Semester Project

Android Expense Tracker

Software Engineering

1 General Remarks

- This document covers the assignment of the **Semester Project**.
- You can achieve a total of **49 points** for this project.
- The mid-term design submission (**DESIGN**) is on **24.11.2021 at 23:59**.
- The final assignment submission (**FINAL**) is on **19.01.2022 at 23:59**.
- Templates for the **DESIGN** and **FINAL** reports that outline the required report structure will be available in time on Moodle.
- This is a **team assignment**. You shall collaborate within your team to solve this assignment. All team members must contribute approximately equally to each of the design, the implementation and the documentation activities.
- If you copy code or other elements from sources other than the lecture slides, you have to provide a reference to them in a comment above the corresponding entry.
- In case you need assistance:
  - If you encounter problems related directly to this assignment document, please post in the **Moodle**[1] **Discussion Forum - Practical Assignment**, as it is probably also of interest to other colleagues.
  - For implementation/coding or technology-related questions, you can contact our tutors via se2.tutor@swa.univie.ac.at.
  - For specific questions regarding your solution or team issues, please make use of the **Gitlab Issue Tracker** in your team project and mention the handle of your team's supervisor (announced in Moodle).

- You can discuss your inquiries regarding the practical part with the supervisors during the **online project question/answer sessions** in Moodle. To manage the sessions effectively, please write your questions in a **Gitlab Issue** and mention the handle of your team's supervisor **at the latest two hours before** the session.
- As a **last resort**, you can contact the course email: se2@swa.univie.ac.at.

2 Git[Lab Submission] System

Development for this assignment is to be undertaken in the faculty GitLab system at **https://git01lab.cs.univie.ac.at/** into your GitLab Team Project (repository) located in the Gitlab subgroup: **VU Software Engineering 2 / Students / 2021W**. All individual contributions to the project should be regularly updated. If you use branches, these should be regularly merged into the master branch which will contain the current state of your project. At the respective deadline, the **master** branch is archived automatically to a special branch for each of the DESIGN and FINAL submissions, which will be submitted for grading.

2.1 Plagiarism

We remind you that all of your submissions, implementations, designs etc. shall be created by you and your team members. Your submission may be subject to an automated plagiarism check for verification purposes. **Do not** use solutions from previous semesters, other teams, the Internet, or any other third party, as this would count as plagiarism and you would be directly given an "X" for the entire course. Code parts from Internet sites like Stack Overflow can be used, but must be clearly marked as copied solution parts (put a comment both at the start and the end of each copied part).

3 Semester Project: Android Expense Tracker

The goal of this project is to create an Android expense tracking application to manage and report financial activities for multiple accounts. Users should be able to view their account balances and keep track of cash inflows and outflows. By tracking income and expenses by type (category) users get an overview about their financial status and gain useful insights about their spendings.

Your assignment is to collaboratively develop a high-quality software solution that fulfills the requirements listed below. Focus on how to write and structure your source code cleanly, on correctly and reasonably applying design patterns, and on how to determine, track, and gain insights from code metrics. Java is the accepted language of the project. If you would like to use another langauge, it must be approved by your supervisor.

3.1 Design and Implementation Hints

To get familiar with core concepts of the Android framework and to get started with Android App development, we recommend to go through our hints and FAQs available in Moodle (see Section "Practical Part", "Hints and FAQs").

One important design decision in Android App development is how to properly structure your code alongside its necessary integration with the Android framework. Please study

the "Guide to app architecture"[2] and follow recommendations and best practices, if in your opinion applicable and reasonable. In particular, you may want to consider adopting the **Model-View-Controller design pattern**[3] but carefully decide whether your App needs to access remote data sources and if it makes sense to also use a repository.

Note that lower-level parts of some of the required functionality (related to, e.g., UI, persistence) are often well supported by the Android SDK or open-source libraries. Hence we recommend for such lower-level parts to make use of the available functionality in your technology stack and/or extend your technology stack appropriately by an external library.

3.2 Requirements

General Design Requirements (GDRs)

GDR1 You must apply object-oriented design.

GDR2 Minimum requirements for object-oriented assets in your project[4]:

- At least 1 interface (with one or more implementations)
- At least 5 classes (implementing the application logic)
- At least 1 abstract class (with subclasses)
- At least 1 custom exception class (with usages in your code)
- At least 1 level of depth in inheritance

GDR3 Apply principles of and follow best practices for object-oriented design, such as:

- Use data encapsulation
- Use inheritance, abstraction and polymorphism properly.
- Favor object composition over class inheritance
- Use exception handling
- Program to an interface, not an implementation
- Follow the principles of strong cohesion and loose coupling

Functional Requirements (FRs)

FR1 Accounts: Users should be able to create accounts with account data and account types (e.g. Cash, Bank Account, Card, Stock). Accounts can be updated (changed) or deleted. All transactions from FR3 belong to accounts.

