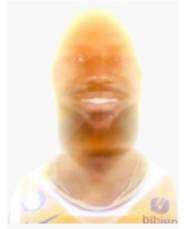


Team 8 - LeCoders

Project Pitch



User Persona:

- Ben is a busy developer who works on multiple projects simultaneously. He is often overwhelmed with his tasks and needs help keeping track of his progress and ideas. He needs a journal that allows him to quickly jot down notes, and access his old notes to resume his workflow.
- Carla is a curious developer who enjoys experimenting with new technologies and ideas that come across her head. Her main goal is to use a journal to be able to document her learning process and upload helpful visuals to digest the code she's working on.
- Tom is an intern at a big tech company and he is learning so much from his mentors but he's scared that he might forget all the insights he's receiving. He needs a way to track down all the information he's getting.
- Jerry is a student who's trying to learn how to program. As a beginner he does not follow proper coding procedures and just programs however he likes. During a project, he made many functions to the point where he does not remember what each function does or what each variable is for and stores. Having a journal that reminds him to document and storing it in a separate application will help him get used to the practice.
- Sherry is an experienced developer at a big tech company. Her boss gives vague projects for her to accomplish, but she has many ideas on implementation and how to design the overall program. She wants a space where she can place ideas and jot down sporadic brainstorming that is related to the files she is working on.
- Francis works at a tech company and his main role is to test codes. The project is nearing its launch date and he has been very busy testing every feature and is getting overwhelmed so he needs some type of documentation journal to track his fixes and findings.
- Walter works in cybersecurity so he is very cautious about security issues. He needs his journal entries to be protected and private.
- Jefferson is a developer at a startup and their team doesn't have many funds or resources to play around with. So they must be focused and be organized or else they might sink as a company. They need a way to streamline their work process and remember what tasks are needed and complete. Additionally in case a dev is onboarded if they secure investors, they need to be able to have them be caught up as quickly as possible.
- Mark is a freelance web developer who juggles multiple client projects simultaneously. With varying deadlines and project requirements, he often finds herself struggling to stay organized and track his progress. Additionally, Mark wants the ability to upload screenshots or visual aids to better communicate her ideas or document her progress on specific tasks. This way, he can efficiently manage his workload and ensure he meets his clients' expectations on time.

Brainstorming Process:

To combat the users problem we used sticky notes for solutions/features to problems. We met on Figma which helped us collaborate and visualize our thought process. We initially used Timers to get our possible problems as a Developer and requirements that we would have. As we finished the idea process, we would group the similar problems and create categories. We each offer features that would accommodate the user's needs and fix their problems.

Then we used a Tree diagram to refine our definition of the problem and break down the problems for personal issues. Then we jot down ideas for the five problem themes each contributing to what we think could be possible problems could be.

Continuity between Work (Problem):

In the field of software engineering, the issue of "Continuity between Work" presents significant challenges to developers' productivity and clarity of the project. Our team discussions have emphasized numerous concerns that contribute to this problem. In general, developers often struggle to recall the particulars of their coding activities, such as the content coded on a particular day or the purpose of certain code segments, especially when the naming conventions are unclear or comments are lacking. There is also difficulty in tracking which files have been edited recently. Moreover, remembering key insights during coding sessions, determining whether previously written code contains bugs, and recalling tasks from the previous week are further aspects of this challenge. These difficulties are worsened by inadequate methods for tracking ongoing and completed tasks, disrupting the flow of work and thought processes. Additionally, developers find it hard to document new discoveries and review their developmental progress due to insufficient documentation practices. This range of issues underscores the critical need for improved continuity in managing software development projects.

In regards to the group mapping, we outlined the key components of the "Continuity Between Work" issue within our software engineering processes. Central to this issue, we broke it up into two main themes, "Remembering what was done" and "Remembering what to do," which together frame the challenges of day-to-day coding assignments. Under the "Remembering what was done" section, the chart highlights the difficulty in keeping track of which files were edited—an issue compounded when dealing with multiple projects or an excessive number of files. It also notes the importance of recalling new concepts learned during the process, alongside receiving and integrating feedback, which is essential for continuous improvement but often overlooked. The second theme, "Remembering what to do," deals with organizing and prioritizing tasks such as addressing GitHub issues and fixing bugs encountered during development. Challenges here include remembering the specifics of bugs and the exact lines of code that caused them, continuing implementation without lapses, and the crucial but frequently neglected task of documenting changes and decisions made. Together, these themes illustrate a complex web of tasks that need systematic tracking and documentation to ensure smooth transitions from day to day and from one task to another, thereby enhancing overall workflow and productivity in our software development projects.

Group 1

Continuity between Work

Can't Remember what was coded for that day

Kevin Kuang

For long coding projects, developers might have forgotten what a specific part of their code does (i.e. due to improper naming // lack of comments)

Arthur

Forgot what files were edited

Kevin Kuang

Remembering Key Observations while Coding

Kaustubh Pathak

Can't remember whether the code you left off was buggy or not

Kevin Kuang

Forgot what they did last week.

Nicholas Nguyen

Difficulty in Remembering the work that has to be done

Kaustubh Pathak

Keeping track of their work

Utkar

Hard to keep track of the work they've done, work they need to do, and flow of work (thought process).

Joshua Chen

try to keep track of their progress and growth as a developer

Utkar

Want to document any new discoveries of technologies or tools that seems useful for the future.

Nicholas Nguyen

Documentation using certain tools

Eric Huang

Want to go back and look at their progress but there's no clear documentation on what they did.

