COMP3025 – Mobile and Pervasive Computing

Assignment 4 – Todo List App - Part 2 – Data Persistence for the Todo List App

Due Week #13 (August 11th, 2024) @ midnight

Value 10%

Todo List **Maximum Mark: 100**

Overview: This is the second part of a two-part Assignment. Using Android Studio and the **Kotlin** programming language, you will create a simple **Todo List app**.

In this part, you will create the logic that powers the **User Interface (UI)** for the Todo App that you built in Assignment 3.

The Todo List will be saved to some sort of persistent storage solution (internal storage, app preferences, Room database, Firebase, etc). You will be able to perform full CRUD for your Todo list.

Instructions :

The Todo App must include the following components – **a Todo List Screen, a Todo Detail Screen.**
(80 Marks: Functionality, 6 Marks: Internal Documentation, 4 Marks: Version Control, 10 Marks Video Demo)

1. Your **Todo List Activity** should include the following components **(35 Marks: Functionality):**
    a. A List of any current Todos (tasks) is displayed in a **RecyclerView**. Ensure that a new Todo can be added by pressing on an empty Cell in the **RecyclerView** or use another control (example a plus button). The devices keyboard will pop up and allow the user to enter the new Todo. As soon as they enter the short name of the Todo, the Switch Control and Edit Buttons will become visible (10 Marks: Functionality).
    b. Each Todo Item entered in the **RecyclerView** will use the same persistent storage solution of the developer's choice (internal storage, app preferences, database, Firebase, etc). The storage solution must record the following properties for each Todo (10 Marks: Functionality).:
        - name (String)
        - isCompleted (Boolean)
        - notes (String)
        - hasDueDate (Boolean)
        - dueDate (Date) – this is optional and should store some default date

c. Each Todo Cell's **Switch** Control will indicate a Todo is completed by either greying out or crossing out the Todo. This will also update the record in the persistent data storage solution (5 Marks: Functionality).
d. Each Todo Cell's **Edit Button** (or other UI Control) will link the user to the **Todo Details Activity** (5 Marks: Functionality).
e. If an *optional* Due Date has been selected on the Todo Details Activity (see below) then the **Due Date** should appear as **smaller text** below the Task Name in the same Todo Cell. The colour of the Due Date area may change to indicate that an item is late. (5 Marks: Functionality).

2. Your **Todo Details Activity** should include the following components **(45 Marks: Functionality):**
   a. The user may modify the **Todo Name** by changing the Todo Name TextView. This will also update the record in the persistent data storage solution when the update button is pressed (5 Marks: Functionality).
   b. The user may modify the contents of the **Notes** TextView. This will also update the record in the persistent data storage solution when the update button is pressed (5 Marks: Functionality)
   c. When the user turns the **Date Switch** on, the **Calendar** (or other control) will become enabled (by default the **Calendar** will be disabled). The user may select a **Due Date** for the Todo. This will trigger the Due Date to be displayed for the Todo in the **Todo List Activity,** as indicated above. This will also update the record in the persistent data storage solution when the update button is pressed (10 Marks: Functionality)
   d. When the **isCompleted Switch** control is selected, the Todo should be marked as completed by either greying out the Todo or showing in crossed out in the Todo List Activity, as indicated above. This will also update the record in the persistent data storage solution when the update button is pressed (5 Marks: Functionality)
   e. When the **Update Button** is pressed, a Message will be displayed asking the user if they are sure they wish to update the Todo. If the user selects a positive response to the Message, the record in the persistent data storage solution **will be updated** and the user will be taken back to the **Todo List Activity**. However, if they select a negative response to the Message, the user will remain on the **Todo Details Activity** and may continue editing the Todo. (10 Marks: Functionality)
   f. When the **Delete Button** is pressed, a Message will be displayed asking the user if they are sure they wish to delete the Todo. If the user selects a positive response to the Message, the record **will be deleted** from the persistent data storage solution and will be removed from the **Todo List Activity**. The user will then be taken back to the **Todo List Activity**. However, if they select a negative response to the Message, the user will remain on the **Todo Details Activity** and may continue editing the Todo. (5 Marks: Functionality).
   g. When the **Cancel Button** is pressed, and if the user has changed any data for the Todo, a Message will be displayed asking the user if they are sure they wish to discard the changes. If the user selects a positive response to the Message, they will be taken back to the Todo List Activity ***without updating*** the persistent data. However, if they select a

