Performance Testing Assessment Submission

Candidate Name: Jovita Shainy Dsouza
Date: 29/05/2025
Tool Used: Apache JMeter
API Under Test: https://restful-booker.herokuapp.com

---

1. Introduction

1.1 Scope

The objective of this assessment is to evaluate the performance, scalability, and stability of the Restful Booker API by creating efficient and reusable performance test scripts using Apache JMeter. This involves simulating realistic user load and traffic patterns, measuring system behavior under various conditions, and identifying any performance bottlenecks or error-prone areas.

1.2 Scope

This performance testing effort includes:

- Designing JMeter scripts to simulate key user interactions across all major API endpoints (`/auth`, `/booking`, `/booking/{id}`).
- Executing different types of performance tests including:
    - **Load Testing** to assess response time and throughput under normal usage
    - **Stress Testing** to determine the API's breaking point
    - **Soak Testing** to verify long-term stability
- Testing both successful and negative scenarios (e.g., valid vs. invalid tokens, missing data).
- Implementing key JMeter features like correlation, parameterization, assertions, and timers to ensure robust and accurate simulations.
- Monitoring and analyzing performance metrics such as average response time, error rate, and transactions per second (TPS).
- Identifying performance bottlenecks and providing optimization recommendations based on the test results.

---

2. Test Plan Structure (JMeter Script Design)

2.1 Thread Groups: Success Case

This thread group contains all test cases expected to succeed, organized by operation.

Controlled by the Throughput controller to hit 99% of the Threads .

2.1.1 Authentication Success

- **Sampler**: `Authentication`
- **Endpoint**: `POST /auth`
- **Purpose**: Retrieves authentication token
- **Note**: Token value (`token`) is extracted and reused for authorized endpoints

2.1.2 Create Booking Success

- **Sampler(s)**: `Create Booking Success`, `Create Booking Success-Retry`
- **Endpoint**: `POST /booking`
- **Purpose**: Creates a new booking with provided data

2.1.3 Get Booking IDs Success

- **Sampler**: GetBookingIds Success Example 1(All IDs)
- **Endpoint**: `GET /booking`
- **Purpose**: Fetch list of booking IDs (to use in dependent tests)

2.1.4 Update Booking Success

- **Sampler(s)**: `Update Booking Success`
- **Endpoint**: `PUT /booking/{id}`

- **Purpose**: Fully updates booking details using the extracted token

2.1.5 Partial Update Booking Success

- **Sampler(s)**: `Partial Update Booking Success`
- **Endpoint**: `PATCH /booking/{id}`
- **Purpose**: Partially updates booking fields (e.g., firstname, lastname)

2.1.6 Delete Booking Success

- **Sampler(s)**: `Delete Booking Success`
- **Endpoint**: `DELETE /booking/{id}`
- **Purpose**: Deletes the created booking

2.2 Thread Group: Failure Cases

This group tests how the API handles incorrect or invalid operations/inputs.

Controlled by the Throughput controller to hit 1% of the Threads

2.2.1 Authentication Failure

- User inputs invalid credentials for `/auth`

2.2.2 Get Booking ID Failure

- Invalid payload or user does not provide all required fields

2.2.3 Create Booking Failure

- Invalid payload or user does not provide all required fields

2.2.4 Update Booking Failure

- Update without token or with invalid `bookingid` such as session expiry, invalid credentials etc

2.2.5 Partial Update Booking Failure

- Malformed JSON or missing token

2.2.6 Delete Booking Failure

- Attempting to delete nonexistent `bookingid` like 999999 or without auth

2.3 Controllers Used:

  - Throughput Controller
  - Transaction Controller
  - If Controller
  - While controller

- **Elements Implemented:**
  - CSV Data Set Config (for data-driven testing)
  - Regular Expression Extractor (for correlation of booking IDs & token)
  - Constant Timer
  - JSR223 PreProcessor and HTTP Request JSR223 Sampler
  - Assertions (Status code, response message, fields)
  - Summary Report, Aggregate Report, View Results Tree,

Data-Driven Tests:

Inputs like firstname, last name, checkin,checkout and totalprice are read from a CSV file to simulate multiple bookings.

---

4. Performance Testing Types Implemented

Type	Description	Load Profile

Load Test	Simulates expected traffic (1–10 TPS)	10 users, 10 min
Stress Test	Identifies max capacity before failure	100 users, ramp-up 1 min
Soak Test	Checks stability over time	10 users for 30 mins

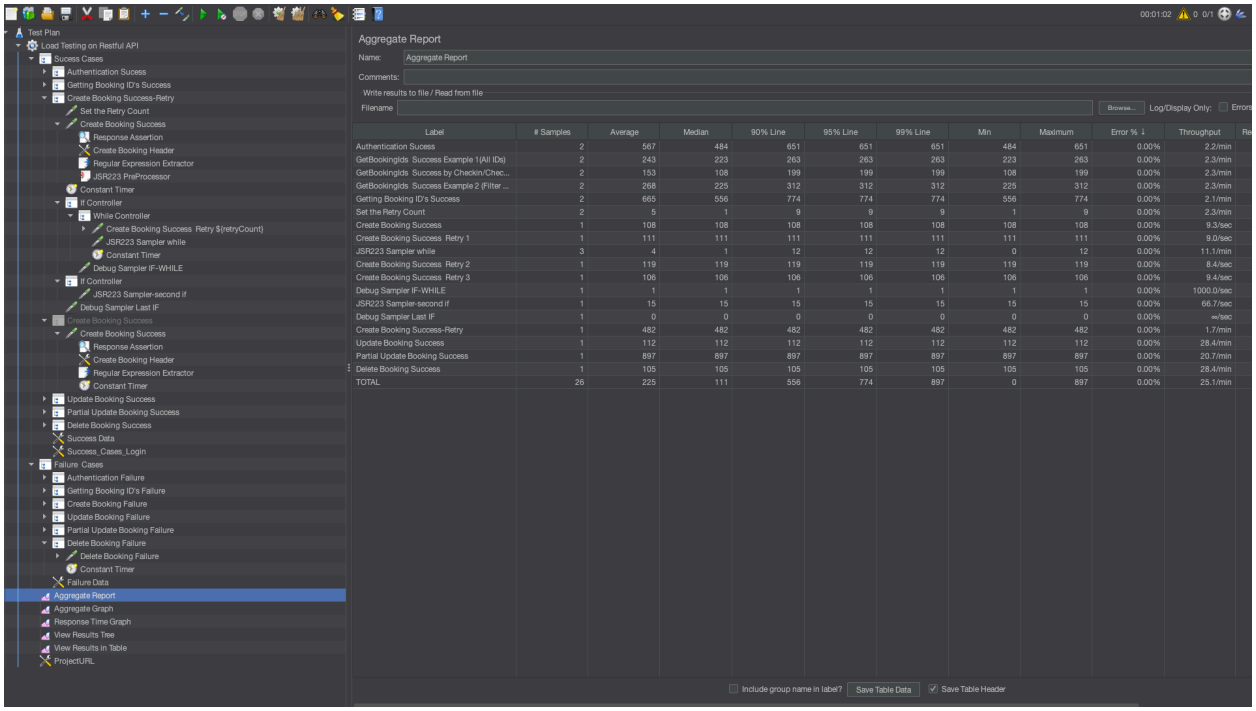
Failure Handling:

