An Extensible Model-View-Controller (MVC)-Based Cross-Platform Framework for Next-Generation AI-Enabled Email Applications

Muhammad Hammadullah Arain, Madeha Memon, Zartasha Baloch, Ali Nafees Siddiqui, Uzaif Talpur
Department of Computer Systems Engineering, Mehran University of Engineering and Technology
*Correspondence: hammadullaharain25@gmail.com

Citation | Arain. M. H, Memon. M, Baloch. Z, Siddiqui. A. N, Talpur. U, "An Extensible Model-View-Controller (MVC)-Based Cross-Platform Framework for Next-Generation AI-Enabled Email Applications", IJIST, Vol. 8 Issue. 3 pp 1397-1407, June 2026
Received | May 08, 2026 **Revised |** June 10, 2026 **Accepted |** June 14, 2026 **Published |** June 23, 2026.

Cross-platform application development is becoming increasingly important because of the fast-growing trend towards developing heterogeneous computing systems such as Android, iOS, and desktops. The traditional approach of native development requires maintaining separate codebases, thereby increasing development costs and time to market. This paper details the development and design of a cross-platform email application termed 1TapMail, which utilizes the Flutter framework with Firebase services acting as its back-end infrastructure. The application utilizes one codebase that guarantees consistency of operations and adequate performance on various computing devices while ensuring high user convenience. This is achieved by implementing basic email functionality such as authentication, sending, receiving, and management of emails while also offering the facility of data storage. Experimental evaluation included functional, usability, and platform compatibility testing. The proposed framework achieved a 100% functional success rate by passing all 13 predefined test cases and demonstrated successful deployment across android, iOS, and desktop platforms using a single codebase. The results indicate that the proposed framework provides a reliable and practical alternative to conventional native development approaches.

Keywords: Flutter, Firebase, Cross-Platform Applications, Email Systems, Mobile Computing

Introduction:

Although there have been various advancements in terms of technology over the past few years, electronic mail (email) still remains one of the most popular modes of digital communication [1]. However, although there are various email clients available, most of them are designed natively, thus making their implementation quite costly and complex in nature [2][3][4].

Cross-platform development frameworks try to solve these issues by allowing the development of applications that can run across multiple platforms, thus eliminating the need for developing and maintaining individual platforms. One of the leading cross-platform development frameworks, which has gained popularity because of its speed, versatility, and native experience, is the UI toolkit known as Flutter, created by Google Inc. In combination with backend services offered by the Firebase platform, Flutter allows fast prototyping of applications [5][6][7][8][9][10].

The purpose of this paper is to design and develop a cross-platform email application using the Flutter framework and Firebase services.

But the research conducted on email systems to date has mainly concentrated on improving performance from the server side, whereas little work has been done on the cross-platform email client development and the effects of doing so on development effort and deployment to multiple platforms [11][12][13][14][1][15]. Furthermore, many of the current cross-platform development tools have not been tested thoroughly for use in a complete communication system like email clients.

Novelty of the Study:

Existing research has primarily focused on server-side email architectures, email protocols, or general cross-platform application development, with limited attention given to complete cross-platform email applications. The novelty of this study lies in the development of an extensible Model-View-Controller (MVC)-based cross-platform framework that integrates Flutter and Firebase to provide unified architecture for user authentication, email management, real-time synchronization, and multi-platform deployment through a single codebase. Unlike previous studies that mainly emphasize backend infrastructure or generic mobile applications, the proposed framework demonstrates a complete implementation of an email application and validates its functionality, usability, and platform compatibility across Android, iOS, and desktop environments. The proposed framework also provides a maintainable and extensible foundation for future research and the development of AI-enabled cross-platform email applications and advanced intelligent communication services [14][16][17][18][19].

Research Objectives:

The primary objective of this study is to design and implement an extensible Model-View-Controller (MVC)-based cross-platform framework for email applications. Specifically, this study aims to:

Design a cross-platform email application using Flutter and Firebase based on the Model-View-Controller (MVC) architecture.

Implement core email functionalities, including user authentication, email composition, synchronization, and mailbox management through a single codebase.

