International Journal of Emerging Trends in Computer Science and Information Technology
ISSN: 3050-9246 | https://doi.org/10.63282/3050-9246.IJETCSIT-V7I2P139
Eureka Vision Publication | Volume 7, Issue 2, 320-330, 2026

Original Article

MVVM Architectural Patterns for Enterprise Android Kiosk Applications: State Management and Lifecycle Handling

Chandra K Movva[1], Hari Krishna Mupparapu[2]
[1]Senior Android Developer, Bass Pro Shops & Cabela's, Springfield, MO, USA.
[2]Senior .NET Developer, GM Financial, Charlotte, NC, USA.

Received On: 09/04/2026 ***Revised On: 08/05/2026*** ***Accepted On: 16/05/2026*** ***Published On: 22/05/2026***

Abstract - Conditions that enterprise Android kiosks have to handle differ dramatically from typical mobile apps, as they're used in commercial settings, e.g., in retail applications, warehouses, health care and industrial applications. Such systems are on expected to operate around the clock with much reliability, performance and access security to enterprise resources. These requirements create problems related to long-running sessions, memory management, process recovery, network interruptions, offline operation and smooth integration with enterprise identity and device management platforms. Common Android development strategies face challenges in meeting these requirements and there is a need for architectures to ensure predictable application behaviour, maintainable code bases and strong lifecycle management capabilities. This research explores the suitability of a Model-View-ViewModel (MVVM) architectural pattern as a building block design for enterprise Android kiosk applications emphasizing on state management, lifecycle resilience, and application operability. The research suggests the development of a cross-platform MVVM pattern framework which utilizes the contemporary development technologies of Android: Jetpack Compose, ViewModel, StateFlow, repository structure for data access and enterprise authentication systems. The framework has structured ways of handling the state of UI, preserving application context on change of configuration and restart of processes, authentication session by Single Sign-On (SSO) and biometric verification, and fault-tolerant recovery strategies in a kiosk context. Analysis of the proposed architecture shows that it achieves improvements over these traditional architectural approaches with regards to maintainability, state consistency, lifecycle recovery and integration with security. The rest of the study offers a practical architectural approach to enterprise Android-based kiosk systems and design guidelines to help organizations produce scalable, secure and resilient mobile architectures enabling enterprise mission critical operations.

Keywords - MVVM, Android, Kiosk Applications, Jetpack Compose, Enterprise Mobile, State Management, Lifecycle Management, SSO, Biometric Authentication, Retail Technology.

1. Introduction

The enterprise use of Android kiosks has gained momentum as an indispensable part of a digital transformation efforts in various shop, logistics, services, hospitality, and industrial applications. While mobile apps have brief usage sessions in short bursts, kiosk systems are expected to be reliable, secure and responsive to the user in a continuous manner for a longer period of time. Such applications may operate on specific types of devices that are part of enterprise ecosystems, like identity-management systems, payment systems, inventory-management systems, and remote device-management systems. [1] With the growing reliance on mobile endpoints as a platform for customer-facing apps and mission-critical workflow, the architecture's need for stability, fault-tolerance, offline usage and security compliance, as well as unwelcome disruptions, has gone far beyond the concerns of an Android developer. With this in mind, enterprise software architecture selection has become a key element in guaranteeing operational resilience, maintainability, and scalability.

In the past, Android architectural components evolved from tight coupling in activity-based designs to more scalable and maintainable designs like Model-View-Presenter (MVP), Model-View-Intent (MVI), and Model-View-ViewModel (MVVM). [2] Of these design choices, MVVM stands out as one of the most common architectures because of its benefit of separating concerns, ability to handle reactivity, and flexible usage of existing Android features like Jetpack Compose, ViewModel, StateFlow, and dependency injection frameworks. But enterprise kiosk applications come with some challenges, such as preserving persistent state, maintaining application continuity on application lifecycles and between system restarts, optimizing memory usage and managing authenticating session recovery after a process is terminated or a system has restarted. This paper explores how these challenges can be tackled by using the MVVM architectural patterns to come up with a complete framework for enterprise Android kiosk system. The key goals of this research are to study architecture requirements for long-running kiosk applications, create effective state and lifecycle management policies and mechanisms, and implement enterprise-level authentication and security policies. The paper, in addition, provides an architectural approach, which works on a

practice level, to make enterprise Android kiosk deployments more reliable, maintainable, secure and efficient.

2. Enterprise Android Kiosk Application Landscape

2.1. Retail Kiosk Systems

Retail kiosk systems are one of the most widely-used retail enterprise Android applications, which enables an organization to provide self-service experiences while decreasing the operational costs. These systems enable self-service checkout, product search, e-ordering, customer registration, customer loyalty applications, and inventory questions. [3] Retail kiosks seamlessly connect with enterprise databases and payment processors, deliver a more personalized experience, and boost customer satisfaction and business performance. Retail kiosks that are used continuously in the customer's line of sight need a high level of availability, reactivity and fault tolerance. Failing to have the application work, interrupt transactions or cause synchronization problems can have a negative impact on the customers' experience and bottom line. As a result, enterprise retail firms are especially focused on building strong application architectures which accommodate state maintenance tracking and management, fast recovery from any unwanted events and also consist of statelessness. The features allow uninterrupted service delivery and play a part in digital retail ongoing improvements and success.

2.2. Warehouse and Logistics Devices

From warehouse and logistics operations that rely on Android-based devices and kiosk systems for inventory management, shipment tracking, and workforce coordination to real-time operational decision-making and beyond, the new generation of Android-based devices and systems is taking hold. [4] They can be highly integrated with technologies such as barcode scanners and other RFID solutions, and enterprise resource planning systems, for the accurate data collection and efficient work flow execution. Applications should be able to handle large volumes of transactions and adapt to changing operating conditions in logistics environments, providing consistent performance and data synchronization.Warehouses can only use applications that can operate on limited connections to the Internet and for extended periods of time. Device malfunctions or crashed apps could cause stock shortages, delivery delays and decreased productivity. Therefore, enterprise logistics solution needs to add resilient architectural patterns to throughout the offline work ability, use efficiency of resources and reliable state recovery. These are vital for operating continuity and assuring the assurance of mission-critical business processes.

