

# **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**

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

**HEAD OF THE DEPARTMENT**

Department of Computer Science and  
Engineering,  
Anand Institute of Higher  
Technology,  
Kazhipattur, Chennai-603103

**SIGNATURE**

**Mrs. V. Keerthiga, M.E.,**

**ASSISTANT PROFESSOR**

Department of Computer Science and  
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**

<b>CHAPTER</b>	<b>TITLE</b>	<b>PAGE NO</b>
	<b>ABSTRACT</b>	<b>iv</b>
	<b>LIST OF FIGURES</b>	<b>vii</b>
	<b>LIST OF ABBREVIATIONS</b>	<b>viii</b>
	<b>LIST OF TABLES</b>	<b>ix</b>
<b>1.</b>	<b>INTRODUCTION</b>	<b>10</b>
	1.1 OBJECTIVE	11
	1.2 SCOPE	11
<b>2.</b>	<b>LITERATURE SURVEY</b>	<b>12</b>
<b>3.</b>	<b>ANALYSIS</b>	<b>20</b>
	3.1 SYSTEM ANALYSIS	20
	3.1.1 Problem Identification	20
	3.1.2 Existing System	20
	3.1.3 Proposed System	21
	3.2 REQUIREMENT ANALYSIS	21
	3.2.1 Functional Requirements	22
	3.2.2 Non-Functional Requirements	22
	3.2.3 Hardware Specification	23
	3.2.4 Software Specification	23
<b>4.</b>	<b>DESIGN</b>	<b>24</b>
	4.1 OVERALL DESIGN	24

4.2	UML Diagram	25
4.2.1	Work Flow Diagram	25
4.2.2	Use Case Diagram	26
4.2.3	Class Diagram	27
4.2.4	Activity Diagram	28
4.2.5	Sequence Diagram	29
<b>5.</b>	<b>IMPLEMENTATION</b>	<b>30</b>
5.1	MODULES	30
5.2	MODULE DESCRIPTION	31
<b>6.</b>	<b>TESTING</b>	<b>32</b>
6.1	TESTING AND VALIDATION	32
6.2	BUILD THE TEST PLAN	35
<b>7.</b>	<b>RESULT AND DISCUSSION</b>	<b>39</b>
<b>8.</b>	<b>USER MANUAL</b>	<b>41</b>
<b>9.</b>	<b>CONCLUSION</b>	<b>44</b>
<b>10.</b>	<b>FUTURE ENHANCEMENT</b>	<b>45</b>
	<b>APPENDICES</b>	
	APPENDIX 1 BASE PAPER	
	APPENDIX 2 SCREENSHOTS	
	<b>REFERENCES</b>	

## **LIST OF FIGURES**

<b>FIGURE NO</b>	<b>FIGURE DESCRIPTION</b>	<b>PAGE NO</b>
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 ABBREVIATIONS**

### **SYMBOLS**

### **ABBREVIATIONS**

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**

<b>TABLE NO</b>	<b>TABLE NAME</b>	<b>PAGE NO</b>
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.

## **1.1 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.

## **1.2 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**

#### **3.1 SYSTEM ANALYSIS**

##### **3.1.1 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.

##### **3.1.2 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.

### **3.1.3 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.

## **3.2 REQUIREMENT ANALYSIS**

### **3.2.1 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:

#### **1. User Authentication:**

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

## **2. 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.

## **3. 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.

## **4. 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

## **5. 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.

### **3.2.2 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.

**3.2.3 Hardware Specifications**

Processor	: i3 and above
Hard disk	: 500GB and above
RAM	: 4GB and above

**3.2.4 Software Specifications**

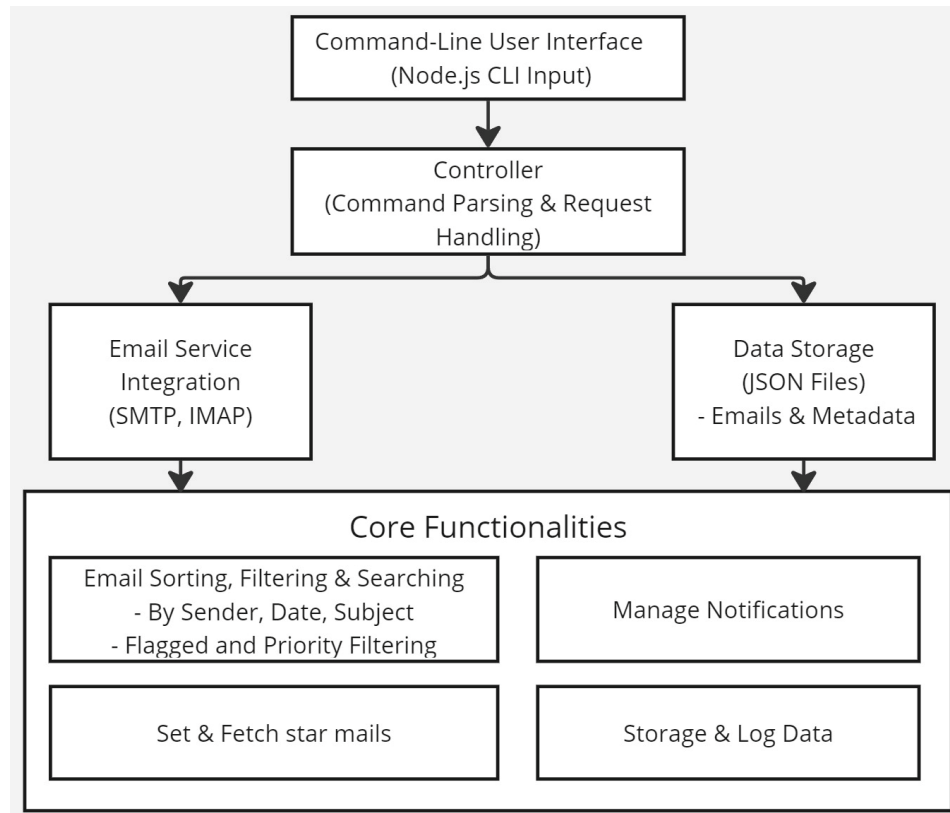
Operating System	: Windows (64bit)
Tool	: Command Prompt, Visual Studio Code.
Language	: JavaScript and Node.js.

## CHAPTER 4

### DESIGN

#### 4.1 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.



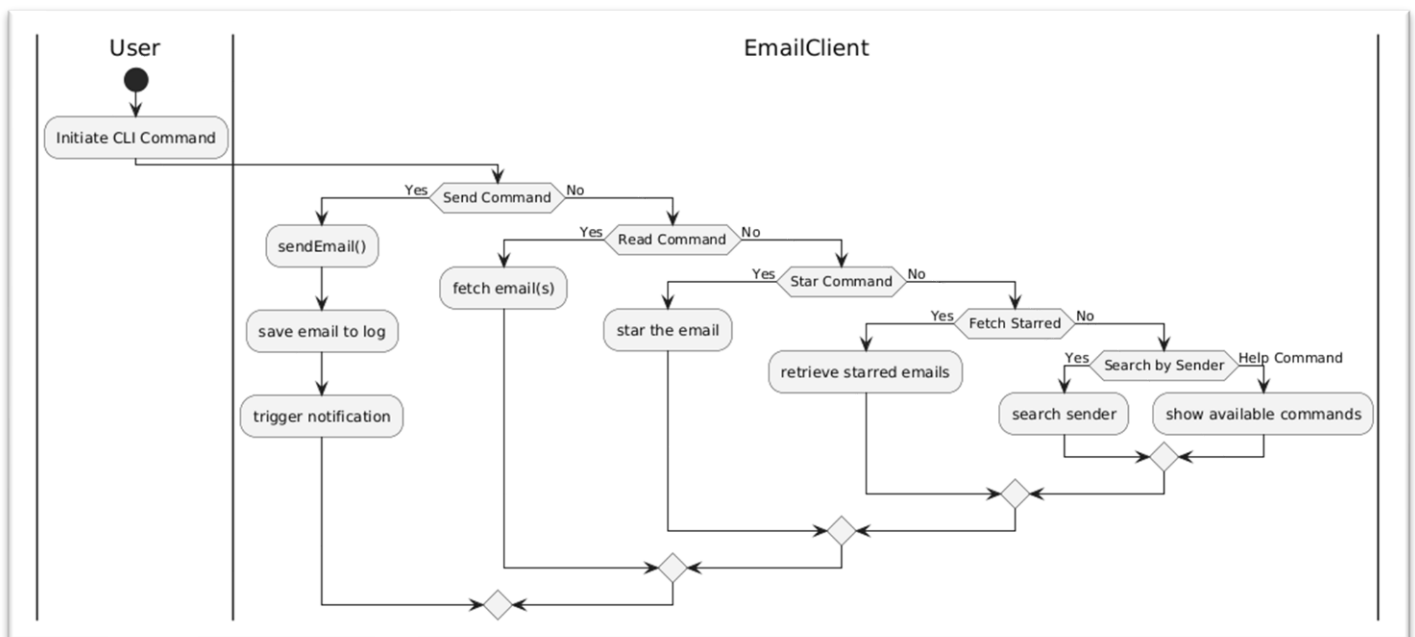
**Fig 4.1 Proposed System Architecture**



