PROJECT DOCUMENTATION

# Debian Home Cloud Storage Server

*A Self-Hosted, Globally Accessible Family Cloud Storage Solution*

Submitted by

**Ayushman Sahoo**

BS in Electronic Systems — 1st Year

**Indian Institute of Technology Madras**

Date: March 2026

Domain: storagepaglu.pp.ua

Total Project Cost: ₹6,000

# Abstract

This document presents the complete design, implementation, and technical documentation of a self-hosted home cloud storage server built using repurposed hardware and free, open-source software. The project was conceived to solve a real-world storage problem faced by a family member and was executed on a total budget of under ₹6,000, including hardware, drives, networking equipment, and a paid domain name.

The server runs Debian 12.6 (Bookworm) on a repurposed early-2000s Intel Core 2 Duo desktop PC. Nextcloud is deployed natively using Apache, PHP, and MariaDB to provide a Google Drive and Google Photos-like experience. Storage is managed across two physical hard drives pooled together using MergerFS, with a separate cold backup disk on a weekly automated rsync schedule. Global internet access is achieved without port forwarding through a Cloudflare Tunnel bound to a paid custom domain (storagepaglu.pp.ua), providing a permanent, stable HTTPS URL.

Family members access the server from Android devices using the Nextcloud mobile app and the Memories app for automatic photo and video backups. The entire system is currently live and operational, serving as a private family cloud accessible from anywhere in India.

# 1. Introduction

## 1.1 Motivation and Background

In today's digital life, the storage of high-resolution photos and videos has become a significant concern for households using low-end or older smartphones. My elder brother, who lives in a college hostel, preferred saving every photograph in full HD quality but owned a phone with only 64 GB of internal storage. His Google Drive was quickly filling up with WhatsApp backups and photos, making a Google One subscription seem necessary.

Rather than paying for cloud storage that felt unnecessary and expensive, I had been watching videos of people building their own NAS (Network-Attached Storage) and home cloud servers. Most of these builds used expensive, brand-new hardware well beyond what was needed for a family use case. This gave me the idea: could I build a full-featured cloud storage system at a fraction of the cost, using second-hand components and free software?

This project is the result of that question. It is a completely self-built, self-hosted, globally accessible cloud storage server — running on hardware that cost less than ₹4,000 from a Facebook Marketplace listing, serving the family as a private alternative to Google Drive and Google Photos.

### 1.2 Initial Prototype: FileBrowser on a Laptop

Before acquiring dedicated hardware, the first version of this project used a partition on my existing laptop's HDD. The empty 360 GB partition was used to run FileBrowser — a lightweight, open-source web-based file manager. Family members connected to the same local network could access the server successfully.

However, this approach had two major limitations. First, it only worked on the local network, meaning my brother in his hostel could not access it. Second, the server was only available when the laptop was running and connected to Wi-Fi. A Wi-Fi disconnection would immediately take the server offline. This led to the need for a dedicated, 24/7 running machine and a proper global access solution.

### 1.3 Project Scope and Goals

- Build a dedicated home cloud server using second-hand hardware at minimum cost
- Deploy Nextcloud for a Google Drive and Google Photos-like experience
- Support multiple family member accounts with automatic photo and video backup
- Enable global access from anywhere without port forwarding on the router
- Implement automated weekly cold backups to a separate hard drive
- Keep total project expenditure under ₹6,000

## 2. Hardware Procurement and Setup

### 2.1 PC Acquisition

The hardware search began online and in local second-hand markets. New mini PCs and second-hand laptops were priced at a minimum of ₹10,000, which was well above the target budget. The breakthrough came from a Facebook Marketplace listing: a postal office was being cleared, and two complete PC setups were listed for sale.

The listing offered a complete desktop setup — monitor, UPS, PC with 500 GB HDD, Intel Core 2 Duo processor, 2×1 GB DDR2 RAM, keyboard, and mouse — for just ₹3,500. Upon inspection at the site, one of the two PCs also had a 128 GB SSD installed. That unit was chosen and purchased.

> **Hardware Find**
> A postal office clearance sale on Facebook Marketplace was the key to keeping costs under budget. Two complete PC setups were listed, and the one with the 128 GB SSD was selected for ₹3,500 — including UPS, monitor, keyboard, and mouse.

### 2.2 Hard Drive Acquisition

A second Facebook Marketplace seller was found nearby, selling used hard drives at ₹300 for 500 GB. Two 1 TB drives were purchased — one for live storage and one for weekly backup. The seller also included spare screws and SATA cables at no extra charge.

