
Assignment- internetworking security

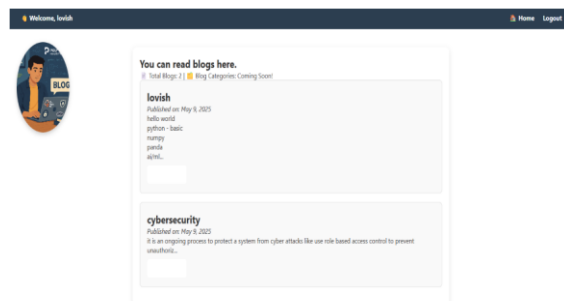
Group members: Deansingh Ramphul Kushalsing
Bhantooa

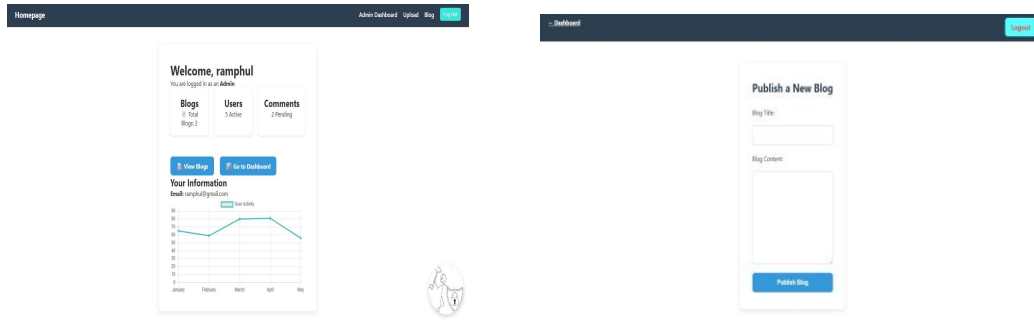
Contents

Introduction about the website:	2
Part 1: Website and Web Server Setup	3
Part 2: Implement Access Control Feature.....	11
Part 3: Firewall Configuration Using iptables	13
Part 4: Server Monitoring with Nagios or RABBIX	18
Task 5: Website Security Testing – SQL Injection & Forced Browsing.....	21
Task 6: Accessing the Website from Another Virtual Machine	23
Conclusion	25

Introduction about the website:

This project involves setting up a personal portfolio website on a virtual machine running Ubuntu. The website includes a login system and blog functionality, using a MySQL database to store user and blog data. The main goal is to deploy the site securely, then simulate and test common cybersecurity attacks such as SQL injection, brute-force login attempts, and unauthorized access to admin pages. This helps in understanding how vulnerabilities can be exploited and how to defend against them using secure coding practices and tools like firewalls (iptables).





This is our website.

Part 1: Website and Web Server Setup

Step 1: Update and Upgrade System Packages

Before installing any software, I made sure the system's package list and installed packages were up to date.

```

lovish@lovish-VirtualBox: ~
lovish@lovish-VirtualBox: ~$ sudo apt update && sudo apt upgrade
[sudo] password for lovish:
Get:4 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Hit:1 https://mu.archive.ubuntu.com/ubuntu noble InRelease
Get:2 https://mu.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 https://mu.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:5 https://mu.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [677 kB]
Get:6 https://mu.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [159 kB]
Get:7 https://mu.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [132 kB]
Get:8 https://mu.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [499 kB]
Get:9 https://mu.archive.ubuntu.com/ubuntu noble-updates/restricted Translation-en [96.4 kB]
Get:10 https://mu.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:11 https://mu.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [722 kB]
Get:12 https://mu.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [215 kB]
Get:13 https://mu.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [309 kB]
Get:14 https://mu.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:15 https://mu.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 B]
Get:16 https://mu.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:17 https://mu.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [11.7 kB]
Get:18 https://mu.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:19 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [501 kB]
Get:20 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [102 kB]
Get:21 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [7,232 B]
Get:22 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [482 kB]
Get:23 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-en [93.1 kB]
Get:24 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:25 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [564 kB]
Get:26 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [150 kB]
Get:27 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [51.9 kB]
Get:28 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]
Fetched 5,154 kB in 8s (616 kB/s)
Reading package lists... Done
Building dependency tree... Done

```

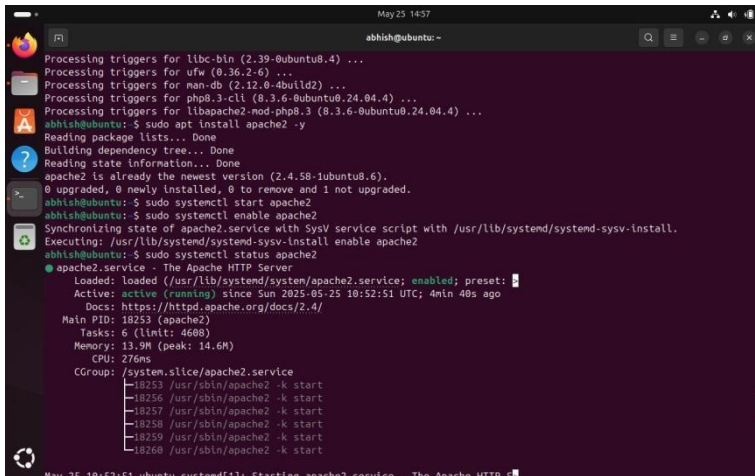
Explanation:

- **sudo apt update:**
This command updates the list of available packages and versions from the Ubuntu repositories. It does **not install** or upgrade any packages — it just refreshes the list.
- **sudo apt upgrade:**
This command installs the **newest versions** of all packages that are currently installed on the system. It upgrades the software to make sure everything is up to date and secure.

Figure 1: Running `sudo apt update && sudo apt upgrade` to update the system.

Step 2: Install Apache Web Server

Apache is the software that serves the website over the internet.



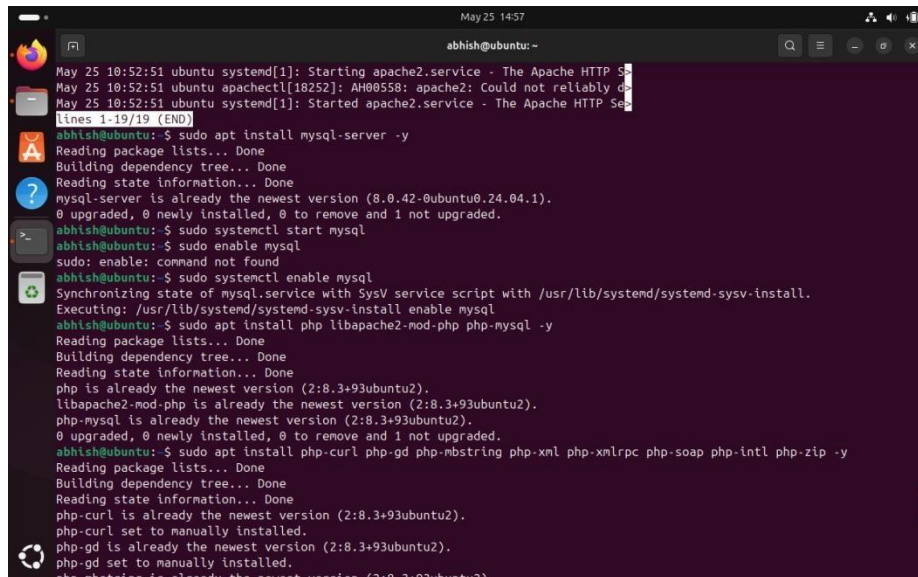
```
Processing triggers for libc-bin (2.39-0ubuntu8.4) ...
Processing triggers for ufw (0.36.2-6) ...
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for php8.3-cli (8.3.6-0ubuntu0.24.04.4) ...
Processing triggers for libapache2-mod-php8.3 (8.3.6-0ubuntu0.24.04.4) ...
abhis@ubuntu:~$ sudo apt install apache2 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apache2 is already the newest version (2.4.58-1ubuntu8.6).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
abhis@ubuntu:~$ sudo systemctl start apache2
abhis@ubuntu:~$ sudo systemctl enable apache2
Synchronizing state of apache2.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable apache2
abhis@ubuntu:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset:
   Active: active (running) since Sun 2025-05-25 10:52:51 UTC; 4min 40s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 18253 (apache2)
     Tasks: 6 (limit: 4608)
    Memory: 13.9M (peak: 14.6M)
       CPU: 276ms
   CGroup: /system.slice/apache2.service
           └─18253 /usr/sbin/apache2 -k start
             └─18256 /usr/sbin/apache2 -k start
               └─18257 /usr/sbin/apache2 -k start
                 └─18258 /usr/sbin/apache2 -k start
                   └─18259 /usr/sbin/apache2 -k start
                     └─18260 /usr/sbin/apache2 -k start
```

- apache2: The Apache web server package.
- -y: Automatically confirms the installation.

Figure 2: Apache installed successfully.

Step 3: Install MySQL Server

MySQL is the database system used to store website data like users and blogs.



