TempShare : Local network file sharing app

Submitted in partial fulfillment of the requirements of the degree of Bachelor of Engineering (Information Technology)

By

Jai Talreja - Roll Number 59

Department of Information Technology
VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY,
Chembur, Mumbai 400074
(An Autonomous Institute, Affiliated to University of Mumbai)

April 2025

TempShare

Name of student	Jai Talreja
Class_Roll no	D15A_59
D.O.P	3/04/25
D.O.S	17/04/25
Sign and Grade	

Project Description :

TempShare is a local file sharing web-app that is built using Python Flask. This web-app does not need global internet access and primarily works within local networks, such as multiple devices connected to a router or a mobile hot-spot, neither of which need to have WAN. The app, when started on a host machine, lets other users connect to the session using a QR code, and alternatively, a link. Once connected, the users can upload files to the session and download files that have been uploaded to the session. The users can also delete files from the session for security. For efficiency and security, the app deletes all files once a session is over (all users disconnected from the session), and new sessions always start fresh.

Requirement gathering :

The requirement gathering for TempShare was conducted through observation of daily task management challenges, and analysis of common file sharing utilities. The primary goal was to identify pain points in existing systems, such as reliance on global internet access and lack of digital privacy due to the aforementioned. Individual users were the main concern for this project. Functional requirements included File Management, File Validation, Sharing Features, Sharing Features, Real-Time Updates, and Non-Functional requirements included Performance, Security, Usability, Reliability, Maintainability, Compatibility, Scalability and Non-Reliance on global internet access. Assessment of the implemented features helped refine the project further.

System Requirements :

1. Hardware Requirements :

- **Processor:** Any 64-bit x86 or ARM CPU with at-least 2 cores running at at-least 1 GHz
- **RAM:** 4 Gigabytes or higher
- **Storage:** 10 Gigabytes or more free (For everything used to build the project as well as the project itself)
- **Network:** Any device that can behave like a router (Internet plan/package not needed)

2. Software Requirements :

- **Operating System:** Any OS that supports Python, JS, HTML and CSS (Linux, Windows, MacOS, Chrome OS and Android)
- **Code Editor:** Any IDE that can interpret or compile Python code (VSCode and IDLE)
- **Version Control :** Any version of Git higher than 2.0.0

Technologies Used :

Development	VS Code, Git
Frontend	JavaScript, Jinja2
Backend	Flask (Python 3.x)
File Transfer	WebSockets, Flask-SocketIO
Styling	CSS, Google Material Icons
APIs	Pillox, QRCode, Python-EngineIO

Setup Instructions :

(If using a Linux distribution, update your package manager's package cache before installing the required software, to get the latest versions of all required software)

- **Python 3.x :** On Ubuntu, and other Linux distributions, install the latest version of Python (python3) using the native package manager (APT on Ubuntu, and DNF or Yum on Fedora/RHEL).
- **Virtual environment :** To avoid compatibility issues or conflict between multiple Python installations, install and setup a virtual environment for Python and then install all dependencies inside the virtual environment. Install Venv for Python (python3-venv) using the native package manager (APT on Ubuntu). Then, create a virtual environment using 'python3 –m venv venv' and activate it using 'source venv/bin/activate', to deactivate it, use 'deactivate'.
- **Python package installer (pip) :** Install pip (python3-pip) using the native package manager (APT on Ubuntu) inside the virtual environment.
- **Other project dependencies :** Use pip to install all other project dependencies inside the virtual environment, such as Pillow and QRCode for the QR code functionality, Flask for the backend web framework, WebSockets and Flask-SocketIO for real-time updates and file sharing.

Project requirements : This project has the following dependencies other than Python,

Flask==3.0.2
qrcode==7.4.2
Pillow==10.2.0
flask-socketio==5.3.6
python-engineio==4.9.0
python-socketio==5.11.1

Running the project :

Once the virtual environment is setup and all project files have been downloaded and installed, run the Python code from the root directory using (python3 app.py). The project will start running at the port 5000 of the host machine.

