Tan Kai Li Catherine — Project Portfolio

About this Portfolio

The purpose of this project portfolio is to document my role and contributions to the project, **COMPal**. **COMPal** was developed by our team of 5, comprising myself, Jaedon Kee Shaowei, Liu Peize, Muhammad Sholihin Bin Kamrudin and Yue Jun Yi.

Overview

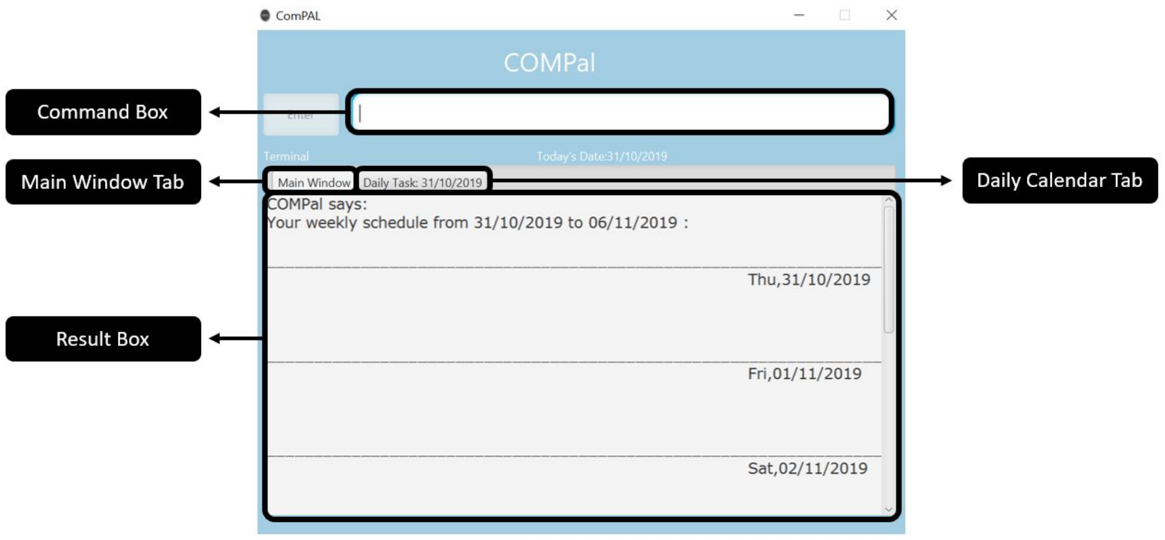
For our Software Engineering project, my team and I were tasked with enhancing a basic command line interface desktop application, **Duke**, a Personal Assistant Chatbot which helps with task management.

We chose to morph it into a student life planner called **COMPal**, with the aim of helping students organise their hectic schedules to improve their productivity and time management. This targeted student life planner enables students to manage class schedules, submission deadlines, co-curricular activities, and even project meetings, all in one application.

COMPal supports:

- Command-line based interface with intuitive commands for efficient and convenient use
- Addition, edit and deletion of tasks with a single command
- Searching for a specific task in the myriad of tasks by its keyword effortlessly
- Setting reminders for tasks to keep track of important deadlines
- Viewing of upcoming deadlines and events to prevent missing deadlines
- Finding free time slots on a specific day to arrange a project meeting

This is what our project looks like:



```
COMPal says:
Your weekly schedule from 31/10/2019 to 06/11/2019 :
________________________________________________________
                                         Thu,31/10/2019
________________________________________________________
                                         Fri,01/11/2019
________________________________________________________
                                         Sat,02/11/2019
```

Figure 1. Graphical user interface for **COMPal**

My role was to design and write codes for the view-reminder, set-reminder and findfreeslot features. The following sections illustrate these enhancements in more detail, as well as the relevant documentation I have added to the user guide and developer guide in relation to these enhancements.

For ease of understanding, *classes/activities/events/deadlines* will be broadly referred to as *tasks* throughout this document.

Note the following symbols and formatting used in this document:

Icon	**Description**
ℹ	Additional important information about a term/concept
💡	A tip that can improve your understanding about a term/concept
⚠	A warning that you should take note of

Summary of Contributions

This section shows a summary of my coding, documentation, and other helpful contributions to the team project.

Enhancement added: I implemented the view reminder, set reminder and find free slot features.

