Chia De Xun - Project Portfolio

Overview of Project

AddMin+ is student developed application under the National University of Singapore's *CS2103T Software Engineering Module*. In this module, we were organized to a team of 5 members for the software engineering project. The project requires us to enhance a basic command line interface desktop application, the AddressBook - Level 3 developed by the se-edu team for the the purpose of teaching Software Engineering principles.

About AddMin+, the All-in-One Administration App

*Figure 1. The Graphical User Interface (GUI) for *AddMin+**

AddMin+ is an all-in-one administration desktop application, that is specially designed for any events management start-up company with limited manpower and resources. These companies would usually only be able to afford to hire one admin staff to handle both the management of events and employees and this can be tough. The Admin+ Team understands the struggles that start-ups faces on a daily basis, and we are here to help!

AddMin+ is specially designed to ease the workload of the admin and allow him/her to effectively handle the administrative tasks of the company by providing the following functions: manpower management, event creation and deletion, editing of event details after creation, manual and automatic manpower allocation for events and providing an overview of all the data via statistics. The purpose of this user guide is show the user how they can use our app to carry out their administrative tasks efficiently and also answer any questions they may have while using our app.

- AddMin+ is specially designed to help ease the workload by offering a one-stop platform to help

deal with the various administrative needs of the company.

- AddMin+ uses a Command Line Interface(CLI) and displays the results through a Graphical User Interface(GUI) to assist the employee in their specific needs.

My Roles & Responsibilities

As Team Leader, I took charge of the overall direction of the project, working and communicating closely with all my team members to ensure that the project is well-maintained and coordinated. In addition, I defined, assigned and tracked project tasks for the group, maximizing use of Github's Issue Tracker and Project Board for task and milestone managememt. By frequenting providing feedback, either through Pull-Request Reviews or direct, verbal communication, I also helped to enforce high code quality and software engineering standards.

Introduction to Application Components

This is a short section that aims to get the reader familiar and up to speed with the components in the AddMin+ app. AddMin+ contains two main entities: the `Event` and `Employee` components. Besides *Create, Read, Update, Delete (CRUD)* commands on both entities, AddMin+ is capable of manpower allocation, scheduling, salary management and statistics generation which requires interaction with both the `Event` and `Employee` objects through the use of the Date & Time mapping that sets the schedule and duration of an `Event`.

Summary of contributions

- **Major enhancement**: Implemented the `Event` **Component**, and most significantly the **Event Date-Time Mapping and its ancillaries**
    - What it does: The `Event` Component features basic *Create, Read, Update and Delete (CRUD)* functionality, extended with the ability to store a list of `Employees` in a Manpower List and a Date-Time Map that stores the schedule of the `Event`. The Date-Time Mapping feature enables the user to accurately and precisely map a schedule of an existing `Event`, broken down into its individual dates and time period.
    - Justification: The `Event` Component is critical for our application's functionality as its target audience is Administrative Staff in small companies managing events. The `Event` Component provides full flexibility for the user to manage event details through its CRUD functionality. Additionally, having a known date-time schedule of each `Event` supports the ability for administrative matters such as manpower allocation and schedule generation, ensuring that the employees are allocated to events efficiently without overlaps/conflicts in schedule.
    - Highlights: This Date-Time Mapping feature is significant and is required for two other significant features of our application, manpower allocation and scheduling. To enhance the user experience and achieve our goal of simplifying tedious administrative work, I extended the functionality of the command to enable *quick assignment* of dates and time by stating a date range, instead of just a single date to map. Overall, the implementation was challenging as I also needed to consider how edits to the start/end dates of the `Event` may break the manpower allocation or assigned date-time mapping and deal with them accordingly.