FR2 Categories: Users should be able to assign categories to income (e.g. Salary, Dividend) and expenses (e.g. Food, Transportation, Education). They can create new categories, rename, or delete existing ones. When a category is deleted that is still in use, it shall not be usable on new transactions anymore but still exist for the transactions that are in use. Optionally, you can also support sub-categories.

[2] https://developer.android.com/jetpack/guide

[3] https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

[4] These requirements relate to ensuring the existence of certain minimalistic components; a reasonable solution surpasses these requirements.

FR3 Transactions: Users should be able to track income and expense transactions. Income and expenses can be categorized using the categories from FR2. Transactions belong to accounts from FR1. Existing transactions can be changed and deleted.

FR4 CRUD[5] functionality for persistent storage is required for FR1, FR2, FR3.

FR5 Reports: A view summary of transactions (income and expenses) grouped by category over the different accounts and given time periods (user-defined, daily, monthly, yearly) must be supported and augmented with at least two graphical representations (e.g., as pie/bar/line charts).

FR6 Spending Limit: A warning shall occur if the current account balance is reaching a predefined account threshold.

Quality Requirements (QRs)

QR1 **Comment** your code and provide a **code documentation** in an appropriate manner (e.g., use JavaDocs for Java).

QR2 Your implementation must be in compliance with a **style guide** (e.g., use Google Java Style Guide[6] for Java)

QR3 Use **common coding practices** (see lecture slides).

QR4 Use **defensive programming** (see lecture slides).

QR5 Use **key design principles** (see lecture slides).

QR6 Make sure that your implementation works properly by **testing** it thoroughly (e.g., using JUnit). Your testcases have to verify that your implementation meets all FRs. Consider applying other quality measures during development such as UI testing, manual testing, code reviews, etc. (see lecture slides for more details).

QR7 Your implementation **must use** at least **8 different design patterns**[7] from the following list, where each pattern must be used at least once wherever appropriate. Although some frameworks may already contain default implementations for some of the patterns, **you must provide and use your own implementation**.

- **Strategy Pattern**
- **Observer Pattern**
- **Decorator Pattern**
- **Factory Method Pattern / Abstract Factory Pattern**
- **Iterator Pattern**
- **Composite Pattern**
- **Proxy Pattern**

[5] https://en.wikipedia.org/wiki/Create,_read,_update_and_delete

[6] https://google.github.io/styleguide/javaguide.html

[7] You are advised to familiarize yourself with the lecture material about design patterns in Moodle ahead of the actual lecture dates.

- **Adapter Pattern**
- **Facade Pattern**
- **Template Method Pattern**

4 Submission

Only material that has been submitted **in time**, i.e., pushed to your GitLab Team Project in the **master** branch, and following **the naming conventions** will be taken into consideration. Provide your Android App not only in source code, but also explicitly include the Android Package Kit (APK) file and specify in the report an Android Virtual Device (AVD) configuration (hardware profile, system image) where the APK is supposed to run well.

For grading we rely in large parts on your provided documentation. Hence, we suggest to ensure that all documentation describes your system design and implementation concisely and includes the requested information.

4.1 Mid-Term Design Submission (DESIGN)

Deadline 24.11.2021 at 23:59

The DESIGN submission shall include the following artifacts in the root directory:

`implementation` A folder containing your **current** (skeleton like, possibly incomplete, but already compileable and executable/testable) implementation (including generated documentation, test cases, etc.). Provide the **Android Application Package (APK)** and **do not** bundle your code in archives like `.zip`, `.7z`, `.rar`, etc. The implementation for **DESIGN** shall adhere **at minimum** to the following:

- All **general design requirements (GDRs)** must be met.
- Provides **basic functionality**, i.e., a rudimentary coverage of the **FR1**, **FR2**, and **FR3** requirements.
- Contains an implementation of **at least two** of the patterns mentioned in QR7.

`report_design.pdf` A single **PDF** file report on your **current** project status, with the following **compulsory** contents:

- A **design draft** of the design approach regarding a possible implementation of the given task considering the application of design patterns. This should include one or more **UML class diagrams**, depicting the planned class layout. Further include a description of the used technology stack (e.g., Android Jetpack, other libraries/frameworks).
- **Design patterns** discussing how and in what context you applied the (at least) two selected design patterns in your current code.
- A **planned distribution** of design patterns among team-members, which, in case of approval by your supervisor, will be implemented **individually** for FINAL. Please indicate in your DESIGN deliverable which two design patterns will be realized by each student for the final submission.

