**Trello\_quora - Course 5 SOCIAL Q&A Group Project**

1. In this project, we will work on developing REST API endpoints from scratch. We should be using the **Swagger** user interface and store the data in the **PostgreSQL** database, and also the project has to be implemented using **Java Persistence API (JPA)**.
2. This is a group project, we would be working in a group of 4 students.
3. Use Git and GitHub to conduct version control of your assignment code throughout your assignment development.
4. One of the team members should act as the project lead, create a master repository for the project, and push the initial code stub to the master repository. After which, the project lead would create different branches for different functionalities to be developed, and share the repository URL with other team members.
5. The other team members should then fork and clone the master repository to their own repository on GitHub, so they can work on a specific branch and make updates on the project via pull requests. Also, it would be the project leader's responsibility to merge the pull requests into the master repository.
6. Each member of the team to review a pull request before it is merged into the master repository and give your comments on the pull request to help the project leader.
7. Lastly, if one is working off a fork, he/she should fetch from the upstream repository often, so he/she can get the latest commits and updates of the various branches in the upstream repository.
8. Once all the required code implementation is done on a specific branch and is working fine, then the project lead can go ahead and merge the branch with the master repository.

**EVALUATION CRITERIA**

1. The code for the assignment compiles and produces no error when executed. The entity classes include all the attributes with proper annotations and their relationships as per the given database design. Controller methods are created for each of the API endpoint listed in the project description.
2. The REST API endpoints fulfil all the conditions given and perform the required functionality. Authentication and authorization of the endpoints are implemented as per the given instructions. Separate services and DAO classes are created for each functionality. The DAO must communicate with the database.

1. The service classes must contain the required business logic and interact with the DAO classes for any change required in the database. Exception Handlers for all the exceptions are implemented.
2. **The student uses Git and GitHub to conduct version control on his/her assignment code.**

**The student makes small, incremental commits.**

**The student follows the instructions provided for the group collaboration.**

**The student writes clear and concise commit messages.**

1. The code passes all the unit tests written for the assignment. The code is formatted correctly; it uses the right spacing and indentation and follows the formatting guidelines laid out in the Google Java Style Guide.
2. Variables, classes, and function names follow the appropriate upper and lower camelCase.

Classes, functions, and variables are named after the contents or functionalities that they represent.

The code contains good comments that explain how the complicated portions of the code work.

**MARKS WILL BE PENALISED FOR ALL TEAM MEMBERS IN THE FOLLOWING SCENARIOs**

1. The code produces compilation errors or run-time errors when executed.
2. The entity class and its attributes are not created as per the given database design or the proper annotations are not used.
3. Controller methods are not created for each of the API endpoint listed in the project description.

 The REST API endpoints neither follow the given conditions nor perform the required functionality.

 Authentication and authorization of the endpoints are not implemented as per the given instructions.

1. Separate services and DAO classes are not created. The project interacts or accesses the database outside of the DAO classes.

The business logic is not implemented in the service classes and service classes do not interact with DAO classes for any change required in the database

1. Exception Handlers for all the exceptions are not implemented.
2. **The student does not use Git or GitHub to conduct version control on his/her assignment code.**

**The student makes big commits that contain multiple features or bug fixes.**

**The student does not follow the instructions provided for the group collaboration.**

**The student writes short or unclear commit messages.**

1. The code does not pass all the unit tests written for the assignment.
2. The code is not formatted correctly. Extra spaces, line breaks, incorrect indentation, and bad formatting are used throughout the code.
3. Variable, classes and function names do not follow appropriate upper and lower camelCase.

Classes, functions, and variables are not named after the contents or functionalities that they represent.

The code does not contain any comment, or it contains poor comments that do not properly explain how the complicated portions of the code work.

**DEADLINE FOR FINAL SUBMISSION: 15th May 2019**

**GitHub Collaboration Instructions**

1. One of the team members should act as the project lead.
2. Project should create a master repository for the project, and push the initial code stub to the master repository.
3. Project lead would create different branches for different functionalities to be developed, and share the repository URL with other team members.
4. The other team members should then fork and clone the master repository to their own repository on GitHub, so they can work on a specific branch and make updates on the project via pull requests.
5. Project lead is responsible to merge the pull requests into the master repository.
6. It is always a good practice for each member of the team to review a pull request before it is merged into the master repository and give your comments on the pull request to help the project lead in accepting the pull request. Reference <https://help.github.com/en/articles/syncing-a-fork>
7. If you are working off a fork, don't forget to fetch from the upstream repository often, so you can get the latest commits and updates of the various branches in the upstream repository.
8. Once all the required code implementation is done on a specific branch and is working fine, then the project lead can go ahead and merge the branch with the master repository.

Additional notes to help team collaboration:

* Use GitHub to track issues and bugs for the project. Reference <https://guides.github.com/features/issues/>
* Use GitHub to conduct code reviews, so each push request or commits are reviewed by another teammate before the code changes are merged into the main repository. Reference <https://github.com/features/code-review>