PLANORAMA

Sprint 2 Retrospective

Team 22: Sruthi Bhamidipati, Yana Sheoran, Vihaan Khajanchi, Jack Kenny

What went well?

This sprint was fairly successful, with several major user stories completed that significantly improved the platform's usability, organization, and collaboration features. Subtask functionality was fully implemented with real-time updates, editing, deletion, and visual consistency across views. The drag-and-drop feature allowed for intuitive task reordering with helpful visual feedback and persistence across sessions. Task dependencies added meaningful structure to task flows, preventing completion of dependent tasks until prerequisites were met, and were visually represented in multiple views. The archival system helped reduce clutter by moving completed tasks out of the main workspace, while still allowing retrieval and manual or automatic archival. Additional quality-of-life improvements such as note-taking, task templates, and future task scheduling enabled better planning and organization. Time logging and a streak tracker added valuable productivity insights and motivation for consistent progress. Finally, the introduction of team management and user search made collaboration more seamless, with intuitive invite handling and user profile previews. Overall, the sprint delivered a robust and cohesive set of features that enhanced both individual and team workflows.

User Story #1: As a user, I would like to be able to create subtasks within a task.

1	Create UI for displaying subtasks under a parent task.	3 hrs	Sruthi
2	Implement logic to allow users to add subtasks dynamically.	3 hrs	Sruthi
3	Implement functionality for editing or deleting subtasks.	3 hrs	Sruthi
4	Ensure subtasks can be marked as complete, and reflect their status in the parent task.	2 hrs	Sruthi

Completed:

The subtask feature is fully implemented within the task modal overlay, allowing users to add, edit, and delete subtasks directly in the modal. An input field appears within the overlay when "Add Subtask" is clicked, and each subtask can be edited inline or removed with a confirmation prompt. Subtasks can be marked as complete from both the modal and the main task table, with visual feedback—such as a checked box and strikethrough styling—consistently displayed in both views. Changes are saved in real time and persist across sessions.

User Story #2: As a user, I would like to drag and drop tasks so that I can easily visualize my task order.

1	Design and implement the UI to allow dragging and	4 hrs	Sruthi

	dropping tasks within a task list.		

2	Implement logic to update task positions dynamically when a task is moved.	3 hrs	Sruthi

3	Ensure task order persists after refreshing or navigating away from the page.	3 hrs	Sruthi

4	Add visual feedback (e.g., highlight, placeholder) while dragging a task.	1 hr	Sruthi

Completed:
The drag-and-drop task reordering feature is complete and fully functional. Users can now click and hold on a task row to initiate dragging, with the selected task following the cursor. While dragging, a visual placeholder appears between tasks to indicate potential drop positions, helping users understand where the task will be placed. When a task is dropped in a new location, the task list updates immediately to reflect the new order. If a task is dropped back into its original position or outside of a valid drop zone, no changes are made. All updates to task order persist across page refreshes, ensuring the user's preferred order is maintained.

User Story #3: As a user, I would like to be able to set task dependencies to organize connected tasks better.

1	Design and implement UI for setting dependencies between tasks.	3 hrs	Sruthi

2	Implement logic to allow users to assign one task as dependent on another.	3 hrs	Sruthi

3	Ensure tasks with dependencies cannot be marked as complete until prerequisite tasks are completed.	2 hrs	Sruthi

4	Visually represent dependencies.	1 hr	Sruthi

5	Ensure task dependencies persist after refreshing or navigating away from the page.	2 hrs	Sruthi

Completed:
The task dependency feature is now fully implemented, allowing users to create and manage dependencies between tasks directly from the Dependencies page. When a user clicks "+ Add" under a task in the outline, a dropdown menu appears with all eligible tasks that can be set as dependencies. Once selected, the dependency is instantly reflected across all views—appearing as an arrow in the graph, a nested item in the outline, and an entry in the Dependencies column of the task table. Tasks with unresolved dependencies cannot be marked as complete; attempting to do so triggers a clear error message. Once all prerequisite tasks are completed, the dependent task can be marked complete without restriction, and its status updates across all visual representations. Removing a dependency cleans up

all related visuals and re-enables task completion. All dependency relationships persist after refreshing or navigating away, and the layout remains consistent.

User Story #4: As a user, I would like to archive completed tasks so that I can keep my workspace clutter-free.

