:росток:Beet Seed — відпрацюй навички на базовому рівні.

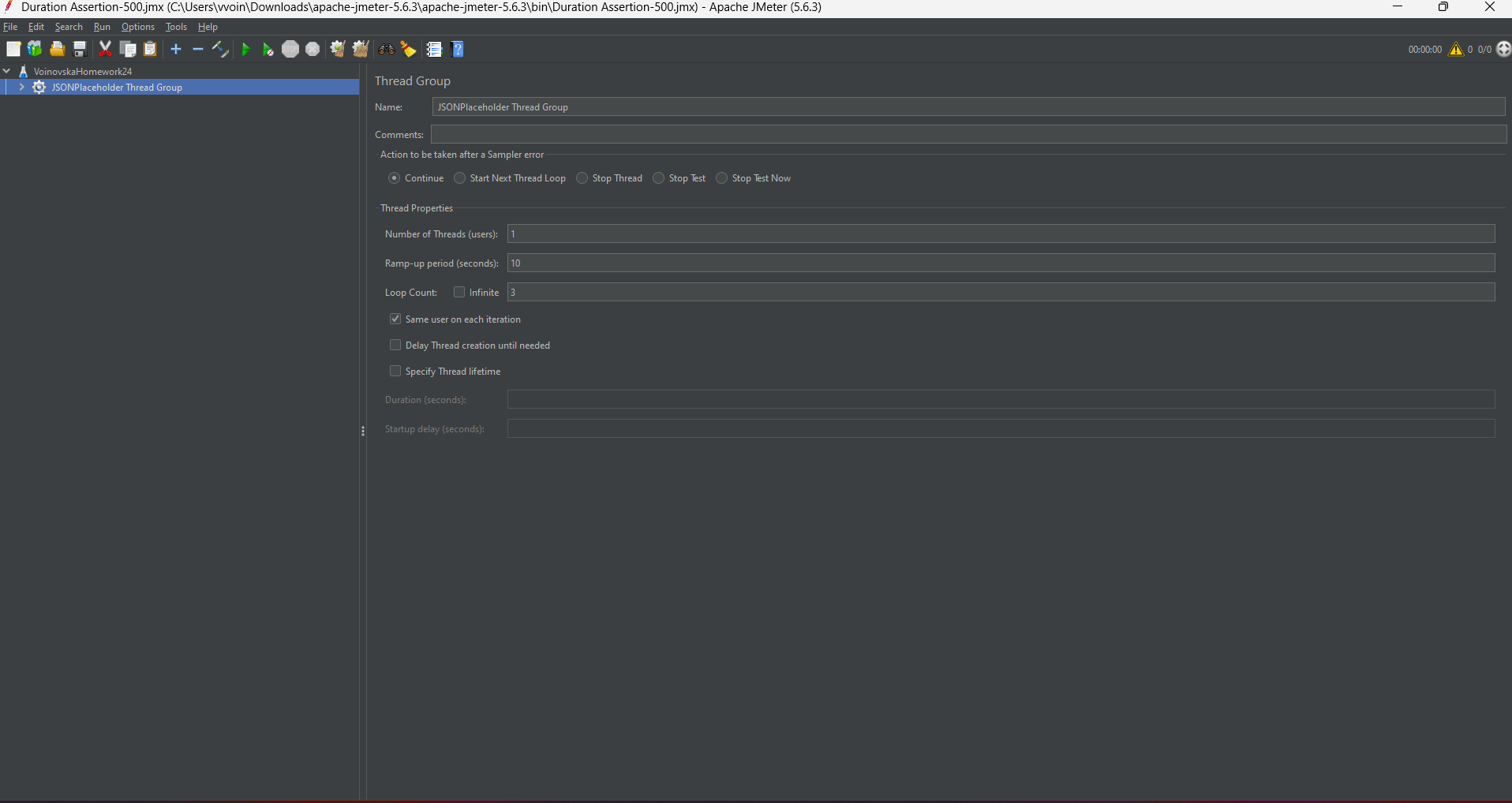
1. Створи набір з GET, POST, PUT, PATCH, DELETE запитів до [JSONPlaceholder](https://jsonplaceholder.typicode.com/), які надсилаються протягом 10 секунд у 3 ітерації.

