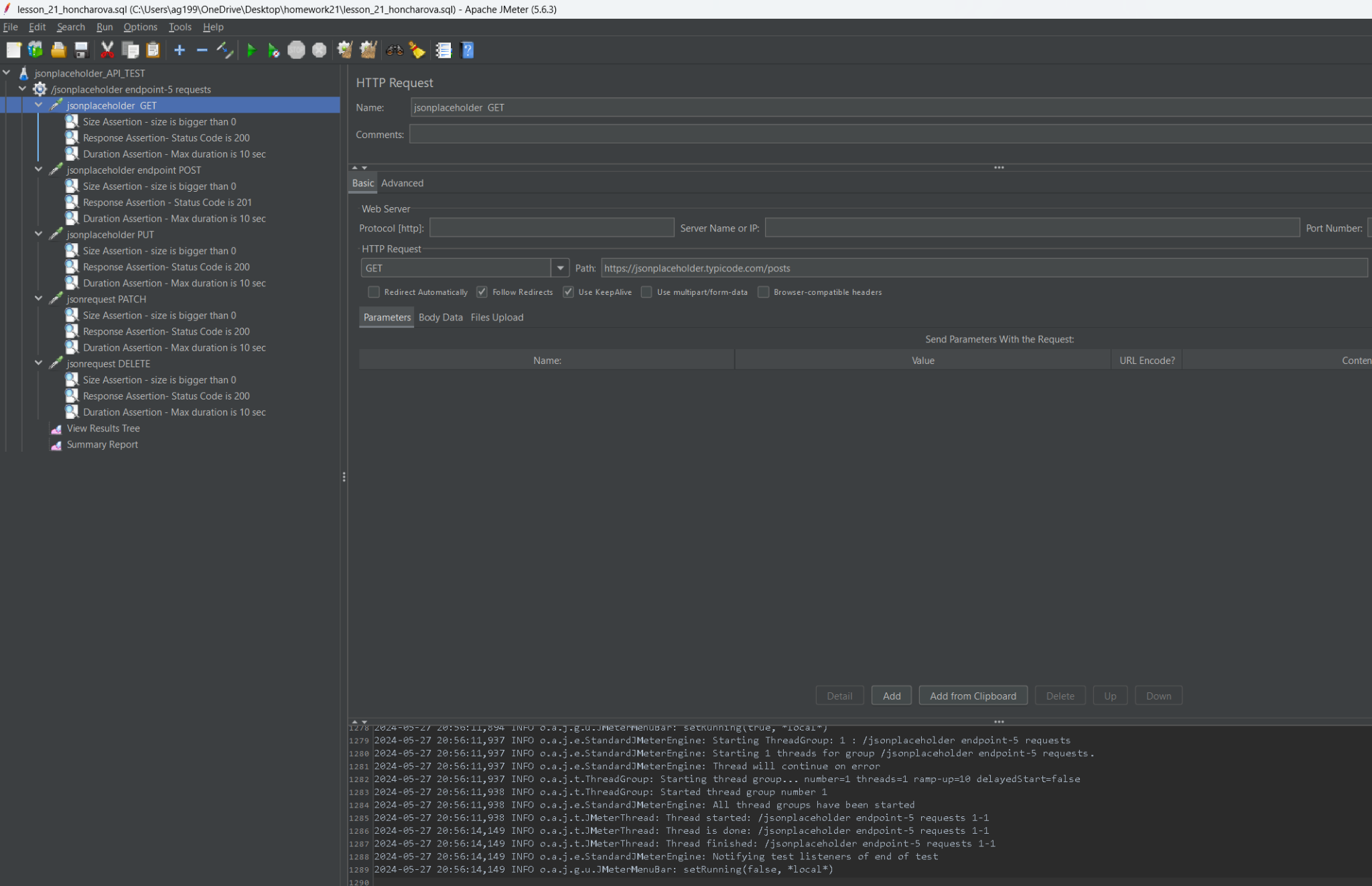
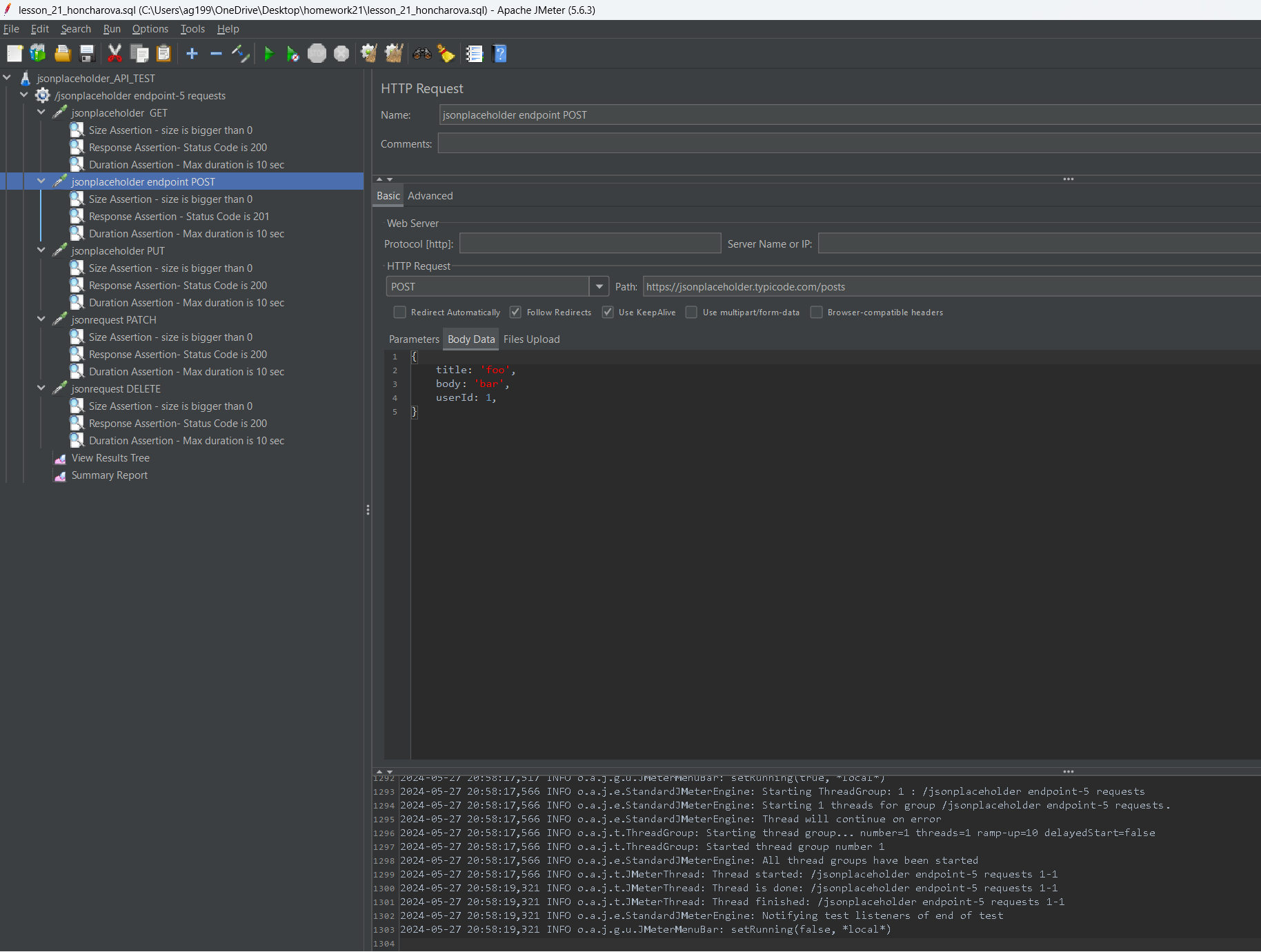
1. Створи набір з GET, POST, PUT, PATCH, DELETE запитів до [JSONPlaceholder](https://jsonplaceholder.typicode.com/), які надсилаються протягом 10 секунд у 3 ітерації.