Evaluate the functionality, usability, and platform compatibility of the proposed framework across Android, iOS, and desktop environments.

Analyze the effectiveness of the proposed framework in reducing development complexity and improving maintainability compared with conventional native development approaches.

Related Work:

Much research has explored email protocols and their infrastructures in depth especially SMTP and massive email server architectures [1][15][11][12]. They discuss various aspects like message delivery processes, behaviors, and traffic patterns that help enhance their back-end performance. However, they don't consider the problems faced during cross-platform client design and simplifying the applications' back-end development process.

Modern email services such as Gmail, Yahoo Mail, and MS Outlook include advanced features and are highly scalable and reliable. This is because of the massive underlying infrastructures and platform-specific implementations. Thus, each operating system requires its own implementation and maintenance effort, leading to increased cost and development time.

Mobile app development across multiple platforms is a possible solution to these problems. There have been several attempts at using frameworks like React Native and Xamarin in the past for multi-platform implementation. The React Native framework relies on a JavaScript bridge to communicate with native components, which could result in a performance penalty in UI-centric apps. Using Xamarin, code can be shared due to .NET but user interfaces need to be adapted per platform [3][4].

In contrast to the above-mentioned frameworks, Flutter translates its source code directly to native code, while also rendering UI via the Skia graphics library [5][6][10]. This results in consistent behavior on different platforms and decreases the reliance on platform-specific components. It has been observed that Flutter offers comparable performance along with increased development efficiency in numerous applications [6].

Nevertheless, most of the studies conducted to date have concentrated either on back-end mail server implementation or on developing generic mobile applications. Only a few studies have considered the implementation of modern cross-platform frameworks for communication applications such as mail client applications which require synchronization and authentication features.

Recent studies have demonstrated the effectiveness of Flutter and Firebase in developing cross-platform applications across various domains, including live streaming, mobile learning, context-aware systems, sustainable web platforms, digital marketplace solutions, and healthcare information systems [5][7][20][21][22][23][8][9][10]. Similarly, comparative studies have shown that Flutter provides competitive performance, improved user interface consistency, and enhanced development efficiency compared with other cross-platform frameworks such as React Native and Xamarin [3][4][6]. However, these studies primarily focus on framework performance or application development in general and do not investigate complete cross-platform email applications requiring secure authentication, real-time synchronization, and comprehensive email management.

Furthermore, recent research on email systems has mainly concentrated on security, email delivery, phishing detection, and AI-assisted email management [11][12][13][14][18][19][1][15]. Although these studies contribute significantly to improving email technologies, they do not address the architectural design and implementation of a complete Model-View-Controller (MVC)-based cross-platform email application. Therefore, a research gap still exists in developing an extensible framework that integrates modern cross-platform technologies with unified software architecture for email applications. This study addresses this gap by proposing and evaluating an extensible MVC-based cross-platform email framework using Flutter and Firebase, with validation through functional, usability, and platform compatibility evaluations.

System Architecture and Methodology:

The proposed framework follows the Model-View-Controller (MVC) design pattern to separate the user interface, application logic, and data management components, ensuring

modularity and scalability. Figure 1 illustrates the overall research workflow adopted in this study. The methodology consists of sequential stages, including problem identification, literature review, system design, implementation, testing, and evaluation.

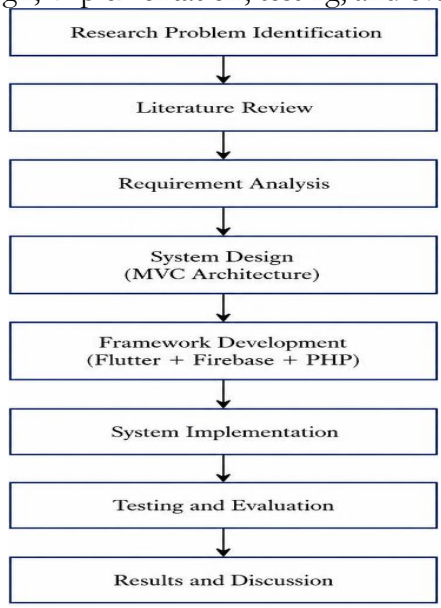

Figure 1. Research workflow adopted in this study.

