Development of FASIH Application for the Badan Pusat Statistisk using Flutter Framework

R Prasetia[1], **L R Maghfiroh**[1,*]

[1] Politeknik of Statistik STIS, Jl. Otto Iskandardinata No.64C, Jakarta, Indonesia

* Corresponding author's e-mail: lutfirm@stis.ac.id

Abstract. One of the data collection methods used by the Badan Pusat Statistik (BPS) is Computer Assisted Personal Interviewing (CAPI). Currently, CAPI, known as FASIH, is continuously updated by BPS using the Kotlin programming language, which can run on the Android platform. It is possible that FASIH will be needed in a multiplatform form. However, there is an alternative for multiplatform application development, namely Flutter, which can be used in the development of FASIH. Nevertheless, BPS has not conducted any study on the development of the FASIH application using Flutter, hence the strengths and weaknesses of implementing this technology in FASIH application development remain unknown. Therefore, the author aims to conduct a study on the development of the FASIH application utilizing Flutter. The application development is carried out using the Rapid Application Development (RAD) Prototyping method. The resulting application is tested using black box testing and performance testing using a third-party application, Apptim. The black box testing results indicate that the application meets the functional requirements of stakeholders. In terms of performance, the Kotlin version of FASIH outperforms the Flutter version. However, Flutter has an advantage in accelerating development time. Additionally, concerning user interface development, the Flutter version of the FASIH application can run on multiple platforms. Nevertheless, further integration is required to ensure the proper functioning of the Flutter version of the FASIH application.

1. Introduction

According to Law Number 16 of 1997, the Badan Pusat Statistik (BPS) is responsible for conducting basic national, macro, and cross-sectoral statistical activities that are beneficial for the government and society [1]. BPS has a business process outlined in the Statistical Business Framework and Architecture (SBFA), which consists of eight phases: determining needs, designing, building, collecting, processing, analyzing, disseminating, and evaluating. The Corporate Statistical Infrastructure (CSI) serves as the foundation for new production processes, including databases and warehouses to support data management (meta), data collection, processing, and distribution. Within the CSI, an Integrated Collection System (ICS) is one of its components, which includes various modules such as Collection Tools for inputting data from the field into the database. Collection Tools consist of four modes: Computer Assisted Personal Interviewing (CAPI), Computer Aided Personal Interviewing (CAWI), Desktop Data Entry (DDE) for Paper and Pencil Interviewing (PAPI), and External Data Acquisition [2]. CAPI is more suitable for officers in areas with good internet connectivity. As a result, the business process in each district can vary depending on the type of survey. For example, in BPS districts or cities that use CAPI as a survey method, survey officers must use CAPI to collect data.

Until now, CAPI is still being developed as a mobile application that can only run on Android smartphones. Based on the need to accelerate the data collection process, the development of CAPI is underway [2]. Interviews with stakeholders indicate that the procurement of CAPI follows a working reference framework that provides flexibility in choosing technology that suits functional needs. The development of CAPI is implemented using the Kotlin programming language in a Native form. Native refers to applications built to run on a specific platform only (single platform) [3]. However, it is possible that the CAPI application may be required in a multiplatform form. Currently, CAPI is known as FASIH and continues to be updated.

FASIH is an innovative application meticulously developed by the Central Statistics Agency (BPS), embodying a pivotal method for data collection known as Computer Assisted Personal Interviewing (CAPI). Currently engineered utilizing the Kotlin programming language, FASIH operates exclusively on the Android platform. Within the research context, FASIH undergoes continuous refinement and enhancement, strategically aligning with the imperative need for accelerating the data collection processes. This ongoing development signifies a committed effort to optimize and streamline the functionalities, ensuring that the application meets the evolving demands and challenges in the realm of statistical data accumulation.

In the context of mobile application development, there is an alternative that can be considered, which is the development of a multiplatform application using a single code base that can be used on various platforms [4]. Some popular multiplatform technologies include Ionic, Xamarin, Flutter, and React Native. According to Palumbo's research [4], the Flutter framework is a suitable choice for building multiplatform mobile applications compared to other frameworks because the research findings show that Flutter outperforms React Native in terms of Central Processing Unit (CPU) usage, Memory usage, and execution time. The Stack Overflow survey in 2022 also states that Flutter is the most popular multiplatform technology currently [5]. Flutter is an open-source framework by Google for building multiplatform applications [6]. In this context, Flutter needs to be considered as a technology for the future development of the FASIH application because it supports multiplatform application development. Another reason for using Flutter is that there are two dominant mobile operating systems widely used in Indonesia from 2019 to 2023, namely iOS and Android [7]. Additionally, FASIH also exemplifies enhanced developmental velocity compared to utilizing native technologies. This acceleration in development timelines signifies a distinctive advantage of employing the Flutter framework, thereby optimizing the efficiency and productivity of the application development process. This strategic utilization of Flutter fosters a conducive environment for rapid and agile application development, facilitating timely and effective responses to evolving requirements and technological advancements [3].

