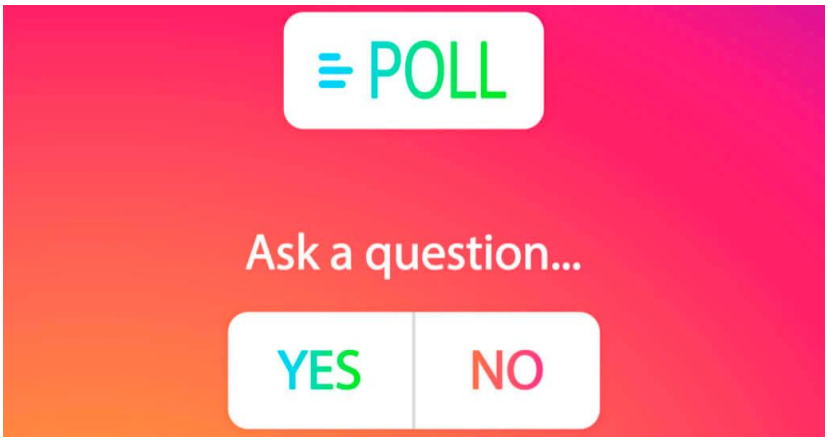
Polls System - Fullstack Implementation

Instructions:

1. Your mission is to create a poll system that will be used to ask the company customer's any questions they want and process calculations about the data.



Example for questions:

- Between the following, what do you most love to do?
  a. Watch TV
  b. Play the computer
  c. Hanging out with friends
  d. Travel the world

- Where is your preferred place to travel
  a. USA
  b. France
  c. South America
  d. Thailand

2. Your system should implement two different services:
   a. First service → **User REST service**
   b. Second service → **Questions REST service**
3. Let's understand what each service should be responsible for:

The Users REST API service → will be responsible to save all the data about your system users. Such data will be:

- user id (PK)
- User first name
- User last name
- User email (Unique)
- User birthday
- User address

Only user can submit answers to any questions.
If a user is deleted from your system, all the answers he gave to any question should be deleted as well.
- Hint: REST Service User – delete should delete all of his answers

You should support full CRUD implementation for your system users meaning you will need to provide API that allow create a user, update a user, delete a user and get user info (by-id), get all users.

The questions REST API service → will be responsible to store all the questions and possible answers.

Each question will be an American question which mean the user should get the question text and 4 optional answers. The user should choose the answer he wants from the 4 available options and your system should save the user choice accordingly.

For each question your system should save the following data:

- question id (PK)
- The question title (Here you should save the question itself)
- The question first answer option
- The question second answer option
- The question third answer option
- The question fourth answer option

You should support full CRUD implementation for the questions meaning you will need to provide API that allow create a question, update a question,

delete a question and get question info- get all or by id.
If a question is deleted from your system, all the answers should be deleted as well.

The poll REST API service → will be responsible to store all the answers for each question that the users answered.
Remember that it's possible that not all users will answers to all the available questions and it’s also possible that will be some questions without any answer.

- Answer number (PK)
- User id (FK)
- Question id (FK)
- Date-Time of answer
- Selected answer

Make sure you do not store 2 answers for the same question from the same user.
(*Bonus: hint: User-id+Question-id combination should be Unique)

Finally, your system should give the company the ability to get relevant info from the users answers to each question meaning you should support API that implement the following:

- By passing the question id → Return how many users choose each of the question options, and the question itself + possible answers
- By passing the question id → Return how many users answer to this question in total, and the question itself
- By passing the user id → Return the user answer to each question he submitted, and the user-name and the title of each quesiton
- By passing the user id → Return how many questions this user answered to, and the user name
- Return all questions and for each question how many users choose each of the question options, and the question title, and each option.

4. Add HTML page(s) for the Poll (which performs AJAX for the REST APIs)
    a. need the user-id (input type number where user write his id)
    b. present the question
    c. present 4 answers (radio button)
    d. send the user-id + question-id + answer to the REST api

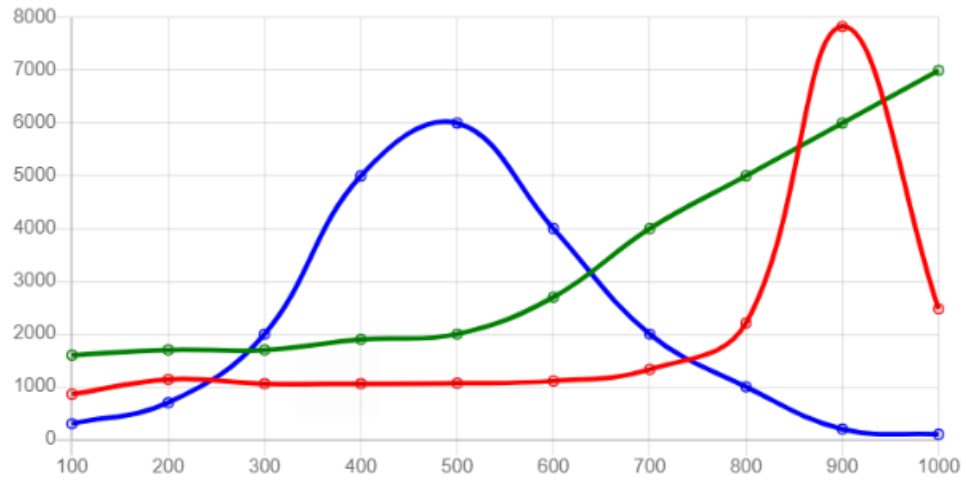
    *Bonus: make sure you do not submit the same question twice for the same user

    **Bonus: check if this user has questions he did not answered

5. Add HTML page(s) for the Statistics which performs AJAX for the REST APIs
    a. Display all the statistics options
    b. Make a button for each option – clicking it will display the result

    **Bonus (crazy): use chartjs

    help link: https://www.w3schools.com/ai/ai_chartjs.asp



6. Add data to the DB using Postman

    Supply screenshots (at least 1 per REST API).
7. REST API should have create-table, delete-table, insert (i.e. insert6)
8. Add logs
9. Add 1 test per REST API (for router or dal)
10. Add Swagger
11. Add comments to your code
12. Make sure the REST API returns correct HTTP status code
    a. Use try - catch

13. Make "start" , "test" scripts
14. Make the code readable 😊
15. Upload your code to Github
16. Don't forget .gitignore

Good Luck

ECOM SCHOOL

המכללה למקצועות הדיגיטל וההייטק