1	Implement a "Mark as Complete" button/check for tasks.	1 Hr	Jack
2	Implement change in task to make the task noticeably complete.	1 Hr	Jack
3	Add an "Archive" section where completed tasks can be stored and accessed later.	3 Hrs	Jack
4	Add an “Archive” button next to completed tasks	1 Hr	Jack
6	Ensure that archived tasks are retrievable but do not clutter the active task board.	1 Hr	Jack
7	Add a dropdown to set how long a task stays until archival.	2 Hrs	Jack

Completed:
The archival feature was working in its base form. When a user archives a task it successfully is removed from the task list and put into the archival list, this is as was expected of this feature. This allows users to complete tasks and not have them clutter their main task interface even after they are finished. This is good for productivity and overall task management. If a user is to accidentally archive a task, or realize they were not fully done when they archived it, they are also able to go to the archived list and unarchive the tasks from the list. As for the auto-archival feature, the layout is complete and the buttons associated with the feature are in the archive page.

User Story #5: As a user, I would like to jot notes down about tasks that I have or have not completed so that I can reference thoughts for later.

1	Implement a “notes” button for each task.	1 Hr	Jack
2	Implement an interactive UI that allows users to put notes in.	1 Hrs	Jack
4	Add a delete notes button in the notes UI.	1 Hr	Jack

Completed:

The notes feature is present for every task, so if a user has any additional information they would like to note or remember about the task, they are able to do so. The notes are also separated from task to task, so making a note in one task does not interfere with the others. Finally, the notes feature is able to be closed, so a user is able to keep their interface clean if they do not want to see the notes for a task anymore.

User Story #6: As a user, I would like to set default task templates so that I can quickly create repetitive tasks.

1	Make default task creation button	1 Hr	Jack
2	Develop and implement UI to hold all task templates	2 Hrs	Jack
3	Create task template editing, creation, and deletion buttons.	1 Hr	Jack

Completed:
The template UI is functional and the user is able to pull it up to see the interface and close it when they no longer need it. The user is also able to click the create template button that pulls up a menu for them to input what things they would want the template to have. When the user is in this interface they are able to save the template and also close the UI if they decide not to finish it.

User Story #7: As a user, I would like to schedule tasks for a future start date so that I can plan ahead without cluttering my current task list.

1	Add a "Start Date" field to the task creation form	1 Hr	Yana
2	Modify the database schema to store future task start dates	1 Hr	Yana
3	Update task retrieval logic to exclude tasks with a future start date	2 Hrs	Yana
4	Implement a mechanism (cron job or scheduled check) to move tasks to active when the start date is reached	2 Hrs	Yana
5	Enable users to manually start a scheduled task before its start date	2 Hr	Yana

Completed:
The scheduled task feature has been successfully implemented, allowing users to assign a future start date when creating a task. A new “Start Date” field has been added to the task creation form, enabling seamless planning ahead without overcrowding the current task list. The database schema has been updated to store this date, and the task retrieval logic now excludes tasks with future start dates from the main task view. A background mechanism periodically checks for tasks whose start date has arrived and automatically moves them

to the active list. Additionally, users have the flexibility to manually activate scheduled tasks before their designated start date. All scheduling changes are handled in real time and persist across sessions, ensuring a smooth and consistent user experience.

User Story #8: As a user, I would like to log time spent on tasks so that I can keep track of progress.

1	Add a "Time Spent" field to the task model in the database	1 Hr	Yana
2	Update the UI to allow users to manually enter time spent on a task	1 Hrs	Yana
3	Store multiple time log entries per task and sum the total time	1 Hrs	Yana
4	Display logged time in task details and task summary	2 Hrs	Yana
5	Allow users to edit or delete logged time entries	1 Hrs	Yana

Completed:
The time logging feature has been fully implemented, which allows users to track the time they spend on individual tasks. A "Time Spent" field has been added to the task model, and users can now manually log time entries through an intuitive UI embedded within each task's detail view. Each task supports multiple time log entries, which are automatically summed to show the total time spent. Logged time is displayed clearly in both the task details and the task summary view, helping users stay informed on their progress. Users also have the ability to edit or delete specific time entries, giving them full control over their time tracking. All updates are saved in real time and persist across sessions, ensuring a seamless and reliable experience.