The researcher also conducted a preliminary survey involving respondents who were involved in recruiting potential field survey officers using CAPI within one year. The survey results showed that 42.86 percent of respondents rejected potential partners because they only had non-Android smartphones. Additionally, the ICS was a finalist in the Top Public Service Innovation competition for Public Service Innovation in the Ministry/Institution Environment, Local Governments, State-Owned Enterprises (BUMN), and Regional-Owned Enterprises (BUMD) in 2022. This indicates the potential of ICS as a public service innovation that can be used by everyone. Therefore, developing the FASIH application using the Flutter framework is a future step to develop the FASIH application as part of the ICS innovation operating on multiple platforms. However, for deploying iOS applications, an Apple Developer Account is required, which costs US$99 per year ([8];[6]). Hence, in this research, FASIH development is conducted using the Flutter framework without deploying the application for iOS.

In this context, a technological transition from Kotlin to Flutter is possible due to the limited number of FASIH programmers, which is only two individuals. Developing native applications can require a large budget and involve many teams [9]. However, the advantages of this multiplatform technology are sometimes balanced by limitations, especially in terms of performance and integration [4]. Since there have been no previous studies on the development of the FASIH application using Flutter, this research aims to study the development of the FASIH application using Flutter.

This research aims to delve into the development of the FASIH application using the Flutter framework, focusing on producing outputs such as source code, documentation, implementation, and results of performance comparison analysis based on application testing. The objectives are twofold: firstly, to develop the FASIH application using the Flutter framework, and secondly, to conduct a comparative study between the Kotlin version and the Flutter version of the FASIH application.

There are several studies related to this research. In Widiyanto's research, the development of a personnel system was examined using three methods to compare their strengths and weaknesses. The methods used were Rapid Application Development (RAD), Waterfall, and Prototype. RAD was found to be highly effective in producing systems that can be directly discussed with stakeholders and is suitable for projects with tight time constraints [10]. Therefore, in the development of the FASIH application using the Flutter framework, the RAD development method was chosen.

Grzmil conducted a performance comparison between basic applications built using Android Native and Flutter. Testing was performed using a third-party application called Apptim, which provided parameters for comparison, such as CPU usage, Memory usage, Power Usage, and FPS Render [11]. Hence, in this research, the Apptim tool was utilized for application performance testing.

In Arif & Ali's study, various tools for application testing were recommended, including options for testing on real devices or emulators for mobile applications. The choice of testing tool depends on the specific testing requirements [12]. In this research, the Apptim application and emulator were used for testing purposes.

Agrawal et al. compared application development using Flutter with native programming languages in the development of mobile applications. The comparison parameters included code size, CPU performance, development time, and application appearance. The parameter of CPU performance was used in this study to compare the CPU usage between the Flutter version and the Kotlin version of the FASIH application [3]. Therefore, several parameters, such as development time and performance, were taken into account as a comparison between the FASIH application developed with Flutter and the FASIH application developed with Kotlin.

2. Research Metodology

In this research, a multifaceted approach to data collection was employed to garner a comprehensive understanding of the subject matter, ensuring that various aspects and nuances of the FASIH application and its development process were meticulously explored and analyzed. Interviews served as a pivotal method in this multifaceted approach, facilitating the acquisition of firsthand insights and perspectives from key stakeholders involved in the FASIH application's development at the Directorate of Statistical Information Systems, BPS. Conducted through various mediums such as Zoom meetings and WhatsApp, these interviews aimed to unravel the intricacies of the system, identifying critical issues and areas necessitating improvement or modification.

Complementing the interviews, a thorough literature review was undertaken, encompassing a diverse array of sources such as ebooks, theses, articles, and journals. This method enabled the assimilation of foundational knowledge and theoretical perspectives pertinent to the research, fostering a well-rounded understanding that was instrumental in informing various stages of the study.

Observation and documentation study further enriched the data collection process, enabling a direct and in-depth exploration of the FASIH application's user interface and source code. This approach facilitated a nuanced understanding of the application's functionalities, complementing and enhancing the insights garnered through interviews.

Lastly, an initial survey was conducted using the snowball sampling method. This technique was chosen due to its effectiveness in identifying and accessing populations that are difficult to reach or poorly organized [13]. In the context of this research, this method facilitates the researcher in accessing and collecting data from respondents with specific involvement and experience in the recruitment process using the FASIH application. The choice of the snowball sampling method was made because there is no existing list of BPS employees who have conducted partner recruitment. Consequently, this allows for the acquisition of rich and relevant data to support the research objectives. This survey aims

to elucidate the practical implications and challenges encountered in the application's usage, such as the rejection of potential partners due to the absence of Android smartphones, thereby providing practical insights to inform the direction and focus of the research.

