Lesson 19: Bugs and Features

45 minutes

Overview

Teams have at this point developed an app prototype that has gone through multiple iterations and rounds of user testing. With the information and guidance gained from the last round of user testing, each student has the opportunity to plan for and implement improvements (which could be adding programmatic functionality, a more eye-catching design, more informative text copy, better uniformity of iconography, or any number of other non-programming related features) to the team app. Depending on the time you have available, and student interest, you can run the cycle of testing and iteration as many times as you see fit.

Question of the Day: How can we create a plan to address bugs and features in our prototype?

Standards

Full Course Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming
- ▶ **CS** - Computing Systems
- ▶ **DA** - Data & Analysis
- ▶ **IC** - Impacts of Computing

Agenda

Warm Up (5 minutes)
- **Getting Prepared**
- **Journal**

Activity (35 minutes)
- **Interpreting User Feedback**
- **Brainstorming Session**
- **Bug and Feature Analysis**

Wrap Up (5 minutes)
- **Journal**

Objectives

Students will be able to:

- Analyze user feedback and test results on a computational artifact
- Categorize and prioritize the issues according to impact and ease of implementation

Preparation

- Print one copy of the activity guide for each team
- Set out sticky notes for each team
- (Optional) Have poster paper or a large whiteboard area prepared for each team
- Check the **"Teacher's Lounge"** forum for verified teachers to find additional strategies or resources shared by fellow teachers
- If you are teaching virtually, consider checking our **Virtual Lesson Modifications**

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the teachers

- **Bugs and Features** - Slides
  ▾ Make a Copy

For the students

- **Bugs and Features** - Activity Guide
  ▾ Make a Copy

Vocabulary

- **Bug** - Part of a program that does not work correctly.
- **Feature** - Part of a program that adds functionality for the user

Teaching Guide

Warm Up (5 minutes)

Getting Prepared

Distribute: Make sure each team has their materials available, especially the Digital Prototype User Testing activity guide from yesterday.

Journal

Prompt: Based on your user testing yesterday, what are some issues you discovered in your apps?

Discuss: Have students journal individually first, then share in their teams.

Discussion Goal: Students can use this prompt to remind themselves of the user feedback from yesterday and as a transition into the main activity.

Remarks

> Based on our feedback from yesterday, we're going to take some time to do one last round of improvements on our prototypes. However, before we get started, we're going to spend today looking at a new way to analyze our feedback from yesterday and prioritize which tasks we want to incorporate into our next prototype. One of our first tasks is grouping our feedback into two categories: a **bug** or a **feature**.

Question of the Day: How can we create a plan to address bugs and features in our prototype?

Activity (35 minutes)

Vocab: introduce the following vocabulary terms:

- **Bug** - Part of a program that does not work correctly.
- **Feature** - Part of a program that adds functionality for the user

Remarks

> When testing your program, your users may have identified bugs in your prototype. These could be programming bugs, where an event didn't work the way you expected. Or these could be design bugs, where a piece of text was too small or the design of a screen wasn't very clear. Your users may have also given you feedback requesting new features for your app that may be worth investigating. Let's see if we can classify our feedback as either a bug fix or a feature request.

Distribute: Poster paper, sticky notes, and a copy of the activity guide for each team.

Teaching Tip

Reducing Printed Materials: This Activity Guide can be completed online or as a journal activity.

Interpreting User Feedback

Do This: Teams start by completing a T chart that will help connect specific user testing observations to the potential bugs or missing features they reveal. They should rely on their notes from yesterday's lesson when testing with another user. This information will be the basis of a brainstorming session to create a list of bugs and features the team would like to focus on in the next activity.

Teaching Tip

Deja Vu: The chart on this activity is identical to the one from yesterday's activity guide. If students had enough time to complete the chart yesterday, they don't need to re-copy it onto this activity guide. However, if students continued to test the app at home with their family or members of their community, then they can use the chart on today's activity guide to summarize that feedback.

