Peer Review of Filmster from OOPP-WITH-THE-BOIS

RAD/SDD

- Introduction in RAD can be rewritten to be more formal and less personal. (More objective)
- Watchlist definition is empty.
- No story names in User Stories.
- No punctuation in User Stories.
- Make the “Acceptance” title more visible.
- Reformulate some user stories to be less ambiguous (e.g. ID 2, can’t understand what “Want to see it” is supposed to mean)
- Everything should be in English. (e.g. some user story, and protocol.txt)
- DoD should be for User Stories, not Tasks.
- Figures should be anchored in the text using figure references.
- Some parts of the user interface are not described in text (e.g. movie details, missed-tab)
- Class responsibilities missing from domain model description.
- Domain model shouldn’t include fields.
- Domain model doesn’t need to include implementation details (like the database) and should show relations (like Movie <- Movies).
- Class diagrams contain template text (like +field: type and method(type): type) that should be removed or changed.
- Interfaces in class diagrams have to show methods.
- protkoll.txt could be split up into multiple documents to make it easier to find a single meeting protocol.

Design/Code

- FilmsterRepository shouldn’t keep track of current media, the ViewModel should.
- Ambiguous names like “movieCategories” in Preferences, is it preferred categories or blocked categories, etc.
- Package names should be better, e.g. remove “.example.” and “test” from name.
- Inconsistent use of interfaces (concrete types returned when an interface could be returned instead).
- Inconsistent code style (some methods have capital first letters, some not), methods comments do not always follow the same style. (Most methods are commented with JavaDoc, but MediaAdapter uses normal comments for some methods.) Private fields are prefixed with “m” in

SingletonRequestQueue, but nowhere else (copy/pasted code?). Inconsistent use of access modifier (writing out private for all fields except one).

- Proper names are not used (e.g. IAdapter, an adapter of what?, get250movies does not return anything). Inconsistent pluralization of classes (e.g. DislikedMedias in plural and WatchedMedia in singular). Ambiguous naming of MediaBack and MediaFront (they seem to be the backs and fronts of the movie card, with the front being the image and back being details, but the names don't tell us this - they should be named more clearly, GestureHelper.GestureListener.onFling - e1 and e2 don't tell us what the events represent and what their differences are.) Inconsistent abbreviation, not used everywhere but used in some places.
- Missing javadoc comments on many public methods.
- WatchedMedia, DislikedMedias and LikedMedias are all the same except for a single variable or two. They should be merged into a common supertype. This is not extensible right now.
- Private API keys should not be in the source code. They should be in an environment variable or some other kind of secret storage.
- Views break Open-Closed principle, adding a new view means performing shotgun surgery (adding it in many places). Adding a new tab to WatchlistView also breaks this.
- Methods do not follow functional decomposition, some are way too big (like MainView.onCreateView - can be broken up into many methods, most onCreateViews are too big, IMDbAdapter.getStringRequest, GestureHelper.GestureListener.onFling)
- Catching Exception is generally a code smell, only specific kinds of exceptions should be catched, and they should be handled (not just printed) (e.g. GestureHelper.GestureListener.onFling, IMDbApiAdapter.jsonObject2Media).
- IMDbApiAdapter does not follow Single Responsibility - it both calls the API and adapts the returned data into a format that the application can use.
- Hardcoded functionality (Filmster constructor, MainView.checkMovieLength).
- Unit Tests do not test the entire application, and they do not everything, and what they do test is simple getters and setters, not any logical functionality (which is what needs to be tested).
- Inner classes should be private or broken out into separate files (MediaAdapter)
- Fetching movies from the API every time the application is started, without caching, will result in possible startup time slowdowns. (Definitely if on a slow network connection) Movies should be fetched on demand, only a couple should be prefetched.
- Out-commented code should not be pushed to master.
- The Singleton pattern is correctly used in SingletonRequestQueue.
- Magic numbers are used throughout the entire project.
- The model has no dependencies on Android, which is good.