National Institute of Technology, Karnataka
Department of Computer Science
CS254 Database System Lab

Video Streaming App

K Krishna Swaroop (181CO125)
Niranjan S Yadiyala (181CO136)

18-06-2020

Contents

1. 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!

2. 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

2.1 Existing system

2.1.1 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

2.1.2 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.

2.2 Proposed system

2.2.1 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 upload/delete videos in the system when they log in.
5. Users must be able to add comments to videos in the system when they log in.
6. Users must be able to watch videos in the system when they log in or logout.
7. Users must be able to search for videos/users/groups when they log in or logout.
8. Users must be able to create channels.
9. Users must be able to like/dislike videos.
10. Users can like or dislike the videos, under this condition, the system should keep numbers of likes, dislikes, comments, views to present these numbers to users.
11. Users can check their watch history
12. Users can see the trending videos based on the number of views
13. Users can subscribe to channels and have a separate page that has videos uploaded in channels of their subscription
14. Users can contact the developers for bug reports

2.2.2 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
- **MySQL** - Database used for the application. We have used the same database system, as used in the original Youtube Architecture
- **Frontend** - HTML, CSS and JS were used for fast templating

3. Entity Relationship Diagram

3.1 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.

3.2 Entities and Relationships in our App

3.2.1 Entity Types

1. **User** : Name (first name, last name), User ID, Username, Password, Email ID
2. **Video** : Video ID, Video Title, Video Description, Video Path

