Extracurricular Activity Recommender System

# 📌 Project Overview

This project aims to assist students in choosing the most suitable extracurricular activities through a web platform powered by machine learning. Built using Flask and Python, the application collects data via a student survey and uses it to train a machine learning model to make personalized recommendations.

# 🧠 Core Features

## 🔍 Intelligent Recommendation Engine

- Implements a machine learning model (Linear Regression, Perceptron, Logistic Regression, or Neural Network).  
- Trained on student-submitted survey data.  
- Entirely built without using high-level ML libraries like scikit-learn or Keras.

## 📄 Survey-Based Dataset

- Custom survey form used to collect student data.  
- Features reflect creativity, technical thinking, dexterity, and other competencies.

## 🌐 Web Interface

- User Authentication with multiple roles: Student, Teacher, Admin.  
- API Authentication with JWT tokens for domain-independent access.  
- Recommendation Portal: students get personalized suggestions.  
- Club Portal: view and apply to clubs; teachers manage applications.  
- Admin Dashboard: manage users and view AI performance.

# 🧩 Tech Stack

Backend: Python, Flask  
Frontend: Flask-Bootstrap  
Database: SQLAlchemy with Flask-Migrate  
Extensions Used:  
- Flask-WTF  
- Flask-SQLAlchemy  
- Flask-Login  
- Flask-JWT-Extended  
- Flask-Migrate  
- Flask-Mail  
Unit Testing: Custom test suite for major features  
AI: Custom ML algorithms using only numpy, pandas, matplotlib, and optionally torch/tensorflow only for tensor operations on GPU

# 🔑 API Authentication

The system provides a domain-independent API authentication mechanism using JWT (JSON Web Tokens):

## Endpoints

POST /api/auth/register: Register a new user  
Request: {"username": "user", "password": "pass"}  
Response: Access token, refresh token, and user info  
  
POST /api/auth/login: Authenticate a user  
Request: {"username": "user", "password": "pass"}  
Response: Access token, refresh token, and user info  
  
POST /api/auth/refresh: Refresh an access token  
Headers: Authorization: Bearer <refresh\_token>  
Response: New access token  
  
GET /api/auth/protected: Example protected endpoint  
Headers: Authorization: Bearer <access\_token>  
Response: User information

## Usage

- Register or login to get tokens  
- Include the access token in the Authorization header for protected requests  
- Use the refresh token to get a new access token when the current one expires  
- A test script (test\_api\_auth.py) is provided to demonstrate the API usage.

# 📎 Resources & Links

Lucidchart database diagram:  
https://lucid.app/lucidchart/3fb2b346-ffb2-45e3-a303-fe230c72d3a8/edit?beaconFlowId=B244F018134E58D0&page=0\_0&invitationId=inv\_bb9706e8-13b6-4bd8-83f2-b5d7f21de3a1#  
  
UserStories diagram:  
https://lucid.app/lucidchart/273f8f3c-1c2d-497a-9de6-ae01b3d9af88/edit?beaconFlowId=D0E264F813893273&page=0\_0&invitationId=inv\_fcc0dbc6-1c16-4d15-944d-c9a7c9bc2994#  
  
GitHub Repository:  
https://github.com/codingburgas/2425-11-g-pp-student-practices-assignment-team8.git