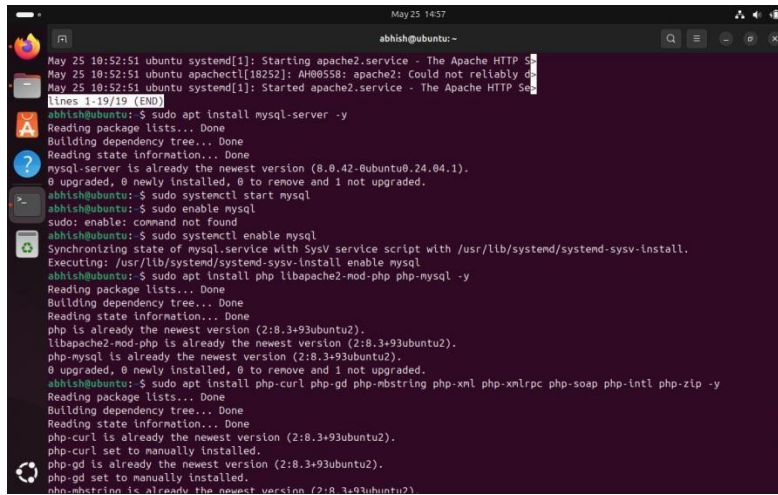
```
May 25 10:52:51 ubuntu systemd[1]: Starting apache2.service - The Apache HTTP S
May 25 10:52:51 ubuntu apachectl[18252]: AH00558: apache2: Could not reliably d
May 25 10:52:51 ubuntu systemd[1]: Started apache2.service - The Apache HTTP S
lines 1-19/19 (END)
abhis@ubuntu:~$ sudo apt install mysql-server -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
mysql-server is already the newest version (8.0.42-0ubuntu0.24.04.1).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
abhis@ubuntu:~$ sudo systemctl start mysql
abhis@ubuntu:~$ sudo enable mysql
sudo: enable: command not found
abhis@ubuntu:~$ sudo systemctl enable mysql
Synchronizing state of mysql.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable mysql
abhis@ubuntu:~$ sudo apt install php libapache2-mod-php php-mysql -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
php is already the newest version (2:8.3+93ubuntu2).
libapache2-mod-php is already the newest version (2:8.3+93ubuntu2).
php-mysql is already the newest version (2:8.3+93ubuntu2).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
abhis@ubuntu:~$ sudo apt install php-curl php-gd php-mbstring php-xml php-xmlrpc php-soap php-intl php-zip -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
php-curl is already the newest version (2:8.3+93ubuntu2).
php-curl set to manually installed.
php-gd is already the newest version (2:8.3+93ubuntu2).
php-gd set to manually installed.
php-mbstring is already the newest version (2:8.3+93ubuntu2).
```

- Installs the MySQL database server.

Figure 4: MySQL server installation completed.

Step 4: Install PHP

PHP is a programming language used to create dynamic web pages.

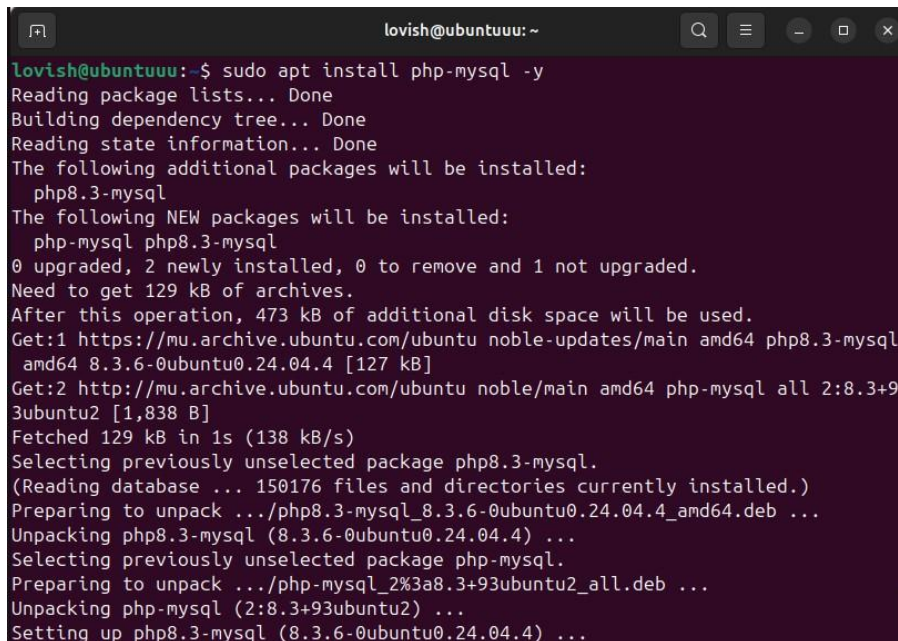
A terminal window titled 'abhishek@ubuntu: ~' showing the installation of PHP and MySQL. The user runs 'sudo apt install mysql-server -y', which installs MySQL 8.0.42. Then, they run 'sudo systemctl start mysql' and 'sudo systemctl enable mysql'. Next, they run 'sudo apt install php libapache2-mod-php php-mysql -y', which installs PHP 8.3.93 and the Apache module. The terminal output shows various status messages and package details.

```
May 25 18:52:51 ubuntu systemd[1]: Starting apache2.service - The Apache HTTP Server...
May 25 18:52:51 ubuntu apachectl[18252]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, please add the appropriate entry to your host file.
May 25 18:52:51 ubuntu systemd[1]: Started apache2.service - The Apache HTTP Server.
abhishek@ubuntu:~$ sudo apt install mysql-server -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
mysql-server is already the newest version (8.0.42-0ubuntu0.24.04.1).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
abhishek@ubuntu:~$ sudo systemctl start mysql
abhishek@ubuntu:~$ sudo enable mysql
sudo: enable: command not found
abhishek@ubuntu:~$ sudo systemctl enable mysql
Synchronizing state of mysql.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable mysql
abhishek@ubuntu:~$ sudo apt install php libapache2-mod-php php-mysql -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
php is already the newest version (2:8.3+93ubuntu2).
libapache2-mod-php is already the newest version (2:8.3+93ubuntu2).
php-mysql is already the newest version (2:8.3+93ubuntu2).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
abhishek@ubuntu:~$ sudo apt install php-curl php-gd php-mbstring php-xml php-xmlrpc php-soap php-intl php-zip -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
php-curl is already the newest version (2:8.3+93ubuntu2).
php-curl set to manually installed.
php-gd is already the newest version (2:8.3+93ubuntu2).
php-gd set to manually installed.
php-mbstring is already the newest version (2:8.3+93ubuntu2).
```

- php: Installs the core PHP package.
- libapache2-mod-php: Allows Apache to run PHP files.

Step 5: Install PHP-MySQL Extension

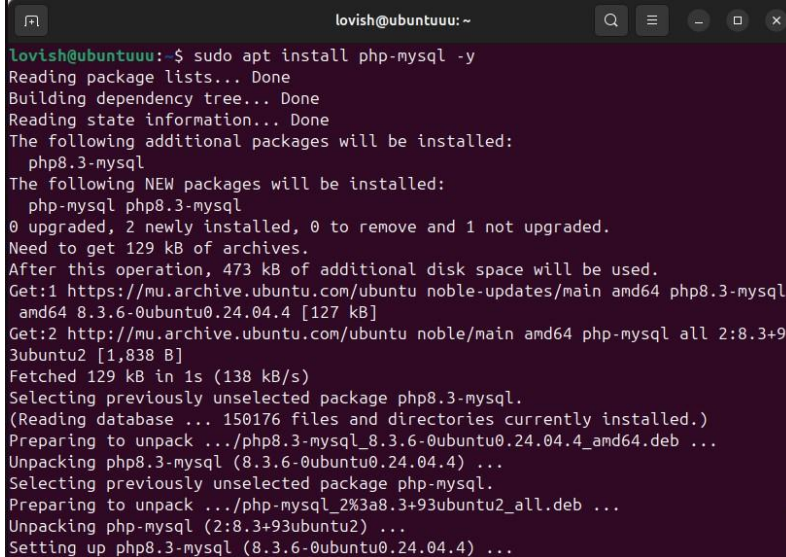
This package allows PHP to connect and communicate with MySQL databases.

A terminal window titled 'lovish@ubuntu: ~' showing the installation of the PHP-MySQL extension. The user runs 'sudo apt install php-mysql -y'. The terminal output shows the package lists, dependency tree, and state information. It indicates that php8.3-mysql will be installed along with php-mysql. The user then runs 'sudo apt install php8.3-mysql -y', which installs php8.3-mysql. The terminal output shows the package lists, dependency tree, and state information. It indicates that php8.3-mysql will be installed along with php-mysql. The user then runs 'sudo apt install php8.3-mysql -y', which installs php8.3-mysql. The terminal output shows the package lists, dependency tree, and state information. It indicates that php8.3-mysql will be installed along with php-mysql.


```
lovish@ubuntu:~$ sudo apt install php-mysql -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  php8.3-mysql
The following NEW packages will be installed:
  php-mysql php8.3-mysql
0 upgraded, 2 newly installed, 0 to remove and 1 not upgraded.
Need to get 129 kB of archives.
After this operation, 473 kB of additional disk space will be used.
Get:1 https://mu.archive.ubuntu.com/ubuntu noble-updates/main amd64 php8.3-mysql amd64 8.3.6-0ubuntu0.24.04.4 [127 kB]
Get:2 http://mu.archive.ubuntu.com/ubuntu noble/main amd64 php-mysql all 2:8.3+93ubuntu2 [1,838 B]
Fetched 129 kB in 1s (138 kB/s)
Selecting previously unselected package php8.3-mysql.
(Reading database ... 150176 files and directories currently installed.)
Preparing to unpack .../php8.3-mysql_8.3.6-0ubuntu0.24.04.4_amd64.deb ...
Unpacking php8.3-mysql (8.3.6-0ubuntu0.24.04.4) ...
Selecting previously unselected package php-mysql.
Preparing to unpack .../php-mysql_2%3a8.3+93ubuntu2_all.deb ...
Unpacking php-mysql (2:8.3+93ubuntu2) ...
Setting up php8.3-mysql (8.3.6-0ubuntu0.24.04.4) ...
```

