ABSTRACT

In an era where cloud storage is a fundamental necessity for digital data management, concerns over privacy, security, and recurring costs associated with third-party cloud services have become increasingly significant. This research proposes the development of a personal/private cloud storage system using a Raspberry Pi and Nextcloud, providing users with full control over their data while eliminating dependencies on commercial cloud providers. Unlike conventional cloud storage services, which often impose storage limitations and privacy risks, this system ensures data security, cost-effectiveness, and flexibility. The proposed solution employs a Raspberry Pi as a dedicated cloud server, integrated with an external storage device for scalability. The system leverages Nextcloud encryption techniques to safeguard stored data and incorporates a real-time notification system via the LINE app to alert users of any unauthorized access or modifications. Additionally, secure remote access is enabled through encrypted connections, ensuring that data remains protected from cyber threats. The system supports a variety of applications, including file sharing, data backup, multimedia streaming, and remote file access from multiple devices, making it a robust alternative to mainstream cloud services. The implementation process involves setting up a Raspberry Pi OS, configuring a web server (Apache), establishing a database (MariaDB), and deploying Nextcloud as the cloud storage platform. Security measures such as SSL encryption, user authentication, and access control mechanisms further enhance data protection. The proposed architecture is designed to be easily deployable, even for users with moderate technical expertise, offering an affordable, customizable, and scalable cloud storage solution. By combining low-cost hardware, open-source software, and strong security features, this research presents a viable alternative to commercial cloud storage services, particularly for individual users, small businesses, and privacy-conscious organizations.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

In today's digital landscape, cloud storage has become a necessity for individuals, businesses, and organizations to store, manage, and share data efficiently. With the increasing reliance on cloud services, concerns about data privacy, security, and cost have emerged as significant challenges. Traditional cloud storage providers such as Google Drive, Dropbox, and OneDrive operate on centralized servers, where users must entrust third-party companies with their sensitive information. This dependency raises concerns about unauthorized access, data breaches, and lack of control over personal files. Additionally, most commercial cloud storage solutions offer limited free storage, forcing users to pay expensive subscription fees as their data requirements grow. To address these issues, there is a growing interest in self-hosted, personal cloud storage solutions that provide users with full control over their data. A Raspberry Pi-based private cloud storage system offers a cost-effective, secure, and customizable alternative to mainstream cloud services. By setting up a Raspberry Pi as a mini cloud server, users can store, manage, and access their files remotely without relying on third-party providers. With the integration of Nextcloud encryption, real-time security alerts, and expandable storage, this system ensures enhanced data privacy, security, and flexibility. Furthermore, conventional Raspberry Pi-based cloud storage solutions lack a notification system to alert users of unauthorized access or file modifications, making them vulnerable to security threats. This project aims to bridge these gaps by developing a personal/private cloud storage system that incorporates encryption and real-time alerts using the LINE app. This ensures that users are promptly notified of any suspicious activity, providing an additional layer of security. By implementing this self-hosted cloud solution, users can eliminate recurring costs, mitigate security risks, and have complete ownership of their data. Whether for individuals, freelancers, small businesses, or tech enthusiasts, this project presents a reliable, scalable, and privacy-focused alternative to traditional cloud storage services.

1.1 Motivation

In today's digital world, cloud storage has become an essential tool for individuals and businesses to store, share, and manage data. However, most commercial cloud storage services come with several drawbacks, including privacy risks, limited free storage, and high subscription costs for additional space. Users often store sensitive data on third-party cloud platforms, making them vulnerable to data breaches and unauthorized access. Additionally, many cloud services link accounts to email credentials or social media logins, increasing the risk of cyberattacks. As data privacy and security concerns continue to rise, there is a strong need for a cost-effective, secure, and self-hosted cloud storage solution that allows users to have complete control over their data. This project is motivated by the need for a personalized, secure, and expandable cloud storage system that eliminates reliance on third-party providers while ensuring data protection, accessibility, and affordability.

1.2 Objectives

The primary objectives of this project are:

- To develop a personal/private cloud storage system using Raspberry Pi.
- To provide full control over data storage, eliminating dependency on external cloud providers.
- To enhance data security and privacy through Nextcloud encryption and secure authentication methods.
- To implement a real-time notification system (via the LINE app) to alert users of unauthorized access or file modifications.
- To ensure expandable storage by integrating external hard drives for seamless scalability.
- To enable remote access, allowing users to store, manage, and retrieve files from any internet-connected device.

1.3 Problem Statement

In the modern digital age, cloud storage has become an essential part of data management, offering convenience and accessibility. However, mainstream cloud storage solutions come with several limitations, including privacy risks, high costs, and security vulnerabilities. Many cloud services require users to link their accounts with email credentials or social media logins, making them susceptible to cyberattacks, unauthorized access, and data breaches. Additionally, most commercial cloud providers offer only a limited amount of free storage, forcing users to subscribe to expensive premium plans as their data needs grow. This creates a financial burden, especially for individuals, small businesses, and freelancers who require secure and scalable storage without recurring costs. Another major concern is the lack of real-time security alerts in cloud storage systems, particularly in Raspberry Pi-based personal cloud setups. If unauthorized access or modifications occur, users are often unaware until significant damage has been done. Furthermore, dependency on third-party services means users have no control over their data, making them vulnerable to potential service downtimes, policy changes, or even data loss due to external failures. Existing solutions fail to provide an integrated, cost-effective, and user-controlled approach that ensures privacy, security, and scalability. To address these issues, this project proposes the development of a personal/private cloud storage system using Raspberry Pi, offering users full control over their data, secure encryption, real-time access alerts, and an expandable storage solution without the reliance on external cloud providers. By integrating Nextcloud encryption, a notification system, and remote access capabilities, this project aims to provide an affordable and customizable alternative to mainstream cloud storage services, ensuring enhanced security, flexibility, and data ownership for users.

2.LITERATURE SURVEY

Below is a table summarizing four notable studies, followed by detailed explanations of each.

2.1 Public, Private, and Hybrid Cloud Storage Models

Authors: Wang, Ren, Lou, and Li (2010)

Methodology: Analyzed different **cloud storage models (public, private, and hybrid)** and compared their **cost, security, and scalability**. Identified security risks in public cloud, high costs in private cloud, and management complexity in hybrid cloud.

Advantages:

- Public cloud is cost-effective and easy to manage.
- Private cloud offers better security and data control.
- Hybrid cloud provides flexibility and scalability.

Disadvantages:

- Public cloud has security risks due to third-party access.
- Private cloud is expensive to set up and maintain.
- Hybrid cloud is complex to configure and manage.

2.2 Methods of Accessing Cloud Storage

Authors: Malik & Nazir (2012)

Methodology: Analyzed different cloud service models, including Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). Evaluated how users access cloud storage through these methods and how costs vary based on service usage.

Advantages:

- SaaS provides easy access to cloud applications without requiring installation.
- PaaS offers flexibility for developers to build and deploy applications.
- IaaS allows scalable infrastructure without large upfront investments.

Disadvantages:

- SaaS limits user control over the backend infrastructure.

- PaaS can be costly for small businesses due to pricing models.
- IaaS requires technical expertise to configure and manage resources effectively.

2.3 Personal Storage Using Raspberry pi

Authors: Fairuz Rauf, Maalik Ithing, and Zuraidy Adnan. Methodology: Implemented a Raspberry Pi-based cloud storage using Raspbian OS and OwnCloud. Evaluated the ease of setup, scalability, and security. Identified the need for continuous internet connectivity for better accessibility. Advantages:

- Low-cost solution for cloud storage.
- Provides user control over data privacy.
- Easy to develop and deploy.

Disadvantages:

- Requires continuous internet connectivity for remote access.
- Limited performance compared to commercial cloud solutions.

2.4. Raspberry PI -Based Cloud Storage Using ARM Architecture

Authors: Maksimović, Vujoivć, Davidović, Milošević, and Perišić (2014) Methodology: Developed a low-cost cloud storage system using ARM-based Raspberry Pi architecture. Focused on custom security enhancements and remote access.

Advantages:

- Enhanced security with user-defined encryption settings.
- Cost-effective compared to commercial cloud services.
- Customizable according to user needs.

Disadvantages:

- Requires additional cooling mechanisms to prevent overheating.
- Hardware reliability concerns, as Raspberry Pi has limited processing power.

3.PROPOSED METHODS

This project proposes a Raspberry Pi-based personal cloud storage with Nextcloud encryption, real-time security alerts, expandable storage, and remote access, ensuring cost-effective, secure, and private data management.

3.1 System Architecture

The proposed Raspberry Pi-based personal cloud storage system is designed to provide a secure, cost-effective, and customizable alternative to traditional cloud storage services. The architecture consists of a Raspberry Pi as a mini-server, an external hard drive for expandable storage, and Nextcloud software for cloud management. The system ensures data security through encryption, supports remote access via web interfaces and mobile applications, and includes real-time notifications for unauthorized access attempts.

3.2 Workflow of Raspberry Pi-Based Personal Cloud Storage

The workflow begins with users uploading, accessing, and managing files via the Nextcloud interface. The Raspberry Pi processes requests, encrypts data, and stores it on an external hard drive. When a user logs in remotely, the server authenticates the request, allowing secure access to files. In case of unauthorized access, the system triggers a real-time alert via the LINE app, notifying the user immediately.

3.3 Technologies Used

The system integrates Raspberry Pi OS as the base operating system, Nextcloud for cloud storage management, Apache, PHP, and MariaDB for web server and database functionalities, and SSL encryption for secure data transfer. Additionally, the LINE app is used for real-time notifications, ensuring users are alerted of any security breaches.

3.4 Advantages of the Proposed Method

The proposed system provides full control over data storage, eliminating reliance on third-party cloud providers. It ensures enhanced security with encryption and real-time alerts, offers scalable and expandable storage, and is cost-effective, making it an ideal solution for individuals and small businesses.

3.5 Challenges and Limitations

Despite its benefits, the system has some limitations, including initial setup complexity, hardware constraints of Raspberry Pi, and dependence on stable internet connectivity for remote access. Additionally, storage expansion requires manual configuration, and security features need continuous updates to counter evolving cyber threats.

3.6 Future Enhancements

To improve functionality, future enhancements may include AI-based intrusion detection systems, automated backup mechanisms, integration with blockchain for enhanced security, and support for multi-device synchronization. Additionally, optimizing hardware compatibility and improving data transfer speeds will enhance performance for larger-scale use.

