Agenda
2ND MAY 2024

What we're doing
Today, we're going to brainstorm ideas

Steps
1. Ideating the Solution
2. Creating Solution Trees
3. Competitor Analysis

What's next
After today's brainstorm, we'll...

How to SOLVE the Problems our User (a Developer) faces?

Ideas here

Ideate & Group
Brainstorm ideas around an area of concern. Then cluster them into groups.

Prompt

Themes
Copy the stickies into the groups based on themes that emerged.

Group 1 – Continuity between Work
- Can't Remember what was coded for that day
- 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)
- Forgot what files were edited
- Remembering Key Observations while Coding
- Can't remember whether the code you left off was buggy or not
- Forgot what they did last week.
- Difficulty in Remembering the work that has to be done
- Keeping track of their work
- Hard to keep track of the work they've done, work they need to do, and flow of work (thought process).
- try to keep track of their progress and growth as a developer
- Want to document any new discoveries of technologies or tools that seems useful for the future.
- Documentation using certain tools
- Want to go back and look at their progress but theres no clear documentation on what they did.
- Documenting the code after the programming
- Between work hours, encourage members to learn other parts of the project that go outside of their role to increase overall awareness

Group 2 – Developing a workflow
- Feel a sense of disconnection between the day before and the work you are trying to do today
- Collaborating with other devs with either unclear communication or work flow
- Problems they need help with or need to discuss with other people
- Not having a clear flow of tasks/work to do
- Keep track of productivity for the day
- 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
- Keeping tack of the work progress
- Conflicting code with other developers
- Need way to prioritize certain tasks over another
- Wanting to compare results with another developer.
- A link with the Github issues of the repo
- Want to get the work done ASAP

Group 3 – Personal Issues/ Conflicts/ Documentaion
- Unsure whether you enjoy the company work style or not
- Managing Personal and Work Life
- Accounting for Hours or Overtime in Crunch Hours
- If code is unclear, make sure to include comments and reasoning on what you changed
- Miscommunication within the team

Group 4 – Testing and Maintainance
- Testing features
- Need a way to understand code base (via diagramming and pictures)
- Maintenance checking and testing.
- Maintaining their product and how to keep track of that.
- Program doesn't sometimes work
- Have a system for others to read your code and review it for quality control

Group 5 – Resources for Help
- Not having clear documentation when using a technology to build a piece of software
- help identify areas of improvement

Themes
Copy the stickies into the groups based on themes that emerged.

Group 1 – Continuity between Work
- Create a summary of what was accomplished that day and share it to everyone.
- Write down thoughts you have for future implementation so you don't forget
- Sustained effort so there is no need to cram in a lot of work close to the deadline
- Have a Completed List next to the To-Do List in the App which can remind yourself of the completion
- Creating labels to indicate which files that were edited that day
- Creating a texarea/journal to talk about a summary of the work that was done for the day.
- Clear documentation e.g comments in code for implemented features so others can work on it
- Remind yourself of the things you intended to do by the end of the day
- Have a feature where the developer can add priority to tasks so developer can finish tasks accordingly
- Frequent meetings so it isn't too long between work sessions
- Clear comments on code about what each section of code does and how it functions

Group 2 – Developing a workflow
- Create a diagram like a flow chart to know how to your implementation should go.
- Exhaustively detail all issues/tasks for each subteam until the project is "done"
- Develop a Guidemap of what is needed to be done similar to a waterfall diagram
- Write down a detailed approach to your implementation, breaking it up into organized tasks
- Having a To-Do List for the Entire Team and the team lead having the ability to separate them.
- Split the work process into a "waterfall" like diagram to show what is done and what needs to be finished.
- Documenting every program you have coded
- Different stages of complete with Under Review, In Testing, Published, Half way done
- Have a flow map to organize tasks
- Streamlined comms between different parts of the pipeline

Group 3 – Personal Issues/ Conflicts/ Documentaion
- 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
- Communicate conflicting issues with personal life and work life
- Have a .md file open on the side while you work.
- Flexible and able to like have a backup plan in case conflicts arise

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
- Draw and upload diagrams of the current codebase to help understanding of how everything works together
- Create issues when bugs are found and left unfixed
- Need a Way to Un-Complete that task if the test fails
- Developing multiple test files to ensure the program works correctly
- Keeping track of what tests is testing what specific feature
- Create tests as you develop your code and test it every time you add something new.
- Developing more tests to ensure program works

Group 5 – Resources for Help
- Write down questions you have that you want to ask later
- Have a Contact for Resources section, every team member is assigned a strength and you can ask them about the issues
- Coding Language documentation
- Seek help from leads/TA/ Sources that are more informed
- Ask team members for help and assistance in the team chat.
- Asking TA or come to office hour for the resource
- Provide a channel and central resource location so we can fill in the gaps of team knowledge
- Feedback option submit questions where other experience developers can answer the questions (like a forum ?)

