OPEN SOURCE ENGINEERING

Name : POLA AKSHITH
Student ID : 2400030388
Semester : ODD
Academic Year : 2025-2026
Course Code : 24CS02EF

Submitted to:

Dr. Sripath Roy
Department of Computer Science and Engineering
KL University

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

LocalSend Server is an **open-source, self-hosted file transfer system** designed to provide a fast, private, and reliable method for sharing files across devices within the same local network. Built as a lightweight and privacy-focused alternative to cloud-based transfer tools, it ensures that all data stays on the user's network, giving complete **control over security and privacy**. The project is licensed under the permissive **MIT License**, allowing full transparency, modification, and customization.

6.1.1 Key Features

- **Local Network Transfers:** Allows devices to automatically discover each other on the same LAN and exchange files without requiring internet access or cloud services.
- **Privacy-Focused:** Since all transfers occur locally, no external servers or third-party services are involved, ensuring complete data security.
- **Simple UI:** Provides an intuitive interface with options to **Send**, **Receive**, and manage **Settings** in a minimal and user-friendly layout.
- **Cross-Platform Support:** Works seamlessly across Linux, Windows, macOS, and mobile platforms.

- **Lightweight Operation:** Designed to run smoothly even on low-resource devices like Raspberry Pi or older systems.
- **Flexible Self-Hosting:** Can be deployed easily using Docker, system services, or as a standalone application.

6.2 Installation Process (Docker Compose)

The recommended and simplest method to self-host **LocalSend Server** is through **Docker Compose**. This setup ensures efficient management of the LocalSend backend as a containerized service. Before beginning, ensure that **Docker** and **Docker Compose** are installed on your system.

The installation starts by creating a working directory and adding a `docker-compose.yml` configuration file. This file defines the LocalSend container, port mapping, and storage options.

```
version: "3.8"

services:
  localsend:
    image: localsend/localsend
    container_name: localsend-server
    ports:
      - "3000:3000"
    volumes:
      - ./data:/app/data
    restart: unless-stopped
```

Once the configuration file is ready, the next step is to **pull the image and start the service** using:

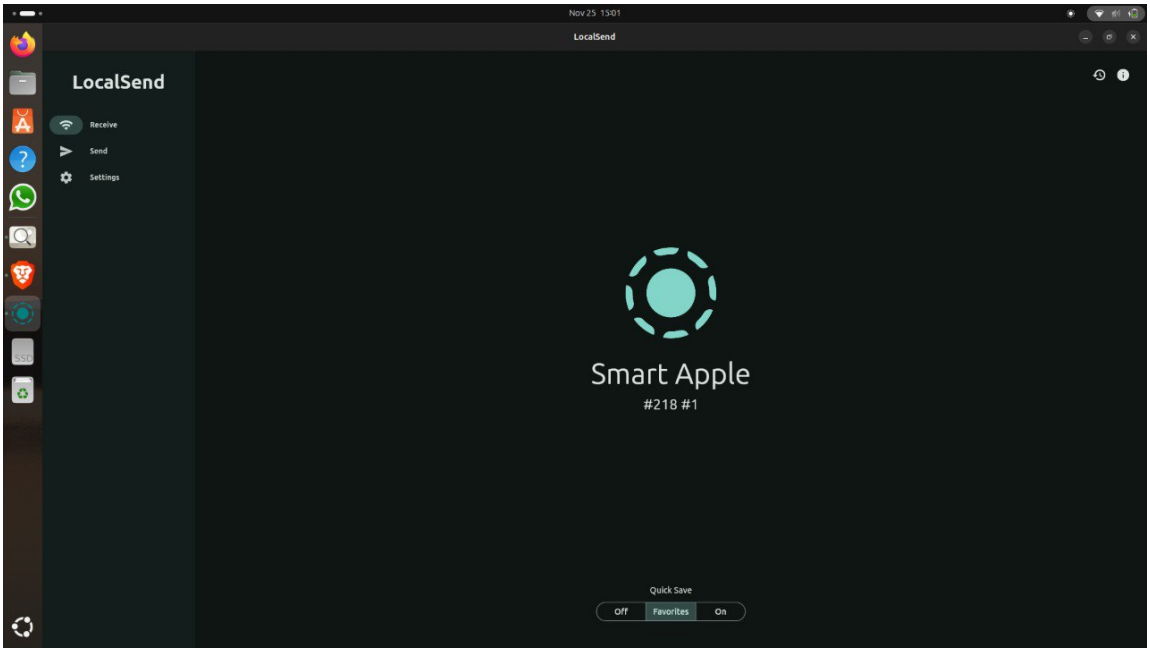
```
docker compose up -d
```

Docker Compose will automatically download the LocalSend image and initiate the container in `-d` detached mode.

