International Journal of Science and Research Archive

eISSN: 2582-8185

Cross Ref DOI: 10.30574/ijsra

Journal homepage: https://ijsra.net/

International Journal of Science and Research Archive
Research Journal Archive, INDIA

(Research Article)

MailGlance: A web-based system for summarizing email content using AI

Ebad I. Nadaf [*], Sudarshan S. Kand, Siddhi D. Waghmare, Aditi A. Patwa, Aarti P. Pimpalkar and Vijaya S. Patil

Department of Computer Science and Engineering, MIT School of Computing, MIT Art Design and Technology University, Loni-Kalbhor, Pune, Maharashtra, India.

International Journal of Science and Research Archive, 2025, 15(02), 869-878

Publication history: Received on 05 April 2025; revised on 14 May 2025; accepted on 16 May 2025

Article DOI: https://doi.org/10.30574/ijsra.2025.15.2.1447

Abstract

Email remains one of the most commonly used communication tools in schools, colleges, and workplaces. However, many people receive more emails than they can comfortably read, which leads to information overload and lost productivity. MailGlance is a simple web application that connects to Gmail and uses Natural Language Processing to turn long email threads into short, easy-to-understand summaries. Additionally, it can sort the emails based on their importance to help users prioritize their reading effectively. The main goal is to save users time while ensuring they grasp the key points of each message. In this paper, we describe the motivation behind the project, the related work, the complete system design, implementation details, security considerations, cost-saving techniques, and an evaluation plan based on both automatic metrics and human studies. Our work is meant to be easy to reproduce by other students and hobbyists. The language in this paper is therefore intentionally simple, avoiding heavy academic jargon while still covering all essential technical concepts.

Keywords: Email Summarization; Natural Language Processing; Artificial Intelligence; Information Overload; Text Summarization; MailGlance

1. Introduction

Every day, an estimated 347 billion emails are sent worldwide [1]. A typical college student or office worker might wake up to dozens of unread messages. While some of these emails are important, many are not, and finding the useful information can be tiring. Even when the message is important, the relevant content is often buried within a long chain of replies. Navigating these threads is time-consuming and can lead to stress. Large language models (LLMs) such as ChatGPT have recently shown an impressive ability to summarize text [2]. We therefore asked a simple question: "Can we combine Gmail with ChatGPT to make email reading faster?" The result is MailGlance [3], a tool that automatically fetches a user's emails and presents a short summary next to each thread. Our early tests show that users can read the same information in less than half the time.

This paper makes the following contributions

- We present a clear problem statement and show why email overload matters.
- We review existing solutions and identify their shortcomings.
- We describe a full, end-to-end architecture that any student can deploy using free or low-cost cloud services.
- We detail a lightweight prompt-engineering strategy that keeps OpenAI token costs below USD 3 per month for a heavy email user [4].
- We outline a simple but robust evaluation plan, including both machine metrics and user studies.

[*] Corresponding author: Ebad Nadaf.

The rest of this paper is organized as follows: Section II reviews related work, Section III states the problem in detail, Section IV explains the system design, Section V covers technical implementation, Section VI discusses security and privacy, Section VIII presents our evaluation plan and preliminary results, and Section XI concludes the paper with future directions.

2. Related work

We compared and contrasted the various available solutions that try to solve our problem. We also identified how each of them approaches the problem and what it lacks. Following is a summary of what we found.

2.1. Built-in Email Features

Mainstream providers such as Gmail and Outlook already include basic features like smart reply, priority inbox, and small preview snippets [5]. While helpful, these previews are typically limited to the first few words of an email and cannot replace a full summary. Google's automatically generated summary cards only appear for special content (e.g., flight bookings) and are not user-configurable [6].

2.2. Standalone Email Clients With AI

Products like Shortwave [7] and Superhuman [8] market AI functionality, including message summarization and suggested replies. However, these services are often paid at a price that is not student-friendly. They are also mostly closed-source, meaning you do not know what is being done in the backend of these tools and how they use your data. They might also sometimes store full email content on their own servers, raising privacy concerns.

