**Web Streaming Website**

**Graduation Project Report**

**CMSE405**

**Team members:**

**Abdulaziz MAHMOUD 18700859**

**Doğukan RAMAZAN 20450049**

**Zeynep Pelin ÇOLAK 17300009**

**Supervisor:**

**Asst. Prof. Dr. Ahmet ÜNVEREN**

**Computer Engineering Department**

**Eastern Mediterranean University**

**2023-2024 Spring**

# 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 1](#_Toc478723914)

[2. PROJECT PLANNING AND MANAGEMENT](#_Toc478723915)

[3. REQUIREMENTS ANALYSIS](#_Toc478723916)

[3.1 Functional Requirements](#_Toc478723917)

[3.2 Non-Functional Requirements](#_Toc478723918)

[3.3 Realistic constraints](#_Toc478723919)

[3.4 Ethical issues](#_Toc478723920)

[4. DESIGN](#_Toc478723921)

[4.1 High level design (architectural)](#_Toc478723922)

[4.2 Software design](#_Toc478723923)

[5. IMPLEMENTATION](#_Toc478723924)

[5.1 Tools, technologies and platforms used](#_Toc478723925)

[5.2 Algorithms](#_Toc478723926)

[5.3 Standards](#_Toc478723927)

[5.4 Detailed description of the implementation (coding)](#_Toc478723928)

[6. TESTING](#_Toc478723929)

6.1 Testing Strategy…………………………………………………………………………..…

6.2 Quality Assurance Measures………………………………………………………………..

[7. USER GUIDE OF THE SYSTEM](#_Toc478723930)

7.1 . Index Page………………………………………………………………….………………

7.2 About Page………………………………….……………………………….………………

7.3 Channel Page……………………………………………………………….………………

7.4 Channel Videos Page……………………………………………………….………………

7.5 Community Page……………………………………………………..…….………………

7.6 Community Page…………………………………….…………….……….………………

7.7 Community Detail Page…………………………………………………….………………

7.8 Login Page………………………………………………………………….………………

7.9 Upload Page ………………………………………………….…………….………………

7.10 Video Page……………………………………………..………………….………………

[8. DISCUSSION](#_Toc478723931)

[9. CONCLUSION](#_Toc478723932)

[10. REFERENCES](#_Toc478723933)

[APPENDICES](#_Toc478723934)

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

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

B.1. About.html…………………………………………………………………...………….

B.2. İndex.html………………………………………………………………………............

B.3. Upload.html……………………………………………………………………………...

# LIST OF FIGURES

[Figure 1 Gantt chart part 1](file:///C:\Users\Admin\Desktop\SOFTWARE_PROJECT_CMPE406_SPRING_2021_2022\SOFTWARE_FINAL_REPORT_CMPE406_ERS_SPRING-2021-2022.docx#_Toc105976985)

[Figure 2 Gantt chart part 2](file:///C:\Users\Admin\Desktop\SOFTWARE_PROJECT_CMPE406_SPRING_2021_2022\SOFTWARE_FINAL_REPORT_CMPE406_ERS_SPRING-2021-2022.docx#_Toc105976986)

[Figure 3 Organization scheme](file:///C:\Users\Admin\Desktop\SOFTWARE_PROJECT_CMPE406_SPRING_2021_2022\SOFTWARE_FINAL_REPORT_CMPE406_ERS_SPRING-2021-2022.docx#_Toc105976988)

## Figure 4 System Architecture…………………………………………………………

Figure 5 index page…………………………………………………………………….

Figure 6 About page……………………………………………………………………

Figure 7 Channel Page…………………………………………………………………

Figure 8 Channel videos page…………………………………………………………

Figure 9 Community Page……………………………………………………………..

Figure 10 Community create page…………………………………………………….

Figure 11 Community detail page……………………………………………………..

Figure 12 Login page…………………………………………………………………...

Figure 13 Upload page…………………………………………………………………

Figure 14 Video page…………………………………………………………………..

# LIST OF TABLES

[Table 1 Work package 1](#_Toc105977057)

[Table 2 Work package 1 activities](#_Toc105977058)

[Table 3 Work package 2](#_Toc105977059)

[Table 4 Work package 2 activities](#_Toc105977060)

[Table 5 Work package 3](#_Toc105977061)

[Table 6 Work package 3 activities](#_Toc105977062)

[Table 7 Work package 4](#_Toc105977063)

[Table 8 Work package 4 activities](#_Toc105977064)

[Table 9 List of milestones](#_Toc105977065)

[Table 10 List of risks](#_Toc105977066)

[Table 11 Project Team](#_Toc105977067)

[Table 12 Instruments/ Equipment/ Software/ Release Purchases](#_Toc105977068)

[Table 13 (M030) Quarterly Estimated Cost Form (TL-USD)](#_Toc105977069)

[Table 14 Activity Table](#_Toc105977070)

[Table 15 Path calculations Table](#_Toc105977071)

Table 16 Network Diagram……………………………………………………………..

Table 17 Calculating probability………………………………………………………..

Table 18 Crashing Approach …………………………………………………………..

Table 19 Z-table………………………………………………………………………...

Table 20 Algorithm Description for Main Functions…………………………………

**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

In Tables 1-8, work packages and their criteria as feasibility, methods, experiments, success, and outputs where displayed in details.

**Table 1 Work package 1**

|  |  |
| --- | --- |
| **Work Package No** | 1 |
| **Work Package Name** | **Project Feasibility and Pre-Research** |
| **Start-End Date and Time** | 1/4/2024 – 14/4/2024 |
| **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 criterias.** |
| **Outputs:**   * Having a preliminary idea of the scope of the system * Estimation on profits and costs * Establishing the distribution of tasks   **Success Criterias:**   * 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 |

**Table 2 Work packages 1 activities**

**Table 3 Work package 2**

|  |  |
| --- | --- |
| **Work Package No** | 2 |
| **Work Package Name** | **System Design** |
| **Start-End Date and Time** | 15/4/24 – 27/5/24 |
| **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 criterias.** |
| **Outputs:**   * Compatible requirements report * Diagrams that describe the system structure   **Success Criterias:**  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 |

Table 4 Work package 2 activities

Table 5 Work package 3

|  |  |
| --- | --- |
| **Work Package No** | 3 |
| **Work Package Name** | **Development of System Software (Development Stage)** |
| **Start-End Date and Time** | 1/6/24 – 1/11/24 |
| **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 criterias.** |
| **Outputs:**  Functional System, ready to be tested  **Success Criterias:**   * 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 |

Table 6 Work package 3 activities

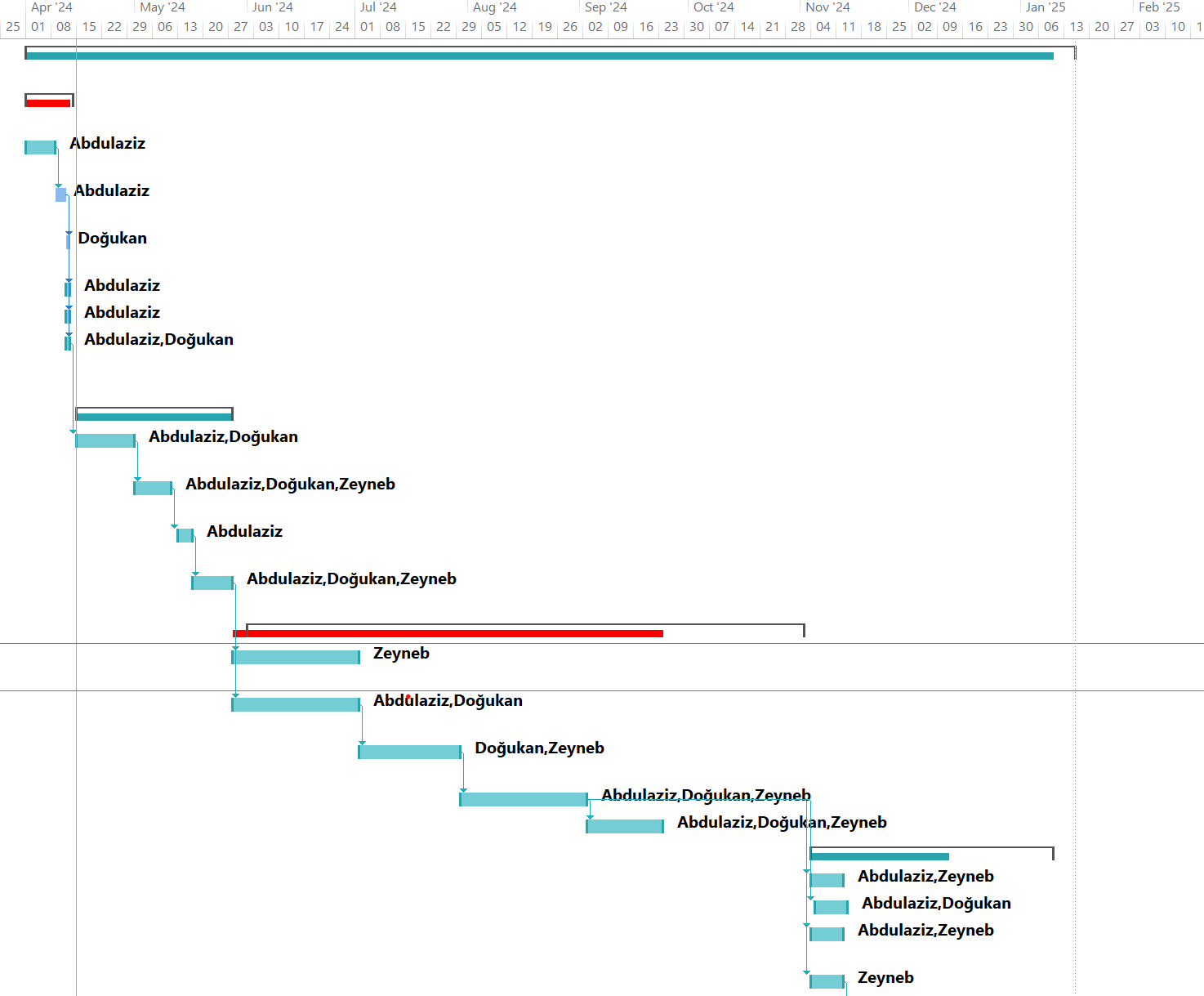
Table 7 Work package 4

|  |  |
| --- | --- |
| **Work Package No** | 4 |
| **Work Package Name** | **Prototype Implementation and Test Study and Maintenance (Test & Maintenance stage)** |
| **Start-End Date and Time** | 4/11/24 – 9/1/25 |
| **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 criterias.** |
| **Outputs:**  Reports of Tests  **Success Criterias:**  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 |

Table 8 Work package 4 activities

# 2.2 Gantt Chart



**Figure 1 – 2 Gantt chart**

# 2.3. List of Milestones

In Table 9, list of milestones with their description and expected time interval where displayed.

Table 9 List of milestones

|  |  |  |
| --- | --- | --- |
|  | Description of Output | Expected Time Interval |
| 1 | **Project Feasibility and Pre-Research** | 1/4/2024 – 14/4/2024 |
| 2 | **System Design** | 15/4/24 – 27/5/24 |
| 3 | **Development of System Software** | 1/6/24 – 1/11/24 |
| 4 | System Testing | 4/11/24 – 9/1/25 |

# 2.4. List of Risks

In Table 10, list of risks with their probability of happening, effects, and the strategy that will be used to solve them where displayed.

Table 10 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. |

# 

# 2.5. Project Team

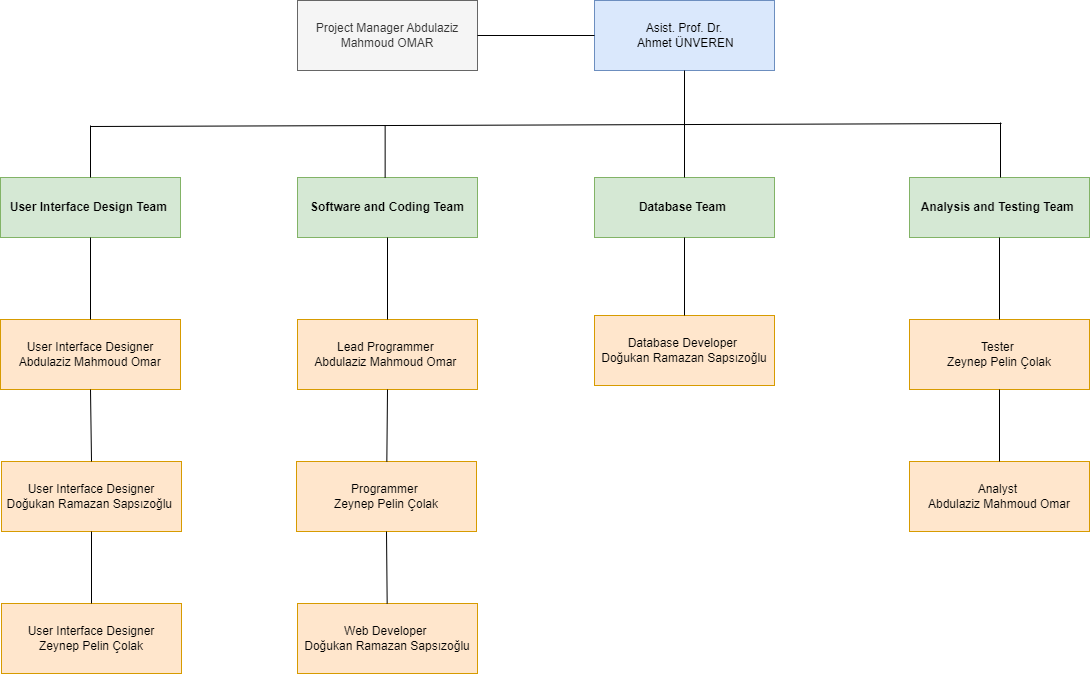
In Table 11, project team including their names, title, Ids, education status, graduation date, and other information where displayed.

