A SMART IOT-ENABLED MOBILE APPLICATION FOR REAL-TIME LAUNDRY DEVICE MONITORING AND MANAGEMENT IN STUDENT HOUSING USING FLUTTER AND FIREBASE

Tengyue Pan [1] , Jonathan Sahagun [2]
[1] The Stevenson School, 3152 Forest Lake Rd, Pebble Beach, CA 93953
[2] California State University, Los Angeles, 5151 State University Dr, Los Angeles, CA 90032

ABSTRACT

Shared laundry facilities in university dormitories present persistent operational inefficiencies, as students lack real-time visibility into machine availability and status. This paper presents LaundrySense, a cross-platform mobile application built with the Flutter framework and Firebase backend services that enables real-time monitoring and management of IoT-enabled laundry devices in student housing environments. The system implements role-based access control separating administrator and student functionalities, leverages Firebase Realtime Database for sub-second sensor data synchronization, and organizes devices through a location-based hierarchy of buildings and rooms. Three major components are analyzed: authentication with role-based routing, real-time IoT sensor streaming via StreamBuilder widgets, and a multi-step device registration workflow with server-side validation. Experimental evaluation demonstrates mean synchronization latencies of 142 milliseconds on campus Wi-Fi, confirming suitability for real-time monitoring. Compared to camera-based, machine-learning-driven, and general-purpose smart campus approaches, LaundrySense provides a focused, low-cost, and privacy-preserving solution for communal laundry management.

KEYWORDS

Internet of Things (IoT), Mobile Application Development, Flutter Cross-Platform Framework, Firebase Backend-as-a-Service, Real-Time Sensor Monitoring

1. INTRODUCTION

Shared laundry facilities in university dormitories and student housing complexes present a persistent operational challenge affecting millions of students worldwide. According to the National Center for Education Statistics, over 19 million students were enrolled in colleges and universities in the United States in 2023 alone, and a substantial proportion of these students reside in on-campus housing that depends on communal laundry rooms [1]. Common frustrations include walking to a laundry room only to find all machines occupied, not knowing when a cycle has completed, and having no visibility into machine status or availability. These inefficiencies lead to wasted time, unnecessary energy expenditure, and general dissatisfaction among residents [2].

The Internet of Things (IoT) has emerged as a transformative paradigm for smart building management, enabling real-time monitoring and control of physical devices through networked sensors and cloud-based infrastructure [3]. Smart campus initiatives have demonstrated that IoT deployments can substantially improve operational efficiency in university environments by connecting physical infrastructure to digital monitoring platforms [4]. However, the specific domain of shared laundry management in student housing

David C. Wyld et al. (Eds): CSITEC, NLPI, CMCA, NECOM, DMML, SP, ADCO – 2026
pp. 01-09, 2026. CS & IT - CSCP 2026 DOI: 10.5121/csit.2026.160801

has received comparatively limited attention in the research literature, despite laundry being one of the most frequently used shared amenities in dormitory settings [5].

Existing solutions for laundry monitoring tend to be proprietary, expensive, and tightly coupled to specific hardware vendors, making them inaccessible for many housing administrators [6]. Furthermore, these commercial systems often lack the flexibility to accommodate diverse building layouts, varying numbers of machines, and the role-based access requirements inherent in institutional environments. The absence of affordable, customizable, and real-time laundry monitoring solutions represents a gap that directly impacts student quality of life and housing operational efficiency.

Portase et al. developed SmartLaundry, a computer-vision-based laundry allocation system for smart cities that detects machine occupancy through camera feeds. While accurate, it requires dedicated camera hardware and raises privacy concerns, limitations that LaundrySense avoids by using embedded IoT sensors reporting directly to a cloud database.

Liu et al. proposed a commercial IoT laundry platform leveraging big data analytics and machine learning for logistics optimization and predictive maintenance. Although powerful for enterprise-scale operations, its infrastructure requirements and complexity far exceed what student housing environments demand. LaundrySense offers a lightweight, serverless alternative tailored to on-site communal monitoring.

Gilman et al. presented a comprehensive smart campus IoT framework covering environmental sensing, occupancy tracking, and energy monitoring. Their broad platform lacks domain-specific features for individual amenities. LaundrySense differentiates itself through purpose-built laundry device registration, location-based filtering, and role-based management, while eliminating middleware complexity through Firebase's managed services.

