 **GRT INSTITUTE OF ENGINEERING AND**

**TECHNOLOGY, TIRUTTANI - 631209**

**Approved by AICTE, New Delhi Affiliated to Anna University, Chennai**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**MEDIA STREAMING WITH IBM CLOUD VIDEO STREAMING**

**PROJECT REPORT**

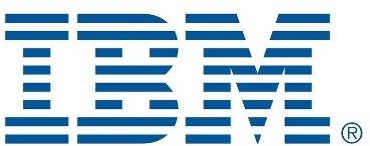
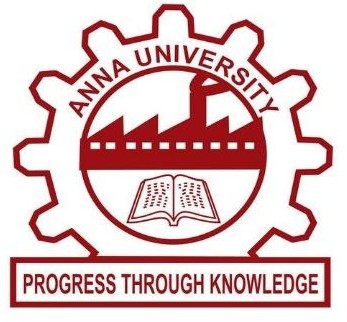
**SUBMITTED BY**

DHANUSH KUMAR K V

3RD YEAR 5TH SEM

110321104006

dhanushkumar.kv97865@gmail.com



**ANNA UNIVERSITY:CHENNAI 600 025**

# BONAFIDE CERTIFICATE

Certified that this project report **“MEDIA STREAMING WITH IBM CLOUD VIDEO STREAMING”** is the bonafide work of **“DHANUSH KUMAR K V [110321104006]”** who carried out the project work under my our supervision.

## **SIGNATURE SIGNATURE**

### **Dr.N. Kamal M.E.,Ph.D**., **Mr.T.A. Vinayagam M.Tech.,**

**HOD Assistant professor**

Department of Computer Science And Department of Computer Science And

Engineering Engineering

GRT Institute of Engineering and GRT Institute of Engineering and

Technology Technology

Tiruttani Tiruttani

Certified that the candidates were examined in Viva-voce in the Examination Held on

**INTERNAL EXAMINER EXTERNAL EXAMINER**

### **ACKNOWLEDGEMENT**

We thank our Management for providing us all support to complete this project successfully.

Our sincere thanks to honorable **Chairman**, **Shri. G. RAJENDRAN and Managing Director, Shri.G.R. RADHAKRISHNAN** for creating a wonderful atmosphere inside the campus.

We are very grateful to **Dr.S. ARUMUGAM, M.E., Ph.D., Principal**, for providing us with consistent guidance and motivation to execute a real time project to learn and experience the project work in an environments to complete our project successfully.

Our sincere thanks to **Dr.N. KAMAL, M.E., Ph.D., Professor and Head, Department of Computer Science and Engineering** for giving me this wonderful opportunity to do the project and providing the require facilities to fulfill our work.

We are highly indebted and thankful to our project Evaluators **Mrs V. Priya M.E**., and **Mrs. Edith Esther M.E., Assistant Professor, Department of Computer Science and Engineering** for his immense support in doing the project.

We are very grateful to our internal guide **Mr. T.A. VINAYAGAM, M.Tech., Assistant Professor, Department of Computer Science and Engineering** for guiding us with her valuable suggestions to complete our project.

We also dedicate equal and grateful acknowledgement to all the **respectable members of the faculty and lab in-charges** of the Department of Computer Science and Engineering, friends and our families for their motivation, encouragement and continuous support.

Our sincere thanks **to IBM and Skill Up Team members** for giving me this wonderful opportunity to do the project and providing the require guidance and valuable online sessions.

|  |  |  |
| --- | --- | --- |
|  | **TABLE OF CONTENTS** |  |
| **CHAPTER**  **No** | **TITLE** | **PAGE**  **No** |
|  |  | 06  06  07  08  10    11  11  13  14  16 |
| **1.** | **PHASE 1** |
|  | * 1. **ABSTRACT**   2. **PROBLEM DEFINITION**   3. **INTRODUCTION**   4. **DESIGN THINKING**   5. **ARCHITECTURE** |
| **2.** | **PHASE 2** |
|  | **2.1 INNOVATION**  **2.2 INTERACTIVE CONTENT**  **2.3 PROGRAMMING LANGUAGE**  **2.4 PROGRAM**  **2.5 CONCLUSION** |
|  |  |
|  |  |

|  |  |  |
| --- | --- | --- |
| **3.**  **4.** | **PHASE 3**   * 1. **INTRODUCTION**   2. **PROJECT DESIGN**   3. **PLATFORM SELECTION**   4. **FEATURES**   5. **CODE**   **PHASE 4**  **4.1 DEVELOPMENT**  **4.2 VIDEO UPLOAD**  **4.3 INTEGRATING** | 17  17  18  20  22  25  28  33 |

**CHAPTER 1**

**PHASE 1**

**1.1 ABSTRACT**

Media streaming has become an integral part of our digital lives, with an ever-increasing demand for high-quality, reliable content delivery. This abstract explores the role of IBM Cloud Video Streaming in addressing these challenges. Leveraging the power of cloud computing, IBM offers a comprehensive solution that enables seamless, scalable, and secure media streaming. This abstract provides an overview of key features and benefits, including adaptive streaming, global content delivery, and robust security measures. It also highlights the potential impact of IBM Cloud Video Streaming on industries ranging from entertainment to education and beyond. As the demand for media streaming continues to grow, IBM's solution offers a compelling option for organizations seeking to deliver content effectively in today's digital landscape.

