Course Recommendation System

Date: 29 Sept 2024

Introduction

This document aims to provide a comprehensive overview of the design and architecture of the Course Recommendation System. It details the software components, design considerations, and the development approach for a system designed to offer personalized course recommendations to users based on their preferences and historical data.

System Overview

The Course Recommendation System is designed to interact with users through a web interface, allowing them to view, search, and enroll in courses that are recommended based on their interests and past activities. The system integrates a backend developed in Django, which handles data management, user authentication, and recommendation logic, with a frontend developed in React, providing a responsive user experience.

Design Considerations

Assumptions and Dependencies:

- **Assumptions:**
    - Users will have continuous internet access to interact with the system.
    - The recommendation engine's accuracy is dependent on the availability of sufficient historical user data.
- **Dependencies:**
    - The system relies on the Django framework for the backend and React for the frontend.
    - Database operations assume the availability of a SQLite/PostgreSQL server.

General Constraints

- The system must operate under the constraint of handling multiple users concurrently without degradation of performance.

Goals and Guidelines

- **Goals:**
    - To provide accurate course recommendations that enhance the learning experience.
    - To ensure the system is scalable and maintainable.
- **Guidelines:**
    - All code must adhere to the DRY principle to avoid redundancy.
    - The system should be easy to update and maintain.

Development Methods

- The project will utilize Agile development methodologies with two-week sprints, allowing iterative and incremental development.

Architectural Strategies

Strategy 1: Microservices Architecture

- The backend will be divided into microservices, each handling a specific part of the system's functionality (courses, users, recommendations).

Strategy 2: API-First Development

- Development will prioritize the creation of RESTful APIs to ensure that the system's services are modular and can be easily accessed by the React frontend.

System Architecture

The system is divided into several key components, each responsible for a segment of functionality:

Component 1: User Management

- Handles user registration, authentication, and profile management.

Component 2: Course Management

- Manages all aspects of course data, including creation, modification, and retrieval.

Component 3: Recommendation Engine

- Generates personalized course recommendations using machine learning algorithms based on user data.

Detailed System Design

Module 1: User Authentication

- Implements security measures for user login processes.
- Integrates with OAuth for social media login capabilities.

Module 2: Course

- Allows users to search for courses based on various criteria.
- Supports enrollment processes and tracks user course interactions.

Glossary

- **API** (Application Programming Interface): A set of protocols for building and integrating application software.
- **Backend**: Server-side components of a computing architecture, typically handling data storage and processing.
- **CRS** (Course Recommendation System): Refers to the system designed to provide course recommendations to users.
- **Django**: A high-level Python web framework that encourages rapid development and clean, pragmatic design.
- **React**: A JavaScript library for building user interfaces, particularly for single-page applications where you need fast interaction.