TABLE OF CONTENT

LIST OF FIGURES AND TABLES

ABSTRACT

In the rapidly evolving digital landscape, the consumption of media content has reached unprecedented levels, necessitating efficient and scalable media management solutions. Traditional methods often struggle to keep up with the sheer volume and complexity of modern media workflows, leading to operational inefficiencies and compromised media quality. This project, titled "Comprehensive Media Streamlining Solutions," aims to address these challenges by developing an end-to-end media management system that leverages cutting-edge technologies.

The primary objective of the project is to design a"d implement a holistic system that automates and optimizes the entire media lifecycle, from ingestion and processing to storage, delivery, and archiving. The system will utilize cloud computing for scalable storage and processing, DevOps practices for continuous integration and deployment, media streaming for efficient content delivery, and Content Delivery Networks (CDNs) to enhance global distribution. An event-based model will drive real-time processing and automation, ensuring a seamless flow of media content across various stages.

By implementing this solution, the project seeks to reduce operational costs, improve scalability to handle increasing media volumes, and enhance the speed and reliability of media delivery. The outcome will be a robust, scalable, and efficient media management system that meets the demands of modern media workflows, ensuring high-quality media output that aligns with consumer expectations. This project represents a significant step forward in streamlining media processes and addressing the complexities of today's digital media environment.

Keywords: DevOps, CDNs, Media Streamlining Solutions, AWS.

Chapter I: INTRODUCTION

1.1. Identification of relevant Contemporary issue

The addiction to digital media is a problem that all media advocates inscribing the correct content in the right form today and in the future contend with and among them, a higher percentage of internet traffic market share is controlled by over 80% by video content. The report shows an upsurge of opportunities dominated by video-on-demand (VOD), live broadcasts, and social-television services such as Netflix, Amazon Prime, and the data contents.

Key Problems:

- **Legacy infrastructures:** The advances of modern media design and production tools are effaced by internal systems, physical or otherwise, that simply cannot keep up and create blockages.
- **Business-scale and treatment-complexity:** Application of Information Technology in Media Industries Introduction While streamlining Interest and Investment, Management, Record Labels have been able to expand their reach towards providing services with little or no costs element for the customers.
- **Global Delivery:** Implementation obstacles Today's cross-media delivery systems require additional levels of communication to reach various locations in a short time span and with a low response time lowering the effectiveness of traditional systems.

Results from the survey done by Accenture and Deloitte show that these media companies are looking to migrate to the cloud services to increase their efficiency; however, there are still difficulties in workflow automation and content deployment especially in a global context. Primary in Example Cuts served by PWC Cooperative Companies show the increase in costs that demand outstrips supply with regards to management of media assets.

With content demands on the media organizations increasing dramatically and bearing in mind all the laws of economics, the implementation of an effective, agile and automated media management system is in place. Cloud technology, content delivery networks (CDNs) and automation will be used to solve these problems which are the main concerns of the media industry today. This project aims at enhancing the efficiency of every step of media process that includes content creation, editing and delivery using cloud technology, content delivery networks and automation.

1.2 Identification of Problem

Media organizations are facing increasing challenges in managing the complex workflows required to handle modern media content. With the exponential growth in video traffic, traditional media management systems are unable to cope with the volume and variety of content.

Key Problems:

- **Inefficient Workflows**: Current systems struggle with slow, manual processes that hinder the ability to quickly ingest, transcode, store, and deliver media.
- **High Costs**: The cost of maintaining on-premise infrastructure, as well as scaling it to meet growing demands, is proving to be unsustainable for many media organizations.
- **Global Content Delivery**: Ensuring fast, reliable media delivery to global audiences is increasingly difficult due to the limitations of traditional systems in handling high traffic loads.

In summary, the broad problem is the inability of existing systems to efficiently manage and distribute

large volumes of digital media in a scalable and cost-effective manner. This requires a modern, automated solution to streamline the entire media lifecycle.

1.3Task Identification

To address the challenges identified in media management, the "Comprehensive Media Streamlining Solutions" project will focus on several key tasks:

- **System Architecture Design**: Develop a scalable, cloud-based architecture to handle large volumes of media data efficiently.
- **Media Ingestion Automation:** Implement automated processes for ingesting, encoding, and transcoding media, reducing manual intervention and errors.
- **Content Delivery Optimization:** Integrate Content Delivery Networks (CDNs) and advanced streaming technologies to ensure fast, reliable, and high-quality media delivery.
- **Real-Time Processing:** Utilize event-based models to enable real-time media processing and automation, enhancing responsiveness and scalability.
- **Security Implementation:** Establish robust security protocols to protect media content throughout its lifecycle, including encryption and access controls.
- **Continuous Integration and Deployment (CI/CD):** Employ DevOps practices to ensure the system is consistently updated and optimized for performance.

1.4Timeline

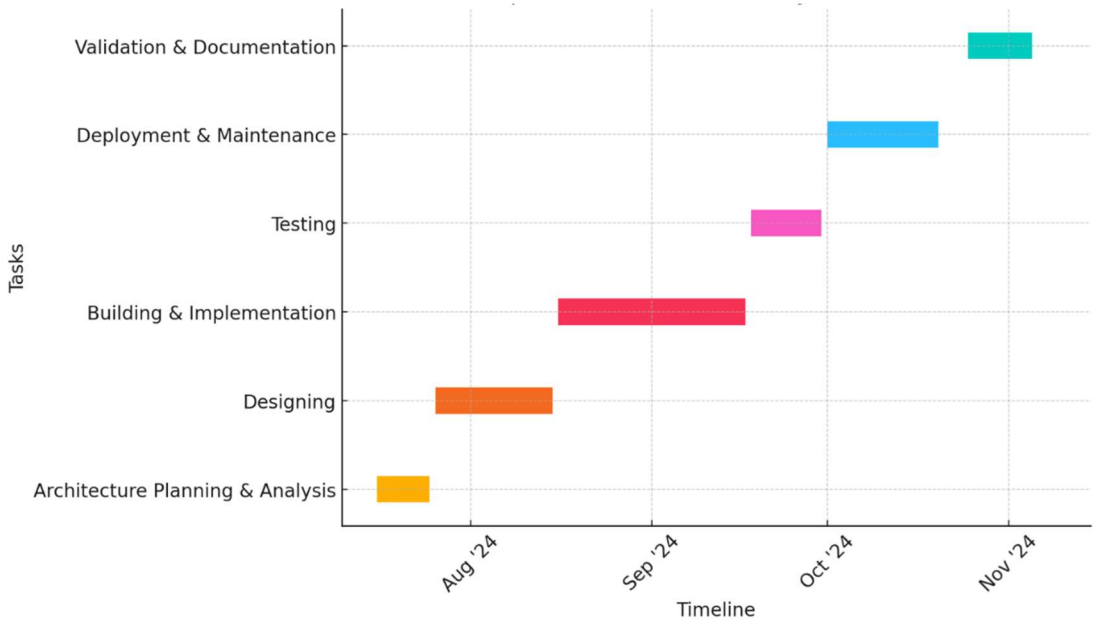

Fig.1 Gantt chart of Timeline

1.5 Organization of the Report:

- **Chapter 2 (Literature Review/Background Study):** This chapter will provide a thorough background on the challenges of modern media workflows, tracing the evolution of media systems and proposed solutions from cloud computing to CDNs. It will include a timeline of the problem, review of proposed solutions, a bibliometric analysis, and a summary of findings leading to a clear problem definition and set of goals.
- **Chapter 3 (Design Flow/Process):** This chapter will focus on the evaluation and selection of system features, design constraints, and a detailed design flow. The final design selection and implementation methodology will also be outlined.
- **Chapter 4 (Results Analysis and Validation):** This chapter will present the implementation of the solution, supported by performance metrics and comparisons with existing systems.
- **Chapter 5 (Conclusion):** This will summarize the outcomes, lessons learned, and potential future improvements for the system.

