“WEB BASED MAIL CLIENT MODEL”

A

Project Report

submitted

in partial fulfillment

for the award of the Degree of

Bachelor of Technology

in Department of Information Technology

Mentor:
Dr. Astha Joshi
Assistant Prof. IT

Submitted By:
Anushka Sharma
21ESKIT018
Abhyudaya Saraswat
21ESKIT004

Department of Information Technology
Swami Keshvanand Institute of Technology, M & G, Jaipur
Rajasthan Technical University, Kota
Session 2024-2025

Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur

Department of Information Technology

CERTIFICATE

This is to certify that Mr. Abhyudaya Saraswat, a student of B.Tech (Information Technology) 8th semester has submitted his Project Report entitled ”Web Based Mail Client Model” under my guidance.

Mentor	**Coordinator**
Dr. Astha Joshi	Priyanka Yadav
Assistant Prof.-IT	Assistant Prof.

Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur

Department of Information Technology

CERTIFICATE

This is to certify that Ms. Anushka Sharma, a student of B.Tech (Information Technology) 8th semester has submitted her Project Report entitled ”Web Based Mail Client Model” under my guidance.

Mentor	**Coordinator**
Ms. Astha Joshi	Priyanka Yadav
Assiatant Prof.-IT	Assistant Prof.

DECLARATION

We hereby declare that the report of the project entitled ” Web Based Mail Client Model” is a record of an original work done by us at Swami Keshvanand Institute of Technology, Management and Gramothan, Jaipur under the mentorship of ”Ms. Astha Joshi” (Dept. of Information Technology) and coordination of ”Dr. Priyanka Yadav” (Dept.of Information Technology). This project report has been submitted as the proof of original work for the partial fulfillment of the requirement for the award of the degree of Bachelor of Technology (B.Tech) in the Department of Information Technology. It has not been submitted anywhere else, under any other program to the best of our knowledge and belief.

Team Members	**Signature**
Anushka Sharma 21ESKIT018	
Abhyudaya Saraswat 21ESKIT004	

Acknowledgement

A project of such a vast coverage cannot be realized without help from numerous sources and people in the organization. We take this opportunity to express our gratitude to all those who have been helping us in making this project successful.

We are highly indebted to our faculty mentor Ms. Astha Joshi. She has been a guide, motivator source of inspiration for us to carry out the necessary proceedings for the project to be completed successfully. We also thank our project coordinator Dr. Priyanka Yadav for her co-operation, encouragement, valuable suggestions and critical remarks that galvanized our efforts in the right direction.

We would also like to convey our sincere thanks to Dr. Anil Chaudhary, HOD, Department of Information Technology, for facilitating, motivating and supporting us during each phase of development of the project. Also, we pay our sincere gratitude to all the Faculty Members of Swami Keshvanand Institute of Technology, Management and Gramothan, Jaipur and all our Colleagues for their co-operation and support.

Last but not least we would like to thank all those who have directly or indirectly helped and cooperated in accomplishing this project.

Team Members:
Anushka Sharma
21ESKIT018
Abhyudaya Saraswat
21ESKIT004

Table of Content

List of Figures

Chapter 1
Introduction

1.1 Problem Statement and Objective

With growing concerns over data privacy and vendor lock-in, many organizations seek alternative solutions to commercial email services. However, deploying a complete email solution that includes both user-friendly access and administrative tools is often complex, expensive, or lacks customization. Many open-source webmail solutions either rely on deprecated technologies or do not offer robust admin control panels. Furthermore, systems with comprehensive functionality often demand heavy resources or complex configurations that are not suitable for small to mid-sized institutions.

1.2 Literature Survey /Market Survey/Investigation and Analysis

The current email communication landscape is dominated by cloud-based services such as Gmail, Outlook, and Yahoo Mail. While these platforms provide convenience, they often fall short in environments where data confidentiality and internal control are paramount. Cloud services may also introduce risks such as data breaches, vendor lock-in, and limited customization. To counter these issues, various open-source solutions have been developed. Projects like Roundcube and RainLoop offer modern interfaces and support protocols like IMAP and SMTP. However, they typically require complex setups and may not integrate well with existing system-level mail management tools in Linux environments.

SquirrelMail, another once-popular webmail client, is now outdated and no longer maintained, featuring a plain UI and lack of support for modern web standards. On the other hand, solutions like Zimbra and Horde offer robust features but demand significant resources and are better suited for enterprise-level deployments. These tools often introduce additional layers of configuration that can be cumbersome for small organizations.

1.3 Introduction to Project

The proposed project is a browser-based email client application built using Java technologies and hosted on an Apache Tomcat server. It connects to a Linux-based mail server, utilizing standard email protocols and services such as POP3 for message retrieval and Sendmail for dispatching messages. The client allows users to log in using system credentials, check their inbox, read messages, delete unwanted emails, and send new messages. Beyond this, the application includes an administrative panel accessible to system administrators. This panel enables key system functions like adding new email users, resetting passwords, and viewing activity logs—all through a secured web interface.

The user interface is developed using JavaServer Pages (JSP) and Servlets, ensuring that the system is responsive and compatible with most modern web browsers. The backend connects directly with the Linux system using Java libraries or shell scripts to execute system-level commands.

1.4 Proposed Logic / Algorithm / Business Plan / Solution / Device

The proposed solution integrates multiple components working cohesively to deliver a seamless email experience to users. At the heart of the system is a web application built using JSP and Java Servlets, hosted on the Apache Tomcat server. Upon successful login, the user interface displays the inbox by retrieving messages from the Linux mail server using the POP3 protocol.

