FastAPI Coding Test - 90 Minutes

Overview:

You are building a mini backend API for a restaurant to manage its daily menu and customer orders.

This test uses FastAPI and PostgreSQL and is scoped for completion within 90 minutes.

Requirements:

Tech Stack:

- Python 3.9+
- FastAPI
- PostgreSQL (local or Docker)
- SQLAlchemy (sync or async)
- Pydantic

Models:

1. MenuItem
   - id (int, primary key)
   - name (str)
   - price (float)
   - is_available (bool)

2. Order
   - id (int, primary key)
   - customer_name (str)
   - items (many-to-many with MenuItem)

- created_at (datetime, default = now)

Endpoints to Build:

Route	Method	Functionality
/menu/	POST	Add a new menu item
/menu/available/	GET	Return all available menu items
/order/	POST	Place an order with item IDs
/orders/today/	GET	Return all orders created today

Business Logic:

- When placing an order:
  - Validate all menu item IDs exist and are marked is_available=True.
  - Calculate the total price.
  - Save order with timestamp.
  - Append a log to a file orders_log.txt in the format:
    Timestamp | Customer Name | Ordered Item IDs | Total Price

Technical Requirements:

- Use PostgreSQL via SQLAlchemy (sync or async).
- Use Pydantic models for request and response validation.
- Inject DB session using FastAPI's Depends().
- Organize your code using files like main.py, models.py, schemas.py.

Bonus (Optional if time allows):

- Add a middleware that logs request path and execution time to console.
- Add OpenAPI schema examples using example= in Pydantic models.

Time Limit:

Maximum allowed time: 90 minutes

Deliverables:

- Source code in a GitHub repo or zipped folder.
- Include:
  - main.py or equivalent
  - models.py, schemas.py
  - requirements.txt
  - .env.example with DB settings
  - README.md with setup instructions