KL University

Open Source Engineering

Project Report

Submitted in partial fulfilment of the requirements for

Open Source Engineering

Submitted by:

Atmakuri Dhanush

ID: 2400033102

Branch: Computer Science and Engineering (CSE)

Academic Year: 2025–2026

Acknowledgment

I would like to express my sincere gratitude to **KL University** and the Department of Computer Science and Engineering (CSE) for giving me the opportunity to work on this project as part of the Open Source Engineering course. This project allowed me to explore the practical side of Linux, open source contribution, and self-hosted services beyond what is usually covered in theory.

I am deeply thankful to my faculty mentors for their continuous guidance, valuable feedback, and encouragement. Their support helped me understand concepts such as encryption, licensing, privacy, and server management in a structured and systematic way. The discussions during lab sessions and doubt-clearing hours were particularly helpful in connecting theoretical topics with real tools and commands.

I am also grateful to the global open source community. Official documentation, community forums, GitHub issues, and blog posts played a huge role in solving practical problems during this work. Finally, I wish to thank my friends and classmates for their cooperation, motivation, and peer learning during the execution of this project.

Declaration

I, Atmakuri Dhanush, bearing ID number 2400033102 of the branch Computer Science and Engineering (CSE), hereby declare that this report titled **Open Source Engineering** is a record of my own work carried out during the academic year 2025–2026. This work has not been submitted to any other institution for the award of any degree, diploma, or certificate.

Wherever external sources of information have been used, they have been duly acknowledged in the references or within the text. Any contributions from other individuals or projects have been clearly mentioned.

Signature of Student: _____

Date: _____

Contents

1. About the Linux Distribution (Ubuntu)

Ubuntu 25.0 (latest version) was used during this project. Ubuntu is a Debian-based Linux distribution that provides security, stability, and excellent package support. It is widely used for development, server hosting, and cloud-based environments.

Technical Features

- Uses Linux kernel with GNOME desktop
- Secure system services via systemd
- AppArmor-based sandboxing
- Large official software repositories

Package Management

Ubuntu uses APT as the primary package manager:

```
sudo apt update
sudo apt install <package>
```

This helped install tools such as GPG and Mattermost with ease.

Why Ubuntu Was Suitable

- Easy installation
- Active support and documentation
- Great for server hosting and learning Linux

2. Encryption and GPG

Encryption plays a crucial role in ensuring confidentiality and integrity of data in modern computing. GNU Privacy Guard (GPG) is a powerful open source tool that implements the OpenPGP standard, providing encryption, decryption, signing, and verification functionalities.

In the context of this project, GPG was used to generate key pairs, encrypt files, and sign content. This helped understand how real-world developers protect sensitive data and verify authenticity in distributed environments.

Key Concepts

- Public key – encryption and verification
- Private key – decryption and signing

Commands used:

```
gpg --full-generate-key
gpg --list-keys
gpg --encrypt -r <email> file.txt
```

3. Sending Encrypted Email

Encrypted email communication was tested using my institutional email: **2400031161@kluniversity.in**. The goal was to understand how end-to-end encryption works using GPG and how secure message transfer is achieved without exposing the contents of the message to intermediate servers or applications.

How Email Encryption Works

When sending encrypted email, GPG uses asymmetric cryptography. The core workflow is:

1. The sender obtains or imports the public key of the recipient.
2. The sender encrypts the message using the recipient's public key.
3. Only the recipient, who possesses the corresponding private key, can decrypt it.
4. Optionally, the sender can sign the message to verify identity and integrity.

Two concepts are involved:

- **Confidentiality** – nobody except the intended recipient can read the mail.
- **Authentication & Integrity** – signing ensures that the message was not modified.

Commands Used for Sending Encrypted Mail

GPG supports ASCII-armored output which is suitable for sending through email clients. For example:

```
gpg --armor --encrypt --sign \
    -r 2400031161@kluniversity.in message.txt
```

This produces a readable encrypted block such as:

```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v2
hQEMA5htS....
-----END PGP MESSAGE-----
```

The encrypted text can be pasted directly into an email body or attached as a file.

