1. What is .NET CORE? Is it upgraded version .NET Framework?

* .NET Core is a open source Framework for building modern, high-performance applications. such as Web, Windows, Mobile ,AOT & Cloud based applications

* .NET Core, developed by Microsoft, is managed under the .NET Foundation. .NET Core is written in C# and C++ and licensed under MIT license. The first version, .NET Core 1.0, was released in 2016 & Latest version is .NET CORE 7.0 , we can call it .NET 7

* It's completely different from .NET Framework

2. What are the advantage of .NET Core ?

* Open source

* Cross platform support, so it can be run on Windows, Linux, and Mac.

* ASP.NET Core can handle more requests than the ASP.NET

* Provides better testability

* It Can deploy in IIS Server & Cloud

3. What are the Features in .NET Core?

* Built-in supports for Dependency Injection
* Built-in supports for the logging framework and it can be extensible
* Introduced a new, fast and cross-platform web server - Kestrel. So, a web application can run without IIS
* Command-line supports to creating, building, and running of the application
* It has good support for asynchronous programming
* There is no web.config file. We can store the custom configuration into an appsettings.json file
* Rate Limiting & Minimal API

4. What is metapackages?

* The .NET Core 2.0 introduced Metapackage which includes all the supported packages by ASP.NET code with their dependencies into one package.

* It helps us to do fast development as we don't require to include the individual ASP.NET Core packages.

* The assembly Microsoft.AspNetCore.All is a meta package provided by ASP.NET core.

5. Can ASP.NET Core application work with full .NET 4.x Framework?

* Yes. ASP.NET core application works with full .NET framework via the .NET standard library.

6. What is the startup class in ASP.NET core?

* The startup class is the entry point of the ASP.NET Core application. This class contains the application configuration related items.

* .NET 6 onwards this startup file removed . Whatever actions handled in startup class that moved to Program.cs file.

7. What is the use of the ConfigureServices method of the startup class?

* This is an optional method of startup class.

* It can be used to configure the services that are used by the application.

* This method calls first when the application is requested for the first time.

* Using this method, we can add the services to the DI container, so services are available as a dependency in the controller constructor.

8. What is the use of the Configure method of the startup class?

* It defines how the application will respond to each HTTP request.

* We can configure the request pipeline by configuring the middleware.

* It accepts IApplicationBuilder as a parameter and also it has two optional parameters: IHostingEnvironment and ILoggerFactory.

* Using this method, we can configure built-in middleware such as routing, authentication, session, etc. as well as third-party middleware.

9. What is middleware?

* It is software that is injected into the application pipeline to handle requests and responses.

* They are just like chained to each other and form as a pipeline.

* The incoming requests are passed through this pipeline where all middleware is configured, and middleware can perform some action on the request before passing it to the next middleware. Same as for the responses, they are also passing through the middleware but in reverse order.

10. What is routing in ASP.NET Core?

* Routing is functionality that map incoming request to the route handler.
* The route can have values (extract them from the URL) that are used to process the request.
* Using the route, routing can find a route handler based on the URL. All the routes are registered when the application is started.

* There are two types of routing supported by ASP.NET Core

1, The conventional routing

2, Attribute routing

11. How to enable Session in ASP.NET Core?

* The middleware for the session is provided by the package Microsoft.AspNetCore.Session. To use the session in the ASP.NET Core application, we need to add this package to the csproj file and add the Session middleware to the ASP.NET Core request pipeline.

```
public class Startup
{
public void ConfigureServices(IServiceCollection services)
{
....
....
services.AddSession();
services.AddMvc();
}
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
....
....
app.UseSession();
....
....
}
}
```

12. What are the various JSON files available in ASP.NET Core?

* There are the following JSON files in ASP.NET Core :

1, global.json - allows you to define which . NET SDK version is used when you run . NET CLI commands.

2, launchsettings.json - describes how the application can be launched

3, appsettings.json - used to store the application configuration settings such as database connection strings, any application scope global variables, and much other information.

13. What are the various JSON files available in ASP.NET Core?

4, bundleconfig.json - to control the minifying process including the option to rename locals for JavaScript files and whether to create a SourceMap file for the JavaScript file.

5, brower.json – contains browser information

6, package.json - contains a list of NPM packages that the developer want to be restored before the project starts.

14, What are the Common HTTP Response in .NET Core API?

Code	Action	Code	Action
200	OK	404	Not Found
201	Created	406	Not Acceptable
202	Accepted	423	Locked
204	No Content	500	Internal Server Error
400	Bad Request	501	Not Implemented
401	Un Authorized	502	Bad Gateway
403	Forbidden	503	Service Unavailable

15, How to achieve dependency injection in .NET Core API?

