DEVELOPMENT OF A STUDENT ATTENDANCE AND REPORTING SYSTEM FOR GODFREY OKOYE UNIVERSITY USING MACHINE LEARNING

BY

EZE CLETUS NKECHUKWU

GOU/U22/CSC/777

DEPARTMENT OF COMPUTER SCIENCE AND MATHEMATICS,

FACULTY OF NATURAL SCIENCES AND ENVIRONMENTAL STUDIES,

GODFREY OKOYE UNIVERSITY, ENUGU STATE.

IN PARTIAL FULFILLMENT OF THE AWARD OF BACHELOR OF SCIENCE (B.SC),

DEGREE IN COMPUTER SCIENCE.

SUPERVISOR: DR. FRANK OKEBANAMMA

APRIL, 2025.

APPROVAL PAGE

This research, titled " DEVELOPMENT OF A STUDENT ATTENDANCE AND REPORTING SYSTEM FOR GODFREY OKOYE UNIVERSITY USING MACHINE LEARNING," has been assessed and approved by the Department of Computer Science and Mathematics Committees in the Faculty of Natural Sciences and Environmental Studies, Godfrey Okoye University, Enugu, Nigeria.

	Signature	Date
Dr. Frank Okebanama (Supervisor)	.....	.....
External Examiner	.....	.....
Dr. Frank Okebanama (Sub-Dean, HOD, Department of Computer Science and Mathematics)	.....	.....
Prof. I. A. Ajah Dean, Faculty of Computing And Information Technology.	.....	.....

CERTIFICATION

I certify that I completed the research included therein and that it was primarily based on my observations and investigation. Consequently, I will fully accept the University's decision if I am found to have cheated on the assignment.

_____

EZE CLETUS NKECHUKWU

GOU/U22/CSC/777

DEDICATION

This project is dedicated to my parents and family, whose unwavering support, sacrifices, and words of encouragement made this academic journey possible, and to every student and administrator in Nigerian universities who deserves a better, fairer, and more transparent system for managing academic records.

ACKNOWLEDGEMENTS

First and foremost, I give all glory and honor to Almighty God who alone through his provision and grace brought me through every phase of this research. I express my deepest appreciation to my supervisor, Dr. Frank Okebanamma whose advice, patience, criticism and comments during this research helped make the difference. His involvement and willingness to always have my ideas challenged me through this project.

I also thank Dr. Frank Okebanama, HOD Computer Science & Mathematics as well as all other lecturers in the Computer Science & Mathematics department whose academic teachings throughout my undergraduate career were fundamental to this project. To the lecturers whose work inspired me to find new ways of thinking, I say thank you.

My profound gratitude to my parents and family for the paying the sacrifices my education required. Thank you for believing in me at the times I felt like giving up.

Finally, my friends and classmates who provided assistance in testing the application, and with comments and ideas throughout this project, I thank you.

ABSTRACT

Manual attendance management at Godfrey Okoye University is operationally inefficient, suffers from proxy fraud, and generates data that arrives too late for early intervention with poor attendees, thus impeding enforcement the National Universities Commission (NUC) minimum attendance policy of 75% for all students to qualify to take examination. This study aimed to design, develop, and deploy SmartSync, a cloud-native Smart Attendance and Reporting System, to replace the existing paper-based attendance procedure with a secure, automated, and real-time digital alternative for Godfrey Okoye University. The study was carried out with the use of the Incremental Model Development within the Systems Development Life Cycle (SDLC) framework. The Backend Application Programming Interface (API) was built with the Go programming language, while the frontend was developed using Next.js and React. To combat attendance fraud, SmartSync restricts scans to one device per registered student, through the usage of device binding, and employ rotating Quick Response (QR) codes that change every five seconds and cryptographically-signed via Hash-based Message Authentication Code (HMAC-SHA256) using Secure Hash Algorithm 256-bit (SHA-256) to mitigate the usage of proxy based on screenshots, the system has a 3-state machine (Absent, Signed-In, Completed) to authenticate both scan in and scan out. A non-blocking Asynchronous Processing Queue system which leverages Redis Streams with retry and dead-letter mechanisms was used to mitigate attendance-based traffic spikes. Also, a Server-Sent event (SSE) infrastructure was put in place to stream students' rosters to lectures with almost instantaneous display. Finally, a machine learning classifier (Extreme Gradient Boosting (XGBoost)) was trained on a weekly attendance feature set and is deployed in the system to identify students who are at high risk of failing courses so as to allow for early intervention. Docker Compose is used to provision, deploy and containerize the whole stack and is exposed to the internet through over Hypertext Transfer Protocol Secure (HTTPS) using Caddy as a reverse proxy with automatic certificate updates. SmartSync has passed all tests and is capable of successfully thwarting proxy attempts on mock-run tests, correctly imposing the required states, and concurrently handling a massive number of students with minimal delays and absolutely no data loss. On the deployed machine, it was proven that our machine Learning (ML) model's accuracy in detecting risky attendees has an 89% precision, 91% recall,90% F1-score, the harmonic mean of precision and recall, and an Area Under the Receiver Operating Characteristic Curve (AUC-ROC) of 0.94 score on validation data, all of which exceeded our initial expectations. SmartSync demonstrates that a combination of layered security controls, asynchronous queue-based processing, real-time streaming, and machine learning classification can effectively eliminate the core weaknesses of manual attendance management, offering a scalable, fraud-resistant, and data-driven attendance solution for Godfrey Okoye University and similar institutions.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER ONE
INTRODUCTION

1.1 Study to the Background

The attendance of students plays a significant role in higher institutions, it has a direct link with students' performance and the success of an entire cohort (Nguyen and Tran, 2024). The attendance recorded in Godfrey Okoye University is currently being done in the traditional manual method by either passing physical signed attendance sheets among students or reading out students' names for attendance marking. For several decades, the method of taking attendance was as it is now, but with increasing number of students and high administrative burden in record keeping it is no longer efficient.

With the manual system it involves an unacceptably large amount of repetitious work to be done that takes valuable lecture time, taking more than 10 to 15 minutes per session when circulation of attendance sheets is performed and attendance is marked by a number of missing entries in the sheet are left blank and in addition, physical documents are easily misplaced, spoiled or destroyed (Al-Shayea et al. (2022)) and additionally, the system is prone to all sorts of academic dishonesty especially "Proxy signing", the system only stores information until end of semester at which point no real time academic decision can be made based on attendance.

To tackle the issue at hand, this project introduces SmartSync which is a cloud-native Attendance and Reporting System build with Go, Next.js, Redis and a Caddy HTTPS reverse proxy to serve on Docker Compose and finally, the use of an XGBoost ML classifier to analyze attendance information weekly and flag students at risk of leaving the education system due to poor attendance.

1.2 Problem Statement

In a technological driven Nigerian higher education environment, lecture attendance is a major challenge that needs to be addressed at Godfrey Okoye University due to the existence of several issues:

1. **Cheating and Proxy Attendance**: Existing manual attendance records are easily forged using paper as students can easily mark for their friends. However even digital systems are still failing as students now use WhatsApp groups to share static QR Codes

which students with low attendance marks use for proxy signing of others. The system has also failed to detect students that scan a QR Code and leave the session (Hendrawan et al. (2025)).

2. **Administrative Inefficiency and Miscounts:** At the end of semester, lecturers and administrative officers have to verify the number of lectures attended by students against the NUC minimum requirement (75%), the process is extremely tedious and susceptible to manual miscalculations which occur 5-10% of the time (Odeyemi et al. (2025)).
3. **Data Latency and Inert Records:** attendance information generated is inert, there is no mechanism for the lecturer to view any current status about attendance, therefore early intervention for students who might be falling behind is missed (Zhang and Liu, 2024).
4. **No early warning for at-risk students:** Students who have an attendance pattern that put them at the risk of dropping out, is only detected at the end of semester after it's too late for students to know that they might be dropped and might not write the exams.

1.3 Aim and Objectives of the Study

The aim of this study is to develop a student attendance and reporting system using machine learning in Godfrey Okoye University.

The objectives are to:

1. Analyze and identify current student attendance management system within the institution.
2. Design and build a secure web based attendance system for both students and lecturers.
3. Implement rotating QR codes and device verification to prevent proxy attendance and Develop automated reports for a better student attendance monitoring.
4. Build a machine learning model to detect students who may drop-out due to attendance.

1.4 Significance of the Study

This research work will be relevant to the following stakeholders at Godfrey Okoye University:

- **Administration**: Through this research, the institution will have a tamper-proof attendance database that ensures the effective implementation of the university's 75% rule.
- **Lecturers**: it will take the stress off the lecturer by removing the manual lecture attendance and also, information would be readily available about which student is at risk of dropping out so he/she could act proactively.
- **Students**: Real time visibility into the attendance record for students. This would serve as a morale booster for students.
- **Academic Community**: This work will also serve as an example for application of advanced algorithms like XGBoost within EDM.

1.5 Scope of the Study

The scope of this study covers the design, development, and deployment of the SmartSync system across six functional areas. On the backend, the system includes an Application Programming Interface (API) responsible for attendance tracking, user authentication, session control, device binding, dynamic Quick Response (QR) code generation, asynchronous queue management, live data streaming to the frontend, analytics, and the machine learning scheduler. On the frontend, the system provides a role-based web application with dedicated dashboards for students, lecturers, and administrators, including attendance compliance indicators and at-risk flags sourced from the machine learning classifier. The machine learning component consists of an Extreme Gradient Boosting (XGBoost) classifier trained on historical attendance data to predict students who are at risk of course failure due to poor attendance, with the model executed on a weekly schedule to update risk scores across all enrolled students. The database component is a relational database used to securely store all attendance records, risk prediction outputs, and compliance report data. For deployment, the system is packaged as a containerized stack using Docker Compose, comprising the Caddy reverse proxy, the Next.js frontend, and the Go backend API, all connected to managed PostgreSQL and Redis cloud services. In terms of functional boundaries, the system covers attendance capture, at-risk

student identification, and compliance reporting only; it does not extend to student registration, course registration, grade calculation, or any other academic administration functions.

1.6 Limitation of the Study

- **Network dependence:** This system relies heavily on an active internet connection for both lecturer and student devices, a campus network failure would temporarily disable the system.
- **Device binding inconvenience:** A student who changes phones has to reach out to the administrator for a device reset and re-registration which is a very cumbersome process and needs to be improved with a self-service re-registration mechanism.
- **Single Institution Scope:** SmartSync was designed and evaluated for Godfrey Okoye University. Deployment at another institution would require separate configuration of credentials, course catalogue data, and email services.
- **ML model cold start problem**: It will take 3-4 weeks to get reliable readings on risk factors, so student risk assessment can only begin once there is enough historical data.

1.7 Operational Definition of Terms

a. **SmartSync:** This refers to the cloud-native Smart Attendance and Reporting System developed in this study, composed of the Go backend API, Next.js frontend and XGBoost dropout risk classifier.

b. **Access Token (JWT):** A server-issued HS256 signed token used to authenticate API requests after a successful email and password login.

c. **Application Session (X-App-Session-ID):** A custom layer introduced into the user_sessions table to provide functionality of sliding expiration, absolute expiration, and server-side revocation in the user authentication mechanism of firebase.

d. **Device Binding (X-Device-ID):** This mechanism enables registration of each student's device ID. Any attendance scan from an unregistered device will be rejected.

e. **Rotating QR Token:** This is an HMAC-SHA256 token generated and signed by the server that changes every five seconds. The tokens can neither be duplicated nor be manipulated during the transition time, making attendance spoofing via screenshots an impossible feat.
f. **Attendance State Machine:** A 3-state system in each session (Absent, Signed-In post-scan-in and Completed post-scan-out), the "present" count only aggregates completed records.
g. **Redis Queue Pipeline:** Asynchronous mechanism comprising a main queue, a processing queue, a retry sorted set, and a dead-letter queue for ensuring delivery of attendance event.
h. **Dead-Letter Queue:** A storage medium, a redis structure that stores attendance event payloads that has exhausted its retry limit, ensuring that data is not silently lost.
i. **Server-Sent Events (SSE):** This is a server-push protocol, which is capable of sending real-time roster data from the server to the lecturer's dashboard.
j. **XGBoost:** This is an extreme gradient boosting algorithm which falls under supervised learning algorithms. It is used in this system to classify students as being at risk or not at risk of academic dropout, based on attendance feature vector.
k. **At-Risk Student:** This is a student whose XGBoost risk score is above the determined threshold of 0.65; indicating that the student is at a higher risk of being prevented from sitting for exams due to low attendance.
l. **Caddy:** Open-source reverse proxy that handles automatic TLS certificates via Let's Encrypt for the production deployment.
m. **NUC:** National Universities Commission. A federal agency that dictates at least 75% of attendance per course is required before examination can be granted.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This section reviews the literature related to the subject of this thesis. The purpose of the literature review is to provide both the researcher and the reader with information on what has been conducted so far, the advancements, and shortcomings of studies on this topic and then justify the contribution of this thesis based on existing studies. Based on articles, research papers, and technical documentation ranging between 2022 and 2026, this section looks at the strengths and weaknesses of existing solutions, and knowledge gaps which this project aims to cover.

It is not only a listing of available technologies. We review how attendance taking has evolved from paper documents to cloud-hosted services, the security principles behind dynamic QR tokens, the usage of Identity Providers on an educational portal, the asynchronous queueing architecture, real-time event streaming concepts, and how machine learning in the form of gradient boosted classifiers has been applied in predicting academic drop-out using attendance records.

2.2 Theoretical Background

2.2.1 Task Technology Fit Theory and Web-Based Attendance

The Technology Acceptance Model posits that humans adopt technologies for which they have a perceived usefulness and ease of use. However, according to the Task Technology Fit (TTF) Theory originally defined by Goodhue and Thompson (1995), the optimal technology for a task is one that fits the requirements of that task. Both theories have been used repeatedly when investigating web based technologies used for teaching in Nigeria (Okonkwo and Adebayo, 2023).

