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.**

- **Usability:** Text on display must be black and a readable size.
- **Reliability:** When creating a task, the system must save every 3 seconds.
- **Performance:** When using the chat, responses should be sent within 2 seconds.
- **Supportability:** The system should allow frequent and easy changes within the tasks.
- **Implementation/Constraints:** Must be on IOS

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

- Must allow users to create tasks
- Must allow users to the chat feature to communicate
- Must allow users to add code to the tasks and chat
- Must allow users to delete tasks
- System must send notifications to Users

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.**

- Set all color for text boxes to be black
    - Estimated function points: 2
    - For this task we just have to make sure all the textboxes that we use are black, which will not be hard to do, hence we are giving this a 2
- Implement an autosave feature that is trigged every 3 seconds
    - Estimated function points: 6
    - We are not 100 percent sure how to do this yet which means we will need to do some research and aslo some trial and error which is why we are giving this a 6
- Optimize chat system to ensure that responses are delivered within 2 seconds
    - Estimated function points: 10
    - It will be hard to make it to our 2 second time constraint which is why we are giving this a 10

- Implement a task management system that allows for easy updates
    - Estimated function points: 10

        - This will take quiet a bit of effort so we are giving this a 10

- Research how to develop a system to compatible with iOS
    - Estimated function points: 2
    - Since we are just researching and learning, this should not be hard to do, so we are givint his 2 points.
- Develop a user interface to input task details such as title, description, and deadline
    - Estimated function points: 5
    - This task might require some trial and error, so 5 seems appropriate
- Implement a real-time chat feature with the ability to send and receive messages.
    - Estimated function points: 10
        - Creating a chat feature requires front-end and back-end development, real-time communication, so 10 points seems appropriate
- Enable users to attach code snippets or files to tasks and chat messages
    - Estimated function time: 7
        - This task involves adding file upload functionality, so a 7 seems fair
- Develop a feature to delete tasks, considering data removal
    - Estimated function time: 6
    - We need to have some user prompts so a 6 seems fair
- Implement a notification system to alert users of task updates or new messages.
    - Estimated function time: 9
    - We need to have push notification services so a 9 seems fair

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

- As a programmer I want to be able to link my code repository to my task.
    - Acceptance criteria:
        - **Given:** Programmer is logged into account
        - **When:** Programmer attempts to link code
        - **And:** Programmer provides all the necessary information
        - **Then:** System allows code to be linked and viewed on task
- As a project manager I want to be able to view what my team is working on.
    - Acceptance criteria:
        - **Given:** Project Manager is logged into account with proper admin rights
        - **When:** Project Manager clicks on team
        - **And:** Project manager goes to view all tasks

        - **Then:** System allows Project Manager to view all team members tasks and progress
- As a project manager, I want to be able to assign tasks to my team members.
    - Acceptance criteria:
        - **Given:** Project manager is logged into account with proper admin
        - **When:** Project Manager clicks on individual team members
        - **And:** Project Manager goes to assign tasks
        - **Then:** System will notify team members of new 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.**

Two examples of risk that could potentially impact this project are delegation of priorities due to the code and fix model and overcomplexity that is not previously thought of. The way to mitigate these risks is to make sure that there is an understanding of what is required before delegating tasks which can help remove any confusion in the future and not have surprises while part way through the project. Discussion is also very important to make sure that everyone is on the same page. Discussion makes sure that everyone understands their priorities and will be able to log and report all issues and possible overcomplexities that they may think of for the future of the project.

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

This process is what our team would use for requirements elicitation:

- Requirements being given to our team
- Requirements understanding
- Requirements added to project design
- Requirements prioritization
- Requirements delegation to appropriate team members
- Requirements fulfilled
- Requirements and completed product shown to client/customers for confirmation

We do this process because it allows us to cover all our tracks and the requirements when working through the project. We need to make sure we understand the requirements that are given to us so that we can properly add it to our project design. Then we will be able to prioritize the most important parts of those requirements and which will take the longest time to complete, so we can delegate the tasks accordingly for best time optimization. After the delegation work can be started and when it is completed we would go back to the customer/client to check that it fulfills everything that they were imagining and wanted in the project.

Use cases

Use Case: Creating A Task

1 Preconditions

User must be logged into their account

2 Main Flow

User will input a task and provide a description[S1].