## 4.2 UML DIAGRAMS

### 4.2.1 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.



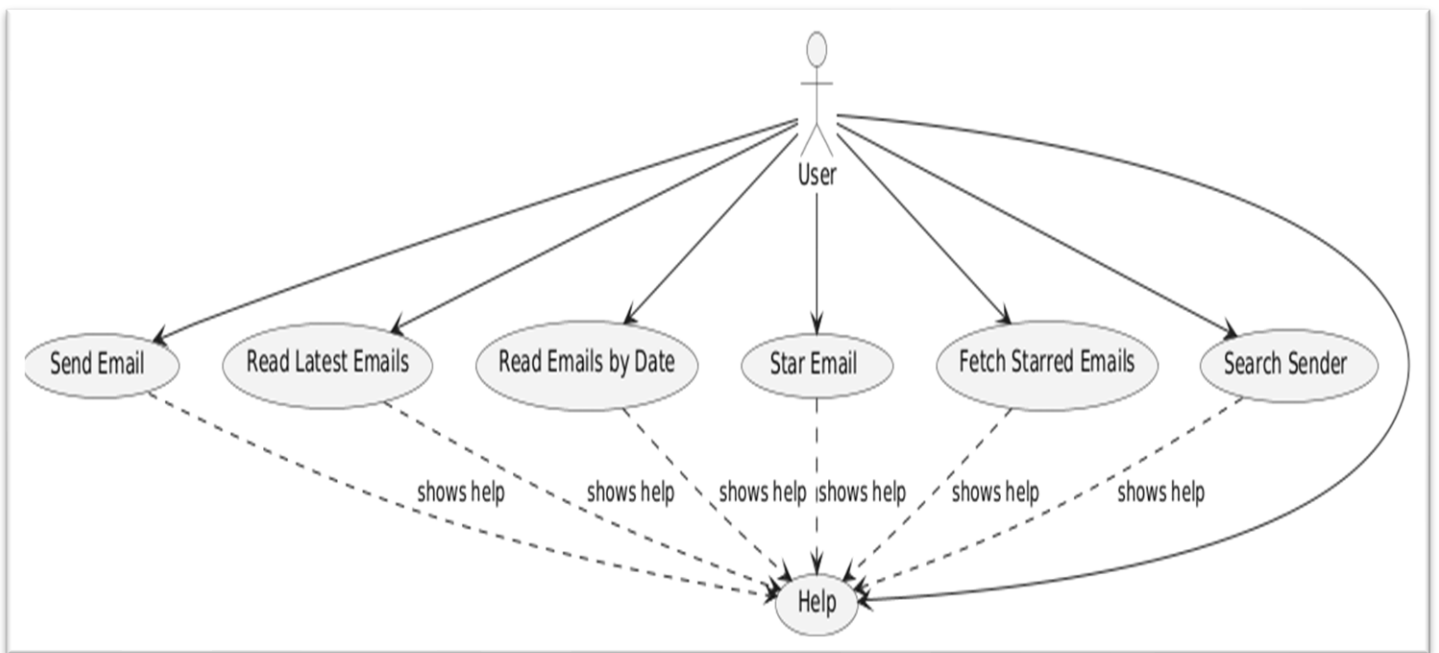
**Fig 4.2 Work flow Diagram**

### 4.2.2 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.

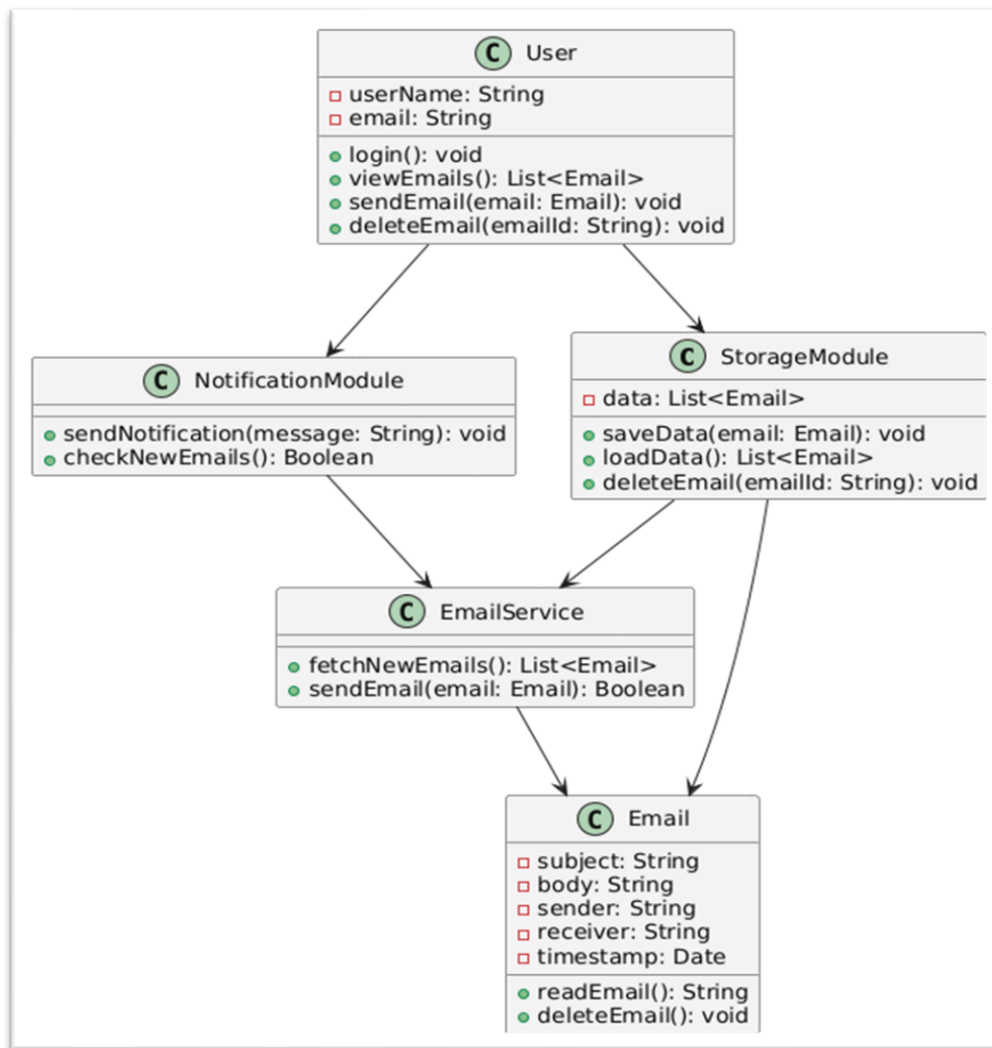
**Actor:** An actor is a person, organization or external system that plays a role in one or more interaction with the system.



