Web Server – Apache Configuration

A server that deliver contents over the **Hyper Text Transfer Protocol (HTTP)** is called **Web Server**. When you are browsing a website through any web browser, you are communicating with the web server which means you are requesting for the web pages that the server is currently hosting. Many organizations make use of websites to deliver or gather information in an efficient way.

Apache is a widely used open source HTTP server software. The software provides numerous benefits thus, dominating the web server market in terms of use and deployments. The apache is:

- Stable, flexible and secure.
- Used, backed and supported by several major sites and organizations.
- Open source – the entire program and related components.
- Supported on most variants of Linux/UNIX and Microsoft Windows.

In this unit we will install, configure and test the functionality of the apache Web Server in our virtual network.

Understanding HTTP:

Most of the world's Internet traffic is HTTP. Apache is the server implementation of HTTP whereas applications such as Firefox, Chrome, Opera, etc. are client implementation of HTTP. Today, HTTP 1.1 is the most widely used but HTTP 2 is also already implemented.

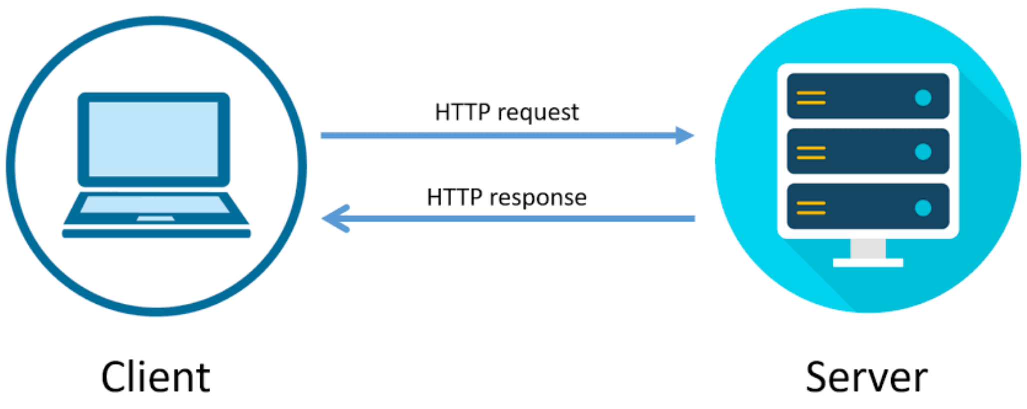
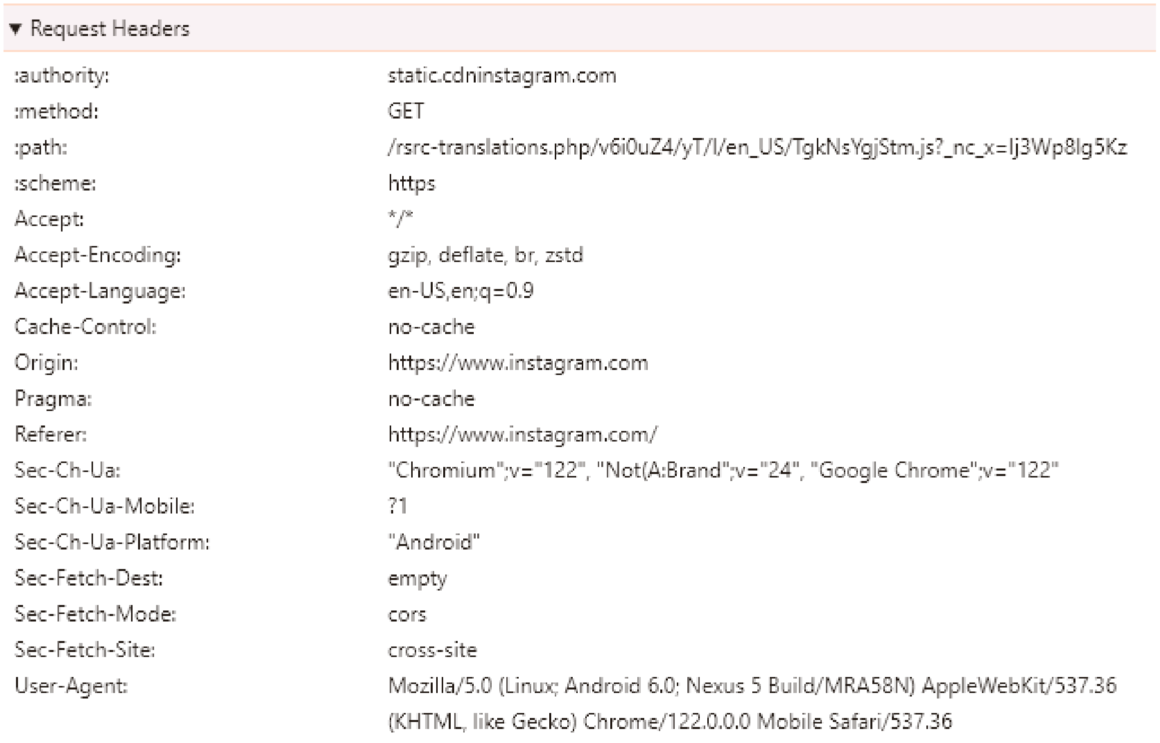


Figure 1.0: HTTP request and response

Headers:

Normally, when a client connects to the web server, the client contacts the server's TCP port 80. Once the connection is made, the web server says nothing unless the client issues HTTP-complaint commands (also called *verbs* or *methods*) for its request to the server. With each command comes a request header. It contains information about the client. The HTTP request header will look like the figure given below:



▼ Request Headers

:authority:	static.cdninstagram.com
:method:	GET
:path:	/rsrc-translations.php/v6i0uZ4/yT/l/en_US/TgkNsYgjStm.js?_nc_x=Ij3Wp8lg5Kz
:scheme:	https
Accept:	*/*
Accept-Encoding:	gzip, deflate, br, zstd
Accept-Language:	en-US,en;q=0.9
Cache-Control:	no-cache
Origin:	https://www.instagram.com
Pragma:	no-cache
Referer:	https://www.instagram.com/
Sec-Ch-Ua:	"Chromium";v="122", "Not(A:Brand";v="24", "Google Chrome";v="122"
Sec-Ch-Ua-Mobile:	?1
Sec-Ch-Ua-Platform:	"Android"
Sec-Fetch-Dest:	empty
Sec-Fetch-Mode:	cors
Sec-Fetch-Site:	cross-site
User-Agent:	Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 Mobile Safari/537.36

Figure 1.1: The HTTP request header

Note: Every HTTP request has a set of mandatory and optional headers. Following are some of the common headers and brief description. The value in each header is as an example.

