DECENTRALIZED VIDEO SHARING PLATFORM

LAI JUNLIN , YAZRINA YAHYA

Fakulti Teknologi & Sains Maklumat, Universiti Kebangsaan Malaysia, 43600 UKM Bangi, Selangor Darul Ehsan, Malaysia

ABSTRACT

Nowadays, most of the video sharing platforms are centralized and controlled by a single entity, which means that all data uploaded by users are under the control of the platform, and in fact, users do not have absolute control over their data. This project aims to combine blockchain and smart contracts to ensure that users have absolute control over their data, using traditional Web2 storage systems like google cloud storage to improve playback performance. The backend of this project uses a microservice and event-driven architecture to ensure scalability and flexibility. The frontend is developed with Next.js and React(Vercel. 2024.) to provide a responsive and user-friendly web application.

INTRODUCTION

With centralized websites like YouTube taking front stage in the market, video-sharing sites have become a main tool for consumers as well as content creators in recent years. These platforms have major drawbacks even if their user bases are somewhat high and they are rather easily available. Centralized control of data distribution has led to growing issues about censorship and content manipulation. The content creator often runs the danger of being demonetized, having their work deleted at will, and having opaque visibility and distribution under control of blackbox algorithms. Moreover, by keeping control over user-generated content, these sites restrict users' capacity to completely own their works. The project, DeCentralTube, aims to provide a distributed peer-to-peer video-sharing platform that uses IPFS(IPFS Docs. n.d.) decentralized storage to provide users with total ownership and control over their content while

integrating traditional storage systems such as Google Cloud Storage, to enhance video playback performance. The video stores on IPFS can be minted as NFTs which serve as digital certificates proving content ownership. This platform combines the transparency of blockchain with the efficiency of Web2 infrastructure, to ensure both data sovereignty and a smooth user experience.

METHODOLOGY

To ensure rapid iteration, high availability, and consistent delivery of the DecentralTube platform, the system will follow a strong DevOps approach that promotes automation, modularity, and progressive deployment. Continuous integration and deployment practices, along with a growing infrastructure-as-code strategy, are key to keeping development fast and reliable.

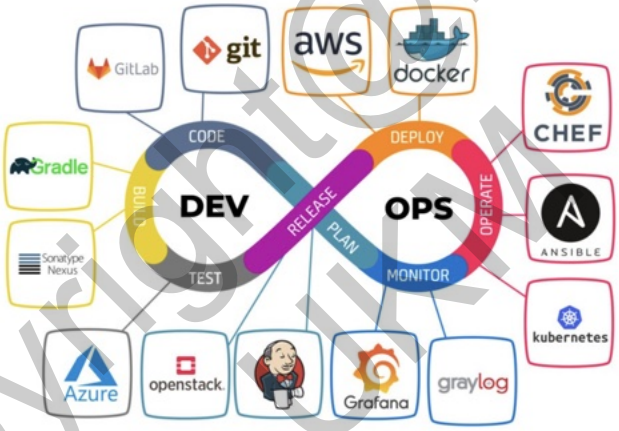

Figure 1 DevOps Methodology

Source: https://www.linkedin.com/pulse/devops-methodology-keerthana-s/

I carefully manage all code contributions using GitHub. And use feature branches and pull requests to ensure code quality. The Continuous Integration (CI) pipelines, powered by GitHub Actions, automate important steps like code linting, static analysis, and complete build processes for both the frontend and backend. Additionally, I rigorously build and test smart contracts using Anchor CLI. These pipelines also run integration tests across all main modules—authentication, video, NFT, and authentication—to make sure that every change is fully verified before being merged.

For Continuous Deployment (CD), I practice progressive delivery using tiered environments to ensure safe and reversible updates. Changes automatically deploy to the Development environment using GitHub Actions, with platforms like Vercel for the frontend and Railway for the backend. Staging deployments are manually triggered and include testnet contracts and seeded data for thorough quality assurance. Finally, I carefully control Production releases, focusing on environment isolation and full audit logs.

Although the current infrastructure is light, I am actively incorporating Infrastructure-as-Code principles. And containerize local development environments with Docker Compose for MongoDB(MongoDB, Inc. 2024.), Redis(Redis Ltd. 2024.), Postgres(The PostgreSQL Global Development Group. 2024.), and Apache Kafka(The Apache Software Foundation. 2024.), which ensures consistency across all developer setups. Looking ahead, I will fully codify the production infrastructure with tools like Terraform, allowing for scalable and automated deployments.

RESULTS AND DISCUSSION

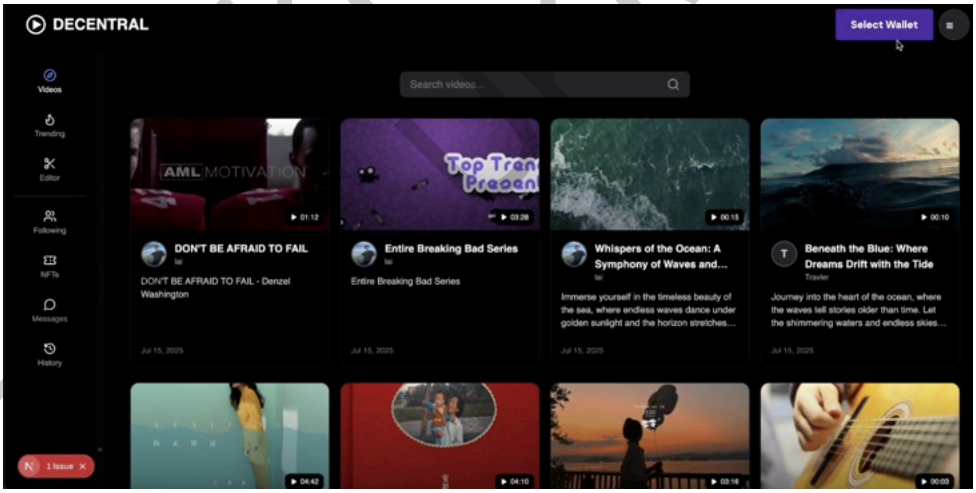

Figure 2 video main page

This is the main page of the system. The user can watch videos here. On the left is a navigation bar corresponding to different pages, with the default being the video page where users can see a variety of videos. At the top of the video section, there is a search bar where users can enter keywords to search for videos. In the top right corner, there is a "Select Wallet" button, which

users will use later to connect to a cryptocurrency wallet installed in their browser. Next to it, the three horizontal lines represent the menu.

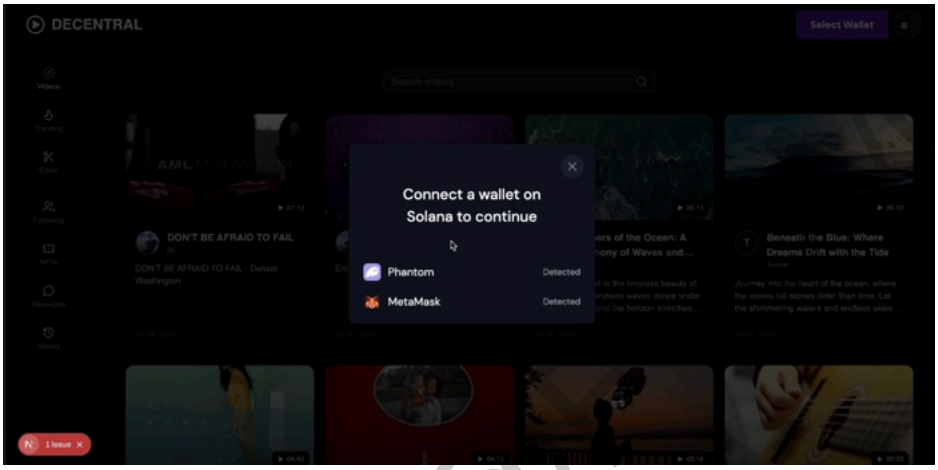

Figure 3 Select Wallet button window

When the user clicks the "Select Wallet" button, a pop-up window appears showing the cryptocurrency wallets detected in the browser. Figure 3 shows that two wallets have been detected, which are the ones I have already installed. The first one is called Phantom, which is primarily used for the Solana blockchain, and the second one is MetaMask, mainly used for Ethereum blockchain. For now, I will choose the first one, as my platform is based on the Solana blockchain.

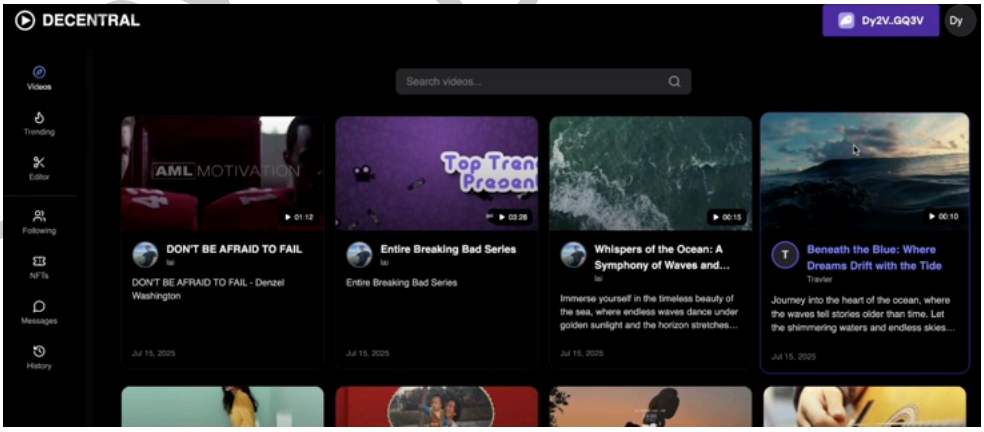

Figure 4 After connect wallet

As you can see, once the user successfully connects their wallet, the "Select Wallet" button in the top right corner changes its state to display the connected wallet address. A blockchain wallet address essentially serves as an identity in the blockchain world.

