Project Requirements Document

“Control-ef”

Table of Contents

Development Team

Htoo Lwin (120832)
Fathima Shafana (121985)
Gleb Cirkov (122002)

Problem

The recent COVID-19 pandemic has forced many educational institutions at all levels to conduct lectures online in lieu of physical classes. Most such online lectures are recorded by the institution for student review or administrative purposes and are hence are usually opened to student access. Students can use these videos to either review course material or learn about new concepts from courses not directly related to their curriculum.

Currently, however, navigating through the lecture content can prove to be time consuming and difficult without any prior processing of the videos. **The main difficulties are twofold**:

1. Students can take some time manually going through video lists to find the lectures of topics they would like to learn (if the videos are publicly accessible in the first place).
2. Students can also have a hard time searching through a particular video for a time when the lecturer is talking about the topic they want to learn about, especially if that student is reviewing the lecture.

Goals

The goals of the CONTROL-EF project are:

1. To provide a platform for students to easily access video lectures in an educational institution.
2. To provide students with the ability to quickly search for videos or timestamps using different facets such as tags, keywords, etc.

Stakeholders

The focus of CONTROL-EF are medium to large educational institutions. The system will support three different types of users:

1. **Students**. Students will consume the content in the system and will perform search, feedback, and video watching. Feedback can be regarding the transcripts and the videos themselves. Incorrect transcripts can be reported for editing.
2. **Lecturers.** Lecturers will upload and edit the video recordings of their lectures to the system via YouTube.
3. **Institution**. The institution is the organization responsible for the development and deployment of this system. They would want the system to achieve its intended goals.

User Roles

There will be three user roles:

1. **Users**. Users can only watch videos and give feedback.
2. **Lecturers**. Lecturers can upload videos and edit video transcripts. It is opted to have such a secondary role with less privileges than a full system **admin**.
3. **Admins**. Admins have full control over the system and can upload/update/delete videos. They can also ban users and videos.

Functional Requirements

Must-have Requirements

1. The lecturers should be able to upload video lectures through a YouTube link and edit the transcripts of the videos.
2. The students should be able to play the videos and their transcripts on YouTube.
3. The students should be able to request for edits of the transcripts.
4. The students should be able to perform a global search through every available video transcript.
5. The students should be able to perform multiple types of searches.
    a. The students should be able to search by tag.
    b. The students should be able to search by a single keyword.
    c. The students should be able to search by a phrase.
6. The students should be able to view search results showing the timestamp of the searched phrase or word.
7. The system should be able to generate transcripts using an external service.
8. All users should be able to see user-friendly error messages.
9. The students should be able to make comments on videos.

Good-to-have Requirements

1. The students should be able to search by synonyms of the desired tags/keywords. For example, a student searching for web development should also see software development videos as results.
2. The students should be able to report errors and flag videos. The error report should have a snapshot of the system state.
3. The students should be able to search in a specific video.
4. The admin and lecturers are able to make restrictions on who is able to watch videos.
5. The students should be able to make a list of favorite videos.
6. The users should be able to to filter their search results via multiple facets such as phrase frequency, upload date, views, etc.