**Fig 4.3 Use Case Diagram**

### 4.2.3 Class Diagram

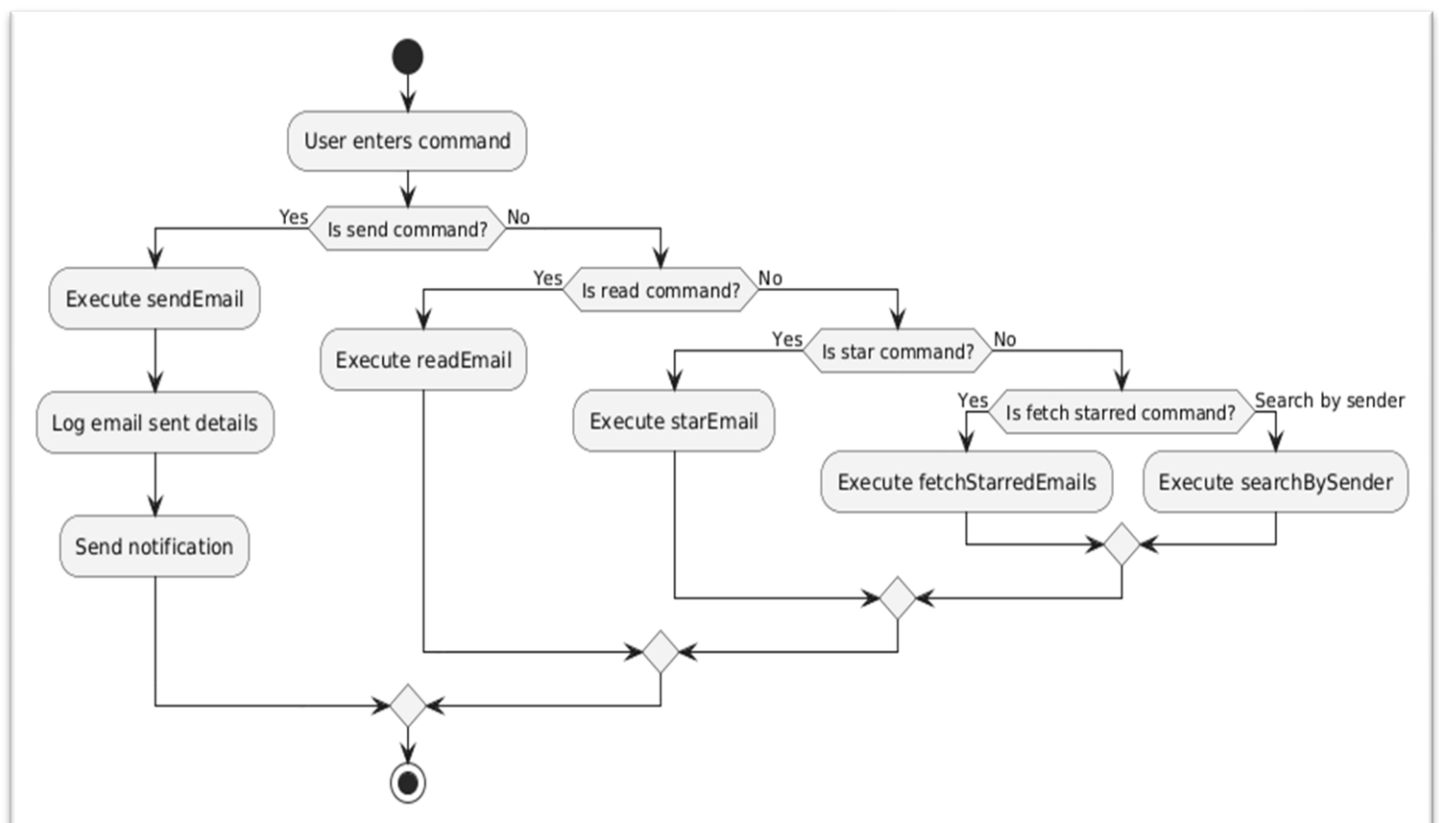
The class diagram outlines the structure of the Email Management System by detailing its classes, attributes, and methods. It highlights the relationships between different entities, such as User, Email, and MailServer classes. This diagram provides a blueprint for the implementation of the system, showcasing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



**Fig 4.4 Class Diagram**

#### 4.2.4 Activity Diagram

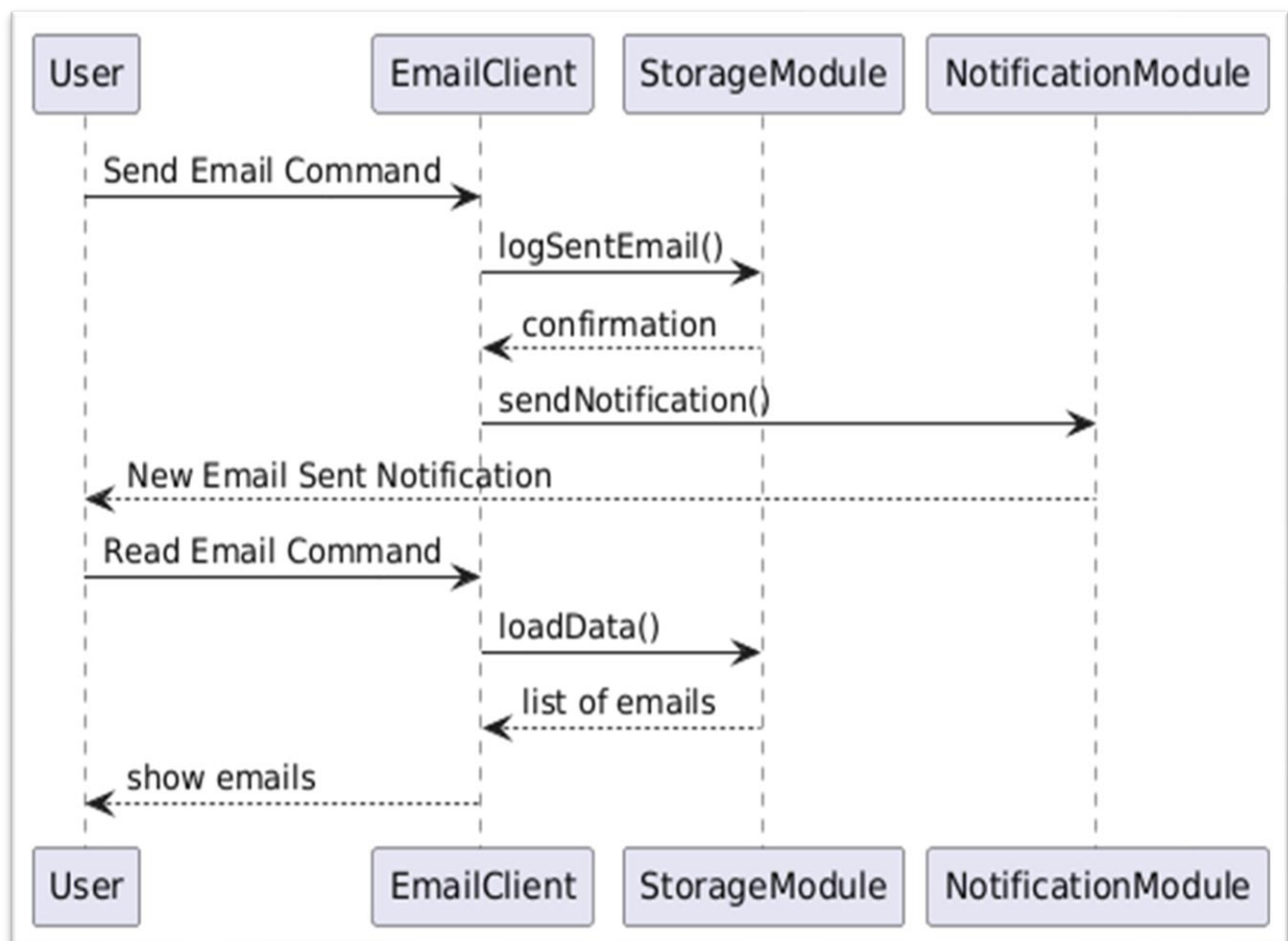
The activity diagram visualizes the flow of activities involved in key processes within the system, such as the email sending process. It shows the sequence of actions, decisions, and parallel processes, providing insights into how users will interact with the system. This diagram helps in identifying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

#### **5.1 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

#### **5.2 MODULE DESCRIPTION**

##### **5.2.1 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.

##### **5.2.2 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

The 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

#### 6.1 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.

## 6.2 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

<b>S.No</b>	<b>TestID</b>	<b>Test Description</b>	<b>Test Status (Pass/Fail)</b>
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**

# Personalised Email Tools: A Solution to Email Overload?

**Marta E. Cecchinato**  
UCL Interaction Centre  
University College London  
London, WC1E 6BT  
marta.cecchinato.13@ucl.ac.uk

**Jon Bird**  
City University London  
London, EC1V 0HB  
jon.bird@city.ac.uk

**Anna L. Cox**  
UCL Interaction Centre  
University College London  
London, WC1E 6BT  
anna.cox@ucl.ac.uk

## ABSTRACT

The stress resulting from the daily demands of email exchange and management has been labelled *email overload* [4, 13]. The extent to which individuals are affected by email overload has much to do with personal, cultural, and contextual differences. However, in general people are inefficient at dealing with email and could potentially reduce the stress associated with it if they changed their behaviour. In this paper, we review some of the strategies offered in the literature, as well as some email tools that have been developed to help people manage their inboxes. We point out the benefits and disadvantages of them, suggesting that adaptive approaches might be more effective at facilitating email behaviour changes than fixed one-size-fits-all solutions. We argue that the adaptation should be the result of personalisation (controlled by the system) and customisation (controlled by the user) because these processes support behaviour change in different ways.

## Author Keywords

Email overload; email tools; behaviour-change technologies

## ACM Classification Keywords

H.5.m. Information interfaces and presentation (e.g. HCI): Miscellaneous.

