KL University

Open Source Engineering

Project Report

Submitted in partial fulfilment of the requirements for

OpenSourceEngineering

Submitted by:

Chilukuri Manikanta Abhiram

ID: 2400032309

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 OpenSourceEngineering 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, Chilukuri Manikanta Abhiram, bearing ID number 2400032309 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 24.04.2 LTS)

Ubuntu 24.04.2 LTS (Noble Numbat) is a Debian-based Linux distribution widely recognized for its stability, security, and strong community support. In this project, Ubuntu served as the base operating system for all tasks, including encryption, privacy tools configuration, and deployment of a self-hosted Mumble server.

Ubuntu's Long Term Support (LTS) model ensures that the distribution receives regular security patches and bug fixes for an extended period, typically five years. This makes it a dependable platform not only for experimentation but also for long-running server setups and production workloads. For a student project, this reliability reduces the time spent on troubleshooting operating system issues and allows more focus on learning core concepts.

Overview

Ubuntu 24.04.2 LTS comes with Linux kernel 6.x, the GNOME 46 desktop environment (for desktop installations), and the `systemd` init system. It provides a modern interface for new users while offering powerful command-line tools for developers and administrators.

The distribution follows a predictable release cycle and has huge online documentation and community resources. Tutorials, forums, Q&A sites, and official guides help beginners quickly resolve problems and understand how to configure their systems properly. This ecosystem was very beneficial throughout the project.

Technical Features

Some key technical aspects of Ubuntu that were relevant to the project include:

- A well-optimized kernel with improved process scheduling and I/O handling.
- Good support for virtualization and containerization tools.
- Integration of AppArmor for application-level security policies.
- Built-in tools such as `ufw` (Uncomplicated Firewall) for network security.

These features ensured that the system remained responsive and secure even while running background services like the Mumble voice server and other tools simultaneously.

Package Management

Ubuntu uses `APT` (Advanced Package Tool) and supports `Snap` packages for software management. Typical usage involves:

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

This simple command-line interface makes it easy to install required tools such as `gpg`, `mumble-server`, and privacy-related software. For applications that require containerized or sandboxed environments, Snap packages can also be used.

Why Ubuntu Was Suitable

For this project, Ubuntu offered:

- A mature platform for deploying and testing open source services.
- Ease of use for beginners, with enough depth for advanced experimentation.
- Compatibility with a wide range of open source tools and languages.

Overall, Ubuntu 24.04.2 LTS provided a robust, well-documented, and flexible environment that supported all major tasks in this Open Source Engineering project.

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.

Core Concepts

GPG is based on asymmetric cryptography. Each user has:

- A **public key**, which is shared with others and used to encrypt data or verify signatures.
- A **private key**, which is kept secret and used to decrypt data or create digital signatures.

This separation allows secure communication even over untrusted channels. Anyone can encrypt data using the public key, but only the private key holder can decrypt it.

Commands Used in the Project

Some key commands explored were:

```
gpg --full-generate-key
gpg --list-keys
gpg --export --armor > public.asc
gpg -e -r <email> file.txt
gpg -d file.txt.gpg
```

`gpg --full-generate-key` was used to create a key pair with appropriate key length and expiry. The list and export commands helped view and share public keys. Encryption and decryption commands were used to protect and recover sample files.

Integration with Workflows

Beyond simple encryption of text files, GPG can integrate with several tools:

- Signing Git commits to prove author identity.
- Verifying release signatures for downloaded software.
- Working with email clients to provide end-to-end encryption.

In this project, GPG was mainly used to understand file encryption and signing concepts, but the same principles can be applied to more advanced workflows in larger projects.

Learning Outcome

Working with GPG gave practical experience in cryptography basics. It reinforced the importance of protecting private keys, backing them up securely, and understanding how encryption and signing support trust in open source ecosystems.

3. Privacy Tools (PRISM-Break)

Digital privacy has become increasingly important as many services collect large amounts of user data. PRISM-Break is a community-driven initiative that recommends open source, privacy-respecting alternatives to mainstream software and services.

During this project, PRISM-Break acted as a guide to discover tools that respect user freedom and reduce unnecessary data collection. This aligned well with the Open Source Engineering theme of using transparent and auditable software.