To send messages, the application collects user input—recipient, subject, message body—and formats the email in compliance with SMTP standards. The message is then passed to the Sendmail program running on the Linux server, which handles the actual delivery to the recipient's mail server. The interaction with Sendmail can be managed via command-line scripts or Java’s Runtime class to execute commands from within the application.

1.5 Scope of the Project

The scope of the project is clearly defined to ensure focused development and deployment. The primary scope includes the creation of a web-based interface that enables users to send, receive, and manage their emails via a Linux-based mail server. The system supports only the POP3 protocol for email retrieval, which is suitable for users who prefer to download and manage their emails locally within the session. For sending emails, the system utilizes the Sendmail program available in most Linux distributions. The interface also supports basic organization features such as message deletion and folder views (if extended to support it).

On the administrative side, the scope includes functionalities such as adding new users, resetting passwords, and maintaining logs. The project also focuses on secure user authentication and session management using Java’s security libraries.

Chapter 2
Software Requirement Specification

2.1 Overall Description

This project is a web-based mail client system designed to work in conjunction with a Linux-based mail server. It allows users to interact with their email accounts via a web browser and enables administrators to perform essential user management operations. The mail client is developed using Java technologies (JSP and Servlets), hosted on an Apache Tomcat server, and interacts with the Linux system for email operations using the POP3 protocol (for receiving emails) and Sendmail (for sending emails).

2.1.1 Product Perspective

2.1.1.1 System Interfaces

The system interfaces with several key components within the Linux environment. It communicates with the Sendmail service to dispatch emails and with the POP3 server to fetch incoming messages. It also interfaces with the Linux shell or system APIs to execute commands related to user management, such as adding new users or resetting passwords. These system-level interfaces are invoked securely from within the application’s backend using Java libraries or custom shell scripts.

.

2.1.1.2 User Interfaces

The user interface is web-based and designed using HTML, CSS, and JavaScript, with dynamic content rendered through JSP pages and Servlets. The application provides two types of interfaces—one for general users and one for administrators. The user interface includes screens for logging in, viewing inboxes, reading individual emails, composing and sending messages, and deleting messages. The administrator interface includes secure login authentication and features for managing users, such as adding accounts, resetting passwords, and viewing system logs.

2.1.1.3 Hardware Interfaces

The server hosting the application should be a standard Linux-based system equipped with a network interface card (NIC) for LAN or internet connectivity. The server must be capable of running the Apache Tomcat server, Sendmail, and POP3 services concurrently. A minimum of 4 GB RAM and a dual-core processor is recommended for handling moderate user load efficiently.

2.1.1.4 Software Interfaces

The software components used in this project include the Java Runtime Environment (JRE), Apache Tomcat server for hosting the web application, and the Sendmail program for managing outgoing -

mail. Additionally, a POP3 server like Dovecot is required to serve incoming messages from the user's mailbox. The web application uses Java Servlets and JSP to generate dynamic web content and manage user sessions.

2.1.1.5 Communications Interfaces

The primary mode of communication between users and the application is through the HTTP or HTTPS protocol, depending on whether SSL is configured. Users access the application through a web browser by sending HTTP requests, which are handled by the Tomcat server. The application communicates with the POP3 server using the POP3 protocol to retrieve emails, and uses SMTP standards via the Sendmail interface to send messages.

2.1.1.6 Memory Constraints

Memory usage considerations apply primarily to the server side. The Linux server hosting the application should have a minimum of 4 GB of RAM to efficiently run the Tomcat server alongside Sendmail and the POP3 service. Disk space requirements will vary based on the number of users and volume of emails but should begin at a minimum of 20 GB to accommodate user mail directories and log files.

2.1.1.7 Operations

The application is designed to run as a continuously available service. Once the server is booted, the Tomcat service should be initialized, and the web application deployed automatically or manually through the Tomcat manager. The mail server components, such as Sendmail and the POP3 service, must also be active. The system supports concurrent user sessions, allowing multiple users to interact with their mailboxes simultaneously. Administrators can log in at any time to perform user management tasks or check.

2.1.1.8 Project Functions

The system offers a set of clearly defined functions. Regular users can log into the web interface, view their inbox, read messages, delete unnecessary emails, and compose new messages. All these operations are routed through the backend, which interacts with the Linux mail server.

2.1.1.9 User Characteristics

The primary users of the system are classified into two categories: general users and administrators. General users are expected to have basic computer and internet usage skills. They should be able to navigate a web browser, log into their account, and perform basic email operations without requiring training. Administrators, on the other hand, are technically proficient users who understand Linux systems and command-line operations.

2.1.1.10 Constraints

The project operates under a few important constraints. It supports only the POP3 protocol for retrieving emails, meaning synchronization of messages across devices is not inherently supported as it would be in IMAP-based systems. The web application must be deployed on a Linux server with root or sudo access, as some operations require elevated privileges (e.g., user creation or password reset). The application assumes that Sendmail is configured correctly to prevent unauthorized relaying and that firewall settings allow access to necessary ports (e.g., 80, 110, 25).

2.1.1.11 Assumption and Dependencies

The project is built on several assumptions and depends on key external components. It is assumed that the operating environment is a stable, modern Linux distribution (such as Ubuntu Server or CentOS) where all required services—Tomcat, Sendmail, and POP3—are pre-installed and properly configured.

