PLANORAMA

Sprint 3 Retrospective

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

What went well?

Most user stories went well, as we were able to add a lot of final important features that help polish the final product. Recommended subtasks really completed the subtask feature in a way that made sense, we were able to get the productivity feature up and running, and we added a notification system which is vital for a project involving planning tasks with deadlines. We also added vital productivity features like seeing estimated vs. real time and helped promote healthy habits by suggesting breaks. The most important features were also added to the teams feature in order to wrap it up, including adding tasks, assigning them to members, marking tasks as complete, adding comments to team tasks, and having display names. More details were added to tasks as well, like roll-overs for incomplete tasks. Finally, we were able to finish some incomplete tasks from previous sprints that were important to add before the project ended, like undoing accidental deletions, archiving tasks, and more.

User Story #1: As a user, I would like to receive smart suggestions for breaking up long tasks into smaller subtasks.

1	Create a suggestion bank that maps common task phrases (e.g., “write essay”, “plan trip”) to curated subtasks.	1.5 hrs	Sruthi
2	Implement keyword/fuzzy matching logic to detect when a task matches an entry in the suggestion bank (e.g., lowercasing, stopword filtering, partial token overlap).	2 hrs	Sruthi
3	Use lightweight NLP (e.g., with spaCy or nltk) to segment long task descriptions into sentence-like chunks or clauses as a fallback when no match is found	2.5 hrs	Sruthi
4	Design a frontend suggestion prompt that shows generated subtasks and allows users to accept, reject, or edit the list.	1.5 hrs	Sruthi
5	Implement editable subtasks in the UI and convert accepted suggestions into actual subtasks tied to the main task.	2 hrs	Sruthi

Completed:

This user story aims to streamline task creation by providing smart subtask suggestions when users enter long or complex task descriptions. The system first checks if the task matches a known phrase in a suggestion bank using keyword or fuzzy matching. If no

match is found, it falls back on a delimiter-based approach, splitting the task description into smaller steps using punctuation or common conjunctions. A frontend prompt then displays these suggestions, allowing users to accept, edit, or reject them. Accepted subtasks are added to the main task, edited suggestions are added after confirmation, and rejected suggestions are permanently removed from the list. All accepted subtasks behave like regular ones—editable and completable—when the user revisits the task.

User Story #2: As a user, I would like to receive a weekly summary of my most and least productive days so that I can adjust my workflow.

1	Define productivity metric (e.g., number of completed tasks per day, optionally weighted by priority/status if needed).	1 hr	Sruthi
2	Implement backend logic to aggregate completed task data for the current calendar week (Sunday to Saturday).	2 hrs	Sruthi
3	Compute most and least productive days based on task completion count.	1 hr	Sruthi
4	Design frontend component to display a weekly breakdown (bar graph or list view with 7 days, highlight top/bottom).	2 hrs	Sruthi
5	Add a “Weekly Summary” page with the ability to view the current and previous week(s).	2 hrs	Sruthi
6	Add date filtering logic to view previous full weeks (e.g., “Week of Mar 24 – Mar 30”).	1.5 hrs	Sruthi

Completed:
This functionality is implemented on a "Weekly Summary" page designed to help users reflect on their productivity across the current and previous weeks. Productivity is quantified through a combined score based on completed tasks and logged time, aggregated for each day from Sunday to Saturday. The summary is visualized using an area chart that displays one line across all seven days. For days in the future (relative to the current date), the chart remains flat, indicating no data, and those days are excluded when identifying the most and least productive days. Below the chart, the most and least productive days are labeled using a comma-separated format in the event of ties, giving users immediate insight into patterns. The page also supports interactivity: when users hover over any day on the chart, a detailed tooltip appears. This tooltip includes a “Tasks Completed” section listing all tasks finished that day (or stating "no tasks completed" if none), a “Time Logged” section showing time logged per task (or “no time logged” if empty), and the computed productivity score. Even if only task completions or only time logs exist for a day, the system still visualizes whatever data is available. Additionally, users can explore historical productivity by selecting previous weeks via a dropdown filter, with the data and labels dynamically updating to reflect the selected timeframe. All of these elements work together to provide a comprehensive and responsive view of the user’s workflow habits.

User Story #3: As a user, I would like to receive notifications so that I can be reminded of important tasks and stay productive.

