I. Marathon - CUSTOMIZABLE SCRUM SYSTEM

A. Mollie Mero (molliem16)
B. Hannah Solis (hannahs2)
C. Shreya Thothathri (shreyathoth)
D. Lucas Kopp (lucaszacharykopp)

II. ABSTRACT

Many companies struggle with finding project management systems that are suited for their team's needs. Some teams prefer longer sprints and daily standups, while others prefer the opposite. Furthermore, project managers prefer automation over manual input. Having to collect availability of developers or notify developers of tasks or other announcements is seen as wasted time. The Marathon team planning app allows teams to create a system that's suited to their specific team instead of the other way around.

III. INTRODUCTION

Many companies struggle with scheduling sprints. A standard sprint doesn't always suit every team. With many teams having different requirements from their sprint, they need the ability to customize how often they do standups and how long their sprints are. It could even account for periods between sprints where teams can prepare for upcoming projects by gaining skills necessary or discuss with other teams how the sprint may be improved. When necessary, it could even mark in calendars when people within or across teams need to meet at the end of their sprint to discuss their next step. Our more customizable platform will allow for teams to change sprint requirements for their needs. Beyond helping the overall team, Marathon helps the managers of the project by saving them time. Meetings would be automatically created and developers would be notified of any important changes the manager makes.

IV. MOTIVATING EXAMPLE

A team is using an older scrum system that has less customization. The scrum master notices that during sprints a majority of the time is spent preparing for codebase changes or learning the content that needs to be applied; the sprint time is not being spent correctly. The default sprint configuration is not working for this project team, but it is not possible to change the settings. Furthermore, this manager spends lots of their time scheduling and notifying their team. They wish there could be a system that attempts to alleviate some of the management roles.

V. BACKGROUND

Key Terms:

A. Agile: The Agile methodology is a project management approach that involves breaking the project into phases and emphasizes continuous collaboration and improvement. Teams follow a cycle of planning, executing, and evaluating (Atlassian).
B. Sprint: A sprint is a short, time-boxed period when a scrum team works to complete a set amount of work (Atlassian).
C. Scrum: Scrum is an agile project management framework that helps teams structure and manage their work through a set of values, principles, and practices. Much like a rugby team (where it gets its name) training for the big game, scrum encourages teams to learn through experiences, self-organize while working on a problem, and reflect on their wins and losses to continuously improve (Atlassian).

Related Work:

A. When doing Agile planning and methodology, it is important to plan sprints in the most optimal way. It is essential to combat the issue of sprint planning so that teams can work the best in the most optimal way. Some key ideas include having the most appropriate planning time, set sprint goals, project goals, etc. (Golfarelli et.al, 2014)
B. In Onur Erdoğan's report, "more effective sprint retrospective with statistical analysis", their team analyzed the correlation between "Story Point and Actual Effort' ' in order to improve the efficiency of future sprints. Essentially, by breaking down the effort per story point, the point system assigned to each task, the team was able to find ways to improve product quality and efficiency. (Erodogan, 2012)

C. By exploring patterns found in high performance scrum teams, this paper discovers that teams that follow certain patterns finish earlier and accelerate through work faster. Some of these patterns include swarming, interrupt patterns, daily clean code, and emergency procedures. 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. Another pattern found in high performing teams was that they allotted time for interruptions based off of past projects. Each unexpected item went through the product owner for priority and often would be pushed towards subsequent sprints if they were not found to be critical. Our platform could implement easier ways to push tasks between sprints and assign multiple people to higher priority tasks. It could also take down low priority tasks during "swarm" times, allowing teams to focus their full effort towards the biggest problems (Sutherland et.al, 2014).

