**WEB STREAMING WEBSİTE**

**Graduation Project Report**

**CMSE 406**

**Team members:**

**Zeynep Pelin Çolak, 17300009**

**Doğukhan Ramazan Sapsızoğlu, 20450049**

**Abdulaziz Mahmoud 18700859**

**Supervisor:**

**Asst. Prof. Dr. Ahmet Ünveren**

**Computer Engineering Department**

**Eastern Mediterranean University**

**Fall 2024**

# ABSTRACT

This chapter presents a comprehensive design approach for developing a video streaming platform with a focus on scalability, efficiency, and user experience, akin to YouTube. The principal objectives include designing a robust architecture capable of handling large user volumes and diverse content types while ensuring seamless video uploading, transcoding, and streaming. Leveraging cloud services and optimization techniques, the methods employed prioritize cost-effectiveness and reliability. The results outline a high-level system architecture encompassing client interaction, content storage, transcoding pipelines, and content delivery networks. Through deep-dive analyses and practical insights, the chapter concludes that effective utilization of existing cloud infrastructure, parallel processing, and error handling mechanisms are pivotal for building a successful video streaming service.

**Keywords:**

Video streaming, YouTube, architecture design, cloud services, transcoding, CDN, scalability, reliability, optimization, error handling, user experience, content delivery, system architecture.

**TABLE OF CONTENTS**

[ABSTRACT II](#_Toc478723910)

[Table of Contents III](#_Toc478723911)

[LIST OF FIGURES V](#_Toc478723912)

[LIST OF TABLES VI](#_Toc478723913)

1. INTRODUCTION

2. PROJECT PLANNING AND MANAGEMENT

2.1 List of Work Packages

2.2 List of Milestones

2.3 List of Risks

2.4 Project Team

2.5 Organization Scheme

2.6 Using Basic Cocomo Approach

3. REQUIREMENTS ANALYSIS

3.1 Functional Requirements

3.1.1.Video Upload

3.1.1.1 User Authentication

3.1.1.2 Video Uploading

3.1.1.3 Video Splitting

3.1.1.4 Pre-signed URL Generation

3.1.2. Video Processing

3.1.2.1 Video Storage

3.1.2.2 Video Transcoding

3.1.2.3 Metadata Update

3.1.3. Video Streaming

3.1.3.1 Content Delivery Network (CDN)

3.1.3.2 Adaptive Streaming

3.1.3.3 Playback

3.1.4. User Interaction

3.1.4.1 Comments and Reactions

3.1.4.3 Playlists and Favorites

3.1.5. Metadata Management

3.1.5.1 Metadata Storage

3.1.5.2 Search and Recommendations

3.1.6. System Requirements

3.1.6.1 High Availability:

3.1.6.2 Scalability:

3.1.6.3 Security:

3.1.6.4 Cost Optimization:

3.1.6.5 Error Handling:

3.1.7. Additional Features

3.1.7.1 Live Streaming:

3.1.7.2 Video Takedowns:

3.2 Non-Functional Requirements

3.2.1 Scalability

3.2.2 Performance

3.2.3 Availability

3.2.4 Reliability

3.2.5 Security

3.2.6 Maintainability

3.2.7 Usability

3.2.8 Compliance

3.2.9 Cost Efficiency

3.3 Realistic constraints

3.4 Ethical issues

4. DESIGN

4.1 High level design (architectural)

4.2 Software design

4.2.1Use Case Diagram

4.2.2Sequence Diagram

4.2.3. Activity Diagram

4.2.3Class Diagram

4.2.4 BPMN Diagram: Video Processing Workflow

4.2.5 DFD Diagrams

4.2.5.1 Context Diagram

4.2.5.2 Level-0 Diagram

4.2.5.3 Level-1 Diagram

4.2.6 E-R Diagram

4.2.7 Relational Table

4.2.8 Physical Database Tables

5. IMPLEMENTATION

5.1 Tools, technologies and platforms used

5.2 Algorithms

5.3 Standards

5.4 Detailed description of the implementation (coding)

6. QUALITY AND TESTING

6.1 Quality Assurance Activities During Project Life Cycle

6.2 Thought Process Map (TMAP)

6.3 Kano Model

6.4 Affinity diagram

6.5 Pareto Chart

6.6 SWOT Analsis

A.Quality Metric Tables

B.Quality Checklist Tables

C.Quality Audit Checklist Tables

D.Quality Process Analysis (QPA)

6.1 Quality Control (QC) Activities After Implementation Completed

A.Test Schedule

B.Inspection Tables

C.Test-Cases for Critical Modules

7. USER GUIDE OF THE SYSTEM

7.1 About Page

7.2 Channel Page

7.3 Community Page

7.4 Community Create Page

7.5 Index Page

7.6 Upload Page

7.7 Video Page

[8. DISCUSSION 8](#_Toc478723931)

[9. CONCLUSION9](#_Toc478723932)

[10. REFERENCES10](#_Toc478723933)

[APPENDICES11](#_Toc478723934)

[A. Instructions for installing the system11](#_Toc478723935)

[B. Code for the system11](#_Toc478723936)

[C. Other relevant material11](#_Toc478723937)

# LIST OF FIGURES

**1. Organization Scheme**

**2. Network Diagram**

**3. Gannt Chart**

**4. High level design (architectural)**

**5. Use case diagram**

**6. Sequence diagram**

**7. Activity Diagram**

**8. Class Diagram**

# 9. BPMN Diagram

**10. Context Diagram**

**11. Level-0 Diagram**

**12. Level-1 Diagram**

**13. E-R Diagram**

**14. Relational Table**

**15. Physical Database Tables**

**16. Thought Process Map (TMAP)**

**17. Kano Model**

**18. Affinity Diagram**

**19. Pareto Chart**

**20. Fishbone diagram**

**21. About Page**

**22.** **Channel Page**

**23.** **Community Page**

**24.** **Community Create Page**

**25.** **Index Page**

**26.** **Upload page**

**27.** **Video Page**

# LIST OF TABLES

**1. List of Work Packages Table 1**

**2. List of Work Packages**

**3. List of Work Packages**

**4. List of Work Packages**

**5. List of Milestones**

**6. List of Risks**

**7. Project Team**

**8. Cocomo Table**

**9. Task Table**

**10. Algorithms Table**

**11. Brainstorming**

**12. Swot Analysis**

**13. Quality Metric Tables**

**14. Quality Checklist Tables**

**15. Quality Audit Checklist Tables**

**16. Audit Plans**

**17. Audit Plans Table**

**18. Test Schedule Table**

**19.** **Inspection Table**

**20.** **Test-Cases for Critical Modules Table**

# 1. INTRODUCTION

# The proliferation of online video content consumption has led to a surge in demand for robust video streaming platforms capable of delivering high-quality content to users worldwide. In response to this growing need, the project aims to address the complexities inherent in designing and implementing a scalable video streaming service reminiscent of YouTube.The problem at hand revolves around the need for a platform that can efficiently handle the uploading, transcoding, and streaming of videos while ensuring optimal user experience and minimal infrastructure costs. With the rise of digital content creation and consumption, there is a pressing need for a reliable solution that caters to the diverse needs of content creators and viewers alike.The project's beneficiaries include content creators seeking a reliable platform to share their creations, viewers looking for seamless streaming experiences across various devices, and organizations aiming to tap into the lucrative market of online video content delivery. By providing a robust and efficient video streaming service, the project aims to empower content creators, engage viewers, and drive revenue generation opportunities.While various video streaming platforms exist in the market, attempts by others to solve the problem have often faced challenges related to scalability, cost-effectiveness, and infrastructure complexity. Many platforms struggle to efficiently manage large user bases and diverse content libraries, leading to suboptimal user experiences and increased operational costs.In light of these challenges, this project endeavors to explore innovative design approaches, leverage cutting-edge technologies, and draw insights from existing solutions to develop a scalable and efficient video streaming platform. Through rigorous analysis, design iterations, and practical implementations, the project seeks to deliver a solution that not only meets the demands of today's digital landscape but also sets a benchmark for future video streaming endeavors.

# 2. PROJECT PLANNING AND MANAGEMENT

# 2.1 List of Work Packages

**List of Work Packages Table 1-2-3-4**

|  |  |
| --- | --- |
| **Work Package No** | 1 |
| **Work Package Name** | **Project Feasibility and Pre-Research (Feasibility Analysis)** |
| **Start-End Date and Time** | 26/09/24 - //24 |
| **Related Organizations** |  |
|  |  |

|  |
| --- |
| **1- List the activities of work packages.** |
| **1.1 Project Process and Economic Feasibility:**   * Determination of system scope * Analysis of Similar products * Economic Feasibility analysis * Risk Analysis * Distribution of Tasks   **1.2 Technological Feasibility:**  Determination of technical resources needed |
| **2- Describe the methods and parameters that will be used for work package.** |
| Searching online documentation of similar projects |
| **3- List the experiments, tests and analysis in the work package.** |
| * Online Research   Cost Management |
| **4- List the output of work package and its success criteria’s.** |
| **Outputs:**   * Having a preliminary idea of the scope of the system * Estimation on profits and costs * Establishing the distribution of tasks   **Success Criteria’s:**   * Better grasp on the timeline, time constraints and budget constraints * Better understanding of the system |
| **5- Explain the relation of output with other work packages** |
| This is the first step of the project where every other step depends on it |

|  |  |
| --- | --- |
| **Work Package No** | 2 |
| **Work Package Name** | **Based System Design Technology (Analysis & Design stage)** |
| **Start-End Date and Time** | // 24 - // 24 |
| **Related Organizations** |  |

|  |
| --- |
| **1- List the activities of work packages.** |
| * Determining System Requirements * Produce SRS document * Design System Architecture * Produce SDS document |
| **2- Describe the methods and parameters that will be used for work package.** |
| Online Research for Design Documents and System Requirements |
| **3- List the experiments, tests and analysis in the work package.** |
| * Research about convenient development interface * Research about coding language   Research about application of system architecture to project |
| **4- List the output of work package and its success criteria’s.** |
| **Outputs:**   * Compatible requirements report * Diagrams that describe the system structure   **Success Criteria’s:**  Understanding system structure and scope |
| **5- Explain the relation of output with other work packages** |
| Development is extremely dependent on the success of this step |

|  |  |
| --- | --- |
| **Work Package No** | 3 |
| **Work Package Name** | **Development of System Software (Development Stage)** |
| **Start-End Date and Time** | //24 - //24 |
| **Related Organizations** |  |

|  |
| --- |
| **1- List the activities of work packages.** |
| * Front End Development * Back End Development * Database Development * Video processing * Security |
| **2- Describe the methods and parameters that will be used for work package.** |
| Html, CSS, JavaScript for Front end development  Node.js for Back end development  MySQL for Database development |
| **3- List the experiments, tests, and analysis in the work package.** |
| Develop Code and Database |
| **4- List the output of work package and its success criteria’s.** |
| **Outputs:**  Functional System, ready to be tested  **Success Criteria’s:**   * Low error rate in coding procedures * Effective database design and implementation * Well designed and friendly user interface |
| **5- Explain the relation of output with other work packages** |
| Only once sufficient progress is made in this step, can the next step commence |

|  |  |
| --- | --- |
| **Work Package No** | 4 |
| **Work Package Name** | **Prototype Implementation and Test Study and Maintenance (Test & Maintenance stage)** |
| **Start-End Date and Time** | //24 - //25 |
| **Related Organizations** |  |

|  |
| --- |
| **1- List the activities of work packages.** |
| * Security testing * Functional Testing * Non-Functional Testing * User Acceptance Testing * Interface Testing * Compatibility Testing * Testing on Real Users |
| **2- Describe the methods and parameters that will be used for work package.** |
| * Performance tests * Usability tests * Unit and Integration tests |
| **3- List the experiments, tests and analysis in the work package.** |
| * Black Box Testing * White Box Testing   Integration Testing |
| **4- List the output of work package and its success criteria’s.** |
| **Outputs:**  Reports of Tests  **Success Criteria’s:**  No errors while testing  All bugs are fixed |
| **5- Explain the relation of output with other work packages** |
| Success in this step means all other steps were completed and the system is ready to be used |

# 2.2 List of Milestones

|  |  |  |
| --- | --- | --- |
|  | Description of Output | Expected Time Interval |
|  |  |  |
| 1 | **Project Feasibility and Pre-Research** | 26/9/2024 – //2024 |
| 2 | **System Design** | //24 – //24 |
| 3 | **Development of System Software** | //24 – //24 |
| 4 | System Testing | //24 – //25 |
|  |  |  |
|  |  |  |
|  |  |  |

# List of Milestones Table 5

# 2.3 List of Risks

|  |  |  |  |
| --- | --- | --- | --- |
| Risk | Probability | Effects | Your Strategy |
| The time required to develop the software is underestimated. | High | Serious | We drop some functionalities to finish in time |
| The rate of defect repair is underestimated. | Moderate | Tolerable | Replace potentially defective components with more reliable bought-in components. |
| The size of the software is underestimated. | High | Serious | Investigate buying SW components.  Investigate use of a program generator. |
| Key staff are ill at critical times in the project. | Moderate | Serious | Reorganize team so that there is more overlap of work and people therefore understand each other’s jobs. |
| The database used in the system cannot process as many transactions per second as expected. | Moderate | Serious | Investigate the possibility of buying a higher-performance database. |
|  | |  |  |
|

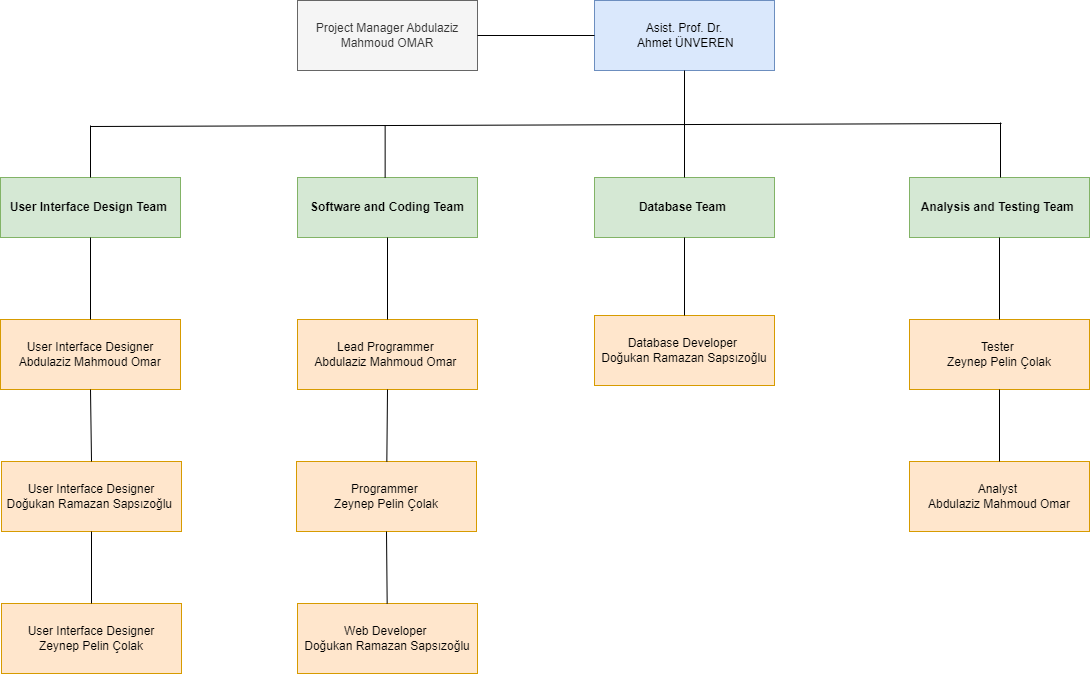
# List of Risks Table 6

# 2.4 Project Team

**Project Team Table 7**

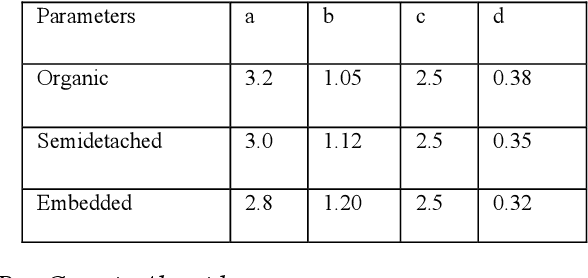
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Personnel Name | Title | ID | Education Status | Graduation Date | Date of Starting Work | Idea Owner |
|  |  |  |  |  |  |  |
| Doğukan Ramazan SAPSIZOĞLU | Database Developer | 20450049 | Undergraduate | 2025 | 26.09.2024 | Yes |
| Zeynep Pelin ÇOLAK | User Interface Designer | 17300009 | Undergraduate | 2025 | 26.09.2024 | Yes |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

