Online Video Streaming Application

Gayathri Devi, Vinuta I, Harshini A, Dheekshna L S

ABSTRACT

The YouTube clone project is a web application that aims to replicate the functionality of the popular video-sharing platform, YouTube. The application will allow users to upload, watch, and share videos, as well as interact with each other through comments, ratings, subscriptions, and notifications. The application will be built using a combination of front-end and back-end technologies, including HTML, CSS, JavaScript, ReactJS, Rapid APIs. User authentication and authorization will be implemented using OAuth or JWT tokens and role-based access control or attribute-based access control. Video upload and management will be handled using cloud storage services and video transcoding services. Video playback and streaming will be implemented using HTML5 video player and streaming protocols like HLS or DASH. A search functionality, comments and rating system, subscription and notification system, and analytics and insights will also be included. Overall, the YouTube clone project is a complex and challenging project that requires a skilled team of developers and designers to build a robust and scalable application.

CHAPTER 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, Video 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 Video and YouTube Premium, subscription services respectively offering premium and ad-free Video 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.

1.1 PROJECT AIMS AND OBJECTIVES

The primary aim of the proposed project is to develop a robust and user-friendly online video streaming application that offers users access to a vast library of video content from various genres, including movies, TV shows, documentaries, and original series. To achieve this aim, the project has the following objectives:

1. Design and develop a scalable and secure video streaming platform using modern web technologies such as React, Node.js, and Rapid APIs.

2. Offer users fast and reliable video streaming with high-quality video, utilizing the latest streaming technologies to ensure smooth playback.

3. Provide users with an intuitive and user-friendly interface, allowing them to easily search and navigate the video library, create playlists, and manage their viewing history.

4. Develop a recommendation system that suggests videos to users based on their viewing history and preferences, providing a personalized streaming experience.

5. The application will also offer no ad-interruptions as well as free content to ensure accessibility to all users.

6. Ensure data privacy and security, implementing robust measures to protect user data and prevent unauthorized access.

7. Continuously improve and update the application based on user feedback, adding new features and functionalities to enhance the user experience.

By achieving these objectives, the project aims to provide users with a seamless and enjoyable video streaming experience, offering a wide range of content that caters to their interests and preferences while prioritizing their privacy and security.

1.2 BACKGROUND OF THE PROJECT

The background of the proposed project lies in the growing popularity of online video streaming services, which have transformed the way people consume video content. With the advent of high-speed internet and the widespread availability of affordable streaming devices, more and more people are turning to online video streaming platforms to watch their favorite movies, TV shows, and original series.

However, despite the availability of numerous online video streaming services, many users still face challenges in finding a platform that offers a wide range of content, fast and reliable streaming, and user-friendly features. Additionally, concerns over data privacy and security have become increasingly important, with users demanding greater control over their personal data and the content they consume.

In this context, the proposed project aims to address these challenges by developing an online video streaming application that offers users access to a vast library of video content from various genres, with an emphasis on fast and reliable streaming, high-quality video, and user control. The application will be designed using modern web technologies and will prioritize user privacy and security, providing a seamless and enjoyable streaming experience for all users.

1.3 OPERATION ENVIRONMENT

The proposed online video streaming application will operate within a web-based environment and will be accessible through popular web browsers such as Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.

The application will be developed using modern web technologies such as React, Node.js, and MongoDB, and will be hosted on a server infrastructure that is designed to handle high traffic loads and provide fast and reliable streaming. The server infrastructure will be configured to optimize the delivery of video content, using content delivery networks (CDNs) and caching mechanisms to ensure smooth playback and reduce buffering times.

The application will also be designed to be responsive and mobile-friendly, allowing users to access the platform from their smartphones, tablets, and other mobile devices. The application will be compatible with both iOS and Android devices, and will be available for download from the respective app stores.
Additionally, the application will be designed to integrate with popular third-party platforms and services, such as social media platforms and payment gateways, to provide users with a seamless and integrated streaming experience.

CHAPTER 2
SYSTEM ANALYSIS

System analysis is an important phase in the software development lifecycle that involves understanding the requirements of the system and designing a solution that meets those requirements. In the context of the proposed online video streaming application, the system analysis phase involves the following steps:

1. Understanding the requirements: This involves gathering information on the functional and non-functional requirements of the system, including the features and functionalities that the application must offer, the user interface design, the performance and scalability requirements, and the security and privacy requirements.
2. Defining the system architecture: Based on the requirements gathered in the previous step, the system architecture is designed, which includes the components of the application, the technologies used, and the data flow between different components of the application.
3. Designing the database schema: The database schema is designed to store and manage the video content and user data, including user preferences, viewing history, and payment information.
4. Developing the application: The application is developed using modern web technologies such as React, Node.js, and MongoDB, implementing the features and functionalities defined in the requirements gathering phase.
5. Testing and debugging: The application is tested and debugged to ensure that it meets the functional and non-functional requirements, including performance, scalability, security, and privacy.
6. Deployment and maintenance: Once the application has been tested and debugged, it is deployed to a production environment and maintained, with regular updates and bug fixes to ensure its continued reliability and usability.

Overall, the system analysis phase of the proposed online video streaming application involves understanding the requirements of the system, designing a solution that meets those requirements, and implementing, testing, deploying, and maintaining the application to ensure its continued functionality and usability.

2.1 PRODUCT DESCRIPTION

Online video Streaming is an application developed which helps user to listen to Video without downloading. It reduces the risk of interruptions. This streaming app differs from downloads in that no copy of the Video is saved to your hard drive. If you want to hear it again, you can easily do so. This app provides many premium features online streaming, lyrics, recommendation system and top charts.

2.2 PROBLEM STATEMENT

A problem statement is a clear and concise description of an issue or challenge that needs to be addressed. It outlines the gap between the current state and the desired state, and describes the impact of the problem on individuals or organizations affected by it. A problem statement is typically used in research, business, or project management to define the scope and objectives of a project, and to identify the underlying problem that needs to be solved. A well-written problem statement is essential for effective problem-solving and decision-making, as it provides a clear direction for identifying and implementing solutions.

