**API Testing Report – Restful Booker API**

### **1. Introduction**

The purpose of this project is to perform **end-to-end API testing** on the [Restful Booker API](https://restful-booker.herokuapp.com/?utm_source=chatgpt.com).  
 The testing process covered authentication, booking creation, retrieval, updating, deletion, and negative testing.  
 In addition, **automation scripts** and **global variables** were used to enhance test reusability and efficiency.

### **2. Tools & Approach**

* **Tool Used:** Postman
* **Testing Approach:**
  + CRUD operations (Create, Read, Update, Delete)
  + Authentication using tokens
  + Positive & Negative test scenarios
  + Automation scripts for validation
  + Global variables for reusability (e.g., storing baseUrl, bookingId, Token)

### **3. Test Scenarios & Implementation**

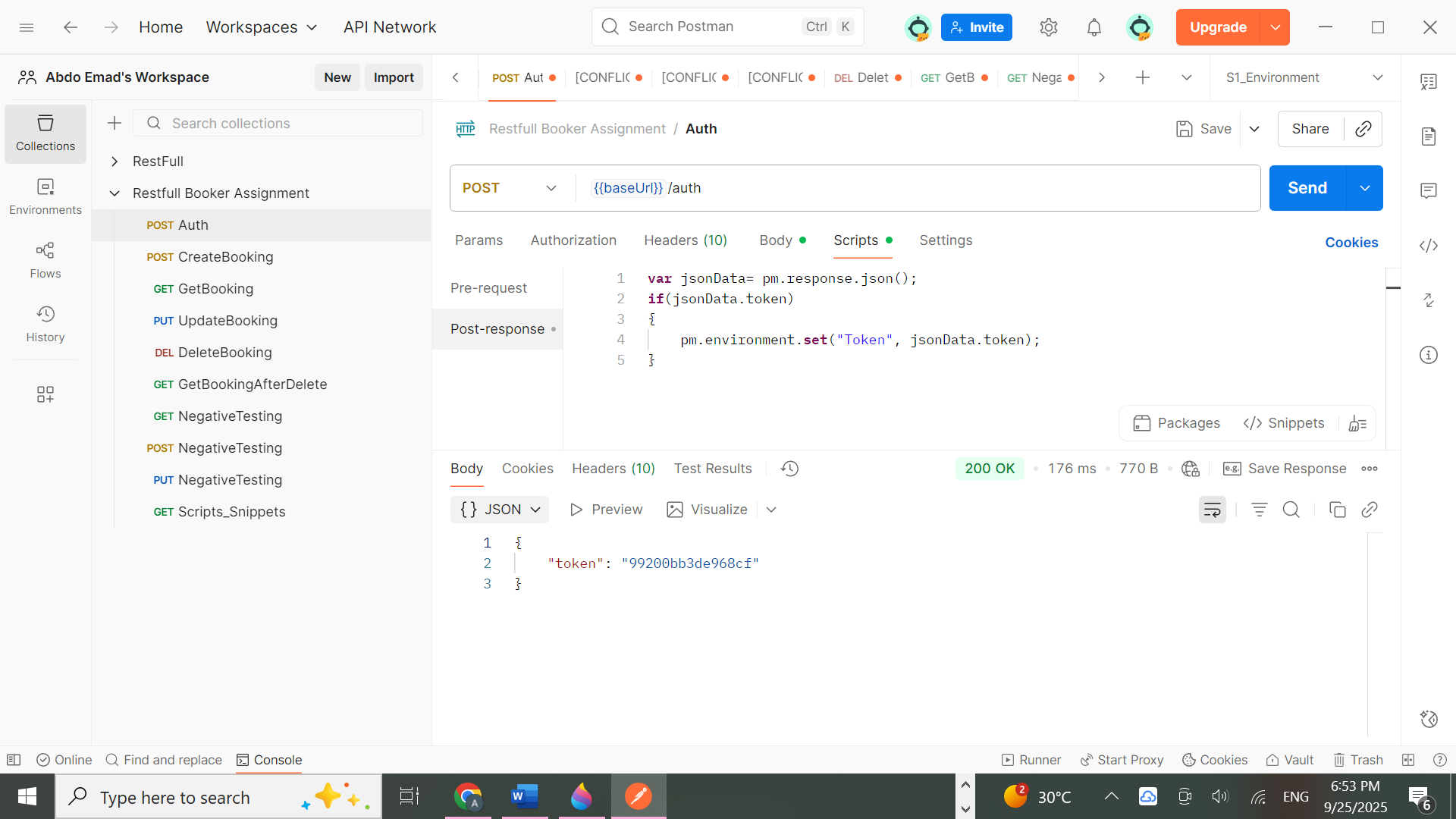
#### **3.1 Authentication (POST /auth)**

Generated token using valid credentials.

Token automatically stored in a **global variable** using a Postman script:

}

var jsonData = pm.response.json();  
if (jsonData.token) {  
 pm.environment.set("Token", jsonData.token);  
}



#### **3.2 Create Booking (POST /booking)**

Created a new booking with all required fields.

