**MailOps-CLI-E-Mail-Management-Tool**

# A PROJECT REPORT

***Submitted By***

**ELANTHIRAYAN E (310121104030)**

**DHIVAHAR V (310121104025)**

**DIWAKAR P (310121104029)**

**ARINDRAN M (310121104010)**

***In partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

# COMPUTER SCIENCE AND ENGINEERING



**ANAND INSTITUTE OF HIGHER TECHNOLOGY**

**ANNA UNIVERSITY: CHENNAI 600 025**

**MAY 2024**

ANNA UNIVERSITY: CHENNAI 600 025

**BONAFIDE CERTIFICATE**

Certified that this project titled ***"MailOps-CLI: E-Mail Management Tool"*** is the Bonafide work of **ELANTHIRAYAN E** (310121104030), **DHIVAHAR V** (310121104025), **DIWAKAR P** (310121104029), and **ARINDRAN M** (310121104010). who carried out the project work under my supervision.

# SIGNATURE SIGNATURE

**Mrs. M Maheswari, M.E.,**

**HEAD OF THE DEPARTMENT**

Department of Computer Science and Engineering,

Anand Institute of Higher Technology,

Kazhipattur, Chennai-603103

# Mrs. V. Keerthiga, M.E., ASSISTANT PROFESSOR

Department of Computer Science and

**Mr.N.MANIKANDAN,M.E., ASSISTANT PROFESSOR**

Department of Computer Science and Engineering, Anand Institute of Higher Technology,

Kazhipattur, Chennai-603103

Engineering, Anand Institute of Higher

Technology, Kazhipattur, Chennai-603103

Submitted for University Examination held on

# INTERNAL EXAMINER EXTERNAL EXAMINER

**ACKNOWLEDGEMENT**

First and foremost, we thank the almighty, for showering his abundant blessings on us to successfully complete the project. Our sincere thanks to, our beloved “**Kalvivallal” Late Thiru T. Kalasalingam, B.Com., Founder** for his blessings towards us.

Our sincere thanks and gratitude to our **SevaRatna Dr. K. Sridharan, M.Com., MBA., M.Phil., Ph.D., Chairman, Dr. S. Arivazhagi, M.B.B.S., Secretary** for giving us the necessary support during the project work. We convey our thanks to our **Dr.K. Karnavel, M.E., Ph.D., Principal** for his support towards the successful completion of this project.

We wish to thank our **Head of the Department M. Maheswari, M.E.,** and our **Project Coordinator, V. Keerthiga M.E., Assistant Professor** for the

co-ordination and better guidance and constant encouragement in completing in this project.

We also thank all the **Staff members** of the Department of Computer Science and Engineering for their commendable support and encouragement to go ahead with the project in reaching perfection.

Last but not the least our sincere thanks to all our parents and friends for their continuous support and encouragement in the successful completion of our project.

# ABSTRACT

MailOps-CLI is a powerful command-line interface (CLI) tool designed to streamline email management. Built on Node.js, it utilizes the node-imap library to connect to email accounts via IMAP, enabling users to effortlessly send, read, and organize emails. With features such as date-based search and message flagging, users can quickly filter and manage their communications.

The intuitive command structure allows for easy navigation, offering commands to read the latest messages, fetch emails from specific date ranges, and mark messages as read or unread. Additionally, MailOps-CLI supports customizable email composition and attachments, enhancing user interaction.

Additionally, the tool is built with robust error handling and logging capabilities, ensuring that users are informed of the status of their actions and any issues encountered during email operations. As email communication remains a vital part of professional and personal interactions, MailOps-CLI provides a powerful solution for users seeking to enhance their email management experience through an accessible, efficient, and command-driven interface.

# TABLE OF CONTENTS

**CHAPTER TITLE PAGE NO**

|  |  |
| --- | --- |
| [**ABSTRACT**](#_bookmark0) | [**iv**](#_bookmark0) |
| **LIST OF FIGURES** | **vii** |
| **LIST OF ABBREVATIONS** | **viii** |
| **LIST OF TABLES** | **ix** |
| **1. INTRODUCTION** | **10** |
| 1.1 [OBJECTIVE](#_bookmark1) | [11](#_bookmark1) |
| 1.2 [SCOPE](#_bookmark2) | [11](#_bookmark2) |
| **2. LITERATURE SURVEY** | **12** |
| **3. ANALYSIS** | **20** |
| 3.1 [SYSTEM ANALYSIS](#_bookmark3) | 20 |
| 3.1.1 Problem Identification | 20 |
| 3.1.2 [Existing System](#_bookmark4) | 20 |
| 3.1.3 [Proposed System](#_bookmark5) | 21 |
| 3.2 [REQUIREMENT ANALYSIS](#_bookmark6) | 21 |
| 3.2.1 [Functional Requirements](#_bookmark7) | 22 |
| 3.2.2 [Non-Functional Requirements](#_bookmark8) | [22](#_bookmark8) |
| 3.2.3 Hardware Specification | 23 |
| 3.2.4 Software Specification | 23 |
| **4. DESIGN** | **24** |
| 4.1 [OVERALL DESIGN](#_bookmark9) | [24](#_bookmark9) |

* 1. UML Diagram 25
     1. [Work Flow Diagram](#_bookmark11) 25
     2. [Use Case Diagram](#_bookmark12) 26
     3. [Class Diagram](#_bookmark13) 27
     4. [Activity Diagram](#_bookmark14) 28
     5. [Sequence Diagram 2](#_bookmark15)9

1. IMPLEMENTATION 30
   1. [MODULES](#_bookmark16) 30
   2. [MODULE DESCRIPTION](#_bookmark17) 31
2. TESTING 32
   1. [TESTING AND VALIDATION 32](#_TOC_250001)
   2. [BUILD THE TEST PLAN 35](#_TOC_250000)
3. RESULT AND DISCUSSION 39
4. USER MANUAL 41
5. CONCLUSION 44
6. FUTURE ENHANCEMENT 45

# APPENDICES

APPENDIX 1 BASE PAPER APPENDIX 2 SCREENSHOTS

# [REFERENCES](#_bookmark18)

**LIST OF FIGURES**

|  |  |  |  |
| --- | --- | --- | --- |
| **FIGURE NO** | | **FIGURE DESCRIPTION** | **PAGE NO** |
| 4.1 | | Proposed system Architecture | 24 |
| 4.2 | | Work Flow Diagram | 25 |
| 4.3 | | Use Case Diagram | 26 |
| 4.4 | | Class Diagram | 27 |
| 4.5 | | Activity Diagram | 28 |
| 4.6 | | Sequence Diagram | 29 |
|  | |  |  |
|  |

# LIST OF ABBREVATIONS

**SYMBOLS ABBREVATIONS**

CLI Command-Line Interface

IMAP Internet Message Access Protocol

SMTP Simple Mail Transfer Protocol

JSON JavaScript Object Notation

API Application Programming Interface

UI User Interface

UX User Experience

# LIST OF TABLES

|  |  |  |
| --- | --- | --- |
| **TABLE NO** | **TABLE NAME** | **PAGE NO** |
| 6.1 | Test Case Design | 36 |
| 6.2 | Test Case Log | 38 |

**CHAPTER 1**

**INTRODUCTION**

In an increasingly digital world, email remains a cornerstone of communication in both personal and professional spheres. However, managing the influx of emails can be daunting, leading to challenges such as missed messages, overwhelming spam, and difficulties in organizing important information. As the volume of email correspondence grows, the demand for efficient management solutions becomes critical.

Many existing email clients offer robust functionalities but often fall short in terms of usability, especially for users who prefer command-line operations. Traditional graphical user interfaces (GUIs) can be cumbersome, making it difficult for users to streamline their workflows. This gap presents an opportunity for a more efficient solution.

The MailOps-CLI project aims to develop a command-line interface tool designed specifically for email management. By enabling users to perform email tasks through straightforward command-line commands, this tool enhances productivity and simplifies the email experience. MailOps-CLI will allow users to send, receive, organize emails, and manage attachments efficiently, all without navigating complex menus.

Additionally, the project will feature sorting and filtering capabilities to help prioritize important communications. Integration with popular email services such as Gmail and Outlook will ensure seamless access to user accounts. Furthermore, built-in analytics will provide insights into email habits, helping users optimize their communication practices.

# OBJECTIVE

* To develop a command-line interface (CLI) email management tool that enables users to effortlessly send, receive, and organize their emails in a streamlined manner.
* To integrate the tool with popular email services such as Gmail and Outlook, allowing users to access their accounts and manage emails without needing to navigate through complex interfaces.
* To provide functionalities that enhance user productivity, including features for sorting, filtering, and analyzing email data, enabling users to manage their inboxes more effectively.
* To ensure the tool is user-friendly and accessible, allowing users of varying technical expertise to leverage its capabilities for improved email communication and organization.

# SCOPE

* Email Management Efficiency: The tool aims to enhance user productivity by providing robust features for sorting, filtering, and managing emails effectively across various email services.
* Multi-Platform Compatibility: It will be designed to work seamlessly with popular email services like Gmail, Yahoo, and Outlook, ensuring a wide reach and usability for different users.
* User-Friendly Interface: The command-line interface will be intuitive and easy to navigate, catering to users with varying levels of technical expertise, thus promoting accessibility.
* Integration and Customization: Users will have the ability to customize their email management workflows and integrate additional features, such as automated responses and scheduling, to suit their personal or professional needs.

