Code Critiquer System for the C Language and Embedded C

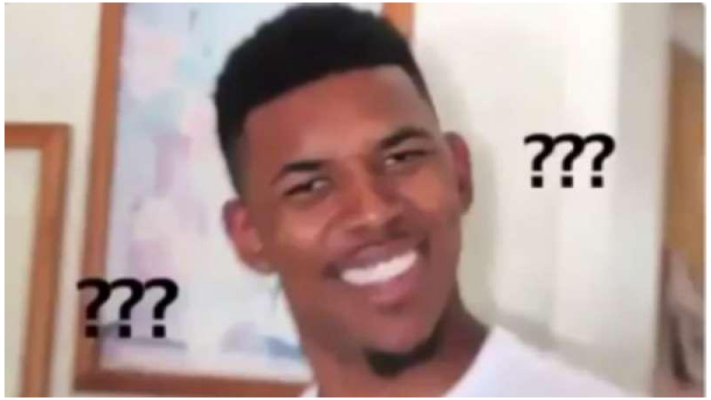
sdmay25-23:

https://sdmay25-23.sd.ece.iastate.edu/

James Joseph
Samuel Lickteig
Owen Sauser
Andrew Sand
Alix Noble

Client and Advisor: Dr. Diane Rover

```
0.705 /usr/bin/ld: /app/build/objs/static_example.o:
in function `main':
0.705 static_example.c:(.text+0x0): multiple
definition of `main';
/app/build/objs/blah.o:blah.c:(.text+0x8c): first
defined here
0.705 /usr/bin/ld: /app/build/objs/static_example.o:
in function `max':
0.705 static_example.c:(.text+0x87): multiple
definition of `max';
/app/build/objs/example.o:example.c:(.text+0x0):
first defined here
0.736 collect2: error: ld returned 1 exit status
0.737 make: *** [Makefile:60: /app/build/blah.out]
Error 1
0.737 make: Target 'test' not remade because of
errors.
```

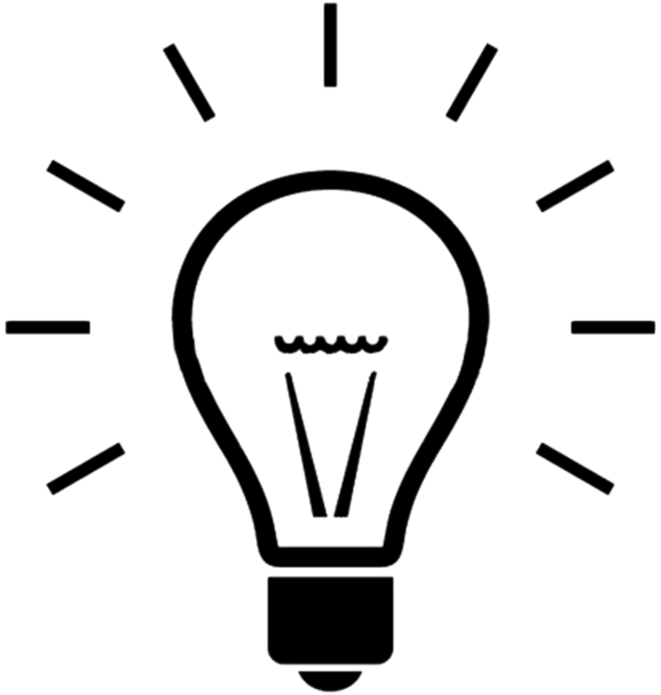


```
/app/build/objs/static_example.o: in function
'main':
        static_example.c: multiple definition of 'main'
```

Feedback: In static_example.c, press CTRL+f. Type in "main" without the quotes. Look at both sections of the code, find the one you do not want to run, and delete everything between and including: main(...){...}

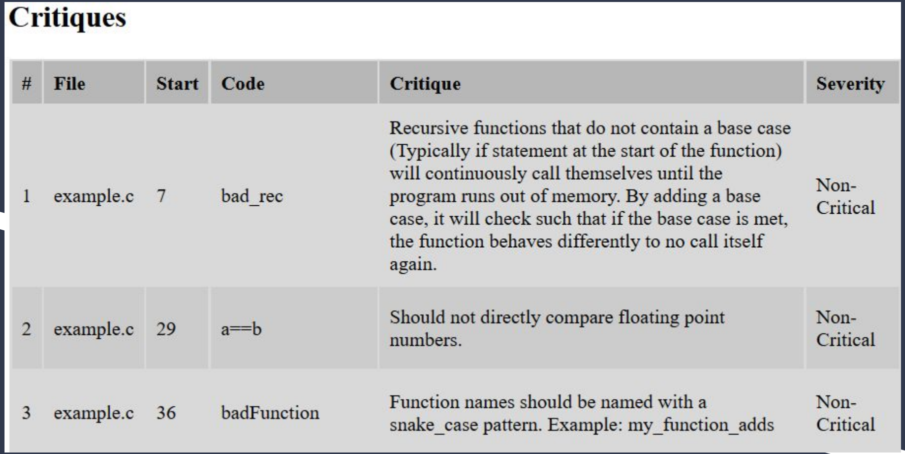
```
/app/build/objs/static_example.o: in function 'max':
        static_example.c: multiple definition of 'max'
```

Feedback: In static_example.c, press CTRL+f. Type in "max" without the quotes. Look at both sections of the code, find the one you do not want to run, and delete everything between and including: max(...){...}



Problem Statement

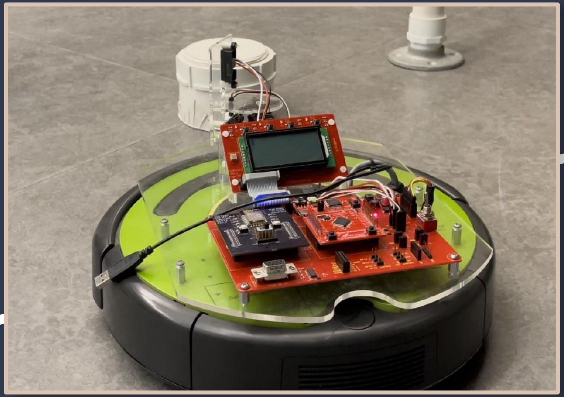
- 2880 is many students first experience with C
- Errors can be confusing and overwhelming
- TAs are not always available to help