User Story #9: As a user, I would like to see a streak tracker to visualize my consistent task completion.

1	Modify the database schema to track task completion dates per user	2 Hrs	Yana
2	Implement logic to calculate and update streak counts based on consecutive days of completion	3 Hrs	Yana
3	Design a UI component to display the streak tracker (e.g., a progress bar or calendar view)	2 Hrs	Yana
4	Show current streak, highest streak, and total completed days in the UI	2 Hrs	Yana
5	Allow users to view past streak history and performance	2 Hrs	Yana

	trends		

Completed:
The streak tracker feature is now live, giving users a visual and motivational way to monitor their task completion consistency. The database schema has been enhanced to store task completion dates for each user, enabling accurate tracking over time. A backend logic calculates streak counts based on consecutive days of task completion and updates the user's current and highest streaks dynamically. The front-end now includes a sleek streak tracker UI component, with a calendar-style view. The tracker clearly displays the user's current streak, highest streak, and total number of days with at least one completed task. Additionally, users can view their past streak history and see the performance trends, offering meaningful insight into their productivity habits. All streak data is updated in real time and persists across sessions.

User Story #10: As a team leader, I would like to create a "team" so that I can manage group tasks.

1	Add a new button to navigate to a teams page	30 mins	Vihaan
2	Add a "create team" button that will prompt you to create a name for the team	1 hr	Vihaan
3	Any teams created will show up on the leader's teams page	1 hrs	Vihaan
4	Can click on a button to open that team and view its details, like members and tasks	3 hrs	Vihaan
5	Within a team, add a button for the team leader to add members to the team	2 hrs	Vihaan
6	Add a button for the team leader to delete their team, erasing all data and members	2 hr	Vihaan

Completed:
Created a new page for all team management. You can click a button to enter a name and create a team, which will then show up on the page and then can be clicked to view and edit that specific team. There are options to add members, delete the team if you are an owner, and leave the team if you are a member. The information on the team such as members, owner, and name appear on each unique team page.

User Story #11: As a user, I would like to be able to search for other users so that I can use other functionality like teams.

1	When adding members to a team, add a search bar where you can type in a username and any matching usernames will show up	3 hrs	Vihaan

2	Each username should have a “view” and “add” button next to it	2 hrs	Vihaan

3	Also add the same search feature to home page, except with just “view” button	1 hr	Vihaan

4	“view” button will preview their profile page	3 hrs	Vihaan

Completed:
There are two places where the search feature can be accessed. The first one is when adding members to a team. After clicking add member, a search bar will appear where you can enter some letters and any usernames that match the pattern will show up as you type. Alongside their username there will be two buttons, to view or invite that user. The second place where you can search is from navigating to a search page using a button on the navigation bar. From this search bar, it will show any users you search for except they will only have the view button. When clicking the view button in any search bar, it will show you the profile of whoever you select.

User Story #12: As a team leader, I would like to be able to add users to a team.

1	When clicking “add”, send an invite to the user, which will show up in their teams page	3 hrs	Vihaan

2	Once added, the option will be greyed out unless the member denies the invite	2 hrs	Vihaan

3	Users can choose to join or deny an invite	2 hrs	Vihaan

4	If joining, the team should show up in their teams page	1 hr	Vihaan

Completed:
When searching users when adding to a team, users have an invite button next to their name. You can click invite and it will send them an invite to their team page. For anyone already invited or already a member, the invite option will not appear. Once the invite is sent, it will appear in a box in the bottom right, showing the team name and owner of the team, with options to join or deny. If the recipient clicks deny, the invite will disappear and the invite option will appear again when searching for them. If they click join, they will become a member of the team and the team will show up on their own page.

What did not go well?

While the sprint delivered several important features, a few areas encountered setbacks that impacted overall progress. The undo functionality for task deletion remains incomplete, with key components like the 10-second undo window and recently deleted task storage still pending implementation. The implementation of user profile picture uploads faced backend integration issues with the existing authentication system, preventing successful storage and linking of uploaded images. The archival feature, although partially functional, is incomplete—archived tasks reappear on page refresh, and the auto-archival system with time-based triggers and UI controls remains non-functional. The notes feature also lacks critical capabilities, such as text formatting and persistence across

sessions, which limits its reliability for users wanting to retain detailed information. Additionally, the task template functionality was not completed; users are currently unable to save or retrieve templates, which also halted progress on related features like template editing, favoriting, and the “most recent” view.

