# React-Handling Form Submission

IIHT

Time To Complete: 10 to 12 hr

# CONTENTS

# 1 Project Abstract

Forms are a critical part of many web applications, and managing their state, validation, and submission efficiently is key to creating robust user experiences. This project introduces **form handling in React** using controlled components, dynamic input generation, validation, and integration with an API (JSON Server). You will build a dynamic form where input configurations are abstracted from logic, showcasing how to handle user input, validate form fields in real time, and submit the data asynchronously to a backend service.

# 2 Problem Statement

You are tasked with creating a **Dynamic Form Submission** application in React. The form should:

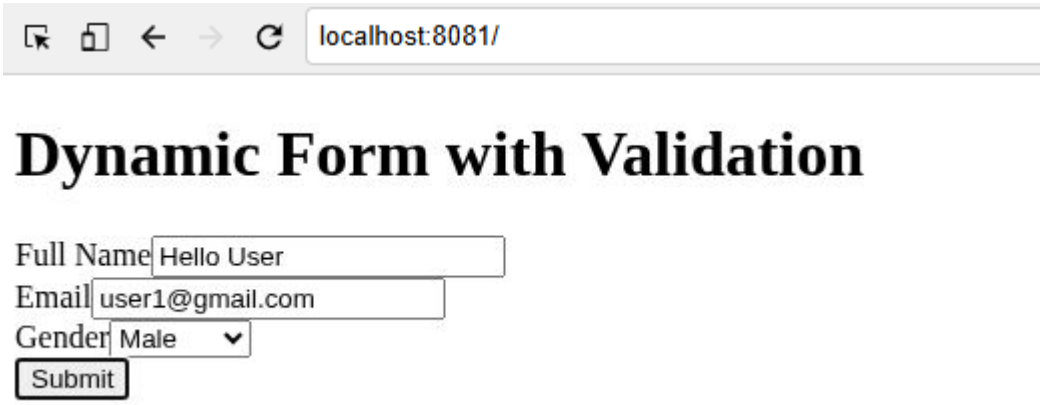
- Dynamically render form fields based on a configuration array.
- Use controlled inputs to manage form state.
- Perform validation for required fields both onBlur and onSubmit.
- Show contextual error messages only after fields are touched or on form submission.
- Submit the collected data to a mock backend (`http://localhost:4000/formData`) using the `fetch` API.
- Disable the submit button until the form is valid.

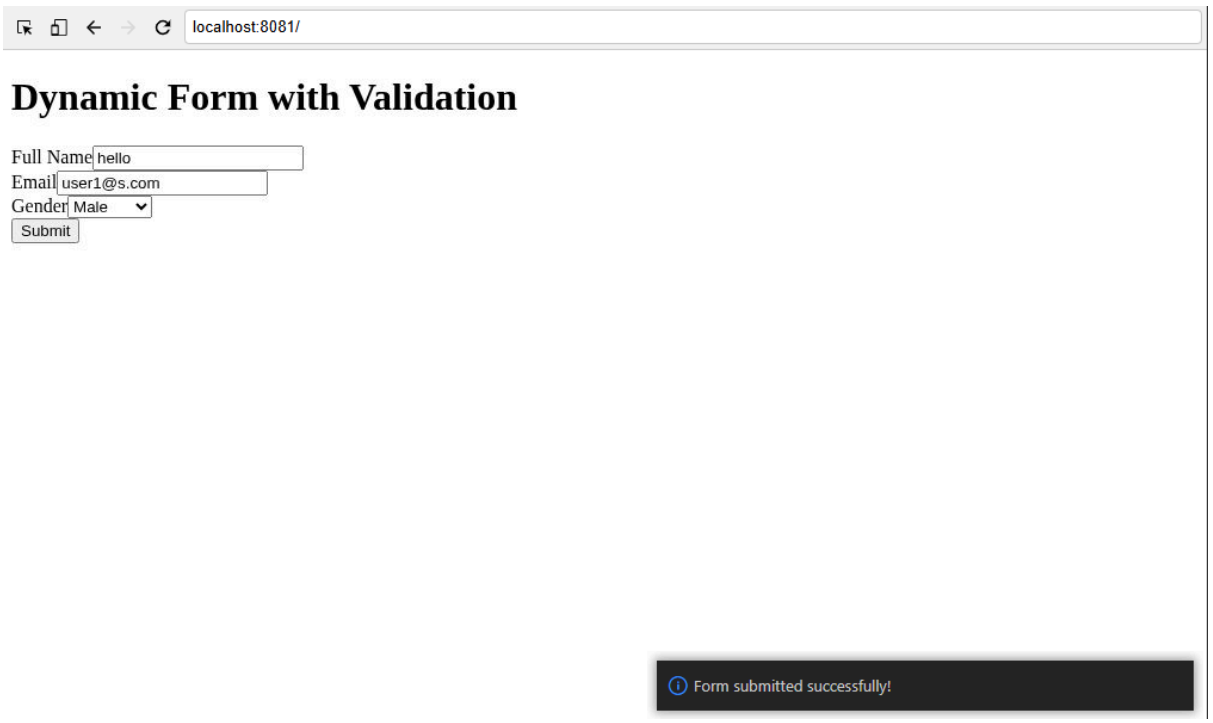
This project helps reinforce React's best practices for form handling, validation flow, and integration with external data submission.

