International Journal of Science and Research Archive

eISSN: 2582-8185

Cross Ref DOI: 10.30574/ijsra

Journal homepage: https://ijsra.net/

(REVIEW ARTICLE)

Check for updates

Automating you tube video uploads using cloud-native technologies

Vardhan Shrinath Hegde *, Tanya Srinivas and R. Indra

Department of Information Science and Engineering, B.M.S. College of Engineering, Bangalore, India.

International Journal of Science and Research Archive, 2025, 15(02), 900-907

Publication history: Received on 03 April 2025; revised on 11 May 2025; accepted on 13 May 2025

Article DOI: https://doi.org/10.30574/ijsra.2025.15.2.1437

Abstract

In the evolving landscape of digital content creation, video editing remains a critical bottleneck for YouTube creators aiming to produce high-quality, consistent content. This paper presents a comprehensive full-stack platform designed to streamline collaboration between YouTube creators and freelance video editors. The system facilitates an end-to-end workflow, enabling creators to upload raw footage, define editing preferences, and interact seamlessly with editors through a structured, role-based interface. Built with a scalable microservices architecture and powered by React.js on the frontend, the platform leverages Amazon S3 for advanced cloud storage, ensuring efficient handling of large media files with fast and reliable transfers. Supa base is employed for backend data management, user authentication, and real-time collaboration features, providing a robust foundation for managing concurrent user interactions. Secure authentication mechanisms and access control policies further ensure data privacy and system integrity. Additionally, the integration of the YouTube Data API allows creators to directly publish finalized content with appropriate metadata and scheduling, simplifying the content production lifecycle. This work demonstrates how the platform optimizes video editing collaborations through automation, scalability, and user-centric design, offering a professional and efficient ecosystem for content delivery.

Keywords: You tube Automation; Cloud-Native Platform; Video Editing Workflow; Amazon S3; Supa Base; Freelance Collaboration; Microservices Architecture; React. Js; Youtube Data API; Media Content Delivery

1. Introduction

1.1. Context and Motivation

In the digital age, YouTube has emerged as a dominant platform for content dissemination, with millions of creators producing video content daily across diverse domains such as education, entertainment, technology, and lifestyle. As audience expectations for video quality rise, creators increasingly rely on professional editing to enhance their content's visual appeal and coherence. However, the editing process is time-consuming and often requires specialized skills, creating a bottleneck in the content production pipeline. Freelance editors offer a viable solution, but collaboration between creators and editors is frequently hindered by the lack of structured tools and workflows tailored to their needs.

1.2. Problem Statement

Despite the availability of general-purpose file sharing and communication platforms, there is a distinct lack of a unified, purpose-built system that addresses the unique requirements of video editing collaboration. Challenges include the transfer of large raw footage, clarity in editing preferences, version control, communication overhead, and secure handling of intellectual property. Additionally, the absence of seamless integration with publishing platforms like YouTube results in redundant workflows and increased time-to-publish.

* Corresponding author: Vardhan Shrinath Hegde

1.3. Contribution

This paper presents a full-stack platform that bridges the gap between YouTube creators and freelance video editors by offering an integrated, cloud-based solution for video editing collaboration. The platform supports:

- Structured workflows for submitting, editing, reviewing, and delivering video content.
- Cloud storage for efficient handling of large media files.
- Secure user authentication with role-based access control.
- Direct integration with the YouTube Data API for publishing finalized videos with metadata and scheduling support.

The system is built on a scalable microservices architecture, using React.js for the frontend and robust backend services to ensure high availability and performance. The platform is designed to enhance productivity, reduce friction in collaboration, and accelerate the content publishing process.

2. Related Work

In recent years, several platforms and systems have emerged to support collaborative content creation and media processing workflows. Tools such as Frame.io and Wipster have gained popularity for enabling creators and editors to share video drafts, provide feedback, and manage revisions in cloud environments [1][2]. These systems focus primarily on post-production collaboration but often require integration with external storage or editing tools, making them less suitable for end-to-end workflows involving raw footage ingestion, editing instructions, and automated publishing.