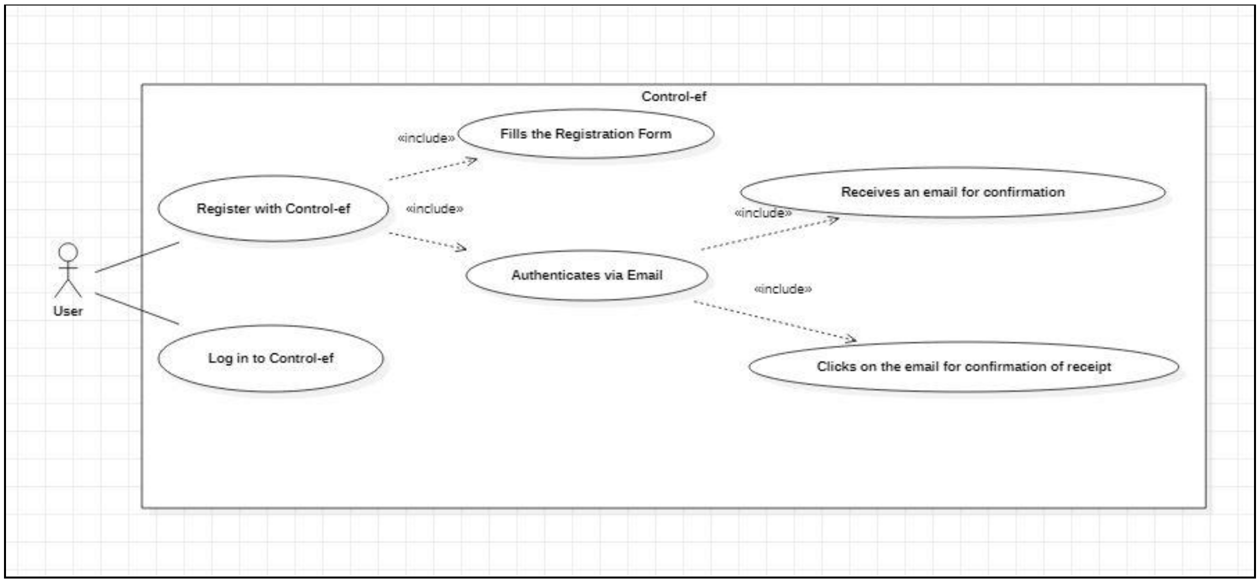
Non-Functional requirements

- **Availability**. The platform's main purpose is to provide aid in the learning process, so it should be available as much as possible. It should not fail the student when he needs it.
- **Usability**. The students should be able to easily access their desired videos from an accessible UI.
- **Performance**. When a student is learning, it is important not to lose focus. So, the platform's search function should work as quickly as possible.
- **Scalability**. As this platform is basically a database, any kind of expansion (most likely vertical) should be seamless and unnoticed by the user. The system should be able to handle large amounts of videos.

