Code Critiquer System for the C Language and Embedded C

Design Document

Team sdmay25-23
Client: Dr. Diane Rover
Advisor: Dr. Diane Rover

Team Members:

James Joseph	Secure System Design, CPR E 2880 Liaison
Samuel Lickteig	Backend System Design
Alix Noble	Testing
Andrew Sand	Team Organization, CPR E 2880 Liaison
Owen Sauser	Client Interaction, Frontend System Design, CPR E 2880 Liaison

Team Website:
sdmay25-23.sd.ece.iastate.edu

Team Email:
sdmay25-23@iastate.edu

Revised:
7 Dec. 2024
Version 2.0

Executive Summary

Programming in C, especially in embedded contexts, can be a rather challenging language for students and novice programmers to comprehend, resulting in a higher barrier to entry and numerous struggles. For instance, compiler feedback can be cryptic and deeply nested or confusing errors can arise that are formidable for beginners to debug.

The Code Critiquer System for the C Language and Embedded C is a web application and database that checks submitted C source files for antipatterns or other poor programming practices, and it is a continuation of team sdmay24-34's project. The project is targeting usage in the CPR E 2880 (Embedded System I: Introduction) course at Iowa State University, where students will be able to upload their code for critiquing, and teaching assistants and instructors will be able to create assignments, select which antipatterns to test for, manage student user accounts, and view usage statistics and telemetry. The project's overarching goal is to improve student learning while reducing the workload on instructors and teaching assistants, resulting in a significantly enhanced course experience.

The design must be easy to use by both students and staff, and the generated feedback must be returned in a timely manner and understandable to beginner programmers. Additionally, the project must be well documented and modular to support further programming languages and Iowa State courses in the future. The design's frontend web application is what users will interact with, allowing them to upload files to check, view the generated feedback, and more. It is written in HTML and CSS, with routing handled by the Flask framework, and Python code serves dynamic data using templates. As for the project's backend, a database maintained using MySQL stores all of the system's information. Additionally, a server sandboxes each critiquer run using Docker. As this project is a continuation from team sdmay24-34's, most of the core frontend and backend design is already implemented, but it requires refinement and various improvements, such as a user interface overall, bug fixes, a better account system, and statistics viewing before it is ready to be used in CPR E 2880. The team has made great strides in bringing it closer to a public release by fixing bugs, beginning work on a statistics page, and sandboxing the critiquer runs using Docker. With frequent contact with the team's advisor and client, Dr. Diane Rover, the team ensures that the design stays focused on fulfilling the functional and non-functional requirements. Looking toward the future, the team aims to improve the system's aforementioned requirements before it can be used in CPR E 2880.

Learning Summary

Development Standards & Practices Used

- Software Practices
    - Version Control - Project uses GitLab to ensure all developers can contribute simultaneously on different features
    - Continuous Integration and Continuous Deployment - Ensuring that the published build is up to date and deployed
    - Testing - Running tests to ensure the system is stable and secure
- ABET Criteria
    - Applying principles of engineering, mathematics, and science
    - Identifying, formulating, and solving complex engineering problems
    - Communicate effectively with a wide range of audiences
- IEEE and ISO Engineering Standards (See section 2.2 for more details)

Summary of Requirements

- The critiquer must be easy and quick to use with a clean user interface
- The critiquer must catch antipatterns in C and in embedded contexts
- Generated feedback must be useful, concise, and aid the user in learning
- Usable in different contexts by students, teaching assistants, and CPR E 2880 instructors at Iowa State University
- Usage statistics must be aggregated and presented in ways useful to instructors for improving the CPR E 2880 course

Applicable Courses from Iowa State University Curriculum

- COM S 1850: Introduction to the C programming language
- COM S 3090: Team project management and web application development
- COM S 3170: Understanding how software is tested
- COM S 3270: Deepening knowledge of C
- COM S 3630: Understanding and managing database systems
- CPR E 2880: Embedded Systems and the primary concern of this project

New Skills/Knowledge acquired that was not taught in courses

- Regular Expressions
- XPath
- Abstract Syntax Trees

Table of Contents

List of Figures and Definitions

Definitions:

Antipattern: A distinguishable error, bad practice, or flaw in written code. For example, an antipattern can manifest as a syntax error, stylistic inconsistency, inefficient structure, or compiler error.

Regular Expression: A series of characters used to identify and select patterns in a plaintext file or input. It is used in this project to search for specific patterns in code to perform static analysis.

XPath: An expression language primarily used to explore files of XML format. It is used in this project to traverse and search the abstract syntax tree representations of code to perform static analysis.

List of Figures:

List of Tables:

1.0 Introduction

1.1. Problem Statement

The C programming language can be a rather tricky subject for students, beginner programmers, and even experts to comprehend and learn successfully. Compared to other programming languages, C can often be considered to have none of the "training wheels" or safeguards common with the others. However, the language is still widely used in several fields due to its efficiency and more nuanced control over computer hardware. Therefore, C is still an essential topic for most computer and software engineers. The overarching goal of this project is to reduce the barrier to entry for the language in an academic context tailored explicitly for the CPR E 2880 course at Iowa State University by improving upon the work done by team sdmay24-34. The created software system will allow students to upload their code to be automatically analyzed and have constructive, easily understood feedback generated. Furthermore, instructors and teaching assistants will be able to view and collect analytics about what students are submitting to the tool. This software solution will also aid in reducing the workload on instructors and teaching assistants in addition to the response times to answer student programming questions.

1.2. Intended Users

Since the Code Critiquer System for the C Language and Embedded C project is targeting usage for Iowa State's CPR E 2880 course, three primary user groups have been identified:

User Group #1: Students

Description: CPR E Students are usually rather busy with their other classes and work, so they may not be available at the same times as the CPR E 2880 teaching assistants or professors. The students typically have some knowledge of the C programming language but need more in-depth or nuanced comprehension. They can frequently run into errors that they need help understanding.

Needs Statement: CPR E students need a way to get fast and reliable feedback on their work when teaching assistants or professors are unavailable.

Benefits: This product would allow the students to get feedback anytime and anywhere. It would describe to them the issues they are encountering and how they may fix or understand them. The feedback provided by the software solution will allow students to learn what mistakes they are making, why they are considered bad practice, and how to correct them.

User Group #2: Teaching Assistants

Description: Teaching assistants typically have a busy schedule, especially when working in large, lab-based courses such as CPR E 2880. They usually have the information a student may need to progress successfully but lack the ability to get to and help all students promptly.

Needs Statement: CPR E 2880 teaching assistants need a way for students to troubleshoot problems without their direct help.

Benefits: This product would give teaching assistants a lighter workload and allow them to focus more on their classes, work, or research. Furthermore, this software solution will significantly reduce the need for teaching assistants to answer trivial or recurring questions, allowing them to focus on giving students more complex or conceptual assistance.

User Group #3: Instructors

Description: Instructors design and create assignments and field questions from students regarding these assignments. Though they have contact with students, it can be challenging to gauge how their students are doing in class due to a lack of one-on-one time with students and student feedback.

Needs Statement: Instructors need a way to compare many students' work against a common set of issues.

Benefits: This product would reduce the number of questions instructors get from students, freeing up more time, similar to the benefits teaching assistants will see. It would also allow them to see what issues students have most often, allowing them to tailor their curriculum to students' needs better.

2.0 Requirements, Constraints, And Standards

2.1 Requirements & Constraints

1. **Functional**
   - Provide automated feedback based on the antipatterns in the database
   - Compile and run C code
   - Allow professors to add and remove custom antipatterns to the database
   - Allow students to submit code files for critiquing
   - Add clarity to existing static C analysis tools
   - Use simulated CyBots to analyze the results and function of the code
   - Allow instructors and teaching assistants to view analytics of student-analyzed code per assignment per section
2. **Resource**
   - Linux server for running the web application
     - 1 Core
     - 4GB Memory
     - 64GB Storage
     - Network interface card (NIC)
     - Public IP address with addressable port (Constraint)
     - Compatible with Python and program libraries (Constraint)
     - Compatible with libc (Constraint)
   - Server for running the CyBot simulation (possibly the same as the aforementioned)
     - Same as above
     - Same ISA and ABI of CyBot
   - Git repository for project development (To hold code and run pipelines)
3. **Aesthetic**
   - Clean and clear to make it easy to understand and receive the information
     - No overlapping text
     - Scales to multiple resolutions appropriately (Targeting only desktop resolutions such as 1920x1080p, as mobile is not a prioritized platform)
     - Using a legible serif or sans-serif font that is at least 12pt in size
4. **UI/UX**
   - Takes less than 10 seconds to analyze and return feedback
   - Easy-to-click buttons

    - No stacking buttons on top of each other
    - Buttons give visual feedback when they are clicked
    - Can click anywhere on the button to register clicking it