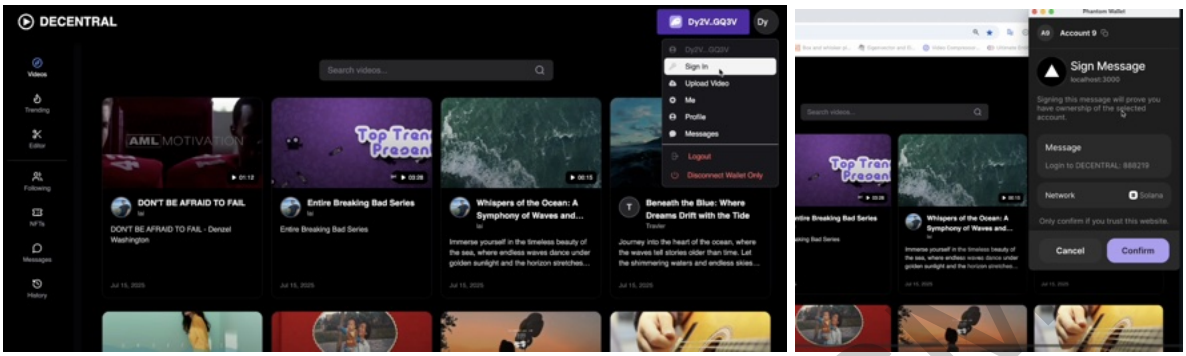

Figure 5 Sign in window

But connecting to the wallet only happens on the client side, the server doesn't know the user's identity yet, so it must also click Sign in on the menu, after clicking this will pop up a popup window, the middle part of the popup window content shows a hardcoded message, this message is the same for both the server side and the client side. After the user clicks the confirm button, the javascript code responsible for the wallet logic will use the private key of the user's cryptocurrency wallet to sign the message, generating a signature. The front-end code will send this signature along with the address of the user's wallet to the back-end via a HTTP request, and the back-end will use the address of the wallet that the user sends along with the signature to verify the signature. The crypto wallet address is Ed25519 public key base58 encoding results, the server side by solving the base58 encoding using the same algorithm, get the public key, and then use this public key and message to verify the signature sent by user, if the signature verification is successful, the server side can confirm that the wallet corresponding to the private key under user control. This is based on the mathematical principle of asymmetric encryption, and the JWT token will be granted to users representing the session access credentials after verification. The JWT token is then cached in the backend's Redis for one day, meaning that all subsequent operations on resources will use this JWT token for authentication. If the JWT token expires, the user must re-verify using the wallet signature.

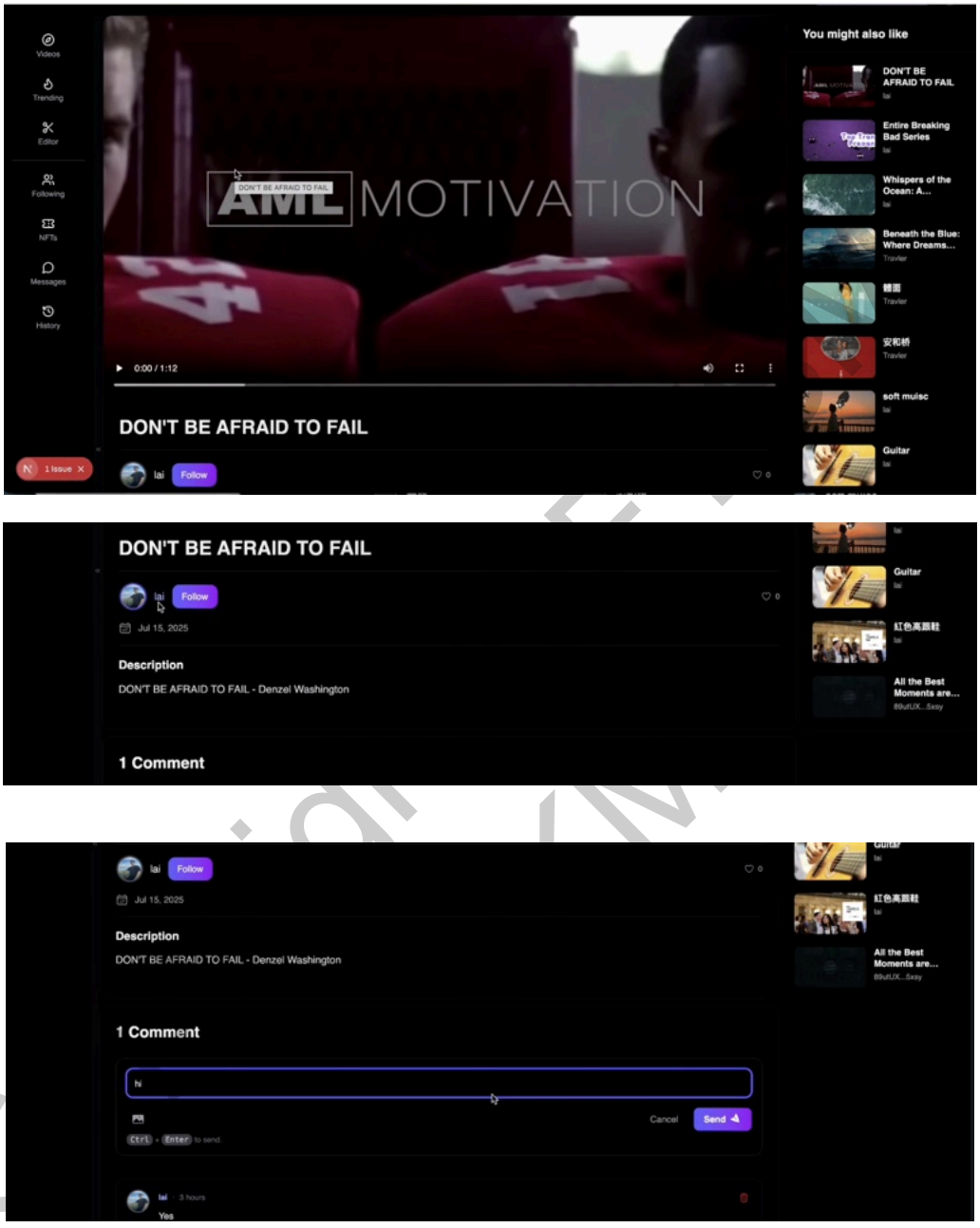

Figure 6 Video Page

When you click on one of the videos, you will enter the viewing page of the video, right in the middle is the video that can be played, as shown in Figure 6, and on the right side is the recommended video, which will be recommended according to the video that the user likes to watch. Below the video is the title of the video, a short description of the video, and the avatar and username of the uploading author. On the right is a button to follow the creator, which the

user can click to follow. At the bottom right of the video is a small heart button that allows users to like the video.

Then below that is the video's comment section, as shown in Figure 6 and 7where users can post their thoughts about the video in the input box and then click the send button. The video comment section supports both text and image formats.

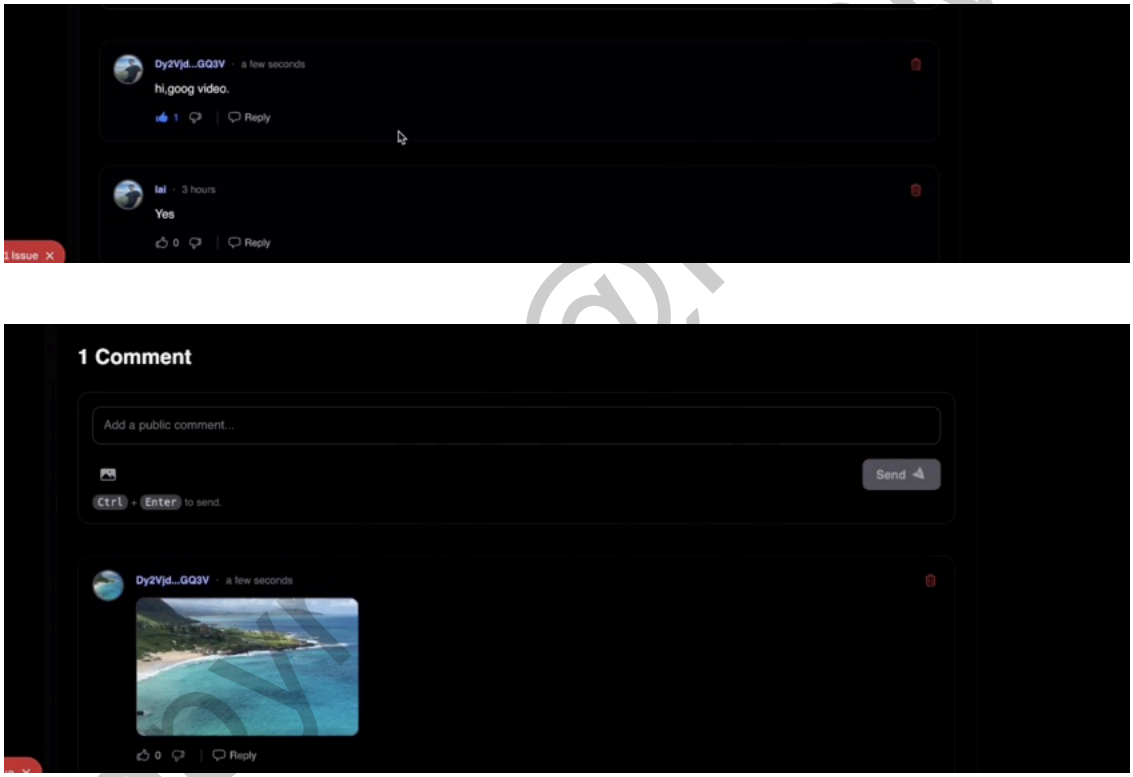

Figure 7 Comment Section

Each comment will be displayed in columns and each comment has basic information about the author of that comment and when the comment was posted. Users can like and dislike the comments with one upward facing thumb and one downward facing thumb, as well as reply to the comments, which will further broaden the social depth of the video.