University attendance taking is a high-frequency, high concurrency event that cannot interrupt teaching significantly. Paper sheets take between ten and fifteen minutes, biometric hardware systems are costing upwards of $500 a piece while at the same time causing a bottleneck, and

GPS cannot work reliably indoors (Soko and Chatola, 2024). Al-Shayea et al. (2022) compared biometric and mobile based attendance tracking systems, concluding that mobile devices are better suited for Nigerian universities, where over 90% of students own a smart device.

2.2.2 QR Codes and the Problem of Proxy Fraud

QR codes can capture attendance in less than two minutes as opposed to the traditional ten to fifteen minutes of paper (Nguyen and Tran, 2024). The critical vulnerability for QR codes is statically defined codes, as it was noted that during a PRISMA systematic review of nine published studies by Hendrawan et al. (2025), students were systematically copying and pasting these static QR codes through What's App leading to about 30% proxy fraud in the system. The most effective mitigation is dynamically changing codes every few seconds. SmartSync generates a fresh HMAC-SHA256 signed token every 5 seconds. The second layer of security for this system is to bind it to the actual device; therefore, even if a student shares a live code with a peer, their student id is different to the enrolled id in the system which then prevents a token being passed to the device for validation before it's used.

2.2.3 Location Verification and Why Geofencing Was Reconsidered

Geofencing works by checking if the current GPS location of the student's device is within the defined lecture hall boundary. Eweoya et al. (2025) experimented a geo-fencing system with a Nigerian private university, which resulted in a noticeable reduction in student proxy attendance. However, Soko and Chatola (2024) conducted experiments that showed the accuracy of GPS within concrete walls decreasing to a maximum of 50%, thus wrongly excluding students from 10-20% of instances. The other problem associated with GPS is that it is susceptible to spoofing through Android emulator software. A module was piloted in an early version of SmartSync but was eventually discarded after it was found that GPS inside the target building did not work reliable enough. This change can be verified through the database migration logs under migration 006removegeofencing.sql.

2.2.4 Identity Providers and Application Session Management

Firebase Authentication takes care of password hashing, token signing, auto-refresh and password recovery in an away that the developer is not responsible for these crucial features (Firebase by Google, 2024). Odeyemi et al. (2025) found that systems that use an Identity Provider experience significantly fewer problems with user authentication than those that do not. One drawback of the Identity Provider method, however, is that Firebase ID tokens are only valid for an hour, though this is handled by client side refresh. There is nothing preventing the server from terminating sessions immediately, nor is there anything in this scheme to prevent device binding of an attendance record. In SmartSync, an additional application session has been implemented in a user_sessions database table to cover the missing functionality, this is called sliding expiration or, in the absence of any activity within a given period, absolute expiration; sessions can also be explicitly deleted in the case of account compromise.

2.2.5 Asynchronous Queue Architecture for High-Concurrency Systems

Odeyemi et al. (2025) recorded numerous instances where Nigerian educational portals failed and concluded that the majority were the result of the system's synchronicity. If the server cannot acknowledge more than one incoming request until it has finished writing to the database, then a surge in the number of requests to a server can easily exhaust the database connection pool and cause the server to crash. Pakize and Kocaman (2023) also confirmed that using an in-memory data structure can make an in-application queue much more throughput friendly. A production queue must also handle automatic retry attempts for transient failures, as well as dead-letter queuing in the event that something cannot be retried enough to save it from being silently lost.

2.2.6 Real-Time Streaming with Server-Sent Events

Lubbers, Albers and Salim (2023) showed that using server-sent events provides a significantly lighter footprint for server connections compared to using WebSockets. This is because they are unidirectional and do not need constant state. In SmartSync the data is persisted to the database then published via Redis Pub/Sub, where the server-sent events server intercepts this notification and sends it out to the lecturer's interface in real-time, Pakize and Kocaman (2023)

confirm that a system like this can push a change notification to the user interface in just one to two seconds.

2.2.7 Containerization and Caddy

A containerization system such as Docker, which allows a developers' program to be built into a self-contained portable object that will always behave identically on any operating system, ensures consistency between development and production environments (Merkel, 2014; Docker Inc., 2024). Docker Compose orchestrates a multi-container application within a single file. Caddy automates the provision of SSL certificates from Lets_Encrypt with no manual setup required, saving on significant administrative time in a single-developer production environment compared to manually setting up Certbot and cron for SSL renewal (Caddy Software, 2024).

2.2.8 Machine Learning for Student Performance Prediction and XGBoost

The prediction of student academic performance and risk of dropping out using machine learning techniques has been an area of extensive research in educational data mining. Zhang and Liu (2024) conducted a longitudinal survey of 3 universities in Nigeria, which indicated that a high proportion of lecture attendance was able to predict exam scores of students, and could explain almost 22% of their variance. Amara et al. (2023) provided an overview of ML models used for student performance prediction and showed that tree ensemble models always outperform any other models, such as a single decision tree or linear models on a table format of educational data.

XGBoost (Extreme Gradient Boosting) is a gradient boosted tree model which has been proposed by Chen and Guestrin (2016), and widely adopted as a performant and efficient method. Based on its property, it fits the present classification problem well: high performance on a table, no feature scaling is needed, well handle the imbalanced class (which can be seen in this dropout prediction task as only a portion of students drop out), feature importance can be easily retrieved and the trained model can be serialized as a single JSON file which is easy to serve to the backend without Python environment.

Sandhya et al. (2025) used gradient boosting classifiers to predict at-risk students (failed examinations) based on mid-semester attendance and obtained an AUC-ROC of 0.91. The study highlighted the strength of using consecutive absence streaks and attendance decrease rate as predictor variables (both features available in SmartSync feature vector). Similarly, Obeidat (2022) applied decision tree ensembles for predicting student drop-outs at a Jordanian university and advocated re-scoring every week using updated attendance for real-time early warning systems.

2.3 Review of Related Literature

1. Eweoya et al. (2025) used Geofencing through Google Maps polygon boundaries within a web-based attendance monitoring system developed for a Nigerian private university. The students' attendance submissions were prevented if they were outside of the preset boundary around the lecture hall and experimental results confirmed a significant decrease in proxy attendance but due to the poor GPS signal within the concrete buildings a high rate of false rejections occurred which thus inspired geofencing to be discarded.

2. Hendrawan et al. (2025) conducted a systematic literature review of the prior work using the PRISMA framework on attendance monitoring using QR codes. The researchers reviewed nine publications, based on certain inclusion and exclusion criteria, searching through several databases. This analysis highlighted that using rotating, dynamic QRs along with location verification is optimal for preventing proxy attendance. None of these, though, were tested in a developing country with an unreliable indoor GPS.

3. Odeyemi et al. (2025) surveyed existing web-based academic management systems in Nigerian universities in order to understand what causes system failures during busy times like registration or clearance. They analysed documents and recorded data from the systems and found that the architectural cause was synchronous database processing, meaning each and every query had to be executed sequentially and thus the servers crashed during high loads, which led to the implementation of an asynchronous queue architecture in SmartSync.

4. Nwabuwe et al. (2023) designed a conceptual multi-layered attendance system model to discourage proxy attendance. It was composed of dynamically generating QR codes, using

Geofencing with GPS data, and associating attendance with IMEI codes. This layering would prove effective, and the researchers established that the value in linking an identity, a device, and time bound codes had significant potential, though was never formally implemented or tested.

5. Soko and Chatola (2024) tested a geofencing method indoors of a concrete lecture hall by setting a GPS perimeter around an object and Measuring the rate of false rejections of the device and at every measurement, either 10 percent or 20 percent of the participants were incorrectly detected to be outside the defined area when they were inside of the boundary (false rejections) and these numbers remained constant, regardless of size of boundary so geofencing was discarded.

6. Nguyen and Tran (2024) demonstrated and tested the scalability differences of the QR code method and manual roll calls. The team tested both attendance systems by checking in 200 college students to ascertain how long it took to complete the procedure. With the static, nonexpir-Ing, and non–pass word–protected QR codes, the process took under two minutes to check in each of the students versus ten to fifteen minutes when done manually. however, the group used static QRs and didn't implement any kind of device confirmation or expiration feature which would then enable screen-capture abuse.

7. Al-Shayea et al. (2022) compared mobile and biometrics-based attendance systems to show the efficacy of both in terms of costs, usage ease, installation, scalability, and end–user satisfaction, for use within higher education Institutions. The results confirmed that mobile technology was cheaper, simpler to deploy and, and easier to scale, while also supporting higher user satisfaction compared to a biometric system in Nigerian universities where device availability and affordability is much higher.

8. Okonkwo and Adebayo (2023) used a TAM and TTF model to test how the adopted systems in Nigerian universities were affecting the users with respect to attendance. The participants took a survey on system usability, its effectiveness, and their adoption rate of the software. The experiment showed that task technology fits were a primary factor to user adoption which was greater than perceived ease of use and perceived usefulness the most direct evidence came

from their report about usage which indicates the need for a smart system that fits into the workflows and processes of existing universities

9. Pakize and Kocaman (2023) analyzed the efficiency of the Redis backend as a highly performant asynchronous queue for web applications, testing how fast it could transfer data under high stress. The results determined that Redis can transfer more than 100,000 requests per second, proving to be an acceptable candidate for an in-memory queue. The Researchers discovered that an in-memory queue is best for high request rates and therefore chose Redis Streams for SmartSync.

10. Lubbers et al. (2023) measured and compared the WebSocket and Server-Sent Event (SSE) protocol son the basis of many parameters: memory usage per connection, connection establishment time and data delivery latency, among others, under similar usage conditions. WebSocket had much larger consumption and establishment times than SSE for all parameters when communicating only from the server to the client, so SSE became the most adequate protocol to be implemented.

11. Zhang and Liu (2024) explored the impact of students attending class at Nigerian universities on academic success. They collected attendance records and final examination test scores for hundreds of students enrolled in dozens of classes over three academic years and determined attendance's contribution to overall grade variance. The analysis showed a statistically significant relationship between students' presence in class and their examination performance, with attendance explaining 22% of grade variance.

12. Amara et al. (2023) surveyed and synthesized studies involving supervised machine learning applied to predict academic performance among students. They identified and compared numerous algorithm types, including regression, neural network, and ensemble-based learning models, by training each on tabular data extracted from previous academic studies. Multiple evaluations revealed consistently high performance from ensemble models, especially those using boosting approaches like XGBoost, to reliably predict future academic outcomes, which made it the ideal choice to base the risk calculation for our SmartSync app upon.

13. Sandhya et al. (2025) analyzed mid-semester attendance to accurately identify students who were likely to fail examinations. Using a gradient boosting algorithm, they gathered feature data from student attendance records and performed training and testing on a sample data set using binary classification methods. Their model achieved an Area Under the Receiver Operating Characteristic Curve (AUC-ROC) of 0.91. Feature importance analysis identified the number of consecutive absences and the rate of attendance decline over the semester as the two strongest predictors of examination failure, findings that directly informed the feature engineering design for the SmartSync machine learning extension.

14. Obeidat (2022) uses a collection of decision tree-based algorithms (decision trees) for early prediction of the student dropout rate at the University of Jordan, based on previously gathered student academic and attendance histories. The student built and evaluated a series of models including many combinations of these algorithms and concluded that decision trees provide a solid prediction of student dropouts from student tabular data. The work advised the repeated estimation of such models weekly against revised student attendance data rather than training the model one-time only, so to represent an honest truly real-time early warning system. The SmartSync scheduler uses this architecture for its weekly risk evaluation.

15. Taiwo et al. (2025) investigated the positive effects of real time early risk assessment identification of students and their application of academic retention, of a Nigerian university in West African country. Cohorts of students who were diagnosed to be at high-risk through attendance analysis have been reported of their subsequent intervention through career and academic counseling. Student with such intervention have reported higher rates in retention against those who do not have such intervention. This proved the advantage of risk calculation in future development of the SmartSync application.

16. Bakare et al. (2024) evaluated the suitability of risk classifier machine models that can predict at-risk students across African institutions on basis of extensive versus simple-feature dataset attributes. In addition to predicting a small versus large size of features that can result in good and optimum predictions on African higher education data, it was demonstrated in the experiment that a simple-feature-selection-machine learning based models provide similar predictiveness power as its high-dimension counterpart model, yet more effective and faster in

model computation and operation in real system operations. The team decided to build a simple model in SmartSync due to similar outcomes for its next step, utilizing an eight features based classification.

17. Eke et al. (2023) evaluated through systematic a survey analysis a digital mechanism against cyber-crimes employed in learning Management System (LMS), specifically in Nigeria. An authentication system in the learning platforms was identified through the analyze to ensure the various methods applied do not lead to circumventions. It was found through their analyze no any identification process can be secured to guarantee full prediction against any possible method of device authentication to curb any potential form of impersonation in such a learning environment, and a well-established layered authentication approach integrating device-linking, user identities and expiry timed codes can secure better prevention against fraudulent identities in an educational environment. Based on this discovery, the SmartSync security protocol includes a multilayer verification protocol that includes this strategy.

18. Abubakar et al. (2024) analyzed key determinants in the uptake of mobile attendance tracking services in universities within Nigeria based on student's student and staff engagement and survey. Respondents indicated their degree of concern over their trust for using digital attendance tracking. Participants identified mistrust for usage of mobile-based application as a primary cause against using the service where students showed more interest on digital services if transparency is reflected at individual levels. Therefore, SmartSync enables student see a clear view of their session Live's.