After deployment, LocalSend Server becomes accessible in the browser via:

`http://localhost:3000`

The dashboard allows users to manage transfers, view device identity (e.g., **Smart Apple**), and configure file handling preferences. No additional login credentials are needed, making it easy to use across a local network.



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.

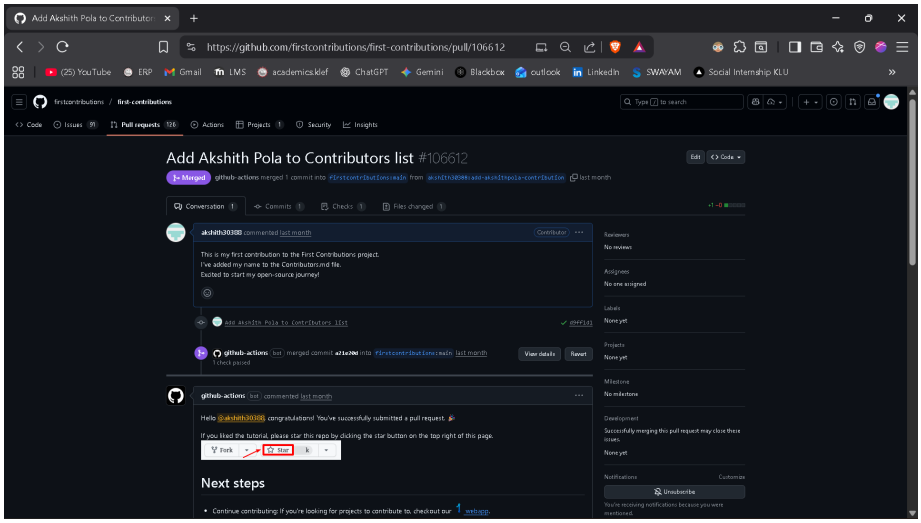
7.1.6 Difficulties and Solutions

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, configure an **SSH key** or a **Personal Access Token**. 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 : Free Code Camp

7.3 FreeCodeCamp Pull Request Contribution

FreeCodeCamp is a widely recognized **open-source learning platform** that provides extensive resources for web development, programming, and computer science. As part of my open-source journey, I contributed to one of its core curriculum repositories by submitting a pull request aimed at improving the clarity and formatting of an educational lesson. This contribution reflects FreeCodeCamp's mission of community-driven enhancement, where learners help improve content for millions of users worldwide.

7.3.1 Summary of Contribution

The pull request focused on refining the lesson titled: `Working with Regular Expressions` located within the FreeCodeCamp curriculum blocks. My contribution involved enhancing the example explanation and improving the readability of the instructional content.

Key improvements included:

- **Clarified the example** in the regular expressions lesson for better understanding.
- **Improved formatting** to maintain consistency with the rest of the curriculum.
- **Corrected minor grammar and wording issues** for cleaner presentation.

This PR was successfully merged into the `main` branch by a project maintainer, marking a valuable milestone in my open-source learning path.

