Name: Dhritabrata Mitra

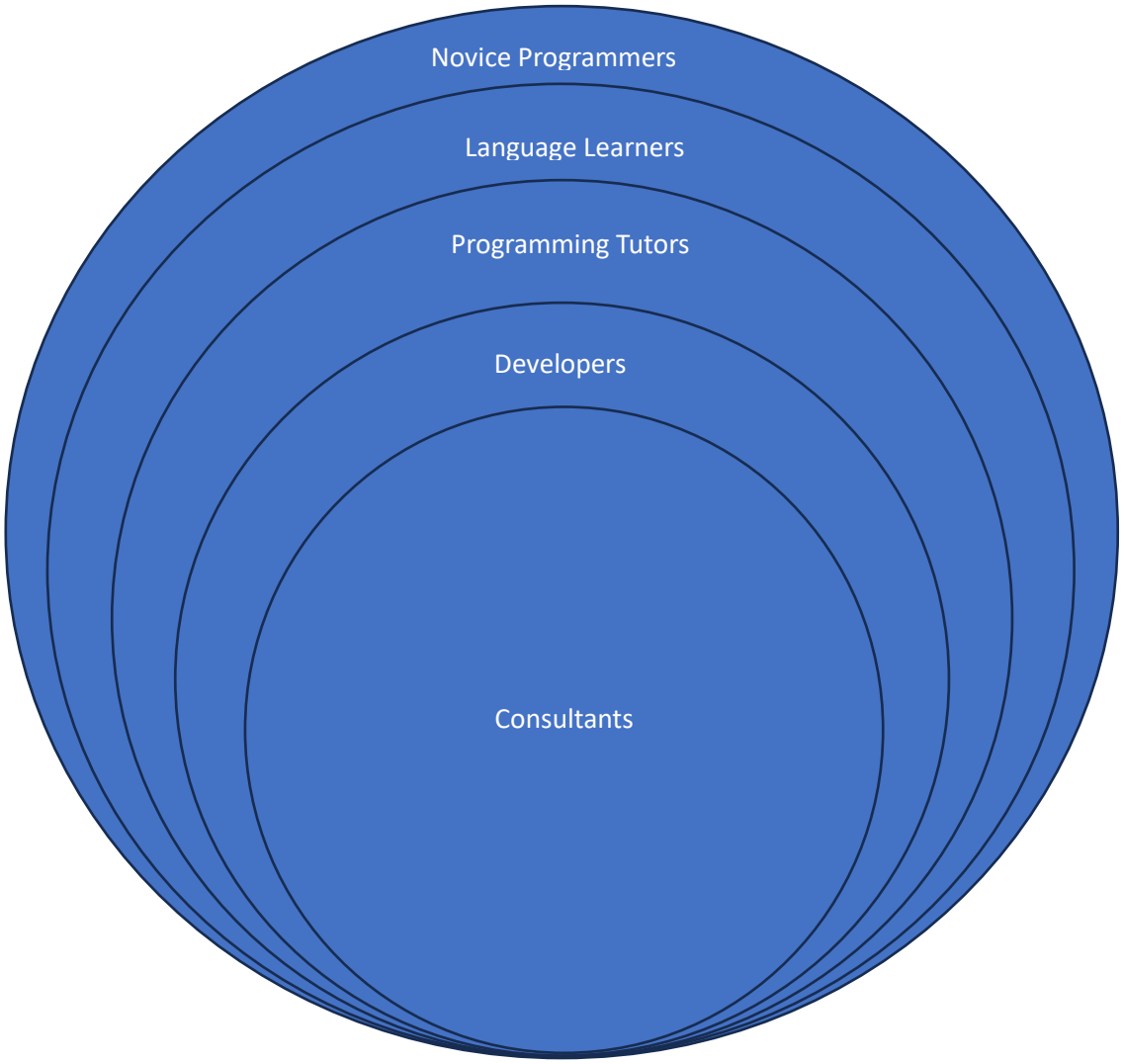
Project: Code Critique

App Description:

Code Critiquer is a static analysis tool designed to identify prevalent anti-patterns in code snippets submitted by users. Similar to grammar or plagiarism checkers, users can submit either their entire source code or specific error-ridden segments. The application provides feedback regarding the detected anti-patterns, such as bugs and errors, within the provided code. Currently, Code Critiquer is tailored for Java code analysis, and there are plans to implement MatLab support if time allows.

Stakeholder Analysis:

Onion Model



Stakeholder Description:

Consultants: Conduct a UI assessment and perform usability testing on the application. Simultaneously, consistently offer feedback to app developers throughout the app development process.

Developers: Developers will have the main responsibility of developing the app on the basis of consultants feedback.

Programming Tutors: Individuals aiming to clean or optimize their code can gain advantages by addressing prevalent anti-patterns or incorporating efficient programming patterns.

Language Learners: Users learning new or unfamiliar programming language or environments may encounter problem caused by common anti-patterns. They can take advantage of the app to learn the language faster.

Novice Programmers: Programmers who are new to programming can use this app to find their drawbacks and learn programming.