5. **Maintainability**

- The application must be well documented so it can be maintained and expanded upon after the team is done
- Instructors should be able to add, edit, and remove assignments in the system to better fit the needs of CPR E 2880 for a particular semester
- Database of antipatterns should be easily updatable, allowing the tool to be used for future applications

2.2 Engineering Standards

1. IEEE 1028-2008 - This standard relates to the project because it talks about reviewing code. Reviewing code and work from other teams is a significant part of this project, as the group needs to build off of what both the prior senior design team and the Michigan Tech University.
2. IEEE 2675-2021 - This standard covers the concepts of reliably, securing, and safely building, packaging, and deploying applications in relation to DevOps. Since our project focuses on having both a frontend and backend, practicing efficient and safe DevOps will be critical to the group's success.
3. IEEE 1016-1998 - It covers the recommended practices for Software Design Descriptors, which are a medium used for conveying the structure of a software system. It will be important to be able to effectively and concisely communicate how the senior design project is structured to the clients and advisor.
4. ISO/IEC 9899:2018 - Explains how C code is made to compile and run. This will be a useful resource to compare code against. Most code that goes against these patterns will include antipatterns
5. ISO/IEC/IEEE 15288:2023 - This standard describes the terminology, concepts, and frameworks that deal with the lifecycle of a software system. It applies to bespoke and mass-produced systems, so it will be applicable to our project.
6. ISO/IEC TS 17961:2013 - The rules for secure coding in the C language are covered in this standard. Since our project heavily deals with identifying errors and antipatterns in C programs, this standard will be an excellent reference for a more memory-safe and cybersecurity standpoint.

3.0 Project Plan

3.1 Project Management/Tracking Procedures

This project will use a hybrid management approach, drawing inspiration from both Agile and Waterfall. The overarching project is broken down into large Waterfall-like tasks that must be completed sequentially. However, each of these tasks will be worked on in an Agile manner with "sprints" taking place each week, consisting of a team meeting, advisor meeting, and planning goals to be completed for the next week. Furthermore, once a functioning prototype is completed by the second semester of Iowa State University's academic year, there will be "sprints" in the sense that functionality, feedback, and improvements will need to be worked on between each CPR E 2880 lab. To aid in this development cycle, the team will use GitLab issues to track the progress of features and manage Git progress. For communication between team members, the project advisor, and members from the prior senior design group, the team will continue to use Discord, as it allows for quick, efficient bursts of communication that do not rely on availability.

3.2 Task Decomposition

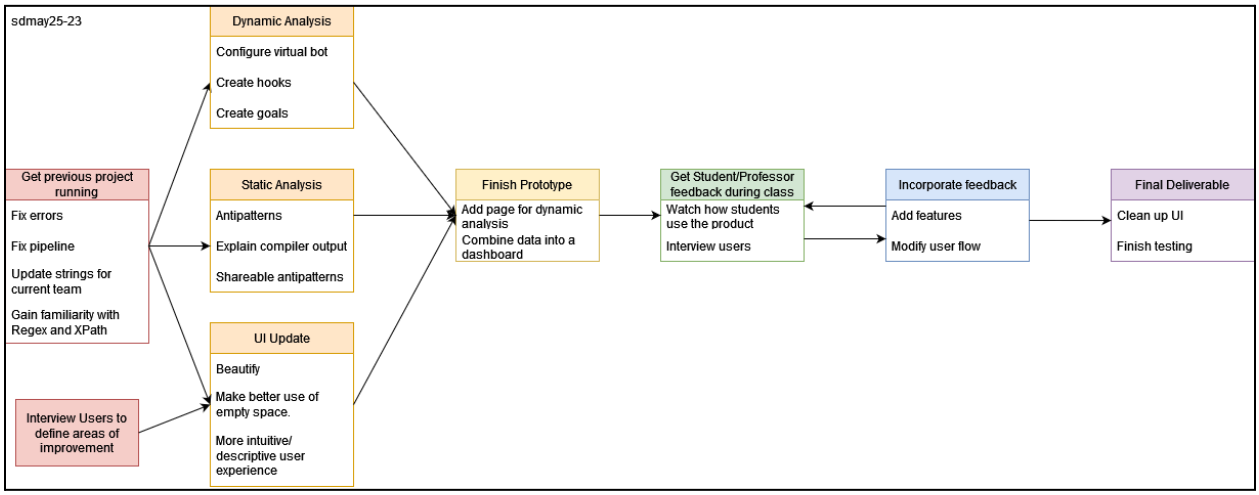

Fig. 1 Task Decomposition Chart

1. Get the previous project running
    a. Fix errors
    b. Fix pipeline
    c. Update strings for current team
    d. Gain familiarity with Regex/XPath
2. Feature development
    a. Dynamic analysis
        i. Configure virtual bot
        ii. Configure hooks
        iii. Create goals
        iv. Containerize
    b. Static analysis
        i. Antipatterns
        ii. Explain compiler output
        iii. Shareable antipatterns
    c. UI update
        i. Beautify
        ii. Make better use of space
        iii. More intuitive/descriptive user experience
3. Finish prototype
    a. Add page for dynamic analysis
    b. Combine data into a dashboard
4. Get student/professor feedback
    a. Watch how students use the product
    b. Interview users
5. Incorporate feedback
    a. Add features
    b. Modify user flow
    c. Jump back to step 4
6. Final deliverable
    a. Clean up UI
    b. Finish testing

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

1. The following functions of the previous project will work with zero critical errors (Errors that prevent the program from continuing)
    a. Create instructor account
    b. Login as an instructor
    c. Create an assignment as an instructor
    d. Create antipatterns as an instructor
    e. Upload unit tests as an instructor
    f. Upload project as a student
    g. Identify antipatterns in uploaded code based on antipatterns for the assignment
    h. Identify runtime and compile time errors based on tests uploaded for the assignment
2. Code will run on the virtual CyBot without critical errors (Errors that prevent the program from continuing)
3. Virtual CyBot results will be returned to and parsed by the program
4. Create at least one antipattern relevant to each CPR E 2880 lab
5. Increase user satisfaction of UI based on survey of select CPR E 2880 students
    a. Have a user and professor satisfaction of over 80%
    b. Determinant by polling of students and instructors

3.4 Project Timeline/Schedule

sdmay25-23	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9
Get previous project running									
Fix Errors									
Fix pipeline									
Update strings for current team									
Gain familiarity with Regex and XPath									
Interview Users to define areas of improvement									
Dynamic Analysis									
Configure virtual bot									
Create hooks									
Create goals									
Static Analysis									
Antipatterns									
Explain compiler output									
Shareable Antipatterns									
UI Update									
Beautify									
Make better use of empty space.									
More intuitive/descriptive user experience									
Finish Prototype									
Add page for dynamic analysis									
Combine data into a dashboard									

Fig. 2 Gantt Chart of Project Prototype Timeline

3.5 Risks and Risk Management/Mitigation

- **Get previous project running**
    - Risk: Previous senior design group's project has errors running - 0.8
        - Reasoning: This web project is rather complex with multiple interconnected parts that need to be configured in a rather specific way.
        - Mitigation Strategy: Contact members from the prior group and attempt to get assistance (The team is already in direct contact with a prior member).
- **User requirements gathering**
    - Risk: Unable to contact and interview users - 0.1
    - Risk: User requirements gathering does not produce sufficient patterns to focus development - 0.3
- **Implement Dynamic Analysis**
    - Risk: Team is unable to implement dynamic analysis - 0.5
        - Reasoning: Runtime analysis is an incredibly challenging topic that has a lot of complicated research surrounding it.
        - Mitigation Strategy: Ask advisor if there is any professor at Iowa State University that has expertise in dynamic analysis, reach out to the researchers at Michigan Tech University who developed the system this project is based on, potentially drop this requirement (Worst-case scenario)
    - Risk: Dynamic analysis does not correctly identify errors - 0.4
- **Improve Static Analysis**
    - Risk: Static analysis does not correctly identify errors - 0.4
    - Risk: Static analysis seems a degradation of accuracy and precision - 0.1
- **UI Update**
    - Risk: UI Update worsens application appearance and usability - 0.2
    - Risk: Newly designed pages can not be integrated - 0.1
    - Risk: UI update decreases application accessibility - 0.1
- **Finish Prototype**
    - Risk: Prototype can not be deployed - 0.3
    - Risk: Prototype exhibits never-seen-before problems - 0.7
        - Reasoning: This project is rather complicated with several interconnected pieces of functionality, so there is bound to be issues that the team can not feasibly account for.

- Mitigation Strategy: Prior to deploying a prototype, the team will need to thoroughly do testing and stress tests. Furthermore, user testing feedback will be extremely beneficial to identifying and correcting erroneous problems.