7.3.2 Repository and Licensing

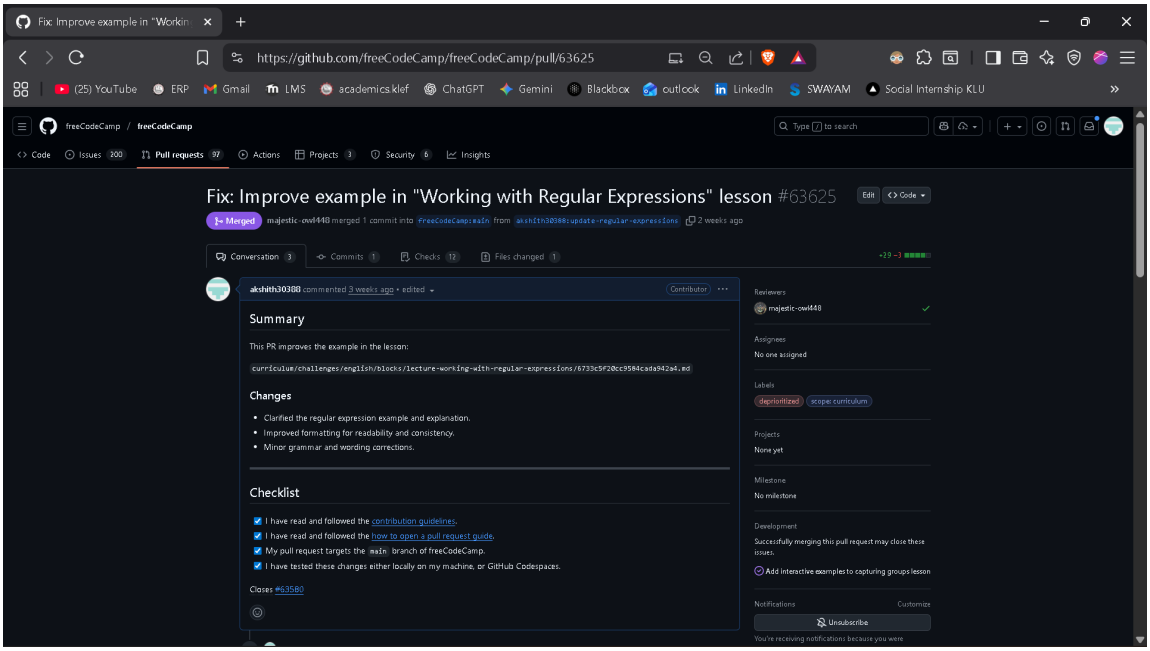
FreeCodeCamp operates under the **BSD-3-Clause License**, a permissive license that allows users to view, modify, and distribute the curriculum. The project follows strict contribution guidelines, which include forking the repository, creating a new branch, making improvements, and submitting a pull request for review. All contributions undergo automated checks and manual review before merging, ensuring high-quality educational content.

7.3.3 Community and Review Process

The FreeCodeCamp community actively participates through GitHub discussions, issue tracking, curriculum feedback, and contributions. My PR was reviewed by a maintainer, who verified the changes and approved the merge. The checklist ensured compliance with the following:

- I followed the official **contribution guidelines**.
- I reviewed the **How to Open a Pull Request** documentation.
- I targeted the correct branch (the `main` branch).
- I tested my changes locally before submitting.

This contribution helped me understand how professional open-source review workflows operate—particularly code review, automated checks, and collaborative refinement.



7.4 PR 3 : Y24 Open Source Engineering

7.4.1 Introduction and Purpose

7.5 About LocalSend

LocalSend is a lightweight **open-source, self-hosted file transfer tool** designed to enable fast, secure, and offline sharing of files across devices within the same local network. Its primary purpose is to provide users with full **data privacy and control**, eliminating dependency on external cloud-based transfer services. LocalSend is especially useful for classrooms, small teams, and home users who want a simple and reliable method to share data without exposing it to the internet.

7.5.1 Technical Components

A LocalSend setup typically runs on any device capable of executing a Linux-based operating system such as **Ubuntu**, **Debian**, or **Fedora**. Essential components include sufficient **storage** for received files (SSD or HDD), stable network **connectivity** via Wi-Fi or Ethernet, and a standard **power supply**. The application relies on modern networking libraries that allow devices to automatically discover each other on the same LAN.

LocalSend's backend architecture is built around peer-to-peer communication principles, ensuring that no external servers are required. Its browser interface enables users to interact with the service through any modern web browser.