 *Figure 5: PHP-MySQL module installed to connect PHP with MySQL Step*

6: Install phpMyAdmin phpMyAdmin is a web-based interface to manage MySQL databases easily.



```
lovish@ubuntu:~$ sudo apt install php-mysql -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  php8.3-mysql
The following NEW packages will be installed:
  php-mysql php8.3-mysql
0 upgraded, 2 newly installed, 0 to remove and 1 not upgraded.
Need to get 129 kB of archives.
After this operation, 473 kB of additional disk space will be used.
Get:1 https://mu.archive.ubuntu.com/ubuntu noble-updates/main amd64 php8.3-mysql
amd64 8.3.6-0ubuntu0.24.04.4 [127 kB]
Get:2 http://mu.archive.ubuntu.com/ubuntu noble/main amd64 php-mysql all 2:8.3+9
3ubuntu2 [1,838 B]
Fetched 129 kB in 1s (138 kB/s)
Selecting previously unselected package php8.3-mysql.
(Reading database ... 150176 files and directories currently installed.)
Preparing to unpack .../php8.3-mysql_8.3.6-0ubuntu0.24.04.4_and64.deb ...
Unpacking php8.3-mysql (8.3.6-0ubuntu0.24.04.4) ...
Selecting previously unselected package php-mysql.
Preparing to unpack .../php-mysql_2%3a8.3+93ubuntu2_all.deb ...
Unpacking php-mysql (2:8.3+93ubuntu2) ...
Setting up php8.3-mysql (8.3.6-0ubuntu0.24.04.4) ...
```

 *Figure 6: phpMyAdmin installed.*


Step 7: Secure the MySQL Root User

I logged into MySQL and changed the root password for security and phpMyAdmin compatibility.



```
Query OK, 0 rows affected (0.53 sec)

mysql> ALTER USER 'root'@'localhost'
-> -> IDENTIFIED WITH
-> -> mysql_native_password BY
-> -> '1234'
-> -> FLUSH PRIVILEGE
-> -> EXIT;
```

 *Figure 7: MySQL root user secured with native password.*

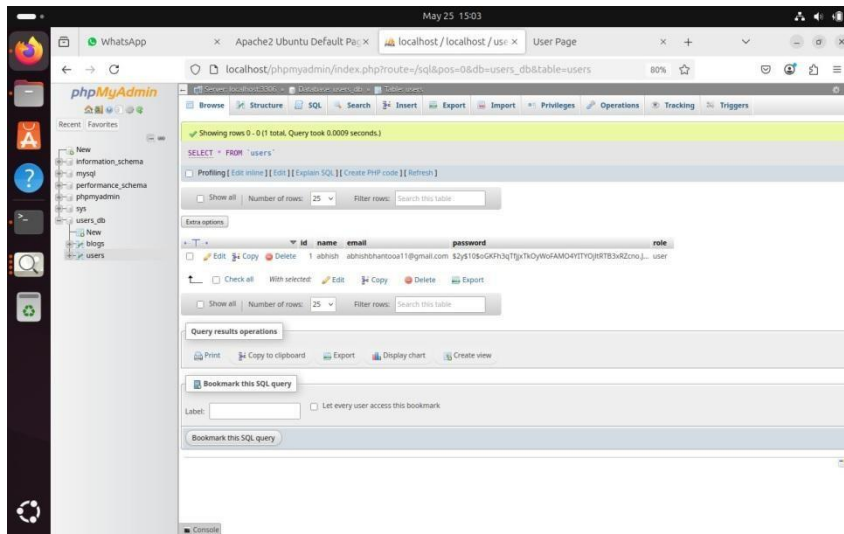
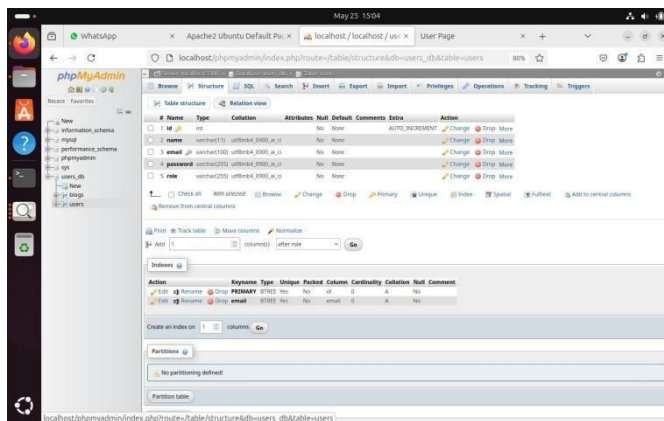
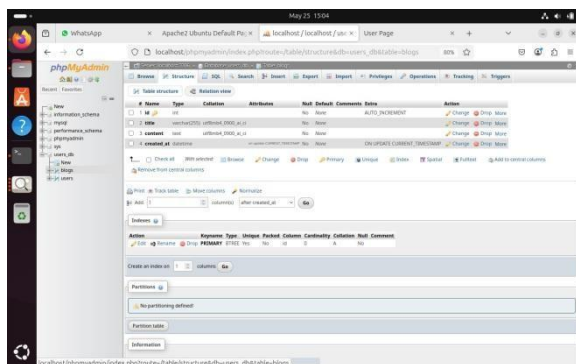


Figure 8: Database



Users table



Blogs table

Step 8: Move Website to Apache Directory and Set Permissions

After preparing the website files, I moved the portfolio folder to Apache's web root, set the correct ownership and permissions, and then reloaded Apache to apply the changes.

```
lovesh@ubuntu: ~  
affiliates. Other names may be trademarks of their respective  
owners.  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '1234';  
Query OK, 0 rows affected (0.07 sec)  
mysql> FLUSH PRIVILEGES;  
Query OK, 0 rows affected (0.08 sec)  
mysql> EXIT;  
Bye  
lovesh@ubuntu:~$ sudo systemctl restart mysql  
lovesh@ubuntu:~$ sudo ln -s /etc/phpmyadmin/apache.conf /etc/apache2/conf-available/phpmyadmin.conf  
ln: failed to create symbolic link '/etc/apache2/conf-available/phpmyadmin.conf': File exists  
lovesh@ubuntu:~$ sudo a2enconf phpmyadmin  
Conf phpmyadmin already enabled  
lovesh@ubuntu:~$ sudo systemctl reload apache2  
lovesh@ubuntu:~$ sudo mv ~/Downloads/portfolio /var/www/html/  
lovesh@ubuntu:~$
```

- **sudo mv:** Moves the portfolio directory from the Downloads folder to /var/www/html/, which is Apache's default web directory.

Step 9: Set Ownership of Website Files

To ensure that Apache has the correct permissions to read and serve the website files, I changed the ownership of the entire portfolio directory to the Apache user and group.

```
lovesh@ubuntu: ~  
owners.  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '1234';  
Query OK, 0 rows affected (0.07 sec)  
mysql> FLUSH PRIVILEGES;  
Query OK, 0 rows affected (0.08 sec)  
mysql> EXIT;  
Bye  
lovesh@ubuntu:~$ sudo systemctl restart mysql  
lovesh@ubuntu:~$ sudo ln -s /etc/phpmyadmin/apache.conf /etc/apache2/conf-available/phpmyadmin.conf  
ln: failed to create symbolic link '/etc/apache2/conf-available/phpmyadmin.conf': File exists  
lovesh@ubuntu:~$ sudo a2enconf phpmyadmin  
Conf phpmyadmin already enabled  
lovesh@ubuntu:~$ sudo systemctl reload apache2  
lovesh@ubuntu:~$ sudo mv ~/Downloads/portfolio /var/www/html/  
lovesh@ubuntu:~$ sudo chown -R www-data:www-data /var/www/html/portfolio  
lovesh@ubuntu:~$
```

- **sudo chown:** Changes the owner of files or directories.

- **-R**: Applies the ownership change **recursively** to all files and subdirectories.

This step is important to prevent **403 Forbidden** errors and to allow Apache to properly access the website files.

Figure 9: Ownership of the portfolio directory set to www-data for secure web server access.

Step 10: Set File Permissions for Website Directory

After setting the ownership, I configured the appropriate permissions on the portfolio directory to ensure that Apache could read and execute the website files securely.

```

lovish@ubuntu:~$ mysql
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY '1234';
Query OK, 0 rows affected (0.07 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.08 sec)

mysql> EXIT;
Bye
lovish@ubuntu:~$ sudo systemctl restart mysql
lovish@ubuntu:~$ sudo ln -s /etc/phpmyadmin/apache.conf /etc/apache2/conf-available/phpmyadmin.conf
ln: failed to create symbolic link '/etc/apache2/conf-available/phpmyadmin.conf': File exists
lovish@ubuntu:~$ sudo a2enconf phpmyadmin
Conf phpmyadmin already enabled
lovish@ubuntu:~$ sudo systemctl reload apache2
lovish@ubuntu:~$ sudo mv ~/Downloads/portfolio /var/www/html/
lovish@ubuntu:~$ sudo chown -R www-data:www-data /var/www/html/portfolio
lovish@ubuntu:~$ sudo chmod -R 755 /var/www/html/portfolio
lovish@ubuntu:~$

```

- **sudo chmod**: Changes file and directory permissions.
- **-R**: Applies the permissions **recursively** to all files and subdirectories.
- **755**:
 - Owner (www-data) → Read, write, and execute
 - Group and others → Read and execute only This step is

critical for maintaining **security and functionality**:

- Apache can execute and read the files.
- Others can access the website content, but **cannot modify** it.

Figure 10: File permissions set to 755 for all files and folders inside the portfolio directory.

