MESSAGER APP PROJECT

Andres Ramos*
*Department of Computer Engineering
Universidad Distrital Francisco José de Caldas, Bogotá, Colombia
Email: ancramosr@udistrital.edu.co

***Abstract*—Current digital communication via email clients is hampered by information overload and distractions, making it difficult to manage private and focused conversations. To address this problem, we introduce "Messager App," a private messaging application developed using the object-oriented programming paradigm, offering a dedicated communication channel free from distractions such as ads or subscriptions. The project culminated in a working prototype with an intuitive user interface, successfully validating the concept of a closed messaging system and demonstrating its viability for effective communication.**

***Index Terms*—Project guidelines, IEEE LaTeX, engineering education, templates**

I. Introduction

Email remains a cornerstone of modern digital communication, facilitating billions of professional and personal exchanges daily. However, its widespread adoption has led to its use for a multitude of purposes, creating a significant challenge: information overload. The contemporary email inbox has evolved into a repository for everything from critical business correspondence and legal notices to promotional materials, social media notifications, and automated security alerts. While services like Outlook[1] and Gmail[2] provide spam filtering, users often hesitate to block certain senders who, despite sending marketing content, may also deliver important account-related information, such as invoices or password resets. This entanglement of essential and non-essential messages makes it difficult for individuals to prioritize their attention, often resulting in important communications being overlooked in a saturated inbox.

This project directly addresses the problem of ineffective communication caused by cluttered digital environments. In both professional and personal contexts, there is a clear need for a focused communication channel that is not subject to the constant influx of unsolicited content. While voice calls offer an alternative, they are limited by the mutual availability of participants and lack a persistent, searchable history of the conversation, which is often crucial for accountability and reference. Furthermore, popular messaging applications like WhatsApp, while ubiquitous, often blur the lines between personal and professional life, and many users are uncomfortable using a single platform for all interactions. Therefore, we propose the development of a private messaging application designed specifically for targeted communication, allowing users to create a distinct space for work or other specific purposes, thereby avoiding the inconvenience of content mixing and digital noise.

This work was funded by *Your Funding Source* under grant No. 12345.

Existing solutions, such as the major email providers (Outlook, Gmail, Yahoo [3]), offer powerful and versatile platforms. However, their core design as open-ended communication tools is also their primary weakness in this context. These are the same email addresses we use to register for online services, subscribe to newsletters, and engage with commercial websites. As a result, our inboxes are automatically populated with a mix of user-initiated communication and system-generated messages, making it nearly impossible to maintain a clear hierarchy of importance. Our proposed solution fundamentally differs by creating a closed-loop system. This application will be a dedicated environment for timely and effective communication, intentionally designed to prevent registrations on external websites or subscriptions. The system will be free from distractions, ensuring that the only content a user receives is direct messages from other known users within the network.

This article details the design and development of this messaging application project—an email-like platform engineered to transform digital interaction by providing a streamlined and user-friendly experience. Our goal is to empower users to avoid mixing disparate types of content by offering a single-purpose messaging service strictly for interpersonal communication, devoid of promotions or automated account alerts. The technical foundation for this solution is Object-Oriented Programming (OOP) [4]. The initial design phase yielded a theoretical prototype, leading to the definition of core classes and their attributes. Based on a focused approach to select only what was essential, we identified the need for a User class, a Email class. This structure was chosen to ensure a logical organization of conversations and user data without requiring complex database management for the prototype, setting a clear path for the subsequent development and implementation of the application.

II. Methods and Materials

1) Analysis and Architectural Foundation: From its inception, the project was architecturally grounded in the

object-oriented programming (OOP) paradigm. This was a strategic decision, not merely a choice of style, but a commitment to building a system that was inherently modular, scalable, and maintainable over the long term. The core tenets of OOP provided direct benefits. Encapsulation, for example, allowed us to bundle data (attributes) with the functions (methods) that operate on them, creating robust, self-contained classes that protected their internal state from outside interference. This structure is instrumental in simplifying development and debugging, as it effectively isolates functionalities, reduces complex interdependencies, and minimizes the ripple effect of any future changes.

Following this paradigm selection, we initiated a critical abstraction process to distill the application's essential purpose. The primary goal was to filter out superfluous features and concentrate exclusively on the core functionality required to solve the problem of cluttered communication. This involved a rigorous inquiry into the system's absolute necessities, posing fundamental questions like, "What is the minimum set of attributes needed to uniquely identify a user and their state?" and "What elemental components constitute a single, complete message?" This disciplined analysis led us to the conclusion that the entire system could be effectively and efficiently modeled with just two primary classes: a User class, responsible for managing the registration, authentication, and identification of each individual with system access, and an Email class, designed to encapsulate and store the content, metadata, and state of messages exchanged between users. In accordance with OOP design principles, each class was meticulously conceptualized with its own distinct attributes and behaviors, ensuring the correct and independent functionality of each component. With this lean foundational architecture established, we proceeded to define the project's functional and non-functional requirements. This step was crucial for creating a clear and unambiguous roadmap, outlining specific deliverables, which provided a clear mandate for all subsequent development phases.

2) System Design and Modeling: Moving into the design phase, our focus shifted to translating the abstract requirements from the analysis into a concrete, detailed technical blueprint. For this, we relied heavily on the industry-standard Unified Modeling Language (UML) to visually represent the system's architecture and behavior. This formal design process began with the creation of Class-Responsibility-Collaboration (CRC) cards for our proposed User and Email classes. This hands-on exercise proved invaluable for validating our initial class structure and for clearly delineating the responsibilities of each entity. For instance, it solidified that the User class was solely responsible for managing credentials and its associated message collections, while the Email class was responsible only for holding the data of a single message.