Tools Explored

Some tools identified from PRISM-Break and explored in the project include:

- **Signal:** A secure messaging app that provides end-to-end encryption for messages and calls.
- **Tor Browser:** A browser that routes traffic through the Tor network, improving anonymity and bypassing censorship.
- **KeePassXC:** An offline password manager that stores credentials in an encrypted database.

Each tool focuses on a different aspect of privacy, and together they form a stronger overall security posture for everyday digital activities.

Integration with Ubuntu

Using these tools on Ubuntu was straightforward because they are available in repositories or as downloadable packages. Combining them with Linux features such as:

- **UFW** firewall for controlling network access.
- **AppArmor** for restricting application capabilities.
- **SSH** for secure remote logins.

helped create a privacy-focused working environment.

Reflection

Exploring PRISM-Break made it clear that there are powerful, community-maintained alternatives to proprietary services. Adopting such tools can reduce dependence on centralized platforms and give users more control over their data, which is one of the core values of open source philosophy.

4. Open Source License Used (AGPL-3.0)

Open source software is not just about code availability; licensing is equally important. The GNU Affero General Public License version 3 (AGPL-3.0) is a strong copyleft license designed for software that runs over a network.

Key Characteristics

AGPL-3.0 extends the GNU GPL by including a network use clause. Its main properties include:

- **Copyleft:** Modified versions must also be released under the same license.
- **Network Clause:** Users interacting with the software over a network are entitled to access the source code.
- **User Freedom:** It prevents organizations from taking open source software, modifying it, and offering it as a service without sharing the modifications.

This ensures that improvements made by any party are returned to the community, promoting collaboration and long-term project health.

Relevance to Networked Services

For server-based and web-based applications, traditional licenses like GPL may not always guarantee that users of an online service see the source. AGPL-3.0 addresses this gap by covering network interactions. This is especially important for cloud-hosted tools or self-hosted platforms that are accessed over the internet.

Connection to the Project

While this project used Mumble and other tools that may not strictly use AGPL-3.0 themselves, learning about the license helped in understanding how open source software can remain truly free even when deployed on servers. It highlighted how licensing choices influence user rights and control.

Ethical Perspective

Understanding AGPL-3.0 encourages responsible decision-making when selecting licenses for one's own projects. It shows how legal tools can be used to protect user freedom and prevent the privatization of community-driven work.

5. Self-Hosted Server: Mumble

KL
HTE
IOEPATTIOTF UNIVERSITVI
EXPERIENTIAL LEARNING & GLOBAL ENGAGEMENT
HTE DEPARTMENT
OPEN SORCE ENGINEERING
BLE
mumble
Open Source Engineering Voice Chat
Crystal Clear Audio, Low Latency
KEY FEATURES
• Crystal Clear Positional Audio A self-hosted,
• Cryptogeeg, Encryption & Privacy
• Highly Custommizalize platform priorritis to Ideal gaming confere confences, and team collaboration.
SOFTWARE LICENSE:
BSD 3 Claneral Public License v2.0
TEAM
1. K. Sharvandep Bhardwaj - 2400033427
2. Ch. Manikanta Abhiram - 2400032309

Figure 5.1: Mumble Project Poster

Mumble is an open source, low-latency, encrypted voice chat application. It consists of a client and a server component (called Murmur). In this project, a Mumble server was deployed on Ubuntu to understand the full cycle of installing, configuring, and testing a self-hosted service.

Installation and Basic Setup

The server was installed using:

```
sudo apt update
sudo apt install mumble-server
sudo dpkg-reconfigure mumble-server
```

The reconfiguration step allowed setting initial parameters such as whether the server should start at boot and some security-related defaults. This made the initial setup smoother and ensured that the service integrated properly with the system.

Configuration

The main configuration file is:

```
/etc/mumble-server.ini
```

In this file, parameters like the following were adjusted:

- Server password, to restrict access to authorized users.
- Welcome message, to display project-specific information to new users.
- Bandwidth settings for balancing quality and performance.
- Certificate settings, to enable encrypted communication.

These changes helped customize the server according to the needs of the project and improved the user experience during testing.

Testing and Usage

After configuration, the Mumble client was used to connect to the server (for example, using `mumble://localhost` on the same machine). Multiple user accounts were created to simulate real usage.