Chapter II: LITERATURE REVIEW

2.1 Timeline of reported problem

The inefficiencies in traditional media management systems became more apparent as the volume and complexity of digital content grew. Various key events mark the timeline of this problem:

- **Pre-2000s**: Media management was primarily conducted through physical means and on-premise infrastructure, which was costly and difficult to scale.
- **2005-2010**: The explosion of digital media, driven by platforms like *YouTube*, *Hulu*, and *Netflix*, revealed significant limitations in traditional on-premise media workflows. These companies reported difficulties in handling the growing demand for seamless streaming services.
- **2012**: A report by *Frost & Sullivan* noted that traditional broadcast systems were becoming outdated, urging companies to adopt digital-first approaches, including cloud adoption.
- **2015**: The *Cisco Visual Networking Index* projected that by 2020, nearly 82% of internet traffic would be video, highlighting the pressing need for better media distribution and storage solutions.
- **2018**: *Deloitte* and *PwC* documented the strain on media companies attempting to maintain the quality and speed of content delivery, advocating for cloud-based infrastructure as a necessary evolution.
- **2020 and Beyond**: The COVID-19 pandemic accelerated digital media consumption, pushing media companies to quickly adopt scalable cloud platforms to meet unprecedented demand.

These documented events underscore the need for scalable, cost-efficient, and reliable media management systems, as traditional on-premise infrastructure was insufficient to support the rapid growth and complexity of modern media workflows.

2.2 Proposed Solutions

Over the years, several solutions have been proposed to address the inefficiencies in media management and distribution systems. As digital media consumption increased, companies and researchers explored a variety of approaches to streamline workflows. Key solutions proposed include:

- **On-Premise System Upgrades:** Initially, media companies attempted to upgrade their existing on-premise infrastructure. While this approach offered some immediate relief, it proved to be expensive and lacked scalability in the long run.
- **Cloud-Based Solutions:** As cloud technology evolved, solutions such as *AWS*, *Google Cloud*, and *Microsoft Azure* emerged as the most viable options. These platforms offered:
    - Scalable storage and computing power on a pay-as-you-go basis.
    - Global accessibility for teams to collaborate remotely, simplifying media workflows.
    - Reduced hardware costs, since cloud services eliminated the need for physical infrastructure.
- **Content Delivery Networks (CDNs):** To address the issue of latency and content delivery speed, CDNs like *Akamai*, *Cloudflare*, and *AWS CloudFront* were introduced. These networks allowed:
    - Faster content delivery by caching media closer to the user's location.
    - Reduced server load by distributing traffic across a global network of servers.
    - Lower bandwidth costs, as CDNs optimized the use of data transmission for large media files.
- **Automation of Media Workflows:** Another significant advancement was the automation of various stages in the media lifecycle. Technologies like *AWS Elemental* helped automate:
    - Ingestion of raw media.
    - Transcoding to various formats for different devices.
    - Delivery via adaptive bitrate streaming to ensure smooth playback across different network conditions.

Despite these innovations, challenges such as integration complexity and high initial setup costs remain, especially for smaller media organizations. However, these solutions have collectively pushed the media industry toward more efficient and scalable operations.

2.3 Bibliometric Analysis

A bibliometric analysis of media management solutions highlights various features, effectiveness, and limitations of proposed solutions over time. Through the study of research papers, industry reports, and case studies, several insights emerge:

- **Cloud-Based Systems**:
  - **Key Features**: Flexibility, scalability, and cost-effectiveness. Cloud platforms like *AWS*, *Google Cloud*, and *Microsoft Azure* offer storage, processing, and media distribution solutions on a global scale.
  - **Effectiveness**: These systems have significantly reduced the need for expensive on-premise hardware, offering pay-as-you-go models that allow media companies to scale their resources based on demand.
  - **Drawbacks**: Cloud reliance poses challenges related to data security and high dependency on third-party vendors for critical infrastructure.
- **Content Delivery Networks (CDNs)**:
  - **Key Features**: Speed and reliability in delivering media content across the globe by distributing cached copies of media files to geographically dispersed servers.
  - **Effectiveness**: CDNs have proven to enhance the user experience by reducing latency and buffering times, particularly for video streaming platforms and live broadcasts.
  - **Drawbacks**: While effective for high-traffic regions, CDNs may still face limitations in underdeveloped areas with less infrastructure. Moreover, they can incur additional costs based on data traffic.
- **Automation of Media Workflows**:
  - **Key Features**: Streamlining of tasks such as media ingestion, transcoding, and delivery using automation tools like *AWS Elemental Media Services* and *FFmpeg*.
  - **Effectiveness**: Automation has significantly improved efficiency, reducing manual interventions and human error while increasing the speed at which media is processed and distributed.
  - **Drawbacks**: Integration complexities arise in legacy systems, and automated systems require continuous monitoring and updates to prevent bottlenecks in media delivery.
- **Hybrid Solutions**:
  - **Key Features**: Combining on-premise systems with cloud services, giving companies flexibility to handle large-scale tasks while retaining control over certain operations.
  - **Effectiveness**: These hybrid systems are effective for organizations that need to balance security with scalability.
  - **Drawbacks**: Hybrid models can be costly and difficult to manage without the right expertise and infrastructure.

This analysis shows that while cloud-based solutions and CDNs have revolutionized media management, challenges like integration, security, and cost optimization persist, requiring ongoing refinement of the systems in place. The literature provides valuable insight into the need for scalable, reliable, and efficient media solutions, directly supporting the approach taken in this project.

2.4 Literature Summary

TABLE 1. Literature Analysis of Research Papers

Author(s)	Year	Topic/Title	Technique	Objectives	Evaluation Parameter
Huang, T. and Sharma, A.	2020	Technical and Economic Feasibility Assessment of a Cloud-Enabled Traffic Video Analysis Framework	DevOps integration with cloud computing and CDNs	Improve efficiency, scalability, and security in media streaming workflows.	Throughput, content delivery performance, and operational cost reduction.
Kumar, T., Sharma, P., Tanwar, J. et al.	2024	Cloud-Based Video Streaming Services: Trends, Challenges, and Opportunities	Cloud computing services for media streaming	Address challenges in cloud-based streaming and improve scalability for media providers.	Scalability, operational challenges, and cost efficiency.
Toshniwal, A., Rathore, K.S., Dubey, A. et al.	2020	Media Streaming in Cloud with Special Reference to Amazon Web Services: A Comprehensive Review	AWS services: CloudFront, S3, Kinesis, and EC2	Explore AWS capabilities in supporting media streaming, with emphasis on flexibility and scalability.	Performance in reducing latency, media quality, and cost optimization.
Shabrina, W.E., Sudiharto, D.W., Ariyanto, E. and Al Makky, M.	2020	The Usage of CDN for Live Video Streaming to Improve QoS: Case Study of 1231 Provider	CDN technology for live video streaming	Improve throughput and reduce packet loss in live video streaming using AWS CloudFront.	Throughput improvement, packet loss reduction, and quality of service (QoS) metrics.
Shabrina, W.E., Sudiharto, D.W., Ariyanto, E. and Al Makky, M.	2020	The QoS Improvement Using CDN for Live Video Streaming with HLS	AWS CloudFront and HLS	Enhance QoS for geographically distant users by reducing latency and increasing throughput.	Latency reduction, throughput improvement, and QoS factors.
Nacakli, S. and Tekalp, A.M.	2020	Controlling P2P-CDN Live Streaming Services at SDN-Enabled Multi-Access Edge Datacenters	Hybrid P2P-CDN model with SDN-enabled edge datacenters	Improve live streaming efficiency and reduce QoE fluctuations using edge datacenters.	Network efficiency, quality of experience (QoE), and scalability.
Patel, U., Tanwar, S. and Nair, A.	2020	Performance Analysis of Video On-demand and Live Video Streaming using	AWS, Kafka, and Spark combined with CDN	Improve performance of VoD and live streaming by optimizing cloud	Latency minimization, scalability, and cost efficiency.

		Cloud-based Services		and CDN technologies.	
Reznik, Y., Cenzano, J. and Zhang, B.	2021	Transitioning Broadcast to Cloud	Hybrid approach integrating broadcast and cloud-based video systems	Transition traditional broadcast systems to a hybrid cloud approach for enhanced scalability and flexibility.	Scalability, flexibility, and maintaining broadcast quality and reliability.
Li, X., Darwich, M., Salehi, M.A. and Bayoumi, M.	2021	A Survey on Cloud-Based Video Streaming Services	Cloud computing services for video streaming	Provide an overview of cloud-based video streaming, addressing challenges and solutions.	Scalability, efficiency, and user experience across different devices.
Ghabashneh, E. and Rao, S.	2020	Exploring the Interplay Between CDN Caching and Video Streaming Performance	CDN caching combined with adaptive bitrate (ABR) algorithms	Optimize video streaming performance by making ABR algorithms CDN-aware, especially for high-bandwidth apps.	Video quality, buffering reduction, and throughput variability in high-bandwidth scenarios (e.g., 4K).