Table 11 Project Team

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Personnel Name | Title | ID | Education Status | Graduation Date | Date of Starting Work | Idea Owner |
| Abdulaziz Mahmoud OMAR | Project Manager / Lead Programmer | 18700859 | Undergraduate | 2025 | 26.02.2024 | Yes |
| Doğukan Ramazan SAPSIZOĞLU | Database Developer | 20450049 | Undergraduate | 2025 | 26.02.2024 | Yes |
| Zeynep Pelin ÇOLAK | User Interface Designer | 17300009 | Undergraduate | 2025 | 26.02.2024 | Yes |

# 2.6. Organization Scheme

In Figure 3, organization scheme by the aid of the Table 11 the tasks of the team members where displayed and the connections among them and with the supervisor Assoc. Prof. Ahmet Unveren where displayed.



**2.7. (M013) Instrument / Equipment / Software / RELEASE PURCHASES**

In Table 12, instruments, equipment, software, and release purchases where displayed including number of items used, capacity, technical specification, purpose of project activities, and unit pricing in Turkish lira and USD.

Table 12 Instruments/ Equipment/ Software/ Release Purchases

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Project Name** | | **Design Youtube** | | | | | | | | | |
| **Line no** | **Instrument / Equipment / Software / Publication Name** | | **No. of Item** | **Capacity** | **Technical specification** | **Purpose of Project Activities** | **Post-Project Place of Use / Purpose** | | **Unit Price (USD)** | **Unit Price (TL)** | **Total Amount (TL)** |
| **R & D** | **Production** |
| **1** | **Computers** | | **3** | **3** | **8-core CPU, 10-core GPU, 8GB of RAM, and 512GB SSD, M3 chip, intel i5 or faster** | **Using every aspect of tasks to create project.** |  | **YES** | **800** | **26000** | **78000** |
| **2** | **Internet connection** | | **1** | **3** | **Minimum 15Mbps of connection speed** | **Making a connection between projects** |  | **YES** | **80** | **2500** | **2500** |
| **3** | **MS OFFICE** | | **1** | **3** | **Office related tools** | **Using tools for scheduling tasks to document our applications** |  | **YES** | **100** | **3240** | **3240** |
| **4** | **Oracle MySQL / Atom** | | **1** | **1** | **Database management and modify tools** | **To store data in tables that map to objects for database management** |  | **YES** | **300** | **9600** | **9600** |
| **5** | **JavaScript(React)** | | **1** | **1** | **Language for developing Website platform.** | **Coding objectives and purposes** |  | **YES** | **60** | **1980** | **1980** |
| **6** | **HTML/CSS** | | **1** | **1** | **Language for developing and evaluating website design and contents.** | **Coding objectives** |  | **YES** | **40** | **1280** | **1280** |
| **7** | **Video.js or Plyr(Github)** | | **1** | **1** | **Libraries for developing the website platform.** | **Coding objectives** |  | **YES** | **20** | **640** | **640** |
| **8** | **Google Analytics** | | **1** | **1** | **For website analytics** | **Analytics purposes** |  | **YES** | **0** | **0** | **0** |
| **9** | **VSCODE** | | **1** | **1** | **Software coding and compiling platform** | **Coding objectives** |  | **YES** | **20** | **640** | **640** |
|  |  | |  |  |  |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |  | **TOTAL** | **97880 TL** |

**2.8. (M030) Quarterly Estimated Cost Form (TL)**

In Table 13, the quarterly estimated cost in TL/USD where displayed in details including personal pricing, traveling, etc.

Table 13 (M030) Quarterly Estimated Cost Form (TL-USD)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Project Name :** | | | | |
| **Cost Item** | **WHICH YEAR?** | | **TOTAL**  **(TL)** | **TOTAL COST RATE OF CONTENTS (%)** |
| **I** | **II** |
| **Personnel** | 25000 | 25000 |  | 25 |
| **Travel** | 2500 | 2500 |  | 5 |
| **Instrument / Equipment / Software / Publications** | 6000 | 4000 |  | 10 |
| **Domestic Works Made By R & D and Testing Institutions** | 10000 | 10000 |  | 20 |
| **International Works Made By R & D and Testing Institutions** | - | - | - | - |
| **Domestic Services Procurement** | 5000 | 5000 |  | 10 |
| **Overseas Service Procurement** | 2500 | 2500 |  | 5 |
| **Material** | 25000 | 25000 |  | 25 |
| **CUMULATIVE COST** |  |  |  | 100 |
| **IN THE PROJECT TOTAL MAN-MONTH** | | | 150000 | |

## 2.9.Activity/ Path/ Variance/ Probability

In Tables 14, activities of building the system where displayed in alphabetical order with their description, duration, prerequisites, and other dimensions.

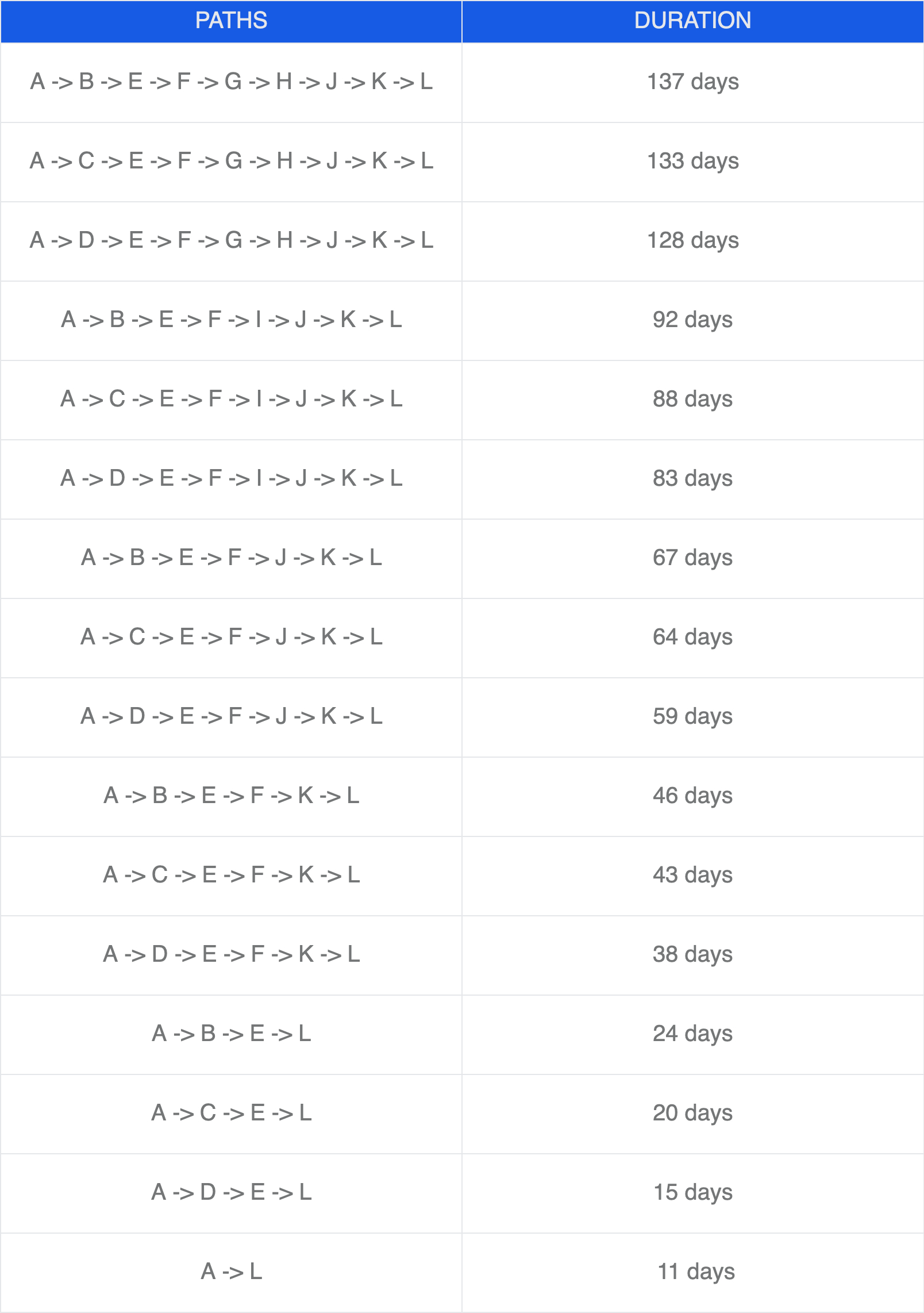
**Table 14 Activity Table**

A list of tasks with black text

Description automatically generated

In Table 15, the paths according to the activities where displayed with their calculation, and total expected time for each path.

Table 15 Path calculations Table



**Table 16 Network Diagram**

A diagram of a network diagram

Description automatically generated

Calculating probability of successful completion rate for each paths

**Table 17 Calculating probability**

A table with numbers and letters

Description automatically generated

**Crashing Approach**

Z-value = (Maximum Time - Expected Time) / Standard Deviation

We are going to calculate the Z value for each path in the section on critical path management.

**Table 18 Crashing Approach**

A table with numbers and a number of objects

Description automatically generated with medium confidence

**Probability determined by using the Z-table.**

**Table 19 Z-table**

A table with numbers and a few lines

Description automatically generated with medium confidence

Normal total cost = 58450 TL

Reduction for chosen activities : A,B,C,D,E,H,I,J,K

New Cost = Total Crash Cost = 60300 TL

We can recover from a one-week crash at an extra cost of about 235.71 TL every day.

# 3. REQUIREMENTS ANALYSIS

## Functional Requirements

**3.1.1.Video Upload**

**1.1 User Authentication:**

* Users must be authenticated to upload videos.
* Support for OAuth or other authentication mechanisms.

**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.

**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.

**1.4 Pre-signed URL Generation:**

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

**3.1.2. Video Processing**

**2.1 Video Storage:**

* Store original videos in a blob storage system.
* Transcode videos into multiple resolutions and formats.

**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.

**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 Content Delivery Network (CDN):**

* Serve videos from a CDN for low latency and high availability.
* Cache popular videos in CDN to reduce costs.

**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.

**3.3 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**

**4.1 Comments and Reactions:**

* Allow users to comment on videos.
* Enable users to like, dislike, and share videos.

**4.2 Subscriptions and Notifications:**

* Users can subscribe to channels and receive notifications for new uploads.
* Manage subscription lists and notification settings.

**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**

**5.1 Metadata Storage:**

* Store metadata in a sharded and replicated database for high performance and availability.
* Cache frequently accessed metadata for faster retrieval.

**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**

**6.1 High Availability:**

* Ensure system components are redundant and can handle failures gracefully.
* Use load balancers to distribute traffic across multiple servers.

**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.

**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.

**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.

**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**

**7.1 Live Streaming:**

* Support for live streaming with low latency and smooth playback.
* Handle live video encoding and streaming in real-time.

