Team Members: Jayda C., Nhung N., Sar V., Shriya K., Thais R

Test Case 1: Task Overlap

Input:
1. A task for a specific date and time
2. A specific date and time

Test:
1. Test whether the new task coincides or clashes with any already published tasks.
2. Test whether the new task has a unique date and time.
3. Test whether the timing dedicated to the task does not overlap another task

Output:
Provide an error message stating that the task overlaps another task.

Test Case 2: Task notes

Input:
1. Task notes for a specific task.(For eg: add note to Design and Analysis midterm: practice study guide)

Test:
1. Test whether the following notes are added to the task and to that task only.
2. Test if the notes are displayed when the task is due.

Output:
Should output notes successfully added or edited whenever a user adds notes to a task or goes in and edits them later.

Test Case 3: Task Progress

Input:
1. A task with a dedicated timer.
2. A time assigned to that task

Test:
1. Test whether the timer of the task overlaps another task.
2. Test whether the timer is displayed when the task is taking place.

Output:
Provide an error message from test case 1 if overlap is detected, otherwise show the timer when a task is occurring.

Test Case 4: Dynamic Reminders

Input:
1. Task with input: “Submit end of month review for scrum sessions.”

2. Task with input: “Attend scrum session on Tuesday.”

Test:

1. Test whether the given tasks are due in a day, a month, the following week etc.
2. Set a reminder based on the time that is present between now and the task.

Output:

Task for the end of month review should be reminded one day before the task is due, whereas a task for a team meeting on Tuesday should be reminded within an hour before that task.

Test Case 5: Track Task Progress

Input:

1. Setting a task as completed, postponed, or in-progress.
2. Mark task as either complete, postponed, or in-progress.

Test:

1. Check for the input from the user and if it is completed, in-progress or postponed.

Output:

Task marked as completed should be removed, task marked as in progress should display a timer, and task marked as postponed should ask for another date and time to add.

Test Case 6: Task Organization

Input:

1. Setting tasks as high priority vs. low priority.
2. Test whether or not a task marked as high priority appears at the top of the list.

Test:

1. Check for the input from the user (whether or not the user marks the task as high priority).

Output:

Tasks that user marks as high priority should move to the top of the list, whereas other tasks without this marking would be considered as low priority.

Test Case 7: Security Since Task Monitor May Have Sensitive Information

Input:

1. User inputs a username and password

Test:

1. Check whether the username and password associated with that username is matching the data or not.

Output:

If the username and password is entered correctly, grant user access, otherwise if the user enters the wrong password 3 times, lock the user out and send an email to the corresponding email address.

Test Case 8: Reliability and Ability To Perform Under Heavy Workload

Input:

1. Verify that the system works properly under heavy usage and high amounts of data being transferred within the database and the UI, and code.

Test:

1. See that all systems are working properly while stress testing the code and the system.
2. Verify that the system is still able to perform under heavy usage and still have a usable and efficient response time.

Output:

Users should be able to use the system even if they have a high amount of tasks/reminders.

Github Repository With Links to Files:

https://github.com/jcaviness1/catcommandos