International Journal of Advance and Innovative Research
Volume 12, Issue 2 (XXII): April - June 2025
ISSN 2394 - 7780

STREAM SYMPHONY: ONE STOP SOLUTION FOR STREAMING USING RTMP PROTOCOL

Nil Mani[1], Pranjal Srivastava[2] and Dr. Sourabh Jain[3]
[1,2]Department of Computer Science and Engineering Indian Institute of Information Technology Sonepat Haryana, India
[3]Assistant Professor, Department of Computer Science and Engineering Indian Institute of Information Technology Sonepat Haryana, India

ABSTRACT

Stream Symphony is the live streaming solution made specifically for content creators everywhere, from LinkedIn to YouTube and Facebook. Given how the digital landscape is constantly shifting in which streamed live content will always play a great role in attracting audiences and passing over information, Stream Symphony came into being to answer the call for intuitive, secure, high-performance solutions for growing business needs. This paper demonstrates the overall architecture of Stream Symphony, unveiling the user-centric design, dynamic styling and media streaming capabilities. The key contributions in this paper included password hashing using Bcrypt towards secure user authentication by minimizing data breach-related security risks. The system employed low- latency streaming through Real-Time Messaging Protocol (RTMP) and WebSocket to stream data at very high speed. In addition, the new model, usability and interactivity have been introduced such that a user can very easy handle audio and video streams. Future enhancement incorporates the introduction of complex features such as user dashboards, revenue-generating models, and instruments for real-time engagements that broach new scalability and security issues in a solution. In a nutshell, Stream Symphony represents an innovative turn in live streaming technology regarding a viable, user-oriented platform that will respond to ever-changing requirements from digital content developers.

Keywords: *RTMP, FFmpeg, Media recording, Video encoding, Real time communication*

I. INTRODUCTION

A. Background Study:

Digital media has changed the consumption and sharing of content. Live video streaming is an integral part of modern day to day communications, whereby an event can be broadcast over the internet in real time during a meeting, a personal moment, or any other incident. Market leaders in live streaming comprise YouTube, Twitch, LinkedIn, and Facebook, among others, as they enable users to connect with a global audience in an instant. However, many premium features, such as customized RTMP destination

URLs and stream keys, are behind paywalls with many current live streaming services, that significantly limit accessibility, increase costs for content creators and broadcasters, and eliminate economical solutions [1].

B. Problem Study:

Custom RTMP destination URLs and stream keys on existing popular live streaming platforms such as Stream Yard, Vimeo, and Stream Labs are offered for its premium services. This limits those who cannot pay for the astronomical costs associated with such features and, by extension limits their capacity to stream live video content on different platforms. Additionally, these platforms often impose watermarks or logos on the streamed video, which can detract from the viewing experience and compromise the branding efforts of content creators.

C. Rationale:

The need for a free, accessible, and high-quality live streaming solution is evident. By providing a platform that offers custom RTMP destination URLs and stream keys without any associated costs, content creators can enhance their streaming capabilities and reach a broader audience. Furthermore, ensuring that the platform does not impose watermarks or logos on the video will enhance the professional appearance of the streams, benefiting both individual creators and businesses [2].

D. Definition of Concepts

- RTMP (Real-Time Messaging Protocol): A protocol used for live video streaming over the internet, which ensures low latency and efficient data transfer.
- FFmpeg: A free and open-source software project that provides a complete solution to record, convert, and stream audio and video.
- Media Recording: The process of capturing and storing audio and video data in real-time.
- Video encoding: The method of converting raw video into something that can be streamed or stored.

- Real-Time Communication: Facilitates the transfer of data in real-time, thereby allowing users to interact instantly.

E. *Research Questions*

There are particular challenges relating to protocol range present in video transmission that the direct engagement of live streaming technology has with social networks. In our work, we center these considerations around user stream conversion to real-time processable format that can then be broadcasted. Specifically, we capture feeds as streams by users, which cannot be sent directly over TCP as there are specific inherent limitations in doing so; we thus use UDP for transporting video. However, because RTMP runs over TCP, here arises a great need: the user stream needs to be converted to a binary that can be sent to a Node.js server. This is a procedure for the real-time recording of the stream and encoding, all at once, as the binarized file is properly transferred to the Node.js server. The encoded data are then forwarded through an RTMP server using FFmpeg for proper live streaming on the server-side. The approaches in solving these problems are discussed in detail about architecture and procedures for efficient capturing, conversion, and transmission of video streams that support low latency but high-quality video. Our work will help further the development of solutions provided for live streaming as a working approach toward real-time media handling across different protocols.

