



**MBARARA UNIVERSITY OF SCIENCE AND  
TECHNOLOGY**

**FACULTY OF COMPUTING AND INFORMATICS**

**WEB APPLICATION DEVELOPMENT**

**CSC2207**

**Youtube Clone App**

Name: **KAYONDO ABDULATIF**  
Reg No: **2021/BCS/037/PS**

# Contents

<b>1 Introduction.....</b>	<b>3</b>
<b>2 System Architecture.....</b>	<b>4</b>
1.1 Existing system.....	4
1.1.1 Statistics.....	4
2.1.1 Tech Stack.....	5
2.2 Proposed system.....	5
2.2.1 Features.....	5
2.2.2 Tech Stack.....	5
<b>3 Entity Relationship Diagram.....</b>	<b>5</b>
3.1 ER Diagram and Introduction.....	6
3.2 Entities and Relationships in our App.....	6
3.2.1 Entity Types.....	6
3.2.2 Relationship Types.....	7
3.3 Conversion of ER Diagram to Relational schema.....	7
<b>4 Normalisation.....</b>	<b>8</b>
4.1 Introduction.....	8
4.2 Functional Dependency in the Youtube Database.....	8
4.3 Normal forms of the Database.....	8
<b>5 User Authentication and Security Measures.....</b>	<b>8</b>
<b>6 Features Implemented and Screenshots.....</b>	<b>9</b>
<b>7 Conclusion and References.....</b>	<b>12</b>
7.1 Conclusion.....	12
7.2 References.....	12

# Introduction

YouTube is an American online video-sharing platform headquartered in San Bruno, California. Three former PayPal employees—Chad Hurley, Steve Chen, and Jawed Karim—created the service in February 2005. Google bought the site in November 2006 for US 1.65 billion; YouTube now operates as one of Google's subsidiaries.

Since YouTube works on such a massive scale, we decided to use this as an inspiration for the project. YouTube's database schemas are one of the most complicated ones currently being scaled and used massively impacting billions of users! YouTube allows users to upload, view, rate, share, add to playlists, report, comment on videos, and subscribe to other users. It offers a wide variety of user-generated and corporate media videos. Available content includes video clips, TV show clips, music videos, short and documentary films, audio recordings, movie trailers, live streams, and other content such as video blogging, short original videos, and educational videos. Most content on YouTube is uploaded by individuals, but media corporations including CBS, the BBC, Vevo, and Hulu offer some of their material via YouTube as part of the YouTube partnership program. Unregistered users can only watch (but not upload) videos on the site, while registered users are also permitted to upload an unlimited number of videos and add comments to videos. Videos that are age-restricted are available only to registered users affirming themselves to be at least 18 years old. YouTube and selected creators earn advertising revenue from Google AdSense, a program that targets ads according to site content and audience. The vast majority of its videos are free to view, but there are exceptions, including subscription-based premium channels, film rentals, as well as YouTube Music and YouTube Premium, subscription services respectively offering premium and ad-free music streaming, and ad-free access to all content, including exclusive content commissioned from notable personalities. As of February 2017, there were more than 400 hours of content uploaded to YouTube each minute, and one billion hours of content being watched on YouTube every day. As of August 2018, the website is ranked as the second-most popular site in the world, according to Alexa Internet, just behind Google. As of May 2019, more than 500 hours of video content are uploaded to YouTube every minute. Based on reported quarterly advertising revenue, YouTube is estimated to have US 15 billion in annual revenues.

Since YouTube works on such a massive scale, we decided to use this as an inspiration for the project. YouTube's database schemas are one of the most complicated ones currently being scaled and used massively impacting billions of users!