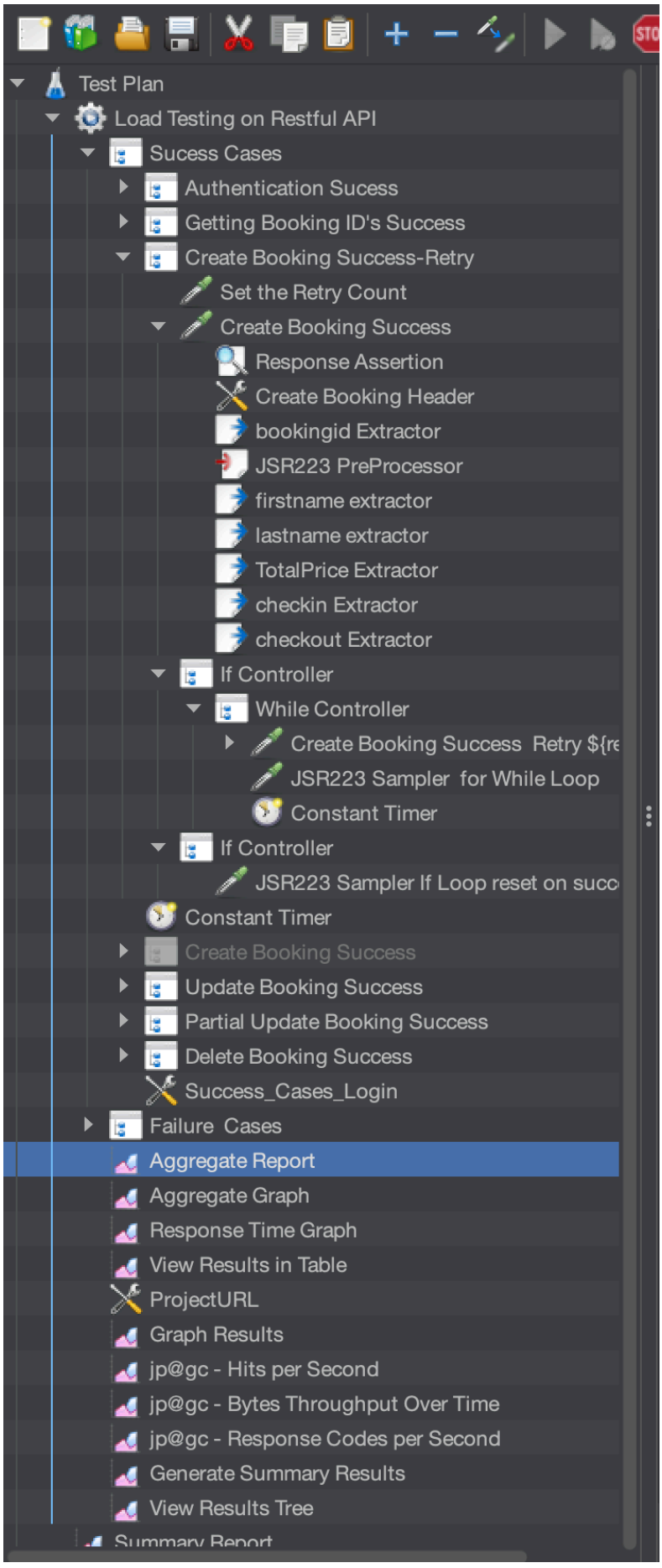
- Retries for creating booking with limited attempts for 5xx errors
- Assertions log failures for response code mismatches

Retry Logic

Before initiating the transaction, create a ‘JSR223 Sampler’ and initiate the Counter as Value ‘1’

If the transaction is successful and is returning as 200, that means the transaction was successful and it will follow the rest of the workflow. Incase of transient failures like 500 and 503, it will retry the same request thrice before killing the thread and starting the next thread.




Test Plan
Load Testing on Restful API
Sucess Cases
Authentication Sucess
Getting Booking ID's Success
Create Booking Success-Retry
Set the Retry Count
Create Booking Success
Response Assertion
Create Booking Header
bookingid Extractor
JSR223 PreProcessor
firstname extractor
lastname extractor
TotalPrice Extractor
checkin Extractor
checkout Extractor
If Controller
While Controller
Create Booking Success Retry ${re
JSR223 Sampler for While Loop
Constant Timer
If Controller
JSR223 Sampler If Loop reset on succ
Constant Timer
Create Booking Success
Update Booking Success
Partial Update Booking Success
Delete Booking Success
Success_Cases_Login
Failure Cases
Aggregate Report
Aggregate Graph
Response Time Graph
View Results in Table
ProjectURL
Graph Results
jp@gc - Hits per Second
jp@gc - Bytes Throughput Over Time
jp@gc - Response Codes per Second
Generate Summary Results
View Results Tree
Summary Report

---

5. JMeter Features Used

Feature	Purpose
CSV Data Set Config	Parameterized multiple inputs
Regular Extractor	Correlation of dynamic values (booking ID, token)
Assertions	Validate status code, response time, fields
Constant Timer	Random delays to mimic user behavior
Header Manager	Custom headers to prevent 418 errors ("teapots")
Listeners	Aggregate Report, Graph Results, Summary Report

