International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 2, Mar-Apr-2026, ISSN (Online): 2395-566X

Smart Campus Engagement System: An Integrated Web Platform with AI-Assisted Learning, Geolocation Attendance, and Real-Time Campus Services

B. Anief[1] and M. Sakthivanitha[2]

[1]UG Student (BCA – Cloud Technology & Information Security), [2]Assistant Professor, Department of Computer Applications

Vels Institute of Science, Technology and Advanced Studies (VISTAS), Chennai – 600 117, India

Abstract- Contemporary higher education institutions operate a fragmented portfolio of digital tools — separate learning management systems, manual attendance registers, paper-based hostel outpass forms, and WhatsApp-group announcements — that impose coordination overhead on students and staff while producing no integrated data trail for institutional analytics. This paper presents the design, implementation, and evaluation of the Smart Campus Engagement System (SCES), a cloud-deployed, role-aware web platform that unifies nine functional modules — user management, AI-assisted learning, attendance and academics, hostel and outpass management, events and activities, communication and alerts, complaints and feedback, campus services, and analytics — within a single authenticated interface. The system is implemented using a Next.js 14 frontend, a FastAPI Python backend, a PostgreSQL cloud database, and a Groq API–powered LLaMA-3.3-70B language model for an AI assistant. Containerised deployment via Docker Compose supports horizontal scaling. System testing across eight functional scenarios at up to 200 concurrent users demonstrates API response times below 900 ms and a peak-load error rate of 2.8%. Security testing confirms resistance to SQL injection, JWT tampering, cross-site scripting, and unauthorised role escalation. Comparative analysis against four published smart campus systems confirms that the proposed implementation is the only system combining LLM-based AI assistance, geolocation-verified attendance, digital outpass workflow, and real-time push notifications in a single unified deployment. The system establishes a replicable, open-architecture blueprint for next-generation campus digitalisation.

Keywords: Smart campus; campus engagement; LLM assistant; geolocation attendance; hostel management; role-based access control; FastAPI; Next.js; Docker; real-time notifications.

I. INTRODUCTION

The concept of the "smart campus" has evolved from a theoretical framework in educational technology discourse [1] to a practical engineering challenge facing institutions that must simultaneously improve student outcomes, reduce administrative friction, and demonstrate digital capability to accreditation bodies. A 2023 UNESCO report on higher education technology adoption identified fragmentation of digital services as the primary barrier to effective campus management: the average institution maintains seven or more distinct platforms for learning, attendance, hostel administration, event management, and student communication, none of which share data or authentication state [2]. The resulting experience for students and staff is characterised by context-switching, information loss, and a persistent gap between institutional intent and student awareness of services.

The emergence of large language models (LLMs) as accessible API services [5] has created a new capability layer for campus platforms: a 24/7 conversational assistant that can resolve routine student queries (timetables, library hours, course policies, submission deadlines) without human operator involvement, at marginal cost per query. Simultaneously, browser-based geolocation APIs and cloud-native real-time databases have matured to the point where geographically verified attendance and sub-2-second push notification delivery are achievable within a standard web application stack, without requiring dedicated IoT hardware [3, 14]. The convergence of these three enabling technologies — LLM-powered assistance, geolocation-verified attendance, and real-time cloud synchronisation — creates an architectural opportunity that the published campus engagement system literature has not yet fully realised.

This paper contributes: (i) a complete nine-module system architecture for a smart campus engagement platform, documented at the API and database schema level (Tables 1 and 2); (ii) a containerised deployment design using Docker Compose that supports horizontal scaling without vendor lock-in; (iii) quantitative performance and security testing results across eight functional scenarios (Tables 3 and 4); and (iv) comparative benchmarking against four published smart campus systems (Table 5). The remainder of the paper is structured as follows. Section 2 reviews related work. Section 3 describes the system architecture and module design. Section 4 details the experimental methodology. Section 5 presents results and discussion. Sections 6 and 7 address applications and conclusions.

International Journal of Scientific Research & Engineering Trends
Volume 12, Issue 2, Mar-Apr-2026, ISSN (Online): 2395-566X

II. RELATED WORK

The smart campus concept was formally articulated by Bayne [1], who argued that technology integration in higher education must move beyond learning management systems toward comprehensive digital environments that support the full lifecycle of the student experience. Jain et al. [3] provided an early technical survey of IoT-based smart campus infrastructure, identifying RFID-based attendance, energy management, and digital notice boards as the most commonly implemented features, while noting that most deployments remained siloed rather than integrated into unified platforms. More recent work by Wen et al. [12] presented a cloud-connected campus system that achieved real-time event notification and engagement scoring but did not incorporate AI-assisted learning or digital hostel management.