The driving force behind this project is YouTube's global impact on how we consume and share video content. YouTube's seamless interface, personalized recommendations and interactive features have created a digital ecosystem where users can engage with a variety of content. My goal is to take this essence and provide a similar experience for users with my clone

## System Architecture

Since the original architecture of Youtube is complex and implementing is out of scope for the current scenario, we will be discussing the differences of both the systems

## Existing system

### **Statistics**

- 4 billion views a day
- 60 hours of video is uploaded every minute
- 350+ million devices are YouTube enabled
- Revenue doubled in 2010
- The number of videos has gone up 9 orders of magnitude and the number of developers has only gone up two orders of magnitude.
- 1 million lines of Python code

### **Tech Stack**

- Python - most of the lines of code for YouTube are still in Python. Every time you watch a YouTube video you are executing a bunch of Python code.
- Apache - Apache keeps Youtube simple. Every request goes through Apache.
- Linux - The benefit of Linux is there's always a way to get in and see how your system is behaving. No matter how bad your app is behaving, you can take a look at it with Linux tools like strace and tcpdump.
- MySQL - is used a lot. When you watch a video you are getting data from MySQL. Sometimes it's used a relational database or a blob store. It's about tuning and making choices about how you organize your data.

These are some of the technologies which Youtube uses in their current system. They also use Zookeeper (distributed lock system), Vitess (frontend for SQL), and other templating engines to keep the app up and running.

## Proposed system

### **Features**

In our implementation of the popular video streaming service, we have incorporated the following features for our web application.

1. Users must be able to create a personal account.
2. Each registered user must have their own personal account page.
3. Users must be able to log in to the system and logout from the system.
4. Users must be able to watch videos in the system when they log in or logout.

### **Tech Stack**

To implement the above features, we have used the following tech stack

- **Django** - Lightweight web framework which can scale on the fly. Has added layer of security to prevent malware attacks
- **SQLite Database** - Employed as the backend database to store user data, video metadata, and related information.
- **Frontend** - HTML, CSS and JS were used for fast templating

## Entity Relationship Diagram

### ER Diagram and Introduction

An E-R model is usually the result of systematic analysis to define and describe what is important to processes in an area of a business. It does not define the business processes; it only presents a business data schema in graphical form. It is usually drawn in a graphical form as boxes (entities) that are connected by lines (relationships) which express the associations and dependencies between entities.

An ER model can also be expressed in a verbal form, for example: one building may be divided into zero or more apartments, but one apartment can only be located in one building.

Entities may be characterized not only by relationships, but also by additional properties (attributes), which include identifiers called "primary keys". Diagrams created to represent attributes as well as entities and relationships may be called entity-attribute-relationship diagrams, rather than entity-relationship models.

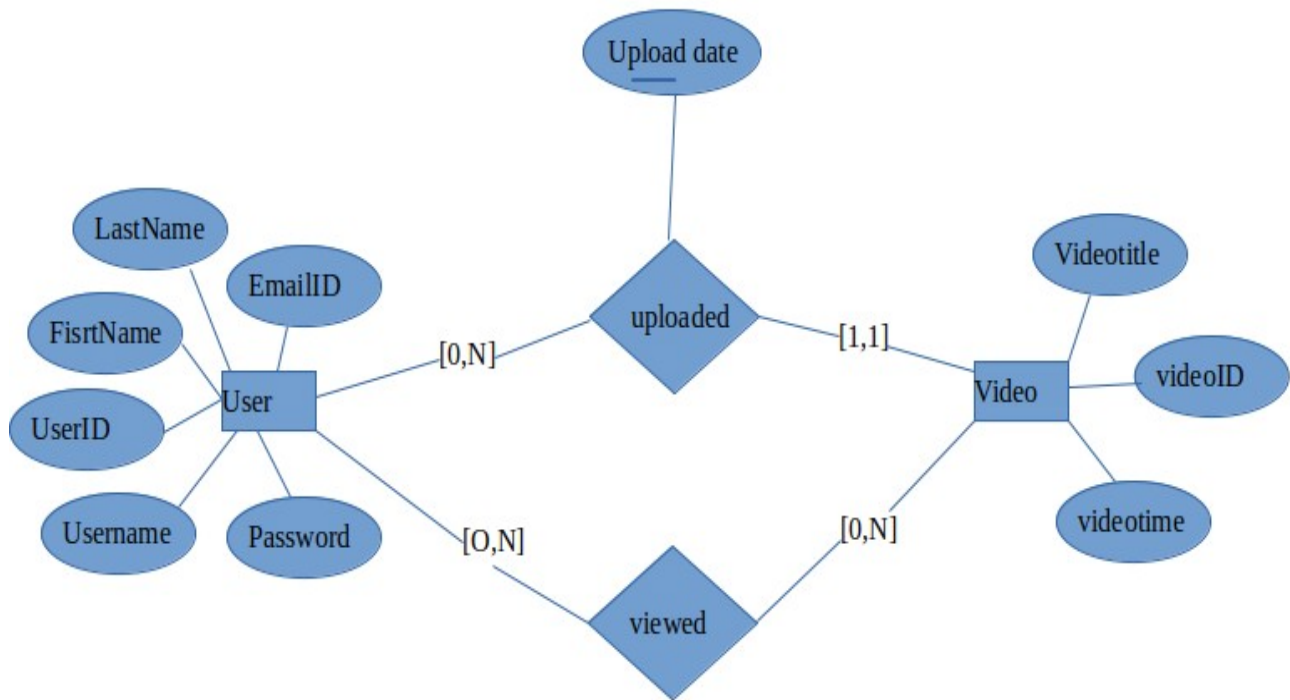
An ER model is typically implemented as a database. In a simple relational database implementation, each row of a table represents one instance of an entity type, and each field in a table represents an attribute type. In a relational database a relationship between entities is implemented by storing the primary key of one entity as a pointer or "foreign key" in the table of another entity.