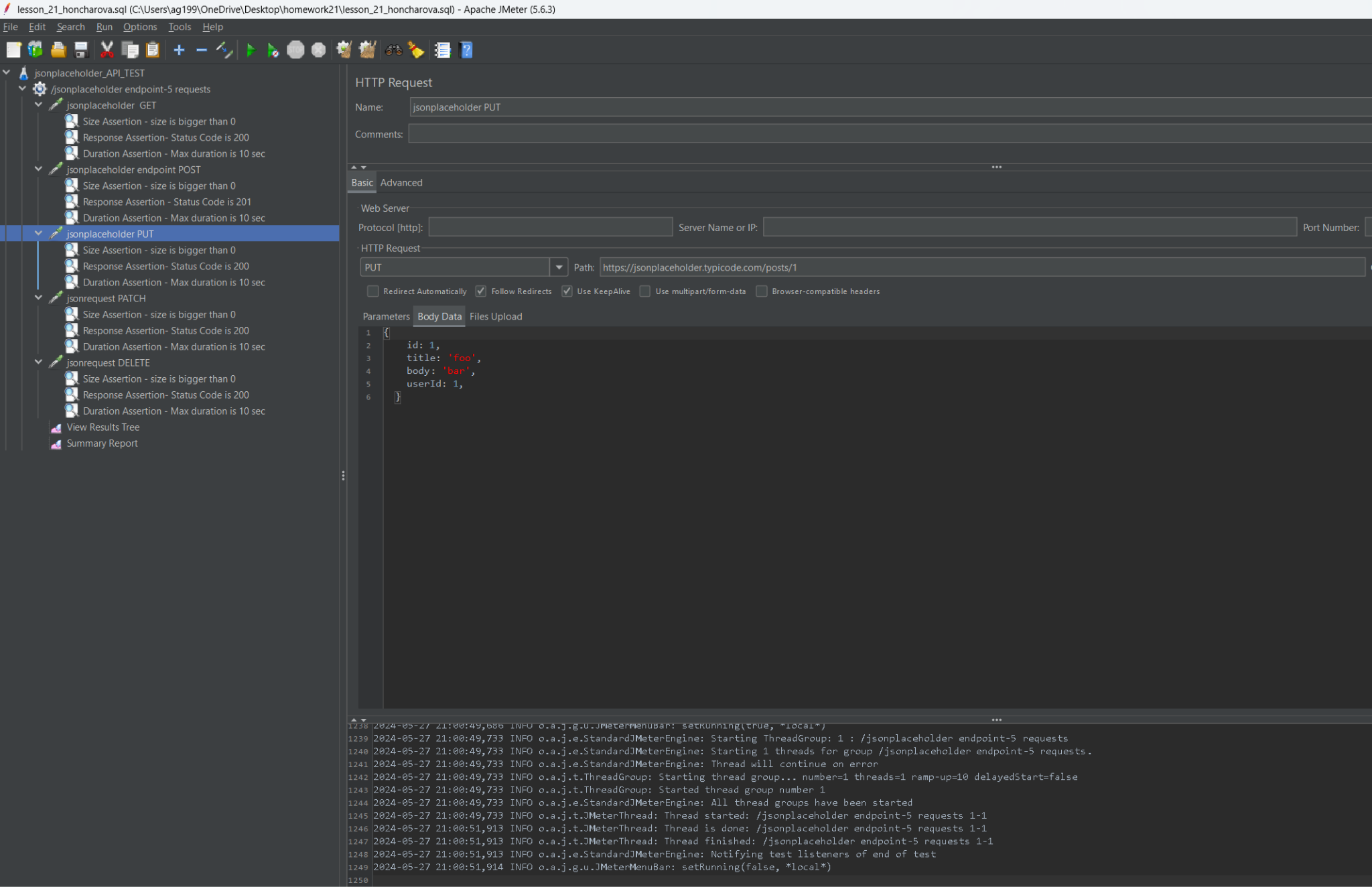
GET



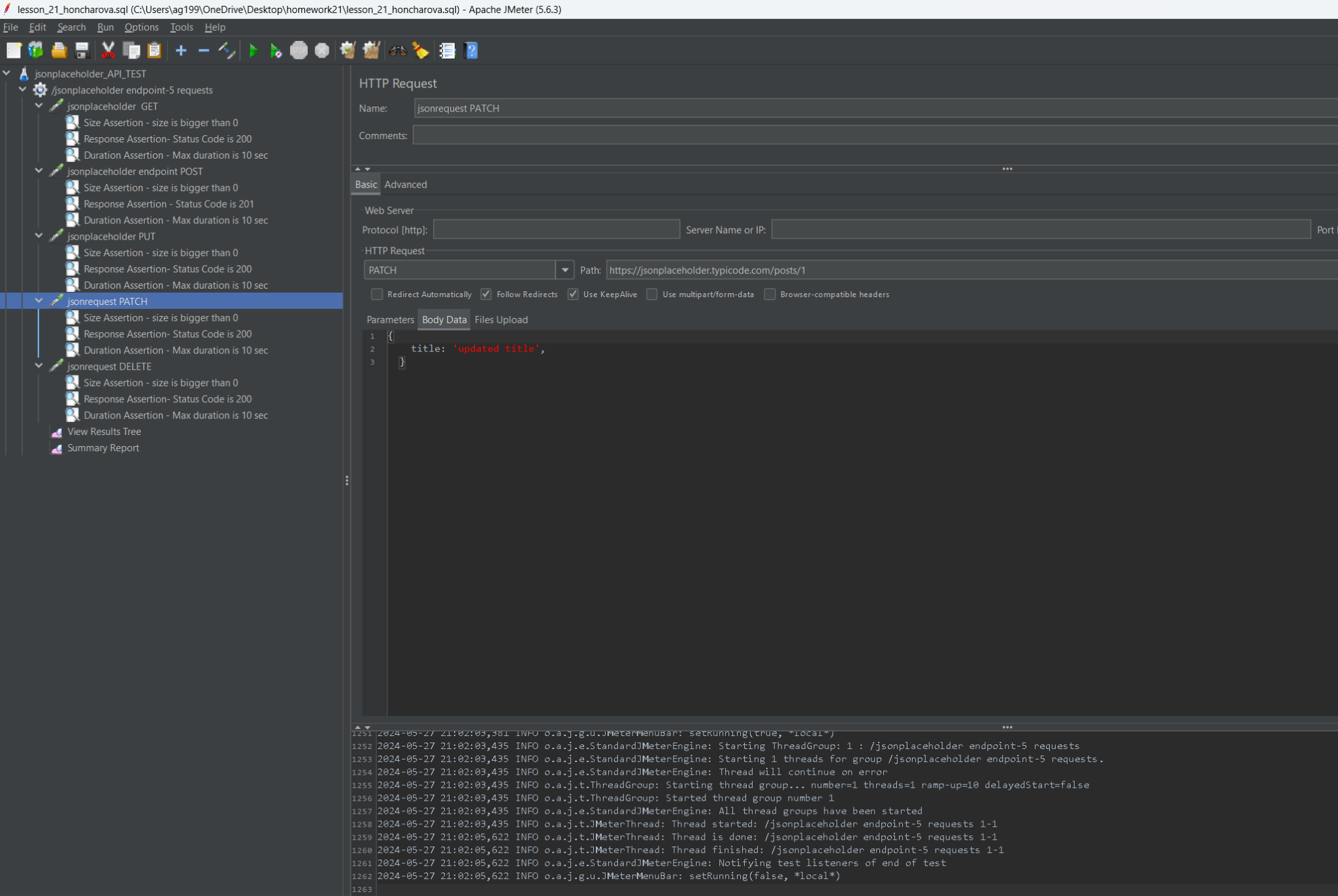
POST



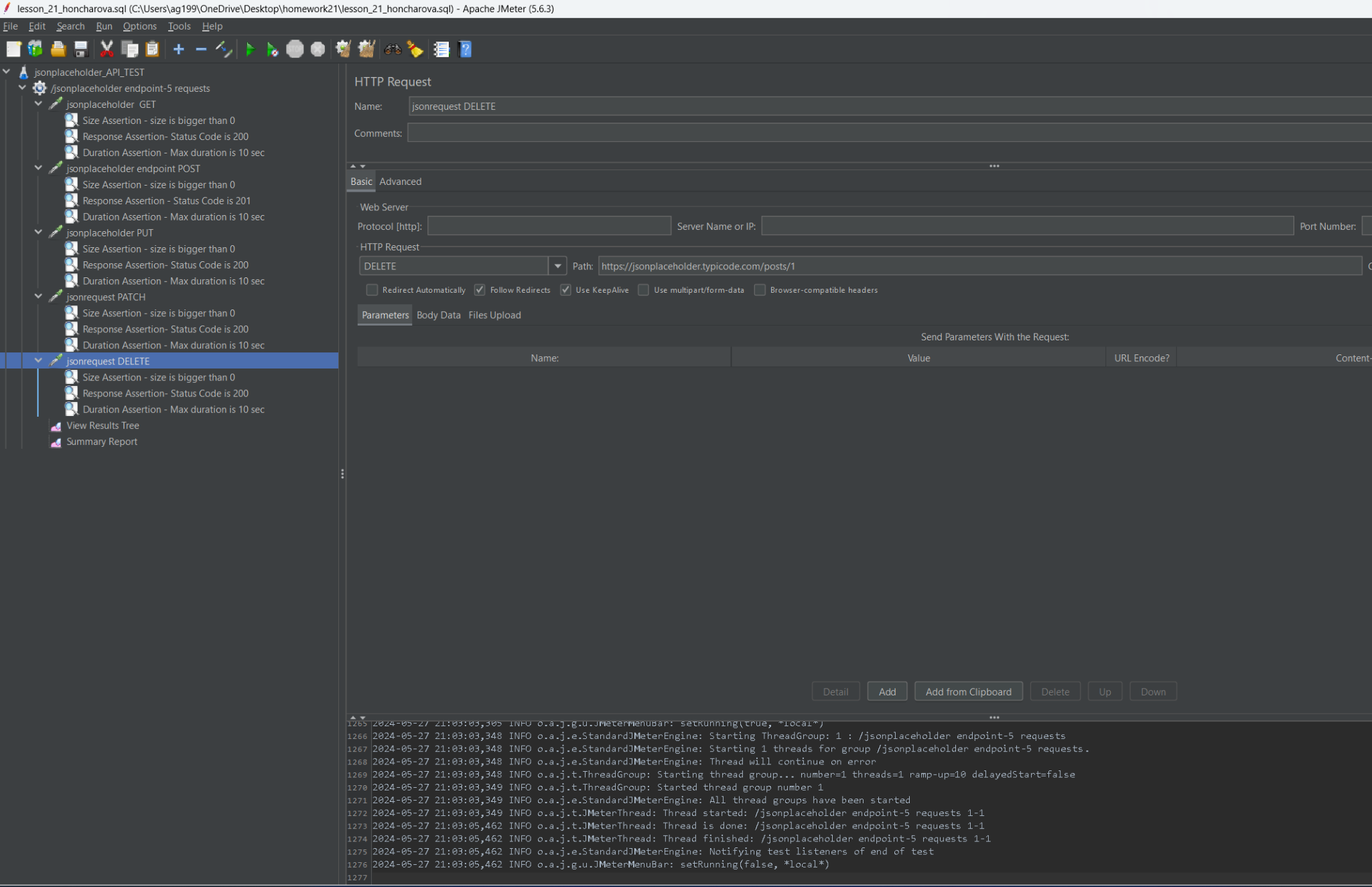
PUT



PATCH

