OPEN SOURCE ENGINEERING

Student ID: 2400032695
Name: Yerrabothula Navya Sree
Semester: Odd
Academic Year: 2025-2026
Course Code: 24CS02EF

1 Understanding the Core Ubuntu Linux Distribution

1.0.1 1. Overview and Philosophy

Ubuntu is a widely adopted, community-focused operating system built on top of the Debian Linux base. It is known for combining reliable performance with simplicity, making it suitable for both new learners and experienced professionals. Canonical, the organization behind Ubuntu, follows the philosophy of making Linux approachable and efficient for all users. This commitment to openness and stability has made Ubuntu one of the most trusted Linux distributions worldwide, supported by a massive ecosystem of developers and contributors.

1.0.2 2. The Desktop Experience (GNOME)

The default Ubuntu desktop experience is powered by the **GNOME** environment, which is designed to offer a modern and minimal interface. A left-side dock provides quick access to frequently used applications, while the top panel manages essential system indicators like volume, networking, and notifications. The **Activities Overview**, activated using the Super key, allows users to switch between applications, search the system, and manage workspaces. GNOME focuses on reducing distractions, offering smooth transitions, and ensuring that the system remains user-friendly and productive. Additionally, Ubuntu is widely appreciated for recognizing hardware components automatically, making the setup process straightforward.

1.0.3 3. Software Management and Packaging

Ubuntu offers a dual-approach software installation system. The first method uses the traditional **APT package manager** along with **DEB packages**, which are sourced from Ubuntu's official repositories and guarantee stability. The second method utilizes **Snap packages**, a containerized format that includes applications bundled with their dependencies. Snaps run in a **sandbox**, increasing security and ensuring consistent performance across different Ubuntu releases. This dual system gives users the flexibility to choose between long-tested packages and newer, regularly updated applications.

2 Encryption and GPG

2.1 Types of Encryption in Ubuntu

Ubuntu provides strong tools to safeguard user data. Two commonly used techniques include **Full Disk Encryption** and **File-Based Encryption**.

2.1.1 1. Full Disk Encryption (FDE)

- **Definition:** Encrypts the entire storage device, including system files, user data, and swap areas.
- **Mechanism:** Ubuntu uses **LUKS**, a secure encryption standard. A passphrase is required during system startup.

- **Purpose:** Prevents unauthorized data access in case the device is stolen or the disk is accessed externally.
- **Setup:** Usually enabled during OS installation for maximum security.

2.1.2 2. File and Directory Encryption

- **Definition:** Allows users to encrypt specific files or folders instead of the entire disk.
- **Tools:**
  - **GPG:** The preferred tool for secure document protection and communication.
  - **eCryptfs:** Older method used for encrypting home directories, now replaced by FDE.

2.2 GPG (GNU Privacy Guard) Explained

GPG is an essential encryption utility that follows the **OpenPGP** standard. It allows users to encrypt files, authenticate identities, and sign data securely.

2.2.1 1. Core GPG Concepts

- **Public Key:** Shared with others so they can encrypt data for you or verify your signatures.
- **Private Key:** Stays secure with the owner and is used for decryption and signing.
- **Asymmetric Encryption:** Ensures only the rightful owner can read encrypted information.

2.2.2 2. Basic GPG Command-Line Usage

A. Generate a Key Pair

```
gpg --full-generate-key
```

B. Symmetric Encryption

```
gpg -c myfile.txt
```

C. Encrypt for a Recipient

```
gpg --encrypt --recipient "recipient@example.com" myfile.txt
```

D. Decrypt a File

```
gpg --decrypt myfile.txt.gpg
```

3 Sending Encrypted Email

3.1 Prerequisite: Setting Up GPG

1. Both sender and receiver must generate a GPG key pair:

   ```
   gpg --full-generate-key
   ```

2. Exchange Public Keys using:

   ```
   gpg --armor --export "Name" > publickey.asc
   ```

3. Import the received Public Key:

   ```
   gpg --import publickey.asc
   ```

3.2 Sending the Encrypted Email

The most efficient method is using **Mozilla Thunderbird**, which has built-in OpenPGP support.