# 2.5 Organization Scheme



**Organization Scheme Figure 1**

**2.6 Using Basic Cocomo Approach:**



**Cocomo Table 8**

**Calculations**:

Estimated line of codes are 10 KLOC

Using Semi-detached:

Effort = a\*(KLOC)^b

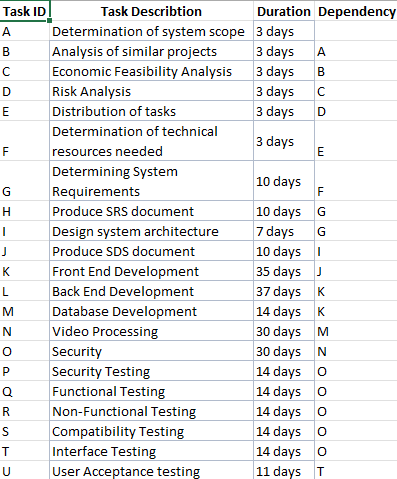
E= 3.0\*(10) ^1.12 = 39.5477

Duration = c\*(Effort)^d

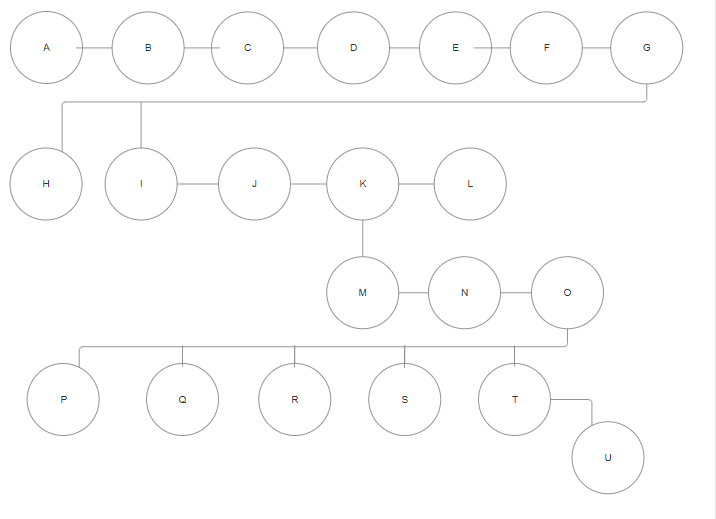
D = 2.5(39.5477) ^0.35 = 9.0559

Team Members = Effort/Duration

T = 39.5477/9.0559 = 4 people



**Task Table 9**

**Network Diagram Figure 2**

Critical Path= A > B > C > D > E > F > G > I > J > K > M > N > O > T > U

Duration= 3 + 3 + 3 + 3 + 3 + 3 + 10 + 7 + 10 + 35 + 14 + 30 + 30 + 14 + 11 = 179 days

|  |
| --- |
| **Gannt Cahrt Figure 3** |
|  |

# 3. REQUIREMENTS ANALYSIS

## 3.1 Functional Requirements

**3.1.1.Video Upload**

**3.1.1.1 User Authentication**:

• Users must be authenticated to upload videos.

• Support for OAuth or other authentication mechanisms.

**3.1.1.2 Video Uploading:**

• Users should be able to upload videos up to 1GB in size.

• Support for various video formats and resolutions.

• Video upload progress tracking for users.

**3.1.1.3 Video Splitting:**

• Split videos into smaller chunks by GOP alignment for faster uploads and resumable uploads.

• Preprocessing on the client-side to split videos before uploading.

**3.1.1.4 Pre-signed URL Generation:**

• Generate pre-signed URLs for secure video uploads to cloud storage.

**3.1.2. Video Processing**

**3.1.2.1 Video Storage:**

• Store original videos in a blob storage system.

• Transcode videos into multiple resolutions and formats.

**3.1.2.2 Video Transcoding:**

• Encode videos to support different bitrates, resolutions, and formats.

• Generate thumbnails and apply watermarks as needed.

• Store transcoded videos in a separate storage system for delivery.

**3.1.2.3 Metadata Update:**

• Update video metadata (title, description, tags, URL, size, format, resolution) in the metadata database and cache.

• Handle video transcoding completion events and update metadata accordingly.

**3.1.3. Video Streaming**

**3.1.3.1 Content Delivery Network (CDN):**

• Serve videos from a CDN for low latency and high availability.

• Cache popular videos in CDN to reduce costs.

**3.1.3.2 Adaptive Streaming:**

• Support adaptive streaming protocols (e.g., MPEG-DASH, Apple HLS) for smooth playback across different network conditions.

• Allow users to manually change video quality during playback.

**3.1.3.3 Playback:**

• Ensure videos start streaming immediately upon clicking play.

• Support for playback on web browsers, mobile apps, and smart TVs.

**3.1.4. User Interaction**

**3.1.4.1 Comments and Reactions:**

• Allow users to comment on videos.

• Enable users to like, dislike, and share videos.

**3.1.4.3 Playlists and Favorites:**

• Users can create and manage playlists.

• Allow users to save videos to their favorites or watch later lists.

**3.1.5. Metadata Management**

**3.1.5.1 Metadata Storage:**

• Store metadata in a sharded and replicated database for high performance and availability.

• Cache frequently accessed metadata for faster retrieval.

**3.1.5.2 Search and Recommendations:**

• Implement search functionality to find videos by title, description, tags, and other metadata.

• Provide personalized video recommendations based on user preferences and viewing history.

**3.1.6. System Requirements**

**3.1.6.1 High Availability:**

• Ensure system components are redundant and can handle failures gracefully.

• Use load balancers to distribute traffic across multiple servers.

**3.1.6.2** **Scalability**:

• Design the system to scale horizontally to handle increasing numbers of users and videos.

• Implement auto-scaling for API servers, transcoding servers, and other components.

**3.1.6.3 Security:**

• Implement encryption for video uploads and storage.

• Protect videos with DRM, AES encryption, or visual watermarking.

• Ensure secure access to APIs and storage systems.

**3.1.6.4 Cost Optimization:**

• Optimize CDN usage to reduce costs by caching popular videos and serving less popular videos from high-capacity storage.

• Analyze viewing patterns to make informed decisions about video distribution and storage.

**3.1.6.5 Error Handling:**

• Implement retry mechanisms for recoverable errors in uploading, transcoding, and other processes.

• Gracefully handle non-recoverable errors and provide appropriate feedback to users.

**3.1.7. Additional Features**

**3.1.7.1 Live Streaming:**

• Support for live streaming with low latency and smooth playback.

• Handle live video encoding and streaming in real-time.

**3.1.7.2 Video Takedowns:**

• Implement mechanisms for removing videos that violate policies or are flagged by users.

• Automate detection of inappropriate content during the upload process.

## 3.2 Non-Functional Requirements

**3.2.1 Scalability**

**Vertical Scalability**: The system must be able to improve performance by increasing hardware resources to handle growing loads.

**Horizontal Scalability**: The system must be able to expand by adding more servers to handle increasing numbers of users. This is critical for API servers and the database.

**3.2.2 Performance**

**Response Time:** User requests should be answered quickly. Low latency should be ensured for video playback requests using a CDN.

**Throughput:** The system should be able to handle a high number of operations per second. It should support simultaneous high-volume video uploads and views.

**3.2.3 Availability**

**Uptime:** The system should ensure 99.9% uptime, minimizing downtime even during maintenance or unexpected failures.

**Redundancy:** All critical components (databases, API servers, CDN, etc.) should be configured redundantly.

**3.2.4 Reliability**

**Fault Tolerance:** The system should continue to operate without interruption in case of component failures, with automatic error detection and fallback mechanisms.

**Error Recovery:** The system should quickly recover from any errors, with minimal disruption to ongoing operations.

**3.2.5 Security**

**Data Encryption:** User data and videos should be encrypted both in transit and at rest.

**Authentication and Authorization:** User authentication should be secure, and only authorized users should be able to perform certain operations.

**DDoS Protection:** The system should provide protection against Distributed Denial of Service (DDoS) attacks.

**3.2.6 Maintainability**

**Code Modularity:** The system should be developed in a modular structure for easy maintenance. Each component should be developed and tested independently.

**Logging and Monitoring:** Comprehensive logging and monitoring mechanisms should be in place to track system performance and security.

**3.2.7 Usability**

**User Interface:** The user interface should be user-friendly and intuitive, providing accessibility for all user groups.

**Responsiveness:** Web and mobile applications should offer a responsive design, adapting to different device screen sizes.

**3.2.8 Compliance**

**Regulatory Compliance:** The system should comply with relevant legal regulations such as GDPR and COPPA.

**Content Moderation:** Video content should be moderated to protect against copyright infringement, pornography, or other illegal content.

**3.2.9 Cost Efficiency**

**Resource Optimization:** The system should use resources efficiently to minimize costs, especially optimizing CDN usage costs.

**Cloud Services:** Cloud services should be utilized cost-effectively, with capacity increases as needed.

## 3.3 Realistic constraints

## Economic:

## Accessibility: The system should be designed to be accessible to a broad audience without requiring significant financial investment. This includes offering free usage tiers with basic features and affordable premium plans.

## Cost of Entry: Users should not need expensive hardware or software to use the platform effectively. The system should function well on commonly available devices such as smartphones, tablets, and personal computers.

## Environmental:

## Power Consumption: The platform should aim to minimize power consumption by optimizing server efficiency and leveraging green data centers.

## Pollution: Efforts should be made to reduce the carbon footprint of the platform by using environmentally friendly practices in server maintenance and choosing data centers that use renewable energy sources.

## Social:

## Age Restrictions: There should be clear policies prohibiting users under a certain age (e.g., under 13 or 18, depending on content) from using the platform.

## Content Moderation: The platform should have robust content moderation policies to prevent the sharing of inappropriate or harmful content, protecting vulnerable segments of society.

## Political:

## Geopolitical Sensitivity: The platform should respect local laws and political climates, avoiding content or functionalities that might be considered sensitive or illegal in certain regions.

## Compliance with Local Regulations: The platform should comply with all relevant national and international regulations, ensuring that it does not inadvertently violate any political constraints.

## Ethical:

## Intellectual Property: The platform should enforce strict policies against the use of pirated content or plagiarism. Users must be required to upload only their original content or content they have rights to.

## Transparency: The platform should be transparent about data usage policies and obtain explicit consent from users for data collection and processing.

## Health and Safety:

## Screen Time: The platform should provide features that encourage healthy screen time habits, such as reminders to take breaks.

## Content Safety: There should be mechanisms in place to prevent the sharing of harmful or dangerous content, ensuring a safe environment for all users.

## Manufacturability:

## Resource Efficiency: The platform should be designed to run efficiently, requiring minimal hardware resources to operate, which helps in scalability and reduces operational costs.

## Availability of Resources: The necessary technological resources, such as cloud services and development tools, should be readily available and cost-effective.

## Sustainability:

## Long-term Viability: The platform should be built with a focus on long-term sustainability, ensuring that it can adapt to technological advancements and changing user needs over time.

## Maintenance: The platform should be designed for easy maintenance and updates, reducing the long-term cost and effort required to keep it running efficiently.

## 3.4 Ethical issues

# Content Responsibility:

# Illegal Activities: The platform must implement stringent policies and monitoring systems to prevent and report the sharing of illegal content, such as copyrighted material without permission, hate speech, and other unlawful activities.

# User Privacy: The platform must prioritize user privacy, ensuring that personal data is collected and used ethically, with transparent privacy policies and user consent.

# Misuse of Platform:

# Cyberbullying: The platform should have mechanisms to detect and prevent cyberbullying and harassment. This includes reporting tools and support for victims.

# Misinformation: There should be measures to prevent the spread of misinformation and fake news, such as fact-checking features and clear guidelines on content verification.

# Ethical Development:

# Originality: The development process must avoid unethical practices such as copying code or ideas from other projects without proper acknowledgment. The platform should encourage innovation and respect intellectual property.

# Bias and Fairness: The platform’s algorithms and policies should be designed to avoid bias, ensuring fair treatment and equal opportunities for all users regardless of their background.4. DESIGN

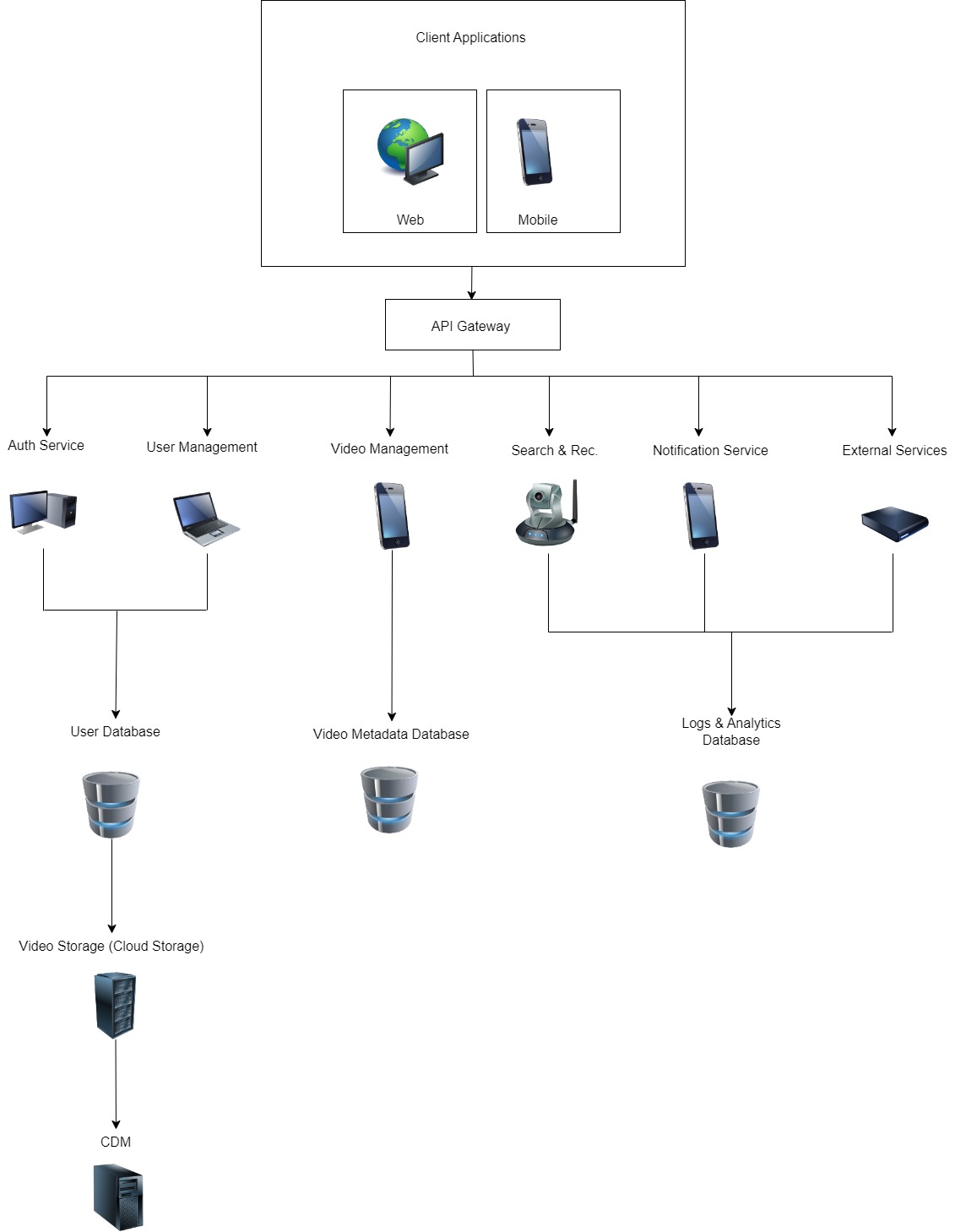
## 4.1 High level design (architectural)

 **Client**: Provides access via web or mobile applications.

 **API Servers**: Handles operations like video uploading, user authentication, and adding comments.

 **CDN (Content Delivery Network)**: Ensures fast and efficient video playback.

 **Database**: Stores user, video, and comment information.  
This stage focuses on how the components interact and communicate with each other.