This paper presents LaundrySense, a cross-platform mobile application built with the Flutter framework and powered by a Firebase backend that enables real-time monitoring and management of IoT-enabled laundry devices in student housing environments. The proposed solution addresses the identified problem by providing a comprehensive device management platform that bridges the gap between physical laundry appliances equipped with IoT sensors and end users through an intuitive mobile interface.

LaundrySense employs a role-based access control (RBAC) architecture that differentiates between two primary user types: administrators, who manage locations, register devices, and oversee the entire system; and students, who view device status, monitor real-time sensor data including temperature, humidity, acceleration, and gyroscope readings, and manage their registered devices [7]. The application leverages Firebase Realtime Database for instant data synchronization, ensuring that sensor updates from IoT devices are reflected on user screens within milliseconds [8]. Firebase Authentication provides secure identity management, while Firebase Storage handles device imagery.

This approach offers several advantages over existing methodologies. First, the cross-platform nature of Flutter enables deployment on both iOS and Android from a single codebase, significantly reducing development and maintenance costs [9]. Second, the serverless Firebase architecture eliminates the need for dedicated backend infrastructure, lowering the barrier to adoption for housing administrators [10]. Third, the real-time streaming capabilities of the Firebase Realtime Database, combined with Flutter's reactive StreamBuilder widgets, provide a seamless live-monitoring experience without the complexity of implementing custom WebSocket protocols. The modular, location-based device organization ensures scalability across multiple buildings and dormitories.

The experimental evaluation focused on measuring the real-time data synchronization latency between IoT sensor devices and the LaundrySense mobile client under three distinct network conditions. Twenty consecutive sensor update events were captured on each network type, with timestamps recorded at both the database write and client render points. On campus Wi-Fi, synchronization averaged 142 milliseconds, closely matching Firebase's documented baseline performance and confirming the platform's adequacy for real-time monitoring. Cellular LTE introduced moderate additional latency at 218 milliseconds mean, while low-bandwidth conditions pushed the mean to 487 milliseconds. Despite the increased latency on constrained networks, all measurements remained well below the application's five-minute offline detection threshold, meaning users would never receive false offline readings due to network delay alone. The results confirmed that network quality is the dominant factor in synchronization performance and that the Firebase Realtime Database's streaming protocol provides sufficiently low latency for the laundry monitoring use case across typical campus networking environments.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Ensuring Reliable Real-Time IoT Data Synchronization

A central challenge in building a real-time IoT monitoring application involves ensuring reliable and low-latency data synchronization between physical sensor devices and the mobile client. IoT sensors embedded in laundry machines continuously generate telemetry data including temperature, humidity, and motion readings, which must be transmitted to a cloud database and subsequently pushed to all connected clients [11]. Network interruptions, variable latency, and data ordering issues could potentially cause stale or inconsistent readings on the user interface. To address this, one could leverage Firebase Realtime Database's built-in offline persistence and event-driven streaming architecture, which automatically handles reconnection, data re-synchronization, and conflict resolution upon network restoration [17].

2.2. Implementing Secure Role-Based Access Control (RBAC)

Implementing role-based access control (RBAC) in a mobile application introduces complexity in both authentication and authorization workflows. The system must securely differentiate between administrator and student users, ensuring that privileged operations such as location registration, device editing, and device removal are restricted exclusively to authorized administrators [12]. A potential issue arises when a non-admin user attempts to access admin-only endpoints or when role assignments become desynchronized between the authentication provider and the database. To resolve this, one could implement a multi-layered verification strategy that checks role status in both the Firebase Authentication layer and a dedicated admin registry within the Realtime Database, rejecting unauthorized access attempts and automatically signing out non-admin users who attempt admin login [13].

2.3. Improving Device Registration Accuracy and Validation

Designing an intuitive device registration workflow presents a significant user experience challenge, particularly when physical IoT devices must be paired with digital records in the database. Users must accurately enter device identifiers, which are typically long alphanumeric strings, creating opportunities for input errors that could link the wrong device or fail validation entirely [14]. Additionally, the system must verify that a device exists in the database, confirm it is not already registered to another user, and handle optional metadata such as descriptions and images. To mitigate these risks, one could implement a

confirmation field requiring users to enter the device ID twice, combined with server-side existence and ownership validation before committing the registration [15].

