System Requirements Specification Index

For

Registration Form Application

Version 1.0

IIHT Pvt. Ltd.
IIHT Ltd, No: 15, 2nd Floor, Sri Lakshmi Complex, Off MG Road, Near SBI LHO,
Bangalore, Karnataka – 560001, India
fullstack@iiht.com

Registration Form Application
System Requirements Specification

1.BUSINESS-REQUIREMENT:

1.1 PROBLEM STATEMENT:

Registration Form Application is a simple .Net Core 3.1 Blazor Server Application.

1.2 FOLLOWING IS THE REQUIREMENT SPECIFICATION:

	Registration Form Application
1	Registration Form
Registration Module Functionalities	
	1. Add a form with name and email inputs,
	2. Add validations for above inputs.
	3. On submit show the validations are firing and submission not happened if validations failed
	4. Use validation summary to display both input validations failed message together.
	5. Show a plain submission, when validation not failed

2. ASSUMPTIONS, DEPENDENCIES, RISKS / CONSTRAINTS

2.1 Common Constraints

- For all operations, validation check must be done and must throw custom exception if data is invalid
- Do not change, add, remove any existing methods

2.2 ToolChain

- .NET Core 3.1, Blazor Application

3. BUSINESS VALIDATIONS

3.1 Form Properties

- Name Property:
  Data Type: string
  Validation Message (Required): "Name is required" (The name field must not be empty.)Name (string) is not null,min 3 and max 100 characters.

- Email Property:
  Data Type: string
  Validation Message (Required): "Email is required" (The email field must not be empty.)
  Validation Message (EmailAddress): "Invalid email address" (The entered email address must be in a valid email format.)Salary (decimal) is not null.

- Age Property:
  Data Type: int
  Validation Message (Required): "Age is required" (The age field must not be empty.)
  Validation Message (Range): "Invalid age" (The age value must be an integer between 1 and 120.)

- MobileNumber Property:
  Data Type: string
  Validation Message (Required): "Mobile number is required" (The mobile number field must not be empty.)
  Validation Message (RegularExpression): "Invalid mobile number" (The entered mobile number must be a 10-digit number.)

- Address Property:
  Data Type: string
  Validation Message (Required): "Address is required" (The address field must not be empty.)

