MAJOR PROJECT

ON

Smart Email Generator & Chrome Browser Extension

Submitted to

Amity University Kolkata

For the partial fulfillment of the award of the degree

BACHELOR OF COMPUTER APPLICATIONS

by

AYUSH GUPTA

Enrolment No. -A91404822003

Guide- Dr. Subha Ghosh

AMITY INSTITUTE OF INFORMATION TECHNOLOGY

AMITY UNIVERSITY KOLKATA

April 2025

DECLARATION

I hereby declare that the dissertation entitled “GemMail Smart Email Generator & Chrome Browser Extension ” submitted by me in partial fulfilment of the requirements for the award of the Degree of Bachelor of Computer Applications to the AMITY UNIVERSITY, KOLKATA is based on the experiments and studies carried out by me. This work is original and has not been submitted in part or full for any other degree or diploma of any university or institution.

Date:
Place: Kolkata

AYUSH GUPTA
(A91404822003)

AMITY INSTITUTE OF INFORMATION TECHNOLOGY

AMITY UNIVERSITY KOLKATA

CERTIFICATE

This research work embodied in this dissertation entitled “GemMail **Smart Email Generator & Chrome Browser Extension”** submitted by **AYUSH GUPTA**, Enrollment No. **A91404822003** in partial fulfilment of the requirements for the award of the Degree of Bachelor of Computer Applications to the AMITY UNIVERSITY, KOLKATA is based on the experiments and studies carried out by him/her. This work is original and has not been submitted in part or full for any other degree or diploma of any university or institution.

Date:

Place: Kolkata

Dr. Subha Ghosh

Assistant Professor,

Amity Institute of Information Technology,

Amity University Kolkata

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Amity University, Kolkata for providing me this valuable opportunity. The experience has been instrumental in advancing my skills and deepening my understanding of partial design applications in a real-world setting.

My heartfelt appreciation goes to my mentor **Dr. Subha Ghosh** at Amity University, Kolkata, whose guidance and constructive feedback were invaluable in overcoming challenges and achieving my project objectives.

I would also like to thank my professors and college faculty for their continuous support and encouragement throughout my journey. Their insights and teachings laid a solid foundation for the skills I applied during this project.

Finally, I am thankful to my family and friends for their encouragement and patience as I pursued this exciting endeavour. This journey has been immensely rewarding, and I am deeply grateful to everyone who supported me along the way.

AMITY UNIVERSITY KOLKATA

FEEDBACK BY EXAMINERS

A. <u>Comments From Seminar Guide</u>

B. <u>Comments From External Examiner</u>

Table of Contents

ABSTRACT

GemMail is an AI-powered tool designed to revolutionize email communication by making it faster, smarter, and more intuitive. In a world where emails remain central to professional and personal interactions, many users struggle with drafting timely and well-phrased responses. GemMail solves this problem using Google's **Gemini API** to generate context-aware, personalized email replies that go beyond generic templates.

Key features include **tone customization**—offering options like formal, friendly, or persuasive—and a **copy-to-clipboard** function for quick integration into your workflow. To further enhance usability, a **Chrome extension** brings GemMail's functionality directly into Gmail, enabling seamless, real-time suggestions without disrupting the user experience.

This report outlines the development of GemMail using **React, Vite, Material UI, Spring Boot, and Gemini API**, highlighting the challenges faced, solutions implemented, and the tool's current capabilities. Looking ahead, future enhancements may include multi-language support, advanced sentiment analysis, and smarter intent detection. GemMail is more than just an email assistant—it's a leap toward AI-enhanced communication.

GemMail Smart Email Generator & Chrome Browser Extension

1. Introduction

In this day and age, email is still a staple of how we communicate, both professionally and personally. But let's get real: composing good, personalized emails is kinda a time-suck, especially when you've got a pile of messages to sort through. It's not always easy to know what to say, sound how you're supposed to sound, and get your replies to be clear, concise, and professional. That's precisely the issue this project seeks to remedy. We've developed GemMail, an AI-powered email generator that streamlines and speeds up the whole process of writing emails.

GemMail takes advantage of the Gemini API's sophisticated natural language processing capabilities to:

• Create intelligent, contextually appropriate email responses.

• Enable users to choose the tone of the produced text (think formal, informal, friendly, etc.).

• Possess a convenient copy-to-clipboard feature for integration with email programs with ease.

• Possess a browser extension for integrating with Gmail natively.

Using these capabilities, GemMail enables individuals to compose emails better and quicker, making time savings while enhancing the overall writing quality of the emails. We have kept the app straightforward to operate and use, and more intuitive as well, so it becomes easy to access by any user, regardless of his/her level of technical competency. Lastly, GemMail seeks to minimize the effort involved in composing emails, freeing users to concentrate on the content and purpose of their message, rather than being occupied with the mechanics of writing.

2. Project Goals and Objectives

The main goals we had for the GemMail project were to:

•Develop a working web application that could create AI-generated email responses.

•Utilize the Gemini API to carry out the natural language processing and text generation.

•Offer user control over the tone of the output email responses.

•Create a browser extension to make GemMail integrate nicely with Gmail.

In order to achieve these goals, we established the following main goals:

• Develop and deploy the user interface (UI) in React and Material UI. This involved creating a tidy, intelligent, and responsive user interface that could readily take in email text, select a tone, and generate a reply with minimal trouble.

• Create a robust backend API using Spring Boot to handle the requests from the frontend and send requests to the Gemini API. This included implementing the server stack, determining endpoints needed for API, and performing the core implementation of handling the email content and generating responses.

•Employ the Gemini API to execute the necessary email creation logic. This involved creating helpful prompts to guide the AI, handling the API calls and results, and converting the returned text into factual, relevant, and naturally readable text.

•Develop the browser extension using JavaScript. This involved diving into the Gmail API, working with the Document Object Model (DOM) to insert our features, and ensuring compatibility was present among different versions of Gmail.

• Provide seamless integration between the frontend, backend, and browser extension. This required synchronizing communication between these multiple components, transmitting data, and resolving any issues of compatibility that arose.

• Test the application and the extension extensively so that they function correctly and are stable. This comprised executing unit tests for every component, integration tests for the system itself, and user acceptance testing so as to ascertain the usability and overall user experience of the application.

3. System Architecture

GemMail's architecture has several key parts that work together:

• Frontend (React): The frontend, built with React and Material UI, is the interface that users use to interact with the functionality of GemMail. React's component-oriented nature allows modular and manageable code, and Material UI provides pre-styled components that provide the application with a consistent and modern look.

• Backend (Spring Boot): The backend server, which is based on Spring Boot, handles API requests from the frontend, communicates with the Gemini API, and orchestrates the email generation process. Spring Boot, through its dependency injection and auto-configuration capabilities, greatly simplifies the construction of robust and scalable web applications.

• Gemini API: The Gemini API from Google is the power behind GemMail, generating the email content based on the input text and tone. The Gemini API reveals advanced natural language processing models capable of understanding and generating human language with remarkable accuracy.

•Browser Extension (JavaScript): The browser extension, built with JavaScript, integrates GemMail's functionality into the Gmail UI. This allows users to access GemMail's functionality from within their inbox and not have to leave it, making it a seamless and user-friendly experience.

The system architecture is briefly outlined as below:

[Diagram of System Architecture (React Frontend -> Spring Boot Backend -> Gemini API) with Browser Extension interacting with Gmail and Backend]

3.1 Frontend (React)

The frontend of GemMail is an SPA built in React. Some of its main features are:

• Material UI Components: We made use of Material UI to provide a contemporary and consistent user interface. Material UI comes with an extensive collection of pre-designed components, such as buttons, input fields, and dialogs, which we used to create the application's UI. These components not only become customizable but responsive as well, providing a consistent user experience on all types of devices.

•Email Input Field: This is where the user enters the email they want to respond to. Typically, it's a text field where users can paste or type in the contents of the email. The frontend handles this user input and makes sure that it's properly formatted before handing it off to the backend.

• Tone Selection: This is the ability for choosing the desired tone of the response to be produced (e.g., formal, informal, friendly). This is necessary, as it allows users to tailor the produced email to meet the provided context and audience. The frontend provides a simple selection choice, such as a dropdown menu or radio buttons, for users to choose from a predetermined list of tones.

• Generate Response Button: The button triggers the API call to the backend, which then generates the email response. When the user clicks this button, the frontend makes a request to the backend API with the email content and the selected tone.

• Copy to Clipboard Button: It provides the user an ability to quickly and easily copy the generated response. It is a small feature, yet it provides a convenient way for the user to insert the generated email into his/her workflow of utilizing the email client. The frontend uses JavaScript to implement this copy-to-clipboard functionality.

• Loading Indicator: To keep the user informed, we've included a loading indicator that provides visual feedback while the AI is generating the response. This improves the user experience by letting the user know that the application is processing and the response will be available shortly. The frontend displays a loading animation or message while waiting for the response from the backend.

3.2 Backend (Spring Boot)

GemMail's backend is a RESTful API developed using Spring Boot. Below is a summary of its key functionalities:

•API Endpoints: Backend defines some endpoints via which the frontend can ask for the generation of an email. Endpoints are utilized to get the content and tone details of an email and send back the resulting email response.

•Gemini API Integration: The backend will manage the interaction with the Gemini API to send the email responses. It will use the Gemini API client library for sending requests to the Gemini API, including the email content and tone information.

• Tone Management: The most significant operation of the backend is managing the user-selected tone and adding it to the prompt to be sent to the Gemini API. The backend translates the user's tone selection into a specific prompt so that the generated response will have the same tone.

• Error Handling: The backend possesses robust error handling processes to catch any exceptions or errors that occur during the process of email generation. It then returns a corresponding error message to the frontend, providing the user with useful information regarding what occurred.