Decrypting the Message

The recipient decrypts using:

```
gpg --decrypt message.asc
```

If a signature exists, GPG verifies the sender's identity:

```
gpg --verify message.asc
```

Email Client Integration

Tools like Thunderbird and Outlook support GPG through plugins:

- Thunderbird has built-in OpenPGP integration.
- Emails are encrypted and decrypted automatically once keys are imported.

- The signature validity and trust level are displayed to the recipient.

This demonstrates how encrypted email systems are used in real organizations and secure communication channels such as:

- Developer communication
- Security incident reporting
- Signed release announcements

Why Email Encryption Matters

Handling sensitive content such as passwords, confidential attachments, personal data, or private discussions requires protection. Without encryption, messages can be intercepted or viewed by:

- Mail servers
- Internet service providers
- Man-in-the-middle attackers
- Compromised systems

GPG solves these problems by ensuring:

- Only the intended person can decrypt
- The sender can prove the message is authentic
- The data remains unchanged in transit

4. Privacy Tools (PRISM-Break)

The following open-source privacy tools were used:

- Firefox – privacy focused browser
- LibreOffice – open-source document editor
- Thunderbird – secure email client
- Tor Browser – anonymity network
- KeePassXC – secure password manager

These tools provide better privacy and freedom from proprietary tools that track user data.

5. Open Source License Used (MIT)

In this project, the open-source license chosen and explored was the **MIT License**. It is one of the most widely used and highly permissive open-source licenses in the world. The MIT License is commonly selected in software projects because of its simplicity, flexibility, legal clarity, and developer-friendly nature. Many modern software frameworks, tools, and libraries are licensed under MIT because it encourages both open-source usage and commercial integration.

Core Characteristics of the MIT License

The MIT License gives users and developers the freedom to:

- Use the software for any purpose, including personal, academic, or commercial use.
- Modify and extend the source code.
- Distribute copies of the original or modified software.
- Combine the software with proprietary applications.

Unlike restrictive licenses (such as GPL or AGPL), MIT does **not** require modified versions of the software to be open-sourced. This permissive nature makes the license suitable for small projects, startups, and individuals who want their work reused widely without complex legal constraints.

Legal Requirements

Even though MIT is permissive, it includes one very important requirement:

- Any distributed version of the software must include the original license notice.

This means that users must preserve:

- The original copyright
- The permission statement
- The disclaimer of warranty

This ensures proper credit is given to the original author.

Why MIT License is Developer-Friendly

The MIT License is extremely short (typically 160–200 words) and easy to understand. Other licenses may contain long legal statements, copyleft restrictions, or ambiguity. MIT focuses on developer freedom rather than control

Why I Used MIT in This Project

For this project, MIT was selected because:

- It offers maximum freedom for others to use or improve the work.
- It is simple and widely accepted in the open-source community.
- It is suitable for both personal learning projects and real-world applications.
- It does not restrict how the software is shared or deployed.

6. Self-Hosted Server: Mattermost

Mattermost is an open-source, self-hosted communication and collaboration platform used as an alternative to Slack. It allows teams to send messages, share files, and collaborate securely without depending on a third-party cloud service. Mattermost comes with two main components: the server backend and the web/mobile client interfaces. The backend is written in Go and handles authentication, message storage, real-time communication, and API requests. The frontend is built using React and provides the user interface for channels, chats, and file sharing.

Mattermost works using a client–server architecture. Clients connect to the server over HTTP/HTTPS, and messages are transmitted using WebSockets for real-time updates. All chat messages, channel details, and users are stored in a database such as PostgreSQL. When a user sends a message, the server saves it in the database and immediately broadcasts it to all connected clients in the channel. Mattermost supports features like private chats, channels, file uploads, notifications, and role-based access control.

