Software Requirements Specification (SRS)

EduVidShare System

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document provides a comprehensive description of the EduVidShare platform - an educational video sharing system that allows verified teachers to share educational content with students.

1.2 Scope

EduVidShare is a web-based application that enables:

* Teacher verification through certificate submission and approval
* Educational video upload, approval, and distribution
* Social features like comments and likes on educational content
* Role-based access control for students, teachers, and administrators

1.3 Definitions and Acronyms

* **SRS**: Software Requirements Specification
* **API**: Application Programming Interface
* **JWT**: JSON Web Token
* **CRUD**: Create, Read, Update, Delete

2. System Description

2.1 User Roles

The system supports three distinct user roles:

1. **Students**:
   * Can register and login
   * Browse and watch approved videos
   * Like videos to show appreciation
   * Comment on videos to ask questions
   * Download videos for offline viewing
2. **Teachers**:
   * Can register and login
   * Upload certificates for verification
   * Upload educational videos once verified
   * Manage their uploaded videos
   * View analytics on their content
   * Respond to student comments
3. **Administrators**:
   * Manage user accounts
   * Approve/reject teacher certificates
   * Approve/reject uploaded videos
   * Moderate comments
   * View system-wide analytics

2.2 Key Features

1. **Authentication System**
   * JWT-based authentication for secure access
   * Role-based authorization to control feature access
2. **Certificate Management**
   * Upload interface for teaching credentials
   * Admin review workflow
   * Automatic teacher role assignment upon approval
3. **Video Management**
   * Upload interface supporting multiple formats
   * Content moderation through admin approval
   * Streaming capabilities for online viewing
   * Download functionality for offline access
4. **Social Interaction**
   * Commenting system for discussions
   * Like system to indicate content quality
   * View counting for popularity tracking
5. **Administration**
   * User management interface
   * Content approval workflows
   * Moderation tools

3. Technical Implementation

3.1 API Endpoints

1. **Authentication API**
   * **POST /api/auth/register** - Register a new user
   * **POST /api/auth/login** - Authenticate and receive JWT token
2. **Certificate Management API**
   * **POST /api/certificate** - Upload a certificate
   * **GET /api/certificate** - Get all certificates (admin)
   * **GET /api/certificate/pending** - Get pending certificates
   * **GET /api/certificate/user/{userId}** - Get user certificates
   * **GET /api/certificate/{id}** - Get certificate details
   * **GET /api/certificate/download/{id}** - Download certificate file
   * **PUT /api/certificate/approve/{id}** - Approve certificate
   * **PUT /api/certificate/reject/{id}** - Reject certificate
   * **DELETE /api/certificate/{id}** - Delete certificate
3. **Video Management API**
   * **POST /api/video** - Upload a video
   * **GET /api/video** - Get all approved videos
   * **GET /api/video/pending** - Get pending videos
   * **GET /api/video/user/{userId}** - Get user videos
   * **GET /api/video/{id}** - Get video details
   * **GET /api/video/stream/{id}** - Stream video
   * **GET /api/video/download/{id}** - Download video
   * **PUT /api/video/approve/{id}** - Approve video
   * **PUT /api/video/reject/{id}** - Reject video
   * **DELETE /api/video/{id}** - Delete video
4. **Comment API**
   * **POST /api/videocomment** - Add a comment
   * **GET /api/videocomment/video/{videoId}** - Get video comments
   * **GET /api/videocomment/{id}** - Get comment details
   * **PUT /api/videocomment/{id}** - Update a comment
   * **DELETE /api/videocomment/{id}** - Delete a comment
5. **Like API**
   * **POST /api/videolike/{videoId}** - Like a video
   * **DELETE /api/videolike/{videoId}** - Unlike a video
   * **GET /api/videolike/video/{videoId}** - Get video likes count
   * **GET /api/videolike/status/{videoId}** - Check if user liked video

3.2 Data Models

1. **User**
   * UserId (PK)
   * Username
   * PasswordHash
   * Email
   * RoleId (FK)
2. **Role**
   * RoleId (PK)
   * RoleName
3. **Certificate**
   * CertificateId (PK)
   * UserId (FK)
   * Title
   * Description
   * FilePath
   * FileType
   * FileContent
   * UploadDate
   * Status (Pending/Approved/Rejected)
   * AdminRemarks
   * ApprovalDate
4. **Video**
   * VideoId (PK)
   * UserId (FK)
   * Title
   * Description
   * FilePath
   * FileType
   * UploadDate
   * Status (Pending/Approved/Rejected)
   * AdminRemarks
   * ApprovalDate
   * ViewCount
5. **VideoComment**
   * CommentId (PK)
   * VideoId (FK)
   * UserId (FK)
   * Content
   * CommentDate
6. **VideoLike**
   * LikeId (PK)
   * VideoId (FK)
   * UserId (FK)
   * LikeDate

4. System Workflows

4.1 Certificate Approval Workflow

1. Teacher uploads certificate with credentials
2. Administrator reviews the certificate
3. System notifies teacher of decision
4. If approved, teacher role is granted
5. If rejected, teacher can submit a new certificate

4.2 Video Upload and Approval Workflow

1. Teacher uploads educational video
2. System stores video and creates database record
3. Administrator reviews pending videos
4. Administrator approves or rejects the video
5. System notifies teacher of decision
6. Approved videos become available to students

5. Non-Functional Requirements

5.1 Security

* JWT authentication for all protected endpoints
* Role-based access control
* Encrypted storage of sensitive information

5.2 Performance

* Support for large video file uploads (up to 500MB)
* Efficient video streaming capabilities
* Support for multiple concurrent users

5.3 Usability

* Intuitive interfaces for all user roles
* Clear workflow for certificate and video submission
* Responsive design for various devices

5.4 Reliability

* Proper error handling throughout the application
* Data validation to ensure integrity
* Transactional operations for critical workflows

6. Deployment Architecture

The system uses a multi-tier architecture:

* Client devices (web browsers) connect via HTTPS
* ASP.NET Core 8.0 web server hosts the API
* SQL Server database stores application data
* File storage for videos and certificates

7. Conclusion

The EduVidShare system provides a secure platform for sharing educational content between verified teachers and students. The detailed workflows and role-based access ensure quality content while the social features enhance engagement and learning outcomes.