Step 11: Reload Apache to Apply Changes

After setting the correct file ownership and permissions, I reloaded Apache to apply any changes without interrupting ongoing connections.

```
abhisht@ubuntu: ~  
Processing triggers for libc-bin (2.39-0ubuntu8.4) ...  
Processing triggers for ufw (0.36.2-6) ...  
Processing triggers for man-db (2.12.0-4build2) ...  
Processing triggers for php8.3-cli (8.3.6-0ubuntu0.24.04.4) ...  
Processing triggers for libapache2-mod-php8.3 (8.3.6-0ubuntu0.24.04.4) ...  
abhisht@ubuntu:~$ sudo apt install apache2 -y  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
apache2 is already the newest version (2.4.58-1ubuntu8.6).  
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.  
abhisht@ubuntu:~$ sudo systemctl start apache2  
abhisht@ubuntu:~$ sudo systemctl enable apache2  
Synchronizing state of apache2.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.  
Executing: /usr/lib/systemd/systemd-sysv-install enable apache2  
abhisht@ubuntu:~$ sudo systemctl status apache2  
● apache2.service - The Apache HTTP Server  
   Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset:   
   Active: active (running) since Sun 2025-05-25 10:52:51 UTC; 4min 48s ago  
     Docs: https://httpd.apache.org/docs/2.4/  
   Main PID: 18253 (apache2)  
     Tasks: 6 (limit: 4608)  
    Memory: 13.9M (peak: 14.6M)  
       CPU: 276ms  
    CGroup: /system.slice/apache2.service  
            └─18253 /usr/sbin/apache2 -k start  
              18256 /usr/sbin/apache2 -k start  
              18257 /usr/sbin/apache2 -k start  
              18258 /usr/sbin/apache2 -k start  
              18259 /usr/sbin/apache2 -k start  
              18260 /usr/sbin/apache2 -k start  
May 25 10:52:51 ubuntu systemd[1]: Starting apache2 service - The Apache HTTP S
```

- **sudo systemctl:** Manages system services (like Apache).
- **reload apache2:** Reloads Apache configuration and updates the file structure changes (like new folders or permissions) **without stopping the server**.

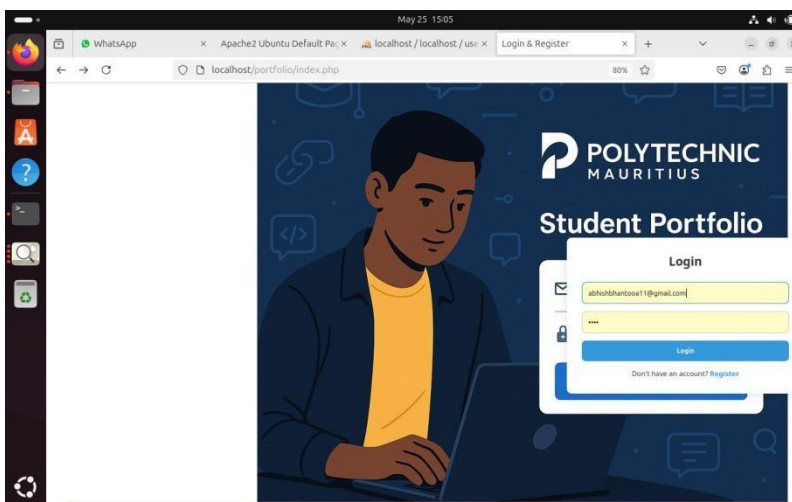
Reloading is a safe way to apply changes without downtime.

Figure 11: Apache server reloaded to apply updated ownership and permissions for the portfolio site.

Step 12: Website Access

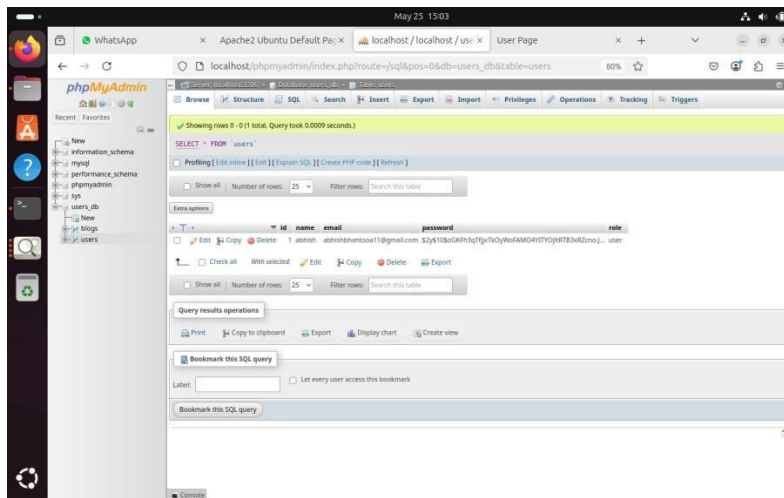
Portfolio Site:

<http://192.168.100.197/portfolio>



phpMyAdmin Interface:

<http://192.168.100.197/phpmyadmin>



Part 2: Implement Access Control Feature

Implement Role-Based Access Control (RBAC)

To ensure that users only access content they're authorized for, I implemented **Role-Based Access Control (RBAC)** using PHP. Each user is assigned a role during registration, and upon login, the user is redirected to a different page depending on their role.

```
$name = $_POST['name']; // Get name input from the form
$email = $_POST['email']; // Get email input from the form
$password = password_hash($_POST['password'], PASSWORD_DEFAULT); // Hash the password securely
$role = $_POST['role']; // Get selected user role from the form
```

✓ When a user registers, their role (admin or user) is saved in the database.

```

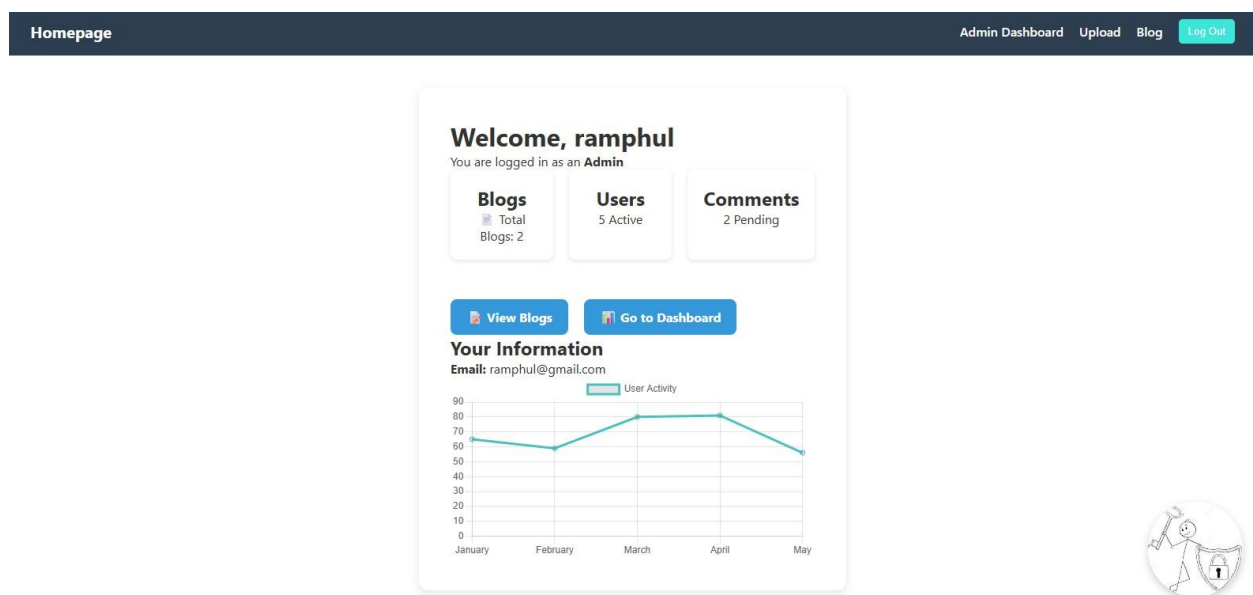
$user = $result->fetch_assoc(); // Fetch user data as associative array
if (password_verify($password, $user['password'])) {
    // If password is correct, store user info in session
    $_SESSION['name'] = $user['name'];
    $_SESSION['email'] = $user['email'];

    // Redirect user based on their role
    if ($user['role'] === 'admin') {
        header("Location: admin_page.php");
    } else {
        header("Location: user_page.php");
    }
    exit(); // Stop further script execution
}

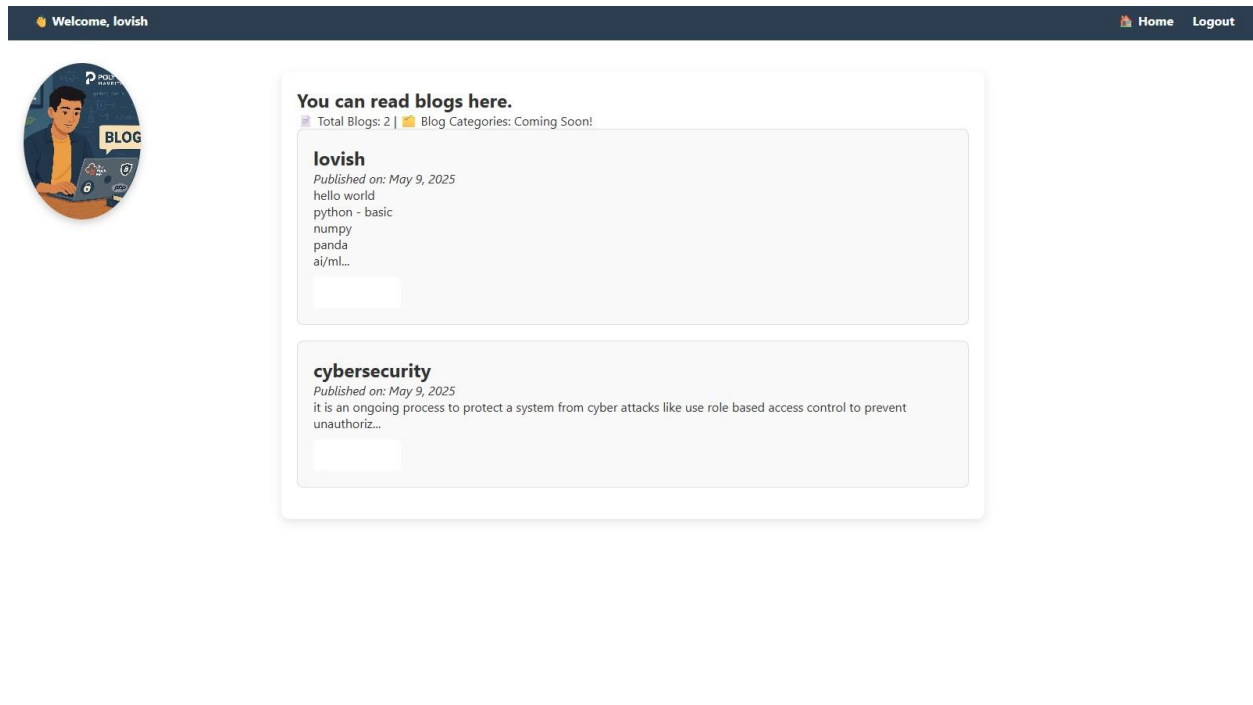
```

