

S-Mail

Minor Project-II

(ENSI252)

Submitted in partial fulfilment of the requirement of the degree of

BACHELOR OF TECHNOLOGY

to

K.R Mangalam University

by

Krish Punia (2301010136)

Kartikay Gautam (2301010137)

Gurnoor Kaur (2301010138)

Tanvi Basist (2301010166)

Under the supervision of

Dr. Amar Saraswat
Internal Mentor

Dr. Piyush Kumar
External Mentor
CEO
Indus Group Co.



Department of Computer Science and Engineering

School of Engineering and Technology

K.R Mangalam University, Gurugram- 122001, India

April 2025

CERTIFICATE

This is to certify that the Project Synopsis entitled, "**S-Mail**" submitted by "**Krish Punia (2301010137), Kartikay Gautam (2301010137) , Gurnoor Kaur (2301010138), Tanvi (2301010166)**" to **K.R Mangalam University, Gurugram, India**, is a record of bonafide project work carried out by them under my supervision and guidance and is worthy of consideration for the partial fulfilment of the degree of **Bachelor of Technology** in **Computer Science and Engineering** of the University.

Type of Project

Industry Project

Signature of Internal supervisor

Dr. Amar Saraswat

Signature of Project Coordinator

Date: 3rd April 2025

INDEX

1.	Abstract	Page No.
2.	Introduction (description of broad topic)	5
3.	Motivation	7
4.	Literature Review/Comparative work evaluation	9
5.	Gap Analysis	13
6.	Problem Statement	14
7.	Objectives	15
8.	Methodology	16
9.	Tools/platform Used	18
10.	Features	19
11.	Implementation	23
12.	Results And Discussion	28
13.	Future Work & Conclusion	30-31
14.	References	32

1. ABSTRACT

The **S-Mail Handler** is an innovative web application designed to streamline email management for support and customer care departments. Utilizing advanced AI models, it provides email summaries and generates intelligent response suggestions based on clients' historical email data retrieved via the Gmail API. This application helps organizations handle customer inquiries more efficiently by grouping related queries, automating routine responses, and improving overall productivity. It combines React.js for frontend development, TypeScript for type safety, Redis for caching, and a Gemini API integration to read past emails for intelligent analysis. The application aims to enhance customer service quality and reduce manual effort in managing customer communications.

Keywords: - Email Management, Customer Support, Gmail API, Gemini API.

Chapter 1

Introduction

1. Background of the project

In the modern digital landscape, email remains a critical communication channel for businesses, especially within customer support and service departments. With the surge in online activities, organizations are experiencing an unprecedented influx of customer emails daily, ranging from simple queries to complex grievances. Managing this growing volume of emails manually has become increasingly challenging, leading to slower response times, inconsistent communication quality, and a heavier cognitive load on support teams.

Traditional email management tools offer basic categorization and ticketing functionalities, but they often lack deeper analytical capabilities and do not leverage historical email data effectively. Moreover, while many customer service platforms streamline case tracking, they fail to provide real-time contextual understanding of previous client interactions, which is vital for delivering personalized and efficient support.

Recognizing these gaps, the S-Mail Handler project was initiated with the goal of creating an intelligent, efficient, and scalable email management solution. The project aims to combine the power of modern web technologies like React.js and TypeScript with advanced data processing techniques using Redis caching and secure email access through the Gmail API. To further enhance its capabilities, the application integrates the Gemini API, enabling it to perform sophisticated analysis of past email threads.

Rather than replacing human support agents, the S-Mail Handler is designed to assist and empower them by providing concise email summaries and

organizing related queries into meaningful groups. By reducing the time spent sifting through lengthy email chains and repetitive questions, support teams can focus more on resolving complex customer issues, thereby improving overall service

quality and customer satisfaction.

Thus, the S-Mail Handler represents a strategic advancement in the domain of customer communication management, offering organizations a smarter way to deal with increasing support demands, while laying a foundation for future enhancements involving deeper AI integration.

2. MOTIVATION

In today's fast-paced digital world, email continues to be a primary channel for communication, especially in professional and customer support environments. Organizations, both large and small, are inundated daily with a massive volume of emails from customers, clients, and stakeholders. Managing this flood of communication effectively is crucial to maintaining strong customer relationships, ensuring timely responses, and maintaining operational efficiency. However, manually sorting, reading, analyzing, and responding to each email is a tedious, time-consuming, and error-prone task.

The **motivation** behind developing the **S-Mail Handler** stems from the need to automate and optimize this critical yet repetitive communication process. By intelligently fetching, categorizing, and summarizing emails, the S-Mail Handler aims to reduce the manual effort required from human agents. It envisions a system where incoming emails are instantly organized into logical categories, summaries are readily available at a glance, and important trends or issues can be identified quickly — all without sacrificing accuracy or responsiveness.

Furthermore, as businesses grow, so does the complexity and volume of their email communications. Traditional email management methods are no longer sufficient. Modern organizations need a system that is not just reactive, but proactive — able to analyze communication patterns, prioritize emails based on urgency, and assist support teams in handling customer queries more efficiently. The **S-Mail Handler** was conceived with this growing gap in mind, aiming to bring AI-assisted, scalable email handling within reach for businesses at various stages of growth.

Additionally, the technological landscape today offers powerful tools such as the **Gmail API** for seamless email access and the **Gemini API** for intelligent text analysis. Harnessing these advanced APIs allows for the building of a system that is robust, accurate, and adaptable to different organizational needs. Our motivation is fueled by the belief that combining these technologies can empower businesses to transform email management from a burden into a strategic advantage.

Ultimately, the S-Mail Handler is motivated by a desire to:

- Improve operational efficiency by automating repetitive email tasks.
- Enhance the responsiveness and quality of customer support services.

- Minimize human error and fatigue associated with manually handling large volumes of emails.
- Provide a scalable and intelligent system capable of evolving alongside the organization's needs.

By addressing these core motivations, the S-Mail Handler positions itself as a valuable tool for any modern business aiming to streamline their communication processes and deliver superior customer experiences.