Automated validation script:

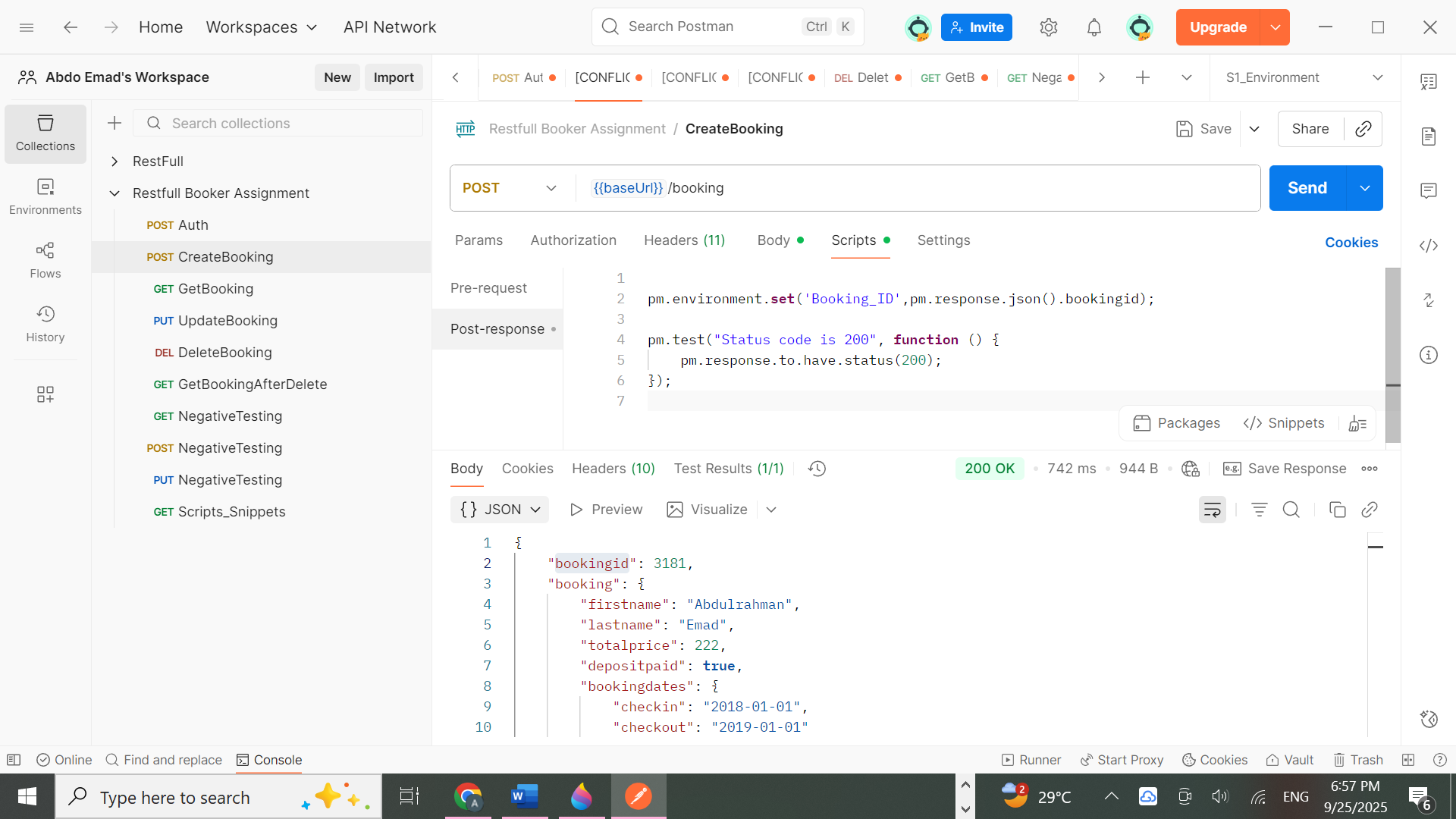
**pm.test("Status code is 200", function () {**