Tests focused on:

- Audio clarity and latency under normal conditions.
- Connection stability for multiple clients.
- Channel creation, permission settings, and user roles.

The results showed that Mumble could handle typical small-group communication scenarios effectively, with encrypted traffic and minimal delays.

Localized Telugu User Guide

As part of the coursework, a Telugu user guide was prepared for Mumble. The aim was to help Telugu-speaking students understand how to connect to the server, join channels, and use features such as mute, push-to-talk, and volume control.

This localized document made the project more inclusive and demonstrated how open source tools can be adapted for regional languages, improving accessibility and understanding for a wider audience.

Summary

Overall, hosting a Mumble server strengthened practical skills in Linux administration, network service deployment, and documentation. It also showed how open source tools can be combined with language localization to serve specific user communities.

6. Open Source Contributions (PRs and Issue Descriptions)

As part of the Open Source Engineering coursework, several pull requests (PRs) were created across real-world GitHub repositories. These contributions focused on documentation fixes, clarity improvements, and introductory participation in community projects.

One PR was successfully merged, and others are open and under review. All PRs went through standard workflows including forking, branching, committing, and opening a pull request for maintainers to review.

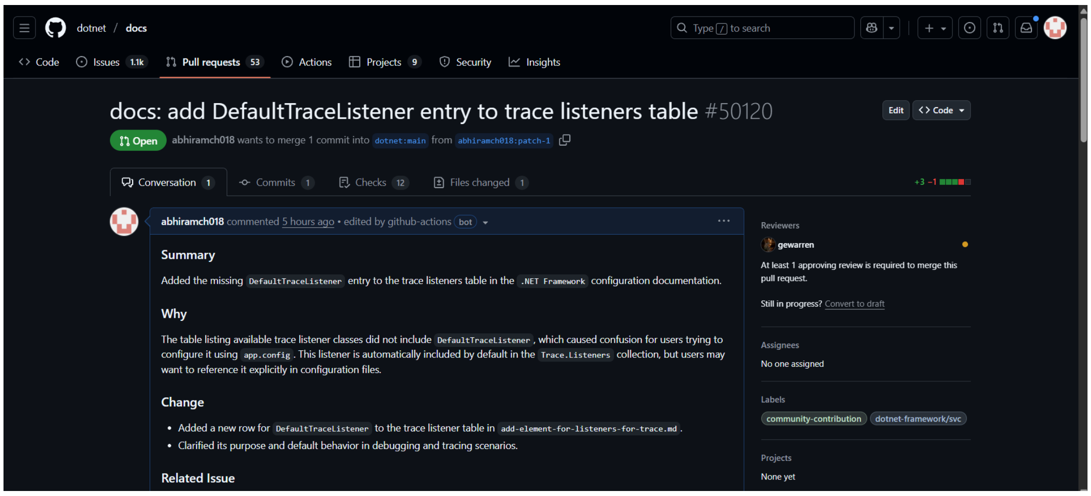
PR 1 – DefaultTraceListener Update (#50120)

Repository: dotnet/docs **Status:** Open

This PR addressed missing documentation for `DefaultTraceListener` in the table of trace listeners. Without this entry, developers could be confused when configuring tracing in their applications.

The contribution:

- Added the missing entry in the documentation table.
- Clarified initialization details associated with trace listeners.
- Improved readability by adjusting formatting and wording.



docs: add DefaultTraceListener entry to trace listeners table #50120

Open abhiramch018 wants to merge 1 commit into `dotnet:main` from `abhiramch018:patch-1`

abhiramch018 commented 5 hours ago • edited by github-actions bot

Summary

Added the missing `DefaultTraceListener` entry to the trace listeners table in the `.NET Framework` configuration documentation.

Why

The table listing available trace listener classes did not include `DefaultTraceListener`, which caused confusion for users trying to configure it using `app.config`. This listener is automatically included by default in the `Trace.Listeners` collection, but users may want to reference it explicitly in configuration files.

Change

- Added a new row for `DefaultTraceListener` to the trace listener table in `add-element-for-listeners-for-trace.md`.
- Clarified its purpose and default behavior in debugging and tracing scenarios.

