

AttendEase: Fraud-Resistant Attendance Tracking in Higher Education Using Dynamic Tokens and Geofencing

Silvia Gaftandzhieva, Gabriela Lyavova
University of Plovdiv Paisii Hilendarski, Plovdiv, Bulgaria

Abstract—Preparing qualified specialists depends directly on student engagement during the educational process. For this reason, many higher education institutions (HEIs) view attendance as a key element of their educational policy and mandate strict requirements in internal regulatory documents. Traditional student attendance tracking is often time-consuming and susceptible to proxy-scanning fraud. Existing automated solutions, such as RFID or biometric systems, require expensive hardware infrastructure, while static QR-code approaches fail to prevent remote image sharing. This study proposes AttendEase, a cross-platform mobile application that secures and automates attendance management using a Bring Your Own Device (BYOD) paradigm. The system introduces a four-layer validation mechanism that combines time-limited dynamic QR codes (UIDv4), duplicate registration, device-binding verification (SSAID/IDFV) to prevent multi-account logging from a single smartphone, and location validation using a Haversine-based geofencing algorithm. Beyond automated check-in, AttendEase features an analytics module that enables educators and students to generate real-time visualisations and reports with aggregated data. The empirical evaluation at Plovdiv University across 5 courses involving 48 students demonstrates the system's initial feasibility, usability, and potential to streamline attendance tracking, eliminate proxy check-ins, and reduce administrative overhead during lectures. The findings confirm that AttendEase has high potential to support effective attendance monitoring, operational management, and data-driven decision-making in HEIs.

Keywords—Attendance tracking; QR code; geofencing; flutter; cloud systems; mobile security; educational technology, smart campus

I. INTRODUCTION

Higher education plays a critical role in preparing qualified specialists and developing the knowledge and skills that students need to succeed in modern society. The quality of this process depends on various factors, among which the students' active participation holds a particularly important place. A key indicator to measure this participation and engagement is attendance at lectures, seminars, and laboratory sessions, as regular classroom presence directly affects learning outcomes and overall academic performance [1].

Attending lectures allows students to receive systematised information, practical examples, and teacher guidance that are difficult to fully replicate through self-study, ask questions, and receive immediate feedback. Its importance is especially critical during seminars and laboratory sessions for students in

engineering, technical, and natural science programmes, where they apply theoretical knowledge in practice and develop skills in using specialised software, measurement systems, and equipment. Regular attendance also builds discipline, responsibility, and a professional attitude. Therefore, many HEIs consider attendance a vital component of the overall educational process [1] and require a minimum number of attendances for semester validation or exam admission. For instance, according to the Regulations of Plovdiv University Paisii Hilendarski [2], attendance at lectures, seminars, laboratory sessions, and practical exercises is mandatory and monitored by the respective instructors and department heads. This makes accurate and reliable attendance tracking critically important.

On the other hand, attendance information is a valuable tool for teachers and administrative staff. It indicates interest in specific subjects and teaching effectiveness, and can improve educational planning and compliance monitoring. Low attendance can signal organisational problems, insufficient motivation, an unsuitable schedule, or challenges in grasping the material. Therefore, collected attendance data can be used as feedback to improve educational quality.

Despite the key significance of this indicator, the process of registering and tracking attendance is still carried out conventionally in many HEIs - via paper sign-in sheets with manual signatures, oral roll calls, or subsequent manual entry of data into spreadsheets. These methods waste instructional time, increase the risk of technical errors, create difficulties with subsequent information processing, and limit analytical and reporting capabilities. There are also opportunities for abuse, such as marking a student who is actually absent as present.

For this reason, modern automated attendance tracking solutions are needed. With advancements in mobile technologies, cloud services, and wireless communication, new opportunities emerge to modernise attendance tracking through solutions based on QR codes, geofencing, online synchronisation, and rapid report generation.

To overcome the disadvantages of traditional methods, facilitate teachers' work, improve data accuracy, eliminate opportunities for fraud, and give students fast, transparent access to information, this article presents the mobile application AttendEase, which automates student attendance registration. The application allows students to register their presence by scanning a QR code dynamically generated by the

teacher. Before registering attendance, it checks the user's location against defined geographical coordinates, the scanned code's validity, duplicate registration, and the device's identification number used for registration. AttendEase supports user role management and automated, real-world, real-time attendance report generation. The data is stored in a cloud database, ensuring centralised management, reliability, and availability. The developed application is ready for deployment in a real university environment and can be easily extended with additional functionalities.

The study is organised as follows. Section II analyses existing attendance registration approaches. Section III describes the application's conceptual model. Section IV outlines the practical realisation of AttendEase's core functional modules. Section V presents and analyses the results of the application's experimental testing in a real environment. The conclusion outlines a plan for the system's future development.

II. LITERATURE REVIEW

As information technology has developed, various approaches and systems for automated attendance tracking have emerged, using a variety of student identification methods. A common approach to tracking attendance is using paper lists. In this method, the teacher records attendance manually, or students sign themselves on a shared sheet. This method requires no technical tools but consumes significant instructional time in large groups. During the subsequent processing, there is a continuous risk of technical errors, such as marking an absent student present. Another problem is the necessity for physical archiving and document management. As an alternative to paper lists, contact-free Radio Frequency Identification (RFID) access cards have emerged [3-5], in which each student scans their smart card on a specialised reader at the classroom entrance. The database stores the captured information, which allows subsequent processing and analysis. This approach automates registration and saves time, but requires capital investments to purchase and install specialised hardware equipment (RFID readers) and build infrastructure to connect to the information system. The main disadvantage is weak fraud protection, as a student can easily scan absent peers' cards. Biometric systems can also be deployed to register attendance, recognising unique biological characteristics such as fingerprints, facial features, palm geometry, or the iris [6-7]. These systems provide the highest degree of identity validation, practically eliminate proxy registration, and, when configured correctly, offer fast identification. The main disadvantages include high equipment costs, bottlenecks at classroom entrances for large student flows, and strict legal restrictions on storing sensitive personal data under GDPR requirements. Two-dimensional barcodes that can be read with a mobile device's camera, Quick Response (QR) codes, are also used for rapid identification. In this approach, the teacher generates a QR code for a specific class session containing identifying details (e.g., subject, date, time) [8-10]. Students confirm their attendance by scanning the code, after which the data is sent to a server or cloud database for storage. The approach operates entirely on the Bring Your Own Device (BYOD) principle. It does not require the university to purchase specialised hardware, making it an

economically viable solution with easy integration capabilities. A disadvantage of this approach is the lack of protection mechanisms against remote abuse, such as scanning the codes outside the lecture hall.

Based on the reviewed technologies, automated attendance registration solutions for educational institutions have been developed. Kassem et al. [3] develop a classical attendance registration system using personal RFID cards, and Kurniali and Mayliana [11] integrate RFID readers with a web-based information system for data analysis and report generation. Despite automation, both solutions remain vulnerable to card passing and require hardware investments. Other researchers investigate the applicability of biometric systems. Osasuyi et al. [12] present a fingerprint-based solution that guarantees personal authentication, while Smitha et al. [6] design a facial recognition system using static cameras in the room. These systems achieve high security but require substantial computational resources and data security enforcement when processing sensitive personal data. The greatest flexibility is demonstrated by solutions that require mobile device usage. Masalha and Hirzallah [13] prove that scanning a unique QR code for each class drastically reduces check-in times and facilitates information processing for the teacher. The main disadvantage of early QR systems was the possibility of abuse, such as photographing the code and sending it to a student outside the hall. To overcome this disadvantage, Taju et al. [14] and Gangurde et al. [15] propose solutions that combine QR with geolocation technologies to track device location in real time. Muthulakshmi et al. [16] utilise the concept of Geo-Fencing, where a virtual geographical zone is defined around the classroom. This significantly reduces the risk of unlawful attendance registration outside the defined perimeter.

