AI LAB EXAM -3

NAME: NALLALA SIRI

HT.NO: 2403A52037

BATCH NO: AIB03

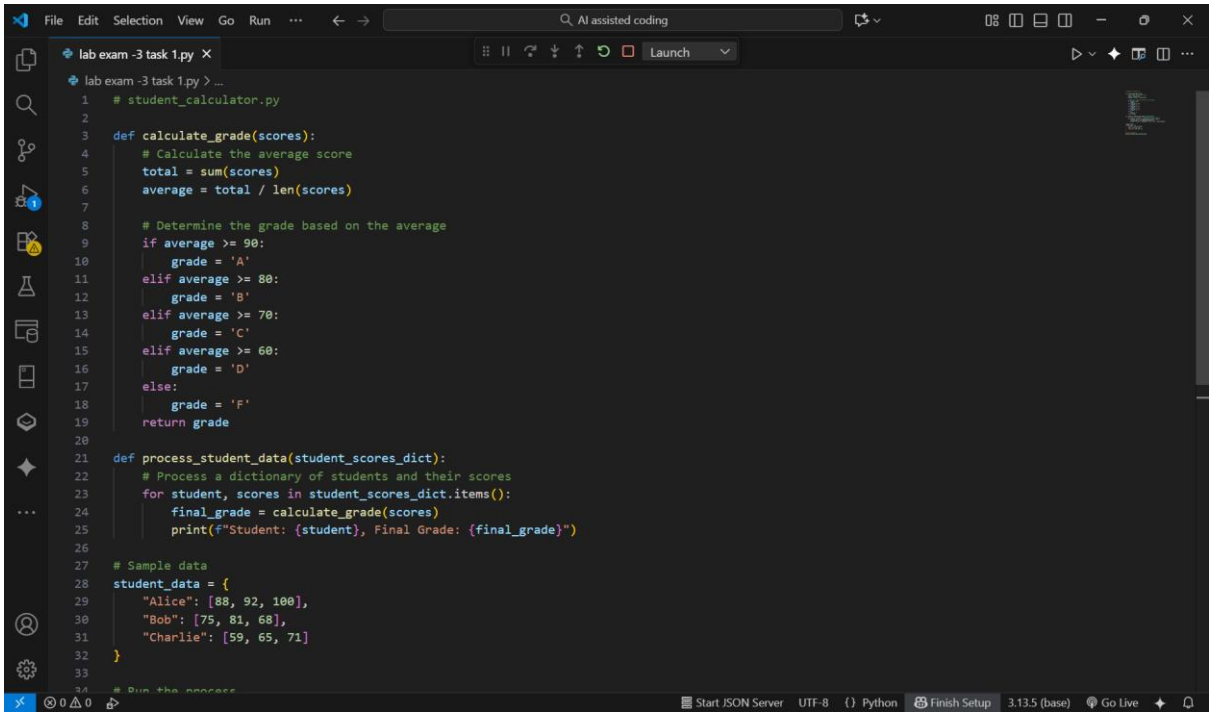
SET E9

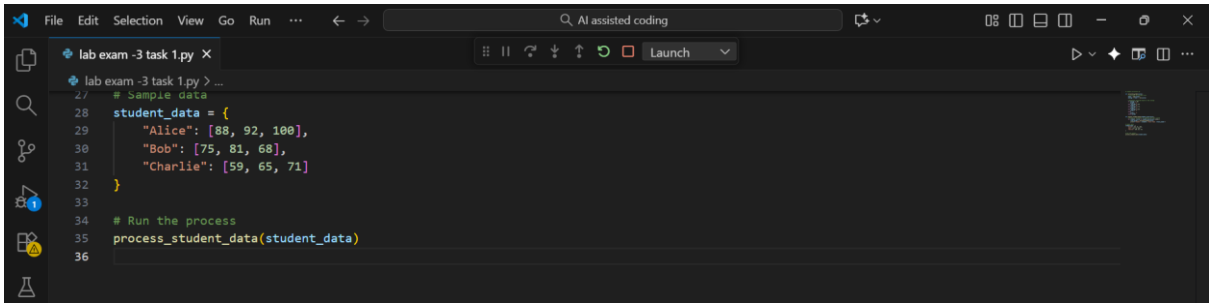
TASK 1:

PROMPT:

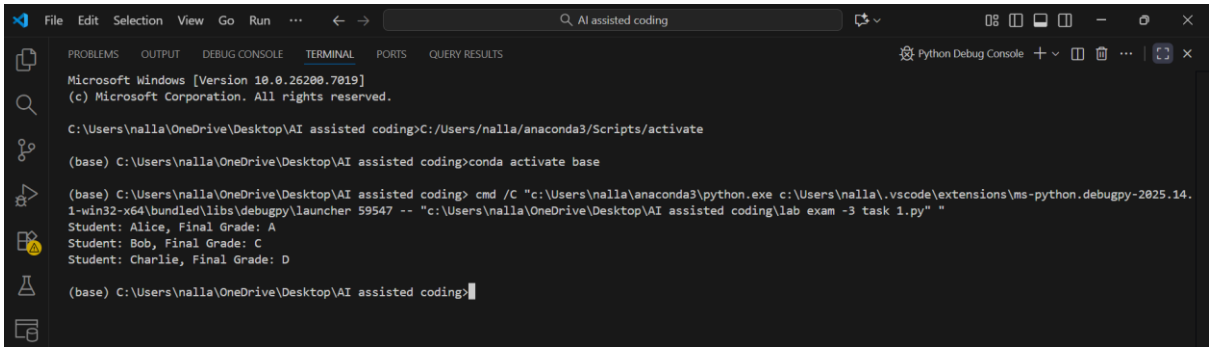
Use AI-assisted code refactoring tools to improve a simple student score calculator by cleaning and optimizing its code for better readability and efficiency.

CODE:





OUTPUT:



OBSERVATION:

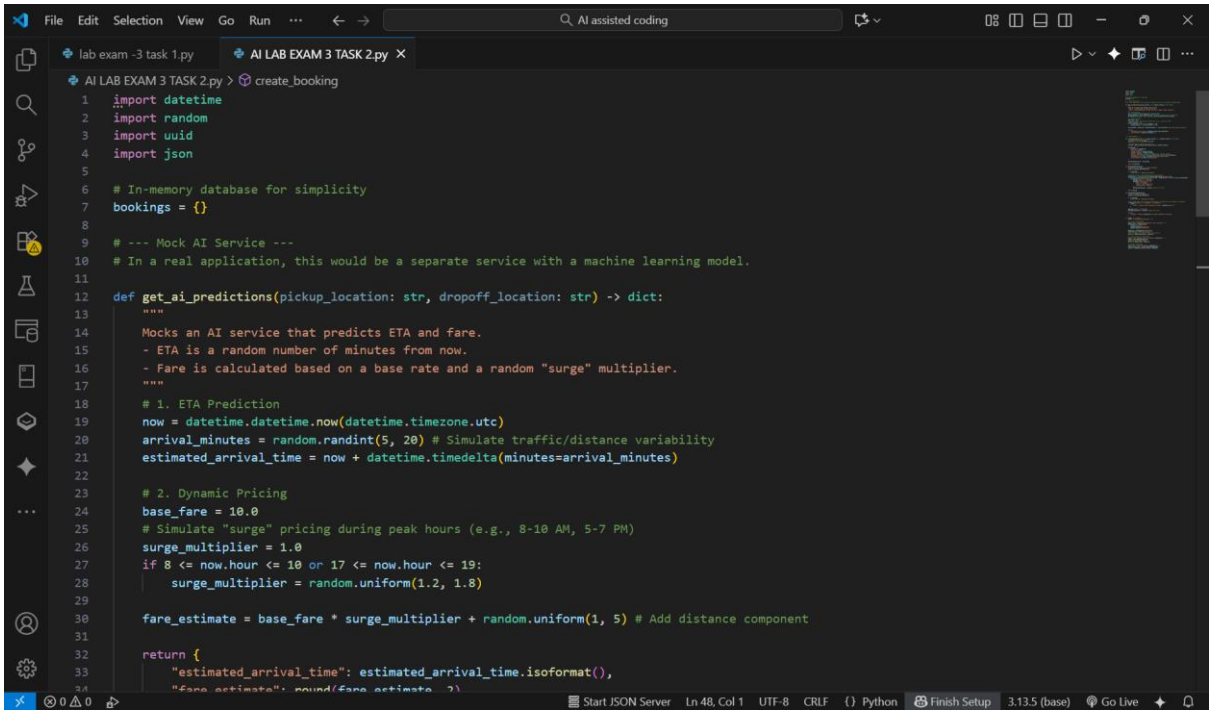
The given code effectively calculates the average score for each student and assigns a letter grade based on that average. It is straightforward and easy to understand, with clear separation of concerns between calculating grades and processing student data. However, the code could be enhanced to handle edge cases, such as empty score lists, for improved robustness.