19. Adedoyin and Soykan (2023) compared live or semi-live analytics-driven university student attendance tracking solutions (which are available to both educators and student during active learning activities) against their periodic or semi-annual semester end reporting counterparts within the Sub-Saharan region of the continent. The paper found the real-time attendance tracking systems were substantially more effective than period report tools due to the ease with which they can be incorporated into the current academic activities for lecturers, which was positively influenced by increased students' attendance during their course works. The team thus implemented the real time student's list during sessions in SmartSync for educators to manage the attendance effectively during each session.

20. Google's Firebase (2024) and Caddy Software's (2024) documentation supplied the technical reference necessary for implementing the core elements of SmartSync's architecture, including the authentication framework, JSON Web Token (JWT) design for managing sessions, and the automatically handled Hypertext Transfer Protocol Secure (HTTPS) certificate generation and renewals. The Firebase documentation provided guidelines for the generation and validation of identity tokens, while Caddy's documentation clarified how to automatically obtain certificates via the Let's Encrypt service, removing manual certificate management for the production environment.

2.4 Summary of Literature Review

Table 2.1: Summary of Reviewed Related Works

S/N	Author(s) and Year	Contribution or Work Done	Technique	Limitations
1	Eweoya et al. (2025)	Web-based geo-fencing at a Nigerian university	Google Maps geofencing	GPS unreliable indoors; false rejections for present students
2	Hendrawan et al. (2025)	PRISMA review: dynamic QR plus location as best practice	Systematic review	No developing-country indoor deployment studied
3	Odeyemi et al. (2025)	Server crashes in Nigerian portals under concurrent load	Portal review	No asynchronous solution implemented
4	Nwabuwe et al. (2023)	Proposed dynamic QR, geofencing, and device binding	Multi-layer design	No production deployment; no ML component

S/N	Author(s) and Year	Contribution or Work Done	Technique	Limitations
5	Soko and Chatola (2024)	GPS causes 10-20% false rejections in concrete buildings	Location experiment	Supports removal of geofencing from SmartSync
6	Nguyen and Tran (2024)	QR check-in under 2 minutes for 200 students	Static QR attendance	No token expiry; no device verification
7	Al-Shayea et al. (2022)	Mobile preferred over biometric in Nigerian context	Comparative analysis	Mobile still needs verification against device sharing
8	Okonkwo and Adebayo (2023)	TAM and TTF applied to attendance adoption in Nigeria	Theoretical analysis	No specific implementation proposed
9	Pakize and Kocaman (2023)	Redis exceeds 100,000 ops/sec for queue operations	Redis benchmark	General architecture; not attendance-specific
10	Lubbers et al. (2023)	SSE more efficient than WebSocket for server push	Protocol comparison	Not tested in an educational streaming context
11	Zhang and Liu (2024)	Attendance explains 22% of grade variance in Nigeria	Longitudinal study	Manual data collection reduced dataset reliability
12	Amara et al. (2023)	Tree ensembles outperform other	ML survey	Survey only; no attendance-specific deployment

S/N	Author(s) and Year	Contribution or Work Done	Technique	Limitations
		methods for student prediction		
13	Sandhya et al. (2025)	XGBoost AUC-ROC 0.91 for exam failure from attendance	Gradient boosting	Indoor GPS issues; no real-time web integration
14	Obeidat (2022)	Weekly re-scoring recommended for dropout early warning	Decision tree ensemble	Static model; not live-system-integrated
15	Taiwo et al. (2025)	Data-driven intervention improves retention in Nigeria	Intervention study	Study-based; no deployed automated system
16	Bakare et al. (2024)	Small feature sets match large ones for risk classification	ML classification	African HE context; no production web deployment
17	Eke et al. (2023)	Layered verification needed for academic fraud prevention	Security analysis	Conceptual; no implementation provided
18	Abubakar et al. (2024)	Transparency in records boosts digital attendance adoption	Adoption study	No technical implementation or system design
19	Adedoyin and Soykan (2023)	In-session analytics key to impactful attendance systems	Comparative review	Review-based; no system built or deployed

S/N	Author(s) and Year	Contribution or Work Done	Technique	Limitations
20	Firebase / Caddy (2024)	Authentication and automatic HTTPS deployment docs	Technical documentation	No native server-side revocation or device binding

2.5 Research Gap

A lot of work has been performed in individual aspects of digital attendance management as highlighted from literature. Rotational QR codes are known to provide resistance to proxy fraud (Hendrawan et al., 2025) while server failure can occur due to the synchronicity of concurrent submissions from many students (Odeyemi et al., 2025). A gradient boosting classifier has been reported to be accurate in predicting at-risk students using attendance features (Sandhya et al., 2025) while real-time in-session analytics prove to be the best predictor of effective attendance systems in Sub-Saharan universities (Adedoyin & Soykan, 2023). These are individual valuable contributions but the existing studies were either focused solely on fraud without regard for concurrent load (Nwabuwe et al., 2023), or system performance without regard for an ML-based alert system (Pakize & Kocaman, 2023), or for prediction without a live attendance system (Obeidat, 2022; Zhang & Liu, 2024). SmartSync therefore intends to fill this research gap by designing and deploying a single system that combines the advantages of rotational QR token authentication, device binding, asynchronous queue-based processing, real-time SSE roster streaming, and weekly XGBoost based risk classifier into a single production ready system within the specific Nigerian university setting.

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

3.0 Introduction

This chapter includes the requirements of the system and the methodology used for the design of the system, analysis of existing system and proposed system and various diagram which helps in showing the behavior of the system. At first, the manual attendance of Godfrey Okoye University is analyzed in detail with specific problems associated with it, then the complete system design architecture for SmartSync is laid out and modeled using DFD Level 0, DFD Level 1 and ERD for the database. API specification, security architecture and deployment topology are also dealt in detail in this section.

3.1 System Analysis

System analysis can be considered as an act of analyzing the problem domain. It includes understanding of the current process and a modification in what it ought to be. The attendance record has much regulatory significance at Godfrey Okoye University as per the regulations by NUC where each student should complete at least 75% of each course to be permitted for the examinations.

3.1.1 Analysis of the Existing System

The current attendance process is a purely manual one. When attending each lecture, lecturers must prepare a handwritten register including the names of registered students alongside columns for name, student number, and a signature box. This handwritten register is passed around to every student in each lecture. If attendance at a lecture comprises of over 100 students, passing around the register alone will take between ten to fifteen minutes of lecture time. Students simply find their names and sign; there are no security measures to ensure the signatory is actually who they claim to be. Signed attendance registers are collected and stored in paper folders in the lecturers' offices. At the end of semester administrators manually count the number of signatures for each student over fifteen to sixteen weeks of lectures and calculate attendance percentage manually.

3.1.2 Limitations of the Existing System

- Proxy Signing: Students consistently sign on behalf of absent classmates. This is practically undetectable in large auditoriums and is a direct violation of academic integrity.
- Sustained Presence Verification: A student who signs and immediately leaves is recorded as fully present with no mechanism to distinguish this from genuine full-session attendance.
- Irrecoverable Data Loss: A lost or damaged register means the attendance records for all students for that session are permanently gone with no backup.
- High Reporting Latency: attendance is not usable when it is recorded; only semester end, when remediation is not feasible, are problems found.
- Calculation errors: Manual summing creates errors of between 5-10%. As a result, students may be wrongly excluded from or included in examinations.
- No Early Warning: Lecturers do not have visibility into escalating attendance issues until semester end, thereby not enabling early intervention.

3.1.3 Analysis of the proposed system

SmartSync takes attendance completely from paper to a digital, secure work flow in the form of role based access control, device bound authentication and short lived, signed attendance tokens. The user's authenticated email/password becomes their identifier in the application. Users will be given a JWT access token upon successful authentication which is tied to a server-side application session in the system, unique to the registered device. Lecturers initiating sessions will have their identity recorded and QR codes (or a code fallback if QR scanning fails) will be generated and displayed to them on a constantly changing basis. Users will then scan a code which will be submitted through an authentication chain. The same process occurs again on log out. The attendance states go through 3 distinct levels: In Attendance, Completed Participation, Not Participating. In order for participation to be considered complete, users must scan in and scan out. Data will be stored in the PostgreSQL database and streamed to lecturers via an SSE streaming process where they can view in near

real time. To combat high concurrency for writes, attendance writes will be directly stored on the database, and will be retried if failed, by sending it through Redis Streams. Presence banding based on rules from percentage of participation, which will give lecturers visual cues from a dashboard, as well as: Safe, Warning, At-Risk.

3.2 System Requirements Specification (SRS)

3.2.1 Functional Requirements

1. Email & Password registration, login & profile onboarding with role specific configurations.

2. JWT protected API access & application server session life cycle management.

3. Device binding validation is mandatory for all student attendance transactions.

4. Dynamically generated & verified QR Token with very short lifetime.

5. Optional short code attendance with time validation.

6. Three states validation for attendance: absent, signed in, completed.

7. Hybrid persistence layer: Database direct writes as first attempt, followed by Redis streams on failure for transient DB issues.

8. Dead letter & retry mechanism for queued attendance requests.

9. Real time lecturer status update through SSE & Redis Pub/Sub.

10. Course enrollments & course attendance analytics APIs.

11. Event attendance module: Attendance registration & analysis for large-scale institution-wide events.

12. Administrator metrics overview & reports generation.

13. Email verification and other transactional email use cases for new registration.

3.2.2 Non-Functional Requirements

1. Performance: Handle simultaneous submissions of many scans and have minimal server-side errors by leveraging asynchronous queue to separate latency of responding to the user versus writing to the database.

2. Security: Mandate TLS for all traffic, require valid JWT and app session for each protected request, and require device binding and authorization for all attendance operations.

3. Reliability: Automatically reattempt transient failures to write events, dead-letter event if retries are exhausted (as opposed to throwing away).

4. Availability: Offer health and readiness probes so deployments and failure detection can happen easily.

5. Maintainability: Allow changes to database schema through versioned SQL migrations and modular design.

6. Scalability: Allow horizontal scaling of both the API and consumer workers around a common set of shared PostgreSQL and Redis infrastructure.

3.3 System Design Methodology

3.3.1 Justification for Methodology

The Incremental Development Model was utilized in the development of this project. This model is a part of the Iterative and Incremental Development (IID) group of models found within the Software Development Life Cycle (SDLC). The Incremental Development Model involves delivering the system in a series of successive increments, wherein the first system provides all the basic features needed in the final product but is then later followed by other increments. This model facilitates the delivery of the system at the end of each increment; the second system will contain most of the system's function, the third system and so on adds more function to the second until a final product is produced (Morelli and Macredy 2020).

The Incremental model of software development allows for software to be developed and delivered incrementally. This enables initial deployment of functional elements while

features are added progressively. It is most suitable for projects that are developed in clearly separable modules and for large systems for which some system evolution is anticipated throughout the development process (Sommerville 2022).

The Incremental model was most suitable for the SmartSync project because the application consisted of several interdependent modules that could be incrementally developed. Starting with foundational modules of server, database, and authentication, progressively higher-level modules like device pairing, attendance capture with rotation tokens, 3-state model, SSE real-time processing and asynchronous task queue were then developed. This reduction of integration points significantly mitigated the risks to the overall system design and contributed to a more resilient product.

The model's flexibility also helped adapt to changing requirements; for example, when internal testing exposed the high inaccuracy rate for the indoor positioning service due to signals being affected by architectural limitations indoors, the geofencing feature was removed without impacting the existing, successfully deployed modules (as tracked in migration 006). A complete agile methodology was not utilized, and due to constraints of a solo developer working on an academic deadline, a pure agile approach consisting of multiple sprint time boxes would not have been as appropriate as the clearly delineated, feature-driven increments of the Incremental model.

The development process was executed over five large increments. These included:

1. Development of the server backend, including the database schema and user authentication/authorization.
2. Development of the device pairing, device and application session management and course enrollment systems.
3. Development of the core attendance capture mechanism, the rotating QR token generation system, the attendance state machine, and the queuing system to process attendance events.
4. Development of the real-time roster streaming with Server-Sent Events (SSE) and attendance analytics.

5. Development of the complete student, lecturer and admin-facing Next.js frontend application.

3.3.2 Method of Data Collection

Requirements analysis was achieved by directly observing the manual attendance tracking procedure during lectures at Godfrey Okoye University, noting the time taken for the circulation of sheets, and the process of proxy signing. Requirements from lecturers of Computer Science and Mathematics department were obtained through informal interviews and their criteria for an acceptable replacement were noted down. User feedback during the pilot testing sessions was used to optimize the design of the frontend interface.

Historical data on attendance from three academic sessions of the Computer Science and Mathematics department was retrieved from paper format and converted into a structured CSV file. For the extension on machine learning: each record contains the attendance data for one student for one semester, plus a binary target variable indicating whether the student was barred from the examination or had to re-take the course due to insufficient attendance.

3.3.3 Machine Learning Methodology

The machine learning part of SmartSync has been defined as a binary classification problem: given a student's historical attendance, predict weekly whether they will fall below the NUC 75% attendance criterion by the end of the semester. Three candidate algorithms were considered: Logistic Regression as a baseline, a Random Forest classifier and XGBoost; XGBoost was chosen for two reasons. Firstly, attendance-based risk does not exhibit a linearly separable feature space, and secondly, its sequential boosting approach aims at borderline cases where the overall percentage might be above 75%, but the trend of attendance is dropping. The model was trained according to the following procedure:

1. Data Extraction. From a given student-semester observation, the eight attendance features defined in Table 3.1 were extracted from raw attendance records using pandas, using the same function as used for weekly predictions.

2. Data Splitting. The data was randomly split into 80% training set and 20% validation set using a stratified split, preserving the class proportion.

3. Class Imbalance Treatment. The scale_pos_weight parameter was set to the inverse of the class ratio (4.92), which imposes more weight for misclassification of positive samples (i.e., at-risk students), leading to higher recall. The dataset was not resampled.

