.Net Core

Day 1

* Suited for container based/web based application
* Used for portability
* Base line version is 2.2 , Visual Studio 2017 or + to be used
* Based on modular framework
* Modularity(loosely coupled system or plug and play) is achieved by Dependency Injection, Inversion of control
* Multithreading is used but with asynchronous calls
* No config file(JSON is used),launchsettings.json is used
* Dependency Injection is part of .net
* Application as exe but with dll extension

Day 2

* Data Access
* OLEDB used by Microsoft for Data access
* MTS used by implementing interface “ITransactionHandler”
* Even if database changes using LINQ there is no need to change the code.
* This is done using provider (below two are required)
  + LINQ
  + LINQ to DB provider
* Three ways provided by Microsoft
  + DBFirst
  + CodeFIrst/ModelFirst
* Using Delegate to hook to HTTP pipeline

**Query Used for creating model**

Scaffold-DBContext "Server=(localdb)\mssqllocaldb;Database=DeloitteDB;Trusted\_Connection=true;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models

Model First Approach

->Add-Migration Initial\_Baseline\_V1.0

(Creates Infracture script from model)

->Database-Update

(Update Infracture)

**V & V model**

Cross-cuts

1. Authentication
2. Authorization
3. Caching
4. Encrypt-Decrypt
5. Logging
6. Inflate/deflate
7. Virus Scan

CAB (Composite UI Application Block ) 🡪 SCSF 🡪 PRISM 🡪 Unity

ASP .net core 🡪 Action Filter , Middleware

Bridge design pattern can be implemented using “Extension”

Assignments:

MVC Application for creating ,opening word documents

Hint: Docx is open xml document

Day 3:

TagHelper which take input as a model

Microsoft Flow

**Service Oriented Architecture**

Can be implemented using:

* WCF
  + IIS
  + Post/Binding
  + Throttling
* Asmx
  + IIS
* WebAPI
  + Http/https/signalr
  + IIS
* ASPCore
  + Portable API
  + Not dependent on IIS and uses JSON