With these responsibilities clarified, we developed a detailed class diagram that served as the static architectural guide for the entire development process. This diagram meticulously detailed the specific methods for each class. Critically, it also specified the relationships between them—a one-to-many relationship where a single User could possess collections of Email objects. To complement this static view, we designed sequence and activity diagrams to map out the system's dynamic behavior and user flow. These UML diagrams illustrated the precise, step-by-step interactions expected within the system, modeling use cases like a user logging in, composing a message, and successfully sending it. This practice was instrumental in anticipating and mitigating risks, allowing us to cover all expected pathways and minimize the potential for unexpected runtime errors. The final design solidified the two-class model, where the Email class contained standard attributes like subject, content, and sender/receiver information, and the User class held basic user data, with the username serving as the key identifier for message routing. The resulting diagrams outlined a straightforward interaction flow, simplifying the system by focusing exclusively on text-based exchange and preventing overly complex user actions that could lead to system failures.

3) Implementation, Refactoring, and Testing: The coding phase represented the most tangible stage of progress, where our theoretical designs were translated into a functional application. This phase proved to be extensive and highly iterative, primarily because the process of implementation revealed unforeseen complexities and flaws in our initial design. We discovered that some of the classes originally proposed in the early UML diagrams, such as separate Inbox and Outbox managers, were superfluous and unnecessarily complicated the application's logic. Consequently, a significant refactoring effort was undertaken to streamline the architecture down to the two essential User and Email classes. This involved revising their methods and responsibilities and introducing a main controller class, implemented using a Singleton design pattern, to manage the in-memory storage of all User and Email objects. This controller acted as a centralized, temporary data store, preventing data loss during a runtime session and simplifying object retrieval across the application. These critical adjustments were made in close consultation with the project supervisor, ensuring the final product was both functional and architecturally sound.

Once the core logic was deemed robust and complete, our focus shifted to enhancing the user interface. The initial console-based interaction, while functional for testing, was not intuitive for an end-user. Therefore, we developed a simple graphical interface designed to lower the user's cognitive load and guide them through each step of the messaging process, thereby minimizing the potential for input errors. The development phase concluded with a multi-tiered testing strategy. This included unit tests to verify the correctness of individual methods within each class, integration tests to ensure that the end-to-end flow of sending and receiving messages worked seamlessly, and informal user acceptance

testing to gather feedback on the usability and clarity of the new graphical interface. This rigorous process ensured the final application was stable, reliable, and met its core objectives.

III. Results

The primary outcome of this project is a functional application prototype that successfully realizes the initial design goals. The resultant codebase establishes a self-contained, closed-loop messaging environment where registered users can securely and directly exchange messages. Comprehensive internal testing has validated the core functionalities of the system. These tests confirm that message transmission between different registered users is reliable and operates as intended, allowing for a focused and efficient communication channel.

This working prototype serves as a successful validation of the project's central concept: that by eliminating external digital noise, communication can become more effective. The system inherently promotes simplicity because users only receive messages pertinent to their direct interactions, from contacts they have personally approved within the network. This stands in stark contrast to conventional email, where inboxes are saturated with a mixture of crucial and non-essential content.

At the current stage, a direct quantitative comparison with established solutions like Gmail or Outlook is not feasible, as the application has not been deployed in a real-world business or organizational setting. Therefore, there are no empirical figures to definitively measure the reduction in message volume a user would experience. However, a substantial difference is logically projected. The application's architecture, by design, prevents the mass sending of information via distribution lists, third-party subscriptions, or promotional blasts. The main constraint of the system—its closed nature—is thus also its greatest strength. To enhance usability and provide greater clarity for end-users, the functional backend is complemented by a clean and intuitive graphical user interface developed in Swift, ensuring a straightforward and error-free user experience.

IV. Conclusions

The "Messager App" project was initiated to address the significant problem of information overload and distraction inherent in modern email communication. We developed a private, closed-loop messaging application using the object-oriented programming paradigm, focusing on creating a dedicated and secure channel for communication. The project progressed from conceptual analysis and UML design to the development of a functional prototype with an intuitive user interface. The primary result is a working application that successfully enables users to send and receive messages within a controlled environment, free from the advertisements, subscriptions, and spam that clutter conventional email inboxes.

The project is considered a success because it fully realized our initial goal: to create a viable alternative for focused communication. By building a functional prototype, we validated our core concept and demonstrated that a closed messaging system can effectively solve the problem of digital distraction. Our work directly helps users by providing them with a tool to separate important, targeted conversations from the noise of their primary email accounts. This allows for more efficient and private interactions, ensuring that critical messages are not overlooked and that users can maintain distinct communication channels for different aspects of their lives, such as professional and personal matters.

Acknowledgment

References

https://learn.microsoft.com/es-es/outlook/ [1] https://developers.google.com/workspace/gmail/api/reference/rest?hl=es-419 [2] https://developer.yahoo.com/?guccounter=1guce$_referrer = aHR0cHM6Ly93d3cuYmluZy5jb20vguce_referrer_sig = AQAAAMVfCVxWraAkwcyLtEIdxieyIhYqfjvRi2dtqryVkJ3Lj MREj--m1OXV5Usnc71Oq3rKagWqsjCYMQuCL1l7mm2PL- RSvoo - rP1DrGkHy - BD53ODs4u0595meYWB9[3]https : //archive.org/details/objectorientedso00meye_0[4]$