SDPVocabQuiz: An app for sharing and practicing vocabulary quizzes (Deliverable 2–Inception and Elaboration)

Background

One of your professors is looking for a way to help students study vocabulary terms for their classes and challenge one another. Knowing that you and your team members are expert developers, she asked you to develop a program that students can use on a locally shared (Android) tablet to (1) create and share vocabulary quizzes and (2) compare their progress. After reviewing your design, she has requested that your team continues to develop this app for the Android OS following a Unified Software Process model.

Goals

- Develop an Android app that implements the vocab quiz app that your professor envisioned (and that your team already designed).
- Get experience with (a simplified version of) the Unified Software Process.

Requirements

See the requirements in Assignment 5.

Your professor has also supplied some additional clarification for some of the requirements, based on discussions with the students in your team and in other teams:

1. Requirement 7 (quiz score statistics):
    a. The quiz score statistics for a student S should list *all* quizzes, whether they were played by S or not, and including the quizzes created by S.
    b. The quizzes not played by S can be displayed in any order (after the ones played).
    c. For quizzes not played by S, only the names of the first three students to score 100% on the quiz should be displayed.
    d. The names displayed (and used to sort) in the statistics for the first three students to score 100% on the quiz can be either their usernames or their real names.
2. Requirement 6 (practicing a quiz):
    a. Every word in a quiz should be shown once and only once.

    b. Incorrect definitions, conversely, may repeat.
3. General clarification: all relevant data (scores, statistics, quizzes, student logins) should persist between uses of the application.

Deliverables

- ~~D1 – *Software design*~~ **[done]**
- D2 – *Inception and Elaboration* (see corresponding lessons): **[due this week]**
    - Project plan (**`ProjectPlan.md`**), showing **project development process** and roles (see template).
    - Use-case model (**`UseCaseModel.md`**) (see template).
    - Supplementary requirements (i.e., requirements that do not fit the use-case model, such as non-functional requirements). These can be provided as a simple list, but you must include this file. Put this in an MD file called **`ExtraRequirements.md`**
    - Design document (**`DesignDocument.md`**) (see template).
    - Test plan without results (**`TestPlan.md`**) (see template).
- D3 – *Construction* (see corresponding lesson): **[not due yet]**
    - Possibly revised earlier documents (based on your better understanding of the system).
    - Initial version of the app (think of this as an alpha/beta release).
    - Test plan **with (possibly partial) results**.
    - User manual. Put this in an MD file called **`UserManual.md`**
- D4 – *Transition* (see corresponding lesson): **[not due yet]**
    - Possibly revised versions of earlier documents (based on your better understanding of the system).
    - Test plan **with final results**.
    - Final version of the app.

Notes (important–make sure to read carefully)

1. For Section 3.1 of the design document (Class Diagram), you should simply use the design that your team created for Deliverable 1.
2. The project plan should cover your entire intended project development process, including the work done in Deliverable 1.
3. **Avoid jumping to coding right away and do not just focus on the code**. Documents (use case model, design document, …) are valued just as much as the code, and

sometimes more, so having a great app with a set of poorly prepared documents will likely result in a low grade.

4. If in a later deliverable you need to update any of the documents in an earlier deliverable based on your better understanding of the system, you can do so, as explicitly stated in the description of the deliverables. If you do so, please make sure to add a version number at the beginning of the document and provide a concise description, as indicated in the document templates.
5. To clarify the previous point, please note that this is not an invitation to procrastinate. Although we will grade the various artifacts submitted based on their latest version, **teams will be penalized if they simply submit "placeholder documents"**. In other words, we are perfectly fine with the documents evolving throughout the project, and will grade their latest versions, but **the required documents have to be present and reasonable in every deliverable**.
6. If you identify incompleteness or inconsistency issues in the requirements, feel free to either ask for clarifications or make **explicit** assumptions and develop the system accordingly.
7. Automated test cases are always appreciated, and ultimately useful for you, but it is fine to have a list of test cases that can be run manually (as long as for each test case you describe how to run it, what inputs to provide, and its expected output). Whether manual or automated, make sure that your tests adequately cover the functionality of the app.
8. Go to the directory "`GroupProject`" that you created for Deliverable 1 in the **team repo** that we assigned to you. Hereafter, we will refer to this directory as `<dir>`. Create a directory `<dir>/Docs`. All the documents that you will produce for Deliverable 2 should be placed in this directory,
9. As usual, after submitting a deliverable by pushing your changes to the assigned team repository in GT GitHub, the project manager must post on Canvas the corresponding commit ID. The group should check that this ID is correct in Canvas.