**pm.response.to.have.status(200);**

**});**

**pm.environment.set('Booking\_ID',pm.response.json().bookingid);**

**Booking ID** saved as a global variable for later requests.



#### **3.3 Retrieve Booking (GET /booking/{id})**

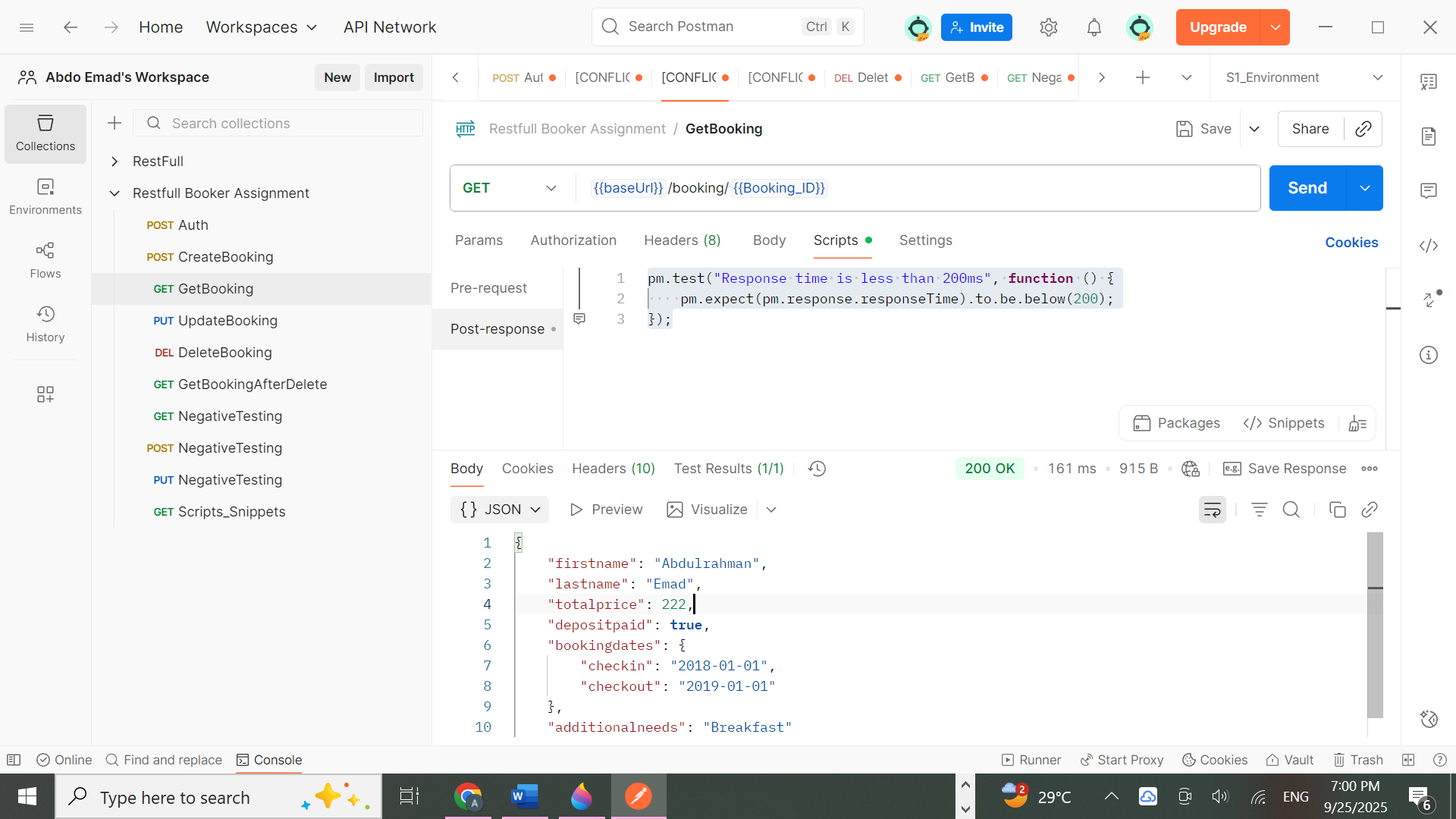
Fetched booking using the stored bookingId.

Validated the response body against expected values.

