Flask takes form input → converts it into Python object → saves it into SQLite database → shows it back using HTML templates

- Creates a Flask app
- Uses SQLite database
- Stores form data
- Shows stored users

Install required package

```
pip install flask flask_sqlalchemy
```

FULL `app.py` CODE (UPDATED)

```
from flask import Flask, render_template, request
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)

# ---------------------------
# DATABASE CONFIGURATION
# ---------------------------
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///users.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

db = SQLAlchemy(app)

# ---------------------------
# USER TABLE MODEL
# ---------------------------
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)

    name = db.Column(db.String(100), nullable=False)
    email = db.Column(db.String(120), nullable=False, unique=True)
    phone = db.Column(db.String(15), nullable=False)

    def __repr__(self):
        return f"<User {self.name}>"

# ---------------------------
# CREATE DATABASE TABLE
# ---------------------------
with app.app_context():
    db.create_all()

# ---------------------------
```

```python
# HOME PAGE (FORM)
# ---------------------------
@app.route('/')
def home():
    return render_template('index.html')

# ---------------------------
# SAVE USER DATA
# ---------------------------
@app.route('/submit', methods=['POST'])
def submit():
    name = request.form['name']
    email = request.form['email']
    phone = request.form['phone']

    # Create new user object
    new_user = User(name=name, email=email, phone=phone)

    # Save to database
    db.session.add(new_user)
    db.session.commit()

    return f"User {name} saved successfully!"

# ---------------------------
# SHOW ALL USERS
# ---------------------------
@app.route('/users')
def users():
    all_users = User.query.all()
    return render_template('users.html', users=all_users)

# ---------------------------
# RUN APP
# ---------------------------
if __name__ == '__main__':
    app.run(debug=True)
```

□ REQUIRED HTML FORM (`index.html`)

```html
<!DOCTYPE html>
<html>
<head>
    <title>User Form</title>
</head>
<body>

    <h2>Register User</h2>

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

        <input type="text" name="name" placeholder="Enter Name" required>
        <br><br>
```

```
        <input type="email" name="email" placeholder="Enter Email" required>
        <br><br>

        <input type="text" name="phone" placeholder="Enter Phone" required>
        <br><br>

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

    </form>

    <br>
    <a href="/users">View Users</a>

</body>
</html>
```

□ USERS DISPLAY PAGE (`users.html`)

```
<!DOCTYPE html>
<html>
<head>
    <title>Users List</title>
</head>
<body>

    <h2>All Users</h2>

    <table border="1" cellpadding="10">

        <tr>
            <th>ID</th>
            <th>Name</th>
            <th>Email</th>
            <th>Phone</th>
        </tr>

        {% for user in users %}
        <tr>
            <td>{{ user.id }}</td>
            <td>{{ user.name }}</td>
            <td>{{ user.email }}</td>
            <td>{{ user.phone }}</td>
        </tr>
        {% endfor %}

    </table>

    <br>
    <a href="/">Go Back</a>

</body>
</html>
```