The problem occurred in existing application includes:

1.Interruptions

The existing video streaming applications contains lots of advertisements making the experience more interrupted.

2.Downloading videos

Users are forced to download songs in order to listen to
them which results in taking up large amount of space in their respective devices.

3.Difficult To Search videos

When there is no computerized system there is always a difficulty in searching for videos if the records are large in number.

4.Space Consuming

After downloading songs become large the space for physical storage of file.

5.Cost Consuming

Existing applications provide premium features for a specific cost which might not be affordable for everyone.

2.3 SYSTEM REQUIRENTS

1) NON-FUNCTIONAL REQUIREMENTS

1.Efficiency Requirement:

When a video streaming app is implemented user will easily access songs as searching and listening will be very faster.

2.Reliability Requirement:

The system should accurately performs genre choosing, top songs ,top artists and search songs.

3.Usability Requirement:

The system is designed for a user friendly environment so that user can perform the various tasks easily and in an effective way.

4.Implementation Requirements:

In implementing whole system it uses HTML in front endwith JavaScript as server side scripting language which will be used for database connectivity and the backend that is the database part is developed using Rapid API.

2) FUNCTIONAL REQUIREMENTS

1.Discover Genre

This feature used by the user to choose particular genre that interests them and songs are recommended according to their choice of genre.

2.Top List

This feature allows user to find the most popular songs and artists at one click, making it very reliable and faster.

3.Search

This feature provides the users ability to search songs, artists, albums and so more along with their recommendations and song details.

4.Around you feature

This feature allows you to discover songs that are popular at your country which provides an environment to listen.

CHAPTER 3

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

3.1 Existing system

3.1.1 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

3.1.2 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.

3.2 Proposed system

3.2.1 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

3.2.2 Tech Stack

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

- Operating System

  Windows 10 is used as the operating system as it is stableand supports more features and is more user friendly.
- JavaScript
- This project is completely developed through react jswhich enables advanced implementation of the application.
- Development Tools And Programming Language

HTML is used to write the frontend and develop webpageswith CSS.

CHAPTER 4
ENTITY RELATIONSHIP DIAGRAM

4.1 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. Notethat the conceptual-logical-physical hierarchy below is used in other kinds of specification, and is different from the three schema approach to software engineering.

4.2 Entities and Relationships in our App

4.2.1 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
3. Channel : Channel ID, Channel Name, Number of Subscribers
4. Comment : Comment ID, Comment Text, Comment Date/Time

4.2.2 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.