Use Cases

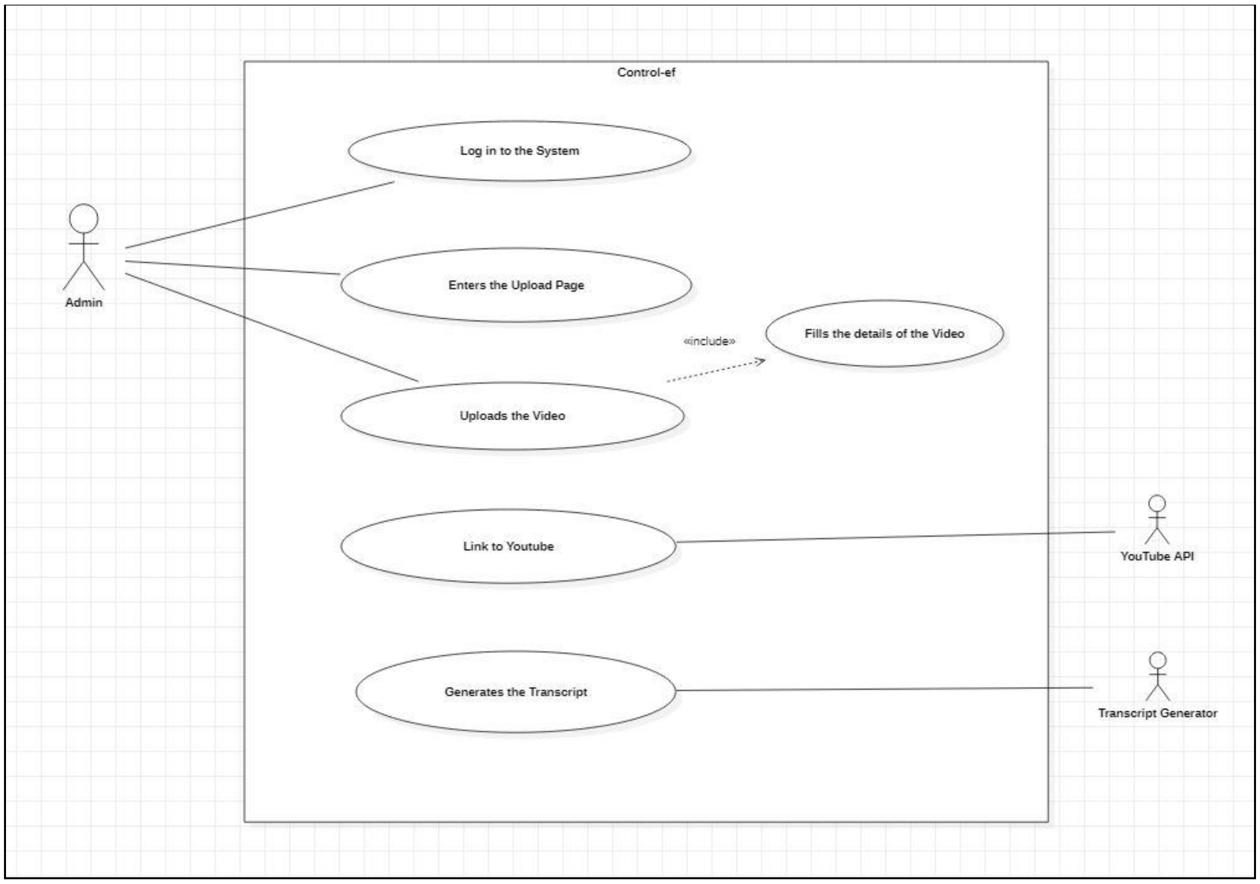
Use case: RegisterUser

Use case name	RegisterUser
Participating actors	Initiated by **User** Communicates with **System**
Flow of events	1. The **User** fills the Registration Form and submits it. 2. The **User** gets the Confirmation Email. 3. The **User** clicks to confirm the receipt of email. 4. The **User** Logs into the System.
Entry condition	● The **User** enters the Registration Page. ● The **User** is not logged in.
Exit condition	● The **User** successfully gets registered with the system. ● The **User** cancels the registration.



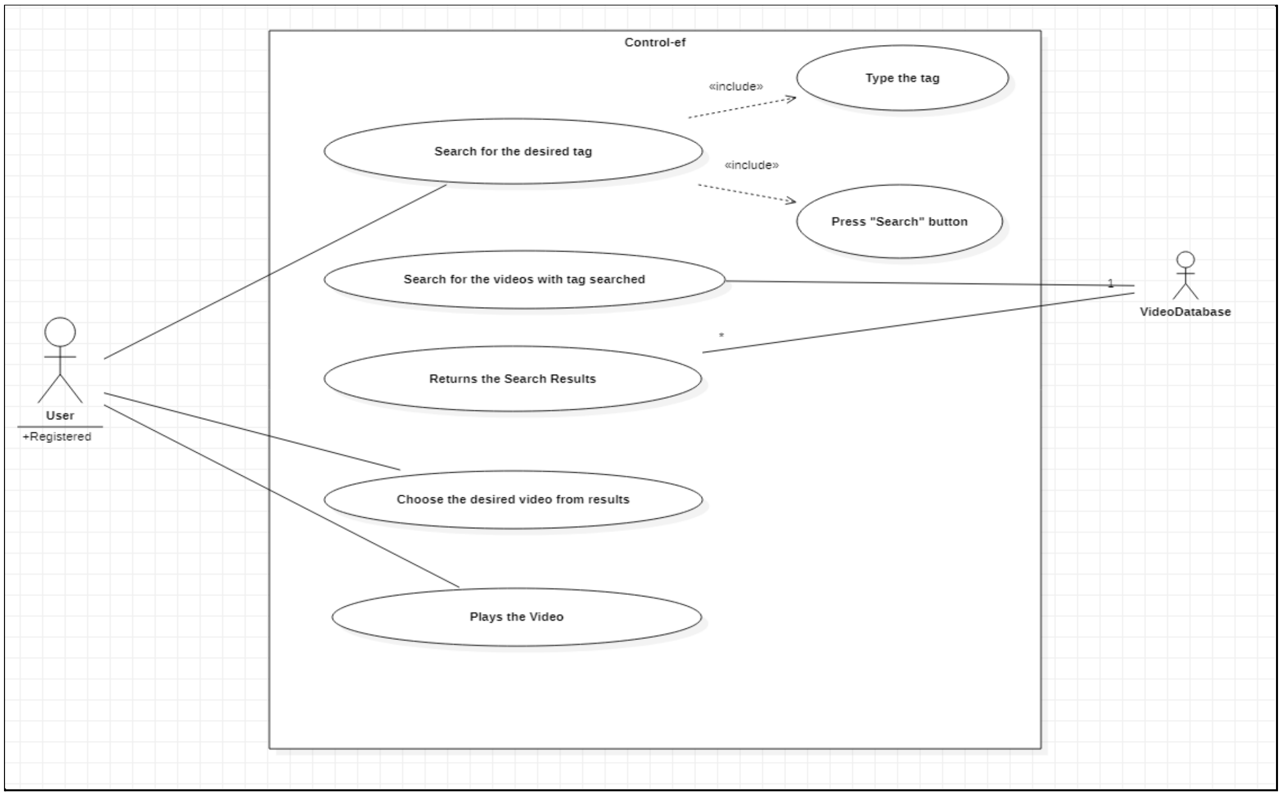
Use case: UploadVideo

Use case name	UploadVideo
Participating actors	Initiated by **Admin** Communicates with **System, TranscriptGenerator** and **YouTube**
Flow of events	1. The **Admin** enters the upload page. 2. The **Admin** fills out the details of the video. 3. The **Admin** uploads the video. 4. The **Video** gets uploaded and linked to YouTube. 5. The **TranscriptGenerator** generates the transcript. 6. The transcript is stored in the **Database.**
Entry condition	● The **Admin** is logged in.
Exit condition	● The **Admin** successfully uploads the video. ● The **Admin** cancels the upload.



