CSE 210 Team 10 Pitch

Statement of Purpose

Developer dashboards were created due to the need for an organized, centralized hub for tools and task tracking. Developers have a clear timeline with deadlines, schedules, and meetings. They must easily understand dependencies and distribute their time effectively to succeed at their jobs. There are many types of developer dashboards, each with their own functions. One of the most popular dashboards utilized by organizations is Microsoft Teams[1], which integrates with Code Helper (an AI helper tool), Azure DevOps work items, GitHub issues and Microsoft Planner tasks. They also have a dedicated Features Backlog section. In terms of the design, there aren't many charts and the visuals are simple, the Code Helper tool appears to take a significant amount of space in the UI, and there appears to be no dedicated place for links or task details. It also has communication features like Slack or Google Meet. However, Microsoft Teams doesn't consider any prioritization or tagging of tasks by urgency. It also does not provide the ability to add more details, links, or an estimate of how long it will take to complete the task. A centralized place for these types of features, in addition to the GitHub integration, would be extremely helpful to developers who are trying to efficiently meet deadlines.

With students, even fewer alternatives exist. Although there are applications like Trello for general organization and task progress, we found no dashboard with features specific to students that would allow prioritization based on deadlines, track multiple projects in different classes, and provide some guidance and integration with version control or industry tools. Therefore, a student dashboard would be a novel idea to pursue. Next is a dashboard optimized for software development managers. Some key parts of this job include keeping track of performance goals and progress, enforcing deadlines, coordinating with others including team members, other teams, and clients. Managers must also be able to schedule meetings, and assign tasks effectively. Existing metrics dashboards are the most relevant manager related product that exists currently[2]. However, it does not show deadlines or include task assignment or communication app integration; essentially, it only includes Jira-related features and metrics. Overall, although there are many dashboards for specific use cases, there are few that centralize most of the features that would be helpful for each user persona.

User Personas

Student Persona

Name: Studious Samantha

User Story: Samantha is a graduate CS student at UCSD, taking 3 rigorous classes this quarter. A large part of her daily life as a student revolves around staying on top of assignment deadlines, class schedules, and meetings. Furthermore, every class, and sometimes each assignment, has different assignment submission platforms, links, and documents. This makes it overwhelming for Samantha, who needs to somehow keep track of her assignments and effectively manage her time to complete them before the deadline. Additionally, she does not have much industry experience, which often makes it difficult for her to decide the best tool or framework to use for an open-ended assignment. With the chaos of juggling 3 very different classes, Samantha needs an organized approach to prioritize, keep track of her progress, and easily make her deadlines.

---

[1] https://github.com/aycabas/dev-dashboard

[2] https://github.com/aricma/developer-dashboard

Developer Persona
Name: Developer Dan
User Story: Dan is a mid-level software developer at a tech company. His job revolves around the many tasks he must complete before their given deadlines. He has a clear timeline, with well-defined deadlines, schedules, and meetings. A large part of his job involves collaborating with other people and understanding the dependencies tied to his work. To effectively do his job, he must distribute his time efficiently across the various tasks and projects to which he has been assigned and prioritize to complete them in a timely manner. Dan must have a clear and structured plan in order to successfully keep track of his many project dependencies and maximize his productivity.

Manager Persona
Name: Meticulous Manager Maria
User Story: Maria is the newly promoted manager of a team of 8 software engineers. Her job involves scheduling and leading meetings, strategically assigning tasks to her team members, tracking their progress in those tasks, setting performance goals that align with project objectives, and evaluating the overall progress in the project. She creates and adheres to strict timelines, with clear deadlines and daily meetings to check on progress. Skilled in managing client interactions and inter-team communication, she makes sure that expectations are clearly defined and consistently met. To facilitate smooth coordination and ensure efficient progress toward project completion, Maria must be able to assign tasks, check the progress of her team members, and keep track of dependencies in an organized manner.

Features

The feature set for each proposed dashboard was carefully evaluated based on the needs of the user personas. Each dashboard has a set of must-have and nice-to-have features. There are some features that are ubiquitous and relevant to all the dashboards. These include grouping, categorizing, and filtering tasks as well as assigning estimated time overheads per task. Additional features include deadline tracking and urgency indicators for impending deadlines. The ability to link resources, instructions, or other relevant information to tasks is also critical to the dashboard's core functionality. Finally, a nice-to-have feature for all dashboards would be importing GitHub issues and PRs to be presented as dashboard tasks.