Critiques

#	File	Start	Code	Critique	Severity
1	example.c	7	bad_rec	Recursive functions that do not contain a base case (Typically if statement at the start of the function) will continuously call themselves until the program runs out of memory. By adding a base case, it will check such that if the base case is met, the function behaves differently to no call itself again.	Non-Critical
2	example.c	29	a==b	Should not directly compare floating point numbers.	Non-Critical
3	example.c	36	badFunction	Function names should be named with a snake_case pattern. Example: my_function_adds	Non-Critical

Code Critiquer System for the C Language and Embedded C


A CPR E 2880 CyBot

Project Overview

- Project is a web-based critiquer tool
  - Continuation of sdmay24-34
  - Students upload C files to tool
  - Files are statically analyzed to search for antipatterns
  - Tool generates student feedback
- Will modify current system developing new features
- Tailored for CPR E 2880
- Targeting a Spring Semester Prototype

Project Requirements

User Requirements (For Students)

- Easy to use
- Clear, concise, and aesthetically pleasing UI
- Compiler output must be parsed and explained

User Requirements (For Instructors)

- Access to lots of statistics/analytics
  - Ability to sort student tests
  - Visual chart interpretations of data
  - Customizable
  - Ability to download information
- Easy to maintain

Functional Requirements

- Files in C upload successfully
- The program compiles provided code
- Provide proper feedback based on…
  - Static analysis (antipatterns)
  - Dynamic analysis (unit tests)
- Interact using GUI
- Accounts for data sheet errors

Nonfunctional Requirements

UI

- GUI is simplistic and intuitive
- Quick to navigate
- All feedback is beginner-friendly

Performance

- Feedback provided in < 10 seconds
- File uploads should take no more than a few seconds

Nonfunctional Requirements (cont.)

Maintainability

- Adaptable for:
  - Variety of programming purposes
  - Instructor needs and preferences
- Well-documented
- Modular
- Easy to add antipatterns

Related Products

MTU Projects (Dr. Ureel)

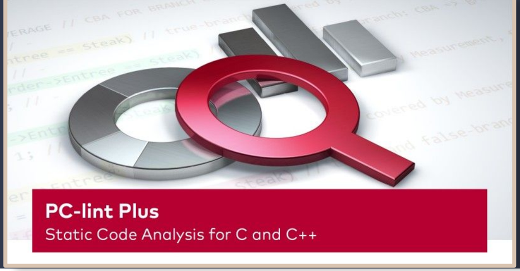
Pros:

- Multiple critiquers for different languages
- Education-focused
- Canvas integration

Cons:

- Products are just prototypes and not widely available
- Only uses static analysis

PC-lint Plus



Pros:

- Professional grade tool
- Identifies security issues as well as bad coding practices
- Undergone rigorous auditing and testing

Cons:

- Costs money
- Restrictive licensing

CodePal.AI

CodePal
Your Ultimate Coding Companion

Pros:

- It gives detailed feedback on:
  - Syntax/structure
  - Code readability
  - Functionality
- Visually appealing feedback

Cons:

- AI can be unreliable
- Advanced features cost money
- Limited number of queries at free level

Market Gap

- Tailored to CPR E 2880/embedded C
- Reliability and Confidence
- Beginner-friendly explanations
- Customizable antipatterns

Key Risks

- Possibly used for cheating
- Uploading malicious code
- Should help, not solve
- Should not mislead students
- Need to be aware of “false positives”

