Jurnal Teknologi dan Manajemen Industri Terapan (JTMIT) Vol. 5, No. 2, Juni 2026 pp.725 - 734
P-ISSN: 2829-0232 E-ISSN: 2829-0038

Design, Integration, And Deployment Of A Flutter–Firebase Mobile Commerce Application At Berlian Tech

Labib Falah Athallah[1*], Achmad Habib[2], Geri Kusnanto[3]
[1,2,3)] Computer Science, Faculty of Electrical Engineering and Intelligent Computing, 17 August 1945 University, Surabaya.
Email: 1. athallahlfa@gmail.com [*], habib@untag-sby.ac.id , gerikusnanto@untag-sby.ac.id

ABSTRACT

The development of e-commerce in Indonesia has shown an upward trend, with gross merchandise value (GMV) reaching USD 59 billion in 2022. This study aims to develop a mobile-based e-commerce application, with a case study of the Berlian Tech Online Store. This study employs a Research and Development (R&D) method using a software engineering approach based on the Software Development Life Cycle (SDLC) with the waterfall model. The waterfall model provides a sequential lifecycle approach, beginning with the analysis phase and proceeding to design, coding, testing, and maintenance. The requirements analysis is divided into two categories: functional requirements and non-functional requirements. The system architecture comprises four entities: authentication, products, and the shop. The software architecture adopts the Model-View-ViewModel (MVVM) pattern due to its superior CPU efficiency, with an average CPU usage of 8.92% and a memory consumption of 121.48 MB. The MVVM architecture is combined with a service layer and an API gateway. The Online Store application was successfully designed using the Model-View-ViewModel (MVVM) system architecture, combined with a service layer that facilitates communication between the API gateway and the view. The NoSQL database implemented with Firebase provides high flexibility for storing data in a non-relational structure. The results of black-box testing indicate that all core system features, including user authentication, product management, shopping cart, checkout, and transaction processing, function properly and align with user requirements.

Keywords: MVVM, Database NoSQL, Software Architecture

Introduction

The development of e-commerce in Indonesia has shown an upward trend, with gross merchandise value (GMV) reaching USD 59 billion in 2022 [1]. Based on data from the Central Statistics Agency (BPS), e-commerce transaction values in Indonesia increased by 33.2% in 2023, with an estimated 220 million internet users [2]. This growth serves as an indicator that the adoption of mobile applications is essential for businesses competing in the modern market. This study aims to develop a mobile-based e-commerce application with a case study on the Berlian Tech Online Store.

Prior to the application design process, the researcher conducted an analysis of mobile application characteristics. The characteristics of mobile applications include operating systems and network availability as independent variables, and response time, responsiveness, and interface accessibility as dependent variables [3]. Based on the literature review, e-commerce companies are advised to focus on enhancing the user experience during the checkout process to reduce shopping cart abandonment rates [1].

The technologies used to develop the e-commerce application include Flutter and Firebase as a backend-as-a-service. Flutter was selected for its ability to create mobile applications with responsive user interfaces and efficient cross-platform deployment [4]. Firestore was chosen because it facilitates seamless integration of end-user data into a cloud-hosted database, as its services are part of the Google Cloud Platform [5].

The e-commerce application was developed using the Model-View-ViewModel (MVVM) architecture. The implementation of MVVM significantly improves the separation between the view and business logic, enabling developers to manage code more efficiently while enhancing the application's scalability and maintainability [6]. Black-box testing was used to evaluate all system features and ensure they functioned as intended; based on the test results, all validations passed successfully [7]. The successful development of this application is expected to provide tangible benefits for Berlian Tech, both in terms of increased sales and improved operational efficiency.

Research Method

This study employs a Research and Development (R&D) method using a software engineering approach based on the Software Development Life Cycle (SDLC) with the waterfall model. Research and Development is a research method used to produce a specific product and evaluate its effectiveness [8]. The R&D method was

chosen because this study aims to produce a product in the form of an online shop application implemented for Berlian Tech. The SDLC, or waterfall, system development life cycle is one of the classical approaches that follow a linear, sequential model [9]. The SDLC waterfall model was selected because it aims to design and build a software system to meet Berlian Tech's needs. The waterfall model provides a sequential lifecycle approach, starting from the analysis phase, followed by design, coding, testing, and maintenance [10].

