JOISIE (Journal Of Information Systems And Informatics Engineering)
Vol. 9, No.1, Juni 2025, Hlm 246-259
Received: 15 Junuari 2025 Revised: 5 Februari 2025, Accepted: 29 Juni 2025

p- ISSN: 2503-5304
e- ISSN: 2527-3116

IMPLEMENTATION OF FLUTTER AND FIREBASE TECHNOLOGIES IN MODERN LIVE STREAMING APPLICATION DEVELOPMENT

Fauzi Hidayatul Anmi[1)], Muhammad Jazman[2),] Anofrizen[3)], Eki Saputra[4)], Syafril Siregar[5)], T. Khairil Ahsyar[6)].
[1,2,3,4,5,6]Fakultas Sains dan Teknologi, Universitas Islam Negeri Syarif Kasim Riau, Jl. H.R. Soebrantas Km. 15 Panam, Pekanbaru, Riau
email: 11850311422@students.uin-suska.ac.id, jazman@uin-suska.ac.id, anofrizen@uin-suska.ac.id, eki.saputra@uin-suska.ac.id, syafrilsrg@gmail.com, tengkukhairil@uin-suska.ac.id

Abstract
Live streaming applications play a vital role in entertainment, education, and business communication. Despite their growing adoption, developing efficient and scalable live streaming applications remains challenging, particularly in ensuring real-time data synchronization, cross-platform compatibility, and optimal user experience under varying network conditions. This study addresses these challenges by designing and implementing a live streaming application using Flutter and Firebase, leveraging their advantages in cross-platform development and backend services. The research employs an Agile-Scrum methodology to iteratively develop and refine the application, incorporating essential features such as real-time video streaming, user authentication, live chat, and voice changers. Performance evaluation is conducted through direct user testing to assess functionality, responsiveness, and network adaptability. The results demonstrate that Flutter effectively streamlines the development of cross-platform interfaces, while Firebase ensures seamless real-time data synchronization and authentication. However, challenges persist in optimizing the application for low-spec devices and ensuring high-quality video streaming under limited bandwidth conditions. This study contributes to the field by providing a structured development approach for real-time live streaming applications, offering insights into best practices for integrating Flutter and Firebase. Future enhancements may explore automatic video quality adjustment and artificial intelligence-driven user interaction improvements to further refine the user experience.
Keywords: *Flutter, Firebase, Agile-Scrum, Live Streaming.*

1. INTRODUCTION

Live streaming applications have become essential in various sectors, including entertainment, education, business communication, and social activities. Their popularity continues to rise with increasing internet penetration and the widespread adoption of smart devices. According to a Grand View Research report (2023), the global live streaming application market is projected to reach USD 247.3 billion by 2027, with an annual growth rate of 28.3% from 2020 to 2027. However, developing live streaming applications poses challenges, including ensuring performance efficiency, scalability, and integrating complex technologies.

Flutter, a cross-platform framework, has gained prominence due to its development efficiency and ability to create consistent user experiences across various platforms. Features like "hot-reload" and a flexible user interface make Flutter a preferred choice for modern application developers (Zou & Darus, 2024). Additionally, Firebase, a backend-as-a-service platform developed by Google, provides comprehensive solutions for real-time applications, including user authentication, data storage, and synchronization. Firebase's scalability supports the efficient management of extensive streaming data (K. U. Singh et al., 2024).

Combining Flutter and Firebase presents a cost effective, reliable solution for building complex applications (C. Sharma et al., 2024).

The Agile-Scrum methodology complements real-time application development by offering flexibility, rapid iteration, and enhanced team collaboration. Research demonstrates that Scrum allows developers to dynamically adjust priorities based on user feedback and technical challenges (Crum, Li, & Kou, 2024). This structured approach has proven beneficial for distributed software project management (Al-Ahmari et al., 2022).

Examples include leveraging Scrum for effective team coordination in Flutter and Firebase-based projects and using Flutter for single code based real-time applications (Chancusig-Chisag, Martinez-Freire,

https://doi.org/10.35145/joisie.v9i1.4808

Cantuña-Flores, & Banda Casa, 2023). Firebase's real-time synchronization capabilities further enhance the integration process in Flutter applications (Martins et al., 2024).

Despite the growing adoption of Flutter and Firebase in live streaming application development, comprehensive documentation on their integration remains limited. This study aims to document the design and development processes of a live streaming application using Flutter and Firebase, providing a valuable reference for future developers.