**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.

## 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.

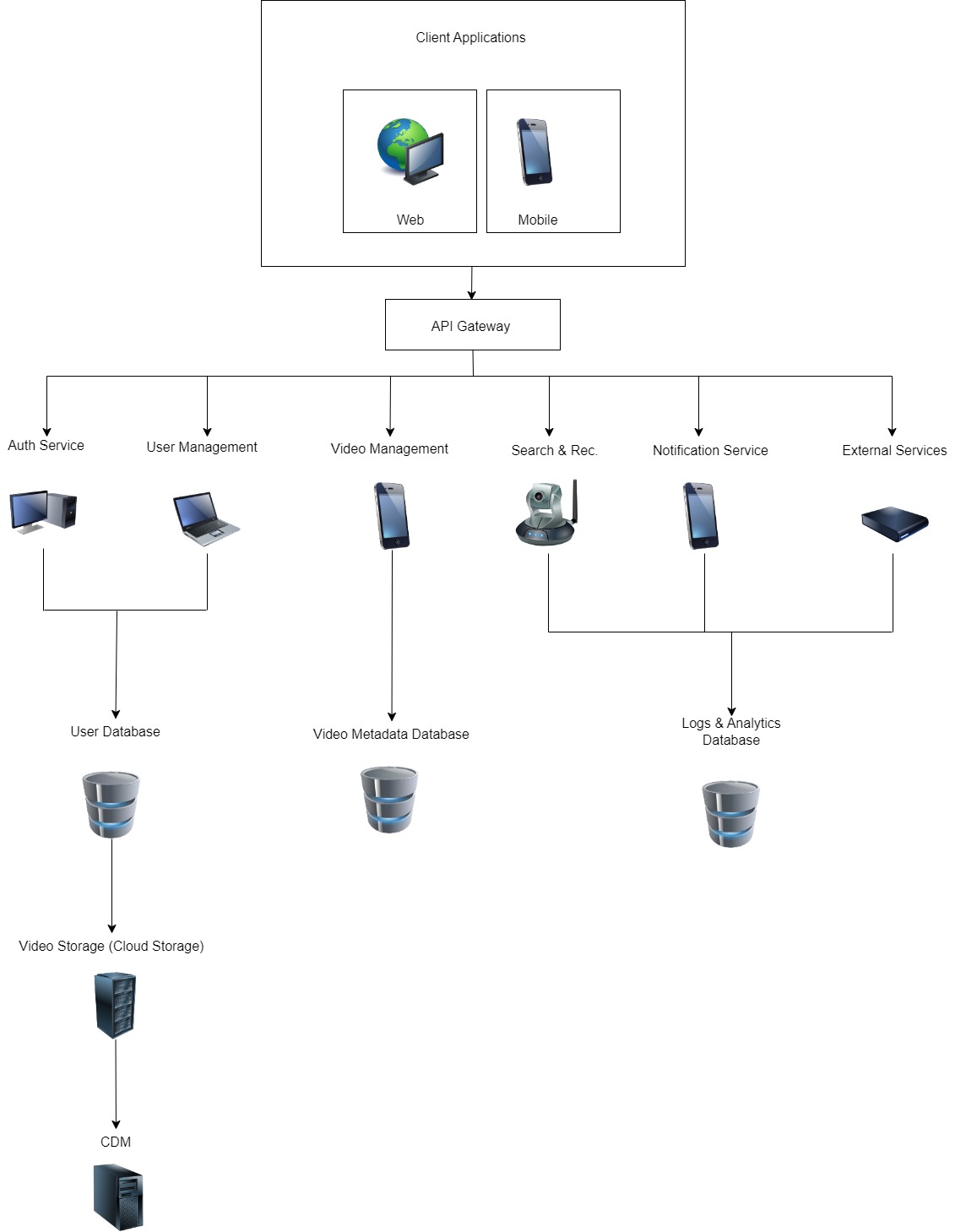
# Ethical Development:

# 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)

## 



## Figure 4 System Architecture

## 4.2 Software design

**a. General System Architecture Diagram**

System Architecture

The architecture comprises the following main components:

1. Client-Side

* Web Application
* Mobile Application

1. Server-Side

* API Server
* Video Processing Server
* Authentication Server

1. Database

* User Database
* Video Metadata Database

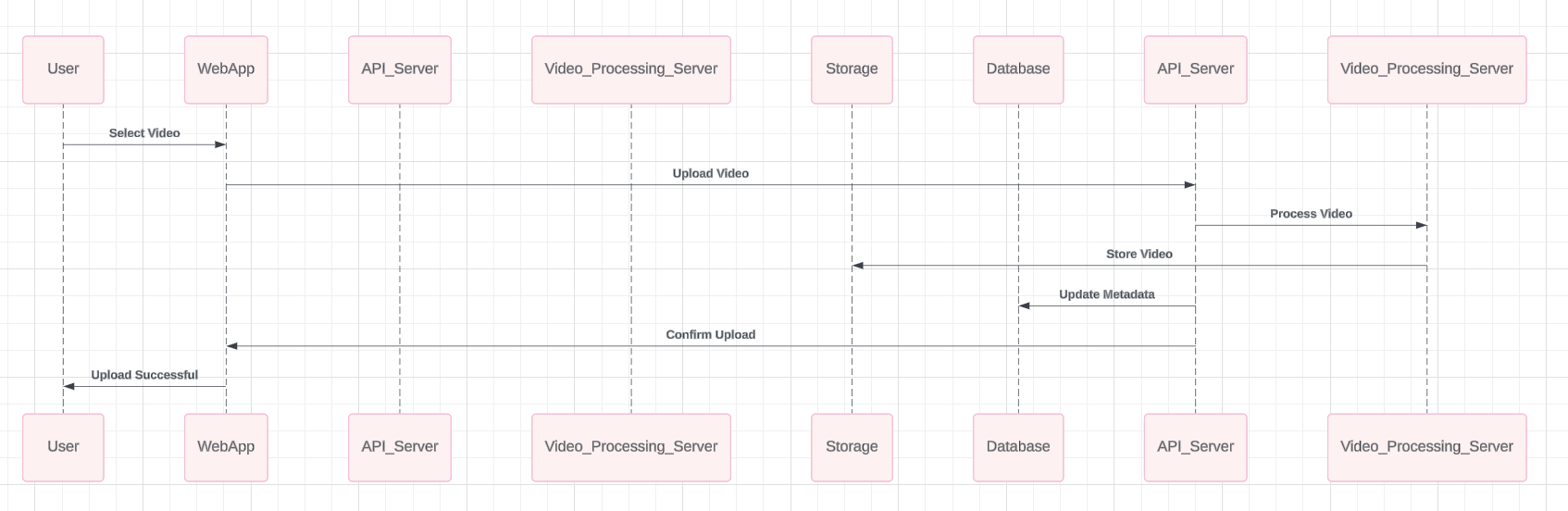
1. External Services

* CDN (Content Delivery Network)

\* Cloud Storage

## b. UML Modelling

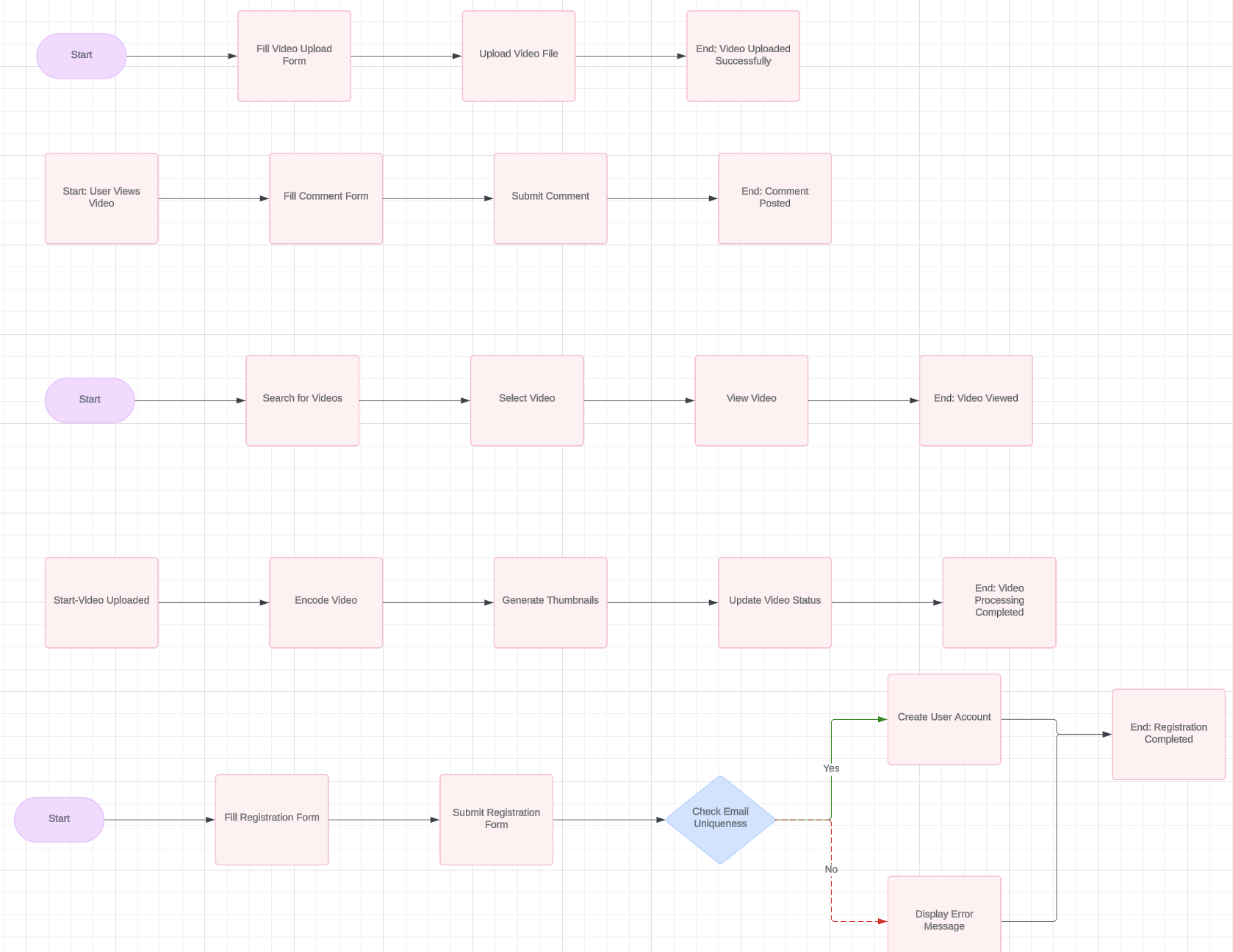
### **i. Sequence Diagrams**

**Sequence Diagram: Video Upload**

### **ii. Activity Diagram**

### iii. **BPMN Diagram**

**BPMN Diagram: Video Processing Workflow**

****

**Description of BPMN Diagram**

**User Registration Process**

1. **Start Event**: User clicks on "Register".
2. **Task**: User fills in registration details (username, email, password).
3. **Task**: User submits registration form.
4. **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.

1. **End Event**: Registration process ends.

**Video Upload Process**

1. **Start Event**: User clicks on "Upload Video".
2. **Task**: User fills in video details (title, description, category).
3. **Task**: User uploads video file.
4. **End Event**: Video upload process ends.

