KL UNIVERSITY

Open Source Engineering Project Report

Submitted in partial fulfilment of the requirements for the course
Open Source Engineering

Submitted by:

Kancharan Sharvandeep Bhardwaj
ID: 2400032427
Branch: CSE
Academic Year: 2025-2026

Declaration

I, KANCHARANA SHARVANDEEP BHARDWAJ (ID: **2400032427**), hereby declare that this report titled **"Open Source Engineering Project Report"** is a genuine record of my own work carried out during the academic year **2025–2026** as part of the coursework for the **Open Source Engineering** subject at **KL University**.

I confirm that the work presented in this report is based on my personal learning, experimentation, self-hosting activities, open-source contributions, and documentation prepared by me.

Signature of Student: _____

Acknowledgment

I would like to express my sincere gratitude to **KL University** for giving me the opportunity to work on this Open Source Engineering project. This course has helped me gain practical knowledge in Linux, GPG, GitHub workflows, privacy tools, and self-hosting servers, all of which played an important role in building my technical skills.

I am especially grateful to **Dr. Sripath Roy Koganti**, our course faculty, for his continuous guidance, support, and encouragement throughout the project. His teaching, feedback, and expertise in open-source concepts helped me understand the subject deeply and motivated me to explore beyond the classroom.

I also thank the **KLGLUG community** and the open-source maintainers of the repositories I contributed to. Their documentation, reviews, and community support made my contribution experience smoother and more meaningful.

My heartfelt thanks to my classmates and friends for their cooperation during practical sessions, discussions about self-hosting, and mutual support during the project timeline.

Finally, I would like to acknowledge all open-source developers whose contributions make tools such as Ubuntu, Git, Mumble, and GPG freely available for students and learners across the world. This project has been a valuable learning experience, and I am thankful to everyone who contributed directly or indirectly to its completion.

Contents

1. About the Linux Distribution (Ubuntu 22.04 LTS)

Ubuntu 22.04 LTS (Jammy Jellyfish) is the Linux distribution used throughout this project. It is a stable, user-friendly, open-source operating system based on Debian. Ubuntu is widely used by developers, students, and professionals because it provides a balance of simplicity, strong performance, and excellent community support. Since it is an LTS (Long Term Support) version, Ubuntu 22.04 receives security updates and maintenance patches for five years, making it suitable for academic work, open-source development, and self-hosting practical servers. Throughout this course, Ubuntu was used for working with GitHub, managing GPG keys, executing commands, preparing documentation, and handling basic server tasks.

Ubuntu 22.04 uses the Linux 5.15+ kernel, which supports modern hardware such as Intel processors, SSDs, Wi-Fi adapters, and graphics drivers. The system uses systemd for handling system services, booting, and background processes. The GNOME 42 desktop environment provides a clean and modern interface, making navigation smooth even for new Linux users. The combination of GNOME and the Linux kernel contributes to a reliable and stable system suitable for both development and general usage.

System Architecture and Features

Ubuntu 22.04 provides an efficient and responsive environment for development tasks. Some of the features I used include:

- Smooth and clean GNOME desktop experience
- Easy access to the terminal
- Good performance even during multitasking
- Pre-installed utilities like Nano, Gedit, and System Monitor
- Strong compatibility with development tools like Git and VS Code

The Linux kernel ensures good support for modern hardware, and Ubuntu's updates kept the system secure throughout the project period.

Installation Method

Ubuntu 22.04 was installed as a dual-boot system alongside Windows. During installation, I faced a partition error, which I resolved using a partition wizard tool under the guidance of my professor. After successfully fixing the partition, Ubuntu installed properly and the system booted without any issues. After installation, I performed basic updates and started installing the necessary packages and tools needed for the course.

Commands Used Frequently

Most of the work in this project involved using the terminal. Some of the main commands I used are:

Basic Navigation:
ls
cd
pwd

Package Management:
sudo apt update
sudo apt upgrade
sudo apt install <package>

File Editing:
nano

Git Commands for PR Work:
git add .
git commit -m "message"
git push
git status

These commands helped me understand how Linux works internally and how GitHub workflows operate in a real open-source environment.

Software Installation

Using APT, I installed several tools during the project, such as:

- Git
- Mumble client and server
- VS Code
- Other supportive utilities

Ubuntu's package manager made installation simple and reliable without requiring manual installers.

Desktop Environment

The GNOME 42 desktop environment provided:

- A clean and minimal design
- Smooth animations
- Easy access to settings

- A simple workflow for switching between apps
- Support for light and dark modes
- Good integration with VS Code, Terminal, and GPG tools

This made Ubuntu easier and more comfortable to use even for someone new to Linux.

Why Ubuntu Was Chosen

I chose Ubuntu 22.04 for this project because:

- It is stable and secure
- Beginner- friendly
- Recommended by my professor
- Works smoothly with Git and GitHub
- Supports GPG encryption
- Has strong online documentation
- Lightweight and performs well
- Suitable for self-hosting servers

These reasons made Ubuntu a perfect choice for learning and performing open-source engineering tasks.

How Ubuntu Helped in This Project

Ubuntu was used for:

- Running Linux commands
- Managing GitHub repositories
- Generating and managing GPG keys
- Signing Git commits
- Installing and running the Mumble server
- Editing documentation and markdown files
- Testing and contributing to open-source projects

Overall, Ubuntu 22.04 LTS provided a smooth and powerful platform for completing all parts of the Open Source Engineering course.