2. METHODS

The study employed an Agile-based software development approach, specifically the Scrum framework. Agile was chosen for its high flexibility, rapid iteration capability, and focus on user needs. In the context of live streaming application development, this method allows the team to dynamically adjust priorities based on user feedback and technical challenges encountered during the development process (Alsharari, Zainon, Letchmunan, Mohammed, & Alsharari, 2023). Previous studies have shown that the Scrum approach enhances the efficiency of software project management through structured task allocation and effective collaboration (Chang & Wongwatkit, 2024).

The software development process in this study followed the Scrum stages as described below :

1. Sprint Planning

At this stage, the main features of the live streaming application were identified and planned. These features included real-time streaming, user authentication, and user data management. Each feature was broken down into specific tasks to be completed during the sprint (Yan, 2024).

2. Sprint Execution

Sprint Execution, was conducted in two-week sprint cycles. Tasks were allocated as follows: frontend development was carried out using Flutter, selected for its cross-platform compatibility and efficient UI rendering. Backend development was managed via Firebase, which was chosen for its real-time database capabilities, scalability, and authentication services. Additionally, the integration of SDKs, particularly the ZEGOCLOUD SDK, ensured seamless live streaming functionalities, including low-latency video transmission and multi-user interactions. Flutter was chosen for its efficient cross-platform capabilities, while Firebase was selected for its scalability in handling real- time request and robust support for cloud-based application development (K. U. Singh et al., 2024).

3. Sprint Review and Retrospective

a Sprint Review and Retrospective session was conducted. In this phase, the developed prototypes were tested by users to gather direct feedback. The team analyzed performance metrics and identified areas for improvement. Based on these findings, necessary adjustments were incorporated into subsequent sprint cycles to refine application functionality and enhance the user experience (Visescu, Larusdottir, & Islind, 2023).

The testing phase utilized black-box testing, focusing on validating key functionalities based on specific input and output scenarios. The primary test cases included user registration, login authentication, live streaming initialization, and multi-user interactions. Each of these test cases was executed, and all were successfully completed as expected.

Each sprint produced clear deliverables, ranging from initial protorypes to user-testable applications. The final outcome of this process was a stable version of the live streaming application that met user needs and delivered optimal performance. This iterative process ensured that the application was developed with high quality and responsiveness to market demands. The Scrum approach proved relevant in the context of modern software development, especially in applying technologies like Flutter and Firebase for real- time applications (Zmudczynska & Chen, 2024).

3. RESULT AND DISCUSSION

The implementation of live streaming technology using Flutter, Firebase, and Zego SDK follows a structured and systematic workflow to ensure security, efficiency, and real-time user engagement (Bartlett, Kabir, & Han, 2023). Refer to the image below :

https://doi.org/10.35145/joisie.v9i1.4808

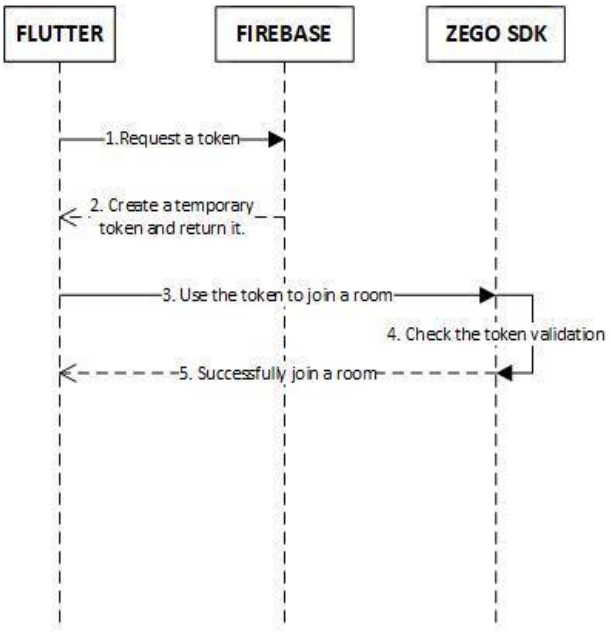

Figure 1. Live streaming workflow

The process begins with the Flutter frontend, which functions as the primary user interface for initiating requests to access live streaming sessions (Achilleos et al., 2023). Upon user interaction, the Flutter application communicates with the Firebase backend to request the generation of a temporary authentication token. Firebase processes this request by creating a secure and time-limited token, which is encrypted and uniquely tied to the specific session or user. This token is then returned to the Flutter application to facilitate subsequent steps in the process (Sarker, Jesser, & Speidel, 2023).

