OPEN SOURCE ENGINEERING

Name: Golla Sai Sree

Student ID: 2400032694

Semester: Odd

Academic Year: 2025-2026

Course Code: 24CS02EF

Under the guidance of

Dr. Sripath Roy Koganti

1 Understanding the Core Ubuntu Linux Distribution

1.0.1 1. Overview and Philosophy

Ubuntu is a free and open-source operating system based on Debian Linux. It is one of the most popular Linux systems used on laptops and desktops. Ubuntu is known for being simple to use, even for beginners, while still offering powerful features for developers.

Ubuntu is developed and supported by Canonical Ltd. The main idea behind Ubuntu is "Linux for everyone." This means Ubuntu wants to make computers easy to use, stable, and accessible for all people, whether they are new learners or experienced programmers.

1.0.2 2. The Desktop Experience (GNOME)

Ubuntu uses the **GNOME** desktop environment by default. It gives a clean, modern, and easy-to-understand look. On the left side, there is a dock that shows important applications, making it simple to open apps quickly.

The **Activities Overview** can be opened by pressing the Super (Windows) key. It allows users to see all open windows, switch between workspaces, and search for apps or files in one place. This makes working on Ubuntu smooth and productive.

Ubuntu also supports most hardware automatically. When we install it, many devices like Wi-Fi, Bluetooth, and display drivers work without extra setup, which makes the installation process easier for users.

1.0.3 3. Software Management and Packaging

Ubuntu uses two main systems to install and manage software. The first one is the traditional **Advanced Packaging Tool (APT)**, which handles **DEB** packages from the official Ubuntu repositories. It is mainly used for system applications and trusted software.

The second system is **Snaps**, created by Canonical. Snaps include the application and all the files it needs to run, so they work the same on different Ubuntu versions. Snaps run in a **sandbox**, which means they are separated from the main system, improving security.

With both APT and Snaps, Ubuntu gives users a large collection of software that is easy to install, updated regularly, and safe to use.

2 Encryption and GPG

2.1 Types of Encryption in Ubuntu

Ubuntu mainly provides two important methods for keeping data safe: **Full Disk Encryption (FDE)** and **File/Directory Encryption**.

2.1.1 1. Full Disk Encryption (FDE)

- **What it is:** Full Disk Encryption protects the whole storage device or a large partition, including system files, swap, and user data.
- **How it works:** Ubuntu commonly uses **LUKS** (Linux Unified Key Setup) for this purpose. When the computer starts, the user must enter a **passphrase**. Only then does LUKS unlock and allow the system to load, while all encryption and decryption happens automatically in the background.
- **Purpose:** The main goal is to prevent others from accessing personal data if the laptop or hard drive is stolen or someone tries to read it while the device is powered off.
- **Implementation:** FDE is usually enabled during installation by selecting the option "Encrypt this Ubuntu installation." Setting it up later is possible, but much harder and risky.

2.1.2 2. File and Directory Encryption

- **What it is:** This method protects only selected files or folders, giving users more control over what information they want to secure.
- **Tools:**
  - **GPG (GNU Privacy Guard):** The most widely used tool for encrypting single files and messages, especially when using **public and private keys** for secure communication.
  - **eCryptfs (older):** Previously used for encrypting the Home folder, but now replaced mostly by Full Disk Encryption.

2.2 GPG (GNU Privacy Guard) Explained

GPG is the GNU version of the **OpenPGP** standard (based on Pretty Good Privacy - PGP). It is mainly used to secure files and verify communication by providing encryption and digital signatures.

2.2.1 1. Core GPG Concepts

A. Generating a Key Pair The first step in using GPG is creating a pair of keys: one public and one private. The public key can be shared with others, while the private key must always be kept secret.

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

During this process, you will be asked to choose the key type (usually RSA and RSA), key size (4096 bits is a good choice), expiry date, and enter your Name, Email ID, and a strong **passphrase** to protect your private key. Once completed, your key pair will be stored in your keyring.

