**Formik**

1. **Task Description:**

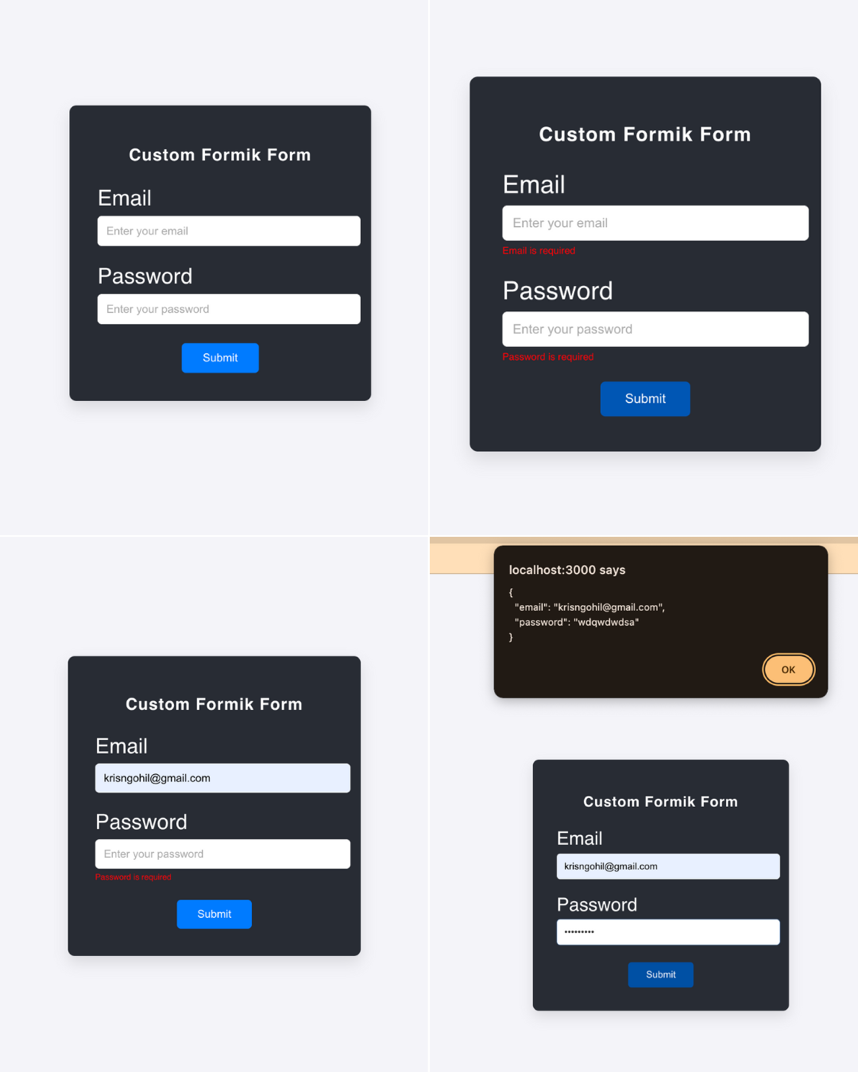
The task involves creating a custom form using Formik with validation using Yup. The form contains two fields: email and password. The goal is to:

* Create reusable form input components using Formik’s useField hook.
* Implement validation for both fields (email and password) using Yup, including required and format validation for the email, and minimum/maximum length for the password.
* Display error messages dynamically based on validation results.
* Use a custom CustomInput component to manage the input fields with validation and error handling.

**The core features of this task include:**

* **Custom Input Component:** A reusable input component created using Formik’s useField hook, which takes care of field values, validation, and error display.
* **Validation:** Email and password fields are validated with Yup schema, ensuring proper input format and constraints.
* **Error Handling:** Error messages are displayed dynamically beneath each field when validation fails.
* **Form Submission:** On successful validation, the form values are alerted in JSON format.

1. **Task Output Screenshot**



1. **Widget/Algorithm Used In Task**

**Widgets/Components Used:**

* **Formik:**
  + Formik is used for managing form state and validation.
  + Field is used to automatically handle field values and validation.
  + useField is used for creating custom form components with validation and error display.
* **Yup:**
  + Yup is used to define a validation schema for the email and password fields, ensuring the data conforms to specific requirements (email format, password length).
* **Input Fields (HTML <input>):**
  + HTML input fields are used for entering the email and password values.
  + These fields are dynamically controlled by Formik.
* **CustomInput Component:**
  + A custom component for handling input fields with validation using Formik’s useField hook.
  + Handles displaying errors, and conditionally applies the error class to highlight invalid fields.
* **Error Message (Conditional Rendering):**
  + Conditional rendering is used to display error messages when the corresponding field has been touched and contains an error.
  + The meta.touched && meta.error condition is used to display error messages only after the user has interacted with the field.
* **Validation Schema (Yup):**
  + Yup.object() is used to define a validation schema for the form.
  + The email field uses .email() for format validation and .required() for ensuring the field is not left empty.
  + The password field uses .min(), .max(), and .required() for enforcing password length constraints and ensuring the field is not empty.
* **Button (HTML <button>):**
  + The button is used to submit the form, triggering Formik’s onSubmit function.
  + It will only be active when the form passes validation, and clicking it will alert the form values in JSON format.

**Algorithm/Logic:**

* Validation: Formik integrates with Yup to validate input values for both email and password fields. If the form values do not meet the defined validation rules, Formik triggers error messages to inform the user.
* Custom Input Handling: The CustomInput component handles the state for the form fields, including their values, error messages, and validation state. It makes the component reusable and scalable across forms with similar structure.
* Form Submission: Once all fields are valid, the form is submitted, and the entered values are shown in the form of an alert.