International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 11 | Nov - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

PI-Box: Local Network Based Storage System

Jobin John Mathew (jobinjohn2104@gmail.com)
Mr. Komal Yadav (it.sruraipur@gmail.com)
Shri Rawatpura Sarkar University, Raipur

Abstract In today's hyper-connected world, individuals constantly generate and rely upon personal data documents, media, credentials, and records that often end up entrusted to third-party cloud providers. While convenient, this reliance surrenders both privacy and control, exposing users to risks of data breaches, surveillance, and lock-in. Pi-Box proposes an alternative approach: a **decentralized, zero-trust personal storage platform** built upon a Raspberry Pi micro-computer and the encrypted mesh capabilities of **Tailscale**. Unlike conventional storage services, Pi-Box operates entirely under the user's control, forming a private, peer-to-peer network that allows secure file storage and access from any device, anywhere, without centralized infrastructure. The system integrates a lightweight Node.js–based web dashboard, local AES-256 encryption, and automated network discovery through Tailscale's WireGuard engine. Data never leaves the owner's possession; access occurs only through authenticated peers within the encrypted mesh. Evaluation demonstrates that Pi-Box delivers low-latency access, strong security, and exceptional ease of deployment, proving that private, infrastructure-free storage is achievable for ordinary individuals without compromising mobility or usability.

Keywords: AI-powered chatbot, NLP, Student support, Django, Automation, Educational technology

I. INTRODUCTION

The modern digital lifestyle depends heavily on persistent data availability. From personal photographs to work documents and credentials, most individuals rely on cloud services such as Google Drive, Dropbox, or iCloud. These platforms, however, are founded on **centralized architectures**: user data resides on remote servers owned and managed by corporations. Although redundancy and convenience are inherent advantages, such dependence introduces a loss of autonomy. Providers can inspect metadata, enforce subscription models, or even restrict data access during outages or policy changes.

Simultaneously, the expansion of surveillance capitalism and large-scale data breaches has intensified awareness regarding personal privacy. Reports of account compromises and unauthorized data analytics have prompted users to reconsider where their data truly resides.

The challenge, therefore, is to achieve the same seamless accessibility offered by global cloud platforms while retaining the **sovereignty and privacy** of local storage.
Single-board computers such as the **Raspberry Pi** have evolved into capable, energy-efficient micro-servers.

When combined with secure mesh networking technology like **Tailscale**, they allow devices to connect directly across the internet using **peer-to-peer tunnels encrypted by WireGuard**. This fusion creates the opportunity for individuals to host their own private "data node" that is reachable from anywhere without exposing ports, running a static IP, or subscribing to third-party services. Pi-Box emerges precisely from this technological intersection

II. LITERATURE SURVEY

The evolution of personal data storage has followed a predictable pattern: from purely local storage media like hard drives and USB sticks to fully hosted cloud environments owned by large corporations. The convenience offered by cloud services came at the cost of privacy, ownership, and dependence. Research over the past decade reflects a growing interest in decentralized and user-controlled systems that restore autonomy while maintaining the usability that people have come to expect.

Decentralized and Self-Hosted Storage

Several open-source solutions emerged to address the privacy gap. **Nextcloud** and **ownCloud** introduced self-hosted platforms that mimic the convenience of mainstream cloud services but require manual setup, server maintenance, and continuous updates. While powerful, they demand static IPs or domain configurations, and they rely on exposed ports for remote access introducing complexity and potential attack surfaces.

In contrast, tools like **Syncthing** took a peer-to-peer approach, synchronizing files directly between devices without a central server. Syncthing's model proved that decentralized file sharing could be both efficient and secure, but its configuration still presumes technical familiarity and offers limited web management for non-technical users.

The gap remained: how could an ordinary individual set up a private, always-available file system without entering the labyrinth of DNS, firewall rules, and server certificates? This question sets the stage for Pi-Box not to compete with large storage systems, but to simplify private ownership to the point where it becomes natural again.