**Video Processing Process**

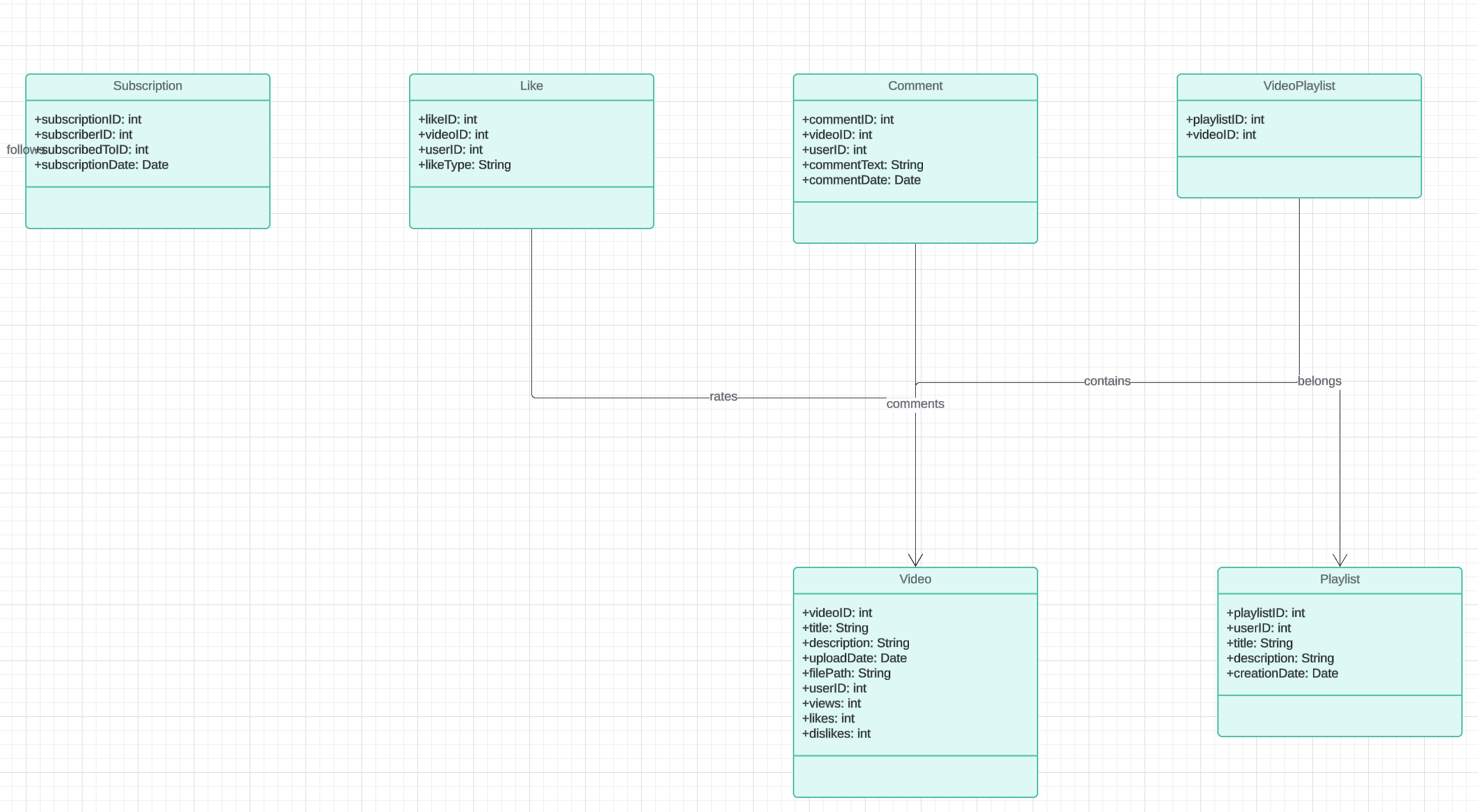
1. **Start Event**: Video upload completes.
2. **Task**: System encodes video.
3. **Task**: System generates video thumbnails.
4. **Task**: System updates video status to "Processed".
5. **End Event**: Video processing completes.

**Commenting on Videos Process**

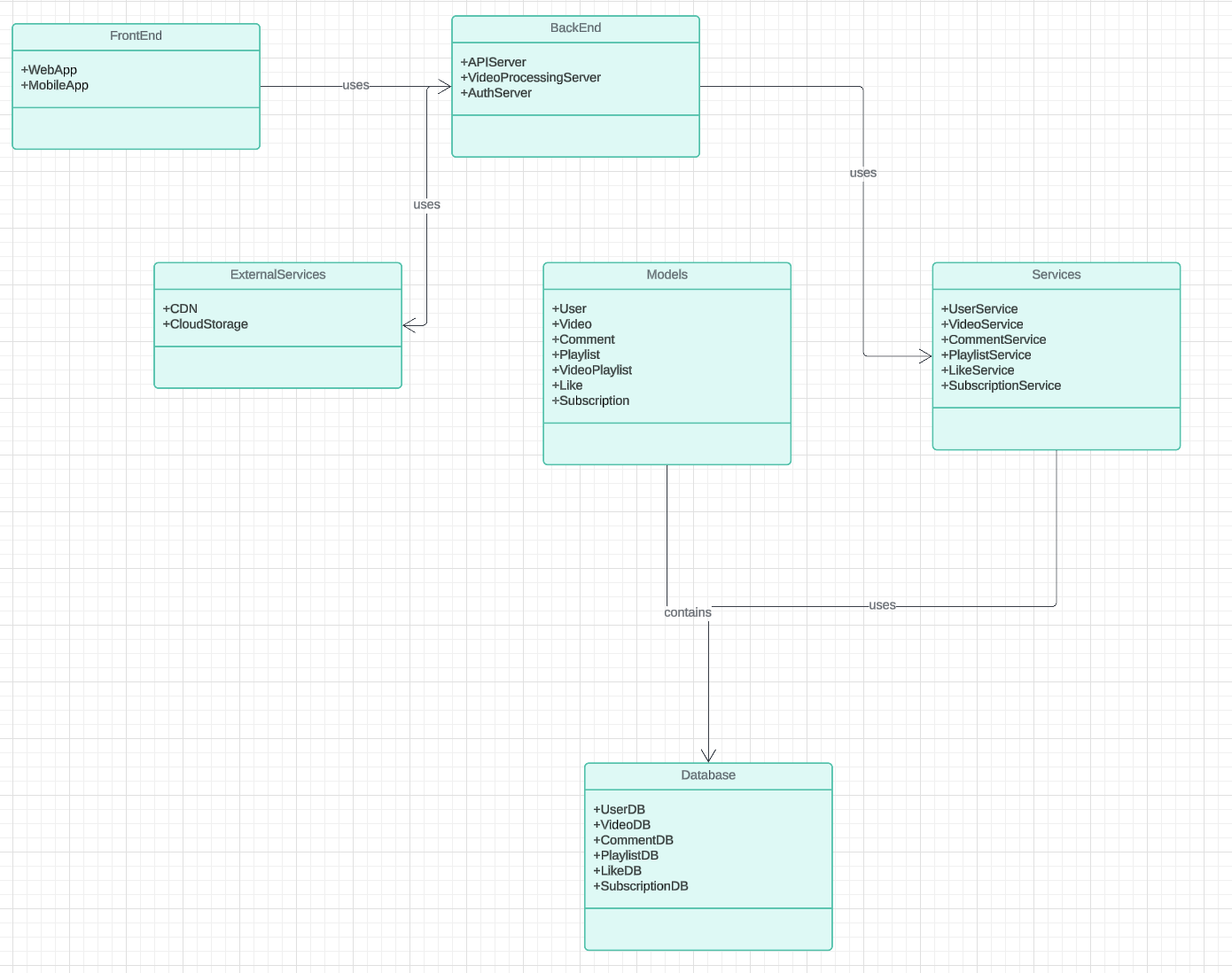
1. **Start Event**: User views a video.
2. **Task**: User fills in comment text.
3. **Task**: User submits comment.
4. **End Event**: Comment posting ends.

**Video Viewing Process**

1. **Start Event**: User initiates search for videos.
2. **Task**: User selects a video from search results.
3. **Task**: User views video.
4. **End Event**: Video viewing process ends.

**iv. Class Diagram**

**v. Package Diagram**



**Frontend**

* **WebApp**: Represents the web application used by users.
* **MobileApp**: Represents the mobile application used by users.

**Backend**

* **APIServer**: The server that handles API requests from the frontend.
* **VideoProcessingServer**: The server responsible for processing video uploads.
* **AuthServer**: The server that handles user authentication and authorization.

**Services**

* **UserService**: Manages user-related operations such as registration, login, and profile updates.
* **VideoService**: Handles operations related to video uploads, retrieval, and management.
* **CommentService**: Manages comments on videos, including adding, deleting, and retrieving comments.
* **PlaylistService**: Handles creation, management, and retrieval of user playlists.
* **LikeService**: Manages likes and dislikes on videos.
* **Subscription Service**: Handles user subscriptions and notifications.

**Database**

* **UserDB**: Database storing user information.
* **VideoDB**: Database storing video metadata.
* **CommentDB**: Database storing comments on videos.
* **PlaylistDB**: Database storing playlists and their relationships to videos.
* **LikeDB**: Database storing likes and dislikes on videos.
* **SubscriptionDB**: Database storing user subscription information.

**Models**

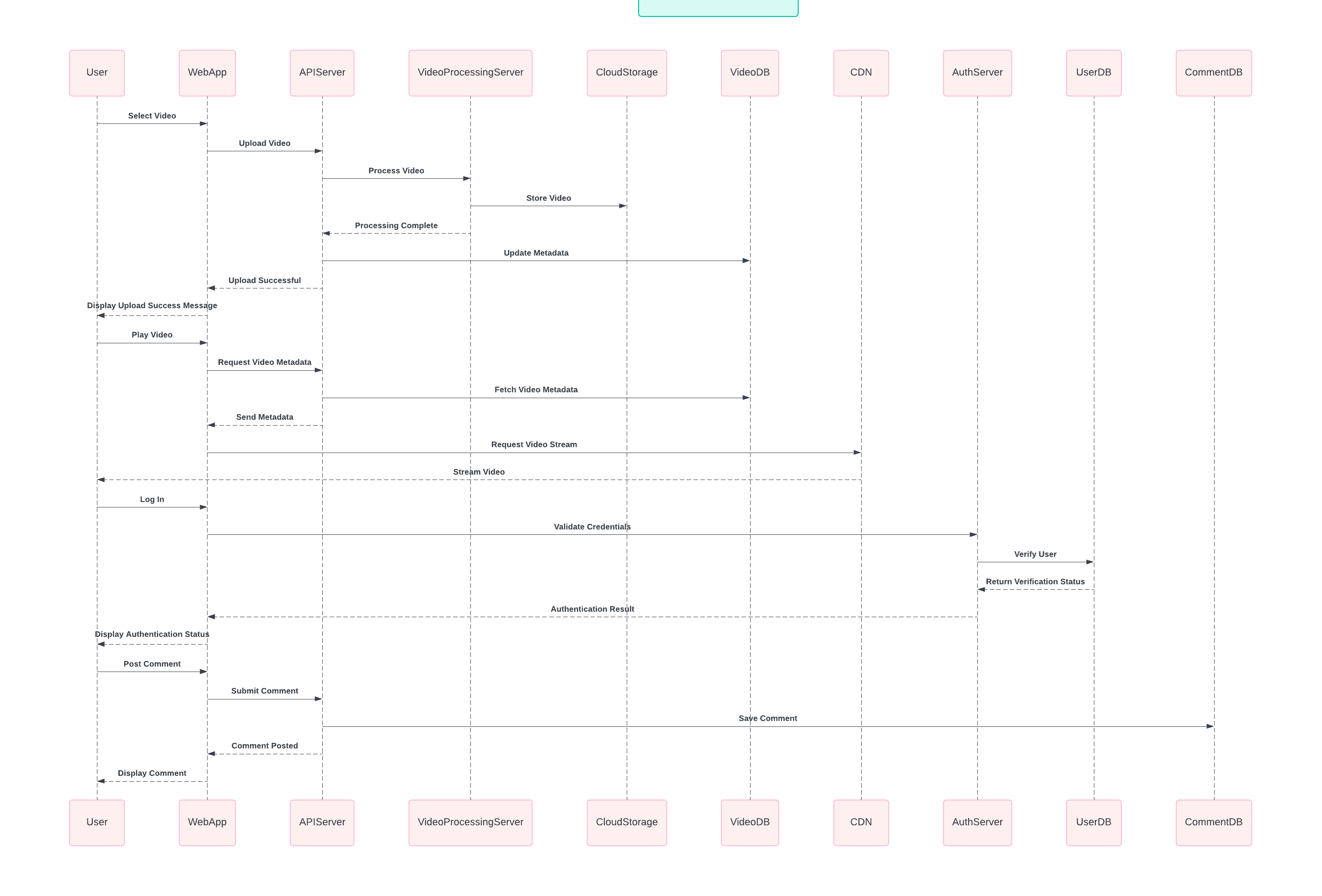
* **User**: Model class representing the user entity.
* **Video**: Model class representing the video entity.
* **Comment**: Model class representing the comment entity.
* **Playlist**: Model class representing the playlist entity.
* **VideoPlaylist**: Model class representing the relationship between videos and playlists.
* **Like**: Model class representing likes and dislikes on videos.
* **Subscription**: Model class representing user subscriptions.

**External Services**

* **CDN**: Content Delivery Network service for streaming videos.
* **CloudStorage**: Cloud storage service for storing video files.

#### vi. Communication Diagram

**Communication Diagram: Video Playback**



1. **Video Upload**:

* **User to WebApp**: The user selects a video to upload.
* **WebApp to APIServer**: The WebApp sends the video to the API server for uploading.
* **APIServer to VideoProcessingServer**: The API server forwards the video to the Video Processing Server for processing.
* **VideoProcessingServer to CloudStorage**: The Video Processing Server stores the processed video in cloud storage.
* **VideoProcessingServer to APIServer**: The Video Processing Server informs the API server that processing is complete.
* **APIServer to VideoDB**: The API server updates the video metadata in the database.
* **APIServer to WebApp**: The API server informs the WebApp that the upload is successful.
* **WebApp to User**: The WebApp displays a success message to the user.

2. **Video Playback**:

* **User to WebApp**: The user selects a video to play.
* **WebApp to APIServer**: The WebApp requests video metadata from the API server.
* **APIServer to VideoDB**: The API server fetches the video metadata from the database.
* **APIServer to WebApp**: The API server sends the metadata to the WebApp.
* **WebApp to CDN**: The WebApp requests the video stream from the CDN.
* **CDN to User**: The CDN streams the video to the user.