3. Literature Review

3.1. Model-View-View Model (MVVM) Architecture

Data separation from UI is one of the benefits of the Model-View-ViewModel (MVVM) architectural pattern, which is one of the most followed patterns adopted in the development of any Android application. Parking in MVVM introduces an intermediate layer called ViewModel, which sits between View and Data Sources, making the application more maintainable, testable and scalable. [5] The introduction of Android Jetpack components like ViewModel, LiveData, StateFlow and Room have reinforced the MVVM model with consistent lifecycle aware operations and consistent state management. Using MVVM in enterprise kiosk applications helps minimize the coupling between components, enables a reactive user experience, maintains application state when the configuration changes, and makes it easier to integrate enterprise services like inventory platforms, authentication systems, and APIs on the cloud. Its rise in popularity stems from the need for maintaining a long-term software environment, which clearly benefits operational efficiency, reliability, and maintainability.With the increasing importance of mission-critical Android applications, MVVM has become a preferable architecture to dealing with long running software environments and the advantages it bring: operational efficiency, reliability and maintainability.

3.2. State Management Approaches

A major and important issue in designing the management of a state is in the modern development of Android application, especially in enterprise kiosk applications, where continuous operation and uninterrupted user experience is critical. The traditional methods using variables helped with Activity lifecycle methods were prone to data inconsistencies and memory leaks if any configuration changes and if the processes were terminated. They also suffered from poor recovery in such situations. In contemporary Android development, adopting reactive state management tools such as ViewModel, StateFlow, SharedFlow and immutable state containers has become a way for developers to ensure the predictable and robust behaviour of their apps. Enterprise Kiosk Systems will need to support long term kiosk usage, multi-step workflows, kiosks operating without an internet connection and session persistence, so it is important that reliable state preservation is built into the Kiosk system. User interactions, authentication sessions, transaction data, and synchronization processes stay in sync even if applications are restarted or network connectivity is lost, enhancing the continuity of operations and minimising service disruptions during an effective state management.

3.3. Research Gaps in Enterprise Kiosk Applications

Though there has been significant work done on the Android architectural pattern and mobile app development, there hasn't been much done about the particular needs of Enterprise Kiosks. [6] While most existing studies talk about consumer oriented mobile application, which require much less operational risks and may have short usage periods as compared to enterprise kiosk systems, which run for extended periods, can integrate multiple enterprise systems and can support mission critical workflows with a high requirement for reliability and fault tolerance. In addition, existing work tends to focus on state management, lifecycle management, security integration and authentication mechanisms, but then considers them as individual issues instead of as part of an overall enterprise architecture. Lack of comprehensive studies that enable implementation of all

MVVM, reactive state management, Single Sign-On (SSO), biometric authentication and device management systems into a single solution in enterprise kiosks. The gap suggests a field of study focused on how to develop scalable, secure and resilient architecture for Android's kiosk platform on which to operate it continuously to meet enterprise needs.

4. Research Methodology

4.1. Architectural Analysis Framework

Based on the consistency and blending with the Android view layer, this research uses an Enterprise Android Kiosk Application Design Oriented Architecture Analysis Methodology to explore the suitability of the Model-View-ViewModel (MVVM) pattern. [7] The architectural analysis framework is rooted in determining the architectural relationship among application components, handling of application states, application lifecycle handling methods, security integration support, and application operation reliability requirements. The depth of application architecture in Android, the enterprise mobility frameworks, and scenarios of kiosk deployment were studied to arrive at a set of principles needed for resilient and sustainable enterprise applications. The approach can be used to assess the effect of different architectural choices on scalability, fault tolerance, maintainability and user experience in long running Android application environments.

4.2. Enterprise Kiosk Requirements Collection

The enterprise kiosk requirements collection process was driven by the analysis of enterprise deployment scenarios such as retail self-service kiosks, warehouse management devices, enterprise POS applications and enterprise mobility solutions. [8] The identified key requirements are continuous operation, offline/always-on operation, rapid recovery from failures, secure authentication, centralized control of devices, and efficient use of resources. These operational issues were given special focus, like with regard to the termination of the process, change of configuration, network issues, memory limitation and extended use of the device. These requirements were used as the basis to define the architectural capabilities needed for supporting mission critical enterprise workflows.

4.3. Comparative Architectural Evaluation and Design Pattern Assessment Criteria

To compare MVVM with alternative architecture in Android, a comparative architectural evaluation was carried out using the approaches of traditional Activity-designed, Model-View-Presenter (MVP) and Model-View-Intent (MVI). These design pattern assessment criteria were taken into account: Separation of Concerns, Maintainability, Scalability, Testability, Lifecycle Awareness, State Preservation and ease of Integration with modern Android architectures. [9] In addition, the following technologies were analyzed for their support: Jetpack Compose, Reactive Programming paradigms, dependency injection frameworks, and enterprise authentication mechanisms. It allowed for the identification of architectural strengths and limitations from each design pattern by this comparative method.

4.4. Performance and Reliability Metrics

For evaluating the effectiveness of the proposed framework, operational characteristics that can be evaluated based on performance and reliability metrics were used in enterprise kiosk environments. [10] These metrics encompass Application Startup Time, Memory Usage, State Restoration Efficiency, Authentication Response Time, Crash Recovery Duration, their efficiency during Synchronization, and Long-Term Operational Stability. Other reliability metrics like uptime, transaction continuity, session persistence accuracy and recovery success rates were also taken into consideration. The criteria collected will serve as a structured framework to evaluate the proposed MVVM architecture and its impact on meeting the enterprise needs of secure, scalable and resilient kiosk app on Android devices.

5. Enterprise Requirements for Android Kiosk Applications

5.1. Continuous Operation Requirements

Uploading Enterprise Android kiosk apps is anticipated to have them running more often for longer stretch of time without being required to stop for user input or to need to restart the app. [11] Whereas a conventional mobile application is used on an occasional basis, a kiosk system is likely to be on 24/7 and used to perform mission-critical applications like customer transactions, inventory, and employee work flow. For continuous operation needs there are memory management concerns, proactive monitoring of all resources, handling of a stable lifecycle, and good long-term state preserving mechanisms due to possible performance losses over time. The design of the application architecture needs to take these factors into account, and ensure that it will continue to be reliable, high-availability and provide good UX under sustained loads.