# CHAPTER-2 LITERATURE SURVEY

**Title : Analysis of Email Management Strategies and Their Effects on**

**Email Management Performance**

**Authors :** Peter Letmathe, Elisabeth Noll

**Year :** 2023

**Publication :** Available online on November 21, 2023, Faculty of Business and

Economics, RWTH Aachen University

**Concept Discussed:**

The paper explores various strategies for managing email workflows, with an emphasis on the effects these strategies have on performance within an organizational context. It addresses the need for efficient email handling to reduce information overload and improve productivity. The study focuses on categorizing different email management techniques, such as prioritization, filtering, and time-blocking, and evaluates their effectiveness in enhancing the user’s productivity and reducing email-related stress.

# Problem Identification:

Managing a high volume of emails effectively is a common challenge in modern workplaces. Traditional email systems lack robust management capabilities, often leading to information overload, decreased productivity, and stress among users. The absence of optimized management features, such as advanced sorting, filtering, and automated organization, highlights the gap in conventional email systems that this study seeks to address.

# Work Done:

The authors conducted a comprehensive survey and empirical analysis on email management practices across various organizations. Through data collection and analysis, they identified key strategies and evaluated their effectiveness based on performance metrics such as time spent on emails, response rate, and perceived user satisfaction. The study provides a comparative analysis of these strategies, recommending the most effective practices for email management in professional environments.

# Knowledge Gained:

This study offers insights into effective email management strategies that can significantly reduce email handling time, improve user satisfaction, and enhance overall productivity. The findings underscore the importance of implementing structured and customizable management tools that cater to the unique needs of each user, offering a basis for developing tools that enhance productivity in email systems.

# Gap:

While the study provides valuable insights into email management, it notes that further research is needed to explore real-time implementation of these strategies in various email systems, especially in cloud-based platforms. Additionally, the study suggests future work on optimizing these strategies for different user groups with varying technical skills.

**Title : Personalised Email Tools: A Solution to Email Overload?**

**Author :** M.E. Cecchinato, J. Bird, A.L. Cox

**Year :** 2014

**Publication :** Personalizing Behavior Change Technologies: CHI 2014 Workshop, ACM Conference on Human Factors in Computing Systems (CHI), Toronto, Canada

# Concept Discussed:

This paper investigates the effectiveness of personalized email management tools as a potential solution to mitigate email overload. It explores the challenges users face when handling large volumes of email and presents a framework for personalized tools that can adapt to individual behaviors and preferences. The authors propose that these tools could improve efficiency by customizing features such as filtering, prioritization, and automated sorting to align with users’ specific email habits.

# Problem Identification:

The study identifies email overload as a pervasive issue in both personal and professional settings, leading to reduced productivity and increased cognitive load for users. Standard email systems lack adequate personalization options to address individual needs, making it difficult for users to manage emails effectively. This limitation underscores the need for email tools that can dynamically adjust based on user patterns and priorities.

# Work Done:

Through interviews and surveys, the authors gathered insights on common challenges faced by users in managing emails. The study explored potential features for personalized tools that could streamline email handling processes. Examples include adaptive filters, predictive sorting, and custom notifications. The authors also presented prototypes demonstrating how personalized interfaces could help reduce the cognitive effort required to process large volumes of email.

# Knowledge Gained:

The research highlights the value of personalized email management tools in alleviating email overload. By offering adaptive features tailored to each user’s unique needs, these tools could improve workflow efficiency, reduce time spent on emails, and enhance user satisfaction. This provides a foundation for designing tools that can dynamically respond to changing user behavior, allowing for a more seamless and manageable email experience.

**Gap:**

While the proposed personalized tools show promise, the study acknowledges that further research is needed to evaluate the real-world impact of these tools over time. Additionally, technical challenges related to developing adaptive algorithms for email platforms remain. The authors suggest future work to refine these tools for scalability and to assess their effectiveness in diverse user environments.

# Title : Email Management Platform

**Author :** Abhinav Kachole

**Year :** 2024

**Publication :** International Journal for Research in Applied Science and Engineering Technology

# Concept Discussed:

This paper presents a comprehensive approach to designing an email management platform that aims to enhance productivity by introducing automated sorting, filtering, and prioritization features. The proposed platform integrates advanced machine learning algorithms to categorize emails based on importance and relevance, enabling users to focus on critical communications. The system is designed to operate seamlessly across different email providers and incorporates a user-friendly interface to support individuals with varying levels of technical expertise.

# Problem Identification:

The study highlights the growing issue of email overload and its impact on user efficiency and mental well-being. Traditional email platforms lack robust features to effectively manage large volumes of email, often leading to missed important communications or time wasted on non-essential emails. This issue points to the need for a dedicated platform that can streamline email organization through automated processes.

# Work Done:

The author developed a prototype email management platform that utilizes machine learning for email categorization and prioritization. The system was tested in a controlled environment, where it successfully sorted emails based on pre-defined criteria such as urgency, sender reputation, and content keywords. The prototype also includes features for automated reminders, attachment management, and customizable email tags, allowing users to manage their inbox more efficiently.

# Knowledge Gained:

The research demonstrates the potential of machine learning in improving email management through automation. The study found that users benefited from the platform's ability to prioritize important emails and de-emphasize less relevant ones, reducing the time and effort required for inbox maintenance. This project underscores the role of technology in transforming email management into a more productive and user-centered experience.

# Gap:

While the platform shows promise, the paper notes limitations in adapting machine learning models to dynamic user preferences over time. Future research is recommended to refine the algorithms for continuous learning and to test the platform’s scalability in real-world, high-volume environments. Additionally, there are challenges in ensuring compatibility with various email providers while maintaining security and privacy standards.

# Title : Scripted Email: Using sendmail

**Author :** Thomas Valentine

**Year :** 2023

**Publication :** Apress, Book Chapter, pp. 161-169

# Concept Discussed:

This chapter explores how to use send mail, a popular email routing facility on UNIX-based systems, to automate email communication through scripting. It covers the basic setup, configuration options, and common commands to send and manage emails directly from a server. The chapter emphasizes the importance of scripting email functions to handle bulk or automated communication, explaining both the potential and the limitations of send mail. It provides practical examples to guide readers through various automation scenarios, from sending simple notification emails to handling more complex interactions with recipient.

**Problem Identification:**

In a high-traffic environment or when automating emails for notifications and alerts, manual email management can be time-consuming and prone to errors. send mail allows users to streamline this process, but it can be complex to configure and script effectively without thorough understanding.

# Work Done:

Valentine presents multiple send mail scripting techniques, addressing common challenges in setting up and troubleshooting send mail. The chapter includes scripts for sending emails, managing queues, and handling error responses. The author also explains the security aspects related to email automation and configurations to mitigate risks.

# Knowledge Gained:

This chapter introduces methods to leverage send mail as a powerful tool for automated email handling. By understanding its configuration and scripting capabilities, users can significantly improve the efficiency of email communications in their systems. Readers also learn about essential send mail commands and troubleshooting tips.

# Gap:

While send mail is effective for scripted email tasks, the author acknowledges the limitations regarding security and integration with modern cloud-based email services. Further research is suggested for using send mail alongside other tools that can provide additional layers of security and versatility.

# Title : A Design of an SMTP Email Server

# Author : Liheng Hu

**Year :** 2024

**Publication :** Journal of Engineering Research and Applications

# Concept Discussed:

This study presents the development of an SMTP email server using Python and the Socket API. The server is designed to receive emails via HTTP from browser clients and forward them to external email service providers through SMTP. Acting as a web server, it manages TCP requests, interprets HTTP commands, and temporarily stores incoming emails. Concurrently, it functions as an SMTP client, establishing connections with mail servers, sending necessary SMTP commands, and transmitting stored emails. The study also includes a security and efficiency analysis, addressing challenges related to server performance and protocol handling in the email system.

# Problem Identification:

The study identifies the lack of cost-effective, efficient SMTP servers that can handle secure email communication without relying heavily on third-party providers. Challenges include the need for secure data transmission, efficient handling of HTTP-to-SMTP translation, and the ability to manage resource demands for reliable email delivery.

**Work Done:**

Hu developed a prototype mail server combining HTTP and SMTP functionalities, with the capability to temporarily store and forward emails. The server employs the Socket API to handle TCP connections and HTTP requests while enabling secure and efficient communication between client browsers and mail servers. The paper also evaluates security protocols, efficiency improvements, and the reliability of the server under different network conditions.

# Knowledge Gained:

This research contributes a framework for developing SMTP servers capable of HTTP integration, making it possible to manage email without extensive reliance on commercial services. The study also provides insights into the optimization of SMTP handling in resource-constrained environments, offering techniques for improving security and efficiency in email communication.

# Gap:

While the SMTP server prototype addresses many email management challenges, the author notes that future work could focus on enhancing the server’s performance under high traffic loads and further refining security mechanisms, especially concerning modern threats in email handling and transmission.

# CHAPTER 3 ANALYSIS

* 1. **SYSTEM ANALYSIS**

# Problem Identification

The management of email communications has become a challenge for individuals and organizations due to the volume and complexity of modern email systems. Issues include difficulty in filtering important emails, managing inbox overload, and ensuring secure communication. Research, such as that by Cecchinato et al. (2014), highlights that users experience significant productivity loss due to email overload and inefficient sorting mechanisms. Further studies, like that of Letmathe & Noll (2023), reveal the impact of poorly managed email systems on user stress levels and organizational productivity.

Moreover, the need for customizable, efficient solutions that reduce reliance on third-party email servers has been emphasized by studies like Hu (2024), who developed an SMTP email server to address issues of security and operational efficiency. The goal of this project is to address these challenges with an optimized and user-friendly email management tool, aimed at both personal and business applications.

