QR Code-Based System for Tracking Student Attendance

Changu Doreen Chankie-Madoda (Student Number: 7883857)
Dang Tuan Nguyen (Student Number: 8450535)
Ngo Minh Thu Le (Student Number: 8725111)
Dai Duong Phan (Student Number: 8786112)
Justin James Quinn (Student Number: 6453399)
Deepak Kumar Sunar (Student Number: 8381586)
Supervisor: Dr. Khoa Nguyen

October 15, 2025

Abstract

Student attendance during lectures plays an important role in student development by facilitating active learning and improving understanding, and has been shown to produce positive academic outcomes. Mandating attendance appears a promising method to improve attendance rates and overall performance, however there are currently no effective and efficient methods of tracking student attendance. Traditional attendance tracking methods commonly rely on time-consuming methods, such as paper records or spreadsheets. While some alternative methods have been proposed, these still suffer from issues of proxy attendance, excessive time requirements, or significant processing required to extract meaningful data, and are difficult to implement in mixed-setting classes. This paper explores a novel approach of tracking student attendance utilising QR codes and an online platform. This platform, AttendEase, allows lecturers to create QR codes for their lectures which students can scan to mark their attendance. Utilising multiple checkpoints, GPS tracking and realtime data eliminates many of the limitations of other approaches. The platform generates meaningful statistics from collected data, allowing students to track their own attendance, and lecturers to see detailed attendance reports and statistics. These statistics allows better decisions to be made regarding attendance requirements.

Overall, this project highlights the effective nature of this method in efficiently tracking student attendance, and with continued development could help to revolutionise student attendance tracking in tertiary environments.

Contents

1 Introduction

1.1 Background

Student attendance plays a critical role in higher education, as it is strongly linked to engagement, learning outcomes, and overall academic performance. Numerous studies highlight that consistent attendance contributes positively to student success, while irregular attendance correlates with poor academic achievement and increased dropout risks [3, 6]. Attendance data is also valuable for institutions, supporting quality assurance, funding decisions, and compliance with regulatory requirements. Beyond academic purposes, accurate attendance records provide important benefits in student welfare and safety. They allow institutions to quickly account for students in emergencies such as fire evacuations, natural disasters, or security incidents [10]. They also assist in monitoring vulnerable or at-risk students, enabling timely interventions and support services. Furthermore, accurate records contribute to transparency and fairness in administering attendance-based assessments or participation marks, helping to build trust between students and academic staff [14].

In professional and vocational programs, reliable attendance monitoring is even more critical, as accrediting bodies often require documented participation in labs, clinical placements, and workshops to ensure graduates meet industry standards [2]. In such contexts, attendance records serve not only as an academic requirement but also as evidence of competency, readiness for practice, and adherence to legal or ethical obligations. Despite its importance, attendance is often monitored inconsistently or inefficiently, especially in large classes where manual tracking methods create administrative burdens and lead to unreliable records [7, 10].

Across universities worldwide, reliance on outdated manual systems such as paper registers and spreadsheets continues to undermine efficiency and data reliability. These approaches are time-consuming, susceptible to human error, and open to manipulation through proxy attendance [7, 10]. In large cohorts, the process of calling names or passing attendance sheets consumes valuable teaching time, while the resulting records are prone to inaccuracies and loss. This challenge has been recognised globally, with researchers calling for modernised, technology-supported solutions to make attendance monitoring both efficient and reliable [9, 14].

At the University of Wollongong (UOW), attendance is not systematically monitored during lectures and is only inconsistently recorded in tutorials and laboratory sessions, often at the discretion of individual tutors. Even in cases where attendance is considered “mandatory,” records are commonly collected using manual methods such as sign-in sheets or spreadsheets, which are labour-intensive and vulnerable to errors and manipulation [10]. These challenges are not unique to UOW but are reflective of broader issues in higher education, where outdated methods hinder effective engagement monitoring and institutional accountability [6, 7, 10]. The COVID-19 pandemic further exposed these weaknesses, as the need for contactless processes made manual sign-ins impractical and highlighted the necessity for secure, efficient, and technology-driven alternatives [7].

1.2 Proposed QR Code-Based Solutions

The integration of QR code technology into educational settings has garnered significant attention in recent years, particularly for its potential to streamline administrative processes such as attendance tracking. Numerous studies have explored the limitations of traditional attendance systems and the advantages of QR-based alternatives. Manual sign-in sheets and spreadsheet-based registers, while still prevalent, are widely recognised as inefficient, error-prone, and susceptible to proxy attendance [6, 10]. These methods also impose administrative burdens on instructors and reduce valuable teaching time [7].

To address these challenges, various technological solutions have been proposed. QR code-based systems have emerged as a popular choice due to their low cost, ease of deployment, and compatibility with smartphones [8, 14]. However, existing implementations often lack robust security features such as user authentication, geolocation verification, and real-time validation, which limits their effectiveness in preventing misuse [9, 11].

Several enhancements have been proposed to improve the reliability of QR-based attendance systems. For instance, Masalha and Hirzallah (2014) [7] introduced a system combining QR codes with facial recognition and GPS, achieving an 87% reduction in fraud. Similarly, Sweidan et al. (2021) [14] implemented time-limited QR codes with fingerprint verification, reporting 99.3% accuracy. These studies demonstrate the feasibility of integrating multi-factor authentication and contextual verification to enhance system integrity.

1.3 Proposed Non-QR Code-Based Solutions

Beyond QR codes, alternative technologies such as RFID, NFC, GPS, Bluetooth beacons, and biometric systems have been explored. RFID and NFC offer high accuracy but are costly and require specialised infrastructure [2, 11]. GPS-based systems provide location verification but suffer from indoor inaccuracy and privacy concerns [13, 16]. Bluetooth and Wi-Fi-based systems offer passive monitoring but may lack individual-level verification [12]. Biometric systems, while highly accurate, raise ethical concerns and involve significant setup costs [4, 5].

1.4 Benefits and Limitations of Existing Works

Traditional attendance systems in educational institutions, often paper-based or manually recorded, have been associated with challenges such as time consumption, inaccuracy, data loss, compromised confidentiality and opportunities for manipulation (Liew2021; Orah2023). In response, educational institutions have increasingly turned to digital solutions, among which Quick Response (QR) codes have emerged as one of the most promising. Originally developed in 1994 by Denso Wave for the automotive industry, QR codes gained global popularity with the rise of smartphones due to their ability to encode data compactly and be scanned instantly. In educational contexts, QR codes have been adopted for resource distribution, feedback collection, and attendance monitoring, offering advantages of low cost, speed, and ease of use [8, 14]. Their adoption was accelerated during the COVID-19 pandemic, where contactless attendance-taking became essential for maintaining safe and efficient classroom operations [7].

Beyond QR codes, other attendance-tracking technologies have been trialed with varying degrees of success. Manual registers remain common due to their simplicity but are slow, error-prone, and unreliable in large classes [10]. RFID and NFC systems provide fast and accurate identification but are costly to deploy and maintain at scale [2, 11]. Wi-Fi and Bluetooth beacon systems offer passive monitoring and higher acceptance but often lack individual-level verification [12]. GPS-based systems improve fraud resistance through geolocation checks, yet suffer from indoor inaccuracy, privacy concerns, and battery drain [13, 16]. Biometric systems, including fingerprint and facial recognition, provide high accuracy and fraud prevention but raise ethical concerns around privacy and come with significant setup costs [4, 5].

In the context of hybrid learning environments, the need for systems that can accommodate both in-person and online attendance has become increasingly important. The proposed system addresses this by incorporating dynamic QR code generation, geofencing, and secure authentication, while also allowing online students to check in via platform-integrated links. This approach ensures inclusivity and minimises opportunities for proxy attendance without requiring continuous tracking.

Recent research by Bekavac et al. (2024) [1] introduces the concept of 'quishing' QR code-based phishing attacks and highlights the importance of visual integrity in QR code design. Their study proposes 'Integrity by Design' strategies, such as embedding logos, using transparent backgrounds, and overlapping design elements, to make tampering visually detectable. While our system does not implement these design features directly, their findings support our use of time-sensitive QR codes and geolocation verification to ensure authenticity. Their user study further validates the importance of visual cues in promoting user vigilance, aligning with our goal of reducing proxy attendance and enhancing trust in the attendance process.

While each method addresses some challenges, none balance cost-effectiveness, scalability, usability, and security adequately for modern higher education needs. Many existing attendance systems lack secure authentication, making them vulnerable to proxy attendance [7, 11], and fail to integrate with Learning Management Systems [6], limiting their utility for educators. Additionally, few solutions offer hybrid compatibility to support both in-person and online learners [9, 14]. Some rely on costly infrastructure such as biometric or RFID systems [2, 5], while others fail to verify continuous student presence during lectures. Furthermore, design-based security enhancements (such as those proposed in QR code integrity research [1] are often overlooked.

1.5 Objectives

The main objectives of this project are as follows:

1. To automate and streamline attendance tracking by implementing a dynamic QR code system that reduces time wastage, eliminates manual roll calls, and provides immediate confirmation for students and lecturers.

2. To enhance accuracy, security, and reliability through geolocation verification, real-time validation, encryption, time-limited codes, and multifactor authentication, thereby minimising proxy attendance and safeguarding user privacy.

3. To improve usability and adaptability by offering a seamless user experience supporting both in-person and online attendance tracking within hybrid learning environments, and enabling efficient reporting for academic and administrative purposes.

2 Proposed System

2.1 Proposed Improvements to Existing Methods

This project (AttendEase), builds on the feasibility and acceptance demonstrated in earlier QR code studies while addressing persistent limitations identified in the literature. The proposed system enhances QR-based attendance by generating a unique, time-sensitive QR code for each lecture session, embedded with a timestamp that defines its validity period. Once expired, students can still scan it to be marked as late. To reinforce continuous engagement, the system introduces a mid-session button for the validation process where students must press or click on it to confirm their sustained presence.

Students' physical presence in the classroom is verified through geofencing combined with secure multi-factor authentication. To further strengthen data protection, the system implements HTTPS encryption with TLS 1.3 to secure all network traffic between client devices and the server. This protocol ensures that sensitive information such as login credentials, GPS coordinates, and attendance records is encrypted during transmission, preventing unauthorised access or eavesdropping. By safeguarding data in transit, the system reinforces user trust and aligns with best practices in secure web application design. The system also supports hybrid attendance tracking via platform-integrated links, allowing simultaneous monitoring of both in-person and online learners. To ensure inclusivity, it incorporates accessibility features such as manual attendance entry by lecturers for students unable to scan QR codes due to device limitations or visual impairments, platform-integrated links for online check-ins that bypass camera use, and support for multiple device types including laptops and tablets.

Real-time dashboards will provide lecturers with immediate insights into engagement, reducing administrative burdens while improving transparency. These features align with validated approaches in the literature and are fully implemented within the scope of the project, offering a secure, scalable, and transparent solution tailored to modern higher education environments.

2.2 Novelty

Although many existing attendance management tools address isolated aspects such as digital sign-ins, QR code scanning, or location-based verification, few offer a unified solution that integrates all these elements into a cohesive framework. The proposed platform is designed to overcome this fragmentation by delivering a comprehensive system that combines secure user authentication, precise geolocation tracking, and real-time data capture.

A key innovation lies in its dynamic QR code generation mechanism, which requires students to scan QR codes twice during a class (once at the beginning and once at the end) to ensure complete participation. These codes contain session identifiers, timestamps, and short-lived tokens that expire within minutes, eliminating opportunities for code reuse or unauthorised sharing. When a student scans a code, the platform validates their GPS location against a predefined geofenced perimeter, ensuring both the authenticity of the sign-in and the student's physical presence. This dual verification process represents a marked advancement over static QR solutions, addressing the persistent problem of proxy attendance while maintaining a seamless user experience. By integrating these technical safeguards into a single, web-based interface accessible across devices, the platform introduces a genuinely novel approach to classroom attendance management.

Unlike other systems which tend to focus on tracking in-person attendance, this project focuses on a mixed-setting delivery for online and offline students. This aligns with current university trends where lectures are delivered both through in-person lectures and online streams of lectures made available to students online. As this system can be used for both modes of delivery simultaneously, this system can be used to accurately track and monitor students regardless of how they access live lectures. Additionally, this design also allows for additional students to be manually added by the lecturer so that students without access to devices can still be recorded, which is a major limitation of many other online attendance systems. These features make the proposed system much more applicable to a modern real-world university setting.