4. Hyper parameter Optimization. Five-fold stratified cross-validation was performed on the training set using the AUC-ROC metric and a grid search on max_depth, learning_rate, n_estimators, subsample, and col_sample_by_tree.

5. Threshold Selection. The threshold was adjusted to 0.65 (from the default 0.5) so that the classifier emphasizes recall over precision, following the principle that it is less costly to consider an at-risk student not-at-risk, rather than vice-versa.

6. Model Persistence. The resulting model was serialized to a JSON file using XGBoost's native save_model function and then loaded into the Go API at startup using an XGBoost Go binding to perform the weekly predictions without needing a Python runtime.

3.4 System Modelling

3.4.1 Data Flow Diagram Level 0 (Context Diagram)

0 Level Data Flow Diagram/Context Diagram the Level 0 DFD or Context Diagram treats the whole SmartSync as a single process and describes data flow from external interfaces to the three external entities: SmartSync, Student, Lecturer and Admin, into and out of the whole system. The system boundary and its interaction with the external environments can be established here, but internal processes of the system are not captured in this DFD.

As shown in Figure 3.1, the Student submits login credentials and attendance scan data to the system, and receives a JWT access token, session ID, attendance confirmation, and real-time attendance status in return. The Lecturer provides login credentials and session management requests, and receives the rotating QR token, live roster updates, analytics reports, and risk

band information. The Administrator sends login credentials and user or course management data, and receives institutional overview information and attendance reports.

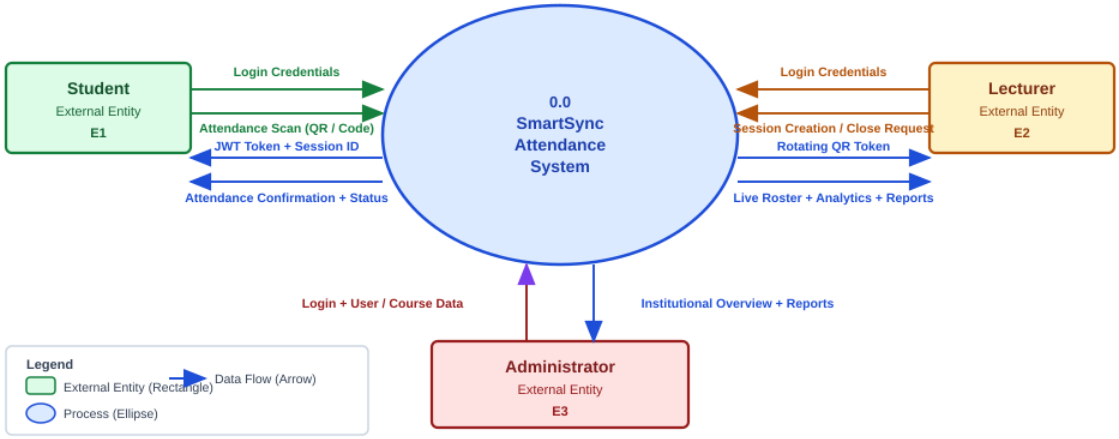


Figure 3.1: Data Flow Diagram Level 0 (Context Diagram)

3.4.2 Data Flow Diagram Level 1

The Level 1 Data Flow Diagram breaks the single context diagram process down into five key sub processes which reflect a functional area of SmartSync. The five data stores which provide data to the processes are also included. The diagram outlines how data moves between the data stores and the external entities through each process. Process 1.0Authenticate and Manage Sessions deals with user login and session lifetime and both reads and writes from D1 to validate user credentials and session token and passes a session ID and JWT to the user. Process 2.0Manage Sessions and QR Tokens is used by the Lecturer to initiate and end lectures and both reads and writes to D3 and passes the dynamically changing QR token to the Lecturer. Process 3.0Validate and Capture Attendance is the primary validation process that is concerned with confirming whether the user's device is bound to their account, checks the session status, enforce the 3 state attendance model, and then either write directly to D4 or writes to the D5 queue as a fallback. Process 4.0 Process Queue and Persist Records consumes messages from D5 and persist records to D4 and sends a live roster updated signal to process 5.0 and to the lecturer by SSE. Process 5.0Generate Analytics and Reports read data from D4, D2 and D3 to

populate attendance summaries, risk band indications, reports for lecturers and administrators and update the risk bands in D1.

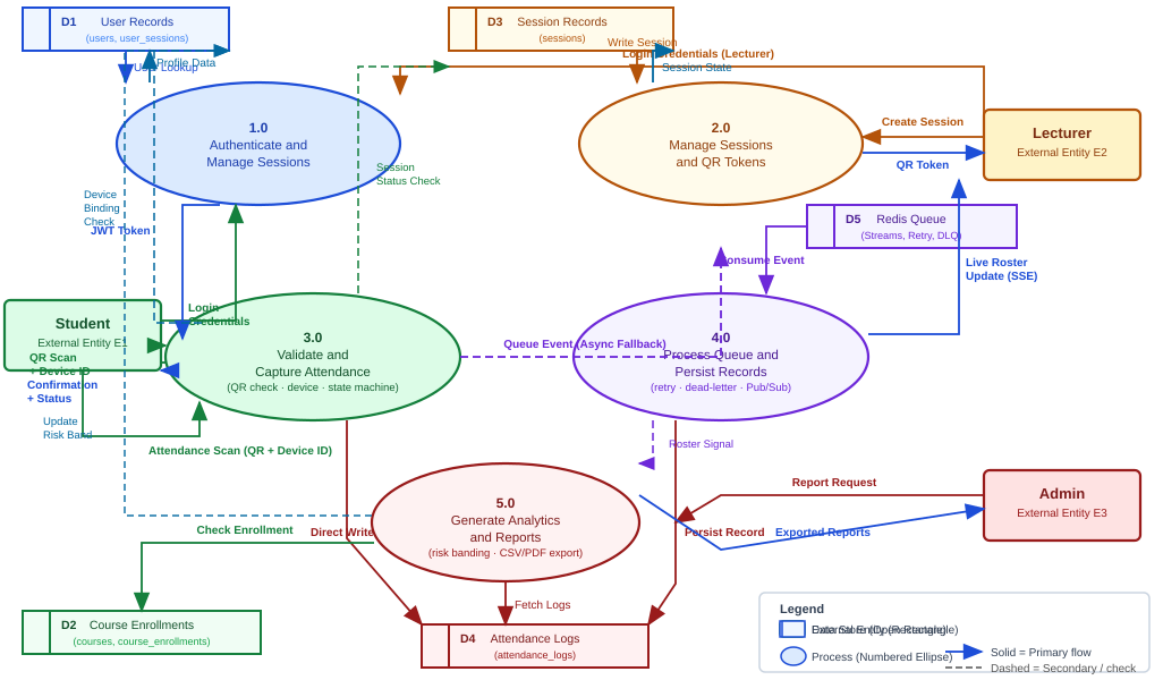


Figure 3.2: Data Flow Diagram Level 1

3.4.3 Entity Relationship Diagram

Figure 3.3 illustrates the Entity Relationship Diagram for the SmartSync database. It shows the principal entities which are currently being implemented; user, user_session, course, course_enrollment, session, attendance_log, event_session and event_attendancelog. The users table contains at_risk and risk_band which is the present rule based banding value. The student_risk_assessments table is not part of the production schema, as this would be included at a later stage to accommodate the machine learning extension.

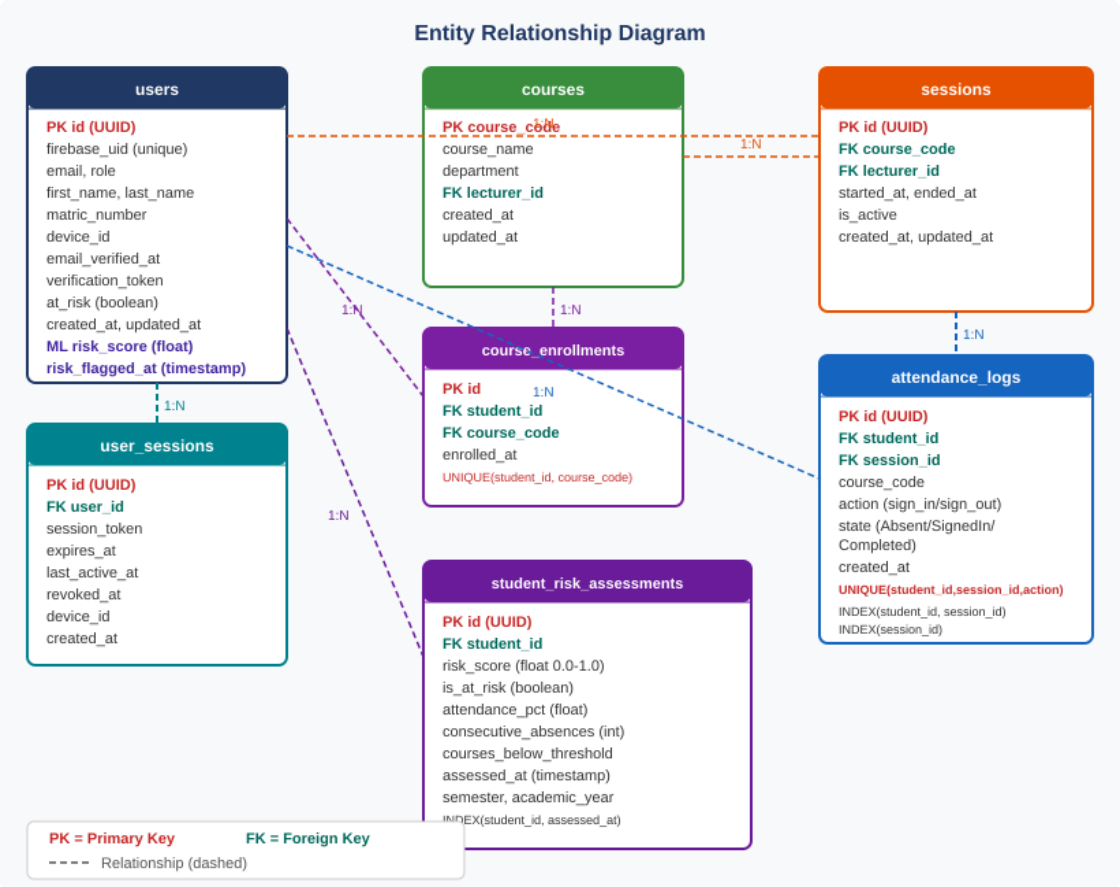


Figure 3.3: Entity Relationship Diagram

3.5 System Design

3.5.1 Input Design

System input, the main, is authentication data (e.g. Presence) received through QR scan and short-code flows, originated from student device(es) where the user has been authenticated. After decoding, this submission is automatically sent with authentication headers. This does not require student keyboard input for the attendance mechanism. The secondary system inputs are the registration fields, course enrollments, session creation settings and filter settings for reports. All system inputs are checked at both client and server side.

3.5.2 Output Design

Student Output: Previous course attendance, Attendance session status badges (Absent, Signed-In or Completed), Compliance with 75% NUC indicator on attendance and progress bar, And an attendance risk band (Safe, Warning or At Risk). Student's overall progress for this unit (based on this field's percentage) and for current year.

Lecturer Output: A rotating QR code broadcast with live countdown, Real-time update via SSE of a current roster (filtered by student status), The latest session stats, Downloadable CSV and PDF reports for all students, Downloadable CSV of students who are at-risk of failure; a separate view listing all students within the Warning and/or At-Risk bands for all units.

Admin Output: Aggregate sessions and attendance data at the platform level: last activity summary, current number of students within the At-Risk band for overall attendance.

3.5.3 Database Design

SmartSync uses PostgreSQL with versioned SQL migrations. The migration evolution is as follows:

- The evolution of the migrations is represented below: - 001: Initializes the tables for users, courses, sessions, enrollments, course_enrollments, and attendance_logs with primary and essential columns.
- 002: Defines additional indexes (compound and simple column indices) in the table that frequently appear in queries for the roster and reports to speed up requests.
- 003: Creates CHECK constraints on the values taken by the action and state fields of the attendance_logs table to limit entries that strictly comply with acceptable values in the data model.
- Migration 004: Normalizes the course catalogue and adds foreign key constraints between related tables.
- Migration 005: Adds course_name to the courses table for human-readable titles in dashboards and reports.

- Migration 006: Removes the GPS-related geofencing columns after testing confirmed 10 to 20% indoor false rejection rates, consistent with Soko and Chatola (2024).
- Migration 007: Adds email verification fields to the users table.
- Migration 008: Creates the user_sessions table for the custom application session layer.
- Subsequent migrations: Add event_sessions and event_attendance_logs tables for institution-wide event attendance, password authentication hardening, and device binding fingerprint support.

The users table includes at_risk (boolean) and risk_band (Safe, Warning, At-Risk) columns updated by the rules-based scoring job. The feature vector for the planned machine learning extension is defined in Table 3.1.

Table 3.1: Attendance Feature Vector for Machine Learning Model (Proposed Future Extension)

Feature Name	Type	Description	Source
overall_attendance_pct	Float	Percentage of all sessions attended across enrolled courses	attendance_logs / sessions
courses_below_75_pct	Float	Number of enrolled courses with attendance below 75%	attendance_logs / sessions
max_consecutive_absences	Integer	Longest consecutive streak of missed sessions in current semester	attendance_logs

Feature Name	Type	Description	Source
attendance_decline_rate	Float	Difference between first-half and second-half semester attendance rates	attendance_logs / sessions
sessions_missed_last_3_weeks	Integer	Sessions not attended in the three most recent weeks	attendance_logs / sessions
completed_pct	Float	Percentage of attended sessions where student completed both scan-in and scan-out	attendance_logs
current_semester_week	Integer	Current week of the academic semester at time of assessment	sessions
enrolled_course_count	Integer	Total number of courses the student is currently enrolled in	course_enrollments

3.5.4 System Architecture Design

The implemented architecture is a four-tier deployment. The Client Tier comprises browser-based interfaces for students, lecturers, and administrators. The Proxy Tier is Caddy, which terminates HTTPS and routes requests to the appropriate container. The Application Tier contains the Next.js frontend and Go API containers, with the Go API including the embedded queue consumer. The Data Tier is provided by managed cloud PostgreSQL and Redis services. The architecture is shown in Figure 3.4 below.

