Sholihin Kamarudin - Project Portfolio

About this Portfolio

Hi! I am Sholihin Kamrudin and I am currently a Year 2 Information Security undergraduate at the National University of Singapore (NUS).This portfolio documents my contributions to the project, COMPal, which was developed together with my team; Jun Yi, Jaedon, Pei Ze and Catherine.

PROJECT: COMPal

For our Software Engineering project, my team and I were tasked with enhancing a basic command line interface,*Duke*,a Personal Assistant Chatbot which helps a person to keep track of various tasks.We choose to morph it into a text-based scheduling calendar application called **COMPal**, which caters to NUS students who prefer to use a desktop application for managing their hectic schedule with chosen priorities levels.

COMPal offers the following features and benefits:

- Flexible commands to facilitate easy creation, editing and deleting of *tasks*.
- A recursion system so as to allow you to add recurring items efficiently.
- An intuitive interface to view your daily schedule of a particular day.
- Setting priority levels of *tasks* for the system to optimise your schedule.
- Viewing reminders of *tasks* that you are deemed to be reminded.

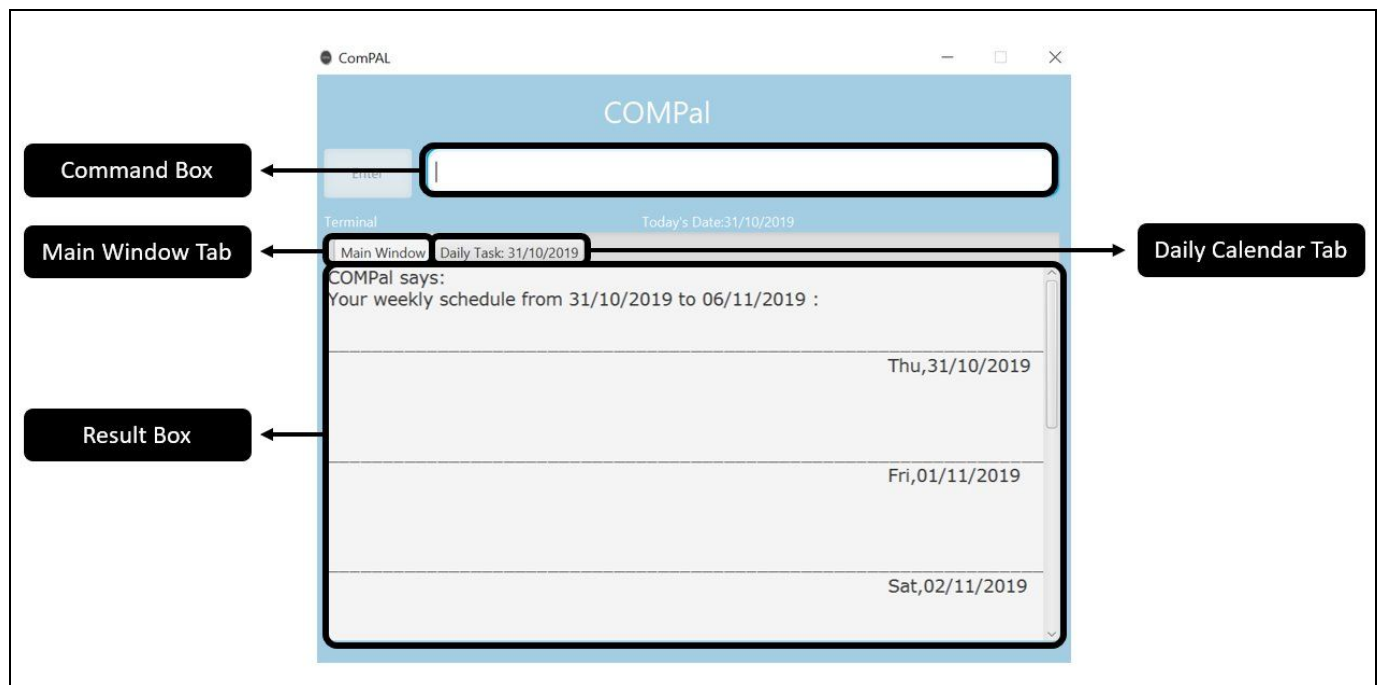This is how COMPal looks like:

Figure 1. Description of COMPal components.