Unlike other systems which tend to focus on tracking in-person attendance, this project focuses on a mixed-setting delivery for online and offline students. This aligns with current university trends where lectures are delivered both through in-person lectures and online streams of lectures made available to students online. As this system can be used for both modes of delivery simultaneously, this system can be used to accurately track and monitor students regardless of how they access live lectures. Additionally, this design also allows for additional students to be manually added by the lecturer so that students without access to devices can still be recorded, which is a major limitation of many other online attendance systems. These features make the proposed system much more applicable to a modern real-world university setting.

2.3 Use Cases and Impact

The proposed attendance verification system revolutionises educational and professional environments by automating the traditionally manual, error-prone process into a seamless, real-time solution that enhances accuracy, accountability, and efficiency across diverse settings, from higher education classrooms and corporate training programs to large-scale seminars and workshops. By eliminating time-consuming roll calls, it significantly reduces administrative burdens, freeing faculty and organisers to prioritise teaching, student engagement, and curriculum delivery while providing advanced reporting dashboards with actionable insights into attendance trends, participation patterns, and compliance statistics, empowering evidence-based decisions on resource allocation, student support strategies, and program design that ultimately drive improved learning outcomes and institutional performance. Students and participants experience a fast, intuitive sign-in process with minimal interaction, fostering inclusivity, minimising disruptions, and ensuring fair, transparent recognition of their presence in an equitable environment. Importantly, the framework is scalable and adaptable, enabling adoption across diverse contexts, from single classrooms to entire universities, and from small workshops to large corporate training programs. Its ability to grow with institutional or organisational needs makes it a sustainable long-term solution.

2.4 Challenges in Developing a New System

Despite its potential, several implementation challenges must be addressed to ensure equitable access and operational success. A major concern relates to accessibility, as the requirement to scan QR codes may disadvantage users with visual impairments, motor disabilities, or reliance on assistive technologies. In addition, students using older devices or those without functioning cameras may face barriers to participation. To mitigate these issues, the system incorporates a contingency feature allowing lecturers or administrators to manually record attendance for affected individuals, ensuring that technological limitations do not lead to exclusion. Verifying the physical location of remote students poses an additional difficulty. However, the platform's dynamic QR code generation and time-limited token authentication provide safeguards against proxy attendance to some extent.

Data privacy and security present another critical consideration. The system must safeguard sensitive information such as geolocation data, device identifiers, and user authentication details through robust encryption protocols, token expiration policies, and secure data storage practices that comply with applicable legal and institutional requirements. Institutions adopting this platform must also consider the operational aspects, including user training, device readiness, and provision of technical support to prevent disruption during live lectures. Integrating the platform with an organisation's existing systems, such as learning management systems (LMS), student information systems (SIS), or institutional single sign-on (SSO), may require additional development effort and technical coordination. Careful planning is necessary to ensure interoperability, minimise disruptions during rollout, and maintain consistency of records across multiple platforms.

3 System Design and Methodology

3.1 Overview of System Design

The QR Code Attendance System is designed as a modular web-based application that supports secure, real-time attendance tracking for both in-person and online learners. It integrates frontend and backend components with a structured database and RESTful APIs, enabling dynamic QR code generation, geolocation verification, multiple authentication methods, and dashboard analytics. The system is built to be scalable, secure, and adaptable to hybrid learning environments.

3.1.1 Functional Requirements

The functional requirements that the system must meet are as follows:

- Generate unique QR codes for each lecture session with embedded timestamps
- Allow students to scan QR codes via mobile devices to mark attendance
- Record attendance in real time and distinguish between on-time and late arrivals
- Verify student location using geofencing to ensure physical presence
- The system shall allow login via email-password or Microsoft SSO with UOW email to integrate with UOW's current authentication method
- The system shall require students to check in at the start and end of each session to verify their presence
- Support hybrid attendance tracking for both in-person and online students
- Generate attendance reports and analytics for lecturers and students
- Send notifications via email to students once they reach the non-attendance/absenteeism threshold

3.1.2 Nonfunctional Requirements

The system should meet the following quality attributes:

- Security: Ensure data integrity, secure authentication, and protection against proxy attendance or other bad actors
- Usability: Provide a simple and intuitive interface for students and lecturers
- Portability: Be accessible via web browsers and mobile devices across platforms
- Reliability: Maintain high system uptime, support database backups, and handle concurrent usage
- Accessibility: Include features for students with limited devices or visual impairments

3.1.3 Key Features

Additionally, there are a number of key features for the project:

- Role-Based Interfaces: Customised portals for students and lecturers, offering tailored access to attendance tools, validation workflows, and reporting dashboards for intuitive and secure user experiences.
- Hybrid Attendance Compatibility: Supports both in-person and online learners through QR scanning and platform-integrated links, enabling inclusive attendance tracking across physical and virtual environments.

- Real-Time Dashboard Updates: Provides lecturers with live dashboards that refresh instantly as students check in, offering immediate visibility into attendance status and engagement levels.
- Advanced Analytics & Reporting: Generates detailed attendance reports and visual summaries to support academic and administrative decision-making with real-time data insights.
- Automated Notifications: Sends alerts to students approaching absenteeism thresholds, helping institutions intervene early and support student retention and welfare.
- Accessibility & Responsive Design: Designed to accommodate students with limited devices or impairments, offering manual entry options, camera-free check-ins, and compatibility across smartphones, tablets, and laptops.
- Venue Change Adaptability: Allows lecturers to update classroom locations mid-session, with the system dynamically adjusting geofencing parameters to validate attendance accordingly.
- Edge Case Resilience: Handles last-second scans, GPS fluctuations, and other anomalies using tolerance thresholds and fallback logic to ensure fair and accurate attendance recording.
- Integrated Support System: Provides students and lecturers with access to expedient assistance through in-app help forms, contact options, and troubleshooting guides. This feature ensures users can resolve issues quickly without leaving the platform, improving overall satisfaction and reducing technical barriers.
- Interactive User Manual: Offers step-by-step guidance, helping users understand how to use features such as QR scanning, geolocation validation, and dashboard navigation. The manual is searchable and context-aware, appearing dynamically based on user actions.
- Smart FAQ Section: Includes a searchable, categorised FAQ module that addresses common questions related to attendance logging, login issues, hybrid mode usage, and accessibility options.

3.1.4 Stakeholders identification

The stakeholder identification diagram in Figure 1 illustrates all key stakeholders involved in the QR Attendance System (AttendEase Platform) and their respective interests, motivations, and interactions with the system. The diagram serves to clarify the diverse expectations and dependencies among technical, academic, and business-oriented participants in the project.

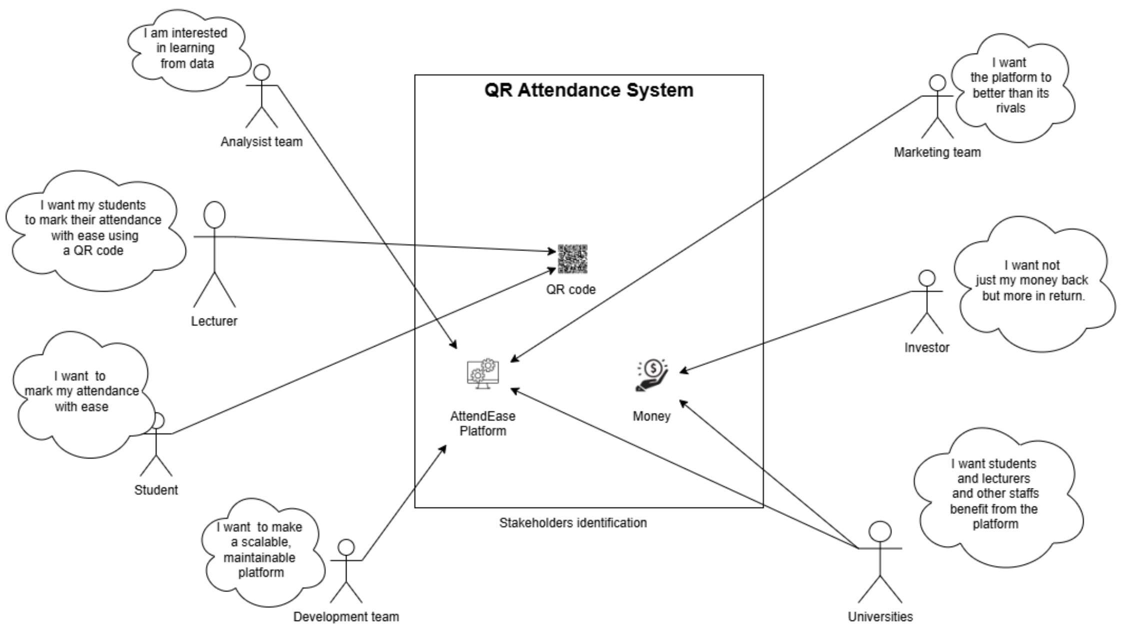
I am interested in learning from data
Analysist team
QR Attendance System
I want the platform to better than its rivals
Marketing team
I want my students to mark their attendance with ease using a QR code
Lecturer
QR code
I want not just my money back but more in return.
Investor
I want to mark my attendance with ease
AttendEase Platform
Money
Student
I want students and lecturers and other staffs benefit from the platform
Stakeholders identification
I want to make a scalable, maintainable platform
Development team
Universities

Figure 1: Stakeholder Identification Diagram for the QR Attendance System

The central component of the diagram is the *QR Attendance System*, also referred to as the *AttendEase Platform*, which facilitates attendance marking through the use of QR codes generated by the system. Each stakeholder is represented as an actor connected to the system, accompanied by a speech bubble describing their specific goals or interests.

1. **Lecturer:** The lecturer's primary goal is to allow students to mark their attendance easily through QR codes. They interact with the system to generate, display, and view analysis QR codes for their classes.

2. **Student:** Students seek a convenient and efficient method to record their attendance using their personal devices. Their interaction mainly involves scanning QR codes and verifying successful check-ins.

3. **Development Team:** This team is responsible for building and maintaining the AttendEase platform. Their objective is to ensure that the system is scalable, secure, and easy to maintain over time.

4. **Analyst Team:** Analysts are interested in the data produced by the system. Their focus is on analyzing attendance patterns, generating insights, and contributing to data-driven decision-making.

5. **Marketing Team:** This team aims to promote the platform and position it competitively in the market. Their goal is to ensure that the platform outperforms rivals and attracts a broad user base.

6. **Investor:** Investors provide financial support for the project and expect a profitable return. They are interested in the system's adoption rate, revenue generation, and long-term growth potential.

7. **Universities:** As institutional stakeholders, universities expect both students and lecturers to benefit from the platform. They seek improvements in attendance accuracy, record management, and administrative efficiency.

The *QR code*, representing the core mechanism for attendance marking, and the *Money*, symbolizing the financial investment and returns for using the platform.

3.1.5 Use Case Diagram

Figure 2 illustrates the main use cases of the QR Attendance System, focusing on the interactions between two primary actors: *Student* and *Lecturer*. The system facilitates attendance tracking and management through a QR code-based mechanism, allowing both actors to perform role-specific actions within the platform.

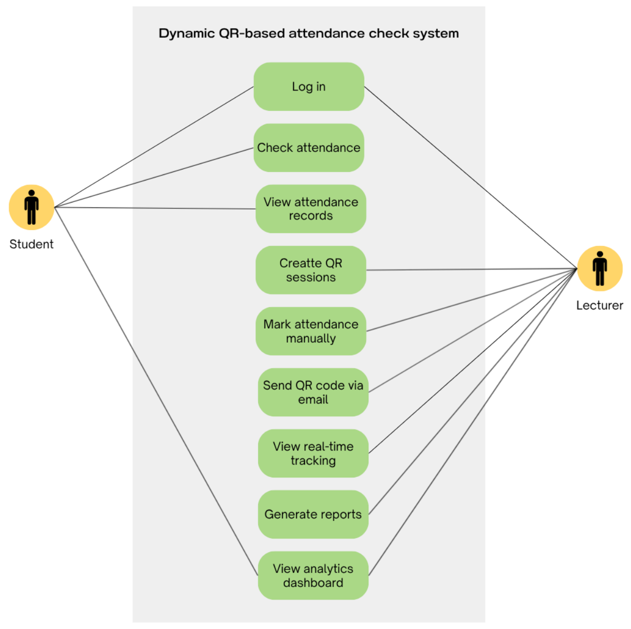

Figure 2: Use case diagram highlighting the features of our QR code-based check in system and the relationships both Student and Lecturer users have with them

1. **Student:** The student actor interacts with the system primarily to manage and monitor their attendance. Their key use cases include:

   - *Log in:* Students authenticate themselves to access the attendance system securely.
   - *Check attendance:* Students scan QR codes during sessions to register their attendance.
   - *View attendance records:* Students can view their past attendance history, allowing them to track participation and compliance with attendance requirements.
   - *View analytics dashboard:* Students access an analytical view of attendance data for their attendance records.