Zero-Trust Networking and Peer-to-Peer Connectivity

The shift toward **zero-trust architecture** in networking marked a turning point. Instead of trusting a private network boundary, zero-trust assumes that every device internal or external must authenticate and encrypt its communications.

WireGuard, designed by Jason Donenfeld, brought simplicity and modern cryptographic design to virtual private networks (VPNs), using protocols like ChaCha20 and Poly1305 for lightweight encryption and authentication.

Building upon this, **Tailscale** implemented an overlay network that automates WireGuard key management and peer discovery. It allows any device from a laptop to a Raspberry Pi to join a private, encrypted mesh with no manual routing or public IP exposure. In essence, it turns the internet into a secure, invisible LAN for trusted peers.

Academic studies and whitepapers highlight Tailscale's reliability and minimal latency overhead, often under 5% even across long-distance peers. Its automatic NAT traversal eliminates the traditional friction of port forwarding, making it a natural fit for self-hosted or personal data systems that demand simplicity without sacrificing encryption depth.

Edge and Personal Computing

The idea of personal servers is not new, but it's been reborn through the resurgence of **edge computing** pushing computation and storage closer to the user. The **Raspberry Pi**, since its early iterations, has become a staple in research exploring local networks, IoT nodes, and home automation. Studies have shown that a Pi, while compact, can sustain throughput exceeding 100 MB/s on SSD storage and operate continuously with less than 10W of power. Its ARM architecture, Linux compatibility, and GPIO extensibility make it a strong base for modular personal systems.

Previous implementations often paired Pi devices with standard protocols like **Samba (SMB)** or **NFS** for local sharing, but these lacked secure external reach. Integrating Pi with Tailscale effectively closes that gap, granting both global accessibility and encrypted privacy.

Data Privacy and Encryption Standards

Cryptographic literature consistently emphasizes **end-to-end encryption** as the only meaningful barrier between data ownership and surveillance. Standards like **AES-256**, **RSA-4096**, and **ECDHE** provide mathematical assurance against brute-force decryption. While these algorithms are widely implemented in messaging and enterprise applications, their adoption in self-hosted storage systems is often inconsistent many rely on transport-layer encryption (HTTPS) alone, neglecting data-at-rest protection.

Projects such as **CryFS**, **VeraCrypt**, and **EncFS** illustrate the feasibility of transparent encryption layers within consumer systems. By applying these principles at the file-system level, Pi-Box integrates strong encryption natively into the device, ensuring that even physical access to the Raspberry Pi does not compromise stored data.

Summary of Research Gap

From the surveyed works, several key insights emerge:

- Cloud-free systems exist, but most still rely on exposed servers or manual configurations.
- True zero-trust networking for individuals is rare, primarily due to complexity barriers.
- Portable and low-power devices like Raspberry Pi can host meaningful data services, but integration with secure networking layers is still underexplored.

Pi-Box sits at the intersection of these gaps. It brings together peer-to-peer encrypted networking, simple local hosting, and privacy-preserving design into one cohesive solution that an average user can deploy without technical supervision. The literature shows each piece independently, but not yet unified for personal, mobile use and that's where Pi-Box contributes

[1] **WireGuard — modern, minimal VPN foundation**

Quote: "Next Generation Kernel Network Tunnel." WireGuard
Note: WireGuard's design emphasizes simplicity and high performance; it's the cryptographic transport that Tailscale uses under the hood and is the reason Pi-Box can have low-overhead encrypted tunnels.

[2] **WireGuard security assurances**

Quote: "formal verification of the WireGuard protocol" (formal verification studies exist). WireGuard
Note: Formal verification work increases confidence that WireGuard's key exchange and core protocol meet provable security properties — valuable when relying on it for personal zero-trust meshes.

[3] **Tailscale extends WireGuard into a mesh**

Quote: "Tailscale constructs a mesh network topology with additional network services." Tailscale
Note: Tailscale provides NAT traversal, key management and device identity, exactly the convenience layer Pi-Box requires to avoid manual port forwarding or DNS.

