Reactive forms

IIHT

Time To Complete: 10 to 12 hr

Contents

1 Project Abstract

Form handling is a crucial part of modern web applications. This assignment focuses on implementing **Reactive Forms with Validations** in Angular. Reactive Forms provide a model-driven approach to handling form inputs, offering better scalability and testability.

The objective is to create a user registration form where the inputs are validated using Angular’s Reactive Form features, ensuring real-time feedback and error handling.

2 Problem Statement

You are tasked with developing a **Reactive Form SPA** using Angular.

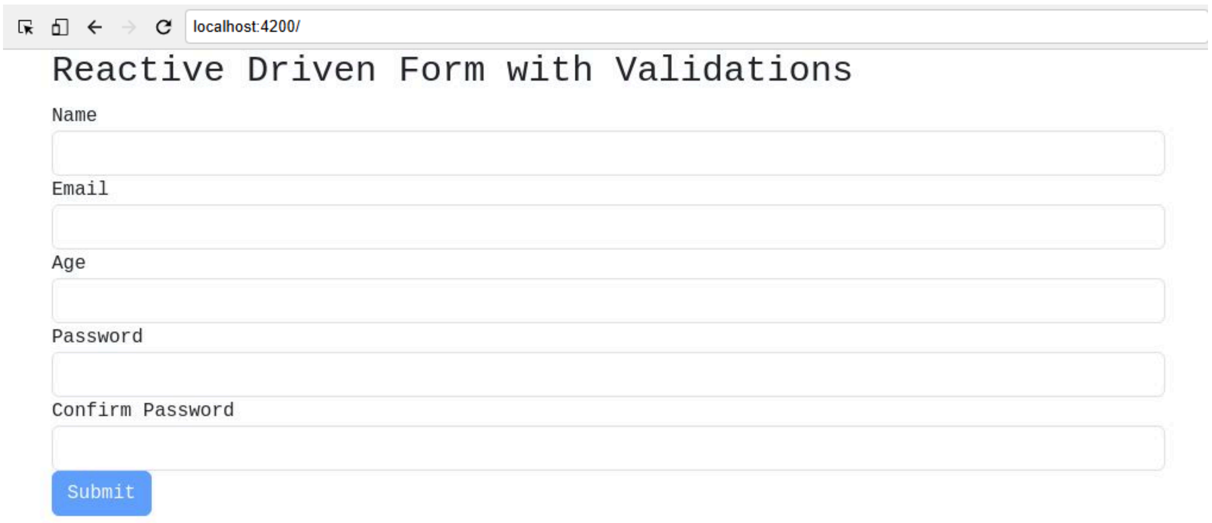
The application should:

- Present a form with fields: **Name, Email, Age, Password, Confirm Password**.
- Apply appropriate validations to each field.
- Provide real-time validation feedback.
- Disable form submission until all validations are satisfied.

