Tech Stack Used :

+.Net Core MVC -Razor -Logging -Dependency Injection

+.Net Core Restful Web API

- Ajax FrameWork

+SQL Server DB

- Tables
- Triggers
- Stored Procedures
- Functions

+Entity FrameWork "DB First Approach"

- just used as a middle layer between the application and DB for the sake of calling to stored procedures , Functions

+Design Patterns Used "MVC , Onion design Pattern for build DB transactional library" which expose the effecient use of OOP

Separation Of Concerns

Though the app is very simple but i tried to put more than what's requested to expose knowledge in different areas as Next :

DB Tier :

+SQL Server DB

- Tables MineSweeper

  ```
    (to store the input to be processed later , no. of times requested for
  processing , Audit data "Added Date , Modified Date , IP Address" )
  ```

- Triggers ValidateInput

  ```
    (before insert or update to make sure the input containt '0 0' as latest 3
  characters)
  ```

- Stored Procedures SP_MineSweeper_GetMineSweeperInput

  ```
    (will be called by the application tier to return the input data to be
  processed , call SP_MineSweeper_InputProcessCounterUpdate to increment
     no of times data requested for processing)
  ```

  call SP_MineSweeper_InputProcessCounterUpdate

  ```
    (to increment no of times data requested for processing)
  ```

- SQL Functions FN_MineSweeper_InputCount

```
  (iterate over input via cursor and return total number of inputs to be
processed,exact as built in Count Funvtion
   but made to expose knowledge in creating functions and calling them from the
application tier)
```

Middle Tier (Between the App./WebApi and the DB) :

+MineSweeper.Lib project

it's a separate library project that is using

> *Onion Design pattern "Model, Repository, Unit Of work" to implement proper transactional interface between the DB Tier and the App tier which expose a good experience in OOP*

> *Entity FrameWork Core "DB First Approach"*

- just used as a middle layer between the application and DB for the sake of calling to stored procedures , Functions
- Even the Data Queries will not be done via queries on tables using EF Core but will be done via the communications between EF Core and Stored Procedures , DB Functions

API Tier & Web Application Tier

+MineSweeper .Net Core MVC project Contain both "Restful Web API , .Net MVC project" we are using (Logging , Service providers for the middle tier and Dependency Injection , Auto Mapping between Models,.....)

API Tier :

> *MineSweeperAPIController which expose Restful Web API to be consumed by the ajax calles in the .net pages later. Restful Web API is a resource based as known so we Handled "GET/POST/PUT/DELETE" on MineSweeper resource on these API plus it exposed a restful interface /minesweeper_inputcount to call the SQL Fn. of getting input count via the Middle Tier Library*

Web Application Tier :

> *MineSweeperInputList.cshtml "to list previously entered input" , MineSweeperInputCreate.cshtml "to create new input" UI : using razor syntax for display rendering the UI only Data manipulation : the Data manipulation is handled between interactions between WEB API Tier and Ajax FW calls the ajax functions defined in Site.JS file which is globally imported to all pages*

> *Index.cshtml "to list the input and the calculated output of the mine fields" UI and Data manipulation : MVC "HomeController.cs , razor views , Model DTOs to provide security for posting the data on the application which are mapped using auto mapper"*

> *Business Logic is separated in a separate class PTL.MineSweeperLogic/MineSweeperBL.cs*

Testing Tier

+MineSweeper.xUnitTests project

> *Is a .Net Core xUnit Testing project have testing cases for the MineSweeper.Lib project Testings for the API done via PostMan , so only samples of testing put on the MineSweeper.xUnitTests project*

Steps to run the project :

SQL Server : -Create Schema called ProjectsSchema -open and run "MineSweeperScript.sql, data.sql"

Visual Studio : Open the soltion > hit run on IIS

Screen Shots : Explore the included screenshots to :

> *create input "must be valid - end with 0 0" list the input , audit data , no of times it's requested for processing home page contains the input and calculated output*

Note : there were no focus on the styling of components , to have time to expose knowledge in other areas