Survey Application

Problem :

Design the back-end for a survey application: It should give the capability for simple polls comprising questions with multiple choice answers

Entity design

- Survey Entity:
    1. Id (long)
    2. Survey Name (text)
    3. Created User
    4. Expiring Date
    5. List of Questions
        a. This will be a one to many mapping to Questions Entity

- Questions Entity
    1. Id (long)
    2. Question (text)
    3. QuestionType (Enum : MCQ, TEXT, TRUE_OR_FALSE)
    4. List<Options>
        a. One to many mapping form question to Options
        b. This will store the values of , mcq, and boolean questions

- Options Entity
    1. Id(long)
    2. Value(text)

- Anwer Entity
    1. (SurveyId, QuestionId, UserId) composite Key
    2. Answer (text) (will save all answers as text for now)

Important Rest Endpoints:

```
Submit survey
POST /api/surveys/{surveyId}/submitSurvey
{
    "userId": 0,
    "answerList": [
      {
        "questionId": 0,
        "answer": "string",
        "userId": 0
      }
    ]
}


GET all answers of a question
GET /api/surveys/{surveyId}/questions/{questionId}/answer
{
    "question": {
      "id": 0,
      "questionType": "string",
      "question": "string",
      "optionsList": [
        {
          "id": 0,
          "value": "string"
        }
      ]
    },
    "answers": [
      {
        "questionId": 0,
        "answer": "string",
        "userId": 0
      }
    ]
  }
```

```
GET All questions of a survey
GET /api/surveys/{surveyId}
  {
    "id": 0,
    "surveyName": "string",
    "createdUserName": "string",
    "expiringDate": "string",
    "questionList": [
      {
        "id": 0,
        "questionType": "string",
        "question": "string",
        "optionsList": [
          {
            "id": 0,
            "value": "string"
          }
        ]
      }
    ]
}
```

Rest of the endpoints can be referred in https://github.com/prasad14/survey/blob/main/src/main/java/com/example/survey/rest/SurveyController.java

Swagger document in generated in http://localhost:8080/swagger-ui/index.html#/ if the application is started locally.

Enhancements TODO

- We could use **Elasticsearch** to store the answers.(in scaling aspects where millions of people are responding to the survey)
- The survey and questions will be stored in Database only. Since the amount of data saved will be less. Further , we can use local caching to enhance the performance of these APIs

- Elasticsearch will allow the data search to be simplified and it in build feature of aggregation can be used to get the relative distribution of answers in a MCQ or true/false question
- Elasticsearch index design
    - One index will be created for one survey
    - The id of the document will be surveyId + questionId + userId
    - User id is important to distinguish from various answers , we can chose to add user email or other fields to distinguish them later
    - Document structure

```
put survey_name/_doc/{id}
{
  "surveyId":"",
  "questionId":"",
  "questionType":"",
  "userId":"",
  "answer":""
}
```

    - QuestionType is added to check with the answer is MCQ or not
    - We can easily fetch answers of one survey by in a simple query specifying the survey Id

```
{
    "query" : {
        "term" :{
            "surveyId" : 123;
        }
    }
}
```

        - We can further do terms aggregation on the anwer field so that we could get a count of set of unique answers , easily determine the relative distribution of responses

- If we have a cluster of Elasticsearch , We can store the data consistently and scaling the elastic nodes is easier to accommodate more data.