After login:

- **Admins** are redirected to admin_page.php



- **Regular users** are redirected to user_page.php



Part 3: Firewall Configuration Using iptables

To secure the server, I configured the firewall using iptables. The goal was to **restrict incoming traffic** and only allow essential services such as HTTP (port 80), HTTPS (port 443), and SSH (port 22) from a specific IP address.

Step 1: Update and Upgrade System Packages

Before making firewall changes, I ensured the system was up to date.

```
lovish@ubuntu:~$ sudo apt update && sudo apt upgrade
[sudo] password for lovish:
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:1 https://mu.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 https://mu.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 https://mu.archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
1 package can be upgraded. Run 'apt list --upgradable' to see it.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
Get more security updates through Ubuntu Pro with 'esm-apps' enabled:
  php-symfony-config php-symfony-dependency-injection php-symfony-cache
  php-symfony-var-exporter php-symfony-expression-language
  php-symfony-filesystem
Learn more about Ubuntu Pro at https://ubuntu.com/pro
The following upgrades have been deferred due to phasing:
  ubuntu-drivers-common
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
lovish@ubuntu:~$
```


`sudo apt update`: Updates the list of available packages. `sudo`

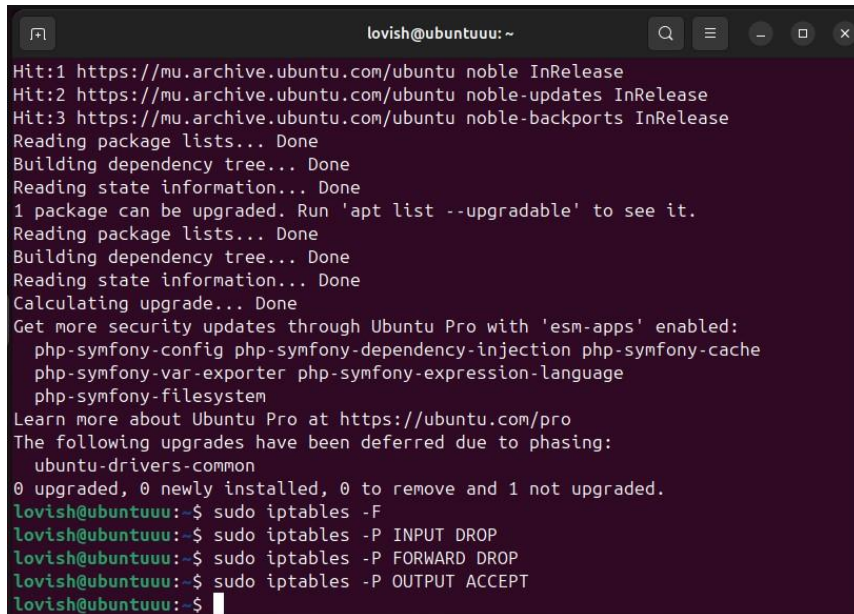
`apt upgrade`: Installs the latest versions of all installed packages.

Step 2: Flush Existing iptables Rules

Cleared all existing firewall rules to start with a clean configuration.

Step 3: Set Default Policies

Configured the default iptables behavior to **deny incoming and forwarded traffic** but **allow outgoing** connections.

A terminal window titled 'lovish@ubuntu: ~' showing the output of 'sudo apt update' and subsequent 'iptables' commands. The update output includes package lists, dependency trees, and a list of deferred upgrades. The iptables commands set the default policy for INPUT, FORWARD, and OUTPUT chains to DROP, ACCEPT, and ACCEPT respectively.

```
lovish@ubuntu: ~  
Hit:1 https://mu.archive.ubuntu.com/ubuntu noble InRelease  
Hit:2 https://mu.archive.ubuntu.com/ubuntu noble-updates InRelease  
Hit:3 https://mu.archive.ubuntu.com/ubuntu noble-backports InRelease  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
1 package can be upgraded. Run 'apt list --upgradable' to see it.  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
Calculating upgrade... Done  
Get more security updates through Ubuntu Pro with 'esm-apps' enabled:  
  php-symfony-config php-symfony-dependency-injection php-symfony-cache  
  php-symfony-var-exporter php-symfony-expression-language  
  php-symfony-filesystem  
Learn more about Ubuntu Pro at https://ubuntu.com/pro  
The following upgrades have been deferred due to phasing:  
  ubuntu-drivers-common  
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.  
lovish@ubuntu:~$ sudo iptables -F  
lovish@ubuntu:~$ sudo iptables -P INPUT DROP  
lovish@ubuntu:~$ sudo iptables -P FORWARD DROP  
lovish@ubuntu:~$ sudo iptables -P OUTPUT ACCEPT  
lovish@ubuntu:~$
```

- INPUT DROP: Deny all incoming traffic by default.
- FORWARD DROP: Deny forwarding traffic.
- OUTPUT ACCEPT: Allow all outgoing connections.

Step 4: Allow Localhost (Loopback) Access

Enabled internal communications on the system.

Step 5: Allow Established and Related Connections

Permits return traffic from outgoing connections and connections related to already established sessions.

```
lovish@ubuntu: ~  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
1 package can be upgraded. Run 'apt list --upgradable' to see it.  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
Calculating upgrade... Done  
Get more security updates through Ubuntu Pro with 'esm-apps' enabled:  
  php-symfony-config php-symfony-dependency-injection php-symfony-cache  
  php-symfony-var-exporter php-symfony-expression-language  
  php-symfony-filesystem  
Learn more about Ubuntu Pro at https://ubuntu.com/pro  
The following upgrades have been deferred due to phasing:  
  ubuntu-drivers-common  
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.  
lovish@ubuntu:~$ sudo iptables -F  
lovish@ubuntu:~$ sudo iptables -P INPUT DROP  
lovish@ubuntu:~$ sudo iptables -P FORWARD DROP  
lovish@ubuntu:~$ sudo iptables -P OUTPUT ACCEPT  
lovish@ubuntu:~$ sudo iptables -A INPUT -i lo -j ACCEPT  
lovish@ubuntu:~$ sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT  
lovish@ubuntu:~$
```

- `-i lo`: Targets the loopback interface (127.0.0.1).
- `-j ACCEPT`: Allows the traffic.
- Uses `conntrack` module to allow traffic that's part of an existing or related connection. Step 6:

Allow HTTP Traffic (Port 80)

Allows access to the web server via HTTP.

Step 7: Allow HTTPS Traffic (Port 443)

Allows encrypted traffic (HTTPS) to reach the web server.

```
lovish@ubuntu:~$ sudo apt-get update
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
Get more security updates through Ubuntu Pro with 'esm-apps' enabled:
  php-symfony-config php-symfony-dependency-injection php-symfony-cache
  php-symfony-var-exporter php-symfony-expression-language
  php-symfony-filesystem
Learn more about Ubuntu Pro at https://ubuntu.com/pro
The following upgrades have been deferred due to phasing:
  ubuntu-drivers-common
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
lovish@ubuntu:~$ sudo iptables -F
lovish@ubuntu:~$ sudo iptables -P INPUT DROP
lovish@ubuntu:~$ sudo iptables -P FORWARD DROP
lovish@ubuntu:~$ sudo iptables -P OUTPUT ACCEPT
lovish@ubuntu:~$ sudo iptables -A INPUT -i lo -j ACCEPT
lovish@ubuntu:~$ sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
lovish@ubuntu:~$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
lovish@ubuntu:~$ sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables v1.8.10 (nf_tables): unknown option "--dport"
Try 'iptables -h' or 'iptables --help' for more information.
lovish@ubuntu:~$ sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT
lovish@ubuntu:~$
```

- --dport 80: Targets TCP traffic on port 80.
- --dport 443: Targets TCP traffic on port 443.