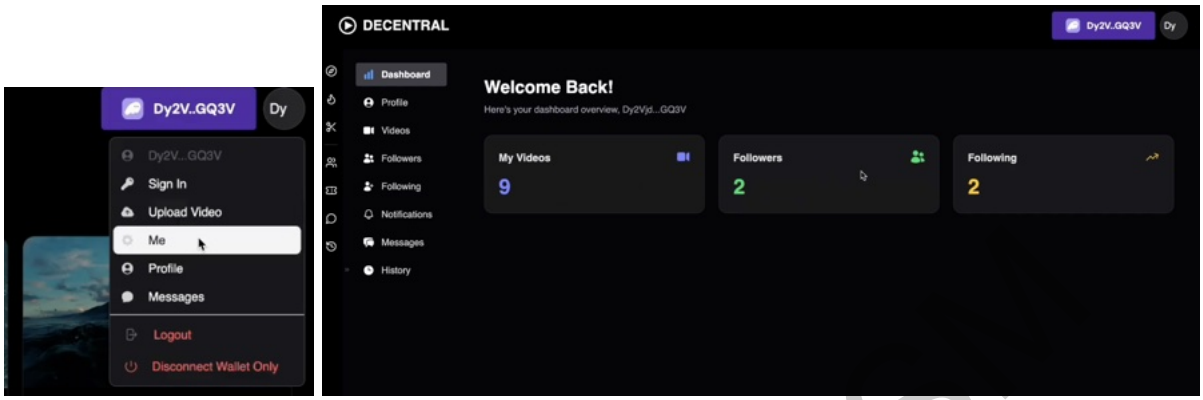

Figure 8 Me Page

When you click on the Me of the menu, as shown in Figure 8 it will jump to a new page, a page where the user can view a lot of information. The left navigation bar will be narrowed down to the icons, and then a new sub-navigation bar will appear next to the leftmost navigation bar. In the middle of the page is the content corresponding to each entry in the navigation bar, which is by default a dashboard subpage. Users can view basic statistics, such as how many videos have been uploaded, how many followers they have, and who they follow. When clicked, they will be redirected to the corresponding page.

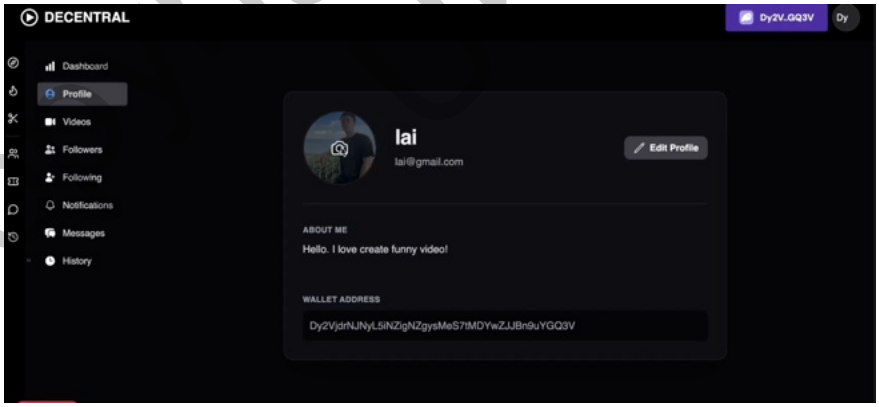

Figure 9 Profile Page

The second in the navigation bar is the Profile page, which focuses on basic information about the user, such as avatar, username, and a short description and the user's wallet address, as shown in Figure 9. Users can click Edit Profile Button to change their username or description. After clicking on the avatar directly, the user can upload a picture from the local storage as an avatar.

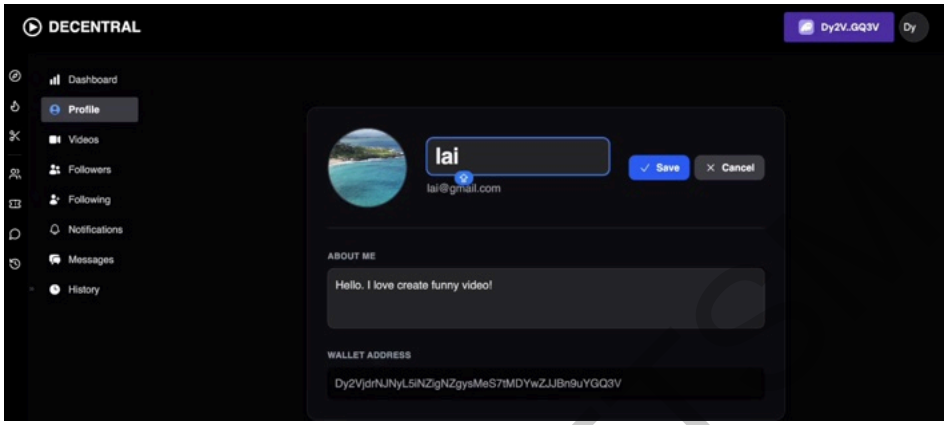

Figure 10 Edit Profile

Once the user has clicked Edit Profile, the user can directly modify the username and then click the Save or Cancel button, as shown in Figure 10.

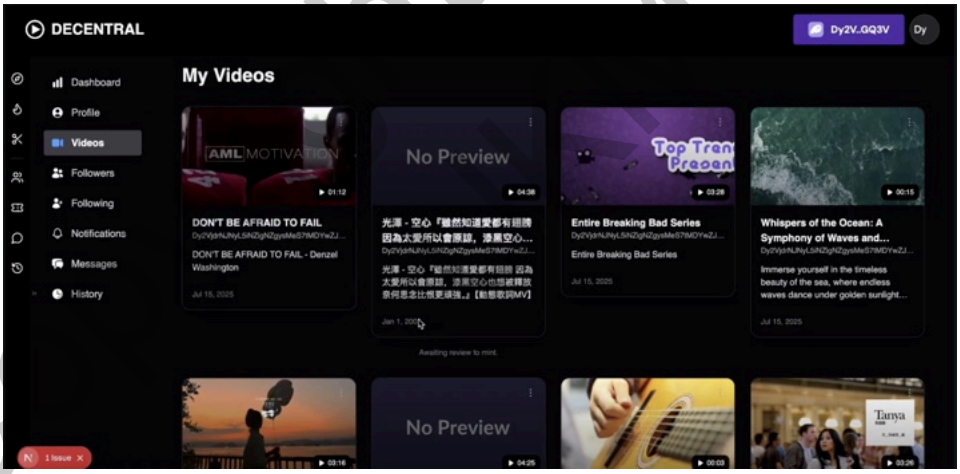

Figure 11 My upload video page

The third item is that the user can view their own uploaded video. As shown in Figure 11, some videos that do not have a cover picture are still in the back-end processing, because the back-end will call the Google Video Intelligence API to review the video and the cover, if it does not pass, the user needs to re-upload.

