KL University

Open Source Engineering Project Report

Project Report

Submitted in partial fulfilment of the requirements for

Open Source Engineering

Submitted by:

KANCHARANA 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: _____

Acknowledgment

I would like to express my sincere gratitude to KL University and the Department of CSE 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.

I am especially grateful to **Dr. Sripath Roy Koganti**, our course faculty, for his continuous guidance and support throughout the project.

I also thank the KLGLUG community and all open-source maintainers for their documentation and reviews.

My heartfelt thanks to my classmates and friends for their cooperation and encouragement.

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.

1.1 System Architecture and Features

Ubuntu 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

1.2 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 installed required packages.

1.3 Commands Used Frequently

Most work in this project involved the terminal. Commands used:

```
ls
cd
pwd
sudo apt update
sudo apt upgrade
sudo apt install <package>
nano
git add .
git commit -m "message"
git push
git status
```

1.4 Software Installation

Using APT, I installed:

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

1.5 Desktop Environment

The GNOME desktop provided a clean and minimal design, smooth animations, easy access to settings, and good integration with development tools.

1.6 Why Ubuntu Was Chosen

I chose Ubuntu because it is stable, beginner-friendly, recommended by my professor, works well with GitHub, supports GPG, and suits self-hosting needs.

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 22.04 includes the GnuPG tool by default, so generating and managing keys through the terminal was simple and efficient.

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

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

These are the commands used during the project:

```
gpg --list-keys
gpg --list-secret-keys
gpg --full-generate-key
gpg --fingerprint
```

These commands helped verify the key details, fingerprints, and confirm that the keypair was generated correctly.

Signing Git Commits

A major use of GPG in this project was to sign Git commits. Signed commits increase trust and ensure the identity of the author on GitHub. After generating the GPG key, Git was configured using:

```
git config --global user.signingkey 75A43C246E1F30C0
git config --global commit.gpgsign true
```

With this configuration, every commit pushed to GitHub was automatically signed using the new GPG key, resulting in a "Verified" badge on GitHub.

Importance of GPG in This Project

GPG was useful in multiple ways:

- Verified commit authorship on GitHub.
- Helped create secure and trustworthy contributions.
- Introduced real–world workflows used in open–source communities.
- Improved understanding of cryptographic principles such as keys and signatures.

Even though advanced GPG features such as encrypted file sharing and email encryption were not required, learning to generate keys and sign commits was an important part of secure development practices.

Overall, using GPG in this project strengthened my understanding of secure workflows, cryptographic identity, and practical open–source engineering.

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 popular community website that recommends open–source alternatives to proprietary and data–tracking applications. These tools help protect user identity, prevent third–party tracking, secure communication, and create a more private online 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 part of privacy such as secure browsing, messaging, password protection, and anonymous networking.

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. Firefox blocks third–party trackers, fingerprinting attempts, and intrusive ads through Enhanced Tracking Protection. It also supports privacy–focused extensions such as uBlock Origin, HTTPS Everywhere, and Privacy Badger.

I used Firefox throughout this project to browse documentation, access GitHub repositories, and download Linux tools. Since Firefox is open–source, its code is publicly available for auditing, making it trustworthy for privacy–minded users.

2. Signal Messenger (Secure Messaging)

Signal is one of the most secure messaging applications available today. It provides end–to–end encryption for chats, voice calls, video calls, and file transfers using the Signal Protocol. Signal does not store chat history, metadata, or backups on servers, and does not track user activity.

Signal is commonly used by developers, journalists, and researchers who require private communication. Exploring Signal helped me understand the importance of encrypted, secure communication.

3. Tor Browser (Anonymous Browsing)

Tor Browser provides anonymous browsing by routing internet traffic through a decentralized network of volunteer–operated nodes. This technique, known as onion routing, hides the user's IP address and identity.

Tor Browser is useful for:

- Avoiding tracking and fingerprinting
- Bypassing internet censorship
- Accessing blocked websites
- Protecting privacy on public networks

Although slower than normal browsers due to multiple layers of encryption, Tor offers the highest level of online anonymity available.