- **Minor enhancement**: Implemented the **Fetch Employee** Command
    - What it does: Shows the schedule of a single `Employee`, which displays a list of dates which itself contains a list of `Event` that the `Employee` is allocated to.
    - Justification: It would be useful for either the adminstrative staff or the employee to view the current schedule of an `Employee` for tracking and logging purposes.
- **Code contributed**: [tp Code Dashboard]
- **Functional Code Contributed**:
    - **Event**: [`Event`, `EventDate`, `EventDateTimeMap`, `EventDayTime`]
    - **Command**: [`FetchEmployee`, `AssignDate`, `DeleteDateMapping`, `ClearDateMapping`, `AddEvent`, `EditEvent`]
    - **ParserUtil** [**ParserUtil**]
        - `parseAnyDate()`, `parseEventDate()`, `parseTimePeriod()`, `parseEventDateTimeMap()`
    - **Parser**: [`AssignDate`, `DeleteDateMapping`, `ClearDateMapping`, `AddEvent`, `EditEvent`]
    - **Processor**: [**EmployeeEvent Processor**]
- **Test Code Contributed**:
    - **Event**: [`EventDate`, `EventDateTimeMap`, `EventDayTime`]
    - **Parser**: [`EventParsers`, `CommandParserTest`]
- **Other contributions**:
    - Project management:
        - Set-Up the Team Organization and Project Repo on Github
        - Implemented and Enforced Protected Branching to ensure that PRs failing CI are not merged.
        - Contributed consistently in PR Reviews and identifying bugs and code flaws for Team Members. PRs reviewed (with non-trivial review comments): #146, #139, #132, #111, #85
        - Managed releases `v1.2.1`, `v1.3`, `v1.3.2`, `v1.3.3` on GitHub
        - Hosted internal Practical Examination for Team to find Bugs. Manage to find multiple medium-severity bugs [#253, #258, #259, #260], which were later resolved
        - Extensively used Github Project Board and Issue Tracker to manage Task Allocations for the Team
        - Reorganized files to improve code tidiness (Pull requests #129)
    - Enhancements to existing features:
        - GUI: Upgraded the `Employee` and `Event` Cards with a gradient background and image (Pull requests #140)
        - Assisted in the Refactoring of `Person` to `Employee` (Pull requests #92)
        - Assisted teammate in the implementation and code quality of the `Finance` component (#238)
    - Documentation:

        - Did refactoring, overall editing and polishing to the User Guide: (#14)
        - Added `Parameter Constraint` section of UG which is referenced by most commands listed in the User Guide (#242)
    - Community:
        - PRs reviewed (with non-trivial review comments): #56
        - Contributed to forum discussions (#153, #159)
    - Tools:
        - Integrated Travis CI to the Team Repo
        - Integrated Netify (Automatic Website Hosting) to the Team Repo

Contributions to the User Guide

Given below are extracted sections I contributed to the User Guide. They showcase my ability to write documentation targeting end-users. My other contributions are found in Section 2.2, 3.2, 3.4.5, 3.4.6, 3.4.7.

Setting a Date&Time to an Event: `set_ev_dt`

Sets a Date-Time Schedule Mapping for a specific Event.

NOTE

Even though the start date and end dates of each event is stated, the event is not assumed to be held for the entire range. Hence, the Date-Time setting feature enables the user to *declare and set* the schedule of the event. Events are initialized without any schedule, other than their stated start and end date with a default time of *0800-1800*.

Format: `set_ev_dt EVENT_INDEX [on/EVENT_DATE] [till/EVENT_DATE] time/EVENT_DAYTIME`

- Note the [Constraints] for **INDEX**, **DATE**, and **TIME PERIOD**.
- If **both** the `on/` and `till/` prefixes are **not used**, then all dates inclusive of the start to end date of the target event will be set with the stated `EVENT_DAYTIME`.
- If **both** the `on/` and `till/` prefixes are used, then the date range (inclusive) from the start to end date stated will be set with the stated `EVENT_DAYTIME`.
- If **only** the `on/` prefix is used, it will just set the `EVENT_DAYTIME` for the *single* stated `EVENT_DATE`.
- The `till/` prefix must be used with the `on/` prefix, and not by itself.
- All `EVENT_DATE` **must be within the range of the Target Event's Start and End Date**
- The `EVENT_DATE` declared by the `on/` prefix cannot be after the `EVENT_DATE` declared by the `till/` prefix

Examples:

- `set_ev_dt 2 on/02/10/2019 time/1000-2000`
  Sets the 2nd Event from the Event List a time period of 10am-8pm on the 2nd of October 2019.
- `set_ev_dt 3 time/0500-1500`
  Sets the 3rd Event from the Event List a time period of 5am-3pm on all dates from the start to the end date (inclusive) of the Event.
- `set_ev_dt 1 on/02/10/2019 till/10/10/2019 time/0500-1500`
  Sets the 1st Event from the Event List a time period of 5am-3pm on all dates from the 2nd of October to the 10th of October (inclusive).

