Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційних радіотехнологій та технічного захисту інформацій

Кафедра Радіотехнологій інформаційно-комунікаційних систем

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Розробка програм для навантажувального тестування сайту NURE
(тема)

Виконав:
студент II курсу, групи ІКТм -19-1
Іонов Г. П.
(прізвище, ініціали)

Спеціальність
122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми
освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Інформаційно-комунікаційні технології
(повна назва освітньої програми)

Керівник доцент Бітченко О.М.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри РТІКС _____ Цопа О.І.
(підпис) (прізвище, ініціали)

2020 р.

Не містить відомостей заборонених для відкритого публікування.

Студент Г.П. Іонов

Керівник О.М. Бітченко

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Інформаційних радіотехнологій та технічного захисту інформацій
Кафедра Радіотехнологій інформаційно-комунікаційних систем
Рівень вищої освіти другий (магістерський)
Спеціальність 122 Комп'ютерні науки
Тип програми Освітньо-професійна
Освітня програма Інформаційно-комунікаційні технології

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2020 р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові **ІОНОВУ Георгію Павловичу**
(прізвище, ім'я, по батькові)

1. Тема роботи **РОЗРОБКА ПРОГРАМ ДЛЯ НАВАНТАЖУВАЛЬНОГО ТЕСТУВАННЯ САЙТУ NURE**

затверджена наказом по університету від **2 листопада 2020 р. № 1506Ст**

2. Термін подання студентом проекту (роботи) **12 грудня 2020 р.**

3. Вихідні дані до проекту (роботи)
3.1 Працюючий сайт NURE
3.2 Рекомендований час відгуку 2...3 секунди
3.3 Кількість віртуальних користувачів не більше 50
3.4 Розмір пачки у одному скрипті не більше п'яти

4. Перелік питань, що потрібно опрацювати в роботі
Реферат. Перелік умовних позначень, символів, одиниць, скорочень і термінів. Вступ. 4.1 Огляд видів навантажувального тестування. 4.2 Огляд програмних продуктів для проведення тестування. 4.3 Концепція системи. 4.4 Аналіз результатів тестування.. Висновки. Перелік посилань. Додатки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)

Комп'ютерна презентація

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по-батькові	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	доц. Бітченко Олександр Миколайович		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Вступ	5.09-10.09	Виконано
2	Огляд видів навантажувального тестування	11.09-20.09	Виконано
3	Огляд програмних продуктів для проведення тестування	21.09-30.09	Виконано
4	Концепція системи	1.10-20.10	Виконано
5	Аналіз результатів тестування	21.10-11.11	Виконано
6	Реферат	12.11-15.11	Виконано
7	Перелік умовних позначень, символів, одиниць, скорочень і термінів	16.11-17.11	Виконано
8	Висновки	18.11-20.11	Виконано
9	Оформлення пояснювальної записки	20.11-30.11	Виконано
10	Оформлення презентації	1.12-11.12	Виконано
11	Подання роботи на кафедру	12.12.2019	Виконано

Дата видачі завдання **4 вересня 2020 р**

Студент _____
(підпис)

Керівник роботи_____ доц. Бітченко О.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи містить 97 сторінок тексту, 58 рисунків, 11 джерел та 3 додатки.

Мета роботи: тестування навантажувальної здатності офіційного сайту Харківського національного університету радіоелектроніки.

НАВАНТАЖЕННЯ, ТЕСТУВАННЯ, ВІДГУК СИСТЕМИ,
СКРИПТ, САЙТ

У результаті роботи були розглянуті різні види тестування і вивчені методи побудови тестів. За результатами дослідження був проведений огляд основних методів і типів тестування та найбільш відомих інструментів для автоматизації навантажувального тестування. Були виявлені і виправлені типові помилки, з якими може зіткнутися тестувальник під час тестування реально працюючої системи.

Під час роботи над атестаційною роботою були розроблені та налагоджені 9 програм навантажувального тестування. Був проведений аналіз роботи програм, та аналіз поведінки сайту NURE під час роботи під навантаженням. Також були зроблені висновки щодо навантажувальної здатності сайту NURE.

