Open Source Engineering Report

Student Name: Atmakuru Hema Venkata Sai Sathvik (2400040068)

Branch: B.Tech – Electronics and Communication Engineering (ECE)

Course: Open Source Software Engineering

Academic Year: 2025–2026

Submitted To:

Dr.Sripath Roy Koganti / EL&GE / KL University

Contents

1 About Linux Distro Used: Ubuntu

Ubuntu is one of the most popular Linux distributions used by developers, students and beginners. It is based on Debian and is known for its stability, regular updates and a friendly graphical interface. Ubuntu is widely used in software development, cloud computing and open-source learning labs.

Ubuntu provides thousands of free and open-source packages through the `apt` package manager. Using simple commands, we can install compilers, editors, servers and security tools. This makes it a very good choice for students who are just starting with Linux.

A key advantage of Ubuntu is its Long-Term Support (LTS) releases. LTS versions receive security and bug fix updates for five years, so they are trusted by companies and universities. Most major cloud platforms like AWS, Azure and Google Cloud support Ubuntu images by default.

In this course, Ubuntu helped me learn:

- Basic terminal commands for navigation and file handling
- Installing and updating software using `apt`
- Managing users, permissions and executable files
- Using Git and GitHub directly from the terminal
- Running and testing self-hosted services such as paperless-ngx

Overall, Ubuntu gave me a strong foundation in using Linux as a development environment for open source engineering.

2 Encryption and GPG

GNU Privacy Guard (GPG) is a free and open-source implementation of the OpenPGP standard. It is used for encrypting files, signing data and verifying signatures. The main idea is public-key cryptography: each user has a **public key** (can be shared) and a **private key** (kept secret).

When someone wants to send us a secret message, they encrypt it with our public key. Only our private key can decrypt that message. In the same way, if we sign a file with our private key, others can verify the signature with our public key and confirm that it really came from us and has not been modified.

Common GPG Commands

- `gpg --full-generate-key` – Generate a new key pair (public + private)
- `gpg --list-keys` – Show the public keys stored in our keyring
- `gpg --export --armor > publickey.asc` – Export our public key so that we can share it
- `gpg --encrypt --recipient <email> file.txt` – Encrypt `file.txt` for a specific user

- `gpg --decrypt file.txt.gpg` – Decrypt an encrypted file using our private key

In the lab we practised generating keys, exporting the public key and encrypting and decrypting sample files. This helped me understand how many open-source projects sign their releases and how users can verify authenticity.

3 Sending Encrypted Email

Normal email is like sending a postcard: anyone on the path can read the content. To protect privacy, we can combine email with GPG encryption. For this we can use tools such as Thunderbird with built-in OpenPGP support or browser plugins like Mailvelope.

Steps for Encrypted Email

- Both sender and receiver generate their own GPG key pairs.
- Each person shares their **public key** with the other, usually as a `.asc` file or via a key server.
- In the email client, we import the other person's public key and mark it as trusted.
- While composing a mail, we select the option "Encrypt" (and optionally "Sign").
- The email body is encrypted with the recipient's public key and sent over the internet.
- The recipient opens the mail, enters their passphrase and decrypts the message using their private key.

This activity showed me how encryption is used in real life for secure communication and how public-key infrastructure works beyond theory.

4 Privacy Tools (PRISM-BREAK)

PRISM-BREAK is a community-driven website that lists privacy-respecting alternatives to many popular services. Its goal is to help users avoid mass surveillance and tracking by using open-source and decentralised software.

Some tools we explored are:

- **Signal** – An end-to-end encrypted messaging application for chats, voice and video calls.
- **Tor Browser** – A browser that routes traffic through multiple relays, hiding the real IP address and making tracking difficult.
- **KeePassXC** – A local password manager that stores all passwords inside an encrypted database file.
- **Jitsi Meet** – An open-source video conferencing platform that can also be self-hosted.

- **LineageOS** – A free Android-based operating system that removes bloatware and gives more control over permissions.

These examples helped me see that privacy is not only a theory topic. There are real open-source tools available for almost every daily use-case.

