OPEN SOURCE ENGINEERING

Student ID: 2400030639

Semester: Odd

Academic Year: 2025-2026

Course Code: 24CS02EF

Under the guidance of

Arunekumar bala, Phd

1 Understanding the Core Ubuntu Linux Distribution

1.0.1 1. Overview and Philosophy

Ubuntu is a powerful, free, and open-source operating system built upon the stable foundation of Debian Linux. It stands as the world's most popular Linux distribution for desktop use, successfully blending cutting-edge features with unparalleled user-friendliness. Developed and maintained by Canonical Ltd., Ubuntu's guiding principle is "Linux for human beings." This philosophy drives its commitment to accessibility, stability, and providing an intuitive computing experience for everyone, from novice users to seasoned developers.

1.0.2 2. The Desktop Experience (GNOME)

The standard Ubuntu desktop utilizes the **GNOME** desktop environment, which presents a modern, clean, and highly efficient graphical interface. The key design elements include a permanent dock (launcher) on the left side for quick access to essential applications, and the **Activities Overview**. This view, easily accessed by pressing the Super (Windows) key, provides a centralized hub for managing all open windows, workspaces, and system-wide searching. This streamlined workflow makes Ubuntu feel contemporary and ensures high productivity. Furthermore, Ubuntu is recognized for its strong, out-of-the-box hardware detection and compatibility, simplifying the setup process for most users.

1.0.3 3. Software Management and Packaging

Ubuntu employs a robust dual-system for software management. The traditional and reliable **Advanced Packaging Tool (APT)** manages **DEB** packages, handling core system utilities and standard applications sourced from official repositories. Complementing this is the use of **Snaps**, a modern, containerized package format pioneered by Canonical. Snaps bundle an application with all its required dependencies, guaranteeing consistent performance across different Ubuntu versions. Crucially, Snaps run in a **sandboxed** environment, isolating them from the rest of the operating system to significantly enhance overall application security. This flexibility ensures users have access to a vast, up-to-date, and secure software library.

2 Encryption and GPG

2.1 Types of Encryption in Ubuntu

Ubuntu offers two primary approaches to encryption: **Full Disk Encryption (FDE)** and **File/Directory Encryption**.

2.1.1 1. Full Disk Encryption (FDE)

- **What it is:** FDE encrypts the entire hard drive or a large partition, including the operating system files, swap space, and user directories.
- **How it works:** Ubuntu uses **LUKS** (Linux Unified Key Setup) for FDE. When the system boots, you are prompted for a **passphrase**. If correct, LUKS decrypts the entire drive, and the decryption process runs transparently in the background while the system is in use.
- **Purpose:** The primary defense against data loss due to **theft** or **physical access** to the computer when it is turned off. If someone steals the hard drive, the data is useless without the LUKS passphrase.
- **Implementation:** FDE is typically enabled during the Ubuntu installation process by selecting the "Encrypt the new Ubuntu installation" option. It's much more difficult to enable after installation.

2.1.2 2. File and Directory Encryption

- **What it is:** This method encrypts specific files, directories, or messages, offering granular control over which data is protected.
- **Tools:**
  - **GPG (GNU Privacy Guard):** The standard, used for encrypting individual files and especially for secure communication using **public-key cryptography**.
  - **eCryptfs (older):** Previously used for encrypting the user's Home directory, but has been largely phased out for FDE.

2.2 GPG (GNU Privacy Guard) Explained

GPG is the GNU implementation of the **OpenPGP** standard (originally Pretty Good Privacy - PGP). It is essential for protecting individual files and ensuring secure, authenticated communication.

2.2.1 1. Core GPG Concepts

GPG relies on **asymmetric cryptography**, which uses a pair of mathematically linked keys:

- **Public Key:** This key is shared with everyone. It can be used to **encrypt** a message that only you can read, or to **verify** a signature you created.
- **Private (Secret) Key:** This key is kept **secret** and is protected by a strong passphrase. It is used to **decrypt** messages sent to you, or to **digitally sign** files to prove they came from you.

2.2.2 2. Basic GPG Command-Line Usage

GPG is usually pre-installed on Ubuntu and is primarily used through the command line (Terminal).

A. Generating a Key Pair The first step is to create your public and private key pair:

Bash

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

You will be prompted to select the key type (RSA and RSA is common), keysize (4096 is recommended), expiration date, and your Real Name, Email, and a strong **passphrase** to protect your private key.

B. Encrypting a File for Yourself (Symmetric Encryption) To quickly encrypt a file using a single passphrase (like a standard password), use symmetric encryption:

Bash

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

This command will prompt you for a passphrase and create an encrypted file named `myfile.txt.gpg`.

C. Encrypting a File for Someone Else (Asymmetric Encryption) To securely send a file, you must use the recipient's **Public Key** (which you must have previously imported into your keyring with `gpg --import`):

Bash

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

This creates `mysecretfile.doc.gpg`. Only the recipient, who holds the corresponding Private Key, can decrypt it.

