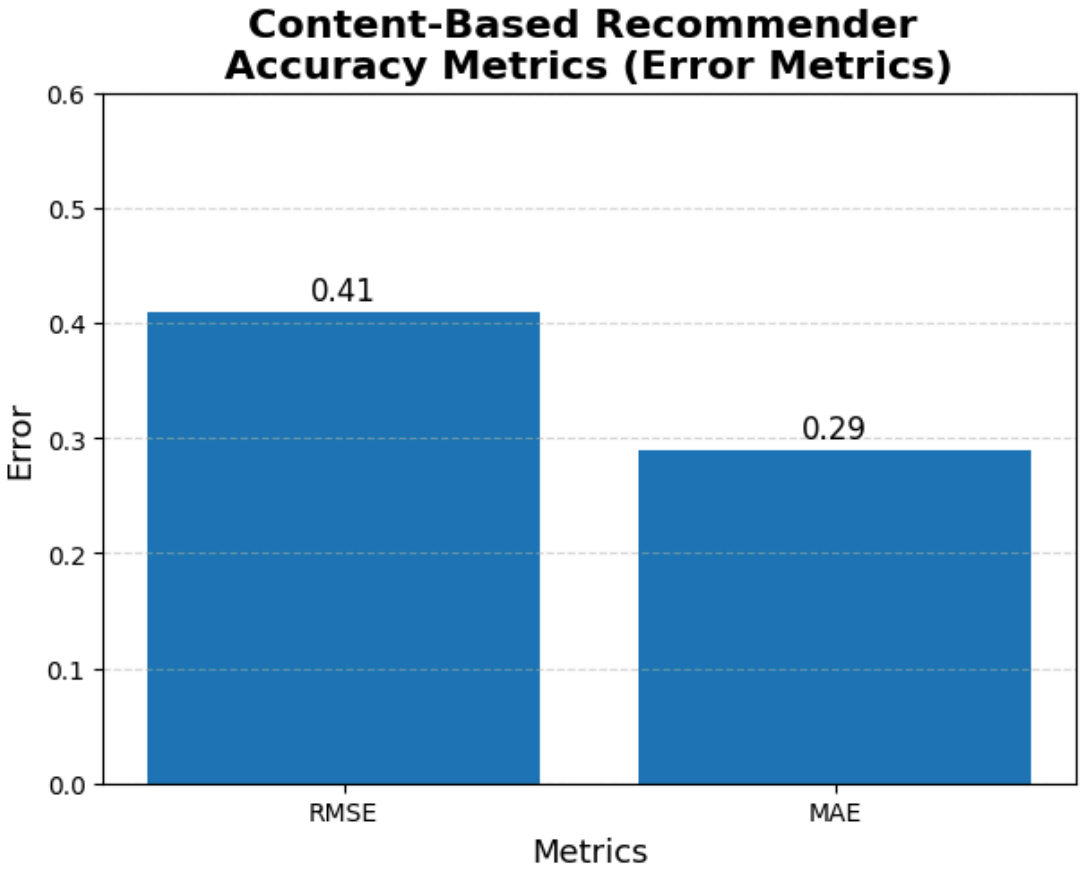

Fig 4.2: Error Metrics

4.4.2 Skill Difficulty Classifier Model Evaluation

This model was evaluated using a confusion matrix, a five-fod cross-validation analysis, precision, recall, and F1-score across all three classes: beginner, intermediate, and advanced.

4.4.2.1 Performance Metrics

The bar chart in Fig 4.3 displays the performance of the skill difficulty classifier model using performance metrics such as Precision, Recall, and F1-Score metrics for the classes: Advanced, Beginner, and Intermediate. Thus, the classifier performed very well in its classification ability for all skill difficulties with an equally balanced predictive ability.

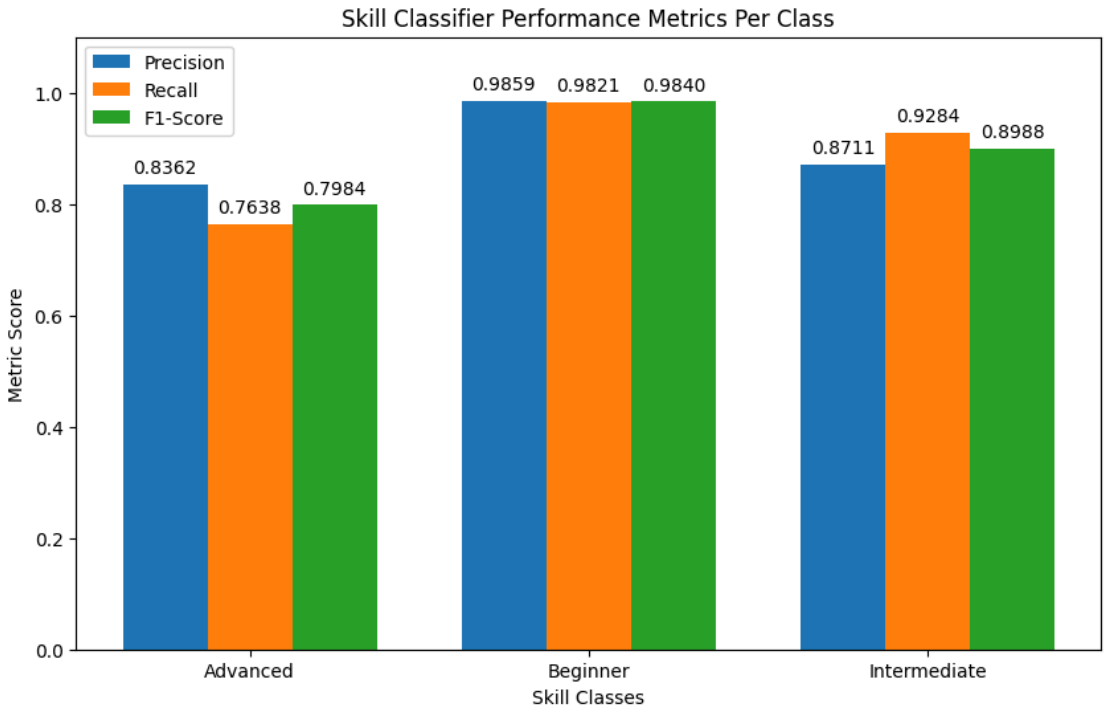

Fig 4.3: Skill Classifier Performance Metrics

4.4.2.2 Confusion Matrix

The confusion matrix in Fig 4.4 reveals how well the classifier performs for each skill category through comparison of actual skill label output to predicted skill label output. The classifier performs well in all skill categories but is still slightly influenced by dataset imbalance within the minority advanced class.