Other platforms, like Dropbox and Google Drive, provide generic cloud storage solutions that support large file uploads and sharing, but they lack purpose-built features for video editing tasks, such as revision tracking, editing preference templates, or secure handoff mechanisms between editors and clients [3]. Similarly, freelance job platforms such as Upwork or Fiverr enable creators to connect with video editors but do not offer the infrastructure or workflow management features needed to support the editing process itself.

From a technical perspective, scalable video processing systems have been explored in research and industry settings, particularly in the context of distributed media pipelines and microservices-based architectures [4]. For instance, some systems use containerized backends and cloud-native technologies to handle video encoding, transformation, and distribution [5]. However, these systems often lack user-facing components tailored to creative professionals and require significant technical overhead to deploy and maintain.

Our platform distinguishes itself through its comprehensive, full-stack design that unifies collaboration, storage, editing instruction, publishing, and user management in a single ecosystem. By integrating the YouTube Data API, we go beyond traditional media handling to include scheduling and metadata publishing, a feature rarely found in existing platforms. Additionally, our system supports concurrent usage by creators and editors through a secure, role-based interface and leverages a cloud-native backend to ensure scalability and performance under high load. This level of vertical integration and automation provides a more seamless and efficient user experience than the fragmented approaches commonly found in the field.

3. Development of the System

3.1. Requirement analysis

The development of project is guided by a thorough analysis of the collaborative needs between YouTube content creators and video editors. The platform is designed to enable seamless interaction through secure user registration, role-based access control, and structured workflows for video upload, editing, and approval.

Creators upload raw video files, which are accessed by editors for processing and re-upload. Once completed, the system notifies the creator, who can review the final product and, if satisfied, schedule its release via the YouTube Data API. OAuth 2.0 is integrated to ensure secure authentication and account access. The system also allows configuration of video metadata such as titles, tags, and descriptions during the approval phase.

Non-functional requirements emphasize performance in handling large media files, horizontal scalability, and strong security via HTTPS and AES encryption. The platform is built to be intuitive and accessible, with a target uptime of 99.9% to support continuous availability.

Technologically, the system leverages a Linux server environment, Node.js for backend logic, React.js for the frontend, and PostgreSQL or MongoDB for data storage. AWS S3 handles scalable video storage, and CI/CD processes are supported through Jenkins or GitHub Actions. These requirements collectively ensure the platform's reliability, usability, and scalability for modern content collaboration.

