Group 9 A02 Iteration 1 Worksheet

Adding a feature

Pomodoro Feature

The Pomodoro feature was first prototyped as a basic timer, which can be found in the draft timer merge request (draft: feat: basic Timer implementation). As a group, we discussed and rethought how the timer should be implemented which resulted in an improved version of the timer prototype that can be found in the feature timer merge request (feat(timer): implements a basic timer). We used the old pomodoro timer draft code as a starting point and added to it, resulting in a new pomodoro logic merge request (feat(pomodoro): add Pomodoro timer logic). Next, we built the notification function of the pomodoro timer and in the process, we discovered a bug in how the timer reports progress to other components. We discussed again and refactored the pomodoro timer to change how the data is passed. We decided to decouple the timer from the pomodoro state machine to allow a controller to drive the logic instead of the pomodoro state machine controlling everything. After that was completed, tests were added and the refactoring solved the above issue.

User stories associated with this feature:

- Reminder to get back to work
- Reminder to take breaks
- Have a timer for working and taking breaks

Merge requests associated with this feature:

- feat(pomodoro): add Pomodoro timer logic
- feat(timer notification): basic notification for when the timer has ended
- refactor(pomodoro)!: decouple Pomodoro state machine from timer
- fix(pomodoro): make sure that the timer complete state is emitted
- draft: feat: basic Timer implementation
- feat(timer): implements a basic timer
- feat(timer tracker): Implemented basic timer tracker

Merge commits:

- feat(timer): implements a basic timer
- feat(pomodoro): add Pomodoro timer logic
- feat(timer notification): basic notification for when the timer has ended
- refactor(pomodoro)!: decouple Pomodoro state machine from timer
- feat(timer tracker): Implemented basic timer tracker

Associated tests:

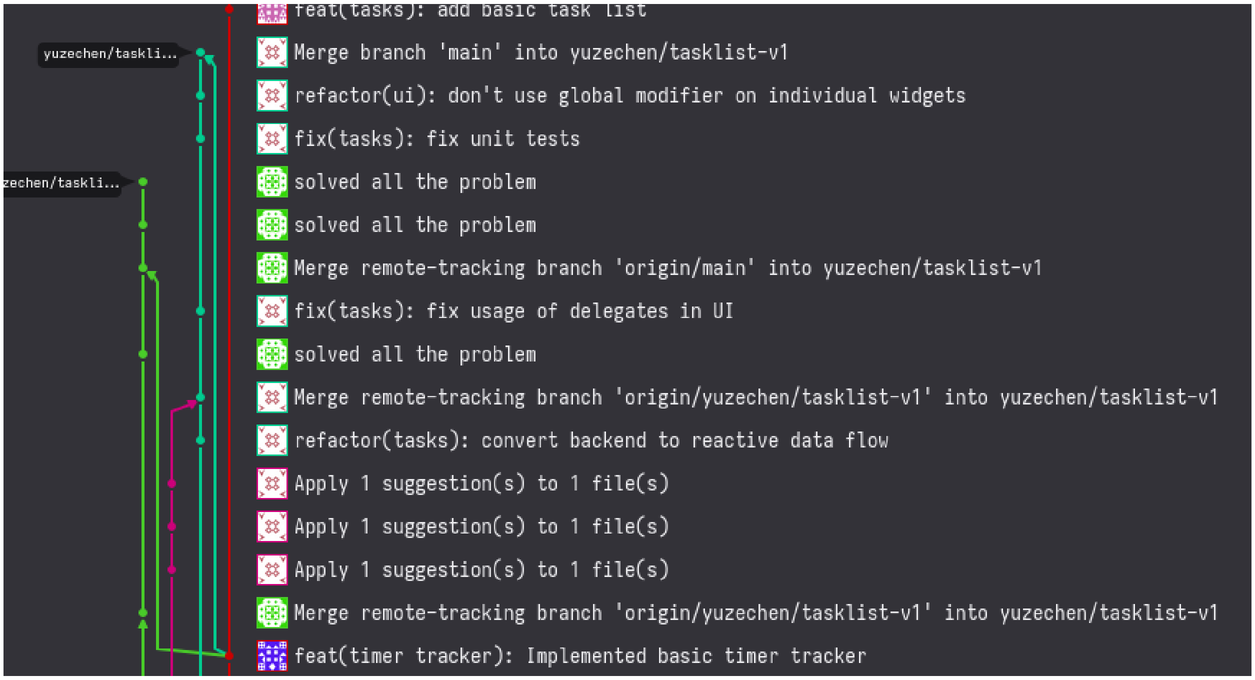
- PomodoroUseCaseTest
- TimerUseCaseTest
- FormatTimerDurationTest