Chapter 3
System Design Specification

3.1 System Architecture

The system is based on a three-tier architecture comprising the presentation layer, application logic layer, and data/communication layer. The presentation layer consists of a web-based user interface developed using JSP, HTML, and JavaScript. This layer interacts with the middle layer, which includes Java Servlets and backend logic deployed on the Apache Tomcat server. The backend logic processes user requests, manages sessions, handles authentication, and communicates with the Linux OS. The data/communication layer includes the Linux Mail Server (Sendmail for SMTP, Dovecot for POP3), user mailboxes, and the optional database system used for logging. This tiered architecture ensures modularity, ease of maintenance, and scalability.

3.2 Module Decomposition Description

The system is decomposed into several key modules:

1. User Authentication Module: Manages login, logout, and session handling.

2. Mail Operations Module: Handles email retrieval (via POP3), sending emails (via Sendmail), and mailbox organization (delete/read).

3. Admin Control Module: Executes system-level operations such as creating user accounts, resetting passwords, and managing mail directories.

4. Logging Module: Optionally logs activities into a database for monitoring and audit.

5. Interface Layer: Displays content, error messages, and forms for user interaction, separated for both users and admins.

Each module is designed to function independently while communicating through defined interfaces to achieve seamless integration.

3.3 High Level Design Diagrams

3.3.1 Use Case Diagram

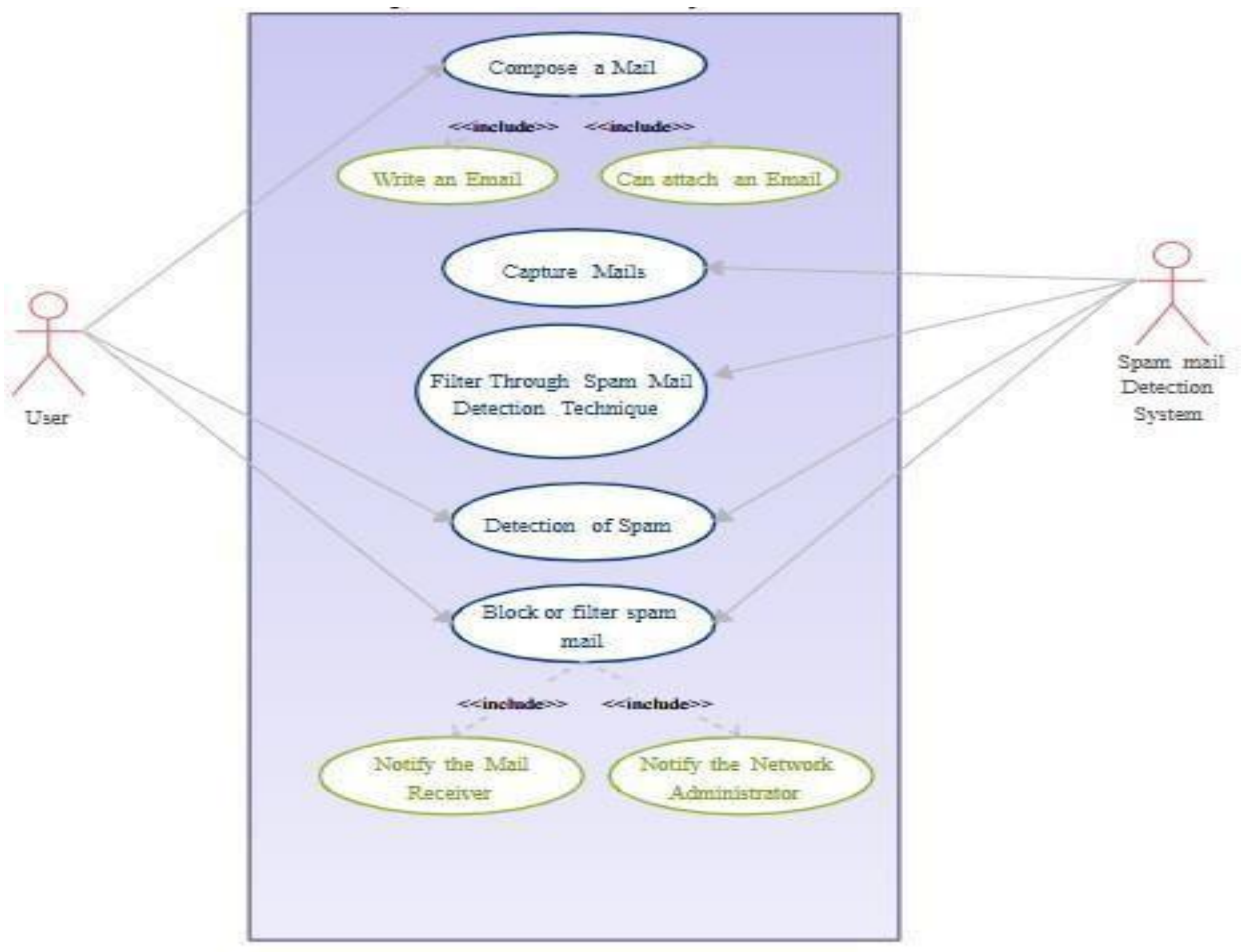

Figure 3.1: USE-CASE DIAGRAM

3.3.2 Activity Diagram

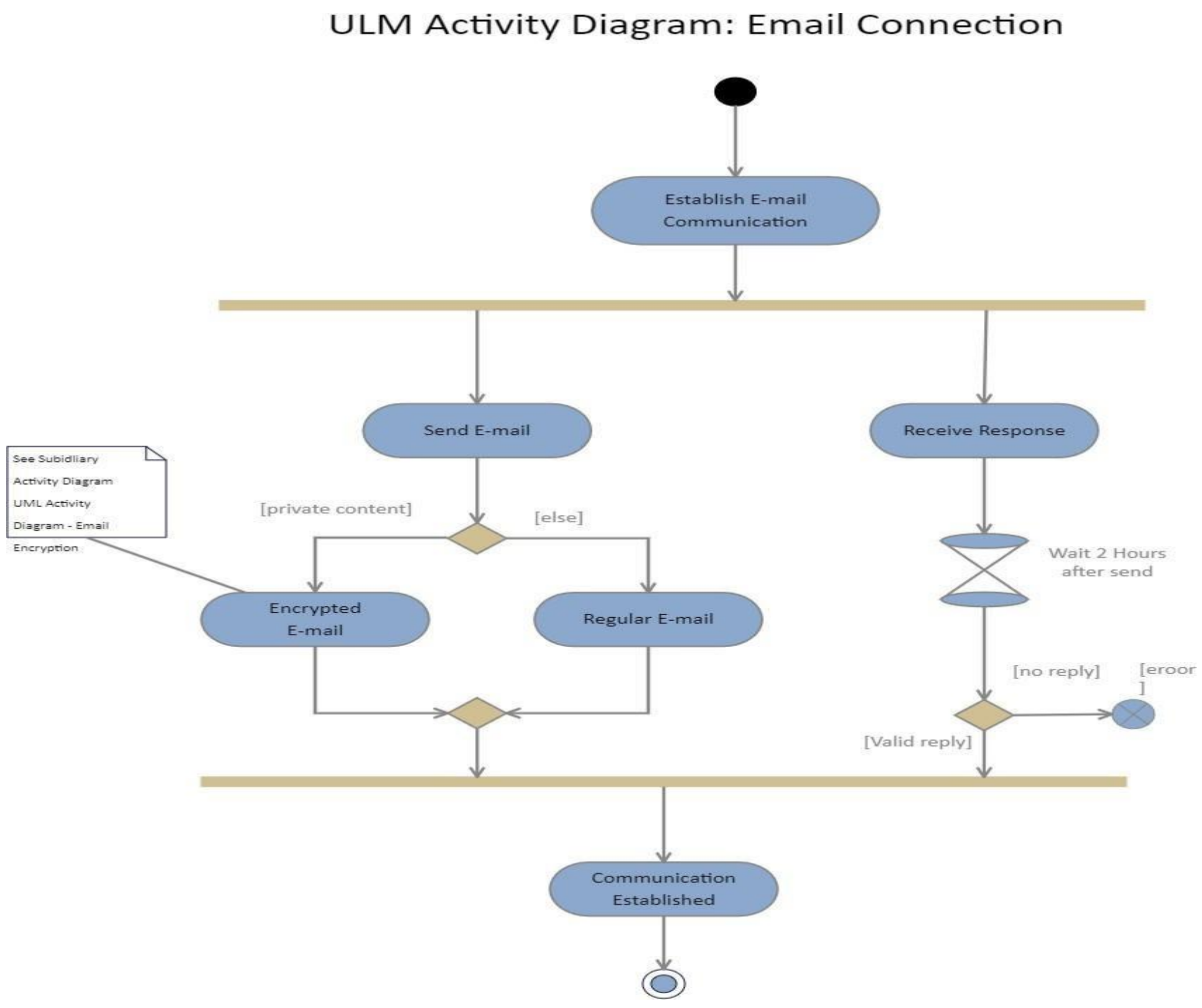
ULM Activity Diagram: Email Connection
Establish E-mail Communication
Send E-mail
Receive Response
See Subidiiary Activity Diagram UML Activity Diagram - Email Encryption
[private content]
[else]
Wait 2 Hours after send
Encrypted E-mail
Regular E-mail
[no reply]
[eroor]
[Valid reply]
Communication Established

