Neha Bangari, Vrishank Bangari, Jake Frohlich, Ethan Werner
Intermediate Software Design

# Project Milestone 2

## Process Deliverable I

**Agile Process Model**

| **Name** | Jake | Ethan | Neha | Vrishank |
|---|---|---|---|---|
| **What went well?** | Everyone communicates when they will work on quickly and effectively. | When I had trouble completing work, the rest of the team stepped up to help out. | Everyone in the team was responsive, motivated, and quick to help out when needed. | Communication was very open and people were quick to respond to questions. |
| **What didn't go well?** | Understanding the actual workload between each part of the assignment was difficult to measure. | The part I had chosen to do was a lot more work than I had expected. | I think it would have been more helpful if we split up the work in a more even way, which we didn't initially realize. | The delegation of work was done improperly, leading to problems later on |
| **What will we change next time?** | We will coordinate the work distribution more effectively. | We will try to better understand the tasks we must complete before distributing the work among the team. | Have more thorough discussions about the tasks as a group so that work is better distributed. | Go thoroughly through each part so work is split up better |

**Prioritized Tasks for PM3:**

1. Begin discussing plans for user interface design - create wireframes to discuss and share amongst team members and narrow down design plans.
2. Discuss storage method and input collection of task data.
3. Figure out the logic of task assignment and scheduling.

## Requirements A

1.

Usability : The text on the scheduler interface should be clearly readable from a distance of 1 meter to ensure accessibility and ease of use across various screen sizes.

Reliability : The scheduler should maintain a mean time between failures (MTBF) of 30 hours under normal operation, particularly during high-demand task reordering processes.

Performance: The scheduler's task prioritization and reordering should complete in less than 1 second for 90% of task input requests to ensure efficient processing and minimal wait time for users.
Supportability : The system should support frequent updates and customization in task ordering logic, allowing users to configure new priority rules easily as their task management needs evolve.
Implementation Constraints: The scheduler must be developed using Python and JavaScript for compatibility with the chosen platforms and to meet cross-platform requirements.

2.
Task Editing: The system must allow users to edit task details (such as name, difficulty level, or priority) at any time, with changes reflected immediately in the task order.
Deadline Management: Users should be able to assign deadlines to tasks, and the scheduler must reorder tasks to prioritize those with upcoming deadlines if the user enables deadline-based prioritization.
Task Categorization: The system should enable users to categorize tasks into different groups (e.g., "Work," "Personal") and view or reorder tasks by category for more organized scheduling.
History Tracking: The scheduler must store a history of completed tasks for at least 30 days, allowing users to view their productivity trends and completed task summaries.
Time Estimation: The system should allow users to input an estimated time to complete each task and display a cumulative time estimate for all tasks, helping users manage their time effectively based on the total workload.

3.
**<u>Use Case 1: Plan Weekly Work Distribution</u>**

**1 Preconditions**

- User must have a list of tasks for their week and have an estimated range of the time needed to complete each.

**2 Main Flow**

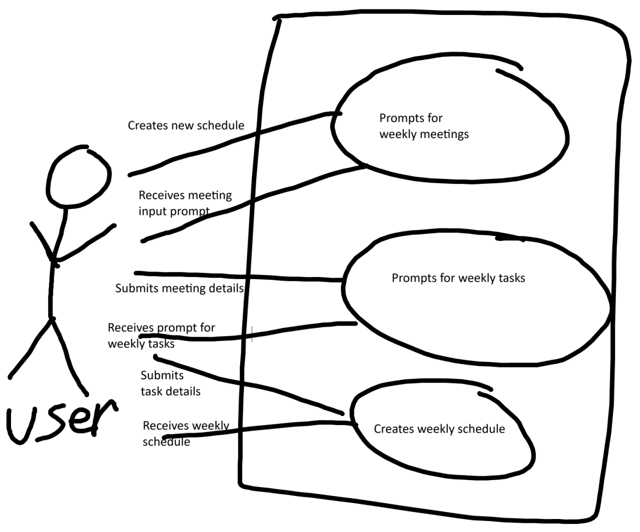
- User launches the system and selects the option to plan their weekly work distribution. The system will prompt the user to enter their meetings and must input each meeting they have scheduled for the week and submits their meeting schedule [S1]. The system will prompt the user to input their weekly tasks and submits each task and its details [S2]. User submits their tasks. System will create and display an optimized schedule for their week.

