Requirements Workshop questions

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.

1. Performance: The system must be capable of handling concurrent user access from a large software development team (e.g., 100 users) with response times for key actions not exceeding 2 seconds.
2. Security: The system should enforce strong password policies and support two-factor authentication to ensure that sensitive project data remains secure.
3. Usability: The user interface should be intuitive and user-friendly, requiring minimal training for new team members to use effectively.
4. Scalability: The system should be able to scale with the team's growth, allowing for easy addition of new projects, tasks, and users.
5. Reliability: The system should have a minimum uptime of 99.9%, ensuring that it's available when the development team needs it.

Provide an example of five hypothetical functional requirements for this system.

1. Task Management: Users can create, assign, and prioritize tasks. Each task should have fields for title, description, priority, due date, and assigned developer.
2. Project Tracking: Users can create and manage software development projects. Each project should have a name, description, and a list of associated tasks.
3. Notifications: The system will notify users of task assignments and due dates via email or in-app notifications.
4. Time Tracking: Users can log the time spent on each task, and the system will provide reports on the time spent by each developer and on individual tasks.
5. 3rd party calendar integration: The system should be able to export to calendar apps such as google calendar.

Think of a specific task required to complete each of the functional requirements and non-functional requirements mentioned above (10 total). Estimate the amount of effort needed to complete this task using function points (i.e., using the values here). Briefly explain your answer.

Task Creation (Functional Requirement: Task Management)

- Task: Implement the ability for users to create tasks with fields for title, description, priority, due date, and assigned developer.
- Effort Estimation: This task is relatively straightforward and would require around 5 function points. It involves basic data entry and form processing.

Project Creation (Functional Requirement: Project Tracking)

- Task: Develop the functionality to allow users to create and manage software development projects, including fields for project name and description.

- Effort Estimation: Creating a project should be similar in complexity to task creation, so it would also be around 5 function points.

Notification System (Functional Requirement: Notifications)

- Task: Implement a notification system that sends email or in-app notifications to users when tasks are assigned or when task due dates are approaching.
- Effort Estimation: Developing a notification system is more complex and would require approximately 10 function points.

Time Tracking (Functional Requirement: Time Tracking)

- Task: Develop the time tracking feature that allows users to log time spent on each task and provides reports on time spent by developers and individual tasks.
- Effort Estimation: Time tracking is a significant feature and would require approximately 15 function points due to its complexity and reporting capabilities.

3rd Party Calendar Integration (Functional Requirement: 3rd Party Calendar Integration)

- Task: Implement the ability to export tasks to calendar apps such as Google Calendar.
- Effort Estimation: Calendar integration, while not as complex as time tracking, still involves handling third-party APIs and would require around 10 function points.

Concurrent User Handling (Non-Functional Requirement: Performance)

- Task: Optimize the system to handle concurrent user access from a large software development team. This may involve performance tuning and scalability enhancements.
- Effort Estimation: Ensuring the system meets performance requirements is complex and would require approximately 20 function points.

Security Implementation (Non-Functional Requirement: Security)

- Task: Implement strong password policies and multi-factor authentication to enhance system security.
- Effort Estimation: Enhancing security is a critical task and would require approximately 15 function points due to the complexity of security measures.

User Interface Design (Non-Functional Requirement: Usability)

- Task: Design an intuitive and user-friendly user interface that requires minimal training for new team members to use effectively.
- Effort Estimation: User interface design is significant and would require approximately 15 function points.

Scalability Enhancement (Non-Functional Requirement: Scalability)

- Task: Implement features that allow easy addition of new projects, tasks, and users to ensure scalability with team growth.
- Effort Estimation: Ensuring system scalability involves some complexity and would require around 10 function points.

Uptime Maintenance (Non-Functional Requirement: Reliability)

- Task: Implement measures to maintain a minimum uptime of 99.9%, including redundancy and failover capabilities.
- Effort Estimation: Ensuring system reliability is complex and would require approximately 20 function points.

Write three user stories from the perspective of at least two different actors. Provide the acceptance criteria for these stories.

