

# AI LAB EXAM -3

NAME: NALLALA SIRI

HT.NO: 2403A52037

BATCH NO: AIB03

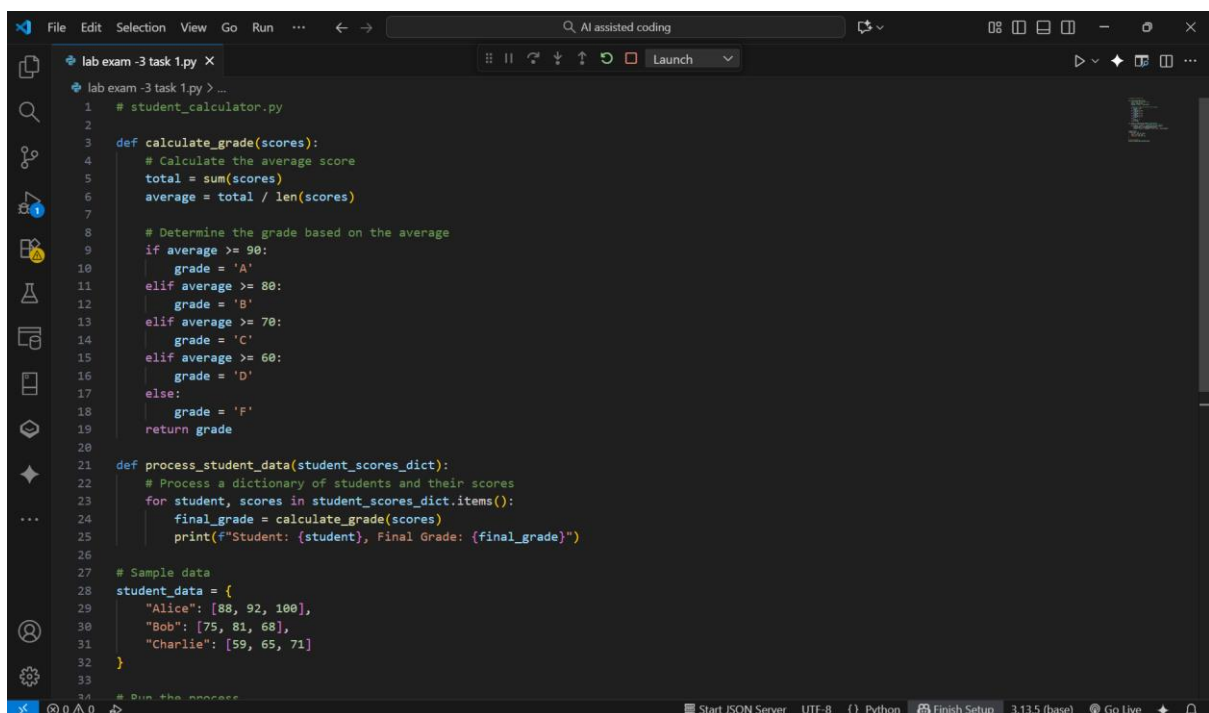
SET E9

TASK 1:

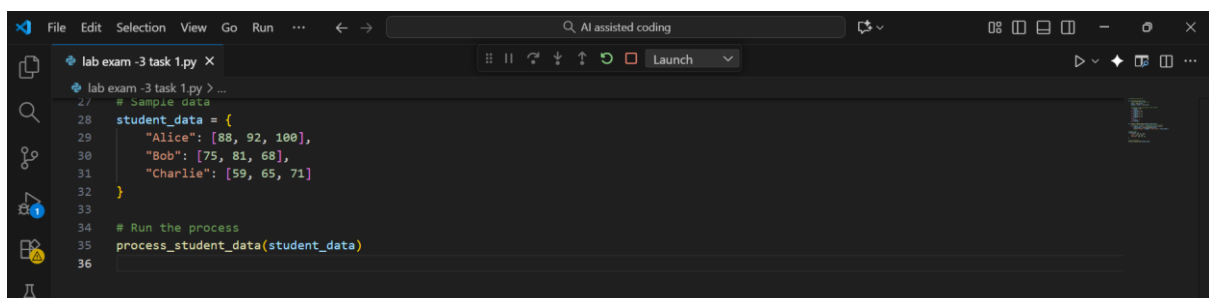
PROMPT:

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:

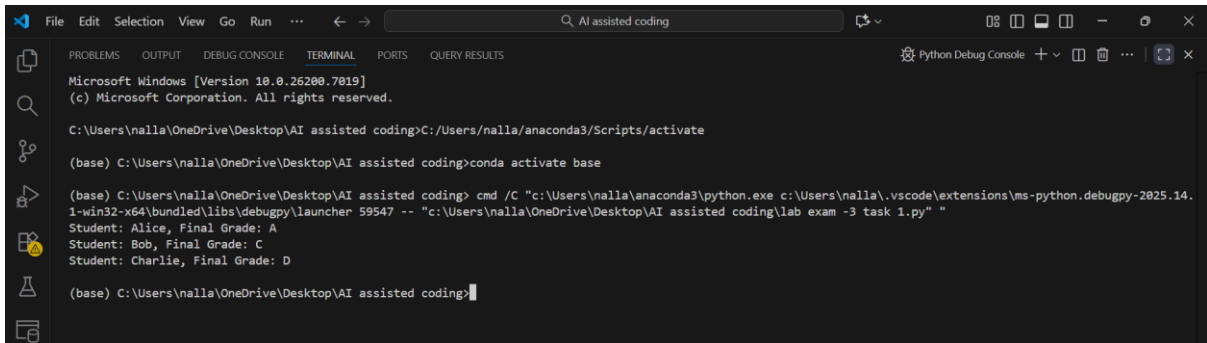


```
1 # student_calculator.py
2
3 def calculate_grade(scores):
4     # Calculate the average score
5     total = sum(scores)
6     average = total / len(scores)
7
8     # Determine the grade based on the average
9     if average >= 90:
10         grade = 'A'
11     elif average >= 80:
12         grade = 'B'
13     elif average >= 70:
14         grade = 'C'
15     elif average >= 60:
16         grade = 'D'
17     else:
18         grade = 'F'
19     return grade
20
21 def process_student_data(student_scores_dict):
22     # Process a dictionary of students and their scores
23     for student, scores in student_scores_dict.items():
24         final_grade = calculate_grade(scores)
25         print(f"Student: {student}, Final Grade: {final_grade}")
26
27 # Sample data
28 student_data = {
29     "Alice": [88, 92, 100],
30     "Bob": [75, 81, 68],
31     "Charlie": [59, 65, 71]
32 }
33
34 # Run the process
```



```
27 # Sample data
28 student_data = {
29     "Alice": [88, 92, 100],
30     "Bob": [75, 81, 68],
31     "Charlie": [59, 65, 71]
32 }
33
34 # Run the process
35 process_student_data(student_data)
36
```

## OUTPUT:



```
Microsoft Windows [Version 10.0.26200.7019]
(c) Microsoft Corporation. All rights reserved.

C:\Users\nalla\OneDrive\Desktop\AI assisted coding>C:\Users\nalla\anaconda3\Scripts\activate

(base) C:\Users\nalla\OneDrive\Desktop\AI assisted coding>conda activate base

(base) C:\Users\nalla\OneDrive\Desktop\AI assisted coding> cmd /C "c:\Users\nalla\anaconda3\python.exe c:\Users\nalla\.vscode\extensions\ms-python.debugpy-2025.14.1-win32-x64\bundled\libs\debugpy\launcher 59547 -- "c:\Users\nalla\OneDrive\Desktop\AI assisted coding\lab exam -3 task 1.py" "
```

Student: Alice, Final Grade: A  
Student: Bob, Final Grade: C  
Student: Charlie, Final Grade: D

```
(base) C:\Users\nalla\OneDrive\Desktop\AI assisted coding>
```