**3.User Authentication**:

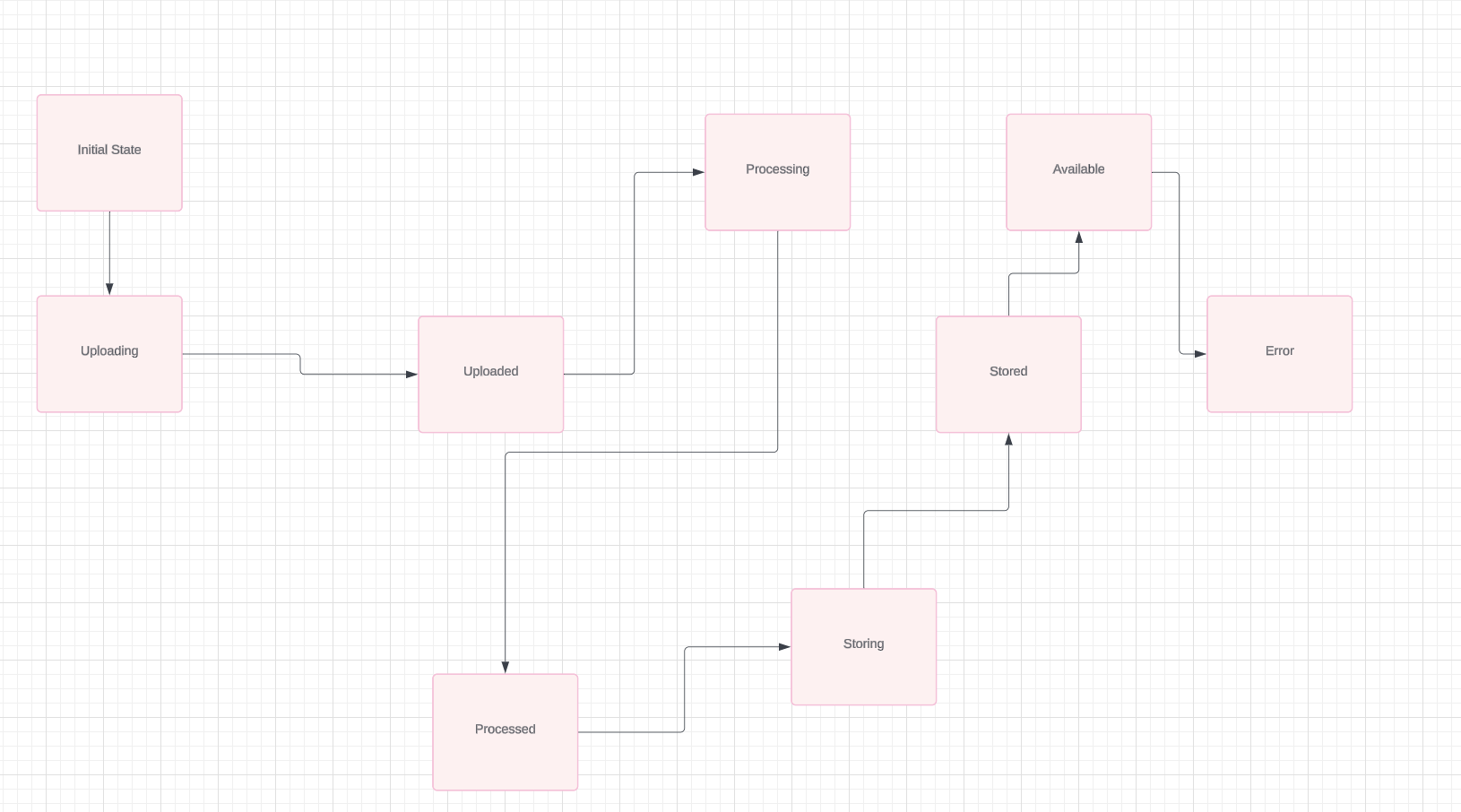
* **User to WebApp**: The user logs in by entering credentials.
* **WebApp to AuthServer**: The WebApp sends the credentials to the authentication server for validation.
* **AuthServer to UserDB**: The authentication server verifies the user against the database.
* **UserDB to AuthServer**: The database returns the verification status to the authentication server.
* **AuthServer to WebApp**: The authentication server returns the authentication result to the WebApp.
* **WebApp to User**: The WebApp displays the authentication status to the user.

**4**.**Comment Posting**:

* **User to WebApp**: The user posts a comment on a video.
* **WebApp to APIServer**: The WebApp submits the comment to the API server.
* **APIServer to CommentDB**: The API server saves the comment in the database.
* **APIServer to WebApp**: The API server informs the WebApp that the comment has been posted.
* **WebApp to User**: The WebApp displays the comment to the user.

#### vii. State Diagram

**State Diagram: Video Processing**



**States and Transitions:**

1. **Initial State**: The starting point when a video upload is initiated.
2. **Uploading**: The video is being uploaded to the server.
3. **Uploaded**: The video has been successfully uploaded.
4. **Processing**: The video is being processed (e.g., encoding, generating thumbnails).
5. **Processed**: The video has been processed.
6. **Storing**: The processed video is being stored in cloud storage.
7. **Stored**: The video has been successfully stored.
8. **Available**: The video is available for viewing.
9. **Error**: An error occurred during any of the steps.

**Transitions**

1. **Initial State to Uploading**:

**Trigger**: User initiates a video upload **Action**: The system starts uploading the video file.

1. **Uploading to Uploaded**:

**Trigger**: Video upload completes successfully. **Action**: The system confirms the video is uploaded.

1. **Uploaded to Processing**:

**Trigger**: Video upload is confirmed. **Action**: The system begins processing the video

1. **Processing to Processed**:

**Trigger**: Video processing completes successfully. **Action**: The system confirms the video has been processed.

1. **Processed to Storing**:

**Trigger**: Video processing is confirmed. **Action**: The system begins storing the processed video in cloud storage.

1. **Storing to Stored**:

**Trigger**: Video storage completes successfully. **Action**: The system confirms the video is stored.

1. **Stored to Available**:

**Trigger**: Video storage is confirmed. **Action**: The system makes the video available for viewing.

1. **Error State**:

**Trigger**: An error occurs during any step. **Action**: The system transitions to the Error state and may retry or alert the user.

### **c. DFD Diagrams**

#### Context Diagram



**Actors and System**

1. **User**: Represents an end-user who interacts with the system to upload and watch videos.
2. **System**: Represents the central video management system that handles video uploads, storage, streaming, and content management.
3. **Admin**: Represents an administrator who manages the content within the system.

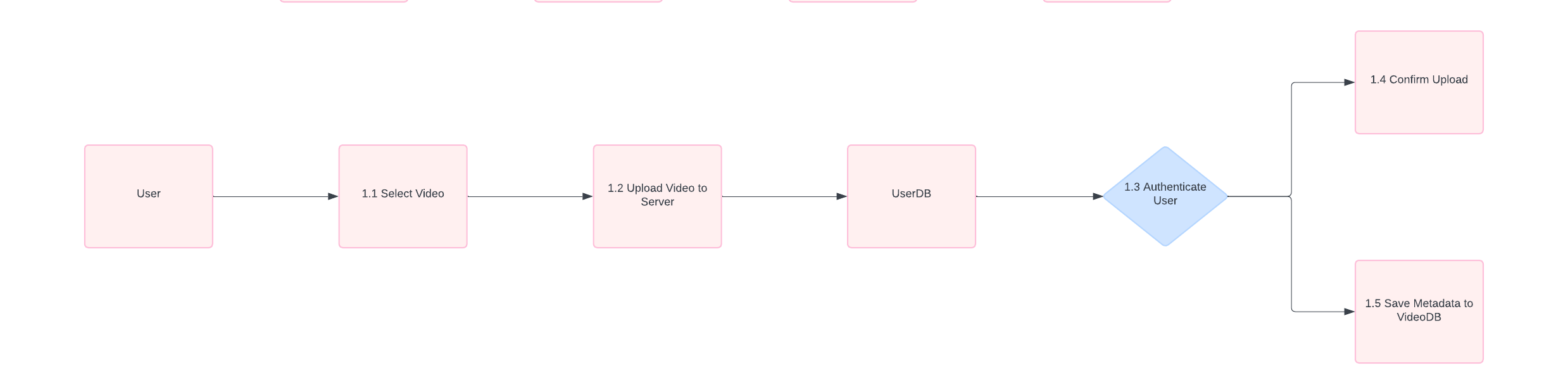
**Interactions**

1. **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.
2. **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.
3. **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.

Level-0 Diagram



Level-1 Diagram



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.

**d. E-R Diagram**

****

**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.

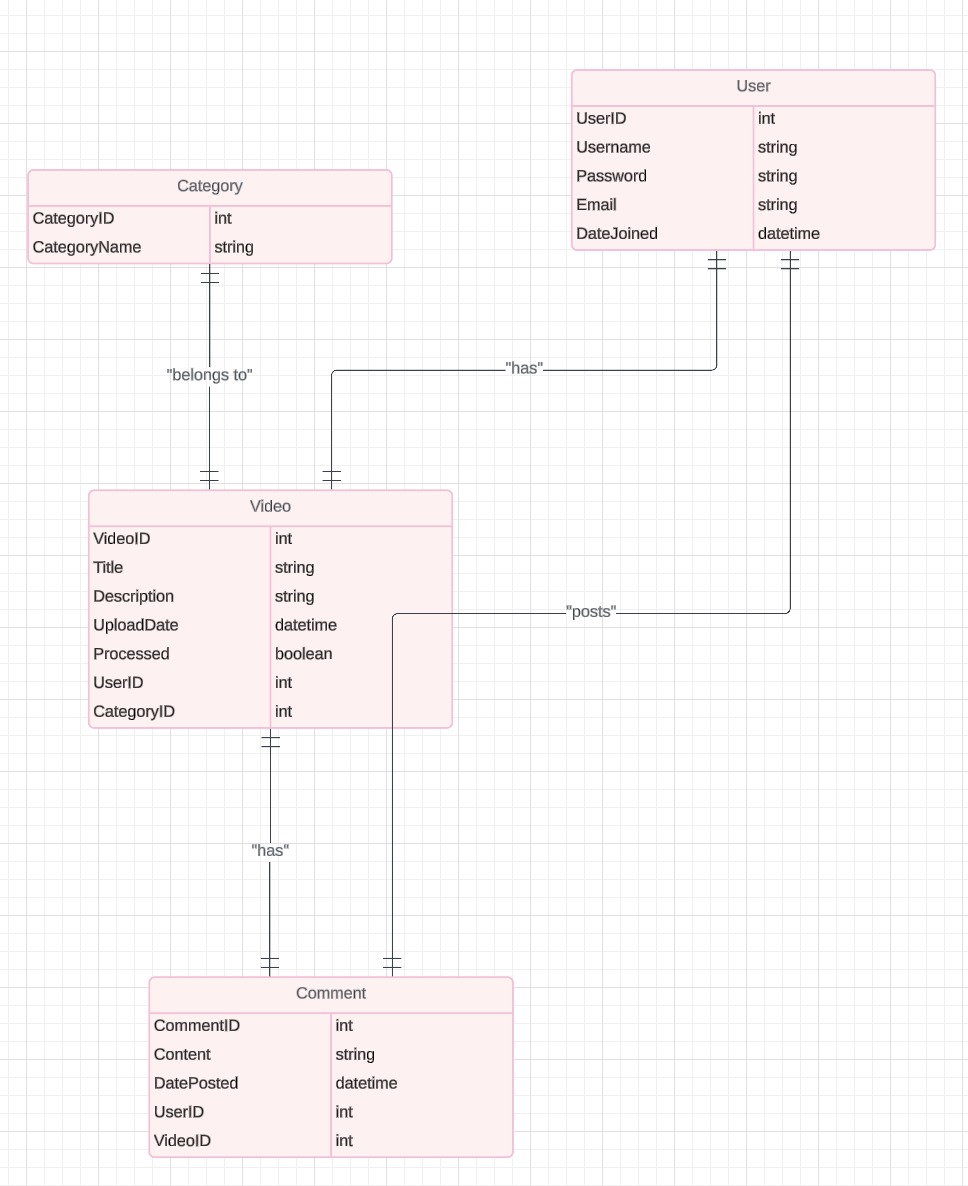
1. Category and Video:

Relationship: A Category can have multiple Videos.

Type: One-to-Many

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

**e. Relational Table**

****

**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

);

**f. Physical Database Tables**



**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

**Table 20 Algorithm Description for Main Functions**

| **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. |

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

Ensuring the quality and reliability of our web and mobile application is paramount. This section outlines the testing strategies and quality assurance measures undertaken to deliver a robust system.