F. *Objectives*

- Design a user-friendly interface for a live streaming application using React, HTML, CSS, and JavaScript.
- Implement dynamic styling and layout to enhance the responsiveness and beauty of the UI.
- To integrate media streaming and recording functionality using the Media Recorder API.
- Providing features to stream on/off audio and video streams during streaming.
- To ensure secure authentication and authorization of users with an aesthetically pleasing interface.
- To ensure significant decrease in latency compared to other platforms, enhancing the live streaming experience for content creators and broadcasters.

G. *Significance of the Study*

The Stream Symphony would cover the gap in the present market of free high-quality live streaming services. With the offer of providing free stream keys and customizable RTMP destination URLs without forcing any watermarks or logos on content creators' streams, Stream Symphony offers enhancement to the experience of streaming. It contributes to the advancement of streaming technology because it democratizes the access of premium streaming features among a wide population of users-from individual creators to businesses and educational institutions [3].

II. KEY CONTRIBUTION

The main contribution in this project are as follows.

A. *User Interface (UI) Design:*

- Developed a user-friendly interface for a streaming application using React, HTML, CSS, and JavaScript.
- Utilized modern design principles to enhance the usability of the application.
- Organized elements such as buttons and video display in a visually pleasing and intuitive layout.

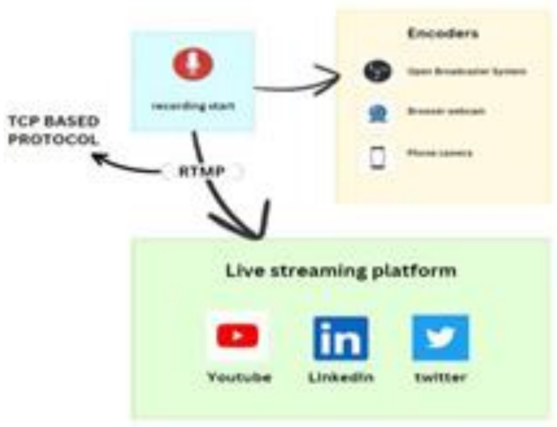


Fig. 1: Introduction of live stream process

B. ***Dynamic Styling and Layout:***

- Implemented dynamic styling to improve the responsiveness and visual appeal of the UI.
- Utilized flexbox layout for flexible and efficient arrangement of UI components.
- Applied hover effects to buttons to enhance user interactivity and feedback.

C. ***Media Streaming and Recording:***

- Integrated media streaming and recording functionality using the Media Recorder API.
- Enabled users to start and stop recording video streams with configurable audio and video settings.
- Implemented real-time streaming of binary data to a server using WebSocket's for further processing or broadcasting.

D. ***User Interaction Features:***

- Implemented features to toggle audio and video streams on/off during streaming.
- Provided feedback to users through alerts and button text changes to indicate the status of streaming and recording operations.
- Ensured seamless user experience by dynamically updating UI elements based on user actions and media stream status.

E. ***Server Communication and Data Handling:***

- Established communication with a server using the Socket.IO library to transmit binary data streams.
- Implemented event handlers to capture and process binary data chunks generated during media recording.
- Facilitated server-side processing or broadcasting of recorded streams for further analysis or distribution.

F. ***Overall System Functionality and Usability:***

- Combined UI design, media handling, and server communication to create a comprehensive streaming application.
- Prioritized usability and user experience by offering intuitive controls and clear feedback mechanisms.
- Contributed to the advancement of streaming technology by providing a functional and user- friendly solution for streaming applications.