Accept *– The media type/types acceptable*

Accept-Encoding *– List of acceptable encodings*

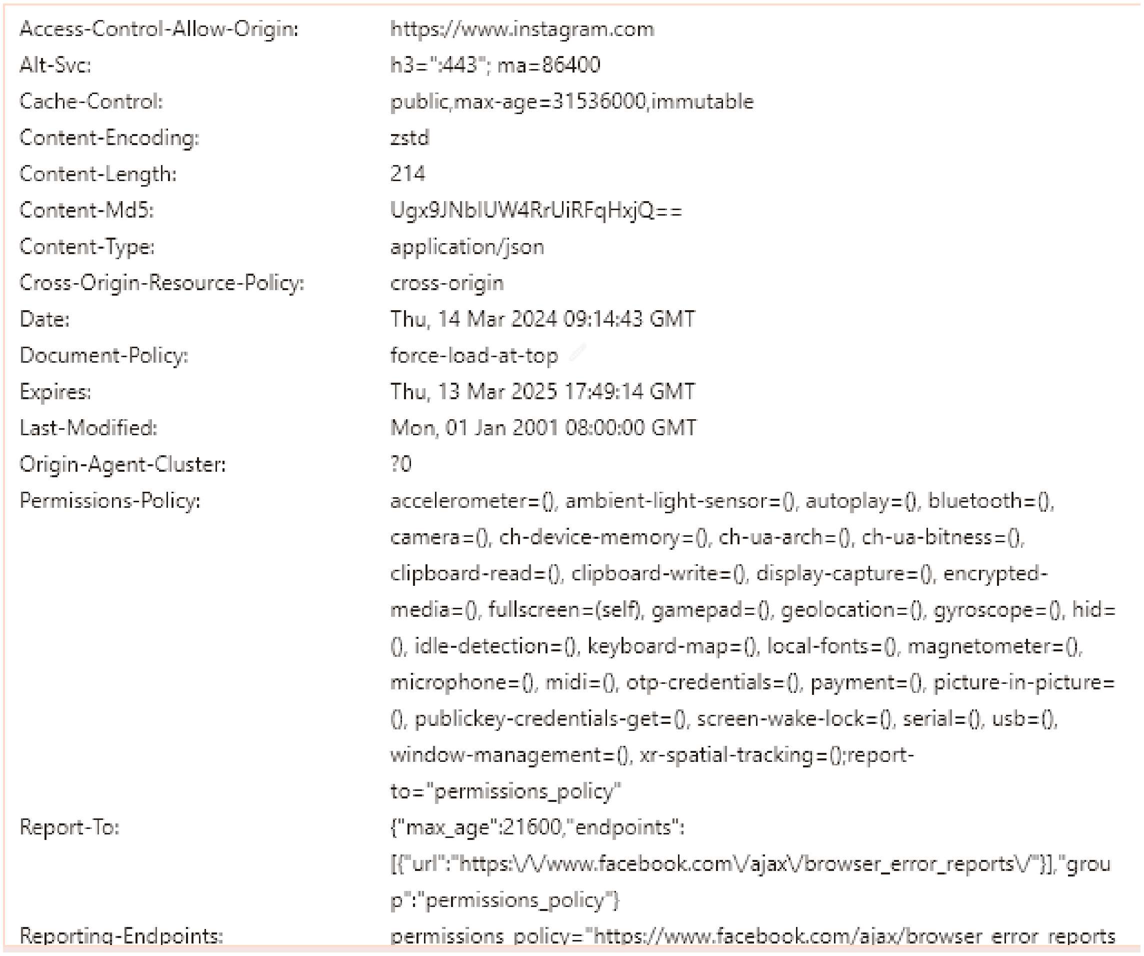
Accept-Language *– List of acceptable languages*

Referer *– The address of the previous web page from which a link to the currently requested page was followed*

User-Agent *– The string that identifies the user agent.*

The second line in the figure above shows that the method is GET, which asks the servers to fetch a file. The remaining information makes up the header, which tells the server about the client, the kind of file formats the client will accept, and so on.

Once the server knows a request is complete, it responds with the actual requested content, prefixed by a server header. It contains information about the server, the amount of data the client is about to receive, the type of data coming in, and other information. The response header from the above request is shown below:



```
Access-Control-Allow-Origin:      https://www.instagram.com
Alt-Svc:                          h3=":443"; ma=86400
Cache-Control:                    public,max-age=31536000,immutable
Content-Encoding:                 zstd
Content-Length:                   214
Content-Md5:                      Ugx9JNbIUW4RrUiRFqHxjQ==
Content-Type:                     application/json
Cross-Origin-Resource-Policy:     cross-origin
Date:                             Thu, 14 Mar 2024 09:14:43 GMT
Document-Policy:                  force-load-at-top
Expires:                          Thu, 13 Mar 2025 17:49:14 GMT
Last-Modified:                    Mon, 01 Jan 2001 08:00:00 GMT
Origin-Agent-Cluster:             ?0
Permissions-Policy:               accelerometer=(), ambient-light-sensor=(), autoplay=(), bluetooth=(),
                                  camera=(), ch-device-memory=(), ch-ua-arch=(), ch-ua-bitness=(),
                                  clipboard-read=(), clipboard-write=(), display-capture=(), encrypted-
                                  media=(), fullscreen=(self), gamepad=(), geolocation=(), gyroscope=(), hid=
                                  (), idle-detection=(), keyboard-map=(), local-fonts=(), magnetometer=(),
                                  microphone=(), midi=(), otp-credentials=(), payment=(), picture-in-picture=
                                  (), publickey-credentials-get=(), screen-wake-lock=(), serial=(), usb=(),
                                  window-management=(), xr-spatial-tracking=();report-
                                  to="permissions_policy"
Report-To:                        {"max_age":21600,"endpoints":
                                  [{"url":"https:\/\/www.facebook.com\/ajax\/browser_error_reports\/"}],"grou
                                  p":"permissions_policy"}
Reporting-Endpoints:              permissions_policy="https://www.facebook.com/ajax/browser_error_reports
```

Figure 1.2: The HTTP response header

Note: Some of the common HTTP response headers are given below:

Content-Encoding *– The type of encoding used on the data*

Content-Length *– The length of the response body expressed in 8-bit bytes*

Date *– The date and time that the message was sent*

Last-Modified *– The last modified date for the requested object*

HTTP Methods:

There are so many HTTP requests out of which some of them are as follows:

GET: Retrieves data from the server

POST: Submit data to the server

PUT: Update data already on the server

DELETE: Deletes data from the server

HTTP Status Codes:

The HTTP status code is a response made by the server to the client’s request. These are three-digit codes, while there are more than 60 error status codes, but some of the common ones are:

Status codes with their meanings:

Code Range	Meaning
1xx	Informational Response (They are all about the information received by the server when a request is made)
2xx	Success (They depict that the request made has been fulfilled by the server and the expected response has been achieved)
3xx	Redirection (The requested URL is redirected elsewhere)
4xx	Client Errors (This indicates that the page is not found)
5xx	Server Errors (A request made by the client but the server fails to complete the request)

Activity-01:

List at least 20 HTTP status codes with their meaning and brief description. Submit the PDF file in the VLE.

Ports:

Default port for HTTP requests is port 80 but you can also configure a web server to use a different port that is not used by another service. This is useful for running multiple web servers or sites on the same host with each server or site on a different port.

When a site runs a web server on a nonstandard port, port number will be visible in the site's URL. For example, the web address www.example.com with the default port number (80) implicitly and explicitly displayed would read http://www.example.com and http://www.example.com:80, respectively. But serving the same site on a nonstandard port such as port 8080 will require the port number to be explicitly stated as in http://www.example.com:8080.

Network Details:

We will now make use of our network to implement the Apache Web Server. The details of the Web Server is given below:

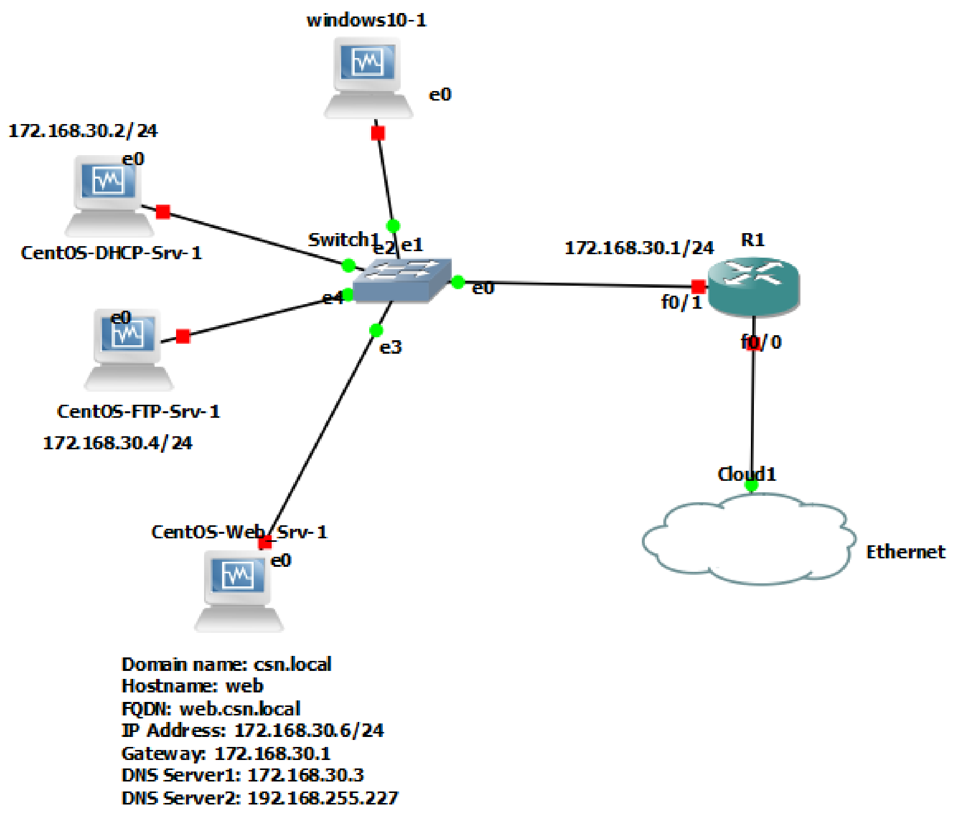


Figure 1.3: The network

Note: If the system slows down when running multiple virtual machines, you can stop the DHCP Server and give static IP address to the client devices to be in the same network as other servers or either you can configure DHCP on the router so that it assigns IP address to the client devices.

Firewall:

The Apache service uses port **80** so, we will open this port and port **22** too to later connect to the server to upload some files. We will enable the **http** and **ssh** service as well.

```
# firewall-cmd --permanent --zone=public --add-service=http
# firewall-cmd --permanent --zone=public --add-service=ssh
# firewall-cmd --permanent --zone=public --add-port=80/tcp
# firewall-cmd --permanent --zone=public --add-port=22/tcp
# firewall-cmd --reload
```

Installing Packages:

Before we configure the server with static IP address, we will connect the system to the internet to download the packages as web service needs several packages. First on our **Linux**: the operating system (server), we will install **Apache:** the web server, **MySQL/MariaDB:** the databse server and **PHP:** the programming language which collectively is known as **LAMP** stack. LAMP stack is a bundle of four different software technologies that are used to build websites and web applications.

Install Apache Web Server Package:

On most linux distributions, the Apache HTTP server software is packaged as RPM, .deb and other binaries, so installing the software is as simple as using package management tool on the system. On Fedora/RHEL/CentOS systems, the package that provides the Apache HTTP server is named **httpd**. For this session, as before we will use YUM utility to install the package.

```
# yum install -y httpd
```

Note: If the above command gives error message and cannot download the package, it is because CentOS met its EOL (End Of Life), thus it no longer receives development resources from the official CentOS project.

*We use the **sed** command to edit the required directives or parameters in the repo configuration files:*

```
# sed -i 's/mirrorlist/#mirrorlist/g' /etc/yum.repos.d/CentOS-*
# sed -i 's|#baseurl=http://mirror.centos.org|baseurl=http://vault.centos.org|g'
/etc/yum.repos.d/CentOS-*
```

Now you should be able to update CentOS or install packages without any error.