**6.1 Testing Strategy**

**Unit Testing:**

Each function and module was tested individually to ensure they work as expected.

JavaScript functions and interactions were tested using frameworks such as Jasmine or Mocha.

**Integration Testing:**

The interaction between different modules (e.g., HTML with JavaScript) was tested to ensure seamless integration.

Focused on testing the interactions between the user interface and backend processes.

**End-to-End Testing:**

Simulated real user scenarios to verify the system works as intended from start to finish.

Used tools like Selenium for automated testing of user interactions.

**Cross-Browser Testing:**

Ensured the application works consistently across different web browsers (e.g., Chrome, Firefox, Safari, Edge).

Tested on various mobile browsers to guarantee a seamless mobile experience.

**Performance Testing:**

Evaluated the application's performance under various conditions, such as high traffic loads.

Used tools like Google Lighthouse to measure and improve loading times and responsiveness.

**Usability Testing:**

Conducted user testing sessions to gather feedback on the application's interface and usability.

Implemented changes based on user feedback to improve the overall user experience.

**6.2 Quality Assurance Measures**

**Code Reviews:**

Regular code reviews were conducted to maintain code quality and adherence to coding standards.

Peer reviews helped identify potential issues early in the development process.

**Version Control:**

Used Git for version control to track changes, manage code versions, and facilitate collaboration.

Ensured a stable codebase by merging and testing changes in a controlled manner.

**Continuous Integration:**

Implemented CI pipelines to automate the building and testing of the application.

Ensured that each code commit was tested automatically, reducing the likelihood of introducing bugs.

**Bug Tracking:**

Utilized tools like JIRA or Trello to track and manage reported bugs.

Prioritized and addressed bugs systematically to maintain a high-quality application.

**6.3 Testing Outcomes**

**Functionality**: All core functionalities were thoroughly tested and verified to work correctly.

**Compatibility**: Ensured compatibility across major browsers and mobile devices, providing a consistent user experience.

**Performance**: Optimized performance to ensure fast loading times and smooth interactions.

**User** **Satisfaction**: Received positive feedback from usability testing, with improvements made based on user suggestions.

Through these rigorous testing and quality assurance practices, we ensured that the web and mobile application meets high standards of reliability, performance, and user satisfaction.

# 7. USER GUIDE OF THE SYSTEM

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

**Index Page**

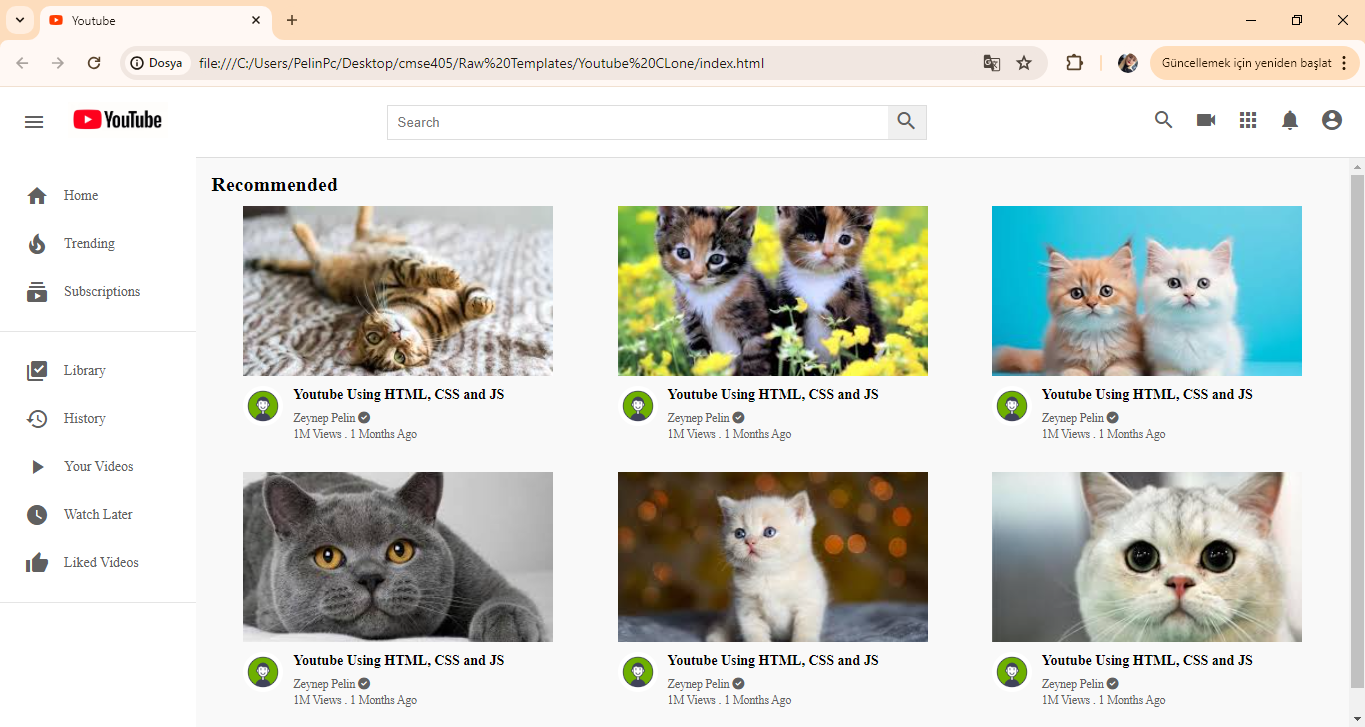


Figure 5 index page

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.

**About Page**



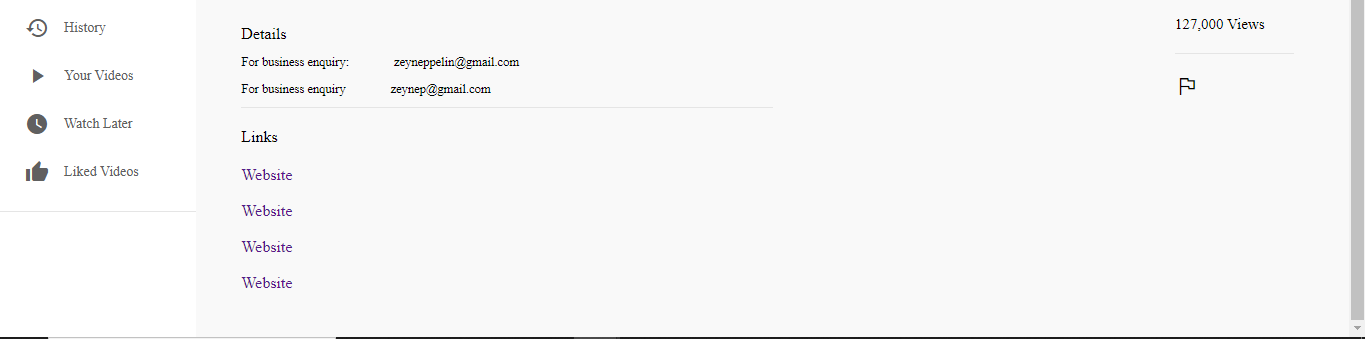
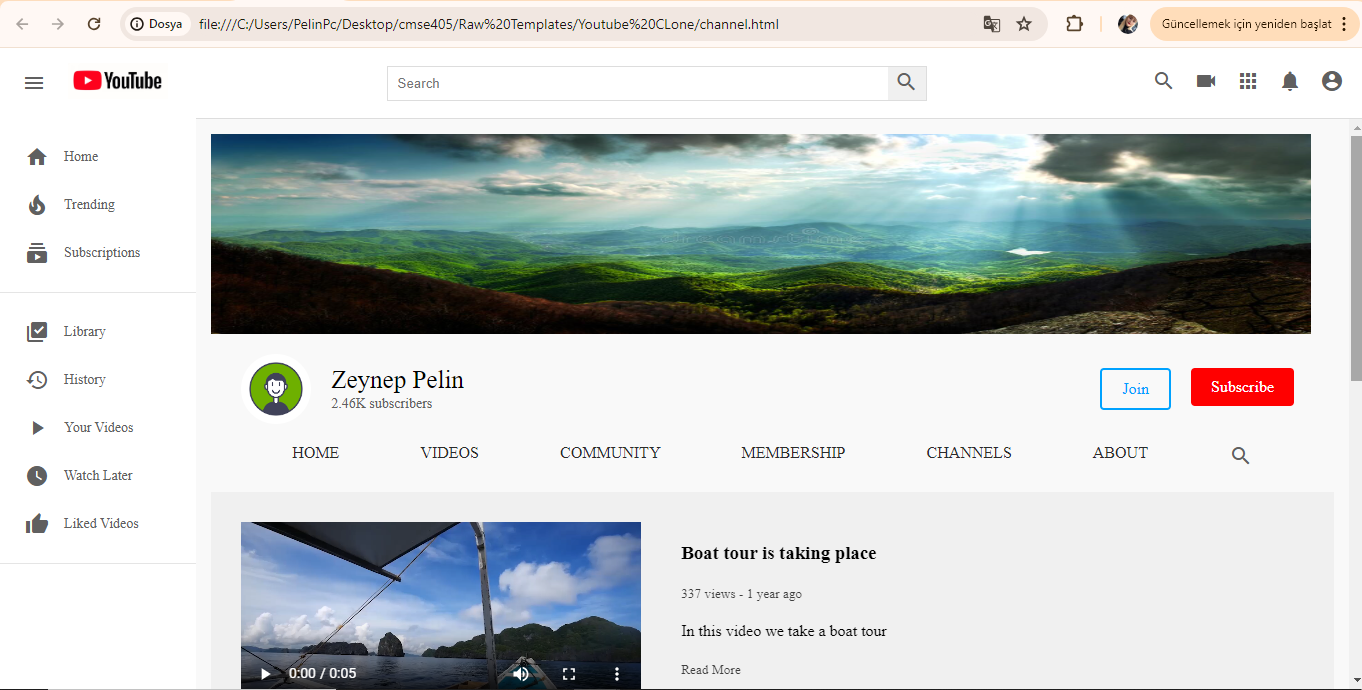


Figure 6 About page

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.

**Channel Page**



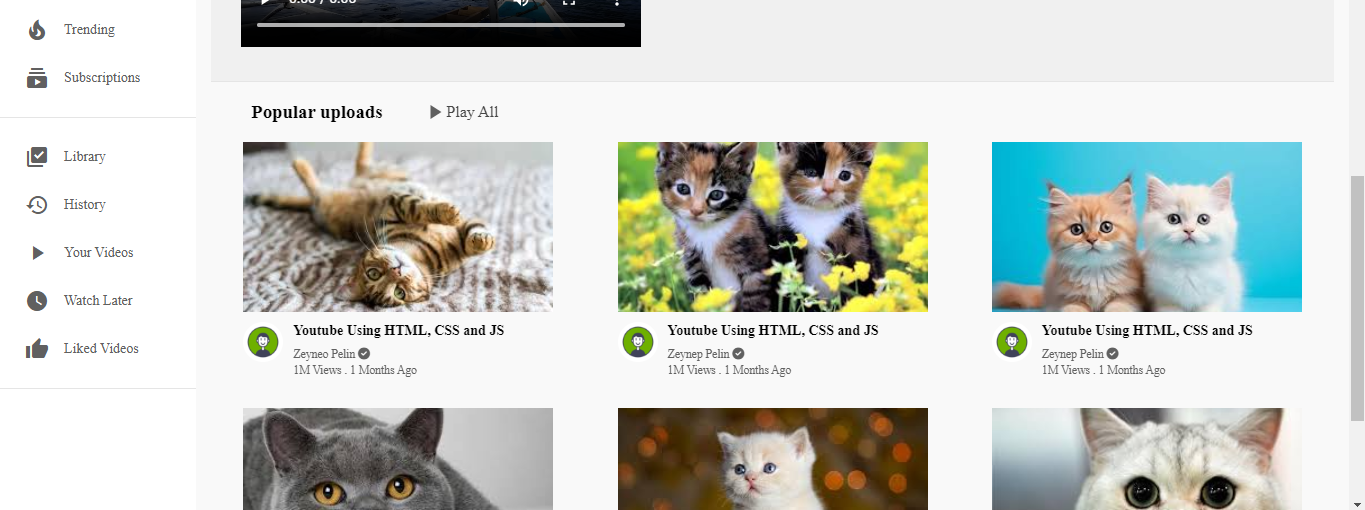


Figure 7 Channel Page

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.