3.2.1 1. Compose the Email

Open Thunderbird and write your message normally.

3.2.2 2. Apply Encryption and Signing

Enable:

- **Encrypt** – Uses the recipient's Public Key.
- **Sign** – Uses your Private Key for identity verification.

3.2.3 3. Verification and Sending

Thunderbird checks available keys and then encrypts the message before sending.

3.2.4 4. Recipient's Decryption Process

- Uses their Private Key to decrypt.
- Uses your Public Key to confirm the signature and message authenticity.

4 Privacy Tools From Prism Break

4.0.1 1. Tor Browser

A privacy-focused browser that routes traffic through the Tor network, hiding IP addresses and preventing tracking.

4.0.2 2. Debian

A fully open-source Linux distribution known for strict free software guidelines and long-term stability, recommended for privacy-conscious users.

4.0.3 3. Thunderbird

An open-source email application with built-in OpenPGP encryption, ideal for securely managing mail.

4.0.4 4. KeePassXC

A secure password manager that stores encrypted credentials locally, reducing dependency on cloud services.

4.0.5 5. Firefox

A customizable browser with advanced tracking protection and strong privacy controls, suitable for everyday browsing.

5 Open Source License

5.1 The Core Purpose and Classification

The MIT License is one of the most flexible open-source licenses available. Its fundamental aim is to encourage developers and organizations to reuse code freely without imposing restrictive conditions.

5.2 Granted Rights and Permissions

The license allows users to copy, modify, distribute, publish, and even commercialize the software. This makes MIT-licensed projects ideal for individuals and companies looking for minimal licensing barriers.

5.3 The Only Two Conditions

The MIT License requires only:

- The original copyright notice.
- The inclusion of the full license text.

5.4 Disclaimer of Warranty

It clearly states that the software is provided "as is," with no guarantees. The author cannot be held liable for damages or issues that arise from using the code.

6 Self Hosted Server

6.1 About

PhotoHub is a modern, open-source platform built for organizing and storing photos on your own server. It allows users to upload, manage, and share their images securely without depending on external cloud services. Designed with simplicity and performance in mind, it runs smoothly on personal machines, home servers, or small team environments.

6.1.1 Key Features

- **Self-Hosted Photo Storage:** PhotoHub allows you to run your own secure photo server, giving complete ownership and privacy over your images.
- **Lightweight Performance:** Designed to run smoothly on low-power devices and small servers, making it ideal for personal hosting setups.
- **Secure Access Control:** Supports private albums, user authentication, and restricted sharing to keep your media fully protected.
- **Simple Web Interface:** Comes with an intuitive browser-based UI that makes uploading, organizing, and viewing photos effortless.
- **Database Flexibility:** Compatible with commonly used databases such as MySQL, PostgreSQL, and SQLite for easy deployment.
- **Developer Extensibility:** Offers plugin and API support that allows developers to extend functionality and integrate new features.
- **Automatic Image Optimization:** Includes intelligent resizing, compression, and metadata handling for efficient storage and smooth browsing.

6.2 Installation Process (Docker Compose)

Hosting **PhotoHub** using **Docker Compose** is one of the easiest and most stable deployment methods, as it automatically manages all required components such as the web application, database, and background workers in a clean multi-container setup. Before beginning the installation, ensure that both **Docker** and **Docker Compose** are properly installed on your system. These tools are included in Docker Desktop for Windows/macOS or can be installed separately on Linux-based systems.

The installation process starts by **downloading the official configuration templates**. The recommended approach is to clone the PhotoHub repository or fetch the `docker-compose.yml` file along with the sample `.env` configuration file using `curl` or `wget`. After these files are placed in your deployment folder, the next step is to **configure the environment variables**. Inside the `.env` file, update essential fields such as `PHOTOHUB_URL` to point to your server's domain or IP address (for example, `http://photos.example.com`). Additionally, you need to define secure credentials for the admin account, database connection, and storage directory. These can be generated using commands like `openssl rand -hex 32` for improved security.

