

Subject Name: Front End Engineering

Subject Code: CS186

Cluster: iGamma

Group: G19

Department: DCSE



CHITKARA
UNIVERSITY

Password Generator

Submitted By:
Kavya Nagpal
2110992030
G19

Submitted To:
Ms. Pritpal Kaur

1. Introduction:

The "Password Generator" app is a web application built using React, a popular JavaScript library for building user interfaces. This report provides an overview of the app's structure, functionality, and key components.

2. Application Structure:

The application is structured as a React component, named App, which serves as the main container for all the features. It consists of several key components and functions:

The app consists of the following components:

- App Component:- This is the main component of the application, which manages the state and rendering of the entire app.
- Password Strength Indicator:- A component that visually indicates the strength of the generated password.

```
ete > src > components > JS StrengthChecker.js > ...
1  import React from "react";
2
3  const PasswordStrengthIndicator = ({ password = "" }) => {
4    const getPasswordStrength = () => {
5      const passwordLength = password.length;
6
7      if (passwordLength < 1) {
8        return "";
9      } else if (passwordLength < 4) {
10       return "Very Weak";
11     } else if (passwordLength < 8) {
12       return "Poor";
13     } else if (passwordLength < 12) {
14       return "Medium";
15     } else if (passwordLength < 16) {
16       return "Strong";
17     } else {
18       return "Very Strong";
19     }
20   };
21
22   const passwordStrength = getPasswordStrength();
23
24   if (!passwordStrength) return <React.Fragment />;
25
26   return (
27     <div className="password-strength">
28       Strength: <span style={{ fontWeight: "bold" }}>{passwordStrength}</span>
29     </div>
  
```

- Button Component:-A reusable button component used for actions like copying the password and generating a new password.

```
ete > src > components > JS Button.js > ...
1  const Button = ({ onClick, text, customClass }) => {
2    return (
3      <button className={customClass} onClick={onClick}>
4        {text}
5      </button>
6    );
7  };
8
9  export default Button;
10
```

- **Checkbox Component:-** A component to handle checkboxes for selecting password criteria.

```
const Checkbox = ({ title, state, onChange }) => {
  return (
    <div>
      <input type="checkbox" onChange={onChange} checked={state} />
      <label>{title}</label>
    </div>
  );
};

export default Checkbox;
```

3. Functionality:

The app serves as a password generator and strength checker. It allows users to:

- Generate a random password based on selected criteria, such as character length, uppercase letters, lowercase letters, numbers, and symbols.
- Adjust the character length of the password using a range input.
- Enable or disable password criteria by toggling checkboxes.
- Copy the generated password to the clipboard.
- View the strength of the generated password.
- Receive error messages in case of issues with the password generation.

4. State Management:

The app utilizes the React `useState` hook to manage the following pieces of state:

- `length`:- Represents the character length of the generated password.

```
<div className="charlength">
  <span>
    <label>Character Length</label>
    <label>{length}</label>
  </span>
  <input
    type="range"
    min="4"
    max="20"
    value={length}
    onChange={(e) => setLength(e.target.value)}
  />
</div>
```

- `checkboxData`:- An array of objects representing the state of each checkbox (uppercase letters, lowercase letters, numbers, and symbols).

```
<div className="checkboxes">
  {checkboxData.map((checkbox, index) => {
    return (
      <Checkbox
        key={index}
        title={checkbox.title}
        onChange={() => handleCheckboxChange(index)}
        state={checkbox.state}
      />
    );
  })}
</div>
```

- `copied`:- A boolean state variable to track whether the password has been copied to the clipboard.

```
const [copied, setCopied] = useState(false);

const handleCheckboxChange = (i) => {
  const updatedCheckboxData = [...checkboxData];
  updatedCheckboxData[i].state = !updatedCheckboxData[i].state;
  setCheckboxData(updatedCheckboxData);
};
```

- `password`:- Stores the generated password.

```
{/* Generate Button */}
<Button
  text="Generate Password"
  onClick={() => generatePassword(checkboxData, length)}
  customClass="generateBtn"
```

- `errorMessage`:- Stores any error messages related to password generation.

```
{/* Error Handling */}
{errorMessage && <div className="errorMessage">{errorMessage}</div>}
```

5. Event Handling:

The app uses event handling functions, including:

1. Checkbox Handling:

- The `handleCheckboxChange` function allows users to toggle the state of each character type checkbox.

```
<div className="checkboxes">
  {checkboxData.map((checkbox, index) => {
    return (
      <Checkbox
        key={index}
        title={checkbox.title}
        onChange={() => handleCheckboxChange(index)}
        state={checkbox.state}
      />
    );
  })}
</div>
```

