Journal of Information System Research (JOSH)
Volume 5, No. 2, Januari 2024, pp 583−591
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v5i2.4698

Performance Analysis of BLoC and GetX State Management Library on Flutter

Mohamad Zulistiyan*, Monterico Adrian, Yanuar Firdaus Arie Wibowo

School of Computing, Informatics, Telkom University, Bandung
Jl. Telekomunikasi. 1, Terusan Buahbatu - Bojongsoang, Telkom University, Sukapura, Kec. Dayeuhkolot, Kabupaten Bandung, Jawa Barat, Indonesia
Email: [1,*]mzulistiyan@student.telkomuniversity.ac.id, [2]monterico@telkomuniversity.ac.id, [3]yanuar@telkomuniversity.ac.id
Correspondence Author Email: mzulistiyan@student.telkomuniversity.ac.id
Submitted: **20/12/2023;** Accepted: **25/01/2024;** Published: **27/01/2024**

Abstrak−This research delves into evaluating Flutter's BLoC and GetX state management libraries, focusing on their memory and CPU usage across diverse dataset sizes of 1,000, 5,000, and 10,000 entries. The objective is to identify which library offers better performance efficiency in Flutter application development. Addressing the critical problem of resource optimization, the study uses a comparative analytical approach. Preliminary results indicate GetX's higher memory efficiency with smaller datasets and BLoC's superior CPU efficiency in handling larger datasets. These findings are pivotal for developers in choosing the right library, aligning with the specific performance demands and scalability needs of their applications, thereby ensuring enhanced functionality and user experience. This contributes significantly to the understanding of state management's impact on Flutter app performance, offering a detailed guide for optimal library selection in the ecosystem.

Keywords: Flutter State Management; BLoC Library; GetX Library; Performance Analysis; Application Scalability

1. INTRODUCTION

The popularity of mobile apps is due to the mobility enabled by these devices, offering services regardless of the user's location[1]. Differences in development too ls and languages between the major operating systems, Android and iOS, and the need to develop, test, and debug separate apps for each platform led to increased maintenance time and costs[2]. Flutter is one of the popular frameworks in multi-platform mobile application development. It is an SDK for cross-platform application development. Flutter is an open-source SDK that allows developers to create Android, iOS, and web applications using a single codebase. Flutter's development speed and flexibility advantages make it the right choice for rapid application development[3]. Fast app development is a crucial aspect of mobile app development. Flutter uses its engine called Skia and does not use bridges like in react native, so it is claimed to have better performance and efficiency[4].

With the increased computing power, smartphones have become an indispensable part of everyday life. Therefore, the number of smartphone users has increased exponentially in the past five years.[5], [6]. With the increased computing power, smartphones have become an indispensable part of everyday life. Therefore, the number of smartphone users has increased exponentially in the past five years. [7]. Survey by Apigee [8] shows that issues such as app freezing, crashes, slow response, inefficient battery usage, and excessive ads are the main reasons behind the negative reviews given by mobile app users. This survey found that 44% of users will immediately delete an app that does not perform to their expectations. Almost all respondents (98%) emphasized that performance is the top priority. The App Attention Index 2019 survey by AppDynamics revealed that performance issues occur across all types of digital services, such as social media, entertainment, productivity, retail, and finance, with 55% of users feeling frustrated by performance issues[9]. Apps with low performance often experience issues such as slow load times, crashes, or freezing. With technological advancements every day in the mobile industry, it is imperative to focus on the quality of mobile application software, specifically in terms of performance.[10]. Therefore, it is crucial to perform performance optimization in mobile application development.

BLoC (Business Logic Component) is a popular state management library on Flutter[11]. Meanwhile, GetX is a relatively new state management library in Flutter. It offers a more accessible and concise approach to state management than BLoC. However, not all state management libraries are suitable in all cases. Applications developed with Flutter are structured from user interface components called widgets, where the appearance of these widgets can change according to configuration and state. State management is crucial because only widgets that experience state changes must be rebuilt, thus minimizing resource usage[12].

Information system development has various methods. Methods that can be used for system development include structured and object-oriented methods[13]. This research analyzes the performance of two popular state management libraries in Flutter, BLoC, and GetX. This research aims to test and compare the performance of both libraries using appropriate testing methods. This includes measuring response time, memory usage, CPU consumption, and other metrics such as battery and network usage on mobile devices. This research aims to provide valuable insights for application developers in choosing the most suitable state management library for their developing applications. The results of this performance analysis will provide a deeper understanding of the advantages and disadvantages of each state management technique and provide recommendations for their optimal use in app development using Flutter.

