

Task-1: Setting up an Apache Web Server

Step 1- Installing Apache: First, I updated the package repository and installed Apache2. To do this I used commands:

```
sudo apt update  
sudo apt install apache2
```

Step 2-Checking the Web Server Status: I verified that Apache was running properly. For this I used the command: sudo systemctl status apache2

The output showed that Apache was active and running successfully.

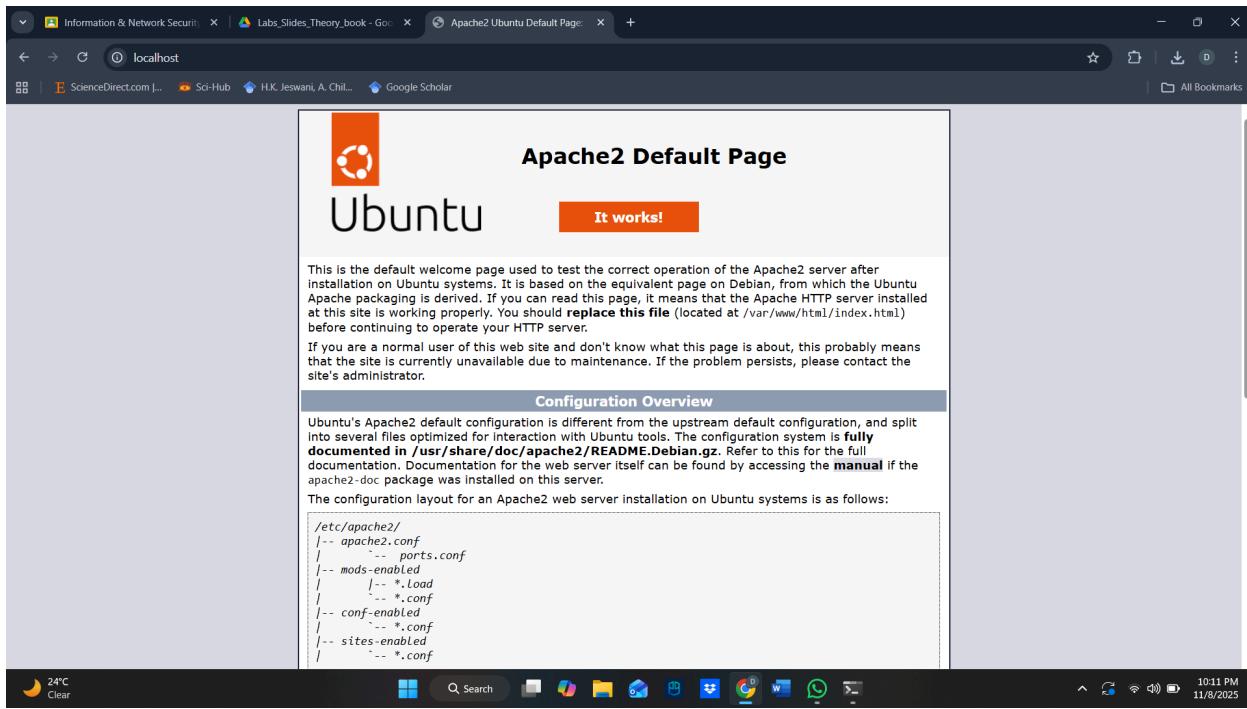
Step 3- Configuring Domain Name: To access the web server using a custom domain name, I edited the hosts file by: sudo nano /etc/hosts

And added the following entry: 127.0.0.1 webserverlab.com

Step 4-Testing Apache Installation: I accessed the Apache default page through the browser using:

http://localhost
http://127.0.0.1
http://webserverlab.com

The Apache2 Ubuntu Default Page was displayed successfully, confirming the installation. **The screenshot is given on the next page.**



Task-2: Setting Up Virtual Hosts

Part 1- Managing Apache Process

I learned the basic Apache management commands:

```
sudo systemctl stop apache2    # Stop the server
sudo systemctl start apache2   # Start the server
sudo systemctl restart apache2 # Restart the server
sudo systemctl reload apache2  # Reload configuration
sudo systemctl status apache2  # Check status
```

Part 2-Setting Up First Virtual Host (example.com)

Step 1: Creating Directory Structure

```
sudo mkdir -p /var/www/example.com/html
sudo chown -R $USER:$USER /var/www/example.com/html
sudo chmod -R 755 /var/www/example.com
```