Once the configuration is complete, proceed to **start the containers and download the necessary images**. Running the command `docker compose up -d` will automatically fetch the PhotoHub application image, create the configured database container, and initialize the internal services. Docker Compose ensures that all dependencies, such as storage volumes and networking, are set up correctly without requiring manual intervention.

After the services are successfully launched, accessing PhotoHub is straightforward. Simply open a browser and navigate to the URL specified in the `PHOTOHUB_URL` variable. If deploying locally, the default access point is usually `http://localhost:8000`. Upon first access, PhotoHub will guide you through a short initialization process, allowing you to set up your administrator account and begin uploading and organizing your photos. Once the setup is done, you can immediately start managing albums, uploading images, and customizing your self-hosted photo library.

PhotoHub Resources

(DEEMED TO BE UNIVERSITY) HTE HONORS THROUGH EXPERIENTIAL LEARNING

EXPERIENTIAL LEARNING & GLOBAL ENGAGEMENT

Open Source Engineering

PhotoHub

PhotoHub Self Hosted Server

- Photo Hub is a lightweight, open-source photo management server.It allows users to securely upload, organize, and share images via a web interface.
- Designed for speed and simplicity, it works perfectly on personal servers or small teams.

Highlights / Features

- Self-Hosted — Full control over your photos and data.
- Fast & Lightweight — Optimized for low-power devices and small servers.
- Secure Access — Private albums and authentication support.
- Web-Based Interface — Easy to manage from any browser.
- Database Support — Compatible with MySQL, PostgreSQL, or SQLite.
- Extensible — API and plugin support for developers.
- Automatic Image Optimization — Smart resizing and metadata handling.

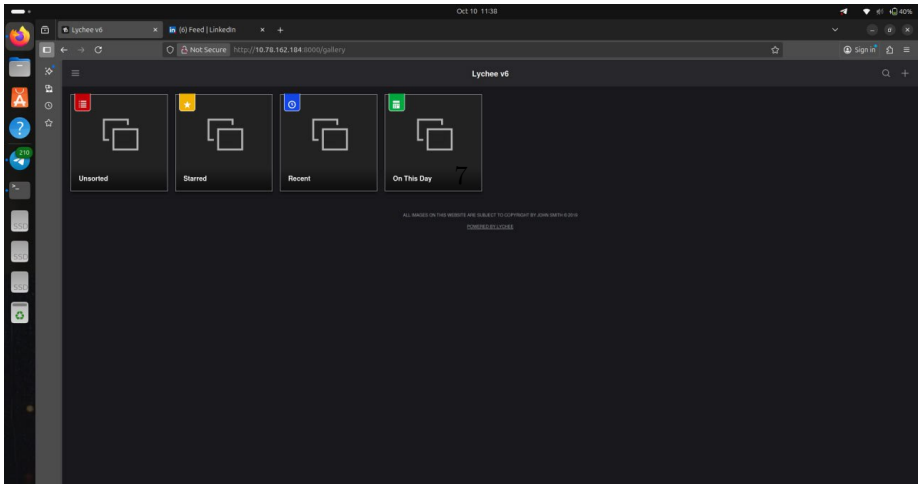
License :

MIT License

Hosted by

Y. Navya Sree - 2400032695

D. Sirisha - 2400031818



7 Open Source Contribution

7.1 PR 1 : First Contribution

7.1.1 Overview

My first open-source contribution was made to the popular beginner-friendly repository **firstcontributions/first-contributions**. This project helps new developers understand the workflow of contributing to open-source projects. My pull request, titled *"Added NavyaSree to Contributors list"*, was successfully merged as PR #106560.

7.1.2 Objective of the Contribution

The main purpose of this contribution was to complete the standard Git workflow for the first time. The repository is designed to teach newcomers how to fork, clone, modify, and submit a pull request by adding their name to the `Contributors.md` file.