Upon testing at home, one of the purchased HDDs turned out to be defective: it would crash and fail when transferring files larger than 6 GB. The seller was contacted, negotiated with at half the original price, and a replacement was provided. During this visit, the disk was verified using CrystalDiskInfo, and two additional DDR2 2 GB RAM sticks were also purchased for ₹150.

### 2.3 Networking

The PC had no Wi-Fi capability and only an integrated Realtek 10/100 Ethernet NIC. The Ethernet connection proved unstable (detailed in Section 6), so a TP-Link TL-WN725N USB Wi-Fi adapter (Realtek RTL8188EUS, USB ID 0bda:8179) was purchased to provide reliable wireless connectivity.

### 2.4 Final Hardware Specification

| Component | Specification | Source |
|---|---|---|
| CPU | Intel Core 2 Duo (early generation) | Facebook Marketplace (₹3,500 bundle) |
| RAM | 4 GB DDR2 (2×2 GB sticks) | ₹150 (2 sticks) + original 2×1 GB |
| System Drive | 128 GB SSD | Included in PC bundle |
| Live Storage Drive | 1 TB Toshiba 2.5" HDD | Facebook Marketplace (₹300 each) |
| Live Storage Drive 2 (Removed) | 500 GB Seagate HDD (faulty) | Included in PC bundle |
| Backup Drive | 1 TB 3.5" HDD | Facebook Marketplace (₹300 each) |
| Wi-Fi Adapter | TP-Link TL-WN725N (RTL8188EUS) | Separate purchase |
| PSU | 350W ATX | Included in PC bundle |
| UPS | Standard UPS unit | Included in PC bundle |
| Monitor/KB/Mouse | Full desktop set | Included in PC bundle |

Note: The original 500 GB Seagate HDD included in the bundle was found to be faulty and was removed from the system. The final live storage pool consists of the two purchased 1 TB drives, and the 128 GB SSD serves as the system drive.

## 3. Operating System Installation

### 3.1 OS Selection: Debian 12.6 Bookworm

Debian 12.6 (Bookworm) was selected as the server operating system for its long-term support, stability, low resource overhead, and suitability for a 24/7 server environment. XFCE was chosen as the optional desktop environment since it is extremely lightweight and can be disabled when not in active use, preserving RAM for Nextcloud and other services.

## 3.2 Bootable Media Creation

- Downloaded Ventoy from ventoy.net and extracted it on a Windows PC
- Mounted a USB drive (minimum 4–5 GB free space) and ran Ventoy2Disk.exe to install Ventoy onto it
- Downloaded the Debian 12.6 DVD ISO from debian.org
- Copied the ISO file directly to the Ventoy USB drive

## 3.3 BIOS and Installation

- Inserted USB into the server PC and entered BIOS by pressing Delete repeatedly on startup
- Under Advanced BIOS Settings, set the USB as the 1st Bootable Device
- Pressed F10 to save and exit, then selected the Debian ISO in the Ventoy GUI
- Chose Graphical Installation from the Debian installer menu
- Selected XFCE as the desktop environment, GRUB as the bootloader, and enabled SSH server during installation

## 3.4 Disk Partitioning

The 128 GB SSD was partitioned as follows during installation:

| Partition | Size | Format | Mount Point |
|---|---|---|---|
| Root | 100 GB | ext4 | / |
| Swap | 3 GB | swap | [swap] |
| Home | 23 GB | ext4 | /home |

## 3.5 Post-Installation System Update

```
sudo apt update && sudo apt full-upgrade -y
sudo apt install -y htop curl unzip git ufw smartmontools rsync
```

XFCE and the display manager can be started manually when needed for GUI configuration and disabled at other times to save RAM, keeping the server lean for Nextcloud operation.

# 4. Storage Configuration

## 4.1 Storage Architecture Overview

The storage design prioritises maximum usable space while protecting against complete data loss. Two 1 TB hard drives are combined into a 2 TB live storage pool using MergerFS, a

FUSE-based union filesystem. A cold backup strategy is used: data from the live pool is rsync'd to a separate physical drive every Saturday at 02:00, and that drive is unmounted and effectively spun down after each backup.