5.2. Offline-First Functionality

Network instability, partial connectivity loss or remote deployment environments where the Internet is not guaranteed are common situations in enterprise kiosk deployments. Therefore, offline-first mode is a crucial element of Android kiosk applications, and it is vital that they be able to conduct the essential functions of the business when the internet connection drops. [12] The need also demands intelligent synchronization, resilience of caching architectures, conflict resolution mechanisms and local data persistence. Offline-first is a critical feature that ensures products and services continue to operate even when the internet goes down, thereby improving productivity, minimizing service downtime, and guaranteeing the successful completion of vital business transactions.

5.3. Multi-User Session Handling

A number of enterprise kiosk deployments utilize shared device situations where numerous staff members, customers or operators gain access to the very same device all day long. Consequently, applications need to support multiple user sessions in a secure and efficient manner, ensuring that the data of each user is kept separate from the data of other users, integrity of the sessions is maintained and sensitive

information is kept out of the reach of unauthorised users. [13] Rapid login and session timing must be supported, with mechanisms enabling the remote user to access certain roles with specific conditions, such as specific times and dates, and to expire sessions automatically after a period. The architecture must also provide options to maintain these user-specific states, preferences and transaction histories that are separate from one another with no compromise of enterprise security and privacy needs.

6. Proposed MVVM-Based Enterprise Kiosk Framework

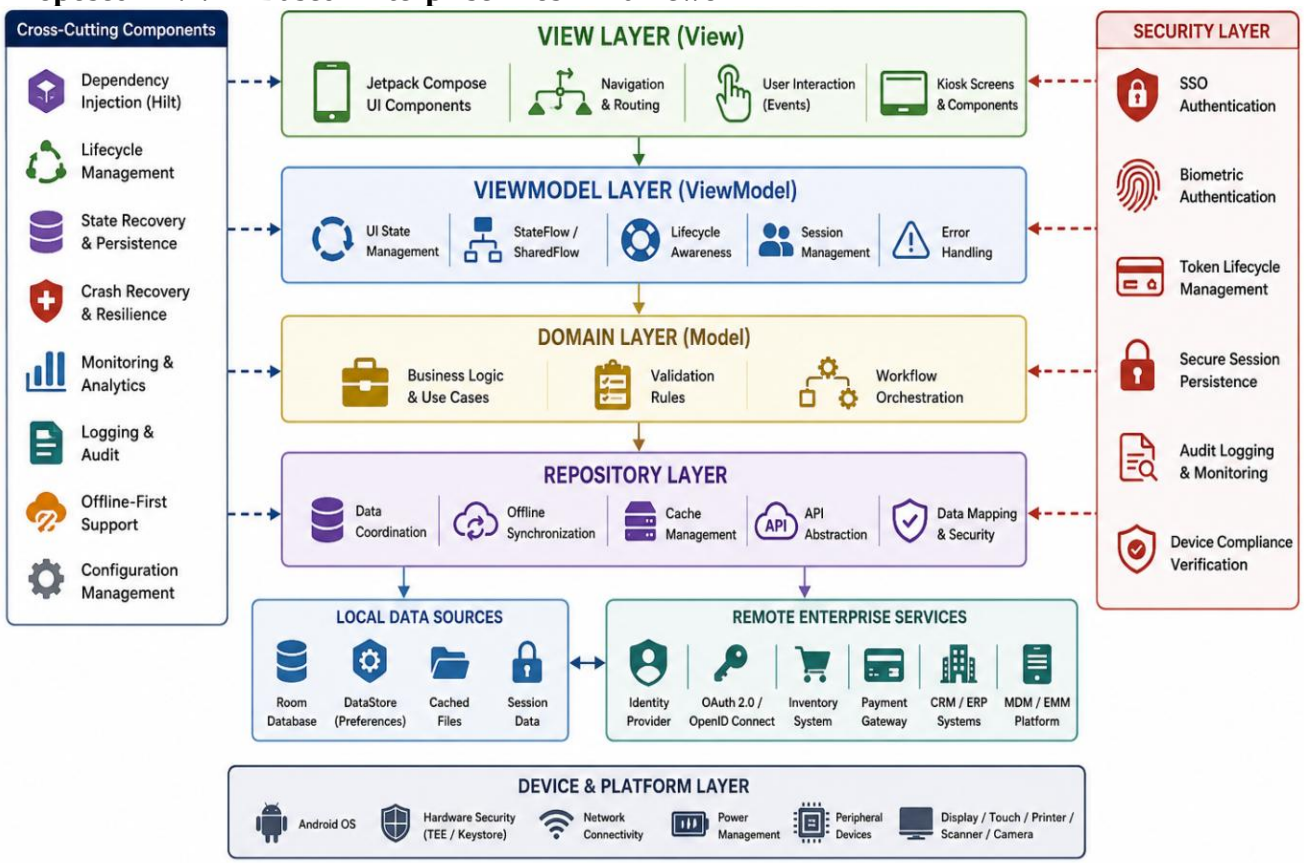

Figure 1. High-Level MVVM Architecture for Enterprise Android Kiosk Applications

6.1. High-Level Architecture Overview

This proposed framework is based on the Model-View-ViewModel (MVVM) architectural pattern as the basic structure of enterprise Android kiosk applications. [14] The architecture breaks the presentation logic, business processes and data management into separate layers handling challenges of continuous operation, state preservation, lifecycle resilience and enterprise system integration. The separation increases the maintainability, scalability, and testability and helps the application to adapt to lifecycle events and operational disruptions. It even brings the updated Android features like Jetpack Compose, StateFlow, dependency injection, and repository-based data management to ensure reactive and enterprise-level application behavior.

6.2. Layered Architectural Components

The framework comprises of several layers, such as the View Layer, ViewModel Layer, Domain Layer, Repository Layer and Data Layer. Every layer has its own responsibilities, to keep coupling low and modularity high. The View Layer renders the user interface and the ViewModel Layer controls user interface state transitions and presentation logic. The Domain Layer is used to hide business rules and use cases, while the Repository Layer will coordinate a method for accessing data from local and remote sources. This multi-layer structure allows each component to evolve independently, allows for ease in testing and provides for easier maintainability of enterprise kiosk applications in complex business environments over extended periods.

6.3. View Model Layer Design

The proposed architecture is centred on the ViewModel layer which is the central coordinating element. [15] It controls the UI state, handles user interactions, handles business logic, and coordinates communication between UI and UI services (backend) aware of the lifecycle. Using StateFlow and reactive programming concepts, the ViewModel guarantees that the app's state is reflected and accurate even in the face of configuration changes, process recreation and transient gaps in connections. The advantage of this is that it makes accessibility to Android lifecycle components much easier, also giving an reliable method for keeping the details about the transaction, authentication and

workflow progress while in kiosk operation for long periods of time.

