Code Critiquer

App Idea: Code Critiquer is a static analysis tool for finding common anti-patterns in user-submitted code snippets. Similar to grammar or plagiarism checkers in which users can submit either their full source code or erroring segments, and have the app respond with feedback pertaining to the found anti-pattern (bugs, errors, etc.) found within the given code. Currently, Code Critiquer will be used on Java code with the goal of implementing MatLab support, if time permits.

App Users:

- **Novice Programmers.** Users with less experience are more likely to encounter problems caused by common anti-patterns.
- **Language Learners.** Users working in unfamiliar languages or environments are more likely to encounter problems caused by common anti-patterns.
- **Programming Tutors.** Users trying to clean or optimize their code can benefit from resolving common anti-patterns or implementing efficient programming patterns.

App Usage: There should be a text box or file submission where code is put into, and there should also be a button to submit it, then there should be a text box or message displayed on where the anti pattern is located and give feedback on the code. Will create a guest and login features allowing users to store their growth over time within the database, if they so desire.

App Data:

- **Remote Database**. We will remotely store patterns and anti-patterns as strings in our database, along with the associated critiques. If a user chooses to log in, their code submission will be stored, along with the critiques detected in that file. We will also store metadata about user submissions, such as time submitted and critique presence over time.
- **App Output**. We will create tooltip style critiques appearing over the associated line number within the provided code on the website..We will also provide output files in .txt, .pdf, and .java formats. All will contain the found critiques in an appropriate format. (.txt and .pdf will have the submitted code, with the critiques and associated line numbers following the code, whilst the .java will insert the critiques as TODO comments above the line)

App Views: There should be a text box or file submission button, and once the code is critiqued, it brings you to a critique view showing the critiqued code. There can be a view with the critiqued code and the past submission of the users code.

List of Implementation and Usability Challenges:

- **Understanding parsing:** The premade parser that will be provided to us may have a learning curve for the team.
- **Database Storage:** Precisely determining how and where this data is stored will be a challenge.
- **Unique Link Authentication:** Avoiding full user authentication should be easier but regardless, unique link authentications come with challenges that need to be considered.
- **User Progress Tracking:** Another challenge is determining how our team will track, store and display progress of the user.
- **Critique Display:** Designing UI elements that convey only critical information to the user is crucial to our project's usability.