Contextualization/Design Check-In

Code Critiquer System for the C Language and Embedded C

sdmay25-23:
James Joseph
Samuel Lickteig
Alix Noble
Andrew Sand
Owen Sauser

Client and Advisor: Dr. Diane Rover

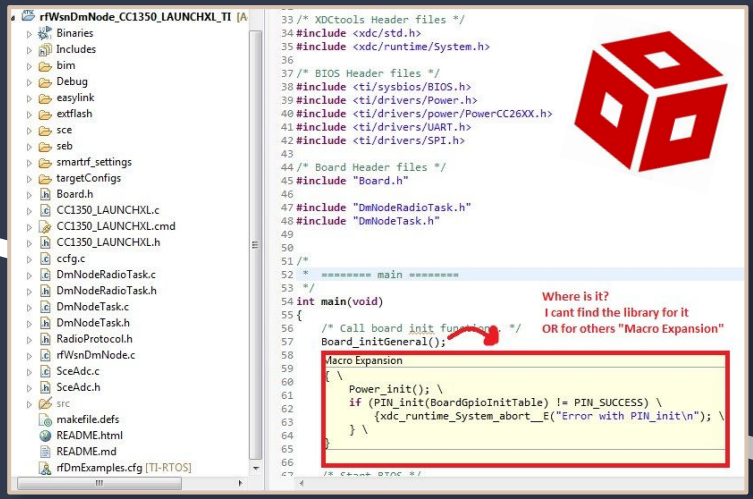
Code Critiquer System for the C Language and Embedded C

Project Overview

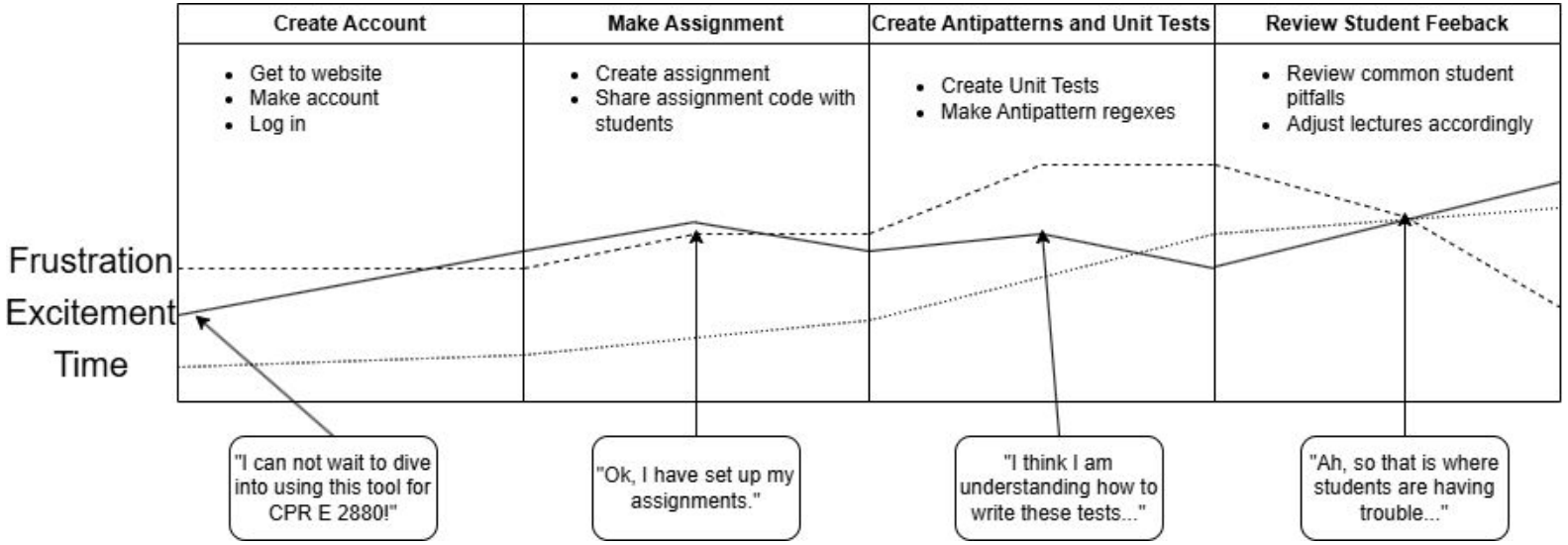
- Current state of project (continuation of sdmay24-34) is a web-based critiquer tool
  - Students upload C files to tool
  - Files are statically analyzed to search for antipatterns
  - Tool generates feedback
  - Students use feedback to improve skills
- Will modify current system and/or develop new prototypes
- Ideally tailored for CPR E 288
- Targeting a Spring Semester Prototype