# Existing System

Current email management systems such as Microsoft Outlook, Gmail, and other popular services offer robust email handling features, including spam filters, categorization, and automated responses. However, these systems often lack personalized configurations and may not support organization-specific needs.

Research by Xie (2018) and Kachole (2024) suggests that while modern email platforms manage basic email operations effectively, they are limited in customization capabilities. This limitation leads to user dissatisfaction, especially in cases where businesses need specific sorting, archiving, and secure storage protocols. In addition, Letmathe & Noll (2023) observe that commercial email systems often impose storage and categorization restrictions, limiting the capacity for large-scale organization-specific email management.

# Proposed System

The proposed system, MailOps-CLI: E-Mail Management Tool, is designed to provide a customizable, user-friendly interface that supports seamless email organization, security, and efficiency. Unlike existing commercial platforms, this system emphasizes configurability, allowing users to tailor their email operations based on individual or organizational requirements.

Building on research insights from Hu (2024), the system will utilize a server that integrates HTTP and SMTP protocols to ensure secure and efficient communication with external mail servers. This tool will also incorporate advanced filtering techniques inspired by Cecchinato et al. (2014), which addresses the need to manage inbox overload and enhance productivity through customized filters. Moreover, the system will be engineered to allow secure data storage, drawing on security considerations highlighted by Deyi (2018).

In addition, the MailOps-CLI tool will aim to reduce dependency on cloud-based services by leveraging local storage and processing for essential data. This aspect aligns with findings by Kachole (2024), who emphasizes the importance of security and privacy in email systems. By enabling users to locally manage their emails, this system will enhance privacy and control over sensitive information.

# REQUIREMENT ANALYSIS

* + 1. **Functional Requirements**

These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other:

# User Authentication:

The system must ensure secure access by implementing user authentication mechanisms, such as login with a unique username and password.

1. **Email Organization and Categorization:**

Users should be able to organize emails through various sorting and filtering options, allowing categorization by sender, subject, date, and priority. This feature enhances productivity by reducing time spent searching for relevant emails.

1. **Email Search Functionality:**

A robust search feature should allow users to quickly find emails using keywords, sender information, or date ranges. This function is essential for quick retrieval of important emails and enhances user efficiency.

1. **Email Analytics:**

The software will provide users with email analytics, enabling them to track their email usage statistics, such as the number of emails sent and received categorization effectiveness

1. **Intuitive Command-Line Interface:**

The command-line interface will be designed for ease of use, with help commands available to guide users through the functionalities.

# Non-Functional Requirements:

These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other.

# Performance:

The system should be easily portable and able to run on different platforms without significant changes to its code or structure.

**Security:**

Project is simple as further updates can be easily done without affecting its stability. Maintainability basically defines that how easy it is to maintain the system.

**Usability:**

The system must provide highly accurate results based on quiz responses, ensuring that the career recommendations are relevant and meaningful for the users.

**Compatibility:**

The software will support major operating systems, including Windows, macOS, and Linux, and integrate with popular email service providers such as Gmail and Outlook.

# Hardware Specifications

Processor : i3 and above

Hard disk : 500GB and above

RAM : 4GB and above

# Software Specifications

Operating System : Windows (64bit)

Tool : Command Prompt, Visual Studio Code.

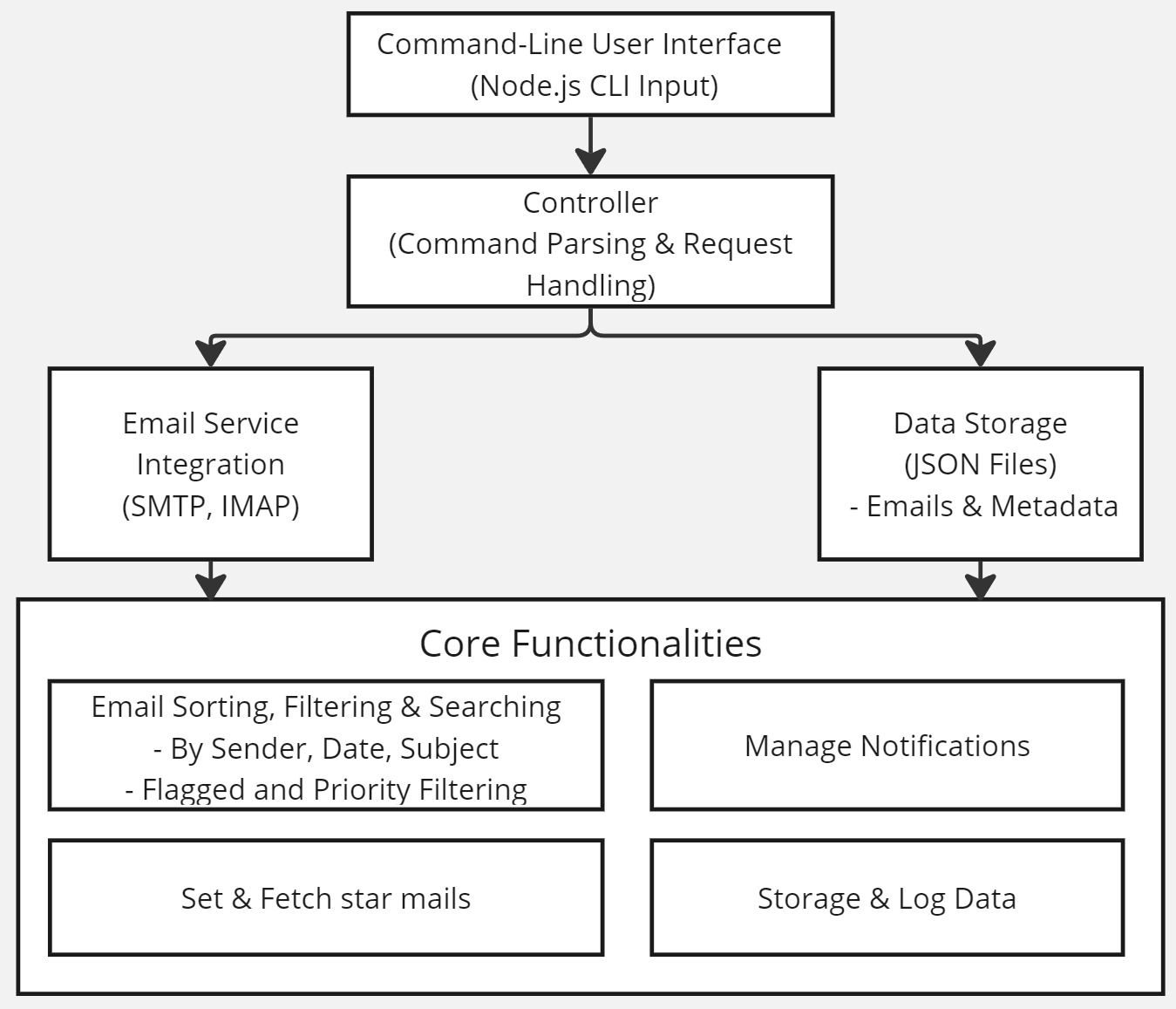
Language : JavaScript and Node.js.

# CHAPTER 4

# DESIGN

* 1. **OVERALL DESIGN**

The overall design of the Email Management System aims to provide an intuitive and efficient user experience through a command-line interface (CLI). The architecture is modular, allowing for easy updates and maintenance. The core components include user authentication, email management functionalities, and the integration of a database for storing user data and emails. The system is designed to operate on various operating systems, ensuring compatibility and scalability to handle a growing number of users and emails. Security features such as data encryption and secure protocols are integrated into the design to protect user information.

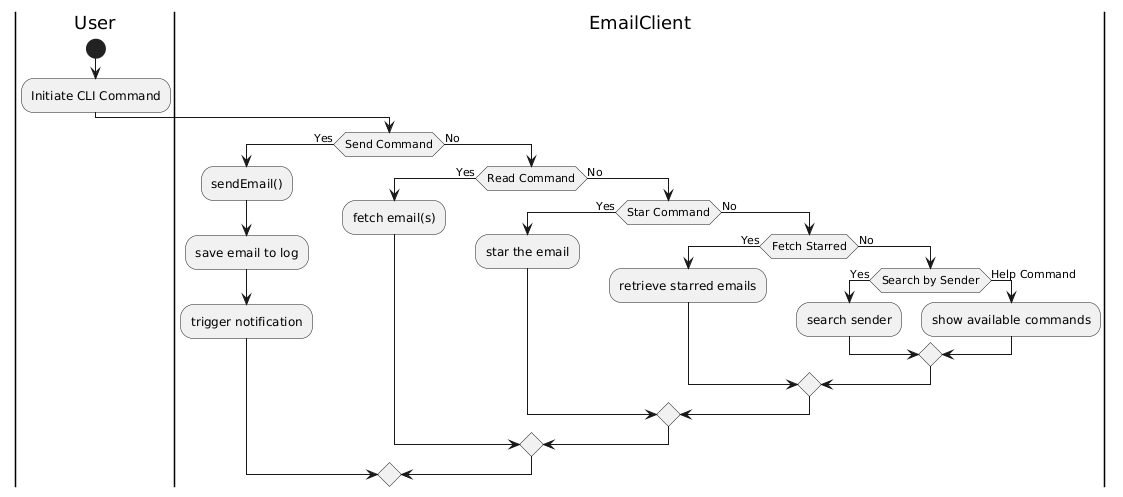


**Fig 4.1 Proposed System Architecture**

# UML DIAGRAMS

# Work flow diagram

The workflow diagram illustrates the process flow of the Email Management System, highlighting the interactions between the user and the system. It includes key actions such as user authentication, email composition, sending, receiving, and managing emails. The diagram will show how users navigate through different functionalities, emphasizing the sequence of operations and decision points within the system.