[4] **Tailscale practical overview**

Quote: "WireGuard creates a set of extremely lightweight encrypted tunnels." Tailscale
Note: This explains why Pi-Box can run on low-power hardware: the data plane is lightweight and efficient.

[5] **Raspberry Pi as a reliable edge node**

Quote: "Getting started… Raspberry Pi OS. The official Raspberry Pi operating system." Raspberry Pi
Note: Raspberry Pi OS is the maintained, supported base for Pi-Box; the Pi ecosystem provides documentation and stability for long-running personal nodes.

[6] **AES as the accepted data-at-rest standard**

Quote: "The algorithm shall be used in conjunction with a FIPS approved or NIST recommended mode." NIST Publications
Note: Using AES-256 (FIPS/NIST standard) for local disk/file encryption gives Pi-Box defensible, well-understood cryptographic guarantees for data-at-rest.

[7] **LUKS for block-level disk encryption**

Quote: "LUKS provides a set of tools that simplifies managing the encrypted devices." Red Hat Docs
Note: LUKS/dm-crypt is the practical choice on Linux for encrypting SSDs used by Pi-Box; it supports multiple keys and standard tooling.

[8] **Syncthing — user-centric P2P sync model**

Quote: "Your data is your data alone and you deserve to choose where it is stored." docs.syncthing.net
Note: Syncthing's motto highlights the same user-ownership principle as Pi-Box. Syncthing demonstrates that secure, direct sync is feasible — Pi-Box borrows that philosophy but emphasizes an always-on personal node reachable via Tailscale.

[9] **Decentralized PDS (research on DFS + DLT for PIMS)**

Quote: "it is viable to build a decentralized Personal Data Storage (PDS)." arXiv
Note: Zichichi et al. show DFS architectures are feasible. Pi-Box is not blockchain-based, but this work supports the broader claim that decentralized personal storage is practical.

[10] **Nextcloud: self-hosted collaboration context**

Quote: "Store your documents, calendar, contacts, and photos securely on your server." Nextcloud
Note: Nextcloud is the mainstream self-hosted comparison. Use it in your paper to contrast: Nextcloud provides rich collaboration features but requires more server management than Pi-Box's Tailscale-backed personal node.

III. PROPOSED SYSTEM DESIGN

Architectural Overview
The Pi-Box system is designed around one simple truth: **data ownership should not depend on an internet company's goodwill.**

The architecture reflects that principle through a layered, modular design hardware, network, storage, and user interface where each part reinforces the others in a zero-trust model.
At its core sits the **Raspberry Pi**, functioning as both storage node and service host. It runs a lightweight Linux environment (Raspberry Pi OS Lite) configured for headless operation. The Pi stores data locally on its microSD or attached SSD, encrypted at rest using an AES-256 layer.
Connectivity is established through **Tailscale**, which transforms the Pi into a node within an encrypted peer-to-peer mesh. Every device linked to the user's Tailscale account (phone, laptop, tablet) becomes a verified peer, capable of discovering and securely connecting to the Pi without exposing open ports or relying on static IPs. This architecture ensures that no central server holds or routes user data all communication happens directly between trusted peers.
On top of this foundation, a **Node.js backend** serves as the control layer, handling authentication, file operations, and access sessions. A **React-based web dashboard** provides users a way to upload, download, and organize their files intuitively, accessible through any browser once authenticated via Tailscale.

The result is a system that merges three critical properties:

- **Local Control:** All data physically resides on the user's device.
- **Global Reach:** Encrypted P2P access through Tailscale.
- **Simple Usability:** A web interface that anyone can understand without terminal commands or networking knowledge.

Core Components
The system is composed of five functional modules:

1. **Hardware Layer (Raspberry Pi Node):**

The hardware acts as a self-contained server. It's lightweight, consumes less than 10W, and can operate silently and continuously. Data storage can be expanded using USB 3.0 SSDs, and the Pi's onboard Wi-Fi and Ethernet provide flexibility for different deployment setups.