Related Issue

Reviewers
gewarren
At least 1 approving review is required to merge this pull request.
Still in progress? Convert to draft

Assignees
No one assigned

Labels
community-contribution dotnet-framework/svc

Projects
None yet

Figure 6.1: PR 1 – DefaultTraceListener Documentation Update

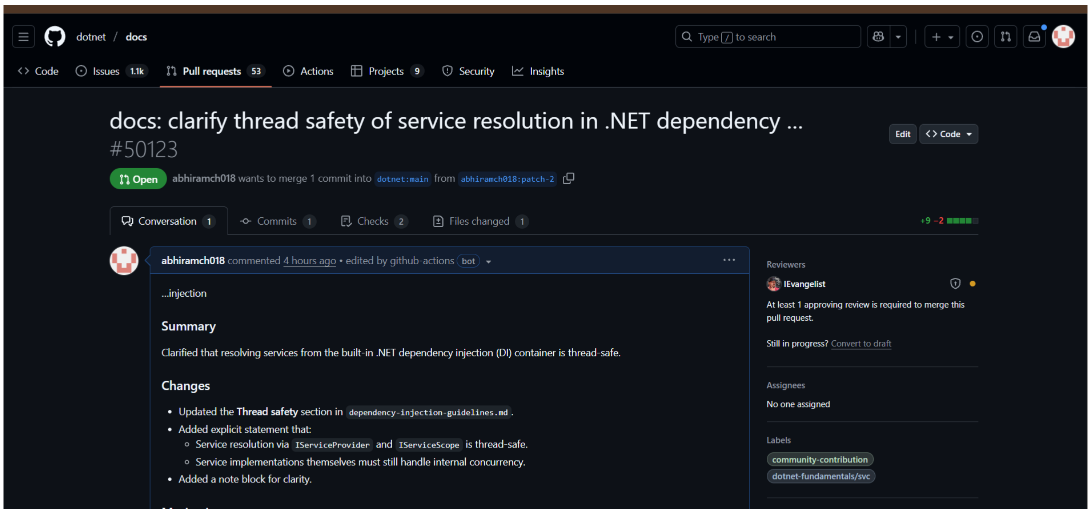
PR 2 – Dependency Injection Thread Safety Clarification (#50123)

Repository: dotnet/docs **Status:** Open

This PR clarified how thread safety works with dependency injection containers in .NET. Many developers were unsure whether resolving services concurrently was safe.

The documentation update:

- Explained that resolving services is thread-safe.
- Mentioned how lifetimes such as singleton and transient behave in multi-threaded scenarios.
- Linked to relevant official guidance for deeper reading.



docs: clarify thread safety of service resolution in .NET dependency ... #50123

Open abhiramch018 wants to merge 1 commit into `dotnet:main` from `abhiramch018:patch-2`

abhiramch018 commented 4 hours ago • edited by github-actions bot

...injection

Summary

Clarified that resolving services from the built-in .NET dependency injection (DI) container is thread-safe.

Changes

- Updated the **Thread safety** section in `dependency-injection-guidelines.md`.
- Added explicit statement that:
  - Service resolution via `IServiceProvider` and `IServiceScope` is thread-safe.
  - Service implementations themselves must still handle internal concurrency.
- Added a note block for clarity.

Reviewers
IEvangelist
At least 1 approving review is required to merge this pull request.
Still in progress? Convert to draft