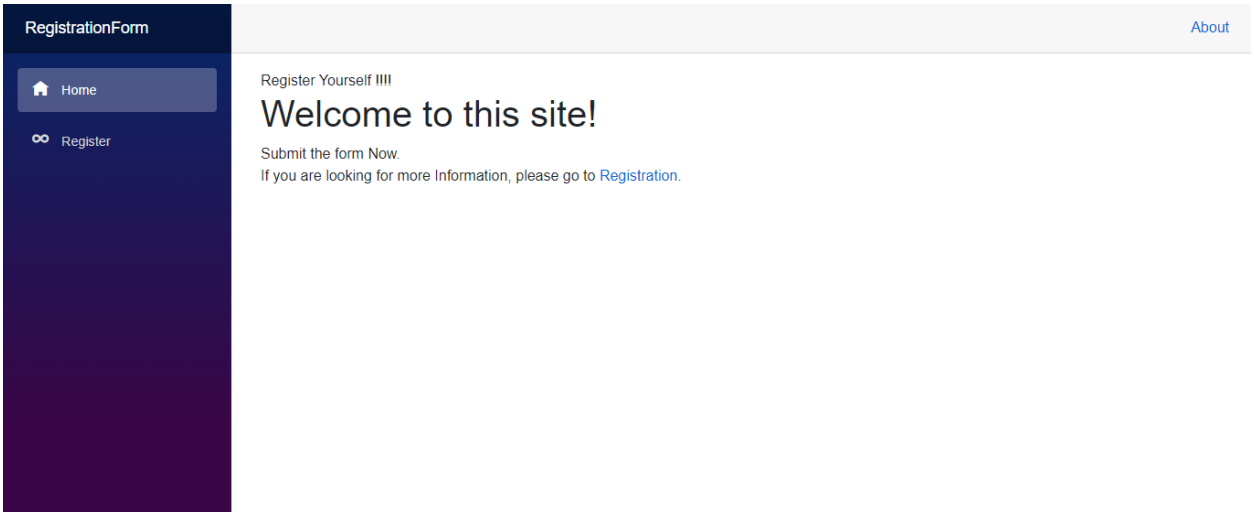
3.2 Functional Requirement

Method Name: HandleValidSubmit

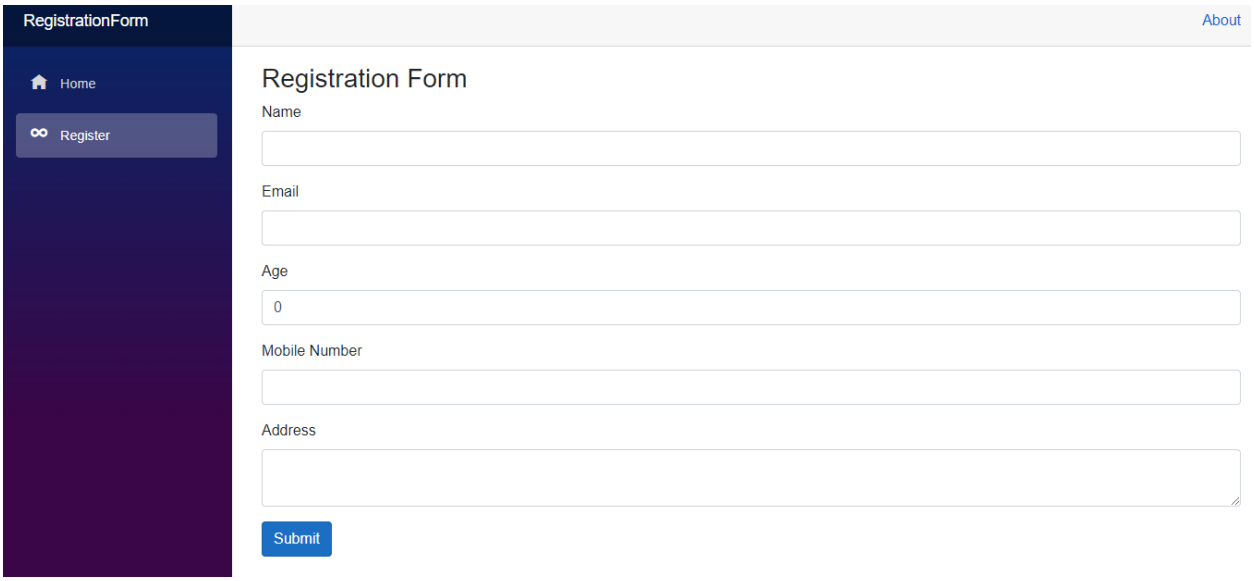
- Summary: Handles the form submission logic when the form is valid.
- Description:
  This method is called when the form is submitted and all validation rules pass.
  It should contain the logic to process the form submission when the user has provided valid data. Redirects to a new page after form submission.