Journal of Information System Research (JOSH)
Volume 5, No. 2, Januari 2024, pp 583−591
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v5i2.4698

Mobile application performance testing is an essential activity in the mobile application development cycle. [14]. Thorough performance testing of the mobile app should be done before the app is launched[1]. Cross-platform software development that provides testing and development for various operating systems while offering an alternative structure for the application development process[15], this research examines the impact of both libraries on memory consumption and CPU usage, as well as the effect on the operational stability of the application. Limitations of this research include exclusivity to Flutter as a development platform and not considering other state management libraries or applications developed outside the Flutter ecosystem. The main goal is to provide a reliable and valuable analysis for developers to choose between BLoC and GetX based on objective technical performance criteria. This research is expected to contribute to an increased understanding of state management optimization in cross-platform application development and provide practical guidance that can be applied in the application development process on Flutter.

2. RESEARCH METHODOLOGY

2.1 Research Related

Literature Study, where we conduct an in-depth study of the literature related to state management in Flutter, especially BLoC, and GetX, and the Android application performance testing methodology. Related research that can be used as a reference for this research is "Analysis of BLoC Performance and Provider State Management Library on Flutter," conducted by Regawa Rama Prayoga. [12]. The research aims to analyze the performance of two state management techniques in Flutter, namely BLoC, and Provider, in managing states in Flutter applications. Regawa Rama Prayoga conducted performance testing in the study on two Flutter applications that use BLoC and Provider as their state management techniques. The test results show that the performance of both techniques is quite good, but BLoC has slightly better performance than Provider in some test cases.

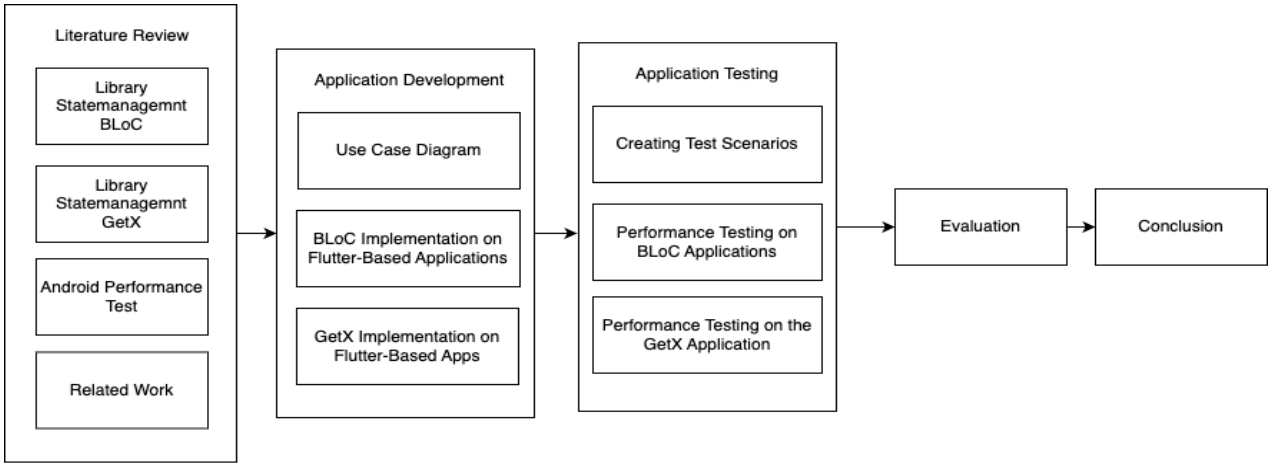

Figure 1. Research methodology performance analysis of BLoC and GetX State Management Library on Flutter

Building on these insights, our research methodology, as illustrated in Figure 1, encompasses a comprehensive analysis of state management solutions within the Flutter framework, with a specific focus on comparing the performance of BLoC and GetX. This methodological approach is designed to extend the existing body of knowledge by not only comparing BLoC with Provider, as done in previous studies, but also by including GetX in the comparison. Through this comparative analysis, we aim to provide a deeper understanding of the efficiency and effectiveness of these state management techniques in real-world application scenarios, thereby offering valuable insights for developers and researchers in the field of mobile application development.

2.1.1 Flutter

Flutter is an open-source framework for building mobile, web, and desktop applications. Google developed Flutter, and first released in 2017[16]. Flutter uses the Dart programming language, a modern language developed by Google and optimized for mobile and web application development.