3 Subflows

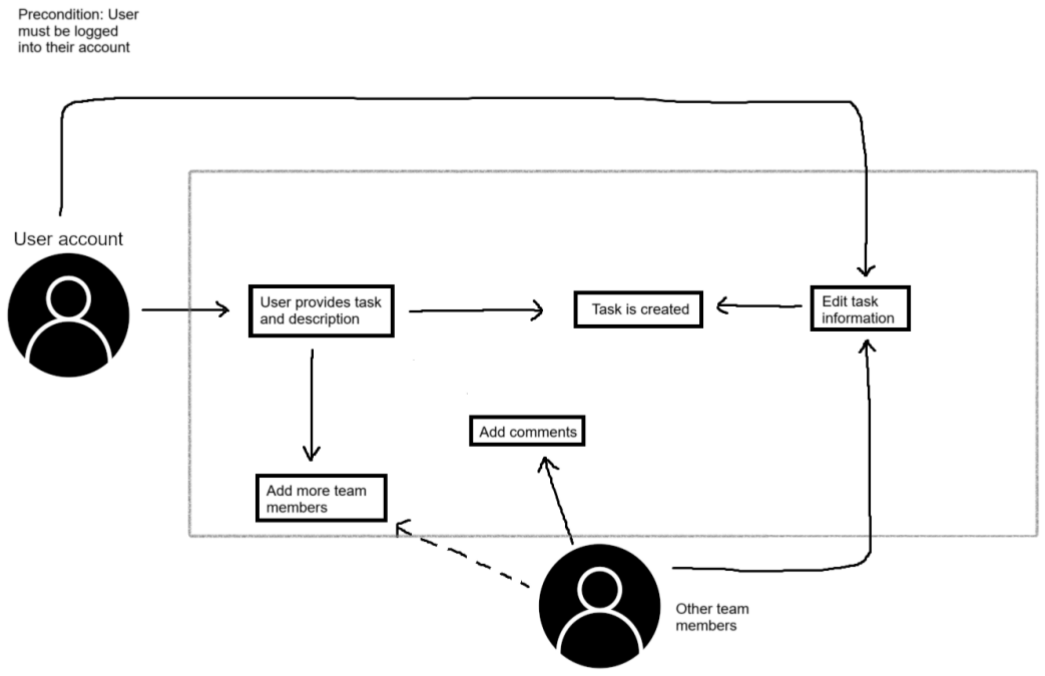
[S1] Users can add any additional team members to tasks.

[S2] Other users can see the created task and comment on it.

[S3] All users can discuss tasks and make adjustments.

4 Alternative Flows

[E1] No changes to tasks are needed.



Use Case: Engage in Real-Time Chat

1 Preconditions

User must be logged into their account.

User navigates to a specific task.

2 Main Flow

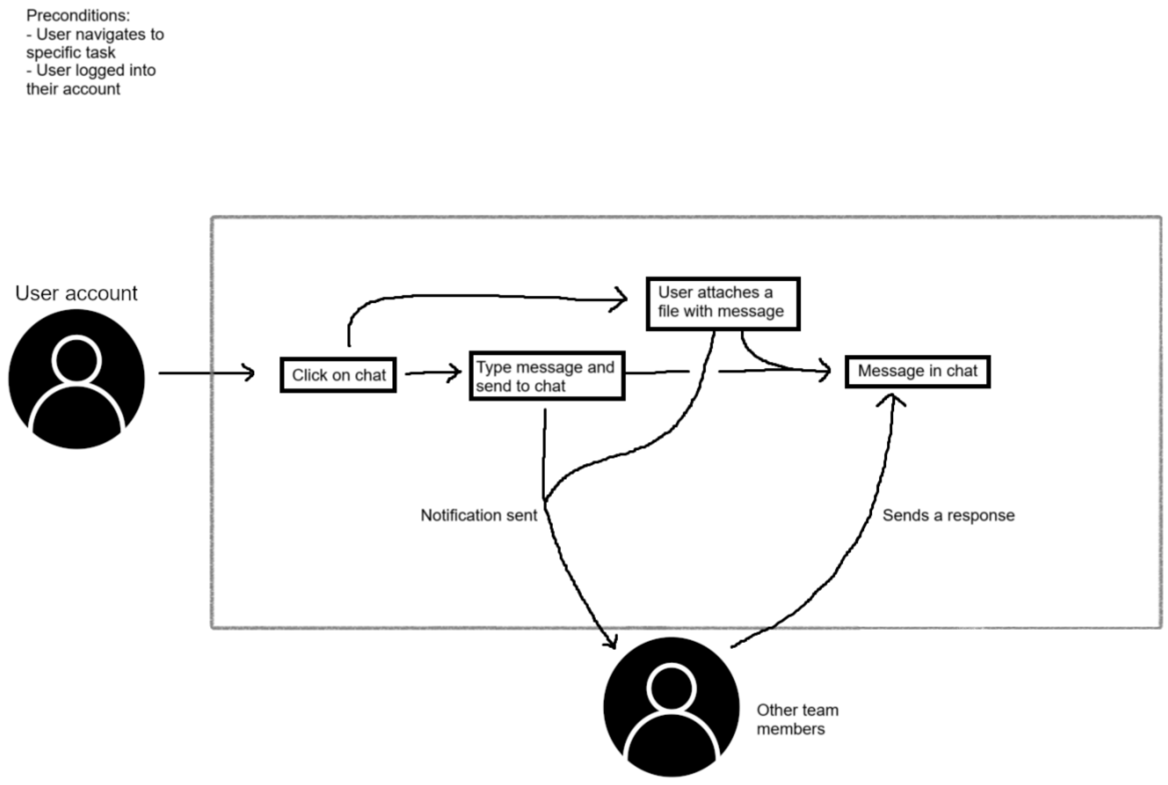
User will click on “Chat” associated with task[S1]. User types messages and sends[S2].