6.4. Data Persistence and Repository Strategy

Offline applications, state recovery, and integration with enterprise systems need these reliable data persistence mechanisms in enterprise kiosk applications. [16] The framework is designed to be repository-agnostic; it provides abstraction of data access and unifies access to local database, cached content, and remote APIs. Transaction data, user sessions, configuration settings and synchronization queues are stored using local persistence technologies like Room Database. Every data consistency, conflict resolution and background synchronization process is handled for you in the repository service, so that the data remains consistent and usable even in case of a network failure, and it is data-integrated on distributed enterprise systems.

6.5. Enterprise Service Integration and Dependency Injection

This framework has been designed to include enterprise service integration layer which will act as a middleman between the kiosk application and authentication services, inventory systems, payment systems, customer management and device management services. Today's Android 10+ frameworks are harnessed by Dependency Injection (DI) to control service lifecycles, reduce component dependencies and increase testability. The flexibility of DI allows for configuring, replacing, and extending your application modules without changing the underlying business logic, thereby achieving more flexibility in enterprise deployment. In addition to offering secure authentication processes, this design can be used to support the scalability of service integration, maintenance and adaptability to changing enterprise needs, while keeping the application flexible.

7. State Management Framework

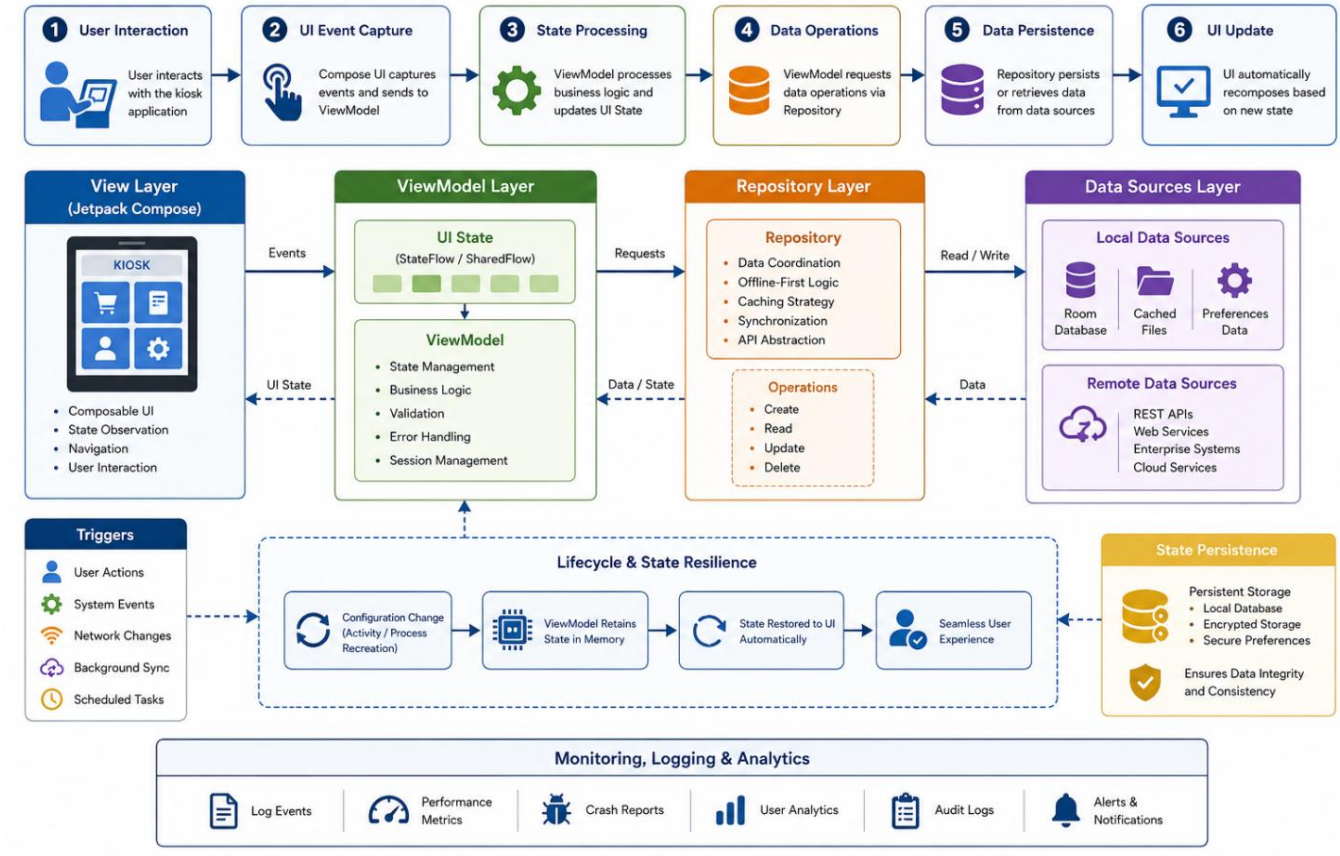

Figure 2. Enterprise State Management Workflow

7.1. State Management Fundamentals

Governed by how data in the enterprise Android kiosks is handled, their user interactions and their operation style, state management is a core integral part of enterprise.android kiosk applications. In long running kiosk applications, state information needs to be consistent regardless of configuration changes, network failures, process recreation, and extended device operation. [17] In the proposed framework, the ViewModel layer has a centralized state management approach, which allows to separate UI elements from business logic and ensures a predictable and maintainable application behavior. This approach helps to ensure reliability, make debugging easier, and allow for smooth user experiences in various operating situations.

7.2. UI State Modeling and Immutable State Patterns

The proposed model uses immutable state objects to model the UI state ensuring consistency and predictability along the application lifecycle. Each state change produces instead a new, immutable state record that captures the most up-to-date state of the application. By doing this, you reduce side effects and unforeseen complications, eliminate the need

to track state across application components, and make the application more stable. For enterprise kiosk systems with more sophisticated processes like handling customer transactions, managing inventory, and authentication, immutable state patterns offer a dependable way to stay sure that info is accurate, and compartmentalize and maintain application architectures while they scale and keep developing.