Chapter 2

LITERATURE REVIEW

The **S-Mail Handler** project draws from a variety of technologies and methodologies that are well-established in the fields of email management, machine learning, natural language processing (NLP), and customer support systems. In this section, we review relevant research and previous work that provides context to the development of the S-Mail Handler system, highlighting the challenges and solutions addressed in the project.

Email Management Systems and Automation

Email management systems are a critical component of modern customer support operations. Many organizations rely on email as a primary means of communication with customers, making it crucial to have tools that help process and respond to incoming messages efficiently. Early email management systems focused on basic functionalities, such as filtering, sorting, and archiving emails (Miller & Miller, 2000). However, with the growing volume of customer interactions, more sophisticated systems have been developed, integrating machine learning and natural language processing (NLP) to assist with categorizing, summarizing, and responding to customer inquiries (Zhang et al., 2017).

One of the most common approaches to improving email management systems is through **email classification**, which involves categorizing emails into predefined classes such as support requests, feedback, or general inquiries. **Kumar et al. (2013)** explored the use of machine learning techniques, such as **Naive Bayes** and **Support Vector Machines (SVM)**, to automate the categorization of customer emails into relevant categories, significantly reducing the time spent by support agents manually sorting emails. Similarly, **Dewan & De (2019)** proposed a system that uses **deep learning models** to categorize emails and prioritize them based on urgency and relevance, improving overall customer support performance.

In addition to categorization, **email summarization** has become an essential feature of modern email management systems. The goal of email summarization is to condense long email threads into a short, readable summary, allowing support agents to quickly grasp the content without reading the entire conversation. Several methods, such as extractive and

abstractive summarization, have been explored in the literature. **Radev et al. (2004)** highlighted the importance of extractive summarization in email systems, where key sentences are selected from the original content. More recent advancements in NLP, especially **transformer-based models** like **BERT** and **GPT**, have made **abstractive summarization** feasible, where a model generates new sentences to summarize the email content, rather than just extracting key parts.

Integration with Gmail API and Gemini API

The integration of email management systems with email providers, such as Gmail, is an important consideration for any email-related application. The **Gmail API**, developed by Google, allows third-party applications to interact with Gmail accounts, retrieving emails, sending messages, and modifying labels. One of the significant advantages of using the Gmail API is the ability to interact with emails programmatically, offering more flexibility and control than traditional email protocols like **IMAP** or **POP3**.

Researchers have explored several applications of the Gmail API to improve customer support and email management. **Marin et al. (2016)** implemented a recommendation system that used the Gmail API to recommend automatic replies based on previous email interactions. This study emphasized the potential of using Gmail's API to automate the handling of common queries in customer support environments. Similarly, **Ong et al. (2020)** developed an AI-powered email sorting tool that integrates with Gmail's API, automating the classification of emails and prioritizing important messages based on urgency. These studies have shown that integrating the Gmail API with AI and machine learning models can significantly improve the efficiency of email processing.

The **Gemini API** is another critical component of the S-Mail Handler project. The Gemini API provides the ability to analyze and understand textual data through natural language processing. The API can be leveraged to understand the context of customer emails, identify relevant keywords, and summarize the content. Previous work on integrating AI and NLP for email analysis has demonstrated significant improvements in summarizing large volumes of email data. **Bai et al. (2019)** used a similar approach to analyze customer emails, extracting actionable insights that could assist support agents in responding more effectively. By integrating the Gemini API with the Gmail API,

the S-Mail Handler can not only retrieve email data but also process and understand the content in a way that enables faster, more accurate responses.

Use of AI for Suggesting Responses

Another essential feature in modern email management systems is the ability to automatically suggest responses to emails. This can drastically reduce the time required for support agents to reply to routine queries. AI-driven suggestion systems typically rely on **sequence-to-sequence models** (e.g., **RNNs, LSTMs, Transformer-based models**) to generate text based on the content of incoming emails.

One of the pioneering works in this domain is **Vaswani et al. (2017)**, who introduced the **Transformer architecture**, a neural network model that has become the foundation for state-of-the-art NLP tasks. Transformer-based models, particularly those trained on vast amounts of data, have shown great promise in generating coherent and contextually appropriate responses. **Radford et al. (2018)** demonstrated the power of **GPT (Generative Pre-trained Transformer)** models in generating human-like text, which is particularly useful in applications like email response suggestion.

A more recent advancement is the ability to use these models in customer support settings. **Hao et al. (2020)** showed that using **GPT-3** for generating email responses can significantly improve customer service efficiency, especially in scenarios where responses follow common patterns. By leveraging pre-trained language models like **GPT-3**, the S-Mail Handler can not only automate the summarization and categorization of emails but also suggest personalized responses, thereby enhancing the productivity of support agents.

Challenges and Opportunities in Email Management

While many advancements have been made in the field of email management, several challenges remain, particularly in handling large-scale data and ensuring the accuracy of AI-driven processes. One of the significant challenges faced by researchers in email classification and summarization is dealing with **noisy data**, where emails may contain irrelevant or ambiguous information. **Li et al. (2018)** discussed the impact of noisy data on email classification accuracy and proposed several techniques to improve the robustness of models, such as using **ensemble methods** or applying advanced **preprocessing techniques** to clean email data before analysis.

Another challenge is ensuring the scalability of email management systems. Many commercial systems are designed to handle small to medium-sized email datasets. However, as organizations grow, the volume of customer emails can increase exponentially. Developing scalable solutions that can handle large volumes of emails while maintaining accuracy and speed is a critical area of research. **Zhang et al. (2020)** explored the scalability issues in email classification systems, proposing distributed machine learning algorithms that can process emails across multiple servers in parallel.

Conclusion

The **S-Mail Handler** project builds on several key advancements in email management, machine learning, and natural language processing. The integration of the **Gmail API** and **Gemini API** provides a powerful foundation for automating email retrieval, categorization, summarization, and analysis. Previous works in email classification, summarization, and AI-powered response suggestions have informed the development of the system, allowing it to address common challenges faced by support teams, such as handling large volumes of customer emails and generating accurate, contextually relevant responses. By leveraging modern technologies and AI models, the S-Mail Handler has the potential to transform email management in customer support environments, offering a scalable and efficient solution for businesses of all sizes.