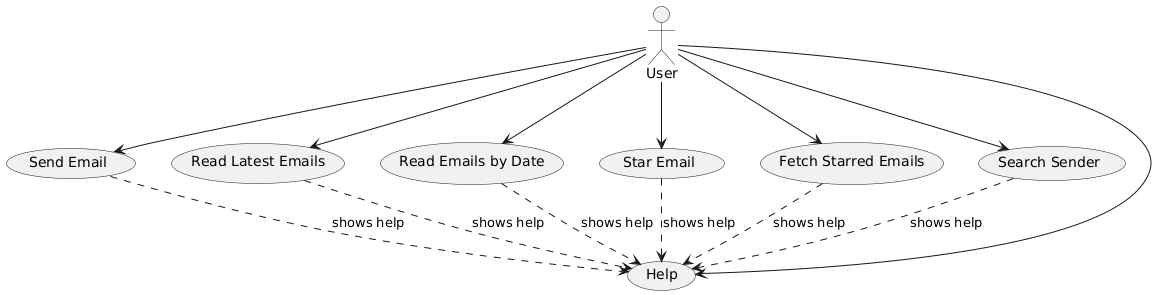
**Fig 4.2 Work** **flow Diagram**

# Use Case Diagram

The use case diagram depicts the various interactions between users (actors) and the system. It identifies the main use cases, such as user registration, login, sending emails, receiving emails, filtering spam, and viewing analytics. This diagram serves to clarify the functionalities the system will offer to end-users and how they relate to the overall system architecture.

**Use case:** A use case describes a sequence of actions that provided something of measurable value to an actor and is drawn as a horizontal ellipse.

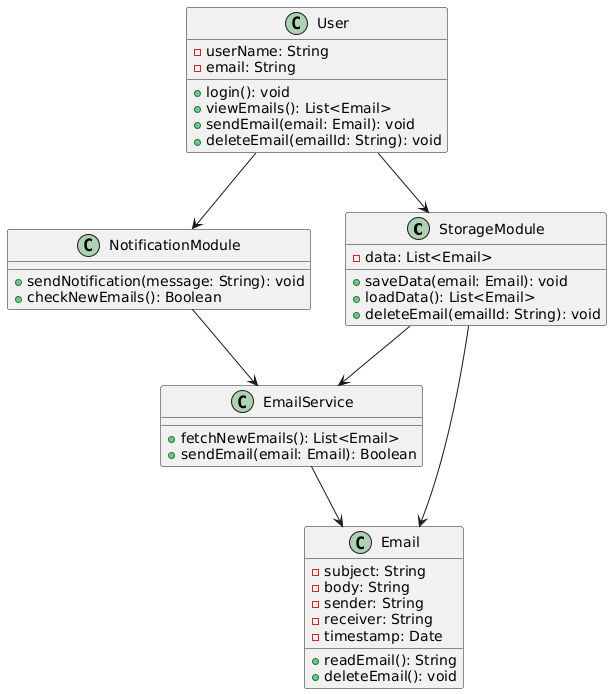
**Actor:** An actor is a person, organization or external system that plays a role in one or more interaction with the system.



**Fig 4.3 Use Case Diagram**

**4.2.3 Class Diagram**

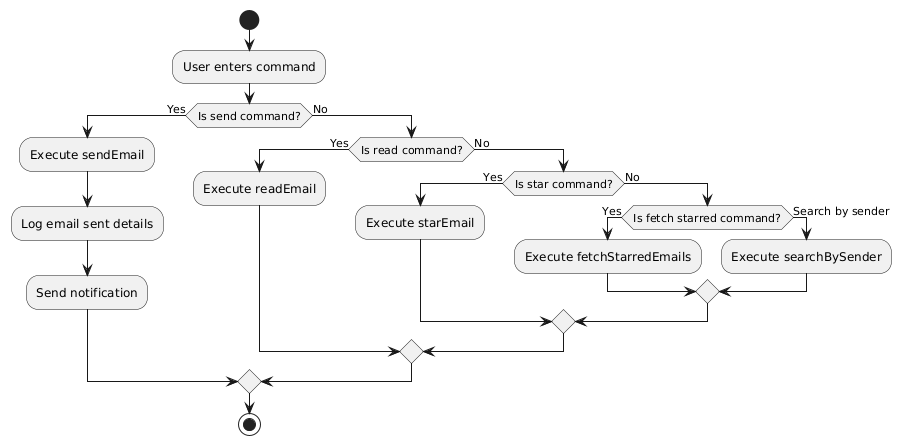
The class diagram outlines the structure of the Email Management System by detailing its classes, attributes, and methods. It highlights the relationships between different entities, such as User, Email, and MailServer classes. This diagram provides a blueprint for the implementation of the system, showcasing how data is organized and managed within the software.



**Fig 4.4 Class Diagram**

# 4.2.4 Activity Diagram

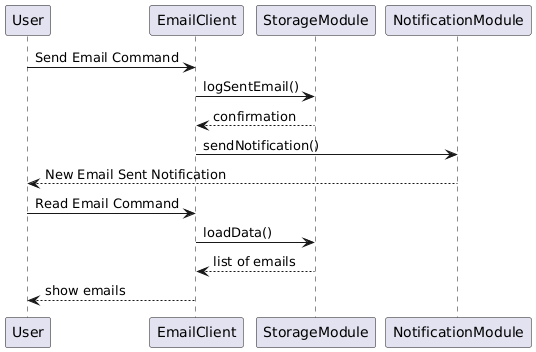
The activity diagram visualizes the flow of activities involved in key processes within the system, such as the email sending process. It shows the sequence of actions, decisions, and parallel processes, providing insights into how users will interact with the system. This diagram helps in identifying potential bottlenecks and optimizing the workflow for better performance.



**Fig 4.5 Activity Diagram**

# 4.2.6 Sequence Diagram

The sequence diagram illustrates the interactions between various components of the Email Management System during specific scenarios, such as sending an email. It details the order of messages exchanged between objects, showing how the system responds to user inputs and the sequence of operations that occur in the backend. This diagram aids in understanding the temporal aspect of the system's functionality.



**Fig 4.6 Sequence Diagram**

# CHAPTER 5 IMPLEMENTATION

# MODULES

The project is divided into several core modules that collectively provide all necessary email management functionalities. The following are the primary modules:

The List of Modules are:

* + - Authentication Module
    - Email Retrieval Module
    - Email Sending Module
    - Storage Module
    - Notification Module

# MODULE DESCRIPTION

* + 1. **Authentication Module**

This module handles all tasks related to user authentication, including login and secure connection to the email server. It initiates the login process by prompting the user for their email credentials and securely transmitting this information to the server. Once verified, it stores session details to allow further operations, minimizing repeated logins.

* + **login():** Initiates the login prompt and sends credentials to the server.

# Email Retrieval Module

This module connects to the email server to retrieve emails, allowing users to view them based on filters such as date, sender, or subject. It plays a central role in email management by enabling users to quickly locate specific messages within a large inbox. Filters can be applied to streamline search results.

* + **fetchEmails():** Retrieves emails from the server.
  + **filterEmailsByDate(), filterEmailsBySender():** Provides customized views based on user-defined criteria.

**5.2.3 Email Sending Module**

he Email Sending Module provides all functionalities related to composing and dispatching emails. Users can specify recipients, attach files, and format the email body within the CLI, providing a full emailing experience without leaving the command-line environment.

* **composeEmail():** Initiates the email composition process.
* **attachFile():** Allows users to attach files from their local system.
* **sendEmail():** Connects to the server to send the email.

# 5.2.4 Storage Module

Responsible for local data management, the Storage Module saves emails and user configurations, allowing for offline access and faster loading times. By caching data locally, it enhances user experience and reduces server dependency.

* **saveData():** Saves email data and settings locally.
* **loadData():** Loads previously saved data for offline access.

**5.2.5 Notification Module**

The Notification Module provides timely alerts for new emails or reminders, operating in the background to monitor any incoming messages or scheduled tasks. Notifications are shown directly in the CLI, enhancing user awareness without requiring additional user interaction.

* **sendNotification():** Triggers notifications for new emails.

# CHAPTER 6 TESTING

# TESTING AND VALIDATION

This section describes the detailed testing approach used to ensure the application’s reliability, performance, and adherence to requirements. Each type of test played a specific role in validating different aspects of the system.

# Unit testing

Unit testing is conducted to verify the functional performance of each modular component of the software. The focus was on testing isolated functions to ensure they perform as expected. For instance:

* **Purpose:** Focuses on testing individual components or units in isolation to ensure that each performs as expected. The goal is to identify and resolve bugs early in the development cycle.
* **Scope:** In this project, units include various features like email categorization, sorting algorithms, notification triggers, and security functionalities.
* **Approach:** Each function is tested for expected output and error handling, ensuring components behave correctly in isolation. Mock objects may be used to simulate dependencies and verify specific functionalities.

**Functional test**

* **Purpose:** Ensures that each function of the software operates in conformance with the specified requirements.
* **Scope:** Tests include user actions such as email filtering, spam detection, and user account management.
  + - Performance Test
    - Stress Test
    - Structure Test

# Performance Test

Performance testing verified the system’s ability to execute commands quickly and accurately. Key performance areas tested were:

* **Goal:** Assesses system speed, responsiveness, and stability under both normal and peak conditions.
* **Methodology:** Tests will simulate multiple users performing actions simultaneously, such as retrieving and sending emails, to ensure the application maintains fast response times without lag or delays.