My role was to implement the ability of how users can view their schedule through the `view` commands, ensure that the tasks stored were sorted in chronological order with our specialised priority ranking and also give the user the ability to `export` and `import` COMPal schedules.

The next sections will illustrate my implementations in more detail, as well as the relevant sections I have contributed to the user and developer guides concerning my implementations.

For ease of communication, this document will refer to classes/activities/events/deadlines that you might add to the scheduling planner as tasks.

Additionally, throughout this Project Portfolio, there will be various icons used as described below.

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 added the ability to `import` **and** `export` **iCalendar file**.
    - What it does: Allows users to export their stored schedule. Additionally, allow users to import schedules from any scheduling application that allows export of their schedule to iCalendar files.
    - Justification: This feature improves COMPal significantly because the user can import existing appointments on other calendar applications into COMPal.Thus, allowing him to combine external appointments with his school schedule. Additionally, the user is also able to export his stored schedule in COMPal to another scheduling application or share it with others.
    - Highlights: This enhancement works with existing as well as future commands. Hence, it required an in-depth analysis of design alternatives about the aspect of how we would want COMPal to be compatible with other scheduling application. The chosen implementation must be reliable to ensure compatibility throughout.
- Code contributed: [Contributed code] [Contributed pull requests]

- Other contributions:
  - Project Management:
    - I managed the codebase for COMPal up till the release of version 1.4. These releases can be viewed here.
  - Enhancements to existing features:
    - I Wrote JUnit tests for my own existing features to increase overall coverage. I have ensured that the coverage of my contributed code is at 90%.
  - Documentation:
    - I updated all documentation related to the features that I am in charge which can be found in the User Guide and Developer Guide.
  - Community:
    - I reviewed Pull Requests and provided non-trivial review comments for some requests. These Pull Requests can be viewed here.
  - Tools:
    - I Integrated continuous Integration,code reviews and code analytics tools :
      - Continuous Integration tools: Travis-CI
      - Automated code reviews and code analytics tools: Codacy

Contributions to the User Guide

This section shows some of my contributions to the User Guide. Which showcase my ability to write documentation targeting end-users.

Listed below are my contributed sections in the User Guide:

- General Commands: Completing a Task: `done`, Viewing Tasks: `view`, Listing Tasks: `list`, Importing iCalendar files: `import`, Exporting COMPal schedule: `export`
- A quick overview of the generated daily schedule.
- Frequently Asked Questions

Click here to see the User Guide!

Below is an excerpt of my addiction to the User Guide for the `export`, `import` commands and explanation on the `daily calendar` feature.

Exporting COMPal Schedule: `export`

Want to export your COMPal schedule to another application? With the export command, you can export it as an iCalender file!

Format: `export /file_name FILE_NAME`

i	Your exported file will be saved in the same folder you run COMPal at!

With this function, you can import your COMPal schedule to another COMPal application or share it with your friends or even import it to google calendar or any calendar that is able to import iCalendar files!