ABSTRACT

The explanatory note of the attestation work contains 97 pages of text, 58 figures, 11 sources and 3 appendices.

Purpose: testing the load capacity of the official website of Kharkiv National University of Radio Electronics.

LOADING, TESTING, SYSTEM RESPONSE,
SCRIPT, SITE

As a result of work various types of testing were considered and methods of construction of tests were studied. According to the results of the study, a review of the main methods and types of testing and the most well-known tools for automation of stress testing was conducted. The typographical errors that the tester may encounter when testing a real-world system have been identified and corrected.

During the work on the attestation work, 9 load testing programs were developed and adjusted. An analysis of the robots of the programs and an analysis of the behavior of the NURE site during work under load was performed. Conclusions were also made regarding the load capacity of the NURE site.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Скрипт – програма навантажувального тестування.

UC (Use Case) – програма/скрипт навантажувального тестування.

VUSer (Virtual User) – віртуальний користувач, призначений для імітації роботи реального користувача, має заздалегідь завдану поведінку.

Transaction – транзакція, атомарна дія у рамках навантажувального скрипту.

TPS (Transaction per second) – транзакції за секунду, універсальний метод виміряння подаваємого навантаження.

PACING – затримка між викликами скриптів під час роботи сценарію.

Модуль – сукупність файлів додатку, пов'язаних спільним призначенням та використанням, що використовують однакові або дуже схожі алгоритми взаємодії між собою або даними.

Веб-сервер – це сервер, що приймає HTTP-запити від клієнтів, зазвичай веб-браузерів, видає їм HTTP-відповіді, зазвичай разом з HTML-сторінкою, зображенням, файлом, медіа-потоком або іншими даними.

Веб-клієнт – це браузер який отримує інформацію з серверу і з яким взаємодіє юзер.

Параметризація – данні які неможливо отримати з запиту, такі данні потрібно генерувати.

Кореляція – данні отримані з запиту і записані в параметр.

ВСТУП

У сучасному світі цифрових технологій тестування стало невід'ємною частиною життєвого циклу будь-якого програмного продукту, будь-то призначений для користувача інтерфейс або критична система.

Оскільки програмування, як і будь-яка людська діяльність неможлива без помилок, важливо правильно виставляти вимоги до програмного продукту, планування етапів його розробки та мети тестування.

В етапи розробки програмного забезпечення входять:

- збір та аналіз вимог;
- системний аналіз;
- розробка системи;
- кодування;
- тестування;
- реалізація.

На етапі аналізу вимог до програмному продукту пишуться тест-плани, розробляються тестові набори, а також оцінюється необхідність використання автоматизації тестування. Як і будь-який інший складний процес, тестування програмного забезпечення складається з різних етапів, і кожен з них представлений конкретною спрямованістю дій.

Основною метою тестування є пошук недоліків в специфікаціях, а також пошук фактичних помилок програміста. Також один з важливих етапів перевірки якості програмного продукту, це верифікація - процес оцінки проміжних робочих продуктів життєвого циклу розробки, для перевірки чи правильно створений кінцевий продукт.

Метою даної атестаційної роботи є тестування навантажувальної здатності офіційного сайту Харківського національного університету радіоелектроніки.

Для досягнення поставленої мети в роботі сформульовані наступні задачі:

- проаналізувати сучасні засоби навантажувального тестування;
- провести огляд програмних продуктів для тестування сайтів;

– розробити програми навантажувального тестування:

a) програму імітуючу дії користувача при зміні мови на сайті;

b) програма імітуючу дії користувача який використовує пошук по сайту NURE;

c) програма імітуючу дії користувача при послідовному переході користувача з головної сторінки на вкладку UNIVERSITY і далі на вкладку ABOUT_UNIVERSITY;

d) програма імітуючу дії користувача при послідовному переході користувача з головної сторінки на вкладку APPLICANTS і далі на вкладку EDUCATION_IN_ENGLISH;

e) програма імітуючу дії користувача при послідовному переході користувача з головної сторінки на вкладку STUDENTS і далі на вкладку Timetable_of_Classes для перегляду розкладу;

f) програма імітуючу дії користувача при послідовному переході користувача з головної сторінки на вкладку SCIENCE і далі на вкладку SCIENCE_Topic;

g) програма імітуючу дії користувача при послідовному переході користувача з головної сторінки на вкладку EDUCATION і далі на вкладку SCIENTIFIC_LIBRARY;

h) програма імітуючу дії користувача при послідовному переході користувача з головної сторінки на вкладку PRESS_CENTER і далі на вкладку PRESS_SERVICE;

i) програма імітуючу дії користувача при послідовному переході користувача з головної сторінки на вкладку CONTACTS і далі на вкладку ABOUT_UNIVERSITY;

– проаналізувати отримані результати тестування;

– зробити висновки за результатами тестування.