4. System Design

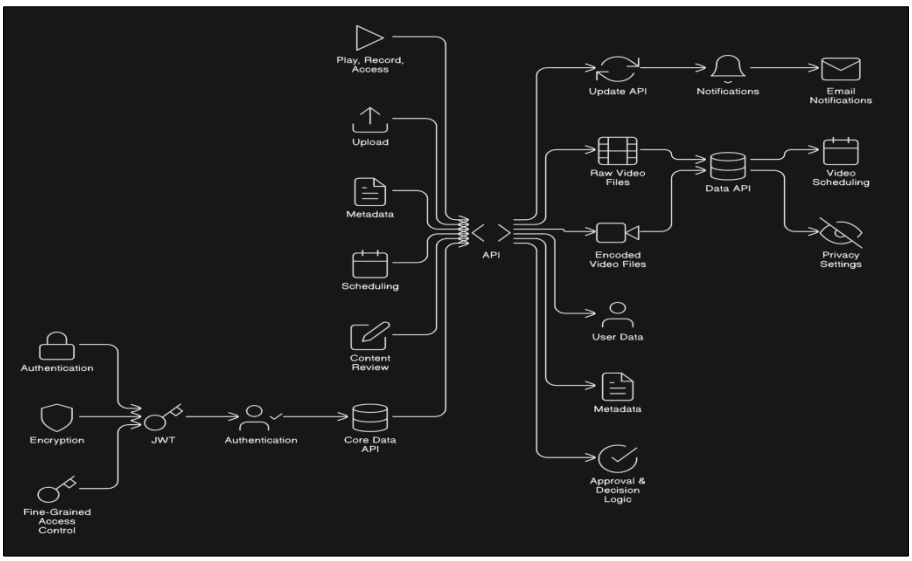

Figure 1 System Design architecture

The system design of the project, as illustrated in the diagram, presents a modular and secure architecture centered around a robust API layer that interconnects key services such as video upload, scheduling, review, and metadata management. The design begins with an authentication and authorization pipeline incorporating encryption, JSON Web Tokens (JWT), and fine-grained access control to ensure secure user identification and role-based permissions. This authentication system feeds into the Core Data API, which acts as the backbone for managing user data, video assets, and workflow operations.

At the core of the system is a comprehensive API interface that mediates interactions between the user-facing components (e.g., play, upload, and content review) and backend services. Users can upload raw video files, provide metadata, and schedule content for review and publication. The content then flows through the processing pipeline, where it is stored, encoded, and linked with user and metadata records. These assets are subsequently routed to an approval and decision logic module, enabling creators to review and approve content before public release.

Once approved, the system utilizes the YouTube Data API to perform operations such as scheduling video publications and configuring privacy settings. Notifications are handled through a dedicated update API, which triggers system alerts and email notifications to inform users of changes in the workflow—such as successful uploads or content awaiting review. This real-time feedback loop enhances collaboration between creators and editors.

Overall, the system is designed to scale with user demand while maintaining a high degree of security and reliability. The separation of concerns—evident in discrete modules for authentication, video processing, metadata management, and decision logic—ensures maintainability and facilitates future feature enhancements. This architecture provides a secure, efficient, and user-friendly platform tailored for the collaborative nature of digital content creation.

4.1. Tools and Technologies

4.1.1. Server-Side Configuration

The server-side infrastructure is built upon a Linux-based operating system, with Ubuntu 22.04 (or its equivalent) selected for its stability and support for long-term deployments. The backend services are developed using FastAPI, a high-performance Python web framework known for its asynchronous capabilities and automatic OpenAPI documentation support. Data persistence is managed using PostgreSQL, a reliable and extensible relational database

system suitable for transactional operations. For scalable and durable storage of video content, the platform integrates Amazon S3 cloud storage. Integration with YouTube is achieved via the YouTube Data API, enabling secure and programmatic video uploads, metadata configuration, and scheduling functionalities.

4.1.2. Client-Side Configuration

The client-side application is developed using Next.js, a React-based framework optimized for performance and server-side rendering. It is complemented by Express.js to manage server-side routing and middleware for client interactions. The application is designed to be accessible through modern web browsers including Google Chrome, Mozilla Firefox, and Microsoft Edge, with compatibility ensured for the latest stable versions. Development dependencies such as Node.js and package managers like npm or yarn are used to support local builds and frontend package management.

4.1.3. Development and Testing Tools

For development, Visual Studio Code is employed as the primary code editor, offering a rich ecosystem of extensions and integrated tooling. Version control is managed using Git, with repositories hosted on GitHub to enable collaborative development and CI/CD integration. Testing and API validation are facilitated through interactive Swagger documentation, automatically generated by FastAPI, which allows developers to interact with and verify backend endpoints directly from the browser.

5. Result and Discussion

The implemented project's platform demonstrates a complete end-to-end workflow for collaborative video editing and publishing, from user registration to final YouTube upload. The following results describe how a creator and an editor interact through the system, how authentication and role management are handled, and how the modular backend architecture enables seamless integration of storage and publishing services.

5.1. Collaboration Workflow

The creator's dashboard user interface serves as the central hub for creating projects (workspaces), uploading videos, assigning editors, and tracking progress. This is visually presented in Figure 1, which shows the intuitive, role-based dashboard interface for creators, highlighting buttons to upload content, assign editors, and monitor project status — designed for ease of use even for non-technical users.

Additionally, the frontend is hosted using Vercel, which provides a scalable, serverless cloud platform with global CDN integration, enabling fast page loads and seamless deployment pipelines.