B. Encrypting a File for Yourself (Symmetric Encryption)

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

After running this command, GPG will ask for a passphrase and then generate an encrypted file named `myfile.txt.gpg`. Only someone who knows the same passphrase can open it.

C. Encrypting a File for Someone Else (Asymmetric Encryption)

```
gpg --encrypt --recipient "recipient@example.com" mysecretfile.doc
```

This creates an encrypted file named `mysecretfile.doc.gpg`. Only the person who owns the matching Private Key can decrypt and read the file, ensuring safe communication.

D. Decrypting a File

```
gpg --decrypt mysecretfile.doc.gpg
```

GPG will automatically identify the required Private Key and prompt for the passphrase. Once verified, the decrypted output will either display in the terminal or be saved as a file, depending on the command used.

2.3 Sending the Encrypted Email

The most common and user-friendly way to send GPG-encrypted emails in Ubuntu is by using **Mozilla Thunderbird**, which now includes built-in OpenPGP support.

2.3.1 1. Compose the Message

- **Open Thunderbird** and start composing a new email.
- Type your message normally.

2.3.2 2. Encryption and Signing

When sending a secure email, two important actions take place:

1. **Encryption:** The email is encrypted using the **recipient's Public Key**. Only their **Private Key** can decrypt it, ensuring privacy.
2. **Digital Signature:** The email is **signed** using your **Private Key**, allowing the recipient to verify that the message truly came from you and was not modified.

In Thunderbird, this is done through the **OpenPGP / Security** menu in the compose window by enabling both **"Encrypt"** and **"Sign"** options before sending.

2.3.3 3. Verification and Sending

- The client will check that you have the required **Public Key** for the recipient(s). If a key is missing, it will warn you.
- When you click **Send**, Thunderbird uses GPG to encrypt the message body and attach your digital signature before transmitting the scrambled data.

2.3.4 4. Recipient's Experience (Decryption)

1. The receiver gets the encrypted email in their inbox.
2. Their email software automatically uses their **Private Key** (which is protected by their passphrase) to unlock and read the original message.
3. At the same time, the software checks your **Public Key** to verify the digital signature, making sure the message really came from you and was not changed.

3 Privacy Tools From Prism Break

3.0.1 1. Tor Browser (Web Browsers / Anonymizing Networks)

- **What it is:** A special web browser based on Firefox that sends your internet traffic through the Tor network made of volunteer-operated servers.
- **Privacy Focus:** Gives **strong anonymity** by hiding your IP address and your real location from websites. It also helps prevent browser fingerprinting.
- **PRISM Break Note:** PRISM Break suggests Tor Browser when users need the highest level of online anonymity.

3.0.2 2. Debian (Operating Systems)

- **What it is:** A widely trusted GNU/Linux distribution known for following Free Software rules and ethical values very strictly.
- **Privacy Focus:** Unlike closed-source systems such as Windows or macOS, Debian is fully open-source, which means anyone can review the code for security. It supports software freedom and transparency.
- **PRISM Break Note:** It is recommended as a strong GNU/Linux option for users moving away from proprietary operating systems due to its stability and freedom-focused design.

3.0.3 3. Thunderbird (Email Clients)

- **What it is:** A free, open-source email application created by Mozilla that works on multiple platforms.
- **Privacy Focus:** Thunderbird supports **OpenPGP** (GPG) encryption and digital signing directly, making it easy to secure and verify emails.
- **PRISM Break Note:** Recommended for safe email usage because of its built-in PGP features and open-source nature.

3.0.4 4. KeePassXC (Password Managers)

- **What it is:** An open-source password manager available on different operating systems.
- **Privacy Focus:** Saves all passwords inside one strongly encrypted database stored **locally** on your computer, giving full control to the user without needing cloud storage.
- **PRISM Break Note:** Preferred because of its strong encryption, open-source license, and no dependency on third-party cloud services.