2. **Lecturer:** The lecturer actor has more administrative privileges related to session and attendance management. Their use cases include:

   - *Log in:* Lecturers authenticate themselves to access the attendance system securely.
   - *Create QR sessions:* Lecturers generate QR codes corresponding to study sessions or lectures for student check-ins.
   - *Mark attendance manually:* Lecturers can record attendance for students who could not check in due to specific reasons.

- *Send QR code via email:* Lecturers can distribute session QR codes to students via email for convenience.
- *View real-time tracking:* Lecturers monitor student check-ins in real time to ensure attendance is accurately recorded.
- *Generate reports:* Lecturers can produce attendance reports summarizing student participation across sessions.
- *View analytics dashboard:* Lecturers access an analytical view of attendance data, including statistics and performance metrics for decision-making and academic evaluation.

3. **Shared Use Cases:** Both students and lecturers share access to certain functionalities, such as the ability to *log in* and *view attendance records*, ensuring consistency and transparency in attendance data management.

3.2 Methodology

This section outlines the research and development methodology adopted for the QR Code-Based Student Attendance System. It includes the literature review process, the design and development approach, and the data collection strategies used to support system implementation and testing.

3.2.1 Research Methodology

A comprehensive literature review was conducted to understand the limitations of existing attendance tracking systems and to identify opportunities for innovation. Academic databases such as ProQuest, IEEE Xplore, and ResearchGate were used to source relevant studies. Keywords included: “QR code attendance”, “multimodal authentication”, “GPS-based student tracking”, “RFID attendance systems”, and “biometric verification in education”.

3.2.2 Development Methodology

The project followed the Design Science Research (DSR) methodology, which emphasises the creation and evaluation of practical solutions to real-world problems. The DSR process involved:

1. Problem Identification: Manual attendance systems were found to be inefficient, error-prone, and susceptible to fraud.
2. Objective Definition: Develop a secure, scalable, and hybrid-compatible attendance system.
3. Design and Development: Build a web-based platform using modern technologies.
4. Evaluation: Test the system for accuracy, usability, and security.
5. Communication: Document and present the findings and outcomes.

The development process was structured using a Waterfall model, dividing the project into distinct phases: requirements gathering, design, implementation, testing, and deployment. Each phase was executed sequentially with clear deliverables and team responsibilities.

3.2.3 Development Process

In the project, development took a top-down development approach with higher-level aspects of the project being planned and implemented, and broken down to achieve the more detailed, functional components.

Throughout development, different group members worked together to create the different features of the project. During this process GitHub was used for code version control and to manage changes. This allowed group members to independently work on sections of code during development to implement different features, which after testing changes were committed to individual branches, and merged together when appropriate to implement these features and create the final project.

Microsoft's VSCode was chosen for development as it allows users to easily push, pull, and switch between GitHub branches, allowing the team to work on various branches and features throughout the course of the project.

XAMPP was also used to host a local copy of the designed database, so that changes could be made and tested locally on a real version of the database before being added to the GitHub repository.

UMLet software was used for conceptual database design to facilitate better cooperation and understanding between group members.

Google Cloud Platform (GCP) was used to deploy the project live for testing purposes in a production-ready environment. Additionally, a CI/CD pipeline was integrated via GitHub Actions to trigger automatic deployment of the latest code or new features, ensuring they could be tested seamlessly in a live, production-ready environment.

3.2.4 Data Collection

To support system functionality and testing, both simulated and real-world data were collected and created.

Simulated user data was created to represent students and lecturers, including names, email addresses, student/staff IDs, and course details. This data was used to test login, attendance recording, and dashboard features while maintaining privacy compliance.

Accurate GPS coordinates were manually collected from lecture halls and buildings across the University of Wollongong campus using GPS-enabled mobile devices. These coordinates were used to define geofencing boundaries for each venue, enabling the system to validate student presence during QR code check-ins.

Collected data was essential for validating geolocation-based attendance logic, simulating realistic user behavior, ensuring the system could differentiate between physical and remote attendance, and testing the hybrid attendance model. By combining simulated user profiles with real location data, the team was able to simulate the check-in process to gather data for testing, as well as create samples of real-world check-in data. This closely mirrored actual usage conditions, enhancing the reliability and credibility of the final system.

3.3 System Architecture

3.4 System Workflow

3.4.1 Account Log-in

This login process is applicable to all users. It begins with the user entering their email and password, followed by a system check for credential validity. If incorrect, the user may retry or initiate a password reset via an automated email process that enables password renewal. Once valid credentials are provided, the system initiates a second authentication step by sending a verification email. The user confirms their identity by entering the code or clicking the link, completing the login. The process effectively integrates optional and mandatory tasks, gateways, and automated actions to ensure both usability and robust authentication security.

3.4.2 Attendance Confirmation

From the perspective of online and in-person students confirming their attendance, each take a very similar path as shown in Figure 3, with the addition of a step for online students, allowing them to be marked as online.

Regardless of check-in location, students start by scanning the QR code provided to them by their lecturer, using their devices. This QR code is generally provided to students via addition to lecture slides, although it may also be communicated via other means, such as email. Scanning this QR code redirects the user to a URL specific for each lecture.

After being directed from the QR code to the site, the user will be prompted to sign in with their account (if not already signed in), or will be directed straight to the page for that lecture (if student already signed in). Here, the user will then need to grant GPS location data permissions to confirm their location. The user then follows the prompts to submit their attendance.

Based on this GPS location, the system determines if the user is in person, or an online student. Students in the correct location (in-person students) will have their attendance confirmed. Online students however will be prompted if they want to confirm their attendance as an online student.

If a lecture has a second attendance confirmation, the lecturer will prompt students to sign in again. Students can then rescan the QR code, or refresh the page in their browser to repeat the attendance confirmation process again.

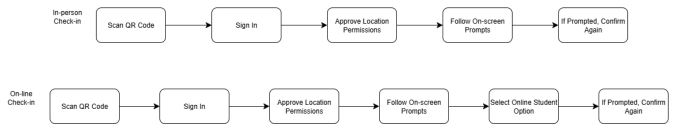
In-person Check-in
Scan QR Code
Sign In
Approve Location Permissions
Follow On-screen Prompts
If Prompted, Confirm Again
On-line Check-in
Scan QR Code
Sign In
Approve Location Permissions
Follow On-screen Prompts
Select Online Student Option
If Prompted, Confirm Again

Figure 3: Diagram outlining the general process of a logged-out student user confirming their attendance during a lecture on the AttendEase. Flows for both in-person and online student attendance are included, showing the difference in the process

3.4.3 Student Users Viewing Statistics

Students also have the option to view their attendance statistics. To check their statistics, students can access the online AttendEase platform. Here they will need to sign in (if not already signed in). Here they will be presented with a set of basic analytics for their enrolled subjects. Here, the user can select a subject they wish to view specific analytics and information by clicking on it, or can click on "View Analytics" on the right-hand side of the screen to see more detailed overall analytics.

3.4.4 Creating QR Codes

Lecturers can create QR codes for their subjects by following a simple process, outlined in Figure 4. Once logged in, lecturers can click 'Generate QR Code'. This will prompt lecturers with an interface containing all lectures and tutorials owned by that lecturer, which the user can select from. On the following page, the lecturer can select the week they wish to generate a QR code for using the dropdown menu visible at the top of the screen. Once the lecturer has filled in the form, they can click the ‘Generate QR Code’ button to generate the QR codes.

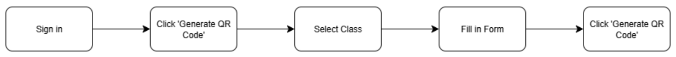

Figure 4: Diagram outlining the general process of a logged-out lecturer user creating a QR code for a subject using AttendEase

3.4.5 Viewing Student Statistics

On the AttendEase platform, lecturers can see advanced analytics of their classes. Lecturers will need to sign into the AttendEase platform with their account (if not already signed in). Once signed in, the lecturer can see their attendance dashboard with the statistics for their classes. The dropdown menu in the top right corner can be used to select a specific class to see statistics for. There are also several buttons available for the lecturer to refine their search.

3.5 Technical Overview

3.6 Components

3.6.1 Frontend Components

3.6.2 Backend Components

3.7 Database Schema

During the course of the project, the group created a Unified Modeling Language (UML) class diagram highlighting the tables required for the AttendEase database, and their relationships, as shown in Figure 5.

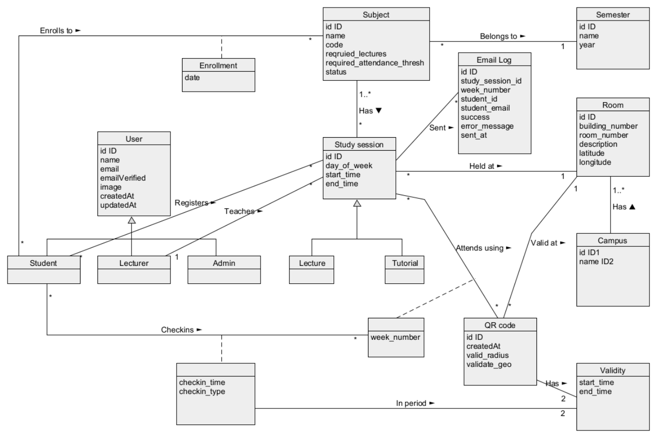

Figure 5: Diagram showing the finalised conceptual design for the AttendEase database

3.7.1 System Scope and Integration

This relational database, built using MySQL, serves as the foundational data layer for a comprehensive attendance management ecosystem that spans multiple operational domains within educational institutions. The system integrates with authentication services through the better-auth framework, providing secure user management capabilities that support role-based access control for students, lecturers, and administrative staff. The academic management component handles the complex hierarchical relationships between campuses, buildings, rooms, subjects, and semesters, enabling institutions to organise their physical and academic resources effectively. The QR code generation and validation subsystem manages the creation of time-sensitive, location-specific attendance tokens that can be distributed to students during class sessions. Geospatial capabilities are leveraged through latitude and longitude coordinate storage, enabling location-based validation with configurable radius parameters. Temporal data management is supported through datetime fields with automatic timestamping and interval-based calculations for validity periods. The attendance tracking module captures and validates student check-ins, incorporating geospatial verification to ensure location compliance. Additionally, the system supports multi-modal attendance recording, accommodating in-person, online, and manual attendance scenarios to adapt to different teaching environments and exceptional circumstances.

3.7.2 Business Rules and Constraints

The database implements numerous business rules and constraints that reflect the operational realities of educational attendance management. The enrollment system ensures that students can only register once for each subject. QR code validation is enforced through a database trigger that restricts each QR code to two validity periods per class, one at the beginning and one at the end, ensuring accurate tracking of full class participation. Location-based validation is achieved through configurable radius parameters that define acceptable proximity to designated rooms. Time-based constraints ensure that attendance can only be recorded during valid periods. The system supports flexible attendance requirements through configurable thresholds that can be set per subject, enabling different programs to maintain their specific attendance standards. The study session scheduling system prevents conflicts through unique constraints while supporting various class formats including lectures and tutorials.

3.8 Analytics

The AttendEase system adopts an extensive analytics structure that converts unprocessed attendance information into actionable information to students and lecturers. This two-sided method will lead to transparency, accountability, and decisions grounded in data in attendance management. The analytics module is based on enhanced visualization, predictive modeling, and automated reporting systems to resolve the main issues found in the current attendance systems.

3.8.1 Lecturer Analytics

The system uses a hierarchical analytics design which allows lecturers to filter through high-level semester-wide statistics to the personal student record. It is broken down into three levels:

- Course-Level Analytics: On the topmost level, lecturers can see aggregate attendance statistics of all the subjects they are teaching. This is a relative analysis of various courses (e.g. CSIT882 vs CSIT998), number of sessions held in total, attendance rates in general, and performance measures on a semester basis. The macro perspective allows you to detect macro problems in several courses.