- View reminder:
  - What it does: The view-reminder command allows the user to view all reminders this week. Undone tasks that are due within the week and overdue tasks are pre-set to be included. Additionally, users can manually turn on reminders for important tasks they want to keep track of using the set-reminder command.
  - Justification: This feature allows users to keep track of upcoming and important tasks so that they can better manage their deadlines and submissions.
- Set reminder:
  - What it does: The set-reminder command allows the user to set reminders for important tasks they want to keep track of. Once the reminder settings are turned on for that specific task, it will automatically be included in the reminder list.
  - Justification: Some tasks have higher importance than other tasks. For instance, the user may want to start working on high weightage school assignments at an earlier time. Hence, the set-reminder command enables the user to track that task easily.
- Find free slot:
  - What it does: The findfreeslot command allows the user to search for a free time slot in the specified date with the specified duration.
  - Justification: In the event that the user wants to arrange a project meeting, this feature is extremely useful as it can find a free time slot in the user's existing schedule with the specified meeting duration, which enables the user to arrange project meetings effortlessly.

Code contributed:

- Please click this link to view my code contributions on RepoSense :

[Contributed code]

- Please click this link to view my contributed pull requests :

  [Contributed pull requests]

Other contributions:

- Project Management:
  - I managed the User Guide, including the content and formatting.
- Enhancements to existing features:
  - Wrote JUnit tests for existing features and ensured that the coverage of my contributed code is at 90% (Pull requests #202, #208, #218)
- Documentation:
  - Wrote all documentation for features I implemented which can be found in the User Guide and Developer Guide
- Community:
  - Reviewed Pull Requests: #91

Contributions to User Guide

The following sections are the additions that I have made to our COMPal User Guide:

- 4.1.7. View Reminder: view-reminder
- 4.1.8. Set Reminder: set-reminder
- 4.1.9. Finding Free Time Slots: findfreeslot
- 7. Command Summary

The link to the User Guide can be found here.

The following is an excerpt of my contribution from *Section 4.1.9. Finding Free Time Slots: findfreeslot* of our COMPal User Guide.

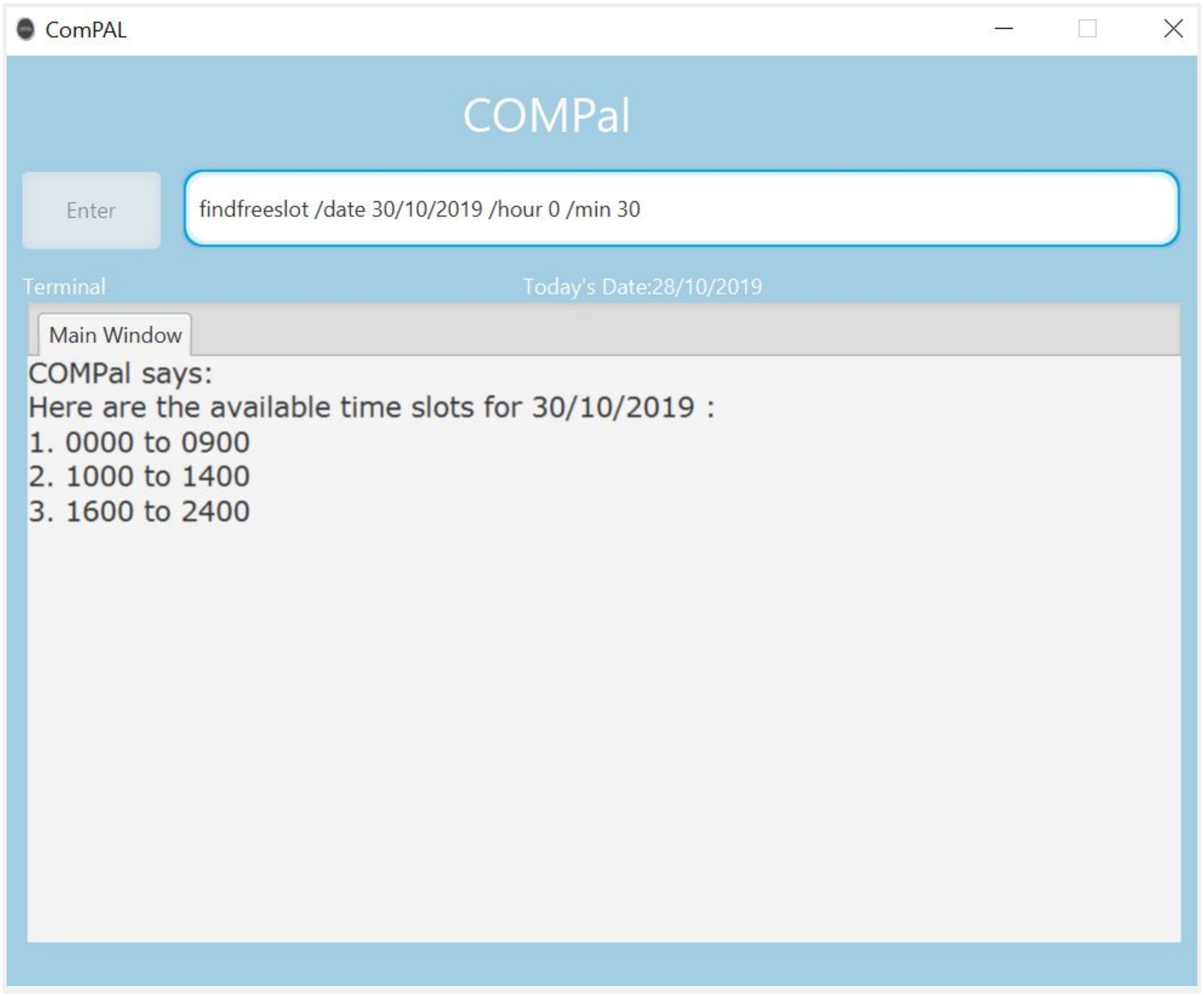
4.1.9. Finding Free Time Slots: `findfreeslot`