**3 Subflows**

- [S1] User is prompted to enter each meeting scheduled for the week. User must enter the title, day, time, and length of each meeting. User may also enter a description for any meeting. The system collects this information when the user submits.
- [S2] User is prompted to enter each task that must be completed for the week. User must enter the title and estimated length of each task. User may also enter a deadline or description for any task. User submits and the system collects the information.

**4 Postconditions**

- The optimized work schedule is successfully created and displayed.



## Use Case 2: Change an Existing Task

**1 Preconditions**

- User must have an existing schedule for the current week in the system, with at least one task.

**2 Main Flow**

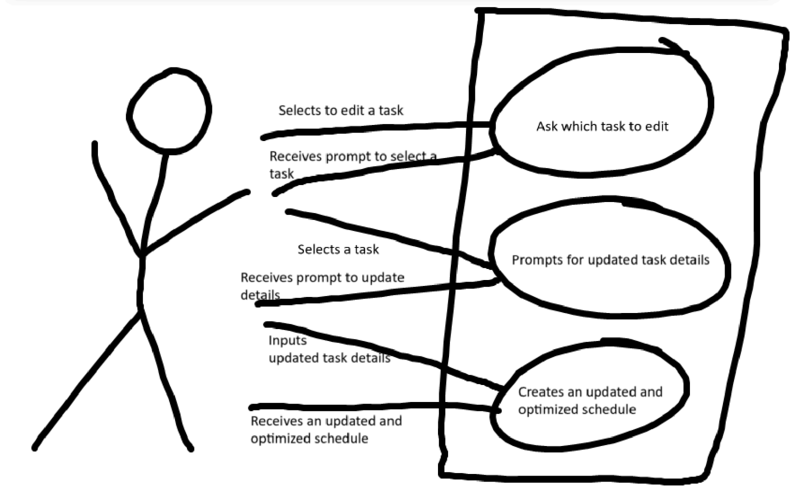
- User launches the system and selects the option to modify an existing task. The system will prompt the user to select the specific task they wish to change. User must update the task's title and length [S1]. User submits the modified task information. The system will recalculate the weekly schedule and display the updated, optimized work distribution.

**3 Subflows**

- [S1] User selects the task's title and enters the new title. User selects the length of the task and enters their new length estimate. The system will collect the updated information once the user submits.

**4 Postconditions**

- The weekly schedule is successfully recalculated, redistributed, and displayed with the modified task.


Selects to edit a task
Receives prompt to select a task
Selects a task
Receives prompt to update details
Inputs updated task details
Receives an updated and optimized schedule
Ask which task to edit
Prompts for updated task details
Creates an updated and optimized schedule

**Use Case 3: Add a Task**

**1 Preconditions**

- User must have an existing schedule for the current week in the system.

**2 Main Flow**

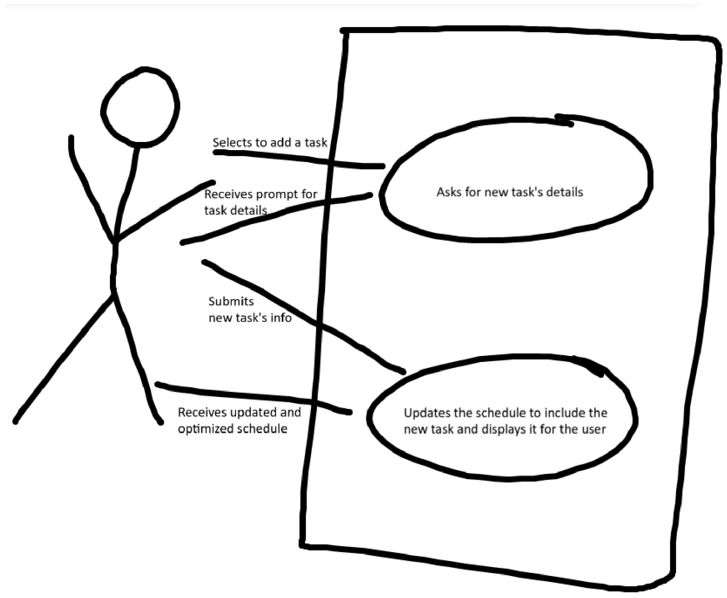
- User will launch the system and selects the option to add a new task to their schedule. The system will prompt the user to input the new task's title and estimated length. User may input the task's deadline or description [S1]. User will submit the new task's information. The system will then recalculate the weekly schedule to include the new task and display the updated and optimized schedule.