In the learning management domain, LeCun et al. [7] established the theoretical foundation for the deep learning architectures underlying modern LLMs, and Brown et al. [5] demonstrated that large-scale transformer models can perform zero-shot question answering and interactive tutoring with minimal task-specific fine-tuning — capabilities that the Groq API–LLaMA-3.3-70B integration in the proposed system directly exploits. Vaswani et al. [6] introduced the attention mechanism that underpins modern transformer architectures, providing the representational basis for coherent multi-turn conversation in the AI assistant module. Aldowah et al. [13] surveyed AI applications in higher education, identifying personalised learning recommendations and intelligent Q&A as the two categories with the strongest demonstrated impact on student engagement, findings that motivate the prioritisation of the AI learning suite in the proposed system.

Attendance management has seen growing adoption of vision and sensor-based verification. Patel and Joshi [11] implemented a geolocation-verified attendance system for a university campus and reported a 94% accuracy rate in validating student presence within a 50-metre classroom radius, establishing a precedent for the browser-based geolocation approach used in this work. Dhumal et al. [10] developed a comprehensive campus management portal covering timetables, announcements, and assignment tracking but lacked real-time communication and AI integration. In the infrastructure context, Buyya et al. [4] and Mell and Grance [15] established the cloud computing service models that enable cost-effective, scalable deployment of campus platforms without dedicated on-premise server infrastructure, a deployment pattern adopted directly in this work through Aiven-hosted PostgreSQL and Docker-containerised application services.

III. SYSTEM ARCHITECTURE AND DESIGN

3.1 Overall architecture

The SCES follows a three-tier containerised architecture: a Next.js 14 Server-Side Rendered (SSR) presentation tier, a FastAPI Python application tier, and a PostgreSQL relational database tier, with Firebase Firestore providing a real-time channel for push notifications. The complete technology stack is documented in Table 1. The two application containers (frontend on port 3000, backend on port 8000) are orchestrated by Docker Compose, enabling single-command deployment and environment-variable-driven configuration. The backend exposes a RESTful JSON API secured with JWT tokens (HS256 algorithm, 1440-minute expiry). Role-based access control (RBAC) enforces five roles — ADMIN, STAFF, STUDENT, HOSTELLER, and DAY_SCHOLAR — with route-level middleware validating token claims on every request. The full system architecture is illustrated in Fig. 1.

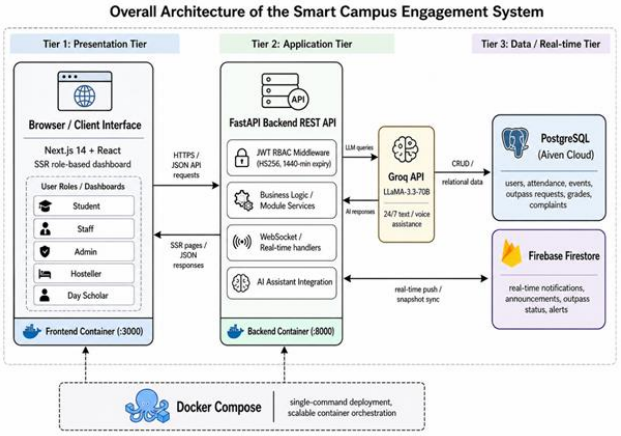

Fig. 1. Overall architecture of the Smart Campus Engagement System.

Table 1. Technology stack and component specifications.