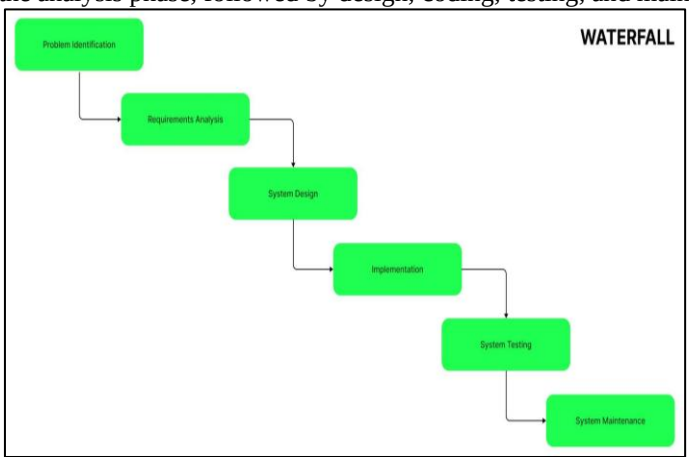

Figure 1. Waterfall Model

Requirements Analysis

To understand the abstraction or problem domain of the given case, it is necessary to establish a scenario or description of the domain that can be reviewed against predefined business processes, business rules, and business needs [11]. The requirements analysis is divided into two categories: functional requirements and non-functional requirements. Functional requirements represent the features to be developed during the iterative development stages. Non-functional requirements are formulated based on the system's non-technical needs [12]. The list of requirements is prioritised using a scale to enhance focus and productivity, ensuring that the primary requirements are met.

Tabel 1. Priority Description

Code	Priority Description
1	Low
2	Medium
3	High

Tabel 2. Functional Requirements

Code	Functional Requirements	Priority
KF01	User Registration	3
KF02	User login	3
KF03	User Logout	3
KF04	Shop page	3
KF05.a	Product search using keywords	2
KF05.b	Product filtering by price and name	2
KF05.c	Product categories	2
KF06	Product details	3
KF07.a	Shopping cart	3
KF07.b	Add and remove products in the cart	3
KF08.a	Display the total price on the checkout page	3
KF08.b	Users can select a shipping address on the checkout page	1
KF08.c	Users can choose a payment method	3
KF08.d	Users can choose a shipping method	2
KF09	Users can view transaction status and payment notifications	3
KF10.a	Users can view and edit their profile	2
KF10.b	Add and update the shipping address	1

Table 3. Non-Functional Requirements

Code	Non-Functional Requirements	Priority
Reliability		
KNF01	The system operates 24 hours a day.	3
KNF02.a	Ensures proper data integration.	3

Code	Non-Functional Requirements	Priority
Reliability		
KNF02.b	Capable of storing transaction data.	3
KNF03	Able to handle increased workloads without performance degradation.	3
Availability		
KNF04	Perform data backup three times within a 24-hour period.	3
Security		
KNF05.a	Passwords are encrypted using SHA-256.	3
KNF05.b	The system can send OTPs (One-Time Passwords).	3
KNF06	Stores data logs.	3
KNF07	Verifies checkout forms.	3
Maintainability		
KNF08	The system is updated every three months.	2
KNF09	Documents issues and problems systematically.	1
Performance		
KNF10	The system has a fast response time.	2
Software Quality		
KF11	The user interface is easy for users to understand.	2

System Architecture

The system architecture illustrates the entities involved in the application [13]. As shown in Fig 1, the system architecture is divided into four entities: authentication, which acts as the initial stage before users perform transactions, product information, shop which allows users to save products in the shopping cart and proceed to checkout with transaction status later displayed on the transaction page and external integration, which manages the process of sending OTPs and storing data through Google Firestore services.

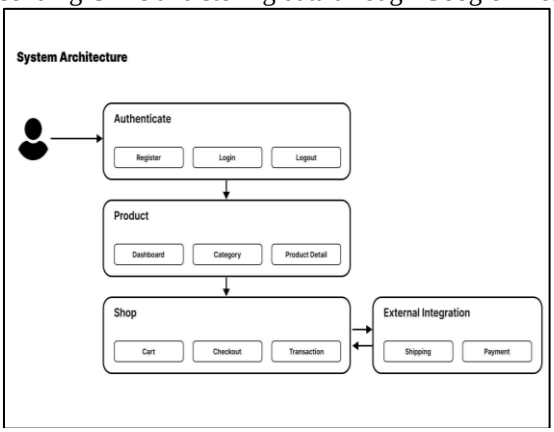

Figure 2. Functional Requirements

System Design

The modelling stage of this research encompasses all aspects of system design, including system integration, as illustrated in Fig. 2. Software model design, including use case diagrams, database design through the Entity Relationship Diagram (ERD), and user interface (UI) design [14]. Activity diagrams and sequence diagrams were also utilized. Use case diagrams serve to identify the roles of each actor [15]. Activity diagrams describe the sequence of processes or activities occurring from the beginning to the end of the system [16]. Sequence diagrams explain the order of processes within the system and the interactions among the objects involved [17].

