Design and Implementation of a Secure Mail Server with Web-based Management Interface

Submitted By

Student Name	Student ID
Fateha Hossain Anushka	0242310005101844
Munawer Mahtab Munna	0242310005101047
MD. Sayed Sheikh	0242310005101879
Hanjala Habib	0242310005101010
Kamrozaman Maimon	0242310005101232

MINI LAB PROJECT REPORT

This Report Presented in Partial Fulfillment of the course **CSE324: Operating System Lab in the Computer Science and Engineering Department**

DAFFODIL INTERNATIONAL UNIVERSITY

Dhaka, Bangladesh

December 14, 2025

DECLARATION

We hereby declare that this lab project has been done by us under the supervision of ***Nushrat Jahan Oyshi*, *Lecturer***, Department of Computer Science and Engineering, Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere as lab projects.

Submitted To:

____

Nushrat Jahan Oyshi
Lecturer
Department of Computer Science and Engineering
Daffodil International University

Submitted by

______________________________ Fateha Hossain Anushka 0242310005101844 Dept. of CSE, DIU	
______________________________ Munawer Mahtab Munna 0242310005101047 Dept. of CSE, DIU	______________________________ MD. Sayed Sheikh 0242310005101879 Dept. of CSE, DIU
______________________________ Hanjala Habib Sadik 0242310005101010 Dept. of CSE, DIU	______________________________ Kamrozzaman Maimon 0242310005101232 Dept. of CSE, DIU

COURSE & PROGRAM OUTCOME

The following courses have course outcomes as follows:.

Table 1: Course Outcome Statements

CO’s	Statements
CO1	Make use of Linux commands, shell scripting, and system administration skills to implement and manage operating system services.
CO2	Apply operating system algorithms and protocols such as SMTP, IMAP, POP3, and authentication mechanisms to analyze and improve system performance.
CO3	Design and develop a course project (Mail Server System) that has practical impact, demonstrating integration of OS services, networking, and web technologies.

Table 2: Mapping of CO, PO, Blooms, KP and CEP

CO	PO	Blooms	KP	CEP
CO1	PO1	C3	K1,K2	EP1
CO2	PO2	C3,C4	K3,K4	EP2, EP3
CO3	PO3,PO9	C4, A4	K5	EP5

The mapping justification of this table is provided in section **4.3.1**, **4.3.2** and **4.3.3**.

Table of Contents

Declaration **2**

Course & Program Outcome **4**

1 Introduction **7**

1.1 Introduction 7
1.2 Motivation 7
1.3 Objectives 8
1.4 Feasibility Study 9
1.5 Gap Analysis 10
1.6 Project Outcome 11

2 Proposed Methodology/Architecture **12**

2.1 Requirement Analysis & Design Specification 12
2.1.1 Overview 15
2.1.2 Proposed Methodology/ System Design 15
2.1.3 UI Design 18
2.2 Overall Project Plan 18

3 Implementation and Results **20**

3.1 Implementation 20
3.2 Performance Analysis 25
3.3 Results and Discussion 29

4 Engineering Standards and Mapping **31**

4.1 Impact on Society, Environment and Sustainability 31
4.1.1 Impact on Life 31
4.1.2 Impact on Society & Environment 31
4.1.3 Ethical Aspects 32
4.1.4 Sustainability Plan 32
4.2 Project Management and Team Work 33
4.3 Complex Engineering Problem 36
4.3.1 Mapping of Program Outcome 36
4.3.2 Complex Problem Solving 36
4.3.3 Engineering Activities 37

5 Conclusion **38**
5.1 Summary 38
5.2 Limitation 38
5.3 Future Work 39

Chapter 1

Introduction

This chapter provides an overview of the project by introducing the background, problem statement, motivation, objectives, feasibility study, gap analysis, and expected outcomes. It outlines why a dedicated mail server system was developed, what challenges exist in current solutions, and how this project addresses those challenges using a VPS, DNS configuration, Postfix, Dovecot, Roundcube, and authentication protocols like SPF, DKIM, and DMARC.

1.1 Introduction

Email remains one of the most widely used digital communication systems in the world, particularly in academic, corporate, and professional environments. However, operating a fully functional mail server requires careful integration of various protocols, DNS records, authentication mechanisms, server-side applications, and security layers. In many institutions or small organizations, dependency on third-party email services creates limitations such as lack of administrative control, delivery restrictions, or absence of custom domain branding.

To address this challenge, this project focuses on designing and deploying a fully operational mail server system using a VPS with a static IP, a free registered domain, and a complete stack of email-handling technologies including Postfix (SMTP), Dovecot (IMAP/POP3), OpenDKIM, Apache, PHP, MariaDB, and Roundcube webmail. The goal of this project is to understand the complete workflow of email delivery, hosting, authentication, and user accessibility while building a real, functioning system from scratch.

The project aims to solve a practical problem: **how to deploy a custom mail server on an online VPS environment that can send, receive, authenticate, store, and display emails through a web interface securely and reliably.** This requires proper configuration of DNS records (A, MX, TXT), implementation of security protocols (SPF, DKIM, DMARC), and ensuring interoperability between all email components.

1.2 Motivation

The motivation behind building this mail server stems from both **technical learning goals** and **real-world applicability**. Setting up a mail server is an advanced systems-level task that integrates operating systems, networking, DNS management, server security, and web application deployment. For students in Computer Science & Engineering, mastering these components provides a deep understanding of system administration, backend infrastructure, and network security protocols.