III. LITERATURE REVIEW

Live video streaming using real time messaging protocol (RTMP), it is a protocol for online video streaming, This literature review mentions evolution of techniques used in video streaming as well as authentication using password hashing, RTMP still hold its value despite of having newer options like HLS because of its ability to provide video with lower latency, in the following given below table we will learn more about various techniques, algorithms, advantages, limitations etc. in live video streaming and processing as mentioned in Fig. 3.

IV. PROPOSED MODEL

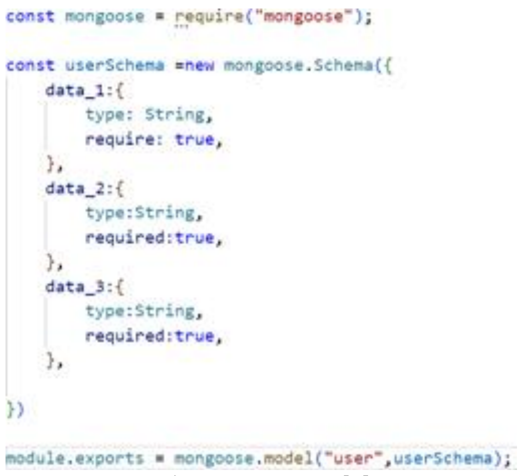
Our model can be classified into two parts that are user registration and live streaming part, initially we will study about user authentication then followed by live stream portion of our project:

A. *User Authentication*

To execute the user Authentication initial what we need to do is to create database where our client's data will be stored here, we are using MongoDB database, moving further on the coding we create a user model it is a structure of a data stored in the database, its syntax can be explained in Fig. 2. Here we are done with the database part, now let's understand the functionality of how users get authenticated, we get data that is username and password from the client, now for the security reason, password is hashed and stored in the database along with username.

Understanding how hashing of user's password is done before we should know why we can't we store user's password as it is in the MongoDB database:

It's generally an easy way to store the password as it is and to track it, but it will lead to huge security concerns. When hackers attack a database, attackers can access user login details easily. This is especially bad news for people who use it. Sometimes you can use special passwords all the time, yet, even so, you are not safe entirely due to the thing known as "rainbow table attacks". This is where criminals utilize already available hash values lists to identify the ones, they have just taken from you.



```
const mongoose = require("mongoose");

const userSchema =new mongoose.Schema({
    data_1:{
        type: String,
        require: true,
    },
    data_2:{
        type:String,
        required:true,
    },
    data_3:{
        type:String,
        required:true,
    },

})

module.exports = mongoose.model("user",userSchema);
```

Fig. 2. User model

*B. **Introducing Bcrypt***

Bcrypt is a library which is used popularly to hash passwords. It relies on the Blowfish cipher and a technique called "salting" to hash passwords. Salting involves adding a random value (called the "salt") to the original password before hashing it. This ensures that even if two users happen to have the same password, their hashed values will be different because of the unique salt added to each one. Bcrypt remains a reliable choice for password hashing, as it is more resistant to brute-force attacks because of its time complexity. However, Acar et al. [4] discuss that newer algorithms like Argon2 provide better memory-hard functions, making it more secure against more sophisticated attacks.

Work Algorithms Advantages Limitations/Disadvantages Zhongxiao Luo et al. [5] Live SR for providing universal streaming Improvement in QoE to 65.5% and video quality to 5.7% and achieved frame rate of 30 Frame per second. The difficulty in training the SR model at low cost and maintaining the quality in real time.

Iván Santos-González et al. [3] Use of titles within modern video codecs enabling bandwidth efficient adaptive streaming. Saving bitrate from 45% to up to 65% and serve as a basis for advanced technique. The generalizability of the result of saving bitrate across various network conditions, device type or used behaviors.

Table I. Previous work done in this field.

Work	Techniques/Algorithms	Contributions/Advantages	Limitations/Disadvantages
Zhongxiao Luo et al. [5]	Live SR for providing universal streaming	Improvement in QoE to 65.5% and video quality to 5.7% and achieved frame rate of 30 Frame per second.	The difficulty in training the SR model at low cost and maintaining the quality in real time.
Iván Santos-González et al. [6]	Use of titles within modern video codecs enabling bandwidth efficient adaptive streaming.	Saving bitrate from 45% to up to 65% and serve as a basis for advanced technique.	The generalizability of the result of saving bitrate across various network conditions, device type or used behaviors.
Liyang Sun et al. [7]	Dynamic models and optimization strategies to increase QoE	Design of low latency live streaming to establish QoE upper bound as a function of the allowable end to end latency.	QoE metrics used for evaluation in not mentioned
Yunlog Li et al. [8]	Use of Adaptive Bitrate algorithm.	Achieves an average QoE improvement of 39.7% and 37.9%	Doesn't provide insight into scalability and adaptability in different network conditions.