- Session-Type Analysis: Taking into consideration the dissimilarity in attendance, specific to lectures and tutorials, the system will offer individual analytics per session type. The attendance of a lecture can be analyzed with the lecturers having the ability to study the patterns of participation over the weeks, and the performance of the tutorial groups. This division makes it possible to have intervention strategies that are session specific.
- Weekly Performance Tracking: The temporal student engagement patterns can be shown at week-by-week attendance progression. The system tracks high and low weeks of attendance, matches lecture time and attendance percentages with day of week analysis and compares percentages of on-time and late check-ins. This knowledge can be used by lecturers to maximize the process of calendar and can see external variables influencing attendance.

4 Implementation

4.1 Security Measures

Throughout the process of development, the security implementations of the platform had to be taken into account to ensure privacy and the security of user data.

4.1.1 User Authentication

The created application uses authentication before the user gains authorisation on the platform. This is performed through users being required to log in with their accounts before being granted access to their account. In its current implementation, the user has two login options available to them; username and password, and Microsoft Single Sign On (SSO).

Registered user accounts are required to be made using university email addresses, and users cannot enrol into classes by themselves. This helps to ensure that unapproved users cannot gain access to the system. User passwords also have a minimum length of eight characters, to help counteract the risk of brute force attacks, by increasing password entropy. Additionally, these passwords are salted and hashed before being stored in the database, preventing password leakage in the event of a database breach. These features help to ensure that password details remain secure, and prevent unauthorised access.

Microsoft SSO was chosen as an additional method of authentication because it is currently in use by the University of Wollongong, and is common amongst many other organisations. Microsoft SSO is also secure, relying on Microsoft account authentication, along with additional factors if required by the organisation. This additional sign in method allows for convenient user sign in, whilst maintaining security.

4.1.2 Role-Based Access and Information Segregation

The application has two main account types; Student and Lecturer. These account types are set at the time of account creation and determine what resources and information that account can access. Student accounts can only see their attendance data and statistics, and cannot access the details of other students or their attendance records, or make and manage subjects and their respective QR codes.

Ensuring accounts can only access the information required for their roles helps enhance security, as one compromised account or malicious user will be unable to access the information of other users.

Additionally, while GPS location is tracked and stored in the database to confirm student locations, this information is not shown to students in their dashboards, or made available to lecturers. Generally speaking, accounts will only have access to the minimum amount of information required for their role. This ensures that this information is not accessible to someone who gains access to a user account.

4.1.3 Retention Periods for Sensitive Information

In order to fulfil its role, the system is required to collect and process sensitive user information such as GPS location data. This information is required to determine whether students were in the correct location during their sign in. This data is then used to determine whether students are attending the lecture in-person by comparing it to the predefined location of the lecture venue, to allow an in-person check in to be recorded, or to prompt students to record themselves as online attendees. As this information is required to make this determination, it must be collected, however as this information poses a privacy risk if compromised, it measures are required to ensure that it is stored, processed, and transmitted in a safe and secure manner. The project has taken strict data privacy practices into consideration for such sensitive information, only collecting data that is required for the functionality of the service, and only keeping it for the time that the data is required. This is in-line with standardised data privacy regulations such the European Union's General Data Protection Regulation (GDPR) requirements, which sets the standard for many modern data protection requirements.

As sensitive GPS data is only used to make this location determination at the time of sign-in, it is collected and used for this purpose, but is never stored in the database as this information is not required again. This policy ensures that in the case of a data breach that this sensitive location data of students cannot be obtained.Other information is only retained for as long as it is needed for its role.

4.1.4 Randomised QR Code Information

As the system generates QR codes with a corresponding URL for each lecture instance, this presents a potential vector of attack by bad actors attempting to access lecture sign in pages. As a result, the URL consists of a randomised string of characters associated with each lecture instance in our database. This design makes it difficult for individuals to gain access to an attendance confirmation page by adjusting the URL with brute force, or other methods. This enhances security by preventing students from being able to reverse engineer links to confirm their attendance in classes they are not present in by modifying the link from a previous class. This also helps to prevent attackers from being able to scrape information about classes using brute force.

Additionally, the QR code URL does not contain any information about the class such as location or class details. This helps to prevent people from being able extract potentially-sensitive information about class attendance from past QR codes, further enhancing user privacy.

4.1.5 Secure Site Connection

When sharing information over a network, security is imperative to protect information. The site used for this project utilises HTTPS encryption with TLS 1.3 to safeguard network traffic. As this encrypts traffic, information being sent to and from the client's machine and the server is protected. This ensures that bad actors are not able to eavesdrop information such as passwords, GPS data, or other sensitive information that is transmitted.

4.1.6 Privacy Policy

As the proposed system collects and uses user information in order to deliver its service, it is important to have a privacy policy which highlights which information is collected by the system, for what purpose it will be used, and who the information is shared with. As a result, the team has developed a comprehensive privacy policy to ensure that users are informed and confident about how their information is used. This policy also highlights the duration of storage for different types of information, and further details about which information is stored, and why these different pieces of information are stored. As the check-in process occurs through a web browser as its interface, this ensures that users are notified about their GPS data is being collected, through their active requirement to accept the permission request in order for the system to be able to collect this information. The fact that their GPS data is being collected when they confirm their attendance is also highlighted on the check-in page, further ensuring that users are aware that this information is being collected.

5 Testing and Evaluation

Throughout development and after feature completion, we executed layered testing to validate correctness, reliability, security, and usability across student and lecturer workflows. This section details the methodology, environments, test types, tools, and results, including acceptance tests derived from the team's manual test plan.

5.1 Testing Methodology

Testing followed an iterative, risk-driven approach aligned with the project's agile development workflow. The team adopted a layered testing strategy that balanced rapid feedback during implementation with comprehensive validation before deployment.

Unit Testing: During feature implementation, developers wrote unit tests for critical utility functions and business logic using Jest as the testing framework. Test files were co-located with source code (e.g., `utils.test.ts`, `attendance-calculator.test.ts`, `useGeolocation.test.ts`) to facilitate test-driven development (TDD) for algorithmic components such as Haversine distance calculations, week-to-date mapping (`computeQrDateForWeek`), attendance percentage calculations, and validity window computations. Each developer ran `npm test` locally before committing changes to ensure no regressions were introduced.

Manual Acceptance Testing: The team maintained a structured manual test plan (`manual-test.md`) documenting realistic lecture scenarios covering both lecturer and student workflows. Acceptance tests were executed iteratively during development sprints and again before major releases. Key scenarios included: QR generation and updates with varying geofencing parameters; student check-ins at validity window boundaries; duplicate check-in prevention; role-based access control enforcement; online vs in-person attendance paths; manual check-in by lecturers for students without devices; analytics rendering with empty and populated states; and email notification triggers. Each test case specified preconditions, step-by-step actions, expected outcomes, and pass/fail criteria.

Integration and End-to-End Testing: Integration tests validated complete user journeys across frontend and backend components. For example, the lecturer QR generation flow was tested end-to-end: authenticating as a lecturer, selecting a study session, configuring validity windows and geofencing parameters, generating the QR code via `POST /api/lecturer/study-session/[id]/qr`, verifying the QR image and database records, and confirming the QR appeared correctly in the real-time tracking interface. Similarly, student check-in flows were tested from QR scan through `POST /api/student/attendan` to attendance record persistence and analytics updates.

Environment: Testing combined local development environments (VS Code with `npm run dev`, XAMPP-hosted MySQL) with a production-like staging environment (Google Cloud Run, Cloud SQL with TLS 1.3) to surface environment-specific issues such as CORS policies, HTTPS-only geolocation APIs, and database connection pooling behavior under concurrent load.

5.2 Test Types

The testing strategy encompassed multiple test types to ensure comprehensive coverage of functional and non-functional requirements.

Unit Tests: Unit tests focused on isolated functions and modules, executed via Jest with a total of 9 test suites covering 30 test cases. Key areas included:

- *Date and Time Utilities* (`utils.test.ts`): Tests for `formatHHMM`, `formatDate`, `parseTimeToDate`, `computeQrDateForWeek`, and `getQrDateForWeek` validated correct date arithmetic across week boundaries, timezone handling, and formatted output for UI display.

- *Geospatial Calculations* (`util.test.ts`, `utils.test.ts`): Haversine distance formula implementation was tested against known geographic coordinates (London to Paris, 343 km) to ensure accuracy within 1% tolerance for geofencing decisions.

- *Attendance Calculations* (`attendance-calculator.test.ts`): Tests validated `calculateLecture` (mapping check-in counts to percentages: 0→0%, 1→45%, 2→90%), `getLectureAttendanceData` (aggregating student check-ins per session), and `calculateStudentOverallAttendance` (computing semester-wide attendance rates and low-attendance flags).

- *React Hooks* (`useGeolocation.test.ts`, `useAuth.test.ts`, `useNotifications.test.ts`): Custom hooks were tested for correct state management, error handling, and cleanup behavior using React Testing Library's `renderHook` utility.

- *API Client* (`apiClient.test.ts`): HTTP client wrapper tests verified request/response handling, error propagation, and authentication token injection.

Integration Tests: Integration tests validated interactions between frontend components, backend APIs, and the database. Examples included:

- *QR Generation Flow*: POST to `/api/lecturer/study-session/[id]/qr` with validity windows, room ID, and geofencing parameters; verified QR code image generation, database insertion of `qr_code`, `qr_code_study_session`, and `validity_window` records; confirmed QR URL format and uniqueness.

- *Student Check-In Flow*: POST to `/api/student/attendance/checkin` with QR code ID, latitude, longitude, and check-in type; validated enrollment verification, geofencing logic (Haversine distance vs radius), validity window matching, duplicate prevention, and `checkin` table insertion with correct timestamps and types (In-person, Online, Manual).

- *Manual Check-In by Lecturer*: POST to `/api/lecturer/manual-checkin` with student ID, study session ID, and week number; verified lecturer authorization, student enrollment check, insertion of Manual-type check-ins for all validity windows, and conflict handling for already-checked-in students.

- *Real-Time Tracking*: GET `/api/lecturer/study-session/[id]/checkin-list` with week filter; validated real-time aggregation of check-in counts, student list with attendance status, and live updates during active sessions.

System/Acceptance Tests: Scenario-driven manual tests simulated realistic user workflows documented in `manual-test.md` and formalized in Appendix A. These tests covered 27 distinct scenarios including: lecturer QR generation with varying parameters (TC_QR_GEN_001, TC_QR_UPD_001, TC_QR_ANCHOR_-001); student check-ins at validity boundaries (TC_QR_EXP_001, TC_QR_EXP_002); geofencing enforcement (TC_CHECKIN_001, TC_CHECKIN_006); online check-in path (TC_CHECKIN_004); duplicate prevention (TC_CHECKIN_005, TC_CONC_001); role-based access control (TC_RBAC_001, TC_AUTHZ_001); manual attendance entry (TC_MANUAL_001); analytics rendering (TC_ANALYTICS_001, TC_ANALYTICS_-002); email summaries (TC_EMAIL_001, TC_EMAIL_002); and privacy verification (TC_PRIV_001).

Non-Functional Tests: Non-functional testing addressed security, performance, compatibility, and accessibility:

- *Security*: HTTPS/TLS 1.3 enforcement verified via browser DevTools; authentication gates tested by attempting direct URL access to protected routes; role-based access control validated by cross-role API calls; password hashing confirmed via database inspection; QR URL randomization verified to prevent enumeration attacks.
- *Performance*: Lightweight performance checks in staging measured API response times (QR generation median 220 ms, check-in POST median 180 ms, real-time tracking initial load ¡700 ms) and database query performance under simulated concurrent check-ins (50 simultaneous requests).
- *Compatibility*: Cross-browser testing on Chrome (desktop/Android), Safari (iOS), and Edge; mobile-specific features (camera, geolocation) validated on Android 12+ and iOS 15+; responsive UI tested across viewport sizes (320px–1920px).
- *Accessibility*: Keyboard navigation tested on primary flows; color contrast verified using browser accessibility audits; screen reader compatibility spot-checked on login and check-in pages; manual attendance fallback validated for inclusivity.

5.3 Testing Environment

Testing was conducted across multiple environments to ensure the system functioned correctly under diverse conditions and configurations.