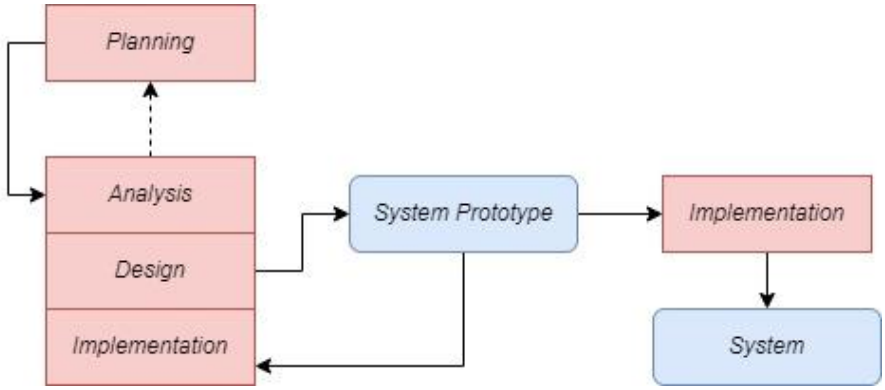

Figure 1. RAD Prototyping Method Cycle [14]

The Rapid Application Development (RAD) Prototyping method is a development methodology that prioritizes speed. RAD Prototyping is a prototyping-based methodology that simultaneously performs analysis, design, and implementation processes in iterative cycles until the application system is accepted [14]. In this research, the FASIH team provided feedback during the analysis, design, and implementation processes, and iteratively accepted the system. The stages of this method are as follows (See Figure 1).

This stage involves determining the scope and objectives of the system to be developed, as well as identifying the existing problems. Problem identification is done through interviews with stakeholders and a study of the source code of the FASIH application's Android version, developed using the Kotlin language. In the initial phase of planning, the research delineates the scope and objectives of the system under development, identifying prevailing issues through stakeholder interviews and a meticulous study of the FASIH application's existing Android version source code, crafted using the Kotlin language. This stage is pivotal for establishing a foundational understanding and direction for the subsequent phases of the development process.

Transitioning into the analysis stage, a deeper exploration and comprehension of the system’s requirements and specifications are pursued. Engaging in a thorough analysis, the author assimilates stakeholders' requirements, discerns the intrinsic business processes, and accumulates pertinent information, incorporating the creation of an Activity diagram to visually encapsulate the sequential flow of business activities [15].

Following the analysis, the design phase unfolds, focusing on the conceptualization of the system’s structure and interface. This involves a meticulous design process where the user interface is crafted, and the system’s architectural blueprint is devised, laying the groundwork for the implementation phase.

The implementation phase signifies the translation of the meticulously crafted designs into executable program codes, embodying the envisioned specifications. Subsequent to the initial implementation, a prototype of the system is constructed, embodying fundamental features for preliminary usage and evaluation. This prototype undergoes iterative cycles of stakeholder evaluation and refinement, ensuring alignment with stakeholder expectations and requirements.

Preceding the testing phase, the application, cultivated through the Rapid Application Development (RAD) methodology, is subjected to black box testing, emphasizing the verification of requirements congruent with the stipulated specifications [14]. Performance testing ensues, utilizing third-party applications like Apptim, facilitating the acquisition of critical performance parameters such as RAM usage and frames per second (FPS).

An evaluative comparison is conducted, juxtaposing the performance of specific features between the Flutter-developed FASIH application and its Kotlin-developed counterpart, operational on Android

mobile devices. This evaluative process encompasses testing on an emulator, leveraging its flexibility and expedited testing capabilities [16], culminating in a comparative analysis informed by the developmental and testing outcomes. Conclusively, the development timeline is ascertained through a meticulous examination of the commit history within the version control of the FASIH application, corroborated by the developers at BPS.

3. Theoretical Framework

This research begins with the development of the FASIH application using the Kotlin programming language. The selection of Kotlin as the programming language for the FASIH application was determined by external developers since there were no BPS team members with expertise in Kotlin. However, there is an alternative technology, Flutter, which allows for multi-platform application development, making it a viable solution for the development of the FASIH application. Nevertheless, no studies have been conducted on the development of the FASIH application using Flutter. Therefore, the proposed solution in this research is to conduct a study on the development of the FASIH application using Flutter. Subsequently, a study is conducted to compare the FASIH application developed with Flutter and the FASIH application developed with Kotlin. The development of the FASIH application with Flutter utilizes the Rapid Application Development (RAD) Prototyping methodology (See Figure 2).

