# Paying the debt

## Screen separation

Originally, our app only had 1 set of files that handled the UI for our app which was only a single screen. At the time, the features we had implemented only required one screen and we simply added on to the screen throughout the iteration. In [this commit](https://code.cs.umanitoba.ca/comp3350-winter2024/timeline-a02-9/-/commit/6eaf48cce3da883b6e907815d7d3cda635fc5224), the UI was separated into different files to handle the different screens we would have in our app and a way to navigate to them. In the future, we will now have an easier way to add to certain screens in the app or to create new ones entirely.

We think this is prudent and deliberate technical debt because we knew in advance that we would need more screens for future features but still decided to implement only one screen. We figured that making our logic layer work was more important and we could more easily focus on it by having a single screen we could just add to.

## Stubs for Goal UI

At the time [this commit](https://code.cs.umanitoba.ca/comp3350-winter2024/timeline-a02-9/-/commit/f47fc3d5c9a56a8a06357091bcdc12b7ab97cb7d#cd76b6d9fc5b23be56a9160098d72e72a28944ed_0_62) was made, many of the UI elements such as date tracking and completion status were stubs. This technical debt arose from trying to make faster merges and prioritizing basic screen functionality and navigation between screens, making it prudent and deliberate. We fixed this technical debt in two subsequent commits. [[this commit](https://code.cs.umanitoba.ca/comp3350-winter2024/timeline-a02-9/-/commit/e0cbefcecc2360fa0c71472661daa69f885c4313)](https://code.cs.umanitoba.ca/comp3350-winter2024/timeline-a02-9/-/commit/e0cbefcecc2360fa0c71472661daa69f885c4313) is where we introduced progress tracking within the Goal screen and [this commit](https://code.cs.umanitoba.ca/comp3350-winter2024/timeline-a02-9/-/commit/b0430a0d1cf9c3bf5e552bf1b24fb0df4e0b59aa) is where we updated the Goal’s UI so that it displays the current date seamlessly. These changes enhance the user experience and contribute to a more robust application interface.

## Consolidate dialog implementations

Initially, our application's task dialog implementations were scattered all over the App file, making effective updating or maintaining these parts quite a chore. First used to quickly integrate dialogue capabilities in the respective context to enable swift feature development and iterative improvements informed by feedback. So, at the very beginning of our project, this strategy served its objective, but eventually, with the growth of our application's complexity, joined by the new functionality (goals), has given birth to a dispersed and less cohesive code structure.

Majorly, in this update, we have collected everything that concerns task dialogs in one section of the Task List, centralizing the dialog system. This will help in simple development, support, and easy updating of task dialogs. Doing that will make our codebase cleaner, but it also prepares our framework for more scaling and maintainability, which consequently helps in all future development efforts.

In [this commit](https://code.cs.umanitoba.ca/comp3350-winter2024/timeline-a02-9/-/commit/91561e245165a6fb6b39b3b0fc4aeb04d6406c6b), refactoring is done with technical debt that we accrued previously in this project lifecycle. We considered this debt to be reckless and deliberate because we knew there was a better way to implement this but since it was added near the end of the last iteration, we went with a simple solution that was good enough at the time.

# SOLID Violation

Open-Close Principle (OCP) violation: [comp3350-winter2024/internetenemies-a02-10#200](https://code.cs.umanitoba.ca/comp3350-winter2024/internetenemies-a02-10/-/issues/200)

# Retrospective

After the last retrospective, we opted to limit the scope of MRs and encourage earlier review cycles and the merging of incomplete features.

* Implement Goal UI and Navigation ([!44](https://code.cs.umanitoba.ca/comp3350-winter2024/timeline-a02-9/-/merge_requests/44)): The implementation is not meant to be clean, but to work, hence a lot of stubs. But the smaller MR makes it very easy to review and merge.
* Goals database ([!38](https://code.cs.umanitoba.ca/comp3350-winter2024/timeline-a02-9/-/merge_requests/38) and [!47](https://code.cs.umanitoba.ca/comp3350-winter2024/timeline-a02-9/-/merge_requests/47)): The implementation was split into 2 MRs from the first MR ([!37](https://code.cs.umanitoba.ca/comp3350-winter2024/timeline-a02-9/-/merge_requests/37)) to self-contain the changes and make it easier to review and merge.

# Design patterns

**Singleton:**

* [Notifications](https://code.cs.umanitoba.ca/comp3350-winter2024/timeline-a02-9/-/blob/7dcdde799fb4b56432e4861240f87ad7899ab3b7/app/src/main/java/ca/umanitoba/cs/timeline/ui/notification/Notifications.kt): a singleton managing all Android notifications channels and constants

**Adapter:**

* [DefaultGoalDao](https://code.cs.umanitoba.ca/comp3350-winter2024/timeline-a02-9/-/blob/7dcdde799fb4b56432e4861240f87ad7899ab3b7/app/src/main/java/ca/umanitoba/cs/timeline/data/DefaultGoalDao.kt): converts SQL results into goal objects and goal-related operations.
* [DefaultTaskDao](https://code.cs.umanitoba.ca/comp3350-winter2024/timeline-a02-9/-/blob/7dcdde799fb4b56432e4861240f87ad7899ab3b7/app/src/main/java/ca/umanitoba/cs/timeline/data/DefaultTaskDao.kt): converts SQL results into task objects and task-related operations.

# Iteration 1 Feedback Fixes

We did not have any issues opened by the grader, so we compensated with a bonus debt above.