2. Encryption and GPG

GNU Privacy Guard (GPG) is an open-source tool used for encryption, decryption, digital signing, and secure key management. It follows the OpenPGP standard and is widely used in open-source projects to verify identities, protect sensitive data, and authenticate commit authors. In this project, GPG was mainly used for generating a personal keypair and signing Git commits during pull request submissions.

Ubuntu includes the GnuPG tool by default, so generating and managing keys through the terminal was simple. Using GPG helped ensure that all commits made during the project were verified and linked securely to my identity.

The GPG key created for this project includes one primary key and one subkey. The details are:

Name: sharvandeep
Email: 2400032427@kluniversity.in
Key ID: 75A43C246E1F30C0
Subkey: 9C7AF06DEB093987
Created on: October 29, 2025
Key Type: RSA 4096
Expiry: 1 year

All key-related data is stored in the default GPG directory:
~/.gnupg/

Basic GPG Commands Used

Here are the commands I used during the project:

To list all public keys:
gpg --list-keys

To list secret/private keys:
gpg --list-secret-keys

To generate a new key pair:
gpg --full-generate-key

To view key fingerprints:
gpg --fingerprint

These commands helped me confirm the existence of my key, verify fingerprints, and manage my keypair easily.

Signing Git Commits

One major use of GPG in this project was signing Git commits. Signed commits show a "Verified" badge on GitHub, proving the commit was authored by me.

After generating the GPG key, I configured Git with these commands:

To set the signing key:
git config --global user.signingkey 75A43C246E1F30C0

To enable commit signing permanently:
git config --global commit.gpgsign true

With this setup, every commit I pushed to GitHub was automatically signed using my GPG key.

Importance of GPG in This Project

GPG was useful in the following ways:

- Verified my identity on GitHub
- Ensured authenticity of commits
- Helped me understand secure developer workflows
- Prepared me for contributing to larger open-source communities

Even though I did not use advanced features such as encrypted email or file encryption, learning to generate keys and sign commits was an important step in understanding open-source engineering and secure development practices.

3. Privacy Tools (PRISM-Break)

Privacy is an important part of open-source engineering and secure digital usage. During this course, I explored several privacy-focused tools listed on PRISM-Break, a well-known open-source resource that recommends alternatives to proprietary and data-tracking software. These tools help protect user identity, prevent third-party tracking, secure online communication, and provide a more controlled digital environment.

From PRISM-Break, I selected five important privacy tools that I used or explored during this project: Firefox, Signal Messenger, Tor Browser, KeePassXC, and Mullvad VPN. Each tool focuses on a different aspect of privacy such as web browsing, messaging, password storage, and anonymous networking. These tools helped me understand the importance of privacy and data protection while working with open-source applications.

1. Firefox (Web Browser)

Firefox is an open-source web browser developed by Mozilla. It is widely known for its strong focus on privacy, transparency, and user control. Unlike many commercial browsers, Firefox does not track user behavior for advertising purposes. It includes built-in features such as Enhanced Tracking Protection, which blocks third-party cookies, ads, and trackers automatically. Firefox also supports a large number of privacy-oriented extensions, such as uBlock Origin, Privacy Badger, and HTTPS-Everywhere.

I used Firefox during this project for browsing documentation, opening GitHub repositories, and downloading Linux tools. Since it is open source, its code is publicly available for anyone to audit, making it a trustworthy choice for privacy-focused users.

2. Signal Messenger (Secure Messaging)

Signal is a widely trusted messaging application that provides end-to-end encryption for all chats, calls, voice messages, and file transfers. Unlike traditional messaging apps that store data on centralized servers, Signal uses the Signal Protocol, which ensures that only the sender and receiver can access the content of messages. Signal does not store metadata, chat backups, or conversation history on cloud servers.

I explored Signal as part of the privacy tools section to understand how encrypted communication works in real-time applications. Many security researchers, journalists, and developers use Signal because of its open-source nature and zero-data collection policies.

3. Tor Browser (Anonymous Browsing)

Tor Browser allows anonymous browsing by routing internet traffic through a decentralized network of volunteer-run servers known as nodes. This technology, called onion routing, hides the user's IP address and makes it very difficult for websites, ISPs, or advertisers to track online activity.

Tor Browser is often used for:

- Preventing identity tracking
- Bypassing censorship
- Accessing restricted content
- Protecting privacy on public networks

Although Tor is slower than normal browsers due to multi-layer routing, it is one of the most powerful privacy tools available. During this project, I learned how Tor handles anonymity and why it is recommended for people who need high levels of privacy.

4. KeePassXC (Password Manager)

KeePassXC is an offline, open-source password manager that stores all passwords locally in an encrypted database secured by AES-256 encryption. Unlike cloud-based password managers, KeePassXC does not sync or upload any data automatically, which makes it more privacy-friendly. Users have complete control over where their password file is stored.

I explored KeePassXC to understand how password managers work and why using strong, unique passwords is important for security. KeePassXC can also store SSH keys, OTP codes, and secure notes, making it useful for developers working with private repositories and login credentials.

5. Mullvad VPN (Privacy-Focused VPN)

Mullvad VPN is a no-logs VPN service known for its strict privacy policies. It allows users to create an account without providing any personal information, not even an email address. Mullvad uses modern VPN protocols such as WireGuard and OpenVPN, providing strong encryption and fast speeds.

Mullvad does not track user activity, does not store IP logs, and does not monitor browsing behavior. The service is open about how it handles user data and is one of the few VPNs trusted by privacy researchers. Learning about Mullvad helped me understand how VPNs protect internet traffic by encrypting data and hiding the user's real IP address.