Assignees
No one assigned

Labels
community-contribution
dotnet-fundamentals/svc

Figure 6.2: PR 2 – Dependency Injection Thread Safety Clarification

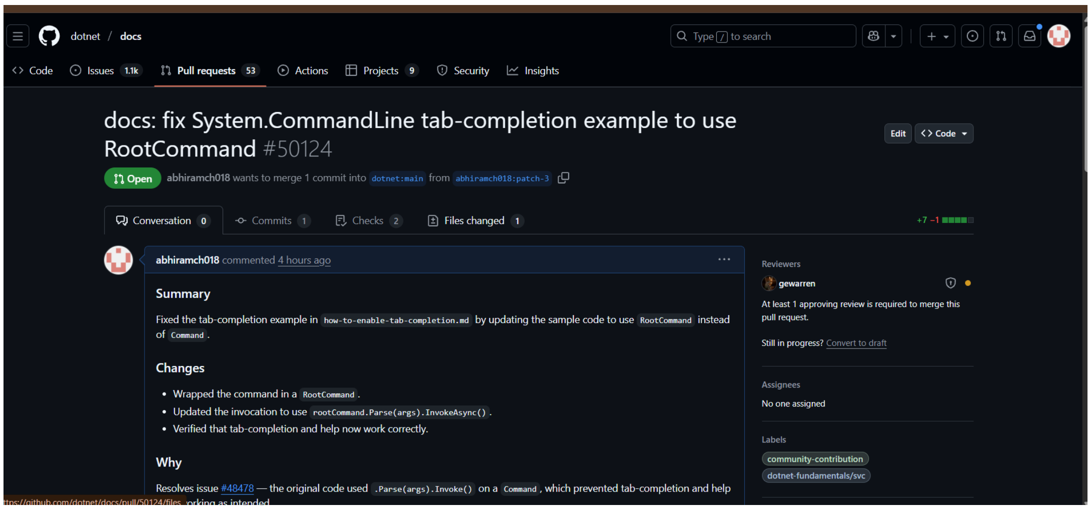
PR 3 – System.CommandLine Tab Completion Fix (#50124)

Repository: dotnet/docs   **Status:** Open

This PR fixed an example related to tab completion using `System.CommandLine`. The original sample used an incorrect invocation which could mislead developers trying to implement similar functionality.

Key changes:

- Replaced the incorrect example with one using `RootCommand`.
- Ensured that the updated sample matched current library behavior.
- Improved comments and structure for easier understanding.



docs: fix System.CommandLine tab-completion example to use RootCommand #50124

Open abhiramch018 wants to merge 1 commit into `dotnet:main` from `abhiramch018:patch-3`

abhiramch018 commented 4 hours ago

Summary

Fixed the tab-completion example in `how-to-enable-tab-completion.md` by updating the sample code to use `RootCommand` instead of `Command`.

Changes

- Wrapped the command in a `RootCommand`.
- Updated the invocation to use `rootCommand.Parse(args).InvokeAsync()`.
- Verified that tab-completion and help now work correctly.

Why

Resolves issue #48478 — the original code used `.Parse(args).Invoke()` on a `Command`, which prevented tab-completion and help orking as intended

ttps://github.com/dotnet/docs/pull/50124/files

Reviewers

gewarren

At least 1 approving review is required to merge this pull request.

Still in progress? Convert to draft

Assignees

No one assigned

Labels

community-contribution

dotnet-fundamentals/svc

Figure 6.3: PR 3 – System.CommandLine Tab Completion Example Fix

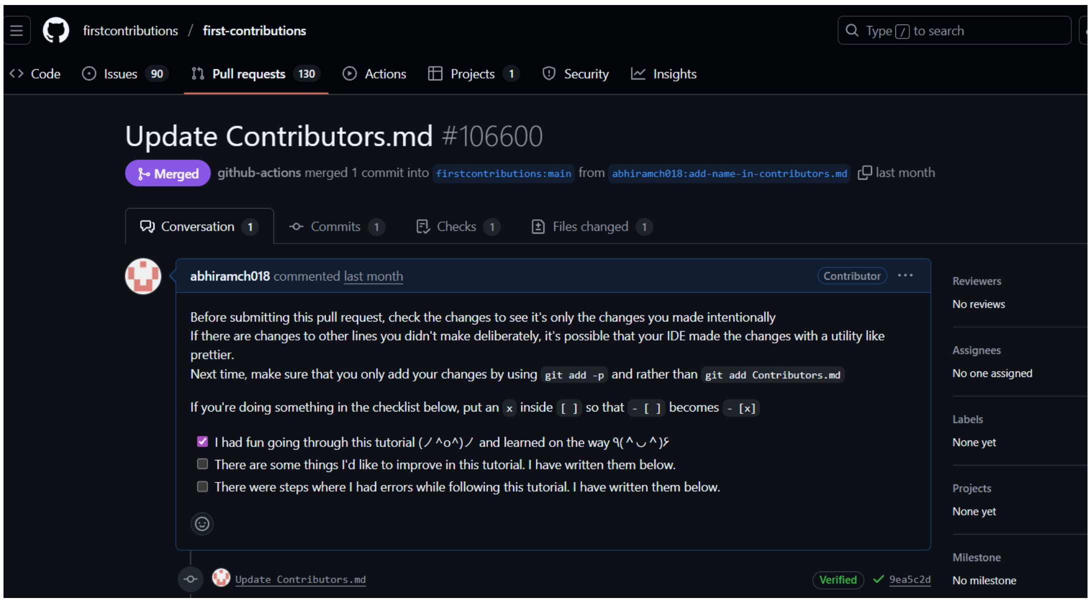
PR 4 – First Contributions (#106600)

