Registration Form Project Instructions

1 Project Abstract

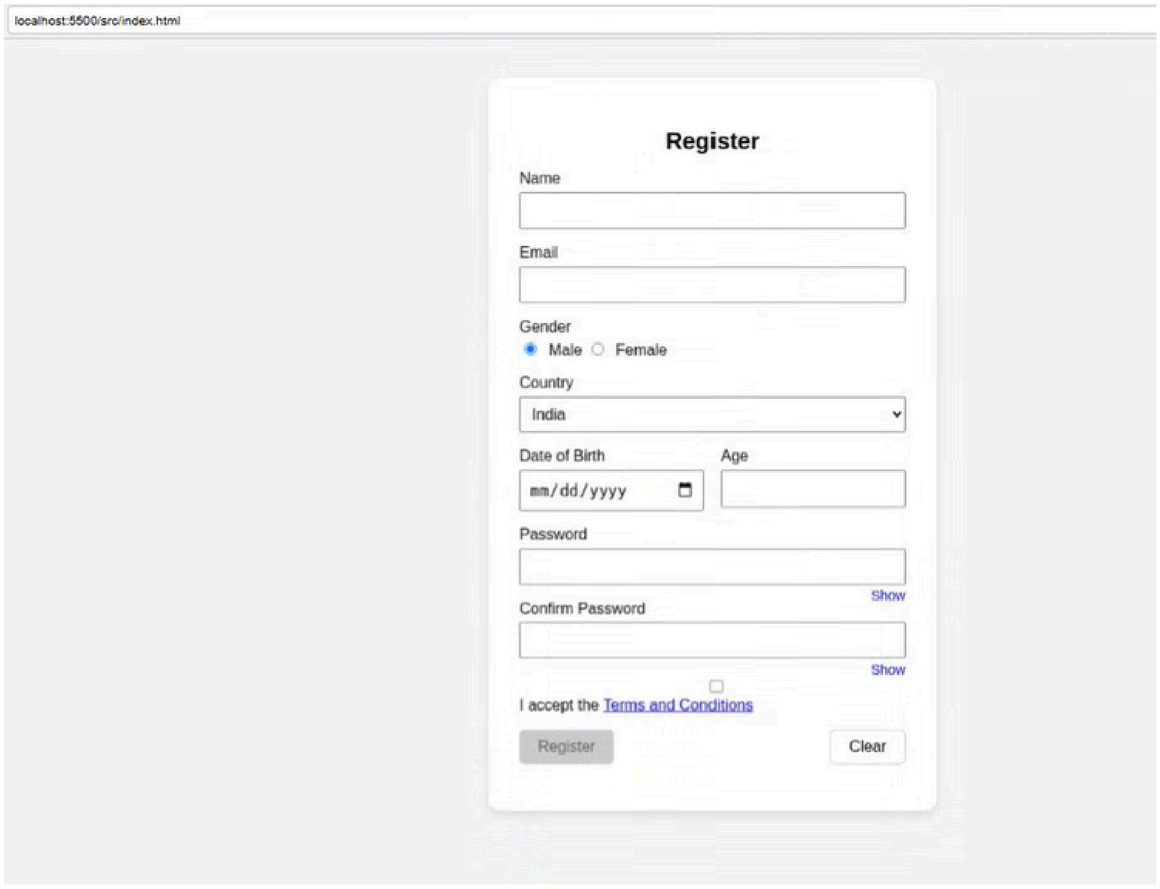
The **User Registration Form** is a front-end development project that focuses on form validation, UI design, and user-friendly feedback using **HTML**, **CSS**, and **JavaScript**. This project enables users to fill out a registration form with fields like name, email, gender, country, date of birth, age (auto-calculated), password, and confirmation password, with real-time validation and error feedback.

This project emphasizes:

- Validating dynamic user inputs such as email, password strength, and matching passwords using JavaScript.
- Automatically calculating and displaying the user's age from the selected date of birth.
- Creating a clean form layout using **HTML** and **CSS** techniques.
- Enhancing input error highlighting, toggleable password visibility, and real-time feedback.
- Managing form state effectively, including enabling/disabling the register button and handling form reset.

ScreenShots:

