# Normal Application

**Github Url:** <https://github.com/humayun-ahmed/OnlineCourse.git>

Requirement Covered: Part 1: API for signing up

## Technologies Used

1. Asp .Net Web Api
2. Unity Container for dependency injection
3. EntityFramework
4. SpecExpress for fluent validation
5. Microsoft SQL Server
6. Swashbuckle for api documentation
7. NLog for logging
8. AutoMapper
9. Moq, NBuilder, MSTest for unit test
10. Effort.EF6 for in memory data store for integration test
11. Angular for test

## Architecture

# CQRS Application

**Github Url:** <https://github.com/humayun-ahmed/CQRS.git>

Requirement Covered: Part 2: Scaling out, Part 3: Querying

## Technologies Used

1. Asp .Net Core
2. EntityFramework
3. Microsoft SQL Server
4. Swashbuckle for api documentation
5. Rabbitmq for store or pass messages
6. MassTransit framework for message to make the bus

## Architecture

1. Send a command to CommandServer
2. CommandServer will store that command to rabbitmq
3. Used the same CommandServer to pull the command and handler
4. In CommandServer, it handles the command and publish event
5. ReadModel.EventSubscriber listen the events and update the read database according to the requirement
6. QueryServer reads the data.

According the CQRS pattern, instead of event source, domain data are storing in sql server due to time limitation of implementation

## A screenshot of a cell phone Description automatically generated