MAILOPS CLI E-MAIL MANAGEMENT TOOL

A PROJECT REPORT

Submitted by

DHIVAHAR V	**310121104025**
ELANTHIRAYAN E	**310121104030**

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

ANAND INSTITUTE OF HIGHER TECHNOLOGY

(AN AUTONOMOUS INSTITUTION)

AFFILIATED TO ANNA UNIVERSITY: CHENNAI 600025

CHENNAI- 603 103

MAY 2025

ANAND INSTITUTE OF HIGHER TECHNOLOGY

(An Autonomous Institution)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

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

SIGNATURE	**SIGNATURE**
Mrs. M Maheswari, M.E., (Ph.D.)	**Mrs. A.S. Balaji, M.E., (Ph.D.)**
HEAD OF THE DEPARTMENT	**ASSISTANT PROFESSOR**
Department of Computer Science and Engineering,	Department of Computer Science and Engineering,
Anand Institute of Higher Technology,	Anand Institute of Higher Technology,
Kazhipattur, Chennai-603103	Kazhipattur, Chennai-603103

Submitted for Project and Viva Voice 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.,(Ph.D.)** and our **Project Coordinator A.S Balaji M.E., (Ph.D.) 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

LIST OF FIGURES

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

CHAPTER 1
INTRODUCTION

In the ever-evolving digital landscape, email continues to be a fundamental mode of communication for professionals across the globe. However, as inboxes grow increasingly cluttered, managing emails efficiently through traditional GUI-based clients can become a cumbersome and time-consuming process. This challenge is especially pronounced for developers, IT professionals, and tech-savvy users who thrive in command-line environments. Enter MailOps-CLI a revolutionary, command-line-based email management tool designed to streamline inbox navigation, simplify email organization, and supercharge productivity directly from the terminal. With its minimalist design and intuitive commands, MailOps-CLI allows users to send, reply to, and organize emails without the constant switching of windows or distractions of a graphical interface.

At its core, MailOps-CLI seamlessly integrates with Gmail using industry-standard OAuth2 authentication, ensuring secure, password-free access to your emails. This integration allows for instant connectivity to your inbox, sent items, drafts, and more, all controlled through simple and efficient terminal commands. The tool is not just limited to basic email operations; it is supercharged with AI-powered features that enhance the user experience. With auto-reply generation, you can instantly craft context-aware responses, while subject enhancement optimizes your email headers for better clarity and engagement. For those looking to fine-tune their communication style, tone adjustment allows you to shift your message from formal to friendly or even persuasive with a single command. Additionally, sentiment detection helps prioritize responses by analysing the emotional tone of incoming messages, and real-time translation enables seamless multilingual communication without ever leaving the terminal.

Beyond individual email handling, MailOps-CLI is designed to optimize bulk operations and search capabilities. Its advanced search functionality enables users to locate emails based on sender, date, keywords, or even detected sentiment, saving valuable time that would otherwise be spent scrolling through endless threads. With batch processing commands, users can swiftly archive, delete, or mark multiple emails as read with just a few keystrokes, making inbox management not only faster but remarkably efficient.

Built with developers, DevOps engineers, system administrators, and tech enthusiasts in mind, MailOps-CLI transforms email handling into a smooth, distraction-free experience. It bridges the gap between powerful email management and the streamlined efficiency of the command line, empowering users to stay focused and productive. By removing the clutter and inefficiencies of traditional email clients, MailOps-CLI redefines how professionals interact with their inboxes, allowing them to communicate faster, organize smarter, and work seamlessly—all without ever leaving the terminal.

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.

1.2 SCOPE

- Email Management Efficiency**:** The tool focuses on improving productivity by offering powerful options to sort, filter, and manage emails across platforms like Gmail and Outlook with ease.

- User-Friendly Interface: Designed with simplicity in mind, the command-line interface is accessible even to non-technical users, ensuring smooth navigation without a steep learning curve.

- AI-Powered Enhancements: With features like smart reply generation, subject line improvement, tone rewriting, and sentiment detection, the tool intelligently enhances communication quality and speed.

- Seamless Integration and Extensibility**:** MailOps-CLI is designed to integrate effortlessly with third-party productivity tools and workflow automation platforms. Its modular architecture allows for easy extension, enabling users to add plugins or custom scripts to further enhance functionality and adapt to specific organizational 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	**: Exploring ChatGPT for Email Content Compression and Summarization.**
Author	: R. Bhuvaneswari, Tharaniesh P. R
Year	: 2023
Publication	: 4th International Conference on Communication Computing And Industry 6.0 (C216), IEEE.

Concept Discussed:

This paper explores the use of ChatGPT, a conversational AI model, for automating the summarization of email content. The authors propose a system that integrates Gmail API and the ChatGPT model to extract, preprocess, summarize, and serve summarized emails via a Flask-based API. The goal is to reduce information overload by enabling users to quickly understand the essence of their inbox messages without manually reading each one. The model was evaluated against popular metrics like ROUGE and BLEU scores, and results indicated that ChatGPT produces high-quality, context-aware summaries.

Problem Identification:

The paper identifies the problem of email overload, where users struggle to process large volumes of daily emails, leading to reduced productivity and potential oversight of critical information. Traditional email systems lack automated summarization capabilities and often demand manual review of entire threads. There is a clear need for intelligent tools that can reduce cognitive load and improve email consumption efficiency.

Work Done:

The authors developed a pipeline to fetch emails using the Gmail API, extract key elements (subject, sender, body), and summarize the content with ChatGPT. The system includes preprocessing, tokenization, ChatGPT integration for summarization, and post-processing for improved readability. Summaries are delivered via a Flask API, with optional caching and custom dataset support. Evaluations using real Gmail data and NLP metrics like ROUGE, F1 Score, and BLEU showed the system's superior performance in generating coherent and relevant summaries compared to previous models.

Knowledge Gained:

The study confirms the efficacy of using large language models like ChatGPT for practical summarization tasks. The results show that such models are capable of understanding context and generating meaningful summaries that enhance user productivity. It also validates the use of transformer-based architectures for handling natural language in real-world applications like email summarization.

Gap:

The proposed approach, while delivering high-quality results, does have some acknowledged gaps. A major limitation is the reliance on third-party APIs like OpenAI, which introduces latency and costs. Additionally, the model's performance can fluctuate depending on the structure and language of the emails, affecting the quality of summaries. The study also points out the need to enhance multilingual support and improve the summarization of complex email threads. Lastly, real-time scalability and the ability to customize summaries for individual users are areas identified for future improvement and research.

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

With the exponential growth in daily email volume, users often face information overload, reduced productivity, and stress due to inefficient handling of emails. Existing systems lack intelligent automation features such as tone adjustment, content summarization, and priority-based filtering, resulting in increased manual effort and missed critical messages. There is a clear need for a smart, customizable email management tool that can automate routine tasks and enhance communication efficiency using AI.

3.1.2 Existing System

Traditional email clients such as Gmail and Outlook offer basic filtering and labeling features but rely heavily on manual input. Tools like sendmail or simple SMTP servers can automate email delivery, yet lack intelligence for summarization, prioritization, or tone optimization. Some platforms incorporate basic automation, but they lack integration with NLP models, do not support context-aware summarization, and are not optimized for command-line usage or AI-based personalization.

3.1.3 Proposed System

The proposed system, MailOps, is an AI-powered command-line based email management tool that integrates Hugging Face NLP models to provide advanced features such as tone conversion (casual to formal), email summarization, subject enhancement, sentiment analysis, and smart filtering. It allows users to send, read, and manage emails with enhanced automation and contextual understanding via simple CLI commands. The system bridges the gap between traditional email tools and modern AI capabilities, improving productivity, reducing stress, and enabling Additionally, MailOps supports multi-recipient messaging, attachment handling, and custom tagging for improved organization. The AI models continuously learn from user input to improve response quality and personalization over time. Its lightweight, script-friendly design ensures compatibility across platforms, making it ideal for developers, professionals, and teams seeking intelligent email automation.