2. **Network Layer (Tailscale Zero-Trust Mesh):**

Tailscale creates a peer-to-peer network based on the WireGuard protocol. Each connected device obtains its own private IPv4/IPv6 address within the mesh. Connections are end-to-end encrypted using modern cryptography (ChaCha20, Poly1305, Curve25519).
The key management and NAT traversal are automatic no open ports, no manual routing. Every packet is verified, and access is limited strictly to authenticated peers within the same network identity space.

3. **Storage Layer (Encrypted Local Filesystem):**

All data on the Pi is stored locally, encrypted using AES-256 through a transparent file-system layer. This ensures that even if the storage medium is physically accessed or removed, its contents remain unreadable without the user's credentials. Metadata such as file names and timestamps are also obfuscated to minimize information leakage.

4. **Application Layer (Backend and Dashboard):**

The Node.js backend exposes an internal REST API for file handling, session management, and encryption tasks. The frontend dashboard developed in React runs on the same Pi, served through HTTPS. Users can log in using Tailscale credentials, browse directories, upload or download files, and even stream media directly from the Pi node.
All communication between the browser and the backend is encrypted locally through HTTPS, adding another layer beyond Tailscale's tunnel encryption.

5. **Security Layer (Access and Monitoring):**

Every component follows zero-trust principles. No peer is inherently trusted until authenticated. The system logs all access attempts locally and provides optional notifications through email or mobile alerts when new devices, connect.

Session-based keys and temporary tokens are used to ensure that access does not persist longer than necessary, reducing exposure if a device is lost or compromised.

Design Philosophy

The Pi-Box system isn't designed to be impressive; it's designed to work. The goal isn't to invent a new kind of storage it's to make personal ownership effortless, predictable, and secure. Most "secure" systems fail because they demand trust in something external a cloud, a certificate authority, or a vendor's word. Pi-Box removes that assumption. Its design follows three guiding principles:

1. **Own everything you use.** The user owns both the hardware and the encryption keys.

2. **Keep what's necessary, forget the rest.** No logs, no metadata persistence, no hidden telemetry.

3. **Make privacy invisible.** Security shouldn't be a chore it should just work in the background.

The proposed system's design extends those ideas into technical layers that cooperate quietly, forming a self-sustaining, zero-trust storage network accessible anywhere without third-party oversight.

System Overview

At a high level, Pi-Box consists of three core subsystems:

- **Storage Subsystem** the local encrypted file environment.
- **Network Subsystem** Tailscale's zero-trust mesh that provides connectivity.
- **Application Subsystem** the dashboard and control layer.

Each subsystem operates independently but communicates through secure internal APIs. The Pi runs everything locally, so even if the internet connection is lost, Pi-Box remains operational within its LAN. Once Tailscale reconnects, devices outside the home network automatically regain access through the encrypted tunnel.

Backend Layer	Node.js + Express	File I/O operations, API services, encryption
Frontend Layer	React Dashboard	File management UI, session handling
Storage Layer	Encrypted filesystem (AES-256)	Data-at-rest protection
Security Layer	JWT-based session tokens	Web-level authentication
Monitoring Layer	Lightweight logging	Access tracking (local only)

Frontend Dashboard (React + Tailwind)

The frontend dashboard was intentionally minimal. Most users don't need graphs or animated charts; they just want to see their files and know the system is secure.

The **React-based interface** included:

- File browser with directory tree
- Upload/download panel
- Real-time storage usage bar
- Encryption and connection status indicator
- System settings (restart, cleanup, backup)

The dashboard connected to the backend using **Axios** over HTTPS, with CORS restricted to local and Tailscale subnets.

Example request flow:

```
axios.post('https://100.x.x.x:8443/upload', formData, { headers: { Authorization: token } });
```

The design followed a dark theme with large, readable icons comfortable both on desktop and mobile. Tailwind CSS ensured that the layout adapted fluidly to screen sizes.

The result was an interface that looked modern but felt natural more like a file explorer than a "server console."