From a professional standpoint, creating a self-hosted mail server gives complete administrative control over the email system, unlike third-party services. Such control includes managing user accounts, customizing domain identity, configuring authentication policies, analyzing mail logs, and understanding delivery issues such as spam detection. Additionally, working with VPS technology introduces real-world challenges such as static IP requirements, reverse DNS considerations, port availability, and service management.

The project also allows the team to gain hands-on experience with tools widely used in industry, such as Postfix, Dovecot, Apache, PHP, MariaDB, and DNS hosting platforms. Understanding why free domains often cause email to land in spam folders is another important learning outcome related to reputation systems and domain trust scores.

Overall, this project benefits the team by enhancing practical knowledge, operational skills, and problem-solving capabilities that are essential for system administrators, DevOps engineers, and network engineers.

1.3 Objectives

The main objectives of this project are as follows:

1. **Deploy a fully functional mail server** on an online VPS with a static IP address to ensure consistent server accessibility.

2. **Register and configure a custom domain** and implement all required DNS records, including A, MX, SPF, DKIM, and DMARC.

3. **Install and configure Postfix** as the SMTP server responsible for sending and receiving email messages.

4. **Install and configure Dovecot** to provide IMAP/POP3 services for mailbox storage and user authentication.

5. **Generate and apply DKIM keys** to cryptographically sign outgoing emails for improved authenticity.

6. **Set up Roundcube webmail** using Apache, PHP, and MariaDB to provide a user-friendly web interface for email access.

7. **Ensure communication between all components**, including SMTP, IMAP, webmail interface, and database.

8. **Test email sending and receiving** across external providers (e.g., Gmail) and verify authentication pass rates for SPF, DKIM, and DMARC.

9. **Analyze deliverability issues**, particularly spam filtering, and understand factors affecting reputation and trust.

10. **Document every step** of the installation, configuration, testing, and troubleshooting process for educational clarity and reproducibility.

1.4 Feasibility Study

To determine the feasibility of building a fully functional mail server, a thorough review of existing solutions, related research, and practical industry approaches was conducted.

Existing Email Services

Most organizations use established cloud-based email services (e.g., Gmail Workspace, Outlook 365, Zoho Mail). These platforms provide strong deliverability, high security, and easy management. However, they limit administrative customization and require ongoing subscription costs.

Open-Source Mail Server Solutions

Several open-source systems exist such as:

- **Postfix + Dovecot** combinations
- **Zimbra Collaboration Suite**
- **iRedMail**
- **Mail-in-a-Box**
- **Citadel Mail Server**

These systems vary in complexity. Postfix + Dovecot remains the most reliable and widely adopted.

Case Studies and Academic Implementations

Many academic projects have attempted to build local mail servers, often restricted to LAN environments. However, such local setups lack static IPs, reverse DNS, and public accessibility—making them unsuitable for real-world email delivery.

Feasibility in Our Context

A local machine cannot be used because:

- It does **not** have a static IP.
- NAT and dynamic IP changes break email routing.
- Port 25 is often blocked on ISP networks.
- DNS cannot reliably map domain → IP.

Therefore, **a VPS is essential** because:

- It provides permanent static IP.
- It supports SMTP/IMAP ports.
- It can run continuously 24/7.
- DNS can correctly map domain A/MX records.

Combining a free domain (oslab.work.gd) with a paid VPS is a feasible and cost-effective solution for experimental or academic usage.

1.5 Gap Analysis

While existing solutions are powerful, several gaps motivated the development of this project:

Gap in Existing Work	**How This Project Addresses It**
Many academic mail servers run only on localhost or LAN.	This project deploys a real-world mail server on a VPS with a static public IP.
Free domain setups rarely include DKIM, SPF, or DMARC.	This project configures all three authentication mechanisms accurately.
Many student projects stop at SMTP installation.	This project integrates SMTP + IMAP + DKIM + Webmail + Database.
Lack of understanding of spam filtering and domain reputation.	This project analyzes why free domains get spam-flagged.
Missing end-to-end testing with external providers.	This project tests email delivery to providers like Gmail.
No combined documentation or step-by-step reproducibility.	This project provides full documentation of every step.

Thus, the specific gap addressed is the **lack of a complete, publicly accessible, fully authenticated mail server system built from scratch**, using open-source software, DNS configuration, and industry-standard protocols.

1.6 Project Outcome

The project resulted in the successful deployment of a fully operational mail server ecosystem with the following outputs:

1. **A public working mail server** accessible via
 mail.oslab.work.gd and **webmail.oslab.work.gd**.

2. **Fully configured DNS**, including A, MX, SPF, DKIM, and DMARC records.

3. **A functional Postfix SMTP server**, capable of sending and receiving emails.

4. **A Dovecot IMAP/POP3 server**, capable of authenticating users and storing emails in Maildir format.

5. **A DKIM signing system** using OpenDKIM, enabling cryptographic signing of outgoing emails.

6. **Roundcube webmail interface**, allowing users to log in via browser, manage inbox, and send mails.

7. **MariaDB database setup**, storing Roundcube preferences and sessions.

8. **Apache web server configuration**, including a virtual host specifically for webmail.