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.

- **User Authentication:**

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

- **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.

- **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.

- **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.

3.2.2 Non-Functional Requirements:

These are basically the quality constraints that the system must satisfy according to the project contract.

- **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.

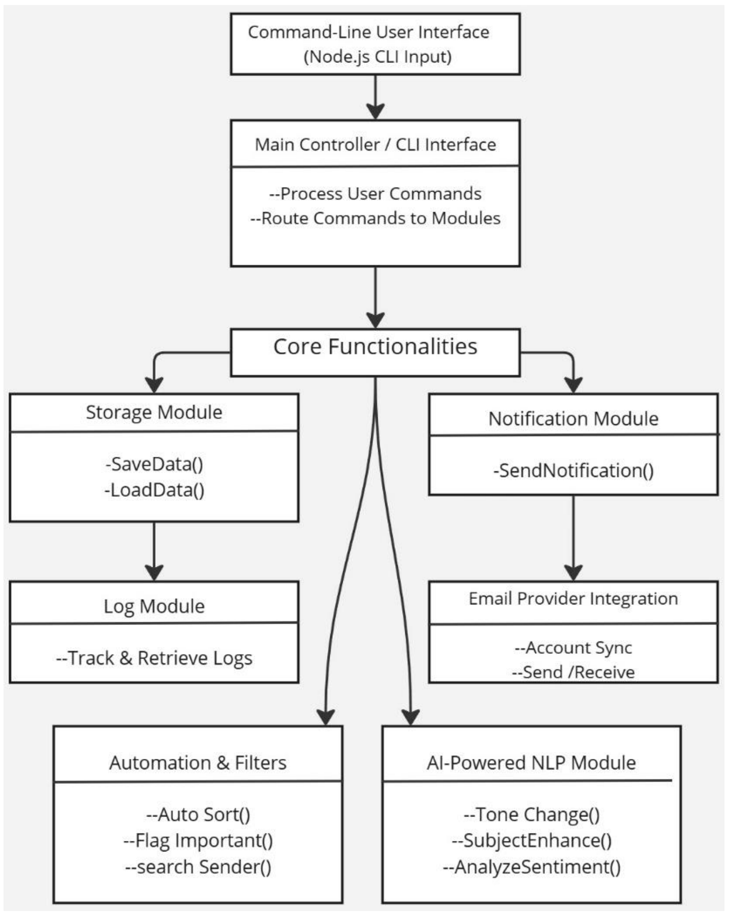


Fig 4.1 Proposed System Architecture

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.

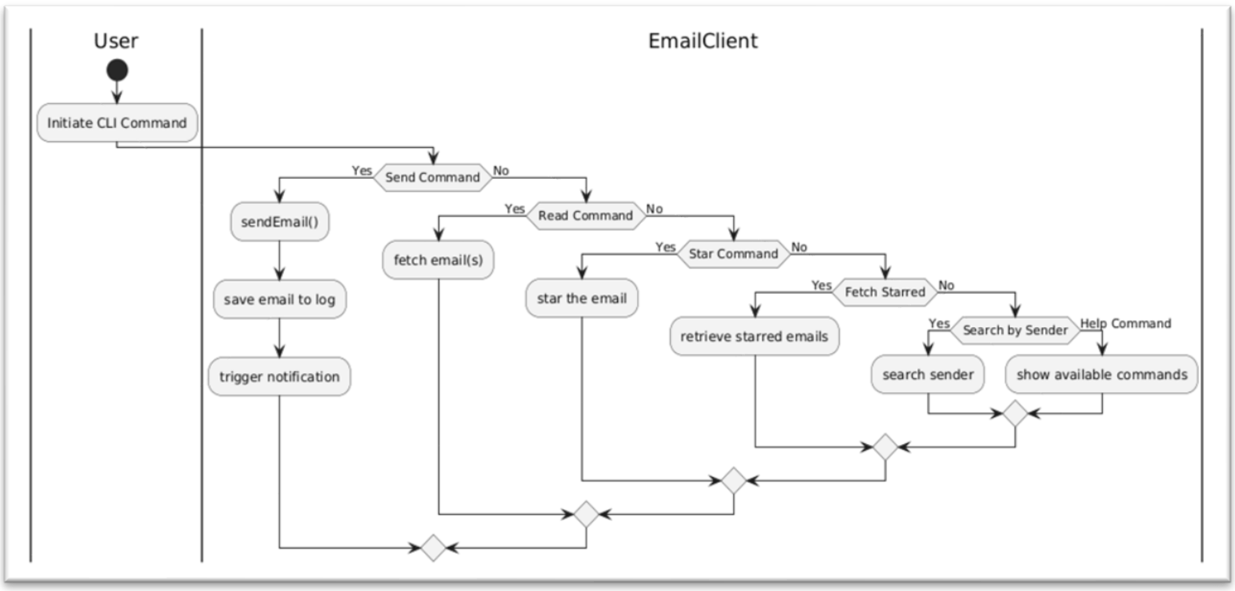


Fig 4.2 Work flow Diagram

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 inone or more interaction with the system.

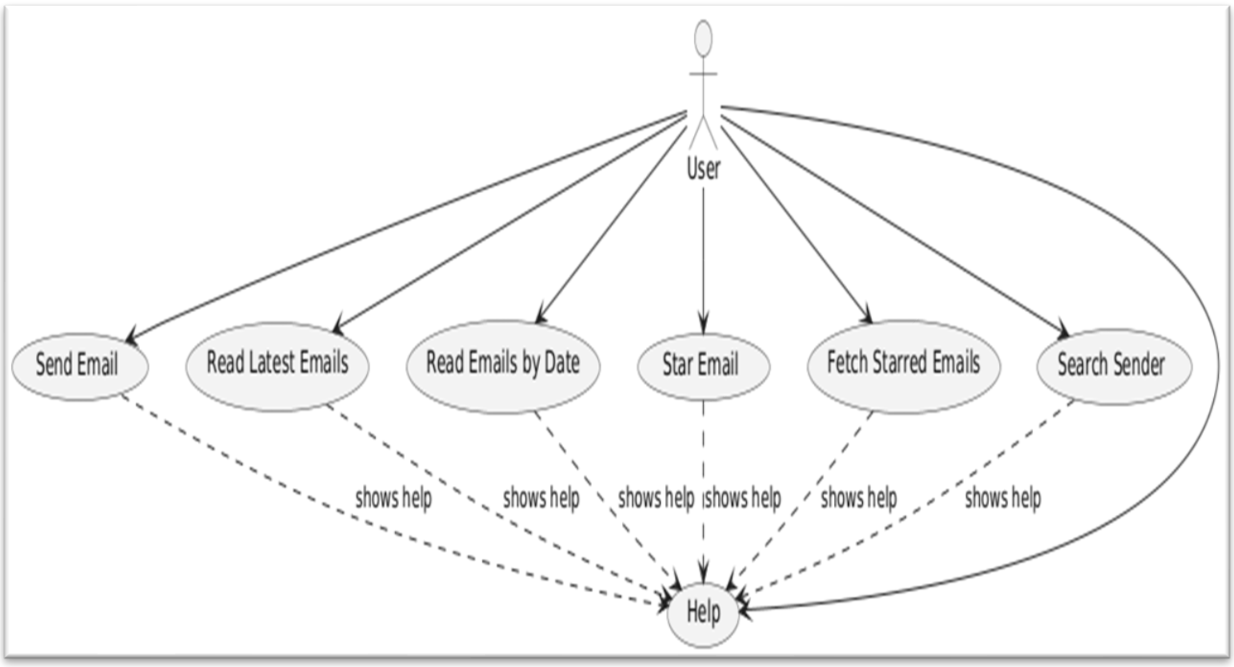


Fig 4.3 Use Case Diagram

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.