4. DESIGN

4.1 Use Case Diagram

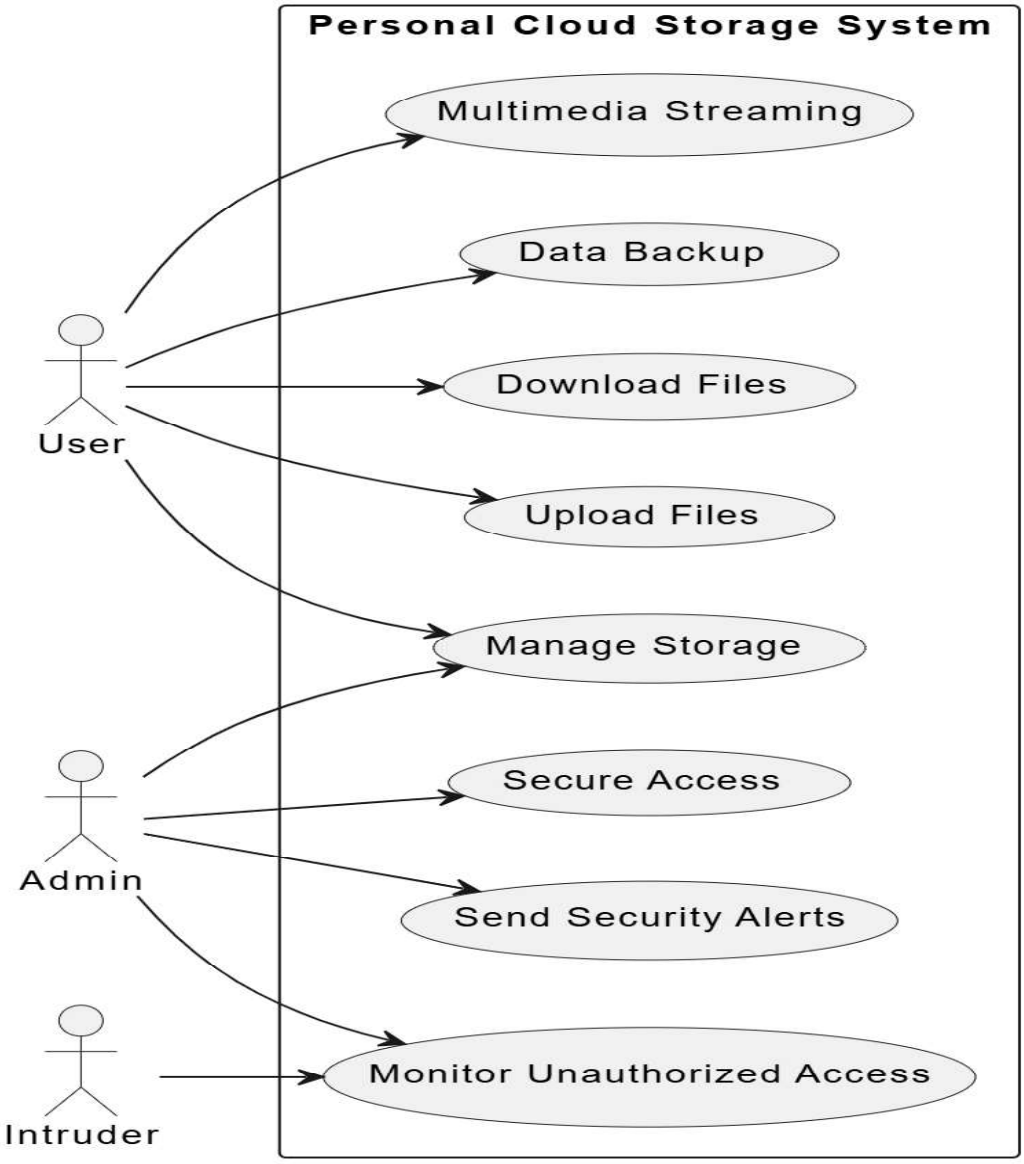


Fig 4.1

4.2 State Diagram

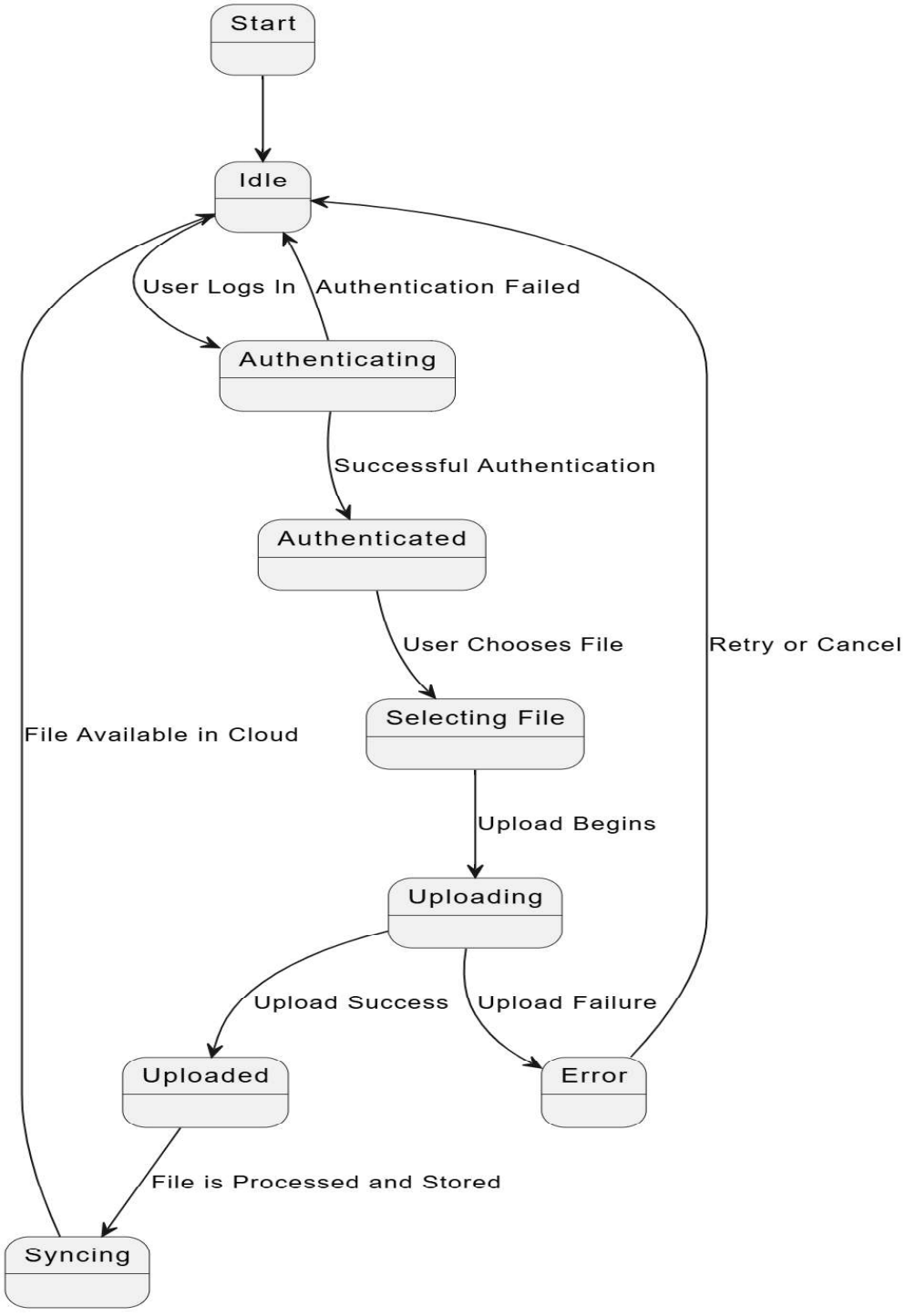


Fig 4.2

4.3 Class Diagram

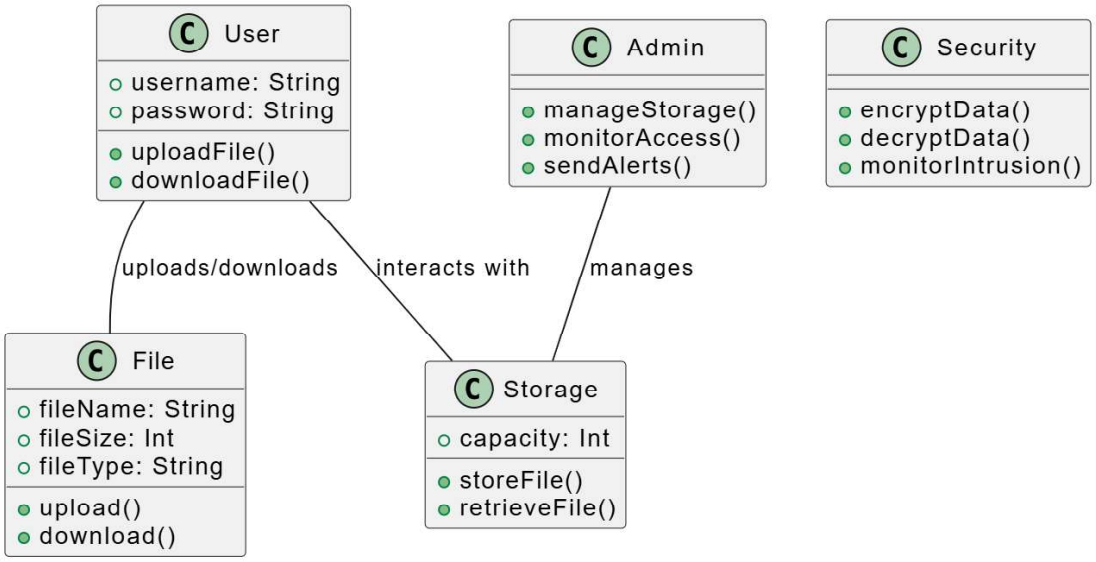


Fig 4.3

4.4 Sequence Diagram

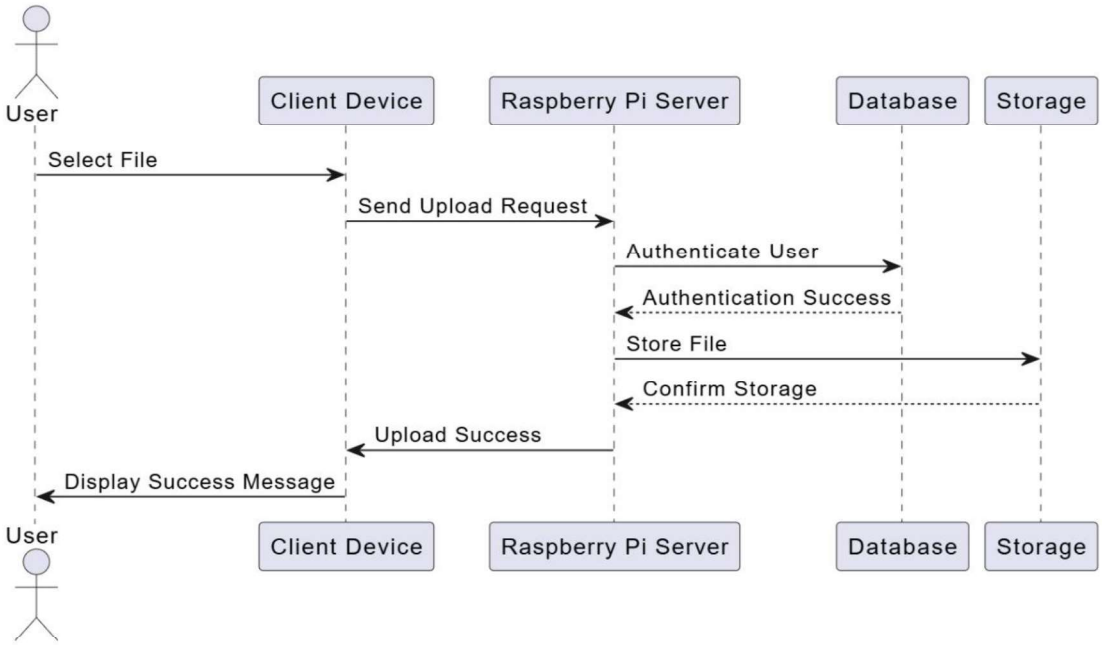


Fig 4.4

5. IMPLEMENTATION

5.1 Installing Raspberry Pi OS to your Pi

The Imager is the best tool for installing any OS for the Pi as it handles everything for you. Best of all it also supports the major three operating systems. You can follow this tutorial on Windows, macOS, Ubuntu or even your Raspberry Pi. All you need before you continue is either an SD Card and a reader, or alternatively a USB drive. Newer Pi's can boot from a USB rather than the SD Card.

The following are the steps:

Step – 1: Installing Raspberry Pi Imager

You must first download and install the Raspberry Pi Imager to your operating system. To do this, you will want to head to the official software page in your favourite web browser (https://www.raspberrypi.com/software/).

Step – 2: Once you are on the software page, scroll down until you see the heading "Install Raspberry Pi OS using Raspberry Pi Image". You should see a list of links, click the one relevant to the operating system you are currently running.

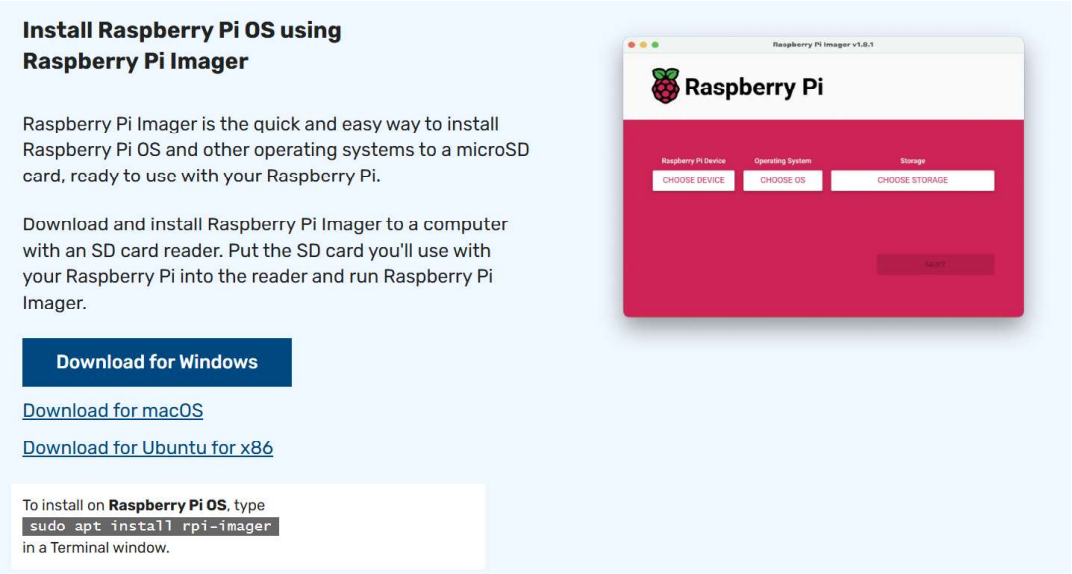


Fig-5.1

Step – 3: After downloading the installer, follow the prompts to install the software to your device. We won't be covering these steps here as they are relatively simple and are different for the three major operating systems.

5.1.1 Using Pi Imager to Install your Raspberry Pi OS

Step – 4: After installing Imager to your device, let us walk you through the process of using it to install an operating system for your Raspberry Pi.

At this point, you will want the SD Card or USB device you want to write the operating system to be plugged in to your computer.

Step – 5: Selecting your Raspberry Pi

The first thing you will want to do is select the Raspberry Pi you want to download and install the operating system. While doing this is optional it ensures the software will only offer you compatible operating systems.

To start this process, click the "CHOOSE DEVICE" button.

Fig-5.2

Step – 6:

You will now see a list of the Raspberry Pi's. Click the Pi that you want to install an operating system for.

Here, we will be selecting the Raspberry Pi 5.

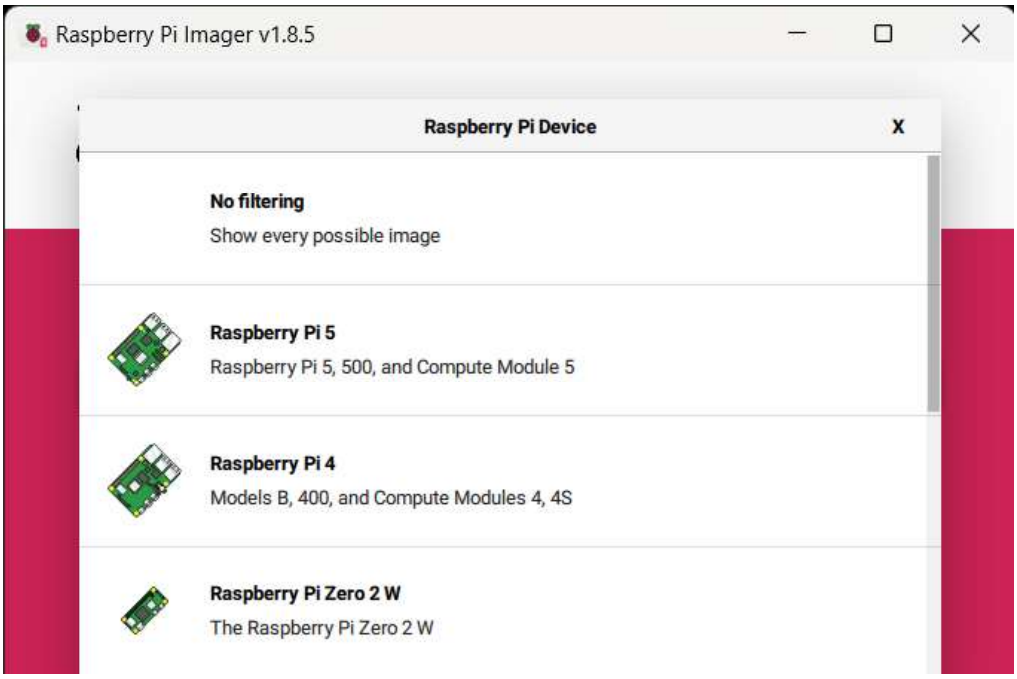


Fig-5.3

Step – 7: Selecting the OS to Install to the Raspberry Pi

After selecting the Raspberry Pi you are using, the next step is to select the OS you want to download and install.

To begin the selection process, click the "CHOOSE OS" button

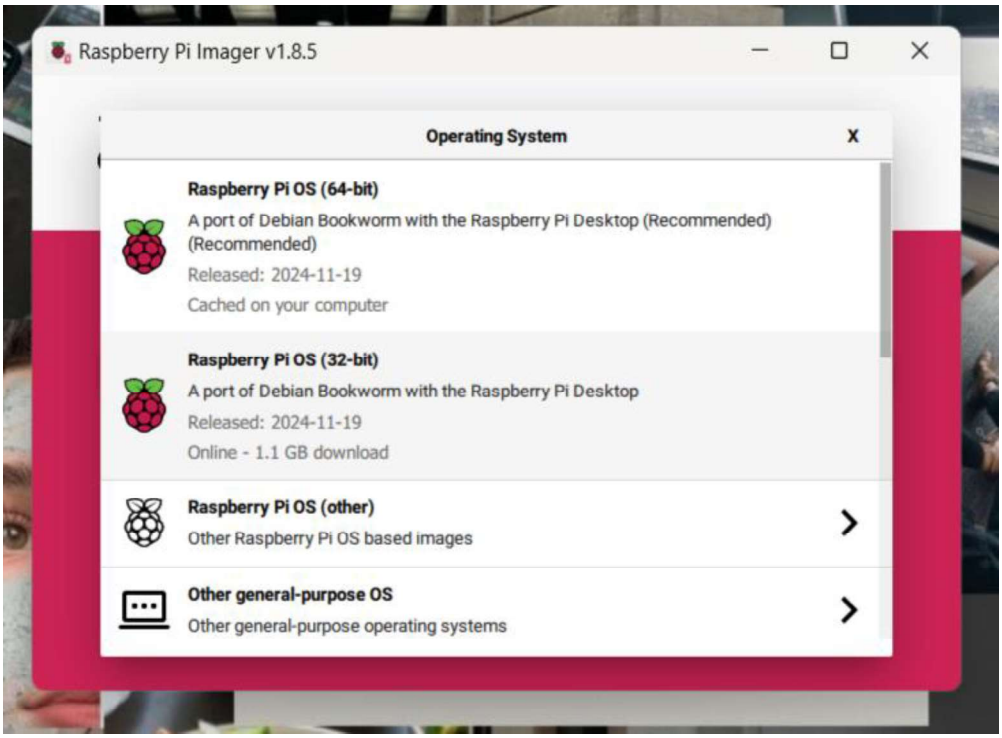


Fig-5.4

The recommended OS to install to your Raspberry Pi will be at the top of this list. For modern Pi's the OS you will likely want to install is **Raspberry Pi OS (64-bit)**. This tool offers you other operating systems you can install such as Ubuntu, OSMC and more. But for our guide we will be sticking with the official operating system.

