Team 5 Pattern Panda

Interaction Design Document

App Summary 1

App Idea

The main purpose of the app is to help users with little to no knowledge of Regular Expressions to create regular expressions that identify Antipatterns (bad code formats) in the PatternDB. The database is a secondary concern and could be copied from the existing collection or ignored and implemented after this semester.

Users

- Regex Authors
    - Instructors of some sort, are using the app to outline some bad programming behavior they have identified
    - May have little formal programming experience, app may be used by e.g. highschool math instructors using Matlab or even something like data entry
    - Likely have little to no experience with Regular Expressions

App Usage

Users interact with the application to create Antipattern identifying Regular Expression strings and test them on both positive and negative test cases. The site will mainly consist of a single Regular Expression creation page where the users will give examples of strings the pattern both should and should not match with. The interface will allow them to make a regular expression easily based on those example cases.

Data

- Regular Expressions
- Matching and non-matching test cases for each expression

App Summary 2

App Idea

The main purpose of the app is a Scratch-esque app with draggable blocks to help users with little to no knowledge of regular expressions to create regular expressions that identify Antipatterns (bad code formats) and upload them to a PatternDB. The app will also match against positive and negative test cases during construction. The app will mainly consist of a single, ideally non-scrolling, regular expression creation page.
The database is a secondary concern and should be largely copied from the existing collection or ignored and implemented after this semester.

Users

- Regex Authors
    - Teachers and professors, using the app to outline some bad programming behavior they have identified
    - May have little formal programming experience, app may be used by e.g. highschool math instructors using Matlab or even something like data entry
    - Likely have little to no experience with regular expressions

Workflow

- Login page: users will be greeted by a login page that uses MTU SSO login credentials, users will need to login prior to using the app
- Home page: users will enter their examples of good and bad code snippets and use blocks to construct regular expressions
    - Multiple steps, primary two are creating blocks and then combining them
    - Stretch/non-essential: Users familiar with regular expressions can access an “advanced” menu to start with an existing regular expression (or edit one created with the blocks) and blocks will be generated
- Stretch/non-essential: Regular expression showcase page: users will be able to view the detailed regular expressions broken down into blocks regular expressions

- Stretch/non-essential: Page will automatically generate a string that matches the regex as a sanity check that the regular expression is actually looking for the right thing.

Data

- Antipattern metadata - title, language
- Antipattern regular expression
- Antipattern matching and non-matching test cases

Anticipated Challenges

- Providing a comprehensive tutorial that does not overwhelm new users
- Balancing information presented to users and the complexity of regular expressions
- Learning and adapting Google's Blockly framework to produce the regex blocks

System Overview

The app will be a web app. Users can access the web app through any web browser on any devices such as desktop computers, laptops, and mobile devices with internet connection. The web app will be a single page application where users can input their code snippets and interact with the generated regex information on the same page.

Stakeholders and Users

Users

User	Expected Age	Tech Expertise
University Instructor (CS Background)	24+	Adept - Expert
University Instructor (No CS Background)	24+	Novice - Adept
High School Instructor	24+	Novice - Adept

User Goal Table

User	Goal
University Instructor (CS Background)	Directly input regex into software (advanced mode).
University Instructor (No CS Background)	Provide an antipattern through a user-friendly interface.
High School Instructor	Provide an antipattern through a user-friendly interface.

Primary User Personas

Persona 1
Name: Dr. X
Age: 35
Residence: Michigan Technological University (Houghton, MI)
Job: Adjunct professor in the computer science department
Goal: Input antipattens into the system to build the database
Behavior: Eager to improve a database of regular expressions that will help students in the long run.
Relationship: Uses app to communicate with students and research team.

Persona 2
Name: Dr. Y
Age: 55
Residence: Case Western Reserve University (Cleveland, OH)
Job: Tenured professor in the computer science department
Goal: Input antipatterns into the system to build the database, use the database to teach students in their lab
Behavior: Interested in researching antipatterns and improving an overall shared application
Relationship: Uses app to communicate with students and lab.

Secondary User Personas

Persona 1
Name: John Doe (he/him)
Age: 19
Residence: Michigan Technological University (Houghton, MI)
Job: Undergraduate Research Assistant, College of Computing Student
John is a busy bee as both an undergraduate computer science student and a research assistant for Dr. Ureel. He is a part of the research team. His role is to program the MATLAB Code Critiquer/WebTA. John will connect the software that the Pattern Pandas team develops to the backend he is currently working on.

Persona 2
Name: Lauren Albrecht (she/her)
Age: 22
Residence: Michigan Technological University (Houghton, MI)
Job: Graduate Research Assistant
Lauren is a workaholic that leads the development team of Dr. Ureel's research and aids in development of the WebTA system. She will implement a subsystem related, but separate to software that the Pattern Pandas' are implementing.

User Environment

- On the given host the user is using, a wifi connection is required to connect to the web server and access the application.
- The users will interface with a web browser program. The users are using a keyboard to type characters, and the mouse to drag blocks around the screen of the app.

Scientist Notes

- Scientist Meeting Note 1
- Scientist Meeting Note 2

Nominal Use Scenarios

Use Scenario 1: Dr. X Creates Antipattern for importing util.*

Dr. X is an adjunct professor at Michigan Tech. His weekly meeting with his research team consists of improving the database of anti patterns. Dr. X logged into the Anti Pattern app with his Michigan Tech SSO credential as an admin. Dr. X is then greeted by the antipattern creation page. Dr. X is presented with a palette of regex building blocks and input fields for test cases. Dr. X inputs the test case “import java.util.*;” and sets that to be true. A new input for test cases then appears below that test case. Dr. X then inputs the test case “import java.io.*;” and sets that case to be false. Dr. X then creates an antipattern regex by grabbing the “Word” block and dragging that into the canvas and typing “import” into it. Dr. X then grabs another “Word” block and connects it to the right side of the existing block, and types “java.util.*” into the new block. Dr X. then looks at his test cases and confirms that a checkmark is next to both of them, indicating that they both pass. Dr X. is now satisfied that his newly created antipattern regex finds the correct pattern and clicks the Save button. The new save view then pops up with fields for the Title, Language, Advice, and Notes. Dr. X then inputs “imports util.*” in the Title and sets Language to Java, and clicks Save. Dr. X is then presented with a confirmation message saying that the antipattern was added to PatternDb.