Step 2: Creating Index Page

Created an HTML file at /var/www/example.com/html/index.html:

```
<html>
<head>
<title>Welcome to Example.com!</title>
</head>
<body>
<h1>Success! The example.com server block is working!</h1>
</body>
</html>
```

Step 3: Configuring Virtual Host

Created configuration file at /etc/apache2/sites-available/example.com.conf:

```
<VirtualHost *:80>
    ServerAdmin admin@example.com
    ServerName example.com
    ServerAlias www.example.com
    DocumentRoot /var/www/example.com/html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Step 4: Adding Domain to Hosts File

sudo nano /etc/hosts

Added 127.0.0.1 example.com

In the file.

Step 5: Enabling the Virtual Host

```
sudo a2ensite example.com.conf
```

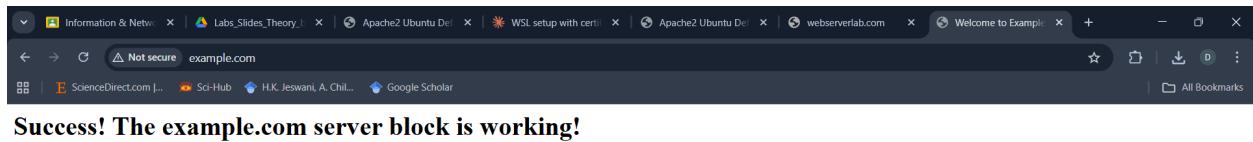
```
sudo a2dissite 000-default.conf
```

```
sudo apache2ctl configtest
```

```
sudo systemctl restart apache2
```

Step 6: Testing

Accessed <http://example.com> in the browser, which displayed: "Success! The example.com server block is working!" Here is the screenshot:



Success! The example.com server block is working!

Checkpoint-3: Understanding Virtual Host Behavior

After disabling the default site (000-default.conf), I observed interesting behavior:

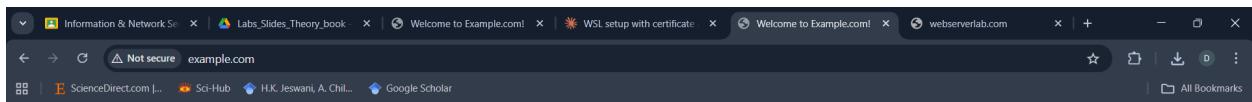
<http://localhost> showed the example.com content

<http://127.0.0.1> showed the example.com content

<http://webserverlab.com> showed the example.com content

Explanation: When the default site is disabled, Apache uses the first enabled virtual host as the fallback for any request that doesn't match a specific ServerName. Since example.com was the only enabled virtual host, all requests defaulted to it. This demonstrates Apache's default behavior in handling unmatched domain requests.

I verified the enabled sites using: ls -la /etc/apache2/sites-enabled/



Success! The example.com server block is working!



Which showed only example.com.conf was enabled.

Checkpoint- 4: Setting Up Second Virtual Host ([anothervhost.com](#))

Step 1: Creating Directory Structure:

```
sudo mkdir -p /var/www/anothervhost.com/html  
sudo chown -R $USER:$USER /var/www/anothervhost.com/html  
sudo chmod -R 755 /var/www/anothervhost.com
```

Step 2: Creating Index Page

```
<html>  
<head>  
<title>Welcome to AnotherVhost.com!</title>
```

```
</head>
<body>
<h1>This is Another Virtual Host!</h1>
<p>You are viewing anothervhost.com</p>
</body>
</html>
```

Step 3: Creating Virtual Host Configuration

Created /etc/apache2/sites-available/anothervhost.com.conf:

```
<VirtualHost *:80>
    ServerAdmin admin@anothervhost.com
    ServerName anothervhost.com
    ServerAlias www.anothervhost.com
    DocumentRoot /var/www/anothervhost.com/html
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Step 4: Adding to Hosts File

127.0.0.1 anothervhost.com

Also added entries to Windows hosts file at

C:\Windows\System32\drivers\etc\hosts:

127.0.0.1 example.com

127.0.0.1 anothervhost.com

Step 5: Enabling and Testing

```
sudo a2ensite anothervhost.com.conf
```

```
sudo apache2ctl configtest
```

```
sudo systemctl restart apache2
```

Step 6: Verification

Both virtual hosts were verified using curl commands:

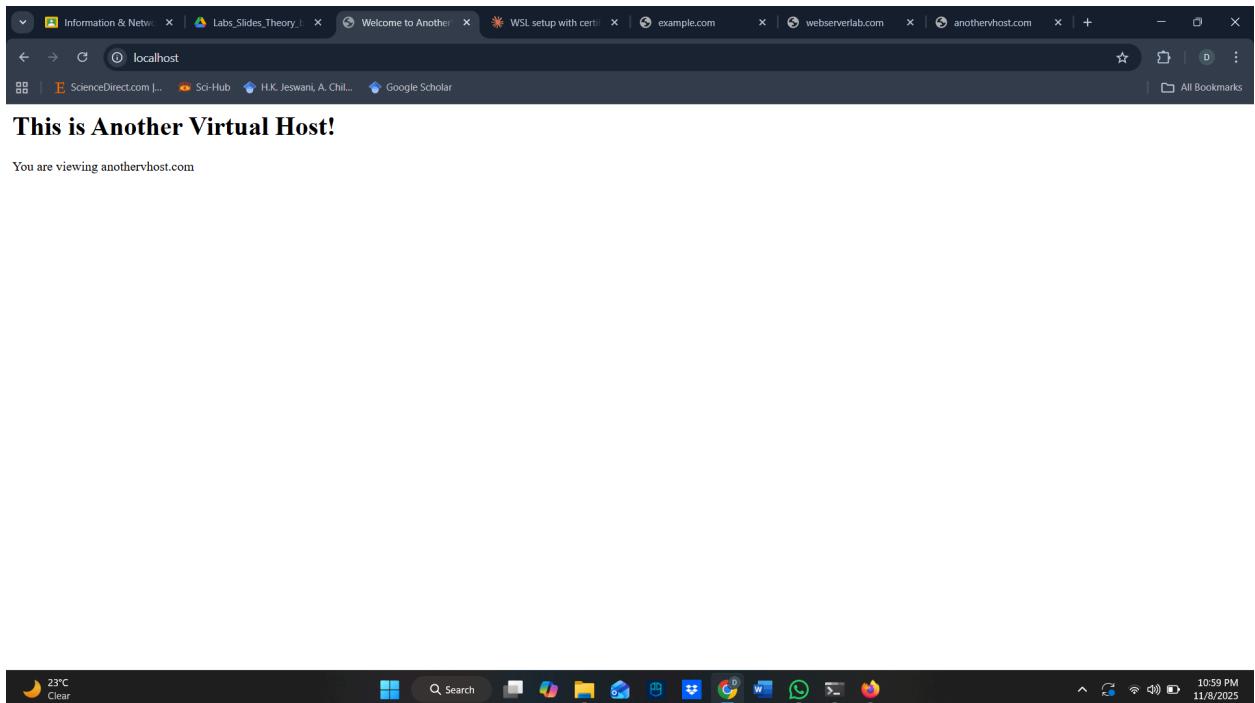
```
curl http://example.com
```

```
curl http://anothervhost.com
```

```
curl -H "Host: example.com" http://localhost
```

```
curl -H "Host: anothervhost.com" http://localhost
```

All commands returned the correct HTML content for their respective virtual hosts, confirming proper configuration. Given screenshot:



Task 3: Hosting Dynamic Websites using HTML and JavaScript

I created two dynamic websites with interactive forms that process user input using JavaScript.