3.0.5 5. Firefox (Web Browsers)

- **What it is:** A secure, customizable web browser developed by the non-profit Mozilla Foundation.
- **Privacy Focus:** Firefox is open-source and offers powerful privacy tools like enhanced tracking protection, container tabs, and many security add-ons (e.g., uBlock Origin).
- **PRISM Break Note:** When Tor Browser cannot be used, Firefox is suggested for everyday browsing after adjusting privacy settings and using a privacy-friendly search engine.

4 Open Source License Overview

The MIT License is known worldwide for being one of the simplest and most flexible open-source licenses. It originally came from the Massachusetts Institute of Technology. The main idea behind this license is to allow people to freely use the software with very few legal limitations. It is considered a **permissive license**, meaning developers can use, change, and share the software without being forced to release their own source code, unlike stricter licenses such as the GNU General Public License (GPL). Because of this freedom, the MIT License is widely used in both open-source and commercial software projects.

4.1 Granted Rights and Permissions

Anyone who receives a copy of the software and its documentation is given almost complete freedom to work with it. The license clearly allows users to **use, copy, modify, merge, publish, distribute, sublicense, and sell** the software. This makes the MIT License very attractive for developers who want to include MIT-licensed code in products that may later become closed-source or sold for profit, as long as they follow the basic rules of the license.

4.2 The Only Two Conditions for Distribution

Unlike licenses that require sharing improvements publicly, the MIT License only asks for two simple things when distributing the software. First, the original **Copyright Notice** must be included (for example, `Copyright <YEAR> <AUTHOR>`). Second, the complete **License Text** must also be included. If these two conditions are met, the user is free to handle the code in any way, including releasing modified versions under a different or proprietary license.

4.3 Disclaimer of Warranty and Liability

One of the most important parts of the MIT License is the strong disclaimer that protects the original developers. The software is provided **"AS IS"** without any kind of warranty or guarantee, whether expressed or implied, such as being safe to use or working for a specific purpose. The license also states that the authors **cannot be held responsible** for any damage, issues, or losses that occur while using the software. This means the user accepts all risks when using the software.

5 Self Hosted Server

5.1 About

KitchenOwl is an **open-source, self-hosted recipe and grocery management tool** that helps users store recipes, create shopping lists, and manage ingredients in an organized way. It focuses on being simple, fast, and easy to use for everyday cooking needs. It is licensed under the permissive **MIT License**, allowing anyone to use, modify, and host it with full control over their data.

5.1.1 Key Features

- **Recipe Management:** Users can create, edit, and organize recipes with ingredients, steps, and images.
- **Grocery Lists:** Automatically generates shopping lists from selected recipes, making shopping easier.
- **Meal Planning:** Includes a weekly meal planner to schedule recipes for each day.
- **Multi-Device Support:** Accessible from mobile, tablet, or desktop through a web interface.
- **User Accounts:** Supports user login and sharing with family or friends.
- **Self-Hosted:** All data stays on the user's own server, ensuring privacy and control.

5.2 Installation Process (Docker Compose)

The recommended and easiest method for self-hosting **KitchenOwl** is by using **Docker Compose**, which manages both the application and its **PostgreSQL database** as separate but connected containers. Before beginning, the system must have **Docker** and **Docker Compose** installed.

The installation begins by **downloading the official Docker Compose file**. This is done using the `curl` command to fetch the `docker-compose.yml` file into the desired directory. After downloading, a few important changes must be made: the `BASE_URL` should be set to your domain or local IP (e.g., `http://localhost`), a strong value must be set for the `SECRET_KEY` (generated using `openssl rand -hex 64`), and a secure password must be configured for the PostgreSQL database.

