🧠 Understanding: Frontend vs Backend vs Database

Layer	Meaning	Example in This Project
Frontend	What the user sees and interacts with	The **HTML form** you fill in your browser
Backend	Logic behind the scenes that processes user input	The **Flask app in Python** that receives form data
Database	Where data is stored permanently	The **MySQL table** where user info is saved

---

🔁 How They Work Together:

1. The **frontend** (HTML page) collects data (Name, Email)
2. When the form is submitted, it sends data to the **backend** (Flask)
3. The backend receives the data, processes it, and inserts it into the **database** (MySQL)
4. A confirmation is sent back to the **frontend** (like "Thanks, Gowtham!")

🧾 Real-Life Analogy:

Layer	Real-World Example
Frontend	Waiter taking your order
Backend	Kitchen that prepares the dish
Database	The order log / kitchen board

The waiter doesn’t cook (frontend), the kitchen does (backend), and they keep a log of every order (database).

🔥 Project: "Simple Web Form to Save Data"

✅ 1. Overview

Layer	Tech Used	Description
Frontend	HTML	A form to collect Name and Email
Backend	Python (Flask)	Handles form submissions using traditional routing
Database	MySQL	Stores the submitted user data

✅ 2. Prerequisites

- Python 3 installed
- MySQL server running locally
- Python packages: `flask`, `pymysql`

Install dependencies:

```
pip install flask pymysql
```

☑ 3. MySQL Setup

Login to MySQL and create database and table:

```
CREATE DATABASE test;


USE test;


CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100)
);
```

☑ 4. Project Folder Structure

```
project/
├── app.py
├── templates/
│   └── form.html
```

☑ 5. Frontend Code (HTML Form)

File: `templates/form.html`

```
<!DOCTYPE html>
```

```
<html>

<head>

    <title>User Form</title>

</head>

<body>

    <h2>Submit Your Details</h2>

    <form method="POST" action="/submit">

        <label>Name:</label><br>

        <input type="text" name="name" required><br><br>


        <label>Email:</label><br>

        <input type="email" name="email" required><br><br>


        <input type="submit" value="Submit">

    </form>

</body>

</html>
```

☑ 6. Backend Code (Flask + MySQL)

File: `app.py`

```
from flask import Flask, request, render_template

import pymysql
```

```
app = Flask(__name__)


# Connect to MySQL

db = pymysql.connect(

    host="localhost",

    user="root",         # Your MySQL username

    password="root",     # Your MySQL password

    database="test"      # Your DB name

)


@app.route('/')

def form():

    return render_template('form.html')


@app.route('/submit', methods=['POST'])

def submit():

    name = request.form['name']

    email = request.form['email']


    cursor = db.cursor()

    sql = "INSERT INTO users (name, email) VALUES (%s, %s)"

    cursor.execute(sql, (name, email))

    db.commit()
    return f"<h3>Thanks, {name}! Your data is saved in MySQL.</h3>"


if __name__ == '__main__':
    app.run(debug=True)
```

☑️ 7. How to Run the App

```
python app.py
```

Then visit http://localhost:5000 in your browser.

☑️ 8. What's Happening Behind the Scenes?

Step	Action
1️⃣	User opens the form in browser
2️⃣	User enters Name and Email, clicks submit
3️⃣	HTML sends form data to Flask backend (`/submit`) via POST
4️⃣	Flask reads data from `request.form`

5️⃣ Flask inserts data into MySQL table

6️⃣ User sees a thank-you confirmation message

✅ 9. How to Add in Resume

Project Title: Full Stack Web Form App using Flask and MySQL
Description:

> Built a full-stack application that collects user input from an HTML form and stores the data in a MySQL database using Python Flask. The backend handles traditional POST requests and processes form data without REST API. Used PyMySQL for database connection and routing with Jinja templates.

✅ 10. How to Explain in Interview

> "I built a simple full-stack app without using REST APIs. I used HTML forms for the frontend, Flask as the backend to handle POST submissions, and MySQL to store the user data. This helped me understand how traditional form-based web systems work before learning REST or frontend frameworks."