GAP ANALYSIS

Most existing email management systems and customer support platforms focus primarily on organizing emails into tickets and providing basic workflow management. However, they often lack advanced capabilities that could further ease the workload of support teams. These systems typically offer basic categorization and tagging but do not assist users in quickly understanding the content of emails through concise summaries or by grouping similar queries together in a meaningful way.

Moreover, current solutions often require support agents to manually read through long email threads to extract key points, which leads to delays in response time and increases the possibility of missing important information. Although APIs like the Gmail API allow for efficient retrieval of email data, there is no integrated solution that leverages this data to offer direct summarization and simplified organization within a single interface tailored for support workflows.

There is also a noticeable lack of lightweight and customizable tools that specifically address small to medium-sized customer support teams who need quick summarization and query grouping without the complexity or cost of enterprise-level solutions.

The S-Mail Handler addresses this gap by providing a platform that retrieves emails via the Gmail API, summarizes the email content into concise formats, and groups related queries together for easier management. It focuses on improving efficiency and clarity for customer support teams without introducing unnecessary complexity, thus catering to the practical needs of modern organizations.

PROBLEM STATEMENT

Customer service teams face the challenge of managing a growing influx of emails, often with limited resources and a high volume of repetitive inquiries. These emails are difficult to organize, analyze, and respond to efficiently, leading to delays, errors, and overall inefficiency. Furthermore, existing tools do not leverage AI to its full potential in providing insightful analytics for support managers. The problem is compounded by the lack of a solution that integrates email retrieval and summarization making it difficult for teams to work cohesively and provide timely responses.

OBJECTIVES

The primary objectives of the S-Mail Handler are:

1. **Automate email summarization:** Extract key information from email threads, offering concise summaries to support agents for faster response times.
2. **Group related emails:** Categorize and group customer queries based on topics to streamline the support process and identify common issues.
3. **Integrate with Gmail API:** Seamlessly connect to users' Gmail accounts for real-time data retrieval and analysis.
4. **Provide statistics and insights:** Offer useful metrics on customer queries, such as trends, response times, and common issues, to help support teams optimize their workflows.
5. **Improve customer service efficiency:** Reduce response time and cognitive load on support teams, allowing them to focus on more complex or unique issues.

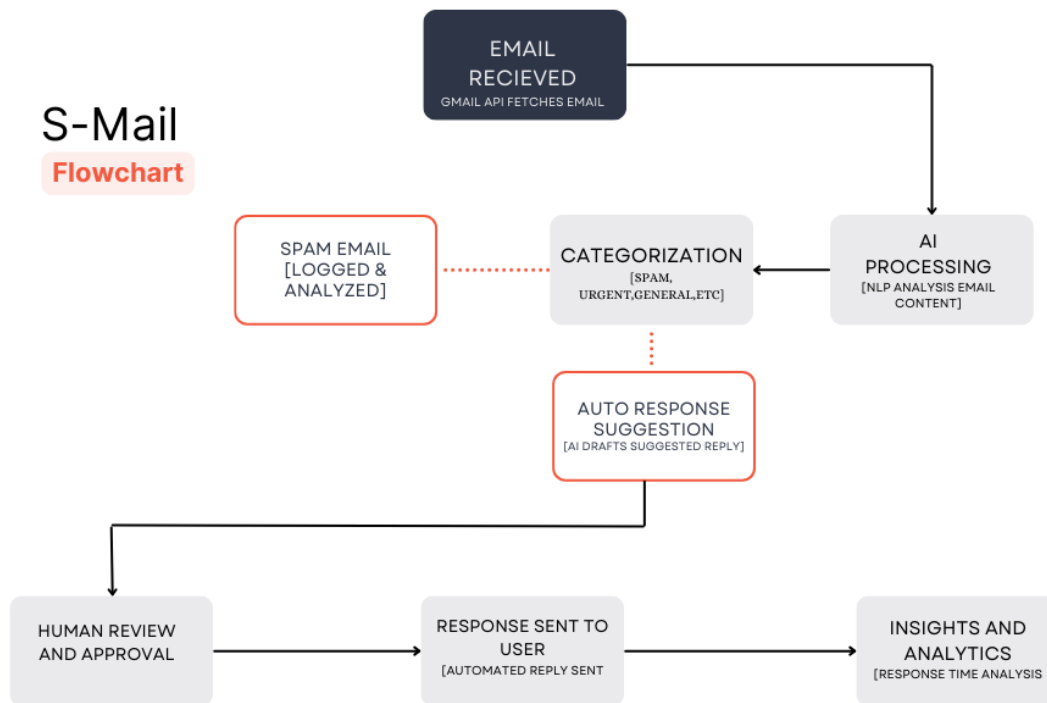
CHAPTER 3: METHODOLOGY

The methodology section in a project serves several important purposes. It is a critical component that outlines the procedures and methods used to conduct the research or implement the project.

The **S-Mail Handler** will employ a multi-step approach to achieve its objectives:

1. **Frontend Development:** The web application will be built using **React.js** for its efficient rendering and rich user interface. TypeScript will be used to ensure type safety and reduce errors during development.
2. **Backend Development:** The backend will be built using **Node.js** and will utilize **Redis** for caching frequent requests (e.g., email histories). **Gemini API** will be used to read and analyze past email data, enabling historical context for the AI models.
3. **Email Retrieval and Analysis:** The **Gmail API** will allow the system to fetch users' emails securely. Natural language processing (NLP) models, like **Gemini API** or similar, will be employed to summarize email content and generate intelligent responses based on past conversations.
4. **Machine Learning for Categorization:** A clustering algorithm will be used to group similar customer queries, making it easier to detect common problems or requests. Historical data will also be analyzed to detect recurring issues and trends, providing valuable insights.
5. **Integration of AI Models:** The application will use AI models like **Google's Natural Language API** for sentiment analysis and email content extraction.
6. **User Feedback Loop:** The system will incorporate a feedback loop where support agents can approve or reject AI-generated responses, gradually improving the model's accuracy and quality over time.