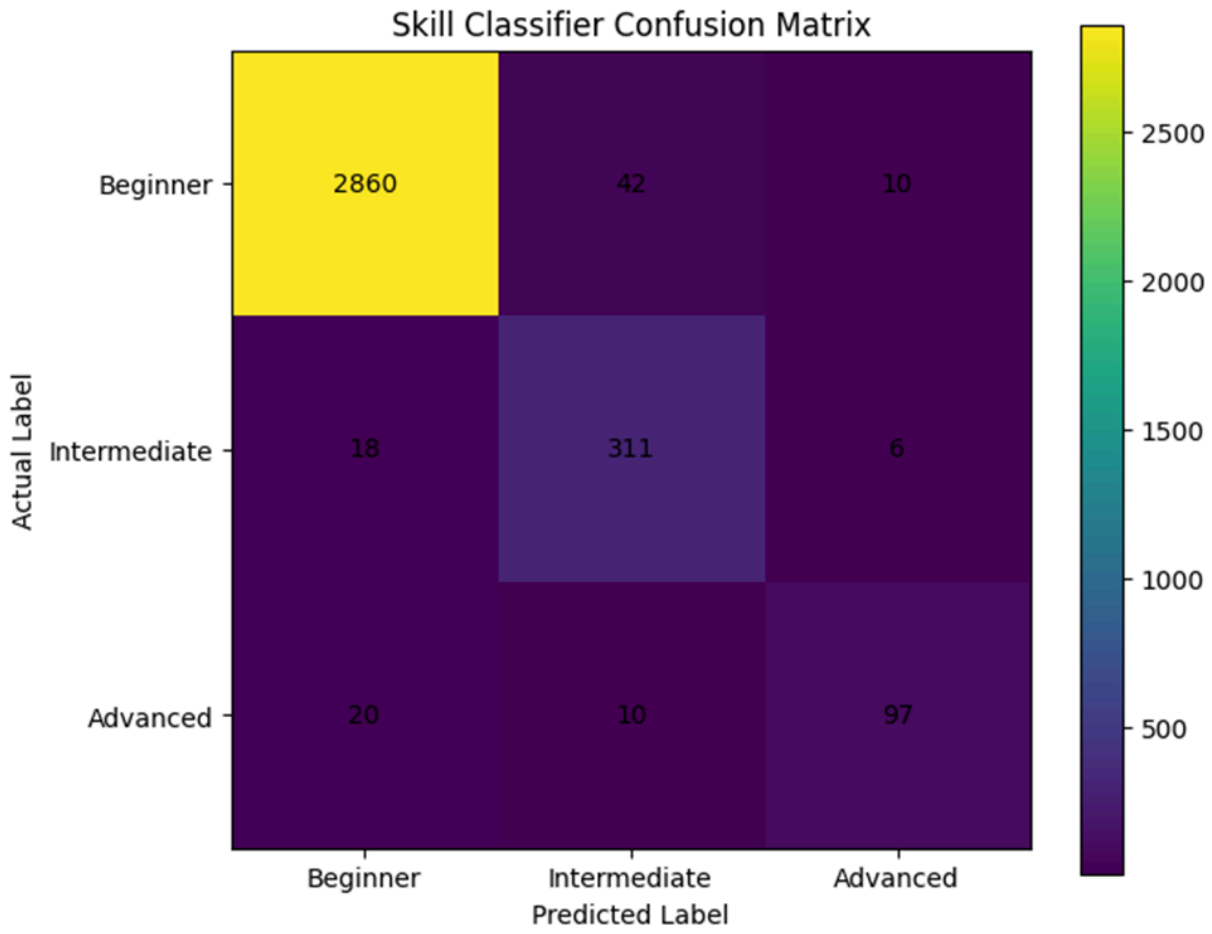

Fig 4.4: Confusion Matrix

4.4.2.3 Cross-Validation Analysis

The 5-fold cross-validation results illustrated in Fig 4.5 indicate that the skill classifier model maintained stable and consistent performance across different training and validation partitions of the dataset. The cross-validation results validate the robustness, reliability, and generalization capability of the proposed classifier model.