- **Code metrics** (number of packages, lines of code, comment lines of code, number of classes, code bugs) covering your **current** implementation. It is expected that a work in progress will show a number of bugs; you are encouraged to use one or more static code analysis tools (e.g., the default lint tool[8] provided by Android Studio) and discuss the findings in the report.

The **DESIGN** submission will be graded with up to **15 Points**, with up to **7 points** for the quality of your report and up to **8 points** for the extent and quality of your implementation. If the minimum functional requirements are not met, the implementation part will be graded with **0 points**.

In any case, we strongly recommend that you have at least a rudimentary implementation of **all functional requirements** ready by the **DESIGN** deadline, so that afterwards you can focus on improving your solution regarding QRs with the techniques learned in the lectures.

4.2 Final Assignment Submission (FINAL)

Deadline 19.01.2022 at 23:59

The FINAL submission shall include the following artifacts in the root directory:

`implementation` A folder containing your **final** implementation (including javadoc, test cases, etc.). Provide the **Android Application Package (APK)** and **do not** bundle your code in archives like `.zip`, `.7z`, `.rar`, etc. The implementation for **FINAL** shall adhere to **all** of the following:

- All **general design requirements (GDRs)** must be met.
- All **functional requirements (FRs)** must be met.
- All **quality requirements (QRs)** must be met.

`report_final.pdf` A single **PDF** file report on your **final** project status, with the following **compulsory** contents:

- **Design patterns** discussing where the required design patterns occur in your solution. Discuss in detail the occurrence of each design pattern in the code. Support the decision to use a specific pattern with compelling arguments.
- Create **UML diagrams** to illustrate how the patterns have been applied and include relevant **code snippets** from your implementation. It is **mandatory** that the UML diagrams, the code snippets, and the source code are consistent. **Each team member is responsible for the documentation of their specified two design patterns**
- **Coding practices** describing how and to what extent you have considered coding practices. Discuss and show examples from your code.
- **Defensive programming** about how and to what extent you have considered defensive programming. Discuss and show examples from your code.

[8] https://developer.android.com/studio/write/lint

- **Code metrics** of your **final** implementation, covering the same code metric requirements as in DESIGN. Include also a discussion on code bugs found, and their resolution.
- **Team contribution**, which contains a **listing of contributions** and **time effort** for **each team member**.
- A **HowTo** documenting how to launch, initialize, and test the application.

`video` A folder containing either a video file, or a text file with a link to the video with the following contents:

- The video must follow the **HowTo** in the `report_final.pdf` to launch and initialize the application.
- The video must showcase the **execution and testing of all functional requirements** in sequential order (FR1, FR2, FR3, ..., FR9). The execution of each functional requirement must be indicated either verbally or by text overlays.
- We recommend that you upload the video (as private or unlisted) to a common and well known video platform and provide a corresponding link in the report and the repository. You may use YouTube, Vimeo, etc. These platforms take care of compressing, delivering and also verifying the videos such that, e.g., fundamental encoding issues are detected upfront. Alternatively you can upload the video to the repository (encode the video with a codec, such as, VP9, AV1, or h265).

The final submission will be graded with up to **34 Points**, with up to **16 points** for the complete and correct implementation of the required patterns, up to **10 points** for the quality of your implementation, including the correct use of proper coding practices, and up to **8 points** for the completeness and correctness of your implementation in terms of meeting all the given functional requirements, verified by test cases and executing the user scenarios expected of such an application. As mentioned before, we rely in large parts on your provided documentation for grading. Therefore, high quality documentation, i.e., **the report and the video**, is a prerequisite when awarding points.

5 Examination

- Each team has to present their solution at the end of the semester during the **Presentation Talk** slots (**PRES**). The assignment of teams to each slot will be announced in due course.
- The Presentation Talk is **essential** for the final grading. The grading of the final submissions, i.e., DESIGN and FINAL, are **preliminary** and can be revised upwards or downwards depending on the team and individual performance in the Presentation Talk, as well as the individual contributions to the semester project.
- Each team member should have knowledge of the entirety of the submitted solution, not simply the part he or she has worked on, and be able to explain the team's design process, decisions, and the rationale behind them.
- **No** elaborate presentation material (PowerPoint slides etc.) is required apart from the material you submitted as your solution.