## INTRODUCTION

The daily demands of email exchange in both a personal and professional environment can be overwhelming and can act as a stressor. Email overload was defined by Dabbish and Kraut [4, p.431] as “users’ perceptions that their own email use has gotten out of control” and as a result causes them stress. How we react to and deal with email overload has much to do with our personal, cultural, and contextual differences. Researchers have proposed fixed, personalised and customised approaches to dealing with email overload. Fixed strategies, such as checking email once a day, do not require adaptation by the user or the system, whereas

personalised email tools automatically change in response to a user’s action. These are distinguished from customised email tools, which are changed directly by the user based on their own needs. Building on a review of email strategies and tools, we argue there is no one-size-fits-all solution and that effective behaviour change will result from approaches that can change according to the user’s idiosyncratic needs. Whether personalisation or customisation, or both should drive behaviour changes, is an open research question and an issue that could be discussed at the workshop.

## EMAIL MANAGEMENT LITERATURE

In this section we present an overview of the literature, first establishing that the way people manage their email varies depending on personal, cultural and contextual factors. We do so to underline the importance of our conclusions, in which we state that these differences must be taken into account when considering the design of behaviour change tools. We then give an overview of the different types of strategies proposed to manage email overload, distinguishing between fixed, personalised and customised strategies. We argue that users can actually change their behaviour by using these strategies, but they might not always choose the best goal for themselves.

## Differences in Email Management Strategies

As early as 1988, Mackay [9, p.383] reports that one of her participants complained she felt she was “*on the edge of losing control of her mail*”. This feeling is unsurprisingly still common today, given that the number of emails sent has increased massively in the last 25 years. Mackay [9] studied email usage in an organization and found that email was being employed for time and task management as well as communication purposes. Her findings suggest that email usage is extremely diverse and two main groups of email handling strategies were identified: *prioritizers* and *archivers*. The former limited their time spent in the inbox and maximized efficiency by prioritizing; the latter monitored all incoming messages for fear of missing important information.

In 1996 Whittaker and Sidner [14] also investigated email management behaviour. They initially proposed a simple ‘one-touch model’ for email management, according to which emails are read, replied to (if required), and then either deleted or filed immediately. However, they noticed that this model did not meet the demands of white-collar workers who used email not only for asynchronous communication but also for task management and personal

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

CHI’14, April 26 – May 1, 2014, Toronto, Canada.

Copyright 2014 ACM 978-1-XXXX-XXXX-X/XX/XX...\$10.00.



archiving. Their study identified different email management strategies: *no filers* (who never cleared their inbox), *frequent filers* (who constantly cleaned their inbox) and *spring cleaners* (who cleaned their inbox every few months). Ten years later, in 2006, Fisher et al. [5] revisited Whittaker and Sidner's study to assess whether its findings were still valid. Their study suggested that there is a wide variety of email handling behaviours and that users adopt more than one email strategy depending on circumstances.

Research has also identified cultural differences in email management strategies. Tang et al. [11] conducted a study with the goal of exploring differences in email usage across the world. They noticed there are some tendencies that seem confined to geographical regions. For example, North American users tend to keep more messages in their inboxes compared to South American users; European users seem more inclined to be frequent filers or spring cleaners, compared to Asian users who are typically no filers.

These studies show that there is a range of personal, cultural and contextual factors that affect people's behaviour in how they process their email and that typically the processing strategies do not alleviate the effects of email overload. In the next section we consider the benefits of fixed and adaptive approaches to email management and their consideration of these differences.

#### **Fixed or Adaptive Approaches to Email Management?**

A study by Brumby, Cox and Bird [2] showed how the choice of email management strategies impacts the time one spends in their inbox. The authors investigated the effects of two fixed approaches to email management: a once-a-day checking strategy and a frequent checking strategy that aims to maintain 'inbox-zero'. They found that participants who adopted the once-a-day strategy made fewer visits to email applications and that there was some indication that the overall time spent in one's inbox was lower than when adopting a frequent checking strategy. The results provide evidence that people can adopt new email strategies. However a once-a-day strategy, might be considered too extreme and not suitable for many people: Brumby et al. note that a number of participants were unable to complete the study because they found the strategy too restrictive and worried they could not carry out their work.

Although fixed but less restrictive strategies might be effective, for example, checking email three times a day, a general concern is that they ignore personal, contextual and cultural differences, which, as indicated in the previous section, strongly influence email habits. On the basis of her pioneering research, Mackay [9, p.394] proposed that email tools should be developed to help people manage their email arguing that they should be customisable to meet personal usage preferences because "[...] *no single set of rules is likely to be useful for everyone, providing users with the ability to write their own personal rules should be an effective solution*". Mackay also emphasizes the fact that

circumstances may change so users should be aware that their set of rules within the tool can "*vary according to how busy the user is*" [9, p.395]. Based on their cultural differences research, Tang et al. [11] suggested, "*tailor[ing] user interfaces to account for those contextual differences. [...] Default settings for certain email features may be set differently for different countries*" [11, p.192].

So although a fixed approach to managing email overload can be beneficial at reducing email overload, there is a concern that it is not flexible enough to accommodate individual differences and frequently changing contextual factors. Customised tools have the advantage that they can potentially take into account individual and contextual differences and support behavioural changes towards developing more effective email management strategies.

#### **EMAIL TOOLS**

In this section we present an overview of six tools that are available for free on the Internet and that were designed to help users better manage their inbox, for instance by helping people reflect on their behaviour, or by explicitly suggesting ways in which they can change their habits. We distinguish between tools that provide information on email usage but do not try to actively change behaviour and adaptive tools that do try to encourage behaviour change: personalised tools (system-controlled) and customised tools (user-controlled).

##### **Information Tools**

Many of the existing email tools offer metrics on email behaviours, most of which enable users to compare their behaviour within a community or through the most popular social networks. The types of metrics that are provided to users include: hourly/daily/weekly/monthly inbox volume; top senders and recipients; most active hours; average response time; word count; and attachments. The most complete in terms of how many metric features it offers is *Gmail Meter* (<http://gmailmeter.com/>), but it is limited to a weekly report for Gmail accounts. However, *Gmail Meter* does not allow comparisons between weeks, leaving users unable to spot possible behaviour patterns. Supporting data comparison across time could improve such tools, enabling users to better identify their behavioural patterns, which could then lead people to act upon them. An extension of *Gmail Meter*, for example, could be based on a tool developed by MIT Media Labs, called *Immersion* (<https://immersion.media.mit.edu/>), which provides a user-centric perspective of one's email history. It creates a social network analysis graphic representing one's email communications and how they change over time. Again, a tool like this helps users reflect on their behaviours and makes them aware of habits they might not be fully conscious of. Once self-knowledge is gained, people find it easier to change their habits [3].

##### **Personalised Tools**

An example of a personalised tool that changes in response to a user's action is the *Email Game* (<http://emailgame.me/>). It



is designed to teach users how to save time while processing their inbox by giving a default time of 3 minutes to reply to each email. Being designed as a game, the more efficient one is, the more points one receives. Depending on the user's score, a smiley face changes its expression throughout the game. If a user replies to emails within the given time and reaches an empty inbox at the end, they receive more points and the face becomes happier. If a user does not reply within the predefined 3 minutes, or leaves emails in their inbox, then the face becomes sad. To the best of our knowledge, there has been no evaluation of the effectiveness of this tool for reducing email overload. Of course a pitfall of this tool is that it may not be possible to reply to some emails in 3 minutes. However, as for the once-a-day strategy, a study on the use of this tool could probably prove that a 3-minute strategy can be easily adopted for certain types of emails.

#### Tools that allow customisation

*Boomerang* (<http://www.boomerangmail.com/>) is a tool that allows flexibility in sending and receiving email: users can decide to schedule a message to be sent at a point in the future or to have a received message returned to the inbox at a time and date of their choice. Users can select when to defer the inbox message, either before or after reading it. The advantage of this app is that it gives the user the ability to decide when it would be more appropriate to deal with an email. This is an example of a customisable tool because it allows users to take action on their inbox so that it meets their needs, e.g. receiving an email at a more suitable time. Other customisable tools include websites that give advice on how to actively change one's behaviour in order to better deal with emails, such as adopting a policy of replying to emails using between two to five sentences, regardless of the recipient or content (<http://sentence.es>). *Calmbox.me* recommends customising one's email signature with brief email tips (e.g. "This is a Calm Inbox: email is checked once in the AM and once in the PM"). Doing so could nudge whoever receives that email to change consequently their email behaviour accordingly.

#### DISCUSSION

In this paper we have described different types of Internet tools developed to aid users in their email processing. We have presented tools that return metrics and statistics on email usage, encourage healthier email behaviours, provide a visual insight on the user's inbox or that give advice. There has been very little academic research that has evaluated these tools and therefore currently it is difficult to assess how effective they are at reducing email overload.