Step 8: Allow SSH Access From Specific IP

Restricts SSH access to only one trusted IP address.

```
lovish@ubuntu:~$ sudo iptables -F
lovish@ubuntu:~$ sudo iptables -P INPUT DROP
lovish@ubuntu:~$ sudo iptables -P FORWARD DROP
lovish@ubuntu:~$ sudo iptables -P OUTPUT ACCEPT
lovish@ubuntu:~$ sudo iptables -A INPUT -i lo -j ACCEPT
lovish@ubuntu:~$ sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
lovish@ubuntu:~$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
lovish@ubuntu:~$ sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables v1.8.10 (nf_tables): unknown option "--dport"
Try 'iptables -h' or 'iptables --help' for more information.
lovish@ubuntu:~$ sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT
lovish@ubuntu:~$ sudo iptables -A INPUT -p tcp -s 192.168.100.198 --dport 22 -j ACCEPT
lovish@ubuntu:~$ sudo iptables -L -n -v
Chain INPUT (policy DROP 72 packets, 16224 bytes)
```

- -s 192.168.100.198: Only allows SSH from this IP address. □ --dport 22: SSH port.

Step 9: View Current Firewall Rules

To confirm the firewall rules are set correctly, I listed them.

```
May 25 14:14
lovish@ubuntu:~$ sudo iptables -F
lovish@ubuntu:~$ sudo iptables -P INPUT DROP
lovish@ubuntu:~$ sudo iptables -P FORWARD DROP
lovish@ubuntu:~$ sudo iptables -P OUTPUT ACCEPT
lovish@ubuntu:~$ sudo iptables -A INPUT -i lo -j ACCEPT
lovish@ubuntu:~$ sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
lovish@ubuntu:~$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
lovish@ubuntu:~$ sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT
iptables v1.8.10 (nf_tables): unknown option "--deport"
Try 'iptables -h' or 'iptables --help' for more information.
lovish@ubuntu:~$ sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT
lovish@ubuntu:~$ sudo iptables -A INPUT -p tcp -s 192.168.100.198 --dport 22 -j ACCEPT
lovish@ubuntu:~$ sudo iptables -L -n -v
Chain INPUT (policy DROP 72 packets, 16224 bytes)
  pkts bytes target     prot opt in     out     source               destination
    16 3424 ACCEPT     0  --  lo      *      0.0.0.0/0            0.0.0.0/0
     7 1399 ACCEPT     0  --  *      *      0.0.0.0/0            0.0.0.0/0            ctstate RELATED,ESTABLISHED
      0 0 ACCEPT     6  --  *      *      0.0.0.0/0            0.0.0.0/0            tcp dpt:80
      0 0 ACCEPT     6  --  *      *      0.0.0.0/0            0.0.0.0/0            tcp dpt:443
      0 0 ACCEPT     6  --  *      *      192.168.100.198      0.0.0.0/0            tcp dpt:22

Chain FORWARD (policy DROP 0 packets, 0 bytes)
  pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 81 packets, 8892 bytes)
  pkts bytes target     prot opt in     out     source               destination
lovish@ubuntu:~$
```

- -L: Lists all rules.
- -n: Shows numeric values (no DNS). □ -v: Verbose output.

Step 10: Save iptables Rules

Ensures the iptables configuration is preserved after a system reboot.

```
May 25 14:17
lovish@ubuntu:~$ sudo ufw
The following NEW packages will be installed:
  iptables-persistent netfilter-persistent
0 upgraded, 2 newly installed, 1 to remove and 1 not upgraded.
Need to get 14.3 kB of archives.
After this operation, 700 kB disk space will be freed.
Do you want to continue? [Y/n] y
Get:1 https://nu.archive.ubuntu.com/ubuntu noble/universe amd64 netfilter-persistent all 1.0.20 [7,402 B]
Get:2 https://nu.archive.ubuntu.com/ubuntu noble/universe amd64 iptables-persistent all 1.0.20 [6,946 B]
Fetched 14.3 kB in 1s (13.7 kB/s)
Preconfiguring packages ...
(Reading database ... 156736 files and directories currently installed.)
Removing ufw (0.36.2-6) ...
Skip stopping firewalls ufw (not enabled)
Selecting previously unselected package netfilter-persistent.
(Reading database ... 156641 files and directories currently installed.)
Preparing to unpack .../netfilter-persistent_1.0.20_all.deb ...
Unpacking netfilter-persistent (1.0.20) ...
Selecting previously unselected package iptables-persistent.
Preparing to unpack .../iptables-persistent_1.0.20_all.deb ...
Unpacking iptables-persistent (1.0.20) ...
Setting up netfilter-persistent (1.0.20) ...
Created symlink /etc/systemd/system/netfilter-persistent.service → /usr/lib/systemd/system/netfilter-persistent.service.
Created symlink /etc/systemd/system/netfilter-persistent.service → /usr/lib/systemd/system/netfilter-persistent.service.
Created symlink /etc/systemd/system/multi-user.target.wants/netfilter-persistent.service → /usr/lib/systemd/system/netfilter-persistent.service.
Setting up iptables-persistent (1.0.20) ...
Processing triggers for man-db (2.12.0-4build2) ...
lovish@ubuntu:~$ sudo netfilter-persistent save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/15-ip4tables save
run-parts: executing /usr/share/netfilter-persistent/plugins.d/25-ip6tables save
lovish@ubuntu:~$
```

- Saves current iptables rules using the netfilter-persistent service.

Part 4: Server Monitoring with Nagios or RABBIX

Installation of Zabbix Agent

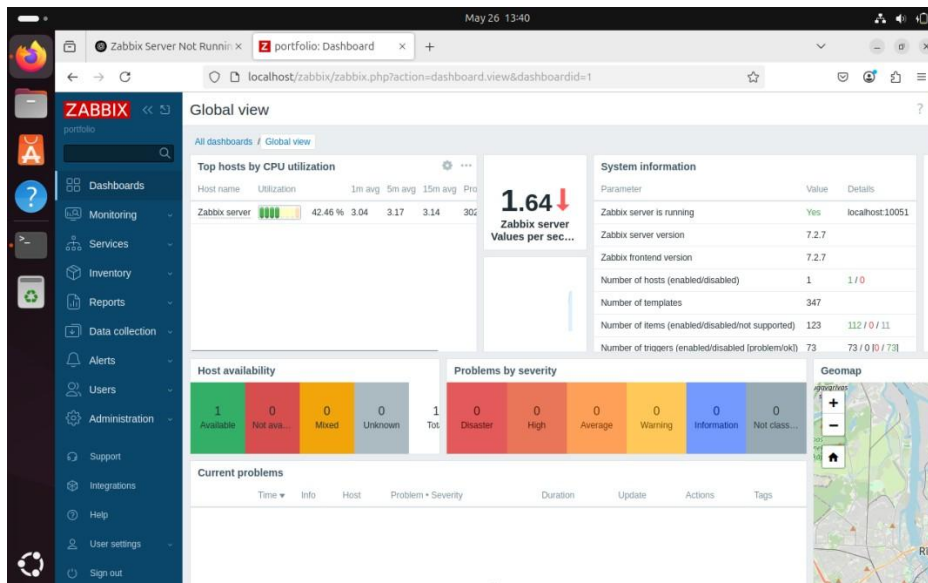
To monitor the website server, I installed the Zabbix agent, which collects system and service data and sends it to the Zabbix monitoring server.

The installation was done using the following commands on the web server:

```
lovis@ubuntu:~$ sudo apt install zabbix-agent2-plugin-mongodb zabbix-agent2-plugin-mssql zabbix-agent2-plugin-postgresql
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  zabbix-agent2-plugin-mongodb zabbix-agent2-plugin-mssql zabbix-agent2-plugin-postgresql
0 upgraded, 3 newly installed, 0 to remove and 1 not upgraded.
Need to get 8,667 kB of archives.
After this operation, 29.4 MB of additional disk space will be used.
Get:1 https://repo.zabbix.com/zabbix/7.2/stable/ubuntu noble/main amd64 zabbix-agent2-plugin-mongodb amd64 1:7.2.7-1+ubuntu24.04 [3,435 kB]
Get:2 https://repo.zabbix.com/zabbix/7.2/stable/ubuntu noble/main amd64 zabbix-agent2-plugin-mssql amd64 1:7.2.7-1+ubuntu24.04 [2,414 kB]
Get:3 https://repo.zabbix.com/zabbix/7.2/stable/ubuntu noble/main amd64 zabbix-agent2-plugin-postgresql amd64 1:7.2.7-1+ubuntu24.04 [2,818 kB]
Fetched 8,667 kB in 22s (396 kB/s)
Selecting previously unselected package zabbix-agent2-plugin-mongodb.
(Reading database ... 159280 files and directories currently installed.)
Preparing to unpack .../zabbix-agent2-plugin-mongodb_1k3a7.2.7-1+ubuntu24.04_amd64.deb ...
Unpacking zabbix-agent2-plugin-mongodb (1:7.2.7-1+ubuntu24.04) ...
Selecting previously unselected package zabbix-agent2-plugin-mssql.
Preparing to unpack .../zabbix-agent2-plugin-mssql_1k3a7.2.7-1+ubuntu24.04_amd64.deb ...
Unpacking zabbix-agent2-plugin-mssql (1:7.2.7-1+ubuntu24.04) ...
Selecting previously unselected package zabbix-agent2-plugin-postgresql.
Preparing to unpack .../zabbix-agent2-plugin-postgresql_1k3a7.2.7-1+ubuntu24.04_amd64.deb ...
Unpacking zabbix-agent2-plugin-postgresql (1:7.2.7-1+ubuntu24.04) ...
Setting up zabbix-agent2-plugin-postgresql (1:7.2.7-1+ubuntu24.04) ...
Setting up zabbix-agent2-plugin-mssql (1:7.2.7-1+ubuntu24.04) ...
Setting up zabbix-agent2-plugin-mongodb (1:7.2.7-1+ubuntu24.04) ...
```