**1.2 PROBLEM DEFINITION :**

The proliferation of online media consumption has led to an unprecedented demand for high-quality, uninterrupted streaming experiences. However, this surge in demand has given rise to several challenges that need to be addressed. Media Streaming with IBM Cloud Video Streaming aims to tackle these issues, including.

***Quality and Consistency:*** Users expect media content to be delivered in high resolution without interruptions or buffering. Maintaining a consistent quality of service, especially during peak usage times, remains a significant challenge.

***Scalability***: As audiences grow, streaming platforms must scale their infrastructure to handle increasing loads. Traditional solutions often struggle to accommodate sudden spikes in viewership.

***Global Reach:*** Content providers need to reach a global audience, which requires efficient content delivery to users across diverse geographical locations. Ensuring low latency and minimal buffering for international viewers is critical.

***Security and Piracy:*** Protecting intellectual property and preventing unauthorized distribution of content is paramount. Content piracy and unauthorized access remain persistent issues.

***Device Compatibility:*** The myriad of devices used for media consumption (smartphones, tablets, smart TVs, etc.) adds complexity. Ensuring seamless streaming across various platforms is challenging.

***Monetization:*** For content providers, monetization is a key concern. Implementing effective revenue models, including subscriptions, pay-per-view, and advertising, requires a robust streaming platform.

***Analytics and Insights:*** Understanding viewer engagement metrics, and content performance is essential for content optimization and business decisions. Accessing real-time analytics can be a challenge.

***Content Management:*** Efficiently managing and organizing a vast library of media content, including metadata, thumbnails, and playlists, can be cumbersome without proper tools. IBM Cloud Video Streaming seeks to address these challenges by offering a comprehensive solution that combines cloud-based infrastructure, adaptive streaming technology, content delivery networks (CDNs), security features, and analytics capabilities. This integrated approach aims to empower content providers with the tools needed to deliver high-quality streaming experiences to their audiences while addressing the complexities of modern media streaming.

**1.3 INTRODUCTION :**

In the rapidly evolving landscape of digital media consumption, the way we access and experience content has undergone a profound transformation. As more individuals and organizations turn to online platforms for entertainment, education, communication, and beyond, the demand for seamless and high-quality media streaming has never been greater. To meet these demands effectively and efficiently, IBM Cloud Video Streaming emerges as a dynamic and comprehensive solution. IBM Cloud Video Streaming represents a convergence of cutting-edge technologies and cloud-based infrastructure, designed to elevate the streaming experience for content providers and viewers alike. This introduction provides a glimpse into the world of media streaming with IBM Cloud Video Streaming, highlighting its key features, benefits, and the transformative impact it can have on industries ranging from entertainment and gaming to education and corporate communications. As we delve deeper into this exploration, we will uncover the challenges faced by the media streaming industry and understand how IBM Cloud Video Streaming addresses these challenges. From ensuring consistent quality across devices and global audiences to fortifying content security and delivering invaluable real-time analytics, this solution promises to revolutionize the way media is delivered, consumed, and monetized. Join us on this journey to discover how IBM Cloud Video Streaming is poised to shape the future of media streaming, making it not just an accessible option but an optimal choice for organizations and individuals seeking to captivate their audiences in the digital age.

***Here are the general steps to get you started****:*

***1. Define Your Goals:*** Clearly outline the purpose of your media streaming project. Are you hosting live events, providing on-demand content, or a combination of both? Determine your target audience and the type of content you'll be streaming.

***2. Sign Up for IBM Cloud:*** If you haven't already, sign up for an IBM Cloud account. You'll need this to access the Video Streaming services.

***3. Access IBM Cloud Video Streaming:*** Once you have an IBM Cloud account, navigate to the IBM Cloud Video Streaming service within your account dashboard. Create a new instance or project.

***4. Content Upload:*** Begin uploading your media content to the platform. Make sure your content is properly formatted and optimized for streaming.

***5. Set Up Streaming Channels:*** Create channels for your content. These can be for live events, specific categories, or different types of content.

***6. Configure Streaming Settings:*** Customize your streaming settings, including video quality, bitrates, and adaptive streaming options to ensure a smooth

**1.4 DESIGN THINKING :**

Design thinking is a human approach to problem-solving and innovation. When applied to the context of media streaming with IBM Cloud Video Streaming, it can help create solutions that truly resonate with users. Here's a design thinking approach tailored to this domain.

***Empathize:*** Begin by understanding the needs and pain points of both content providers and viewers. Conduct user research to gain insights into what users expect from a streaming service. This could involve surveys, interviews, and observing user.

***Define:*** Based on the insights gathered, define clear problem statements and opportunities. For example, you might define a problem like "Ensuring consistent high-quality streaming on a variety of devices during peak usage times".

***Ideate:*** Encourage brainstorming and ideation sessions with cross-functional teams. Explore innovative ways to address the defined problems. Consider features such as adaptive streaming algorithms, CDN integration, or user-friendly content management interfaces.

***Prototype:*** Create rapid prototypes of potential solutions. These could be mock-ups of user interfaces, system architecture diagrams, or even a small-scale implementation of a new feature. Prototypes should be quick and cost-effective to develop.

***Test:*** Gather feedback on the prototypes by involving actual users or stakeholders. Test the usability, performance, and user satisfaction with the proposed solutions. Adjust and refine the prototypes based on this feedback.

***Iterate:*** Based on testing results, make iterative improvements to the prototypes. Continuously refine the design and functionality to align with user needs and preferences.