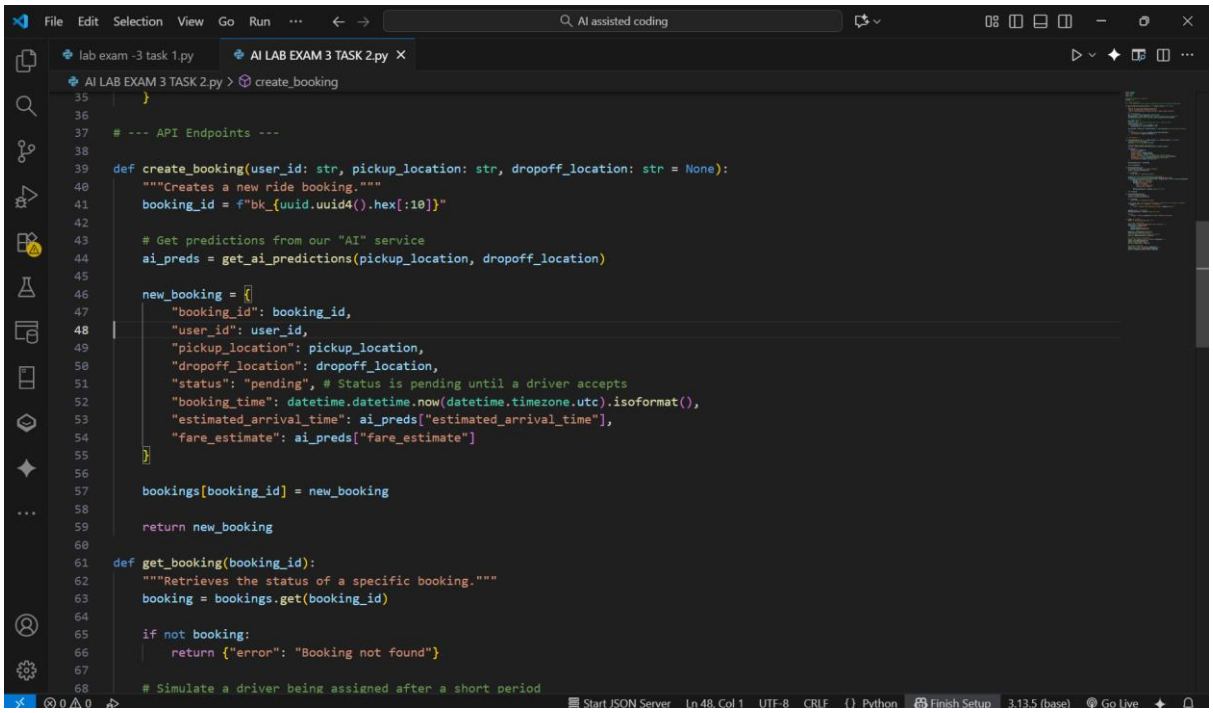
TASK 2:

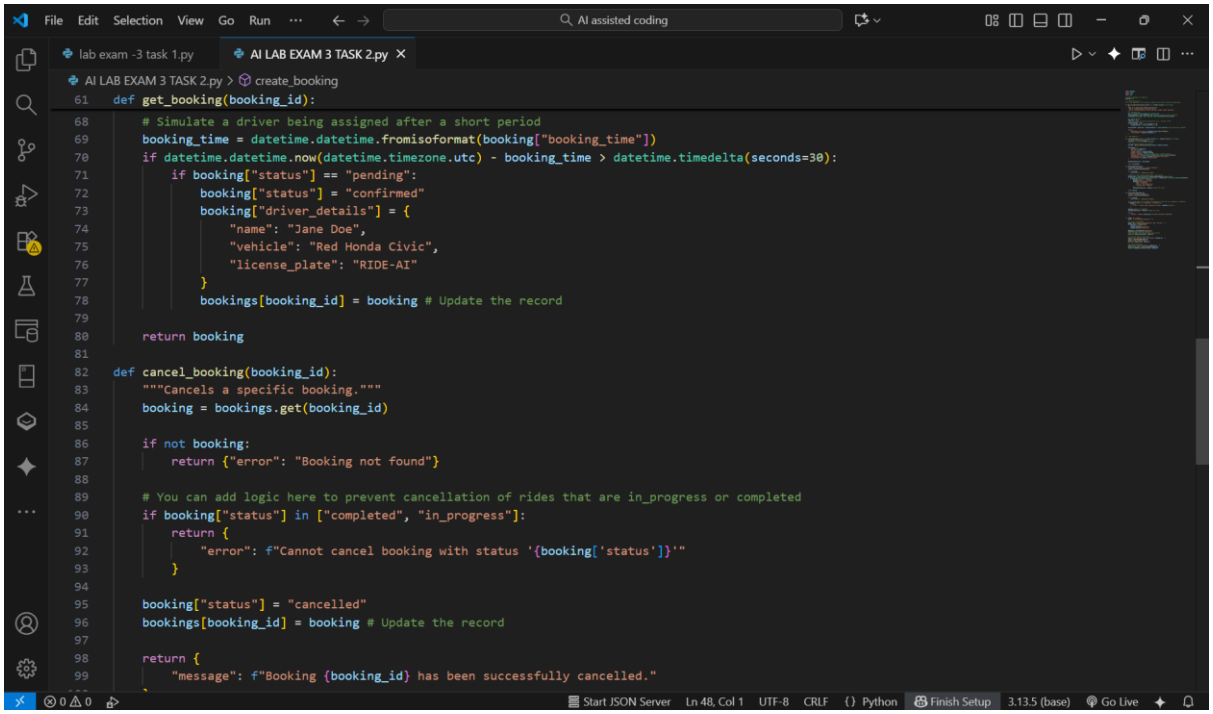
PROMPT:

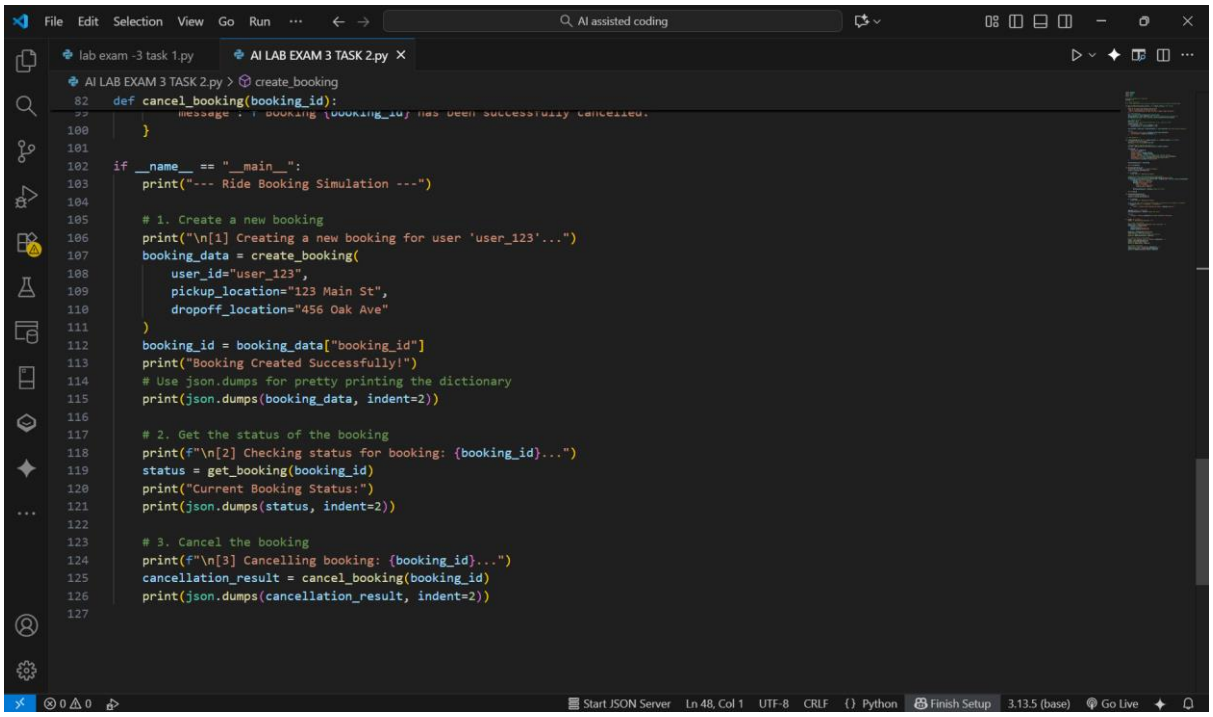
Design a simple backend API for a transportation company to manage and retrieve ride booking status using AI-assisted tools.