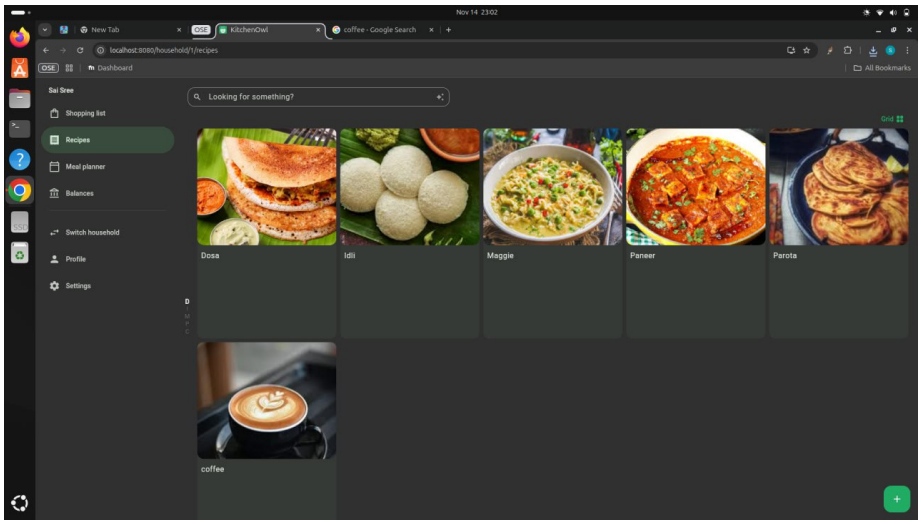
Once configured, the services can be started using:

```
docker compose up -d
```

This command downloads the required images and runs both the KitchenOwl and PostgreSQL containers in detached mode. The application can then be accessed through the browser using the `BASE_URL` value, commonly `http://localhost:8080` for local installations.

KitchenOwl Resources

KitchenOwl GitHub Repository



6 Open Source Contribution

6.1 PR 1 : First Contribution

6.1.1 Goal

The goal of this contribution was to make my first successful change in an open-source project and understand the complete contribution process. My task was simple but meaningful — adding my name to the project's `Contributors.md` file and getting the pull request reviewed and merged.

6.1.2 The Contribution Workflow

This contribution followed the common **fork - clone - edit - pull request** workflow, which is the basic structure followed in most open-source projects.

6.1.3 1. Setup

- **Fork:** I first forked the repository to my own GitHub account.
- **Clone:** Then I cloned the forked repository to my local system using the `git clone` command with the SSH link.

- **Prerequisites:** Git installation was required, and GUI alternatives were available for users who are not comfortable with the terminal.

6.1.4 2. Making Changes

- **Branch:** I created a separate branch for my changes using `git switch -c my-first-branch`.
- **Edit:** I added my name to the `Contributors.md` file.
- **Commit:** I staged the file using `git add Contributors.md` and committed it with the message `"Add Sai Sree to Contributors list"`.

6.1.5 3. Submission

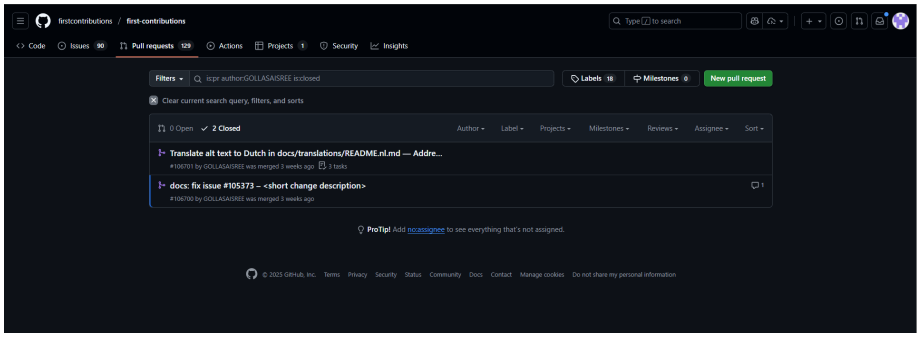
- **Push:** I pushed the branch to my GitHub remote using `git push -u origin my-first-branch`.
- **Pull Request (PR):** After pushing, I opened a pull request using the "Compare & pull request" option and submitted it for review by the maintainers.

