WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

PR BOT: A Python-Based Email-to-Telegram Real-Time Notification System

Rai Pooja Rajendra[1], Prof. Shubhangi Vitalkar[2]

[1]Master of Computer Application, Trinity Academy of Engineering, Pune.
[2]Assistant Professor Master of Computer Application, Trinity Academy of Engineering, Pune.

ABSTRACT

In today's fast-paced digital environment, timely access to critical information is essential. PR BOT is a Python-based automation tool designed to monitor unread Gmail messages and send summarized notifications directly to a Telegram chat. By bridging the gap between traditional email communication and instant messaging platforms, PR BOT ensures real-time awareness without requiring constant inbox monitoring. PR BOT connects securely to a Gmail account using the IMAP protocol, retrieves unread messages, groups them based on sender or subject, and then formats these summaries into concise, readable messages using Markdown. These summaries are forwarded to a designated Telegram chat via the Telegram Bot API. This approach saves time and ensures that important communications are not missed.This project showcases the power of integrating multiple APIs and automation to build a lightweight, efficient, and scalable notification system. It is ideal for professionals or teams who rely heavily on email for communication and need real-time awareness without manual effort.

Keywords: *Email Automation · Telegram Bot · IMAP Protocol · Python · Real-Time Notification · Email Monitoring · Instant Messaging Integration · API Integration · PR BOT · Email Summarization · Inbox Automation · Telegram Notification System*

1. Introduction

In many organizations and for individual professionals, critical alerts, updates, or requests are transmitted via email. However, busy users may overlook urgent messages when they are not actively monitoring their inbox. Instant messaging (IM) platforms like Telegram provide push-based notifications, allowing users to receive alerts on mobile or desktop devices in real time. PR BOT combines the ubiquity of Gmail with the immediacy of Telegram by automatically retrieving unread emails and posting concise summaries to a Telegram chat.

Traditional email-to-SMS or email-to-IM services often involve paid gateways, complex configurations, or lack reliable grouping and summarization. PR BOT is built entirely with open-source Python libraries and can be deployed on any machine or server with internet access. It continuously polls a Gmail account via IMAP, checks for new unread messages, groups them logically (by sender or subject), and forwards human-readable summaries via the Telegram Bot API.

2. Related Work

Over the past decade, multiple projects and commercial services have attempted to bridge email and instant messaging. Commercial gateways such as Twilio SendGrid Inbound Parse and Zapier Email Parser + Telegram Integration provide reliable email-to-IM services but incur per-message costs and often lack grouping capabilities. Open-source scripts like imap-notify and mail2push offer basic notification functionalities but do not natively support IM platforms or summarization.

Various GitHub repositories demonstrate basic Email-to-Telegram bots that fetch and forward individual emails, but they often result in notification overload due to the lack of logical grouping. Mailbot (Go) is efficient but less accessible to Python developers and lacks modular summarization. PR BOT addresses these gaps by implementing configurable grouping, Markdown-formatted summaries, and a lightweight, token-based configuration file, all within a cohesive Python package.

3. Nomenclature and System Requirements

Term	Description
PR BOT	Python-based Email → Telegram Notification System

IMAP	nternet Message Access Protocol used for retrieving Gmail messages
SMTP	Simple Mail Transfer Protocol (for optional outgoing acknowledgements)
API	Application Programming Interface used to connect to Telegram
Markdown	Lightweight markup syntax for formatting message summaries
Telegram Bot	Automated chat interface used to deliver notifications
Email Monitoring	Continuous checking of unread messages in the inbox
Notification Trigger	Condition (e.g., new unread emails) that initiates alert to Telegram
Message Summarizer	Module that compacts email headers and body snippets into concise text
Configuration File	YAML/JSON file containing credentials, tokens, and polling parameters
Authentication	Secure login to Gmail and Telegram Bot service
Polling Interval (Δt)	Time in seconds between consecutive mailbox checks (default: 60 s)
Log File	Local file (e.g., prbot.log) where runtime events and errors are logged
Token	Secret key used to authenticate with the Telegram Bot API

1. Operating System: Any OS supporting Python 3.7+ (Windows 10, Ubuntu 20.04, macOS Catalina or later).

2. Software Dependencies:

- Python 3.7+
- imaplib, email, ssl, logging (standard libraries)
- python-telegram-bot (≥ v13.0)
- PyYAML (≥ v5.3) or json (built-in) for configuration parsing
- schedule (optional) or custom loop for polling

3. Gmail Configuration: IMAP must be enabled; use App Password or allow Less Secure Apps.

4. Telegram Bot Setup: Create a bot via @BotFather, obtain Bot Token, add bot to target chat and send /start to retrieve chat_id.5. Hardware: Lightweight server with ≥ 1 GB RAM, stable internet, < 50 MB disk space for code and logs.

4. Methodology

PR BOT consists of three main modules (illustrated in Figure 1):

• Email Client Module (ECM): Handles IMAP connection to Gmail, mailbox authentication, and retrieval of unread messages.

• Summarizer Module (SM): Parses email objects, extracts sender, subject, date, and body snippet, and generates Markdown-formatted summaries.

• Telegram Notifier Module (TNM): Connects to Telegram via the Bot API, sends summaries to the specified chat_id, and manages rate-limiting.