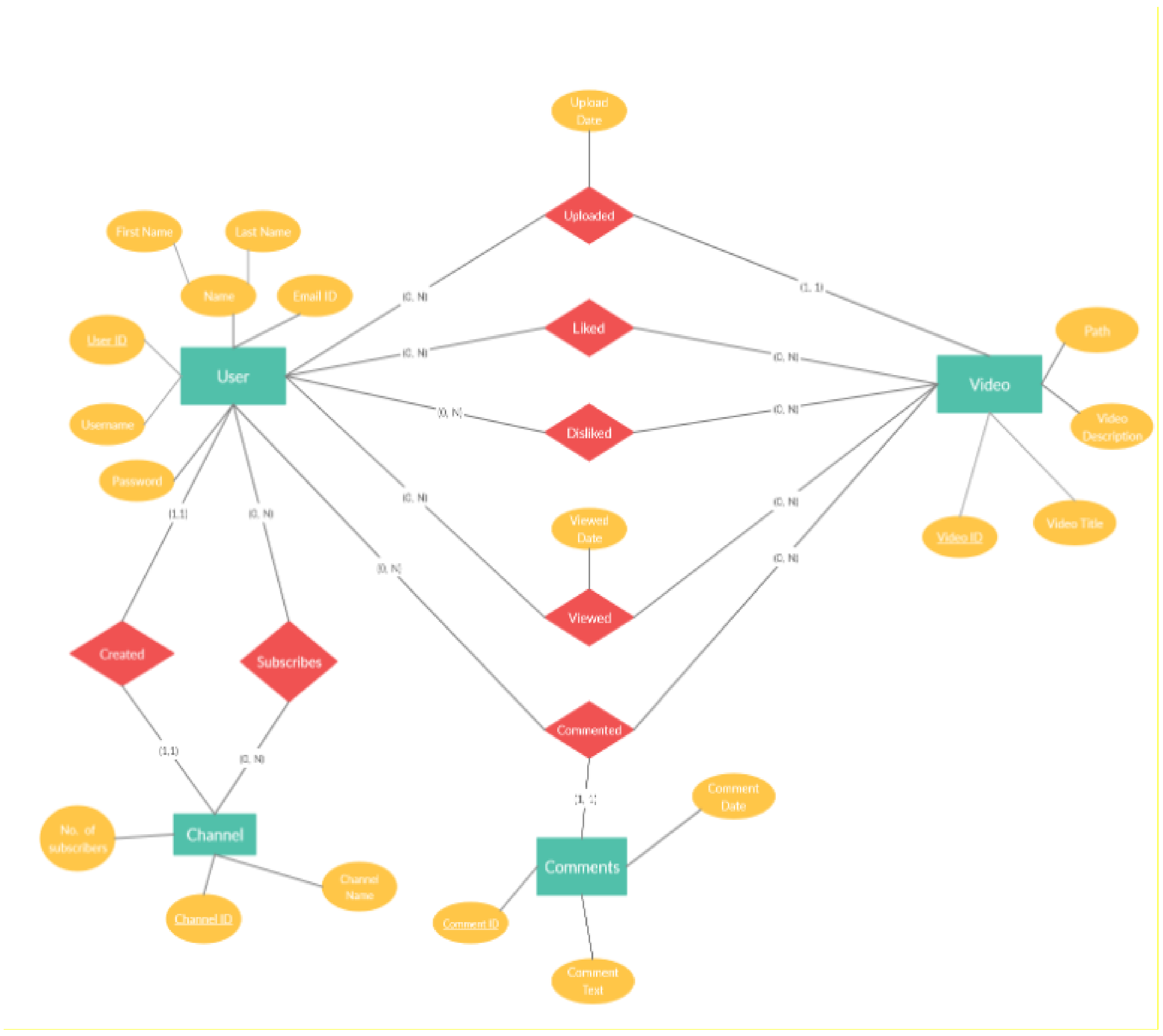

Figure 4.1: Entity Relationship diagram for youtube clone

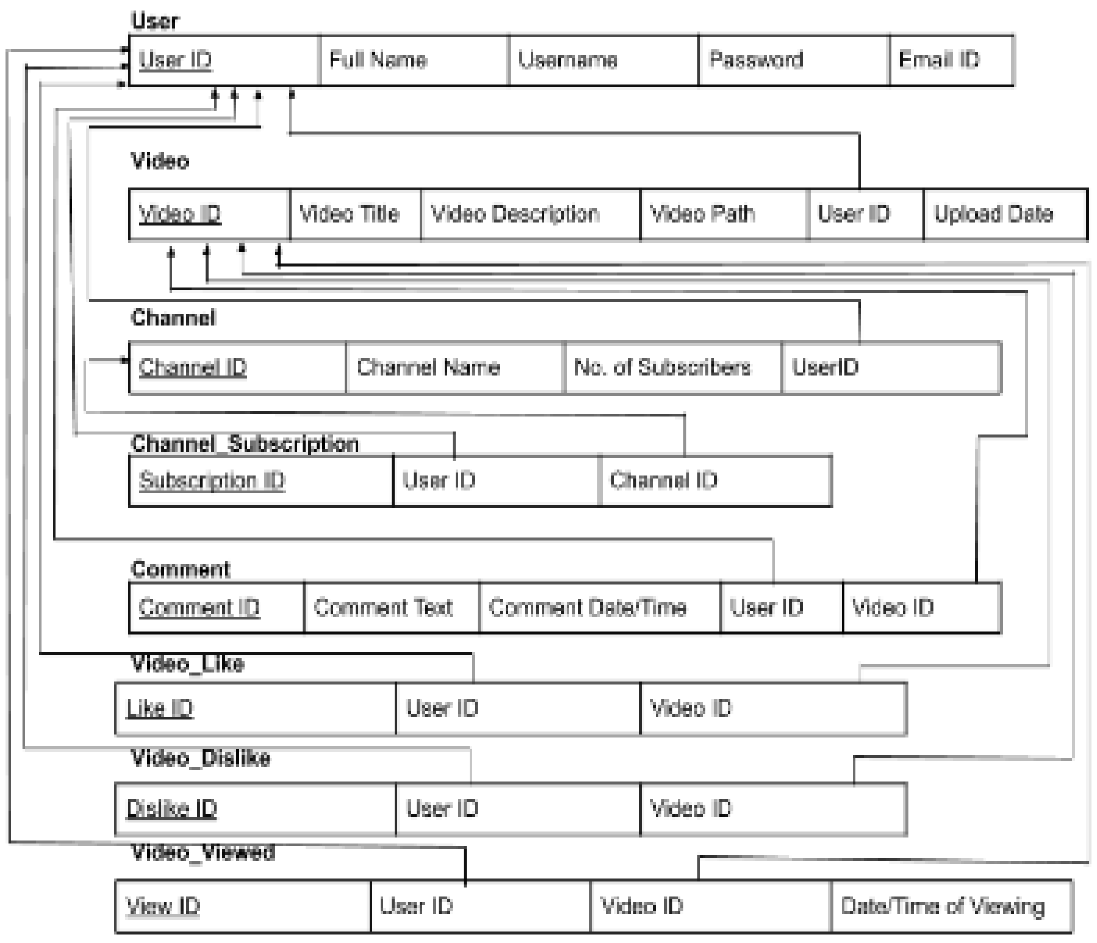

Figure 4.2: Conversion of ER diagram to Relational Scheme