□ HOW THIS WORKS

1. User fills form:

```
Name: omsir
Email: omsir@gmail.com
Phone: 8149996597
```

---

2. Flask receives:

```
name = request.form['name']
email = request.form['email']
phone = request.form['phone']
```

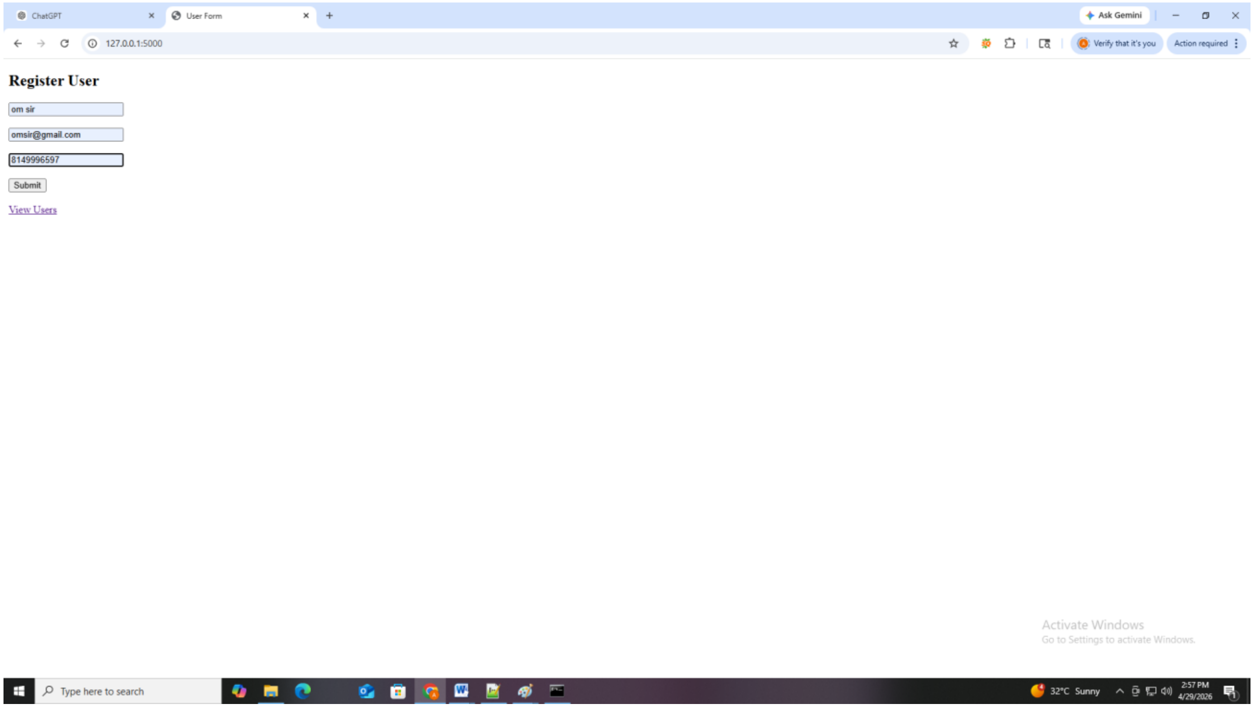
3. Stored in database:

```
new_user = User(name=name, email=email, phone=phone)
db.session.add(new_user)
db.session.commit()
```

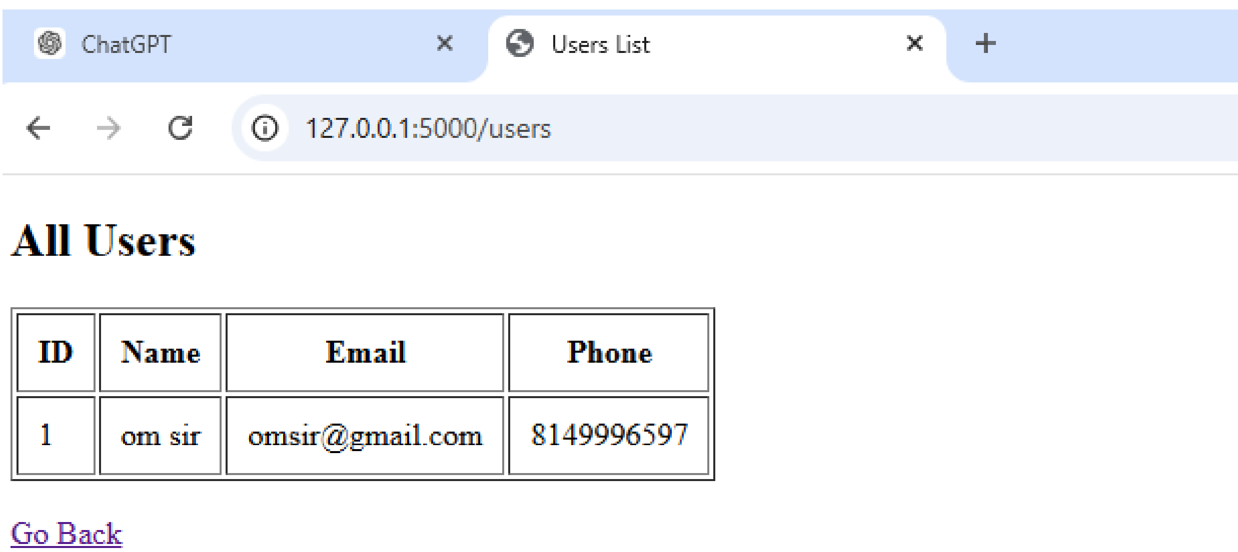
---

4. Database saves:

id	name	email	phone
1	omsir	omsir@gmail.com	8149996597



And user list:-



I’ll explain your code **line by line in a clear, structured way** so you understand exactly what is happening.