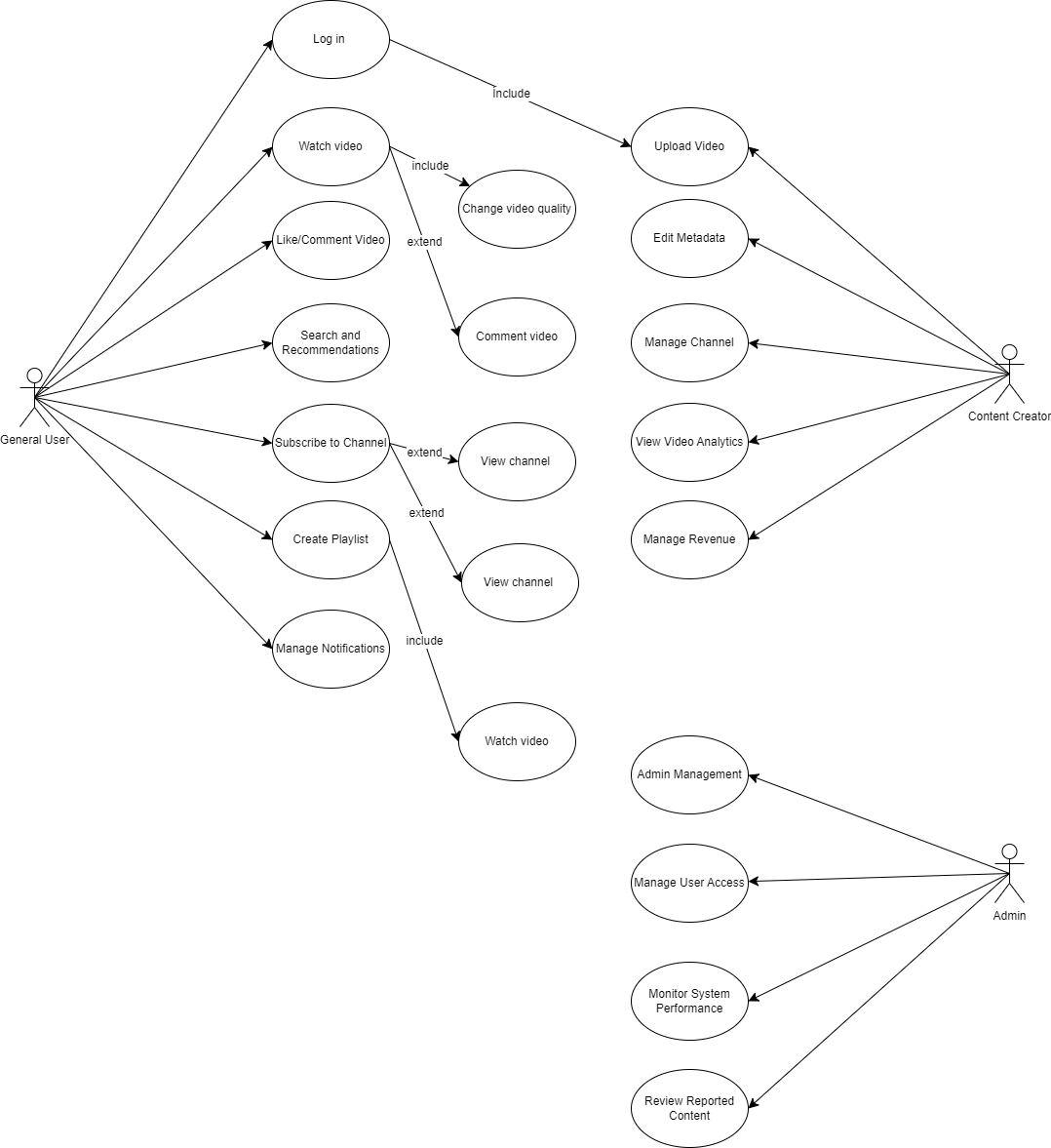


**High level design (architectural) Figure 4**

## 4.2 Software design

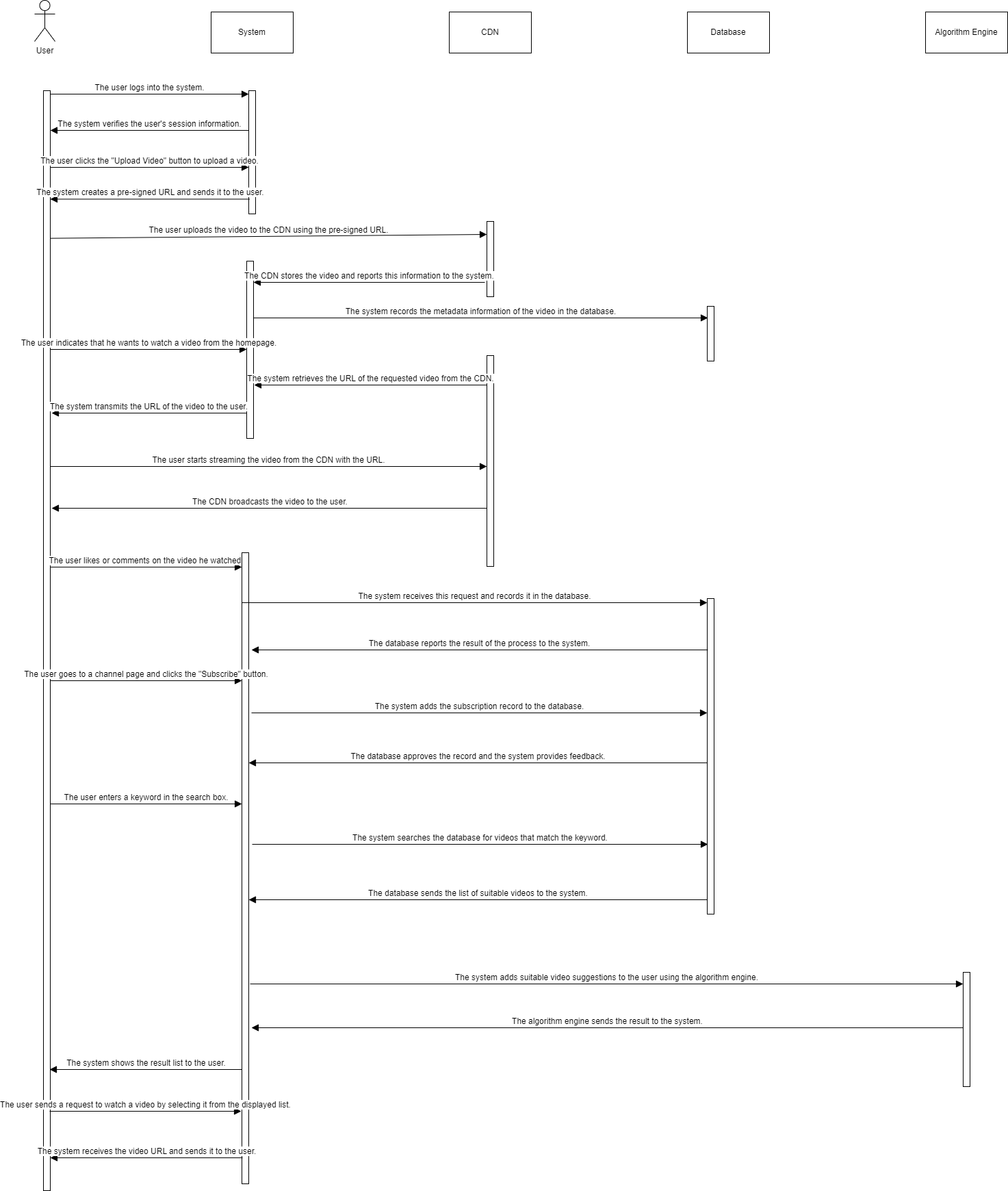
* + 1. **Use Case Diagram**

The Use Case Diagram illustrates how the system interacts with its users. It identifies the primary actors (e.g., user, content creator, admin) and the actions they perform (e.g., uploading a video, watching a video, commenting, subscribing to a channel). This diagram is ideal for understanding the functional scope of the system.



**Use case diagram Figure 5**

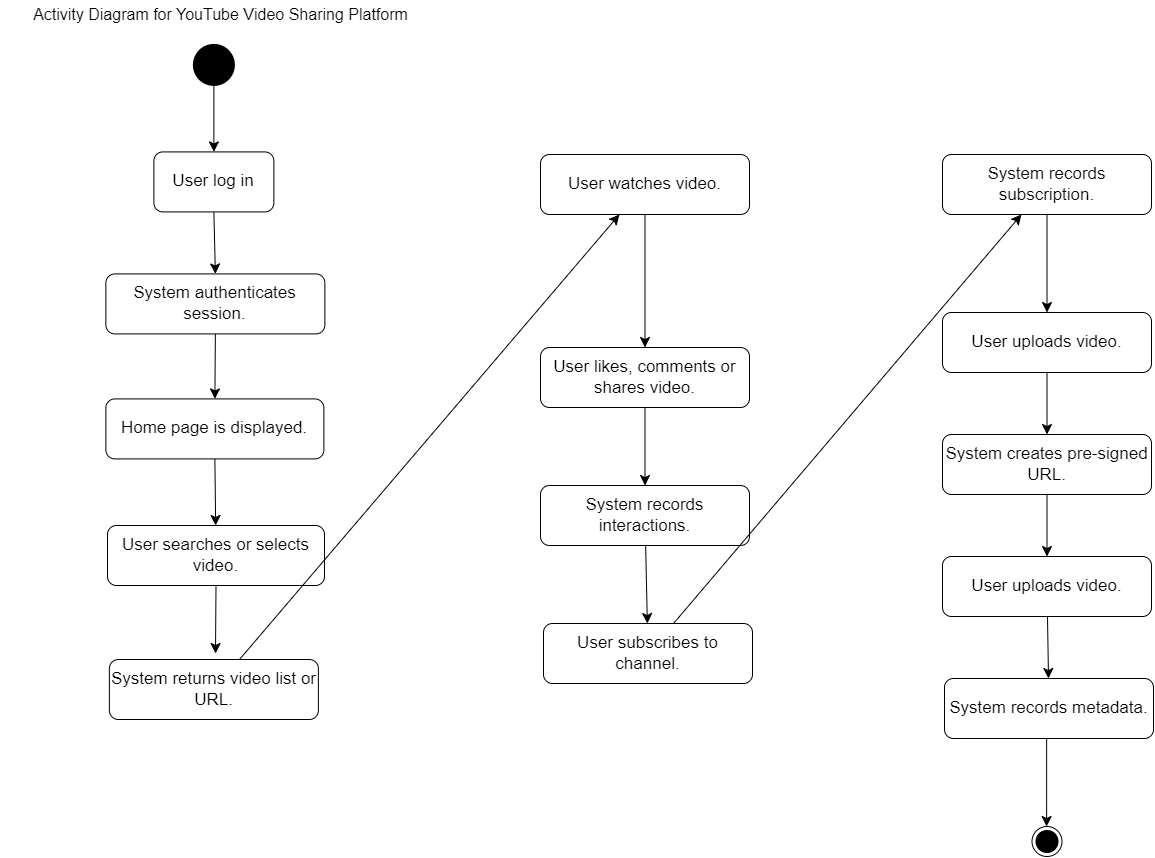
* + 1. **Sequence Diagram**
* The user logs into the system.
* The user selects a video to watch.
* The system fetches the video URL and sends it to the user.
* The user watches the video, leaves a comment, or likes it.  
  This diagram visualizes the timeline and interactions between the actors and the system.



**Sequence diagram Figure 6**

**4.2.3. Activity Diagram**

The user logs in → searches for a video → watches the video → leaves a comment → logs out.  
This diagram highlights decision points, parallel processes, and the sequence of operations in a simplified manner.



**Activity Diagram Figure 7**

* + 1. **Class Diagram**

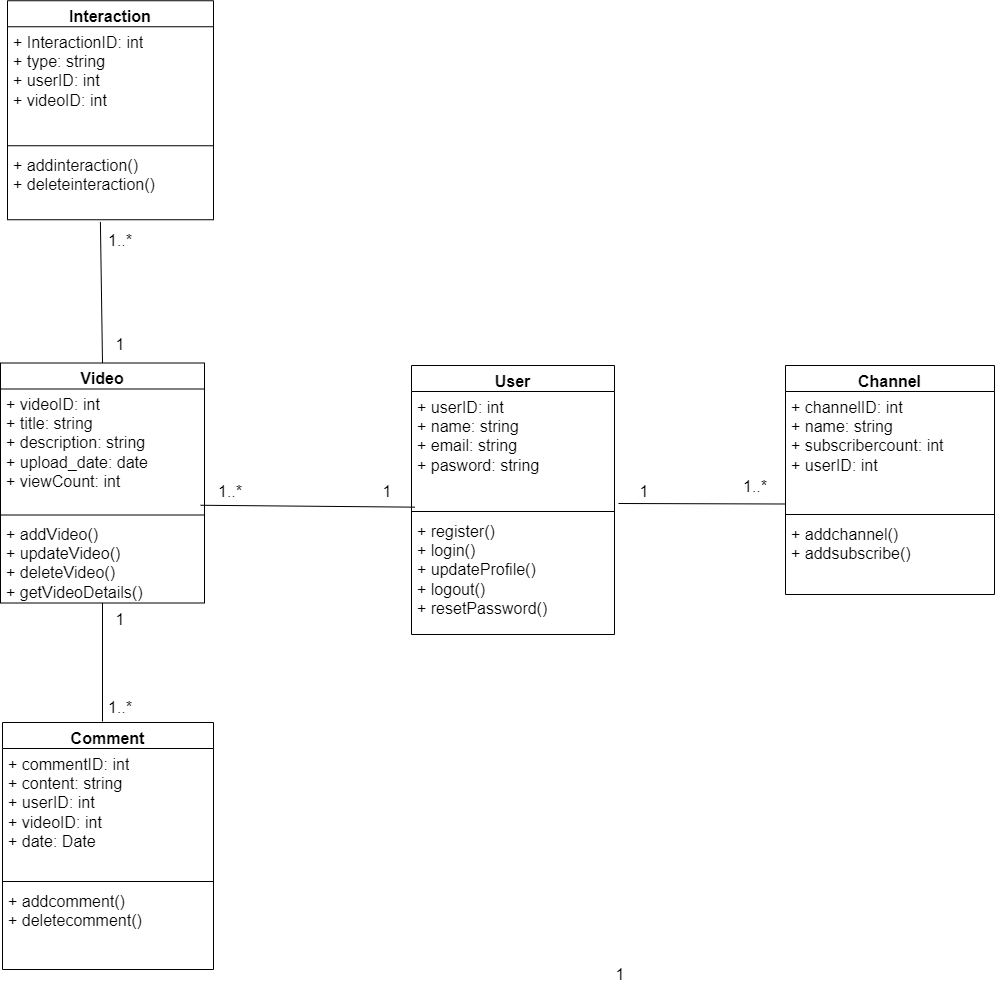
 **User Class**: Stores user information (id, name, email).

 **Video Class**: Stores video details (id, title, description).

 **Channel Class**: Manages channels and subscriptions.

 **Comment Class**: Contains comments associated with videos.

 **Interaction Class**: Represents actions like likes and shares.  
The diagram includes attributes and operations for each class to provide a detailed representation of the system’s design.



# Class Diagram Figure 8

# 4.2.4 BPMN Diagram: Video Processing Workflow

# Description of BPMN Diagram

# User Registration Process

# Start Event: User clicks on "Register".

# Task: User fills in registration details (username, email, password).

# Task: User submits registration form.

# Gateway: System checks if the email is unique.

# If Yes: System creates a new user account.

# If No: System displays an error message to the user.

# End Event: Registration process ends.

# Video Upload Process

# Start Event: User clicks on "Upload Video".

# Task: User fills in video details (title, description, category).

# Task: User uploads video file.

# End Event: Video upload process ends.

# Video Processing Process

# Start Event: Video upload completes.

# Task: System encodes video.

# Task: System generates video thumbnails.

# Task: System updates video status to "Processed".

# End Event: Video processing completes.

# Commenting on Videos Process

# Start Event: User views a video.

# Task: User fills in comment text.

# Task: User submits comment.

# End Event: Comment posting ends.

# Video Viewing Process

# Start Event: User initiates search for videos.

# Task: User selects a video from search results.

# Task: User views video.

# End Event: Video viewing process ends

# 

# BPMN Diagram Figure 9

# 4.2.5 DFD Diagrams

# 4.2.5.1 Context Diagram

# 

**Context Diagram Figure 10**

# Actors and System

# User: Represents an end-user who interacts with the system to upload and watch videos.

# System: Represents the central video management system that handles video uploads, storage, streaming, and content management.