3.6 Personnel Effort Requirements

Task Name	Estimated Person-Hours Required
Get the previous project running	60 Hours
User requirements gathering	20 Hours
Dynamic analysis	80 Hours
Static analysis	120 Hours
UI update	60 Hours
Finish prototype	40 Hours
Get student/professor feedback	Unknown
Incorporate feedback	Unknown
Clean up deliverable	60 Hours
Total	**440 hours + unknown**

Table 1 Estimated Personnel Effort Requirements

Many of these estimates are rather rough. Knowing all the errors and problems still left in the previous program is impossible. The team still needs access to the virtual CyBot for dynamic analysis, so it is difficult to tell how complex the integration will be. On the static analysis side, many members are presently getting up to speed with regex and XPath. Based on what the previous team accomplished over the course of a year, it will be challenging to gauge how long it will take to get lab-specific antipatterns ready. As for the UI update, several team members have some experience developing a UI on this scale. This time estimate is based on a few code jams and projects. Finishing the prototype and cleaning up deliverables will focus on testing, and those times are

based on experience with COM S 3090 and other development classes that the team members have taken. Additionally, the volume and complexity of implementing student and professor feedback are impossible to estimate precisely. It will ultimately rely on the experiences and amount of feedback given at that time.

3.7 Other Resources Requirements

Technical Cost:

- A server from Iowa State's Electronics and Technology Group
- Energy Costs will be minimal
- Uses open-source software
- Iowa State University CyBot boards

Human Cost:

- Staff required to maintain system once completed
    - Either training teaching assistants or a specialist in the Iowa State Electronics and Technology Group
- Whoever enters the antipatterns needs to be knowledgeable of regular expressions

4.0 Design

4.1 Design Context

4.1.1 Broader Context

Area	Description	Effects of C Code Critiquer
Public health, safety, and welfare	How does your project affect the general well-being of various stakeholder groups? These groups may be direct users or may be indirectly affected (e.g., solution is implemented in their communities)	Students will be able to get feedback on their code at any time of day, instead of just when TA's and Instructors are available which could reduce stress. TA's and Instructors are able to get more feedback on common student pitfalls.
Global, cultural, and social	How well does your project reflect the values, practices, and aims of the cultural groups it affects? Groups may include but are not limited to specific communities, nations, professions, workplaces, and ethnic cultures.	This product could be a double-edged sword when it comes to how it will affect the profession of software developers. Because on one hand it can make it easier for students to learn to code and identify mistakes. But on the other hand it could become a crutch to those students, making them overall worse programmers.
Environmental	What environmental impact might your project have? This can include indirect effects such as deforestation or unsustainable practices related to materials manufacture or procurement.	The environmental impact should only be the energy cost of running the server. All the other costs are negligible because it uses hardware that Iowa State already has access to.
Economic	What economic impact might your project have? This can include the financial viability of your product within your team or company, cost to consumers, or broader economic	There are no plans to monetize this product. The technical costs of running this after the initial setup is just the energy costs.

	effects on communities, markets, nations, and other groups.	The human cost is more prevalent. After the product is completed, it will need to be maintained and it may need to be updated if the course changes enough. This would require training an employee or TA to do so.

Table 2 Broader Context Table

4.1.2 Prior Work/Solutions

As the team is continuing the work of a previous senior design team, we were able to begin working through our design with a basic prototype at hand. The previous team was in frequent contact with Michigan Technology University, who had made several code critiques for other languages, which they used as an aid in their development. References to each prior work or solution can be found in section 8.2.

Previous Team's (sdmay24-34) C Code Critiquer [1]:

Advantages: Looking at a functional prototype has allowed us to get a good understanding of how both regex and XPath are used in static analysis. It also gave us a strong foundation to begin building new ideas/changes off of.

Shortcomings: Our team was unfamiliar with Python, which the critiquer is coded in, at the start of the project. It has slowed us down in understanding the code.

Michigan Technology University Critiquers [2]:

Advantages: Michigan Technology University has made multiple critiques for several different languages, with functioning prototypes.

Shortcomings: As prototypes, they are not widely available for use. Also, the critiquers only use static analysis.

We also spent some time looking into code critiquers currently on the market, and found two to compare ours with.

PC-lint Plus [3]:

A commercial command-line tool that performs static analysis of C and C++, indicating issues or concerns in the code.

CodePal [4]:

A platform that provides several helpers and tools to assist developers. When looking into it, we mainly focused on the code review portion of the platform.

Pros of our code critiquer compared to others:

Designed to be used in CPR E2880, our critiquer will be tailored to the course and embedded C programming as a whole. It will be reliable in finding the specific antipatterns that the professor is looking for. The explanations provided by the critiquer will be beginner friendly.

Cons of our code critiquer compared to others:

It requires more bespoke knowledge of its inner workings to maintain and a deeper understanding of regular expressions to add more antipatterns to the system.

4.1.3 Technical Complexity

The team's design will expand upon the previous team's (sdmay24-34) code significantly. It is planned to add more antipatterns to the system based on what is being covered in the CPR E 2880 labs and the most common issues encountered in those assignments. This section of the project focuses substantially on leveraging XPath and regular expression technologies to represent and find antipatterns. Furthermore, the expanded analysis features will include the introduction of dynamic analysis, which comes with unique engineering and mathematical challenges. Uploaded code will be run through a virtualized CyBot system, simulating a lab runtime environment, which is what the dynamic analysis portion of the critiquer will use to generate additional feedback.

The team is also adding a feedback page that the instructors and teaching assistants can use to see the most common antipatterns for each lab, among other statistics. This new functionality, along with the desired user interface overhaul, will require deep knowledge of frontend development, such as managing the router, writing accessible HTML and CSS, and hooking in third-party libraries for displaying charts.

Overall, this project is rather technically complex, with frontend and backend development skills requirements, static code analysis, dynamic code analysis, and connecting with external software and hardware. Each design section has unique challenges that must be overcome to ensure a successful outcome.

4.2 Design Exploration

4.2.1 Design Decisions

1. The first significant design decision we had to make was if we wanted to reuse the previous team's code. Remaking it would allow us more flexibility and make it so we can develop the program with methods or languages that we are more familiar with. However, we decided against this, primarily because having a mostly working baseline allows us to spend more time on the product-adjacent tasks. Such as integration into other systems or further functionality, without us having to spend time remaking the project's core.
2. Another decision we made was whether or not to include artificial intelligence in some form in this project. This could consist of having a large language model give answers about the 1,400-page long datasheet or having it train on the CPR E 2880 course materials as a whole using something like CourseGPT(https://arxiv.org/pdf/2407.18310). This is still being decided upon between our group and the CPR E 288 professors based on what they want us to include and what is feasible for us to accomplish in the allotted time.
3. Another design decision that we are still deciding upon is if we want to revise how the assignments are handled. The way it is currently being handled is that the instructor will make an assignment, and then they will have to share the assignment code with their students. However, we have discussed changing it so the instructor can make class sections and add each student. Then, the students would log in to their accounts and be added to the proper section. This is an important decision because the highest priority behind the product working is having the product be something that the instructors would want to use. And if it is too inconvenient to set up the assignments, they won't do it.

4.2.2 Ideation

The decision to stick with the current code came with several potential options. Our team as a whole had little experience in python (which the previous team used) and jumping into a relatively complete project could be difficult.

Our first option was to stick with the previous team's work and spend some time getting to know the code. Even if we are unfamiliar with python, it shouldn't take long to gain a decent understanding of it.

Our second option was to build a new project from scratch using C, the language being critiqued by the system. This would theoretically make it easier to test the code being uploaded by students.

Our third option was to build a new project from scratch using Java, the language we were most familiar with. It wouldn't be as easy to critique C code, but the structure of the project would be more easy to build.

Our fourth option was to scrap the idea of a web application and spend our time making a plugin for Code Composer Studio, the IDE being used by the CPR E 2880 lab. This would mean that the students could directly critique the code without uploading it through a browser.

The final option we considered was making a plugin for Visual Studio Code, and having the CPR E 2880 lab switch to using it instead of Code Composer Studio. This is the most difficult option currently as it involves changing the lab structure.

4.2.3 Decision-Making and Trade-Off

We identified the pros and cons of the ideated approaches throughout a few meetings by first making a list of the options. Everyone brought forth their opinion, and we discussed which option would be easy to jump into, develop in a timely manner, and produce satisfactory results. With our client, Dr. Rover, wanting to see a prototype in the Iowa State Spring Semester, the option with the least risk came down to sticking with the current code. Starting fresh with a different language would be challenging to complete in time, considering we would both need to understand the project and create it all in one semester. While both the Code Composer Studio option and the Visual Studio Code option were appealing initially, they slowly lost favor after discussions continued with Dr. Rover and Dr. Jones (the other professor teaching CPR E 2880), and the project design continued being refined. Code Composer Studio is undergoing an overhaul in December 2024, so it is still being determined what that software will look like throughout development. Switching to Visual Studio Code would

create some risk, as it has yet to be used in the CPR E 2880 labs. As such, we have decided to work on the current code and add to the previous team's design, especially as it seems like the natural evolution of this project. However, we may work some aspects of the other options into the project in the future, depending on talks with our client.