Nicholas Nguyen

Documenting the code after the programming

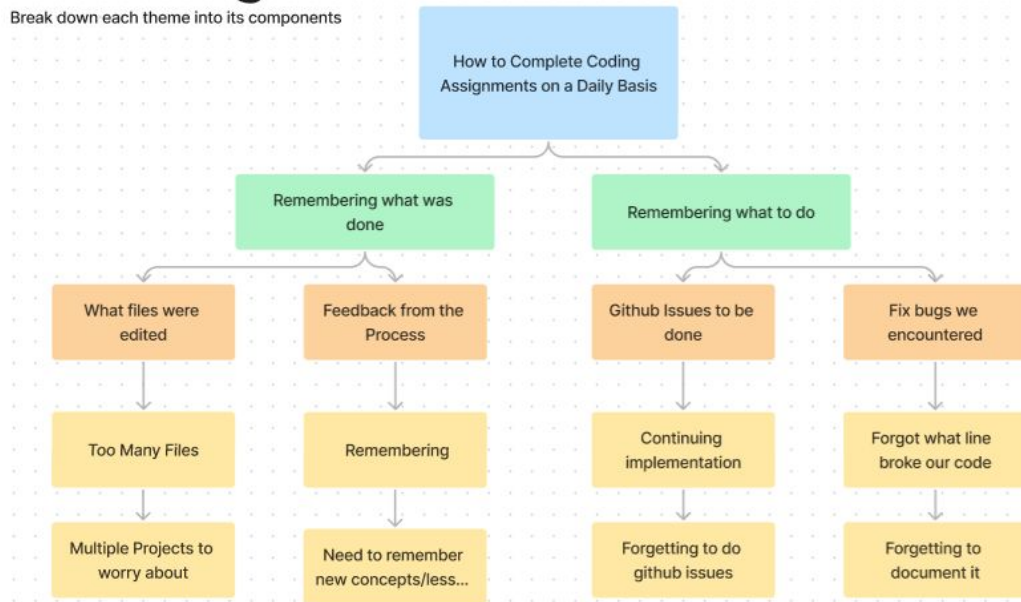
Sam

Between work hours, encourage members to learn other parts of the project that go outside of their role to increase overall awareness

Brian Lee

Continuity Between Work

Break down each theme into its components



Developing a Workflow (Problem):

As part of software engineering task, crafting an effective workflow is essential yet burdened with difficulties, as revealed in our recent team discussions. A common issue we discussed was the feeling of disconnection between the work completed the previous day and the tasks at hand today, which often leads to a disorganized work process. Collaborative efforts among developers are sometimes hindered by unclear communication or poorly defined workflows, complicating joint tasks and code integration. Developers frequently encounter obstacles that require discussion or assistance from colleagues, yet there is no efficient process for addressing these collaborative needs. Additionally, there is an absence of a clear task sequence, making it challenging for developers to organize their work into manageable subtasks. This lack of structure often affects day-to-day productivity tracking and causes code conflicts with peers. In addition, developers express a need for mechanisms to compare results and prioritize tasks effectively, indicating a broader need for a more refined and structured workflow to enhance efficiency and collaboration within the team.

The flowchart from our meeting concisely delineates the core issues encountered while developing a workflow within our software development projects, organized under two main themes: "Collaboration between Teammates" and "Lack of Workflow/Direction." Under "Collaboration between Teammates," we identified a series of challenges that significantly impede efficient teamwork. These include a lack of task delegation which leads to uneven work distribution, with some team members being overburdened while others are neglected. Additionally, the absence of effective communication and routine checks on platforms like Slack and GitHub disrupts the flow of information, often resulting in missed meetings and unaddressed issues in collaborative workspaces. Such gaps can lead to overlapping code efforts and inefficiencies. The "Lack of Workflow/Direction" theme encapsulates problems related to the strategic planning and execution of tasks. Notable issues include the absence of clear scheduling and planning, which results in waiting for other parts of the project to advance or complete. This is further complicated by a lack of clarity in task ownership and the chain of handover, making it difficult to break down large tasks into manageable subtasks and maintain a consistent routine. Together, these issues illustrate a need for more structured and communicative workflow management to ensure that all team members are synchronized and that tasks are executed efficiently, enhancing both individual and collective productivity.

Group 2

Developing a workflow

Feel a sense of disconnection between the day before and the work you are trying to do today

Kevin Kuang

Collaborating with other devs with either unclear communication or work flow

Eric Huang

Problems they need help with or need to discuss with other people

Kaushub Pathwal

Not having a clear flow of tasks/work to do

Eric Huang

Keep track of productivity for the day

Joshua Chen

Need a better way to break down work they need to do into sub tasks

ex.

X work for X Company
1. do this
2. do that

Joshua Chen

Keeping track of the work progress

Sam

Conflicting code with other developers

Arthur

Need way to prioritize certain tasks over another

Joshua Chen

Wanting to compare results with another developer.

Nicholas Nguyen

A link with the Github Issues of the repo

Kaushub Pathwal

Want to get the work done ASAP

Sam

Developing a workflow

Break down each theme into its components



Personal Issues/Conflicts/Documentation (Problem):

In today's fast-paced work environments, employees often find themselves questioning their alignment with the company's work style and culture, unsure of how to optimize their productivity and well-being amidst uncertainty. When employees lack a sense of belonging, they become disconnected from the team, experience a decline in productivity, and may feel apprehensive about sharing personal or work-related information with their colleagues. Furthermore, the potential existence of a toxic work environment worsens these issues, leading to decreased motivation, increased stress levels, and ultimately, reduced efficiency and effectiveness in tasks and projects. Therefore, it is crucial to document experiences that cause this as they serve as valuable insights for improving work environments and fostering a sense of belonging among employees.

Group 3

Personal Issues/ Conflicts/ Documentation

Unsure whether you enjoy
the company work style or
not

Kevin Kuang

Managing Personal and
Work Life

Kaustubh Patilwal

Accounting for Hours or
Overtime in Crunch Hours

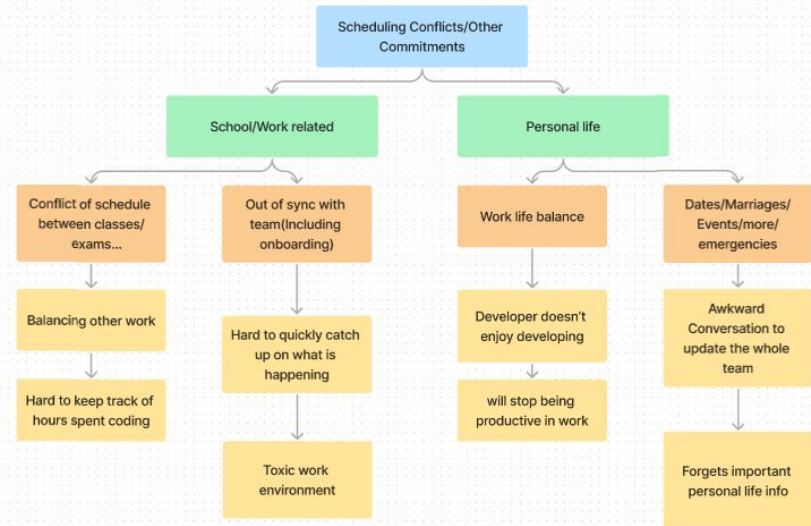
Kaustubh Patilwal

Miscommunication within
the team

Sam

Personal Issues/Conflicts/Documentation

Break down each theme into its components



Resources for Help (Problem):