Check this link here to learn how to import to Google Calendar!
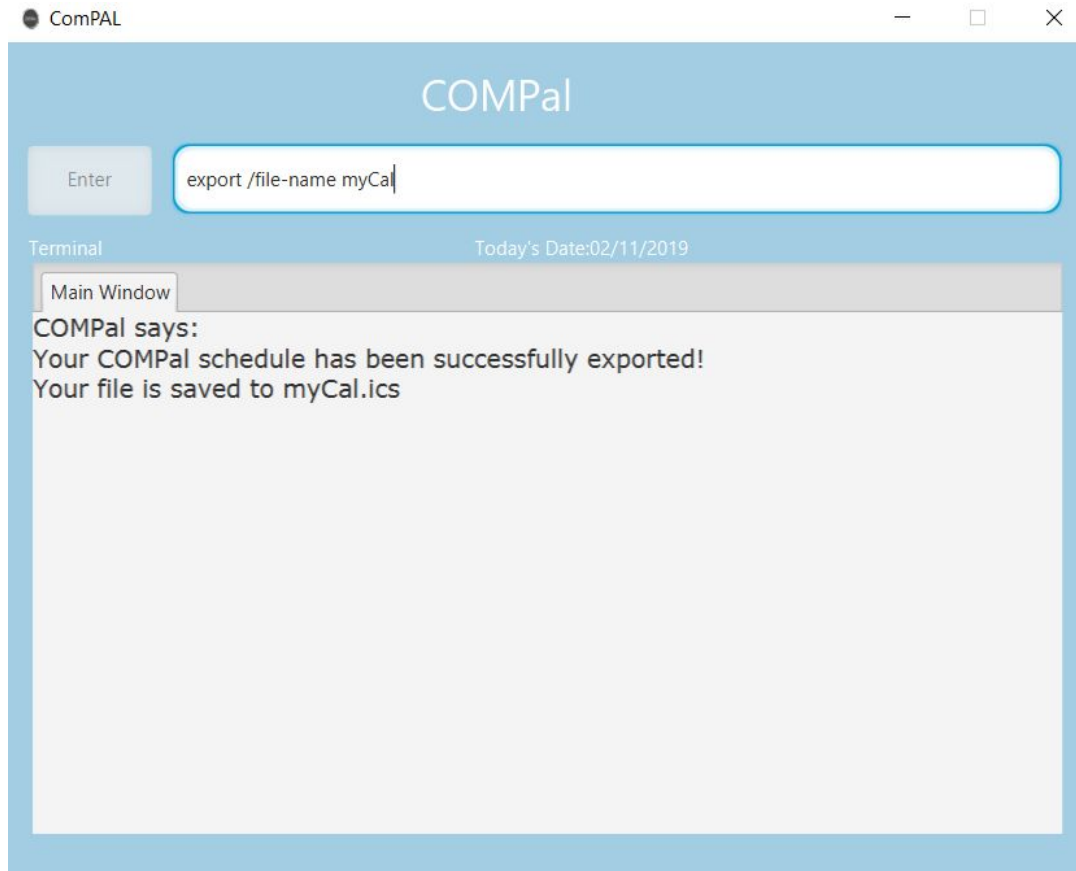
Examples:

- `Export /file-name myCal`

Figure 2. Expected output of `export /file-name myCal` command.

Importing iCalendar files: `import`

Want to import your COMPal schedule from another computer or maybe other calendar applications you have used into COMPal? Yes, you can import it as long the file is an `iCalendar` file!

Format: `import /file_name FILE_NAME`

Examples:

- `import /file_name myCal`

⚠	File must be in the same directory as **COMPal** launch application for this command to work!

Figure 3. Sample output of `import /file-name myCal` command with iCalendar file generated from google calendar.

⚠	This feature ***works best*** with the `iCalendar` file generated from COMPal. However, for `iCalendar` files that are not generated by COMPal, It will be imported if the event ***has a start and end time with no rules and timezone attached.*** Additionally, added events priority will be set to low for importing `iCalendar` not generated from COMPal.

Quick overview of the generated daily schedule.

After you have filled COMPal with **tasks** from your hectic schedule, you can use the `view day` command to look at your generated daily schedule.

The quick and intuitive interface enables you to quickly understand what deadlines are due and what clashing events you may have! Fret not as COMPal has sorted what is important and ***shows you the important tasks at hand that are not completed****!* Additionally, the more important **tasks** are placed on the left while the less-important one is to the right!

You ***should*** view the full information on the viewed **task** in the `main window`! As the daily **task** serves as a quick reference for you!

⚠	The Daily Task tab is static! Meaning you need to invoke `view day` to refresh your updated calendar if you were to add a task!
ⓘ	Only the **5 most important tasks** will be **displayed** for each **hourly slot** to allow you

