Fatin - Project Portfolio

Project: TA-Tracker

Overview

TA-Tracker is a productivity tool made for NUS School of Computing (SoC) Teaching Assistants (TAs). Rather than using several excel spreadsheets or notes, TA-Tracker enables TAs to manage their students and track teaching duties in a single, convenient-to-use platform . The application is mainly written in Java and spans a considerable 20,000 Lines of Code (LoC). With a rigorous system of checks and tests put in place, users can be assured that the codebase is well-maintained, and that the code quality is consistently high. A comprehensive set of guides are also provided to ensure a smooth on-boarding process for both users and contributors alike.

Summary of contributions

As the main developer of the application's User Interface (UI), I play a crucial role in integrating the features my teammates create with the UI. During group discussions, I placed extra emphasis on guiding my teammates to engineer solutions that could be more easily assimilated into the UI, to ensure that their work becomes user-visible. As a result, my team was able to morph the given codebase from a trivial application into a polished product.

With the substantial amount of experience I gained while designing the application, I was available and prepared to help out with various tasks, like design considerations and debugging. As the most experienced UI developer in the team, I was highly involved in helping my teammates become more familiar with JavaFX and CSS. My role in the development of the project was especially crucial, as I enabled my teammates to be able to display all the hard work that they have put into the development of their respective features to the users. My major contributions are as follows:

Updated the User Interface

The UI is at the heart of TA-Tracker, displaying the output of TA-Tracker to the user visually. As the main contributor to the MainWindow of the UI, I play an integral role in ensuring that the content is being displayed to the user correctly, while keeping the interface simple and informative. The overall layout of TA-Tracker was changed by adding tabs and icons (#120, #182, #227), and by creating all the ListPanels and their respective ListCards (#120, #182, #204).

I also took care to ensure that the information displayed in the Cards was integrated with the backend whenever my teammates made new contributions to the application, such as adding new fields (#322) or commands (#330). A "Total Earnings" label in Claims tab was also added to improve user experience, as money makes the world go round (#243, #322).

Enabled highlighting of applied filters

As the TA-Tracker was initially based on AB3, the UI at the beginning of the project looked plain. Instead of indiscriminately adding colours to TA-Tracker, I favoured a different approach, and

enabled the relevant ListCells in the Student Tab and Claims Tab to be highlighted whenever filter commands were entered (#210, #227, #235, #238). This not only made a huge improvement in the visual differences between TA-Tracker and AB3, but also enabled users to better focus on the information displayed.

This contribution also required extensive debugging and contributions to the inner workings of the filter command, which was a rather challenging command to implement (#243, #314, #322). In the Session Tab, highlighting the ListCells was a less favourable option, since there was only one ListPanel to display. I overcame this challenge by creating a filter header (#322).

Implemented relevant commands to improve User Experience

Goto Command : To achieve the goal of making TA-Tracker a keyboard-only application, I implemented the GoTo Command to allow users to switch between tabs via the command-line rather than clicking on the tab-headers (#189).

In a similar spirit, I enabled **switching to relevant tabs for all commands**, to better the user experience. This allows new information to be displayed instantaneously upon entering a command (#189, #210, #212). This involved creating an enum for UI handling in CommandResult (#189, #212) and as a result, the painstaking process of updating the entire code-base.

SetRate Command : The hourly rate for all the displayed Earnings was initially set to $40, which is the rate at which the majority of TAs are being paid. Based on feedback from the PE Dry-Run, I created a command to change this value due to the possibility of changes being made to it. (#321)

Other UI Improvements

I also contributed to the development of HelpWindow and StatisticsWindow (#227, #235) by fixing sizing issues and adding ScrollPanes. Moreover, I included the option to close both windows by pressing the ESC key to achieve the goal of making TA-Tracker a keyboard-only application (#236).

Added extensive automated tests

I made thorough JUnit tests for the StudentCommand, StudentCommandParser, and Student as well as its relevant fields. (#340, #341, #347)

Improved overall code quality

- Packaged all commands, parsers, models and UI components (#143, #212)
- General quality fixes to the entire code-base based on Codacy report (#350, #351)
- Created enum classes for SessionType and GroupType (#120, #182)

Other contributions

- Created a skeleton for the student delete command (#113)
- Removed the requirement for compulsory phone and email fields in student add (#146)
- Managed the project by commenting on critical pull requests (various)

Here is the code that I have written for this product: [All commits]

Contributions to the User Guide

Given below are sections I contributed to the User Guide. They showcase my ability to write documentation targeting end-users.

Contributions to the Developer Guide

Given below are sections I contributed to the Developer Guide. They showcase my ability to write technical documentation and the technical depth of my contributions to the project.