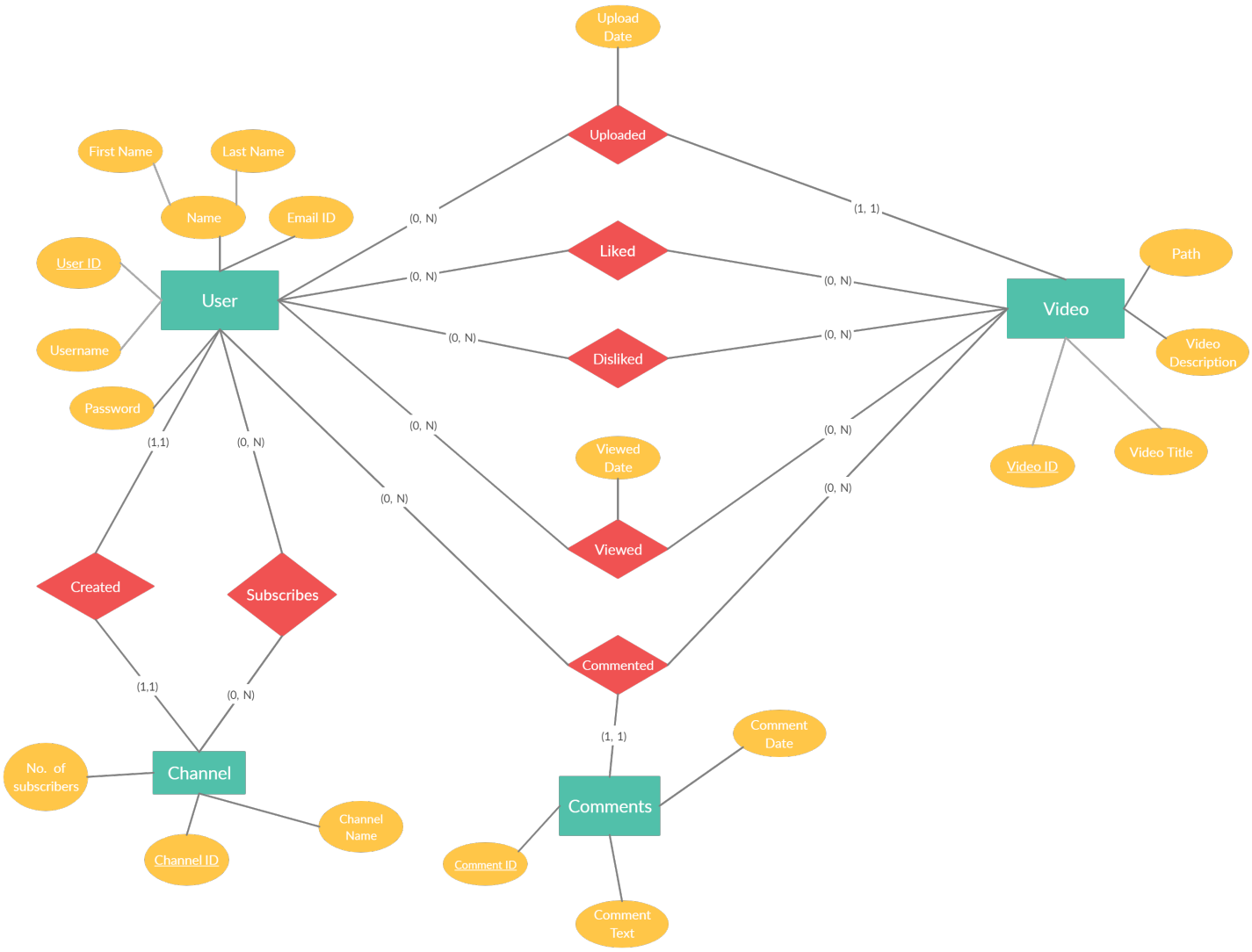


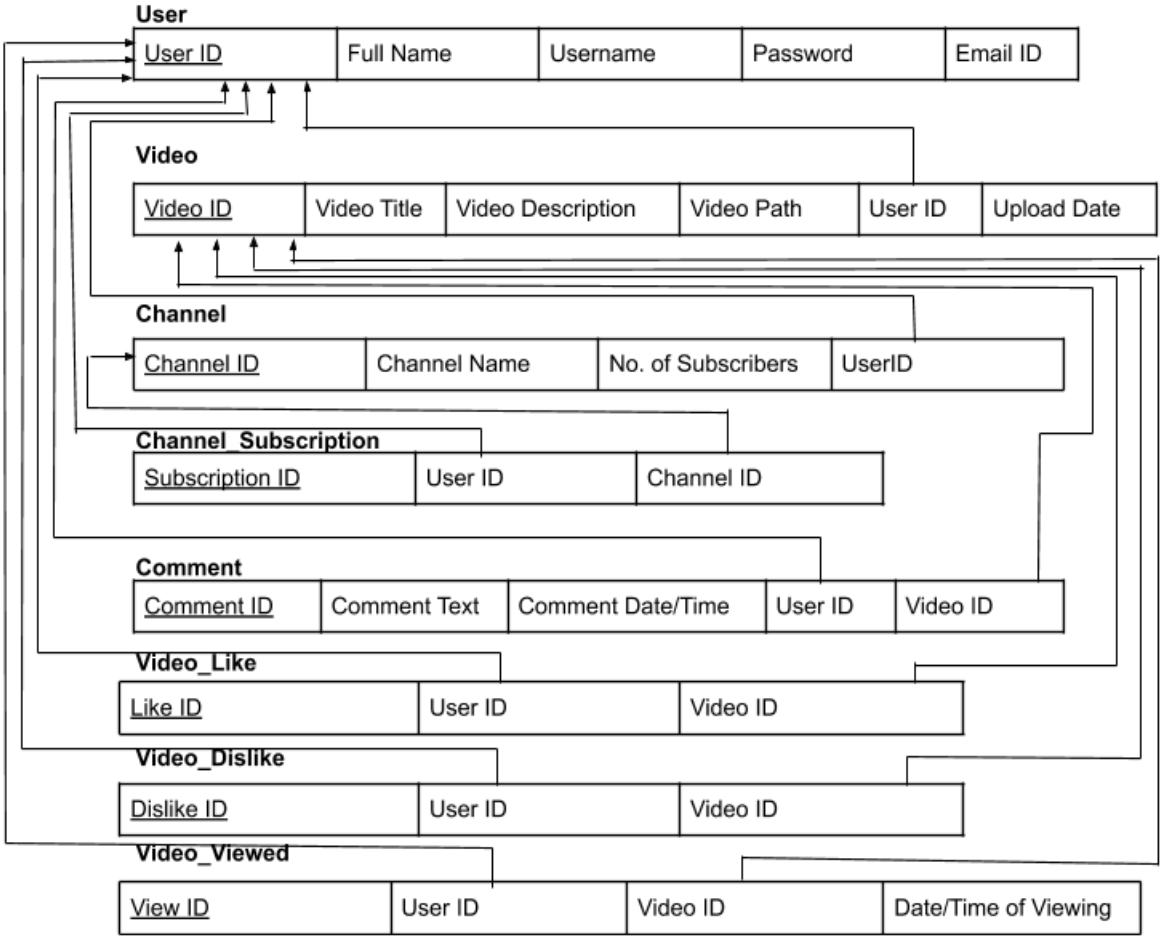
Figure 3.1: Entity Relation diagram for our streaming service

3. **Channel** : Channel ID, Channel Name, Number of Subscribers
4. **Comment** : Comment ID, Comment Text, Comment Date/Time

3.2.2 Relationship Types

1. **Uploaded** : Upload Date/Time : Binary 1:N relationship between **User** and **Video**
2. **Liked** : Binary M:N relationship between **User** and **Video**
3. **Disliked** : Binary M:N relationship between **User** and **Video**
4. **Viewed** : Viewed Date/Time : Binary M:N relationship between **User** and **Video**
5. **Commented** : Ternary relationship between **Comment**, **User** and **Video**
6. **Created** : Binary 1:1 relationship between **User** and **Channel**
7. **Subscribes** : Binary M:N relationship between **User** and **Channel**

3.3 Conversion of ER Diagram to Relational schema



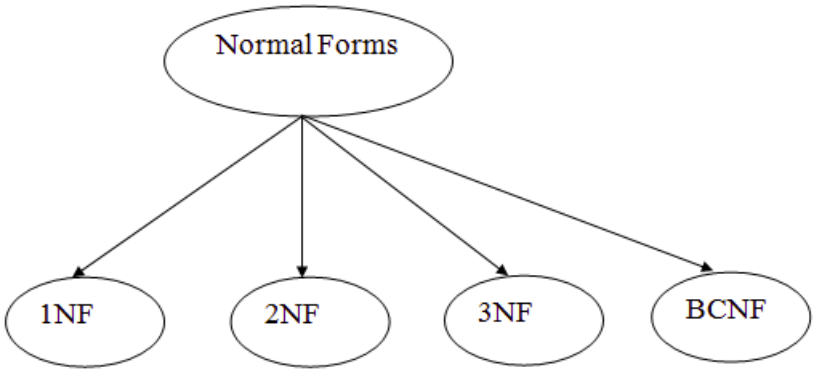
4. Normalisation

4.1 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.