Installing MariaDB:

CentOS has switched from MySQL to MariaDB as its default database management system. Use the following command to install MariaDB:

```
# yum install -y mariadb mariadb-server
```

Installing PHP:

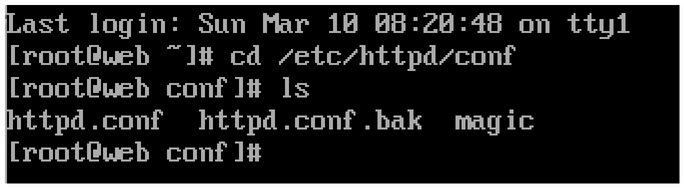
PHP is responsible for processing code to display dynamic content. It runs scripts, connects to our MySQL/MariaDB database to get information and hand the processed content over to the web server to display. Use following command to install PHP and other dependencies:

# yum install php -y*

Configuring the Web Server:

As with many other services, the main configuration file of the web server is located on /etc directory - **/etc/httpd/conf/** directory. Before making changes to the main configuration file, it is always advised to keep the backup of that file in case we mess things up. There are numerous methods to back up a file but we will use **copy** command to do this.

# cp /etc/httpd/conf/httpd.conf /etc/httpd/conf/httpd.conf.bak



```
Last login: Sun Mar 10 08:20:48 on tty1
[root@web ~]# cd /etc/httpd/conf
[root@web conf]# ls
httpd.conf  httpd.conf.bak  magic
[root@web conf]#
```

Figure 1.4: The backup file

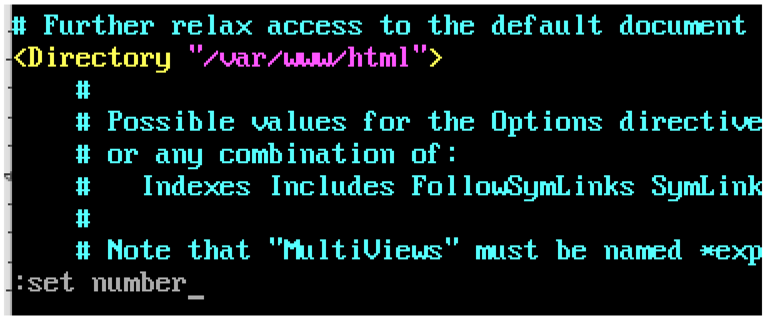
As everything in Linux is considered a file, the configuration files are no exception. Open the file using any text editor and make changes as desired.

# vim /etc/httpd/conf/httpd.conf

After the file is opened, make changes accordingly as indicated in the following:

Note: Before we make changes to the file, we will open the file along with line numbers showing on the left side. To achieve this open the file with vim and in escape mode type the following:

:set number



```
# Further relax access to the default document
<Directory "/var/www/html">
    #
    # Possible values for the Options directive
    # or any combination of:
    #   Indexes Includes FollowSymLinks SymLink
    #
    # Note that "MultiViews" must be named *exp
:set number_
```

Figure 1.5: Setting the line numbers

In the line number 89, change the administrator’s email address as:

ServerAdmin root@csn.local

Uncomment the line number 98 and change the server name according to the domain we have chosen:

ServerName www.csn.local:80

In the line 122, we have to give the absolute path to the directory from where the web server will serve the documents:

DocumentRoot “/var/www/html/hostelmanagement”

In the line 167, add filename as below. This directive sets the list of resources to look for when the client requests an index of the directory.

DirectoryIndex index.html index.cgi index.php

Finally for this server, add the following to the end of the file to hide the server information from being added to the http header (server’s response header).

ServerTokens Prod

Note: The ServerTokens directive controls whether Server response header field which is sent back to clients includes a description of the generic OS-type of the server as well as information about complied-in modules.
ServerTokens Full – Server sends (e.g): Server: Apache/2.4.2 (Unix) PHP/4.2.2 MyMod/1.2
ServerTokens Prod [uctOnly] – Server sends (e.g): Server: Apache

ServerTokens Major – Server sends (e.g): Server: Apache/2

ServerTokens Minor – Server sends (e.g): Server: Apache/2.4

ServerTokens Min [imal] – Server sends (e.g): Server: Apache/2.4.2

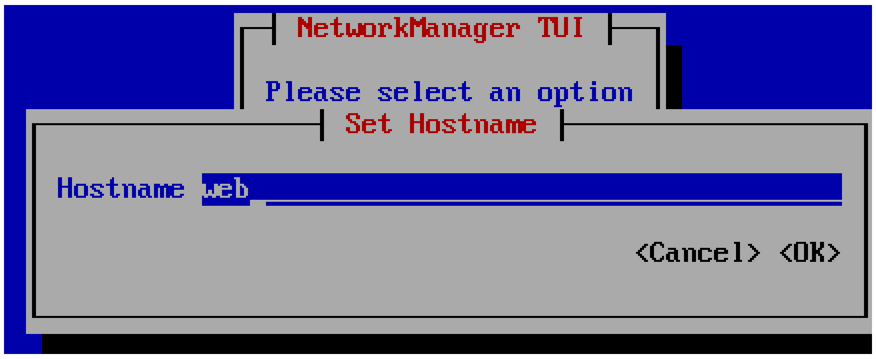
ServerTokens OS – Server sends (e.g): Server: Apache/2.4.2 (Unix)

And add the following to allow the same TCP connection for HTTP conversation instead of opening a new one with each new request.

KeepAlive On

Assigning Static IP Address to the Server:

Now that we have done with the installation and configuration of the Web Server, we will assign IP address to the server so that it is in our same virtual network. Use **nmtui** to do this action.



```
Hostname web
```

Figure 1.6: The hostname

If your interface is not activated, it is time to activate it. When you select the interface and it is saying '**Deactivate**' as shown below, it means the interface is activated.