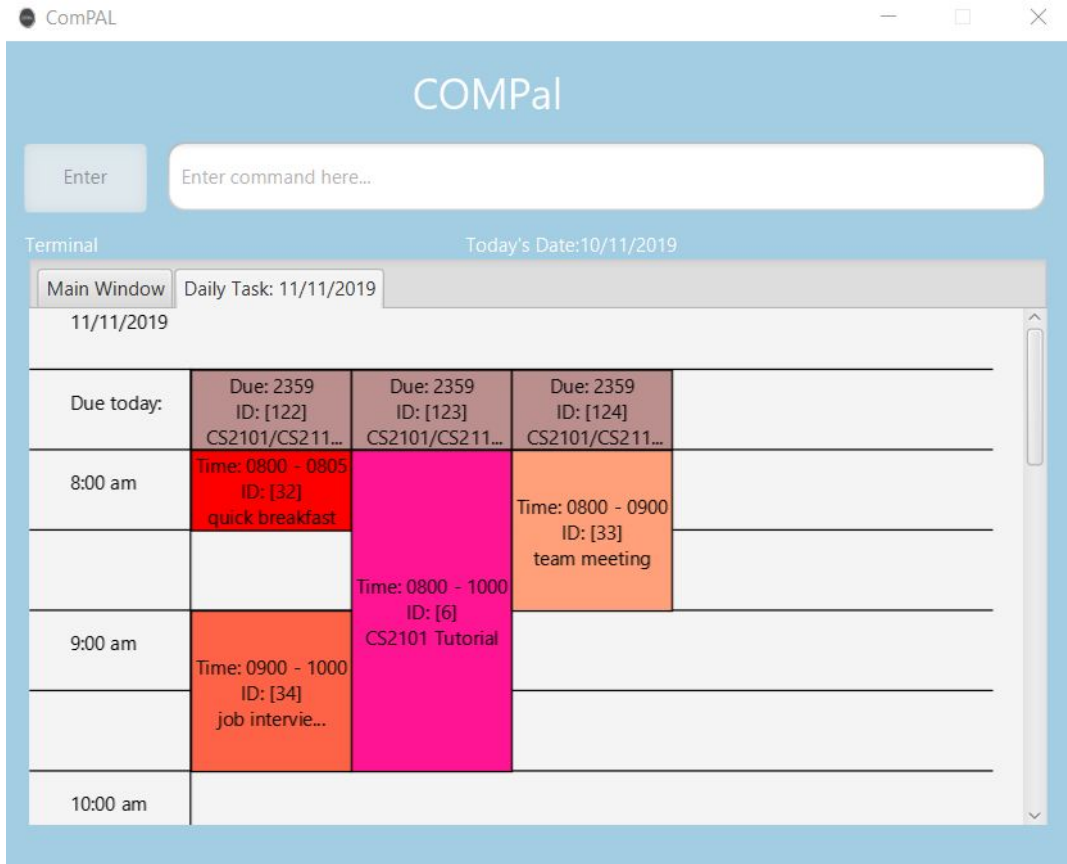|	
	to *focus on the tasks with higher priority*!

Figure 4. Generated planner for `view day` command in `Daily Task: DATE` tab. The most left schedule for each hourly slot represents the most important **task** to be completed first!

i	The below highlights are the color-coded type of **task** that can be seen when viewing COMPal: Non-school related event: High, medium, low priority School-related event: Lecture, Tutorial, Sectional or lab Task stored as deadline: deadline

Contributions to the Developer Guide

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

Listed below is my contributed sections in the Developer Guide:

- **Design**: Architecture Component,UI Component
- **Implementation**: View Feature,Export Feature
- **Appendix F**: Launch and Shutdown,Viewing the schedule,Listing all tasks,Changing tasks status,Exporting schedule into iCalendar file,Importing iCalendar schedule

You can click here to see the Developer Guide.

The below is an excerpt of my addition to the Developer Guide for the `export` feature.

Export Feature

This feature allows COMPal to exports its stored **tasks** into an iCalendar file. This section will detail how this feature is implemented.

Current Implementation

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

For clarity, the sequence of events will be in reference to the execution of an `export /file-name myCal` command. A graphical representation is also included in the Sequence Diagram below for your reference when following through the sequence of events. The sequence of events are as follows:

1. The `export /file-name myCal` 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 view command which in this case, is `ExportCommandParser`.

4. Once the parsing is done,`ExportCommandParser` would instantiate the `ExportCommand` object which would be returned to the `LogicManager`.

5. `LogicManager` is then able to invoke the `commandExecute` function of the returned `ExportCommand` object.

6. In the `execute` function of the `ExportCommand` object, **task** data will be retrieved from the `TaskList` component.

7. Now that the `ExportCommand` object has the data of the current **task** of the user, it is able to invoke the `creatIcsCal` function which converts all stored **tasks** of user into `ical4j Calendar model` .

8. Once all **tasks** are converted, using `ical4j calenderOutputer` api, which writes an iCalendar model to an output stream. The **task** converted will be saved into `myCal.ics`.

9. The `CommandResult` object would then be returned to the `LogicManager`, which then returns the same `CommandResult` object back to the `UI` component.

10. Finally, the `UI` component would display the contents of the `CommandResult` object to the user. For this `export /file-name myCal` command, the displayed result would be that the program has successfully exported to mycal.ics file.