Therefore, we now consider personalisation and customisation email management tools within a behavioural theory framework, to better understand their potential benefits. Behaviour change technologies can use behavioural theory in three ways: informing the design of systems; guiding evaluation strategies; and defining target users [6]. In this paper we focus only on how behavioural

theory could inform the design of effective email tools. We will consider three theoretical frameworks: the nudge theory; goal-setting theory; and the theory of planned behaviour.

According to the nudge framework of Thaler and Sunstein [12], behaviour can be changed by making changes in the environment. This framework is based on the idea developed by Tversky and Kahneman [7] that our cognitive system operates on two levels: automatic and reflective and nudging techniques target the automatic brain system. Personalised email systems could be effective by incorporating nudge techniques, for example, by limiting the time allowed to reply to an email. Personalised email tools can also use goal-setting theory, which is a psychological framework for understanding how to motivate behaviour change [8]. Specifically for emails, users can be guided towards more effective and efficient goal selection (e.g. checking email less frequently) through a personalised tool, if the goal setting is adopted. This can be especially beneficial, because, as Scott et al. [10] observed, people often set themselves easier goals in comparison to the goals they set for others. However, according to the goal-setting framework, challenging goals are more effective at bringing about behaviour change. For this reason it might be most effective if the email tool sets goals for the users.

In contrast to nudging, the theory of planned behaviour [1] is concerned with the reflective cognitive system. It proposes that intentions predict behaviour and these are influenced by: (i) the attitude towards a specific behaviour; (ii) subjective norms; and (iii) perceived behavioural control. This implies that the user has a more active role in changing their behaviour and would suggest that customisation is beneficial. One pitfall of this theory is that it does not consider behaviours which are not 100% voluntary or under the user's control. It is likely that, at least sometimes, people are not aware of their email behaviour. A customisable email system can be combined with information tools that quantify users' behaviours. These can facilitate self-reflection, which in turn can help change habits and the attitudes towards them [3]. For example, based on a study of social networking, Zhou et al. [15] suggest that estimating usage could reduce email related stress by reducing the tendency to overestimate time spent on the activity.

It is clear that both personalised and customised email management tools have the potential to support email behaviour change. An ideal system would incorporate both approaches in order to maximise the effectiveness of the tool in adapting changes in the personal, cultural, and contextual factors that affect the extent to which users experience overload.

#### CONCLUSION

In this paper we have presented the problem of email overload: stress caused by managing our ever-growing

inboxes. We highlighted that there are different ways of handling email, depending on personal, cultural and contextual influences. For this reason we argue that adaptive approaches might be more effective at facilitating email behaviour change than fixed one-size-fits-all solutions. We argue that the adaptation should be the result of both personalisation (controlled by the system) and customisation (controlled by the user) because these processes support behaviour change in different ways. Personalisation can potentially nudge changes in behaviour by altering the email management environment, for example limiting the time users have to reply to messages. Following goal setting theory, personalisation can also be used to set challenging goals. Information tools can provide users with insights into their behaviour and this can lead to active intentions to change habits that can be supported by customisation. Our future research will continue to investigate how behaviour theory can be incorporated into the design of effective email management tools that can lead to less email overload but still consider personal, cultural and contextual differences.

#### AIMS FOR WORKSHOP

We look forward to discussing whether combining system-controlled personalisation and user-controlled customisation with a single behaviour change technology offers a solution to the problem of matching a behaviour change technology to users.

#### ACKNOWLEDGMENTS

This work is funded by an EPSRC DTG studentship and supported by the EPSRC grant EP/K025392/1 for the Digital Epiphanies project.

#### BIOS

**Marta E. Cecchinato** is a PhD candidate at UCL Interaction Centre, University College London, with a MSc in social psychology. She is interested in understanding how and to what extent new technologies can influence and change our everyday life by reducing stress and increasing wellbeing. In particular, her research focuses on work-life balance and how personal informatics tools can help regain control over digital habits.

**Jon Bird** is a Lecturer in Pervasive Computing at City University London. His research focuses on how technology can be used to address social and health issues. A particular interest is using technology to facilitate behavioural change, with a focus on sustainable behaviour and healthy eating. He is currently using mobile phones for public health applications in the developing world.

**Anna L. Cox** is a Reader in Human-Computer Interaction and Deputy Director of UCL Interaction Centre, University College London. Her current research falls in the broad area of HCI for Health & Wellbeing: this includes work on reducing human error in the use of medical devices, exploring the ways in which technology can influence digital practices that may affect wellbeing, and immersion

and engagement in digital hobbies, such as games and citizen science projects, and their influence on post-work recovery.

