"Serious game to teach software testing"
Game Design Document

Concept

A player interacts with a board and pieces to hit specific targets and achieve the maximum score and learn about white-box testing.

Goals

- Learning about white box testing concepts, such as statement coverage, decision coverage, branch coverage and MCDC.
    - Other topics that may be learned include Game logic (specific to board game), Boolean logic and Code reading.

Platform Requirements

- Authentication, for students and teachers.
- The architecture of the capsule must allow for easy expansion – the prototype will be based around Checkers, but it should allow for adding new board games.

Game Mechanics

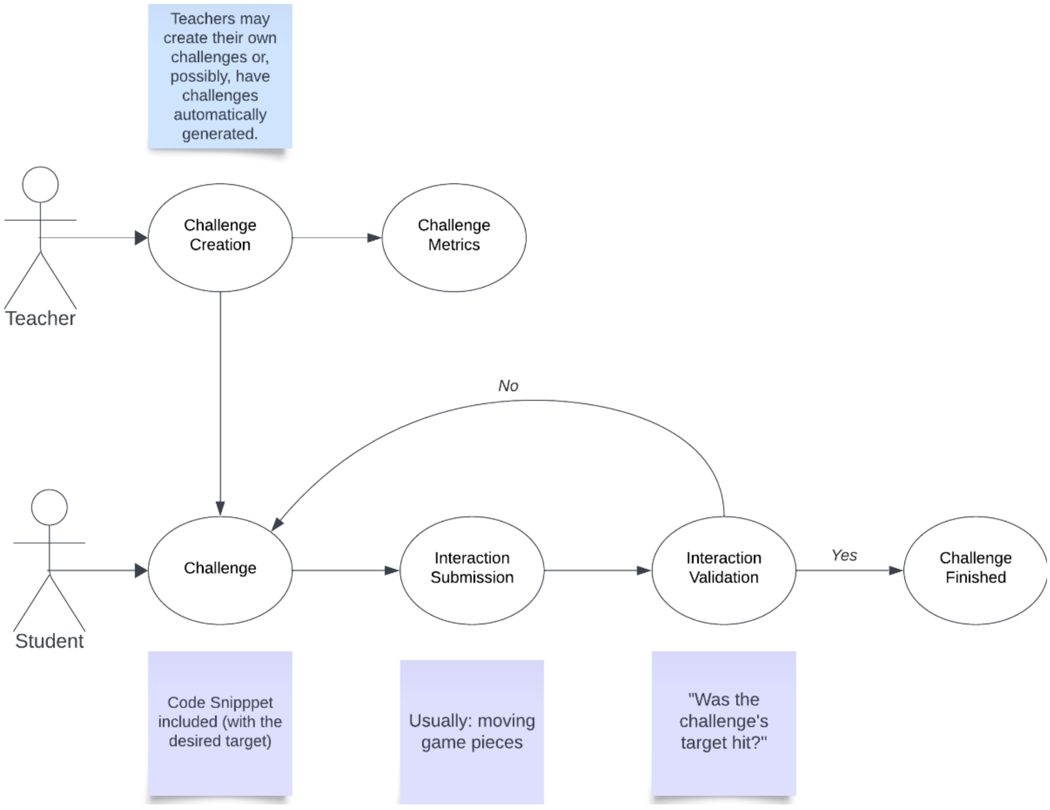
- *Challenges*: each challenge comprises a task or set of tasks to perform, most often based on a snippet of code.
    - A challenge includes: a code snippet, a (predisposed) board state and a target.
        - The code snippet **is shown to the player**, giving insight into what interaction(s) to perform.
        - A target is a specific line of code the player should hit via their interactions with the board.
            - **Example:** if the target is a specific line of code, and that line of code is inside a specific *If* block, the user is expected to create the conditions for the *If* block's condition to pass, thus, when the code is executed, the target is hit.
    - A student may *pass* a challenge *once.*
- *Difficulty Levels:* challenges will be designed around their target and scored appropriately.
    - Some challenges will be harder to beat *because* of what they require to beat – for example, a *statement coverage* target is easier to hit than a *decision coverage* target.
- *Score System:* passing challenges grants points!
    - *Not passing challenges* introduces penalizations (less points).

- *Achievement Challenges:* different strokes for different blokes, via restrictions such as:
    - Time ("You only have 15 seconds to pass this challenge");
    - Anti-targets ("You have to hit this line of code, but *this* line of code CANNOT be executed before it!");

Teacher User Perspective

- Teacher users design the challenges students perform.
    - Targets are specified on a specified collection of validation functions (the code snippets).
    - As expected, the game board state *can* be defined by the teacher, or provided in a default state.
- Teachers may check how their challenges have been received, via relevant metrics (how many students have passed it, how many students have failed it, et cetera);

Made a little use case diagram below to better show the users' flow within the game.


Teachers may create their own challenges or, possibly, have challenges automatically generated.
Teacher
Challenge Creation
Challenge Metrics
No
Student
Challenge
Interaction Submission
Interaction Validation
Yes
Challenge Finished
Code Snipppet included (with the desired target)
Usually: moving game pieces
"Was the challenge's target hit?"