\*Entity Framework vs SQL

\*When to use which and which one is better

\* LeetCode

\* Design Pattern (https://www.dofactory.com/net/design-patterns)

\* Design Principle (https://www.youtube.com/watch?v=A5OD1gtaMm8)

\* TDD (https://www.youtube.com/watch?v=a8QvABt1l4M) -- not mandatory right now

\* BDD (https://www.youtube.com/watch?v=cq\_3wPnaX-w) -- not mandatory right now

\* Clean Archtecture (https://www.c-sharpcorner.com/article/clean-architecture-in-asp-net-core-web-api/)

\* Domain Dirven Design (https://www.youtube.com/watch?v=1Lcr2c3MVF4&t=44s)

\* Mediator (https://www.dofactory.com/net/mediator-design-pattern)

\* Terraform for orchestration (https://www.youtube.com/watch?v=l5k1ai\_GBDE)

\* APIs (REST, GraphQL, gRPC) {https://www.youtube.com/watch?v=opVobybT\_1w}

\* System Design:

- Don't go very deep for each component

- Don't have a set architecture in mind

- KISS

- Need to have justification for each component choosing (Casandra, Zookeeper, MQ, DynamoDB, ELB, Azure Load Balancer need to know)

- Steps (Understanding the problem and find design scope, propose high level design, deep dive, wrap up)

\* 7 network layers (https://www.checkpoint.com/cyber-hub/network-security/what-is-the-osi-model-understanding-the-7-layers/)

\* Difference between gateway and load balancer (https://www.c-sharpcorner.com/article/azure-application-gateway-vs-azure-load-balancer-vs-azure-traffic-manager-vs-azu/)

\* oAuth 2.0 (access token and refresh token) https://www.youtube.com/watch?v=JM8V55jNYt0

\* HTTP/HTTPS/TLS/SSL (https://www.youtube.com/watch?v=hExRDVZHhig, https://www.youtube.com/watch?v=eWdPWSBKxso)

\* Interface and Abstract Class (https://www.geeksforgeeks.org/difference-between-abstract-class-and-interface-in-c-sharp/)

\* Thread vs Process vs Task (https://stackoverflow.com/questions/3042717/what-is-the-difference-between-a-thread-process-task#:~:text=A%20process%20invokes%20or%20initiates,thread%20results%20in%20a%20task.)

\* Garbage Collection (https://www.geeksforgeeks.org/garbage-collection-in-c-sharp-dot-net-framework/)

https://www.youtube.com/watch?v=FLZc4xH4E8U&list=PLrtCHHeadkHp92TyPt1Fj452\_VGLipJnL&index=17

\* Class vs Struct (https://www.bytehide.com/blog/struct-vs-class-csharp)

\* Diamond Problem (https://www.educative.io/answers/what-is-a-diamond-problem-in-object-oriented-programming)

\* Why no multiple inheritence (https://www.c-sharpcorner.com/interview-question/why-multiple-inheritance-is-not-possible-in-c-sharp)

\* CQRS Pattern (https://www.c-sharpcorner.com/article/using-the-cqrs-pattern-in-c-sharp/)

\* DI (https://www.c-sharpcorner.com/blogs/what-is-dependency-injection-and-what-are-its-types2)

\* IoC (https://stackoverflow.com/questions/3058/what-is-inversion-of-control)

\* Class vs Record:

https://josipmisko.com/posts/c-sharp-class-vs-record

https://stackoverflow.com/questions/64816714/when-to-use-record-vs-class-vs-struct

\* String vs StringBuilder:

https://www.geeksforgeeks.org/c-sharp-string-vs-stringbuilder/

\* Boxing and Unboxing:

https://www.geeksforgeeks.org/c-sharp-boxing-unboxing/

\* Design patterns:

Creational:

- Abstract Factory => Animals with continent

- Factory Method => Pages, Document, Resume, Report

- Builder => Vehicle builder, Shop, Construct

- Prototype => Color clone

- Singleton => Load Balancer

Structural:

- Adapter => ChemicalDatabBank, Compound, RichCompund

- Bridge => Message, SystemMessage, UserMessage, IMessageSender, EmailSender, SmsSender (https://www.dotnettricks.com/learn/designpatterns/bridge-design-pattern-dotnet)