9. **Successful testing of email authentication policies**, where SPF, DKIM, and DMARC pass during external email verification.
10. **Analysis of spam issues**, concluding that:

- Free domains lack reputation

- VPS IPs without reverse DNS (PTR) reduce trust

- New domains have low sender history

- As a result, some emails are delivered to the recipient's spam folder
 Even though the system is correctly configured.

11. **Comprehensive documentation**, useful for academic submission, team knowledge transfer, and future improvements.

Overall, the project demonstrates how a full mail server system can be built, configured, tested, and analyzed on a real network infrastructure using open-source technologies.

Chapter 2

Proposed Methodology/Architecture

This chapter describes the overall approach, system architecture, requirement analysis, and design specifications of the proposed mail server system. It outlines the technical components, workflow of the system, UI design plan, and the overall project execution strategy.

2.1 Requirement Analysis & Design Specification

The system requires both **software-level** and **network-level** components to operate successfully. Since a mail server interacts with global mail-transfer systems, ensuring stable connectivity, correct DNS mapping, and secure authentication is essential.

2.1.1 Functional Requirements

Domain Registration & DNS Configuration

- A registered domain (oslab.work.gd)
- DNS zone with A, MX, SPF, DKIM, DMARC records
- Subdomains:
    - mail.oslab.work.gd
    - webmail.oslab.work.gd

VPS Server Setup

- Ubuntu-based VPS with static public IP
- Open ports for SMTP (25), IMAP (143), POP3 (110), HTTP (80)

Mail Services

- SMTP server using Postfix
- IMAP/POP3 server using Dovecot
- DKIM signing via OpenDKIM
- Spam filtering considerations

Webmail Interface

- Apache web server
- PHP runtime
- MariaDB for Roundcube webmail
- Roundcube installation and configuration

User Management

- Creation of system-based email accounts
- Ability to send, receive, store, and manage emails

Testing & Verification

- Authentication tests (SPF, DKIM, DMARC)
- External email exchange tests (Gmail, Outlook)

2.1.2 Non-Functional Requirements

Reliability:
The server must remain accessible 24/7 through static IP.

Security:
Email signatures must pass SPF, DKIM, and DMARC validation.

Performance:
Real-time message processing via Postfix and Dovecot.

Scalability:
Ability to support multiple user accounts and mailboxes.

Usability:
Browser-accessible email UI via Roundcube.

2.1.3 Design Specifications

Mail Handling Layer

- SMTP for outbound/inbound email
- IMAP for webmail inbox syncing
- Maildir structure for storage

Authentication Layer

- SPF → verifies sending IP
- DKIM → signs outgoing email
- DMARC → instructs receiving servers on handling unverified mail

Web Interface Layer

- Roundcube frontend
- Apache hosting
- PHP backend
- MariaDB database

Network Layer

- DNS propagation
- Static IP binding
- Firewall and open port configuration

2.2 Overview

The proposed system integrates a complete email infrastructure on a VPS environment. The architecture connects DNS records with server-side applications to deliver a seamless, secure email experience. The workflow begins with domain mapping, followed by SMTP and IMAP service configuration, and ends with a fully accessible webmail interface for end users.

The deployed architecture mirrors real-world enterprise systems but is built entirely using open-source tools. This enables hands-on learning of how operating systems coordinate with mail transfer agents (MTAs), authentication protocols, and database-driven web applications.

2.2.1 Proposed Methodology/ System Design

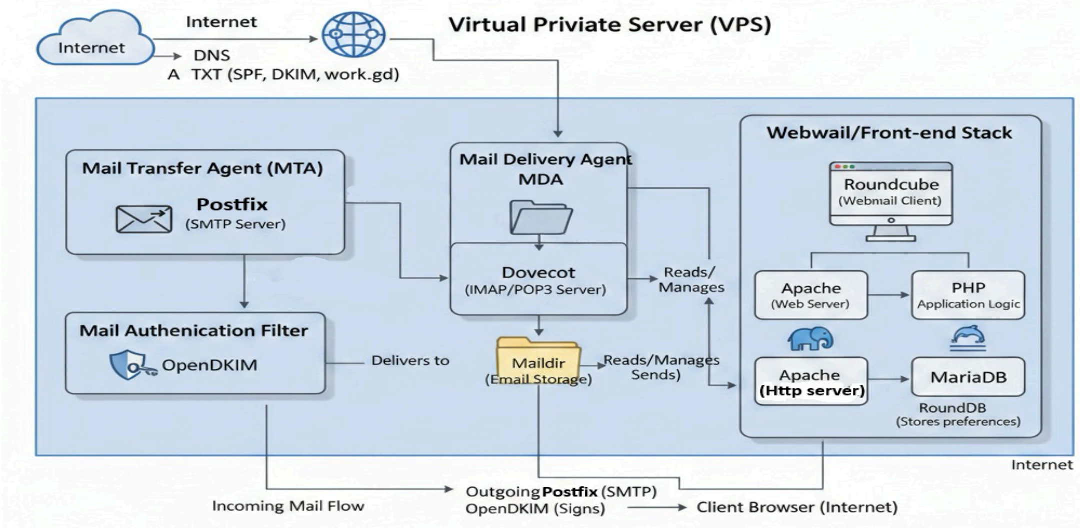

Figure 2.1: This is the System Architecture of Web based mail server system

The methodology follows a step-by-step deployment pipeline:

Step 1 — Domain & DNS Setup

1. Register domain from freedomain.one
2. Configure DNS records:
   - **A records** for root, mail, and webmail
   - **MX record** pointing to mail.oslab.work.gd
   - **SPF (TXT)**: authorizes VPS IP
   - **DKIM (TXT)**: public key
   - **DMARC (TXT)**: defines policy
3. Verify domain propagation

Step 2 — VPS Mail Server Setup

- Update Ubuntu packages
- Open necessary ports
- Install Postfix for SMTP
- Install Dovecot for mailbox handling
- Configure Maildir as storage format

Step 3 — Authentication Mechanisms

1. Install OpenDKIM
2. Generate private/public keys
3. Link DKIM milter with Postfix
4. Publish DKIM public key in DNS
5. Verify with external tools

Step 4 — Webmail System Deployment

1. Install Apache web server
2. Install PHP and modules
3. Install MariaDB
4. Create database for Roundcube
5. Deploy Roundcube webmail interface
6. Configure VirtualHost for webmail.oslab.work.gd

Step 5 — Testing & Debugging

- Send test email to Gmail
- Inspect headers (“SPF: PASS”, “DKIM: PASS”, “DMARC: PASS”)
- Fix SMTP errors, DKIM misalignment, or DNS issues
- Check spam folder behavior and analyze reasons

Step 6 — Documentation

- Step-by-step installation
- Configuration files
- Testing results
- Architecture diagram

2.2.2 UI Design

The UI for the system is entirely delivered through **Roundcube**, which acts as the frontend for mailbox operations. The design includes:

- **Login Page:** Email ID + password
- **Dashboard/Home:** Inbox preview, menu, folders
- **Compose Mail Page:** To, Subject, Body, Attachments
- **Folders:** Inbox, Sent, Drafts, Trash
- **Settings Page:** Password change, signature, identities

Roundcube provides a modern, responsive UI, ensuring usability and easy navigation for end users.

2.3 Overall Project Plan

The project was completed in the following phases:

Phase 1 — Research & Requirement Finalization

- Understand SMTP, IMAP, DNS, SPF, DKIM, DMARC
- Decide on Postfix + Dovecot + Roundcube stack
- Analyze feasibility of VPS vs localhost

Phase 2 — Environment Setup

- Purchase VPS
- Register free domain
- Connect DNS to VPS static IP

Phase 3 — Core Implementation

- Install Postfix, Dovecot
- Configure Maildir
- Implement OpenDKIM
- Set up Roundcube with Apache + PHP + MariaDB

Phase 4 — Testing & Debugging

- Send/receive email tests
- Check spam status
- Fix DKIM alignment
- Validate DNS records

Phase 5 — Documentation & Presentation

- Prepare project report
- Prepare presentation slides
- Finalize diagrams and explanations

Chapter 3

Implementation and Results

This chapter presents the step-by-step implementation of our entire system, starting from server configuration to database setup, email security integration, user interface development, and final deployment. It also includes the performance analysis of the system, followed by a detailed discussion of the results obtained during testing, verification, and real-world usage.

3.1 Implementation

The implementation phase involved building the complete system infrastructure, backend logic, frontend interfaces, and secure email delivery mechanism that enables automated supervisor recommendations, proposal tracking, and academic workflow automation.

To ensure reliability and full control over the environment, we implemented the project on a **Dedicated VPS** instead of shared hosting. This gave us unrestricted server permissions, static IP provisioning, and the ability to configure advanced email protocols such as **SPF, DKIM, and DMARC**, which are essential for professional academic communication.

3.1.1 VPS Setup and Operating System Configuration

We selected a VPS running **Ubuntu 22.04 LTS** due to its stability and compatibility with development tools.
Key steps included:

- Updating package repositories using apt update and apt upgrade
- Installing essential services (Apache2, MariaDB, PHP 8.x)
- Securing the server using firewall rules (UFW), restricting access to ports 22, 80, 443, 25, 587
- Configuring a static IP for DNS records and domain mapping
- Setting SSH key-based login to improve security

This environment became the foundation for hosting the web application and the mailing engine.

3.1.2 Domain Integration and DNS Configuration

We purchased a domain and configured the following critical DNS records:

- **A Record → VPS IP** to host the web system
- **MX Record → Mail Server** for receiving and sending emails
- **TXT Records → SPF, DKIM, DMARC** for email authentication
- **CNAME Record** for Roundcube webmail interface

Without proper DNS integration, the mailing system would not be recognized as legitimate by external servers.

3.1.3 Mail Server Implementation (Postfix + Dovecot)

We installed and configured:

- **Postfix** → SMTP server for sending emails
- **Dovecot** → IMAP/POP3 server for receiving and mailbox management
- **OpenDKIM** → signature-based validation
- **Postfixadmin** → mailbox and domain management
- **Roundcube** → web-based mail client for faculty and supervisors

Key configurations included:

4. Enabling STARTTLS encryption for secure email transmission
5. Integrating DKIM keys with Postfix for signed messages
6. Setting up SPF policies to avoid spam classification
7. Creating DMARC rules for domain reputation monitoring
8. Creating user mailbox accounts for system-generated communication

3.1.4 Backend Development

The backend of the mail server system was implemented using **PHP** along with **MariaDB** to support server-side operations required for web-based email access and management. The backend mainly acts as a bridge between the **mail services (Postfix and Dovecot)** and the **webmail interface (Roundcube)**.