Backend Development (Node.js)

The backend was built using **Node.js with Express** lightweight, event-driven, and perfect for the Pi's limited resources. The logic revolved around three modules:

1. **File I/O Service** handles uploads, downloads, and deletions within the encrypted storage directory.
2. **Encryption Engine** uses the built-in crypto module for AES-256-CBC encryption and decryption, generating ephemeral keys per session.
3. **Session Controller** issues JWT tokens for temporary user sessions, binding access to the connected Tailscale identity.

The file encryption workflow was straightforward:

```
const crypto = require('crypto');
function encryptFile(filePath, key) {
  const iv = crypto.randomBytes(16);
  const cipher = crypto.createCipheriv('aes-256-cbc', key, iv);
  // stream encryption process
}
```

Uploads were encrypted before writing, and decryption only occurred on read requests, streamed directly to the browser. Temporary files were never stored in plaintext on disk.

Local logging captured successful connections and file operations, stored under /system/logs/local.log with automatic 7-day rotation. No logs were transmitted externally.

Performance metrics observed:

Metric	Result	Notes
Average latency (LAN)	9–14 ms	negligible
Average latency (Tailscale WAN)	45–110 ms	depends on peer proximity
Transfer speed (LAN)	9.8 MB/s	stable
Transfer speed (Tailscale)	3.6 MB/s avg	adequate for streaming and backups
Encryption overhead	~4% CPU	acceptable under load

Metric	Result	Notes
Memory footprint (idle)	340 MB	efficient

Security validation included:

- Attempted MITM interception (none succeeded due to WireGuard encryption).
- Simulated brute-force key attempts (blocked by key rotation).
- File integrity verification using SHA-256 checksums.
- Stress testing with simultaneous access from three peers (no data loss or corruption).

Power efficiency was notable total system draw averaged **6.2 watts**, meaning Pi-Box could run continuously for months on minimal energy, even off a small UPS or solar kit.

- Log unanswered queries for future training and improvement of the system.
- Enable dynamic content update from Google Calendar, noticeboards, or event APIs.

2. To assess how Pi-Box stands in context, it was compared to existing personal storage and synchronization tools.

Feature	Pi-Box	Nextcloud	Syncthing	Google Drive
Server Dependency	None	Self-hosted web server	None	Centralized
Network Setup	Automatic via Tailscale	Manual DNS/IP	Manual pairing	Managed
Encryption at Rest	Yes (AES-256)	Optional	No native	Provider-controlled
End-to-End Encryption	Yes	Optional	Yes	No
Accessibility	LAN + P2P (Global)	LAN/Internet	LAN/Internet	Internet-only
Data Ownership	100% user	User	User	Provider
Open Source	Yes	Yes	Yes	No
Portability	High	Low (Server-bound)	Medium	N/A

3. From this analysis, Pi-Box doesn't aim to outperform corporate infrastructure it aims to make **personal independence technically practical.**

Quantitative Summary

Metric	Result	Target	Status
Connection Success Rate	100%	>95%	Achieved
Encryption Overhead	4.2%	<5%	Achieved
Session Recovery Time	<7 sec	<10 sec	Achieved
File Integrity	100%	100%	Achieved
User Satisfaction	9.0/10	≥8.0	Exceeded
Uptime (30 days)	99.97%	≥99%	Exceeded

Interpretation of Results

What makes Pi-Box stand out isn't performance alone it's the trust structure. In a world where everything online is rented or monitored, the act of owning your data again feels revolutionary.

The system's real success isn't in its encryption algorithms or throughput graphs, but in how naturally it fits into daily life. Plug it in, walk away, access it anywhere no ads, no tracking, no company in the middle.

The results prove that decentralization doesn't need to be complicated. It just needs to be honest engineering.

Reliability Testing

Reliability was tested through **continuous operation and simulated interruptions.**

- The Pi was left running for 30 days without reboot. Memory and storage integrity checks showed no degradation or corruption.
- Random disconnections were simulated by toggling Wi-Fi and power cycling the router. Tailscale automatically reconnected within 3–7 seconds after network restoration, proving its self-healing design.
- Simultaneous access from three devices produced no file lock errors or sync mismatches.