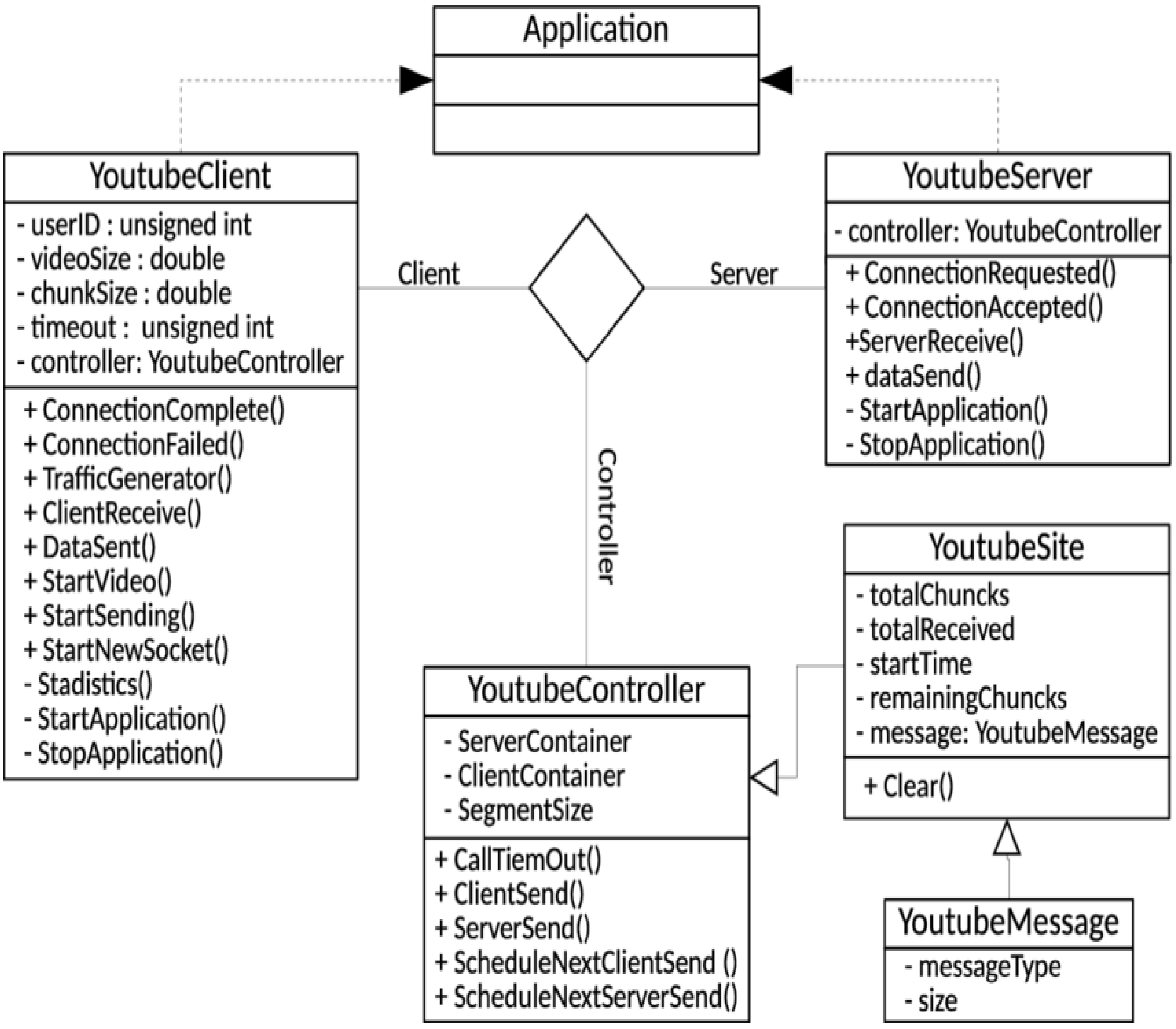

Figure 4.3: Class diagram

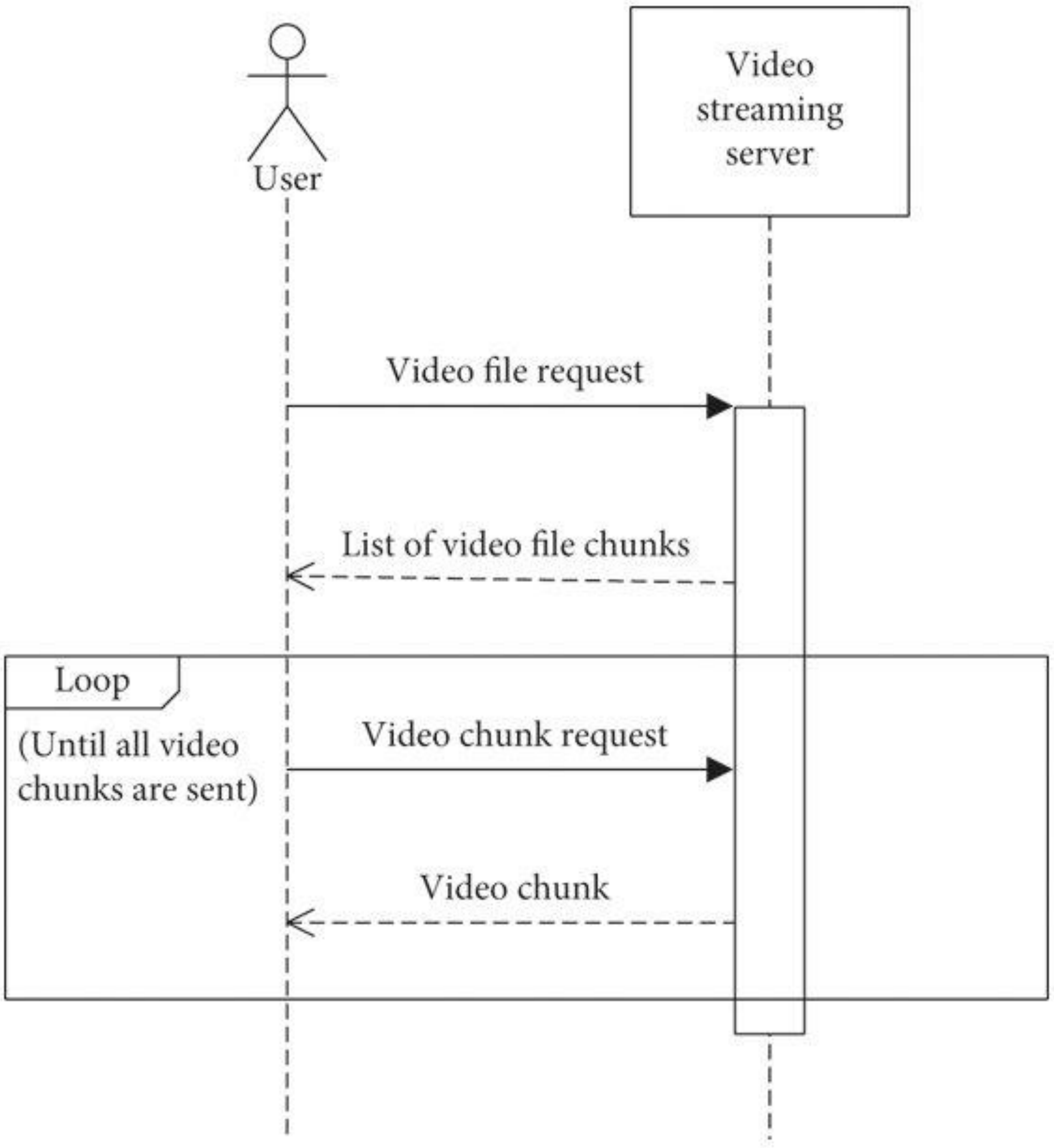

Figure 4.4: (a) Sequence diagram

CHAPTER 5

NORMALISATION

5.1Introduction

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.