The use case diagram in Fig. 3 illustrates the relationship between the user, the main actor and the system. The user performs registration, login, product search, cart management, transactions, profile access, and shipping address management. The activity diagram Fig 4 describes the workflow, starting with user registration, login, OTP verification, and navigation to the dashboard. Users can then search for products or view product details, add items to the cart, or proceed to checkout. On the checkout page, users must complete the required form fields, including shipping address, product quantity, shipping method, and payment method. After payment, users can view payment status and transaction details.

The sequence diagram Fig 5 illustrates the interactions between users and the system, beginning with registration and OTP verification. Successful verification results in data being stored in the database and the system displaying the shop page. After login, the system retrieves user data and proceeds with OTP verification. The dashboard displays product data retrieved from the database. Keyword-based product searches are validated, and appropriate feedback is provided. Checkout processes include form validation, storing transaction

data, and displaying the virtual account (VA) page. Upon payment, the system updates the transaction data and displays updated transaction details.

A class diagram Fig 6 is used to represent the structure of classes, attributes, and their relationships. The diagram includes six main classes: user, address, product, cart, transaction, and transaction detail. Relationships include one-to-many between user and cart, user and transactions, and user and address. Product relates one-to-many with cart and transactions, while cart represents a many-to-many relationship with products. Transactions relate many-to-one with users and one-to-many with transaction details. These relationships support integration between product data and shipping address data. The ERD Fig 6, illustrates entity relationships along with their attributes and data types.