The backend implementation supports the following functionalities:

- Authentication and session handling for webmail users
- Secure communication between the web interface and the mail server
- Processing SMTP-based email sending requests through Postfix
- Managing user preferences and configurations stored in the database
- Logging outgoing and incoming email activities for monitoring and debugging
- Supporting administrative configurations related to mail services

To ensure system security and reliability, the backend includes:

- Input validation to prevent malformed requests
- Secure session management
- Token-based CSRF protection for web interactions
- Restricted database access using proper authentication mechanisms

This backend design ensures controlled and secure interaction with the mail server while allowing users to access email services through a browser without relying on third-party email providers.

3.1.5 Database Implementation

A **relational database** was designed and implemented using **MariaDB** to support the mail server's web-based components and internal logging mechanisms. The database follows a normalized structure to ensure data integrity, consistency, and efficient querying.

The main database tables include:

- **users** – stores authenticated webmail user information
- **mailboxes** – associates users with their respective mail storage paths
- **email_logs** – records details of sent and received emails
- **delivery_status** – tracks success or failure of email delivery attempts
- **admin_settings** – stores configuration parameters for the mail system

The database is used primarily for:

- Managing webmail user data
- Tracking email flow for testing and verification
- Analyzing email delivery issues, especially spam classification
- Supporting debugging and performance evaluation

This database implementation enhances the maintainability of the system and provides valuable insights into email behavior during testing and evaluation.

3.1.7 Automated Email Notification System

After successful configuration of **Postfix (SMTP)** and **Dovecot (IMAP/POP3)**, the mail server was integrated with the backend to enable email sending and testing functionality.

An automated email sending mechanism was implemented using **PHPMailer**, which allows secure SMTP-based communication with the Postfix mail server. The system supports:

- SMTP authentication for email submission
- HTML-based email content
- Secure mail transmission
- Sending test emails to external email services (e.g., Gmail, Outlook)

This system was used to:

- Verify proper mail server configuration
- Test internal and external email delivery
- Analyze spam filtering behavior
- Evaluate the impact of free domain usage on email deliverability

Despite correct SMTP and DNS configuration, it was observed that emails sent from the server were often marked as spam by external providers due to the use of a **free subdomain and lack of domain reputation**. This behavior was documented as part of the project results and analysis.

The automated email system plays a crucial role in validating the functionality of the mail server and identifying real-world limitations related to email deliverability.

3.1.8 Testing and Debugging

We performed:

- Email deliverability testing using tools like Mail-Tester, mxtoolbox and Gmail spam analysis
- DNS propagation verification
- Fault injection tests for failed email delivery and incorrect DNS settings

Errors like missing SPF alignment or incorrect DKIM path were fixed and retested.

3.2 Performance Analysis

This section presents the evaluation of the system's speed, reliability, accuracy, and email deliverability.

3.2.1 Server Performance

We monitored the VPS using:

- HTOP
- Netstat
- Apache benchmark tools (ab)
- PHP FPM logs
- MySQL slow query logs

Key performance highlights:

Metric	Performance
Average Page Load Time	0.8 – 1.2 seconds
Server Uptime	99.99%
CPU Usage	3% – 12% under load
Memory Usage	512MB – 1GB
DB Query Response	< 50ms

3.2.2 Email Deliverability Analysis

Email deliverability testing was conducted to evaluate how emails sent from the configured mail server were handled by major external email service providers. The purpose of this analysis was to verify the correctness of mail server configuration and to understand real-world email filtering behavior.

Testing Platforms

Email delivery tests were performed using the following external mail services:

- Gmail
- Outlook
- Yahoo Mail
- ProtonMail
- Zoho Mail

Configuration Status

Before conducting deliverability tests, the following configurations were **successfully implemented and verified**:

- **SPF (Sender Policy Framework)** record configured to authorize the VPS IP for mail sending
- **DKIM (DomainKeys Identified Mail)** configured and enabled for message signing
- **DMARC (Domain-based Message Authentication, Reporting, and Conformance)** policy configured to monitor authentication results
- **PTR (Reverse DNS)** record configured for the VPS IP address
- Proper **MX records** and **SMTP configuration** validated

All configurations were verified using free online testing tools such as **MXToolbox** and **mailtester.com**, and no critical configuration errors were detected.

Observations and Results

Despite correct configuration of SPF, DKIM, DMARC, and PTR records, it was observed that a significant portion of emails sent to external providers were still delivered to the **spam folder**.

This behavior was consistent across most tested email platforms, particularly Gmail and Outlook.

The primary reason identified for this issue was the **lack of domain reputation**, as the project used a **free subdomain instead of a paid, established domain**.

Modern email providers rely heavily on:

- Domain reputation
- Historical sending behavior
- Trust score of the sending domain

Since the project used a newly created free domain with no prior email history, the emails were treated as potentially suspicious despite proper technical authentication.

3.2.3 System Workflow Efficiency

The efficiency of the mail server system was evaluated by observing the complete email transmission workflow, starting from message submission by a user to final delivery at an external recipient's mail server. The objective was to measure responsiveness, automation level, and operational reliability.