4.3 Proposed Design

4.3.1 Overview

Our current design is a program that will be run on a server. Students and instructors will be able to access this server via a website. Instructors will be able to upload a list of student Net-IDs for student account creation, create assignments for the students, with specified parameters and patterns it wants to look out for. Students will receive an email once their account has been created with a generated password for them to login. Once logged in, students will be able to click on an assignment and upload code. The program will then search the students' code for any anti-patterns or other ineffective coding practices in their code and give them feedback on it. The server will store this feedback so the instructor will be able to look at the data later.

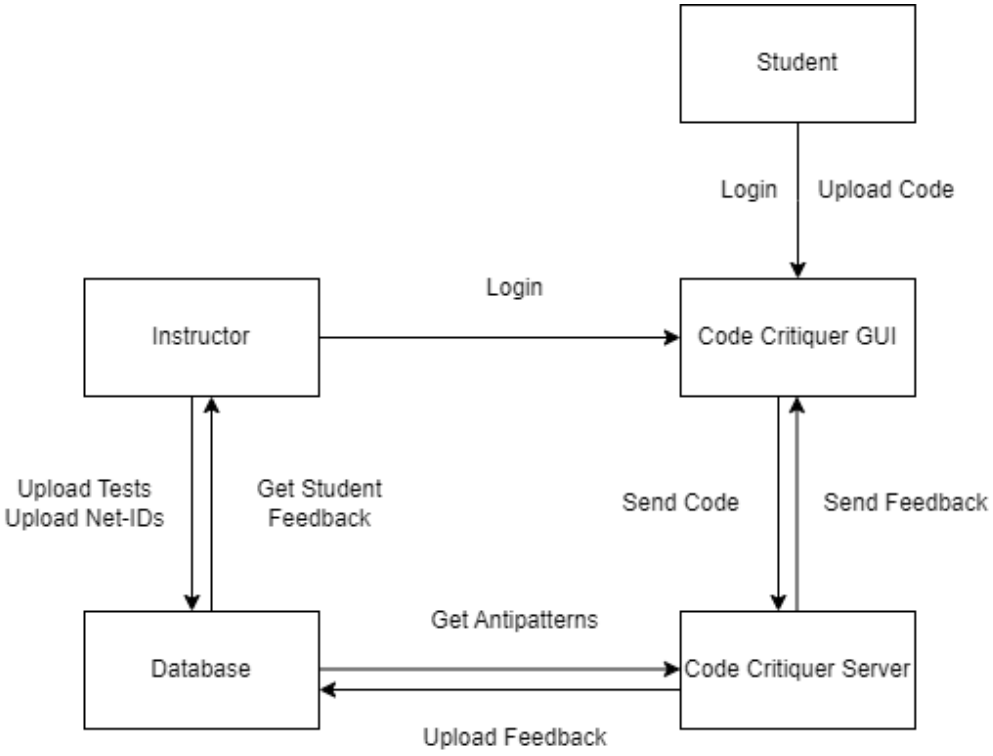

Fig. 3 System Overview Sketch

4.3.2 Detailed Design and Visual(s)

The project's design is a comprehensive web application that has a frontend system of web pages that users will access to communicate with the backend database server that stores account, assignment, and analysis information.

Additionally, there is also a critiquer server that is used to sandbox and analyze uploaded student code.

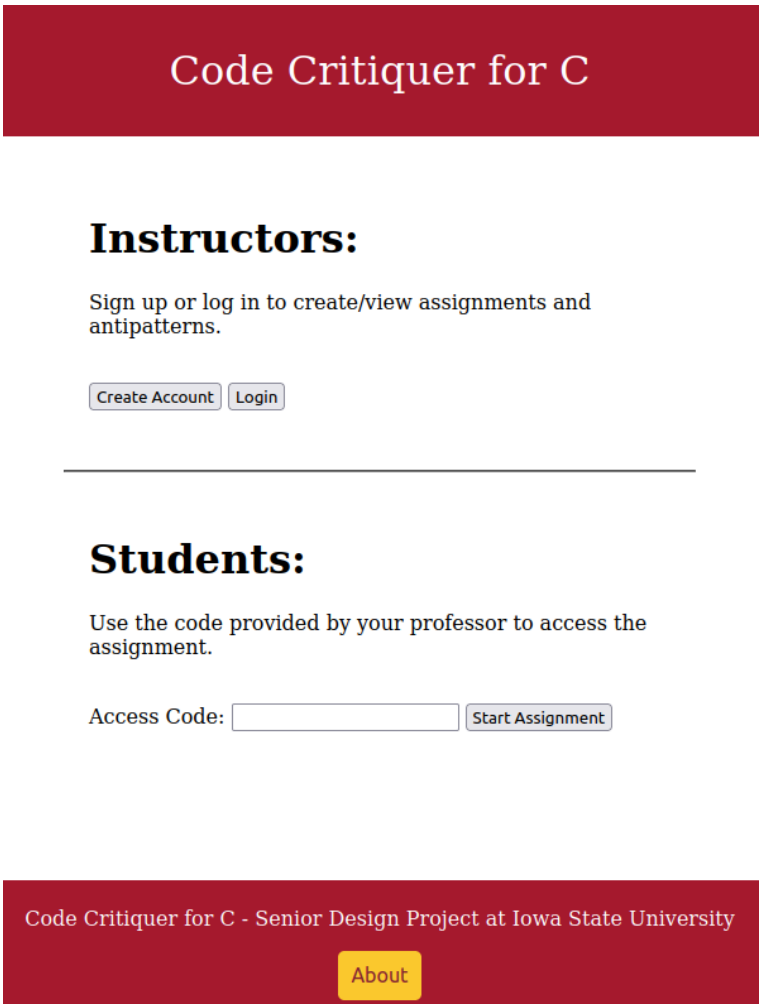

Fig. 4 The Web Application's Landing Page

Frontend Graphical User Interface:

The frontend web application graphical user interface is built using HTML, CSS, JavaScript, Templates (driven by Python code), and Flask so that multiple users can access it simultaneously and be able to navigate pages with the correct information as expected. The frontend GUI receives and displays the feedback from the critiquer server after a student submits code to be analyzed.

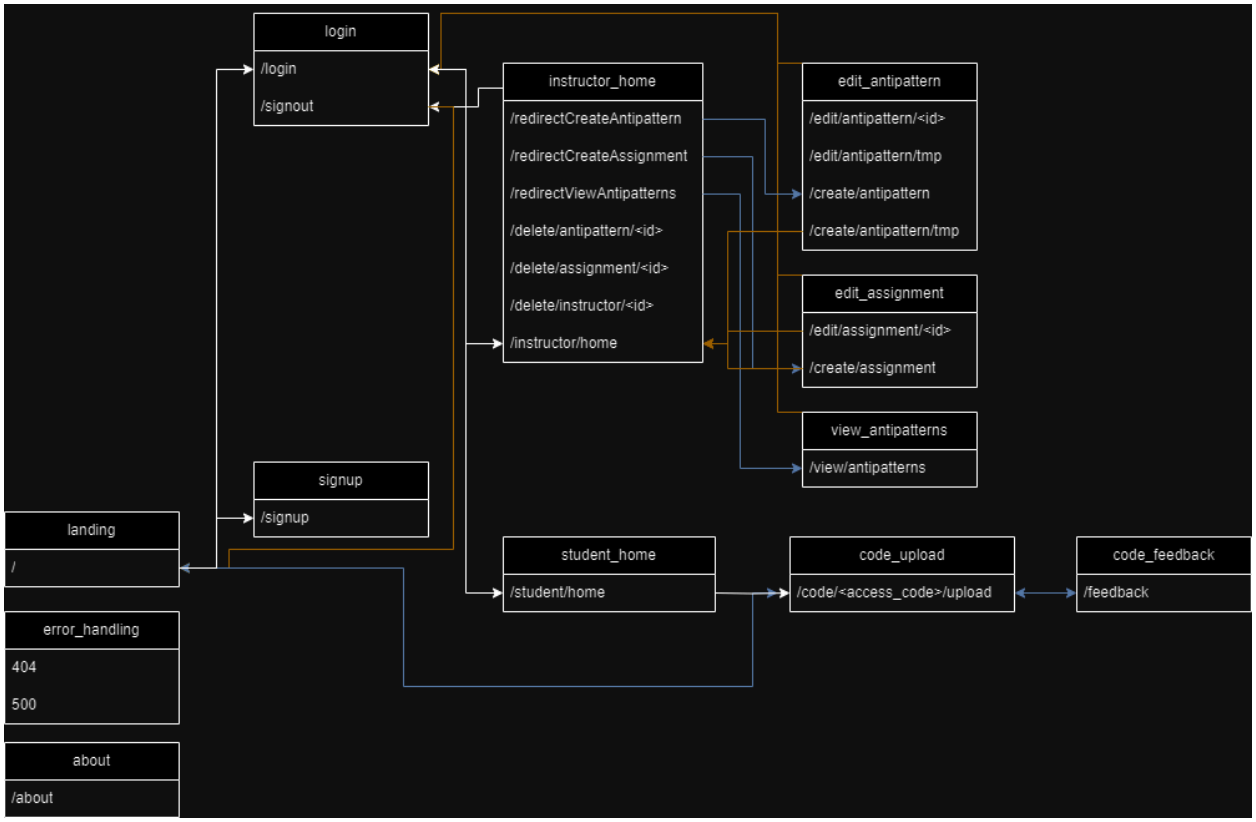

Fig. 5 Page Flow Diagram

Above is a diagram of the page flow through the GUI, as experienced by a user. It depicts how each page in the system is connected and how it can be accessed. Additionally, the design will incorporate pages for assignment analytics and managing accounts. The routing of pages in the web application is handled through Flask.