- Composite => DrawingElement, PrimitiveElement, CompositeElement

- Decorator => LibraryItem, Book, Video, Dcorator, Borrowable

- Facade => Loan, Bank, Credit , Mortgage (IsEligible)

Behavioral:

- Chain of Responsibility => Approvar, Successor

- Command => Command (execute, unexecute), CalculatorCommand, Calculator

- Observer => Stock, Investor

- Memento => SalesProspect, Memento

- Strategy => Sort, SortedList

- Mediator => Chatroom, Participants

\* What is kubernetes (https://www.youtube.com/watch?v=VnvRFRk\_51k)

\* What is cookies, sessions and tokens (https://www.youtube.com/watch?v=GhrvZ5nUWNg)

S => Single Responsibility Principle: Report add, get, save (should be separate)

O => Open close principle: Salary calculator for dev, senior dev, lse and if jse comes

L => Lisckov Substitution Principle: objects in a program should be replaeable by its subtype without altering the correctness of the program.

Loanee calculator, Savings account calculator, inquery calculator [GetAccounts()]

I => Interface Seggregation Principle: Many client-sepcific interfaces are better than one general purpose fatty interface.

IPrinterTask (Print(), PhotoCopy()) + (CannonPrinter, HPPrinter) => IPrinterTask, IPhotoCopyTask

D => Dependecy Inversion Principle: High level modules should not depend upon low level modules rather both should depend upon abstractions.

(ShowData() with logging feature) => Logging should be separate inteface DI into show data.

Factory Method: Burger, VeggieBurger, ChickenBurger, abstract Restaurant => CreateBurger(), VaggieRestauratnt, ChickenRestaurant

https://www.youtube.com/watch?v=EdFq\_JIThqM

Abstract Factory:allows you to produce families of related objects withour specifying their concrete classes.

Gpu, MsiGpu, AsusGpu, Monitor, MsiMonitor, AsusMonitor, abstract Company (this is the abstract factory),

MsiCompany (CreateGpu, CreateMonitor), AsusCompany(CreateGpu, CreateMonitor)

https://www.youtube.com/watch?v=QNpwWkdFvgQ

Builder: produces different types of representations of an object using same construction process.

Car with brand, model, color, id, Cabuilder with BuildBrand, BuildColor, BuildModel and return Build() with a car constructor. Car builder should

use a fluent api type construction. More over CarBuilder, CarSchemaBuilder then there will be Builder interface with Director DI with Builder.

https://www.youtube.com/watch?v=MaY\_MDdWkQw

Prototype: Create a exact copy of an object.

Prototype interace with Clone method. Car, Bus, Vehicle. Shallow Copy and Deep Copy.

https://www.youtube.com/watch?v=DcFhITC9v0E [extra info, prototype registry]

Singleton: https://www.youtube.com/watch?v=tSZn4wkBIu8

Chain of Responsibility: https://www.youtube.com/watch?v=FafNcoBvVQo

Command: The Command design pattern encapsulates a request as an object, thereby letting you parameterize clients with different requests.

Calculator (Undo, Redu, Calculate), abstract Command and concrete CalculatorCommand

https://www.dofactory.com/net/command-design-pattern

https://www.youtube.com/watch?v=UfGD60BYzPM

Mediator: The Mediator design pattern defines an object that encapsulates how a set of objects interact.

Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently.

https://www.dofactory.com/net/mediator-design-pattern

https://www.youtube.com/watch?v=35D5cBosD4c

Memento: TextArea, MementoTextArea, Editor

https://www.youtube.com/watch?v=\_Q5rXfGuyLQ

Observer: Stock, IInvestor, Investor

https://www.dofactory.com/net/observer-design-pattern

https://www.youtube.com/watch?v=-oLDJ2dbadA

Strategy: defines a family of algorithms puts each of them in a separate class and makes their object interchangeable.

Payment: ByCreditCard, ByPaypPal

https://www.youtube.com/watch?v=Nrwj3gZiuJU

Difference betn IEnumerable, ICollection and IList:

https://code-maze.com/csharp-ienumerable-icollection-ilist-list/#:~:text=ICollection%20extends%20IEnumerable%20.,perform%20any%20index%2Drelated%20operations.