1 ОГЛЯД ВИДІВ НАВАНТАЖУВАЛЬНОГО ТЕСТУВАННЯ

1.1 Класифікація видів тестування

На даний момент існує безліч видів тестування, а також існує велика кількість класифікацій цих видів. Основна класифікація видів тестування відбувається за метою, яка наведена на рисунку 1.1.

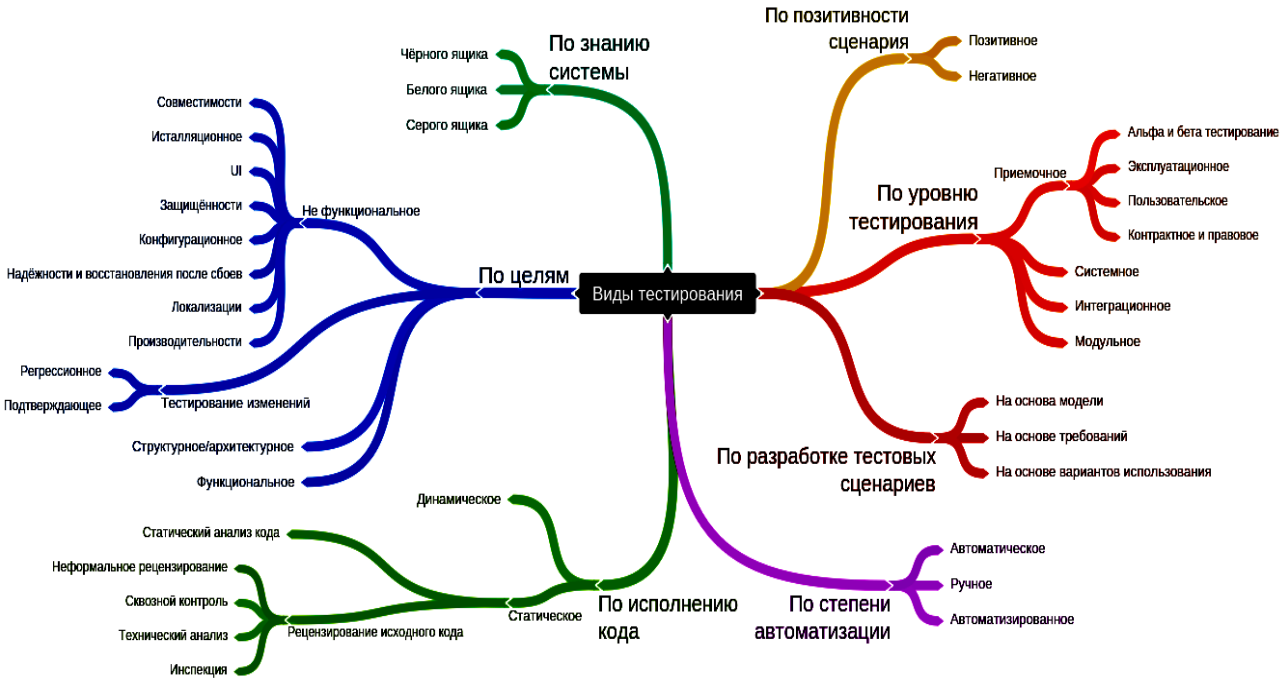


Рисунок 1.1 – Класифікація видів тестування за метою

Класифікацію тестування можна продовжувати і далі, розділяючи на ще більш атомарні одиниці дані типи тестування. Основна складність при класифікації видів тестування полягає в тім, що в основному всі стандарти розроблені англійською мовою, і при перекладі термінів можуть виникати колізії взаємозалежності від того, як той або інший термін буде переведений на українську мову. У зв'язку з цим, окрім україномовного перекладу виду тестування в скобках буде вказуватися його англійське найменування. У рамках даної роботи особлива увага звернута на тестування продуктивності (Performance Testing) [4].