# Admin: Represents an administrator who manages the content within the system.

# Interactions

# User to System: Upload/Watch Video: Users can upload videos to the system or watch videos that are already available. This interaction indicates that users initiate the upload process and request to watch videos.

# System to User: Stream Video: The system streams the video content to the user upon request. This interaction ensures that users receive the video data for playback.

# Admin to System: Manage Content: Admins interact with the system to manage the video content. This includes tasks such as approving uploads, removing inappropriate content, organizing video libraries, and handling metadata.

# 4.2.5.2 Level-0 Diagram

# 

**Level-0 Diagram Figure 11**

# 4.2.5.3 Level-1 Diagram

# 

**Level-1 Diagram Figure 12**

# 1. Context Diagram:

# User: Interacts with the system to upload and view videos.

# Video Sharing System: The core system that handles video uploads, processing, storage, and playback.

# External Services: Includes CDN for video streaming and CloudStorage for storing video files.

# 2. Level 0 DFD:

# 1.0 Video Upload: Handles the user's video upload request.

# 2.0 Video Processing: Processes the uploaded video (e.g., encoding).

# 3.0 Video Storage: Stores the processed video in cloud storage.

# 4.0 Video Playback: Manages video playback requests from users.

# 3. Level 1 DFD for Video Upload:

# 1.1 Select Video: User selects a video to upload.

# 1.2 Upload Video to Server: The selected video is uploaded to the server.

# 1.3 Authenticate User: Ensures the user is authenticated.

# 1.4 Confirm Upload: Confirms that the video upload is successful.

# 1.5 Save Metadata to VideoDB: Saves the video metadata to the VideoDB.

**4.2.6 E-R Diagram**

****

**E-R Diagram Figure 13**

**Relationships**

1. User and Video:

Relationship: A User can upload multiple Videos.

Type: One-to-Many

Foreign Key: UserID in Video table references UserID in User table.

1. User and Comment:

Relationship: A User can post multiple Comments on Videos.

Type: One-to-Many

Foreign Key: UserID in Comment table references UserID in User table.

1. Video and Comment:

Relationship: A Video can have multiple Comments.

Type: One-to-Many

Foreign Key: VideoID in Comment table references VideoID in Video table.

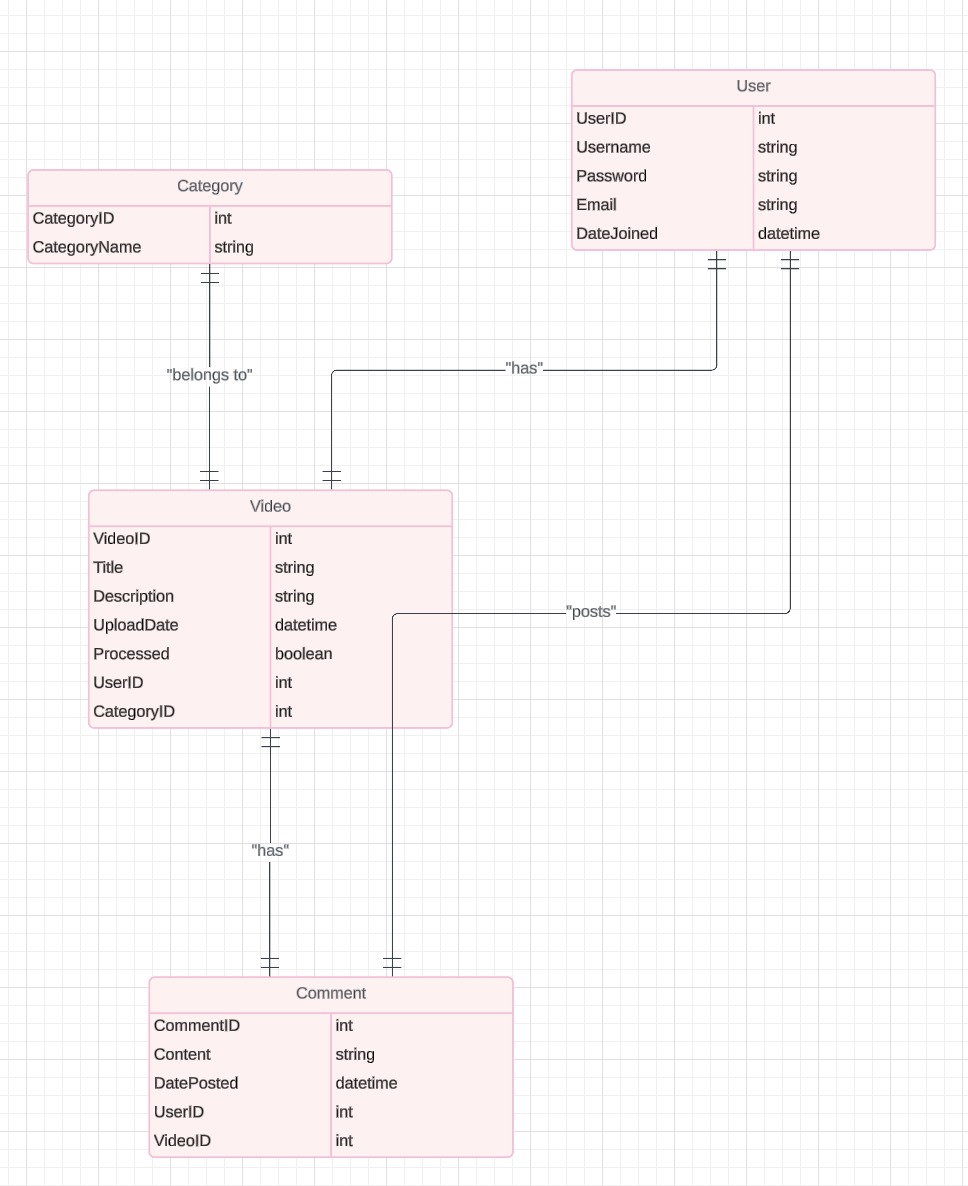
1. Category and Video:

Relationship: A Category can have multiple Videos.

Type: One-to-Many

Foreign Key: CategoryID in Video table references CategoryID in Category table.

**4.2.7 Relational Table**

****

**Relational Table Figure 14**

**SQL CODES**

CREATE TABLE User (

UserID INT PRIMARY KEY AUTO\_INCREMENT,

Username VARCHAR(50) NOT NULL,

Password VARCHAR(255) NOT NULL,

Email VARCHAR(100) NOT NULL UNIQUE,

DateJoined DATETIME DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE Video (

VideoID INT PRIMARY KEY AUTO\_INCREMENT,

Title VARCHAR(255) NOT NULL,

Description TEXT,

UploadDate DATETIME DEFAULT CURRENT\_TIMESTAMP,

Processed BOOLEAN DEFAULT FALSE,

UserID INT,

CategoryID INT,

FOREIGN KEY (UserID) REFERENCES User(UserID),

FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID)

);

CREATE TABLE Comment (

CommentID INT PRIMARY KEY AUTO\_INCREMENT,

Content TEXT NOT NULL,

DatePosted DATETIME DEFAULT CURRENT\_TIMESTAMP,

UserID INT,

VideoID INT,

FOREIGN KEY (UserID) REFERENCES User(UserID),

FOREIGN KEY (VideoID) REFERENCES Video(VideoID)

);

CREATE TABLE Category (

CategoryID INT PRIMARY KEY AUTO\_INCREMENT,

CategoryName VARCHAR(100) NOT NULL

);

**4.2.8 Physical Database Tables**



**Physical Database Tables Figure 15**

**Attributes for User table**:

UserID: INT, Primary Key, Auto Increment

Username: VARCHAR(50), Not Null

Password: VARCHAR(255), Not Null

Email: VARCHAR(100), Not Null, Unique

DateJoined: DATETIME, Default CURRENT\_TIMESTAMP

**Attributes for Video table**:

VideoID: INT, Primary Key, Auto Increment

Title: VARCHAR(255), Not Null

Description: TEXT

UploadDate: DATETIME, Default CURRENT\_TIMESTAMP

Processed: BOOLEAN, Default FALSE

UserID: INT, Foreign Key referencing User(UserID)

CategoryID: INT, Foreign Key referencing Category(CategoryID)

**Attributes for Comment table**:

CommentID: INT, Primary Key, Auto Increment

Content: TEXT, Not Null

DatePosted: DATETIME, Default CURRENT\_TIMESTAMP

UserID: INT, Foreign Key referencing User(UserID)

VideoID: INT, Foreign Key referencing Video(VideoID)

**Attributes for Category table**:

CategoryID: INT, Primary Key, Auto Increment

CategoryName: VARCHAR(100), Not Null

**THE SQL CODES**

CREATE TABLE User (

UserID INT PRIMARY KEY AUTO\_INCREMENT,

Username VARCHAR(50) NOT NULL,

Password VARCHAR(255) NOT NULL,

Email VARCHAR(100) NOT NULL UNIQUE,

DateJoined DATETIME DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE Video (

VideoID INT PRIMARY KEY AUTO\_INCREMENT,

Title VARCHAR(255) NOT NULL,

Description TEXT,

UploadDate DATETIME DEFAULT CURRENT\_TIMESTAMP,

Processed BOOLEAN DEFAULT FALSE,

UserID INT,

CategoryID INT,

FOREIGN KEY (UserID) REFERENCES User(UserID)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (CategoryID) REFERENCES Category(CategoryID)

ON DELETE SET NULL

ON UPDATE CASCADE,

INDEX (UserID),

INDEX (CategoryID)

);

CREATE TABLE Comment (

CommentID INT PRIMARY KEY AUTO\_INCREMENT,

Content TEXT NOT NULL,

DatePosted DATETIME DEFAULT CURRENT\_TIMESTAMP,

UserID INT,

VideoID INT,

FOREIGN KEY (UserID) REFERENCES User(UserID)

ON DELETE CASCADE

ON UPDATE CASCADE,

FOREIGN KEY (VideoID) REFERENCES Video(VideoID)

ON DELETE CASCADE

ON UPDATE CASCADE,

INDEX (UserID),

INDEX (VideoID)

);

CREATE TABLE Category (

CategoryID INT PRIMARY KEY AUTO\_INCREMENT,

CategoryName VARCHAR(100) NOT NULL

);

**Normalization**

Each table has been normalized up to the Third Normal Form (3NF):

1. First Normal Form (1NF):

* Ensured that each column contains atomic (indivisible) values.
* Each table has a primary key.

1. Second Normal Form (2NF):

* Removed partial dependencies; all non-key attributes are fully functional dependent on the primary key.

1. Third Normal Form (3NF):

* Removed transitive dependencies; non-key attributes do not depend on other non-key attributes.

# 5. IMPLEMENTATION

## 5.1 Tools, technologies and platforms used

For the development of our web and mobile application, we utilized a combination of modern tools, technologies, and platforms to ensure a seamless and efficient development process:

Programming Languages: HTML, CSS, JavaScript

Frameworks and Libraries: Bootstrap for responsive design, jQuery for simplified DOM manipulation

Version Control: Git for source code management and collaboration

Development Tools: Visual Studio Code as the primary code editor, Chrome Developer Tools for debugging

Testing Tools: Jasmine or Mocha for unit testing, Selenium for end-to-end testing

Build and Deployment: GitHub for code hosting, Netlify for continuous deployment

Design Tools: Adobe XD for UI/UX design and prototyping

These tools and technologies were chosen to facilitate the development of a robust, user-friendly application that works well on both web and mobile platforms.

## 5.2 Algorithms

|  |  |
| --- | --- |
| **Algorithm** | **Description** |
| **Video Upload** | This algorithm handles the process of uploading videos to the system. |
|  |  |
| 1. Select Video | The user selects the video file to upload from their device. |
| 2. Validate Format | Check if the selected file format is supported (e.g., MP4, AVI). |
| 3. Validate Size | Ensure the selected video file size does not exceed the maximum allowed size. |
| 4. Process Upload | Begin the upload process to transfer the video file to the server. |
| 5. Store Video | Save the uploaded video file to a designated storage location on the server. |
| 6. Update Database | Update the database with metadata related to the uploaded video (e.g., title, description, upload date). |
| **Video Playback** | This algorithm manages the playback of videos within the system. |
|  |  |
| 1. Select Video | User selects a video to play from the available options. |
| 2. Retrieve Metadata | Fetch metadata (e.g., title, duration) associated with the selected video from the database. |
| 3. Retrieve Video | Retrieve the video file from the server storage based on the selected video's unique identifier. |
| 4. Prepare Player | Initialize the video player component and load the video for playback. |
| 5. Playback Controls | Provide user controls (e.g., play, pause, seek) for interacting with the video during playback. |
| **User Authentication** | This algorithm handles user authentication and authorization processes. |
|  |  |
| 1. User Login | User provides credentials (username, password) to access the system. |
| 2. Validate Credentials | Verify the provided credentials against stored user data in the database. |
| 3. Generate Token | Upon successful validation, generate a secure authentication token for the user session. |
| 4. Access Control | Check the user's role and permissions to determine access rights for various system features. |
| 5. Session Management | Maintain user session state and handle logout actions to invalidate the authentication token. |

**Algorithms Table 10**

These algorithms provide a structured overview of the main functions of the system, including video upload, playback, and user authentication. Each step outlines the key actions and processes involved in performing the respective functions.

## 5.3 Standards

For our project, the following standards are applicable:

HTML and CSS: W3C Standards for Markup and Styling.

JavaScript: ECMAScript Standards for Scripting Language.

Accessibility: WCAG (Web Content Accessibility Guidelines) for ensuring accessibility.

Security: OWASP (Open Web Application Security Project) guidelines for web application security.

Responsive Design: Followed industry best practices for responsive web design.

These standards were adhered to ensure compatibility, accessibility, security, and best practices in our project development process.

## 5.4 Detailed description of the implementation (coding)

# Algorithms

# For the main functions of our system, we employed the following algorithms:

# Algorithm 1: User Authentication

# Input: Username, Password

# Procedure:

# Check if the username exists in the database.

# If the username exists, retrieve the corresponding password hash from the database.

# Hash the provided password and compare it with the stored password hash.

# If the hashes match, authenticate the user and grant access.

# If the hashes do not match, deny access and display an error message.

# Output: Authentication status (Authenticated/Denied)

# Algorithm 2: Video Upload

# Input: Video File, Metadata (Title, Description, Tags)

# Procedure:

# Validate the video file format and size.

# Generate a unique identifier for the video.

# Save the video file to the server storage.

# Store the metadata (title, description, tags) in the database along with the video identifier.

# Update the user's video library with the uploaded video.

# Output: Success/Failure message

# Algorithm 3: Video Recommendation

# Input: User's Watch History, User Preferences

# Procedure:

# Analyze the user's watch history to identify preferences and viewing patterns.

# Retrieve videos from the database that match the user's preferences.

# Apply a recommendation algorithm (e.g., collaborative filtering, content-based filtering) to suggest relevant videos.

# Filter out already watched videos and present the top recommendations to the user.

# Output: List of recommended videos

# These algorithms provide the core functionality of our system, including user authentication, video uploading, and personalized video recommendations.

# 6. QUALITY AND TESTING

## 6.1 Quality Assurance Activities During Project Life Cycle

**1.Brainstorming**

**Key Features**: Video upload, streaming, personalized recommendations, user authentication, comments, and subscriptions.