Sprint 1, User Story #4: As a user, I would like to be able to delete tasks.

3	Implement undo functionality for accidental deletions.	2 Hrs	Jack

Not completed:
The feature for holding the last 10 deleted tasks still needs to be pushed to the main branch. Also, the feature for giving the user 10 seconds to decide to undo before it is officially deleted still needs to be added.

Sprint 1, User Story #11: As a user, I would like to be able to change my profile picture so that I can customize my account.

1	Update backend to store the uploaded profile picture in the database	2 Hrs	Yana

Not completed:
The backend update to store profile pictures faced issues due to conflicts with existing user authentication logic. As a result, the upload couldn’t be properly linked or persisted to the user profile.

User Story #4: As a user, I would like to archive completed tasks so that I can keep my workspace clutter-free.

5	Implement an auto-archive function that moves completed tasks after a set time.	3 Hrs	Jack

8	Add a time of day button next to the auto archive dropdown	1 Hr	Jack

9	Add time of day UI	1 Hr	Jack

10	Sync time of day with auto archival dropdown functionality	2 Hrs	Jack

Not completed:
The backend functionality for archived tasks is not complete, so when a user refreshes the page the tasks will be put back into the main task page. Also, the auto-archival functionality and its features are not complete. So when a user changes the buttons and UI associated with the feature, it does not currently do anything.

User Story #5: As a user, I would like to jot notes down about tasks that I have or have not completed so that I can reference thoughts for later.

3	Implement text editing interface within notes UI with basic formatting button options.	2 Hr	Jack

5	Make notes persist across multiple sessions and have them saved in the database.	2 Hrs	Jack

Not completed:
The formatting features like bold, bullet, and italics were not completed. When the user highlights their text or puts their cursor in the text box and clicks a formatting button, they are still not able to type with that new formatting. Also, when a user refreshes the page or logs out, the notes they made do not persist across sessions.

User Story #6: As a user, I would like to set default task templates so that I can quickly create repetitive tasks.

3	Create task template editing, creation, and deletion buttons.	1 Hr	Jack

4	Create and implement “favorite” feature for default task templates	2 Hrs	Jack

5	Implement “most recent” button	1 Hr	Jack

6	Ensure that the task templates persist across sessions and are saved in the database.	2 Hrs	Jack

Not completed:
The creation of the template is not functional, so when a user clicks save template, the template is not actually created. This means the template board is not ever filled, meaning the user is not able to see any templates. Because of this the favorite feature is also not complete or the editing and deletion. Finally, the “most recent” button is not implemented as there are no templates to have as more recent.

How should we improve?

During Sprint 1, we struggled with being intentional enough in defining our acceptance criteria, which led to vague expectations and shifting scopes. In response, we overcorrected in this sprint by writing highly detailed and rigid acceptance criteria for many user stories. While this helped clarify deliverables and reduced the need to change our goals mid-sprint, it also introduced new challenges. The abundance of low-priority, granular requirements made tasks more time-consuming and cognitively demanding, increasing the likelihood that we would either forget features or run out of time to complete them. It also made it harder to focus on delivering core functionality, since we were often preoccupied with edge cases and small UI elements. In future sprints, we plan to strike a better balance—being more precise and intentional than we were in Sprint 1, but avoiding the overly prescriptive approach we took this time.

Another area that warrants improvement is backend integration, which proved more fragile and error-prone this sprint than in the previous one. In several cases—including profile picture uploads, task templates, and notes—the user interface was built out, but functionality remained incomplete due to backend operations not working as expected or failing to persist data correctly. These issues

revealed gaps in how backend logic was validated during development and ultimately impacted our ability to deliver fully functional features. Ensuring that database operations, API interactions, and authentication flows are thoroughly implemented and tested alongside the UI is critical for delivering a seamless user experience.

Moving forward, we should treat each feature as a cohesive unit of work that includes both the interface and its supporting logic, rather than tackling them sequentially. This means placing greater emphasis on backend reliability during development and testing features end-to-end before considering them complete. By reinforcing this approach, we can reduce rework, catch integration issues earlier, and improve the overall stability and quality of the product delivered at the end of each sprint.