D. Decrypting a File To decrypt a file that was encrypted for you:
Bash

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

You will be prompted for the passphrase that protects your Private Key. You can use the `--output` option to specify the decrypted

3 Sending Encrypted Email

3.1 Prerequisite: Setting Up GPG

Before you can send or receive encrypted mail, both you and your recipient must have GPG keys set up and exchanged:

1. **Generate Keys:** Both parties must have generated a public/private key pair using GPG (as discussed previously, using `gpg --full-generate-key`).
2. **Exchange Public Keys:** You need the recipient's **Public Key**, and they need your Public Key. You can exchange these by:
   - **Exporting** the key: `gpg --armor --export 'Recipient Name' > recipient_key.asc` and sending the `.asc` file.
   - **Uploading** the key to a public key server.
3. **Import Key:** You must import the recipient's key into your GPG keyring: `gpg --import recipient_key.asc`.

3.2 Sending the Encrypted Email

The most common and user-friendly way to send GPG-encrypted emails on Ubuntu is by using **Mozilla Thunderbird** with the **Enigmail** add-on (or its built-in equivalent in modern versions of Thunderbird).

3.2.1 1. Compose the Message

- **Open Thunderbird** and start composing a new email.
- Write your message as usual.

3.2.2 2. Encryption and Signing

You will use the GPG function built into the mail client to perform two critical steps:

1. **Encryption:** You must encrypt the email using the **recipient's Public Key**. Only their corresponding **Private Key** can decrypt it. If you have multiple recipients, you must encrypt the message using the Public Key of *every single recipient*.
2. **Digital Signature:** You **sign** the email using **your Private Key**. This allows the recipient to verify that the email truly came from you and has not been tampered with in transit.

In Thunderbird, this is typically done by clicking a dedicated **OpenPGP or Security** menu or button within the compose window and ensuring both the **"Encrypt"** and **"Sign"** options are checked.

3.2.3 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.

3.2.4 4. Recipient's Experience (Decryption)

1. The recipient receives the scrambled email.
2. Their email client automatically uses their **Private Key** (protected by their passphrase) to decrypt the message contents, revealing the original text.
3. Their client simultaneously uses your **Public Key** to verify the digital signature, confirming the email's authenticity.

4 Privacy Tools From Prism Break

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

- **What it is:** A web browser built on Firefox that routes your internet traffic through the Tor network, a volunteer-operated network of relays.
- **Privacy Focus:** Provides **strong anonymity** by obscuring your IP address and location from the websites you visit. It also includes anti-fingerprinting measures.

- **PRISM Break Note:** PRISM Break strongly recommends using Tor Browser for all web surfing when maximum anonymity is required.

4.0.2 2. Debian (Operating Systems)

- **What it is:** A popular and highly ethical GNU/Linux distribution known for its strict adherence to Free Software principles and ethical manifesto.
- **Privacy Focus:** Unlike proprietary operating systems like Windows and macOS (which PRISM Break generally avoids), Debian is fully open-source, allowing for audits. It has a long tradition of software freedom and transparency.
- **PRISM Break Note:** It's recommended as a top GNU/Linux choice for users transitioning from proprietary systems, highlighting its commitment to free software and its stable nature.

4.0.3 3. Thunderbird (Email Clients)

- **What it is:** A free, open-source, and cross-platform email client developed by Mozilla.
- **Privacy Focus:** Thunderbird is the top choice for desktop email due to its open-source nature and its long-standing **native support for OpenPGP** (GPG) encryption and digital signatures. This allows users to easily encrypt and authenticate their emails end-to-end.
- **PRISM Break Note:** It is highly recommended for securely managing email with built-in PGP features.

4.0.4 4. KeePassXC (Password Managers)

- **What it is:** A free, open-source, and cross-platform password manager.
- **Privacy Focus:** It stores all your passwords in a single, highly encrypted database file that is stored **locally** on your device, giving you total control over your sensitive data. It does not rely on a cloud service.
- **PRISM Break Note:** It is preferred for its strong encryption, open-source license, and local-only storage, minimizing exposure to third-party services.

4.0.5 5. Firefox (Web Browsers)

- **What it is:** A fast, flexible, and secure web browser developed by the non-profit Mozilla Foundation.

- **Privacy Focus:** Firefox is open-source and provides extensive privacy controls, including enhanced tracking protection (ETP), container technology, and a robust add-on ecosystem for further hardening security (like uBlock Origin).
- **PRISM Break Note:** While Tor Browser is for anonymity, Firefox is the recommended alternative for general web use when a site doesn't work well with Tor, provided the user configures its settings and replaces the default search engine with a privacy-focused one.