Because it is self-hosted, organizations maintain full privacy and control over their data. Administrators can configure authentication methods, storage, backup, and integrations with external tools. This makes Mattermost suitable for secure on-premise deployment in academic, enterprise, and developer environments.

Installation

```
wget https://releases.mattermost.com/latest
tar -xvzf mattermost.tar.gz
sudo systemctl start mattermost
```

Poster

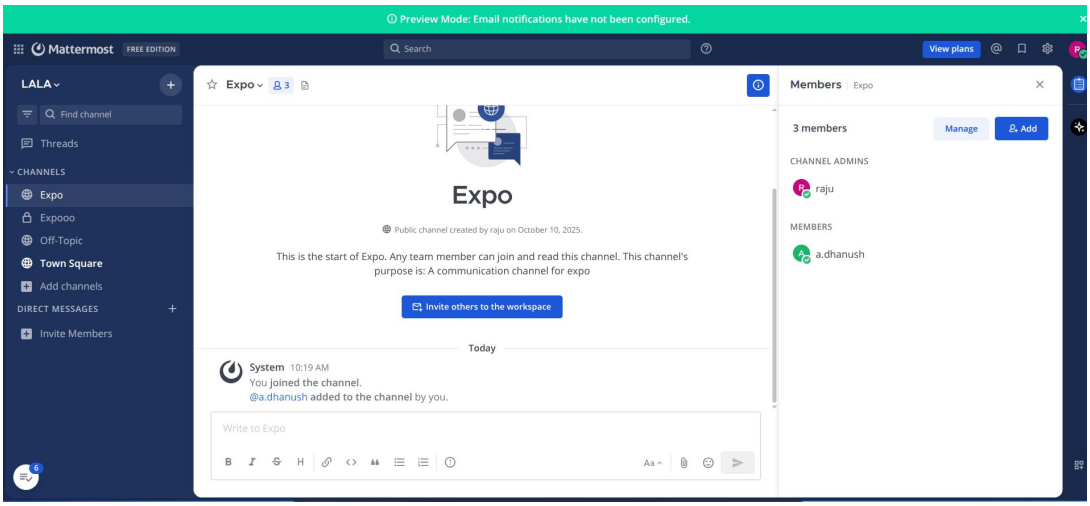

Figure 6.1: Mattermost Self-Hosting Poster

Localization

A comprehensive Hindi-language documentation guide was prepared for the Mattermost platform to enhance accessibility for native speakers. The document included detailed instructions on installation procedures, channel navigation, and basic communication features. This localization effort supported inclusivity and promoted wider usability among Hindi-speaking users.

7. Open Source Contributions (PRs and Issues)

PR 1 – Api to weather category in meto

Repository: public-apis/public-apis
PR: https://github.com/public-apis/public-apis/pull/5012