7.1.3 Steps I Followed

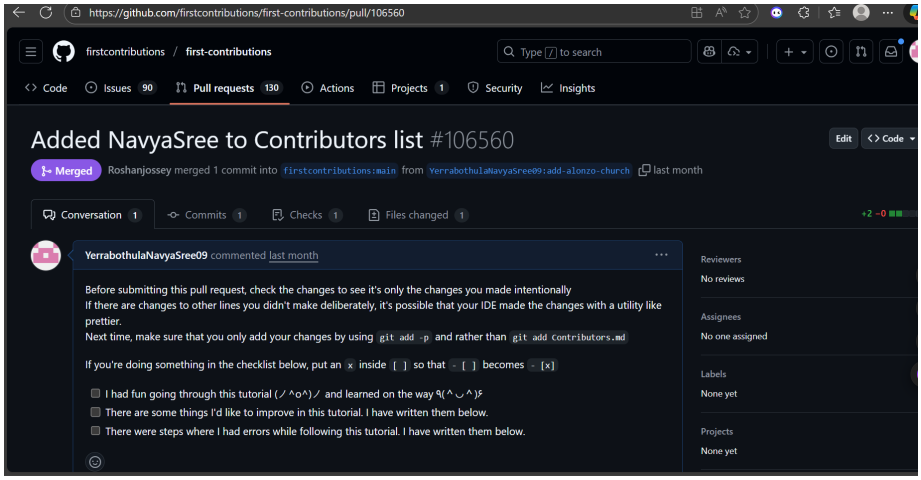
- **Forked the Repository:** Created my own copy of the project from GitHub.
- **Cloned the Fork:** Downloaded the repository to my local machine using the SSH `git clone` command.
- **Created a New Branch:** Used `git switch -c add-navya-name` to keep my changes separate.
- **Updated Contributors File:** Added my name inside the `Contributors.md` document.
- **Committed the Change:** Staged and committed the update using `git commit -m "Added NavyaSree to Contr`
- **Pushed the Branch:** Uploaded the branch to my GitHub fork using `git push -u origin add-navya-name`.
- **Created Pull Request:** Submitted the PR through GitHub's "Compare & pull request" option.

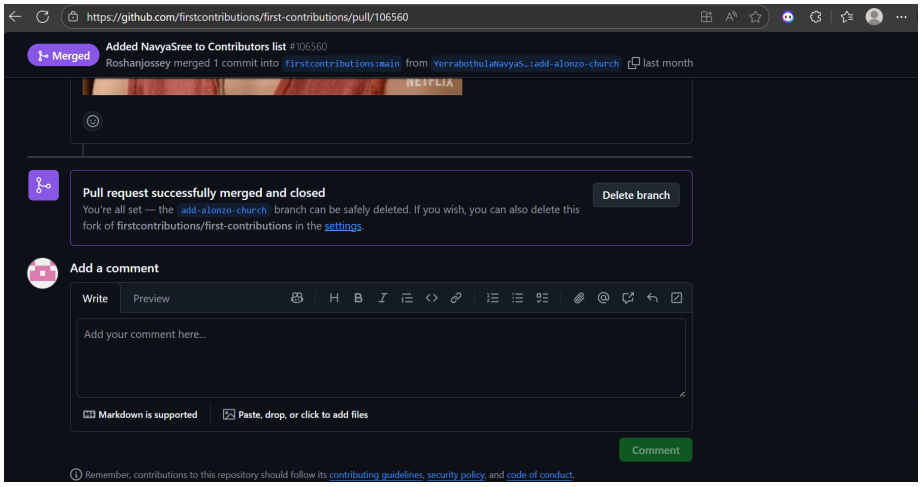
7.1.4 Challenges Faced

- **SSH Setup:** Initially faced a push error because GitHub no longer supports password authentication. Configured SSH keys to fix the issue.
- **Branch Naming:** Learned the importance of meaningful branch names for better project organization.

7.1.5 Result

The pull request was reviewed and merged successfully. Completing this contribution helped me understand the full GitHub workflow and boosted my confidence to work on more advanced open-source issues.