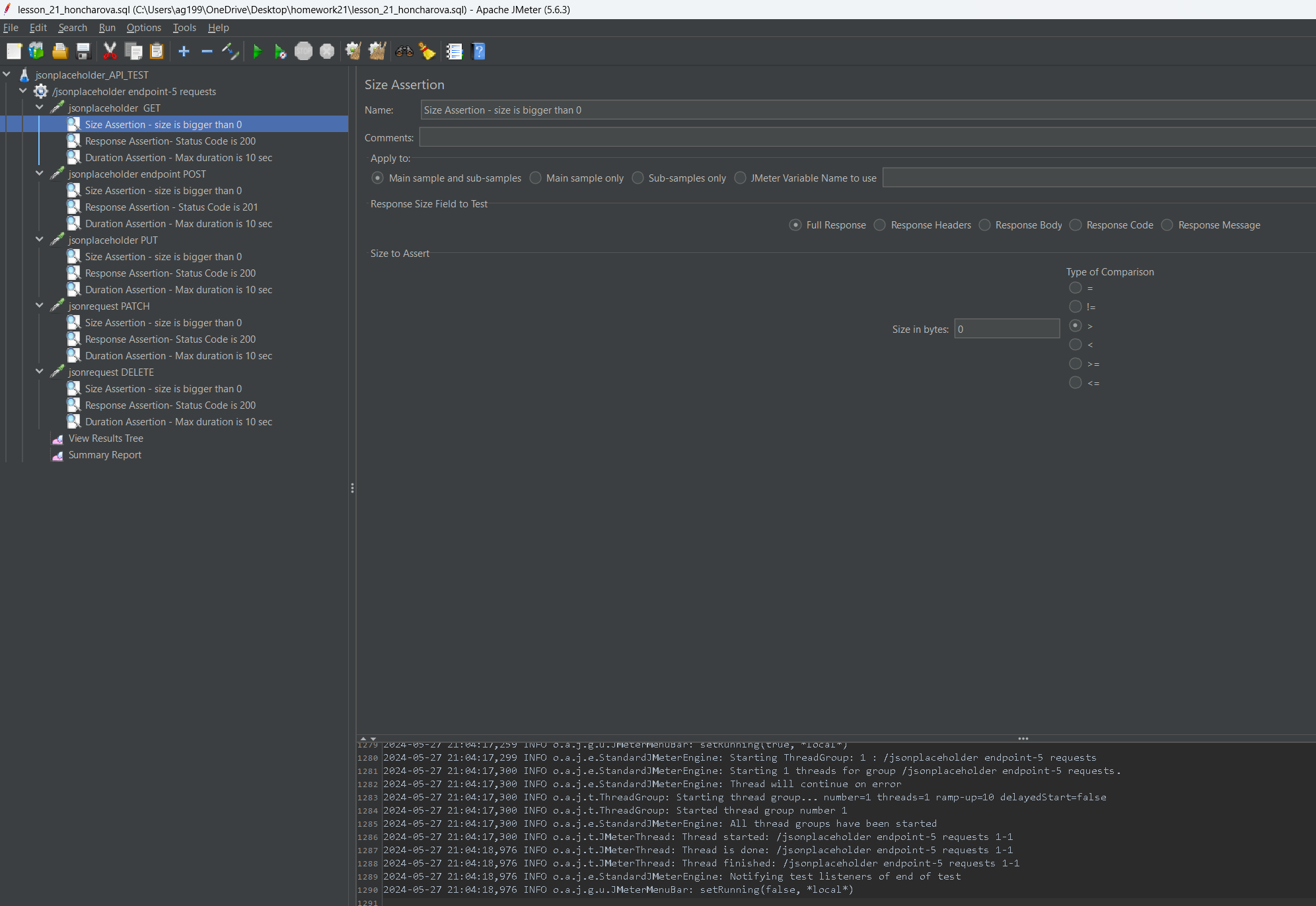


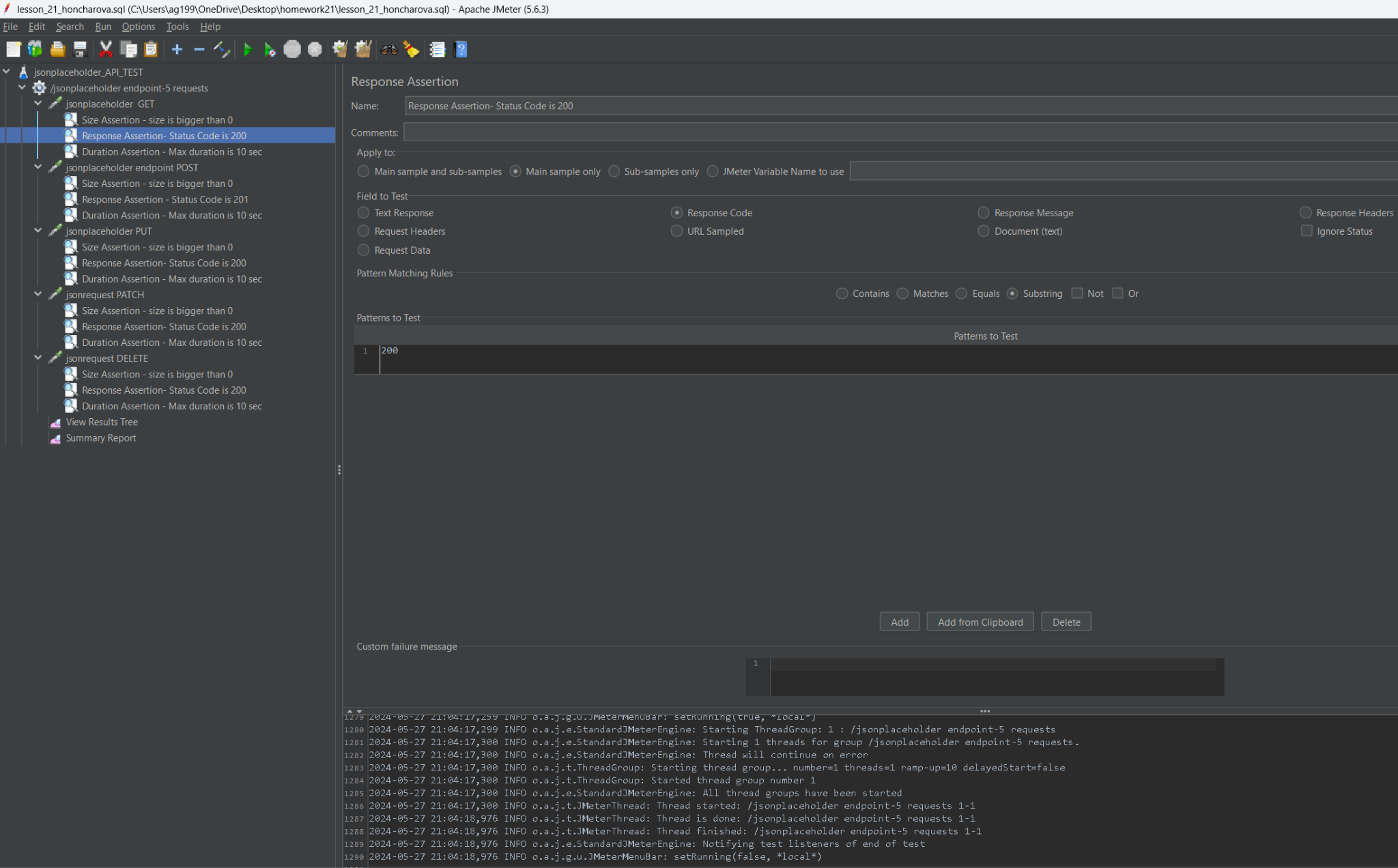
DELETE

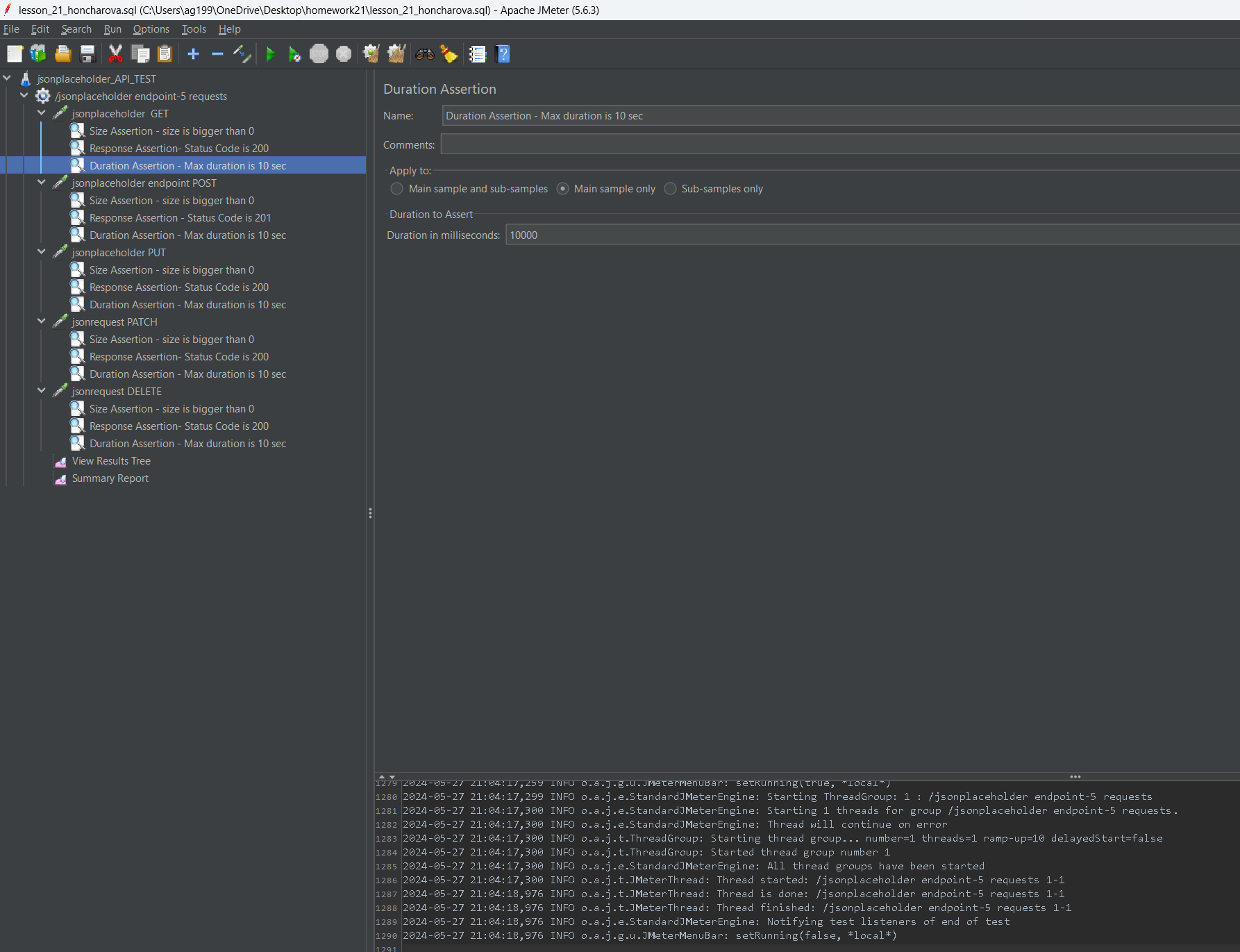