Developers face many challenges daily because it's hard to find help when they need it. Learning new tech stuff can be tough, especially when company rules or agreements limit what they can access. Even if they can find resources, using new tools or frameworks can be confusing. Sometimes they forget important things, which makes fixing problems even harder. Also, they need to balance asking for help with trying to figure things out on their own. Getting help can take a lot of time and effort because they have to go through company processes or deal with complex technical issues. Plus, if they're introverted, they might be scared to ask for help because they worry about what others will think.

Remembering company rules or knowing who to ask for help can be tough too. And if they wait too long to ask for help, it might be too late, and no one's available. To solve these problems, we need to create better systems to help developers learn and work together. This means making it easier for them to find resources, ask for help, and feel comfortable doing so. With these changes, developers can tackle their work with more confidence and success.

Testing and Maintenance (Problem):

In the software development industry, developers invest extensive hours in debugging, testing, and reviewing code, encountering numerous challenges along the way. These challenges range from insufficient test cases and software glitches to a scarcity of information within code segments and the struggle to recall every team member's inputs and ideas. Addressing these challenges is crucial for maintaining project timelines, ensuring software quality, and fostering effective collaboration within development teams. However, without a systematic approach to documenting these challenges and their resolutions, valuable insights can be lost, leading to repeated mistakes, prolonged debugging cycles, and a decrease in overall productivity.

Group 5

Resources for Help

Not having clear documentation when using a technology to build a piece of software

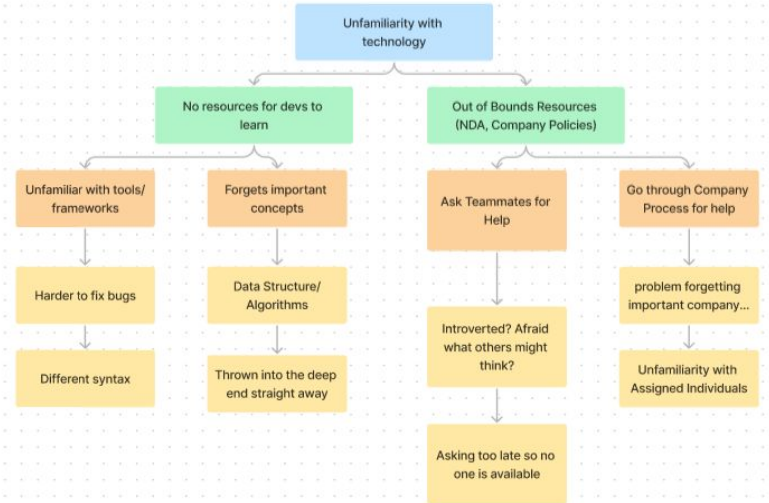
Arthur

help identify areas of improvement

Ulisses

Resources for Help

Break down each theme into its components



Group 4

Testing and Maintenance

Testing features

Need a way to understand code base (via diagramming and pictures)

Eric Huang

Joshua Chen

Maintenance checking and testing.

Maintaining their product and how to keep track of that.

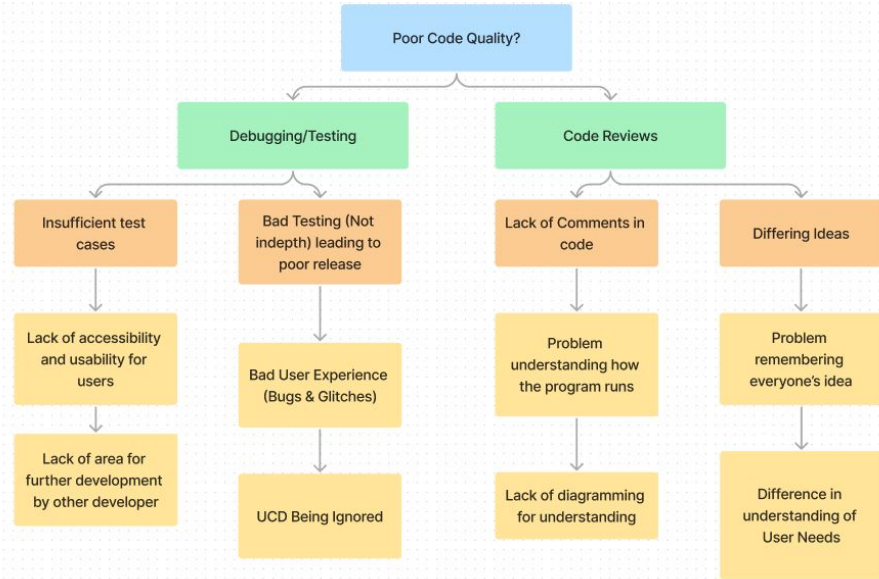
Nicholas Nguyen

Eric Huang

Program doesn't sometimes work

Sam

Testing and Maintenance



Problem Statement:

In the realm of software engineering, the challenge of "Continuity between Work" presents significant hurdles for developers, as they grapple with recalling coding specifics, tracking edits, and remembering key insights, while facing disruptions in workflow due to inadequate task tracking, poor documentation practices, and unclear communication, exacerbating issues in collaboration, productivity, and employee well-being, alongside challenges encountered in testing and maintenance, such as insufficient test cases and lack of information within code segments, underscoring the urgent need for improved continuity management and systematic documentation to ensure success in software development projects.

Appetite:

We are given around 5-6 weeks for the project, so we would want to focus on the primary features that can be implemented as local first. Learning from the warmup exercise, we may allocate 2 weeks or 2.5 weeks max to implement the local first features first and ensure that the calendar, task list, journal summary, code log, documentation, meeting notes, and code feedback will save on the cache and ideally work while there is no internet connection. These are the features that we want to be the most polished by the end of our project. For the rest of the 2-3 weeks we will be adding any additional features that were mentioned like templates, online cloud sync, and making the application more visually appealing. If we were only given 2 weeks we will be focusing on the necessities of a developer's journal like a task list and the journal entry and having them work offline. We are giving ourselves a decent amount of time to implement some of these features because some of us are not as experienced in JavaScript and it will take some time to debug JavaScript and sometimes CSS visual effects.

