LearnToWin FlashQuiz App
ReadMe Doc

The folder comprises of:

- ***'quiz.json':*** The file consists of data in json format used in application. (List of Question Objects)
- ***'Frontend':*** Folder consists of frontend code in ReactJS
- ***'Backend':*** Folder consists of backend code in Spring-boot Java
- Screen recording of demo
- ReadMe doc for whole process description.

Requirements

- IntelliJ IDE for running backend code
- Maven dependencies
- NPM package manager for react
- Visual Studio Code / Sublime Editor for frontend code
- Terminal (to run *npm* commands)

Folder Structure

Backend:

- *./src/main/java/com/example/FlashQuiz*
- Main file: ***FlashQuizApplication.java***
- App runs from ***InitialDataPusher.java*** which parses all the data and stores in JPA Repository
- Consists of packages:
  - controllers: Rest Controller for REST APIs
  - models: Database model of question object
  - parser: Contains InitialDataPusher.java
  - repositories: Interface for JPA repository
  - services: Handles functionalities for querying data from and to repository

Frontend:

- Run *'npm install'* to install required dependencies for frontend to run.

./src folder:

- *App.js* is the main class for application from where code flow starts
- Consists of folders:
  - ***Backend:*** folder consists of network call files (api calls from ReactJS to REST APIs)
  - ***Components:*** Class components used in application (Welcome page, question page)
  - ***CSS:*** Styling code

Algorithm:

- Parsed data present in ***quiz.json*** file, and allotted random time to each question in the range of 25 secs to 50 secs.
- Every question has the properties:
  - *id, question_text, option1, option2, option3, answer,*
  - *time it is created,* (randomly assigned from 2020-03-01 to 2021-03-21)
  - *time allotted to answer,*
  - *frequency its shown, and incorrect number of attempts.*
- All the questions are then stored in database with its frequency and incorrect attempts set to 0 initially.
- All the questions will then inserted in priority queue in order of their difficulties, frequency shown and number of incorrect attempts.
- **Formula used for comparison for two objects,**

  if object frequency = 0, ***score1 = score1 * timeAllotted***

  else if incorrect attempts of object = 0, ***score = score * timeAllotted / frequency shown***

  else, ***score = score * timeAllotted * IncorrectAttempts / frequency shown;***

- Once the api /getQuestion is hit from client, top question in priority queue will be shown.
- Once the question timer ends or it is attempted from client, api /getQuestion with params (answered, timer) will be hit.
  - answered: Signifies if question is answered or not
  - timer: Signifies if question timer is end or not
- Then, priority queue current top element will be removed, its properties will be changed based on above api call and again will be inserted in priority queue.
- The above process will be repeated until quiz time ends.
- Algorithm would take O(log n) time (where n is number questions shown within quiz time frame)