Specifically for the student dashboard, some nice-to-have features would be a calendar auto populated with tasks in a week or month view. Additionally, a built-in search engine or chatbot to perform rubber duck programming would be a useful feature to develop the user's skills and aid them with their assignments.

For industry developers, the auto populated calendar would once again be a nice-to-have feature. Additionally, a visualization of the life cycle of the project currently being worked on would add value and give context to day-to-day work, especially if this was pulled from Jira directly. Another nice-to-have feature would be a section for quick access links to relevant communication tools. Developers would also benefit from being able to select tasks from their backlog to prioritize for the day and have a display indicator to ensure that developers are not overextending themselves on too many tasks.

For managers, some must-have features are keeping track of team members' performance goals and interactively placing checkpoints and recording relevant information. Another

must-have is being able to assign tasks to members through their dashboard. Finally, a section for sprint visualization and consistent metrics related to sprints would be relevant to ensure the manager keeps the team on track. Some of the nice-to-have features overlap with the industry developer dashboard including: quick access links and Jira integration. Some additional nice-to-have features are being able to see a specific team member's daily tasks and share project updates to externally dependent teams and clients. Finally, being able to track progress through the client lifecycle would be a relevant feature that would be nice-to-have.

Risks and Rabbit Holes

Each of the proposed dashboard projects has its own set of risks and rabbit holes that must be carefully considered and discussed. Beginning with the student dashboard, the major risks are associated with the features integrating GitHub and implementing a tutorial chatbot to help with rubber duck programming. These are areas where the team lacks experience, which could lead to high time overhead. Another feature, the implementation of a task list, relies on GitHub integration to pull issues and assign relevant information to task structures in the dashboard. The major decision that needs to be settled is if the time overhead to integrate GitHub is worth the cost to the project. As of now, this feature is nice-to-have, and after further research, a decision will be made on whether it should be added to base functionality. The dashboard can operate without it, however some user research could help clarify how impactful this feature could be.

For the industry developer dashboard, there is a similar feature for integrating GitHub that would, again, come with the time overhead. In addition, the team has identified Jira as another integration that could bring benefit to users. The APIs for both of these are unexplored and as such will take time to understand and utilize effectively. Another feature depends on Jira integration which is the calendar functionality for project life cycle visualization. Once again, the team must decide whether this nice-to-have feature is worth the time cost associated with it.

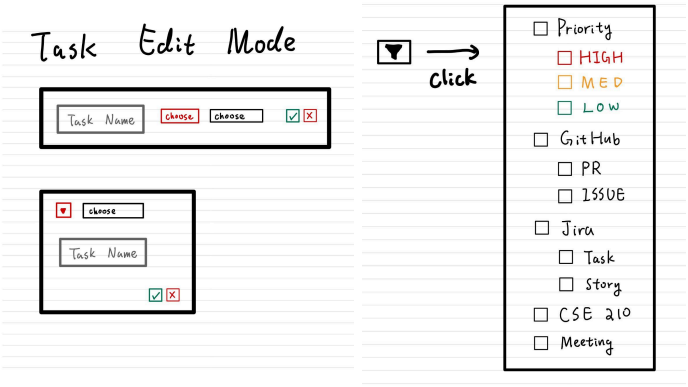
Finally, the manager dashboard has similar risks to the industry developer dashboard with GitHub and Jira integration. An additional risk is associated with the client progress tracking feature. Since the team does not have enough relevant experience with this feature, it will require time to understand how to implement and optimize this feature to successfully display relevant information from the client cycle. Additional research can be done to see how current manager dashboards implement this type of feature, and their shortcomings and strengths.