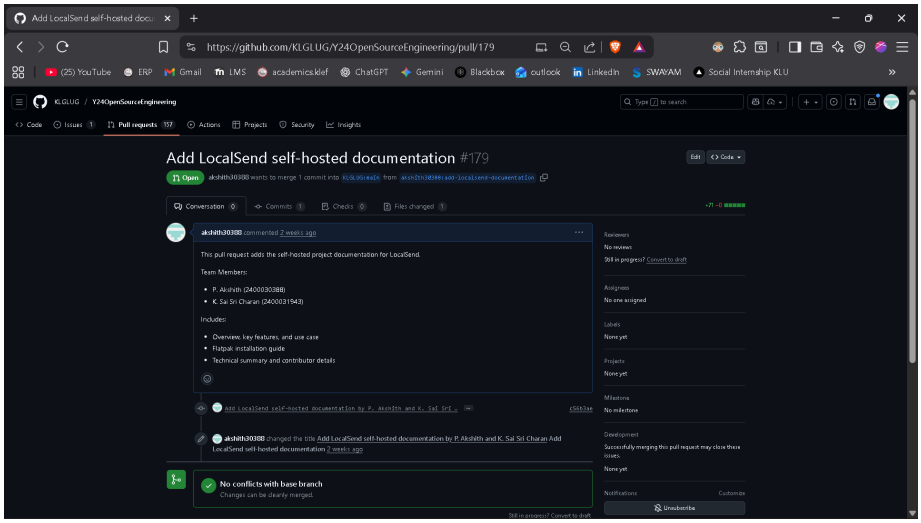
7.5.2 Operation and Usage

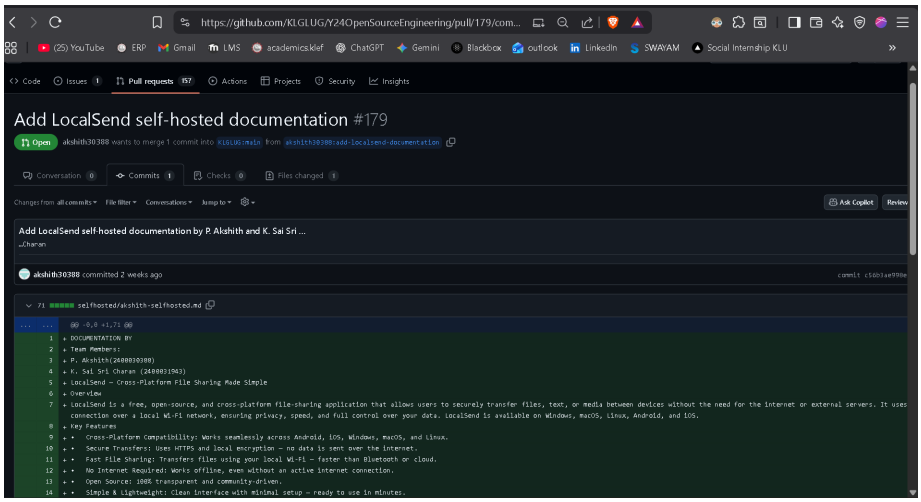
LocalSend operates by running a lightweight local server process that broadcasts its device identity across the LAN. Other devices detect this broadcast and can initiate file transfers using the **Send** or **Receive** features. All transfers occur securely within the local network, making the process both private and efficient.

Users access the LocalSend dashboard using a web browser through:

```
http://localhost:3000
```

This interface displays the device name (for example, **Smart Apple**) along with transfer options. LocalSend is ideal for use cases such as **private file sharing**, offline communication in labs or classrooms, transferring media between devices, or creating a simple home-based data exchange system. It offers complete user control and provides meaningful practical experience in self-hosting, networking, and local server management.





7.6 PR 4: JavaScript Interview Questions

Here is a short description of the Pull Request, organized into paragraphs with headings.

7.7 JavaScript Interview Questions Pull Request

The **JavaScript Interview Questions** repository is a community-driven collection of modern JavaScript questions and explanations aimed at helping learners and developers prepare for technical interviews. As part of my open-source contributions, I added a new interview question to this repository, expanding its coverage of advanced JavaScript concepts and improving the learning material for future contributors and readers.

7.7.1 Summary of Contribution

My Pull Request introduced a new question to the main README:

477. What is `structuredClone` and how is it used for deep copying objects?

The contribution explains the `structuredClone()` method — a modern JavaScript API that reliably performs deep copying, including support for Maps, Sets, Dates, TypedArrays, and even circular references. The explanation follows the repository's established formatting rules and includes a clear example to demonstrate usage.