After the token is received, the Flutter application employs the token to request entry into the live streaming room through the ZEGOCLOUD SDK. At this stage, the ZEGOCLOUD SDK plays a critical role in validating the token by verifying its authenticity and ensuring that it has not expired (Dua, Reddy, Vishnoi, Tomar, & Dua, n.d.). This validation process safeguards the live streaming session by restricting access to authorized users, thereby maintaining the session's security and integrity (S. Sharma et al., 2024).

Once the token validation is successfully completed, the ZEGOCLOUD SDK establishes a connection to the live streaming room. The Flutter application then transitions to the live streaming interface, enabling users to participate in real-time video broadcasting or viewing (Kim, Ko, Choi, & Lee, 2023). Additionally, interactive features such as live chat further enhance user engagement during the session.

The integration of Flutter, Firebase, and ZEGOCLOUD SDK leverages the unique strengths of each technology. Flutter, as a cross-platform development framework, ensures compatibility across multiple devices (Android and iOS) while maintaining a responsive and intuitive user interface. Firebase provides robust backend functionalities, including secure authentication, token management, and seamless real-time data synchronization, which are essential for managing user interactions and ensuring application reliability (D. Singh, Agarwal, Marwaha, Mishra, & Pathak, 2024). Meanwhile, the ZEGOCLOUD SDK incorporates advanced live streaming capabilities, delivering minimal latency and dependable video transmission for an optimal user experience.

After explaining the basic workflow of live streaming technology using Flutter, Firebase, and ZEGOCLOUD SDK, it is essential to understand how this technology facilitates real-time interaction among multiple users within a single virtual room. Figure 2 illustrates the multi-user communication scheme, demonstrating how two or more users can simultaneously participate in a live streaming session with the assistance of ZEGOCLOUD SDK. The following section provides a detailed explanation of the processes involved in this scheme.

https://doi.org/10.35145/joisie.v9i1.4808

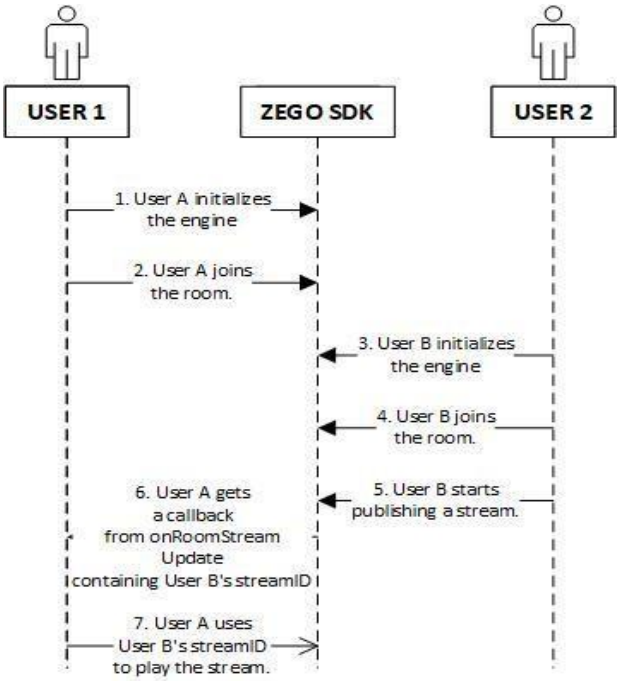

Figure 2. Multi-user workflow

In a live streaming system based on ZEGOCLOUD SDK, real-time communication between two or more users is carried out through a structured workflow. The presented diagram illustrates how two users, namely User A and User B, interact within a live streaming room using the streaming mechanism provided by the ZEGOCLOUD SDK. Below is a scientific explanation of the process:

1. Initialization of the engine by User A

User A begins the process by initializing the ZEGOCLOUD SDK engine. This step prepares the application to communicate with the ZEGOCLOUD server. The initialization includes initial configuration and connection setup, enabling the system to send or receive streaming data.

2. User A joins the live streaming room

After completing the initialization, User A sends a request to join a specific live streaming room. This request is transmitted through the ZEGOCLOUD SDK to the server, which validates the request and allows User A to enter the room.

3. Initialization of the engine by User B

In parallel, User B begins the process by performing a similar step, initializing the ZEGOCLOUD SDK engine. This step ensures that User B establishes the same connection with the server and gains access to the streaming features provided.

4. User B joins the same room

After successfully initializing the engine, User B sends a request to join the same room as User A. The ZEGOCLOUD server processes this request, allowing User B to join the room and connect with other users.

