A Python-Based Telegram API Framework for Automated Attendance Monitoring - A Review

Diwakar Singh (Student)

Department of Computer Science Engineering, Shri Ramswaroop College of Engineering and Management

diwakar317@gmail.com

Emily Shristi Kumar (Student)

Department of Computer Science Engineering, Shri Ramswaroop College of Engineering and Management

emilykr1311@gmail.com

Divyanshu Mishra (Assistant Professor)

Department of Computer Science Engineering, Shri Ramswaroop College of Engineering and Management

divyanshu87651707@gmail.com

Sarika Singh (Assistant Professor)

Department of Computer Science Engineering, Shri Ramswaroop College of Engineering and Management

sarikasingh2494@gmail.com

Abstract— **Attendance management has moved from paper logs to digital platforms that improve accuracy and operational efficiency. This review surveys literature on attendance technologies with an emphasis on systems that use the Telegram API and Python for near real-time tracking. Manual attendance methods are simple but prone to human error. Technologies such as RFID and biometrics increase reliability but add cost and deployment complexity. Telegram-based solutions developed with Python can offer a low-cost, remotely-accessible alternative suitable for small institutions, distributed teams, and pilot deployments. Drawing on academic papers, GitHub projects, and tutorials, this paper reviews representative work, proposes an example Python–Telegram bot architecture, outlines implementation considerations, examines evaluation practices, and suggests directions for AI-assisted verification and stronger privacy controls.**

Keywords: Telegram; attendance; bot; verification

I. INTRODUCTION

Automated attendance systems are increasingly used in educational and organizational settings to reduce manual labor, minimize errors, and improve accountability. Historically, organizations relied on paper-based logs and spreadsheets—methods that are simple but vulnerable to missed entries, duplicate records, and unauthorized changes [1][9]. Digital technologies such as RFID readers, biometric scanners (fingerprint, iris), and mobile applications have improved reliability and throughput, yet they often require specialized hardware, complex installation, and ongoing maintenance [2][3].

Messaging platforms—particularly Telegram—have recently been adopted as lightweight interfaces for attendance systems because of their robust API, cross-platform availability, and ease of integration with backend services [4][8]. Python, with libraries such as python-telegram-bot and requests, enables rapid prototyping and deployment of Telegram bots that can handle check-ins, request user location or media, send reminders, and interact with databases or dashboards [5][10]. This review focuses on Telegram-based attendance systems implemented in Python, highlighting how they serve contexts where dedicated hardware is impractical (e.g., remote teams, small-scale deployments, or rapid pilots).

The remainder of the paper is organized as follows. Section II surveys relevant literature and implementations. Section III synthesizes themes, trade-offs, and open problems observed across works. Section IV provides an example system architecture and implementation notes. Section V discusses evaluation, limitations, and future directions. Section VI concludes.

II. LITERATURE REVIEW

This literature review organizes prior work on automated attendance systems into thematic categories, compares verification strategies and system architectures, and highlights evaluation practices and persistent gaps. The sources surveyed include proof-of-concept prototypes, small pilot deployments, integration tutorials, and open-source implementations that use Telegram as a user-facing channel or notification bus [1–10].

A. Taxonomy of approaches

Prior work falls into four practical categories:

1. Command- or bot-driven check-ins

- These systems rely on user-issued commands (e.g., /check-in) to register presence. They are inexpensive and fast to

deploy because they only require a Telegram bot and a backend service. Several community projects and tutorials illustrate quick prototypes that collect a simple acknowledgment or timestamp from users via bot commands and persist records to a database [5][8][10].

- Strengths: minimal hardware, low cost, familiar UI for users already on Telegram.

- Weaknesses: weak guarantees about physical presence or user authenticity — vulnerable to impersonation or remote spoofing.

2. Location-aware (geofencing) solutions

- Tools such as AttendanceBot perform GPS-based validation, accepting user-shared location data and comparing it to permitted geofenced areas [4]. These approaches add a coarse spatial check that is useful for dispersed field workers.

- Strengths: reasonable for cases where presence within a geographic zone is sufficient.

- Weaknesses: GPS can be inaccurate in dense urban or indoor settings and is susceptible to falsified location data unless combined with additional sensors or detection methods.

