WEB API – FINAL PROJECT

PART I

1. Create a web tournament DataBase for online games

   Game table:

ID	Game_Name	Player1	Player2	Who_Won?

   Create it in MSSQL (using query)
   - Use Entity Framework

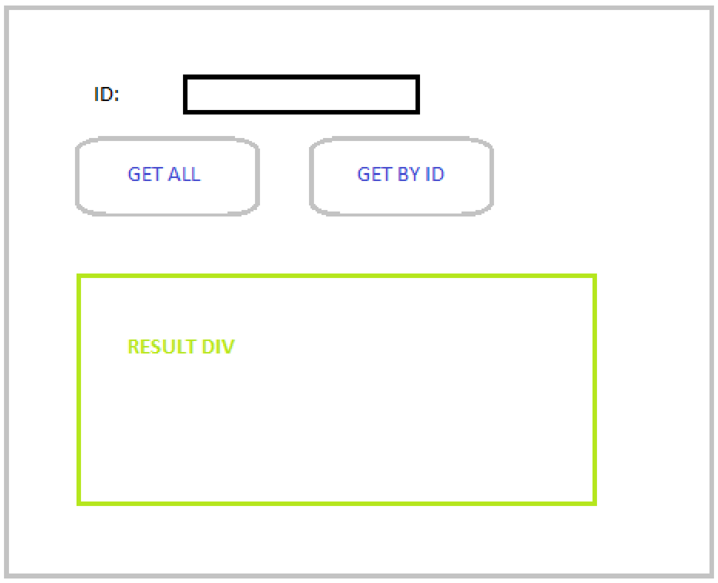
2. Create a WEB API with ApiController which allows GET/ POST/ PUT/ DELETE
   a. Implement GET/{ID}
   b. Implement GET/{Player}
   c. Implement GET ? ID & GAME_NAME & Player1 & Player2 & Who_Won
3. Test the API using **Postman**
4. Test the auto generated **API** (from VS 2017 site)
5. Create a **Swagger** documentation

PART II - XMLHttpRequest

Create a Controller, call it PageController which will redirect the user to the requested page. All the pages should be uploaded to the WEB API project

- Bonus: use Bootstrap buttons in the page
- Bonus: add nav-bar to browse between the pages

6. Create a page which performs GET and GET/{ID} by clicking a button,
   This page will have 2 buttons and a text-box:
   - Button 1 will GET ALL of the games and displays the results into a DIV.
   - Button 2 will GET the game which his ID is taken from the text-box and displays the results into a DIV.



7. Create a page which performs POST by clicking a button, and allow the user to enter the data into text boxes / radio buttons/ etc
8. Create a page which performs PUT+DELETE by clicking a button, and allow the user to enter the data into text boxes / radio buttons/ etc
   This page will have 3 buttons:
   - Button 1 will GET the data from the server (the user will enter id in the text box) The data will be brought from the server and inserted into the text boxes
   - Button 2 will PUT the data to the server
   - Button 3 will DELETE the game (it will delete the game which the id is the one appearing in the text box)

PART III - $. AJAX

Duplicate all of the pages from PART II, upload them to the server.
Modify the AJAX to use $.AJAX (JQuery) instead of XMLHttpRequest

PART IV – Observables RXJS

Duplicate only the Get page, upload it to the server.
Modify the GET to use Observables in RXJS

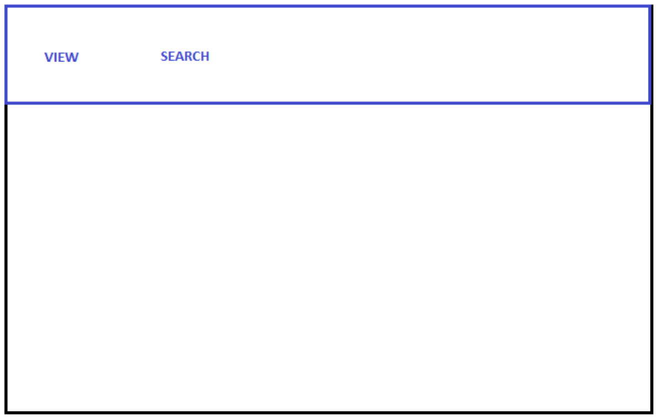
PART V – Testing

Duplicate only the Get page, upload it to the server.
Mock the Get all method using
- Promise
- NODE server WEB API
    - Test empty result
    - Test large result
    - Test error message