Solution:

We had many ideas for features that we can include into the developer's journal ranging from adding labels for what files and documents were modified for that day to incorporating GitHub's API to get a list of all code changes in files. These will be things like having notes, comments, text area for a summary of the day, code blocks to add what was edited, and labels to work well without the internet. These solutions will cover the issue of continuity of work. Calendar, task list will help with developing a workflow and parts of conflict resolution. This will require us to use local storage/cache and possibly convert the data to a JSON file or other file formats to be downloaded on the computer. Another feature we may want to add is templates of meeting minutes or flowchart templates to guide developers on how to break down their large tasks into smaller more manageable pieces. Once we have the local features down, we will see if we have remaining time to have the data sync up to the cloud with a backend and set up a system where users can login. Once we start implementing the templates and guide maps, we should be "done" after those features, but if we have less time than expected, we need to at least have code blocks and labels implemented.

Brainstorming Process:

To combat the users problem we used sticky notes for solutions/features to problems. We met on Figma which helped us collaborate and visualize our thought process. As we finished the idea process, we would group the solutions into categories that would address the respective problem. We each offer features that would accommodate the user's needs and fix their problems and vote on the “better” features.

Then we used a Tree diagram to refine our solution and break down the solutions for personal issues into four solutions that would help mitigate these problems for our users. Then we jot down ideas for the four solution themes each contributing to what we think could be possible features that address the overarching problem.

Continuity between Work (Solution):

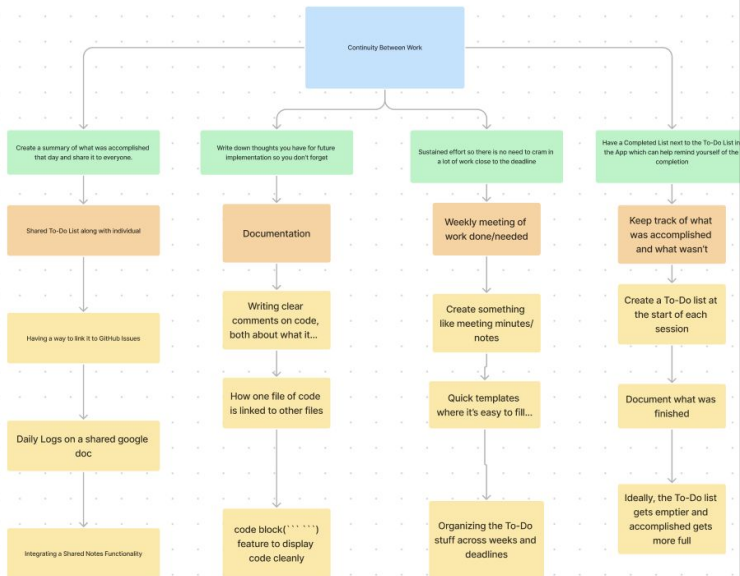
To maintain continuity between work, the user would need a space to input a summary of the work they accomplished for today, work they need to work on in the future (can be related to the Task List), and any other thoughts they have. There should be a space to input code to talk about, whether it's comments or documentation. There could also be a shared documents with other developers where they could input important meeting minutes, shared deadlines, and tasks that rely upon each other.

Developing a workflow(Solution):

To first develop workflow, our user would need some sort of Task-List to help split up the work, into smaller and more digestible sub-tasks, with an end goal in mind. These could be shared with other developers, so that they all have a similar understanding of workflow. There could be different labels or colors to separate topics, tasks, and responsibilities that the user needs to keep in mind. Another way this label could be utilized is so that it's colored based on team member's responsibility. A flow-chart and other images can be uploaded to the journal, allowing the user to have an accessible visual to grasp the scope of the project they're working on.

