Requirements

Requirements Workshop

1. **Provide an example of five hypothetical non-functional requirements for this system. Be sure to include the specific type of requirement discussed in class, with each requirement coming from a unique category.**
    - Performance requirement
        - The system should respond to a user's inputs within a half a second as to make the system feel smooth and clean when using.
    - Usability requirement
        - The system should have tools in place for users with a disability – e.g. a mode for users with bad vision that enlarges text.
    - Reliability requirement
        - The system should not lose or corrupt any data/information if it were to crash.
    - Supportability requirement
        - The system should allow and make it simple for users to customize the system to fit their needs/wants.
    - Implementation requirement
        - The system should be developed using JavaScript, HTML and Java.

2. **Provide an example of five hypothetical functional requirements for this system.**
    - Task Management Feature
        - Users should be able to create, edit, and delete tasks – with the ability to set task description, deliverable(s), estimated duration, difficulty, priority and deadline.
    - Integration with Version Control
        - The system should integrate with popular version control systems (e.g., Git) to track code changes related to each task.
    - Time tracking feature
        - The system should keep track of how much time users spend working during certain weeks and on certain tasks, to evaluate the user's productivity and how it can be affected by various factors.
    - Blocking feature

        - The system should offer users the ability to block certain applications, such as social media platforms, to stop them from getting distracted during a ‘work session’
    - Productivity recommendations
        - The system should offer productivity aids (e.g., suggestions of when to take a break, focus on other tasks, seek help…), when needed.

3. **Think of a specific task required to complete each of the functional requirements and non-functional requirements mentioned above. Estimate the amount of effort needed to complete this task using function points (i.e., using the values here). Briefly explain your answer.**
    - Performance Requirement task:
        - Optimize the frontend and backend code to achieve minimal delay in user-system interactions and ensure a smooth and responsive user experience. (20 points)
    - Usability Requirement task:
        - Implement text size adjustments and other accessibility features to make the system more user-friendly and accessible to a wider audience. (8 points [subject to change depending on the number of accessibility features implemented])
    - Reliability Requirement task:
        - Develop a robust backup system in the event of a system crash (40 points)
    - Supportability Requirement task:
        - Create customizable options to allow users to modify the system’s interface. (40 points [subject to change depending on the number of customizable options we implement])
    - Implementation Requirement task:
        - Implement the code in the defined languages (10 points)
    - Task Management Feature task:
        - Develop the Create, Read, Update, Delete (i.e., CRUD) operations for tasks (20 points)
    - Version Control task:
        - Integrate the system with the GitHub API (30 points)
    - Time tracking task:
        - Implement a time tracking system by week and for each task (10 points)
    - Blocking feature task:
        - Create a tool that blocks certain applications during ‘work times’ (20 points)
    - Productivity feature task:

    - Develop tools to look at users' performance and recommend suggestions to stay productive (50 points)

4. **Write three user stories from the perspective of at least two different actors. Provide the acceptance criteria for these stories.**
    - Developer Perspective (First User Story)
        - As a developer, I want to be able to plan my tasks for the week, so that I can be more focused and productive during work.
        - *Acceptance criteria*:
            - GIVEN developer is logged into the app,
            - WHEN developer enters in tasks for the week,
            - THEN the app orders the tasks and generates a 'Tasks Plan' for the week.
    - Project Manager Perspective (Second User Story)
        - As a project manager, I want to be able to see the tasks my employees are currently working on, so that I am better aware of the progress we're making.
        - *Acceptance criteria*:
            - GIVEN project manager is logged into the app,
            - WHEN project manager selects an employee's recent 'work session',
            - THEN the app shows them the tasks that were worked on and the duration for each task in that session.

5. **Provide two examples of risk that could potentially impact this project. Explain how you would mitigate these risks if you were implementing your project as a software system.**
    - Security (First Risk)
        - Given the collaborative nature of the application, there is a potential risk associated with user data handling, including data breaches and unauthorized access.
        - *Mitigation*:
            - Data encryption to prevent unauthorized access,
            - Access control to regulate access to different projects, ensuring. that only authorized members can view and modify project data.
    - Data Loss (Second Risk)
        - Another risk to consider is the potential loss of data due to issues like not saving work, application crashes, or encountering bugs.
        - *Mitigation*:
            - Feature an automatic save system that periodically saves user data,

        - Conduct weekly automated performance and reliability testing on the application to identify and address potential issues that could lead to data loss.