Figure 3.2: ACTIVITY DIAGRAM

3.3.3 Data-Flow Diagram

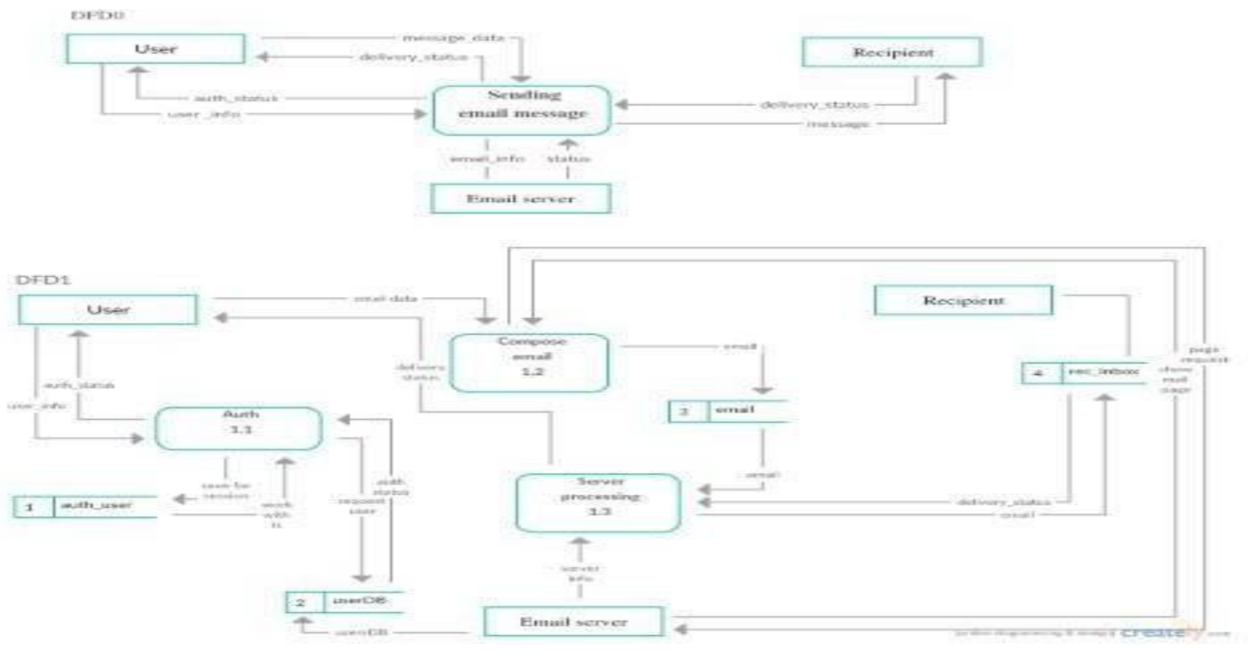

Figure 3.3: DATA FLOW DIAGRAM

3.3.4 Class Diagram

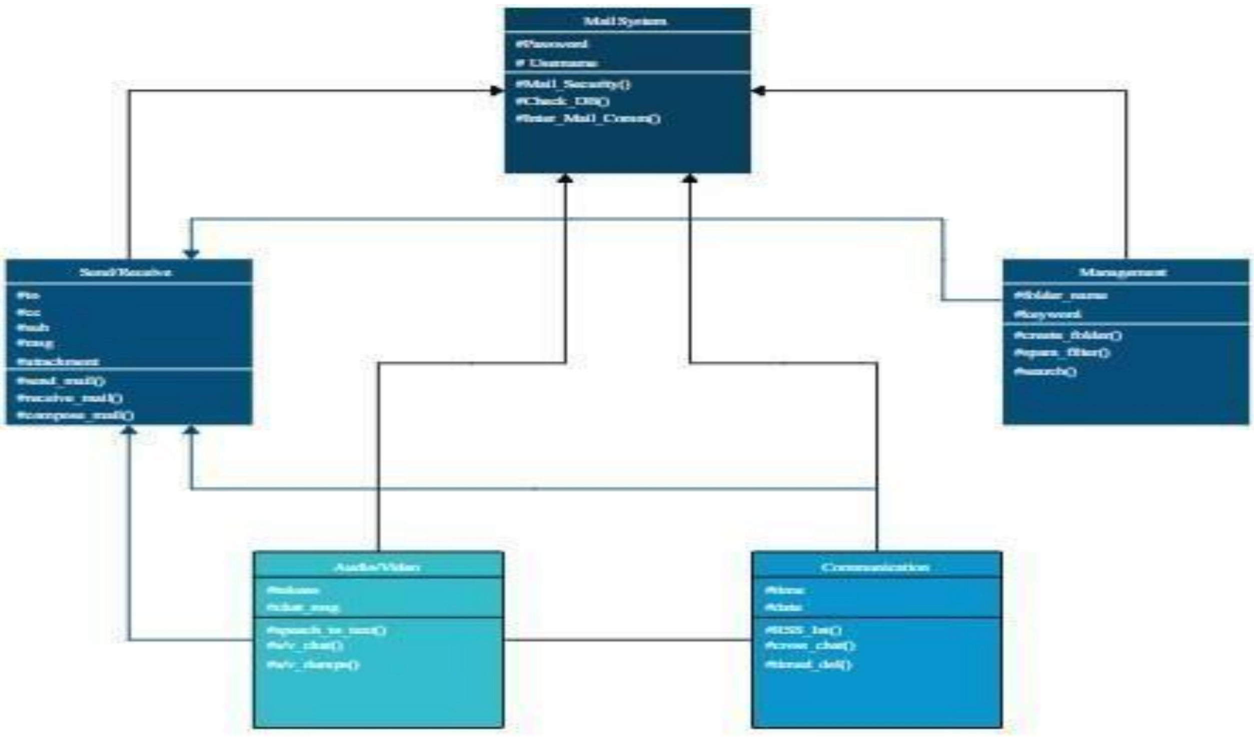

Figure 3.4: CLASS DIAGRAM

Chapter 4
Methodology and Team

4.1 Introduction to Waterfall Framework

The **Waterfall Framework** is a traditional software development methodology that follows a **linear and sequential** approach. It is one of the earliest and most widely used development models, especially suitable for projects with well-defined requirements and a predictable outcome.

In the context of our **Food Delivery System** project, the Waterfall model provides a **structured and systematic approach**, ensuring that each development phase is completed thoroughly before the next one begins. This methodology is particularly beneficial for our project because the core features — such as vendor registration, user authentication, cart management, order tracking, and Razorpay payment integration — are clearly outlined from the beginning.

Using the Waterfall model, we divided the project into distinct phases:

- Requirements gathering
- System design
- Implementation
- Testing

- Deployment
- Maintenance

Each phase builds upon the deliverables of the previous one, ensuring clarity, traceability, and controlled progress. This disciplined approach helped us minimize risks and maintain quality throughout the project lifecycle.

By following the Waterfall Framework, we were able to deliver a **robust, well-tested, and user-friendly food delivery system** with essential features tailored for both vendors and end users.