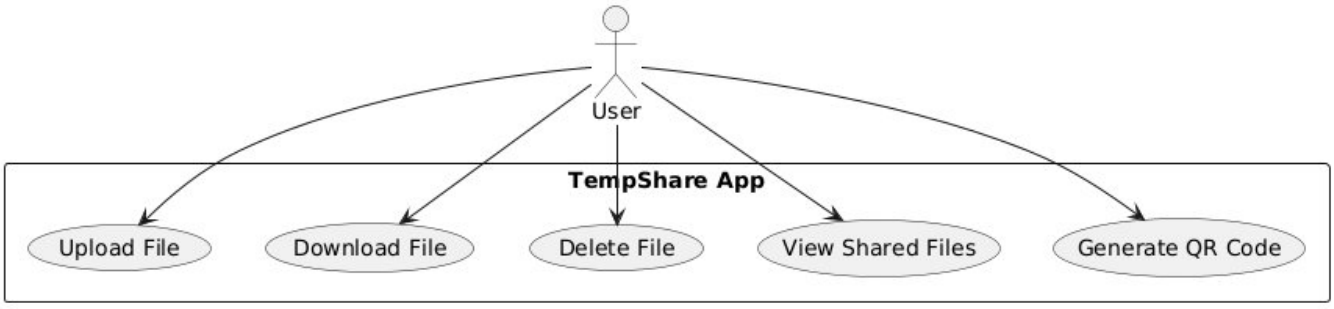
Project file structure : The file and directory structure of this project is as follows :

```
tempshare/
├── app.py
├── requirements.txt
├── static/
│   ├── css/
│   │   └── style.css
│   └── js/
│       └── app.js
├── templates/
│   ├── base.html
│   └── index.html
└── uploads/
```

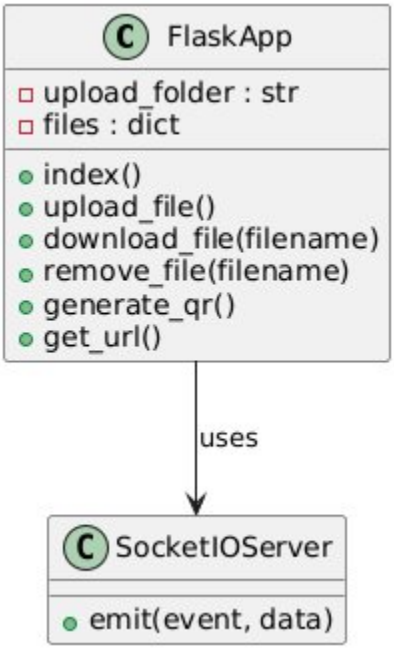
Project features :

- Real-time file sharing within local network
- Material Design UI
- QR code generation for easy sharing
- File upload with extension validation
- Real-time updates using WebSocket
- Temporary file storage
- Download functionality for all users
- Share via link or QR code

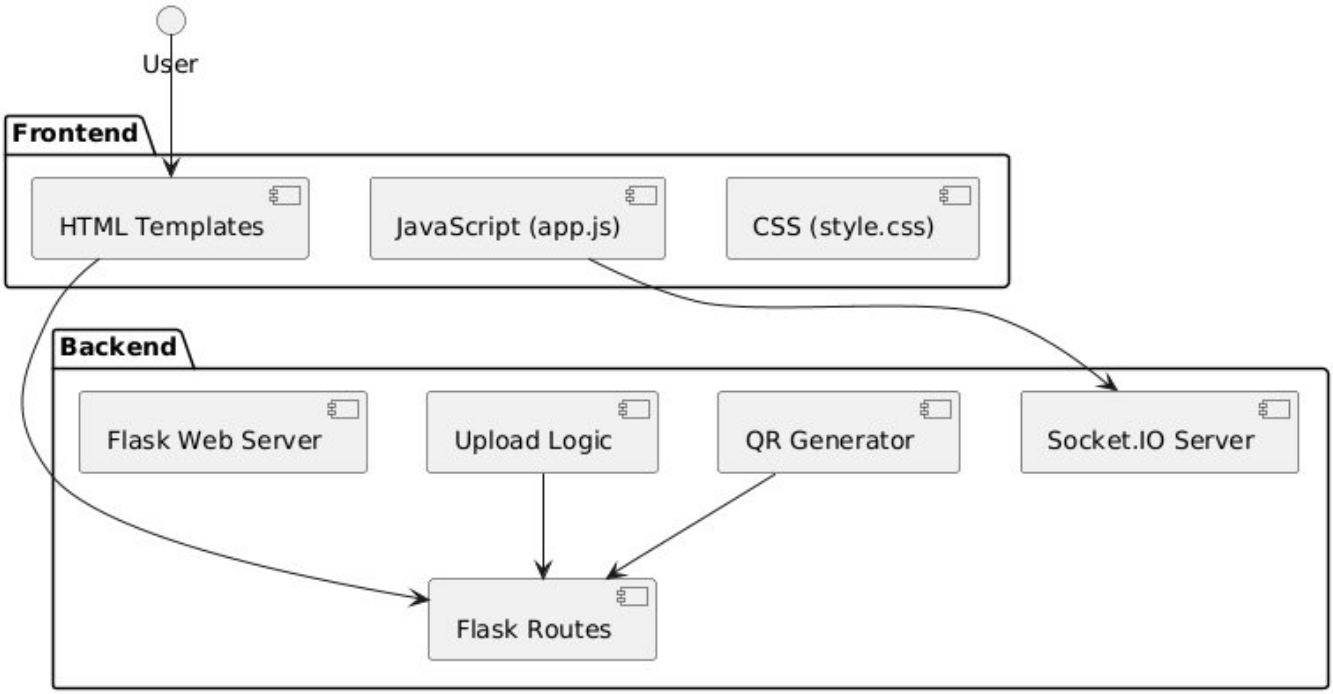
Architectural Diagrams :