**Challenges**: Handling high traffic, ensuring low latency for video streaming, and managing large-scale storage.

In a brainstorming session, team members can suggest creative solutions like integrating a CDN for faster streaming, using AI for recommendations, or leveraging cloud services for scalable infrastructure.

|  |  |  |  |
| --- | --- | --- | --- |
| **Brainwriting 6-3-5 Worksheet** | | | |
| **Job To Be Done:** | **Design a video streaming platform similar to YouTube.** | **Date:** | **12-2024** |
| **Team:** | **1** |
| **Member** | **Pelin** |
| **1** | **2** | **3** | |
| Allow users to upload videos up to 1GB with minimal latency. | Use cloud storage (e.g., AWS S3) for scalable and reliable video storage. | Implement a simple drag-and-drop interface for uploading videos. | |
| |  | | --- | | Provide video playback in multiple resolutions. |  |  | | --- | |  | | |  | | --- | | Auto-adjust video quality based on user bandwidth. |  |  | | --- | |  | | |  | | --- | | Include a manual quality selection feature for users. |  |  | | --- | |  | | |
| |  | | --- | | Build a recommendation engine for personalized feeds. |  |  | | --- | |  | | |  | | --- | | Use collaborative filtering algorithms for better suggestions. |  |  | | --- | |  | | |  | | --- | | Analyze viewing history to improve recommendation accuracy. |  |  | | --- | |  | | |
| |  | | --- | | Implement a robust user authentication system. |  |  | | --- | |  | | |  | | --- | | Use OAuth 2.0 for secure login and account linking. |  |  | | --- | |  | | |  | | --- | | Include two-factor authentication for added security. |  |  | | --- | |  | | |
| |  | | --- | | Develop a mobile app for video streaming. |  |  | | --- | |  | | |  | | --- | | Optimize the app for offline viewing by enabling downloads. |  |  | | --- | |  | | |  | | --- | | Ensure cross-platform compatibility (iOS, Android, web). |  |  | | --- | |  | | |
| |  | | --- | | Monitor user activity and gather analytics. |  |  | | --- | |  | | |  | | --- | | Visualize analytics via dashboards to track usage patterns. |  |  | | --- | |  | | Send weekly performance reports to content creators. | |

**Brainstorming Table 11**

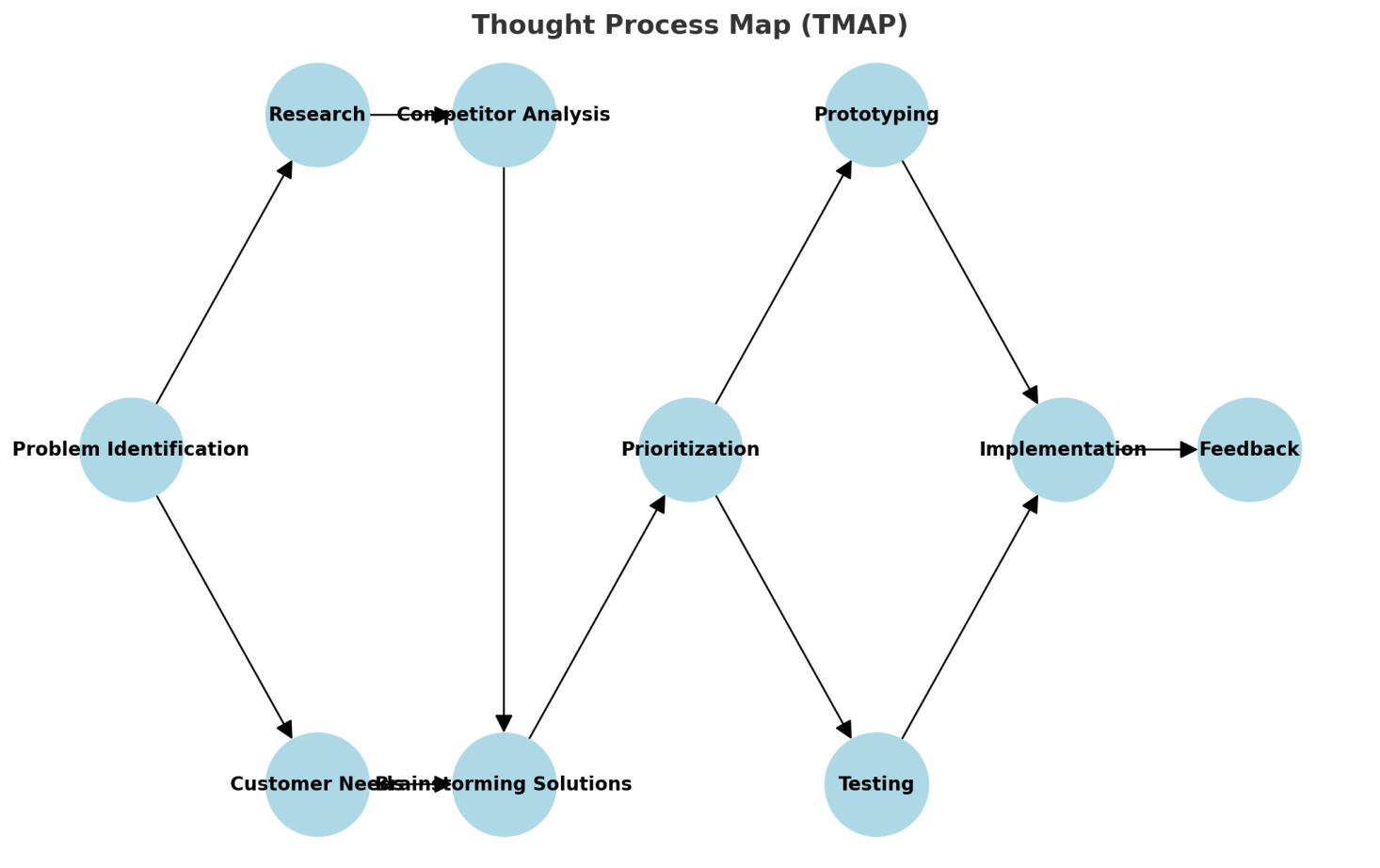
**6.2 Thought Process Map (TMAP)**

 **User Need**: Users want smooth video streaming.

* **Requirement**: Low-latency content delivery.
* **Implementation**: Use a CDN to distribute video content globally.

 **User Need**: Personalized recommendations.

* **Requirement**: User data analysis.
* **Implementation**: Use machine learning algorithms to process watch history and suggest content.



**Thought Process Map (TMAP) Figure 16**

**6.3 Kano Model**

 **Must-Have Features**:

* Video upload and playback.
* User authentication and profile management.
* Commenting and liking on videos.

 **Performance Features**:

* Adaptive streaming for varying internet speeds.
* Fast search and filtering options.

 **Delighters**:

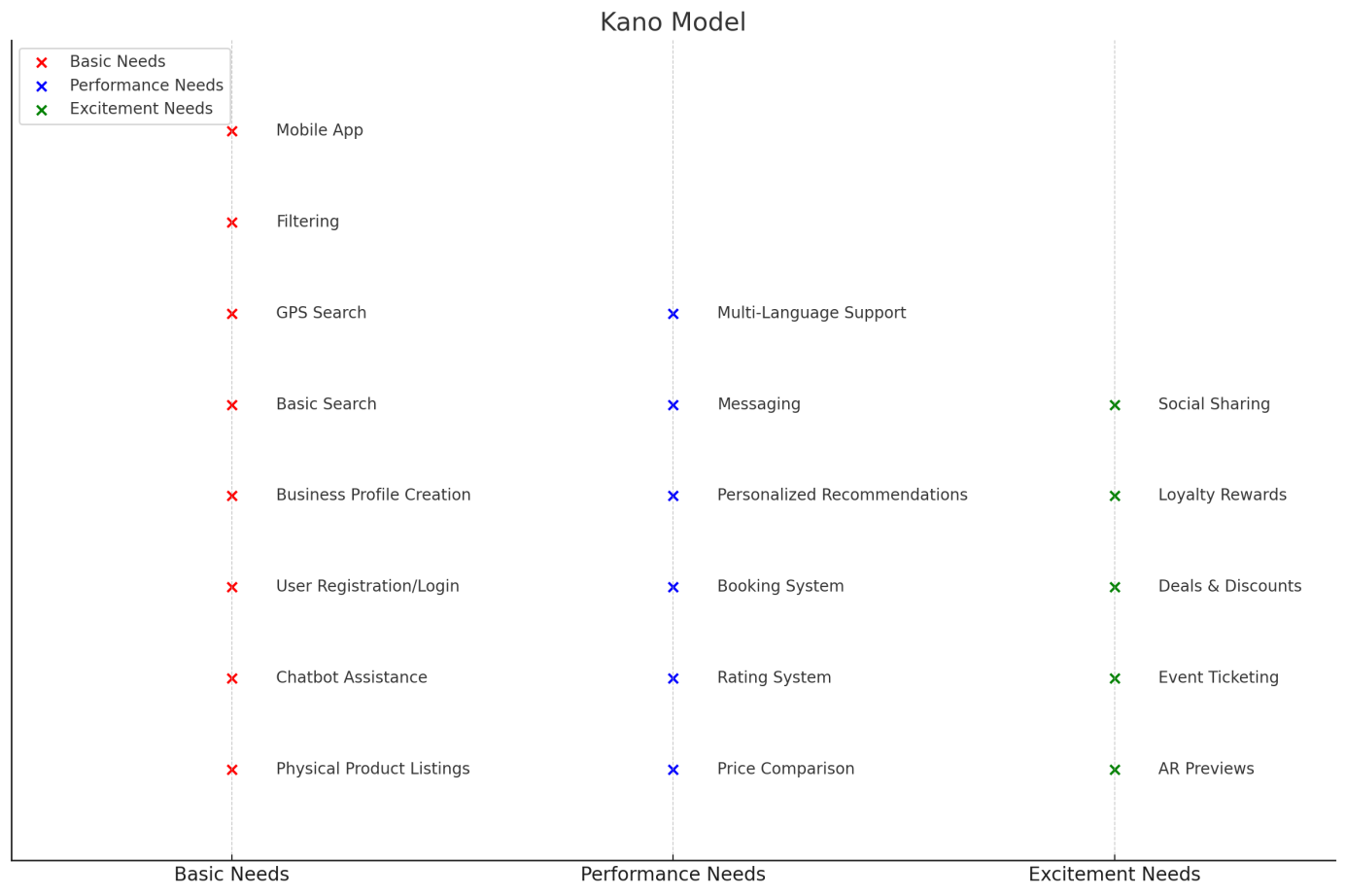
* AI-driven personalized recommendations.
* Social sharing integrations.

 **Indifferent Features**:

* Multiple playback speed options for casual users.

 **Reverse Features**:

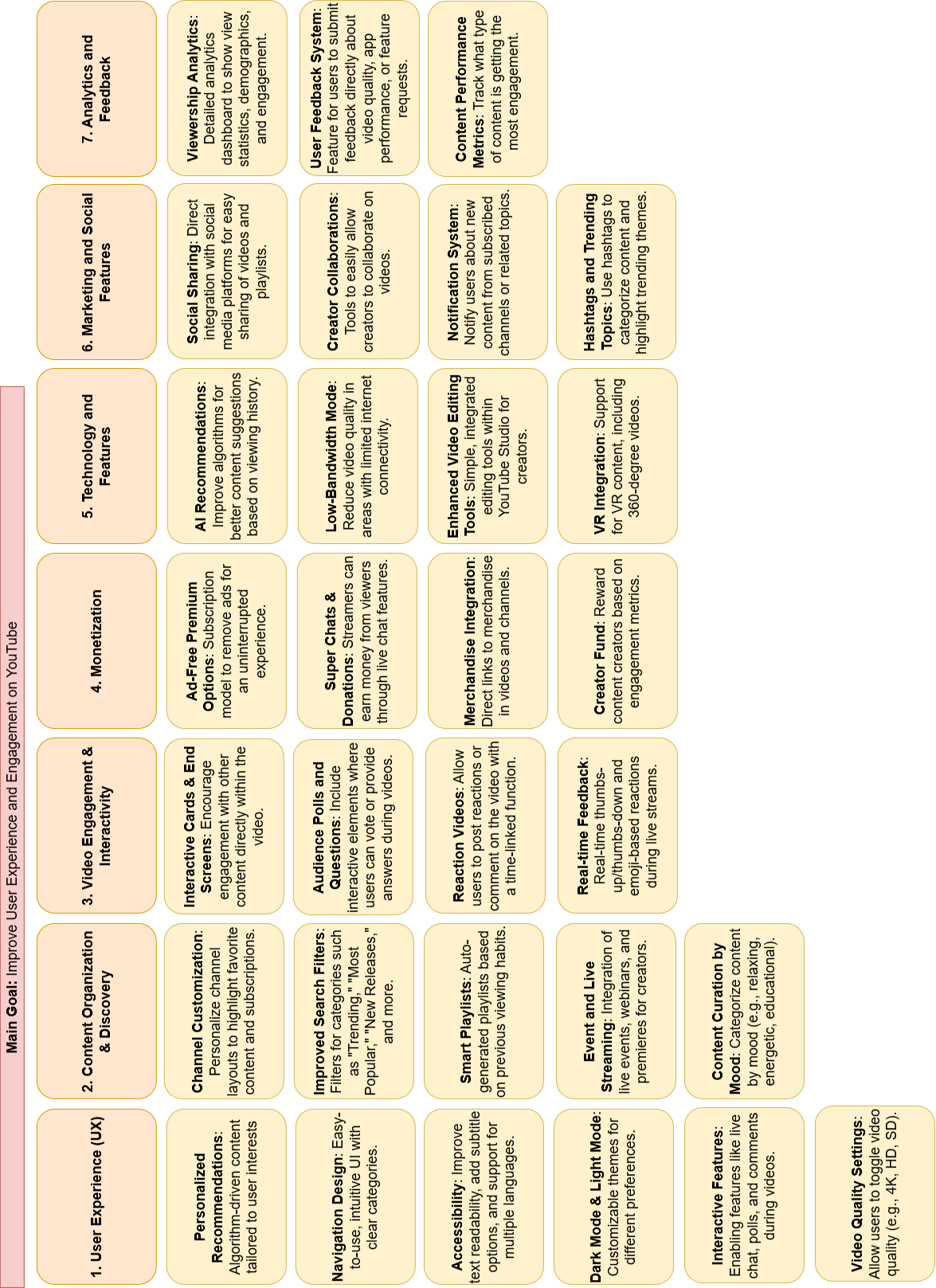
* Autoplay of unrelated videos, which may annoy users.  
  By categorizing features, you can allocate resources to maximize user satisfaction.

****

**Kano Model Figure 17**

**6.4 Affinity diagram**

* **Feedback Categories**:
  + **Streaming Issues**: Buffering, resolution changes.
  + **User Interface Requests**: Easier navigation, dark mode.
  + **Content Management**: Playlist creation, content sorting.
  + **Social Features**: Improved commenting system, community tabs.
* **Idea Clusters**:
  + **Backend Optimization**: Scalable storage, video encoding efficiency.
  + **Frontend Enhancements**: Faster loading UI, mobile-friendly design.  
    Affinity diagrams help consolidate user insights and development priorities.



**Affinity Diagram Figure 18**

**6.5 Pareto Chart**

 **User Complaints Analysis**:

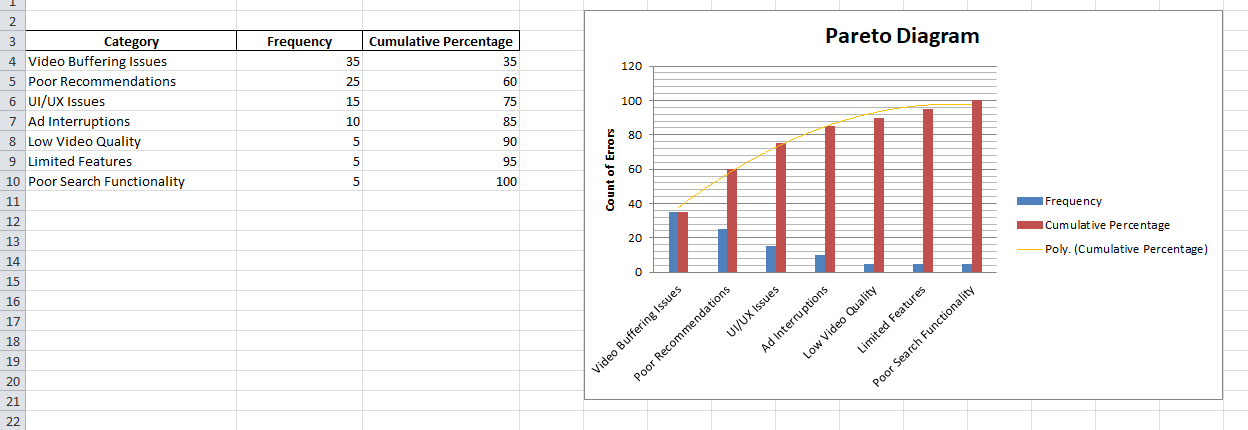
* **80% of complaints** might stem from **20% of issues**, such as buffering or login errors.
* Address these top contributors first to improve overall satisfaction.

 **Feature Usage**:

* **80% of user engagement** might come from **20% of features**, like video playback and recommendations.
* Prioritize optimizing these critical features.

 **Resource Allocation**:

* **80% of costs** might come from **20% of components**, like CDN usage or storage.
* Optimize these areas to reduce operational expenses.  
  The Pareto Chart ensures your efforts are directed where they yield the most impact.