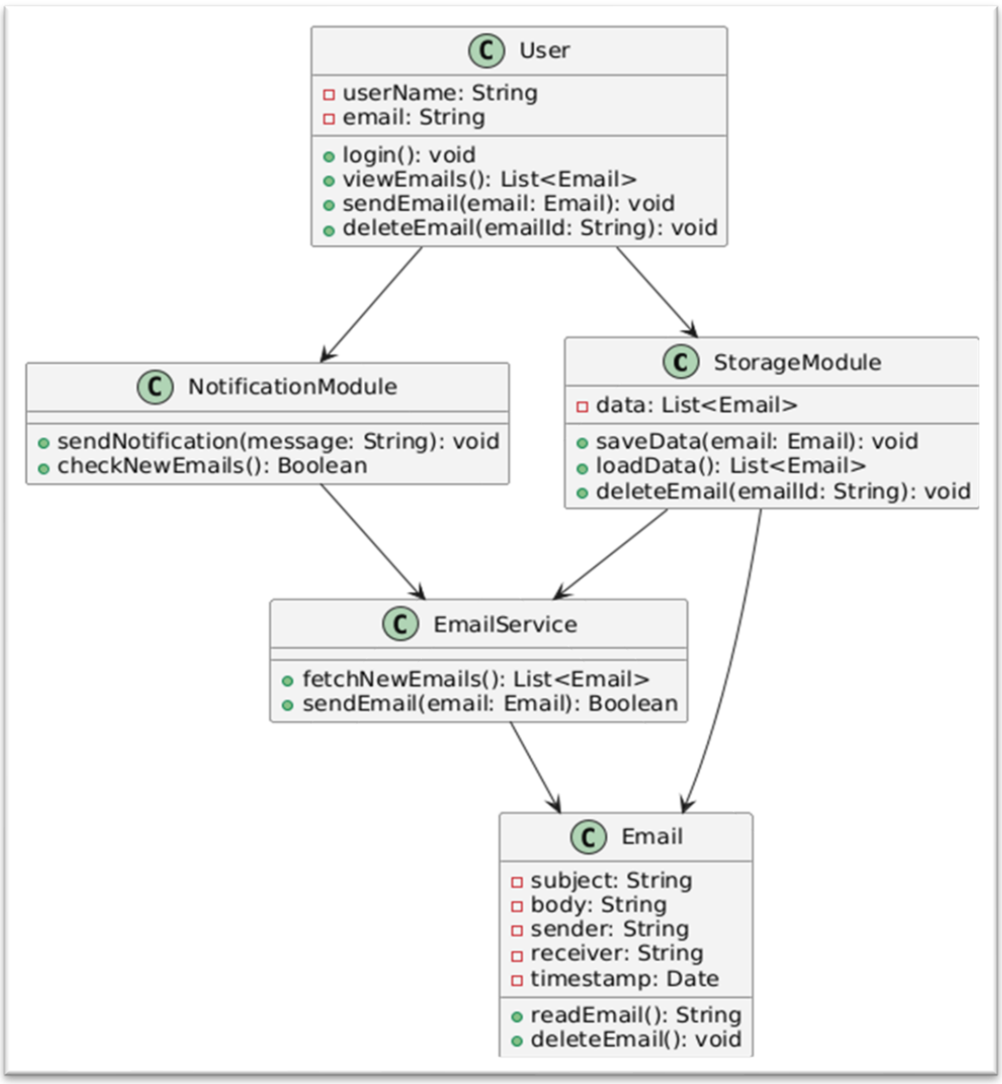

User
userName: String
email: String
login(): void
viewEmails(): List<Email>
sendEmail(email: Email): void
deleteEmail(emailId: String): void
NotificationModule
sendNotification(message: String): void
checkNewEmails(): Boolean
StorageModule
data: List<Email>
saveData(email: Email): void
loadData(): List<Email>
deleteEmail(emailId: String): void
EmailService
fetchNewEmails(): List<Email>
sendEmail(email: Email): Boolean
Email
subject: String
body: String
sender: String
receiver: String
timestamp: Date
readEmail(): String
deleteEmail(): void

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.

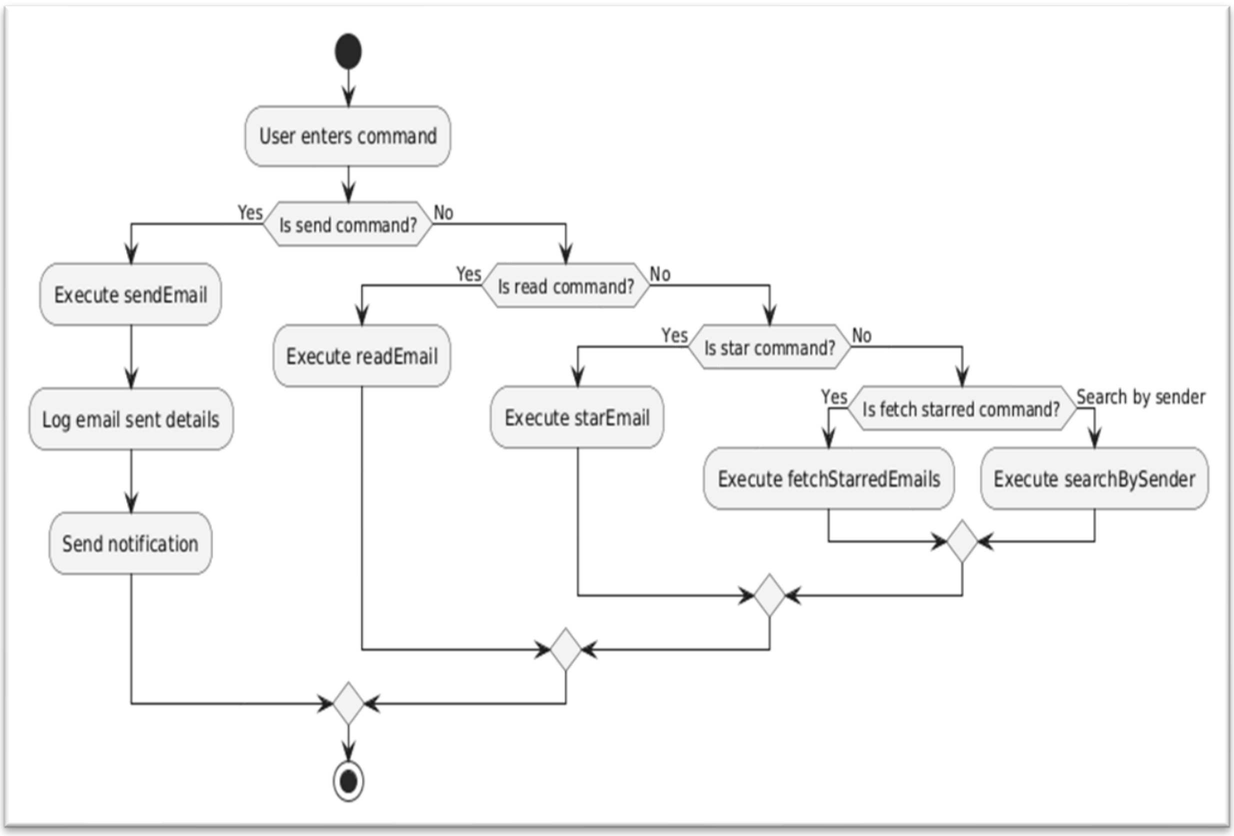


Fig 4.5 Activity Diagram

4.2.5 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.