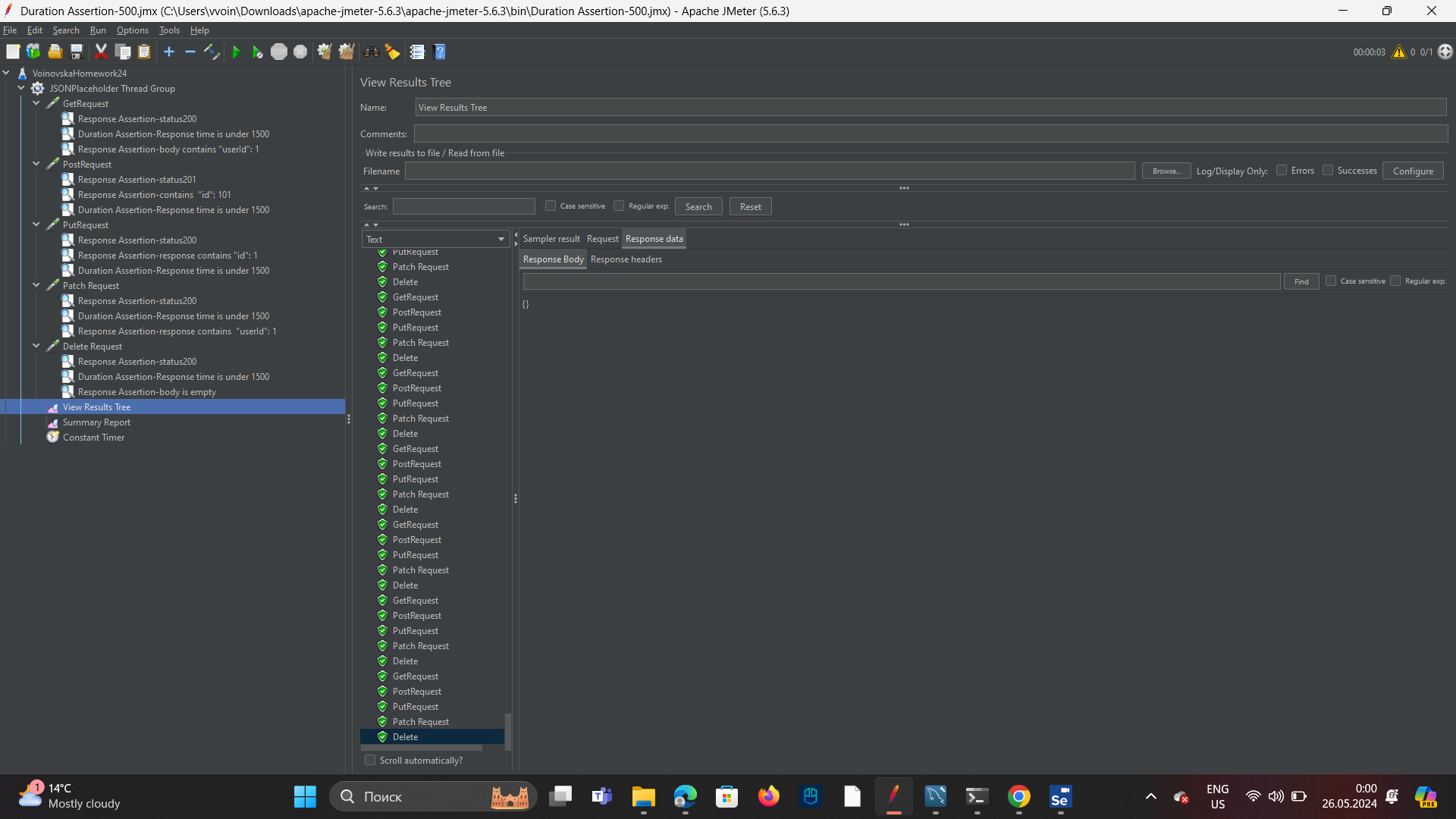
2. До кожного з запитів застосуй 3 різних assert’и.