*****Register Page*****


localhost:5500/src/index.html

Register

Name

Email

Gender

Male [✓] Female

Country

India

Date of Birth mm/dd/yyyy

Age

Password

Show

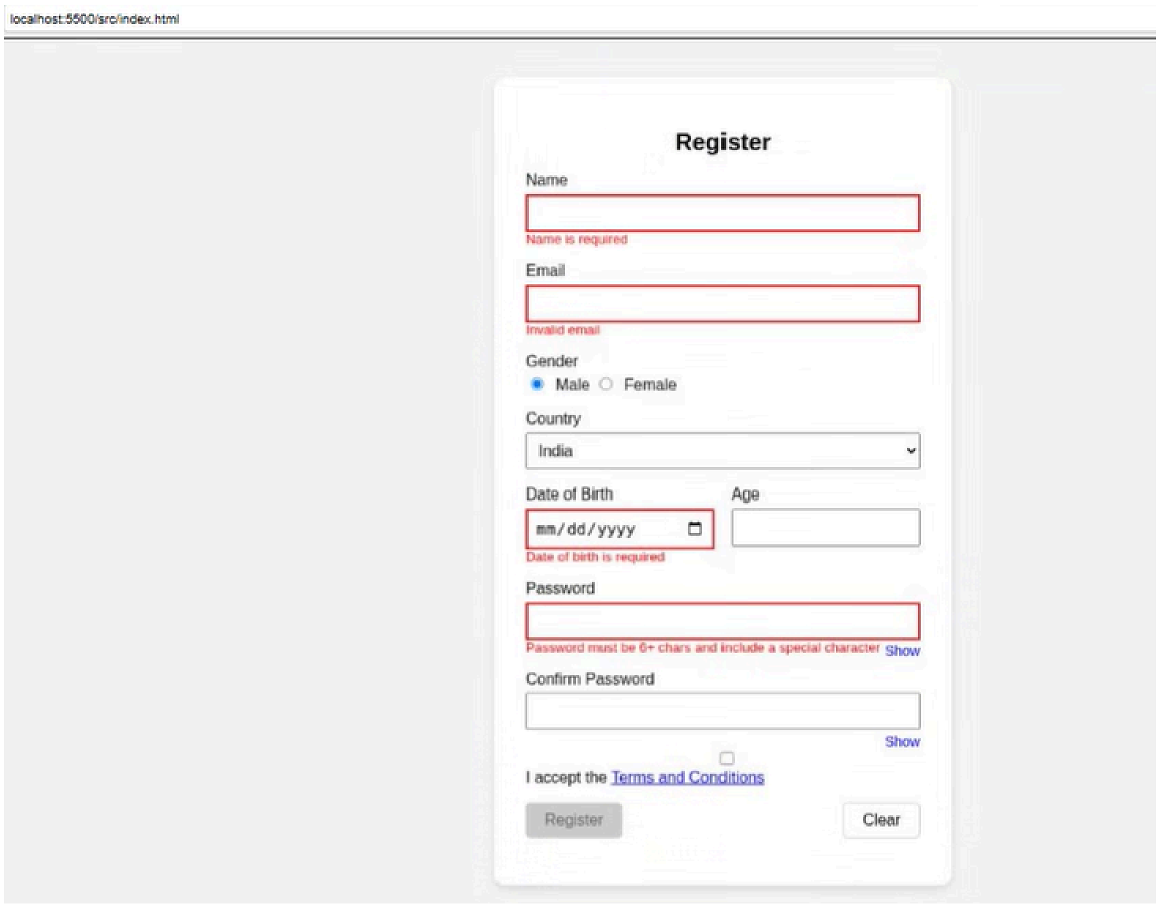
Confirm Password

Show

I accept the Terms and Conditions

Register Clear

*****Validations*****



localhost:5500/src/index.html

Register

Name

Name is required

Email

Invalid email

Gender

[✓ Male] Female

Country

India

Date of Birth

mm/dd/yyyy

Date of birth is required

Age

Password

Password must be 6+ chars and include a special character

Show

Confirm Password

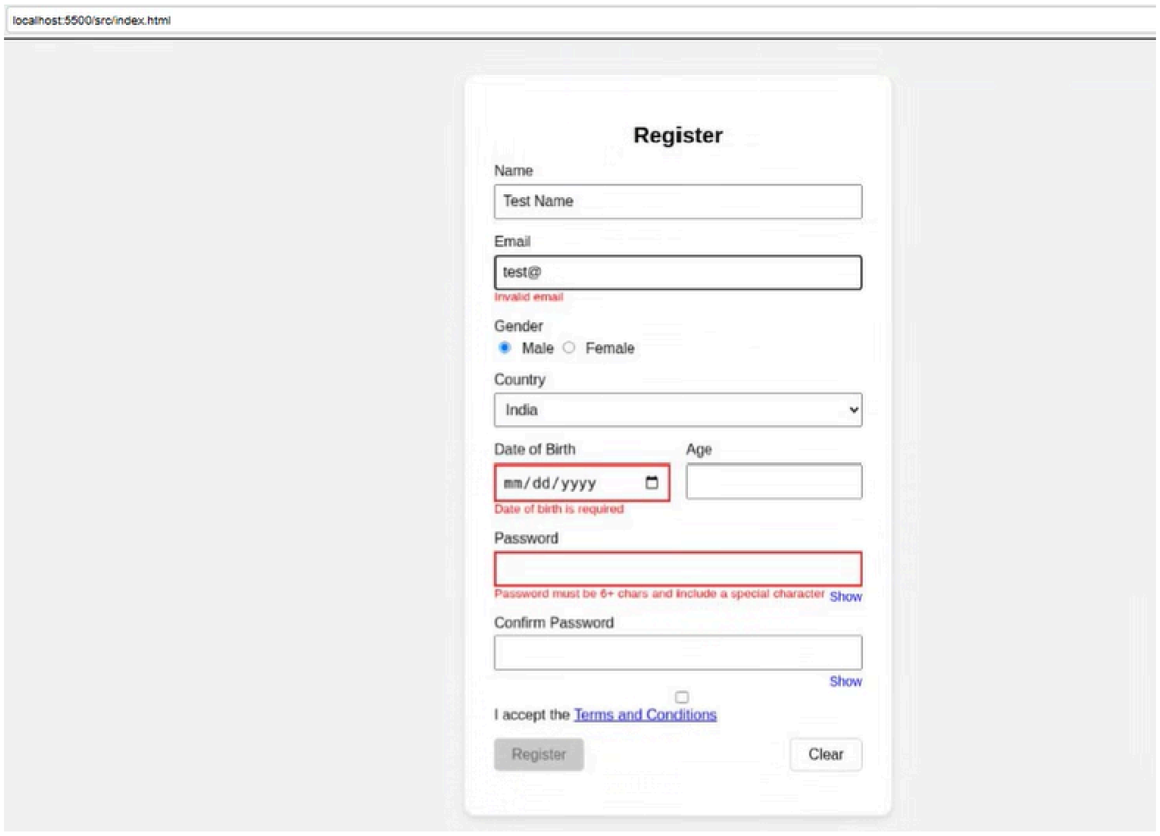
Show

I accept the Terms and Conditions

Register

Clear

The form performs real-time validations as each field is filled, providing instant feedback. In this case, the email input is invalid, so the form displays an "Invalid email" message, highlights the field, and disables the register button until all validations are satisfied.


localhost:5500/src/index.html

Register

Name: Test Name

Email: test@

Invalid email

Gender: Male [✓] Female

Country: India

Date of Birth: mm/dd/yyyy

Date of birth is required

Age:

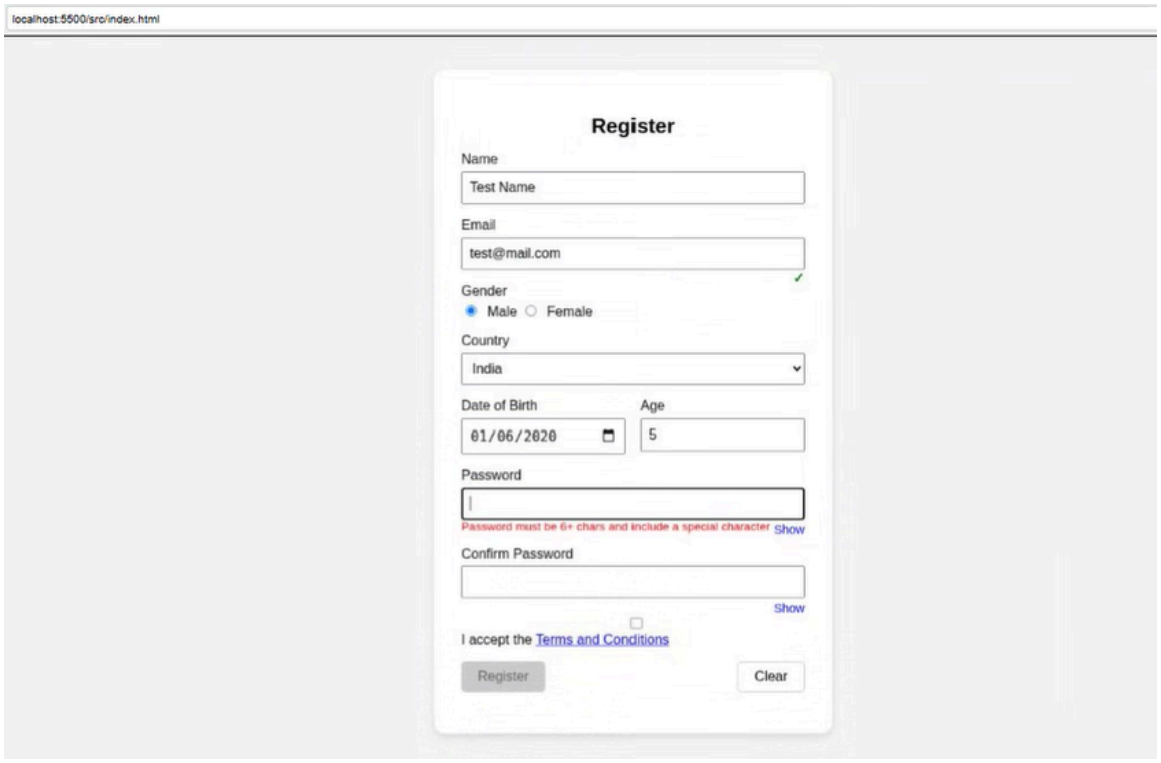
Password:

Password must be 6+ chars and include a special character

Confirm Password:

I accept the Terms and Conditions

*****Password field must be at least 6 characters long and include a special character.*****



localhost:5500/src/index.html

Register

Name: Test Name

Email: test@mail.com ✓

Gender: (● Male) ○ Female

Country: India

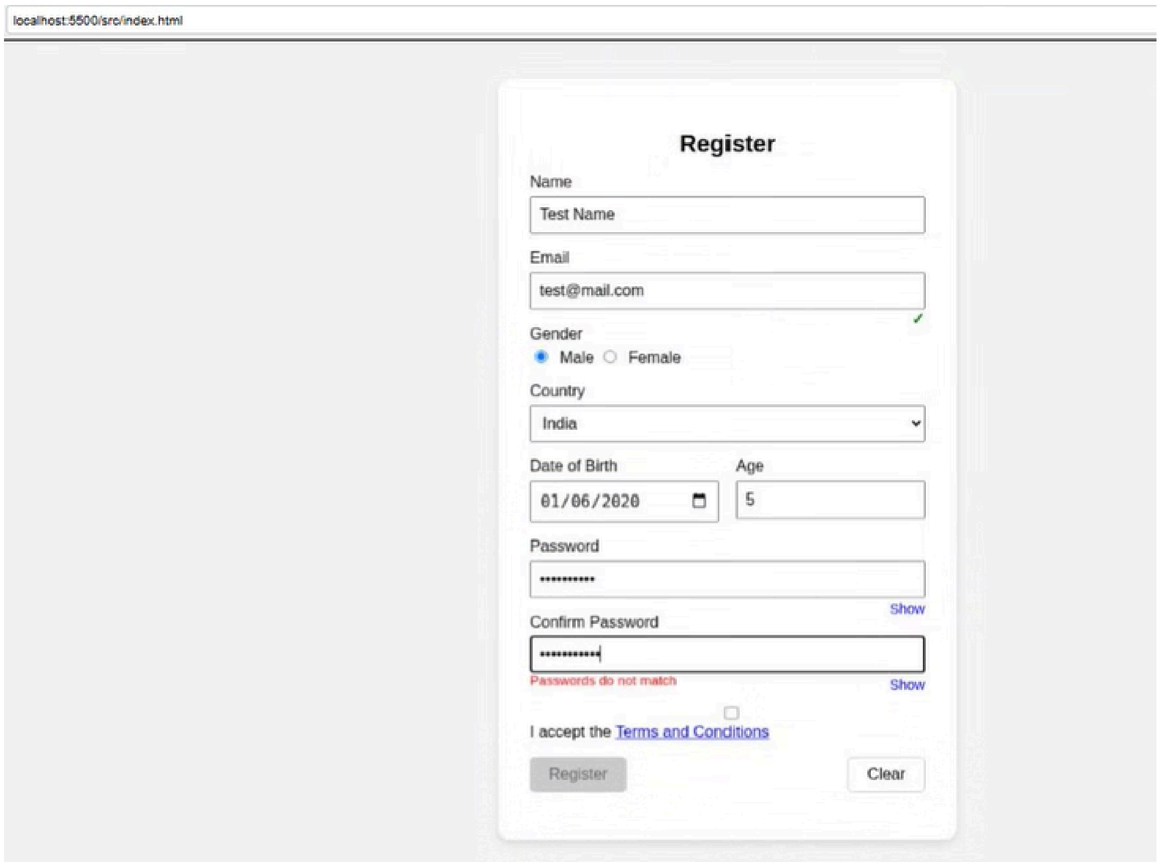
Date of Birth: 01/06/2020

Age: 5

Password:

Password must be 6+ chars and include a special character

*****Password not match*****


localhost:5500/src/index.html

Register

Name

Test Name

Email

test@mail.com ✓

Gender

● Male ○ Female

Country

India

Date of Birth: 01/06/2020

Age: 5

Password

•••••••••••

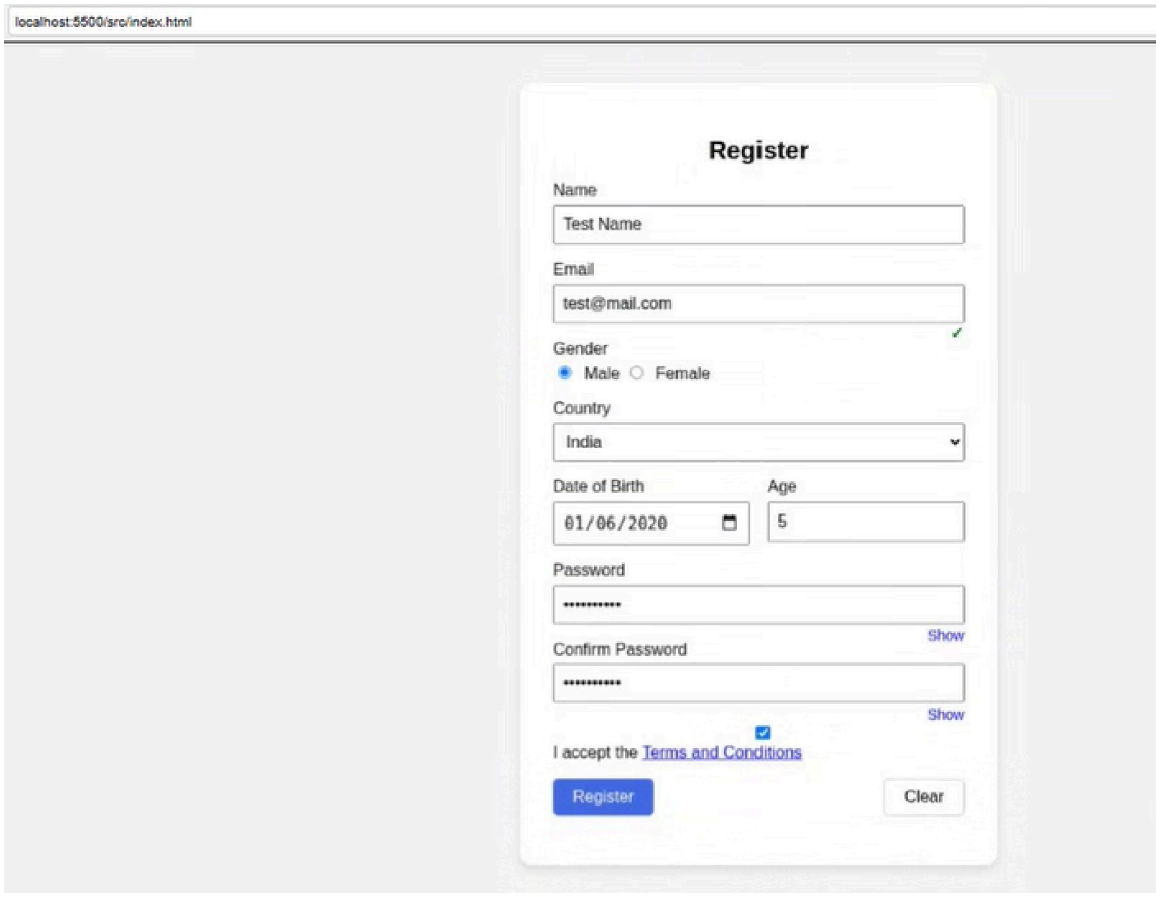
Confirm Password

••••••••••••

Passwords do not match

☐ I accept the Terms and Conditions

*****All required fields are correctly filled, Additionally, the terms and conditions checkbox is checked, which is essential for enabling the Register button*****