Branching

Our branching structure can be found here.

We delete feature branches whenever they are merged, as such we do not possess a good graph for this branching scheme.

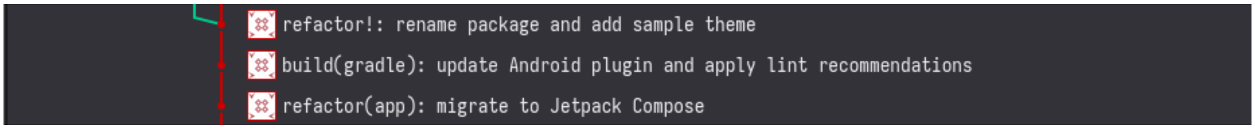
The line for main branch is always straight (red line) and branches are always branched out instead of having many merge cycles:



```
feat(tasks): add basic task list
Merge branch 'main' into yuzechen/tasklist-v1
refactor(ui): don't use global modifier on individual widgets
fix(tasks): fix unit tests
solved all the problem
solved all the problem
Merge remote-tracking branch 'origin/main' into yuzechen/tasklist-v1
fix(tasks): fix usage of delegates in UI
solved all the problem
Merge remote-tracking branch 'origin/yuzechen/tasklist-v1' into yuzechen/tasklist-v1
refactor(tasks): convert backend to reactive data flow
Apply 1 suggestion(s) to 1 file(s)
Apply 1 suggestion(s) to 1 file(s)
Apply 1 suggestion(s) to 1 file(s)
Merge remote-tracking branch 'origin/yuzechen/tasklist-v1' into yuzechen/tasklist-v1
feat(timer tracker): Implemented basic timer tracker
```

The task list was one of the longer branches that was not deleted, before being merged as the red dot on top. The corresponding merge request can be found here.

For other branches of ours, they do not show up on the graph due to the merge being squashed:



```
refactor!: rename package and add sample theme
build(gradle): update Android plugin and apply lint recommendations
refactor(app): migrate to Jetpack Compose
```

The best way to look at the work done for them is via the merged MR list as GitLab keeps the commits that the squashed MRs are based on in there.