Moreover, Dart's versatility extends beyond mobile. With its ability to compile to JavaScript, Dart is also used for web development. This makes it possible for developers to use a single programming language and a unified codebase to build applications that run seamlessly across multiple platforms, including iOS, Android, and the web. This cross-platform capability reduces development time and effort, as developers don't have to write and maintain separate codebases for each platform.

Dart's design also emphasizes developer productivity and ease of learning. Its syntax is clean and familiar to developers who have experience with other mainstream object-oriented languages like Java or C#. This lowers the learning curve for new developers and makes Dart an accessible choice for a wide range of programmers.

Journal of Information System Research (JOSH)
Volume 5, No. 2, Januari 2024, pp 583−591
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v5i2.4698

2.1.2 State Management in Flutter

State management is a technique to organize and store states in an application [17]. State management is critical in mobile application development, especially in complex applications that require much interaction between the user and the application. Some standard state management techniques used in Flutter include setState, Provider, BLoC, and GetX. [18].

2.1.3 BLoC (Business Logic Component)

Business Logic Component (BLoC) is a design pattern used in Flutter development to separate business logic from the user interface [19]. Its main goal is to increase the modularity, testability, and reusability of code. BLoC works on the concept of events and state:

a. Event: User actions or interactions that are translated as events are to be processed by BLoC.
b. State: A UI state or condition that changes based on an event.

In the latest version of BLoC, state and event management has been simplified, leaving behind the complex dependency on Stream and Sink. Now, BLoC facilitates event processing and new state delivery in a more integrated way with Flutter. This makes the separation between UI and business logic more intuitive, speeds up the development process, and simplifies testing and code maintenance[20]. This new approach helps build more scalable and manageable applications by reducing the complexity of state and event management.

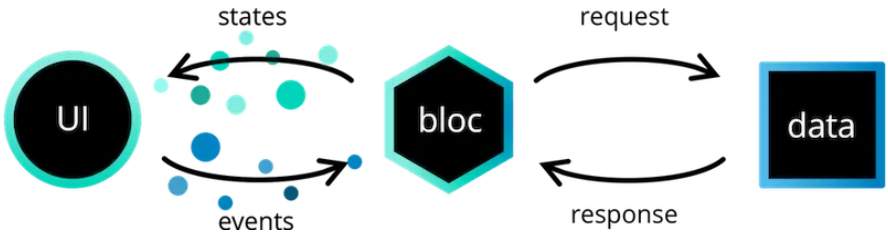

Figure 2. BLoC Flow Diagram, illustrating state and event handling in the Movie App

2.1.4 GetX State Management Library

GetX, popular with Flutter developers, combines efficiency in state management, navigation, and dependencies with three fundamental principles: performance, productivity, and organization[21]. The focus on high performance with minimal resource usage, intuitive syntax for high productivity, and good code organization is an evolution of the BLoC concept. With a reactive and declarative approach, GetX enables intuitive and responsive state management, reduces boilerplate code, and improves overall performance[22 These principles and ease of use make GetX a popular choice for Flutter app development.

2.2 Application Development

Application Development involves creating use cases and class diagrams and practical implementation using BLoC and GetX.

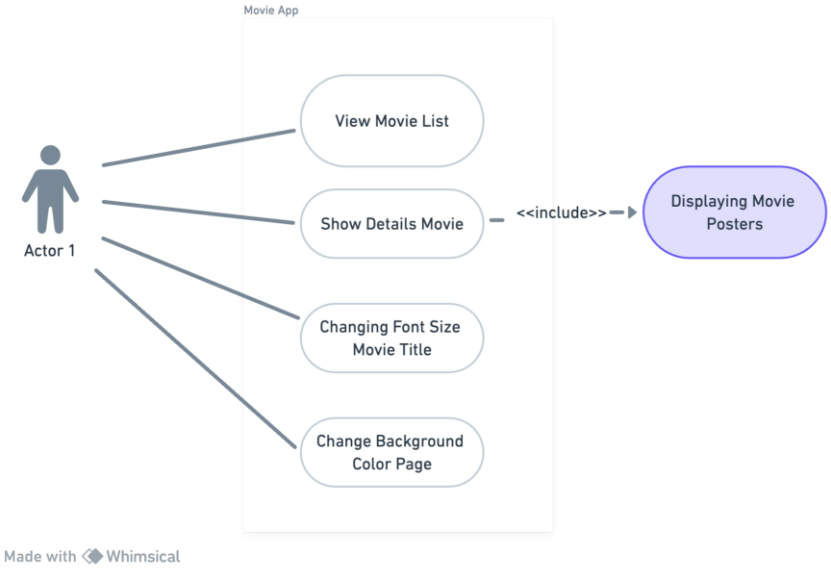

Figure 3. Use Case Diagram for the Movie App, outlining user interactions and testable features with Flutter's DevTools.