negative response to the Message, the user will remain on the **Todo Details Activity** and may continue editing the Todo. (5 Marks: Functionality).

3. Include **Internal Documentation** for your site **(4 Marks: Internal Documentation):**
    a. Ensure you include a **comment header** for *each* of your **`Kotlin`** files that indicates: the **File name**, **Author's name**, **StudentID**, **Date**, **App description**, and Version information (4 Marks: Internal Documentation).
    b. Ensure you include a **comment header** for each of your **methods** and **classes** (1 Marks: Internal Documentation)
    c. Ensure your program uses contextual variable names that help make the program human-readable (1 Marks: Internal Documentation).
    d. Ensure you include inline comments that describe your code's functionality only where required (1 Marks: Internal Documentation)
4. **Mandatory:** Share your files on **GitHub** to demonstrate Version Control Best Practices **(4 Marks: Version Control).**
    a. You must create a **new repository** for this Assignment. Keep the repository for Assignment 4 separate.
    b. Your repository must include **your code** and be well structured (2 Marks: Version Control).
    c. Your repository must include **commits** that demonstrate the project being updated at different stages of development – each time a major change is implemented (2 Marks: Version Control).
    d. You must keep your repository private and include your instructor as a collaborator (tsiliopoulos@hotmail.com)
5. **Mandatory:** Create a **Short Video** presentation with your favourite screen capture and streaming tool (OBS Recommended) and upload it to YouTube. You must also include a short PowerPoint (or Google Slides) Slide Deck that includes a **single slide** to start your video **(10 Marks: Video Demo)**
    a. The first (and only) Slide of your Slide Deck must include a **current image of you** (no avatars allowed) that is displayed appropriately on the page. You must also include your Full Name, Student ID, the Course Code, Course Name, and your Assignment information. (2 Marks: video)
    b. You will **demonstrate** your app's functionality in the Emulator. Your UI must be clearly visible (2 Marks: Video)
    c. You will **describe** each design for your app (2 Marks Video).
    d. Sound for your Video must at an appropriate level so that your voice may be clearly heard. Your Screen should be clearly visible (2 Marks: Video).
    e. Your Short Video should run no more than 5 minutes (2 Marks: Video).

SUBMITTING YOUR WORK

Your submission should include:

1. A zip archive of your Project files (rar files will **not** be accepted)
2. Your External Document (Word or PDF file).
3. A link to your private working GitHub Repository (your instructor added as a collaborator: tsiliopoulos@hotmail.com)
4. Your unlisted Video Demo link (YouTube preferred)

Evaluation Criteria

Feature	Description	Marks
Functionality	App deliverables are met and the app functions as intended. No errors, including submission of user inputs.	80
Internal Documentation	A program header is present and includes the name of the program, the name of the student, student number, date last modified, a short revision history and a short description of the program.	6
Version Control	GitHub commit history demonstrating regular updates.	4
Video Presentation	Your short video must demonstrate your app working in the simulator and discuss each of your designs	10
Total		**100**

This assignment is weighted **10%** of your total mark for this course.

Late submissions:

- 20% deducted for each day late.

External code (e.g. from the internet or other sources) can be used for student submissions within the following parameters:

1. The code source (i.e. where you got the code and who wrote it) must be cited in your internal documentation.
2. It encompasses a maximum of 10% of your code (any more will be considered cheating).
3. You must understand any code you use and include documentation (comments) around the code that explains its function.
4. You must get written approval from me via email.