[Requirements Workshop Questions 1-6]

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.

- **Usability**: should have a user friendly interface that's easy to navigate and provides clear error messages when needed
- **Reliability**: cannot fail often (more than once a month) and data must be recoverable to help keep teams on track
- **Performance**: must have quick response times to save teams time rather than lose it
- **Supportability**: must be maintainable for quickly changing systems and adaptable for different team types and structures
- **Implementation/Constraint**s: must indicate what software limitations and tools/languages that are available

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

- **User authentication**: Users will have a login and password unique to them
- **Task management**: Users will have the ability to create, assign, track, and delete different tasks in the system
- **Project management**: Users can create different teams and projects and organize them
- **Backups**: backs up data often (every few hours or every day) to make sure teams can keep their data safe from failure
- **Communication/Collaboration**: users must be able to communicate with one another

3. 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.

- User Authentication: 3 story points
    - Database to store different users
    - Frontend to allow users to enter their information, access website, and create an account
- Task Management: 2 story points
    - Frontend representation of tasks
    - Database storing tasks, categories for tasks, and handling access
- Project Management: 1 story point
    - Frontend to allow users to edit or create teams and projects
    - Regular updates across user boards to make sure all team members have access
- Backups: 4 story points
    - Frequent backups to save user data
    - Restoration on event of failure of lost data
- Communication/Collaboration: 2 story points
    - Store user chats between each other with set deletion
    - Frontend allowing users to easily switch between chats
- Usability: 4 story points

    - Create a frontend that follows HTML, CSS, and other standards
- Reliability: 4 story points (integrated with backups)
    - Use of backups supports reliability
- Performance: 5 story points
    - Make sure system runs successfully and in ideal amount of time
- Supportability: 3 story points
    - Must work and be maintainable across a variety of systems without significant loss of function
- Implementation/Constraints: 2 story points
    - Highlights project constraints and communicates that to user
    - System will function within the constraints

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

Actor 1: John Doey
John wants to set up his account for his teams management system

Acceptance Criteria:
- John is allowed in the website and his authentication data is saved to database
- John signs up for the application with his google account, is allowed access, and starts to manage his project.

Actor 2: Benny Gonzolas
Benny wants to create a new task in a team management system so he can assign it to team members

Acceptance Criteria:
- The project manager needs to be able to access tasks
- There should be a button to create a new task with task specifications (title, assignee, due date)
- Project manager can save task and assign it to a member

Actor 3: Tina Chompers
Tina wants to mark her tasks in the system as complete so her team knows it's done

Acceptance Criteria:
- Tina needs to be able to access her assigned tasks
- Each task should have an option to track work and mark it as complete
- When completing a task, Tina should be able to leave any comments related to her task

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: If the system has any security problems, team data may leak, so it is important to minimize any threats and document any security gaps. It is also important to regularly conduct tests that will ensure that data does not leak, any new threats don't arise, and that there is enough process documentation.

Loss of Data: If teams lose their boards, their project would largely be derailed as they wouldn't know which tasks have been completed and which tasks still must be done. To protect against this, we will have frequent backups of their projects and keep them ready to restore on the event of failure. It would also allow for merge conflicts when users update tasks at the same time as they can choose which parts of the backups they wish to keep.

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

We would follow an iterative process (agile) that would allow us to frequently meet with our client to get feedback on our development and allow them to give us further requirements. This will tell us which features to prioritize for our final design and which feature might not be as useful. It will also allow us to personalize features for the client's future use that would benefit their specific needs.