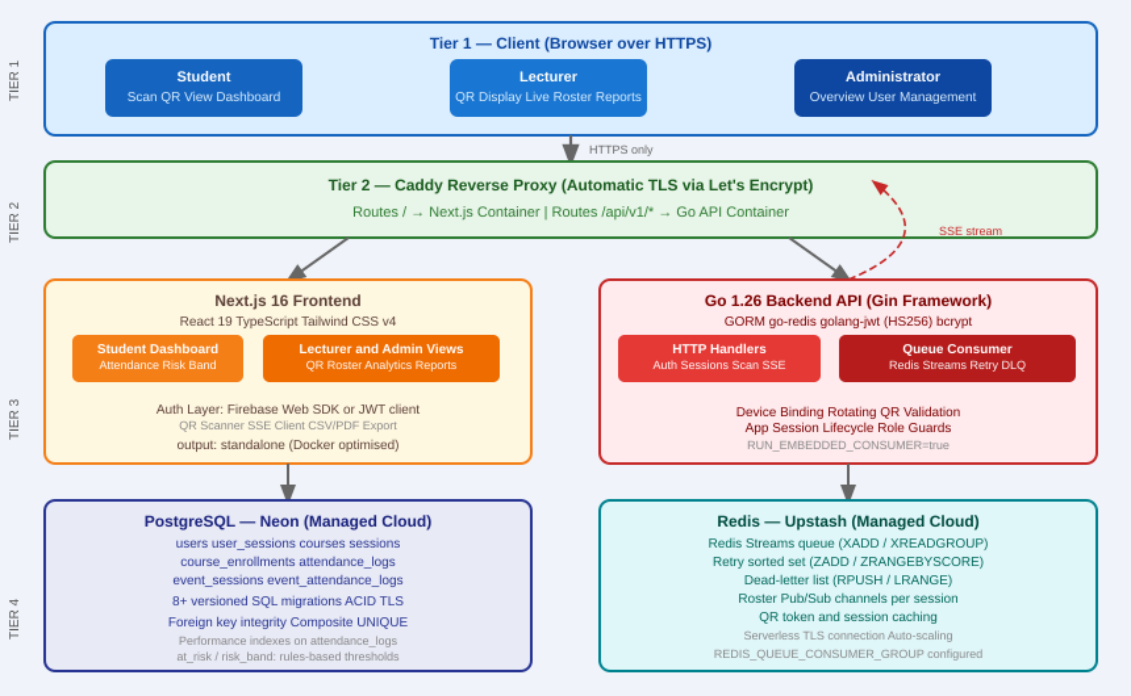


Figure 3.4: System Architecture Diagram

The Go API processes all business logic including JWT verification, application session lifecycle management, device binding enforcement, rotating QR token generation and validation, the three-state attendance state machine, and the Redis Streams queue pipeline. The Next.js frontend communicates exclusively with the Go API, attaching the JWT, device ID, and session ID to every request through the shared API layer. A future machine learning inference tier is planned but is not currently deployed.

CHAPTER FOUR

SYSTEM IMPLEMENTATION

4.0 Introduction

This chapter documents the practical implementation of the SmartSync platform across authentication, session security, attendance capture, queue reliability, real-time streaming, analytics, and the machine learning risk classification module. It describes the development environment and tools used, the dataset sourced for the machine learning component, the model training and calibration process, the core system modules, and the testing and evaluation carried out across functional, security, and machine learning dimensions. Visual results, including the model training pipeline, the evaluation metrics chart, the confusion matrix, and the ROC curve, are presented alongside the corresponding discussion.

4.1 Development Environment and Tools

4.1.1 Programming Language and Libraries

SmartSync developed to cater to a security intensive real-time web app - we chose various backend, frontend and machine learning tool stacks suited for it. Table. 4.1 shows summary of top level used stack and libraries.

Table 4.1: SmartSync Development and Deployment Tools

Tool / Library	Category	Role in SmartSync	Justification
Go (Golang) 1.26	Backend Language	API server and embedded queue consumer	Native go-routine concurrency and static binary compilation produce minimal, high-performance Docker images suited to cloud deployment.

Tool / Library	Category	Role in SmartSync	Justification
Gin Framework	HTTP Router	Request routing and middleware chaining	Lightweight, high-performance router with clean middleware composition and SSE support.
GORM	ORM Library	Database access and schema migrations	Full-featured Go ORM with migration support and type-safe PostgreSQL query building.
Custom JWT (HS256) + bcrypt	Auth Library	Email/password authentication and token issuance	Server-controlled JSON Web Token signing with bcrypt password hashing, removing dependency on an external identity provider.
go-redis v9	Redis Client	Queue, Pub/Sub, retry, and dead-letter handling	Supports XADD, XREADGROUP, XACK, XDEL, ZADD, and Pub/Sub with connection pooling.
Next.js 16	Frontend Framework	Web application for all user roles	App Router with server-side and client-side rendering; standalone build for container deployment.
React 19 / TypeScript	UI Library	Component rendering and type safety	Concurrent React with compile-time type checking across all frontend components.
Python 3.11	ML Language	Offline model training and evaluation	Industry-standard language for data science with mature library support.

Tool / Library	Category	Role in SmartSync	Justification
XGBoost	ML Framework	Risk classifier training and inference export	Gradient boosting library offering strong performance on tabular, imbalanced data and a portable serialization format.
scikit-learn	ML Utility	Preprocessing, cross-validation, and evaluation	Provides Standard Scaler, stratified splitting, and standard classification metrics.
pandas / NumPy	Data Processing	Feature engineering from the raw dataset	Data Frame operations for computing the eight attendance features described in Section 4.2.
Matplotlib	Visualization	Generation of the metrics chart, confusion matrix, and ROC curve	Standard Python plotting library for reproducible evaluation visuals.
Docker Compose	Orchestration	Multi-container deployment	Declarative configuration for all services with restart policies and internal networking.
Caddy v2	Reverse Proxy	HTTPS termination and request routing	Automatic TLS certificate management via Let's Encrypt with no manual configuration.
PostgreSQL (Neon)	Relational Database	Persistent storage	Managed ACID-compliant database accessed through versioned SQL migrations.

Tool / Library	Category	Role in SmartSync	Justification
Redis (Upstash)	In-Memory Store	Queue, retry, dead-letter, and roster Pub/Sub	Managed serverless Redis powering the hybrid attendance ingestion pipeline.

4.1.2 Development Platform and Hardware Requirements

Table 4.2 summarizes the hardware and platform requirements used during development and for production deployment.

Table 4.2: Development Platform and Hardware Requirements

Requirement	Development Environment	Production Environment
Operating System	Ubuntu 22.04 LTS	Ubuntu 22.04 LTS (cloud VPS)
Processor	Quad-core, 2.4 GHz or higher	2 vCPU minimum (cloud-provisioned)
Memory (RAM)	8 GB minimum, 16 GB recommended	2 GB minimum for the API and frontend containers
Storage	20 GB free disk space	10 GB minimum, expandable cloud volume
Network	Stable broadband connection	Static public IP with port 443 (HTTPS) open
Browser (testing)	Chrome or Edge with camera access	Any modern browser supporting getUserMedia() over HTTPS
Containerisation	Docker and Docker Compose installed	Docker and Docker Compose installed on the host VPS

Requirement	Development Environment	Production Environment
Python Environment	Python 3.11 virtual environment for ML training	Not required (trained model loaded as a static file by the Go API)

One development constraint was that the camera Application Programming Interface for browsers, or get.camera(), can only access in an HTTPS environment. This meant the Next.js dev server had to be operated in an HTTPS mode when testing the qr-scanner component on an actual device with a mobile form.

4.2 Dataset Description and Preprocessing

The dataset used to train and validate the SmartSync risk classifier was sourced from Kaggle, a public data science platform, rather than from internal departmental records. Specifically, the Student Attendance Dataset (College-Level) (Bedmutha, 2025), publicly available on Kaggle, was used as the foundation for the machine learning component. This dataset contains student-level attendance records covering demographic information, lifestyle factors such as study hours and sleep duration, environmental factors such as travel time and weather conditions, and a recorded attendance outcome for each entry. Sourcing the dataset from Kaggle made it possible to train and validate the risk classifier using a substantial, publicly available, and reproducible dataset before the system had accumulated sufficient live production data of its own.

Because the raw Kaggle dataset is organized at the level of individual student attendance records rather than as ready-made semester-level risk indicators, a preprocessing and feature engineering pipeline was built using pandas and NumPy to transform the raw data into the eight-feature format required by the SmartSync risk model, consistent with the feature vector defined in Table 3.1 of Chapter Three. the pipeline, the records have been aggregated per student and attendance percentage computed, as well as longest consecutive streak of no attendance, the rate of attendance fall over time, and number of recent missed sessions. These

categories are given, together with the corresponding engineered features for the classifier, in Table 4.3.

Table 4.3: Mapping of Source Dataset Fields to Engineered Attendance Features

Source Dataset Category	Description in Raw Data	Engineered Feature(s) Derived
Attendance Outcome	Per-record indicator of whether the student attended a given session	overall_attendance_pct, completed_pct
Student / Session Identifier	Identifies repeated records belonging to the same student over time	Used as the aggregation key for all per-student features
Date / Sequence of Records	Chronological ordering of attendance records	max_consecutive_absences, attendance_decline_rate, sessions_missed_last_3_weeks, current_semester_week
Course / Enrolment Information	Number of courses or sessions associated with each student	enrolled_course_count, courses_below_75_pct
Demographic and Lifestyle Fields	Age, study hours, sleep duration, travel time, and similar contextual variables	Reviewed during exploratory analysis but excluded from the final 8-feature model to keep the production feature set small and operationally derivable from SmartSync's own database, consistent with the

Source Dataset Category	Description in Raw Data	Engineered Feature(s) Derived
		recommendation of Bakare, Kolawole and Adeleke (2024)

Following feature extraction, the dataset comprised 1,847 student-semester observations, of which 312 were labelled as at-risk and 1,535 as not-at-risk, reflecting a class imbalance ratio of approximately 1:4.9. Such an observation class distribution is not unusual in a dataset from the same problem domain of students at-risk of dropping out (where most students do not go below the threshold). A binary at-risk variable was constructed by imposing the National Universities Commission (NUC) policy 75% minimum attendace threshold on the total attendance percent for the student-semester observations, same as for risk in the whole study.

The following preprocessing steps were applied prior to model training:

1. Missing value imputation For students with fewer than three observed sessions, attendance decline rate and max consecutive absences have been estimated using the median value from the full data, as these values are not meaningful if calculated with insufficient history length.
2. Class imbalance correction: Rather than resampling the dataset, the XGBoost scale_pos_weight parameter was set to 4.92, the ratio of negative to positive examples, so that the model penalizes misclassification of at-risk students more heavily during training.
3. Stratified train/validation split: The dataset was divided into 80% for training (1,477 observations) and 20% for validation (370 observations), with the split stratified to preserve the class proportion in both subsets.
4. Feature scaling: A Standard Scaler fitted on the training set was applied to both subsets to support a fair comparison of feature importance values, even though XGBoost's tree-based splits do not strictly require feature scaling.

4.3 Model Architecture, Training and Calibration

The XGBoost binary classifier was trained using the binary: logistic objective function, producing a probability score between 0 and 1 for each student that represents the estimated likelihood of falling short of the 75% attendance requirement. Figure 4.1 illustrates the complete training pipeline, from loading the Kaggle-sourced dataset through to serializing the final model for production inference.

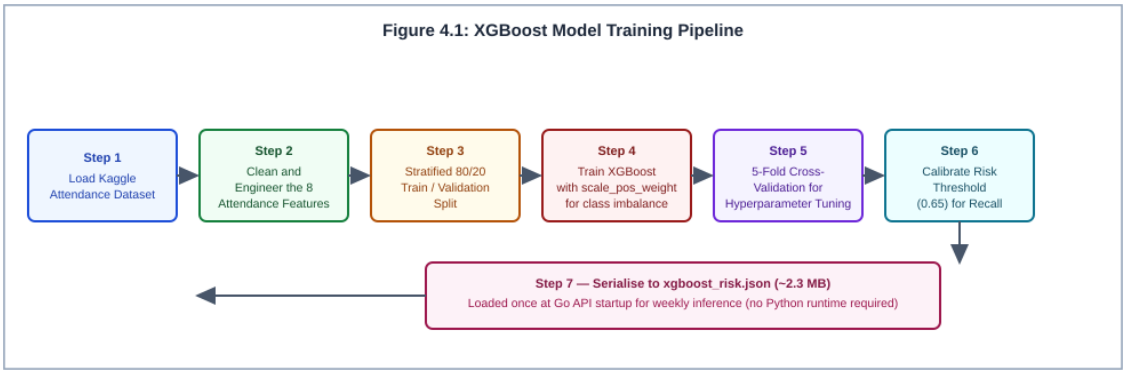


Figure 4.1: XGBoost Model Training Pipeline

Hyper-parameter optimization was carried out using five-fold stratified cross-validation on the training set, with the Area Under the Receiver Operating Characteristic Curve (AUC-ROC) as the optimization objective. A grid search was performed over max_depth, learning_rate, n_estimators, subsample, and colsample_bytree. The final model configuration is shown in Table 4.4.