Group mapping
Break down each theme into its components

- Top level theme
  - Second R theme
    - Third level theme → Individual idea → Individual idea
    - Third level theme → Individual idea → Individual idea
  - Second level theme
    - Third level theme → Individual idea → Individual idea
    - Third level theme → Individual idea → Individual idea

Continuity Between Work
Break down each theme into its components

- How to Complete Coding Assignments on a Daily Basis
  - Remembering what was done
    - What files were edited → Too Many Files → Multiple Projects to worry about
    - Feedback from the Process → Remembering → Need to remember new concepts/less...
  - Remembering what to do
    - Github Issues to be done → Continuing implementation → Forgetting to do github issues
    - Fix bugs we encountered → Forgot what line broke our code → Forgetting to document it

Developing a workflow
Break down each theme into its components

- Lack of Workflow/Direction
  - Collaboration between Teammates
    - Lack of task delegation among team members → Overlapping Code and Inefficient work → One person working harder than others
    - No communication → Forgetting to check slack and github → Skip the meeting
  - Lack of planning out future work
    - Waiting for other parts of the project → Need to keep track of who is in charge of... → Lack of clarity of Chain of Handover
    - Bad Scheduling → Problem breaking down tasks into su... → Hard to maintain a consistent routine

Testing and Maintainance

- Poor Code Quality?
  - Debugging/Testing
    - Insufficient test cases → Lack of accessibility and usability for users → Lack of area for further development by other developer
    - Bad Testing (Not indepth) leading to poor release → Bad User Experience (Bugs & Glitches) → UCD Being Ignored
  - Code Reviews
    - Lack of Comments in code → Problem understanding how the program runs → Lack of diagramming for understanding
    - Differing Ideas → Problem remembering everyone's idea → Difference in understanding of User Needs

Personal Issues/Conflicts/Documentation
Break down each theme into its components

- Scheduling Conflicts/Other Commitments
  - School/Work related
    - Conflict of schedule between classes/ exams... → Balancing other work → Hard to keep track of hours spent coding
    - Out of sync with team(Including onboarding) → Hard to quickly catch up on what is happening → Toxic work environment
  - Personal life
    - Work life balance → Developer doesn't enjoy developing → will stop being productive in work
    - Dates/Marriages/ Events/more/ emergencies → Awkward Conversation to update the whole team → Forgets important personal life info

Resources for Help
Break down each theme into its components

- Unfamiliarity with technology
  - No resources for devs to learn
    - Unfamiliar with tools/ frameworks → Harder to fix bugs → Different syntax
    - Forgets important concepts → Data Structure/ Algorithms → Thrown into the deep end straight away
  - Out of Bounds Resources (NDA, Company Policies)
    - Ask Teammates for Help → Introverted? Afraid what others might think? → Asking too late so no one is available
    - Go through Company Process for help → problem forgetting important company... → Unfamiliarity with Assigned Individuals

Click each section and press DEL to remove instructions

Competitor Analysis
Determine where you stand within your market with our competitor analysis template. Uncover advantages, disadvantages, and potential opportunities for your business.

GOOD FOR
- identifying areas to improve or differentiate
- Uncovering market trends and opportunities to help guide your decision-making process

PEOPLE
1 to as many as you want!

MORE TEMPLATES
SWOT Prioritization Matrix
Affinity Diagram
User Persona

QUICK TIPS

Toolbar
It's at the bottom of your screen, with stickies, stamps, and anything you need.

Move and Zoom
Holding Spacebar, use the hand tool to pan around. Zoom controls are in the top right corner.
Spacebar

Work with your team
FigJam's better with others. Click the Share button above to invite your team.

HELPFUL TOOLS

Stickies
Press "S" to quickly add a sticky to the canvas.
S

Rectangles
Press "R" to quickly place a rectangle onto the canvas. Other shapes can be selected from the toolbar or swapped from the shape menu.
R

Hyperlinks
Hyperlink a text selection to a URL. This is great for linking to external docs, websites, Figma files or prototypes!