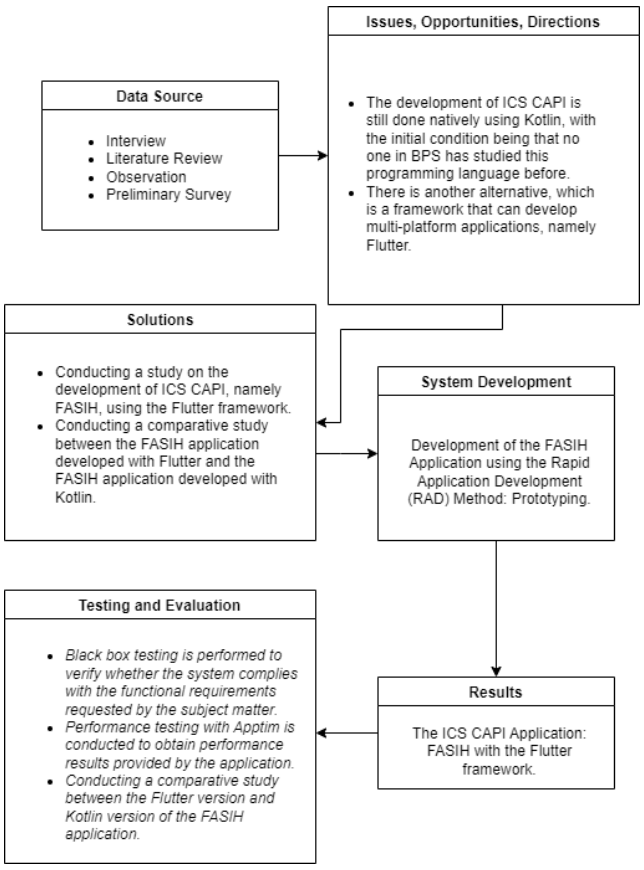

Figure 2. Theoretical Framework of the Research

4. Results and Discussion

The scope of the system's development was meticulously delineated, grounded in the functional requirements gleaned from direct observations of the FASIH application's Android version, corroborated by affirmations from the FASIH team. These boundaries were strategically established by

the team, underscoring features deemed indispensable, thereby necessitating their incorporation within the FASIH application.

Embarking on the trajectory of continuous development, the application is envisioned to embody a spectrum of functionalities, pivotal for enhancing its operational efficacy and user experience. Foremost, the system is envisaged to facilitate seamless login and logout processes, integrated adeptly with the BPS SSO system, fostering a cohesive user experience for partners.

In alignment with the imperatives of data integrity and consistency, the system is engineered to support proficient data synchronization capabilities. Augmenting its functional repertoire, the system is also designed to facilitate the downloading of FormGear, fostering enhanced accessibility and convenience.

A testament to its integrative design philosophy, the system is imbued with capabilities facilitating seamless integration with FormGear, promoting operational synergy and coherence. Central to its user-centric design approach, the system is configured to empower users to effortlessly navigate through questionnaires, enabling them to input responses, save the populated data, and subsequently submit the questionnaire results, thereby enhancing the system's usability and functional versatility.

In the implementation phase of the third iteration, the design of the FASIH application's interface by the FASIH team was implemented using Flutter and the Dart programming language (design example shown in Figure 3). The result was a prototype representing the functionality of the previous interface design. This prototype was evaluated by stakeholders, especially the FASIH team, and their feedback was used to improve the interface to meet the users' vision and needs.

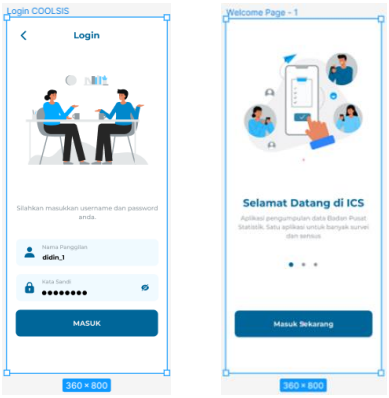

Figure 3. Login Page Design (left) and Welcome Page (right)

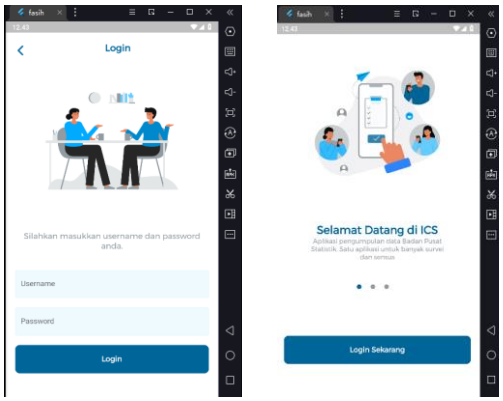

Figure 4. Login Page Implementation (left) and Welcome Page (right)

The implementation process successfully incorporated relevant functionalities with the existing FASIH application (implementation example shown in Figure 4). The results indicated the alignment of the interface with the predetermined design and provided a solid foundation for further development of the FASIH application.

The second iteration phase focused on developing the application according to functional requirements and improving the interface design for user-friendliness. The analysis involved the analysis of the running system, problems, and needs (Results of the need analysis can be seen in Table 1). Changes were made to align the application with the functional limitations and enhance the efficiency of the features. The problem analysis identified difficulties in selecting surveys, data model complexity, and extension of synchronization processes.