3. Hardware-assisted identity binding (RFID, IoT gateways)

- Systems integrating RFID readers, microcontrollers (e.g., ESP32), and other sensors aim to link physical events (card swipes, proximity) to user identities and then push notifications through Telegram [1][2]. Combining RFID with cameras or ultrasonic sensors is an emerging design pattern to increase confidence that the person and the device co-locate.

- Strengths: stronger identity binding than command-only methods; suitable for controlled-entry environments.

- Weaknesses: added hardware cost, deployment complexity, and potential failure modes (tag loss, reader range limitations). Performance under higher throughput needs more evidence [2].

4. Biometric and image-based systems

- Face-recognition pilots use OpenCV-based algorithms (e.g., LBPH), neural models, and Telegram notifications to confirm presence with an image submitted by the user or captured at a gate; some studies outline pipelines that detect faces locally and notify administrators via Telegram [6][7]. More advanced proposals include liveness detection to reduce spoofing from photos or video replays.

- Strengths: can provide strong per-person verification when reliably implemented.

- Weaknesses: privacy concerns, legal/regulatory requirements for biometric data, and vulnerability to adversarial attacks unless anti-spoofing measures are used.

B. Representative implementations and findings

- AttendanceBot (location-based) reported rapid confirmation times and a user-friendly workflow in field trials, but evaluation was limited to a very small sample and the design remained exposed to location spoofing risks [4].

- Akbar et al. presented a multi-sensor lecturer attendance system (RFID + ESP32-CAM + ultrasonic sensors) that forwards events through Telegram, demonstrating how sensor fusion can raise confidence in an attendance record while also increasing costs and privacy exposure [1]. Their work highlights practical trade-offs like RFID read range and the added privacy impact of image capture.

- Kurniawan et al. documented RFID-driven student attendance with Telegram notifications and a backend database; they reported throughput problems when processing many reads in quick succession, signaling the need for performance engineering in production deployments [2].

- Conversational NLU systems (Rasa-based) have been explored as front-ends to let users query attendance data and related records via Telegram; however, prototypes focused on query handling rather than mark-in verification and lack robust evaluation of NLU accuracy and error-handling in live scenarios [3].

- Tutorials and GitHub projects provide practical, reproducible examples of integrating Python Telegram libraries and simple recognition pipelines. These resources are valuable for prototyping but frequently omit rigorous evaluation and security hardening [5][7][8][10].

C. Evaluation practices and methodological gaps

Across these works, evaluation tends to emphasize feasibility and functional correctness over rigorous, generalizable metrics. Common weaknesses include:

- Small-scale pilots or synthetic tests that do not measure behavior under realistic concurrency, network variability, or adversarial conditions [2][4].

- Limited reporting on user-centered metrics such as task completion rates, false-accept / false-reject rates for

verification, or user perception of privacy and usability [3].

- Sparse attention to end-to-end performance: bot latency, backend throughput, and database scaling are seldom stress-tested beyond ad-hoc experiments. One practical report found substantial processing time when handling dozens of RFID card reads [2].

D. Security, privacy, and legal considerations

Many implementations use Telegram primarily for delivery and user interaction; Telegram itself rarely provides cryptographic verification of physical presence and typically functions as a conduit for events [4][8]. Important concerns raised across the literature include:

- Data minimization and retention: captured images, location traces, and biometric templates require explicit retention and access policies to meet regulatory norms [1][6].

- Consent and transparency: users must be informed about what is collected and why; the literature shows limited documentation of consent procedures.

- Attack surface: command- or location-only schemes are easier to subvert (remote check-ins, location spoofing); hardware and biometric methods reduce some attack vectors but introduce others (tag cloning, image replay) unless anti-spoofing and secure enrollment are implemented.

E. Synthesis: trade-offs and design patterns

From the surveyed work, several recurring design patterns emerge:

- A spectrum of assurance versus cost: lightweight Telegram-only bots provide convenience at low cost, RFID strengthens identity binding but incurs hardware expenses, and full multi-modal systems (RFID + camera + liveness detection) yield the strongest assurance with higher deployment complexity and privacy obligations [1–4].

- Hybrid strategies appear promising: a primarily low-friction Telegram check-in regimen augmented by periodic, stronger checks (e.g., on-boarding biometric enrollment or occasional RFID spot-checks) can balance usability and security.

- Developer resources: Python libraries and step-by-step tutorials accelerate prototyping, but production systems need additional attention to secure token management, HTTPS, role-based admin access, and audit logging [5][8][10].