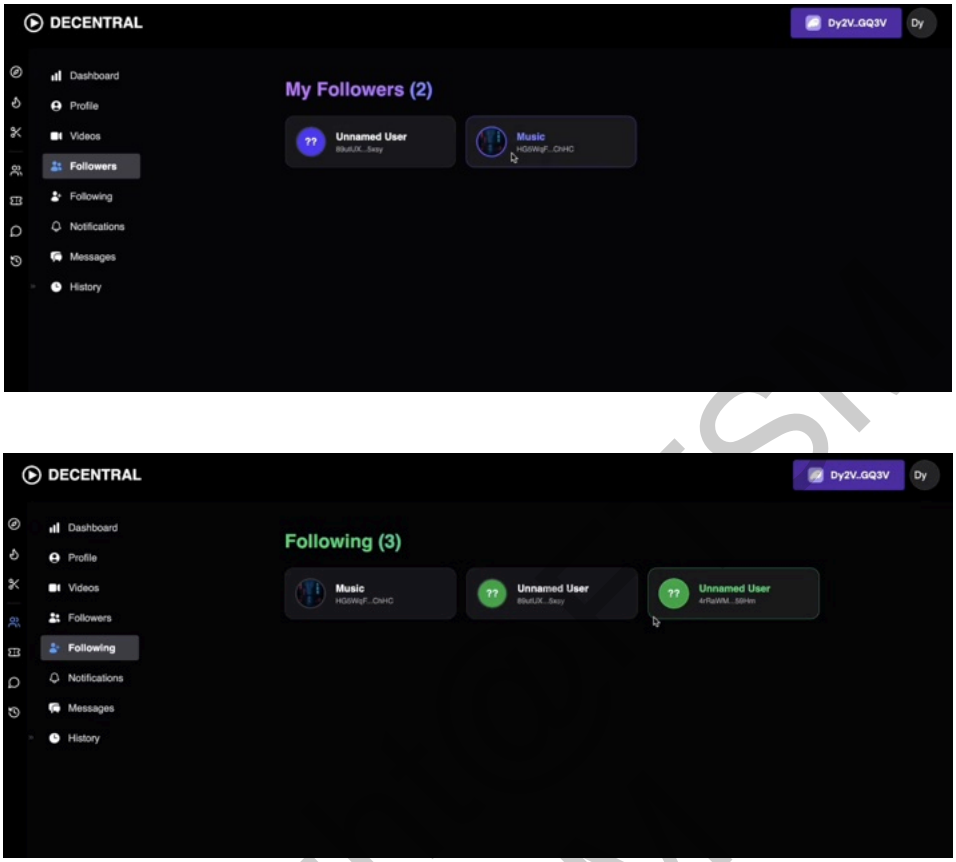

Figure 12 Follower and Following Page

Users can view their followers and also their fans, as shown in Figure 12, each user is displayed in a grid with basic information such as avatar, username and wallet address.

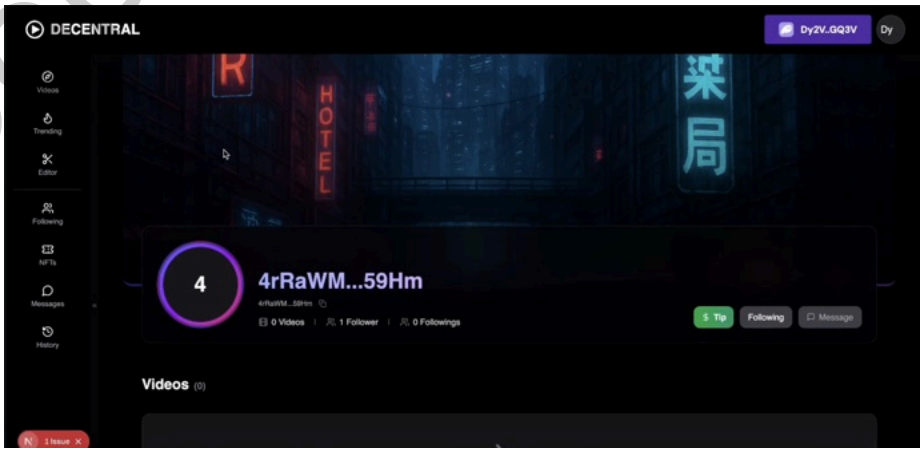

Figure 13 The creator page

When the user clicks on the avatar of the creator, the users will be taken to the creator's page, which shows the videos that the creator has already uploaded. At the top of the creator's page is a background image area where each creator can upload their favourite background image. Below that is a medium size avatar, along with basic information, wallet address, number of followers and followers and how many videos have been uploaded. On the right side are some buttons, such as a Tip button, where users can give tips to their favourite creators, a follow button what shows following status, and a private message button, where private messages can only be sent if they follow each other.

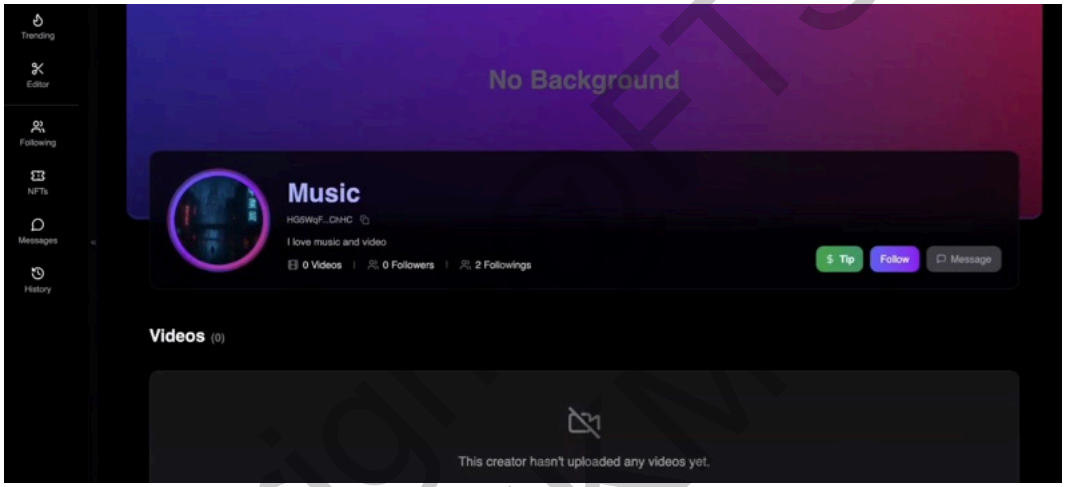

Figure 14 The creator without background

If the video creator has not uploaded a background image, it will show No Background, the video creator can upload it on this page after logging in and there will be a button to upload a background image, but it will not be visible to this user logged in right now, and the back-end and front-end will do a basic permission check. If no video has been uploaded, it will also remind that no video has been uploaded yet.

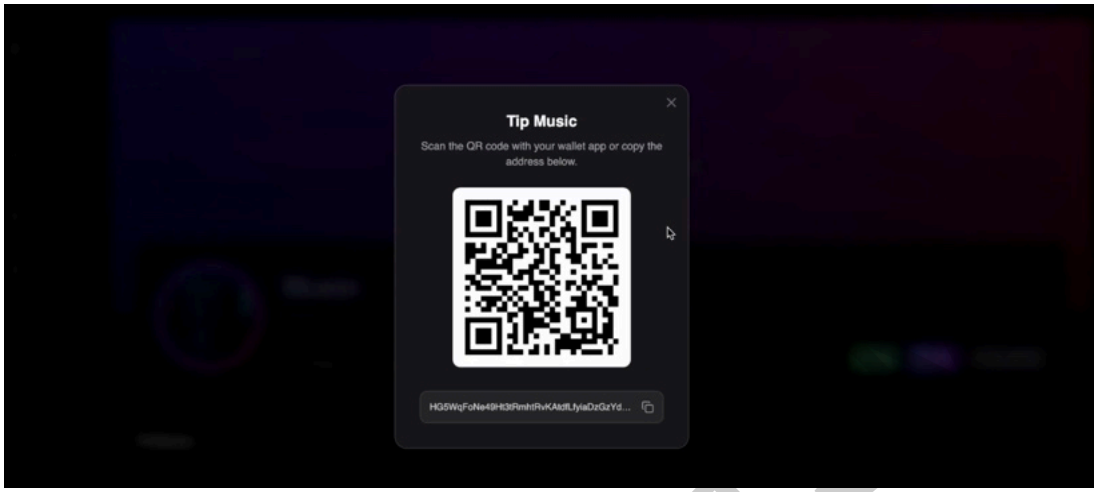

Figure 15 Tip Window

Once the Tip button is clicked, a pop-up window is displayed with a QR code representation of the wallet address, and the user can reward that creator, via their own cryptocurrency wallet, which can be tipped by scanning the code on the cryptocurrency wallet app on their mobile phone.

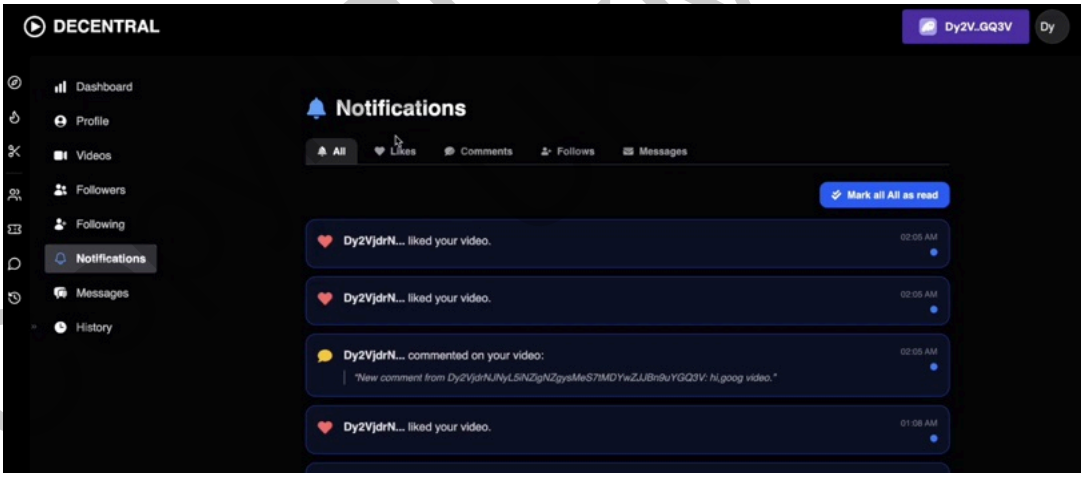

Figure 16 Notification Page

Users can also view their notifications, as shown in Figure 16, each notification message will be divided into different types, which are like, comment, follow and private message, or they can view all the notification messages at once. The blue button on the right can mark messages as read and unread messages will be highlighted in light blue.