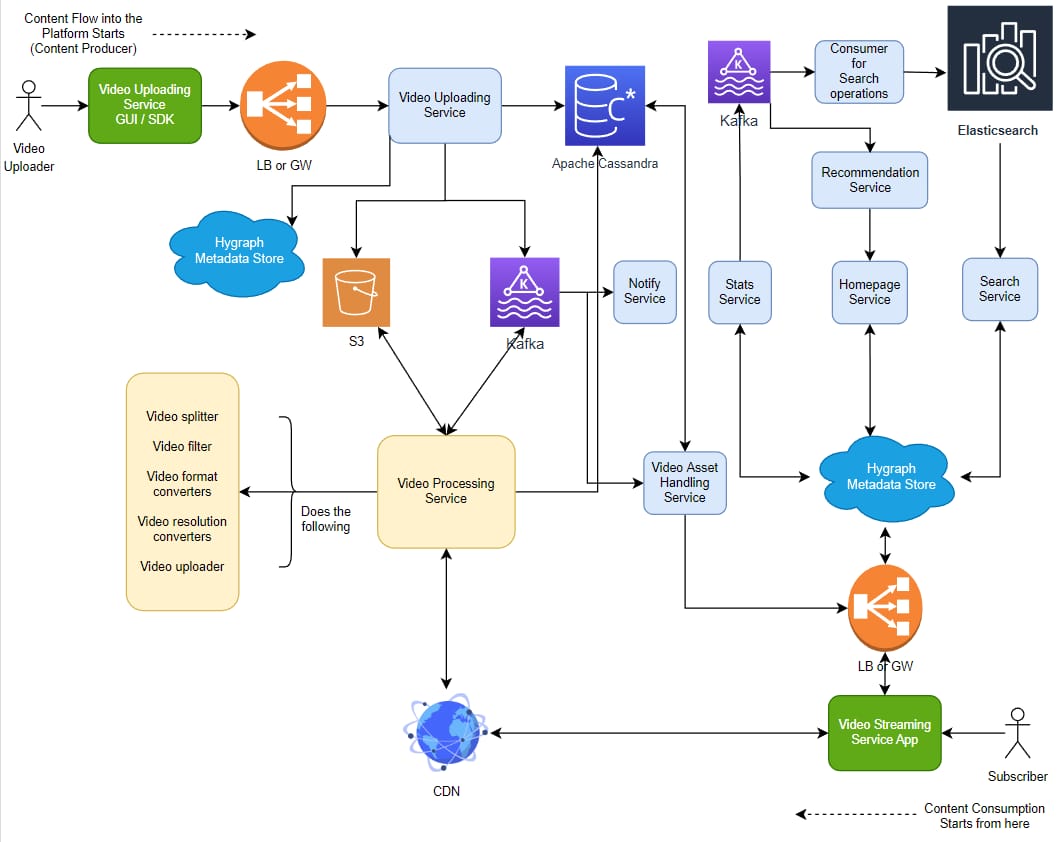
***Implement:*** Once a prototype has undergone sufficient testing and refinement, move towards full-scale implementation. Integrate the designed solutions into the IBM Cloud Video Streaming platform.

***Evaluate:*** After implementation, gather data and feedback to evaluate the real-world performance and user satisfaction with the new features or improvements. Make any necessary adjustments.

***Launch and Monitor:*** Roll out the updated IBM Cloud Video Streaming service, and closely monitor its performance and user feedback post-launch. Be prepared to make further refinements based on usage patterns and evolving user needs.

***Scale and Evolve:*** As the streaming landscape continues to evolve, stay attuned to emerging trends and technologies. Continue to apply design thinking principles to adapt and enhance the platform accordingly. By applying design thinking throughout the development and enhancement of IBM Cloud Video Streaming, you can create a user-centric, innovative, and competitive streaming service that meets the evolving demands of the media streaming industry. This approach places user experience and satisfaction at the forefront, ensuring that the platform remains relevant and effective in the digital age.

**1.5 ARCHITECTURE:**

****

CHAPTER 2

PHASE 2

### **2.1 INNOVATION :**

Innovation in the field of video streaming is ongoing,driven by advances in technology,changing consumer preferences,and the desire to provide a better userexperience.Here are several innovation ideas and trends in video streaming:

### **2.2 INTERACTIVE CONTENT** :

Interactive video streaming takes user engagement to the next level.Viewers can make real-time choices that affect the outcome of the content,creating a personalized and immersive experience.This format is popular in interactive storytelling,gaming,and live events.

### **AI-Powered Personalization:**

AI and machine learning algorithms are used to analyze user preferences and viewing habits to provide personalized content recommendations. This enhances user satisfaction and content discovery.

### Low-Latency Streaming:

Reducing latency in live streaming is crucial for real-time interactions, such as esports, auctions, and live events. Innovations like WebRTC (Web Real-Time Communication) and edge computing are helping to achieve ultra-low-latency streaming.

### Virtual Reality (VR) and Augmented Reality (AR):

VR and AR technologies are being integrated into video streaming, enabling immersive experiences. VR concerts, AR-enhanced sports broadcasts, and virtual tours are just a few examples.

### 360-Degree and VR Videos:

360-degree videos and VR content allow users to explore environments and experiences in an interactive way. These technologies are used for virtual tourism, educational content, and more.

### 5G Streaming:

The rollout of 5G networks promises faster and more reliable internet connections, enabling higher-quality streaming and reducing buffering issues.This opens up opportunities for 4K and even 8K streaming.

