Flask Framework

Route()-> this function says which URL should execute that function.

Dynamic Routing

@app.route('/about/<username>')  
def about\_page(username):  
 return f'<h1> About Page is of {username}</h1>'

Eg.2 Correct

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

def delete(id):

Eg.2 Wrong way to write

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

def delete(id):

How to run the app?

set FLASK\_APP=file\_name.py (Avoid space after =)

flask run (To run the command)

3.Jinja Template

{{}} 🡪 To access variables sent from routes in html.

e.g. @app.route('/market')  
def market\_page():

items = 1  
 return render\_template('market.html', items = items)

{{items}} 🡪 access items.

4. Jinja Template to write python code

{% %}🡪 you can write codes like if-else, for, etc.

e.g.

market.py

@app.route('/market')  
def market\_page():  
 items = [{'id': 1, 'name': 'Phone', 'barcode': '893212299897', 'price': 500},  
 {'id': 2, 'name': 'Laptop', 'barcode': '123985473165', 'price': 900},  
 {'id': 3, 'name': 'Keyboard', 'barcode': '231985128446', 'price': 150}]  
 return render\_template('market.html', items = items)

market.html

{% for item in items %} --> Start for loop  
<!-- Your rows inside the table HERE: -->  
 <tr>  
 <td>{{item.id}}</td>  
 <td>{{item.name}}</td>  
 <td>{{item.barcode}}</td>  
 <td>{{item.price}}</td>  
 </tr>  
{% endfor %} --> end for loop

5. Template Inheritance

Base.html

<head>

{{% block title %}}

{{% endblock %}}

</head>

<body>

{{% block content %}}

{{% endblock %}}

</body>

App.html

{{% extends ‘Base.html’ %}} 🡪 Inherits base.html design and contents.

{{% block title%}} 🡪 We can add our Title in inherited page.

{{%endblock%}}

{{% block content %}} 🡪 We can add our own content in inherited page.

{{% endblock %}}

6. Linking Pages using Jinja Templates

Market.py

@app.route('/home')  
@app.route('/')  
def home\_page():  
 return render\_template("home.html")

base.html

<a class="nav-link" href="{{ url\_for('home\_page') }}">Home <span class="sr-only">(current)</span></a>

7. SQLite DB Connection

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///market.db'  
db = SQLAlchemy(app) #database instance Passing application as an argument in db instance  
  
class Item(db.Model):  
 id = db.Column(db.Integer(), primary\_key = True)  
 name = db.Column(db.String(length = 30), nullable=False, unique=True)  
 price = db.Column(db.Integer(), nullable=False)  
 barcode = db.Column(db.String(length=12), nullable=False, unique=True)  
 description = db.Column(db.String(length = 1024), nullable=False, unique=True)  
  
 def \_\_repr\_\_(self):  
 return f'Item {self.name}'

In terminal write

from market import db

db.create\_all() 🡪 This statement will create market.db file in directory

from market import Item 🡪name of the table

item1 = Item(name=”name”, price=”1”, barcode=”123456789012”,description=”desc”) 🡪 Creating instance of item

db.session.add(item1) 🡪 To add item1 to db

db.sssion.commit() 🡪 commits(persists) the changes. (changes bane rahte hai)

Item.query.all() 🡪 it is returning records in Item table.

o/p: [<Item1>]

***This for loop is to return each value in table.***

for item in Items.query.all():

item.name

8. backref Attribute

class User(db.Model):  
 id = db.Column(db.Integer(), primary\_key=True)  
 items = db.relationship('Item', backref='owned\_user', lazy = True)  
  
class Item(db.Model):  
 id = db.Column(db.Integer(), primary\_key = True)  
 owner = db.Column(db.Integer(), db.ForeignKey('user.id'))

Basically, if we want to know about user of item then we can specify “backref” attribute in “User” and access user of item by “owned\_user”.

lazy = True 🡪 It is necessary to grab all objects of Item.

User.query.filter\_by(username=’mridul’)🡪 will return base query

User.query.filter\_by(username=’mridul’).first()🡪 will return object

User.query.filter\_by(username=’mridul’).first().id 🡪 will return object’s attribute