4.2 Types and their description

There are four types of Normal Forms



Their descriptions are provided below

Normal Form	Description
1NF	A relation is in 1NF if it contains an atomic value.
2NF	A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transition dependency exists.
4NF	A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency.
5NF	A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless.

4.3 Functional Dependency in the Youtube Database

- User ID –> Full Name, Username, Password, Email ID
- Video ID –> Video Title, Video Description, Video Path, User ID, Upload Date
- Channel ID –> Channel Name, Number of Subscribers, User ID
- Comment ID –> Comment Text, User ID, Video ID, Comment Date/Time
- Subscription ID –> User ID, Channel ID
- Like ID –> User ID, Video ID
- Dislike ID –> User ID, Video ID
- View ID –> User ID, Video ID

4.4 Normal forms of the Database

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

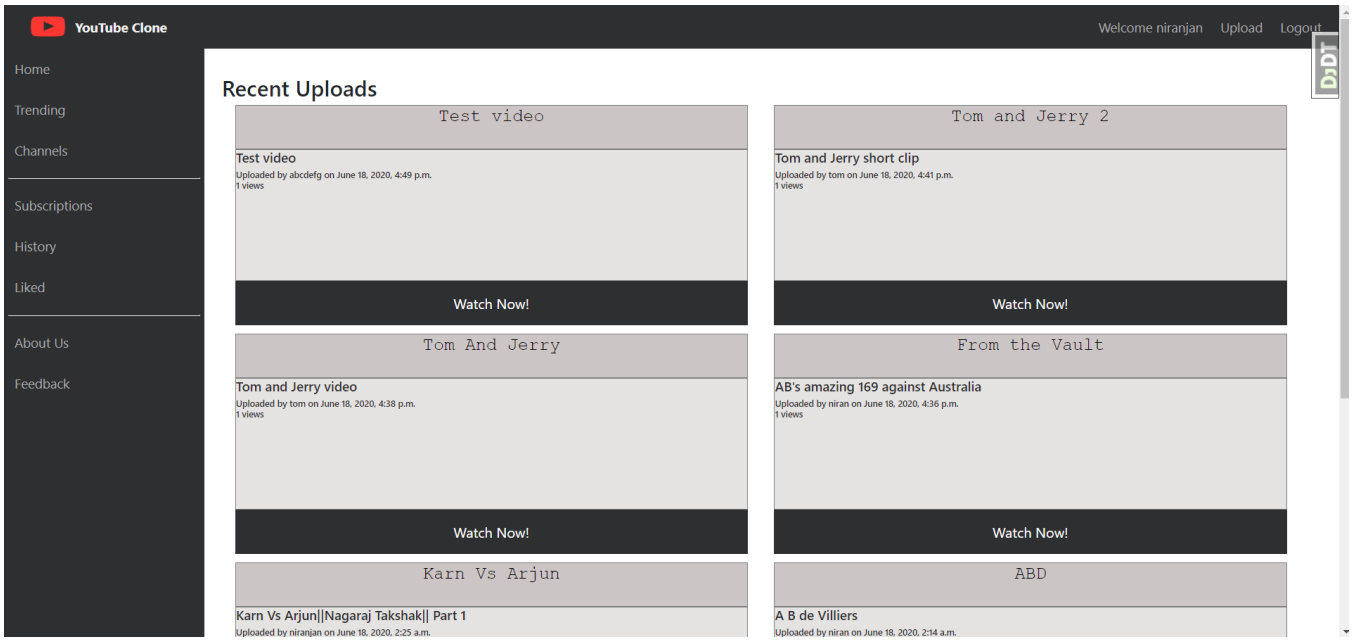
- All attribute values in all relations are atomic. So the relations are in First Normal Form.
- Since the keys of all the relations are single attributes, there are not partial functional dependencies in any of the relations. So the relations are in Second Normal Form.
- There are no non-prime attributes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

5. Implementation and Screenshots

Here is the link to the github repository that contains our project folder : Project Repo

This is a YouTube video link, where the woking and the functionalities of this application are explained : Video Link

