Progress Report for Final Sprint

Vision

For this project we are planning on building a command-line productivity app. Users will be able to enter commands to create tasks with a description, deadline, priority and category. After constructing one or more tasks, the user will be able to issue a command to list all tasks. Additionally, the user would also be able to list all tasks with an upcoming deadline, filtering out the tasks whose deadline has already passed. After completing a task, the user can mark it as complete—this will hide it from the command that lists the tasks with an upcoming deadline. If the user fails to complete a task, or simply wants to delete a task permanently from storage, he or she can execute a remove command. The user can also type in a command to see the recommended order in which to complete tasks. He can also issue a command to see all the overdue tasks he has. He can also issue a command to see a specific task color coded to indicate its urgency. For storage of task items, we plan to use JSON and will use Yojson to load the data.

Summary of Progress

As a team, we've accomplished a considerable amount of work this sprint. Users may create tasks with priorities as well as categories they fit in (such as school or work); they may now also complete and remove a task not only by its id but also description; users can also now type a toDo command that shows them the recommended order in which to complete tasks as determined by the scheduling algorithm which accounts for priority as well as deadline. They can also now see tasks that are overdue. In addition, they can see a specific task either by its id or description which when printed will provided colored based feedback to the user indicating whether it is completed(green), urgent (due in a day i.e red), not that urgent (yellow), and if that task does not have a deadline or is overdue then the task is white.

Activity Breakdown

Rafael Jacobovitz

- Parsing new commands into strings
- Added Overdue command and functionality
- Added the tags field and functionality for tags
- Cleaned up multiple functions throughout code

Phillip Cipollina

- Writing test cases
- Code cleanup
- Implementation of priority field, toDo command

- Updating of complete and remove to work on both ids and descriptions
- Fixed futureDue to include all tasks without deadlines and to also filter by deadline, rather than just completion status

Ahmed Moustafa

- Implementing the data-driver module
- Implementing the task module
- Writing unit tests for the data-driver and task modules
- Utilized the bisect system to ensure all systems were well-tested.

Kartikay Jain

- Color coded printing
- Printing modifications made to account for new fields such as priority, tags, as well as there being no deadlines etc. as well as borders added to make Todo list look better.
- See command and help command functionality

Productivity Analysis

As a team, we were extremely productive. We utilized productivity tools like Trello, Slack, and WhenIsGood to stay organized and efficient. After organizing team meets using WhenIsGood, we split the work up into smaller subproblems and posted each subproblem on the Trello board, assigned to specific team members along with a due date. We also used a Slack group to stay informed on each other's progress. We accomplished everything we planned for this sprint, and more. Our estimates of what we could do were very accurate, considering our plans for the project were completed very efficiently in an organized manner.

Coding standard grades

Documentation Grade: Meets Expectations - as we provide ocaml docs for most of the important functions (which includes some helpers) both in ml and mli
Testing Grade: Exceeds Expectations - we did both glass box and black box testing for our functions in different test suites as well as utilized bisect system to get as much coverage as possible.
Comprehensibility Grade: Meets Expectations - for the most part we set reasonable names for our functions that described what they did as well as used the best syntax applicable when coding a specific function.
Formatting Grade: Satisfactory - We could have improved formatting more by making sure no code passes 80 character limit.

Scope grade

Here's what we consider a satisfactory, good, and excellent solution for the Final Sprint:
Satisfactory: The solution submitted to CMS compiles. The see command functionality is implemented so you can see a specific task color coded with how urgent it is.
Good: The solution additionally re-implements make with a tags and priority field. All commands that act on an id can also instead act on the description. The scheduling algorithm is implemented which based on priority and deadline gives the order in which to do tasks. Overdue task functionality is also implemented.
Excellent: The solution additionally implements the following ideas: a much more enhanced GUI, a setDeadline function that allows you to modify deadlines. seeAll command prints tasks in different orders depending on a second argument (for eg. seeAll deadline prints tasks in ascending order of their deadline, seeAll priority prints tasks in ascending order of their priority). Have tasks be able to repeat weekly or biweekly as well as add graphing functionality like gnuplot where y axis represents task id and x-axis is time taken to complete a task.

We gave ourselves a Good for this sprint. This is because we feel we did the barebone for extending our original basic todo list by adding in the scheduling algorithm, color coded printing as well as additional commands to make it more user friendly for the user.