Key highlights of the contribution:

- Added a new interview question aligned with the existing question format.
- Followed the repository's content style and structural guidelines.
- Included a practical example demonstrating how `structuredClone()` works.
- Enhanced the repository's coverage of modern JavaScript features.
- Included Table of Contents entry support where applicable.

The PR was reviewed and merged by the repository owner, acknowledging the importance of covering newer JavaScript APIs that are increasingly common in interviews.

7.7.2 Repository and Licensing

The repository is publicly available on GitHub and licensed under the **MIT License**, allowing users to freely use, modify, and distribute its content. Contributions must follow the standard GitHub workflow involving forking, creating a branch, making edits, and opening a Pull Request for review.

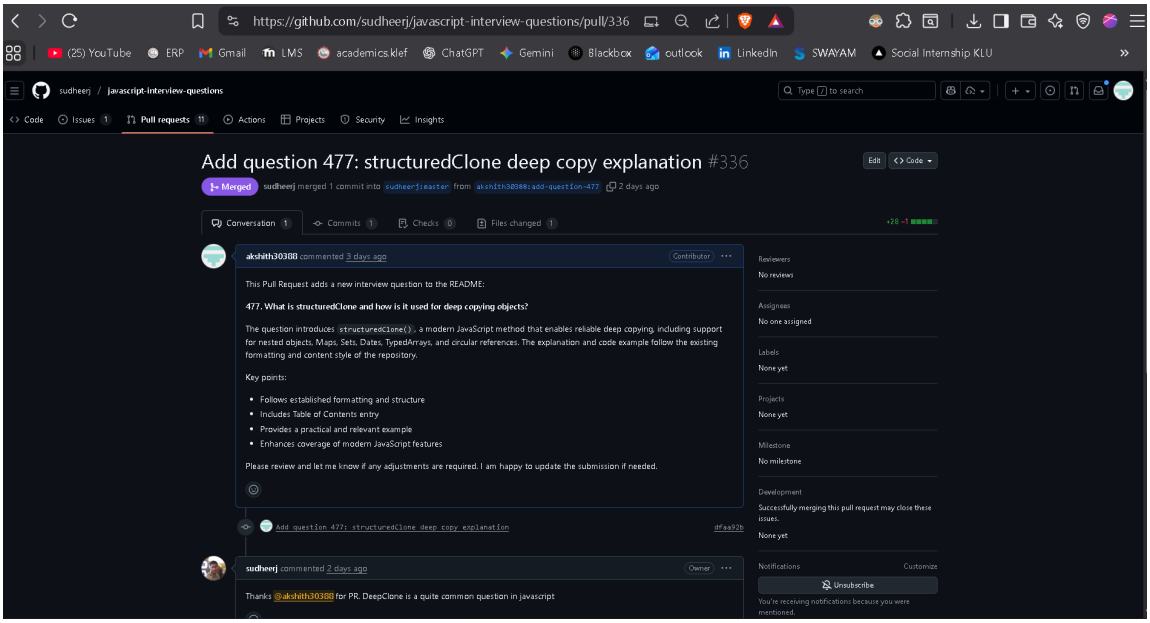
All submissions are checked for formatting, clarity, and overall usefulness before being merged to ensure quality learning material for the community.

7.7.3 Community and Review Process

The maintainers encourage contributions from developers worldwide. My contribution underwent community visibility and was merged after a quick review by the repository owner. The supportive review environment makes this project ideal for new contributors who want to practice real-world Git workflows.

- Contribution followed formatting and style conventions.

- The PR was tested and validated before submission.
- Reviewer acknowledged the relevance of the new question.



7.8 PR 5 : Zero To Mastery (ZTM)

7.9 Zero To Mastery (ZTM) Contribution Pull Request

The **Zero To Mastery (ZTM) Start-Here Guidelines** repository is a beginner-friendly open-source project designed to help newcomers understand the fundamentals of contributing to open-source software. It provides clear instructions for creating a pull request, following contribution rules, and becoming part of the ZTM global developer community. As part of my open-source learning experience, I contributed to this repository by adding my GitHub username to the official contributors list.