6. **Describe which process your team would use for requirements elicitation from clients or customers, and explain why.**

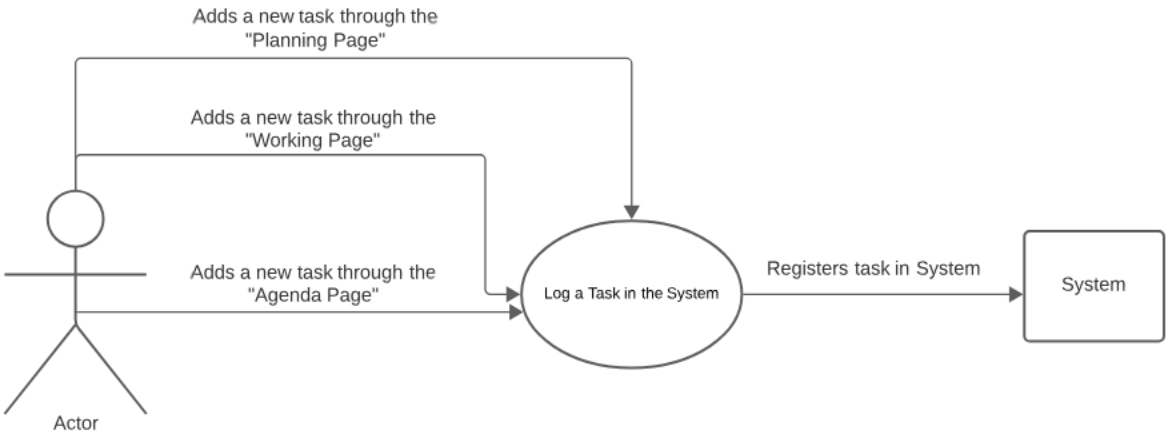
During the development process, we will employ Agile methodologies to elicit requirements from our clients. Since agile promotes collaboration and flexibility, we believe we can remain closely aligned with any evolving requirements if we use agile processes.
We will start the process by establishing a strong foundation with clients through clear communication during bi-weekly meetings – where ideas, specifications and feedback can be discussed and defined. If things line up, we might even rope in some focus groups to mix things up and bring in fresh perspectives to redefine our project requirements. In addition, we'll use short sprints to efficiently tackle and quickly gather feedback on smaller feature sets. This iterative approach will enable us to maintain regular, transparent communication with our clients and eliminate potential misunderstandings, allowing us to make any necessary changes easily and as soon as possible.

Requirements Analysis

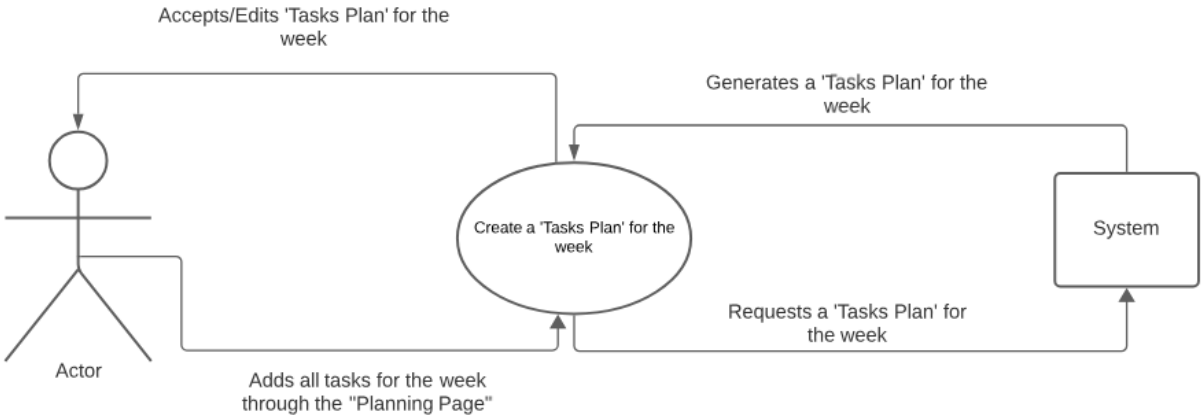
A few use cases for our project are:

1. Logging-in a Task:
    - *Precondition*: User must be logged into the app and have at least one task to enter.
    - *Main Flow*:
        - User navigates to the agenda page [S1].
        - User clicks on "Add a Task" [S2].
        - User inputs information (i.e., type of task, deliverables, estimated duration, difficulty, deadline…) about the task [S3].
    - *Sub Flow*:
        - User navigates to the working page [S1].
        - User clicks on "Start Working Session" [S2].
        - User selects "Add a New Task" [S3].
        - User is prompted to enter information (i.e., type of task, deliverables, estimated duration, difficulty, deadline…) about the task they're currently working on [S4].
    - *Alternative Flow*:
        - User navigates to the planning page [S1].
        - User clicks on "Plan Weekly Tasks" [S2].

        - User inputs information (i.e., type of task, deliverables, estimated duration, difficulty, deadline…) about the task [S3].
    - *Postcondition*: User has logged in a task.