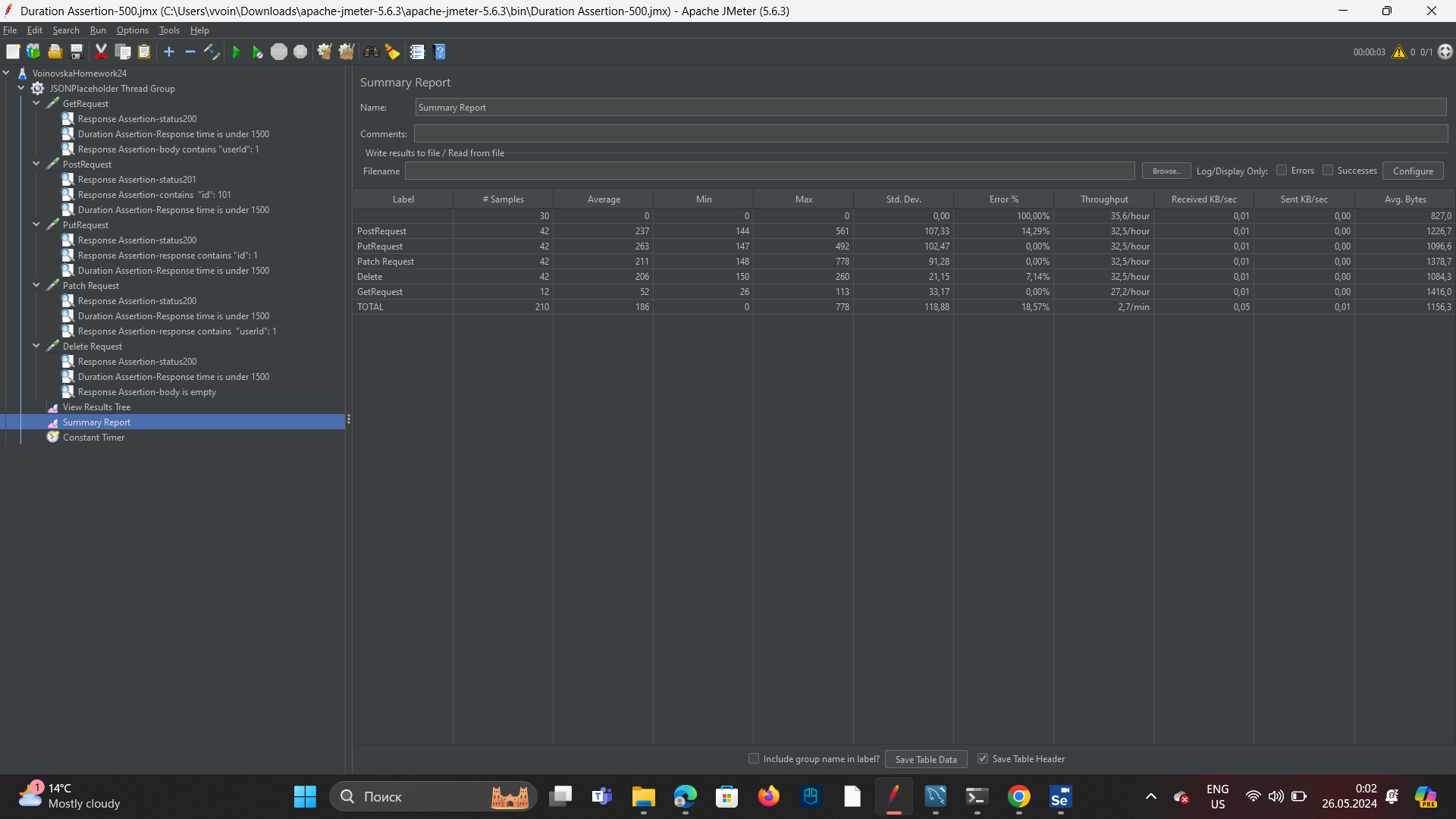
3. Результати виконання тестів мають бути отримані за такими Listener’ами:

View Results in Tree

Summary Report.