Below, we show the actual working screenshots of the app



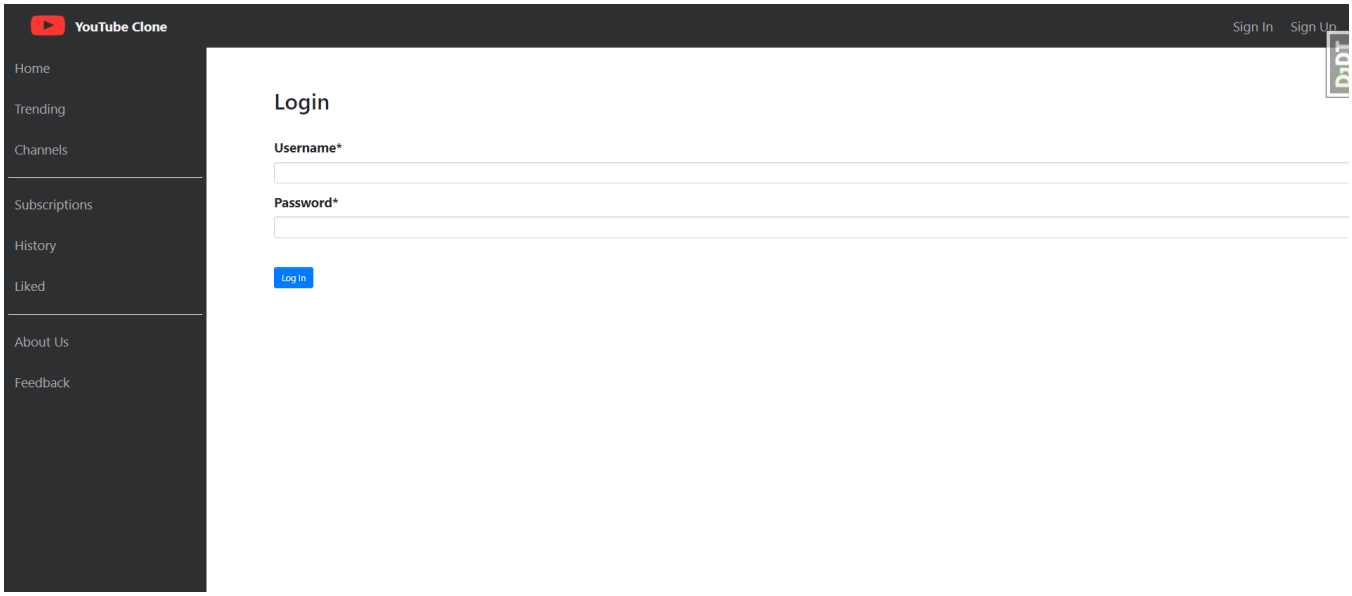


Figure 5.1: Login View

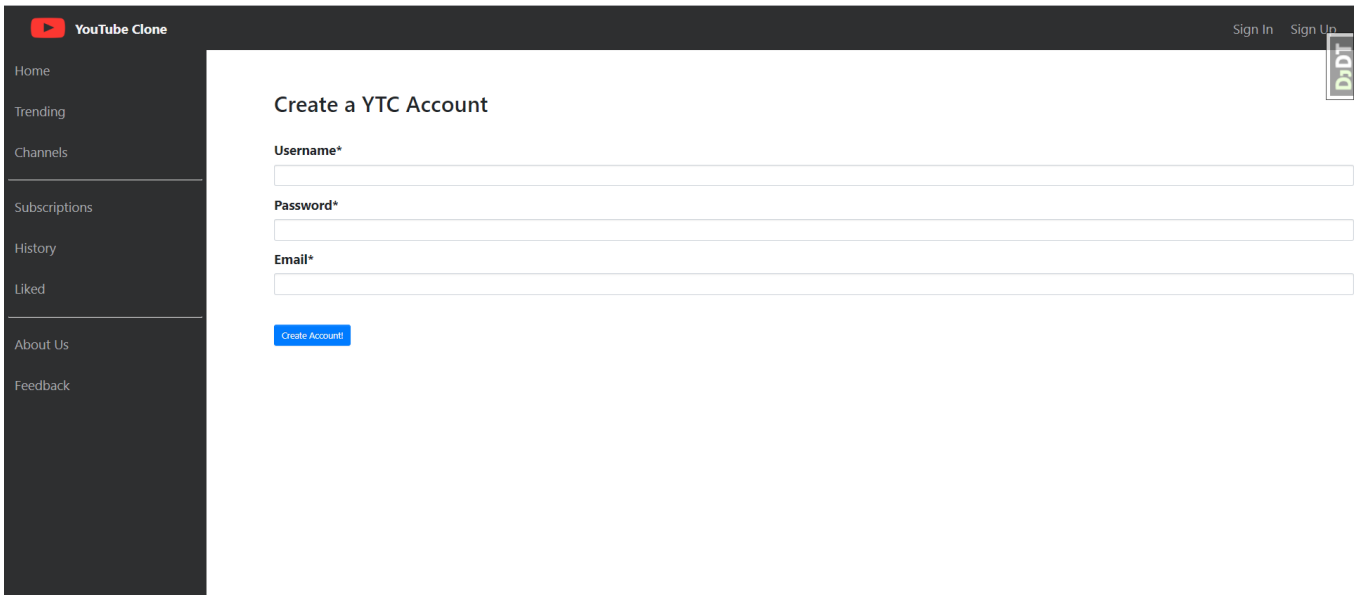


Figure 5.2: Sign Up View

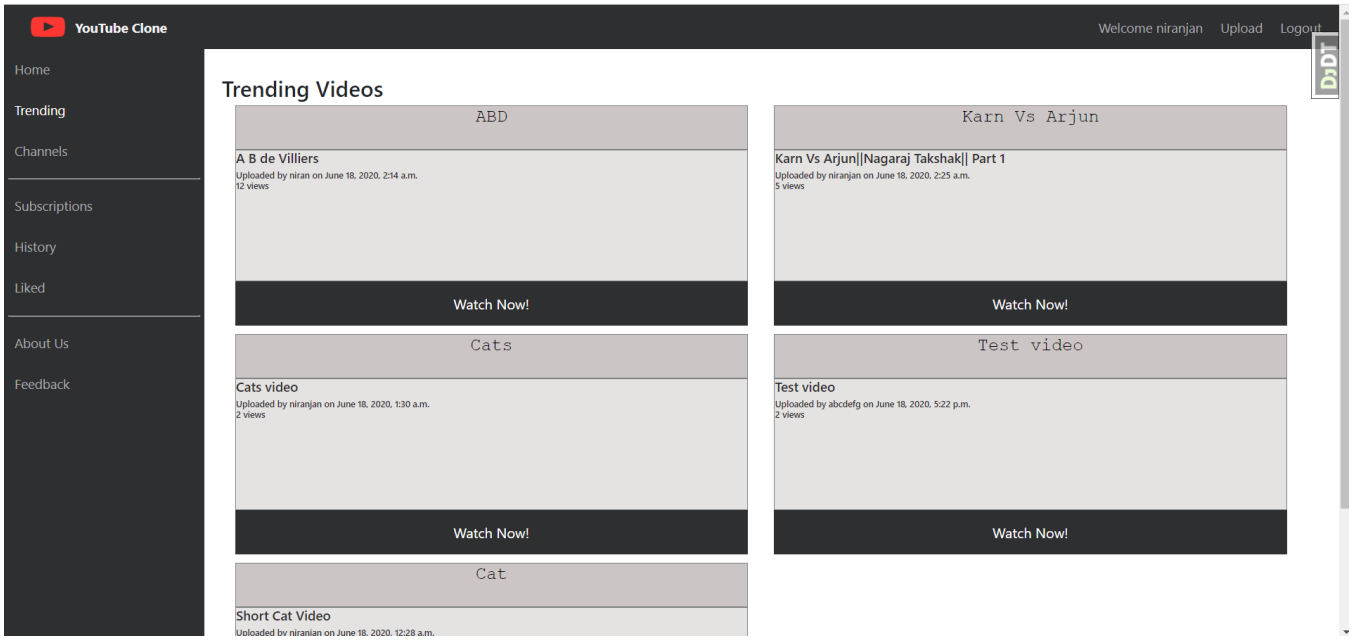


Figure 5.3: Trending Videos Section (based on the number of views)