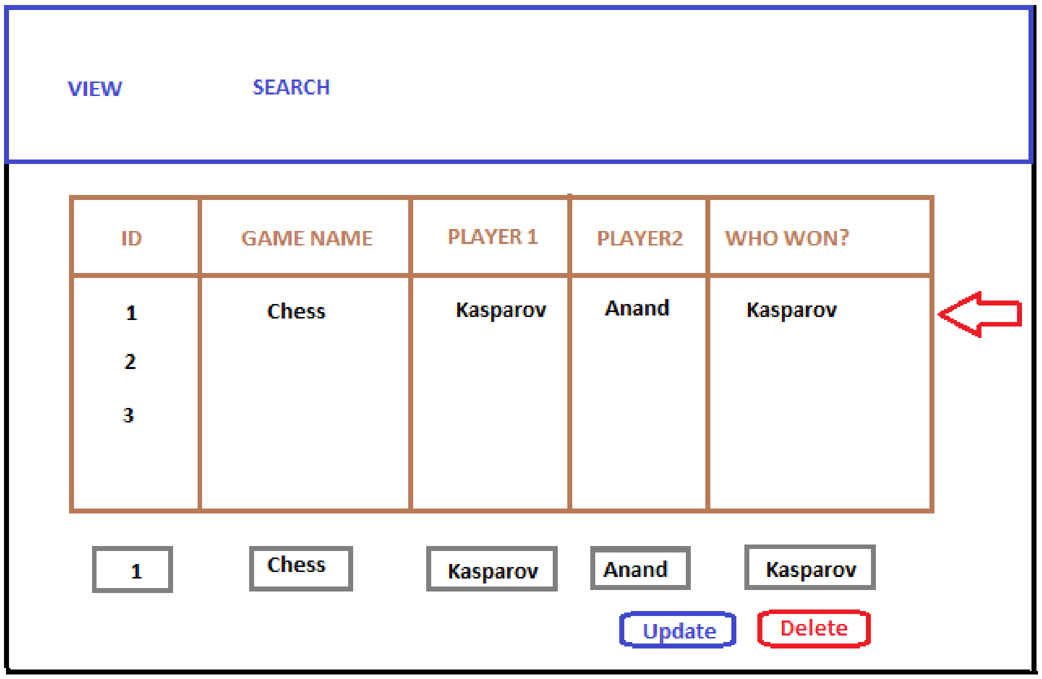
PART VI – Angular

- Don't forget to open CORS and preflight…

Create an Angular application with two options menu: VIEW and SEARCH (using ROUTER)

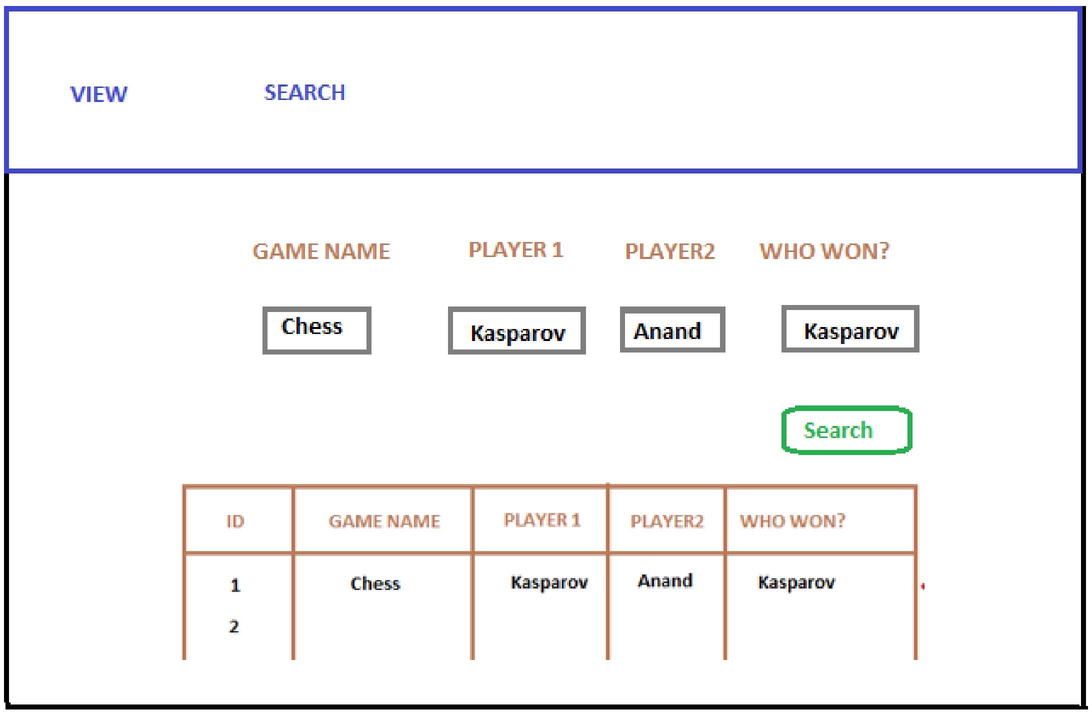


VIEW OPTION



- All the data will be displayed in a Table.
- Each click on a table row will copy the data into text-boxes below
- The text-boxes are editable.
- When user clicks Update it will send the update to the server
- When the user clicks Delete it will send delete to the server

SEARCH OPTION



ID	GAME NAME	PLAYER 1	PLAYER2	WHO WON?
1	Chess	Kasparov	Anand	Kasparov
2				

- This time the user will enter search data into the text boxes and click Search. It will send query parameter search and return a set of results. The results will be displayed in a table below

Notes:

- Use at least 2 components
- Use a Service for shared data
- Use a service for WEB API calls
- Bonus: add bootstrap