User Story 1 - Claiming Tasks (from a Team Member's Perspective):
As a team member, I want to claim tasks assigned to me so that I can take ownership of them and ensure their completion.

Acceptance Criteria:
- Given that I am a team member with appropriate permissions in the system,
- When I view the list of unclaimed tasks assigned to the project,
- And I select a task from the list to claim it,
- Then the system should mark the task as "claimed" by me and notify relevant parties, such as the project manager, updating the task's status to "claimed."

User Story 2 - Creating a New Project (from a Manager's Perspective):
As a manager, I want to create a new project within the app to initiate and oversee the development of a new software project for the team.

Acceptance Criteria:
- Given that I have managerial permissions in the system,
- When I access the project creation interface,
- And I provide essential project details, including the project name, description, and key objectives,
- Then the system should generate a unique project identifier and allocate resources for the new project.
- I should be able to assign project leads and team members to the project, ensuring that the right individuals are involved from the outset.

User Story 3 - Exporting Task List to Google Calendar (from a Team Member's Perspective):
As a team member, I want to export my task list, including task details and time spent, to my Google Calendar so that I can effectively schedule and manage my work.

Acceptance Criteria:
- Given that I am a team member with appropriate permissions in the system,
- When I access my task list and select tasks that I want to export,
- And I choose a date and time for each task in my Google Calendar,
- Then the system should generate calendar events for the selected tasks, including task titles, descriptions, and the time spent on each task.

- Additionally, the system should provide a link to the Google Calendar events or sync them directly with my Google Calendar.

User Story 4 - Creating Project Reports (from an Administrator's Perspective):
As an administrator, I need to create and export project reports for various stakeholders to track project progress and make data-driven decisions.

Acceptance Criteria:
- Given that I have administrator permissions in the system,
- When I access the project reporting module,
- And I select the project and reporting parameters (e.g., date range, project metrics),
- Then the system should generate a detailed project report containing key project data, task progress, and other relevant information.
- I should be able to export the report in a commonly used format (e.g., PDF, Excel) for sharing with stakeholders.

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.

Technical Risk: Integration Challenges with Google Calendar API
Risk Description: Integrating your task management system with the Google Calendar API could pose technical challenges. This may include difficulties in synchronizing data, handling authentication, or dealing with changes in the Google Calendar API.

Mitigation Strategy:
- **Thorough Testing:** Prior to deployment, conduct extensive testing of the integration to identify and address issues proactively.
- **Backup Solutions:** Develop a backup plan that allows users to manually export task lists or rely on alternative calendar systems if the Google Calendar API integration encounters problems.
- **Continuous Monitoring:** Keep an eye on Google Calendar API updates and changes, and be prepared to adapt your integration accordingly.

Security Risk: Data Breach or Unauthorized Access
Risk Description: There's a potential risk of data breaches or unauthorized access to sensitive project and user data, which could have legal and reputational consequences.

Mitigation Strategy:
- **Security Best Practices:** Implement robust security measures, including data encryption, strong authentication, and access control.
- **Regular Security Audits:** Conduct routine security audits and penetration testing to identify vulnerabilities.

- **User Education:** Educate users on security best practices, such as strong password management and two-factor authentication.
- **Data Backup:** Implement regular data backup procedures to ensure data recovery in the event of a breach.

Describe which process your team would use for requirements elicitation from clients or customers, and explain why.
Our team would use the group elicitation technique. This would mainly involve focus groups. This would allow us to get together software developers who would be using our software and can tell us, in their experience, what works and doesn't work for apps like ours. This will be extra helpful in our circumstance because as software developers they will have an understanding of how the app works and what is realistic to implement.
We will also use surveys Through the data collected through surveys, we can understand the priorities of user needs, and then improve software functions and features.

Use Cases

Use Case 1: Create a New Task

Preconditions:

- User must be logged into the system.
- User has the necessary project access permissions.