□ 1. Importing required modules

```
from flask import Flask, render_template, request
from flask_sqlalchemy import SQLAlchemy
```

Meaning:

- **Flask** → creates your web application
- **render_template** → loads HTML files (like index.html)
- **request** → gets data from user (forms, input, etc.)
- **SQLAlchemy** → connects Python to database easily

□ In simple words:

“Bring tools needed to build website + database”

□ 2. Creating Flask app

```
app = Flask(__name__)
```

Meaning:

- Creates your Flask application
- `__name__` tells Python this is the main file

□ Think:

“Start my web app here”

□ 3. Database configuration

```
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///users.db'
```

Meaning:

- Tells Flask to use SQLite database
- Database file will be: `users.db`

```
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
```

Meaning:

- Turns off extra tracking system
- Improves performance
- Prevents warnings

□ 4. Connecting database to Flask

```
db = SQLAlchemy(app)
```

Meaning:

- Links database system with Flask app
- Now you can create tables and store data

□ 5. Creating User table (MODEL)

```
class User(db.Model):
```

Meaning:

- Creates a table called `User`

Columns:

ID column

```
id = db.Column(db.Integer, primary_key=True)
```

- Unique number for each user
- Auto-increases
- Primary key = main identifier

Name column

```
name = db.Column(db.String(100), nullable=False)
```

- Stores user name
- Max 100 characters
- Cannot be empty

Email column

```
email = db.Column(db.String(120), nullable=False, unique=True)
```

- Stores email
- Must be unique (no duplicates allowed)

Phone column

```
phone = db.Column(db.String(15), nullable=False)
```

- Stores phone number
- Cannot be empty

Debug representation:

```
def __repr__(self):
    return f"<User {self.name}>"
```

Meaning:

- Helps display user in readable format while debugging

□ 6. Creating database tables

```
with app.app_context():
    db.create_all()
```

Meaning:

- Creates database file (`users.db`)
- Creates User table inside it

□ If already exists, it will NOT change it

□ 7. Home route (form page)

```
@app.route('/')
def home():
    return render_template('index.html')
```

Meaning:

- / is homepage URL
- Loads `index.html`

□ 8. Submit route (form handling)

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

Meaning:

- This route only accepts POST requests
- Used when form is submitted

Function starts:

```
def submit():
```

- This runs when form is submitted

Getting form data:

```
name = request.form['name']
email = request.form['email']
phone = request.form['phone']
```

Meaning:

- Reads data from HTML form inputs

Example form:

```
<input name="name">
<input name="email">
<input name="phone">
```

Creating new user:

```
new_user = User(name=name, email=email, phone=phone)
```

Meaning:

- Creates a Python object (not saved yet)
- Prepares data for database

Save to database:

```
db.session.add(new_user)
```

- Adds user to session (temporary stage)

```
db.session.commit()
```

- Permanently saves to database

Response:

```
return f"User {name} saved successfully!"
```

- Sends message back to browser

□ 9. Show all users

```
@app.route('/users')
def users():
```

Meaning:

- Creates `/users` page

Get all users:

```
all_users = User.query.all()
```

Meaning:

- Fetches all rows from database

Send to HTML:

```
return render_template('users.html', users=all_users)
```

Meaning:

- Passes data to HTML file
- `users` variable available in template

□ 10. Run the app

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

Meaning:

- Runs Flask server
- `debug=True`:
  - auto reloads on code changes
  - shows errors clearly

□ FULL FLOW SUMMARY

1. User opens:

```
/
```

→ sees form

---

2. User submits form:

```
/submit (POST)
```

→ Flask:

- reads form data
- saves to database

---

3. Data stored in:
`users.db`

---

4. User opens:
`/users`

→ sees all saved users

□ ONE-LINE SUMMARY

Flask takes form input → converts it into Python object → saves it into SQLite database → shows it back using HTML templates

We'll upgrade your project into a **FULL CRUD app**:

CRUD =

- **C**reate → add user
- **R**ead → view users
- **U**pdate → edit user
- **D**elete → remove user

We'll use your same setup: Flask + SQLAlchemy + SQLite.