IQueryable:

https://josipmisko.com/posts/c-sharp-iqueryable-vs-ienumerable

Yield: https://www.programiz.com/csharp-programming/yield-keyword

Async void vs Async task:

https://stackoverflow.com/questions/12144077/async-await-when-to-return-a-task-vs-void

https://www.oreilly.com/library/view/async-in-c/9781449337155/ch04.html#:~:text=A%20async%20void%20method%20is,finished%20or%20whether%20it%20succeeded.

https://blog.devgenius.io/understanding-c-methods-async-void-and-task-edbc7430451e#:~:text=Void%20methods%20are%20used%20when,and%20responsive%20code%20in%20C%23.

Transaction Isolation Levels:

https://learn.microsoft.com/en-us/sql/t-sql/statements/set-transaction-isolation-level-transact-sql?view=sql-server-ver16

Optimistic Locking:

https://www.mssqltips.com/sqlservertip/1501/optimistic-locking-in-sql-server-using-the-rowversion-data-type/

Task Cancellation Token:

https://hackernoon.com/why-do-you-need-a-cancellation-token-in-c-for-tasks

DDOS:

https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/

Rate Limiting:

https://www.quinbay.com/blog/understanding-rate-limiting-algorithms

https://betterprogramming.pub/4-rate-limit-algorithms-every-developer-should-know-7472cb482f48

User defined implicit and explicit conversion:

https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/user-defined-conversion-operators

https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/arithmetic-operators#user-defined-checked-operators

Horizontal and vertical scalling:

https://www.nops.io/blog/horizontal-vs-vertical-scaling/#:~:text=The%20primary%20difference%20between%20horizontal,.)%20to%20an%20existing%20machine.

Horizontal and vertical DB sharding:

https://www.linkedin.com/advice/0/what-trade-offs-between-horizontal-vertical-sharding#:~:text=Horizontal%20sharding%2C%20otherwise%20known%20as,zip%20code%2C%20or%20last%20name.

https://www.digitalocean.com/community/tutorials/understanding-database-sharding

CAP theorem (Consistency, Availibility, Partition Tolerance):

https://www.youtube.com/watch?v=BTKBS\_GdSms&list=PLrtCHHeadkHp92TyPt1Fj452\_VGLipJnL&index=15

https://www.geeksforgeeks.org/the-cap-theorem-in-dbms/

Eventual consistency (Example: DNS servers)

BFF:

https://blog.bitsrc.io/bff-pattern-backend-for-frontend-an-introduction-e4fa965128bf

CDN Servers (Pull and Push CDN):

https://www.youtube.com/watch?v=rwBv7FqZ77g&list=PLrtCHHeadkHp92TyPt1Fj452\_VGLipJnL&index=19

Load Balancer (Round Robin, Least Connections, Consistent Hashing):

https://www.youtube.com/watch?v=chyZRNT7eEo&list=PLrtCHHeadkHp92TyPt1Fj452\_VGLipJnL&index=22

https://blog.hubspot.com/website/api-gateway-vs-load-balancer#:~:text=An%20API%20gateway%20vs.,network%20traffic%20across%20multiple%20servers.

Delegates and Events (Func, Predicate & Action):

https://www.c-sharpcorner.com/UploadFile/84c85b/delegates-and-events-C-Sharp-net/

https://www.c-sharpcorner.com/article/func-action-predicate-delegates-in-c-sharp/

Caching:

https://www.youtube.com/watch?v=6GY1akbxyEo&list=PLrtCHHeadkHp92TyPt1Fj452\_VGLipJnL&index=18

SQL and No-SQL:

https://www.youtube.com/watch?v=\_Ss42Vb1SU4&list=PLrtCHHeadkHp92TyPt1Fj452\_VGLipJnL&index=29

ACID Properties:

https://www.geeksforgeeks.org/acid-properties-in-dbms/

Saga Patterns (Choreography, Orchestration):

https://microservices.io/patterns/data/saga.html

https://www.youtube.com/watch?v=C0rGwyJkDTU

https://www.youtube.com/watch?v=HiwOx-W1TIA

Circuit Breaker Pattern (No transient failures, Resilient Systems, Cascading Failures, Fault Tolerance):

https://www.youtube.com/watch?v=HRS9mIfiNn4

Bit wise manipulation:

https://www.geeksforgeeks.org/bitwise-hacks-for-competitive-programming/

Immutable and mutable (primitive types, string, string builder)