Backend Database:

The backend server stores all of the information regarding the operations of the system and it will be hosted in a single location, serving each of the users connecting using the frontend web application. The database facilitates the logging in and registering of users while also storing necessary data for operation, such as antipatterns and analysis feedback. Students will be able to submit their code using the frontend GUI to the database, which will then be sent to the critiquer server for running. The database will then receive feedback information from the critiquer once the tests have finished running. It will be maintained using MySQL with the following tables to store the information.

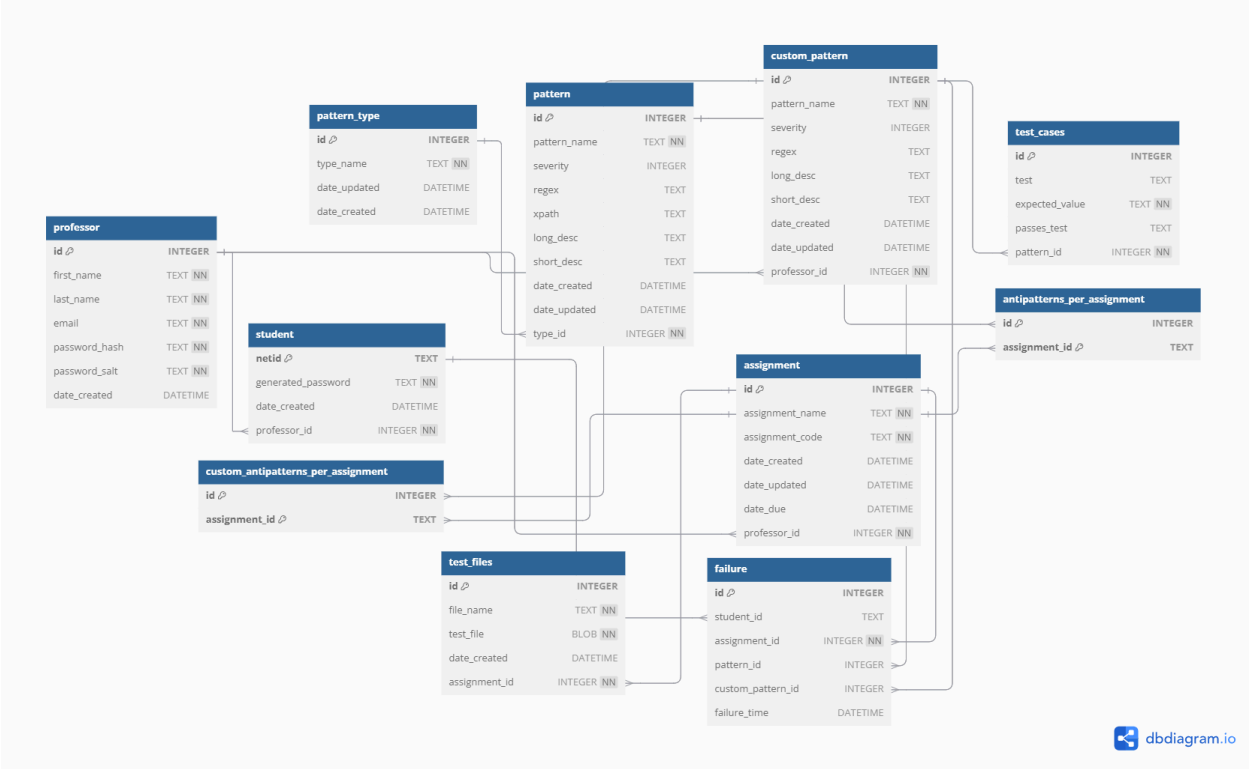

Fig. 6 Database Table Diagram

Furthermore, additional tables and data will be added to represent student and teaching assistant accounts, course sections, and assignment analytics.

Critiquer Server:

Additionally, there is a critiquer server in the project that houses the sandbox and C critiquer written in Python that is used for testing uploaded student code, generating feedback based on analysis, and getting antipatterns for testing from the database. This critiquer will be expanded upon to implement dynamic analysis using a virtualized CyBot system. The virtualized CyBot is a program maintained by Iowa State's Electronics and Technology Group that uses code combined with CyBot boards used in the CPR E 2880 course. The critiquer system will optionally (as determined by the user) run dynamic analysis testing by sending the uploaded C code files to the virtualized CyBot, which then runs it on its testing boards. The virtualized system will then send feedback to the critiquer, which it will parse, generate feedback, and display to the user.

4.3.3 Functionality

The project's design has three user groups in mind, each of which will interact with the system in slightly different ways.

1) CPR E 2880 Students

In the system, the CPR E 2880 student user group will be able to create an account, log into the web application, select a specific assignment, and submit code to be analyzed . Once submitted to the system by a student user, their code will be analyzed and feedback generated, which the student can then use to learn and improve their programming skills.

2) CPR E 2880 Professors

The CPR E 2880 professor user group will be able to sign up for an account on the system's web application, and from there, they will be able to set up user accounts for teaching assistants, create, edit, or delete assignments, view an access code for each created assignment, assign students to various sections, edit or add antipatterns to test for, and view analytics from student submissions. Each of these pages will be hosted through the web application, but all of the user, assignment, and class information will be hosted on the backend server. Additionally, instructors can view statistics and telemetry about students' critiquer submissions at either an assignment or global level. There will be a unique page within the teacher login section to reach this functionality, and it will be exclusive to instructor accounts. The statistics will be presented numerically and visually through bar charts, which can be downloaded for external applications or usages. This data will aid instructors in determining problem areas for students, allowing them to focus on concepts prone to causing struggles, improving the CPR E 2880 class overall.

3) CPR E 2880 Teaching Assistants

The teaching assistants interacting with the system will utilize it in a similar way to the professors. The teaching assistants will be able to log into the web application using an account provided to them by a professor, and they will also be able to view student analytics and edit assignments. However, they will not be able to create or delete assignments or edit student account assignments.

4.3.4 Areas of Concern and Development

Once everything is implemented, the design will satisfy the project requirements and user needs. As the tool is used, test runs are executed, and a more detailed user feedback period is completed, the team will have a greater understanding of what users are specifically looking for and would like changed with the design. First and foremost, the design should allow students to receive feedback on their code that will enable them to learn and grow as programmers while also giving instructors the resources and data to improve the CPR E 2880 course, and the design currently meets that ideal. Specifics and preferences will evolve and change as the project develops and more user feedback is received, but the design will be adapted to best fulfill and meet the user's needs.

4.4 Technology Considerations

For the most part, the team continues to build off the designs and technological considerations of the prior senior design group, sdmay23-34. This stance means that in the current design stage, the frontend web application is made using HTML, CSS, and JavaScript with Flask as a routing solution, and the backend server is maintained using MySQL. Additionally, the critiquer system itself is built using the ones designed at Michigan Tech University as a starting point, but it is using Python instead of the language it is analyzing, which is C in this case.

While the team has discussed moving toward more sophisticated solutions, such as a frontend framework, the primary goal of the design currently is to build off the prior work and create a working prototype. Additionally, there were concerns about working on the Python critiquer since many of the team members had little to no experience with the language. However, translating it to a different technology was ultimately decided against since it would require rewriting the system while using precious time. Additionally, since Python is one of the easier to learn languages for programmers, the team members can pick it up quickly while also learning a sought after skill.

4.5 Design Analysis

Presently, the team has gotten the system designed by the previous senior design team, sdmay24-34, running locally and has fixed several issues with the codebase to get it operational and stable. Team members have also begun implementing and experimenting with the newly designed functionality, such as allowing instructors to view analytics about the assignments and common problems that students are encountering. Currently, the project is not in a place where testing a user experience or judging design decisions would be beneficial, but the development timeline is on schedule. Looking toward the future, the team will continue to work on and implement the functionality of different design aspects not found in the prior senior design group's project, with student logins, improved file uploading, and dynamic analysis as a few examples. With the current state of the project's development, there are no identified feasibility issues with the design, but the team will continuously be investigating the feasibility and potential pitfalls throughout the task's growth.

