

|  |  |  |  |
| --- | --- | --- | --- |
| Academic Year | 2023 | | |
| Semester | Fall | Winter | Summer |
| Course Code - Name | WEB 422 - Web Programming for Apps and Services | | |
| Instructor | Dr. Razi Iqbal | | |
| Assessment | Test |  | |
| Deadline | Wednesday, October 18, 2023 | | |

**Student ID 127930212**

**Student Name Sangin(Jenny) Lee**

**Test**

The main purpose of this test is to check your knowledge of Bootstrap, React and Form hooks. This test assumes that you have already setup your environment.

**Instructions:**

* This test is 10% of your total marks in this course.
* Please use this document to provide your answers. Scroll down to the Solution section and paste your code for required files in that section. Do not forget to provide the screenshot of your final output. Upload this word document on Blackboard as the submission.
* Submissions through emails will not be accepted.
* Students having exactly similar code will get a straight 0.
* The deadline for submission of this assignment is midnight Wednesday, October 18, 2023.

**Submission Details:**

For this test, you are required to create a simple react app (make sure to install bootstrap and react-hook form frameworks). Once you have setup the app, create a Registration.js file in src directory that should return RegistrationForm(). All of your code should go inside this function including all the JSX, constants and other functions.

You are most welcome to create a style.css in src directory as well which can then be called in Registration.js file.

***For submission, please use the Solution section below to submit the code for Registration.js and style.css files. Please DO NOT create or submit code for any other file.***

**Question**

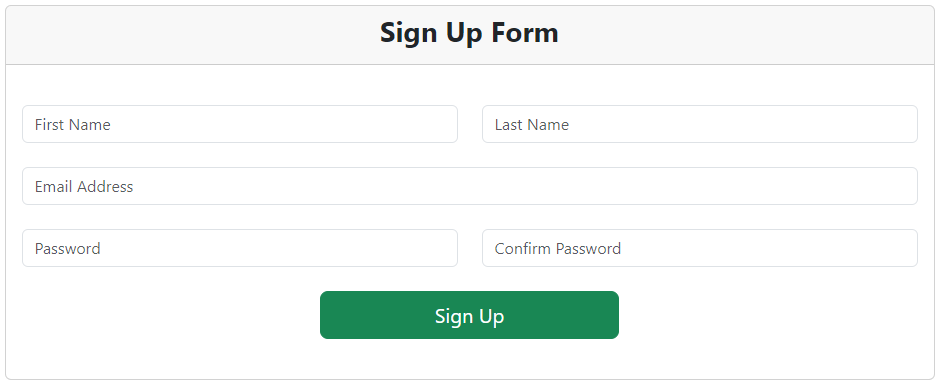
In this test, you are required to create a sign-up form in React using react-hook-form. You are required to validate each of the input before the form can be submitted.

This test has 3 requirements. In order to get full marks in this test, you need to fulfill all 3 requirements completely. Please refer to the submission details above to make sure you are submitting the correct files.

**Requirement 1:**

In this section, you are expected to create a Registration (Sign Up) Form. Please note the following:

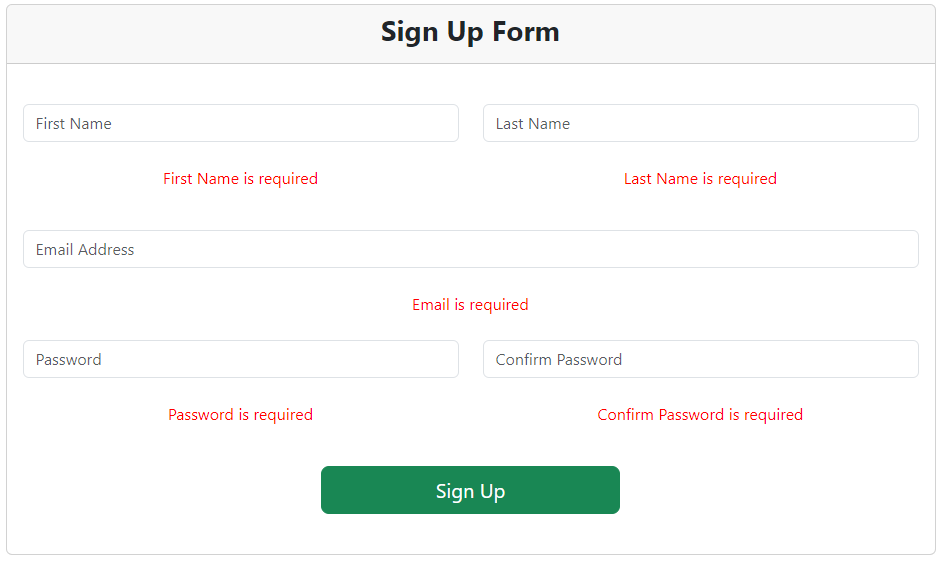
* Use Bootstrap’s components to design it
* The Sign-Up form is enclosed in a card with a title “Sign Up Form”
* There are 4 input fields and for each input field, you should have at least the following properties, however, you are most welcome to use additional properties if required:
  + type
  + tabIndex
  + placeholder
* There is one button on the form which should be a large success button titled “Sign Up”
* Make sure not to deviate too much from the required design
* Below is the expected output of this requirement:



**Requirement 2:**

In this requirement, you are expected to validate each input. Please note the following:

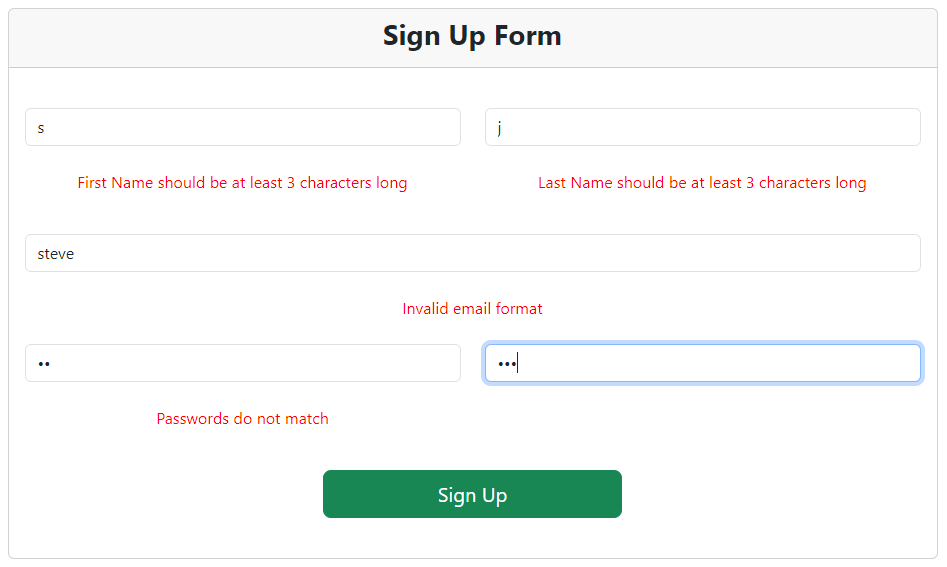
* Validations must be checked using react-hook-form
* Validations for each input must be checked on onChange event of react-hook-form. Please note that this onChange is different from regular javascript onChange. Please refer to register’s onChange from [react-hook-form](https://react-hook-form.com/docs/useform/register#registerRef) documentation.
* For this requirement check for required validation for each field only, e.g., for example, if a field is left empty, an error message must be shown underneath that field saying “This field is required”. Please refer to the screenshot below for more details.
* Do not forget to import useForm and its appropriate APIs, e.g., register, handleSubmit, forState: {errors} etc.
* Below is the expected output of this requirement:



**Requirement 3:**

In this requirement, you are expected to validate individual fields based on their inputs. Please note the followings:

* Validations must be checked using react-hook-form
* Validations for each input must be checked on onChange event of react-hook-form.
* For First and Last Name, make sure the input is more than 3 characters. Use a single validation method validateNames(value, name) that validates to check if the fields have more than 3 characters or not. If not, show an error saying “Name should be at least 3 characters long”. The second parameter name of this function can help you in identifying which field has triggered this method, e.g., First Name or Last Name. Hint: Pass this function to validate of {…register}.
* For email field, check if the input contains at least one @ sign. If not, show an error saying “Invalid email address”.
* For Password and Confirm Password, use a method validatePasswordMatch(value) that checks to see if both fields are equal or not? If not, show an error saying “Passwords do not match”. Look into the documentation of [getValues](https://react-hook-form.com/docs/useform/getvalues) from useForm() that can help you in getting values of multiple fields in the same function.
* Below is expected output of this requirement:



**Solution**

**Registration.js**

import React, { useState } from "react";

import "bootstrap/dist/css/bootstrap.min.css";

import "./style.css"

import {useForm} from "react-hook-form"

export default function RegistrationForm() {

    const {register, handleSubmit, getValues, formState: {errors}} = useForm({defaultValues : {

        firstName: "",

        lastName: "",

        email: "",

        password: "",

        confirmPassword: "",

    }});

    function validateNames(value, name) {

        if (value.length >= 3)

            return true;

        return `${name} should be at least 3 characters long`;

    }

    function validatePasswordMatch(value) {

        if(value === getValues("confirmPassword"))

            return true;

        return "Passwords do not match";

    }

    return (

    <div className="card">

        <div className="card-header">

            <h3>Sign Up Form</h3>

        </div>

        <div className="card-body">

            <form onChange={handleSubmit()}>

                <div className="row">

                    <div className="col">

                    <input type="text" tabIndex="1" className="form-control" placeholder="First name"

                        {...register("firstName", {

                            required: "First Name is required",

                            validate: (value) => {

                                return validateNames(value, "First Name");

                            }

                        })}

                    />

                    <br/>

                    {errors.firstName && (<p>{errors.firstName.message}</p>)}

                    </div>

                    <div className="col">

                    <input type="text" tabIndex="2" className="form-control" placeholder="Last name"

                        {...register("lastName", {

                            required: "Last Name is required",

                            validate: (value) => {

                                return validateNames(value, "Last Name");

                            }

                        })}

                    />

                    <br/>

                    {errors.lastName && (<p>{errors.lastName.message}</p>)}

                    </div>

                </div>

                <div className="row">

                    <div className="col">

                    <input type="text" tabIndex="3" className="form-control" placeholder="Email Address"

                        {...register("email", {

                            required: "Email is required",

                            validate: (value) => {

                                if(value.indexOf('@') > -1)

                                    return true;

                                return "Invalid email address";

                            }

                        })}

                    />

                    <br/>

                    {errors.email && (<p>{errors.email.message}</p>)}

                    </div>

                </div>

                <div className="row">

                    <div className="col">

                    <input type="text" tabIndex="4" className="form-control" placeholder="Password"

                        {...register("password", {

                            required: "Password is required",

                            validate: (value) => {

                                return validatePasswordMatch(value);

                            }

                        })}

                    />

                    <br/>

                    {errors.password && (<p>{errors.password.message}</p>)}

                    </div>

                    <div className="col">

                    <input type="text" tabIndex="5" className="form-control" placeholder="Confirm Password"

                        {...register("confirmPassword", {

                            required: "Confirm Password is required"})}

                    />

                    <br/>

                    {errors.confirmPassword && (<p>{errors.confirmPassword.message}</p>)}

                    </div>

                </div>

                <div className="text-center">

                    <button type="submit" tabIndex="6" className="btn btn-success btn-lg">Sign Up</button>

                </div>

            </form>

        </div>

    </div>

    );

}

**style.css**

h3 {

    text-align: center;

}

.card {

    width: 50vw;

    margin: 0 auto;

}

.card-body {

    padding: 20px;

}

p {

    color: red;

    text-align: center;

}

button {

    width: 20vw;

}

Final Output:

A screenshot of a sign up form

Description automatically generated