3.3 Gemini API

The Gemini API is an integral component of GemMail, and it provides the following core functionality:

• Natural Language Processing (NLP): Gemini analyzes the incoming email text to understand its meaning and context. NLP in Gemini enables it to identify critical information, sentiment, and intent of the email, which is necessary in order to generate a relevant and apt response.

• Text Generation: Gemini employs its sophisticated language models to generate human-like text that is grammatically accurate, contextually relevant, and in line with the tone specified by the user.

3.4 Browser Extension

The GemMail browser extension really enhances the user experience by complementing Gmail nicely. Here's why:

•Gmail Integration: The add-on places a "Generate with GemMail" button directly within the Gmail interface. The personalized button starts the email creation process and is placed in the Gmail compose window to make it readily accessible.

•Context Extraction: The add-on uses JavaScript to navigate the Gmail DOM and extract the relevant email content from the current Gmail thread, including the sender, recipient, subject, and body.

• API Communication: The extension sends the extracted email content to the backend of GemMail to process using an API call. The backend can then use the Gemini API to generate an appropriate response.

• Response Insertion: The moment the backend returns the created email response, the extension places it in the Gmail compose window through JavaScript so the user can inspect and send it with minimal effort.

4. Development Process

The development of GemMail was an iterative process, and it involved the following key phases:

4.1 Phase 1: Planning and Design:

- We started by defining the project's goals, objectives, and scope. This involved clearly outlining the purpose of the application, the features it would provide, and the intended target audience.
- We then created the system architecture and how each of the different components would interact with each other. This included creating diagrams and flowcharts to illustrate data flow and how the different parts of the system interacted with each other.

- We then created UI mockups using wireframing tools. These low-fidelity and high-fidelity mockups enabled us to visualize the layout, design, and user flow of the app.
- Finally, we selected the tools and technologies we would use for the project, for example, React, Spring Boot, Material UI, and the Gemini API. This involved looking at various possibilities and selecting the most suitable ones for the project.

4.2 Phase 2: Frontend Development:

- We set up the React development environment by installing the necessary software and tools like Node.js, npm, and a decent code editor.
- We then designed the user interface using Material UI components, constructing the various screens and components of the app, such as the email input field, tone option dropdown, and generate response button.
- We implemented the frontend's key functionalities, which involve email input, tone selection, and button clicks. This involved writing the JavaScript code for the response to users and managing the application logic.
- Finally, we linked the frontend to the backend API, writing the logic to issue API requests and handle the server responses.

4.3 Phase 3: Backend Development:

- We set up the Spring Boot development environment, installing Java, the Spring Boot CLI, and our chosen IDE.
- We built the RESTful API endpoints that would handle the email generation requests from the frontend.
- We integrated the backend with the Gemini API, writing the interactions with the API, sending requests, and receiving the responses generated.

- We implemented error handling and response formatting so that the backend could deal with any issues gracefully and return data in a formatted manner that would be easy for the frontend to consume.

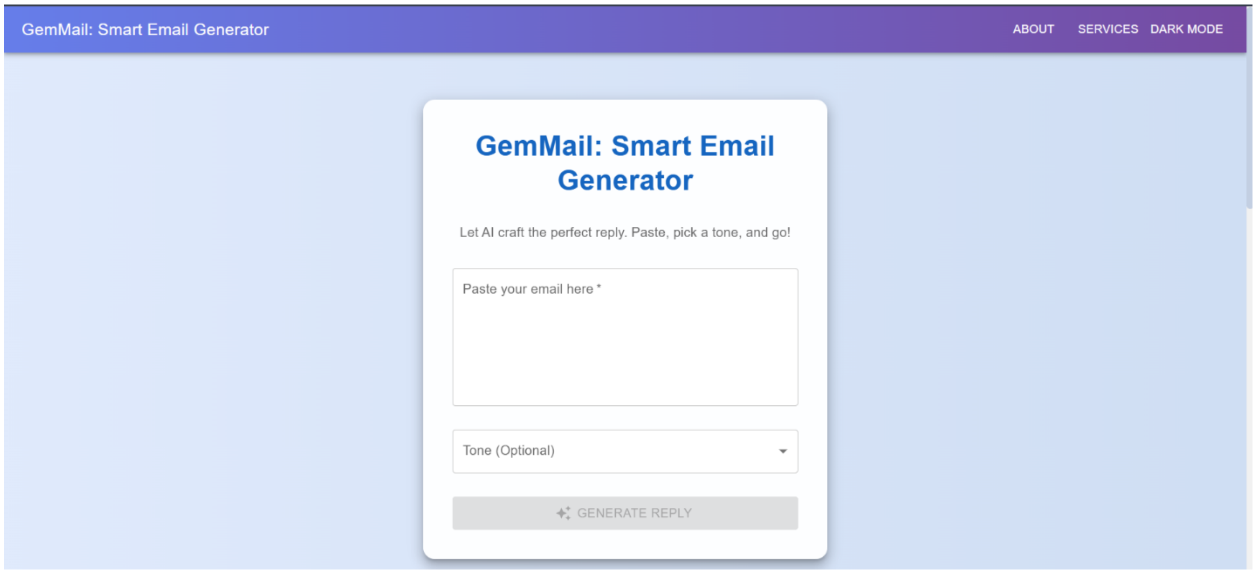
4.4 Phase 4: Browser Extension Development:

- We set up the development environment for the browser extension, such as a code editor and the browser's developer tools.
- We developed the JavaScript code to enable the Gmail integration, such as working with the Gmail DOM, modifying the interface, and handling user interactions within the Gmail environment.
- We implemented the context extraction and response insertion functionality, writing code for extracting the email content from the Gmail thread and inserting the generated response into the compose window.
- We tested the extension extensively with different versions of Gmail for compatibility and to identify any potential problems.

4.5 Phase 5: Testing and Deployment:

- We performed intense testing, including unit testing, integration testing, and user acceptance testing, to validate the functionality of every component, integration between components, and overall usability of the application.
- We identified and fixed any faults or issues that were uncovered during testing.
- We deployed the web application to a suitable hosting platform, including choosing a hosting provider, setting up the server, and deploying the application code.
- We packaged and published the browser extension, creating a package of the extension and releasing it to the respective browser's extension store.

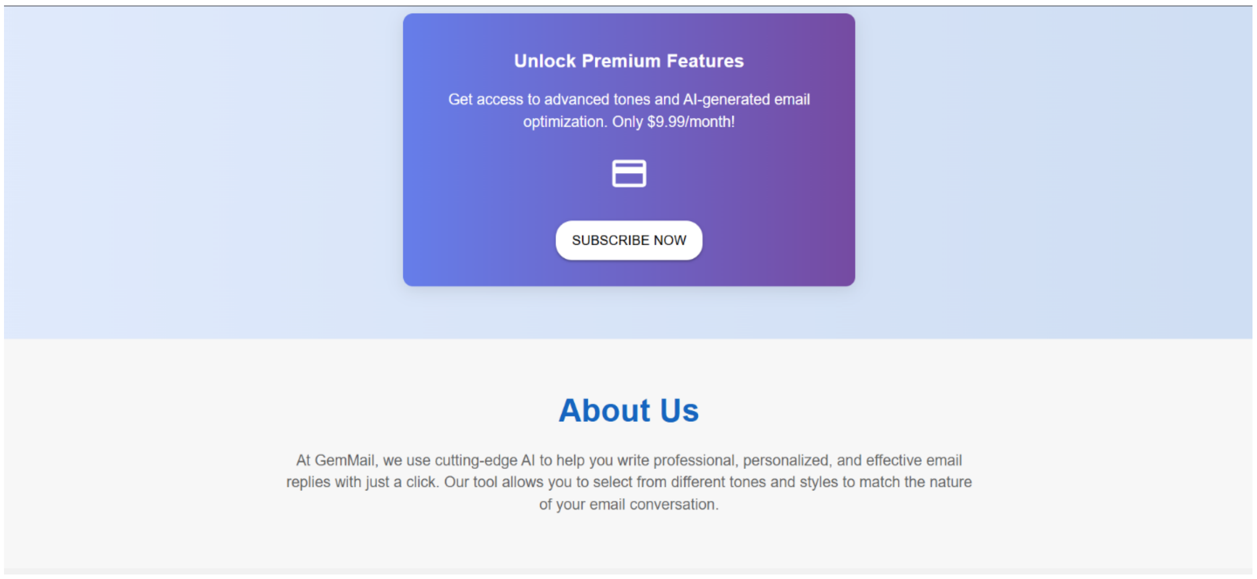
5. Designs and Screenshots



GemMail: Smart Email Generator

Let AI craft the perfect reply. Paste, pick a tone, and go!

