

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, [COMPai](#), which was developed together with my team; Jun Yi, Jaedon, Pei Ze and Catherine.

PROJECT: COMPai

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 **COMPai**, which caters to NUS students who prefer to use a desktop application for managing their hectic schedule with chosen priorities levels.

COMPai 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 COMPai looks like:

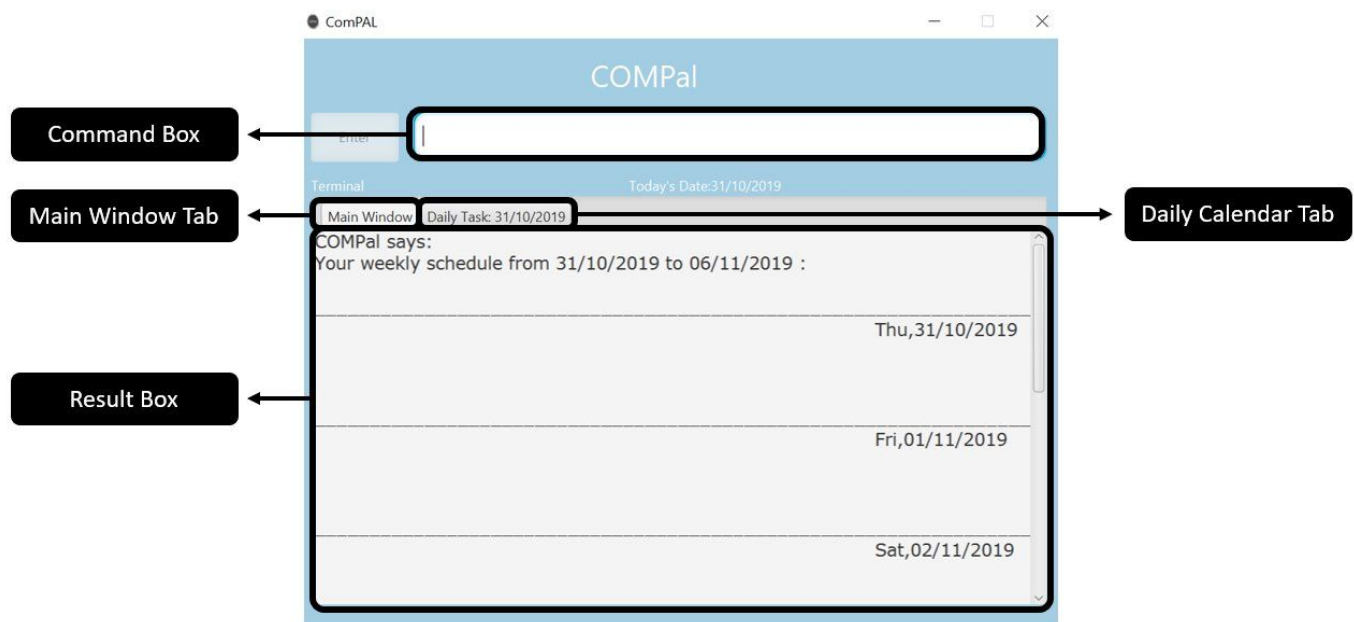




Figure 1. Description of COMPai components.

My role was to implement the graphical interface of COMPal, ensure that the tasks stored were sorted in chronological order and also the ability to `export` and `import` COMPal schedules. Additionally, I also implemented the `view`, `list` and `done` feature.

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 in relation to my implementations.

For ease of communication, this document will refer to classes/activities/events/deadlines that you might add into 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 view the users schedule in a daily view format
 - Daily calendar: I added the ability to view tasks stored in a specified day based on prioritisation levels.
 - What it does: The view day command generates a daily planner that shows the user what deadline and tasks that are needed to be done first.
 - Justification: Allows the application to be intuitive as the user is able to quickly grasp what deadline are due, events that occur throughout the day and which *tasks* that are needed to be done. Additionally reducing the need for the user to plan for their daily schedule.
 - Highlights: This enhancement works with existing as well as future commands. An in-depth analysis of design alternatives was necessary to compare the need of an intuitive GUI against a text output. The implementation was also challenging because of how our team wanted how intuitive the daily scheduler would look like.
- 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 existing features to increase overall coverage. I have ensured that the coverage of my contributed code is at 95%.
- 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

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

Listed below is my contributed sections in the User Guide:

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

Click [here](#) to see the user Guide!

Below is an example of my addition to the User Guide for the export,import and daily calendar feature.

Exporting COMPal schedule: export

Want to export your COMPal schedule to? Yes you can export it as an .ics file !

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

Check this link [here](#) to learn how to import to google calendar!

Format: export /file_name FILE_NAME



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

Examples:

- `Export /file-name myCal`



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

Importing .ics files: `import`

Want to import your schedule from COMPal or other application? Yes you can import it as long the file is an .ics file !

Format: `import/file_name FILE_NAME`

Examples:

- `import/file-name myCal`



	This feature works best with .ics file generated from COMPal. However , for .ics files that are not generated by COMPal. It can be imported with the exception of events stored in your exported calendar without start and end time . Additionally , added events priority will be set to low for importing .ics not generated from COMPal.
	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 .ics file generated from google calendar.

Quick overview of the generated daily schedule.

After you have filled COMPAL with *tasks* from your hectic schedule, you are able to 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 the more important tasks are placed on the left while the less-important one are to the right!