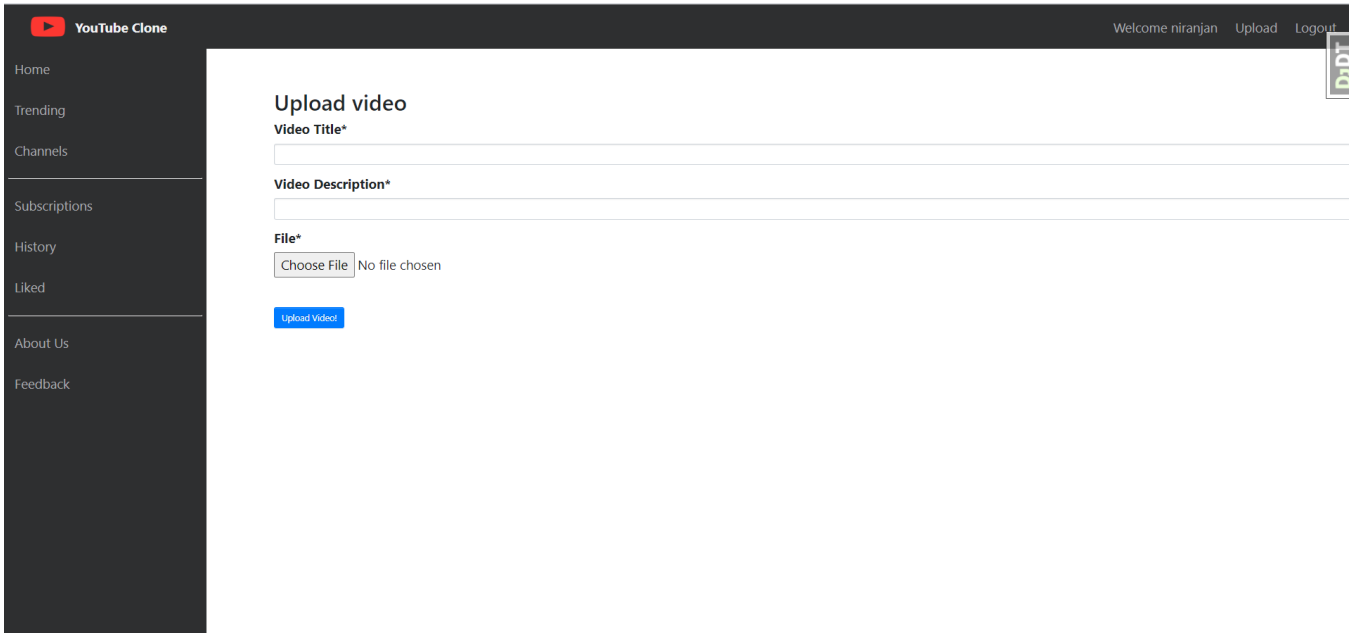


Figure 5.4: Upload your video

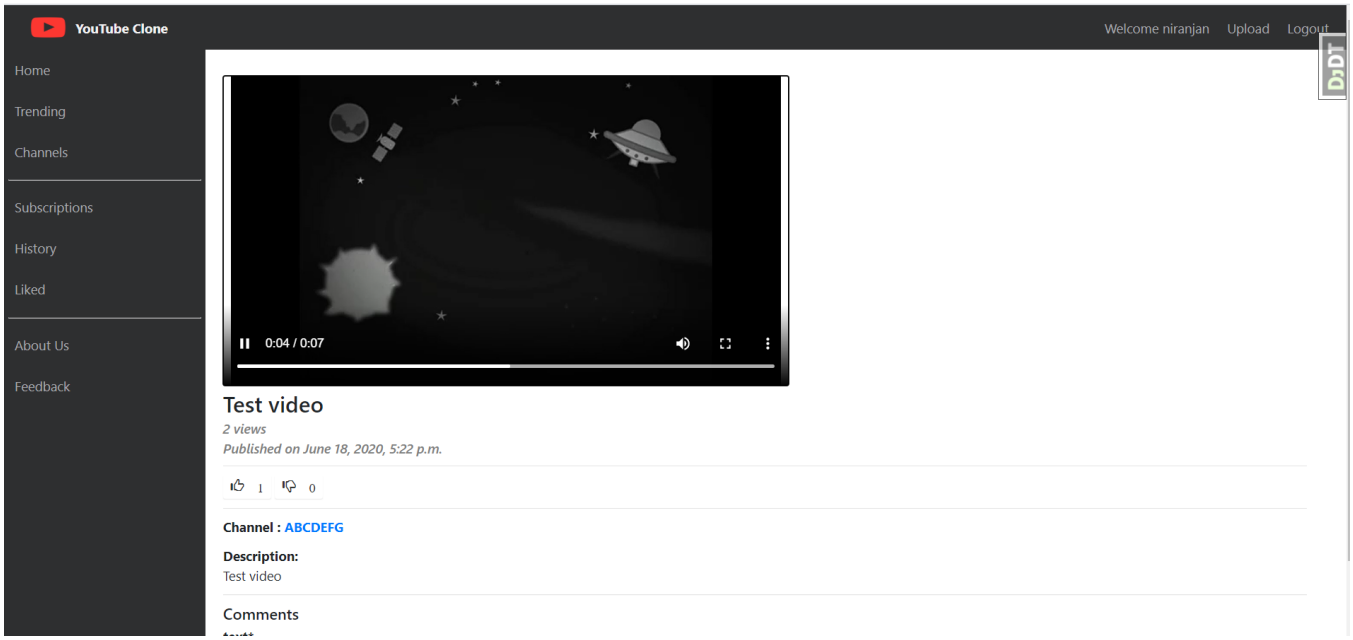


Figure 5.5: Video Playback

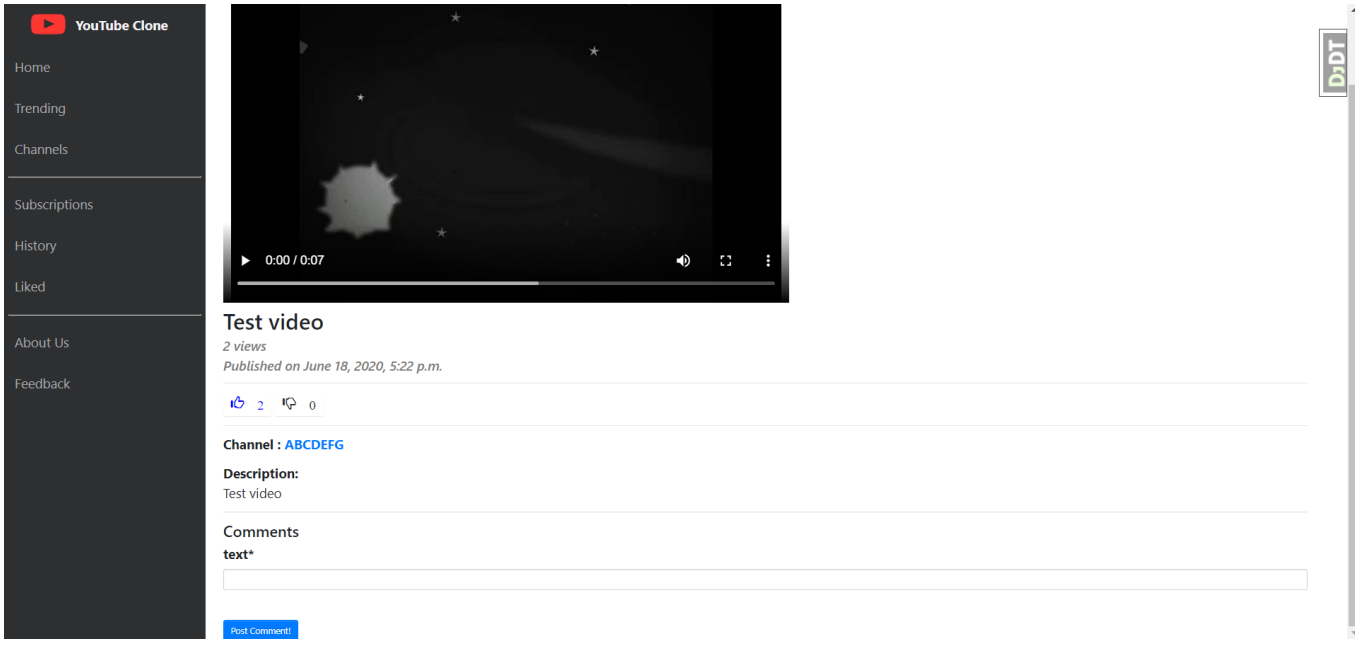