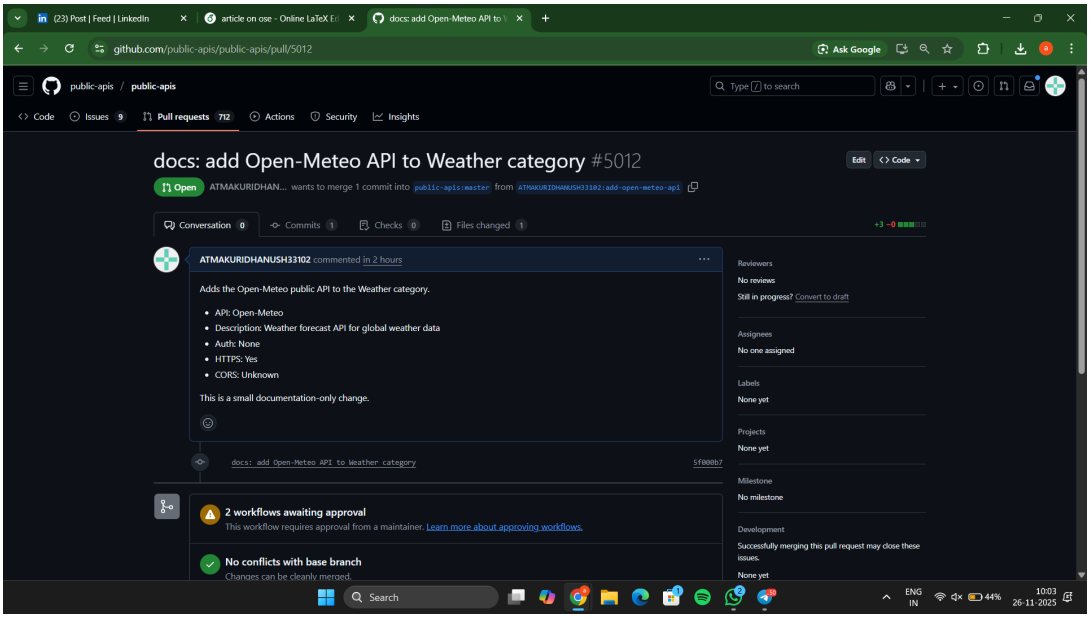

Figure 7.1: PR 1 – Open meto API to Weather

PR 2 – Add missing CLI options to manpage and cli docs

Repository: nodejs/node **PR:** https://github.com/nodejs/node/pull/60857

Added Baby-Step Giant-Step implementation + tests.

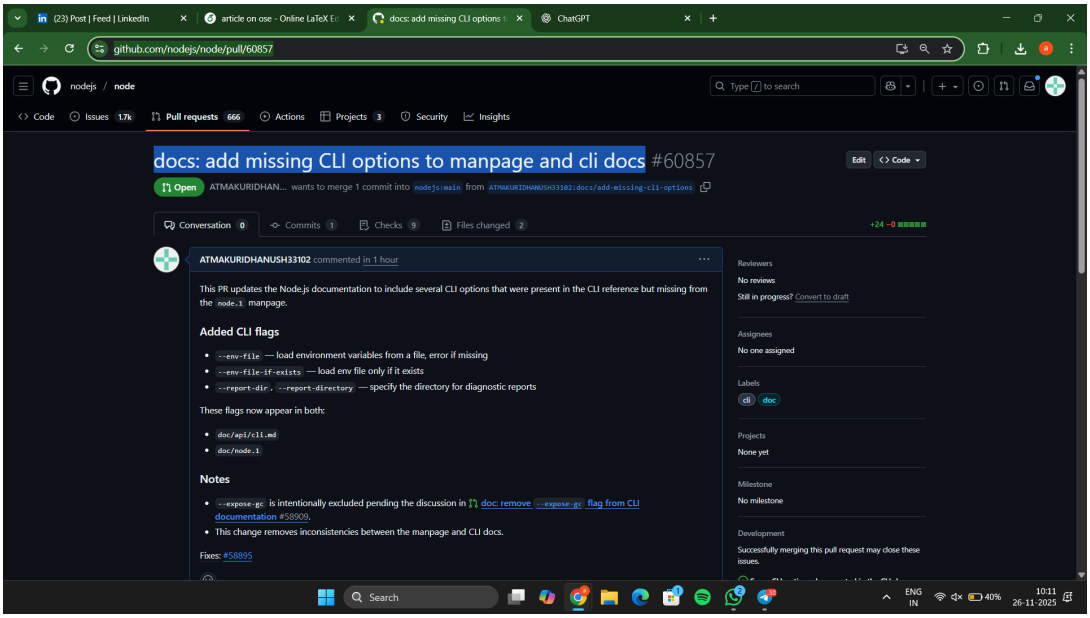

Figure 7.2: PR 2 – Algorithm Contribution

PR 3 – First Contribution

Repository: firstcontributions/first-contributions **PR:** https://github.com/firstcontributions/first-contributions/pull/106608

Added my name to contributor list.