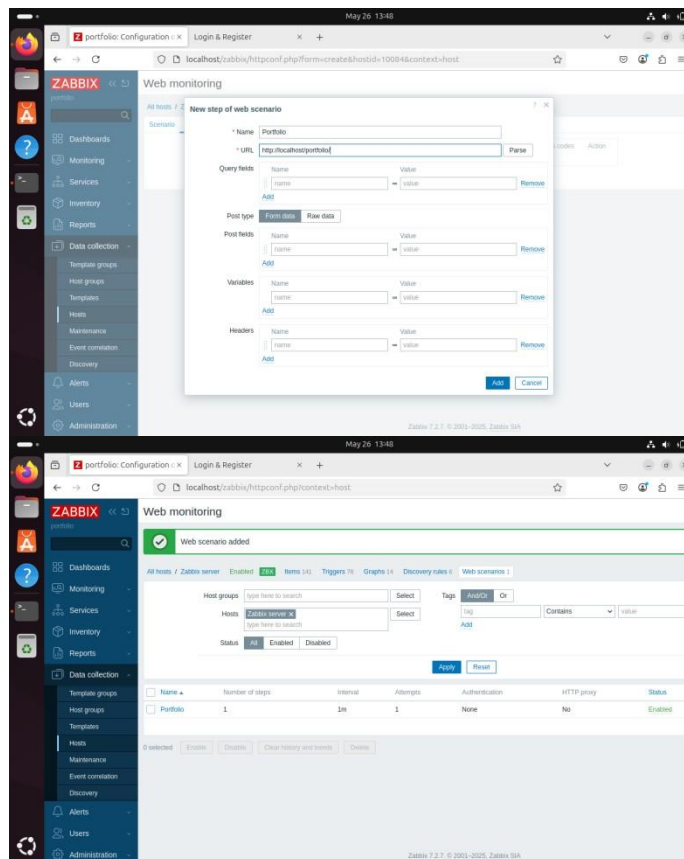
Zabbix Monitoring Dashboard

This screenshot displays the Zabbix dashboard monitoring the web server. It visualizes important performance metrics and the status of services to ensure the website remains available and responsive.



Linking Zabbix Agent to Monitor the Website Server

This screenshot shows the configuration of the Zabbix agent on the web server. The agent is responsible for collecting server performance data and sending it to the Zabbix monitoring server, enabling continuous monitoring of the website's health.

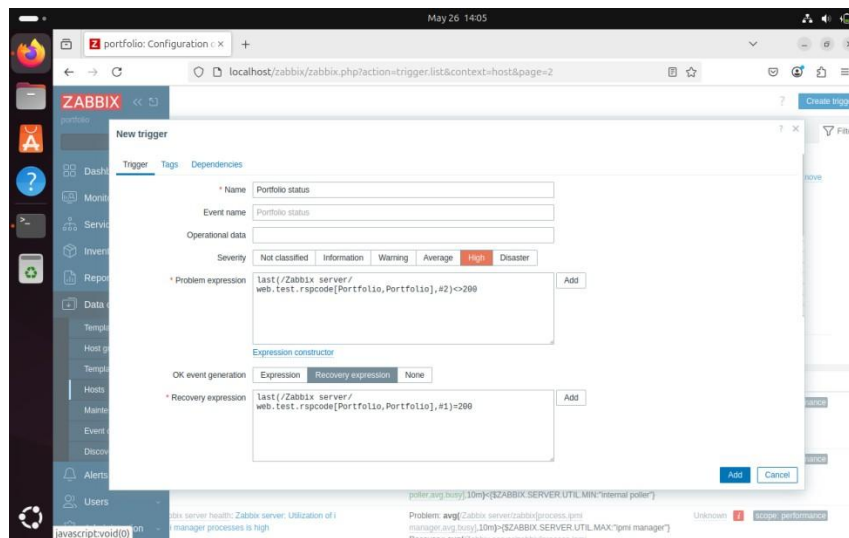


Creating a Trigger to Monitor Website Uptime

This screenshot shows the setup of a trigger in Zabbix to monitor the website's HTTP response status.

- The trigger checks if the website returns a **200 OK** status, indicating the site is up and functioning correctly.
- If the response changes to **404 Not Found** or any other error status, the trigger activates an alert.

This helps to quickly detect when the website is down or unreachable, ensuring prompt action can be taken to restore service.



Trigger for CPU Load and RAM Usage

This screenshot shows the configuration of a combined trigger monitoring both CPU load and RAM usage on the server.

- The trigger monitors the **average system load** and **memory consumption** to detect when the server is under heavy resource usage.
- If the CPU load or RAM usage exceeds the defined thresholds (e.g., CPU load above 80%, RAM usage above 75%), the trigger activates an alert.
- This combined monitoring helps ensure the server remains responsive and prevents performance degradation due to resource exhaustion

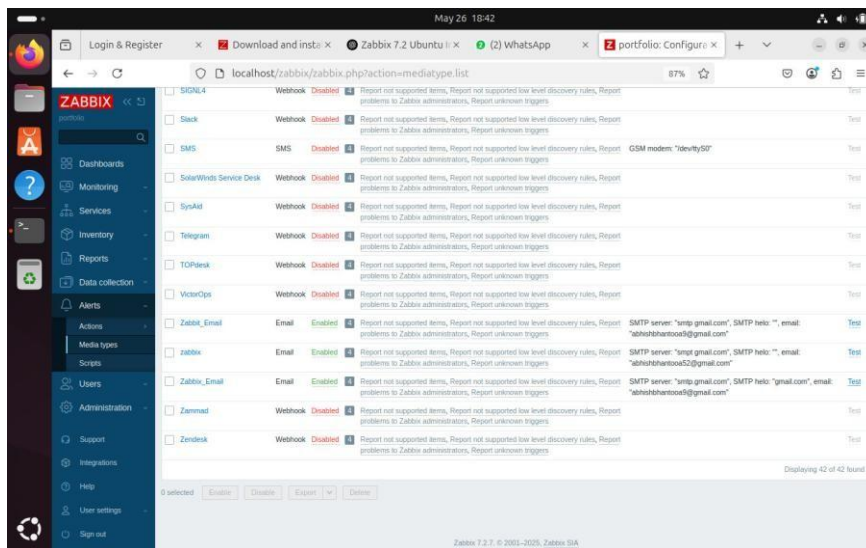
.....

Setting Up Email Alerts for Critical Events

To ensure timely responses to critical server and website issues, email alerts were configured in Zabbix for the following conditions:

- **Website Downtime:** An email notification is sent whenever the website trigger detects downtime, such as receiving a 404 error or no response, enabling immediate investigation and resolution.
- **High CPU/Memory/Disk Usage:** Alerts are triggered and emailed when server resources exceed predefined thresholds (e.g., CPU load over 80%, memory usage above 75%, or disk usage reaching critical levels), helping to prevent performance degradation or system failure.

This alert system allows administrators to proactively monitor server health and maintain website availability by responding quickly to issues as they arise.



Task 5: Website Security Testing – SQL Injection & Forced Browsing

To verify the security of my deployed website, I performed basic **attack simulations** such as **SQL injection** and **forced browsing**. These tests were unsuccessful, confirming that the website had some level of defense mechanisms in place.

Step 1: SQL Injection Protection via Prepared Statements

In the login and registration process, I implemented **prepared statements** to prevent SQL injection.

```
// Use prepared statement to check if the email is already registered (prevents SQL injection)
$checkEmail = $conn->prepare("SELECT email FROM users WHERE email = ?");
$checkEmail->bind_param('s', $email); // Bind email as string parameter
$checkEmail->execute(); // Execute the query
$checkEmail->store_result(); // Store result to check number of rows returned
```