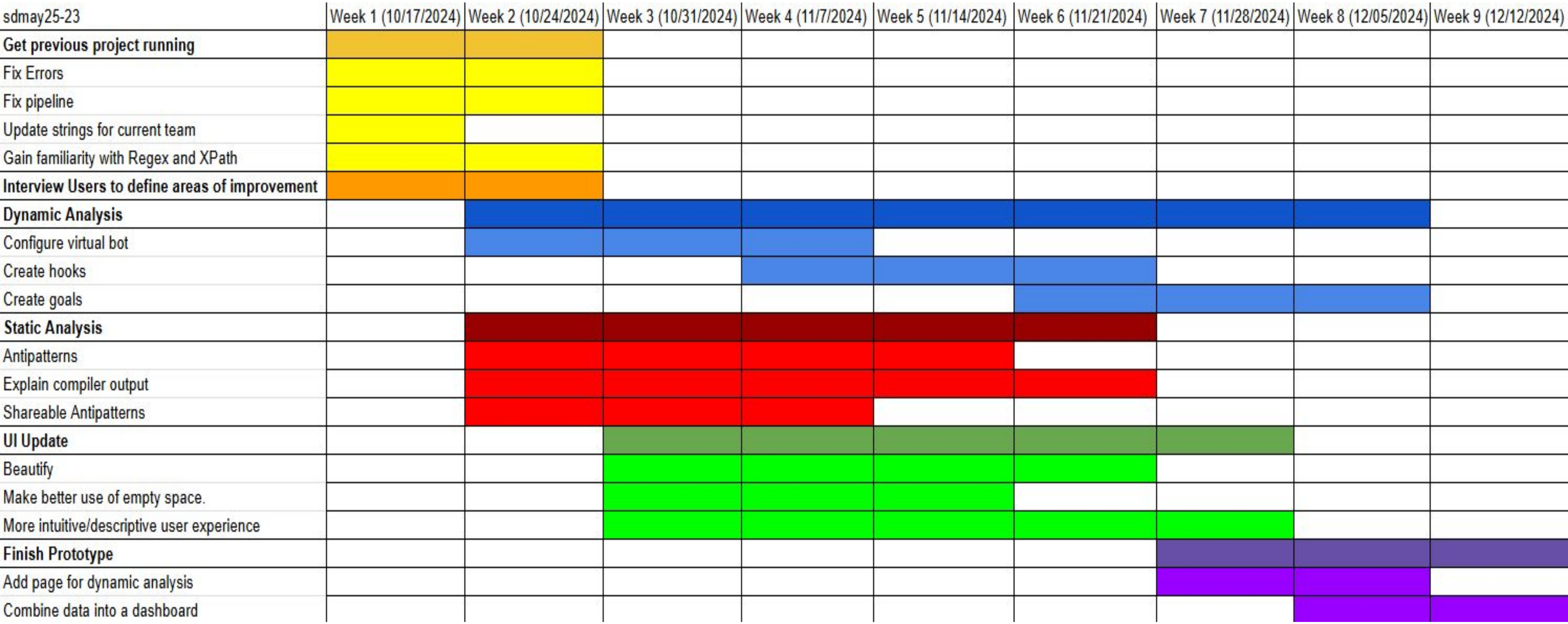
Risk Mitigation Strategies

- Accounts will be secured
- Feedback given is static
- Containerization
- Segmentation
- Contained in Iowa State network

Resource/Cost Estimate

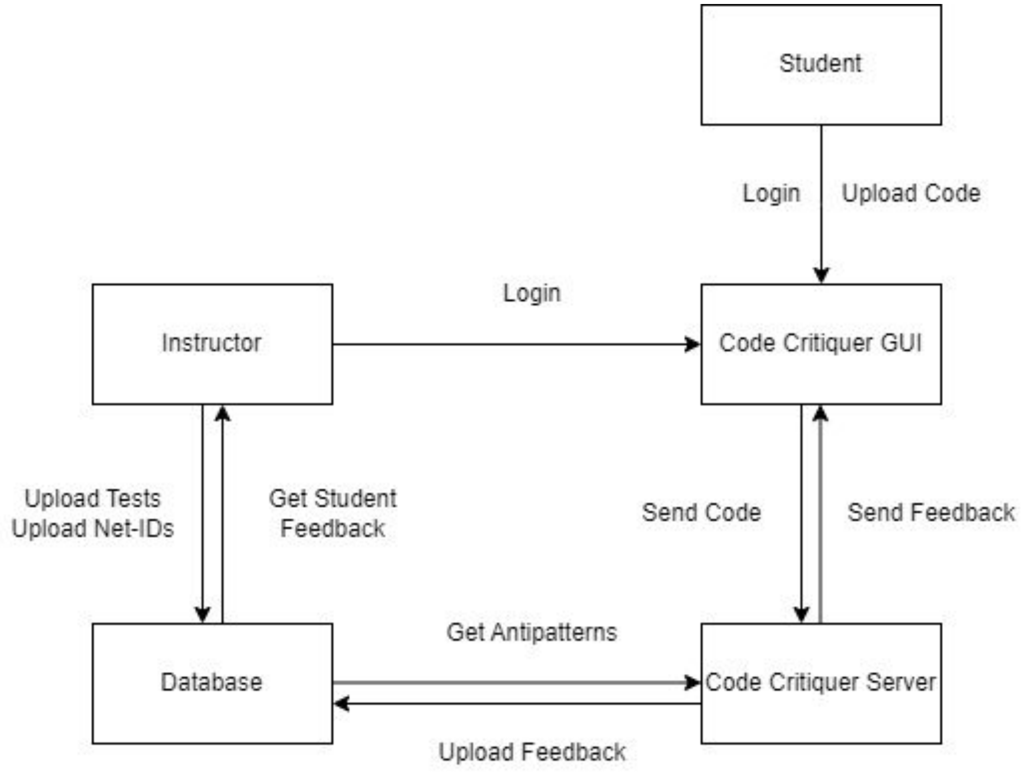
- Technical cost:
    - Uses available university resources
    - Minimal energy costs
    - Open-source software
- Human cost:
    - Maintenance once project is completed
    - Training to maintain system

