StudyBuddy, reviewed by Grupp 14 - Down to Earth Rats

We all agree that the application is very well made and were impressed by the code and solutions. Well done!

1. Do the design and implementation follow design principles?

1.1. Does the project use a consistent coding style?

Yes! We believe that it uses a consistent coding style.

1.2. Is the code reusable?

Yes, the code utilizes interfaces. But more abstractions can be added, like the concrete Broadcast class maybe can use an abstraction.

1.3. Is it easy to maintain?

Factories are used which greatly increases the maintainability, the factory encapsulates all future changes. Nice! There are comments to help other programmers to understand the code, but it might be a little uneven throughout the project, too little at some places (Broadcast) and maybe too specific at some (AddBroadcastFragment).

1.4. Can we easily add/remove functionality?

We feel that it seems to be easy to add or remove functionality. The application is structured into well sized packages which makes it modular and thus helps the possibility of adding or removing functionality.

1.5. Are design patterns used?

The singleton pattern is used in combination with database instances, nice stuff! Also the Observer Pattern is used, for example in BroadcastRepository with the class LiveData.

2. Is the code documented?

There are areas that are not commented and other areas that are. We believe that the areas that are lacking need comments. There are also places with a lot of comments, maybe even too specific if the reader is a programmer.

3. Are proper names used?

Proper names are used, they keep a consistent style throughout the project and are descriptive of the classes’ responsibilities. As outsiders, we could easily understand each class's responsibilities.

4. **Is the design modular? Are there any unnecessary dependencies?**

   Refer to point 1.4

   Looking at the class diagram the dependencies seem reasonable and no circular dependencies.

5. **Does the code use proper abstractions?**

   Refer to point 1.2

6. **Is the code well tested?**

   We could not run the tests due to an error, but that might be on us. But we could see that you have written tests and they look okay. But it looks like you can add a “setUp-method” in “UserTest” so you only have to create a user and BroadcastObjects one time instead of doing it in many tests. Maybe reduce the number of asserts used in each test method.

7. **Are there any security problems, are there any performance issues?**

   The code exposes some concrete classes which can lead to unsafe dependencies and rigid code.

   You use FireBase which we think is good for security.

8. **Is the code easy to understand? Does it have an MVC structure, and is the model isolated from the other parts?**

   We all agree that the code is easy enough to understand in its complexity. The model doesn’t use any GUI components, thus isolated from the rest.

9. **Can the design or code be improved? Are there better solutions?**

   The Course class currently only has a single String with getter/setter, something seems a little bit off. Why does it need to be a single class that only contains a String?

   Some methods also have a lot of code, so more functional decomposition would be nice.

   More abstractions could be added to improve the project modularity and to future proof additionally.

SDD uses really nice package diagrams!