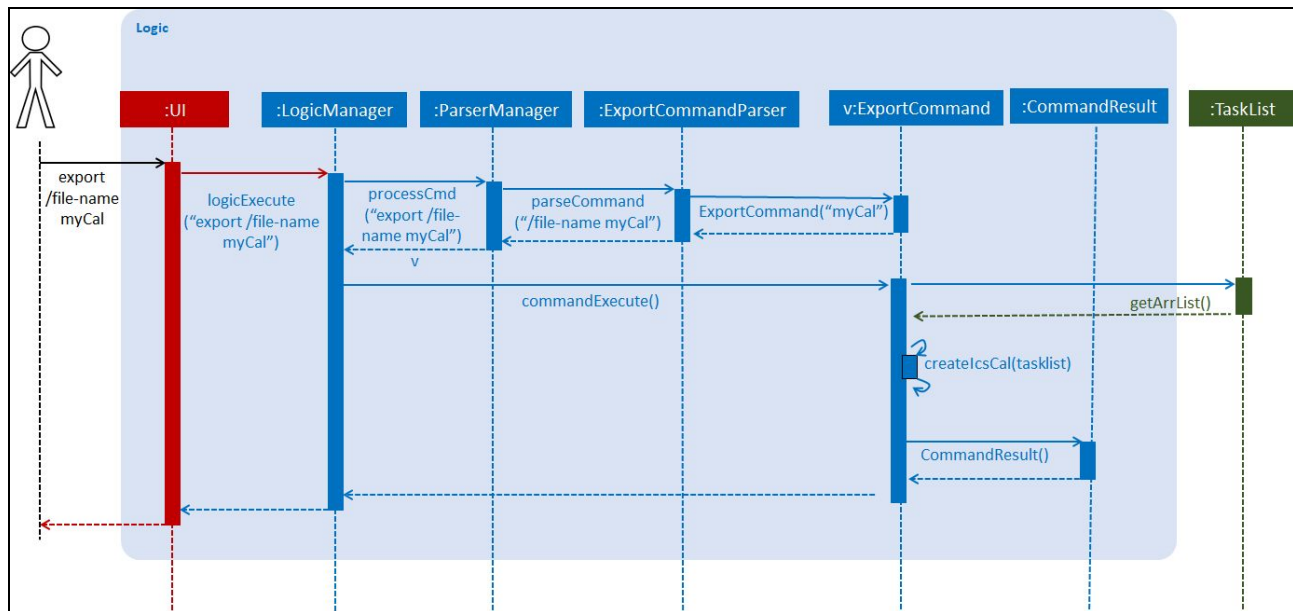Given below is the Sequence Diagram upon executing the `export` command:

Figure x. Sequence Diagram executing the `export /file-name myCal` command.

Design Considerations

This portion explains alternative implementations as well as the rationale behind my chosen method.

Aspect: Export type
- **Alternative 1 (current choice):** Using `iCal4j` library to convert **tasks** and write iCalender file.
  - **Pros:** Able to use `iCal4j` library to write `iCalendar` data streams. This **API** provides a quick and easy method to convert a current stored **task** to `iCalendar` data stream such as events.
  - **Cons:** Does not have a special field to store the **priority** object of each **task**.
- **Alternative 2:** Output all stored data into a text file or `.csv` file.
  - **Pros:** Able to store all needed fields stored in **COMPal** to be exported in files which can be used to import to **COMPal** application.
  - **Cons:** Only able to export and import files generated from **COMPal** application.

Alternative 1 was chosen because by using `iCal4j`, in which we are reusing tried-and-tested components, the robustness of **COMPal** can be enhanced while reducing the manpower and time requirement. Additionally, the ability to export into `iCalendar` file format which allows the users to import to any external calendar application enhances the usability of **COMPal**. Furthermore, a workaround to the **alternative 1** cons would be inputting priority field in ICS description field.

Future Implementation

Though the current implementation has much-allowed reusability of reliable open-source code and also allow COMPal to iCalendar files. There are still possible enhancement for COMPal to take advantage of ical4j library and the export command as currently we only create iCalendar `vevents` components.

1. Add `Todo` interface to the current **tasks** Model to allow the creation of `vtodo` components to export to other calendar application.
2. Improved `reminder` function and update the **task** model to create `valarm` component for **tasks** that have reminders.