1	Define rules for triggering notifications (e.g., tasks due today, overdue, or marked high priority) and ensure logic runs at the beginning of each session.	1 hr	Sruthi
2	Build backend logic or middleware that compiles notification data on session load (or user login) based on task metadata.	2.5 hrs	Sruthi
3	Design a notification UI (e.g., bell icon or dropdown panel) with badge and expandable list of notifications.	1.5 hrs	Sruthi
4	Implement frontend logic to show notifications on session start unless the user has opted out in settings.	2 hrs	Sruthi
5	Add per-notification and “dismiss all” functionality that hides notifications for the current session only.	1 hr	Sruthi
6	Create settings option to enable/disable task notifications.	1 hr	Sruthi

Completed:
This feature introduces a session-based notification system that alerts users about important tasks—those due today, overdue, or marked as high priority. At the start of each session, backend logic identifies qualifying tasks and displays them in a notification panel accessed via a bell icon. Users can dismiss notifications individually or all at once, hiding them for the remainder of the session. The system also updates in real time: if a task becomes urgent or its status changes to meet notification criteria, a new notification is added immediately. Conversely, if a task is completed or no longer qualifies, its notification is removed. Clicking a notification highlights the corresponding task in the task table with a temporary black border, drawing clear visual attention. The highlight disappears once the user interacts elsewhere. A settings toggle allows users to enable or disable notifications, giving them full control over the experience. This system keeps users informed of key tasks while staying responsive to changes and maintaining a smooth, non-intrusive workflow.

User Story #4: As a user, I would like to set an estimated time and an actual time for each of my tasks, and see the difference between the two.

1	Add an "Estimated Time" field to the task model and task creation/edit UI.	2.5 Hrs	Yana
2	Use existing time log data to calculate total "Actual Time" per task.	1 Hr	Yana

3	Display estimated vs. actual time and the variance in task details view.	2 Hr	Yana

4	Color-code the time difference: green if under, red if over, gray if equal.	1.5 Hrs	Yana

5	Add optional filters/sorting: tasks over time, under time, or on time.	2 Hrs	Yana

Completed:
This user story enhances task tracking by allowing users to set an estimated time for each task and automatically recording actual time using existing time log data. When viewing a task, users can now see the estimated time, actual time, and the variance between the two. To improve clarity, the system color-codes the time difference: green for tasks completed under the estimate, red for those exceeding it, and gray for tasks completed exactly on time. Additionally, users can filter and sort tasks based on whether they were completed over, under, or exactly within the estimated time, making it easier to identify patterns in time management and adjust future planning accordingly.

User Story #5: As a user, I would like to compare my productivity across different weeks/months to track my long-term progress.

1	Modify the database schema to include total logged time per user for weekly tracking.	1 Hr	Yana

2	Implement logic to aggregate and compare logged time over weeks.	2 Hrs	Yana

3	Design a UI component for the Productivity Tracker section, displaying weekly comparisons.	2 Hrs	Yana

4	Show weekly trends in productivity, highlighting improvements or decreases in logged time	2 Hrs	Yana

5	Allow users to click and view previous weekly productivity trends in the Productivity Tracker section.	2 Hrs	Yana

Completed:
This user story introduces a Productivity Tracker that enables users to monitor their logged time across different weeks and observe long-term trends. The database schema was updated to track total logged time per user weekly, and logic was implemented to aggregate and compare these values over time. A new UI component was created for the Productivity Tracker section, visually displaying week-to-week comparisons and highlighting increases or decreases in productivity. Users can also click to view historical productivity data, making it easier to identify patterns, set goals, and stay motivated by tracking their overall progress over weeks and months.

User Story #6: As a user, I would like to be given gentle reminders through pop-ups or alerts suggesting I take breaks after long periods of continuous work.

1	Implement logic to track continuous work time based on logged tasks.	3 Hrs	Yana
2	Set a predefined threshold for continuous work (e.g., 2 hours) and enable logic checking.	1 Hr	Yana
3	Design and implement a pop-up or alert to notify users to take breaks after reaching the continuous work threshold.	2 Hrs	Yana
4	Implement UI functionality to accept or snooze break reminders.	2 Hrs	Yana
5	Create a system to track whether the break reminder has been accepted or snoozed, and show the reminder accordingly.	2 Hrs	Yana

Completed:
This user story focuses on promoting healthier work habits by providing users with gentle break reminders after long periods of continuous work. Logic was implemented to track uninterrupted work time based on task logging, with a predefined threshold of two hours. When the threshold is reached, a pop-up or alert notifies users to take a break. The UI allows users to either accept the reminder or snooze it for later. A system was also created to track the user's response (acceptance or snooze) and manage the behavior of subsequent reminders accordingly, ensuring that alerts are timely, respectful, and supportive of user productivity and well-being.

User Story #7: As a team member, I would like to be able to add tasks.

1	Create a simpler version of the main task page unique to each team	5 hrs	Vihaan
2	Have all members be able to add tasks	2 hrs	Vihaan
3	Tasks should have a section for assigning	1 hr	Vihaan
4	Adding tasks should require a name and date but the rest optional	1 hr	Vihaan
5	Add option to delete tasks	1 hr	Vihaan

Completed:
On each team page, they have a task table where members can input a name and deadline to add a new task. Each task has those columns as well as some more for assigned members and options. When adding tasks, it requires you to input both the name and deadline, or it will display a warning message and not go through. Additionally, the team owner can delete any task while members cannot.