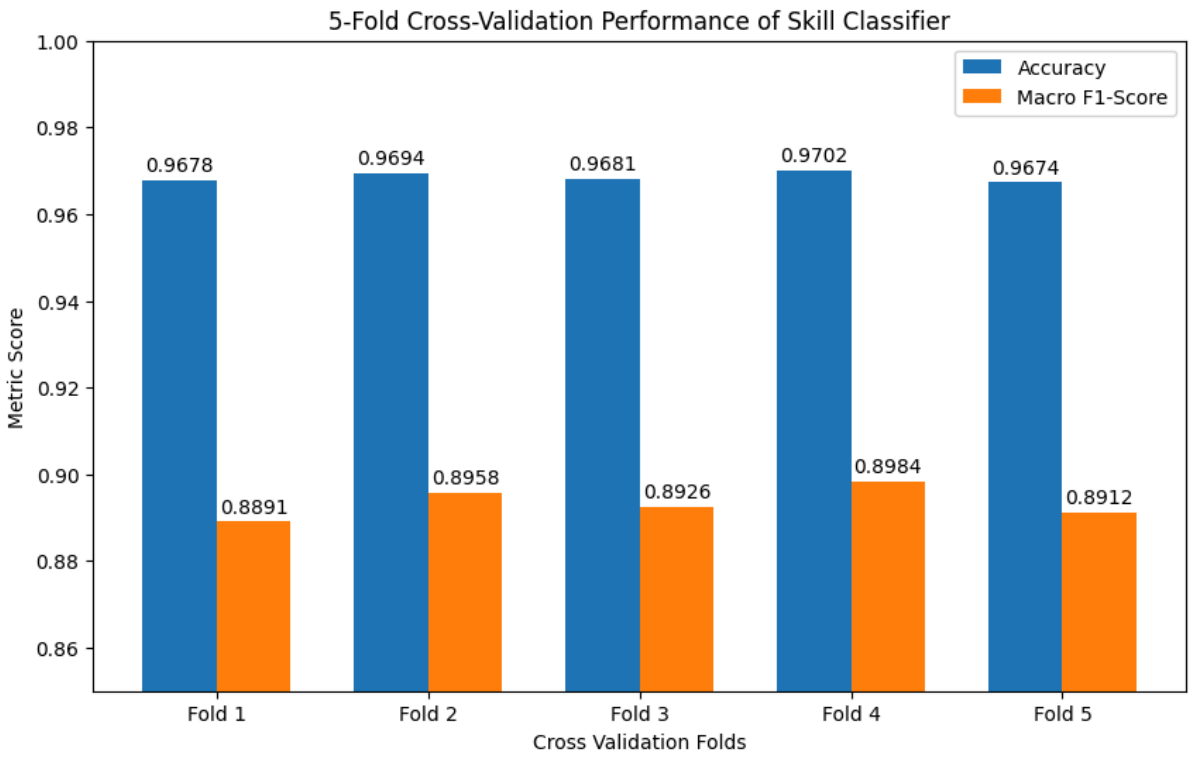

Fig 4.5: 5-Fold Cross-Validation Performance Chart

4.4.3 Singular Value Decomposition (SVD) Personalization Model

In recommendation systems, an AUC (Area Under the ROC Curve) score above 0.5 indicates that the model performs better than random guessing. Therefore, the obtained value of 0.6062 demonstrates that the personalization model has learned meaningful behavioral patterns from the interaction data. The model also achieved an accuracy of 62.21 which means that the personalization system correctly predicts user preference outcomes in approximately six out of every ten interactions. The model has a relatively low prediction error when estimating user preference scores, indicating stable recommendation predictions. The metric is visualized in Fig 4.6.

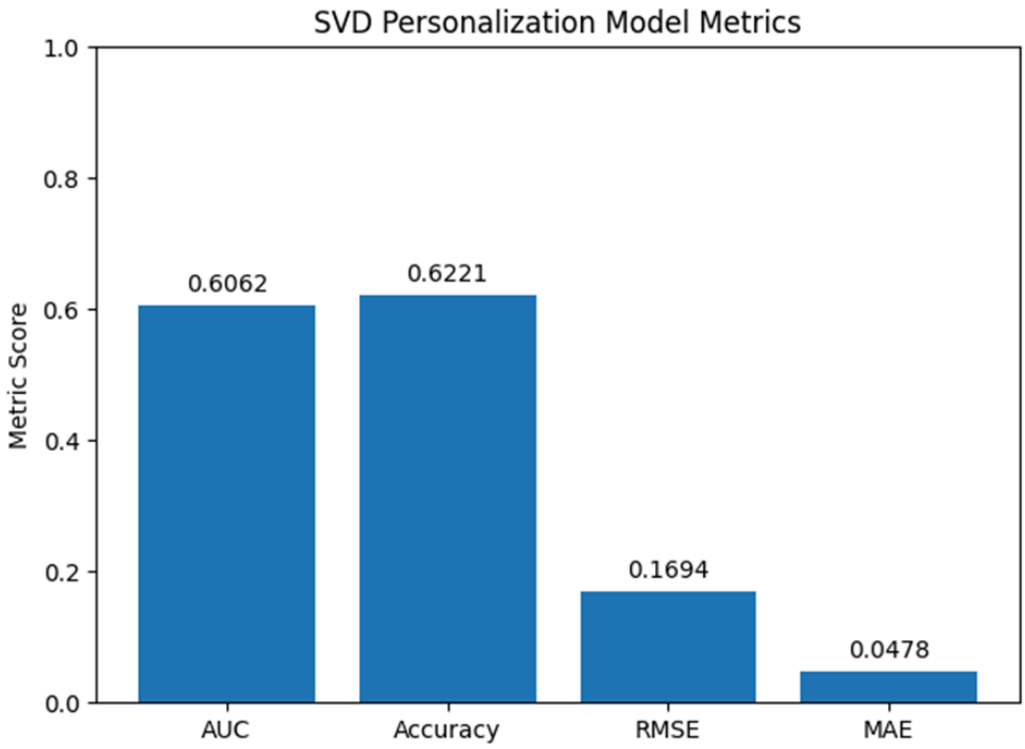

Fig 4.6: SVD Performance Testing

4.4.4 Concept Dependency Graph (CDG) Evaluation Metrics

Because the Concept Dependency Graph (CDG) is rule-based and not algorithmic, it was evaluated based on correctness, coverage, ordering, and accuracy. A visualization of the output depicted in Fig 4.7.