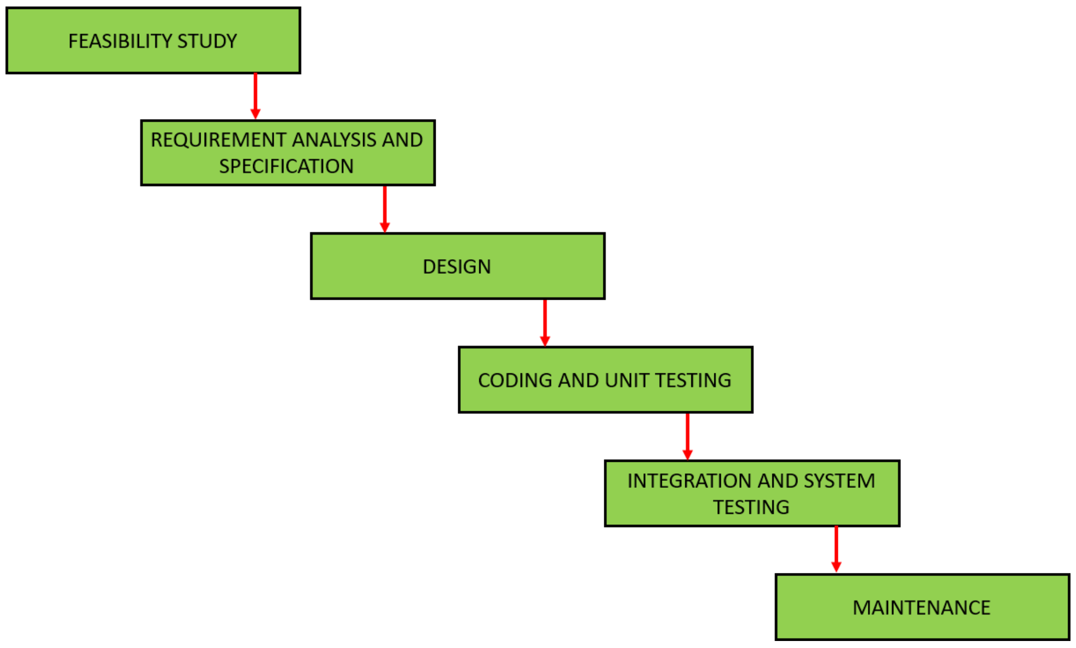
FEASIBILITY STUDY
REQUIREMENT ANALYSIS AND SPECIFICATION
DESIGN
CODING AND UNIT TESTING
INTEGRATION AND SYSTEM TESTING
MAINTENANCE

Figure 4.1: WATERFALL MODEL

Waterfall Model Pros & Cons

Advantage

- **Clear Structure and Phases**

  Each phase (requirements, design, implementation, etc.) is well-defined, making it easier to manage and track progress.

- **Easy to Manage**
  Because of its linear nature, the Waterfall model is simple to understand and follow, especially for small teams or academic projects.
- **Well-Documented Process**
  Each step produces clear documentation (e.g., requirement specs, design docs), which is helpful for future reference or handoffs.
- **Predictability**
  Project timelines and deliverables are easier to estimate when the requirements are stable and fully known from the start.
- **Testing Comes After Development**
  Complete modules are available for testing, so you can do thorough system testing once all pieces are in place.

Disadvantage

- **No Flexibility for Changes**
  Once a phase is completed, it’s difficult and costly to go back and make changes (e.g., updating requirements or redoing design).
- **Late Testing Feedback**
  Errors or mismatches may only be discovered at the testing stage, which can delay delivery or require rework.
- **Not Ideal for Evolving Projects**
  If customer needs change frequently or if new features arise (common in food delivery systems), this model becomes rigid.

- **Risk of Misaligned Expectations**
  Since users see the final product only at the end, there's a chance the solution may not fully meet their expectations without mid-way feedback.

- **Slow Delivery**
  Working software is only available late in the development cycle, which can be a downside in fast-moving industries or startups.

4.2 Team Members, Roles & Responsibilities

- **Project Leader**: Coordinates development activities, schedules meetings, and ensures milestones are met.
- **Backend Developer**: Implements the core mail-handling logic, system integration, and administrative functionalities.
- **Frontend Developer**: Designs user interfaces and ensures responsiveness and usability.
- **System Administrator**: Configures the Linux mail server, handles Sendmail and Dovecot setups, and manages deployment.
- **Tester**: Develops test cases and ensures software quality through functional and performance testing.

Chapter 5
Centering System Testing

The designed Food Delivery System has been tested through the following parameters to ensure correct functionality, user interaction, and data integrity.

5.1 Functionality Testing

In testing the functionality of the web sites the following features were tested:

1. Links
   (a) a) Internal Links: All internal links were tested by navigating through the UI and submitting valid inputs. Navigation between pages worked smoothly.
   (b) External Links: Currently, the system does not include external links. However, in future updates, we plan to integrate third- party restaurant links and real-time delivery tracking via ex- ternal APIs.
   (c) Broken links :During the testing phase, all links were verified. No broken links were found; every button or hyperlink cor-

rectly redirected to its intended page or component

- **(a) Error Messages for Invalid Input:**
  All forms in the system (User Registration, Vendor Signup, Login, Checkout) display proper error messages when invalid or incomplete input is entered.
  - Example: Entering an incorrect password displays ”Invalid credentials.”
  - Empty mandatory fields prompt a validation message (e.g., ”Email is required”).
- **(b) Optional and Mandatory Fields:**
  Mandatory fields are marked with a red asterisk (*). If these are left blank, appropriate error messages are shown. Example: On the vendor registration form, omitting the ”Restaurant Name” field results in an error prompt.