SOLID

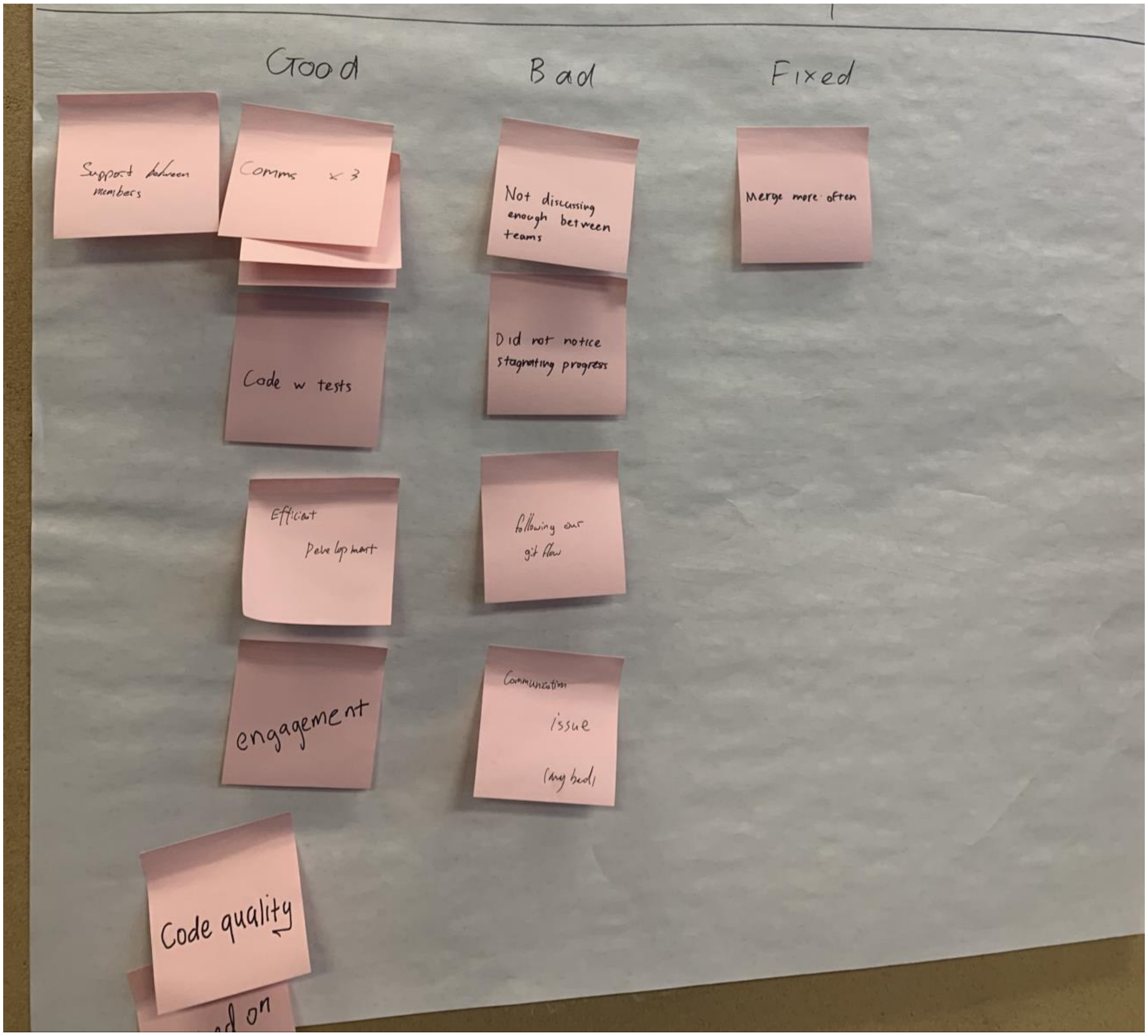
We found a Dependency inversion principle violation in Group 10's project and opened an issue for it: comp3350-winter2024/internetenemies-a02-10#110

Agile Planning

Our user stories remained the same, but we pushed one feature to Iteration 2: Track time spent on goal. This was done as we did not finish the task list feature on time, so there was no time left to work on this feature.

Retrospective

Group:



Good:

1. We maintained effective communication throughout the iteration
2. The quality of our code is good
3. We were able to write appropriate tests for all our code
4. There was good engagement for everyone in the group as well as support between the members
5. There was efficient development

Bad:

1. The group was divided into 2 teams, one worked on the Pomodoro timer feature and the other worked on the task-list feature. There was not enough discussion between both teams which led to some delays and stagnation.
2. Some members had issues following the group's git flow.

Solution:

1. We plan to create merge requests and commit more often to facilitate faster prototypes and making sure that everyone gets a handle on what is going on.

Individual:

Adrian Hizon

- I utilized our branching strategy to help minimize the amount of possible merge conflicts
- I implemented the notification popping up when the timer is finished

Chineze Obi

- I worked with two other group members to implement the pomodoro timer feature
- I wrote the logic for the pomodoro feature

Yuze Chen

- I worked with one group member to implement the task-list feature
- I wrote the basic layers code for the tasklist feature

David Schwarz

- I worked with a group member on implementing functions for the task-list feature
- I implemented the delete and edit functions for the task-list

Hiếu Lê

- I kept the team communication running and made sure that code is reviewed by at least one other person before they can be merged

- I refactored the Pomodoro logic to reduce coupling and fixed a few bugs