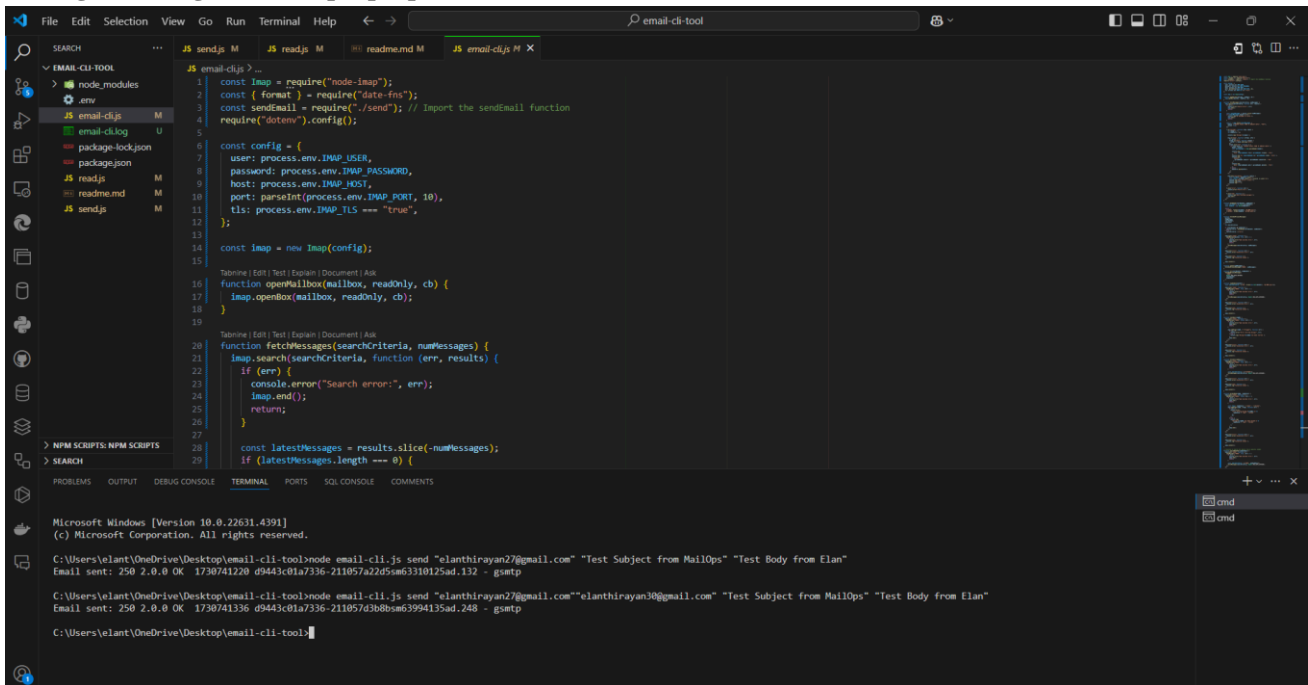
#### REFERENCES

1. Ajzen, I. The theory of planned behavior. *Organizational behavior and human decision processes*, 50(2), (1991), 179-211.
2. Brumby, D. P., Cox, A. L., & Bird, J. Too much email, too much checking. *BCS HCI 1st Workshop on Habits in Human-Computer Interaction*, (2013).
3. Cox, A. L., Bird, J. & Fleck, R. Digital Epiphanies: How self-knowledge can change habits and our attitudes towards them. *BCS HCI 1st Workshop on Habits in Human-Computer Interaction*, (2013).
4. Dabbish, L. A., & Kraut, R. E. Email overload at work: an analysis of factors associated with email strain. *Proc. CSCW* (2006), 431-440.
5. Fisher, D., Brush, A. J., Gleave, E., & Smith, M. A. Revisiting Whittaker & Sidner's email overload ten years later. In *Proc. CSCW* (2006), 309-312.
6. Hekler, E. B., Klasnja, P., Froehlich, J. E., & Buman, M. P. Mind the theoretical gap: interpreting, using, and developing behavioral theory in HCI research. *Proc. CHI* (2013), 3307-3316.
7. Kahneman, D. *Thinking, fast and slow*. Farrar, Straus and Giroux, New York, NY, USA, 2011.
8. Locke, E. A., & Latham, G. P. Building a practically useful theory of goal setting and task motivation: A 35-year odyssey. *American psychologist*, 57(9), (2002), 705-717.
9. Mackay, W. E. Diversity in the use of electronic mail: A preliminary inquiry. *ACM Transactions on Information Systems (TOIS)*, 6(4), (1988), 380-397.
10. Scott, M., Barreto, M., Quintal, F., & Oakley, I. Understanding goal setting behavior in the context of energy consumption reduction. In *Human-Computer Interaction-INTERACT* (2011), 129-143.
11. Tang, J. C., Matthews, T., Cerruti, J., Dill, S., Wilcox, E., Schoudt, J., & Badenes, H. Global differences in attributes of email usage. *Proc. Intl. Workshop on Intercultural Collaboration* (2009), 185-194.
12. Thaler, R. H., & Sunstein, C. R. *Nudge: Improving decisions about health, wealth, and happiness*. Yale University Press, New Haven, CT, USA, 2008.
13. Venolia, G. D., Dabbish, L., Cadiz, J. J., & Gupta, A. Supporting Email Workflow: Technical Report MSR-TR-2001-88 (2001). *Microsoft Research, Redmond and WA*.
14. Whittaker, S., & Sidner, C. Email overload: Exploring personal information management of email. *Proc. CHI* (1996), 276-283.
15. Zhou, Y., Bird, J., Cox, A., & Brumby, D. Estimating usage can reduce the stress of social networking. *CHI Personal Informatics Workshop* (2013).

**APPENDIX-II**  
**SCREENSHOTS**

# MailOps Functions

## Sending Mail Single or Multiple people



The screenshot shows the VS Code editor with the `email-cli.js` file open. The code defines an `Imap` class and functions for sending and fetching emails. The terminal at the bottom shows the execution of the tool, sending two test emails to different email addresses.

```
1 const Imap = require("node-imap");
2 const { format } = require("date-fns");
3 const sendEmail = require("./send"); // Import the sendEmail function
4 require("dotenv").config();
5
6 const config = {
7   user: process.env.IMAP_USER,
8   password: process.env.IMAP_PASSWORD,
9   host: process.env.IMAP_HOST,
10  port: parseInt(process.env.IMAP_PORT, 10),
11  tls: process.env.IMAP_TLS === "true",
12 };
13
14 const imap = new Imap(config);
15
16 function openMailbox(mailbox, readOnly, cb) {
17   imap.openBox(mailbox, readOnly, cb);
18 }
19
20 function fetchMessages(searchCriteria, numMessages) {
21   imap.search(searchCriteria, function (err, results) {
22     if (err) {
23       console.error("Search error:", err);
24       imap.end();
25       return;
26     }
27
28     const latestMessages = results.slice(-numMessages);
29     if (latestMessages.length === 0) {
```

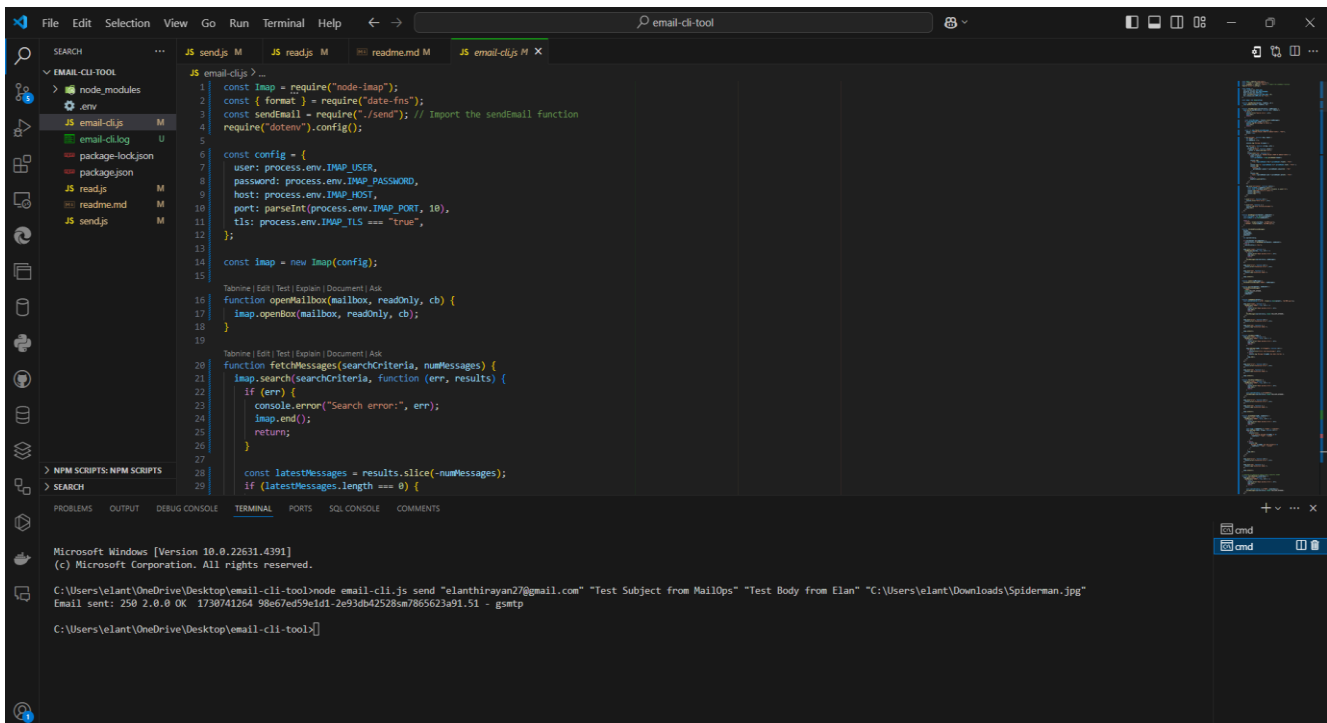
Microsoft Windows [Version 10.0.22631.4391]  
(c) Microsoft Corporation. All rights reserved.

C:\Users\elant\OneDrive\Desktop\email-cli-tool>node email-cli.js send "elanthirayan2@gmail.com" "Test Subject from MailOps" "Test Body from Elan"  
Email sent: 250 2.0.0 OK 1730741220 d9443c81a7336-211057a22d5a63310125ad.132 - gsmtsp

C:\Users\elant\OneDrive\Desktop\email-cli-tool>node email-cli.js send "elanthirayan3@gmail.com" "Test Subject from MailOps" "Test Body from Elan"  
Email sent: 250 2.0.0 OK 1730741336 d9443c81a7336-211057d3b8b5a63994135ad.248 - gsmtsp

C:\Users\elant\OneDrive\Desktop\email-cli-tool>

## Sending mail with Attachments



The screenshot shows the VS Code editor with the `email-cli.js` file open. The code is identical to the previous one, but the terminal output shows an email being sent with an attachment, `C:\Users\elant\Downloads\Spiderman.jpg`.

```
1 const Imap = require("node-imap");
2 const { format } = require("date-fns");
3 const sendEmail = require("./send"); // Import the sendEmail function
4 require("dotenv").config();
5
6 const config = {
7   user: process.env.IMAP_USER,
8   password: process.env.IMAP_PASSWORD,
9   host: process.env.IMAP_HOST,
10  port: parseInt(process.env.IMAP_PORT, 10),
11  tls: process.env.IMAP_TLS === "true",
12 };
13
14 const imap = new Imap(config);
15
16 function openMailbox(mailbox, readOnly, cb) {
17   imap.openBox(mailbox, readOnly, cb);
18 }
19
20 function fetchMessages(searchCriteria, numMessages) {
21   imap.search(searchCriteria, function (err, results) {
22     if (err) {
23       console.error("Search error:", err);
24       imap.end();
25       return;
26     }
27
28     const latestMessages = results.slice(-numMessages);
29     if (latestMessages.length === 0) {
```