**pm.test("Response time is less than 200ms", function () {**

**pm.expect(pm.response.responseTime).to.be.below(200);**

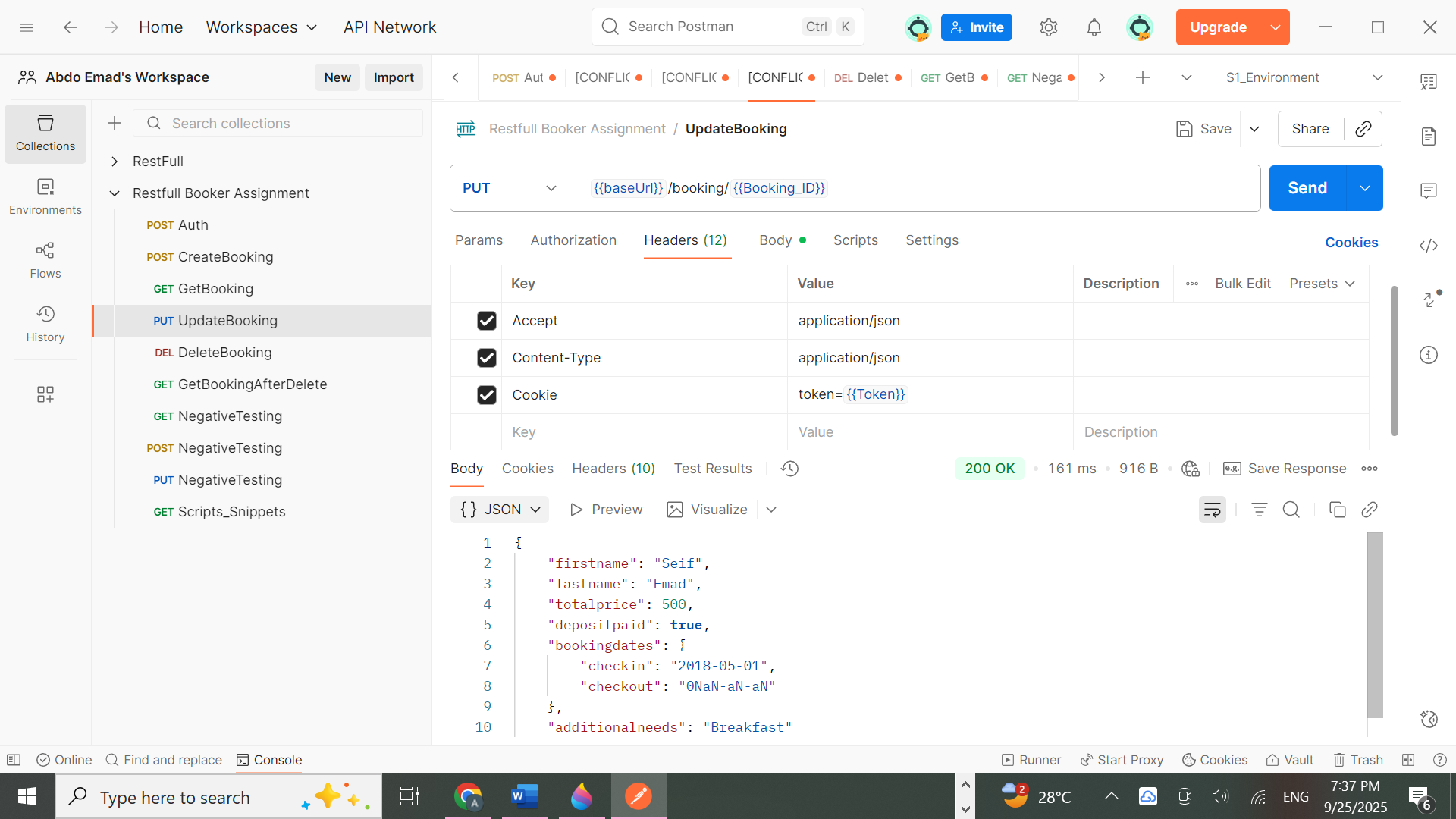
**});**



#### **3.4 Update Booking (PUT /booking/{id})**

Updated booking details using stored Token for authorization.

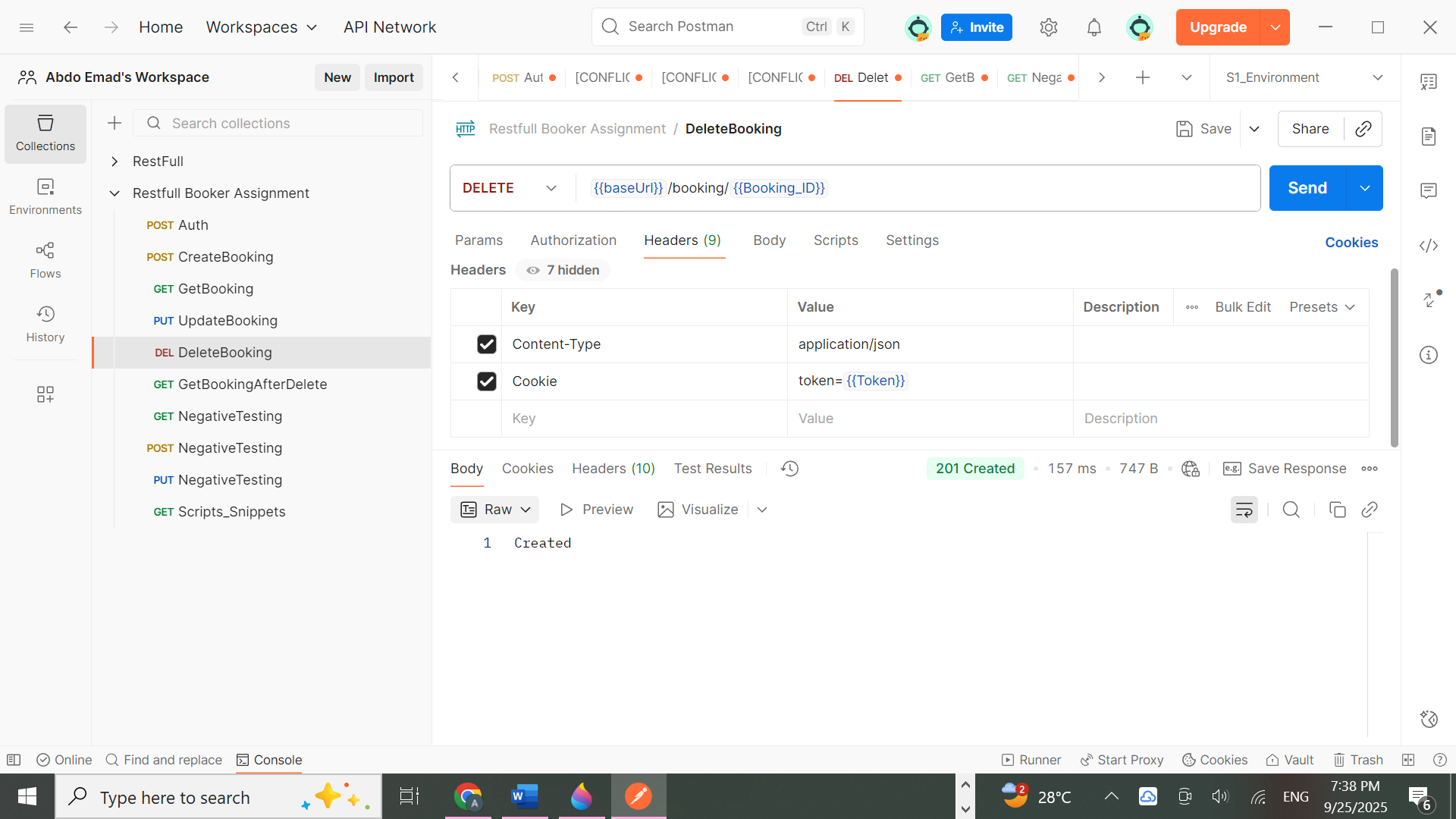
Confirmed that changes were reflected correctly.