Bo She et al [9].	Using Nginx and FFmpeg, facilitating live video broadcasting.	Test the various functionalities and overall performance of the system to verify the effectiveness of the system.	The system relies heavily on specific technologies such as Nginx, FFmpeg, and HDFS. While these technologies are widely used, the paper does not explore alternative solutions or discuss the implications of technology selection.
Iván Santos-González et al.[10]	RTSP vs WebRTC	Comparison of different streaming protocols.	The paper details building the apps but doesn't analyze the actual performance comparison or compare to existing solutions.
This work	Socket.io, RTMP, node js	Contributed to the advancement of streaming technology by providing a functional and user-friendly solution for streaming applications.	N/A

Liyang Sun et al. [7] Dynamic models and optimization strategies to increase QoE Design of low latency live streaming to establish QoE upper bound as a function of the allowable end to end latency. QoE metrics used for evaluation in not mentioned Yunlog Li et al. [8] Use of Adaptive Bitrate algorithm. Achieves an average QoE improvement of 39.7% and 37.9% Doesn’t provide insight into scalability and adaptability in different network conditions.

Bo She et al [9]. Using Nginx and FFmpeg, facilitating live video broadcasting. Test the various functionalities and overall performance of the system to verify the effectiveness of the system. The system relies heavily on specific technologies such as Nginx, FFmpeg, and HDFS. While these technologies are widely used, the paper does not explore alternative solutions or discuss the implications of technology selection.

Iván Santos-González et al. [10] RTSP vs WebRTC Comparison of different streaming protocols. The paper details building the apps but doesn't analyze the actual performance comparison or compare to existing solutions.

This work Socket.io, RTMP, node.js Contributed to the advancement of streaming technology by providing a functional and user-friendly solution for streaming applications. With the help of cryptographically secure hash function, it makes it harder for the attackers to use brute- force techniques for guessing passwords, because bcrypt slows down the hashing process thus making it computationally expensive.

C. *Understanding bcrypt's workings*

We will breakdown into key components and understand it,

- Generation of salt: This is the first step, where the unique salt is generated for every password, it is a random value which is merged with the password, this is to make sure that two entries having the same password get unique identification because of unique salt attached to it.
- key Stretching: Hashing Process can be slowed down with the help of the process called key stretching, in this cryptographic function is applied many times, it is done intentionally, so that the hackers have to spend more time in generating various password combinations, work factor and cost factor are two important factors which governs the number of times the hash functions to be iterated, higher work factor will eventually increase the time, thus security of hashed password can be increased.
- Password hashing: After the generation of salt and work factor, bcrypt combines these values with the client's password and passes it through the Blowfish cipher, as an output cryptographic hash is generated that represents the password and salt. The hash length is fixed that is irrespective of the password length hash length will remain constant, this is a key property for comparing password hashes and storing it securely.

