Project Planning

How to get started!
TODAY'S AGENDA

01. ANNOUNCEMENTS

02. DEV STYLES AND PLANNING TOOLS

03. MVP AND SCHEDULE
   Including an example

04. TIPS ON TEAMWORK

05. CHOOSING A PROJECT ACTIVITY
- Dev styles
- Project planning tools (issues, kanban board)
- Tips on picking an MVP and schedule
- Leadership? Skills (conflict resolution)
- Example project proposal
- Activity - evaluating plans
  - Assigning difficulty/time of a task? (time is difficult)
ANNOUNCEMENTS!

- Team formation form: Feb 1st
- Project Proposal First Draft - Feb 8th (final draft - Feb 15th)
THE DEVELOPMENT METHODOLOGIES

- A variety of strategies of software development (how to split up work among team, how to split up project into tasks, how to split up time for each task/what to work on each week, etc.)
- Each strategy has pros and cons, can incorporate different parts that fit your team’s dynamics and goals
- Adjust your strategy as you go along
- We’ll cover two of the most popular methodologies, read more about them here;
  - https://www.uptech.team/blog/software-development-methodologies
THE DEVELOPMENT METHODOLOGIES
- Agile Development

- Assign tasks to “sprints”, or constant stretches of time (~1-3 weeks). Team will meet at least once every sprint, constant communication
- Constantly test and ask for user feedback, work on project iteratively
- Variations on Agile - Rapid Application Development (constantly re-working a prototype), Scrum (shorter, more efficient springs with assigned scrum master)

Pros:
- Clear and useful team communication
- Iterative development process reduces mistakes

Cons:
- Suited for a large number of small, rapidly changing tasks (doesn't fit all project requirements)
- Requires developers to work independently for extended stretches

https://www.uptech.team/blog/software-development-methodologies
THE DEVELOPMENT METHODOLOGIES
- Waterfall Development

- Tasks are completed in sequential phases (often described in a Gantt chart)
  - Ex: (Brainstorming, Designing, Coding, System Tests, Bug fixing, Deploying)

Pros:
- Simple, easy to implement for beginners
- No room for miscommunication, everyone always on the same page
- Requires full project be scoped out ahead of time

Cons:
- Inflexible, not suited for complex projects that require re-thinking as you go
- Doesn’t incorporate feedback/testing in early stages
- Requires estimating time per task!

https://www.uptech.team/blog/software-development-methodologies
YOUR PROJECT PLANNING TOOLKIT

- Several online tools to break down project into
  - tasks (ex. ‘finish frontend homepage’, ‘setup database’)
  - features (ex. ‘add back button to header’, ‘implement form validation’)
  - bugs (ex. ‘fix inconsistent styling’, ‘figure out broken submit button’)

- Collectively called issues

- Once an issue has been created, assign it to a developer and keep track of its progress

- Why?
  - Won’t lose track of any required work
  - Easier to prioritize tasks, plan out what to work on each week, stay on track to meet deadline
  - More rewarding to see tangible progress
  - Industry standard
YOUR PROJECT PLANNING TOOLKIT

- Popular choice - GitHub Projects (stop at kanban board)

YOUR PROJECT PLANNING TOOLKIT

- Alternatives
  - Trello - very visual tool, create customized Kanban boards
  - Asana - “all-in-one” type tool, integrates well with GitHub, Google Drive, etc.
  - Honestly just a Google Doc/Spreadsheet/Notion page with a list of tasks

- Tip: *don’t* overcomplicate this part! (avoid high-learning-curve tools like Jira - not needed!)

- Tip: treat each week as a sprint, and the weekly check-in as a stand-up meeting. At every weekly check-in,
  - evaluate what tasks have been completed and what is taking longer than expected.
  - Go through any blocks, help each other out
  - Decide what needs to be done next week. Keep a list of ‘weekly tasks’, including who they’re assigned to, very easily accessible
REVIEW: what is an MVP?
○ Minimal Viable Project = bare-bones of a project, with only the features needed to satisfy the problem statement.
○ So minimal UI, no bells-and-whistles type features, less edge-case handling

Why?
○ Picking an MVP helps with timeline (more in a bit)
○ Prioritization of tasks - keeping the end goal in mind when making early decisions

Examples
○ A suitcase - a box that opens and shuts with a clasp, with wheels
○ VS Code - a desktop application that opens code files in an editing environment, with a terminal on the bottom and a file explorer on the side
○ Google - a web app with a search bar that takes text and searches through a database of all website titles
Once you’ve split project into tasks, categorize them into
- Setup
- MVP
- Bells-and-whistles
- Reach for the moon

Try finishing the setup as early as you can - everyone should start on the same page, “dev env issues” should be resolved ASAP

Aim to complete MVP ~60-75% into the timeline

Allocate time for testing and resolving bugs

Reserve buffer time near the end! Especially important to allow for flexibility
- if you’re learning a new technology, allow for flexibility
- if you want to incorporate user feedback into project
- if you’re new to estimating time for tasks - this is almost always an underestimate

https://www.atlassian.com/work-management/project-management/project-planning/timeline
TEAMWORK SKILLS

- COMMUNICATION COMMUNICATION COMMUNICATION
  - Don’t silo yourself (it sucks)
- Use GitHub correctly!
- (Giving and Receiving) Feedback is important
- **Conflict Resolution:** Think as the user, not as a programmer
- It’s okay to not know something
- Did I say communication?
AN EXAMPLE
PROJECT PROPOSAL

- Just a guideline (as usual)
PROJECT PLAN
EVALUATION - TEAM
ACTIVITY
THANKS!

Any questions?

Feel free to hang around after class or email us at cs222sp2023leads-group@office365.illinois.edu!