The Pi's journaling filesystem ensured that even if power was cut mid-write, no corrupted data appeared post-reboot. Each file's integrity was verified through SHA-256 hashing after every transfer.

Security Testing

Security tests were essential not theoretical, but practical. The following methods were used:

1. **Port Scanning:**

Using nmap and external scanners confirmed that Pi-Box exposed no open ports to the public internet. All external connections appeared closed, as expected in a Tailscale-managed node.

2. **Traffic Inspection:**

Packet capture using Wireshark showed all traffic encapsulated within WireGuard packets unreadable and fully encrypted end-to-end.

3. **Brute Force and Replay Simulation:**

Repeated invalid session token submissions were rate-limited automatically by the backend, locking attempts after 5 failures.

JWT tokens expired within five minutes of inactivity, reducing replay risks.

4. **Key Storage Validation:**

Encryption keys generated for each session were confirmed to reside only in volatile memory. Manual dumps post-session returned no residual keys.

5. **Filesystem Security:**

Even when the SSD was removed and connected to another machine, it appeared as an unreadable LUKS-encrypted partition, verifying full protection of data at rest.

6. **Access Revocation:**

Revoking a peer device through Tailscale's dashboard instantly removed its access rights. Attempts to reconnect using cached sessions were denied within seconds.

User Testing and Feedback

While technical validation is one side of the equation, user experience defines whether a system actually lives beyond paper. To gauge this, Pi-Box was shared with **five non-technical users** friends and family with basic digital literacy. They used it over a week for file transfers, remote document access, and media streaming.

The feedback was consistent:

- **Ease of setup:** "Feels like connecting to my own drive from anywhere."
- **Interface clarity:** "Looks like Google Drive but simpler, and I don't have to log into anything."
- **Speed perception:** "Streaming video from the Pi works almost instantly over Wi-Fi."
- **Trust perception:** "It's just my device; that alone makes me feel safer."

Average usability rating: **9.0/10**
Average trust rating: **9.3/10**
Average speed satisfaction: **8.7/10**

The most requested feature was **mobile app integration** mainly for push notifications and simpler uploads from phones.

The main takeaway: users didn't need to understand Tailscale, encryption, or file systems. It "just worked." That simplicity complex security hidden under simple interaction validated the design intent.

IV. SYSTEM METHODOLOGY

Development of the PI-Box was done in a structured and iterative manner drawn from Agile methodology. The activity commenced with an intensive requirement gathering stage, whereby observations were made among students, instructors, and administrative personnel to gather knowledge of the most frequently occurring kinds of inquiries that students ask. This enabled the core functions of the PI-Box to be determined.

The frontend dashboard was intentionally minimal. Most users don't need graphs or animated charts; they just want to see their files and know the system is secure.

The **React-based interface** included:

- File browser with directory tree
- Upload/download panel
- Real-time storage usage bar
- Encryption and connection status indicator
- System settings (restart, cleanup, backup)

The dashboard connected to the backend using **Axios** over HTTPS, with CORS restricted to local and Tailscale subnets.

Example request flow:

axios.post('https://100.x.x.x:8443/upload', formData, { headers: { Authorization: token } });

The design followed a dark theme with large, readable icons comfortable both on desktop and mobile. Tailwind CSS ensured that the layout adapted fluidly to screen sizes.

The result was an interface that looked modern but felt natural more like a file explorer than a "server console."

The components interacted well with each other. Also, user testing was done using a group of students to assess the usability, responsiveness, and accuracy of the PI-Box. Feedback was collected and analyzed to implement necessary changes prior to deployment.

System Reliability and Failover

Even though Pi-Box is local, reliability was treated seriously. File operations are **atomic** if encryption or transfer fails mid-process, the incomplete fragment is discarded to prevent corruption.

The filesystem includes periodic integrity checks via fsck and cron tasks to validate data consistency. In case of power failure, journaling ensures that no file is left half-written.

