BRAVE

Explore. Innovate. Create.

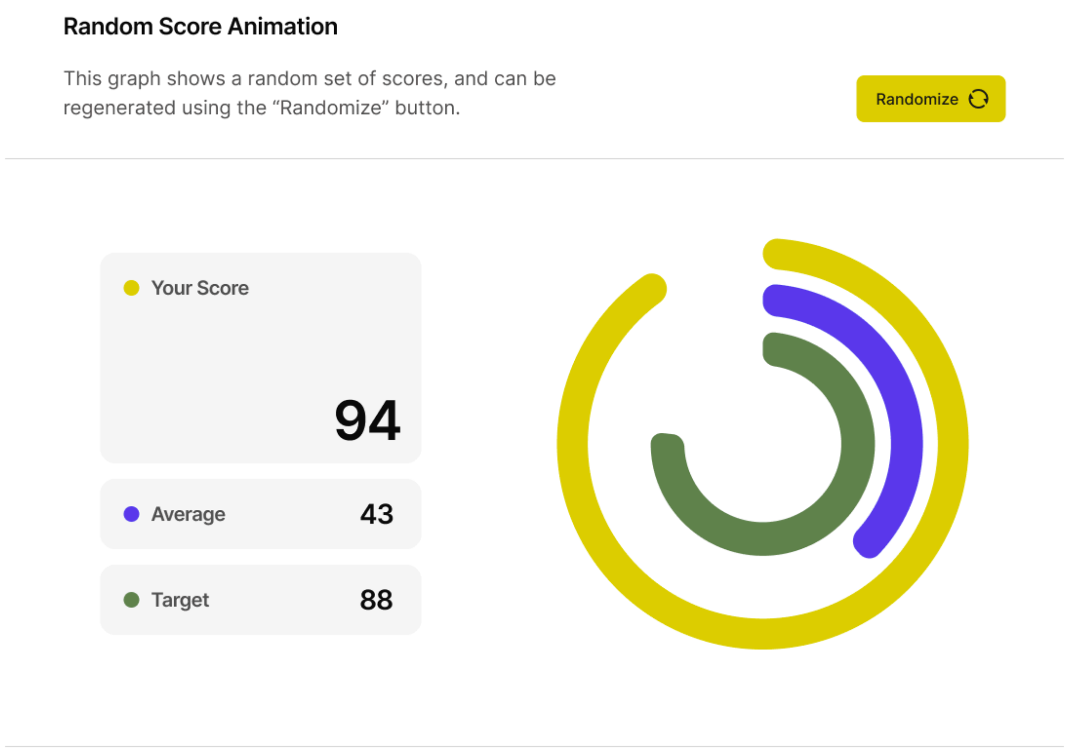
TYPESCRIPT CODE TEST

The following two part test is meant to showcase your knowledge of Full Stack TypeScript development. Please read the instructions carefully

Part 1 - Front-End

Part one is meant to test your understanding of Vue, Tailwind and TypeScript by building a randomizable Graph with a complex animation.

DESIGN:



Random Score Animation

This graph shows a random set of scores, and can be regenerated using the “Randomize” button.

[Figma Link]

OBJECTIVE

Please create a set of Vue components that accomplish the above design. Implement the provided design as a Vue.js component, utilizing Tailwind CSS for styling and scoped styles for component-specific styling.

The visualisation on the right is a representation of the values of the left. The values should be between 0-100, and the coloured rings should be a completed circle at 100.

Project Setup:

- Create a new Vue project using Vue CLI:

```
Unset
npm install -g @vue/cli
vue create random-score-animation
```

- Ensure Tailwind CSS is installed and configured correctly within the Vue project, including any necessary configuration files.

Component Structure:

- Create a primary Vue component (`RandomScoreAnimation.vue`) to encapsulate the entire design.
- Additionally create the additional components for each file:
  - `ScoreCircle` component: Render the circular progress visualization.
  - `ScoreCard` component: Display "Your Random Score," "Average," and "Target" values.
  - `RandomizeButton` component: Implement the "Randomize" button.

Styling:

- Use Tailwind CSS classes for all layout and styling where feasible.
- Adhere to the design's spacing, color, typography, and visual hierarchy.
- The colours used in the design are:
  - Joquil: `#edc727`;
  - Fern Green: `#588157`;
  - Magician Purple: `#331832`;
  - Savoury Blue: `#3b60e4`;
- Ensure responsiveness across screen sizes using Tailwind's responsive modifiers (e.g., `sm:`, `md:`, `lg:`).
- Pay close attention to:
  - Container and element spacing (padding, margins).
  - Text styles (font size, weight, color).
  - Color palette.
  - Alignment and layout.
  - Button styles (background, rounded corners).
- Implement simple interaction states (hover, focus, etc)

- Use scoped styles (`<style scoped>`) within each Vue component for component-specific styling not achievable with Tailwind.
- Example: Use `<style scoped>` in `ScoreCircle` for custom SVG styling and/or animations.

IMPLEMENTATION DETAILS:

- **RandomizeButton Component:**
  - Implement the "Randomize" button.
  - Use Tailwind classes for appearance.
  - On click this button should update both the `ScoreCard` and `ScoreCircle`, and on the `ScoreCircle` component.
- **ScoreCard Component:**
  - Display "Your Random Score," "Average," and "Target" values.
  - Use Tailwind classes for layout, spacing, and typography.
- **ScoreCircle Component:**
  - Use Vue props to receive score values to drive the visualization.
  - Apply colors as per the design.
- **Data Handling:**
  - The parent component (`RandomScoreAnimation.vue`) should manage score data.
  - Initialize scores with values from the design.
  - Pass scores as props to `ScoreCircle` and `ScoreCard`.
  - Handle the `"randomize"` event from `RandomizeButton` to update score values.

Bonus Task:

Render the circular progress visualization using SVG or Canvas, and implement an animation. The animation should start from the top center as an origin position, and expand clockwise to the relative position based on the related value as generated by the Randomize button component. Any changes to the props should transition the animation to the new values.

If you don’t intend to link the backend you develop in part two, you can implement the following:

- Implement "Randomize" button functionality:
  - Generate new random scores for "Your Random Score," "Average," and "Target" on click.
  - Update `ScoreCircle` and `ScoreCard` to reflect new data.

Code Quality Tips:

- Write clean, well-organized, and maintainable code.
- Ensure responsiveness and scalability.
- Follow Vue.js best practices.

- Use meaningful variable and component names.
- Comment code to explain complex logic.

BRAVE

Explore. Innovate. Create.

PART 2 - BACK END

OBJECTIVE

Please create a simple RESTful API using Node and Express that provides and manages score data as described below.

1. Set up a new Node.js project and install the necessary dependencies.
2. Create two API endpoints:
    - GET /api/scores: Returns the current ‘Score', 'Average', and 'Target' from a json file.
    - POST /api/scores/randomize: Generates new random values (between 0 and 100) for 'Your Score', 'Average', and 'Target', and returns the new values.
3. Implement basic error handling.

BONUS CHALLENGES

The following tasks Optional Bonus Challenges - Attempt these if you have extra time):

- Connect up these endpoints with the Vue front-end using a Pinia store.
- Implement JWT Bearer Token authentication.