7.9.1 Summary of Contribution

My pull request added my GitHub username **akshith30388** to the `CONTRIBUTORS.md` file, fulfilling the standard onboarding requirement for first-time contributors in the ZTM community.

Key actions performed:

- Added my username to the contributors list in the appropriate alphabetical section.

- Ensured no other entries were modified during the update.
- Followed all contribution guidelines as mentioned in the project README.
- Submitted the PR from a dedicated branch (`add-akshith30388`).

This PR represents an important first step in the ZTM open-source workflow, helping contributors practice Git, GitHub branching, and pull request etiquette.

7.9.2 Repository and Licensing

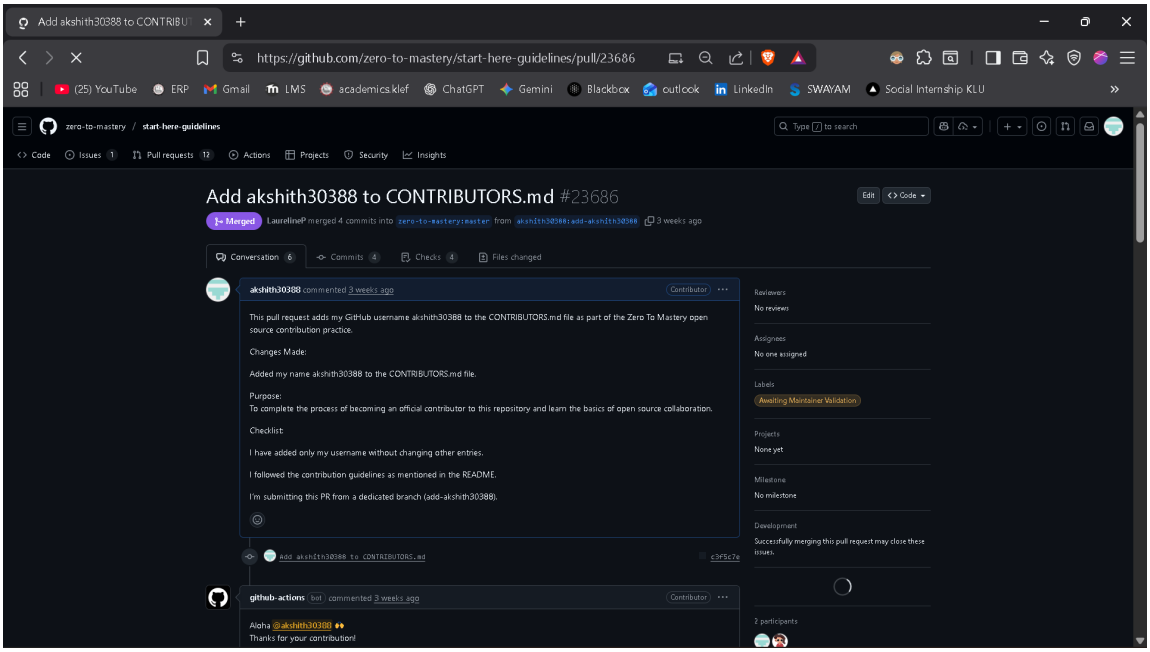
The Start-Here Guidelines repository is open-source and distributed under the **MIT License**. This license allows contributors to freely modify, share, and build upon the project while maintaining transparency and community collaboration. The project encourages beginners to explore Git commands, understand version control, and participate in open-source contributions with confidence.

7.9.3 Community and Review Process

ZTM maintains an active, supportive community that encourages collaboration through GitHub discussions, Discord, and learning groups. My pull request was automatically acknowledged by the project bot, and later reviewed and merged by a maintainer. The validation process verifies that:

- Only the required contributor entry was added.
- The PR follows formatting and style rules.
- The branch structure aligns with the contribution workflow.

This contribution helped reinforce my understanding of the open-source pipeline and improved my familiarity with GitHub collaboration practices.