There is a tradition for ER/data models to be built at two or three levels of abstraction. Note that the conceptual-logical-physical hierarchy below is used in other kinds of specification, and is different from the three schema approach to software engineering.

### Entities and Relationships in our App

#### *Entity Types*

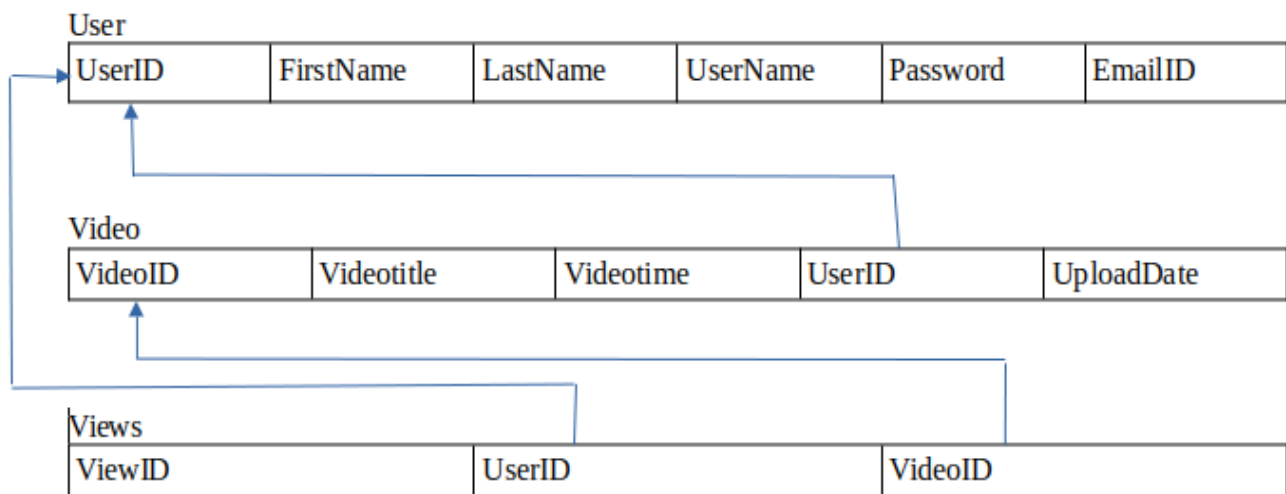
1. User : Name (first name, last name), User ID, Username, Password, Email ID
2. Video : Video ID, Video Title, Video Time, Upload Date



### Relationship Types

1. Uploaded : Upload Date/Time : Binary 1:N relationship between User and Video
2. Viewed : Viewed Date/Time : Binary M:N relationship between User and Video

### Conversion of ER Diagram to Relational schema



# Normalisation

## Introduction

Normalization is the process of organizing the data in the database. It is used to minimize the redundancy from a relation or set of relations. Normalization divides the larger table into the smaller table and links them using relationship. The normal form is used to reduce redundancy from the database table.