ISSN 2394 - 7780

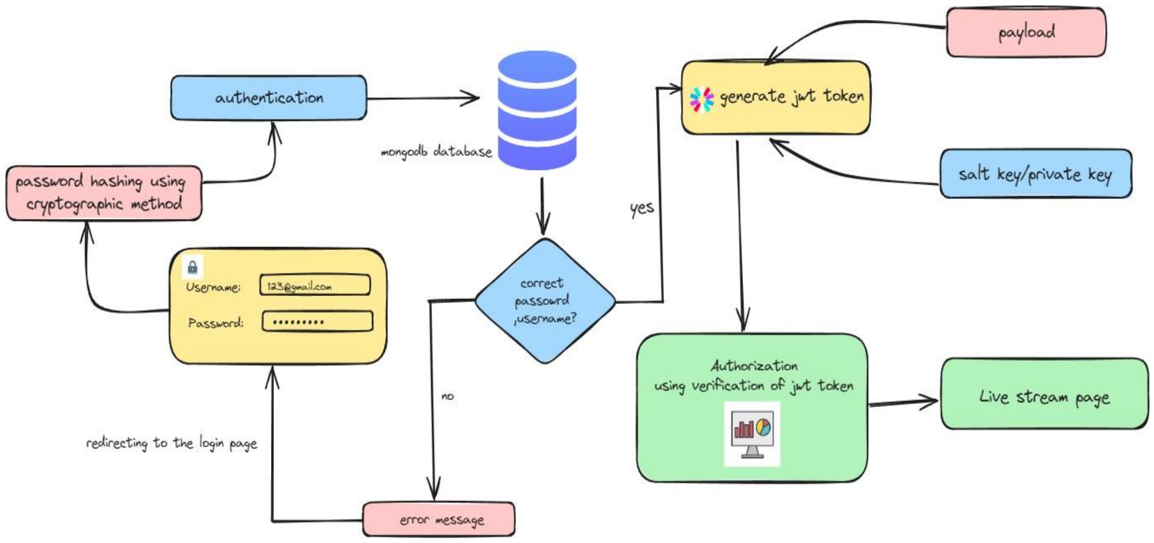


Fig. 3: Part I

- **Verification Process:** During login process, when a user logins with credentials that is username and password, the password is hashed with the same steps as discussed earlier , now the new hashed password is being compared with the one stored in the database, if it matches authentication is successful other it is failed, that's how authentication is implemented, by undergoing series of steps, with the help of extra hashing security layer is applied hence preventing them from being attacked by the hackers.

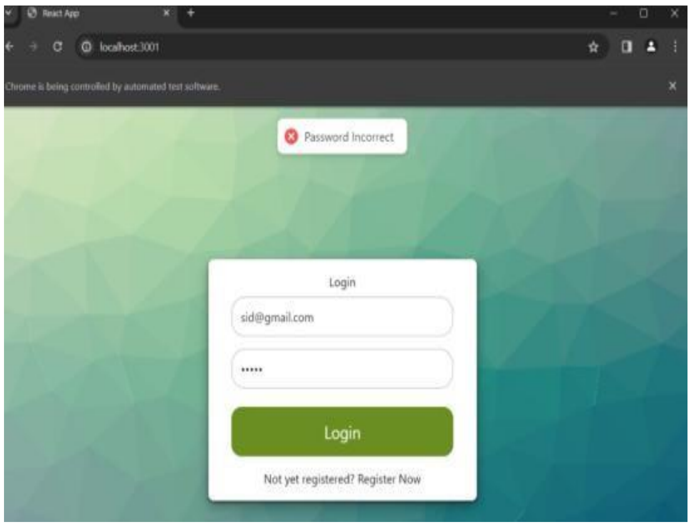

React App
localhost:3001
Chrome is being controlled by automated test software.
Password Incorrect
Login
sid@gmail.com
•••••
Login
Not yet registered? Register Now

Fig. 4. Incorrect Credential

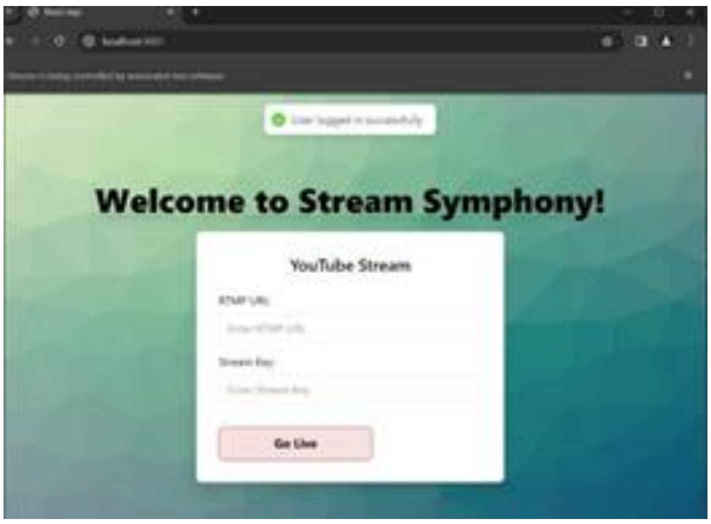


Fig. 5. Login with correct credentials

D. *User Authorization*

JSON web token is used to provide authorization to the authenticated user.

