CSE 3120

Final Project

Learning Objectives

- Solve a problem by constructing a simple, interactive application using Android and Java.
- Document an object-oriented design in Unified Modeling Language (UML).
- UI design mockup by designing the activity screens of your app in advance.
- Work collaboratively using git and GitHub.
- Create Documentation using Javadoc and host it using github.io
- Create documentation and user guide using README file and GitHub Wiki
- Write unit tests and intent tests to do rigorous checking of your code and UI.
- Continuous integration using Travis CI with GitHub.

Problem Description

Consider the situation of someone who needs to monitor their blood pressure and heart rate data. Make a simple, attractive, intuitive, Android mobile app to track this data. Let us call this app: **CardiacRecorder**.

Specifically, each measurement has the following fields:

- date measured (presented in dd-mm-yyyy format)
- time measured (presented in hh:mm format)
- systolic pressure in mm Hg (non-negative integer)
- diastolic pressure in mm Hg (non-negative integer)
- heart rate in beats per minute (non-negative integer)
- comment (textual, up to 20 characters)

Only the comment field may be left blank for a measurement.

The app should allow the user to:

- show a list of measurements
- add a new measurement (which always appends to the bottom end of the list)
- view and edit the details of an existing measurement
- delete a measurement
- **see unusual blood pressures highlighted or flagged.**

Normal pressures are systolic between **90 and 140** and diastolic between **60 and 90.**

The list need not show all the information for a measurement if space is limited. Minimally, each record in the list should show the date, systolic pressure, diastolic pressure, and heart rate.

The app must assist the user in proper data entry. For example, use appropriate user interface controls to enforce particular data types and avoid illegal values.

The app must be persistent. That is, exiting and fully stopping the app should not lose data. **Therefore, you need to store the data offline or online.**

You can use a splash screen to decorate your app.

Deliverables (100 marks)

1. **Code Base: (40 marks)**

   App code: Your complete source code and compiled apk fle, implementing the working app and its user interface, will be inspected by me during the final presentation. Each java class must contain comments describing its purpose, design rationale, and any outstanding issues.

   Test Code: Your code must contain the test cases for both the unit and intent testing.

2. **Video: (15+25 = 40 marks)**

   App demonstration (maximum 1.5 minutes): The video is a demonstration of the app. The video files must be included in the GitHub repo (as links). The video is meant to show that the demonstration actions below actually work. No audio is needed. Maximum duration is 1.5 minutes. You can use any screen recording software. Focus on just the screen of the app, not the whole desktop. For visual clarity, do not use a handheld camera.

   Unit, Intent Testing, and Continuous Integration (maximum 2.5 minutes): Run tests manually on the IDE and run the tests using Travis CI and GitHub then capture the video.

3. **System Documentation: (20 marks)**

   UML: Describe the structure of your app's object-oriented design using UML class diagram(s), saved as non-lossy image file(s). Focus on the most important classes that you designed and implemented. Add notes to describe the main responsibilities of these classes.

GitHub README and Wiki: Create README file and wiki documentation like the following repository: https://github.com/CMPUT301F19T09/vibes/wiki

GitHub Projects: You should use GitHub Projects to track your feature implementations like this: https://github.com/CMPUT301F19T09/vibes/projects/1

Javadoc: Create Javadoc from the Javadoc comments and host at your github.io site. Create one if you do not have.

Note: All components must be delivered through a GitHub public repository (Create separate branches for every team member and finally merge with the main branch). You will show the repository, code, and videos during the presentation. You do not need to prepare a ppt presentation.

Demonstration Actions

1. Open the app from the launcher.
2. Show the list of measurements, with no measurements so far. (This should be the initial screen.)
3. Add a measurement with date 2019-02-01, time 22:00, systolic 126, diastolic 62, heart rate 52, and no comment.
4. Show the list, with this measurement.
5. View/edit this measurement to be systolic 106, and comment "resting".
6. Show the list, with this updated measurement.
7. Add a measurement with date 2019-02-02, time 23:00, systolic 85, diastolic 49, heart rate 60, comment "sitting"
8. Show the list, with the two measurements.
9. Add a measurement with date 2019-02-03, time 19:30, systolic 97, diastolic 63, heart rate 51, comment "laying"
10. Show the list, with the three measurements.
11. Delete the measurement dated 2019-02-02.
12. Show the list, with the two remaining measurements.
13. Exit and stop the app, showing in the running apps list that the app is no longer running.
14. Open the app again.
15. Show the list, with the two measurements.
16. View the details of the 2019-02-01 measurement.
17. View the details of the 2019-02-03 measurement.

Hints

This is a description of the core functionality. Often, problem statements from users lack details. As you are prototyping a design, you may uncover other behaviors that have not been specified but make sense in the context and intent of the application. For example, think about how someone might effectively use your application. It is up to you to decide what functions your design will need, based on the given problem description and valid assumptions, in discussion with your users (other students or me). You should consider asking the customer (in this case: me) what they want to see.

While you may discuss your design with other students, the code and documentation must be your team's own work. Code from publicly available sources may be used within reason and only if their licenses permit so. Always fully cite to give proper credit to the original developers in the source code and in the system documentation. For example, in citing a work, at least state: from whom, the date of publication, license, and URL. Do what is required by its license.

I will be inspecting your code, so "major" commented-out experiments should be cleaned up so that the code is readable.

For neatness and readability, diagrams should be created or drawn using a vector graphics editing tool and exported in a common, non-lossy graphics format. You may use pen and paper; however, the picture must be neat and readable.

Besides addressing the problem correctly, your software design will be evaluated on its proper use of object-oriented design concepts, such as separation of concerns and information hiding.