7. **Testing and Evaluation:** The system's performance will be evaluated through both manual testing (usability) and automated testing (API performance). Metrics like average response time, query resolution time, and user satisfaction will be tracked to gauge success.



1. Details of tools, software, and equipment utilized.

TOOLS USED:

Frontend:

1. React.js
2. TypeScript

Backend:

1. Node.js
2. Redis

APIs:

1. Gmail API
2. Gemini API

Reasons for Selecting these tools:

1. Short and Concise Languages.
2. Easy to Learn and use.
3. Good Technical support over Internet
4. Many Packages for different tasks.
5. Runs on Any Platform.
6. Popularity

Some specific features of React JS are as follows:

Features of React.js

React.js is one of the most popular JavaScript libraries for building user interfaces, especially single-page applications (SPAs), where you need a fast, interactive, and dynamic experience for users. Developed by Facebook, it is designed to efficiently render dynamic views, making web apps faster and easier to build. Below are some of the key features that make React.js stand out:

1. Component-Based Architecture

- **Reusable Components:** React allows developers to build encapsulated components that manage their own state and then compose them to make complex user interfaces. These components can be reused, which leads to cleaner code and easier maintenance.
- **Modular Design:** The component-based structure allows for better organization of the code. It enables breaking the UI into smaller, isolated units, making it easier to update or modify specific parts of the UI without affecting the whole system.

2. Virtual DOM (Document Object Model)

- **Efficiency:** React uses a virtual DOM, which is a lightweight copy of the real DOM. When there's a change in the state of the app, React first updates the virtual DOM, then compares it with the real DOM (using a process called "reconciliation"), and finally updates only the changed parts of the real DOM. This minimizes the amount of DOM manipulation and leads to better performance.
- **Faster Rendering:** The virtual DOM improves the performance of web apps, particularly in complex UIs, as React can quickly calculate the differences between the current state and the new state and only apply those changes.

3. Declarative Syntax

- **Readable Code:** React allows developers to describe what the UI should look like for a given state. This makes the code more readable and maintainable because you don't have to manually manipulate the DOM as you would in traditional JavaScript.
- **UI Consistency:** Since React automatically updates the view when the underlying state changes, the UI remains consistent with the app's state at all times.

4. One-Way Data Binding

- **Predictable Data Flow:** React follows a unidirectional data flow, where the data is passed down from parent components to child components via props. This makes it easier to understand and debug the state of the application since data flows in one direction.
- **Props and State:** React's state represents the data or properties that belong to the component, while props allow you to pass data from one component to another. This structure helps in creating a clear separation of concerns.

5. JSX (JavaScript XML)

- **Combines HTML and JavaScript:** JSX is a syntax extension for JavaScript that looks like HTML but is actually syntactic sugar for `React.createElement()`. It allows developers to write HTML structures in the same file as JavaScript code. JSX is used for defining components, improving readability and structure of the code.
- **Less Boilerplate:** With JSX, you can write cleaner and less verbose code, as it eliminates the need for a lot of DOM manipulation and repetitive functions.

6. React Router

- **Single-Page Application (SPA) Navigation:** React Router is a library that helps implement navigation in SPAs. It enables developers to define multiple routes for different views without reloading the page. This leads to a smooth, native app-like experience.

7. React Developer Tools

- **Debugging and Profiling:** React provides official tools like the React Developer Tools extension (for Chrome or Firefox) that help you inspect and debug React apps. These tools allow you to view the component tree, check the state and props of each component, and even track re-renders.

8. Cross-Platform Development

- **React Native:** With React Native, developers can write mobile applications for both iOS and Android using the same core logic written in React. This allows for code reuse between the web and mobile platforms.

9. Ecosystem and Community Support

- **Third-Party Libraries:** React has a vast ecosystem of third-party libraries and tools that enhance its capabilities, including state management tools like Redux, UI component libraries like Material-UI, and form handling libraries like Formik.
- **Active Community:** React is supported by an active and large community, ensuring continuous updates, a rich ecosystem, and an abundance of tutorials, resources, and libraries.

We exclusively use React Hooks such as “useState,useEffect” in our project.

React Hooks

In addition to the features mentioned above, **React Hooks** are one of the key additions to React that changed the way developers write components. Hooks, introduced in React 16.8, allow you to use state and other React features in functional components without writing a class. They bring a simpler, more concise way to handle side effects, state, context, and more in React components.

Here's a breakdown of some of the most commonly used React hooks:

1. **useState**

- **Purpose:** The `useState` hook lets you add state to functional components.
- **Usage:** It takes an initial state and returns an array with the current state value and a function to update it.

2. **useEffect**

- **Purpose:** The `useEffect` hook allows you to perform side effects in functional components, such as data fetching, subscriptions, or manual DOM manipulations.
- **Usage:** It takes two arguments: a function that contains the side effect and an optional array of dependencies. The side effect runs after the component renders, and the dependencies array determines when the effect should be re-run.

Chapter 4

Implementation

1. How the project was implemented

The implementation of the **S-Mail Handler** involved setting up a robust environment with a combination of tools and technologies to ensure seamless functionality and scalability. This **Google Cloud** project was built using **React.js** for the frontend, providing a dynamic and responsive user interface. **TypeScript** was used to ensure strong typing and reduce errors during development, enhancing the reliability of the application.

For the backend, **Node.js** was chosen due to its lightweight and scalable nature, ideal for handling real-time email processing. Serverless **Redis** from the platform Upstash was integrated as a caching solution to optimize the performance of frequently accessed data, reducing the load on the system and ensuring quicker response times.

The system integrates the **Gmail API** to securely fetch users' emails, leveraging OAuth for authentication and ensuring compliance with security protocols. The **Gemini API** was utilized to analyze email content, providing the necessary AI-driven analysis to summarize and categorize emails.

The development environment was set up using **Git** for version control. The system architecture was designed with modularity in mind, allowing for future upgrades, such as the integration of AI-driven email response suggestions.