The review indicates that while mobile applications and QR codes are the most practical and cost-effective solutions, they require additional protection mechanisms during implementation. In most existing systems, security against remote abuse (such as using another person's account or scanning outside the hall) remains unsatisfactory. Since local storage or semi-automated processing hinders subsequent analysis and timely report generation, data management presents yet another challenge. Last but not least, the complex user interface of some solutions hinders daily adoption by students and teachers. For these reasons, systems under development must ensure high reliability, ergonomics, and rapid deployment. To achieve this objective, we chose a hybrid approach for the AttendEase mobile application design and development, combining dynamic QR codes with automated checks for code expiration, duplicate registration, hardware identification numbers, and the device's current GPS coordinates relative to the hall's geographical perimeter. A cloud architecture is utilised for data storage, enabling centralised management and rapid real-time report generation. The developed application achieves high reliability, security, and efficiency without requiring HEIs to invest in hardware.

III. CONCEPTUAL MODEL

AttendEase application aims to replace traditional paper-based and manual attendance registration methods in HEIs through mobile technologies, QR codes, and a secure cloud

infrastructure. The functional and non-functional requirements of the system were defined following a detailed analysis of the needs of the primary user groups – teachers and students. Functional requirements determine the core capabilities provided to users, such as user authentication (login) into the system, user role management, schedule and class session management, dynamic generation of unique QR codes for specific sessions, QR code scanning by students, automated attendance registration and validation, ability for manual entry or correction of attendance by a teacher, centralised storage of session and attendance information, visualisation of statistical data and report generation. Non-functional requirements describe the system's quality characteristics, such as secure encryption and storage of user data, protection against unauthorised access and tracking fraud, high processing speed and minimal request processing time, a convenient, responsive, and intuitive user interface, seamless operation across a wide range of mobile devices, and efficient cloud database connectivity and real-time synchronisation.

The workflow follows a strict logical sequence. After successfully logging in, the user is redirected to an interface personalised according to their role. The teacher activates a new session for the current class, and the system displays a unique QR code (e.g., on a multimedia screen in the room). To allow organisational flexibility, the system allows teachers to start sessions and generate QR codes up to 7 days in advance. This allows teachers to pre-print codes on paper or embed them in presentation materials. Students authenticate their attendance by scanning the code through the mobile application. After the system executes the embedded verification algorithm, it records the attendance in the database. This approach increases the process' speed and reliability and reduces errors. If issues occur, the teacher can make manual corrections (adding or removing attendance records).

QR code usage provides a quick and convenient mechanism for session registration. Without additional verification, the QR code could be shared with absent students, or students could hand over their accounts to peers to check in on their behalf. To ensure reliability and eliminate fraud during attendance tracking, AttendEase applies multi-factor verification to prevent check-ins during expired sessions, lateness, duplicate code scanning, remote scanning, or use of someone else's device.

Let S is the set of students, R the set of classrooms, L the set of all class sessions, Q the set of all initiated QR sessions, TC the set of instructors, and T a continuous time space. An ordered 11-tuple describes each active QR session $q_k \in Q$:

$$q_k = \langle id_k, l_j, teach_n, t_{start}, t_{expire}, StatusQR_k, Hash_k, (locLat, locLng), maxDistMet, sessionDate \rangle, \quad (1)$$

where, $StatusQR_k \in \{Active, Expired, Closed\}$, $teach_n \in TC$ is the identification number of the teacher who created the session, $t_{start} \in T$ and $t_{expire} \in T$ are the defined start and end of the session (code validity), $Hash_k = H(id_k, l_j, t_{expire})$ is a cryptographic token, $(locLat, locLng)$ are the coordinates of the classroom's centre, $maxDistMet > 0$ is the permitted deviation radius in meters, and $sessionDate$ is the session date.

An attendance registration request by a student $s_i \in S$ for a given lecture l_k in room $r \in R$ is represented as:

$$Req = \langle s_i, Hash_k, q_k, t_{req}, (lat_s, lon_s), DevID_s \rangle, \quad (2)$$

where, $Hash_k$ is the transmitted token from the scanned QR code, $t_{req} \in T$ is the system time when the request is sent, (lat_s, lon_s) are the student's GPS coordinates at the moment of scanning, and $DevID_s$ is the device's hardware identifier (SSAID/IDFV).

The validation decision for the attendance registration $V(Req) \in \{Approved, Late, Pending_Approval, Rejected\}$ is determined after validating the dynamic token's status and time interval, record uniqueness, location, and the device identifier.

The dynamic QR code is generated $u_s = UUIDv4()$ with generation time t_{gen} . The request is valid if $\Delta t_{token} = t_{req} - t_{gen} \leq \Delta T_{Max}$, where ΔT_{Max} is the allowed grace period delay. The time and validity status predicate $P_{time}(Req)$ verifies if the session is marked as active and if the time window has not expired:

$$P_{time}(Req) = (StatusQR_k = active) \cap (t_{req} \leq t_{expire}) \quad (3)$$

The lateness status relative to the class start t_{start} is determined as:

$$Status_{time} = \begin{cases} Approved, & \text{if } t_{req} - t_{start} \leq 30 \text{ min} \\ Late, & \text{if } t_{req} - t_{start} > 30 \text{ min} \end{cases} \quad (4)$$

Let $A \in S \times L \times Q$ be the set of all successfully confirmed attendance records, and $A = \{ \langle s_i, l_j, q_k, t_{req} \rangle \}$ is an already recorded attendance for student s_i for session q_k . A duplicate entry request will be blocked if such a record already exists for the current session q_k :

$$P_{uniq_{rec}}(Req) = \neg \exists aA: (a.s = s_i \cap a.l = q_k.l_j) \quad (5)$$

The distance $d(s, r)$ between the student (lat_s, lon_s) and the coordinates of the room's centre $(locLat_r, locLon_r)$ is calculated using the Haversine formula:

$$\Delta lat = \frac{\pi}{180} * (locLat_r - lat_s)$$

$$\Delta lon = \frac{\pi}{180} * (locLon_r - lon_s) \quad (6)$$

$$\alpha = \sin^2\left(\frac{\Delta lat}{2}\right) + \cos\left(\frac{\pi}{180} * lat_s\right) * \cos\left(\frac{\pi}{180} * locLat_r\right) * \sin^2\left(\frac{\Delta lon}{2}\right)$$

$$c = 2 * \text{atan2}(\sqrt{a}, \sqrt{1-a})$$

$$d(s, r) = R_E * c, \text{ where } R_E = 6371000 \text{ m}$$

The location validity condition is $d(s, r) \leq R_{allowed}$, where, $R_{allowed} \leq 50$ meters.