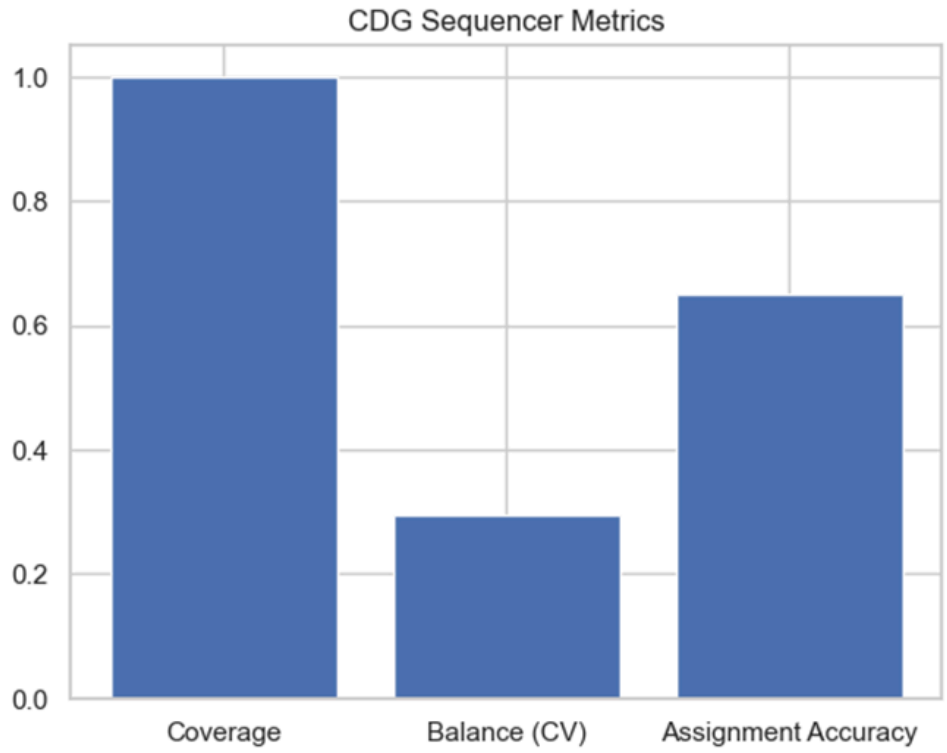

Fig 4.7 CDG Structural and Functional Evaluation Metric

4.5 System Testing

An automated test suite of 8 test files was implemented across all three tiers that includes unit tests, integration tests, and end-to-end routing tests.

The frontend tests involved the use of Vitest 1.5 with React Testing Library and Mock Service Worker (MSW) for API mocking at the network level. The unit tests involved the ThemeContext test(default theme, toggle function, and localStorage), the VideoCard test (metadata render, format video duration and views, and callback calls), and the CreateWatchlist test (form validation, loading states, and error rendering). The integration tests performed on the Dashboard includes rendering of a seed watchlist on live server, displaying API-loaded watchlist upon successful fetch, and navigation callback firing. The integration tests of ViewWatchlist involved checking the POST payload of thumbs-up and thumbs-down ratings, toggle-off functionality on multiple click events of rating, and PATCH call when a video is marked as watched.

The backend tests were performed using Jest 29 with Supertest to perform a complete HTTP integration test. The authentication middleware tests ensure that a valid token gets passed with a

user present. It further checks to ensure that a missing or invalid token returns a 401 status, and that the user_id property in rating submission always comes from the token itself and not from other sources to avoid any possible user ID manipulation attacks. The watchlist controller tested for 400 status code to ensure there are no missing parameters, a 201 status code if the payload is valid. The tests also check to ensure that the absence of the ML service doesn't hinder the watchlist creation (fault isolation).

ML service tests were implemented using pytest 8.1 with pytest-mock. Unit tests for preprocess.py covered 22 granular assertions across clean_text(), infer_difficulty(), infer_skill(), infer_industry(), and load_and_label_dataset(). The SVD recommender test suite included a mathematically grounded validity test that verified the model's training RMSE was lower than the global-mean baseline which is the minimum criterion. FastAPI integration tests covered all five endpoints, verifying correct HTTP status codes, response structure, field presence, and error handling. Pipeline integration tests verified the four-stage chain produced coherent output and that an empty user_id correctly bypassed personalisation.

4.5.3 User Interface Testing

The user interface testing was done manually by performing each workflow in YouLearn: creating an account, setting up the profile, creating a watchlist with each of the supported skills on the Concept Dependency Graph (CDG), checking the created watchlist for curriculum staging, marking videos as watched and observing the progression of the progress bar, giving thumbs up and thumbs down ratings, modifying the watchlist by adding a video through its URL and seeing the preview of metadata, deleting a video, saving the changes, and returning to the dashboard. Dark mode / light mode was checked and all CSS tokens were observed to render correctly in both modes.

4.6 Results Presentation and Analysis

4.6.1 Sample Output and Interface Screens

Upon creating a watchlist for 'Data Analysis'' at 'Advanced' level and using "Business" as a tool, the system returns a set of videos ordered into a curriculum sequence visible in the ViewWatchlist screen. Each VideoCard displays the video metadata such as thumbnail, title,

channel name, formatted view count, publication year, and ISO 8601 duration rendered as a human-readable string (e.g., '2:30:00'). The progress bar at the top of the screen reflects the proportion of watched videos in real time. Fig 4.8 depicts the generated videos of Data Analysis at Advanced Level and Business as a Tool

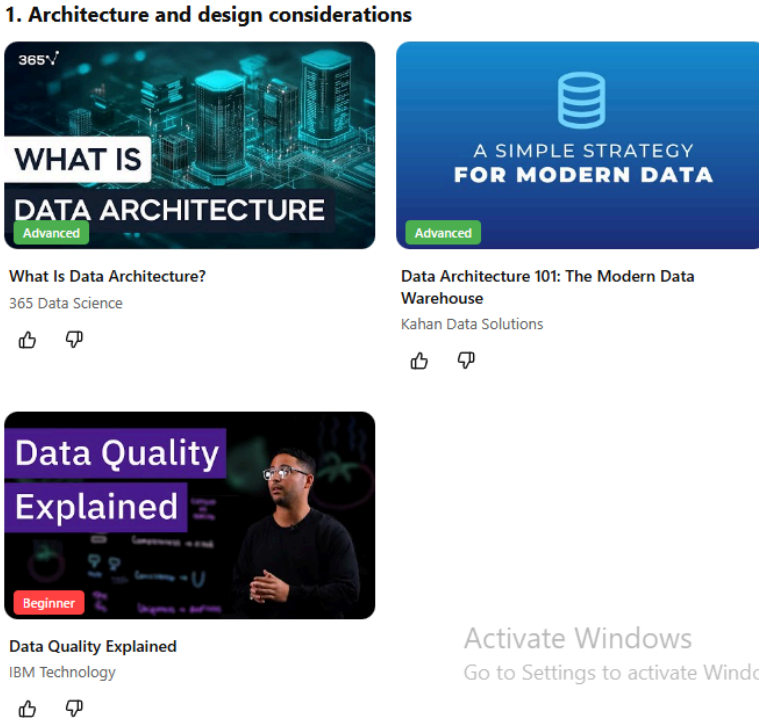

Fig 4.8 Sample Screen Output

The Dashboard screen in Fig 4.9 displays all watchlists as cards in a responsive grid, each showing the watchlist name, a colour-coded skill level indicator (green for beginner, blue for intermediate, red for advanced), a thin progress bar, and the watched/total video ratio. The mock seed watchlist 'Python for Data Science with 40% progress is always visible at the top of the grid to demonstrate the progress tracking feature before any real watchlists are created.

