how to create simple web application using flask in python with crud operation for user management for table user id, Full name, email id with explanation of each separate file in project

Create simple web application using flask and mysql in python with crud operation for user management for table user id, Full name, email id with explanation of each separate file in project

To create a **Flask** web application using **MySQL** for **CRUD (Create, Read, Update, Delete) operations** for user management, follow this step-by-step guide.

**Step 1: Install Required Packages**

Ensure you have the necessary Python packages installed:

pip install flask flask-mysql flask-sqlalchemy flask-wtf

**Step 2: Project Structure**

arduino

CopyEdit

flask\_crud\_mysql/

│── app.py

│── config.py

│── models.py

│── forms.py

│── database.py

│── static/

│── templates/

│ ├── base.html

│ ├── user\_list.html

│ ├── user\_form.html

│ ├── user\_confirm\_delete.html

**Step 3: Set Up the MySQL Database**

Create a MySQL database and user:

CREATE DATABASE flask\_crud\_db;

CREATE USER 'flaskuser'@'localhost' IDENTIFIED BY 'password';

GRANT ALL PRIVILEGES ON flask\_crud\_db.\* TO 'flaskuser'@'localhost';

FLUSH PRIVILEGES;

**Step 4: Configure Flask (config.py)**

Create a config.py file to store database configurations.

import os

DB\_USERNAME = "flaskuser"

DB\_PASSWORD = "password"

DB\_NAME = "flask\_crud\_db"

DB\_HOST = "localhost"

SQLALCHEMY\_DATABASE\_URI = f"mysql+pymysql://{DB\_USERNAME}:{DB\_PASSWORD}@{DB\_HOST}/{DB\_NAME}"

SQLALCHEMY\_TRACK\_MODIFICATIONS = False

SECRET\_KEY = os.urandom(24) # Needed for form security

**Step 5: Database Connection (database.py)**

Create a database.py file to initialize the database.

from flask\_sqlalchemy import SQLAlchemy

db = SQLAlchemy()

**Step 6: Define the User Model (models.py)**

Create a models.py file to define the MySQL table schema.

from database import db

class User(db.Model):

id = db.Column(db.Integer, primary\_key=True)

full\_name = db.Column(db.String(100), nullable=False)

email = db.Column(db.String(100), unique=True, nullable=False)

def \_\_repr\_\_(self):

return f'<User {self.full\_name}>'

**Step 7: Create Forms (forms.py)**

Create a forms.py file to handle user input.

from flask\_wtf import FlaskForm

from wtforms import StringField, SubmitField

from wtforms.validators import DataRequired, Email

class UserForm(FlaskForm):

full\_name = StringField('Full Name', validators=[DataRequired()])

email = StringField('Email', validators=[DataRequired(), Email()])

submit = SubmitField('Save')

**Step 8: Build Flask Routes (app.py)**

Create app.py, which serves as the main entry point.

from flask import Flask, render\_template, request, redirect, url\_for

from config import SQLALCHEMY\_DATABASE\_URI, SECRET\_KEY

from database import db

from models import User

from forms import UserForm

app = Flask(\_\_name\_\_)

app.config['SQLALCHEMY\_DATABASE\_URI'] = SQLALCHEMY\_DATABASE\_URI

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

app.config['SECRET\_KEY'] = SECRET\_KEY

db.init\_app(app)

# Create tables

@app.before\_first\_request

def create\_tables():

db.create\_all()

# Read - Display all users

@app.route('/')

def user\_list():

users = User.query.all()

return render\_template('user\_list.html', users=users)

# Create - Add a new user

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

def user\_create():

form = UserForm()

if form.validate\_on\_submit():

new\_user = User(full\_name=form.full\_name.data, email=form.email.data)

db.session.add(new\_user)

db.session.commit()

return redirect(url\_for('user\_list'))

return render\_template('user\_form.html', form=form)

# Update - Edit a user

@app.route('/update/<int:user\_id>', methods=['GET', 'POST'])

def user\_update(user\_id):

user = User.query.get\_or\_404(user\_id)

form = UserForm(obj=user)

if form.validate\_on\_submit():

user.full\_name = form.full\_name.data

user.email = form.email.data

db.session.commit()

return redirect(url\_for('user\_list'))

return render\_template('user\_form.html', form=form)

# Delete - Remove a user

@app.route('/delete/<int:user\_id>', methods=['GET', 'POST'])

def user\_delete(user\_id):

user = User.query.get\_or\_404(user\_id)

if request.method == 'POST':

db.session.delete(user)

db.session.commit()

return redirect(url\_for('user\_list'))

return render\_template('user\_confirm\_delete.html', user=user)

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

app.run(debug=True)

**Step 9: Create HTML Templates**

Inside the templates/ directory, create:

**Base Template (base.html)**

<!DOCTYPE html>

<html>

<head>

<title>Flask CRUD MySQL</title>

</head>

<body>

<h1>User Management</h1>

<a href="{{ url\_for('user\_list') }}">Home</a>

<hr>

{% block content %}{% endblock %}

</body>

</html>

**User List (user\_list.html)**

{% extends 'base.html' %}

{% block content %}

<h2>Users</h2>

<a href="{{ url\_for('user\_create') }}">Add New User</a>

<ul>

{% for user in users %}

<li>{{ user.full\_name }} - {{ user.email }}

<a href="{{ url\_for('user\_update', user\_id=user.id) }}">Edit</a>

<a href="{{ url\_for('user\_delete', user\_id=user.id) }}">Delete</a>

</li>

{% endfor %}

</ul>

{% endblock %}

**User Form (user\_form.html)**

{% extends 'base.html' %}

{% block content %}

<h2>User Form</h2>

<form method="post">

{{ form.hidden\_tag() }}

<p>{{ form.full\_name.label }} {{ form.full\_name }}</p>

<p>{{ form.email.label }} {{ form.email }}</p>

<p>{{ form.submit }}</p>

</form>

{% endblock %}

**Delete Confirmation (user\_confirm\_delete.html)**

{% extends 'base.html' %}

{% block content %}

<h2>Are you sure you want to delete {{ user.full\_name }}?</h2>

<form method="post">

<button type="submit">Yes, Delete</button>

</form>

<a href="{{ url\_for('user\_list') }}">Cancel</a>

{% endblock %}

**Step 10: Initialize the Database and Run the Application**

**Step 10.1: Create the Database Tables**

Run the following command in Python:

python app.py

This will create tables in MySQL.

**Step 10.2: Start the Flask Server**

Run:

python app.py

**Step 10.3: Open the App in a Browser**

Visit:  
👉 http://127.0.0.1:5000/

**Summary of Files**

csharp

CopyEdit

flask\_crud\_mysql/

│── app.py # Main Flask app

│── config.py # Database configuration

│── models.py # User model

│── forms.py # Flask-WTF forms

│── database.py # Database connection

│── static/ # Static files (CSS, JS)

│── templates/ # HTML templates

│ ├── base.html

│ ├── user\_list.html

│ ├── user\_form.html

│ ├── user\_confirm\_delete.html

**Conclusion**

This **Flask + MySQL** web application supports **CRUD operations** for managing users. You can extend it with authentication, pagination, or Bootstrap styling.