5.2 Types and their description

There are four types of Normal Forms;

- 1NF
- 2NF
- 3NF
- BCNF

Their descriptions are provided below,

Normal Forms	**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 transaition dependency exists.
BCNF	A relation will be in it is in Boyce code normal form and has no multi-valued dependency.

5.3 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

5.4 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.

CHAPTER 6
IMPLEMENTATION AND SCREENSHOTS

This is a YouTube video link, where the woking and the functionalities of this application are explained : https://yt-youtube.netlify.app/

Below, we show the actual working screenshots of the application

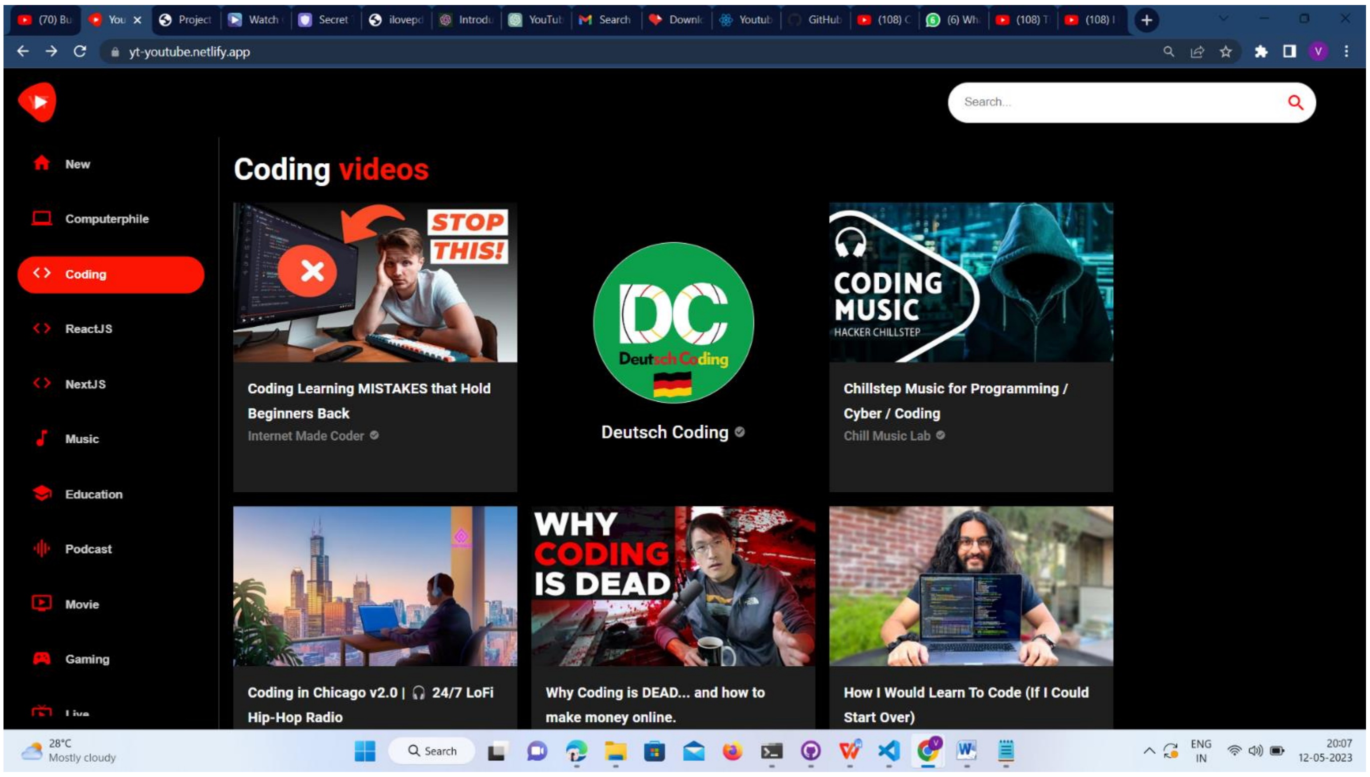
yt-youtube.netlify.app
Search...
New
Computerphile
Coding
ReactJS
NextJS
Music
Education
Podcast
Movie
Gaming
Live
Coding videos
Coding Learning MISTAKES that Hold Beginners Back
Internet Made Coder
Deutsch Coding
Chillstep Music for Programming / Cyber / Coding
Chill Music Lab
Coding in Chicago v2.0 | 🎧 24/7 LoFi Hip-Hop Radio
Why Coding is DEAD... and how to make money online.
How I Would Learn To Code (If I Could Start Over)