Stakeholders' Goal-influence Table:

Stakeholder	Goals	Contributing Influences
Consultants	Work with developer to create an ideal application.	Offering constructive feedback during testing to guide developers and lead the team in the correct direction.
Devlopers	Work with consultants to create an ideal application.	Based on the feedback from consultants, design and write code for the project.
Programming Tutors	Use the app as a tool for teaching	Provide feedback on how to improve the app
Language Learners	Use the app as a tool to learn a new programing language	Provide feedback on how to improve the app
Novice Programmers	Use the app as a tool to learn programing	Provide feedback on how to improve the app

Personas for the Primary stakeholder:

Persona 1:

Holland

Age: 18

Height: 5’ 7”

Weight: 133 lbs

Right-handed

Holland is a freshman at a community college in Michigan. He enrolled in computer science as a major but has basic programming experience. He can use the app to improve programming skills as he will learn the proper techniques for coding.

Persona 2:

Sayantan

Age: 23

Height: 5’ 8”

Weight: 147 lb

Left-handed

Sayantan is a recent graduate. He has a background as an Environment Engineer. He wants to learn coding as he is interested in knowing how an app works. He can use this app to help him become aware of where he needs to improve his knowledge.

Personas for the Secondary Stakeholder:

Mrinal

Age: 25

Height: 5’ 7”

Weight: 139 lb

Right-handed

Mrinal is a high school teacher who teaches programming. He can use the app to check the solutions submitted by the students or check his solution before showing it to his students.

Forest

Age: 30

Height: 5’ 9”

Weight: 160 lb

Right-handed

Forest is a biology professor at Michigan Technological University. He is interested in programming and wants to learn more about it. He can use the app to find the drawbacks he has in his concepts and use it to improve on them.

Hierarchical Task Analysis

1. Access the application through the browser and log in to the application using their account or by guest account.
2. Students can paste their code in a text box or upload the file containing their code to the website.
3. Possible bugs at line numbers will be shown which the users can use to fix their issues.
4. Users can store their previously submitted code along with the critiques they got if they want to look at it again in the future.
5. Users can download their output files in .txt, .pdf, and .java formats.

Appendix

Interview 1 Notes

1. Please provide details about your app idea, including an app overview and the languages/frameworks used.

    a. Code patterns are sought after in the source code.

    b. Coding utilizes structures known as patterns.

    c. The scientist (Ureel) focuses on identifying anti-patterns.

    d. Anti-patterns encompass either bug issues or unintended solutions with unforeseen consequences.

    Example: A viable solution with a security issue oversight

2. What language is preferred for the app?

    a. Java

    b. Groovy

    c. Grail

3. Why is a code critique app necessary, and how would it be beneficial?

    a. Especially useful for novices, such as those in classes CS1122 and CS1121.

    b. Identifies problems that may be overlooked.

    c. Offers solutions to problems that users may not know how to solve.

    d. Enhances coding skills.

    e. Allows users to review code, provide feedback, and make improvements.

4. Who are the users?

    a. Novice programmers

    b. Individuals with limited programming background

    c. Those seeking to improve their programming skills.

    d. People who make mistakes that experts may not anticipate.

5. How would users utilize the app?

    a. Upload source code through a designated field.

    b. Click the submit button.

    c. The program will analyze the uploaded files and provide a critique output.

6. What data should the app display?

    a. The user receives the uploaded code along with the provided critique.

7. How should the data be stored?

    a. Utilize Grail with a model-view system; information can be stored in a database or outputted.

    b. Pages can be stored in a document.

8. Is there a specific way the data should be presented?

    a. The presentation format is left to our discretion.

9. How should the UI (User interface) be implemented?

    a. Design the UI per class principles.

10. When and where is the app used?

    a. Courses and engineering fundamentals

    b. Examples include MATLAB, Python, and Java.

11. What information is presented to the user?

    a. The user views the submitted program and the accompanying feedback.

12. Do you have any documents or data before starting the project?

    a. Code and database files.

    b. Grail and Groovy domain classes (e.g., user.find(userid)).

    c. Domain files specifying data storage.

d. Modifiable domain files.

e. Descriptions of MYSQL data tables.

13. Are there previous versions we can reference?

No, Ureel provided guidance.

a. Start small and iteratively improve.

b. Create our framework.

14. What devices or interactions are envisioned for the app?

a. The app should be accessible through a web browser.

Interview 2 Notes

Parsing

We anticipate utilizing a pre-built parser as recommended by Ureel, and there is a possibility that they will furnish one for us. The parser should possess the capability to handle both Java and MATLAB, but for our specific scenario, the emphasis should be solely on Java.

Storage

Data is to be retained in the current database, with the option of client-side storage also considered acceptable.

Authentication

In the event of non-client side storage, a comprehensive user authentication process is not required if users are supplied with a unique link for each submission.

Held Time

Stored code and critiques should persist for a minimum of one semester before potential deletion, with an extended duration of up to three semesters to enable users to track their progress across multiple courses.

User Tracking

Users are to receive a score for their code(s), determined by factors such as the quantity of critiques, critique removal rates, and the time between submissions. The scoring may also take into account the severity of identified anti-patterns, including passes, failures, comments, errors, and warnings.

Critique Display

Critiques can be showcased in the form of an HTML page and/or a PDF document.