1.2 Тестування продуктивності (Performance Testing)

Вважається, що тестування продуктивності – це те тестування, що не є функціональним. Існує безліч видів тестування продуктивності. Класифікація видів тестування продуктивності будується на основі того, яку мету переслідує визначений вид тестування. Як правило, тестування продуктивності переслідує не одну мету, а декілька. У зв'язку з цим, багато типів тестування в ході його проведення сполучаються з іншою метою або повторюються кілька разів у ході циклу тестування.

Основна відмінність тестування продуктивності також полягає в тім, що воно відбувається тільки після повного функціонального тестування. Помилки функціональності не виправляються в ході тестування продуктивності. Для даного виду тестування найчастіше виділяється окремий навантажувальний стенд, що повторює копію промислового стенда. У зв'язку з масовим поширенням Agile методологій, тестування продуктивності також інтегрується в життєвий цикл розробки програмного забезпечення [4].

На рисунку 1.2 показана основна класифікація видів тестування продуктивності.

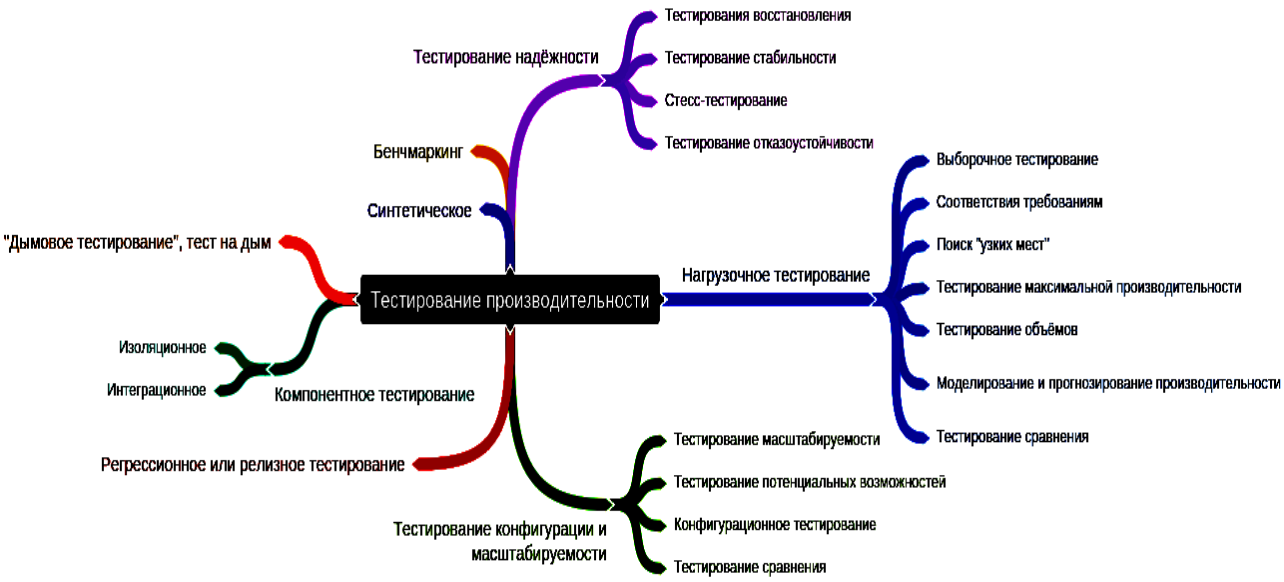


Рисунок 1.2 – Основна класифікація видів тестування продуктивності

На основі класифікації, наведеній на рисунку вище можна виділити основні види тестування, що здійснюються під час етапу тестування продуктивності системи. Основні типи тестування і питання, які вони вирішують наведено в таблиці 1.2.