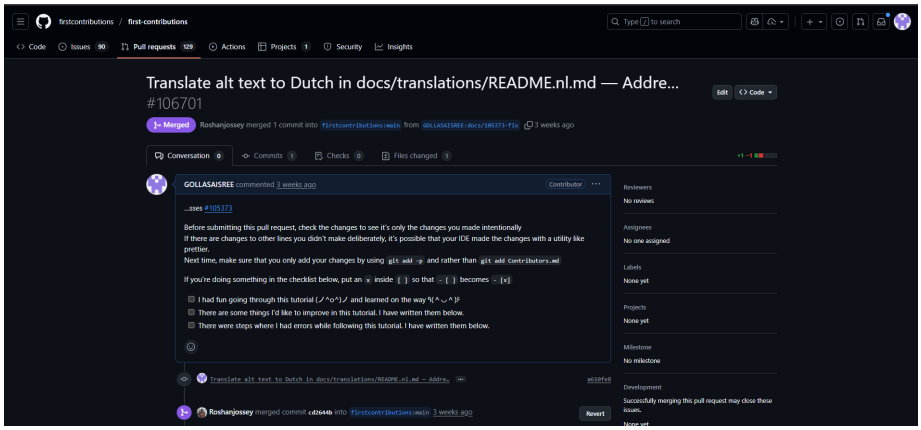
6.1.6 Difficulties and Solutions

- **SSH Authentication:** My first push failed due to authentication problems. I solved it by generating an **SSH key** and connecting it to GitHub.
- **Branch Command Issue:** The `git switch` command did not work initially, so I used the alternative `git checkout -b my-first-branch` which worked successfully.

6.1.7 Next Steps

After my PR was reviewed and **successfully merged**, I gained confidence to explore more beginner-friendly issues and continue contributing to open-source projects, including KitchenOwl.





6.2 PR 2 : Automagik Tools

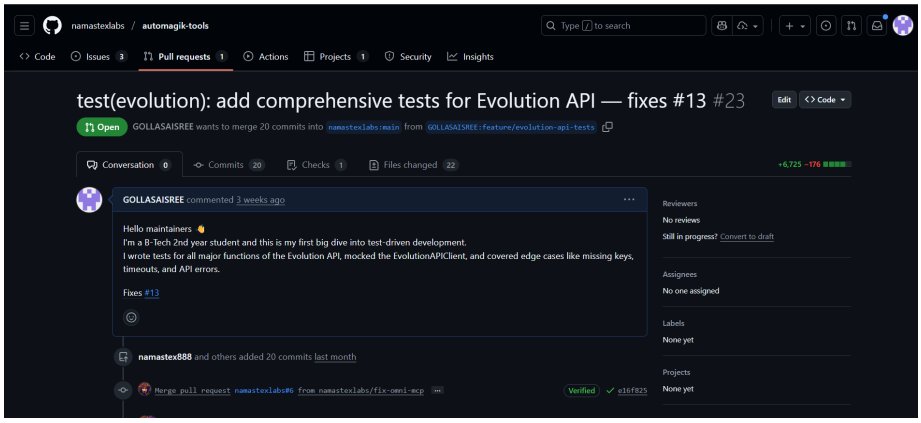
Automagik Tools is a collection of open-source automation scripts and utilities designed to simplify repetitive tasks, streamline workflows, and enhance productivity. The project is structured for easy extensibility, allowing contributors to add custom scripts in Python and Bash. It provides both CLI and GUI options, catering to both novice and advanced users.

6.2.1 Licensing and Contribution

The project is distributed under the **MIT License**, emphasizing maximum freedom for modification and redistribution. Contributors are encouraged to submit PRs with new scripts, bug fixes, or documentation improvements. The system is designed to be lightweight, requiring minimal dependencies, making it easy to run on any Linux or Windows environment.

6.2.2 Community Engagement

Users can connect with maintainers and other contributors via GitHub Issues and Discussions. Bugs, feature requests, and suggestions are tracked openly, allowing for transparency and collaborative problem-solving.