3 Subflows

[S1] Team members receive notification and can respond.

4 Alternative Flows

[E1] User can attach files in chat along with text.


Preconditions:
- User navigates to specific task
- User logged into their account
User account
Click on chat
Type message and send to chat
User attaches a file with message
Message in chat
Notification sent
Sends a response
Other team members

Use Case: Integrate Code into Task

1 Preconditions

User must be logged into their account

2 Main Flow

User will select a task and click on “Integrate Code”[S1]. System will prompt for repository details[S2]. User provides details and confirms[S3]. System links the codebase to the task[S4].

3 Subflows

[S1] Users can add any additional team members to tasks.

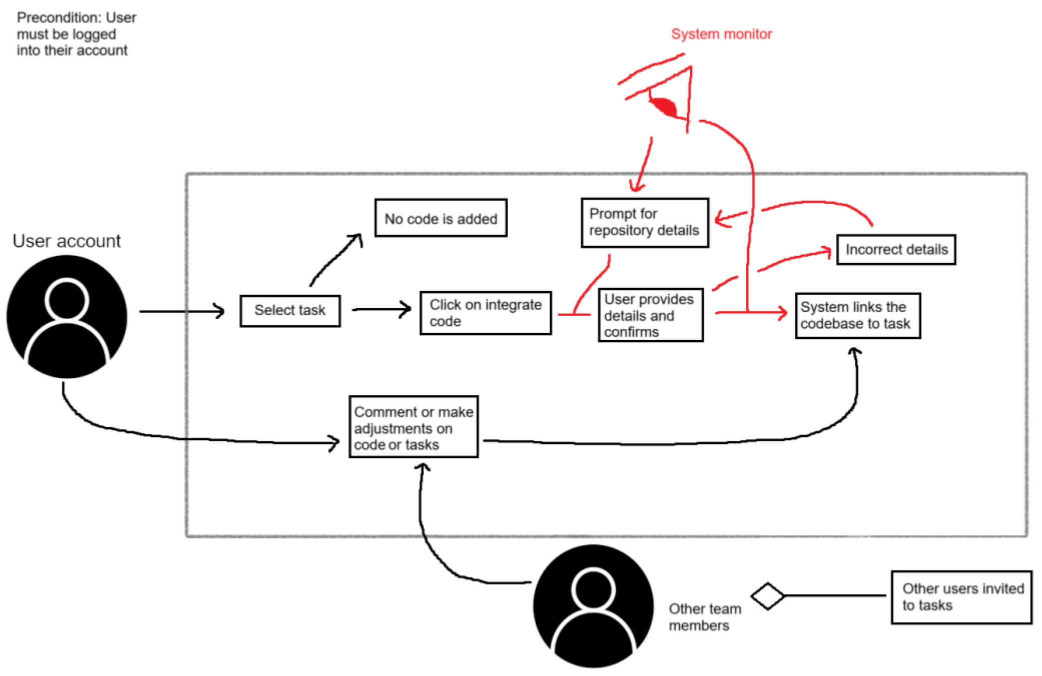
[S2] Other users can see the code and comment on it.

[S3] All users can discuss tasks and make adjustments.

4 Alternative Flows

[E1] No code is added to the task.

[E2] Right repository details were not entered so system reprompts.