Need to find a free time slot to arrange a project meeting? Enter `findfreeslot /date DATE /hour HOUR /min MINUTE` in the **command box** to find a free time slot in the specified date with the specified duration in hours and minutes.

- `DATE` is the date of the free time slot needed
- `HOUR` is the input hour duration of the time slot
- `MIN` is the input minute duration of the time slot

Examples:

- `findfreeslot /date 29/10/2019 /hour 2 /min 0`
- `findfreeslot /date 01/11/2019 /hour 1 /min 30`



```
findfreeslot /date 30/10/2019 /hour 0 /min 30
```

```
COMPal says:
Here are the available time slots for 30/10/2019 :
1. 0000 to 0900
2. 1000 to 1400
3. 1600 to 2400
```

Figure 12. Expected output of `findfreeslot /date 30/10/2019 /hour 0 /min 30` command.

Contributions to Developer Guide

The following sections are the additions that I have made to our COMPal Developer Guide:

- 4.6. Model Component
- 5.4. Reminder Feature
- 5.8. Find Free Slot Feature

The link to the Developer Guide can be found here.

The following is an excerpt of my contribution from *Section 5.8. Find Free Slot Feature* of our COMPal Developer Guide.

5.8. Find Free Slot Feature

This feature allows users to find a free time slot of a specified duration on a specified date. This section will detail how this feature is implemented.

5.8.1. Current Implementation

Command: `findfreeslot`

Upon invoking the `findfreeslot` command with valid parameters (refer to User Guide for `findfreeslot` usage), a sequence of events is then executed.

For clarity, the sequence of events will be in reference to the execution of a `findfreeslot /date 12/11/2019 /hour 1 /min 30` command. A graphical representation is included in the Sequence Diagram below for your reference when following through the sequence of events. The sequence of events is as follows:

1. The `findfreeslot /date 12/11/2019 /hour 1 /min 30` command is passed into the `logicExecute` function of `LogicManager` to be parsed.
2. `LogicManager` then invokes the `processCmd` function of `ParserManager`.

3. `ParserManager`, in turn, invokes the `parseCommand` function of the appropriate parser for the `findfreeslot` command which in this case, is `FindFreeSlotCommandParser`.
4. Once the parsing is done, `FindFreeSlotCommandParser` would instantiate the `FindFreeSlotCommand` object which would be returned to the `LogicManager`.
5. `LogicManager` is then able to invoke the `commandExecute` function of the returned `FindFreeSlotCommand` object.
6. In the `commandExecute` function of the `FindFreeSlotCommand` object, **task** data will be retrieved from the `TaskList` component.
7. Now that the `FindFreeSlotCommand` object has the task data, it invokes the `sortTask` method to sort the **tasks** in chronological order.
8. The `getFreeSlots` function is then invoked to obtain the list of free time slots stored in an array list of strings.
9. With the output returned from the `getFreeSlots` function, the `CommandResult` object will be instantiated.
10. The `CommandResult` object would then be returned to the `LogicManager` which then returns the same `CommandResult` object back to the `UI` component.
11. Finally, the `UI` component would display the contents of the `CommandResult` object to the user.