Fig: Front Page of the GemMail



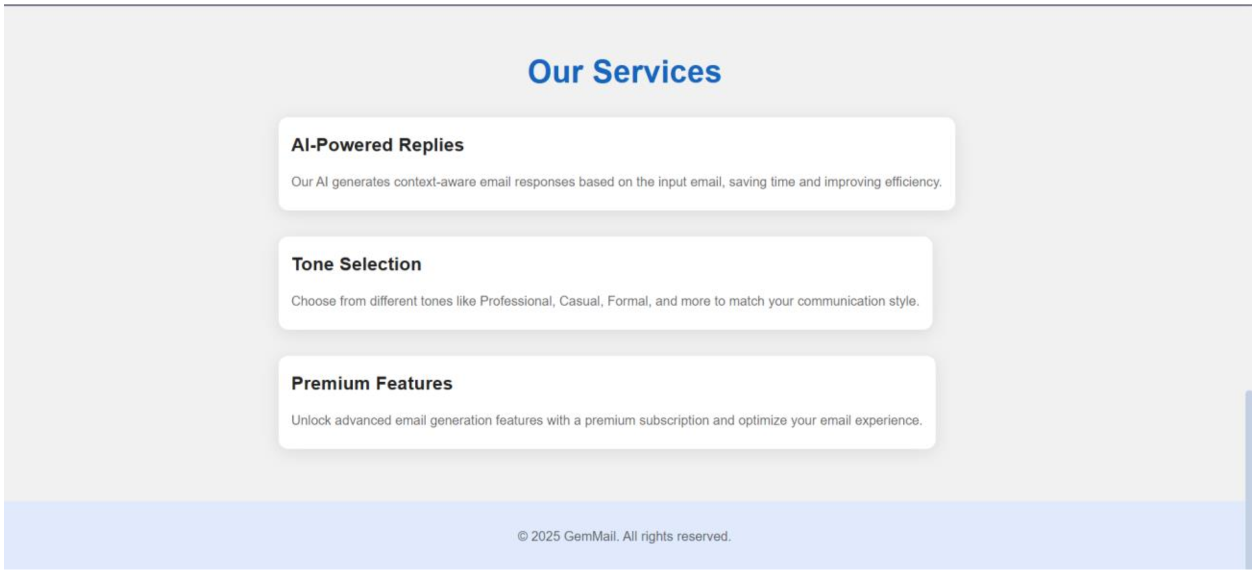
Unlock Premium Features

Get access to advanced tones and AI-generated email optimization. Only $9.99/month!

About Us

At GemMail, we use cutting-edge AI to help you write professional, personalized, and effective email replies with just a click. Our tool allows you to select from different tones and styles to match the nature of your email conversation.

Fig: PaymentCard and About Us



Our Services

AI-Powered Replies

Our AI generates context-aware email responses based on the input email, saving time and improving efficiency.

Tone Selection

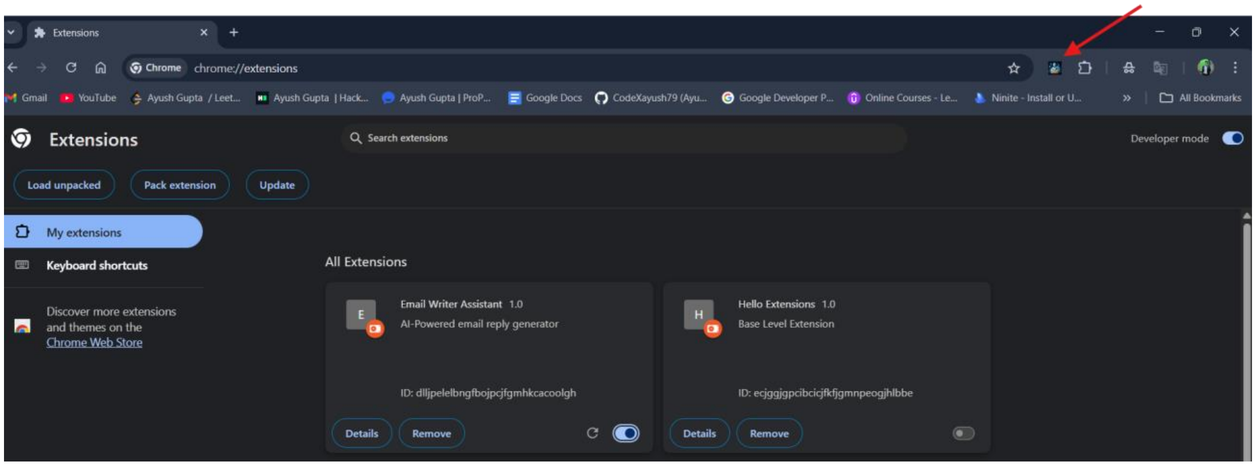
Choose from different tones like Professional, Casual, Formal, and more to match your communication style.

Premium Features

Unlock advanced email generation features with a premium subscription and optimize your email experience.

© 2025 GemMail. All rights reserved.

Fig: Service Page



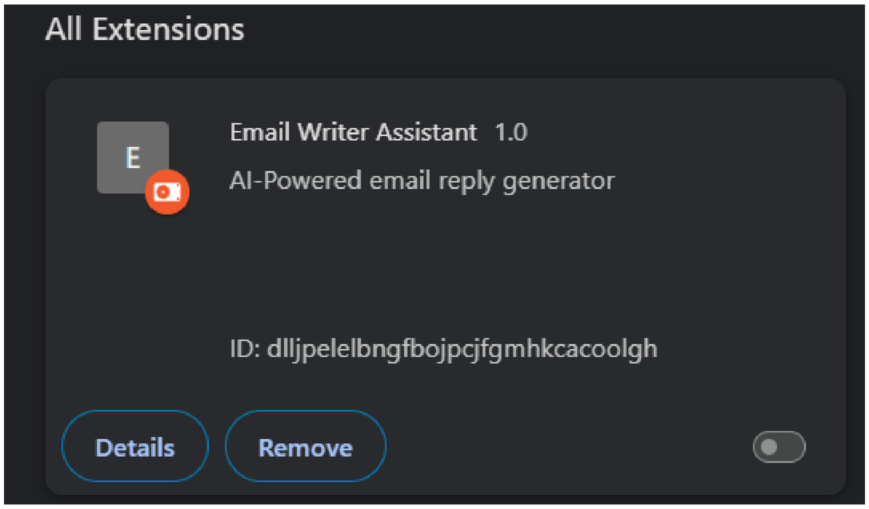
Discover more extensions and themes on the Chrome Web Store

All Extensions

Email Writer Assistant 1.0
AI-Powered email reply generator
ID: dlljpelelbngfbojpcjfgmhkcacoolgh

Hello Extensions 1.0
Base Level Extension
ID: ecjggjgpcibcicjfkfjgmnpeogjhlbbe

Fig: Extension Running in the Chrome


All Extensions

Email Writer Assistant 1.0

AI-Powered email reply generator

ID: dlljpelelbngfbojpcjfgmhkcacoolgh

Fig: Extension Unpack in Chrome DevExtension Tools

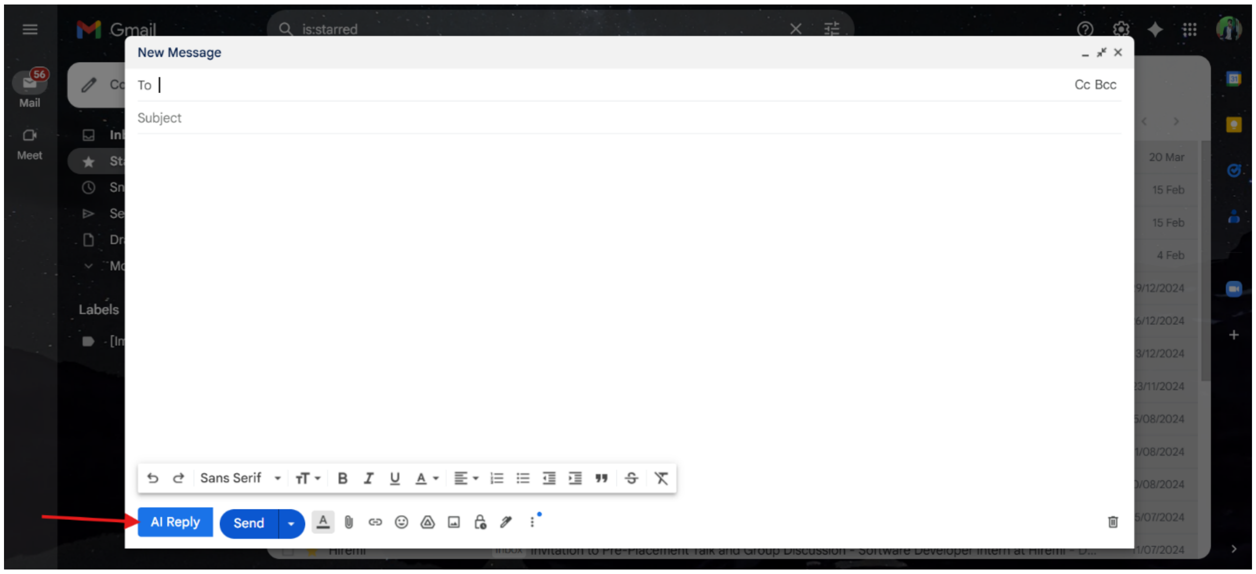


Fig: AI button due to Extension on Gmail

6. Challenges Faced

GemMail development involved surmounting several challenges:

6.1 Gemini API Integration: Smooth integration and utilization of the Gemini API required meticulous planning of API usage limits, request structure, and response parsing. We had to ensure efficient utilization of the API by the application, proper error handling in case of any errors, and correct interpretation of responses.

6.2 Gmail Integration: Building a robust and reliable browser extension that would seamlessly integrate with Gmail's dynamic environment was extremely difficult. Gmail's complex architecture and constant updates meant the extension had to be developed in a manner that would adapt to changes in Gmail's HTML and JavaScript code so that it would remain functional.

6.3 Tone Personalization: Acquiring the desired tone conveyed to the Gemini API and ensuring the generated response from it was in the tone required some good amount of experimentation and prompt engineering. Several prompts and parameters had to be tested in order to find out the best way to influence the output of the API.

6.4 Asynchronous Operations: Managing asynchronous operations, like API calls and interactions with Gmail, involved the critical management of callbacks and promises so that there could be a smooth user experience. We needed to architect the app to process such operations without hindering the user interface or the app crashing.