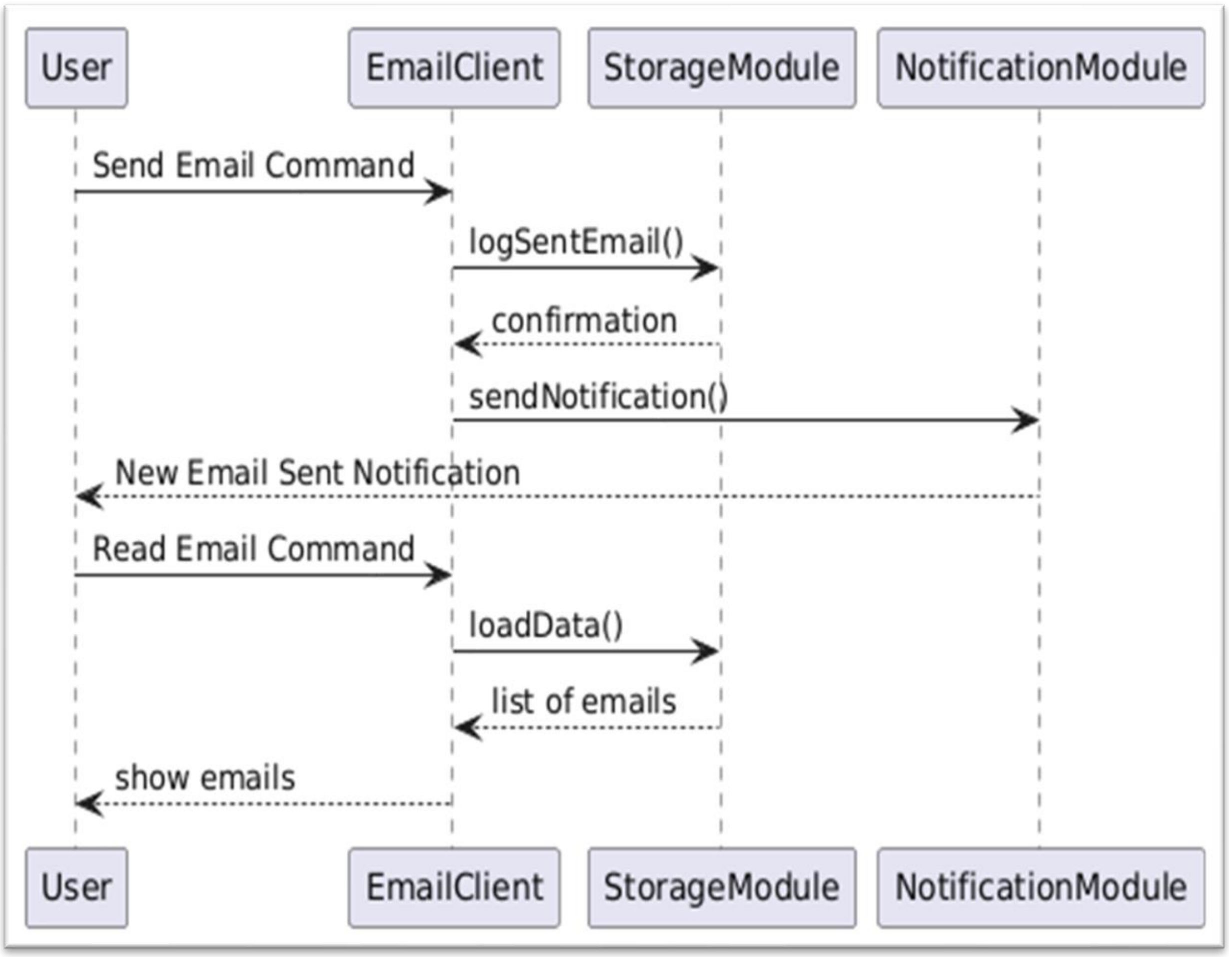


Fig 4.6 Sequence Diagram

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.

The List of Modules are:

- Email Retrieval Module
- Email Sending Module
- Storage Module
- Notification Module

5.2 MODULE DESCRIPTION

5.2.1 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.

5.2.2 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.

5.2.3 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.

5.2.4 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.

5.2.5 AI-Powered NLP Module

This module integrates transformer-based NLP models from Hugging Face to enhance email content understanding and transformation. It provides intelligent features such as tone conversion (casual to formal), subject enhancement, summarization, and sentiment analysis. These tools help users communicate more effectively and process large volumes of information with ease.

- toneChange(): Converts informal email content into formal tone.
- subjectEnhance(): Generates improved subject lines from email content.
- summarizeEmail(): Summarizes long email threads into key points.
- analyzeSentiment():Evaluates the emotional tone of an email (positive, negative, neutral).

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

The test plan ensures all MailOps-CLI functionalities work as intended. Testing covers unit, integration, and system tests to verify commands like send, read, star and AI Enhancer. Key scenarios include sending emails, reading by date, and storing logs. Performance tests check for responsiveness under load, and user acceptance tests ensure usability. Testing is conducted across Linux, Mac, and Windows in the CLI environment. Success is defined as all commands functioning smoothly and accurately.

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 Procedre	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 acr oss 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 corr ectly.

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: AI-Powered Email Management Tool have yielded significant results, demonstrating substantial improvements in automating and enhancing email communication workflows. The updated tool integrates core functionalities such as sending, receiving, smart filtering, and managing emails through a command-line interface, now enriched with AI-powered capabilities including tone conversion, subject enhancement, email summarization, and sentiment analysis.

Performance evaluations indicate that the system handles bulk operations efficiently, with AI tasks such as sentiment analysis and tone adjustments executing quickly with minimal latency. The integration of Hugging Face NLP models significantly enhanced the tool's ability to process and transform email content intelligently, providing users with more context-aware communication options. Early user feedback highlighted the usefulness of the tone-change and subject-enhancement features, noting that it helped maintain professional communication standards effortlessly.

The tool's CLI-based interaction model, supported by clear documentation and intuitive command structures, enabled users to adopt AI features without a steep learning curve. Rigorous testing confirmed effective error handling, with the system providing informative error messages for invalid operations and maintaining robustness across various scenarios. Integration with multiple email providers remained seamless, while the added AI functionality introduced more personalized, intelligent handling of emails.

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.

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 agreem ent 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 appli cation by running: node email-cli.js <Commands>

Step 6: Observe the terminal output to verify that your project is run ning correctly. If you encounter any errors, review the messa ges 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. 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 improve its functionality, usability, and performance, aligning it with emerging user needs and advancements in email management. Introducing automated email responses would allow users to configure responses based on criteria like keywords or sender information, improving efficiency during high-volume periods. Advanced analytics tracking metrics such as open rates and response times could help users refine their email habits, while machine learning-based categorization and smart filtering would streamline incoming email organization. Adding a graphical user interface (GUI) could enhance accessibility for users who prefer visual interaction, broadening the tool's appeal. Integrations with productivity tools like calendars, task managers, and note-taking apps would allow seamless synchronization with users' broader workflows. Enhanced security features, such as end-to-end encryption, spam filtering, and phishing detection, would provide additional protection against evolving threats. Expanding multi-platform support, including mobile, could further increase the user base by allowing on-the-go email management. Establishing a community forum or support system would foster a collaborative environment, enabling users to share tips, report issues, and request features. By focusing on these enhancements, MailOps-CLI can evolve into a comprehensive, adaptable solution for efficient email management in a dynamic digital landscape.

