***Cqrs & Event sourcing***

***CQRS***

* Is a software design pattern that stands for command Query Responsibility Segregation
* Commands: operations that alters the state of an object or entity
* Operations that return the state of an object or entity

***Benefits***

* Allows us to scale up command and query sides independently from each other
* This is a great advantage for example in systems where the reads outnumbers the writes
* Separating commands and queries allows us to optimise each for high performance
* Executing command and query operations on the same model could cause data contention
* Read and write representations of data generally differs substantially
* Separation provides the ability to manage command and query security and permissions differently

***Event sourcing***

Defines an approach where all the changes that are made to an object or entity, are stored as a sequence of immutabl events to an event store, as opposed to storing just the current state

A screenshot of a computer

AI-generated content may be incorrect.

***Saga pattern***

What Is a distributed transaction?

Rememeber the purpose of a database transaction is to guarantee an all or nothing outcome and comply to all four ACID properties

Is a database transaction that involves 2 or more networks hosts (microservices)

What is the Saga pattern?

Design pattern that provides a solution for implementing transactions in the form of sagas that span across two or more microservices.

Saga sequence of local transactions