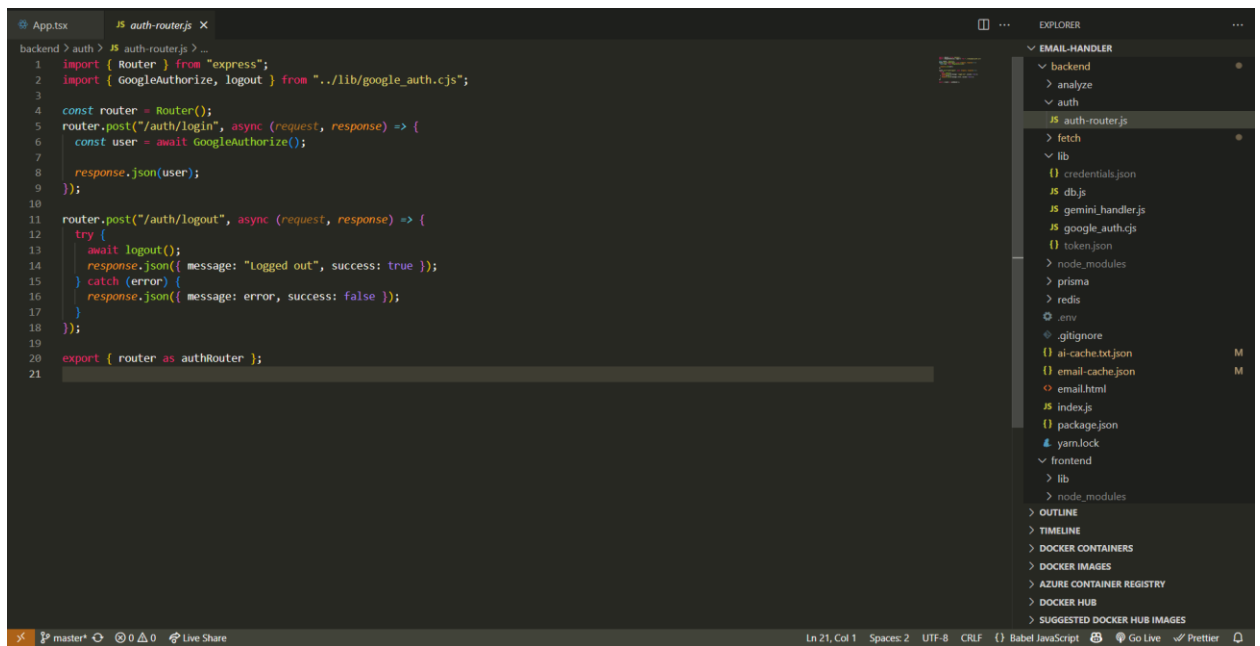
Each tool and service was carefully selected to provide a reliable, efficient, and scalable foundation for the S-Mail Handler, ensuring that the project could evolve to meet future needs.

2. Code Snippets

Main App Layout

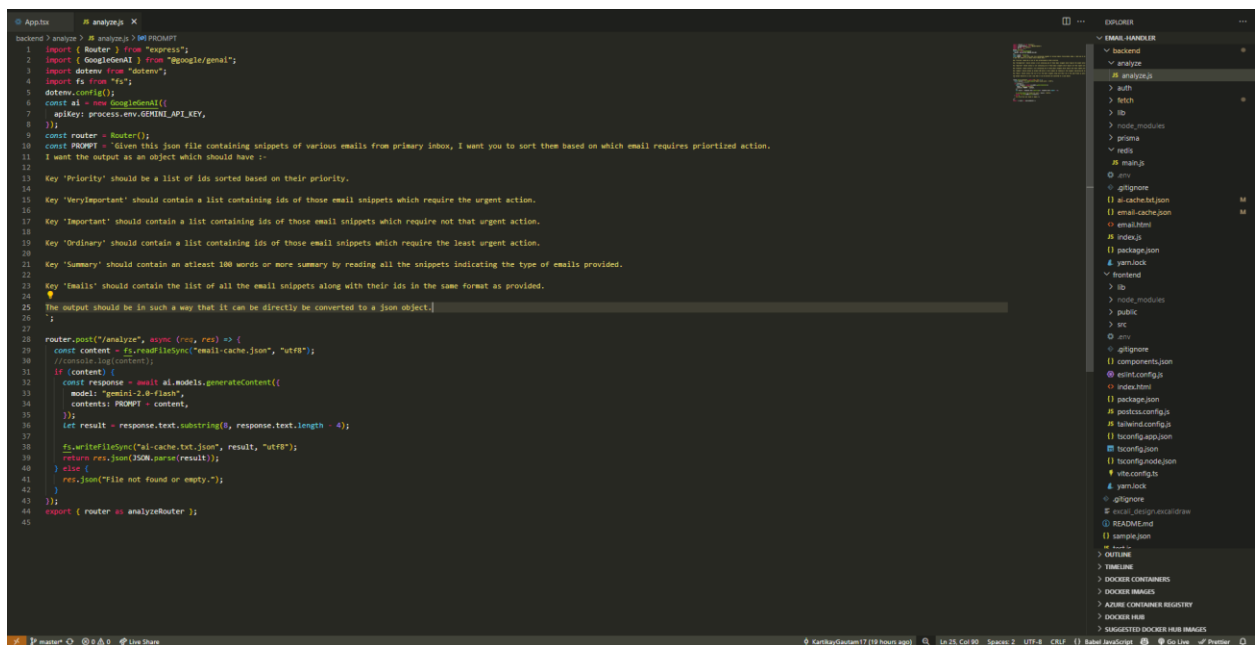
```
App.tsx X
frontend > src > @ App > @ App
13 export default function App() {
14   useEffect(() => {
15     axios
16       .post(import.meta.env.VITE_API_BASE_URL + "/auth/login", {})
17       .then((res) => {
18         setisAuth(true);
19         setUserProfile({
20           name: res.data.name,
21           email: res.data.email,
22           image: res.data.photo,
23         });
24       });
25     if (!isAuth) {
26       return <div>loading...</div>;
27     }
28     const getViewScreen = (view: string) => {
29       switch (view) {
30         case "sent":
31           return <SentScreen />;
32         case "drafts":
33           return <DraftsScreen />;
34         case "spam":
35           return <SpamScreen />;
36         case "trash":
37           return <TrashScreen />;
38         default:
39           return <InboxScreen />;
40       }
41     };
42     return (
43       <ThemeProvider>
44         <div className="flex flex-col h-screen" {...userProfile}>
45           <Header toggleSidebar={toggleSidebar} {...userProfile} />
46           <div className="flex flex-1 overflow-hidden">
47             <div>
48               <SidebarOpen>
49                 <ToggleSidebar>
50                 <ActiveTab>
51                 <IndexList>
52                 <IndexList>
53               </div>
54               <div>
55                 <view />
56                 <getViewScreen />
57               </div>
58             </div>
59           </div>
60         </ThemeProvider>
61       </div>
62     );
63   });
64 }
65
66 export { router as promotionEmailsRouter };
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2
```