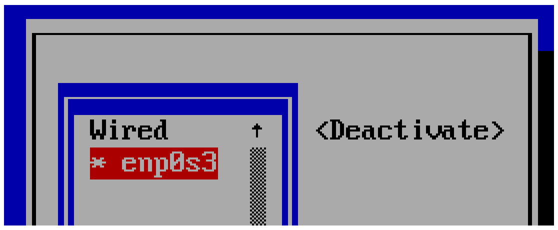
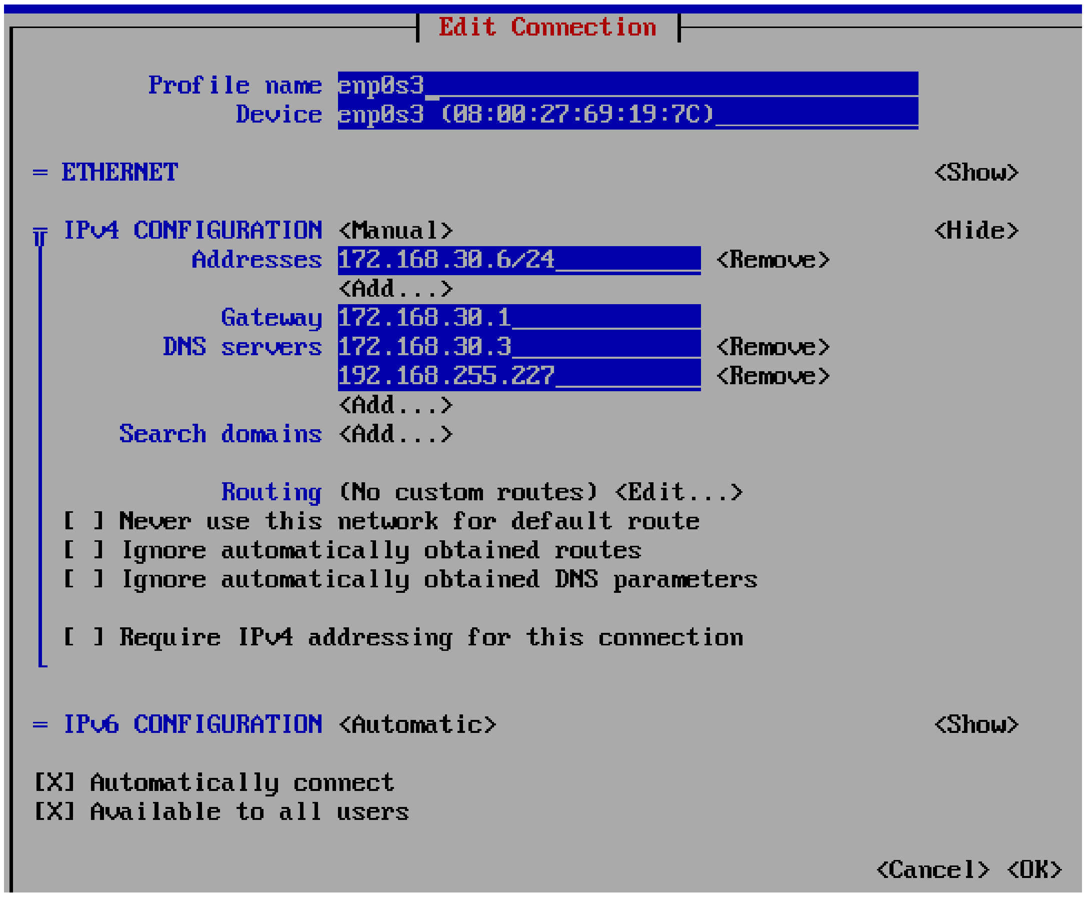


Figure 1.7: Activated interface

Now, configure the IP Address details of the server and make sure to tick the 'Automatically connect' button as shown below:


```
                        Edit Connection

        Profile name enp0s3_
              Device enp0s3 (08:00:27:69:19:7C)

= ETHERNET                                                     <Show>

╤ IPv4 CONFIGURATION <Manual>                                  <Hide>
│          Addresses 172.168.30.6/24_______ <Remove>
│                    <Add...>
│            Gateway 172.168.30.1__________
│        DNS servers 172.168.30.3__________ <Remove>
│                    192.168.255.227_______ <Remove>
│                    <Add...>
│     Search domains <Add...>
│
│            Routing (No custom routes) <Edit...>
│ [ ] Never use this network for default route
│ [ ] Ignore automatically obtained routes
│ [ ] Ignore automatically obtained DNS parameters
│
│ [ ] Require IPv4 addressing for this connection
└

= IPv6 CONFIGURATION <Automatic>                               <Show>

[X] Automatically connect
[X] Available to all users

                                                     <Cancel> <OK>
```

Figure 1.8: IP Address details

Once the IP address to the Web server is assigned, check the connectivity by sending ping request to the router or any other devices on the network:

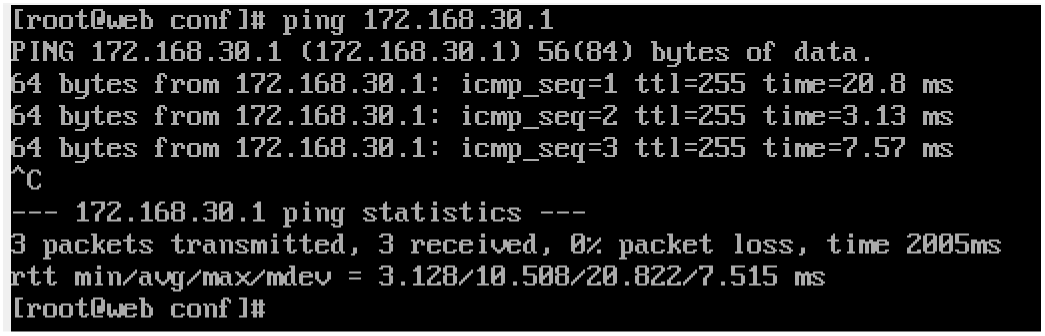
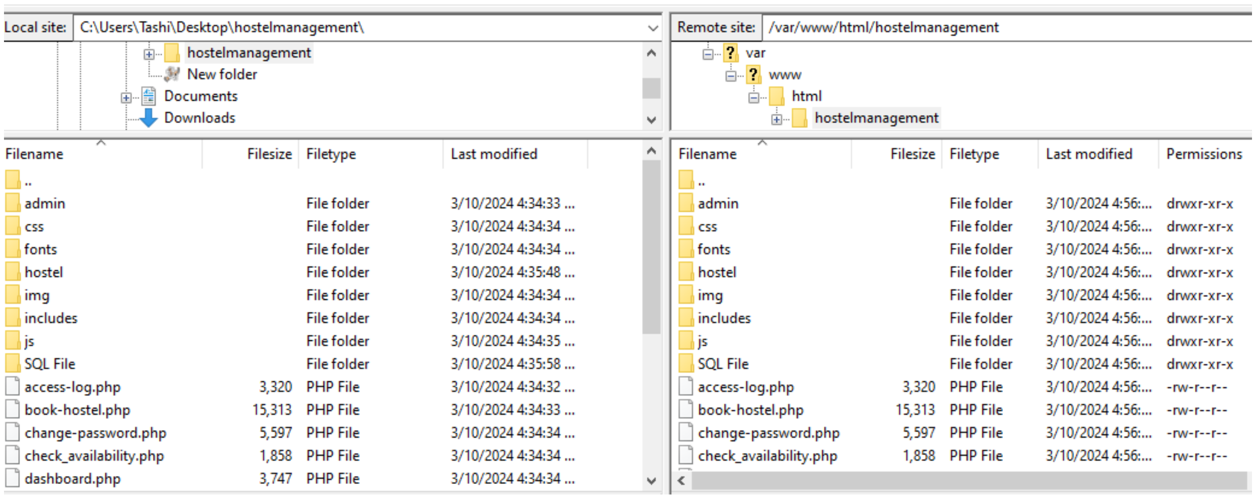

```
[root@web conf]# ping 172.168.30.1
PING 172.168.30.1 (172.168.30.1) 56(84) bytes of data.
64 bytes from 172.168.30.1: icmp_seq=1 ttl=255 time=20.8 ms
64 bytes from 172.168.30.1: icmp_seq=2 ttl=255 time=3.13 ms
64 bytes from 172.168.30.1: icmp_seq=3 ttl=255 time=7.57 ms
^C
--- 172.168.30.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 3.128/10.508/20.822/7.515 ms
[root@web conf]#
```