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

#### **GET Request**

* Запит: Отримання даних поста з id=1.
* Очікуваний статус код: 200.
* Реальний статус код: 200.
* Перевірка тіла відповіді: Відповідь містить ключ "userId" зі значенням 1.
* Висновок: Запит GET пройшов успішно

#### **POST Request**

* Запит: Створення нового поста з даними
* Очікуваний статус код: 201.
* Реальний статус код: 201.
* Висновок: Запит Post пройшов успішно.
* Перевірка тіла відповіді: Відповідь містить створений пост з id101.

#### **PUT Request**

* Запит: Оновлення поста з новими даними
* Очікуваний статус код: 200.
* Реальний статус код: 200.
* Висновок: Запит POST пройшов успішно.
* Перевірка тіла відповіді: Відповідь містить оновлені дані

#### **PATCH Request**

* Запит: Часткове оновлення поста "userId": 1 з новими даними
* Очікуваний статус код: 200.
* Реальний статус код: 200.
* Перевірка тіла відповіді: Відповідь містить оновлені дані

#### **DELETE Request**

* Запит: Видалення поста з "userId": 1
* Очікуваний статус код: 200.
* Реальний статус код: 200.
* Перевірка тіла відповіді: Відповідь порожня.

# **План тестування продуктивності**

**Мета**  Метою цього плану тестування продуктивності є оцінка продуктивності ендпоінтів API програми. Це тестування спрямоване на визначення ефективності обробки запитів

Кінцеві точки API виконують різні операції з даними, кожна з яких має свої специфічні вимоги та впливає на продуктивність системи:

* **GET:** Отримання даних з сервера.
* **DELETE:** Видалення ресурсів на сервері.
* **POST:** Створення нових ресурсів на сервері.
* **PUT:** Повне оновлення існуючих ресурсів.
* **PATCH:** Часткове оновлення існуючих ресурсів.

**Цілі** Виміряти час відповіді на кожен тип запиту (GET, POST, PUT, PATCH, DELETE) та визначити, як швидко сервер обробляє ці запити Встановити, чи залишається API стійким під час виконання серії запитів протягом 10 секунд у 3 ітерації, а також чи не виникає помилок або відмов у відповіді.Переконатися, що кожен тип запиту працює як очікувалося: GET запит повертає дані, POST створює новий ресурс, PUT та PATCH оновлюють ресурс, а DELETE видаляє ресурс.

### **Інструменти тестування**

#### JMeter

### **Тестововий сценарій**

Налаштування параметрів потоку роботи

Кількість користувачів(1), час розподілу навантаження (10), тривалість тесту (3).

### **Базова лінія**

При запуску тесту протягом 10 сек було перевірено дані про час відповіді

**Навантажувальне тестування**

Після того, як базові показники зібрані, запускається та сама симуляція, яка імітує профіль навантаження, зі збільшеною кількістю користувачів для тестування з цільовими обсягами. Ідея цього навантажувального тесту полягає в тому, щоб протестувати систему з типовим денним навантаженням, імітуючи підйоми, денні піки та падіння. 3 користувача за 10 сек

#### **Стрес-тестування**

#### Мета стрес-тестування полягає в тому, щоб знайти точку зламу системи, тобто в який момент система перестає реагувати. Якщо діє автоматичне масштабування, стрес-тест також буде хорошим індикатором того, в який момент система масштабується і додаються нові ресурси. Для стрес-тестування використовується та ж симуляція, яка використовується для навантажувального тестування, але з більш високим, ніж очікувалося, навантаженням.