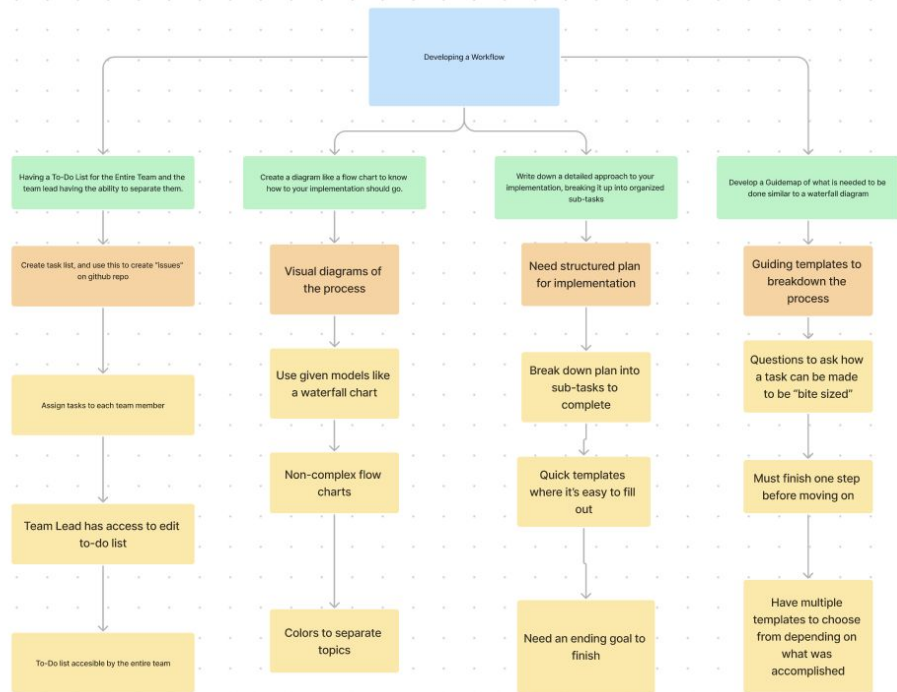
Continuity Between Work

Break down each theme into its components



Developing a Workflow

Break down each theme into its components

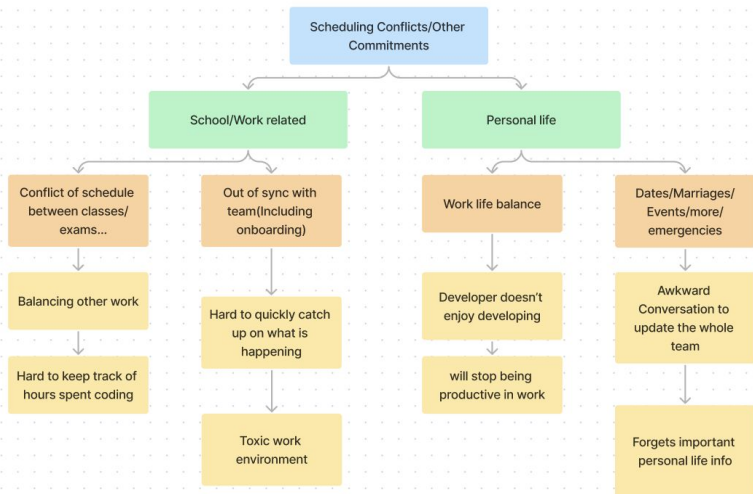


Personal Issues/Conflicts/Documentation(Solution):

A feature such as linking their calendars with their work schedules. This allows for a clearer view of their priorities and work obligations(tasks), resulting in fewer conflicts between work and personal life. And overall better time management. Also, we encourage users to proactively communicate with their team/leads by having the option to create a list of who to contact for their respective issues (scheduling conflicts). By communicating these issues early on one can avoid schedule conflicts. Ensuring that their personal life and work lives maintain a good balance. Additionally, finding available time slots within one's schedule to shift tasks around can make the user more productive. Finding their best time to work would encourage a productive work environment and a better sense of control over their time. Having an option to adjust these tasks would resolve the user's conflicts. To conclude, we wanted our users to have a way to jot down their thoughts or rants about the company. An option for a healthy outlet can help users understand their emotions and provide insight and a better understanding of areas of improvement or issues with the company this can be a section off as "Miscellaneous thoughts".

Personal Issues/Conflicts/Documentation

Break down each theme into its components



Personal Issues/Conflicts/Documentation Group 3

Communicate to your team/leads if there are any schedule conflicts and try to get any work done beforehand



Link your Personal Calendar, which can help see the things you have or need to do at home



Finding available timeslots within schedule to shift tasks around



Write down thoughts/complaints/rants about company to help visualize your thinking



Never be afraid to address your issues in the meeting for further discussion



Sam

Communicate conflicting issues with personal life and work life



Ulises

Have a .md file open on the side while you work.

Nicholas Nguyen

Flexible and able to like have a backup plan in case conflicts arise

Brandon Lou

Testing and Maintenance (Solution):

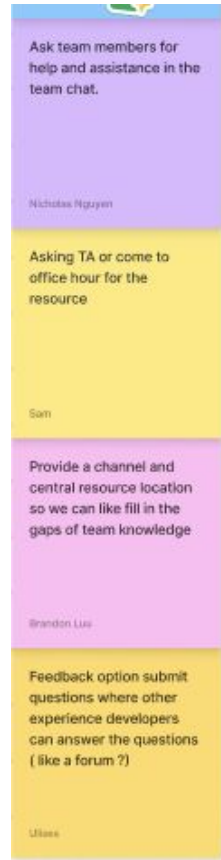
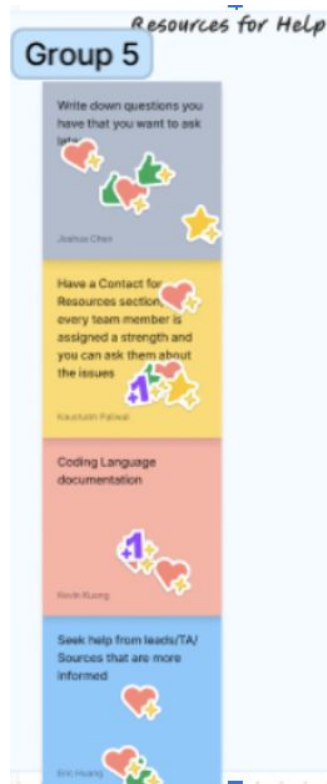
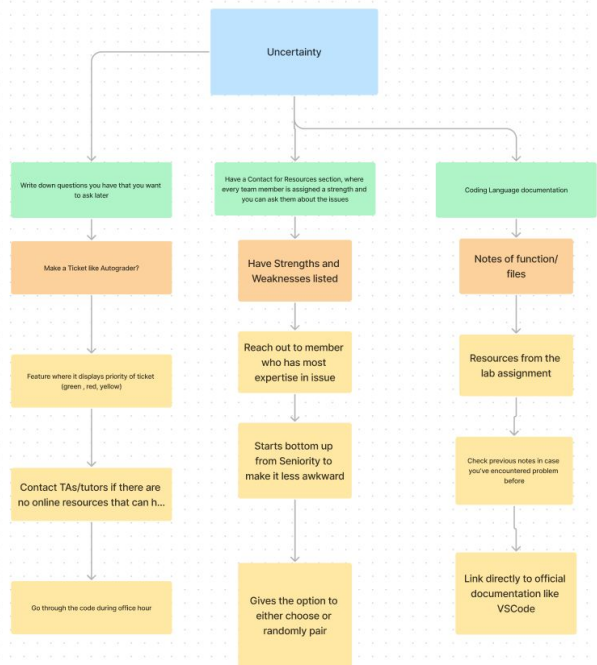
For example for testing and feedback we will have a mechanism that allows for direct input from users and transforming it into tasks and be able to give feedback in the form of notes. Additionally, for “needing a way to un-complete tasks if the test fails we have a solution of having a history of previously created tasks/completed. We visualized the feature as a “Code Feedback” section where we can user feedback and bugs problems with a checklist signifying if the user completed or not. Lastly, a feature that can be used is a code log that helps record and have documentation attached of changes made to the file this can help mitigate the problem. This would enhance code quality and team collaboration through a centralized logging and tracking system for code changes, issues, and team insights. By integrating user feedback into the code log, the developers can see how their changes affect the user experience.