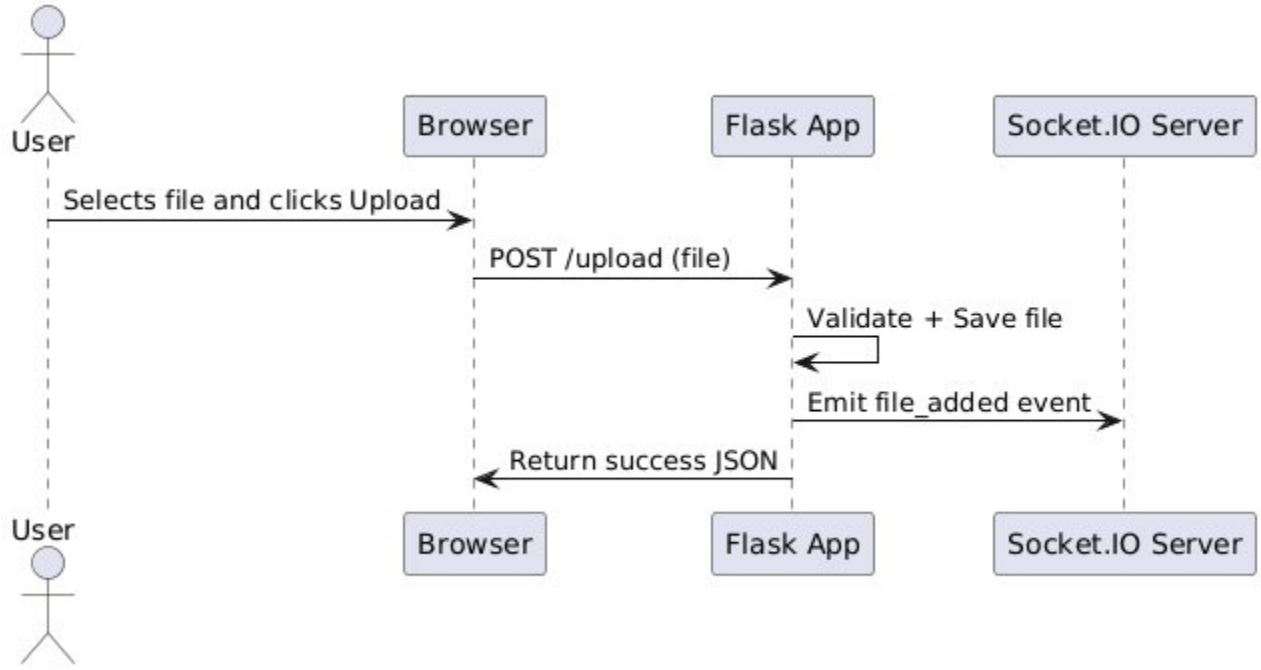
Use-Case Diagram



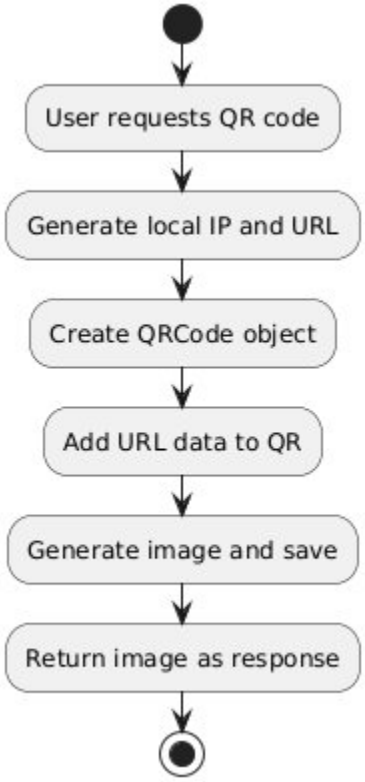
Class Diagram



Component Diagram