# 3 Proposed React-Handling Form Submission Application Wireframe

UI needs improvisation and modification as per given use case and to make test cases passed.

## 3.1 ScreenShots

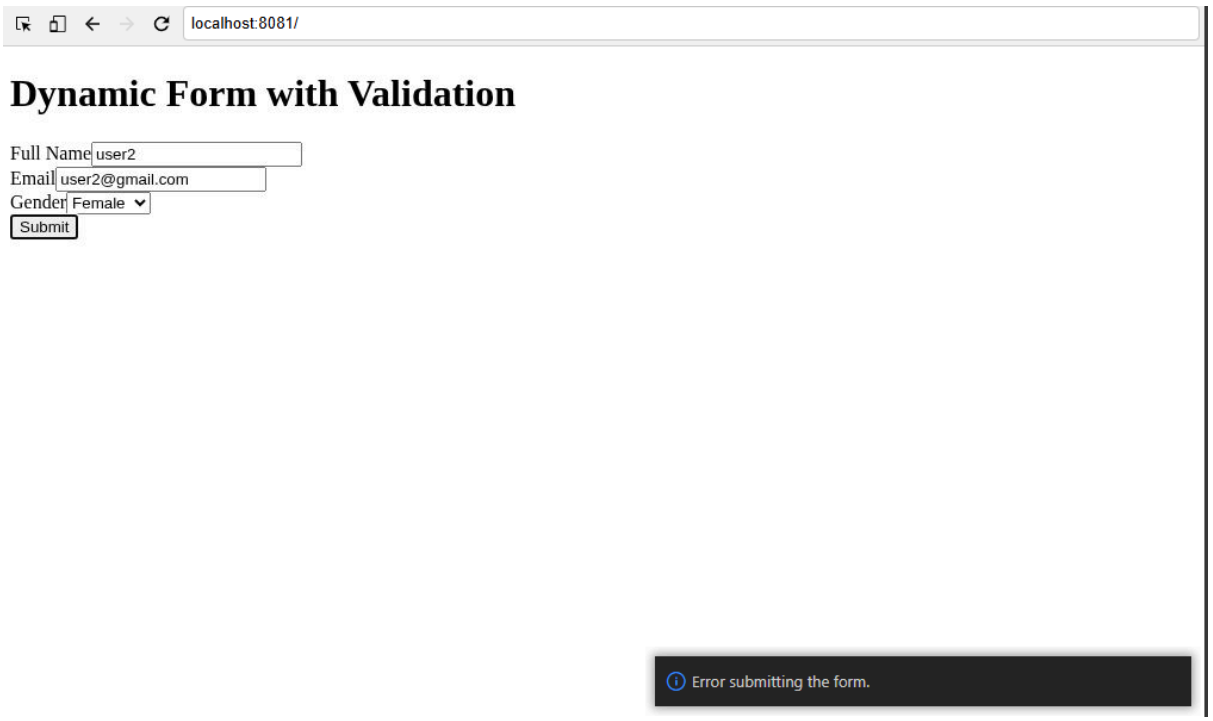




localhost:8081/
Dynamic Form with Validation
Full Name hello
Email user1@s.com
Gender Male
Submit
Form submitted successfully!

***Valiations***


localhost:8081/
Dynamic Form with Validation
Full Name hello
Email user1
Please include an '@' in the email address. 'user1' is missing an '@'.


localhost:8081/
Dynamic Form with Validation
Full Name hello
Email user1@
Please enter a part following '@'. 'user1@' is incomplete.