APPENDIX-I

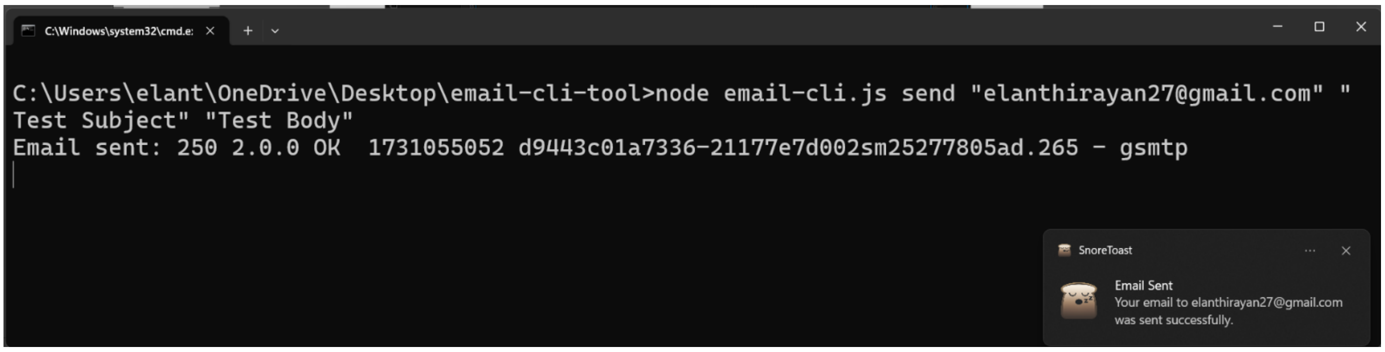
BASE PAPER

APPENDIX-II

SCREENSHOTS

MAILOPS FUNCTIONS

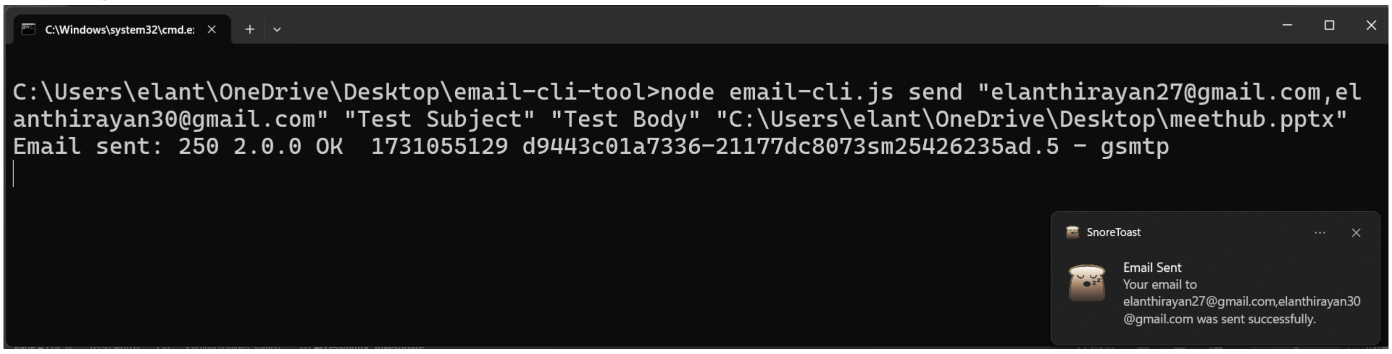
Sending Mail Single or Multiple people



```
C:\Users\elant\OneDrive\Desktop\email-cli-tool>node email-cli.js send "elanthirayan27@gmail.com" "
Test Subject" "Test Body"
Email sent: 250 2.0.0 OK  1731055052 d9443c01a7336-21177e7d002sm25277805ad.265 - gsmtp
```

SnoreToast
Email Sent
Your email to elanthirayan27@gmail.com was sent successfully.

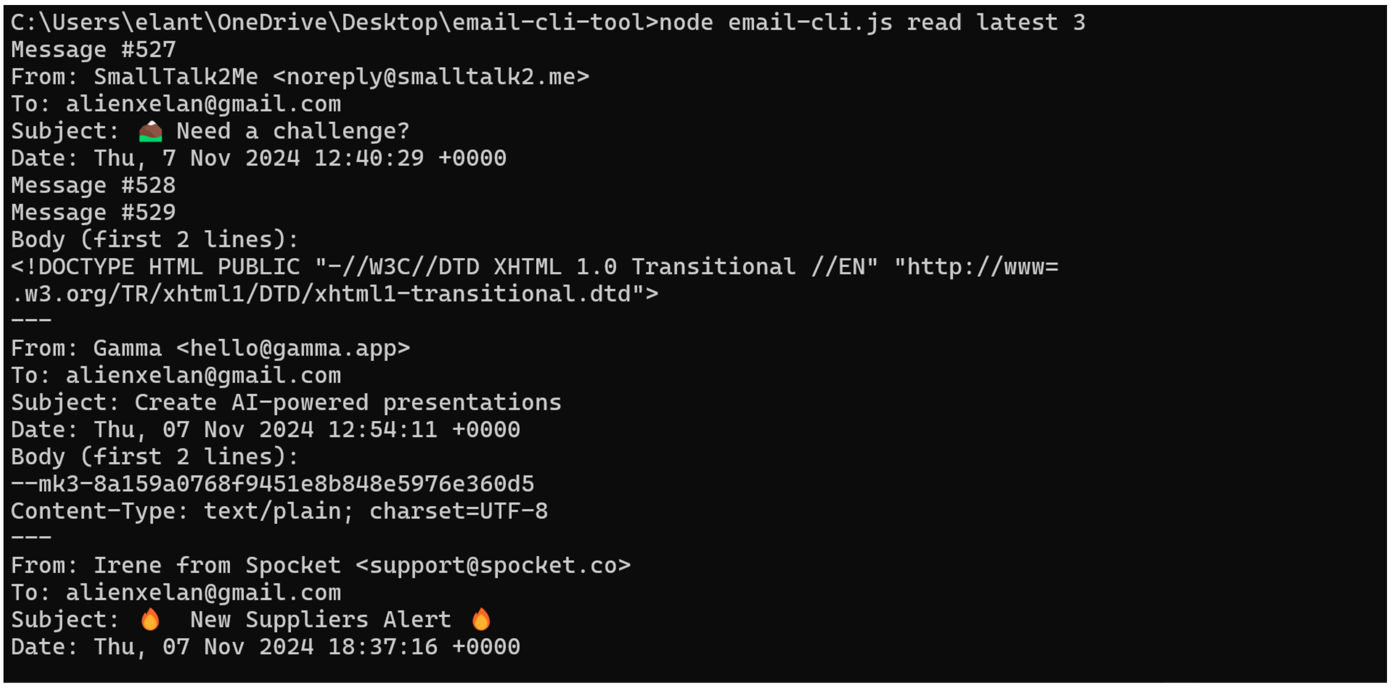
Sending mail with Attachments



```
C:\Users\elant\OneDrive\Desktop\email-cli-tool>node email-cli.js send "elanthirayan27@gmail.com,el
anthirayan30@gmail.com" "Test Subject" "Test Body" "C:\Users\elant\OneDrive\Desktop\meethub.pptx"
Email sent: 250 2.0.0 OK  1731055129 d9443c01a7336-21177dc8073sm25426235ad.5 - gsmtp
```

