**📘 Overview: User Registration System (HTML + CSS + Flask + MySQL)**

**🔷 What is this Project?**

This project builds a **complete User Registration System** where users can enter their **Name**, **Email**, and **Password**, and their data will be stored in a **MySQL Database**. It's a mini full-stack web application.

| **Layer** | **Technology** | **Role** |
| --- | --- | --- |
| Frontend | HTML + CSS | UI Form + Styling |
| Backend | Python + Flask | Server logic + Request handling |
| Database | MySQL | Data storage |

**🔷 What You Will Learn:**

1. **HTML Forms** – How to collect user input.
2. **CSS Styling** – How to make forms look clean and centered.
3. **Flask** – How to build routes and handle POST data.
4. **MySQL** – How to store data from the form into a database.
5. **Integration** – How everything connects together as a system.

# 🡺 **📘 Full Book-Style Notes: Registration Form with HTML, CSS, Python Flask, and MySQL**

**📖 Overview**

**🔷 What is this Project?**

This project is a **User Registration System** where users enter their Name, Email, and Password in a form. This data is securely stored in a **MySQL database** using Python **Flask** for backend processing and **HTML/CSS** for the frontend interface.

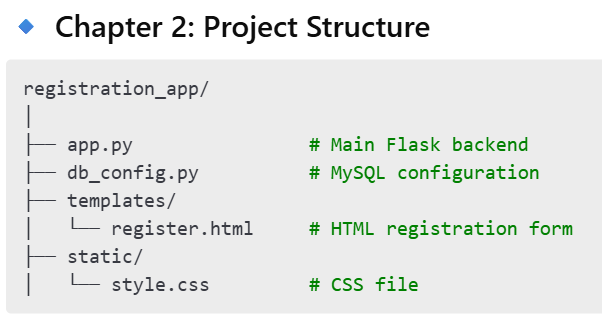
**🔷 Why Build This?**

* It is the **foundation** for most web applications.
* Helps you learn **Full Stack Development** (Frontend + Backend + Database).
* Covers real-world web technologies.
* Prepares you for bigger projects like login systems and dashboards.

## **🔹 Chapter 1: Introduction**

### ✅ Objective:

To develop a **complete user registration system** where users can submit their name, email, and password via a web form. The system stores this data securely in a MySQL database using Python and Flask as the backend and HTML + CSS for frontend presentation.



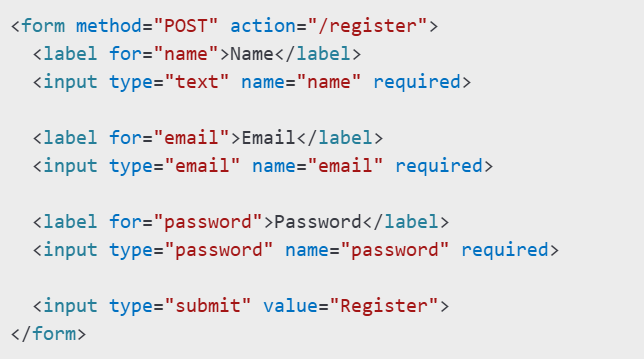
### 📁 Folder Roles:

* **templates/** → All HTML templates go here. Flask renders these pages.
* **static/** → CSS, JavaScript, and images. Flask links to these for UI design.
* **app.py** → This is where your backend logic lives.
* **db\_config.py** → Contains DB credentials to avoid repeating in your code.

## 🔹 Chapter 3: HTML (register.html)

### ✅ Purpose:

This HTML page builds the **User Interface** where the user enters registration details.



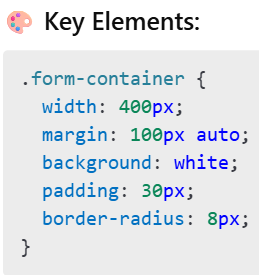
### 🧠 Key Points:

* method="POST" – Ensures data is sent securely.
* action="/register" – Tells the browser to send data to /register route.
* required – Adds basic client-side validation.

## 🔹 Chapter 4: CSS (style.css)

### ✅ Purpose:

Improves the **appearance and layout** of the HTML form.



## 🔹 Chapter 5: Flask Backend (app.py)

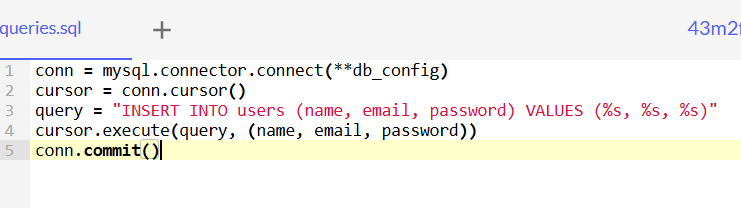
### ✅ What It Does:

* Renders HTML page
* Handles form data
* Connects and inserts data into MySQL



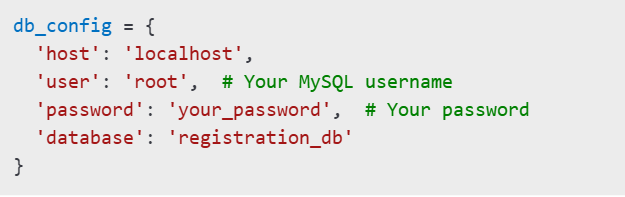


🡺Insert Data

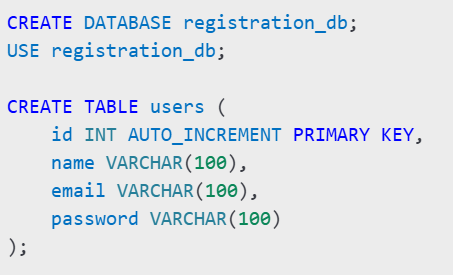


## **🔹 Chapter 6: MySQL Configuration (db\_config.py)**

### ✅ Centralized Configuration:

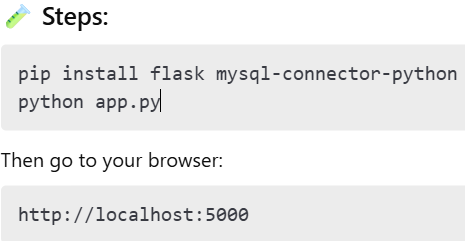


## **🔹 Chapter 7: MySQL Table Setup**



* AUTO\_INCREMENT creates a unique ID for every user.
* VARCHAR(100) sets character limits.