Problem Statement

- Static Code Analysis is a challenging problem
- Many off-the-shelf solutions
  - Many leave a lot to be desired
  - They are not bespoke for CPR E 288 usage
  - No off-the-shelf “perfect combination” for what the project client needs
- Client needs a code critiquer that can...
  - Be accessed by students and instructors
  - Provide beginner-oriented feedback
  - Ability to give embedded and datasheet-focused feedback
  - Potentially integrate with Code Composer Studio



Journey Map For Instructor User



This shows the experience of our instructor user, while using the product. While there are certain parts of the process that are more frustrating to use, the end result will have them be less frustrated than they would normally and with a better understanding of how students are learning.

Pro/Con Table

	PC-Lint Plus	CPP Check	Code Pal	Clang-Tidy	Our Solution
Pros	-Certified by ISO 26262 and IEC 61508 -Also identifies security issues -Also supports C++	-Lots of resources online about it due to large user base -Quick and easy setup -Fast runtime -Also supports C++	-Browser-based, meaning it is highly accessible -AI based, so feedback is much more specific and points to additional resources	-The most customizable alternative -Allows users to customize what checks or groups of checks to run -Supports some popular 3rd party libraries -Can be used as a linter as well	-Allows instructors to specialize feedback to a class -Students get feedback more tuned to the course -University will have access to source code, allows great flexibility
Cons	-Costs money to use -Non-open source-friendly license, which limits flexibility	-Poor consistency with more false positives compared to alternatives	-Costs money to use -AI based, so it can be inaccurate or lie about topics -Unable to run locally	-Slow runtime (Potentially up to a whole day) -Review files eat up a large amount of disc space	-Requires developer maintenance -Requires instructors to be knowledgeable of Regex and antipatterns -Requires Iowa State to host it, which costs money and resources

Technical Complexity Analysis

- Internal Complexity
  - Uses Regex and XPath analysis
  - Simulates submitted code in sandbox
  - Tests via Unity (Testing framework for C)
  - Many languages and protocols working together
  - Using secure methods and protocol for data safety
- External Complexity
  - Static code analysis and antipatterns are an evolving problem
  - Dynamic analysis is also an evolving and complicated problem in computer science

Addressing Human Needs

- The primary way we plan to address the human needs is by getting feedback on the previous team's design
- This will let us identify any pain points based on the student feedback
- This has led use to have more of a focus on integrating the project into CprE 2880 tools. Such as:
  - Code Composer Studio Support
  - Datasheet-specific functionality
- We also plan to improve the UI/UX to make a better user experience for the students and instructors

Addressing Economic Needs

- The team's solution is tailored for CprE 2880 and its assignments
- Design will allow for both static and dynamic analysis
    - Something other solutions do not offer
- Project will be integrated with resources used in CprE 2880, such as Code Composer Studio and the instructor's assignments
- Bespoke tool, so no external support
    - Offset by using common industry tools and languages such as HTML, Python, Flask, and more

Addressing Technical Needs

- Design uses both static and dynamic code analysis
  - Both are complicated topics in Computer Science
  - The team will be challenged with implementing them to a desirable degree
- Project has many moving parts and protocols interacting
  - This is necessary for an “all-in-one” solution
  - Team is vigilante about ensuring everything interacts correctly, securely, and consistently

Any Questions, Suggestions, or Comments?