**3 Subflows**

- [S1] The system will prompt the user to enter the details of the new task. User will input the title and estimated length of the new task. The user may also enter a deadline or description for the task. The system will collect the information once the user submits.

**4 Postconditions**

- The weekly schedule is successfully updated, optimized, and displayed with the added task.


Selects to add a task
Receives prompt for task details
Asks for new task's details
Submits new task's info
Receives updated and optimized schedule
Updates the schedule to include the new task and displays it for the user

<u>**Use Case 4: Manually Schedule a Task**</u>

**1 Preconditions**

- User must have an existing schedule for the current week in the system, with at least one task.

**2 Main Flow**

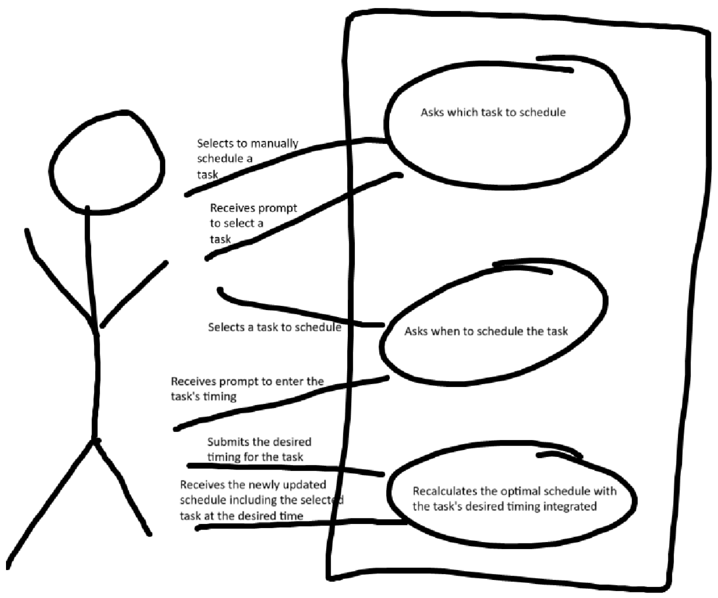
- User will launch the system and select the option to manually schedule a task from their existing schedule. The system will prompt the user to select the specific task they wish to plan. User will select the task they wish to manually schedule and will enter their desired start or finish time/day [S1]. User will submit the updated timing information. The system will update the weekly schedule to include the new task timing and display the updated and optimized schedule.

**3 Subflows**

- [S1] User will select whether to specify the start time or finish time for the task. User inputs the time corresponding to their decision. The system collects the information once the user submits.

**4 Postconditions**

- The weekly schedule is successfully updated and displayed with the manually scheduled task placed at exactly the desired time.


Asks which task to schedule
Selects to manually schedule a task
Receives prompt to select a task
Selects a task to schedule
Asks when to schedule the task
Receives prompt to enter the task's timing
Submits the desired timing for the task
Receives the newly updated schedule including the selected task at the desired time
Recalculates the optimal schedule with the task's desired timing integrated

**<u>Use Case 5: Reassign tasks</u>**

**1 Preconditions**

- Users must have an existing schedule for the current week in the system, with at least one task per user

**2 Main Flow**

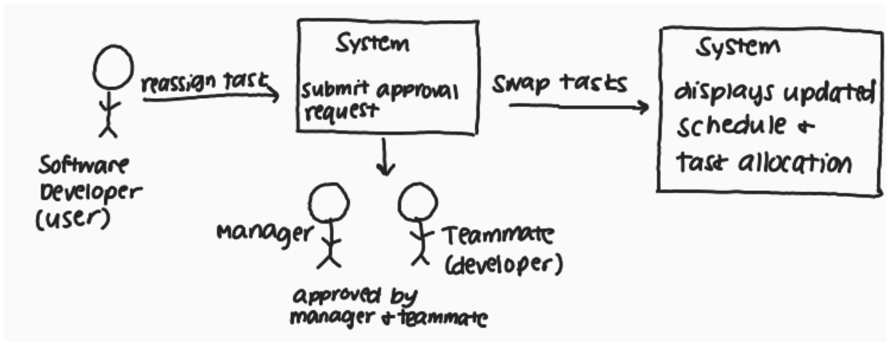
- User will launch the system and view tasks in the current schedule. Pending approval, the user will be able to reassign themselves to another user's task and will assign the other user to their old task [S1]. The user will submit these changes. The system will update the weekly schedule to include the updated task allocation and display these changes to the entire team.