Figure 5.6: Like, Comment on your favourite video

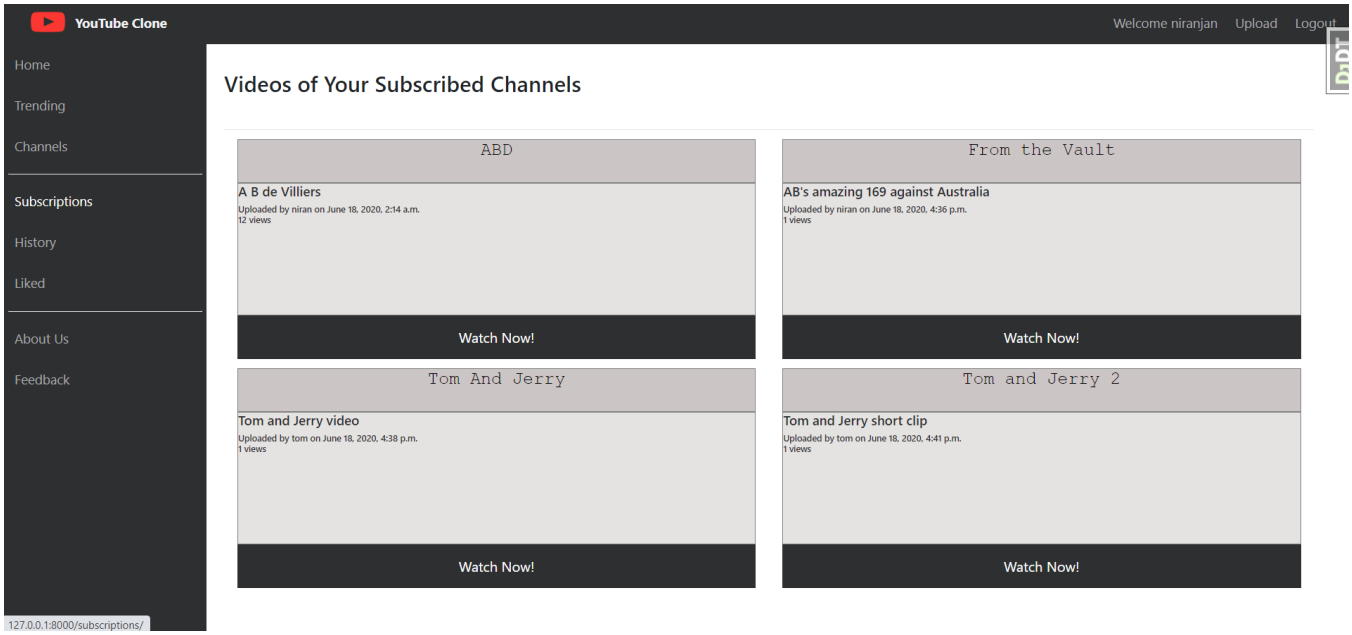


Figure 5.7: Videos you subscribed to

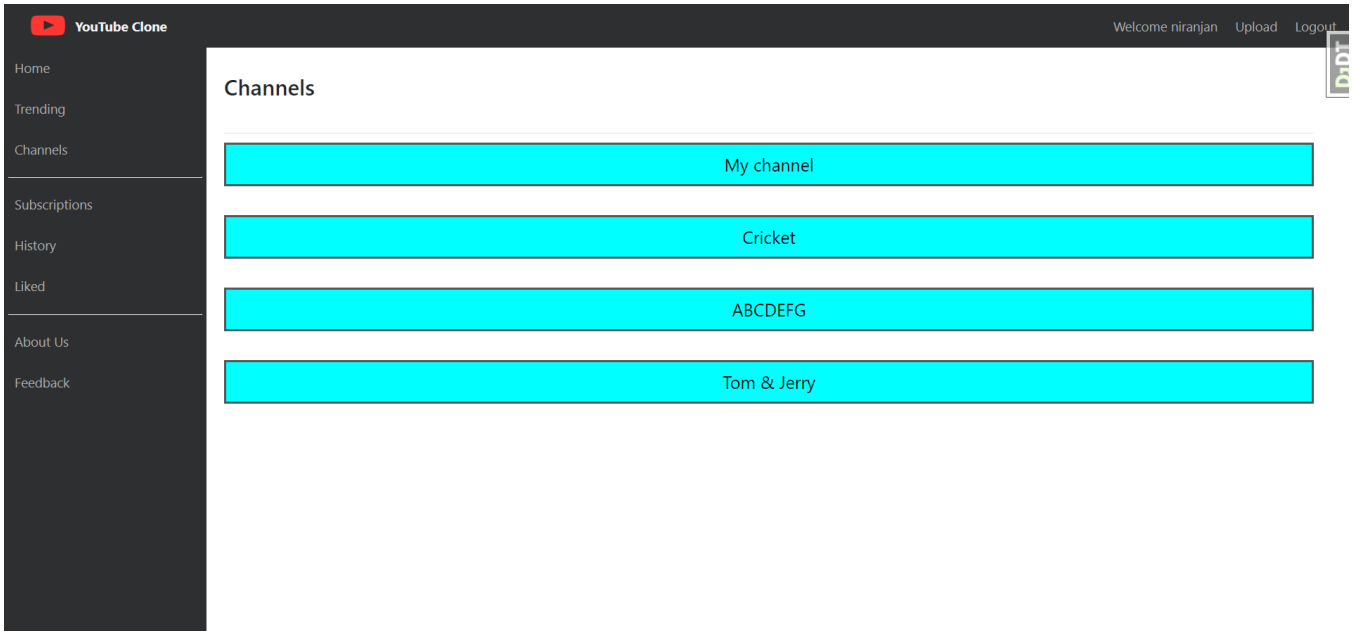