Repository: firstcontributions/first-contributions **Status:** Merged

This PR was an introductory contribution to the `firstcontributions` repository. It involved adding the username `abhiramch018` to the `Contributors.md` file.

Although simple, this PR:

- Helped understand the full GitHub PR workflow.
- Provided confidence to contribute to larger projects.
- Demonstrated how even small contributions are valued in open source.



Update Contributors.md #106600

Merged github-actions merged 1 commit into `firstcontributions:main` from `abhiramch018:add-name-in-contributors.md` last month

abhiramch018 commented last month Contributor

Before submitting this pull request, check the changes to see it's only the changes you made intentionally
If there are changes to other lines you didn't make deliberately, it's possible that your IDE made the changes with a utility like prettier.
Next time, make sure that you only add your changes by using `git add -p` and rather than `git add Contributors.md`

If you're doing something in the checklist below, put an `x` inside `[ ]` so that `- [ ]` becomes `- [x]`

- [x] I had fun going through this tutorial (ノ^o^)ノ and learned on the way ٩(^◡^)۶
- [ ] There are some things I'd like to improve in this tutorial. I have written them below.
- [ ] There were steps where I had errors while following this tutorial. I have written them below.

Update Contributors.md Verified ✓ 9ea5c2d

Reviewers
No reviews

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Figure 6.4: PR 4 – First Contribution Successfully Merged

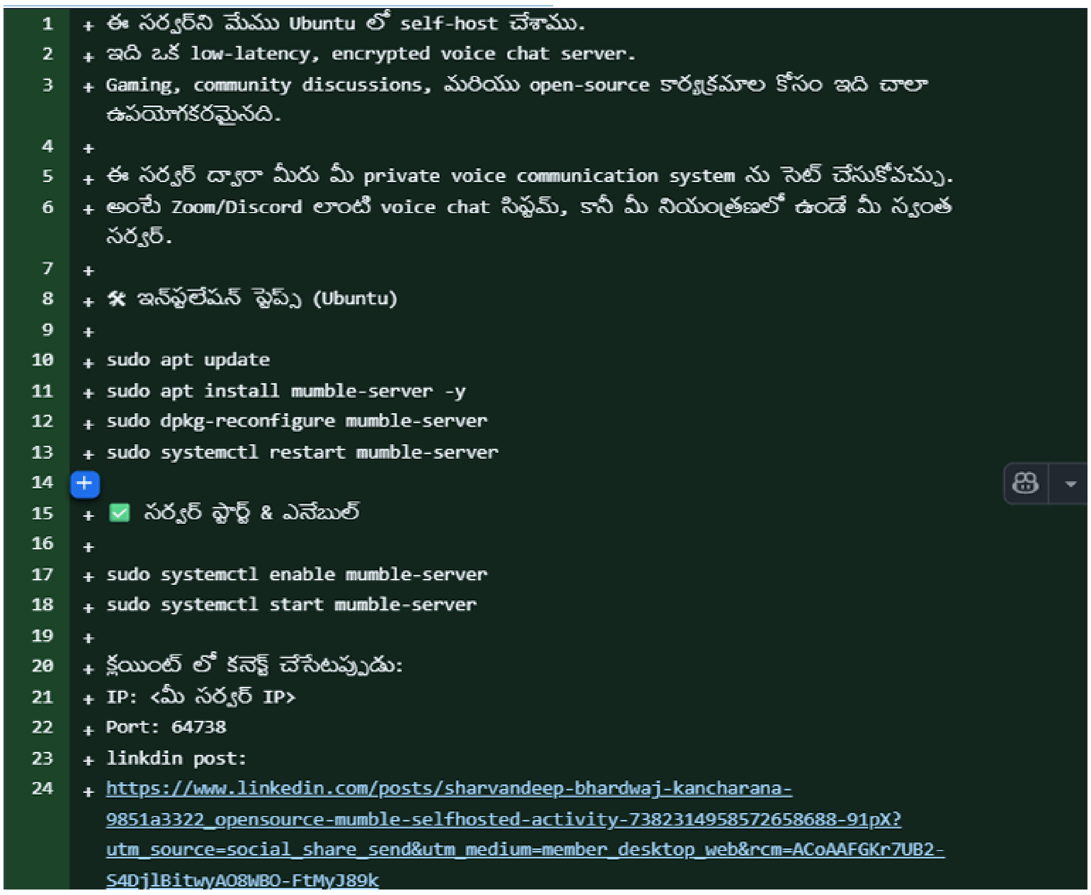
PR 5 – Localized Telugu User Guide for Mumble