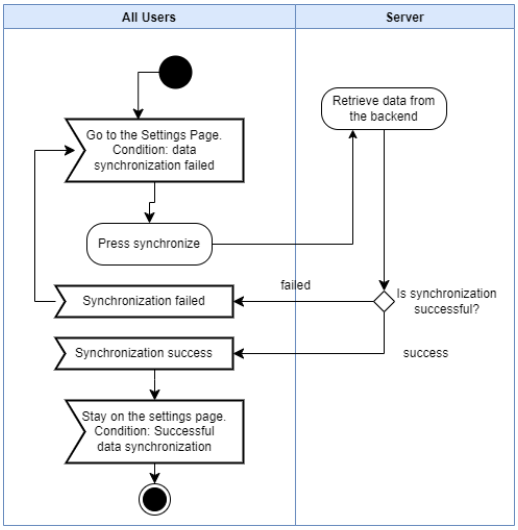

Figure 5. Activity Diagram Design on the synchronization feature

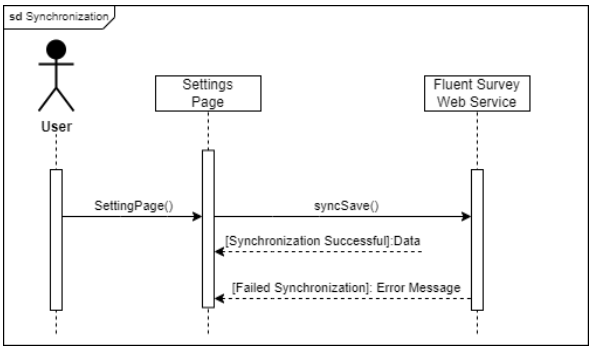

Figure 6. Sequence Diagram Design on the synchronization feature

The need analysis resulted in solutions to address these problems. The design phase involved creating activity diagrams, data modeling, class diagrams, and sequence diagrams as implementation guides (activity and sequence diagram design examples shown in Figure 5 and Figure 6). The interface design underwent changes in the FASIH application being developed using Kotlin. Development was done using Flutter, incorporating features such as FormGear download, data synchronization, live search, notification processes, and memory usage information (implementation example of the synchronization feature shown in Figure 7). The prototype was evaluated with positive feedback from stakeholders.

Table 1. Results of the Second Iteration Phase Analysis: Functional Requirements

No	Requirement
1	The system should be able to synchronize data stored in the database with a single touch.
2	The system should be able to download the Engine.
3	There should be a live search feature on the list of surveys, regions, and assignments.
4	The system should provide notifications if any process fails.
5	The system should provide information on the memory usage.

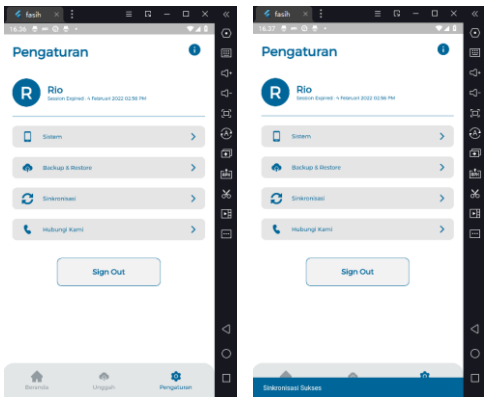

Figure 7. Implementation of the data synchronization feature (left) and successful synchronization process (right)

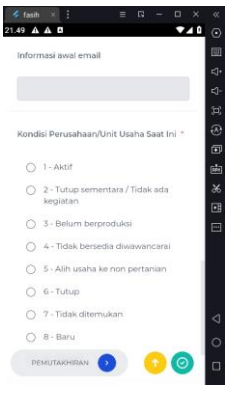

Figure 8. Questionnaire Page Implementation with FormGear

In the analysis phase of the third iteration, the main goal was to develop the established functional limitations and replicate the functionality of the Kotlin version of the FASIH system. The analysis involved a deep understanding of the previously established needs and functional requirements (Results of the analysis can be seen in Table 2). The focus remained on developing functional limitations and replicating the functionality of the Kotlin version of the FASIH system. The analysis results aimed to achieve the desired functionality. The prototype evaluation received valuable feedback. Important features successfully implemented included login, logout, questionnaire progress display, and integration with FormGear to generate questionnaires. The prototype system could accommodate questionnaire filling, data storage, and submission of questionnaire results (FormGear implementation can be seen in Figure 8). In conclusion, the prototype development successfully achieved its objectives with important features that meet the needs.

Table 2. Results of the Third Iteration Phase Analysis: Functional Requirements

No	Requirement
1	The system should be able to perform Login and Logout.
2	The system should display progress on opened questionnaires, sent processes, successful submissions, and failed submissions.
3	The system should integrate with FormGear.
4	The system should enable questionnaire filling, save filled-in data, and submit questionnaire results.