Use Scenario 2: Dr. X Creates Antipattern for using System.arraycopy

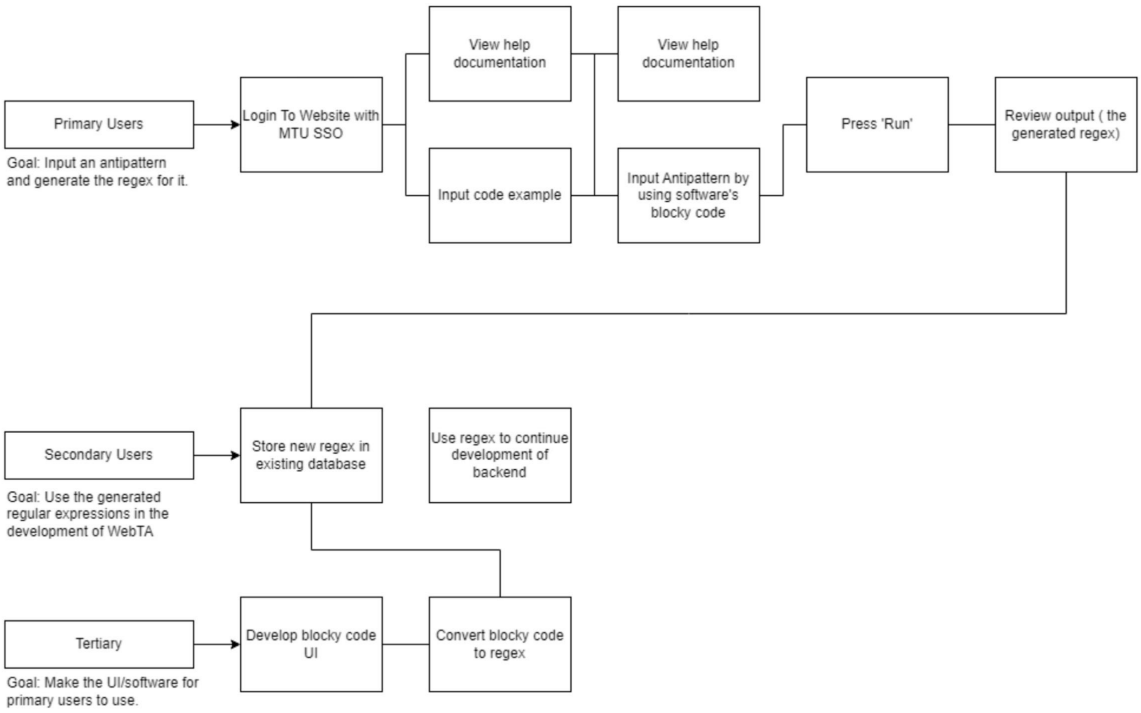
Dr. X is an adjunct professor at Michigan Tech. His weekly meeting with his research team consists of improving the database of anti patterns. Dr. X logged into the Anti Pattern app with his Michigan Tech SSO credential as an admin. Dr. X is then greeted by the antipattern creation page. Dr. X is presented with a palette of regex building blocks and input fields for test cases. Dr. X inputs the test case “arraycopy” and sets that to be false. A new input for test cases then appears below that test case. Dr. X then inputs the test case “System.arraycopy(;” and sets that case to be true. Dr. X then creates an antipattern regex by grabbing the “Word” block and dragging that into the canvas and typing “System.arraycopy” into it. Dr X. then looks at his test cases and confirms that a checkmark is next to both of them, indicating that they both pass. Dr X. is now satisfied that his newly created antipattern regex finds the correct pattern and clicks the Save button. The new save view then pops up with fields for the Title, Language, Advice, and Notes. Dr. X then inputs “arraycopy” in the Title and sets Language to Java, and clicks Save. Dr. X is then presented with a confirmation message saying that the antipattern was added to PatternDb.

User Error Scenario

- An example of user error would be if the user organized the Word blocks of the app in an erroneous way. The produced regex string from the block organization may be incorrect in that it does not follow any syntactical programming format.

The app would simply inspect code against the erroneous regex string(s). Because the erroneous regex string doesn't follow code syntax, *actual* instances of good code will not match. This causes no undefined behavior, so the app will not fail or terminate in any capacity, but it does represent a use case that the user would say impedes their work.

Hierarchical Task Analysis



Database Schema

List of Domain Classes:

- Antipattern - list of antipattern metadata: title, language, etc., along with the regex string
- TestCase - list of test cases associated with the regex

Domain Class: Antipattern - list of antipattern metadata

- Title - string, antipattern title
- Language - string, language of the antipattern
- Description - string, description of the antipattern
- Note - string, optional note field for the antipattern
- Regex - string, field containing the actual regex for the antipattern
- Tests - TestCase [many], link to the TestCase domain for the test cases

Domain Class: TestCase - list of test cases associated with an Antipattern

- Test - string, The test case itself
- Pass - bool, whether the regex is supposed to match the test case
- Result - bool, whether the regex matches the test case
- Antipattern - Antipattern [one], link back to the Antipattern that this is a test case for