Figure 6.1: Coding videos

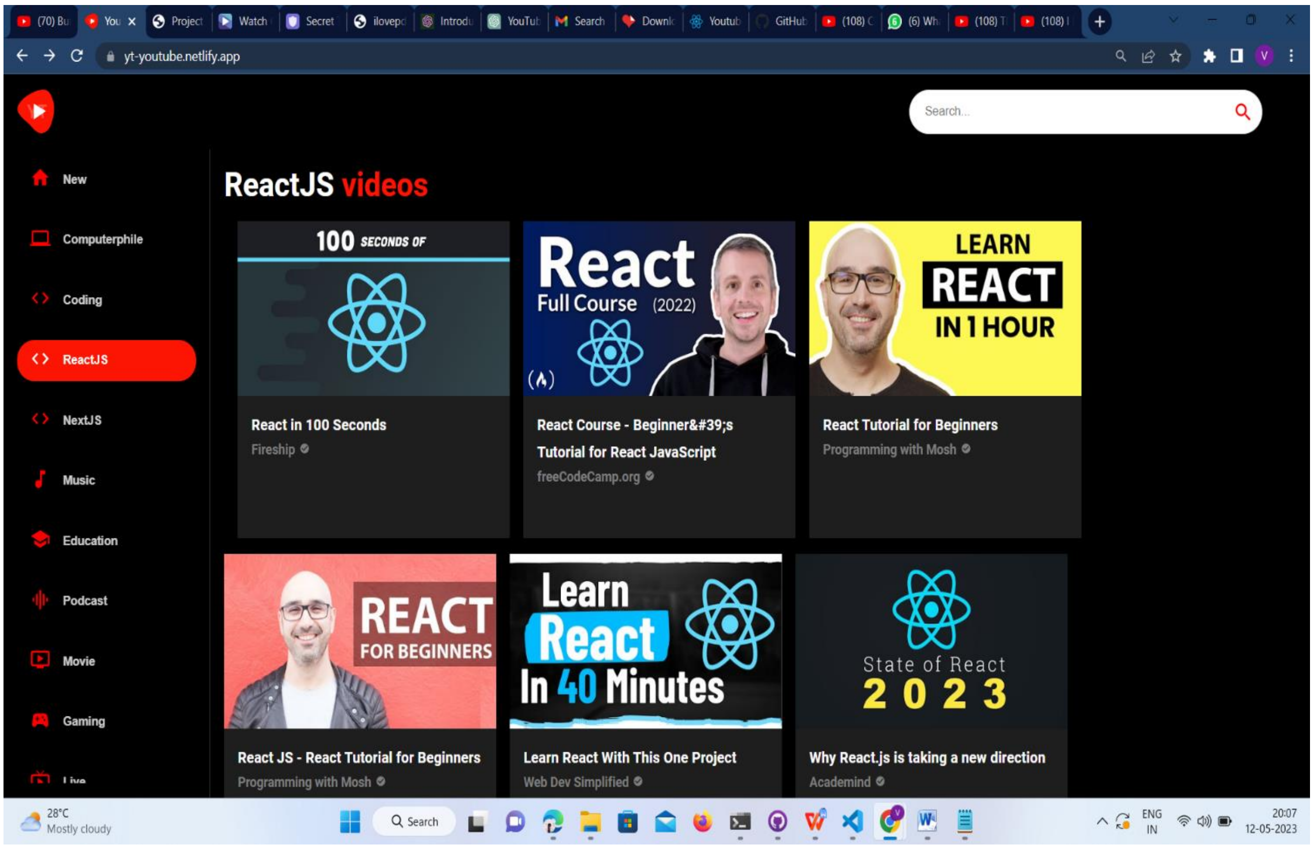

Figure 6.2: ReactJS videos page

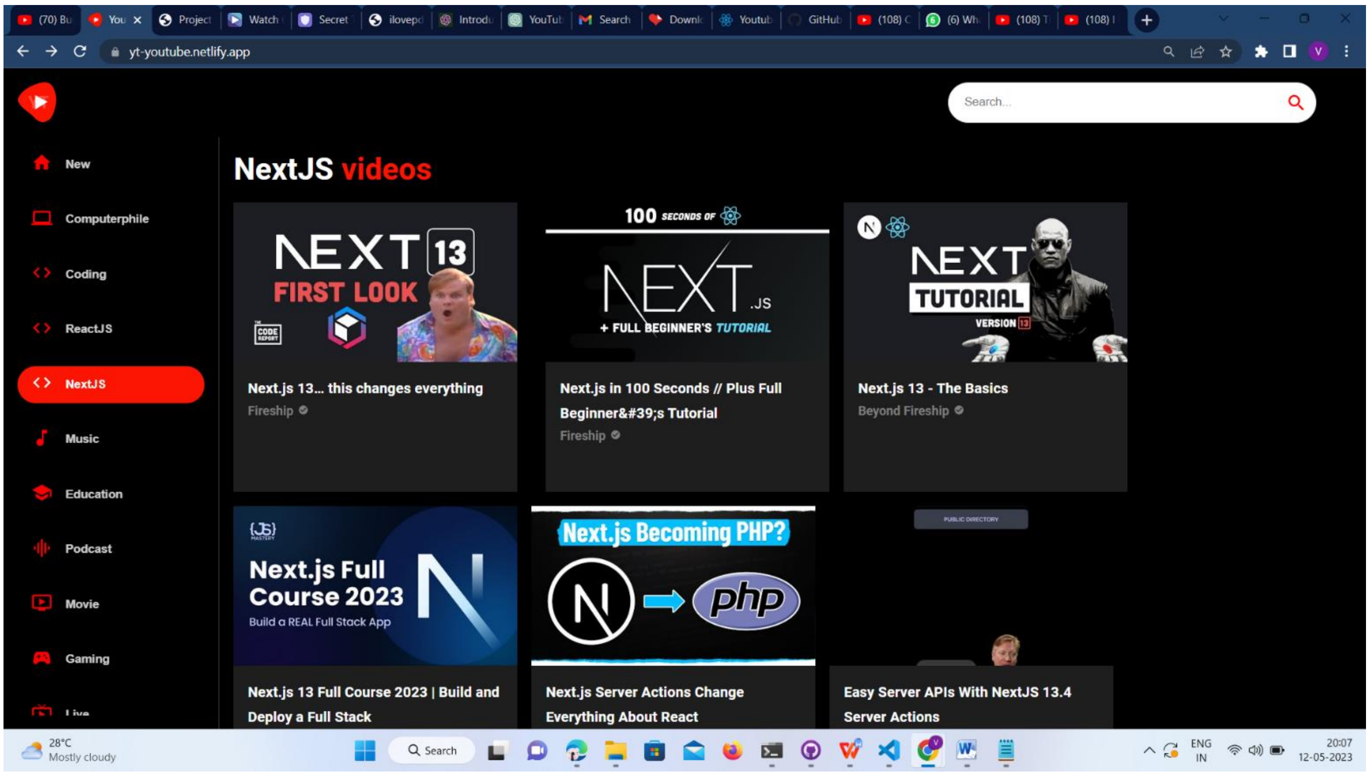
yt-youtube.netlify.app
Search...
New
Computerphile
Coding
ReactJS
NextJS
Music
Education
Podcast
Movie
Gaming
NextJS videos
Next.js 13... this changes everything
Fireship
Next.js in 100 Seconds // Plus Full Beginner&#39;s Tutorial
Fireship
Next.js 13 - The Basics
Beyond Fireship
Next.js 13 Full Course 2023 | Build and Deploy a Full Stack
Next.js Server Actions Change Everything About React
Easy Server APIs With NextJS 13.4 Server Actions