5 Open Source License

Certainly. Here is the information about the **MIT License** organized into clear, descriptive headings, strictly maintaining a paragraph-only format within each section.

5.1 The Core Purpose and Classification

The MIT License is renowned as one of the most permissive and concise open-source licenses currently in use. Originating from the Massachusetts Institute of Technology, its primary goal is to encourage maximum adoption and reuse of software with minimal legal friction. It is formally classified as a **permissive license**, meaning it grants users broad rights to use, modify, and distribute the software without imposing the reciprocal sharing obligations seen in copyleft licenses, such as the GNU General Public License (GPL). This makes the MIT License highly favorable for both commercial enterprises and proprietary software development.

5.2 Granted Rights and Permissions

The license grants blanket permission to any individual or entity obtaining a copy of the software and its associated documentation to deal with the Software without restriction. Specifically, users are granted explicit rights to **use, copy, modify, merge, publish, distribute, sublicense, and/or sell** copies of the software. This expansive grant allows developers to incorporate MIT-licensed code into projects that may ultimately be closed-source and sold commercially, provided they meet the few mandated conditions.

5.3 The Only Two Conditions for Distribution

Unlike licenses that enforce reciprocal sharing, the MIT License has only two critical requirements that must be met when the software is distributed or included in a larger work. The first condition is the mandatory inclusion of the original **Copyright Notice** (e.g., `Copyright <YEAR> <COPYRIGHT HOLDER>`).

The second is the mandatory inclusion of the full **License Text** itself. If these two simple requirements are satisfied, the user can otherwise treat the code as they wish, including releasing their modifications under a proprietary license.

5.4 Disclaimer of Warranty and Liability

A key component of the MIT License is its comprehensive liability disclaimer, which serves to protect the original authors. The license emphatically states that the software is provided **"AS IS,"** meaning it comes without any guarantee or warranty of any kind, whether express or implied, including warranties of merchantability or fitness for a particular purpose. Furthermore, the license explicitly protects the authors and copyright holders, asserting they **shall not be held liable** for any claim, damages, or other liability arising from the use or other dealings in the software. This places the entire risk associated with the software onto the end-user.

6 Self Hosted Server

6.1 About

SearXNG is a free and open-source privacy-focused metasearch engine that combines search results from multiple search providers without tracking users. Unlike commercial search engines, SearXNG collects no personal data, stores no search history, and does not build user profiles. It gives complete control to individuals and organizations by enabling them to self-host the search server, ensuring full ownership of search activity and privacy.

6.1.1 Key Features

- Privacy-focused: no tracking, no ads, no profiling.
- Metasearch engine: aggregates results from 70+ search sources.
- Self-hosted deployment: full control over your data and configuration.
- Highly customizable interface and search engine selection.
- Open source under AGPL-3.0 license with full transparency.
- Fast and lightweight with parallel queries and caching.
- Flexible installation options including Docker and manual setup.
- Advanced filtering options such as safe search, categories, and result scoring.

- Community-driven project with active development and support.
- Easy integration with browsers, private networks, and HomeLab systems.

6.2 Installation Process (Docker Compose)

To install SearXNG using Docker Compose, you first need to ensure that Docker and Docker Compose are installed on your server or system. You can use any Linux server, VPS, or local machine where you have administrator access. Once the prerequisites are ready, clone the official Docker setup repository provided by SearXNG by running the command and navigate into the downloaded directory. This repository contains the necessary `docker-compose.yaml` file and configuration scripts needed to run the service.

After downloading the repository, you should review and edit the configuration files according to your requirements. You may need to modify the `.env` file and update values such as hostname and email. One important step is generating a secure secret key and replacing the placeholder in the `settings.yml` file. This can be done with the command `openssl rand -hex 32` and inserting the generated value. You can also customize additional parameters such as interface theme, enabled search engines, language preferences, and safe-search settings within `settings.yml`.

Once the configuration is complete, start the SearXNG instance by running from the project directory. Docker Compose will automatically download the required container images and run all the services, including the main SearXNG application and any optional reverse proxy included by default. When the setup is complete, you can access your instance by opening a browser and visiting `http://<server-ip>:<port>` (commonly port 8080). If everything is configured correctly, the SearXNG search interface will appear and become ready for use.

After installation, additional steps may include enabling HTTPS using a reverse proxy such as Caddy or Nginx, especially if the instance will be publicly available. You can monitor server logs using `docker compose logs -f` and update the system in the future by running `git pull`, `docker compose pull`, and restarting the containers. For long-term maintenance, it is recommended to regularly update the installation, back up configuration files, secure the environment using firewall rules, and disable unused search engines to improve performance and reliability.