CODE:

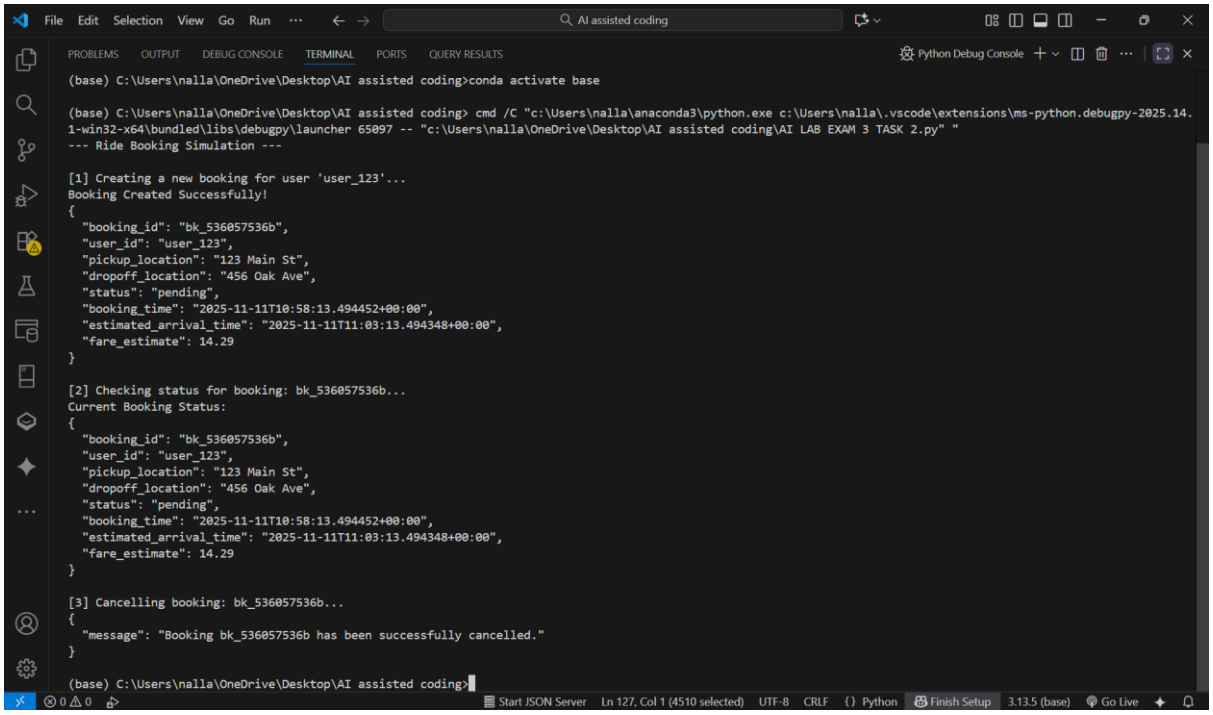

File Edit Selection View Go Run ···
AI assisted coding
lab exam -3 task 1.py
AI LAB EXAM 3 TASK 2.py
AI LAB EXAM 3 TASK 2.py > create_booking
import datetime
import random
import uuid
import json
# In-memory database for simplicity
bookings = {}
# --- Mock AI Service ---
# In a real application, this would be a separate service with a machine learning model.
def get_ai_predictions(pickup_location: str, dropoff_location: str) -> dict:
"""
Mocks an AI service that predicts ETA and fare.
- ETA is a random number of minutes from now.
- Fare is calculated based on a base rate and a random "surge" multiplier.
"""
# 1. ETA Prediction
now = datetime.datetime.now(datetime.timezone.utc)
arrival_minutes = random.randint(5, 20) # Simulate traffic/distance variability
estimated_arrival_time = now + datetime.timedelta(minutes=arrival_minutes)
# 2. Dynamic Pricing
base_fare = 10.0
# Simulate "surge" pricing during peak hours (e.g., 8-10 AM, 5-7 PM)
surge_multiplier = 1.0
if 8 <= now.hour <= 10 or 17 <= now.hour <= 19:
surge_multiplier = random.uniform(1.2, 1.8)
fare_estimate = base_fare * surge_multiplier + random.uniform(1, 5) # Add distance component
return {
"estimated_arrival_time": estimated_arrival_time.isoformat(),
Start JSON Server Ln 48, Col 1 UTF-8 CRLF {} Python Finish Setup 3.13.5 (base) Go Live