Journal of Information System Research (JOSH)
Volume 5, No. 2, Januari 2024, pp 583−591
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v5i2.4698

Evaluation during testing will be directed at the two applications that have been developed, focusing on performance metrics such as CPU and memory usage. As illustrated in Figure 3, performance testing will be performed using Flutter's DevTools, a suite of performance and debugging tools designed for Dart and Flutter applications. DevTools allows users to inspect UI layout and application state, diagnose UI performance issues, perform CPU profiling, and monitor network and memory usage[23]. This tool is expected to be used in conjunction with an existing IDE or command-line development workflow, facilitating testing of code that affects late frames and studying CPU usage in the background.

2.3 Appliaction Testing

The experimental phase is structured around rigorous testing of the application through a series of predefined scenarios, as outlined in Table 1. These scenarios are designed to probe the application's responsiveness, efficiency, and resource management under various conditions.

Table 1. Detailed Test Scenarios for Application Performance Evaluation

ID Scenario	Description
SK - 01	Process API data until all views appear
SK - 02	Displaying Movie Detail Information
SK - 03	Refresh Content and State Management
SK - 04	Scroll Movies on the Movie Page
SK - 05	Change the Background Colour on the Home Page
SK - 06	Change the Font Size of the Film Title on the Home Page

To ensure the accuracy of the performance evaluation, each test scenario is conducted twice across three different datasets comprising 1,000, 5,000, and 10,000 entries. Repeating each scenario thrice stabilizes the dataset for a more reliable analysis. The subsequent analysis of experimental results will focus on the averaged outcomes from these iterations, specifically examining CPU usage, memory consumption, and execution time within the Flutter application. Critical to this evaluation is the comparative performance of the GetX and BLoC state management libraries, which will be assessed in terms of their impact on the number of widgets generated and the overall resource utilization.

This meticulous approach to experimentation and analysis aims to yield a quantifiable comparison of the state management libraries' efficiency. The testbed for these experiments is an Android device configured to Android 10 specifications, featuring 6GB of RAM and an octa-core Qualcomm Snapdragon 680 processor, to reflect a real-world usage scenario.

2.4 Evaluation & Analysis

The analysis stage is carried out to analyze the experimental results and get conclusions about CPU and memory usage in the default Flutter application, applications built using the GetX state management library, and applications built using the BLoC state management library. This analysis calculates the performance efficiency value using a predetermined formula.

2.4.1 CPU Usage Testing

In this research, CPU utilization testing was conducted using the Android Debug Bridge (ADB), a command-line tool that enables communication between Android devices and developers' computers. This method captures CPU usage data directly from the device, providing accurate insight into the application's performance in a natural environment. This approach allows researchers to specifically identify processes and functions that consume the most CPU resources, opening up opportunities for more precise optimization and improving overall app performance. Data from these ADB tests will be the basis for evaluating and comparing CPU efficiency between applications using BLoC and GetX.

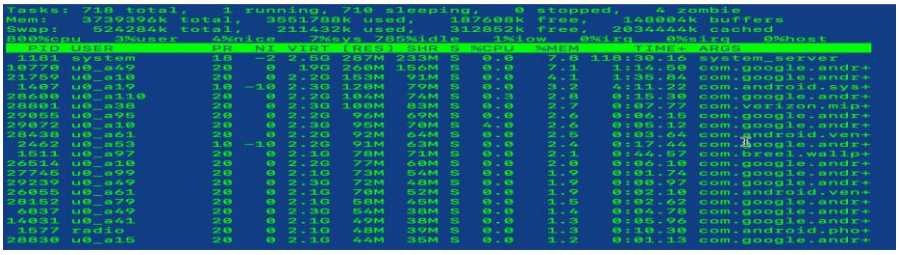

Figure 4. Sample ADB Output Showing CPU Usage

2.4.2 Memory Usage Testing

In this research, the analysis of application memory usage is done using the 'Profile Memory' feature in Flutter DevTools. This feature provides detailed information about the application's memory allocation by class and

Journal of Information System Research (JOSH)
Volume 5, No. 2, Januari 2024, pp 583−591
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v5i2.4698

memory type. Users can view the current memory allocation distribution and have the option to download the data in CSV format, which facilitates further analysis with tools such as Google Sheets. In addition, the 'Refresh on GC' feature enables real-time monitoring of memory allocation, making it easier to identify and diagnose issues related to application memory usage.