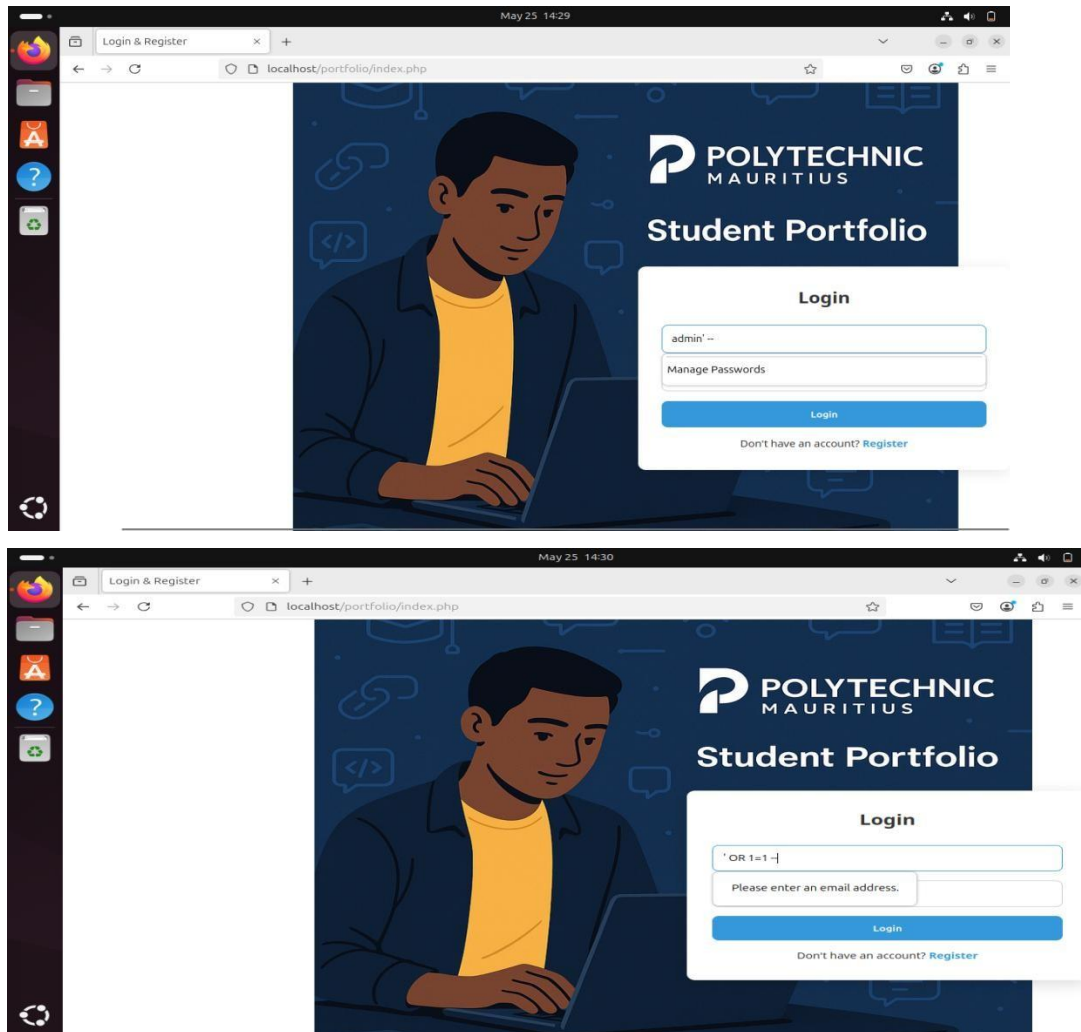
```
// Use prepared statement to fetch user by email (prevents SQL injection)
$result = $conn->prepare("SELECT * FROM users WHERE email = ?");
$result->bind_param('s', $email); // Bind email as string parameter
$result->execute(); // Execute the query
$result = $result->get_result(); // Get result set from executed statement
```

Explanation:

- **Prepared statements** prevent direct injection of SQL code into the query.
- Inputs are **bound** separately from the SQL logic, protecting against attackers trying to bypass authentication or manipulate the database using strings like admin' OR '1'=1.

Step 2: Simulated SQL Injection Attack (Unsuccessful)

I attempted a SQL injection through the login form using inputs like:



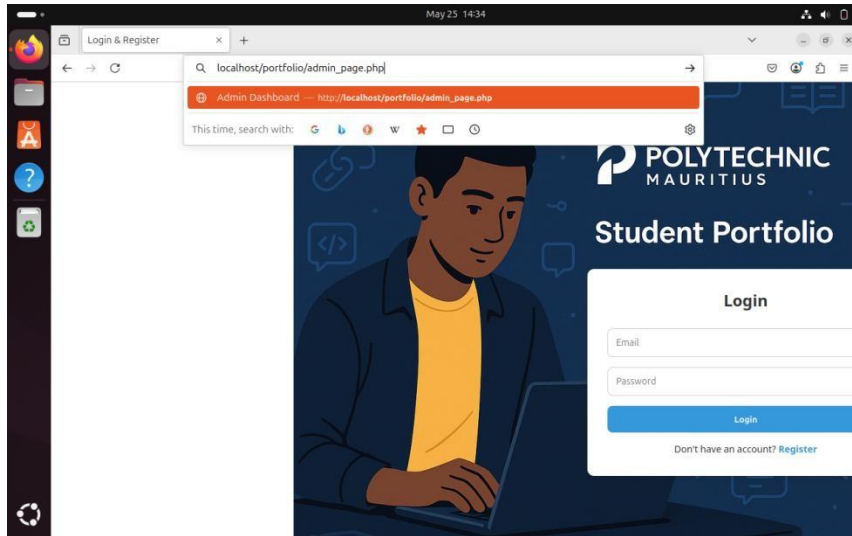
Result: Login failed – due to secure query execution using prepared statements.

Figure: Screenshot showing failed SQL injection login attempt.

Step 3: Forced Browsing Protection (Unsuccessful)

I also tested **forced browsing** by trying to access admin pages directly without logging in.

Example URL tested:



Result: Access denied – the page redirected or blocked access due to lack of session validation.

Figure: Screenshot showing the blocked forced browsing attempt.

Summary of Protection Features

Feature	Status	Protection Type
SQL Injection	Blocked ✓	Prepared Statements (PHP)
Forced Browsing	Blocked ✓	Session-based Access Control

These implementations show that the website has basic but essential security controls against common web attacks.

Task 6: Accessing the Website from Another Virtual Machine

After configuring the web server and firewall rules, I tested external access to ensure that the website could be accessed securely from a **different VM within the same network**.

Step 1: Check Apache Service and Firewall Rules Before

testing access from another machine, I made sure:

- Apache service was running:


```
May 25 14:57
abhis@ubuntu:~
Processing triggers for libc-bin (2.39-0ubuntu8.4) ...
Processing triggers for ufw (0.36.2-6) ...
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for php8.3-cli (8.3.6-0ubuntu0.24.04.4) ...
Processing triggers for libapache2-mod-php8.3 (8.3.6-0ubuntu0.24.04.4) ...
abhis@ubuntu:~$ sudo apt install apache2 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apache2 is already the newest version (2.4.58-1ubuntu8.6).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
abhis@ubuntu:~$ sudo systemctl start apache2
Synchronizing state of apache2.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable apache2
abhis@ubuntu:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset:
   Active: active (running) since Sun 2025-05-25 10:52:51 UTC; 4min 48s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 18253 (apache2)
     Tasks: 6 (limit: 4608)
    Memory: 13.9M (peak: 14.6M)
       CPU: 276ms
    CGroup: /system.slice/apache2.service
            └─18253 /usr/sbin/apache2 -k start
              18256 /usr/sbin/apache2 -k start
              18257 /usr/sbin/apache2 -k start
              18258 /usr/sbin/apache2 -k start
              18259 /usr/sbin/apache2 -k start
              18260 /usr/sbin/apache2 -k start
```

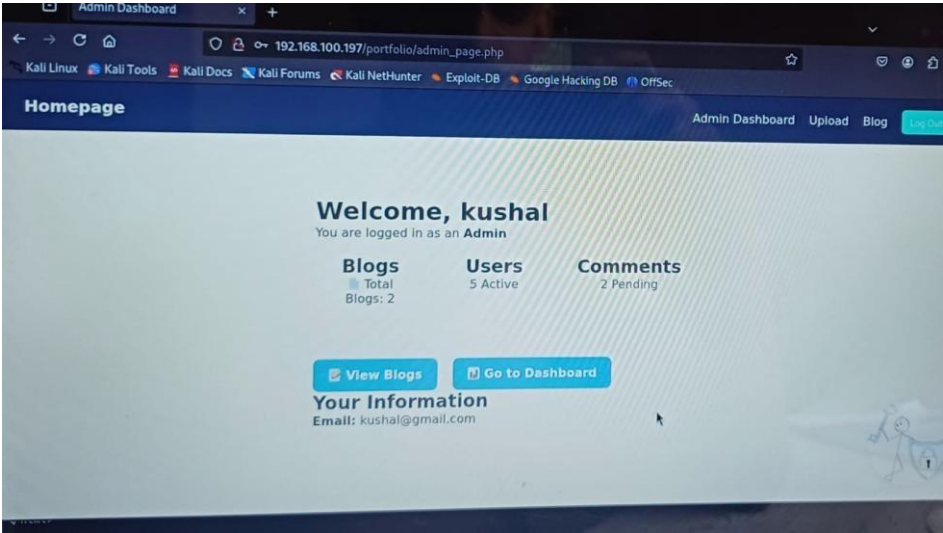
Step 2: Get the Web Server’s IP Address

On the hosting VM (Ubuntu server), I retrieved the internal IP address using:

.....

Step 3: Access the Website from Another VM

On a second VM (client), I opened a browser and typed the server IP:



Result: The website loaded successfully, showing that the firewall, Apache configuration, and network settings were correctly applied.

Figure: Screenshot showing the website opened from a second VM.

Summary

Test	Result	Notes
Website accessible via HTTP	✓ Success	Port 80 was open and Apache was running
Firewall configuration	✓ Pass	Only allowed specific traffic (e.g., HTTP, SSH)

Conclusion

Throughout this project, I successfully deployed and secured a web server on Ubuntu by following essential best practices in secure website deployment, access control, firewall configuration, and monitoring.

📌 Key Achievements:

- ✓ Installed and configured Apache to host a portfolio website.
- ✓ Properly set file permissions and ownership to restrict unauthorized access.
- ✓ Implemented **Role-Based Access Control (RBAC)**, allowing users with different roles (admin/user) to access different pages securely.
- ✓ Hardened the server using **iptables**, blocking all unnecessary traffic and allowing only essential ports such as 80 (HTTP) and 22 (SSH).
- ✓ Successfully simulated and **prevented SQL injection and forced browsing** using prepared statements and access checks.
- ✓ Confirmed secure and functional website access from another virtual machine in the network.
- ✓ Monitored server status using **Zabbix**, ensuring ongoing visibility into system health and uptime.

This hands-on deployment reinforced my understanding of secure web hosting, Linux administration, and real-world cybersecurity defenses. It also demonstrated how practical configurations, such as proper user authentication, access restrictions, and firewall rules, can significantly improve the overall security posture of a system.