Question Answer Platform

OVERVIEW

We need to build backend APIs for a question answer platform, where questions and answers for interviews of various companies are being captured. No UI development has to be done for the scope of this project.

ENTITIES

This platform would have the following high level entities, with following relationship among them. Broadly 1 entity should correspond to 1 table in system

1. Question - Core entity of the platform
2. Company - A company for which question was asked, Eg: Amazon/Microsoft
3. Topic - Topic for which a question was being asked eg: Data Structures, Algorithms
    a. Subtopic - A topic can have various subtopics. Eg: For topic Data Structures, it can have subtopics like Stacks, Queues, Lists etc.
4. Answer - A questions can have multiple answers
5. Answer_comments - Users can comment on answers, hence an answer can have multiple comments
6. Question_likes - User have like a question
7. Question_company_mapping - A question can be asked for many companies. For example, the same question may be asked by Amazon and Microsoft. To capture this information, we need to have this entity.
8. Answer_likes - Users may like an answer to a question. This data would be captured in this table.
9. Tags - Not all information can be captured using topics and subtopics. For example while subtopics can be Stacks, Queues and Lists, master list of tags may contain complexities of algorithms which could be O(n), O(logn) etc.
10. Question_tags - A Question can be associated with multiple tags. This table will contain that information
11. User - User who is posting question/answer/likes etc.

APIs

For scope of this assignment, we need to build following APIs

1. CRUD APIs for all master data, which includes following entities/tables
    a. Company
    b. Topics
    c. Subtopics
    d. Tags
2. /POST Questions
    a. Input
        i. Question Text - Mandatory. Min length 50 characters, Max length 500 characters
        ii. CompanyId - Optional
        iii. SubtopicId - Mandatory
        iv. Tags - List of tags - Optional
        v. User ID - the Id of user submitting the question
    b. Output
        i. ID - ID of question saved in database
3. /POST Answers
    a. Input
        i. Question ID - Question for which answer is being submitted
        ii. Answer Text - Mandatory. Min length 50 characters, Max length 500 characters
        iii. User ID- the ID of user submitting answer
    b. Output
        i. ID - ID of answer saved in database
4. /POST Comments (Post comments for a given answer)
    a. Input
        i. Answer ID - Answer for which comment is being posted - Mandatory
        ii. UserID - User posting the answer - Mandatory
        iii. Comment Text - Comment Text - Mandatory. Min length 50 characters, Max length 500 characters
    b. Output
        i. ID - ID of comment saved in database
5. /POST Question_likes (For a user to like a given question)
    a. Input
        i. QuestionID - Question which is being liked

        ii. UserId - ID of user liking the question
6. /POST Answer_likes (For a user liking the given answer)
    a. Input
        i. AnswerID - Answer which is being liked
        ii. UserId - ID of user liking the answer
7. /GET Filter_Questions - Returns list of questions based on the combination of following query parameters
    a. Input
        i. Companies - List of companies - Optional
        ii. Subtopics - List of subtopics - Optional
        iii. Likes - Questions with more votes than votes sent as input - Optional
        iv. Date - Questions asked after a given date - Optional
        v. Tags - List of tags - - Optional
    b. Output
        i. Question ID
        ii. Question Text
        iii. Companies - List of companies for which the question was asked
        iv. Likes - Number of likes of the question
        v. Answer - Answer with most likes for the given question
        vi. Tags - Tags associated with the given question
8. /GET Question - Get following details of a question for a given question id
    a. Question Text
    b. List of answers for the given question with following details
        i. Answer text
        ii. User Id of user who answered the question
        iii. Number of likes for the answer
        iv. List of comments of the answer, along with date and user who made the comment
    c. Number of likes that the question has
    d. Companies for which the question was asked
    e. Tags for the question
    f. Topics for the question

SPECIFICATIONS

1. For Java backend the code should be structured into following layers
    a. Model - Mapping of tables in DB to Java objects
    b. DAO - Defining how to fetch data from DB and map to Java objects
    c. Service - Business logic
    d. Controller - Rest APIs should be defined here
2. After completing the code,
    a. Share the github URL where the code is being written for review
    b. Setup the working APIs on a cloud environment, preferably AWS (use aws free tier).
    c. Share a documentation of APIs, so that one can test the APIs
    d. Will have a discussion in detail on video call to make sure that you understand the code well and assignment is not done by someone else

Technology choices

1. Java
2. Spring / Spring Boot
3. JPA/Hibernate
4. Maven
5. Mysql/Postgres (Open source relational DB)
6. AWS (preferable)