ASSignment-01

CSA4307

Internet Programming for client Server.

- JAYA HARINI. J.

192371034

BTech. CSBIOS.

Comparison of LAMP and WAMP Stacks for Web Application Development.

Project A - E-commerce Platform.

i) Small to medium sized retail business.

ii) Limited budget.

iii) Linux - based servers environment.

iv) Requires Security and Scalability.

Project B - CMS for Educational Institution.

i) operates in a Windows - based IT infrastructure

ii) Prefers Microsoft Technologies.

iii) Requires integration with existing Windows based Services.

1. Operating Systems : LAMP VS. WAMP.

LINUX (LAMP)

Advantages :

i) Open - Source, free and cost - effective.

ii) Highly Stable and Secure.

iii) Preferred in cloud environments (AWS, GCP, Azure)

iv) Native Support for command - line tools and server - Side automation.

Limitations :

i) Requires some familiarity with terminal commands and Linux file.

ii) Steeper learning curve for non - technical users.

Windows (WAMP)

Advantages:

i) user friendly GUI - based management.

ii) Better integration with Microsoft tools (Active Directory, office 365).

iii) Easier for clients and teams already using Windows Infrastructure.

Limitations:

i) Requires licensing (adds cost).

ii) Less optimal for hosting high - traffic, scalable systems.

iii) Less efficient in CLI / server automation than Linux.

2. Usage Scenarios and Stack Recommendations.

PROJECT - A: E - Commerce Platform.

Recommended stack: LAMP

WHY:

i) Free, open - source stack reduces hosting & setup costs.

ii) Scalable infrastructure for handling future traffic

iii) Strong community and security practices (e.g. regular updates, secure configuration).

iv) Ideal for cloud deployment and auto - scaling.

PROJECT B: Educational CMS.

Recommended Stack: WAMP.

WHY:

i) Compatible with Windows-based internal infrastructure.

ii) Easier for IT staff with Windows experience.

iii) Seamless integration with tools like Active Directory.

iv) GUI tools.

3. Common Application & Real-World Examples.

LAMP STACK (Linux, Apache, MySQL, PHP)

Typical Applications:

i) E-commerce Websites.

ii) Content Management System.

iii) online forums and blogs.

iv) RESTful APIs and backend Services.

Real-World Examples:

i) WordPress.

ii) Magento.

iii) Drupal.

WAMP STACK (Windows, Apache, MySQL, PHP)

Typical Application:

i) Internal Web applications.

ii) HR tools, dashboards.

iii) Educational portals and learning management

iv) Small business websites hosted on Windows.

Real - World Examples.

i) Moodle.

ii) Intranet CMS.

iii) HR Management Tools.

4. Critical Factors in Stack Selection.

FACTOR	LAMP	WAMP.
Security.	Excellent with Linux controls.	Requires extra configurations.
Performance	High, cloud - ready	Moderate, local development
Community support	Strong open - source backing	Smaller, Microsoft forums.
Ease of use.	CLI - Based, powerful	GUI - based, beginner friendly.
Scalability.	Excellent	Limited for large scale

5. Examples of Stack Relevance.

i) LAMP.

Best for startups and e-commerce platforms needing Scalable, Secure and budget - friendly solutions.

ii) WAMP:

Ideal for educational institutions or businesses already invested in Windows infrastructure.

Example scenarios where stack choice is crucial.

Scenario	Best stack	Explanation.
A fast growing e-commerce business planning to use AWS	LAMP	Better scalability, cloud readiness & performance.
A university with existing windows servers and staff with no Linux experience	WAMP	Reduces training overhead integrates with MS Services.
A startup prioritizing tight budget	LAMP	No software licensing cost, vast community Tools.
An internal CMS only used within a windows based office	WAMP	Simplifies setup, integrates with local windows network.

CONCLUSION.