Table 3. Black Box Testing Results

Kelas Uji	Hasil yang diharapkan	Hasil
Login	The system can perform login with BPS SSO	Valid
Logout	The system can perform logout	Valid
Synchronization	The system can perform synchronization	Valid
Download FormGear	The system can perform engine download	Valid
Live search	The system can filter surveys, regions, and assignments	Valid
Guide information	The system can display application guide information	Valid
Send feedback for the application	The system has a menu interface to send criticisms and suggestions	Valid
Failed notification	The system can display notifications if there are failed processes	Valid
Check progress	The system can display assignment progress status	Valid
Check memory info	The system can display memory usage information	Valid
Questionnaire	The system can fill out questionnaires	Valid

Based on the results of the black box testing (in Table 3), all functions are in line with the functional requirements set by stakeholders.

The development time for features in the FASIH application in Flutter and Kotlin was obtained from the commit history documentation in the FASIH application's GitLab and validated with the development team. Based on the development time (see Table 4), Flutter had a faster development time compared to Kotlin. One of the factors supporting this faster development is that Flutter has a hot reload feature. Hot reload is used to inject updated source code files into the running Dart Virtual Machine (VM) [17].

Table 4. Difference In Development Time For Some Features

Developed Feature	Flutter (Day)	Kotlin (Day)
Login with BPS SSO	12	26
Synchronization	18	125
Download FormGear	18	111
FormGear integration	37	172

In the development phase of the FASIH User Interface, it can be developed across multiple platforms such as Desktop, Web, and iOS. This shows that Flutter can facilitate multi-platform development with a single codebase. If Flutter is applied to other ICS modes, it would be possible to integrate other development teams such as Computer Assisted Personal Interviewing (CAPI), Computer Aided Personal Interviewing (CAWI), and Desktop Data Entry (DDE) for Paper and Pencil Interviewing (PAPI) into one project.

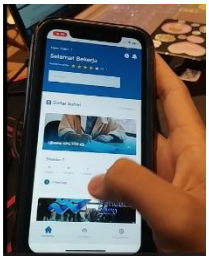

Figure 9. Homepage on iOS User Interface Implementation

Figure 10. Homepage on Web User Interface Implementation

Figure 11. Homepage on Desktop User Interface Implementation

However, further integration is required as not all packages used are compatible with multi-platform development. The implementation results of the User Interface can be seen in (iOS: Figure 9, Web: Figure 10, and Desktop: Figure 11).

According to Stallings [18], the CPU is used for data processing and computer operation control. Abnormally high CPU usage can result in slow or laggy application performance [19]. In (see Figure 12 and Figure 13), the testing results show that the Flutter version of the FASIH application uses more CPU and memory compared to the Kotlin version. This indicates that the Kotlin version of the FASIH application has a lower chance of crashing or lagging during operation. This observation is also supported by the application's start time as shown in (see Figure 14), where the Flutter version has a longer start time compared to the Kotlin version.

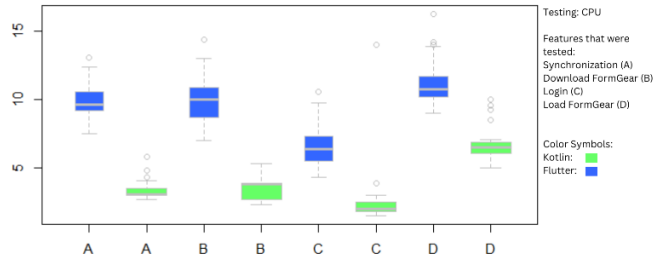

Figure 12. CPU Usage Testing Results (%) for the synchronization feature (A), FormGear download (B), login (C), and load FormGear (D).

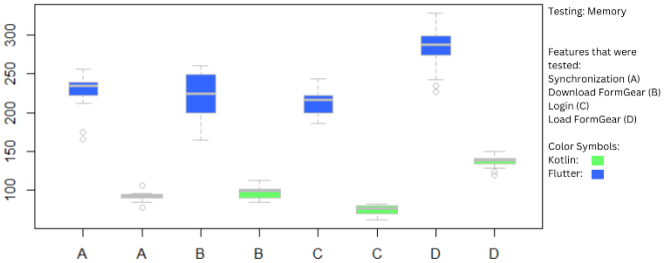

Figure 13. Memory Usage Testing Results (MB) for the synchronization feature (A), FormGear download (B), login (C), and load FormGear (D).

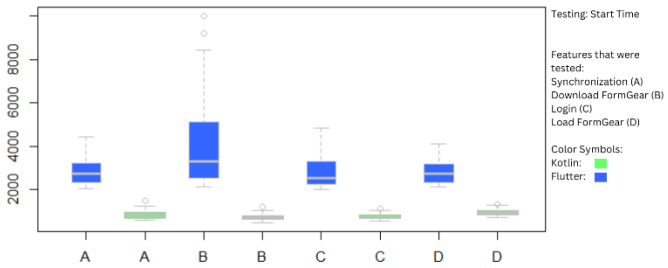

Figure 14. Start Time (ms) for the synchronization feature (A), FormGear download (B), login (C), and load FormGear (D).