5. User B start live streaming

After joining the room, User B begins streaming by publishing real-time audio or video data. This process is managed through the ZEGOCLOUD SDK, which handles the transmission of streaming data from User B's device to the server for distribution to other users in the room.

6. Callback onRoomStreamUpdate to User A

Once User B starts streaming, the ZEGOCLOUD SDK sends a callback named onRoomStreamUpdate to User A's device. This callback contains information about the **streamID** of the live stream published by User B. The **streamID** is used to uniquely identify the live stream within the room.

7. User A plays the stream from User B

Using the streamID information received, User A can play the live stream published by User B. This process enables two-way or multi- way real-time communication, whether in the form of video, audio, or a combination of both.

https://doi.org/10.35145/joisie.v9i1.4808

This mechanism enables efficient and structured interactive communication within a live streaming room. Through engine initialization, room management, streamID callbacks, and the ability to play live streams, ZEGOCLOUD SDK creates an environment that seamlessly supports real-time interaction between users. This implementation is highly

relevant for applications requiring multi-user collaboration and communication, such as webinars, virtual conferences, or entertainment platforms based on live streaming. The standardized workflow ensures that streaming data is handled with minimal latency and high quality, delivering an optimal user experience.

a. SYSTEM REQUIREMENTS ANALYSIS

The system requirements analysis involves determining the hardware needed for software development. The hardware used in this research is detailed in the tables below :

Table 1. PC Hardware

No	Hardware	Specifications
1	Processor	AMD Ryzen 3
2	RAM	4 GB RAM
3	Hard Drive	1 TB

For mobile hardware, specific requirements are essential to ensure the compatibility and performance of the application on mobile devices. The details of the mobile hardware used in this research are as follows:

Table 2. Mobile Hardware

No	Hardware	Specifications
1	Processor	2.0 GHz Octa-core
2	RAM	4 GB RAM
3	Internal Memory	64 TB
4	Android Version	11

The software used in this research is listed in the table below :

Table 3. Software Specifications

No	Software	Specifications
1	Operating System	Windows 10
2	Programming Language	Dart
3	Framework	Flutter 3.7
4	Database	Firebase

b. SPRINT PLANNING

The sprint planning phase is an essential step in agile development, designed to ensure that the team has a clear roadmap for completing the sprint objectives (Hoffmann, 2024). During this phase, the scrum team holds structured meetings or briefings at the start of each team member to complete their assigned

https://doi.org/10.35145/joisie.v9i1.4808

responsibilities. The primary goal of these seesions is to establish a realistic and achievable plan that aligns with the sprint's overall objectives while accounting for team capacity and resource availability. The table below provides an overview of the estimated time allocated for each critical feature planned for the sprint. These estimates were carefully determined based on the complexity and priority of each feature.

Table 4. Sprint Planning Result

Item	Priority	Estimasi (days)
Register Page	High	3
Login Page	High	3
Live Streaming Page	High	6

1. Register page

This feature involves creating a user registration interface that allows users to input their and securely create an account. The priority level for this feature is high as it serves as the entry point for new users. The development of this feature is estimated to take 3 days, considering the design, coding, dan initial testing phases.

2. Login page

This feature enables users to securely access their accounts by providing credentials. As a high-priority task, it is crucial for the overall functionality of the application. The estimated time for completion is also 3 days, which includes implementation and ensuring seamless integration with the backend for user authentication.

3. Live streaming page

This feature is central to the application's functionality, allowing users to access or host live streaming sessions. As the most complex feature in this sprint, it has been assigned the highest estimated time of 6 days. This estimate covers the design of the interface, coding, integration of live streaming capabilities, and rigorous testing to ensure a smootsh user experience.

The combined estimates ensure that each feature is given adequate time for development and testing within the sprint timeframe (Pham & Neumann, 2024). This planning process promotes efficiency and alignment within the team, reducing the risk of delays or bottlenecks during execution. By adhering to these timeframes, the team ensures that the sprint's deliverables are completed with high quality and meet the project's objectives.

c. SPRINT EXECUTION

The sprint execution phase is a critical component of agile development, during which the scrum team schedules tasks and allocates time estimates to efficiently complete the goals outlined in the sprint planning phases (Choetkiertikul et al., 2024). Each sprint is structured to span 6 days, with 6 working hours per day, ensuring a focused and manageable workload for the team. This phase emphasizes task execution, collaboration, and timely delivery of incremental features. The table below details the tasks and their respective time estimates for sprint 1, which focuses on login and registration functionalities:

Table 5. Sprint 1

Sprint 1	Task	Estimasi (hours)
Login/Register	Database Setup	6
	Interface	8
	Coding	14
	Testing	8
Total		**36**

https://doi.org/10.35145/joisie.v9i1.4808

Database setup : allocated 6 hours, this task involves designing and configuring the database to support user registration and login functionality. This includes schema creation and data storage implementation.
Interface : allotted 8 hours, this task focuses on designing and developing the user interface for the registration and login pages, ensuring a user-friendly experience. Coding : assigned 14 hours, this task involves implementing the logic and backend processes required for login dan registration, including authentication and data validation.
Testing : with 8 hours allocated, this task validates the functionality and reliability of the implemented features, ensuring they meet the defined requirements.

Upon completing sprint 1, which delivers the login and registration features, the team proceeds to sprint 2. Sprint 2 centers on developing and refining the live streaming page. The estimated tima allocation for sprint 2 is as follows :

Table 6. Sprint 2

Sprint 2	Task	Estimasi (hours)
Live Streaming	Database Setup	6
	Interface	8
	Coding	14
	Testing	8
Total		**36**

Databases setup : this task involves configuring the database to manage data for live streaming sessions, such as stream scheduling and participation.
Interface : this task focuses on designing and implementing the user interface for the live streaming page, ensuring it is visually appealing and intuitive.
Coding : the most time-intensive task, this involves writing the code to enable live streaming functionality, and ensuring stable performance.
Testing : this task ensures the live streaming page operates as intended, addressing potential issues such as buffering, and UI responsiveness

Once the tasks in sprint execution are comleted, the process transitions to a review phase where all the features prepared during the sprint are evaluated (Becker, 2024). This review involves testing the functionality, identifying any issues or bugs, and ensuring that the features meet the initial design and user requirements. Based on the findings during the review, further refinements or adjustments may be made before progressing to the next development sprint. This iterative process ensures continuous improvement and adherence to project goals.

d. SPRINT REVIEW

During the sprint review, the results of each completed sprint are demonstrated and evaluated to ensure the features meet the requirements and function as intended (Myrvang & van den Tillaar, 2024). Below are the outcomes of the review process, highlighting the key application pages and their functionalities :

1. Register Page

The register page demonstrates the process for users to register for users to register and create an account within the application. This page ensures that users can input the required details accurately and securely.

https://doi.org/10.35145/joisie.v9i1.4808

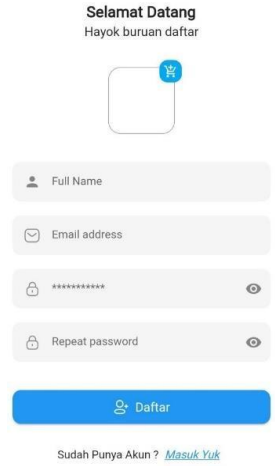

Figure 3. Register page

2. Login Page

The login page illustrates the steps for users to login to the application securely. This page includes fields for user credentials and features secure authentication mechanisms.

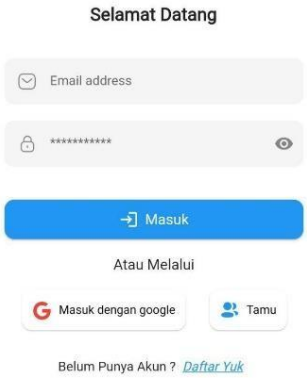

Figure 4. Login page

3. Main Page

The main page showcases the core features of the application. It acts as the central hub, providing users access to various functionalities such as navigationg to live streaming, checking schedules, or managing their profiles.

Figure 5. Main page

https://doi.org/10.35145/joisie.v9i1.4808

4. Live Streaming Schedule Page

The live streaming schedule page displays posts containing all the schedules for upcoming live streaming sessions. This page ensures users can view and manage upcoming events with ease.

Figure 6. Schedule page

5. Live Streaming Schedule Details

The schedule details page appears after a user clicks on a specific post or schedule. This page provides detailed information about the selected live streaming session, including time, topic, dan host details.