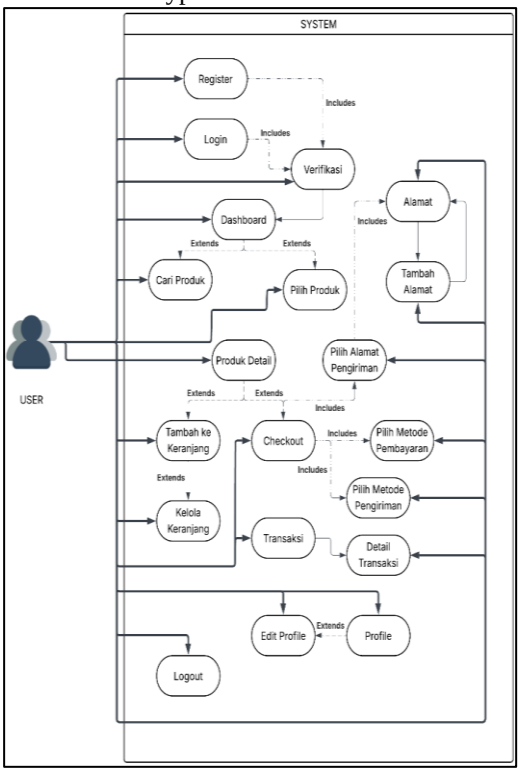

Figure 3. Usecase Diagram

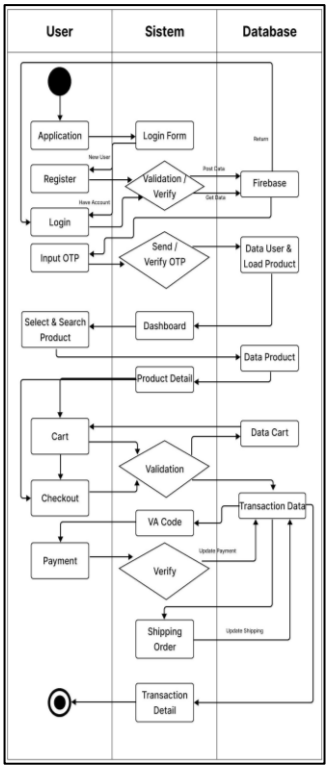

Figure 4. Activity Diagram

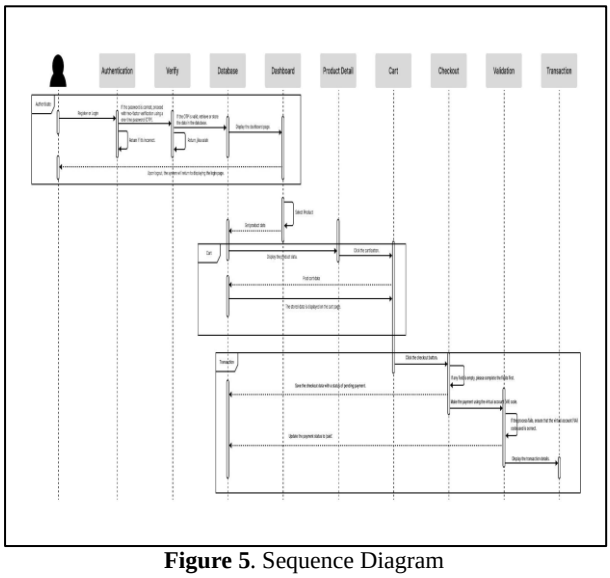

Figure 5. Sequence Diagram

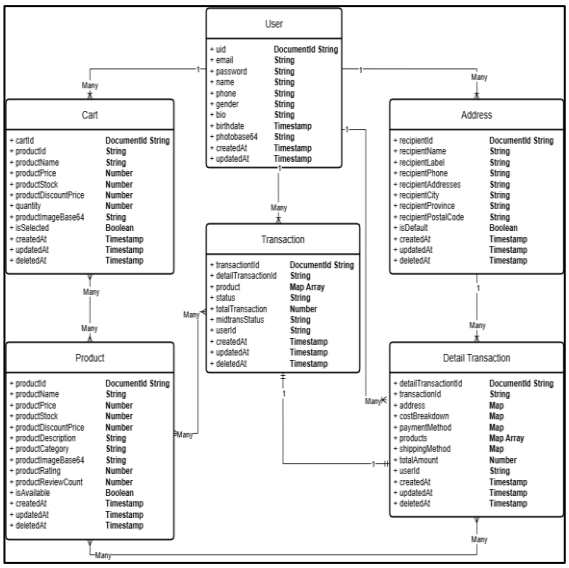

Figure 6. Class Diagram dan Entity Relationship Diagram

Software Development Architecture

In Android application development, several architectures can be used, including Model-View-Controller (MVC), Model-View-Presenter (MVP), and Model-View-ViewModel (MVVM). Both MVP and MVVM offer advantages over the MVC architecture [18]. In this study, the Berlian Tech online store application was developed using the MVVM architecture. MVVM was chosen due to its superior CPU efficiency, with an

average usage of 8.92% compared to 11.15% for MVP. In terms of memory usage, MVVM is also slightly more efficient, averaging 121.48 MB compared to 121.55 MB for MVP [19].

As shown in Fig 7, MVVM separates responsibilities among components to ensure that code remains structured and maintainable. The View represents the UI that interacts with users. The ViewModel acts as an intermediary between the View, Model, and Services. The Model represents the data used in the application. The Service layer serves as a communication bridge between Flutter and the backend via an API gateway, responsible for invoking APIs through HTTP, managing Firebase integration, and forwarding data to the ViewModel.

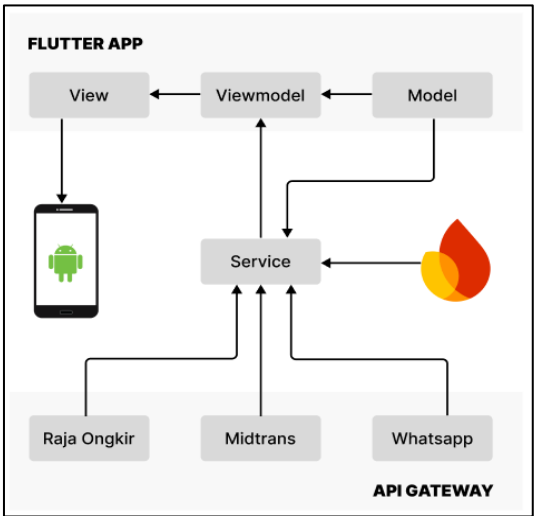

Figure 7. Software Development Architecture

Database Design

This study employs a document-based NoSQL database, as illustrated in Fig 8, 9. NoSQL databases play a crucial role in managing modern data-intensive applications that handle large volumes of data [20]. NoSQL databases enable the storage of big data and offer strong query performance, making them highly suitable when data structures do not align with traditional relational database schemas [21]. One of the NoSQL databases used in this study is Firebase. Firebase was selected because it simplifies data integration from end users to a cloud-hosted database, as its services are part of the Google Cloud Platform [22].