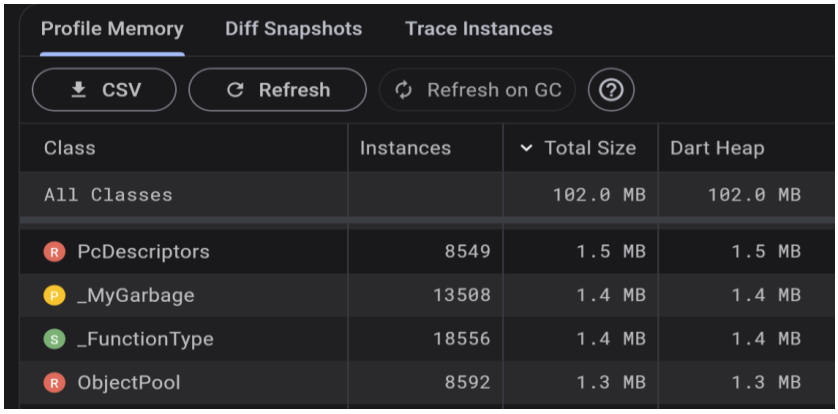

Figure 5. FMemory Allocation Snapshot in Flutter DevTools

3. RESULT AND DISCUSSION

The following are the performance measurement results of the two applications with 1000, 5000, and 10000 data sets, respectively.

Figure 6. Memory Usage Measurement Results with 1000, 5000 and 10000 Carried Data

a. Memory Usage Analysis for 1,000 Data Dataset
In scenario SP-02, as depicted in Figure 6, GetX's memory usage is notably higher, the memory usage by GetX is generally higher than that of BLoC, except in specific scenarios where GetX shows greater efficiency. Especially in scenario SP-02, it is seen that GetX requires more memory, which may indicate that the scenario particularly emphasizes the state management aspect, which GetX handles more memory intensively. However, in scenarios SP-03 and SP-06, GetX's memory usage is more efficient, or at least on par with BLoC, suggesting that in more controlled or predictable data usage situations, GetX can better optimize resource usage.

b. Memory Usage Analysis for 5,000 Data Dataset
Referring to Figure 6 for the 5,000 data dataset, we observe that GetX displayed, GetX showed decreased memory usage in some scenarios compared to BLoC, signaling its ability to manage memory more efficiently under heavier data loads. This demonstrates GetX's adaptability to more complex conditions, suggesting that

Journal of Information System Research (JOSH)
Volume 5, No. 2, Januari 2024, pp 583−591
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v5i2.4698

GetX can dynamically adjust its usage according to the scale and complexity of the data. This observation is essential for developers facing application scenarios with substantial data volumes, where efficient memory management is crucial.

c. Memory Usage Analysis for 10,000 Data Dataset
Figure 6 illustrates that for the 10,000 data dataset, the memory usage analysis shows, the memory usage analysis shows that the differences between GetX and BLoC become more apparent. GetX, in some cases, showed a substantial increase in memory usage, which may reflect the limits of its capacity in managing large memory allocations. On the other hand, BLoC appears to maintain more consistent memory usage even under increased load, confirming its robust architecture for large-scale applications. This suggests that BLoC could be a more stable choice for applications with data-extensive and memory-intensive operations.

Figure 7. CPU Usage Measurement Results with 1000, 5000 and 10000 Carried Data

a. CPU Usage Analysis for 1,000 Data Dataset
As depicted in Figure 7, the CPU Usage Analysis for a 1,000 data dataset shows that, it was seen that GetX generally had lower CPU usage than BLoC in several scenarios. This suggests that GetX is more efficient in handling less demanding tasks, optimizing CPU usage for lightweight operations. On the other hand, BLoC, despite showing slightly higher CPU usage, remained consistent and effective in situations requiring more stable data processing. This indicates that BLoC may be more suitable for applications that require more intensive and continuous processing. On the other hand, BLoC's slightly higher CPU usage may seem like a drawback. However, it is well suited for use in more robust and stable computing environments. Its consistent CPU usage makes it a reliable choice for applications that involve complex state management and need to handle continuous or intensive data processing tasks. This could include applications with real-time data streaming, complex user interactions or those requiring regular communication with a back-end server. BLoC's ability to maintain stable CPU utilisation under such conditions is essential to ensure that the application remains responsive and efficient even under heavy load.