□ FINAL PROJECT STRUCTURE

```
project/
│
├── app.py
└── templates/
    ├── index.html
    ├── users.html
    └── edit.html
```

□ FULL `app.py` (COMPLETE CRUD)

```
from flask import Flask, render_template, request, redirect, url_for
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)

# ---------------- DATABASE ----------------
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///users.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

db = SQLAlchemy(app)

# ---------------- MODEL ----------------
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100), nullable=False)
    email = db.Column(db.String(120), nullable=False, unique=True)
    phone = db.Column(db.String(15), nullable=False)

# ---------------- CREATE DB ----------------
with app.app_context():
    db.create_all()
```

```python
# ---------------- HOME (CREATE FORM) ----------------
@app.route('/')
def home():
    return render_template('index.html')

# ---------------- CREATE USER ----------------
@app.route('/create', methods=['POST'])
def create():
    name = request.form['name']
    email = request.form['email']
    phone = request.form['phone']

    user = User(name=name, email=email, phone=phone)
    db.session.add(user)
    db.session.commit()

    return redirect(url_for('users'))

# ---------------- READ USERS ----------------
@app.route('/users')
def users():
    all_users = User.query.all()
    return render_template('users.html', users=all_users)

# ---------------- DELETE USER ----------------
@app.route('/delete/<int:id>')
def delete(id):
    user = User.query.get_or_404(id)
    db.session.delete(user)
    db.session.commit()
    return redirect(url_for('users'))

# ---------------- EDIT PAGE ----------------
@app.route('/edit/<int:id>')
def edit(id):
    user = User.query.get_or_404(id)
    return render_template('edit.html', user=user)

# ---------------- UPDATE USER ----------------
@app.route('/update/<int:id>', methods=['POST'])
def update(id):
    user = User.query.get_or_404(id)

    user.name = request.form['name']
    user.email = request.form['email']
    user.phone = request.form['phone']

    db.session.commit()
    return redirect(url_for('users'))

# ---------------- RUN APP ----------------
if __name__ == '__main__':
    app.run(debug=True)
```

□ 1. `index.html` (CREATE FORM)

```
<h2>Add User</h2>

<form action="/create" method="POST">
    <input type="text" name="name" placeholder="Name" required><br><br>
    <input type="email" name="email" placeholder="Email" required><br><br>
    <input type="text" name="phone" placeholder="Phone" required><br><br>

    <button type="submit">Create</button>
</form>

<br>
<a href="/users">View Users</a>
```

□ 2. `users.html` (READ + DELETE + EDIT)

```
<h2>All Users</h2>

<table border="1" cellpadding="10">

    <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Email</th>
        <th>Phone</th>
        <th>Actions</th>
    </tr>

    {% for user in users %}
    <tr>
        <td>{{ user.id }}</td>
        <td>{{ user.name }}</td>
        <td>{{ user.email }}</td>
        <td>{{ user.phone }}</td>
        <td>
            <a href="/edit/{{ user.id }}">Edit</a>
            |
            <a href="/delete/{{ user.id }}">Delete</a>
        </td>
    </tr>
    {% endfor %}

</table>

<br>
<a href="/">Add New User</a>
```

✏️ 3. `edit.html` (UPDATE FORM)

```
<h2>Edit User</h2>

<form action="/update/{{ user.id }}" method="POST">

    <input type="text" name="name" value="{{ user.name }}" required><br><br>

    <input type="email" name="email" value="{{ user.email }}"
required><br><br>

    <input type="text" name="phone" value="{{ user.phone }}"
required><br><br>

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

</form>

<br>
<a href="/users">Back</a>
```

