In particular, you should **address the reported bugs and issues (also see Lab 5 document for more details)**. After fixing them, you need to properly address them in Github (under their reported issue, you need to respond and close the bug or if stays open explain why you are not fixing it now and what is the plan for fixing it). Also write a separate document that explains how you addressed each reported issue.

Include a **document** explaining the **code smells** or design issues you resolved with your **refactoring**. Describe what the code/design looked like before, what refactoring you applied, and how does it look like now.

😡

RegisterScreen class code smells
-Constructor is too long and handles too much. It can be split up into smaller methods to handle separate actions.
-Duplicate code email and password have similar setup code and a method could be created to reduce repetition

LoginScreen class code smells
-Constructor is too long and handles too much. It can be split up into smaller methods to handle separate actions.
-Duplicate code email and password have similar setup code and a method could be created to reduce repetition
-Large class it could be split into smaller classes to handle its multiple functions

😁

Complete refactoring that not only shortens the constructors and extracts common email/password field setup into helper methods but also splits shared functionality into a new base class. In this solution, a new abstract class (BaseScreen) contains standard UI setup methods (creating exit buttons, input fields, gradient buttons, fade animations, etc.) that both LoginScreen and RegisterScreen extend.
Explanation

BaseScreen.java:
– Contains shared UI helper methods (e.g. creating text fields, buttons, icons, and fade animations).
– Provides a common look and feel so both screens inherit the same styling and behavior.

LoginScreen.java:
– Extends BaseScreen and builds the login UI by calling the helper methods.
– Contains only login-specific logic (such as authentication and switching to the registration screen).

RegisterScreen.java:
– Extends BaseScreen and builds the registration UI similarly.
– Contains only registration-specific logic and the ability to switch back to the login screen.

This design minimizes duplicate code, breaks long constructors into smaller methods, and splits the responsibilities into smaller classes.

😡
GoalController

Too long methods, can be divided into smaller methods, if possible
Repetitive codes for getGoalsByUserId and getActiveGoalsByUserId methods.

😁
GoalController

The refactoring involved two main improvements:

• Simplified Long Methods: The updateGoalProgress method was broken down by extracting helper functions (e.g., convertToLocalDate) and clarifying the transaction filtering logic.

• Eliminated Code Duplication: Instead of duplicating logic, getGoalsByUserId and getActiveGoalsByUserId now simply delegate to their corresponding DAO methods.

😡
AnalyticsUI

The class includes lengthy methods (e.g., initializeUI(), createBarChart()) that mix UI setup with data processing.

😁

AnalyticsUI

Refactored the long methods initializeUI and the create chart methods by breaking them up into smaller methods to handle long method smell.
Split the chart creating methods into smaller parts to handle the different functionality of the methods
The methods handle dataset creation, chart creation, appearance customization, renderer configuration and axis configuration
This improves readability, maintainability, reusability and consistency of the creation methods

---

😡
QuizUI

- Duplicate UI Component Code
- Methods that combine UI creation, event handling, and component layout (e.g. createCategoryCard()) tend to be lengthy and hard to follow
- Navigating away during an active quiz causes loss of progress without warning

😁
QuizUI

- Refactored the duplicate UI component code by separating it into a smaller class file just to handle the UI component
- Long method is separated from the QuizUI file into a separate class just for it to focus on the UI handling component
- Implementing a confirmation popup that occurs when the user starts a quiz but switches tabs when they aren't done with the quiz. The user will be asked if they're sure that they want to leave, if they click yes then the tab will switch, if they click no then they will stay on the quiz page.

😡
GoalController

Allowed to have a duplicated name of goal

😁
GoalController, GoalsUI, GoalsUI

The refactoring involved two main improvements:

- Added a checker that avoiding duplicated goal name
- Added a pop-up message if the user attempts to do it, letting them know it is not possible no longer

---

😡
AnalyticsController

Able to import random csv files but the program will still accept it

😁
AnalyticsController, TransactionController, Analytics, AnalyticsUI

The refactoring involved two main improvements:

- Ensuring the program is able to handle multiple forms of CSV file formatting by adding codes in AnalyticsUI, AnalyticsController, Analytics, and TransactionController class. Retrieved its received data and check the validation of its format
- As well, added a checker for checking the format of imported CSV file

---

😡
AnalyticsUI

In the handleCsvImport() method of AnalyticsUI, CSV lines are split using String.split(","), which does not correctly handle fields that contain commas when they are enclosed in quotes. For example, if a description field is "Grocery, Walmart", the split will erroneously break the field into two.

😁
AnalyticsController, Analytics, pom.xml

The refactoring involved three main improvements:

- Updated the code from the changes in another bug related to CSV file
- Added OpenCSV dependency in pom.xml
- Edited the codes to ensure handling properly in AnalyticsController and Analytics classes

---

😡
QuestController / QuestUI
The checkAndCompleteQuests method in QuestController does check if the user has completed the quest but does not update the progress bar

Fixed quest bar updating
- Added database sync
- Refactored checkAndCompleteQuests method

😡
QuestController / QuestUI
checkAndCompleteQuest does not update the progress bar for each questsFixed the progress bar logging issue.

- Added clear conditions to set progress to 100% when transactions meet.
- Issues with int and double showing inconsistent values

😁
CalendarUI

Make it optional to fill out 'Description' part when you add a log

Updated processing of spending logging even when description field is left empty
- Modified transaction logging to process
- Updated validation to accept empty desc in TransactionController.java
- Modified constructors to handle empty descriptions
- Database requirements for non null descriptions are satisfied

---

😡

1. GoalsUI, GoalsController
    - Did not reflect the new status on saving budget goal
2. QuizController, QuizUI
    - After completing a quiz, the displayed score correctly reflects the number of correct answers, but the user's XP total does not update (or updates incorrectly).
3. QuestController, QuestUI

    - After completing 3 quizzes, in the quest tab under the 3 transactions quest, the progress bar does display that the quest is completed and instead the bar likes to fluctuate around.
4. QuestController, QuestUI
    - After completing all of the user's financial goals, the complete all goals progress bar does not reflect how many goals are completed and does not show that all of the goals have been completed

😁

This bug seemed to be fixed together when we were finalizing the code. It does not no longer appears.

😡

1. QuestController / QuestUI
    - All quiz questions are embedded directly within the loadDefaultQuestions() method, making the quiz difficult to extend or modify without code changes.
2. TransactionController
    - In the transactionController class there are two addTransaction methods that do the same thing. The parameters for the 2 methods are the two are also almost the same as each other.
3. CalendarUI
    - Unable to see the search results with the 'return' button which is supported in most web services or apps
4. QuestController
    - In the QuestController class there are many different methods that each does something different. Instead, maybe try splitting the different methods to multiple different classes

---

😁

- Due to lack of time and low priority, we decided those bugs to be closed and leave it as it is. We closed them, as a result.
- We would like to fix all of them if we have more permitted time to work on. We are looking forward to completing the bugs.
- More about 2: They are almost the same, not exactly the same. Their usages in codes displayed as the important part. We were concerned any small possibility of disturbing the codes when we make changes, so we decided to leave it as it is.