Importance of Privacy Tools

Using privacy tools is essential in today's digital world. These tools help:

- Protect personal information
- Reduce online tracking
- Improve security while using public networks
- Prevent unauthorized data collection
- Support safe access to websites and communication apps

For open-source engineering, privacy tools ensure that communication with developers, contribution workflows, and online research remain secure. They also teach the importance

of transparency and the value of using open-source alternatives instead of proprietary, data-collecting platforms.

These five tools—Firefox, Signal, Tor Browser, KeePassXC, and Mullvad VPN—gave me a clear understanding of how privacy can be maintained using free and open-source software. Each tool plays a unique role in keeping data secure and maintaining user anonymity, which is becoming increasingly important in modern digital environments.

4. Open Source License Used (BSD 3-Clause License)

The self-hosted application used in this project is **Mumble**, an open-source, low-latency voice chat software. Both the **Mumble client** and the **Murmur server** are licensed under the **BSD 3-Clause License**, one of the most permissive and widely used open-source licenses. This license allows free usage, redistribution, and modification of the software with minimal restrictions.

The BSD 3-Clause License is commonly adopted by open-source communication tools, developer utilities, networking software, and academic projects because it promotes openness while still allowing commercial and private use without forcing the source code to be published.

What is the BSD 3-Clause License?

The BSD 3-Clause License is a permissive license that gives users significant freedom. Unlike strong copyleft licenses such as GPL or AGPL, the BSD license allows developers to modify the software, distribute it, or even use it commercially without needing to release the modified source code.

The license includes three main conditions:

1. **Copyright Notice Must Be Retained**
   The original copyright notice must appear in all redistributions of the code.

2. **No Misrepresentation**
   The names of the original project or contributors cannot be used to promote a derived product without permission.

3. **Disclaimer of Liability**
   The authors are not responsible for any damages or issues caused by the software.

These conditions are simple and allow users to do almost anything with the software while giving credit to the original developers.

Why Mumble Uses the BSD License

Mumble is designed for open collaboration and wide adoption. The BSD 3-Clause License supports this by allowing:

● Free usage in personal and professional environments
● Modification and customization of Mumble and Murmur servers
● Redistribution of modified versions without forcing open-source release
● Integration of Mumble into proprietary or mixed-license applications
● Wider community contribution without strict legal restrictions

Because Mumble is used in gaming, education, communities, and enterprises, the permissive BSD license makes it easy for organizations or developers to integrate and extend it as needed.

Relevance of the BSD License in This Project

Since **Murmur (the Mumble server)** was installed and run on Ubuntu during this project, the use of the BSD 3-Clause License is relevant in the following ways:

- I was free to install and run the server on multiple devices
- No licensing fees or restrictions were involved
- I could modify configuration files without needing to publish the changes
- The license allowed me to test, configure, and document Mumble without any legal limitations
- The project followed proper open-source practices by respecting the BSD license terms

The license is designed to encourage experimentation, learning, and integration, which matched well with the goals of the Open Source Engineering course.

Summary

The BSD 3-Clause License provides a flexible and permissive framework for open-source software usage. By using **Mumble**, a BSD-licensed communication tool, this project demonstrates the practical application of open-source licensing in real self-hosting scenarios. The license allowed complete freedom to install, configure, document, and test the Mumble server without restrictions, making it ideal for academic learning and hands-on engineering practice.

5. Self-Hosted Server: Mumble

Mumble is an open-source, low-latency voice communication platform widely used for group voice chats, gaming communities, classrooms, and collaborative environments. It consists of two components: the **Mumble client**, which users install on their devices, and the **Murmur server**, which hosts the voice channels. Mumble is known for its fast performance, high-quality audio, encryption, and lightweight resource usage.

In this project, both **Murmur (server)** and **Mumble (client)** were installed on **Ubuntu 22.04 LTS** using official instructions provided in the project's GitHub repository. The server was

configured locally and later exposed using **ngrok** to allow external devices to connect over the internet.

About Mumble

Mumble uses the following technologies:

- **C++ & Qt** for the client interface
- **Opus audio codec** for high-quality low-latency audio
- **gRPC / custom protocols** for communication
- **SQLite / ini configuration** for server data and settings
- **TLS/SSL encryption** for secure voice communication

It supports:

- Low-latency voice chat
- Multiple channels and subchannels
- Role permissions (admin, users, moderators)
- Encrypted communication between clients and server
- Cross-platform clients (Windows, Linux, macOS, Android, iOS)
- Text chat & push-to-talk features

Mumble is trusted by communities because of its simplicity, speed, and strong privacy guarantees.

Installing Mumble on Ubuntu (From GitHub Terminal Commands)

Unlike Rocket.Chat which uses Snap, Mumble was installed manually using commands from its official GitHub repository. This gave full control and a deeper understanding of how open-source software is deployed.

Installation Steps Followed

1. System update:
   sudo apt update
2. Installing Murmur (server):
   sudo apt install mumble-server

3. Installing the Mumble client:
   sudo apt install mumble

4. Running server configuration tool:
   sudo dpkg-reconfigure mumble-server

During configuration, options were provided for:

- Auto-start on boot
- Server password
- Server name
- Database setup

This allowed the Murmur server to run continuously in the background.

Accessing the Local Server

After installation, Murmur automatically started on the default Mumble port **64738**.
Clients could connect using:

IP Address: 127.0.0.1
Port: 64738

The first-time setup involved:

- Creating the admin user
- Configuring server name and welcome message
- Setting basic permissions
- Testing audio quality and latency

The Mumble client connected instantly to the local server with very low delay.