---

6. Results & Observations

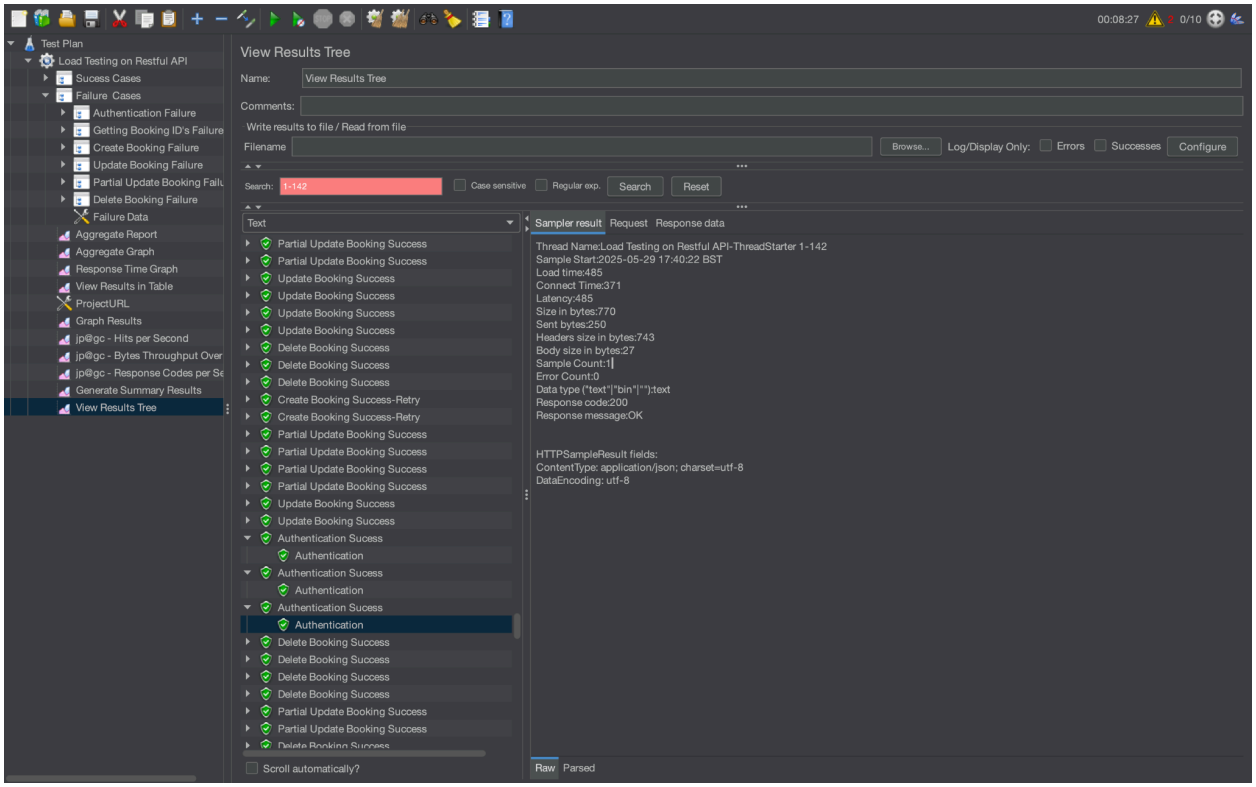
6.1Key Metrics

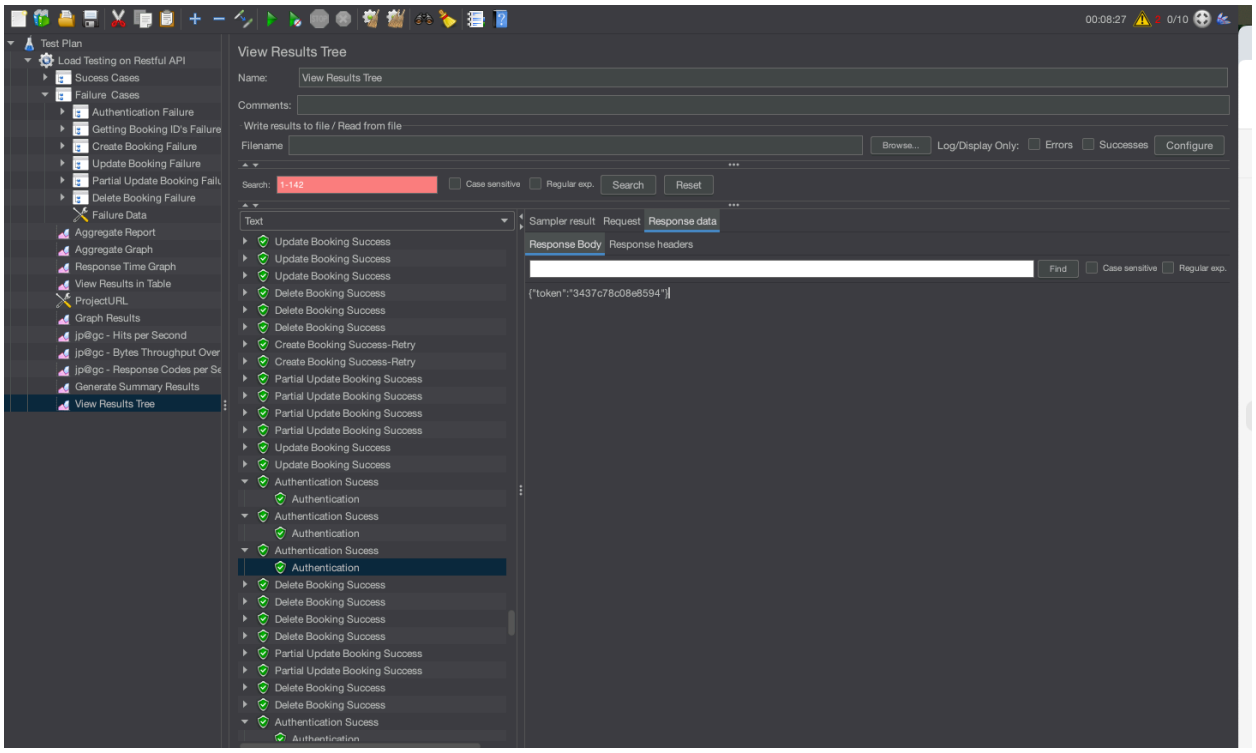
Perf. test type: TPS	Avg Response (ms)	90%ile (ms)	Max (ms)	Errors (%)
Load Testing: 1-2 TPS	<700	<700	<7000	<6
Stress Testing: 117.8 TPS	<800	<1312	<6899	<8
Soap Testing: 1-2 TPS	<625	<677	<3107	<5

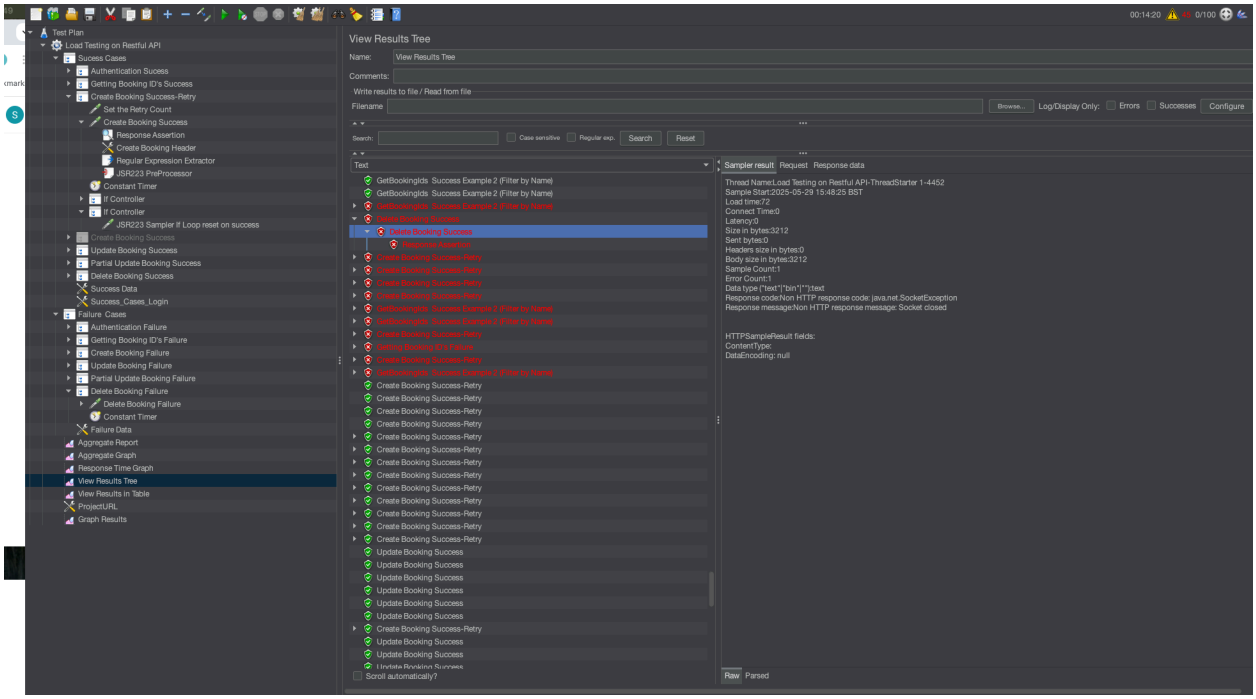
Error Analysis:

403 errors on `Updating Booking,Partial Booking and Deleting API` indicating sessions are not handled correctly .