Step – 8: Choosing the Storage Location to Install to

After selecting an operating system, you must now choose where you want this Raspberry Pi OS installed to.

To select the storage location, click the "CHOOSE STORAGE" button.

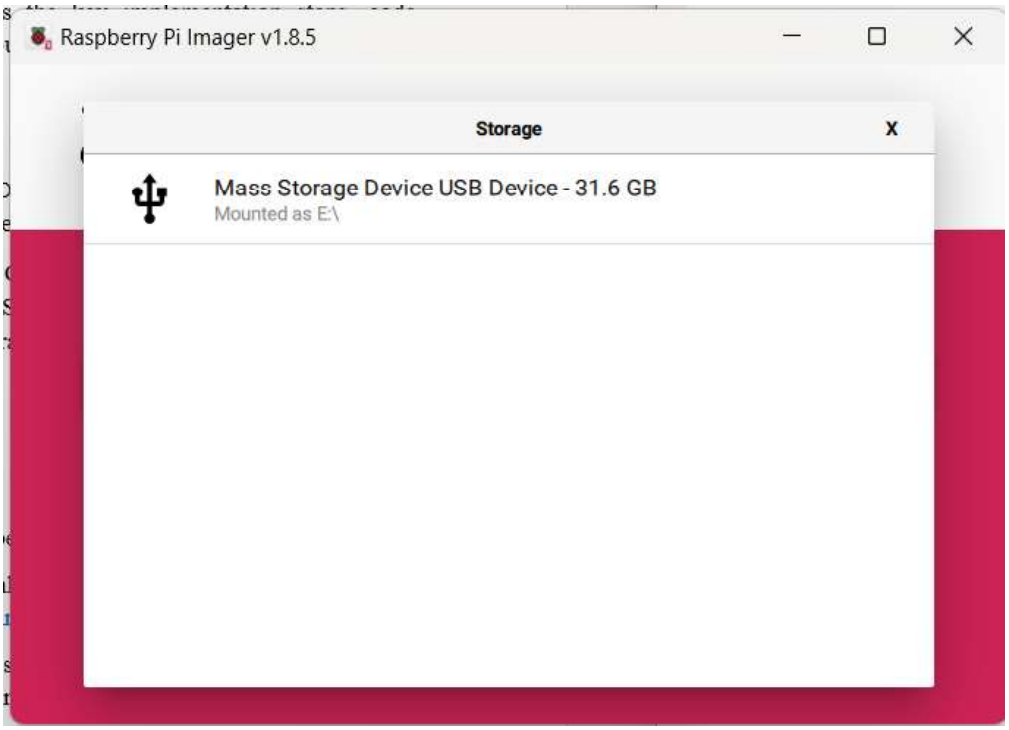


Fig-5.5

Step – 9: Start Installing Raspberry Pi OS

→You should now have all three settings configured within Raspberry Pi Imager (fig 2). If you are happy with your settings and want to install Raspberry Pi OS, click the "NEXT" button.

→If the OS you install to your Raspberry Pi supports customization you will get a message asking if you want to apply changes. To customize the OS, click the "EDIT SETTINGS" button .This will allow you to set up your user, SSH and even your Wi-Fi settings. We recommend most people choose this option as it makes the initial setup experience of Raspberry Pi OS very easy.

Step – 10: Configuring your Raspberry Pi OS Installation

The ability to customize your Raspberry Pi OS install before you run it is a key feature of the Pi Imager. The first tab you are introduced to has settings to create the user, configure wireless and set your locale settings (1.). We recommend at least setting your user as well as your Wi-Fi details.

After configuring everything on this screen, swap over to the "SERVICES" .

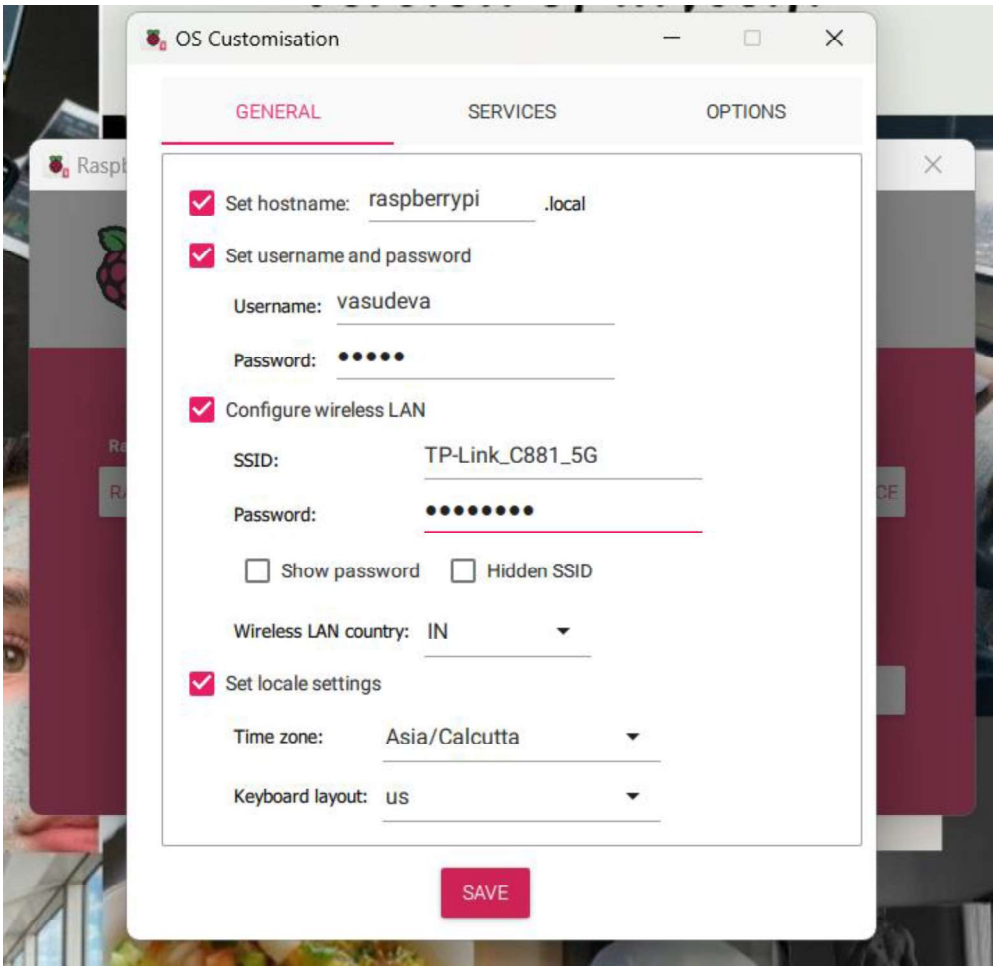


Fig-5.6

→ On this screen you can enable SSH on your Raspberry Pi (**1.**). We highly recommend using this option if you are running a headless Raspberry Pi. If you enable SSH you must set your username and password in the previous tab. After enabling SSH you can close and save by clicking the “SAVE” button (**2.**)

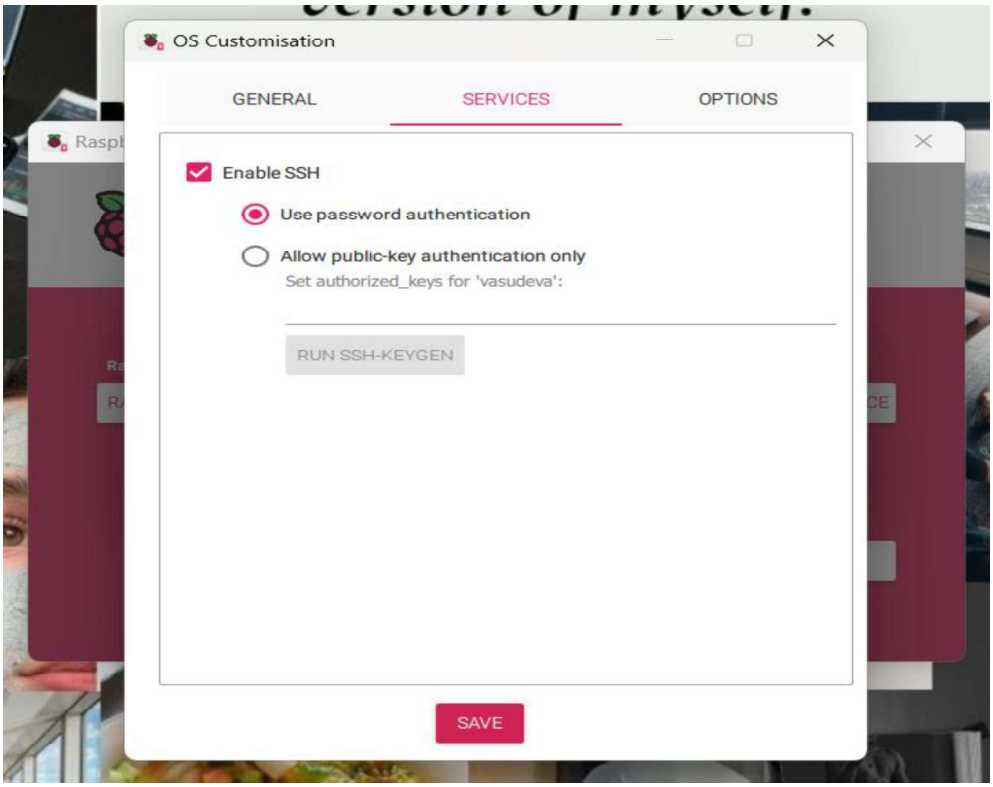


Fig-5.7

→You will again be asked if you want to apply OS customization settings.

To proceed with installing Raspberry Pi OS, click the “YES” button.

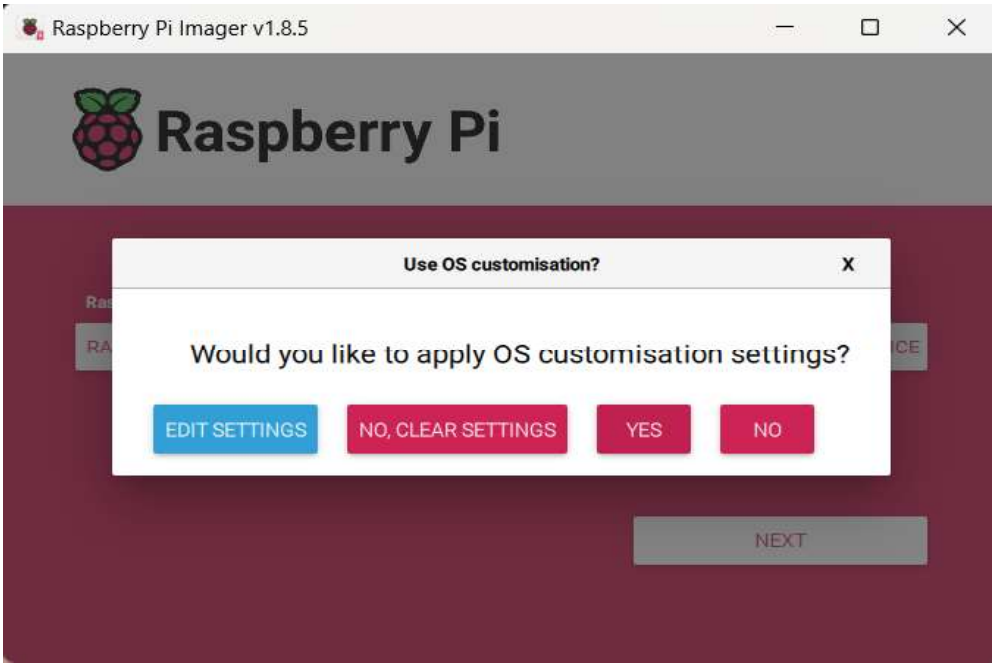


Fig-5.8

Step – 11: Writing the OS to your Storage Device

→Before the imager will install Raspberry Pi OS, you will be prompted that the data on your storage device will be erased during the writing process.

If you are fine with this, click the “YES” button.

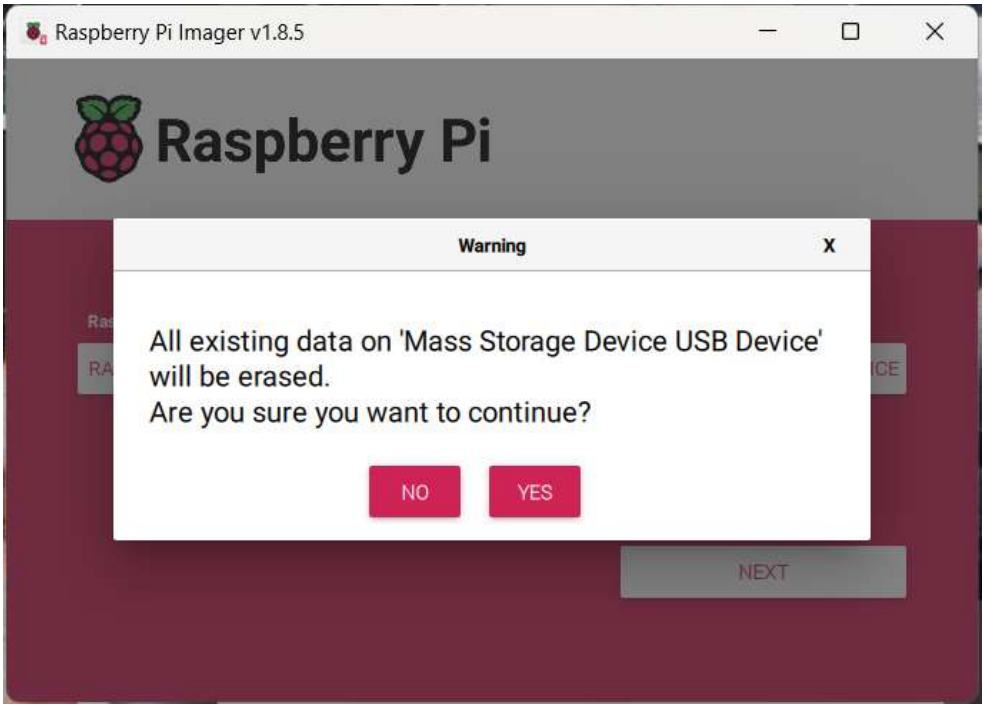


Fig-5.9

→The Raspberry Pi Imager will install your chosen OS to the SD card or USB storage device. During the writing step the tool will also be downloading the operating system itself, so it can feel like its taking longer than it should.

Fig-5.10

→ After the OS is installed to your storage device, the Raspberry Pi Imager will verify the device's contents. This is a crucial step that helps prevent you from running a corrupted operating system on your Pi.