Main Flow:

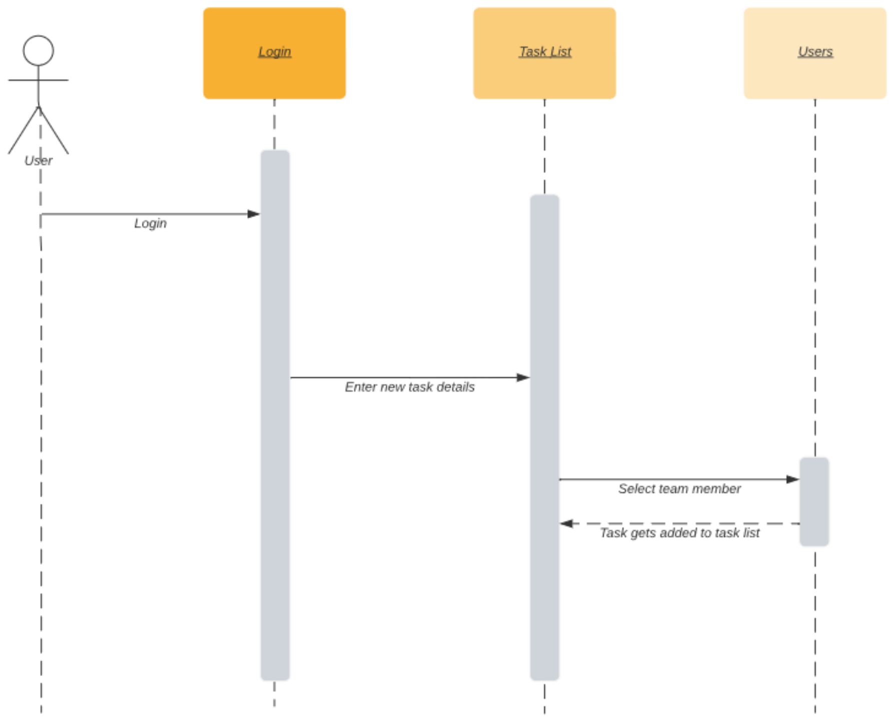
- User initiates the task creation process.
- User provides task details, including title, description, priority, and due date [S1].
- User assigns the task to a team member [S2].
- System validates the input and creates the task [S3].

Subflows:

- [S1] User enters task details through the task creation form.
- [S2] User selects a team member from the list of available team members.
- [S3] System creates the task and adds it to the project's task list.

Alternative Flows:

- [E1] The user cancels the task creation process.



Use Case 2: Assign a Task to a Team Member

Preconditions:

- User must be logged into the system.
- User has the necessary project access permissions.

Main Flow:

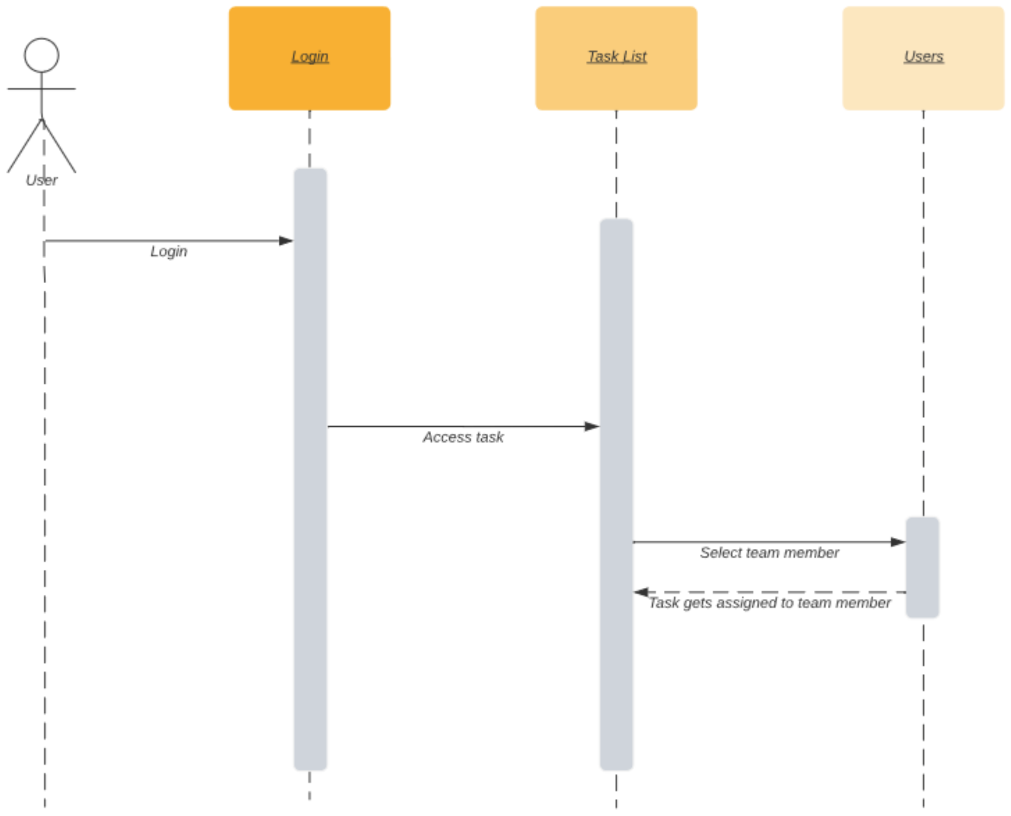
- User selects a task that needs to be assigned to a team member.
- User chooses a team member from the list [S1].
- System validates the assignment and updates the task [S2].