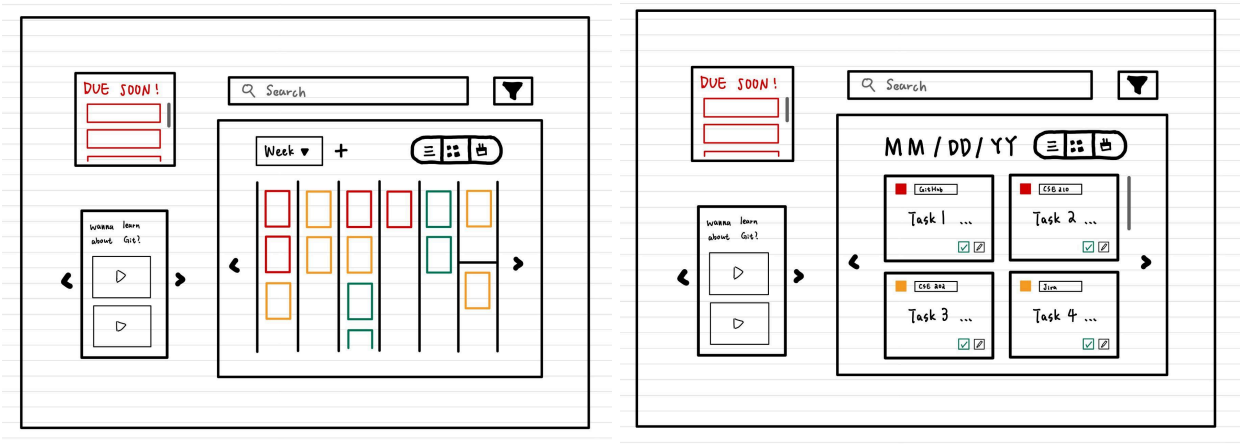
The iron triangle can be considered to understand how to best approach this project. The team has a diverse set of skills, which lends itself well to a project with many unique parts. Additionally, using pair programming, the team has worked to cross-train individuals on subject matters to ensure that at any time at least two members are up to date on a task's progress. Due to the time constraints of the class, the user for the dashboard will be selected from one of the three user personas mentioned above. This limitation in scope allows for a high-quality product that specifically fits the functionality and needs of this user group, and allows for a limited feature set that can be implemented in the time allotted. Finally, the time consideration is that there are approximately five weeks in the project life cycle. As such, decisions must be made quickly, and tasks must be identified and understood early to ensure proper delivery for the users selected.

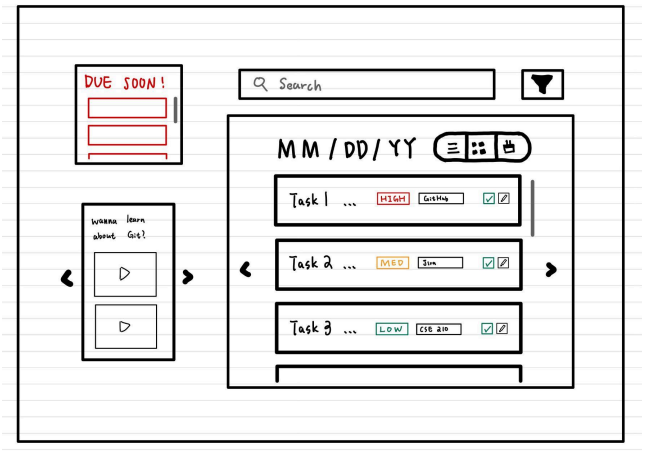
Project Sketches

Please see the GitHub repository for full scale versions of the wireframe sketches related to each dashboard (student, developer, and manager).