Figure 1.9: Verifying connectivity to the router

Importing the Web Application:

For this web server, we will host a web application developed using PHP. To achieve this, we have to copy the Web Application files to the /var/ww/html directory (DocumentRoot) to the Web Server. We will use FileZilla from Windows client to upload these files to the server. Make sure that the **hostelmanagement** folder that I have provided is in your Windows client machine.

Connect using FileZilla through **ssh** to the Web Server and upload these folder to the DocumentRoot directory. You can create a folder named **hostelmanagement** in the **/var/www/html** directory as shown below and upload the files in this new directory.



Local site: C:\Users\Tashi\Desktop\hostelmanagement\

```
hostelmanagement
New folder
Documents
Downloads
```

Filename	Filesize	Filetype	Last modified
..			
admin		File folder	3/10/2024 4:34:33 ...
css		File folder	3/10/2024 4:34:34 ...
fonts		File folder	3/10/2024 4:34:34 ...
hostel		File folder	3/10/2024 4:35:48 ...
img		File folder	3/10/2024 4:34:34 ...
includes		File folder	3/10/2024 4:34:34 ...
js		File folder	3/10/2024 4:34:35 ...
SQL File		File folder	3/10/2024 4:35:58 ...
access-log.php	3,320	PHP File	3/10/2024 4:34:32 ...
book-hostel.php	15,313	PHP File	3/10/2024 4:34:33 ...
change-password.php	5,597	PHP File	3/10/2024 4:34:34 ...
check_availability.php	1,858	PHP File	3/10/2024 4:34:34 ...
dashboard.php	3,747	PHP File	3/10/2024 4:34:34 ...

Remote site: /var/www/html/hostelmanagement

```
var
  www
    html
      hostelmanagement
```

Filename	Filesize	Filetype	Last modified	Permissions
..				
admin		File folder	3/10/2024 4:56:...	drwxr-xr-x
css		File folder	3/10/2024 4:56:...	drwxr-xr-x
fonts		File folder	3/10/2024 4:56:...	drwxr-xr-x
hostel		File folder	3/10/2024 4:56:...	drwxr-xr-x
img		File folder	3/10/2024 4:56:...	drwxr-xr-x
includes		File folder	3/10/2024 4:56:...	drwxr-xr-x
js		File folder	3/10/2024 4:56:...	drwxr-xr-x
SQL File		File folder	3/10/2024 4:56:...	drwxr-xr-x
access-log.php	3,320	PHP File	3/10/2024 4:56:...	-rw-r--r--
book-hostel.php	15,313	PHP File	3/10/2024 4:56:...	-rw-r--r--
change-password.php	5,597	PHP File	3/10/2024 4:56:...	-rw-r--r--
check_availability.php	1,858	PHP File	3/10/2024 4:56:...	-rw-r--r--

Figure 1.10: Uploading the Web files to the server

Start and Enable Apache:

Now that we have copied the web files to the Web Server, we have to start the httpd service using following command:

```
# systemctl start httpd
# systemctl enable httpd
```

Start and Enable MariaDB:

As we have already installed MariaDB, we will now start it:

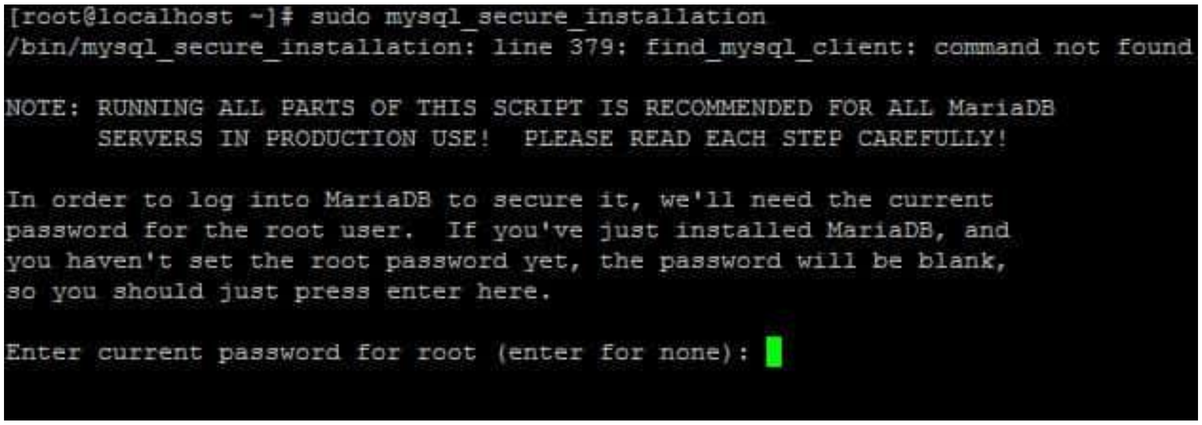
```
# systemctl start mariadb
# systemctl enable mariadb
```

Securing MySQL/MariaDB server:

The mysql_secure_installation script improves the security of the MySQL/MariaDB installation by setting a password for the root account, disabling root account from accessing the MySQL/MariaDB from outside the localhost, removing the anonymous user accounts and the test database which is accessible by all the users. Type the following command:

# mysql_secure_installation

When the script is first opened, the messages as shown below will appear asking for the root user's password. Since it is the first time opening this file, we do not have any password so, just press enter key.



```
[root@localhost ~]# sudo mysql_secure_installation
/bin/mysql_secure_installation: line 379: find_mysql_client: command not found

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
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
```

Figure 1.11: Logging in as root user

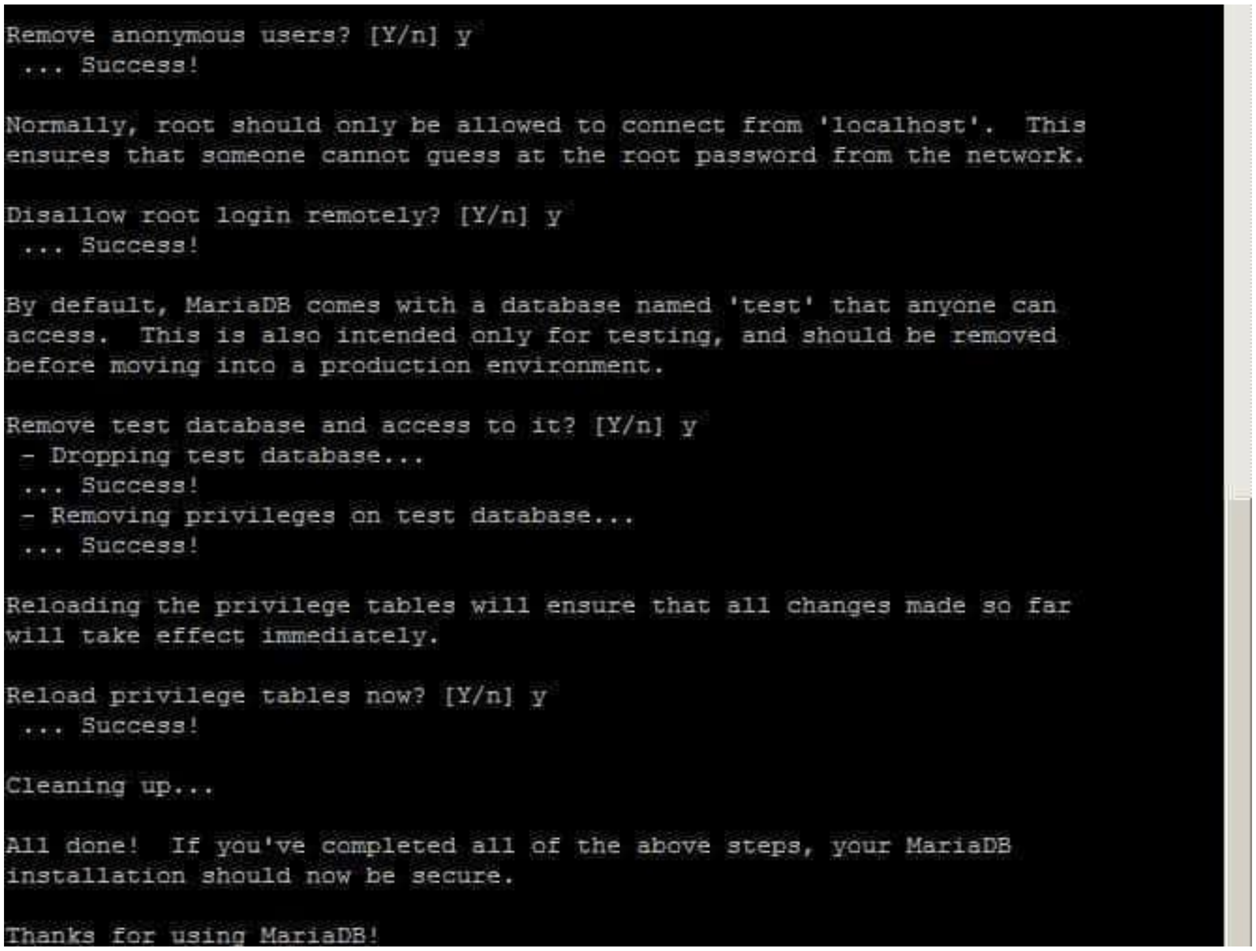
Next, it will ask whether to set the root user's password or not. We will set password so as to make it more secure. Type "Y" and enter.

We have now created a password for the root user. It will now ask whether or not to remove the anonymous users. Type "Y" and enter.

It will now ask whether or not to disallow remote log in of the root user other than the localhost. Type "Y" and continue.

The test database is created when installing the MariaDB package. We will remove the test database and continue.

Finally, we will type "Y" to reload the privilege table to take the changes we made.



```
Remove anonymous users? [Y/n] y
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
 ... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
 ... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
```

Figure 1.12: The complete configuration

Creating Database for the Web Server:

For us to store the web application data, we have to get the details of the database such as name of the database, database username and password, etc. which we can get from of the the web files.

In our sample application, the details are in the **config.php** file under **includes** folder.

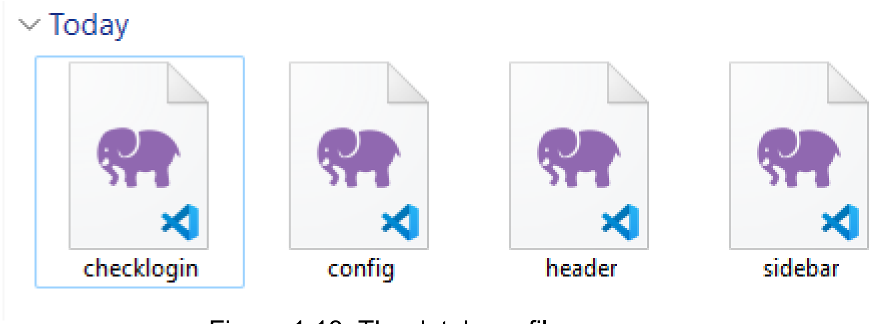
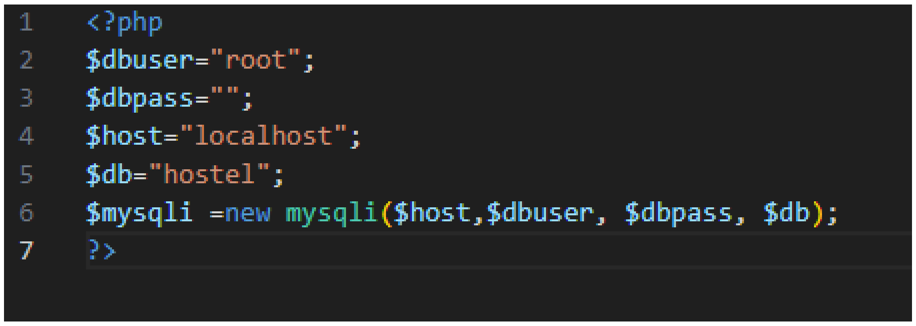