Let $DevID_{registered}(s)$ be the pre-registered unique device ID of the student s in the database. The check will return:

$$Match_{device} = \begin{cases} True, & \text{if } DevID_s = DevID_{registered}(s) \\ False, & \text{if } DevID_s \neq DevID_{registered}(s) \end{cases} \quad (7)$$

AttendEase is designed as a client-server system. The mobile application serves as the client-side interface through which users interact with the system. The server side is based on cloud infrastructure that provides Backend-as-a-Service (BaaS) managed components for authentication, distributed data storage, real-time synchronisation, and user session management. To achieve a loosely coupled architecture and high maintainability, AttendEase is divided into 4 core functional modules:

- Authentication Module – manages user identification, access rights verification, and automatic redirection to the home screen based on the defined role.
- Session Management and QR Code Generation Module – maintains information regarding subjects, classes, teachers, student groups, and semesters, and manages the code generation process.
- Attendance Registration and Validation Module – enables scanning and processing of QR codes, executing automated checks for code validity, preventing code reuse, remote proxy check-ins (comparing the device's current coordinates with the classroom's designated location), and account handovers (checking if the registration device matches the pre-registered one).
- Analytical Module – provides instructors and students with structured access to data on conducted sessions, attendance counts, absences, lateness, and percentage activity, with information fetched from the database in real time and visualised via lists, charts, and/or aggregated metrics.

The AttendEase architecture is designed for scalability. Cloud service usage facilitates integration with university information systems and serving a large number of users without significant changes to the application structure.

IV. SOFTWARE REALISATION

Designing a modern multi-platform application requires selecting software technologies that guarantee high performance and scalability. This section presents the system's structure, interface solutions, user roles, database organisation, and the realisation of core functional modules.

A. Development Technologies

AttendEase is implemented using technologies for creating a cross-platform solution for Android and iOS. The Flutter framework was chosen as the primary development environment, which ensures the construction of multi-platform applications from a single codebase with a modern appearance and an excellent user experience. Dart was selected as the programming language because of its high performance and convenient tools for user interface design. A major advantage of Flutter is the ability to develop apps for Android and iOS simultaneously, which enhances applicability, reduces development costs, and simplifies maintenance. Additionally,

the framework provides ready-made integration tools for cloud services. It has a rich ecosystem of external libraries (including QR code processing, coordinate extraction, permission checks, and distance calculations), an active user community, and exhaustive documentation.

For data storage and real-time synchronisation, the Google Firebase platform was used, including specialised services for authentication, data storage, and push notifications.

The document-oriented NoSQL database Cloud Firestore was selected as the database management system. The main argument for this choice is its native support for real-time automatic synchronisation. When a QR code is scanned and attendance is recorded, the application interface updates automatically, enabling attendance monitoring during an active QR session. Data protection is secured at the cloud level through Firestore Security Rules, which define read and write permissions according to the user's role.

The system's core business logic is realised through dynamic QR code generation and scanning technologies, alongside location tracking services via mobile devices' built-in GPS modules. Combining these two technologies eliminates opportunities for fraud involving capturing and forwarding the code image to absent students. The main reasons for choosing QR technology are its flexibility in encoding heterogeneous content (text, numerical values, identification tokens, and coordinates), the presence of positioning markers, and a built-in Reed-Solomon error correction algorithm. This allows reliable decoding even under partial smudging, poor angles, or low projector resolution in the lecture hall. Process security is enhanced by deploying dynamic QR codes with narrow time windows and automatic disabling after a specified number of minutes. As an additional security mechanism, AttendEase utilises geolocation via the devices' embedded Fused Location Services, combining data from GPS, Wi-Fi networks, cellular towers, and other sensors for accurate positioning. The system calculates the distance between the student's position and the pre-configured classroom coordinates using the Haversine formula. If the calculated distance exceeds the defined threshold (e.g., 50 meters), the system rejects the transaction. To comply with GDPR, the application processes location data solely in the device's local memory during scanning, ensuring no geographic coordinates are permanently stored in the cloud.

The development and testing toolkit includes the Visual Studio Code integrated development environment (IDE), Flutter SDK, Firebase Console for cloud infrastructure management, external libraries, Git and GitHub (for decentralised version control, change tracking, and archive repository management), software emulators, and physical Android and iOS mobile devices for testing AttendEase functionalities.

B. Project Structure

The application utilises a mixed architectural model that combines layer separation with grouping user screens per role. At a conceptual level, the system is split into three main layers:

- Presentation Layer – encompasses the user interfaces (UI screens) implemented in the 'lib/screens' directory and visual components in the 'lib/widgets/' directory.

- Business Logic Layer (Service Layer) – encapsulates the application's business logic ('lib/services/' directory) and contains classes for communicating with the REST API and the Firebase cloud platform, session management, geolocation handling, and system notifications.
- Data Layer – defines the structure of core objects in the system ('lib/models/' directory) - users, subjects, classes, attendances, and QR sessions.

In the root 'lib/' directory, the 'main.dart' file serves as the entry point for the software lifecycle, initialising cloud services and global states. Helper components (validators, extended date and time formatters) and visual themes are distributed across the 'utils/', 'theme/', and 'l10n/' subdirectories (for interface internationalisation and localisation).

Fig. 1 presents the AttendEase architecture.

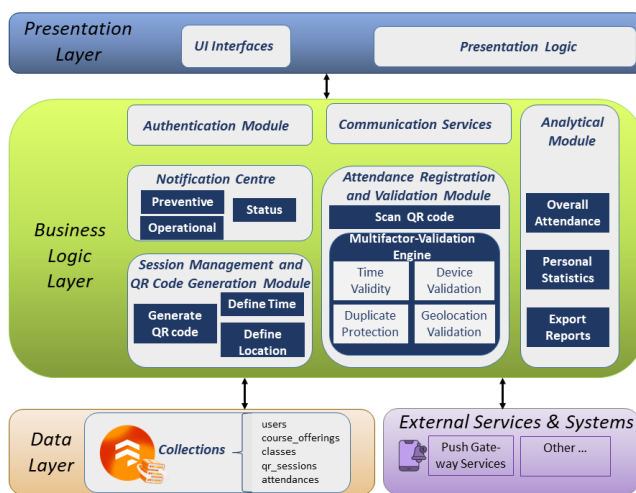


Fig. 1. Architecture.

AttendEase realises state management through a hybrid approach. The Provider architectural pattern, combined with ChangeNotifier, is used for data that affects the entire application (authentication status, current user profile, and role). Ephemeral states of individual screens (loading indicators, selected tab, QR scanner activity) are managed via the standard setState mechanism, optimising performance and resource consumption. The architecture minimises coupling between layers - components in the presentation layer do not communicate directly with the database. Communication is conducted by invoking methods from the service layer. For single transactions, asynchronous requests (Future/Await) are utilised, while real-time changes are tracked using data streams through the StreamBuilder component and Firestore snapshots. AttendEase manages navigation between screens by declarative access control mechanisms that prevent unauthorised student access to administrative panels.

Although the project does not implement Clean Architecture in full, the chosen mixed model offers an optimal balance of readability, development speed, and maintainability. The main advantages of the proposed architecture are the clear separation between UI, business logic, and data, and the

isolation of Firebase communications into independent services, which allows easy future migration to another backend infrastructure or local university information systems without modifications to the presentation layer.

C. User Roles and Access Model

The AttendEase architecture utilises a role-based access control model. This approach enhances personal data security, optimises the UI, and eliminates the risk of unauthorised actions. After successful authentication, the system loads the user token from Cloud Firestore and dynamically configures the accessible functionality.

Fig. 2 presents a Use Case diagram of the AttendEase mobile application.