Comparison

	Unique Value Proposition What makes this company unique?	Company Advantages What are the things that provide a leg up?	Company disadvantages Where might drawbacks exist?
Github	Its Source Control feature to compare and run tests Coding that is shared and edited by multiple people Very integrated with VSCode, a popular IDE for devs Interactive between User and Developer Main source of communication with the software Git Version control Can create a pipeline that onboards people Ability to upload code and share it with other developers. Built by devs for devs	Long. time customer dedication/Biggest standard in other companies Github classroom for learning Microsoft's Backing Pull request for further development Compatability with IDEs Solidify a community for version control and developers Connecting VSCode to GitHub account Partnered with lots of 3rd party tools	Managing every single repository/code pushed to the site requiring a lot of storage Competition with GitLab Unknown tools e.g. need to learn how to use many tools/features Bad UX, only for professional use steep learning curve for beginners No way to communicate with other devs like texting or calling Monopolized Some tools are not managable Specific in the sense that the company can't really expand for more general audience
Notion	Has multiple formats that can take notes/track schedule Users can create templates to reuse Catchy name Graphic appealing Customized templates Adding widgets. Great UI/UX	Notes are easily accessible regardless the device they are on Very clean user interface User interface/ design / aesthetics Useful tool for Academic performance support embedding from a multitude of platforms Large Team, Passionate Users	Many other companies have similar features/ goals No offline access/limited capabilities Disconnected from other EcoSystem (Google, Microsoft, Apple) Can't edit or use offline
Slack	Dedicated for work/coding as channels have bots to accommodate different file formats/extension Very popular chat app for the professional setting Multiple Features (Call, Chat, Files, etc) Directed and focused app versus like discord which is a way to communicate, but slack is more of a disciplined-oriented and professional app for productivity call and screen sharing feature	All the possible tools that is needed for working group environments Can be integrated with Github Infrastructure and Cloud Computing Organizing information into different sections cross platform can use across multiple devices	Doesn't appeal to a broader audience Premium features cost money Very Workspace focused, lack of Personal Touch Separate Accounts for different workspaces Competitive market, heavy competition from discord and other chatting apps Lots of similar features to big competitors

Similar Capabilities
What are things that we want to do?

- Have organization tools similar (side bar create hierarchy - nesting pages) to Notion
- Being able to build an interactive app
- Being able to write text/ journal/summary of the day (Notion)
- Great UX like Notion
- Intuitive UI that isn't too complicated (Notion, Slack)
- Minimal and simple; ease of use
- Able to upload code in an organized/neat way. (GitHub)
- Something like CoPilot which can act as a Resource?
- User friendly -> Not too overly complicated.

Apparent Differences
What are the differences between them and us?

- Focus on a more professional focused Audience unlike Notion
- Focusing on local first, so online api/resources are nice but will not always be available (GitHub/Slack)
- More focused on notes and documentation instead of code (GitHub)
- The audience is for dev versus not devs.
- Integration capabilities like notion
- More focused on communicating rather than documentation (Slack)
- Much smaller user base, which is ultimately us, so we can tailor our functions and UI to whatever fits us best

Key Learnings
What can we learn from this process?

- Understand the strength of each product offerings and what we can learn from them
- Trying to understand what is achievable within the time we have left in class
- How subteams can communicate and collaborate
- Draw from what's doing well in other companies and try to apply that to our product
- Do what is capable with our current abilities, don't make things overly complicated
- How idea and strategy can be generated within the group
- Minimalist approach - the app should have exactly what it needs, but not more not less
- Less is more
- product research and developing a simple theme

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
- Catering these tools towards a developer and having relating features on same page(really easy to access (Keeping related things close together)
- Find value in creating a product that we can end up using daily in the future
- Make it accessible for old people as well
- Sort each implementation by 3 things: which one is most important and is it code-able within our time frame.
- Create it with longevity in mind