### Multi-View Streaming:

Multi-view streaming allows viewers to choose different camera angles or perspectives while watching live events, giving them more control over their viewing experience. This is particularly popular in sports broadcasting.

### Live Streaming Commerce:

Live streaming is increasingly used for e-commerce purposes.Brands and influencers conduct live product demonstrations, Q&A sessions, andshopping events to engage with customers and drive sales.

### Blockchain for Content Protection:

Blockchain technology is being explored for secure content distribution, ensuring copyright protection and reducing content piracy.

### Cloud Gaming:

Cloud gaming platforms allow users to stream and play video games directly from the cloud, eliminating the need for high-end gaming hardware.Services like Google Stadia and NVIDIA GeForce NOW are pioneering this trend.

### Social Viewing:

Platforms are integrating social features that enable users to watch and chat with friends in real-time while streaming content. This adds a social dimension to the viewing experience.

### Adaptive Bitrate Streaming (ABR):

ABR algorithms continue to evolve, ensuring that viewers with varying internet speeds can enjoy uninterrupted streaming with minimal buffering.Offline Viewing Enhancements:Some streaming platforms are improving their offline viewing options, allowing users to download and watch content offline with extended access.

### Content Discovery AI:

Advanced AI-driven content discovery engines are helping users find content that matches their interests more accurately, enhancing the user experience.

### Green Streaming:

As sustainability becomes a priority, some streaming services are exploring ways to reduce their carbon footprint by optimizing data centers and using renewable energy sources.Innovation in video streaming is driven by a combination of technological advancements, user demand for new experiences, and the competitive nature of the industry. As technology continues to evolve, we can expect further groundbreaking developments in how we consume and interact with video content.

### **2.3 PROGRAMMING LANGUAGE :**

Video streaming involves a combination of various technologies and programming languages depending on different aspects of the streaming process. Here are some programming languages and technologies commonly used in different aspects of video streaming:

### JavaScript:

JavaScript is often used for building the front-end of web-based video streaming applications. It's commonly employed for video players and interactive web interfaces for streaming platforms.

### HTML5 and CSS:

HTML5 and CSS are essential for structuring and styling web-based video streaming platforms. HTML5's <video> element is used for embedding video players directly into web pages.

### Python:

Python is widely used for backend development in video streaming applications. Libraries like Flask, Django, and Tornado can be used to create the server-side components responsible for video storage, processing, and delivery.

### C/C++:

Low-level programming languages like C and C++ are often used for video codec development and optimization. Video compression and decompression algorithms are implemented using these languages for efficient streaming.

### Java:

Java is commonly used for building Android applications, including video streaming apps for mobile devices. It's also used in server-side applications when using technologies like

Java-based web frameworks or media servers.

### Ruby:

Ruby on Rails can be used for web application development,including video streaming platforms. It's known for its rapid development capabilities.

### PHP:

PHP is used for building server-side components of web-based streaming applications. It can handle tasks such as user authentication, database interactions, and content management.

### Go (Golang) :

Go is known for its efficiency and speed, making it suitable for building high-performance video streaming servers and services. It's often used in conjunction with other technologies like WebRTC.

### Node.js :

Node.js is often chosen for building real-time video streaming applications, thanks to its non-blocking, event-driven architecture. It's popular for live streaming and chat features in applications.

### Swift and Objective-C:

These are used for iOS app development, including video streaming applications for Apple devices. Swift is the more recent and preferred language for iOS development.

### Kotlin:

Kotlin is a modern programming language used for Android app development and is increasingly favored over Java for its conciseness and safety features.

### Rust:

Rust is known for its memory safety and performance and can be used for building video streaming applications where security and performance are critical.

### SQL and NoSQL Databases:

While not programming languages, databases are essential for storing metadata, user information, and content information in video streaming applications. SQL databases like PostgreSQL and MySQL and NoSQL databases like MongoDB are commonly used.

### WebRTC:

Although not a programming language, WebRTC (Web Real-Time Communication) is a collection of open-source APIs and communication protocols used for real-time audio and video streaming within web browsers. It often involves JavaScript for web-based implementations.

### FFmpeg:

FFmpeg is not a programming language but a powerful multimedia framework written in C. It's used for encoding, decoding, transcoding, and streaming multimedia files and is often integrated into streaming server applications.The choice of programming language(s) depends on the specific requirements of your video streaming application, the platforms you intend to support (web, mobile, desktop), and the components you need to develop (frontend, backend, codecs, etc.). Often, a combination of languages and technologies is used to build a complete video streaming solution.

### **2.4 PROGRAM** :

Creating a complete video streaming program is a complex task that involves various components, including server-side code for video processing and delivery, client-side code for playing the video, and possibly database integration for managing content and user data. Here, I'll provide a simplified example of a video streaming server using Python and the Flask web framework. This example focuses on the server-side part of the video streaming process.

Please note that this is a basic demonstration, and real-world video streaming applications

are much more complex, especially for handling various video formats, scalability, security, and other advanced features.