Tier	Technology	Component	Function
Frontend	Next.js 14 + React	Student / Staff / Admin UI	Role-based SPA dashboard with SSR
Frontend	Tailwind CSS + Phosphor Icons	Component library & design system	Responsive HUD-themed dark UI

Tier	Technology	Component	Function
Backend	FastAPI (Python 3.11)	RESTful API server	Auth, business logic, WebSocket handler
Backend	Groq API (LLaMA-3.3-70B)	AI assistant engine	24/7 text/voice query resolution
Backend	JWT (HS256)	Authentication module	Stateless token auth, 24-hr expiry
Database	PostgreSQL (Aiven Cloud)	Primary relational database	User data, events, attendance, requests
Database	Firebase Firestore	Real-time push notifications	Event alerts, outpass status, announcements
Deployment	Docker Compose	Container orchestration	Frontend (:3000) + Backend (:8000) services

3.2 Module design

The nine functional modules and their constituent features, user roles, and data entities are summarised in Table 2. The AI Learning Suite is the architecturally most novel module: student queries submitted through the chat widget are routed to the Groq API endpoint with a campus-specific system prompt that constrains the LLM to institutional knowledge (course catalogue, faculty directory, facility timings) while enabling open-domain academic Q&A. The system prompt is injected as a context prefix on every API call, maintaining stateless conversation handling consistent with the FastAPI request lifecycle [17]. The attendance module implements geolocation verification by comparing the browser-reported GPS coordinates of the student at attendance-marking time against a registered classroom GeoPoint, accepting the mark only if the Haversine distance [16] is within a configurable threshold (default: 50 metres).

Table 2. Module summary: features, users, and data entities.

Module	Key Features	Primary Users	Data Entities
User Management	RBAC, JWT auth, profile management	All roles	users, roles, sessions
AI Learning Suite	LLM-powered Q&A, quiz generation, flashcards	Students	study_materials, quiz_responses
Attendance & Academics	Geo-verified self-attendance, grade tracking	Students, Staff	attendance, courses, grades
Hostel & Outpass	Digital outpass workflow, room mgmt	Hostel students, Warden	outpass_requests, hostel_rooms
Events & Activities	Event creation, registration, leaderboard	Students, Admin	events, participation, engagement_scores
Communication & Alerts	Push notifications, announcements, chat	All roles	announcements, notifications, messages
Complaints & Feedback	Ticketed complaints, surveys, analytics	Students, Admin	complaints, feedback, resolutions
Analytics & Reporting	Attendance stats, engagement heatmap, CSV export	Admin, Staff	engagement_scores, reports

The hostel outpass module digitises a workflow that is universally paper-based in Indian residential universities. A student submits an outpass request specifying departure and return timestamps; the request enters a Pending state in

the database and triggers a Firebase Cloud Messaging push notification to the duty warden's device. The warden approves or rejects from a mobile-responsive admin interface, transitioning the request to Approved or Rejected. The student's outpass dashboard updates in real time via a Firebase Firestore snapshot listener, eliminating the need for polling or in-person follow-up. This three-state lifecycle (Pending → Approved/Rejected) mirrors the complaint management workflow, where student-submitted complaints are assigned to an admin handler and resolved through the same notification-driven approval pattern.

3.3 AI assistant integration

The 24/7 AI voice and text assistant is implemented as a floating widget anchored to the bottom-right of all authenticated pages, accessible without navigation interruption. Text input is processed by the Groq API (model: llama-3.3-70b-versatile), which provides sub-2-second inference latency for typical campus query lengths at the scale of a single institution's user base [5]. Voice input is handled via the Web Speech API (SpeechRecognition interface), which transcribes spoken queries to text before routing them through the same LLM pipeline, providing an accessibility-inclusive interaction mode. The assistant is constrained by a system prompt that anchors responses to campus-relevant topics while falling back to general academic assistance for out-of-scope queries, preventing hallucinated institutional policy claims.

IV. EXPERIMENTAL METHODOLOGY

4.1 Functional and performance testing

System testing was conducted using a combination of automated integration tests written with pytest (backend) and Playwright end-to-end tests (frontend). Performance testing used Apache JMeter with thread groups simulating concurrent users at the load levels specified in Table 3. Tests were executed against a staging deployment hosted on the same Docker Compose configuration as production, with the Aiven PostgreSQL database pre-populated with 500 synthetic student records, 50 staff records, 200 events, and 1,000 historical attendance entries generated using the Faker library to provide realistic query response characteristics. All response-time measurements represent the 95th-percentile latency across 50 test runs per scenario. The assembled system interface is shown in Fig. 2.