User Story #8: As a team member, I would like to be able to claim tasks.

1	Make each task created contain a button for claiming	1 hr	Vihaan
2	Users can add themselves to the assigned section and it will show for everyone	2 hr	Vihaan
3	Users can remove themselves	1 hr	Vihaan
4	Members cannot add or remove others	1 hr	Vihaan

Completed:
As a non-owner member, each task will have a claim button when created. When it is clicked, the assignee will display as that user. When you are assigned to a task the button will switch from claim to unclaim, so you can click that to remove yourself. If someone else claims a task, you will not see either a claim or unclaim option. Only that user or the owner can remove them.

User Story #9: As a team leader, I would like to be able to assign tasks.

1	Add a dropdown or button for each task which contains all users in the team, only visible to owner	2 hr	Vihaan
2	Selecting a user should save them as the assignee	1 hr	Vihaan
3	The team leader can remove any assignee	1 hr	Vihaan

Completed:
As the owner, a button called Assign is used instead of claim. When clicked, all members in the team will be listed underneath with a Choose button next to them. Clicking one will set them as the assignee for that task. For any task that is assigned, you will instead see the Unassign option. The owner can unassign anyone, and it will remove them. After unassigning, they can then assign anyone again.

User Story #10: As a team member, I would like to mark team tasks as completed.

1	All members can mark a task as completed with a button	1 hr	Vihaan

2	It will move the task under a “completed” section	1 hr	Vihaan

3	The completed section can be hidden or expanded	1 hr	Vihaan

4	If the task had an assignee, it will say “completed by: [their name]”	2 hr	Vihaan

5	Can undo completion to bring it back to the normal table	2 hr	Vihaan

Completed:
Underneath the active tasks table, there is a “Completed” section with an expand/hide button next to it. Clicking this will show the completed task table, and clicking it again will hide it. Back in the active ask table, each task has a check mark button next to them so you can mark them as complete. When clicked, that task will disappear and appear in the completed task section. If the task had an assignee, their name will appear under “Completed by”, while if nobody was assigned, it will say whoever clicked the complete button. In order to move the task back to the active task table, there is an undo button. When clicked, the task will disappear from the completed section and show back up in the active task table, keeping the name and date but becoming unassigned.

User Story #11: As a team member, I would like to leave comments on tasks so that I can communicate updates.

1	In each team’s task page, next to the task, there should be a comment button visible to the user.	1 hr	Jack

2	Comment UI	1 hr	Jack

3	Showing comment on other user’s account	3 hrs	Jack

4	Delete comment button for comment creator	1 hr	Jack

5	Text to display who made the comment	1 hr	Jack

6	Comment buttons on comment UI	1 hr	Jack

Completed:
This user story was creating the feature for members of a team to comment on users’ tasks. This feature allows users to click a comment button that pulls up a UI that the comment can then be typed into. Once on the UI they will be given the option to save whatever is in the comment box. Once the comment is saved, it appears in the comment list of that task, along with who created it. Comments can be made by anyone in the team, but only for the comments that the user created can they see a delete button. The comments also appear across accounts, so one user in a team can see all the same comments as another user of the team.

User Story #12: As a user, I would like to be able to roll over incomplete tasks to the following day, either automatically or manually.

1	Roll over button	1 hr	Jack
2	Automatic roll over button	1 hr	Jack
3	Automatic roll over time	1 hr	Jack
4	Automatic roll over functionality	3 hr	Jack
5	Visual indicator to mark when a deadline has passed	2 hr	Jack
6	Keep track of the amount of times a task has been rolled over	1 hr	Jack
7	Display amount of rollovers next to a task	1 hr	Jack

Completed:
This feature detects when a task has gone over the deadline. It shows tasks that have gone over by changing the background color to a distinct red that only overdue tasks have. Tasks that are overdue also have a rollover button appear in the rollover column, if that button is pressed the deadline will be "rolled over" to the day after the current day and the roll over counter will be incremented. Also, within the rollover column there is an auto rollover checkmark. When that checkmark is clicked, a time input appears beneath it that the user can input their preferred time into. When that time has passed, any tasks that need to be rolled over will be rolled over and their counter will be incremented as well.

User Story #13: As a user, I would like to be able to roll over incomplete tasks to the following day, either automatically or manually.

1	A change display name button in each team page	1 hr	Jack
2	Display name UI, buttons, and functionality	2 hrs	Jack
3	Names don't affect other names	1 hr	Jack
4	Names appear to other users in the team as the new name	3 hrs	Jack
5	Reset button and functionality	2 hr	Jack

Completed:
In the teams section, next to the user's name there is a change display name button. When this button is clicked a menu with an input box, a save button, and a reset button appears. If the user types something into the box and then clicks the save button, whatever they typed into the box will now show up as their display name. This display name is only present within the team and can be seen by any user that is also in the same team. It will change any instance of their real username to the new display name within the team, where other members will also see this new display name. A user can have different display names for different teams and they will save across sessions. Also, when the user is in the change display name menu, if they click the reset button it will change their name back to their real username.