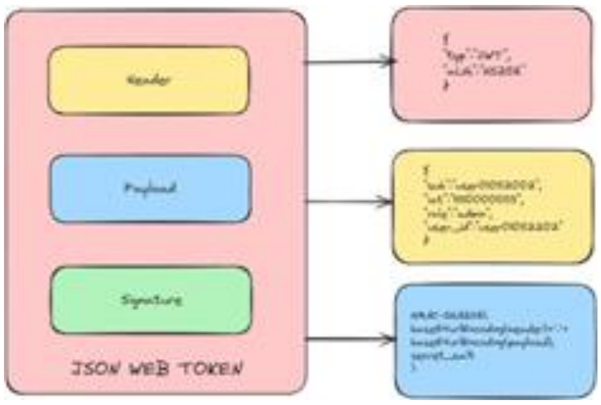


Fig. 6. Structure of JSON Web Token

Users sign-in using their credentials i.e. username and password.

Authentication server verifies the credentials and issues a jwt signed using either a secret salt or a private key.

Client uses the JWT to access protected routes or resources by passing the generated JWT in the HTTP Authorization header.

E. *Live Streaming:*

Authenticated user enters the rtmp url and stream key which is provided by the platform on which they want to stream their video live.

We can decide the process of going live in the following parts:

- Capturing your video and audio: script.js uses navigator. media Devices.getUserMedia function to access your webcam and microphone it will capture audio and video from the browser.

ALGORITHM:

```
const userVideo = document.getElementById('user-video') const startButton = document.getElementById('start-btn') const state = {media: null}

const socket = io() startButton.addEventListener('click', () => {

const mediaRecorder = new MediaRecorder(state.media, { audioBitsPerSecond: 128000, videoBitsPerSecond: 2500000,

framerate: 25 }) mediaRecorder.ondataavailable = ev => socket.emit('binarystream', ev.data) mediaRecorder.start(25)

})

window.addEventListener('load', async e => {state.media = await navigator.mediaDevices.getUserMedia({ audio: true, video: true

})

userVideo.srcObject = state.media})
```

The above code also shows configuration of the video that is audio bitrate per second and video bitrate per second which needs to be sent to the server.

- Preparing the stream (encoding): It uses a FFmpeg which stands for Fast Forward Moving Picture Experts Group. It is a free and open-source software project that provide many features for audio and data processing, it takes the raw video as well as audio from the browser and does encoding that is converting the video into the format which is suitable for live streaming.
- Sending the stream (uploading): we used socket.io library to have a real time connection between server and the browser, when we click on go live button the encoded data is transferred to the server in chunks using socket.io connection.

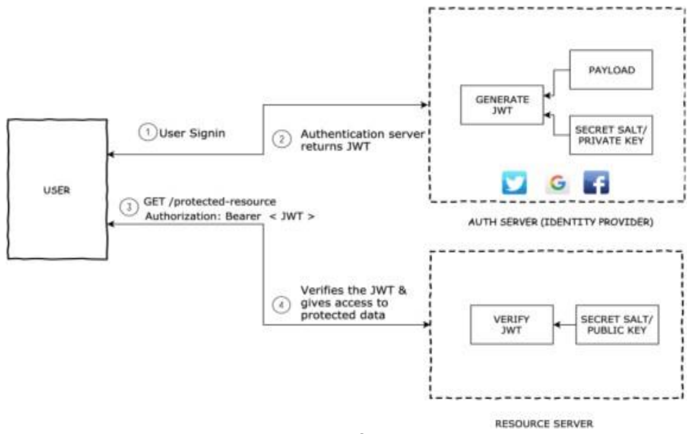

USER
1 User Signin
2 Authentication server returns JWT
3 GET /protected-resource Authorization: Bearer < JWT >
4 Verifies the JWT & gives access to protected data
GENERATE JWT
PAYLOAD
SECRET SALT/ PRIVATE KEY
AUTH SERVER (IDENTITY PROVIDER)
VERIFY JWT
SECRET SALT/ PUBLIC KEY
RESOURCE SERVER