#### **3.5 Delete Booking (DELETE /booking/{id})**

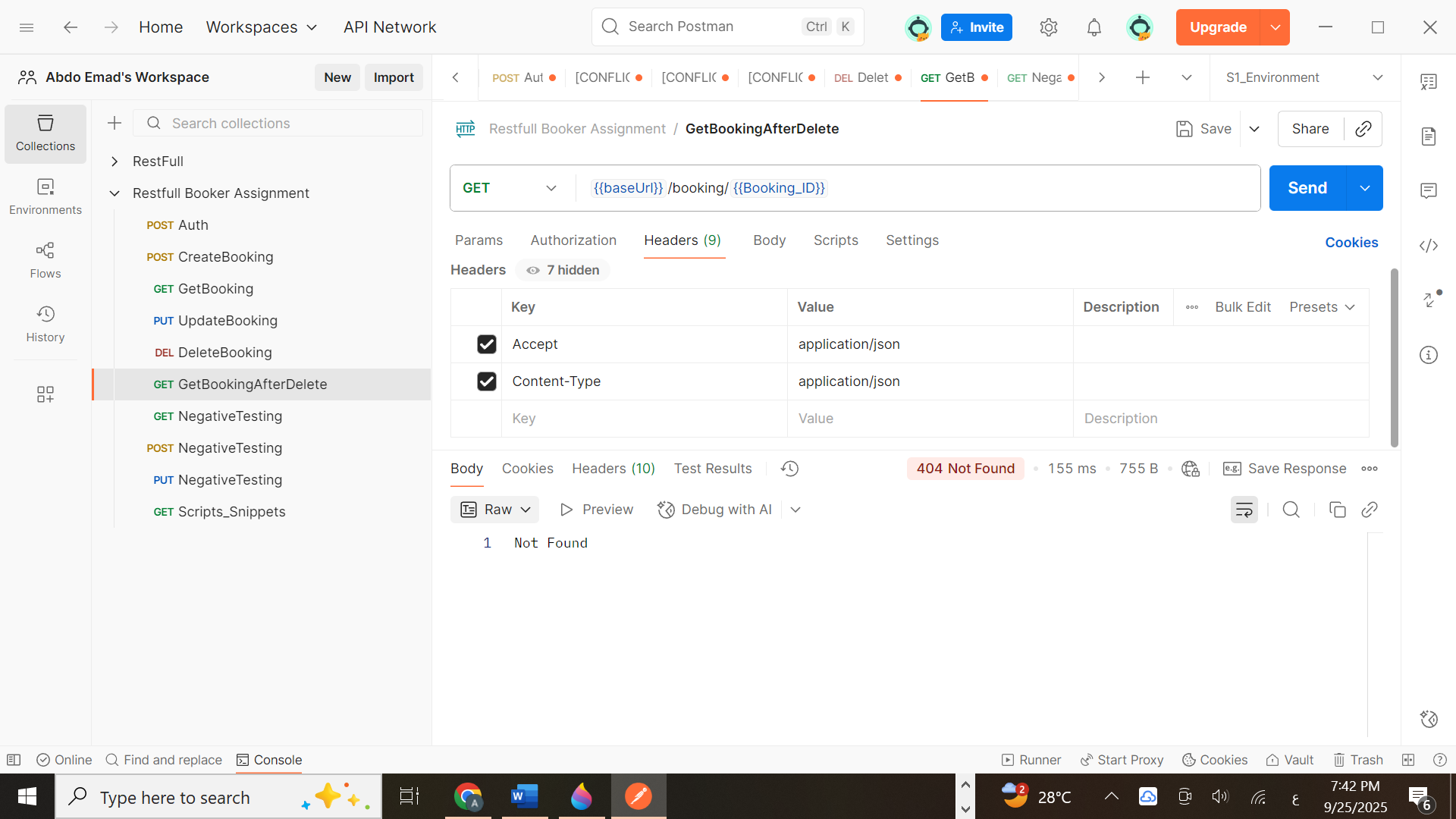
Deleted the booking using authorization token.

Validation: Tried retrieving the booking again and received **404 Not Found**, confirming deletion.



## **3.6 Post-Deletion Validation (GET after DELETE)**

To confirm the deletion, an additional GET request was made using the deleted booking ID.  
 The system correctly returned **404 Not Found**, proving that the booking was no longer accessible.