**3 Subflows**

- [S1] The system will prompt the user to request an approval for this change to be approved by the project lead/manager prior to making the change. The user must also include the other user, who will also get notified prior to the change being made.

**4 Postconditions**

- The weekly schedule is successfully updated and displayed with the new allocation of the existing tasks.


Software Developer (user)
reassign task
System
submit approval request
swap tasks
System
displays updated schedule + task allocation
manager
Teammate (developer)
approved by manager + teammate

## Requirements Specification

### User Stories

1. As a *software engineer*, I want the Task Scheduler to automatically prioritize my tasks by deadlines so that I can spend my time on more important tasks.

   Acceptance Criteria
   *Given* the software engineer has tasks with varying priorities and deadlines
   *When* weekly tasks are inputted to the Task Scheduler
   *And* the scheduler has information about priority levels, completion times, and deadlines
   *Then* the Task Scheduler will automatically sort tasks by priority and deadlines, with the ability to update in real-time.

   Subtasks & Function Points
   *Function points are allocated on a scale of 1-10, 10 being the most challenging*
   Data input for tasks - 4
   - This is a relatively light task as the data would need to be taken in without much manipulation.

   Task data processing/prioritization & storage - 9
   - This is a higher complexity task due to it being the main functionality of the system. We assigned it a higher number of function points due to the sorting logic and real-time processing that this task requires.

   Real-time updating - 7
   - This subtask is also high complexity as it depends on continuous calculations and response to user updates.

   Display task schedule - 8
   - This is another high complexity subtask as we must take into account several factors such as aesthetics, ease of use, and accessibility.

2. As a *project manager*, I want the Task Scheduler to effectively delegate tasks to my team so that my team can maximize productivity.

   Acceptance Criteria

*Given* the project manager has visibility into team members' capabilities and work loads
*When* tasks are added to the Task Scheduler
*And* the system has information about each developer's availability as well as tasks
*Then* the Task Scheduler will order tasks such that work is optimized across the team

Subtasks & Function Points
Task Assignment Logic - 9

- This subtask will take the most amount of time as we will have to utilize an efficient algorithm that will take into consideration multiple factors to prioritize and schedule tasks.

Real-Time Updates - 8

- This subtask is not as time-intensive as the other one, but still will require time to display the schedule in a way that addresses user concerns.

3. As a *project manager,* I want the Task Scheduler to help me reassign tasks if a team member falls behind so that project timelines are maintained.

Acceptance Criteria
*Given* the Task Scheduler has access to real-time task progress and completion status
*When* a team member's tasks are delayed with an effect on overall deadlines
*And* the project manager is notified of these concerns
*Then* the Task Scheduler will support the project manager in reassigning tasks to available team members

Subtasks & Function Points
Task Assignment and Reassignment - 9

- This subtask will take the majority of time to implement, as it relies on saving user input and also must support edits to be made by the user.

Real-Time Schedule Updates - 7

- This is also a high-complexity task as the user interface needs to update along with user changes.

4. As a *client*, I want the Task Scheduler to ensure my project deadlines are met so that I can rely on timely project deliverables from the development team.

Acceptance Criteria
*Given* the client has defined project requirements and deadlines
*When* the Task Scheduler allocates tasks for the development team
*And* provides an expected schedule and timeline of completion
*Then* the Task Scheduler will ensure tasks are optimally delegated to address the client's needs.

Subtasks & Function Points

Task Prioritization and Scheduling - 10

- This user story relies on the main functionality of the Task Scheduler, which will require an algorithm to efficiently organize and schedule tasks.

Task and Project Deadline Notifications - 9

- As clients rely on the outcome of the system, a notification feature is also important as it will keep the client informed throughout development.