4. KeePassXC (Password Manager)

KeePassXC is an open–source, offline password manager that stores all credentials locally in an encrypted database. It uses AES-256 encryption to secure passwords and provides features such as password generation, secure notes, and SSH key storage.

Since KeePassXC does not sync anything to the cloud by default, users maintain full control over their data. I explored KeePassXC to understand the importance of secure password storage and using unique passwords for different accounts.

5. Mullvad VPN (Privacy-Focused VPN)

Mullvad is a privacy–first VPN that follows a strict no-logs policy. It does not require personal information for creating an account and supports modern VPN protocols such as WireGuard and OpenVPN.

Mullvad:

- Hides the user's IP address
- Encrypts internet traffic
- Prevents ISP and website tracking
- Does not store activity logs

Exploring Mullvad helped me understand how VPNs protect data from network-level surveillance.

Importance of Privacy Tools

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

- Protect personal information
- Reduce online tracking
- Improve security on public networks
- Prevent unauthorized data collection
- Support safe browsing and communication

For open–source engineering, privacy tools ensure that development workflows, online research, and communication remain secure. These five tools — Firefox, Signal, Tor Browser, KeePassXC, and Mullvad VPN — helped me understand how privacy can be maintained using free and open–source software.

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 communication software. Both the Mumble client and the Murmur server operate under the BSD 3-Clause License, one of the most permissive and developer-friendly licenses in the open-source ecosystem. This license allows anyone to freely use, modify, and redistribute the software with very few restrictions.

The BSD 3-Clause License is commonly used in academic research projects, networking software, communication tools, and other open-source applications because it supports wide adoption without enforcing strong copyleft requirements. This makes it suitable for both open-source developers and commercial organizations that want flexibility.

What is the BSD 3-Clause License?

The BSD 3-Clause License is a permissive open-source license that provides users with maximum freedom. Unlike copyleft licenses such as GPL or AGPL, the BSD license does not force developers to release the source code of their modified versions.

The license contains three main conditions:

- **Copyright Notice Retention:** The original copyright and license notice must be included in all redistributions of the code.
- **No Endorsement Clause:** The names of the original authors or contributors cannot be used to endorse derived products without permission.
- **Disclaimer of Liability:** The authors are not responsible for damage or legal issues caused by the software.

These simple conditions make the BSD license easy to adopt and understand.

Why Mumble Uses the BSD License

Mumble is designed for broad community usage, ranging from gaming communities to enterprise communication systems. The BSD 3-Clause License encourages such adoption because it allows:

- Free use in both personal and commercial environments
- Integration into proprietary software
- Redistribution of modified versions
- Community contributions without legal complexity

This permissive licensing model aligns well with Mumble's goal of being widely used and easily extendable.

Relevance of the BSD License in This Project

Since Mumble was installed and configured as part of this Open Source Engineering project, the BSD 3-Clause License affects how the software can be used:

- I was free to install and run the server without restrictions
- No publishing of modified configuration files was required
- I could experiment with the software for learning purposes
- There were no legal limitations on documenting or demonstrating the setup

This made Mumble ideal for hands-on self-hosting tasks and educational use.

Summary

The BSD 3-Clause License provides a flexible, permissive structure for open-source software usage. By using Mumble, a BSD-licensed application, this project demonstrates how open-source tools can be deployed, configured, and customized without legal restrictions. The license promotes experimentation, learning, and development, making it perfectly suited for academic and practical engineering work.