**Pareto Chart Figure 19**

**6.6 SWOT Analsis**

|  |  |
| --- | --- |
| **Strengths (Internal, Positive Factors)** | **Weaknesses (Internal, Negative Factors)** |
|  **Large User Base**: YouTube already has a massive and diverse global user base, providing a strong foundation for any new features or improvements.   **Personalization**: Advanced AI-driven recommendations enhance user experience, helping people discover relevant content.   **Monetization Opportunities**: Established systems for monetization benefit both creators and YouTube itself.   **Integrated Features**: Features like video comments, likes/dislikes, and community posts provide a higher level of engagement.   **Content Diversity**: A vast array of content from different creators and industries keeps users engaged across various topics and formats.   **Live Streaming**: The integration of live events and streaming provides real-time interaction, increasing user retention. |  **Ad Interruptions**: Frequent ads can disrupt user experience, leading to potential dissatisfaction, particularly for free-tier users.   **UI/UX Challenges**: The interface can feel cluttered, especially with frequent changes and a wide variety of features.   **Algorithmic Limitations**: YouTube's recommendation system is not always accurate, sometimes recommending irrelevant or repetitive content.   **Video Quality Variability**: In some regions, users face issues with video buffering or the inability to watch videos in high quality.   **Customer Support**: User complaints about slow or unhelpful customer service could deter users from engaging more with the platform.   **Overload of Content**: With millions of videos, users can feel overwhelmed by the vast amount of content available. |
| **Opportunities (External, Positive Factors)** | **Threats (External, Negative Factors)** |
|  **Expansion in Emerging Markets**: YouTube can further penetrate emerging markets with tailored content, language support, and affordability.   **Mobile Optimization**: Given the global increase in mobile device usage, optimizing the app for better mobile experiences can increase engagement.   **Interactive and Immersive Technologies**: Integrating new technologies like AR (augmented reality), VR, or more interactive features could offer users a more immersive experience.   **Partnerships with Content Creators**: Partnering with popular influencers and creators to develop exclusive content or live events could attract more viewers and subscribers.   **Integration with Other Platforms**: Expanding integrations with social media platforms and other content services (e.g., gaming, education) can increase reach.   **Enhanced Monetization Models**: Experimenting with new monetization strategies, such as premium memberships or more interactive advertising, could benefit creators and YouTube. |  **Competition from Other Platforms**: Platforms like TikTok, Instagram, and Twitch are drawing users away with shorter content and real-time interactivity.   **Privacy and Data Concerns**: As privacy concerns rise, YouTube could face regulatory pressure regarding user data, particularly with minors.   **Changing Consumer Habits**: Audiences are increasingly shifting from long-form video to short-form video content, which could make YouTube's traditional format less appealing to some users.   **Ad Blockers**: The use of ad blockers could undermine YouTube’s main revenue model, especially if users continue to avoid ads.   **Content Moderation Challenges**: With the growing issue of misinformation, content moderation on YouTube may face increased scrutiny, possibly affecting the platform's image.   **Legal and Copyright Issues**: YouTube has to constantly deal with copyright claims and potential lawsuits from content creators or rights holders. |

**Swot Analysis Table 12**

1. **Quality Metric Tables**

|  |  |  |
| --- | --- | --- |
| QUALITY METRIC NUMBER | QUALITY METRIC NAME | DESCRIPTION |
| 1 | Video Playback Quality | Measures the video quality during playback, ensuring smooth performance with minimal buffering, and supporting high-definition formats (1080p, 4K). |
| 2 | |  | | --- | | Average Load Time |  |  | | --- | |  | | The time it takes for a video to load and start playing, should be less than 2 seconds on a fast internet connection. |
| 3 | |  | | --- | | User Engagement Rate |  |  | | --- | |  | | The ratio of likes, comments, and shares to total views, aiming for a 20% engagement rate for organic content. |
| 4 | Search Function Accuracy | Measures how accurately the search engine provides relevant video results, aiming for 90% relevance based on keywords. |
| 5 | Ad Interruption Frequency | Tracks how often users experience ad interruptions during video playback, aiming for no more than one ad every 15 minutes for free users. |
| 6 | Customer Support Response Time | The average time it takes to respond to user inquiries through YouTube support, aiming for a response time of less than 2 hours. |
| 7 | App Crash Rate | Measures the frequency of app crashes or freezes, aiming for a crash rate of less than 1% of users. |
| 8 | Content Freshness | |  | | --- | |  |  |  | | --- | | The frequency of new content being uploaded or recommended to users, aiming for 5-10 fresh, relevant recommendations per user session. | |
| 9 | Mobile App Performance | Assesses the responsiveness and speed of the mobile app, with the goal of maintaining a performance rating of 4.5 stars or higher on app stores. |
| 10 | Multi-language Support | Measures the availability of YouTube in different languages, ensuring at least 50 languages are supported for a global user base. |
| 11 | User Retention Rate | Tracks the percentage of users who return to the platform within 30 days after their first visit, aiming for a retention rate of 70% or higher. |
| 12 | Personalization Accuracy | Measures the effectiveness of personalized content recommendations, aiming for 85% user satisfaction in terms of relevance. |
| 13 | UI/UX Satisfaction | Measures user satisfaction with the interface design, aiming for a score of 4.0 or higher on user feedback surveys. |
| 14 | Buffering Time | The average buffering time during video playback, aiming for a maximum of 3 seconds per session for high-quality videos. |
| 15 | Video Resolution | The percentage of videos available in HD or higher, aiming for at least 90% of content available in 720p or 1080p quality. |
|  |  |  |

**Quality Metric Tables 13**

1. **Quality Checklist Tables**

|  |  |
| --- | --- |
| CHECK | ISSUE/TOPIC |
| √ | Verify that videos are uploaded within acceptable time frames for user experience. |
| √Test smooth video streaming | Confirm no buffering issues across supported resolutions and devices. |
| √Enable video quality adjustment | Ensure seamless quality changes without impacting streaming. |
| √ Enforce file size limitations | Validate that uploaded videos do not exceed the 1GB limit. |
| √ Encrypt video files | Ensure all videos are encrypted during upload and storage for security. |
| √ Leverage cloud infrastructure | Optimize performance and scalability using cloud services. |
| √ Verify availability and reliability | Conduct stress tests to confirm system meets high availability requirements. |

**Quality Checklist Tables 14**

1. **Quality Audit Checklist Tables**

|  |  |  |
| --- | --- | --- |
| QUALITY ASSURANCE FUNCTIONS | YES | NO |
| Validate compliance with upload time standards | √ |  |
| Assure smooth playback across all platforms | √ |  |
| Check scalability under increasing user load | √ |  |
| Verify proper encryption protocols | √ |  |
| Test video quality adjustment functionality | √ |  |
| Ensure file size and format compliance | √ |  |
| Confirm integration of cloud infrastructure | √ |  |
|  |  |  |

**Quality Audit Checklist Tables 15**

**Audit plans:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Audit Date | Planned/  Unplanned Audits | Number of Non-Compliant Issues | | Number of Actions Taken on  Non-Compliant Issues | | # of High-Priority Risk Items in Project Risk List | Hours to Prepare & Apply the Audit | Auditor |
| 15/04/24 | Planned | 2 |  | 4 |  | 1 | 8 | Zpç |
| 30/06/24 | Planned | 3 |  | 5 |  | 2 | 12 | Zpç |
| 10/09/24 | Unplanned | 1 |  | 2 |  | 1 | 10 | Zpç |
| 15/12/24 | Planned | 4 |  | 6 |  | 3 | 15 | Zpç |

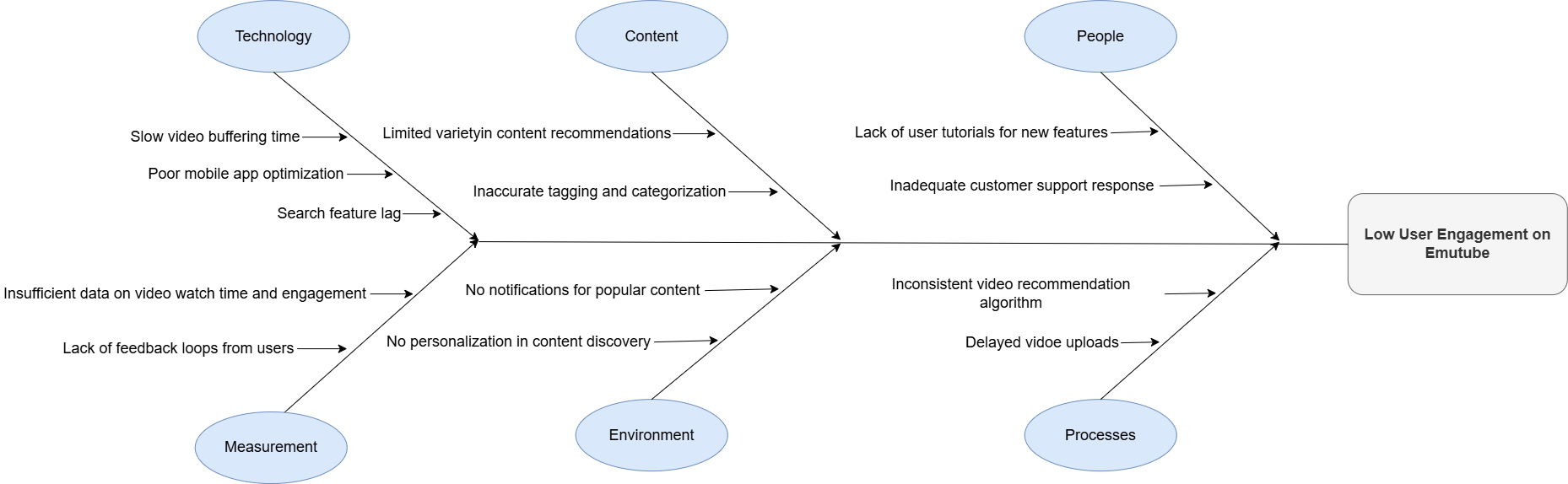
**Audit Plans Table 16**

**Audit plans 2:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item | Attribute | Relevance | Practice | Assessment |
| 1 | Project Planning |  |  |  |
| 1.1 | Does the project have a formal Project Plan? | 5 |  |  |
| 1.2 | Are the key elements of a Project Plan present? |  |  |  |
| a. | Project Definition and Scope? | 5 |  |  |
| b. | Project Objectives? | 5 |  |  |
| c. | Cost/Benefit Analysis? | 4 |  |  |
| d. | Personnel? | 3 |  |  |
| e. | Schedule? | 5 |  |  |
| f. | Risk Analysis? | 5 |  |  |
| g. | Critical Success Criteria? | 5 |  |  |
| 1.3 | Have all stakeholders been identified? | 5 |  |  |
| 1.4 | Is there a Stakeholder Management Plan? Are project responsibilities and accountabilities clearly defined? | 4 |  |  |
| 1.5 | Have the scope, objectives, costs, benefits, and impacts been communicated to all? | 4 |  |  |
| 2 | Design and Development |  |  |  |
| 2.1 | Has the system architecture been? | 5 |  |  |
| 2.2 | Does the design meet scalability and reliability requirements? | 5 |  |  |
| 2.3 | Are cloud services (e.g. AWS, Google Cloud) effectively integrated? | 4 |  |  |
| 2.4 | Are encryption and security measures implemented for videos and user data? | 5 |  |  |
| 2.5 | Are development milestones clearly defined and tracked? | 4 |  |  |
| 3 | Testing and Quality AssuranceTesting and Quality Assurance |  |  |  |
| 3.1 | Is there a formal test plan? | 5 |  |  |
| 3.2 | Are performance metrics (e.g., loading speed, buffering time) monitored and analyzed? | 5 |  |  |
| 3.3 | Is user feedback collected and incorporated into the testing process? | 4 |  |  |
| 3.4 | Have all system components been tested for integration and compatibility? | 5 |  |  |
| 3.5 | Is there a defect tracking system and are defects resolved promptly? | 4 |  |  |
| 4 | Implementation and Deployment |  |  |  |
| 4.1 | Have deployment strategies (e.g., phased deployment, full deployment) been defined? | 5 |  |  |
| 4.2 | Is there a rollback plan in case of deployment failures? | 5 |  |  |
| 4.3 | Are operational metrics (e.g., system uptime, error rates) monitored post-deployment? | 5 |  |  |
| 4.4 | Is the platform optimized for both mobile and web clients? | 4 |  |  |
| 4.5 | Are staff and stakeholders trained on system operations? | 3 |  |  |
| 5 | Project Risk ManagementProject Risk Management |  |  |  |
| 5.1 | Have potential risks (e.g., scalability, data breaches) been identified and documented? | 5 |  |  |
| 5.2 | Is there a risk mitigation plan for high priority risks? | 5 |  |  |
| 5.3 | Are project risks regularly reviewed and updated? | 4 |  |  |
| 6 | Post-Implementation Review |  |  |  |
| 6.1 | Has the platform been reviewed to meet initial project objectives and success criteria? | 5 |  |  |
| 6.2 | Are lessons learned documented for future improvements? | 4 |  |  |
| 6.3 | Are customer satisfaction surveys conducted and analyzed? | 4 |  |  |
| 6.4 | Is an ongoing support and maintenance agreement in place? | 5 |  |  |

**Audit Plans Table 17**

1. **Quality Process Analysis (QPA)**



**Figure 20**. A Fishbone diagram to assess potential project problems.

## 6.1 Quality Control (QC) Activities After Implementation Completed

**System Testing**:

* The system was tested using a variety of scenarios to ensure all modules functioned as expected.
* **Functional Tests**: Verified the ability to upload videos, stream smoothly, and adjust video quality.
* **Performance Tests**: Evaluated system scalability, streaming latency, and responsiveness under high user loads.
* **Security Tests**: Checked encryption protocols for uploaded videos and secure handling of user data.

**Test Data**:

* **Upload Scenarios**:
  + Test videos of varying sizes (e.g., 100MB, 500MB, and 1GB).
  + Formats such as MP4, AVI, and MOV.
* **Streaming Scenarios**:
  + Buffer times measured under different resolutions (480p, 720p, 1080p).
  + Simulated user interactions from web and mobile devices.
* **Edge Cases**:
  + Exceeding the 1GB file size limit.
  + Simulating poor network conditions.

**Deficiencies/Errors Identified**:

* **Upload Module**:
  + Slow upload speeds for high-resolution videos during peak times.
  + Failed uploads when file names included special characters.
* **Streaming Module**:
  + Occasional buffering for 1080p videos on low-bandwidth connections.
  + Streaming quality not adjusting seamlessly when bandwidth fluctuated.
* **Security**:
  + Encryption not consistently applied to some test files.

**Corrections Made**:

* Optimized server configurations and upgraded to a faster cloud storage solution.
* Implemented filename sanitization to handle special characters.
* Enhanced adaptive streaming logic for dynamic resolution adjustments.
* Ensured all video uploads passed through a uniform encryption pipeline.

**Verification and Validation**:

* **User Testing**: Conducted beta tests with a diverse group of users to gather real-world feedback.
* **Automated Testing**: Used scripts to continuously test upload and streaming functionality.
* **Monitoring**: Established real-time monitoring to track system performance and identify issues promptly.
* **Post-Launch Reviews**: Analyzed usage patterns and resolved any reported bugs or performance bottlenecks.