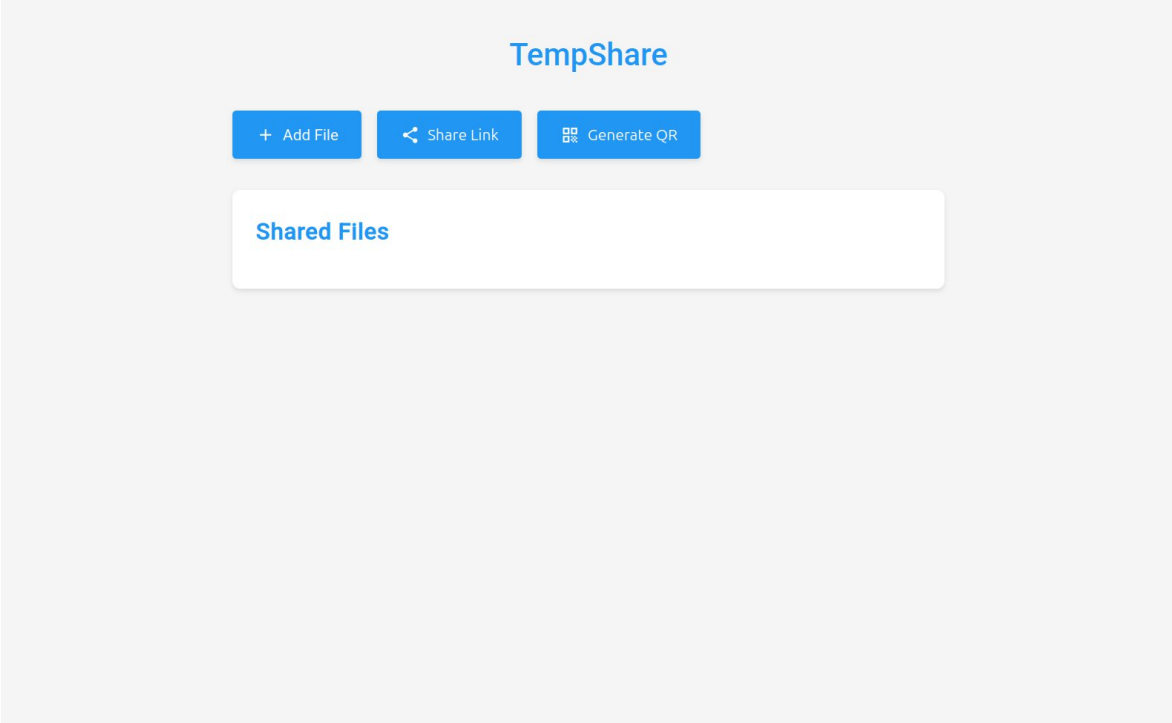


Sequence Diagram : File Upload



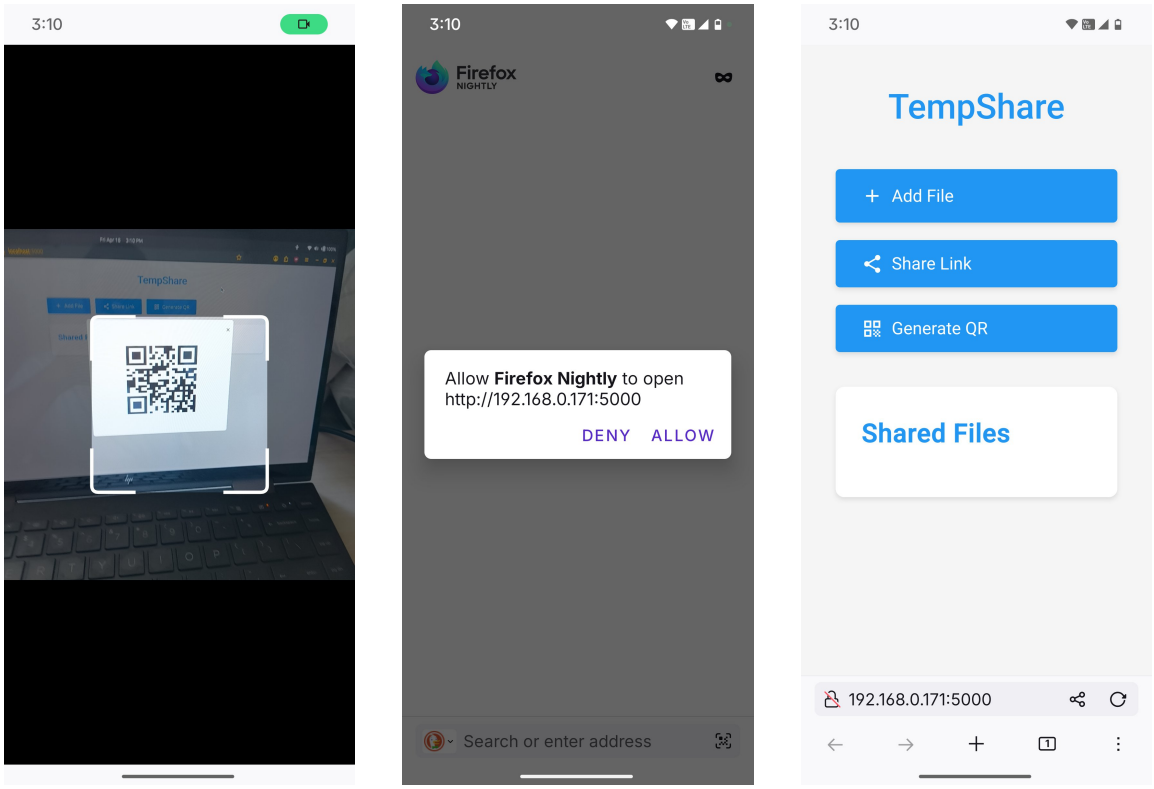
Activity Diagram : QR code Generation

Result screenshots :