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.

```
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.

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

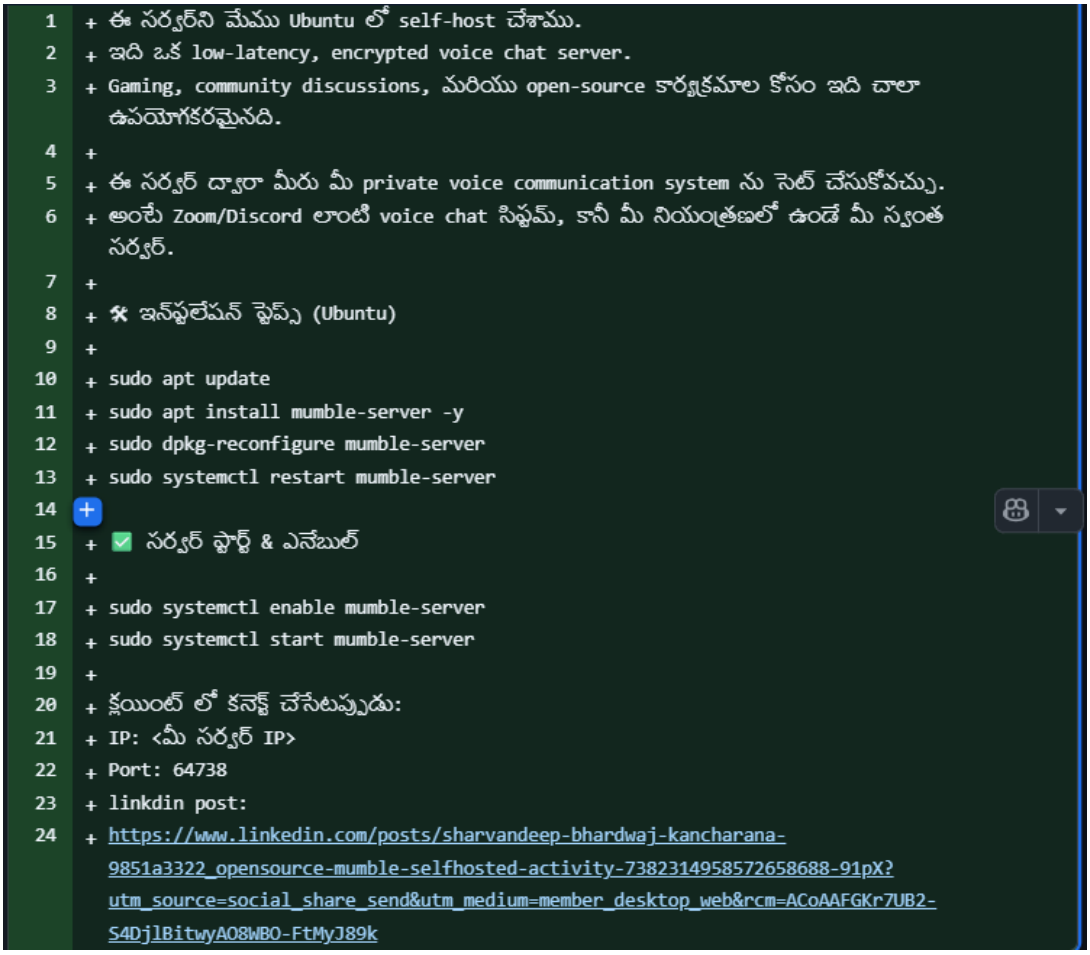

Figure 5.1: Screenshot / excerpt from the Telugu user guide

- 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.

Figure 5.2: Mumble project poster used during demonstration

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`

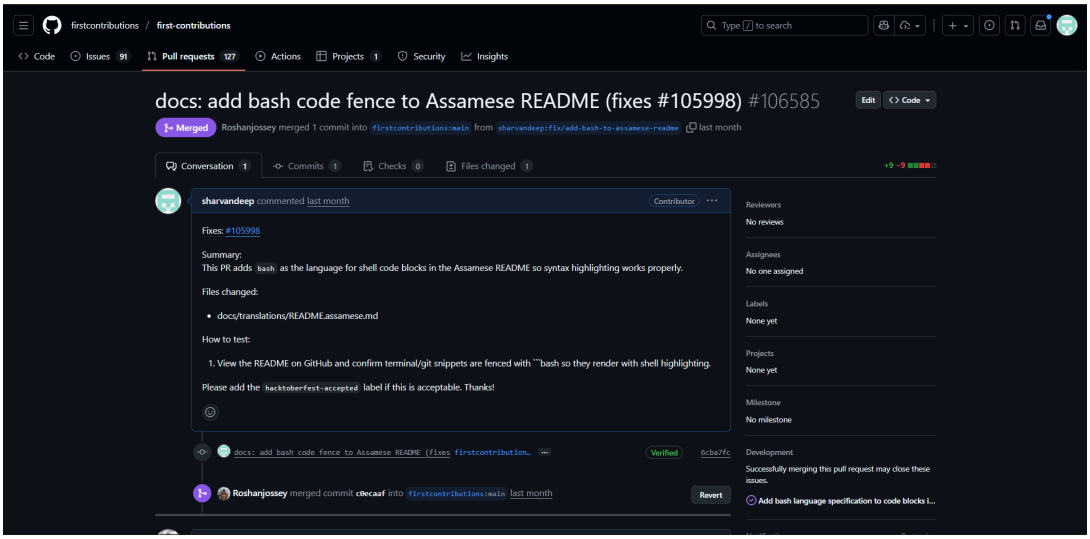