7.2 PR 2 : CodeYourFuture Curriculum

7.2.1 Overview

My second open-source contribution was made to the **CodeYourFuture/curriculum** repository, a community-driven project that provides free learning resources for aspiring developers. The pull request I submitted was titled *"Fix #1066: Added time-stamper for Day Plan TOC timings"* and was opened as PR #1623. This contribution involved improving the readability and structure of the training curriculum.

7.2.2 Goal of the Contribution

The issue #1066 requested that the Day Plan section in the curriculum should include a clear **time-stamp format** for the Table of Contents (TOC). This helps students and mentors quickly navigate through different segments of the daily schedule without confusion.

7.2.3 What I Updated

- Added a consistent **time-stamp** notation alongside each Day Plan topic.
- Ensured the timing format matched the existing curriculum standards.
- Updated markdown entries to improve clarity and flow inside the TOC.
- Verified that the changes reflected correctly in the rendered documentation.

7.2.4 Workflow Followed

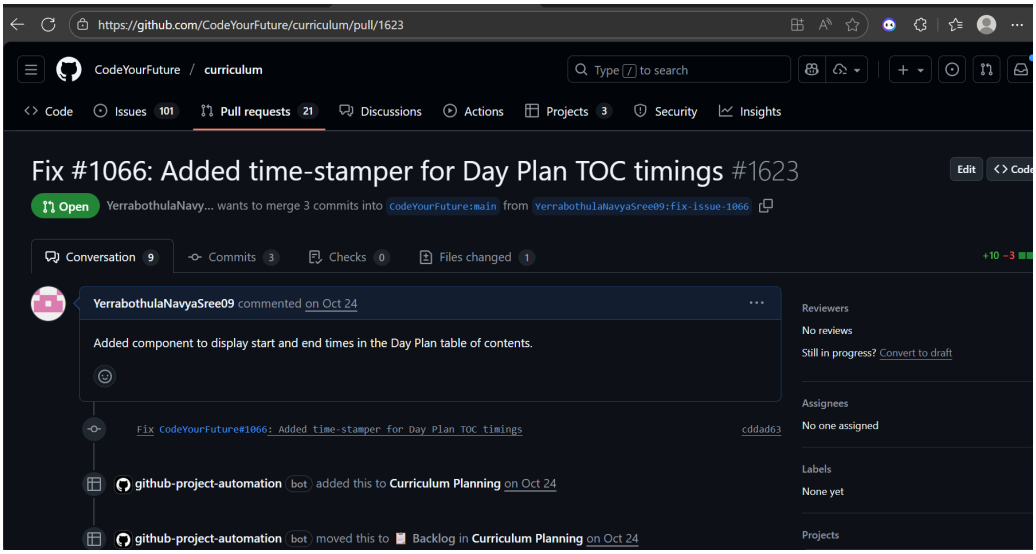
- Identified the issue and understood the expected format from maintainers.
- Forked the **CodeYourFuture/curriculum** repository.
- Created a new branch dedicated to the fix.
- Edited the required markdown files to insert timestamps.
- Committed the changes using an appropriate message: `"Added time-stamper for Day Plan TOC timings"`.
- Pushed the branch and opened a PR for review.

7.2.5 Challenges Faced

- Ensuring the timestamp alignment matched the existing style guidelines.
- Cross-checking multiple curriculum files to maintain consistency.
- Making sure no formatting was broken during markdown updates.

7.2.6 Current Status

The pull request is currently marked as **"Review Required"**. It is awaiting feedback from CodeYourFuture mentors before merging.



7.3 PR 3 : GovDirectory (website)

7.3.1 Introduction

This Pull Request was submitted to the **GovDirectory/website** open-source project, which is a global, community-driven platform that organizes official government contact information in a structured and verifiable way. The aim of the PR was to improve the clarity and organization of project issues by introducing **more specific and meaningful labels**.

7.3.2 The Issue

The existing issue labels in the project were very limited and did not clearly represent the type or priority of tasks. This made it difficult for contributors and maintainers to quickly understand:

- the category of the issue,
- whether it was frontend or backend related,
- the urgency or workflow status,
- and the type of contribution expected.

