**Web Development Stacks**



**State all the web application parts and services**

**Front-End**

The program editor is used is **visual stido code**, front end codes are done with **html, css and javascript.** In front-end, developers codes the site graphics which the users can see them. thus, storage thosgraphics data in the sever called **presentation tier**.

we have two way to show the graphics sites to the user:

**Client side rendering**

* if we excute javascript through **blazor, angular or Next js,** so the code will send to client pc and run code there to get the site graphics.

**Server side rendering**

* if we excute javascript through **MVC or Blazor** so the code will be in the server and ony send the site graphics to the client pc.

**Back-End**

The program editor is used is **visual stido and SSMS ,** back end codes are done in two parts first part is **programming language** and **framework**, which storage in server called **logic tier**, the second part is **database**, which storage in server called **data tier.**  connect between logic tier and data tier using technology called **ORM,** which executes the input SQL query and converts (maps) output DataReader (or equivalent in other languages) to your strongly typed objects.

In front end, if use type of client side rendering in this case we need to connect front end code with program language and framework in backend through **API.** and if use type of server side rendering in this case **direct access** between front end code with program language and framework in backend.

**State all the web stacks that you know and compare between them : where to use, when to use, features**

**.Net**

Where to use: windows application, enterprise level soluations

When to use: in large project appliaction , application with high organization and desisions making.

Features:

1. Build apps that are scalable.
2. fast.
3. simple to update.
4. High performance.

**Java**

Where to use: Android mobile application

When to use: when performance, scalability and origanazition are critical.

Features:

1. Platform Independent
2. Simplicity and Object-Oriented Programming

**Python**

Where to use: real live applications, automation scripts

When to use: use when the application need deal with real time data and we need very high speed of sent data, or when rapid development and ease of learning are priorities.

Features:

1. Easy to read and write syntax
2. storage libraries and framworks ex., Django, Flask
3. Excellent community support.

**Mern**

Where to use: real live applications and single-page applications

When to use: use when the application need deal with real time data and we need very high speed of sent data and also for when building modern, responsive web apps.

Features:

1. Building interactive user interfaces
2. Fast development and deployment cycles
3. Deal with real time data.