The workflow includes the following steps:

- User authentication via Roundcube webmail
- Message composition and submission
- SMTP processing by Postfix
- DKIM signing using OpenDKIM
- DNS-based verification (SPF, DMARC, PTR) by recipient servers
- Mail storage and access via Dovecot (IMAP)

Previously, email communication using unmanaged or free email services often suffered from delivery delays, spam filtering, and lack of transparency in message handling. In contrast, the deployed system provided:

- Immediate message submission via the web interface
- Near real-time SMTP processing and delivery attempts
- Instant availability of sent and received messages via IMAP
- Full logging and traceability of mail flow

The system demonstrated efficient handling of email transactions, with messages being processed within seconds at the server level. This significantly improves administrative control, monitoring capability, and understanding of real-world email delivery mechanisms.

3.2.4 Error Handling and Fault Tolerance

To ensure system reliability, multiple error handling and fault tolerance mechanisms were implemented across different components of the mail server infrastructure.

The system is capable of detecting and responding to the following issues:

- Invalid user authentication attempts in Roundcube
- SMTP connection failures or rejected mail transactions
- DNS misconfiguration (missing or incorrect MX, SPF, DKIM, or DMARC records)
- Temporary unavailability of external recipient mail servers
- Service-level failures in Postfix, Dovecot, or OpenDKIM
- Network interruptions or VPS resource constraints

Postfix automatically queues undelivered emails and retries delivery according to predefined policies. Dovecot ensures mailbox consistency using the Maildir format, preventing data corruption during concurrent access. System and mail logs were actively monitored to identify configuration or runtime errors.

For critical failures, such as service crashes or DNS issues, manual administrative intervention was required. However, detailed logging and modular service architecture enabled rapid diagnosis and recovery. These mechanisms collectively improved the fault tolerance and stability of the system.

3.3 Results and Discussion

This section presents the outcomes of the mail server implementation and discusses how the project objectives were achieved.

3.3.1 Results Achieved

The project successfully achieved the following outcomes:

- Deployment of a fully functional, self-hosted mail server on a VPS
- Successful integration of Postfix (SMTP), Dovecot (IMAP/POP3), and OpenDKIM
- Implementation of industry-standard email authentication mechanisms:
    - SPF
    - DKIM
    - DMARC
    - Reverse DNS (PTR)
- A web-based email interface using Roundcube
- Secure and structured mailbox storage using Maildir
- Full control over mail flow, authentication, and logging
- End-to-end email communication without reliance on third-party email providers

The system was tested against multiple external email platforms, demonstrating correct authentication and protocol compliance.

3.3.2 Discussion

The core problem addressed by this project was the lack of understanding, control, and reliability when using third-party or free email services for institutional or organizational communication. Such services often provide limited transparency, unpredictable spam filtering behavior, and restricted administrative control.

By building a mail server from scratch, this project provided hands-on experience with real-world email infrastructure and demonstrated how modern email systems operate internally. The implementation highlighted the importance of DNS-based authentication mechanisms and their role in combating spoofing and unauthorized mail usage.

Although emails were still classified as spam by some external providers, this behavior was correctly identified as a **domain reputation issue** rather than a configuration fault. This observation reflects real-world deployment challenges and reinforces the fact that email deliverability depends not only on technical correctness but also on long-term domain trust.

Overall, the project achieved its educational and technical goals by delivering a secure, standards-compliant, and scalable mail server architecture. The system serves as a strong foundation for future enhancements such as spam filtering, antivirus integration, enforced TLS encryption, and reputation monitoring.

Chapter 4

Engineering Standards and Mapping

This chapter discusses the broader engineering implications of the implemented mail server system, including its impact on users, society, environment, ethics, and sustainability. It also presents a structured mapping of the project with Program Outcomes (POs), Complex Engineering Problems (EPs), and Engineering Activities (EAs) in accordance with outcome-based education and accreditation standards.

4.1 Impact on Society, Environment and Sustainability

4.1.1 Impact on Life

The proposed mail server system improves the practical understanding and experience of users and administrators by providing a real-world, self-hosted email infrastructure. Users gain reliable access to email services through a web-based interface, while administrators gain hands-on exposure to server deployment, configuration, monitoring, and troubleshooting.

From an educational perspective, the system enhances learning by allowing students to directly interact with core operating system concepts such as process management, networking, service orchestration, and system security. This practical exposure strengthens technical competence and problem-solving ability, which are essential for modern computing professionals.

By automating email handling, authentication, and storage, the system reduces dependency on third-party email services and increases awareness of how critical internet communication infrastructure operates behind the scenes.

4.1.2 Impact on Society & Environment

From a societal perspective, the project promotes digital self-reliance and technical transparency by demonstrating how organizations can deploy and manage their own communication infrastructure instead of depending entirely on external service providers. This is particularly relevant for educational institutions and small organizations seeking greater control over data and communication policies.

Environmentally, the system supports sustainability by enabling fully digital communication and reducing the need for paper-based notices, printed documents, and physical correspondence. Email-based communication minimizes travel, printing, and physical storage requirements, thereby reducing energy consumption and environmental footprint.

The use of virtualized infrastructure (VPS) further optimizes resource utilization compared to traditional physical servers, contributing to energy-efficient computing practices.