## Functional Dependency in the Youtube Database

- User ID  $\rightarrow$  Full Name, Username, Password, Email ID
- Video ID  $\rightarrow$  Video Title, Video Description, Video Path, User ID, Upload Date
- View ID  $\rightarrow$  User ID, Video ID

## Normal forms of the Database

The following can be inferred from the above relational schema and its functional dependencies:

- All attribute values in all relations are atomic. So the relations are in First Normal Form.
- Since the keys of all the relations are single attributes, there are not partial functional dependencies in any of the relations. So the relations are in Second Normal Form.
- There are no non-prime attributes that are transitively dependent on the key in any relation. So the relations are in Third Normal Form.
- In every functional dependency  $X \rightarrow A$  in the relation schema,  $X$  is a superkey of the respective relation. So the relations are in Boyce-Codd Normal Form.

## User Authentication and Security Measures

Security is a paramount concern for the website clone. The following security measures have been implemented:

1. **Password Hashing:** User passwords are securely hashed before storage to prevent unauthorized access.
2. **CSRF Protection:** Cross-Site Request Forgery protection is applied to prevent malicious actions.
3. **Input Validation:** User inputs are validated and sanitized to prevent SQL injection and other forms of attacks.

4. **User Permissions:** Only authorized users are allowed to perform actions like uploading videos and posting comments.

5. **User Registration:** New users can create accounts by providing a username, email, and secure password.

6. **Login and Logout:** Registered users can securely log in and out of their account.

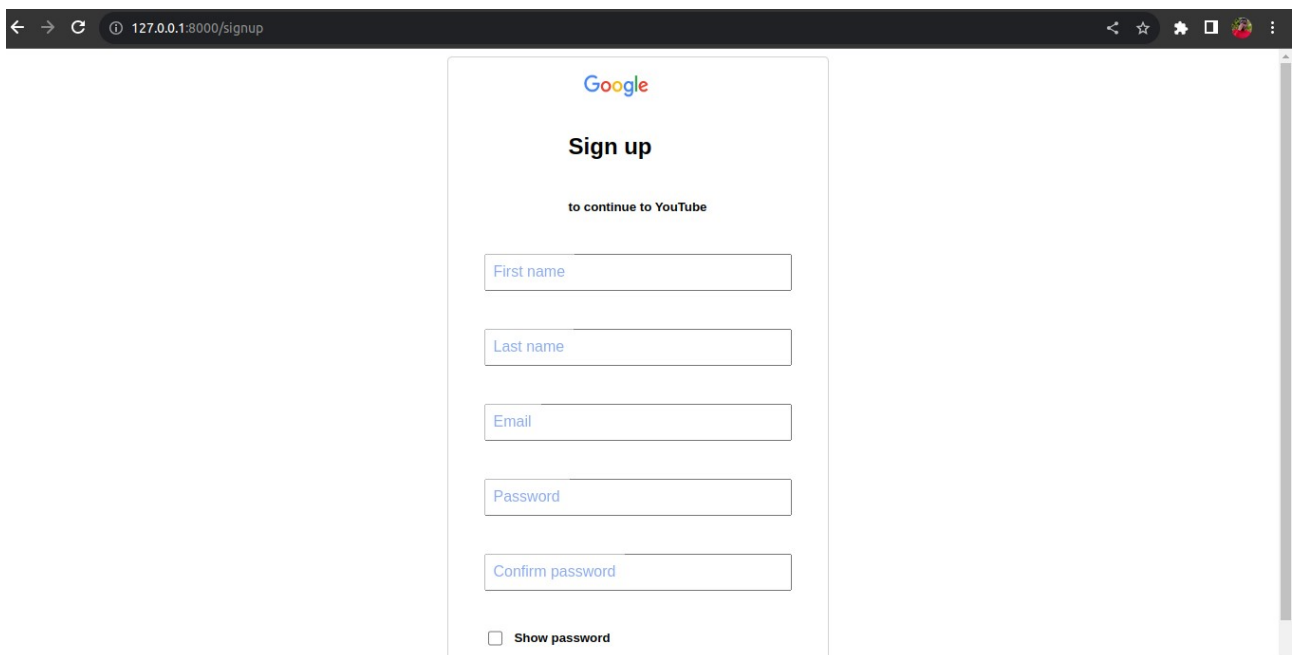
## Features Implemented and Screenshots

### Sign-Up Page

The sign-up page allows new users to create accounts and join the platform. The following features have been implemented:

1. **Registration Form:** A user-friendly form prompts users to provide essential details like username, email, and a secure password.
2. **Password Strength Indicator:** During password input, a visual indicator helps users create stronger passwords.
3. **Email Verification:** Users receive a verification email to confirm their email address, enhancing account security.
4. **Error Handling:** If the provided information is invalid, users receive clear error messages guiding them to correct the inputs.
5. **Terms of Service:** Users must agree to the terms of service before proceeding with the registration process.

Below is the screenshot:



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/signup". The main content area features a sign-up form with the Google logo at the top, followed by the text "Sign up" and "to continue to YouTube". The form contains five input fields: "First name", "Last name", "Email", "Password", and "Confirm password". Below these fields is a checkbox labeled "Show password".

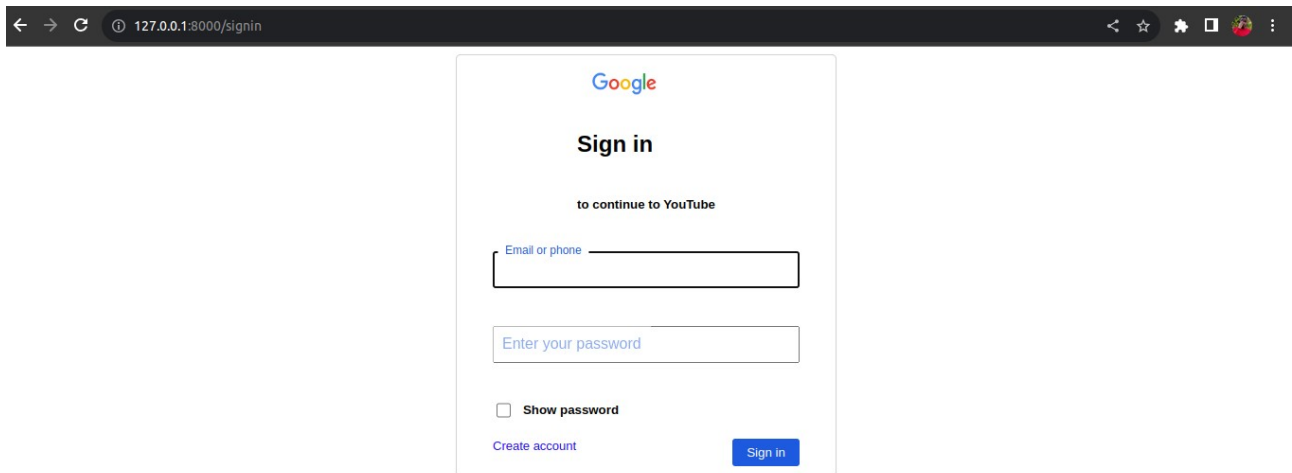
### Sign-In Page



The sign-in page enables existing users to access their accounts securely. The implemented features are as follows:

- 1.Login Form: Users enter their credentials (username and password) via a secure login form.
- 2.Forgot Password: A link to reset a forgotten password is provided for users who encounter login issues.
- 3.Error Handling: Clear error messages are displayed if incorrect credentials are entered, helping users troubleshoot.