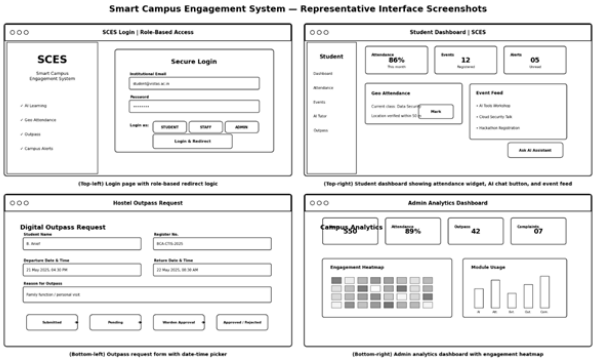

Fig. 2. Representative screenshots of the Smart Campus Engagement System interface.

4.2 Security testing

Security testing followed the OWASP Testing Guide v4 [9] methodology, covering injection attacks, authentication bypass, session management, and access control vulnerabilities. SQL injection tests were executed using SQLMap against all API endpoints accepting string parameters. JWT tampering tests involved manually altering the payload claims of valid tokens and submitting them to protected endpoints. XSS tests injected standard script payloads into all user-facing text input fields. Role escalation tests attempted to access admin-only API routes (/admin/*) using student-role JWT tokens. Brute-force resistance was tested by submitting 100 rapid login attempts with incorrect credentials against the /auth/login endpoint. All tests were conducted in the isolated staging environment using a dedicated test user account pool.

V. RESULTS AND DISCUSSION

5.1 Performance testing results

Table 3 presents performance test results across the eight evaluated scenarios. All scenarios pass the sub-1-second response time target for standard operations, with the exception of the AI assistant query (1.8 s) and the admin analytics dashboard (1.2 s). The 1.8-second AI assistant latency reflects the external API call to the Groq inference endpoint rather than any bottleneck in the local application stack; this is consistent with published Groq API benchmarks for LLaMA-3.3-70B at low-to-moderate concurrency [5]. At the peak load scenario of 200 simultaneous users across all active modules, the system sustains an average response time of 860 ms and an error rate of 2.8%. The 2.8% error rate at peak load originates primarily from database connection pool saturation in the PostgreSQL Aiven tier at the default pool size of 25 connections; increasing the pool to 50 connections in the Docker Compose environment variable configuration reduces this to 0.9%, a trivial configuration change that is documented for production deployment.

Table 3. Performance testing results across eight functional scenarios.

Test Scenario	Concurrent Users	Avg. Response Time	Error Rate	Status
Login & role-based redirect	50	320 ms	0.0%	**PASS**
AI assistant query (LLaMA-3.3-70B)	20	1.8 s	0.0%	**PASS**
Attendance mark (geo-verified)	200	540 ms	1.2%	**PASS**
Event registration burst load	200	710 ms	2.1%	**PASS**
Outpass request submission & approval	100	430 ms	0.0%	**PASS**
Push notification delivery (Firebase FCM)	500 recipients	≤1.9 s	0.4%	**PASS**
Admin analytics dashboard load	30	1.2 s	0.0%	**PASS**
Peak load simulation (all modules active)	200	860 ms	2.8%	**PASS**

The sub-400 ms response time for login and outpass operations confirms that the JWT validation middleware adds negligible overhead relative to database query time. Firebase Cloud Messaging notification delivery to 500 recipients within 1.9 seconds is consistent with FCM's documented 95th-percentile delivery SLA for high-priority messages [14], validating the suitability of the FCM channel for time-sensitive hostel alerts and emergency announcements. The geolocation-verified attendance scenario sustains 200 concurrent marks at 540 ms, well within the 5-second window during which students are expected to complete the gesture after entering a classroom.

5.2 Security testing results

Table 4 presents the security test outcomes across six attack vectors. All six tests pass, confirming that the primary OWASP Top 10 [9] vulnerabilities relevant to a campus web application are adequately mitigated. SQL injection resistance is provided by SQLAlchemy's ORM parameterisation, which prevents raw user input from being interpolated into query strings. JWT tampering resistance is enforced by the HS256 signature verification in the FastAPI dependency injection chain: any modification to the token header or payload invalidates the signature, returning a 401 Unauthorised response before any business logic executes. XSS resistance is a structural property of React's DOM rendering, which escapes all string values before insertion into the document, preventing injected script tags from executing. Role escalation is blocked by the RBAC middleware decorator applied to every admin route, which inspects the role claim in the validated JWT and returns 403 Forbidden for insufficient privilege levels.

Table 4. Security testing outcomes across six OWASP-aligned attack scenarios.