4.1.3 Ethical Aspects

Ethical considerations were incorporated throughout the design and implementation of the mail server system. The key ethical aspects include:

- **Data Privacy:** User emails and credentials are stored securely on the server with controlled access, ensuring confidentiality of personal and organizational communication.
- **Authentication and Authorization:** Access to mailboxes and administrative services is restricted to authenticated users, preventing unauthorized access.
- **Email Integrity and Trust:** The implementation of SPF, DKIM, and DMARC ensures that outgoing emails are authenticated and protected against spoofing and impersonation.
- **Responsible Resource Usage:** The system was built entirely using open-source software in compliance with licensing and ethical usage guidelines.
- **Transparency:** Mail flow, authentication results, and delivery behavior can be analyzed through logs, ensuring accountability and traceability.

The project avoids unethical data usage and aligns with fundamental principles of responsible computing and engineering practice.

4.1.4 Sustainability Plan

The long-term sustainability of the mail server project is ensured through the following measures:

- Use of widely adopted open-source technologies such as Linux, Postfix, Dovecot, OpenDKIM, Apache, PHP, MariaDB, and Roundcube, eliminating licensing costs.
- Deployment on a VPS with a static IP address, allowing scalability and future resource upgrades as required.
- Modular service-based architecture, enabling independent maintenance and upgrades of individual components.
- Low client-side requirements, as users only need a standard web browser or email client to access services.
- Future-ready design that allows integration of additional features such as spam filtering, antivirus scanning, TLS enforcement, and monitoring tools without major architectural changes.

Overall, the system is economically feasible, technically maintainable, and environmentally responsible, making it suitable for long-term use and further development.

4.2 Project Management and Team Work

This project was executed as a collaborative group effort, where responsibilities were divided among team members based on technical domains to ensure efficiency, accountability, and smooth integration. The task allocation allowed parallel development of system components while maintaining coordination through regular discussions and testing phases.

Each member was assigned a specific module of the system, covering infrastructure setup, mail server configuration, security and deliverability, web services, database integration, and webmail client deployment. The modular division ensured that complex system dependencies were handled systematically.

Task Distribution Among Team Members

Member 1: Munawer Mahtab (1047):

Postfix (SMTP Server Configuration)
This member was responsible for installing and configuring **Postfix** as the SMTP server. Tasks included setting the system mail name, configuring hostname and domain parameters, enabling Maildir delivery, and ensuring proper SMTP communication for sending and receiving emails. Integration with OpenDKIM was also handled to support email authentication and improve deliverability.

Member 2: Sayed Sheikh (1879):

Dovecot Configuration and Mail Testing
This member configured **Dovecot** for IMAP and POP3 services, ensuring proper mailbox access and authentication. Mail storage was synchronized with Postfix using the Maildir format. Extensive testing was conducted to verify user authentication, inbox access, message retrieval, and compatibility with the webmail client. Troubleshooting of login and mailbox issues was also handled.

Member 3: Fateha Hossain Anushka (1844):

VPS Installation, Domain Setup, DNS Configuration

This member handled the **VPS setup** and **domain configuration**, including DNS records such as A, MX, SPF, DKIM, and DMARC. The rationale for using a VPS instead of localhost was addressed, as a static public IP is essential for email services. Email deliverability mechanisms were implemented and tested, including DKIM signing and SPF validation. Issues related to spam classification due to the use of a free domain were identified and documented.

Member 4: Kamrozzaman Maimon (1232):

Apache, PHP, MariaDB, and MySQL Driver Configuration
This member installed and configured the **web server and database stack**, including Apache, PHP, MariaDB, and required PHP extensions. The Roundcube database and user credentials were created,

permissions were configured, and database connectivity was tested. This ensured a stable backend environment for the webmail application.

Member 5: Hanjala Habib Sadik (1010):

Roundcube Installation, Configuration, and Webmail Testing
This member was responsible for installing and configuring **Roundcube Webmail**, integrating it with Postfix and Dovecot. Apache virtual hosts were configured for the webmail subdomain, and Roundcube settings were adjusted to ensure proper IMAP and SMTP communication. End-to-end testing was conducted to verify user login, email sending, receiving, and web-based mailbox management.

Regular coordination ensured that dependencies between components—such as DNS records, mail authentication, database connectivity, and webmail access—were correctly aligned. System logs, test emails, and real-world email delivery tests were collectively reviewed to validate functionality.

The collaborative approach allowed the team to successfully deploy a fully functional mail server with secure authentication, webmail access, and real-world testing, demonstrating effective project management and teamwork.

4.2.1 Cost Analysis and Budget Planning

This project was designed and implemented with a strong focus on cost efficiency by using open-source software, free testing tools, and minimal paid infrastructure. The overall budget was kept low while still enabling real-world deployment and testing of a functional mail server system.
Actual Budget (Implemented Cost):

Component	Description	Cost
VPS Hosting	Virtual Private Server with static public IP used to host mail services and webmail	**1200 BDT**
Domain	Free subdomain used for mail and webmail services	**Free**
Mail Server Software	Postfix, Dovecot, OpenDKIM	**Free (Open Source)**
Web Server & Database	Apache, PHP, MariaDB	**Free (Open Source)**
Webmail Client	Roundcube Webmail	**Free (Open Source)**
Testing Tools	Email deliverability and DNS verification tools (e.g., MXToolbox, MailTester)	**Free**