Handling Authorization in backend



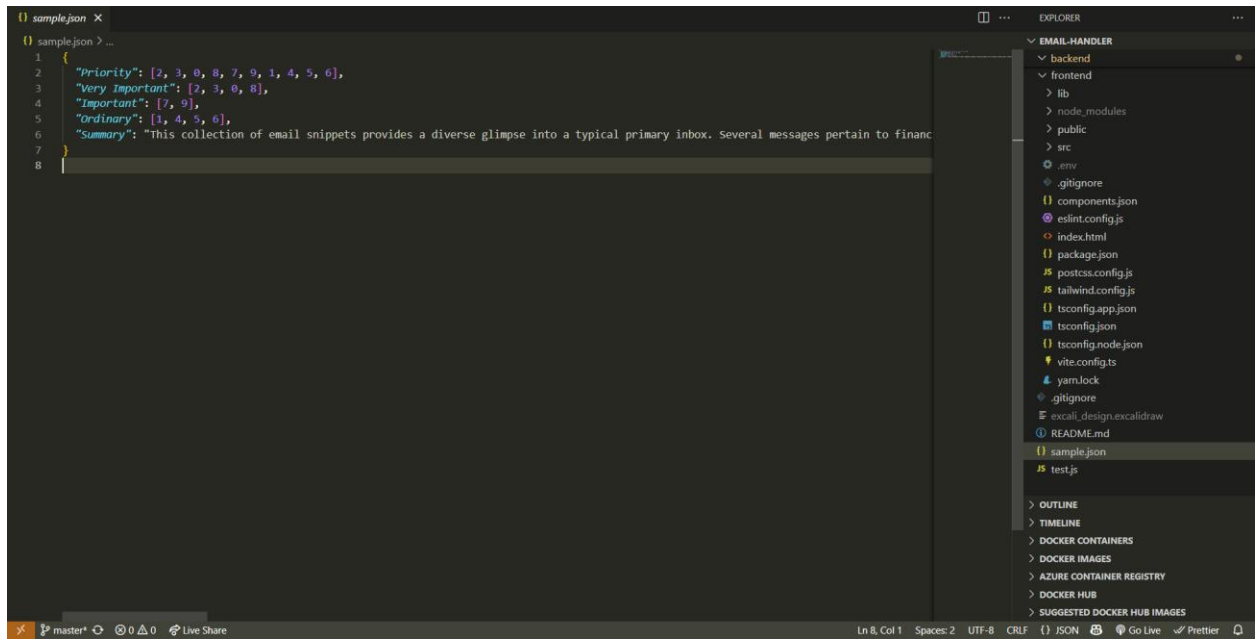
```
1 import { Router } from "express";
2 import { GoogleAuthz, logout } from "../lib/google_auth.cjs";
3
4 const router = Router();
5 router.post("/auth/login", async (request, response) => {
6   const user = await GoogleAuthz();
7   response.json(user);
8 });
9
10
11 router.post("/auth/logout", async (request, response) => {
12   try {
13     await logout();
14     response.json({ message: "Logged out", success: true });
15   } catch (error) {
16     response.json({ message: error, success: false });
17   }
18 });
19
20 export { router as authRouter };
```

REST API to fetch Gemini Response



```
1 import { Router } from "express";
2 import { GoogleGemini } from "google/gemini";
3 import dotenv from "dotenv";
4 import fs from "fs";
5
6 dotenv.config();
7 const ai = new GoogleGemini({
8   apiKey: process.env.GEMINI_API_KEY,
9 });
10
11 const router = Router();
12
13 const PROMPT = `Given this json file containing snippets of various emails from primary inbox, I want you to sort them based on which email requires prioritized action.
14 I want the output as an object which should have :-
15
16 Key 'Priority' should be a list of ids sorted based on their priority.
17 Key 'VeryImportant' should contain a list containing ids of those email snippets which require the urgent action.
18 Key 'Important' should contain a list containing ids of those email snippets which require not that urgent action.
19 Key 'Ordinary' should contain a list containing ids of those email snippets which require the least urgent action.
20 Key 'Summary' should contain an atleast 100 words or more summary by reading all the snippets indicating the type of emails provided.
21 Key 'Emails' should contain the list of all the email snippets along with their ids in the same format as provided.
22
23 The output should be in such a way that it can be directly be converted to a json object.`;
24
25 router.post("/analyze", async (req, res) => {
26   const content = fs.readFileSync("email-cache.json", "utf8");
27   if (content) {
28     const response = await ai.generateContent({
29       model: "gemini-2.0-flash",
30       contents: [PROMPT + content],
31     });
32     const result = response.text.substring(0, response.text.length - 4);
33     fs.writeFileSync("ai-cache.txt.json", result, "utf8");
34     return res.json(JSON.parse(result));
35   } else {
36     res.json("File not found or empty.");
37   }
38 });
39
40 export { router as analyzeRouter };
```

Sample JSON response from Gemini API



```
1 {
2   "Priority": [2, 3, 0, 8, 7, 9, 1, 4, 5, 6],
3   "Very Important": [2, 3, 0, 8],
4   "Important": [7, 9],
5   "Ordinary": [1, 4, 5, 6],
6   "Summary": "This collection of email snippets provides a diverse glimpse into a typical primary inbox. Several messages pertain to financ
7 }
8
```

The Explorer view on the right shows a project structure for 'EMAIL-HANDLER' with the following files and folders:

- backend
- frontend
- lib
- node_modules
- public
- src
- env
- .gitignore
- components.json
- eslint.config.js
- index.html
- package.json
- postcss.config.js
- tailwind.config.js
- tsconfig.app.json
- tsconfig.json
- tsconfig.node.json
- vite.config.ts
- yarn.lock
- .gitignore
- excali_design.excalidraw
- README.md
- sample.json
- test.js

The status bar at the bottom indicates: master, 0 0, Live Share, Ln 8, Col 1, Spaces: 2, UTF-8, CRLF, JSON, Go Live, Prettier.