Figure 6.1: PR 1 — Screenshot / preview of the pull request on GitHub

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`

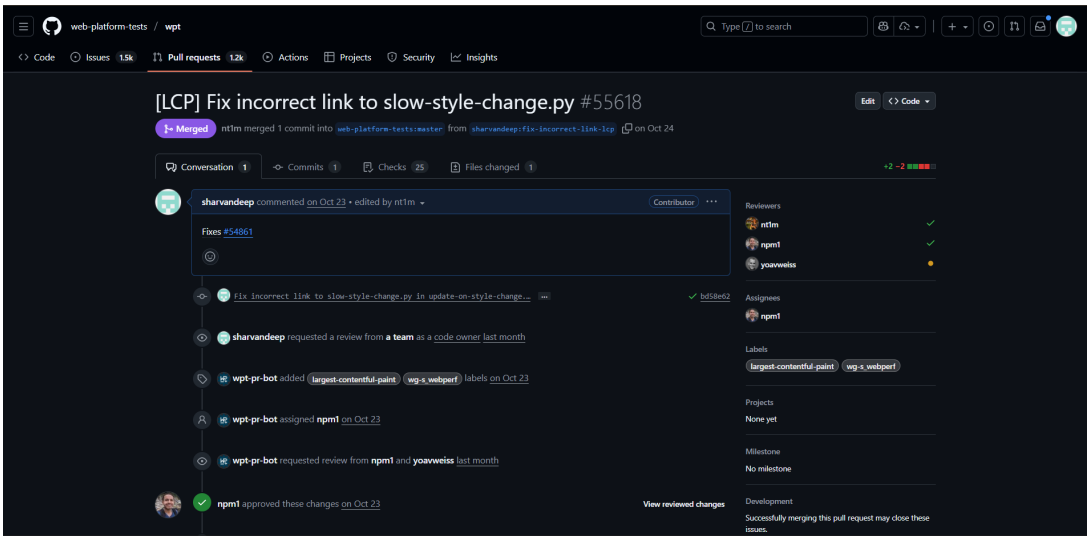

Figure 6.2: PR 2 — Screenshot / preview of the WPT pull request

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`

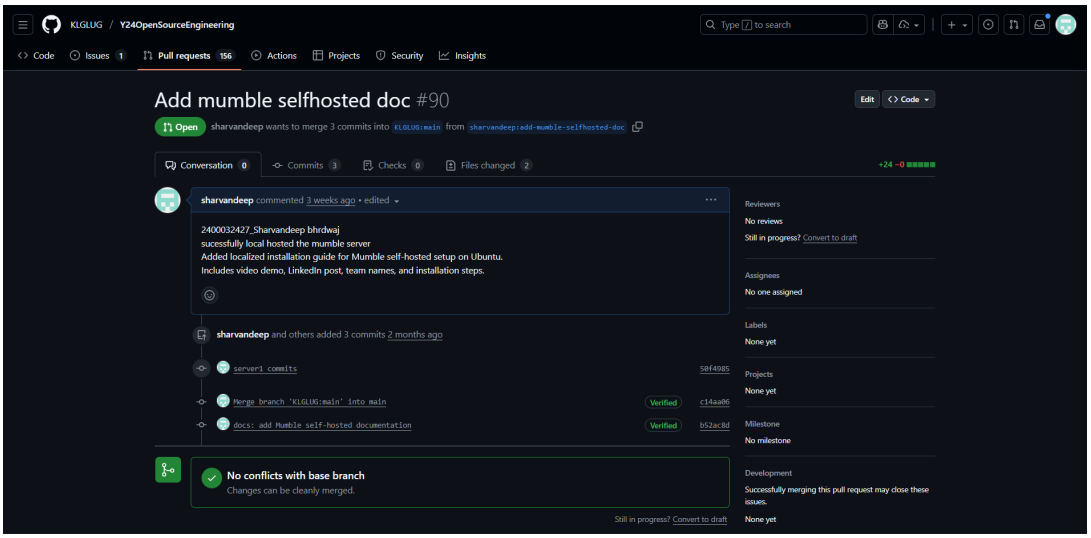

Figure 6.3: PR 3 — Screenshot / preview of the KLGLUG PR

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 a 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.