2. До кожного з запитів застосуй 3 різних assert’и.

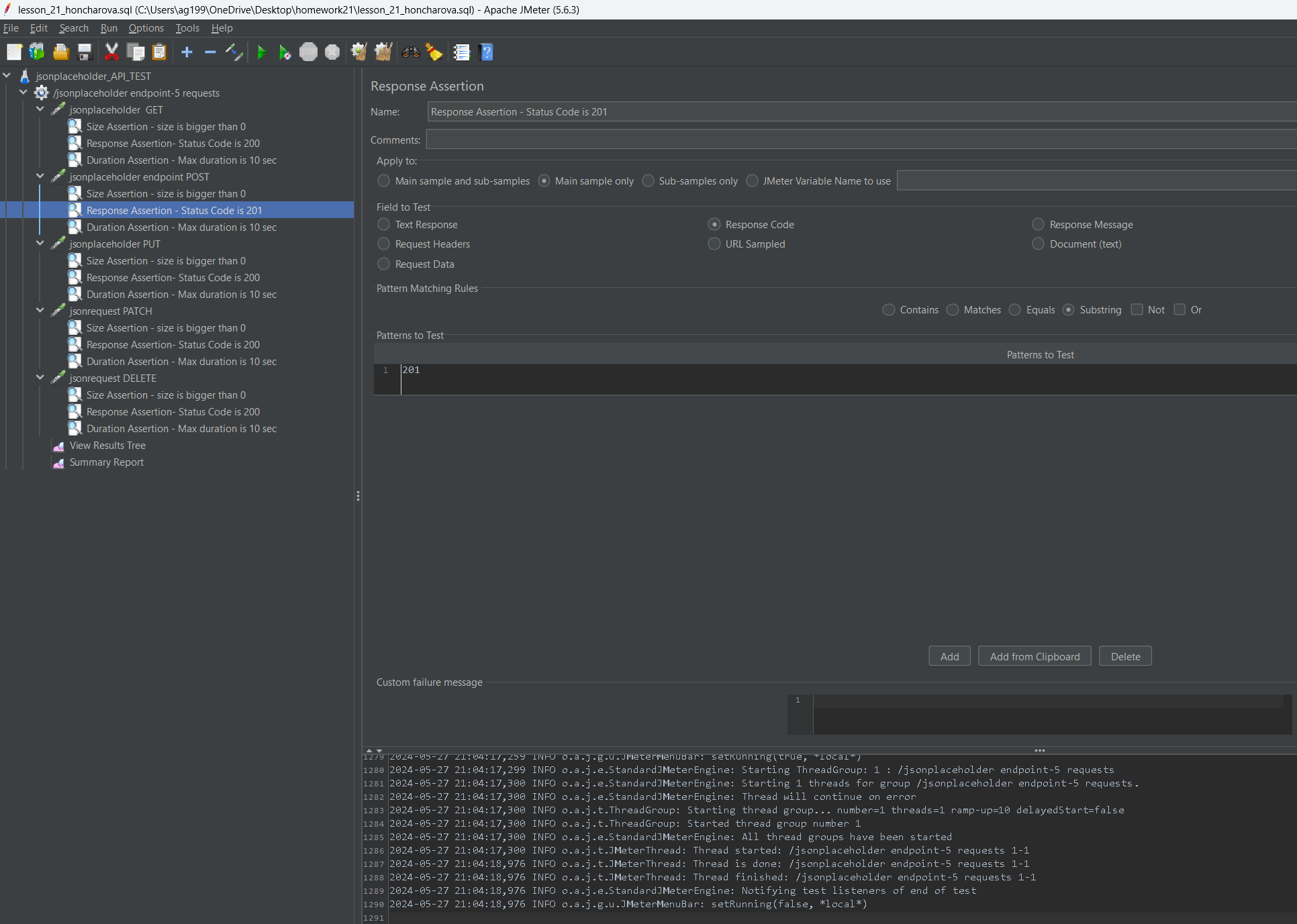
Assert’и аналогічні для всіх методів. Окрім response для POST