Precondition: User must be logged into their account
System monitor
User account
No code is added
Prompt for repository details
Incorrect details
Select task
Click on integrate code
User provides details and confirms
System links the codebase to task
Comment or make adjustments on code or tasks
Other team members
Other users invited to tasks

Use Case: Task Notifications

1 Preconditions

User must be logged into their account

2 Main Flow

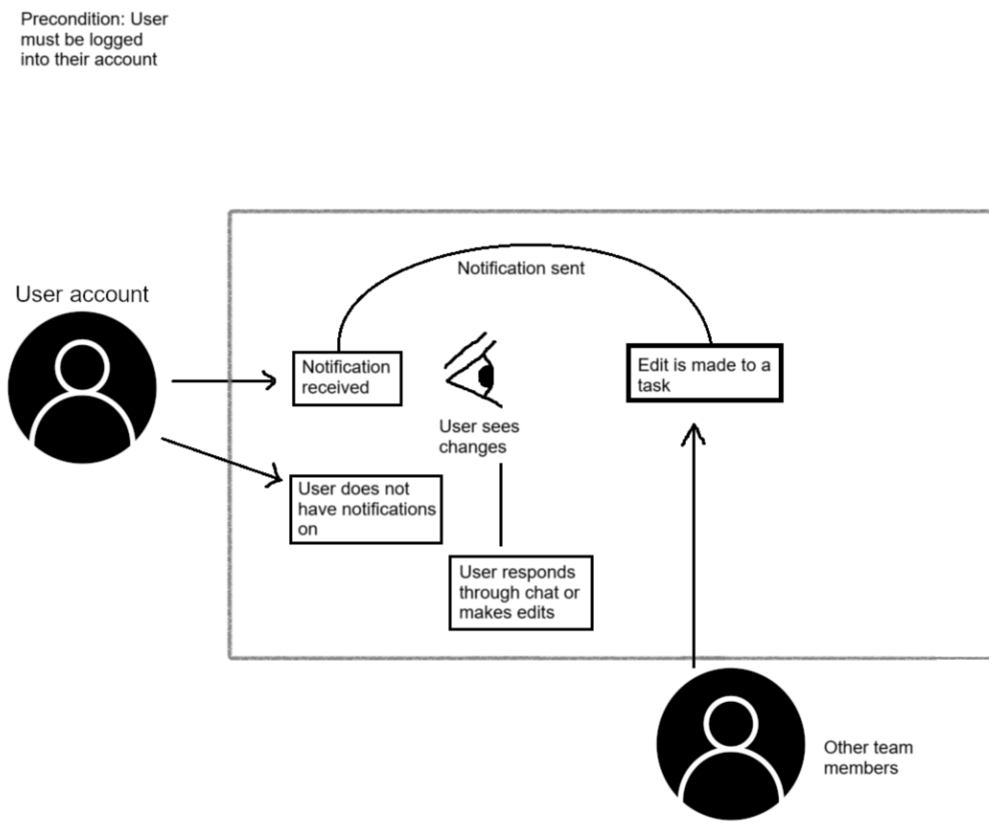
Another User modifies task[S1].System seeds real-time notification to relevant users[S2].

3 Subflows

[S1] Users can view notifications and respond.

4 Alternative Flows

[E1] User does not have notifications turned on.


Precondition: User must be logged into their account
User account
Notification sent
Notification received
User sees changes
Edit is made to a task
User does not have notifications on
User responds through chat or makes edits
Other team members

Use Case: Collaborative Editing

1 Preconditions

User A & B must be logged into their account

2 Main Flow

User A edits task details[S1].User B sees real time edits made by User A[S2]. User B can also make edits[S3]. System changes both Users changes[S3].