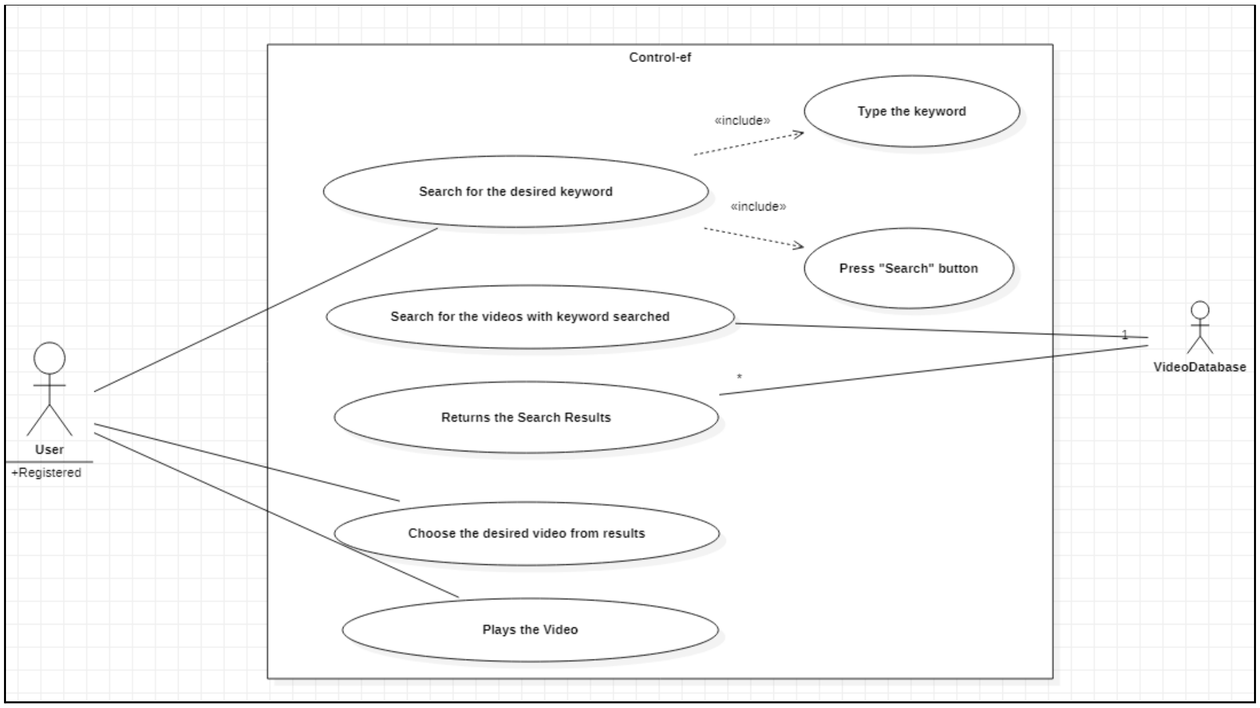
Use case: SearchTag

Use case name	SearchTag
Participating actors	Initiated by **User** Communicates with **System**
Flow of events	1. The **User** types the tag in the search bar and presses the search button. 2. The **System** searches for all videos under that predefined tag in the database. 3. The **System** displays the search results. 4. The **User** chooses the result that they desire. 5. The **User** plays the video.
Entry condition	● The **User** is logged in.
Exit condition	● The **User** finds the video that they are searching for and clicks on the video link to play it. ● The **User** cancels the search.


Control-ef
«include»
Type the tag
Search for the desired tag
«include»
Press "Search" button
Search for the videos with tag searched
1
VideoDatabase
*
Returns the Search Results
User
+Registered
Choose the desired video from results
Plays the Video