4. WIREFRAME

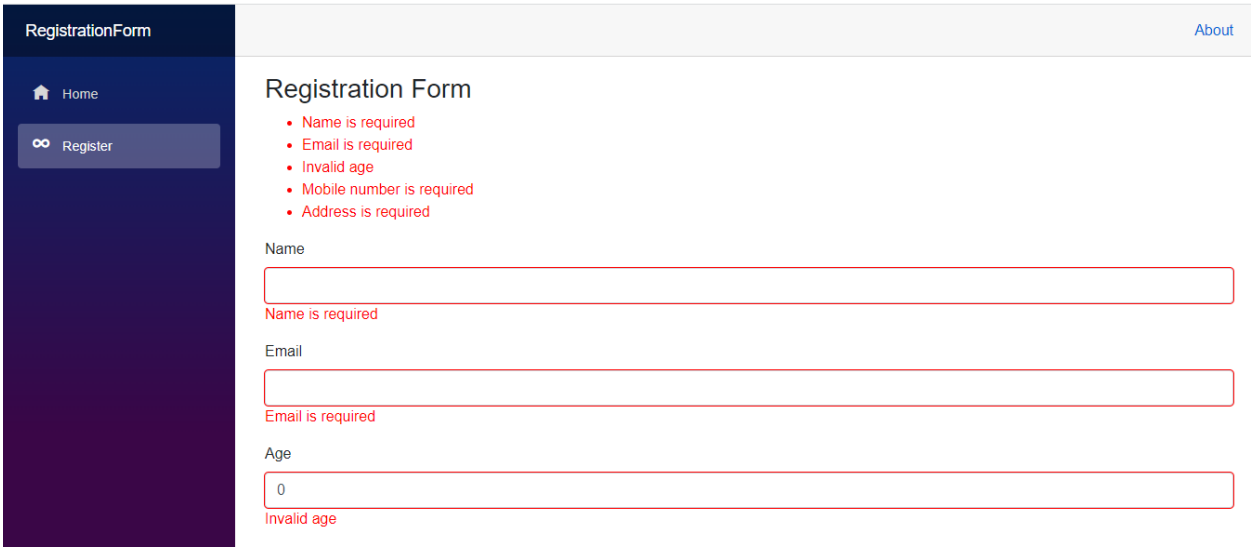
1) Landing Page



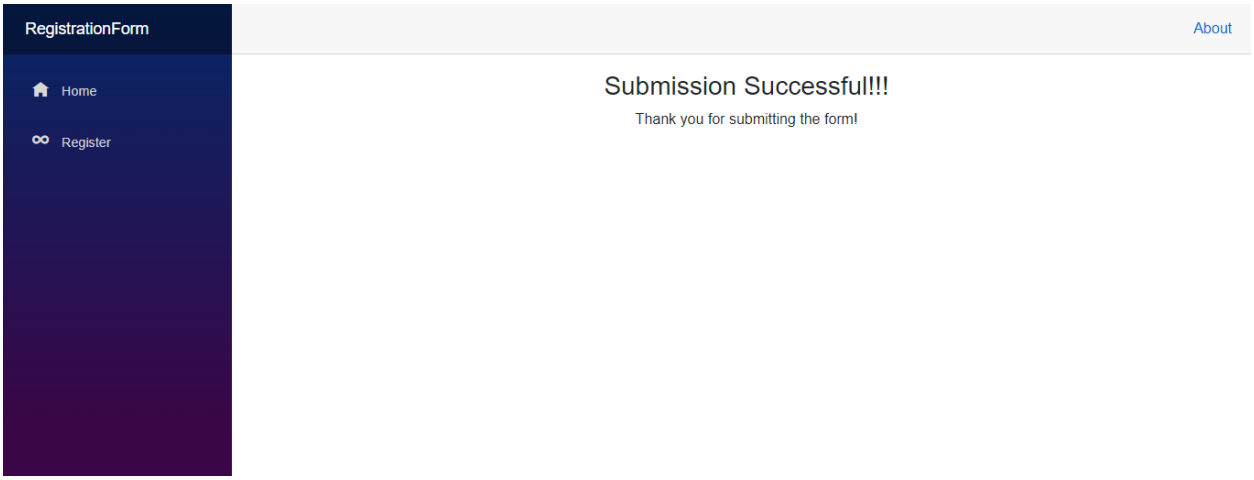
2) On click of Register button it should display registration form.



3) On invalid submission it should show validation summary



4) On valid submission it should redirect to submission.razor page



5. Template Code Structure

5.1 Package: RegistrationForm

Resources

Names	Resource	Remarks	Status
Package Structure			
FormModels	RegisterDataFormModel	All Entities/Domain attribute are used for pass the data .	Partially implemented

Program.cs	Program CS file	Contain all Services settings	Already Implemented
Properties	launchSettings.json file	All URL Setting for API	Already Implemented
Pages	Registration.cs	Business logic for Registration	Partially Implemented
Pages	Submission.cs	Display Submission Information	Partially Implemented

5.2 Package: RegistrationForm.Tests

Resources

The RegistrationForm.Tests project contains all test case classes and functions for code evaluation. Don't edit or change anything inside this project.

6. EXECUTION STEPS TO FOLLOW

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 the Application menu (Three horizontal lines at left top) Terminal → New Terminal.**

3. **On command prompt, cd into your project folder (cd <Your-Project-folder>).**

4. **To build your project use command:**
   (RegistrationForm /dotnet build)

5. **To launch your application, Run the following command to run the application:**
   (RegistrationForm /dotnet run)

6. **This editor Auto Saves the code.**

7. **To test any Restful application, the last option on the left panel of IDE, you can find ThunderClient, which is the lightweight equivalent of POSTMAN.**

8. **To test web-based applications 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

9. **To run the test cases in CMD, Run the following command to test the application: (RegistrationForm /dotnet test --logger "console;verbosity=detailed") (You can run this command multiple times to identify the test case status,and refactor code to make maximum test cases passed before final submission)**

10. **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.**

11. **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.**

12. **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.**