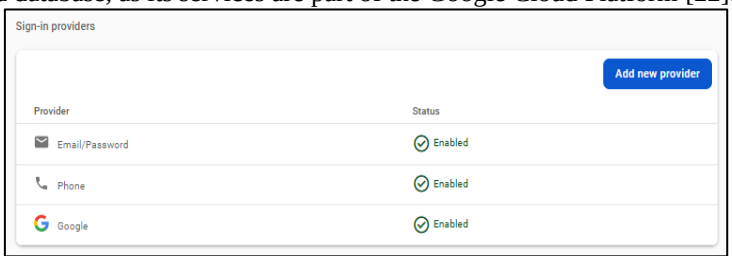

Figure 8. Sign-in Provider

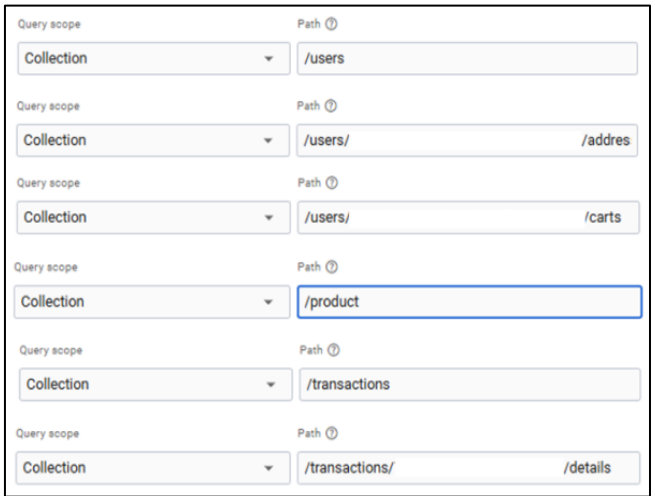

Figure 9. Collection users, address, carts, product, transactions, detail transactions

Application Testing

The application testing in this study uses black-box testing. Black-box testing focuses on evaluating software's functional specifications [23]. The testing process ensures that the application meets the functional requirements. The black-box testing method assesses system functionality without requiring knowledge of the system's internal structure [24].

Results and Discussion

The online store application was developed using Flutter, Firebase, and integrated with API gateways for automated payments, shipping, and WhatsApp notifications. Flutter enables developers to build multi-platform applications using a single codebase [25]. Firebase simplifies many common development challenges, allowing developers to focus on business logic and user experience [26]. As a document-based NoSQL database, Firebase is suitable for data-intensive modern applications [27]. The proposed system architecture uses the MVVM pattern, separating GUI development from business logic and backend processes. This approach supports parallel development, automated unit testing, code reusability, and ease of maintenance [28]. The application is optimized by integrating a payment gateway API to enhance the efficiency and security of payment processing [29]. Firebase's real-time data capabilities and simplified backend processes also support mobile app development without requiring server-side coding [30].

User Interface Implementation

The Berlian Tech Online Store application consists of several main screens, including the login, register, dashboard, product detail, cart, checkout, and transaction pages. As shown in Fig 10, 11, the login and register pages are used for authentication. Fig 12, shows the dashboard displaying available products. Fig 13, presents detailed product information. Fig 14, displays the cart, allowing users to store products and proceed with checkout using different items. After successfully completing the checkout process, Fig 15 shows the virtual account (VA) code for payment. If the payment is successful, the system displays the transaction page in Fig 16, and users can view transaction details as shown in Fig 17.

Figure 10. Login

Figure 11. Register

Figure 12. Homepage

Figure 13. Product Detail

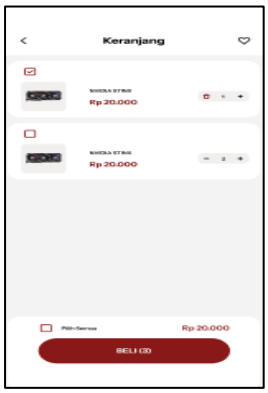

Figure 14. Cart

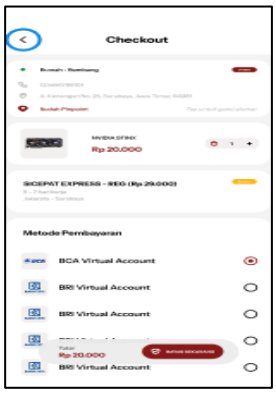

Figure 15. Checkout

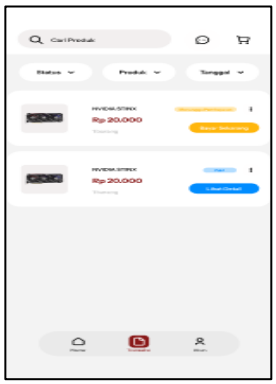

Figure 16. Transaction List

Figure 17. Transaction Detail

Database Implementation

The Firebase services used in this study include Firestore Database, Firestore Authentication, and Firestore Cloud. The Berlian Tech Online Store application uses the following collections: users, product, address, transactions, detail transactions, and carts Fig 18, 19, 20, 21, 22, 23.