Fig. 7. Diagrammatic representation of authorization using Json web token

- Server-side processing (receiving and forwarding): The FFmpeg receives the encoded video and then forwards it to the final streaming destination which is specified by rtmp url and stream Key. Takahashi et al. [11] demonstrated the effectiveness of FFmpeg in live streaming applications. The integration of FFmpeg in our system ensures efficient encoding. Cetin et al. [9] demonstrated that using GPU acceleration with FFmpeg significantly decreases encoding time, which could further boost the performance of Stream Symphony in future iterations.

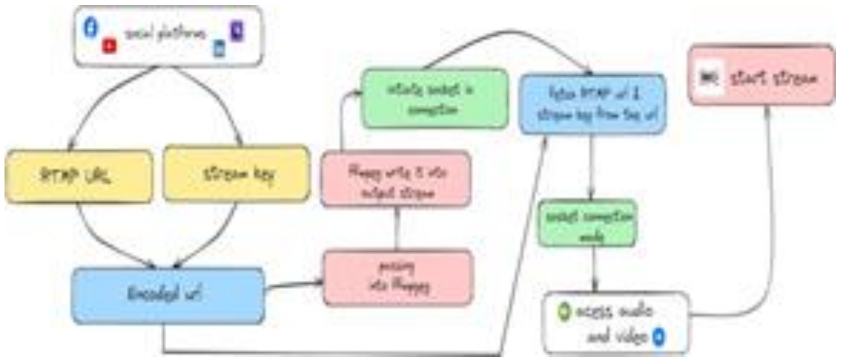


Fig. 3. Part II

- RTMP: It stands for Real time messaging protocol. It is widely used for live video streaming over the internet. A RTMP stream works by taking huge information records, like a video, and chopping them into small parcels. The bundles are sent one by one from an encoder and after that put back together in this process latency and buffering gets minimized, providing a better user experience while streaming their video live, it may be a issue for streaming a video over a larger scale for TCP based protocol, but its ability to provide video with lower latency makes it a significant tool as compared to modern protocols that make real time interaction much better. RTMP remains a widely-used protocol for streaming with low latency [10].

V. FUTURE SCOPE

1. **User Dashboard:** Integration of user dashboard show analysis and the quality of video stream and display of several details like number of views, comments, likes etc.
2. **Platform Integration:** Predefined platform specific live video where users need not to enter the rtmp url and stream key in our service platform.
3. **Monetization:** Monetize the service we provide at comparatively lower price than others popular service platforms.
4. **Real-time Engagement:** Providing interaction features like polls, quizzes, Q&A, and gamification. In addition, deep learning techniques can be used to boost real-time video processing and user engagement. By implementing deep learning-based adaptive bitrate algorithms, the stream quality can dynamically adjust to changing network conditions, ensuring smoother streaming. These models can optimize encoding and reduce latency during live events, leading to upgrade overall streaming performance [11].
5. **Security & Scalability:** Protect user data, handle growing user databases.

VI. LIMITATIONS

- **Security:** lack of input validation and password make it little easier for attacker to access the database.As Zhou et al. [12] discuss, live video streaming services face significant security challenges, including DDoS attacks and data breaches. These security concerns highlight the need for robust security measures to conisder in systems like StreamSymphony.
- **Scalability:** Less scalable and limited to specific platforms only that provide live sessions with the help of rtmp url and stream key
- **Resource Management:** Minimal error handling makes it difficult to diagnose and resolve issues effectively
- **Error Handling:** The difficulty of diagnosing and resolving issues is compounded by the lack of thorough error handling.

VII. CONCLUSION

In conclusion, StreamSymphony represents a promising step in the field of live streaming technologies. By combining use of RTMP, WebRTC, node js, FFmpeg and other advanced features, and improved performance, it stands to empower content creators and potentially reshape the landscape of live video streaming. As the digital content creation space continues to grow, platforms like StreamSymphony will play a crucial role in access to professional-grade streaming capabilities and fostering innovation in online content delivery.

