Great! Now we have our files created. Let's explain the implementation:

**Dynamic Form Builder Implementation**

I've created a complete solution for the dynamic form builder that meets all the requirements. Here's a breakdown of what I've built:

**1. Project Structure**

* **index.html**: The main HTML file that loads React, ReactDOM, and Babel for JSX support
* **styles.css**: CSS styles for the form and its components
* **app.js**: Contains the React components for the dynamic form builder

**2. Components**

* **App**: The main component that defines a sample form schema and handles form submission
* **DynamicForm**: The reusable component that takes a schema and renders a dynamic form

**3. Features Implemented**

**Field Support**

The DynamicForm component supports all required field types:

* Text input (type: "text")
* Email input (type: "email")
* Number input (type: "number")
* Checkbox (type: "checkbox")
* Radio buttons (type: "radio", with options)

**Validation Support**

The component implements comprehensive validation:

* Required fields (required: true)
* Min/max length for text fields (minLength, maxLength)
* Min/max value for numbers (min, max)
* Email format validation for email fields
* Shows clear error messages for invalid inputs

**Default Values**

* The form initializes with default values based on the field type
* For checkboxes, the default is false
* For text, email, number, and radio fields, the default is an empty string

**Form Submission**

* Validates all inputs on submit
* Shows error messages for invalid or missing inputs
* Calls the onSubmit callback with the form data when valid

**Bonus: Local Storage Support**

* The form data is saved to localStorage when the useLocalStorage prop is set to true
* The form automatically reloads saved data when the page is refreshed

**4. How to Use the Component**

To use the DynamicForm component, you need to provide a schema object with a fields array. Each field in the array should have:

* name: The field name (used as the form field's name attribute)
* type: The field type (text, email, number, checkbox, radio)
* label: (Optional) The display label for the field
* Additional validation properties as needed (required, minLength, maxLength, min, max)
* For radio buttons, an options array with the available options

Example usage:

Jsx

Copy

Paste

<DynamicForm

schema={formSchema}

onSubmit={handleFormSubmit}

useLocalStorage={true}

/>

**5. How to Test**

You can test the application by opening the index.html file in a web browser. The form will be rendered with the example schema defined in the App component. You can:

1. Fill out the form fields
2. Test validation by submitting with invalid or missing data
3. Test localStorage persistence by refreshing the page after entering data

**6. Additional Notes**

* The form uses real-time validation when a field loses focus (onBlur)
* Fields are marked as "touched" when the user interacts with them
* Error messages only appear for touched fields to avoid overwhelming the user
* The form has responsive styling and clear visual feedback for errors

This implementation fulfills all the requirements of the task, including the bonus localStorage persistence feature