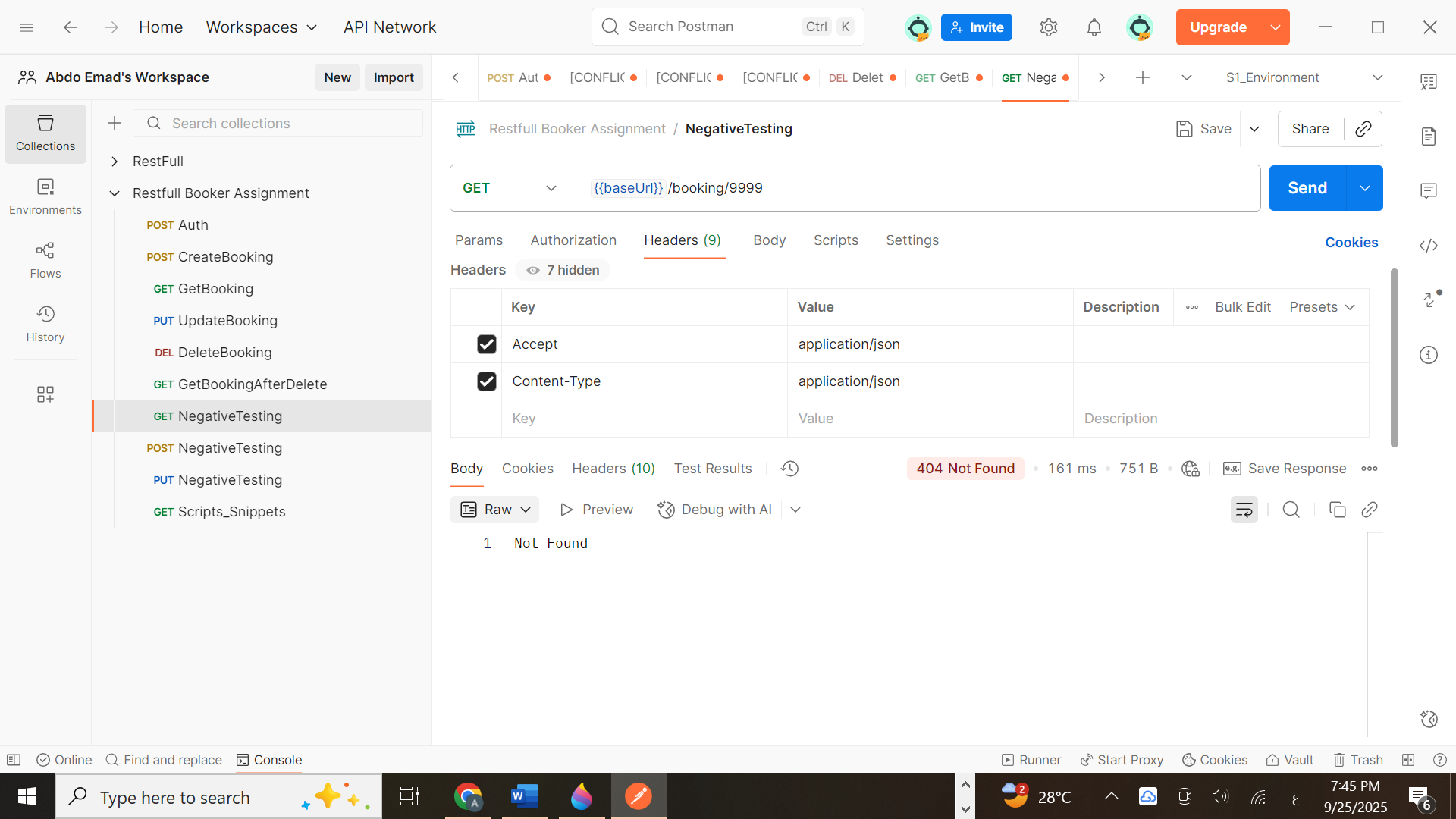


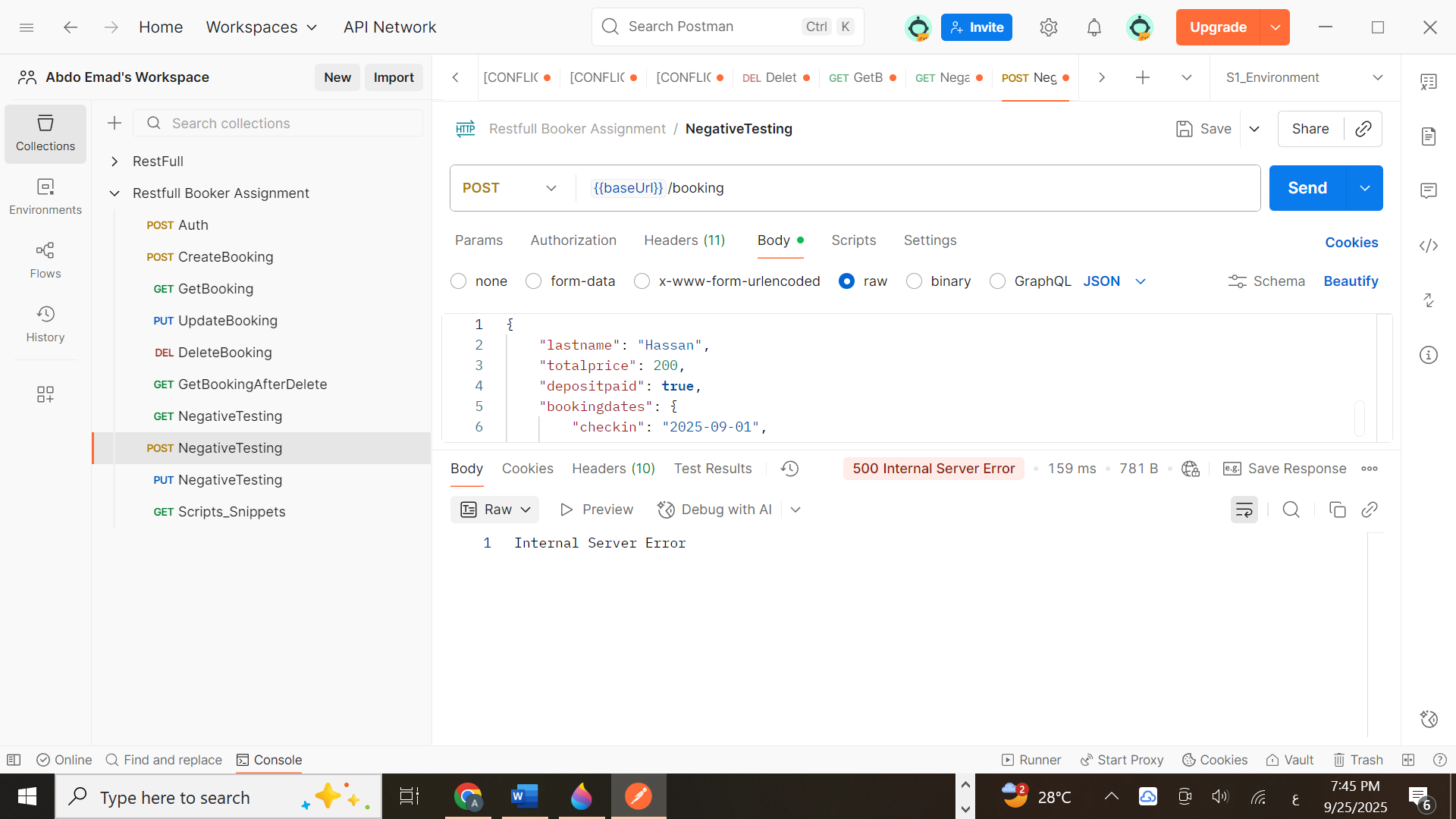
#### **3.7 Negative Testing**