**Channel Videos Page**

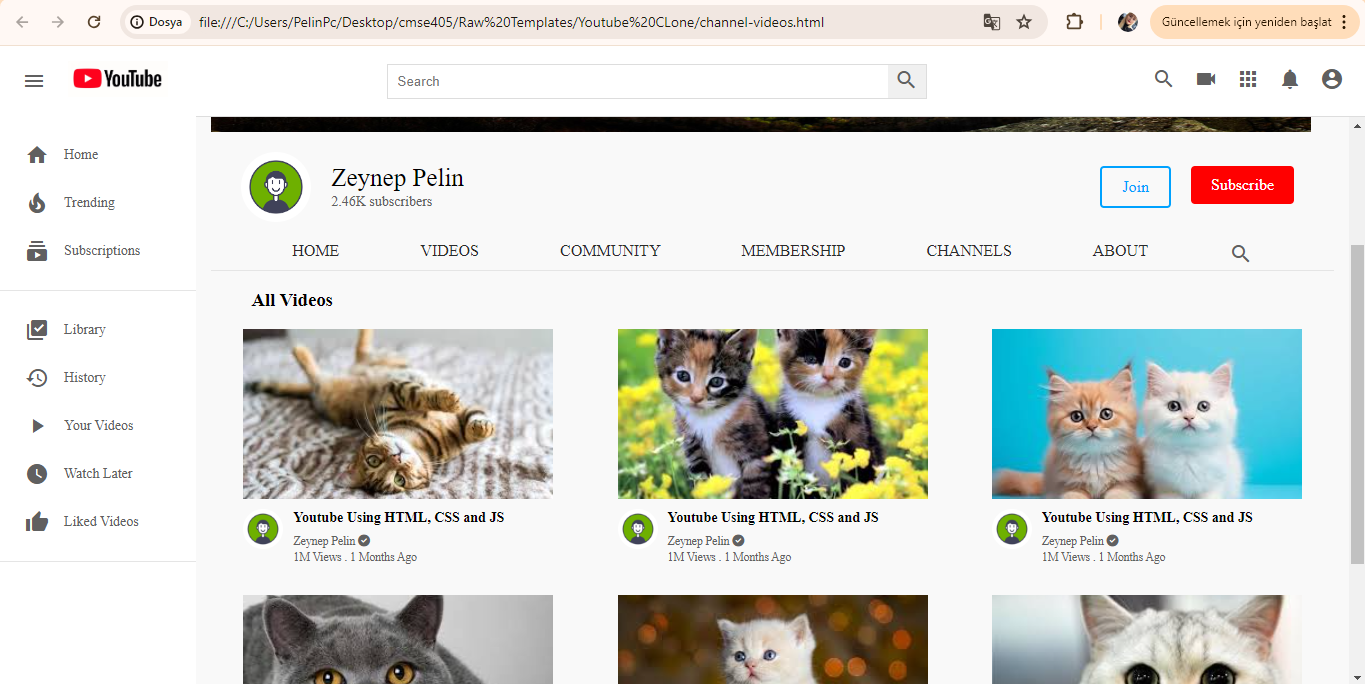


Figure 8 Channel videos page

Description: A detailed view of all videos uploaded by a particular channel. Users can browse through the list and select videos to watch.Key Features: Video thumbnails, titles, upload dates, and views.

**Community Page**

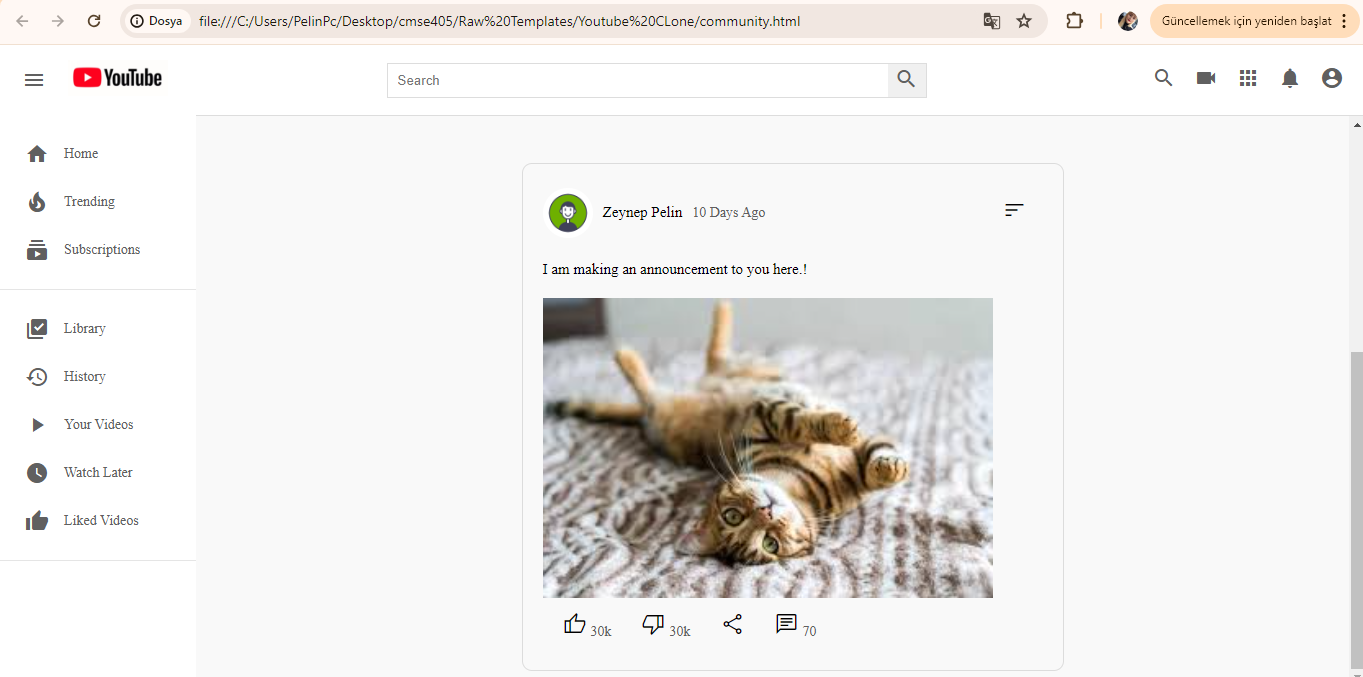


Figure 9 Community Page

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.

**Community Create Page**

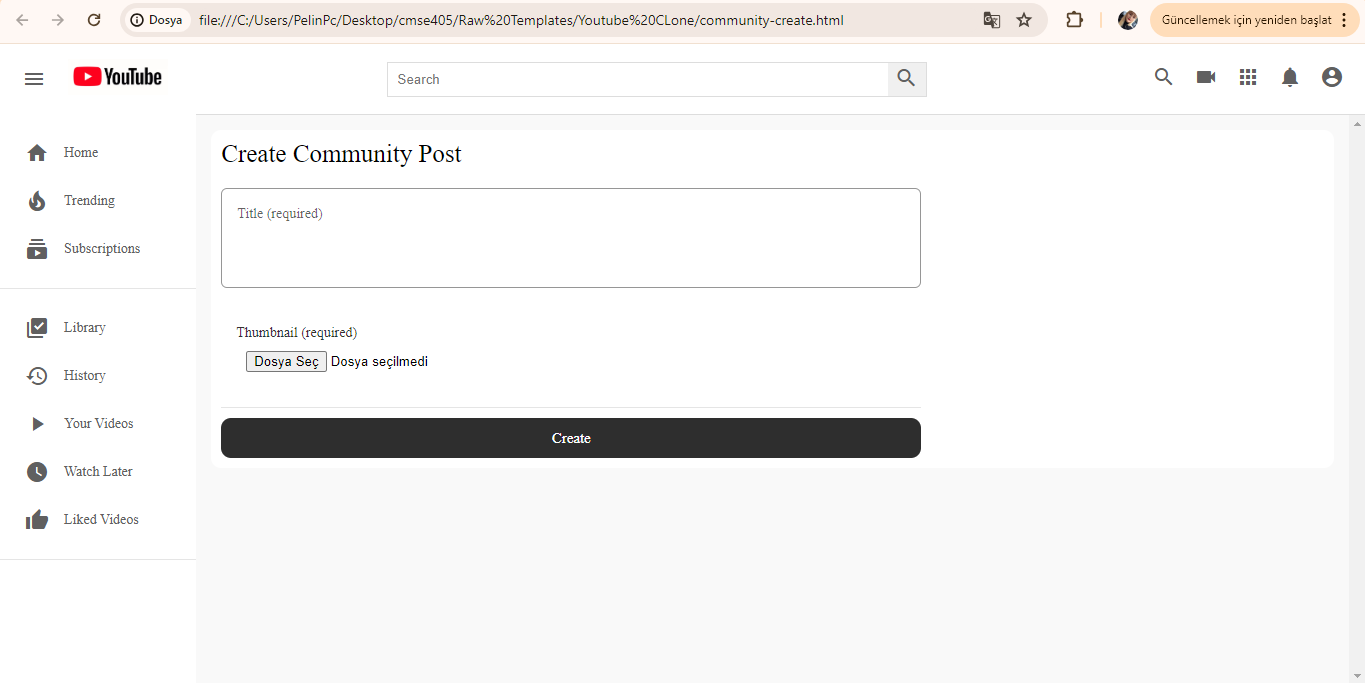


Figure 10 Community create page

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.

**Community Detail Page**

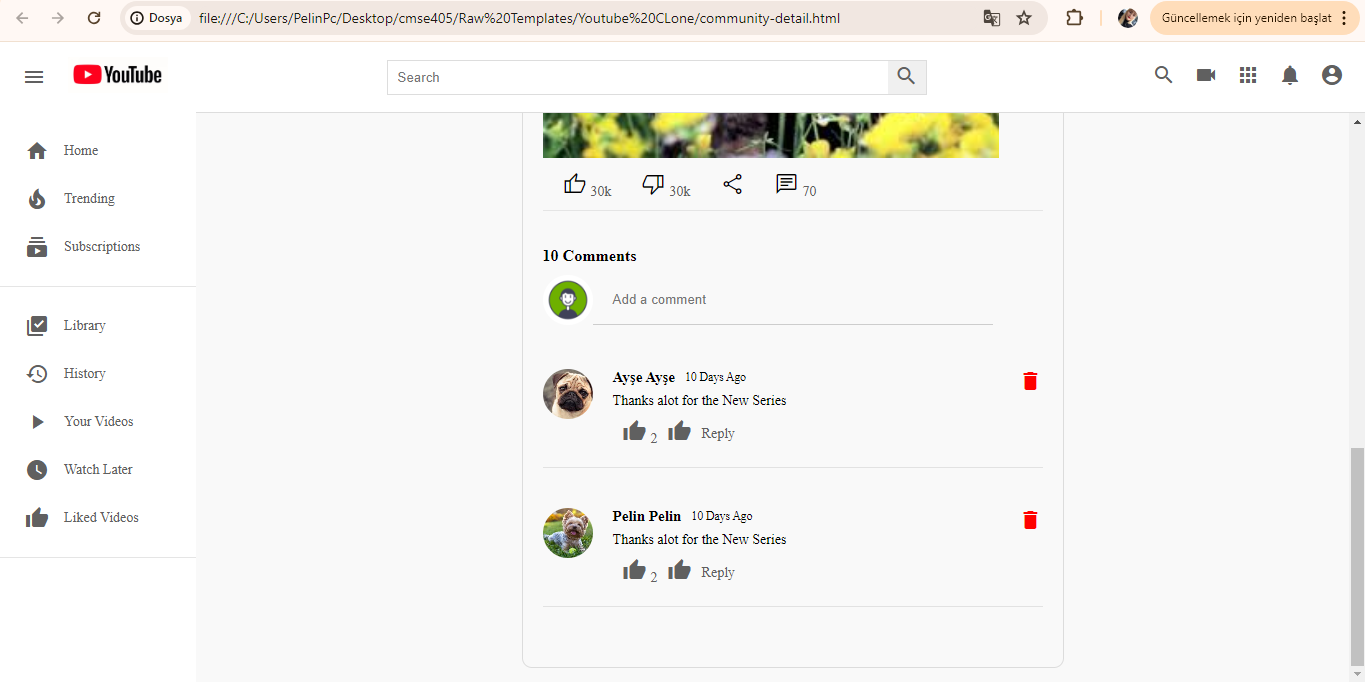


Figure 11 Community detail page

Description: A detailed view of a specific community post, including all comments and interactions. Users can participate in discussions by commenting on the post.Key Features: Post content, comment section, like and share buttons.