Semester 1 Project Timeline

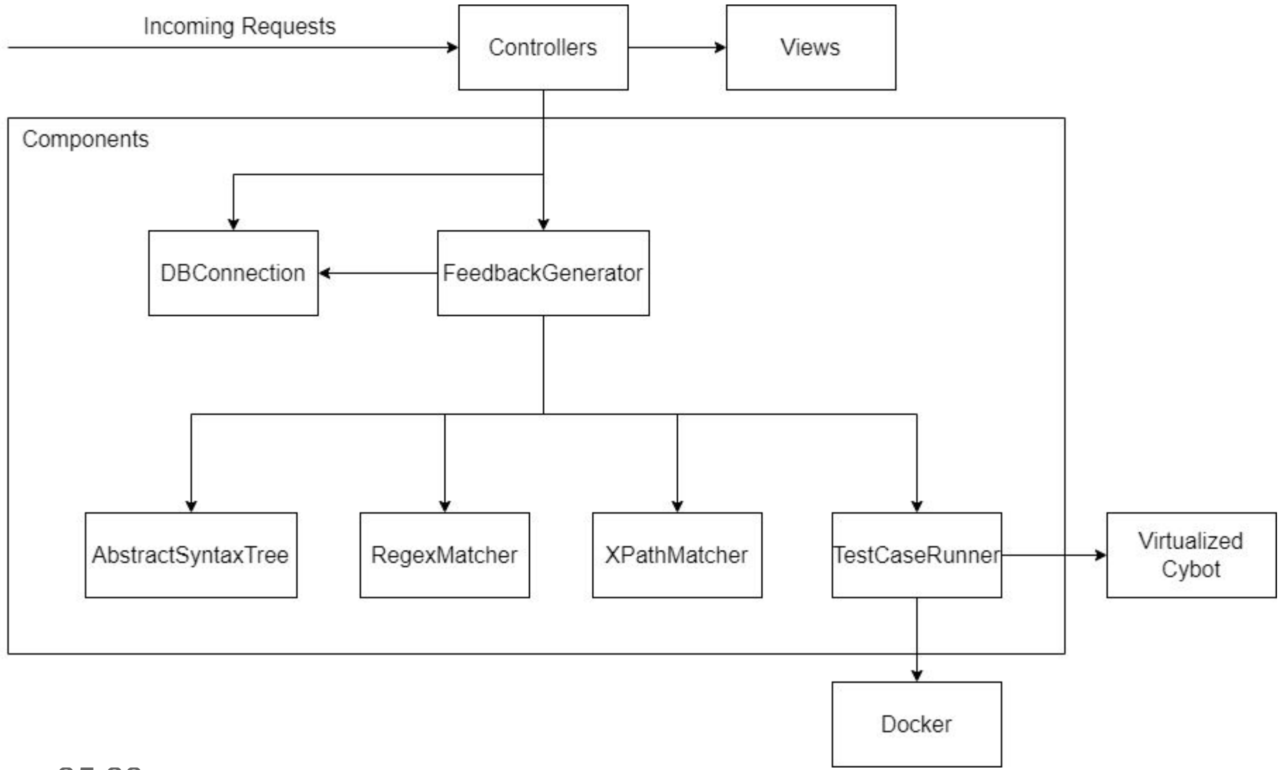


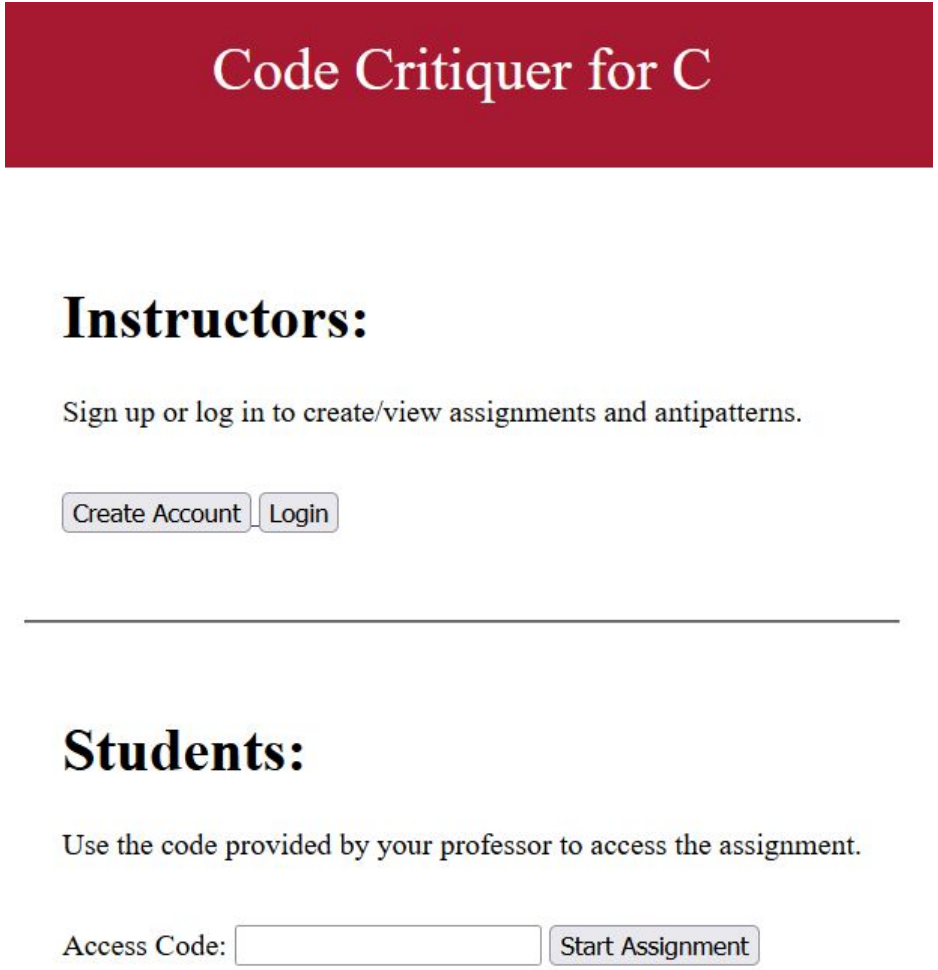
sdmay25-23	Week 1 (10/17/2024)	Week 2 (10/24/2024)	Week 3 (10/31/2024)	Week 4 (11/7/2024)	Week 5 (11/14/2024)	Week 6 (11/21/2024)	Week 7 (11/28/2024)	Week 8 (12/05/2024)	Week 9 (12/12/2024)
Get previous project running	🟧	🟧							
Fix Errors	🟨	🟨							
Fix pipeline	🟨	🟨							
Update strings for current 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								🟪	🟪

System Sketch



System Block Diagram




Code Critiquer for C

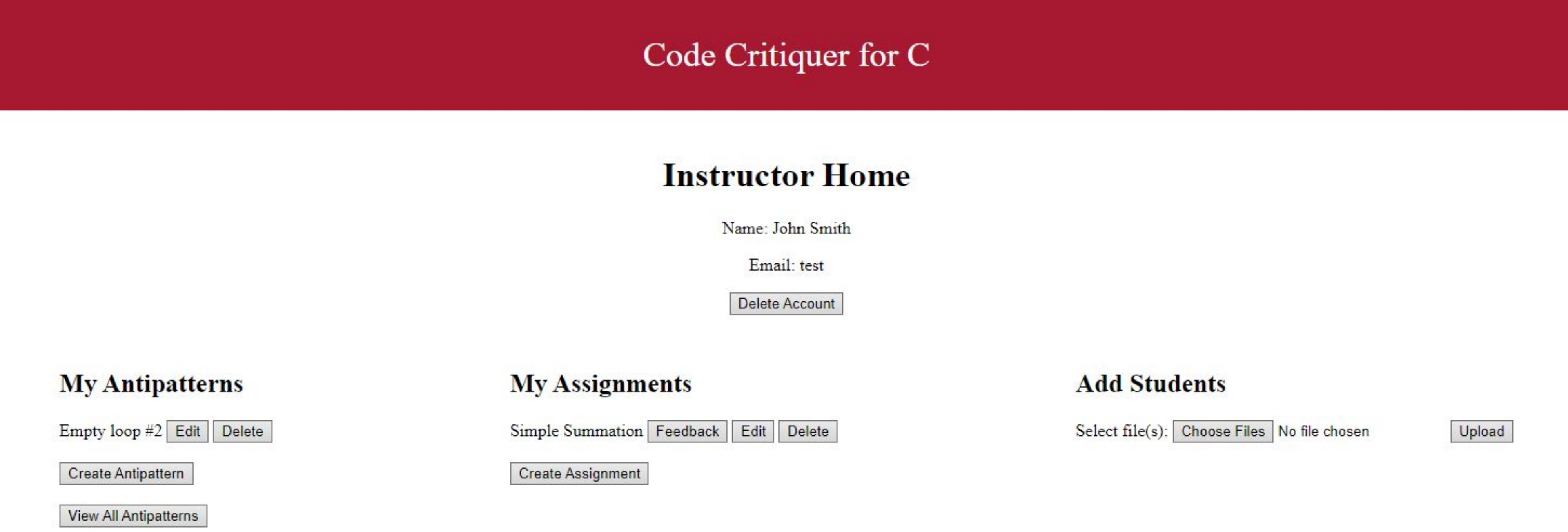
Instructors:

Sign up or log in to create/view assignments and antipatterns.

Students:

Use the code provided by your professor to access the assignment.

Access Code:


Code Critiquer for C

Instructor Home

Name: John Smith

Email: test

My Antipatterns

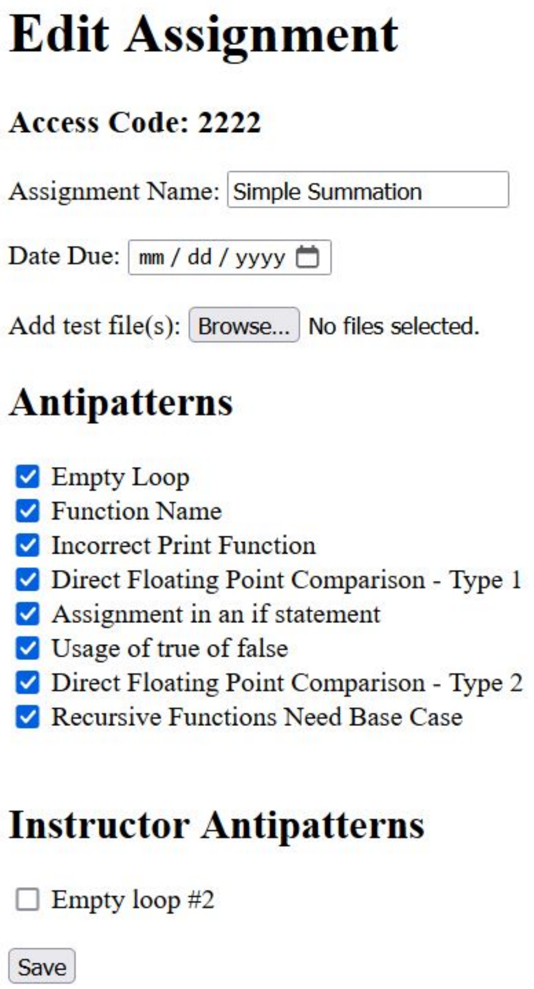
Empty loop #2

My Assignments

Simple Summation

Add Students

Select file(s):



Edit Assignment

Access Code: 2222

Assignment Name: Simple Summation

Date Due: mm / dd / yyyy

Add test file(s): Browse... No files selected.