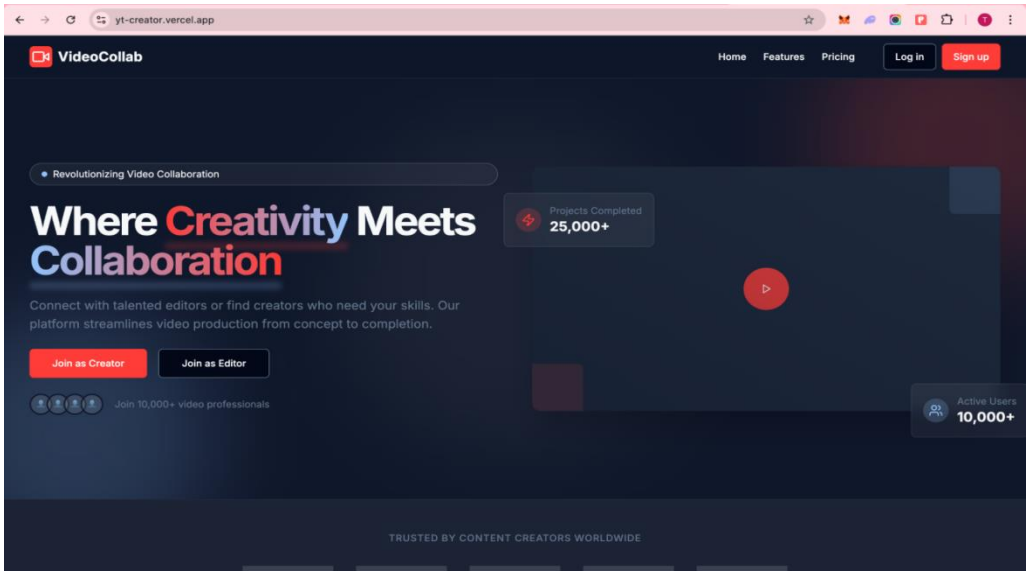

Figure 2 Home page of the frontend

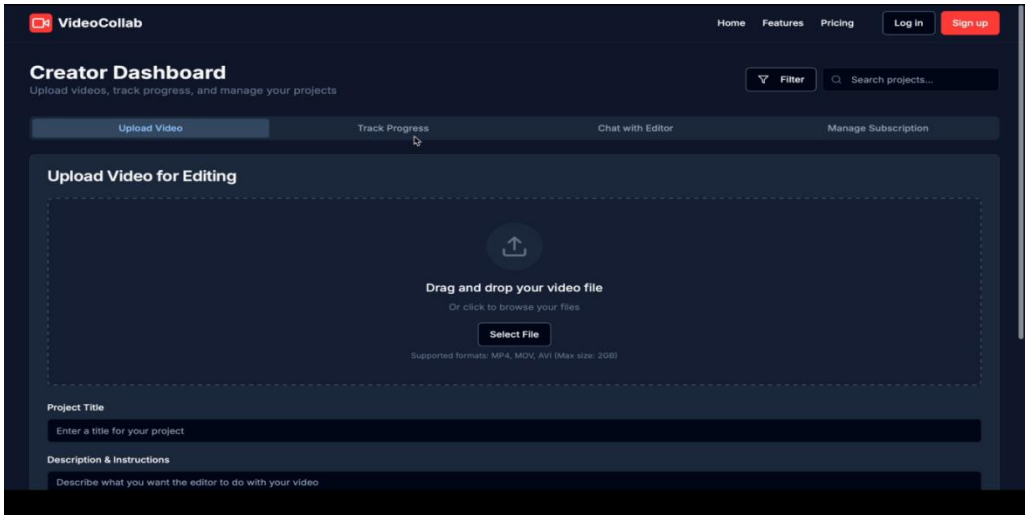

Figure 3 Creater's video upload of the frontend

Using this interface, the typical workflow of the project proceeds as follows:

5.2. User Registration and Role Selection

A new user registers via the front-end (built with React/Next.js) using Supabase's OAuth 2.0 authentication (for example, signing in with a Google account or email). During signup, the user chooses a role as either a Creator or an Editor. This role information is saved in the platform's database and also stored client-side (in a secure cookie) to personalize the user experience.