Repository: Coursework / documentation **Status:** Completed

As an extension of the Mumble deployment, a localized Telugu guide was created. While not a GitHub PR in a public repository, it represents an important contribution to documentation and accessibility.

The guide:

- Described Mumble usage in Telugu.
- Explained basic features like connecting, joining channels, and using push-to-talk.
- Made the project more approachable to Telugu-speaking students.



```
+ 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
```

Figure 6.5: PR 5 – Localized (Telugu) Mumble User Guide

Summary of Contributions

These contributions, though mostly documentation-focused, provided practical experience with open source collaboration. They emphasized clear communication, careful reading of existing documentation, and respectful engagement with maintainers.

7. LinkedIn Posts (Professional Outreach)

As part of the **Open Source Engineering** coursework, three LinkedIn posts were published to document learning progress, share open-source experiences, and communicate achievements with a professional audience. Social media platforms such as LinkedIn provide visibility to student projects, help connect with professionals in the tech industry, and foster participation in open source and developer communities.

Each post represented a different milestone — from deploying a self-hosted server to contributing to real-world repositories, and finally reflecting on the overall journey.

Post 1 – Self-Hosted Mumble Server Deployment

Theory: The first LinkedIn post focused on the deployment of a self-hosted **Mumble voice server**. It summarized the steps followed during the setup on Ubuntu 24.04 LTS and explained how self-hosting helps students understand real-world server management. The post emphasized learning system administration, Linux service management, and the importance of encrypted, decentralized communication platforms. It also served as an introduction to the concept of open-source infrastructure ownership.

LinkedIn Post Link: View Post 1 on LinkedIn

Conclusion: This post received positive engagement and demonstrated how even simple deployments can build confidence in managing self-hosted servers. It marked the first major practical success of the coursework and encouraged others to experiment with open-source hosting tools.

Post 2 – First GitHub Contribution and Merge

Theory: The second post reflected on successfully making the first open source contribution on GitHub and getting it merged. It discussed the overall contribution process — forking a repository, cloning it locally, creating a new branch, committing changes, and submitting a pull request. The focus was on learning the collaborative workflow of GitHub, following contribution guidelines, and understanding the importance of documentation fixes and clarity improvements in real-world repositories.

LinkedIn Post Link: View Post 2 on LinkedIn

Conclusion: This post highlighted the satisfaction of contributing to open source for the first time. It encouraged more students to participate in the open-source ecosystem and helped demonstrate how collaboration improves software quality and developer learning simultaneously.

Post 3 – Open Source Engineering Experience Reflection

Theory: The third post was written as a concluding reflection on the entire **Open Source Engineering** course. It captured insights gained from working with Linux distributions, encryption tools, and real-world GitHub repositories. It also described how the project improved both technical and professional communication skills, helping to document and present one's work effectively.

LinkedIn Post Link: View Post 3 on LinkedIn

Conclusion: This final post summarized the importance of consistency and public sharing in open source learning. It reflected professional growth, confidence in using open tools, and readiness to contribute to larger projects in the future.

Overall Reflection

Combining technical project execution with professional outreach was one of the most rewarding aspects of the course. These LinkedIn posts not only showcased progress but also built a bridge between academic work and public engagement. By communicating technical experiences openly, the project demonstrated how modern engineers can integrate learning, reflection, and community contribution to develop both skill and visibility in the open-source ecosystem.