Below is the screenshot:

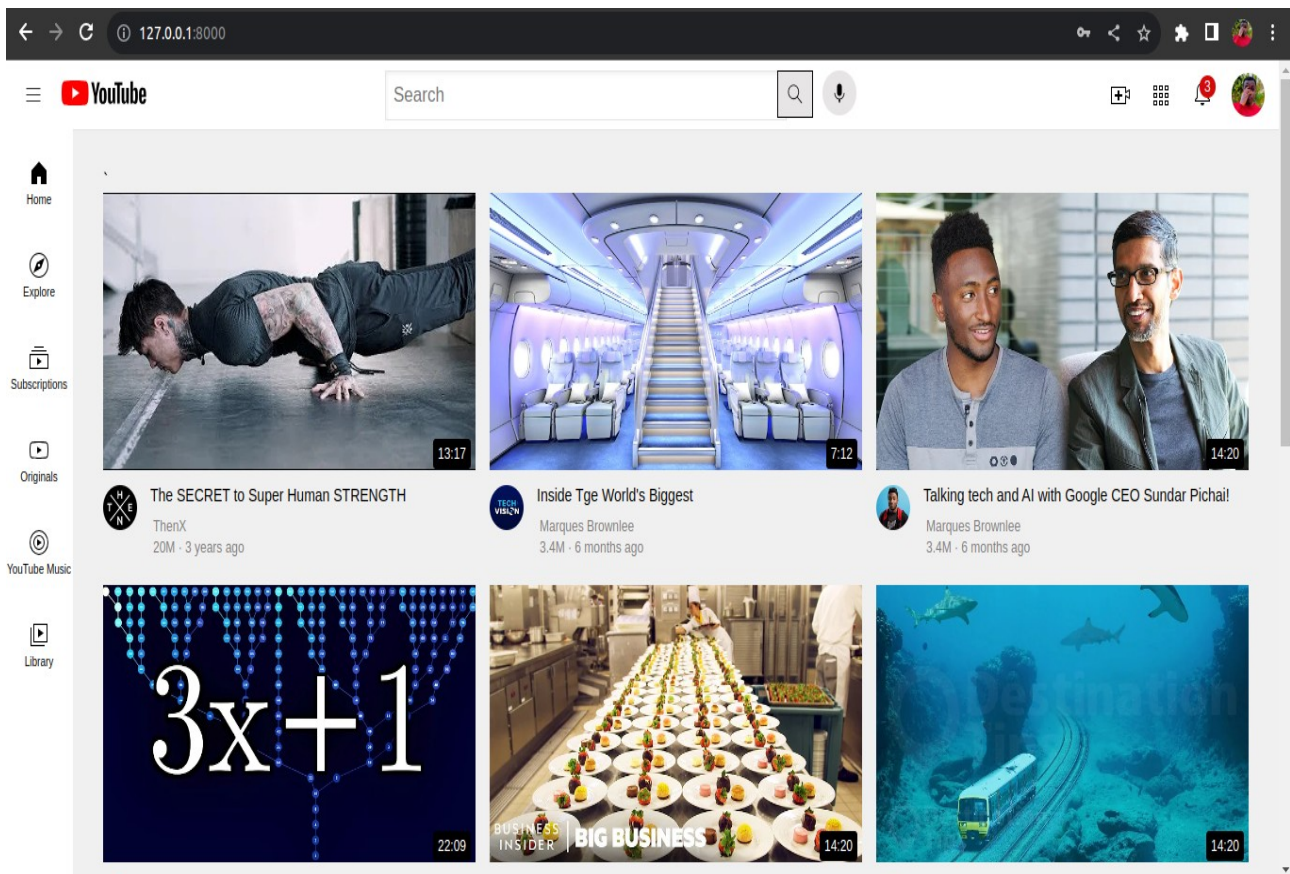


## Home Page

The home page of the YouTube clone replicates the layout and functionality of the original YouTube website. It includes the following features:

1. Featured Videos: Display a selection of featured videos on the top of the page.
2. Trending Videos: Show a list of currently trending videos.
3. Search Bar: Allows users to search for videos using keywords.
4. Video Thumbnails: Display video thumbnails, titles, and basic information.