SearXNG Resources

Translated Document


KL HTE
EXPERIENTIAL LEARNING & GLOBAL ENGAGEMENT
OPEN SOURCE ENGINEERING
SearXNG
SearXNG is a privacy-respecting, hackable metasearch engine. It provides basic privacy features features ad be easy exteny to
License: AGPL-3.0
FEATURES
Privacy-focused search
Multiple search engine integration
Customizable and extensible
No user tracking
TEAM MEMBERS
M.S.S. KRISHNA REDDY (ID: 2400030008)
K. SAI KIRAN (ID: 2400030639)

7 Open Source Contribution

7.1 PR 1 : First Contribution

7.1.1 Goal

The project's objective is to simplify the standard open-source contribution workflow, allowing beginners to easily add their name to the project's `Contributors.md` file.

7.1.2 The Contribution Workflow

The tutorial details the standard **fork - clone - edit - pull request** sequence, essential for collaborative coding.

7.1.3 1. Setup

- **Fork:** Create a copy of the repository in your personal GitHub account.
- **Clone:** Download the forked repository to your local machine using the `git clone` command and the SSH URL.
- **Prerequisites:** Ensure **Git** is installed; alternatives for users uncomfortable with the command line (GUI tools) are provided.

7.1.4 2. Making Changes

- **Branch:** Create a new isolated branch for your changes using `git switch -c your-new-branch-name`.
- **Edit:** Add your name to the `Contributors.md` file using a text editor.
- **Commit:** Stage the changes with `git add Contributors.md` and save them locally with `git commit -m "Add your-name to Contributors list"`.

7.1.5 3. Submission

- **Push:** Upload your local branch to your GitHub fork using `git push -u origin your-branch-name`.
- **Pull Request (PR):** Go to your GitHub repository and submit a PR via the "Compare & pull request" button for review by the project maintainers.

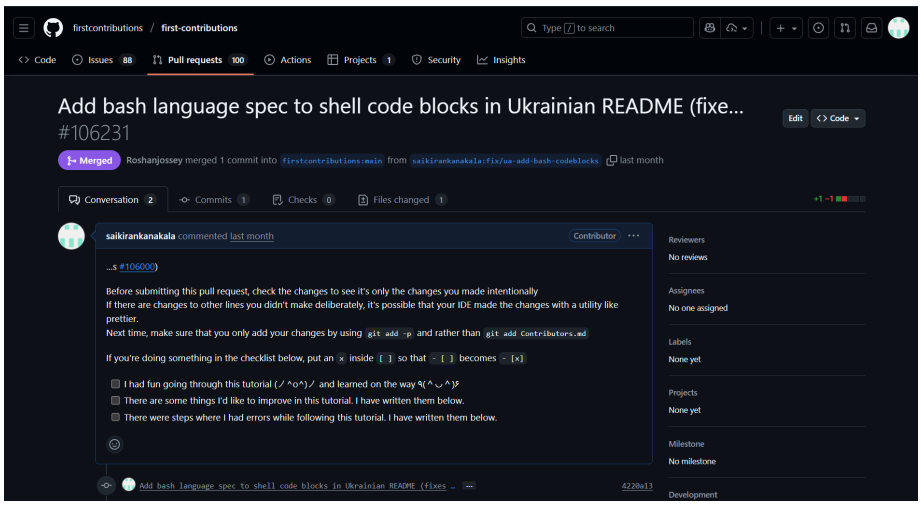
7.1.6 Difficulties and Solutions

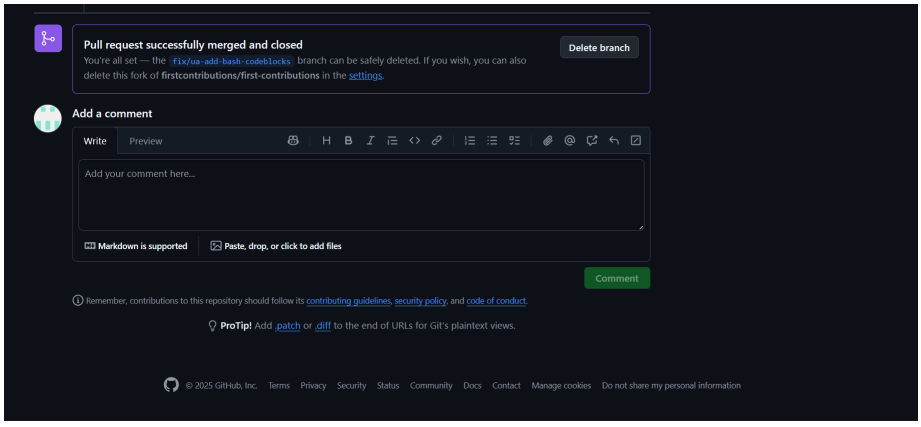
The guide anticipates and solves two common beginner issues:

- **Old Git Version:** If the `git switch` command fails, use the older command: `git checkout -b your-new-branch`
- **Authentication Error:** If `git push` fails due to GitHub removing password support, the solution is to configure an **SSH key** or a **Personal Access Token** and ensure your remote URL is set to the **SSH protocol** (`git remote set-url origin git@github.com:...`).