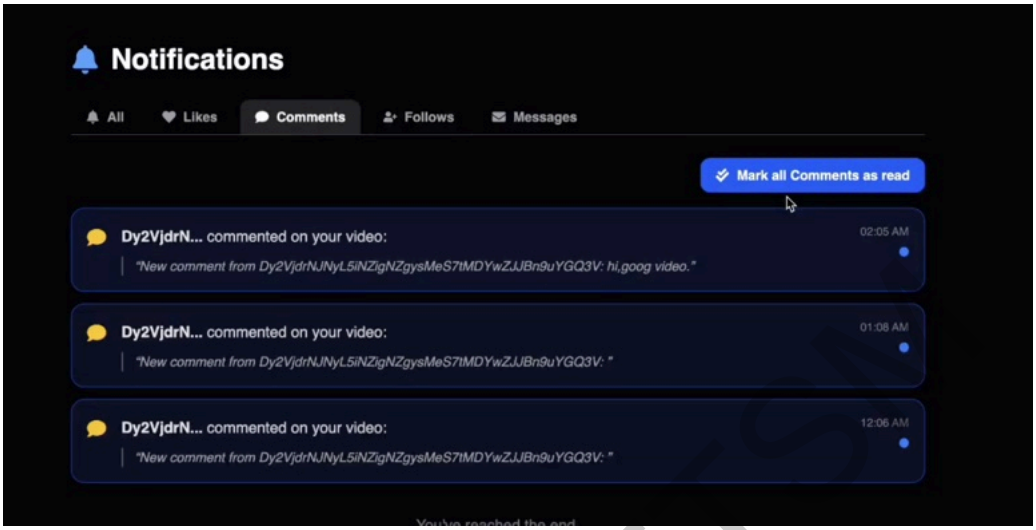

Figure 17 Comment Notification Messages

As shown in Figure 17, when you click on the different taps above, for example, the comments tab will filter out the message with only the comment.

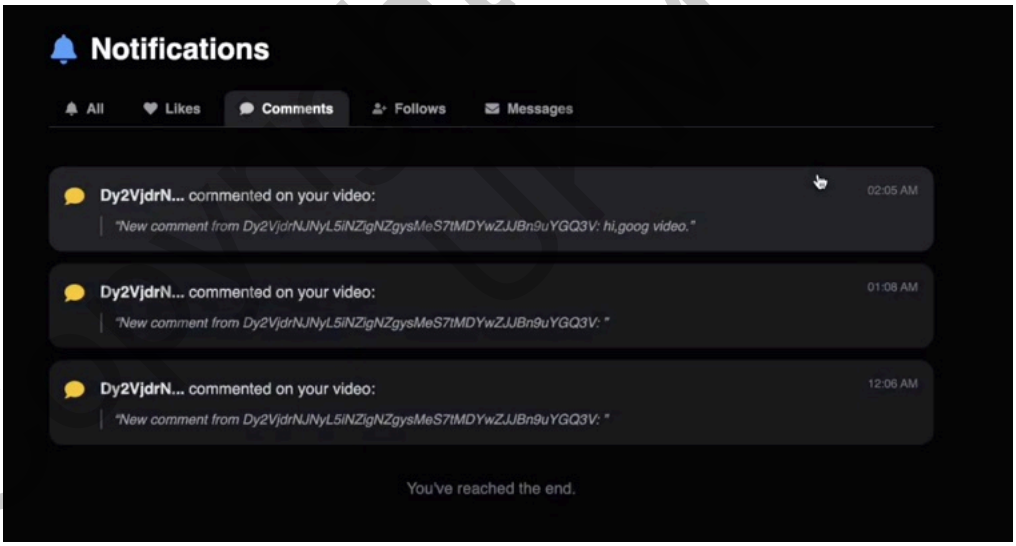

Figure 18 Showing read messages

As shown in Figure 18, when you click on that blue button, it will mark all commented messages as read, and the button will disappear, because it will only be displayed if there are unread messages in that category of notification messages.

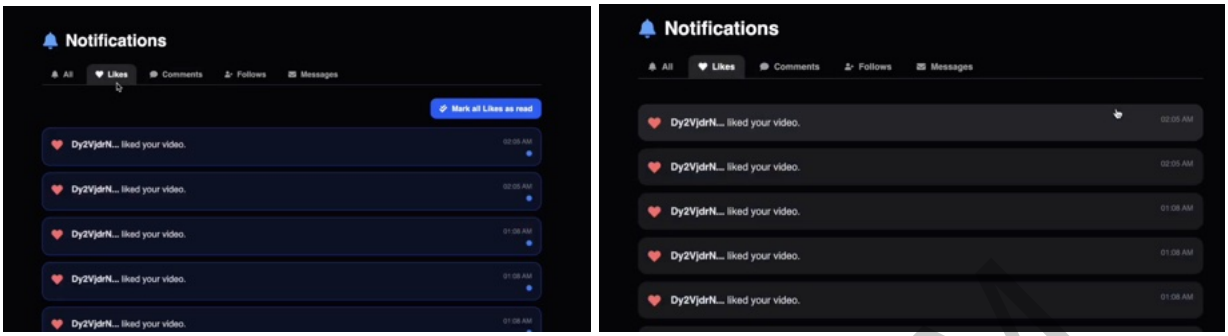

Figure 19 Like Notification Messages

As shown in Figure 19, this is the message for the like notification, indicating that a video has been liked by the user and the creator of the video will be notified. The same goes for clicking the blue button, which will mark all messages about likes as read.

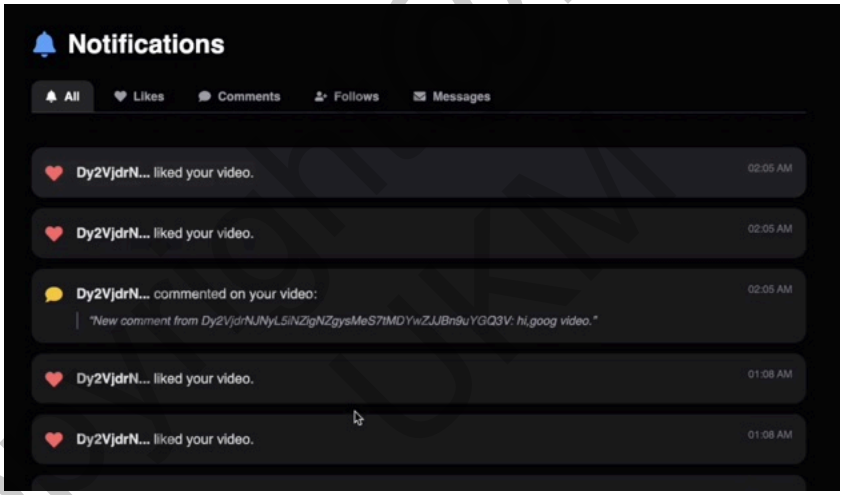

Figure 20 All read notification messages

When all messages have been read, the blue button under the all tab will also disappear.

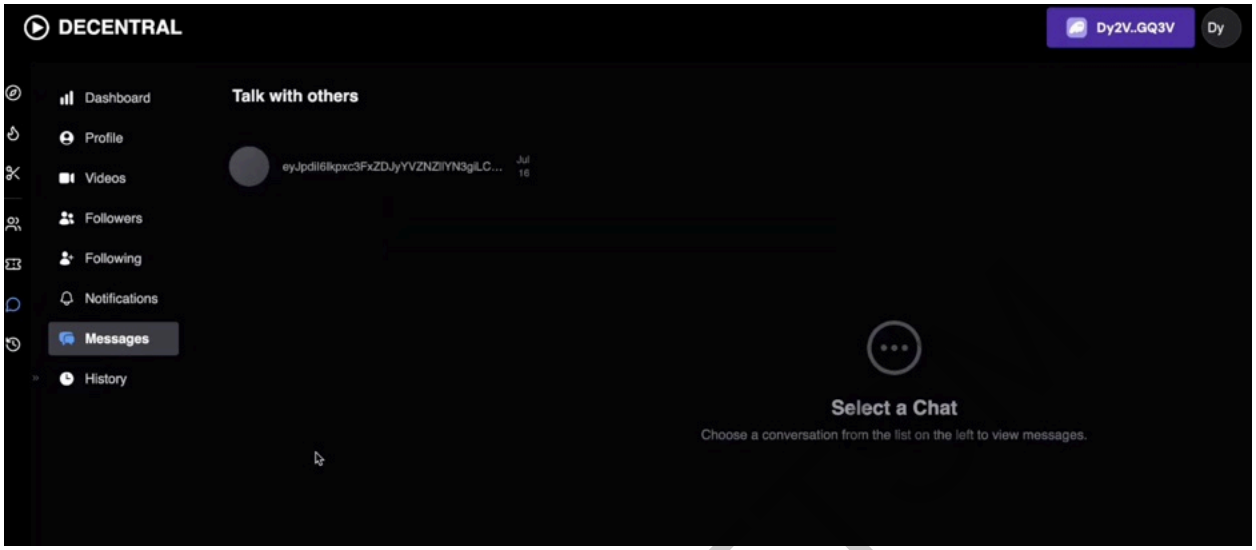

Figure 21 Private Message

Next is the tap for private messaging, this part is the bridge that connects video creators and fans. Users who follow each other can send private messages, on the left is the chat list, and on the far right is the chat window, by default there is no chat selected.