Overall System Architecture:

The architecture of it consists of three primary layers: the presentation layer, the application logic layer, and the data management layer.

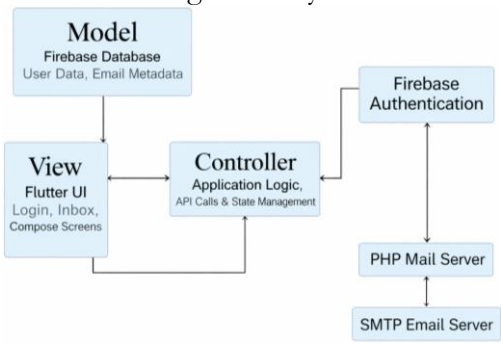

Figure 2. MVC-based system architecture of the proposed cross-platform email application.

The process of operation within this system starts with user authentication using Firebase Authentication services. Once the authentication is successful, the user gains access to the inbox and emails are pulled from the cloud database. When the user composes an email, the controller parses the email and sends the content to the mail service located at the server side using an HTTP request. The server handles message sending, while updating the metadata in real time on Firebase Database and making it accessible to other connected clients.

Model-View-Controller Design:

The MVC model consists of the Model layer that handles all the data used in the program, like usernames, passwords, and email bodies; the View layer, which is responsible for handling the user interface, including the login page, inbox, and email composition interface; and the Controller layer that handles all the logic behind the application.

Backend Integration and Data Flow:

Firebase is utilized as a cloud backend for providing security features such as authentication and storing data in real time so that mailbox contents can be automatically synchronized across multiple devices. The email sending process takes place using server-side scripting through PHP installed on the hosted domain.

Design Objectives:

The primary design objectives of the proposed framework were portability, scalability, maintainability, and consistent user experience across multiple platforms. Several cross-platform frameworks, including React Native, Xamarin, .NET MAUI, and Kotlin Multiplatform, were considered. React Native relies on a JavaScript bridge that may introduce performance overhead, while Xamarin and .NET MAUI require the .NET ecosystem and may increase maintenance complexity. Kotlin Multiplatform supports code sharing but still requires platform-specific user interface development. Flutter was selected because it compiles directly to native code, provides a consistent user interface, and supports android, iOS, and desktop platforms from a single codebase. Firebase was chosen for its integrated authentication, real-time database, and cloud services, reducing backend development effort while supporting scalability and real-time synchronization [24].

Experimental Setup:

The proposed framework was developed using Flutter, Dart, Firebase, and PHP on a Windows 11 system equipped with an Intel Core i5 (11th Generation) processor and 8 GB RAM. The application was developed using Flutter SDK, Android Studio, and Visual Studio Code, and evaluated through functional testing, platform compatibility testing, and usability evaluation across Android, iOS, and desktop platforms.

Technologies used:

Flutter and Dart:

Flutter acts as the main framework used to develop the client-side app on different platforms such as android, iOS, and desktop applications using a single codebase. The widget-driven structure makes it easy to create responsive user interfaces regardless of the screen size or OS used. The Dart programming language is utilized to build app logic, state management, and controllers. The use of Flutter significantly simplifies the development process without compromising performance.

Firebase:

Firebase serves as the cloud-based backend used for providing authentication and real-time database functionality. Firebase Authentication ensures that there is secure access to the application, while the real-time NoSQL database contains the data associated with emails and synchronization of mailboxes across devices. This makes it possible for the application users to get updates of their messages regardless of what platform they use, without having to refresh the information manually.

Server-Side Technologies:

Server side sending of emails is done using scripts that are coded in PHP. This involves the client software sending the email content information to the server in an encrypted form of HTTP communication, after which the server processes the message and then sends it to the destination mail server.

Implementation:

The system that has been developed includes essential email functionalities such as email registration, login process, and email inbox, email composition, managing drafts, sending messages, and deleting messages. This is done using Flutter, which ensures responsive and uniform performance on both Android and iOS devices.