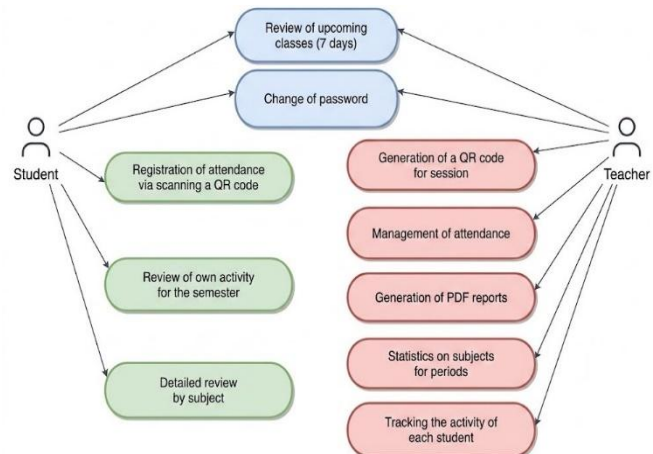


Fig. 2. Use case diagram of AttendEase.

The Student user role is designed for learners whose interaction with the system is restricted to registering attendance and reviewing their own attendance history. Upon logging into the system, the student has access to:

- Home screen – displays an academic schedule with planned sessions for 7 days;
- Attendance registration module – provides quick access to the mobile device's camera for scanning generated dynamic codes and automated feedback regarding request outcomes (successfully recorded attendance or rejection categorised by reasons: expired session, invalid code, lack of location permission, etc.);
- Analytical module – provides access to data on registered attendances and percentage ratios across subjects.

The Teacher user role is intended for individuals who conduct class sessions and manage attendance tracking. Upon logging into the system, the instructor receives access to an interface for:

- QR session management – enables selecting a specific class and generating a unique QR code with a validity period and location coordinates corresponding to the hall where the session takes place;

- Real-time student status monitoring (present, late, absent);
- Manual status correction during extraordinary situations or technical issues with a student's device;
- Detailed attendance statistics review at the student, group, or cohort level with capabilities to export into formats suitable for sharing or archiving (PDF/CSV).

D. Database

The database stores the information in logical collections. Table I describes the core collections, their attributes, and functional purposes.

TABLE I. STRUCTURE OF CORE COLLECTIONS IN THE DATABASE

Collection	Primary Fields (Attributes)	Functional Purpose
users	id, role, email, firstName, lastName	Stores information about system users
faculties	id, name, code	Stores information about faculties
specialties	id, name, code, degree	Stores information about study programmes
cohorts	id, specialtyId, startYear, endYear	Stores information about student streams/cohorts
groups	id, name, specialtyId, cohortId	Stores information about student groups
semesters	id, name, term, academicYear	Stores information about academic semesters
subjects	id, name, code	Stores information about subjects
course_offerings	id, subjectId, semesterId, cohortIds	Links subjects to a specific semester and student stream
classes	id, offeringId, teacherId, room, dayOfWeek	Stores information about individual class sessions
qr_sessions	id, classId, teacherId, token, expiresAt	Stores active QR sessions for attendance registration
attendances	id, classId, studentId, status, scannedAt	Stores information about registered attendances

Although Cloud Firestore is a NoSQL database, the data model implements relational logic through document references (Document IDs). This prevents data duplication across different locations and maintains connection integrity. To improve performance when processing large volumes of data, indexes have been configured to facilitate query execution, such as filtering attendances by a specific subject for a stream of students. The system ensures data-level security through declarative Firestore security rules. Students have read permissions only for their own attendance records, and only users with a teacher role can modify and write to qr_sessions and attendances. Thanks to built-in WebSocket support, the system monitors collection changes in real time via client interfaces and reactive components (StreamBuilder). If necessary, the designed database can be expanded, for instance, by adding new collections and attributes to store notifications and archived reports, or for integration with university systems.

E. Authentication Module

The authentication module provides reliable user identity validation. It implements the authentication process by integrating Firebase Authentication, eliminating the need for local password hashing and storage. Upon launching, the

application checks for an active user session. If the token is active, the system redirects the user to the corresponding home screen. Otherwise, a screen with a login form appears, requiring the user to enter an email address and password.

Code Snippet 1 shows the logical sequence for authentication and user status verification. After successful matching of the entered credentials in the Firebase Authentication cloud directory, the system extracts the subject's unique identifier (uid). Then, it requests the users collection in Cloud Firestore to validate the profile status. If the document does not exist or the isActive attribute is false, signIn() terminates the session and sends an exception to the presentation layer. Upon successful verification, notifyListeners() updates the global Provider state. After successful authentication, the system loads the corresponding home screen depending on the user's role.

Code Snippet 1. User authentication via Firebase Authentication

```
final userCredential = await _auth.signInWithEmail  
    AndPassword(email:email,password: password.);  
final uid = userCredential.user!.uid;  
final user = await getUserById(uid);  
if (user == null) {await _auth.signOut();  
    throw Exception('User not found in database.');}  
if (!user.isActive) {await _auth.signOut();  
    throw Exception('Account is deactivated.');}  
_sessionUser = user;  
_ready = true;  
notifyListeners();
```

The application uses centralised session management via Firebase Authentication, enabling automatic session termination upon logout, protecting sensitive information, and ensuring secure credential storage. As an additional security measure, upon initial login with a system-generated temporary password, the system requires the user to change the password before granting access to functionalities. The modular structure permits seamless integration of additional security methods, such as password recovery mechanisms, multi-factor authentication, and biometric recognition via device hardware modules.

F. Session Management and QR Code Generation Module

Initialising an attendance session is performed by users with a Teacher role in 3 steps: reviewing the academic schedule and selecting a specific class session, defining geographical coordinates for conducting the session by using the mobile device's current location via the built-in GPS module or selecting classroom coordinates from an interactive map, and generating and visualising a QR code.

The system uses indirect encoding to enhance security. Instead of embedding explicit metadata about the subjects, study programme, or lecturer in the QR code, which could be easily decoded and shared, the system encodes a cryptographically secure, compact identification token in the matrix that serves as a secure key (reference) to the corresponding document in the cloud database. This minimises the risk of content counterfeiting or local manipulation of client requests. Code Snippet 2 shows how the application creates a new session. To generate a unique token for a class session,

UUIDv4 is used, and the result is parsed into a compact string to optimise optical recognition. The `QRSession` class object stores the class identifier (`classId`), teacher identifier (`teacherId`), creation and expiration timestamps (`createdAt`, `expiresAt`), current status, geographical coordinates (`locLat`, `locLng`), and the allowed radius fixed at 50 meters (`maxDistMet`). The data is written into the `qr_sessions` collection.

Code Snippet 2. QR session initialisation

```
final token = const Uuid().v4().split('-').first;
final session = QRSession( id: const Uuid().v4(), classId:
    classItem.id,teacherId: teacherId,createdAt:
    now,expiresAt:expiresAt, status: QRSessionStatus.active, token:
token, locLat:
    locLat, locLng:locLng, maxDistMet:50,
    sessionDate: formatSessionDate(sessionDate));
await fs.createQRSession(session);
```

The QR code is displayed on the teacher's screen in high resolution. The interface provides additional session and location metadata, and options to export or download the graphic file (e.g., for visualisation on an external screen, embedding in presentation materials, or external systems). Each generated code has a strictly defined lifecycle, bound to the duration of the academic class hour.

The embedded QR code generation module in *AttendEase* provides a high-performance, cost-effective, and secure mechanism to identify academic sessions. Its software architecture allows easy future scaling and full integration with external university information systems.

G. Attendance Registration and Validation Module

Students can scan a QR code in the application. After logging in, the student can access the scanning function via a dedicated button on the home screen. After the student activates this function, the application opens the mobile device's camera and initiates a real-time QR code recognition algorithm. Code Snippet 3 shows a fragment of the QR code recognition implementation. After successful scanning, the code value is extracted, representing an active QR session token. The token is then returned to the main screen for subsequent validation and attendance logging.