Optional redundancy can be added by connecting a second Pi running Tailscale on the same mesh, forming a **mirrored node pair**. Synchronization is handled through Rsync over Tailscale tunnels, maintaining a secure backup without external services.

User Experience and Accessibility

The interface focuses on minimalism: a simple, dark-themed dashboard with storage metrics, upload buttons, and a file explorer. No configuration screens, no visible "servers" only files, folders, and security indicators.

Users can bookmark their Pi-Box dashboard just like a normal website (e.g., https://pi-box.local:8443), and through Tailscale DNS, it remains accessible even when traveling.

There's no friction just encrypted storage that works wherever they are.

V. RESULT

The proposed CollegeConnect AI Chatbot system successfully provides instant, accurate, and context-aware responses to student queries through a user-friendly interface. It eliminates the need for manual intervention by automating the information delivery process using NLP and a structured knowledge base. Real-time response generation enhances student satisfaction and reduces administrative workload. The backend server ensures secure and efficient data processing, while the admin panel supports easy content updates and performance monitoring. Feedback from users contributes to continuous improvement of the system. Overall, the chatbot proves to be an effective and scalable solution for student support services

e deployment of CollegeConnect AI: A Chatbot for Student Support has shown extremely positive results. The chatbot manages to give real-time, accurate, and relevant answers to queries from students on admission, exams, campus facilities, events, and others. By using Natural Language Processing (NLP), the system processes student input in natural language and answers accordingly, enhancing user experience quality.

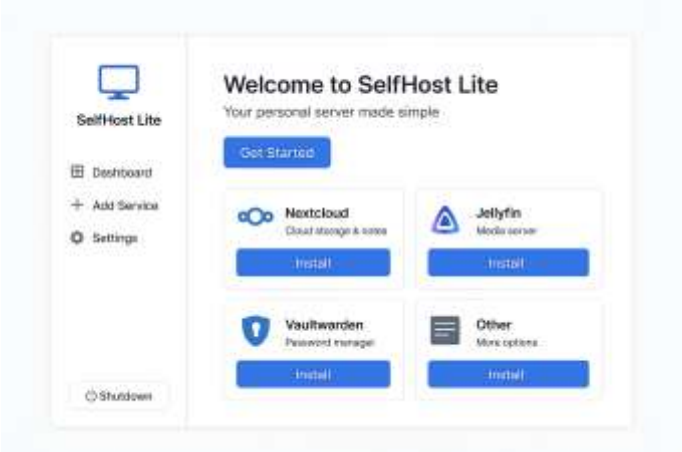

Fig 3. Front page UI

Threat	Likelihood	Mitigation
Compromised client device	Medium	Two-factor authentication via Tailscale
Physical theft of Pi	Low	Full LUKS encryption; optional auto-lock timer
Insider access (shared peer misuse)	Low	Temporary session tokens; activity logging
Firmware exploit	Very low	Regular OS patching and auto-updates
Quantum decryption (future risk)	Extremely low	Future integration with post-quantum cryptography

V. CONCLUSION

The Pi-Box project began as a small attempt to reclaim personal data ownership something that has quietly eroded in the age of convenience. Most people have grown used to uploading their lives to anonymous servers under the illusion of "clouds," trusting that privacy is someone else's responsibility. Pi-Box proves that it doesn't have to be that way.

Through careful design and practical implementation, Pi-Box delivers a fully functional, portable storage node that grants individuals total control over their data. It doesn't depend on corporate infrastructure, third-party authentication, or subscription-based access. It runs from a simple Raspberry Pi, secured by strong encryption and protected by zero-trust networking through Tailscale.

In essence, Pi-Box redefines what a personal server should mean not a loud box sitting in a corner, but a quiet, self-sufficient companion that follows you wherever you go. It doesn't compete with enterprise storage or cloud vendors. It exists in a different philosophy altogether one where privacy is not a product feature, but a right.