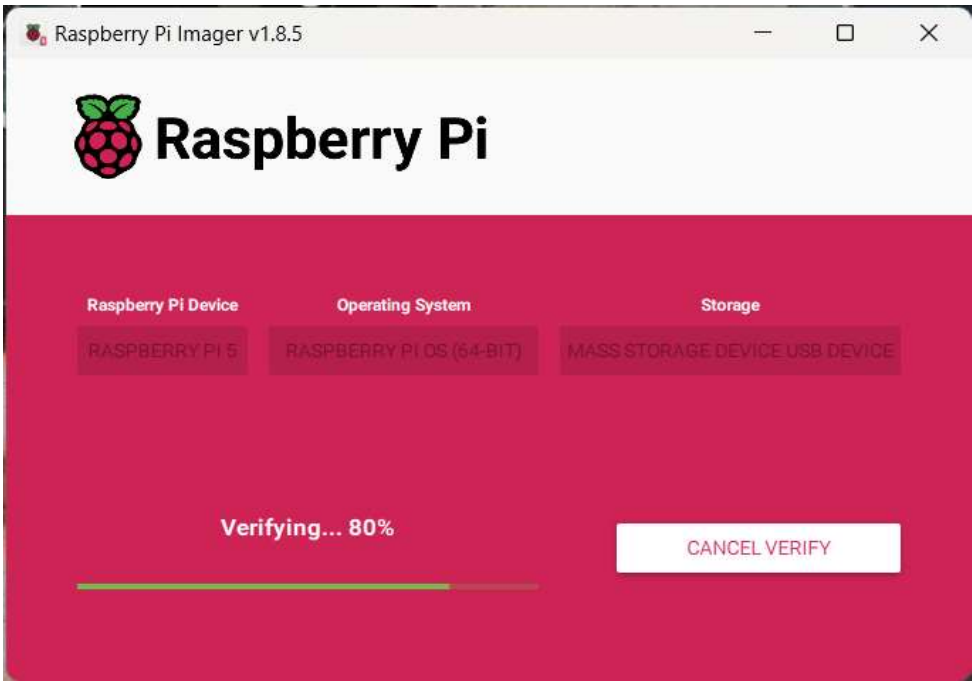


Fig-5.11

→You have successfully installed Raspberry Pi OS on a storage device. Now go and plug your SD Card or USB into your Raspberry Pi and power it on. It should detect the new operating system and launch using it.

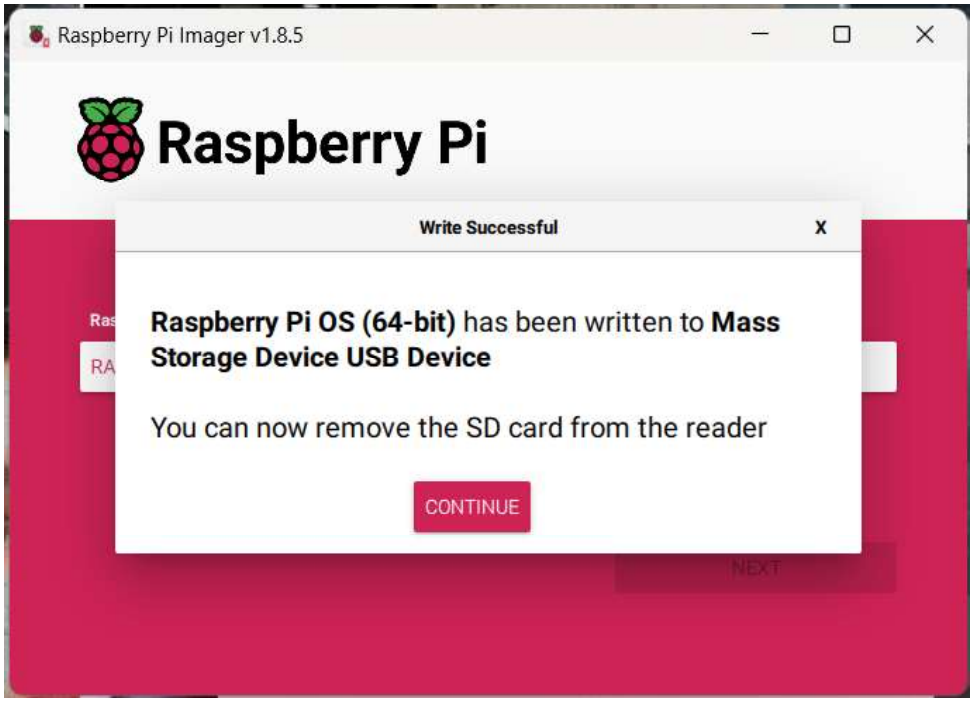


Fig-5.12

5.2 Connecting with SSH

Once Raspian is installed reboot and log in again.

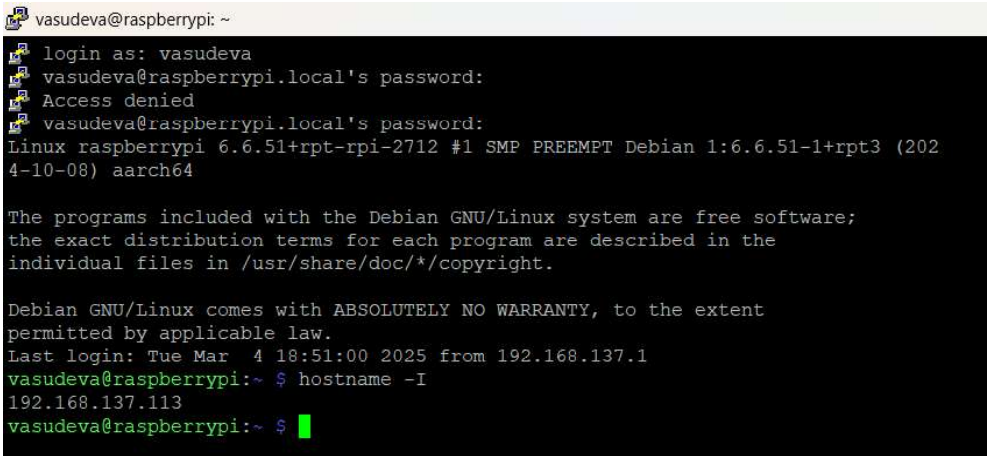


Fig-5.13

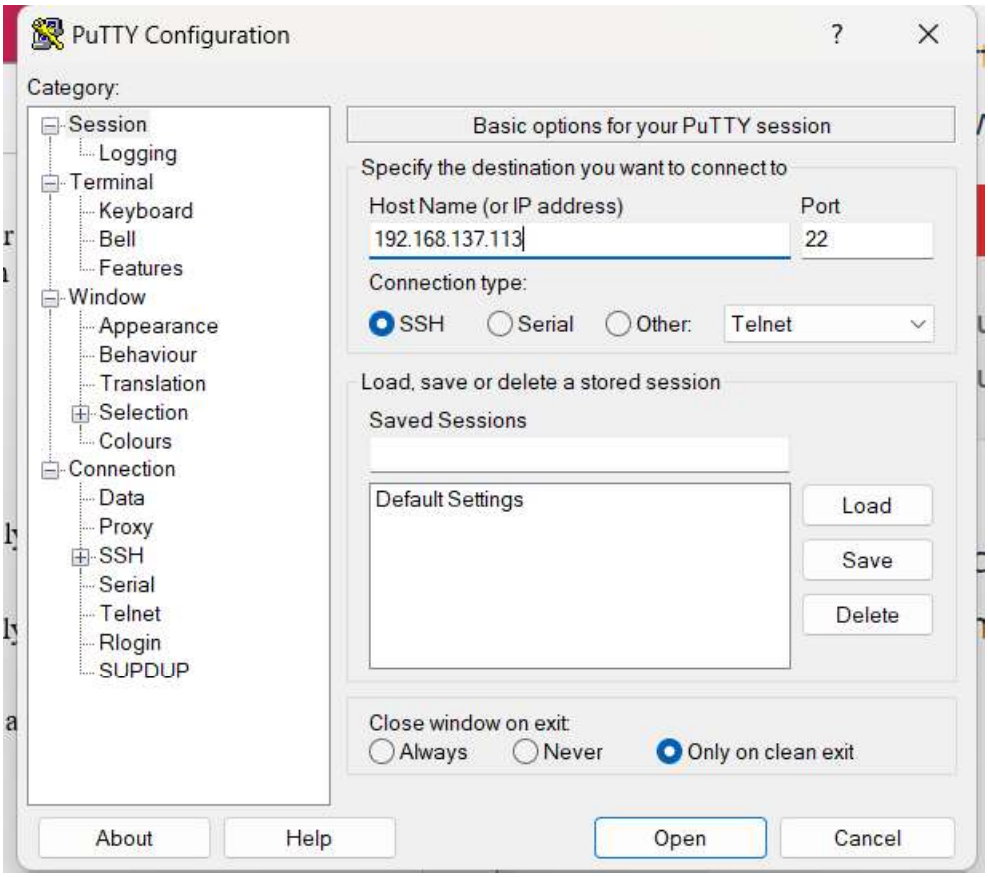


Fig-5.14

Once we are connected to SSH we will then we proceed to next steps.

5.3 Installing Apache and PHP

To run Nextcloud on the Raspberry Pi we will first need to install and setup Apache and PHP.

For this project, we will make use of the latest available version of PHP 8.2.

1.To get started let's first update our package repositories with the following command:

```
$ sudo apt update
```

```
$ sudo apt upgrade
```

2. With that done, let's now install apache with the following command:

```
$ sudo apt install apache2
```

```
vasudeva@raspberrypi: ~
vasudeva@raspberrypi:~ $ sudo apt install apache2
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap
0 upgraded, 8 newly installed, 0 to remove and 144 not upgraded.
  libaprutil1-dbd-sqlite3 libaprutil1-ldap
0 upgraded, 8 newly installed, 0 to remove and 144 not upgraded.
Need to get 2,072 kB of archives.
After this operation, 13.6 MB of additional disk space will be used.
```

3. For this tutorial, we will be using **PHP 8.2** as at the time of writing that is the recommended version for Nextlcoud.

To gain access to this version of PHP you will need to add a third-party PHP repository.

3.1. Adding a new repository within Raspberry Pi OS is a straightforward process but is a process that is required to be done from the terminal. We need to download the GPG key and add the repository as a source for the package manager. Before proceeding we need to ensure the “lsb-release” package is installed on Raspberry Pi OS. This package allows us to quickly get the details we need when setting up the PHP repository.

For our first step we need to update the package list by using the command below.

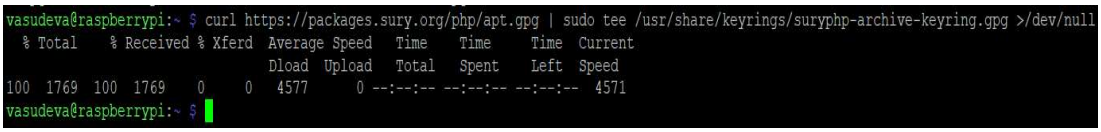
3.1.1. Once the package list has finished updating, run the following command to install the “lsb-release” package.

```
$ sudo apt install lsb-release
```

```
vasudeva@raspberrypi:~ $ sudo apt install lsb-release
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
lsb-release is already the newest version (12.0-1).
lsb-release set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 144 not upgraded.
```

3.1.2. To use any third-party repository within Raspberry Pi OS / Debian, you need to provide its GPG key. This key is used to verify that the packages you are installing do, in fact, come from that repository. We can download the GPG key for this PHP repository to our Raspberry Pi using the following command.

```
$curl
https://packages.sury.org/php/apt.gpg|sudotee/usr/share/keyrings/suryphp-
archive-keyring.gpg >/dev/null
```



3.1.3. Once the key has been saved to your Raspberry Pi, we can create a new source file that points to the repository. Use the following one-liner to create this source file with the link to the repository. Within this line, you can see that we point to the GPG key that we saved earlier. The “signed-by” text tells APT that it should verify the contents of this repository against the provided key.

Command:

```
$ echo "deb [signed-by=/usr/share/keyrings/suryphp-archive-keyring.gpg]
https://packages.sury.org/php/$(lsb_releasecs)main"|sudo
tee/etc/apt/sources.list.d/sury-php.list
```

Within this line, you can see that we point to the GPG key that we saved earlier. The “signed-by” text tells APT that it should verify the contents of this repository against the provided key.

3.1.4. Since we made changes to the APT package sources, we must perform an update of the package lists. By running an update, we are requesting new package lists from all of the sources. This will make APT aware of the packages being provided by our new PHP repository.

```
$ sudo apt update
```

3.1.5. Once the update completes, you can now download older and newer versions of PHP. In the case of Raspberry Pi OS Bullseye, this means you can gain access to no longer supported versions of PHP such as 7.3. Alternatively, if you are using Raspberry Pi OS Buster, you can access newer versions of PHP, such as PHP 8.2

3.1.6. Let us install the CLI version of PHP 8.1 to our Raspberry Pi. You can install this to the device by running the following command within the terminal.

Command:

```
$ sudo apt install php8.2-cli
```

4. With Apache2 now installed onto the Raspberry Pi, we just need to install PHP and several of its packages.

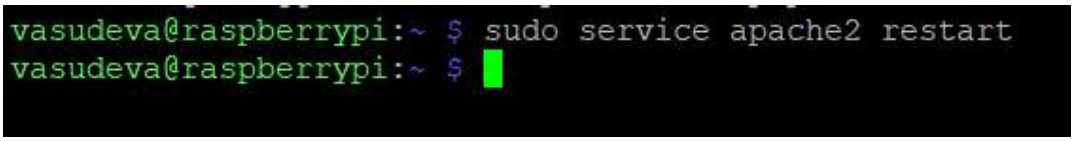
To install PHP and the packages we need, run the following command.

```
$ sudo apt install php8.2 php8.2-gd php8.2-sqlite3 php8.2-curl php8.2-zip php8.2-xml php8.2-mbstring php8.2-mysql php8.2-bz2 php8.2-intl php8.2-smbclient php8.2-imap php8.2-gmp php8.2-bcmath libapache2-mod-php8.2
```

5. With Apache and PHP now installed there is one final thing we need to do, and that is to restart Apache.

You can do this now making use of the following command:

```
$ sudo service apache2 restart
```



5.4 Setting up a MySQL Database and User for Nextcloud

we will be showing you how to set up a user and database for Nextcloud to use to store its data.

5.4.1 First we need to set up a MySQL server on your Raspberry Pi.

1. Before we get started with installing MySQL to our Raspberry Pi, we must first update our package list and all installed packages. We can do this by running the following two commands.

Commands:

```
$ sudo apt update
```

```
$ sudo apt upgrade
```

2. The next step is to install the MySQL server software to your Raspberry Pi. Installing MySQL to the Raspberry Pi is a simple process and can be done with the following command.

```
$ sudo apt install mariadb-server
```

3. With the MySQL server software installed on the Raspberry Pi, we will now need to secure it by setting a password for the "root" user. By default, MySQL is installed without any password configured, meaning you can access the MySQL server without any authentication. Run the following command to begin the MySQL securing process.

```
$ sudo mysql_secure_installation
```

Just follow the prompts to set a password for the root user and to secure your MySQL installation.