1. **Test Schedule**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Date** | **Test Activity** | **Objective** | **Assigned To** | **Duration** | **Status** |
| **12/24** | |  | | --- | | **Unit Testing for Upload Module** |  |  | | --- | |  | | |  | | --- | | **Verify upload functionality for various file sizes and formats.** |  |  | | --- | |  | | |  | | --- | | **Developer Team** |  |  | | --- | |  | | **8** | **Planned** |
| **12/24** | |  | | --- | | **Unit Testing for Streaming Module** |  |  | | --- | |  | | |  | | --- | | **Test video playback quality and buffering time.** |  |  | | --- | |  | | **QA Team** | **6** | **Planned** |
| **12/24** | |  | | --- | | **Integration Testing** |  |  | | --- | |  | | |  | | --- | | **Ensure modules (upload, playback, encryption) work together.** |  |  | | --- | |  | | **QA Team** | **10** | **Planned** |
| **12/24** | |  | | --- | | **Performance Testing** |  |  | | --- | |  | | |  | | --- | | **Measure system scalability and response times under high load.** |  |  | | --- | |  | | **Performance Team** | **12** | **Planned** |
| **12/24** | |  | | --- | | **Security Testing** |  |  | | --- | |  | | |  | | --- | | **Validate encryption and data protection protocols.** |  |  | | --- | |  | | **Security Team** | **8** | **Planned** |
| **12/24** | |  | | --- | | **User Acceptance Testing (UAT)** |  |  | | --- | |  | | |  | | --- | | **Ensure system meets end-user expectations.** |  |  | | --- | |  | | **UAT Team** | **16** | **Planned** |
| **12/24** | |  | | --- | | **Regression Testing** |  |  | | --- | |  | | |  | | --- | | **Verify previously tested functionalities after bug fixes.** |  |  | | --- | |  | | **QA Team** | **8** | **Planned** |
|  | |  | | --- | | **Final System Validation** |  |  | | --- | |  | | |  | | --- | | **Conduct end-to-end validation of the system.** |  |  | | --- | |  | | **QA Team** | **12** | **Planned** |

**Test Schedule Table 18**

1. **Inspection Tables**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Inspection Item** |  |  | | --- | |  | | **Criteria** | |  | | --- | | **Inspection Method** |  |  | | --- | |  | | |  | | --- | | **Pass/Fail** |  |  | | --- | |  | | **Remarks** |
| |  | | --- | | **System Architecture** |  |  | | --- | |  | | |  | | --- | | Scalability, reliability, and adherence to standards. |  |  | | --- | |  | | Peer review and architectural walkthroughs. | Pass | Meets all defined requirements. |
| |  | | --- | | **Upload Module** |  |  | | --- | |  | | File size limits, speed, and format compatibility. | Automated upload tests and edge case scenarios. | Fail | File size >1GB failed; needs adjustment. |
| **Playback Module** | Buffer time and resolution adaptability. | Performance testing under varied network conditions. | Pass | No buffering under normal conditions. |
| **UI Design** | Consistency, usability, and responsiveness. | Usability tests and expert reviews. | Pass | User-friendly and responsive interface. |
| **Database Design** | Normalization, efficiency, and integrity. | SQL performance benchmarks and schema reviews. | Pass | All queries executed efficiently. |
| **Security Measures** | Encryption and data protection protocols. | Penetration testing and code review. | Fail | Inconsistent encryption in some files. |
| **Integration Tests** | Seamless interaction between modules. | Integration testing tools and simulation. | Pass | All components integrated successfully. |
| **Documentation** | Clarity, accuracy, and completeness. | Peer reviews and usability testing with end-users. | Pass | Comprehensive and clear documentation. |

**Inspection Table 19**

1. **Test-Cases for Critical Modules**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Module Name** | **TC Scenario** | **Test Case Name** | **Pre-requisite** | **Test Data** | **Test Steps** | **Expected**  **Results** | **Status** | **Comments** | **Test Executed By** |
| TC001 | Upload Module | Validate video upload functionality | Upload Functionality Test | Cloud server configured | 500MB video file (MP4) | 1. Open upload page.  2. Select a 500MB MP4 file.  3. Click upload.  4. Monitor upload progress and completion. | File uploads successfully within acceptable time frame. | Planned | Ensure server is online. | QA Tester |
| TC002 | Playback Module | Verify smooth playback and buffering behavior | Playback Smoothness Test | Video files uploaded | High-resolution (1080p) video | 1. Open video.  2. Play at 1080p.  3. Observe for buffering.  4. Change resolution to 720p.  5. Check transition smoothness. | Video plays smoothly without buffering. | Planned | Test under different networks. | QA Tester |
| TC003 | Security Module | Ensure video encryption during upload | Encryption Validation | Encryption service active | Test video file (any format) | 1. Upload a test video.  2. Inspect uploaded file on server.  3. Verify encryption status using security tools. | File is encrypted on the server. | Planned | Perform penetration testing. | Security Analyst |
| TC004 | Integration | Test interaction between upload and playback | Upload-Playback Workflow | All modules deployed | Multiple video files | 1. Upload video files.  2. Play uploaded videos.  3. Validate playback quality.  4. Check file integrity post-upload. | Uploaded videos play without errors. | Planned | Ensure all modules are deployed. | QA Engineer |
| TC005 | User Interface | Verify responsiveness across devices | UI Responsiveness Test | Devices ready for testing | Various screen sizes and devices | 1. Open platform on desktop, tablet, and mobile.  2. Navigate through UI.  3. Check layout consistency and responsiveness. | UI adjusts perfectly across all devices. | Planned | Test with a variety of devices. | QA Tester |

**Test-Cases for Critical Modules Table 20**

# 7. USER GUIDE OF THE SYSTEM

In this section, we will explain interfaces of each user;

**7.1 About Page**

# 

**About Page Figure 21**

Description: A page that provides information about the platform, its mission, and its creators. It often includes a brief history of the platform and contact information.Key Features: Platform overview, mission statement, team information, and contact details.

# 7.2 Channel Page

# 

**Channel Page Figure 22**

Description: The channel page showcases a specific user's uploaded videos, playlists, and channel information. Users can subscribe to the channel and view the channel's content.Key Features: Channel banner, subscribe button, video list, playlists, and channel description.

# 7.3 Community Page

# 

**Community Page Figure 23**

Description: A page where users can engage with the community by posting messages, polls, and updates. This page serves as a hub for user interaction and community engagement.Key Features: Community posts, likes, comments, and share options.

# 7.4 Community Create Page

# 

**Community Ccreate Page Figure 24**

Description: The page where users can create new community posts, including text updates, images, and polls. It provides a platform for users to share content and engage with their audience.Key Features: Text editor, image upload, poll creation, and post button.

# 7.5 Index Page

# 

**Index Page Figure 25**

Description: The homepage of the platform where users can view a feed of recommended videos based on their preferences and watch history. It also provides access to trending videos and various video categories. Key Features: Browse videos, search bar, access to trending content, and video categories.

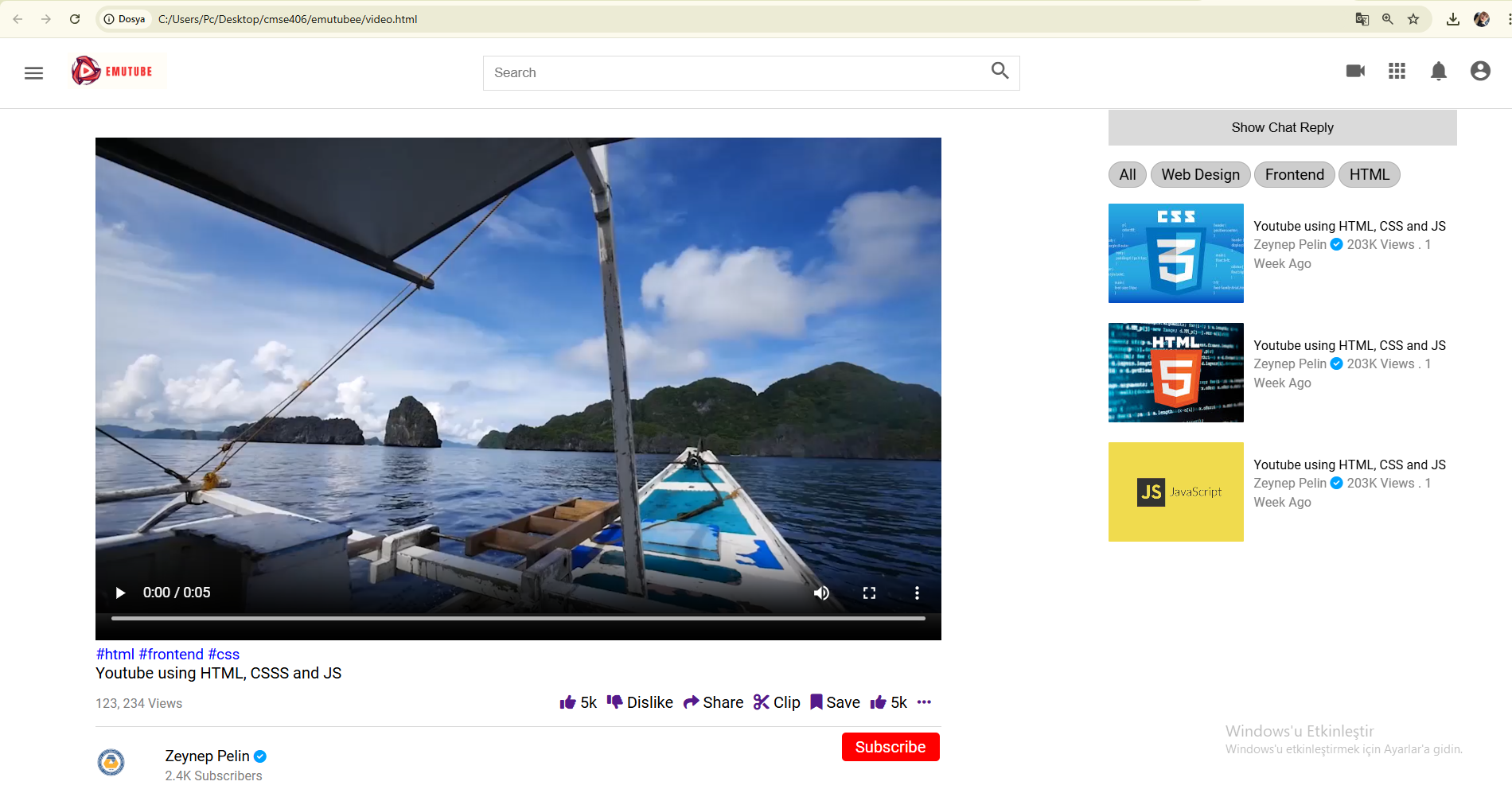
# 7.6 Upload Page

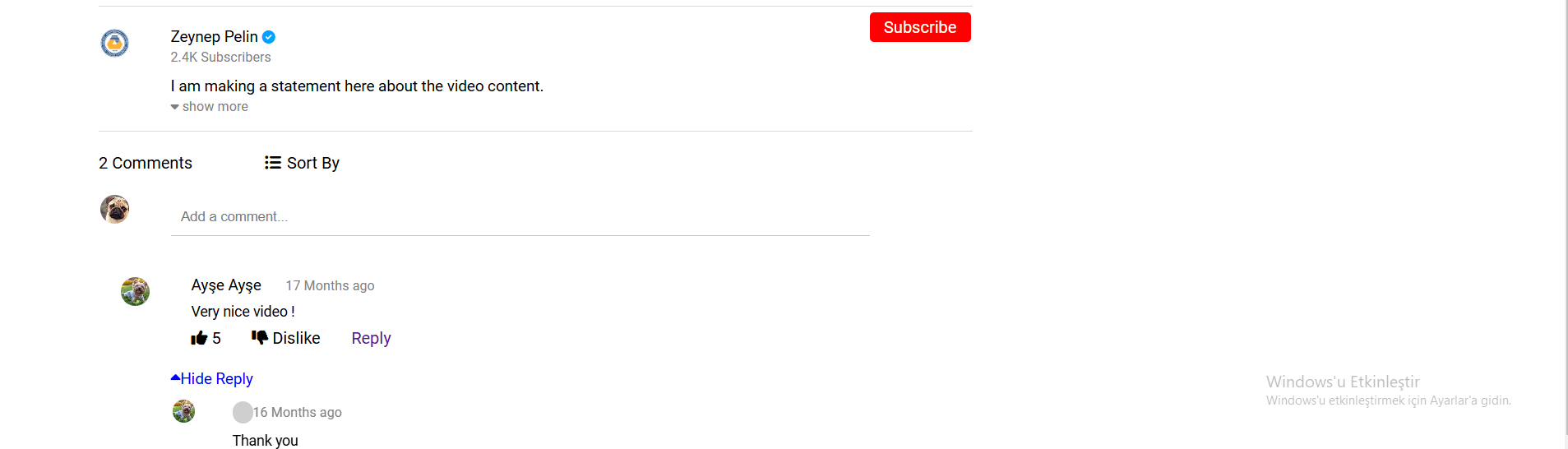
# 

**Upload page Figure 26**

Description: The page where users can upload new videos to their channel. It allows users to add video details such as title, description, tags, and select a thumbnail.Key Features: Video file upload, title field, description field, tags input, and thumbnail selection.

**7.7 Video Page**





**Video Page Figure 27**

Description: The video playback page where users can watch a selected video. It includes the video player, video details, and a comment section for user interaction.Key Features: Video player, video title, description, like/dislike buttons, comments section, and related videos.

# 8. DISCUSSION

The impact of our video-sharing platform in the global, economic, environmental, and societal context can be profound. This section discusses the potential benefits and effects of the platform across various domains.

Global Impact

Our video-sharing platform has the potential to connect people from different parts of the world, fostering global communication and cultural exchange. By providing a space for users to share their experiences, knowledge, and creativity, the platform can promote understanding and appreciation of diverse cultures and perspectives. This global connectivity can lead to a more informed and cohesive world community.

Economic Impact

Economically, the platform offers significant benefits. Content creators can monetize their videos through ads, sponsorships, and viewer donations, creating new income streams and job opportunities. This can be especially impactful in regions with limited traditional employment options, allowing individuals to earn a living through creative and educational content.

Additionally, businesses can use the platform to advertise products and services, reaching a broader audience and potentially increasing sales. The platform can also support small businesses and entrepreneurs by providing a cost-effective way to market their offerings and engage with customers.

Environmental Impact

From an environmental perspective, our platform can contribute positively by reducing the need for physical media and printed materials. As more educational and informational content moves online, the demand for paper-based resources and DVDs decreases, leading to fewer trees being cut down and reduced plastic waste.

Moreover, virtual events and online learning facilitated by the platform can reduce the need for travel, thus lowering carbon emissions associated with transportation. This can significantly contribute to efforts aimed at combating climate change and promoting sustainability.

Societal Impact

Societally, the platform can have several positive effects. It provides a space for free expression and the dissemination of information, empowering individuals to voice their opinions and share valuable insights. This can enhance democratic participation and social activism, enabling communities to address local and global issues more effectively.

The platform also has the potential to democratize education by making learning resources accessible to a broader audience, including those in remote or underserved areas. Educational videos on a wide range of subjects can help individuals acquire new skills and knowledge, leading to personal development and better job opportunities.

Furthermore, by offering content on health and wellness, the platform can improve public health awareness and provide guidance on various health-related issues. This can lead to healthier lifestyles and potentially reduce healthcare costs.

# 9. CONCLUSION

# Summary of the Project

# This project involved the development of a comprehensive video-sharing platform designed to facilitate the creation, sharing, and consumption of video content. The platform includes features such as user registration and login, video uploading, channel management, community interaction, and various user engagement tools. It serves as a hub where content creators and viewers can connect, share their interests, and build communities around shared topics.

# Purpose and Functionality

# The primary purpose of the platform is to provide a user-friendly and robust environment for video content sharing. It allows users to:

# Create and Manage Channels: Users can set up their channels to organize and promote their content.

# Upload and Share Videos: Users can upload videos, categorize them, and share them with a global audience.

# Engage with the Community: Through features like comments, likes, and community posts, users can interact with each other and build a following.

# Access Diverse Content: Viewers can explore a wide range of videos on various topics, enhancing their knowledge and entertainment options.

# Utility and Achievements

# This project is useful for several reasons:

# Empowering Content Creators: It provides a platform for individuals to share their creativity, knowledge, and skills, potentially turning their passions into a source of income.

# Fostering Community: By enabling interaction and engagement, it helps build communities around shared interests, promoting social connections.

# Educational Resource: It democratizes access to educational content, making learning resources available to a broader audience.

# Promotional Tool for Businesses: Businesses can use the platform for marketing and reaching new customers, contributing to economic growth.

# Through this project, we have achieved the creation of a functional, scalable, and user-friendly platform that addresses the needs of various stakeholders, from content creators to businesses and viewers.

# Personal Learning and Development

# Working on this project has been an invaluable learning experience. Personally, I have gained:

# Technical Skills: Enhanced my knowledge and proficiency in web development technologies, including front-end and back-end development.

# Project Management: Improved my ability to plan, execute, and manage a complex project from conception to deployment.

# Problem-Solving: Developed my problem-solving skills by addressing various technical challenges and finding effective solutions.

# Collaboration: Learned the importance of collaboration and communication within a development team.

# Overall, this project has not only resulted in a useful and impactful platform but also contributed significantly to my personal and professional growth. The skills and knowledge gained through this experience will undoubtedly benefit my future endeavors in the tech industry.10. REFERENCES

# [1]. Bass, L., Clements, P., & Kazman, R. (2013). Software Architecture in Practice. Addison-Wesley.

# [2]. Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley.

# [3]. Sommerville, I. (2011). Software Engineering. 9th Edition, Addison-Wesley.

# [4]. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

# [5]. YouTube. (n.d.). YouTube Help - Create and manage your channel. Retrieved from YouTube Help.

# [6]. Nielsen, J. (1994). Usability Engineering. Morgan Kaufmann.

# [7]. Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N., & Diakopoulos, N. (2016). Designing the User Interface: Strategies for Effective Human-Computer Interaction. 6th Edition, Pearson.