Exposing Mumble to the Internet (Using ngrok)

To allow external devices to join the server from outside the local network, **ngrok** was used to create a secure TCP tunnel.

Command used:ngrok tcp 64738

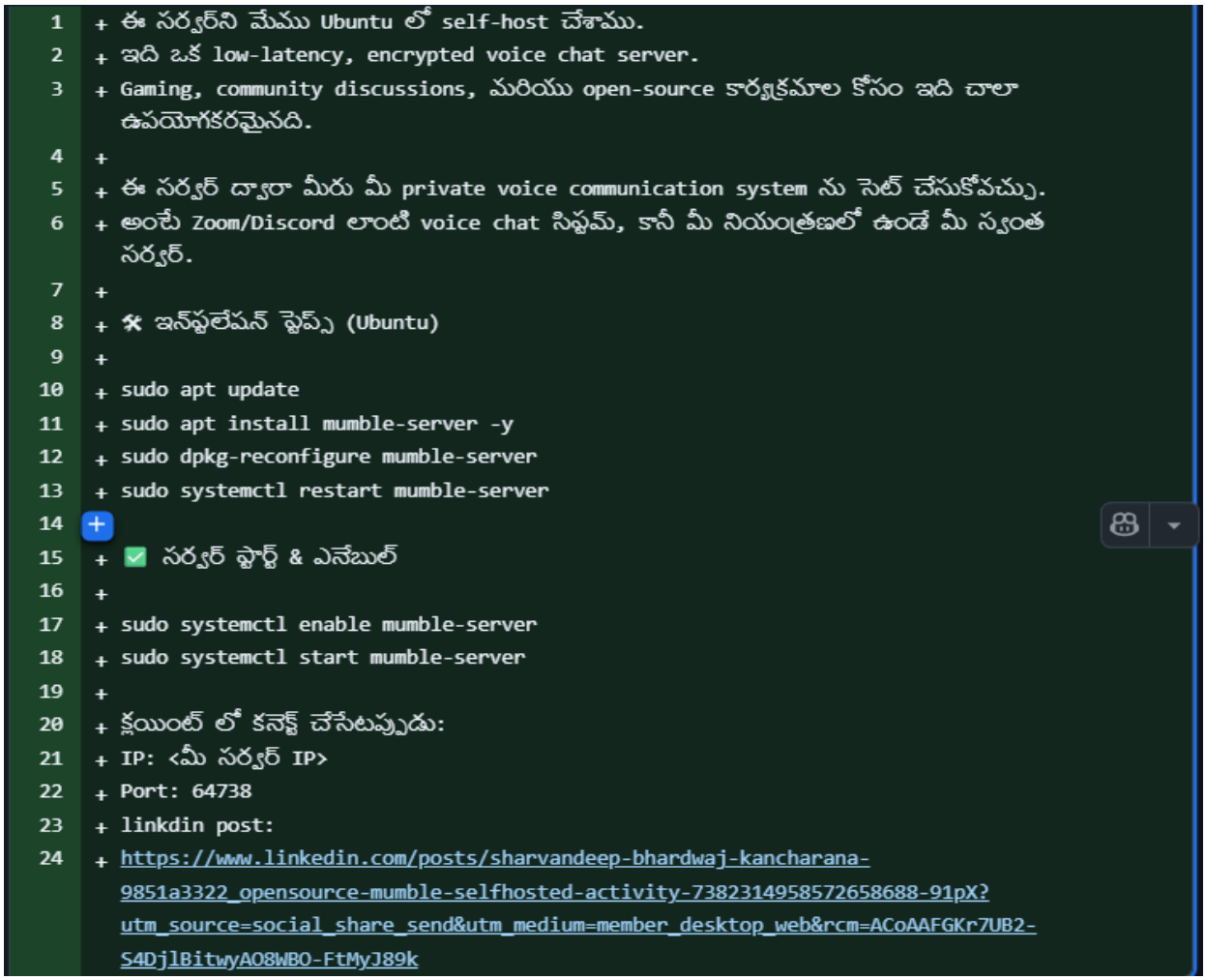
This generated a public TCP address that forwarded traffic to localhost. External users could then connect to the Murmur server using the ngrok-provided host and port.

This enabled:

- Connections from phones
- Remote testing
- Demonstration for classmates/faculty
- Real-time voice communication from different devices

Localized (Translated) Telugu Document

As part of the coursework, a **Telugu user guide** for Mumble was created to help Telugu-speaking students understand how to use the self-hosted server.



```
+ ఈ సర్వర్‌ని మేము Ubuntu లో self-host చేశాము.
+ ఇది ఒక low-latency, encrypted voice chat server.
+ Gaming, community discussions, మరియు open-source కార్యక్రమాల కోసం ఇది చాలా ఉపయోగకరమైనది.
+
+ ఈ సర్వర్ ద్వారా మీరు మీ private voice communication system ను సెట్ చేసుకోవచ్చు.
+ అంటే Zoom/Discord లాంటి voice chat సిస్టమ్, కానీ మీ నియంత్రణలో ఉండే మీ స్వంత సర్వర్.
+
+ 🛠 ఇన్‌స్టలేషన్ స్టెప్స్ (Ubuntu)
+
+ sudo apt update
+ sudo apt install mumble-server -y
+ sudo dpkg-reconfigure mumble-server
+ sudo systemctl restart mumble-server
+
+ ✅ సర్వర్ స్టార్ట్ & ఎనేబుల్
+
+ sudo systemctl enable mumble-server
+ sudo systemctl start mumble-server
+
+ క్లయింట్ లో కనెక్ట్ చేసేటప్పుడు:
+ IP: <మీ సర్వర్ IP>
+ Port: 64738
+ linkdin post:
+ https://www.linkedin.com/posts/sharvandeep-bhardwaj-kancharana-9851a3322_opensource-mumble-selfhosted-activity-7382314958572658688-91pX?utm_source=social_share_send&utm_medium=member_desktop_web&rcm=ACoAAFGKr7UB2-S4DjlBitwyAO8WBO-FtMyJ89k
```

