Interview Question

Please finish and submit the following project in the required time.

Question Description:

One professor would like to have a web application where he can create two courses for 4 students, and students can leave reviews on each course so that the professor can check feedback from students.

Please 1. write down in a report what backend APIs should be offered to frontend to achieve the functionalities.
2. fulfill the backend functionalities in your code which can be run in IDE.

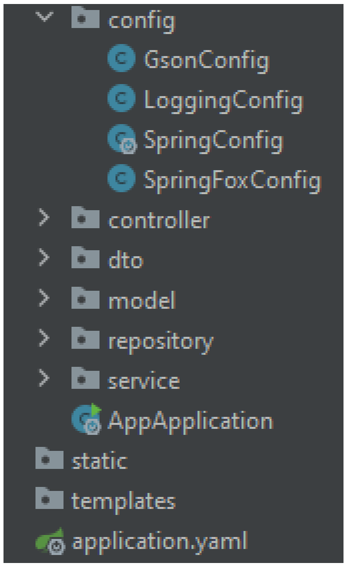
Requirements:

1. Courses names: Computer science, Data science
2. Student names: Jack, Bill, Cindy, and Anna
3. Course introduction functionality: the professor can use this functionality to edit the class descriptions and class time. Note that he can CRUD class descriptions and class time.
4. Review includes two functionalities: rating and feedback. Students can use rating functionality to rate class quality from number 0 - 10. Students can use feedback functionality to write down messages in text regarding the class quality.
5. Students can CRUD only their own feedback information only after the class is created.
6. Professor cannot see who leaves the feedback and cannot edit any feedback and ratings, but can find the feedback with a created timestamp.
7. Professor can sort the feedback descending or ascending order based on the created timestamp.
8. Professor can check the average rating from all students
9. Professor can use a filter to get the description with Key words “good” and “bad”.
10. Professor can reply to feedback with no more than 500 words.

Project Specific Requirements:

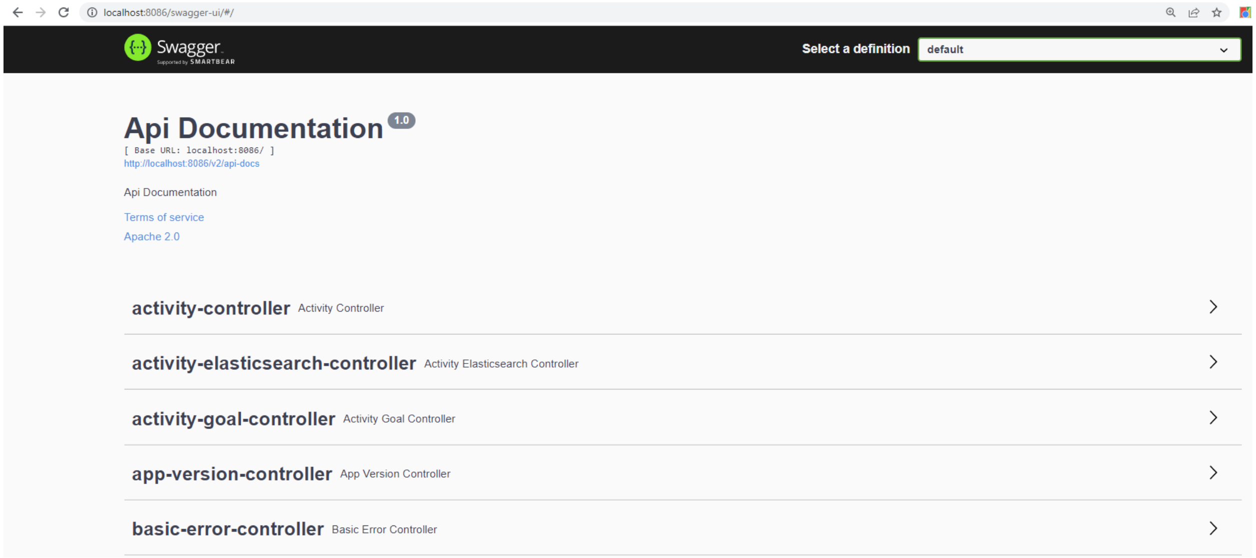
1. Generate a Gradle Project with Java and Spring Boot framework (version 2.7.3).
   1.1 Group name: com.vibrant
   1.2 Artifact name: demo
2. Database: use H2 in-memory database
3. To implement the functionality of each feature, you are required to have following **5-package-component** project structure (refer to the picture below)
   3.1 controller → process RESTful API requests and responses

3.2 dto → predefine the transferred data format for passing the properties in RESTful API
3.3 model → define the java objects in persistence layer (H2 database)
3.4 repository → define JPA interface to for java objects persistence between service and model
3.5 service → develop the business functionalities



```
  ⌄ config
        GsonConfig
        LoggingConfig
        SpringConfig
        SpringFoxConfig
  > controller
  > dto
  > model
  > repository
  > service
    AppApplication
static
templates
application.yaml
```

4. Configure your Spring Boot project in config package (refer to picture above) and yaml file. You have to configure Swagger in SpringFoxConfig java file to test and consume RESTful APIs (similar to Postman). GsonConfig, LoggingConfig and SpringConfig files are optional (you can decide whether or not to configure them as long as the project can be normally run and fulfill requirements)
   4.1 GsonConfig → Gson is a Java serialization/deserialization library to convert Java Objects into JSON and back. Gson was created by Google for internal use and later open sourced. (This configuration is optional)
   4.2 LoggingConfig → Configure Log4j
   4.3 SpringConfig → Configure project model packages for basePackage scan
   4.4 yaml file → configure project settings such as server port, database connection parameters, etc.
   4.5 SpringFoxConfig → configure swagger for API testing (refer to picture below http://localhost:Your Port Number/swagger-ui/#/)



Select a definition: default

Api Documentation 1.0

[Base URL: localhost:8086/]

http://localhost:8086/v2/api-docs

Api Documentation

Terms of service

Apache 2.0

activity-controller Activity Controller

activity-elasticsearch-controller Activity Elasticsearch Controller

activity-goal-controller Activity Goal Controller

app-version-controller App Version Controller

basic-error-controller Basic Error Controller

5. To implement JPA, use Spring Data JPA dependency
6. In your project, please pick one API to implement the Junit test. It is required to have 3 test cases for the according API which can demonstrate the developed API passes all 3 test cases.