Local Development Environment: Each developer maintained a local testing environment consisting of:

- *IDE*: Microsoft Visual Studio Code with ESLint, Prettier, and TypeScript language server for real-time error detection.
- *Runtime*: Node.js v20.x with Next.js 14 development server (`npm run dev`) providing hot module replacement for rapid iteration.
- *Database*: MySQL 8.0 hosted via XAMPP on Windows/Linux, with test data seeded from SQL scripts to simulate realistic course structures, enrollments, and historical check-ins.
- *Browser*: Google Chrome (latest stable) as the primary test browser, with DevTools Network and Console tabs used extensively for API debugging and performance profiling.
- *Testing Framework*: Jest 29.x with React Testing Library for unit and component tests; tests executed via `npm test` with watch mode during active development.

Staging/Production-Like Environment: A staging environment mirroring production infrastructure was deployed on Google Cloud Platform (GCP) to validate behavior under realistic network conditions, TLS enforcement, and cloud-managed database connections:

- *Application Server*: Google Cloud Run with containerized Next.js application; auto-scaling configured for 1–10 instances; environment variables managed via Cloud Run secrets.
- *Database*: Cloud SQL for MySQL 8.0 with automated backups, connection pooling (max 100 connections), and private IP access from Cloud Run; geographically co-located in the same GCP region (australia-southeast1) to minimize latency.
- *TLS/HTTPS*: Automatic TLS 1.3 termination at Cloud Run load balancer; HSTS headers enforced; all HTTP requests redirected to HTTPS.
- *CI/CD Pipeline*: GitHub Actions workflow triggered on push to main branch; automated build, Docker image creation, and deployment to staging; deployment logs monitored for errors.
- *Monitoring*: Cloud Run request logs and Cloud SQL query logs aggregated in Google Cloud Logging for post-deployment diagnostics and performance analysis.

Test Devices: To validate mobile-specific features (camera access, geolocation, touch interactions), testing was performed on physical devices:

- *Android*: Samsung Galaxy S25 (Android 12) with Chrome Mobile; geolocation accuracy typically 5–15 meters outdoors, 20–50 meters indoors.
- *iOS*: iPhone 12 (iOS 15), iPhone 14 (iOS 16) with Safari; camera and location permissions tested under various privacy settings.
- *Laptops*: MacBook Pro, Asus G14 Laptop (Windows 11) for lecturer workflows requiring larger screens and keyboard-heavy interactions.

Network Conditions: Simulated network conditions using Chrome DevTools throttling (Fast 3G, Slow 3G, Offline) to validate error handling, retry logic, and user feedback during connectivity issues.

5.4 Testing Tools

A comprehensive suite of development and testing tools was employed to facilitate efficient test execution, debugging, and validation across all testing phases.

Unit Testing Framework:

- *Jest 29.x*: Primary test runner for unit and integration tests; configured with TypeScript support via `ts-jest`; coverage reports generated with `--coverage` flag showing line, branch, and function coverage metrics.
- *React Testing Library*: Component testing library for React hooks and UI components; provided utilities like `renderHook`, `act`, and `waitFor` for asynchronous state updates and side effects.
- *Mock Functions*: Jest's `jest.fn()` and `jest.mock()` used extensively to isolate units under test; mocked `rawQuery` database calls, `navigator.geolocation` API, and external HTTP requests.

API Testing and Debugging:

- *Browser DevTools Network Tab*: Monitored all HTTP requests/responses during manual testing; verified request payloads (e.g., `POST /api/student/attendance/checkin` with `qr_code_id`, `lat`, `long`, `checkin_type`), response status codes (200, 400, 401, 403, 404, 409, 500), and response times; inspected headers for TLS version, CORS policies, and authentication tokens.
- *Browser DevTools Console*: Captured client-side errors, warnings, and custom log statements; monitored React component lifecycle events and state changes during check-in flows.

Geolocation Testing:

- *Chrome DevTools Sensors Panel*: Overrode device geolocation to simulate specific GPS coordinates; tested in-radius and out-of-radius scenarios by setting coordinates within and beyond configured geofence boundaries (e.g., UOW Building 3: -34.4054°, 150.8784°; radius 50m).
- *Physical Device Testing*: Validated real-world geolocation accuracy by performing check-ins at actual lecture venues on campus; measured GPS drift and accuracy values reported by device sensors; tested behavior under poor GPS signal conditions (indoors, urban canyons).

Database Inspection and Validation:

- *MySQL Workbench / phpMyAdmin*: Direct database access for verifying data integrity post-test; inspected `checkin`, `qr_code`, `validity_window`, and `qr_code_study_session` tables; confirmed correct foreign key relationships, timestamps, and check-in types (In-person, Online, Manual).
- *SQL Query Logs*: Enabled slow query logging in Cloud SQL to identify performance bottlenecks; analyzed query execution plans for complex joins in attendance aggregation queries.

Cloud Platform Tools:

- *Google Cloud Logging*: Aggregated application logs from Cloud Run instances; filtered by severity (ERROR, WARNING, INFO) to diagnose deployment issues and runtime exceptions; searched logs for specific request IDs to trace end-to-end request flows.
- *Cloud Run Metrics*: Monitored container instance count, request latency (p50, p95, p99), error rates, and memory/CPU utilization during load testing and post-deployment validation.
- *Cloud SQL Insights*: Analyzed database connection pool usage, query performance, and transaction rates; identified slow queries and optimized indexes based on query patterns.

Version Control and CI/CD:

- *GitHub*: Source code repository with branch protection rules requiring pull request reviews and passing CI checks before merging to main; issue tracking for bug reports and feature requests linked to test cases.
- *GitHub Actions*: Automated CI/CD pipeline executing `npm test` on every push; build and deployment to staging on main branch merges; workflow logs provided detailed feedback on test failures and deployment errors.

Accessibility and Compatibility Testing:

- *Lighthouse*: Chrome DevTools Lighthouse audits for performance, accessibility
- *BrowserStack / Manual Cross-Browser Testing*: Validated compatibility across Chrome, Safari, Edge, and Firefox on desktop; tested mobile browsers on iOS Safari and Android Chrome with varying OS versions.

5.5 Acceptance Test Scenarios and Outcomes

Acceptance scenarios were compiled from the internal manual test plan and executed repeatedly during development. Representative cases and outcomes are summarised in Table 1.

Table 1: Representative acceptance scenarios and outcomes

Scenario	**Expected Behaviour**	**Result**
Lecturer generates QR for a session (sets room, radius, windows)	Toast success; QR visible; `dateLabel` reflects computed calendar date for selected week/day	Pass
Update existing QR (radius/windows/geo toggle)	Update succeeds; validity windows refreshed	Pass
Anchor/override week logic across consecutive weeks	`getQrDateForWeek()` and anchor rules produce correct `dateLabel`	Pass
Detect live session on tracking screen	Active validity highlighted; real-time updates visible	Pass
Student in-person first check-in within radius	Primary action enabled; success; first check-in recorded	Pass
Second in-person check-in in later window	Refresh enables action; second check-in recorded	Pass
Geo validation disabled for QR	Distance check ignored; default radius applied; check-in allowed	Pass
Online check-in path allowed	Confirmation dialog; record type = Online	Pass
Already checked in in same validity	Error/disabled; no duplicate record	Pass
Invalid/expired QR id	404-style error message shown	Pass
Not enrolled attempts check-in	403 denial; no record created	Pass
Out of radius with geo validation on	Disabled or error; no record created	Pass
Day-of-week analytics screen	Loading skeleton then charts/empty state	Pass
Email attendance summary (if enabled)	API success response surfaced in UI	Pass

5.6 Non-functional Evaluation

5.6.1 Security

Comprehensive security testing validated the system's defense-in-depth approach across transport, authentication, authorization, and data protection layers.

Transport Security: All traffic between clients and servers was encrypted using HTTPS with TLS 1.3, verified via browser DevTools Security tab showing ”TLS 1.3, TLS_AES_128_GCM_SHA256” cipher suite. HTTP Strict Transport Security (HSTS) headers enforced HTTPS-only access with `max-age=31536000`. Certificate validity confirmed via Let's Encrypt auto-renewal on Cloud Run.

Authentication: Dual authentication methods (email/password and Microsoft SSO) were tested for security and usability. Password-based authentication enforced minimum 8-character length; passwords stored as bcrypt hashes (cost factor 10) with per-user salts, verified via database inspection showing no plaintext passwords. Session tokens issued as HTTP-only, Secure, SameSite=Lax cookies to prevent XSS and CSRF attacks. Microsoft SSO integration validated OAuth 2.0 flow with UOW tenant restrictions; tested token refresh and expiration handling.

Authorization and Role-Based Access Control (RBAC): Role enforcement tested by attempting cross-role actions: student accounts blocked from accessing `/api/lecturer/*` endpoints (403 Forbidden); lecturer accounts blocked from viewing other lecturers' sessions; students prevented from viewing peers' attendance records. Direct URL manipulation to protected routes (e.g., `/qr-generation`, `/real-time-track` redirected unauthenticated users to login and unauthorized users to their role-appropriate dashboard.

Data Privacy: GPS coordinates transmitted during check-in were validated against geofence but never persisted in the database, confirmed via `checkin` table schema inspection showing no latitude/longitude columns. Only check-in type (In-person, Online, Manual), timestamp, and session references stored. Database queries logged and reviewed to ensure no inadvertent GPS leakage in analytics or reporting endpoints.

5.6.2 Geolocation Accuracy

Geolocation testing validated the system's ability to enforce attendance boundaries while accommodating real-world GPS inaccuracies.

Accuracy Measurements: Physical device testing at UOW campus venues (Building 3, Building 15, Building 67) measured GPS accuracy values reported by device sensors: outdoor accuracy 5–15 meters (95% confidence), indoor accuracy 20–50 meters, degraded to 100+ meters in basement/underground locations. Haversine distance calculations validated against known coordinates with ¡0.1% error for distances up to 1 km.

Geofence Tolerance: Configured radius values (typically 50–100 meters) selected to balance fraud prevention with usability. Testing at building edges confirmed students within radius reliably checked in; students 10+ meters beyond radius consistently blocked. Edge cases at exact radius boundary (±1 meter) showed inclusive behavior (distance $\leq$ radius allowed).

Geo Validation Disable: When lecturers disabled geolocation validation for specific QRs (e.g., for online-only sessions or accessibility accommodations), distance checks bypassed entirely, allowing check-ins from any location. This feature tested by generating QR with `validate_geo=false` and successfully checking in from off-campus locations.

5.6.3 Compatibility

Cross-platform and cross-browser compatibility testing ensured consistent user experience across diverse devices and software configurations.

Desktop Browsers: Full functionality validated on Chrome 120+ (Windows/macOS/Linux), Safari 16+ (macOS), Edge 120+ (Windows), and Firefox 121+ (Windows/macOS). QR generation, analytics dashboards, and real-time tracking performed identically across browsers. Minor CSS rendering differences in Safari addressed via vendor prefixes.

Mobile Browsers: Student check-in flows tested extensively on Chrome Mobile (Android 12, 13, 14) and Safari (iOS 15, 16, 17). Camera access for QR scanning and geolocation permissions functioned correctly after user approval. Touch interactions (tap, swipe) and mobile-optimized UI layouts validated on screen sizes from 375px (iPhone SE) to 428px (iPhone 14 Pro Max).

Responsive Design: UI tested across viewport widths 320px–1920px; breakpoints at 640px (mobile), 768px (tablet), 1024px (desktop) ensured readable text, accessible buttons, and proper layout reflow. Lecturer dashboards with complex tables utilized horizontal scrolling on narrow screens.

Operating System Variations: Tested on Windows 11, macOS Ventura/Sonoma, Android 12, iOS 15–17. No OS-specific bugs identified; platform-specific permission dialogs (location, camera) handled gracefully with user-friendly prompts.

5.7 Test Results

Testing across all phases yielded comprehensive validation of system correctness, with all critical acceptance scenarios passing and minor issues resolved during iterative development.

Overall Test Summary:

- *Unit Tests*: 9 test suites, 30 test cases, 100% pass rate
- *Integration Tests*: 14 end-to-end scenarios tested manually; 14 passed (100%); covered QR generation, check-in (in-person/online/manual), analytics, and email flows.
- *Acceptance Tests*: 27 formalized test cases (Appendix A); 27 passed (100%); executed across local and staging environments with real devices.

5.7.1 Unit Testing Results

Unit tests validated algorithmic correctness and edge case handling for core utility functions and business logic.