Таблиця 1.2 – Основні типи тестування

Вид тестування	Питання, на які відповідає тестування
Навантажувальне тестування (Load Testing)	Чи досить швидко працює система?
Тестування стабільності (Stability Testing)	Чи досить надійно працює система на довгому інтервалі часу?
Тестування відмовостійкості (Failover Testing)	Чи зможе система перемістится сама на інший сервер у випадку збою основного сервера?
Тестування відновлення (Recovery Testing)	Як швидко відновиться система?
Стресове тестування (Stress Testing)	Що відбудеться при незапланованому навантаженні?
Тестування обсягів (Volume Testing)	Як буде працювати система, якщо обсяг бази даних калічиться в 100 разів?
Тестування масштабованості (Scalability Testing)	Як буде збільшитися навантаження на компоненти системи при збільшенні числа користувачів?
Тестування потенційних можливостей (Capacity Testing)	Яка кількість користувачів може працювати?
Конфігураційне тестування (Configuration Testin)	Як змусити систему працювати швидше?
Тестування порівняння (Compare Testing)	Яке устаткування і ПО вибрати?

1.3 Навантажувальне тестування (load testing)

Навантажувальне тестування (load testing) – даний тип тестування дозволяє оцінити поводження системи при зростаючому навантаженні, метою навантажувального тестування є також визначення максимального навантаження, що може витримати система (рисунок 1.3) [3].

Це найбільш розповсюджений і відомий тип тестування. Часто в рунеті, особливо ті, хто не занурені в тематику QA, під навантажувальним тестуванням розуміють усі види іспитів. Але, в англомовній літературі, це усього лише підвид тестування продуктивності.

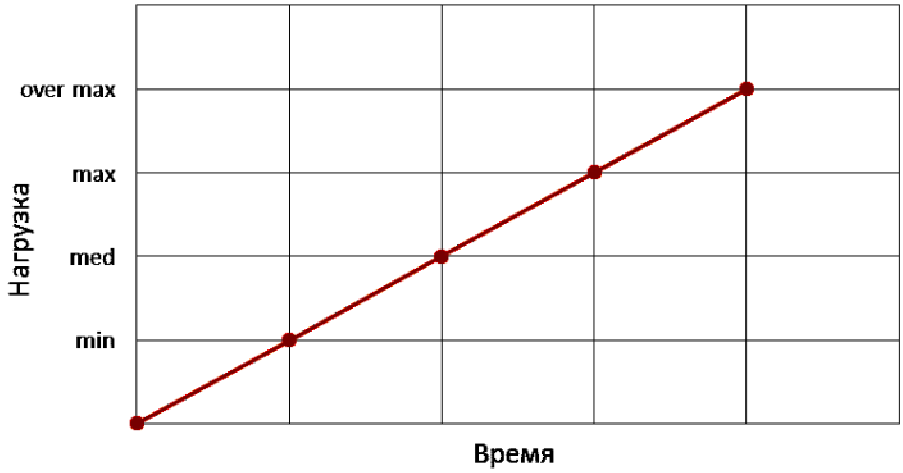


Рисунок 1.3 – Вигляд характеристики навантажувального тестування

У ролі навантаження може виступати кількість користувачів, а також кількість операцій на сервері.

Продуктивність при цьому визначається наступними факторами:

- швидкістю роботи програмного забезпечення;
- швидкістю роботи апаратного забезпечення;
- швидкістю роботи мережі.

Під час тестування можуть здійснюватися наступні операції, що дозволяють більш точно вимірювати продуктивність і визначити “вузьке місце” системи:

– вимір часу виконання обраних операцій при визначених інтенсивностях виконання цих операцій;

– визначення кількості користувачів, що одночасно працюють з додатком;

– визначення границь прийнятної продуктивності при збільшенні навантаження (при збільшенні інтенсивності виконання цих операцій).

Після перебування максимальної продуктивності рекомендується її “підтвердити”. Для цього проводиться додатковий тест із наступним профілем (рисунок 1.4) [3].