Figure 5.8: List of Channels

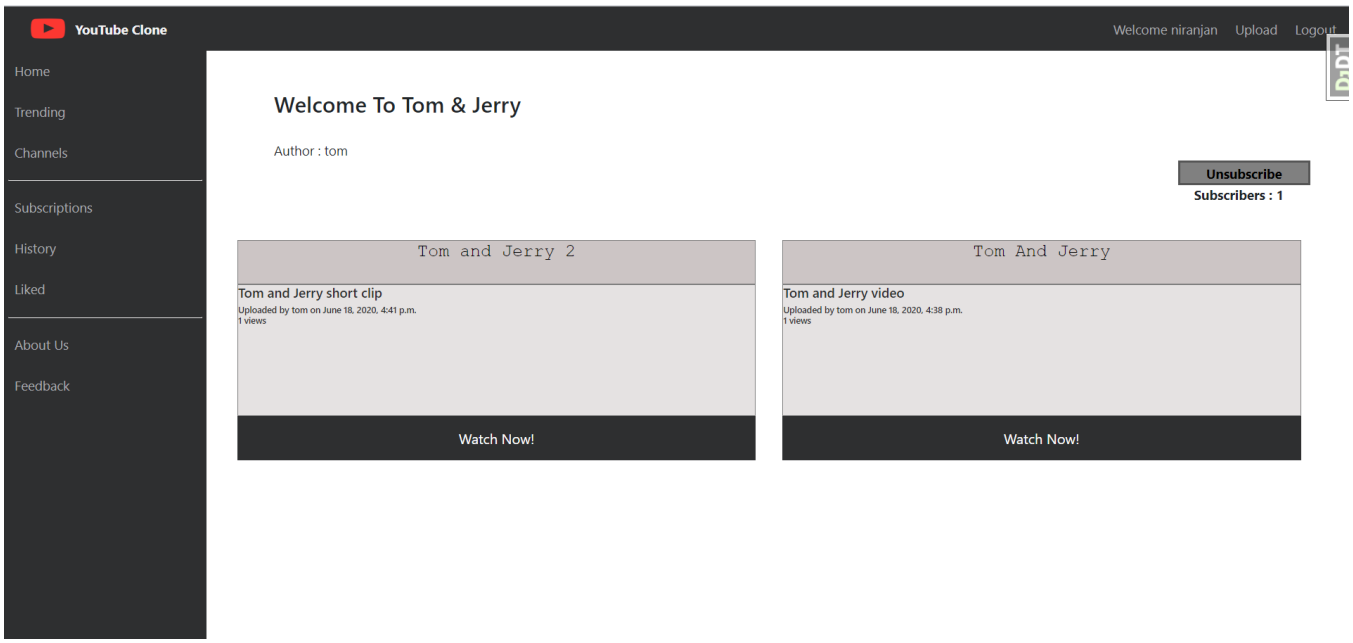


Figure 5.9: Channel View

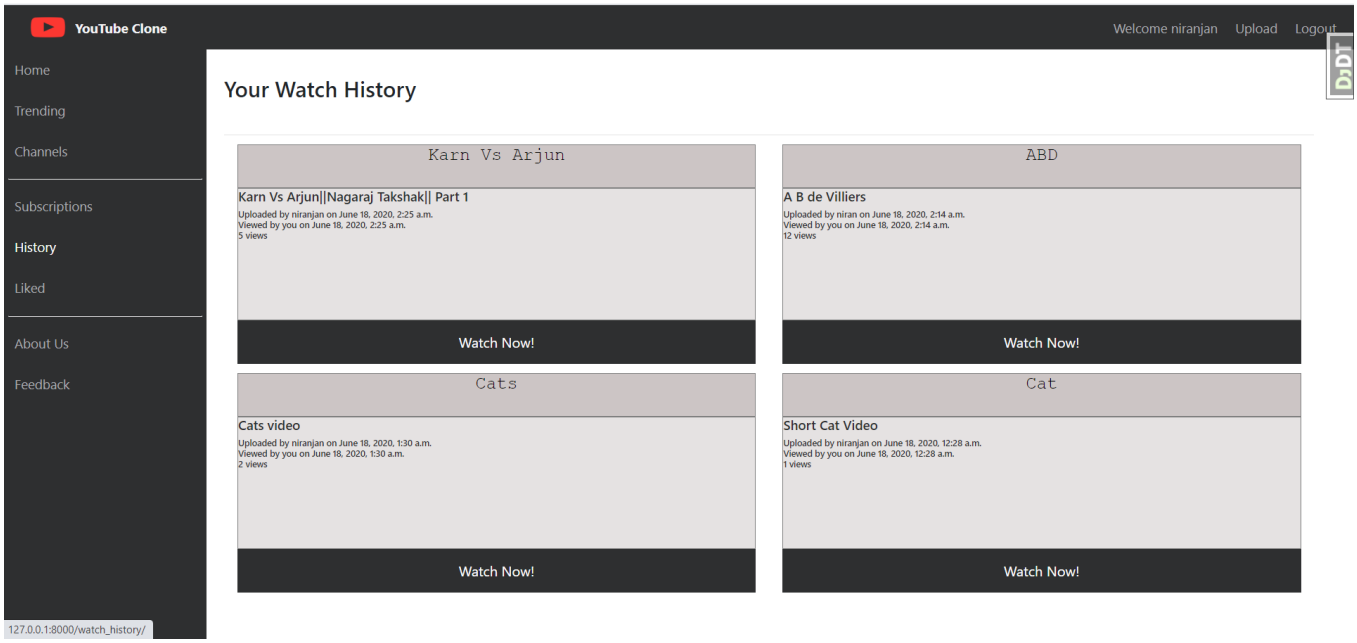


Figure 5.10: Your watch history

6. Conclusion and References

6.1 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.

6.2 References

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