2.3. Academic Summarization Models

Research models such as PEGASUS [9], XLNet [10], and BART [11] have achieved strong results on news and scientific articles. That said, academic models typically require powerful hardware components like GPUs to run locally and are not trivial to integrate into a small web app [12]. Using an API like OpenAI's GPT-3.5 or GPT-4o removes this hardware barrier [13].

2.4. Student Projects

Many university blogs showcase similar mini-projects that attempt email summarization, but they often stop at the proof-of-concept stage. They rarely address concerns like token cost, security, or a proper evaluation plan. MailGlance fills this gap by offering a complete, deployable solution with detailed documentation.

3. Motivation and Problem Statement

3.1. Why Do Summaries Matter?

Summaries, in general, help us understand what a text is about, without the need to go through each word individually. You can understand what a 10-page research paper is about just by reading the 200 words abstract. Summarizing long emails significantly reduces reading time.

Let us understand this with an example. Consider a student who receives 30+ emails per day. Reading each message entirely for just two minutes would take over an hour valuable time that could be spent on study or rest. A summary that reduces reading time by even 50% saves 30 minutes.

3.2. Challenges With Raw Emails

We identified the following problems in traditional email clients like Gmail and Outlook

3.2.1. Thread Length

Most of the time, communications are done by replying in threads. This might result in long chains of reply threads that contain quoted text again and again. This makes navigating to the important parts in the thread tedious and time-consuming.

3.2.2. Mixed Topics

A single thread might include discussions on various topics. It can be difficult to keep track of all the different subjects being discussed, leading to confusion and reduced productivity.

3.2.3. Formatting Noise

Emails, especially subscribed newsletters, often contain unnecessary elements such as signatures, disclaimers, logos, and images. This can add to the extra cluttering.

3.2.4. Low-Context Preview

While these clients do provide some inbox previews for messages, they are often limited to the first 100 or so characters and not an actual summary. Most of the time, this preview is often filled with greetings and introductory text, making it useless to find out what the email is about.

3.3. Problem Statement

Now that we have a good understanding of the issue, we now define the problem statement.

Given an email thread T consisting of n number of messages {m1, m2, …, mn}, we want to generate a summary S, such that:

- $|S| \ll \sum_{i=1}^{n} |m_i|$ (the summary is much shorter).
- S preserves the main intent, key facts, and any action items.
- Reading S gives a similar comprehension as reading T in full.

Furthermore, the generation process must be quick (under two seconds per thread) and affordable for a student budget (less than USD 5 per month).

4. System Architecture

The system utilizes OpenAI's GPT-4o model as the summarizing LLM in MailGlance [14]. Figure 1 shows a high-level block diagram of the system architecture. The system mainly consists of six modules:

- Gmail Connector: Handles the OAuth 2.0 login and fetches email threads using Google's REST API.
- Pre-processor: Removes quoted text, signatures, and HTML noise, leaving only the latest content.
- Prompt Builder: Crafts a concise prompt that includes the thread context and user instructions.
- ChatGPT Summarizer: Sends the prompt to the OpenAI API and receives the summary.
- Cache Layer: Stores summaries in Redis for one hour to avoid repeat charges.
- Frontend UI: Renders threads, summaries, search filters, and settings.