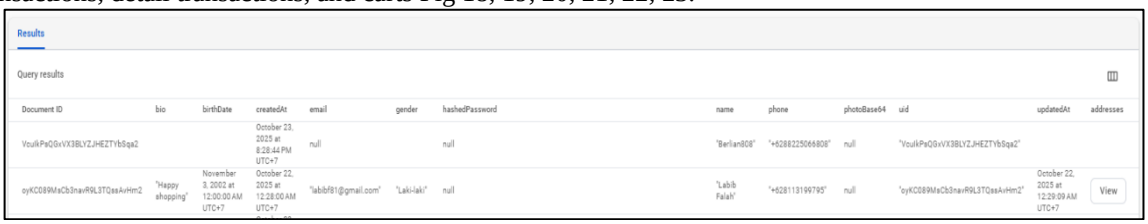

Figure 18. Collection Users

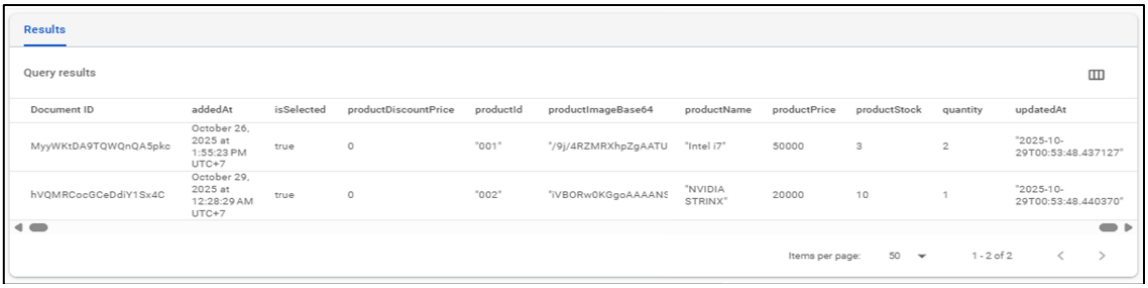

Figure 19. Collection Cart

Figure 20. Collection Address

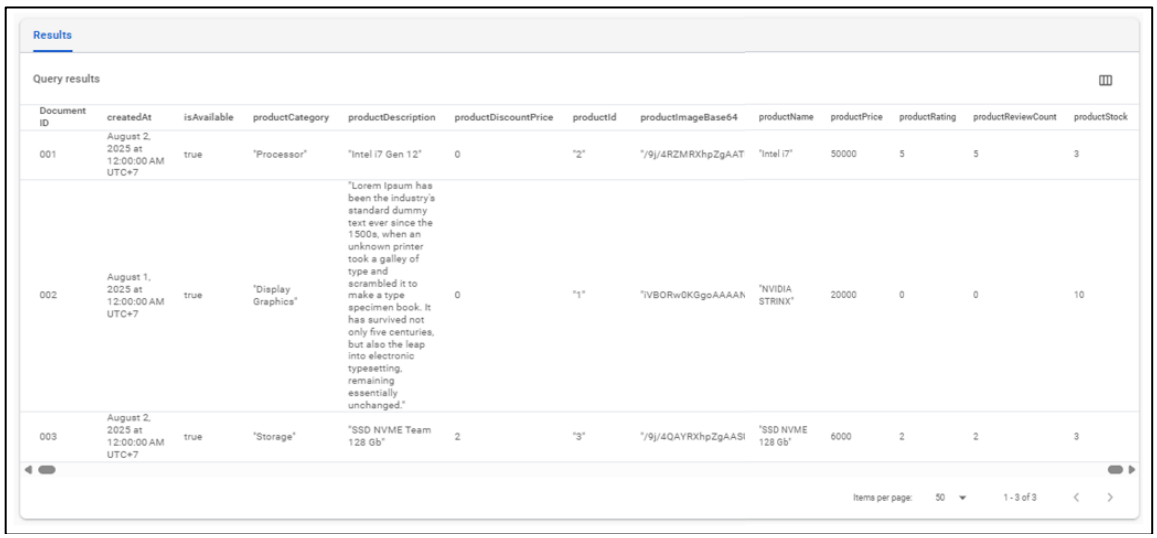

Figure 21. Collection Product

Figure 22. Collection Transaction

Figure 23. Collection Detail Transaction

System Testing

After completing the implementation stages of the user interface and database, the next step is system testing, which aims to ensure that each code function and application feature operates in accordance with the defined functional requirements. The system was evaluated using the black-box testing method, as presented in Table 4.

Table 4. System Testing

Feature	Test Scenario	Input	Expected Result	Testing
Authentication				
Register	Input Email & Password	Email: user@test.com Password: Test@1234	Successfully registers.	Success
Login	Input Email & Password	Email: user@test.com Password: Test@1234	Successfully login.	Success
Logout	Logout from the application	Click the logout button.	Successfully logs out and does not return to the dashboard page if the user has not logged in.	Success
Shop				
Home Page	Login	Account input.	Successfully displays available products.	Success
Search	Search Product	Enter product keyword.	Successfully displays products based on the keyword.	Success
Product Details	Select Product	Click on the product area.	Successfully displays product information.	Success
Keranjang				
Cart	Add, Remove, and Decrease Products in the Cart	Click the cart icon on the product detail page, then click the add, decrease, or remove icons on the cart page.	Successfully adds, removes, and decreases products in the cart.	Success
Transaksi				
Checkout	Shipping Address	Select a shipping address.	Successfully adds the shipping address.	Success
Checkout	Delivery Method	Select a delivery service.	Successfully adds the delivery service.	Success
Checkout	Payment Method	Select a payment method.	Successfully completes the payment and displays the VA (Virtual Account) code.	Success
Checkout	Display Transaction Details	Click the view details button.	Successfully displays the transaction details.	Success