3 Proposed Reactive forms Application Wireframe

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

3.1 ScreenShots


localhost:4200/

Reactive Driven Form with Validations

Name

Email

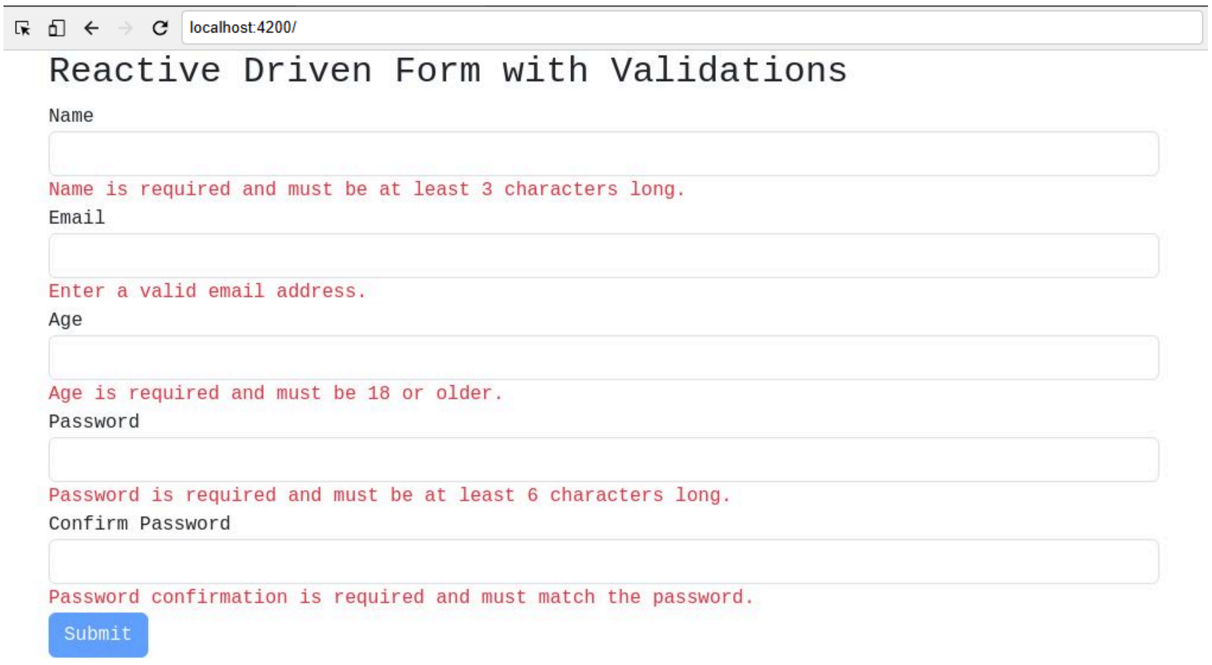
Age

Password

Confirm Password

Submit

Validations


localhost:4200/

Reactive Driven Form with Validations

Name

Name is required and must be at least 3 characters long.

Email

Enter a valid email address.

Age

Age is required and must be 18 or older.

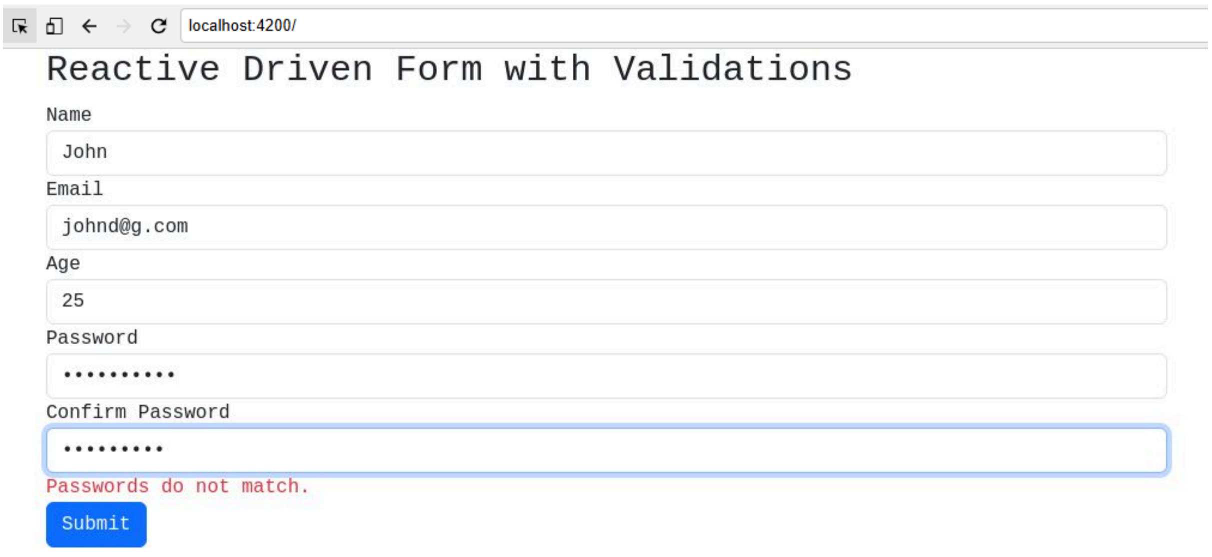
Password

Password is required and must be at least 6 characters long.

Confirm Password

Password confirmation is required and must match the password.