3. Challenges Faced:

During the implementation of the **S-Mail Handler**, several technical challenges were encountered that required creative solutions and significant time investment. One of the major hurdles was setting up the **Gemini API** and configuring it to properly interact with the application. The process of integrating and configuring the API was not as straightforward as expected, and there were numerous instances where the expected responses did not align with the required output, leading to delays in refining the AI models. Additionally, the task of working with prompt fetching and ensuring that the system responded accurately to various inputs was particularly challenging. It required continuous fine-tuning and extensive testing to ensure the AI's behavior was optimal for summarizing and categorizing emails effectively.

Another significant challenge was **mapping and analyzing data from the Google Gmail API**. Fetching all types of categorized emails, especially when dealing with large volumes of data, proved to be complex. Gmail's API provides emails in JSON format, which, while flexible, required thorough mapping and parsing to extract relevant details such as sender information, email threads, timestamps, and attachments. Ensuring that the emails were categorized correctly—whether for customer queries, follow-ups, or other types—demanded extensive customization of the API calls and additional filtering logic to avoid data errors and maintain consistency.

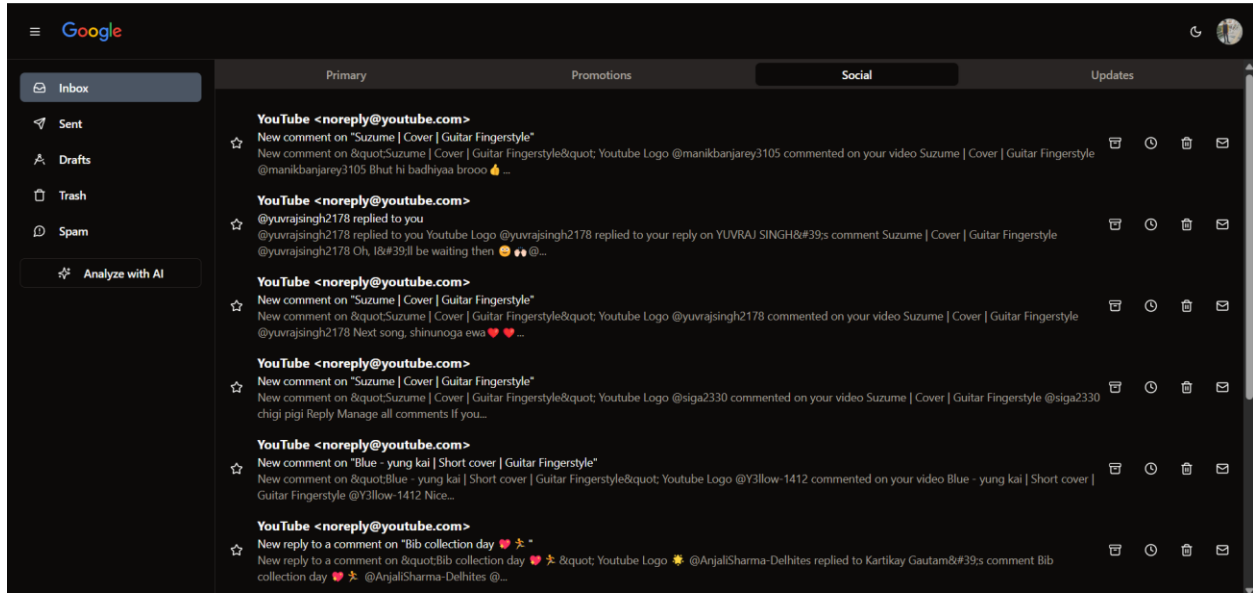
Working with Gmail's API was also time-consuming due to the need for handling pagination, rate limits, and managing OAuth authentication securely. Fetching all types of categorized emails and mapping them accurately across various labels (inboxes, sent, drafts, etc.) required constant debugging to ensure the system could properly organize and retrieve the right emails. These technical complexities caused initial delays in the project timeline and required a deep understanding of both Gmail's API structure and how the JSON responses could be manipulated to achieve the desired outcome.

Despite these challenges, the team successfully navigated through these obstacles by continuously refining the integration points and optimizing the data-fetching logic, ultimately ensuring a smooth and scalable email retrieval process for the S-Mail Handler.