Functional Implementation:

Firebase authentication is used to provide security for user authentication, whereas the real-time database is used for storing user data and metadata of the email. In case the user sends any command like composing or sending an email, the controller handles this command and sends the email to the server in a secure manner using HTTP. The PHP script on the server side will forward this message to the destination mail server. Once done successfully, the new email metadata is stored in Firebase.

Figure 3 illustrates the workflow of user interaction and email operations in the proposed system, including authentication, inbox access, email composition, and message delivery.

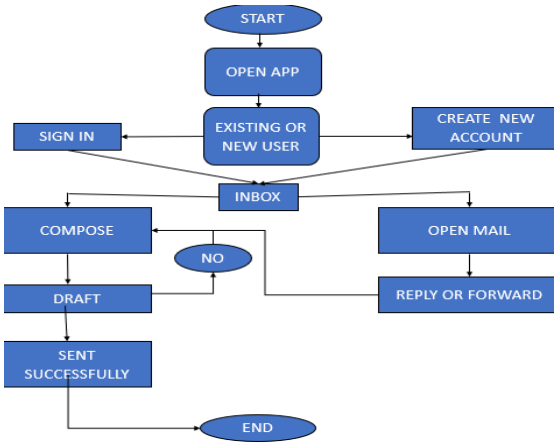

Figure 3. Workflow of user interaction and email operations in the proposed system.

An example use case starts with the user logging into the system and being authenticated. Once authenticated, the inbox displays all synchronized emails stored in the cloud database. The user can also create an email and send it; once the email is sent, it appears in the sent folder. The recipient gets the email through the server-side infrastructure, and the synchronization of mailboxes for the sender and receiver is sustained on all their devices.

User Interface Implementation:

The graphical user interface of the proposed 1TapMail application was developed using Flutter widgets to ensure consistency and usability across multiple platforms. Figures 4-7 present the main interfaces implemented in the system.

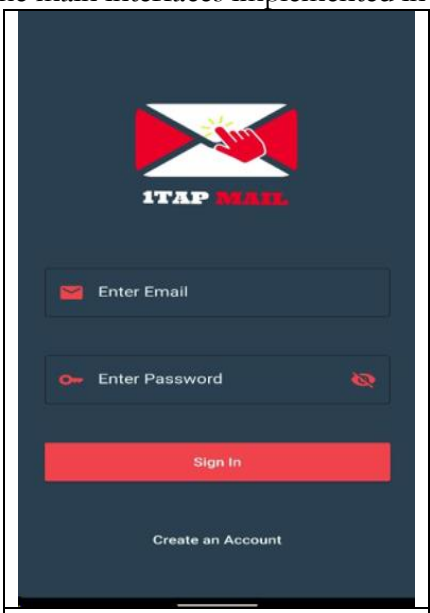

Figure 4. Login Screen of 1TapMail Application

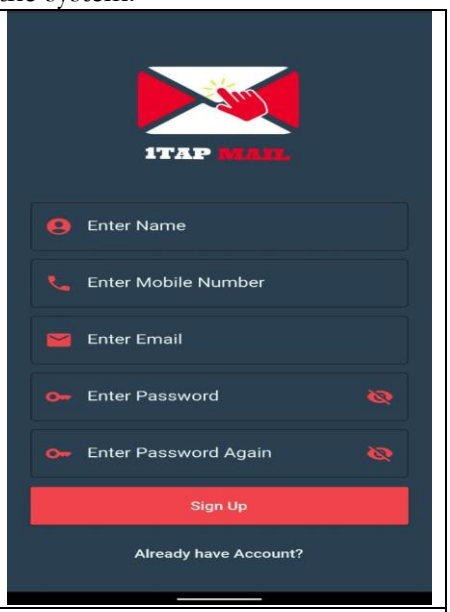

Figure 5. Registration Screen of 1TapMail Application

The login interface of the proposed 1TapMail application is shown in Fig. 4. This screen allows registered users to access their accounts by entering their email address and password. User authentication is performed using Firebase Authentication services, ensuring secure access to the application. The interface was developed using Flutter widgets to provide consistent user experience across Android, iOS, and desktop platforms.

Figure 5 illustrates the registration interface of the proposed 1TapMail application. This interface enables new users to create an account by providing their personal information, email address, and password. Input validation mechanisms are implemented to ensure data integrity and prevent invalid account creation. The registration process is integrated with Firebase Authentication for secure user management.