Circulate: Monitor students as they create their list, ensuring all voices within the team are heard and acknowledged. Look for clear connections between the observations students made as their users tested their apps, and the changes they want to make to their apps based on those observations.

Brainstorming Session

Do This: Once teams have organized all of their feedback into the T chart, they can move into the brainstorming phase. Directions for this stage are in the activity guide and on the slide:

Teaching Tip

Including All Team Members: It's tempting to focus solely on bugs that are the cause of, or can be solved with, code. Remind students that there are many roles and skillsets on a software development team, and not all bugs and features are dealt with by programmers. Inconsistent color, confusing text, and counterintuitive layouts are all potential bugs that are important to deal with.

This is an opportunity to let students with other skillsets shine and make a strong contribution to their apps, so make the extra effort to help highlight those students.

- The top of the sticky note should say **BUG** or **FEATURE**.
- The middle of the sticky note should be a description of what the bug or features is.
- The bottom of the sticky should have a quick estimate of how long (in minutes) it will take to fix this bug or implement this feature.

Circulate: Monitor students as they create their sticky notes. Help answer questions students may have on whether a task is a bug or a feature. It's okay if this distinction isn't always clear, as long as teams can make a best guess. Also help teams determine time estimates for each task - small changes to design elements may not take very long, but entirely new features requiring new screens could take a significant amount of class time.

Assessment Opportunity

The left side of the T-chart on the first page should include at least four descriptions of things that happened in the test, and the right side should list a reasonable interpretation of the descriptions.

Bug and Feature Analysis

Do This: On the second page of the activity guide, students will categorize the post-its they generated in their brainstorm. For each sticky note, discuss whether it is urgent or not and whether it seems to be easy or difficult to implement. Based on that discussion, place the sticky in the appropriate quadrant. Optionally, you can have teams re-create this chart on poster paper or a large whiteboard and organize their sticky notes there.

Circulate: Monitor students as they categorize their post-its. The goal is to think intentionally about both urgency and complexity, which will further help them prioritize their work when they begin implementing these changes.

Assessment Opportunity

The chart should include several improvements to the app, categorized according to urgency and ease of implementation.

Prompt: Now that you have your bugs and features categorized, which of the four categories should be the first that you tackle? Which should be the last?

Discuss: Have students share in their teams first, then facilitate a class discussion by asking each group to share their thoughts with the class.

Discussion Goal: Students should realize that urgent goals should be a higher priority than non-urgent goals, and that easier fixes will probably get done faster than harder fixes. This means the upper-left quadrant is the first group teams will probably tackle. After that, teams may debate which quadrant is more important - urgent fixes that are harder to implement, or non-urgent fixes that are easier to implement. It's worth discussing the pros and cons of both approaches since teams will have to make this decision tomorrow when implementing their changes.

Remarks

> You all have come up with a really organized plan to start making your changes tomorrow. This is a similar process to what real software developers do when updating their apps - they collect feedback and categorize it so they know what changes they will make and how long each change will take. Tomorrow we'll pick which post-its we want to focus on and update our apps

Wrap Up (5 minutes)

Collect: Collect all the materials from each team in a safe storage location. Try to keep the post-its organized from today's activity. If teams used a whiteboard for their post-it notes, consider having them take a picture of how they organized their post-its.

Journal

Prompt: This process of organizing tasks and categorizing into a chart is used a lot when designing new apps. What are some other tasks this process could be used for in your life outside of class?

Discuss: Have students journal individually first, then have students share in their teams before asking a few students to share with the class.

Discussion Goal: Students should make a connection between this organization process and different tasks they have outside of class. Some examples may include:

- Deciding which homework assignments to prioritize
- Deciding which chores to complete at home
- Deciding how to plan for an event, such as a birthday party or school event

If possible, connect this process to strategies related to stress management and not being overwhelmed. When faced with a lot of work all at once, writing each piece down in bite-sized steps is a good strategy for getting started on large projects and managing stress.