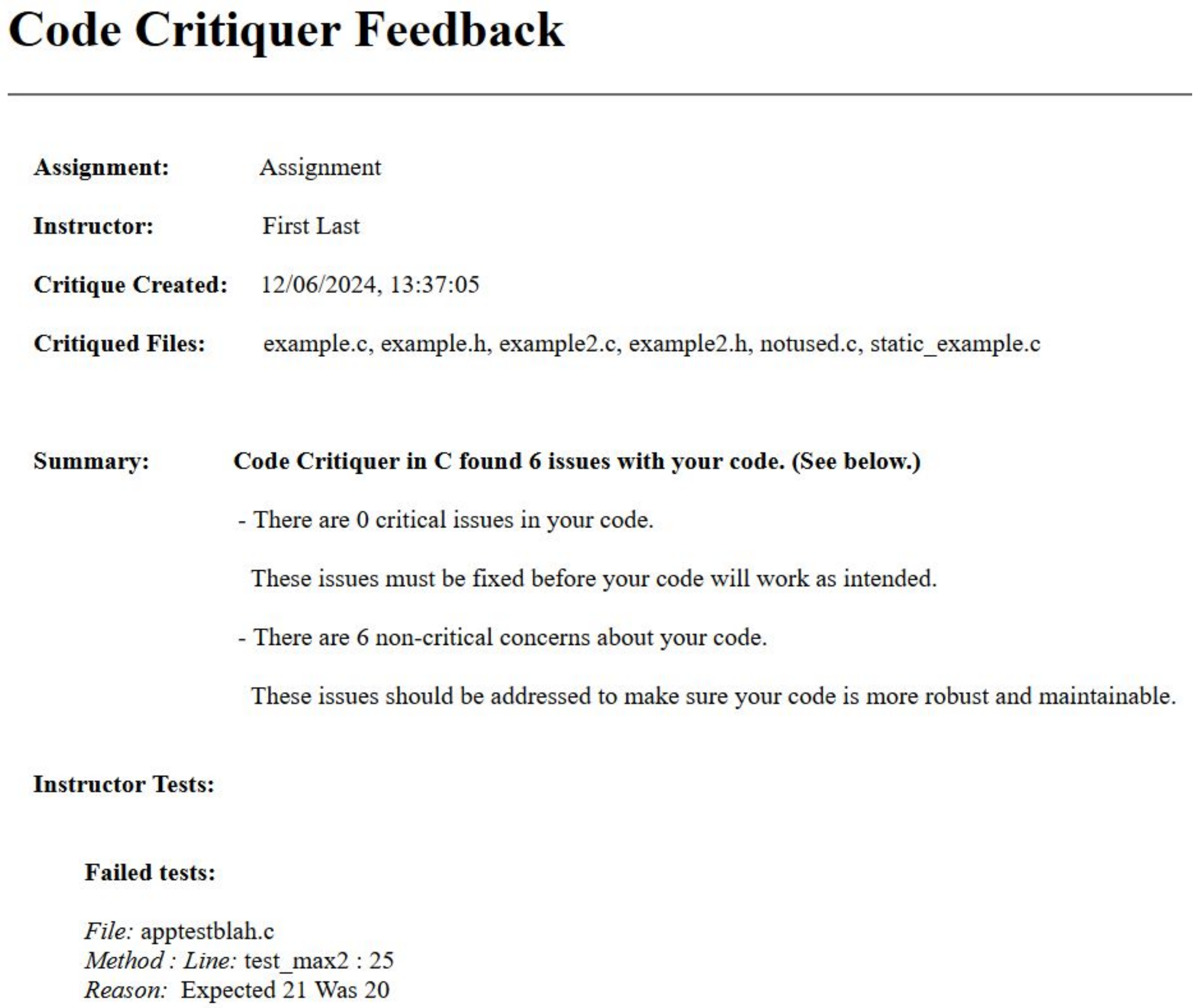
Antipatterns

- [✓] Empty Loop
- [✓] Function Name
- [✓] Incorrect Print Function
- [✓] Direct Floating Point Comparison - Type 1
- [✓] Assignment in an if statement
- [✓] Usage of true of false
- [✓] Direct Floating Point Comparison - Type 2
- [✓] Recursive Functions Need Base Case

Instructor Antipatterns

- [ ] Empty loop #2

Save



Code Critiquer Feedback

Assignment:	Assignment
Instructor:	First Last
Critique Created:	12/06/2024, 13:37:05
Critiqued Files:	example.c, example.h, example2.c, example2.h, notused.c, static_example.c

Summary: **Code Critiquer in C found 6 issues with your code. (See below.)**

- There are 0 critical issues in your code.

  These issues must be fixed before your code will work as intended.

- There are 6 non-critical concerns about your code.

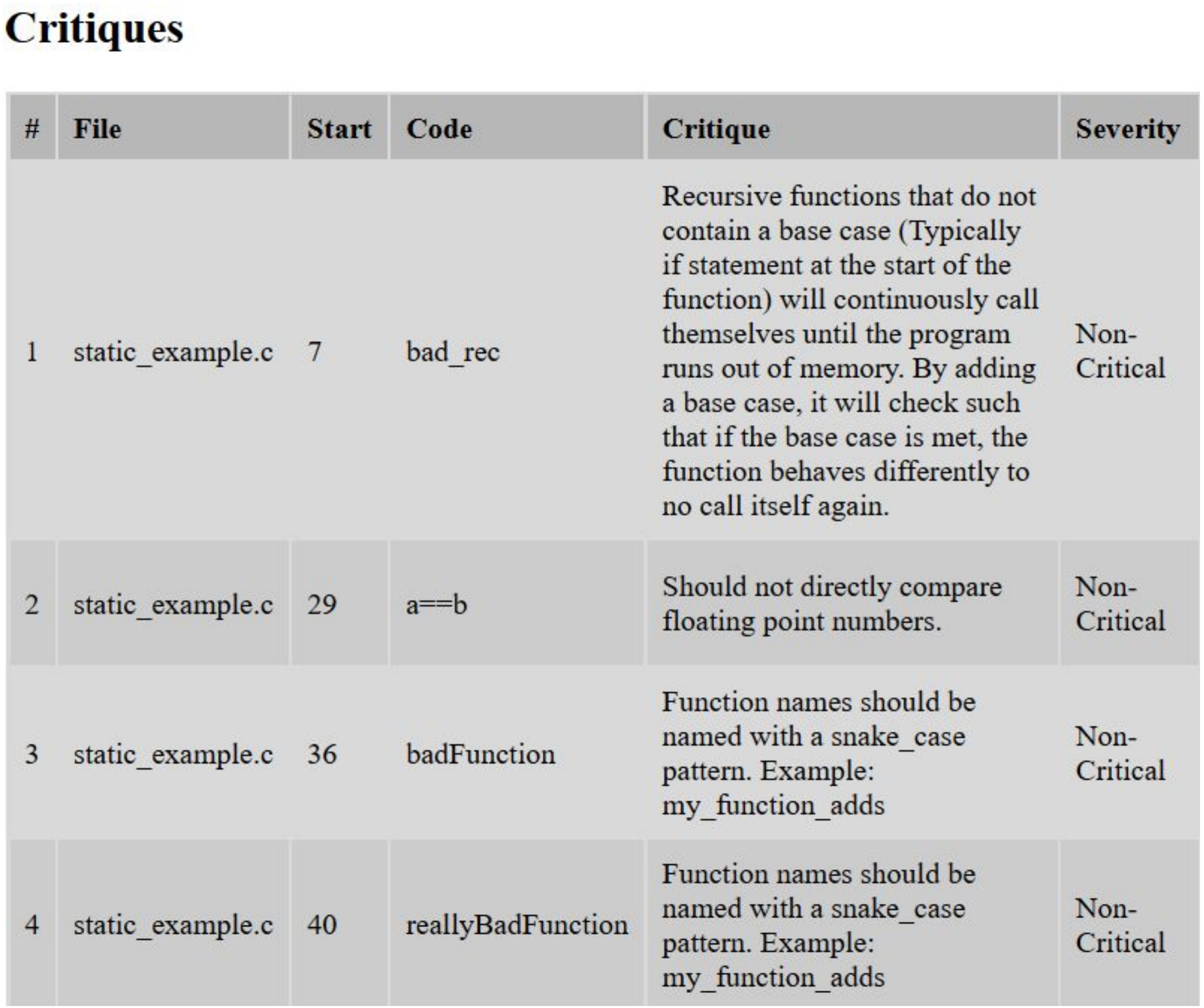
  These issues should be addressed to make sure your code is more robust and maintainable.