localhost:5500/src/index.html

Register

Name: Test Name

Email: test@mail.com ✓

Gender: [✓ Male] Female

Country: India

Date of Birth: 01/06/2020

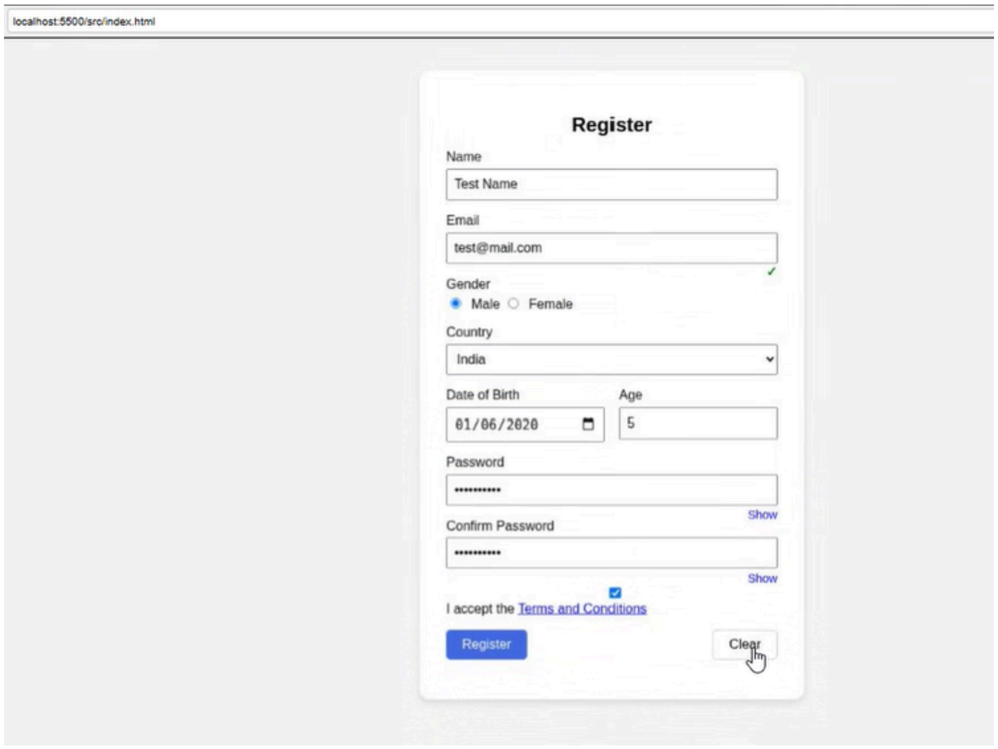
Age: 5

Password: •••••••••••

Confirm Password: •••••••••••

☑ I accept the Terms and Conditions

*****Clicking the Clear button resets the entire form — all input fields are emptied, validations and highlights are removed, and the Register button is disabled again. This ensures a clean state for the user to start a fresh registration.*****



localhost:5500/src/index.html

Register

Name: Test Name

Email: test@mail.com ✓

Gender: ● Male ○ Female

Country: India

Date of Birth: 01/06/2020

Age: 5

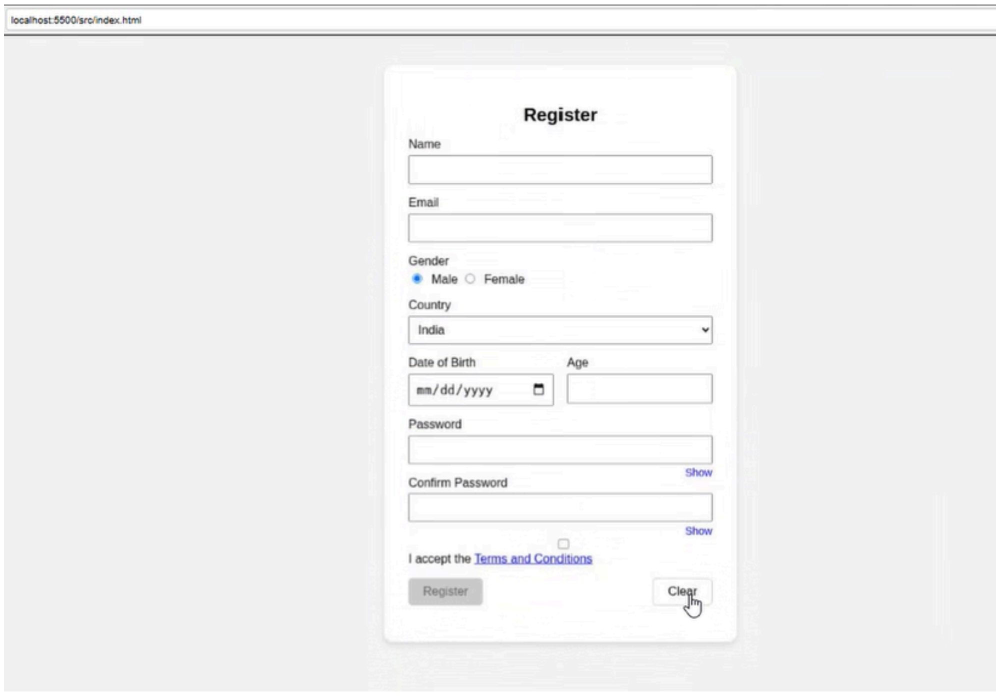
Password: ••••••••••

Show

Confirm Password: ••••••••••

Show

☑ I accept the Terms and Conditions



localhost:5500/src/index.html

Register

Name:

Email:

Gender: ● Male ○ Female

Country: India

Date of Birth: mm/dd/yyyy

Age:

Password:

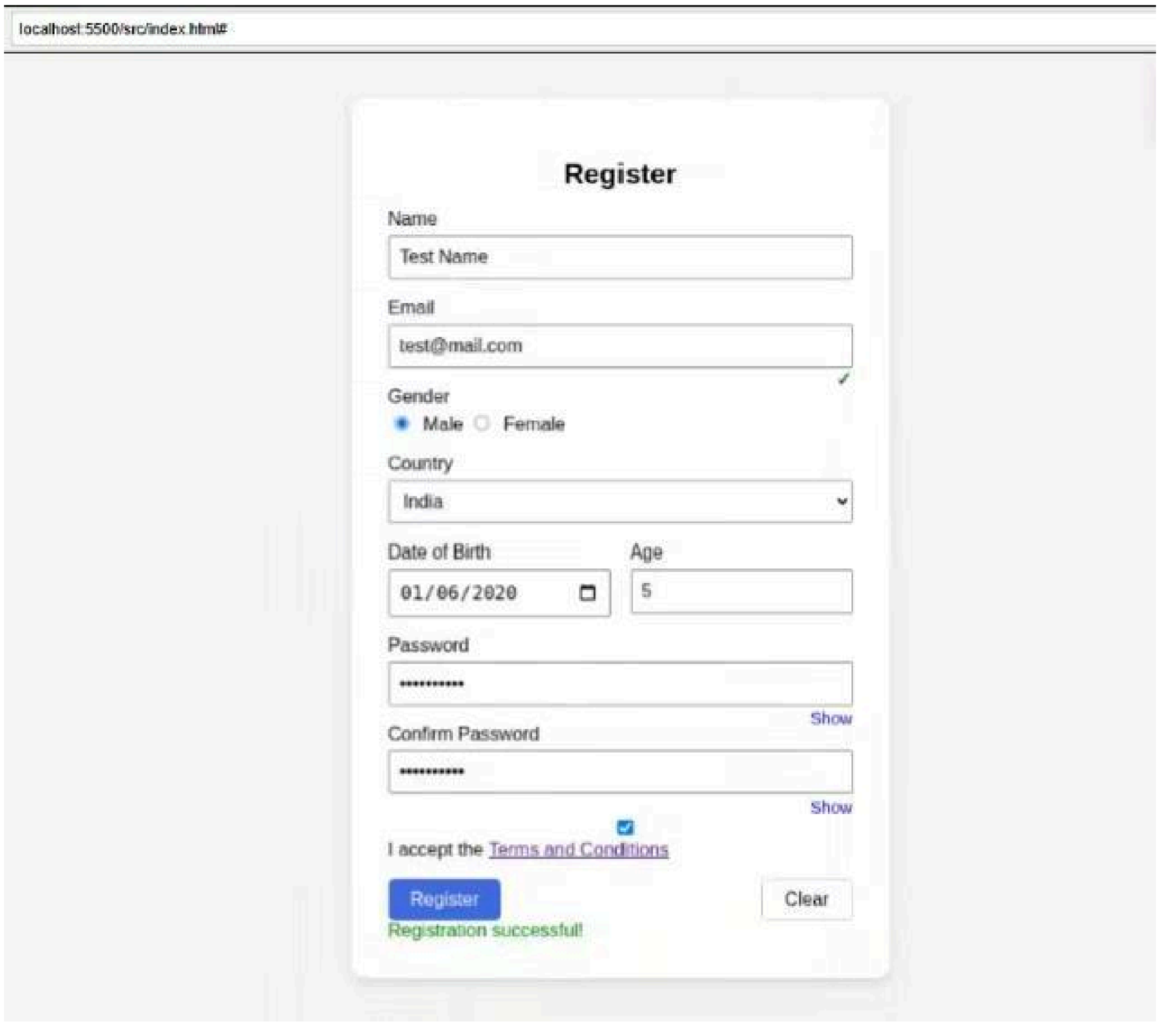
Show

Confirm Password:

Show

☐ I accept the Terms and Conditions

Register Successful


localhost:5500/src/index.html#

Register

Name

Test Name

Email

test@mail.com

✓

Gender

[✓ Male] Female

Country

India

Date of Birth

01/06/2020

Age

5

Password

•••••••••••

Show

Confirm Password

•••••••••••

Show

[✓] I accept the Terms and Conditions

Register

Clear

Registration successful!

Instructions for Core Files

1. index.html

Create the HTML structure for a **User Registration Form** using the following specifications:

- In the `head`, include:
    - Title as **“Register”**
    - Link to external `style.css` for styling
- In the `body`, add a `form` with ID `registerForm`.
- Inside the form:
    - Add a heading `h2` with the text **"Register"**
    - Add input fields for:
        - **Name**: Field with ID `name` for entering the user's name; it should be required.
        - **Email**: Field with ID `email` for entering the user's email; it should be required and show live validation feedback.
        - **Gender**: Radio button group for selecting gender; options include Male and Female.
        - **Country**: Dropdown field with ID `country` to select the user's country; options include India, USA, and UK.
        - **Date of Birth**: Field with ID `dob` for selecting the user's birthdate; it should be required.
        - **Age**: Read-only field with ID `age` that auto-calculates based on the date of birth.
        - **Password**: Field with ID `password` for entering a secure password; it should be required and meet strength criteria.
        - **Confirm Password**: Field with ID `confirmPassword` to re-enter and confirm the password; it should be required.
    - Include a checkbox for **Terms and Conditions** with ID `terms` that must be checked to allow registration.
    - Add two buttons:
        - **Register Button**: Submit button with ID `submitBtn`, initially disabled and only enabled when all fields are valid.
        - **Clear Button**: Button that resets all fields and clears validations.
    - Include an area to display form messages (success or error)
- Link the external `script.js` file at the bottom of the body to handle validation logic.

2. style.css

Define the styles for the registration form with the following specifications:

- Set a clean, readable font and center the form layout.
- Style the `form` with padding, rounded corners, and subtle shadows for a card-like appearance.
- Apply spacing between form groups and consistent input styling.
- Style:
    - `.error` for red error messages
    - `.success` for green success messages
    - `.highlight-error` to outline invalid inputs in red
    - `.checkmark` to show a green tick for valid email
    - `.toggle` for showing/hiding password
- Customize buttons:
    - `Register` button with a **blue background**
    - `Clear` button with a bordered white style
    - Disabled state is greyed out and unclickable

---

3. script.js

Write the JavaScript to handle form interactivity and validations:

- Define validation rules:
    - Email must match regex format
    - Password must be 6+ characters and include a special character
    - Confirm Password must match Password
    - Terms checkbox must be checked
- Functions:
    - `validateField(field)` – validates individual inputs and shows errors
    - `calculateAge()` – computes age based on DOB and sets it in the Age field
    - `toggleVisibility(id)` – toggles password visibility between `text` and `password`
    - `isFormCompletelyValid()` – checks if all required fields are valid
    - `validateForm()` – runs all validations and displays a success or error message
    - `clearForm()` – resets the form, clears all errors and disables the submit button

---

Detailed HTML Structure Guide for Registration Form App

1. Document Declaration and Head Section

Inside the `head` section:

- Set the page title to "Register".
- Link an external stylesheet named `style.css` to handle all layout and styling.

2. Body and Main Form

- Inside the `body`, create a form element and assign it the ID `registerForm`.
  All form fields and buttons should be placed inside this form. This ensures proper validation and submission handling.

3. Heading

At the top of the form, place a heading (level 2) with the text "Register". This serves as the form title.

4. Name Input Field

Create a form-group section for the user's name:

- Add a label with the text "Name".
- Add a text input with the ID `name`. This field is required.
- Below the input, add a `div` with the class `error` and ID `nameError` to display validation messages dynamically.

5. Email Input Field

In the next form-group:

- Add a label with the text "Email".
- Add an input field of type email and assign it the ID `email`. This field is also required.
- Next to it, place a `span` with the class `checkmark` and the ID `emailCheck` to visually show validation status.

- Add another `div` below with the class `error` and ID `emailError` to show error messages.

6. Gender Selection

Add a form-group for gender selection:

- Label it as “Gender”.
- Provide two radio inputs with the same `name` attribute (e.g., "gender"):
    - One with the value "Male" (checked by default).
    - One with the value "Female".

7. Country Dropdown

Add a dropdown menu for country selection inside a form-group:

- Use the label “Country”.
- Create a select element with ID `country`.
- Add three options: “India” (selected by default), “USA”, and “UK”.

8. Date of Birth and Age Row

Use a form-row container to place two form-groups side by side:

- The first half contains:
  Label: “Date of Birth”
    - Input of type `date` with ID `dob`
    - Error container (`div`) with ID `dobError`

- The second half contains:
    - Label: “Age”
    - Input of type `text` with ID `age`, set to read-only

9. Password Field

Create a password section with validation support:

- Label it as “Password”.
- Input of type `password` with ID `password`, and mark it as required.
- Add a clickable `span` with class `toggle` to show/hide the password.

- Include an error `div` with the ID `passwordError` for feedback.

10. Confirm Password Field

Repeat the same structure as the password field:

- Use the label “Confirm Password”.
- Input field with ID `confirmPassword`, also required.
- Toggle button to show/hide password using a `span` with `toggle` class.
- Error `div` with ID `confirmPasswordError`.

11. Terms and Conditions

Include a checkbox for terms acceptance inside a form-group:

- Add a checkbox with ID `terms`.
- Label it with the text “I accept the Terms and Conditions”, including a clickable link.
- Below it, place an error `div` with the ID `termsError`.

12. Action Buttons

Create a container with the class `button-row` for form actions:

- Add a button with type `submit` and ID `submitBtn`. This should be initially disabled and labeled “Register”.
- Add a second button with a type `button` that calls `clearForm()` when clicked. Label it as “Clear”.

13. Output Message

Below the buttons, add a `div` with class `output` and ID `outputMessage`.
This area is used to display confirmation messages or feedback after form submission.