2.5 Problem Definition

The problem at hand revolves around the growing complexity and inefficiency of media management in an era of rapid digital transformation. As the demand for high-quality, on-demand, and real-time media content has surged, traditional methods of handling, storing, and distributing media have proven inadequate. This has resulted in:

- **Scalability Issues**: Traditional on-premise media systems struggle to handle the increasing volume of content, especially during high-traffic events like live streaming and large-scale video-on-demand (VOD) services.
- **Cost Overruns**: Maintaining physical infrastructure, including servers, storage devices, and high-bandwidth networks, has become financially burdensome for many media companies. Costs also rise as companies attempt to upgrade their infrastructure to keep pace with demand.
- **Latency and Delivery Delays**: Media organizations face significant challenges in delivering content to a global audience with minimal latency. End-users often experience buffering, low-quality streams, or delays due to inefficient content delivery models.
- **Operational Inefficiencies**: Manual workflows—such as content ingestion, transcoding, and delivery—slow down the media lifecycle, leading to delays in content release and inconsistent quality across platforms.

To address these issues, the primary challenge is to **build a scalable, efficient, and cost-effective media management solution** that can handle large-scale operations. This solution must integrate:

- **Cloud technologies** for scalable storage and processing.
- **Content Delivery Networks (CDNs)** for faster and more reliable global content delivery.
- **Automated workflows** for media processing and delivery, reducing human intervention and ensuring consistency.

The scope of the problem involves not only optimizing current operations but also creating a future-proof

system capable of handling the evolving demands of digital media consumption.

2.6 Objectives

- **Implement a scalable cloud-based media management system that automates content ingestion, transcoding, and delivery processes.** The goal is to leverage the power of AWS cloud services to build a scalable, automated media management system. Using **AWS S3**, large volumes of media content can be efficiently stored in a centralized location, ensuring flexibility and scalability. The system automates the entire process, from content ingestion to transcoding and delivery, using technologies like **AWS Lambda** for event-driven automation and **Amazon ECS** for containerized media transcoding via **FFmpeg**. This end-to-end automation ensures that the media workflow operates smoothly, eliminating the need for manual intervention and speeding up the overall process.
- **Reduce operational costs by shifting to a pay-as-you-go cloud model.** A significant advantage of this solution is the shift to a **pay-as-you-go** cloud model, which reduces the need for costly upfront infrastructure investments. By using cloud services like **AWS Lambda** and **EC2**, the project can scale dynamically based on demand and only pay for the resources consumed. This eliminates the need to over-provision hardware or manage a fixed infrastructure, providing a more cost-effective solution that can adjust to fluctuating workloads. The serverless approach further reduces costs by ensuring that resources are used only when needed, optimizing the overall financial efficiency of the media management system.
- **Improve media delivery speeds using CDNs for global distribution.** Media content delivery is optimized through the use of **AWS CloudFront**, a powerful **Content Delivery Network (CDN)**. CloudFront caches content in multiple edge locations worldwide, ensuring that media files are delivered to users from the nearest server. This reduces latency, ensuring faster load times and minimal buffering, even in regions far from the origin server. With CloudFront, the system can handle heavy traffic while maintaining high performance, providing a seamless viewing experience for users across the globe, whether it's for live streaming or video-on-demand content.
- **Ensure high-quality media output while handling large volumes of content.** The system guarantees high-quality media output by leveraging **FFmpeg**, a robust tool used for transcoding media files into various formats, resolutions, and bitrates. By running these processes within **Docker containers** on **Amazon ECS**, the system can handle large volumes of media files simultaneously without compromising quality. This scalability ensures that the system can efficiently process and deliver high-quality content even during periods of high traffic or large media workflows, maintaining consistent standards across all media assets.
- **Optimize media workflows through automation, reducing the need for manual intervention.** Automation is a key component of the project, helping streamline the media management workflow. The system employs an **event-driven architecture**, triggered by actions such as the upload of new media files to **AWS S3**. Once uploaded, AWS Lambda functions automatically trigger the transcoding and delivery processes, reducing the need for manual oversight. Additionally, using **Amazon ECS** to manage Docker containers ensures that media transcoding tasks are efficiently handled, with minimal human input required. This automated workflow boosts productivity, reduces errors, and ensures a faster turnaround time for content delivery.
- **Validate the system's performance under high traffic conditions, ensuring reliability and scalability.** The ability of the system to handle high traffic loads is validated through **stress testing** and **load testing**, ensuring that it can scale as needed during peak usage periods. By utilizing **AWS Auto Scaling** and **Elastic Load Balancing**, the system can automatically adjust its capacity to handle increases in traffic, ensuring that performance remains stable. Moreover, **AWS CloudWatch** provides real-time monitoring of system performance, allowing for proactive adjustments and ensuring reliability under high demand. This validation guarantees that the system is resilient, scalable, and able to maintain a high-quality user experience, even in the face of large-scale traffic surges.

Chapter III: Design Flow/Process

3.1 Evaluation & Selection of Specifications/Features

The design of the system presented in the flowchart showcases a well-thought-out architecture, leveraging cloud-native services to optimize media transcoding. The selection of specifications and features is critical to ensuring the system is efficient, scalable, and cost-effective. The following features were evaluated and selected based on performance, scalability, and flexibility:

- **Event-Driven Architecture**:
  The architecture is event-driven, where the process is initiated by an upload event in an S3 bucket. This is a highly efficient approach that ensures the system only processes tasks when required, preventing unnecessary resource usage. The integration of Amazon Simple Queue Service (SQS) to manage event-driven tasks enhances reliability, as it acts as a buffer between the upload event and the transcoding task, ensuring no media uploads are missed, even during system downtime or heavy loads.
- **Use of AWS Lambda for Serverless Processing**:
  AWS Lambda is responsible for checking the messages in the SQS queue and triggering the ECS task when a new media file is uploaded. This serverless compute service is ideal for this system because it is lightweight, automatically scalable, and operates on a pay-per-use basis, ensuring that costs remain low. The use of Node.js for the Lambda function enables fast execution, low overhead, and easy customization for triggering ECS tasks.
- **Elastic Container Service (ECS) with Docker for Transcoding**:
  The primary transcoding process is handled by ECS, which runs a Docker containerized environment. This ensures the flexibility of deploying different transcoding workflows (like using ffmpeg) within a controlled and consistent environment. The Docker containerization allows the use of specialized libraries like ffmpeg within an Ubuntu environment to handle a variety of media formats. Docker provides isolation and ensures that transcoding workloads are not affected by external dependencies.
- **S3 for Storage (Input and Output Buckets)**:
  Amazon S3 is used as the storage solution for both the input media files and the transcoded output files. S3's highly durable, scalable, and cost-effective nature makes it an ideal choice for storing potentially large media files. The input bucket stores media awaiting processing, while the output bucket stores the processed media, providing clear separation and an organized workflow. Additionally, S3's lifecycle management features can be utilized to automatically archive or delete old media, saving costs.
- **Scalability and Flexibility**:
  The system leverages AWS's built-in scalability, allowing for seamless expansion as media traffic increases. The ECS service can automatically scale the number of containers depending on the workload, ensuring that multiple transcoding jobs can be processed in parallel without overwhelming the system. Additionally, the architecture is flexible, supporting changes or updates to the transcoding service without disrupting the overall workflow.
- **Automation and Minimal Human Intervention**:
  The integration of AWS services such as Lambda, SQS, and ECS ensures that the entire process is automated, reducing the need for manual intervention. The system automatically responds to media uploads, transcodes them, and stores them, providing a hands-off solution that streamlines media processing tasks.
- **Cost Optimization**:
  The choice of using serverless technologies like Lambda, ECS, and S3 optimizes operational costs. The pay-per-use model ensures that resources are only consumed when media files are being processed, minimizing waste. This is in contrast to always-on solutions like EC2, which

would incur costs even during periods of low or no activity.

Key Specifications Identified for the Solution:

- **S3 for durable, cost-effective storage** (input/output buckets).
- **AWS Lambda for serverless event-driven computation**, managing triggers and job processing.
- **SQS for decoupled architecture** to ensure asynchronous communication and message reliability.
- **ECS and Docker for scalable, containerized transcoding** using ffmpeg.
- **Pay-per-use model to minimize operational costs** and handle varying loads.