Figure 1.13: The database file

Open this file using any text editor and you will get the details we need.



```php
<?php
$dbuser="root";
$dbpass="";
$host="localhost";
$db="hostel";
$mysqli =new mysqli($host,$dbuser, $dbpass, $db);
?>
```

Figure 1.14: The database details

After opening the file, we know the following details:

Database name: hostel

Database username: root

Database user's password: [empty]

We have now done the secure installation and will proceed to login to MySQL/MariaDB by using the following command:

# mysql -u root –p

Type the MySQL root user's password that you have created during secure installation. We will now create the database for the Web Server as indicated in the following:

# CREATE DATABASE hostel;

#CREATE USER 'root'@'localhost' IDENTIFIED BY '';

Note: If the above command give error, instead to 'root', give any other user such as tashi, etc and give any password.

#GRANT ALL PRIVILEGES ON hostel. TO 'tashi'@'localhost';*

#exit

After creating the database and the database user with the passwords, we have to import the tables and rest of the required data from the database backup dump (.sql file). For this example we have a backup file for the sql provided with the file name **hostel.sql** in the folder. Otherwise, if there is not dump we have to start creating the database tables and other details (but normally the developers would hand you a copy of the dump to be directly imported to the database).

Before we import the database dump from the file, we have to restart the database server once to take the changes into effect if in case it is not.

```
# systemctl restart mariadb
```

To import use the following command where /var/www/html/schoolmanagement is the path to the sql dump file.

Syntax:

```
# mysql -u [DB_Server’s username] -p [database_name] < [full path to the .sql file]
```

```
# mysql -u root -p hostel < /var/www/html/hostelmanagement/hostel.sql
```

Configure PHP:

Since we already installed php, we do not have much to configure it. We will restart the Apache service:

```
# systemctl restart httpd
```

Testing the Web Application Server:

On our Linux operating system, we have installed and configured Apache, MySQL/MariaDB and PHP collectively known as LAMP stack. We have also added http and ssh services and opened port 80 and 22. **ssh** and port **22** was for uploading the web files to the server remotely. We have configured the firewall to allow traffic through http port 80. We now have running Web Server which is ready to serve the sample page we have created.

To test the functionlaity of the Web Server, open the client device and from any web browsers, try to access the web server by typing **http://172.168.30.6** in the URL field.

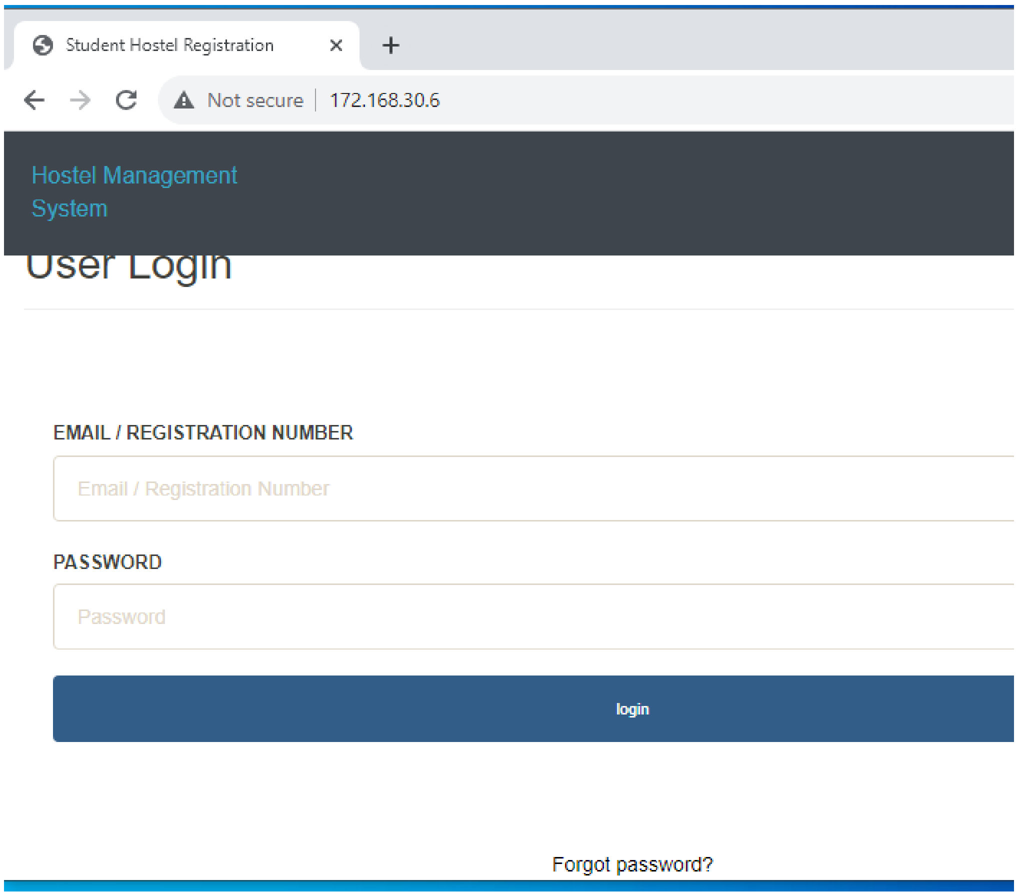


Figure 1.15: Accessing the Web application

We have now configured the Web Server successfully. If the page is not accessible, troubleshoot the other configurations.

Activity-02:

*Using your GNS virtual network and considering that the web server is up and running, inspect the HTTP request header and response header. After examining the headers, write some of the information that you can gather about the client and the server along with the snapshot. Submit the PDF file along with Activity-01 in the same file in VLE with the name – **Web-Srv-Activity**.*