Use case: SearchKeyword

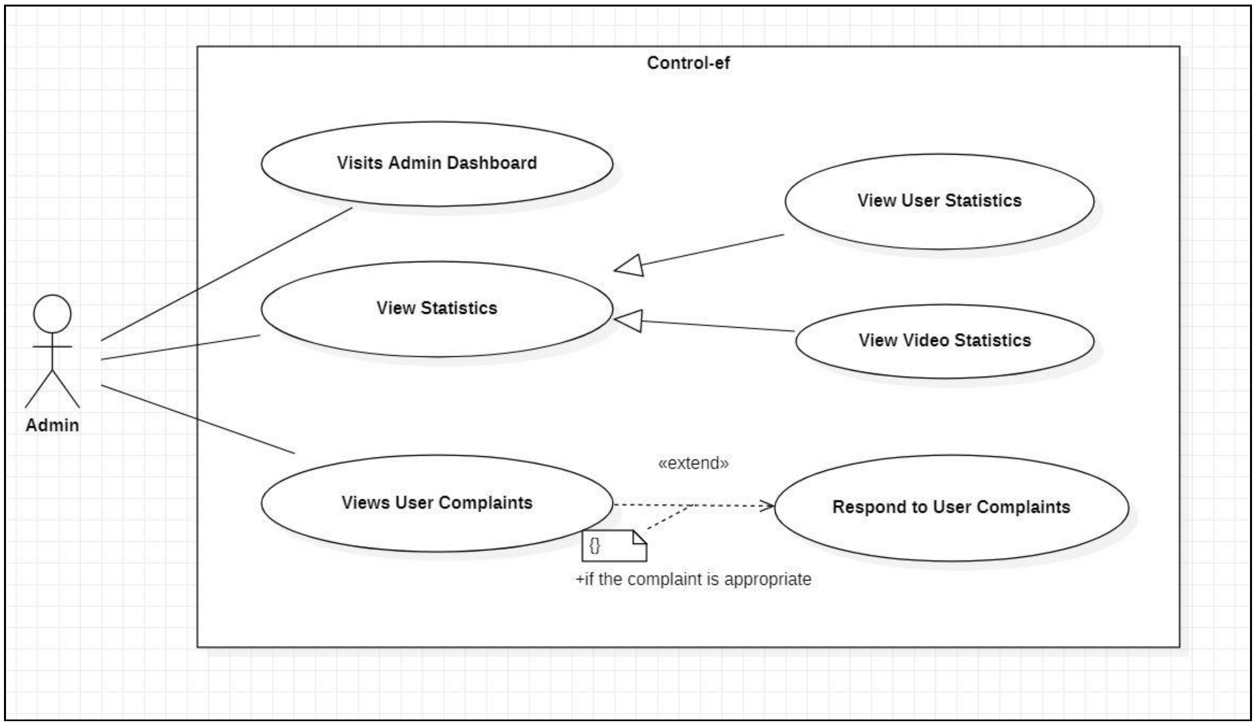
Use case name	SearchKeyword
Participating actors	Initiated by **User** Communicates with **System**
Flow of events	1. The **User** types the keyword in the search bar and presses the search button. 2. The System searches for the keyword in the transcripts of all videos in the database. 3. The **System** displays the search results. 4. The **User** chooses the result that they desire. 5. The **User** plays the video.
Entry condition	● The **User** is logged in.
Exit condition	● The **User** finds the video that they are searching for and clicks on the video link to play it. ● The **User** cancels the search.



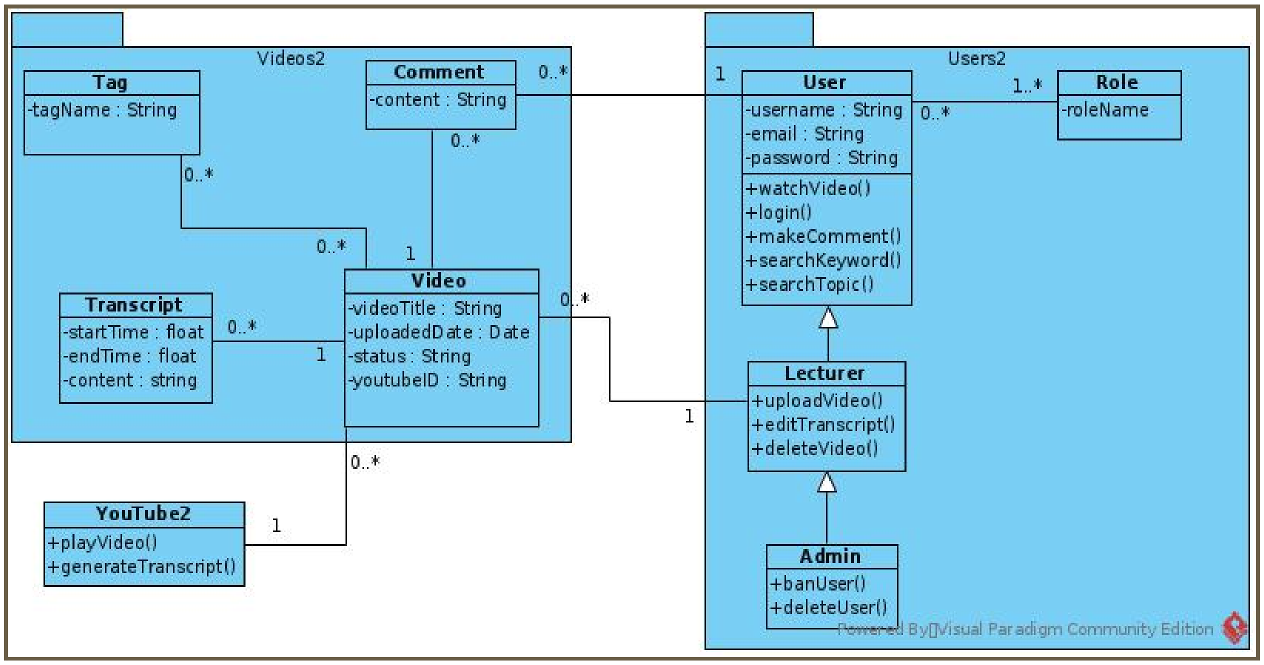
Control-ef
«include»
Type the keyword
Search for the desired keyword
«include»
Press "Search" button
Search for the videos with keyword searched
1
VideoDatabase
*
Returns the Search Results
User
+Registered
Choose the desired video from results
Plays the Video