**Storage Layout**

SSD (128 GB) → System + Nextcloud app + Database + Cache HDD A (1 TB) + HDD B (500 GB) → MergerFS live pool at /srv/data (≈1.5 TB) HDD C (1 TB) → Cold backup target, mounted only during weekly backup

## 4.2 Drive Identification and Formatting

Drives were first identified using lsblk and fdisk, then formatted with the ext4 filesystem and labelled for reliable mounting:

```
# Identify disks
lsblk -o NAME,SIZE,TYPE,MOUNTPOINT
sudo /sbin/fdisk -l | sed -n '1,200p'

# Format and label each drive
sudo /sbin/mkfs.ext4 -F -L DATA1 /dev/sdX
sudo /sbin/mkfs.ext4 -F -L DATA2 /dev/sdY
sudo /sbin/mkfs.ext4 -F -L BACKUP1 /dev/sdZ

# Get UUIDs for fstab entries
sudo /sbin/blkid | grep -E 'DATA1|DATA2|BACKUP1'
```

## 4.3 Persistent Mounting via /etc/fstab

The drives were added to /etc/fstab using their UUIDs for reliable mounting across reboots. The noatime option was applied to reduce unnecessary write operations on the hard drives:

```
UUID=<DATA1_UUID>   /srv/dataA  ext4  defaults,noatime  0  2
UUID=<DATA2_UUID>   /srv/dataB  ext4  defaults,noatime  0  2
```

## 4.4 MergerFS Pool Creation

MergerFS merges /srv/dataA and /srv/dataB into a single unified mount point at /srv/data. Files are written to the drives using the 'first found' (ff) policy, meaning each file is stored entirely on a single drive rather than being split. This makes data recovery simpler if one drive fails.

```
# Install MergerFS
sudo apt install -y mergerfs

# Add pool line to /etc/fstab
/srv/dataA:/srv/dataB  /srv/data  fuse.mergerfs  \
  defaults,allow_other,use_ino,cache.files=off,category.create=ff  0  0

# Mount and verify
sudo mount -a
df -h /srv/data    # Should show ~1.4-1.5 TB total
```

### 4.5 Automated Weekly Cold Backup

A bash script and a systemd timer were created to automate weekly backups. The backup drive (BACKUP1) is mounted, rsync is run from /srv/data to /srv/backup/pool, and the drive is then unmounted. The timer fires every Saturday at 02:00 with a random 15-minute delay to avoid predictable system load.

```
# Backup script: /usr/local/bin/run-backup.sh
#!/bin/bash
set -euo pipefail
BACKUP_DEV="/dev/disk/by-label/BACKUP1"
BACKUP_MNT="/srv/backup"
SRC="/srv/data"
DEST_DIR="$BACKUP_MNT/pool"

mount "$BACKUP_DEV" "$BACKUP_MNT"
mkdir -p "$DEST_DIR"
rsync -aHAX --delete --info=progress2 --numeric-ids "$SRC"/ "$DEST_DIR"/
sync
umount "$BACKUP_MNT" || umount -l "$BACKUP_MNT"
```

```
# Systemd timer: every Saturday at 02:00
[Timer]
OnCalendar=Sat 02:00
RandomizedDelaySec=15min
Persistent=true

# Enable the timer
sudo systemctl daemon-reload
sudo systemctl enable --now run-backup.timer
```

## 5. Network Configuration

### 5.1 Ethernet Instability Problem

The integrated Realtek r8169/RTL8105e NIC on the motherboard proved to be unreliable from the start. Although autonegotiation initially reported 100 Mb/s Full duplex, the link would repeatedly flap, and the connection would fall back to 10 Mb/s Full. The dmesg log showed continuous “Link is Down/Up” messages. Swapping cables and router ports did not resolve the issue.

Diagnosis pointed to a PHY-level compatibility issue between this specific NIC and the router. While forcing 100/Full at both ends was attempted, the only stable speed was 10 Mb/s, which was insufficient for the project’s needs. The decision was made to add a USB Wi-Fi adapter as the primary network interface.

```
# Diagnostic commands used
sudo ethtool enp2s0
sudo ethtool -s enp2s0 autoneg on advertise 0x0F
sudo dmesg | grep -i -e r8169 -e link -e enp2s0
```

### 5.2 Wi-Fi Adapter Driver Installation (RTL8188EUS)

A TP-Link TL-WN725N adapter (Realtek RTL8188EUS) was connected. The kernel’s built-in r8188eu and rtl8xxxu drivers underperformed with this chipset, so a maintained community DKMS driver from the aircrack-ng repository was installed instead.

```
# Confirm adapter is detected and USB 2.0 speed (480M)
lsusb                       # Expect: Realtek 0bda:8179
sudo lsusb -t               # Expect: 480M for Wi-Fi device

# Enable non-free firmware repository
# Edit /etc/apt/sources.list to add: main contrib non-free-firmware
sudo apt update
sudo apt install -y firmware-realtek firmware-misc-nonfree \
  network-manager wpasupplicant wireless-tools \
  git build-essential dkms linux-headers-$(uname -r)

# Blacklist conflicting in-kernel modules
sudo modprobe -r r8188eu 2>/dev/null || true
sudo modprobe -r rtl8xxxu 2>/dev/null || true
echo "blacklist r8188eu" | sudo tee /etc/modprobe.d/blacklist-r8188eu.conf

# Clone and install DKMS driver
cd ~
git clone https://github.com/aircrack-ng/rtl8188eus.git
cd rtl8188eus
sudo dkms add ./
sudo dkms build rtl8188eus/$(awk -F'"' '/PACKAGE_VERSION/ {print $2}' dkms.conf)
sudo dkms install rtl8188eus/$(awk -F'"' '/PACKAGE_VERSION/ {print $2}' dkms.conf)
sudo modprobe 8188eu

# Verify
lsmod | grep 8188eu
```