6.3 PR 3 : Y24 Open Source Engineering

6.3.1 Introduction and Purpose

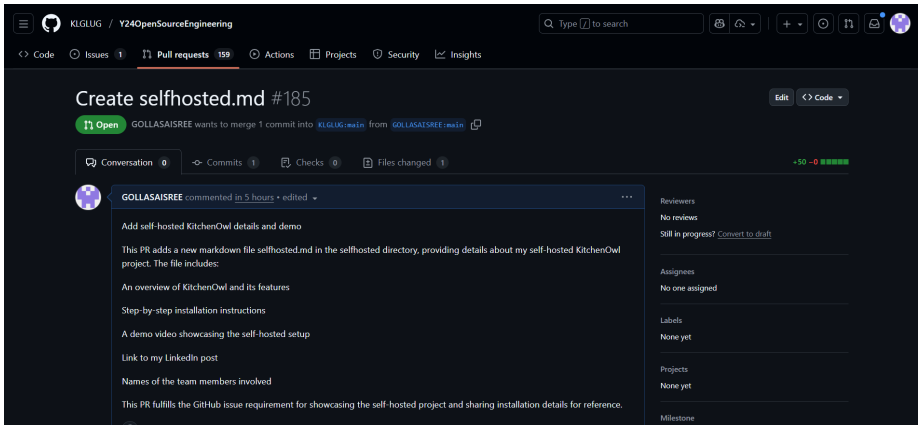
The **Y24 Open Source Engineering** repository provides a structured environment for learning and practicing open-source contributions. Its goal is to help students and beginners understand standard development workflows while contributing to real projects.

6.3.2 Technical Components

The repository includes Python projects, scripts, and configuration files that demonstrate essential software development principles. It uses GitHub workflows to automate tests, documentation checks, and PR validation.

6.3.3 Operation and Usage

Contributors can fork the repository, clone it locally, make changes, and submit PRs for review. These projects serve as hands-on learning tools for Git, GitHub, and collaborative coding practices.



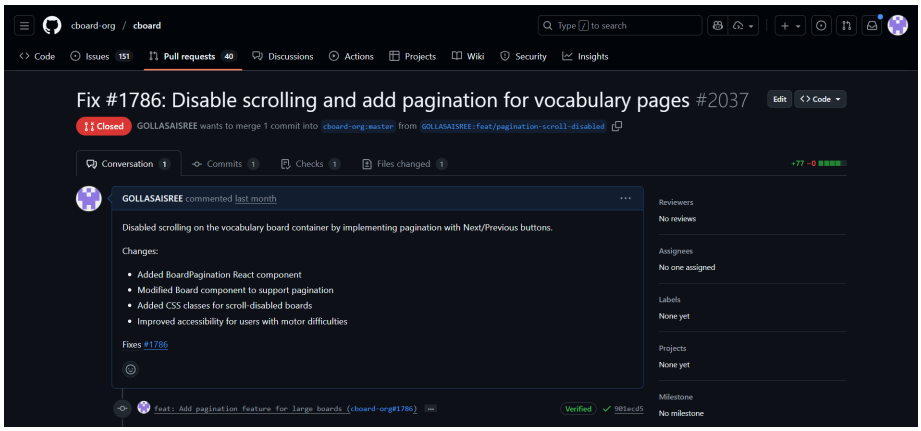
6.4 PR 4 : CBoard

6.4.1 The Core Problem

This PR addresses issues in **CBoard**, an open-source BI dashboard platform. The main difficulty was improving modularity and fixing minor bugs in chart rendering logic, which were causing inconsistent visualization results for end users.

6.4.2 The Solution

The PR fixes the bug by refactoring specific chart components and optimizing the rendering logic for scalability. Additional test cases were added to ensure backward compatibility across different data sources.



