Project Proposal: Development of a Streaming Video Player with Local AI Movie Recommendation System

1. Project Title

Development of a Streaming Video Player Integrated with Local AI-Based Movie Recommendation Considering User Personal Interests

2. Project Overview

2.1 Background

Streaming video players have become essential in modern entertainment, with platforms like Netflix and YouTube dominating the market. However, many users seek personalized experiences that prioritize privacy and offline capabilities. Traditional recommendation systems often rely on cloud-based AI, raising concerns about data security and dependency on internet connectivity. This project proposes a software-based solution: a local streaming video player that incorporates an AI recommendation engine running entirely on the user's device. The system will analyze personal interests derived from user interactions, such as viewing history, ratings, and genre preferences, to suggest movies or videos from a local library or accessible streaming sources.

By leveraging open-source tools and machine learning libraries, this project will enable students to explore full-stack development, AI integration, and user-centric design. It addresses the growing demand for privacy-focused applications while providing a practical tool for media consumption.

2.2 Problem Statement

Existing video players and recommendation systems face issues such as:

- **Privacy Concerns**: Cloud-based recommendations track user data extensively.
- **Offline Limitations**: Many systems require constant internet access for suggestions.
- **Generic Recommendations**: Often fail to deeply incorporate personal interests beyond basic demographics.

A local AI-driven player can mitigate these by processing data on-device, ensuring recommendations are tailored and secure.

2.3 Significance

This project will provide an educational platform for learning AI, software engineering, and UI/UX design. It could evolve into an open-source tool, benefiting users who prefer local media management, and contribute to research in edge AI for personalized content delivery.

3. Objectives

3.1 Primary Objective

To design and implement a streaming video player with an integrated local AI recommendation system that personalizes suggestions based on user interests.

3.2 Secondary Objectives

- Collect and analyze user data (e.g., watch history, ratings) to build personalized profiles.
- Develop a recommendation algorithm using machine learning techniques like collaborative filtering or content-based methods.
- Create a user-friendly interface for video playback, library management, and interaction with recommendations.
- Ensure the system operates offline, with optional integration for fetching metadata from public APIs.
- Evaluate the recommendation accuracy and system performance through user simulations and metrics like precision/recall.

4. Methodology

4.1 System Architecture

The system will include:

- **Video Player Module**: Handles playback of local files (MP4, MKV) or streams from URLs, using libraries like VLC or FFmpeg bindings.

- **Data Management Module**: Stores user data in a local database (e.g., SQLite) for privacy, including watch logs, ratings, and preferences.
- **AI Recommendation Engine**: A local ML model (e.g., using scikit-learn for simple filtering or TensorFlow Lite for neural networks) that processes user data to generate suggestions. Features could include genre similarity, rating patterns, and viewing frequency.
- **User Interface**: A desktop/web app built with Streamlit, PyQt, or Electron for browsing libraries, playing videos, rating content, and viewing recommendations.

4.2 Technologies and Tools

- **Programming Languages**: Python (primary) for backend and AI; JavaScript/HTML for web-based UI if needed.
- **Libraries**: MoviePy or OpenCV for video handling; scikit-learn/TensorFlow Lite for AI; SQLite for database; Streamlit or Tkinter for GUI.
- **Development Environment**: Jupyter for prototyping, Git for version control.
- **Hardware Requirements**: Standard computer; GPU optional for faster model training.

5. Expected Outcomes

- A working prototype application for Windows/Linux/Mac, installable via pip or executable.
- Demonstration of personalized recommendations in scenarios like "action enthusiasts" or "drama preferences."
- Evaluation metrics: Recommendation hit rate >70% in tests, playback latency <2 seconds.
- Open-source repository on GitHub with code, datasets, and usage guides.
- Final report including code walkthrough, performance analysis, and potential expansions (e.g., multi-user support).

6. Budget

Low-cost project using free tools. Estimated costs:

Item	Description
API Usage	TMDB or similar for metadata (free tier sufficient)
Compute Resources	Local machine; optional cloud for initial training (e.g., Google Colab)
Software	All open-source
Testing Data	Public datasets (e.g., MovieLens)

7. Team and Resources

- **Team Members**: Ideal for 3-4 students, with roles in AI, frontend, and testing.
- **Supervision**: Supervised by Prof. Huang.
- **Risks and Mitigation**: Model accuracy issues – Use hybrid algorithms; Performance bottlenecks – Optimize with lightweight libraries; Data privacy – Emphasize local storage in design.