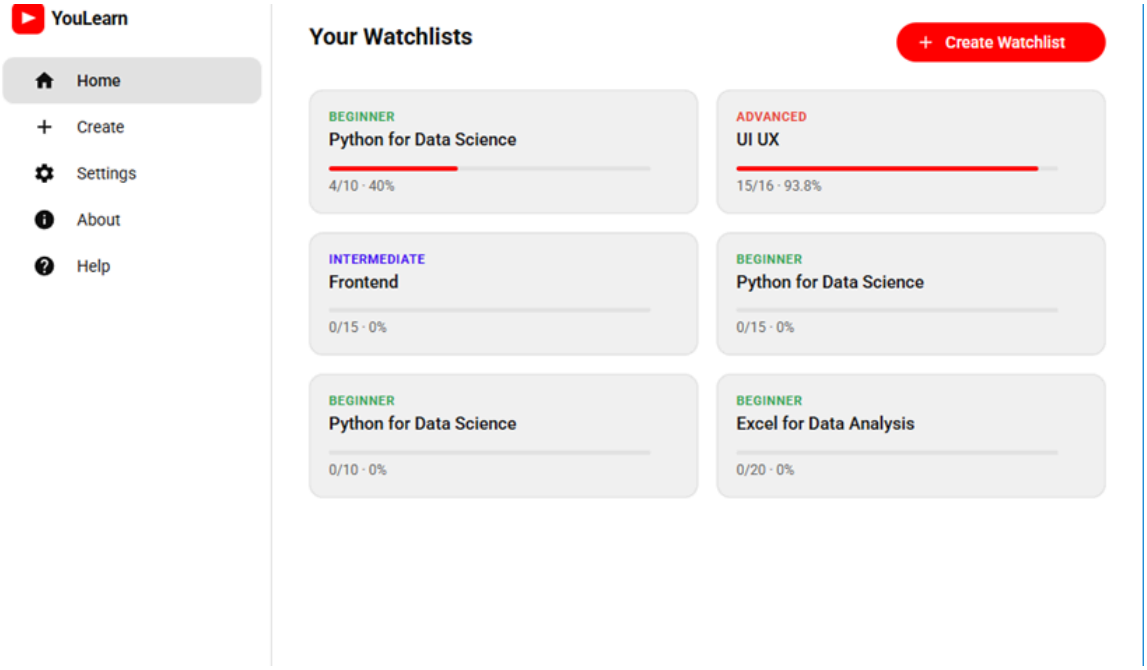

Fig 4.9 Sample Dashboard of Collected Skills

4.6.2 Discussion of Results

The YouLearn system successfully fulfils its primary objective of generating personalised, curriculum-ordered YouTube skill paths from a user's stated skill, difficulty level, and industry context. The four-stage recommendation pipeline of content retrieval, personalisation re-ranking, concept dependency graph sequencing, and trimming consistently returns on-topic, difficulty-appropriate videos in a pedagogically logical order for all six supported skill domains.

The main limitation of the current implementation is that the SVD personalisation layer provides limited benefit until a sufficient number of real user ratings have accumulated in the ratings table.

Chapter Five
Summary, Conclusion, and Recommendations

5.0 Introduction

The YouLearn research project concludes with this chapter, which highlights the most important findings, conclusions, and recommendations for the future. This chapter first gives a structured summary of the work that has been conducted from the problem definition to the implementation as well as evaluation of the given problem in the earlier chapters. Further, it states the conclusion that can be drawn from the results concerning the stated objectives of the project. Lastly, it concludes with the recommendations on the improvement, augmentation, and long-term sustenance of the YouLearn system.

5.1 Summary of the Study

Chapter One provided the background to the study, a discussion of the aim and objectives, significance, scope and limitations. The absence of a pedagogically-structured pathway for skill learning was the major problem. The reason for this is that at present, the YouTube search and recommendation algorithms are designed to optimize for popularity and watch-time engagement metrics rather than pedagogical appropriateness. Thus, a learner finds it hard to recognise a coherent progression from the learning of basic concepts to advanced techniques. The project aimed to implement YouLearn, a personalized web-based system that automatically creates YouTube skill-path watchlists that are ordered by curriculum for a user's selected skill, difficulty level and industry context.

Chapter Two evaluated the literature on recommender systems, e-learning platforms, and personalised learning technology. The paper investigated first the content-based filtering and the collaborative filtering as complementary recommendation paradigms, problem each and the the use of concept dependency graphs in curriculum-aware learning systems.

The recommender systems for structured online courses are well studied. However, to our knowledge, no published system applies structured sequencing to the unstructured domain of YouTube educational content. YouLearn directly tackles this issue.

Chapter Three analyzed the existing YouTube recommendation system and proposed a design methodology for YouLearn. The development of the system was a fully structured process since

all requirements were known. The chapter details technical guidelines of the 4-tier system architecture of React frontend, express backend, FastAPI ML microservice and PostgreSQL database via Supabase. It also included the use case diagrams, Data Flow Diagrams (level 0 and level 1 diagrams), Entity Relationship Diagram (ERD), and the system architecture diagrams used to model users of the proposed system.

Chapter Four describes how the implementation and evaluation was done. From data preprocessing to training and evaluating the three machine learning algorithms and the rule-based concept dependency graph. It also showed the tests carried out for unit, integration and end to end routing across the three layers of services using an 8-file test suite.

5.3 Conclusion

The aim of this study was to develop YouLearn: A Personalized YouTube-Based Skill-Path Recommender System using Machine Learning. To achieve the stated aim, four objectives were set and below is how each of the set objectives have been met.