Additional works and tutorials highlight other approaches: personal automation scripts that send Telegram notifications after scraping attendance websites, face-recognition pilots that use LBPH with OpenCV integrated with Telegram notifications, and hybrid solutions combining sensors and messaging for real-time monitoring [5][6][7][10]. Collectively, these works demonstrate a spectrum of verification strength—from simple command- or location-based check-ins to multi-factor, hardware-assisted systems.

III. SUMMARY

In examining these prior works, several patterns, trade-offs, and gaps emerge.

A. Modes of Verification and Trust

- Lightweight approaches (command-driven or geofenced check-ins) are easy to deploy but vulnerable to spoofing and impersonation [4][5].
- RFID-based systems provide stronger identity binding but lack reliable location guarantees and can be subject to tag misuse [2][1].
- Multi-modal/hardware-augmented systems (RFID + camera + ultrasonic sensors) increase verification fidelity at the expense of cost, installation complexity, and privacy risk [1].
- There is a clear continuum of assurance: command/location → RFID → multi-factor (hardware + biometric). Each increase in trust incurs higher cost and operational complexity.

B. Role and Limitations of Telegram

- Telegram serves as a user-facing channel (commands, media, notifications), a notification bus, and a backend integration point. It reduces friction since many users already have Telegram installed and it avoids the need for custom mobile apps [4][8].
- However, Telegram is typically not the verifier itself; it acts as a conduit for events or user actions that require out-of-band verification to be fully trusted. Security-critical verification still depends on hardware or cryptographic measures external to Telegram.

C. Performance, Usability, and Privacy Concerns

- Experiments are frequently small-scale; scalability and concurrency under real-world loads are underexplored [2][4].
- Usability gaps—ambiguous commands, NLU errors, mis-entries—are rarely quantified; conversational systems report little or no evaluation metrics [3].
- Privacy and legal compliance receive limited attention; image capture and location logging raise consent, retention, and data protection issues that must be addressed before large-scale deployment [1][6].

D. Key Open Questions

- How to bind a Telegram user identity securely to a physical person without heavy hardware?
- What practical countermeasures minimize spoofing (location, image, RFID)?
- How do systems behave at scale, across multiple sites and under constrained network conditions?
- What are acceptable UX trade-offs between verification strength, intrusiveness, and user convenience?
- How can designs embed privacy-by-design and legal compliance from the outset?

TABLE I

DIFFERENT TECHNOLOGIES

Technology	Advantages	Disadvantages	Use Case
Biometric (Fingerprint/Iris)	High accuracy, secure	Expensive, hardware needed	Secure offices
RFID	Fast, low complexity	No location, tag loss	Indoor settings
GPS	Location tracking, low cost	Inaccuracies from environment	Field workers
Telegram API + Python	Real-time, accessible, low cost	Internet reliant, privacy risks	Remote teams, education

IV. EXAMPLE SYSTEM ARCHITECTURE (PYTHON + TELEGRAM) — IMPLEMENTATION NOTES

A practical architecture comprises: a Telegram bot (python-telegram-bot), a backend REST API (Flask or Fast API), a persistent database (SQLite or PostgreSQL), optional sensor gateways (ESP32), and an admin dashboard (web UI). Typical flow: When a user sends a /check-in command, the bot can optionally request a selfie or location, forward the collected data to the backend for validation against defined rules (e.g., geofence, RFID match, or facial check), and finally log the record while notifying the administrator through Telegram.

Implementation recommendations:

- Sensitive tokens should be kept in environment variables, communications with the backend should occur over HTTPS, and administrative panels should enforce role-based access restrictions.
- For image-based verification, perform server-side face matching and preserve privacy by storing only secure hashes, encrypted images, or transiently processing media.
- Implement audit logs, retention policies, and data access controls to support compliance with applicable data-protection requirements.

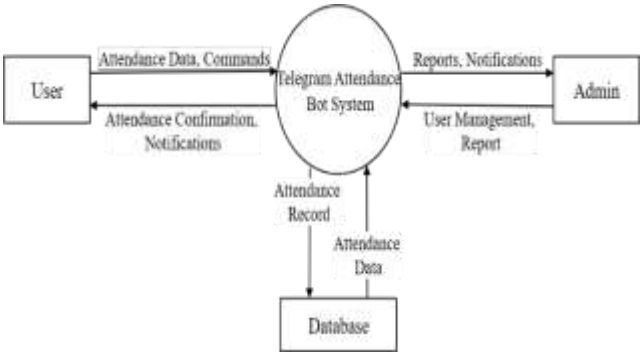

Figure 1 Telegram bot flow diagram

V. DISCUSSION: EVALUATION AND FUTURE WORK

Evaluation observations: Most studies report small pilots or proof-of-concept deployments with limited metrics (latency, basic accuracy). Usability, long-term adoption, and legal compliance are underreported.