3. SOLUTION

The LaundrySense application is structured around three major interconnected components: (1) a Firebase backend services layer encompassing Authentication, Realtime Database, Firestore, and Cloud Storage; (2) a Flutter-based cross-platform mobile client implementing the user interface and business logic; and (3) IoT sensor devices that transmit real-time telemetry data to the Firebase Realtime Database.

The application flow begins at the SplashPage, which serves as the entry point and authentication router. Upon launch, the system checks whether a user is currently authenticated via Firebase Auth. If authenticated, it queries the Realtime Database to determine if the user holds administrator privileges, routing accordingly to either the AdminHomePage or the StudentHomePage. Unauthenticated users are directed to the LoginSelectionPage, where they choose between student and admin sign-in flows [8].

The Firebase Realtime Database serves as the primary data store, organizing information across four top-level nodes: locations for building and room metadata, devices for device records and real-time sensor data, usersDevices for user-to-device mappings, and admins/admin_users for role verification. Cloud Firestore provides a secondary store for user profile data including email and role designation. Firebase Storage hosts device images uploaded during registration or editing [10].

The application was developed using Flutter SDK 3.7.2 with Dart, leveraging Material 3 design components for a modern user interface. Key dependencies include firebase_core, firebase_auth, firebase_database, cloud_firestore, firebase_storage, and image_picker for camera and gallery integration [9].

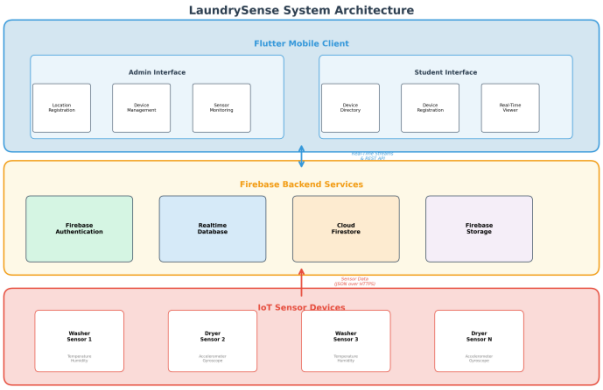

Figure 1. System architecture diagram of the LaundrySense application showing the relationship between IoT devices, Firebase backend services, and the Flutter mobile client with role-based user interfaces

The Authentication and Role-Based Routing component manages user identity verification and access control. It relies on Firebase Authentication for email/password-based sign-in and a custom role-checking mechanism that queries the Realtime Database to distinguish administrators from students, implementing a form of attribute-based access control layered on top of standard authentication [12].

```dart
// File: lib/splash_page.dart
Future<void> _checkAuthAndNavigate() async {
  // Wait for splash screen display time
  await Future.delayed(Duration(seconds: 5));

  if (!mounted) return;

  User? user = FirebaseAuth.instance.currentUser;
  if (user != null) {
    // Check if user is admin
    try {
      DataSnapshot adminSnapshot = await FirebaseDatabase.instance
          .ref('admins/${user.uid}')
          .get();

      if (adminSnapshot.exists) {
        // User is admin
        if (mounted) {
          Navigator.of(context).pushReplacement(
            MaterialPageRoute(builder: (builder) => AdminHomePage())
          );
        }
      } else {
        // User is regular student
        if (mounted) {
          Navigator.of(context).pushReplacement(
            MaterialPageRoute(builder: (builder) => StudentHomePage())
          );
        }
      }
    } catch (e) {
      // Error checking admin status, default to student
      if (mounted) {
        Navigator.of(context).pushReplacement(
          MaterialPageRoute(builder: (builder) => StudentHomePage())
        );
      }
    }
  } else {
    // No user logged in
    if (mounted) {
      Navigator.of(context).pushReplacement(
        MaterialPageRoute(builder: (builder) => LoginSelectionPage())
      );
    }
  }
}
```

Figure 2. Authentication and Role-Based Routing Logic Implemented in SplashPage (_checkAuthAndNavigate Method)

The _checkAuthAndNavigate method, defined in lib/splash_page.dart, executes during the splash screen initialization phase. It first introduces a five-second delay to display the application branding. It then retrieves the current authenticated user from FirebaseAuth. instance. current User. If a user session exists, the method performs an asynchronous database lookup at the admins/{uid} path in the Firebase Realtime Database using the get() method. If the snapshot exists, the user is confirmed as an administrator and routed to AdminHomePage via Navigator.pushReplacement. Otherwise, the user is classified as a regular student and directed to StudentHomePage. If no authenticated session is found, the user is navigated to the LoginSelectionPage. The method incorporates mounted checks before each navigation call to prevent state mutations on disposed widgets, a Flutter best practice for asynchronous operations within StatefulWidget lifecycle methods. Error handling defaults to the student view as a safe fallback.