## **🔹 Chapter 8: Running the App**



CHAPTER 5 ☺ =>



**Explanation:**

**@app.route('/')**: 🡺

Yeh Flask ko batata hai ki jab user **root path** (i.e. http://localhost:5000/) pe visit karega, toh is function ko call karna.

**'/'** ka matlab hai homepage ya root URL. Jab koi user is URL pe jayega, toh Flask is route ko trigger karega.

**def home()**:

Yeh ek function hai, jo **home** route ko handle karega. Jab user **homepage** pe aayega, yeh function execute hoga.

**return render\_template('register.html')**:

Yeh line Flask ko batata hai ki **register.html** template ko render karna hai.

**register.html** ek HTML file hai jisme registration form hoga.

Jab user homepage pe aayega, toh usko registration form dikhai dega. **render\_template** function Flask mein HTML page ko display karne ke liye use hota hai.

🡺  
Jab koi user **home page** (i.e., localhost:5000/) open karega, toh usko registration form dikhega, jo **register.html** mein banaya gaya hai. 📝

2. Register Route (POST Request) 📝 🡺



**@app.route('/register', methods=['POST'])**:

Yeh route tab activate hota hai jab user **registration form** ko submit karta hai.

**/register** ka matlab hai ki jab user form ko submit karega, toh yeh request **POST method** ke through Flask backend ko send hogi.

**POST** ka use tab hota hai jab hum sensitive data (like password) send kar rahe hote hain.

 **def register()**:

* Yeh function tab execute hoga jab user form ko submit karega.

 **name = request.form['name']**:

* Yeh line form se **name** value ko fetch karegi. Jab user form mein apna naam dalta hai, toh woh yeh value backend mein yeh line retrieve karegi.
* **request.form** se hum form ke har input ko fetch karte hain.

 **email = request.form['email']**:

* **email** field ka data fetch kar raha hai yeh line. Jab user form mein apna email dalta hai, toh woh backend ko send ho jayega.

 **password = request.form['password']**:

* **password** field ka data fetch karna. Jab user password dalta hai, toh woh yeh line fetch karegi

#### **Explanation:**

\*\*conn = mysql.connector.connect(**db\_config)**:

Is line mein hum **MySQL** database se connect kar rahe hain. db\_config file mein jo credentials diye gaye hain (username, password, host), unka use karke hum **MySQL** se connection establish karte hain.

**cursor = conn.cursor()**:

cursor ek object hai jo SQL commands ko execute karta hai. **cursor** se hum query run karenge jo database mein data insert karegi.

**cursor.execute("INSERT INTO users (name, email, password) VALUES (%s, %s, %s)", (name, email, password))**:

Yeh SQL query hai jo user ke **name**, **email**, aur **password** ko **users** table mein insert karegi.

**%s** placeholders hain jahan hum data insert karenge (i.e., **name**, **email**, aur **password**).

**conn.commit()**:

commit() function data ko **finalize** karta hai. Matlab, data ab database mein permanently save ho gaya hai.

**cursor.close()** & **conn.close()**:

**cursor.close()** se hum cursor ko close karte hain jab kaam complete ho jaye.

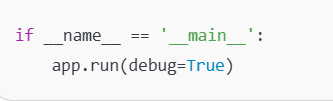
**conn.close()** se hum connection ko close karte hain.

**return "Registration Successful!"**:

Agar sab kuch sahi ho gaya, toh yeh message return hota hai jo user ko dikhega.

**except Exception as e**:

Agar koi error aata hai, toh woh error e mein capture hota hai aur **Error message** show hota hai.



#### 🡺 if \_\_name\_\_ == '\_\_main\_\_':

* **\_\_name\_\_** ek special variable hai jo har Python file mein hota hai.

Agar aap directly apni Python file ko run kar rahe ho (i.e., **python app.py**), toh **\_\_name\_\_** ka value **'\_\_main\_\_'** ho jata hai.

Matlab, jab yeh file directly run hoti hai, toh **Flask app** ko **run** karne ka code execute hoga.

app.run(debug=True)

**app.run()**: Yeh command Flask app ko start karti hai.

**debug=True**: Iska matlab hai jab app run ho, toh agar koi error aata hai, toh wo **browser** pe directly show hoga.

**debug=True** se **auto reload** bhi hota hai, matlab jab aap code mein changes karenge, toh app automatically reload ho jayega bina manual restart ke. 👨‍💻🔄

**WhatsApp Language Explanation:**

**if \_\_name\_\_ == '\_\_main\_\_':** - Jab directly file ko run karte ho, tab yeh condition **True** hoti hai, aur Flask app run karne ka process start hota hai. 🏃‍♂️💨

**app.run(debug=True)** - Yeh line Flask server ko start karti hai, aur agar koi error hota hai, toh wo browser mein dikhai dega. 💻⚠️