Abstract: Video Streaming Project with TCP Server and Flask Backend

This project implements a video streaming system where users can watch and download videos through a web application. The system is composed of a Flask-based web server, a TCP server for video file transfer, and a frontend interface for video playback. The video streaming process is optimized for efficiency, scalability, and smooth user experience by utilizing chunk-based transmission and a multi-threaded server architecture.  
  
Components:  
  
1. Flask Backend:  
 - The Flask web application handles user authentication, video list management, and video playback requests. Users can log in, register, view available videos, and watch or download them.  
 - The /watch/<filename> route streams videos by retrieving them from a TCP server in chunks. The video is streamed to the user’s browser as it is received, allowing for immediate playback without requiring the entire file to be downloaded beforehand.  
 - The /download/<filename> route allows users to download the requested video by streaming it in chunks, ensuring that the server doesn't hold the entire file in memory.  
  
2. TCP Server:  
 - The TCP server is responsible for transferring video files in chunks. When the Flask backend requests a video, the TCP server reads the file, sends its size, and transmits the file in manageable 8 KB chunks.  
 - The server handles multiple client connections concurrently using a thread pool, making it capable of serving multiple video requests simultaneously without significant delay.  
  
3. Chunk-Based Video Streaming:  
 - Video files are streamed in small chunks (8192 bytes), allowing efficient transmission of large files while minimizing memory usage. As the video is streamed, the user's browser can begin playback immediately, avoiding the need to wait for the entire file to be downloaded.  
 - This chunk-based approach ensures smooth streaming even if the video is large or if multiple users are requesting the same content simultaneously.  
  
4. User Interface:  
 - The frontend of the web application allows users to browse and watch videos with custom controls for fast forwarding and rewinding. A video player is embedded within the webpage, which receives the video in real-time from the backend.

5. Database Integration:  
 - The system uses an SQLite database to manage user authentication, allowing users to register, log in, and access personalized content. The database also stores user credentials securely, ensuring proper access control.  
  
Conclusion:  
This video streaming project leverages Flask for web development, a TCP server for efficient file transfer, and chunk-based streaming for seamless playback. The system is designed to scale efficiently, handle multiple users concurrently, and minimize memory consumption, making it suitable for serving video content in a local network. Through this architecture, users can enjoy smooth video streaming with real-time playback controls, while the server can handle large video files without overloading system resources.