File Edit Selection View Go Run ···
AI assisted coding
lab exam -3 task 1.py
AI LAB EXAM 3 TASK 2.py
AI LAB EXAM 3 TASK 2.py > create_booking
}
# --- API Endpoints ---
def create_booking(user_id: str, pickup_location: str, dropoff_location: str = None):
"""Creates a new ride booking."""
booking_id = f"bk_{uuid.uuid4().hex[:10]}"
# Get predictions from our "AI" service
ai_preds = get_ai_predictions(pickup_location, dropoff_location)
new_booking = {
"booking_id": booking_id,
"user_id": user_id,
"pickup_location": pickup_location,
"dropoff_location": dropoff_location,
"status": "pending", # Status is pending until a driver accepts
"booking_time": datetime.datetime.now(datetime.timezone.utc).isoformat(),
"estimated_arrival_time": ai_preds["estimated_arrival_time"],
"fare_estimate": ai_preds["fare_estimate"]
}
bookings[booking_id] = new_booking
return new_booking
def get_booking(booking_id):
"""Retrieves the status of a specific booking."""
booking = bookings.get(booking_id)
if not booking:
return {"error": "Booking not found"}
Simulate a driver being assigned after a short period
Start JSON Server Ln 48, Col 1 UTF-8 CRLF {} Python Finish Setup 3.13.5 (base) Go Live


File Edit Selection View Go Run
AI assisted coding
lab exam -3 task 1.py
AI LAB EXAM 3 TASK 2.py
AI LAB EXAM 3 TASK 2.py > create_booking
def get_booking(booking_id):
# Simulate a driver being assigned after a short period
booking_time = datetime.datetime.fromisoformat(booking["booking_time"])
if datetime.datetime.now(datetime.timezone.utc) - booking_time > datetime.timedelta(seconds=30):
if booking["status"] == "pending":
booking["status"] = "confirmed"
booking["driver_details"] = {
"name": "Jane Doe",
"vehicle": "Red Honda Civic",
"license_plate": "RIDE-AI"
}
bookings[booking_id] = booking # Update the record
return booking
def cancel_booking(booking_id):
"""Cancels a specific booking."""
booking = bookings.get(booking_id)
if not booking:
return {"error": "Booking not found"}
# You can add logic here to prevent cancellation of rides that are in_progress or completed
if booking["status"] in ["completed", "in_progress"]:
return {
"error": f"Cannot cancel booking with status '{booking['status']}'"
}
booking["status"] = "cancelled"
bookings[booking_id] = booking # Update the record
return {
"message": f"Booking {booking_id} has been successfully cancelled."
Start JSON Server Ln 48, Col 1 UTF-8 CRLF Python Finish Setup 3.13.5 (base) Go Live


File Edit Selection View Go Run
AI assisted coding
lab exam -3 task 1.py
AI LAB EXAM 3 TASK 2.py
AI LAB EXAM 3 TASK 2.py > create_booking
def cancel_booking(booking_id):
}
if __name__ == "__main__":
print("--- Ride Booking Simulation ---")
# 1. Create a new booking
print("\n[1] Creating a new booking for user 'user_123'...")
booking_data = create_booking(
user_id="user_123",
pickup_location="123 Main St",
dropoff_location="456 Oak Ave"
)
booking_id = booking_data["booking_id"]
print("Booking Created Successfully!")
# Use json.dumps for pretty printing the dictionary
print(json.dumps(booking_data, indent=2))
# 2. Get the status of the booking
print(f"\n[2] Checking status for booking: {booking_id}...")
status = get_booking(booking_id)
print("Current Booking Status:")
print(json.dumps(status, indent=2))
# 3. Cancel the booking
print(f"\n[3] Cancelling booking: {booking_id}...")
cancellation_result = cancel_booking(booking_id)
print(json.dumps(cancellation_result, indent=2))
Start JSON Server Ln 48, Col 1 UTF-8 CRLF Python Finish Setup 3.13.5 (base) Go Live

OUTPUT:



OBSERVATION:

The code simulates a simple ride booking system with AI-predicted ETAs and dynamic fares. It handles creating, checking, and cancelling bookings with clear, concise output in the terminal. This basic flow is easy to follow and demonstrates essential backend booking operations effectively.