1️⃣ Document type and language

<!DOCTYPE html>

<html lang="en">

- <!DOCTYPE html>:
    - Tells the browser that this is an HTML5 document.
    - Ensures modern browsers use **standards mode** instead of quirks mode (which can break layouts).
- <html lang="en">:
    - Starts the HTML document.
    - The lang="en" attribute indicates the document's language is **English**.
    - Important for accessibility (screen readers) and SEO.

2️⃣ Head section

<head>

<meta charset="utf-8">

<title>Agri Chatbot - Login</title>

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

</head>

- <head>: Contains metadata and resources for the page. Not visible in the main page body.
- <meta charset="utf-8">:
    - Ensures proper **text encoding**, so special characters like emojis 🌾, accented letters, or non-Latin scripts display correctly.
- <title>Agri Chatbot - Login</title>:
    - The text shown in the **browser tab**.
    - Helps users identify the page if multiple tabs are open.
- <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">:
    - Loads your CSS file from the static folder.
    - url_for('static', filename='style.css') is a Flask function that generates the correct URL dynamically, so the CSS loads correctly even if the app is deployed to another domain or folder.
    - rel="stylesheet" tells the browser this is a CSS file.

3️⃣ Body and container

<body class="bg">

<h2>🌾 Agri Chatbot</h2>

- <body class="bg">:
  - Starts the visible part of the page.
  - class="bg" applies CSS styles (likely sets a background color, image, or gradient).
- :
  - A **container div** for the login form and messages.
  - CSS can style it with width, padding, shadows, or alignment.
- <h2>🌾 Agri Chatbot</h2>:
  - Page heading displayed above the login form.
  - The **emoji** 🌾 makes the interface friendlier and contextually agricultural.

4️⃣ Flask flash messages

{% with messages = get_flashed_messages(with_categories=true) %}

{% if messages %}

{% for cat,msg in messages %}

{{ msg }}

{% endfor %}

{% endif %}

{% endwith %}

- **Purpose:** Display temporary notifications like errors (Invalid password) or success messages (Account created successfully).
- **How it works in Flask:**
  1. In your Flask view function, you use flash("Message", "category").
  2. flash("Invalid username or password", "error")
  3. get_flashed_messages(with_categories=True) retrieves all messages with their categories.
- **Template explanation:**
  - {% with messages = get_flashed_messages(with_categories=true) %} → Assigns flashed messages to messages.

- {% if messages %} → Checks if any messages exist.
- {% for cat,msg in messages %} → Loops through each message.
- {{ msg }} → Creates a div for each message, with a CSS class equal to its category (error, success, etc.), so you can style it differently.

Example rendered HTML:

Invalid username or password

Account created successfully

5️⃣ Login form

```
<form method="post">
  <input name="username" placeholder="Username" required>
  <input name="password" type="password" placeholder="Password" required>
  <button type="submit">Login</button>
</form>
```

- <form method="post">:
  - Form sends data to the server **securely** using POST.
  - If method="get" were used, data would appear in the URL (not safe for passwords).
- <input name="username" placeholder="Username" required>:
  - A text field for the username.
  - placeholder is **greyed-out hint text** inside the field.
  - required forces the user to enter a value before submitting.
- <input name="password" type="password" placeholder="Password" required>:
  - Password input field; characters are hidden (dots or asterisks).
- <button type="submit">Login</button>:
  - Clicking it sends the form to the server.

How Flask handles it:

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
```

Authenticate user here

6️⃣ Registration link

New? Create an account

- : Paragraph for extra text.
- Create an account:
    - Link to the **registration page**.
    - url_for('register') dynamically generates the correct URL based on your Flask route.

User experience:

- If someone doesn't have an account, they can click the link to register.

7️⃣ Closing tags

</body>

</html>

- Closes all opened HTML tags.
- Ensures the document is **well-formed**, which prevents rendering issues in browsers.

✅ **Summary of functionality**

1. **Header & styling:** Loads CSS and sets page metadata.
2. **Flash messages:** Shows temporary notifications like login errors.
3. **Login form:** Collects username & password securely using POST.
4. **Registration link:** Helps new users create accounts.
5. **Flask template integration:** Uses url_for for static files and navigation, get_flashed_messages for alerts.
6. **User-friendly:** Includes emojis, placeholders, and required fields.