Instructor Tests:

Failed tests:

File: apptestblah.c
Method : Line: test_max2 : 25
Reason: Expected 21 Was 20

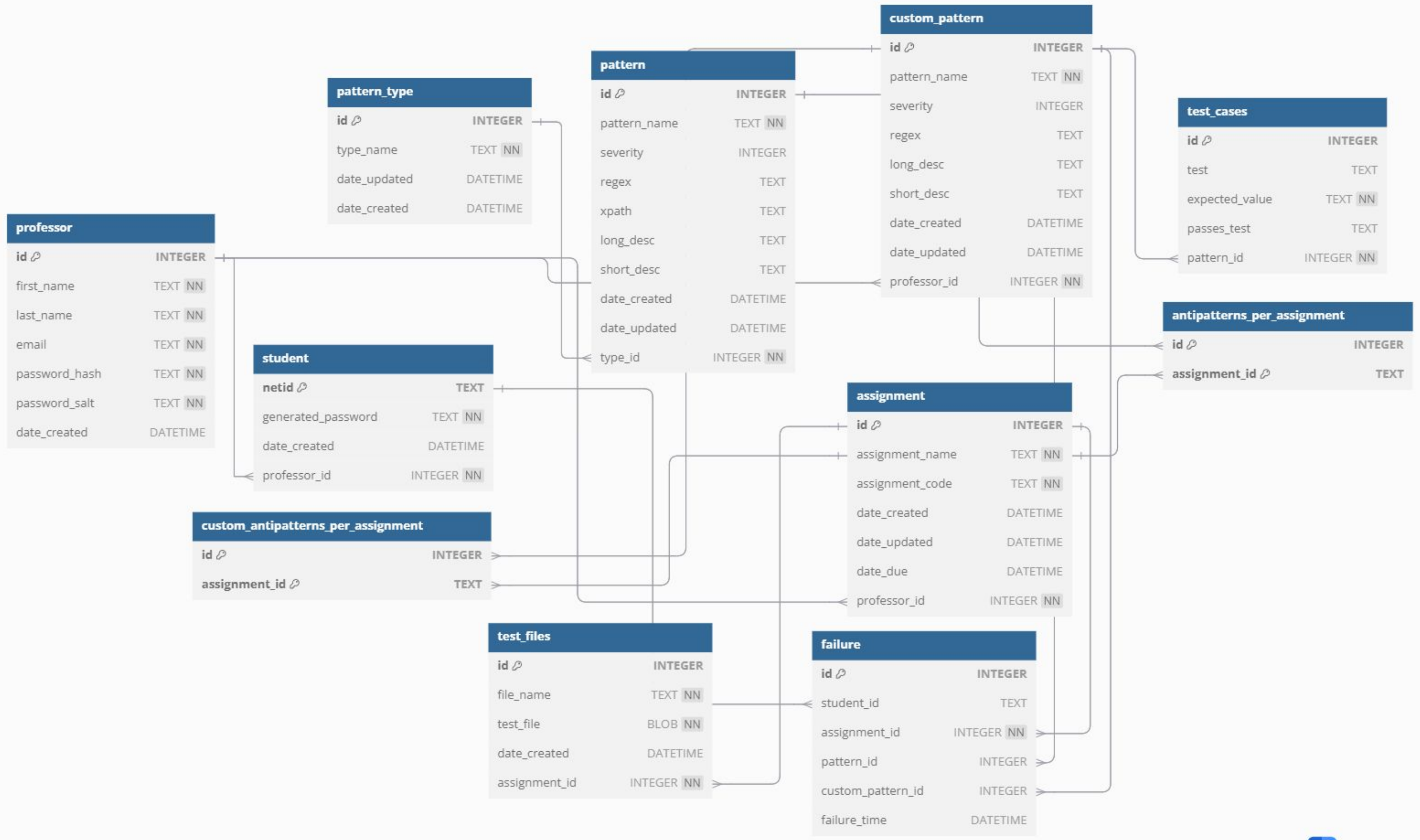
C Code Critiquer - sdmay25-23



Critiques

#	File	Start	Code	Critique	Severity
1	static_example.c	7	bad_rec	Recursive functions that do not contain a base case (Typically if statement at the start of the function) will continuously call themselves until the program runs out of memory. By adding a base case, it will check such that if the base case is met, the function behaves differently to no call itself again.	Non-Critical
2	static_example.c	29	a==b	Should not directly compare floating point numbers.	Non-Critical
3	static_example.c	36	badFunction	Function names should be named with a snake_case pattern. Example: my_function_adds	Non-Critical
4	static_example.c	40	reallyBadFunction	Function names should be named with a snake_case pattern. Example: my_function_adds	Non-Critical

Database Table Diagram



custom_pattern		
id	INTEGER	
pattern_name	TEXT	NN
severity	INTEGER	
regex	TEXT	
long_desc	TEXT	
short_desc	TEXT	
date_created	DATETIME	
date_updated	DATETIME	
professor_id	INTEGER	NN

pattern		
id	INTEGER	
pattern_name	TEXT	NN
severity	INTEGER	
regex	TEXT	
xpath	TEXT	
long_desc	TEXT	
short_desc	TEXT	
date_created	DATETIME	
date_updated	DATETIME	
type_id	INTEGER	NN

pattern_type		
id	INTEGER	
type_name	TEXT	NN
date_updated	DATETIME	
date_created	DATETIME	

test_cases		
id	INTEGER	
test	TEXT	
expected_value	TEXT	NN
passes_test	TEXT	
pattern_id	INTEGER	NN

professor		
id	INTEGER	
first_name	TEXT	NN
last_name	TEXT	NN
email	TEXT	NN
password_hash	TEXT	NN
password_salt	TEXT	NN
date_created	DATETIME	

student		
netid	TEXT	
generated_password	TEXT	NN
date_created	DATETIME	
professor_id	INTEGER	NN

antipatterns_per_assignment	
id	INTEGER
assignment_id	TEXT

assignment		
id	INTEGER	
assignment_name	TEXT	NN
assignment_code	TEXT	NN
date_created	DATETIME	
date_updated	DATETIME	
date_due	DATETIME	
professor_id	INTEGER	NN