In this performance testing, the frame per second (FPS) rate is also measured. FPS refers to how frequently images are displayed. The higher the FPS, the smoother animations are generated in the application [20]. Based on (see Figure 15), the testing results show that the Flutter version of the FASIH application has smoother animations for the synchronization feature, while the Kotlin version has smoother animations for the FormGear download, login, and load FormGear features.

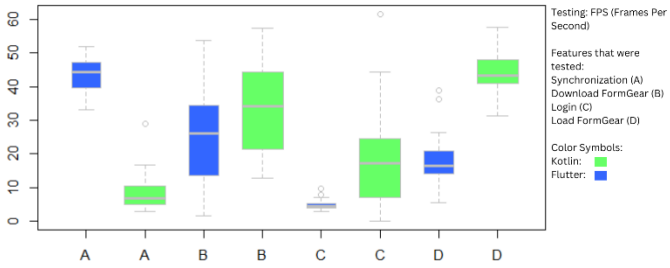

Figure 15. Frame per Second (FPS) Testing Results for the synchronization feature (A), FormGear download (B), login (C), and load FormGear (D).

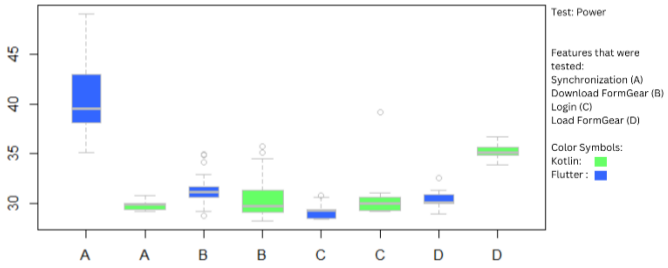

Figure 16. Power Usage Testing Results (%) for the synchronization feature (A), FormGear download (B), login (C), and load FormGear (D).

Regarding power usage, in some cases, the features show different results. According to Figure 16, the testing results show that for the synchronization and FormGear download features, the Flutter version of the FASIH application consumes more power compared to the Kotlin version. On the other hand, for the login and load FormGear features, the Kotlin version of the application consumes more power compared to the Flutter version.

5. Evaluation

The development of the FASIH application with Flutter Framework has been successfully carried out with the specified requirements from stakeholders. This is evidenced by the Black box testing, where all features developed have valid results for all tested scenarios. Based on the development time, Flutter has a faster development time compared to development with Kotlin. One of the supporting factors for

this short development time is that Flutter has a hot reload feature, which allows for rapid code injection during application development [17].

Furthermore, during the development of the User Interface, the Flutter version of the FASIH application can be run on iOS, Web, and Desktop platforms. This allows for collaboration with other ICS mode teams, as multi-platform development can be achieved with a single codebase. However, further integration is needed as some packages used in this research are not compatible with multi-platform development.

Additionally, in terms of performance, the Kotlin version of FASIH outperforms the Flutter version in terms of average memory usage, CPU usage, and application start time. This may be attributed to Kotlin having optimized dependencies for Android applications compared to Flutter, which utilizes dependencies for multi-platform compatibility (but not as optimized as Kotlin).

6. Conclusion

Based on the research findings, the following conclusions can be drawn: the development of the FASIH application with the Flutter framework can be conducted in accordance with the functional requirements set by stakeholders. The developed application can be compared with the Kotlin version of the FASIH application owned by BPS through testing using a third-party application, Apptim. Furthermore, from the performance testing results, the Flutter version of the FASIH application still cannot match the performance of the Kotlin version in terms of CPU usage, memory usage, and application start time. However, Flutter has advantages in terms of fast development time and the potential for multi-platform development. Therefore, the development of the FASIH application with Flutter can be considered as an alternative for future FASIH application development, especially considering its flexibility and time efficiency.

The authors have made efforts to optimize the performance of the Flutter application to approach the performance of the Kotlin version by reducing database communication and eliminating warnings. However, the results are still far from expectations. The authors recommend further research to conduct a more in-depth study of Flutter application performance in order to achieve optimization and bring the performance of the Flutter version closer to that of the Kotlin version.

References

[1] UU RI No. 16 Tahun 1997, "Presiden republik indonesia," *Peratur. Pemerintah Republik Indones. Nomor 26 Tahun 1985 Tentang Jalan*, vol. 2003, no. 1, pp. 1–5, 2004, [Online]. Available: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=2ahUKEwjWxrKeif7eAhVYfysKHcHWAOwQFjAAegQICRAC&url=https%3A%2F%2Fwww.ojk.go.id%2Fid%2Fkanal%2Fpasar-modal%2Fregulasi%2Fundang-undang%2FDocuments%2FPages%2Fundang-undang-nomo.

[2] Biro Perencanaan, "Perjalanan Transformasi BPS STATCAP-CERDAS Statistical Capacity Building: Change and Reform for The Developent of Statistics STATCAP-CERDAS," in *Publikasi Lainnya*, JAKARTA: BPS RI, 2021, p. 122.