Subflows:

- [S1] User selects a team member from the list of available team members.
- [S2] System updates the task's assignment and notifies the team member.

Alternative Flows:

- [E1] The user cancels the assignment process.



Use Case 3: Track Time Spent on a Task

Preconditions:

- User must be logged into the system.
- User is assigned to a task.

Main Flow:

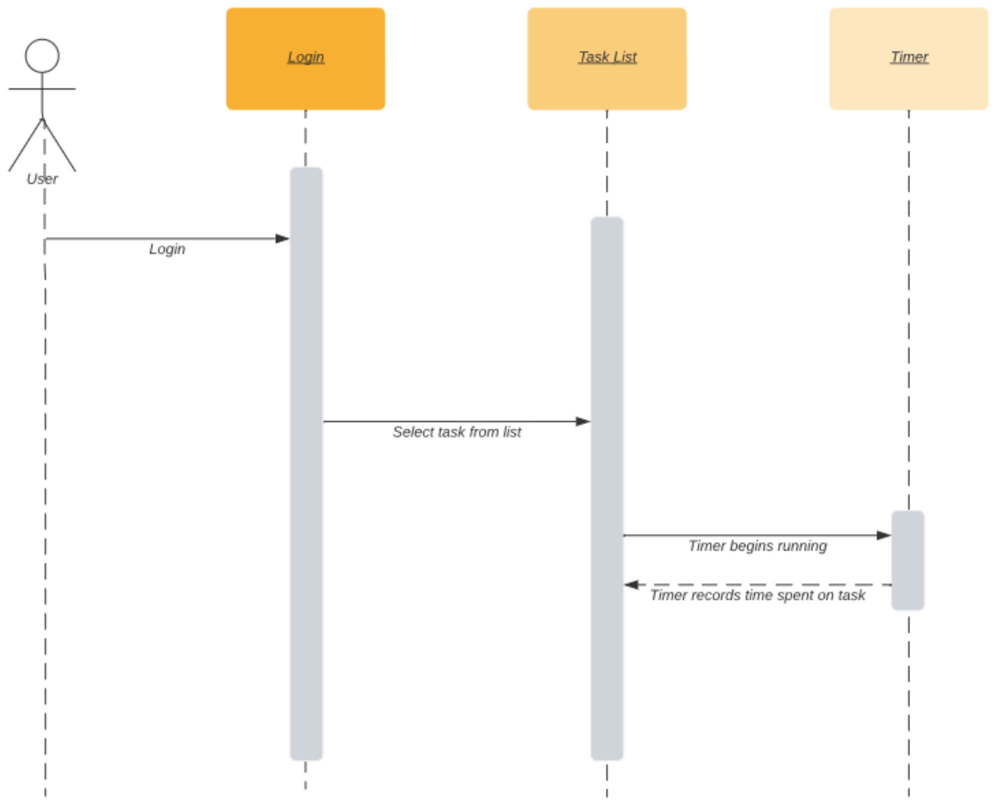
- User selects a task for which they want to track time.
- User starts a timer for the task [S1].
- User stops the timer when work on the task is complete [S2].
- System records the time spent on the task [S3].