## 5.3 Wi-Fi Connection and Performance Tuning

```
# Connect to Wi-Fi network
nmcli radio wifi on
nmcli dev wifi list
nmcli dev wifi connect "SSID" password "PASSWORD"

# Disable Wi-Fi power saving for stable server operation
IFACE=$(iw dev | awk '/Interface/ {print $2; exit}')
sudo iw dev "$IFACE" set power_save off

# Persist power save setting across reboots
sudo mkdir -p /etc/NetworkManager/conf.d
printf "[connection]\nwifi.powersave = 2\n" | \
  sudo tee /etc/NetworkManager/conf.d/wifi-powersave.conf
sudo systemctl restart NetworkManager

# Persist driver module load
echo "8188eu" | sudo tee /etc/modules-load.d/8188eu.conf
```

With the DKMS driver and power save disabled, real-world Wi-Fi throughput on the 2.4 GHz N150 connection reached 25–60+ Mb/s, well above the previous 10 Mb/s wired ceiling. The router channel was fixed to channel 6 with 20 MHz width for stability, and a short USB extension cable was used to position the nano adapter away from the PC case for better signal quality.

# 6. SSH and Remote Management

## 6.1 SSH Server Setup

```
sudo apt update
sudo apt install -y openssh-server
sudo systemctl enable --now ssh
systemctl status ssh
hostname -I   # Note the LAN IP for connecting
```

## 6.2 SSH Key Authentication

SSH key-based authentication was configured from the Windows laptop (using WSL) to eliminate the need for passwords and improve security:

```
# On WSL (laptop side)
ssh-keygen -t ed25519 -C "laptop-key"
ssh-copy-id -p 22 homeserver@192.168.1.9
ssh homeserver@192.168.1.9   # Test login
```

## 6.3 Hardening: Custom Port and Firewall

```
# Edit SSH config
sudo cp /etc/ssh/sshd_config /etc/ssh/sshd_config.bak
sudo nano /etc/ssh/sshd_config
  # Set: Port 2222
  # Set: PasswordAuthentication no

sudo systemctl restart ssh
sudo ufw allow 2222/tcp
sudo ufw reload

# Block ICMP pings (makes server invisible to ping sweeps)
echo "net.ipv4.icmp_echo_ignore_all=1" | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
```

## 6.4 VS Code Remote-SSH Integration

VS Code on Windows was configured to use WSL's SSH binary for connecting to the server, allowing full remote file editing via the VS Code interface. The SSH config was placed inside WSL at ~/.ssh/config:

```
# ~/.ssh/config (inside WSL)
Host OurServer
  HostName 192.168.1.9
  Port 2222
  User homeserver
  IdentityFile ~/.ssh/id_ed25519

# VS Code setting: Remote.SSH: Path
# Set to: C:\\Windows\\System32\\wsl.exe ssh
```

### 6.5 Cockpit Web UI (On-Demand)

Cockpit was installed for browser-based server monitoring and management but configured to be disabled by default to preserve RAM. It is started manually only when needed:

```
sudo apt install -y cockpit
sudo systemctl disable --now cockpit.socket   # Off by default

# Start when needed (access at https://192.168.1.9:9090)
sudo systemctl enable --now cockpit.socket

# Stop when done
sudo systemctl disable --now cockpit.socket
```

## 7. Nextcloud Installation and Configuration

### 7.1 Why Nextcloud

Nextcloud was chosen as the core application for its Google Drive-like web interface, native Android and iOS apps, support for automatic photo and video backup (similar to Google Photos), multi-user account management, and the Nextcloud Memories app which provides a Google Photos-style timeline view. It is completely free and open-source.