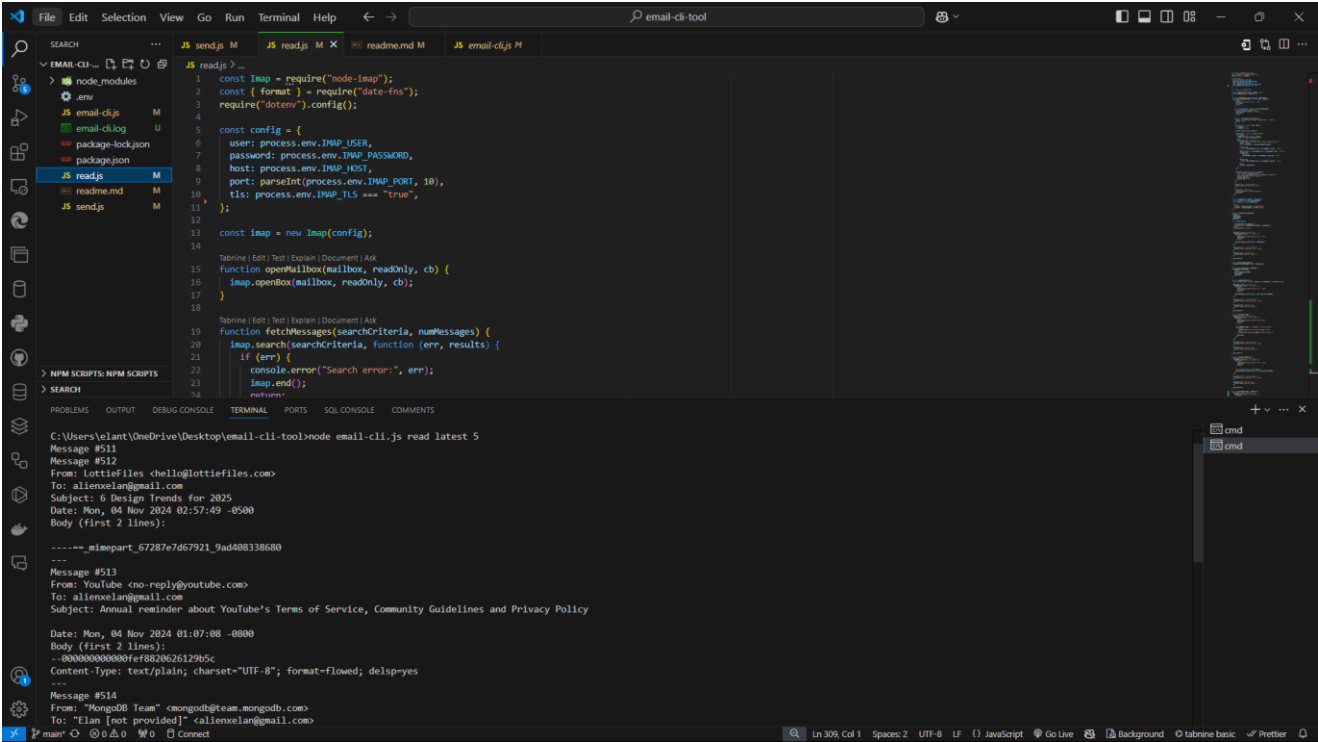
Microsoft Windows [Version 10.0.22631.4391]  
(c) Microsoft Corporation. All rights reserved.

C:\Users\elant\OneDrive\Desktop\email-cli-tool>node email-cli.js send "elanthirayan2@gmail.com" "Test Subject from MailOps" "Test Body from Elan" "C:\Users\elant\Downloads\Spiderman.jpg"  
Email sent: 250 2.0.0 OK 1730741264 98e67ed59e1d1-2e93b042528sm7865623a91.51 - gsmtsp

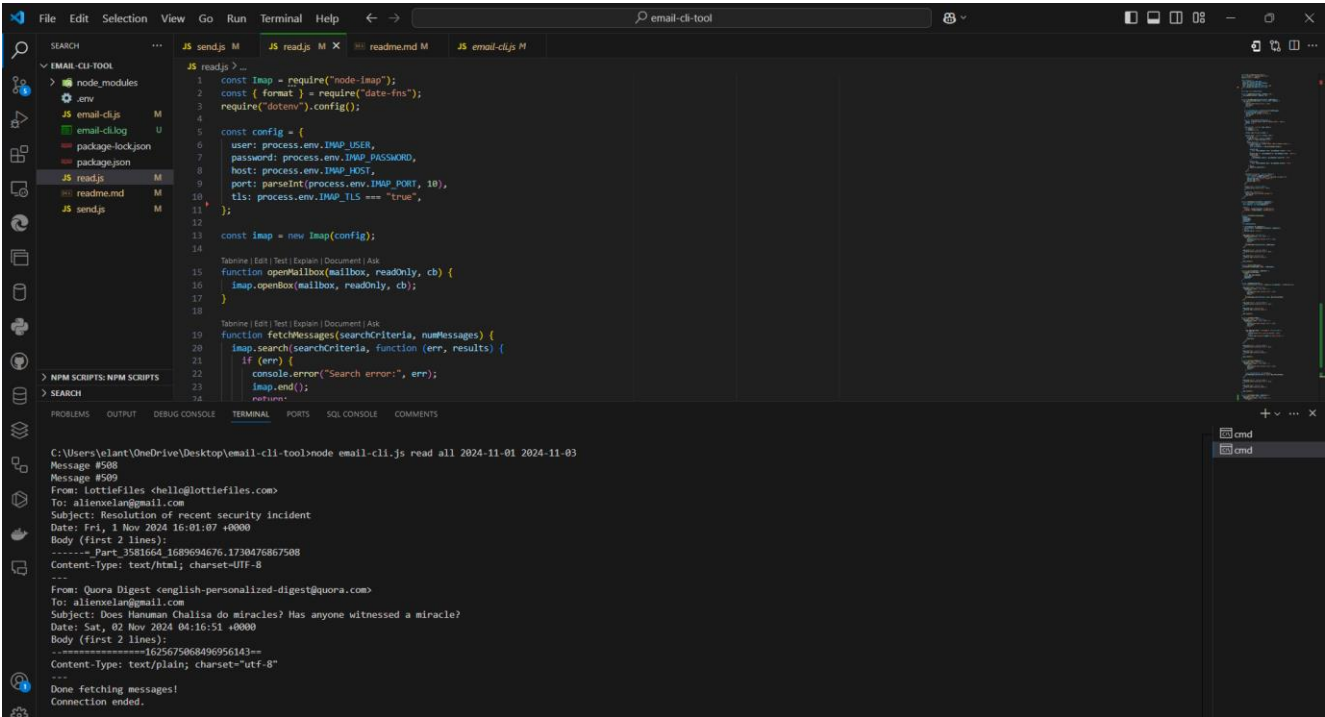
C:\Users\elant\OneDrive\Desktop\email-cli-tool>



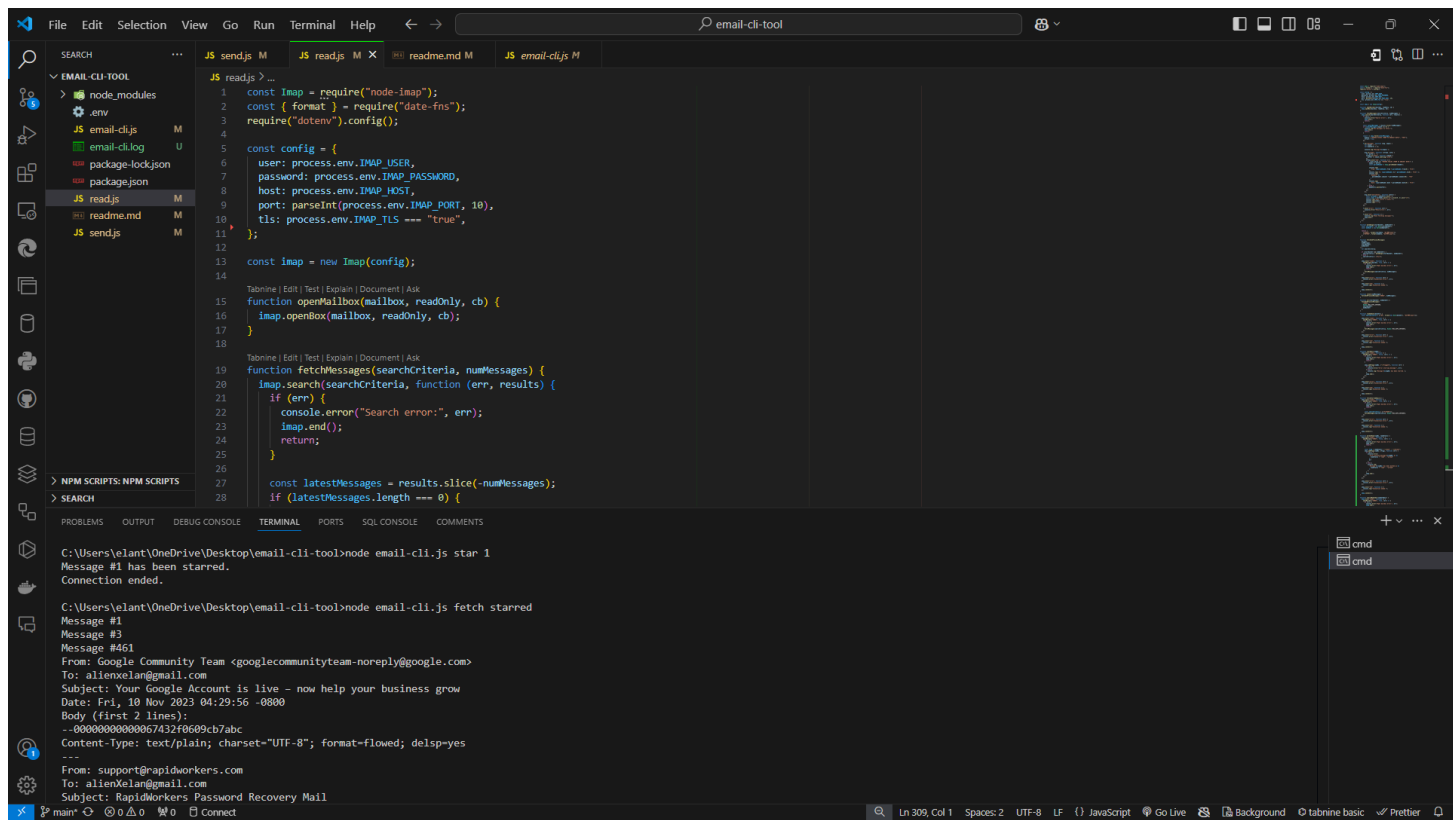
Read recent mails:



Read Mails on Date



## Set & Fetch Star Mails

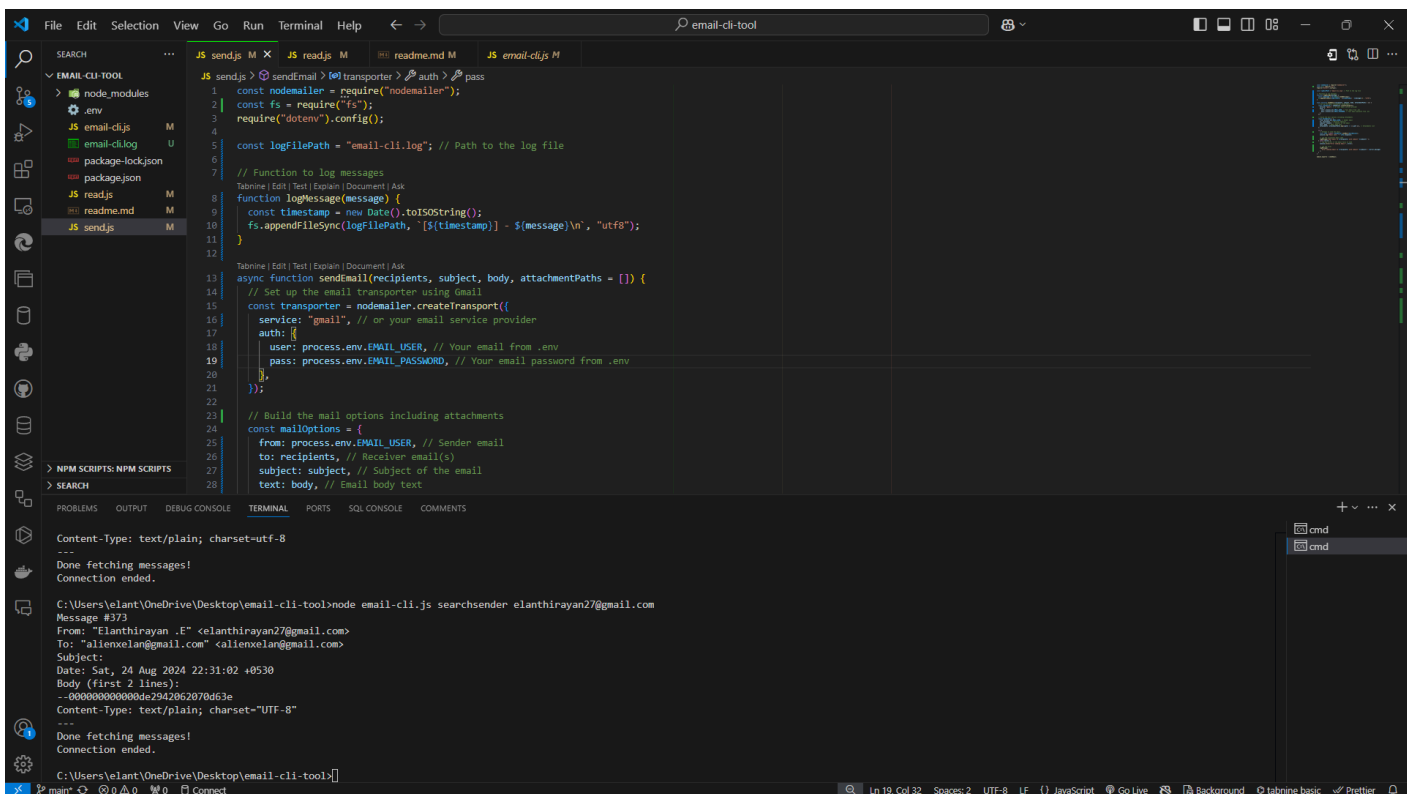


```
1 const Imap = require("node-imap");
2 const { format } = require("date-fns");
3 require("dotenv").config();
4
5 const config = {
6   user: process.env.IMAP_USER,
7   password: process.env.IMAP_PASSWORD,
8   host: process.env.IMAP_HOST,
9   port: parseInt(process.env.IMAP_PORT, 10),
10  tls: process.env.IMAP_TLS === "true",
11 };
12
13 const imap = new Imap(config);
14
15 function openMailbox(mailbox, readOnly, cb) {
16   imap.openBox(mailbox, readOnly, cb);
17 }
18
19 function fetchMessages(searchCriteria, numMessages) {
20   imap.search(searchCriteria, function(err, results) {
21     if (err) {
22       console.error("Search error:", err);
23       imap.end();
24       return;
25     }
26
27     const latestMessages = results.slice(-numMessages);
28     if (latestMessages.length === 0) {
```

C:\Users\elant\OneDrive\Desktop\email-cli-tool>node email-cli.js star 1  
Message #1 has been starred.  
Connection ended.

C:\Users\elant\OneDrive\Desktop\email-cli-tool>node email-cli.js fetch starred  
Message #1  
Message #3  
Message #461  
From: Google Community Team <googlecommunityteam-noreply@google.com>  
To: alienxelan@gmail.com  
Subject: Your Google Account is live - now help your business grow  
Date: Fri, 10 Nov 2023 04:29:56 -0800  
Body (first 2 lines):  
--00000000000067432f0609cb7abc  
Content-Type: text/plain; charset="UTF-8"; format=flowed; delsp=yes  
--  
From: support@rapidworkers.com  
To: alienxelan@gmail.com  
Subject: RapidWorkers Password Recovery Mail

## Search Sender by Mail



```
1 const nodemailer = require("nodemailer");
2 const fs = require("fs");
3 require("dotenv").config();
4
5 const logFilePath = "email-cli.log"; // Path to the log file
6
7 // Function to log messages
8 function logMessage(message) {
9   const timestamp = new Date().toISOString();
10  fs.appendFileSync(logFilePath, `[${timestamp}] - ${message}\n`, "utf8");
11 }
12
13 async function sendEmail(recipients, subject, body, attachmentPaths = []) {
14   // Set up the email transporter using Gmail
15   const transporter = nodemailer.createTransport({
16     service: "gmail", // or your email service provider
17     auth: {
18       user: process.env.EMAIL_USER, // Your email from .env
19       pass: process.env.EMAIL_PASSWORD, // Your email password from .env
20     },
21   });
22
23   // Build the mail options including attachments
24   const mailOptions = {
25     from: process.env.EMAIL_USER, // Sender email
26     to: recipients, // Receiver email(s)
27     subject: subject, // Subject of the email
28     text: body, // Email body text
```

Content-Type: text/plain; charset=utf-8  
---  
Done fetching messages!  
Connection ended.

C:\Users\elant\OneDrive\Desktop\email-cli-tool>node email-cli.js searchsender elanthirayan27@gmail.com  
Message #373  
From: "Elanthirayan .E" <elanthirayan27@gmail.com>  
To: "alienxelan@gmail.com" <alienxelan@gmail.com>  
Subject:   
Date: Sat, 24 Aug 2024 22:31:02 +0530  
Body (first 2 lines):  
--000000000000de2942062070d63e  
Content-Type: text/plain; charset="UTF-8"  
---  
Done fetching messages!  
Connection ended.

C:\Users\elant\OneDrive\Desktop\email-cli-tool>

## 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.