**Login Page**

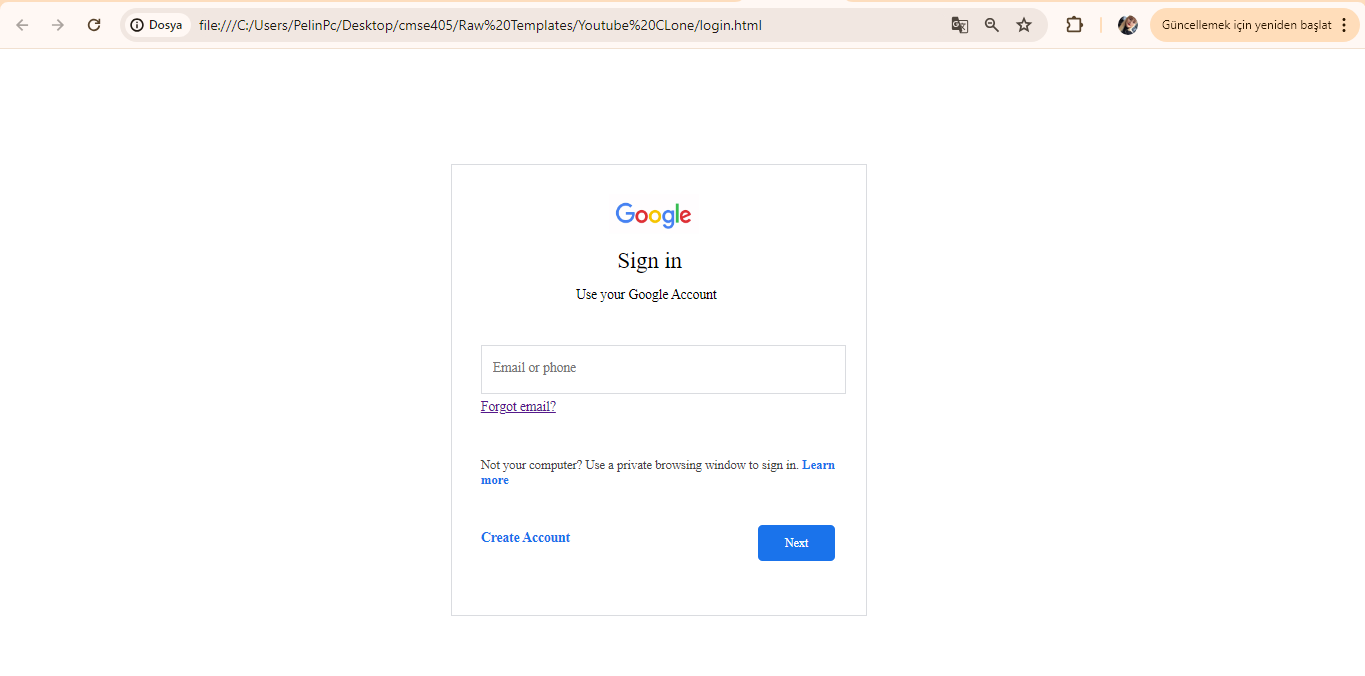
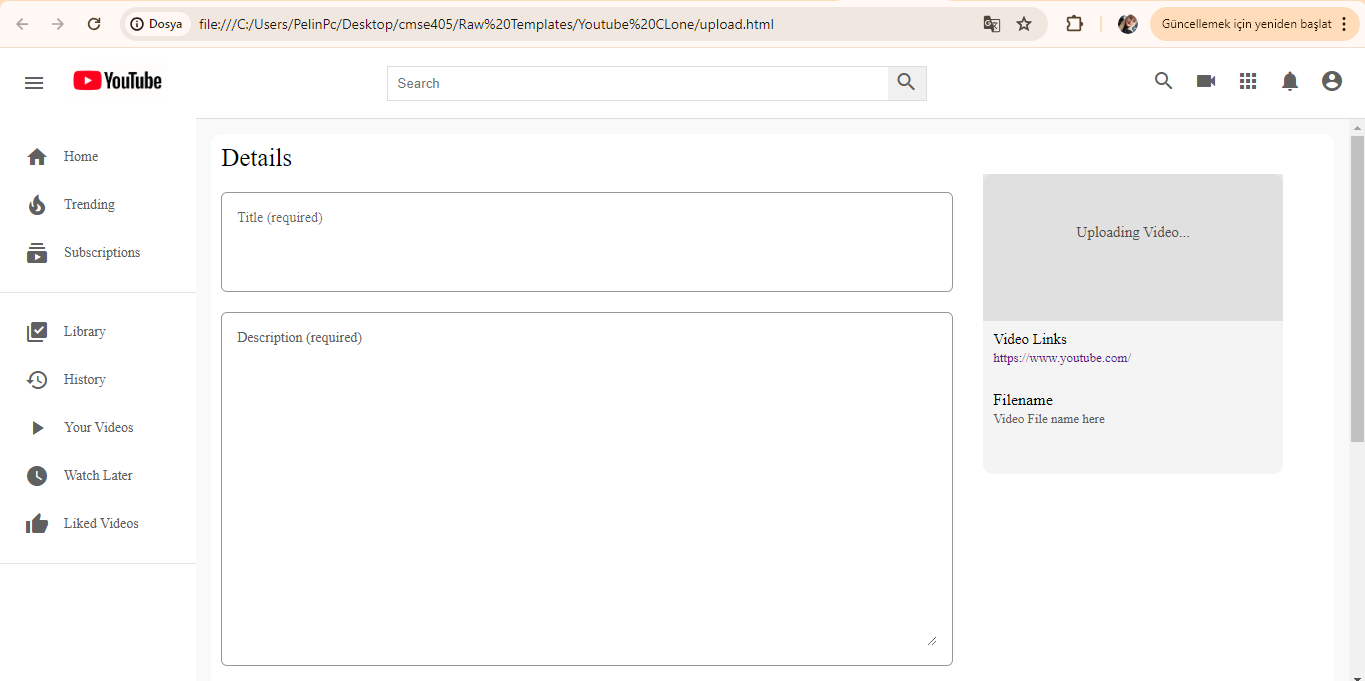


Figure 12 Login page

Description: The login page where users can enter their credentials to access their account. Users can also find options to sign up for a new account or reset their password if needed.Key Features: Username and password fields, login button, sign-up link, and password recovery option.

**Upload Page**



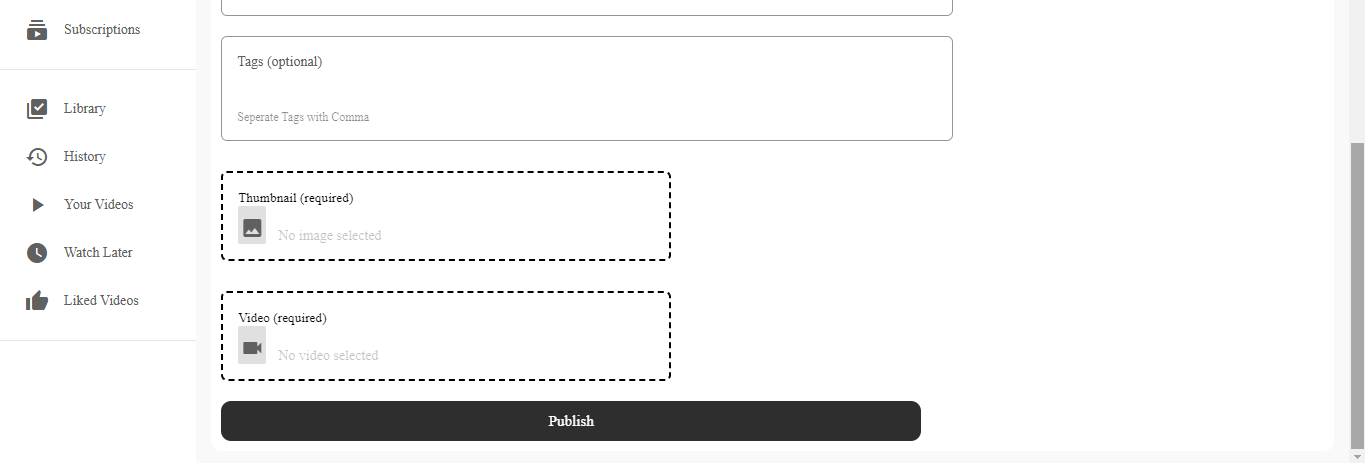
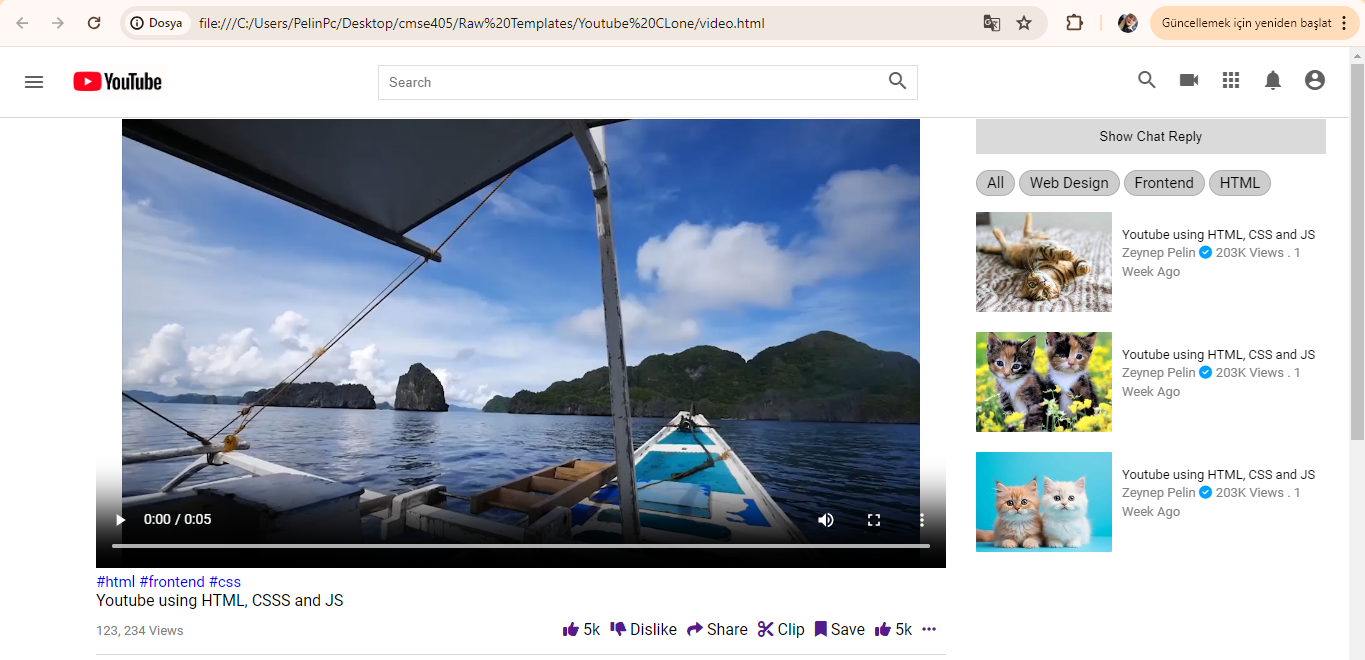


Figure 13 Upload page

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.

**Video Page**



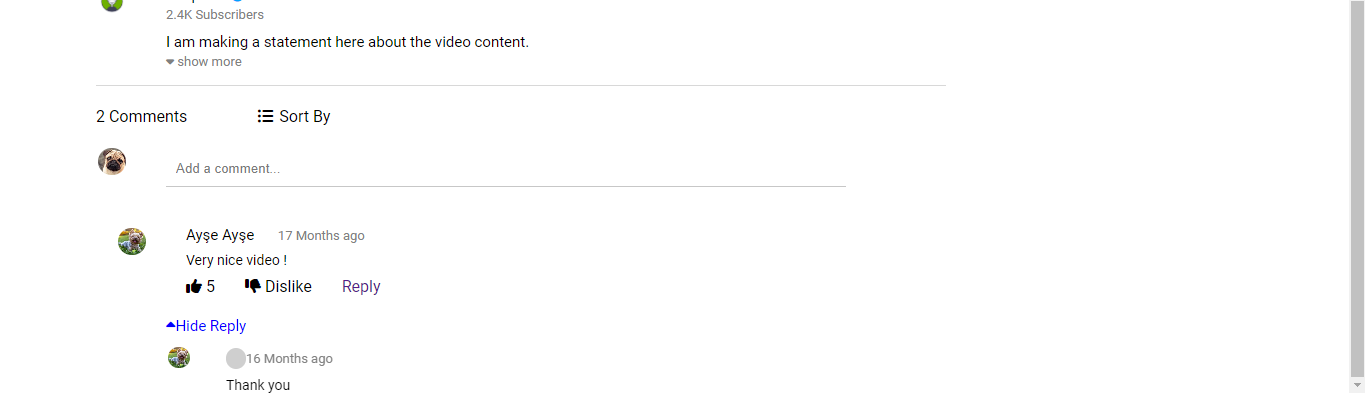


Figure 14 Video page

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">