5.3. Workspace Creation

After logging in, a creator can create a new workspace for a video project using the dashboard (refer to Figure 1). Each workspace corresponds to a project that will go through editing. The creator provides initial details such as the video title and description. In the database, a new project entry is created linking this workspace to the creator's user account.

5.4. Video Upload to S3

The creator uploads the raw video footage for the project through the web interface. The front-end communicates with the Fast API backend to handle the upload. The video file is streamed to an Amazon S3 bucket via the backends' S3 service module. Progress is reported on the front-end, and once complete, the file is securely stored in S3. The database record for the project is updated with the S3 file URL or key.

5.5. Editor Assignment

The creator assigns a freelance editor to the workspace. This can be done by selecting an editor (who has an account on the platform) from a list or by inviting them. The assignment is recorded in the database (associating the editor's user ID with the project). Role-based access control ensures that the assigned editor now has permission to access the project's video file. The editor is notified (for example, via the dashboard or email) that a new project is available for editing.

5.6. Editing Process

The editor accesses the project through the platform (logging in as an editor). They can download the raw video from S3 using the platform's interface (the download is facilitated by the S3 module of the backend). The actual video editing is performed using the editor's preferred editing software off-platform. Once the edits are complete, the editor uploads the finished video back to project via an upload interface similar to the creator's, which again uses the S3 upload endpoint to store the edited video file. The project entry in the database is updated to mark that an edited version has been uploaded and is ready for review.

5.7. Creator Review and Approval

The creator is alerted that the edited video is ready for review. Through the dashboard, the creator can stream or download the edited video (from S3) to evaluate it. If further changes are needed, the creator can communicate with the editor (outside the system or via any messaging feature, if provided) and the editor can upload a revised version. Once satisfied, the creator marks the video project as approved in the workspace. This approval status is recorded in the database and triggers the next step in the workflow.

5.8. Automated YouTube Publishing

Upon approval, project automatically handles publishing the video to YouTube. The backend's YouTube service module is invoked to upload the final video file from S3 to YouTube via the YouTube Data API. The creator's YouTube channel credentials (which were obtained via an OAuth permission flow and stored securely earlier) are used to authorize this upload. The video's title, description, and metadata (entered by the creator in the workspace) are included in the upload. As a result, the video is published on the creator's YouTube channel without the creator manually visiting YouTube's website. This automation ensures a smooth hand-off from editing to distribution.

5.9. Post-Publishing Confirmation

After a successful upload, the platform updates the project's status to published and stores the YouTube video ID or URL in the database. The creator sees a confirmation on the dashboard with a direct link to the YouTube video.

This entire workflow enables seamless collaboration, with the creator and editor each focusing on their tasks while the system handles file management, role permissions, and publishing logistics — minimizing miscommunication and manual errors.

5.9.1. Authentication and User Roles

User authentication is handled through Supabase's OAuth 2.0 service, which offers secure and convenient sign-up/sign-in flows. Supabase Auth supports third-party logins (like Google) and provides a unique user identity. During sign-up, the user specifies their role (creator or editor), which is saved in the Postgres database (managed by Supabase). A visual overview of this database structure is shown in Figure 2, which displays how the users table, projects table, and other relational links define role-based access and manage project-to-user assignments securely.