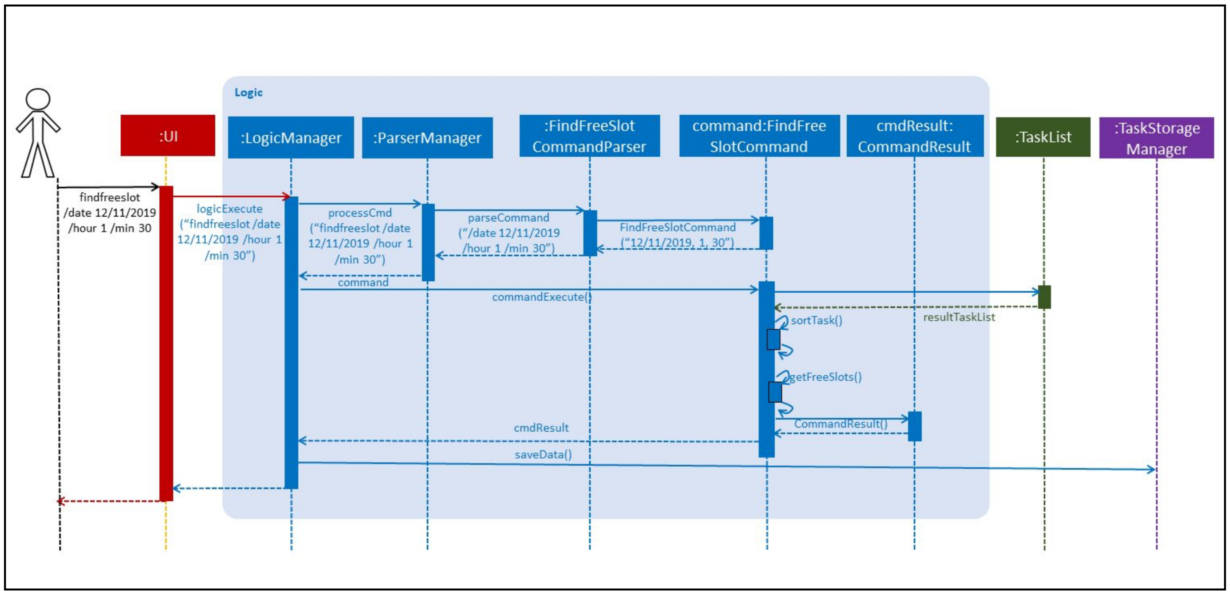

Logic
:UI
:LogicManager
:ParserManager
:FindFreeSlot CommandParser
command:FindFree SlotCommand
cmdResult: CommandResult
:TaskList
:TaskStorage Manager
findfreeslot /date 12/11/2019 /hour 1 /min 30
logicExecute ("findfreeslot /date 12/11/2019 /hour 1 /min 30")
processCmd ("findfreeslot /date 12/11/2019 /hour 1 /min 30")
parseCommand ("/date 12/11/2019 /hour 1 /min 30")
FindFreeSlotCommand ("12/11/2019, 1, 30")
command
commandExecute()
resultTaskList
sortTask()
getFreeSlots()
CommandResult()
cmdResult
saveData()

Figure 10. Sequence Diagram executing the **`findfreeslot`** command.

5.8.2. Design Considerations

This section details our considerations for the implementation of the `findfreeslot` feature.

Aspect: Data structure to support the functionality of `findfreeslot` command

- **Alternative 1 (current choice):** Use 2 pointers to keep track of the end time of the previous event and the start time of the next event to calculate the duration of the free time slot available.
  - **Pros:** Much more efficient, takes up less memory space.
  - **Cons:** More difficult to implement and prone to bugs.
- **Alternative 2:** Use a 2-Dimensional Boolean array of size 24 by 60 to store the 1440 minutes in a day. Use a for-loop to loop through all events and mark the respective array entries as true for the time slot of the events.
  - **Pros:** Can be implemented easily.
  - **Cons:** Inefficient and takes a much longer time to loop through the events. Since the **task** list can potentially contain a large number of events, looping through the events one by one may slow down the application by a significant amount. More memory space is used up as well.

Alternative 1 was chosen as it is more efficient and takes up less memory space. Having a **task** list with a large number of events will not increase the computational time by a huge proportion, and the memory space needed for 2 pointers is small. In contrast, alternative 2's implementation will result in a very slow application, which is not ideal. Storing a 2-Dimensional array of size 24 by 60 is also more space consuming.

5.8.3 Future Implementation

1. Allow the user to input a period of days/weeks/months in which he wants to find a free time slot, instead of the current implementation of one day. Example: If the user inputs a start date and an end date, together with a duration, the output will be all the available time slots with the duration from the start date until the end date.