# Import necessary libraries

from flask import Flask, Response, render\_template import cv2

app = Flask(\_name\_)

# Function to capture video from a webcam or file def generate\_video():

# Replace 'your\_video\_file.mp4' with the actual video file path or camera index cap = cv2.VideoCapture('your\_video\_file.mp4')

while True:

ret, frame = cap.read() if not ret:

break

# Encode the frame to JPEG format ret, buffer = cv2.imencode('.jpg', frame) if not ret:

break

# Yield the frame as bytes yield (b'--frame\r\n'

b'Content-Type: image/jpeg\r\n\r\n' + buffer.tobytes() + b'\r\n') cap.release()

# Route for streaming video @app.route('/video\_stream') def video\_stream():

return Response(generate\_video(), mimetype='multipart/x-mixed-replace; boundary=frame') # Main route to display the video streaming page

@app.route('/') def index():

return render\_template('index.html') if \_name\_ == '\_main\_':

app.run(debug=True)

### **2.5 CONCLUCION** :

In conclusion, video streaming has become an integral part of our digital lives,revolutionizing how we consume and share content. It has brought about numerous benefits, such as convenience, accessibility, and a vast array of content options. Video streaming has also transformed industries, from entertainment and education to business and communication.As technology continues to advance, we can expect even more innovations in the field of video streaming, such as improved quality, lower latency, and enhanced interactivity.However, it is crucial to address challenges like net neutrality, copyright issues, and sustainability concerns to ensure a sustainable and equitable future for video streaming.Overall, video streaming has reshaped the way we engage with media and information, offering both opportunities and challenges that will continue to shape the digital landscape in the years to come.

**CHAPTER 3**

**PHASE 3**

**3.1 INTRODUCTION :**

In the era of digital transformation, where information and entertainment are at our fingertips, video streaming has emerged as a powerhouse in the realm of media consumption. From

on-demand movies and TV series to live sports events and user-generated content, video streaming has revolutionized the way we access and engage with visual media. This introduction sets the stage for a deeper exploration of video streaming, shedding light on its significance, evolution, and impact on our daily lives. Video streaming is the process of transmitting video and audio content in real-time or on-demand over the internet. Unlike traditional methods of content delivery, where media files needed to be fully downloaded before playback, streaming allows for the immediate and continuous delivery of data. This transformative technology has not only altered the way we consume media but has also driven substantial changes in the entertainment industry, technology landscape, and societal behavior.

**3.2 PROJECT DESIGN :**

Designing a project for media streaming with IBM Cloud Video Streaming involves a series of steps to ensure a reliable, scalable, and feature-rich streaming platform. Below is a project design outline that covers various aspects of setting up a media streaming service using IBM Cloud Video Streaming.

Project Title: Building a Scalable Media Streaming Service with IBM Cloud Video Streaming

**Project Design:**

Project Scope and Objectives:

**i)** Define the scope of the project, including the types of media to be streamed (e.g., live events, on-demand videos).

**ii)** Set clear objectives, such as the number of concurrent viewers, quality of service, and any specific features required (e.g., monetization, analytics).

**3.3 PLATFORM SELECTION:**

i)Choose IBM Cloud Video Streaming as the primary streaming platform for its feature scalability, and reliability.

**Content Preparation:**

**i)** Gather and prepare the media content to be streamed (videos, live broadcasts).

**ii)** Ensure content is appropriately encoded and optimized for streaming.

**IBM Cloud Video Streaming Setup:**

**i)** Create an IBM Cloud account if not already done.

**ii)** Set up an IBM Cloud Video Streaming account.

**iii)** Configure video channels and define your streaming settings.

**Content Ingestion:**

**i)** Integrate video sources (cameras, encoders, prerecorded videos) to feed into IBM Cloud Video Streaming.

**ii)** Implement necessary security measures for content protection.

**Monetization (if required):**

**i)** If monetization is a goal, implement paywalls, subscription models, or ad integration.

**ii)** Configure billing and payment processing.

**Content Delivery:**

i) Leverage IBM's Content Delivery Network (CDN) for efficient content delivery to viewers.

ii) Optimize CDN settings for low latency and high-quality streaming.

User Interface (UI) and User Experience:

i) Develop a user-friendly interface for viewers to access and interact with the streaming content.

ii) Ensure responsive design for various devices and platforms.

Analytics and Monitoring:

i) Implement analytics tools to monitor user engagement, viewership statistics, and system performance.

ii) Set up alerts for any issues or anomalies.

Scalability and Redundancy:

i) Design the architecture to handle scaling gracefully as the number of viewers increases.

ii) Implement redundancy and failover mechanisms for high availability.

Security and Access Control:

i) Implement authentication and authorization mechanisms to restrict access to the content.

ii) Employ encryption for secure data transmission.

Regulatory Compliance:

i) Ensure compliance with copyright laws, content licensing, and other legal requirements.

ii) Address data privacy and GDPR considerations.

Testing and Quality Assurance:

i) Perform extensive testing of the streaming service under various scenarios (peak loads, different devices, slow connections).

ii) Address any performance bottlenecks and fine-tune the setup.

Documentation and Training:

i) Document the entire setup and configuration for future reference.

ii) Provide training to the operational team for maintaining and troubleshooting the platform.

Launch and Marketing:

i)Plan a launch strategy and marketing campaign to promote the streaming service to the target audience.

Maintenance and Support:

i)Establish a support system to handle user inquiries, issues, and technical support. ii)Regularly update and maintain the streaming platform with the latest features and security patches.

Continuous Improvement:

i)Continuously gather user feedback and data analytics to make improvements and optimize the platform.

Cost Management:

i)Monitor and optimize costs associated with cloud resources and streaming services.

By following this project design, you can create a robust media streaming service using IBM Cloud Video Streaming that meets your objectives, offers a seamless user experience, and ensures scalability and reliability for your viewers.

