CSS File:

1. Global Styles:

#root, html, body: These selectors reset the default margin and padding to 0, ensuring that no unexpected spacing exists. It also sets the height of the document to 100vh (100% of the viewport height), and the box-sizing: border-box ensures that padding and border are included in the element's total width and height.

2. App Container:

Background: The background uses a linear-gradient overlay (darkening the image) and an image from ./images/city.jpg. The background image covers the entire area (background-size: cover) and is centered (background-position: center).

Flexbox Layout: The display: flex makes the .app a flex container, centering its child elements both vertically and horizontally (justify-content: center, align-items: center).

Text Alignment: The text-align: center centers text horizontally within the container.

3. Card Component:

Background: A light background color (whitesmoke) is applied to the card.

Layout: display: flex and flex-direction: column arrange the contents of the card in a vertical direction, with items aligned in the center (align-items: center).

Dimensions: The width is set to 40% of its parent container and height to 20%. Padding of 2% is added around the content.

Rounded Corners & Shadow: The card has rounded corners with a radius of 25px, and a subtle shadow (box-shadow: 10px 10px) for a soft 3D effect.

4. Heading:

This ensures the heading element inside the card is vertically centered. It has a fixed height of 582px, and flex properties align its children elements.

5. Button Styling:

Appearance: The button has a white background (background-color: #ffffff) and rounded corners with a radius of 50px.

Flex Layout: The button is a flex container, centering its text both vertically and horizontally (justify-content: center, align-items: center).

Size: The button is given a height of 200px and width of 210px.

Border: It has a soft border (border: 1px solid rgba(22, 76, 167, 0.6)), giving it a subtle blue outline.

6. Button Text Styling:

Text Styling: The text inside the button is styled with a blue color (#164ca7), a small font size of 12px, medium weight (font-weight: 500), and slightly spaced letters (letter-spacing: 0.7px).

7. Button Hover Effects:

Rotate Animation: When the button is hovered, it rotates slightly (with a rotate animation) back and forth for a subtle effect.

Storm Animation: The text inside the button also animates with a "storm" effect (slightly shifting left and right).

8. Keyframe Animations:

Rotate Animation (@keyframes rotate): Rotates the button from 0 to 3 degrees, then to -3 degrees, then back to 0.

Storm Animation (@keyframes storm): Moves the text inside the button slightly left and right to give a "stormy" effect.

9. Responsive Design for small screens:

Media Query: This section targets screens with a width of 600px or smaller (e.g., mobile devices). It makes the card take up 80% of the screen width and increases the height to 30% for better mobile display.

JS File:

1. Imports:

React: The core React library.

useState, useEffect: React hooks. useState manages state, and useEffect handles side effects like data fetching.

axios: A library used for making HTTP requests.

'./App.css': CSS file to style the components.

2. State Initialization:

The advice state is initialized to an empty string. setAdvice is used to update the state.

3. fetchAdvice Function:

This function fetches data from the https://api.adviceslip.com/advice endpoint.

axios.get sends a GET request to the API.

Upon a successful response, the advice value from response.data.slip is extracted and stored in the state.

If there's an error, it's logged in the console.

4. useEffect Hook:

The useEffect hook is used to call fetchAdvice once when the component mounts (empty dependency array [] ensures it runs only once).

It ensures the advice is fetched automatically when the app starts.

5. Render JSX:

The returned JSX is the layout of the component.

The advice state is displayed inside an <h1> element with the class heading.

A button with the text "GIVE ME ADVICE!" calls fetchAdvice when clicked to fetch a new piece of advice.

6. Export:

The component is exported so it can be used in other parts of the application.