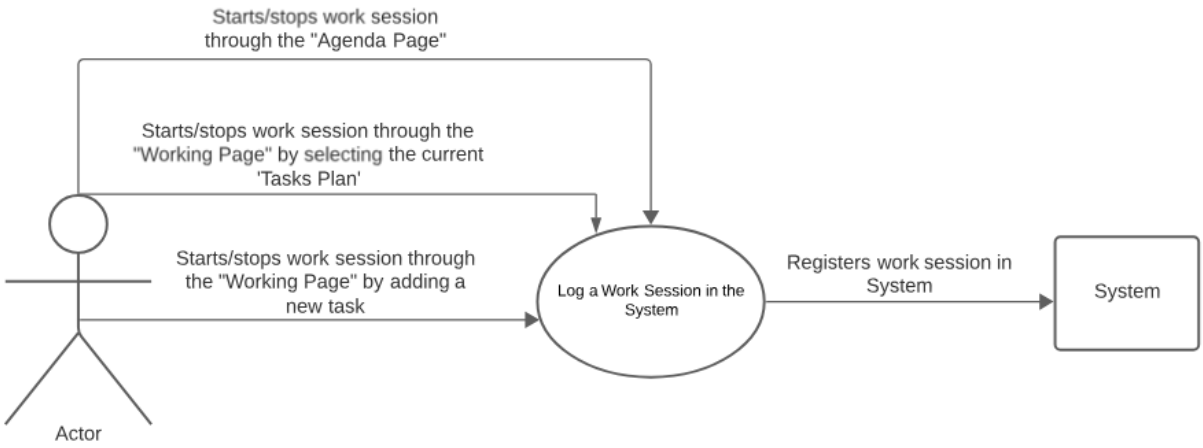


2. <u>Generating a 'Tasks Plan' for the Week:</u>
    - *Precondition:* User must be logged into the app and have a number of tasks to enter.
    - *Main Flow*:
        - User navigates to the planning page [S1].
        - User clicks on "Plan Weekly Tasks" [S2].
        - User inputs information (i.e., type of task, deliverables, estimated duration, difficulty, deadline…) about each task [S3].
        - App orders tasks in a more productive way (i.e., based on deadline, difficulty, priority, and estimated duration) and creates a 'tasks plan' for the week [S4].
        - User accepts the generated plan [S5].
    - *Sub Flow*:
        - User navigates to the planning page [S1].
        - User clicks on "Plan Weekly Tasks" [S2].
        - User inputs information (i.e., type of task, deliverables, estimated duration, difficulty, deadline…) about each task [S3].
        - App orders tasks in a more productive way (i.e., based on deadline, difficulty and estimated duration) and creates a 'tasks plan' for the week [S4].
        - User rejects the generated plan and re-orders the tasks themselves [S5].
    - *Postcondition*: User has a 'Tasks Plan' for the week.