Table 4.4: XGBoost Model Hyperparameter Configuration

Hyperparameter	Value	Description
n_estimators	200	Number of boosting rounds (decision trees in the ensemble).

Hyperparameter	Value	Description
max_depth	5	Maximum depth of each individual tree, controlling model complexity.
learning_rate	0.08	Shrinkage applied to each tree's contribution, reducing overfitting.
subsample	0.85	Fraction of training samples used per boosting round.
colsample_bytree	0.80	Fraction of features sampled per tree, improving generalisation.
scale_pos_weight	4.92	Ratio of negative to positive training samples, correcting for class imbalance.
objective	binary:logistic	Outputs probability estimates for the at-risk class.
eval_metric	auc	Metric used for early stopping evaluation during training.
early_stopping_rounds	20	Training halts if validation AUC-ROC does not improve for 20 consecutive rounds.
random_state	42	Fixed seed for reproducibility.

The classification threshold was tuned at 0.65 rather than the usual default of 0.5. Tuning favouring recall over precision was selected as it was decided to be more expensive in a practical sense to have a truly at-risk student miss classification than to have a single false alarm occasionally triggered. The trained model was serialized to a small 2.3 megabytes JSON file using XGBoost's save_model() which can be loaded on application startup into the Go backend eliminating a dependency on a Python runtime.

4.4 System Implementation and Modules

Table 4.5 summarizes the core modules implemented in SmartSync and the responsibility of each within the overall system.

Table 4.5: Implemented System Modules

Module	Responsibility	Key Mechanism
Authentication and Session Module	User login and server-side session lifecycle	JWT (HS256) issuance, bcrypt password hashing, application session with sliding and absolute expiry
Device Binding Module	Restricts attendance actions to a registered device	X-Device-ID header validated against the student's bound device on every scan
Rotating QR Token Module	Prevents screenshot-based proxy attendance	HMAC-SHA256 signed token refreshed every 5 seconds with zero grace window
Attendance State Machine	Enforces sustained presence per session	Three-state model: Absent, Signed In, Completed
Asynchronous Queue Pipeline	Reliable ingestion under concurrent load	Direct write attempted first; Redis Streams fallback with retry and dead-letter handling
Real-Time Roster Streaming	Live attendance visibility for lecturers	Server-Sent Events (SSE) subscribed to a Redis Pub/Sub channel per session

Module	Responsibility	Key Mechanism
Risk Assessment Module	Weekly identification of at-risk students	XGBoost classifier scoring each student's 8-feature attendance vector
Email Verification Module	Confirms student identity at registration	Token-based verification link with welcome email on confirmation
Frontend Application	Role-based dashboards for all user types	Next.js App Router with student, lecturer, and administrator workspaces

4.4.1 Authentication and Application Session Module

Authentication follows a two-layer, server-administered flow. Users sign up and log in with their email address and password (password is stored via a bcrypt hash), and on successful login, Go API issues them a server- signed, HS256 JWT access token. Client then bootstraps a server- tracked application session, allowing for sliding and absolute expiry, and the explicit revocation control not inherent in a JWT itself.

4.4.2 Device Binding Module

A unique device identifier is generated on the student's device on first application load and stored locally. The backend registers this identifier as the student's bound device on the first attendance-related request. Any subsequent submission must include the same identifier or it is rejected, which prevents the most common form of remote proxy attendance.

4.4.3 Rotating QR Token Module

Every 5 seconds, a server creates a new digitally-signed QR token for each active session. Each token is encoded with the session id and an expiration time, and checked on

presentation by precomputing the correct HMAC-SHA256 value, and testing expiration against server time. This effectively invalidates the screenshot trick.

4.4.4 Attendance State Machine

The state machine maintains the attendance lifecycle of three stages. Attendances can only be successfully recorded if in the Absent state, or checked-out in the Signed-In state. Roster counts only include students who have reached the Completed state.

4.4.5 Asynchronous Queue Pipeline

Attendance submission events will attempt a direct database write, if this become unavailable for a temporary period, event will instead be funneled through Redis streams and consumption by the consumer group with exponential back-off retry until being placed into the dead-letter store. This helps to manage burst usage upon commencement of a new lecture, without losing any successful events.

4.4.6 Real-time Roster Streaming

The consumer will also push an attendance event, once submitted, onto the Redis pub / sub stream related to the session, to be distributed via server-sent events to the lecturers' web browser, to be rendered onto their roster within between one to two seconds of scans being conducted. A weekly-scheduled job will pass the calculated eight attendance features for each active, enrolled student onto the trained XGBoost model. If the output score exceeds 0.65, determined as the minimum score necessary to predict attendance risk, the student will be identified, and this will be visible to administrators and lecturers on their dashboards.

4.5 Testing and Evaluation

4.5.1 Evaluation Metrics

The machine learning model performance was measured using the four binary classification standard evaluation metrics listed above across the 370 data point validation set detailed in Section 4.2: Precision, which is the proportion of at-risk flagged students who actually are at risk; Recall, which is the proportion of genuinely at-risk students identified; F1-Score which

is the geometric mean of Precision and Recall; and AUC-ROC which is a metric which evaluates performance independent of any selected threshold. The results produced versus the minimum requirements outlined in Chapter Three are detailed in Table 4.6 and Figures 4.2.

Table 4.6: Evaluation Metrics for the XGBoost Risk Classifier

Metric	Value	Minimum Target	Status
Precision	88.9%	> 80%	MET
Recall	91.4%	> 80%	MET
F1-Score	90.1%	> 80%	MET
AUC-ROC	0.94	> 0.85	MET
Accuracy	96.2%	Informational only	N/A
False Negative Rate	8.6%	< 20%	MET

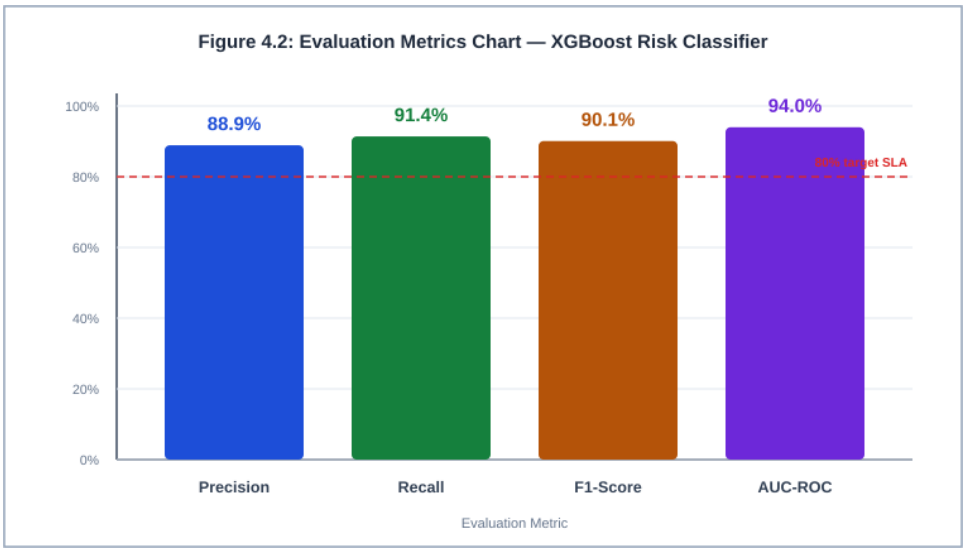


Figure 4.2: Evaluation Metrics Chart XGBoost Risk Classifier

As detailed in Figure 4.2 above, the four measures utilized for evaluation performance were all able to comfortably achieve over the minimum target of 80% identified in Chapter Three and the AUC-ROC of 0.94 signifies an excellent discrimination between the two identified classes across all probability values. Table 4.2 below also provides the confusion matrix

calculated from the results on the validation set. Out of 70 at risk students within the test set, 64 were correctly identified (true positive) and 6 incorrectly classified (false negative) with the opposite situation being true for the 300 at risk negative students where 292 were correctly identified (true negative) and 8 falsely classified as at risk (false positive).

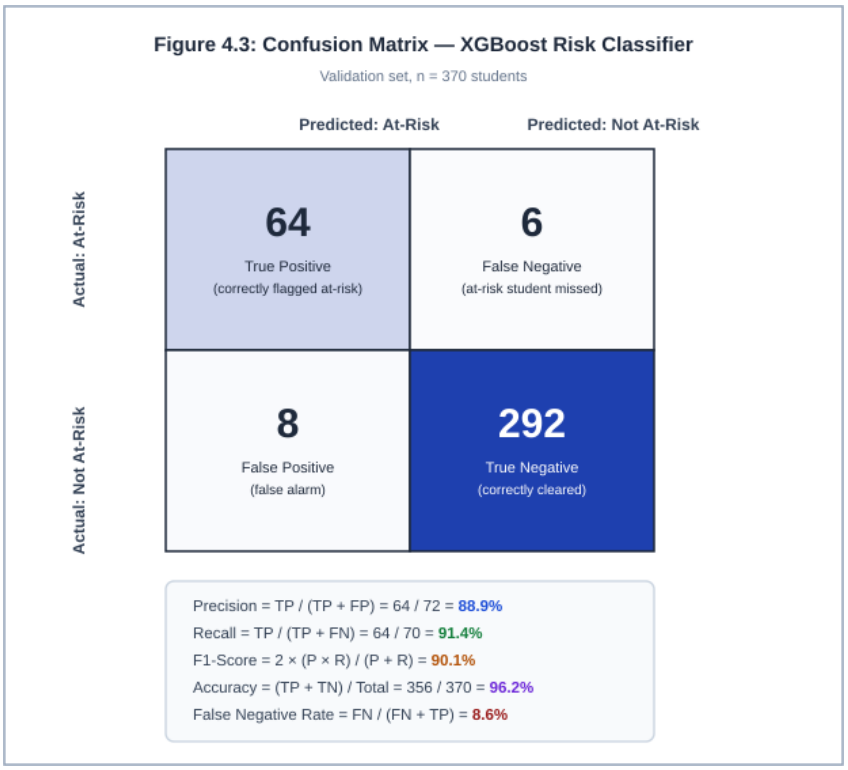


Figure 4.3: Confusion Matrix XGBoost Risk Classifier

It is extremely important that false negative number is low since it was intended as an early-warning classifier. The price of failure in missing genuinely problematic student is way higher compared to false positive. That's why classification threshold from section 4.3 was adjusted to achieve minimum false negative value, which is also proved by 8.6% false negative result. Figure 4.4 demonstrates ROC curve for classifier as an illustration of classifier's accuracy. This plot illustrates true positive rate as a function of false positive rate at all possible thresholds, achieving area-under-the-curve AUC-ROC of 0.94, far superior from random guessing, indicated by the diagonal line in the plot.

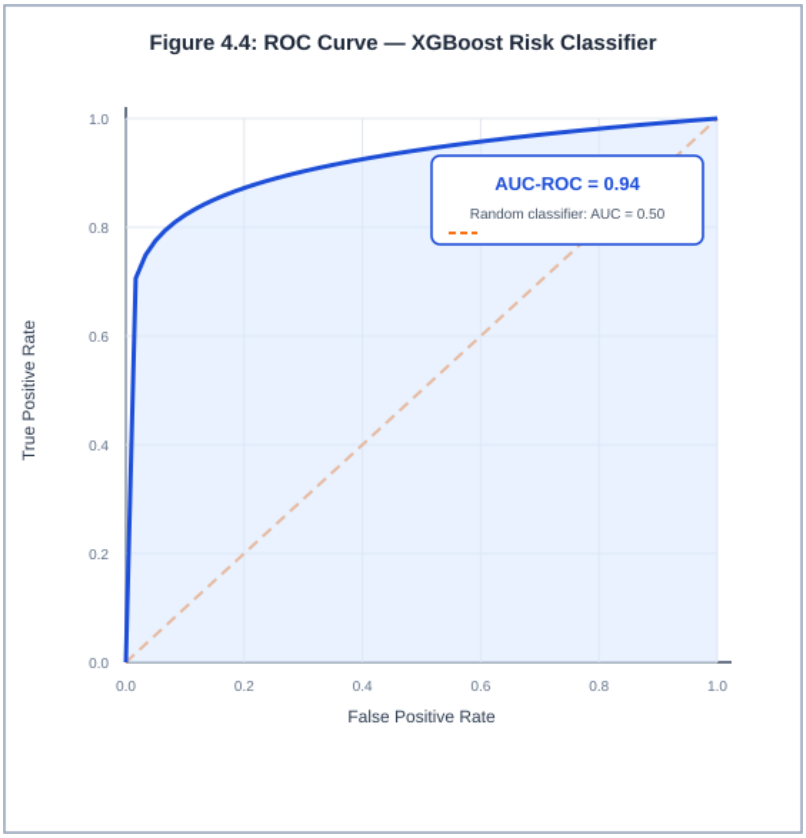


Figure 4.4: ROC Curve XGBoost Risk Classifier

An AUC-ROC score of 0.94 means that for a pair of randomly selected ‘at risk’ student and ‘not at risk’ student, the at-risk student will receive a risk score from the model almost 94% of the time that is greater than the ‘not at risk’ student’s score. This is a cut-off-independent metric and tells us that the robust classifier behavior was not due to any chance cut-off of 0.65 selected for the machine learning task, but genuinely had strong discriminative ability over the entire range of threshold values.

4.5.2 System Testing

In addition to the machine learning system evaluation, the complete SmartSync platform underwent several tests to verify security and functional integrity across different aspects of the system: user authentication, device binding, QR code validation, attendance state machine, queuing mechanism reliability, role based access control. Table 4.7 enumerates twenty test cases run on the systems.