For a more secure installation, you should answer "Y" to all prompts when asked to answer "Y" or "N". These prompts will remove features that allows someone to gain access to the server easier. Make sure you write down the password you set during this process as we will need to use it to access the MySQL server and create databases and users for software such as WordPress or PHPMyAdmin.

```
vasudeva@raspberrypi: ~
vasudeva@raspberrypi:~ $ sudo mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...
```

4. Now, if you want to access your Raspberry Pi's MySQL server and start making changes to your databases, you can enter the following command.

$ sudo mysql -u root -p

```
vasudeva@raspberrypi:~ $ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 41
Server version: 10.11.6-MariaDB-0+deb12u1 Debian 12

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

5. You will be prompted to enter the password that we just created in step **3** for MySQL's root user.

6. You can now enter MYSQL commands to create, alter, and delete databases. Through this interface, you can also create or delete users and assign them the rights to manage any database.

7. There are two different ways you can quit out of the MYSQL command line. The first of those is to type "quit;" into the MySQL interface.

The other way of quitting out of the MYSQL command line is to press CTRL + D.

8. we need to do is open the MySQL command line tool by running the following command. We will be using this command line tool to create a user and database for MySQL.

$ sudo mysql -u root -p

9. Once you have logged in to the tool, we can start by creating a database. We will be creating this database called “nextclouddb” by running the following command.

```
>CREATE DATABASE nextclouddb;
```

10. Our next step is to create a user that we will be using to interact with our new database. We will be creating a user called “nextclouduser” by running the command below. Make sure that you replace “<PASSWORD>” with a secure password and make note of it for later.

```
>CREATE USER 'nextclouduser'@'localhost' IDENTIFIED BY '<PASSWORD>';
```

Here user with the password “panda”, the command would look like this

```
>CREATE USER 'nextclouduser'@'localhost' IDENTIFIED BY 'panda';
```

11. now we need to now give it permissions to interact with our database.

We can do that by running the following command.

```
> GRANT ALL PRIVILEGES ON nextclouddb.* TO 'nextclouduser'@'localhost';
```

This command grants the user “nextclouduser ” all privileges on the “nextclouddb” database and all of its tables.

12. Our final task is to flush the privilege table.

To flush the privileges all we need to do is run the following command.

```
>FLUSH PRIVILEGES;
```

```
vasudeva@raspberrypi:~ $ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 41
Server version: 10.11.6-MariaDB-0+deb12u1 Debian 12

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE nextclouddb;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> CREATE USER 'nextclouduser'@'localhost' IDENTIFIED BY 'nextcloud';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON nextclouddb.* TO 'nextclouduser'@'localhost';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]>
```

we can now proceed to install Nextcloud on our Raspberry Pi

5.5 Downloading Nextcloud on your Raspberry Pi

Getting Nextcloud on your the Raspberry Pi is quite simple, it mainly involves downloading the zip file from their website, extracting it and then making some .

1. To get started let's first move to our html directory with the following change directory command.

$ cd /var/www/

```
vasudeva@raspberrypi:~ $ cd /var/www/
vasudeva@raspberrypi:/var/www $ sudo w
```

2. Now we can download the latest version of Nextcloud to our device. To do this we will use wget to download the latest release to the current folder.

$ sudo wget https://download.nextcloud.com/server/releases/latest.tar.bz2

3. With Nextcloud now downloaded to our Raspberry Pi, let us extract the archive. To extract the archive using "tar" we need to use the command below.

$ sudo tar -xvf latest.tar.bz2

4. We now need to create a data directory for Nextcloud to operate in, for the initial setup of Nextcloud we must make this folder in our "/var/www/" directory. Create the directory by using the mkdir command as shown below:

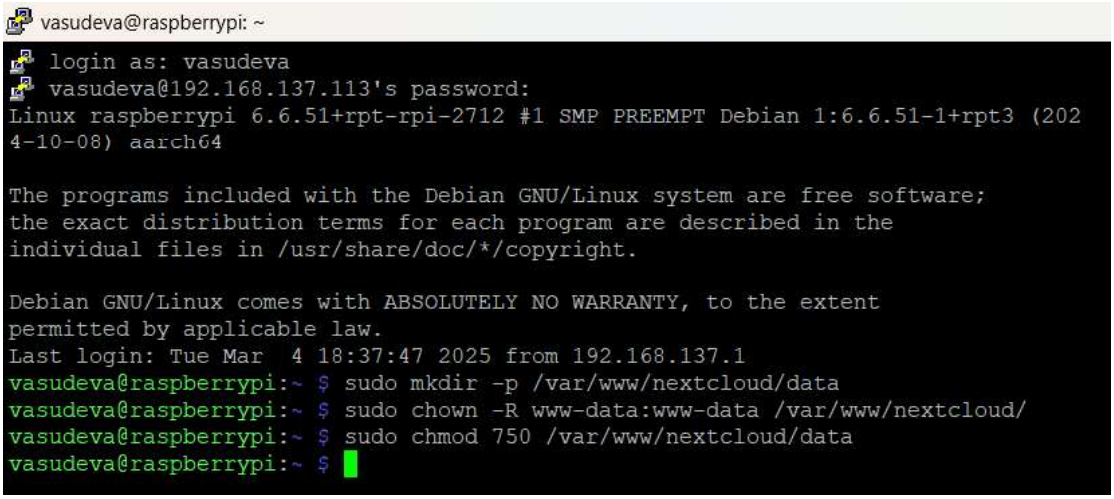
$ sudo mkdir -p /var/www/nextcloud/data

5. Now let's give the correct user and group control over the entire Nextcloud folder and everything inside it by running the following chown command.

$ sudo chown -R www-data:www-data /var/www/nextcloud/

6. Finally we need to give it the right permissions, again run the following chmod command.

$ sudo chmod 750 /var/www/nextcloud/data



5.6 Configuring Apache for Nextcloud

Next, we need to deal with the ".htaccess" file for Nextcloud. Since we installed Nextcloud into the default Apache2 directory "/var/www/", we will need to change some settings in Apache2 to allow the ".htaccess" file to override settings. To handle this just for the Nextcloud directory we will be creating a new configuration file for Apache.

1. To get started let's create a file which will store our configuration changes for Nextcloud. You can begin to write this config file by using the following command in the terminal. Here we are using Nano as it is one of the easiest to use.

```
$ sudo nano /etc/apache2/sites-available/nextcloud.conf
```

```
vasudeva@raspberrypi:~ $ sudo nano /etc/apache2/sites-available/nextcloud.conf
vasudeva@raspberrypi:~ $ sudo a2ensite nextcloud.conf
```

2. Now before you proceed any further you will need to decide whether you want Nextcloud to run under the "/nextcloud" directory or on its own domain or subdomain. If you want NextCloud to be accessible whenever someone goes to "/nextcloud" then all you need to do is type in the following lines into the file. This configuration is simple and means whenever someone goes to your Pi's IP address, followed by "/nextcloud" they will be greeted with its interface.

3.These lines basically tell Apache2 how to handle itself within the "/var/www/nextcloud/" folder. These changes will allow Apache2 to read and utilize the ".htaccess" files within the Nextcloud directory.

4. Now we can save and quit out of the file by pressing CTRL + X then pressing Y and then ENTER.

5. With the file created we now need to tell Apache to make use of it. We can do this by utilizing the a2ensite command followed by "nextcloud.conf".

```
$ sudo a2ensite nextcloud.conf
```

```
vasudeva@raspberrypi:~ $ sudo a2ensite nextcloud.conf
Enabling site nextcloud.
To activate the new configuration, you need to run:
  systemctl reload apache2
```

6. Now we need to restart Apache2 to force it to read in the updated configuration file. We can do that easily with the following command:

```
$ sudo systemctl reload apache2
```

```
vasudeva@raspberrypi:~ $ sudo systemctl reload apache2
vasudeva@raspberrypi:~ $
```

5.7 Nextcloud Initial Setup

1. Now that we have finished with that, we can now finally go to Nextcloud itself and begin its installation process.

To begin go to your Raspberry Pi's IP address followed by the path "/nextcloud".

http://192.168.137.113/nextcloud

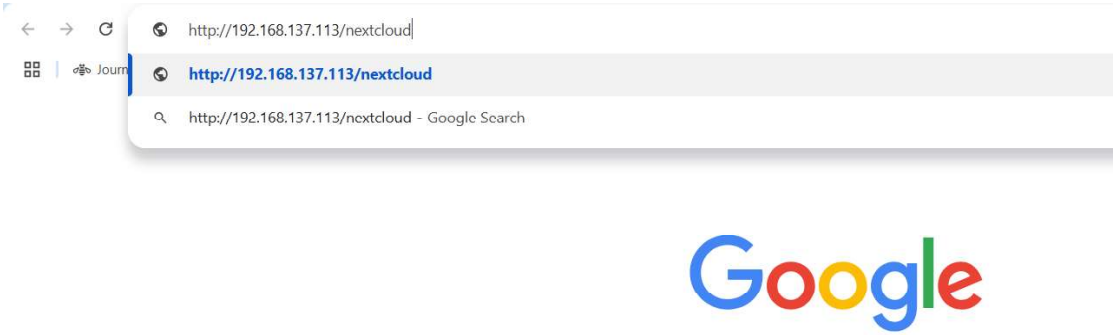


Fig-5.15

2. You will now be greeted with the following screen.

Here you will need to type in the **Username** and **Password** that you intend to use for your admin account.

If you plan on allowing your Nextcloud file service to be accessible from outside your network, make sure that you use a long and secure password.

Next, we need to specify the details for our database server. To get to these options you will need to click the "Storage & Datbase".

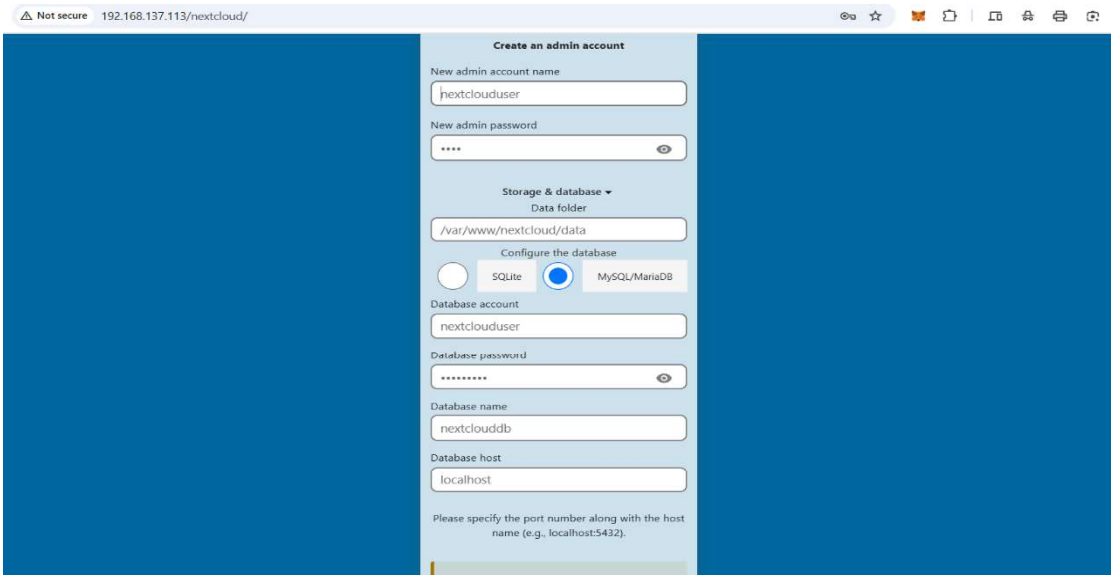


Fig-5.16

Now you need to select the type of database we want to use. As we are using an SQL server click the “MySQL/MariaDB” option.

Finally we need to enter the details for our database server. There are three bits of information that we will need to enter.

1. The username for the user that will interact with our database server. If you are using the same information we used, this setting should be set to nextclouduser.
2. The password that you set for the above user.
3. The final option you will need to set is the database name. If you have been following our guide this will be nextclouddb.

After filling this we press the “**Finish Setup**” button.

3. It goes to log in page now log in with your admin usesrname and password.

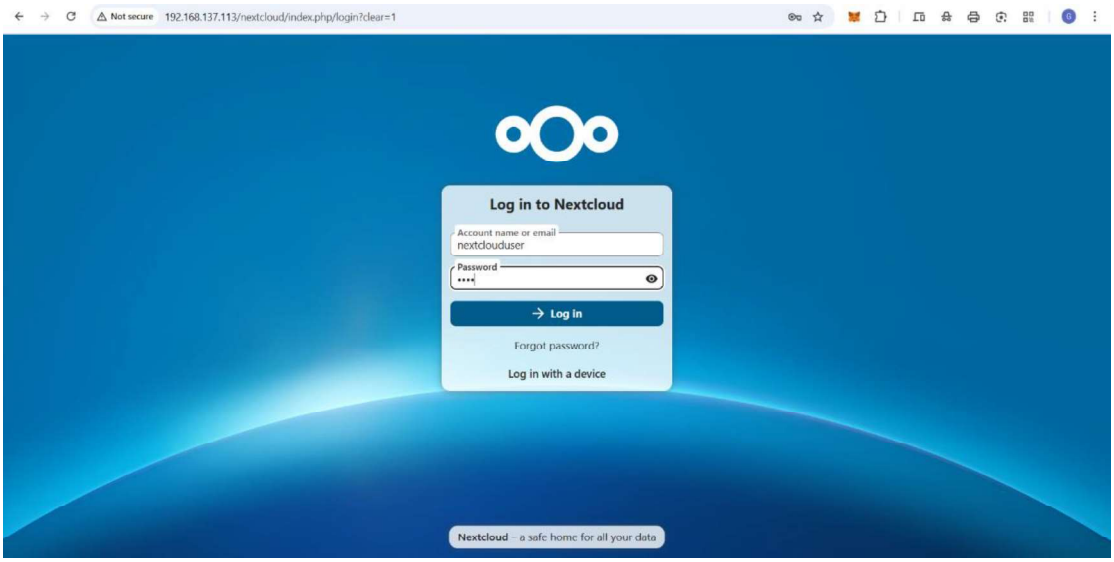


Fig-5.17

4. After this you should now be greeted with the following welcome screen, this just lays out the various programs you can use to connect with your Nextcloud installation. Just click the **X** button in the top right corner to continue. This is how the dashboard look like of nextCloud interface.

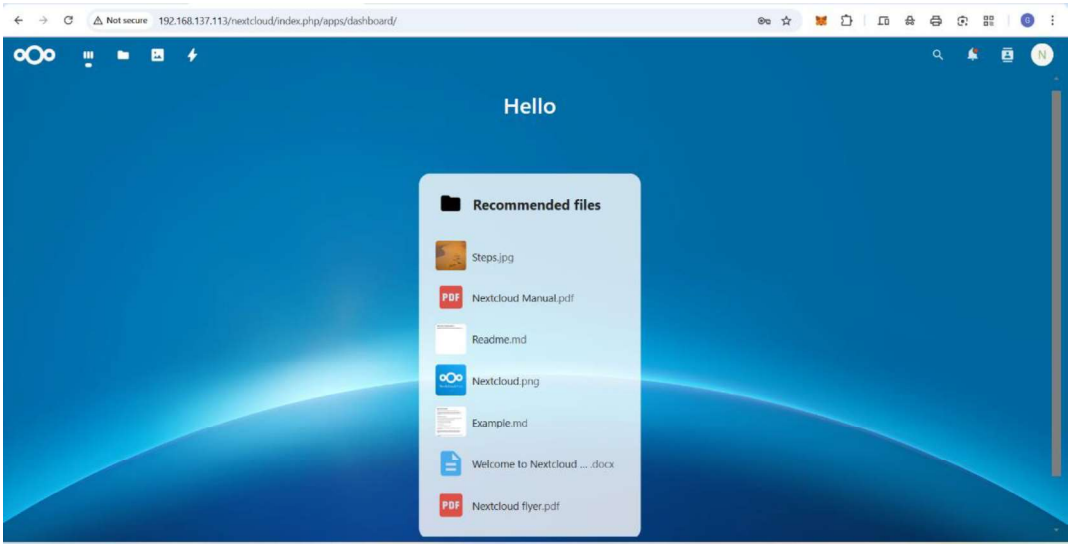


Fig5.18

5. Now you can finally see the interface of the Raspberry Pi Nextcloud, you should take some time to familiarize yourself with all the functionality of Nextcloud's interface.

We won't go too in depth on how to use the Nextcloud interface, if you need more information then I recommend checking out the support section on nextcloud. We have however highlighted some of the key areas to check out in the screenshot below.

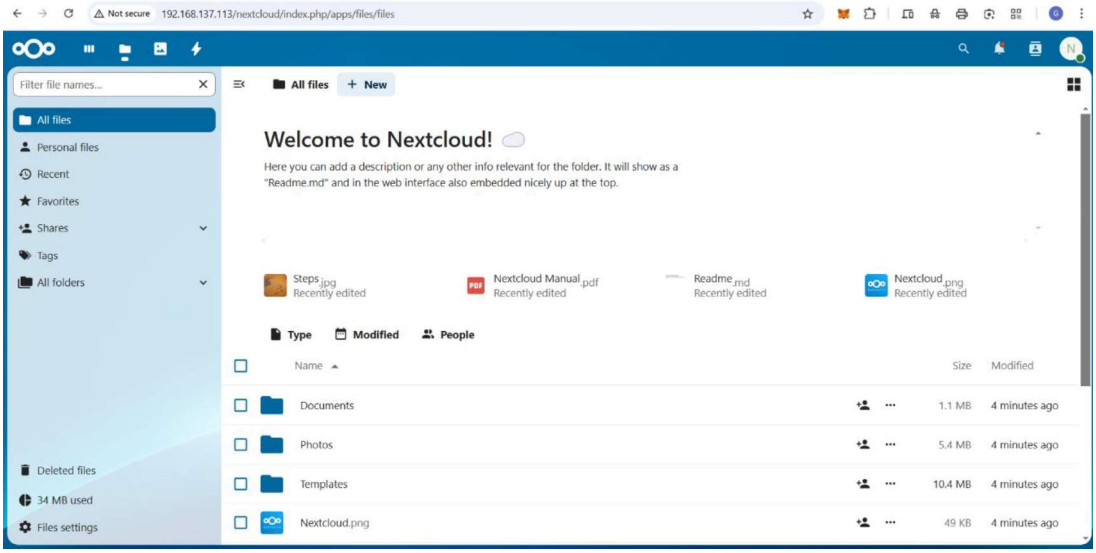


Fig-5.19

5.8 Moving Nextcloud's Data Folder

With Nextcloud now safely installed we can now tweak the setup to both be more secure and a bit more useable. One of the first things we should do is move the data directory so it does not sit in our web accessible directory.This is also the same way you would move your Nextcloud data directory onto a larger external hard drive rather than putting increased load onto the Raspberry Pi's SD Card.

1. To get started let's make our new directory for where we will store our data files. To make it easy we will make a new folder at "/var/nextcloud" and move our data folder into there. Create the folder by running the following command:

```
$ sudo mkdir -p /var/nextcloud
```

```
vasudeva@raspberrypi:~ $ sudo mkdir -p /var/nextcloud
vasudeva@raspberrypi:~ $
```

2. With our new folder we created we will now move our data directory into it, this is easy to do thanks to the mv command. Please note that your Nextcloud system will be out of action while we move the file then adjust the configuration file. To begin the move type in the following command:

```
$ sudo mv -v /var/www/nextcloud/data /var/nextcloud/data
```

```
vasudeva@raspberrypi:~ $ sudo mv -v /var/www/nextcloud/data /var/nextcloud/data
renamed '/var/www/nextcloud/data' -> '/var/nextcloud/data'
vasudeva@raspberrypi:~ $
```

3. Now with the files moved over we can now modify the "datadirectory" configuration to point to our new directory. First, let's change to the config directory for Nextcloud with the following command.

```
$ cd /var/www/nextcloud/config
```

```
vasudeva@raspberrypi:~ $ cd /var/www/nextcloud/config
vasudeva@raspberrypi:/var/www/nextcloud/config $ sudo
```

```
vasudeva@raspberrypi:/var/www/nextcloud/config $
```

4.We can now copy the config file to make a backup of the file, we can do this with the following command:

$ sudo cp -p config.php config.php.bk

```
vasudeva@raspberrypi:/var/www/nextcloud/config $ sudo cp -p config.php config.php.bk
vasudeva@raspberrypi:/var/www/nextcloud/config $
```

5.Finally let's open up the "config.php" file for editing using nano.

$ sudo nano config.php

```
vasudeva@raspberrypi:/var/www/nextcloud/config $ sudo nano config.php
vasudeva@raspberrypi:/var/www/nextcloud/config $ sudo chown -R www-dat
```

6. Within this file we need to change the following line:

Find

'datadirectory' => '/var/www/nextcloud/data',

Replace

'datadirectory' => '/var/nextcloud/data',

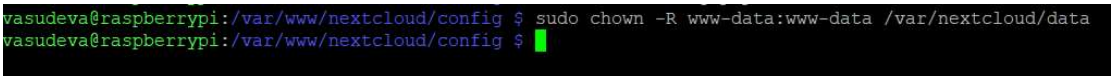
vasudeva@raspberrypi: /var/www/nextcloud/config

```
  GNU nano 7.2
<?php
$CONFIG = array (
  'instanceid' => 'oc2m39obdkdc',
  'passwordsalt' => 'bDU33rWgQaK6PzA0iNNcc1brMGMbZY',
  'secret' => 'rk4OuZvZcg5MPTfyPF8ywAHMQQce6ogRUWS4QG6BRsirc/Ws',
  'trusted_domains' =>
  array (
    0 => '192.168.137.113',
  ),
  'datadirectory' => '/var/nextcloud/data',
  'dbtype' => 'mysql',
  'version' => '31.0.0.18',
  'overwrite.cli.url' => 'http://192.168.137.113/nextcloud',
  'dbname' => 'nextclouddb',
  'dbhost' => 'localhost',
  'dbport' => '',
  'dbtableprefix' => 'oc_',
  'mysql.utf8mb4' => true,
  'dbuser' => 'nextclouduser',
  'dbpassword' => 'nextcloud',
  'installed' => true,
);
```

7. Now we can save and quit out of the file by pressing CTRL + X then Y and then ENTER.

8. As one last precuation we should make sure that the “www-data” user still has ownerships over our new folder.

```
$ sudo chown -R www-data:www-data /var/nextcloud/data
```



You should be able to now refresh your web browser and all your files should be showing exactly as they were previously.

5.9 Increasing Nextcloud’s Max Upload Size

By default, PHP has a very low upload limit, so low it’s only 2 MB. To change this, we need to modify the “php.ini” file and increase the limit. A cloud storage system wouldn’t be very useful if you could only ever upload 2mb files.

1. To get started we need to begin editing the configuration file with the following command:

```
$ sudo nano /etc/php/8.2/apache2/php.ini
```

2. Now we need to find and replace the following two lines.

Find :

post_max_size = 8M

upload_max_filesize = 2M

Replace :

post_max_size = 1024M

upload_max_filesize = 1024M

Of course, you can set the file size limits to something that is much higher than 20M, so feel free to change that number to whatever you think is the maximum size file you will upload to your Nextcloud.

3. Now we can save and quit out of the file by pressing CTRL + X then pressing Y and then ENTER. Now we need to restart Apache2 to force it to read in the updated configuration file. We can do that easily with the following command:

```
$ sudo service apache2 restart
```

```
vasudeva@raspberrypi:/var/www/nextcloud/config $ sudo service apache2 restart
vasudeva@raspberrypi:/var/www/nextcloud/config $
```

4. You should now be able to restart your web browser and begin a new upload to see that the maximum upload size has been increased successfully.

5.10 Setting up SSL for Nextcloud

Now we should really work on setting up your Raspberry Pi Nextcloud server so that it runs through HTTPS and not plain HTTP. For this tutorial, we will assume that you do not have a domain name, so we will be generating our own self signed certificate and not utilizing one from a free service such as Letsencrypt.

1. Before we go modifying our Apache2 configuration we will first generate the self-signed certificate, luckily, we can do this all in one command thanks to OpenSSL. Remember that a **self-signed certificate will throw errors** in your web browser and is not as secure as a properly signed certificate but it is better than nothing. It is also the only option if you're not utilizing a domain name. Before we generate the certificate, let's first make a directory to store it.

```
$ sudo mkdir -p /etc/apache2/ssl
```

```
vasudeva@raspberrypi: /var/www/nextcloud/config
vasudeva@raspberrypi:/var/www/nextcloud/config $ sudo mkdir -p /etc/apache2/ssl
vasudeva@raspberrypi:/var/www/nextcloud/config $
```

2. Now let's generate the certificate itself by running the following command in the terminal:

```
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:4096 -keyout
/etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt
```

If you want to know exactly what these command arguments do, then read our little description below.

req: This specifies a subcommand for X.509 certificate signing request (CSR) management.

-x509: This option specifies that we want to make a self-signed certificate file instead of generating a certificate request.

-nodes: This tells the OpenSSL application that we don't want to specify a passphrase, a passphrase will require us to enter it every time Apache is restarted which is painful to deal with.

-days 365: This specifies the amount of days we want the certificate to remain valid for, after this amount of days you will have to generate a new certificate.

-newkey rsa:4096: This will create the certificate request and a new private key at the same time. You will need to do this since we didn't create a private key in advance. The rsa:4096 tells OpenSSL to generate an RSA key that is 2048 bits long.

-keyout: This parameter names the output file for the private key file that is being created.

-out: This option names the output file for the certificate that we are generating.

After pressing enter you will be presented with the following options to fill out.

Country Name (2 letter code) [AU]:

State or Province Name (full name) [Some-State]:

Locality Name (eg, city) []:

Organization Name (eg, company) [Internet Widgits Pty Ltd]:

Organizational Unit Name (eg, section) []:

Common Name (e.g. server FQDN or YOUR name) []:

Email Address []:

3. Once you have filled out all that information we can then proceed on with setting up Apache2 to run SSL and to also utilize our newly generated certificate. This is a simple process but an important one. First let's enable the SSL module for Apache with the following command:

```
$ sudo a2enmod ssl
```

```
vasudeva@raspberrypi:/var/www/nextcloud/config $ sudo a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Enabling module socache_shmcb.
Enabling module ssl.
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-signed certificates.
To activate the new configuration, you need to run:
  systemctl restart apache2
vasudeva@raspberrypi:/var/www/nextcloud/config $
```

4. Now we need to modify the "default-ssl.conf" file so it will utilize our new certificates and not the default ones that are generated by OpenSSL on installation. To begin modifying this file run the following command:

```
$ sudo nano /etc/apache2/sites-available/default-ssl.conf
```

```
To activate the new configuration, you need to run:
  systemctl restart apache2
vasudeva@raspberrypi:/var/www/nextcloud/config $ sudo nano /etc/apache2/sites-available/default-ssl.conf
```

5. Within this file we need to change the two lines below to point to our new certificates we generated into our "/etc/apache2/ssl" folder.

Find:

SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem

SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key

Replace with:

SSLCertificateFile /etc/apache2/ssl/apache.crt

SSLCertificateKeyFile /etc/apache2/ssl/apache.key

6. Now we can save and quit out of the file by pressing CTRL + X then pressing Y and then Enter.

7. We can now enable the default-ssl configuration and restart Apache to load in our new configuration. We can do this with the following two commands.

$ sudo a2ensite default-ssl.conf

```
vasudeva@raspberrypi:/var/www/nextcloud/config $ sudo a2ensite default-ssl.conf
Enabling site default-ssl.
To activate the new configuration, you need to run:
  systemctl reload apache2
vasudeva@raspberrypi:/var/www/nextcloud/config $
```

$ sudo service apache2 restart

```
vasudeva@raspberrypi:/var/www/nextcloud/config $ sudo service apache2 restart
```

8. You can test to make sure this is working by going to your Raspberry Pi's IP address with "https://" in front of it. It will give you a warning about it potentially being an invalid certificate. This is normal as it is an unsigned certificate. For instance to make sure my own copy of Nextcloud is now running behind SSL I would go to the following.

https://192.168.137.113/nextcloud

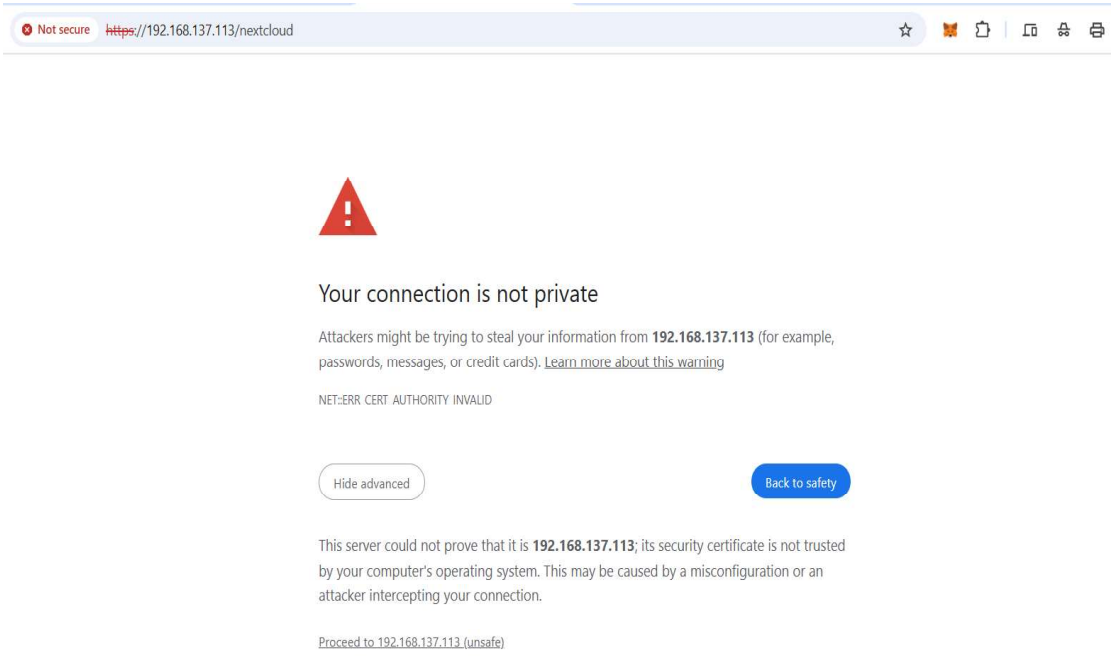


Fig-5.20

9. An optional extra step to ensure that you have the best security for your Nextcloud setup is to enforce SSL so no connection can be made over HTTP, if a connection is made it will redirect you to HTTPS.

We can do this by making some changes to our apache configuration, to begin let's edit the default file with the following command:

```
$ sudo nano /etc/apache2/sites-available/000-default.conf
```

```
vasudeva@raspberrypi:/var/www/nextcloud/config $ sudo nano /etc/apache2/sites-available/000-default.conf
```

10. Replace all the text in this file with the code below. This will basically redirect all HTTP traffic to its HTTPs equivalent. Ensure that you replace "21eg105e21@anurag,edu,in" with an email address you can be contacted on. This will be displayed whenever an error occurs.

11. Now we can save and quit out of the file by pressing CTRL + X then pressing Y and then ENTER.

12. Now before this will work we need to enable the redirect module and restart apache. We can easily achieve this by running the following two commands:

```
$ sudo a2enmod rewrite
```

```
$ sudo service apache2 restart
```

Now going to your Raspberry Pi on HTTP should automatically redirect to the HTTPS version. For example, if I go to "http://192.168.137.113" it will redirect to "https://192.168.137.113".

5.11 Port Forwarding Nextcloud

Since most home public IP addresses are dynamic you will need to look into setting up a dynamic DNS service, you will find our tutorial on how to setup a dynamic DNS service for your Raspberry Pi very handy.

1. To add your domain/IP we need to modify NextCloud's configuration file, we can do that by running the following command:

```
$ sudo nano /var/www/nextcloud/config/config.php
```

2. Within this file you will see a block of text like below. This is an array of all trusted domains that you allow Nextcloud to operate through. For now, it should only include your Raspberry Pi's local IP address. We will add our new domain/IP onto the end of this array. We will be adding "vasudevapiproject.com" to the array. This means we need to increment the array ID and add the domain name. Once you have added a new one it should look something like below. Repeat this procedure for any new IP's or domains you want Nextcloud to be able to operate through.

Find:

```
'trusted_domains' =>
array (
    0 => '192.168.1.105',
),
```

Replace with:

```
'trusted_domains' =>
array (
    0 => '192.168.1.105',
    1 => 'vasudevapiproject.com',
),
```

3. Now we can save and quit out of the file by pressing CTRL + X then pressing Y and then ENTER.

4. Finally, you will need to port forward two ports to finally have Nextcloud up and running. These two ports being Port 80 and Port 443. The protocol required for these is TCP.

6. EXPERIMENTAL RESULTS

6.1 Searching the IP address in Internet

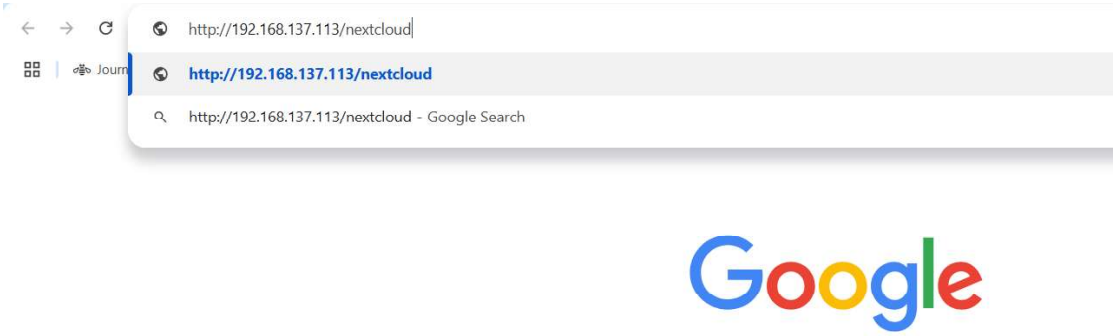


Fig-6.1

Here we enter the ip address to connect to the host.

6.2 Create Admin and Connect to database we created previously

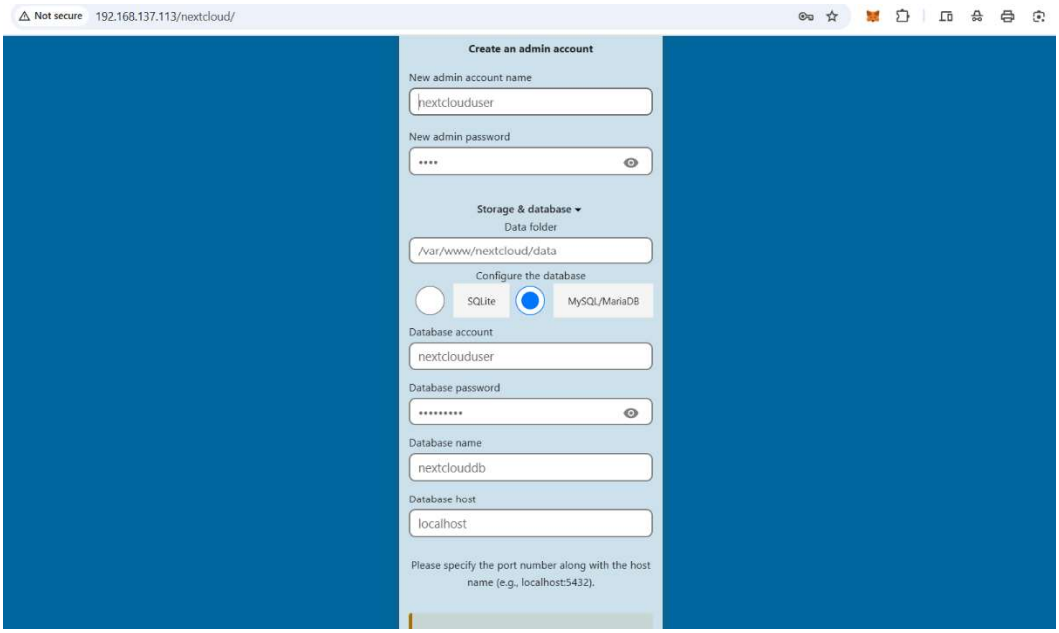


Fig-6.2

In this step we create the admin credential which will be used to connect to database.

6.3 Login Page

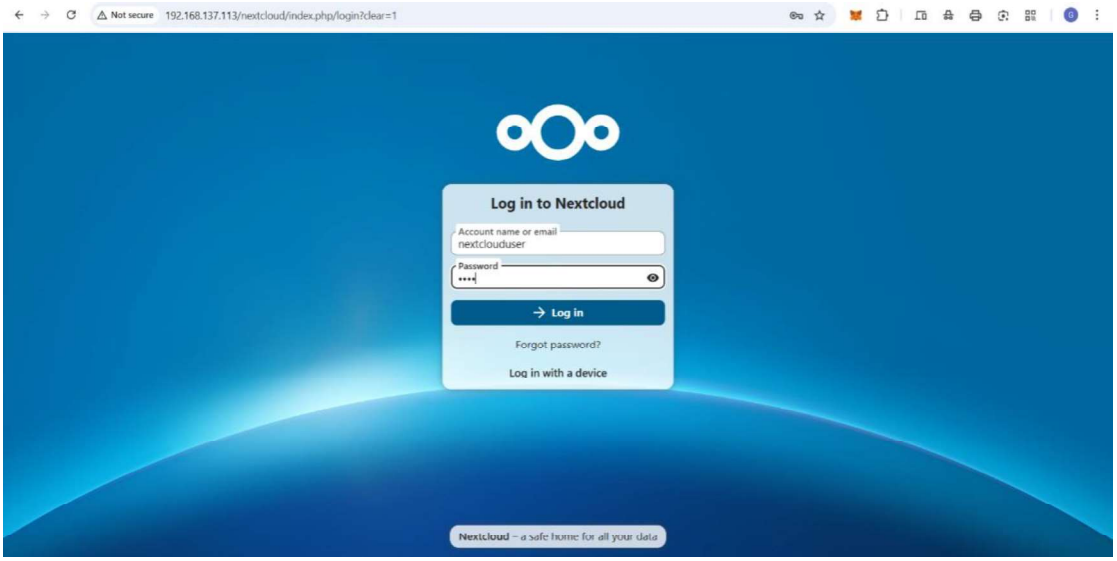


Fig-6.3

Next the login page pops up where we need login using the credentials of the admin which we have created in the previous step.

6.4 File Upload and Download Dashboard

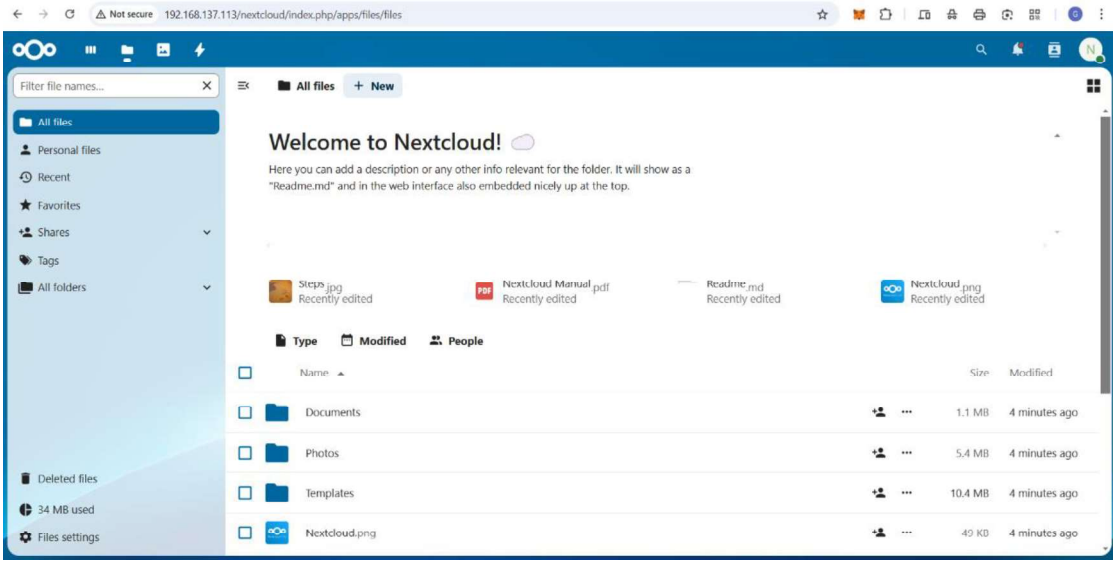


Fig-6.4

The user dashboard is then displayed which allows user to find all his resources in one place

6.5 Uploading New File

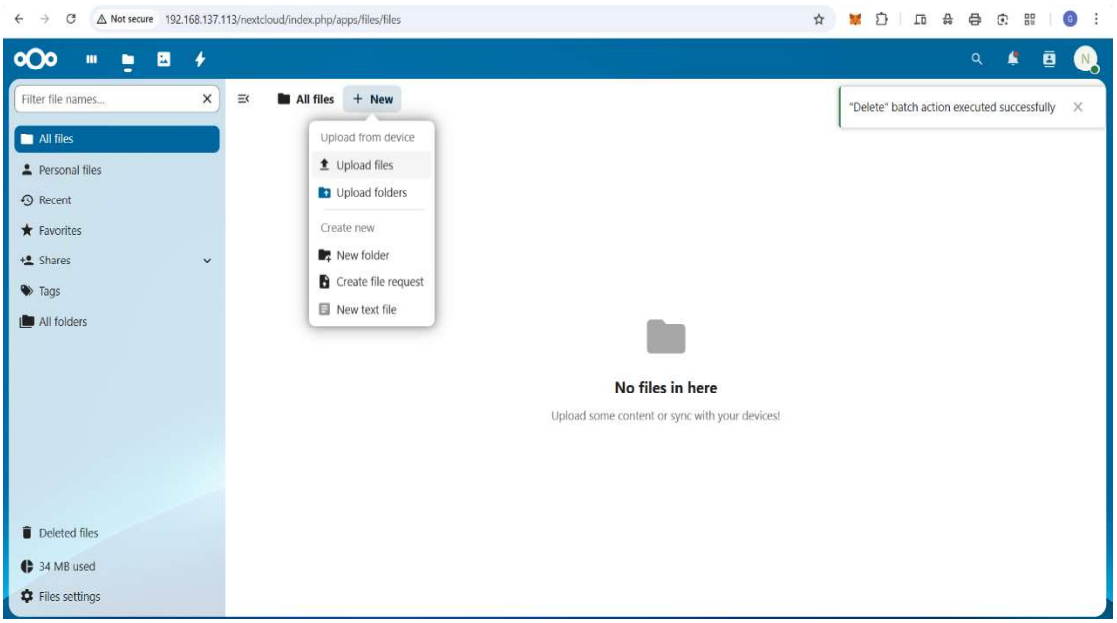


Fig-6.5

We then click on the new option which displays the list containing various formats and select the one applicable

6.6 Uploading the Image by opening the files in computer

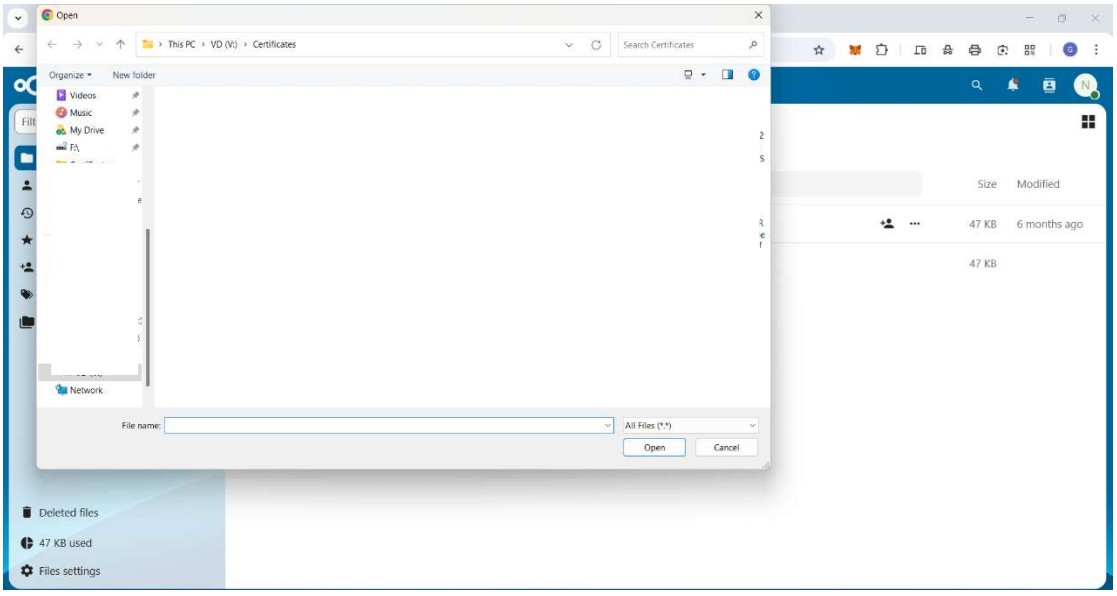


Fig-6.6

Then select the file which you would want to upload to the cloud

6.7 Uploaded file shown in the file in the app

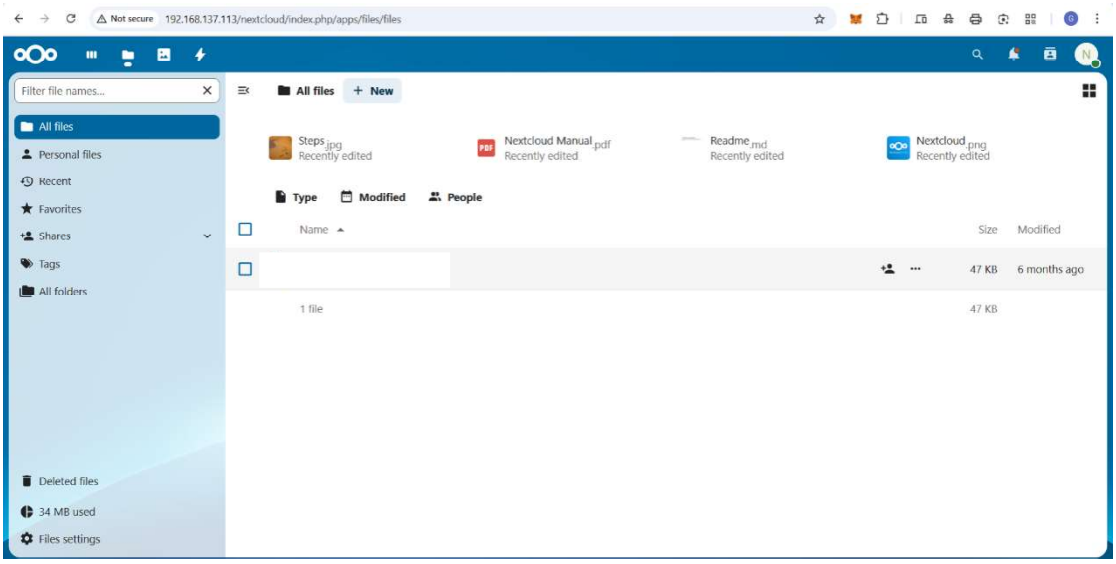


Fig-6.7

The selected file is uploaded into the cloud and can be accessed by connecting to the network

6.8 Uploading Another file as pdf

Here we upload another file with different format as this cloud supports all the file formats . it also upload same as above file

6.9 Uploaded Two file to the cloud

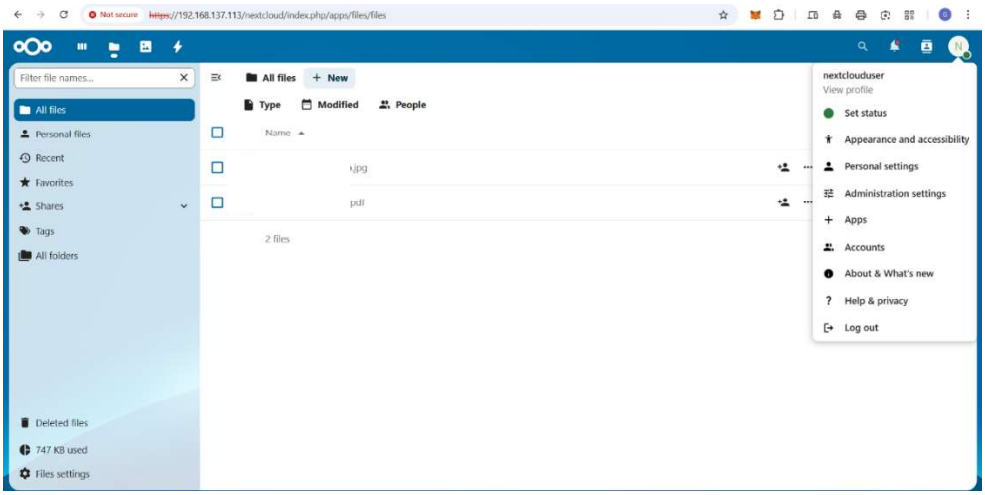


Fig-6.8

We can then find all the uploaded files on the dashboard

7. DISCUSSION OF RESULTS

7.1 Raspberry Pi Performance

The performance of the Raspberry Pi as a personal cloud storage server is largely dependent on the model used, available RAM, processor speed, and network connectivity. In this project, the Raspberry Pi 5 (8GB RAM, 2.4GHz Quad-core ARM Cortex-A76 processor) was used, which provided sufficient processing power for hosting Nextcloud and managing file operations. The system was able to handle multiple file transfers, database queries, and user authentication processes efficiently within a local network.

The storage read/write speeds are influenced by the external hard drive or SSD connected to the Raspberry Pi. Using a USB 3.0 SSD significantly improves file access speeds compared to traditional HDDs. Since the Raspberry Pi lacks native SATA or NVMe support, users must rely on USB interfaces, which may create a bottleneck for high-speed data transfers.

Network speed plays a crucial role in determining upload/download performance. The Ethernet connection (Gigabit LAN) on Raspberry Pi provides stable and faster data transfer compared to Wi-Fi, which can suffer from latency and packet loss. In a well-configured local network, file transfers were quick and reliable, but remote access performance depended on internet bandwidth and latency.

Thermal performance was another key consideration. During extended usage, CPU temperatures increased, especially when running multiple concurrent processes. To prevent thermal throttling, active cooling solutions such as heat sinks and fans were employed, which helped maintain optimal operating temperatures.

The Raspberry Pi was able to run Nextcloud smoothly for personal cloud storage, but performance declined when handling multiple simultaneous connections. For small-scale operations (1-5 users), the system worked efficiently, but for larger groups, performance optimizations, such as load balancing across multiple Raspberry Pi units, may be required.

Overall, Raspberry Pi offers a cost-effective yet powerful solution for personal cloud storage. With proper configuration, it can perform well in a self-hosted cloud environment, but users with higher performance needs might consider alternative hardware solutions like a mini PC or a dedicated NAS.

7.2 User Experience & Interaction

The user experience (UX) of a Raspberry Pi-based personal cloud storage system depends on interface design, ease of use, and accessibility. Nextcloud provides a web-based interface that is intuitive, allowing users to upload, download, and manage files through a drag-and-drop system. The interface is responsive and works seamlessly on both desktop and mobile browsers.

Setting up the system requires moderate technical knowledge, as users must configure Apache, PHP, and MySQL before installing Nextcloud. To improve accessibility, the project could integrate an installation wizard that simplifies setup, guiding users through database configuration, SSL installation, and firewall settings.

File access through mobile devices was tested using the Nextcloud mobile app, which provided a smooth experience. Users could upload, preview, and share files on the go. Additionally, automatic photo backup worked efficiently, making it a good alternative to services like Google Drive or iCloud.

Collaboration features such as file sharing, document editing (via Collabora Online), and version control improved user interaction. Users could create shared folders and manage permissions easily. However, real-time multi-user collaboration was slower compared to commercial cloud services due to hardware limitations.

Accessibility was further enhanced by integrating Dynamic DNS (DDNS), allowing users to remotely access their files without needing a static IP. However, network security measures such as firewall rules and two-factor authentication (2FA) are necessary to prevent unauthorized access.

Overall, user experience was positive, with responsive interfaces, mobile accessibility, and customizable features. However, further improvements in setup automation and real-time collaboration performance would enhance usability for less technical users.

7.3 Security

Security is a major concern for personal cloud storage, as unauthorized access can compromise sensitive user data. This system implemented end-to-end encryption through Nextcloud's built-in encryption module, ensuring that files remain protected during storage and transmission.

Authentication security was enhanced using two-factor authentication (2FA), requiring users to enter a one-time passcode (OTP) in addition to their password. Brute-force protection mechanisms were enabled to prevent repeated unauthorized login attempts.

The SSL/TLS encryption setup ensured secure HTTPS connections, preventing data interception during transmission. Self-signed certificates were initially used, but for better security, it is recommended to install Let's Encrypt SSL certificates to avoid browser warnings and improve trust.

A firewall (UFW) was configured to block unauthorized access to sensitive ports, while fail2ban was used to detect and block suspicious login attempts. Additionally, intrusion detection systems (IDS) can be integrated for enhanced monitoring.

Real-time security notifications via the LINE app provided instant alerts for suspicious activities, such as multiple failed login attempts or unauthorized file modifications. This feature helped users take immediate action against potential threats.

Despite these measures, security improvements can be made by implementing hardware-level encryption and regular security audits to identify vulnerabilities. A backup and recovery plan is essential to protect against ransomware attacks or accidental deletions.

7.4 Performance Metrics & Comparison

To evaluate system efficiency, various performance metrics were tested, including file transfer speeds, CPU load, memory usage, and latency. The Raspberry Pi was compared with other cloud storage solutions such as Google Drive and Dropbox.

CPU utilization was measured during file uploads, downloads, and indexing. The Raspberry Pi's processor usage peaked at 80-90% during heavy operations, while commercial cloud solutions experience no such limitations since they run on high-performance servers.

Memory usage was stable at 600-800MB RAM during normal operations. Performance was affected when multiple simultaneous users accessed the system, causing higher RAM usage and slight slowdowns.

Nextcloud's collaboration tools (document editing, file sharing) worked well, but real-time synchronization was slower than Google Drive. This was due to limited processing power and network bandwidth.

Overall, Raspberry Pi-based cloud storage is ideal for personal use but lacks the scalability of commercial services. However, it excels in data privacy and cost savings, making it a preferred choice for users prioritizing security and control.

7.5 System Performance

The Raspberry Pi-based cloud server performed efficiently for personal cloud storage but had limitations under heavy workloads. File transfer and access speeds were satisfactory within a local network but slower when accessed remotely.

Performance degraded under multiple concurrent user requests, as the Raspberry Pi's limited CPU power led to slower response times. For best results, usage should be restricted to 1-5 users for optimal performance.

The Nextcloud web interface was responsive, but thumbnail generation and indexing were resource-intensive. Enabling caching mechanisms helped reduce delays.

Power efficiency was a significant advantage. The Raspberry Pi consumed far less energy than a traditional NAS, making it an eco-friendly option.

Long-term performance can be improved by upgrading storage to SSDs, optimizing database queries, and enabling hardware acceleration for cryptographic functions.

7.6 Limitations & Areas for Improvement

- Limited processing power restricts multi-user efficiency.

- Lack of hardware RAID support limits data redundancy.
- Remote access speed depends on network bandwidth.
- Manual setup complexity can be reduced via automation.
- Security features can be enhanced with additional monitoring tools.

7.7 User Feedback & Future Enhancements

- Users appreciated cost savings and data privacy.
- Performance bottlenecks were observed under heavy loads.
- A simplified setup wizard was requested.
- Future upgrades could include AI-powered security alerts.
- Integration with IoT devices could enhance automation.

8. CONCLUSION

The implementation of a Raspberry Pi-based personal cloud storage system has demonstrated its effectiveness as a secure, cost-efficient, and flexible alternative to traditional cloud solutions. By configuring Nextcloud and OwnCloud on a Raspberry Pi, users gain full control over their data, eliminating concerns associated with third-party providers, such as unauthorized data access, limited free storage, and expensive subscription fees.

The system is designed with a modular and scalable approach, allowing users to expand storage by integrating additional external drives. The security features, including end-to-end encryption and real-time notifications, provide an added layer of protection against unauthorized access. These enhancements make the system a viable option for individuals and small businesses requiring private cloud storage.

While the system is functional and efficient, some challenges were identified. The initial setup requires a certain level of technical expertise, particularly in configuring Apache, PHP, and MySQL. Additionally, SSL implementation and server maintenance are crucial to ensuring data security and long-term system stability. Network performance is another factor, as internet speed and connection reliability impact remote access efficiency.

Overall, the project successfully delivers a self-hosted cloud storage solution that prioritizes user privacy, security, and cost-effectiveness. With continuous improvements, such as enhanced automation for system updates and security patches, this system can serve as a sustainable long-term alternative to commercial cloud services.

9. REFERENCES

[1] Zhou, K., & Wang, T. (2024). Personalized Interiors at Scale: Leveraging AI for Efficient and Customizable Design Solutions. Available: https://arxiv.org/abs/2405.19188.

[2] Le, M.-H., Chu, C.-B., Le, K.-D., Nguyen, T. V., Tran, M.-T., & Le, T.-N. (2023). VIDES: Virtual Interior Design via Natural Language and Visual Guidance. Available: https://arxiv.org/abs/2308.13795.

[3] Li, X., Benjamin, J., & Zhang, X. (2024). From Text to Blueprint: Leveraging Text-to-Image Tools for Floor Plan Creation. Available: https://arxiv.org/abs/2405.17236.

[4] Çelen, A., Han, G., Schindler, K., Van Gool, L., Armeni, I., Obukhov, A., & Wang, X. (2024). I-Design: Personalized LLM Interior Designer. Available: https://arxiv.org/abs/2404.02838.

[5] Zhou, K., & Wang, T. (2024). Integrating Aesthetics and Efficiency: AI-Driven Diffusion Models for Interior Design. Available: https://www.nature.com/articles/s41598-024-53318-3.

[6] Replicate API Documentation. (2024). Available: https://replicate.com/docs.

[7] Next.js Official Documentation. (2024). Available: https://nextjs.org/docs.

[8] Appwrite Documentation. (2024). Available: https://appwrite.io/docs.

[9] Neon Database Documentation. (2024). Available: https://neon.tech/docs.

[10] Drizzle ORM Documentation. (2024). Available: https://orm.drizzle.team/docs.

[11] Zhou, K., & Wang, T. (2024). Research on the Application of Artificial Intelligence in Interior Design. Available: https://ijsea.com/archive/volume13/issue7/IJSEA13071007.pdf.

[12] Le, M.-H., Chu, C.-B., Le, K.-D., Nguyen, T. V., Tran, M.-T., & Le, T.-N. (2023). Implementation of Artificial Intelligence in Interior Design: Systematic Literature Review. Available: https://www.researchgate.net/publication/382315632_Implementation_of_Artificial_Intelligence_in_Interior_Design_Systematic_Literature_Review.

[13] Li, X., Benjamin, J., & Zhang, X. (2024). Integrating Artificial Intelligence in Interior Design Education. Available: https://www.researchgate.net/publication/379347826_Integrating_Artificial_Intelligence_in_Interior_Design_Education_Concept_Development.

[14] Çelen, A., Han, G., Schindler, K., Van Gool, L., Armeni, I., Obukhov, A., & Wang, X. (2024). Integrating Aesthetics and Efficiency: AI-Driven Diffusion Models for Interior Design. Available: https://www.nature.com/articles/s41598-024-53318-3.

[15] Zhou, K., & Wang, T. (2024). Communicating AI for Architectural and Interior Design. Available: https://www.mdpi.com/2075-5309/14/9/2916.

[16] Li, C., Zhang, T., Du, X., Zhang, Y., & Xie, H. (2024). Generative AI for Architectural Design: A Literature Review. arXiv preprint arXiv:2404.01335. Available: https://arxiv.org/abs/2404.01335.

[17] Li, P., Li, B., & Li, Z. (2024). Sketch-to-Architecture: Generative AI-aided Architectural Design. arXiv preprint arXiv:2403.20186. Available: https://arxiv.org/abs/2403.20186.

[18] Zhou, N. (2024, December 11). How AI Is Making Buildings More Energy-Efficient. Time. Available: https://time.com/7201501/ai-buildings-energy-efficiency/.

[19] Taylor, C., & Fornoni, R. (2024, October 12). Travels in Interiors Hyperreality. Financial Times. Available: https://www.ft.com/content/fec22831-ac74-4d5b-8c6b-6ae4c23a5bf2.

[20] Royal Institute of British Architects. (2024). RIBA AI Report 2024. RIBA.