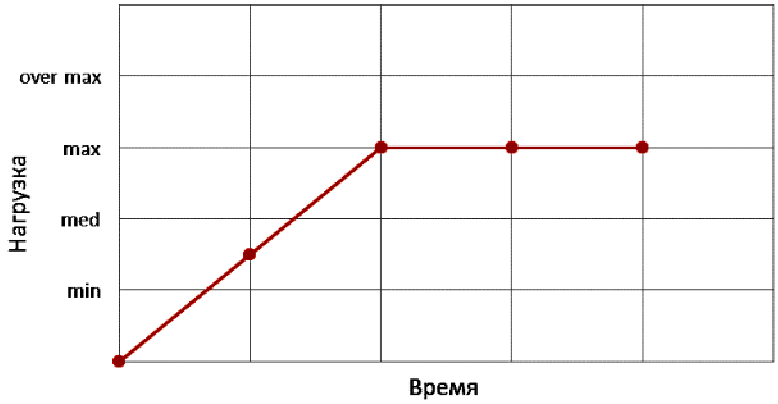


Рисунок 1.4 – Вигляд характеристики додаткового тесту

1.4 Тестування стабільності (stability testing)

Тестування стабільності (stability testing) – дозволяє перевірити працездатність системи на тривалому інтервалі часу (рисунок 1.5). При цьому навантаження може не досягати пікових значень, а мати середні значення, так само сам час виконання операцій не виявляє основним фактором в оцінці результатів тестування [3].

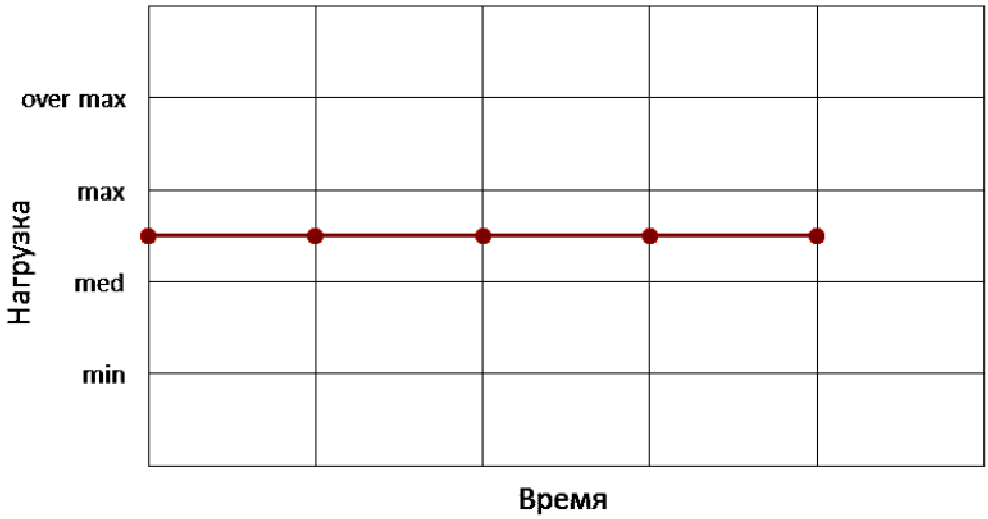


Рисунок 1.5 – Характеристика тестування стабільності

У ході тестування основний акцент робиться на вимірювання:

- відсутності витоків пам'яті;
- відсутності перезапусків серверів;

- відсутності перезапусків програмного забезпечення;
- будь-яких помилок, пов'язаних з накопиченням даних;
- відсутності відключень або збоїв у роботі мережного устаткування.

1.5 Тестування відмовостійкості (failover testing)

Тестування відмовостійкості (failover testing) – даний вид тестування продуктивності дозволяє перевірити поводження системи у випадку збою серверів або при інших несприятливих факторах. Таке тестування особливе важливо в системах, що працюють у режимі 24/7, тому що у випадку їхнього виходу з ладу можливі втрати клієнтів, репутації, грошей і т.п.

Під час тестування перевіряються наступні операції:

– як буде переборюватися відмовлення, а саме як система буде переміщати операції між потужностями працюючих і немає устаткування;

– як буде здійснене перехоплення керування системою при відмовленні керуючого сервера;

– як буде здійснено обхід і обробка відмовлення (переключення на резервний канал зв'язку, відправлення даних по іншому маршруті і т.д.) [2, 3].