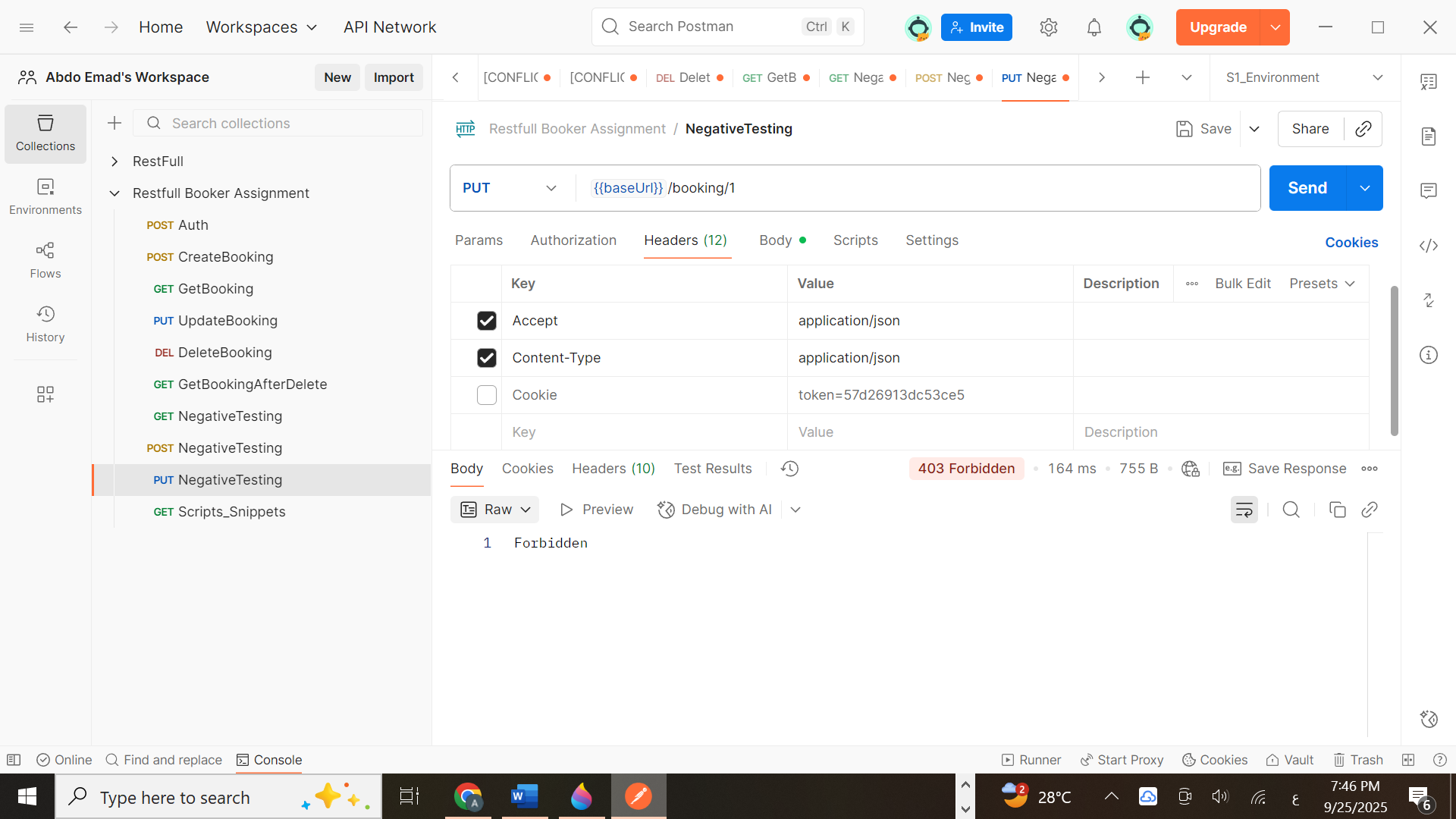
## Several negative scenarios were executed to confirm the robustness of the API.