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

Completed:
This user story dealt with creating a "trash bin" to hold the latest 10 tasks and also have a timer to count down a timer before the undo button goes away. I was able to complete the trash bin feature so that the latest 10 tasks that are deleted go there before they are fully deleted. I also had it so that the oldest tasks in the trash bin are deleted, not the newest ones.

Sprint 2, 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

9	Add time of day UI	1 Hr	Jack

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

Completed:
The main archival page is still working. Each task still has a checkmark for the user to mark the task as complete, along with the visual background change when the user does so. The archive button still works as described and puts the task that is archived into the archive page. The tasks that are in the archive page still have the option to be "unarchived" and put back into the main task page as completed. The auto-archival inputs are present within the archive page.

Sprint 2, 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

4	Add a delete notes button in the notes UI.	1 Hr	Jack

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

Completed:
The basic formatting options were implemented, but in a somewhat inconvenient way. You are able to type the text in the bottom area and then highlight and format it in the top area. This allows the user to format their notes in a high variety of ways. The save button does not have functionality so the user is not able to save what they write for their tasks across sessions.

Sprint 2, 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
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

Completed:
This user story dealt with creating a template UI that users can create task templates in. Within the task template UI the user is presented with the option to create a template. When they press the button, a menu with the options to input basic task options appears. The user can then input the options they want for that template and can either press save or cancel. Once saved, that template will appear in the templates UI. All created templates have 3 options under them, create template, edit template, and delete template. The create template button puts a task with the prescribed settings of the template into the task page, the edit button allows the user to change the template's options, and the delete button deletes that template from the list. The user is also able to add as many templates as they want with the “add template” button.

What did not go well?

For the most part, the sprint 3 tasks went well and delivered the features that were expected, but the full thing cannot be said for the unfinished tasks of sprint 2. While there was good progress on the unfinished sprint 2 tasks, many of them were still left incomplete. In a lot of the stories, they were mostly complete, but with some features that were left out. For example, in the archive user story, the auto archive has yet to be implemented. This pattern rang true for the other unfinished sprint 2 stories, with 1-2 of the features being unfinished despite the main functionality being complete. Another large issue with the sprint 2 stories was the saving to the database of almost all of the unfinished tasks that had yet to be implemented. None of the unfinished sprint 2 stories has their data saved to the database.

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 undo functionality is still not fully implemented as it was listed in the acceptance criteria. While there is an undo button that will bring it back to its normal state, there is no timer that counts down until the task is deleted.

Sprint 2, 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

9	Add time of day UI	1 Hr	Jack

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

Not completed:
While the auto-archival inputs and buttons exist, none of them do what was prescribed within the acceptance criteria. Also, none of the archived or completed tasks save to the database and across sessions.

Sprint 2, 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.

4	Add a delete notes button in the notes UI.	1 Hr	Jack

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

Not completed:
There is no delete button for the notes that exist and no functionality either. Along with this, the notes do not save in the database or persist across sessions.

Sprint 2, User Story #6: As a user, I would like to set default task templates so that I can quickly create repetitive tasks.

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:
While the “most recent” and “favorite” buttons exist, they do not do anything when clicked. The most recent button is supposed to create a task from the most recently used template and the favorite feature is supposed to keep favorited tasks at the top of the templates, but neither do their prescribed functionality. Also, none of the templates or created tasks from templates persist in the database across sessions.

How should we improve?

One key area for improvement is our ability to more accurately predict the estimated time required for our tasks. This was a noticeable issue in both directions–some tasks were completed much faster than expected, while others took significantly longer than we had planned. As a result, certain tasks were left unfinished, and we had to slightly adjust the scope of our work to accommodate these miscalculations. While the benefit of overestimating time was that it gave us the opportunity to assist others on their tasks, it also revealed a lack of precision in our planning. To make our process smoother and more efficient in future iterations, we should focus on breaking tasks down further and using past experience to create more realistic time estimates. This will help us allocate our time more effectively and finish all our tasks in time.

Another important improvement we can make is to communicate more frequently and consistently throughout the process. During this project, we held relatively few meetings, which resulted in a lack of awareness regarding what other team members were working on and who might have needed assistance. With more regular check-ins, we could have even solved the time estimation issues mentioned earlier by reallocating support where it was most needed. More frequent communication would not only help us stay on track but also foster greater collaboration and adaptability. Improving these two areas–time estimation and communication–would have enhanced both the workflow and the overall quality of our team project.