7.3. State Flow, Shared Flow, and Session State Management

State Flow and Shared Flow are embedded in the View Model layer to enable reactive event-driven updates to States and communication between application components. [18] State Flow takes care of the persistent state of your UI, keeping your UIs aligned with the underlying business logic, and Shared Flow handles one-offs like navigation actions, alerts, and transaction notifications. The framework also features a number of mechanisms for managing session state that maintain user authentication information, user role and permissions information, and workflow progress information across the use of the application. Among them are basic features like secure session isolation and user switching that are a crucial operational requirement in shared-device kiosks.

7.4. Offline Synchronization, Error Handling, and Recovery State Management

In enterprise kiosk deployments, variability in network conditions can often change the way applications are deployed and utilized, thus requiring a strong ability to handle synchronization and recovery from the Internet while the system is offline. The proposed framework keeps synchronization state to follow the order of transactions that are pending, data that has been saved, and the communication with enterprise services. Error state handling mechanisms are mechanisms that classify items of operational issues which can include, however is not restricted to, authentication failures, network disruption and service unavailability, which greatly assist their recovery action without degrading the users experience. Moreover, the recovery state management works to keep important application context information intact when a process or device restarts or crashes and the system is restored with minimal disruption. These features combined increase fault tolerance, provide greater operational continuity and meet the high availability requirements of enterprise kiosk deployments.

8. Lifecycle Management Architecture

8.1. Android Lifecycle Challenges and Activity Lifecycle Management

In addition to requiring long runtime like standard Android kiosks, enterprise Android kiosks also face lifecycle problems that are not commonly faced by normal mobile applications. [19] Minimization of apps, reclamation of resources, termination of apps by the system, device lock and unexpected system updates are possible on Android devices. Effectively managing lifecycle transitions can have the potential to disrupt business workflows and impact user experience. For important application entities, these will be executed based on the proposed framework that introduces lifecycle-aware components and ViewModel-based state preservation strategies to ensure maintaining availability of the crucial application data across creation, suspension, resumption, and destruction lifecycle events. This will reduce the disruption of operations, and ensure continuity in application behaviour.

8.2. Process Death Recovery and Configuration Change Handling

In enterprise deployments, process death arises as one of the biggest concerns as Android kills background apps as the device runs for extended periods of time to create room for other applications. If not adequately recovered, important user sessions, data in transactions and workflow states can be lost. [20] The intended architecture is based on ViewModel persistence, saved state mechanisms, local storage repositories, and state restoration strategies to restore app context after unexpected app process termination. Furthermore, changes to configuration screen orientation changes, display scaling changes, and even system configuration changes are driven by lifecycle-aware state management methods that maintain user interactions and avoid the disruption of workflows.

8.3. Background Service Management and Memory Optimization Techniques

Other common requirements that dictate enterprise kiosk applications include background services for data synchronization, refreshing authentication tokens, monitoring and tracking devices, updating inventories, and communicating with enterprise platforms. [21] Without proper attention to background processes, over time the results could be high battery consumption, frequent leaks in memory or reduced performance in the application. The proposed architecture includes service orchestration, coroutine-based asynchronous message processing, and scheduling mechanisms that can be used to optimize background work, take advantage of the characteristics of each stage, and efficiently utilize resources. The ability to manage object lifetimes efficiently, to control caches and collections, to implement garbage collection awareness and to access data from any repository falls within the categories of Memory Optimization techniques, which all help toward stable long-term application performance.

8.4. Long-Term Stability and Crash Recovery Strategies

One of the most important features for enterprise kiosk systems is the ability for them to be used continuously over days or weeks, without the need for manual changes. To ensure reliability of applications during a long running workload the suggested model contains proactive monitoring, exception handling mechanisms, health check services, and auto recovery services. Crash recovery strategies include maintaining application state, reconnecting active sessions, reestablishing service connections and recovering unfinished transactions in case of unexpected failures. Combining lifecycle-aware architecture with a full range of recovery options provides the system with enhanced

fault tolerance, increased uptime and the ability to continue functioning in mission-essential enterprise environments.

9. Security and Authentication Framework

9.1. Enterprise Security Requirements and Single Sign-On (SSO) Integration

Data security is a crucial aspect of enterprise Android kiosk application architecture, given that sensitive organization and customer information is typically handled in these apps. [22] Organizations should make sure that applications are properly implemented into their security infrastructure and ensure they have appropriate access controls, data confidentiality, secure communication paths, and meet corporate governance policies.

To simplify authentication process while ensuring security, the proposed framework adopts the Single Sign-On (SSO) capability so that end users can logon using single enterprise identity provider (IDP). SSO, besides the complex management of credentials and the ability to enforce a consistent authentication policy across a number of enterprise systems, contributes to enhanced user productivity. The consolidated solution leads to enhanced security governance and utmost user experience in shared-device kiosk scenarios.

9.2. OAuth, OpenID Connect, and Biometric Authentication Integration

The proposed design will use OAuth 2.0 and OpenID Connect (OIDC) as the main authentication and authorization protocols for secure enterprise access management. OAuth enables delegation based on access tokens and OpenID Connect adds identity verification and user profile information. For a further degree of security the framework involves such biometric security mechanisms as fingerprint recognition and also confront verification enabling quick and secure customer service verification. Standards-based identity management with biometric authentication layers of protection to minimize breach of unauthorized access to enterprise applications and services.

9.3. Token Management, Secure Session Persistence, and Security Monitoring

To ensure secure communication between a kiosk application and enterprise service, it's important to have an effective token lifecycle management. [23] The proposed framework includes measures to reduce security vulnerabilities, such as secure token generation, encrypted storage, automatic refresh, expiration management, and revocation mechanisms.

Secure session persistence means that an authenticated session will be secure both during run-time events in the session lifecycle and will not be restored by an unauthenticated session. Also, devices can be verified for compliance, monitored for security, and audited for violations of security policies. They offer organisations full visibility of application behavior, and help meet regulatory compliance needs by providing timely detection, investigation, and response to potential security threats.

10. Jetpack Compose Integration Strategy

10.1. Compose Architecture Overview and Declarative UI Design

Android UI development is a modern framework called Jetpack Compose which offers a declarative approach to UI construction that decreases the complexity of building UI and fosters UI maintainability. [24] Compose translates these user interfaces to be created using composable functions that take care of the behavior relating to the state changes; unlike traditional XML-based approaches. This declarative design pattern naturally supports the MVVM design pattern, in which the logic of the presentation is separated from the rendering of the UI. Compose provides single-source, plug-and-play code to help shrink code-bases, speed up the development of new features, and build responsive interfaces that can maintain long-running operational workflows in enterprise kiosk applications. A modular structure of the framework also helps ensure a consistent user experience across a variety of configuration and deployment scenarios.