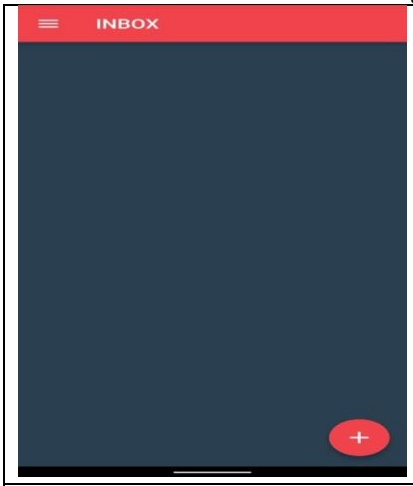

Figure 6. Inbox Screen of 1TapMail Application

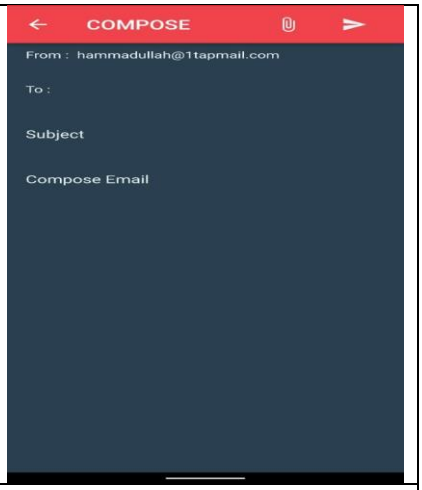

Figure 7. Compose Screen of 1TapMail Application

Figure 6 shows the inbox interface of the proposed application. The inbox displays emails synchronized with the Firebase database in real time. Users can view, open, and manage their emails through a responsive and user-friendly interface.

Figure 7 presents the compose email interface of the proposed application. This screen allows users to create new email messages by specifying recipient details, subject, and message content. Upon submission, the email data is securely transmitted to the server using HTTP requests, where PHP-based services process and deliver the message to the intended recipient.

Security Considerations:

Security is an essential element of any email system. The proposed application utilizes Firebase Authentication to ensure that only authenticated users can access the system. User authentication is performed using secure authentication tokens, while encrypted client-server communication helps protect data during transmission. These mechanisms provide a secure foundation for the proposed framework. However, comprehensive security assessments, including vulnerability analysis, penetration testing, threast modeling, and security performance evaluation, were beyond the scope of this study and are recommended for future research.

Experimental Evaluation:

The proposed system was evaluated thoroughly with respect to its functional, integrative, and usability aspects in order to confirm that it is correct, reliable, and effective. The tests were performed on several platforms to make sure that the same results could be obtained in different settings. The main objective of the evaluation process was to verify the basic functionalities of the e-mail system.

Functional Testing:

The testing done was functional testing, where the aim was to check whether all the functionality within the application met the requirements set for it. Test cases were created, and they focused on key modules such as authentication, inbox, email creation, draft, and sending messages. All test cases were completed successfully without major failures.

The functional test results for the proposed system are presented in Table 1.

Table 1. Summary of Functional Test Results

Test Category	No. of Test Cases	Passed	Failed
Authentication	2	2	0
Inbox Operations	5	5	0
Compose & send	3	3	0
Draft & Trash	3	3	0
Total	13	13	0

The results indicate that all functional requirements were satisfied, demonstrating the correctness and stability of the application during normal operation.

Platform Compatibility Testing:

In order to prove that the suggested solution can be used across multiple platforms, an evaluation of the application was conducted in Android, iOS, and desktop platforms using one single code base. The evaluation of platform compatibility is shown in Table 2**.**

Table 2. Platform Compatibility and Feature Support

Platform	Supported	Verified Features
Android	Yes	Login, Inbox, Send/Receive
iOS	Yes	Authentication, Compose
Desktop	Yes	Inbox Access, Email Synchronization

The results confirm that the application provides consistent functionality across all supported platforms using a single codebase, validating the effectiveness of the chosen cross-platform development approach.