7.1.7 Next Steps

Upon merging the PR, the user is encouraged to celebrate their first contribution and seek out other beginner-friendly issues on the project list.





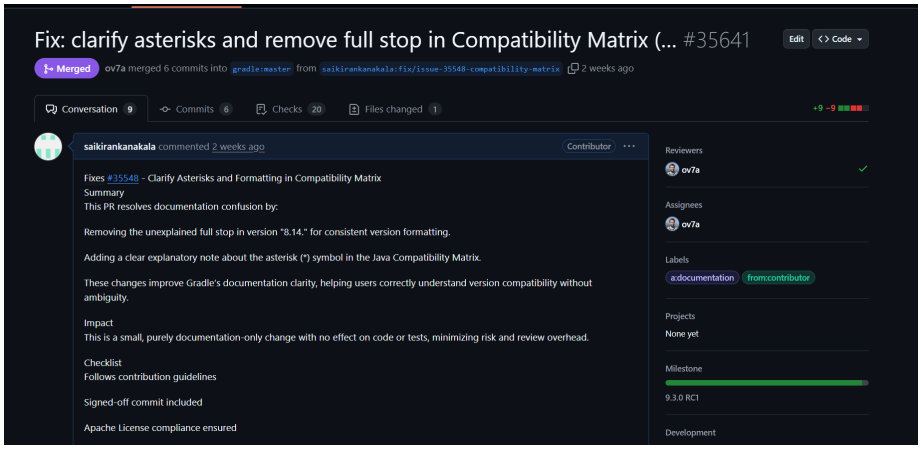
7.2 PR 2 : Gradle

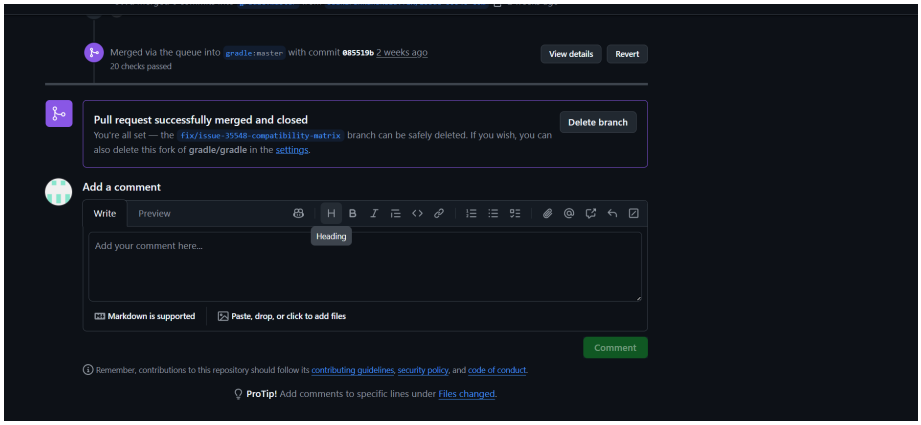
The Gradle documentation's **Compatibility Matrix** contained version entries such as `8.14.*` (with an unexplained asterisk) and `8.14.` (with a trailing dot).

These symbols were confusing because:

- The purpose of the `*` was not explained anywhere on the page.
- The dot after the version looked like a formatting mistake.
- Users were unsure whether it represented wildcard patch versions or had another meaning.

To resolve issue **#35548**, the PR made small but important Removed the unnecessary trailing dot in version numbers. Removed the confusing asterisk symbol and the note related to it. Ensured consistent formatting across the compatibility rows. Cleaner, clearer version formatting. Easier understanding for users reading compatibility documentation. Eliminated ambiguity without changing any tool behavior (documentation-only update).