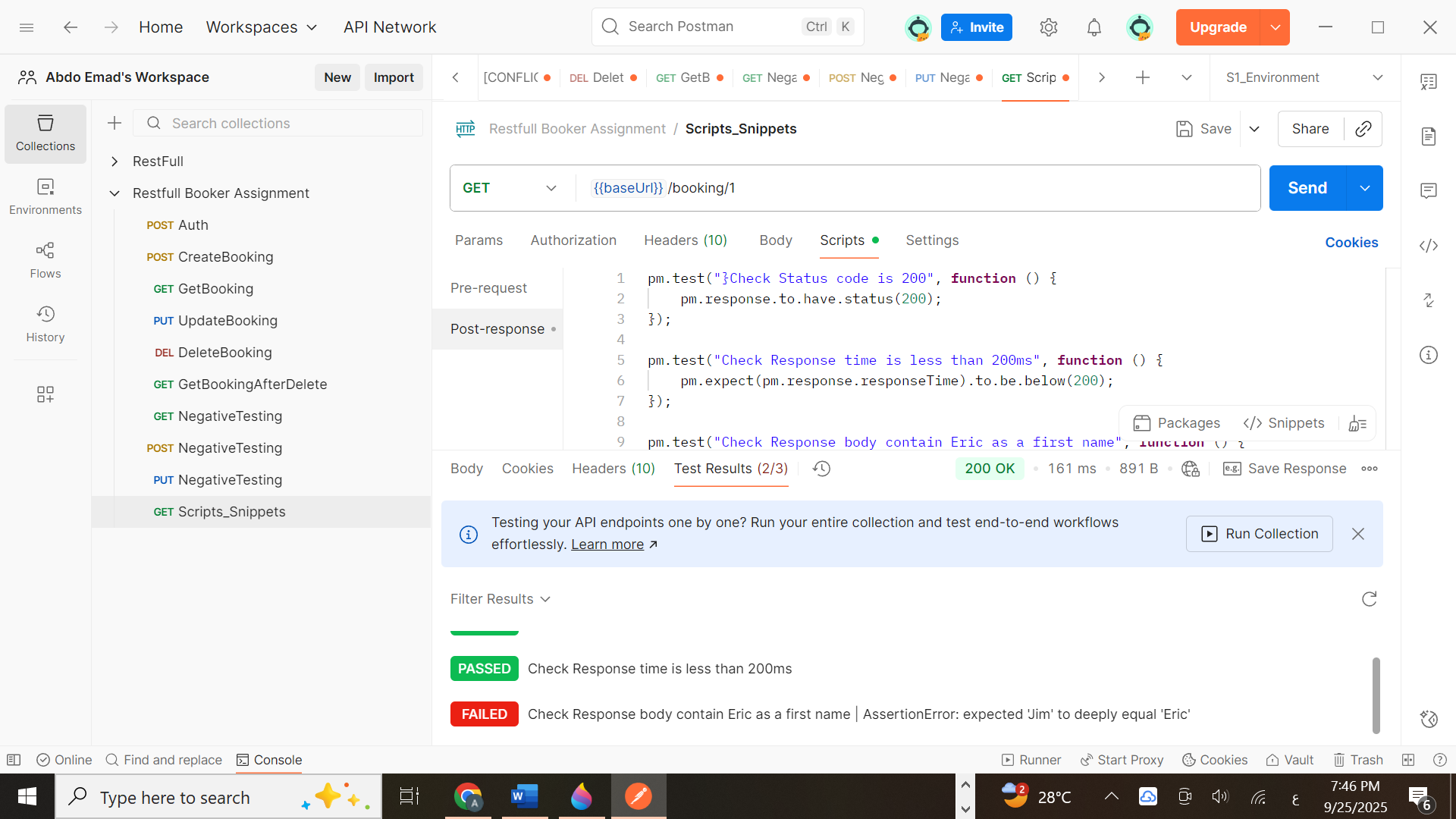
**Test cases:**

* **Create Booking with missing fields** → Returned error (400/500) ✅
* **Access Booking with invalid ID** → Returned 404 ✅
* **Authentication with invalid credentials** → Returned 200 with no token / error message ✅









### **4. Automation & Reusability**

**Global Variables:**

baseUrl → <https://restful-booker.herokuapp.com>

Token → stored dynamically after Auth request

bookingId → stored after Create Booking

**Benefit:** No need to manually copy-paste IDs or tokens. All requests reused stored values automatically.

**Scripts (Automation):**

Added **test scripts** in Postman to validate status codes and response bodies.

Automated check of fields (firstname, lastname, bookingId, etc.).

### **5. Observations**

The API handled CRUD operations effectively.

Response times were fast (< 200ms for most requests).

Status codes were consistent with RESTful standards.

Error handling (negative tests) worked as expected.

### **6. Challenges & Solutions**

**Challenge:** Managing token across multiple requests.

**Solution:** Implemented Postman **scripts** to automatically extract and save token into environment variables.

**Challenge:** Reusing Booking IDs.

**Solution:** Saved bookingId into a global variable to reuse dynamically.

### **7. Conclusion**

The testing of Restful Booker API using Postman was successful.  
 Automation scripts and environment variables made the testing process more efficient and reusable.  
 The API proved to be **robust and reliable**, handling both valid and invalid scenarios properly.

**“All work, testing scripts, and documentation in this report were created and verified by me.”**

**Abdulrahman Emad**  
 *Freelance Software Tester – API & Automation Testing Specialist.*