Figure 6.3: NextJS videos page

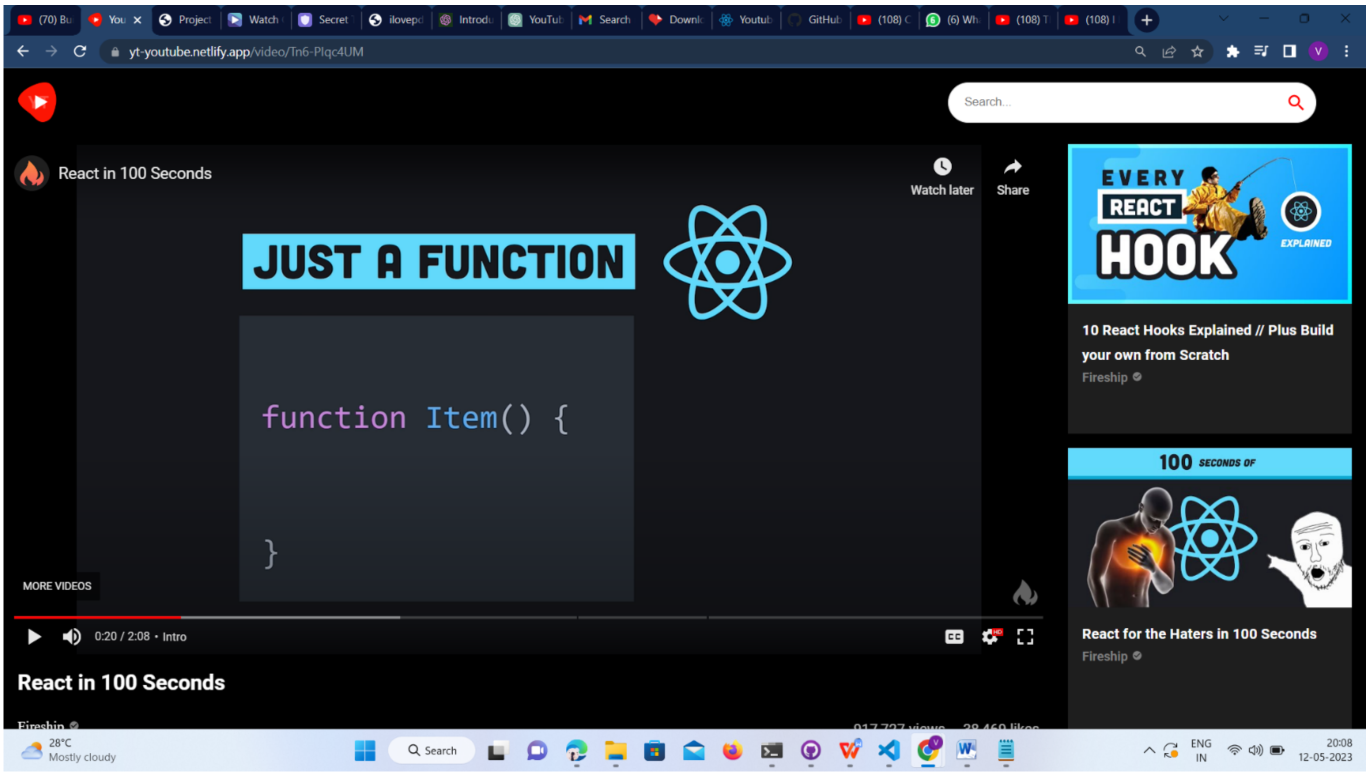

Figure 6.4: Video playing page

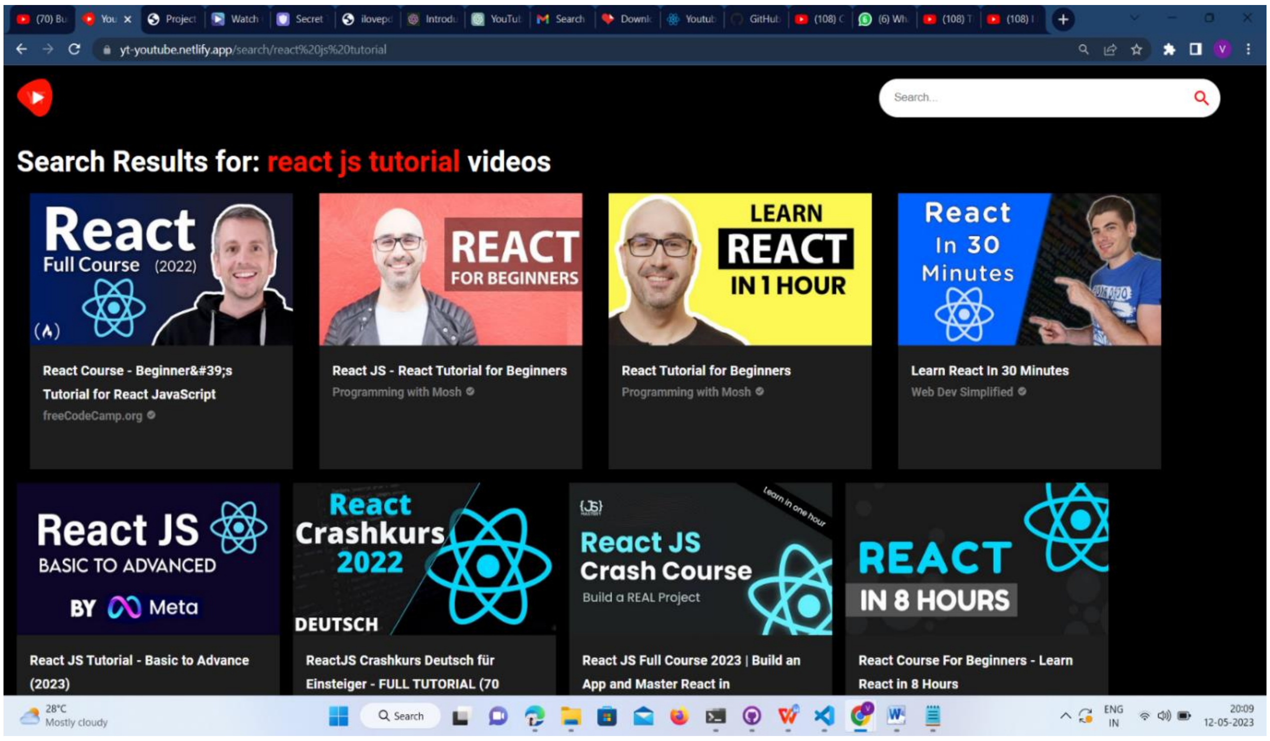
yt-youtube.netlify.app/search/react%20js%20tutorial
Search...
Search Results for: react js tutorial videos
React Course - Beginner&#39;s Tutorial for React JavaScript
freeCodeCamp.org
React JS - React Tutorial for Beginners
Programming with Mosh
React Tutorial for Beginners
Programming with Mosh
Learn React In 30 Minutes
Web Dev Simplified
React JS Tutorial - Basic to Advance (2023)
ReactJS Crashkurs Deutsch für Einsteiger - FULL TUTORIAL (70
React JS Full Course 2023 | Build an App and Master React in
React Course For Beginners - Learn React in 8 Hours