Subflows:

- [S1] User starts a timer by clicking a "Start Timer" button.
- [S2] User stops the timer by clicking a "Stop Timer" button.
- [S3] System records the elapsed time and updates the task with the time spent.

Alternative Flows:

- [E1] The user forgets to stop the timer and needs to adjust the time manually.



Use Case 4: Set Task Priority

Preconditions

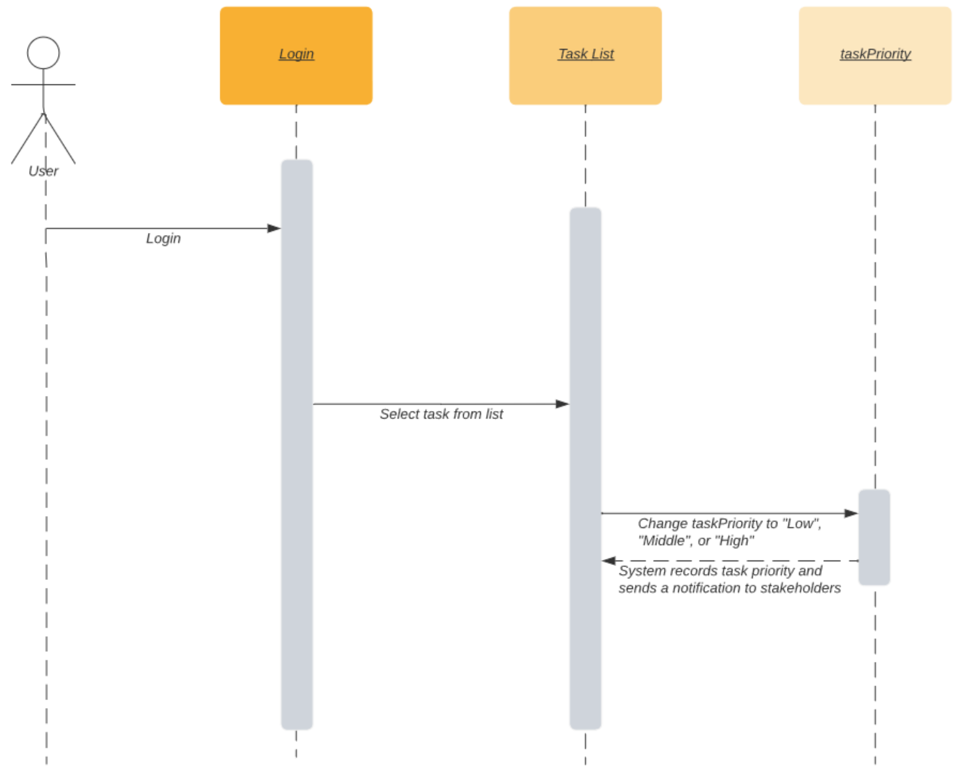
- User must be logged into the task management system.
- User must have the necessary permissions to update task priority.

Main Flow

- User selects a task from the task list. User updates the priority of the task (e.g., from "Low" to "Middle" or "High") [S1]. The system updates the task's priority and notifies relevant stakeholders [S2].

Subflows

- [S1] User selects the desired priority for the task from a predefined list.
- [S2] The system sends notifications to inform relevant stakeholders of the priority change.


Login
Task List
taskPriority
User
Login
Select task from list
Change taskPriority to "Low", "Middle", or "High"
System records task priority and sends a notification to stakeholders

Use Case 5: Generate Task Reports

Preconditions:

- User must be logged into the system.
- User has access to project and task data.