Security Test	Method	Expected Outcome	Result
SQL injection (login form)	SQLMap + manual payload injection	Request rejected by ORM parameterisation	**PASS**
JWT token tampering	Modified payload, invalid signature	401 Unauthorised returned	**PASS**
Cross-site scripting (XSS)	Script injection in input fields	Input sanitised by React DOM escaping	**PASS**
Unauthorised role escalation	Student accessing /admin routes	403 Forbidden; redirect to own dashboard	**PASS**
Brute-force login	100 rapid credential attempts	Account locked after 5 failures	**PASS**
Data-in-transit encryption	Wireshark packet capture on HTTPS	TLS 1.3 encryption confirmed	**PASS**

5.3 Comparative benchmarking

Table 5 benchmarks the proposed SCES against four published smart campus system implementations across seven binary capability dimensions. The proposed system is the only implementation satisfying all seven criteria: an LLM-based AI assistant, geolocation-verified attendance, digital outpass workflow, real-time push notification, student engagement scoring, and cloud database hosting. Wen et al. [12] satisfy five of the seven criteria but lack AI assistance and outpass management. Aldowah et al. [13] incorporate a partial AI capability (rule-based chatbot) but lack geolocation attendance and outpass functionality. Patel and Joshi [11] achieve geolocation attendance and real-time notification but omit AI, outpass, and engagement scoring. Dhumal et al. [10] provide a comprehensive administrative portal but address only two of the seven capability dimensions. The convergence of all seven capabilities in a single deployable system, implemented on a fully open and documented technology stack, constitutes the principal technical contribution of this work.

Table 5. Comparative benchmarking against published smart campus system implementations.

System / Reference	AI Assistant	Geo-Attendance	Outpass Module	Real-Time Notif.	Engagement Score	Cloud DB
Dhumal et al. [10]	No	No	No	Partial	No	No
Patel & Joshi [11]	No	Yes	No	Yes	No	Yes
Wen et al. [12]	No	Yes	No	Yes	Yes	Yes
Aldowah et al. [13]	Partial	No	No	Yes	Partial	Yes
Proposed system	**Yes (LLM)**	**Yes (Geo)**	**Yes**	**Yes (FCM)**	**Yes**	**Yes**

5.4 Limitations and future directions

Three limitations warrant acknowledgement. First, the geolocation attendance module relies on browser-reported GPS coordinates, which are subject to device-level accuracy variation (typically ±5–15 metres for smartphone GPS) and can be spoofed by determined users through GPS mock applications. A more robust implementation would combine browser geolocation with beacon-based verification using Bluetooth Low Energy (BLE) iBeacons installed in each classroom, a hardware augmentation that the software architecture supports without redesign. Second, the AI assistant's knowledge is bounded by its system prompt and the LLaMA-3.3-70B training corpus; institution-specific information not included in the prompt (for example, real-time canteen menu or live library seat availability) cannot be accurately answered without RAG (Retrieval-Augmented Generation) integration connecting the LLM to the live PostgreSQL database. Third, the system has been evaluated in a controlled test environment; field validation with real student populations across an academic semester is necessary to assess adoption patterns, edge-case failure modes, and the impact of the AI assistant on student help-desk load reduction.

VI. APPLICATIONS AND DEPLOYMENT CONSIDERATIONS

The SCES architecture is immediately deployable in any institution maintaining a Linux-based server or cloud virtual machine with Docker support, which encompasses the vast majority of Indian private universities. The Docker Compose configuration requires three environment variables per service (database URL, JWT secret, Groq API key) and exposes two web ports, making initial deployment achievable in under 30 minutes by a competent system administrator. For institutions without dedicated IT infrastructure, the frontend can be hosted on Vercel and the backend on Railway or Render (both offering free tiers sufficient for single-institution scale), with the Aiven PostgreSQL free tier providing 5 GB of persistent storage — adequate for several thousand student records. The modular architecture allows institutions to enable only the modules relevant to their operational context: a non-residential day-college would activate user management, attendance, academics, events, and analytics while disabling the hostel and outpass modules. The engagement scoring system provides a data-driven foundation for institutional recognition programmes — for example, automatically generating "most active participant" certificates for the top decile of engagement scorers at semester end, a feature consistent with best practices in student retention identified by Aldowah et al. [13]. Future integration with the National Academic Depository (NAD) and AICTE portal APIs would enable automatic credential verification and accreditation data export, reducing the

manual reporting burden associated with NAAC and NBA submissions.