1. Identify the challenges associated with learning a new skill on YouTube: The study started by reviewing several literature works on the surrounding topics of personalized recommender systems as used in educational backgrounds. This review laid the groundwork for the design of the system.
2. Design a smart approach to YouTube personalized learning: The system was modeled and designed using Structured System Design Methodology.
3. Develop the approach into a model that personalizes a learning system: To achieve this, a total of three machine learning models and a rule-based keyword graph was trained. Below are the models:
    - A Content-Based Recommender that utilizes TF-IDF cosine similarity retrieval system and a skill alias relevance filter for topic filtering fulfilled the content-based filtering objective.
    - A skill difficulty classifier that classifies videos into two broad levels of “Beginners”, “Intermediate”, and “Advanced.” The classification of skills was completed through a pipeline comprising of TF-IDF and Logistic Regression

which assigns labels indicating difficulty levels to videos. This enables the selection of content appropriate to the respective difficulty level.

- Curriculum structuring was achieved through the Concept Dependency Graph that reliably organizes videos into learning phases for eleven skill domains.
- A personalization model trained with a SVD collaborative filter that re-ranks recommended content based on predicted user preferences. This filter was trained on Coursera proxy data and will subsequently adopt live YouLearn user ratings.

4. Testing and evaluating the model to determine its accuracy: Several evaluation metrics were employed to test the performance of all the component models involved in the development of YouLearn.

5.4 Limitations of the System

YouLearn was not fully developed due to two major limitations.

- Lack of Dataset Availability: The effectiveness of the personalisation layer is limited owing to the absence of sufficient native user interaction data during evaluation as the system was not yet trained with sufficient rating dataset for SVD model retraining on YouLearn data.
- The Concept Dependency Graph has only twelve predefined skill domains with a default graph for all other skills giving less precise sequencing than a domain-specific graph.

The limitations above are recognized as future work and do not undermine the system's ability to produce curriculum-ordered, topic-specific skill paths for the eleven covered domains.

5.4 Recommendations

In light of the results of this study, the recommendations for improvement, adoption, and sustained expansion of YouLearn are:

1. A retraining pipeline should be implemented on a schedule, for instance, as a weekly Supabase Edge Function that is deployed to retrain the SVD personalisation model using the ratings that users of the system have left. User rating data will gradually replace the Coursera proxy as its user base grows, increasing personalization accuracy. YouLearn's

recommendation quality will therefore become an important differentiating factor from generic YouTube search.

2. Use of Natural Language Processing (NLP) for curriculum structuring: The present system uses a Concept Dependency Graph for curriculum structuring. Future studies need to analyze how the prerequisite chains for any arbitrary query can be created automatically using natural language processing. This would remove the existing limit on domain, so YouLearn would be able to sequence content for any skill user wanted.
3. The current system requires a learner to click the link or manually toggle the eye icon on every video to indicate the video has been watched. The future system will use the YouTube iFrame Player API to automatically mark a video as “watched” when it has been seen to a certain configurable percentage (might be 80%).

References

Berg, G.A., Simonson, M. (2025, April 3). distance learning. Encyclopedia Britannica. https://www.britannica.com/topic/distance-learning

Cheung, S.K.S., Kwok, L.F., Phusavat, K. et al. Shaping the future learning environments with smart elements: challenges and opportunities. Int J Educ Technol High Educ 18, 16 (2021). https://doi.org/10.1186/s41239-021-00254-1

Rahmatika, D. P., et al. (2021). Accessibility and flexibility of YouTube as a learning tool for student independence. Education and Information Technologies, 26(4), 4567-4589.

YouTube Press (2026). https://blog.youtube/press/

Joy, E. H., et al. (2021). E-learning effectiveness in self-directed learning: A review. Distance Education, 42(3), 345-362.

YouTube. (2024). YouTube Data API - Quota Calculator. Retrieved from https://developers.google.com/youtube/v3/getting-started#quota

Muniraja, P., & Satapathy, S. M. (2026). KGERA: knowledge graph enhanced reasoning architecture for recommendation systems. Scientific Reports.

Ko, H., Lee, S., Park, Y., & Choi, A. (2022). A Survey of Recommendation Systems: Recommendation Models, Techniques, and Application Fields. Electronics, 11(1), 141.

Khalifeh, F., Santiago, R., & Palau, R. (2026). Redefining personalized learning in the artificial intelligence era. Smart Learning Environments.

Aliberti, L., D'Aniello, G., & Gaeta, M. (2026). Situation-aware recommender systems: a systematic review and framework for trustworthy recommendations. Artificial Intelligence Review.

Kowalczyk, R. et al. (2021). A knowledge-driven digital nudging approach to recommender systems. Expert Systems with Applications, 181, 115170.

Tetzlaff, L., Schmiedek, F., & Brod, G. (2021). Developing Personalized Education: A Dynamic Framework. Educational Psychology Review, 33, 863–882.

Bernacki, M. L., Greene, M. J., & Lobczowski, N. G. (2021). A Systematic Review of Research on Personalized Learning. Educational Psychology Review.

Zawacki-Richter, O. et al. (2021). Systematic review of research on MOOCs. Frontiers in Education.

Zhang, Q., Lu, J., & Jin, Y. (2021). Artificial intelligence in recommender systems. Complex & Intelligent Systems, 7, 439–457.

Kohler, S., & Dietrich, T. C. (2021). Potentials and Limitations of Educational Videos on YouTube for Science Communication. Frontiers in Communication.

Alfaifi, Y. H. (2024). Recommender systems applications: Data sources, features, and challenges. Information, 15(10), 660. https://doi.org/10.3390/info15100660

Lian, J., Zhou, X., Zhang, F., Chen, Z., Xie, X., & Sun, G. (2020). xDeepFM: Combining explicit and implicit feature interactions for recommender systems. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 1754–1763. https://doi.org/10.1145/3219819.3220023

Errakha, K., Samih, A., Marzouk, A., & Krari, A. (2025). Recommender systems in e-learning: Trends, challenges, and future directions. Journal of Theoretical and Applied Information Technology, 103(7). https://www.jatit.org/volumes/Vol103No7/27Vol103No7.pdf

Ramadhan, F., & Musdholifah, A. (2021). Online learning video recommendation system based on course and syllabus using content-based filtering. IJCCS (Indonesian Journal of Computing and Cybernetics Systems), 15(3), 265–274. https://doi.org/10.22146/ijccs.65623