6.5 Cross-Browser Compatibility: Making sure the browser extension worked properly across various web browsers (Chrome, Firefox, etc.) involved extensive testing and tweaking. Every browser has its own idiosyncrasies and variations in how it interprets JavaScript and web standards, which we needed to resolve.

7. Results

GemMail was successful in what it set out to do, and we're satisfied with the outcome:

•We have a functional web application that allows users to generate AI-based email responses with selectable tones. The app has an easy interface for inputting email content, selecting a tone, and generating a response.

•GemMail is integrated with Gmail through a simple browser extension. This browser extension allows the user to see GemMail features within their Gmail interface, which simplifies their work process and provides them with an effortless experience.

•We've received positive user comments regarding the ease of use of the application and the time it saves users. Users have mentioned that GemMail compels them to send emails more quickly and effectively, and that the responses that are generated tend to be accurate and helpful.

• The project has laid a strong foundation for future expansion and development of GemMail's feature set. We've established a robust architecture and codebase that is easily extensible to include new functionality.

GemMail shows how AI can make email communication more efficient and effective. By automating the process of replying to emails, GemMail allows users to save time and focus on more critical work. The application's tone customization feature also allows users to customize responses to specific contexts and recipients to make their communication professional and appropriate at all times.

8. Future Scope

GemMail has enormous potential for expansion and development in the future. We envision the following changes:

•Other Email Platform Integration: We plan to expand the browser extension to support other popular email platforms, such as Outlook and Yahoo Mail. This would boost the base of the application and provide the app to a wider audience.

• Enhanced Tone Personalization: We'd like to provide more control over the generated response tone, i.e., allowing users to specify emotion, style, and formality level. This will enable users to personalize the generated responses according to their specific needs and preferences.

•Smart Reply Suggestions: We would like to add a feature that provides a number of reply options with different tones so that users can choose the most appropriate response. This would allow users to have more freedom and flexibility in their email communication.

•Multilingual Support: We plan to include making GemMail produce email responses in different languages so that the software can be utilized by users who use different languages.

•Personalized Responses: We are also considering the possibility of training the AI model on user-specific email data so that it generates more personalized and customized responses. This would allow the application to learn the writing style and preference of the user and therefore generate more natural and effective communication.

•Paid Version with Additional Features: We can provide a paid version of GemMail with additional features, such as:

- Higher API usage limits.
- Priority support.
- Customizable templates.
- CRM and other business software integration.

•Mobile Application: We also see creating a mobile app for GemMail to provide email support on the move. This would allow users to utilize the capabilities of GemMail on their smartphones and tablets, providing them with greater flexibility and convenience.

9.Appendix

- **Flowchart -**

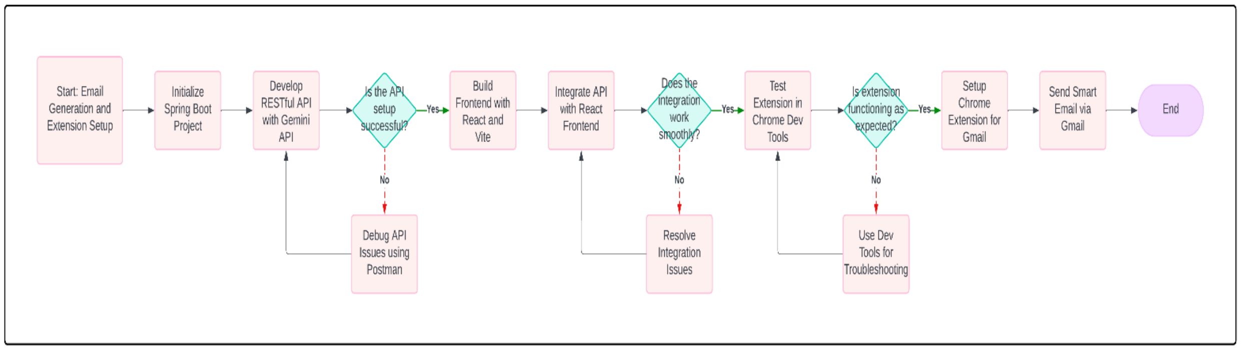

Start: Email Generation and Extension Setup
Initialize Spring Boot Project
Develop RESTful API with Gemini API
Is the API setup successful?
Yes
No
Debug API Issues using Postman
Build Frontend with React and Vite
Integrate API with React Frontend
Does the integration work smoothly?
Yes
No
Resolve Integration Issues
Test Extension in Chrome Dev Tools
Is extension functioning as expected?
Yes
No
Use Dev Tools for Troubleshooting
Setup Chrome Extension for Gmail
Send Smart Email via Gmail
End

Fig: Flowchart of GemMail

- **ER Diagram –**

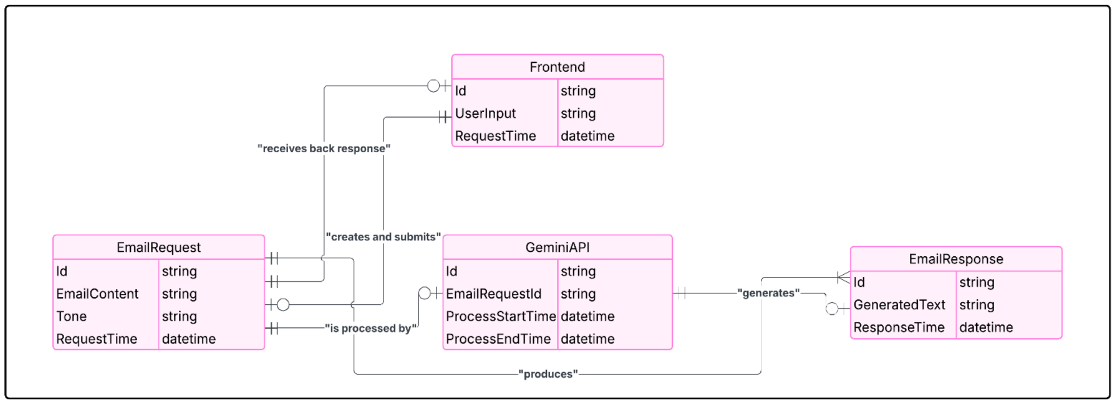


Fig: ER Diagram of GemMail

10. Conclusion

GemMail is an exciting project that leverages the power of AI to fix a collective pain point in email messaging. With intelligent email composition, tone optimization, and seamless Gmail integration, GemMail helps people communicate better and faster. The project showcases the ability of AI to increase productivity and streamline processes. Through constant development and progress, GemMail would be an extremely valuable instrument for both persons and companies. Due to its simplicity of usage, as well as its high-end AI power, it is a very practical instrument for whoever needs to become better at e-mail communication and have more spare time.

11. References

1. Gemini API Documentation: [Link to Gemini API Documentation]
2. React Documentation: [Link to React Documentation]
3. Spring Boot Documentation: [Link to Spring Boot Documentation]
4. Material UI Documentation: [Link to Material UI Documentation]

Ayush Gupta

GemMail Smart Email Generator & Chrome Browser Extension

My Files

My Files

University Of Engineering And Management, Jaipur

Document Details

Submission ID	trn:oid:::9639:93164372
Submission Date	Apr 27, 2025, 9:36 PM GMT+5:30
Download Date	Apr 27, 2025, 9:39 PM GMT+5:30
File Name	GemMail-report (3).docx
File Size	842.0 KB

15 Pages

3,072 Words

17,089 Characters

0% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

- 1 Not Cited or Quoted 0%
  Matches with neither in-text citation nor quotation marks
- 0 Missing Quotations 0%
  Matches that are still very similar to source material
- 0 Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

0%	Internet sources
0%	Publications
0%	Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **1** Not Cited or Quoted 0%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

0%	Internet sources
0%	Publications
0%	Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	
	careers.gore.com	<1%

GemMail Smart Email Generator & Chrome Browser Extension

1. Introduction

In this day and age, email is still a staple of how we communicate, both professionally and personally. But let's get real: composing good, personalized emails is kinda a time-suck, especially when you've got a pile of messages to sort through. It's not always easy to know what to say, sound how you're supposed to sound, and get your replies to be clear, concise, and professional. That's precisely the issue this project seeks to remedy. We've developed GemMail, an AI-powered email generator that streamlines and speeds up the whole process of writing emails.

GemMail takes advantage of the Gemini API's sophisticated natural language processing capabilities to:

• Create intelligent, contextually appropriate email responses.

• Enable users to choose the tone of the produced text (think formal, informal, friendly, etc.).

• Possess a convenient copy-to-clipboard feature for integration with email programs with ease.

• Possess a browser extension for integrating with Gmail natively.

Using these capabilities, GemMail enables individuals to compose emails better and quicker, making time savings while enhancing the overall writing quality of the emails. We have kept the app straightforward to operate and use, and more intuitive as well, so it becomes easy to access by any user, regardless of his/her level of technical competency. Lastly, GemMail seeks to minimize the effort involved in composing emails, freeing users to concentrate on the content and purpose of their message, rather than being occupied with the mechanics of writing.

2. Project Goals and Objectives

The main goals we had for the GemMail project were to:

•Develop a working web application that could create AI-generated email responses.

•Utilize the Gemini API to carry out the natural language processing and text generation.

•Offer user control over the tone of the output email responses.

•Create a browser extension to make GemMail integrate nicely with Gmail.

In order to achieve these goals, we established the following main goals:

• Develop and deploy the user interface (UI) in React and Material UI. This involved creating a tidy, intelligent, and responsive user interface that could readily take in email text, select a tone, and generate a reply with minimal trouble.

• Create a robust backend API using Spring Boot to handle the requests from the frontend and send requests to the Gemini API. This included implementing the server stack, determining endpoints needed for API, and performing the core implementation of handling the email content and generating responses.