After the validation of each requests on view results tree its been concluded even with the rightly correlated Data and right input parameters still Requests are throwing the 403 errors

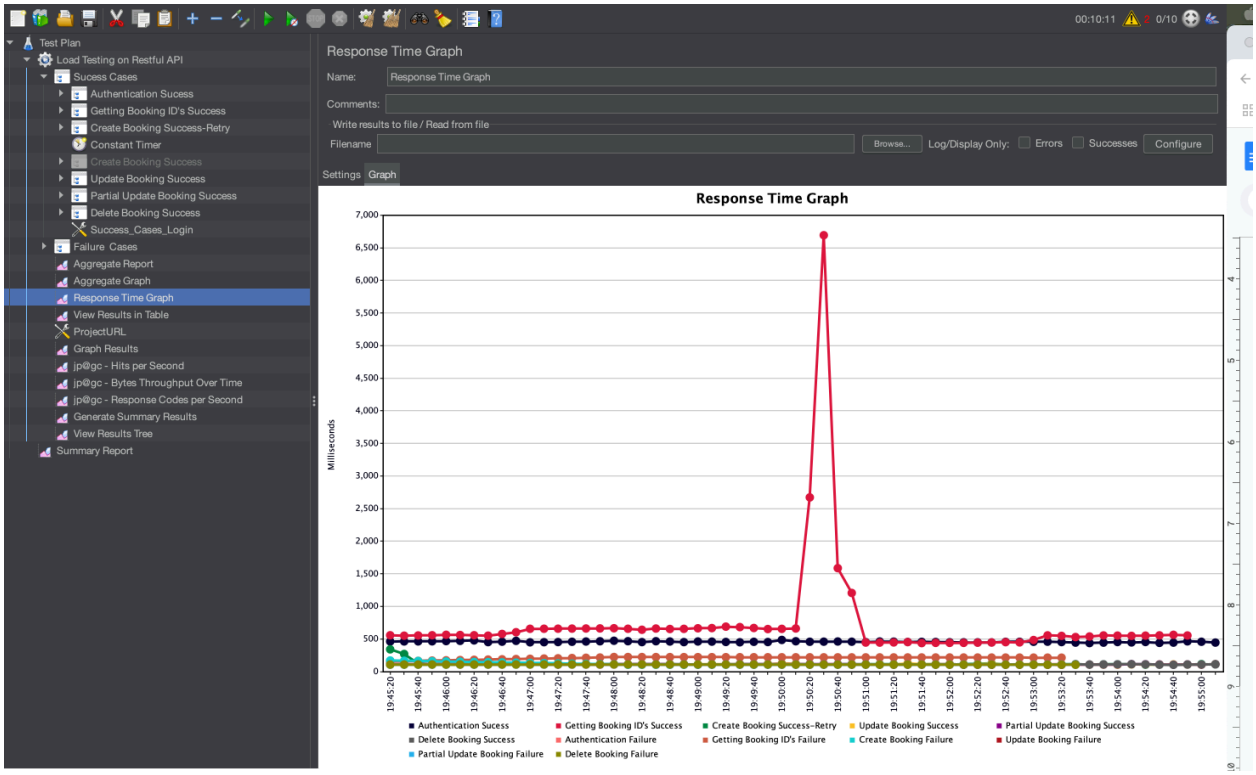




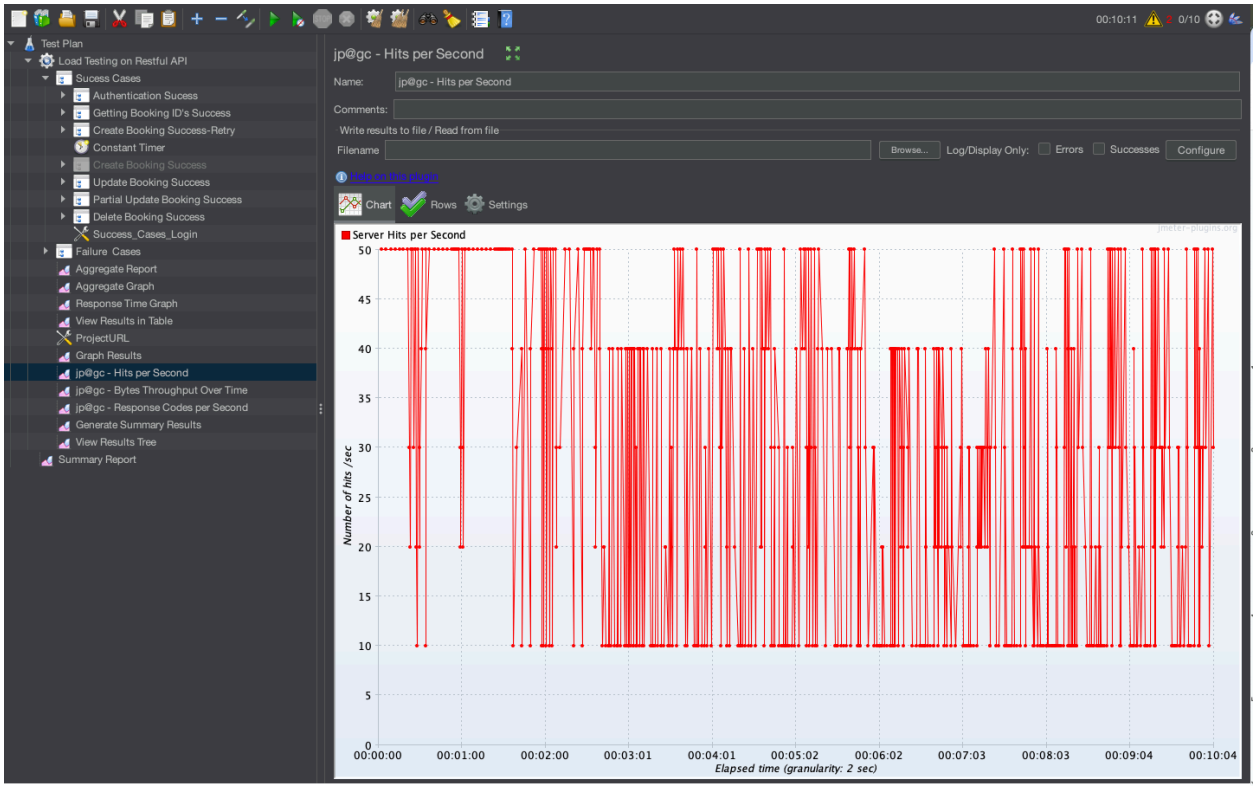


Graphs Attached for Load Testing

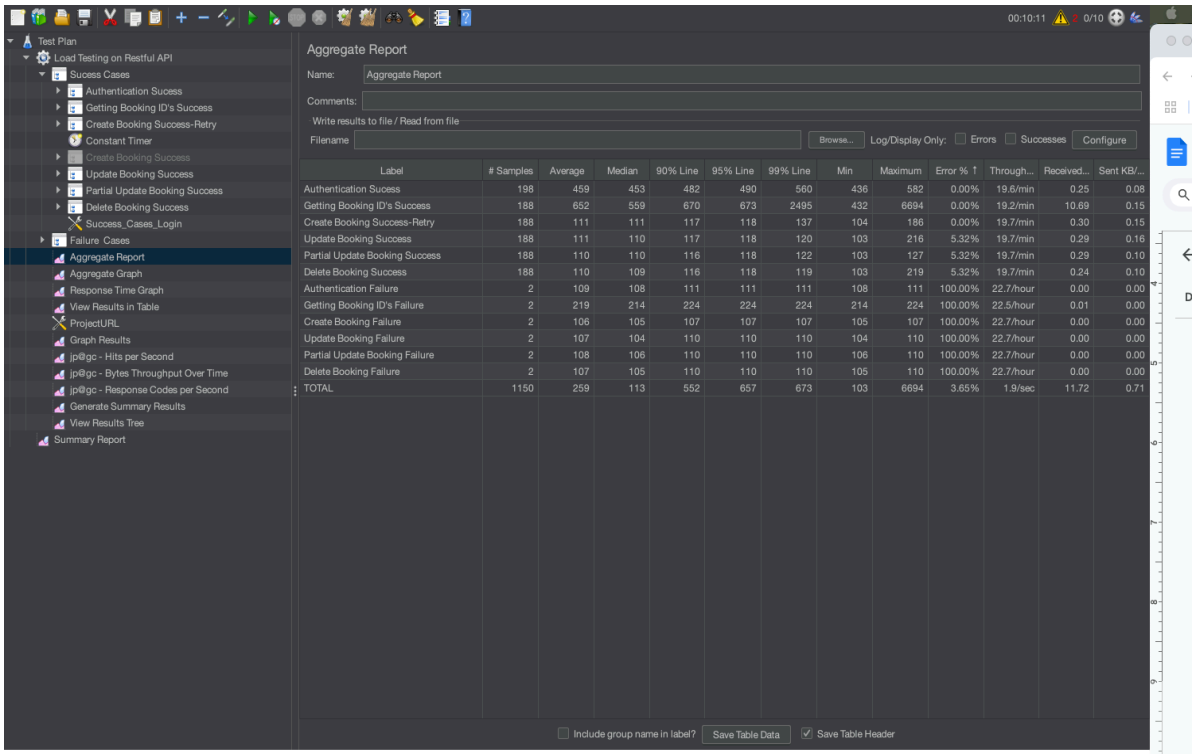
- Response Time Over Time


Test Plan
Load Testing on Restful API
Sucess Cases
Authentication Sucess
Getting Booking ID's Success
Create Booking Success-Retry
Constant Timer
Create Booking Success
Update Booking Success
Partial Update Booking Success
Delete Booking Success
Success_Cases_Login
Failure Cases
Aggregate Report
Aggregate Graph
Response Time Graph
View Results in Table
ProjectURL
Graph Results
jp@gc - Hits per Second
jp@gc - Bytes Throughput Over Time
jp@gc - Response Codes per Second
Generate Summary Results
View Results Tree
Summary Report
Response Time Graph
Name:
Response Time Graph
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only:
Errors
Successes
Configure
Settings
Graph
Response Time Graph
Milliseconds
Authentication Sucess
Getting Booking ID's Success
Create Booking Success-Retry
Update Booking Success
Partial Update Booking Success
Delete Booking Success
Authentication Failure
Getting Booking ID's Failure
Create Booking Failure
Update Booking Failure
Partial Update Booking Failure
Delete Booking Failure