PROJECT	Best Stack	Key Reason.
A (E-commerce).	LAMP	Cost-efficient, Secure & Scalable for growth.
B (CMS for institution)	WAMP	Best fit for windows-based environments & integrations

2. Responsive Event Registration Form using HTML & CSS.

HTML

```
<!Doctype html>.
<html>
<head>
  <meta charset = "UTF-8">
  <meta name = "viewport" content = "width = device-width">.
  <title> Event Registration </title>
  <link rel = "stylesheet" href = "style.css">
</head>
<body>
  <div class = "container">
  <h1> Event Registration </h1>
  <form>.
    <label for = "fullname"> FULL Name : </label>
    <input type = "text" id = "full name" name = "fullname required>
    <label for = "email"> Email Address : </label>
    <input type = "email id" = email name = "email" >
    <label for = "phone"> Phone Number : </label>
    <input type = "tel" id = "phone" name = "phone">
    <label for = "dob"> Date of birth : </label>
    <input type = "date" id = "dob" name = "dob" required>
  <label> Gender : </label>
```

output:

EVENT REGISTRATION.

FULLNAME: [ ] EMAIL: [ ]

NUMBER: [ ] D.O.B: [ ]

o Male o Female o Other

Hobbies / Interests:

☐ Reading ☐ Travelling ☐ Gaming ☐ Sports ☐ Music.

Event type: [-- Select an Event -]

[REGISTER.]

CSS

```
* {
    box-sizing : border-box;
    font-family : sans-serif;
  }

body
 {
   background-color : #f4f7fa;
    margin : 0;
    Padding : 0;
  }

. container
 {
   max-width : 600 px;
    margin : 50px auto;
    background : #fff;
```

```
    Padding : 30 px ;
    border - radius : - 8px ;
    box - shadow: 0 0 10px rgba (    )
  }
  h1
  {
    text - align : center ;
    color : #333;
  }
 form{
    display: flex;
    flex - direction : column;
 }
 label {
    margin : 10 px 0 5px;
    font - weight : bold;
    color : #555;
 }
 input,
 select,
 textarea
 {
    padding : 10px ;
    border : 1px solid #ccc;
    border - radius : 5px;
 }
```

1. Design choices

The form uses modern HMTL5 input types such as email, tel and date to enable built-in validation & device compatibility. Radio buttons, checkboxes, and a dropdown make user input simple and organized. CSS focuses on clean design with a neutral background, readable fonts and consistent spacing for better user experience.

2. Accessibility & usage.

Labels are properly linked to inputs for screen reader support. The form is easy to navigate with a keyboad and offers focus/ hover feedback for better accessibility & interaction.

3. Responsive Design.

The form uses flexible width and a max-width layout for adaptability across device. Flexbox ensures checkbone & radio wrap correctly on small screens.

3. Role of Each Component in LAMP and WAMP & How they communicate.

LAMP stack (Linux, Apache, MySQL, PHP).

PLATFORM: Linux.

LINUX:

The operating system that provides the environment for hosting and running the web Application.

APACHE:

The web server that handles HTTP requests from users and delivers web pages.

MYSQL

The relational database management system used to store application data like user info.

PHP.

The server-side scripting language used to create dynamic content & interact

WAMP STACK (Windows, Apache, MYSQL, PHP).

PLATFORM: Windows.

i) WAMP uses the same components (Apache, MYSQL, PHP) as LAMP, but it runs on Window OS.

ii) It is often preferred to development on windows System

HOW THE COMPONENTS COMMUNICATE

i) User Request: A user Request website (eg., WWW. example. com).

ii) Apache receives the request & identifies the requested resource

iii) PHP scripts are executed by the PHP engine if the request is for .php file.

HOW TO CONFIGURE A LAMP STACK.

1. update the package index

Sudo apt update

2. Install Apache Web Server.

Sudo apt install apache2.

3. Install MYSQL Database Server

Sudo apt install mysql - Server

4. Secure MYSQL Installation

Sudo mysql_secure_installation

5. Install PHP & Required modules

Sudo apt install php libapache2-mod-php.

6. Restart Apache to apply changes.

Sudo systemctl restart apache2.

7. Test PHP.

echo "<?php phpinfo(); ?>" | Sudo tee /var/WWW/html/info.php.