Dynamic Website 1: Simple Calculator

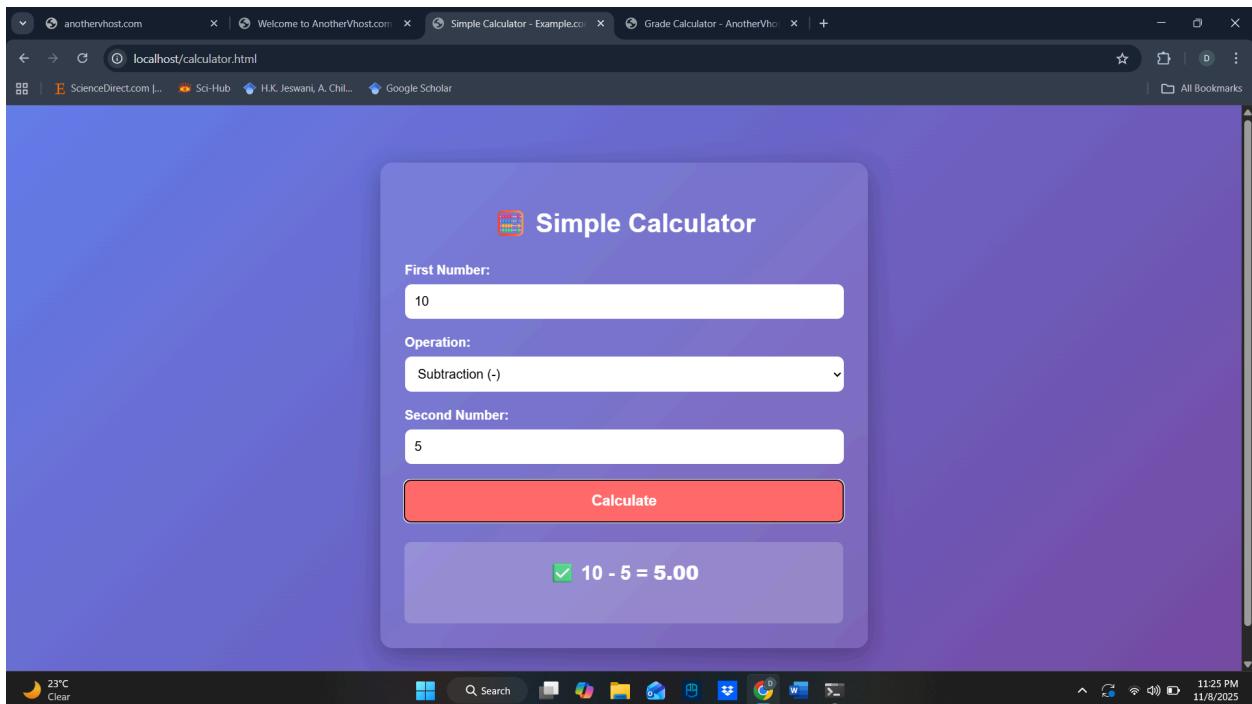
Features :

1. Accepts two numerical inputs
 2. Provides four operations: Addition, Subtraction, Multiplication, Division
 3. Input validation to ensure valid numbers are entered
 4. Error handling for division by zero
 5. Real-time calculation and result display
 6. Responsive design with purple gradient background

Here is the screenshot:

Key JavaScript Functions:

calculate(): Performs arithmetic operations based on user selection
Input validation using parseFloat() and isNaN()
Event listeners for form submission and Enter key