Xu, G., Jia, G., Shi, L., & Zhang, Z. (2021). Personalized course recommendation system fusing with knowledge graph and collaborative filtering. Computational Intelligence and Neuroscience, 2021, Article 9590502. https://doi.org/10.1155/2021/9590502

Troussas, C., & Krouska, A. (2023). Path-based recommender system for learning activities using knowledge graphs. Information, 14(1), 9. https://doi.org/10.3390/info14010009

Zhang, X., Liu, S., & Wang, H. (2023). Personalized learning path recommendation for e-learning based on knowledge graph and graph convolutional network. International Journal of Software Engineering and Knowledge Engineering, 33(1), 109–131. https://doi.org/10.1142/S0218194022500681

Ma, H., Tang, Y., Zhang, X., Zhu, H., Huang, P., & Zhang, H. (2023). Learning resource recommendation via knowledge graphs and learning style clustering. Journal of Intelligent & Fuzzy Systems, 44(3). https://doi.org/10.3233/JIFS-222627

Gu, R. (2025). Personalized learning path based on graph attention mechanism deep reinforcement learning research on recommender systems. International Journal of Learning Technology, 20(1). https://doi.org/10.1177/14727978241313260

Yuan, H., Liu, L., Zhang, Y., Wang, G., He, A., & Zhang, F. (2025). Research on MOOCs course recommendation system based on hybrid attention mechanism in frequency and time domain. PLOS ONE. https://doi.org/10.1371/journal.pone.0338738

Zheng, Y., Wang, D., Zhang, J., Li, Y., Xu, Y., Zhao, Y., & others. (2024). A unified framework for personalized learning pathway recommendation in e-learning contexts. Education and Information Technologies, 30(6), 7911–7948. https://doi.org/10.1007/s10639-024-13045-8

Yun, Y., Dai, H., An, R., Zhang, Y., & Shang, X. (2024). Doubly constrained offline reinforcement learning for learning path recommendation. Knowledge-Based Systems, 284, 111242. https://doi.org/10.1016/j.knosys.2024.111242

Zhang, F., Feng, X., & Wang, Y. (2024). Personalized process-type learning path recommendation based on process mining and deep knowledge tracing. Knowledge-Based Systems, 303, 112431. https://doi.org/10.1016/j.knosys.2024.112431

Li, S., Liu, X., Tang, X., Chen, X., & Pu, J. (2024). MLKT4Rec: Enhancing exercise recommendation through multitask learning with knowledge tracing. IEEE Transactions on Computational Social Systems, 12, 1458–1472. https://doi.org/10.1109/TCSS.2024.3361124

Lin, X., Chen, X., Song, L., Liu, J., Li, B., & Jiang, P. (2023). Tree-based progressive regression model for watch-time prediction in short-video recommendation. Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23). https://doi.org/10.1145/3580305.3599919

Kaladiya, M., & Patel, P. (2025). Leveraging sentiment analysis for enhanced educational video recommendations on YouTube-Easyfy. In Proceedings of the International Conference on Computer Application in Social and Behavioral Sciences. Springer. https://doi.org/10.1007/978-981-96-2724-0_3

Chanaa, A., & El Faddouli, N. (2024). Prerequisites-based course recommendation: Recommending learning objects using concept prerequisites and metadata matching. Smart Learning Environments, 11, 16. https://doi.org/10.1186/s40561-024-00301-0

Mohd Zamani, W. B. (2025). Educational video recommender system for computer science students using content-based filtering. Bachelor's thesis, Universiti Teknologi MARA, Terengganu.

Carvalho, H. C. F. B., Dorça, F. A., Pitanguí, C. G., & Andrade, A. V. (2024). Improving the educational experience on YouTube: A machine learning approach to classifying

and recommending educational videos. Revista de Gestão e Secretariado, 15(4). https://doi.org/10.7769/gesec.v15i4.3587

Yun, Y., Dai, H., Zhang, Y., Shang, X., & Li, Z. (2025). Simulation of personalised English learning path recommendation system based on knowledge graph and deep reinforcement learning. Scientific Reports, 15. https://doi.org/10.1038/s41598-025-17918-x

Kim, Y. K., & Kim, M. H. (2022). Design and implementation of YouTube-based educational video recommendation system. Journal of The Korea Society of Computer and Information, 27(5), 37–45.

Cristos Goodrow (2021) YouTube Blog https://blog.youtube/inside-youtube/on-youtubes-recommendation-system/

Appendix

youlearn/ml_service/train_classifier.py

This file trains the skill classifier on the cleaned dataset using Logistic Regression.

```
import os
import sys
import json
import joblib
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score, f1_score
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline as ImbPipeline

ROOT = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
sys.path.insert(0, ROOT)
from src.visualization import plot_confusion

DATA = os.path.join(ROOT, "data", "processed", "classifier_dataset.csv")
OUT = os.path.join(ROOT, "models", "skill_classifier.pkl")
REPORTS = os.path.join(ROOT, "reports")


def main():
    df = pd.read_csv(DATA).dropna(subset=["text", "label"])
    print(f"[classifier] training on {len(df)} rows")
    print("class distribution:")
    print(df["label"].value_counts().to_string())

    X_train, X_test, y_train, y_test = train_test_split(
        df["text"], df["label"], test_size=0.2, random_state=42, stratify=df["label"]
    )

    pipe = ImbPipeline([
        ("tfidf", TfidfVectorizer(stop_words="english", ngram_range=(1, 2), max_features=20000)),
        ("smote", SMOTE(random_state=42, k_neighbors=3)),
```

```
        ("clf", LogisticRegression(max_iter=1000, class_weight="balanced")),
    ])

    pipe.fit(X_train, y_train)
    y_pred = pipe.predict(X_test)

    acc = accuracy_score(y_test, y_pred)
    f1m = f1_score(y_test, y_pred, average="macro")
    print(f"[classifier] accuracy   = {acc:.4f}")
    print(f"[classifier] macro F1   = {f1m:.4f}")
    print(classification_report(y_test, y_pred, digits=4))

    classes = ["beginner", "intermediate", "advanced"]
    cm = confusion_matrix(y_test, y_pred, labels=classes)
    plot_confusion(cm, classes, REPORTS)

    os.makedirs(os.path.dirname(OUT), exist_ok=True)
    joblib.dump(pipe, OUT)

    os.makedirs(REPORTS, exist_ok=True)
    metrics_path = os.path.join(REPORTS, "metrics.json")
    metrics = {}
    if os.path.exists(metrics_path):
        with open(metrics_path) as f:
            metrics = json.load(f)
    metrics["classifier"] = {"accuracy": acc, "macro_f1": f1m}
    with open(metrics_path, "w") as f:
        json.dump(metrics, f, indent=2)

    print(f"[classifier] saved -> {OUT}")


if __name__ == "__main__":
    main()
```

