Step 1: Document Declaration

<!DOCTYPE html>

- Tells the browser this is an **HTML5 document**.
- Ensures modern browsers render the page correctly.

Step 2: Root HTML Element

<html lang="en">

- The <html> tag wraps all HTML content.
- lang="en" tells the browser (and accessibility tools like screen readers) that the page is in **English**.

Step 3: Head Section

<head>

<meta charset="utf-8">

<title>Agri Chatbot - Register</title>

<link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">

</head>

- <head>: Contains metadata and links, not visible content.

1. <meta charset="utf-8"> → ensures the page supports all characters like emojis, accents, etc.
2. <title> → sets the browser tab title ("Agri Chatbot - Register").
3. <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
   - Links to the **CSS file** for styling.
   - {{ url_for('static', filename='style.css') }} is **Flask syntax** that generates the URL to your style.css file in the static folder.

Step 4: Body Section

<body class="bg">

- <body> contains all **visible content** of the page.
- class="bg" → applies **background styles** defined in CSS.

Step 5: Form Container

<div class="auth-container">

- A <div> acts as a **box** around the registration form.
- class="auth-container" allows CSS to style it (e.g., centering, padding, border).

Step 6: Form Heading

```
<h2>Create Account</h2>
```

- Simple heading that tells the user this is the **account registration page**.

Step 7: Flash Messages (Flask)

```
{% with messages = get_flashed_messages(with_categories=true) %}
 {% if messages %}
  {% for cat,msg in messages %}
   <div class="flash {{ cat }}">{{ msg }}</div>
  {% endfor %}
 {% endif %}
{% endwith %}
```

- Flask feature to show **temporary messages** (success/error/info).

1. get_flashed_messages(with_categories=true) → fetches messages stored on the server.
2. {% if messages %} → checks if there are messages.
3. {% for cat,msg in messages %} → loops through messages.
4. <div class="flash {{ cat }}">{{ msg }}</div> → displays each message in a **styled box**, e.g., green for success, red for error.

Step 8: Registration Form

```
<form method="post">
```

- Begins a form.
- method="post" → data will be sent **securely** to the server.

Username Field

```
<input name="username" placeholder="Username" required>
```

- Input for the **username**.
- placeholder="Username" → gray hint text.
- required → cannot submit the form empty.

Password Field

<input name="password" type="password" placeholder="Password" required>

- Input for **password**.
- type="password" → hides the characters.
- required → user must fill it.

Submit Button

<button type="submit">Register</button>

- Button to **send the form data** to the server.

Step 9: Login Link

<p>Already have one? <a href="{{ url_for('index') }}">Login</a></p>

- Paragraph guiding existing users to **login instead of registering**.
- {{ url_for('index') }} → Flask generates the link to your login page.

Step 10: Closing Tags

</div>

</body>

</html>

- Closes the form container <div>.
- Closes <body> and <html>.

✅ **Summary of Flow for a User:**

1. User opens page → sees **Create Account** form.
2. User enters **username** and **password**.
3. User clicks **Register** → form sends data to Flask server.
4. Server processes the data.
5. If something is wrong (e.g., username exists), server sends a **flash message** → displayed on the page.
6. If successful → account is created.

```
User opens registration page

        |

        ▼

Browser loads HTML page with form

        |

        ▼

User enters username & password

        |

        ▼

User clicks "Register" button

        |

        ▼

Form sends POST request to Flask server

        |

        ▼

Flask server receives data

        |

        ▼

Server checks:

   ├─ Is username already taken?

   ├─ Is password valid?

        |

   ┌─────────┴─────────┐

   ▼          ▼

Yes, error      No, create account

(username exists)  |

   |          |

   ▼          ▼

Store flash     Store new user in database

error message  |

   |          |
Return page     Return page
with flash      with success flash
message         message
   |             |
   └──────▶ Browser displays message
                |
                ▼
User sees success or error message on page
```