#### 

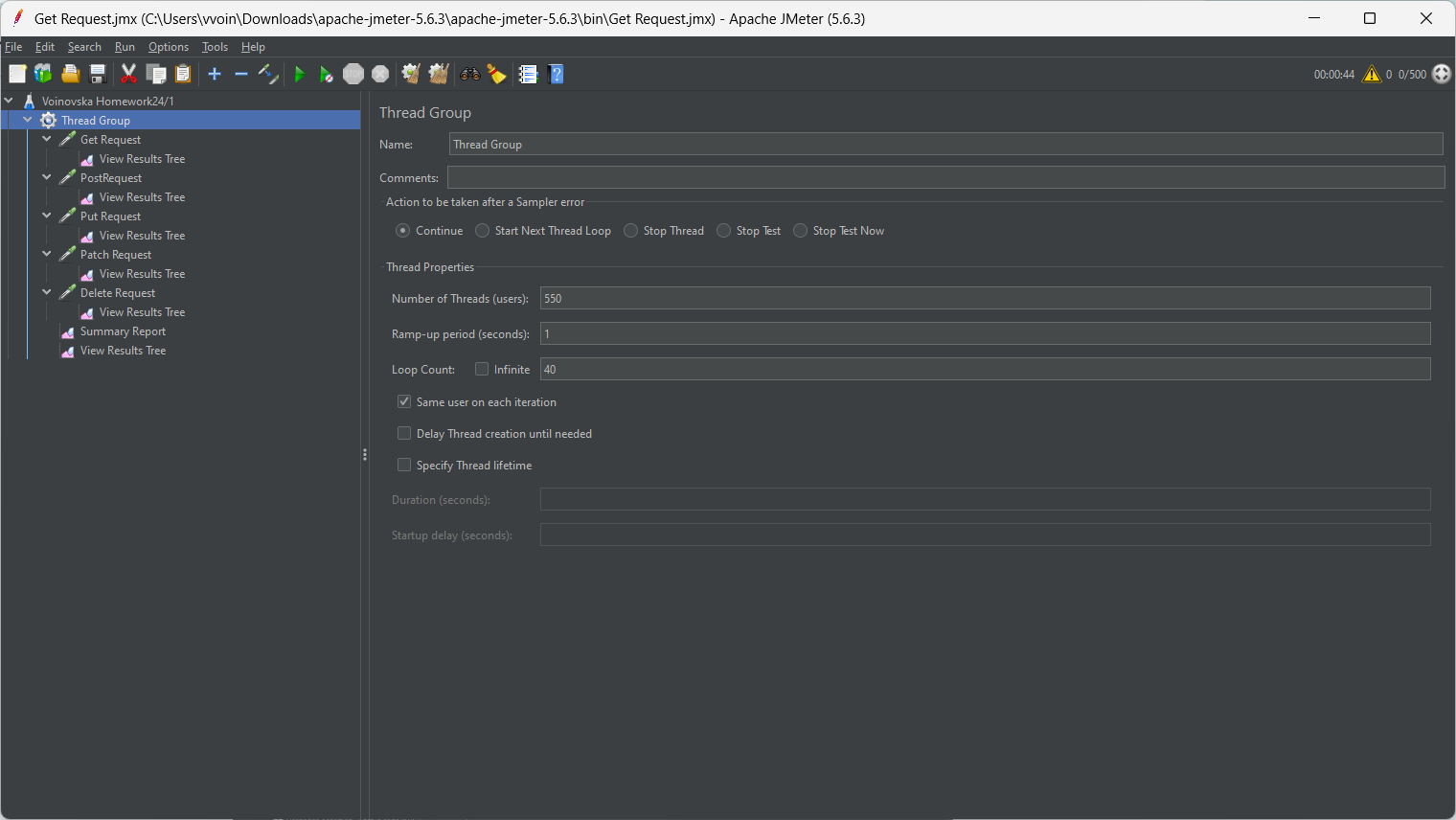
:травы: Beet Sprout — детальніше заглибся в практику.