•Employ the Gemini API to execute the necessary email creation logic. This involved creating helpful prompts to guide the AI, handling the API calls and results, and converting the returned text into factual, relevant, and naturally readable text.

•Develop the browser extension using JavaScript. This involved diving into the Gmail API, working with the Document Object Model (DOM) to insert our features, and ensuring compatibility was present among different versions of Gmail.

• Provide seamless integration between the frontend, backend, and browser extension. This required synchronizing communication between these multiple components, transmitting data, and resolving any issues of compatibility that arose.

• Test the application and the extension extensively so that they function correctly and are stable. This comprised executing unit tests for every component, integration tests for the system itself, and user acceptance testing so as to ascertain the usability and overall user experience of the application.

3. System Architecture

GemMail's architecture has several key parts that work together:

• Frontend (React): The frontend, built with React and Material UI, is the interface that users use to interact with the functionality of GemMail. React's component-oriented nature allows modular and manageable code, and Material UI provides pre-styled components that provide the application with a consistent and modern look.

• Backend (Spring Boot): The backend server, which is based on Spring Boot, handles API requests from the frontend, communicates with the Gemini API, and orchestrates the email generation process. Spring Boot, through its dependency injection and auto-configuration capabilities, greatly simplifies the construction of robust and scalable web applications.

• Gemini API: The Gemini API from Google is the power behind GemMail, generating the email content based on the input text and tone. The Gemini API reveals advanced natural language processing models capable of understanding and generating human language with remarkable accuracy.

•Browser Extension (JavaScript): The browser extension, built with JavaScript, integrates GemMail's functionality into the Gmail UI. This allows users to access GemMail's functionality from within their inbox and not have to leave it, making it a seamless and user-friendly experience.

The system architecture is briefly outlined as below:

[Diagram of System Architecture (React Frontend -> Spring Boot Backend -> Gemini API) with Browser Extension interacting with Gmail and Backend]

3.1 Frontend (React)

The frontend of GemMail is an SPA built in React. Some of its main features are:

• Material UI Components: We made use of Material UI to provide a contemporary and consistent user interface. Material UI comes with an extensive collection of pre-designed components, such as buttons, input fields, and dialogs, which we used to create the application's UI. These components not only become customizable but responsive as well, providing a consistent user experience on all types of devices.

•Email Input Field: This is where the user enters the email they want to respond to. Typically, it's a text field where users can paste or type in the contents of the email. The frontend handles this user input and makes sure that it's properly formatted before handing it off to the backend.

• Tone Selection: This is the ability for choosing the desired tone of the response to be produced (e.g., formal, informal, friendly). This is necessary, as it allows users to tailor the produced email to meet the provided context and audience. The frontend provides a simple selection choice, such as a dropdown menu or radio buttons, for users to choose from a predetermined list of tones.

• Generate Response Button: The button triggers the API call to the backend, which then generates the email response. When the user clicks this button, the frontend makes a request to the backend API with the email content and the selected tone.

• Copy to Clipboard Button: It provides the user an ability to quickly and easily copy the generated response. It is a small feature, yet it provides a convenient way for the user to insert the generated email into his/her workflow of utilizing the email client. The frontend uses JavaScript to implement this copy-to-clipboard functionality.

• Loading Indicator: To keep the user informed, we've included a loading indicator that provides visual feedback while the AI is generating the response. This improves the user experience by letting the user know that the application is processing and the response will be available shortly. The frontend displays a loading animation or message while waiting for the response from the backend.

3.2 Backend (Spring Boot)

GemMail's backend is a RESTful API developed using Spring Boot. Below is a summary of its key functionalities:

•API Endpoints: Backend defines some endpoints via which the frontend can ask for the generation of an email. Endpoints are utilized to get the content and tone details of an email and send back the resulting email response.

•Gemini API Integration: The backend will manage the interaction with the Gemini API to send the email responses. It will use the Gemini API client library for sending requests to the Gemini API, including the email content and tone information.

• Tone Management: The most significant operation of the backend is managing the user-selected tone and adding it to the prompt to be sent to the Gemini API. The backend translates the user's tone selection into a specific prompt so that the generated response will have the same tone.

• Error Handling: The backend possesses robust error handling processes to catch any exceptions or errors that occur during the process of email generation. It then returns a corresponding error message to the frontend, providing the user with useful information regarding what occurred.

3.3 Gemini API

The Gemini API is an integral component of GemMail, and it provides the following core functionality:

• Natural Language Processing (NLP): Gemini analyzes the incoming email text to understand its meaning and context. NLP in Gemini enables it to identify critical information, sentiment, and intent of the email, which is necessary in order to generate a relevant and apt response.

• Text Generation: Gemini employs its sophisticated language models to generate human-like text that is grammatically accurate, contextually relevant, and in line with the tone specified by the user.

3.4 Browser Extension

The GemMail browser extension really enhances the user experience by complementing Gmail nicely. Here's why:

•Gmail Integration: The add-on places a "Generate with GemMail" button directly within the Gmail interface. The personalized button starts the email creation process and is placed in the Gmail compose window to make it readily accessible.

•Context Extraction: The add-on uses JavaScript to navigate the Gmail DOM and extract the relevant email content from the current Gmail thread, including the sender, recipient, subject, and body.

• API Communication: The extension sends the extracted email content to the backend of GemMail to process using an API call. The backend can then use the Gemini API to generate an appropriate response.

• Response Insertion: The moment the backend returns the created email response, the extension places it in the Gmail compose window through JavaScript so the user can inspect and send it with minimal effort.

4. Development Process

The development of GemMail was an iterative process, and it involved the following key phases:

4.1 Phase 1: Planning and Design:

- We started by defining the project's goals, objectives, and scope. This involved clearly outlining the purpose of the application, the features it would provide, and the intended target audience.
- We then created the system architecture and how each of the different components would interact with each other. This included creating diagrams and flowcharts to illustrate data flow and how the different parts of the system interacted with each other.

- We then created UI mockups using wireframing tools. These low-fidelity and high-fidelity mockups enabled us to visualize the layout, design, and user flow of the app.
- Finally, we selected the tools and technologies we would use for the project, for example, React, Spring Boot, Material UI, and the Gemini API. This involved looking at various possibilities and selecting the most suitable ones for the project.

4.2 Phase 2: Frontend Development:

- We set up the React development environment by installing the necessary software and tools like Node.js, npm, and a decent code editor.
- We then designed the user interface using Material UI components, constructing the various screens and components of the app, such as the email input field, tone option dropdown, and generate response button.
- We implemented the frontend's key functionalities, which involve email input, tone selection, and button clicks. This involved writing the JavaScript code for the response to users and managing the application logic.
- Finally, we linked the frontend to the backend API, writing the logic to issue API requests and handle the server responses.

4.3 Phase 3: Backend Development:

- We set up the Spring Boot development environment, installing Java, the Spring Boot CLI, and our chosen IDE.
- We built the RESTful API endpoints that would handle the email generation requests from the frontend.
- We integrated the backend with the Gemini API, writing the interactions with the API, sending requests, and receiving the responses generated.

- We implemented error handling and response formatting so that the backend could deal with any issues gracefully and return data in a formatted manner that would be easy for the frontend to consume.

4.4 Phase 4: Browser Extension Development:

- We set up the development environment for the browser extension, such as a code editor and the browser's developer tools.
- We developed the JavaScript code to enable the Gmail integration, such as working with the Gmail DOM, modifying the interface, and handling user interactions within the Gmail environment.
- We implemented the context extraction and response insertion functionality, writing code for extracting the email content from the Gmail thread and inserting the generated response into the compose window.
- We tested the extension extensively with different versions of Gmail for compatibility and to identify any potential problems.

4.5 Phase 5: Testing and Deployment:

1

- We performed intense testing, including unit testing, integration testing, and user acceptance testing, to validate the functionality of every component, integration between components, and overall usability of the application.
- We identified and fixed any faults or issues that were uncovered during testing.
- We deployed the web application to a suitable hosting platform, including choosing a hosting provider, setting up the server, and deploying the application code.
- We packaged and published the browser extension, creating a package of the extension and releasing it to the respective browser's extension store.