Figure 7. Schedule details page

6. Join Live Streaming Page

The join live streaming page is displayed after users click the “Join” button. This page enables the host to broadcast live sessions while allowing audiences to watch the stream, participate in live chat, and use the voice changer feature for an enhanced interactive experience.

https://doi.org/10.35145/joisie.v9i1.4808

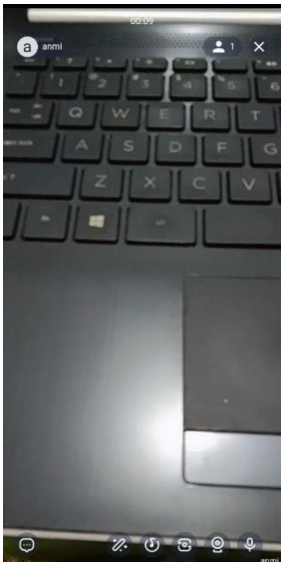

Figure 8. Live streaming page

The sprint review process is crucial for validating the progress of each sprint. It provides an opportunity to test and refine features while gathering feedback to address any issues before moving to the next development phase. Each demonstrated page ensures that the application aligns with the project goals and delivers an optimal user experience.

e. SOFTWARE TESTING

The software testing conducted in this study utilized the black-box testing method. This method focuses on examining the functionality of the software without considering its internal structure or coding architecture. The purpose of this testing approach is to ensure that the software meets its functional requirements as intended. The result of the black-box testing are detailed in the table below:

Table 7. Software Testing Result

Test Case	Expected Result	Result
Register	Data is saved to the firebase, and the page redirects to the login page	Success
Login	Login succeeds and the user is directed to the main page	Success
Live streaming schedule	Displays the live streaming schedule page	Success
Details of a live streaming schedule page	Displays the details of the selected live streaming schedule page	Success
Join live streaming room	Display the live streaming room	Success

https://doi.org/10.35145/joisie.v9i1.4808

The black-box testing methodology was employed to validate key functionalities of the software, focusing on user interactions and outputs based on specific input scenarios (Hassija et al., 2024).

The black-box testing results demonstrated that all testet features of the applications met the expected outcomes. This validates the software's readiness for deployment, ensuring that the critical functionalities operate effectively to deliver an optimal user experience (Denden, Tlili, Salha, & Abed, 2024).

In addition to validating the core functionalities of the application, the study also evaluated the live streaming performance across various network types and device specifications. The results of the performance testing are summarized in the table below:

Table 8. Live streaming performance testing

Test Skenario	Device Specification	Streaming Latency (ms)	Expected Latency (ms)	Result
Live streaming on Wi-Fi (High-end device)	2.0 GHz Octa- core, 6 GB RAM	150	≤ 200	Success
Live streaming on 4G network (High-end device)	2.0 GHz Octa- core, 6 GB RAM	180	≤ 200	Success
Live streaming on Wi-Fi (Low-end device)	1.8 GHz Quad- core, 2 GB	250	≤ 300	Success
Live streaming on 4G network (Low-end device)	1.8 GHz Quad- core, 2 GB RAM	280	≤ 300	Success
Live streaming on 3G network (High-end device)	2.0 GHz Octa- core, 6 GB RAM	500	≤ 600	Success
Live streaming on 3G network (Low-end device)	1.8 GHz Quad- core, 2 GB	600	≤ 700	Success

Test Scenario, The test scenario explains the specific conditions being tested, such as the use of the application with particular network types (Wi-Fi, 4G, or 3G) and device configurations. **Purpose:** To ensure the application provides optimal live streaming performance under various network and device conditions. **Relevance:** These scenarios simulate real-world situations experienced by users in different environments, ensuring the application performs well universally. The network type refers to the type of internet connection used during testing, which influences the speed and stability of live streaming. **Wi-Fi:** The fastest and most stable connection, ideal for home or office use. **4G:** A fast mobile network with low latency, though performance may vary depending on the signal strength of the carrier. **3G:** A slower connection used to simulate low-bandwidth network conditions. **Analysis:** The application's performance on Wi-Fi is expected to be optimal, while testing on 3G evaluates its tolerance to less ideal network conditions.

Device specifications, include the hardware configurations used during testing, which can affect the device's ability to handle live streaming processes. **High-end Device:**
2.0 GHz Octa-core processor with 6 GB of RAM. This configuration provides strong performance and is ideal for testing on premium devices. **Low-end Device:** 1.8 GHz Quad-core processor with 2 GB of RAM.

https://doi.org/10.35145/joisie.v9i1.4808

This configuration simulates entry-level devices with limited capabilities, often used by users with basic devices. **Analysis:** Comparing performance on high-end and low-end devices allows evaluation of the application's compatibility across device classes.

Streaming latency is the measured time (in milliseconds) between a user's action and the corresponding data response displayed in a live streaming session. **Low:** Streaming latency under 200 ms ensures smooth and real- time interactions. **Moderate:** Latency between 200–600 ms is acceptable for most streaming applications but may exhibit slight delays. **High:** Latency above 600 ms can disrupt the user experience, particularly in interactive features like live chat. **Analysis:** Streaming latency data is used to assess the application's performance under varying network and device conditions.