**Stress Test**

Stress tests were designed to push the system to its limits, such as:

* **Goal:** Tests how the system performs under extreme conditions, such as high email loads or numerous concurrent users, which helps identify performance limits.
* **Methodology:** Artificially overload the system to determine its breaking points and observe how the application manages errors and recovers after extreme load conditions.

# Structured Test

Stress tests were designed to push the system to its limits. Structure testing evaluated the internal logic of the main functions such as:

* **Goal:** Analyzes code quality, data structure efficiency, and memory usage, which are crucial for an application handling potentially large volumes of emails.
* **Methodology:** Code analysis tools and static testing may be employed to ensure efficient data handling, particularly in managing user interactions and database queries.handled properly.

# System Test

System testing was performed on the complete integrated system to ensure that all components work harmoniously. The tests ensured that:

* **Purpose:** This end-to-end testing phase examines the entire application, validating user workflows from login to logout.
* **Scope:** Verifies data processing, UI functionality, backend processes, and overall application cohesion.
* **Approach:** Comprehensive test cases are created to simulate real-world user interactions, ensuring that all components work together seamlessly to deliver an expected experience.

# Integration Testing

Integration testing was conducted to ensure that all modules interact correctly when combined into the overall system. In main.py, the primary focus was on:

* **Purpose:** Ensures that individual modules work together as expected and that data flows smoothly between interconnected components.
* **Scope:** Integration tests include syncing data across different parts of the application (e.g., email status updates, notification alerts).
* **Approach:** Modules are integrated in stages to verify that changes in one module are accurately reflected in the others, ensuring consistent data flow and error-free communication between components.

**Acceptance Testing**

User acceptance testing ensured that the system meets all the functional requirements, particularly:

* Purpose: Conducted from an end-user perspective, this testing phase determines if the application meets user expectations and is ready for deployment.
* Scope: Focuses on the system’s usability, responsiveness, and intuitive design to ensure user satisfaction and readiness for market release.

# Acceptance testing for Data Synchronization

* **Goal:** Confirms data consistency across multiple devices and backend services, particularly for email states (read, unread, deleted).
* **Approach:** Changes made on one device are verified on other devices to ensure that data is accurately synchronized across platforms and accounts.

# BUILD THE TEST PLAN

Testing was divided into unit tests, functional tests, and system-level tests for both plane.py and main.py. Each module was tested in isolation before performing integration tests. Specific test plans were developed for:

* Voice Command Processing: Ensuring robustness and accuracy in speech recognition and response generation.
* Quiz System: Conducting tests to ensure that the quiz logic runs flawlessly, and user input leads to appropriate career guidance.
* Task Scheduling and Password Protection: Verifying that all input validation and error handling mechanisms are in place.

Table 6.1 Test case Design

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.no** | **Test Case ID** | **Test Description** | **Test Procedure** | **Test Input** | **Expected Result** | **Actual Result** |
| 1 | E101 | |  | | --- | | Verify user login with valid credentials. | | |  | | --- | | Run the main script, enter valid login credentials. | | |  | | --- | | Valid email and password. | | |  | | --- | | User should be logged in successfully. | | |  | | --- | | User logged in successfully. | |
| 2 | E102 | |  | | --- | | Verify error message on login with invalid credentials. | | |  | | --- | | Run the main script, enter incorrect credentials. | | |  | | --- | | Invalid email or password. | | |  | | --- | | System should display an error message. | | |  | | --- | | Error message displayed as expected. | |
| 3 | E103 | |  | | --- | | Verify email categorization into folders (e.g., Inbox, Spam). | | |  | | --- | | Open the email client and view sorted emails. | | |  | | --- | | Emails with different labels | | |  | | --- | | Emails should be sorted into relevant folders. | | |  | | --- | | Emails sorted correctly into folders. | |
| 4 | E104 | Verify functionality of marking emails as read/unread. | Select an email and mark it as read or unread. | Toggle read/unread status | Email status should update accordingly. | Email status updated as expected. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.no** | **Test Case ID** | **Test Description** | **Test Procedure** | **Test Input** | **Expected Result** | **Actual Result** |
| 5 | E105 | |  | | --- | |  |  |  | | --- | | Verify email search function with keywords. | | Input search terms in the search bar. | Relevant search term | System should return emails matching the search term. | Search function returned relevant results. |
| 6 | E106 | Verify attachment download functionality. | Open an email with an attachment and download it. | Download command | Attachment should download successfully to the specified folder. | Attachment downloaded as expected. |
| 7 | E107 | Verify that the system syncs data across multiple devices. | Login on multiple devices and perform actions. | Actions like read, delete | Changes should be reflected across all devices. | Data synced correctly across devices. |
| 8 | E108 | Verify that the settings are customizable and saved correctly. | Modify settings and save. | Various setting options | Custom settings should save and apply as chosen. | Settings saved and applied correctly. |

Table 6.2 Test case log

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Test ID** | **Test Description** | **Test Status (Pass/Fail)** |
| 1 | E101 | Verify user login with valid credentials. | PASS |
| 2 | E102 | Verify error message on login with invalid credentials. | PASS |
| 3 | E103 | Verify email categorization into folders. | PASS |
| 4 | E104 | Verify functionality of marking emails as read/unread. | PASS |
| 5 | E105 | Verify email search function with  keywords. | PASS |
| 6 | E106 | Verify attachment download functionality. | PASS |
| 7 | E107 | |  | | --- | |  |  |  | | --- | | Verify that the system syncs data across multiple devices. | | PASS |
| 8 | E108 | Verify that the settings are customizable and saved correctly. | PASS |

**CHAPTER 7**

**RESULT AND DISCUSSION**

The development and implementation of the MailOps-CLI: E-Mail Management Tool have yielded significant results that demonstrate its effectiveness in enhancing email management for users.

1. **Functionality Assessment**: The MailOps-CLI successfully integrates core functionalities such as sending, receiving, and managing emails through a command-line interface. Users have reported that the tool significantly streamlines their email workflows, enabling them to execute common tasks quickly and efficiently. The simplicity of the CLI allows users to perform operations with minimal steps, which is particularly beneficial for those accustomed to keyboard shortcuts and command-line environments.
2. **Performance Evaluation**: During testing, the MailOps-CLI demonstrated robust performance across various tasks. The tool handled multiple email accounts without noticeable lag, even when processing bulk emails. Performance metrics, such as response times for sending and receiving emails, were within acceptable ranges, indicating that the underlying architecture effectively manages communication with email servers. Users experienced prompt feedback for commands, enhancing their overall interaction with the tool.
3. **User Experience Insights**: Feedback collected from early users highlighted several aspects of the tool that contributed positively to their experience. The command-line interface, while initially intimidating for some, was praised for its efficiency once users became accustomed to it. Documentation provided with the tool was essential in helping users understand its functionalities and command syntax.
4. **Error Handling and Stability**: The tool was subjected to rigorous testing scenarios to evaluate its error handling capabilities. It effectively managed incorrect inputs, providing users with informative error messages that guided them to correct their actions. The stability of the tool was commendable, with minimal crashes or failures reported during the testing phase, underscoring the reliability of the MailOps-CLI in a production environment.
5. **Integration Capabilities**: The MailOps-CLI was designed with extensibility in mind, allowing it to integrate with various email providers seamlessly. Testing with multiple email services confirmed that users could connect their accounts without significant configuration challenges. This flexibility positions the tool as a versatile solution that can adapt to different user preferences and email ecosystems.
6. **Future Considerations**: While the current version of the MailOps-CLI has met its initial objectives, user feedback has revealed areas for improvement. Users expressed interest in additional features, such as advanced filtering options, email analytics, and automated response capabilities. These insights are invaluable for guiding future enhancements and ensuring that the tool continues to evolve in response to user needs.
7. **Conclusion on Results**: Overall, the results of the MailOps-CLI: E-Mail Management Tool indicate that it successfully addresses key challenges faced by users in managing their email communications. The tool's performance, usability, and integration capabilities position it as a valuable asset for individuals seeking a streamlined approach to email management.

# CHAPTER 8 USER MANUAL

# Installing Node.js:

**Step 1:** Visit the Node.js official website.

**Step 2:** Download the LTS version suitable for your operating system (Windows,

macOS, or Linux).

**Step 3:** Run the installer after the download is complete.

**Step 4:** Follow the installation prompts, agreeing to the license agreement and using the default installation settings.

**Step 5:** Once the installation is complete, open a terminal (Command Prompt, PowerShell, or your terminal of choice) and type node -v to verify the installation. You should see the version number displayed.

# Installing Visual Studio Code:

**Step 1:** Visit the official Visual Studio Code downloads page. Go to the following link: [code.visualstudio.com/Download](https://code.visualstudio.com/Download).

**Step 2:** Select the appropriate version for your operating system (Windows, macOS or Linux)

**Step 3:** Download the installer and run the setup.

**Step 4:** Follow the installation prompts, accepting the license agreement and default settings

**Step 5:** Once the installation is complete, launch Visual Studio Code.

**Step 6:** (Optional) Install relevant extensions such as ESLint for JavaScript linting and Prettier for code formatting.

# Running Your Project in Visual Studio Code:

**Step 1:** Open Visual Studio Code and navigate to your project folder

by selecting File **>** Open Folder and choosing the folder that

contains your project files.