5.0 Testing

5.1 Unit Testing

For automated unit testing, two main components need to be checked: regex/XPath antipatterns and unit tests for dynamic testing. For the regex/XPath antipatterns, multiple files will be uploaded, and the resulting found antipatterns will be checked against the expected list. Dynamic testing will use previous students lab solutions, after professor review, to check that all tests pass correctly. After that, modifications will be made to the lab to check that each test fails individually, and a random combination of tests will be checked to assure that they fail in the correct order. Of note, an error such as two mains will cause the program not to compile, so runtime errors will not be checked.

5.2 Interface Testing

As mentioned above, site flow will be manually tested to ensure correctness. Furthermore, the user interface will also be compared against common internet standards for accessibility and usability, ensuring that our project can be used by a wide audience conveniently.

5.3 Integration Testing

Example files will be sent through the API on the website to make sure that both the static and dynamic analysis portions are functioning properly. For static analysis, it will check that it only gives results for enabled and triggered antipatterns. Tests regarding dynamic analysis will make sure that the docker container is run properly, the makefile compiles properly, and results get parsed and passed back to the website.

5.4 System Testing

For the system test, the following steps will be taken:

1. Create a new professor
2. Login as the professor
    a. Incorrect password fails
    b. Incorrect email fails
    c. Correct password succeeds

3. Create a new antipattern
4. Create a new assignment
5. Select specific antipatterns for the new assignment
6. Upload a unit test for the new assignment
7. Upload example code to the new assignment
    a. Bad file format fails gracefully
    b. Correct antipatterns are checked (including the new one)
    c. Correct tests pass and fail

5.5 Regression Testing

To ensure that features don't regress, a pipeline will be run whenever code is merged into the main branch. This pipeline will fail if any of the aforementioned tests fail. This process happens in the project's GitLab repository, ensuring that any code a team member pushes is tested this way.

5.6 Acceptance Testing

For acceptance testing, the team is in close and frequent communication with the project advisor/client and other faculty or teaching assistants closely related to the CPR E 2880 course. The team meets weekly with Dr. Diane Rover, the project advisor and one of the instructors for CPR E 2880, to discuss the design and its effectiveness, allowing for continuous evaluation, testing, and feasibility assessment. Furthermore, in the coming Iowa State Spring Semester, a prototype of the critiquer system will be used as an optional resource, allowing students to test the design first-hand, provide feedback, and generate usage telemetry, which will be invaluable for reflecting on and refining the design. Continuous communication with CPR E 2880 teaching assistants, instructors, and students while also supplying a prototype for feedback will allow the team to test the design's acceptance rigorously, ensuring that the functional and non-functional requirements are being fulfilled satisfactorily.

5.7 Security Testing

Since user data is being stored for this project, security is an utmost concern. First, SQL queries are being sanitised to ensure that injection can not be possible. Pip will be used to scan for out-of-date and vulnerable libraries. Nmap and burpsuite will scan for simple web exploits. Aside from that baseline testing, cybersecurity students

and professionals will have a chance to review and analyse the project for vulnerabilities. This process will happen after every major version update.

5.8 Results

Certain previous testing is currently on hold as the primary server is inaccessible and an overhaul of site flow is underway. As the next major version comes out and testing gets updated, this section will be populated. Regardless, testing is still a critical part of the team's software development process, as it provides validation and verification of the system and the design.

6.0 Implementation

Thus far, the team has made decent progress toward getting an Iowa State Spring Semester prototype completed by laying the foundations for several features that will be implemented. Additionally, as this is a continuation of team sdmay23-34's project, a good part of this project's first phase was getting oriented with their work and applying some bug fixes to ensure it runs properly and in a stable environment. Most of the work completed during this first half was research, familiarization, and planning, but a few tangible things were implemented to showcase.

A Docker container has been set up and implemented into the system. This implementation allows each test run of the code to be sandboxed into a predetermined and identical environment, reducing the possibility of erroneous occurrences or errors. Perceptually, there is no change from the user's view as this is exclusively a backend and non-visual improvement to the project.

Instructors are now able to upload files containing lists of student Net-IDs which will then automatically be turned into accounts for students to use for login. A student home page has been created that lists all assignments their professor has created with redirects to the code upload pages, but the page is still a work in progress in terms of UI design and functionality. Also, the students currently have a password automatically generated for them but we haven’t set up the automatic sending of those passwords to students.

There has also been some work on updating information on the web application to align more with the project's current state and its developers. This alteration can be seen in Fig. 7 below with a comparison of how the About page has changed.

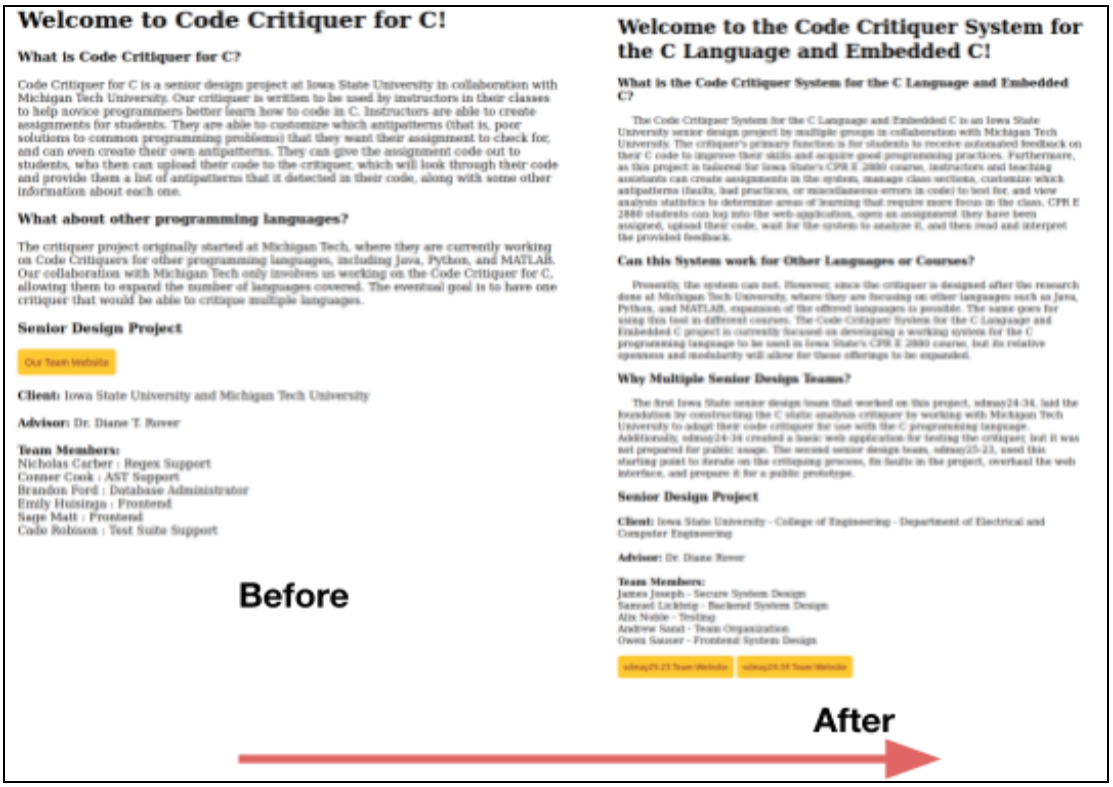

Fig. 7 A before and after comparison of the About page

Furthermore, work was begun on the instructor statistics and telemetry page with some testing of a third-party library to display a bar chart with data, but there is, unfortunately, nothing to show presently.

7.0 Ethics and Professional Responsibility

7.1 Areas of Professional Responsibility/Codes of Ethics

Area of Responsibility	**Definition**	**Related NSPE Code of Ethics**	**How Our Team Interacts with this Area of Responsibility**
Work Competence	Completing your work with a professional level of competency and timeliness.	II.2.a: Engineers shall undertake assignments only when qualified by education or experience in the specific technical fields involved.	When we make a feature we try to make sure it is thoroughly tested before pushing it to main.
Financial Responsibility	Being conscientious about the costs required to not just launch but maintain the project.	II.4.c: Engineers shall not solicit or accept financial or other valuable consideration, directly or indirectly, from outside agents in connection with the work for which they are responsible.	We are keeping the costs in mind while developing. While the upfront costs might be low, the cost to maintain and expand upon the project in the future might add up.
Communication Honesty	Being realistic about the capabilities of the team and the product to the stakeholders/users.	III.3: Engineers shall avoid all conduct or practice that deceives the public.	We are staying honest with our client about our progress and regularly sending them progress reports.
Health, Safety, Well-Being	To put the wellbeing of mankind above every other aspect	III.2.a: Engineers are encouraged to participate in civic affairs; career	We plan to have the project to make the website compatible with text readers

	of the product or project.	guidance for youths; and work for the advancement of the safety, health, and well-being of their community.	for the visually impaired.
Property Ownership	Handling user-provided materials and information with due care and respect.	I.4: Act for each employer or client as faithful agents or trustees.	We are handling user data with care. Making sure that the TA and instructor accounts are secure to make sure students can't cheat.
Sustainability	Being conscientious of the impact the project can have on the environment, both directly and indirectly.	III.2.d: Engineers are encouraged to adhere to the principles of sustainable development in order to protect the environment for future generations.	The only thing for sustainability we have to account for is the energy cost of the server because all the hardware we use is stuff that ISU already has.
Social Responsibility	Making sure that the product is a net good for society without negatively impacting some.	II.1.e: Engineers shall not aid or abet the unlawful practice of engineering by a person or firm.	We want to make sure that students are using the product as a tool to supplement learning instead of something to replace their learning.