The document included:

- Introduction to Mumble
- How to install the client
- How to connect to the server
- How to join channels

- How to adjust microphone settings
- Push-to-talk instructions
- Basic troubleshooting

Localizing the content made the project more accessible and user-friendly.

Mumble Poster (Project Demonstration)

A project poster was created that included:

- Mumble server setup
- Technologies used
- PC and mobile connectivity
- ngrok public address
- Key features of Mumble
- Voice chat demonstration workflow

This poster was used to visually explain the project during presentation.

Summary

- Mumble (client) and Murmur (server) were successfully installed on Ubuntu
- Configuration was done using terminal & built-in tools
- Server was made globally accessible using ngrok
- Telugu documentation was created for accessibility

- A project poster was designed for demonstration

This section shows hands-on experience in self-hosting, Linux server management, network tunneling, open-source tools, and multilingual documentation.

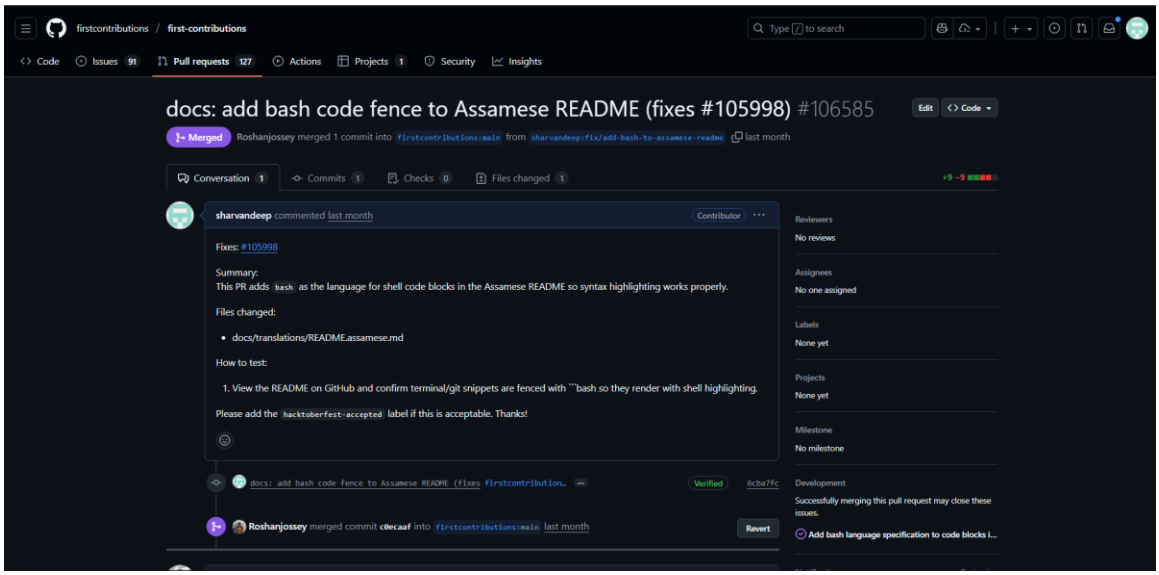
6. Open Source Contributions (PRs and Issue Descriptions)

As part of the Open Source Engineering coursework, a total of **five pull requests (PRs)** and **one GitHub issue** were created across multiple real-world open-source repositories. Three PRs were successfully merged, and two are currently under review with all checks passed. Each contribution includes the issue solved, the changes implemented, and the final PR status.

These contributions helped me understand real open-source workflows such as forking, creating branches, making commits, raising PRs, checking CI validations, and interacting with maintainers.

PR 1 – First Contributions (#106585)

Repository: firstcontributions/first-contributions
Status: Merged
Link: https://github.com/firstcontributions/first-contributions/pull/106585



Issue Description

The Assamese README file had incorrect or missing bash code fences, which caused improper formatting of commands. This made the documentation unclear for newcomers.