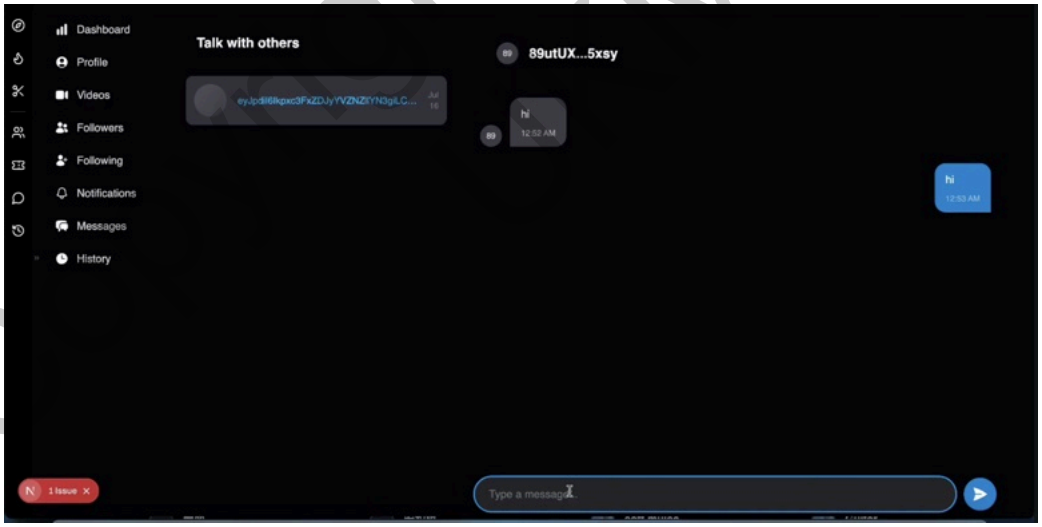

Figure 22 chat window

When the user selects one of the chat objects in the chat list, the history of messages with the chat object will be loaded on the right side, and the following is the chat input box, the user can input the content they want to send to the other party, and the other party will receive

notifications in real time, which uses the same websocket protocol(Fette, I. & Melnikov, A. 2011.) as the notification page, to ensure real-time connectivity. In order to save server resources, the whole application will only have one websocket connection, multiple different websocket messages reuse this one connection channel, in the client according to different messages distributed to different UI components. And the underlying heartbeat monitoring mechanism will be sent every 30 seconds to ensure that the connection still exists, because the backend will remove the websocket connection without messages.

Chat messages are not sent out in clear text, they are encrypted locally and sent to the backend, and then the messages are routed through the websocket to the corresponding users. Even the server doesn't know the content of the chat messages between users. How does this work? The main use of Elliptic Curve Diffie-Hellman (Practical Cryptography for developers.) which is a key agreement protocol that uses Elliptic Curve Cryptography to establish a shared secret between parties. In short, is to use the user's wallet to sign a controllable message, which is to ensure that the deterministic output, and then the message for sha256 hash operation to get a message digest, this message digest will be combined with an elliptic curve base point G to calculate a value as this user's temporary public key and then upload this calculated value to the backend. When two users want to send a private message, they will first need to get each other's public key, and then use each other's public key and the user's own computed sha256 message digest to get a shared key. This shared key will encrypt the message to be sent, both sides of the shared are the same, here is symmetric encryption now.

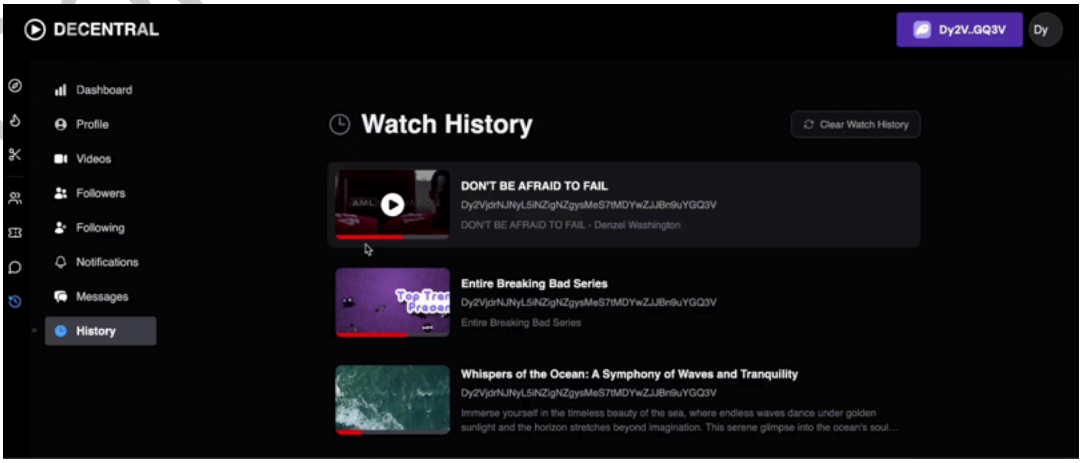

Figure 23 User Watch History

Next is the history list of videos that the user has watched, as shown in Figure 23. Each video will have a progress bar showing the user's viewing progress for that video, and clicking on the corresponding video will jump to the corresponding video page as shown in Figure 6. Here the redis bitmap is mainly used to track the viewing progress of the video, each video is converted into seconds and then each second is mapped to a bit of the redis bitmap to show whether it has been watched or not, 0 means not watched and 1 means it has been watched. Every once in a while, the front-end will send a viewing interval of this video to the back-end, and the back-end will use redis to perform a batch SetBit operation on the viewing interval of this video. If the user is watching the video for the first time, the service responsible for tracking the viewing progress will publish an event that the user has started to watch the video, and the logic of the service responsible for managing the user's viewing history will subscribe to the event via Apache Kafka, and then create a list of the user's historical viewing history and add the corresponding video. When the viewing progress reaches 80%, the service will also publish an event that the user watched 80% of the video, and the video metadata service will subscribe to this event to update the number of times the corresponding video has been watched. All the operations on the event are implemented with idempotency to ensure that the business logic is executed only once.

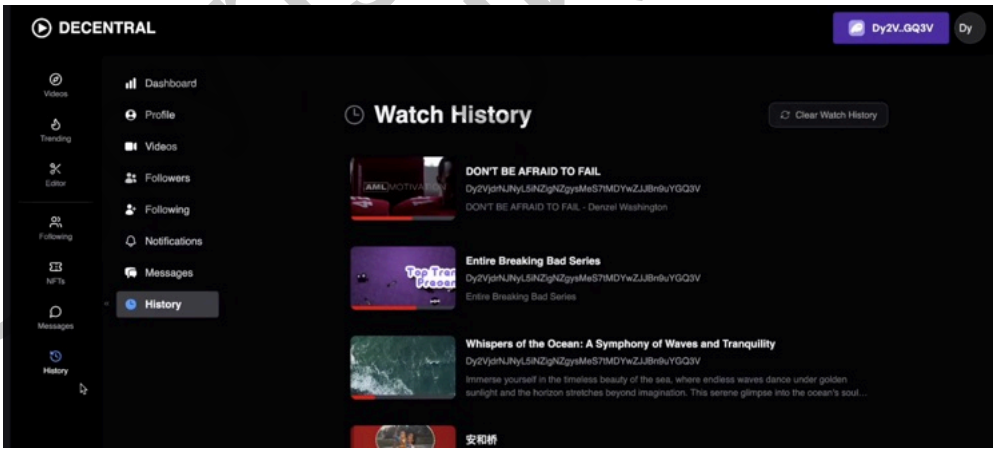

Figure 24 Expanded sidebar

In addition, the user can also expand the leftmost sidebar, as shown in Figure 24, each ui component will automatically adjust the distance between each other to ensure that the ui neat and beautiful.

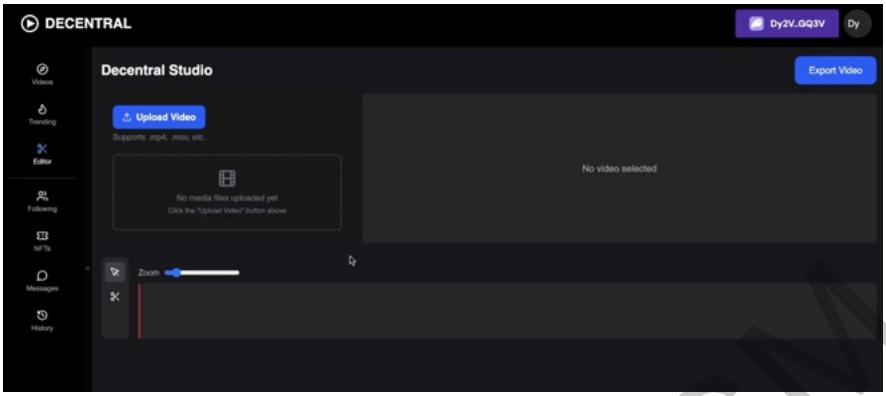

Figure 25 Video Editor Page

Users can also check the Editor tab in the side navigation bar to perform light editing operations on the video. On the top left is the loaded video to be edited, and on the right is a monitor to view the video screen. Below that is the timeline, which allows for basic cropping and selection of the video.