14. JavaScript Inclusion

At the very end of the `body`, add a script tag linking to the external file `script.js` to enable form validation and interactivity.

Detailed CSS Styling Guide for Registration Form App

This guide explains how to style the `style.css` file for the Registration Form project. Each section outlines the purpose of the styles and how they improve usability and design clarity.

1. Body Styling

Applies the base page styling and centers the form:

- Font: `Arial, sans-serif` for clean, readable text.
- Background color: `#f4f4f4` (light gray for contrast).
- Display: `flex` for flexible alignment.
- Horizontal alignment: `justify-content: center`.
- Padding: `2rem` around the page content for spacing.

2. Form Container (`form`)

Styles the card-like box containing the form:

- Background color: `white` for clean visual separation.
- Padding: `2rem` inside the form for internal spacing.
- Border-radius: `12px` for softly rounded corners.
- Width: `400px` (fixed for readability).
- Box-shadow: `0 4px 8px rgba(0, 0, 0, 0.1)` for subtle elevation.

3. Form Title (`h2`)

- Text alignment: `center`.
- Bottom margin: `1rem` to separate the heading from form fields.

4. Form Group Wrapper (`.form-group`)

- Bottom margin: `1rem` between each form field section.

5. Form Row Layout (`.form-row`)

- Display: `flex` to arrange child elements in a row.

- Horizontal spacing: `justify-content: space-between`.

6. Half-Width Fields (`.form-group.half`)

- Width: `48%` to allow two fields side-by-side with spacing.

7. Form Labels (`label`)

- Display: `block` to appear above the field.
- Bottom margin: `0.3rem` for spacing between label and input.

8. Inputs and Select Dropdowns (`input`, `select`)

- Width: `100%` to span full container width.
- Padding: `0.5rem` for comfortable input area.
- Font size: `1rem` for readable text.
- Box sizing: `border-box` ensures consistent sizing across elements.

9. Radio Buttons (`input[type="radio"]`)

- Width: `auto` so buttons don't stretch unnecessarily.
- Right margin: `0.5rem` to space between radio options.

10. Error Messages (`.error`)

- Text color: `red` to indicate issues.
- Font size: `0.85rem` (slightly smaller than base text).

11. Success Feedback (`.success`)

- Text color: `green` (used optionally for indicators like checkmarks).

12. Output Message (`.output`)

- Text alignment: `center` to place message in middle.
- Top margin: `1rem` to separate from the form.

- Font weight: `bold` for emphasis.

13. Email Checkmark (`.checkmark`)

- Float: `right` to align with the end of the input.
- Text color: `green`.
- Font weight: `bold`.

14. Input Error Highlighting (`.highlight-error`)

- Border: `2px solid red` used dynamically for invalid fields.

15. Password Toggle Text (`.toggle`)

Used to toggle visibility for password fields:

- Float: `right` to position to the end of the input.
- Font size: `0.9rem` for smaller than base size.
- Text color: `blue` to suggest interactivity.
- Cursor: `pointer` to show it's clickable.
- Top margin: `0.2rem` to align better with input.

16. Action Button Group (`.button-row`)

- Display: `flex` for side-by-side button layout.
- Justify content: `space-between` to push buttons apart.
- Top margin: `1rem` to separate from previous fields.

17. General Button Styling (`button`)

- Padding: `0.5rem 1.2rem` for adequate click area.
- Font size: `1rem` for consistency with inputs.
- Cursor: `pointer` to indicate clickability.
- Border: `none` by default.
- Border radius: `5px` for rounded edges.

18. Submit Button (`button[type="submit"]`)

- Background color: `royalblue`.
- Text color: `white`.

19. Clear Button (`button[type="button"]`)

- Background color: `white`.
- Border: `1px solid #ccc` for neutral appearance.

20. Disabled Button State (`button:disabled`)

- Background color: `#ccc` (light gray).
- Text color: `#666`.
- Cursor: `not-allowed` to indicate inactivity.
- Border: `none` (removes border when disabled).
- Uses `!important` on background to override any other styling.

Detailed JavaScript Logic Guide for Registration Form App

This guide breaks down the JavaScript code used in `script.js`, explaining the **purpose, flow, and behavior** of each section and function within the `initializeRegistrationForm()` wrapper.

1. Regex Patterns

- **Email Regex**: Basic pattern to check `@` and `.` placement.
- **Password Regex**: Ensures at least 6 characters and one special character.

2. Validation Utilities

- **showFieldError(field, message)**
  Displays a specific error message and highlights the input with a red border.
- **clearFieldError(field)**
  Removes error text and resets field styling.

3. validateField(field)

Validates fields based on their ID and displays **specific error messages**:

Field ID	**Rule**	**Error Message**
name	Must not be empty	"Name is required"
email	Must match the email regex format	"Invalid email"
dob	Must not be empty	"Date of birth is required"
password	Must match `strongPassword` regex.	"Password must be 6+ chars and include a special character"
confirmPassword	Must match the password exactly	"Passwords do not match"

Each condition applies the appropriate error or clears it based on the result.

4. calculateAge()

- Calculates user age from selected date of birth.
- Subtracts birth year from current year.
- Adjusts if the current month/date hasn't reached the birthday yet.
- Updates the readonly `age` field.

5. toggleVisibility(id)

- Toggles visibility for password inputs.
- Changes the input type between `"password"` and `"text"` on click.

9. isFormCompletelyValid()

- Checks all fields for valid input:
    - Name, email, DOB, password, and confirm password.
    - Ensures terms checkbox is ticked.

- Returns `true` only if all validations pass.

10. updateSubmitState()

- Enables or disables the **Register** button (`submitBtn`) dynamically.
- Button remains disabled unless the form is fully valid.

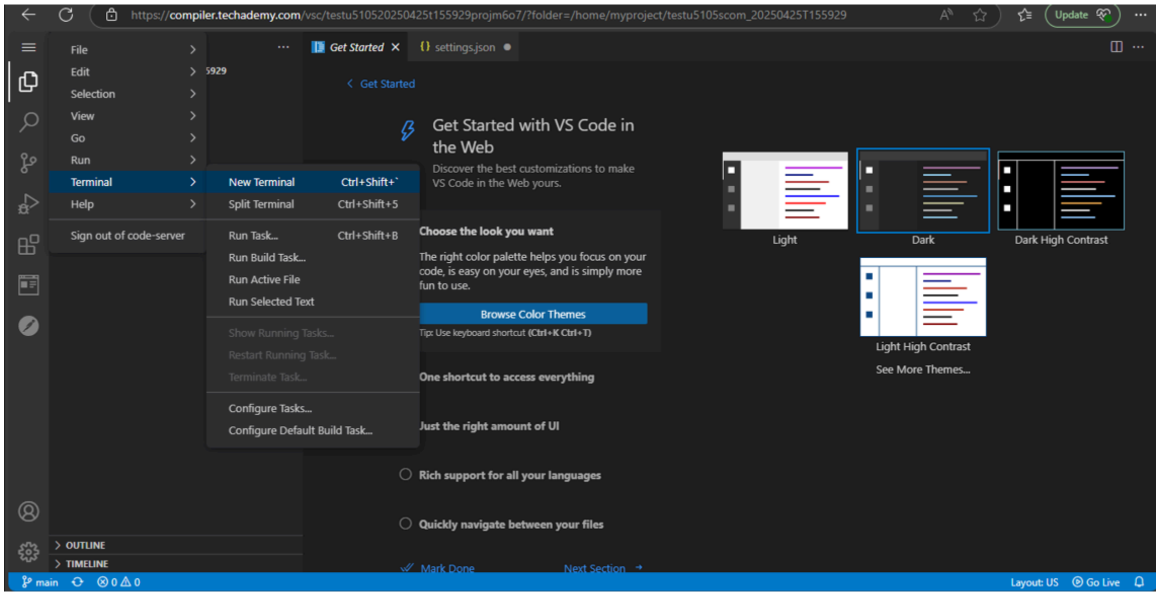
Assessment Guidelines

Step 1:

- Once the VS Code interface loads in the browser, wait until you see the workspace and left sidebar.
- To open the command terminal the test takers, need to go to

  Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.

