**STATEMENT OF PURPOSE**

*Planner Application*

**Software development requirements & planning document**

The planner application will require:

* User input and output
* Scanner
* Sorting method
* If-else statement
* Timer
* Switch statement
* While loop
* LocalDateTime

**What does the application do?**

The planner-app application was designed to display the date and time to allow users to prioritize and set reminders of upcoming tasks for that current day.

**What business problem does it address?**

This application help assist individuals list out their daily tasks as well as setting reminders of which task to execute first. This allows users to set out their day in an organized manner and keep well on track with their schedules.

**Target audience?**

This application is mainly aimed more towards business individuals in the corporate world.

**Application features:**

* Intake of user input such as the agenda for the day, list of daily tasks and selection from options.
* Application output of stored data.
* Consistent output reminders of user’s to-do lists.

**Implementation Plan:**

Step 1:

Create empty GitHub repository

Step 2:

Create project in IntelliJ and committing to GitHub repository.

Step 3:

Have a running app with “Hello World” printed to console.

Step 4:

Print current local date time to console and create scanner method to take input from user. Commit and push to Git.

Step 5:

Print questions to console and allow users to input their day’s agenda and to-do lists using the collections sort method to sort the listing from 1(most important) onwards.

Step 6:

Created method to allow user to select from the options available using the if-else and switch block statement.

Step 7:

Added in reminder method to set reminders consistently of upcoming tasks.

Step 8:

Initialized testing.

**How to use the planner application**

Step 1:

Run application from Main.

Step 2:

Input today’s agenda.

Step 3:

Enter your to-do list numbered from 1(being most important) onwards. Submit your list by printing “end” once done.   
For e.g.,   
1. Report to manger

2.Reply back to emails

3.Meeting

“end”

If numbers are unsorted, the collections sort method will list it from ascending order.

Step 4:

If you would like to add to your tasks, select ‘1’, otherwise select ‘2’ to move on. If user did not select either option, it will ask to try again.

If option 1 was selected, user can add extra tasks into to-do list, remembering to number each task. Print “end” once all tasks are added. The application will again order from ascending order.

Step 5:

Now that tasks are up to date, the number of tasks listed will be the same number of reminders alerted. Application will conclude with “Task completed”

**Acknowledgements**

Teacher: Brendon

<https://www.youtube.com/watch?v=0kLegsyaVf>

<https://www.youtube.com/watch?v=YgkRoY3-ng8>

[Ways to read input from console in Java - GeeksforGeeks](https://www.geeksforgeeks.org/ways-to-read-input-from-console-in-java/)

[Java Scanner Tutorial - YouTube](https://www.youtube.com/watch?v=Wgkb0zg7WOM)

[Java basics of the LocalDate, LocalTime, LocalDateTime, ZonedDateTime and the DateTimeFormatter - YouTube](https://www.youtube.com/watch?v=0XgdX5hDL4U)

[Java TimerTask ⌚ - YouTube](https://www.youtube.com/watch?v=QEF62Fm81h4)

[Day19: Creating a console selection menu in Java - YouTube](https://www.youtube.com/watch?v=zCjY59M8qtY)

[Insertion Sort Java Algorithm - 2 Ways | Java Sortings (javatutoring.com)](https://javatutoring.com/insertion-sort-java/)