* Dependency injection is the inbuilt feature in .NET Core.
* It provides 3 ways to register DI to service

Transient

It creates an instance each time they are requested and are never shared. It is used mainly for lightweight stateless services.

Singleton

This creates only single instances which are shared among all components that require it.

Scoped

It creates an instance once per scope which is created on every request to the application.

15, How to achieve dependency injection in .NET Core API?

* Dependency injection is the inbuilt feature in .NET Core.
* It provides 3 ways to register DI to service

Transient

It creates an instance each time they are requested and are never shared. It is used mainly for lightweight stateless services.

Singleton

This creates only single instances which are shared among all components that require it.

Scoped

It creates an instance once per scope which is created on every request to the application.

16, How to use Entity Framework core in .NET Core?

* Entity Framework is an object-relational mapper (O/RM).
* The Entity Framework provides three approaches to create an entity model

1, Database First

2, Code First

3, Model First

17, What is auto Mapper?

* AutoMapper is a simple library that helps us to transform one object type into another.

* Convention-based object-to-object mapper that requires very little configuration.

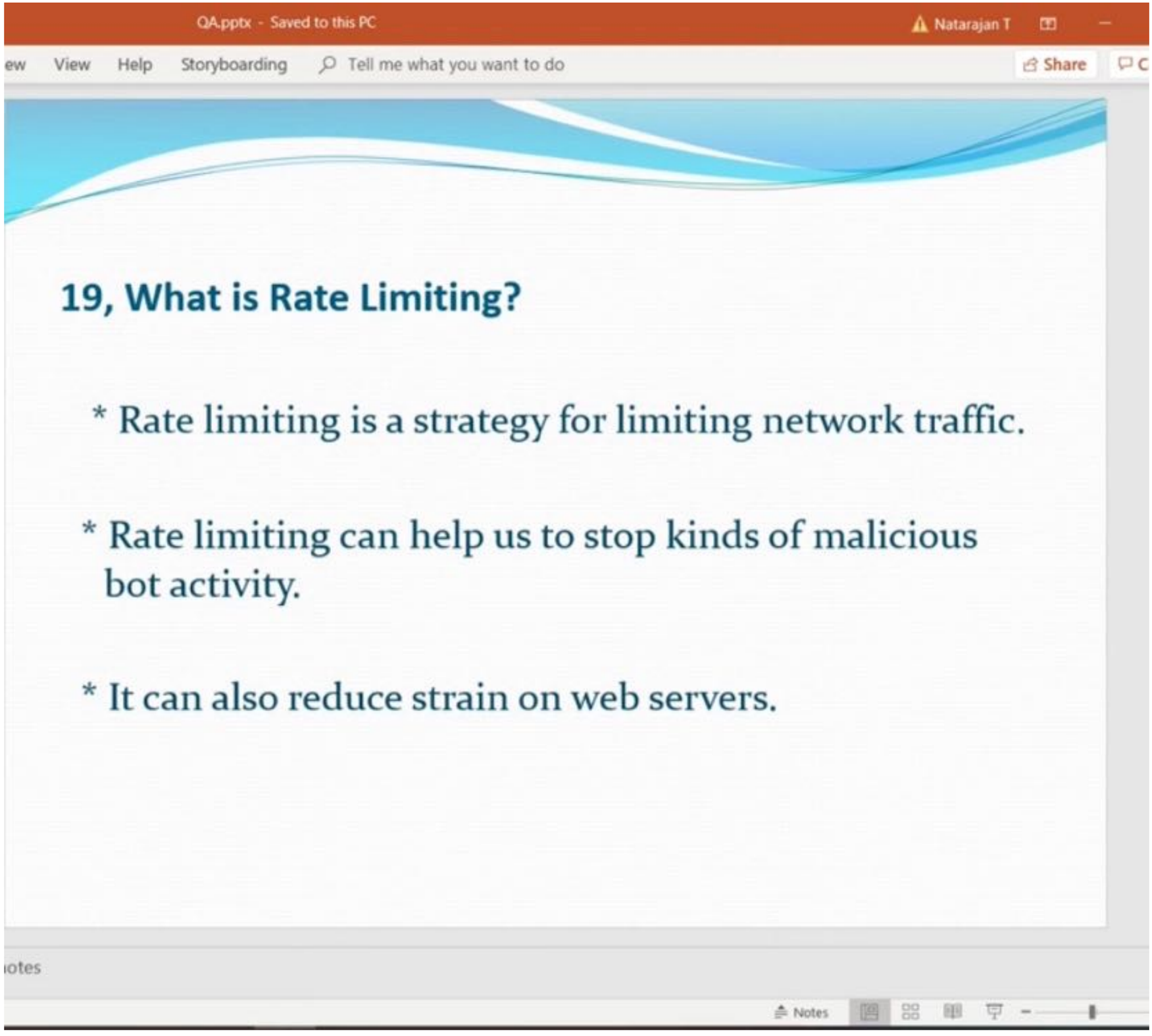
* Library Used Auto mapper

18, What is CORS? How to enable in .NET Core?