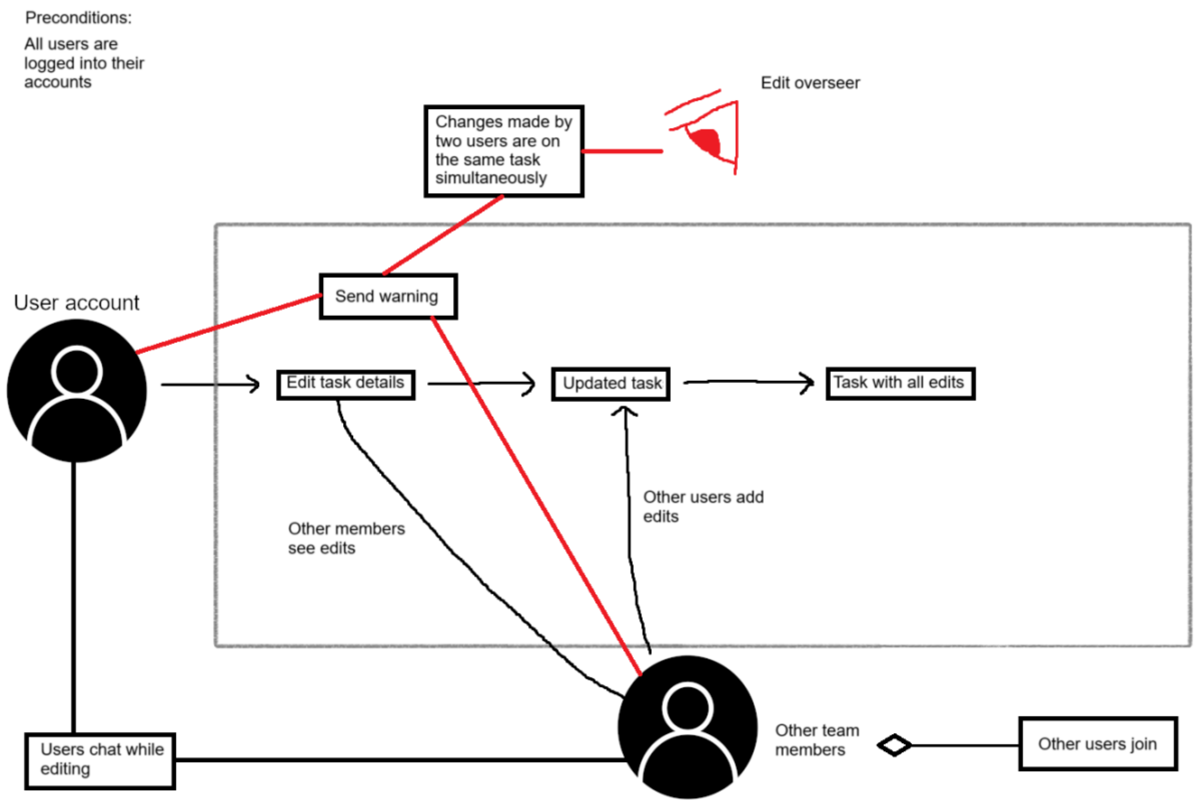
3 Subflows

[S1] More users can join.

[S2] Users can use chat features to communicate.

4 Alternative Flows

[E1] Users change the same thing, then System warns them.



Why Code and Fix ?

In the ever-evolving landscape of software engineering, one fundamental challenge stands out: the need for efficient task management and seamless collaboration in an increasingly fast-paced environment. Software engineering projects often involve teams that struggle with scattered tasks, poor communication, and disorganized workflows. As the developers of the CheckMate application, we recognized the pressing need for a solution that would empower software engineers to overcome these challenges and streamline their work processes. So they can stop worrying about the planning and organization and get to creating. To address these issues, we decided to adopt the "Code and Fix" methodology for our project. We will explore the reasoning behind our choice to use “Code and Fix”, demonstrating how this approach aligns with the unique requirements of our collaborative ToDo Application for software engineers.

Given the dynamic nature of software development, traditional, plan-heavy methodologies may not always be the best fit. Handwritten to-do lists or basic digital notepads fall short in addressing these needs. "Code and Fix" is a methodology that aligns with this dynamic reality, as it emphasizes rapid development followed by testing and immediate fixes. We, as Software developers, are able to jump into coding without the need for extensive planning. Not relying on extensive planning or documentation upfront is one of the distinguishing features of this approach. Instead, it encourages developers to dive into coding from the outset. In our case, developing a collaborative ToDo Application, we understand that the scope and requirements of such a project may evolve as we gain more insights and feedback. The "Code and Fix" methodology enables us to remain flexible and adapt to these changes quickly. We are able to take feedback and learn from testing and implement changes at a fast pace. We saw the need for a development approach that aligns with our goal of enhancing collaboration.

Real-time chat and code integration are core features of our CheckMate application, facilitating efficient communication and minimizing the potential for overlapping work and unorganized ideas. The "Code and Fix" methodology complements this objective. It allows us to work on the coding part of the application rapidly while addressing issues or new requirements through immediate fixes and improvements. This dynamic approach ensures that our team remains on the same page, making our work smoother and more efficient.

The "Code and Fix" methodology is also well-suited for small-scale projects in a group setting. CheckMate is a focused application designed for software engineers. While some methodologies may be more appropriate for large-scale, long-term projects, our emphasis is on creating a lean and effective tool for task management and collaboration. By adopting "Code and Fix," we avoid the complexity and overhead associated with extensive planning and documentation that may not be necessary for our project's scope. This approach allows us to allocate more time and effort into the development and fine-tuning of the application, aligning with our goals of efficiency and rapid progress. We are able to pay closer attention to finer details as we code, without getting bogged down on planning intensely.

Overall, we found that “Code and Fix” was the best approach for this project because it allows us to skip the heavy planning and get coding immediately, it is better suited for smaller scale projects with less people, and because it allows us to be dynamic and emphasizes rapid development followed by testing and immediate fixes.