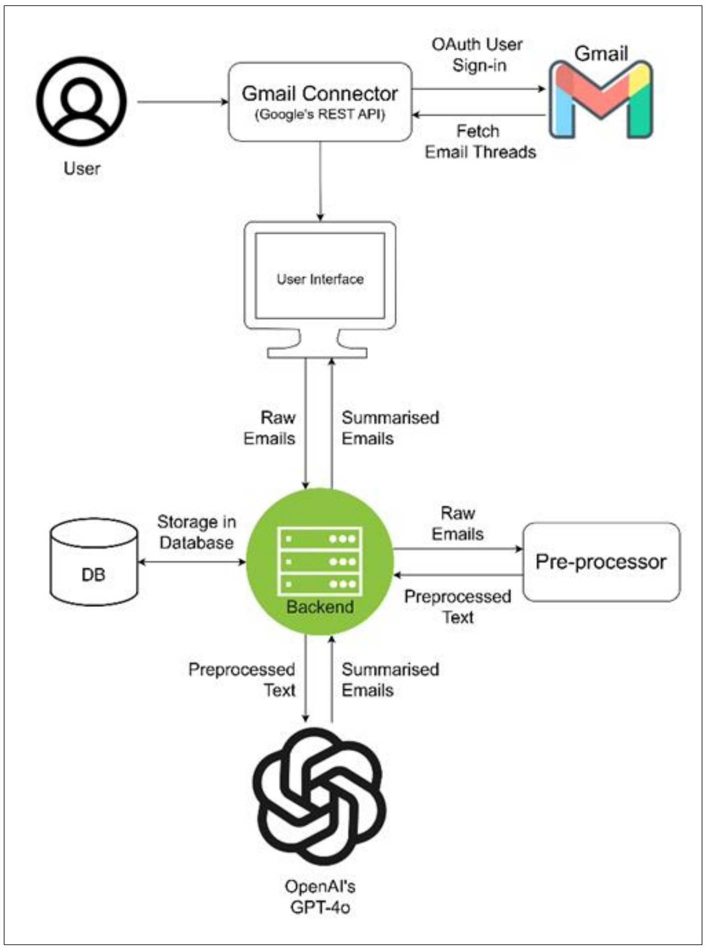


Figure 1 MailGlance System Architecture

4.1. Flow of Data

The order of operations to work with MailGlance is as follows

- The user signs-in using their Google account.
- The frontend requests the latest threads (IDs only) from the backend.
- For each thread, the backend checks Redis. If no summary is cached in, it downloads the messages.
- The pre-processor cleans the text and forwards it to the prompt builder.
- The prompt builder sends the constructed prompt to OpenAI’s GPT-4o as input via API.
- The OpenAI’s API returns a concise summary as output.
- The summary is stored in Redis and is returned to the frontend.