Conclusion

This study produced an Online Store application implemented for Berlian Tech. The Berlian Tech Online Store application was designed using the Model-View-ViewModel (MVVM) system architecture combined with a service layer that facilitates communication between the API gateway and the view. The integration of Flutter and Firebase enables real-

time user experience, high security, and efficient cross-platform development. Additionally, Firebase provides high flexibility in data storage through its NoSQL structure. Results from black-box testing demonstrate that all core system features, including user authentication, product management, shopping cart, checkout, and transaction processes, function properly and in accordance with user requirements. The implementation of the Berlian Tech Online Store application significantly expands market reach, enhances promotional effectiveness and improves customer service. This digital transformation represents a strategic step in addressing business competition in the digital economy era. This study has limitations and requires further research in terms of UI and UX improvements, customer reviews, and user-based application testing for the Berlian Tech Online Store application.

References

[1] Y. D. R. A. Moro Sundjaja, A. Velasco Tatuil, D. V. Scholus, "The Determinant Factors of Shopping Cart Abandonment Among E-commerce Customers in Indonesia," 2024.

[2] Badan Pusat Statistik, "Data BPS," *BPS Stat.*, vol. 6, 2025.

[3] and N. W. D. N. Ketut Dewi Ari Jayanti, E. Triandini, G. Sastrawangsa, "Mobile Application Characteristics and User Perspective in Smart Healthcare Service Applications," 2022.

[4] R. Ripai, "Jurnal Informatika dan Rekayasa Perangkat Lunak Implementasi dan Perancangan Sistem Informasi Penjualan Vapestore Berbasis Mobile Flutter," vol. 6, no. 2, pp. 433–441, 2024.

[5] O. O. O. F. M. Dahunsi, A. J. Joseph, O. A. Sarumi, "Database Management System for Mobile Crowdsourcing Applications," *Int. J. Adv. Res. Comput. Commun. Eng.*, 2024.

[6] M. Ibra Alfathar, "Penerapan Mvvm (Model-View-Viewmodel) Pada Pengembangan Aplikasi Bank Sampah Digital," *J. Ris. dan Apl. Mhs. Inform.*, vol. 5, 2025.

[7] Y. D. P. I. P. A. E. Pratama, P. V. Andreyana, "Smart Expo UMKM Based on Extreme Programming Method: Evaluating on Black Box and UAT," *Int. J. Adv. Data Inf. Syst.*, vol. 5, no. 2, 2024, doi: 10.59395/ijadis.v5i2.1344.

[8] Sugiyono, "Metode Penelitian Kuantitatif," 2017.

[9] I. R. I. A. R. Hanafi, Y. Findawati, "Aplikasi Sistem Informasi Pelayanan Jasa Laundry Berbasis Website Pada Blue Laundry," *JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.*, vol. 9, no. 2, pp. 829–840, 2024.

[10] D. S. N. W. J. K. Dewi, I. G. M. Y. Antara, "Application of The Waterfall Method to the Website-Based JM Leather & Shoes Point of Sales Information System," *TIERS Inf. Technol. J.*, vol. 5, no. 1, pp. 1–12, 2024, doi: 10.38043/tiers.v5i1.5116.

[11] H. H. M. R. Alifi, T. Semiawan, D. C. U. Lieharyani, "Pemodelan Data Relasional pada NoSQL Berorientasi Dokumen," 2022.

[12] S. H. W. M. Ro'if, T. Afirianto, "Pengembangan Sistem Informasi Praktik Kerja Lapangan (PKL) Siswa Berbasis Website Menggunakan Metode Extreme Programming (Studi Kasus: SMK Negeri 1 Sumenep)," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 11, no. 1, pp. 1–10, 2024, doi: 10.25126/jtiik.20241116452.

[13] D. P. S. . O. Sudana, I. W. W. Ivan M.J, "Implementation Of Tree Model In The Development Of E-Mantram Android Application," *Lontar Komput. J. Ilm. Teknol. Inf.*, vol. 13, no. 2, 2022, doi: 10.24843/lkjiti.2022.v13.i02.p05.

[14] A. S. N. F. Rahmasari, A. Nursyahid, T. Agung Setyawan, "Analisis Kinerja Aplikasi Pemantauan dan Pengendalian Smart Agriculture Berbasis Android." [Online]. Available: www.omahiot.net

[15] N. C. K. A. P. Wibawa, M. Diantoro, I. Idris, A. Purnomo, "Pengembangan Sistem Informasi Pengelolaan Konferensi Internasional Universitas Negeri Malang dengan Menggunakan Metode Waterfall," *J. Teknol. Sist. Inf. Dan Apl.*, vol. 7, no. 1, pp. 352–361, 2024, doi: 10.32493/jtsi.v7i1.34974.