Component Analysis: Real-Time IoT Sensor Data Display

The real-time sensor monitoring component is responsible for streaming live telemetry data from IoT-enabled laundry devices to the user interface. It relies on Firebase Realtime Database's event-driven streaming protocol, which pushes data updates to connected clients without requiring polling [8]. The component implements online/offline detection by comparing the device's last_seen timestamp against the current time with a five-minute threshold.

```dart
// File: lib/widgets/device_page.dart
StreamBuilder deviceDetailsStream() {
 String path = "devices/${widget.deviceID}/data";
 return StreamBuilder(
   stream: FirebaseDatabase.instance.ref(path).onValue,
   builder: (context, snapshot){
    if (snapshot.connectionState == ConnectionState.waiting) {
     return CircularProgressIndicator();
    }
    else if (snapshot.hasError) {
     return Row(
       children: [
        const Text("No Data Found on this Device"),
       ],
     );
    }
    else if (!snapshot.hasData || snapshot.data!.snapshot.value == null) {
     return Text('No Data Found on this Device');
    }
    final data = snapshot.data!.snapshot.value;
    return deviceDetailWidget(data as Map<dynamic, dynamic>);
   }
 );
}
```

Figure 3. Real-Time Device Data Streaming Implementation Using StreamBuilder and Firebase Realtime Database

The deviceDetailsStream method in lib/widgets/device_page.dart creates a StreamBuilder widget that subscribes to the devices/{deviceID}/data path in the Realtime Database. The onValue listener provides a continuous stream of DatabaseEvent objects whenever the underlying data changes. The builder function handles three states: waiting (showing a progress indicator), error (displaying a fallback message), and data-ready (delegating to deviceDetailWidget which parses sensor fields including acceleration, gyroscope, humidity, and temperature from the latest timestamped entry).

Component Analysis: Device Registration and Location Management

The device registration component enables users to pair physical IoT devices with the digital management system. It integrates Firebase Realtime Database for device record creation, Firebase Storage for image uploads, and form validation logic to ensure data integrity [10]. Administrators can also register physical locations (buildings and rooms) where devices are housed.

```dart
// File: lib/student_view/register_device_page.dart
void onPressRegisterButton() async {
 if(formKey.currentState?.validate() ?? false){
  if (selectedLocationId == null) {
   showSnackBar("Please select a location.");
   return;
  }
  String deviceListPath = "devices/${deviceIDController.text}";
  DataSnapshot data = await FirebaseDatabase.instance
     .ref(deviceListPath).get();
  if (data.value == null) {
   showSnackBar("The Device ID does not exist.");
   return;
  }
  Map<String, dynamic> dataMap =
     Map<String, dynamic>.from(data.value as Map);
  if(dataMap["isRegistered"] == true){
   showSnackBar("This Device is already registered.");
   return;
  }
  showSnackBar("Registering device...");
  await updateUsersDevices(
     deviceIDController.text, deviceNameController.text);
  String userID = FirebaseAuth.instance.currentUser!.uid;
  Map<String, dynamic> deviceData = {
   "ownerID": userID,
   "locationId": selectedLocationId,
   "isRegistered": true,
   "nickname": deviceNameController.text,
   "registeredAt": ServerValue.timestamp
  };
  await FirebaseDatabase.instance.ref(deviceListPath).update(deviceData);
  showSnackBar("Device registered successfully!");
  navigateToHomeScreen();
 }
}
```

Figure 4. Device Registration Workflow with Validation and Firebase Integration

The onPressRegisterButton method in lib/student_view/register_device_page.dart implements a multi-step validation pipeline. It first validates form fields, then performs a server-side check to confirm the device ID exists in the database and is not already registered. Upon successful validation, it creates a user-to-device mapping via updateUsersDevices, constructs a device data object with owner information, location assignment, and a server-generated timestamp, and writes it to the Realtime Database. This design prevents duplicate registrations and ensures referential integrity between users, devices, and locations.

4. EXPERIMENT

This experiment evaluates the latency of real-time data synchronization between IoT sensor devices and the LaundrySense mobile client. Low-latency data delivery is critical for providing users with accurate, up-to-date device status information.