Use case: AdminMonitoring

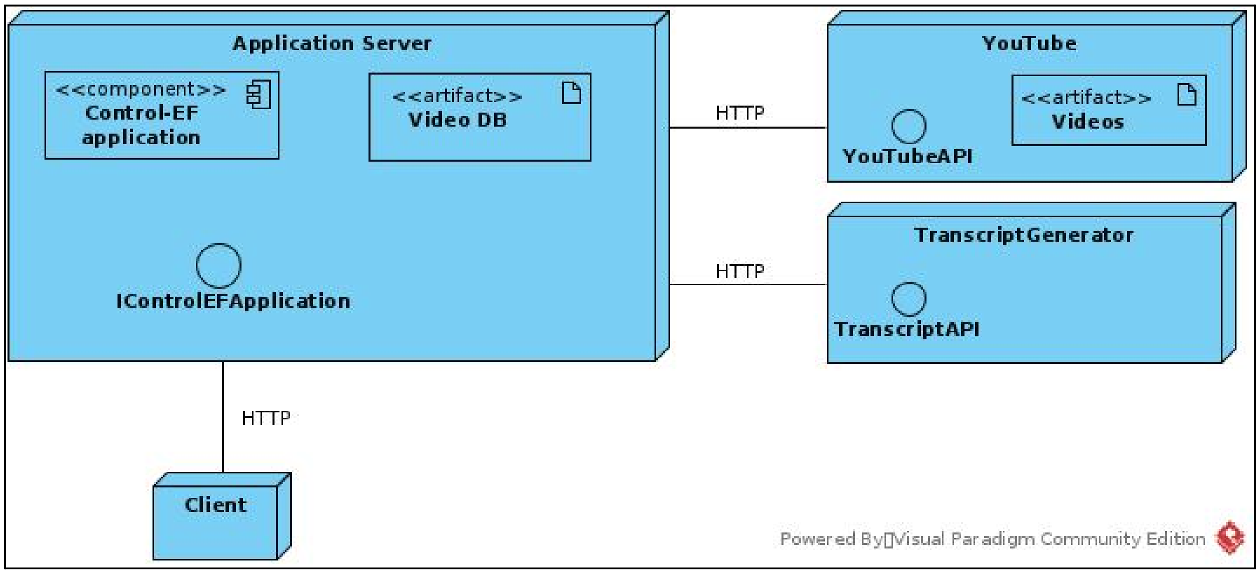
Use case name	AdminMonitoring
Participating actors	Initiated by **Admin** Communicates with **System**
Flow of events	1. The **Admin** visits the admin dashboard. 2. The **Admin** views user statistics. 3. The **Admin** views video statistics. 4. The **Admin** views user complaints. 5. The **Admin** responds to user complaints.
Entry condition	● The **Admin** is logged in.
Exit condition	● The **Admin** leaves the admin dashboard page.