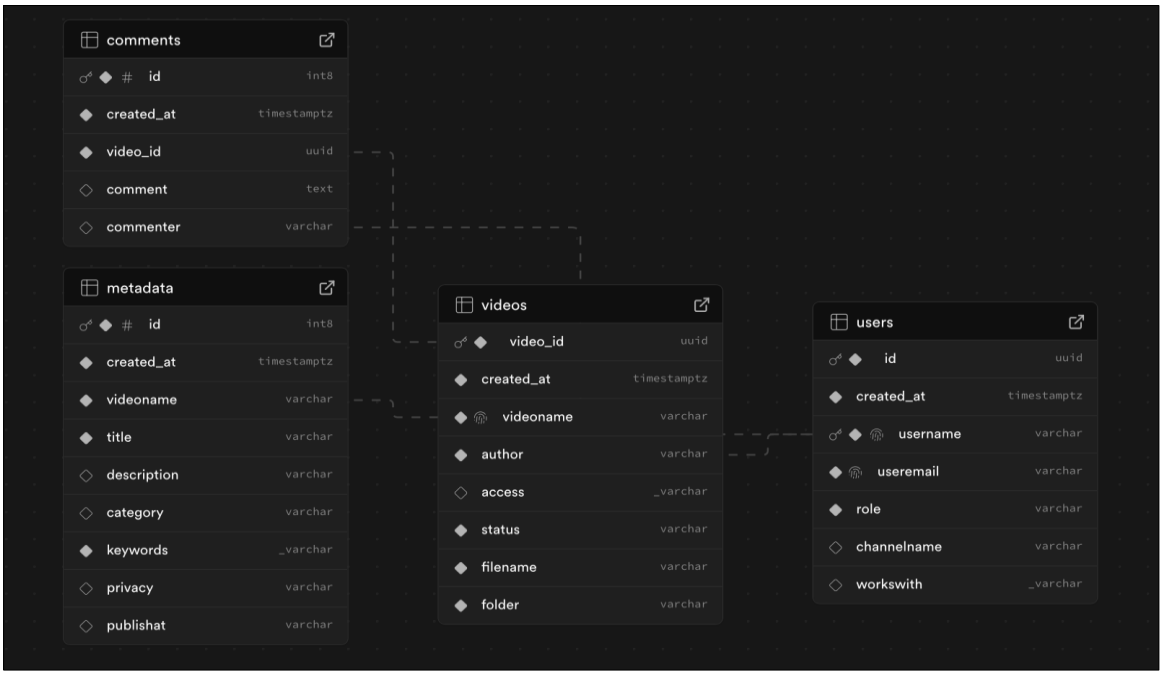

Figure 4 Database table design

The role is also stored in a front-end cookie to dynamically adjust the user interface. Additionally, Supabase provides a JWT session token that the backend uses to securely verify identity before sensitive operations like project creation, editor assignment, file upload, or video approval. This layered security approach ensures that only authorized users can perform specific actions.

5.9.2. Backend Architecture and API

The backend is implemented using FastAPI and is divided into modular services: user, s3, and youtube. The modular architecture which has the backend directory structure — clearly separating routers, models, and services for each functionality. This separation enhances maintainability, scalability, and the ability to independently upgrade each service.

The FastAPI application exposes a unified API under versioned endpoints (for example, /api/v1/user/, /api/v1/s3/, /api/v1/youtube/). As shown in Figure 3, the Swagger UI automatically documents these endpoints, making it easy for developers to test, debug, and verify the backend's capabilities interactively. This documentation proves the completeness and transparency of the system, helping ensure robust development practices.

