Crypto6300: An app for creating and playing cryptograms (Deliverable 3–Construction)

Background

A new customer, Boston Towers, has approached you looking for a way to keep “those meddling kids” in his neighborhood busy. As he shares their interest in secret messages, he would like you and your team to provide an app for the kids to practice solving cryptograms. In addition, he would like to be able to ensure that they create fair cryptograms for each other by monitoring the results. After reviewing your design, he 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 cryptogram app that Boston Towers 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.

Deliverables

- ~~D1 – *Software design*~~ **[done]**
- ~~D2 – *Inception and Elaboration* (see corresponding lessons):~~ **[done]**
- D3 – *Construction* (see corresponding lesson): **[due this week]**
    - Possibly revised earlier documents (`ProjectPlan.md`, `UseCaseModel.md`, `ExtraRequirements.md`, `DesignDocument.md`, and `TestPlan.md`), 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 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.
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. Once you satisfy all of the requirements, feel free to be creative and extend the functionality of the app if you feel like it and believe you can afford it. This is not required, of course.
8. 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 test cases adequately cover the functionality of the app. If you are keen to generate automated system test cases, you can try Barista, a tool developed at Georgia Tech for recording test cases and generating automated Espresso test cases based on these recordings.
9. 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>` .

a. The code of the app must be uploaded to the repository as a regular Android Studio project called `Crypto6300`, whose root is directory `<dir>/Crypto6300`. All classes should be in package `edu.gatech.seclass.crypto6300` and all the files needed to run your app must be uploaded to the repo. (To check this, we recommend that you clone your repo in a separate directory/workspace and run your project from there, in a clean AVD. Cloning your repo on a different machine would be even better, in case you have the option of doing that.).
b. The minimum API level for the app should be Level 23.
c. **Your application state should persist between runs.** If your database requires initialization, please test your application in a clean checkout with a cleared AVD to be sure that it will run for us on a fresh AVD. Simply redeploying within Android Studio will NOT clear your database on the AVD, and so will not not test any database initialization changes.
d. All other deliverables, including the user manual, must also be pushed to your repository, in directory `<dir>/Docs`.
e. Initial test results should be added to the previous `TestPlan.md`.
f. In addition, to help with cases in which we may have problems compiling your app, **add an APK file for your app in directory** `<dir>/APK`.

10. 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, as they are jointly responsible for the submission.