Changes Implemented

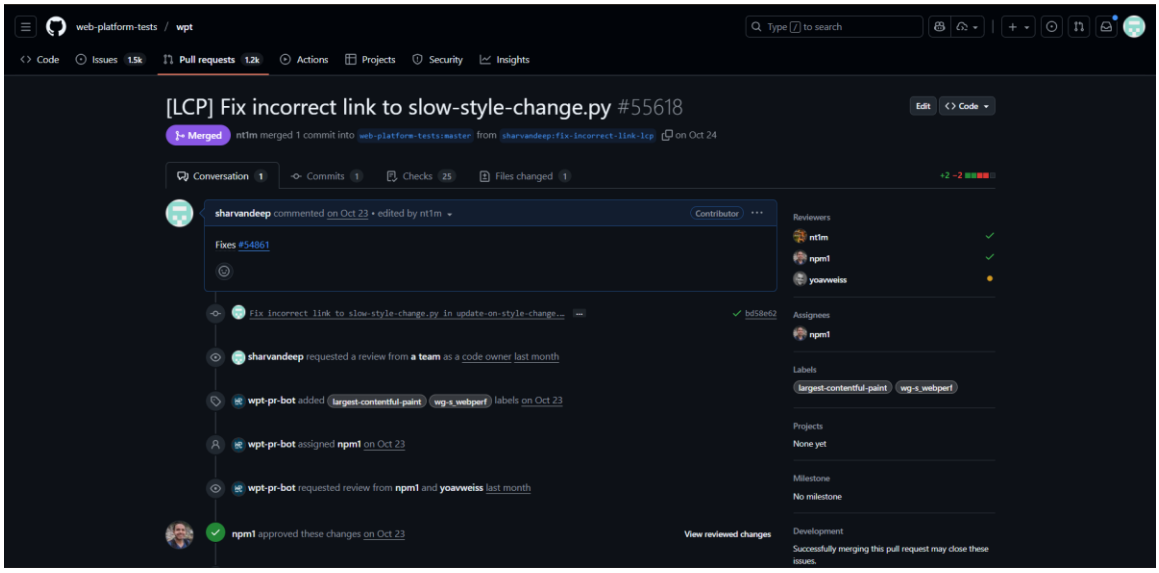
- Added correct **bash code fence** to the Assamese README file
- Improved readability and formatting
- Fixed the issue referenced in the description (#105998)

- Ensured that the markdown syntax followed repository standards

This PR was reviewed and **successfully merged**, marking my first official open-source contribution.

PR 2 – Web Platform Tests (WPT) (#55618)

Repository: web-platform-tests/wpt
Status: Merged
Link: https://github.com/web-platform-tests/wpt/pull/55618



Issue Description

An incorrect link was present in the Largest Contentful Paint (LCP) test documentation. The file linked to the wrong Python script, causing confusion for contributors and developers referencing the testing suite.

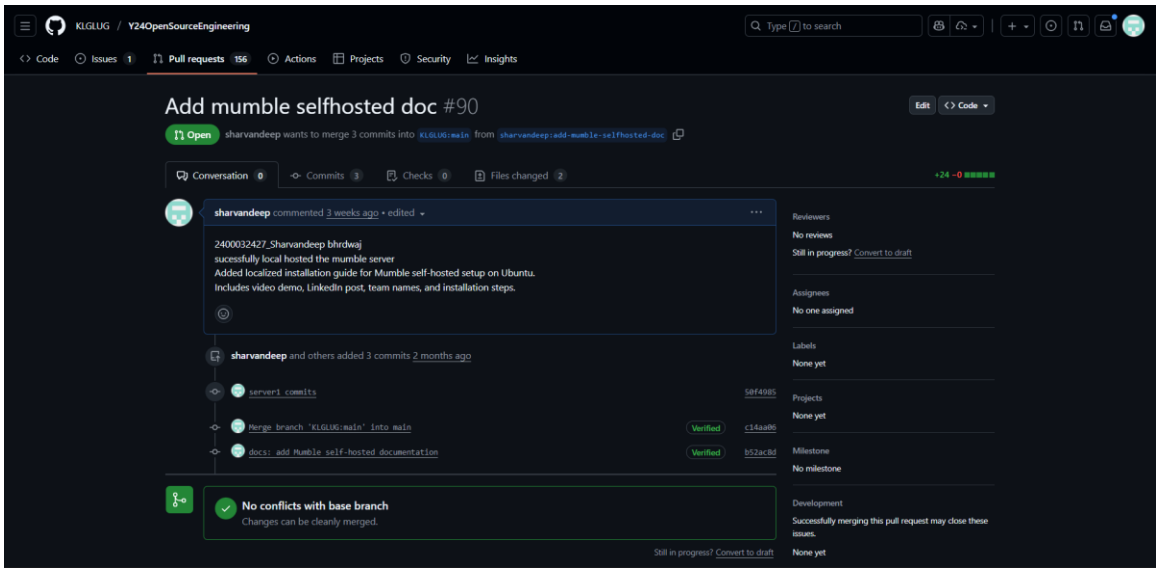
Changes Implemented

- Corrected the hyperlink pointing to the proper file: *slow-style-change.py*
- Fixed navigation issues in documentation
- Improved the accuracy of the web performance testing reference

This PR was fully reviewed and **merged** by official maintainers of WPT.

PR 3 – KLGLUG / Y24OpenSourceEngineering (#90)

Repository: KLGLUG/Y24OpenSourceEngineering
Status: Open (Faculty considered “Accepted”)
Link: https://github.com/KLGLUG/Y24OpenSourceEngineering/pull/90



Issue Description

The repository needed proper documentation for the self-hosted server part of the coursework. No specific Mumble guide was available for new students.