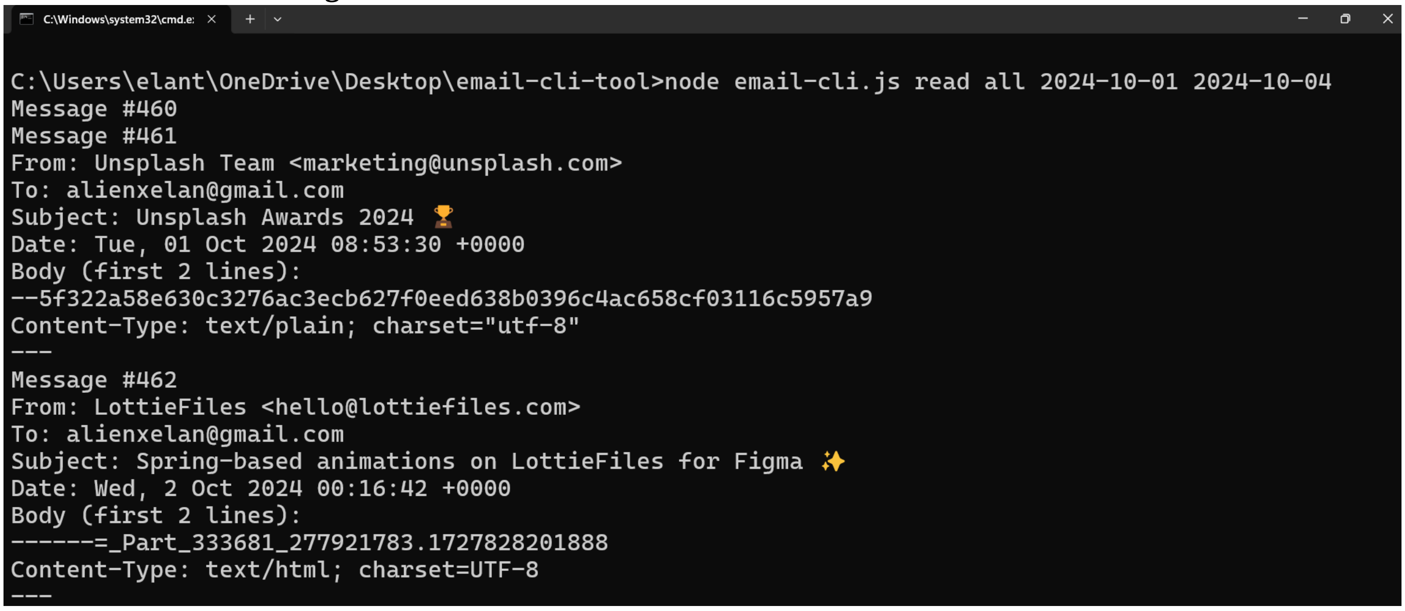
SnoreToast
Email Sent
Your email to elanthirayan27@gmail.com,elanthirayan30@gmail.com was sent successfully.

Read recent mails:



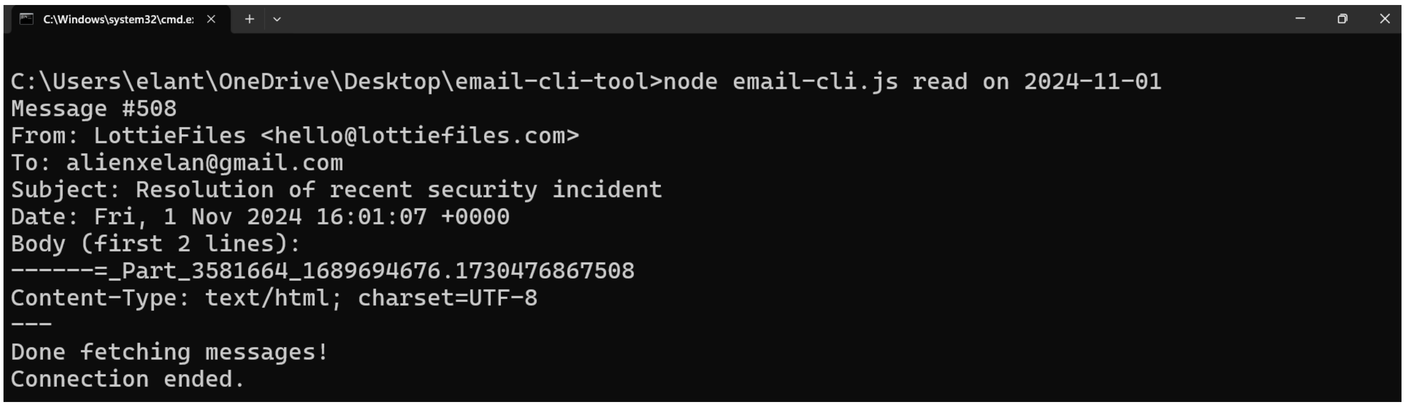
```
C:\Users\elant\OneDrive\Desktop\email-cli-tool>node email-cli.js read latest 3
Message #527
From: SmallTalk2Me <noreply@smalltalk2.me>
To: alienxelan@gmail.com
Subject: 🧗 Need a challenge?
Date: Thu, 7 Nov 2024 12:40:29 +0000
Message #528
Message #529
Body (first 2 lines):
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Transitional //EN" "http://www=
.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
---
From: Gamma <hello@gamma.app>
To: alienxelan@gmail.com
Subject: Create AI-powered presentations
Date: Thu, 07 Nov 2024 12:54:11 +0000
Body (first 2 lines):
--mk3-8a159a0768f9451e8b848e5976e360d5
Content-Type: text/plain; charset=UTF-8
---
From: Irene from Spocket <support@spocket.co>
To: alienxelan@gmail.com
Subject: 🔥  New Suppliers Alert 🔥
Date: Thu, 07 Nov 2024 18:37:16 +0000
```

Read Mails on Date Range



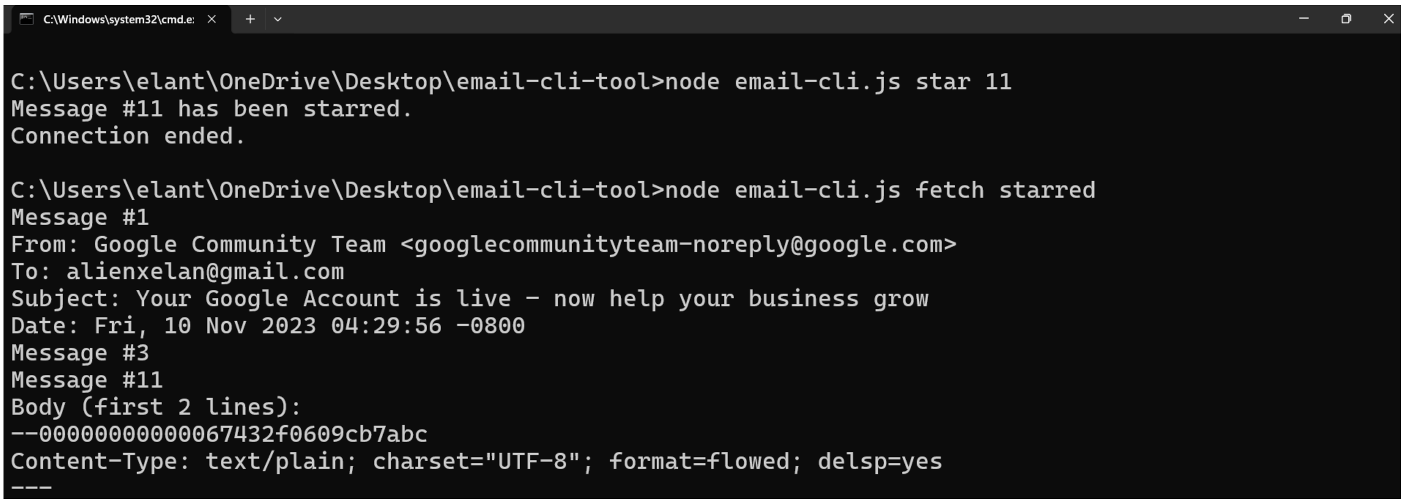
```
C:\Users\elant\OneDrive\Desktop\email-cli-tool>node email-cli.js read all 2024-10-01 2024-10-04
Message #460
Message #461
From: Unsplash Team <marketing@unsplash.com>
To: alienxelan@gmail.com
Subject: Unsplash Awards 2024 🏆
Date: Tue, 01 Oct 2024 08:53:30 +0000
Body (first 2 lines):
--5f322a58e630c3276ac3ecb627f0eed638b0396c4ac658cf03116c5957a9
Content-Type: text/plain; charset="utf-8"
---
Message #462
From: LottieFiles <hello@lottiefiles.com>
To: alienxelan@gmail.com
Subject: Spring-based animations on LottieFiles for Figma ✨
Date: Wed, 2 Oct 2024 00:16:42 +0000
Body (first 2 lines):
------=_Part_333681_277921783.1727828201888
Content-Type: text/html; charset=UTF-8
---
```