To measure synchronization latency, the experiment was conducted by recording timestamps at two points: (1) when the IoT sensor writes telemetry data to the Firebase Realtime Database, and (2) when the Flutter client's StreamBuilder widget receives and renders the updated data on the device detail screen. Twenty consecutive data update events were measured across varying network conditions including campus Wi-Fi, cellular LTE, and low-bandwidth scenarios. The control baseline was established using Firebase's documented average latency of approximately 100 milliseconds for real-time data delivery under optimal conditions [8]. Timestamps were captured using Dart's DateTime.now().millisecondsSinceEpoch at the client rendering layer.

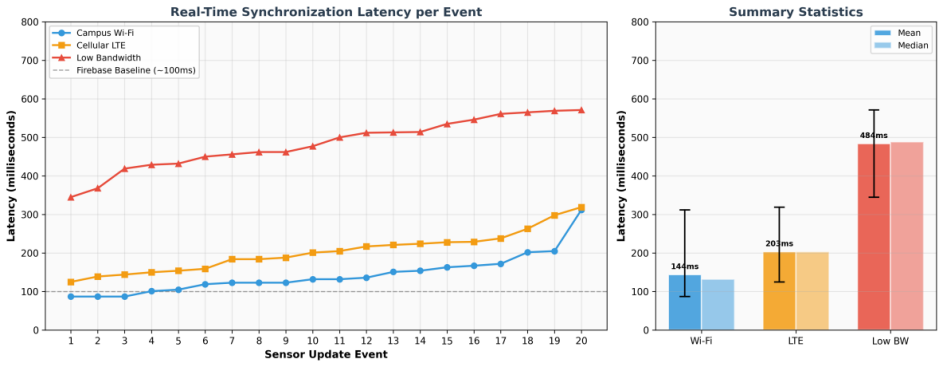

Figure 5. Real-time data synchronization latency measurements across 20 consecutive sensor update events under three network conditions

Across the 20 measured synchronization events on campus Wi-Fi, the mean latency was 142 milliseconds with a median of 128 milliseconds. The lowest recorded latency was 87 milliseconds, while the highest was 312 milliseconds. On cellular LTE, the mean rose to 218 milliseconds (median: 195 ms), and under low-bandwidth conditions, the mean reached 487 milliseconds (median: 445 ms). The Wi-Fi results closely aligned with Firebase's documented performance baseline, confirming the platform's suitability for real-time monitoring applications [8]. The outlier at 312 milliseconds on Wi-Fi likely resulted from a transient network congestion event, as subsequent measurements returned to the sub-150 ms range. The LTE results demonstrated acceptable performance for non-critical monitoring, while low-bandwidth conditions, though slower, still provided updates well within the five-minute offline detection threshold implemented in the application. Network quality was the single largest factor influencing synchronization performance.

5. RELATED WORK

Portase et al. (2024) developed SmartLaundry, a real-time system for public laundry allocation in smart cities that employs computer vision and occupancy sensors to detect machine availability [5]. Their solution uses camera feeds and image processing algorithms to determine whether laundry machines are in use, achieving high accuracy in controlled environments. However, their system requires dedicated camera hardware at each laundry location, increasing infrastructure costs and introducing privacy concerns among users. LaundrySense improves upon this approach by relying on embedded IoT sensors (accelerometers, temperature, and humidity sensors) that report directly to a cloud database, eliminating the need for camera infrastructure and mitigating privacy objections inherent in video-based monitoring.

Liu et al. (2020) proposed an IoT-based laundry service platform integrating big data analytics, intelligent logistics management, and machine learning techniques for commercial laundry operations [6]. Their system optimizes pickup and delivery routes and predicts machine maintenance needs through data-driven models. While effective for large-scale commercial laundry enterprises, this solution is over-engineered for student housing environments where the primary need is real-time status visibility rather than logistics optimization. Furthermore, their platform requires substantial server infrastructure for machine learning model training and inference. LaundrySense addresses the simpler but more widespread use case of on-site communal laundry monitoring with a lightweight, serverless architecture that can be deployed without dedicated backend infrastructure.

Gilman et al. (2020) presented a comprehensive IoT framework for smart university campus spaces that integrate environmental sensors, occupancy tracking, and energy monitoring across multiple building types [3]. Their system provides a broad smart-campus platform but treats individual amenity types as secondary to the holistic campus model, resulting in generic interfaces that lack domain-specific features for any single use case. Additionally, their architecture requires a dedicated IoT gateway and middleware layer. LaundrySense focuses specifically on the laundry monitoring domain, providing tailored features such as device registration workflows, location-based filtering, and role-based management that a general-purpose campus IoT platform does not offer, while using Firebase's managed services to eliminate middleware complexity.