Now in the terminal you need to install all dependencies using the **"npm install"** command.



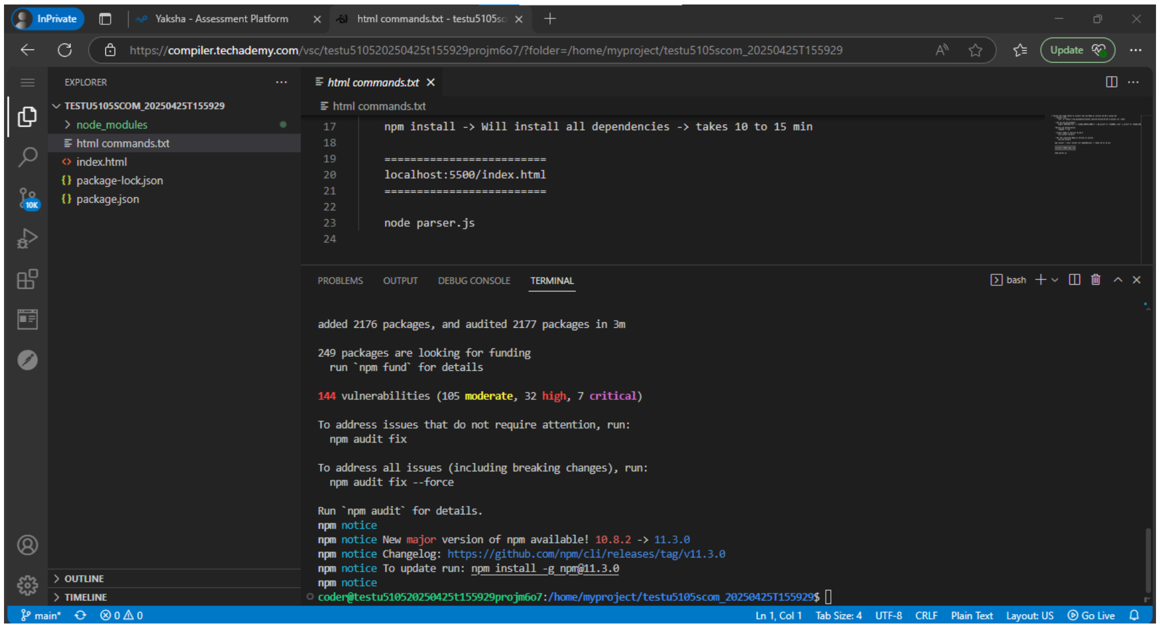
Discover the best customizations to make VS Code in the Web yours.

The right color palette helps you focus on your code, is easy on your eyes, and is simply more fun to use.

Tip: Use keyboard shortcut (Ctrl+K Ctrl+T)

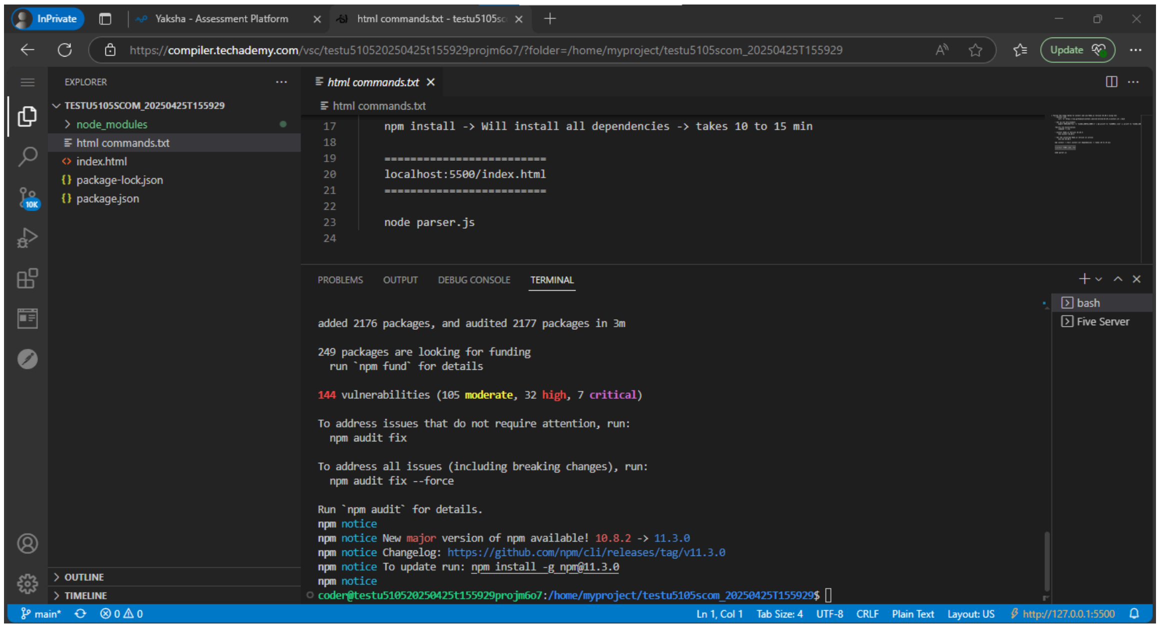
Step 2:

- Once installation completes, go to the **bottom right corner** of the VS Code screen.
- Click the **"Go Live"** button – This will start a **live server**, The server will run at port `5500` (e.g., `http://localhost:5500/`)


InPrivate
Yaksha - Assessment Platform
html commands.txt - testu5105sc
https://compiler.techademy.com/vsc/testu510520250425t155929projm6o7/?folder=/home/myproject/testu5105scom_20250425T155929
Update
EXPLORER
TESTU5105SCOM_20250425T155929
node_modules
html commands.txt
index.html
package-lock.json
package.json
OUTLINE
TIMELINE
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
bash
main*
Ln 1, Col 1
Tab Size: 4
UTF-8
CRLF
Plain Text
Layout: US
Go Live

```
npm install -> Will install all dependencies -> takes 10 to 15 min

========================
localhost:5500/index.html
========================

node parser.js
```

```
added 2176 packages, and audited 2177 packages in 3m

249 packages are looking for funding
  run `npm fund` for details

144 vulnerabilities (105 moderate, 32 high, 7 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
npm notice
npm notice New major version of npm available! 10.8.2 -> 11.3.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.3.0
npm notice To update run: npm install -g npm@11.3.0
npm notice
coder@testu510520250425t155929projm6o7:/home/myproject/testu5105scom_20250425T155929$
```