## 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:

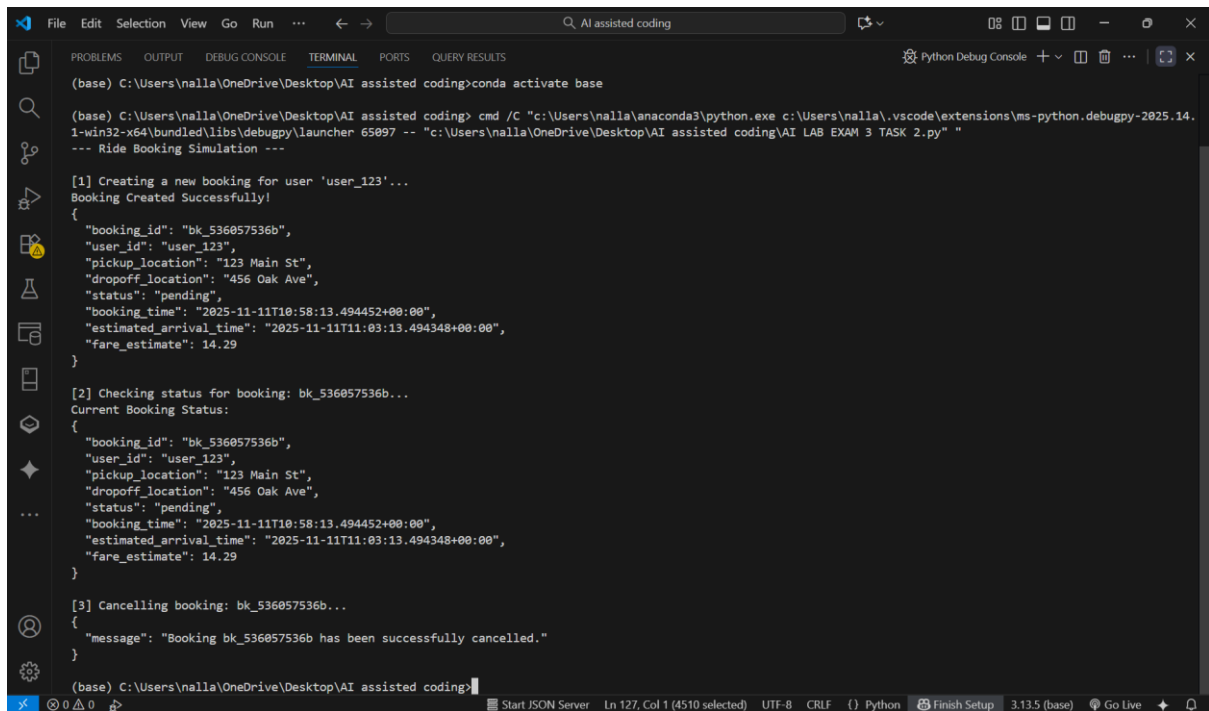
```
File Edit Selection View Go Run ... AI assisted coding
lab exam -3 task 1.py AI LAB EXAM 3 TASK 2.py X
AI LAB EXAM 3 TASK 2.py > create_booking
1 import datetime
2 import random
3 import uuid
4 import json
5
6 # In-memory database for simplicity
7 bookings = {}
8
9 # --- Mock AI Service ---
10 # In a real application, this would be a separate service with a machine learning model.
11
12 def get_ai_predictions(pickup_location: str, dropoff_location: str) -> dict:
13     """
14     Mocks an AI service that predicts ETA and fare.
15     - ETA is a random number of minutes from now.
16     - Fare is calculated based on a base rate and a random "surge" multiplier.
17     """
18     # 1. ETA Prediction
19     now = datetime.datetime.now(datetime.timezone.utc)
20     arrival_minutes = random.randint(5, 20) # Simulate traffic/distance variability
21     estimated_arrival_time = now + datetime.timedelta(minutes=arrival_minutes)
22
23     # 2. Dynamic Pricing
24     base_fare = 10.0
25     # Simulate "surge" pricing during peak hours (e.g., 8-10 AM, 5-7 PM)
26     surge_multiplier = 1.0
27     if 8 <= now.hour <= 10 or 17 <= now.hour <= 19:
28         surge_multiplier = random.uniform(1.2, 1.8)
29
30     fare_estimate = base_fare * surge_multiplier + random.uniform(1, 5) # Add distance component
31
32     return {
33         "estimated_arrival_time": estimated_arrival_time.isoformat(),
34         "fare_estimate": round(fare_estimate, 2)
35     }
```

```
File Edit Selection View Go Run ... AI assisted coding
lab exam -3 task 1.py AI LAB EXAM 3 TASK 2.py X
AI LAB EXAM 3 TASK 2.py > create_booking
35 }
36
37 # --- API Endpoints ---
38
39 def create_booking(user_id: str, pickup_location: str, dropoff_location: str = None):
40     """Creates a new ride booking."""
41     booking_id = f"bk_{uuid.uuid4().hex[:10]}"
42
43     # Get predictions from our "AI" service
44     ai_preds = get_ai_predictions(pickup_location, dropoff_location)
45
46     new_booking = {
47         "booking_id": booking_id,
48         "user_id": user_id,
49         "pickup_location": pickup_location,
50         "dropoff_location": dropoff_location,
51         "status": "pending", # Status is pending until a driver accepts
52         "booking_time": datetime.datetime.now(datetime.timezone.utc).isoformat(),
53         "estimated_arrival_time": ai_preds["estimated_arrival_time"],
54         "fare_estimate": ai_preds["fare_estimate"]
55     }
56
57     bookings[booking_id] = new_booking
58
59     return new_booking
60
61 def get_booking(booking_id):
62     """Retrieves the status of a specific booking."""
63     booking = bookings.get(booking_id)
64
65     if not booking:
66         return {"error": "Booking not found"}
67
68     # Simulate a driver being assigned after a short period
```

```
File Edit Selection View Go Run ... AI assisted coding
lab exam -3 task 1.py AI LAB EXAM 3 TASK 2.py X
AI LAB EXAM 3 TASK 2.py > create_booking
61 def get_booking(booking_id):
62     # Simulate a driver being assigned after a short period
63     booking_time = datetime.datetime.fromisoformat(booking["booking_time"])
64     if datetime.datetime.now(datetime.timezone.utc) - booking_time > datetime.timedelta(seconds=30):
65         if booking["status"] == "pending":
66             booking["status"] = "confirmed"
67             booking["driver_details"] = {
68                 "name": "Jane Doe",
69                 "vehicle": "Red Honda Civic",
70                 "license_plate": "RIDE-AI"
71             }
72             bookings[booking_id] = booking # Update the record
73         return booking
74
75 def cancel_booking(booking_id):
76     """Cancels a specific booking."""
77     booking = bookings.get(booking_id)
78
79     if not booking:
80         return {"error": "Booking not found"}
81
82     # You can add logic here to prevent cancellation of rides that are in_progress or completed
83     if booking["status"] in ["completed", "in_progress"]:
84         return {
85             "error": f"Cannot cancel booking with status '{booking['status']}'"
86         }
87
88     booking["status"] = "cancelled"
89     bookings[booking_id] = booking # Update the record
90
91     return {
92         "message": f"Booking {booking_id} has been successfully cancelled."
93     }
```

```
File Edit Selection View Go Run ... AI assisted coding
lab exam -3 task 1.py AI LAB EXAM 3 TASK 2.py X
AI LAB EXAM 3 TASK 2.py > create_booking
82 def cancel_booking(booking_id):
83     """Cancels a specific booking."""
84     booking = bookings.get(booking_id)
85
86     if not booking:
87         return {"error": "Booking not found"}
88
89     # You can add logic here to prevent cancellation of rides that are in_progress or completed
90     if booking["status"] in ["completed", "in_progress"]:
91         return {
92             "error": f"Cannot cancel booking with status '{booking['status']}'"
93         }
94
95     booking["status"] = "cancelled"
96     bookings[booking_id] = booking # Update the record
97
98     return {
99         "message": f"Booking {booking_id} has been successfully cancelled."
100     }
101
102 if __name__ == "__main__":
103     print("--- Ride Booking Simulation ---")
104
105     # 1. Create a new booking
106     print("\n[1] Creating a new booking for user 'user_123'...")
107     booking_data = create_booking(
108         user_id="user_123",
109         pickup_location="123 Main St",
110         dropoff_location="456 Oak Ave"
111     )
112     booking_id = booking_data["booking_id"]
113     print("Booking Created Successfully!")
114     # Use json.dumps for pretty printing the dictionary
115     print(json.dumps(booking_data, indent=2))
116
117     # 2. Get the status of the booking
118     print(f"\n[2] Checking status for booking: {booking_id}...")
119     status = get_booking(booking_id)
120     print("Current Booking Status:")
121     print(json.dumps(status, indent=2))
122
123     # 3. Cancel the booking
124     print(f"\n[3] Cancelling booking: {booking_id}...")
125     cancellation_result = cancel_booking(booking_id)
126     print(json.dumps(cancellation_result, indent=2))
127
```