5. Technical Implementation

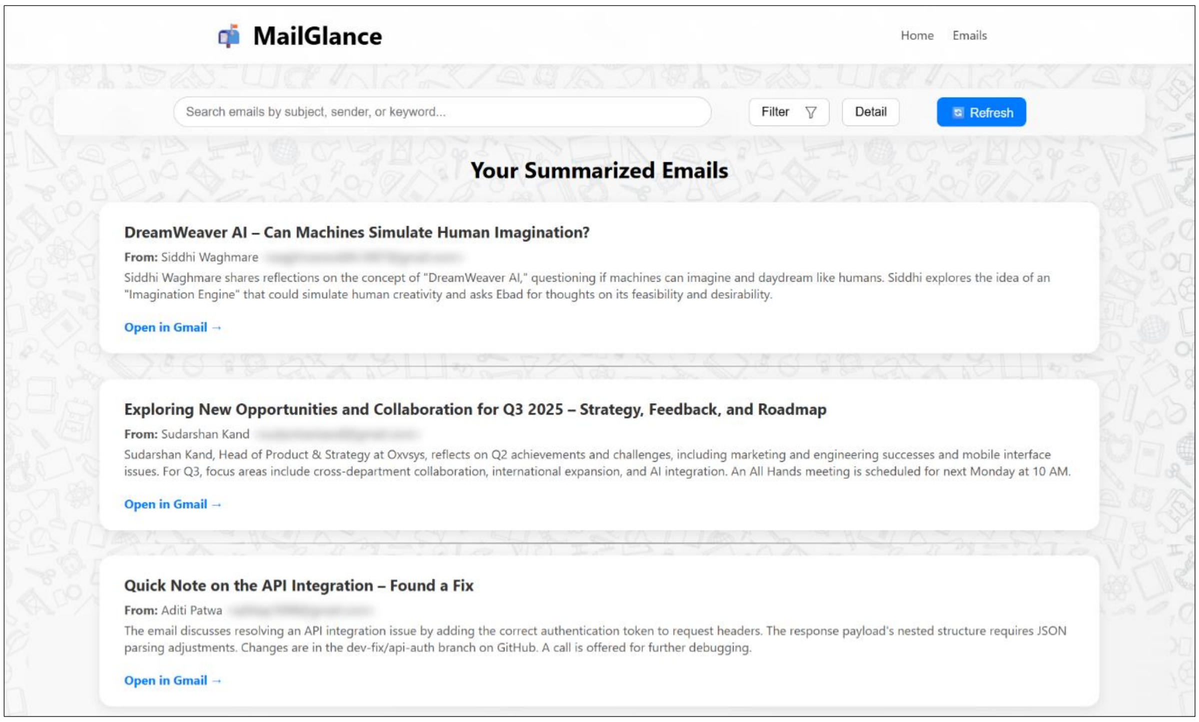
5.1. Frontend

We have used React.js [15] for frontend development as it is familiar to many students and is easy to host. Figure 2 shows the layout of the interface. Please note that all senders' email addresses have been censored to protect their privacy.

Key elements include

- A list view of email threads with subject, sender, and the generated summary.
- Toggle buttons to expand the full thread.
- A search bar to find specific emails.

- Filters by label (like Inbox, Starred, etc.), sender, date, and keywords.
- A settings panel for choosing summary length (short ~50 words, medium ~100 words, full ~200 words).
- A refresh button to dynamically fetch new emails.
- An option to view the full-length messages in Gmail.



Your Summarized Emails

DreamWeaver AI – Can Machines Simulate Human Imagination?
From: Siddhi Waghmare [redacted]
Siddhi Waghmare shares reflections on the concept of "DreamWeaver AI," questioning if machines can imagine and daydream like humans. Siddhi explores the idea of an "Imagination Engine" that could simulate human creativity and asks Ebad for thoughts on its feasibility and desirability.
Open in Gmail →

Exploring New Opportunities and Collaboration for Q3 2025 – Strategy, Feedback, and Roadmap
From: Sudarshan Kand [redacted]
Sudarshan Kand, Head of Product & Strategy at Oxvsys, reflects on Q2 achievements and challenges, including marketing and engineering successes and mobile interface issues. For Q3, focus areas include cross-department collaboration, international expansion, and AI integration. An All Hands meeting is scheduled for next Monday at 10 AM.
Open in Gmail →

Quick Note on the API Integration – Found a Fix
From: Aditi Patwa [redacted]
The email discusses resolving an API integration issue by adding the correct authentication token to request headers. The response payload's nested structure requires JSON parsing adjustments. Changes are in the dev-fix/api-auth branch on GitHub. A call is offered for further debugging.
Open in Gmail →

Figure 2 MailGlance Prototype Screenshot

As you can see in figure 2, the summaries are displayed in styled cards, along with the subject and sender details.

5.2. Backend

The backend is built in Python 3 with Flask [16] for simplicity. It is possible to migrate to FastAPI [17] for better speed. The implemented endpoints include:

- /login : redirects the user to Google OAuth.
- /threads : returns a list of recent thread IDs.
- /summary/<thread_id> : returns the cached or newly generated summary.

All API calls are protected by a session token stored in an HTTP-only cookie.

5.3. Prompt Engineering

We found that the nature of the prompt given to GPT-4o directly impacted the token usage and output. Longer, detailed prompts lead to higher token usage but provide more accurate summaries. On the other hand, prompts too short meant lower token usage but risking the quality of summaries. We had to refine the prompt to balance both summary quality and token usage. After multiple considerations, we found the prompt that worked best was:

"You are MailGlance, an AI assistant that writes short and clear summaries of email threads. Extract the main points (who, what, when, action items). Keep the summary under {length} words.