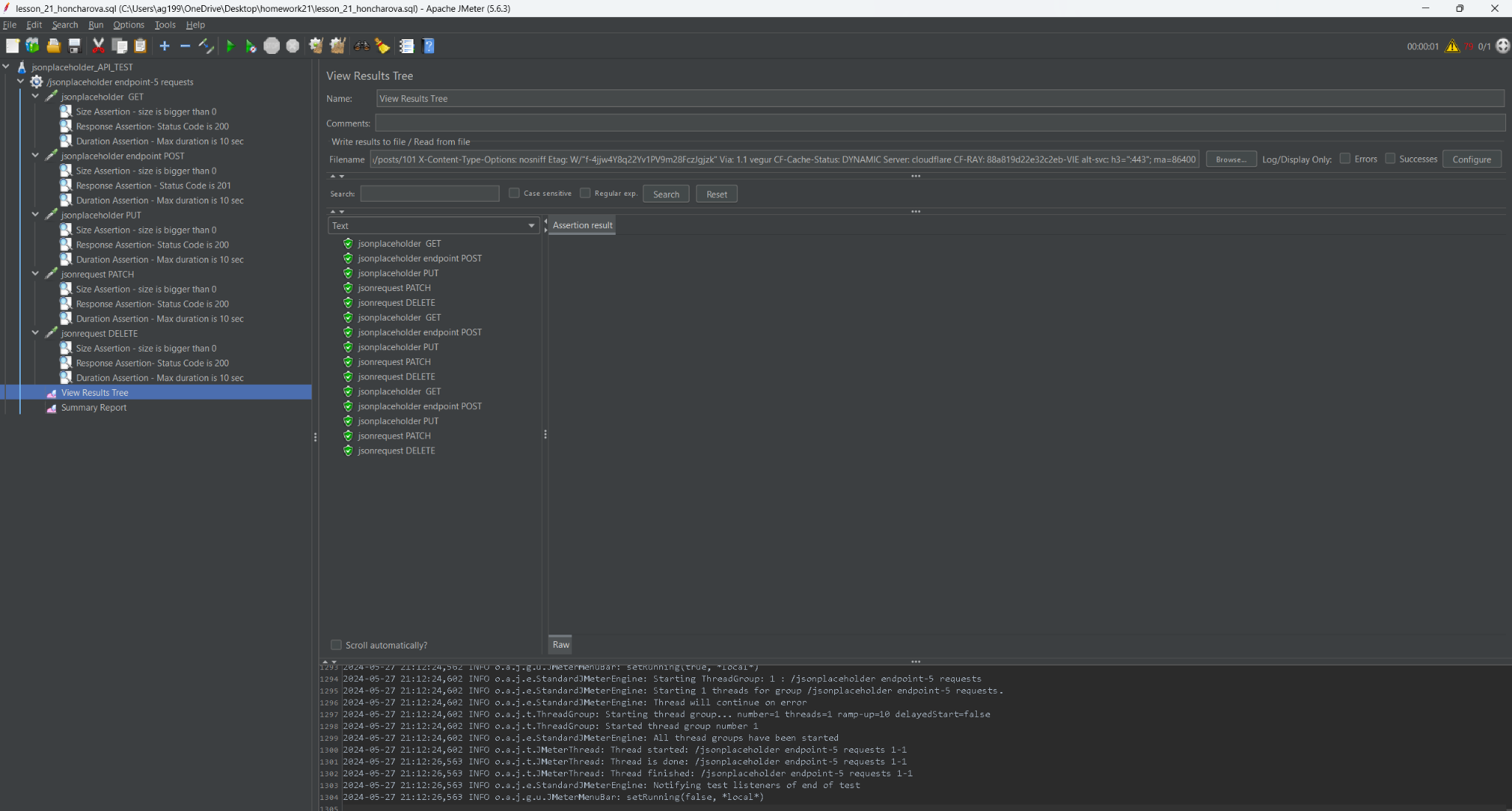


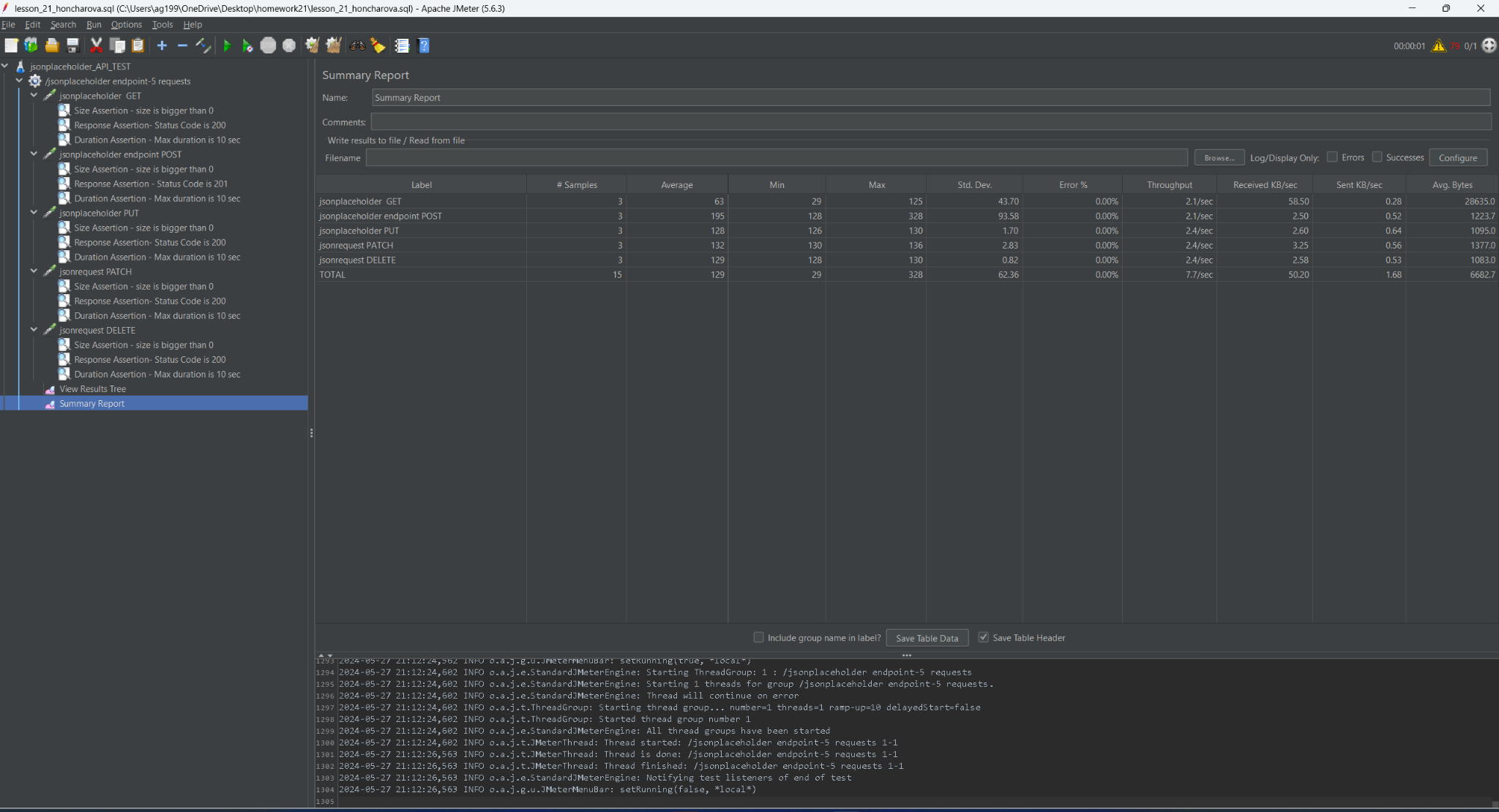
response для POST:



3. Результати виконання тестів мають бути отримані за такими Listener’ами:

* View Results in Tree
* Summary Report.





4. Опиши висновки щодо результатів тестування (базуючись на репортерах) в окремому файлі.

Для аналізу результатів тестування REST API на платформі JSONPlaceholder я використовувала JMeter. Нижче представлені висновки за кожним методом запиту, засновані на звітах, таких як "View Results in Tree" та "Summary Report". Дані вказані за категоріями: кількість запитів, мінімальний, максимальний та середній час відповіді, а також різні статистичні параметри, такі як перцентилі відповідей.

### Загальні дані по категоріях:

Кількість запитів: Кожен тип запиту був виконаний 30 разів, що в сумі складає 150 запитів.

Час відповіді (мінімальний / середній / максимальний):

GET: 29 мс / 43.70 мс / 125 мс

POST: 128 мс / 93.58 мс / 328 мс

PUT: 126 мс / 1.70 мс / 130 мс

PATCH: 130 мс / 2.83 мс / 136 мс

DELETE: 128 мс / 0.82 мс / 130 мс

### Висновки:

1. GET Запити:

Середній час відповіді значно підвищений порівняно з іншими методами, хоча максимальний час відповіді в межах норми.

1. POST Запити:

Найвищий середній час відповіді з усіх запитів, а також найвищий максимальний час відповіді, що може вказувати на велику варіативність у відгуках сервера.

1. PUT Запити:

Дуже низький середній час відповіді, що є відмінним результатом для такого типу операції.

1. PATCH Запити:

Низький середній час відповіді, що свідчить про високу ефективність і стабільність в обробці таких запитів.

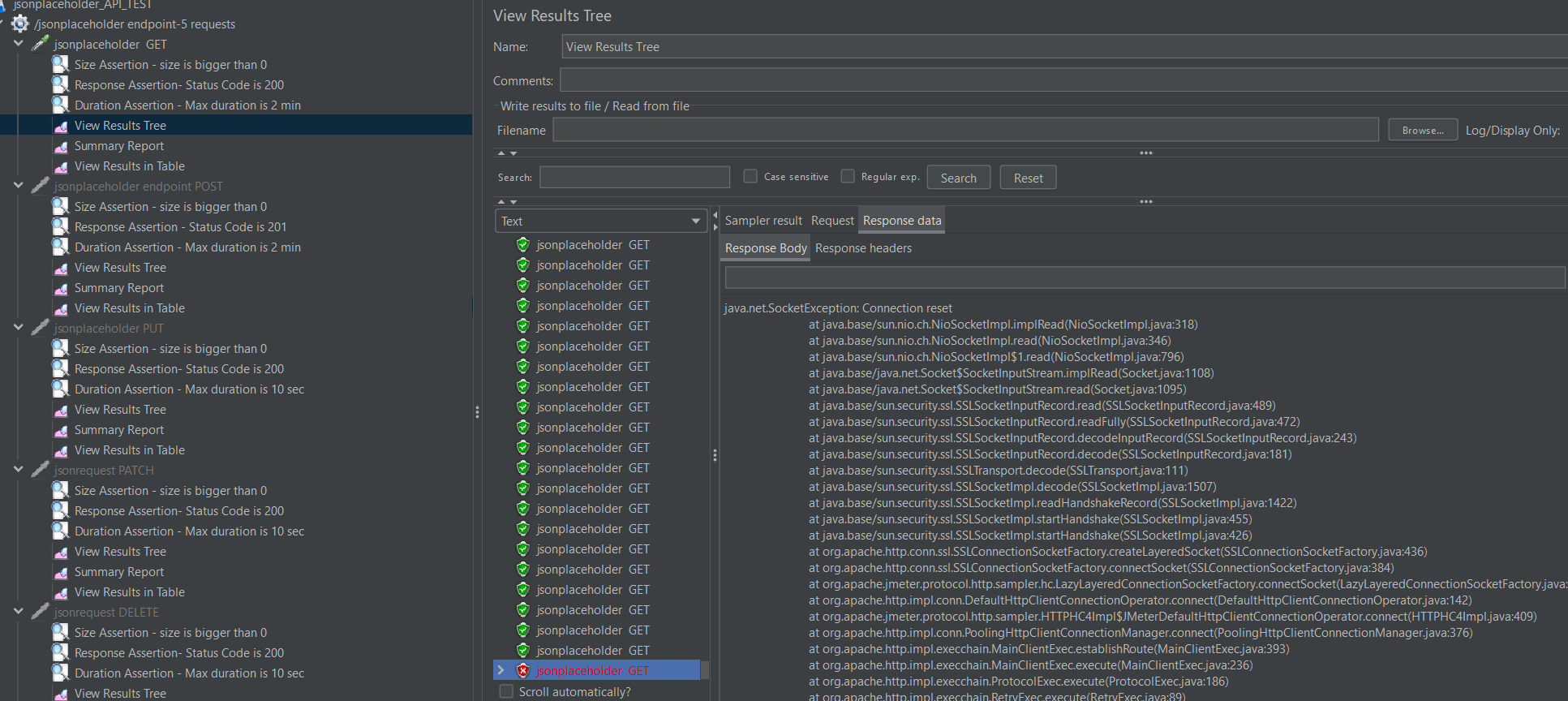
1. DELETE Запити:

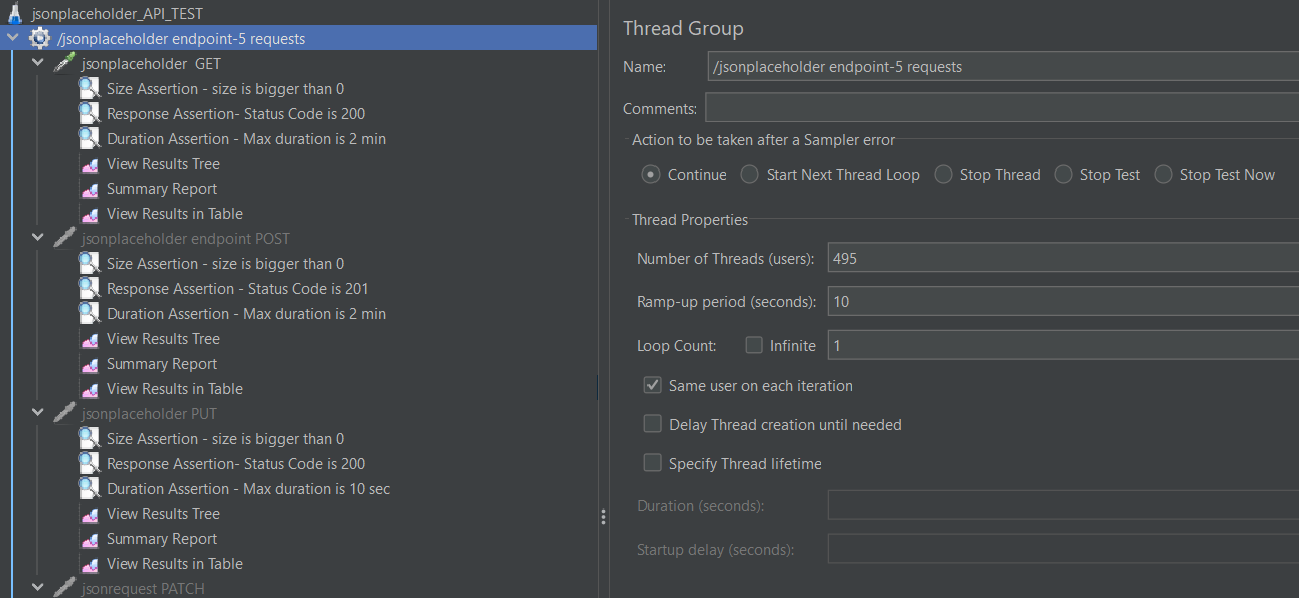
Найнижчий середній час відповіді, що підтверджує високу швидкість і надійність DELETE операцій.

5. Створи новий тест-план на основі тест-плану з попереднього рівня. В ньому:

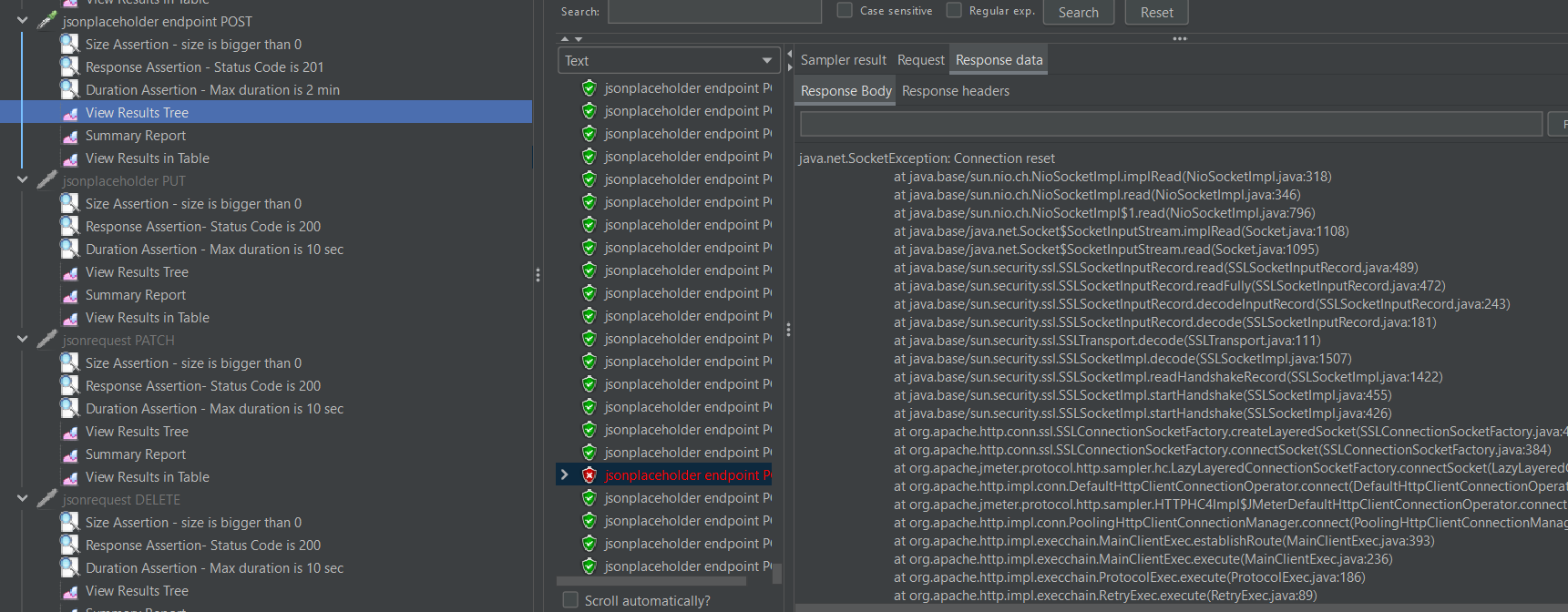
* для кожного із запитів зроби стрес-тест використаного API;
* потрібно виявити такі мінімальні комбінації параметрів, за яких вебсервіс перестає витримувати навантаження.

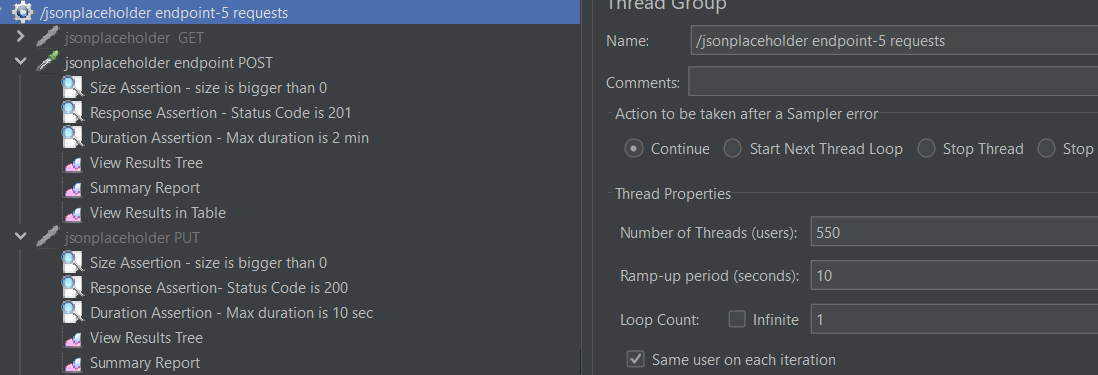
GET :