[Scrum Meeting Notes]

Meeting 1:
Date: 2/14/2024
Topic: Basic functionality
Members: Lucas, Mollie, Hannah, Shreya
Notes:

- Should follow similar design to Jiras
    - Backlog
    - Board
    - Reports
    - Settings
- The board will follow the essential workflow
    - Todo
    - In progress
    - Review process
    - Finished
- Settings will be highly customizable
    - In sprint length
    - In which team members/teams are involved in different meetings
    - Color, shape, size

- The custom sprint configuration will allow multiple settings to be changed
    - Total length
    - Length of each section
    - Type of sections in sprint
    - etc …
- Will promote better sprint behavior
    - Emphasize learning period
    - Follow sprint research?
    - Easier ways to implement story points?
    - Swarming - When a team swarms, it means that a team focuses most of their attention on the most important Sprint Backlog items. It allows teams to finish the most important parts of their project quicker than lesser value tasks.
    - Sprint review - maybe a way to emphasize ways to improve from old sprints?
    - Auto Schedule meetings between team members and teams? Or send reminders?

Meeting 2:
Date: 2/21/24
Topic: Use Cases Discussion
Members: Lucas, Shreya, Mollie, Hannah
Notes:

- Brainstorming some ideas for use cases for customizable sprints
    - Able to add meetings
    - Change sprint length
    - Make specific subteams
    - Different themes for site?
- Divide work:
    - Mollie: Record meeting notes, start work on use cases
    - Hannah: Revise requirements workshop (if necessary), start work on use case
    - Shreya: Start working on specific use case moving forward
    - Lucas: Start work on specific use case
- Software design ideas:
    - Known design from last meeting:
        - Should follow similar design to Jiras
            - Backlog
            - Board
            - Reports
            - Settings
    - Design revisions:
        - Need to make it easily accessible and user friendly
        - Updated interface compared to Jira
        - Customizable colors

        - Include a sidebar with different tabs for Backlog, Board, Reports
        - Settings button on the top right

Meeting 3:
Date: 2/27/24
Topic: Discussed use case work updates
Members: Lucas, Mollie, Hannah, Shreya
Notes:

- Mollie: Design a custom sprint use case
    - Worked on and completed the preconditions for my case
    - Planned out main flow for use case
        - Manager opens, changes settings, etc.
        - Might revise main flow for more detail
    - Did not start work on sub flow for main cases, too busy this week
    - Problems: No problems with understanding the work, I just have not had time yet this week to get to sub flow cases. Will discuss with teammates if I need further help.
    - Plan: Plan to complete by the end of this week
- Hannah: Add new task to sprint
    - Worked and completed the preconditions for the use case
    - Worked on the main flow for use case
        - Still need to complete and update main flow
        - Needed verification with team
    - Started planning out sub flow for sprints
    - Problems: had a few question and needed clarification with main flow cases
    - Plan: Plan to complete main and sub flow use cases
- Lucas: Changing task status
    - Worked on and completed the preconditions of the case
    - Finished main flow for changing task status
        - Main flow showed to team and confirmed they look good
    - Made progress on sub flows, but have not finished yet
    - Problems: No problems yet, needed clarification for sub flow cases but resolved with the team. Other class workload may be an issue for the next week.
    - Plan: Plan to complete sub flow cases early next week
- Shreya: Team enters a swarm
    - Worked on and completed preconditions
    - Worked and finished main flow cases
    - Worked and finished sub flow use cases
        - Competed, but needs verification with team
    - Plan out alternative flow cases
    - Problems: Had no problems, needed sub flow cases to be confirmed with the team.
    - Plan: Plan to get alternative cases started early and completed by next meeting

Meeting 4:
Date: 3/8/24
Topic: PM2 Completion and looking forward at PM3
Members: Lucas, Mollie, Hannah, Shreya
Notes:

- Completed our PM2 requirements
    - Mollie: Completed ‘Design a custom sprint use case’ Use Cases
        - Worked on and completed subflow cases
        - Started and completed alternative flow cases
        - Use cases discussed and verified with team
    - Hannah: Completed ‘Add new task to sprint’ Use Case
        - Completed main flow cases
        - Worked on and completed sub flow cases
        - Started and completed alternative flow cases
        - Use cases discussed and verified with team
    - Lucas: Completed ‘Changing task status’ Use Case
        - Completed subflow cases
        - Completed alternative flow use cases
        - Use cases discussed and verified with team
    - Shreya: Completed ‘ Team enters a swarm ’ Use Case
        - Worked on and completed alternative flow use cases
        - Added and fully completed ‘Create a schedule based off of developer’s time available’ Use Case
            - Verified case with team
        - Use cases discussed and verified with team
- PM3 Discussion
    - Need to decide on an architectural pattern for high-level design
        - Layered architecture
        - Event-driven pattern
        - Client-server pattern
        - Microservices pattern
    - Need to research patterns and discuss pattern families helpful for implementing the project
    - Use use case drawings to create a design sketch for the project
        - Need to make a main interface with side bar
        - Need to design each individual page and include buttons for features
        - Need to create a storyboard with the features above
    - Survey completion for project update
    - Keep up with scrum notes and submit

[Use Cases]

1: Design a custom sprint

Preconditions:

- A project team is formed
- A project manager is assigned

Main flow:

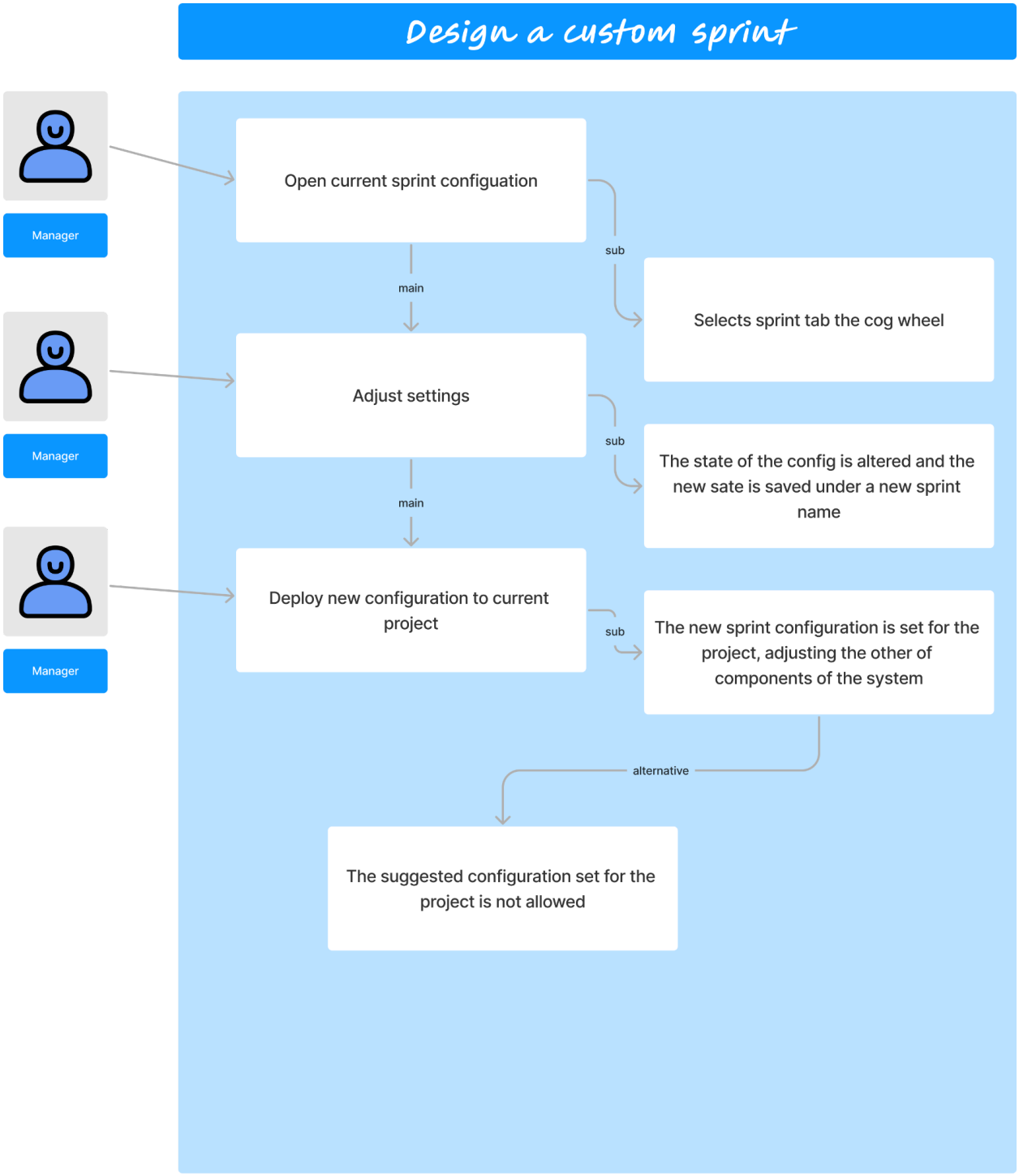
1. Manager opens the current sprint configuration [S1]
2. Manager adjust settings using the text inputs and sliders [S2]
3. Manager chooses to deploy this sprint configuration to the current project [S3]