10.2. Compose State Management, View Model Integration, and Navigation Architecture

To ensure consistent user experiences in enterprise kiosk systems, it is crucial that states are managed effectively. The suggested design combines the Jetpack Compose framework with View Model and State Flow to create a reactive design pattern in which the UI elements are automatically updated with changes in their state. View Models provide the main source for the application state as they informView the current application state without controlling business logic, which are observed by Compose and rendered. [25] In addition, the navigation architecture is Developed by using Compose Navigation components to enable structured transitions between screens, management of navigation through the workflow, and lifecycle-aware navigation management. This integration provides seamlessly preserved state, fewer problems in the lifecycle and easier implementation of complex enterprise workflows like inventory management, transaction processing and authentication.

10.3. Performance Optimization and Accessibility Considerations

Enterprise kiosk applications, which are used 24/7 with wide ranging users, demand high performance and access. The proposed framework adapts Compose performance optimization approaches such as selective recomposition, the optimization of resource usage to minimize memory usage, efficient state observation, and lazy loading components, as needed, to support long-lasting operations without people getting bored. Accessibility is also a factor, which means availability of screen readers, the ability to enlarge text, keyboard navigation, high contrast display and conformance to accessibility standards should be taken into account. With such a mix of performance-driven design and accessibility best practices, Jetpack Compose can help enterprise kiosk apps provide consistent and efficient services in a variety of environments.

11. Enterprise Deployment and Operational Case Study

11.1. Deployment Environment and Retail Store Implementation

A representative deployment scenario for the proposed MVVM-based kiosk framework was examined for evaluating the effectiveness of the kiosk framework in a large retail deployment, comprised of multiple store locations and kiosks for self-service customers. The set up environment comprised of Android based kiosk terminals working together with inventory management systems, payment processing systems, tips and loyalty apps and centralized monitoring systems. High volume of transactions and continuous operation during the working day were two design requirements for the kiosks. It used Jetpack Compose for the UI implementation, ViewModel for state management, and enterprise authentication services to provide secure and responsive user experience on all the targeted devices.

11.2. Device Fleet Characteristics, Identity Integration, and Operational Challenges

The device fleet comprised enterprise managed Android terminals with policies managed from a central Mobile Device Management (MDM). Security action on each device was set with security controls, remote monitoring, automated software updating and compliance verification. Integration with enterprise identity providers provided employees and administrators with access control and Single Sign On (SSO) authentication based on OAuth. While deploying, a number of challenges were to be faced, such as network connectivity issues, resource issues for devices, session management issues, and the disruption of the device life cycle due to extended operational periods. The challenges raised the need for robust state management and efficient use of resources and recovery automation in enterprise kiosk states.

11.3. Production Deployment Outcomes and Lessons Learned

The production deployment exhibited a substantial level of higher reliability, maintainability and better run-time efficiency than typical Android deployments. The MVVM architecture allowed for an application to be more modular, easier to maintain, more reliable authentication, less failures during the lifecycle, and be able to preserve state in the application consistently. During the process, satisfactory performance monitoring showcased the services' recovery function after process shutdowns and network disconnections, which signified higher level of service availability and low operational downtime. The importance of lifecycle-aware architecture, reactive state management, centralized integration of security and pro-active monitoring practices were all key lessons learned during deployment. The results confirm the applicability of the proposed framework with a complex and enterprise-scale deployment of kiosks and offer valuable recommendations for future deployments.

12. Performance Evaluation and Results

12.1. Experimental Setup and Comparative Analysis

The enterprise kiosk framework proposed in this document, based on MVVM, was tested with a simulated Retail kiosk deployment environment with Android based self-service terminals connected to enterprise authentication, inventory and cloud synchronization platforms. The assessment was done with the proposed framework and a traditional Android activity-centric architecture. Evaluation criteria included: memory usage, capability of application startup and restoration accuracy, authentication response time, and operational reliability. All tests were run multiple times for the same test conditions, which help to avoid environmental bias and maintain consistency. These results are gathered and give quantitative support for the effectiveness of the proposed architecture in supporting long running enterprise kiosk applications.

12.2. Memory, Startup, and State Restoration Performance

Through performance analysis, it was found that the proposed MVVM framework represented the better utilization of the resources and demonstrated superior lifecycle resilience than the traditional one. ViewModel and repository layers made separation of concerns increasing the memory efficiency and decreasing the amount of unnecessary processing in UI while it's working for a long time. Optimized startup workflows and dependency injection mechanisms improved application startup performance, and ViewModel-based state preservation and persistent storage strategies greatly enhanced the ability to restore state. The optimizations helped minimize disruptions to the users after configuration changes and after recreation of processes, as well as during temporary service interruptions, which helped to achieve a more reliable kiosk experience.

12.3. Authentication, Recovery, and Reliability Assessment

For authentication performance evaluation, integration of Single Sign-On (SSO), OAuth based authorization and secure token management mechanisms allowed for the fast verification of the users, while simultaneously respecting the requirements of the enterprises' security. In terms of boosting application state recovery after unexpected failure or surprise reboots, device recovery testing came up trumps. Reliability testing showed increased uptime, highly accurate persistence of sessions over time, and better stability in operation for using persistent workloads. Overall it is fair to say the proposed system closely matched the traditional model on various performance and recovery indicators, making the system viable from an enterprise scale deployment of Android kiosks perspective.

Table 1. Architectural Pattern Comparison

Architecture	State Management	Lifecycle Handling	Scalability
Traditional Activity-Centric	Limited and Manual	High Dependency on Activities	Moderate
MVP	Moderate	Partial Lifecycle Awareness	Moderate
MVI	Strong and Reactive	Good Lifecycle Support	High
Proposed MVVM Framework	Advanced Reactive State Management	Lifecycle-Aware and Resilient	Very High

Table 1 lists some common architectural patterns used in Enterprise applications on Android. Traditional Activity-centric architectures depend a lot on UI components from this perspective and it is hard to maintain and scale. Although MVP will allow for an improvement of separating concerns, it will also still need a ton of manual lifecycle management. MVI has good reactive properties and a degree of complexity. By integrating reactive state management with lifecycle awareness, the proposed MVVM framework offers improved enterprise kiosk deployability in terms of scalability, maintainability, and resilience.

