****

**Tayyaba Rehman**

**49690**

**Assignment 1**

**BSCS**

**Web programming**

**Submitted to:**

**Ma’am Habiba Khatoon**

**Web Development Fundamentals Assignment**

**Part 1: Understanding Web Basics**

1. **Research on How the Web Works**:

The World Wide Web which is commonly denoted as WWW, W3, or the Web, represents a system of interconnected public webpages that are available through the Internet .To access websites securely and efficiently, we need HTTP/HTTPS, the client-server model, DNS, and browsers.

**HTTP:**

* HTTP (Hyper Text Transfer Protocol) is the protocol used for transferring data between a web server and a browser to load web pages.
* It operates in a request-response model, where the browser sends a request for a webpage, and the server responds with the page's data.
* HTTP is stateless, meaning each request is independent, and it does not retain information about previous requests.

**HTTPS:**

* HTTPS (Hyper Text Transfer Protocol Secure) is the secure version of HTTP, using encryption to protect data transferred between the browser and the server.
* It uses SSL/TLS certificates to encrypt the data, ensuring privacy and preventing third parties from intercepting sensitive information.

**Difference between Http and https:**

The key difference between HTTP and HTTPS is that HTTPS uses encryption (SSL/TLS) to secure data transmission, while HTTP transmits data without encryption, making it vulnerable to interception

**Client server model:**

* The **client-server model** is distributed **network architecture** that provides a frame work that establish the connection over a network between clients (devices or applications) and servers (systems) to exchange data.
* The client requests resources, while the server provides them, enabling efficient communication and resource sharing in distributed systems.
* This architecture enhances **scalability** and **security** by separating responsibilities, allowing easier management of resources and centralized control.

**DNS:**

* **DNS (Domain Name System)** is a hierarchical system that translates human-readable domain into machine-readable IP addresses, enabling computers and devices to locate and communicate with servers hosting websites over the internet.
* When you enter a website address (e.g., www.example.com) in your browser, DNS translates it into an IP address, allowing the browser to connect to the correct server and display the website.
* DNS simplifies web navigation by enabling users to access websites using easy-to-remember domain names instead of complex IP addresses.

**Roles of browser:**

* A software application used to access information on the World Wide Web is called a Web Browser.
* A **browser** (fire fox, Google chrome, Opera, Apple safari) which acts as a client and sends a request to a **web server** using HTTP/HTTPS, asking for specific data (like a webpage), which the server processes and sends back to the browser.
* The browser **renders** the received data (HTML, CSS,JavaScript) into a visual webpage, allowing users to interact with the content, while the server handles requests from multiple clients.

1. **Exploring Web Hosting Options**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Types of hosting system | Role | Cost | Reliability | Scalability |
| Shared Hosting | Hosts multiple websites on a single server | $2 - $10/month | Good (99.9% uptime) but shared resource | Limited; upgrade to VPS or Dedicated hosting required as traffic grows |
| VPS hosting | |  | | --- | |  |  |  | | --- | | Provides isolated resources for each user | | $20 - $80/month | More reliable, isolated resources | Highly scalable; easy to increase CPU, RAM, and storage |
| Cloud Hosting | Uses multiple servers to distribute load and resources | $10 - $200/month | Very reliable (99.9% uptime+), distributed across servers | Highly scalable; resources can be adjusted based on demand |
| Dedicated hosting | Provides full server control and dedicated resources | $80 - $200+/month | Very reliable, full control over server | Maximum scalability; full control to upgrade hardware as needed |