Sub flows:

1. [S1] Manager clicks on the sprint tab, then the settings cog wheel
2. [S2] The state of the default config is altered and the new state is saved under a new sprint name
3. [S3] The new sprint configuration is set for the project, adjusting the other components of the system and what the team sees

Alternative Flows:

1. The suggested configuration is not allowed and needs to be changed



2. Adding a new task to sprint

Precondition:

- A sprint has to exist

Main flow:

1. Team member adds task to potentially be added to list [S1]
2. Manager reviews task [S2]

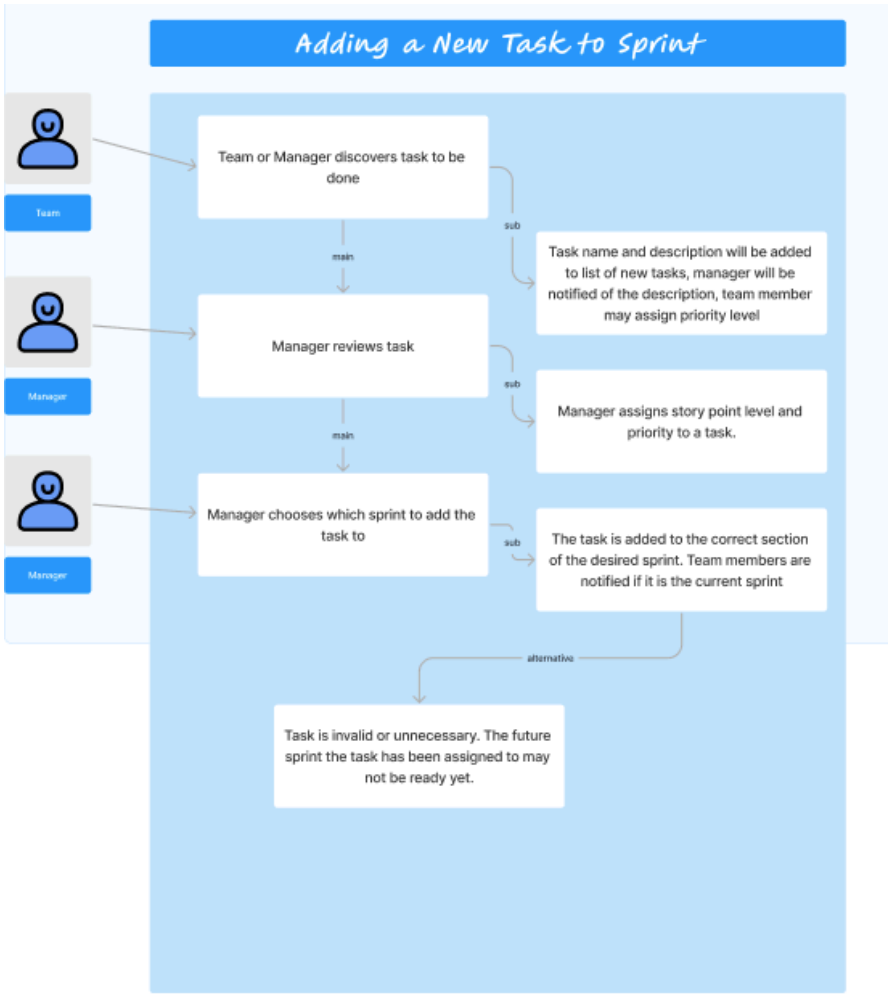
3. Manager chooses which sprint it will be added to [S3]

