A blue and white logo

AI-generated content may be incorrect.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **ASSIGNMENT COVER SHEET** | | | | | | |  |  |
|  |  |  |  |  |  |  | **ID** | | |  |  |
|  | **Name** | | Billy Hancock |  |  |  | **No. 141099798** | | |  |  |
|  | **Lecturer** | | Gary Dix |  |  |  |  |  |  |  |  |
|  | **Course No.** | | 603/1 |  |  |  |  |  |  |  |  |
|  | **Course Name** | | DSD |  |  |  |  |  |  |  |  |
|  | **Assign. Title** | | Server-Side Environments Evaluation and Razor Pages project |  |  |  |  |  |  |  |  |
|  | **Date Due** | | 05/06/2025 | **Date Submitted** 03/06/2025 | | | | | |  |  |

**Disclaimer of plagiarism and Collusion**

I declare that, to the best of my knowledge, no part of this assignment for the above course has been copied from any other student's work or from any other source except where due acknowledgement is made in the text.

I declare that no part of this assignment has been written for me by another person except where such collaboration has been authorized by the lecturer concerned.

I declare that none of the work has been submitted for any other assignment

**Signature:** Billy Hancock

## 1. Differences between server-side and client-side programming:

**Server-side programming purpose:**

The key purpose of server-side programming is to handle requests / input from the client on the server or perform other functions that need to remain hidden from the client. This server-side code runs on our servers and not the user’s browser, like client-side code would.

For example, let’s imagine a user tries to login to their account on our site, they would send that login request to our server, and then our server-side code would validate if the credentials are correct, and if so we would return them an access token we generated for them to use in future requests so we can validate they are authenticated and allowed to access private data associated with their account.

**Client-side programming purpose:**

The main purpose of client-side programming, on the other hand, is to handle tasks that run in the user’s browser, such as displaying the site’s UI, formatting requests based on form input, navigating the site using buttons, and other non-sensitive functions.

Because this code is run in the user’s browser, it can be viewed and accessed by the user, meaning we should never handle any private data or logic in our client-side code.

In our previous server-side code example, our client-side code would be responsible for reading the user input from the login form, then submitting a correctly formatted request to our server for validation when the “Login” button is pressed.

**Server-side programming functions:**

There are many functions of server-side programming, but most commonly they are things such as handling requests from the client (form submissions, API calls, etc.), authenticating users and controlling their access to specific data, interacting with the database to store or retrieve data.

Server-side code is also responsible for handling other specialized business logic. For example, if a business offers a web tool for generating personalized resumes, these resumes would be generated in server-side code, and the result would be sent back to the user.

**Client-side programming functions:**

Just as with server-side programming, client-side programming has a wide range of functions. The most common include things such as rendering the site via HTML, CSS & JavaScript, handling user input and checking it was entered correctly (such as checking a password is long enough, or an email was entered correctly) before submitting it in a request to the server.

Client-side code can also be responsible for storing non-sensitive data in the user’s browser. For example, if the user logs in and receives their access token, we might want the user to stay logged in even if they close their browser tab and reopen it later. To handle this, our client-side code stores this token in our browser’s local storage, and checks to see if there is a valid token there whenever we load our website, to see if it can automatically log the user in.

**Why are they different?**

There are many key differences between server-side and client-side programming, but it all comes down to their unique purposes and functions. Client-side programming is for non-sensitive code and functionality that we assume the user can tamper with, so it is used to reduce load on the server when the code doesn’t need to remain private. Server-side programming, however, handles logic we don’t want the client to be able to access, and we can safely assume it can’t be tampered with (presuming we correctly sanitize any user inputs we are passing through it).

For example, when the client submits a login request, our client-side code can do some basic sanitization on the users input to ensure they haven’t done something like miss an @ symbol in their email when entering it into the form. If we unnecessarily put this check on the server, we would have to wait for the request to reach the server, then the response to return to the client, telling them their email is incorrect. With client-side code, however, we can immediately spot this problem and save time and resources by pointing it out instantly. Now, since this code is on the client, the user can easily go in and remove this check if they desire, but this does not matter and everything we care about remains secure in our server-side code.

## 2. Security & Permissions:

**Server-side programming:**

Server-side programming’s security and permissions are very straightforward. Your server-side code runs on your controlled server, which you can assume is secure and that the code will remain untampered (presuming your server infrastructure doesn’t fall victim to malware). Any incoming data to your server, however, cannot be trusted at all, and we must assume that any and every part of it can be forged by a malicious user, therefore we must thoroughly validate and filter this incoming data. Despite the fact we completely trust our server-side code, sensitive data that our server-side code has access to (such as credit card information, passwords, etc.) should still only be stored in an encrypted way, ensuring that even if our server is breached by attackers, they will not have access to plaintext sensitive information.

**Client-side programming:**

Client-side programming’s security and permissions are even more straightforward. Not a single line of client-side code should be trusted, and we should assume it is totally insecure, and a malicious client can modify it in any way they like. Because of this, any important logic should also be performed on the server (such as password requirements when creating an account, etc.).

## 3. Features of server-side programming:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Django** | **Asp.NET Core** | **Laravel** |
| **Framework** | Python web framework | Open-source .NET framework | PHP web framework |
| **Language** | Python | C# (or F# / VB.NET) | PHP |
| **Security Mechanisms** | * SQL injection prevention * XSS prevention * CSRF prevention * Clickjacking prevention * Strong password hashing (defaults to PBKDF2) | * CSRF prevention * Built in identity framework for authentication * SQL injection prevention * XSS prevention | * CSRF prevention * XSS prevention * SQL injection prevention * Strong password hashing (bcrypt / argon2) |