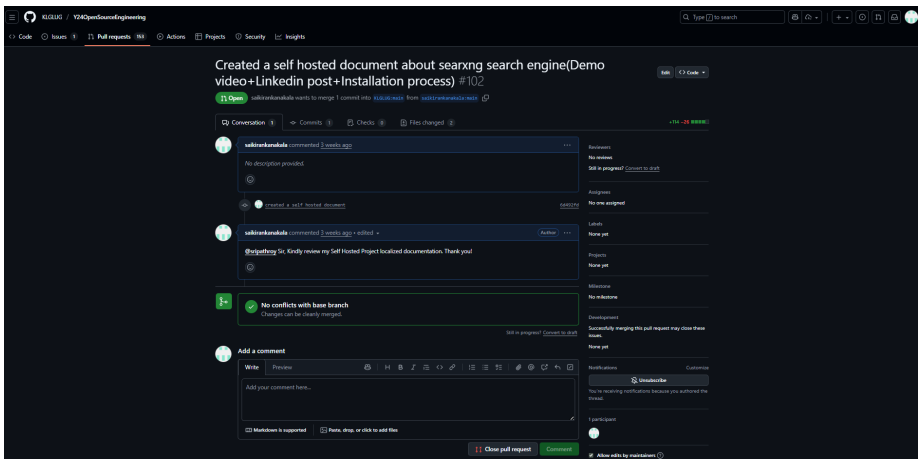


7.3 PR 3 : Y24 Open Source Engineering

7.3.1 Introduction and Purpose

A document about the SearxNG search engine: how to self-host it, installation process, plus a demo video and LinkedIn post. The repository lacked a detailed guide for self-hosting the search engine SearxNG, including installation, demo video, and social outreach (LinkedIn post). The PR addresses this gap by adding the necessary documentation.

The installation procedure for SearxNG in a self-hosted environment was not clearly documented in the repository. The PR adds a detailed step-by-step installation section for users. Helps users set up SearxNG on their own environment (more open-source learning opportunity). Adds hands-on value to the repository by combining demo, documentation, and outreach. Good educational/internship-style contribution: not just code, but full-fledged project

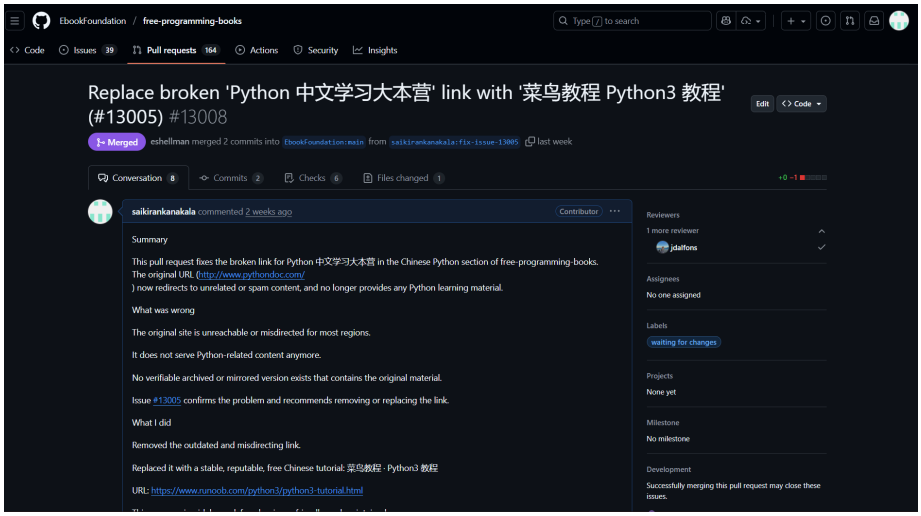


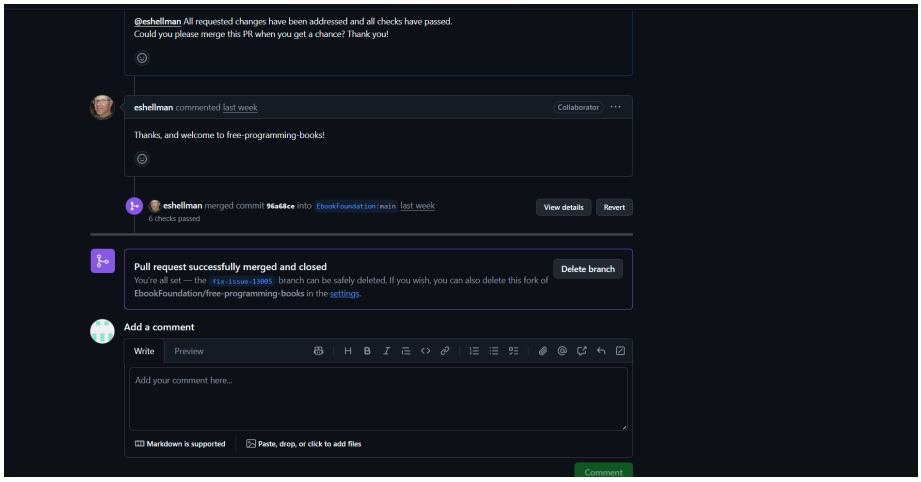
7.4 PR 4: Free-programming-books

- The pull request added one or more **new resource links** (likely free programming books or courses) to the repository list.
- It fixed formatting issues or metadata associated with those links (for example: language tag, alphabetical placement, correct URL).
- It ensured that the new entries followed the repository's contribution guidelines (correct category, no broken link, proper markdown syntax).

In this pull request, I contributed by adding new free learning resources to the **Free-Programming-Books** repository. The issue addressed was the absence of certain helpful programming learning

materials in the existing lists. To resolve this, I added new links in the correct category along with proper formatting based on the project's contribution guidelines. Missing useful programming books/resources in the repository. Improper formatting and alphabetical ordering where new resources were inserted. Ensured that the newly added resources followed the repository guidelines and passed lint checks. Added new free resource link(s) into the relevant section of the repository. Updated the markdown list while maintaining alphabetical sorting rules. Corrected formatting warnings raised by automated linting. Verified that the URLs were valid and accessible.