5. Designs and Screenshots

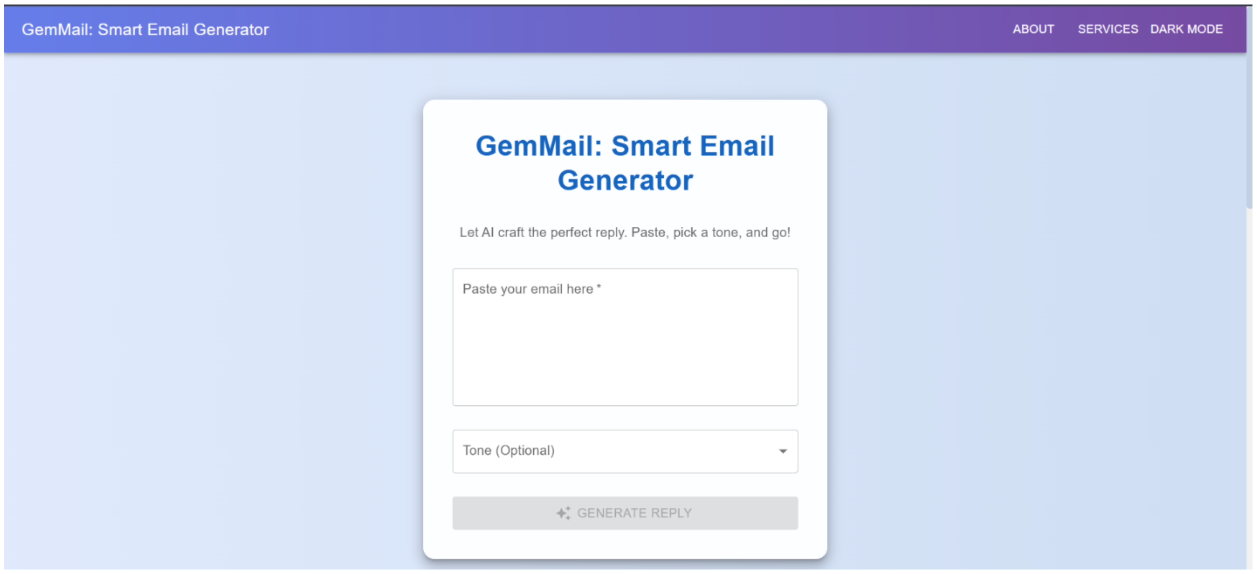
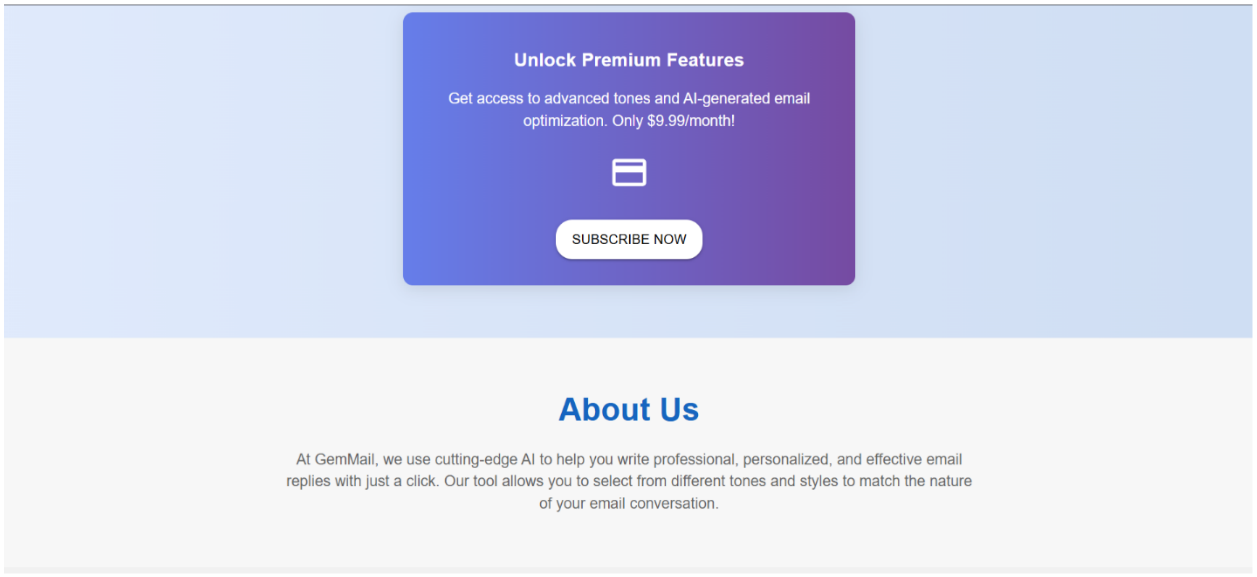

GemMail: Smart Email Generator
ABOUT
SERVICES
DARK MODE
GemMail: Smart Email Generator
Let AI craft the perfect reply. Paste, pick a tone, and go!
Paste your email here *
Tone (Optional)
GENERATE REPLY

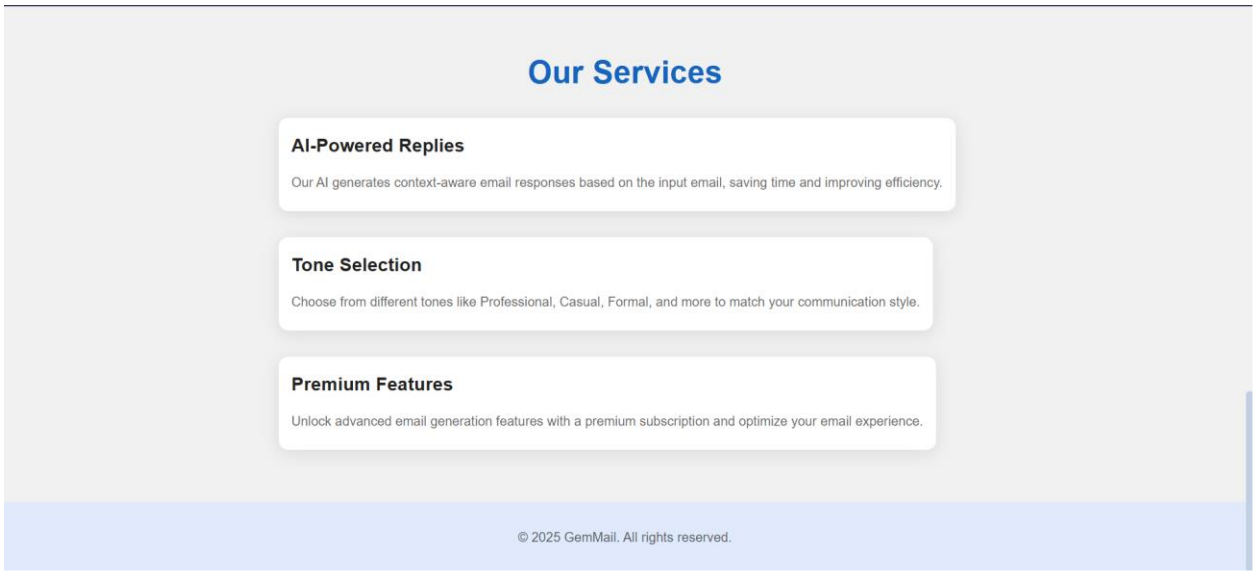
Fig: Front Page of the GemMail



Get access to advanced tones and AI-generated email optimization. Only $9.99/month!

At GemMail, we use cutting-edge AI to help you write professional, personalized, and effective email replies with just a click. Our tool allows you to select from different tones and styles to match the nature of your email conversation.

Fig: PaymentCard and About Us



Our Services

AI-Powered Replies

Our AI generates context-aware email responses based on the input email, saving time and improving efficiency.

Tone Selection

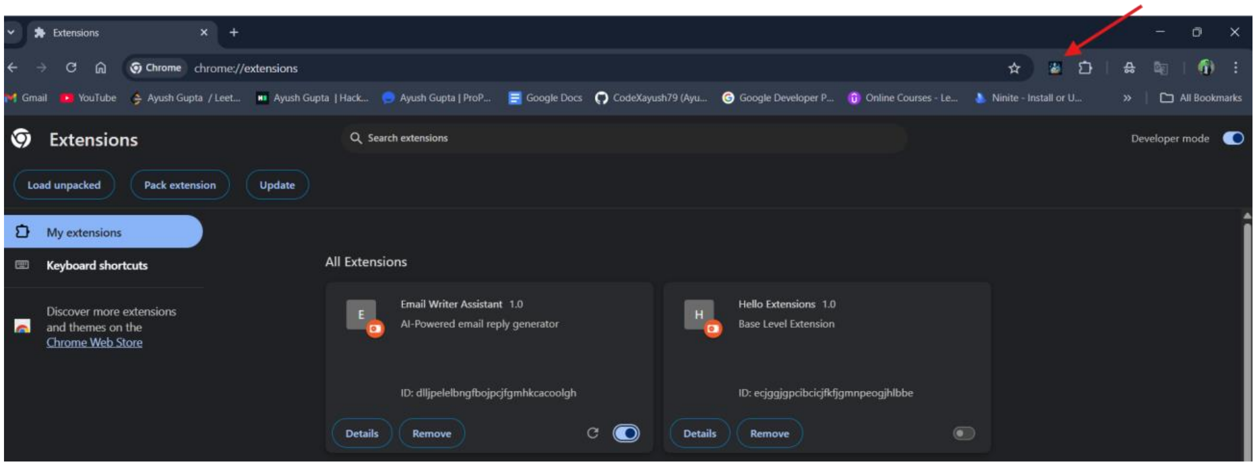
Choose from different tones like Professional, Casual, Formal, and more to match your communication style.

Premium Features

Unlock advanced email generation features with a premium subscription and optimize your email experience.

© 2025 GemMail. All rights reserved.