1.6 Тестування відновлення (recovery testing)

Тестування відновлення (recovery testing) – зазвичай невід'ємно пов'язано з тестуванням відмовостійкості і дозволяє визначити, як швидко система зможе відновиться після збою її програмної або апаратної частини. Сам збій здійснюється тестувальником шляхом відключення, наприклад, одного із серверів або його перезавантаження. Тестування не націлене на перевірку надійності системи. При цьому навантаження на систему не зменшується і має середні або граничне значення.

У ході тестування виміряються наступні показники:

– час, за який система відновиться після збою;

– коректність відновлених даних.

1.7 Стресове тестування (stress testing)

Стресове тестування (stress testing) – метою даного виду тестування продуктивності є оцінка продуктивності системи при граничних значеннях робочого навантаження або за її межею (рисунок 1.6). Також у ході тестування можна оцінювати роботу системи при зміні ресурсів доступних системі, таких як процесорний час, пам'ять, ширина мережного каналу і т.д.

У ході тестування вимірюється [1, 2, 3]:

– можливість і час регенерації системи – можливість і час повернення системи до нормального стану після стресових навантажень;

– коректність логування помилок і оповіщень про їхнє виникнення;

– продуктивність системи при стресовому навантаженні;

– оцінка впливу збоїв системи, що тестується, на зовнішні системи.

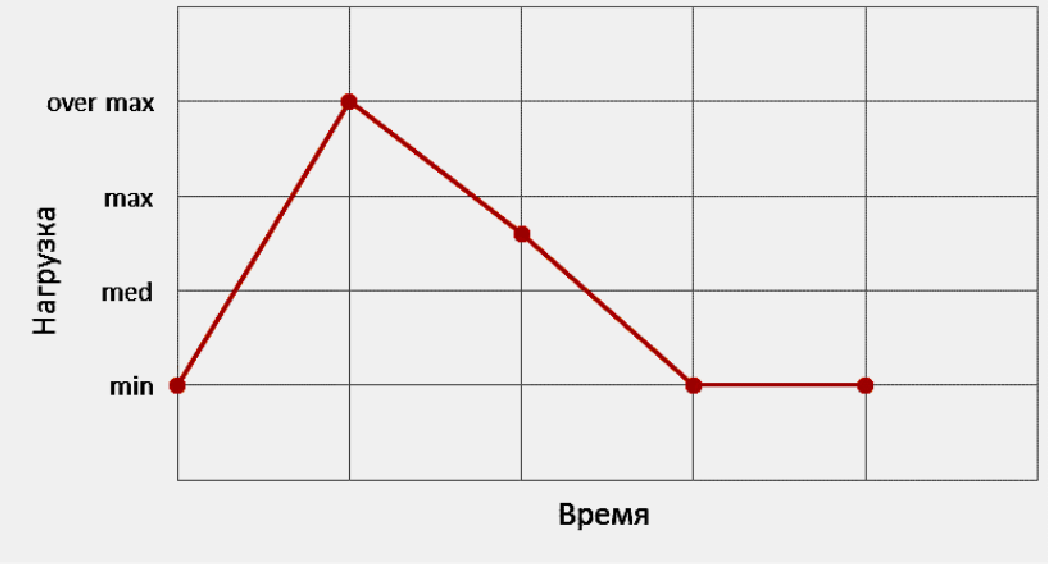


Рисунок 1.6 – Стресове тестування

1.8 Тестування обсягів (volume testing)

Об'ємне тестування (volume testing) – тестування дозволяє оцінити продуктивність системи при збільшенні обсягів даних як самого додатка, так і його бази даних. Основне питання, на яке відповідає даний вид тестування продуктивності: “Що буде завтра з цим додатком або через рік при збільшенні числа

користувачів і/або збільшення збережених користувальницький і системних даних?” [4].

Під час тестування виміряються наступні параметри:

– залежність часу виконання операцій на сервері від обсягу даних;

– кількість користувачів, що можуть одночасно працювати з додатком “швидко”;

– як швидко збільшується обсяг даних при роботі додатка.

1.9 Тестування масштабованості (scalability testing)

Тестування масштабованості (scalability testing) – дане тестування здійснюється для перевірки можливостей масштабування додатка під будь-яким видом навантаження. Також необхідно перевіряти продуктивність системи під час масштабування [3].