Table 3 Area of Professional Responsibility/Codes of Ethics

Overall, the team is performing well in the property ownership category, especially given the number of parties involved with the project. Since the critiquer will need access to CPR E 2880 lab answers, the team is ensuring that only the appropriate people have access to those materials, as it could be damaging if they were publicly

released. Furthermore, CyBot-specific information is also carefully handled by keeping it contained within the scope of the critiquer and virtualized CyBot environment. The team communicates frequently with and receives feedback from the project client, ensuring that it is known which information is considered sensitive. The principle of least privilege and access is heavily utilized to ensure that only the correct users have access to only the subset of information that they need.

The area of professional responsibility that our team could improve on is work competence. Due to some unforeseen issues with the previous team's code we started to fall behind on our deadlines. And once we fell behind the deadlines we never adjusted them to account for this, so we fell even further behind. In the future we will be talking about our progress and how we are keeping up with the deadlines and adjusting them accordingly.

7.2 Four Principles

	Beneficence	Nonmaleficen-ce	Respect for Autonomy	Justice
Public health, safety, and welfare	Helps improve stress levels of students and TAs	Aim to reduce student reliance on critiquer	Allow students to use as much as needed	Provides a resource for busy students
Global, cultural, and social	Addresses the needs of instructors and TAs	Available for all students	Respects cultural practices	All user types benefit from implementatio n
Environmental	Design uses existing hardware and will receive energy from ISU	Design doesn't require more manufacturing	Will provide user choice between just static analysis or both static and dynamic (uses boards)	Allow students off campus to access CyBot without producing more
Economic	Design is free for students	Design would not be	Will allow users to use	Free resource for people

		disruptive	only static analysis if they do not have access to a server with boards	who can't attend office hours

Table 4 Four Principles Table

One broader context-principle pair that is important to our project is public health, safety, and welfare paired with beneficence. A main goal of the critiquer is to provide timely feedback to students. This ties directly to improving stress levels as both students and TAs will need to spend less time going over basic code issues that could be solved without the process of office hours or a simple hand raise.

A broader context-principle pair that our project is lacking a little in is economics. While our design has a minimal cost impact, it is rather difficult to measure or predict, given that it is highly dependent on Iowa State University resources. To better accommodate for this, the team will need to investigate maintenance costs more closely from different perspectives, such as how the Electronics and Technologies Group at Iowa State handles maintaining proprietary software and what they protocols for that are.

7.3 Virtues

Three virtues that are important to our team are:

- Cooperativeness: a willingness and ability to work with others
    - Cooperativeness is very important in making sure our team can make progress on the project without getting in each other's way. We will communicate frequently and make it clear what we are currently working on so as to not create confusion.
- Clear and thorough documentation: detailed documentation that avoids unnecessary complexity and covers all aspects of the project.
    - We will document the changes we have made to the previous team's work and provide clear instructions on how to use the critiquer.
- Honesty: being truthful and communicating openly
    - As a team of individuals with our own ideas and opinions, we will continue to openly express them. If someone disagrees with a decision

an individual has made, they will bring their thoughts forth, not simply sit back and accept the decision without any discourse.

Individual Reflections:

- James Joseph
    - One important virtue that I feel I've demonstrated best is communication. With our team, I feel like I've been able to express my ideas and thoughts around the direction of the project, and I've been able to listen to other people's thoughts about direction. While the more cooperative nature of our communication has led to more overhead, I believe that it has been worth it to bring out the best idea of what the project should be
    - The most important virtue to me is work competence. I feel that projects are nothing without a good result. My biggest effort will be towards making the product functional and worth the user's time. Even with this, I have felt that my performance this semester has been a little lackluster as I've waffled too much with the direction and task breakdowns to tackle. Coming into the next semester, I will lay out a more definitive direction and focus on developing and completing a backlog.
- Samuel Lickteig
    - A virtue I feel I have demonstrated thus far is responsiveness. Responding quickly to team members helps their work flow more smoothly. I try my best to respond to the Discord as quickly as possible so as to not potentially delay their work.
    - A virtue I need to work on is decisiveness. It is important to me because there are many small decisions that need to be made when working on a large project like this. Spending too much time questioning my own thoughts results in me putting more time than necessary into what should be small decisions. I will try to demonstrate this by having a more structured means of comparing options that results in a quicker resolution.
- Alix Noble
    - A virtue I feel I have demonstrated well is civic-mindedness. Always keeping the end users and our clients in mind is important to the project's success, as they will ultimately be the ones who use the system. I feel like I always keep in mind how decisions will impact the user experience. Whenever I am making a decision about something in the project, I am cognizant of how that will ripple into the usability and social usage of the

tool.
    - A virtue I want to demonstrate more of is clear and thorough documentation. A lot of the project is lacking documentation or code comments about how different parts of it work. Demonstrating this virtue more by writing documentation about the things I work on and commenting code will go a long way toward our goal of modularity and ease of maintenance.
- Andrew Sand
    - A virtue I feel I have demonstrated in my work throughout the project thus far is honesty. I do my best to be as honest as possible when communicating with others and doing work for this project, as it ensures that everyone is in agreement regarding a topic, concept, or idea. I am never intentionally dishonest about information or work that needs to be communicated. Honesty is extremely important for this project due to the number of parties involved with its creation and who will eventually use the finished version of it in the future. It is critical to be honest and truthful with what the project can and can not do, and each party needs to trust each other for this to be successful.
    - A virtue I wish to work on in the upcoming second half of this project's development is my willingness for self-sacrifice. Throughout the project's first phase, I was not particularly happy with the amount of time I dedicated to working on it, and I feel that there needs to be more on my end to match the other member's progress. There must be a delicate balance of willingness for self-sacrifice on the project, not too much but not too little. In the future, I wish to dedicate more of my time toward working on the project to not let the team down and ensure that I am putting in my fair amount of effort to complete the project.
- Owen Sauser
    - The virtue that I have demonstrated is the virtue of commitment to the public good. I have always made preventing cheating with our product a priority. Doing things like suggesting a way to disable the tool during exams or logging the submissions. Then also we want the students to use our product to supplement their learning, not to replace it.
    - The virtue that is important to me that I have not fully demonstrated thus far is timeliness. This is because I had trouble keeping up with the deadlines we had set. While some of this is due to factors I am not able to control, there were other times when it was definitely a lack of motivation

on my part. And in the upcoming semester I will be working to prevent this in the future by setting aside more time for the project

8.0 Closing Material

8.1 Conclusion

For the start of the project, the team's primary goals have been to get familiarized with the work accomplished by the prior group (sdmay24-34), research the technologies and concepts involved, determine a list of features and improvements to implement and begin preparing for a prototype for the Iowa State Spring Semester. The team has successfully obtained these goals to a satisfactory degree. In total, bug fixes have been implemented, plans have been laid for full development to begin in the coming months, each member has a much better understanding of the topics involved, a Docker container has been set up for sandboxed execution, a statistics page is in development, and the project is well on its way to being in a suitable shape for trial runs early 2025. A big part of the team's success thus far has been constant communication between members but also with the project advisor, and other third parties related to the CPR E 2880 Iowa State course. Moving into the project's next phase, the team aims to overhaul the user interface, implement working dynamic analysis, finish the statistics page, implement a better account system, and set up more robust antipatterns that can be used with CPR E 2880 labs.

8.2 References

[1] *Code Critiquer System for the C Language*. [Online]. Ames, IA: Iowa State University, 2024. Available: https://sdmay24-34.sd.ece.iastate.edu/.

[2] Albrant, L., Pendse, P., Dasker, D., Brown, L., Sticklen, J., Jarvie-Eggart, M. E., & Ureel, L. C. (2024). Work-in-Progress: Python Code Critiquer, a Machine Learning Approach. Proceedings - Frontiers in Education Conference, FIE. http://doi.org/10.1109/FIE58773.2023.10343017.

[3] *PC-lint Plus 2.2*. [Online]. Stuttgart, Germany: Vector Informatik GmbH, 2024. Available: https://pclintplus.com/.

[4] *CodePal C Code Reviewer*. [Online]. Nigeria: CodePal, 2024. Available: https://codepal.ai/code-reviewer/c.