5 Open Source License Used – MIT

The MIT License is a permissive open-source license that allows:

- Free use, modification and distribution of the software
- Commercial use without extra restrictions
- Closed-source modifications and inclusion in proprietary projects
- Only requirement: include the original copyright and license notice

Because it is short and easy to understand, many open-source libraries and student projects select the MIT license. It encourages reuse and sharing while still giving credit to the original author.

6 Self Hosted Server – paperless-ngx

paperless-ngx is an open-source document management system (DMS). It allows us to scan, upload and organise PDFs and images so that they become searchable digital documents. Instead of keeping files in random folders, paperless-ngx stores them in a structured way with tags, correspondents and document types.

Features

- Web interface for uploading and browsing documents
- OCR (Optical Character Recognition) to make scanned PDFs searchable
- Tags and metadata like correspondent, document type and storage path
- Powerful search across content and tags
- Can be fully self-hosted using Docker and Docker Compose

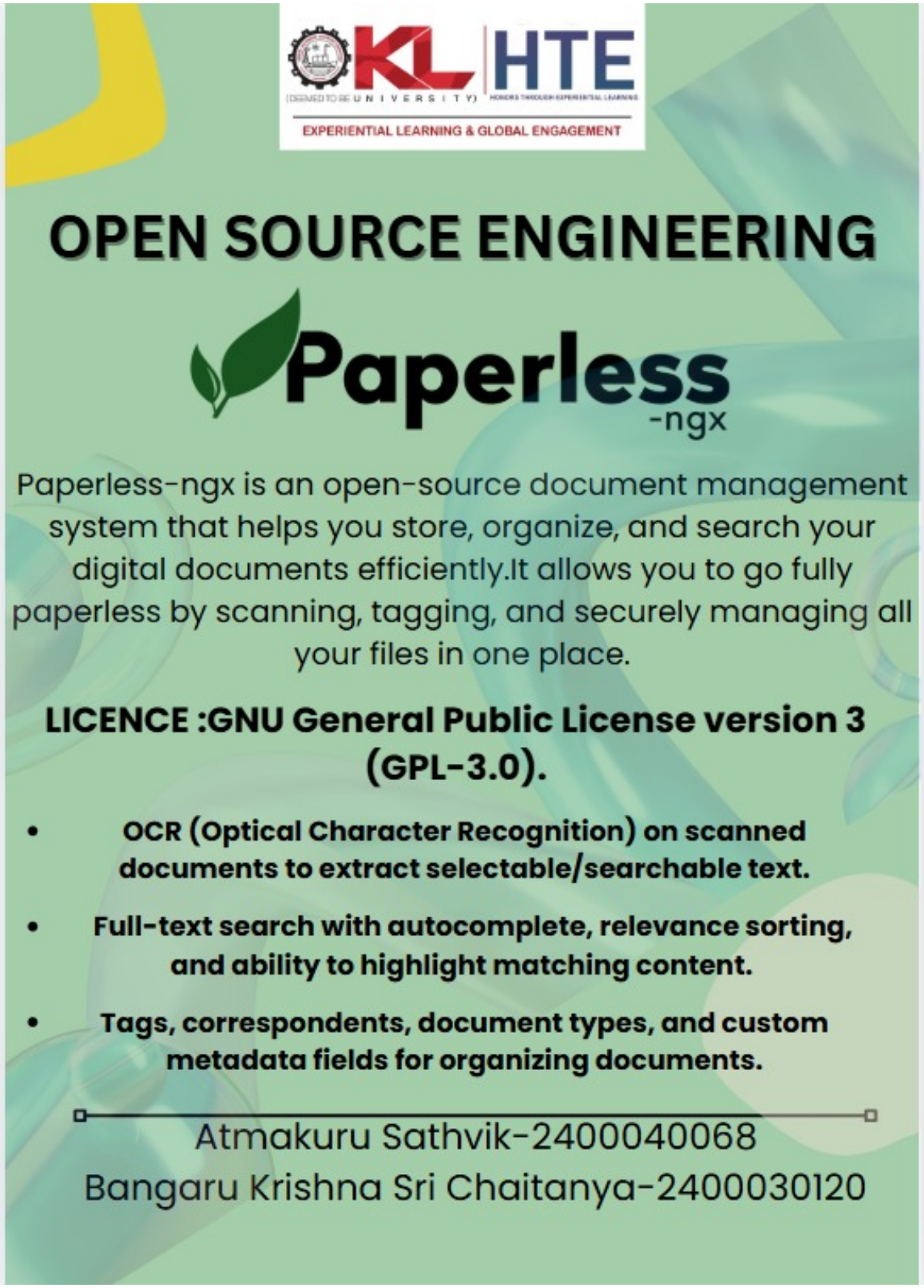
How I Self-Hosted paperless-ngx

- Installed Docker Engine and Docker Compose on my Ubuntu system.
- Created a dedicated folder for the paperless-ngx stack and wrote a `docker-compose.yml` file.
- Used the official images from `ghcr.io/paperless-ngx/paperless-ngx` along with Redis and PostgreSQL containers.

- Configured environment variables such as `PAPERLESS_URL`, ports, admin user and data volumes.
- Started the services using `docker compose up -d` and checked the logs to ensure all containers were healthy.