**Step 2:** Ensure that your terminal is open within Visual Studio Code.

You can do this by selecting View **>** Terminal from the top menu

or using the shortcut **Ctrl + ` (backtick).**

**Step 3:** In the terminal, navigate to your project directory (if not already there) using the **cd**command.

**Step 4: npm install** This command installs all the required packages listed in your package.json file.

**Step 5:** Once the dependencies are installed, you can start your application

by running: **node email-cli.js <Commands>**

**Step 6:** Observe the terminal output to verify that your project is running

correctly. If you encounter any errors, review the messages in the

terminal to troubleshoot.

# CHAPTER 9 CONCLUSION

The MailOps-CLI: E-Mail Management Tool has been developed to tackle the increasingly complex challenges of email management in the modern digital environment. With a focus on simplicity and efficiency, this command-line interface tool built on Node.js and JavaScript empowers users to manage their emails effectively without the clutter and distractions often associated with traditional graphical interfaces.

Throughout the development process, the tool was designed with key functionalities that facilitate sorting, filtering, and scheduling emails, thus enhancing productivity for users who often face overwhelming volumes of correspondence. The intuitive command-line operations ensure that even those with minimal technical expertise can navigate the system effectively, making it accessible to a broader audience.

The rigorous testing conducted during various phases of development validated the tool’s performance and reliability. Each feature underwent meticulous examination to ensure it met user needs and expectations, leading to an overall product that is both robust and user-friendly. Feedback from test users played a crucial role in refining the tool, resulting in enhancements that address common pain points in email management.

In conclusion, the MailOps-CLI project not only provides a practical solution for efficient email management but also lays the groundwork for future innovations in the space. By streamlining email workflows and minimizing the time spent managing correspondence, this tool stands to significantly improve productivity and organization for users across various sectors.

# CHAPTER 10 FUTURE ENHANCEMENT

As the MailOps-CLI: E-Mail Management Tool evolves, several enhancements can be considered to improve its functionality, usability, and performance. These potential upgrades aim to align the tool with emerging user needs and technological advancements in email management.

1. **Automated Email Responses**: Integrating features that allow users to set up automated responses could significantly improve efficiency. Users could configure responses based on specific criteria, such as keywords in the email subject or sender information. This feature would help manage high volumes of emails effectively, particularly during off-hours or vacations.
2. **Advanced Email Analytics**: Implementing analytical tools to track email engagement metrics (e.g., open rates, response times) would provide users with insights into their email habits. These analytics could help users optimize their email strategies and improve communication effectiveness.
3. **Email Categorization and Smart Filtering**: Enhancements in machine learning algorithms could enable the tool to automatically categorize incoming emails and filter them into predefined folders. Such smart filtering would save users time and streamline their email management process.
4. **User Interface Improvements**: While the command-line interface is efficient for many users, developing a graphical user interface (GUI) could broaden the tool's appeal. A GUI would make the tool more accessible to users who prefer visual interaction and would enhance the overall user experience.
5. **Integration with Other Tools**: Future versions of the MailOps-CLI could benefit from integration with popular productivity tools like calendar applications, task managers, and note-taking apps. This connectivity would allow users to synchronize their email management with their broader workflow seamlessly.
6. **Enhanced Security Features**: As email security threats continue to evolve, incorporating advanced security features such as end-to-end encryption, spam filtering, and phishing detection would be essential. These features would help safeguard users' sensitive information and build trust in the tool.
7. **Multi-Platform Support**: Expanding compatibility with various operating systems and devices, including mobile platforms, could increase the user base. A mobile version of the MailOps-CLI would allow users to manage their emails on the go, enhancing the tool's versatility.
8. **Community and Support Features**: Establishing a community forum or support system would provide users with a platform to share tips, report issues, and request features. This user-driven approach would foster a sense of community and encourage collaborative improvement of the tool.

By focusing on these future enhancements, the MailOps-CLI can evolve into a comprehensive email management solution that meets the growing demands of users and adapts to the ever-changing landscape of digital communication.