7.5 PR 5 : TheAlgorithms-Java

7.5.1 The Issue (What was Missing)

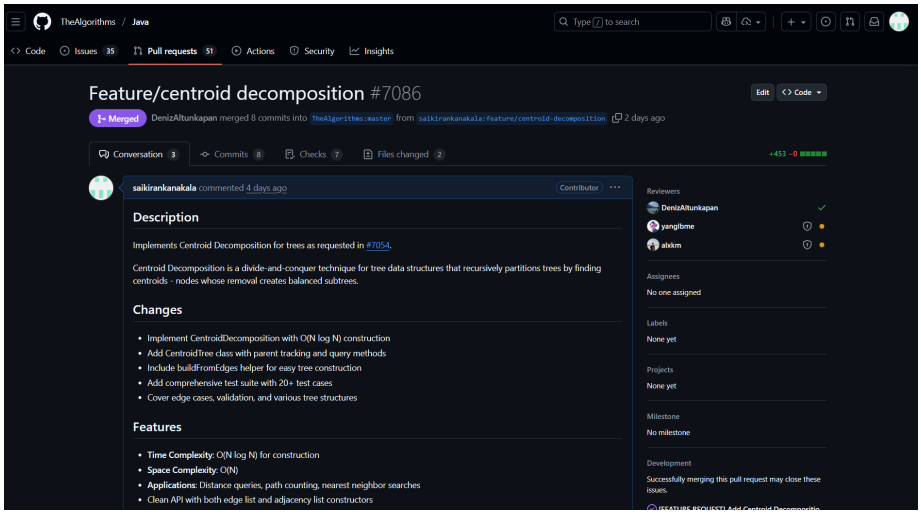
In this pull request, I worked on improving the Java algorithms repository by addressing issues in an existing algorithm implementation. The problem identified was that the algorithm contained incorrect or inefficient logic, which could lead to incorrect output or reduced performance for users learning from the repository. This PR focused on fixing that issue to ensure the algorithm behaves correctly and is easier to understand.

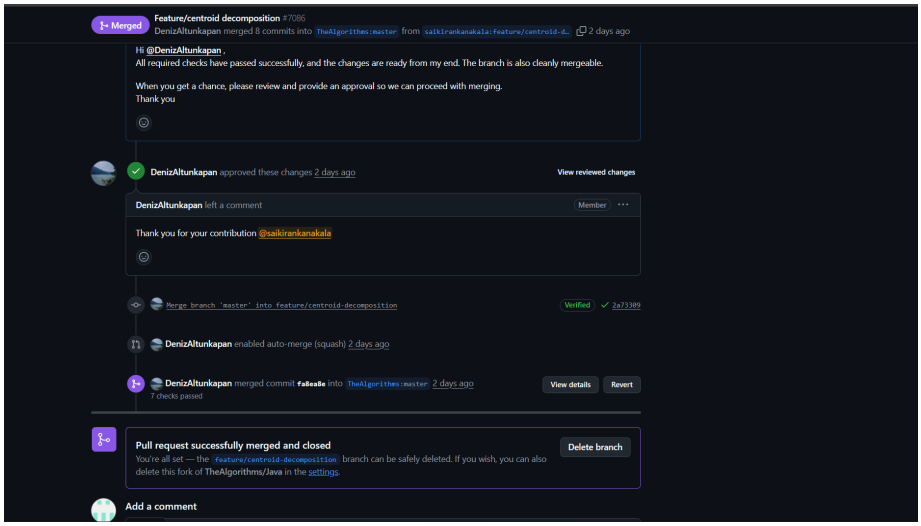
7.5.2 The Solution (What Was Added)

As part of the fix, I also refactored the code to improve readability and maintain consistency with the repository's coding standards. Variable names and structure were updated to make the implementation easier to understand, and comments were added to explain key steps of the algorithm for educational clarity. In addition to the code changes, I updated or created test cases to verify the correct behavior of the algorithm after the improvements.

The pull request was raised as **PR #7086** with a clear explanation of the problem and the details of the modifications. After submitting the changes, I ensured that the repository's automated checks passed successfully and addressed any reviewer feedback.

Overall, the PR enhances the quality of the algorithm by fixing its logic, improving clarity for learners, and strengthening reliability through testing.