Resources for Help (Solution):

The solution proposed for the Developer's Journal in Resources aims to enhance collaboration and problem-solving processes in software development projects. It suggests breaking down each theme into components, jotting down questions for later, implementing a ticket-like autograder feature, establishing a "Contact for Resources" section, providing coding language documentation, listing strengths and weaknesses, noting functions/files, reaching out to members with expertise, curating resources from lab assignments, providing contact details for additional support, fostering a bottom-up approach for seeking assistance, checking previous notes, scheduling office hours for collective code review, offering options for choosing or pairing for assistance, and linking directly to official documentation like VSCode. These features are designed to help developers address challenges more effectively and streamline their workflow.

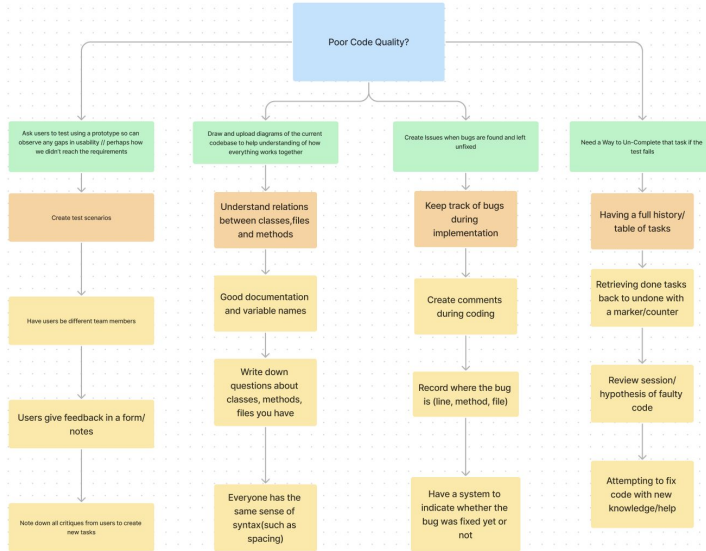
Resources

Break down each theme into its components



Testing and Maintainance

Break down each theme into its components



Group 4

Testing and Maintainance

Ask users to test using a prototype so can observe any gaps in usability // perhaps how we didn't reach the requirements



Arthur

Draw and upload diagrams of the current codebase to help understanding of how everything works together



Joshua Chan

Create Issues when bugs are found and left unfixed



Eric Huang

Need a Way to Un-Complete that task if the test fails



Kaustubh Paliwal

Developing multiple test files to ensure the program works correctly



Sam

Keeping track of what tests is testing what specific feature



Kuang

Create tests as you develop your code and test it every time you add something new.














Nicholas Nguyen

Developing more tests to ensure program works

Ulises

Initial Sketches to address solutions

TASK LIST

TO-DO	Sort ↓	COMPLETED
<input checked="" type="checkbox"/> 		<input checked="" type="checkbox"/> 
<input type="checkbox"/> 		<input type="checkbox"/> 
<input type="checkbox"/> 		<input type="checkbox"/> 
<input type="checkbox"/> 		<input type="checkbox"/> 
<input type="checkbox"/> 		<input type="checkbox"/> 
<input type="checkbox"/> 		<input type="checkbox"/> 
<input type="checkbox"/> 		<input type="checkbox"/> 

Annotations:

- Description of task & due date (points to TO-DO column)
- Main task (points to red bar)
- Subtasks (points to yellow and green bars)
- Add to github as issue (points to bottom row)

CODE LOG

File name	Change(s) made / updates to file
~	~
~	~
~	~
~	~
~	~
~	~
~	~
~	~
~	~
~	~

DOCUMENTATION

File name	Variable / function name	Description of what the var // function does / approach to do
~	~	~
~	~	~
~	~	~
~	~	~
~	~	~
~	~	~
~	~	~
~	~	~
~	~	~
~	~	~

TEAM CALENDAR

S	M	T	W	T	F	S
			- Team meeting @ 8PM			

Urgent Due (points to red bar on the left)

<NAME>'S CALENDAR

S	M	T	W	T	F	S

Clickable days to show availability that day (points to a day in the calendar)

MISC. THOUGHTS

Dev's name	Text of thoughts	Thought has been addressed?
~	~	<input checked="" type="checkbox"/>
~	~	<input type="checkbox"/>
~	~	<input type="checkbox"/>
~	~	<input type="checkbox"/>
~	~	<input type="checkbox"/>
~	~	<input type="checkbox"/>
~	~	<input type="checkbox"/>
~	~	<input type="checkbox"/>
~	~	<input type="checkbox"/>
~	~	<input type="checkbox"/>

MEETING NOTES

Date	Expandable content
~	~
~	~
~	~
~	~
~	~
~	~
~	~
~	~
~	~
~	~

Add new minutes using template (points to a box in the bottom row)

CODE FEEDBACK

User Feedback	Bugs & Problems
~	~
~	~
~	~
~	~
~	~
~	~
~	~
~	~
~	~
~	~

Fixed or not? (points to a checkbox in the bottom row)

Wireframes

The image displays eight wireframe frames for a productivity application, each with a unique title and layout.