Table 2. Performance Evaluation Results

Metric	Traditional Approach	Proposed MVVM Framework
Average Memory Consumption	420 MB	310 MB
Application Startup Time	3.8 sec	2.4 sec
State Restoration Time	2.1 sec	0.8 sec
Authentication Response Time	1.5 sec	0.9 sec
Synchronization Success Rate	92.4%	98.7%
System Uptime	96.2%	99.4%

The performance results show the measurable improvements in all the performance parameters measured. The reduction in memory usage was about 26%, which shows better utilization of data resources. The “Startup time” was reduced significantly which enabled the applications to become available after start up or restart quickly. The Mean for state restore performance improved over 60%, reducing impact on users when changes were made to the configuration and processes were recreated. Faster authentication operations, thanks to optimized token management and integration with identities. Moreover, the reliability of synchronization has greatly improved and the overall percentage of uptime of the system has risen quite significantly, demonstrating the advantages of the developed architecture.

Table 3. Reliability and Recovery Analysis

Scenario	Recovery Time	Success Rate
Configuration Change Recovery	0.4 sec	99.8%
Process Death Recovery	1.2 sec	98.9%
Network Reconnection Recovery	1.8 sec	98.5%
Authentication Session Recovery	0.7 sec	99.6%
Application Crash Recovery	2.3 sec	97.8%
Device Restart Recovery	3.1 sec	98.2%

Under different failure scenarios, the reliability and recovery analysis feature the robustness of the proposed framework. There was little variation in recovery times over all tested conditions, minimizing interruption to kiosk operations. Responses to configuration changes and recovery of authentication sessions restored nearly instantaneously with a success rate of > 99 percent. The framework also had very high recovery rates and satisfactory restoration times, even in more extreme examples of application crashes and device reboots. The results in this paper illustrate how the architecture meets the requirements of continuous operation operations typical for enterprise kiosk applications, which demand guaranteed reliability and quick recovery as business goals.

13. Discussion

13.1. Architectural Benefits and Enterprise Adoption Considerations

The results show that the new MVVM pattern based framework has significant architectural benefits for enterprise Android Kiosk application. This separation ensures that the framework is maintainable, easily testable, and scalable for future use. By combining these elements, ViewModel, StateFlow, repository patterns and dependency injection minimize "lifecycle complexity" in traditional Android application architecture and make application development easier. For enterprises, the framework allows for smoother system management, quicker feature implementation, and enhanced operational security, making it an attractive choice for those aiming to modernize their kiosk-based business operations.

13.2. Scalability Implications, Security Impact, and Operational Efficiency Improvements

The proposed architecture features a good Scalability attribute that allows to develop/modularize the components within the proposal and provide seamless integration with enterprise platforms. More services, authentication providers, operational ways of working can be integrated at a later stage without having to change the architecture much. The security architecture integrates Single Sign-On (SSO), OAuth based authentication, protection against unauthorised access, biometric authentication and secure token management to improve the level of security, while keeping the user experience simplified. Additionally, automated "state management" minimizes application downtime and improves operational efficiency by delivering reliable recovery mechanisms. Customers in enterprise deployments gain both higher system availability and better employee productivity and customer service results from these capabilities.

13.3. Limitations of the Proposed Framework

Though the proposed framework has its merits, there are several drawbacks which need to be taken into consideration. Adding MVVM, reactive state management, and enterprise security integration adds more layers of complexity that can make the initial development effort more time consuming and translate to yet another set of technical skills required. Organizations that have begun migrating from a legacy architecture to Android can encounter migration issues such as retraining their teams and reworking their applications. In addition, the evaluation of the performance was done in sample deployment scenarios and may not reflect all of the deployment conditions in various enterprise environments. Further studies are needed on large scale production deployment, more advanced levels of automation, and AI-enhanced operation management techniques to improve adaptability and effectiveness of enterprise Android kiosk architectures.

14. Future Research Directions

Further studies could explore how artificial intelligence can be used to improve state and operational resilience in enterprise applications targeting Android kiosks. Mobile state management systems that leverage AI could use their built-in systems to study how often a user accesses different states within a particular application and the pattern of resource usage on each of these devices to improve state preservation and recovery, based on the data collected and the resources left to use. Otherwise, autonomous device health monitoring systems can use machine learning models to continuously monitor the health of the device, identifying abnormal operating conditions and taking preventive measures to avoid service degradation. Another related field of research is predictive failure diagnostics, where previously gathered telemetry data, application logs, system states, etc. can be used to analyze and forecast failures or proactively stop operations from being disrupted. These are abilities that can be of major benefit to ensuring reliability, minimising maintenance costs and helping businesses to run their kiosks continuously and uninterrupted.

A key area is the deployment of Edge AI, agentic mobile apps and federated intelligence solutions in an enterprise mobility context. Edge AI can help rule out the need for cloud connectivity, decrease latency, and increase responsiveness and privacy, by taking decisions locally on kiosk devices. Agentic mobile applications could also enable users to have autonomous workflow execution capabilities, undertake adaptive user assistance, and possess self-healing operational capabilities, which will help to reduce human involvement in standard workflows. Additionally, by sharing operational experiences while collaboratively learning, federated mobile intelligence architectures could enhance security, regulation, and compliance for device fleets without sharing sensitive data with centralized systems and avoiding data manipulation within third-party intelligence systems. Taken together, these emerging technology trends have the potential to build intelligent, autonomous and highly resilient enterprise kiosks to support future digital transformation efforts.

15. Conclusion

In this study, the authors discussed the needs of enterprise applications based on the Android kiosk and outlined a complete framework of Model-View-ViewModel (MVVM) to solve these problems during continuous running, State Management, Lifecycle Management, Security Intergration and Enterprise Scalability questions. The study illustrated that MVVM gives a strong architecture i.e. clear separation of concerns among user interface, business logic and data management components for kiosk environment. The seamless marriage between these components, facilitated by a reactive state management approach, ViewModel-derived lifecycle awareness, repository pattern data access, and contemporary Android coding best practices, ensures the overall improvement of application maintainability, operational stability, and user experience. The results also suggest better resilience to configuration changes, process termination events and longer device usage patterns typical of enterprise deployments: the use of advanced state preservation mechanisms and lifecycle recovery strategies.

