Name: Dhikrullahi Ganiyat Kofoworola.

Class: HND I Computer Science.

1. Evolution of .NET Framework and C#

* The evolution of the .NET Framework and C# has been marked by continous updates, enhancements and the introduction of new features to meet the changing needs of developers and technology advancements.

The framework's early releases focused on unified Windows application development, but subsequent versions introduced key features. C# evolved with generics in 2.0, LINQ in 3.5, and async/await in 4.0, enhancing developer productivity. The pivotal shift occurred with .NET Core in 2016, emphasizing cross-platform compatibility and open-source collaboration. This trend continued with .NET 5.0 in 2020, unifying .NET Core and .NET Framework into a single platform, streamlining development. In 2021, .NET 6.0 prioritized performance and cloud-native applications, unveiling .NET MAUI for cross-platform app development. Throughout this journey, Microsoft's commitment to adaptability and responsiveness to industry needs has positioned .NET and C# as a versatile and modern development ecosystem.

1. Explain ;

* Mono: is an open-source implementation of the .NET Framework, designed to be cross-platform and compatible with various operating systems. Originally developed by Xamarin (later acquired by Microsoft), Mono allows developers to build and run .NET applications on platforms beyond Windows. Its key features include Cross-Platform Compatibility, Common Language Infrastructure (CLI) compliance, Development with C#, Libraries and APIs, and Integration with Xamarin.
* Xamarin: is a cross-platform app development framework that allows developers to create native mobile applications using the C# programming language and the .NET framework. Originally an independent company, Xamarin was later acquired by Microsoft in 2016, and its technology is now integrated into the Visual Studio IDE. Its key features includes Cross-Platform development, Native user Interfaces, code reusability, integration with Visual Studio, Xamarin.forms, access to Native APIs, Xamarin.essentials.
* COM: COM is a binary-interface standard for software components introduced by Microsoft, primarily used in Windows operating systems. COM is a set of specifications and guidelines for creating and using binary software components in a way that promotes reusability and interoperability. It enables communication between software components regardless of the programming language they are written in.
* .Net Core: .NET Core is an open-source, cross-platform framework developed by Microsoft for building modern, scalable, and high-performance applications. It is designed to be modular, lightweight, and suitable for a wide range of applications, including web, cloud, mobile, gaming, IoT, and more. Its key features includes Cross-Platform, Open source, Modularity, Unified Development Model, Performance,Command-Line Interface (CLI), ASP.NET Core, Container Support, Language Support, Community and Ecosystem.
* Unity with C#: Unity is a popular cross-platform game development engine that allows developers to create interactive and immersive experiences for various platforms, including PC, consoles, mobile devices, and virtual reality (VR). C# is the primary programming language used in Unity for scripting game logic and behavior. Unity with C# allows developers to create interactive and visually compelling games and simulations by leveraging a powerful game engine and the flexibility of the C# programming language.
* REST: or Representational State Transfer, is an architectural style for designing networked applications. It was introduced by Roy Fielding in his doctoral dissertation in 2000. REST is commonly used in the design of web services and APIs (Application Programming Interfaces) that enable communication between different software systems. RESTful principles are widely used in designing APIs for web applications, allowing for scalable, maintainable, and loosely-coupled systems. APIs following RESTful principles are often referred to as RESTful APIs.

1. Key Functions of CLR

* Exception Handling: is a crucial programming practice for managing unexpected errors. It allows developers to gracefully handle and recover from runtime issues, maintaining application stability. Well-implemented exception handling enhances reliability, facilitates debugging, and ensures a more robust and resilient software system.
* Assembly Loading and Execution: Assembly loading in .NET involves locating, loading, and executing compiled code units (assemblies). The Common Language Runtime (CLR) dynamically loads assemblies at runtime. It enhances flexibility, supports versioning, and enables modular development, allowing applications to adapt to changing requirements without requiring a full rebuild.
* Thread Management: Thread management in computing involves creating, scheduling, and synchronizing threads for concurrent execution. Efficient thread management, facilitated by the operating system or programming language runtime, optimizes parallel processing. It's critical for improving application responsiveness, scalability, and achieving efficient resource utilization in multithreaded environments.