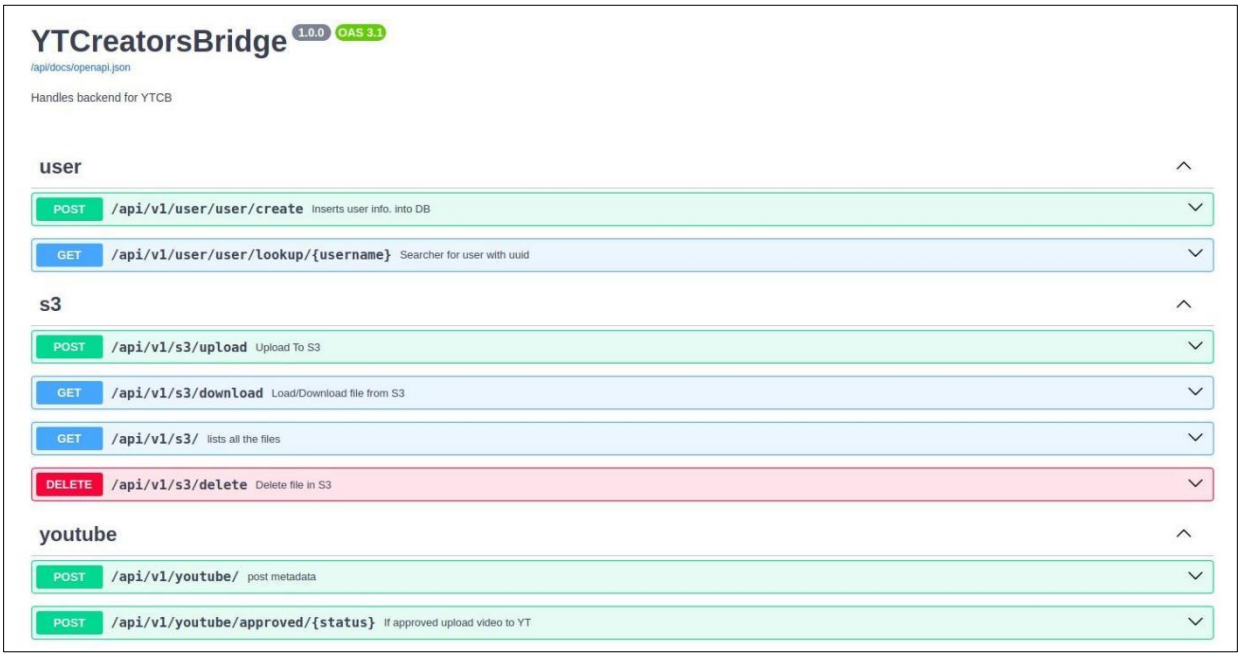

Figure 5 Backend Swagger documentation snapshot

The overall architecture, with its modular routing, RESTful principles, and stateless backend design (with persistent data stored in Supa base or S3), allows the system to scale horizontally by adding backend instances behind a load balancer. This ensures the platform can serve many creators and editors concurrently without performance degradation.

6. Conclusion

The project demonstrates a fully integrated full-stack solution for collaborative video editing and publishing[6]. It connects YouTube creators and freelance editors through a unified platform, providing tools to manage projects efficiently. The React/Next.js front-end offers an intuitive interface, while the FastAPI backend manages all core operations. Supa base simplifies identity management and data storage, and external services like Amazon S3 and YouTube Data API integrate seamlessly.

A major strength is the system's modular backend architecture, which promotes maintainability and scalability. Role-based access control ensures security and tailored user experiences, while automation reduces manual effort — particularly in the final publishing stage.

Future enhancements could include AI-powered editing suggestions, advanced analytics, and DRM features to improve security and content protection. These additions would strengthen the platform's ability to not only connect creators and editors but also provide intelligent tools to improve the creative process and ensure the safety of the content.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Frame.io. (2024). Frame.io's massive productivity update is now available for everyone. The Verge. Retrieved from https://www.theverge.com/2024/10/14/24269780/frameio-v4-productivity-update-availability-c2c

[2] Wipster. (n.d.). Wipster.io | The World's Leading Video Review and Approval Platform. Retrieved from https://www.wipster.io/

[3] Dropbox. (2024). How to edit video and audio files in Dropbox. Dropbox Help. Retrieved from https://help.dropbox.com/view-edit/edit-video-audio

[4] Google Drive. (2025). Google Drive. Wikipedia. Retrieved from https://en.wikipedia.org/wiki/Google_Drive

[5] Poms, A., Crichton, W., Hanrahan, P., & Fatahalian, K. (2018). Scanner: Efficient Video Analysis at Scale. arXiv. Retrieved from https://arxiv.org/abs/1805.07339

[6] Github Link for the Repository https://github.com/VardhanHegde/YTcreaterbridge