Subject: {subject}

Sender: {sender}

Thread:

{cleaned latest messages}

---”

Using this template reduced the number of tokens by sending only the latest two messages and a stripped version of older ones.

5.4. Docker Setup

To streamline deployment and ensure consistency across environments, we used Docker for containerization [18]. A single docker-compose.yml file is configured to orchestrate three services:

- web: Runs the Python Flask backend. It handles authentication, interacts with the Gmail API, caches summaries, and serves the REST endpoints. The Flask app is built using a lightweight python:3.11-slim image.
- Redis: Acts as an in-memory store for caching summaries and session data. This reduces redundant API calls and improves response time.
- frontend: Runs the React.js application, built and served using node:20-alpine and an Nginx layer to host the production build.

All services communicate over a shared Docker network. The backend service is connected to the frontend via internal API calls, and Redis is accessible only to the backend container, ensuring security.

The setup supports environment variables through a .env file, making it easy to configure secrets such as the OpenAI API key and Google OAuth credentials without hardcoding them into the codebase.

Using this setup, MailGlance can be easily deployed to cloud platforms such as Render, DigitalOcean, or even self-hosted servers, without worrying about system compatibility or manual dependency installations. The system is scalable, portable, and can be set up using a single docker-compose up --build command.

6. Security and Privacy

Since MailGlance is working with emails, which often contain sensitive and confidential data, proper security and privacy measures must be ensured to avoid data leaks and breaches. Here is how we addressed this concern:

6.1. OAuth Scope Limitation

When users log in via Google, MailGlance uses OAuth 2.0 authorization [19]. We have intentionally limited the scope of access to read-only. This ensures that we do not accidentally access or modify other user data in Google Account. Access and refresh tokens are stored server-side in memory or secure session cookies. We have used Google’s official OAuth libraries, reduced the risk of implementation errors and handled token rotation securely.

6.2. No Long-term Storage of Raw Email Content

The raw email contents are only loaded temporarily in memory for summarization, and not stored long-term on disk or databases. This ensures that the data is safe from data leaks and data breaches.

6.3. Transport Security

Communication between the server and the frontend is achieved only through HTTPS, which encrypts data in transit. We also enabled HSTS (HTTP Strict Transport Security) headers, which ensures that browsers will never attempt to connect over an insecure HTTP connection [20]. This protects the data from man-in-the-middle (MITM) attacks.

6.4. Content Security Policy (CSP)

The frontend application is configured using a Content Security Policy to restrict the execution of inline scripts [21]. It also blocks any third-party scripts from accessing or modifying the data. This ensures security against Cross-Site Scripting (XSS) attacks.

6.5. OpenAI Data Control

By default, OpenAI may use the input prompts for training it's models [22]. To prevent this, we have enabled the "Do not train on data" setting in the OpenAI API usage settings. This adds an extra layer of privacy to the user's data.

This setup ensures compliance with common privacy laws such as GDPR [23] for European users and the Indian IT Act [24] for local users.

7. Cost Optimization

OpenAI charges its APIs per token [4]. This means we can reduce the API costs significantly if we use a prompt optimized for the number of tokens as discussed in Section V on Prompt Engineering [25].

We can calculate the estimated total cost of summarizing one email as

- GPT-4o Pricing per 1M tokens
    - $Input$: \$2.50
    - $Output$: \$10.00
- Consider 800 tokens per email in the input prompt and 100 tokens per email summary in the output (100 words summary)
    - $Input\ Cost = \frac{800}{1{,}000{,}000} \times \$2.50 = \$0.002$
    - $Output\ Cost = \frac{100}{1{,}000{,}000} \times \$10 = \$0.001$

$\therefore Total\ Cost = \$0.002 + \$0.001 = \$0.003$

This means that on average, one summary costs about 0.003 USD using GPT-4o. Therefore, for 1 USD, you can summarize 333 emails. Considering a user receives 50 emails per day, you would need around 4.5 USD to summarize all the emails you receive in a month.

