💳 Payment Gateway Fullstack

🎯 Objective:

This challenge is provided as a reference to guide you. You are not required to follow it exactly — feel free to go beyond it or implement the solution using any other technology stack you're comfortable with. The goal is to effectively use **GitHub Copilot** 🤖 to build a complete, functional solution.

📋 Guidelines

- 🧑‍💻 **Tech Stack Freedom**: Use any programming language or framework — JavaScript, Python, Java, C#, etc.
- ⚡ **Use Copilot's Power**: Utilize autocomplete suggestions, Copilot Chat, and agent mode.
- 💡 **Best Practices with Copilot**:
    - 🧠 *Provide Context*
    - 🧾 *Use Meaningful Names*
    - 🧱 *Start Small and Iterate*
    - ✅ *Validate and Learn*

❓ Problem Statement:

Build the **APIs** and **UI screens** that:

- Capture 📝 payment information
- Store in 💾 DB
- Display 📊 payment list on screen

💰 Payment Information to be Collected:

- **From Account** ➡️

    - 🏦 Account from which payment is made
    - 🆔 Unique identifier: Account Name or Number
    - 📁 Maintain list in DB
- **To Account** ➡️
    - 👥 Payee accounts (e.g., Mobile/Internet providers, Credit Cards)
    - 🧾 Use Payee Account or Name
    - 📁 Maintain in DB
- 📅 **Payment Date** ➡️
    - Date of scheduled payment (today/future)
    - Format: DD/MM/YYYY
    - ❌ Reject past dates
- 💸 **Payment Amount** ➡️
    - In Rupees ₹
    - Show ₹ symbol in listings
- 💲 **Fee Amount** ➡️
    - Auto-calculated based on payment
    - Show ₹ symbol
    - Based on below fee table:

💰 **Min Amount**	💰 **Max Amount**	🧮 **Fee (₹)**
0	99	10
100	999	25
1000	9999	50
10,000	99,999	100
> 100,000	-	500

- 📝 **Memo** ➡️
  - Optional field
  - 💬 Comment (max 100 chars)

🧪 API

- GET /api/payment → 📋 List all payments
- GET /api/{payment-id}/payment → 📄 Payment detail
- POST /api/payment → ➕ Create payment
- PUT /api/payment → ✏️ Update payment
- DELETE /api/payment → ❌ Delete payment

🖥️ UI Requirements

- Build landing screen for payment info (all fields mandatory except memo)
- Show review screen post submission with ✏️ Edit and ✅ Continue options
  - Edit → takes user back to form
  - Continue → shows confirmation screen with 🆔 Transaction ID
- Responsive UI 🌐
- ✅ Code coverage ≥ 80%

⚙️ Technology Stack

- **Frontend (UI)** → Node.js / Angular / React.js
- **API** → Java / Python / C#
- **Database** → MySQL / PostgreSQL

🧱 Table Structure

Account

- account_id, account_number, account_name, account_balance, account_status, updated_datetime

Payee

- payee_id, payee_number, payee_name, amount_due, due_date, updated_datetime

Fee

- fee_id, fee_amount, amount_min, amount_max, updated_datetime

Payment

- payment_id, account_id, payee_id, fee_id, updated_datetime

📌 Key Points

- UI/UX: Mock Swagger UI 🧪
- API with dropdowns (accounts list) – Mock only 🔽

🧩 Microservice with Observability & Tracing

Use Microservices Architecture divides an application into small, independent services that communicate via APIs (e.g., HTTP/REST). Each service is responsible for a specific function, has its own database, and can be deployed and scaled independently. This approach enables fault isolation, flexibility, and faster development cycles. It's commonly managed using tools like **Docker**, **Kubernetes**, and monitored with **Prometheus** and **Jaeger** for observability.

✅ Features:

- Health check endpoint 💓
- Structured logging 📜
- Basic metrics 📈
- Bonus: Include k6 or Locust performance scripts 🧪

🧪 **Testing (JUnit, Jest, Playwright)**

- ✅ Unit & integration tests
- 🧠 Copilot-generated tests
- 🧪 Visual regression tests
- 🧪 Endpoint integration tests

🔐 **DevSecOps POC**

Security, CI/CD, Maintenance

🔧 Tools: GitHub Actions + Trivy/Snyk + Dependabot

✅ Features:

- Docker image scan 🐳
- SAST: CodeQL/SonarQube 🛡️
- Dependency scanning 📦
- Bonus: Copilot auto-fixes 🛠️

🏁 **Final Output (Generated via Copilot 🤖)**

- 📄 Requirement Document
- ✅ User Stories with acceptance criteria
- 🗃️ DB Scripts
- 💻 Working Code
- 🧪 Test Plan & Unit Tests
- 🚀 Deployment Script using GitHub Actions
- 📘 Final Project Document