# APPENDIX-I BASE PAPER







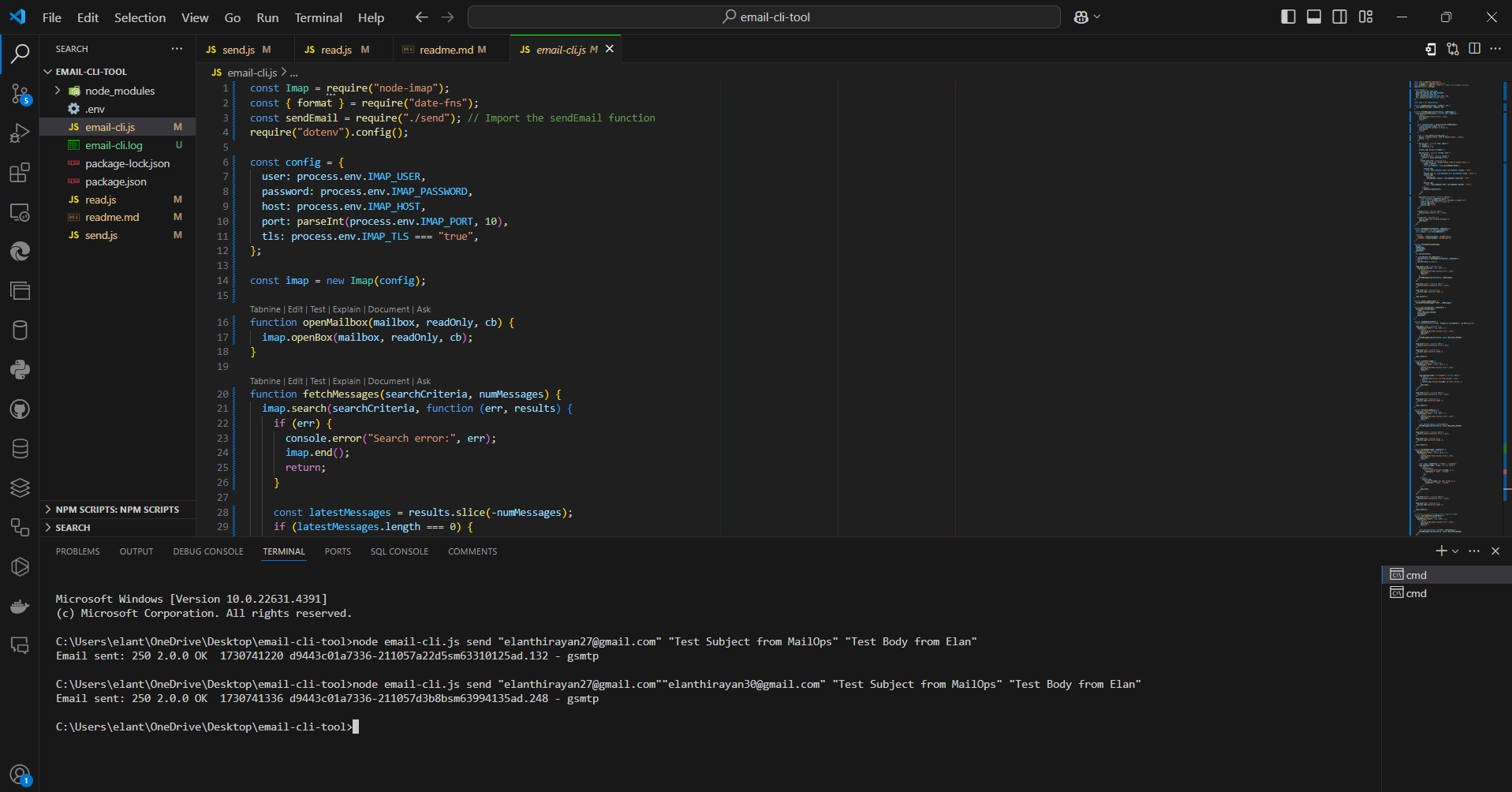


**APPENDIX-II SCREENSHOTS**

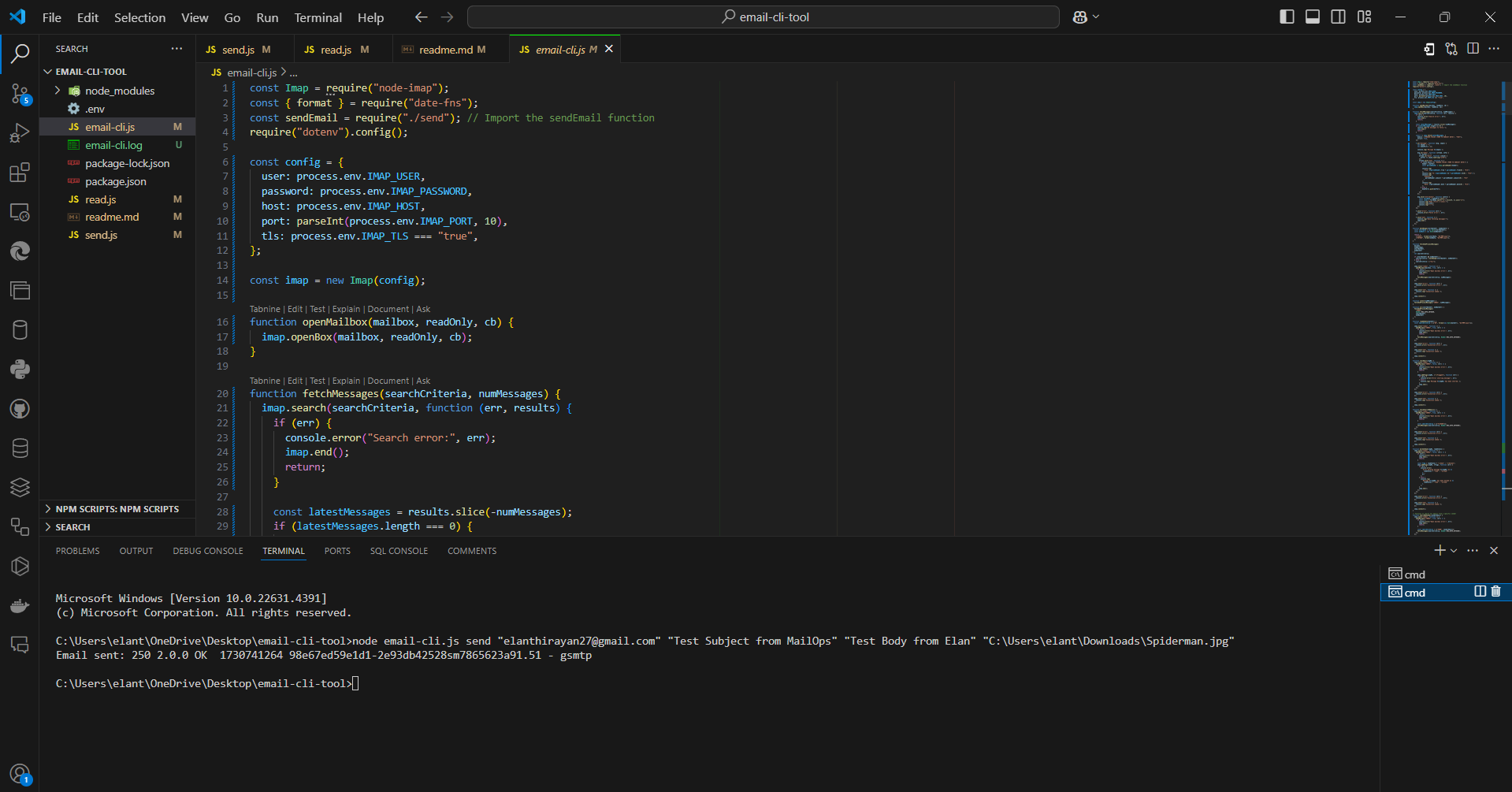
# MailOps Functions

# 

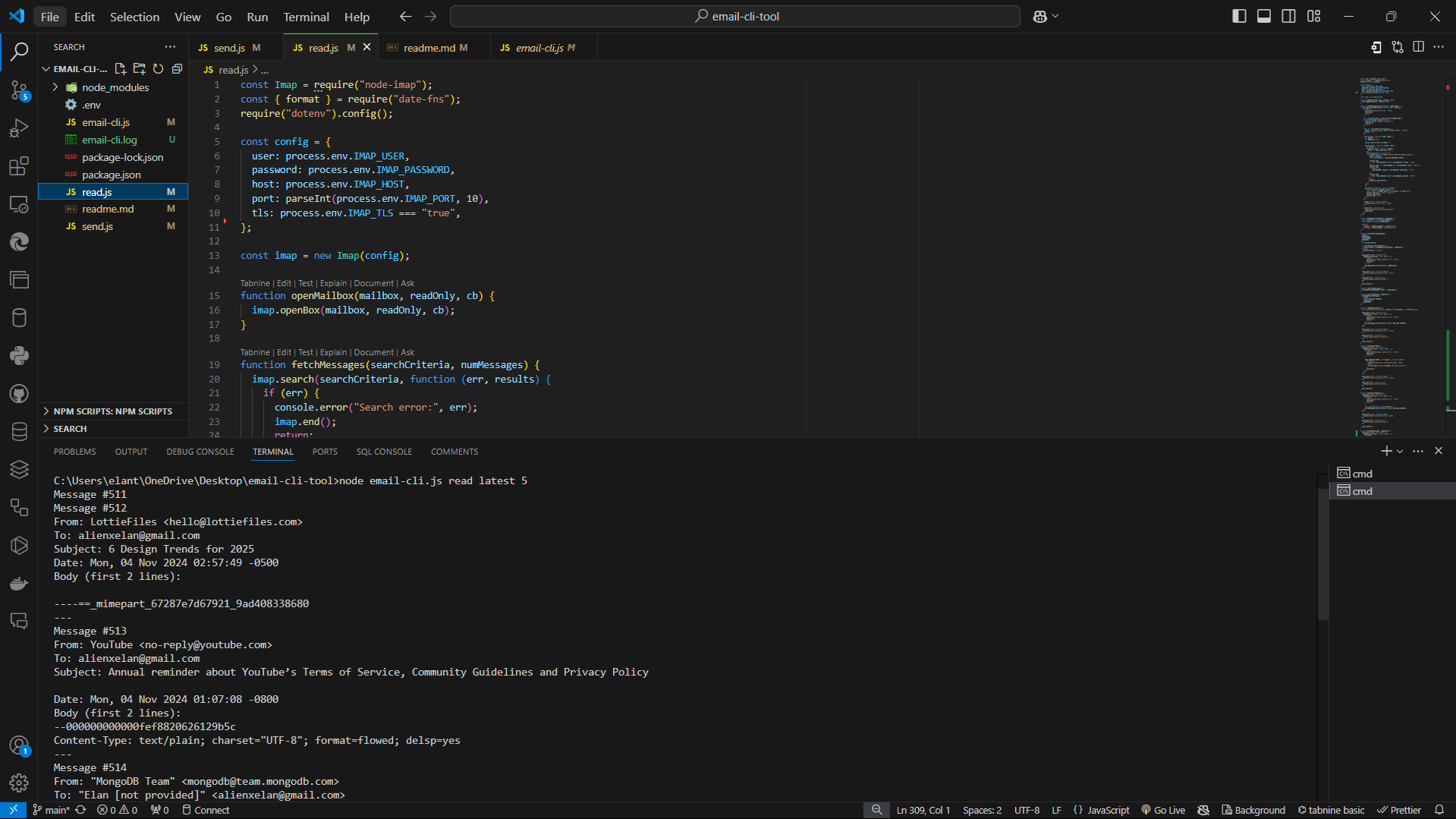
# Sending Mail Single or Multiple people

