Lesson 8.2 - Defining Requirements ## Learning Objectives Students will be able to... * Define key scenarios for a project and the features required to implement each scenario * Explain the importance of flow charting when designing an application ## Materials/Preparation * [Final Project Plan Organizer] handout ## Pacing Guide | Duration | Description | | --------- | ---------- | | 5 minutes | Do Now | | 10 minutes | Review pitches | | 20 minutes | Defining Scenarios | | 15 minutes | Flow Chart | | 5 minutes | Debrief and wrap-up| ## Instructor's Notes ### 1. Do Now * Project the Do Now on the board, circulate around the class to check that students are working and understand the instructions. ### 2. Review Pitches * If desired, give students a few minutes to rework their pitches or get more feedback from a classmate or instructor. * Ask students to choose which idea they want to pursue, and write it down on the top of their [Final Project Plan Organizer]. ### 3. Scenarios * A description of a set of interactions and/or tasks that describe a start-to-finish example of how a user might want to use the application_ * Explain that defining scenarios helps a programmer focus on what features are actually necessary to enable the key user interactions for their application * Instruct students to write down at least **3** scenarios for their project describing, from start to finish, interactions a user might have with their program to accomplish a specific goal * The scenarios should have a moderate level of detail in the description of the user interaction (e.g. "print a board," "input their name," etc.) but should not include any design or implementation details. * Since this semester is text based a scenario would more likely be: text-based user flows (interactions with the console) * Once students have written their scenarios, they should review them and develop a list of the necessary features to enable each scenario * there should be minimal technical detail in these descriptions, instead focusing on details of the user experience. The feature lists should should be more about _requirements_ than implementation. ### 3. Flow Chart * The different paths a user can take through a program. * At each step write down what options the user has and what scenarios that can lead to. * Similar to the wire frame of last semester (wireframes typically deal with screens, so a flow chart is a good alternative) * Flow Charts do not include any details (such as specific text), but instead provide a broad impression of what an application will be like to aid in design and planning * Students will complete page 1 of the organizer by writing out the important interactions of their program. * Encourage students to reference their feature lists to ensure they include _all_ necessary interactions for their project, including simple things like a intro interaction, help interaction, or exit ("game over"). ### 4. Debrief * As class ends, ensure students retain their work as they will use it to construct a detailed specification and implementation plan tomorrow. ## Accommodation/Differentiation [Final Project Plan Organizer]:https://teals-introcs.gitbooks.io/2nd-semester-introduction-to-computer-science-pri/content/units/8_unit/final_project_plan_organizer.docx [Final Project Development Plan]:https://teals-introcs.gitbooks.io/2nd-semester-introduction-to-computer-science-pri/content/units/8_unit/final_project_development_plan.docx ## Forum discussion [Lesson 8.2 - Defining Requirements (TEALS Discourse Account Required)](https://forums.tealsk12.org/c/2nd-semester-unit-8-final-project/lesson-8-02-defining-requirements)