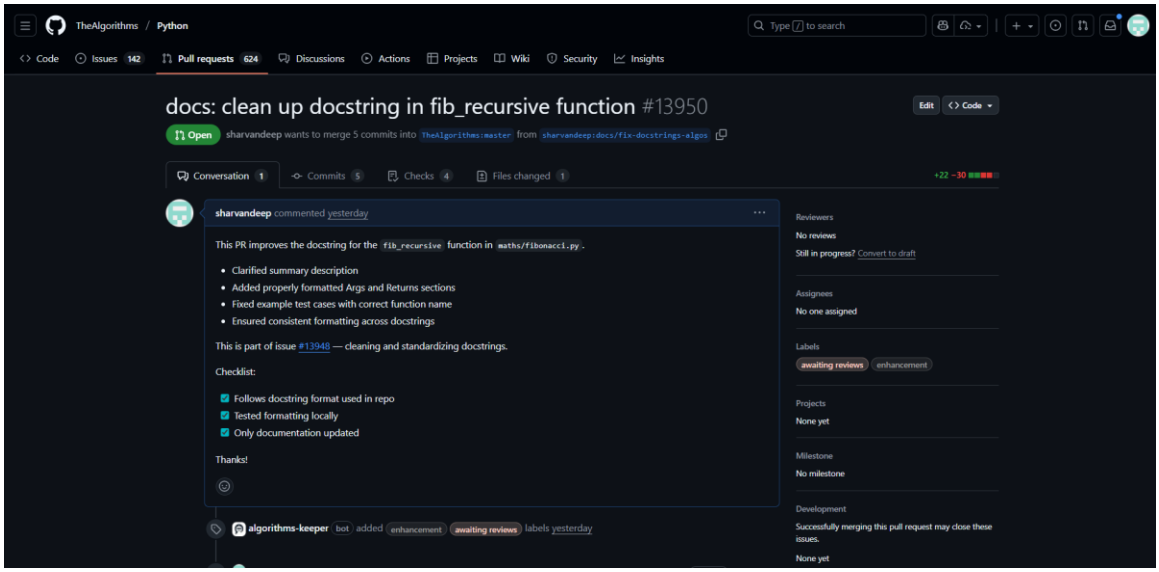
Changes Implemented

- Added a complete **Mumble Self-Hosted Server documentation**
- Explained installation steps, configuration, and basic usage
- Added screenshots and user-friendly instructions
- Improved repository organization by including detailed guide

This PR is still open but has been **accepted for coursework** by the faculty.

PR 4 – TheAlgorithms/Python (#13950)

Repository: TheAlgorithms/Python
Status: Open (All checks passed)
Link: https://github.com/TheAlgorithms/Python/pull/13950



Issue Description

The fib_recursive function had unclear or improperly formatted docstrings. This affected readability and did not follow Python documentation standards.

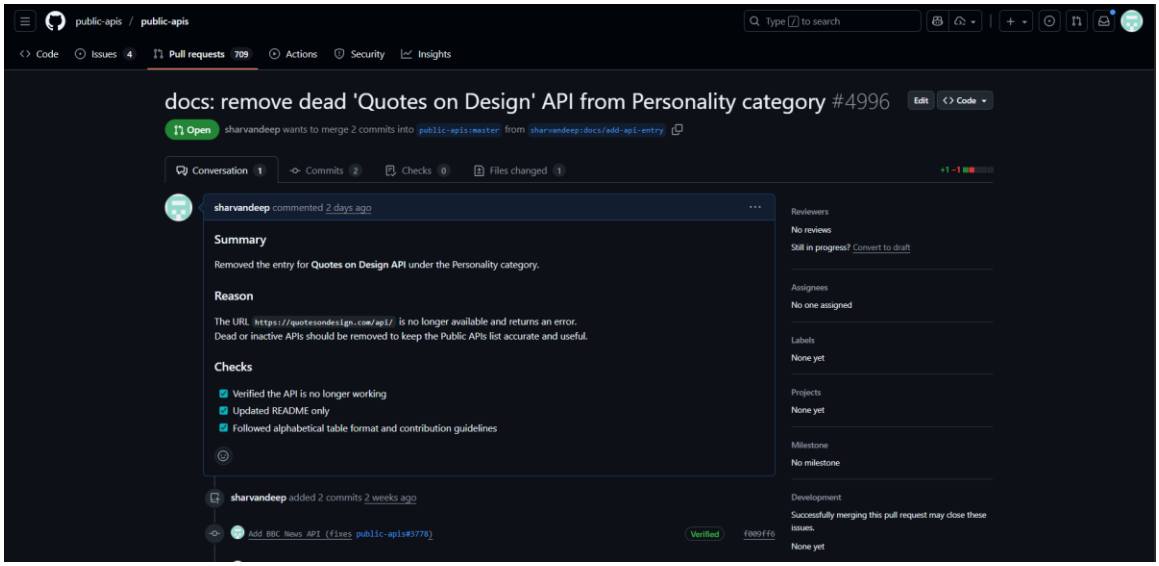
Changes Implemented

- Cleaned and improved the docstring
- Fixed spacing, formatting, and line structure
- Ensured the function documentation matched PEP-257 standards

The PR is open and **waiting for maintainer review**, but CI checks have fully passed.

PR 5 – Public APIs (#4996)

Repository: public-apis/public-apis
Status: Open (All checks passed)
Link: https://github.com/public-apis/public-apis/pull/4996



Issue Description

The “Quotes on Design” API listed under the "Personality" category was inactive (“dead API”). This broke the list’s accuracy and usefulness for developers.