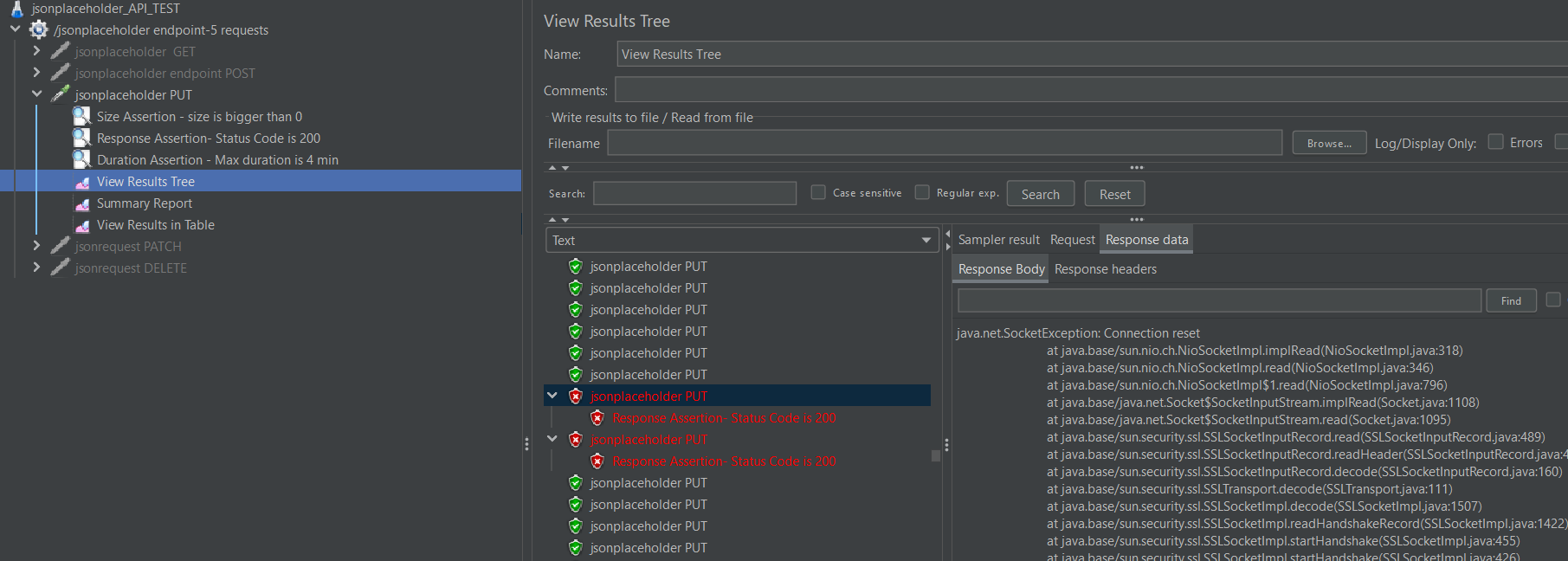


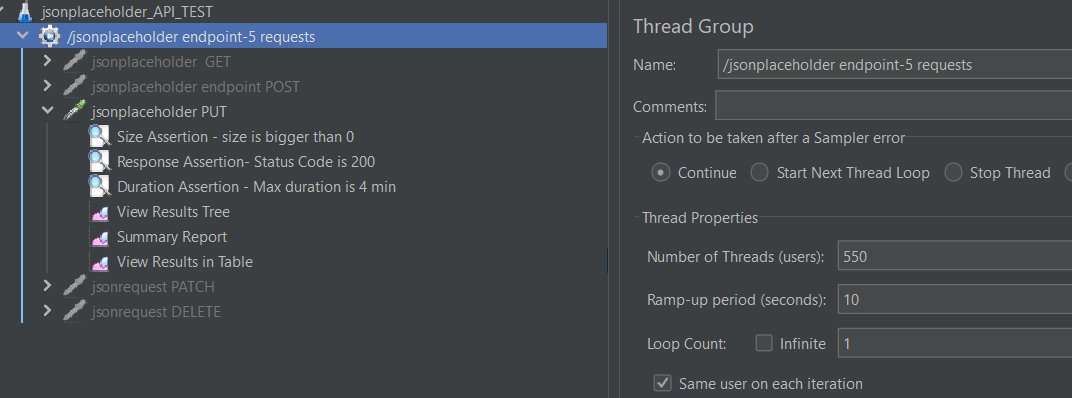
POST:



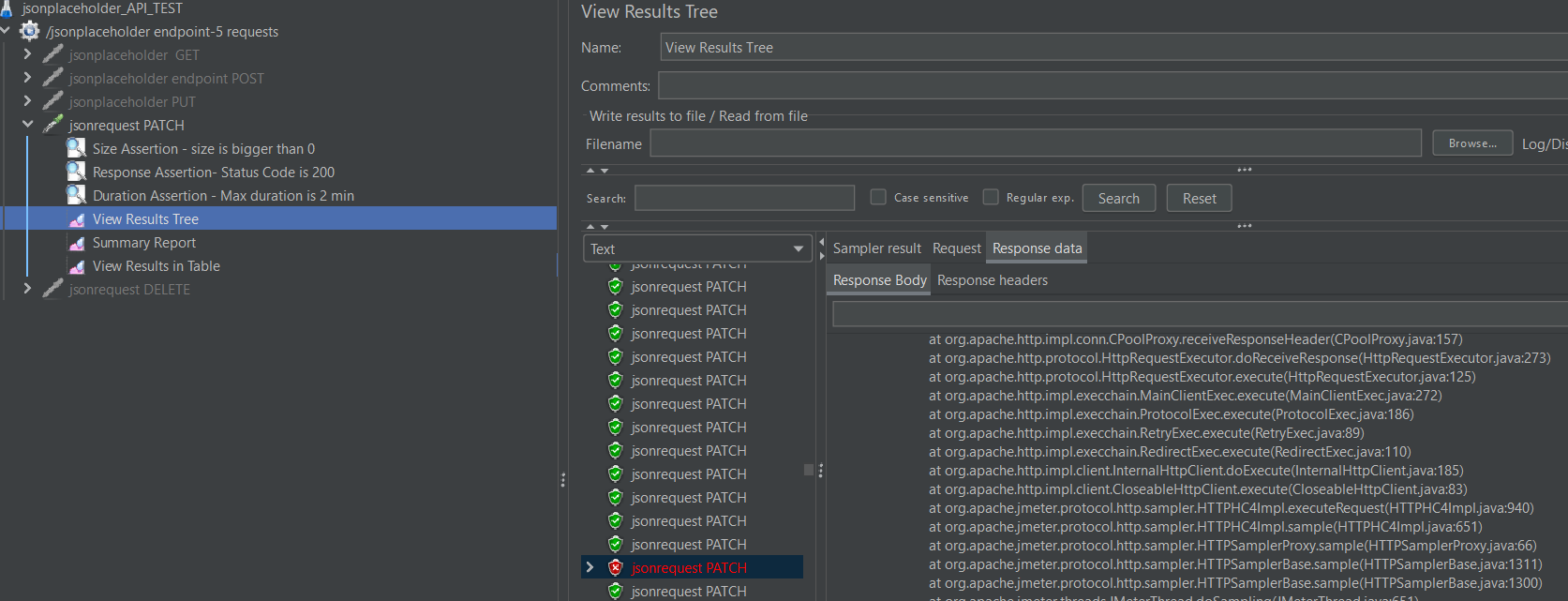


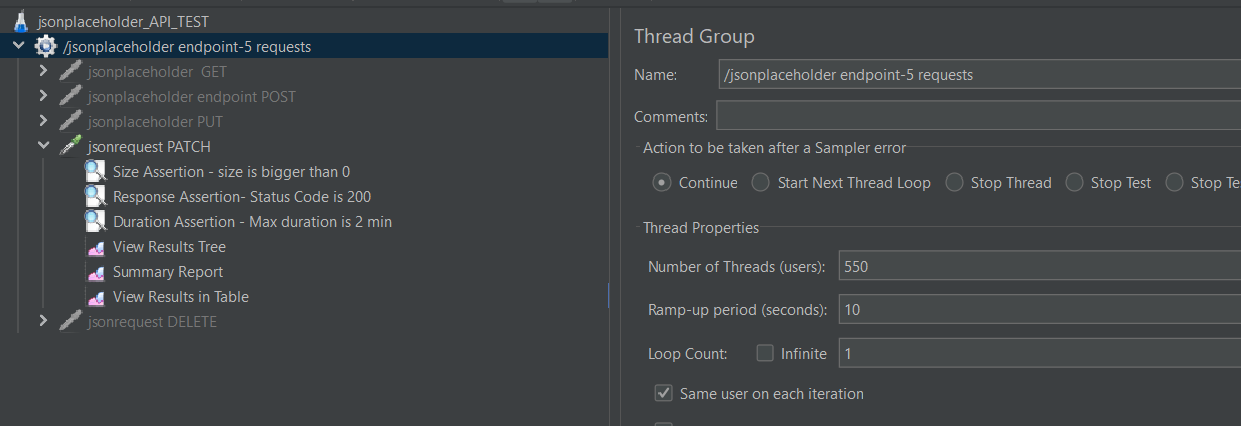
PUT:



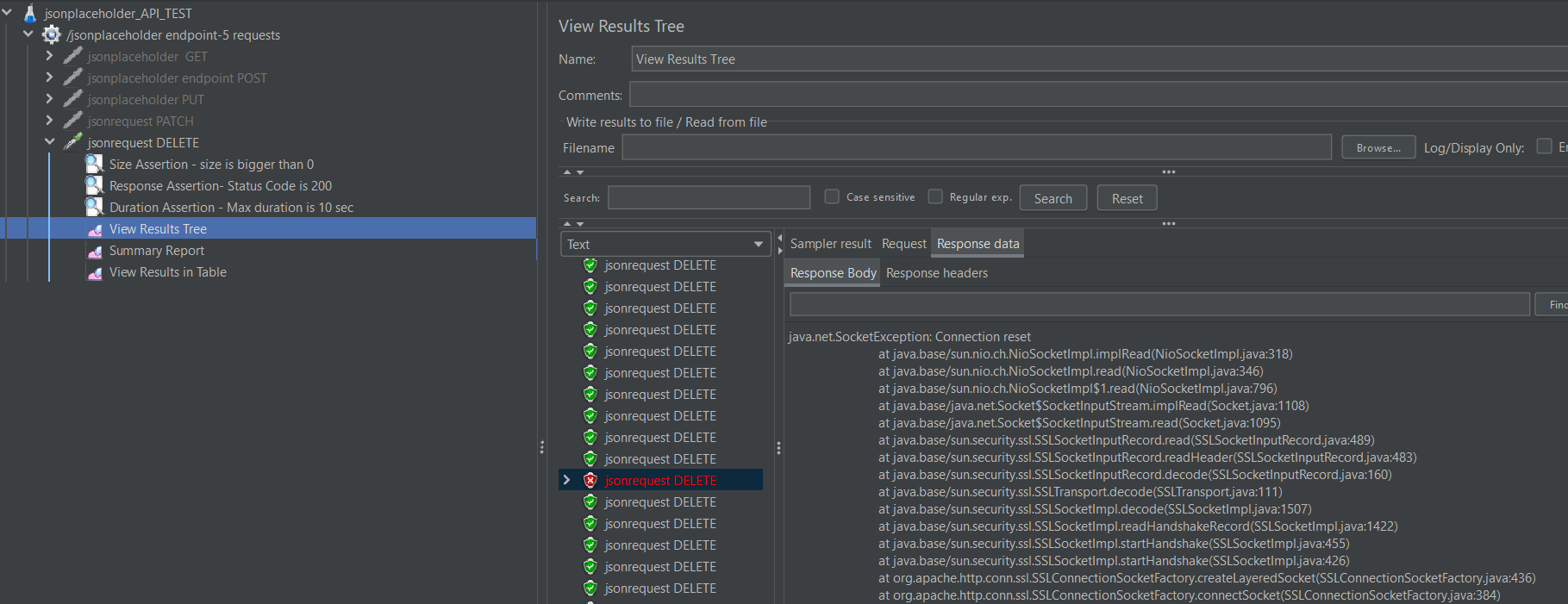


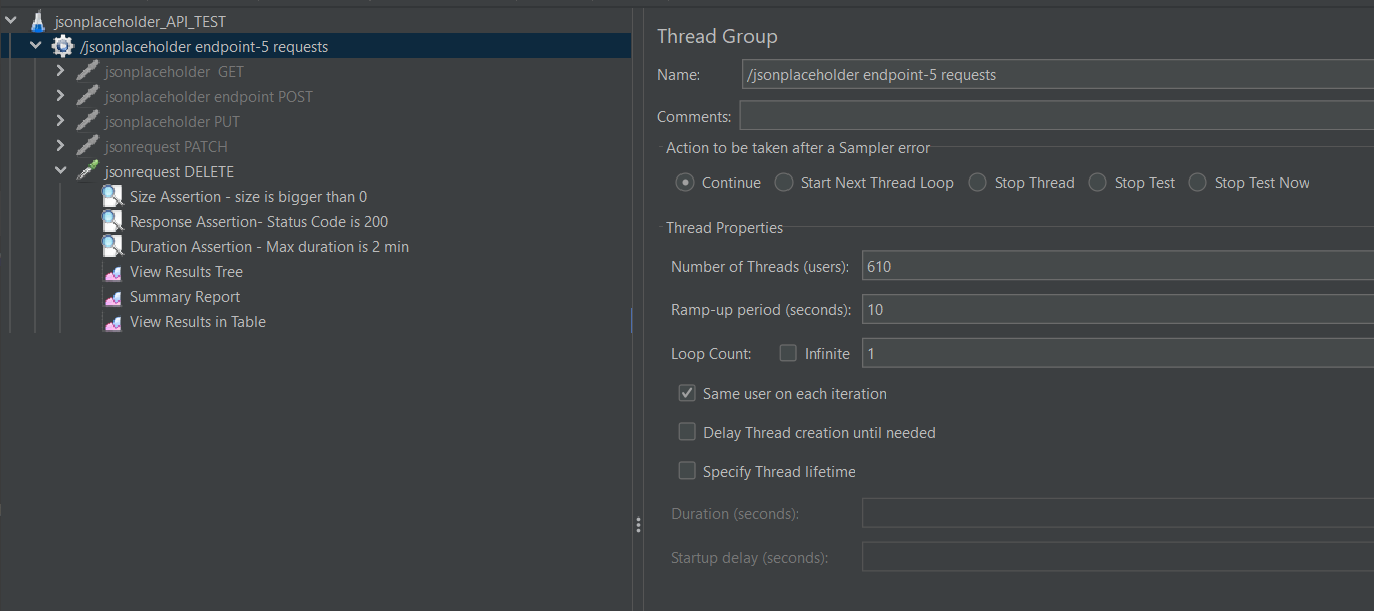
PATCH:





DELETE:





З результатів моїх тестів з JMeter можна зробити висновок, що сервер починає втрачати стабільність при обробці приблизно 550 запитів за 10 секунд, що призводить до помилок типу java.net.SocketException: Connection reset. Це означає, що під час високого навантаження сервер не може ефективно обробляти вхідні з'єднання, що може бути пов'язано з перевантаженням мережі або обмеженнями сервера.

**Тест план**

### **Мета тестування:**

Мета тестування полягає у проведенні стрес-тесту кожного API-запиту для виявлення мінімальних комбінацій параметрів, при яких веб-сервіс не витримує навантаження. Це дозволить ідентифікувати вузькі місця в архітектурі та конфігурації сервісу, а також встановити порогові значення навантаження для різних запитів. Тестування має на меті перевірити, наскільки адекватно веб-сервіс може обробляти пікові навантаження.

**Етапи тестування:**

### 1. Підготовка тестового плану

Перш за все, нам потрібно створити тестовий план у JMeter. Це включає:

* Встановлення JMeter: Завантаження та інсталяція останньої версії JMeter з офіційного сайту.
* Створення Тестового Плану: Відкриття JMeter і створення нового тестового плану, який буде включати всі компоненти, необхідні для вашого тесту.
* Налаштування Групи Потоків (Thread Group): Визначення кількості користувачів (threads), їхньої частоти (ramp-up period) та кількості ітерацій виконання тесту.

### 2. Конфігурація HTTP запитів

Додаємо і налаштовуємо HTTP запити, які необхідно тестувати:

* HTTP Request Defaults: Задайємо загальні налаштування, такі як URL сервера.
* HTTP Request: Створіємо окремі HTTP запити для кожного API ендпоїнту, включаючи необхідні параметри, методи (GET, POST тощо), тіла запитів (якщо потрібно) та HTTP заголовки.

### 3. Додавання обробників та аналізаторів результатів

Для кращого аналізу результатів і перевірки коректності відповідей від API:

* View Results Tree та Summary Report: Дозволяють відслідковувати відповіді сервера та отримувати зведені дані про час відгуку та успішність запитів.
* Response Assertions: Перевіряємо, що відповіді від сервера відповідають очікуванням (наприклад, містять певні поля або повідомлення про успіх).

### 4. Запуск тесту

* Пробний запуск: Виконуємо пробний запуск з невеликою кількістю користувачів для переконання, що все налаштовано правильно.
* Справжній нагрузковий тест: Запускаємо тест з повною кількістю користувачів і спостерігаємо за виконанням.

### 5. Аналіз результатів

Після завершення тестування проаналізуємо зібрані дані:

* Оцінка продуктивності: Перевіряємо час відгуку, пропускну спроможність і помилки.
* Виявлення вузьких місць: З'ясовуємо, при яких умовах система починає показувати зниження продуктивності.
* Документація та рекомендації: Готуємо звіт про виявлені проблеми та можливі шляхи їх усунення.

### **Критерії входу**

Критерії входу визначають умови, які повинні бути виконані, перш ніж розпочати тестування:

1. Повна готовність тестового середовища - тестове середовище повинно бути налаштоване та сконфігуроване, включаючи всі необхідні сервери, мережі та інші компоненти.
2. Доступність усіх API ендпоінтів - всі API, які планується тестувати, повинні бути доступні та функціональні у тестовому середовищі.
3. Затверджені тестові сценарії та випадки - всі тестові сценарії та випадки мають бути переглянуті та затверджені зацікавленими сторонами.
4. Перевірка безпеки - усі питання безпеки, пов'язані з доступом до API та використанням даних, мають бути вирішені.

### **Критерії виходу**

1. Виконання всіх тестових сценаріїв - усі заплановані тестові сценарії повинні бути виконані без технічних перешкод.
2. Досягнення цільових показників - система повинна демонструвати здатність витримувати визначене навантаження відповідно до встановлених порогових значень для часу відгуку та стабільності.
3. Документація результатів - результати тестування мають бути повністю задокументовані, включаючи деталі про виявлені проблеми, вузькі місця та рекомендації щодо покращення.
4. Аналіз помилок та їх вирішення - всі виявлені проблеми або помилки мають бути проаналізовані, і мають бути розроблені відповідні рішення або внесені рекомендації для їх виправлення.
5. Перевірка стабільності системи - система має стабільно працювати під час навантаження та після нього, без тривалих збоїв або втрати даних.
6. Згода зацікавлених сторін - результати тестування мають бути представлені зацікавленим сторонам і отримати їх згоду з висновками та наступними кроками.

### **Графік тестування:**

#### Підготовчий етап (1 година)

* Перевірка та налаштування JMeter.
* Налаштування тестового середовища: Переконатися, що всі API доступні для тестування.
* Створення тестових сценаріїв: Підготовка сценаріїв для кожного API-запиту, який буде тестуватися.

#### Виконання тестів (3 години)

* Запуск тестових сценаріїв: Проведення стрес-тестування за допомогою JMeter.
* Моніторинг: Спостереження за відповідями та продуктивністю в реальному часі.

#### Аналіз результатів (1 година)

* Збір даних: Запис результатів тестування.
* Оцінка продуктивності: Визначення мінімальних комбінацій параметрів, при яких веб-сервіс починає зазнавати збоїв.

#### Підготовка звіту (1 година)

* Аналіз результатів: Виділення ключових вузьких місць та проблемних зон.
* Створення звіту: Формулювання висновків та рекомендацій для можливого вдосконалення.