Code Snippet 3. Scanning a QR code via the device camera

```
MobileScanner(controller: _controller, onDetect: (capture){
    if (!_isHandlingDetection) return;
    final barcode = capture.barcodes.isNotEmpty?
        capture.barcodes.first: null;
    final value = barcode?.rawValue;
    if (value == null || value.isEmpty) return;
    _isHandlingDetection = true;
    Navigator.pop(context, value);},)
```

The camera processes and displays the image in real time, showing a targeting frame or indicator for proper QR code positioning. After successful recognition, the application extracts the token and sends a request to the database to check the corresponding active QR session. Before recording attendance as a new document in the `attendances`

collection, the system performs four checks to guarantee tracking reliability:

- Time validity – verifies whether the QR session exists and has an Active status and whether the current system time is less than `expiresAt`;
- Duplicate protection – verifies if an entry already exists for the same `studentId` and `classId` on the same date;
- Geolocation validation – checks whether the scanning device is within range of the location specified during the QR code creation by calculating the distance to the reference point;
- Device validation – checks for a match between the hardware identifier and that of the student's initially registered device.

The system applies an automatic logical constraint: if a student scans the code more than 30 minutes after the official start of the class, the transaction completes successfully, but the attendance status attribute in the `attendances` collection is logged as `Late`. After the class ends or after a specified time window, the session automatically transitions to `inactive` status, preventing late tracking attempts.

When the system receives a scanned token from a client device, it uses Firestore Security Rules and the corresponding service classes to verify whether the associated session in `qr_sessions` is active and whether the current system time does not exceed `expiresAt`. At the same time, it executes a verification query to prevent duplicate records for the same student in the same class.

Verifying users' locations via geolocation restricts off-site attendance registration and prevents non-present students from fraudulently logging in. The location verification procedure utilises Fused Location Services, which pull data from GPS modules, cellular networks, and Wi-Fi access points. When scanning is initiated, the client application requests permission to access geolocation services. Upon receiving permission, the system extracts the mobile device's current geographical coordinates (latitude and longitude) and calculates the distance `d` to the defined reference point using the Haversine formula. The `Geolocator.distanceBetween` method accepts the student's current (`lat`, `lng`) and the reference coordinates (`locLat`, `locLng`) from the current session as parameters. When the distance variable value exceeds the pre-configured limit in the `session.maxDistanceMeters`, the system generates a location validation exception and changes the attendance status to `Rejected`. In the current implementation, the system enforces a 50-meter geofencing threshold around the lecture hall. This value is good for indoor university spaces, as it allows some tolerance for GPS hardware deviations in buildings with massive reinforced concrete structures while preserving verification reliability.

Code Snippet 4. Calculating distance between current position and class location:

```
double _distanceToSession(QRSession session, double lat, double lng)
```

```
{return Geolocator.distanceBetween(lat, lng,  
    session.locLat, session.locLng, );}  
    final distance = _distanceToSession(session, studentPos.latitude,  
studentPos.longitude);  
    if (distance > session.maxDistanceMeters)  
        { // attendance is rejected }
```

The application design provides facilities to handle exceptions. If location access is denied, the application prompts the user to enable the permission. Upon a second refusal, it automatically redirects the user to the device settings. If the system can't determine precise coordinates due to a lack of GPS signal, indoor environments, or technical limitations, it displays an error message, and it does not complete registration until it retrieves valid data. To ensure information security and personal data protection (in compliance with GDPR directives), the application fetches and uses the user's location only within the millisecond timeframe in which it processes the scanned token, and does not perform continuous background tracking.

Verifying the device used for registration enhances the reliability of the automated tracking process by limiting the voluntary handover of user accounts. With standard email and password authentication systems, an absent student can give their login credentials to a colleague, who can then log fictitious attendance on their behalf. To mitigate this risk, AttendEase implements an additional check via the Device Identifier (Device ID). The procedure for binding a student's profile to the device used is initiated automatically during the first successful authentication in the application. The system extracts the device's unique identifier and writes it to the corresponding document in the `users` collection in Cloud Firestore. Code Snippet 5 presents how the application extracts the device identifier depending on the operating system. For Android, it fetches the Android ID (SSAID), which is unique for each combination of application, device, and user profile. For iOS, it uses `identifierForVendor` (IDFV) to ensure a constant value for apps from the same hardware vendor. During each subsequent request to scan a QR code and record attendance, the application extracts the current device identifier and compares it with the stored value. If the two values match, the process continues normally. If the system detects a mismatch, it updates the record state in the `attendances` collection to "Pending Approval". Then it displays a message on the student's mobile device informing them that manual verification is needed.

Code Snippet 5. Fetching device identifier

```
Future<String?> getDeviceId() async {try {  
    if (Platform.isAndroid) {  
        AndroidDeviceInfo androidInfo=await _deviceInfo.  
            androidInfo; return androidInfo.id;}  
    else if (Platform.isIOS) {IosDeviceInfo iosInfo = await  
        _deviceInfo.iosInfo; return iosInfo.identifierForVendor;}  
    return null;  
} catch (e) {return null;}}
```

When the student uses a different device, the system does not block registration completely. The teacher's interface captures changes in real time via a reactive Firestore Stream and displays a notification about a registration attempt from a

different device. It provides lecturers with detailed information, allowing them to manually confirm or cancel the attendance after a physical verification in the auditorium. Upon confirming attendance, the system offers the option to update the registered device ID number. This capability to change devices is particularly useful if a student has replaced their smartphone, is temporarily using another device, or has reinstalled the application.

To ensure compliance with GDPR directives, the module does not collect or process sensitive personal data (such as IMEI numbers, phone numbers, or biometrics) and stores only anonymised strings. Combining device validation with QR codes and geolocation checks creates a multi-layered defence mechanism. The QR code validates participation in a specific class session, the Device ID check confirms the use of the registered hardware, and the location check proves physical presence within the designated zone. After the checks pass, the system displays a confirmation of successful registration on the client device, or a corresponding descriptive message in the event of failure or delay. Due to cloud data synchronisation within a few hundred milliseconds, the teacher can monitor attendance on their device screen in real time.

H. Analytical Module

Accumulated attendance data has real value only when it can be analysed and used for decision-making. Deploying an analytical module provides numerous practical benefits - teachers gain fast access to summarised information on student attendance, students have transparent access to their participation records, and academic leadership can use accumulated data to analyse activity patterns. The application sends a request to retrieve attendance information by student, subject, period, or group, performs statistical calculations, and visualises the summarised data as numerical indicators and via graphics to facilitate analysis. After a new attendance registration, data updates dynamically without requiring a manual screen reload.

Teachers can access data for all subjects they teach, select a specific one, and receive attendance data for all enrolled students. The system allows filtering by time periods (week, month, semester) and can visualise summarised data for attendance statuses, the attendance ratio for the selected period in a pie chart, and individual student attendance percentages for the chosen period in a bar chart. The module allows the teacher to export a report for each subject containing summarised attendance information for each student throughout the semester, to exercise precise control over participation in the educational process and semester validation.

Students can review their attendance and absence data over the semester, along with summarised attendance and absence figures across the subjects, enabling them to take timely measures to increase attendance to secure semester validation at the end of the term. Beyond aggregate values, the student can find detailed information for individual classes. For each session, AttendEase displays the date and time, subject, and attendance status (e.g., present, late, absent, or rejected due to location).