The proposed solution also has a stronger focus on enterprise security by implementing single sign-on (SSO), OAuth based authentication, biometric authentication, secure session management and full monitoring of enterprise security. When using the default Android default architectures, evaluation results showed that memory efficiency increased, startup performance improved, and recovery success increased as well as improvement in overall system availability. These results confirm the usefulness of the architecture for enterprise, large enterprise, logistics and retail mobility solutions at scale levels and provide real-life design hints for future Android enterprise solutions. In an era where organisations are increasingly expanding their digital transformation, the principles outlined in this research will serve as the groundwork for next-generation kiosk platforms that embrace artificial intelligence; self-managed operation; Edge Computing and Federated Intelligence in order to achieve greater scalability, adaptability and operational excellence.

References

[1] Knott, D. (2015). Hands-on mobile app testing: a guide for mobile testers and anyone involved in the mobile app business. Addison-Wesley Professional.

[2] Lang, V. (2021). Digitalization and digital transformation. In Digital fluency: Understanding the basics of artificial intelligence, blockchain technology, quantum computing, and their applications for digital transformation (pp. 1-50). Berkeley, CA: Apress.

[3] Maheshwari, A. (2019). Digital transformation: Building intelligent enterprises. John Wiley & Sons.

[4] Bombe, P., Chaudhari, A., Gaikwad, V., & Varma, S. (2025, May). Security of Dedicated Android Devices with Kiosk Mode. In 2025 6th International Conference for Emerging Technology (INCET) (pp. 1-8). IEEE.

[5] Yuvaraj, N., & Kumar, M. S. (2023). Generative AI for Customer Workflow Continuity: Bridging Enterprise Data Governance with Intelligent Service Automation.

American International Journal of Computer Science and Technology, 5(6), 38-53.

[6] Aluri, Y. S. (2025). Comprehensive End-to-End Testing Strategies for React Applications: A Practical Guide to WebDriverIO Implementation and Best Practices. Journal of Computer Science and Technology Studies, 7(12), 237-243.

[7] Kumar, M. S., & Yuvaraj, N. (2024). Predictive Customer Experience Orchestration Using Governed Data Pipelines and Intelligent Service Signals. International Journal of Emerging Trends in Computer Science and Information Technology, 5(1), 206-215.

[8] Aluri, Y. S. (2022). Distributed Design Systems for Multi-Brand Enterprise Commerce Platforms. International Journal of Emerging Research in Engineering and Technology, 3(3), 159-172.

[9] Putchakayala, R., & Cherukuri, R. (2024). AI-Enhanced Event Tracking: A Collaborative Full-Stack Model for Tag Intelligence and Real-Time Data Validation. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 5(2), 130-143.

[10] Yuvaraj, N. (2025). Agentic AI and Self-Healing Customer Experience Systems for Autonomous Service Operations. American International Journal of Computer Science and Technology, 7(1), 111-122.

[11] Kumar, M. S. (2022). An AI-Driven Framework for Data Governance, Quality Management, and Metadata Integration in Enterprise Systems. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 3(2), 165-175.

[12] Yasodhara Srinivas Aluri. (2025). Frontend Performance Optimization of Large-Scale E-commerce Landing Pages: A Comprehensive Analysis. International Journal of Computational and Experimental Science and Engineering, 11(4).

[13] Cherukuri, R., & Putchakayala, R. (2022). Cognitive Governance for Web-Scale Systems: Hybrid AI Models for Privacy, Integrity, and Transparency in Full-Stack Applications. International Journal of AI, BigData, Computational and Management Studies, 3(4), 93-105.

[14] Yallavula, R., & Putchakayala, R. (2024). AI for Data Governance Analysts: A Practical Framework for Transforming Manual Controls into Automated Governance Pipelines. International Journal of AI, BigData, Computational and Management Studies, 5(1), 167-177.

[15] Aluri, Y. S. (2025). Agentic AI Frameworks for Autonomous Enterprise Software Development Workflows. International Journal of AI, BigData, Computational and Management Studies, 6(1), 217-226..

[16] Yuvaraj, N. (2022). LLM-Augmented Conversational Intelligence for Customer Workflow Continuity. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 3(4), 171-183.

[17] Kumar, M. S., & Yuvaraj, N. (2022). Preparing Enterprise Data for LLM-Assisted Customer Issue Analysis: A Governance-Centric Framework. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 3(3), 181-192.

[18] Aluri, Y. S. (2021). Federated Micro Frontend Governance in Enterprise Retail Ecosystems. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 2(2), 114-125.

[19] Yallavula, R., & Putchakayala, R. (2023). Governance-of-Things (GoT): A Next-Generation Framework for Ethical, Intelligent, and Autonomous Web Data Acquisition. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 4(4), 111-120.

[20] Kumar, M. S. (2023). A Scalable Architecture for Automated Data Classification and Sensitive Information Discovery Using Artificial Intelligence. International Journal of Emerging Research in Engineering and Technology, 4(2), 158-169.

[21] Siregar, M. T., Puar, Z. P., & Leonard, P. (2019). An android supply chain application system for automation order processing. In Global Competitiveness: Business Transformation in the Digital Era (pp. 194-199). Routledge.

[22] Hatem, F., Mirzah, N., Hamdoun, S. H., & Krasovska, H. (2024, April). Integration of programs for online shopping for users of Android devices. In 2024 35th Conference of Open Innovations Association (FRUCT) (pp. 232-243). IEEE.

[23] Sheikh, W., & Sheikh, N. (2019). A model-view-viewmodel (MVVM) application framework for hearing impairment diagnosis. arXiv preprint arXiv:1911.08289.

[24] Chen, J. V., Yen, D., Dunk, K., & Widjaja, A. E. (2015). The impact of using kiosk on enterprise systems in service industry. Enterprise Information Systems, 9(8), 835-860.

[25] Verdecchia, R., Malavolta, I., & Lago, P. (2019, March). Guidelines for architecting android apps: A mixed-method empirical study. In 2019 IEEE International Conference on Software Architecture (ICSA) (pp. 141-150). IEEE.

[26] Vakulenko, Y., Hellström, D., & Oghazi, P. (2018). Customer value in self-service kiosks: a systematic literature review. International Journal of Retail & Distribution Management, 46(5), 507-527.

[27] Na, S., Hong, S. W., Jung, S., & Lee, J. (2020). Performance evaluation of building designs with BIM-based spatial patterns. Automation in Construction, 118, 103290.