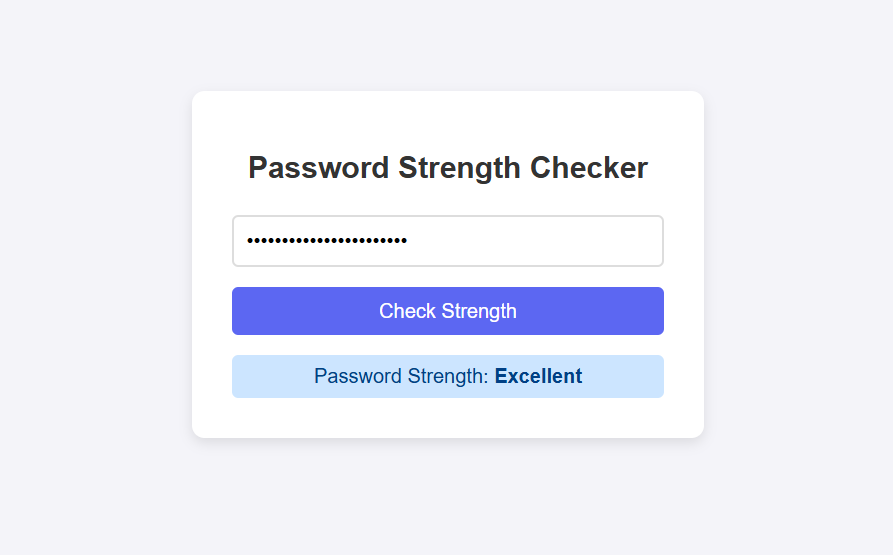
**Report**

1. **Task Description**

The task was to build a **Password Strength Validator** using the **MERN stack** (MongoDB, Express.js, React, Node.js). The application evaluates password strength based on length, complexity, and character types, categorizing it as **Weak**, **Good**, **Strong**, or **Excellent**. The frontend communicates with the backend to provide real-time feedback, ensuring a responsive and user-friendly experience.

1. **Task Output Screenshot**



1. **Widget/Algorithm Used In Task**
2. **Password Evaluation Algorithm**: The core of the application is the password strength evaluation algorithm. It checks the password against several criteria, such as length, the presence of uppercase and lowercase letters, digits, and special characters, assigning a score based on these factors to determine the password strength.
3. **Regular Expressions (Regex)**: Regex is used to check the presence of specific character types in the password, such as lowercase letters (/[a-z]/), uppercase letters (/[A-Z]/), digits (/\d/), and special characters (/[!@#$%^&\*(),.?":{}|<>]/). These patterns help validate password complexity.
4. **Backend API (Express.js)**: The backend, built with Express.js, provides the /api/validate-password endpoint. When a password is submitted, it is sent to the server via a POST request, where the strength is evaluated and a response is sent back to the frontend.
5. **Frontend (React.js)**: The frontend uses React to create a dynamic user interface. It captures user input, sends it to the backend, and displays the password strength. React’s state management handles dynamic updates based on user interaction.
6. **CSS for User Feedback**: The frontend uses CSS to visually indicate the password strength. Different colors and styles (e.g., red for weak, green for excellent) are applied dynamically based on the evaluation results, offering users clear visual feedback.