Expected Latency, The acceptable latency threshold is defined based on specific performance criteria to ensure an adequate streaming experience for users. **≤ 200 ms:** Ideal for stable networks like Wi-Fi and high- end devices. **≤ 300 ms:** Still acceptable for low-end devices or 4G networks. **≤ 600 ms:** Suitable for slower networks like 3G. **Analysis:** By establishing these thresholds, the testing process identifies areas that may require optimization.

The Results column indicates whether the live streaming performance in each scenario meets expectations. **Success:** The application meets the expected latency thresholds for all tested scenarios. **Analysis:** All tests in the table showed "Success," indicating the application is well-designed to handle a variety of conditions.

Conclusion This table demonstrates that the live streaming application:

1. Provides optimal performance on Wi- Fi with high-end devices, achieving latency as low as 150 ms.
2. Remains reliable on low-end devices and 3G networks, with slightly higher latency that is still within acceptable limits.
3. Utilizes Firebase as the underlying technology to maintain consistent application performance across various conditions.

If further optimization is needed, efforts can focus on reducing latency for 3G networks or improving efficiency on low-end devices. This table confirms that the application is suitable for use by a diverse range of users in various environments and on different devices.

4. CONCLUSION

This study successfully developed an Android-based live streaming application using Flutter and Firebase through the Agile- Scrum approach. The application effectively implemented core features such as real-time streaming, user authentication, live chat and voice changer functionalities. Flutter enabled efficient cross-platform development, while Firebase supported reliable real-time data synchronization.

The study is relevant for the development of similar applications in the future, with potential adaptations in various sectors such as education and entertainment. However, there are limitations, including compatibility with low-spec devices and the adjustment of features for diverse network conditions, which present opportunities for further research.

Recommendations for future development include optimizing the application for low-spec devices, integrating automatic video quality adaptation features, and applying artificial intelligence to enhance user experience. This study is expected to serve as a reference for developers looking to implement similar technologies.

5. REFERENCES

Achilleos, A., Mettouris, C., Yeratziotis, A., Starosta-Sztuczka, J., Moza, S., Hadjicosta, A., … Pecyna, K. (2023). Lessons learned from older adults fusing of an augmented reality, assisted living and social interaction platform. *SN Computer Science*, *4*(4), 378.

Al-Ahmari, M. M., Al Moaleem, M. M., Khudhayr, R. A., Sulaily, A. A., Alhazmi, B. A. M., AlAlili, M. I. S., … Shagagi, A. M. (2022). A systematic review of publications using the Dundee Ready Education Environment Measure (DREEM) to Monitor Education in medical colleges in Saudi Arabia. *Medical Science Monitor: International Medical Journal of Experimental and Clinical Research*, *28*, e938987-1.

Alsharari, A. S., Zainon, W. M. N. W., Letchmunan, S., Mohammed, B. A., & Alsharari, M. S. (2023). A Review of Agile Methods for Requirement Change Management in Web Engineering. *2023 International Conference on Smart Computing and Application (ICSCA)*, 1–9. IEEE.

Bartlett, L., Kabir, M. A., & Han, J. (2023). A review on business process management system design: the

role of virtualization and work design. *IEEE Access*.

Becker, T. (2024). Successfully Implementing Process Improvements. In *Optimising and Digitising Supply Chain Processes* (pp. 291–325). Springer.

Chancusig-Chisag, J.-C., Martinez-Freire, M.-N., Cantuña-Flores, K.-S., & Banda Casa, M. A. (2023). Development of a Web and Mobile Application for the Management and Sales of Flower Inputs for Clients of the Company "Flor Insumos SAS" in the City of Cayambe, Through the Use of Agile Practices. *International Conference on Computer Science, Electronics and Industrial Engineering (CSEI)*, 830–849. Springer.

Chang, S.-C., & Wongwatkit, C. (2024). Effects of a peer assessment-based scrum project learning system on computer programming's learning motivation, collaboration, communication, critical thinking, and cognitive load. *Education and Information Technologies*, *29*(6), 7105–7128.

Choetkiertikul, M., Banyongrakkul, P., Ragkhitwetsagul, C., Tuarob, S., Dam, H. K., & Sunetnanta, T. (2024). Sprint2Vec: a deep characterization of sprints in iterative software development. *IEEE Transactions on Software Engineering*.