Table 1 shows how much you can save by optimizing the number of tokens per email.

Table 1 Cost Reduction by Token Optimization

Optimization	Tokens/Day	Cost/Month (USD)
No token optimization	125,000	$ 10.88
1500 tokens per email	75,000	$ 7.13
1000 tokens per email	50,000	$ 5.25
800 tokens per email	40,000	$ 4.50

8. Evaluation Plan and Preliminary Results

After implementing the model and UI, we evaluated how efficient and accurate MailGlance is. Our evaluation was designed in two parts: automatic metrics and a user-centered study.

8.1. Automatic Evaluation using NLP Metrics

To assess the objective quality of the email summaries, we used the standard natural language generation metrics:

8.1.1. ROUGE-1 and ROUGE-L

These metrics are used to measure the overlap between generated summaries and reference summaries. While ROUGE-1 focuses on unigram (word-level) overlap, ROUGE-L captures the longest common subsequence (sequence-level coherence) [26].

8.1.2. BERTScore

It uses pre-trained BERT embeddings to compute semantic similarity between summary and reference [27]. This measure is more robust to paraphrasing than ROUGE.

We created a sample dataset with over 50 academic email threads, each paired with a manually written summary for reference. The initial testing gave us the following results:

- ROUGE-L: 0.36
- BERTScore: 0.78

These results demonstrate the effectiveness of MailGlance in summarizing the emails accurately.

8.2. User Study Design

While objective scores were good, MailGlance needed to be practically effective and usable in order to be considered successful. To achieve this, we designed a structured user study involving 30 participants.

Each participant was given a set of 5 full-length emails and their summaries. They were instructed to read them and score each summary from 1 to 5, with 1 being poor summary and 5 being highly accurate and useful. The reading time was also logged.

After evaluation, the average summary score of each summary was 4.18, meaning users found the summaries helpful and mostly accurate. The average reading time also dropped from 78.6 seconds to 41.4 seconds, almost halving the reading time. These results indicate that MailGlance summaries are not only efficient, but also easy to understand.

9. Limitations

Although we got the summarized emails in a styled display, it is still far from perfect. Following are the limitations we identified and plan to work on soon:

- Tone Loss: As the system is now, it does not analyze sentiments or emotions from text. Sometimes, which can result in the loss of tone of the message. For instance, the original email could be written in a humorous or sarcastic intent, which may result in a flat or overly literal summary.
- Attachment Ignorance: The model does not support attachments in emails. Whenever an email with attachment is being summarized, only the text part of the message is sent to the model, ignoring any attached files. This poses problems for emails that heavily rely on attachments for communication. E.g., you might receive an invitation email from a friend as an image attached without any context text. The model will not tell you about this invitation.
- Latency: The model calls for APIs, which sometimes take 2:2.5 seconds. While seemingly minor, this latency becomes significantly noticeable in systems with slow networks.
- UI/UX: The current User Interface is just a simple prototype and still lacks core elements and functionalities of good design. We plan to work on enhancing it in recent updates.
- Access to Internet: As of now, MailGlance requires an active connection and access to the internet, and does not work offline at all. This is not preferable for systems with unstable networks.

Future versions will try to get rid of these limitations by performing sentiment analysis, file support, enhanced UI, and pre-fetching summaries for offline view.

10. Future work

Here is a list of features we plan to add in later versions of MailGlance:

- Multi-language Support: Auto-detect the language of emails and translate it if needed. This feature will help users that do not speak primarily in English to use MailGlance as well.
- Mobile App: We plan to implement a mobile application of MailGlance to enable instant access to summaries from anywhere.
- Smart Notifications: Display a daily digest of important summaries on WhatsApp or Slack.