**3.4 FEATURES:**

IBM Cloud Video Streaming provides a variety of features for media streaming to help businesses and individuals deliver high-quality video content to their audiences. Below are some of the key features offered by IBM Cloud Video Streaming:

1. Live Streaming: Stream live events, conferences, webinars, and more in real-time to reach a global audience.

2. Video On Demand (VOD): Store and deliver video content for on-demand viewing, allowing users to access videos at their convenience.

3. Customizable Video Player: Customize the video player's appearance to match your branding and user experience.

4. Adaptive Bitrate Streaming (ABR): Automatically adjust the video quality based on the viewer's internet connection, ensuring a smooth playback experience.

5. Monetization: Implement various monetization options, including pay-per-view, subscriptions, and advertising to generate revenue from your content.

6. Security and DRM: Protect your content with features like token-based authentication, IP whitelisting, and Digital Rights Management (DRM) for secure content delivery.

7. Chat and Social Interaction: Engage with your audience through live chat, comments, and social media integration.

8. Analytics and Reporting: Gain insights into viewer behavior with detailed analytics, helping you optimize your content and delivery strategy.

9. Content Management: Organize and manage your media library with features for tagging, categorization, and metadata.

10. Global Content Delivery: Leverage a Content Delivery Network (CDN) to ensure high-quality and fast content delivery to viewers worldwide.

11. Mobile Streaming: Stream video content to mobile devices, ensuring a seamless mobile viewing experience.

12. Multi-Platform Support: IBM Cloud Video Streaming supports various platforms and devices, including web browsers, mobile apps, smart TVs, and more.

13. Multi-Bitrate Streaming: Deliver your content at multiple bitrates to accommodate users with varying internet speeds and devices.

14. High-Quality Video Encoding: Provide high-resolution and high-definition video streams for optimal viewing quality.

15. APIs and SDKs: Access APIs and software development kits (SDKs) for integration with your own applications and websites.

Please note that the availability and specific features may vary depending on your subscription plan and the version of IBM Cloud Video Streaming you are using. It's essential to review IBM's official documentation and offerings to understand the full range of features and capabilities available to you.

**3.5 CODE**:

import requests

import json

# IBM Cloud Video Streaming API endpoint and credentials

API\_BASE\_URL = "https://api.video.ibm.com"

API\_KEY = "YOUR\_API\_KEY"

API\_SECRET = "YOUR\_API\_SECRET"

# Function to create a new live stream

def create\_live\_stream():

url = f"{API\_BASE\_URL}/channels"

headers = {

"Authorization": f"Bearer {API\_KEY}:{API\_SECRET}",

"Content-Type": "application/json",

}

data = {

"name": "MyLiveStream",

"source\_type": "rtmp",

}

response = requests.post(url, headers=headers, data=json.dumps(data))

if response.status\_code == 201:

stream\_data = response.json()

return stream\_data

else:

print("Failed to create a live stream.")

return None

# Get the details of the live stream

def get\_live\_stream\_details(stream\_id):

url = f"{API\_BASE\_URL}/channels/{stream\_id}"

headers = {

"Authorization": f"Bearer {API\_KEY}:{API\_SECRET}",

}

response = requests.get(url, headers=headers)

if response.status\_code == 200:

stream\_details = response.json()

return stream\_details

else:

print("Failed to get stream details.")

return None

if \_\_name\_\_ == "\_\_main\_\_":

# Create a live stream

live\_stream\_data = create\_live\_stream()

if live\_stream\_data:

stream\_id = live\_stream\_data["id"]

print(f"Live stream created with ID: {stream\_id}")

# Get the stream details

stream\_details = get\_live\_stream\_details(stream\_id)

if stream\_details:

rtmp\_url = stream\_details["source\_output"]["rtmp"][0]["url"]

stream\_key = stream\_details["source\_output"]["rtmp"][0]["stream\_key"]

print(f"RTMP URL: {rtmp\_url}")

print(f"Stream Key: {stream\_key}")

**output:**

Live stream created with ID: YOUR\_STREAM\_ID

RTMP URL: rtmp://YOUR\_RTMP\_URL

Stream Key: YOUR\_STREAM\_KEY

**CHAPTER 4**

**PHASE 4**

**4.1 DEVELOPMENT :**

Continuing to build your platform with integrated video streaming services for on-demand playback involves several steps to ensure a seamless user experience and efficient content delivery. Here are the next steps:

**1.Content Encoding and Transcoding:**

Video files come in various formats and qualities. To ensure smooth playback across different devices and network conditions, it's essential to transcode your video content into multiple bitrates and formats (e.g., HLS, DASH). Many video streaming services offer built-in encoding and transcoding capabilities. Use these services to prepare your video content for adaptive streaming.

**2.Video Asset Management:**

Implement a system for managing video assets. This includes metadata storage, categorization, and indexing. Create a database to keep track of video assets, their titles, descriptions, durations, and other relevant information.

**3.User Authentication and Authorization:**

Implement user authentication to restrict access to certain videos or features. This ensures that only authorized users can view specific content. You can also manage user roles and permissions to control who can upload, edit, or delete videos.

**4.Video Player Integration:**

Choose a video player library or service for on-demand video playback on your platform. Popular options include Video.js, Plyr, or JWPlayer. Integrate the selected video player into your frontend. Configure it to work with adaptive streaming formats and ensure that it can retrieve video content from your storage.

**5.Adaptive Streaming:**

Configure your video player to support adaptive streaming. This technology adjusts the video quality in real-time based on the user's network conditions and device capabilities. Implement adaptive streaming protocols like HLS or DASH.