Crum, S., Li, B., & Kou, X. (2024). Generative Artificial Intelligence and Interactive Learning Platforms: Second Language Vocabulary Acquisition. *International Conference on Human-Computer Interaction*, 48–53. Springer.

Denden, M., Tlili, A., Salha, S., & Abed, M. (2024). Opening up the gamification black box: effects of students' personality traits and perception of game elements on their engaged behaviors in a gamified course. *Technology, Knowledge and Learning*, *29*(2), 921–940.

Dua, M., Reddy, G. S., Vishnoi, R., Tomar, S., & Dua, S. (n.d.). A Secure, Scalable, and Integrated Smart Platform for Teaching and Coding. In *Internet of Things and Big Data Analytics-Based Manufacturing* (pp. 258–272). CRC Press.

Hassija, V., Chamola, V., Mahapatra, A., Singal, A., Goel, D., Huang, K., … Hussain, A. (2024). Interpreting black-box models: a review on explainable artificial intelligence. *Cognitive Computation*, *16*(1), 45–74.

Hoffmann, S. (2024). Introduction to Digital Product Management: Classification and Basic Concepts. In *Digital Product Management: Frameworks–Tools–Cases* (pp. 1–30). Springer.

Kim, M., Ko, H., Choi, J., & Lee, J. (2023). FlumeRide: Interactive Space Where Artists and Fans Meet-and-Greet Using Video Calls. *IEEE Access*, *11*, 31594–31606.

Martins, P., Silva, D., Pinto, J., Varanda, J., Váz, P., Silva, J., & Abbasi, M. (2024). Smart City Air Quality Monitoring: A Mobile Application for Intelligent Cities. *International Conference on Disruptive Technologies, Tech Ethics and Artificial Intelligence*, 123–132. Springer.

Myrvang, S., & van den Tillaar, R. (2024). The Longitudinal Effects of Resisted and Assisted Sprint Training on Sprint Kinematics, Acceleration, and Maximum Velocity: A Systematic Review and Meta-analysis. *Sports Medicine-Open*, *10*(1), 110.

Pham, K. P., & Neumann, M. (2024). How to Measure Performance in Agile Software Development? A Mixed-Method Study. *2024 50th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 443–450. IEEE.

Sarker, A., Jesser, A., & Speidel, M. (2023). Advancing Decentralized IoT with Privacy-preserving AI: Harnessing Federated Learning and NLP Techniques. *2023 IEEE International Conference on Artificial Intelligence, Blockchain, and Internet of Things (AIBThings)*, 1–5. IEEE.

Sharma, C., Singh, A., Kaur, G., Dadhwal, M., Chauhan, V., Singh, I., & Singh, M. (2024). Spectroscopy Sensor and Mobile App and Artificial Intelligence Integration for Real-Time Soil Analysis and Crop Management in Precision Agriculture. *2024 First International Conference on Electronics, Communication and Signal Processing (ICECSP)*, 1–11. IEEE.

Sharma, S., Singh, J., Gupta, A., Ali, F., Khan, F., & Kwak, D. (2024). User Safety and Security in the Metaverse: A Critical Review. *IEEE Open Journal of the Communications Society*.

Singh, D., Agarwal, K., Marwaha, M., Mishra, N., & Pathak, D. M. (2024). Intelligent Reporting System: Engineered with Application Development. *2024 11th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*, 1–5. IEEE.

Singh, K. U., Varshney, N., Gupta, P., Kumar, G., Singh, T., & Dogiwal, S. R. (2024). Mobile Application Control with Firebase Cloud Messaging. *International Conference On Innovative Computing And Communication*, 527–535. Springer.

Visescu, I., Larusdottir, M., & Islind, A. S. (2023). Supporting Active Learning in STEM Higher Education

Through the User-Centred Design Sprint. *2023 IEEE Frontiers in Education Conference (FIE)*, 1–10. IEEE.

Yan, J. (2024). Effectiveness evaluation of sprint sports techniques and tactics based on deep learning. *Service Oriented Computing and Applications*, 1–14.

Zmudczynska, E. W., & Chen, H.-C. (2024). Using Scrum to Build Tourism Information Mobile Application. *International Conference on Advanced Information Networking and Applications*, 426–437. Springer.

Zou, D., & Darus, M. Y. (2024). A Comparative Analysis of Cross-Platform Mobile Development Frameworks. *2024 IEEE 6th Symposium on Computers & Informatics (ISCI)*, 84–90. IEEE.

https://doi.org/10.35145/joisie.v9i1.4808