Submit



localhost:4200/

Reactive Driven Form with Validations

Name: John

Email: johnd@g.com

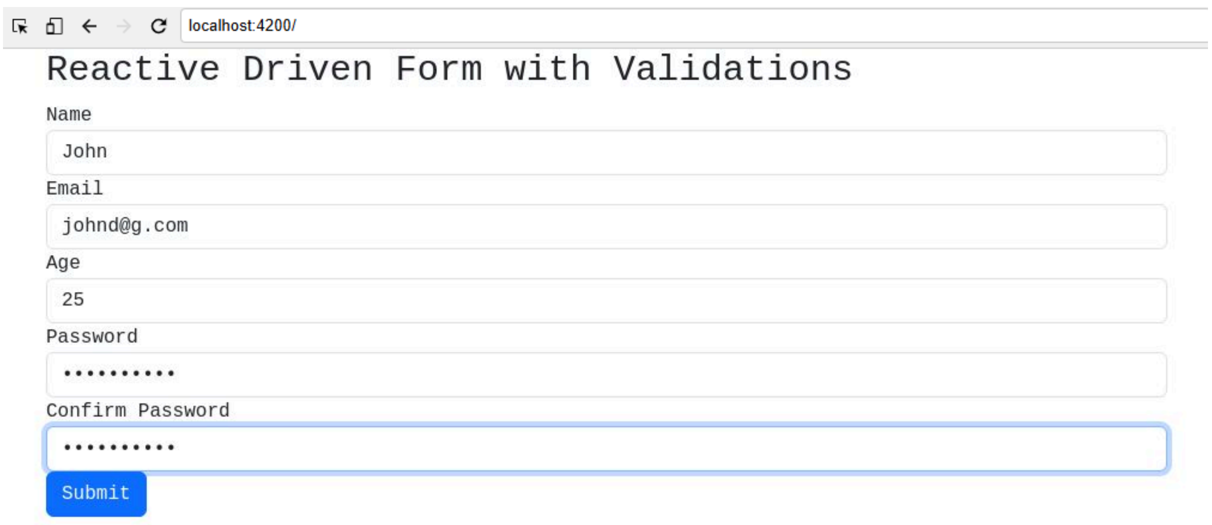
Age: 25

Password: ••••••••••

Confirm Password: •••••••••

Passwords do not match.

Valid Entries



localhost:4200/

Reactive Driven Form with Validations

Name: John

Email: johnd@g.com

Age: 25

Password: ••••••••••

Confirm Password: ••••••••••

4 Business-Requirement:

As an application developer, develop the Reactive forms (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. Acceptance criteria: **Reactive Driven Form with Validations**

User Story

As a user, I should be able to fill out a registration-style form with real-time validations using Angular’s Reactive Forms.

Acceptance Criteria

1. Display a heading: “Reactive Driven Form with Validations”
2. Provide a form with the following fields:
   - Name
   - Email
   - Age
   - Password
   - Confirm Password
3. Apply real-time field-level validations:
   - Show error messages only when a field is invalid and touched
   - Confirm Password must match the Password
4. The Submit button should remain disabled if the form is invalid.
5. On valid form submission, log the form values to the console.

Component Overview

Selector: app-root
Template Used: app.component.html

Reactive Form Setup

The form is created using Angular’s Reactive Forms API, specifically by:
- Defining a FormGroup that holds multiple FormControls.
- Each form control (e.g., name, email, age) is initialized with:
  - A default value
  - An array of validation rules like required, minLength, email, or min
- A separate control is used for confirming the password, which must be compared manually to the original password field.

The form is entirely reactive, meaning:
- It is defined and managed in the component’s TypeScript class
- Angular automatically tracks field states (valid, invalid, touched)

Method - onSubmit()

- Triggered when the user submits the form.
- If the form is valid:
  - The data is logged to the console.
- If the form is invalid:
  - A warning message is shown in the console.

HTML Structure (app.component.html)

• Form Container: Use a <div class="container">
• Heading: <h2> with “Reactive Driven Form with Validations”
• Form Element: <form> bound to the FormGroup and listens for (ngSubmit)

Form Fields

Each form field is inside a <div class="form-group"> block with:
- A <label> describing the field
- An <input> tied to a specific form control
- A <div class="text-danger"> for validation feedback

Field Descriptions

🔹 Name Field (name) with label “Name”
- Required and minimum 3 characters

- Incase of error, it should show “**Name is required and must be at least 3 characters long.**”

🔹 Email Field (email) with label “Email”
- Required and valid email format

- Incase of error, it should show “**Enter a valid email address.**”

🔹 Age Field (age) with label “Age”
- Required and must be 18 or older

- Incase of error, it should show “**Age is required and must be 18 or older.**”

🔹 Password Field (password) with label “Password”
- Required and minimum 6 characters

- Incase of error, it should show “**Password is required and must be at least 6 characters long.**”

🔹 Confirm Password Field (confirmPassword) with label “Confirm Password”
- Required and must match Password

- Incase of error, it should show “**Password confirmation is required and must match the password.**”

- If passwords do not match, it should show “**Passwords do not match.**”

Password Match Check

• Shows an extra message below Confirm Password if passwords do not match.

		Submit Button • Labeled 'Submit' • Disabled until the form is fully valid Dynamic Behavior - Angular evaluates field validity in real-time. - Error messages show only after a field is touched. - Password mismatch warning is shown separately. - Submit button activates only when all checks pass.

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) then you need to use CTRL+Shift+B-command 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 Angular 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 the <Ctrl> key while clicking on localhost:4200 to open the project in the browser -> takes 2 to 3 min.**
   c. **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.**
9. **You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.**