* Cross-origin resource sharing (CORS) is a browser security feature that restricts cross-origin HTTP requests .
* If your REST API's resources receive non-simple cross-origin HTTP requests, you need to enable CORS support.

1, https://domain1.com

2, https://domain2.com


QA.pptx - Saved to this PC
Natarajan T
ew View Help Storyboarding Tell me what you want to do
Share
19, What is Rate Limiting?
* Rate limiting is a strategy for limiting network traffic.
* Rate limiting can help us to stop kinds of malicious bot activity.
* It can also reduce strain on web servers.
otes
Notes

20, What is Authentication & Authorization in .NET Core API?

* **Authentication** is the process of validating user identity & **Authorization** is the process of providing permission to access the resource

* Authentication is used to protect our applications data from unauthorized access.

21, What is Basic Authentication?

* Basic authentication sends user names and passwords over the Internet as text that is Base64 encoded, and the target server is not authenticated.

* This form of authentication can expose user names and passwords. If someone can intercept the transmission, the user name and password information can easily be decoded.

22, What is JWT Authentication?

* JSON Web Tokens (JWT) are an open standard, which is defined in JSON Web Token (JWT) Specification RFC 7519.

* They securely represent claims between two parties.

23, What is Refresh Token?

* A refresh token is a special token that is used to obtain additional access tokens.

* This allows you to have short-lived access tokens without having to collect credentials every time one expires.

24, What is Minimal API?

* Minimal APIs are architected to create HTTP APIs with minimal dependencies.

* They are ideal for micro services and apps that want to include only the minimum files, features, and dependencies in ASP.NET Core.

* It's introduced in .NET 6.0

25, How do you implement security in a .NET Core web application?

1, Authentication and Authorization: You can use the built-in Microsoft.AspNetCore.Authentication package to implement various authentication schemes such as cookies, OpenID Connect, and JSON Web Tokens (JWT). You can also use the built-in Microsoft.AspNetCore.Authorization package to implement role-based and claim-based authorization.

2, HTTPS: You can use the built-in Microsoft.AspNetCore.Server.Kestrel package to configure your application to use HTTPS.

3, CORS: You can use the built-in Microsoft.AspNetCore.Cors package to configure Cross-Origin Resource Sharing (CORS) headers to control which domains are allowed to make cross-origin requests to your application.

4, Cross-Site Request Forgery (CSRF): You can use the built-in Microsoft.AspNetCore.Antiforgery package to protect your application from CSRF attacks.

5, SQL Injection: You can use the built-in Microsoft.EntityFrameworkCore package to implement parameterized queries to prevent SQL injection attacks.

6, Cross-Site Scripting (XSS): You can use the built-in Microsoft.AspNetCore.Mvc.ViewFeatures package to implement encoding and validation to prevent XSS attacks.

28, How do you implement exception handling in a .NET Core application?

* In .NET Core, exception handling is implemented using try-catch blocks or using global exception handlers.

30, Can you explain the concept of Identity in .NET Core?

* Identity is a feature of ASP.NET Core that provides a way to handle authentication and authorization for web applications. It allows you to easily add support for user registration, login, and password management to your application.

* In .NET Core, identity is implemented using the Identity library, which provides a set of classes and interfaces for handling user identities and roles.

31, How do you implement Caching in a .NET Core application?

* Caching is a technique for improving the performance of a web application by storing frequently-used data in memory so that it can be quickly retrieved without having to be recalculated or fetched from a database.

* In .NET Core, there are several caching options available, such as in-memory caching, distributed caching, and response caching.

32, Can you explain the concept of SignalR in .NET Core?

* SignalR is a real-time communication library for ASP.NET Core that allows you to easily add real-time functionality to your web applications. It enables bi-directional communication between a client and a server, allowing the server to push updates to the client in real-time.

* SignalR uses a variety of underlying technologies to provide real-time communication, including WebSockets, Server-Sent Events, and Long Polling. It automatically chooses the best transport based on the client's capabilities and the network conditions.

33, How do you implement background tasks in a .NET Core application?

* Background tasks are a way to run long-running or scheduled operations in a .NET Core application without blocking the main thread. This allows the application to continue processing requests while the background task is running.

* In .NET Core, background tasks can be implemented using the IHostedService interface, which allows you to create a service that runs in the background when the application starts.

34, What is Serilog?

* Serilog is a third party library for enable logging in .NET core application