b. CPU Usage Analysis for 5,000 Data Dataset
Referring to Figure 7, in the analysis of the 5,000 data dataset, it is observed that, the CPU utilization by GetX showed a more noticeable improvement over BLoC in most scenarios. This could indicate that GetX may have difficulty managing processing efficiency at larger data scales. BLoC, on the other hand, maintained a more stable CPU usage profile, demonstrating its ability to manage heavier and more complex tasks with more consistent efficiency, thus offering reliability in applications with more extensive data volumes. BLoC, on the other hand, shows a commendable consistency in CPU usage, even as data volumes increase. This stability is indicative of its robustness in handling complex and heavy tasks. BLoC's ability to handle such demanding scenarios without significant fluctuations in CPU usage makes it a reliable option for larger, more complex applications. Its structured approach to state management not only helps organise the codebase for scalability, but also ensures efficient resource utilisation under heavy loads. This consistency is particularly valuable in applications where performance and reliability are paramount, such as enterprise-level solutions, demanding content management systems, or applications requiring real-time data processing and visualisation. BLoC's

Journal of Information System Research (JOSH)
Volume 5, No. 2, Januari 2024, pp 583−591
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v5i2.4698

predictable performance under these conditions makes it a strong candidate for projects where maintaining high levels of efficiency and reliability as the application scales is critical.

c. CPU Usage Analysis for 10,000 Data Dataset
This consistency in BLoC and GetX performance is shown in Figure 7, which shows that when analysing the CPU usage for the 10,000 data set, GetX showed a more noticeable improvement in CPU usage than BLoC in most scenarios. This could indicate that GetX may have difficulty maintaining processing efficiency at larger data scales. BLoC, on the other hand, maintained a more stable CPU usage profile, demonstrating its ability to handle heavier and more complex tasks with more consistent efficiency, thus offering reliability in applications with larger data volumes. BLoC's performance showed a more stable CPU usage profile, even as data volumes increased. This stability is indicative of its robust architecture, designed to consistently handle complex and heavy computing tasks. BLoC's ability to maintain CPU usage efficiency under larger data loads makes it a reliable choice for more complex applications or those that need to handle large amounts of data. This could include applications in areas such as data analytics, machine learning or large enterprise systems where the volume of data and complexity of operations are significant. BLoC's consistent performance in these tests underscores its suitability for scenarios where stability and reliability of data processing are critical. For developers and organisations planning applications that not only start with significant data handling requirements, but also anticipate scaling in the future, BLoC's ability to efficiently handle heavy and complex tasks makes it a compelling choice. This is particularly relevant in a scenario where predictable performance and the ability to handle increasing loads without a proportional increase in resource consumption is key to the success of the application.

d. CPU usage analysis
In the trials conducted, the CPU usage analysis showed that GetX, on small data sets, efficiently reduced the CPU load compared to BLoC. BLoC maintains a more even CPU usage profile as the data volume increases, whereas GetX shows more significant fluctuations. These findings imply that BLoC may be preferable for applications that process large data volumes consistently, offering the stability required in wide-scale operations. In contrast, GetX optimizes applications with lighter and varied processing demands. Application needs to handle sudden spikes in data or complex real-time operations. This variability in GetX's performance suggests that while it excels in less intensive data processing scenarios, its ability to scale efficiently for heavier workloads may be limited. This is an important consideration for developers planning applications that may experience variable or increasing data loads over time. In addition, the difference in CPU usage patterns between GetX and BLoC may also influence the choice of architecture in terms of long-term maintenance and scalability. BLoC's consistent performance under varying data loads makes it a more predictable framework to work with. This predictability is critical for long-term application maintenance, as developers can anticipate how the application will behave as it scales. For large applications or those expected to grow significantly, this can lead to more efficient resource allocation and a better overall management strategy. In summary, analysing the CPU usage of BLoC and GetX provides valuable insights into their suitability for different types of applications. While GetX shows excellent performance with lighter tasks, its fluctuating CPU usage at higher data volumes could pose a challenge in more demanding scenarios. BLoC's consistent performance under heavy data loads positions it as a more reliable choice for complex, large-scale applications. The decision between these two frameworks should therefore be based on the specific performance requirements of the application, taking into account factors such as data volume, scalability, long-term maintenance and energy efficiency.