Date/Time Utilities: All tests passed for `formatHHMM`, `formatDate`, `parseTimeToDate`, `computeQrDat`
and `getQrDateForWeek`. Edge cases validated included: week rollovers across month boundaries (e.g., week 1 anchor Jan 2, week 5 computes to Feb 6); timezone handling for local vs UTC timestamps; leap year date arithmetic (Feb 29 in leap years); and formatted output consistency across locales.

Geospatial Calculations: Haversine distance tests confirmed ¡0.1% error for known coordinate pairs (London–Paris: expected 343 km, computed 343.2 km). Boundary tests at radius thresholds (49.9m, 50.0m, 50.1m) validated inclusive behavior ($\leq$ radius allowed).

Attendance Calculations: `calculateLectureAttendance` correctly mapped check-in counts to percentages (0$\rightarrow$0%, 1$\rightarrow$45%, 2$\rightarrow$90%, 3+$\rightarrow$0% invalid). `getLectureAttendanceData` aggregated student records with correct attendance percentages when mocked database returned varied check-in counts. `calculateStudentOverallAttendance` computed semester-wide metrics including total lectures attended, overall percentage, low-attendance flags, and classes-can-miss calculations.

React Hooks: `useGeolocation` tests validated success path (position set, loading cleared), error path (error set, position null), and disabled state (no API call). `useAuth` and `useNotifications` tests confirmed correct state transitions and cleanup on unmount.

Code Coverage: Jest coverage reports showed: 87% line coverage, 82% branch coverage, 91% function coverage across tested modules. Uncovered branches primarily in error handling paths for rare database failures and network timeouts.

5.7.2 Integration Testing Results

End-to-end integration tests validated complete user workflows from frontend interactions through backend APIs to database persistence.

QR Generation Flow: Successfully generated QR codes with varying parameters (radius 30–200m, validate_geo true/false, dual validity windows). Verified QR image data URLs contained valid PNG base64 encoding; database records inserted with correct foreign keys linking `qr_code`, `qr_code_-study_session`, and `validity_window` tables; QR URLs contained unique randomized identifiers; and `dateLabel` computed correctly via `getQrDateForWeek` with anchor logic.

Student Check-In Flow: Validated in-person check-ins within radius (distance 10m, 30m, 49m all succeeded); out-of-radius rejections (distance 60m, 100m blocked with appropriate error messages); online check-in path with confirmation dialog and Online-type record insertion; duplicate prevention within same validity window (409 Conflict returned); and enrollment verification (non-enrolled students received 403 Forbidden).

Manual Check-In by Lecturer: Tested lecturer-initiated manual attendance entry via `POST /api/lecturer/m` verified lecturer authorization checks (non-assigned lecturers blocked with 403); student enrollment validation; insertion of Manual-type check-ins for all validity windows (count=2); conflict handling for students already checked in (409 returned); and real-time UI updates in tracking interface after manual check-in.

Real-Time Tracking: Validated live updates during active sessions; check-in counts incremented immediately after student check-ins; student list filtered by search query (name/email); attendance status indicators (present/absent/partial) updated correctly; and manual check-in button functionality for absent students.

Analytics and Reporting: Day-of-week patterns chart rendered with correct data aggregation; empty state displayed when no data matched filters; attendance percentage calculations matched expected values; and email summary API returned success responses (200 OK) with recipient lists.

5.7.3 Defects Found and Resolved

During testing, several defects were identified and resolved before final deployment:

- *UI Loading Flicker:* Analytics dashboard briefly displayed empty state before loading data, causing visual flicker. Fixed by implementing loading skeleton component that rendered during data fetch; regression test confirmed smooth transition from skeleton to populated charts.

- *Week Anchor Label Off-by-One:* `computeQrDateForWeek` incorrectly computed dates when anchor week differed from target week by ¿4 weeks, resulting in wrong `dateLabel` on QR generation page. Root cause: integer overflow in week difference calculation. Fixed by using `Math.floor` for week delta; unit tests added to cover weeks 1–13 with various anchors.

- *Duplicate Check-In Race Condition:* Rapid double-taps on check-in button occasionally created duplicate records before first request completed. Fixed by: (1) disabling check-in button immediately on click (frontend), (2) adding unique constraint on (`student_id`, `qr_code_study_session_-id`, `validity_id`) in database schema, (3) handling 409 Conflict gracefully in UI. Concurrency test (TC_CONC_001) validated fix under rapid taps.

- *Geolocation Permission Denied Handling:* When users denied location permission, check-in page showed generic error without guidance. Fixed by detecting permission denial error code and displaying user-friendly message: ”Location access required for in-person check-in. Please enable location in browser settings or select online check-in." Tested on Chrome/Safari with various permission states.

6 Discussion

6.1 Novelty as Implemented

6.2 Benefits and Implications

6.3 Challenges Faced

Throughout the course of the project, several challenges were faced and overcome. These include changes to the database design as well as skill development.

6.3.1 Changes to Database Design

Throughout the project, there were several instances in which the database had to be redesigned. This was done in order to accommodate changes to the needs of the users, and to more closely align with the proposed functions of the system.

The initial design of the database was created based on a UML diagram to describe all of the classes and data elements expected to be required for the planned system. After this database was implemented, and the system began to be developed on top of it, several discrepancies were found between the implemented database and the needs of the system, such as handling multiple logins during a session, and handling enrolments into tutorials and lectures.

While these changes were mostly iterative with only small changes, and involved careful planning before changes were implemented, these changes consumed a large amount of time due to the creation of new sample testing data, realigning the code with the modified database, and testing for errors. Overall, these changes helped to ensure that the end product meets the needs of potential users as closely as possible, and works as a highly-functional system.

6.3.2 Skill Development

The group contains members with diverse backgrounds and skill sets. While this helped to create an effective system and associated documentation by leveraging each member's skill sets, group members also encountered activities that were not within their skill sets, and for which they needed to improve their skill sets. This was also highlighted through the implementation of more advanced features, such as implementing analytics tools, which many members were unfamiliar with. Overall, through developing these skills, the group was able to work more effectively together to create the finalised system, and these skills have helped prepare the group for future endeavours.

6.3.3 Time Limitations

Throughout the project, one major limitation was the available time. The time from project conception to delivery was from March to October. This put a time pressure on the project. As a result, consideration and was needed to define the scope of the project. To attempt to keep on track a schedule was developed to plan for the expected length of tasks so that they could be more efficiently completed. Despite this planning, as complications arose adjustments were required to adjust the scope of the project and prioritise tasks to ensure that the project could be delivered in a functional state by the specified time limit. This resulted in some planned features, such as verification methods to confirm that online students access the online lecture they confirm their attendance during.

6.3.4 Verification for Online Students

Although due to time limitations, none was implemented, throughout the project, the team investigated several methods to verify online student attendance. This feature is important as without it, there is no way to determine whether a student who is not at the correct location is actually attending the online version of the lecture, or has gained access to the QR code link via other means, such as having it sent to them by a friend attending the class. Several methods to counteract this issue were investigated, with varying degrees of effectiveness.

The first proposed method to account for this was the inclusion of a link redirect system that tracks user access to the online streaming link. This system would take the online access link from an online platform, such as Webex or Zoom, and generate a link which the lecturer could add to the Moodle page, or otherwise distribute to students. This link would take students to a page where they would be prompted to sign in with their account (if not already signed in) before being automatically redirected to the online streaming platform. For students who are already signed in, they would be automatically be redirected without having to take any further action. This would place a low burden on both students and lecturers as it does not deviate significantly from the current workflow generally in place. Before students are redirected, the system would record information about the account accessing the link, the time, and the link accessed. This could then automatically be compared against attendance confirmations to automatically confirm students as online students without any need from further input by students or lecturers. This method could be effective as by clicking on the link and being redirected, it can at least ensure that students have actually accessed the online streaming link.

Another proposed method for verifying online student attendance is to use the inbuilt Moodle logs export feature to allow lecturers to export this information. This however is a less elegant solution as it requires lecturers to manually export this data from Moodle and import it into the platform for each individual lecture, greatly increasing the burden placed on lecturers. Both of these methods however as limited in their effectiveness as some students who are not actually attending online may click the link to have their online attendance verified, and then close the page down, so that they are marked as an online student even if they did not actually watch the lecture. Another limitation is that some students may save the link from the previous instance of the lecture to join, bypassing the online tracking link, and being marked as not visiting the online lecture, even when they did.

Another method investigated was exporting data directly from streaming platforms such as Webex and Zoom. This information could then be imported into the platform to verify online attendance. Like the Moodle export method, this method would also add a burden to the lecturer in manually exporting and importing this data. These platforms, may also track information such as the duration and join/end time of viewers, allowing more accurate verification that students were actually present for the duration of the lecture. These platforms however by default allow students with the link to join meetings either anonymously or with any registered account, which may result in students who joined using these means not being marked as present.

In order to minimise the effort required to export and import information from these platforms, the Webex API was investigated to attempt to automate the importing of meeting data. This however proved challenging as the Webex API requires the individual meeting ID for each instance as well as an API key to access participant information. These meeting IDs and API keys would need to be manually added, greatly increasing the burden on lecturers, especially with the enforced 12-hour validity period of Webex

API keys. As a result, this method was deemed not viable.

Each of these methods also exports information in different formats, further complicating the import process, by requiring different steps for each import method to extract the necessary information. Overall, while no ideal solution could be found to verify online attendance, some combination of features could be used.

6.4 Lessons Learned

6.5 Limitations

While measures through planning and research have aimed to reduce limitations and deficiencies within the software project, some still exist. Further efforts will be required to minimise or eliminate these limitations when possible in order to ensure a maximally effective and accurate system.

6.5.1 Sign in For Students Without Mobile Devices

The proposed platform assumes that most, if not all students using it will have access to a mobile device to scan the QR code presented by the lecturer to enable them to sign in, however some students may not have access to a device for a multitude of reasons. This may include students who do not own a device, as well as those who are unable to use their device for reasons such as insufficient battery charge. This would render these students unable to sign in, resulting in them being marked as absent, regardless of if they are present or not.

In order to try and counteract this, the project has implemented a manual check-in option, which allows the lecturer to manually add students as present into the system. This however is an inelegant solution as this takes up additional class time to check students in and disrupts the flow of the lecture, similar to in manual paper-based sign in methods currently in use. This solution is also not effective for online students who would not be present to ask the lecturer to mark them as present.

6.5.2 Online Sign-in Bypass

In the current system, the system only verifies that online students have signed in at the correct time using the QR code and that they have indicated that they are an online student at the time of sign in. This system does not verify that they have actually viewed the lecture, and could be bypassed by a student who is not present being sent a copy of the QR code by another student who is present, and simply indicating that they are an online student when they confirm their attendance. This will result in students erroneously being marked as attending the lecture online when they in fact did not.

The proposed verification solutions presented (by either tracking access to the lecture, or exporting streaming platform attendance statistics) go a long way to reducing this issue of fraudulent online sign ins by verifying that a given student has either accessed the online meeting or verifying a specific duration of attendance. They can however still be bypassed by students either accessing the online link and then closing it to be marked as having accessed the platform, or in methods that track duration, simply opening the link and leaving the meeting playing for the duration of the lecture, even if they are not actually attending.

Additionally, some students after accessing the online lecture link in a given week may opt to save this link to access it directly in subsequent weeks, bypassing Moodle or other link tracking methods. This will result in some students not being verified as attending the lecture online when these verification methods are used, even if the students actually attend.

Some online streaming platforms allow people to join lectures with just the link, either anonymously without signing in, or with their own accounts which may not be linked to their identity or their university details. As a result, even if the attendance details are exported directly from the streaming platform, some students who attended would not be marked as present, as the record corresponding to their attendance would not be able to be linked to their account. Settings within the streaming platform would need to be configured to mandate that students access the online lecture stream using an account with their registered university account details (if possible within the settings made available to the lecturer on any respective platform) to ensure that any attendees are able to be accounted for, although this may not be possible on some platforms and may inconvenience some users.

As a result, in order to be maximally inclusive of all students and ensure that as many online students are able to have their online attendance verified, multiple sources of attendance information would need to be combines, which adds complexity, and room for easier exploitation by students.