Usability Evaluation:

A usability evaluation was conducted to assess the ease of use and effectiveness of the proposed application. Users performed common tasks, including user login, inbox access, email composition, and message sending. The evaluation indicated that these tasks could be completed with minimal guidance, suggesting that the application provides a simple and user-friendly interface for everyday email communication.

Performance Observations:

The application proved to be highly responsive on all compatible operating systems throughout the testing process. Basic functions like signing in, loading emails from the inbox, and composing emails were performed instantly, without any lag, given normal network connectivity. Flutter ensured seamless rendering of the UI, whereas Firebase offered almost instantaneous data synchronization.

Although a comprehensive benchmarking test for comparing it with native application development was out of the scope of this research, based on observations, it can be concluded that the suggested system offers sufficient performance for conventional use cases involving emails.

Discussion:

The results demonstrate the effectiveness of cross-platform app development using Flutter. As opposed to native apps, the suggested approach reduces the costs of development and support by providing a single codebase on several platforms. Although the system performance depends on the optimization of the framework used, the overhead experienced in normal operation has been very small and satisfactory for regular mail service. Firebase facilitates more efficient backend implementation; nevertheless, the approach requires relying on cloud services. Possible future development might focus on hybrid or hosted backends.

Though proven to be effective, the system still comes with some weaknesses. First, since the program uses cloud-based services such as Firebase, there must be a steady internet connection. This may prove to be challenging when the program is used in areas with limited or unreliable connectivity. Second, the testing process largely revolves around functional tests

and usability test and does not focus much on quantitative benchmarking using native email applications.

A comparative analysis of development approaches is summarized in Table 3, highlighting the differences between conventional native development and the proposed cross-platform solution.

Table 3. Comparison between Cross-Platform and Native Development Approaches

Aspect	Native Development	Proposed System (Flutter & Firebase)
Codebase	Multiple	Single
Development Cost	High	Reduced
Maintenance Complexity	Higher	Lower
Time-to-Market	Longer	Shorter
Platform Coverage	Platform-specific	Android, iOS, Desktop
Backend Integration	Custom per platform	Simplified via Firebase

The improved development efficiency of the proposed framework is achieved through a single codebase for Android, iOS, and desktop platforms, reducing duplicate development and maintenance efforts. Furthermore, Firebase simplifies backend implementation by providing integrated authentication, real-time database, and cloud services, resulting in faster application development compared with conventional native approaches.

Implications of the Study:

The findings of this study have practical, industrial, academic, and technological implications. Practically, the proposed framework provides developers with a maintainable approach for building cross-platform email applications using a single codebase. From an industrial perspective, it can help reduce development time, maintenance effort, and deployment costs across multiple platforms. Academically, the study contributes to research on cross-platform software engineering by demonstrating the application of the Model-View-Controller (MVC) architecture with Flutter and Firebase. Technologically, the proposed framework provides a scalable foundation for the future development of intelligent and AI-enabled cross-platform email applications.

Recommendations:

Based on the findings of this study, practitioners are encouraged to adopt modular architectures such as the Model-View-Controller (MVC) pattern and cross-platform development frameworks to improve software maintainability and reduce development effort. Future developers should consider integrating advanced security mechanisms, comprehensive performance benchmarking, and AI-enabled features to further enhance the functionality, scalability, and reliability of cross-platform email applications.

Conclusion and Future Work:

This paper describes the implementation of a cross-platform email client app that has been developed using Flutter and Firebase technologies. This system demonstrates the feasibility of writing the code once and run it on Android, iOS, and desktop platforms without sacrificing performance or introducing significant maintenance challenges.

Experimental evaluation demonstrated that the app functions as needed and that the application provides a convenient and interactive environment for completing everyday messaging operations. The results indicate that a cross-platform application can be an effective substitute for a native one.

The results of this research show that the combination of Flutter with Firebase provides a robust technological solution for building communication applications within an environment of limited development capabilities.

The scope of future research can include more thorough quantitative performance testing compared to native applications, implementation of advanced security tools (such as

end-to-end encryption and multi-factor authentication), and incorporation of various other features available in enterprise apps (e.g., filtering and analytics).