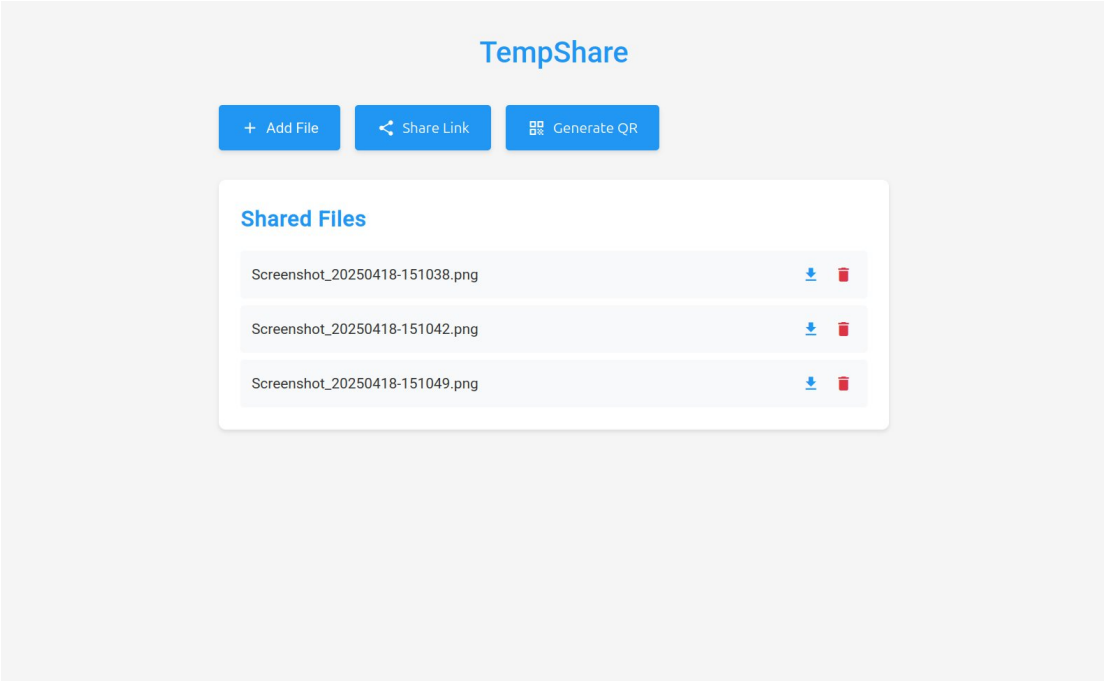


The web-app graphical user interface

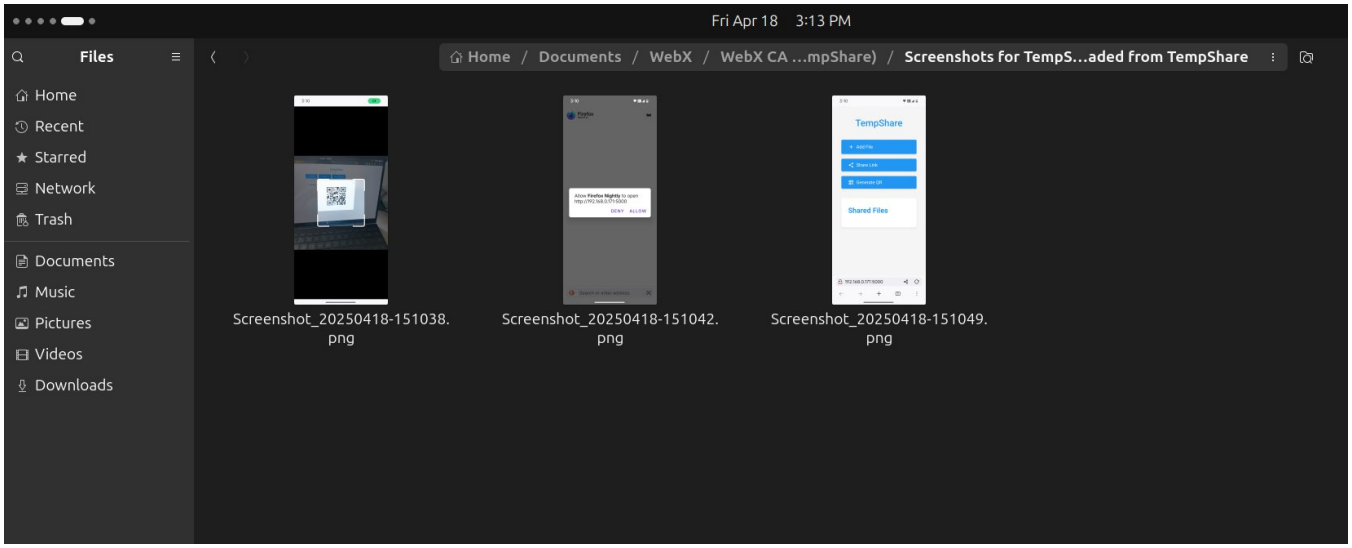
Generated a QR code to be scanned



Scanned the QR code using a web browser that supports it (Firefox)



Added files from mobile device and received on desktop device



Added files from mobile and received on desktop, also downloaded

Future Scope :

While TempShare effectively facilitates real-time file sharing over a local network, there are several potential enhancements that could expand its functionality and improve user experience. Implementing session tracking with user connection counts would enable automatic file cleanup when the last user disconnects, aligning the system behavior with its intended temporary nature. Introducing time-based expiration for uploaded files can further automate file life-cycle management. Enhancing the front-end with features such as drag-and-drop file uploads and a more intuitive interface could improve usability. Additionally, incorporating password protection or authentication mechanisms would increase security for sensitive transfers. Finally, an administrative dashboard displaying file statistics, user activity, or logs could provide greater control and insight into system usage.

GitHub Link : https://github.com/D15A-Jai-61/TempShare.git

Conclusion :

TempShare simplifies quick, local file sharing without the need for cables through a well-integrated tech stack consisting of Flask, Jinja2, JavaScript, WebSockets, Pillow, and QRCode. The setup involves installing essential tools like Python, pip, Flask, ensuring a robust development environment.
With a simple, minimal and responsive graphical user interface and fast real-time updates, TempShare streamlines the process of sharing files between devices located nearby without the need for cables.