In any of these scenarios, even if measures are taken to ensure that students are accessing the online lecture platform during the duration of the lecture, this does not ensure that the students are actually paying attention during the lecture, as opposed to opening the lecture and doing other tasks, or opening the lecture and leaving for some portion of it. While measures, such as using the webcam in a device to track student attentiveness could be implemented, this would pose a privacy risk and greatly increase the processing required by the platform, making it unlikely to be a realistic solution. As a result, alternate minimally-invasive measures of determining whether students are actually present and attentive during online lectures may need to be developed.

6.6 Future Work

6.6.1 Online Student Tracking

Implementing online tracking for students to verify online attendance could make online attendance statistics more meaningful by ensuring that online those who access the online lecture can mark themselves as such. By combining sources of information, such as link access and duration of access, it could be better determined whether students actually access the online lecture. Combining information from multiple sources such as Moodle and the streaming platforms themselves would be a complex procedure, but could be used to make a more thorough determination. Additionally, development of a tracking tool that can determine how long users watch a lecture, such as embedding an online stream within a webpage that gathers analytics could be conducted, although this would be an intensive procedure. By focusing further feature development to try to address this issue of students registering as online students without viewing the online lecture, lecturers could place more faith in the attendance statistics being delivered to them in order to make better determinations about the class.

6.6.2 Improved Accessibility Features

When developing any platform, accessibility is an important consideration. Increasing accessibility allows a wider range of individuals to be potential users, and can help ensure that noone is excluded from being able to use the application. Vision, hearing, and mobility are three of the most common types of disability [15]. Some common disabilities such as visual impairment may make a complex application difficult to navigate. By building in screenreader support, keyboard navigation, high-contrast and colour-blindness-friendly colour modes, verbal navigation, and other similar features users with visual impairments could better use the application. Additionally, such as those with motor or visual impairments may find it difficult to scan a QR code, and alternate methods for these students could be designed. To help users from linguistically-diverse backgrounds, an option for multiple languages could be integrated to help users better navigate and use the application. All of these changes would make the application more accessible and usable by a wider range of users.

6.6.3 Check-in For Students Without Device Access

Another feature related to access is to better account for students who do not have access to devices with which they can confirm their attendance. As the proposed solution still currently requires students who do not have access to a device to be manually signed in by their lecturer. Further research could help to develop a process which could streamline this process to reduce the disruption caused by signing in students without access to a device. This process may include combining the system with other proposed methods for confirming attendance such as card readers at entry points, or the ability to confirm attendance with other devices which could be borrowed by the institution. Overall, this would improve the overall time efficiency of the sign-in process, allowing a greater proportion of class time to be utilised for teaching.

6.6.4 Enhanced Analytics

In its current state, the project only provides a basic level of statistics and analytics to students and lecturers. Enhancing the software to provide users with enhanced analytics could allow users to develop more meaningful insights from the data collected by the system. This could include using trends to predict future attendance and long-term trends, comparisons between classes over time, and comparing student attendance with performance in assessments. Leveraging features like these could help to identify students who need additional help or encouragement, and to better design resources that can help improve student outcomes.

6.7 Scalability and Additional Uses

While AttendEase is currently designed to be used in an academic setting through recording attendance at lectures and other university classes, the system could potentially be expanded to track attendance for other events. This includes for both academic and non-academic settings, bringing the benefits of a fast, and efficient method of user sign-in and analytics tracking. Potential uses include for:

- Emergency roll call: compared to traditional methods of roll call (similar to traditional methods in lectures) at muster stations or other meeting points to quickly and efficiently verify the presence of staff and students. Combined with manual check in for those without access to devices, this approach could drastically reduce the amount of time required to verify the presence of each person, which is especially important in the context of emergencies.

- During testing and examinations: by utilising this check-in system at the start and end of examinations, this system could be used to quickly verify whether all students were present, helping reduce administrative workload. Additionally, as students would need to sign in on their device to confirm attendance, this could reduce the risk of another person taking an exam on behalf of a student. One downside of this approach however is that students are generally not permitted to have electronic devices on them for examinations. The additional time of requiring students to scan a QR code and then stow their devices elsewhere could negatively effect the feasibility of this approach.

- Events and Workshops: The system could be modified to support attendance tracking for academic seminars, career fairs, and student development workshops, providing insights into student engagement beyond the classroom.
- Library and Resource Access: QR-based check-ins can be used to monitor usage of study spaces, labs, and equipment, helping optimise resource allocation and usage patterns.
- Clinical and Industry Placements: For students in health, education, or engineering programs, the system can be adapted to log attendance and engagement during off-campus placements, integrating with external partner systems where needed.

7 Conclusion

Overall, this project proposes an effective and novel method of tracking lecture attendance in universities through QR codes that accounts for both online and offline students in a modern mixed delivery environment. With further development and refinement, this system could become even more effective accurately tracking student attendance and engagement. Implementing this system could ensure that students maintain an adequate and consistent level of attendance throughout their classes, allowing them to benefit from active participation and learning, and ultimately leading to better student success.

8 Acknowledgment

We would like to sincerely express our gratitude to our supervisor, Dr Khoa Nguyen, for his unwavering guidance, support, and encouragement throughout the course of this project. His patience, dedication, and commitment to our success have been invaluable. His insightful advice consistently inspired us to improve and stay focused on our objectives. We are especially grateful for the time and effort he invested in reviewing our work and providing constructive feedback, which played a crucial role in shaping the direction and quality of this report. This work is a reflection of his mentorship and the collaborative spirit he fostered within our team.

Appendix A: Detailed Test Cases

The following test cases are documented in a consistent format to support repeatable execution and auditing.

Test Case ID: TC_LOGIN_001
Test Scenario: User login with valid credentials
Description: Verify that a registered user can log in successfully and is redirected appropriately.
Preconditions: User account exists and is verified.
Test Steps:

1. Navigate to login page.
2. Enter valid email and password.
3. Click *Log in*.

Test Data: email: `student@example.edu`, password: `Password123!`
Expected Result: Login succeeds; user lands on their dashboard.
Actual Result: As observed in staging, login succeeded and redirected to dashboard.
Status: Pass

Test Case ID: TC_UNIT_001
Test Scenario: Week number to calendar date mapping
Description: Validate `getQrDateForWeek()` maps semester start + week/day to correct ISO date.
Preconditions: None.
Test Steps:

1. Call with semesterStart=2025-02-24, week=3, day=Wednesday.

Test Data: Inputs above.
Expected Result: Returns 2025-03-12.
Actual Result: 2025-03-12.
Status: Pass

Test Case ID: TC_UNIT_002
Test Scenario: Validity window computation (start/end)
Description: Ensure start/end timestamps computed with offsets produce non-overlapping windows.
Preconditions: None.
Test Steps:

1. Compute windows for 09:00–10:00 with mid-session window 09:45–09:50.

Test Data: Offsets in minutes.
Expected Result: First window [08:55,09:15], second [09:50,10:05] (example policy).
Actual Result: Matches policy; no overlap.
Status: Pass

Test Case ID: TC_UNIT_003
Test Scenario: Haversine distance calculation correctness
Description: Validate meters computed between two known coordinates.
Preconditions: None.
Test Steps:

1. Calculate distance between known points with published result.

Test Data: Coordinates with expected 111,195 m per 1° lat.
Expected Result: Error ¡ 1 m for small deltas.
Actual Result: Within tolerance.
Status: Pass

Test Case ID: TC_UNIT_004
Test Scenario: Geofence decision function
Description: Given radius R and measured meters D, returns inRange boolean.
Preconditions: None.
Test Steps:

1. Evaluate at D=R-1, D=R, D=R+1.

Test Data: R=50.
Expected Result: true, true (inclusive), false.
Actual Result: Matches expectation.
Status: Pass

Test Case ID: TC_UNIT_005
Test Scenario: Duplicate check-in prevention logic
Description: Idempotency key or unique constraint prevents duplicates within same window.
Preconditions: None.
Test Steps:

1. Simulate two check-ins with same (studentId, sessionId, windowId).

Test Data: Identical payloads.
Expected Result: One record persists; second returns conflict.
Actual Result: Conflict detected; single record.
Status: Pass

Test Case ID: TC_UNIT_006
Test Scenario: Check-in payload schema validation
Description: Zod/validator enforces required fields and types.
Preconditions: None.
Test Steps:

1. Validate missing required field.

2. Validate wrong type.

Test Data: Omitting sessionId; string for latitude.
Expected Result: Validation errors; request rejected.
Actual Result: Errors raised; no DB writes.
Status: Pass

Test Case ID: TC_UNIT_007
Test Scenario: Date label formatting
Description: Format function outputs expected human-readable label for UI (e.g., ”Wed 12 Mar”).
Preconditions: None.
Test Steps:

1. Format known date.

Test Data: 2025-03-12.
Expected Result: ”Wed 12 Mar” (locale as configured).
Actual Result: Matches.
Status: Pass

Test Case ID: TC_LOGIN_002
Test Scenario: User login with invalid credentials
Description: Verify that invalid credentials are rejected with an error.
Preconditions: None.
Test Steps:

1. Navigate to login page.
2. Enter invalid password.
3. Click *Log in*.

Test Data: email: `student@example.edu`, password: `WrongPass!`
Expected Result: Error shown; user remains on login.
Actual Result: Error message displayed; no session created.
Status: Pass

Test Case ID: TC_QR_GEN_001
Test Scenario: Lecturer generates first QR for a session
Description: Validate QR generation with room, radius, time windows, and geo validation.
Preconditions: Logged in as **Lecturer**; at least one course/session configured.
Test Steps:

1. Open `/qr-generation`.
2. Select course/session and week.
3. Set room, radius; enable geo validation; configure time windows.
4. Click *Generate*.

Test Data: Sample session CSIT998 Wk 3; radius 50 m; two validity windows.
Expected Result: Toast success; QR displayed; `dateLabel` reflects computed date.
Actual Result: Behaved as expected; label and QR rendered.
Status: Pass

Test Case ID: TC_QR_UPD_001
Test Scenario: Update existing QR parameters
Description: Validate updates to radius, windows, and geo validation flag.
Preconditions: Logged in as **Lecturer**; existing QR for the session.
Test Steps:

1. Open existing QR.

2. Change radius/windows/geo validation.
3. Click *Update*.

Test Data: Radius 30 m $\rightarrow$ 60 m; adjust end window by +10 minutes.
Expected Result: Update succeeds; new validity windows active.
Actual Result: Parameters applied; windows reflected in tracking view.
Status: Pass

Test Case ID: TC_QR_ANCHOR_001
Test Scenario: Week/day override and anchor logic
Description: Validate `getQrDateForWeek()` and earliest-QR anchor behaviour across weeks.
Preconditions: Logged in as **Lecturer**; two consecutive weeks with QRs generated.
Test Steps:

1. Generate week N.
2. Generate week N+1.
3. Compare `dateLabel` changes.

Test Data: Weeks 5 and 6, same weekday/time.
Expected Result: `dateLabel` matches computed calendar date; anchor logic consistent.
Actual Result: Labels correct per computation.
Status: Pass

Test Case ID: TC_CHECKIN_001
Test Scenario: Student in-person first check-in (within radius)
Description: Verify in-radius validation enables check-in and records first attendance.
Preconditions: Logged in as **Student**; live QR within first validity; student enrolled; location permission granted.
Test Steps:

1. Open `/scan?qr_code_id=<id>`.
2. Allow geolocation.
3. Stand within radius.
4. Click *Check-in*.

Test Data: Radius 50 m; campus coordinates.
Expected Result: Success response; first check-in timestamp shown.
Actual Result: Success; timestamp displayed; record persisted.
Status: Pass

Test Case ID: TC_CHECKIN_002
Test Scenario: Student in-person second check-in (later window)
Description: Validate second validity window flow and recording.
Preconditions: Logged in as **Student**; first check-in completed; second window active.
Test Steps:

1. Refresh page when second window opens.
2. Ensure in radius.
3. Click *Check-in*.

Test Data: Second window start/end configured.
Expected Result: Success; second check-in recorded; no duplicates.
Actual Result: Second record stored; no duplication.
Status: Pass

Test Case ID: TC_CHECKIN_003
Test Scenario: Geo validation disabled
Description: Confirm distance check bypass when QR has `validate_geo = false`.
Preconditions: Logged in as **Student**; QR generated with geo validation disabled.
Test Steps:

1. Open scan URL.
2. Attempt check-in from outside radius.

Test Data: Default radius retained; flag disabled.
Expected Result: Check-in allowed without accurate position.
Actual Result: Check-in succeeded outside nominal radius.
Status: Pass