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. My other contributions are found in Section _

Setting Date & Time to Events

Implementation

The `Event` object is constructed with a start date and an end date as class attributes, both of which are `EventDate` objects, which represents a single day by itself. Our implementation of `Event` does not assume that the event will be occurring consecutively from the start to the end date, and requires the user to manually assign each specific date with the time period that the Event is in process.

NOTE	In our implementation, when the Event is instantiated, the time period of 0800-1800 is automatically created and mapped to the Start & End Dates of the Event.

To achieve this functionality, there exists an `EventDayTime` object that encapsulates the period of the day. It has two class attributes - both of which are `LocalTime` objects to represent the start and end time.

Each Event contains an `EventDateTimeMap` object that maps an `EventDate` object to an `EventDayTime` object using a HashMap implementation. This mapping is added through the `EventAssignDate` command.

It requires the use of the following objects/methods from the `event` package.

- `EventContainsKeyDatePredicate` - Check whether the stated date exists within the range of the Event's Start and End Date.
- `Event#assignDateTime` - Calls the `EventDateTimeMap` object to insert a Date-Time mapping.
- `EventDate#datesUntil` - Returns a Stream of `EventDates` from the Start to End Date. Used to auto-set a DateTime mapping for all dates.

NOTE	To improve user productivity and effectiveness, omitting the target date from the command text will automatically create the mapping for every date from the Event's start to end date, inclusive. Alternatively, by specifying both a start and end date range in the command text, a mapping for the range will be created.

Given below is an example usage scenario of the program functionality when a user attempts to assign a Date & Time to an already existent Event.

Step 1. The User executes the command `set_ev_dt 2 on/18/10/2019 time/0900-2000`, with the intention to assign the date of 18th October 2019, time period 9am-8pm to the second event currently displayed in the event list. If the date is omitted, i.e. `set_ev_dt 2 time/0900-2000`, the time period 0900-2000 will be automatically assigned for all dates from the start to end date of the Event. Alternatively, if the end date is stated, i.e. `set_ev_dt 2 on/18/10/2019 time/22/10/2019 time/0900-2000`, the time period of 0900-2000 will be assigned for all dates from 18th to 22nd of October.

Step 2. The parser checks if input format is correct, and attempts to create `Index`, `EventDate` and `EventDayTime` objects from it

Step 3. The command checks if the index of the event stated exists on the displayed list, and if the stated dates is within the start and end date of the Event. (Input Validation)

Step 4. The command calls `Event#assignDateTime()` on the referenced Event object to add the EventDate-EventDayTime mapping into `EventDateTimeMap`.

Step 5. If only a single target date is stated, continue to Step 6. Else, the system will repeat Step 4 through the entire date range - which is either the start and end date of the `Event` or the range specified by the user.

Step 6. DateTimeMapping is converted a String to save and update in Storage.

Step 7. Done.

NOTE	If the command execution fails, a `ParseException` (from Step 2) or a `CommandException` (from Step 3) will be thrown, specifiying the reason of the error.

The following sequence diagram shows how the `AssignDateCommand` works:

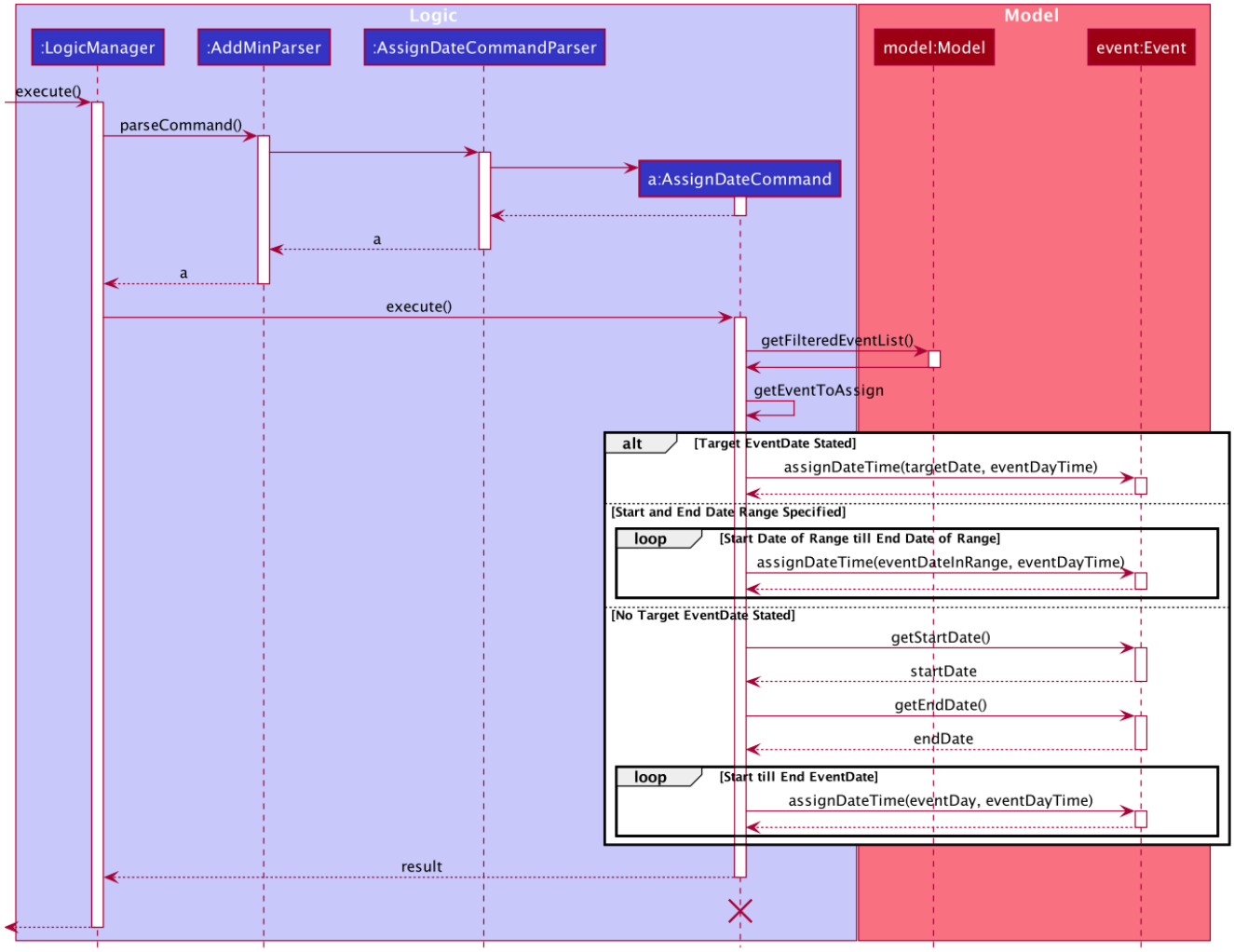

Figure 2. Sequence Diagram for AssignDateCommand Command

NOTE	The lifeline for `AssignDateCommand` ends at the destroy marker (X).

The following activity diagram shows how the Setting of Date&Time to Event work:

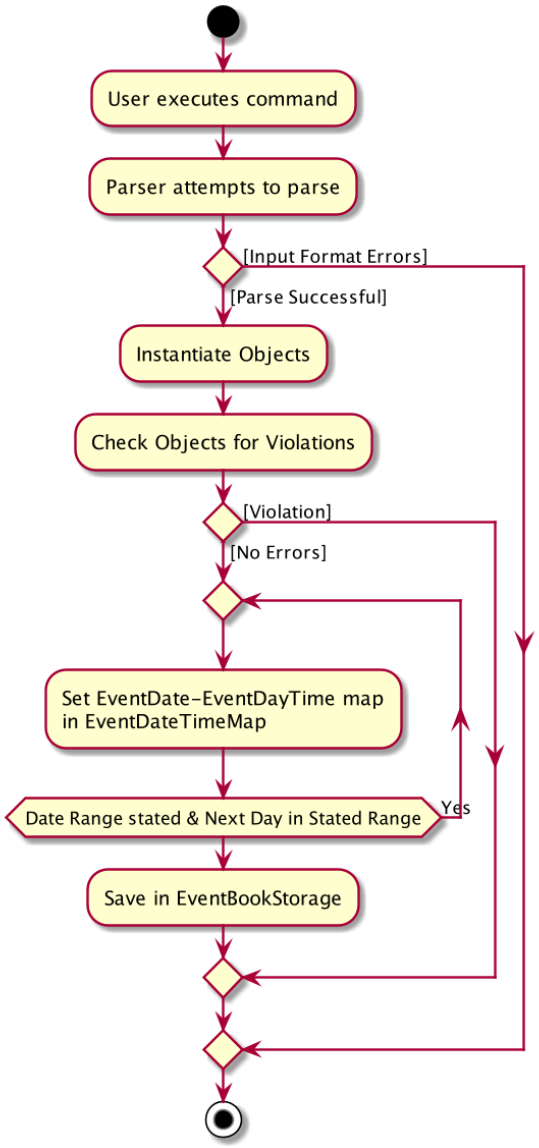

Figure 3. Activity Diagram of the SettingEventDate Command

Design Considerations

Below, we discuss two key aspects - how we store EventDateTimeMap and how commands to edit `Event` affect the `EventDateTimeMap`.

Aspect	Alternative 1	Alternative 2
Storage of DateTimeMap	Stores the DateTimeMap in an string format that is saved in a field of an `Event` in `eventbook.json`. **Pros:** Simplicity in implementation and easier reference as it is loaded and saved to the same JSON file. **Cons:** Performance issues as it needs to update the entire event object although only one attribute is updated	Store the DateTimeMap in a separate file e.g. `EventDateTimes.json` that will be referenced by EventBook during initialization. **Pros:** Faster performance in saving and loading as it is kept separate from `eventbook.json` and hence will not **Cons:** Requires a new storage unit, along with all its supporting functions which will require alot of repeated code. Instantiation of the `Event` object when the app is started will be more complicated as well due to the need to read from two separate files
Decision: Alternative 1 Alternative 2 would make sense if our app is utilizing a DBMS and it would be a best practice to separate the information into separate tables. However, as we are constrained with not utilizing a DBMS, **Alternative 1** is a logically simpler, shorter, and more efficent solution from a software engineering standpoint as it limits the amount of repeated code that we would have written to support another storage unit.		
Impact of Edit Event on `EventDateTimeMap` The `edit_ev` command does not edit the DateTime mapping of an `Event` itself - this is done through the `set_ev_dt` or the `delete_ev_dt` instead. However, if the event has its start or end date fields edited, it will affect the `EventDateTimeMap` as its mapping may suddenly be out of range of the edited `Event` start and end dates.	Prevent the editing of Event Dates if the `Event` date range is reduced and will cause existing Date-Time Mappings to fall out of edited Event's Start-End Date Range. **Pros:** Greatly reduce the potential for buggy behavior, as `EventDateTimeMap` would contain false mappings that do not correspond to the new Event's Start-End Date Range. User would not have to worry about the inadvertent loss of data. **Cons:** Negative User Experience - Will need to take extra steps to manually delete DateTime mappings.	Allow editing of Event Dates if the `Event` date range is reduced, but will clear `EventDate` mappings in `EventDateTimeMap` that fall out of edited Event's Start-End Date Range. **Pros:** Better User Experience - Narrowing the Start/End Date would naturally mean that the user no longer require mappings on those dates and hence they can be safely deleted. **Cons:** Increased risk of inadvertent deletion of existing Date-Time mappings from `Event`.

Aspect	Alternative 1	Alternative 2
Decision: Alternative 2 Alternative 2 offers better user experience by reducing the addition steps of hassle, especially since AddMin+ is focussed on automating and reducing the burdensome workload of administrative staff. Even so, we understand the risk of a user executing a typo and inadvertently delete existing Date-Time mappings. Hence, as a mitgating measure, we would be introducing a 'Confirmation' command in a `V2.0` feature that would allow the user to confirm and proceed with the `edit_ev` command if it were to delete existing Date-Time mappings due to a narrowing of the range.		

Conclusion

While AddMin+ is not my first experience with a software engineering project, it introduced many *good, quality* software engineering practices and principles that I would apply to my future projects. This includes advance uses of the Github Issue Tracker, Project Workflow, PR Reviews, testing, and documentation. Despite the heavy workload, I thoroughly appreciate the learning experiences that would shape me to become a better software developer. Overall, I am satisfied to have worked with my team to bring the project to completion and release an elegant, high-quality product.