youlearn/ml_service/train_personalization.py

This file trains a TruncatedSVD-based personalization model and evaluates through AUC and accuracy.

```
import os
import sys
```

```
import json
import joblib
import numpy as np
import pandas as pd
from scipy.sparse import csr_matrix
from sklearn.decomposition import TruncatedSVD
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score, accuracy_score

ROOT = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
sys.path.insert(0, ROOT)
from src.personalization import PersonalizationModel

DATA = os.path.join(ROOT, "data", "processed", "personalization_dataset.csv")
OUT = os.path.join(ROOT, "models", "personalization.pkl")
REPORTS = os.path.join(ROOT, "reports")


def main():
    if not os.path.exists(DATA):
        print("[personalization] dataset not found — run prepare_datasets.py first")
        return

    df = pd.read_csv(DATA)
    print(f"[personalization] training on {len(df)} ratings")
    print(f"like rate = {df['rating'].mean():.3f}")

    train, test = train_test_split(df, test_size=0.2, random_state=42, stratify=df["rating"])

    users = train["user_id"].unique()
    items = train["item_id"].unique()
    user_index = {u: i for i, u in enumerate(users)}
    item_index = {it: i for i, it in enumerate(items)}

    rows = train["user_id"].map(user_index).values
    cols = train["item_id"].map(item_index).values
    vals = train["rating"].values.astype(float)
    mat = csr_matrix((vals, (rows, cols)), shape=(len(users), len(items)))
```

```
n_components = min(50, min(mat.shape) - 1)
svd = TruncatedSVD(n_components=n_components, random_state=42)
user_factors = svd.fit_transform(mat)
item_factors = svd.components_.T

model = PersonalizationModel(
    user_factors=user_factors,
    item_factors=item_factors,
    user_index=user_index,
    item_index=item_index,
)

# Evaluate on test set
test = test[test["user_id"].isin(user_index) & test["item_id"].isin(item_index)]
scores, labels = [], []
for _, r in test.iterrows():
    scores.append(model.predict_score(r["user_id"], r["item_id"]))
    labels.append(r["rating"])

if scores:
    preds = [1 if s >= 0.5 else 0 for s in scores]
    auc = roc_auc_score(labels, scores)
    acc = accuracy_score(labels, preds)
    print(f"[personalization] AUC      = {auc:.4f}")
    print(f"[personalization] accuracy = {acc:.4f}")
else:
    auc, acc = None, None

os.makedirs(os.path.dirname(OUT), exist_ok=True)
joblib.dump(model, OUT)

os.makedirs(REPORTS, exist_ok=True)
metrics_path = os.path.join(REPORTS, "metrics.json")
metrics = {}
if os.path.exists(metrics_path):
    with open(metrics_path) as f:
        metrics = json.load(f)
metrics["personalization"] = {"auc": auc, "accuracy": acc}
with open(metrics_path, "w") as f:
```

```
        json.dump(metrics, f, indent=2)

    print(f"[personalization] saved -> {OUT}")


if __name__ == "__main__":
    main()
```

youlearn/ml_service/train_recommender.py

This file trains the TF-IDF content-based recommender on the cleaned dataset and saves a TfidfRanker that the API loads at startup.

```
import os
import sys
import joblib
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

ROOT = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
sys.path.insert(0, ROOT)
from src.recommender import TfidfRanker

DATA = os.path.join(ROOT, "data", "processed", "content_dataset.csv")
OUT = os.path.join(ROOT, "models", "recommender.pkl")
REPORTS = os.path.join(ROOT, "reports")


def evaluate(vectorizer, df, k=5, sample=200, seed=42):
    """
    Self-similarity evaluation: for each sampled video, take the top-k most
    similar by TF-IDF cosine; check the proportion sharing the same domain.
    This is a standard intra-list relevance proxy.
    """
    rng = np.random.default_rng(seed)
    n = min(sample, len(df))
    idx = rng.choice(len(df), size=n, replace=False)
```

```
    mat = vectorizer.transform(df["text"].fillna(""))
    hits = 0
    total = 0
    for i in idx:
        sims = cosine_similarity(mat[i], mat).flatten()
        sims[i] = -1
        top = sims.argsort()[::-1][:k]
        domain = df.iloc[i]["domain"]
        hits += sum(1 for j in top if df.iloc[j]["domain"] == domain)
        total += k
    return hits / total


def main():
    df = pd.read_csv(DATA).dropna(subset=["text"])
    print(f"[recommender] training on {len(df)} rows")

    vec = TfidfVectorizer(stop_words="english", ngram_range=(1, 2), max_features=30000)
    vec.fit(df["text"].fillna(""))

    ranker = TfidfRanker(vec)
    os.makedirs(os.path.dirname(OUT), exist_ok=True)
    joblib.dump(ranker, OUT)

    p_at_5 = evaluate(vec, df, k=5)
    p_at_10 = evaluate(vec, df, k=10)
    print(f"[recommender] domain precision@5  = {p_at_5:.4f}")
    print(f"[recommender] domain precision@10 = {p_at_10:.4f}")

    os.makedirs(REPORTS, exist_ok=True)
    import json
    metrics_path = os.path.join(REPORTS, "metrics.json")
    metrics = {}
    if os.path.exists(metrics_path):
        with open(metrics_path) as f:
            metrics = json.load(f)
    metrics["recommender"] = {"precision@5": p_at_5, "precision@10": p_at_10}
    with open(metrics_path, "w") as f:
```

```
        json.dump(metrics, f, indent=2)
    print(f"[recommender] saved -> {OUT}")


if __name__ == "__main__":
    main()
```