Ali Nasir

232931

BSCYSev-F23-4A

SSDD LAB

**Development of a CRUD Web Application using Flask on Linux**

**1. Objective**

The purpose of this project is to develop a web application using the **Flask** framework that performs **CRUD (Create, Read, Update, Delete)** operations on a student database. The app is implemented on a **Linux system**, styled using **Bootstrap**, and backed by a **SQLite** database using **SQLAlchemy ORM**.

**2. System Requirements**

* **Operating System:** Linux (Ubuntu or Debian-based)
* **Programming Language:** Python 3
* **Development Tools:**
  + Python3 and pip
  + Flask and Flask-SQLAlchemy
  + Visual Studio Code (optional)
  + HTML/CSS with Bootstrap
  + SQLite

**3. Implementation Procedure**

**Step 1: Environment Setup**

1. Install Python and pip:
2. sudo apt update
3. sudo apt install python3 python3-pip python3-venv
4. Create the project directory:
5. mkdir flaskapp
6. cd flaskapp
7. Set up a virtual environment:
8. python3 -m venv env
9. source env/bin/activate
10. Install Flask and dependencies:
11. pip install flask flask-sqlalchemy

**Step 2: Project Structure**

flaskapp/

│

├── app.py

├── firstapp.db ← created later

├── env/ ← virtual environment

├── static/

└── templates/

├── index.html

└── update.html

**Step 3: Application Code – app.py**

Handles routing, form processing, database operations.

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

from flask\_sqlalchemy import SQLAlchemy

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

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///firstapp.db'

db = SQLAlchemy(app)

class Student(db.Model):

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

fname = db.Column(db.String(50))

lname = db.Column(db.String(50))

age = db.Column(db.Integer)

city = db.Column(db.String(50))

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

def index():

if request.method == 'POST':

fname = request.form['fname']

lname = request.form['lname']

age = request.form['age']

city = request.form['city']

new\_student = Student(fname=fname, lname=lname, age=age, city=city)

db.session.add(new\_student)

db.session.commit()

data = Student.query.all()

return render\_template('index.html', data=data)

@app.route('/delete/<int:sno>')

def delete(sno):

student = Student.query.get\_or\_404(sno)

db.session.delete(student)

db.session.commit()

return redirect('/')

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

def update(sno):

student = Student.query.get\_or\_404(sno)

if request.method == 'POST':

student.fname = request.form['fname']

student.lname = request.form['lname']

student.age = request.form['age']

student.city = request.form['city']

db.session.commit()

return redirect('/')

return render\_template('update.html', student=student)

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

app.run(debug=True)

**Step 4: Database Initialization**

python

>>> from app import app, db

>>> app.app\_context().push()

>>> db.create\_all()

>>> exit()

Creates the SQLite file firstapp.db with the Student table.

**Step 5: HTML Templates**

**templates/index.html**

Displays form and list of students.

<!DOCTYPE html>

<html>

<head>

<title>Student Manager</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

<div class="container mt-4">

<h2>Add New Student</h2>

<form method="POST">

<input type="text" name="fname" placeholder="First Name" class="form-control mb-2" required>

<input type="text" name="lname" placeholder="Last Name" class="form-control mb-2" required>

<input type="number" name="age" placeholder="Age" class="form-control mb-2" required>

<input type="text" name="city" placeholder="City" class="form-control mb-2" required>

<button type="submit" class="btn btn-success">Add</button>

</form>

<h2 class="mt-5">Student List</h2>

<table class="table table-bordered">

<thead>

<tr>

<th>ID</th><th>First</th><th>Last</th><th>Age</th><th>City</th><th>Actions</th>

</tr>

</thead>

<tbody>

{% for student in data %}

<tr>

<td>{{ student.sno }}</td>

<td>{{ student.fname }}</td>

<td>{{ student.lname }}</td>

<td>{{ student.age }}</td>

<td>{{ student.city }}</td>

<td>

<a href="/update/{{ student.sno }}" class="btn btn-primary btn-sm">Edit</a>

<a href="/delete/{{ student.sno }}" class="btn btn-danger btn-sm">Delete</a>

</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

</body>

</html>

**templates/update.html**

Used to edit a student record.

<!DOCTYPE html>

<html>

<head>

<title>Edit Student</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

<div class="container mt-4">

<h2>Edit Student</h2>

<form method="POST">

<input type="text" name="fname" value="{{ student.fname }}" class="form-control mb-2">

<input type="text" name="lname" value="{{ student.lname }}" class="form-control mb-2">

<input type="number" name="age" value="{{ student.age }}" class="form-control mb-2">

<input type="text" name="city" value="{{ student.city }}" class="form-control mb-2">