Autograder UCSD

amazon

Obsidian

Google Calendar

Notion

Github

Slack

Microsoft Outlook

Native Apple Notes Reminders

Trello

Miro

Discord

spotify

Figma

Turnitin

Todoist

windows sticky notes

OneNote

Google Keep

duolingo

nvidia

Personal Issues and Conflicts
Break down each theme into its components

- Personal Issues/Conflicts
  - Write down thoughts/complaints/rants about company to help visualize your thinking → Inform the Team about Feedback on the Process → Reach out if any help is needed to other team members → Keep track of which team member was in charge of what → Mark areas of Conflict or Concern in the Meeting
  - Link your Personal Calendar, which can help see the things you have or need to do at home → Have a personal calendar with important info → Regularly update your personal calendar → Have a way to stop scheduling things at the same time as personal commitments → Create a user portfolio section with their calendar attached
  - Communicate to your team/leads if there are any schedule conflicts and try to get any work done beforehand → Use general chat on slack → DM team leads → Creating a list of sources of who to contact for certain issues → Check the message in group chat regularly
  - Finding available timeslots within schedule to shift tasks around → Time Management → Creating something similar to when2meet → Having priority levels for every task → Prioritize the meeting time

Continuity Between Work
Break down each theme into its components

- Continuity Between Work
  - Create a summary of what was accomplished that day and share it to everyone. → Shared To-Do List along with individual → Having a way to link it to GitHub Issues → Daily Logs on a shared google doc → Integrating a Shared Notes Functionality
  - Write down thoughts you have for future implementation so you don't forget → Documentation → Writing clear comments on code, both about what it... → How one file of code is linked to other files → code block(``` ```) feature to display code cleanly
  - Sustained effort so there is no need to cram in a lot of work close to the deadline → Weekly meeting of work done/needed → Create something like meeting minutes/ notes → Quick templates where it's easy to fill... → Organizing the To-Do stuff across weeks and deadlines
  - Have a Completed List next to the To-Do List in the App which can remind yourself of the completion → Keep track of what was accomplished and what wasn't → Create a To-Do list at the start of each session → Document what was finished → Ideally, the To-Do list gets emptier and accomplished gets more full

Developing a Workflow
Break down each theme into its components

- Developing a Workflow
  - Having a To-Do List for the Entire Team and the team lead having the ability to separate them. → Create task list, and use this to create "issues" on github repo → Assign tasks to each team member → Team Lead has access to edit to-do list → To-Do list accessible by the entire team
  - Create a diagram like a flow chart to know how to your implementation should go. → Visual diagrams of the process → Use given models like a waterfall chart → Non-complex flow charts → Colors to separate topics
  - Write down a detailed approach to your implementation, breaking it up into organized sub-tasks → Need structured plan for implementation → Break down plan into sub-tasks to complete → Quick templates where it's easy to fill out → Need an ending goal to finish
  - Develop a Guidemap of what is needed to be done similar to a waterfall diagram → Guiding templates to breakdown the process → Questions to ask how a task can be made to be "bite sized" → Must finish one step before moving on → Have multiple templates to choose from depending on what was accomplished

Testing and Maintainance
Break down each theme into its components

- Poor Code Quality?
  - Ask users to test using a prototype so can observe any gaps in usability // perhaps how we didn't reach the requirements → Create test scenarios → Have users be different team members → Users give feedback in a form/ notes → Note down all critiques from users to create new tasks
  - Draw and upload diagrams of the current codebase to help understanding of how everything works together → Understand relations between classes,files and methods → Good documentation and variable names → Write down questions about classes, methods, files you have → Everyone has the same sense of syntax(such as spacing)
  - Create issues when bugs are found and left unfixed → Keep track of bugs during implementation → Create comments during coding → Record where the bug is (line, method, file) → Have a system to indicate whether the bug was fixed yet or not
  - Need a Way to Un-Complete that task if the test fails → Having a full history/ table of tasks → Retrieving done tasks back to undone with a marker/counter → Review session/ hypothesis of faulty code → Attempting to fix code with new knowledge/help

Resources
Break down each theme into its components

- Uncertainty
  - Write down questions you have that you want to ask later → Make a Ticket like Autograder? → Feature where it displays priority of ticket (green , red, yellow) → Contact TAs/tutors if there are no online resources that can h... → Go through the code during office hour
  - Have a Contact for Resources section, where every team member is assigned a strength and you can ask them about the issues → Have Strengths and Weaknesses listed → Reach out to member who has most expertise in issue → Starts bottom up from Seniority to make it less awkward → Gives the option to either choose or randomly pair
  - Coding Language documentation → Notes of function/ files → Resources from the lab assignment → Check previous notes in case you've encountered problem before → Link directly to official documentation like VSCode