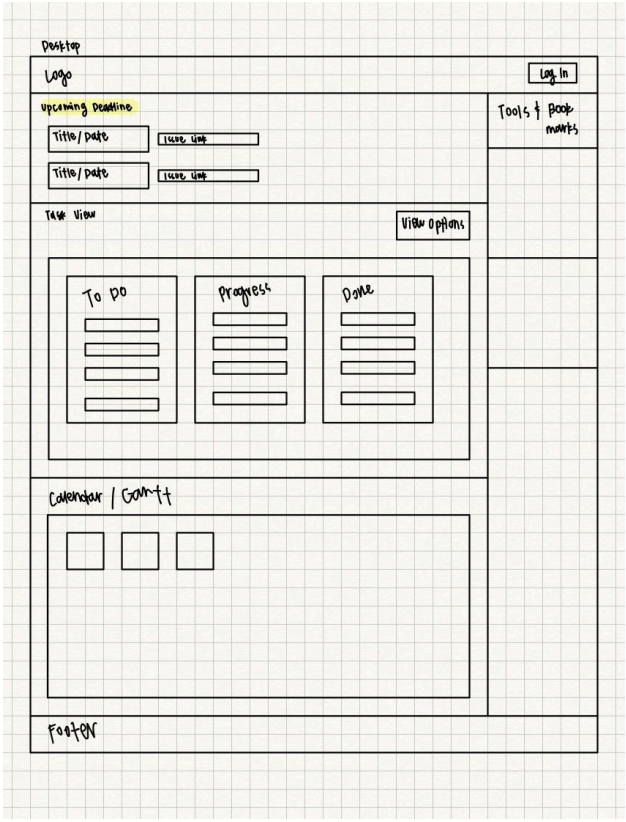
Student Wireframe

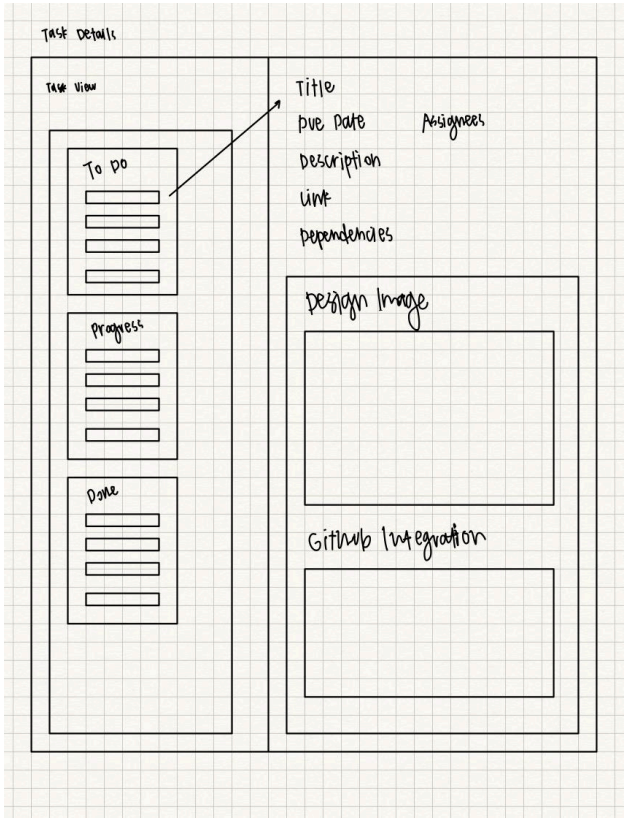




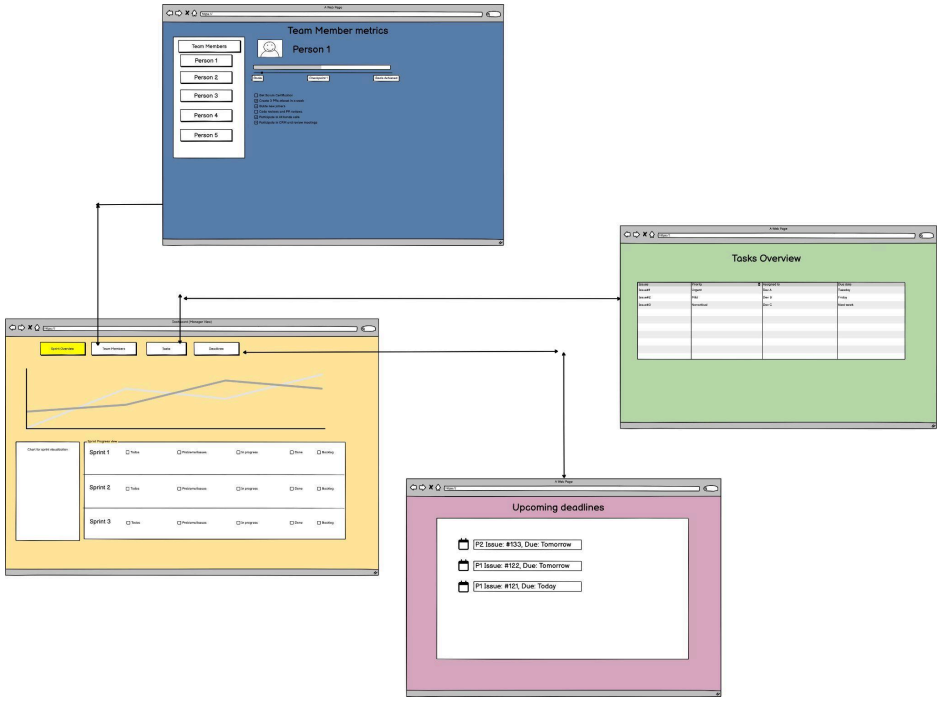


Developer Wireframe



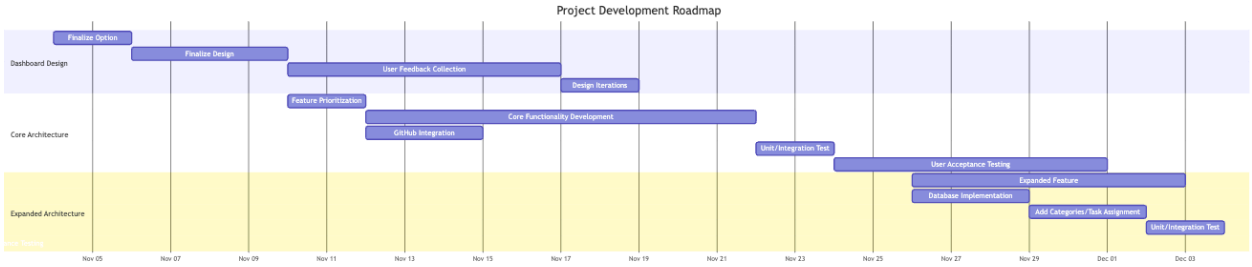


Manager Wireframe



Project Roadmap

Please see the GitHub repository for a full scale version of the roadmap.



5P Model

1. People First
    a. Availability - We plan to make this website accessible to the public with live hosting. To accommodate different devices, we aim to create a dynamic website layout that will adapt to the user's screen size,
    b. Performance - To maintain RAIL metrics, we aim to use plain vanilla JS throughout our development. We also plan to avoid platforms such as React which create unnecessary performance delays.
    c. Accessibility - For different devices, we will include various input modalities such that users will have access to all dashboard features.
    d. Usability - To confirm usability, we will include at least two people outside of our team and our TA, Cora to conduct reviews.
    e. Utility - We will have our user stories and design documents stored in our GitHub for approval by our TA, Cora.
2. Principle Driven
    a. We will use core vanilla client-side technology (HTML, CSS, JavaScript, HTTP, common media types, etc.) to implement our dashboard and ensure sustainability.
    b. We will have a minimalistic design, including a simple but intuitive user interface.
    c. Our app will be online; to take into account network instability, we will use a progressive web application, allowing our app to work similarly when offline.
    d. Our backend will either be written in Python/JavaScript (NodeJS). Databases we may use include server-side databases MySQL, MongoDB, or Postgres.
3. Problem Centric
    a. Design Process Required: We have included user personas, project sketches, wireframes, and roadmaps to show our design process.
    b. Dependencies Require Approval: At this time we do not have any dependencies.
4. Process Followed
    a. We are holding whole team in-person meetings and documenting them. We will continue this trend, since it seems to be the most effective way to make sure everyone understands the progress and direction of the project.
    b. We will hold multiple sprints, at least 3 (the one week design sprint and two 2 week work sprints). For the sprint, we will use issues from the backlog and hold and document retrospectives via Jira.
    c. We will use Github as the main repository to house all of our work and use issues and branching.
    d. We will use Github actions to run the build, test, document, and deploy our code. Our CI/CD pipeline will be easy to run.
    e. We will adhere to clean coding practices by writing consistent, minimalist code with in-line comments.
    f. We plan to conduct unit testing, e2e testing, and integration testing at least once a week starting sprint 2.
5. Pragmatic Solution
    a. We plan to start with a minimal number of features (the must-haves) and move on to the nice-to-haves only after ensuring that the must-haves are functional and well-tested. This way we can secure base functionality in our constrained timeframe.