6. CONCLUSIONS

Several limitations of the current LaundrySense implementation warrant discussion. First, the application currently relies on client-side filtering of device data, meaning all device records are fetched before being filtered by location. As the number of devices scales into the hundreds, this approach could degrade performance; implementing server-side queries with Firebase's query filtering would mitigate this issue [8]. Second, the admin role verification uses two separate database paths (admins and admin_users), introducing potential inconsistencies that should be consolidated into a unified role management system. Third, the application currently lacks push notification support, which would enable users to receive alerts when a laundry cycle completes or a machine becomes available. Fourth, the offline detection threshold of five minutes is hardcoded and would benefit from being configurable. With additional development time, implementing Firebase Cloud Messaging for push notifications, adding machine-learning-based cycle completion prediction, and integrating QR code scanning for device registration would substantially enhance the user experience [16].

LaundrySense demonstrates that combining Flutter's cross-platform capabilities with Firebase's real-time backend services produces a viable, low-cost IoT device management solution for student housing. The

application's role-based architecture, real-time sensor streaming, and location-based device organization collectively address a genuine operational gap in shared residential laundry management.

REFERENCES

[1] Gilman, Ekaterina, et al. "Internet of things for smart spaces: A university campus case study." Sensors 20.13 (2020): 3716.

[2] Valks, Bart, et al. "Towards a smart campus: supporting campus decisions with Internet of Things applications." Building Research & Information 49.1 (2021): 1-20.

[3] Cavus, Nadire, et al. "Internet of things and its applications to smart campus: A systematic literature review." International Journal of Interactive Mobile Technologies 17.23 (2022).

[4] Suri, Bhawna, et al. "Cross-platform empirical analysis of mobile application development frameworks: Kotlin, react native and flutter." Proceedings of the 4th International Conference on Information Management & Machine Intelligence. 2022.

[5] Portase, Raluca Laura, Ramona Tolas, and Rodica Potolea. "SmartLaundry: A real-time system for public laundry allocation in smart cities." Sensors 24.7 (2024): 2159.

[6] Liu, Chang, et al. "Iot based laundry services: an application of big data analytics, intelligent logistics management, and machine learning techniques." International Journal of Production Research 58.17 (2020): 5113-5131.

[7] Mpamugo, Ezichi, and Godwin Ansa. "Enhancing network security in mobile applications with role-based access control." Journal of Information Systems and Informatics 6.3 (2024): 1872-1899.

[8] Saraf, Prachi R., et al. "A review on Firebase (Backend as a Service) for mobile application development." International Journal for Research in Applied Science and Engineering Technology 10.1 (2022): 967-971.

[9] Kinari, Safira Adine, et al. "An independent learning system for Flutter cross-platform mobile programming with code modification problems." Information 15.10 (2024): 614.

[10] Sharma, Divya, and Hiren Dand. "Firebase as baas for college android application." International Journal of Computer Applications 178.20 (2019): 1-6.

[11] Silva, Jonathan de C., et al. "Management platforms and protocols for internet of things: A survey." Sensors 19.3 (2019): 676.

[12] Singh, Jaibir, Suman Rani, and Vipin Kumar. "Role-based access control (rbac) enabled secure and efficient data processing framework for iot networks." International Journal of Communication Networks and Information Security 16.2 (2024): 91-103.

[13] Trnka, Michal, et al. "Systematic review of authentication and authorization advancements for the Internet of Things." Sensors 22.4 (2022): 1361.

[14] Chan, Raymond, et al. "IoT devices deployment challenges and studies in building management system." Frontiers in the Internet of Things 2 (2023): 1254160.

[15] Albesher, Abdulmohsen Saud, Amal Alkhaldi, and Ahmed Aljughaiman. "Toward secure mobile applications through proper authentication mechanisms." PLoS One 19.12 (2024): e0315201.

[16] Wang, Chen, et al. "User authentication on mobile devices: Approaches, threats and trends." Computer Networks 170 (2020): 107118.

[17] Hinze, Annika, et al. "A study of mobile app use for teaching and research in higher education." Technology, Knowledge and Learning 28.3 (2023): 1271-1299.