Class Diagram

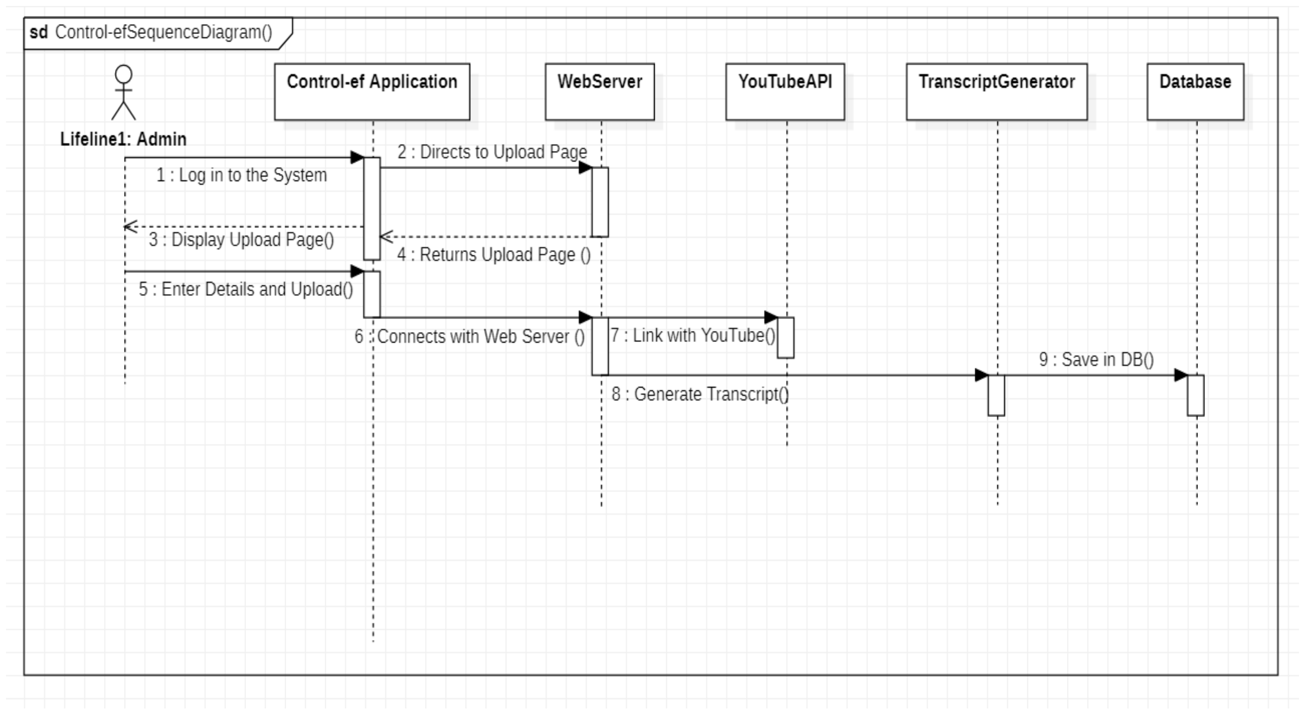


Architecture and Deployment Diagram

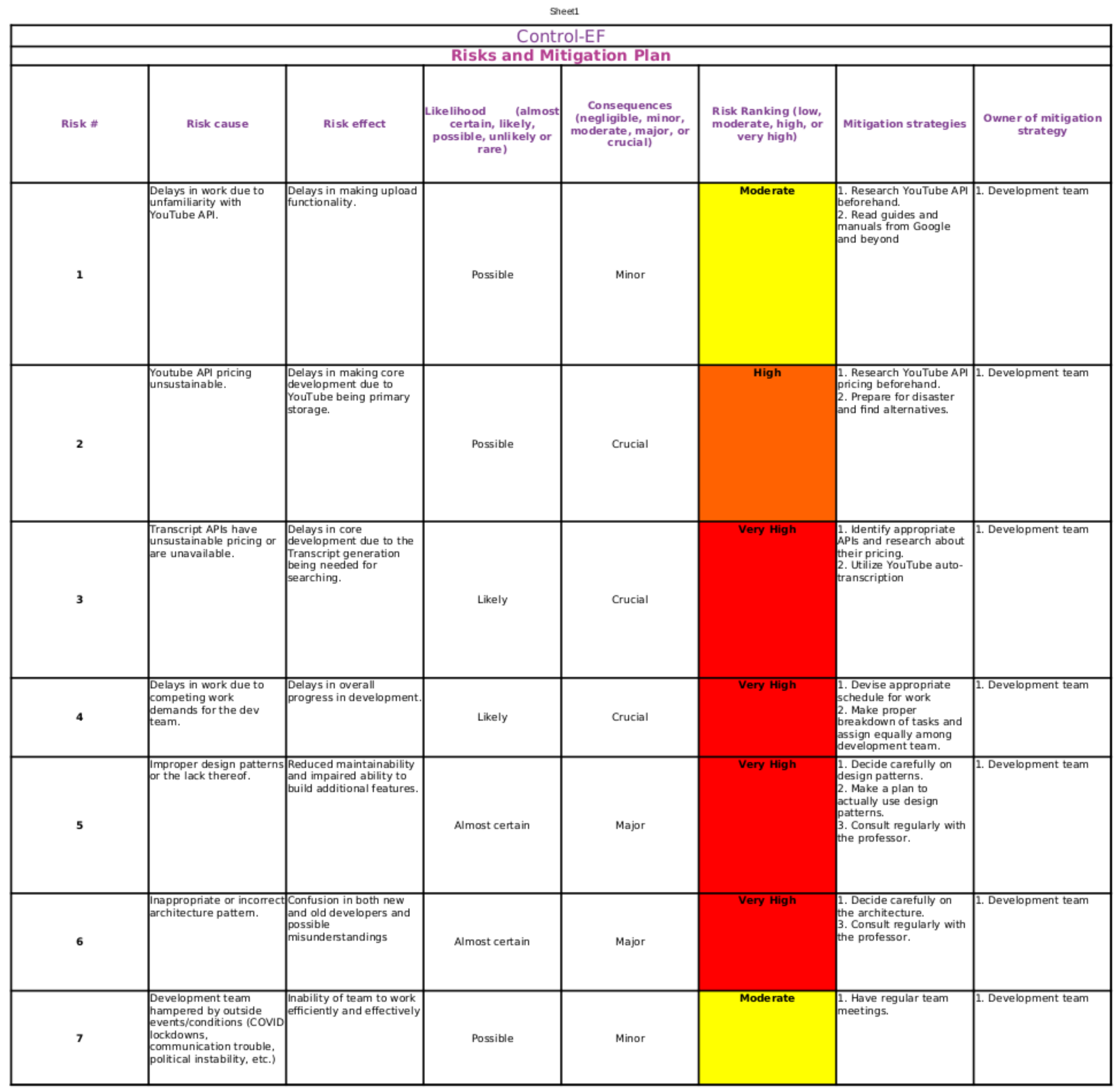


The main ControlEF application will utilize the **model-view-controller (MVC)** architecture pattern as we will be using the MVC framework Spring Boot. The architecture regarding the YouTube and Transcription APIs will utilize **microservices** in the form of the aforementioned APIs themselves. It can also be classified as a **service-oriented architecture (SOA)**.

Sequence Diagram


sd Control-efSequenceDiagram()
Lifeline1: Admin
Control-ef Application
WebServer
YouTubeAPI
TranscriptGenerator
Database
1 : Log in to the System
2 : Directs to Upload Page
3 : Display Upload Page()
4 : Returns Upload Page ()
5 : Enter Details and Upload()
6 : Connects with Web Server ()
7 : Link with YouTube()
8 : Generate Transcript()
9 : Save in DB()

Risk and Mitigation Plan



Sheet1

Control-EF

Risks and Mitigation Plan

Risk #	Risk cause	Risk effect	Likelihood (almost certain, likely, possible, unlikely or rare)	Consequences (negligible, minor, moderate, major, or crucial)	Risk Ranking (low, moderate, high, or very high)	Mitigation strategies	Owner of mitigation strategy
1	Delays in work due to unfamiliarity with YouTube API.	Delays in making upload functionality.	Possible	Minor	Moderate	1. Research YouTube API beforehand. 2. Read guides and manuals from Google and beyond	1. Development team