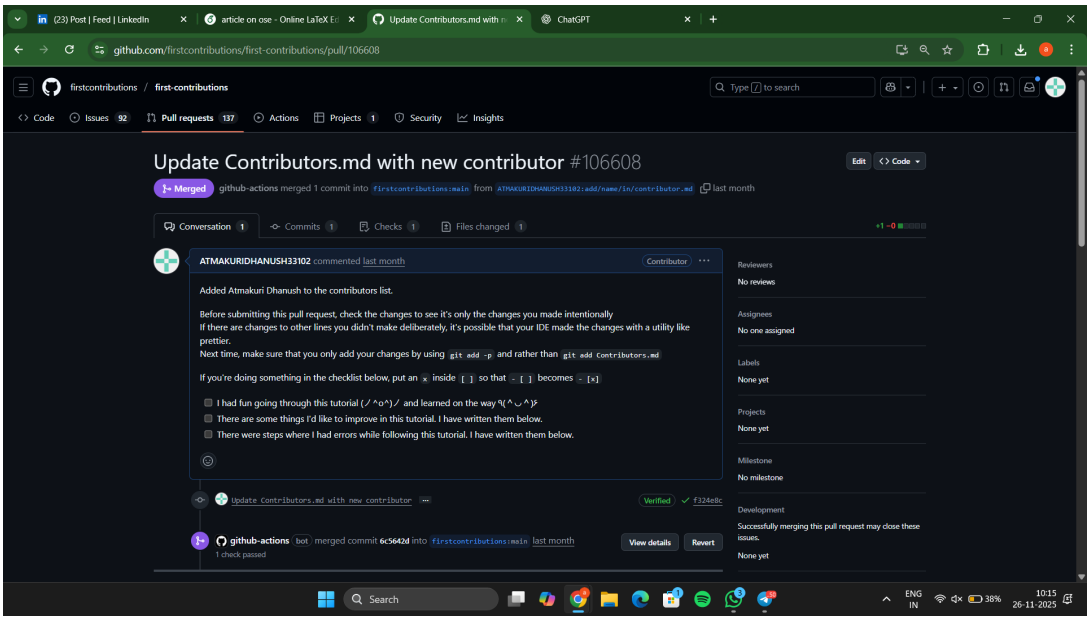

Figure 7.3: PR 3 – Contributor Update

8. LinkedIn Posts (Professional Outreach)

In addition to the practical work completed during the project, I also documented and shared my progress on LinkedIn. Posting about the work helped create visibility for the project and allowed me to connect with other students, developers, and open-source enthusiasts. These posts served as a personal record of growth and helped in building a professional online presence.

Post 1 – Self-Hosted Mattermost Deployment

This post focused on hosting the Mattermost server on Ubuntu. It highlighted the importance of self-hosting for privacy and data control, the steps involved in installation, and the experience of working with a real-time messaging service. The post also emphasized the practical learning involved in configuring backend services, managing dependencies, and understanding system administration concepts.

LinkedIn Link:

https://www.linkedin.com/posts/varada-raju-suggu$_{o}pensource$ – $mattermost$ – $selfhosted$ – $ugcPost$ – 7382299359138385920 – $IIb6?utm_{s}ource$ = $shareutm_{m}edium$ = $member_{d}esktoprcm$ = $ACoAAFL85EIBdQKRkXMVzv0dBcJa3NbjywWHJNQViewPost$

Post 2 – Open Source Engineering Blog

This post shared insights and reflections about the entire Open Source Engineering course. It discussed working with Linux, GPG, privacy tools, licensing, and contributing to open-source repositories through issues and pull requests. The post demonstrated how practical hands-on experience enhances technical learning and motivates continuous participation in open-source development.

LinkedIn Link:

https://www.linkedin.com/pulse/my-learning-journey-open-source-engineering-course-atmakuri-dhanush-shl4fview Post

Conclusion

The Open Source Engineering course helped me understand Linux, encryption, privacy, and open-source development. Hosting Mattermost, contributing PRs, and using open tools improved my confidence in real-world development workflows. I gained knowledge of system administration, cryptography, secure communication, and community contribution.