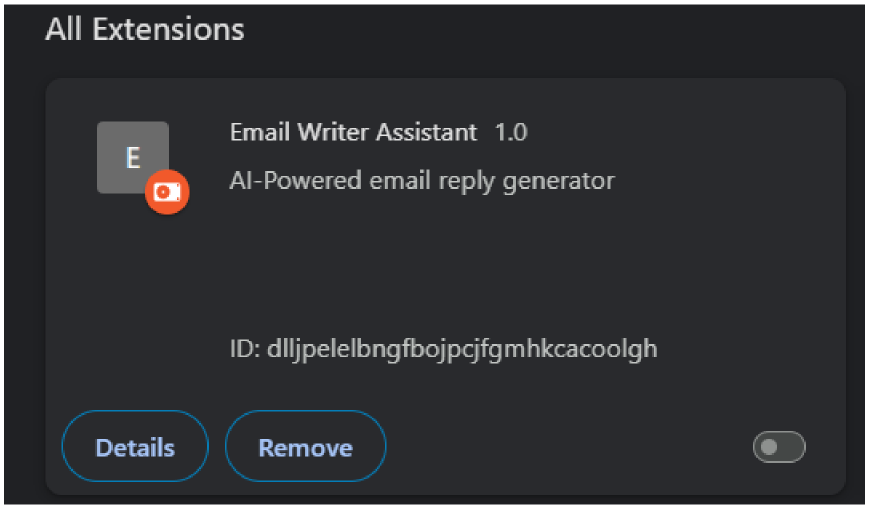
Fig: Service Page



Email Writer Assistant 1.0
AI-Powered email reply generator
ID: dljpelelbngfbojpcjfgmhkcacoolgh

Hello Extensions 1.0
Base Level Extension
ID: ecjggjgpcibcicjfkfjgmnpeogjhlbbe

Fig: Extension Running in the Chrome



All Extensions

Email Writer Assistant 1.0

AI-Powered email reply generator

ID: dlljpelelbngfbojpcjfgmhkcacoolgh

Fig: Extension Unpack in Chrome DevExtension Tools

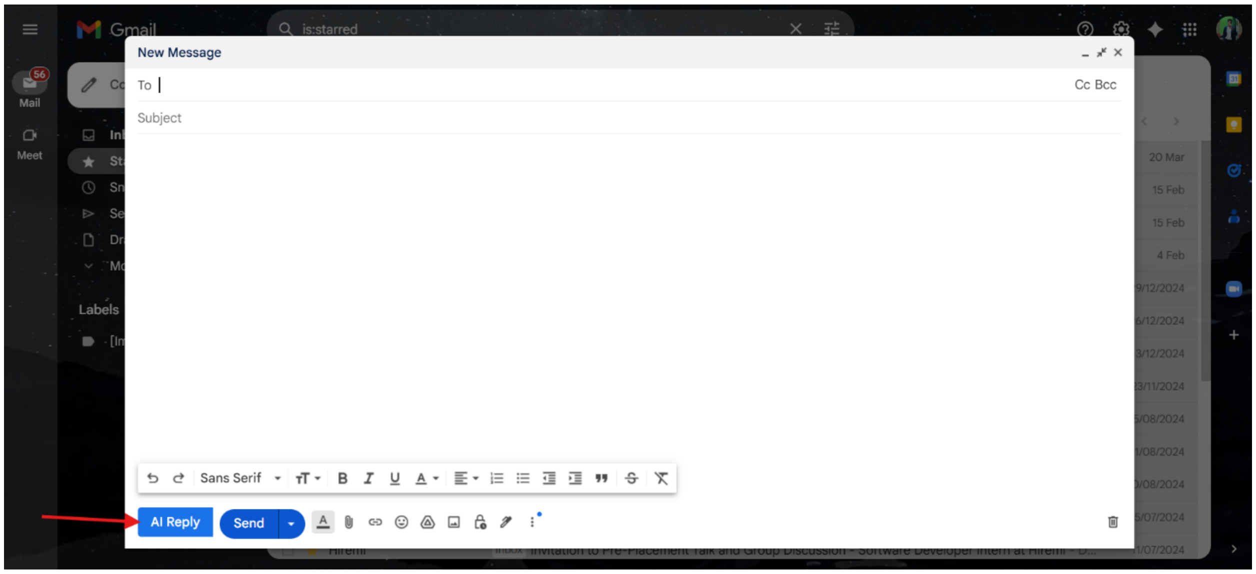


Fig: AI button due to Extension on Gmail

6. Challenges Faced

GemMail development involved surmounting several challenges:

6.1 Gemini API Integration: Smooth integration and utilization of the Gemini API required meticulous planning of API usage limits, request structure, and response parsing. We had to ensure efficient utilization of the API by the application, proper error handling in case of any errors, and correct interpretation of responses.

6.2 Gmail Integration: Building a robust and reliable browser extension that would seamlessly integrate with Gmail's dynamic environment was extremely difficult. Gmail's complex architecture and constant updates meant the extension had to be developed in a manner that would adapt to changes in Gmail's HTML and JavaScript code so that it would remain functional.

6.3 Tone Personalization: Acquiring the desired tone conveyed to the Gemini API and ensuring the generated response from it was in the tone required some good amount of experimentation and prompt engineering. Several prompts and parameters had to be tested in order to find out the best way to influence the output of the API.

6.4 Asynchronous Operations: Managing asynchronous operations, like API calls and interactions with Gmail, involved the critical management of callbacks and promises so that there could be a smooth user experience. We needed to architect the app to process such operations without hindering the user interface or the app crashing.

6.5 Cross-Browser Compatibility: Making sure the browser extension worked properly across various web browsers (Chrome, Firefox, etc.) involved extensive testing and tweaking. Every browser has its own idiosyncrasies and variations in how it interprets JavaScript and web standards, which we needed to resolve.

7. Results

GemMail was successful in what it set out to do, and we're satisfied with the outcome:

•We have a functional web application that allows users to generate AI-based email responses with selectable tones. The app has an easy interface for inputting email content, selecting a tone, and generating a response.

•GemMail is integrated with Gmail through a simple browser extension. This browser extension allows the user to see GemMail features within their Gmail interface, which simplifies their work process and provides them with an effortless experience.

•We've received positive user comments regarding the ease of use of the application and the time it saves users. Users have mentioned that GemMail compels them to send emails more quickly and effectively, and that the responses that are generated tend to be accurate and helpful.

• The project has laid a strong foundation for future expansion and development of GemMail's feature set. We've established a robust architecture and codebase that is easily extensible to include new functionality.

GemMail shows how AI can make email communication more efficient and effective. By automating the process of replying to emails, GemMail allows users to save time and focus on more critical work. The application's tone customization feature also allows users to customize responses to specific contexts and recipients to make their communication professional and appropriate at all times.

8. Future Scope

GemMail has enormous potential for expansion and development in the future. We envision the following changes:

•Other Email Platform Integration: We plan to expand the browser extension to support other popular email platforms, such as Outlook and Yahoo Mail. This would boost the base of the application and provide the app to a wider audience.

• Enhanced Tone Personalization: We'd like to provide more control over the generated response tone, i.e., allowing users to specify emotion, style, and formality level. This will enable users to personalize the generated responses according to their specific needs and preferences.

•Smart Reply Suggestions: We would like to add a feature that provides a number of reply options with different tones so that users can choose the most appropriate response. This would allow users to have more freedom and flexibility in their email communication.

•Multilingual Support: We plan to include making GemMail produce email responses in different languages so that the software can be utilized by users who use different languages.

•Personalized Responses: We are also considering the possibility of training the AI model on user-specific email data so that it generates more personalized and customized responses. This would allow the application to learn the writing style and preference of the user and therefore generate more natural and effective communication.

•Paid Version with Additional Features: We can provide a paid version of GemMail with additional features, such as:

- Higher API usage limits.
- Priority support.
- Customizable templates.
- CRM and other business software integration.

•Mobile Application: We also see creating a mobile app for GemMail to provide email support on the move. This would allow users to utilize the capabilities of GemMail on their smartphones and tablets, providing them with greater flexibility and convenience.

9.Appendix

- **Flowchart -**

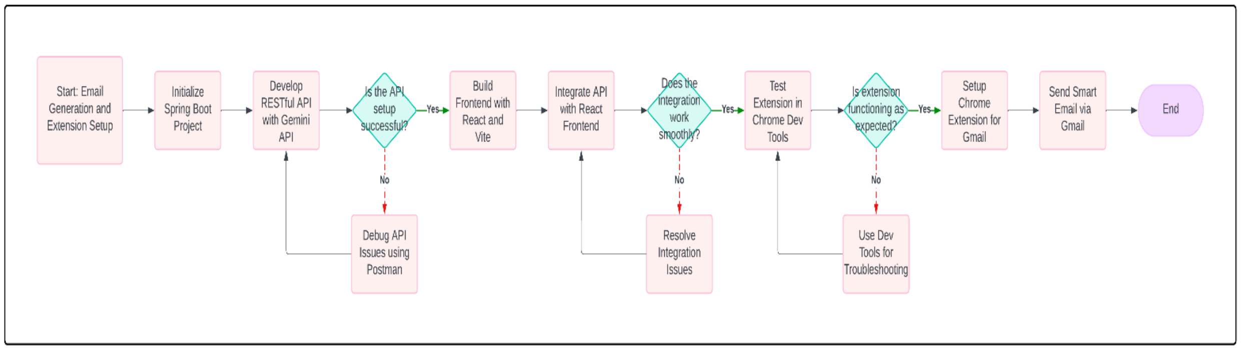

Start: Email Generation and Extension Setup
Initialize Spring Boot Project
Develop RESTful API with Gemini API
Is the API setup successful?
Yes
No
Debug API Issues using Postman
Build Frontend with React and Vite
Integrate API with React Frontend
Does the integration work smoothly?
Yes
No
Resolve Integration Issues
Test Extension in Chrome Dev Tools
Is extension functioning as expected?
Yes
No
Use Dev Tools for Troubleshooting
Setup Chrome Extension for Gmail
Send Smart Email via Gmail
End

Fig: Flowchart of GemMail

- **ER Diagram –**

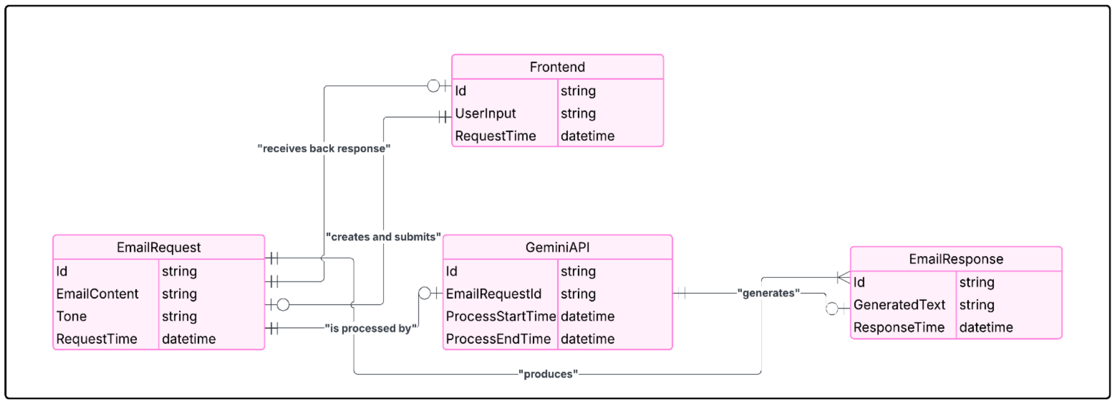


Fig: ER Diagram of GemMail

10. Conclusion

GemMail is an exciting project that leverages the power of AI to fix a collective pain point in email messaging. With intelligent email composition, tone optimization, and seamless Gmail integration, GemMail helps people communicate better and faster. The project showcases the ability of AI to increase productivity and streamline processes. Through constant development and progress, GemMail would be an extremely valuable instrument for both persons and companies. Due to its simplicity of usage, as well as its high-end AI power, it is a very practical instrument for whoever needs to become better at e-mail communication and have more spare time.