Read Mails on Date



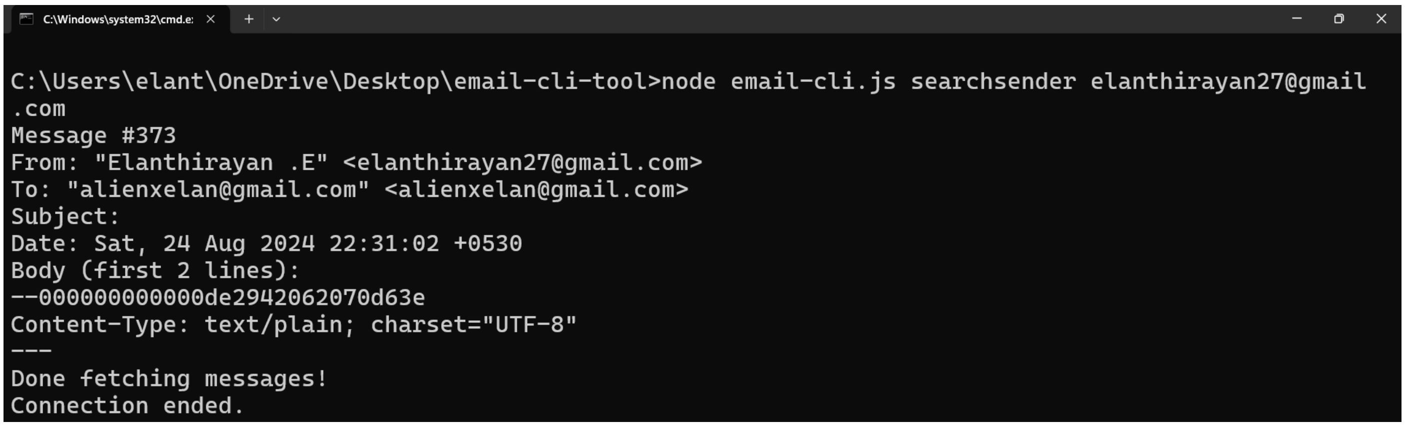
```
C:\Users\elant\OneDrive\Desktop\email-cli-tool>node email-cli.js read on 2024-11-01
Message #508
From: LottieFiles <hello@lottiefiles.com>
To: alienxelan@gmail.com
Subject: Resolution of recent security incident
Date: Fri, 1 Nov 2024 16:01:07 +0000
Body (first 2 lines):
------=_Part_3581664_1689694676.1730476867508
Content-Type: text/html; charset=UTF-8
---
Done fetching messages!
Connection ended.
```

Set & Fetch Star Mails



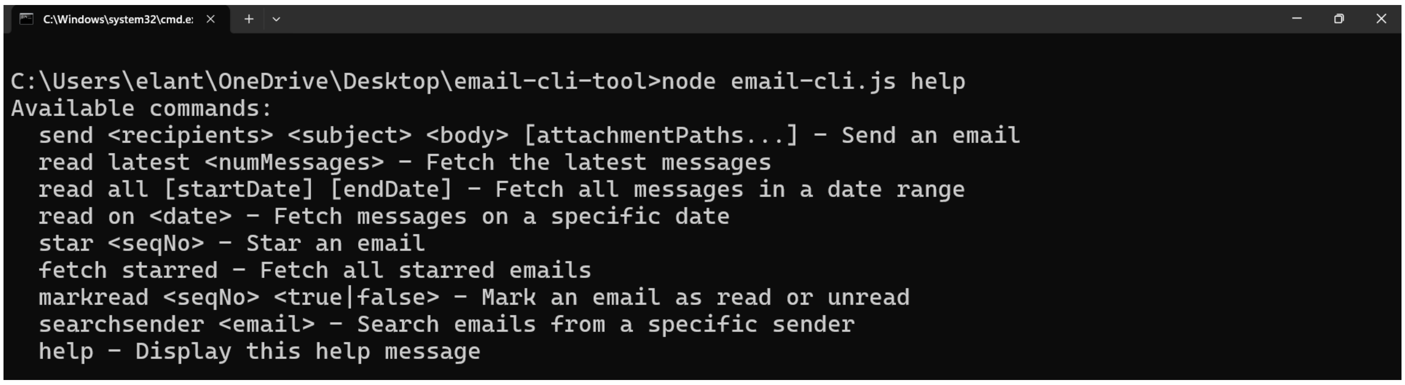
```
C:\Users\elant\OneDrive\Desktop\email-cli-tool>node email-cli.js star 11
Message #11 has been starred.
Connection ended.

C:\Users\elant\OneDrive\Desktop\email-cli-tool>node email-cli.js fetch starred
Message #1
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
Message #3
Message #11
Body (first 2 lines):
--00000000000067432f0609cb7abc
Content-Type: text/plain; charset="UTF-8"; format=flowed; delsp=yes
---
```

Search Sender by Mail

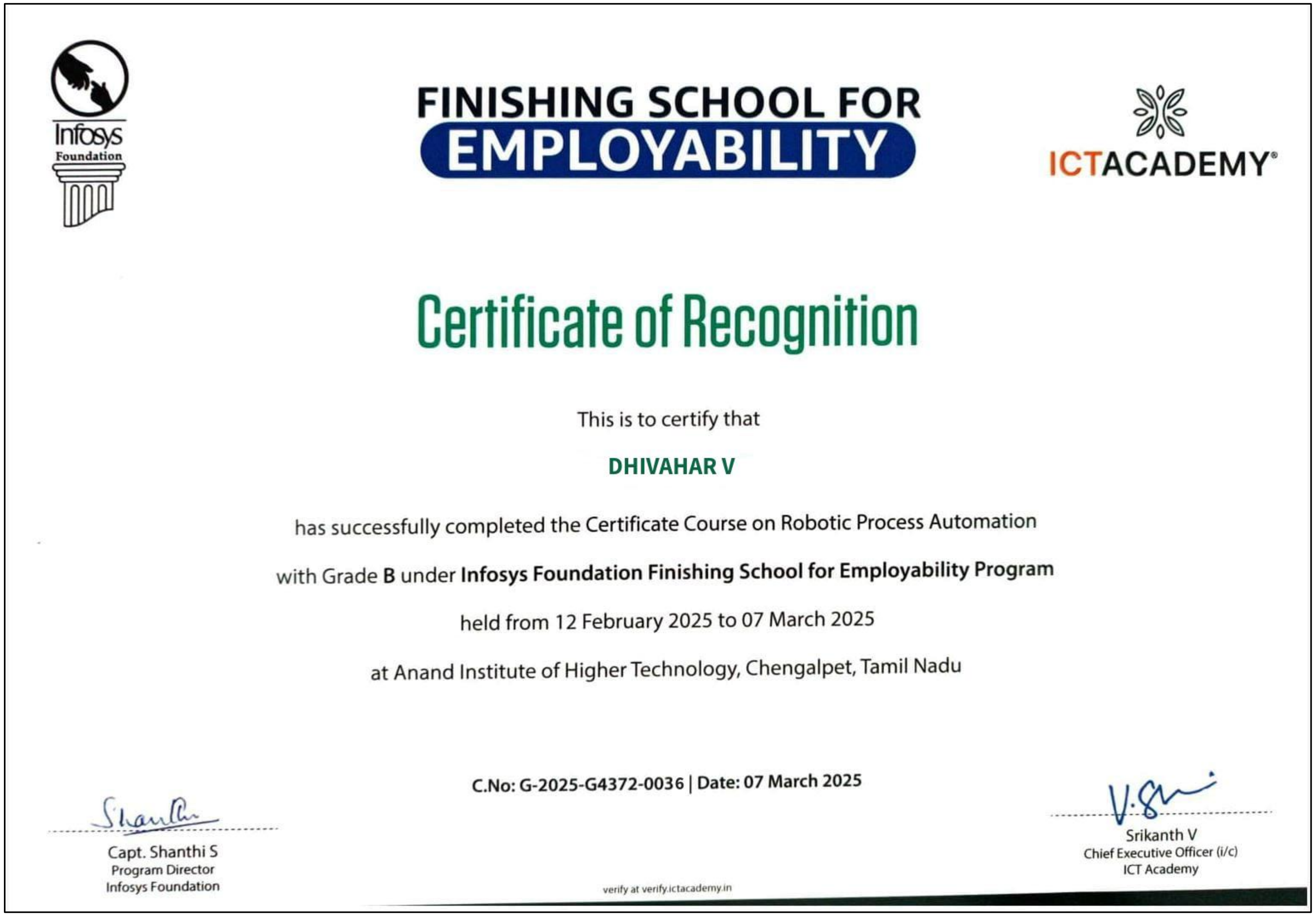


```
C:\Users\elant\OneDrive\Desktop\email-cli-tool>node email-cli.js searchsender elanthirayan27@gmail
.com
Message #373
From: "Elanthirayan .E" <elanthirayan27@gmail.com>
To: "alienxelan@gmail.com" <alienxelan@gmail.com>
Subject:
Date: Sat, 24 Aug 2024 22:31:02 +0530
Body (first 2 lines):
--000000000000de2942062070d63e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

Help Commands