5.1 Database Testing

Database testing was conducted to ensure the **integrity, consistency, and connectivity** of the backend database used in the system:

- Verified that data entered through the frontend (e.g., orders, cart items, user registrations) was correctly stored in the PostgreSQL/MySQL database.

- Payment status, user sessions, and cart data were correctly updated in the database on order confirmation.
- All relationships between tables (e.g., users   orders, ven-dors products) were tested for referential integrity.

5.2 Performance Testing

Performance testing was undertaken to evaluate how the system handles various levels of load, processing speed, and resource utilization. The primary focus was on measuring response time during email operations and ensuring system stability when multiple users access the system concurrently. Several test scripts were run to simulate high-traffic situations, such as multiple users sending or receiving emails at the same time. The average response time for loading the inbox, sending an email, and logging in was recorded and found to be within acceptable limits under normal network conditions.These results confirm that the application can serve as a reliable internal communication tool without degradation in user experience over time or under stress. System performs well under normal loads and can be improved further for high-traffic scenarios.

5.3 Usability Testing

Usability testing focuses on the user's interaction with the system and assesses whether the interface is intuitive, efficient, and easy to navigate. Test users were asked to perform common tasks such as logging in, composing emails, and managing received messages without prior training or guidance. Feedback was collected on the clarity of labels, layout of the interface, responsiveness of UI components, and overall experience. The system's web interface, built using HTML, JSP, and basic JavaScript, was found to be simple and functional. Testers appreciated the minimalistic design and the clear separation of functions for users and admins. Improvements were suggested regarding visual feedback for actions like "email sent" or "invalid login," which were then implemented. Additionally, accessibility considerations such as color contrast, font size, and compatibility with keyboard navigation were reviewed. Overall, the system was found to be user-friendly for its intended audience, especially within a closed or semi-technical organizational environment. These findings support the system's readiness for deployment among staff with varying levels of technical expertise.

Chapter 6

Test Execution Summary

The testing phase of the web-based mail client project involved the execution of comprehensive test cases that covered all critical components and user scenarios. Both manual and automated testing techniques were employed to validate the functional and non-functional aspects of the system. Functionality tests confirmed that users were able to log in with valid credentials, view their inbox, read individual messages, compose and send new emails using the Sendmail utility, and delete messages. Administrative functions such as creating new users, resetting passwords, and monitoring account activity were also thoroughly tested and found to be working as intended. Performance tests evaluated the system's responsiveness under normal and simulated high-load conditions; results showed consistent stability and acceptable response times for core operations, even when accessed by multiple users simultaneously.