Future research directions:

- Hybrid verification combining lightweight

(Telegram commands) with occasional strong checks (RFID, biometric enrollment) to balance convenience and security.

- AI-based liveness and anti-spoofing for selfies and image verification (e.g., liveness detection, adversarial-resistance models) [6].
- Scalability studies to benchmark throughput, latency, and database performance for hundreds to thousands of users.
- Usability studies to measure task completion rates, error rates, and user satisfaction.
- Privacy-first designs with encryption-at-rest, differential retention policies, and transparent consent flows.

VI. CONCLUSION

Python implementations leveraging Telegram bots provide an affordable and easily deployable solution for attendance tracking where hardware solutions are infeasible. Such Telegram-based systems prove efficient for rapid deployment and lightweight interaction but still face verification and scalability constraints. However, these solutions remain constrained in verification strength unless supplemented with hardware components or more sophisticated backend validation techniques. Addressing spoofing, scalability, user experience, and privacy/legal compliance will be essential for broader adoption.

ACKNOWLEDGMENTS

The authors would like to acknowledge the support and resources provided by the Department of Computer Science and Engineering, Shri Ramswaroop College of Engineering and Management. Special thanks are extended to the project supervisors for their constructive comments and continuous encouragement, which greatly improved the quality of this work.

REFERENCES

[1] A. Akbar, W. Fuadi, & N. Nunsina, "Design of attendance system for informatics engineering lecturers using RFID sensors based on IOT and telegram applications", International Journal of Engineering, Science and Information Technology, vol. 5, no. 2, p. 107-117, 2025. https://doi.org/10.52088/ijesty.v5i2.794

[2] B. Kurniawan, L. H. Fitriadi, M. F. Albaroky, & R. Astiani "Radio Frequency Identification and Telegram for Student Attendance", Int. J. Res. Appl. Tech., vol. 4, no. 1, pp. 139–144, Jul. 2024. https://ojs.unikom.ac.id/index.php/injuratech/article/view/14124

[3] S. Bhutada, C. Rakshit, G. Vaishnavi, & V. Varshitha, "Chatra paryesana (student enquiry system)", International Journal for Research in Applied Science and Engineering Technology, vol. 11, no. 6, p. 670-674, 2023. https://doi.org/10.22214/ijraset.2023.53722

[4] B. Nii, O. William, & K. Andoh, "Attendancebot location-based attendance tracker for workers", International Journal of Computer Applications, vol. 178, no. 51, p. 12-18, 2019. https://doi.org/10.5120/ijca2019919414

[5] The Code Life. (2020, August 7). Automating my attendance with Python.TheCodeLife. https://thecodelife.science.blog/2020/08/07/automating-my-attendance-with-python/

[6] Amin, Qois & Budianto, Aris & Budiyanto, Cucuk. (2022). Development of a Facial Recognition-based Attendance System using Binary Patterns Histograms Method and Telegram Bot Notification. Journal of Informatics and Vocational Education. 5. https://doi.org/10.20961/joive.v5i2.65789.

[7] Rohit Raj (2024, May 16). Face attendance system in Python. Medium. https://rohitraj-iit.medium.com/face-attendance-system-in-python-069180b64b55

[8] AsutoshPati Telegram Attendance Bot [Repository]. GitHub. https://github.com/AsutoshPati/TelegramAttendanceBot

[9] Aldabagh, M. (2025). A Review of Students Attendance Management Systems. Eurasian Journal of Science and Engineering, 10(3),136-145. https://doi.org/10.23918/eajse.v10i3p13

[10] Alagasan, Kanageswari & Alkawaz, Mohammed & Hajamydeen, Asif Iqbal & Abdulrazaq Alshekhly, Mohammed. (2021). A Review Paper on Advanced Attendance and Monitoring Systems. 195-200. https://doi.org/10.1109/ICSGRC53186.2021.9515249.