# 4 BUSINESS-REQUIREMENT:

As an application developer, develop the React-Handling Form Submission (Single Page App) with below guidelines:

| User Story # | User Story Name | User Story |
|---|---|---|
| US_01 | Welcome Page | As a user I should be able to visit the welcome page as the default page.<br><br>Acceptance criteria:<br><br>1. Dynamically render form fields using configuration data.<br>2. Validate required fields (name, email, gender).<br>3. Show validation messages **only** when fields are touched or after submission.<br>4. On submit:<br>● Prevent default form refresh<br>● Validate the form again<br>● Send data via a **POST** request to a server (e.g., JSON Server)<br>● Show success or failure message based on response<br>5. Disable the **Submit** button until the form is valid. |

**State & Functionality Overview:**

**Form Component**

**State Variables:**

1. `formData`: Object containing values for each field.
2. `errors`: Object containing validation messages.
3. `touchedFields`: Tracks if a user has interacted with a field.
4. `isFormValid`: Boolean indicating whether form passes all validation rules.
5. `isFormSubmitted`: Boolean tracking if the form has been submitted.

**Methods & Behavior:**

6. `handleChange(e)`: Updates `formData` on user input.
7. `handleBlur(e)`: Marks field as "touched" for error visibility.
8. `validateForm()`: Checks required fields and returns validity.
9. `handleSubmit(e)`: Validates again and submits via `fetch()`.

**DynamicInput Component:**

1. Accepts props:
   - `config`: Field settings (type, label, name, options)
   - `value`: Current field value
   - `onChange`, `onBlur`: Input events
   - `error`: Optional error message
2. Renders:
   - `<input>` for text/email fields
   - `<select>` for dropdowns

**HTML Structure:**

1. **App Component:**
   - Wrap everything inside a `<div>` with class `"App"`.
   - Add:
     - ➔ `<h1>` with the heading: **"Dynamic Form with Validation"**
     - ➔ `<Form />` component to render the form

| | | |
|---|---|---|
| | | **2. Form Component:**<br>● Use a `<form>` element to wrap all fields.<br>● For each field:<br>➔ Use a `<label>` to show the field label (e.g., "Full Name", "Email", "Gender")<br>➔ Use the appropriate input type based on field:<br>➢ `<input type="text">` for full name<br>➢ `<input type="email">` for email<br>➢ `<select>` for gender<br>❖ Dropdown should have:<br>➢ **"Select..."**, **"Male"**, **"Female"**<br>● Add a `<button type="submit">` at the end.<br>➔ Disable the button when the form is invalid.<br>● Show error messages conditionally (only after touching or submitting).<br>**3. DynamicInput Component:**<br>● Text & email inputs:<br>➔ Labeled input fields with required attributes<br>● Dropdown:<br>➔ Labeled `<select>` with options<br>**Dynamic Behavior:**<br>1. All fields are **controlled components**, bound to `formData`.<br>2. Validation runs **on each field change** and on **form submit**.<br>3. Error messages appear:<br>● If the user touches the field (onBlur)<br>4. Submit button remains **disabled** until the form is valid.<br>**** Kindly refer to the screenshots for any clarifications. **** |
| | | |

## 5 CONSTRAINTS

1. You should be able to press the “TAB” key and “SHIFT + TAB” to navigate from top field to bottom field and vice-versa.

# 6 MANDATORY ASSESSMENT GUIDELINES

1. All actions like build, compile, running application, running test cases will be through Command Terminal.
2. To open the command terminal the test takers, need to go to
   Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.
3. This editor Auto Saves the code.
4. If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page)compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.
5. These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.
6. This is a web-based application, to run the application on a browser, use the internal browser in the workspace. Click on the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.
   Note: The application will not run in the local browser
7. You can follow series of command to setup React environment once you are in your project-name folder:
   a. npm install -> Will install all dependencies -> takes 10 to 15 min
   b. npm run start -> To compile and deploy the project in browser. You can press <Ctrl> key while clicking on localhost:8080/8081 to open project in browser -> takes 2 to 3 min
   c. npm run json-start -> As we are using a json server to mimic our db.json file as a database. So, this command is useful to start a json server.
   d. npm run jest -> to run all test cases and see the summary. It takes 5 to 6 min to run.
   e. npm run test -> to run all test cases. It is mandatory to run this command before submission of workspace -> takes 5 to 6 min

8. **Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on "Submit Assessment" after you are done with code.**