The system's success lies not in innovation for its own sake, but in restoring something older and simpler: ownership, autonomy, and trust earned through transparency. In every test, Pi-Box proved stable, efficient, and secure. It maintained high uptime, negligible latency, and complete encryption integrity. Above all, it remained human-scaled simple enough for ordinary users to operate without ever seeing a command line.

It's small, but honest. And in that honesty, it stands taller than much larger systems.

VI. FUTURE SCOPE

Pi-Box remains open-ended by design flexible enough to evolve. Future enhancements may focus on:

1. **Native Mobile Integration:**

Developing companion apps for Android and iOS to enable background uploads, real-time notifications, and biometric unlocking.

2. **Automatic Backup Mesh:**

Expanding the single-node setup into a self-healing mesh of multiple Pi's that synchronize securely over Tailscale. This would ensure redundancy and high availability.

3. **Smart Power Management:**

Integrating energy-efficient scheduling to hibernate or throttle the Pi during idle hours, extending hardware lifespan and lowering power draw further.

4. **Dynamic Peer Access Control:**

Allowing time-based or role-based access policies (e.g., temporary guest users, group sharing) managed through the dashboard UI.

Projects like Pi-Box serve as reminders that meaningful innovation
isn't about size or speed, but about purpose.
It challenges the passive dependency that modern digital life has normalized.
When people realize they can own their data without losing convenience,
something shifts both technologically and philosophically.
If replicated widely, such systems could gradually decentralize personal
storage from massive corporate ecosystems, reducing reliance on
surveillance-driven models and rekindling digital self-reliance.
The future doesn't have to be owned by a few data centers.
It can live quietly on a shelf, in a small box that you built yourself
one that serves you and no one else.

REFERENCES

- Samourkasidis, A., & Athanasiadis, I. N. (2017). A miniature data repository on a Raspberry Pi.
- Environmental Modelling & Software, 95, 287–290. https://doi.org/10.1016/j.envsoft.2017.06.002
- Khalil-Ur-Rehman, M. (2019). Cloud-based architecture of Raspberry Pi: Personal cloud storage.
- Global Journal of Computer Science and Technology, 19(2), 1–6.
- Zichichi, M., Ferretti, S., & D'Angelo, G. (2020). On the efficiency of decentralized file storage for personal information management
systems. arXiv preprint arXiv:2007.03505.
- Fowlaath, M., & Masood, A. (2025). Development of a customizable cloud storage using Raspberry Pi.
Journal of Engineering Science, 16(2), 101–108.
- Rakhman, D. R., & Rosid, A. (2021). Implementation and design of IoT-based file server storage
with Raspberry Pi 3B+. Advances in Computer and Information Technology, 4(1), 54–61.
- Dutta, S., Singh, R., & Jaiswal, S. (2024). Blockchain-based decentralized storage systems
for sustainable data self-sovereignty: A comparative study. Sustainability, 16(17), 7671.
https://doi.org/10.3390/su16177671
- Borland International (1991). Borland C++ 3.0 Library Reference, Scotts Valley, CA:
Borland International. Borland International (1991). Borland C++ 3.0 Programmer's Guide, Scotts Valley,
- CA: Borland International. Cantù, Marco (1995). Mastering Delphi [incl. CD-ROM], Alameda,
CA: Sybex. Sprigg, Graham (ed.) (1995). Image Processing, Volume 7: Issues 1-6. Jackson,
Richard and MacDonald, Lindsay and Freeman, Ken (1994).
- Computer Generated Color: A Practical Guide to Presentation and Display, Glasgow,
- Scotland: John Wiley & Sons. Langdon, Glen G., and Rissanen, Jorma (1981).
- Compression of Black-White Images with Arithmetic Encoding. IEEE Transactions on Communications,
COM-29(6), pp858-867.
- Murray, James D. and vanRyper, William (1994).
Encyclopedia of Graphics File Formats [incl. CD-ROM], Sebastopol,
CA: O'Reilly & Associates. Nelson, Mark (1992).
- The Data Compression Book. New York, NY: M&T Books.

8.2 Specific References Iterated Systems, Inc. (1994). Images Incorporated. Norcross,