Sub flows:

1. Task name and description will be added to list of new tasks, manager will be notified of the description, team member may assign priority level [S1]
2. Manager assigns story point level and priority to a task. [S2]
3. If it is added to the current sprint, it is assigned to the desired section (ex. Todo, In progress); otherwise it will be added to a future sprint. Team members are notified of new task if it is the current sprint [S3]

Alternative Flows:

2. Task is invalid, or incomplete [E1]
3. Manager may decide task is unnecessary [E2]
4. Future sprint may not be created yet [E3]



3. Changing the status of a task

Precondition:

- A project has to exist

- One or more task have to exist

Main flow:

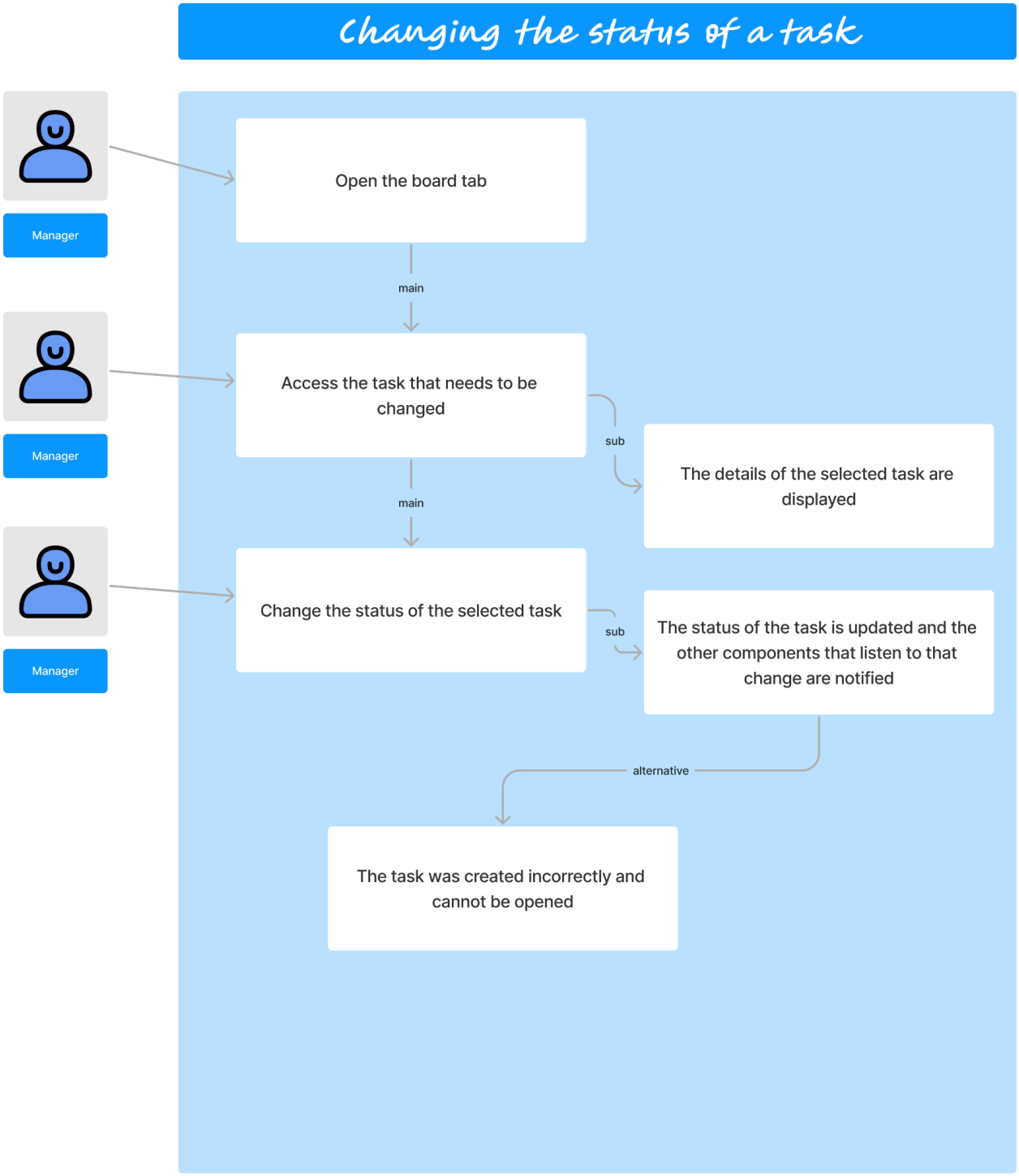
1. Manager opens the board tab in the main interface
2. They access the task that want to change the status of [S1]
3. They change the status of the selected task [S2]

