Team 5: Anti Panda

Date: January 22th, 2023
Time: 02:00 pm EDT
Location: Zoom
Duration: 2pm to 3:05pm

Attendees

- Mya Davis
- Ame Tian
- Isaac Elenbaas
- Noah Riske
- Zane Smalley
- Maritza Gonzalez

Action Items

- Brainstorm design ideas
- Determine the scope of the project
- Look at Leo's pattern examples for inspiration on the above
    - https://webta-dev.cs.mtu.edu/PatternDB/

Discussions

- Discuss user interface general layout and how expressions will be constructed
    - How do we want users to submit their "bad" code snippets?
    - Is generation largely automatic or manual?
- What do we want to display and show to users?
    - Regular expressions
    - Test cases?
    - Advice?
- How do we want to display the regex to users?
    - Scratch like building blocks
    - Highlighting parts of code and explaining it
- Do we just implement a UI or do we need to implement the code logic for translating code to regex?
- Do we want users to have login credentials?
- How many/what kinds of categorization systems?

Conclusions:

- No home/navigation page, landing page contains the main purpose of the application (similar to unit conversion sites or YouTube downloaders)
- explainshell.com is a great resource to take inspiration from
- https://regex-generator.olafneumann.org/
    - Cool ideas (step 2 in particular, their implementation is poor but could be built on)
    - The idea of stages of creating a pattern instead of one big one is good
- More complicated features like lookaheads/lookbehinds are rarely necessary
    - We can be pretty free with abstracting away generation and still should be able to generate a matching expression
    - Grouping is nearly entirely unnecessary, we may implement some sort of shorthand for groups of patterns if we investigate the block format
- Help page with documentation or hints?
    - Click drop down/scroll down for step by step of how the regular expression was produced.

Design Outline:

- Multiple stages outlined below
    - Single-screen, ideally non-scrolling with back/forward stage buttons
        - Stages 2 and 3 may be on one screen, or scrolling moves/organizes elements and transitions between the two

1. Inputting test cases and specifying matching/non-matching
2. Block generation
    - Highlight sections, suggested capture groups
    - Blocks can be combined into aliases for groups of blocks
    - Some default blocks such as quantifiers, any amount of whitespace, limited and infinite loops, any alphanumeric character
        - Character class / individual character selection menu is crucial - graphically including or excluding classes or characters
3. Block assembly
    - User assembles blocks into full regular expression
    - Given test cases are shown below assembly, what text is causing them to incorrectly match or not match is highlighted with arrows to the matching offending block or space between blocks
    - Stretch goal / may not be helpful: Site can display generated matching and nonmatching generated test case examples

Inspiration Sources

- explainshell.com
- regex-generator.olafneumann.org

- Scratch
    - Block-based
    - Very newbie friendly
    - Leaning towards this approach but horizontal