7.10 PR 6 : OpenEMS

7.10.1 The Issue (The Problem Being Fixed)

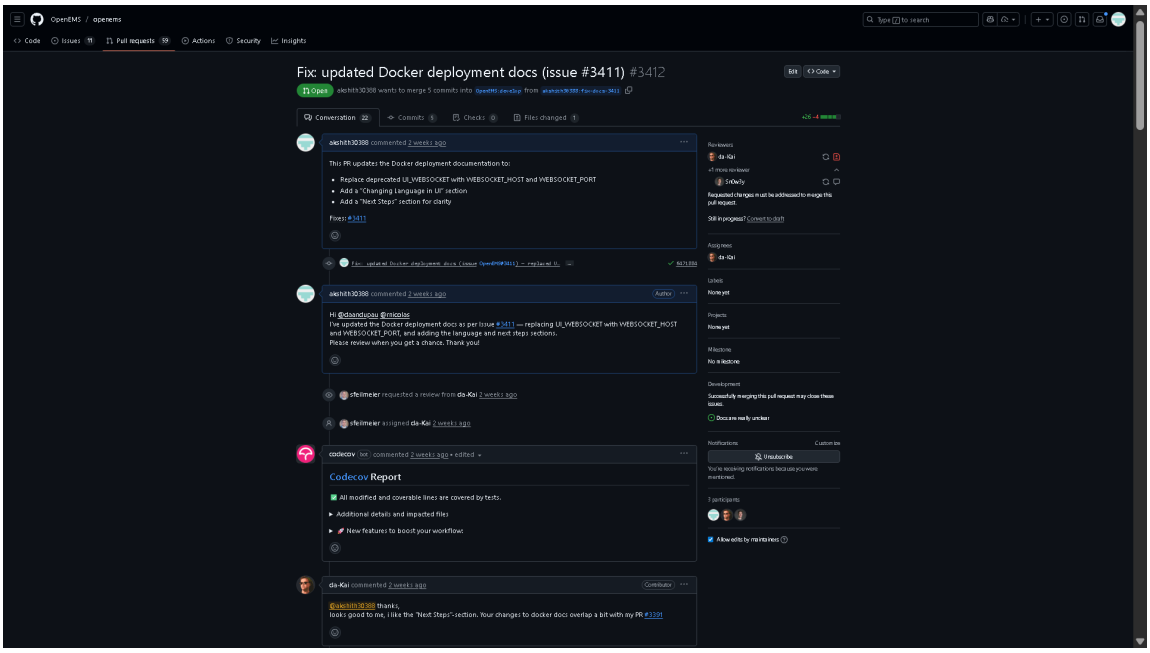
The primary issue addressed by this Pull Request is related to **outdated and unclear Docker deployment documentation** in the OpenEMS project. The original documentation used deprecated environment variables such as `UI_WEBSOCKET` and lacked important explanations, leading to confusion for new developers attempting to deploy OpenEMS via Docker. Additionally, the absence of a **language selection guide** and a **Next Steps** section resulted in incomplete onboarding instructions. These gaps made it difficult for users to correctly configure and understand the deployment process. The PR therefore targets a **documentation improvement (non-breaking change)** meant to enhance clarity, fix outdated content, and improve the overall developer experience.

7.10.2 The Solution (What Was Done)

The submitted solution updates the Docker deployment section to replace old variables with the new, correct parameters `WEBSOCKET_HOST` and `WEBSOCKET_PORT`. It also introduces two new sections:

- **Changing Language in UI** – guidance for modifying the application's display language.
- **Next Steps** – directions for what users should do after deploying OpenEMS.

Throughout the review process, communication took place between contributors and maintainers such as `da-Kai` and `sfleimeier` to ensure accuracy and prevent conflicts with existing pull requests. Automated checks (including the Codecov bot) validated the updates, and the PR now awaits final maintainer approval. These improvements significantly refine the documentation and make the deployment workflow far more accessible.



8 Linkedin Post Links

8.1 PR :

`https://www.linkedin.com/feed/update/urn:li:activity:7399012150155014144/?originTrackingId=BeHgmVSHjwAFBwS8awFu5Q%3D%3D`

8.2 Journey Of Open Source :

`https://www.linkedin.com/feed/update/urn:li:ugcPost:7399051273720774656/`

8.3 Self Hosted Project :

`https://www.linkedin.com/feed/update/urn:li:activity:7399141340653207552/?originTrackingId=2ivKTm1aZpUWxlr1glFclw%3D%3D`