Source Code: AI Generated Report

Generated on: 2026-02-07 16:18:27

```
from fastapi import FastAPI, HTTPException

from pydantic import BaseModel

from fastapi.testclient import TestClient

from sqlalchemy import create_engine, Column, String, Float

from sqlalchemy.ext.declarative import declarative_base

from sqlalchemy.orm import sessionmaker


# Database setup

DATABASE_URL = "sqlite:///:memory:"

engine = create_engine(DATABASE_URL)

SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

Base = declarative_base()


# Models

class Product(BaseModel):

    name: str

    price: float


class BillRequest(BaseModel):

    products: list[Product]


# FastAPI app

app = FastAPI()
```

```
# Create database tables
Base.metadata.create_all(bind=engine)


@app.post("/calculate-bill")
def calculate_bill(bill_request: BillRequest):
    if not bill_request.products:
        raise HTTPException(status_code=400, detail="Product list cannot be empty.")

    total_price = sum(product.price for product in bill_request.products)
    total_gst = total_price * 0.12
    final_bill = total_price + total_gst

    return {
        "total_price": total_price,
        "total_gst": total_gst,
        "final_bill": final_bill
    }

# Testing the FastAPI app
client = TestClient(app)

def test_calculate_bill():
    response = client.post("/calculate-bill", json={
        "products": [
            {"name": "Product 1", "price": 100.0},
```

```
            {"name": "Product 2", "price": 200.0}
        ]
    })
    print(response.json())
    assert response.status_code == 200
    assert response.json() == {
        "total_price": 300.0,
        "total_gst": 36.0,
        "final_bill": 336.0
    }

def test_calculate_bill_empty():
    response = client.post("/calculate-bill", json={"products": []})
    print(response.json())
    assert response.status_code == 400
    assert response.json() == {"detail": "Product list cannot be empty."}

# Run tests
test_calculate_bill()
test_calculate_bill_empty()
```