Hits Per Sec

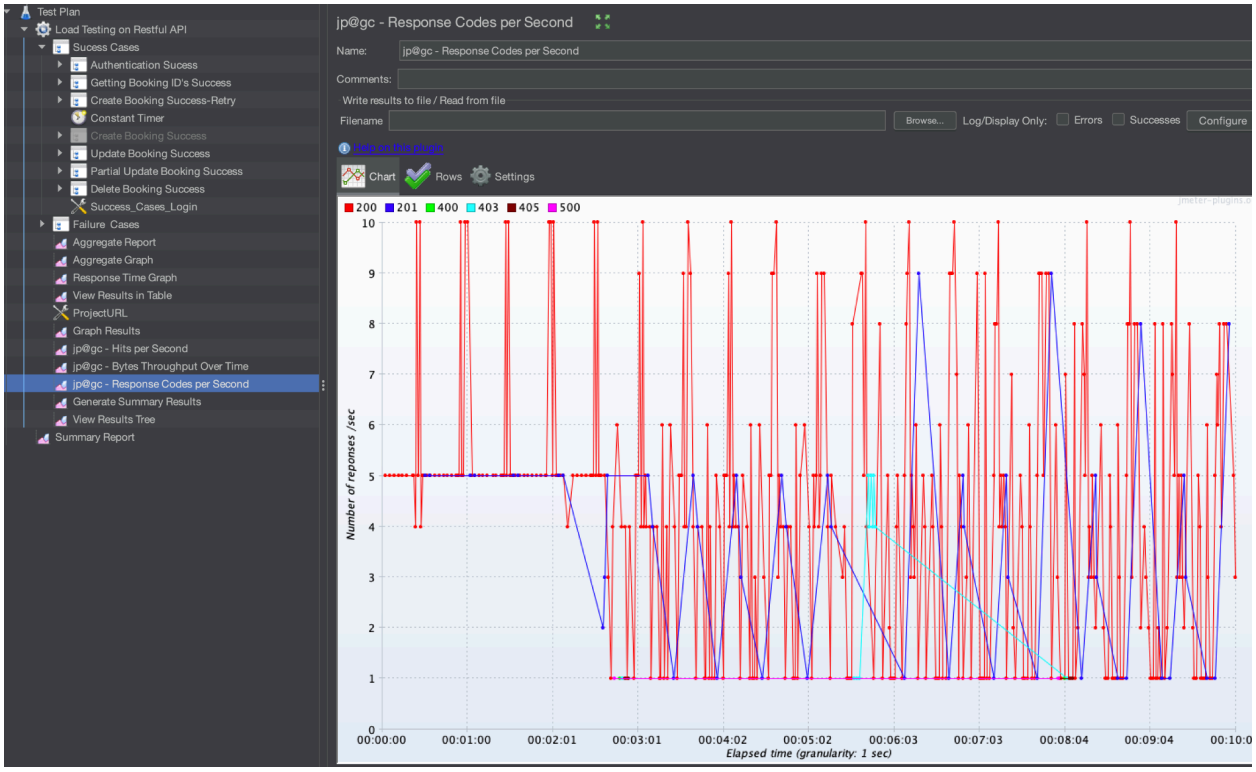


Aggregate Report

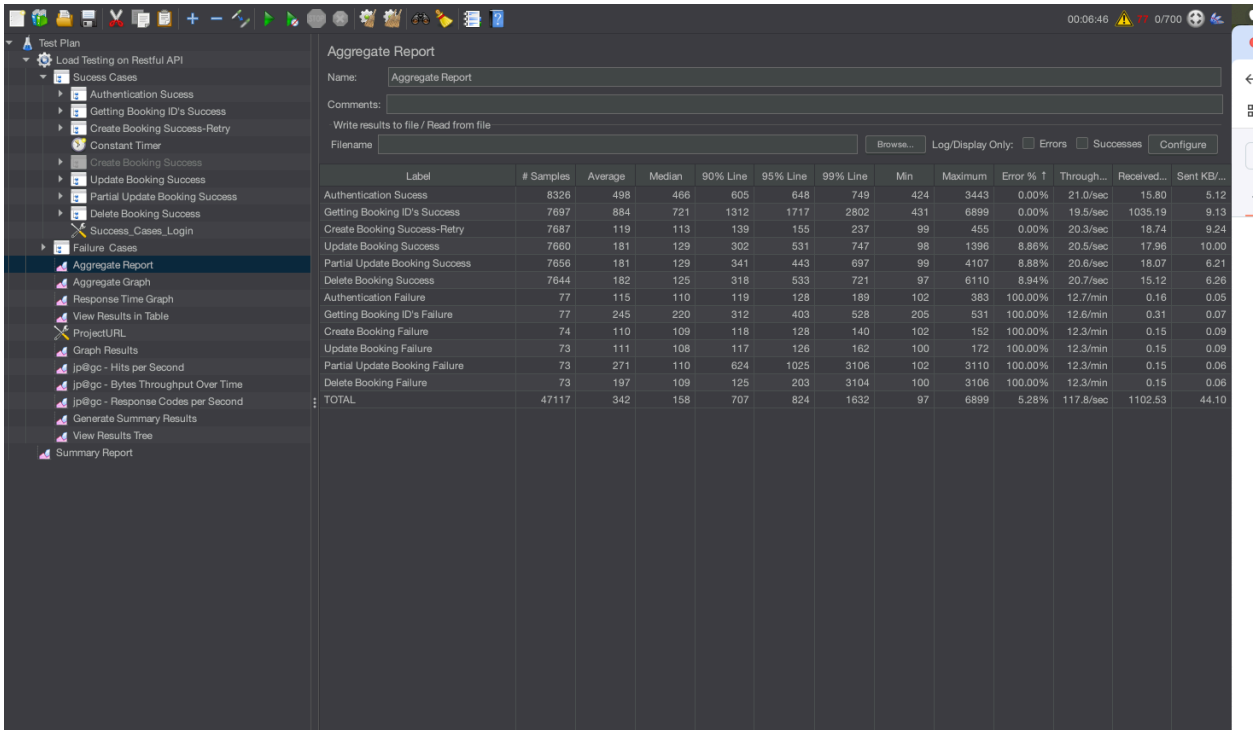


Error Code

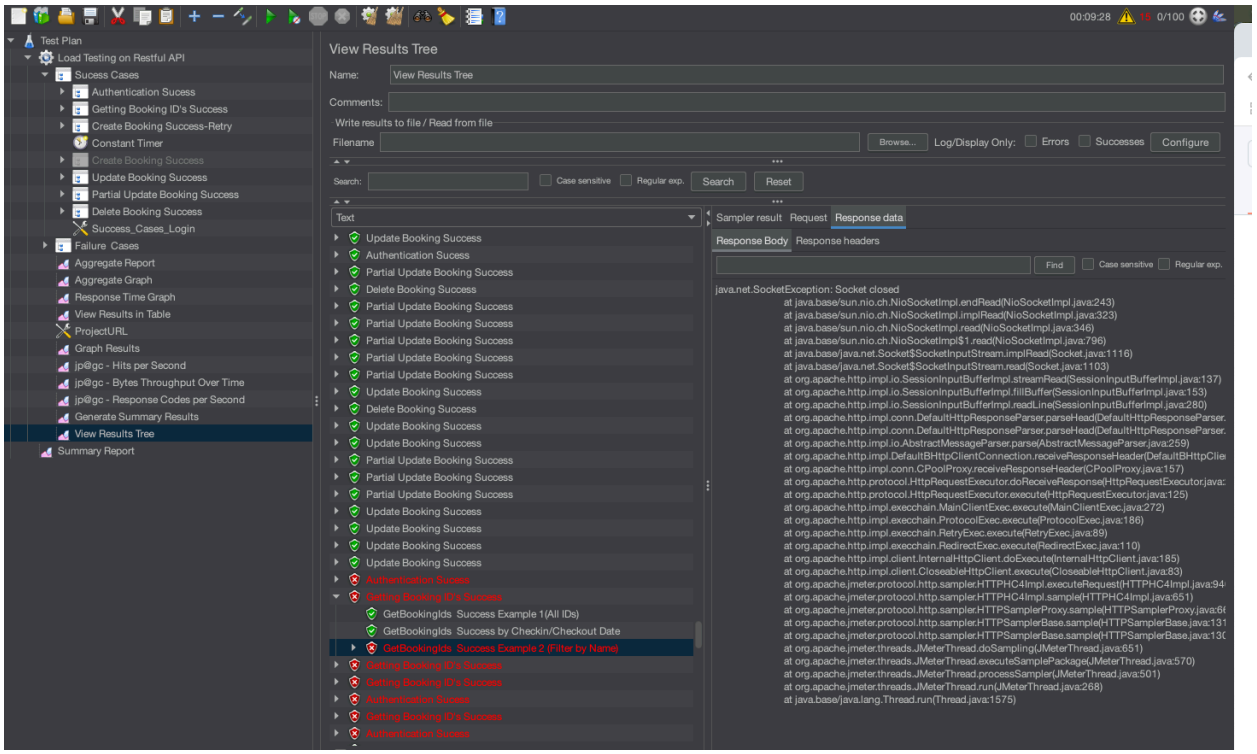
Observed 403 errors after the 5 mins of the tests.