e. Memory usage analysis
Memory usage testing highlighted significant differences between GetX and BLoC, especially on smaller datasets. GetX showed higher memory efficiency in scenarios that required light data processing, which could indicate its leaner approach and more efficient resource management. However, when applications experience increased data demands on larger datasets, both libraries tend to perform similarly, with GetX showing comparative efficiency to BLoC. Given that GetX excels in memory management for applications with small datasets, it is an efficient choice for applications requiring lightweight and responsive resource management. On the other hand, BLoC shows comparable memory efficiency on large-scale applications, which is suitable for projects with higher complexity and data volume. With limited storage capacity. GetX's ability to effectively manage resources in these scenarios makes it an ideal choice for developers looking to create lightweight, smooth applications that are accessible across a wide range of devices with varying memory capabilities. As data requirements increase with larger data sets, the memory consumption of GetX and BLoC begin to converge. This observation suggests that both frameworks are equipped to manage memory efficiently when dealing with larger amounts of data. This is particularly relevant for more complex applications, where data volumes may escalate as the application scales or as more features are added. In such scenarios, the comparable efficiency of GetX and BLoC in terms of memory usage means that the choice between them can be based more on other factors, such as their architectural patterns, ease of use or compatibility with the project's requirements. It's important to note that while GetX continues to demonstrate memory efficiency with larger datasets, BLoC's similar performance under these conditions is noteworthy. This suggests that BLoC, despite its potentially more structured and verbose approach, does not necessarily impose a significant memory

Journal of Information System Research (JOSH)
Volume 5, No. 2, Januari 2024, pp 583−591
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v5i2.4698

overhead. This aspect makes BLoC a viable option for large-scale applications where memory efficiency is as critical as handling complex state management scenarios. In summary, the memory usage analysis shows that GetX is particularly efficient at managing resources in applications with smaller datasets, making it an excellent choice for lightweight, responsive applications. Meanwhile, BLoC, with its comparable memory efficiency in larger applications, is suitable for projects that require high complexity and larger data volumes. This insight allows developers to make more informed decisions when choosing the appropriate framework, ensuring that their application not only meets performance requirements today, but is also scalable and resource-efficient as it evolves.

4. CONCLUSION

Choosing between BLoC and GetX requires an in-depth evaluation of the application's architectural and scalability needs. This decision depends on current performance and how each library can support future application growth. This consideration involves a thorough understanding of various application usage scenarios and a prediction of how the application will evolve in terms of features and users. Proper selection enables the development of applications that are efficient today and adaptive and scalable for future needs. In this review, BLoC showed better ability in managing CPU load at larger data scales. This indicates that BLoC may be more suitable for large-scale applications that face complex and continuous data transactions. The decision in choosing between BLoC and GetX should be based on a comprehensive analysis of the application's specific needs, including architectural considerations and long-term scalability potential. The right choice will determine the application's ability to evolve and adapt to changing market and technology demands. It's important to take a closer look at how BLoC and GetX can influence not only the technical aspects of a Flutter application, but also the development process and team dynamics. BLoC's structured approach can enforce a level of discipline in large teams or projects where maintaining a clean separation of concerns is paramount. Its emphasis on immutability and explicit event handling can lead to more predictable and testable code, which is a significant advantage in enterprise-level applications or projects with complex business logic. However, for rapid prototyping or smaller projects where flexibility and speed are more important, this rigour can slow things down. Conversely, GetX's less verbose and more flexible nature can significantly speed up development, making it ideal for start-ups or projects with tight deadlines. Its ability to handle state management, dependency injection and routing with minimal boilerplate reduces the learning curve for new developers and allows for faster feature development. However, this can be a double-edged sword, as the flexibility offered by GetX can lead to less structured code, which can become a challenge as the project grows. It's also important to consider the future maintenance of the application. BLoC, with its more explicit and verbose approach, can make the codebase more readable and maintainable in the long term, especially for new team members who may join the project later. GetX, while concise, may require more documentation and coding standards to maintain clarity in a larger codebase. In summary, the choice between BLoC and GetX goes beyond mere technical capabilities. It involves considerations of team dynamics, project timelines, maintenance expectations and performance requirements. This decision should be made in the context of the overall project strategy, taking into account both current needs and future growth. An informed choice between these two powerful tools can have a significant impact on the success and scalability of a Flutter application.