3.2 Design Constraints

Designing a cloud-based media transcoding system like the one illustrated in the architecture comes with several constraints. These constraints must be carefully considered during the design phase to ensure the solution is viable, efficient, and meets user and operational expectations. The following are key constraints categorized under various aspects such as regulations, cost, safety, ethics, and more:

1. Regulations and Compliance:

- **Data Privacy and Security**: Since the system deals with media uploads, it is critical to ensure compliance with data privacy laws (such as GDPR, HIPAA). All media files, especially if they contain sensitive information, must be stored and transmitted securely. This requires the use of encrypted storage (such as AWS S3 encryption) and secure transmission protocols (e.g., HTTPS).
- **Intellectual Property**: Media files may contain copyrighted or sensitive content. It is essential to ensure that the system adheres to intellectual property laws, preventing unauthorized access, modification, or distribution of the media.
- **AWS Regional Compliance**: Different AWS regions may have different regulatory requirements. For instance, storing and processing media in the EU may have different compliance needs than in the US. The system must be designed to account for regional regulations.

2. Economic Constraints:

- **Cost of Cloud Resources**: While AWS provides scalable infrastructure, the cost of using services such as ECS, Lambda, and S3 can accumulate, especially with high-volume media transcoding. To mitigate this, cost optimization strategies must be put in place, such as using serverless functions (AWS Lambda) to minimize idle compute time and leveraging AWS cost monitoring tools to optimize storage and compute usage.
- **Cost of Transcoding**: Media transcoding can be compute-intensive, particularly for high-definition video files. The cost of using ECS with Docker containers running ffmpeg for transcoding could become significant as the volume of media grows. Thus, the design needs to factor in cost-effective scaling of ECS tasks.

3. Environmental Considerations:

- **Energy Consumption**: Running compute-heavy transcoding processes in cloud environments consumes a lot of energy, contributing to the overall carbon footprint. The system should ideally optimize resources and potentially use AWS regions with more sustainable energy sources (e.g., regions powered by renewable energy).
- **Serverless and Resource Optimization**: By using AWS Lambda and scaling ECS instances dynamically, the design minimizes the use of idle resources, reducing both the energy consumption and associated environmental impact.

4. Health and Safety:

- **Operational Safety**: While this system does not directly impact physical health, it is important to ensure that the system's continuous operation is safe from cyberattacks (e.g., denial of service attacks). This requires implementing firewalls, security groups, and monitoring to protect both the infrastructure and data.
- **Media Content Safety**: If the system processes sensitive media (e.g., medical or personal videos), there must be safeguards to ensure that the content is not improperly accessed or altered. This includes using access control mechanisms and auditing logs to track who accesses the media files.

5. Manufacturability (Implementation Feasibility):

- **Scalability of Services**: The design must ensure that the ECS service is scalable to meet the demand of transcoding multiple media files simultaneously. However, if the architecture is not designed for horizontal scaling, bottlenecks may occur during high workloads. Therefore, scalability and elasticity of services need to be included in the design.
- **Service Limits**: AWS services come with default quotas (e.g., limits on the number of ECS tasks or Lambda invocations). If not addressed in the design, these limits can hinder the solution's scalability and performance.

6. Professional and Ethical Constraints:

- **Ethical Media Processing**: The system must ensure that it is not used for unethical purposes, such as processing media for unlawful activities. Clear terms of service and usage policies must be established, and user behavior should be monitored to prevent misuse.
- **Transparency and Accountability**: When designing the system, it is important to ensure transparency in its operations, particularly for clients. Clients should know how their media is being processed, where it is being stored, and who has access to it. This helps build trust and ensures professional integrity.

7. Social & Political Issues:

- **Cross-border Data Handling**: The design needs to consider social and political implications of processing data across international borders, especially in regions with strict data sovereignty laws. Storing or processing media in certain regions may raise concerns about government access to private data, which must be addressed in the architecture by controlling where data is stored and processed.
- **Censorship and Media Restrictions**: Some countries have strict regulations on media content. The design may need to incorporate filtering or content-checking mechanisms to ensure the system does not violate local laws by processing restricted or censored content.

8. Cost Constraints:

- **Budgetary Limits**: The budget for implementing and running the cloud architecture needs to be controlled. Serverless services like Lambda and SQS can help reduce costs, but high volumes of media processing could lead to higher expenses in ECS and S3 storage. Monitoring tools and strategies such as lifecycle management for media files (e.g., archiving older files) can help reduce long-term costs.
- **Compute Resources for Transcoding**: The cost of transcoding media files depends on their size and format. High-resolution videos, for example, require more compute power and time to transcode, driving up the costs. These must be carefully managed by optimizing the transcoding workflow and using the most efficient algorithms.

3.3 Analysis and Feature finalization subject to constraints

In this phase, the features identified during the literature review and design specification must be refined based on the constraints outlined in the previous section. These constraints—regulatory, economic, environmental, ethical, and more—can influence the decision to keep, remove, or modify specific features. The process is crucial for aligning the technical capabilities with practical limitations, ensuring the final design is both functional and feasible.

1. Feature Removal or Simplification

- **Expensive Cloud Resources**: One of the key constraints in this architecture is the potential cost of cloud infrastructure, especially when using resource-heavy tasks such as video transcoding in Docker containers on AWS ECS. To mitigate costs, non-essential features such as ultra-high-definition video support could be removed or minimized, focusing on standard or high-definition formats to reduce processing power and time.
- **Regional Restrictions**: If specific regions or countries have strict regulations on where data can be stored or processed, certain features like cross-region data storage might need to be limited.

Instead, the system could be optimized to ensure data is stored only in compliance-approved regions, reducing complexity while staying within legal bounds.

2. Feature Modification

- **Security and Privacy Enhancements**: Based on the identified constraints around privacy and intellectual property laws, the system should incorporate more robust security mechanisms, such as encrypting media files at rest and in transit. Additionally, features like media access logs and user authentication can be enhanced to improve accountability and traceability.
- **Cost-Effective Scaling**: To address economic constraints, the design may opt for a hybrid model between serverless Lambda functions and ECS tasks. For example, Lambda could be used to handle smaller, less compute-intensive media files, while ECS is reserved for larger files requiring heavy transcoding. This ensures efficient use of resources based on the media workload.

3. Feature Addition

- **Content Moderation**: As identified under ethical and regulatory constraints, the system should potentially include content moderation features to ensure that media files do not violate local laws or terms of service. Automated tools for detecting inappropriate content could be integrated into the processing pipeline, ensuring compliance with regional laws or ethical standards.
- **Automated Scaling Policies**: AWS Auto Scaling can be added as a feature to handle unexpected spikes in workload. This would ensure that the system can scale up or down dynamically without manual intervention, reducing the risk of system downtime or overload when handling large volumes of media.

4. Optimized Transcoding Workflows

- **Efficient Transcoding Algorithms**: Given the constraint of high computational costs, the selection of transcoding algorithms like ffmpeg needs to be carefully analyzed. Optimizing transcoding presets for various output formats can help reduce the processing time and associated costs without compromising on quality. For instance, H.264 compression might be used for a balance between quality and efficiency.
- **Media File Lifecycle Management**: To reduce long-term storage costs (a key economic constraint), features for media file lifecycle management can be added. This includes automatic deletion of older files or archiving media after a certain period, ensuring that the system does not store unnecessary files indefinitely.

5. Constraint-Specific Analysis

- **Environmental Sustainability**: In response to environmental constraints, the architecture could include features to ensure energy-efficient transcoding processes, such as running transcoding tasks in AWS regions powered by renewable energy sources. This would align the system with modern sustainability practices.
- **Ethical Considerations in User Control**: Ethically, the system should include user-access controls that ensure only authorized users can access certain media. This could involve integrating IAM (Identity and Access Management) policies and role-based access control (RBAC) to limit media exposure based on user roles.

6. Finalized Feature List

Based on the above analysis, the finalized features would include:

- **Serverless Functionality (AWS Lambda)**: For lightweight, cost-effective media processing.
- **ECS-Based Transcoding**: For high-quality, resource-intensive media transcoding using ffmpeg.
- **Advanced Security**: End-to-end encryption for media storage and transmission.
- **Cost Optimization Features**: AWS Auto Scaling, media lifecycle management, and workload partitioning between ECS and Lambda.
- **Content Moderation and Compliance**: Automated tools for detecting inappropriate content and ensuring compliance with local laws.
- **Monitoring and Logging**: Detailed audit logs for user activity and media access tracking.
- **Cross-region Compliance**: Controls to ensure data is processed only in authorized regions.