Sub flows:

1. [S1] The details of the selected task are displayed
2. [S2] The status of the task is updated and the other components that listen to that change are notified

Alternative Flows:

1. The task was created incorrectly and cannot be opened [S1] [S2]



4. Team enters a swarm

Precondition:

- Team members all know what a swarm is
- Tasks have already been assigned priority levels

Main flow:

1. Manager decides to start a swarm. Team chooses priority items

2. Members are all notified of swarm status and begin to work on high priority tasks
3. As tasks finish, members are given next task to swarm on
4. Swarm is finished and team goes back to their regular work

Sub flows:

1. Relevant team members are notified of the swarm and meet to choose highest priority items. This list will be the only thing team members will be able to access during the swarm
2. All team members are notified on the platform and otherwise that a swarm is about to occur. They are only given access to high priority tasks on the board
3. As tasks get closer to completion, the next task in the queue will move up the list and become visible
4. All changes to the platform due to swarm are rescinded and lower priority tasks become visible again.

Alternative Flows:

1. Not enough or too many priority items
2. Members accidentally do the same work at the same time
3. No more tasks to swarm on before swarm ends
4. Team's work was finished by swarm

5. Create a meeting time based off of developer's time available
Precondition:

- A project exist
- Developers have filled out a survey specifying their availability

Main flow:

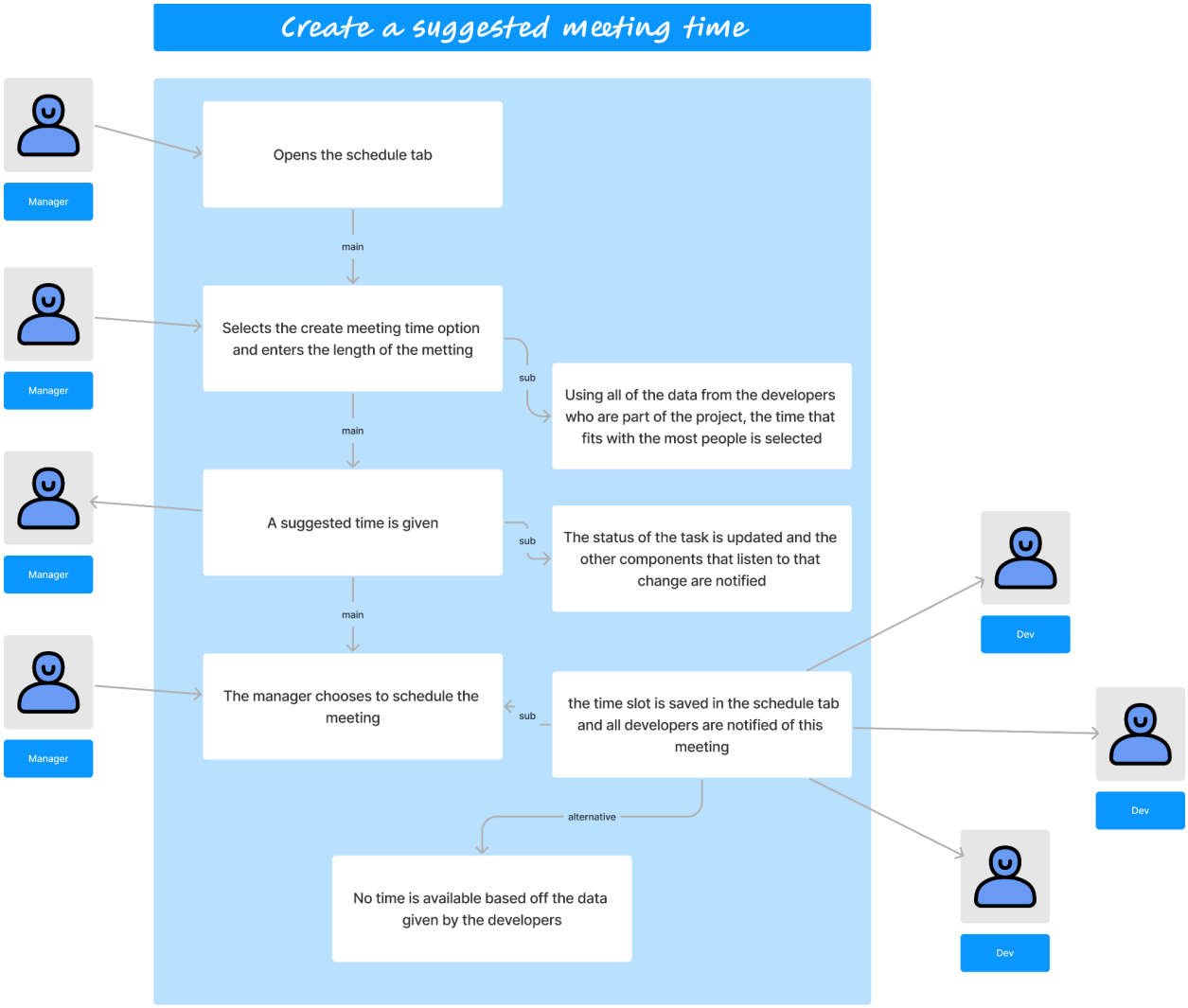
1. The Manager opens up the schedule tab in the main interface
2. The Manager selects the create meeting time option and enter the length of the meeting [S1]
3. A suggested time is given to the manager
4. The manger is given the option to schedule the meeting [S2]

Sub flows:

1. [S1] Using all of the data from the developers who are part of the project, the time that fits with the most people is selected
2. [S2] If yes is selected the time slot is saved in the schedule tab and all developers are notified of this meeting

Alternative Flows:

1. No time is available based off the data given by the developers [S1]


Create a suggested meeting time
Manager
Manager
Manager
Manager
Opens the schedule tab
main
Selects the create meeting time option and enters the length of the metting
sub
Using all of the data from the developers who are part of the project, the time that fits with the most people is selected
main
A suggested time is given
sub
The status of the task is updated and the other components that listen to that change are notified
main
The manager chooses to schedule the meeting
sub
the time slot is saved in the schedule tab and all developers are notified of this meeting
alternative
No time is available based off the data given by the developers
Dev
Dev
Dev