Source Code: AI Generated Report

Generated on: 2026-02-07 16:36:44

```
from fastapi import FastAPI, HTTPException

from pydantic import BaseModel

from sqlalchemy import create_engine, Column, String, Float

from sqlalchemy.orm import DeclarativeBase, Mapped, mapped_column, sessionmaker

import pandas as pd

from fastapi.testclient import TestClient


# Database setup

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

engine = create_engine(DATABASE_URL)

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


class Base(DeclarativeBase):

    pass


class ProductModel(Base):

    __tablename__ = "products"

    id: Mapped[int] = mapped_column(primary_key=True)

    name: Mapped[str] = mapped_column(String)

    price: Mapped[float] = mapped_column(Float)


Base.metadata.create_all(bind=engine)
```

```
# FastAPI application

app = FastAPI()


# Pydantic models

class Product(BaseModel):

    name: str

    price: float


class BillResponse(BaseModel):

    total_price: float

    gst_amount: float

    final_bill: float


@app.post("/calculate-bill", response_model=BillResponse)

def calculate_bill(products: list[Product]):

    if not products:

        raise HTTPException(status_code=400, detail="Product list cannot be empty.")


    prices = [product.price for product in products]

    total_price = pd.Series(prices).sum()

    gst_amount = total_price * 0.12

    final_bill = total_price + gst_amount


                return    BillResponse(total_price=total_price,    gst_amount=gst_amount,

final_bill=final_bill)
```

```
# Testing the application

client = TestClient(app)


def test_calculate_bill():

    response = client.post("/calculate-bill", json={

        "products": [

            {"name": "Product A", "price": 100.0},

            {"name": "Product B", "price": 200.0}

        ]

    })

    print(response.json())


test_calculate_bill()
```