- Opened `http://localhost:8000` in the browser, logged in with the admin account and uploaded sample PDFs to test OCR and search.

Localized(Translated) document –
`https://docs.google.com/document/d/1C2mKpyJvyLZqFp2txewTeIxCa8KxqA6M/edit?usp=drivesdk&ouid=103590189607632093821&rtpof=true&sd=true`

Dashboard Screenshot



7 Open Source Contributions

GitHub Username: **AHVSSATHVIK**

In this course I was encouraged to contribute to real open-source projects. I focused mainly on learning proper Git workflows, self-hosting documentation and beginner-friendly pull requests.

List of Pull Requests

- **firstcontributions / first-contributions** – "Add my name to Contributors list" (First Contribution PR)

  This was my first pull request on GitHub. I forked the repository, cloned it to my local system and added my name to the `Contributors.md` file. Then I committed the change and pushed it back to my fork, and finally created a pull request to the main repository.

  Through this contribution I learnt the full basic workflow of Git and GitHub:

  - how to fork a project,
  - how to create a new branch,
  - how to commit and push changes,
  - and how to wait for maintainers to review and merge the PR.

  Having this PR merged gave me confidence to continue with other open source work.

- **KLGLUG / Y24OpenSourceEngineering** – "Added paperless-ngx self-hosted project documentation in Telugu"

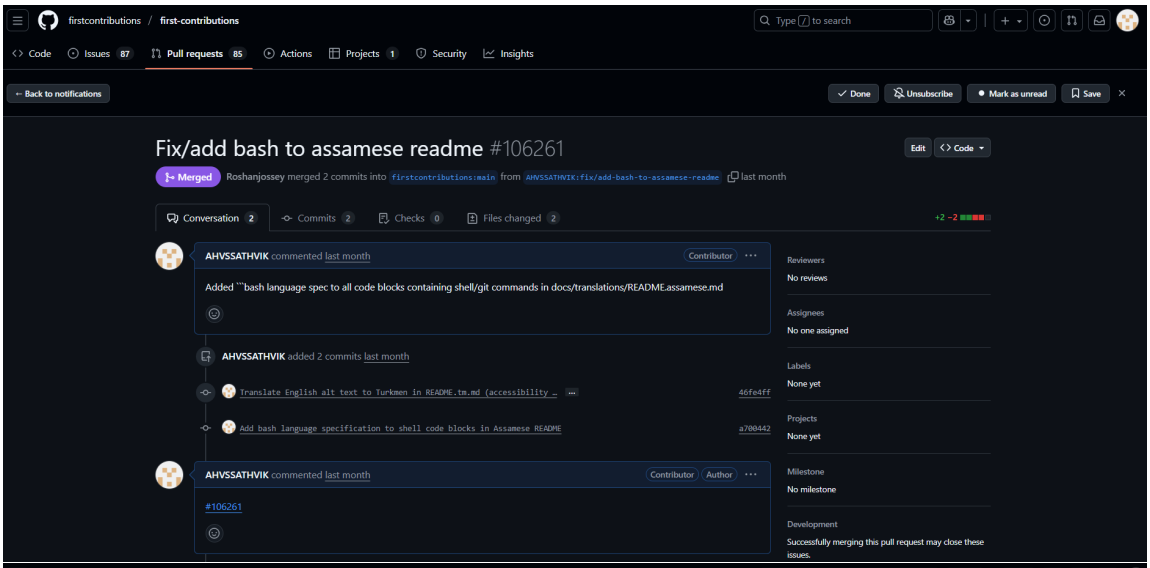
  In this pull request I documented, in Telugu, the complete steps for self-hosting paperless-ngx using Docker and Docker Compose. My goal was to make self-hosting accessible to Telugu-speaking students so that they can build their own document management system at home or in college labs.