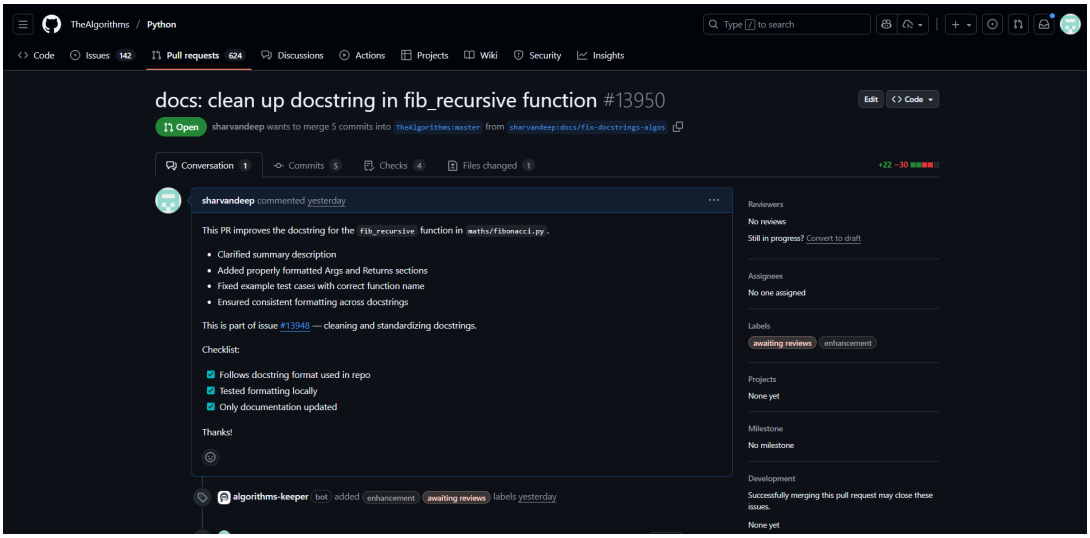

Figure 6.4: PR 4 — Screenshot / preview of the TheAlgorithms PR

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`

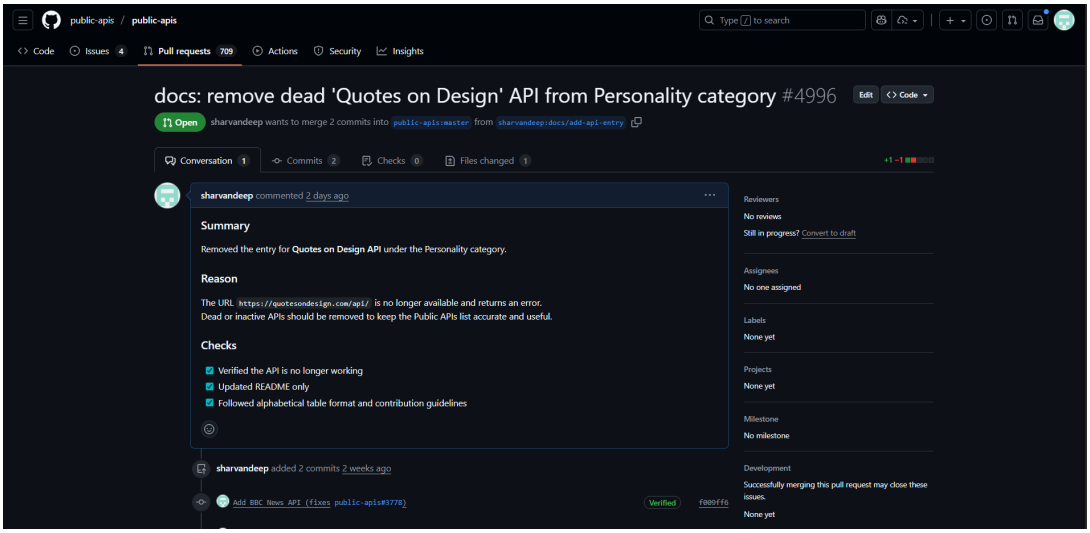

Figure 6.5: PR 5 — Screenshot / preview of the Public APIs PR

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`

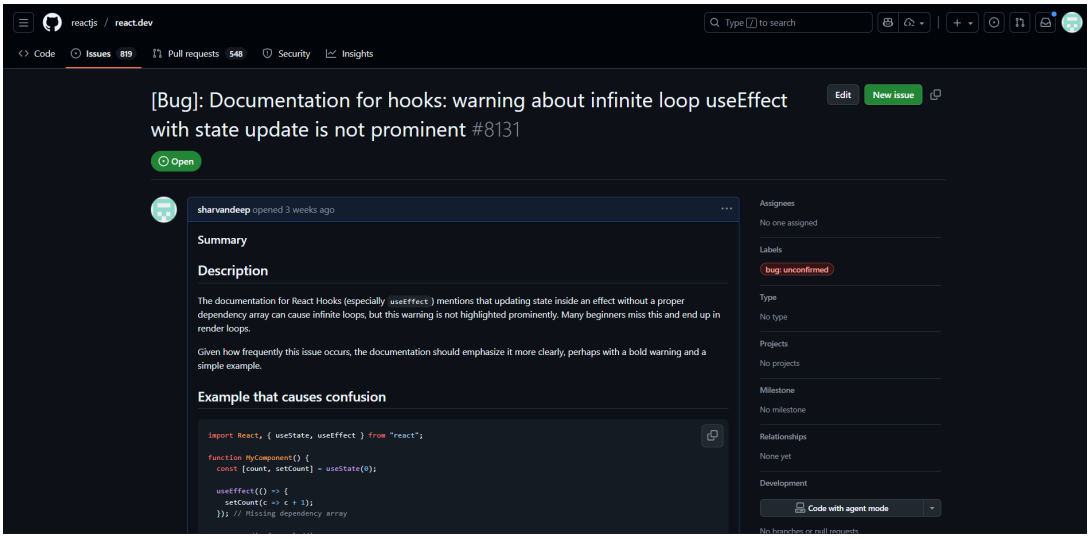

Figure 6.6: Issue — Screenshot / preview of the created ReactJS issue

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 and improving project accuracy.
- Submitting professional pull requests and following CI checks.
- Collaborating with maintainers and the open-source community.
- Learning GitHub branching, commits, CI checks, and review interactions.

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-mu`

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-ex`

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-gi`

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.