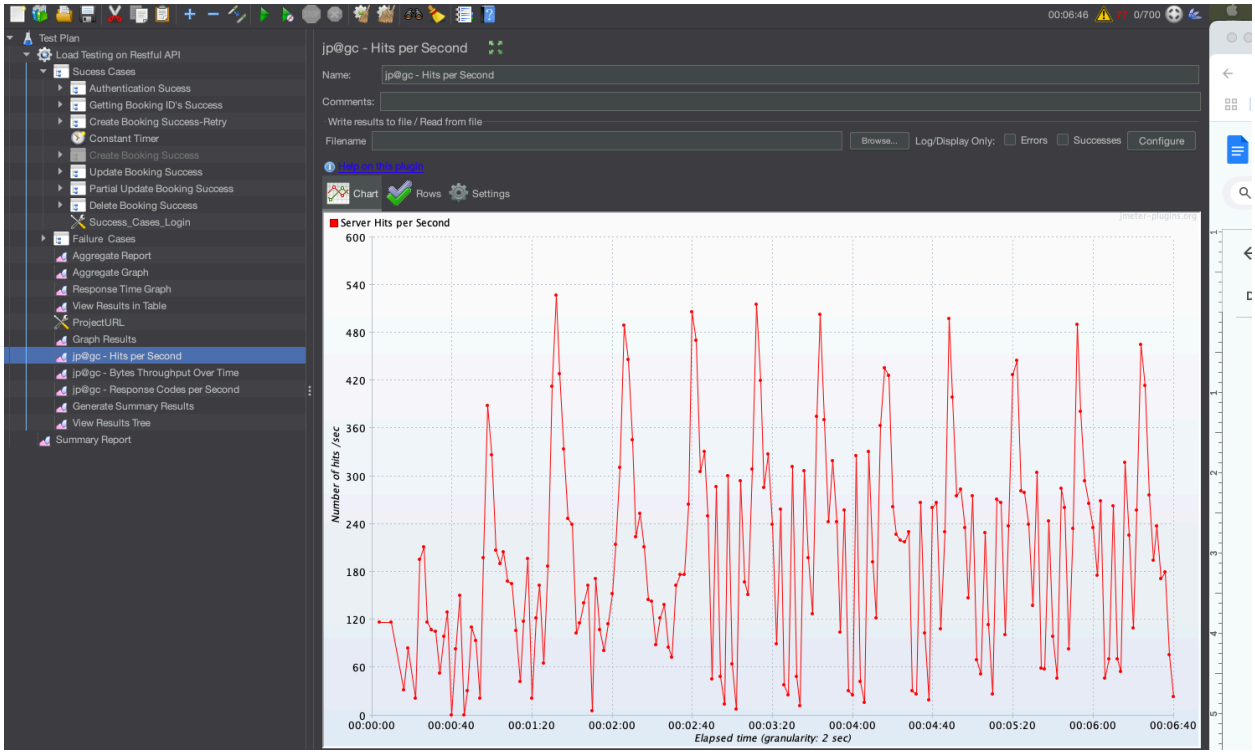


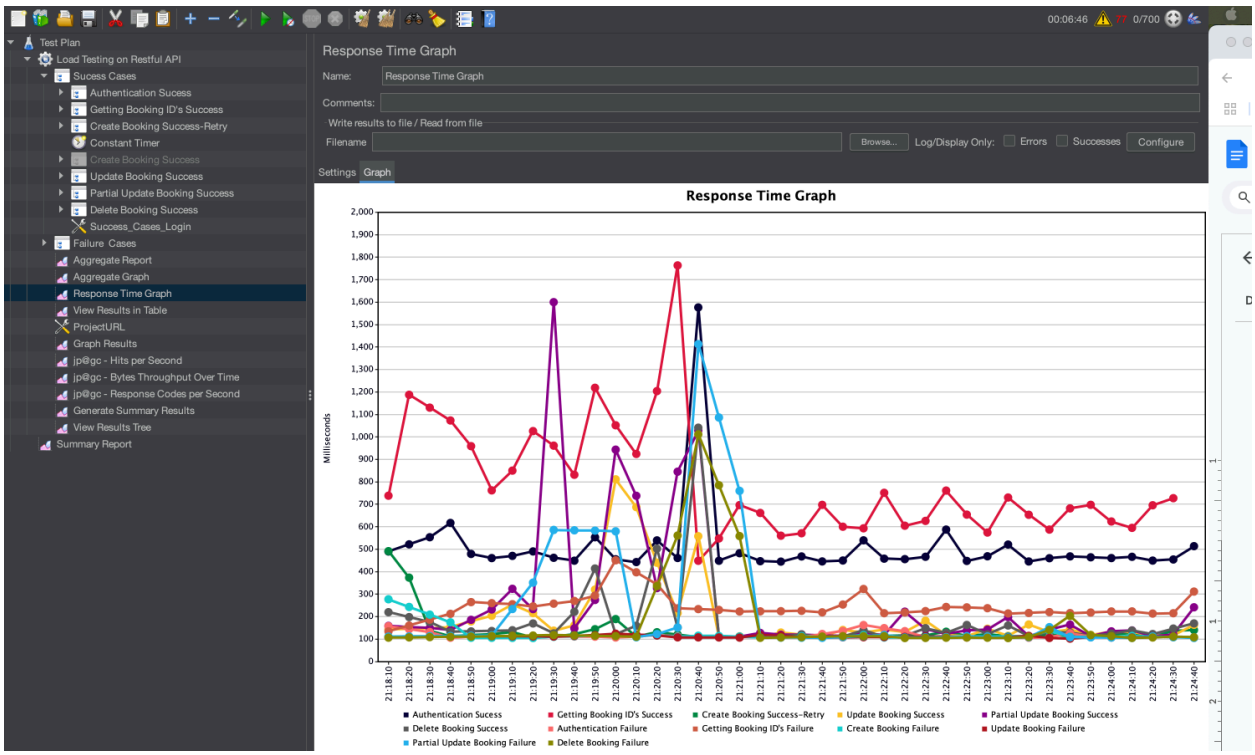
Graphs Attached for Stress Testing



Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error % ↑	Through...	Received...	Sent KB/...
Authentication Sucess	8326	498	466	605	648	749	424	3443	0.00%	21.0/sec	15.80	5.12
Getting Booking ID's Success	7697	884	721	1312	1717	2802	431	6899	0.00%	19.5/sec	1035.19	9.13
Create Booking Success-Retry	7687	119	113	139	155	237	99	455	0.00%	20.3/sec	18.74	9.24
Update Booking Success	7660	181	129	302	531	747	98	1396	8.86%	20.5/sec	17.96	10.00
Partial Update Booking Success	7656	181	129	341	443	697	99	4107	8.88%	20.6/sec	18.07	6.21
Delete Booking Success	7644	182	125	318	533	721	97	6110	8.94%	20.7/sec	15.12	6.26
Authentication Failure	77	115	110	119	128	189	102	383	100.00%	12.7/min	0.16	0.05
Getting Booking ID's Failure	77	245	220	312	403	528	205	531	100.00%	12.6/min	0.31	0.07
Create Booking Failure	74	110	109	118	128	140	102	152	100.00%	12.3/min	0.15	0.09
Update Booking Failure	73	111	108	117	126	162	100	172	100.00%	12.3/min	0.15	0.09
Partial Update Booking Failure	73	271	110	624	1025	3106	102	3110	100.00%	12.3/min	0.15	0.06
Delete Booking Failure	73	197	109	125	203	3104	100	3106	100.00%	12.3/min	0.15	0.06
TOTAL	47117	342	158	707	824	1632	97	6899	5.28%	117.8/sec	1102.53	44.10







Graphs attached for Soak Testing

Aggregate Report

Name: Aggregate Report

Comments:

Write results to file / Read from file

Filename | Browse... | Log/Display Only: Errors Successes | Configure

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maxim... ↓	Error %	Through...	Received...	Sent KB/...
Delete Booking Success	554	125	109	116	120	214	102	3107	5.05%	18.8/min	0.23	0.09
Getting Booking ID's Success	555	613	561	675	687	2438	430	2860	0.00%	18.5/min	10.69	0.14
Authentication Sucess	564	461	449	480	495	629	421	2445	0.00%	18.7/min	0.24	0.08
Create Booking Success-Retry	554	114	110	115	118	141	103	2183	0.00%	18.7/min	0.29	0.14
Partial Update Booking Success	554	113	110	117	119	161	103	586	4.51%	18.8/min	0.28	0.09
Update Booking Success	554	111	110	117	120	126	102	420	4.51%	18.8/min	0.28	0.15
Getting Booking ID's Failure	6	225	225	231	234	234	213	234	100.00%	13.7/hour	0.01	0.00
Update Booking Failure	6	109	107	110	119	119	105	119	100.00%	13.7/hour	0.00	0.00
Create Booking Failure	6	111	111	115	117	117	105	117	100.00%	13.7/hour	0.00	0.00
Delete Booking Failure	6	110	110	112	117	117	105	117	100.00%	13.7/hour	0.00	0.00
Partial Update Booking Failure	6	109	107	113	113	113	105	113	100.00%	13.7/hour	0.00	0.00
Authentication Failure	6	108	105	112	112	112	104	112	100.00%	13.7/hour	0.00	0.00
TOTAL	3371	256	113	555	657	689	102	3107	3.38%	1.9/sec	11.97	0.70