[5] Ureel, L. C., Brown, L., Sticklen, J., Jarvie-Eggart, M. E., and Benjamin, M., "Work in Progress: The RICA Project: Rich, Immediate Critique of Antipatterns in Student Code," in Proceedings of the 6th Educational Data Mining in Computer Science Education (CSEDM), 2022, 75-81. http://doi.org/10.5281/zenodo.6983498.

[6] Ureel, L. C., and Wallace, C., "Automated Critique of Early Programming Antipatterns," in SIGCSE '19: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, 2022, 738-744. http://doi.org/10.1145/3287324.3287463.

8.3 Appendices

8.3.1 A. Operations Manual

Create .env File

1. Open a python shell
2. Run import secrets and secrets.token_hex()
3. Create a .env file in the root directory of the project
4. Enter the following into the file:

```
ENVIRONMENT="qa"
FLASK_SECRET_KEY=[secrets.token_hex() result here]
FLASK_DATABASE=sqlite
```

How to Run

1. Create .env file if you haven't already
2. Open terminal in project directory
3. Install the requirements: python -m pip install -r requirements.txt
    a. Update path variable for libclang library within venv/clang/cidinex.py
    b. Update class variable library_path of Config to library_path = os.path.join(os.path.dirname(os.path.realpath(__file__)), 'native')
4. Initialize the database: python -m flask --app . init-db
5. Install Docker
    a. Windows: Install Docker Desktop and make sure it is running while the code runs
    b. Linux: Install docker with your package manager (Eg. sudo apt install docker)
6. Start the application: python -m flask --app . run --port 8000 --debug

7. Open localhost:8000 in your browser

Accessing the database in the code

1. Import database: from database import get_db
2. Example for fetching data:

```
db = get_db()
results = db.execute(
    "SELECT long_desc, short_desc FROM pattern"
).fetchall()
print(results[0]['long_desc'])
```

3. Example for changing the database:

```
db = get_db()
db.execute(
    "INSERT INTO pattern (long_desc, short_desc) VALUES (?, ?)",
    ("long description", "short description"),
)
db.commit()
```

8.3.2 B. Code

All code for the project can be found in the team's GitLab repository: https://git.ece.iastate.edu/sd/sdmay25-23

9.0 Team

9.1 Team Members

- James Joseph
- Samuel Lickteig
- Alix Noble
- Andrew Sand
- Owen Sauser

9.2 Required Skill Sets for the Project

- Knowledge of the C programming language - Required for CPR E 2880
- Knowledge of embedded programming - Required for CPR E 2880
- Frontend Web Development - Required for the frontend web application
- Backend Database Development - Required for the backend database and critiquer
- Data Security and Best Storage Methods - Required for storing user data in an appropriate way
- Knowledge of different representations of code - Required for finding antipatterns in code
- Antipattern checking mechanisms - Required for finding antipatterns in code

9.3 Skill Sets Covered by the Team

- Knowledge of the C programming language - Andrew, Owen, Samuel
- Knowledge of embedded programming - James, Owen, Samuel
- Frontend Web Development - Alix, Andrew, Samuel
- Backend Database Development - Alix, Samuel
- Data Security and Best Storage Methods - James, Owen
- Knowledge of different representations of code - James, Owen
- Antipattern checking mechanisms - Andrew, James, Samuel

9.4 Project Management Style Adopted by the Team

The team adopted a hybrid management style between Waterfall and Agile for this project, fitting the specific project, team, and work schedule. The overarching

project is split into Waterfall-like tasks, each needing to be completed before the next, but these tasks are worked on in an Agile-like sprint. Weekly, the team meets to share progress, discuss the project's direction, and plan for the next "Sprint" (week). The team's hybrid approach also draws inspiration from Waterfall's larger tasks that must be sequentially completed before the next can be worked on, as seen in the project's task decomposition chart (See Fig. 1). This combination of the two project management styles has served the team well, striking a balance between responsiveness and a structured work order that suits this project.

9.5 Initial Project Management Roles

- James Joseph - Secure System Design, CPR E 2880 Liaison
- Samuel Lickteig - Backend System Design
- Alix Noble - Testing
- Andrew Sand - Team Organization, CPR E 2880 Liaison
- Owen Sauser - Client Interaction, Frontend System Design, CPR E 2880 Liaison

9.6 Team Contract

Team Members:

1) James Joseph 2) Samuel Lickteig
3) Alix Noble 4) Andrew Sand
5) Owen Sauser 6) _____
7) _____ 8) _____

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:
   Monday 2:00–3:00 PM, virtual through Discord
2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):
   a. Discord group chat for member-member communication in addition to updates, reminders, scheduling, etc.
   b. Email for member-advisor communication
3. Decision-making policy (e.g., consensus, majority vote):
   Three or more members must be in complete agreement with the decision before moving forward.

4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):

   In our Google Drive, which is shared with all group members, there will be a folder that contains summaries of each meeting. This archive will always be available to all members. At least one member present should contribute to the summary. The contributing members will vary based on meeting attendance.

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:
   a. Inability to participate in a meeting or tardiness of more than 15 minutes must be communicated at least an hour before the meeting's scheduled start time.
   b. Members are expected to attend and participate in all advisor meetings unless communicated beforehand.
   c. Members are expected to attend and participate in all weekly meetings unless communicated beforehand.
   d. Additional meetings and their expectations will be discussed on a case-by-case basis.
2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:

   Each team member is responsible for setting and meeting deadlines on the tasks they are responsible for. If a task requires or relies on multiple team members, a discussion will be held to determine an appropriate timeline for each party where the parties will be responsible for their given assignment and deadline.
3. Expected level of communication with other team members:

   Every member of the team must give weekly progress reports that include what was accomplished along with what there is still to do. Any tasks blocking theirs must be communicated immediately in the appropriate communication channel.
4. Expected level of commitment to team decisions and tasks:

   Each team member is expected to be able to complete their tasks in a reasonable amount of time. If they need more time they should

communicate what issues they are experiencing with the rest of the team so we can work together to remedy the situation.

Leadership

1. Leadership roles for each team member:
    a. James Joseph: Secure System Design
    b. Samuel Lickteig: Backend System Design
    c. Alix Noble: Testing
    d. Andrew Sand: Team Organization
    e. Owen Sauser: Client Interaction, Frontend System Design
2. Strategies for supporting and guiding the work of all team members:
    As stated in participation expectations, team meetings will include progress reports that allow us to see each others' progress and help if there are questions or concerns.
3. Strategies for recognizing the contributions of all team members:
    a. Git commits and merge requests
    b. Progress reports
    c. Meeting summaries

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.
    a. Alix Noble: Experienced with front end and back end development and with unit testing in Java and JavaScript. Some experience with database management and web development.
    b. Andrew Sand: Experienced with C and low-level development. Additionally is knowledgeable and has experience with frontend web development. Has experience maintaining documentation for development teams.
    c. James Joseph: Experienced with both web development and cybersecurity. Can bring a security perspective along with their experience in programming to develop the application in a way that will keep users and the university protected. On top of that, they have experience with reverse engineering and low-level programming, so they will have a good understanding of what a program aims to do.

d. Owen Sauser: Experienced with Java, C, C#, and VBA. Has experience in IT and cybersecurity. His current job is translating a company's software from VBA to C#, so he has experience working with other peoples' code and making modifications if necessary.
e. Samuel Lickteig: Experienced with C code and web development, as well as common mistakes a beginner may make. Also have experience in backend development.

2. Strategies for encouraging and supporting contributions and ideas from all team members:

   Make sure each team member has a chance to voice their opinions on each issue, even if a majority already believe we should address it one way.

3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)

   They can send a message in the Discord group chat, or if they do not feel comfortable with that they can go through the senior design instructors or our advisor if needed.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:
   a. Improve upon the initial design and gain a better understanding of the C programming language along with its patterns/anti-patterns.
   b. Improve the security of the initial design.
   c. Create a final product that can be beneficially used in its intended environment.
2. Strategies for planning and assigning individual and team work:
   a. As a group we will try to assign equal amounts of work to each team member such that each of their tasks is best suited for their skillset.
   b. If any task is larger than the others, this should be communicated and the timeline can be adjusted accordingly.
3. Strategies for keeping on task:

   Each team member will report on their progress at the weekly team meeting. If a team member is concerned about keeping up with their work, they can bring it up at a team meeting and the team can find a strategy to move forward.

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?
   Collectively talk to the member who made the infraction to see why the infraction was made and discuss what can be done to prevent further infractions.
2. What will your team do if the infractions continue?
   Communicate with the senior design course instructors and/or our project advisor to determine a course of action and what can be done in the situation.

---

a) I participated in formulating the standards, roles, and procedures as stated in this contract.
b) I understand that I am obligated to abide by these terms and conditions.
c) I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.

1) James Joseph	DATE 9/16/2024
2) Samuel Lickteig	DATE 9/16/2024
3) Alix Noble	DATE 9/16/2024
4) Owen Sauser	DATE 9/16/2024
5) Andrew Sand	DATE 9/16/2024