Below is the screenshot:

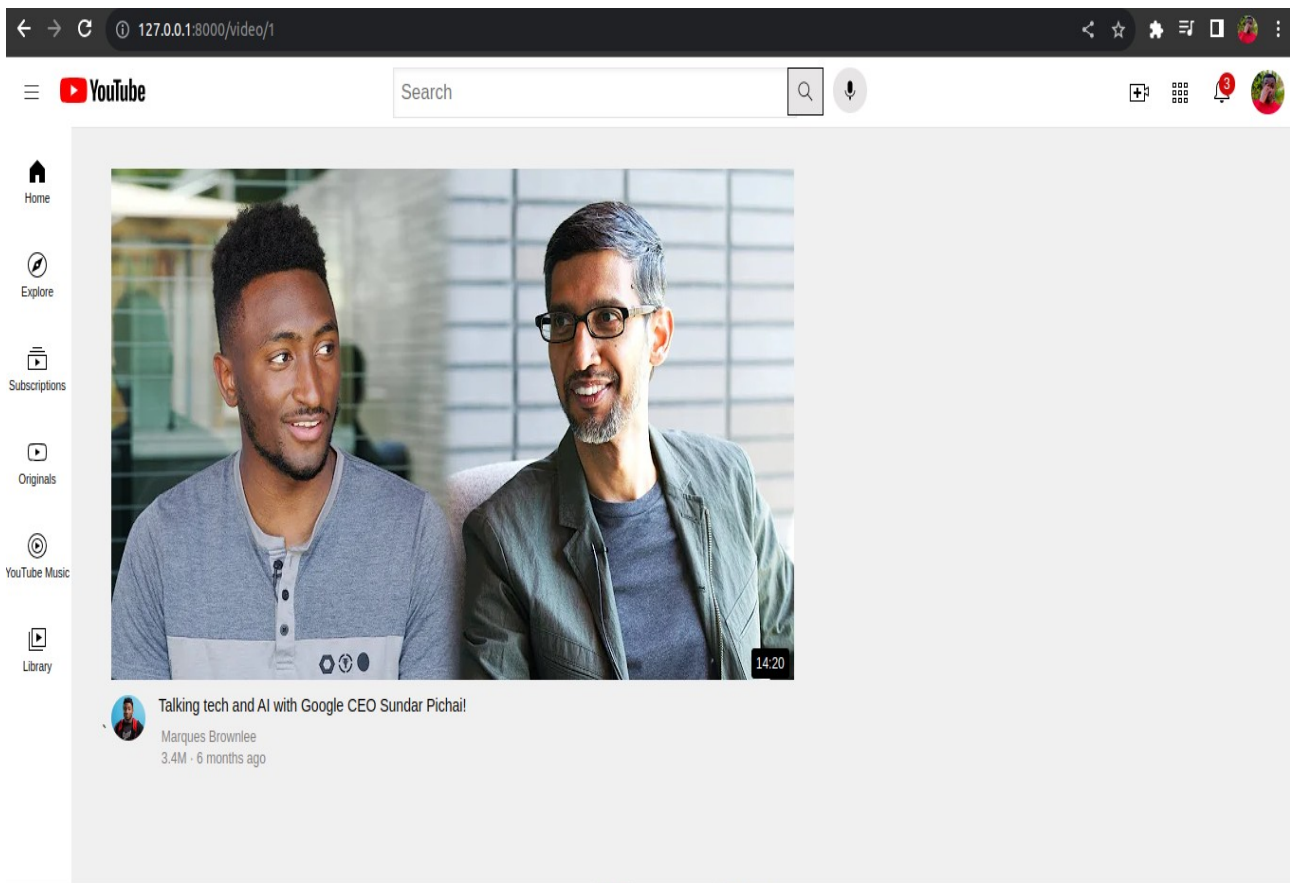


## Video Detail Page

The video detail page provides users with detailed information about a specific video. It includes the following features:

1. Video Player: Embedded video player to play the selected video.
2. Video Title and Description: Display the title and description of the video.
3. Views and Likes: Show the number of views and likes for the video.
4. Comments Section: Allow users to read and post comments on the video.
5. Related Videos: Display a list of related videos based on user preferences.

Below is the screenshot:



## Conclusion and References

### Conclusion

Designing and implementing a large video streaming service from the ground up teaches a lot of things and this project definitely helped us understand the various parts. Also, dealing with complex database designs and the unique and innovative normalisation techniques one needs to come up with, to ensure low latency was challenging and thought provoking.

### References

1. Django Documentation
2. Youtube Original scalability talk
3. System Design lessons from Youtube
4. Designing Youtube