11. References

1. Gemini API Documentation: [Link to Gemini API Documentation]
2. React Documentation: [Link to React Documentation]
3. Spring Boot Documentation: [Link to Spring Boot Documentation]
4. Material UI Documentation: [Link to Material UI Documentation]

AMITY UNIVERSITY KOLKATA

Weekly Progress Report

For the Week Commencing: 1st Week

Enrollment No. : A91404822003

NAME: AYUSH GUPTA

Program: BCA

Faculty Guide's Name : Subha Ghosh

Internship Mode : Online

Project Title: GemMail Smart Email Generator & Chrome Browser Extension

Targets for the week: Project initialization and requirements gathering

Future work plans: Begin design phase and technology selection

Week's Summary

Days / Time	**Jan 20 - Jan 26**
Monday	Project kickoff meeting and core objectives established
Tuesday	Core requirements documented for email generation features
Wednesday	Initial system architecture drafted
Thursday	Technology stack finalized
Friday	Team roles assigned and project timeline created

AMITY UNIVERSITY KOLKATA

Weekly Progress Report

For the Week Commencing: 2nd Week

Enrollment No. : A91404822003

NAME: AYUSH GUPTA

Program: BCA

Faculty Guide's Name : Subha Ghosh

Internship Mode : Online

Project Title: GemMail Smart Email Generator & Chrome Browser Extension

Targets for the week: Complete design phase and UI mockups

Future work plans: Begin frontend development

Week's Summary

Days / Time	**Jan 27 - Feb 2**
Monday	Low-fidelity wireframes created
Tuesday	High-fidelity mockups developed
Wednesday	User workflows mapped
Thursday	Design review conducted
Friday	API specifications finalized

AMITY UNIVERSITY KOLKATA

Weekly Progress Report

For the Week Commencing: 3rd Week

Enrollment No. : A91404822003

NAME: AYUSH GUPTA

Program: BCA

Faculty Guide's Name : Subha Ghosh

Internship Mode : Online

Project Title: GemMail Smart Email Generator & Chrome Browser Extension

Targets for the week: Initialize frontend development

Future work plans: Begin backend development

Week's Summary

Days / Time	Feb 3 - Feb 9
Monday	React environment setup completed
Tuesday	Basic UI components implemented
Wednesday	Email input form developed
Thursday	Tone selection dropdown created
Friday	Copy-to-clipboard functionality added

AMITY UNIVERSITY KOLKATA

Weekly Progress Report

For the Week Commencing: 4th Week

Enrollment No. : A91404822003

NAME: AYUSH GUPTA

Program: BCA

Faculty Guide's Name : Subha Ghosh

Internship Mode : Online

Project Title: GemMail Smart Email Generator & Chrome Browser Extension

Targets for the week: Core backend development

Future work plans: Frontend-backend integration

Week's Summary

Days / Time	**Feb 10 - Feb 16**
Monday	Spring Boot environment configured
Tuesday	API endpoints implemented
Wednesday	Gemini API access secured
Thursday	Prompt engineering logic completed
Friday	Error handling implemented

AMITY UNIVERSITY KOLKATA

Weekly Progress Report

For the Week Commencing: 5th Week

Enrollment No. : A91404822003

NAME: AYUSH GUPTA

Program: BCA

Faculty Guide's Name : Subha Ghosh

Internship Mode : Online

Project Title: GemMail Smart Email Generator & Chrome Browser Extension

Targets for the week: Complete frontend-backend integration

Future work plans: Begin browser extension development

Week's Summary

Days / Time	Feb 17 - Feb 23
Monday	Frontend-backend connection established
Tuesday	API response handling implemented
Wednesday	Response display UI enhanced
Thursday	Editing capabilities added
Friday	Integration testing completed

AMITY UNIVERSITY KOLKATA

Weekly Progress Report

For the Week Commencing: 6th Week

Enrollment No. : A91404822003

NAME: AYUSH GUPTA

Program: BCA

Faculty Guide's Name : Subha Ghosh

Internship Mode : Online

Project Title: GemMail Smart Email Generator & Chrome Browser Extension

Targets for the week: Develop browser extension foundation

Future work plans: Enhance extension functionality

Week's Summary

Days / Time	**Feb 24 - Mar 1**
Monday	Extension environment setup
Tuesday	Gmail DOM research completed
Wednesday	Email context extraction implemented
Thursday	"Generate with GemMail" button added
Friday	Extension-backend communication established

AMITY UNIVERSITY KOLKATA

Weekly Progress Report

For the Week Commencing: 7th Week

Enrollment No. : A91404822003

NAME: AYUSH GUPTA

Program: BCA

Faculty Guide's Name : Subha Ghosh

Internship Mode : Online

Project Title: GemMail Smart Email Generator & Chrome Browser Extension

Targets for the week: Complete extension functionality

Future work plans: Begin user testing

Week's Summary

Days / Time	Mar 2 - Mar 8
Monday	Thread extraction enhanced
Tuesday	Gmail tone selection implemented
Wednesday	Cross-browser testing completed
Thursday	DOM manipulation bugs fixed
Friday	Extension packaged for distribution

AMITY UNIVERSITY KOLKATA

Weekly Progress Report

For the Week Commencing: 8th Week

Enrollment No. : A91404822003

NAME: AYUSH GUPTA

Program: BCA

Faculty Guide's Name : Subha Ghosh

Internship Mode : Online

Project Title: GemMail Smart Email Generator & Chrome Browser Extension

Targets for the week: Gather and implement user feedback

Future work plans: Performance optimization

Week's Summary

Days / Time	**Mar 9 - Mar 15**
Monday	Web app user testing conducted
Tuesday	Extension user testing conducted
Wednesday	Feedback analyzed and prioritized
Thursday	High-priority fixes implemented
Friday	Extension improvements completed

AMITY UNIVERSITY KOLKATA

Weekly Progress Report

For the Week Commencing: 9th Week

Enrollment No. : A91404822003

NAME: AYUSH GUPTA

Program: BCA

Faculty Guide's Name : Subha Ghosh

Internship Mode : Online

Project Title: GemMail Smart Email Generator & Chrome Browser Extension

Targets for the week: Optimize application performance

Future work plans: Implement enhanced features

Week's Summary

Days / Time	**Mar 16 - Mar 22**
Monday	Performance audit completed
Tuesday	React components optimized
Wednesday	Backend performance enhanced
Thursday	Extension performance improved
Friday	Load testing conducted

AMITY UNIVERSITY KOLKATA

Weekly Progress Report

For the Week Commencing: 10th Week

Enrollment No. : A91404822003

NAME: AYUSH GUPTA

Program: BCA

Faculty Guide's Name : Subha Ghosh

Internship Mode : Online

Project Title: GemMail Smart Email Generator & Chrome Browser Extension

Targets for the week: Add enhanced features

Future work plans: Prepare documentation and deployment

Week's Summary

Days / Time	**Mar 23 - Mar 29**
Monday	Additional tone options added
Tuesday	Template saving implemented
Wednesday	Analytics tracking added
Thursday	Gemini API integration refined
Friday	User settings panel created

AMITY UNIVERSITY KOLKATA

Weekly Progress Report

For the Week Commencing: 11th Week

Enrollment No. : A91404822003

NAME: AYUSH GUPTA

Program: BCA

Faculty Guide's Name : Subha Ghosh

Internship Mode : Online

Project Title: GemMail Smart Email Generator & Chrome Browser Extension

Targets for the week: Complete documentation and prepare for deployment

Future work plans: Execute deployment and post-launch activities

Week's Summary

Days / Time	**Mar 30 - Apr 5**
Monday	User documentation completed
Tuesday	Technical documentation finalized
Wednesday	Deployment infrastructure configured
Thursday	Deployment plan finalized
Friday	Final pre-launch review completed

AMITY UNIVERSITY KOLKATA

Weekly Progress Report

For the Week Commencing: 12th Week

Enrollment No. : A91404822003

NAME: AYUSH GUPTA

Program: BCA

Faculty Guide's Name : Subha Ghosh

Internship Mode : Online

Project Title: GemMail Smart Email Generator & Chrome Browser Extension

Targets for the week: Launch product and collect initial feedback

Future work plans: Implement improvements based on user feedback and plan future features

Week's Summary

Days / Time	**Apr 6 - Apr 12**
Monday	Backend services deployed to production
Tuesday	Frontend application launched
Wednesday	Browser extension published to Chrome Web Store
Thursday	Bug fixes and minor improvements deployed
Friday	Future development roadmap drafted based on initial reception