SSH Client	Bitvise SSH Client for secure remote server access	**Free**
Total Cost		**1200 BDT**

Alternate Budget (Cost-Minimized Academic Deployment):

Component	Alternate Option	Rationale
Server	University Lab Server or Institutional Cloud	Eliminates VPS cost
Domain	University-provided subdomain	Improves email trust and deliverability
Testing Tools	Open-source command-line tools	No external dependency
Total Estimated Cost		**Near Zero**

Budget Rationale

- A **VPS was necessary** instead of localhost because mail servers require a **static public IP address** for reliable SMTP communication and DNS-based authentication.
- A **free subdomain** was chosen to minimize cost; however, it impacts email reputation and results in emails being flagged as spam by external providers.
- All core services and tools used were **open-source**, reducing licensing costs.
- Free testing platforms such as **MXToolbox** and **MailTester** were sufficient to validate DNS records, mail headers, and authentication results.
- **Bitvise SSH Client** provided secure and free remote access for server administration.

Limitations Due to Budget Constraints

Due to the use of a free subdomain and lack of paid email reputation services:

- Emails sent to external providers may be classified as spam.
- Domain reputation cannot be fully controlled.
- Advanced email deliverability services and monitoring were not included.

These limitations were documented as part of the project results and provide scope for future improvements.

Conclusion of Cost Analysis

The project successfully demonstrates that a functional and secure mail server system can be implemented with **minimal financial investment**. With a total cost of **1200 BDT**, the system achieves practical deployment, real-world testing, and learning outcomes aligned with operating systems and networking concepts.

4.3 Complex Engineering Problem

The project addresses a **complex engineering problem** involving distributed systems, secure communication, workflow automation, and scalability under real-world constraints.

4.3.1 Mapping of Program Outcome

Table 4.1: Justification of Program Outcomes

PO's	**Justification**
PO1	Applied knowledge of computer networks, operating systems, databases, and web technologies to design a full-stack system
PO2	Analyzed limitations of manual academic workflows and email deliverability challenges
PO3	Designed and implemented a scalable, secure, and automated solution

4.3.2 Complex Problem Solving

Table 4.2: Mapping with complex problem solving.

EP1 Dept of Knowledge	**EP2** Range of Conflicting Requirements	**EP3** Depth of Analysis	**EP4** Familiarity of Issues	**EP5** Extent of Applicable Codes	**EP6** Extent Of Stakeholder Involvement	**EP7** Inter-dependence
✓	✓	✓	✓	✓	✗	✗

4.3.3 Engineering Activities

Table 4.3: Mapping with complex engineering activities.

EA1 Range of resources	**EA2** Level of Interaction	**EA3** Innovation	**EA4** Consequences for society and environment	**EA5** Familiarity
✓	✓	✗	✓	✗

Chapter 5

Conclusion

This chapter summarizes the overall work carried out in the project, highlights the key limitations encountered during implementation, and discusses possible directions for future improvements and extensions.

5.1 Summary

In this project, a complete mail server system was designed, implemented, and tested using open-source technologies and a Virtual Private Server (VPS). The system supports core email functionalities, including sending, receiving, and managing emails through standard protocols such as SMTP, IMAP, and POP3. Postfix was used as the SMTP server for mail transmission, while Dovecot was configured to provide secure access to mailboxes.

To enable web-based email access, Roundcube Webmail was deployed and integrated with Apache, PHP, and MariaDB. DNS records, including A, MX, SPF, DKIM, and DMARC, were configured to establish proper mail routing and authentication. OpenDKIM was implemented to sign outgoing emails, improving message authenticity and reducing spoofing risks.

The project demonstrated practical knowledge of operating system services, networking concepts, and system administration. Real-world testing was conducted by sending emails to external mail providers, validating DNS authentication mechanisms, and analyzing mail logs. Overall, the project successfully achieved its objectives within a limited budget using free and open-source tools.

5.2 Limitation

Despite successful implementation, the project has several limitations:

- Emails sent from the server are often delivered to the spam folder of external email providers. This occurs mainly due to the use of a free subdomain, limited domain reputation, and lack of paid email trust services.
- The system does not include advanced spam filtering mechanisms such as SpamAssassin or machine-learning-based filtering.
- TLS encryption for SMTP and IMAP was implemented at a basic level; advanced security hardening was not fully explored.
- The system is designed for academic and small-scale use and is not optimized for high

traffic or enterprise-level deployment.

- Continuous monitoring and automated backup mechanisms were not implemented due to time and resource constraints.

5.3 Future Work

Several enhancements can be made to improve the system in future iterations:

- Migrating to a paid top-level domain (e.g., .com or .edu) to improve email reputation and deliverability.
- Implementing advanced spam filtering tools such as SpamAssassin and virus scanning using ClamAV.
- Enforcing full TLS encryption for all mail services and applying stricter DMARC policies.
- Adding monitoring, logging dashboards, and automated backup solutions for system reliability.
- Scaling the system to support multiple domains and higher user loads.
- Integrating analytics and usage reporting for administrators.
- Deploying the system on institutional infrastructure or cloud platforms with improved trust and reliability.

Conclusion of the Project

The project successfully demonstrates that a functional and secure mail server can be built using open-source technologies with minimal cost. While certain limitations exist due to resource constraints, the system provides a strong foundation for learning and future enhancement, fulfilling both academic and practical objectives.