**Part 2: Deployment and Security**

1. **Deploying a Website**:
2. **Register a Domain Name**
   * Choose a **relevant and memorable domain** that aligns with your website’s content and brand identity.
   * Example :Register it through a **domain registrar** like GoDaddy or Namecheap.
3. **Choose a Web Hosting Provider**
   * Select a hosting provider based on your website’s needs (e.g., **shared, VPS, or dedicated** hosting).
   * Ensure the hosting provider offers **reliable uptime**, **scalability**, and **support** for any technical issues.
   * Example: **Bluehost** or **HostGator** provide hosting services where your website files are stored
4. **Set Up DNS (Domain Name System)**
   * **DNS** translates your domain name into an **IP address** that browsers use to find your website’s server.
   * After registering your domain, configure your DNS records (such as **A record** or **CNAME**) to point to your hosting server’s IP address.
   * Example: You configure DNS records to point **example.com** to your hosting server’s IP address (e.g., **192.168.1.1**).
5. **Upload Website Files via FTP/SFTP**
   * Use **SFTP** (secure file transfer) to upload your website’s files (HTML, CSS, images) to the server.
   * Upload files to the correct directory on the server, typically **public\_html** or **www**, to make them accessible to visitors.
   * You use **FileZilla (SFTP)** to upload your HTML files from your computer to the server’s **public\_html** folder.
6. **Set File Permissions**
   * File permissions control access levels to your website’s files and directories. Use the **least privilege principle** to minimize security risks.
   * Common settings are **644** for files and **755** for directories, preventing unauthorized access while ensuring the website functions correctly.
   * Example: Setting file permissions to **644** for HTML files allows the server to read them but prevents unauthorized editing.
7. **Test the Website**
   * Ensure your website works across **different browsers** (Chrome, Firefox, Safari) and devices (desktop, tablet, mobile) for compatibility.
   * Test all **website functionality** (links, forms, buttons) to ensure everything operates smoothly and without errors.
   * **Example**: After uploading your website, you test it on different browsers (Chrome, Firefox, Safari) and devices (desktop, tablet, mobile) to ensure that it loads correctly and all links and forms work as expected.
8. **Set Up SSL/TLS (Optional for Security)**
   * Install an **SSL/TLS certificate** to secure your website with **HTTPS**, encrypting communication between your server and users.
   * **HTTPS** enhances security, improves SEO rankings, and builds **trust** with your users by showing the **padlock symbol** in browsers.
   * Example: You get an SSL certificate so that your website starts with **https://** and shows a padlock symbol, meaning it's secure.
9. **Web Security Essentials**:

### ****HTTPS (Hyper Text Transfer Protocol Secure)****

* **Threat**: Without HTTPS, data between the server and the client (user) is transmitted in plain text, making it vulnerable to **man-in-the-middle attacks** (MITM), where attackers intercept or modify data.
* **Prevention**: **Enable HTTPS** on your website by installing an SSL/TLS certificate. This encrypts data and ensures secure communication between the browser and the server, protecting user data (e.g., login credentials, payment details).
* **Example**: Websites like **www.bank.com** use HTTPS to encrypt sensitive user information like bank account numbers and passwords.

### 2. ****SSL/TLS Certificates****

* **Threat**: Without SSL/TLS, your website and its visitors are vulnerable to data interception or tampering.
* **Prevention**: **Install an SSL/TLS certificate** to enable HTTPS. SSL/TLS certificates encrypt data between the server and the client, ensuring confidentiality and integrity.
* **Example**: **Let’s Encrypt** offers free SSL/TLS certificates, which can be installed on your server to provide encryption for your website.

### 3. ****Secure Coding Practices****

* **Threat**: **SQL injection**, **cross-site scripting (XSS)**, and **cross-site request forgery (CSRF)** are examples of attacks that exploit vulnerabilities in improperly coded websites.
* **Prevention**:
  + Follow **secure coding practices** like input validation, output encoding, and using prepared statements in SQL to avoid SQL injection.
  + Regularly update and patch your code to fix vulnerabilities.
* **Example**: Instead of directly embedding user input into an SQL query, use **prepared statements** like SELECT \* FROM users WHERE id = ? to prevent SQL injection.

### ****Web Application Firewalls (WAF)****

* **Threat**: Malicious traffic (e.g., bots, hacking attempts) can exploit website vulnerabilities, causing security breaches.
* **Prevention**: Implement a **Web Application Firewall (WAF)** to monitor, filter, and block malicious traffic before it reaches your web application. A WAF can protect against SQL injection, XSS, and other types of attacks.
* **Example**: Tools like **Cloudflare** and **Sucuri** offer WAF services to protect your website from attacks like DDoS (Distributed Denial of Service) and malicious bots.