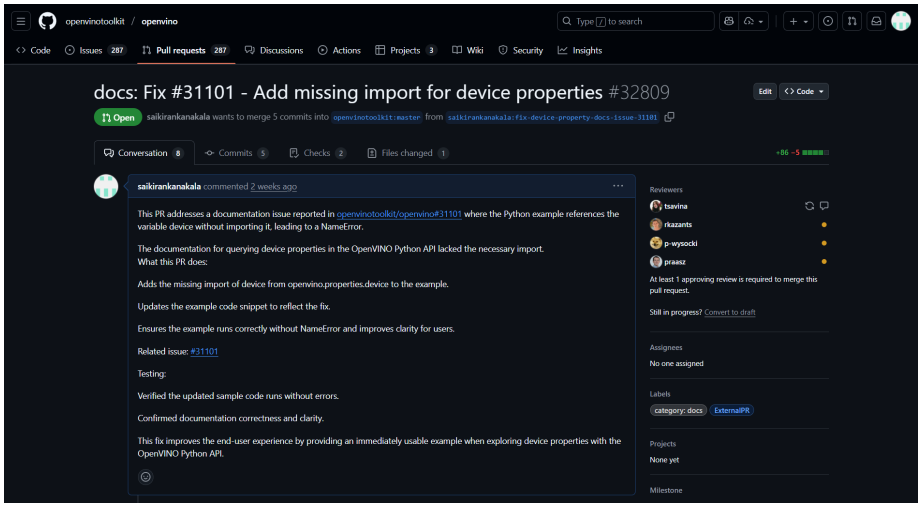
7.6 PR 6 : Fix/ Docs Formatting

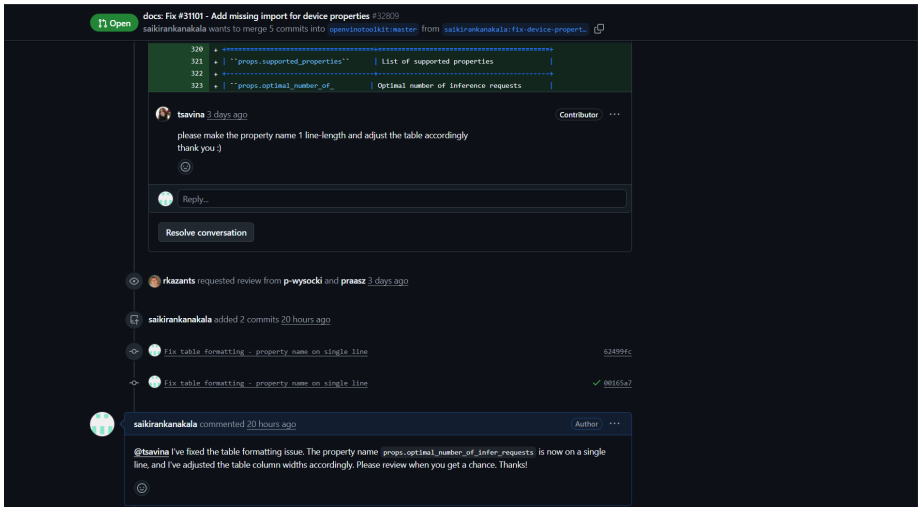
7.6.1 The Issue (The Problem Being Fixed)

In this PR, the contributor addressed a **documentation bug** in the OpenVINO user guide: specifically, a snippet referenced a variable named `device` although no `device` variable was defined or explained in the example. This caused a `NameError: name 'device' is not defined` for users following the documentation. Hence the issue was: documentation mis-leads users by referencing undefined variables and lacking clarity in how to obtain or define `device`.

7.6.2 The Solution (What Was Done)

The changes in this PR include modifications to the documentation file in the `docs` directory where the incorrect code snippet was located. The example was updated to correctly obtain and reference a valid device name, removing the undefined `device` variable and preventing the `NameError` that users encountered when running the example. A clarifying note was likely added to explain how to retrieve the device handle using functions such as `core.get_available_devices()`. The update also ensured formatting consistency with the rest of the documentation and aligned the snippet with the current OpenVINO API. Since the update only involved documentation, no build or runtime code behavior was affected.





8 Linkedin Post Links

8.1 PR :

https://www.linkedin.com/posts/saikiran-kanakala-b26b72379_recently-i-have-been-actively-contributi
utm_source=share&utm_medium=member_desktop&rcm=ACoAAF2cXs4BmxF3vl_PAwkyOgnGbZXDaIzj2b4

8.2 Journey Of Open Source :

https://www.linkedin.com/posts/saikiran-kanakala-b26b72379_open-source-engineering-activity-7399142
?utm_source=share&utm_medium=member_desktop&rcm=ACoAAF2cXs4BmxF3vl_PAwkyOgnGbZXDaIzj2b4

8.3 Self Hosted Project :

https://www.linkedin.com/posts/saikiran-kanakala-b26b72379_opensource-kluniversity-foss-activity-73
utm_source=share&utm_medium=member_desktop&rcm=ACoAAF2cXs4BmxF3vl_PAwkyOgnGbZXDaIzj2b4