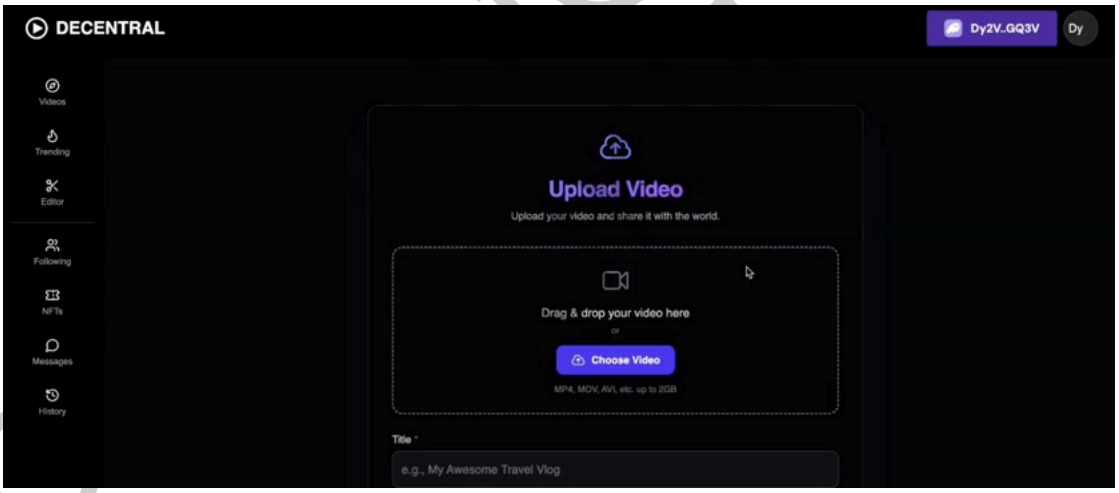

Figure 26 Video Upload Page

Figure 26 shows the page where users can upload videos. After clicking Upload Video in the menu shown in Figure 8, the user will be taken to this upload video page. First of all, users can select the video to be uploaded from the local after the user clicks the purple Choose Video button, and then set the title and other basic information of the video.

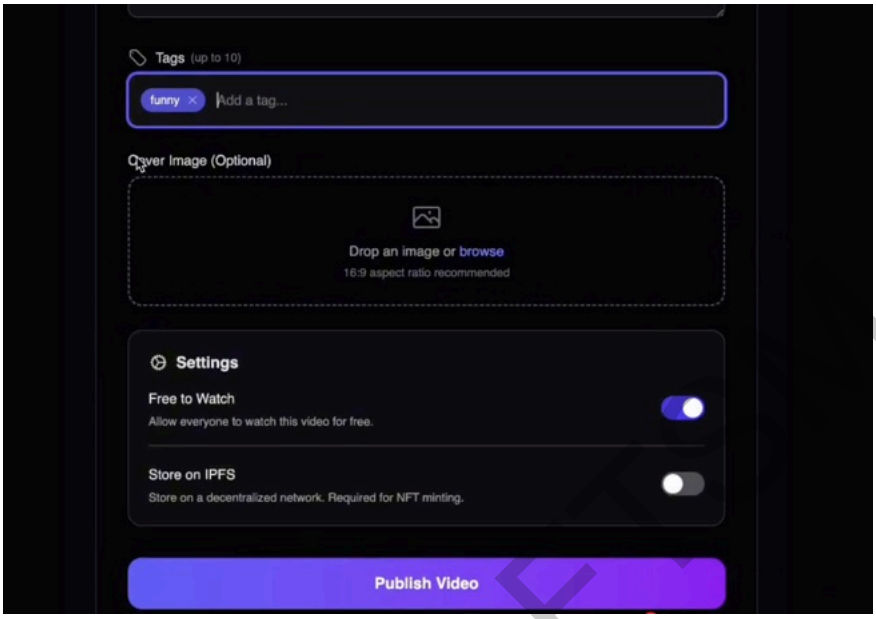

Figure 27 Video Upload Page 2

Users can also fill in the tags of the video, which is convenient to tag and categorise the video with the back-end, and can also choose the cover of the video or leave it unchecked by default, because the back-end will automatically extract a frame of the video as the cover of the video, and at the bottom, there are two options about the video, one is whether it is free to watch, and the other one is whether it is stored in the IPFS or not.The first one is to set the video as a paid one, which means that users can watch the video by buying it, which can provide revenue to the video creator. The second option, if Stored on IPFS is checked, then the video can't be deleted because this is a decentralised storage network, with no central node in charge of managing it. Here I mainly consider for the subsequent creators to mint NFTs, because NFTs need to refer to the content that can not be arbitrarily deleted so as to provide the digital copyright of the content of the proof. NFTs can be in the secondary market, for example Opensea which is one of largest NFTs trading platforms, the NFTs can constantly being resold and traded, each NFTs transactions can be set to get the beginning of the royalties via smart contract to further expand the scope of the creator's income.

In addition to this, Google's video review service is used on the back-end to carry out an initial review of the video to ensure that the content complies with the specifications.

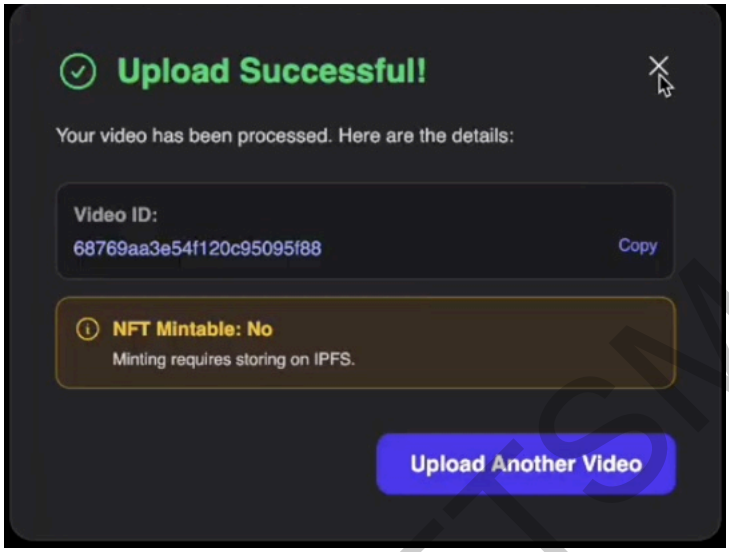

Figure 28 Successfully upload video popup window

When the user clicks publish video, as shown Figure 28, it will show the upload success pop-up window, here the back-end will adopt asynchronous processing through Apache Kafka to improve the throughput, the video will go through a series of state changes in the back-end, and finally only after it is set to publish state, it can be indexed.

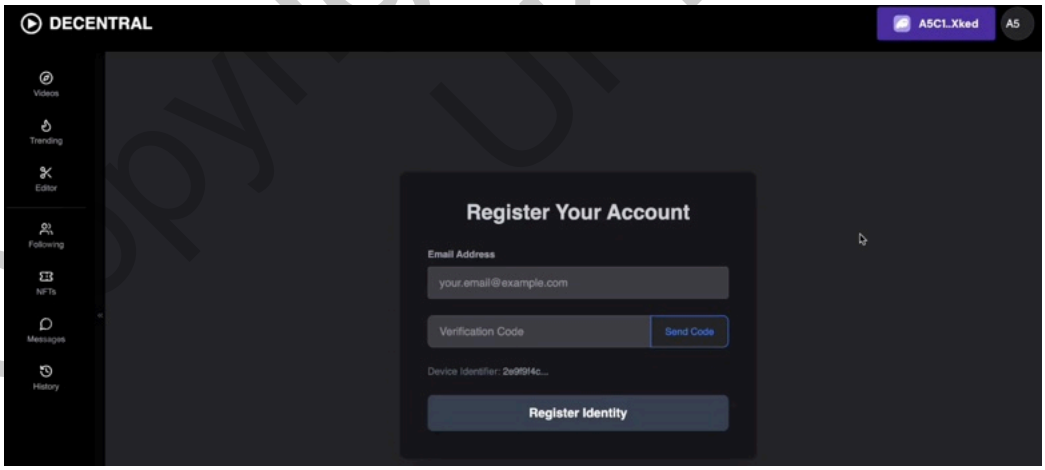

Figure 29 Register Page

By the way, if the user is the first time to click on the login item in the menu as in Figure 8, the page will automatically redirect to this registration page, mainly to prevent account abuse, because a cryptocurrency wallet is essentially a public-private key pair, which can be generated

for thousands, but not for email. So the main thing here is to make a wallet and an email bound, and the backend will send a CAPTCHA to make sure that the user controls the corresponding email. The backend will bind the wallet to the email one to one. The CAPTCHA sent is mainly stored in redis and will automatically expire after 1 hour.