7.3.3 My Contribution

In this PR, I added a new set of structured and descriptive labels that improved the overall workflow. The new labels included:

- **Epic**
- **UI/UX**
- **Blocked / Wait**

- **Frontend** and **Backend**
- **New contact point**

These labels help categorize issues more precisely and make the repository more contributor-friendly.

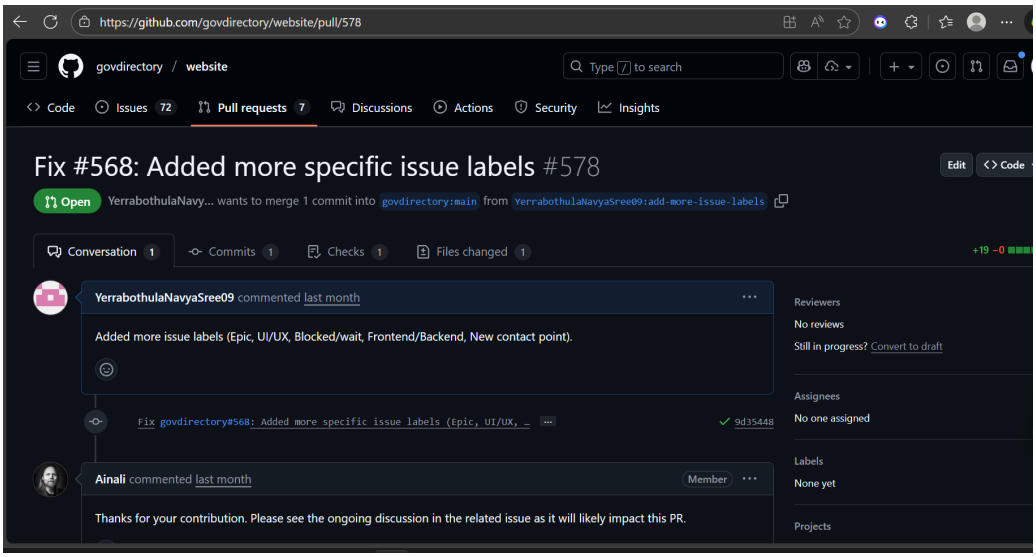
7.3.4 Impact of the Contribution

This update helps maintainers and new contributors:

- quickly filter issues based on type,
- understand priority and workflow,
- improve overall issue management,
- and maintain consistent labeling practices.

7.3.5 Reviewer Feedback

The maintainers appreciated the contribution and noted that ongoing discussions in the related issue would help finalize label usage. The PR is currently **open** and awaiting further review.



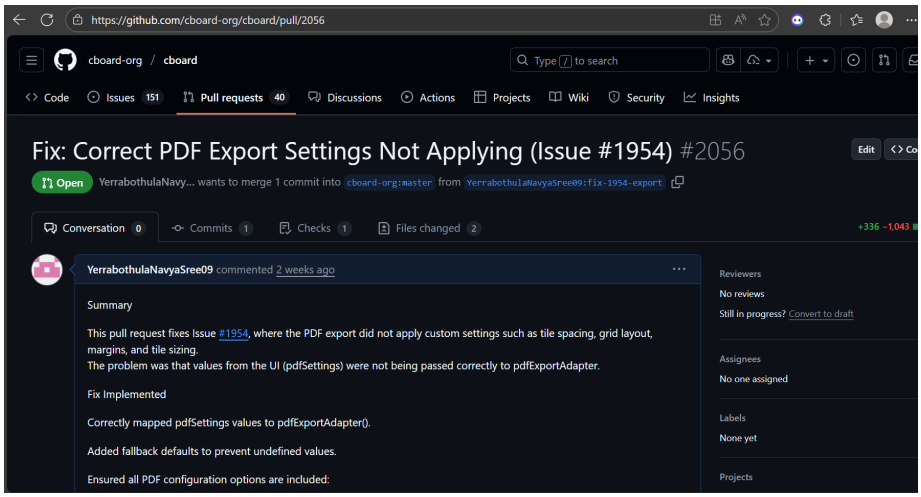
7.4 PR 4: cboard-org (Correct PDF Export Settings Not Applying)