### 7.2 Dependency Installation

```
sudo apt update
sudo apt install -y apache2 mariadb-server libapache2-mod-php \
  php php-cli php-fpm php-mysql php-gd php-curl php-xml \
  php-mbstring php-zip php-bcmath php-intl php-imagick \
  php-redis php-bz2 php-common php-ldap php-gmp php-apcu \
  unzip redis-server
```

### 7.3 Nextcloud Download and Placement

```
cd /tmp
wget https://download.nextcloud.com/server/releases/nextcloud-26.0.4.zip
unzip nextcloud-26.0.4.zip
sudo mv nextcloud /srv/data/nextcloud
```

### 7.4 File Permissions

A shared group called ‘nextcloud’ was created and both the system user (homeserver) and the Apache web server user (www-data) were added to it. This allows both users to read and write to the Nextcloud data directory, enabling files to be added manually via the file manager while still appearing in the Nextcloud web UI after a rescan.

```
# Create shared group and add users
sudo groupadd nextcloud
sudo usermod -aG nextcloud homeserver
sudo usermod -aG nextcloud www-data
```

```
# Set ownership and permissions with setgid bit
sudo chown -R homeserver:nextcloud /srv/data/nextcloud
sudo find /srv/data/nextcloud -type d -exec chmod 2770 {} \;
sudo find /srv/data/nextcloud -type f -exec chmod 660 {} \;
```

## 7.5 MariaDB Database Setup

```
sudo systemctl start mariadb
sudo mysql_secure_installation

# Inside MariaDB prompt:
CREATE DATABASE nextcloud;
CREATE USER 'ncuser'@'localhost' IDENTIFIED BY 'your_strong_password';
GRANT ALL PRIVILEGES ON nextcloud.* TO 'ncuser'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```

## 7.6 Apache Virtual Host Configuration

A virtual host configuration file was created for Nextcloud at /etc/apache2/sites-available/nextcloud.conf:

```
Alias /nextcloud "/srv/data/nextcloud/"

<Directory /srv/data/nextcloud/>
  Require all granted
  Options FollowSymlinks MultiViews
  AllowOverride All
  <IfModule mod_dav.c>
    Dav off
  </IfModule>
</Directory>

<Directory /srv/data/nextcloud/data/>
  Require all denied
</Directory>

ErrorLog ${APACHE_LOG_DIR}/nextcloud_error.log
CustomLog ${APACHE_LOG_DIR}/nextcloud_access.log combined
```

```
# Enable required modules and site
sudo a2enmod rewrite headers env dir mime setenvif ssl
sudo a2ensite nextcloud
sudo systemctl restart apache2
```

## 7.7 PHP Tuning for Large File Uploads

PHP limits were raised in /etc/php/8.X/apache2/php.ini to support large file uploads (photos, videos, documents):

```
upload_max_filesize = 25G
post_max_size       = 25G
max_execution_time  = 3600
max_input_time      = 3600
memory_limit        = 512M
```

```
sudo systemctl restart apache2
```

## 7.8 Redis Caching

Redis was enabled for Nextcloud's memory caching to significantly improve performance, especially on the limited hardware:

```
sudo apt install -y redis-server php-redis

# Add to /etc/php/8.X/apache2/php.ini:
extension=redis.so

sudo systemctl restart apache2
```

## 7.9 Manual File Scanning

When files are added directly to the server via SSH or the file manager (bypassing the Nextcloud web UI), Nextcloud's database must be updated with a scan command. This was necessary during initial setup to make manually copied files visible in the web interface:

```
sudo -u homeserver php /srv/data/nextcloud/occ files:scan homeserver
```

# 8. Global Internet Access

## 8.1 The Challenge

This project faced three compounding challenges for global access: the ISP provides only a dynamic IP address that changes periodically, the router does not allow port forwarding (many ISPs in India provide CGNAT addresses), and a free Cloudflare Tunnel generates a new random URL every time the service is restarted (when used in quick-run ephemeral mode).

The solution required two components working together: a persistent named Cloudflare Tunnel bound to a stable, paid domain, so the URL never changes regardless of IP address or service restarts.

## 8.2 Domain Selection and Purchase

To get a permanent URL, a paid domain was required. After researching cheap TLDs, the domain storagepaglu.pp.ua was registered from Regery.com for just ₹120. The .pp.ua TLD (a free second-level domain under Ukraine's .ua ccTLD) is accepted by Cloudflare and fully compatible with Cloudflare Tunnel for personal use.

**Domain Details**

Domain: storagepaglu.pp.ua Registrar: Regery.com Cost: ₹120 Status: Active, nameservers pointed to Cloudflare

## 8.3 Cloudflare Tunnel Setup

The domain was added to Cloudflare and its nameservers were updated to Cloudflare's. A named tunnel was then created on the server. Because the tunnel is persistent and named (not ephemeral), the URL remains the same even after the tunnel service restarts, the machine reboots, or the public IP changes.