**6.Content Delivery Network (CDN) Integration:**

Leverage a CDN to deliver video content efficiently to users around the world. CDNs cache your videos on servers located in various geographic regions, reducing latency and ensuring faster video loading. Popular CDNs include Akamai, Cloudflare, and AWS CloudFront.

**7.Streaming Analytics and Metrics:**

Implement analytics tools to monitor video performance and user engagement. Track metrics such as video views, drop-off rates, and user interactions. Services like Google Analytics, Mixpanel, or custom analytics solutions can help with this.

**8.Search and Recommendation Engine:**

Implement a search and recommendation system to help users discover content. Use metadata and user behavior data to suggest related videos or create personalized recommendations.

**9.Video Thumbnails and Previews:**

Enhance the user experience with video thumbnails and previews. Create thumbnail images for each video and enable video previews when users hover over video thumbnails.

**10.Content Management and Moderation:**

Implement a content management system to allow content administrators to review and moderate uploaded videos. This helps maintain content quality and ensure compliance with your platform's policies.

**11.Monetization (Optional):**

If your platform includes premium content, integrate monetization features. Implement subscription models, pay-per-view options, or advertising solutions to generate revenue from your videos.

**12.Mobile Apps and Responsive Design:**

Ensure your platform is accessible on both desktop and mobile devices. Develop mobile apps for iOS and Android if necessary.

**13.Testing and Quality Assurance:**

Thoroughly test your platform to ensure video playback is smooth and error-free across various devices, browsers, and network conditions.

**14.Scalability and Performance Optimization:**

As your user base grows, ensure your platform can scale horizontally to handle increased traffic. Optimize performance by minimizing server response times and leveraging caching mechanisms.

**15.Legal Considerations:**

Be aware of copyright and licensing issues related to video content. Implement copyright infringement reporting and takedown procedures if necessary.

**16.User Engagement and Community Building:**

Encourage user interaction through comments, likes, shares, and social media integration.

Build a community around your platform.

**17.User Support and Feedback:**

Offer user support and feedback channels to address issues and improve the user experience continuously.

Remember to keep security a top priority throughout the development process. Secure video access, user data, and the platform as a whole to protect against potential threats and breaches.

**4.2 VIDEO UPLOAD :**

Creating a platform for users to upload movies and videos involves several steps, including setting up a server, implementing a user interface, and handling the actual video upload and storage. Below, I'll outline a simplified web-based solution using popular technologies like Node.js, Express, and MongoDB for backend, and HTML and JavaScript for the frontend. Please note that this is a high-level overview, and real-world implementations may require more features and security considerations.

Backend (Node.js with Express and MongoDB):

**1.Setting Up Dependencies:**

Start by initializing a Node.js project, installing necessary dependencies, and setting up your MongoDB database.

```bash npm init npm

install express mongoose

multer

```

**2.Create Server:**

Create an Express server and set up routes to handle file uploads.

```javascript const express

* require('express'); const mongoose = require('mongoose'); const multer = require('multer'); const app = express();
* Connect to MongoDB mongoose.connect('mongodb://localhost/video\_platfor m', { useNewUrlParser: true,

useUnifiedTopology: true });

* Define a Schema and Model for Video const videoSchema = new mongoose.Schema({ title: String, description: String, videoUrl: String,

});

const Video = mongoose.model('Video', videoSchema);

* Configure multer to handle

file uploads const storage =

multer.diskStorage({

destination: (req, file, cb) => {

cb(null, 'uploads/'); // Folder to store uploaded videos },

filename: (req, file, cb) => {

cb(null, Date.now() + '-' + file.originalname);

},

});

const upload = multer({ storage: storage });

Create an API endpoint for video uploads app.post('/upload', upload.single('video'), (req, res) =>{

const { title, description } = req.body; const videoUrl = req.file.path; const video = new Video({ title, description, videoUrl });

video.save((err) => {

if (err) {

return res.status(500).send(err);

}

return res.status(200).send('Video uploaded successfully!');

});

});

app.listen(3000, () => {

console.log('Server is running on port 3000');

});

```

**3.Frontend (HTML & JavaScript):**

Create a simple HTML form for users to upload videos.

```html

<!DOCTYPE html>

<html>

<head>

<title>Video Upload</title>

</head>

<body>

<h1>Upload Your Video</h1>

<form action="/upload" method="POST" enctype="multipart/form-data"> <label for="title">Title:</label>

<input type="text" name="title" required><br><br> <label for="description">Description:</label> <textarea name="description"></textarea><br><br> <label for="video">Choose a Video:</label> <input type="file" name="video" accept="video/\*" required><br><br>

<input type="submit" value="Upload">

</form>

</body>

</html>

```

**4.Storing and Retrieving Videos:**

Extend the backend to store video metadata in your MongoDB database and create routes to retrieve and display videos on your platform.

**5.User Authentication and Authorization:**

Implement user authentication and authorization to ensure only authorized users can upload and manage videos.

**6.Security Considerations:**

Make sure to validate and sanitize user input, handle file type and size restrictions, and protect against common security issues like Cross-Site Request Forgery (CSRF).

**7.Scalability and Performance:**

Consider using a content delivery network (CDN) for video delivery to ensure optimal performance

**8.User Experience:**

Enhance the user experience with features like video previews, thumbnail generation, and video categorization.

This is a basic outline to get you started. Depending on your platform's requirements, you may need to add more features, such as video processing, user profiles, comments, and likes. Additionally, it's crucial to ensure data privacy and security, especially when handling user-generated content.