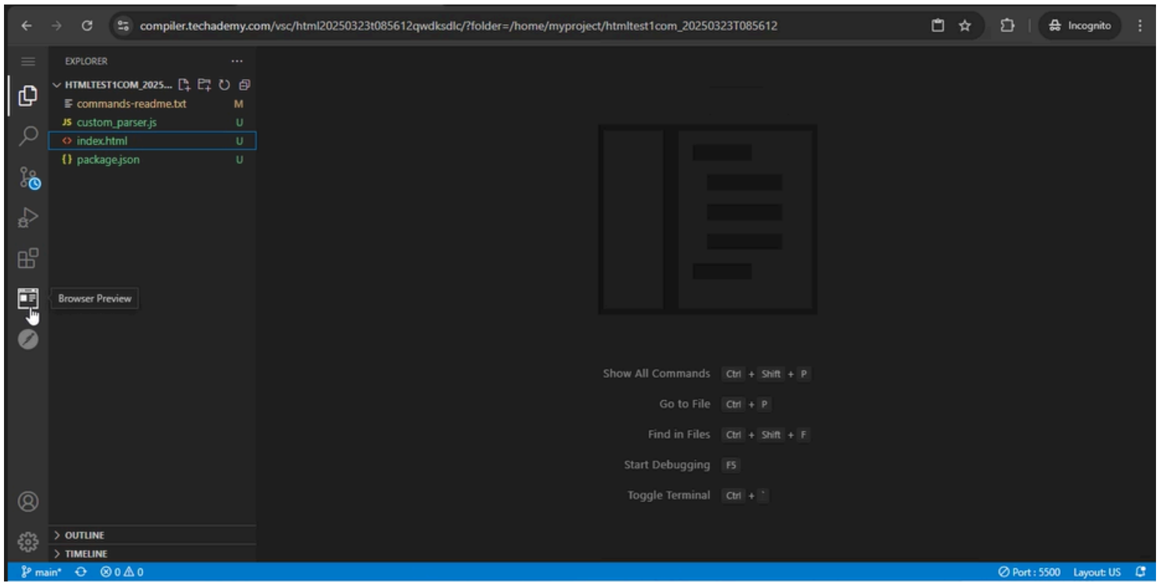


```
npm install -> Will install all dependencies -> takes 10 to 15 min

========================
localhost:5500/index.html
========================

node parser.js
```

```
added 2176 packages, and audited 2177 packages in 3m

249 packages are looking for funding
  run `npm fund` for details

144 vulnerabilities (105 moderate, 32 high, 7 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
npm notice
npm notice New major version of npm available! 10.8.2 -> 11.3.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.3.0
npm notice To update run: npm install -g npm@11.3.0
npm notice
coder@testu510520250425t155929projm6o7:/home/myproject/testu5105scom_20250425T155929$
```

Step 3: Preview Output in Browser

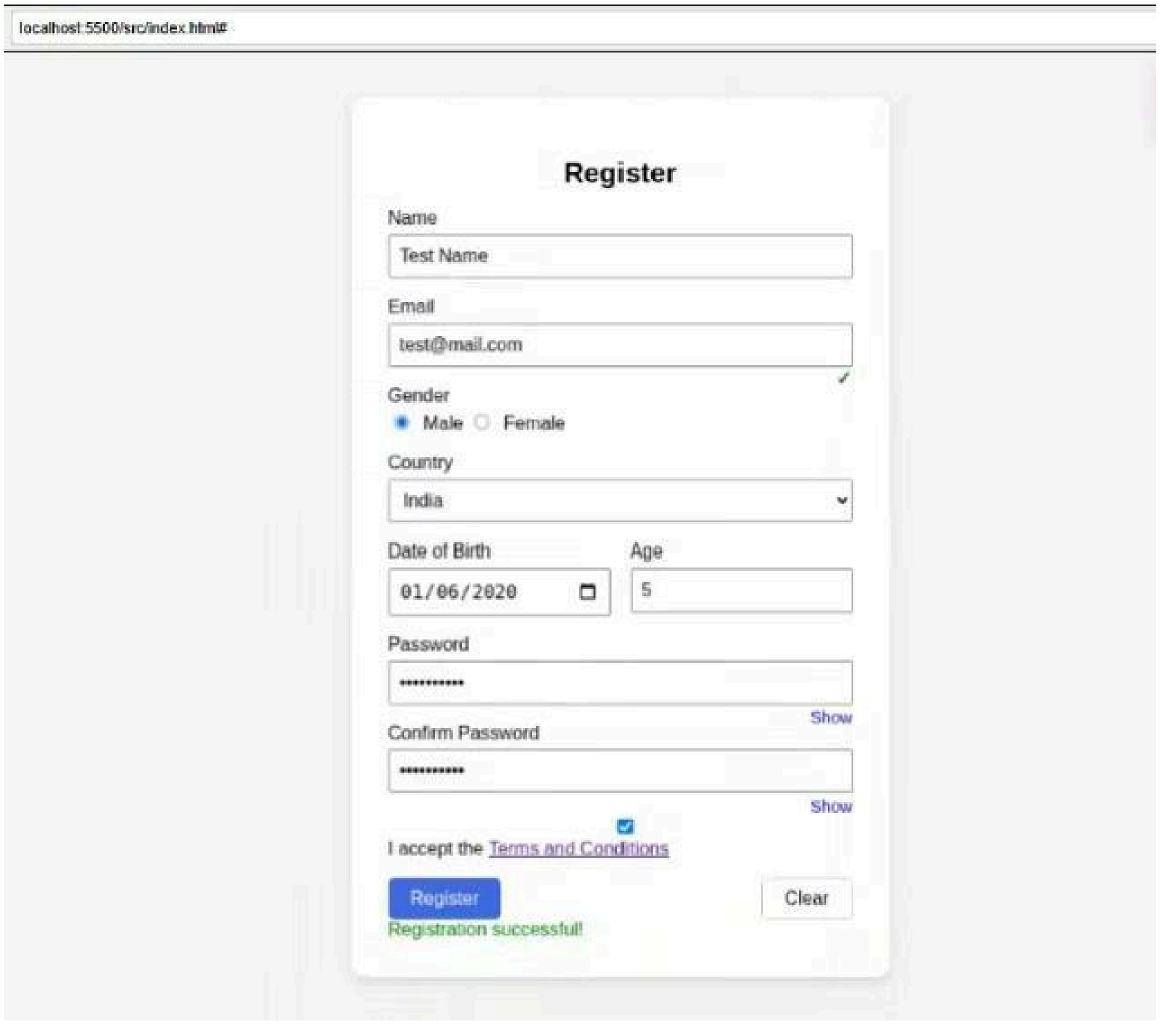
- This is a **web-based application**, so to view it in a browser, use the **internal browser inside the workspace.**
- Click on the **second last icon on the left panel** (the one labeled **"Browser Preview"**). This will open a tab within VS Code where you can **launch and view your application.**
- Note: The application **will not open in your system's local browser** — it must be viewed using the internal browser.



In the **Browser Preview tab**, type the following URL in the address bar and press **Enter**:

Your file is being served on: `localhost:5500/src/index.html`

This will load your HTML file and display the output of your web page **inside the internal browser**.



localhost:5500/src/index.html#

Register

Name: Test Name

Email: test@mail.com

Gender: [✓ Male] Female

Country: India

Date of Birth: 01/06/2020

Age: 5

Password: ••••••••••• Show

Confirm Password: ••••••••••• Show

[✓] I accept the Terms and Conditions

Register | Clear

Registration successful!

Step 4:

- Go back to the **terminal** and type the following command, then press **Enter**:

  node src/test/custom-parser.js

- This command will **execute the validation script** and display the test results for your HTML file in the terminal.

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. **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.**
5. **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
6. **You can follow series of command to setup environment once you are in your project-name folder:**
   a. **npm install -> Will install all dependencies -> takes 10 to 15 min.**
   b. **node src/test/custom-parser.js -> to run all test cases. It is mandatory to run this command before submission of workspace -> takes 5 to 6 min.**
7. **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.**