```
# Install cloudflared
curl -L https://pkg.cloudflare.com/cloudflare-main.gpg | \
  sudo tee /usr/share/keyrings/cloudflare-main.gpg > /dev/null
echo 'deb [signed-by=/usr/share/keyrings/cloudflare-main.gpg] \
  https://pkg.cloudflare.com/cloudflared any main' | \
  sudo tee /etc/apt/sources.list.d/cloudflared.list
sudo apt update && sudo apt install -y cloudflared

# Authenticate and create a named tunnel
cloudflared tunnel login
cloudflared tunnel create homeserver-tunnel

# Route the domain to the tunnel
cloudflared tunnel route dns homeserver-tunnel storagepaglu.pp.ua

# Configure tunnel to point to local Nextcloud
# ~/.cloudflared/config.yml:
tunnel: <TUNNEL-UUID>
credentials-file: /home/homeserver/.cloudflared/<UUID>.json

ingress:
  - hostname: storagepaglu.pp.ua
    service: http://localhost:80
  - service: http_status:404

# Run tunnel as a system service
sudo cloudflared service install
sudo systemctl enable --now cloudflared
```

## 8.4 How It Works

The cloudflared daemon on the server creates a persistent outbound connection to Cloudflare's edge network. All incoming requests to storagepaglu.pp.ua are received by Cloudflare, routed through this outbound tunnel connection, and delivered to the local Apache web server running Nextcloud. This means:

- No inbound ports need to be opened on the router
- No port forwarding is required
- The URL is permanent and does not change on restarts
- Cloudflare provides HTTPS termination automatically
- The dynamic IP address is completely irrelevant

**Architecture**

Internet User → storagepaglu.pp.ua → Cloudflare Edge ↓ (persistent outbound tunnel) cloudflared daemon on server → Apache (localhost:80) → Nextcloud

# 9. Mobile Client Setup

## 9.1 Android App Installation

Family members access the server using two apps from the Google Play Store:

- Nextcloud — the main app for file access, folder browsing, uploads, and account management
- Nextcloud Memories — a companion app providing a Google Photos-like timeline interface for photos and videos

## 9.2 Auto Photo and Video Backup

The Nextcloud Android app supports automatic background upload of photos and videos, replicating the Google Photos automatic backup experience. After logging in with the server URL (storagepaglu.pp.ua), each family member's account was configured for:

- Auto Upload enabled for the DCIM/Camera folder
- Upload restricted to Wi-Fi only (to avoid mobile data usage)
- Upload restricted to when the device is charging (optional, recommended for large batches)
- Nextcloud Memories app configured to display photos in a timeline organised by date

## 9.3 User Accounts

Each family member has a separate Nextcloud account with their own storage quota. The Nextcloud admin panel was used to create accounts and set individual permissions. The server currently serves as the primary photo and file backup destination for the family, replacing Google Drive and reducing dependence on Google's subscription services.

# 10. Problems Encountered and Resolutions

## 10.1 Defective Hard Drive

**Symptom:** One of the purchased HDDs failed when transferring files larger than 6 GB.

**Resolution:** The seller was contacted and negotiated with; the drive was replaced at half price. The replacement was verified using CrystalDiskInfo before installation. The original 500 GB Seagate HDD from the PC bundle was also found to be faulty and was removed from the system entirely.

## 10.2 Ethernet NIC Instability

**Symptom:** The integrated Realtek NIC repeatedly flapped between 100 Mb/s and 10 Mb/s, making wired networking unreliable.

**Root Cause:** PHY-level compatibility issue between the NIC chipset (r8169/RTL8105e) and the router. Cable swaps and router port changes had no effect.

**Resolution:** Replaced wired networking with a TP-Link TL-WN725N USB Wi-Fi adapter running a community DKMS driver (rtl8188eus from aircrack-ng). Achieved 25–60 Mb/s throughput, exceeding the 10 Mb/s wired limitation.

## 10.3 Wi-Fi Adapter USB Speed Fallback

**Symptom:** The USB Wi-Fi adapter was negotiating at 12M (USB 1.1 speed) on some ports instead of 480M (USB 2.0), severely limiting throughput.

**Resolution:** Moved the adapter to a rear I/O USB port and used a short USB extension cable to reduce RF interference. The adapter then correctly negotiated at 480M. This was verified using: sudo lsusb -t

## 10.4 Nextcloud File Scan Requirement

**Symptom:** Files copied manually into the Nextcloud data directory via Thunar file manager or SSH did not appear in the Nextcloud web interface. Only the folder name was visible, not the contents.

**Root Cause:** Nextcloud maintains an internal database index of files. It does not detect filesystem changes made outside the web UI or sync clients automatically.

**Resolution:** Running the occ files:scan command forces Nextcloud to index the filesystem and update its database. The correct command required running as the file owner (homeserver, uid 1000), not www-data:

```
sudo -u homeserver php /srv/data/nextcloud/occ files:scan homeserver
```

## 10.5 File Permission Conflicts

**Symptom:** Files created by www-data (Nextcloud web process) were not accessible by homeserver and vice versa, causing permission errors.

**Resolution:** A shared group named ‘nextcloud’ was created and both users were added to it. The setgid (2770) bit was applied to all directories so new files automatically inherit the group, ensuring both users can always read and write regardless of who created the file.

## 10.6 Ephemeral Cloudflare Tunnel URL

**Symptom:** The Cloudflare Tunnel URL changed every time the tunnel was restarted or the service was interrupted (e.g., during power cuts or system restarts).

**Root Cause:** Using cloudflared in quick-run (ephemeral) mode generates a new random URL each session. This is by design for development use, not for permanent deployments.

**Resolution:** A named, persistent tunnel was created using 'cloudflared tunnel create', bound to the purchased domain storagepaglu.pp.ua via a DNS route. This tunnel maintains the same hostname permanently regardless of service restarts or IP changes.

# 11. Final System Architecture

## 11.1 Network Topology

```
                    INTERNET
                       |
              storagepaglu.pp.ua
                       |
              CLOUDFLARE EDGE
                       |
           [Outbound Tunnel / TLS]
                       |
     +-----------------+-------------------+
     |       HOME SERVER (OurServer)       |
     |  Debian 12.6 / Intel Core 2 Duo     |
     |  4 GB DDR2 RAM / 128 GB SSD         |
     |                                     |
     |  cloudflared -> Apache:80           |
     |              -> Nextcloud           |
     |              -> MariaDB             |
     |              -> Redis               |
     |                                     |
     |  /srv/data  (MergerFS: 2 TB pool)   |
     |    |- HDD A: 1 TB (DATA1)           |
     |    |- HDD B: 500 GB (DATA2)         |
     |  /srv/backup (cold, weekly rsync)   |
     |    |- HDD C: 1 TB (BACKUP1)         |
     +-----------------+-------------------+
                       |
            Home Wi-Fi Router
                       |
         TP-Link TL-WN725N (Wi-Fi)