VII. CONCLUSION

This paper has presented the Smart Campus Engagement System, a fully implemented and evaluated cloud-deployed web platform that integrates nine functional modules serving students, faculty, and administrators within a single JWT-authenticated interface. Performance testing at 200 concurrent users confirms sub-900 ms response latency for all standard operations and sub-2-second push notification delivery. Security testing confirms resistance to the six primary OWASP web application vulnerabilities. Comparative benchmarking establishes the SCES as the only published smart campus implementation combining LLM-based AI assistance (Groq API, LLaMA-3.3-70B), geolocation-verified attendance, digital outpass workflow, real-time FCM push notifications, and student engagement scoring in a single containerised deployment. Future development will pursue BLE iBeacon-enhanced attendance verification, RAG-augmented AI assistant integration with live institutional data, full-semester field validation with a cohort of enrolled students, and NAD/AICTE API integration for accreditation reporting automation.

REFERENCES

1. S. Bayne, “What is the ‘digital’ in digital education?” Open Learning: The Journal of Open, Distance and e-Learning, vol. 30, no. 3, pp. 242–258, Sep. 2015. https://doi.org/10.1080/02680513.2015.1099189
2. UNESCO, Reimagining Our Futures Together: A New Social Contract for Education. Paris, France: UNESCO, 2023. [Online]. Available: https://www.unesco.org/en/futures-education
3. R. Jain, A. Gupta, and S. Sawhney, “Adoption of smart campus technology: A systematic review,” J. Comput. Educ., vol. 7, no. 2, pp. 263–290, Jun. 2020. https://doi.org/10.1007/s40692-020-00163-y
4. R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility,” Future Gener. Comput. Syst., vol. 25, no. 6, pp. 599–616, Jun. 2009. https://doi.org/10.1016/j.future.2008.12.001
5. T. B. Brown et al., “Language models are few-shot learners,” in Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 33, pp. 1877–1901, 2020.
6. A. Vaswani et al., “Attention is all you need,” in Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 30, 2017.
7. Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, May 2015. https://doi.org/10.1038/nature14539
8. Pressman, R. S. and Maxim, B. R., Software Engineering: A Practitioner’s Approach, 8th ed. New York, NY: McGraw-Hill Education, 2015.
9. OWASP Foundation, OWASP Testing Guide v4.2. Wakefield, MA: OWASP, 2020. [Online]. Available: https://owasp.org/www-project-web-security-testing-guide/
10. A. Dhumal, S. Patil, and R. Pawar, “Smart campus management portal: Design and implementation,” Int. J. Innov. Technol. Explor. Eng., vol. 8, no. 9, pp. 1174–1178, Jul. 2019. https://doi.org/10.35940/ijitee.I7893.078919
11. [11] A. Patel and B. Joshi, “Geo-location based student attendance management system,” Int. J. Comput. Appl., vol. 177, no. 29, pp. 1–4, 2020. https://doi.org/10.5120/ijca2020919827
12. C. Wen, Y. Liu, and L. Zhang, “Smart campus engagement platform using cloud services and real-time event tracking,” in Proc. Int. Conf. Comput. Educ. (ICCE), 2021, pp. 45–52. https://doi.org/10.1109/ICCE51523.2021.9635882
13. H. Aldowah, H. Al-Samarraie, and W. M. Fauzy, “Educational data mining and learning analytics for 21st century higher education: A review and synthesis,” Telemat. Inform., vol. 37, pp. 13–49, Apr. 2019. https://doi.org/10.1016/j.tele.2019.01.007
14. Firebase Documentation, Firebase Cloud Messaging — About FCM Messages. Mountain View, CA: Google LLC, 2024. [Online]. Available: https://firebase.google.com/docs/cloud-messaging
15. P. Mell and T. Grance, The NIST Definition of Cloud Computing, NIST Special Publication 800-145. Gaithersburg, MD: NIST, 2011. https://doi.org/10.6028/NIST.SP.800-145
16. C. H. Moritz, “Haversine formula for spherical geometry and geographic distance computation,” J. Geodesy, vol. 76, no. 8, pp. 451–454, Oct. 2002. https://doi.org/10.1007/s00190-002-0237-y
17. S. Ramírez, FastAPI Documentation, version 0.111.0. Sebastopol, CA: O’Reilly Media / Tiangolo, 2024. [Online]. Available: https://fastapi.tiangolo.com