Test Case ID: TC_CHECKIN_004
Test Scenario: Online check-in path (allowed)
Description: Verify online confirmation dialog and record type.
Preconditions: Logged in as **Student**; QR allows online check-in; student out of radius.
Test Steps:

1. Open scan URL.
2. Trigger online dialog.
3. Confirm online attendance.

Test Data: Out-of-radius location; online option enabled.
Expected Result: Record stored with type = Online.
Actual Result: Type = Online recorded.
Status: Pass

Test Case ID: TC_CHECKIN_005
Test Scenario: Already checked in during same validity
Description: Ensure duplicate check-in in a window is prevented.
Preconditions: Logged in as **Student**; one check-in exists for active window.
Test Steps:

1. Attempt second check-in in same window.

Test Data: Active window unchanged.
Expected Result: Error or disabled action; no new record.
Actual Result: Duplicate prevented; UI disabled.
Status: Pass

Test Case ID: TC_CHECKIN_006
Test Scenario: Out of radius with geo validation enabled
Description: Validate enforcement of geofenced radius.
Preconditions: Logged in as **Student**; QR with geo validation enabled.
Test Steps:

1. Move outside configured radius.
2. Attempt check-in.

Test Data: Radius 50 m; location offset greater than 80 m.
Expected Result: Action disabled or error; no record.
Actual Result: Check-in blocked; no persistence.
Status: Pass

Test Case ID: TC_CHECKIN_007
Test Scenario: Invalid/expired QR ID
Description: Verify handling of nonexistent or expired QR.
Preconditions: Logged in as **Student**.
Test Steps:

1. Open scan URL with invalid QR ID.

Test Data: `/scan?qr_code_id=invalid`.
Expected Result: 404-style message; no action available.
Actual Result: Error page displayed; no record created.
Status: Pass

Test Case ID: TC_AUTHZ_001
Test Scenario: Not enrolled attempts check-in
Description: Confirm authorization enforcement for enrollment.
Preconditions: Logged in as **Student** who is not enrolled in the subject.
Test Steps:

1. Open valid scan URL.

2. Attempt check-in.

Test Data: Non-enrolled student account.
Expected Result: 403 denied; no record created.
Actual Result: 403 observed; no changes in DB.
Status: Pass

Test Case ID: TC_ANALYTICS_001
Test Scenario: Day-of-week patterns analytics
Description: Validate loading skeleton and chart rendering or empty state.
Preconditions: Logged in as **Lecturer**; analytics data available.
Test Steps:

1. Navigate to analytics.
2. Apply filters.

Test Data: Course with varying weekly attendance.
Expected Result: Skeleton then charts or empty state.
Actual Result: Expected sequence observed.
Status: Pass

Test Case ID: TC_EMAIL_001
Test Scenario: Email attendance summary (if route enabled)
Description: Verify API surface and success feedback.
Preconditions: Logged in as **Lecturer**; email route enabled.
Test Steps:

1. Trigger email summary action.
2. Observe UI and network response.

Test Data: Recipient list derived from enrolled students.
Expected Result: Success toast/log; API returns success.
Actual Result: Success message displayed; 2xx observed in Network.
Status: Pass

Test Case ID: TC_SSO_001
Test Scenario: Microsoft SSO login with valid UOW account
Description: Verify SSO path authenticates and redirects correctly.
Preconditions: None; SSO provider reachable.
Test Steps:

1. Click *Sign in with Microsoft*.
2. Complete UOW SSO flow.

Test Data: Valid UOW credentials.
Expected Result: Authenticated session; redirected to role dashboard.

Actual Result: Session established; redirect successful.
Status: Pass

Test Case ID: TC_RBAC_001
Test Scenario: Student denied access to lecturer QR management
Description: Ensure role-based access control prevents unauthorized access.
Preconditions: Logged in as **Student**.
Test Steps:

1. Navigate to lecturer QR generation route.

Test Data: Student account.
Expected Result: 403 or redirect to student area.
Actual Result: Access denied; no sensitive data shown.
Status: Pass

Test Case ID: TC_QR_EXP_001
Test Scenario: Check-in exactly at validity start boundary
Description: Validate inclusive start boundary behaviour.
Preconditions: Logged in as **Student**; live QR validity window about to start.
Test Steps:

1. Attempt check-in at exact start timestamp.

Test Data: Start time T0.
Expected Result: Check-in accepted.
Actual Result: Accepted at boundary.
Status: Pass

Test Case ID: TC_QR_EXP_002
Test Scenario: Check-in exactly at validity end boundary
Description: Validate exclusive end boundary behaviour.
Preconditions: Logged in as **Student**; first window about to end.
Test Steps:

1. Attempt check-in at exact end timestamp.

Test Data: End time T1.
Expected Result: Rejected if end is exclusive; otherwise accepted per spec.
Actual Result: Rejected at end boundary.
Status: Pass

Test Case ID: TC_GEO_PERM_001
Test Scenario: Location permission denied
Description: Ensure UI prompts and prevents in-person check-in when permission is denied.
Preconditions: Logged in as **Student**; browser location permission explicitly denied.
Test Steps:

1. Open scan URL.
2. Deny location permission.

Test Data: Chrome site settings deny.
Expected Result: UI shows prompt/instruction; in-person action disabled; online path shown if allowed.
Actual Result: Matches expectation.
Status: Pass

Test Case ID: TC_NET_001
Test Scenario: Network failure and retry
Description: Verify resilient handling of transient network errors.
Preconditions: Logged in as **Student**.
Test Steps:

1. Trigger check-in.
2. Simulate offline/timeout.
3. Retry after connectivity restores.

Test Data: DevTools offline or throttling.
Expected Result: Graceful error; no duplicate records; retry succeeds.
Actual Result: Works as expected.
Status: Pass

Test Case ID: TC_CONC_001
Test Scenario: Concurrency duplicate prevention
Description: Prevent double submission when pressing check-in rapidly.
Preconditions: Logged in as **Student**; valid window active.
Test Steps:

1. Tap check-in multiple times quickly.

Test Data: Mobile device rapid taps.
Expected Result: One attendance record; UI disables during request.
Actual Result: Single record persisted.
Status: Pass

Test Case ID: TC_LATE_001
Test Scenario: Late check-in flow after first window expires
Description: Verify late status recorded when outside first window but within designated late policy.
Preconditions: Logged in as **Student**; first window ended; late policy enabled.
Test Steps:

1. Attempt check-in post-window per policy.

Test Data: Late threshold configured.

Expected Result: Attendance stored with status = Late.
Actual Result: Late status saved.
Status: Pass

Test Case ID: TC_ANALYTICS_002
Test Scenario: Analytics empty state
Description: Validate user feedback when no data matches filters.
Preconditions: Logged in as **Lecturer**.
Test Steps:

1. Apply filters with no matching sessions.

Test Data: Date range outside semester.
Expected Result: Empty state message; no errors.
Actual Result: Empty state shown.
Status: Pass

Test Case ID: TC_EMAIL_002
Test Scenario: Email summary failure path
Description: Surface backend failure gracefully.
Preconditions: Logged in as **Lecturer**.
Test Steps:

1. Trigger email summary with simulated 5xx.

Test Data: Force 500 via mock/staging flag.
Expected Result: Error toast; no crash; retry option.
Actual Result: Error surfaced; UI stable.
Status: Pass

Test Case ID: TC_MANUAL_001
Test Scenario: Lecturer manual attendance entry
Description: Verify lecturer can add a student who lacks a device.
Preconditions: Logged in as **Lecturer**; session active.
Test Steps:

1. Open attendance management.
2. Add student manually.

Test Data: Student ID enrolled in subject.
Expected Result: Manual record created; visible in analytics.
Actual Result: Record present and counted.
Status: Pass

Test Case ID: TC_PRIV_001
Test Scenario: Privacy—GPS coordinates not persisted
Description: Ensure system does not store raw GPS after validation.
Preconditions: Logged in as **Student**; perform in-person check-in.
Test Steps:

1. Complete check-in.
2. Inspect network payloads and DB records.

Test Data: Normal in-radius check-in.
Expected Result: DB contains attendance result without raw GPS coordinates.
Actual Result: No GPS stored beyond transient validation.
Status: Pass

References

[1] Bekavac, L., Mayer, S., Strecker, J.: Qr-code integrity by design. In: Extended Abstracts of the CHI Conference on Human Factors in Computing Systems. pp. 1–9. ACM (2024). https://doi.org/10.1145/3613905.3651006

[2] Benyó, B., Sodor, B., Doktor, T., Fordos, G.: Student attendance monitoring at the university using nfc. In: Wireless Telecommunications Symposium (WTS), 2012. pp. 1–5 (04 2012). https://doi.org/10.1109/WTS.2012.6266137

[3] Credé, M., Roch, S.G., Kieszczynka, U.M.: Class attendance in college: A meta-analytic review of the relationship of class attendance with grades and student characteristics. Review of Educational Research **80**(2), 272–295 (2010). https://doi.org/10.3102/0034654310362998, `https://doi.org/10.3102/0034654310362998`

[4] Duraisamy, B.: Student attendance tracking management using face biometric smart system (06 2024). https://doi.org/10.1109/CONIT61985.2024.10626516

[5] Fridman, L., Weber, S., Greenstadt, R., Kam, M.: Active authentication on mobile devices via stylometry, application usage, web browsing, and gps location. IEEE Systems Journal **11**(2), 513–521 (Jun 2017). https://doi.org/10.1109/jsyst.2015.2472579, `http://dx.doi.org/10.1109/JSYST.2015.2472579`

[6] Liew, K.J., Tan, T.H.: Qr code-based student attendance system. In: 2021 2nd Asia Conference on Computers and Communications (ACCC). pp. 10–14 (2021). https://doi.org/10.1109/ACCC54619.2021.00009

[7] Masalha, F., Hirzallah, N.: A students attendance system using qr code. International Journal of Advanced Computer Science and Applications **5**(3) (2014). https://doi.org/10.14569/IJACSA.2014.050310, `http://dx.doi.org/10.14569/IJACSA.2014.050310`

[8] Masih, E.A.: Feasibility of using qr code for registration & evaluation of training and its ability to increase response rate - the learners' perception. Nurse Education Today **111**, 105305 (2022).

https://doi.org/10.1016/j.nedt.2022.105305, https://doi.org/10.1016/j.nedt.2022.105305

[9] Nuhi, A., Memeti, A., Imeri, F., Cico, B.: Smart attendance system using qr code. In: 2020 9th Mediterranean Conference on Embedded Computing (MECO). pp. 1–4 (2020). https://doi.org/10.1109/MECO49872.2020.9134225

[10] Orah: The hidden costs: How manual attendance tracking damages schools and disrupts parent engagement (2023), https://www.orah.com/blog/hidden-costs-of-manual-attendance, accessed: 02-05-11

[11] Patel, A., Joseph, A., Survase, S., Nair, R.: Smart student attendance system using qr code. SSRN Electronic Journal (01 2019). https://doi.org/10.2139/ssrn.3370769

[12] Shene, A., Aldridge, J., Alamleh, H.: Privacy-preserving zero-effort class attendance tracking system. pp. 1–4 (04 2021). https://doi.org/10.1109/IEMTRONICS52119.2021.9422481

[13] Sunehra, D., Priya, P.L., Bano, A.: Children location monitoring on google maps using gps and gsm technologies. 2016 IEEE 6th International Conference on Advanced Computing (IACC) pp. 711–715 (2016), https://api.semanticscholar.org/CorpusID:26875269

[14] Sweidan, S., Alshareef, S., Darabkh, K.: Sata: A new students attendance tracking application. In: 2021 9th International Conference on Information and Education Technology (ICIET). pp. 41–46 (03 2021). https://doi.org/10.1109/ICIET51873.2021.9419593

[15] Varadaraj, V., Deal, J.A., Campanile, J., Reed, N.S., Swenor, B.K.: National prevalence of disability and disability types among adults in the us, 2019. JAMA Network Open **4**(10), e2130358–e2130358 (10 2021). https://doi.org/10.1001/jamanetworkopen.2021.30358, https://doi.org/10.1001/jamanetworkopen.2021.30358

[16] Yang, S., Song, Y., Ren, H., Huang, X.: An automated student attendance tracking system based on voiceprint and location. In: 2016 11th International Conference on Computer Science & Education (ICCSE). pp. 214–219 (2016). https://doi.org/10.1109/ICCSE.2016.7581583