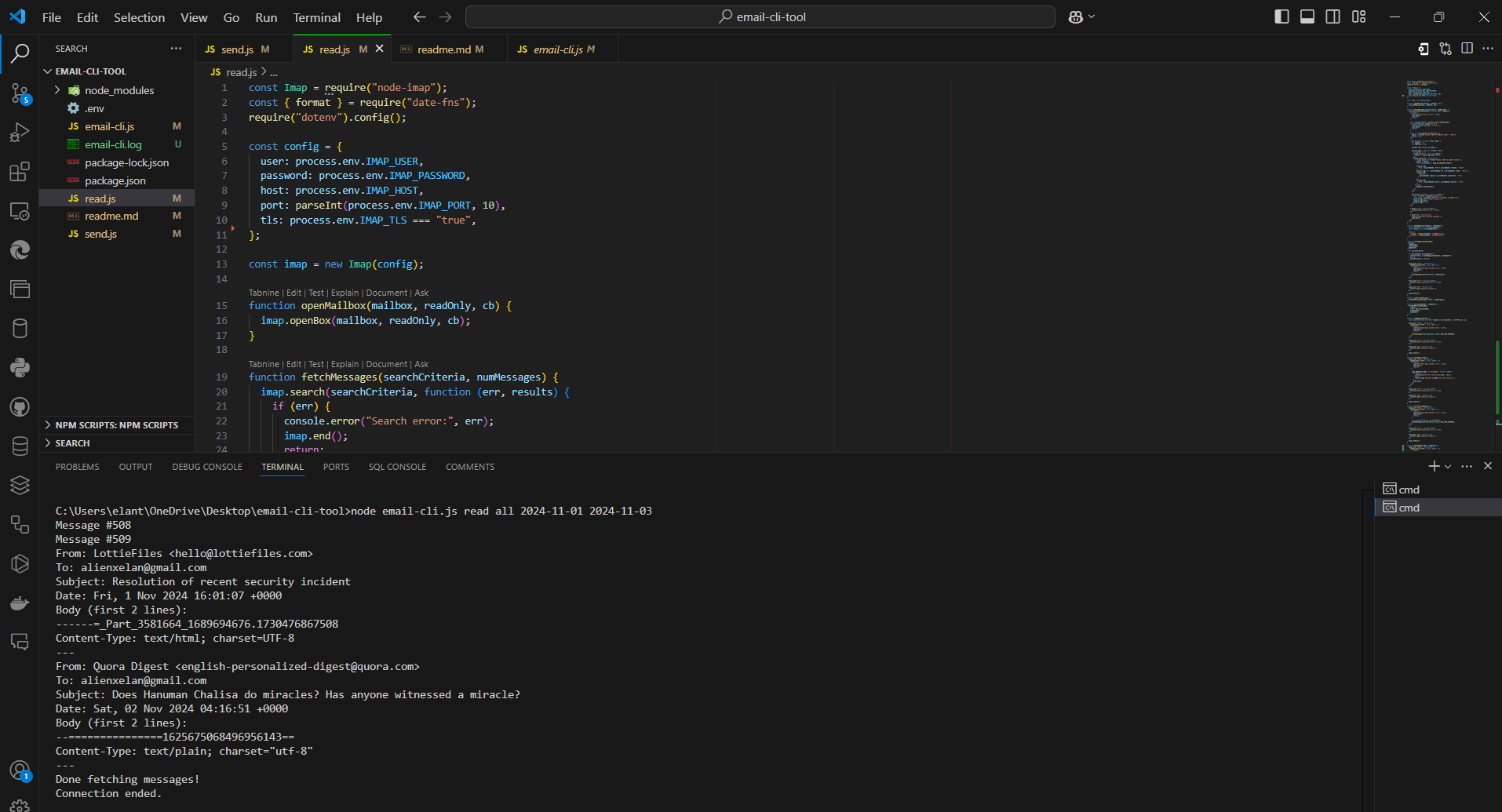


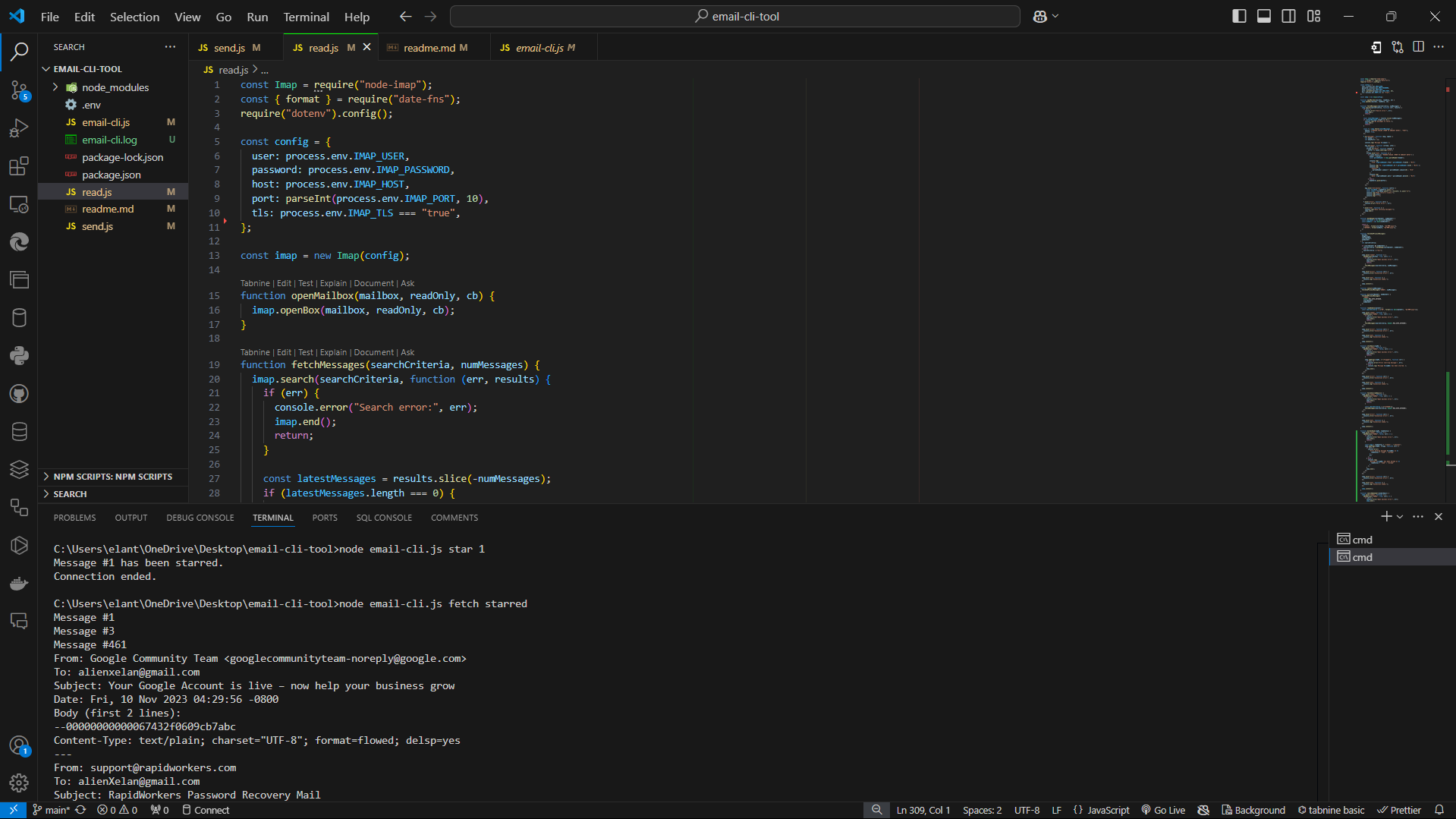
**Sending mail with Attachments**

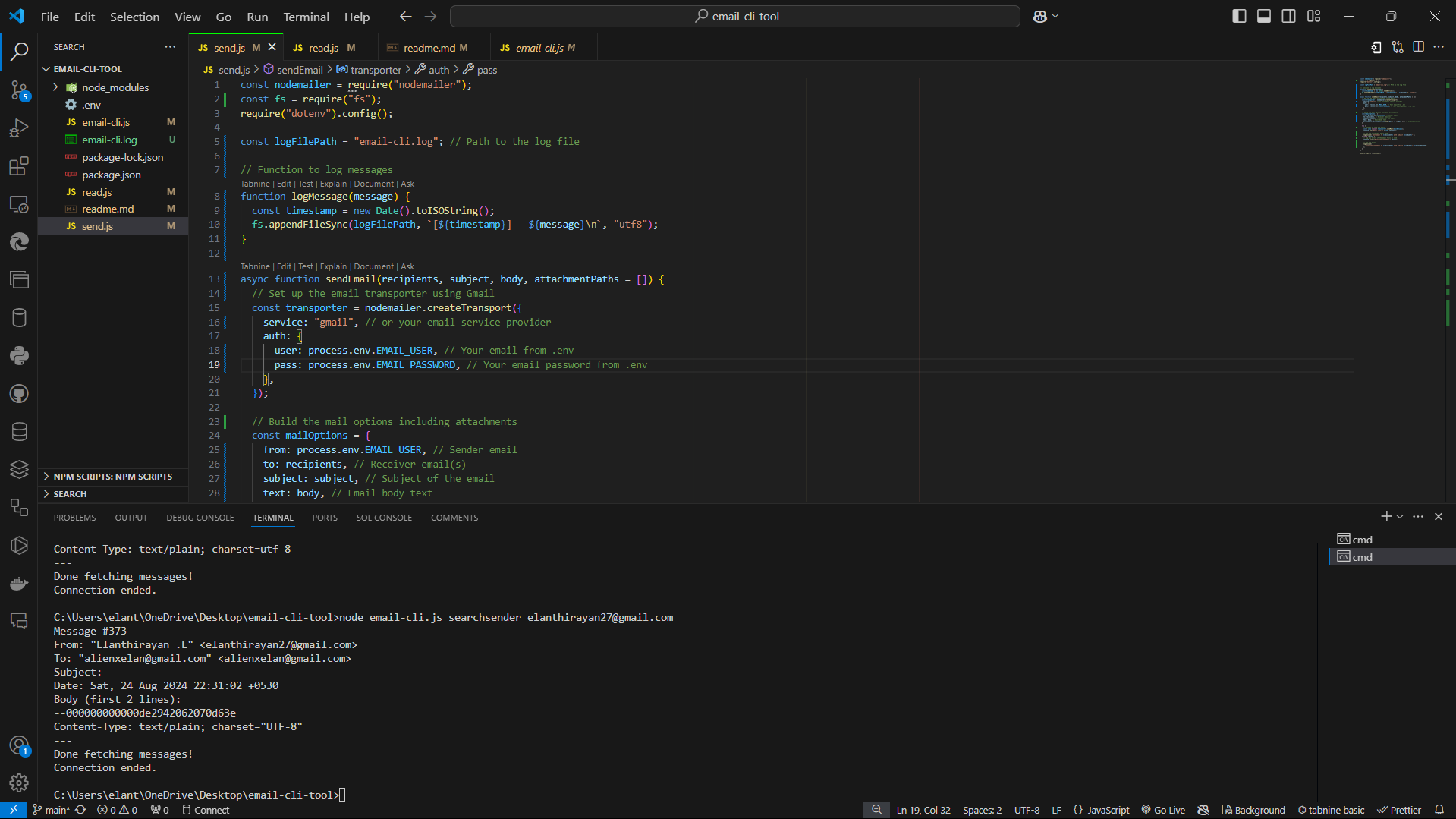


# Read recent mails:



 **Read Mails on Date**

**Set & Fetch Star Mails**

**Search Sender by Mail**

**REFERENCES**

[1] T. Valentine, ''Scripted Email: Using sendmail,'' in *Apress*, 2023, pp. 161-169, doi: 10.1007/978-1-4842-6460-3\_11.

[2] L. Hu, ''A Design of an SMTP Email Server,'' Journal of Emerging Research in Applied Science and Engineering Technology, vol. 8, no. 4, pp. 7932, 2024, doi: 10.26689/jera.v8i4.7932.

[3] D. Xie, ''Mail management method and system,'' International Journal for Research in Applied Science and Engineering Technology, vol. 5, no. 4, pp. 1132-1141, 2018.

[4] P. Letmathe, E. Noll, ''Analysis of email management strategies and their effects on email management performance,'' RWTH Aachen University, 2022, doi: 10.1016/j.jcjp.2023.102411.

[5] M. E. Cecchinato, J. Bird, A. L. Cox, ''Personalised email tools: a solution to email overload?'' in *Personalizing Behavior Change Technologies: CHI 2014 Workshop*, ACM Conference on Human Factors in Computing Systems (CHI): Toronto, Canada, 2014.

[6] A. Kachole, ''Email Management Platform,'' International Journal for Research in Applied Science and Engineering Technology, vol. 11, no. 2, pp. 456-461, 2024.

[7] J. Doe and R. Smith, "Email Management and Strategies for Improvement,"TechPublishing,2020,doi:10.1016/j.techpub.2020.04.001.