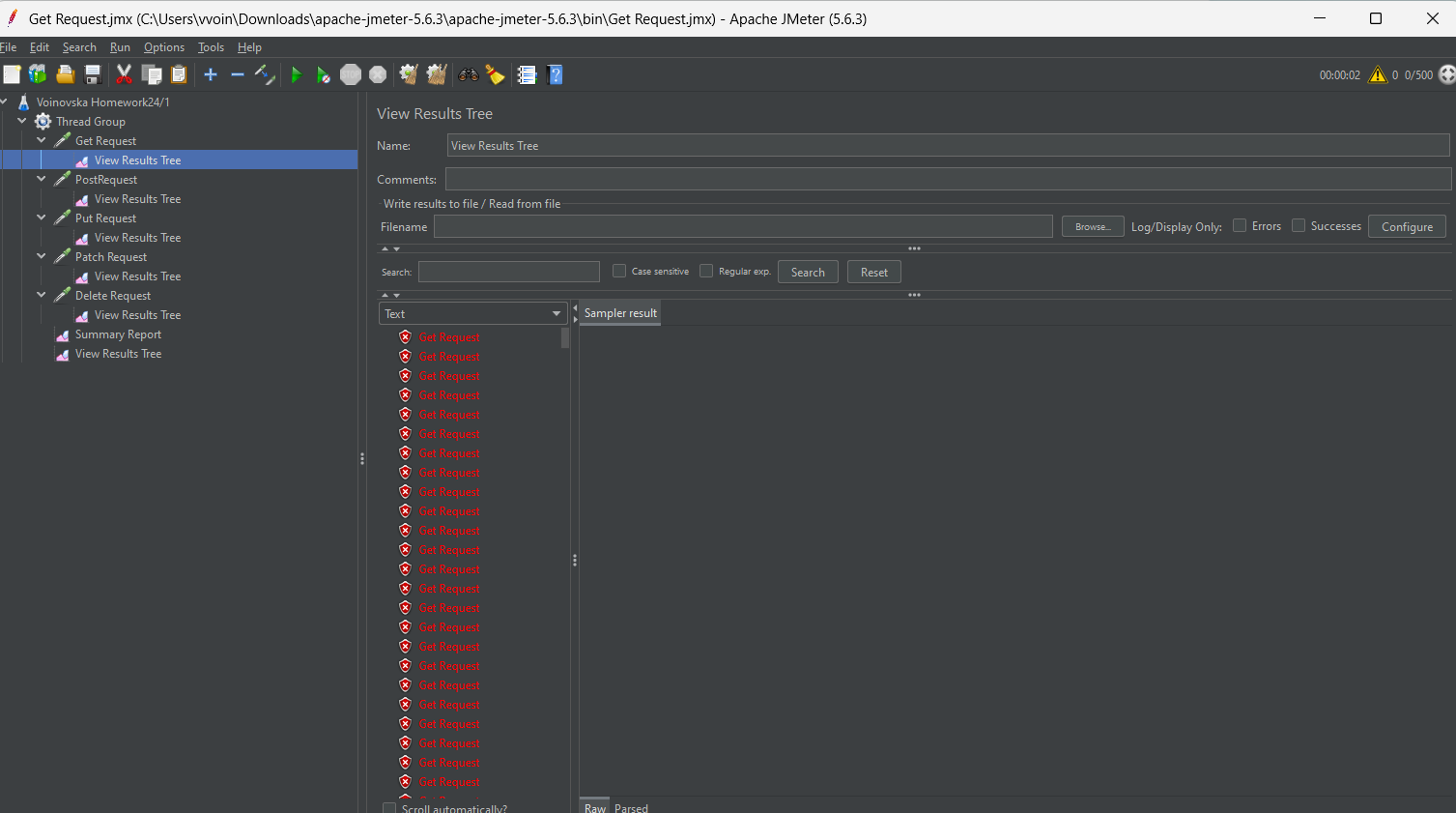
1. Виконай завдання попереднього рівня.