[3] A. Agrawal, A. Agrawal, R. Arya, H. Jain, J. Manoorkar, and A. Professor, "Comparison of Flutter with Other Development Platforms," *Int. J. Creat. Res. Thoughts*, vol. 9, no. 2, pp. 2320–2882, 2021, [Online]. Available: www.ijcrt.org.

[4] D. Palumbo, "The Flutter Framework: Analysis in a Mobile Enterprise Environment," POLITECNICO DI TORINO, 2021.

[5] StackOverflow, "Other frameworks and libraries," 2022. https://survey.stackoverflow.co/2022/#other-frameworks-and-libraries (accessed Nov. 25, 2022).

[6] Flutter, "Build and release an iOS app," 2019. https://docs.flutter.dev/deployment/ios (accessed Nov. 18, 2022).

[7] Statista, "Market share of mobile operating systems in Indonesia from January 2013 to May 2022,

by operating system," 2022. https://www.statista.com/statistics/262205/market-share-held-by-mobile-operating-systems-in-indonesia/ (accessed Nov. 25, 2022).
[8] apple, "Apple Developer Program Enrollment," 1997. https://developer.apple.com/enroll/purchase (accessed Nov. 18, 2022).
[9] M. K. Khachouch, A. Korchi, Y. Lakhrissi, and A. Moumen, "Framework Choice Criteria for Mobile Application Development," *2nd Int. Conf. Electr. Commun. Comput. Eng. ICECCE 2020*, no. June, 2020, doi: 10.1109/ICECCE49384.2020.9179434.
[10] W. W. Widiyanto, "Analisa Metodologi Pengembangan Sistem Dengan Perbandingan Model Perangkat Lunak Sistem Informasi Kepegawaian Menggunakan Waterfall Development Model, Model Prototype, Dan Model Rapid Application Development (Rad)," *J. Inf. Politek. Indonusa Surakarta ISSN*, vol. 4, no. 1, pp. 34–40, 2018, [Online]. Available: http://www.informa.poltekindonusa.ac.id/index.php/informa/article/view/34.
[11] P. Grzmil, M. Skublewska-Paszkowska, E. Łukasik, and J. Smołka, "Comparative Study of Android Native and Flutter App Development," *Informatics Control Meas. Econ. Environ. Prot.*, vol. 7, no. 2, pp. 50–53, 2021, [Online]. Available: https://www.researchgate.net/publication/361208165_Comparative_Study_of_Android_Native_and_Flutter_App_Development.
[12] K. S. Arif and U. Ali, "Mobile application testing tools and their challenges: A comparative study," *2019 2nd Int. Conf. Comput. Math. Eng. Technol. iCoMET 2019*, pp. 1–6, 2019, doi: 10.1109/ICOMET.2019.8673505.
[13] K. Leighton, S. Kardong-Edgren, T. Schneidereith, and C. Foisy-Doll, "Using Social Media and Snowball Sampling as an Alternative Recruitment Strategy for Research," *Clin. Simul. Nurs.*, vol. 55, pp. 37–42, 2021, doi: 10.1016/j.ecns.2021.03.006.
[14] R. Eldridge, *System Analysis & Design An Object-Oriented A pproach with UML*, vol. 31, no. 1. 1989.
[15] J. L. Whitten and L. D. Bentley, *Systems Analysis and Design Methods*. 2007.
[16] Android Studio, "Run apps on the Android Emulator," 2013. https://developer.android.com/studio/run/emulator?hl=id (accessed Jul. 03, 2023).
[17] Flutter, "Hot reload," 2019. https://docs.flutter.dev/tools/hot-reload (accessed Jun. 20, 2023).
[18] W. Stallings, *COMPUTER ORGANIZATION AND ARCHITECTURE DESIGNING FOR PERFORMANCE*. 2010.
[19] intel, "How to Fix High CPU Usage," *intel*, 1968. https://www.intel.com/content/www/us/en/gaming/resources/how-to-fix-high-cpu-usage.html (accessed Nov. 04, 2023).
[20] T. Tamasi, "What is Frame Rate and Why is it Important to PC Gaming?," *NVIDIA Corporation*, 2019. https://www.nvidia.com/en-us/geforce/news/what-is-fps-and-how-it-helps-you-win-games/ (accessed Sep. 04, 2023).

Acknowledgments
We would like to express our deep appreciation to the Directorate of Statistical Information System (SIS) at the Statistics Indonesia (BPS) for their invaluable support and cooperation throughout the implementation of this research. The guidance and insights they provided have played a pivotal role in shaping the direction of our study. We also want to extend our special thanks to Mrs. Lutfi Rahmatuti Maghfiroh, S.S.T., M.T., for her continuous support and guidance. Her expertise and mentorship have been crucial factors in navigating the complexities of this research. Lastly, we wish to convey our gratitude to all individuals who have directly or indirectly supported the completion of this research. Their contributions have played a significant role in the successful conclusion of this study. We thank everyone for their support and encouragement.