References:

[1] Ruixuan Li, Shaodong Xiao, "Bounce in the Wild: A Deep Dive into Email Delivery Failures from a Large Email Service Provider," *Proc. ACM SIGCOMM Internet Meas. Conf. IMC*, pp. 659–673, 2024, [Online]. Available: https://dl.acm.org/doi/10.1145/3646547.3688425

[2] Osinachi Deborah Segun-Falade, Olajide Soji Osundare, Wagobera Edgar Kedi, Patrick Azuka Okeleke, Tochukwu Ignatius Ijomah, and Oluwatosin Yetunde Abdul-Azeez, "Developing crossplatform software applications to enhance compatibility across devices and systems," *Comput. Sci. IT Res. J.*, vol. 5, no. 8, pp. 2040–2061, Aug. 2024, doi: 10.51594/CSITRJ.V5I8.1491.

[3] F. Mushtaq, F. Azam, and M. W. Anwar, "Performance Comparison of Single Code Base Development Tools: Flutter, React Native, and Xamarin," *Proc. - 2024 14th Int. Conf. Softw. Technol. Eng. ICSTE 2024*, pp. 17–23, 2024, doi: 10.1109/ICSTE63875.2024.00011.

[4] D. Zou and M. Y. Darus, "A Comparative Analysis of Cross-Platform Mobile Development Frameworks," *2024 6th IEEE Symp. Comput. Informatics, Isc. 2024*, pp. 84–90, 2024, doi: 10.1109/ISCI62787.2024.10667693.

[5] M. J. Fauzi Hidayatul Anmi, "Implementation of Flutter and Firebase Technologies in Modern Live Streaming Application Development," *JOISIE (Journal Inf. Syst. Informatics Eng.*, vol. 9, no. 1, p. 246, 2025, doi: 10.35145/joisie.v9i1.4808.

[6] J. Nanavati, S. Patel, U. Patel, and A. Patel, "Critical Review and Fine-Tuning Performance of Flutter Applications," *Proc. - 2024 5th Int. Conf. Mob. Comput. Sustain. Informatics, ICMCSI 2024*, pp. 838–841, 2024, doi: 10.1109/ICMCSI61536.2024.00131.

[7] N. H. Herrera, R. Rivera, E. Gómez-Torres, and C. Challiol, "CAMS-F: Extending the Context-Aware Mobile System Framework for Cross-Platform Development with Flutter, Firebase, and Google Maps," pp. 566–576, 2026, doi: 10.1007/978-3-032-07995-4_37/SAVE-RESEARCH.

[8] K. U. Singh, N. Varshney, P. Gupta, G. Kumar, T. Singh, and S. R. Dogiwal, "Mobile Application Control with Firebase Cloud Messaging," *Lect. Notes Networks Syst.*, vol. 1020 LNNS, pp. 527–535, 2024, doi: 10.1007/978-981-97-3588-4_42/SAVE-RESEARCH.

[9] A. K. Das and S. Ghosh, "Real-Time Virtual Classroom Platform: Integrating React, Firebase, WebRTC, and Socket.io," *2025 8th Int. Conf. Electron. Mater. Eng. Nano-Technology, IEMENTech 2025*, 2025, doi: 10.1109/IEMENTECH65115.2025.10959621.

[10] Hanis Diansyah, Syafrinal Syafrinal, "Design and Development of a Mobile Application Using Android Studio and Flutter," *J. Mob. Technol.*, vol. 3, no. 2, pp. 69–76, 2025, doi: 10.59431/jms.v3i2.646.

[11] V. Kavitha, V. P. Harish, A. Ravindar, and G. P. Sivasaran, "Post-Quantum Email: A Practical Implementation of Lattice-Based Cryptography with Crystals-Dilithium for Advanced Email Encryption," *IFIP Adv. Inf. Commun. Technol.*, vol. 750 IFIPAICT, pp. 138–147, 2026, doi: 10.1007/978-3-031-98364-1_12/SAVE-RESEARCH.

[12] R. A. Aladraj, "Evaluation of Secure Email Awareness: A Case Study," *2nd Int. Conf. IT Innov. Knowl. Discov. ITIKD 2024*, 2025, doi: 10.1109/ITIKD63574.2025.11005031.