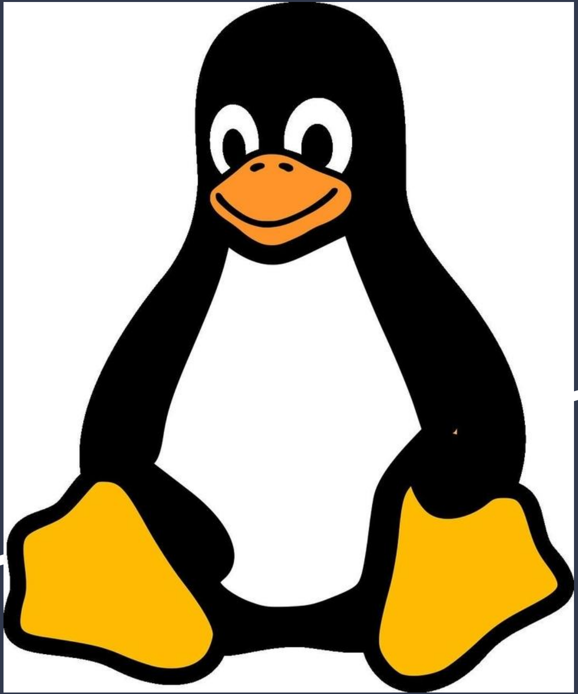
custom_antipatterns_per_assignment	
id	INTEGER
assignment_id	TEXT

test_files		
id	INTEGER	
file_name	TEXT	NN
test_file	BLOB	NN
date_created	DATETIME	
assignment_id	INTEGER	NN

failure		
id	INTEGER	
student_id	TEXT	
assignment_id	INTEGER	NN
pattern_id	INTEGER	
custom_pattern_id	INTEGER	
failure_time	DATETIME	

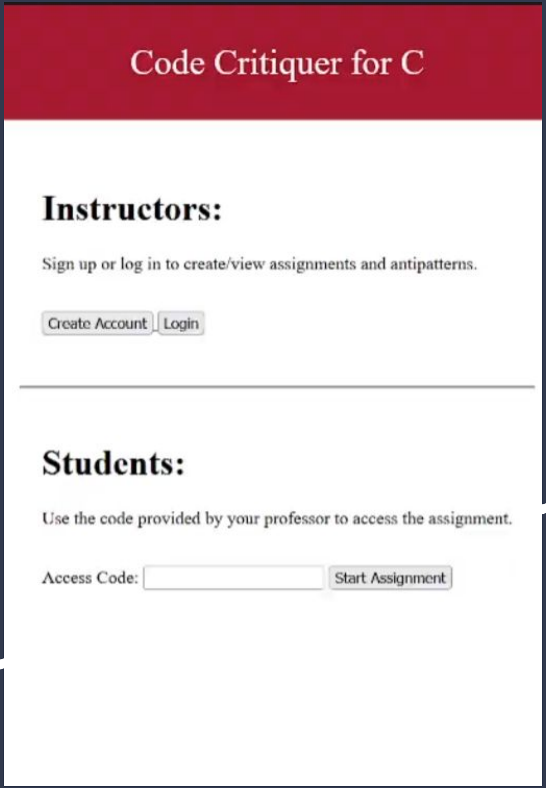
dbdiagram.io

Hardware Platform



- Linux server
  - Used to host a website for the users to access
  - Holds the database containing system data
- Tiva TM4C123GH6PM Microcontroller
  - Used to run the virtualized CyBot for runtime analysis

Software Platform


Code Critiquer for C

Instructors:

Sign up or log in to create/view assignments and antipatterns.

Students:

Use the code provided by your professor to access the assignment.

- HTML, CSS, and Jinja
  - Website Design
- Flask
  - Handles routing of pages in the web application
- Regular Expressions/XPath
  - Used for static analysis
- Docker
  - Used for dynamic analysis

Test Plans

CODE
PLAN
BUILD
TEST
RELEASE
DEPLOY
OPERATE
MONITOR

- Manual Testing
  - Thoroughly testing each feature that we add
  - Testing edge cases
- Continuous Integration and Continuous Delivery (CI/CD)
  - Automated tests that run before updating production code
- User Feedback
  - Fix issues discovered by users

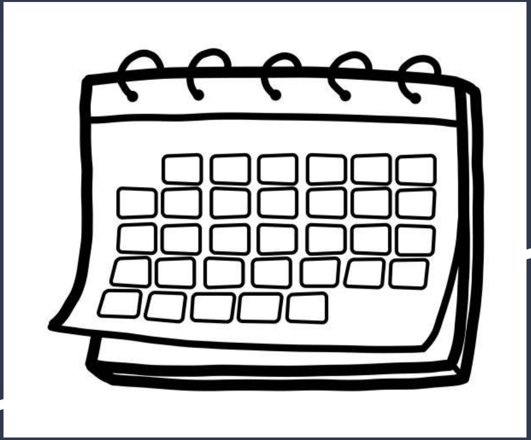
Project Status

- Got previous project running
- Built out containers for analysis
- Created student accounts and home screen
- Began instructor analytics page

Task Distribution (Fall 2024)

- Alix Noble
  - Added pages for data insights and analytics
- Andrew Sand
  - Headed User Requirements Gathering and professional documentation development
- Owen Sauser
  - Developed team website and monitored CPR E 2880 Discord for student issues and questions
- Samuel Lickteig
  - Set-up backend endpoints and database tables for additional features
- James Joseph
  - Configured dynamic analysis

Future Plans (Spring 2025)



- First prototype trial run
- Manual/Unit testing
- Finalize design for and implement UI overhaul
- Add per-lab antipatterns and tests
- Continue defining user account hierarchy (RBAC)
- Finish statistics/analytics pages

Any Questions, Comments, or Suggestions?