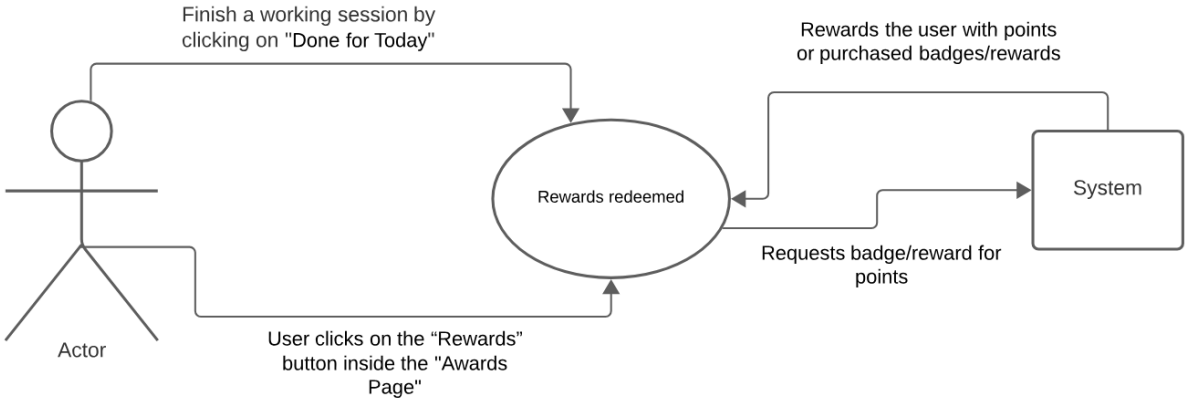
3. <u>Logging-in a work session:</u>

- *Precondition*: User must be logged into the app and be ready to start working.
- *Main Flow*:
  - User navigates to the agenda page and selects the current date [S1].
  - User clicks on “Work on Today’s Tasks” [S2].
  - App starts a timer and displays the task to work on from the generated ‘Tasks Plan’ of the week [S3].
  - User clicks on “Next Task” to continue on to the following task or “Done for Today” to stop the timer and end the work session [S3].
- *Sub Flow*:
  - User navigates to the working page [S1].
  - User clicks on “Start Working Session” [S2].
  - User selects “Work on Current ‘Tasks Plan’” [S3].
  - App starts a timer and displays the task to work on from ‘Tasks Plan’ [S3].
  - User clicks on “Next Task” to continue on to the following task or “Done for Today” to stop the timer and end the work session [S3].
- *Alternative Flow*:
  - User navigates to the working page [S1].
  - User clicks on “Start Working Session” [S2].
  - User selects “Add a New Task” [S3].
  - App starts a timer and displays the added task to work on [S3].
  - User clicks on “Done for Today” to stop the timer and end the work session [S3].
- *Postcondition*: User has logged in a work session.



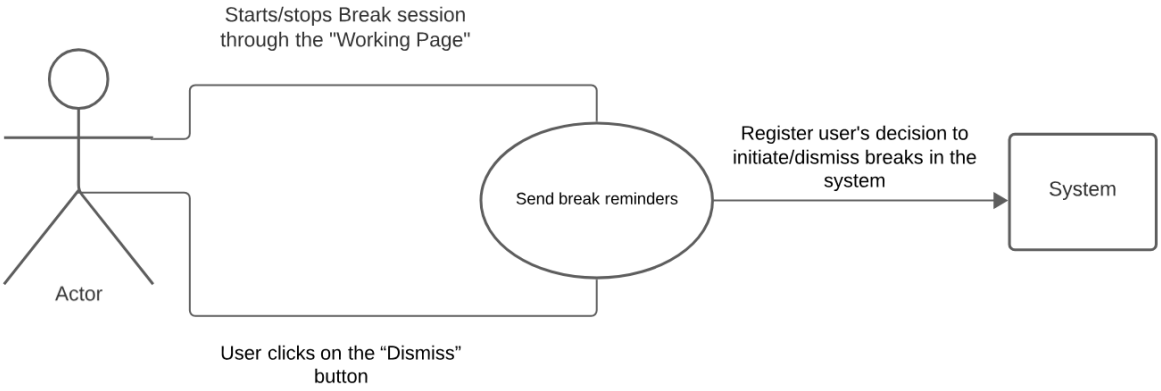
4. Redeeming Rewards:
    - *Precondition:* User is logged into the app.
    - *Main Flow:*
        - User is in a work session [S1].
        - User completes the task and clicks on “Done for Today” to stop session [S2].
        - App rewards the user with points for the tasks they completed during the session [S3].
        - User is provided a page to purchase badges/rewards with the points they received [S4].
        - User selects and purchases the badges/rewards they want [S5].
        - App stores the purchases in the user’s account. [S6]
    - *Sub Flow:*
        - User navigates to the awards page [S1].
        - User clicks on the “Rewards” button [S2].
        - User selects and purchases the badges/rewards they want [S3].
        - App stores the purchases in the user’s account. [S4]

    Postcondition: User has redeemed rewards.



5. Receiving Break Reminders:

- *Precondition:* User is logged into the app and has been working non-stop on a single task for a long period of time (i.e., longer than 2 hours)
- *Main Flow:*
  - App sends a notification to remind the user to take a break [S1].
  - User clicks on “OK” and the app starts a timer for the break [S2].
  - App offers suggested activities (e.g., walking outside, stretching…) [S3].
  - User clicks on “Finish Break” and goes back to their work session [S4].
- *Sub Flow:*
  - User clicks on “Start Break” during their work session [S1]
  - User manually adds activities to do [S2].
  - User clicks on “Finish Break” when they’re done and goes back to their work session [S3].
- *Alternative Flow:*
  - App sends a notification to remind the user to take a break [S1].
  - User clicks on the “Dismiss” button [S2].
  - User’s time on the task continues being tracked [S3].
- *Postcondition:* User has received/started a break.



Process Deliverable

Scrum Meeting notes:

- Andrew:
  - Yesterday, I worked on requirement gathering for PM2.
  - I've documented most of the requirements, but some details need clarification, which I'll address today.
  - No blockers.
- Brandon:
  - I've compiled the list of features to include.
  - I'm refining the list based on some recent research papers. My goal is to finalize it today.
  - No blockers.
- Mali
  - I've been developing the use case scenarios.
  - A few scenarios remain (i.e., about 2), I'll complete those by today
  - For now, I’m just waiting on the other features Brandon is thinking of adding to the project before formulating the remaining use cases.
- Daniel
  - I've created models for most of the use cases.
  - I'm left with two, and I plan to complete them by the end of today.
  - I’m just waiting on Mali to complete her list of use cases before I can move forward.