Tools:

The majority of the tools we used were during the planning phase of the project. To start, we used Google docs for scrum note taking and document generation in general. Next, Google slides allowed us to present updates to the class and envision potential UI decisions. We implemented these UI decisions using Figma;

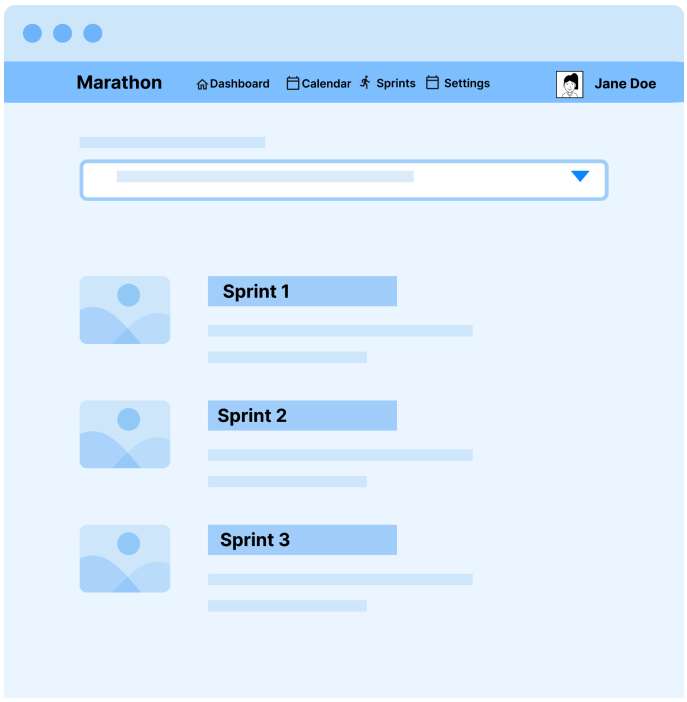
Figma was also used to create the use case diagrams. Finally, to keep track of all of the documents we used Git version control through GitHub.

VI. SOFTWARE ENGINEERING PROCESS

Processes:

Over the course of our project, we plan on using the Scrum methodology as our software engineering process because it is the most relevant process related to our system. Scrum involves having a set project goal and backlog, and planning work for each sprint. Our team will meet at the beginning of the sprint to plan our goals for that sprint. We will also hold near daily standups to update each other on our work and any problems we run into. Afterwards, we will have a retrospective meeting where we discuss if we met our sprint goals and what needs to get done or change in the next sprint. This method will be most effective for our team because we will be able to update each other constantly and work on the project iteratively.

Design decisions:



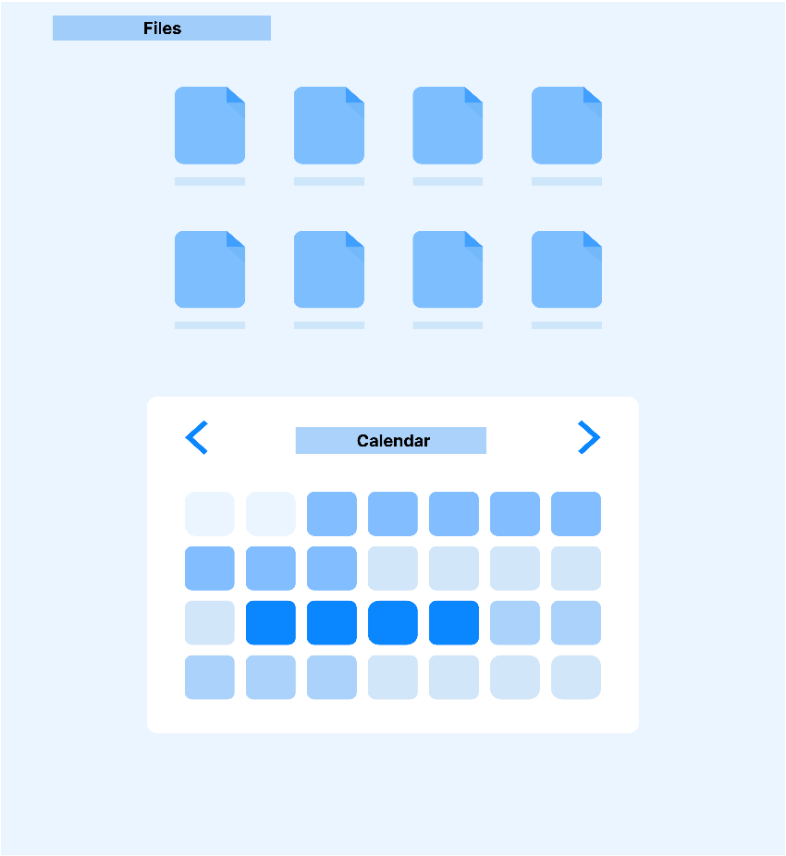
Home Page:

1. Allows users to navigate between their sprints
2. Allows users to search for current and past sprints

3. Automatically shows current sprints first, then upcoming sprints
4. Shows the name of a sprint, the current stage, a short description, and a project picture

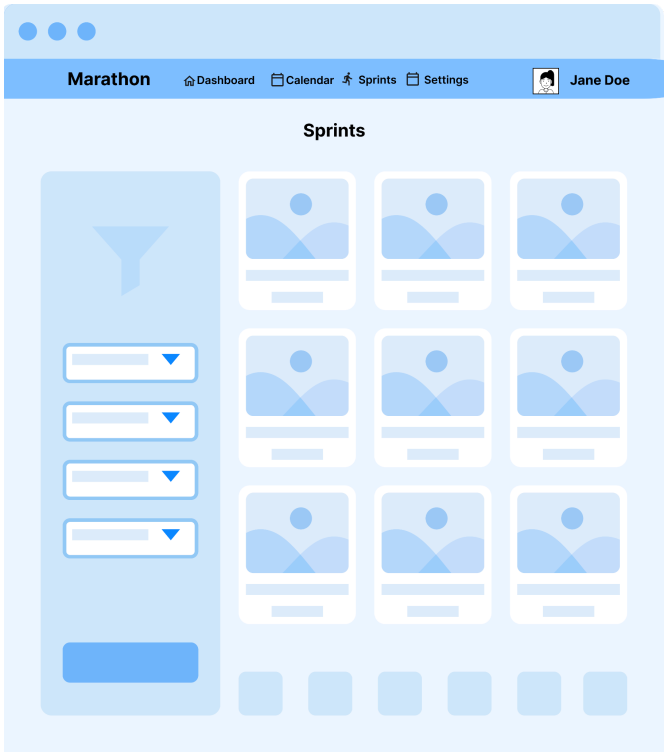
Specific Sprint Page

1. Shows the five basic sprint sections - Todo, In Progress, Code Review, and Done
2. Each section will have a task with
    a. Name of task
    b. People working on it
    c. Short Description including necessary skills
    d. Story Points
3. Allows any user to add to a section
4. Shows the team members that are a part of the sprint



Specific Sprint Page Pt 2

1. Allows team members to upload and access all relevant files to the sprint
2. Shows the stages of the sprint in the calendar in different colors with the current stage highlighted
3. Sprint is highly customizable with phase type, total length, and phase length
4. Will also have a phase for learning included in the sprint to make sure team members are prepared for upcoming phases