□	Only the 5 most important tasks will be displayed for each hourly slot to allow you to focus on the tasks with higher priority !
---	--

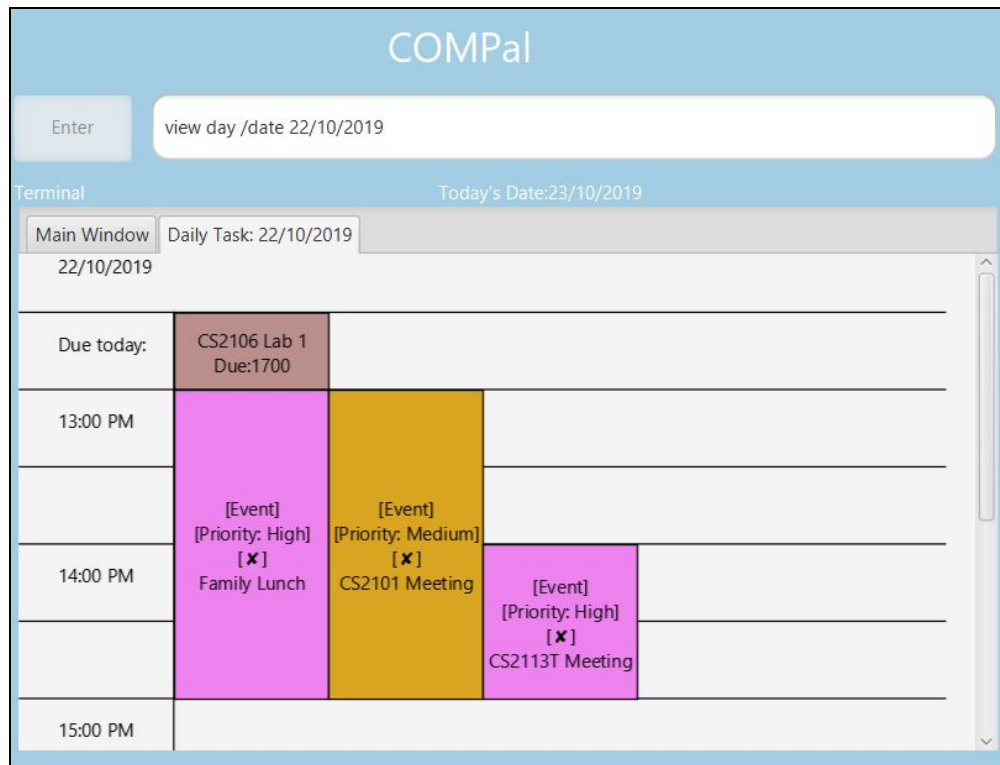


Figure 4. Generated planner for `view day` command in `view task: <DATE>` tab. The most left schedule for each hourly slot represents the most important task to be completed first!

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.

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 .ics file, Importing .ics schedule

Below is an example of my addition to the Developer Guide for the `export` feature.

Export Feature

This feature allows COMPal to exports its stored tasks into a .ics file. This section will detail how this feature is implemented.

Current Implementation

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

For clarity, the sequence of events will be in reference to the execution of a `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 calenderOutputter` 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:

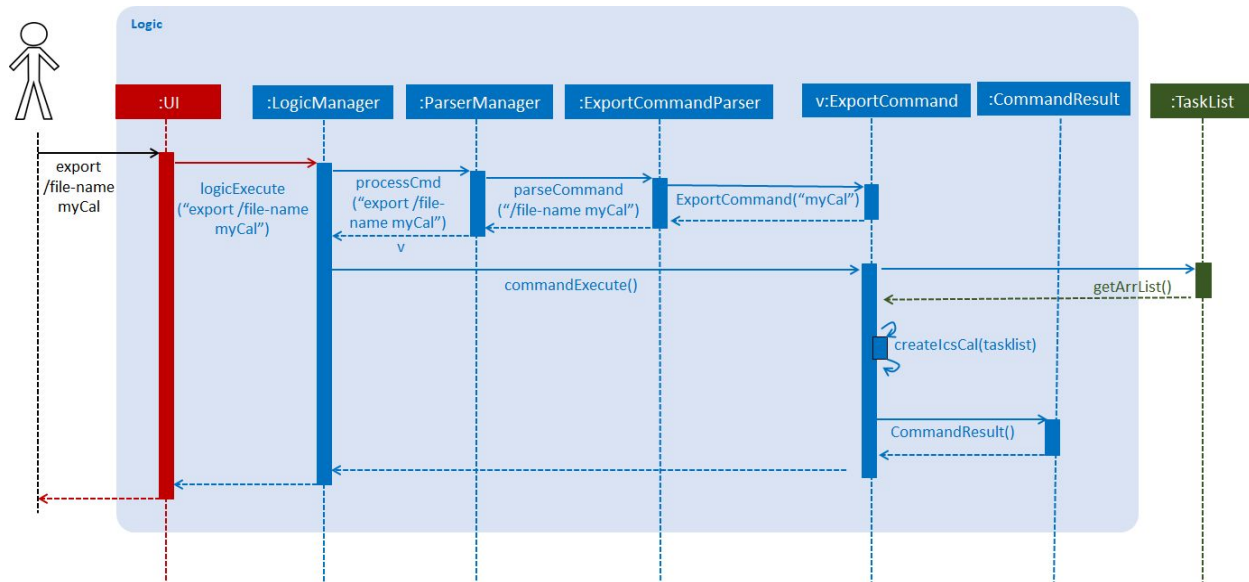


Figure 5. Sequence Diagram executing the `export /file-name myCal` command.

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 .ics file.
 - Pros: Able to use iCal4j library to write iCalendar data streams. Which API provides a quick and easy method to convert current stored task to iCalendar data stream such as events.
 - Cons: Does not have special field to store Priority object of each tasks.
- Alternative 2: Output all stored data into a text file or csv.
 - 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 .ics file format which allows the users to import to any external calendar application enhance 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 .ics 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 creation of [vtodo](#) components to export to other calendar application.
2. Improved `reminder` function and update `Task` model to create [valarm](#) component for tasks that have reminders.