Refactoring Opportunities

---

1. **[Back-end]** Add a **`User`** entity/object and scaffold a controller for it, to allow proper HTTP client methods (**`PUT, POST, GET, DELETE`**) for that entity/object.
    - **Improvement Opportunity:**
        - Currently not coded to allow for multiple users
        - Storing user information (ie. **`userAnswers`**) in the front-end
    - **How Refactoring Will Improve Current Code:**
        - If coded to include a **`User`** entity/object, then it will be easier to refactor in a future iteration to add functionality for multiple users
        - Increased separation of roles between the front- and back-end – the front-end will be used for mainly view-related roles, while the back-end will primarily be used for data storage and manipulation
2. **[Front-end]** Refactor the form to use Reactive Forms.
    - **Improvement Opportunity:**
        - While the code collects accurate information (ie. user's answers) as the user interacts with the web application, it does not take advantage of Angular's built-in Reactive Form capabilities
    - **How Refactoring Will Improve Current Code:**
        - Refactoring could allow for cleaner code
3. **[Testing]** Automate testing using Selenium.
    - **Improvement Opportunity:**
        - The application was tested manually
    - **How Automating the Testing Will Improve the Current Process:**
        - It will allow to test the application and track potential pain points more efficiently, especially if the app will be developed on a larger scale
4. **[Front-end]** Style changes, to make the application look more streamlined and modern:
    - Add icons
    - Add sliding animations for the questions
    - Change the style of the start, finish, and other navigation buttons