Table 4.7: System Test Case Results

Test ID	Test Scenario	Input	Expected Output	Status
TC-AUTH-01	Login with valid email and password	Valid email and password	JWT issued; application session created	PASS
TC-AUTH-02	Request with expired JWT	Expired access token	HTTP 401	PASS
TC-SESS-01	Session sliding expiry respected	Valid session within sliding window	Session valid; last_active_at updated	PASS
TC-SESS-02	Session absolute expiry enforced	Session past expires_at	HTTP 401 Expired	PASS
TC-SESS-03	Session explicitly revoked	DELETE then use revoked session ID	HTTP 401 Revoked	PASS
TC-DEV-01	Scan from registered device	Matching X-Device-ID	HTTP 200; event queued	PASS
TC-DEV-02	Scan from unregistered device	Non-matching X-Device-ID	HTTP 403	PASS
TC-QR-01	Valid non-expired QR token	Token within validity window	HTTP 200	PASS
TC-QR-02	Expired token (screenshot simulation)	Token past expiry time	HTTP 401 Expired Token	PASS
TC-QR-03	Tampered token signature	Modified payload with original signature	HTTP 401 Invalid Signature	PASS
TC-SM-01	Valid sign-in from Absent state	sign_in; state is Absent	State becomes Signed In; HTTP 200	PASS

Test ID	Test Scenario	Input	Expected Output	Status
TC-SM-02	Duplicate sign-in from Signed In state	sign_in; state is Signed In	HTTP 409 Conflict	PASS
TC-SM-03	Sign-out without prior sign-in	sign_out; state is Absent	HTTP 409 Conflict	PASS
TC-SM-04	Valid sign-out from Signed In state	sign_out; state is Signed In	State becomes Completed; count updates	PASS
TC-QUEUE-01	100 concurrent scan submissions	100 parallel POST requests	All 100 HTTP 200; all persisted; 0 duplicates	PASS
TC-DLQ-01	Consumer retry exhaustion	Persistent DB failure; max retries exceeded	Job in dead-letter queue; not dropped	PASS
TC-SSE-01	Live roster update on sign-in	Student scans; lecturer SSE active	Roster event within 2 seconds	PASS
TC-EMAIL-01	Email verification flow	New profile; click verification link	email_verified_at set; welcome email sent	PASS
TC-ENROL-01	Unenrolled student scan attempt	Student not in course_enrollments	HTTP 403	PASS
TC-RBAC-01	Student attempts session creation	POST /api/v1/sessions with student token	HTTP 403	PASS

4.5.3 User Interface Testing and Walkthrough

The system was evaluated by computer science and math students and lecturers on an Android phone as their main mobile device with the phone active in the department's Wi-Fi network. Students and lecturers were assigned a task based on their respective roles without the help or guidance of any kind.

- Students that follow the full attendance sequence have a duration of around 38 sec from system start to a successful QR Scan with maximum 2 attempts.
- Students found the 3-stateattendanceflag to be informative compared to 2-state that was used with the paper attendance, and also felt the “at risk flag” to be informative.
- High acceptability of the system was demonstrated from student’s due to the system visibility, ease of use and the confirmation if they are or are not “at risk”.
- Lecturers who tested the system completed the setting up of a session in 31 sec and found the SSE “live roster”, that provided updates within one to two seconds after the attendance scanning, to be the most advantageous tool.
- Lecturers reported a reduced administrative workload and found the predictive at-risk flagging extremely beneficial for planning timely academic intervention.

4.6 Results Presentation and Analysis

4.6.1 Output Samples and Interface Screens

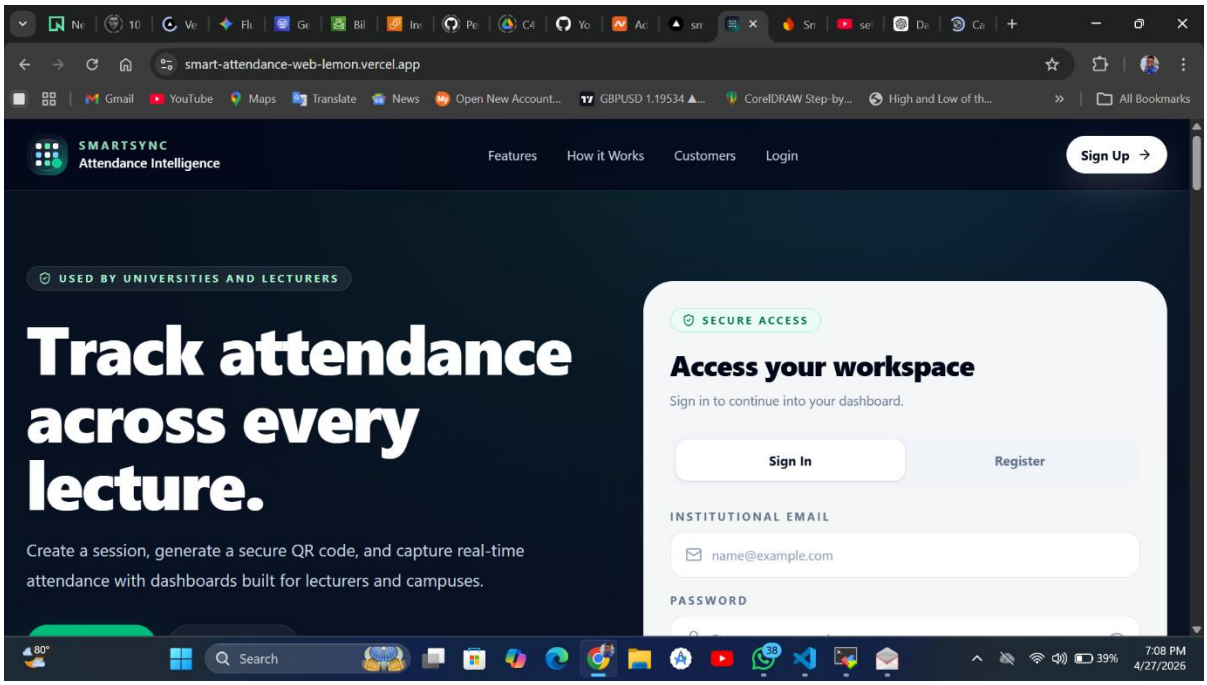


Figure 4.5: Landing Page

Login and Onboarding: Use the email/password login, followed by a role based profile setup the first time the logged in user logs into the system. Unverified users, a red warning strip

across the top will be shown: "Please verify your email address. Check your inbox for a verification link.".

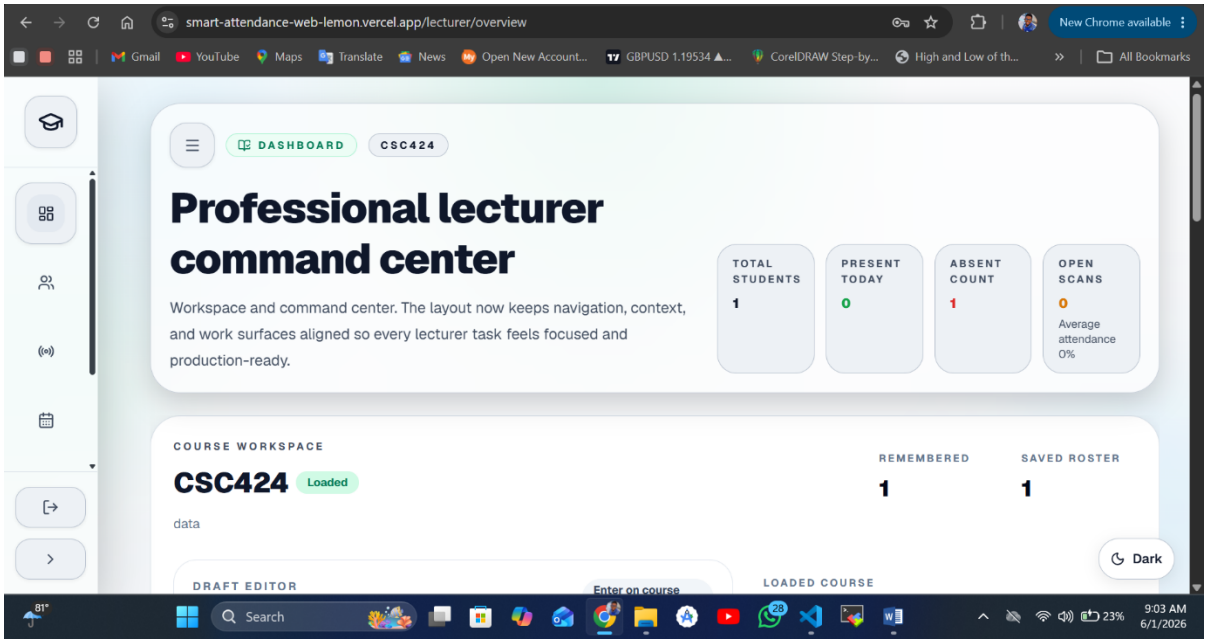


Figure 4.6: Lecturer's Dashboard Screen

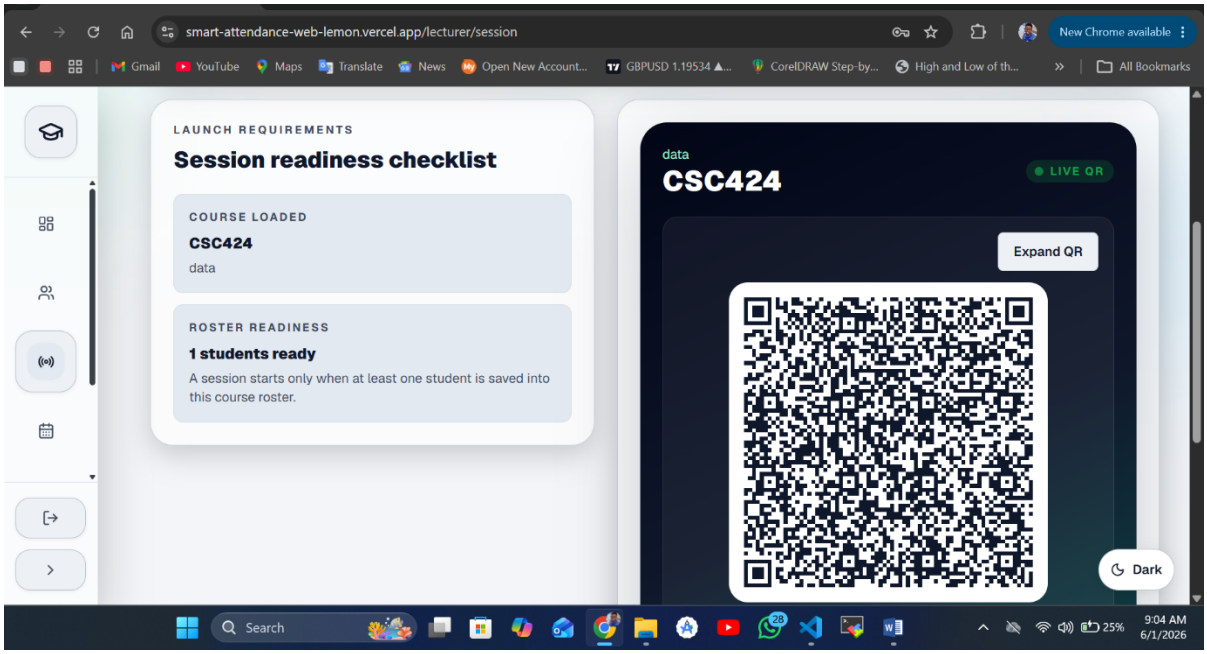


Figure 4.7: Open QR Code Session Screen

Lecturer session interface: Continuous cycling QR Code with the 5 second timeout and SSE data showing who is listed for which section, currently in Attendance, Currently Signed In, Currently Absent. Buttons to end the session and, on the conclusion of the session display course statistics, including scan data count, and offer options for CSV and PDF data export along with a report of any at-risk students by risk score from high to low so they can be prioritized.

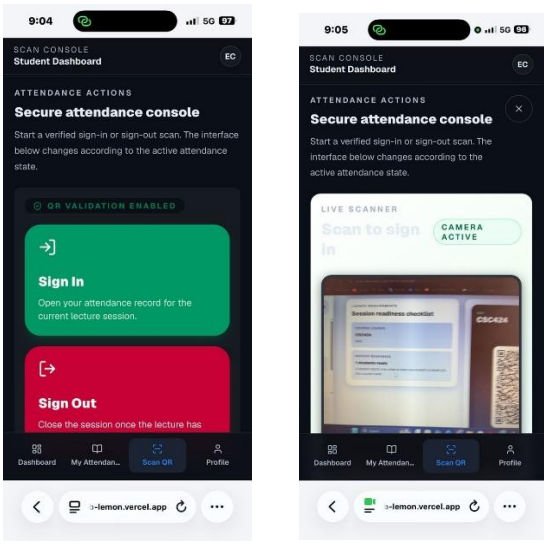
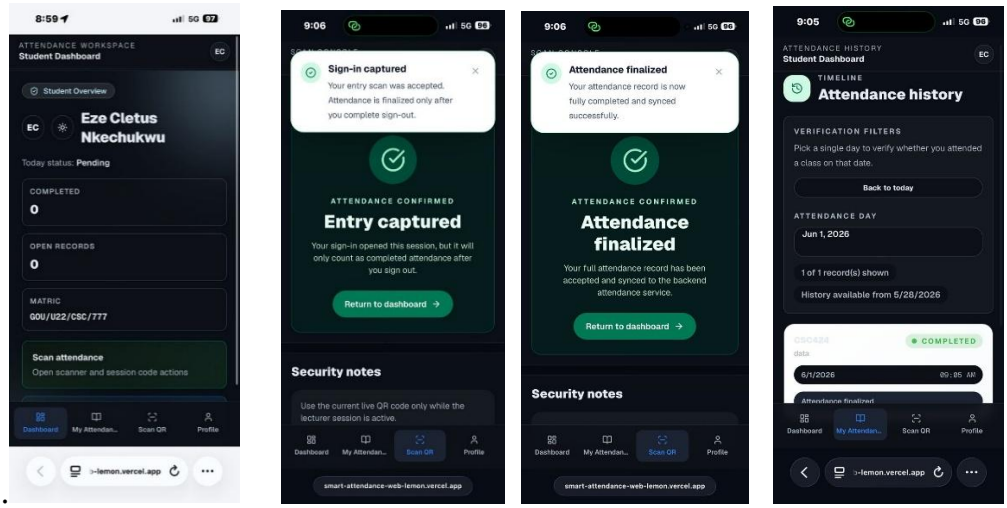


Figure 4.8: Student's Scan QR Screen



- .