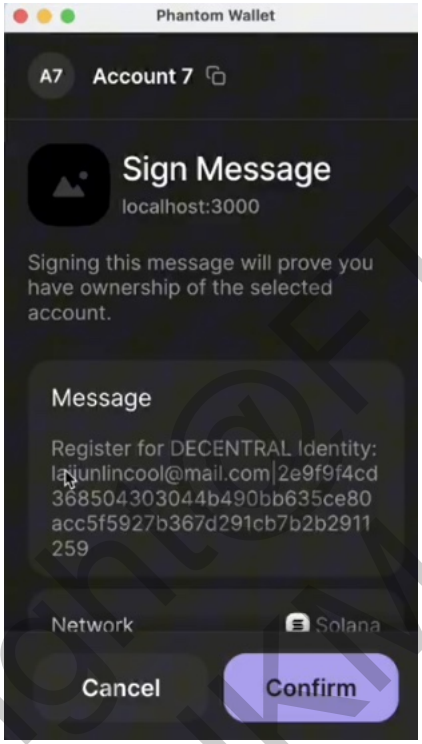

Figure 30 Phantom wallet sign message popup window

After entering the email address and verification code, click Register Identity, a pop-up window will appear as in Figure 30, asking the user to sign the contents of the message, which is mainly the hash value of the user's device and email address, and the back-end will verify that the user is in control of the cryptocurrency wallet after receiving the signed message.

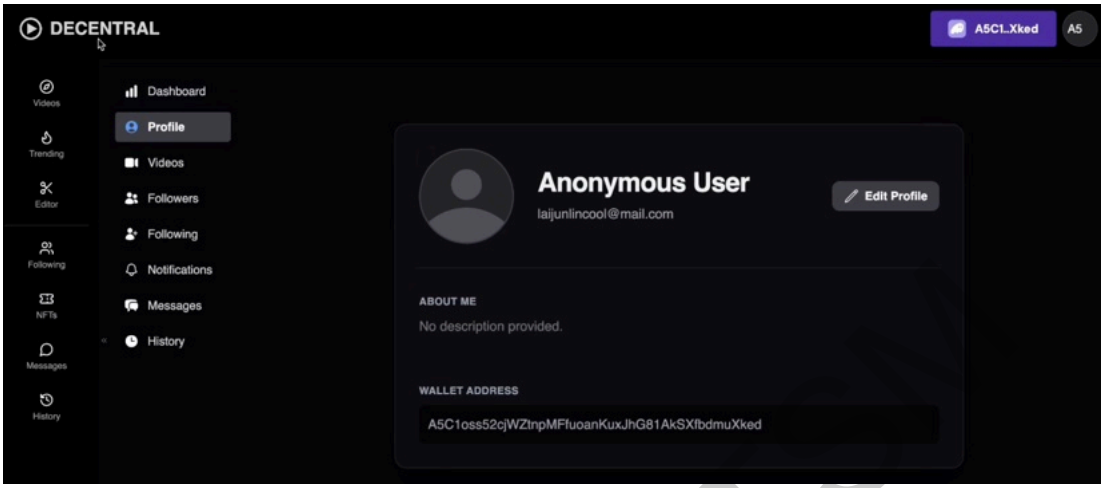

Figure 31 New user after register identity

After successful registration the profile page will automatically set the username as an anonymous user, along with relevant information.

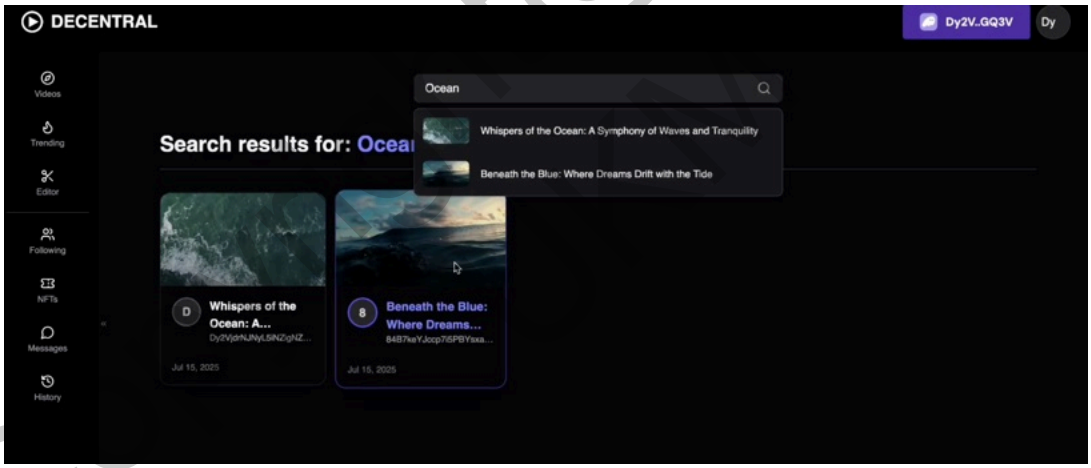

Figure 32 Search Page

Users can search for videos by entering keywords into the search input box on the main video page as shown in Figure 2, which will return relevant videos to the users as shown in Figure 32, so that the user can quickly find the videos they are interested in once there are more videos on the platform. Here the main use of elasticsearch search function, and responsible for the search logic of the service will subscribe to the event of the video release, and then update their own search engine, which largely decoupled, if the video is deleted, it will receive the event of the video is deleted, and then delete their own index of the internal video metadata.

Usability Testing

This is a relatively new area, especially where cryptocurrency wallets are concerned, I asked some of my friends to use this software I have developed, and I have had some feedback from some users that the part of cryptocurrency wallet operations involved makes them feel complicated and difficult to understand, and that this will lose users to some extent, who just want to be entertained on video, not the complicated cryptocurrency wallet operations involved. There are also some users who feel that cryptocurrency is particularly new and looks awesome, so they have a desire to try it out.

Suggestions for Improvement

In order to expand the target user group, in the future I will hide away the details involving the blockchain, and try to be roughly the same as the traditional platform in operation, only the underlying technology is to use the blockchain to protect the user's content and empower the content.

In order to expand the interactive depth of the content in the future, I will also introduce the community function, video creators can form their own community to increase the cohesion of their own fans, as well as real-time live streaming function. All of these features will be coupled with a payment mechanism to generate more revenue for the creators, which will be supported by the technology of NFTs. For example, a video creator released a series of NFTs, only buy the creator's NFTs can enter the relevant community, while the fans who bought the creator's NFTs can also be resold in the secondary market, but also for the fans to increase revenue, if the future of the creator get more traffic, then the price of the hands of the NFTs will be naturally high, and the fans will get a substantial profit, belongs to the early investment for liked creators.

CONCLUSION

Over the course of three to four months of focused development and intensive study of technical documentation and design principles, I successfully built a complete Web3-based video sharing platform. Through this process, I gained deep insights into both the technical and architectural aspects of the project.

The front end of the platform is developed using React and Next.js, while the back end is primarily implemented in Go(The Go Authors. 2024.) and Java. Apache Kafka serves as the communication backbone between microservices, enabling asynchronous messaging and service decoupling.

For data storage, the system uses MongoDB to handle user-related data with flexible schemas, and PostgreSQL to manage entities requiring strong transactional guarantees. Video content is stored on Google Cloud and also distributed via IPFS, ensuring decentralization and content persistence.

The architecture follows the principle of interface-driven design, where services depend on abstractions rather than concrete implementations. Each service has a clearly separated handler layer, which deals with external interfaces (e.g., HTTP requests), and a service layer, which encapsulates core business logic.

Services can publish domain events, while other interested services subscribe to corresponding Apache Kafka topics to consume those events and update their own domain data accordingly. For cases where real-time data access is required, gRPC is used to facilitate efficient synchronous communication between services — for instance, the video management service may query the user service to retrieve user details.

While the current implementation achieves the core functionality, there is still significant room for optimization and scalability improvements. I look forward to iterating on the system and enhancing it further in the future.

ACKNOWLEDGEMENTS

I would like to express my heartfelt appreciation to Prof. Madya Dr. Yazrina binti Yahya for her invaluable guidance and support throughout the development of this project. Her expert advice, encouragement, and insightful feedback were essential in helping me stay focused and motivated during this independent journey.

I am deeply grateful to my family for their constant love, understanding, and emotional support, which gave me the strength to complete this project with confidence.

To my friends, thank you for your encouragement, patience, and for always being there when I needed motivation or a moment of relief. Your presence made the process less overwhelming and more meaningful.

REFERENCES

IPFS Docs. n.d. Welcome to the IPFS docs. [Online] Available at: https://docs.ipfs.tech/ [29 June 2025].

Fette, I. & Melnikov, A. 2011. *The WebSocket Protocol*. RFC 6455. Internet Engineering Task Force (IETF). https://datatracker.ietf.org/doc/html/rfc6455 [11 June 2025].

MongoDB, Inc. 2024. *MongoDB: The developer data platform*. https://www.mongodb.com/ [20 June 2025].

Redis Ltd. 2024. *Redis: The in-memory data platform*. https://redis.io/ [25 June 2025].

The Go Authors. 2024. *Effective Go*. https://go.dev/doc/effective_go#concurrency [30 June 2025].

The PostgreSQL Global Development Group. 2024. *PostgreSQL: The World's Most Advanced Open Source Relational Database*. https://www.postgresql.org/ [1 July 2025].

The Apache Software Foundation. 2024. *Apache Kafka*. https://kafka.apache.org/ [24 June 2025].

Vercel. 2024. *Next.js by Vercel - The React Framework for the Web*. https://nextjs.org/ [1 July 2025].

Practical Cryptography for developers. *ECDH Key Exchange.* [online] Available at: https://cryptobook.nakov.com/asymmetric-key-ciphers/ecdh-key-exchange [2 July 2025]