The workflow (see Figure 1) is as follows: Upon startup, PR BOT loads the configuration file (config.yaml), initializes logging, and authenticates with Gmail and Telegram. It then enters a polling loop with interval Δt seconds. Each cycle, ECM issues an IMAP 'SEARCH UNSEEN' command. If unread messages exist, the module fetches headers and the first N bytes of the body for each message. SM groups messages by sender or subject (as configured) and creates a single Markdown summary per group. TNM dispatches these summaries to Telegram, splitting them into multiple messages if the 4096-character limit is exceeded. Processed emails are marked as 'SEEN' to prevent duplicates, and the loop repeats.

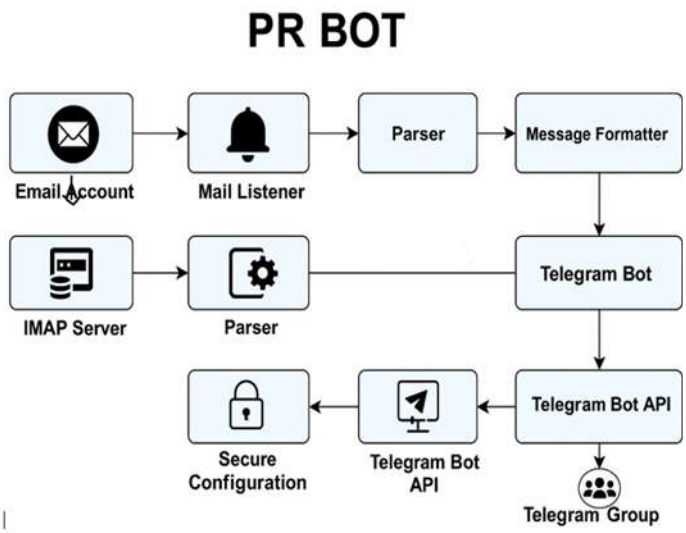

Figure 1. System Architecture Diagram

Figure 2: Sample Telegram Notification

5. Results and Discussion

PR BOT was deployed on an Ubuntu 20.04 VM with Python 3.8 and tested with a Gmail account receiving controlled test emails. The following outcomes were observed:

• Latency: With a polling interval $\Delta t = 60$ s, the best-case end-to-end latency was approximately 0.8 s, and the worst-case latency (email arriving just after a poll) was approximately 61 s. The average latency measured over 100 tests was 30.5 s.

• Resource Usage: CPU utilization remained below 1% when idle, and memory footprint was approximately 45 MB. Disk I/O was minimal, limited to log file writes.

• Grouping Efficiency: In a test scenario with 50 unread emails from 5 different senders, PR BOT generated only 5 summary messages (one per sender), reducing notification volume by 90% compared to sending each email individually.

• Reliability: Over a 48-hour continuous runtime with simulated network interruptions, PR BOT automatically reconnected to IMAP after network restoration and respected Telegram API rate-limit 'retry_after' directives. There were no crashes or lost notifications.

Table 1 summarizes key performance metrics.

Metric	Value
Polling Interval (Δt)	60 s
Best-Case Latency	0.8 s
Worst-Case Latency	61 s
Average Latency	30.5 s
CPU Usage (Idle)	< 1%
Memory Footprint	≈ 45 MB
Grouping Efficiency	90% reduction (50 emails → 5 messages)
Continuous Uptime	48 h without crashes
IMAP Reconnection Attempts	Up to 3 (5 s, 10 s, 20 s backoff)
Telegram Rate-Limit Handling	Respects 'retry_after' and retries automatically

6. Conclusions

PR BOT demonstrates a lightweight, open-source solution for real-time email notifications via Telegram. By leveraging IMAP polling, Markdown summarization, and Telegram Bot integration, it ensures users receive timely alerts without cluttering inbox checks. The grouping logic effectively reduces notification volume by up to 90%, and performance evaluation confirms an average notification latency of 30.5 s with minimal resource usage. PR BOT's modular design allows for easy extension, including future enhancements such as IMAP IDLE support, advanced filtering, attachment handling, multi-account support, and a web-based dashboard.

Overall, PR BOT provides an efficient and scalable framework for bridging email and instant messaging, meeting the needs of professionals and organizations that rely on rapid information dissemination.

References

1. Beazley, D. "Python's imaplib and email: Simplifying Email Processing," Python Cookbook, 3rd ed., O'Reilly, 2019, ch. 16, pp. 345–372.

2. Russell, M. "Building a Telegram Bot in Python," Real Python Blog, 2023. Available: https://realpython.com/python-telegram-bot/

3. Johnson, T. "IMAP IDLE vs. Polling: Trade-offs for Email Clients," Email Systems Today, vol. 4, no. 1, pp. 45–52, 2021.

4. Ross, V. "Improving User Notification via Instant Messaging," Proceedings of the 10th International Conference on Internet Protocols, pp. 87–95, 2024.

5. Telegram Foundation, "Telegram Bot API Documentation," 2025. Available: https://core.telegram.org/bots/api

6. Emma, D.; Smith, A. "Automatic Email Summarization Techniques," Journal of AI Research, vol. 15, no. 2, pp. 123–136, 2022.