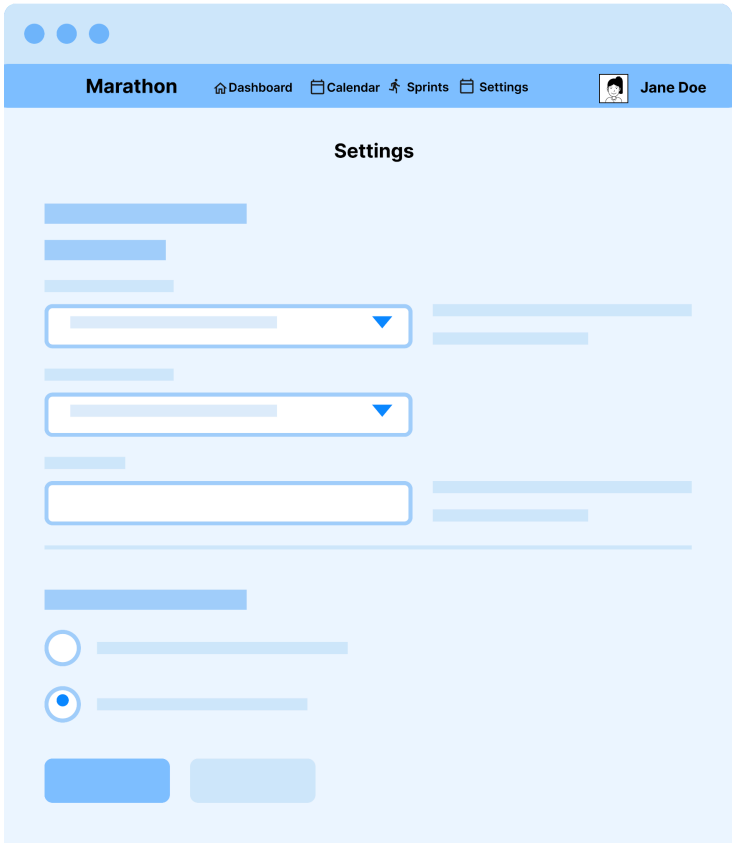


Sprints Page

1. Allows users to filter through past and current sprints
2. Can search by
    a. Team members involved
    b. Projects
    c. Length of Sprint
3. Will list recently used sprints before search
4. Can be used for sprint reviews or to easily find certain files

Swarm Changes

1. When swarms occur, all team members receive notifications and the sprint page will automatically link to the specific sprint
2. High priority tasks will be highlighted in red and lower level tasks will be either blue hidden until the swarm is over
3. Only the team manager will be able to call a swarm



Settings Page

1. Will allow users to change accessibility settings (ex. Contrast, Color Scheme)
2. Will allow users to change their login information, email, language
3. Allows users to change notification settings between different projects or phases

Testing approach:

Our team was unable to create a working implementation so we were also unable to test.

VII. Deployment

Deployment strategy:

We have yet to deploy our project, so the following section is an explanation of how we could potentially do so. Before deployment, we would actually have to create something that can be deployed. The frontend would most likely be powered through a modern JS framework such as svelte or react and the

backend would also use a JS framework due to popularity and ease of use. After actually creating the product, we would most likely utilize AWS for hosting our web application and backend.

Maintenance strategy:

The code will be managed on a version control software like GitHub, GitLab, or GitBucket. This is where any changes are stored and implemented for future updates. To ensure a stable product, dependency management will be automated through services. After any major or maintenance changes, the project will be redeployed. Additionally, build tools like Travis or Jenkins will be used to automate testing and other requirements.

VIII. Discussion

Limitations:

The majority of our limitations stemmed from our team. We are only a team of size four; splitting up tasks became extremely difficult. We ended up working on the same tasks at the same time which increased the amount of time needed for completion. Furthermore, in terms of general knowledge on the topics deployed in this project, we lacked. We do not have much experience in designing software or software interfaces. Overall, time was an issue. Due to the project being limited to a portion of a semester and most of that time being spent on planning, the furthest we got was creating a wireframe model.

Future work:

Our first future task is to turn the wireframe into an actual interface; this is detailed in our deployment section. Next, we would like to perform testing on this interface and the functionality it allows. After analyzing the testing results, we would update the software to fix the issues found. Then we would furthermore polish the software after adding new functionality. Some of these new features or functionality would be: Ability to share sprint configs (exporting configuration), Add more roles for team members, other than dev and manager, Direct messaging between members or sharing of contact information on profile, and Direct surveys for developers to get project feedback. This overall process would be repeated in an iterative way.

IX. Conclusion

Many project managers would prefer to have a customizable scrum system where sprints are easy to change and automation is the focus. Our team was able to model Marathon, a customizable scrum application, that addresses these problems. Marathon would have the ability to use custom sprints that allow for different lengths and periods. Furthermore, it would have the ability to auto

schedule based on given times and automated notifications for events like swarms. We developed use cases, wireframes, and other plans for the software design. Future development could include creating a functional demo and deployment onto a web service like AWS.

X. References

https://www.atlassian.com/agile/scrum/sprints
https://www.thedesigngym.com/3-ways-customize-design-sprint-project/

https://medium.com/dallas-design-sprints/how-to-customize-your-next-design-sprint-4d0d4b6c0654

https://www.scrum.org/resources/blog/5-most-common-pitfalls-scrum-events

https://onlinelibrary.wiley.com/doi/full/10.1002/smr.1933

https://ieeexplore.ieee.org/abstract/document/6759182