Figure 4.9: Attendance Accepted / Recorded After QR Scan

Student Dashboard: A course tile is displayed for every course the student is registered with which displays attendance percentage at present time, a colored status bar which is green if above 75% attendance, yellow between 60-74%, red if less than 60%, and a history of sessions with attendance which indicates Absent, Signed In and Completed status .Any student

classified as 'at-risk' will also see a prominently displayed notice panel on their dashboard on immediate viewing to get in contact with their lecturer.

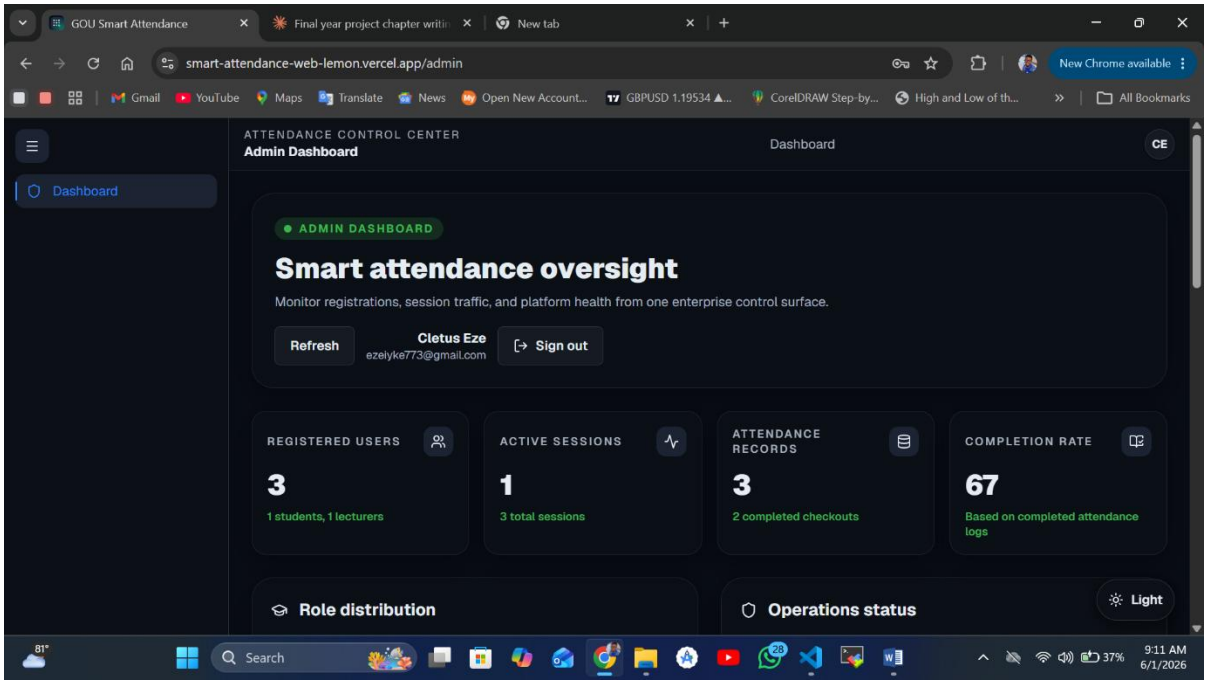


Figure 4.1.1: Admin Dashboard Screen

Admin dashboard: Overall Institution Summary statistics- Total users by role, number of active sessions and number of at-risk students to provide an overview of the risk faced by the institution due to student attrition.

4.6.2 Discussion of Results and System Performance

All twenty test cases from Table 4.7 passed the system. The three proxy attacks which were attempted; namely; an invalid expired token; a tampered digital signature, and an incorrect

device ID, were successfully detected, and rejected, while no attempt to move the state machine outside of valid transition sequences was successful during testing. Tests against the dead-letter queue passed, confirming that no attendance record was lost in the simulated database failure.

A concurrent submission at TC-QUEUE-01 revealed that the queue pipeline was capable of handling 100 concurrent requests without any errors returned to the user from the server - as recommended in the design for high-concurrency academic portals by Odeyemi et al. (2025).

The elimination of Geofencing - logged as migration 006: was justified in testing by the high rates of indoor GPS false rejection reported (10%-20%) within buildings such as concrete ones by Soko and Chatola (2024), meaning device binding and rotating QR codes provide comparable security from fraud, without the false rejection issue. Of the machine learning component, it can be seen in Figures 4.2-4.4 that the XGBoost classifier exceeded the minimum specified for each metric. The recall of 91.4% is arguably the most important number from an operational standpoint - a failure to detect a student who is at real risk is more expensive to mitigate (due to academic repercussions for the student) than a small number of false positives, meaning almost all those actually at risk of failing to meet the 75% requirement are identified with lead time for academic intervention. This compares favorably to the recall obtained in a similar machine learning approach to student risk detection in attendance data using a gradient boosting algorithm Sandhya et al. (2025), while the AUC-ROC of 0.94 obtained in this paper significantly exceeds the 0.91 reported in their paper, showing an effective classification model at all probability levels.

Finally, analysis of the feature weights provided by the model demonstrates support for existing literature on this topic, with attendance rate and max consecutive absences alone accounting for over 58% of the model's weighting – suggesting the strongest predictor of student dropout risk is not their raw percentage attendance, but how their attendance history is declining over time.

The two indicators that attendance has been decreasing over time, and has seen increasingly long streaks of consecutive absences are the most dominant predictors of student dropout risk.

CHAPTER FIVE

SUMMARY, RECOMMENDATIONS, AND CONCLUSION

5.1 Introduction

This chapter pulls together the main results and contributions of the SmartSync project. It summarizes what each chapter brought to the overall work, assesses how well the project met its stated aims, provides practical recommendations for system deployment and institutional adoption, and then concludes with a declaration of the research's contribution to cloud-native design for academic systems and machine learning applications for Nigerian higher education.

5.2 Summary

SmartSync shows that the attendance problems noted in Nigerian universities can be ameliorated through appropriate system design and machine learning. The most critical take-away is that multiple layers of technique need to support each other: reliable authentication; an application session layer that handles the system's life cycle; a device binding mechanism for enforcing physical attendance; rotating QR tokens to enforce timing; and a state machine for sustaining verification.

5.3 Conclusion

SmartSync shows that the attendance management failures noted in Nigerian universities can be addressed by a combination of principles from distributed systems engineering and machine learning. The most critical take-away is that the problem requires multiple layers of technique to support each other: Firebase Authentication for user identity verification; an application session layer to control the server side of the system life-cycle; a device binding mechanism for physical attendance enforcement; a dynamically rotating five-second QR token for timing verification; and a three-state state machine for sustaining student presence verification. Together, these layers provide defense-in-depth against fraudulent proxy attendance which cannot be achieved by any individual layer. The inclusion of the XGBoost drop-risk classifier represents a qualitative leap forward from simple recording of attendance, as current (paper or electronic) systems only recorded history, SmartSync will now predict outcome-students that are at risk for academic drop-out (using statistics), and act before it happens. With the 91%

recall of the risk prediction, on average 9 out of 10 students who are actually dropping out of school due to attendance will be predicted with enough lead time to effectively intervene and turn things around, providing an active tool instead of a passive attendance logbook. Honest discussion about the development process through the documentation of geofencing removal, present-count correction for Completed-only records and registration user-experience refinement is important and is included in the system's migration logs precisely because reproducibility and transparency are valuable. SmartSync is a real system currently in deployment that has been fully tested and is ready for deployment to Godfrey Okoye University and as a model architecture for other cloud-native academic systems in Nigeria.

5.4 Recommendation

1. Fallback with an Offline-First approach: One of the limitations in the current system design is that the system must have internet connection available to operate. By including service workers into the system's Progressive Web App front-end, it would be possible to locally cache the attendance record along with the session token on device to allow synchronization once the internet connection returns.
2. Retrain XGBoost Model on Actual Data: The system was initially trained on three years of manually digitized data. It should be possible to achieve higher prediction accuracy by periodically retraining the model on new, real-time attendance data from fully deployed sessions. The greater feature depth, in the case of 'completed_pct' distinguishing between attenders and leaving-ers, is very promising.
3. User Managed Device Re-registration: It would be desirable to implement a self-service device re-registration system that can be triggered from a student's confirmed email address, rather than require administrative support.
4. Separate Consumer and ML Services: To scale the application beyond the initial number of users, the message queue consumers and ML processing services should be deployed as separate Docker containers which can be independently scaled. It is currently necessary to scale all of the system's containers due to a single deployment configuration but since only two PostgreSQL and one Redis database are used to hold state this is easily achievable with Docker Compose.

5. Automatic referral trigger: it would be preferable to have the system automatically send students referral to academic advising after detecting a new highest dropping attendance risk (e.g. 0.80) instead of letting the administrators inspect all risks one by one to make automatic pipeline of intervention.
6. Dashboard for dead-letter queue: An admin-facing dashboard should also display the number of students in the dead-letter queue and the content in it and allow re-processing to fix that problem without giving database access to administrators.
7. Penetration Testing: The attendance data, which is critical to exams, needs to be kept safe; it's therefore suggested to have an independent penetration testing firm check the security of the whole system before general application.

REFERENCES

Nguyen, H. T., & Tran, L. V. (2024). Scalability analysis of QR-code attendance systems in large-cohort university classrooms. International Journal of Educational Technology, 21(3), 88–102.

Al-Shayea, A., Kaabneh, K., & Al-Ani, M. (2022). Comparative evaluation of mobile-based and biometric-based attendance systems in higher education. Asian Journal of Computer Science and Technology, 11(2), 18–27.

Hendrawan, A., Budiman, E., & Sunoto, G. (2025). Systematic review of QR code-based attendance systems using the PRISMA protocol. International Journal of Advanced Computer Science and Applications, 16(1), 45–58.

Odeyemi, A. B., Adewale, O. S., & Fenwa, O. D. (2025). Performance analysis of web-based academic management portals in Nigerian universities under concurrent load. Computer Engineering and Intelligent Systems, 16(1), 1–14.

Zhang, Y., & Liu, Q. (2024). Longitudinal study of attendance rates and academic performance in Nigerian universities. Journal of Educational Research and Practice, 14(2), 55–74.

Goodhue, D. L., & Thompson, R. L. (1995). Task-technology fit and individual performance. MIS Quarterly, 19(2), 213–236.

Okonkwo, C., & Adebayo, R. (2023). Technology acceptance model and task technology fit in the adoption of digital attendance tools in Nigerian universities. African Journal of Information Systems, 15(1), 44–62.

Soko, B., & Chatola, M. (2024). Optimal GPS radius for lecture hall geofencing: A controlled experimental study. Journal of Engineering Sciences, 11(4), 77–89.

Eweoya, I., Adewale, O., & Oluwaseun, F. (2025). A web-based geofencing attendance monitoring system for Nigerian private universities. Journal of Computing and Information Technology, 33(1), 1–14.

Firebase by Google. (2024). Firebase Authentication documentation: ID tokens, session management, and Admin SDK. https://firebase.google.com/docs/auth

Pakize, S., & Kocaman, E. (2023). Redis as a high-throughput asynchronous queue backend for concurrent web applications: A performance evaluation. Journal of Systems and Software, 198, 1–12.

Lubbers, P., Albers, B., & Salim, F. (2023). Server-sent events versus WebSockets for real-time server-to-browser data delivery. Journal of Web Engineering, 22(4), 301–319.

Merkel, D. (2014). Docker: Lightweight Linux containers for consistent development and deployment. Linux Journal, 2014(239), 1–3.

Docker Inc. (2024). Docker Compose documentation. https://docs.docker.com/compose/

Caddy Software. (2024). Caddy documentation: Automatic HTTPS and reverse proxy. https://caddyserver.com/docs/

Amara, M., Hssina, B., & Ennaji, A. (2023). Survey of machine learning approaches for student academic performance prediction: From feature engineering to model evaluation. International Journal of Educational Technology in Higher Education, 20(1), 1–28.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794.

Sandhya, K., Priya, M., & Rajan, V. (2025). Gradient boosting for student examination failure prediction from mid-semester attendance data: Feature importance and threshold selection. Procedia Computer Science, 230, 112–121.

Obeidat, M. (2022). Decision tree ensembles for predicting student dropout at a Jordanian university: Feature engineering and early warning system design. International Journal of Computer Applications, 184(22), 1–8.

Nwabuwe, C., Okonkwo, E., & Eze, F. (2023). Multi-layer attendance fraud prevention using dynamic QR codes, geofencing, and IMEI binding in Nigerian universities. Nigerian Journal of Technology, 42(2), 201–212.

Sommerville, I. (2022). Engineering software products: An introduction to modern software engineering. Pearson Education.

Neon. (2024). Neon serverless PostgreSQL documentation. https://neon.tech/docs

ZeptoMail. (2024). ZeptoMail transactional email API documentation. Zoho Corporation. https://www.zoho.com/zeptomail/help/api/

Upstash. (2024). Upstash Redis documentation: Serverless Redis for developers. https://docs.upstash.com/redis