Response Time Graph

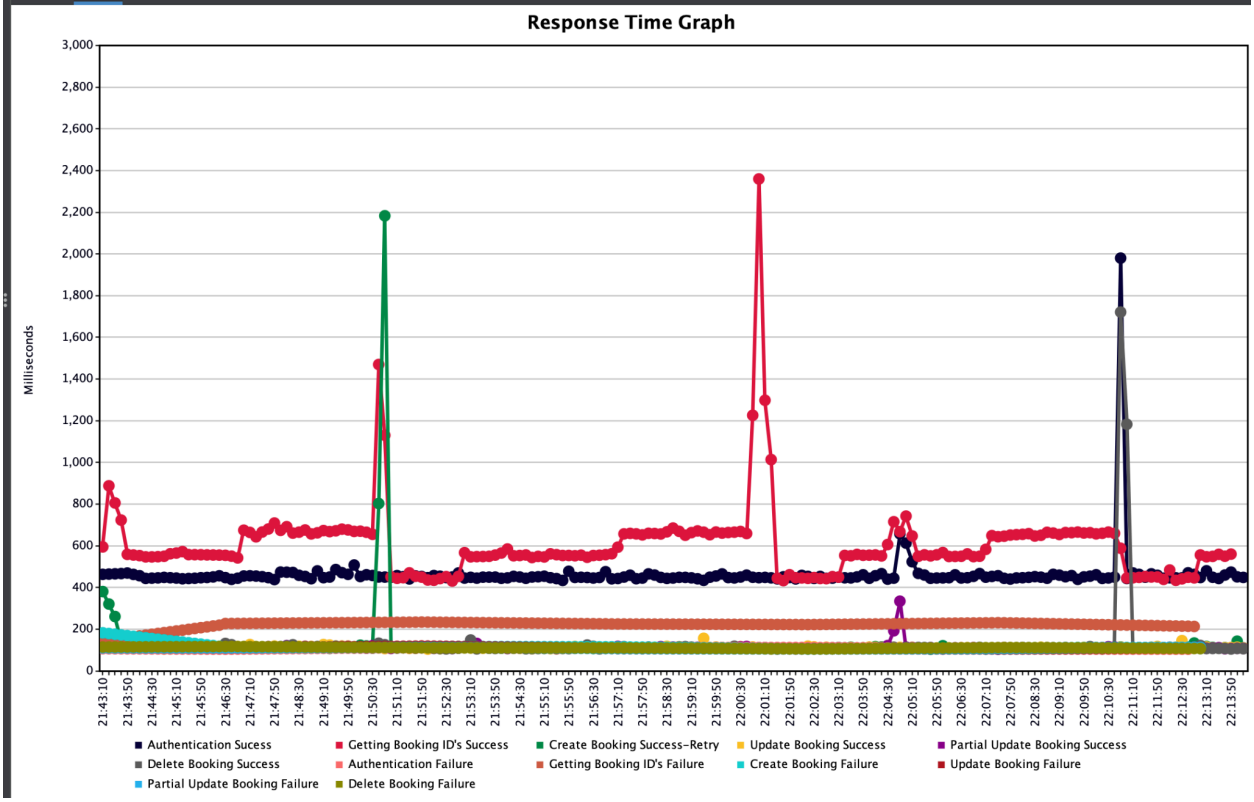
Name: Response Time Graph

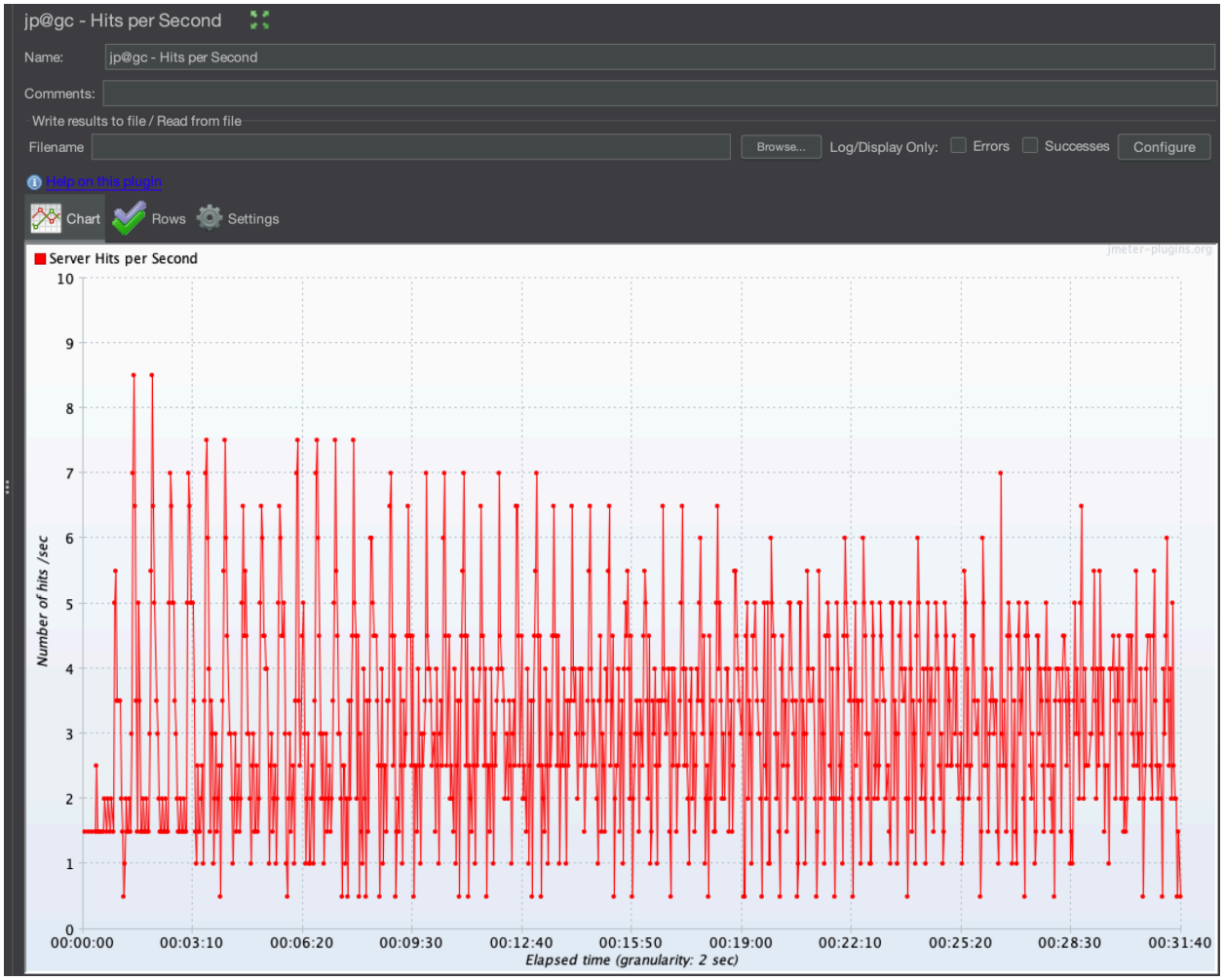
Comments:

Write results to file / Read from file

Filename | Browse... | Log/Display Only: Errors Successes | Configure

Settings | Graph





7. Bottlenecks & Improvement Suggestions

Bottlenecks Observed:

- Creating Booking ID ‘s token not used efficiently → spikes in token requests
- PUT requests fail within 1-2 TPS → potential backend session handling limit

Recommendations:

- Revisit the session logic of multiple threads for update/delete calls
- Ensure proper headers are set (`Accept: application/json`)

- Consider autoscaling for backend/database under high concurrency

---

- **How to Run:**
  Open `Load Testing on Restful API.jmx` in JMeter, update CSV path, run the test plan.

Conclusion

The Restful booker app performs reliably up to ~2 TPS . Issues begin to surface during load str testing, especially around token management and update operations. The JMeter scripts are modular, reusable, and designed to simulate real-world user activity with appropriate load profiles, timers, and error handling.