[16] J. A. Hindarto S. Suprihadi, "Perancangan Sistem Informasi Penjualan Pada Toko Roti Di Kota Cikarang Berbasis Web Menggunakan Framework Laravel," *JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.*, vol. 9, no. 1, pp. 53–66, 2024.

[17] J. A. J. K. N. Nursobah, M. I. Saad, "Implementation of the Flutter Framework for Developing an E-Commerce Application," *TEPIAN*, vol. 5, no. 4, pp. 127–135, 2024, doi: 10.51967/tepian.v5i4.3110.

[18] A. R. A. T. Sondha, U. Sa'adah, F. F. Hardiansyah, M. Bagus, "Framework dan Code Generator Pengembangan Aplikasi Android dengan Menerapkan Prinsip Clean Architecture (Framework and Code Generator for Android Development with Clean Architecture Principles Implementation)," 2020.

[19] N. I. I. Fajar Pradana, Raziqa Izza Langundi, Djoko Pramono, "Comparative Analysis of MVVM and MVP Patterns Performance on Android Dashboard System," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 14, no. 2, pp. 87–95, 2025, doi: 10.22146/jnteti.v14i2.18985.

[20] V. T. N. Sharma, R. N. Tripathi, "FreeFlow: A framework for server-driven mobile apps," *Sci. Talks*, vol. 14, no. 7, 2025, doi: 10.1016/j.sctalk.2025.100445.

[21] J. B. I. Carvalho, F. Sá, "Performance Evaluation of NoSQL Document Databases: Couchbase, CouchDB, and MongoDB," *Algorithms*, vol. 16, no. 2, 2023, doi: 10.3390/a16020078.

[22] A. Ambarwati and A. P. Habibi, "Analisis Maturity Level Business Goals 8 Menggunakan COBIT Pada PT. APLIN," *INTENSIF J. Ilm. Penelit. dan Penerapan Teknol. Sist. Inf.*, vol. 1, no. 2, pp. 137–146, 2017, doi: 10.29407/intensif.v1i2.846.

[23] R. L. R. K. I. M. Dewi, I. W. G. Narayana, "Application of Certification Management Information Systems at LSP Engineering Hospitality Indonesia," *APTISI Trans. Technopreneurship*, vol. 5, no. 3, pp. 227–239, 2023, doi: 10.34306/att.v5i3.317.

[24] M. D. P. A. M. Zen, Irwan, Hafni, "Implementasi dan Pengujian Menggunakan Metode BlackBox Testing Pada Sistem Informasi Tracer Study," *Bull. Comput. Sci. Res.*, vol. 4, no. 4, pp. 327–340, 2024, doi: 10.47065/bulletincsr.v4i4.359.

[25] H. H. S. K. S. A. Kinari, N. Funabiki, S. T. Aung, "A Guided Self-Study Platform of Integrating Documentation, Code, Visual Output, and Exercise for Flutter Cross-Platform Mobile Programming," *Computers*, vol. 14, no. 10, p. 417, 2025, doi: 10.3390/computers14100417.

[26] C. M. R. Kesavan, D. Gay, D. Thevessen, J. Shah, "Firestore: The NoSQL Serverless Database for the Application Developer," *Proc. - Int. Conf. Data Eng.*, vol. 3, no. 1, pp. 3376–3388, 2023, doi: 10.1109/ICDE55515.2023.00259.

[27] N. Tripathi, "NoSQL database education: A review of models, tools and teaching methods," *Elsevier Inc*, vol. 8, no. 1, 2025, doi: 10.1016/j.jss.2025.112391.

[28] W. Sheikh and N. Sheikh, "A Model-View-ViewModel (MVVM) Application Framework For Hearing Impairment Diagnosis A Model-View-Viewmodel (MVVM) Application Framework For Hearing Impairment Diagnosis A Preprint," 2019, doi: 10.48550/arXiv.1911.08289.

[29] F. A. Rahma and S. Samsudin, "Android-Based Sports Infrastructure E-Booking Application at Provincial Youth and Sports Office Using Waterfall Method," *J. Comput. Networks, Archit. High Perform. Comput.*, vol. 6, no. 4, pp. 1769–1780, 2024, doi: 10.47709/cnahpc.v6i4.4804.

[30] I. A. C. M. Ohyver, J. V. Moniaga, I. Sungkawa, B. E. Subagyo, "The comparison of Firebase Realtime Database and MySQL Database performance using the Wilcoxon Signed-Rank Test," *Procedia Comput. Sci.*, vol. 5, no. 1, pp. 396–405, 2019, doi: 10.1016/j.procs.2019.08.231.