- Frame 1: TASK LIST**
 - Header: TASK LIST
 - Left Column: TO-DO (Sort ↓)
 - <Insert Task> (Red button)
 - <Insert Subtask> (Green button)
 - <Color-coded urgency/priority> (Yellow button)
 - Right Column: COMPLETED
 - Green bars representing completed tasks.
- Frame 2: CODE LOG**
 - Header: CODE LOG
 - Form: <Insert File Name> (Date: May 5 14:00:59 PST)
 - * <Insert Changes Made>
 - Form: <Insert Date & Time>
 - * <Insert Changes Made>
 - Form: <Insert File Name> (Date: <Insert Date & Time>)
 - * <Insert Changes Made>
- Frame 3: DOCUMENTATION**
 - Header: DOCUMENTATION
 - Form: <Insert File Name> (Dropdown: var1)
 - <Insert description of what var1 is>
 - Form: <Insert File Name> (Dropdown: var2)
 - <Insert description of what var2 is>
 - Form: <Insert File Name> (Dropdown: foo)
 - <Insert description of what foo does>
 - Form: <Insert File Name> (Dropdown)
 - Form: <Insert File Name> (Dropdown)
- Frame 4: TEAM CALENDAR**
 - Header: TEAM CALENDAR
 - Table: 7 columns (S, M, T, W, T, F, S)
 - Row 1: Empty
 - Row 2: Empty
 - Row 3: Empty
 - Row 4: Empty
 - Row 5: Empty
 - Row 6: Empty
 - Row 7: Empty
 - Row 8: Empty
 - Row 9: Empty
 - Row 10: Empty
 - Row 11: Empty
 - Row 12: Empty
 - Row 13: Empty
 - Row 14: Empty
 - Row 15: Empty
 - Row 16: Empty
 - Row 17: Empty
 - Row 18: Empty
 - Row 19: Empty
 - Row 20: Empty
 - Row 21: Empty
 - Row 22: Empty
 - Row 23: Empty
 - Row 24: Empty
 - Row 25: Empty
 - Row 26: Empty
 - Row 27: Empty
 - Row 28: Empty
 - Row 29: Empty
 - Row 30: Empty
 - Row 31: Empty
 - Row 32: Empty
 - Row 33: Empty
 - Row 34: Empty
 - Row 35: Empty
 - Row 36: Empty
 - Row 37: Empty
 - Row 38: Empty
 - Row 39: Empty
 - Row 40: Empty
 - Row 41: Empty
 - Row 42: Empty
 - Row 43: Empty
 - Row 44: Empty
 - Row 45: Empty
 - Row 46: Empty
 - Row 47: Empty
 - Row 48: Empty
 - Row 49: Empty
 - Row 50: Empty
 - Row 51: Empty
 - Row 52: Empty
 - Row 53: Empty
 - Row 54: Empty
 - Row 55: Empty
 - Row 56: Empty
 - Row 57: Empty
 - Row 58: Empty
 - Row 59: Empty
 - Row 60: Empty
 - Row 61: Empty
 - Row 62: Empty
 - Row 63: Empty
 - Row 64: Empty
 - Row 65: Empty
 - Row 66: Empty
 - Row 67: Empty
 - Row 68: Empty
 - Row 69: Empty
 - Row 70: Empty
 - Row 71: Empty
 - Row 72: Empty
 - Row 73: Empty
 - Row 74: Empty
 - Row 75: Empty
 - Row 76: Empty
 - Row 77: Empty
 - Row 78: Empty
 - Row 79: Empty
 - Row 80: Empty
 - Row 81: Empty
 - Row 82: Empty
 - Row 83: Empty
 - Row 84: Empty
 - Row 85: Empty
 - Row 86: Empty
 - Row 87: Empty
 - Row 88: Empty
 - Row 89: Empty
 - Row 90: Empty
 - Row 91: Empty
 - Row 92: Empty
 - Row 93: Empty
 - Row 94: Empty
 - Row 95: Empty
 - Row 96: Empty
 - Row 97: Empty
 - Row 98: Empty
 - Row 99: Empty
 - Row 100: Empty
- Frame 5: <NAME>'s CALENDAR**
 - Header: <NAME>'s CALENDAR
 - Table: 7 columns (S, M, T, W, T, F, S)
 - Row 1: Empty
 - Row 2: Empty
 - Row 3: Empty
 - Row 4: Empty
 - Row 5: Empty
 - Row 6: Empty
 - Row 7: Empty
 - Row 8: Empty
 - Row 9: Empty
 - Row 10: Empty
 - Row 11: Empty
 - Row 12: Empty
 - Row 13: Empty
 - Row 14: Empty
 - Row 15: Empty
 - Row 16: Empty
 - Row 17: Empty
 - Row 18: Empty
 - Row 19: Empty
 - Row 20: Empty
 - Row 21: Empty
 - Row 22: Empty
 - Row 23: Empty
 - Row 24: Empty
 - Row 25: Empty
 - Row 26: Empty
 - Row 27: Empty
 - Row 28: Empty
 - Row 29: Empty
 - Row 30: Empty
 - Row 31: Empty
 - Row 32: Empty
 - Row 33: Empty
 - Row 34: Empty
 - Row 35: Empty
 - Row 36: Empty
 - Row 37: Empty
 - Row 38: Empty
 - Row 39: Empty
 - Row 40: Empty
 - Row 41: Empty
 - Row 42: Empty
 - Row 43: Empty
 - Row 44: Empty
 - Row 45: Empty
 - Row 46: Empty
 - Row 47: Empty
 - Row 48: Empty
 - Row 49: Empty
 - Row 50: Empty
 - Row 51: Empty
 - Row 52: Empty
 - Row 53: Empty
 - Row 54: Empty
 - Row 55: Empty
 - Row 56: Empty
 - Row 57: Empty
 - Row 58: Empty
 - Row 59: Empty
 - Row 60: Empty
 - Row 61: Empty
 - Row 62: Empty
 - Row 63: Empty
 - Row 64: Empty
 - Row 65: Empty
 - Row 66: Empty
 - Row 67: Empty
 - Row 68: Empty
 - Row 69: Empty
 - Row 70: Empty
 - Row 71: Empty
 - Row 72: Empty
 - Row 73: Empty
 - Row 74: Empty
 - Row 75: Empty
 - Row 76: Empty
 - Row 77: Empty
 - Row 78: Empty
 - Row 79: Empty
 - Row 80: Empty
 - Row 81: Empty
 - Row 82: Empty
 - Row 83: Empty
 - Row 84: Empty
 - Row 85: Empty
 - Row 86: Empty
 - Row 87: Empty
 - Row 88: Empty
 - Row 89: Empty
 - Row 90: Empty
 - Row 91: Empty
 - Row 92: Empty
 - Row 93: Empty
 - Row 94: Empty
 - Row 95: Empty
 - Row 96: Empty
 - Row 97: Empty
 - Row 98: Empty
 - Row 99: Empty
 - Row 100: Empty
- Frame 6: MISC. THOUGHTS**
 - Header: MISC. THOUGHTS
 - Form: <Insert Dev Name>
 - * <Insert Random Thought/Suggestion>
 - * <Insert Random Thought/Suggestion>
 - Form: <Insert Dev Name>
 - * <Insert Random Thought/Suggestion>
 - * <Insert Random Thought/Suggestion>
 - Form: <Insert Dev Name>
 - * <Insert Random Thought/Suggestion>
 - * <Insert Random Thought/Suggestion>
- Frame 7: MEETING NOTES**
 - Header: MEETING NOTES
 - Form: <Insert Date of Meeting> (Sort ↓)
 - <Insert meeting notes, according to (markdown) template>
 - Form: <Insert Date of Meeting>
 - <Insert meeting notes, according to (markdown) template>
 - Form: <Insert Date of Meeting>
 - <Insert meeting notes, according to (markdown) template>
 - Form: <Insert Date of Meeting>
 - <Insert meeting notes, according to (markdown) template>
 - Form: + <Create new Meeting Notes according to MD template>
- Frame 8: CODE FEEDBACK**
 - Header: CODE FEEDBACK
 - Form: User feedback
 - <Insert feedback text>
 - Form: Bugs / Issues
 - <Insert bug report text>