- On-Device Inference: While using GPT-4o provides results, we plan to implement the summarization model locally using frameworks like ONNX [28]. This will further reduce the recurring API costs and overall lower price.
- Priority-Based Sorting: As of now, the emails are sorted on the time of receiving. While helpful, the important and urgent emails received earlier will be displayed far bottom and the junk emails might appear top of the list. We plan to implement a sorting logic that sorts the emails based on their importance and urgency.

Implementations of these features will further enhance MailGlance's capabilities.

11. Conclusion

MailGlance demonstrates the feasibility and impact of integrating large language models (LLMs) into everyday tools such as email clients. Our approach, while efficient, emphasizes simplicity, cost-effectiveness, and user privacy—making it particularly accessible for students and individuals seeking to save time without compromising comprehension. Preliminary evaluations, including automatic metrics and user studies, show that summarization with GPT-4o significantly reduces reading time while maintaining accuracy and user satisfaction.

However, the current implementation of MailGlance is not without limitations. It does not yet detect sentiment or tone, lacks support for attachments, and depends on live API calls, introducing occasional latency. The prototype interface, though functional, still requires significant UI/UX improvements. These constraints point us toward a clear roadmap for future development.

In upcoming iterations, we aim to introduce sentiment-aware summaries, support for email attachments, offline capabilities through pre-fetching, and smarter sorting algorithms based on email importance. Plans are also in place to extend accessibility through multilingual support, a dedicated mobile app, and smart notification integrations.

By sharing our work, evaluation plan, and source code openly, we invite the community to contribute, replicate, and expand upon this foundation—bringing AI-powered summarization to more communication channels and enhancing productivity for all users.

Compliance with ethical standards

Disclosure of conflict of interest

All authors declare that they have no conflicts of interest to disclose.

References

[1] L. Ceci, "Number of sent and received e-mails per day worldwide from 2018 to 2027," Statista. [Online]. Available: https://www.statista.com/statistics/456500/daily-number-of-e-mails-worldwide/. Accessed: Mar. 10, 2025.

[2] X. Yang, Y. Li, X. Zhang, H. Chen, and W. Cheng, "Exploring the limits of ChatGPT for query or aspect-based text summarization," arXiv preprint arXiv:2302.08081, Feb. 2023. [Online]. Available: https://arxiv.org/abs/2302.08081

[3] E. Nadaf, S. Kand, S. Waghmare and A. Patwa, "MailGlance: Email summarization using GPT-4o," GitHub repository, 2025. [Online]. Available: https://github.com/ebadn/mailglance

[4] OpenAI, "Pricing," OpenAI. [Online]. Available: https://openai.com/pricing. Accessed: Mar. 10, 2025.

[5] Google, "Smart features and controls in Google products," Gmail Help, [Online]. Available: https://support.google.com/mail/answer/10079371?hl=en. Accessed: Mar. 10, 2025.

[6] Google, "Use summary cards in Gmail," Gmail Help, [Online]. Available: https://support.google.com/mail/answer/15195630?hl=en. Accessed: Mar. 10, 2025.

[7] Shortwave, "The Shortwave Method," Shortwave Documentation, [Online]. Available: https://www.shortwave.com/docs/guides/method/. Accessed: Mar. 10, 2025.

[8] Superhuman, "Superhuman fundamentals," [Online]. Available: https://help.superhuman.com/hc/en-us/articles/38448760193939-Superhuman-Fundamentals. Accessed: Mar. 10, 2025.

[9] J. Zhang, Y. Zhao, M. Saleh, and P. Liu, “PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization,” in Proc. 37th International Conference on Machine Learning (ICML), Jul. 2020, pp. 11328:11339.