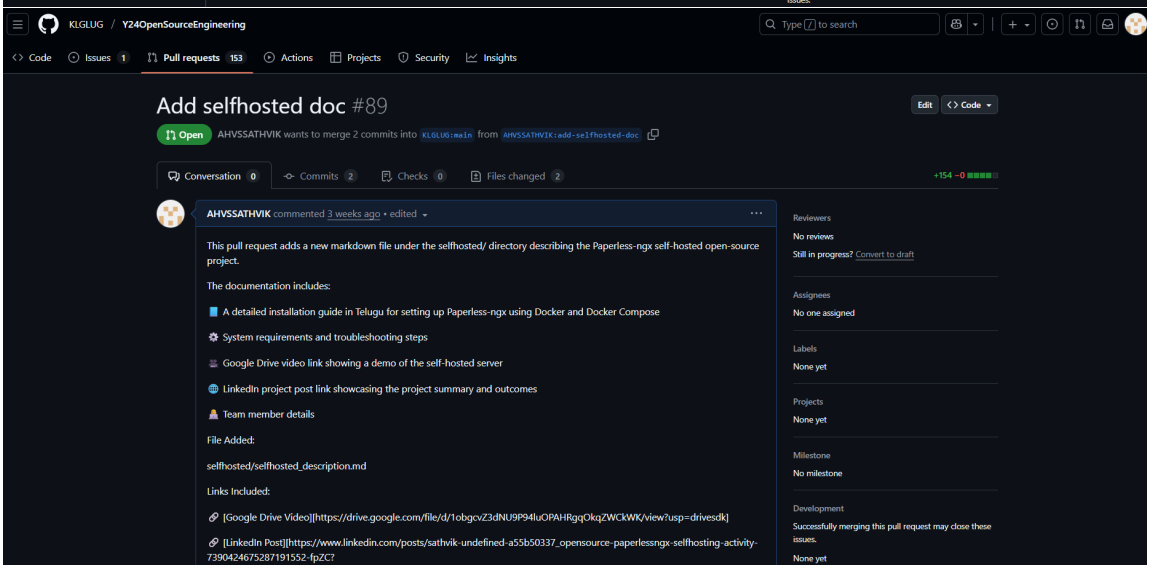
  This contribution taught me:

  - how to write clear and reproducible self-hosting instructions,
  - how to follow a repository's folder structure and naming guidelines,
  - how localisation and native-language content can motivate more students to try FOSS tools.

  At the time of writing this report, the PR is open / under review (or merged, depending on current status).

Screenshots





8 LinkedIn Posts

I have shared my open-source and self-hosting journey on LinkedIn. These posts helped me explain my learning, document my progress and connect with other developers and students.

- Post about my first merged pull request in open source – https://www.linkedin.com/posts/sathvik-undefined-a55b50337_hacktoberfest2025-activity-7398665495937454080-I1eX
- Post about my open source blog – https://www.linkedin.com/posts/sathvik-undefined-a55b50337_open-source-activity-7398403180507791361-TCKB
- Post about self-hosting paperless-ngx and using Docker at home – https://www.linkedin.com/posts/sathvik-undefined-a55b50337_opensource-paperlessngx-selfhosting-activity-7390424675287191552-fpZC