Main Flow:

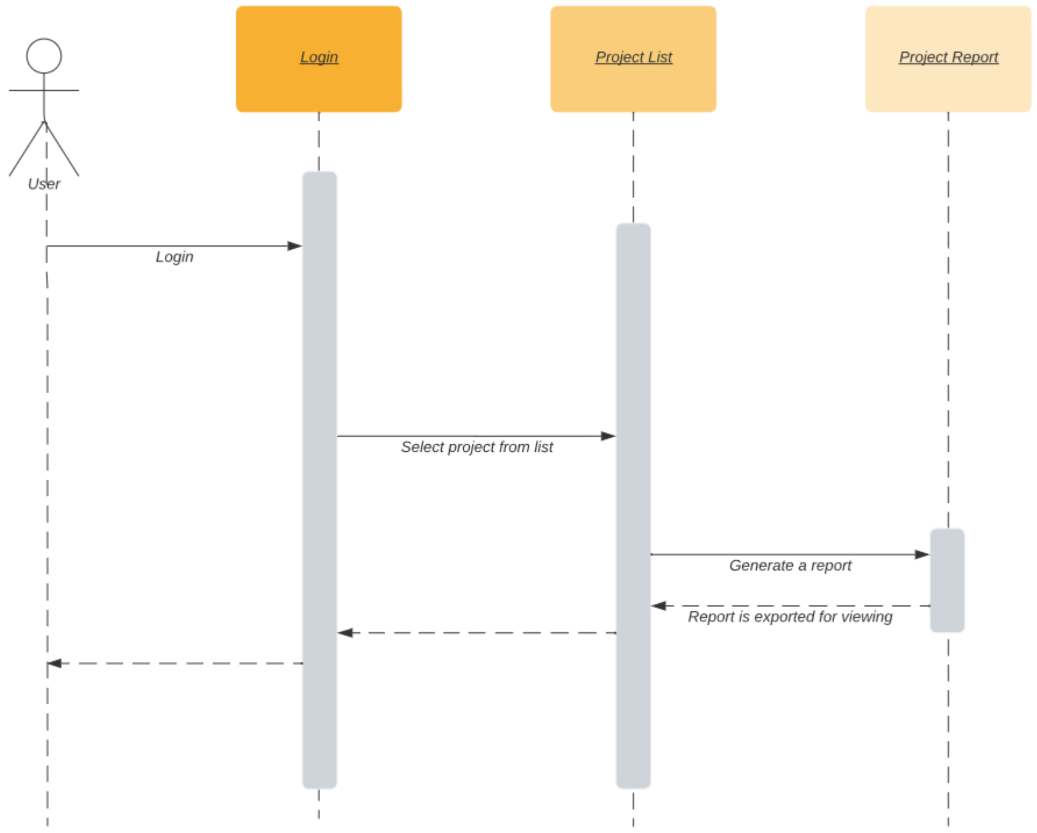
- User selects a project or a specific time frame for the report.
- User requests the system to generate a task report [S1].
- System compiles a report containing task details, time spent, and progress [S2].
- User views and exports the report [S3].

Subflows:

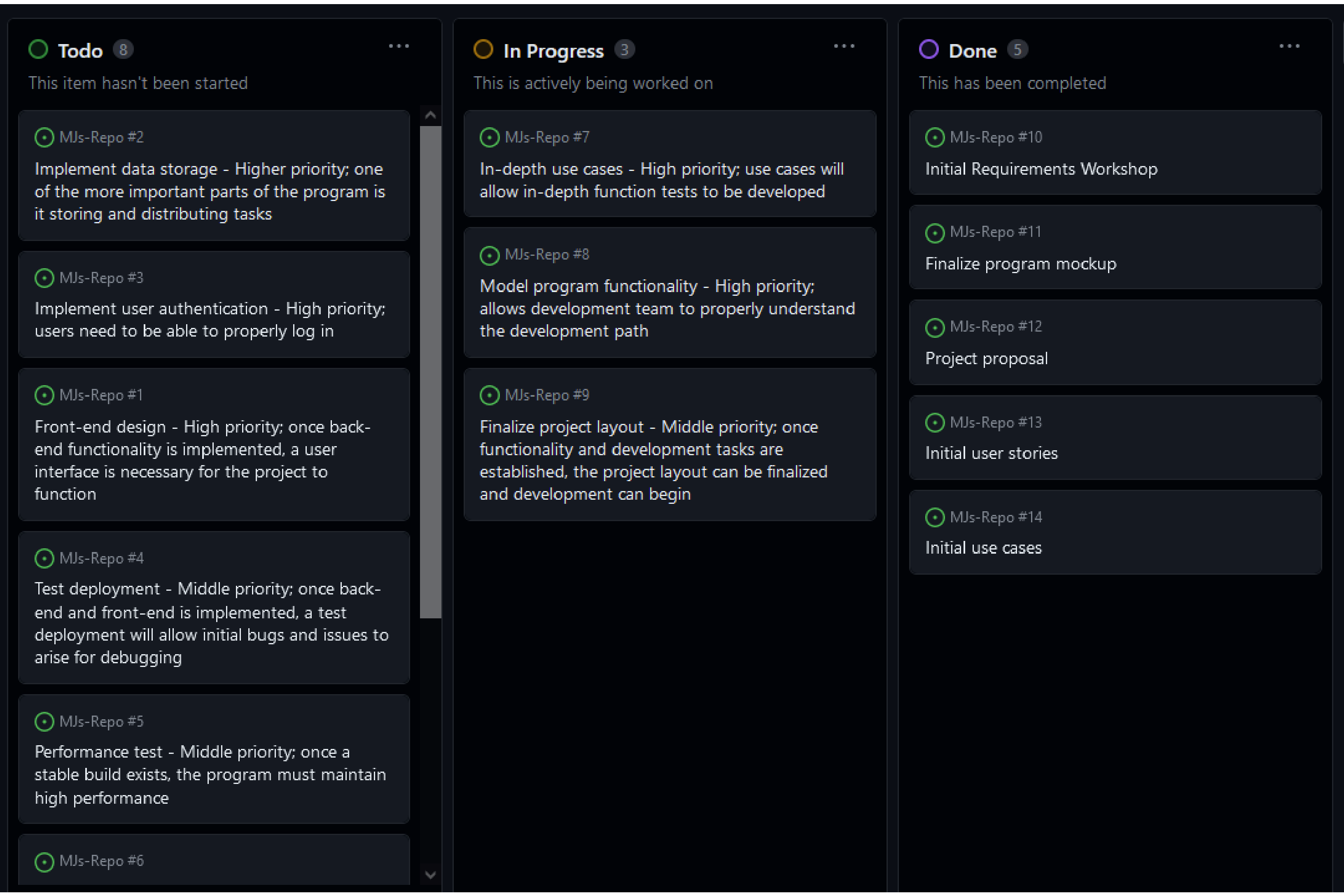
- [S1] User specifies report parameters (e.g., project, time frame).
- [S2] System generates the report with relevant task data.
- [S3] User can view the report in the app and export it if needed.

Alternative Flows:

- [E1] The user cancels the report generation.



Process Deliverable



Todo 8
This item hasn't been started

MJs-Repo #2
Implement data storage - Higher priority; one of the more important parts of the program is it storing and distributing tasks

MJs-Repo #3
Implement user authentication - High priority; users need to be able to properly log in

MJs-Repo #1
Front-end design - High priority; once back-end functionality is implemented, a user interface is necessary for the project to function

MJs-Repo #4
Test deployment - Middle priority; once back-end and front-end is implemented, a test deployment will allow initial bugs and issues to arise for debugging

MJs-Repo #5
Performance test - Middle priority; once a stable build exists, the program must maintain high performance

MJs-Repo #6

In Progress 3
This is actively being worked on

MJs-Repo #7
In-depth use cases - High priority; use cases will allow in-depth function tests to be developed

MJs-Repo #8
Model program functionality - High priority; allows development team to properly understand the development path

MJs-Repo #9
Finalize project layout - Middle priority; once functionality and development tasks are established, the project layout can be finalized and development can begin

Done 5
This has been completed

MJs-Repo #10
Initial Requirements Workshop

MJs-Repo #11
Finalize program mockup

MJs-Repo #12
Project proposal

MJs-Repo #13
Initial user stories

MJs-Repo #14
Initial use cases