  Local LAN Access:  http://192.168.1.12/nextcloud
  Global Access:     https://storagepaglu.pp.ua/nextcloud
```

## 11.2 Services Running

| Service | Description | Start Mode |
|---|---|---|
| apache2 | Web server, serves Nextcloud | On boot |
| mariadb | Database for Nextcloud | On boot |
| redis-server | Memory cache for Nextcloud | On boot |
| cloudflared | Cloudflare Tunnel daemon | On boot |
| ssh (port 2222) | Remote access | On boot |
| run-backup.timer | Weekly cold backup scheduler | On boot |
| cockpit.socket | Web-based server management | Manual only |

## 11.3 Server Credentials Reference

| Parameter | Value |
|---|---|
| Domain / Hostname | OurServer |

| Parameter | Value |
|---|---|
| System User | homeserver |
| Server LAN IP | 192.168.1.12 |
| SSH Port | 2222 |
| Nextcloud Data Path | /srv/data/nextcloud |
| MergerFS Pool Mount | /srv/data |
| Backup Drive Mount | /srv/backup |

# 12. Cost Analysis

## 12.1 Itemised Bill of Materials

| Item | Description | Cost (INR) |
|---|---|---|
| PC Bundle (Facebook Marketplace) | Intel Core 2 Duo PC, UPS, monitor, keyboard, mouse, 500 GB HDD | ₹3,500 |
| 2× 1 TB Hard Drive | Used 2.5" and 3.5" HDDs for live storage and backup | ₹1200 |
| DDR2 2 GB RAM (2 sticks) | Additional RAM sticks from second-hand dealer | ₹300 |
| TP-Link TL-WN725N | USB Wi-Fi adapter (RTL8188EUS) | ₹600 |
| Domain: storagepaglu.pp.ua | 1-year registration from Regery.com | ₹120 |
| Miscellaneous | Cables, screws (included with HDD purchase) | ₹0 |
| | **Total Project Cost** | **₹6,000 (approx.)** |

**Cost vs. Google One**

Google One 2 TB storage plan costs approximately ₹650–₹800/month in India. This self-hosted server provides approximately 2 TB of live storage + 1 TB cold backup at a one-time cost of ₹6,000 — equivalent to roughly 8–10 months of Google One. From month 10 onwards, the server is essentially free.

# 13. Results and Current State

## 13.1 System Status

The server is currently live and fully operational. It runs 24/7 and is accessible globally at https://storagepaglu.pp.ua. The following is the current state of the system:

- Nextcloud is running and accessible to all family members via browser and Android app

- Automatic photo and video backup is active on family members' Android phones using the Nextcloud app
- Nextcloud Memories app provides a Google Photos-style timeline for browsing photos
- The Cloudflare Tunnel provides stable HTTPS access with no port forwarding
- Weekly automated cold backup runs every Saturday at 02:00 to the offline backup drive
- SSH access is secured on port 2222 with key-only authentication and no password login

### 13.2 Real-World Usage

My elder brother, studying at his college hostel, is able to access and back up his photos and videos directly to the home server from his phone, eliminating the need for any Google subscription. The entire family uses the Nextcloud Memories app for photo browsing and the Nextcloud Files app for document backup.

## 14. Future Improvements

- Upgrade RAM to 4 GB DDR2 (maximum supported by the motherboard) for improved Nextcloud performance
- Install Nextcloud Memories face recognition for automatic face tagging of photos (currently too slow on Core 2 Duo; best scheduled as a nightly job)
- Add SMART disk monitoring with email alerts to detect drive health issues before failure
- Consider adding a second USB Wi-Fi adapter or a PCIe Gigabit NIC (e.g., Intel PRO/1000 or Realtek RTL8111) to eliminate Wi-Fi as a single point of failure
- Implement offsite backup by periodically syncing critical data to a free-tier cloud service (e.g., Backblaze B2) for disaster recovery
- Explore OnlyOffice or Collabora Online integration for collaborative document editing within Nextcloud
- Set up a monitoring dashboard (e.g., Netdata or Glances) for real-time server performance visibility

## 15. Conclusion

This project demonstrates that a fully functional, globally accessible, multi-user cloud storage server can be built from second-hand hardware at a cost competitive with just a few months of a commercial cloud subscription. The use of Debian Linux, Nextcloud, Apache, MariaDB, MergerFS, and Cloudflare Tunnel — all free and open-source tools — enables a system that rivals Google Drive in terms of features and usability, while providing complete ownership and privacy over stored data.

The journey involved solving a range of real engineering challenges: unstable hardware, incompatible drivers, Linux filesystem permissions, Cloudflare tunnel configuration, and building a reliable backup strategy around used hard drives of uncertain history. Each problem

was methodically diagnosed and resolved, often with the assistance of AI tools used for rapid troubleshooting and documentation.

The result is a server that has solved the original problem: my brother backs up his full-HD photos from his hostel to a home server, without paying for Google storage. The total hardware and software cost was under ₹6,000, and the server will continue to pay for itself with each passing month.

# Appendix A: Key Commands Reference

## System and Monitoring

```
sudo apt update && sudo apt full-upgrade -y
htop                          # Real-time process monitor
df -h                         # Disk usage
free -h                       # RAM usage
sudo journalctl -p err -n 50  # Recent error logs
sudo systemctl list-units –failed
sudo /usr/local/bin/error-check.sh
```

## Storage Management

```
lsblk -o NAME,SIZE,TYPE,MOUNTPOINT
sudo /sbin/blkid | grep -E 'DATA1|DATA2|BACKUP1'
df -h /srv/data               # Pool usage
sudo mount -a                 # Mount all fstab entries
sudo systemctl start run-backup.service   # Manual backup
```

## Nextcloud Operations

```
# Rescan files (after manual copy)
sudo -u homeserver php /srv/data/nextcloud/occ files:scan homeserver

# Fix permissions
sudo find /srv/data/nextcloud -type d -exec chmod 2770 {} \;
sudo find /srv/data/nextcloud -type f -exec chmod 660 {} \;

# Restart services
sudo systemctl restart apache2
sudo systemctl restart mariadb
```

## Cloudflare Tunnel

```
sudo systemctl status cloudflared
sudo systemctl restart cloudflared
cloudflared tunnel list
cloudflared tunnel info homeserver-tunnel
```

## Wi-Fi Driver

```
lsmod | grep 8188eu            # Verify driver loaded
IFACE=$(iw dev | awk '/Interface/ {print $2; exit}')
sudo ethtool -i "$IFACE"       # Should show driver: 8188eu
sudo lsusb -t                  # Verify 480M speed
dkms status                    # DKMS module status
```

This document was compiled by Ayushman Sahoo, IIT Madras BS Programme (Electronic Systems), March 2026.