Dynamic Website 2: Grade Calculator

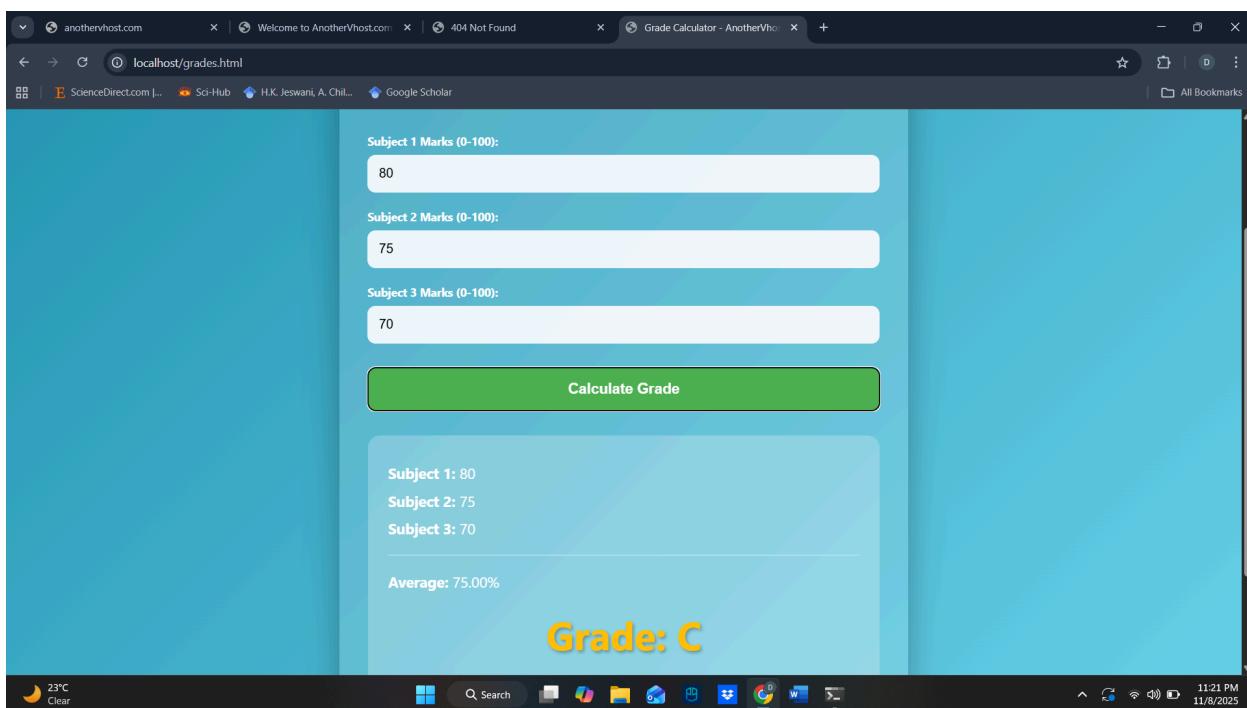
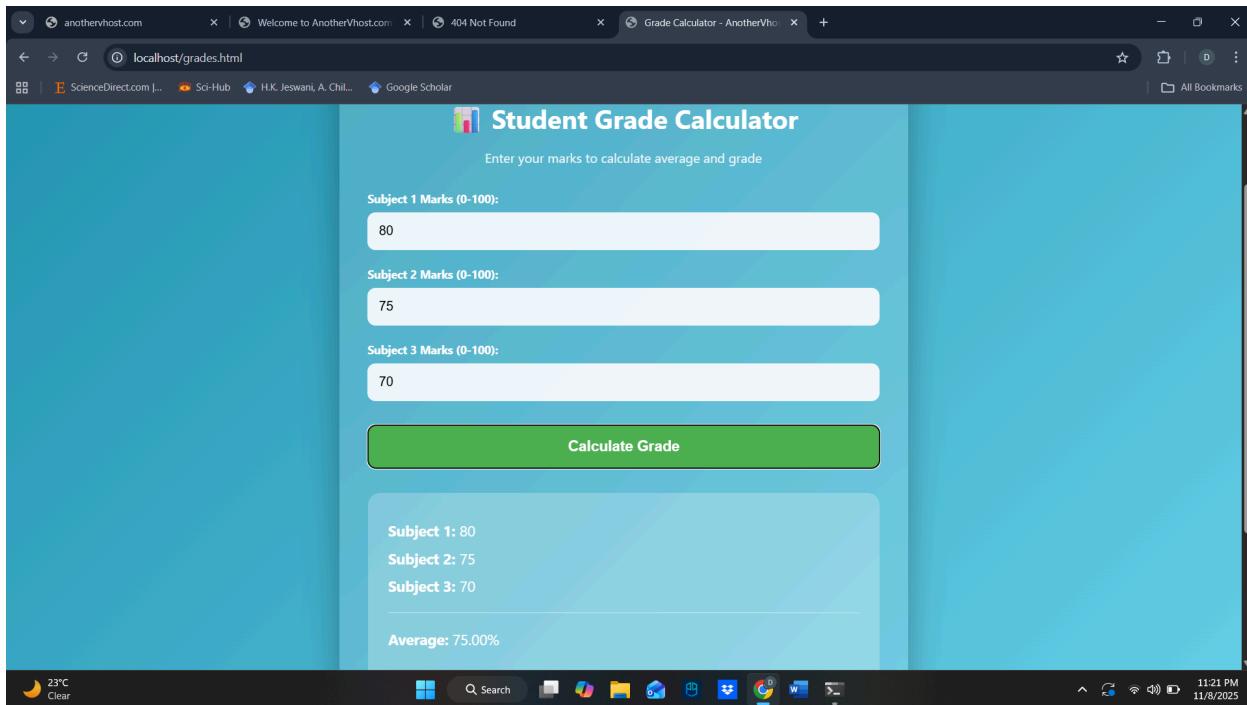
Features:

1. Accepts three subject marks (0-100)
2. Calculates average percentage
3. Assigns letter grade (A, B, C, D, F)
4. Input validation for mark ranges
5. Responsive design with colorful background

Key JavaScript Functions:

calculateGrade(): Computes average and determines grade, range validation (0-100 for each subject), dynamic result display with color-coded grades.

Here is the screenshot:



Deployment and Access: Due to WSL DNS resolution issues, I implemented a practical solution: `sudo cp /var/www/example.com/html/calculator.html /var/www/anothervhhost.com/html/`

This allowed both dynamic websites to be accessed via localhost:

<http://localhost/calculator.html> - Calculator application
<http://localhost/grades.html> - Grade calculator application