REFERENCES

[1] K. Jakimoski and A. Andonoska, “Performance Evaluation of Mobile Applications,” 2018. [Online]. Available: https://www.researchgate.net/publication/337437805
[2] E. GÜLCÜOĞLU, A. B. USTUN, and N. SEYHAN, “Comparison of Flutter and React Native Platforms,” Journal of Internet Applications and Management, Dec. 2021, doi: 10.34231/iuyd.888243.
[3] M. Alif Al Gibran Arif, D. Sulistyo Kusumo, and S. Yulia Puspitasari, “Optimasi Pengembangan Aplikasi Cross-platform Berbasis Flutter Menggunakan Pendekatan Arsitektur Model MVI (Model-View-Intent),” vol. 8, no. 5, p. 10728, 2021.
[4] M. M. F. Abdillah, I. L. Sardi, and A. Hadikusuma, “Analisis Performa GetX dan BLoC State Management Library Pada Flutter Untuk Perangkat Lunak Berbasis Android,” Jurnal Penelitian Informatika, vol. 1, pp. 73–78, 2023, doi: 10.25124/logic.v1i1.6479.
[5] S. Garg and N. Baliyan, “Comparative analysis of Android and iOS from security viewpoint,” Computer Science Review, vol. 40. Elsevier Ireland Ltd, May 01, 2021. doi: 10.1016/j.cosrev.2021.100372.
[6] M. Hort, M. Kechagia, F. Sarro, and M. Harman, “A Survey of Performance Optimization for Mobile Applications,” IEEE Transactions on Software Engineering, vol. 48, no. 8, pp. 2879–2904, Aug. 2022, doi: 10.1109/TSE.2021.3071193.
[7] K. Tasneem, A. Siddiqui, and A. Liaquat, “Android Memory Optimization,” Int J Comput Appl, vol. 182, no. 41, pp. 36–43, Feb. 2019, doi: 10.5120/ijca2019918504.
[8] Apigee, “Apigee Survey: Users Reveal Top Frustrations That Lead to Bad Mobile App Reviews.” 2020
[9] AppDynamics Team, “The App Attention Index 2019: The Era of the Digital Reflex.” 2020
[10] M. Abdul Hakeem, M. Abdul Razack Maniyar, M. Khalid Mubashir Uz Zafar, and R. Scholar, “Performance Testing Framework for Software Mobile Applications,” Int J Innov Res Sci Eng Technol, vol. 7, p. 6225, 2020, [Online]. Available: www.ijirset.com
[11] Flutter Team, “List of state management approaches.” 2023

Journal of Information System Research (JOSH)
Volume 5, No. 2, Januari 2024, pp 583−591
ISSN 2686-228X (media online)
https://ejurnal.seminar-id.com/index.php/josh/
DOI 10.47065/josh.v5i2.4698

[12] J. Mantik, R. Rama Prayoga, G. Munawar, R. Jumiyani, and A. Syalsabila, “Performance Analysis of BLoC and Provider State Management Library on Flutter,” 2021.
[13] E. Triandini et al., “Metode Systematic Literature Review untuk Identifikasi Platform dan Metode Pengembangan Sistem Informasi di Indonesia,” 2019. [Online]. Available: https://www.google.com
[14] Unterkalmsteiner, M., et al., "Software Startups - A Research Agenda," E-Informatica Software Engineering Journal, 2016, 89-124, https://doi.org/10.5277/e-Inf160105
[15] J. Smołka, B. Matacz, E. Łukasik, and M. Skublewska-Paszkowska, “Performance analysis of mobile applications developed with different programming tools,” MATEC Web of Conferences, vol. 252, p. 05022, 2019, doi: 10.1051/matecconf/201925205022.
[16] A. Tashildar, N. Shah, R. Gala, T. Giri, and P. Chavhan, "Application Development Using Flutter," International Research Journal of Modernization in Engineering Technology and Science, vol. 2, no. 8, 2020
[17] Hupp Technologies Pvt. Ltd. “State Management in Flutter: Provider, Riverpod, and BLoC,” Aug.11, 2023.[Online]. Available: https://hupp.tech/blog/programming/state-management-in-flutter-provider-riverpod-and-bloc/
[18] N. Kumar, “State management in Flutter: A comprehensive guide,” Medium, Sep.10,2023. [Online]. Available: https://medium.com/@enitinmehra/state-management-in-flutter-a-comprehensive-guide-7212772f026d
[19] Flutter by Example, “What Are BLoCs?,” Jul.25, 2020.[Online].Available: https://flutterbyexample.com/lesson/what-are-blocs
[20] Wednesday Solutions, “A guide to implementing BLoC architecture in Flutter.” LinkedIn, 2023.[Online]. Available: https://www.linkedin.com/pulse/guide-implementing-bloc-architecture-flutter-wednesday-solutions/
[21] Aruna Technology, “Berkenalan dengan GetX State Management,” Medium, Mar.31, 2023.[Online].Available: https://medium.com/arunatech/berkenalan-dengan-getx-state-management-a800b555bcb8
[22] N. Tanwar, “ Understanding GetX state management solution in Flutter,” LinkedIn, Oct. 3, 2023. [Online]. Available: https://www.linkedin.com/pulse/understanding-getx-state-management-solution-flutter-neha-tanwar/
[23] Anirudh, “Dart DevTools,” FlutterDevs, Jul. 17, 2020 .[Online]. Available: https://medium.flutterdevs.com/dart-devtools-ab7042100570