Chapter 7

Project Screen Shots

1

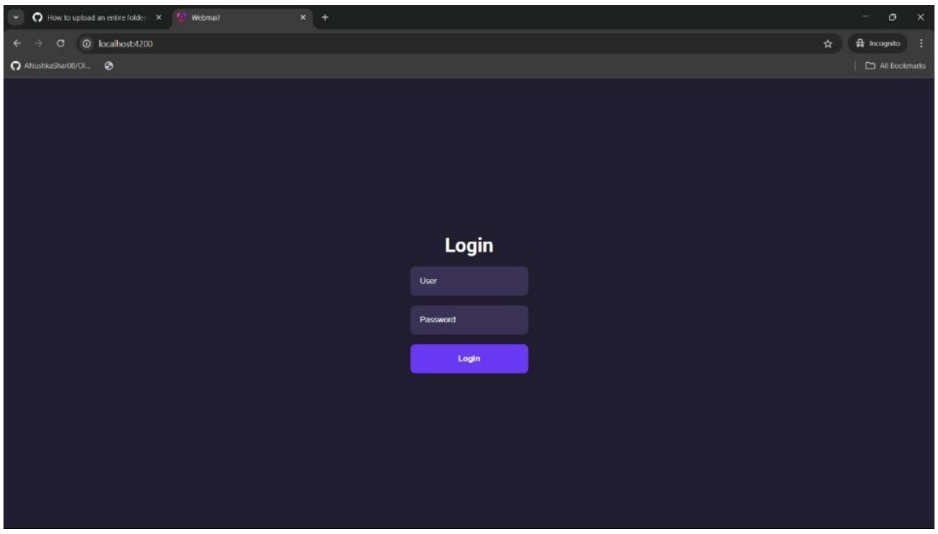

Figure 7.1: WELCOME PAGE-AUTHENTICATION

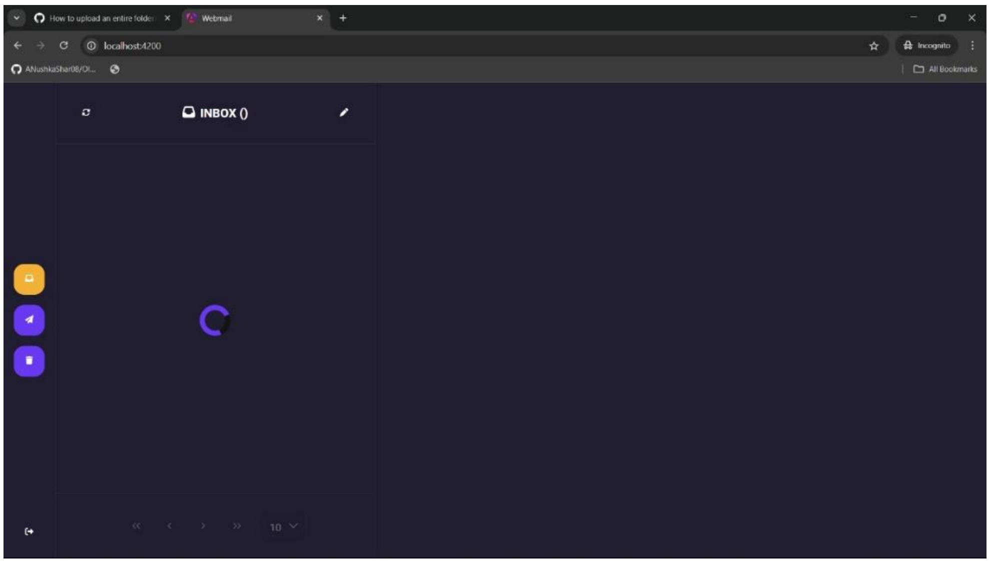

Figure 7.2: INBOX INTERFACE

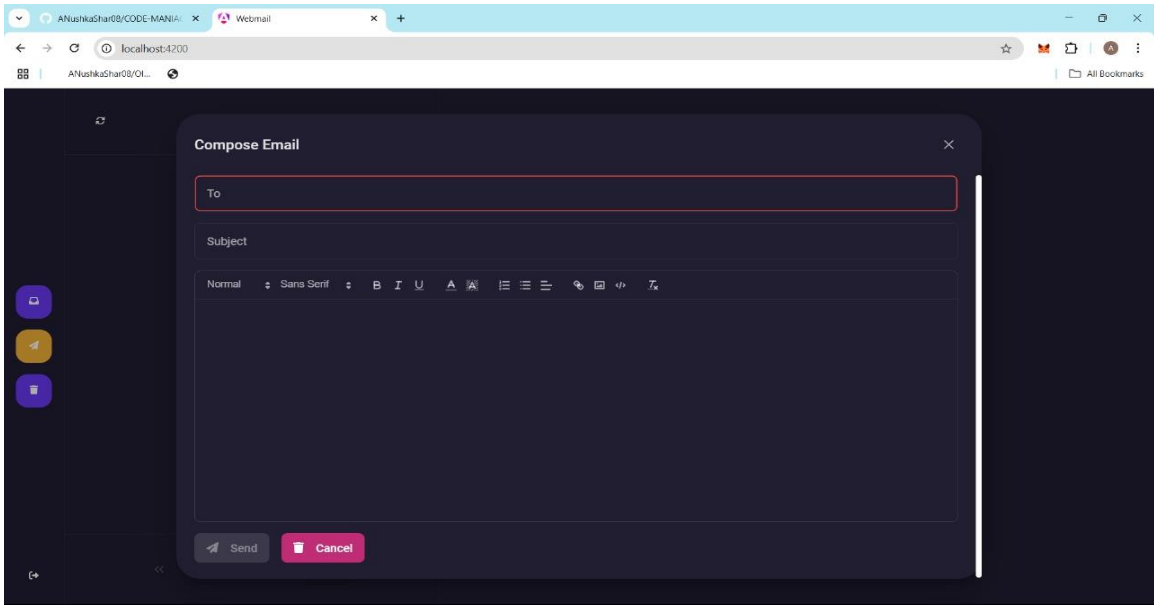

Figure 7.3: COMPOSE MAIL

Chapter 8

Project Summary and Conclusions

8.1 Conclusion

This project demonstrates the development of a lightweight, secure, and accessible mail client using open-source technologies. By integrating Java-based web technologies with Linux system utilities, the project bridges the gap between command-line-based mail systems and user-friendly web access. It supports essential mail functions, secure administration, and extensibility through modular design.

The web-based mail client successfully meets its objectives by enabling users to interact with a Linux mail server without requiring terminal access. The use of standard protocols and services ensures compatibility and stability. The project proves that simple, open-source tools can be orchestrated to build robust communication solutions for localized or organizational use.

Chapter 8

Future Scope

While the current implementation of the web-based mail client provides essential functionalities such as sending and receiving emails, managing users, and performing administrative tasks, there is substantial potential for enhancement and expansion in future iterations. One of the foremost improvements would be the integration of **IMAP (Internet Message Access Protocol)** alongside POP3. While POP3 is sufficient for basic email retrieval, IMAP allows for real-time email synchronization across multiple devices, maintaining consistency in email status (read/unread, flagged, etc.) across platforms, which is highly beneficial for modern users who operate from multiple endpoints.

Another major area of future development is **enhanced security mechanisms**, including the implementation of **Two-Factor Authentication (2FA)** for login processes. As cyber threats continue to evolve, adding a second layer of verification—such as a time-based one-time password (TOTP) sent to the user's mobile device—would significantly strengthen account protection. Additionally, incorporating **end-to-end encryption** for email content using standards like **PGP (Pretty Good Privacy)** or **GPG (GNU Privacy Guard)** would protect sensitive communication and reinforce user trust, especially in enterprise environments.

From a user experience perspective, the interface could be modernized and made **mobile-responsive** using front-end frameworks like Bootstrap or Tailwind CSS. This would ensure seamless operation on smartphones and tablets, catering to the increasing mobile user base.

References

[1] Oracle. Java Platform, Enterprise Edition (Java EE) 7. https://docs.oracle.com/javaee/7/index.html

[2] The Internet Engineering Task Force (IETF). RFC 1939 - Post Office Protocol - Version 3. https://tools.ietf.org/html/rfc1939

[3] TutorialsPoint. Linux - Mail Server Setup. https://www.tutorialspoint.com/linux/linux_mail_server.html

[4] Dovecot.org. Dovecot Wiki and Configuration Guides. https://doc.dovecot.orghttps://developer.mozilla.org

[5] The Linux Documentation Project. https://tldp.org