**4.3 INTEGRATING :**

Integrating IBM Cloud Video Streaming services into your platform can help ensure smooth and high-quality video playback. IBM Video Streaming services provide a robust infrastructure for delivering video content to users. Below, I'll outline the steps to integrate IBM Cloud Video Streaming services into your platform:

**1.Create an IBM Cloud Account:**

If you don't already have one, sign up for an IBM Cloud account at [https://cloud.ibm.com/](https://cloud.ibm.com/).

**2.Provision the IBM Video Streaming Service:**

* Log in to your IBM Cloud account.
* Navigate to the IBM Cloud Catalog and find the "Video Streaming" service.
* Click on the service and follow the instructions to provision it. You may need to select a plan based on your usage.

**3.Get API Credentials:**

* Once the service is provisioned, you'll need API credentials to interact with the service programmatically. You can find these credentials in the IBM Cloud Dashboard.
* Note down the API Key, Service Instance ID, and Endpoint URL.

**4.Integrate with the Backend (Node.js):**

In your Node.js backend, use the IBM Cloud Video Streaming SDK to interact with the service.

You can use the `ibm-watson` Node.js SDK to connect to IBM services. Install it using npm: ```bash

npm install ibm-watson

```

Here's an example of how to use the SDK to create a video asset:

```javascript const IBMCloudVideo = require('ibm-watson/video');

const videoClient = new IBMCloudVideo({

apiKey: 'YOUR\_API\_KEY',

serviceInstanceID:

'YOUR\_SERVICE\_INSTANCE\_ID',

url:

'https://YOUR\_SERVICE\_INSTANC E\_URL',

});

* Create a video asset const createAssetParams = {

asset: 'asset-name', type: 'asset-type', };

videoClient.createAsset(createAssetParams)

.then(response => { console.log('Created video asset:', JSON.stringify(response, null, 2));

})

.catch(error => {

console.error('Error creating

video asset:', error);

});

```

Make sure to replace `'YOUR\_API\_KEY'`,

`'YOUR\_SERVICE\_INSTANCE\_ID'`, and

`'YOUR\_SERVICE\_INSTANCE\_URL'` with the actual credentials and

URL.

**5.Upload Videos to IBM Cloud Video Streaming:**

Use the IBM Cloud Video Streaming SDK to upload your video content to the platform. You can upload video files to your created assets. You may also want to use the `ibm-watson` SDK for this purpose.

**6.Configure Video Playback:**

Use the video assets' information provided by IBM Cloud Video Streaming to configure video playback on your platform. You can embed videos using HTML5 video players and set up adaptive streaming for different resolutions and devices.

**7. Security and Access Control:**

IBM Cloud Video Streaming services offer security features like token-based access and encryption.

Implement these features to protect your videos from unauthorized access.

**8.Monitoring and Analytics:**

IBM Cloud Video Streaming provides monitoring and analytics tools to track the performance of your videos. Set up monitoring to ensure the quality of video playback.

**9.Scalability and Content Delivery:**

IBM Cloud services often include content delivery networks (CDNs) for efficient content delivery. Make sure to configure the CDN settings to optimize video delivery to users globally.

**10.User Interface:**

Enhance the user interface of your platform to provide a seamless video-watching experience, including features like video playlists, recommendations, and user engagement analytics.

**CONCLUSION**

Remember that the specific implementation details can vary depending on your application's technology stack and requirements. IBM Cloud Video Streaming services offer flexibility and scalability, making it a powerful choice for delivering high-quality video content.

**REFERENCE:**

IBM Cloud Video Streaming, formerly known as Ustream, is a platform that allows you to stream and broadcast

video content over the internet. It provides a range of features and tools to help you set up, manage, and distribute your live or on-demand video content. Here are some key points to reference when working with IBM Cloud Video Streaming:

1. \*\*Live Streaming\*\*: You can use IBM Cloud Video Streaming to broadcast live events, webinars, product launches, and more. It supports various live streaming protocols and can handle a large audience.

2. \*\*On-Demand Video\*\*: You can also host on-demand videos, which can be used for video marketing, training, or any content that doesn't need to be streamed live.

3. \*\*Customization\*\*: The platform offers customization options for embedding video players on your website or app. You can choose different player styles and add branding elements.

4. \*\*Monetization\*\*: IBM Cloud Video Streaming provides monetization options, such as pay-per-view and subscription-based content. This is useful if you want to generate revenue from your videos.

5. \*\*Analytics\*\*: You can access detailed analytics to track the performance of your videos. This includes data on viewership, engagement, and more.

6. \*\*Security\*\*: The platform offers security features to protect your content. This includes password protection, access controls, and secure delivery.

7. \*\*Integration\*\*: IBM Cloud Video Streaming can be integrated with various content management systems (CMS) and platforms, making it easy to manage and distribute your video content.

8. \*\*Mobile and Social Streaming\*\*: You can stream to mobile devices and social media platforms, allowing you to reach a wider audience.

9. \*\*APIs\*\*: If you need to build custom solutions, IBM Cloud Video Streaming provides APIs for developers to integrate video streaming into their applications.

10. \*\*Support\*\*: IBM offers support and resources to help you get started and troubleshoot any issues you may encounter while using the platform.

Remember that the platform's features and offerings may change over time, so it's a good idea to check IBM's official website or documentation for the most up-to-date information and details on how to use IBM Cloud Video Streaming for your specific needs.