

Video Upload, Sensitivity Processing, and Streaming Application Assignment

Overview

Build a comprehensive full-stack application that enables users to upload videos, processes them for content sensitivity analysis, and provides seamless video streaming capabilities with real-time progress tracking.

Project Objectives

Core Functionality

1. **Full-Stack Architecture:** Develop using Node.js + Express + MongoDB (backend) and React + Vite (frontend)
2. **Video Management:** Implement a complete video upload and secure storage system
3. **Content Analysis:** Process videos for sensitivity detection (safe/flagged classification)
4. **Real-Time Updates:** Display live processing progress to users
5. **Streaming Service:** Enable video playback using HTTP range requests
6. **Access Control:** Implement multi-tenant architecture with role-based permissions

Technical Requirements

Backend Implementation

- **RESTful API Design:** Create endpoints for video operations
 - Video upload with metadata handling
 - Video listing with filtering capabilities
 - Streaming service with range request support
- **Content Processing:** Implement a video sensitivity analysis mechanism
- **Real-Time Communication:** Use [Socket.io](#) for live progress updates
- **Database Management:** Store video metadata, processing status, and user data in MongoDB
- **Authentication & Authorisation:** Secure API endpoints with proper validation

Frontend Development

- **Upload Interface:** User-friendly video upload with progress indicators
- **Real-Time Dashboard:** Dynamic display of processing status and progress
- **Video Library:** Comprehensive list of uploaded videos with status indicators
- **Media Player:** Integrated video playback for processed content

- **Responsive Design:** Cross-platform compatibility and intuitive user experience

Advanced Features

Multi-Tenant Architecture

- **User Isolation:** Each user accesses only their own video content
- **Data Segregation:** Secure separation of user data and permissions
- **Scalable Design:** Support for multiple organisations or user groups

Role-Based Access Control (RBAC)

- **Viewer Role:** Read-only access to assigned videos
- **Editor Role:** Upload, edit, and manage video content
- **Admin Role:** Full system access, including user management and system settings

Video Processing Pipeline

1. **Upload Validation:** File type, size, and format verification
2. **Storage Management:** Secure file storage with proper naming conventions
3. **Sensitivity Analysis:** Automated content screening and classification
4. **Status Updates:** Real-time progress communication to the frontend
5. **Streaming Preparation:** Video optimisation for efficient streaming

Project Structure Requirements

Organization

- **Clear Separation:** Distinct backend and frontend directories
- **Modular Architecture:** Organised code structure with separation of concerns
- **Configuration Management:** Environment-specific settings and variables
- **Error Handling:** Comprehensive error management and user feedback

Development Standards

- **Clean Code:** Well-commented and maintainable codebase
- **Version Control:** Clear git commit history with meaningful messages
- **Documentation:** Comprehensive setup and usage instructions
- **Testing:** Basic testing implementation for critical functionalities

Stretch Goals (Optional Enhancements)

Advanced Filtering

- **Content-Based Filtering:** Filter videos by safety status (safe/flagged)

- **Metadata Filtering:** Search by upload date, file size, duration
- **Custom Categories:** User-defined video categorisation system

Performance Optimization

- **Video Compression:** Automatic optimisation for different quality levels
- **Caching Strategy:** Implement efficient caching for frequently accessed content
- **CDN Integration:** Content delivery network for improved streaming performance

Deliverables

Application Requirements

1. **Functional Demo:** Complete working application showcasing full workflow
 - Video upload process
 - Real-time processing updates
 - Content sensitivity analysis
 - Video streaming capability
2. **Code Quality:** Professional-grade implementation
 - Clean folder structure
 - Proper separation of concerns
 - Comprehensive error handling
 - Security best practices
3. **Documentation Package:**
 - Installation and setup guide
 - API documentation
 - User manual
 - Architecture overview
 - Assumptions and design decisions
4. **Deployment:**
 - Publicly accessible web application
 - Video demonstration of functionality
 - GitHub repository with complete source code

Workflow Demonstration

Complete User Journey

1. **User Registration/Login:** Secure authentication system
2. **Video Upload:** Intuitive upload interface with progress tracking
3. **Processing Phase:** Real-time updates on sensitivity analysis
4. **Content Review:** Status display (safe/flagged) with appropriate actions
5. **Video Streaming:** Seamless playback of processed videos
6. **Management Tools:** Video library with filtering and management options

Technical Specifications

Backend Technology Stack

- **Runtime:** Node.js (Latest LTS version)
- **Framework:** Express.js
- **Database:** MongoDB with Mongoose ODM
- **Real-Time:** [Socket.io](#)
- **Authentication:** JWT or similar secure token system
- **File Handling:** Multer or similar for video uploads

Frontend Technology Stack







- **Build Tool:** Vite
- **Framework:** React (Latest stable version)
- **State Management:** Context API or Redux
- **Styling:** CSS Modules, Styled Components, or Tailwind CSS
- **HTTP Client:** Axios or Fetch API
- **Real-Time:** [Socket.io](#) client

Infrastructure Requirements







- **File Storage:** Local storage or cloud storage (AWS S3, Google Cloud)
- **Video Processing:** FFmpeg or similar video processing library
- **Hosting:** Cloud platform (Heroku, Netlify, Vercel, or similar)
- **Database Hosting:** MongoDB Atlas or similar cloud database service

Success Criteria

Functional Requirements Met

-  Complete video upload and storage system
-  Real-time processing progress updates
-  Video sensitivity analysis and classification
-  Secure video streaming with range requests
-  Multi-tenant user isolation
-  Role-based access control implementation

Quality Standards Achieved

-  Clean, maintainable code structure
-  Comprehensive documentation
-  Secure authentication and authorisation
-  Responsive and intuitive user interface
-  Proper error handling and user feedback
-  Public deployment with demo video