Thus, the embedded reporting and statistics module transforms AttendEase from a pure attendance logging system into a comprehensive tool for analysing and managing educational activity. Through clear visualisation and fast access to information, the system successfully supports and optimises activities for both instructors and students.

1. Interface and User Experience

The design and implementation of user interfaces (UI/UX) in AttendEase adhere to modern standards of ergonomics, intuitiveness, and responsiveness across varying mobile device screen dimensions. Architectural separation of the presentation layer ensures that visual components remain independent of business logic while providing reliable real-time user feedback.

The primary objective in creating the interface was to achieve maximum clarity and provide rapid access to core functionalities. Although the application is developed for a live educational environment, students must be able to scan the generated QR code immediately, and instructors must be able to initiate a lecture session or review/edit attendance with minimal interactions. The interface is built with Flutter and strictly adheres to modern Material Design specifications. Modern visual components, cards for logical data grouping, rounded buttons, smooth screen transitions, and a consistent colour scheme are utilised. The highly legible font used ensures good visibility across various screen resolutions. Font sizes and element spacing were selected to make information easy to consume without visually overloading the user. To adapt to external lighting conditions and user preferences, the application supports two themes: a light theme that provides high contrast and optimal legibility in daylight, and a dark theme that reduces ocular strain and fatigue in low-light environments. After logging in, the user is directed to the appropriate home screen. Students can register attendance by scanning QR codes, review the upcoming sessions schedule, and view personal attendance statistics. Teachers have access to an administrative panel that allows them to initialise and manage QR sessions, review attendance sheets, visualise attendance-monitoring charts, and export reports.

To minimise user errors and build trust in the system, it displays detailed context-specific error messages. After successful attendance logging, the student receives a confirmation. If an error occurs (e.g., invalid location or expired time window), AttendEase displays specific information about the problem.

In the user interface implementation, visual elements are leveraged for rapid information consumption - colour indicators, icons, progress bars, and graphs. For instance, low attendance is denoted with warning colours (red/orange), while high activity is highlighted in green.

From a technical standpoint, AttendEase uses the standard Flutter sizing mechanisms to ensure interface responsiveness. Elements reorganise flexibly using scrollable views, expanding containers, and adaptive layouts, ensuring proper visualisation across diverse mobile devices. The application is optimised for portrait mode operation. To enhance user experience, AttendEase minimises screen load times, section transitions,

and core action processing times to ensure high system performance.

To improve user experience and facilitate educational organisation, a specialised notification and reminder subsystem was integrated into the application, optimising academic processes and ensuring a reliable, two-way informational stream between the backend infrastructure and end users. Through this subsystem, students and teachers can receive timely information regarding upcoming classes, important events, and potential issues relating to attendance logging. AttendEase manages push notifications through the Firebase Cloud Messaging (FCM) platform. Push notifications can notify users about critical events related to attendance tracking. During initial user authentication, the application requests permission to access the operating system's messaging subsystem and generates a unique FCM token for that specific client device.

Code Snippet 7. Syncing FCM token for push notifications

```
Future<void> syncTokenForUser(String userId) async {  
    final settings = await _messaging.requestPermission( alert: true, badge:  
true, sound: true, );  
    if (settings.authorizationStatus == AuthorizationStatus.denied) return;  
    final token = await _messaging.getToken();  
    if (token == null || token.isEmpty) return;  
    await _saveTokenForUser(userId, token);  
}
```

Code Snippet 7 illustrates how the application requests notification permissions and retrieves an FCM token for a specific device. The application saves the retrieved token in the database, enabling push notifications to be directed to the corresponding user. To ensure data security and prevent messages from being sent to unused hardware, logging out or switching devices automatically deactivates or updates the token. For automated monitoring via Firebase Cloud Functions, a cloud function was developed to track database transactions in real time. After it detects an attendance registration attempt outside the pre-configured radius, the system can automatically dispatch a push notification to the instructor, allowing a timely response and physical inspection. Unlike push notifications, local reminders are scheduled and generated autonomously at the operating system level, without requiring continuous internet connectivity. These notifications alert students and teachers to upcoming classes and reduce the probability of missing sessions. Upon successfully loading the user profile, the application fetches the current upcoming class schedule and schedules corresponding local alarms. In the current software version, the notification subsystem initiates two consecutive events for each class session: a preventive notification generated 15 minutes before the fixed start time and an operational notification generated precisely at the start of the session (see Fig. 3). The algorithm embeds native support for devices' local time zones, guaranteeing precise alignment of notifications with the actual academic schedule regardless of the user's geographical location.

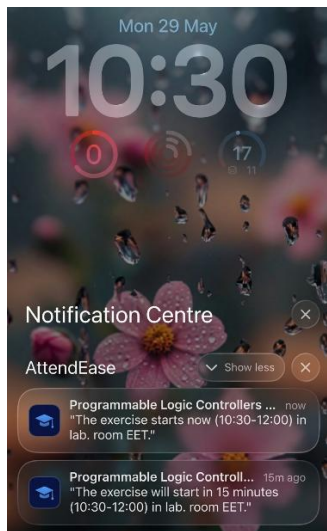


Fig. 3. Showing AttendEase alerts on lock screen.

V. EXPERIMENTAL TESTING AND VALIDATION

To evaluate how effectively AttendEase meets teachers' and students' needs and operates reliably under real-world conditions, AttendEase has been tested at the University of Plovdiv Paisii Hilendarski. The experimental testing involved 1 teacher and 48 students utilising the application on their personal mobile devices under conditions simulating a live academic environment (immediately before and during class hours). Participants downloaded and installed the mobile app via the Google Play Store/Apple App Store and provided access permissions for the device camera and location tracking during operation. All participants received unique login credentials to access the system. To verify application compatibility and stability, testing was performed across physical iOS and Android devices, validating core operation scenarios: authentication processes for varying roles, including redirection to corresponding panels, enforced mandatory initial password changes, and accuracy of push notifications for upcoming classes (sent 15 minutes prior and at the exact start hour).

The testing process simulated real-world stages of logging class attendance - session preparation and launch, scanning, validation, and attendance data analysis. For experimental purposes, attendance tracking simulations were conducted for 5 subjects: Energy Management and Electrical Apparatus, Internet of Things, Programmable Logic Controllers, Optical Communication Technologies, and C# Programming Part 2.

Testing began with the teacher logging in to the application's home screen, which verified correct extraction of the current date, upcoming daily classes, and the planned schedule. Fig. 4 presents the process of initiating a QR session for "Programmable Logic Controllers". The teacher selected a specific class (10:30-12:00 on June 15) from the subjects list (Screen 1), set a geographic reference point using the location setting (EET Lab) and the map (Screen 2). After the teacher tapped the "Start QR" button, AttendEase generated a unique identifier and displayed a temporary QR code (Screen 3) on the device screen, valid for the duration of the class.

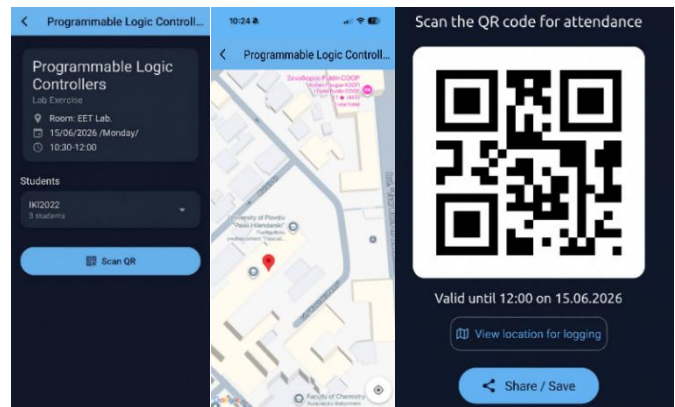


Fig. 4. Creating a QR session steps.