REFERENCES

[1] A. Pal, S. Thakurta, and S. Sengupta, "Live Video Streaming on the Internet: Current State of Services and Challenges," IEEE Communications Surveys & Tutorials, vol. 23, no. 2, pp. 881-906, 2021, doi: 10.1109/COMST.2021.3059986.

[2] S. S. Krishnan and R. K. Sitaraman, "Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-Experimental Designs," IEEE/ACM Transactions on Networking, vol. 21, no. 6, pp. 2001-2014, Dec. 2013, doi: 10.1109/TNET.2013.2281542.

[3] P. Dahlgren, "The democratization of streaming: How free services are reshaping content creation," Journal of Media and Communication Studies, vol. 14, no. 1, pp. 23-36, 2022. doi: 10.1016/j.jmcs.2022.01.005.

[4] Acar, Y., et al. (2017). Comparative Analysis of Cryptographic APIs: A Case Study on bcrypt, scrypt, and Argon2. IEEE Transactions on Dependable and Secure Computing.

[5] Z. Luo, Z. Wang, M. Hu, Y. Zhou and D. Wu, "LiveSR: Enabling Universal HD Live Video Streaming With Crowdsourced Online Learning," in IEEE Transactions on Multimedia, vol. 25, pp. 2788-2798, 2023, doi: 10.1109/TMM.2022.3151259.

[6] Santos-González, I., Molina-Gil, J., Rivero-García, A., Zamora, A., Álvarez, R. (2018). A Comparative Study for Real-Time Streaming Protocols Implementations. In: Moreno-Díaz, R., Pichler, F., Quesada-Arencibia, A. (eds) Computer Aided Systems Theory – EUROCAST 2017. EUROCAST 2017. Lecture Notes in Computer Science(), vol 10671. Springer, Cham. https://doi.org/10.1007/978-3-319-74718-7_13.

[7] Sun, Liyang & Zong, Tongyu & Wang, Siquan & Liu, Yong & Wang, Yao. (2021). Towards Optimal Low-Latency Live Video Streaming. IEEE/ACM Transactions on Networking. PP. 10.1109/TNET.2021.3087625.

[8] J. Ye, S. Qin, Q. Xiao, W. Jiang, X. Tang and X. Li, "Adaptive Bitrate Algorithms via Deep Reinforcement Learning With Digital Twins Assisted Trajectory," in IEEE Transactions on Network Science and Engineering, doi: 10.1109/TNSE.2024.3376451.

[9] She, Bo & Wang, Qiang & Zhong, Xiaoge & Zhang, Zhe & Qin, Zunying & Li, Guodong. (2020). The Design and Implementation of Campus Network Streaming Media Live Video On-Demand System Based on Nginx and FFmpeg. Journal of Physics: Conference Series. 1631. 012158. 10.1088/1742-6596/1631/1/012158.

[10] Santos-González, I., Rivero-García, A., González-Barroso, T., Molina- Gil, J., Caballero-Gil, P. (2016). Real-Time Streaming: A Comparative Study Between RTSP and WebRTC. In: García, C., Caballero-Gil, P., Burmester, M., Quesada-Arencibia, A. (eds) Ubiquitous Computing and Ambient Intelligence. IWAAL AmIHEALTH UCAmI 2016 2016 2016. Lecture Notes in Computer Science(), vol 10070. Springer, Cham. https://doi.org/10.1007/978-3-319-48799-1_36.

[11] M. Takahashi, H. Zhang, and K. Murakami, "An efficient framework for real-time live streaming system using FFmpeg," ACM Trans. Multimedia Comput., Commun. Appl., 2020.

[12] A. C. Begen, T. Akgul, and M. Baugher, "Watching video over the web: Part 1: Streaming protocols," IEEE Internet Comput., vol. 15, no. 2, pp. 54-63, 2011.

[13] W. Wang and Y. Wang, "Deep learning for real-time video processing: A survey," J. Vis. Commun. Image Represent., vol. 71, 2020.

[14] Zhou, Q., et al. (2021). Security and Privacy Challenges in Live Video Streaming Services. IEEE Access.