2. Copy to Clipboard:

- The `handleCopy` function uses the `navigator.clipboard` API to copy the generated password to the clipboard and sets the `copied` state to display a confirmation message.

```
const handleCopy = () => {
  navigator.clipboard.writeText(password);
  setCopied(true);

  setTimeout(() => {
    setCopied(false);
  }, 1000);
};
```


3. Password Generation:

- The `usePasswordGenerator` custom hook is used for generating passwords. It provides the `password`, `errorMessage`, and `generatePassword` function.
- `generatePassword` is called when the "Generate Password" button is clicked and uses the selected options to generate a password.

```
{/* Generate Button */}  
<Button  
  text="Generate Password"  
  onClick={() => generatePassword(checkboxData, length)}  
  customClass="generateBtn"
```

6. Design and Styling:

The app employs CSS for styling and provides a clean and user-friendly interface. It uses CSS classes and layout containers for structuring the elements.

```
c > # styles.css >   
*  
{  
  font-family: Arial;  
}  
  
.container {  
  background-color: #000000;  
  padding: 24px;  
}  
  
.header {  
  color: rgb(255, 255, 255);  
  display: flex;  
  justify-content: space-between;  
  font-size: 20px;  
  font-weight: 700;  
  padding-bottom: 20px;  
}  
  
button {  
  padding: 10px;  
  border-radius: 5px;  
  border: none;  
  background-color: #ff0000;  
  color: white;  
  text-transform: uppercase;  
  font-weight: 700;  
  cursor: pointer;  
}
```

7. Functional Flow:

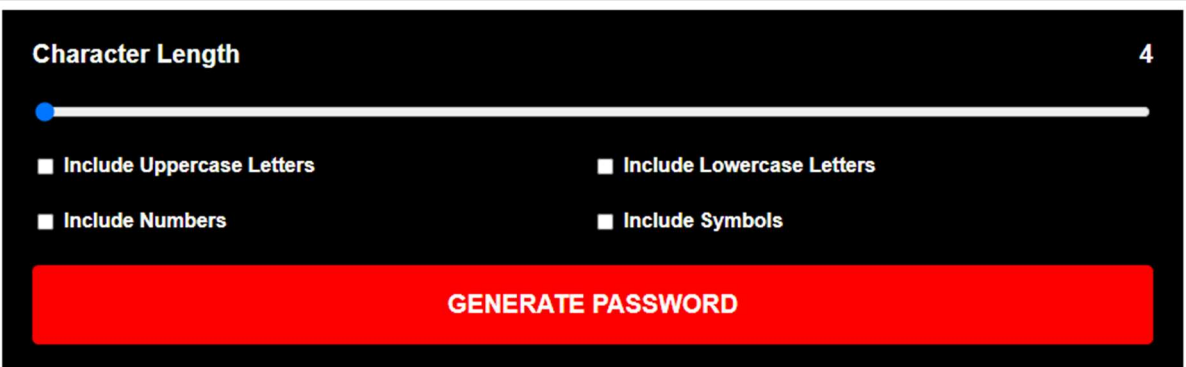
- When the application loads, the default password length is set to 4, and all character type checkboxes are initially unchecked.
- Users can adjust the password length by using a slider input.
- Users can select character types by checking the corresponding checkboxes.
- Clicking the "Generate Password" button triggers password generation.
- The generated password is displayed, along with an option to copy it to the clipboard.
- The password strength indicator provides feedback on the generated password's strength.
- Any errors in the password generation process are displayed as error messages.

8. Conclusion:

The "Password Generator" app is a functional and user-friendly tool for generating and assessing the strength of passwords. It provides a simple yet effective way for users to create strong and secure passwords tailored to their requirements.

Users can customize their passwords' length and character types and receive instant feedback on the password's strength. Additionally, they can copy the generated password to the clipboard for convenience. Error handling ensures that users are aware of any issues during password generation. This application can be further enhanced and styled for a more user-friendly experience.

Enhancements for future development could include the ability to save generated passwords and improve the visual indicators of password strength. Additionally, incorporating testing and validation for password criteria and generating passwords could further enhance the app's reliability.



The screenshot displays the user interface of the Password Generator app. It features a dark-themed layout with a black background. At the top, the text "Character Length" is displayed in white, followed by a horizontal slider bar with a blue dot at the start and the number "4" at the end. Below the slider, there are four toggle switches arranged in a 2x2 grid, each with a white square icon and white text: "Include Uppercase Letters", "Include Lowercase Letters", "Include Numbers", and "Include Symbols". At the bottom of the interface is a prominent red rectangular button with the text "GENERATE PASSWORD" in white, uppercase letters.