After the QR session activation, users with a Student role logged in to the application, reviewed the daily schedule, activated the scanning feature, and scanned the code displayed by the teacher using their mobile devices' cameras. The application recognised the code, extracted the required data to initiate registration, and executed automated background verifications to check session activity, time windows, uniqueness, geolocation constraints around the hall, and hardware ID matches against the initial profile bind. The system was tested against all forms of potential fraud. In every instance, the application executed the designated automated checks, responded according to embedded business logic rules, successfully identified attempts for registration abuse, marked records as "potentially suspicious" on the teacher's panel, and displayed appropriate alert notifications to students. AttendEase accurately determined and displayed student statuses.

During attendance registration, the teacher was monitoring incoming entries in real time on the tracking screen. All incoming check-ins were correctly registered as attended (highlighted in green), late (highlighted in orange), pending confirmation (highlighted in yellow), or no data for incomplete scanning processes (highlighted in grey), and then written to the cloud database (Fig. 5). To assess the system's fraud-detection capabilities in edge cases, we simulated a scenario where a physically present student attempted to log attendance using a peer's hardware. The teacher successfully performed a manual status override through the control panel and updated the device data (Fig. 5, Screens 3 and 4). While the anti-fraud testing evaluated basic location mismatches and device handover scenarios, this phase did not systematically benchmark explicit countermeasures against advanced location spoofing (such as Android mock-GPS tools or root-level location masking). Addressing hardware-level or application-level mock-location detection represents an essential vector for future security enhancements.

In the final stage, users tested the analytical module reporting capabilities. The teacher accessed analytical dashboards to review aggregate attendance data across subjects, track overall student attendance percentages, and identify learners with low activity levels.

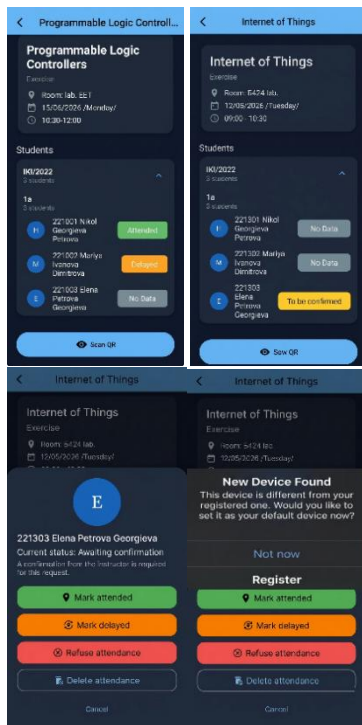


Fig. 5. Tracking the registration process details.

Fig. 6 presents visual dashboard screens with aggregate attendance data for the "Internet of Things" (Screens 1 and 2) and "Programmable Logic Controllers" (Screens 3 and 4) subjects over a selected period. For the chosen timeframe (18.05.2026 - 24.05.2026), 3 students were present, and 3 students were absent in the "Internet of Things" sessions. Aggregate semester data indicated that 3 students attended 67% of the conducted sessions, while the remaining 3 attended only 33%. During May 2026, 22 students (46% of the cohort) attended the "Programmable Logic Controllers" classes. Visualised data signalled that the teacher needs to deploy measures to increase engagement, particularly for students with attendance under 40% (highlighted in red). A comparative time trial showed that taking attendance for a group of 22 students in the "Programmable Logic Controllers" classes took an average of 3.5 minutes using traditional manual roll call. In contrast, AttendEase reduced this duration to 25 seconds in total, as students validate their presence concurrently. This yields an approximate 88% reduction in administrative overhead per seminar session.

A comprehensive summary of present logs (green), lateness (yellow), absences (grey), and rejected entries (red) is accessible for each subject (per student). Fig. 7 displays aggregate attendance data for the "Internet of Things" (Screen 1) and a detailed exported report (Screen 2). The data indicates, for example, that student number 221301 attended 3 sessions and was absent for 3 sessions.

Students tested the personal statistics section, tracking their attendance across enrolled subjects, total sessions conducted, successfully tracked attendances, and calculated attendance percentages for each course. In all cases, the system correctly computed attendance rates and systematised the information. Fig. 8 displays screens with aggregate summaries of

attendances, absences, lateness, and rejected registrations for a student across all semester courses (Screen 1) and detailed historical status data for "Internet of Things" sessions (Screen 2).

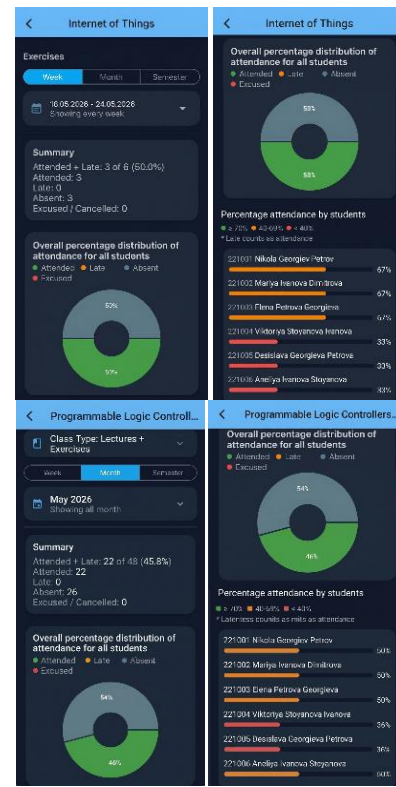


Fig. 6. Analytical panel visualisations.

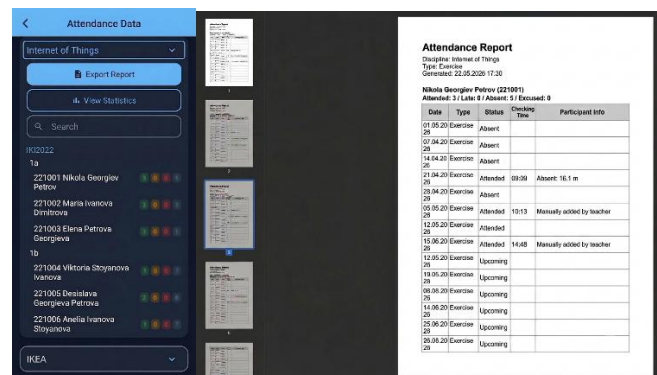


Fig. 7. Analytical panel: Aggregate data export example.

Tests proved that the application interface enables action execution in minimal steps, making it suitable for live classroom deployment where attendance tracking must be completed rapidly and without friction right before class starts. All automated verifications (time, geolocation, and Device ID) were triggered flawlessly, capturing and flagging all abuse attempts. Attendance data visualisation substantially facilitates analysis and identification of low-activity students. No critical errors compromising system functionality were identified during experimental testing. The minor interface inconsistencies identified were promptly resolved during iterative development.