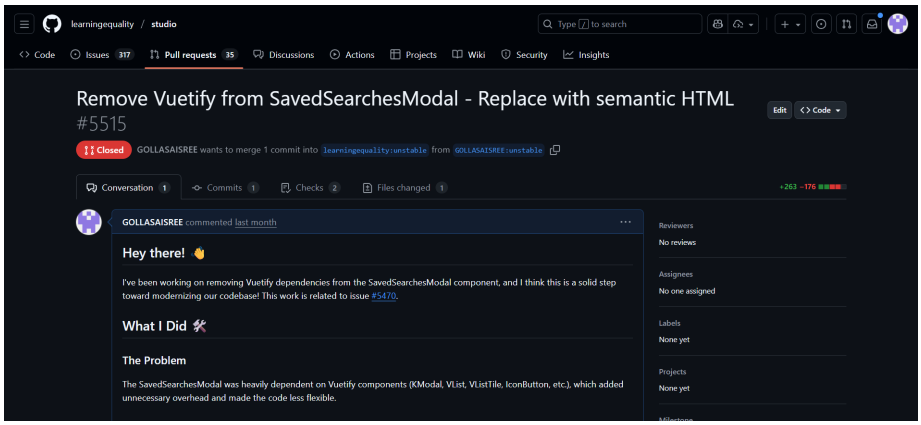
6.5 PR 5 : Studio

6.5.1 The Issue (What was Missing)

In **Studio**, a learning platform, the documentation lacked specific instructions for integrating third-party content and managing database migration scripts. This gap created confusion for new contributors and instructors using Studio for educational purposes.

6.5.2 The Solution (What Was Added)

The PR adds step-by-step documentation and examples for Google Colab integration, database management, and testing, improving usability and onboarding for new users.



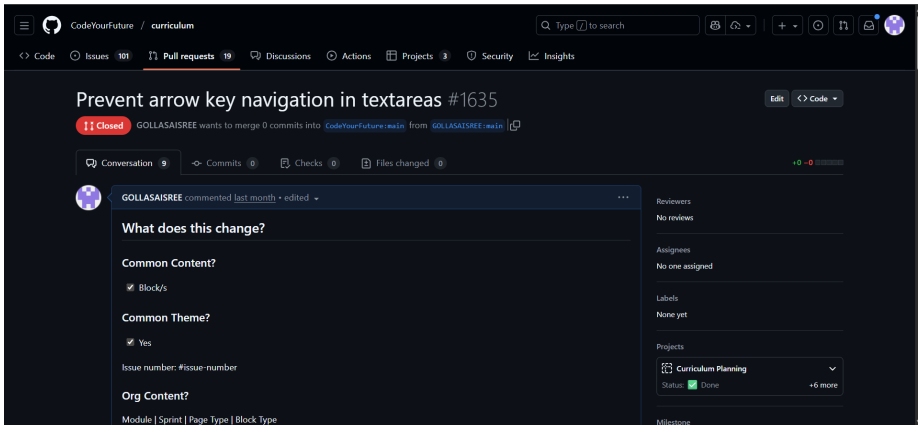
6.6 PR 6 : Curriculum Docs Formatting

6.6.1 The Issue (Formatting Problems)

The PR fixes **documentation formatting errors** in the Curriculum repository, including broken line breaks, inconsistent headings, and code block misalignment. Such errors hinder readability and comprehension.

6.6.2 The Solution

Formatting corrections were applied consistently across Markdown and reStructuredText files. Additionally, duplicate and outdated files were cleaned up, ensuring the documentation is clear, readable, and ready for deployment.



6.7 LinkedIn Post Links

6.7.1 PR:

https://www.linkedin.com/posts/gollasaisree_opensource-firstpr-activity-7399168312963698688-CbNa?utm_source=share&utm_medium=member_desktop&rcm=ACoAAFm4W_cBxfLqhoMdq_Zl92KBpT6NkiBL1sI

6.7.2 Journey Of Open Source:

https://www.linkedin.com/pulse/my-open-source-journey-sai-sree-lea4e

6.7.3 Self Hosted Project:

https://www.linkedin.com/pulse/kitchen-owl-self-hosted-server-sai-sree-gevqe