[13] Najwa Altwaijry, Isra Al-Turaiki, "Advancing Phishing Email Detection: A Comparative Study of Deep Learning Models," *Sensors*, vol. 24, no. 7, p. 2077, 2024, doi: https://doi.org/10.3390/s24072077.

[14] Sakthivel N, Somasundaram R.S, "SmartMail Hub: An AI-Powered Unified Interface for Cross-Platform Email Management," *Authorea Prepr.*, 2025, [Online]. Available: https://www.techrxiv.org/doi/pdf/10.36227/techrxiv.176127050.02840975/v1

[15] A. Rachad, L. Gaiz, K. Bouragba, and M. Ouzzif, "Sending and Receiving Secure Email Based on Blockchain," *Proc. - 10th Int. Conf. Ubiquitous Networking, UNet 2024*, 2024, doi: 10.1109/UNET62310.2024.10794735.

[16] Xiao Tan, Xifeng Tian, "Development of a Digital Intelligence Training System for Faculty in Smart Health and Elderly Care Services and Management Based on the MVC Pattern," *Proc. 2026 2nd Int. Conf. Digit. Educ. Inf. Technol. DEIT 2026*, 2026, [Online]. Available: https://dl.acm.org/doi/10.1145/3802607.3802624

[17] J. Kim, "MVCPVM: Model-View-Controller-Presenter-View Model - A Unified Architectural Pattern," *SSRN *, Mar. 2026, doi: 10.2139/SSRN.6376119.

[18] P. Aadhavan, D. Vasantha Kumar, R. Suganthini Sri, J. Ganapathy, and P. Ramachandran, "AI-Powered Phishing Detection in Email Forensics: A Machine Learning Approach for Cyber Threat," *2025 10th Int. Conf. Front. Signal Process. ICFSP 2025*, pp. 146–150, 2025, doi: 10.1109/ICFSP67350.2025.11353877.

[19] Linga Reddy Alva & Bishwajeet Pandey, "Agentic AI systems in the age of generative models: architectures, cloud scalability, and real-world applications," *Artif. Intell. Rev.*, vol. 59, no. 88, 2026, [Online]. Available: https://link.springer.com/article/10.1007/s10462-025-11458-6

[20] Y. W. Syaifudin, D. D. Yapenrui, Noprianto, N. Funabiki, I. Siradjuddin, and H. N. Chasanah, "Implementation of Self-Learning Topic for Developing Interactive Mobile Application in Flutter Programming Learning Assistance System," *2024 ASU Int. Conf. Emerg. Technol. Sustain. Intell. Syst. ICETSIS 2024*, pp. 1103–1107, 2024, doi: 10.1109/ICETSIS61505.2024.10459432.

[21] Madeha Memon, Irfan Ali Bhacho, Zartasha Baloch, Hanan Shaikh, & Touqeer Ali, "An Eco-Friendly Affordable Web-Based Fashion Sharing Platform Promoting Sustainability," *J. Comput. Biomed. Informatics*, vol. 9, no. 1, 2025, [Online]. Available: https://jcbi.org/index.php/Main/article/view/1022

[22] AmeerJan, Z. Baloch, M. Memon, A. Manzoor, and T. Ahmed, "ASAN MANDI: Digital Transformation of Pakistan's Fruit and Vegetable Market," *Int. J. Innov. Sci. Technol.*, vol. 7, no. 3, pp. 1453–1465, 2025, Accessed: Jul. 08, 2026. [Online]. Available: https://ideas.repec.org/a/abq/ijist1/v7y2025i3p1453-1465.html

[23] K. Dewal, B., Memon, madeha, Rathi, M., Memon, Y. and Fatima, "A Framework for Automatic Blood Group Identification and Notification Alert System," vol. 13, no. 2, pp. 74–84, 2023.

[24] A. M. Yasir *et al.*, "Comparative Performance Analysis of Firebase Realtime Database and Supabase Using Multi-Method Testing Approaches," pp. 303–308, Dec. 2025, doi: 10.1109/ICE3IS66769.2025.11281116.