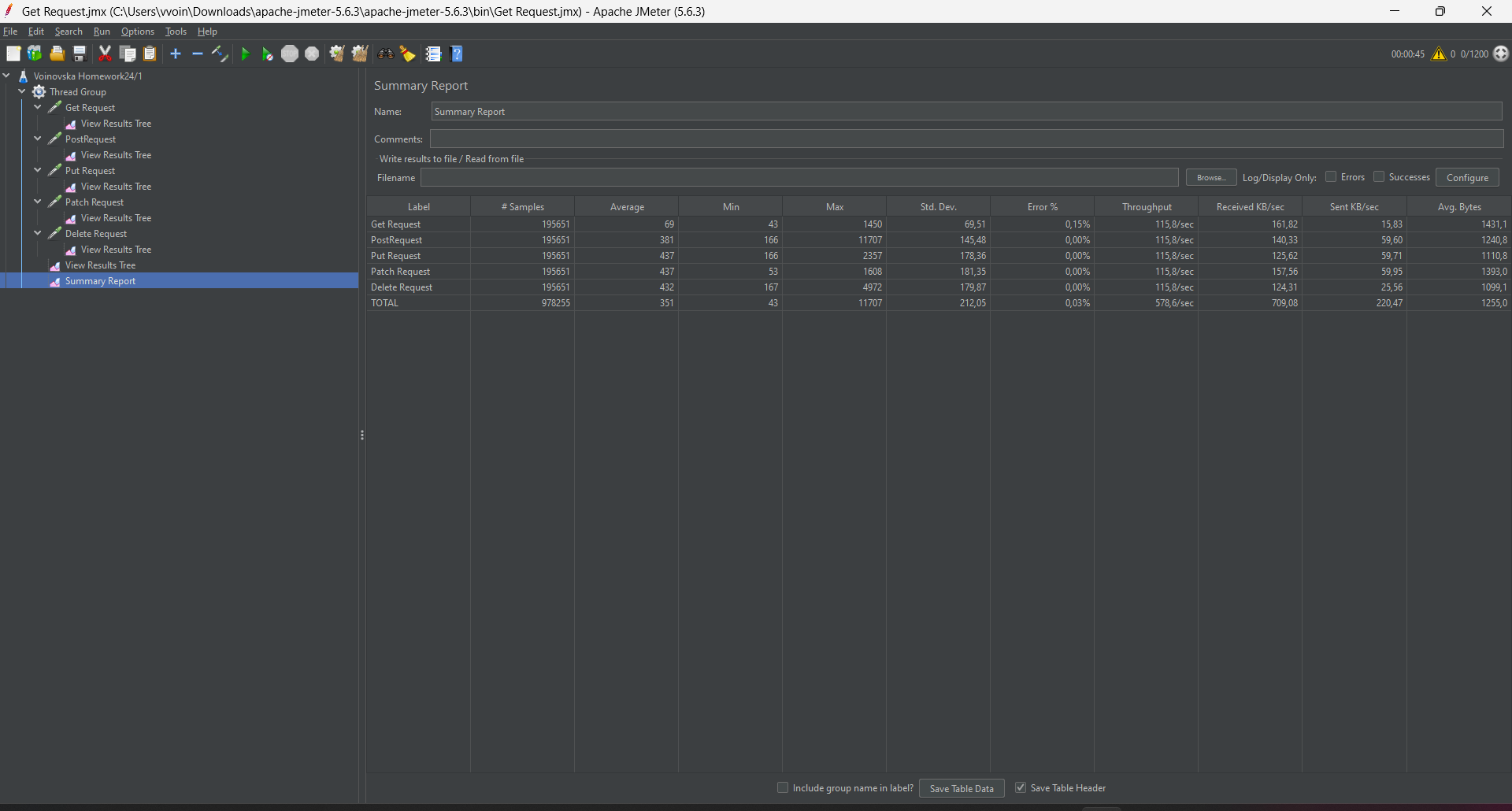
2. Створи новий тест-план на основі тест-плану з попереднього рівня. В ньому:

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

Вебсервіс починає демонструвати ознаки перевантаження при кількості потоків більше ніж 1100. Всі запити, окрім Get, обробляються без проблем до цього рівня навантаження.Запит Get починає мати неуспішні відповіді при кількості потоків 550 ,період нарощування секунд 1,кількість ітерацій 40







рверної обробки.