Rabbit Holes:

Our solution to the problem is a common one that has many implementations. There are a handful of companies that specialize in space that our solution aims to target such as Slack, notion, discord, GitHub, etc. One thing to note is how we can become off-track by focusing on small aspects of our solution that lead us to a 'rabbit-hole'.

1. We focus too much on trying to integrate with other apps (calendars, git, reminders, etc.)
 - a. Version control with git pull, pushes, merge
2. We focus too much on the prospect of real-time collaboration and sharing (sharing journals with other devs) rather than the individual developers themselves
3. Focusing too much on animation, such as badges or rewards for journaling streaks can motivate users, excessively focusing on gamification mechanics. It may overshadow the primary goal of self-reflection and productivity improvement.
4. Spending too much time on an intricate organization system and synchronization
5. Trying to create data visualizations (insight and analytics on developer's journal entry)
6. Trying to implement AI suggestions into our task list to help suggest prompts or autocomplete tasks

Competitors: Github, Notion, Slack

If we prioritize version control, our solution might resemble GitHub. If we emphasize animation and complex organization, it could resemble Notion. Similarly, if we overly prioritize integration with apps like GitHub, our solution might resemble Slack. Compared to us, we might prioritize offline accessibility and a11y. The scale will definitely be smaller due to team size and time restriction.

GitHub

Unique Value Proposition

What makes this company unique?

Its Source Control feature to compare and run tests

Flavio José Padua

Coding that is shared and edited by multiple people

Karen Huang

Very integrated with VSCode, a popular IDE for devs

Arindya Choudhury

Interactive between User and Developer

David

Main source of communication with the software Git

Brandon Lee

Version control

Chen

Can create a pipeline that automatically deploys

Edo Huang

Ability to upload code and share it with other developers.

Michael Nguyen

Built by devs for devs

Arthur

Similar Capabilities

What are things that we want to do?

Have organization tools similar (also for create hierarchy - meeting pages) to Notion

Chen

Being able to build an interactive app

David

Being able to write text/ journals/summary of the day (Notion)

Karen Huang

Great UX like Notion

Flavio José Padua

Intuitive UI that isn't too complicated (Notion, Slack)

Arindya Choudhury

Minimal and simple ease of use

Brandon Lee

Able to upload code in an organized/ neat way (GitHub)

Michael Nguyen

Something like Confluence which can act as a Resource?

Flavio José Padua

User friendly - Not too overly complicated.

Michael Nguyen

Apparent Differences

What are the differences between them and us?

Focus on a more professional focused Audience unlike Notion

Flavio José Padua

Focusing on local first, so online app/resources are nice but will not always be available (GitHub/Stack)

Karen Huang

More focused on notes and documentation instead of code (GitHub)

Arindya Choudhury

The audience is for dev versus not devs

Brandon Lee

Integration capabilities like Notion

Chen

More focused on communicating rather than documentation (Stack)

Michael Nguyen

Much smaller user base, which is ultimately us, so we can tailor our functions and UI to whatever fits us best

Arthur

Opportunities

Where can we progress or create value?

While targeting the Developer, also focus on students and smaller groups, and make something we can use as well

Flavio José Padua

Catering these tools towards a developer and having meeting features on same page/ really easy to access (keeping related things close together)

Karen Huang

Find value in creating a product that we can end up using daily in the future

Arindya Choudhury

Make it accessible for old people as well

Chen

Sort each implementation by 2 things: which one is most important and is it code-wise within our time frame.

Michael Nguyen

Create it with longevity in mind

Edo Huang

Key Learnings

What can we learn from this process?

Understand the strength of each product offering and what we can learn from them

Flavio José Padua

Trying to understand what is achievable within the time we have left in class

Karen Huang

How subteams can communicate and collaborate

Chen

Draw from what's doing well in other companies and try to apply that to our product

Arindya Choudhury

Do what is capable with our current abilities, don't make things overly complicated

Michael Nguyen

How ideas and strategy can be generated within the group

Chen

Minimalist approach - the app should have exactly what it needs, but not more than that

Arthur

Less is more

Chen

product research and developing a simple theme

Brandon Lee

No Go's

For our project a general no go includes the general idea of “cowboy coding” where someone or a part of the team goes off with no plan and develops something that could be detrimental to the rest of the team. It's fine to explore new ways to implement a feature but one must take precautions to this. A general good thing to do is update Github Issues, regularly update the team on your work, and plan out your idea with tools such as figma. This is not an exhaustive list but neglecting to do something like this will potentially hurt our time, team, and flow of the project.

Another part of this is developing for longevity and accessibility. Having features is nice but what about the necessary requirements? We need to have features that are accessible and are completely hidden when not accessible. We don't want broken features because it reflects badly on us as developers. Keeping in mind our restrictions such as time and resources, we need to be coordinated. Tying back to the idea of going solo might be good for discovery but it also considers the risk of losing time and a possible disconnect from the team. Other things to keep in mind are lack of testing, bad documentation, and not following UCD.

Perspective is an important aspect because us as the developers might be consumed by what we want rather than what a user needs. Similar to how Gordon Ramsay thinks, the customer(user) is the most important person. In the case that we had a budget, it costs money and time to have features they didn't ask for. Delivering the base functionality should be the main goal. However this isn't always a direct path as it's a mix of progressive enhancement and graceful degradation. There are aspects such as a calendar that is reliant on JS functionality and other things that might be bottom up light documentation.