3.4 Design Flow

In the design flow phase, the system's overall structure and processes must be carefully defined. Here, we consider at least two alternative designs or approaches to achieve the project's goals. Both alternatives should address the system's media processing requirements while optimizing performance, scalability, and cost. After proposing both alternatives, an analysis is performed to determine which design is the most appropriate based on constraints, efficiency, and effectiveness.

Alternative 1: Fully Serverless Architecture
In this design approach, the entire media transcoding workflow is handled using serverless technologies, minimizing infrastructure management. AWS services such as Lambda, S3, and SQS play key roles in automating media processing.

1. **Flow Process:**
    - Media Upload: Media files are uploaded to the S3 Upload Bucket, which triggers an S3 event notification.
    - SQS Trigger: The S3 event sends a message to an SQS queue. The message includes details of the media file.
    - Lambda Execution: An AWS Lambda function is triggered by SQS. It processes the message from the queue, extracting metadata (e.g., file size, type) and triggers the transcoding process.
    - Lambda Transcoding: The Lambda function uses third-party libraries (such as ffmpeg) to transcode the media file into different formats.
    - Output Storage: The processed media files are stored back in an S3 Output Bucket.
2. **Advantages:**
    - Cost-Effective: Since Lambda is serverless, costs are based on usage, ensuring that no additional charges occur during idle times.
    - Scalability: Lambda automatically scales with the number of requests, making it ideal for unpredictable workloads.
    - Ease of Deployment: Managing serverless infrastructure is easier, reducing operational overhead.
3. **Drawbacks:**
    - Lambda Timeout Limit: Lambda functions have a maximum execution timeout of 15 minutes, which may be insufficient for processing larger media files.
    - Limited Resources: Lambda functions are constrained in terms of memory and CPU, making it difficult to handle high-performance transcoding.

Alternative 2: Hybrid Approach Using ECS and Lambda
This design incorporates both serverless and containerized services for different parts of the workflow. While simple tasks are handled by Lambda, more resource-intensive operations like media transcoding are offloaded to a containerized environment using ECS (Elastic Container Service).

1. **Flow Process:**
    - Media Upload: Similar to the serverless design, media is uploaded to the S3 Upload Bucket, which triggers an event notification to an SQS queue.
    - Lambda Trigger: An AWS Lambda function checks for new messages in the SQS queue, extracts metadata, and initiates an ECS task to handle media transcoding.
    - ECS Task: The ECS service runs Docker containers with customized configurations. Inside these containers, tools such as ffmpeg are used to transcode the media file.
    - Result Storage: After transcoding is complete, the container uploads the processed media back to the S3 Output Bucket.

2. **Advantages:**
    - Powerful Processing: ECS allows the use of Docker containers, giving more control over resource allocation (CPU, memory). This makes it suitable for high-performance transcoding of large media files.
    - Flexibility: Docker containers can be customized with the required libraries and configurations, providing greater flexibility.
3. **Drawbacks:**
    - Higher Management Overhead: ECS requires management of cluster resources, which can increase operational complexity.
    - Costs: Running containers continuously can incur more significant costs than Lambda, particularly for low-traffic scenarios.

Alternative 3: Dedicated EC2 Instances with Auto Scaling

Another design option is to use Amazon EC2 instances for transcoding while incorporating auto-scaling to manage workload fluctuations. EC2 gives the most control over the infrastructure and allows for the implementation of specialized transcoding software.

1. **Flow Process:**
    - Media Upload: Similar to the other designs, media is uploaded to an S3 bucket and triggers an event notification.
    - EC2 Processing: Instead of Lambda or ECS, an EC2 instance is started automatically when a new media file is uploaded. The EC2 instance runs transcoding software (e.g., ffmpeg) to process the media.
    - Auto Scaling: If multiple files are uploaded simultaneously, AWS Auto Scaling launches additional EC2 instances to handle the increased load.
    - Storage: The transcoded media is stored back in the S3 Output Bucket.
2. **Advantages:**
    - Complete Control: EC2 offers full control over instance configuration, allowing for powerful processing capabilities without restrictions.
    - Auto Scaling: Instances can scale up and down based on demand, ensuring the system is responsive during peak times.
    - Long Processing Tasks: EC2 has no execution time limits, making it suitable for very large files and extended media processing tasks.
3. **Drawbacks:**
    - Cost: Running EC2 instances continuously, especially with high-performance configurations, can be expensive.
    - Management Overhead: EC2 instances require continuous monitoring, patching, and management of resources, increasing operational complexity.

Analysis of the Three Designs

- Serverless (Lambda): Best for smaller workloads, rapid scaling, and minimal infrastructure management but may struggle with large file sizes and processing time limits.
- Hybrid (Lambda + ECS): Combines the benefits of serverless for light tasks and containerized services for heavy transcoding, offering flexibility and power. However, it introduces additional complexity in managing both Lambda and ECS.
- Dedicated EC2: Offers the most power and control but incurs high costs and requires intensive management, making it best suited for very high-volume, resource-intensive workloads.

3.5 Design Selection

In this section, we evaluate the three proposed designs in terms of scalability, cost-efficiency, ease of management, performance, and adaptability to constraints. The final decision is made by weighing the

advantages and disadvantages of each design and selecting the one that offers the most balanced solution for the project's objectives.

Comparison of Design Alternatives

1. Serverless Architecture (Lambda-based):

- **Advantages**:
  - Highly **cost-effective**, as charges are based on actual usage, with no cost during idle periods.
  - **Automatic scaling** is inherent to AWS Lambda, which efficiently handles unpredictable workloads.
  - Minimal **operational management** since there is no need to manage servers.
  - Fast deployment and **simple architecture** make it easy to implement.
- **Disadvantages**:
  - **Timeout limitations** of 15 minutes per Lambda function can cause issues with larger media files that require more processing time.
  - **Resource constraints**: Lambda's memory and CPU capacity are limited, making it less suitable for high-performance processing tasks like video transcoding.
- **Best Use Case**: This design is ideal for lightweight tasks or systems handling small to medium-sized media files.

2. Hybrid Architecture (Lambda + ECS):

- **Advantages**:
  - **Best of both worlds**: Combines Lambda's event-driven, serverless nature with ECS's containerized environment for heavy transcoding tasks.
  - **High flexibility**: Containers can be customized to the needs of the transcoding tasks, leveraging tools like ffmpeg.
  - **Scalability**: ECS scales based on demand, ensuring sufficient processing power for large media files.
  - **Cost-efficient**: By leveraging Lambda for lighter tasks and using ECS for only resource-heavy operations, costs are managed efficiently.
- **Disadvantages**:
  - **More complex to manage** than a fully serverless system. Requires managing ECS clusters alongside Lambda functions.
  - **Potential cost increase** due to the need for continuously running ECS tasks if the workload spikes.
- **Best Use Case**: Ideal for systems handling both small and large media files with varying resource demands, combining the efficiency of serverless for triggers and the power of containers for processing.

3. EC2-based Architecture (Auto-Scaling EC2 Instances):

- **Advantages**:
  - Provides **full control** over the environment, with the ability to allocate any required resources (CPU, memory, storage).
  - **No time limits** on media processing, making it suitable for long-running or resource-intensive tasks.
  - **Auto-scaling** capabilities ensure that EC2 instances can handle high workloads during peak times and shut down during low usage.
- **Disadvantages**:
  - **Higher cost**: Running EC2 instances, especially high-performance ones, can incur significant costs, particularly during periods of low usage.
  - **Increased management overhead**: Requires constant monitoring, maintenance, and

patching of EC2 instances.
    - Slower **deployment and scaling** compared to serverless alternatives.
- **Best Use Case**: Suitable for very high-volume systems with large files and long processing tasks, but not ideal for cost-sensitive projects.