7.4.1 The Core Problem

This Pull Request addresses Issue #1954, where the PDF export feature did not correctly apply custom user-defined settings. These settings included tile spacing, grid layout, margins, and tile sizing. Although the user interface allowed customization through `pdfSettings`, the underlying issue was that these values were not being passed correctly to the `pdfExportAdapter`. As a result, exported PDFs ignored the custom configuration, leading to inconsistent or incorrect output.

7.4.2 The Solution: Correct Mapping and Reliable Defaults

To resolve this issue, the Pull Request implements a proper mapping of all `pdfSettings` values to the `pdfExportAdapter()`, ensuring that user-selected configurations are accurately applied during export. Additionally, fallback defaults were introduced to prevent undefined values from causing failures. The update ensures that all relevant PDF configuration options are included and processed reliably, resulting in a consistent and accurate PDF export workflow.



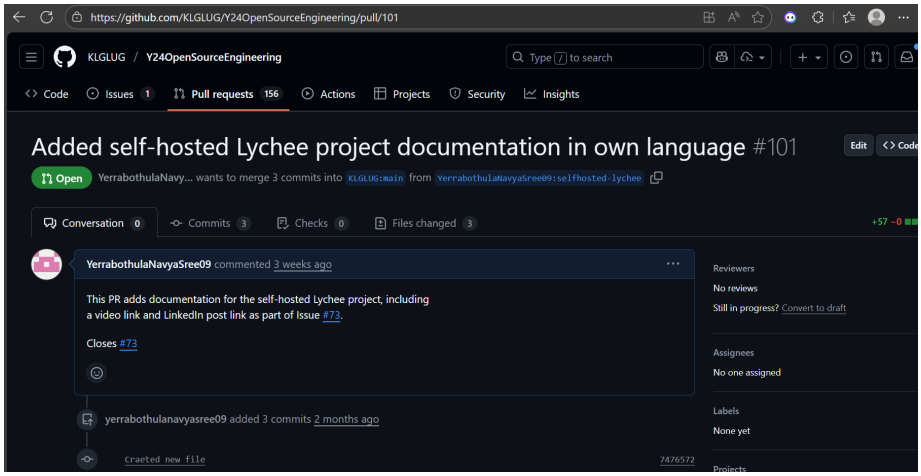
7.5 PR 5: Added Self-Hosted Lychee Project Documentation

7.5.1 The Core Problem

Before this Pull Request, the repository lacked proper documentation for the self-hosted Lychee project. Contributors and users did not have a clear reference explaining how to set up, understand, or navigate the self-hosted version. This gap made it difficult for new contributors to get involved and for users to fully utilize the project's capabilities. Additionally, Issue #73 highlighted the need for adding supporting materials such as tutorial links and external resources.

7.5.2 The Solution: Comprehensive Documentation and Resource Integration

This Pull Request introduces a complete documentation update for the self-hosted Lychee project. It adds new content explaining project usage and includes helpful external resources such as a video link and a LinkedIn post link to give users additional context. These improvements ensure that newcomers have a clear, structured guide while also closing Issue #73 by providing the missing documentation. The changes strengthen the project's overall accessibility and help maintain long-term clarity for contributors and users.



8 Linkedin Post Links

8.1 PR :

https://www.linkedin.com/feed/update/urn:li:activity:7399134826387181568/U

8.2 Journey Of Open Source :

https://www.linkedin.com/pulse/my-open-source-journey-y-navya-sree-n9vjc/?trackingId=sh2FnlXk9i9f1Yz%2F7FLmSA%3D%3D

8.3 Self Hosted Project :

https://www.linkedin.com/pulse/photo-hub-self-hosted-lychee-server-y-navya-sree-w6azc/?trackingId=wX61P4OwRVCg%2FiZ1iKrhhg%3D%3D