<button type="submit" class="btn btn-primary">Update</button>

</form>

</div>

</body>

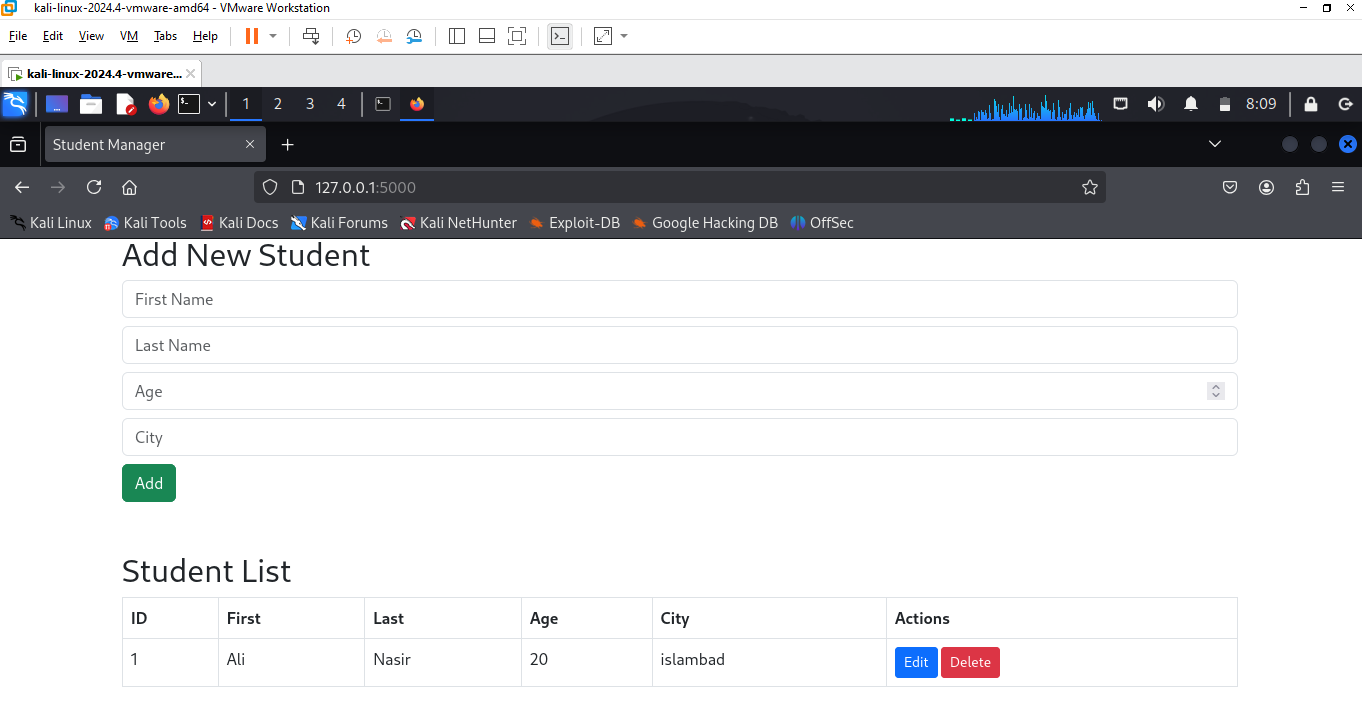
</html>

**Step 6: Run the Application**

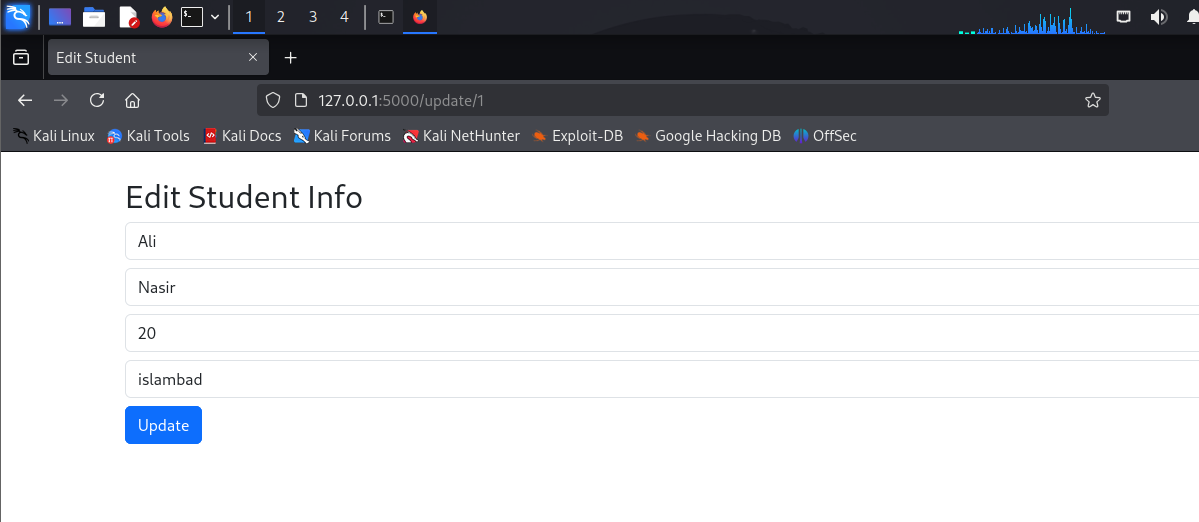
python app.py

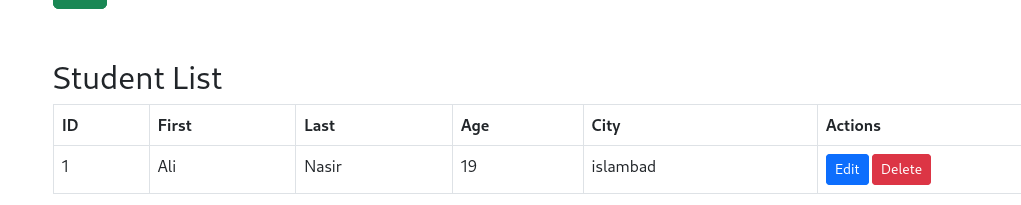
Access via:

<http://localhost:5000>



Now let’s update some data

1. Click on the edit icon , below interface will display
2. Make change and click on update, age is changed from 20 to 19 ,Below is the updated one



**4. Observations and Results**

* The app loads successfully on localhost:5000.
* New student entries can be added via the form.
* Entries appear in a Bootstrap-styled table.
* Each entry can be updated or deleted in real-time.

**5. Conclusion**

This project demonstrates the creation of a basic but functional web-based CRUD system using Python Flask on a Linux platform. It integrates back-end and front-end development and provides hands-on experience with templating, routing, and database management using SQLAlchemy ORM and SQLite.