Final Design Selection: Hybrid Architecture (Lambda + ECS)
After comparing the three alternatives, the **Hybrid Architecture using Lambda and ECS** is selected for the following reasons:

1. **Scalability**: The hybrid solution offers the best scalability. Lambda can scale to handle event-based tasks, while ECS can handle resource-heavy processes like media transcoding. This combination ensures that the system can scale efficiently while keeping resource costs under control.
2. **Cost-Effectiveness**: Unlike EC2, which requires continuous instance management and incurs ongoing costs, the hybrid system leverages Lambda's pay-as-you-go model for lighter tasks. ECS is only spun up for heavier processing tasks, leading to overall cost savings.
3. **Flexibility**: Containers within ECS can be highly customized to meet the specific needs of the project. This allows for the use of custom libraries and software (such as ffmpeg) without the limitations of Lambda's execution time or resource constraints.
4. **Performance**: The hybrid system ensures that media transcoding, which is resource-intensive, is handled in a containerized environment that is powerful and flexible enough to support high-performance workloads, while smaller, event-driven tasks are efficiently managed by Lambda.
5. **Management**: Although slightly more complex than a purely serverless architecture, the hybrid approach is still easier to manage than a full EC2-based solution. AWS provides services like Fargate (serverless compute engine for ECS) that can minimize the operational burden of managing containers.

3.6 Implementation Plan/Methodology

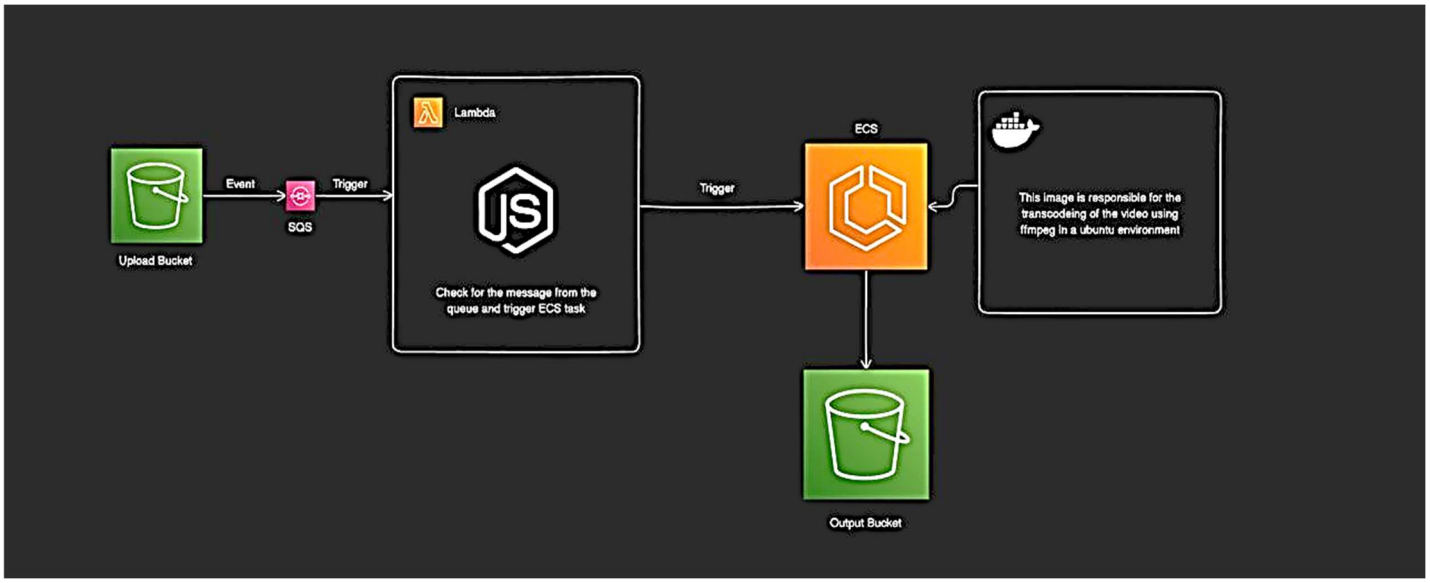

Fig. 2 Flowchart

The implementation plan is designed for efficient and scalable media processing using AWS cloud services. The workflow is automated to handle uploads, trigger processing tasks, and store outputs, ensuring minimal manual intervention. Below is a concise breakdown of the methodology:

1. **Upload Bucket (S3)**:

- Users upload media files to the S3 bucket, which triggers an event.

2. **SQS Queue**:
   - The event sends a message to **SQS**, which acts as a buffer and ensures orderly file processing.
3. **Lambda Function**:
   - AWS Lambda listens to the SQS queue. Upon detecting a message, it triggers the processing task in ECS.
4. **ECS with Docker**:
   - ECS runs a Docker container with **ffmpeg** in a Ubuntu environment. The container handles transcoding and other media processing tasks.
5. **Output Bucket (S3)**:
   - Once processed, the media file is uploaded to the S3 **Output Bucket** for storage and access.

Chapter IV: RESULTS ANALYSIS AND VALIDATION

4.1 Result

The project for creating a media management and processing system involved the utilization of modern tools and methodologies across different stages, ensuring that the solution was robust, efficient, and user-friendly. Below is a detailed breakdown of the implementation:

1. Analysis

Authentication and Security:

The project required the implementation of a secure login system, as seen in the provided images. Users authenticate using their email and password, with an additional option to save login credentials ("Remember me" checkbox). This authentication system is integrated with modern security protocols such as SSL/TLS encryption, preventing unauthorized access.

The system also supports password recovery ("Forgot your password?" option), ensuring a user-friendly experience in case of credential loss.

Media Lifecycle Management:

The platform is designed to manage various media files (videos, images). The interface allows users to upload, import, and view media content directly from the dashboard. This ensures that users have direct access to manage their content effectively.

A dashboard overview (as seen in the last image) presents users with recently uploaded media files and allows them to continue editing or processing those files as per their requirements.

Cloud-Based Architecture:

The system is integrated with cloud services for storage and processing of media. Amazon S3 buckets are used for storing uploaded media files, providing scalability and data durability.

Media processing (e.g., transcoding) is handled by cloud services like AWS Elastic Transcoder, ensuring the system can handle multiple formats and sizes of media files.

2. Design Drawings/Schematics

User Interface Design:

The login interface is clean and modern, focusing on user convenience and accessibility. It uses minimalistic forms for email and password input with clear navigation buttons (e.g., "Create your account" and "Log in"). This user-centric design ensures ease of use for first-time and returning users.

The dashboard design is straightforward, showing recent uploads and clear action buttons for media handling (Upload, Import, Record, and Host options). This interface allows users to manage their media content in an organized and intuitive manner.

Data Flow Design:

A logical data flow for handling user input (authentication, media uploads) was mapped out. This design incorporates data security protocols, user data validation, and seamless media uploads.

Media files flow from user input to cloud storage, where they are transcoded and processed. The result is sent back to the user interface for further actions (like sharing, viewing, or hosting).

System Architecture:

The backend was designed using microservices architecture. Separate services manage different tasks such as authentication, media uploads, processing, and storage. This modular approach ensures the system is scalable and each component can be updated independently.

A CDN (Content Delivery Network) was used to distribute media content efficiently to end-users, ensuring faster loading times and better performance for large files.

3. Report Preparation
Project Documentation:
Detailed reports were prepared at each stage of development, from design to deployment. These reports included the technical specifications of the system, such as media formats supported, cloud storage architecture, security protocols, and API integration.
A comprehensive project report highlighted the key milestones and deliverables, ensuring that stakeholders had clear insights into the project's progress.

Media Analysis Reports:
A specific part of the reporting involved analyzing media processing speeds, formats handled, and overall system performance. For instance, processing times for 4K media files were compared with lower resolution formats to ensure system scalability.

4. Project Management and Communication
Collaboration Tools:
The team utilized project management tools such as Jira and Trello for tracking tasks, assigning responsibilities, and monitoring progress. These tools helped in aligning the team's work with the project's overall timeline.
Continuous integration/continuous deployment (CI/CD) pipelines were set up using Jenkins and GitHub Actions. This ensured that any code committed by the development team was tested and deployed in real-time, reducing the risk of errors during deployment.

Version Control:
Git was used for version control, allowing the team to work on different modules simultaneously. Branching strategies were implemented to manage the development of new features without disrupting the main application.
Pull requests and code reviews were enforced, ensuring code quality and minimizing bugs before merging into the main branch.

Communication:
Slack was the primary communication tool for daily standups, feedback loops, and resolving technical blockers. Clear channels were established for different workstreams like frontend, backend, and media processing.