### Part 3: ****Emerging Trends in Web Development****

### ****Progressive web apps****

**Definition:** A **Progressive Web App (PWA)** is a web application that delivers an app-like experience with features like offline access, push notifications, and fast performance without requiring installation.

* **Benefits:**
  1. **Offline Access**: Works without an internet connection using cached data.
  2. **Fast Performance** : Loads quickly and provides a smooth user experienc
  3. **No App Store Dependence** : Can be used directly from a browser without downloading.
* **Challenges:**

1. **Limited iOS Support** : Some PWA features (e.g., push notifications) are restricted on iOS.
2. **Lower Hardware Access** : Limited access to device features like Bluetooth, Face ID, etc.
3. **SEO Challenges** : PWAs rely on JavaScript, which can affect search engine rankings.

* **Real-Life Examples:**
* **Twitter Lite** : A fast, data-saving version of Twitter that loads quickly and works offline, improving user experience in low-network areas.
* **Uber** : The PWA version allows users to book rides even on slow networks and low-end devices without needing the full mobile app

### ****Server less Architecture****

* **Definition:** Cloud-based computing where the developer focuses on code while the cloud provider manages the infrastructure.
* **Benefits:**
  1. **Cost-Efficiency:** Only pay for the resources used.
  2. **Scalability:** Automatically scales to meet demand.
  3. **Less Maintenance:** No server management needed.
* **Challenges:**
  1. **Cold Start Delay:** Initial requests may have a slight delay.
  2. **Limited Control:** Less control over server infrastructure.
  3. **Execution Time Limits:** Some serverless functions have a time limit.
* **Real-Life Examples:**
  1. **Netflix:** Uses serverless architecture for real-time data processing and scalability.
  2. **Airbnb:** Utilizes serverless computing to handle large volumes of user interactions.

### ****AI and Machine Learning Integration****

* **Definition:** Using AI and machine learning algorithms to enhance user experience and automate tasks.
* **Benefits:**
  1. **Personalized Experience:** Tailors content based on user behavior.
  2. **Automation:** Reduces manual work with smart automation.
  3. **Data Insights:** Analyzes data to predict trends and make decisions.
* **Challenges:**
  1. **Data Requirements:** Needs large amounts of data to function well.
  2. **High Cost:** Implementing AI can be expensive.
  3. **Privacy Concerns:** Can raise issues about user data security.
* **Real-Life Examples:**
  1. **Amazon Recommendations:** Recommends products based on purchase history and preferences.
  2. **Spotify's Music Suggestions:** Suggests playlists based on listening history

### ****Motion UI (Animations and Interactive Design)****

* **Definition:** Using animations and transitions to improve user engagement and experience.
* **Benefits:**
  1. **Engaging UI:** Makes the website more interactive and visually appealing.
  2. **Guides Users:** Animations can highlight key actions.
  3. **Branding:** Helps create a unique identity with custom animations.
* **Challenges:**
  1. **Slower Load Times:** Animations can delay page loading.
  2. **Distractions:** Overuse can distract users from the main content.
  3. **Compatibility Issues:** Not all browsers may support advanced animations.
* **Real-Life Examples:**
  1. **Apple’s Website:** Uses sleek animations to showcase products and features.
  2. **Dropbox:** Interactive elements that animate smoothly during user interaction.

### ****Voice Search and Voice-Activated Interfaces****

* **Definition:** Interfaces that allow users to interact with websites or apps through voice commands.
* **Benefits:**
  1. **Hands-Free Interaction:** Makes browsing and tasks more convenient.
  2. **Accessibility:** Helps users with disabilities interact easily.
  3. **Faster Search:** Allows quicker searches and commands.
* **Challenges:**
  1. **Accuracy Issues:** Voice recognition can struggle with accents or noise.
  2. **Limited Device Support:** Not all devices support voice activation.
  3. **Privacy Concerns:** Storing voice data raises security issues.
* **Real-Life Examples:**
  1. **Amazon Alexa:** Voice-activated assistant for smart home control and shopping.
  2. **Google Assistant:** Uses voice commands for web search and task automation.