```
C:\Users\elant\OneDrive\Desktop\email-cli-tool>node email-cli.js help
Available commands:
  send <recipients> <subject> <body> [attachmentPaths...] - Send an email
  read latest <numMessages> - Fetch the latest messages
  read all [startDate] [endDate] - Fetch all messages in a date range
  read on <date> - Fetch messages on a specific date
  star <seqNo> - Star an email
  fetch starred - Fetch all starred emails
  markread <seqNo> <true|false> - Mark an email as read or unread
  searchsender <email> - Search emails from a specific sender
  help - Display this help message
```



Infosys Foundation

FINISHING SCHOOL FOR EMPLOYABILITY

ICTACADEMY®

Certificate of Recognition

This is to certify that

DHIVAHAR V

has successfully completed the Certificate Course on Robotic Process Automation

with Grade **B** under **Infosys Foundation Finishing School for Employability Program**

held from 12 February 2025 to 07 March 2025

at Anand Institute of Higher Technology, Chengalpet, Tamil Nadu

C.No: G-2025-G4372-0036 | Date: 07 March 2025

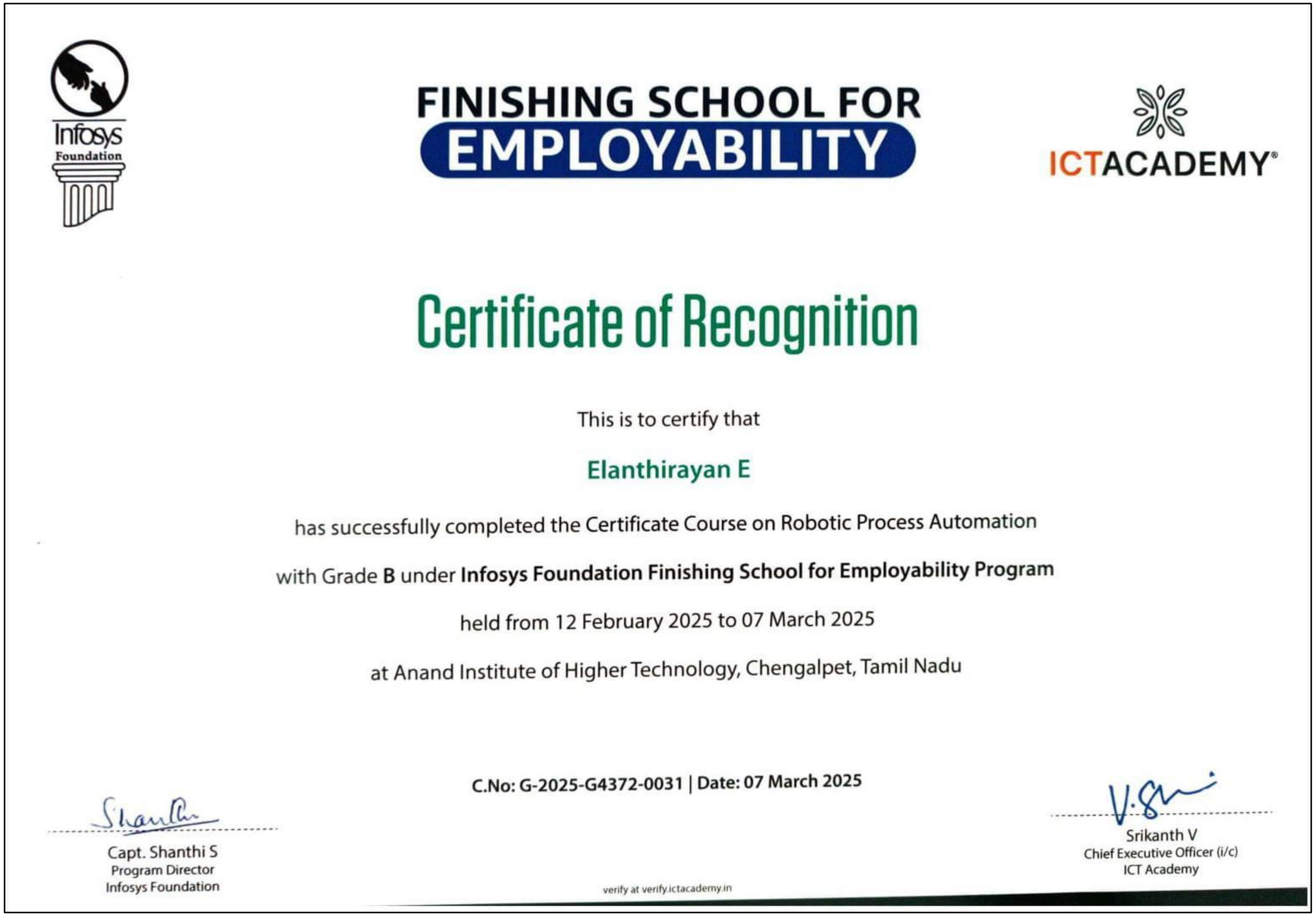
[illegible]

Capt. Shanthi S
Program Director
Infosys Foundation

[illegible]

Srikanth V
Chief Executive Officer (i/c)
ICT Academy

verify at verify.ictacademy.in



Infosys Foundation

FINISHING SCHOOL FOR EMPLOYABILITY

ICTACADEMY®

Certificate of Recognition

This is to certify that

Elanthirayan E

has successfully completed the Certificate Course on Robotic Process Automation

with Grade **B** under **Infosys Foundation Finishing School for Employability Program**

held from 12 February 2025 to 07 March 2025

at Anand Institute of Higher Technology, Chengalpet, Tamil Nadu

C.No: G-2025-G4372-0031 | Date: 07 March 2025

[illegible]

Capt. Shanthi S
Program Director
Infosys Foundation

[illegible]

Srikanth V
Chief Executive Officer (i/c)
ICT Academy

verify at verify.ictacademy.in

REFERENCES

1. Cecchinato, M. E. and Bird, J. Cox, A. L. "Personalised email tools solution to email overload?" in Personalizing Behavior Change Technologies : CHI (2024) Workshop, ACM Conference on Human Factors in Computing Systems (CHI): Toronto, Canada.

2. J. Doe and R. Smith, "Email Management and Strategies for Improve ment, "TechPublishing,2020,doi:10.1016/j .techpub.2020.04.001.

3. Hu, L. "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.

4. Kachole, A. "Email Management Platform," International Journal for Research in Applied Science and Engineering Technology, vol. 11, no. 2, pp. 456-461, (2024).

5. Letmathe, P. and Noll, E. "Analysis of email management strategies their effects on email management performance," RWTH Aachen University, (2022), doi: 10.1016/j.jcjp.2023.102411.

6. T. Valentine, ‘Scripted Email: Using sendmail, "in (2023), pp 161-169, doi: 10.1007/978-1-4842-6460-3_11.

7. 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).