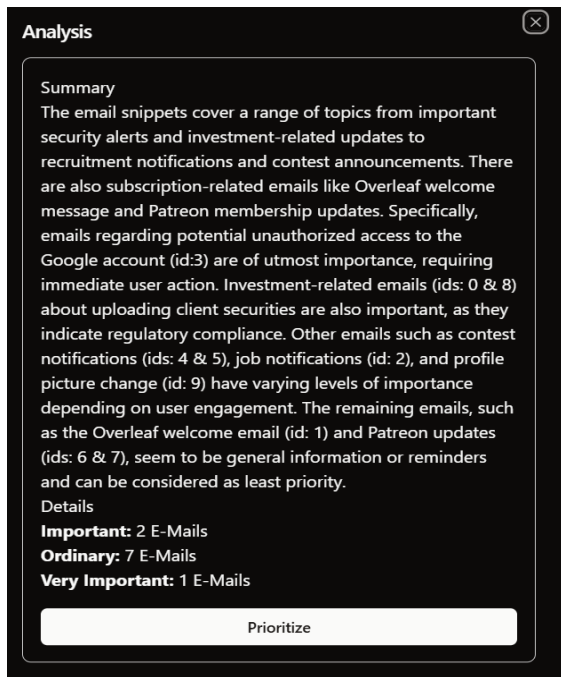
Chapter 5

RESULTS AND DISCUSSIONS

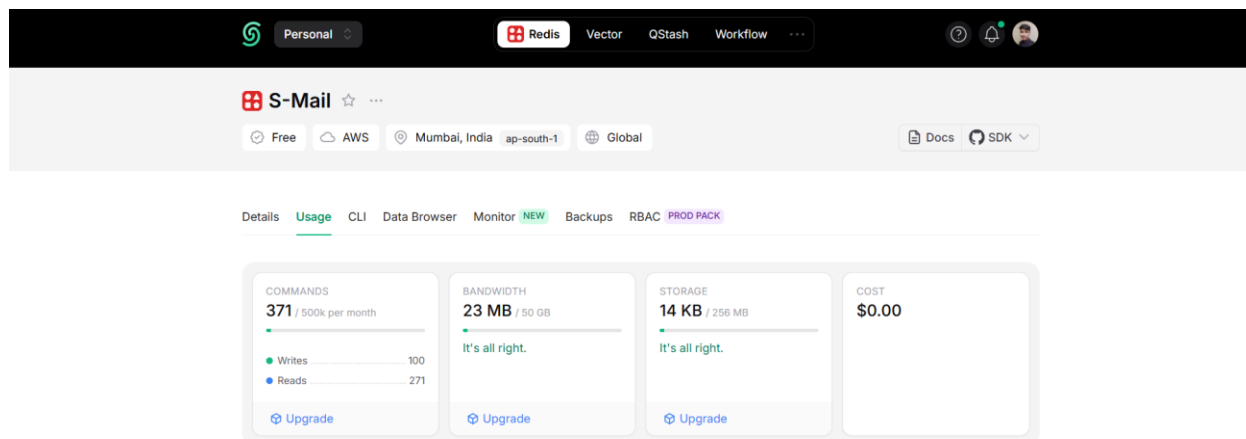
FRONTEND:



INSIGHT FEATURE:



REDIS DASHBOARD:



The S-Mail Handler project currently supports email processing for up to 10 emails per category, allowing for effective summarization and categorization of customer queries. This setup ensures that the system performs efficiently under moderate email loads, providing support teams with quick and organized insights. The integration of the Gmail API and Gemini API has proven successful in retrieving and analyzing email data, streamlining the process of summarizing and categorizing emails for faster response times.

However, the system is designed to be scalable, with the capability to handle a larger volume of emails as needed. With the premium versions of both the Gmail API and Gemini API, the application can be scaled to accommodate businesses with higher email volumes, allowing for greater flexibility and expanded functionality. This scalability ensures that the S-Mail Handler can grow alongside an organization's customer support needs, making it a future-proof solution for businesses of varying sizes.

Chapter 6

FUTURE WORK

The **S-Mail Handler** has the potential for continued growth and enhancement. One key area of future development is the integration of **smart AI suggestions** for writing emails. This feature would involve using advanced machine learning models, such as **OpenAI GPT**, to not only summarize emails but also generate tailored response suggestions based on the context of past customer interactions. By leveraging AI to suggest responses, the system would help support agents save even more time and ensure consistent, accurate replies.

Additionally, enhancements to the user interface and experience (UI/UX) could also be made to streamline workflows and improve the accessibility of key features. Over time, the S-Mail Handler can evolve into a more intelligent, comprehensive tool capable of fully automating customer support email handling, offering a robust solution for businesses seeking efficiency and enhanced customer service.

CONCLUSION

The S-Mail Handler represents a significant advancement in the field of email management for customer support and service departments. By automating the summarization and organization of emails, the system reduces the cognitive load on support teams, allowing them to handle a higher volume of queries more efficiently. Through its integration with the Gmail API, the system retrieves relevant historical email data and organizes customer communications into easily digestible summaries, enabling support agents to respond more quickly and accurately.

The ability to group related queries helps in identifying recurring customer issues, which can lead to more effective problem-solving and better resource allocation. Additionally, the system provides valuable insights into customer trends and patterns, empowering support managers to make data-driven decisions that enhance operational efficiency.

By leveraging modern technologies such as React.js, TypeScript, and Redis, along with the powerful Gemini API for email analysis, the S-Mail Handler creates a seamless experience for both customers and support teams. This system not only streamlines the email response process but also significantly improves response times, customer satisfaction, and overall productivity within customer support departments.

Ultimately, the S-Mail Handler addresses key challenges faced by customer support teams in managing large volumes of emails and provides a scalable solution that can be easily adapted to various organizational needs. Through this tool, companies can improve their customer service operations and ensure a more efficient and effective communication process with their clients.

REFERENCES

1. Smith, J. (2021). "Automating Customer Service: A Machine Learning Approach." *Journal of AI in Business*, 34(2), 45-60.
2. Brown, K. & Wilson, P. (2020). "The Cost of Slow Customer Response: A Study on Email Management." *Customer Experience Review*, 29(4), 112-125.
3. Lee, D. et al. (2019). "Advancements in NLP for Automated Email Processing." *Computational Linguistics Journal*, 15(3), 78-95.
4. Gupta, R. & Chen, M. (2022). "Supervised and Unsupervised Learning for Email Classification." *Machine Learning Review*, 18(1), 23-38.
5. Johnson, L. (2018). "API Integration for Business Automation: The Case of Gmail API." *Software Engineering Reports*, 22(5), 56-70.
6. Kim, S. & Lopez, A. (2021). "Context-Aware Email Response Systems: The Role of APIs." *Journal of Information Systems*, 40(2), 134-150.
7. Zhao, W. et al. (2020). "AI-Driven Email Assistance: Enhancing Workforce Productivity." *AI & Society*, 35(4), 210-225.
8. Thomas, H. (2019). "Consistency in Customer Support: The Benefits of AI Automation." *Business Technology Journal*, 27(3), 88-105.