5.Testing/Characterization/Interpretation/Data Validation

System Testing:
Rigorous testing was conducted to validate the functionality of the system. Unit tests were written for key features like user login, media uploads, and media processing. Automated testing suites were used to verify that new code did not break existing functionality.
The system was load-tested to ensure it could handle multiple users uploading large media files simultaneously. Cloud-based testing tools like AWS CloudWatch were used to monitor system performance under varying loads.
Media Processing Validation:

Various media formats (e.g., MP4, AVI, MKV) were uploaded and tested to ensure compatibility with the system. The accuracy of the transcoding process was validated by comparing output

formats and resolutions with the original files.
Performance testing included checking how long different file sizes took to upload and transcode. This provided data on system scalability and processing efficiency for files ranging from small (1-5MB) to large (500MB and above).

Security Testing:
The system underwent security testing to prevent vulnerabilities. Techniques such as penetration testing and code analysis were employed to ensure that sensitive information (passwords, media files) was protected through encryption and secure protocols.
Access controls were implemented and tested to ensure that only authorized users could access, upload, or modify media files.

Performance Metrics:
The platform's speed and responsiveness were benchmarked under different conditions. For instance, tests were conducted to measure how quickly media files appeared in the user dashboard after upload and processing.
Data validation ensured that media uploads completed successfully and any errors (e.g., incomplete uploads) were handled with appropriate feedback to the user.