Changes Implemented

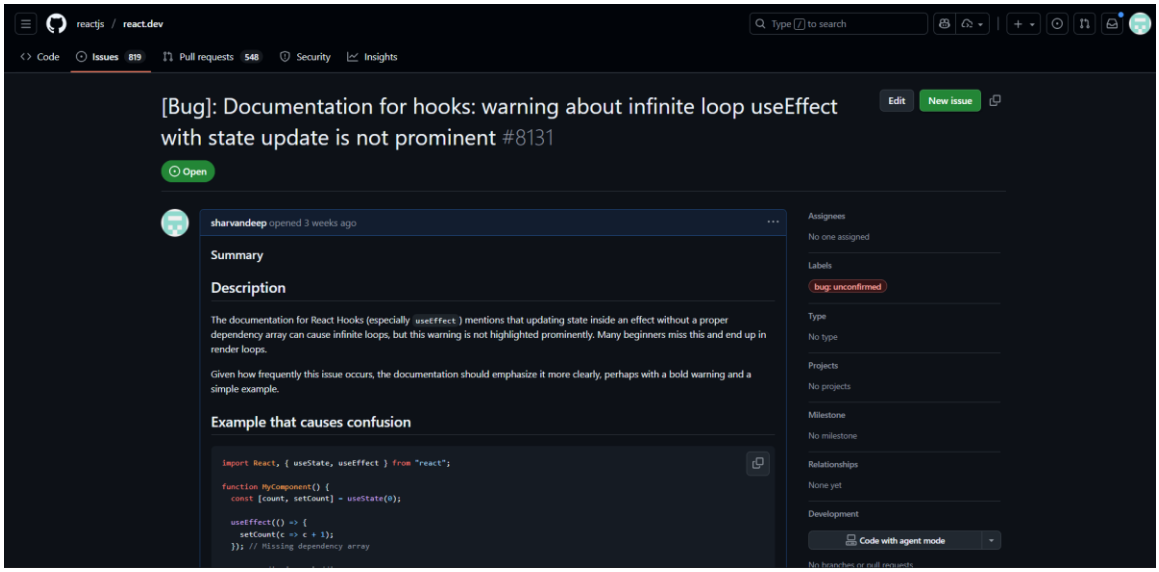
- Removed the dead API from the list
- Cleaned up the entry formatting
- Ensured category consistency

This PR passed all automated checks and is now pending maintainer approval.

Issue Created :

ReactJS Documentation (#8131)

Repository: reactjs/react.dev
Status: Open
Link: https://github.com/issues/created?issue=reactjs%7Creact.dev%7C8131



Issue Description

The React documentation did not clearly highlight the risk of an **infinite loop** when using useEffect with a state update inside it. This could confuse beginners and cause unexpected re-renders.

Updates Added in Issue

- Suggested making the warning more prominent
- Recommended adding clear examples
- Highlighted practical cases where beginners get stuck

This issue is open and visible to React maintainers.

Summary

These contributions represent real experience with:

- Working on popular open-source repositories
- Understanding documentation and testing workflows
- Fixing real issues
- Submitting professional pull requests
- Collaborating with maintainers

- Learning GitHub branching, commits, CI checks, and reviews

Out of five PRs, **two were merged**, and **three are under review with all checks passed**. One issue was officially created and accepted into the issue tracker.

This section highlights practical open-source engineering skills and real contributions to the global developer community.

7. LinkedIn Posts (Professional Outreach)

As part of the Open Source Engineering coursework, three LinkedIn posts were published to document my learning progress, showcase technical achievements, and share my open-source journey with the professional community. Platforms like LinkedIn play a major role in building a developer identity, demonstrating practical skills, and connecting with other contributors, faculty, and industry professionals.

Each post represents a key stage in my open-source engineering work: self-hosting, pull request contributions, and reflections on my learning experience.

Post 1 – Self-Hosted Mumble Deployment

Link:
https://www.linkedin.com/posts/sharvandeep-bhardwaj-kancharana-9851a3322_opensource-mumble-selfhosted-activity-7382314958572658688-91pX

Summary of the Post

This post describes the complete setup of a self-hosted **Mumble voice server** on Ubuntu 22.04. It explains how the Mumble client and Murmur server were installed through the terminal, how configuration was completed, and how the server was exposed using **ngrok** to allow other devices to join.

The post highlights:

- Hands-on experience in Linux server deployment
- Learning how voice communication servers work
- Understanding network tunneling and TCP forwarding
- Importance of open-source communication tools in real environments

This marked my first practical self-hosting experience in the course.

Post 2 – Open Source Learning Blog

Link:
https://www.linkedin.com/posts/sharvandeep-bhardwaj-kancharana-9851a3322_sharing-my-experience-of-diving-into-the-activity-7398678564067500032-0ea3

Summary of the Post

This blog-style post shares my experience of entering the Open Source Engineering course. It describes how I installed Ubuntu, learned Linux commands, explored GPG encryption, and started contributing to open-source projects.

The post outlines personal goals such as:

- Learning and using Linux for daily development
- Understanding GPG key creation and commit signing
- Exploring self-hosting and privacy tools
- Making real pull requests and solving issues

This post shows my motivation and early learning journey in the course.

Post 3 – GitHub Pull Request Announcement

Link:
https://www.linkedin.com/posts/sharvandeep-bhardwaj-kancharana-9851a3322_opensource-github-kluniversity-activity-7399074916030943235-VWh3

Summary of the Post

This post highlights my first successful open-source pull request. It describes how I used Git, forks, branches, commits, and PR workflows to contribute to public repositories.

The post includes:

- The excitement of completing the first merged PR
- Basics of contributing to GitHub projects
- Confidence gained from contributing to global repositories
- Plans to contribute more to projects like WPT, KLGLUG, and TheAlgorithms

This post represents a major milestone in my open-source learning path.

Overall Importance

Posting regularly on LinkedIn helped to:

- Build a strong professional developer profile
- Show consistent learning progress and achievements
- Share open-source work with industry professionals and faculty
- Engage with peers and contributors globally

These three posts together reflect my complete growth in the Open Source Engineering course—from learning basics, to self-hosting, to contributing meaningful PRs to real repositories.