Figure 6.5: Search result page

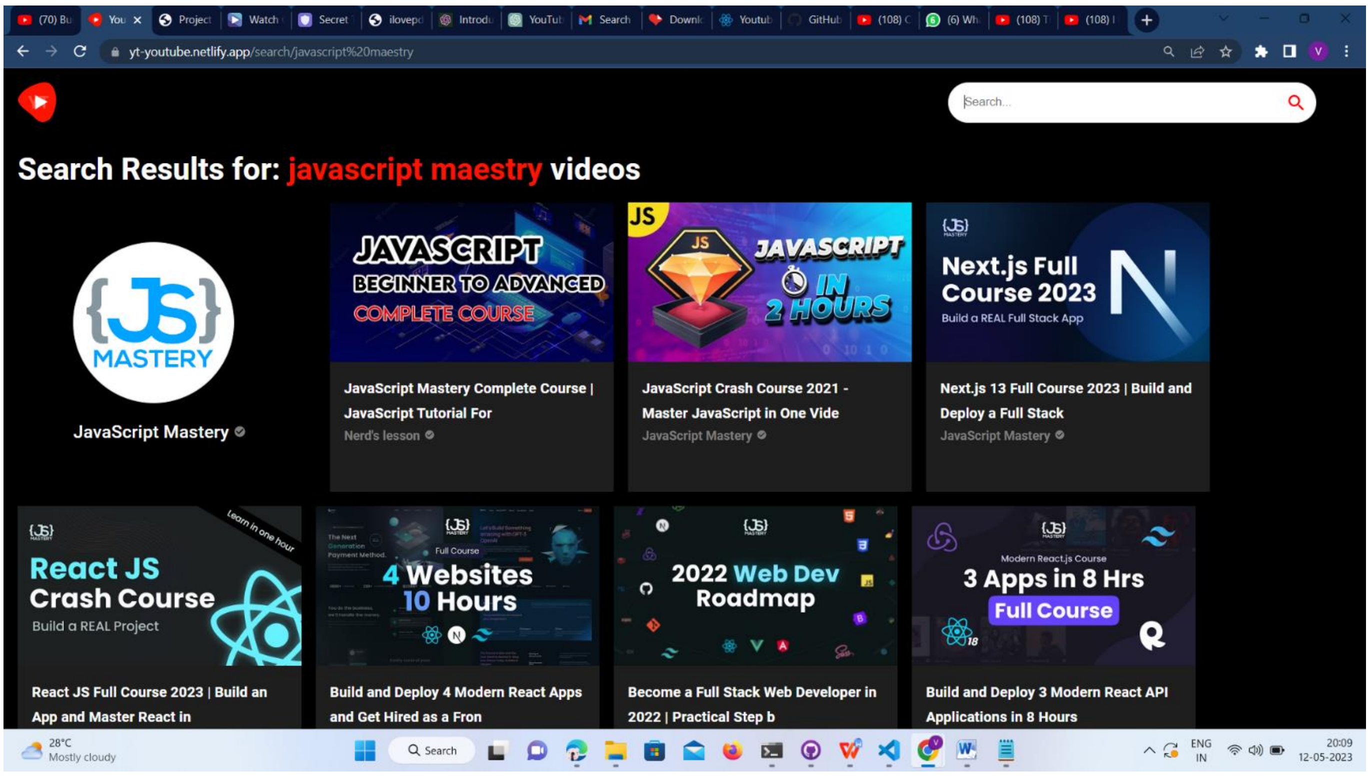
yt-youtube.netlify.app/search/javascript%20maestry
Search...
Search Results for: javascript maestry videos
JavaScript Mastery
JavaScript Mastery Complete Course | JavaScript Tutorial For
Nerd's lesson
JavaScript Crash Course 2021 - Master JavaScript in One Vide
JavaScript Mastery
Next.js 13 Full Course 2023 | Build and Deploy a Full Stack
JavaScript Mastery
React JS Full Course 2023 | Build an App and Master React in
Build and Deploy 4 Modern React Apps and Get Hired as a Fron
Become a Full Stack Web Developer in 2022 | Practical Step b
Build and Deploy 3 Modern React API Applications in 8 Hours

Figure 6.6: Channel view page

CHAPTER 7

CONCLUSION AND REFERENCE

7.1 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.

7.2 References

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