OUTPUT:



```
(base) C:\Users\nalla\OneDrive\Desktop\AI assisted coding>conda activate base

(base) C:\Users\nalla\OneDrive\Desktop\AI assisted coding> cmd /C "c:\Users\nalla\anaconda3\python.exe c:\Users\nalla\.vscode\extensions\ms-python.debugpy-2025.14.1-win32-x64\bundled\libs\debugpy\launcher 65097 -- "c:\Users\nalla\OneDrive\Desktop\AI assisted coding\AI LAB EXAM 3 TASK 2.py" "
--- Ride Booking Simulation ---

[1] Creating a new booking for user 'user_123'...
Booking Created Successfully!
{
  "booking_id": "bk_536057536b",
  "user_id": "user_123",
  "pickup_location": "123 Main St",
  "dropoff_location": "456 Oak Ave",
  "status": "pending",
  "booking_time": "2025-11-11T10:58:13.494452+00:00",
  "estimated_arrival_time": "2025-11-11T11:03:13.494348+00:00",
  "fare_estimate": 14.29
}

[2] Checking status for booking: bk_536057536b...
Current Booking Status:
{
  "booking_id": "bk_536057536b",
  "user_id": "user_123",
  "pickup_location": "123 Main St",
  "dropoff_location": "456 Oak Ave",
  "status": "pending",
  "booking_time": "2025-11-11T10:58:13.494452+00:00",
  "estimated_arrival_time": "2025-11-11T11:03:13.494348+00:00",
  "fare_estimate": 14.29
}

[3] Cancelling booking: bk_536057536b...
{
  "message": "Booking bk_536057536b has been successfully cancelled."
}

(base) C:\Users\nalla\OneDrive\Desktop\AI assisted coding>
```

## 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.