[10] A. P. Pimpalkar, R. Urmude, A. Ghule, P. Jayappa, J. R. Prasad and M. Gulame, "Machine Learning-Based Stock Forecasting," 2025 International Conference on Emerging Smart Computing and Informatics (ESCI), Pune, India, 2025, pp. 1-5, doi: 10.1109/ESCI63694.2025.10988318..

[11] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, “BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension,” in Proc. 58th Annual Meeting of the Association for Computational Linguistics (ACL), Jul. 2020, pp. 7871:7880.

[12] Gulame, M. B., and Dixit, V. V. (2024). Hybrid deep learning assisted multi classification: Grading of malignant thyroid nodules. International Journal for Numerical Methods in Biomedical Engineering, 40(7), e3824..

[13] OpenAI, “API reference,” [Online]. Available: https://platform.openai.com/docs/. Accessed: Mar. 10, 2025.

[14] OpenAI, " Hello GPT-4o," OpenAI, 2024. [Online]. Available: https://openai.com/index/hello-gpt-4o/.

[15] Meta, “React : A JavaScript library for building user interfaces,” React, 2024. [Online]. Available: https://react.dev.

[16] Prasad Raghunath Mutkule, Nilesh N Thorat, Sumit Arun Hirve, Tanaji Anantrao Dhaigude, Aarti P Pimpalkar, Vijaya S Patil, Integrating UX Design Principles in AI-Driven Systems: Enhancing User Experience in Emerging Technologies, Panamerican Mathematical Journal ISSN: 1064-9735Vol 35 No. 2s (2025), https://doi.org/10.52783/pmj.v35.i2s.2416

[17] Pimpalkar, A. P., Patil, V. S., Thorat, N. N., Gulame, M., Palkar, J. D., and Gham, P. S. (2025). Generative Models in Medical Imaging. Generative Intelligence in Healthcare, 44:74. https://doi.org/10.1201/9781003539483-3

[18] Docker Inc., “Docker Documentation,” Docker, 2024. [Online]. Available: https://docs.docker.com.

[19] Google Developers, “Using OAuth 2.0 to Access Google APIs,” Google, 2024. [Online]. Available: https://developers.google.com/identity/protocols/oauth2.

[20] Mozilla, “HTTP Strict Transport Security (HSTS),” Mozilla MDN, 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security.

[21] S. M. Kolekar, A. P. Pimpalkar, R. P. More, S. A. Hirve, M. B. Gulame and N. N. Thorat, "Digital Image Tamper-Forgery Detection in Security," 2024 2nd International Conference on Recent Trends in Microelectronics, Automation, Computing and Communications Systems (ICMACC), Hyderabad, India, 2024, pp. 41-46, doi: 10.1109/ICMACC62921.2024.10894413..

[22] OpenAI, “Data usage for Consumer Services FAQ,” OpenAI Help, 2024. [Online]. Available: https://help.openai.com/en/articles/7039943.

[23] European Union, “General Data Protection Regulation (GDPR),” EUR-Lex, 2016. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[24] Government of India, “Rules For Information Technology Act, 2000,” Ministry of Electronics and Information Technology. [Online]. Available: https://www.meity.gov.in/documents/act-and-policies/rules-for-information-technology-act-2000.

[25] Mayuresh B. Gulame, Nilesh N. Thorat, Aarti P. Pimpalkar, Deepali A. Lokare, Security Aspects of Blockchain Technology, 1st Edition, First Published 2024, CRC Press Pages18, eBook ISBN9781003459347

[26] C. Y. Lin, “ROUGE: A package for automatic evaluation of summaries,” in Proc. ACL Workshop on Text Summarization Branches Out, Barcelona, Spain, Jul. 2004, pp. 74:81.

[27] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger and Y. Artzi, “BERTScore: Evaluating text generation with BERT,” arXiv preprint arXiv:1904.09675, 2019. [Online]. Available: https://arxiv.org/abs/1904.09675.

[28] Microsoft and Facebook, “ONNX: Open Neural Network Exchange,” [Online]. Available: https://onnx.ai.