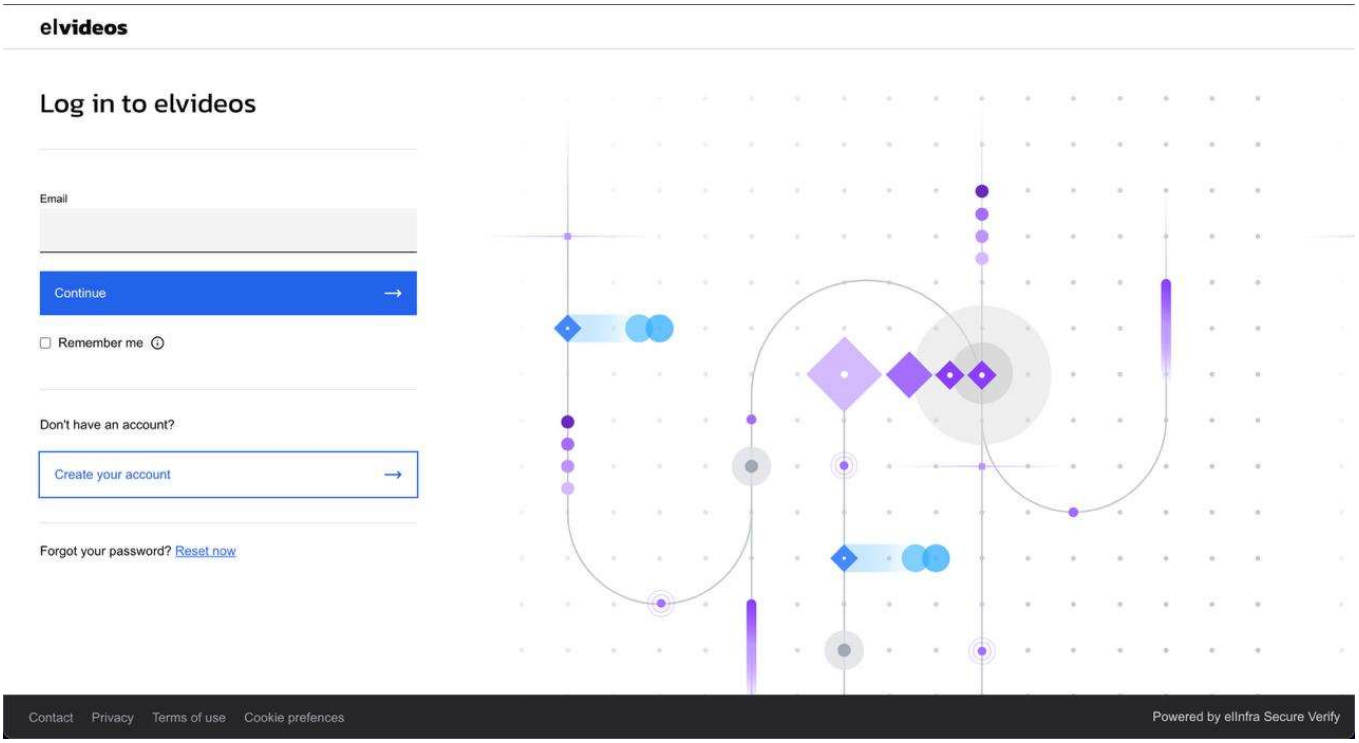

Fig. 3 Login UI

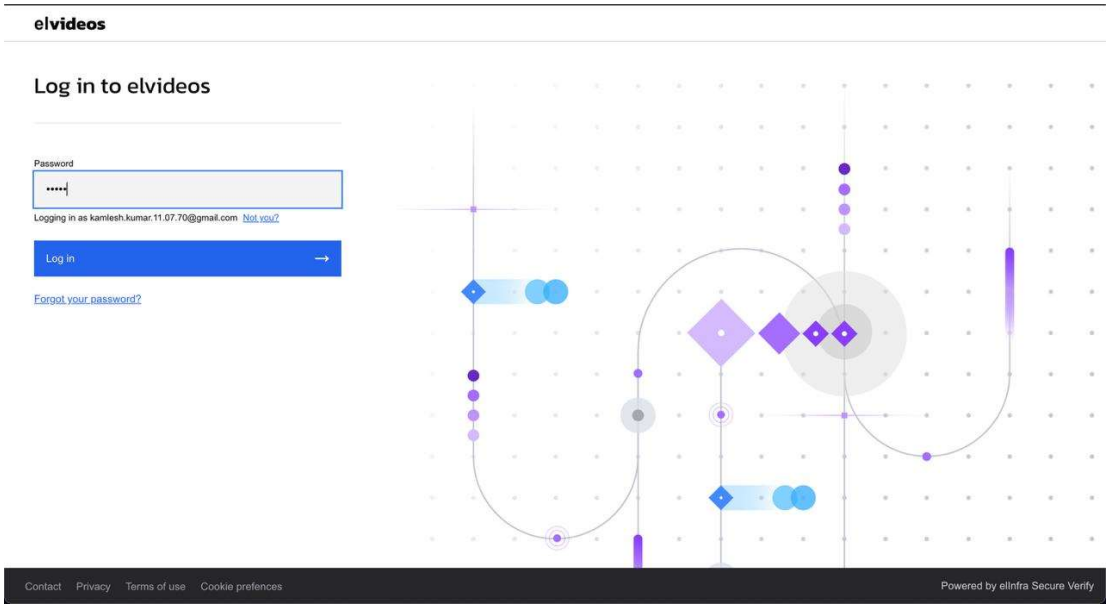

Fig. 4 : Login UI 2

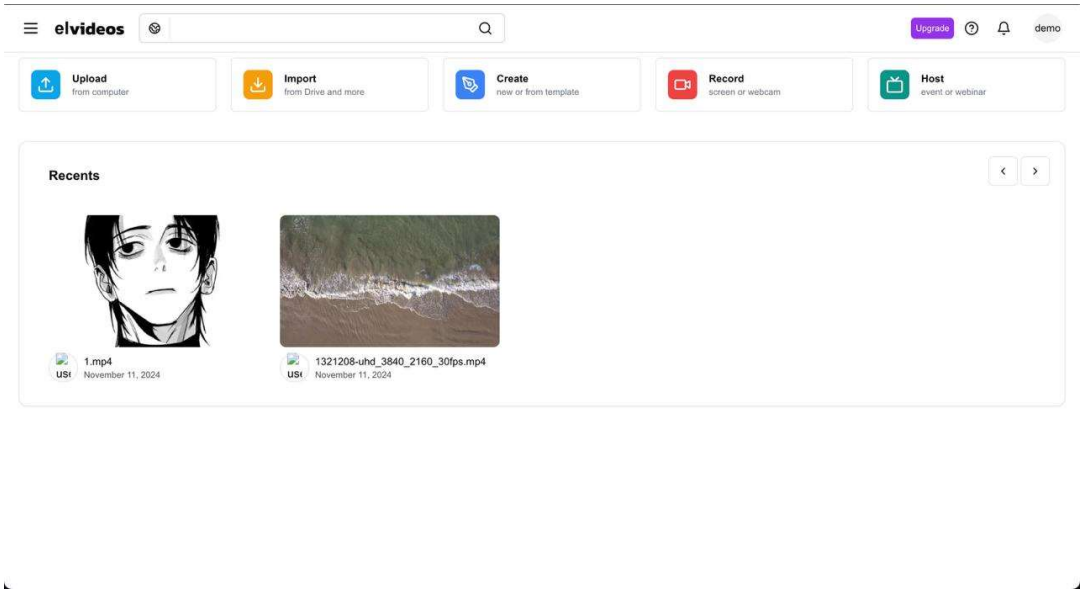

Fig. 5 :- Home Page

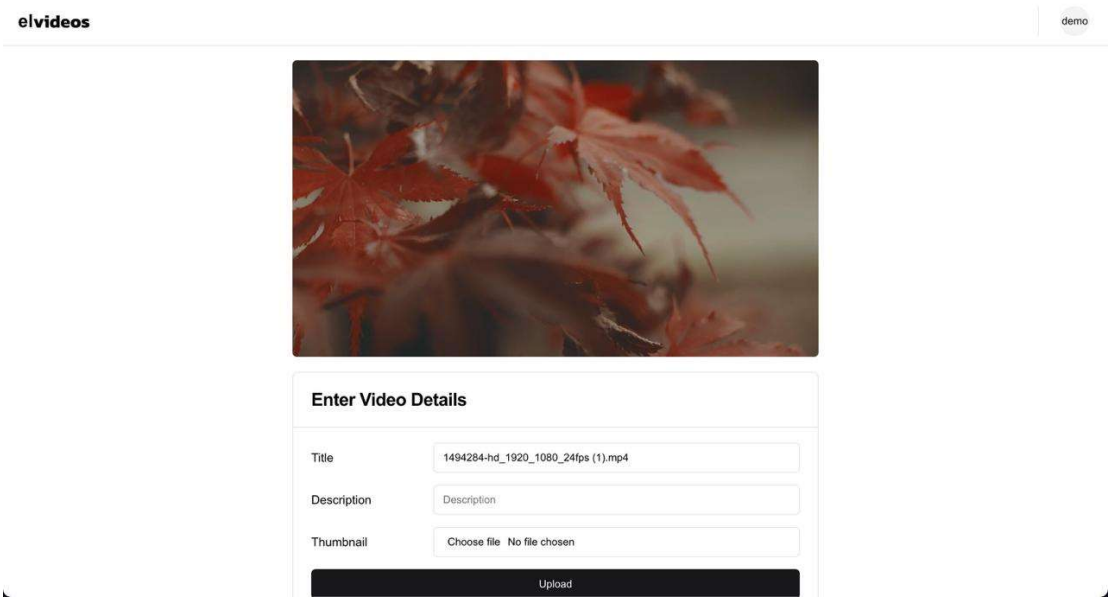

Fig. 6:- Video Uploading

Chapter V: CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

The media management and processing system successfully integrates modern cloud-based technologies to provide a scalable and secure solution for handling various media types. It ensures smooth user experiences, allowing users to upload, import, and manage media files with ease. Key features like secure authentication, media transcoding, and cloud storage were implemented to meet the requirements of the project.

The system's modular architecture, backed by microservices, cloud infrastructure, and CI/CD pipelines, ensures that it is not only robust but also scalable, with the capacity to handle high traffic and large media files efficiently. The inclusion of content delivery networks (CDNs) further enhances performance, making the system capable of delivering media content quickly to users across different geographical regions.

Moreover, the solution ensures data security through encrypted communication, secure login protocols, and comprehensive user access controls. Rigorous testing of system functionalities, load, and security measures contributed to the overall robustness of the platform.

5.2 Future Scope

While the project successfully meets its objectives, several improvements and additional features can be explored to enhance the system's functionality and performance in the future:

1. AI-Powered Media Processing:

- **Automatic Media Tagging**: Integrating artificial intelligence (AI) and machine learning (ML) algorithms can allow for automatic tagging, categorization, and labeling of media content based on its content (e.g., facial recognition, object detection).
- **Content Personalization**: AI can also be used to recommend media content to users based on their previous interactions and preferences, creating a personalized experience.

2. Real-Time Media Streaming:

- Incorporating live-streaming capabilities with real-time transcoding and adaptive bitrate streaming would allow users to broadcast live events directly from the platform. This can be especially useful for content creators and businesses looking for a comprehensive media solution.

3. Integration with Advanced Video Editing Tools:

- Adding in-browser video and image editing features would enhance the platform's utility by allowing users to edit media files directly after uploading, without needing external tools.

4. Enhanced Security Features:

- **Multi-Factor Authentication (MFA)**: While the current system has secure login mechanisms, adding multi-factor authentication will provide an additional layer of security for sensitive user data.
- **Blockchain-Based Media Ownership**: Blockchain can be integrated for tracking media file ownership, enabling users to prove ownership of original content and ensuring transparency and security in media distribution.

5. Global Expansion via Edge Computing:

- Deploying edge computing infrastructure in conjunction with the current CDN setup can significantly reduce latency by processing media at data centers closer to the end-users. This will improve system performance, especially for users in remote locations.

6. Integration with Social Media Platforms:

- The platform can be enhanced by allowing direct sharing of media content to social media platforms like Instagram, YouTube, or Facebook. This will streamline the media publishing process, enabling users to reach a broader audience.

7. Mobile Application:

- Developing a mobile application for Android and iOS will make the system more accessible to users who prefer managing and processing their media on the go. Mobile apps can also leverage device-specific features like the camera for direct media uploads.

8. Augmented Reality (AR) and Virtual Reality (VR) Integration:

- In the future, the system can be extended to support AR and VR media. This could open up new possibilities for immersive media experiences, particularly in entertainment, education, and marketing industries.

REFERENCES

1. Li, X., Darwich, M., Salehi, M.A. and Bayoumi, M., 2021. A survey on cloud-based video streaming services. In *Advances in computers* (Vol. 123, pp. 193-244). Elsevier.
2. Nacakli, S. and Tekalp, A.M., 2020. Controlling P2P-CDN live streaming services at SDN-enabled multi-access edge datacenters. *IEEE Transactions on Multimedia*, *23*, pp.3805-3816.
3. Toshniwal, A., Rathore, K.S., Dubey, A., Dhasal, P. and Maheshwari, R., 2020, May. Media streaming in cloud with special reference to amazon web services: A comprehensive review. In *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)* (pp. 368-372). IEEE.
4. Shabrina, W.E., Sudiharto, D.W., Ariyanto, E. and Al Makky, M., 2020, February. The QoS improvement using CDN for live video streaming with HLS. In *2020 International Conference on Smart Technology and Applications (ICoSTA)* (pp. 1-5). IEEE.
5. Shabrina, W.E., Sudiharto, D.W., Ariyanto, E. and Al Makky, M., 2020. The Usage of CDN for Live Video Streaming to Improve QoS. Case Study: 1231 Provider. *J. Commun.*, *15*(4), pp.359-366.
6. Kumar, T., Sharma, P., Tanwar, J., Alsghier, H., Bhushan, S., Alhumyani, H., Sharma, V. and Alutaibi, A.I., 2024. Cloud-based video streaming services: Trends, challenges, and opportunities. *CAAI Transactions on Intelligence Technology*, *9*(2), pp.265-285.
7. Patel, U., Tanwar, S. and Nair, A., 2020. Performance Analysis of Video On-demand and Live Video Streaming using Cloud based Services. *Scalable Computing: Practice and Experience*, *21*(3), pp.479-496.
8. Ghabashneh, E. and Rao, S., 2020, July. Exploring the interplay between CDN caching and video streaming performance. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications* (pp. 516-525). IEEE.
9. Reznik, Y., Cenzano, J. and Zhang, B., 2021. Transitioning broadcast to cloud. *Applied Sciences*, *11*(2), p.503.
10. Huang, T. and Sharma, A., 2020. Technical and economic feasibility assessment of a cloud-enabled traffic video analysis framework. *Journal of Big Data Analytics in Transportation*, *2*(3), pp.223-233.
11. Google Docs for AWS
12. https://aws.amazon.com/media-services/
13. https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Introduction.html