2	Youtube API pricing unsustainable.	Delays in making core development due to YouTube being primary storage.	Possible	Crucial	High	1. Research YouTube API pricing beforehand. 2. Prepare for disaster and find alternatives.	1. Development team
3	Transcript APIs have unsustainable pricing or are unavailable.	Delays in core development due to the Transcript generation being needed for searching.	Likely	Crucial	Very High	1. Identify appropriate APIs and research about their pricing. 2. Utilize YouTube auto-transcription	1. Development team
4	Delays in work due to competing work demands for the dev team.	Delays in overall progress in development.	Likely	Crucial	Very High	1. Devise appropriate schedule for work 2. Make proper breakdown of tasks and assign equally among development team.	1. Development team
5	Improper design patterns or the lack thereof.	Reduced maintainability and impaired ability to build additional features.	Almost certain	Major	Very High	1. Decide carefully on design patterns. 2. Make a plan to actually use design patterns. 3. Consult regularly with the professor.	1. Development team
6	Inappropriate or incorrect architecture pattern.	Confusion in both new and old developers and possible misunderstandings	Almost certain	Major	Very High	1. Decide carefully on the architecture. 3. Consult regularly with the professor.	1. Development team
7	Development team hampered by outside events/conditions (COVID lockdowns, communication trouble, political instability, etc.)	Inability of team to work efficiently and effectively	Possible	Minor	Moderate	1. Have regular team meetings.	1. Development team

****