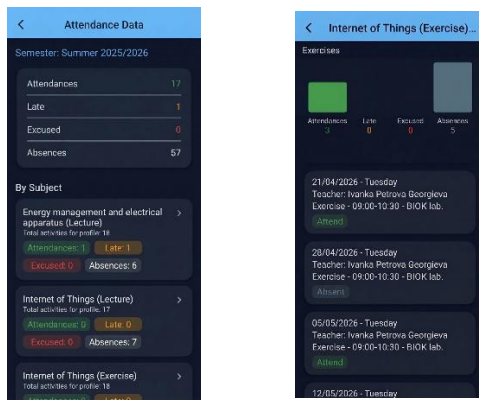


Fig. 8. Analytical panel: student views.

This experimental testing focuses on validating AttendEase's integrated four-layer verification end-to-end rather than comparative benchmarking. Empirical measurement of the incremental fraud-resistance provided specifically by the device-binding layer remains a topic for future comparative testing.

VI. CONCLUSION

The empirical observations from the prototype deployment align with findings in prior literature regarding the efficiency of QR-based attendance [13]. While earlier work cited check-in speed improvements, AttendEase prioritises a security balance, and the extra latency introduced by cloud-side multi-factor checks (time, boundary, device binding) trades off raw check-in speed for fully automated verification.

Functional testing across physical Android and iOS hardware confirms AttendEase's reliability and practical applicability as an effective automated student attendance tracking system in a real university environment. Its UI is engineered with a focus on ergonomics, operational clarity, and process efficiency. By applying modern design concepts, adaptive role-based navigation, and software optimisation, the application provides a high-quality user experience tailored to the academic community's specific needs.

AttendEase is a comprehensive mobile solution for automated student attendance tracking. Driven by its modular architecture, embedded verification mechanisms, and modern development technologies, the system offers a reliable, efficient instrument for optimising administrative processes in HEIs. AttendEase significantly shortens roll call time at the start of classes, eliminates tracking fraud and the risk of human error, ensures high data fidelity, facilitates daily lesson organisation, reduces administrative overhead for tracking and analysis, and fosters active student participation in the educational process.

Although the implemented mechanism for location verification and dynamic QR code scanning demonstrated a 100% success rate in simulated scenarios, its practical application in a real university environment presents certain technical limitations. The first one is related to geolocation and satellite signal drift (GPS drift). Receiving a GPS signal inside massive reinforced concrete buildings often causes interference, leading mobile devices to exceed the set 50-metre

radius and potentially creating false negatives. In the current version, this issue is addressed by allowing teachers to register manually. To achieve full reliability, future versions of AttendEase plan to implement a hybrid geo-validation mechanism that combines GPS data with Bluetooth Low Energy (BLE) Beacons and checks for connectivity to the university network when the satellite signal is weak. This will ensure high precision and micro-location accuracy down to a specific desk in the lecture hall. The second challenge stems from scalability during peak loads and poor network connections. During lectures with a large number of participants (150–200 students), students will scan dynamic QR codes simultaneously within a 10–15 second timeframe. Sending dozens of requests to Google Cloud Firestore simultaneously can lead to temporary delays, request timeouts, or exceeding the per-second request limit. To address this issue, in addition to the currently implemented offline caching, plans include introducing a local queue for asynchronous data submission and using a message broker to handle peak traffic. Additionally, a future version will enable generating a temporary unique offline code based on the student's Device ID, which the lecturer can scan or enter manually at the end of class when connectivity is absent.

Another methodological limitation is the sample size: 1 teacher and 48 students from one institution. Consequently, while the results demonstrate technical feasibility, broader deployment across diverse institutional contexts is required to establish generalisability.

The modular architecture of AttendEase ensures high scalability and extensibility, with future expansion vectors including:

- Developing a web administrative panel for managing users, study programmes, schedules, and classrooms;
- Integrating with university information systems to fetch data regarding users, curricula, and schedules;
- Adding features for comparative attendance analysis across semesters, automated alerts for critically low attendance levels, and predictive attendance modelling;
- Optimising processing throughput under high-concurrency user loads.

ACKNOWLEDGMENT

This research was supported by the Scientific Research Fund of Plovdiv University under project MUPD25-FMI-007 'Modern Solutions for Digital Transformation of Business and Education' (2025-2026).

REFERENCES

- [1] T. Fadelelmoula, "The impact of class attendance on student performance," *International Research Journal of Medicine and Medical Sciences*, vol. 6, no. 2, pp.47-49, 2018.
- [2] University of Plovdiv Paisii Hilendarski, Regulation for Educational Activity, 2025, Retrieved from <https://uni-plovdiv.bg/uploads/site/pravilnici/Pravilnici2025>
- [3] A. Kassem, M. Hamad, Z. Chalhoub, and S. El Dahdaah, "An RFID attendance and monitoring system for university applications," in 2010 17th IEEE International Conference on Electronics, Circuits and Systems, pp. 851-854, 2010.

- [4] S. Ashok, N. Divya, F. Samuel, and D. Mande, "RFID-based attendance system," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 10, no. 12, pp.2239-2248, 2022.
- [5] W. Farag, "An RFID-based smart school attendance and monitoring system," *BOHR Journal of Computational Intelligence and Communication Network*, vol. 1, no. 1, pp. 26-34, 2023.
- [6] C. Akinduyite, A. Adetunmbi, O. Olabode, and E. Ibidunmoye, "Fingerprint-based attendance management system," *Journal of Computer Sciences and Applications*, vol. 1, no. 5, pp. 100-105, 2013.
- [7] S. Shah, K. Sharma, R. Gupta, and M. Bhatia, "Face recognition based attendance management system," *International Journal for Research in Applied Science and Engineering Technology*, vol. 10, pp. 437-439, 2022.
- [8] A. Chinedu, O. Grace, and A. Chibuogu, "Quick Response (QR) Code Based Students' Attendance System in the University," *International Journal of Research and Innovation in Applied Science*, vol. 10, no. 7, pp. 1567-1572, 2025.
- [9] E. Karthik, V. Reddy, Y. Gouda, and D. Sivaprashanth, "QR Code Based Attendance Management System," *International Journal of Research and Innovation in Social Science*, ISSN No. 2454-6186, vol. 9, no. 4, pp. 5173-5220, 2024.
- [10] E. Oyerinde, "Design and implementation of a secure QR code-based attendance management system for higher education," *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, vol. 11, no. 5, pp. 77-89, 2025.
- [11] S. Kurniali, and Mayliana, "The development of a web-based attendance system with RFID for higher education institutions in Binus University," *In EPJ Web of Conferences*, vol. 68, p. 00038, 2014.
- [12] O. Osasuyi, E. Oghogho, and N. Ihedi-Okonkwo, "Design and Implementation of a Fingerprint-Based Attendance Management System," *NIPES-Journal of Science and Technology Research*, vol. 2, no. 2, 2020.
- [13] F. Masalha, and N. Hirzallah, "A student's attendance system using QR code," *International Journal of Advanced Computer Science and Applications*, vol. 5, no. 3, 2014.
- [14] S. Taju, Y. Mamahit, and J. Pongantung, "Implementing QR code and Geolocation Technologies for the Student Attendance System," *CogITo Smart Journal*, vol. 10, no. 1, pp. 221-232, 2024.
- [15] H. Gangurde, R., Dubhalkar, O. Ahire, and P. Borse, "Enhancing Attendance Accuracy with QR Code and Real-Time Location Tracking," *International Journal of Advances in Engineering and Management (IJAEM)*, vol. 7, no. 02, pp. 480- 483, 2025.
- [16] M. Muthulakshmi, P. Saravanan, M. Srihari, and P. Shunmugapandian, "Smart QR Code and Geo-Fenced Attendance System," *International Journal of Scientific Research & Engineering Trends*, vol. 12, no. 2, 2025.