# [8]. Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach. 8th Edition, McGraw-Hill.

# [9]. IEEE. (2014). IEEE Standard for Software and Systems Test Documentation. IEEE Std 829-2008 (Revision of IEEE Std 829-1998).

# [10]. Krug, S. (2014). Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability. 3rd Edition, New Riders.

# [11]. Mitchell, W. J. (1999). City of Bits: Space, Place, and the Infobahn. MIT Press.

# [12]. Eriş, E. (2017). Yazılım Mühendisliği: Temeller ve Uygulamalar. Papatya Yayıncılık Eğitim.

# [13]. Demirel, T. (2015). Web Tasarımı ve Programlama. Kodlab Yayınları.

# [14]. Yıldız, T., & Çelik, M. (2018). Veritabanı Sistemleri: Temeller ve Tasarım. Nobel Akademik Yayıncılık.

[15]. Öztürk, B. (2019). Mobil Uygulama Geliştirme. Seçkin Yayıncılık.**APPENDICES**

## A. Instructions for installing the system

To install and run the HTML, CSS, and JavaScript-based project for both web and mobile use, copy the project folder from the provided CD to your desired directory. Open the index.html file in any web browser to access the application. Ensure you have a compatible web server if you need to serve the files remotely. For mobile use, access the same index.html file via a mobile browser or wrap the web application using a tool like..

## B. Code for the system

About.html

<div class="header">

<!-- Header menu Start -->

<div class="header\_\_left">

<i id="menu" class="material-icons">menu</i>

<a href="index.html">

<img src="https://www.xda-developers.com/files/2017/08/After-12-Years-Google-Gives-YouTube-a-New-Logo-1900x700\_c.png" alt="">

</a>

</div>

<!-- Header menu end -->

<!-- Header seach Start -->

<div class="header\_\_search">

<form action="">

<input type="text" placeholder="Search">

<button><i class="material-icons">search</i></button>

</form>

</div>

<!-- Header search Start -->

<!-- Header Icons Start -->

<div class="header\_\_icons">

<i class="material-icons display-this">search</i>

<i class="material-icons ">videocam</i>

<i class="material-icons">apps</i>

<i class="material-icons display-this">notifications</i>

<a href="channel.html"><i id="drop-down" class="material-icons display-this">account\_circle</i></a>

</div>

<!-- Header Icons Ends -->

</div>

<!-- Main Body Starts -->

<div class="mainBody">

<div class="sidebar">

<div class="sidebar\_\_categories">

<!-- Single Category -->

<a href="index.html">

<div class="sidebar\_\_category">

<i class="material-icons">home</i>

<span>Home</span>

</div>

</a>

<!-- Single Category -->

<div class="sidebar\_\_category">

<i class="material-icons">local\_fire\_department</i>

<span>Trending</span>

</div>

<!-- Single Category -->

<div class="sidebar\_\_category">

<i class="material-icons">subscriptions</i>

<span>Subscriptions</span>

</div>

</div>

<hr>

<div class="sidebar\_\_categories">

<!-- Single Category -->

<div class="sidebar\_\_category">

<i class="material-icons">library\_add\_check</i>

<span>Library</span>

</div>

<!-- Single Category -->

<div class="sidebar\_\_category">

<i class="material-icons">history</i>

<span>History</span>

</div>

<!-- Single Category -->

<a href="channel-videos.html">

<div class="sidebar\_\_category">

<i class="material-icons">play\_arrow</i>

<span>Your Videos</span>

</div>

</a>

<!-- Single Category -->

<div class="sidebar\_\_category">

<i class="material-icons">watch\_later</i>

<span>Watch Later</span>

</div>

<!-- Single Category -->

<div class="sidebar\_\_category">

<i class="material-icons">thumb\_up</i>

<span>Liked Videos</span>

</div>

</div>

<hr>

</div>

<!-- Sidebar Ends -->

<!-- Video Section Starts -->

<div class="videos">

<div class="channel-art">

<img src="images/channel-art.jpg" alt="">

</div>

<div class="channel-header flex">

<div class="channel-img-title flex">

<img class="img" src="images/logo.png" alt="">

<div class="channel-image-texts">

<h4>Zeynep Pelin</h4>

<span>2.46K subscribers</span>

</div>

</div>

<div class="channel-header-btns">

<div><a href="" class="channel-join">Join</a></div>

<div><a href="" class="channel-subscribe">Subscribe</a></div>

</div>

</div>

<div class="channel-header-links flex">

<a href="channel.html" class="display-this">Home</a>

<a href="channel-videos.html" class="display-this">Videos</a>

<a href="community.html" class="dont-display-thdis">Community</a>

<a href="index.html" class="dont-display-this">Membership</a>

<a href="index.html" class="dont-display-this">Channels</a>

<a href="about.html" class="display-this">About</a>

<a href="" class="dont-display-this"><i class="material-icons">search</i></a>

</div>

<br><br>

<hr>

<div class="videos\_\_container">

<!-- Single Video Starts Here -->

<div class="conteent-border">

<div class="about-div">

<div class="about-left">

<h3>Description</h3>

<h4>Here you will see information about me.Here you will see information about me.Here you will see information about me.Here you will see information about me.Here you will see information about me.Here you will see information about me.Here you will see information about me.</h4>

<hr>

<div class="about-details">

<div>

<h4>Details</h4>

</div>

<div class="about-biz">

<h3><small>For business enquiry:</small></h3>

<h3 class="left-h4"><small>zeyneppelin@gmail.com</small></h3>

</div>

<div class="about-biz">

<h3><small>For business enquiry</small></h3>

<h3 class="left-h4"><small>zeynep@gmail.com</small></h3>

</div>

</div>

<hr>

<div class="links">

<h4>Links</h4>

<div class="links-content">

<a href="">Website</a> <br><br>

<a href="">Website</a> <br><br>

<a href="">Website</a> <br><br>

<a href="">Website</a>

</div>

</div>

</div>

<div class="about-right">

<h4><b>Stats</b></h4>

<hr>

<div class="stats">

<h5>Joined Jan 27, 2024</h5>

<hr>

<h5>127,000 Views</h5>

<hr>

<h5>

<i class="material-symbols-outlined">flag</i>

</h5>

</div>.

İndex.html

<div class="header">

<!-- Header menu Start -->

<div class="header\_\_left">

<i id="menu" class="material-icons">menu</i>

<a href="index.html">

<img src="https://www.xda-developers.com/files/2017/08/After-12-Years-Google-Gives-YouTube-a-New-Logo-1900x700\_c.png" alt="">

</a>

</div>

<!-- Header menu end -->

<!-- Header seach Start -->

<div class="header\_\_search">

<form action="">

<input type="text" placeholder="Search">

<button><i class="material-icons">search</i></button>

</form>

</div>

<!-- Header search Start -->

<!-- Header Icons Start -->

<div class="header\_\_icons">

<i class="material-icons display-this">search</i>

<i class="material-icons ">videocam</i>

<i class="material-icons">apps</i>

<i class="material-icons display-this">notifications</i>

<a href="channel.html"><i id="drop-down" class="material-icons display-this">account\_circle</i></a>

</div>

<!-- Header Icons Ends -->

</div>

<!-- Main Body Starts -->

<div class="mainBody">

<div class="sidebar">

<div class="sidebar\_\_categories">

<!-- Single Category -->

<a href="index.html">

<div class="sidebar\_\_category">

<i class="material-icons">home</i>

<span>Home</span>

</div>

</a>

<!-- Single Category -->

<div class="sidebar\_\_category">

<i class="material-icons">local\_fire\_department</i>

<span>Trending</span>

</div>

<!-- Single Category -->

<div class="sidebar\_\_category">

<i class="material-icons">subscriptions</i>

<span>Subscriptions</span>

</div>

</div>

<hr>

<div class="sidebar\_\_categories">

<!-- Single Category -->

<div class="sidebar\_\_category">

<i class="material-icons">library\_add\_check</i>

<span>Library</span>

</div>

<!-- Single Category -->

<div class="sidebar\_\_category">

<i class="material-icons">history</i>

<span>History</span>

</div>

<!-- Single Category -->

<a href="channel-videos.html">

<div class="sidebar\_\_category">

<i class="material-icons">play\_arrow</i>

<span>Your Videos</span>

</div>

</a>

<!-- Single Category -->

<div class="sidebar\_\_category">

<i class="material-icons">watch\_later</i>

<span>Watch Later</span>

</div>

<!-- Single Category -->

<div class="sidebar\_\_category">

<i class="material-icons">thumb\_up</i>

<span>Liked Videos</span>

</div>

</div>

<hr>

</div>

<!-- Sidebar Ends -->

<!-- Video Section Starts -->

<div class="videos">

<div class="drop-down-container">

<h1>Recommended</h1>

<div class="dropdown-content-background">

<div class="dropdown-contents">

<div class="dropdown-content">

<i class="material-icons">videocam</i>

<span>Create</span>

</div>

<div class="dropdown-content">

<i class="material-icons">settings</i>

<span>Settings</span>

</div>

<div class="dropdown-content">

<i class="material-icons">person</i>

<span>Switch Account</span>

</div>

</div>

</div>

</div>

<div class="videos\_\_container">

<!-- Single Video Starts Here -->

<div class="video">

<div class="video\_\_thumbnail">

<img src="images/arka.jpeg" alt="">

</div>

<div class="video\_\_details">

<div class="author">

<img src="images/logo.png" alt="">

</div>

<div class="title">

<h3>Youtube Using HTML, CSS and JS</h3>

<a href=""><small>Zeynep Pelin <i class="fas fa-check-circle"></i></small></a>

<span><small>1M Views . 1 Months Ago</small></span>

</div>

</div>

</div>

<!-- Single Video Starts Here -->

<div class="video">

<div class="video\_\_thumbnail">

<img src="images/arka1.jpeg" alt="">

</div>

<div class="video\_\_details">

<div class="author">

<img src="images/logo.png" alt="">

</div>

<div class="title">

<h3>Youtube Using HTML, CSS and JS</h3>

<a href=""><small>Zeynep Pelin <i class="fas fa-check-circle"></i></small></a>

<span><small>1M Views . 1 Months Ago</small></span>

</div>

</div>

</div>

<!-- Single Video Starts Here -->

<div class="video">

<div class="video\_\_thumbnail">

<img src="images/arka2.jpeg" alt="">

</div>

<div class="video\_\_details">

<div class="author">

<img src="images/logo.png" alt="">

</div>

<div class="title">

<h3>Youtube Using HTML, CSS and JS</h3>

<a href=""><small>Zeynep Pelin <i class="fas fa-check-circle"></i></small></a>

<span><small>1M Views . 1 Months Ago</small></span>

</div>

</div>

</div>

<!-- Single Video Starts Here -->

<div class="video">

<div class="video\_\_thumbnail">

<img src="images/arka3.jpeg" alt="">

</div>

<div class="video\_\_details">

<div class="author">

<img src="images/logo.png" alt="">

</div>

<div class="title">

<h3>Youtube Using HTML, CSS and JS</h3>

<a href=""><small>Zeynep Pelin <i class="fas fa-check-circle"></i></small></a>

<span><small>1M Views . 1 Months Ago</small></span>

</div>

</div>

</div>

<!-- Single Video Starts Here -->

<div class="video">

<div class="video\_\_thumbnail">

<img src="images/arka4.jpeg" alt="">

</div>

<div class="video\_\_details">

<div class="author">

<img src="images/logo.png" alt="">

</div>

<div class="title">

<h3>Youtube Using HTML, CSS and JS</h3>

<a href=""><small>Zeynep Pelin <i class="fas fa-check-circle"></i></small></a>

<span><small>1M Views . 1 Months Ago</small></span>

</div>

</div>

</div>

<!-- Single Video Starts Here -->

<div class="video">

<div class="video\_\_thumbnail">

<img src="images/arka5.jpeg" alt="">

</div>

<div class="video\_\_details">

<div class="author">

<img src="images/logo.png" alt="">

</div>

<div class="title">

<h3>Youtube Using HTML, CSS and JS</h3>

<a href=""><small>Zeynep Pelin <i class="fas fa-check-circle"></i></small></a>

<span><small>1M Views . 1 Months Ago</small></span>

Upload.html

<div class="header">

<!-- Header menu Start -->

<div class="header\_\_left">

<i id="menu" class="material-icons">menu</i>

<a href="index.html">

<img src="https://www.xda-developers.com/files/2017/08/After-12-Years-Google-Gives-YouTube-a-New-Logo-1900x700\_c.png" alt="">

</a>

</div>

<!-- Header menu end -->

<!-- Header seach Start -->

<div class="header\_\_search">

<form action="">

<input type="text" placeholder="Search">

<button><i class="material-icons">search</i></button>

</form>

</div>

<!-- Header search Start -->

<!-- Header Icons Start -->

<div class="header\_\_icons">

<i class="material-icons display-this">search</i>

<i class="material-icons ">videocam</i>

<i class="material-icons">apps</i>

<i class="material-icons display-this">notifications</i>

<a href="channel.html"><i id="drop-down" class="material-icons display-this">account\_circle</i></a>

</div>

<!-- Header Icons Ends -->

</div>

<!-- Main Body Starts -->

<div class="mainBody">

<div class="sidebar">

<div class="sidebar\_\_categories">

<!-- Single Category -->

<a href="index.html">

<div class="sidebar\_\_category">

<i class="material-icons">home</i>

<span>Home</span>

</div>

</a>

<!-- Single Category -->

<div class="sidebar\_\_category">

<i class="material-icons">local\_fire\_department</i>

<span>Trending</span>

</div>

<!-- Single Category -->

<div class="sidebar\_\_category">

<i class="material-icons">subscriptions</i>

<span>Subscriptions</span>

</div>

</div>

<hr>

<div class="sidebar\_\_categories">

<!-- Single Category -->

<div class="sidebar\_\_category">

<i class="material-icons">library\_add\_check</i>

<span>Library</span>

</div>

<!-- Single Category -->

<div class="sidebar\_\_category">

<i class="material-icons">history</i>

<span>History</span>

</div>

<!-- Single Category -->

<a href="channel-videos.html">

<div class="sidebar\_\_category">

<i class="material-icons">play\_arrow</i>

<span>Your Videos</span>

</div>

</a>

<!-- Single Category -->

<div class="sidebar\_\_category">

<i class="material-icons">watch\_later</i>

<span>Watch Later</span>

</div>

<!-- Single Category -->

<div class="sidebar\_\_category">

<i class="material-icons">thumb\_up</i>

<span>Liked Videos</span>

</div>

</div>

<hr>

</div>

<!-- Sidebar Ends -->

<!-- Video Section Starts -->

<div class="videos">

<div class="upload-content">

<div class="left-video">

<h4>Details</h4>

<div class="input-div">

<span>Title (required) <span id="count"></span> </span><br>

<input id="title" type="text" onkeyup="titleValidate()">

<div class="error-div" id="error-div">

</div>

</div>

<script>

</script>

<div class="desc-div">

<span>Description (required) <span id="desc-count"></span> </span> <br>

<textarea name="" id="description" onkeyup="desc\_validate()"></textarea>

</div>

<div class="tags-div">

<span>Tags (optional) </span> <br>

<input type="text" >

<p>Seperate Tags with Comma</p>

</div>

<div class="video-div file-input" id="file-input">

<small>Thumbnail (required) </small> <br>

<input type="file" class="file">

<!-- <input type='file'> -->

<span class='button'><i class="material-icons">image</i></span>

<span class='label' data-js-label>No image selected</label>

</div>

<div class="video-div file-input">

<small>Video (required) </small> <br>

<input type="file" class="file">

<!-- <input type='file'> -->

<span class='button'><i class="material-icons">videocam</i></span>

<span class='label' data-js-label>No video selected</label>

</div>

<div class="upload-btn-div">

<button class="upload-btn">Publish</button>

</div>

</div>

<div class="right-video">

<div class="r-thumb">

<div class="uploading-video">

<h5>Uploading Video...</h5>

</div>

<!-- <img src="images/logo.jpg" alt=""> -->

<div class="video-details">

<h5>Video Links</h5>

<a href="">https://www.youtube.com/</a>

<h4>Filename</h4>

<span >Video File name here</span>

</div>

</div>

</div>

</div>

<div class="videos\_\_container">