HOW CRUD WORKS

CREATE

```
/create (POST)
→ saves new user
```

READ

```
/users
→ shows all users
```

UPDATE

```
/edit/<id>
→ shows edit form

/update/<id>
→ saves changes
```

DELETE

```
/delete/<id>
→ removes user
```

□ REAL FLOW

1. User opens:

```
/
```

2. Adds data → CREATE

3. Redirects to:

```
/users
```

4. Sees list

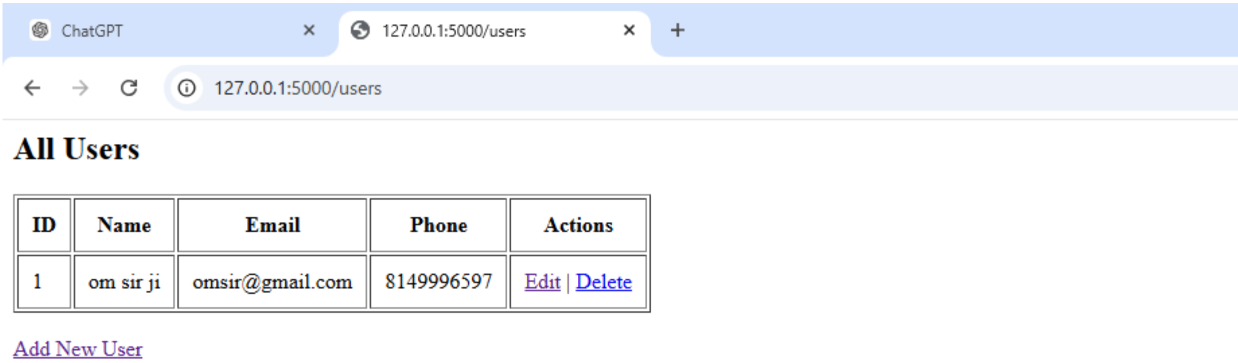
5. Clicks:

- Edit → update data
- Delete → remove user

□ WHAT YOU JUST BUILT

✔ Full backend system
✔ Database integration
✔ CRUD operations
✔ Real-world architecture

Output:-

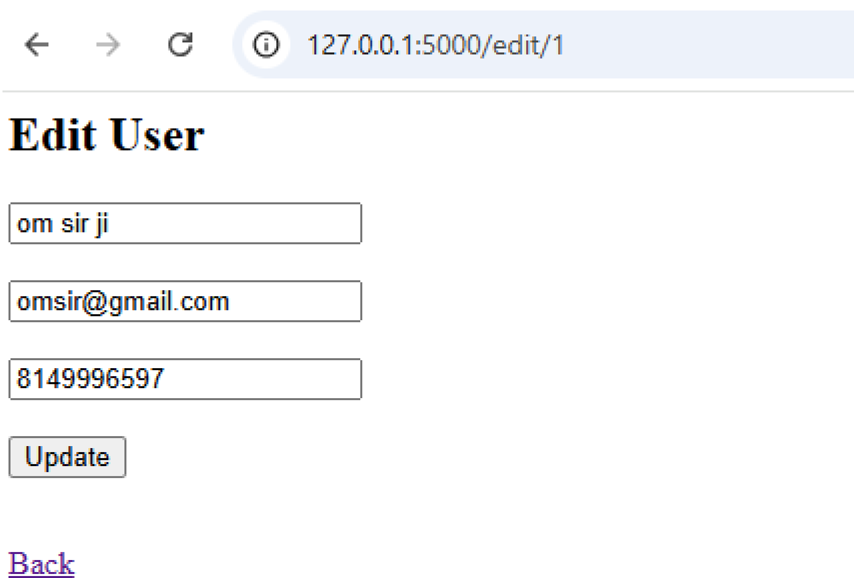


All Users

ID	Name	Email	Phone	Actions
1	om sir ji	omsir@gmail.com	8149996597	Edit | Delete

Add New User

After click on edit



Edit User

om sir ji

omsir@gmail.com

8149996597

Update

Back