Види масштабування, що перевіряються в ході тестування:

– вертикальне масштабування – збільшення продуктивності кожного окремого компонента системи (додавання оперативної пам'яті на сервері, заміна процесора і т.д.) для підвищення продуктивності всієї системи в цілому;

– горизонтальне масштабування – розподіл системи на більшу кількість серверів паралельно працюючих і виконуючих ті самі функції;

– застосування часового масштабування усередині системи за допомогою черг, асинхронних запитів і т.п.

1.10 Тестування потенційних можливостей (capacity testing)

Тестування потенційних можливостей (capacity testing) – є почасти підвидом тестування масштабованості. Якщо в тестуванні масштабованості основне питання, що ставиться: наскільки добре справляється система зі зростаючою кількістю користувачів (навантаження). То в даному випадку питання звучить у такий спосіб: скільки користувачів (з яким навантаженням) може працювати із

системою при цьому час відгуку й інші параметри продуктивності повинні знаходиться в межах припустимих значень? Даний вид тестування дозволяє визначити стратегію масштабування і взагалі зрозуміти, а чи варто масштабувати систему взагалі? [2, 3]

1.11 Конфігураційне тестування (configuration testing)

Конфігураційне тестування (configuration testing) – даний вид тестування перевіряє продуктивність системи на різних апаратних і програмних конфігураціях. У ході тестування виміряються основні показники продуктивності системи при середніх і граничних значеннях навантаження (рисунок 1.7). Даний вид тестування продуктивності дозволяє переконається, що на інших конфігураціях апаратного і програмного забезпечення система буде працювати з однаковою продуктивністю [3, 4].

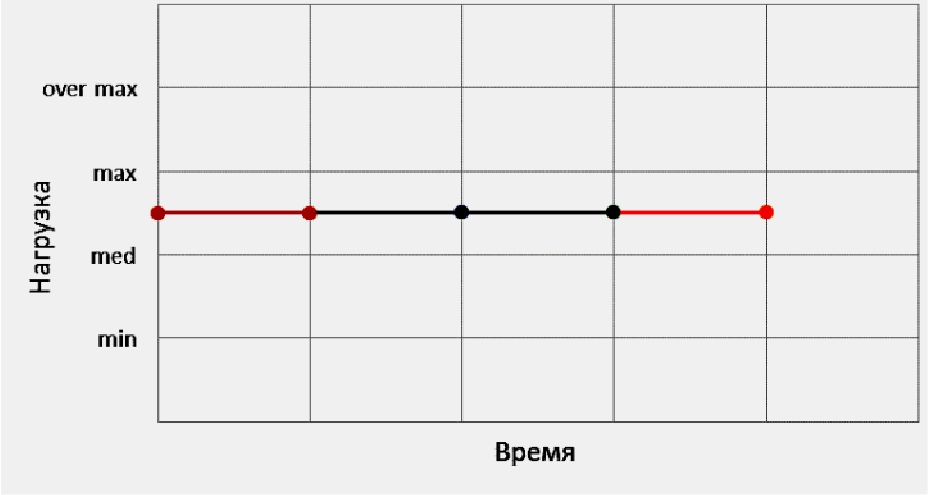


Рисунок 1.7 – Конфігураційне тестування

1.12 Тестування порівняння (compare testing)

Тестування порівняння (compare testing) – дозволяє порівняти продуктивності на різній конфігурації програмної й апаратної частини системи. Дане тестування допомагає обрати найбільш оптимальну конфігурацію апаратного і програмного забезпечення. У ході тестування здійснюється перевірка на різних

конфігураціях, при цьому профіль тестування не змінюється від конфігурації до конфігурації і має середню або граничну інтенсивність навантаження [3 – 6].

Тестування дозволяє відповісти на такі питання як:

- яку СУБД обрати?
- яке устаткування обрати (платформа, виробник, ціна і т.д.)?
- як вплинуть на роботу додатка відновлення і патчі?

2 ОГЛЯД ПРОГРАМНИХ ПРОДУКТІВ ДЛЯ ПРОВЕДЕННЯ ТЕСТУВАННЯ

2.1 Загальні положення

Уявимо, що ми з вами написали якийсь сервіс – тепер потрібно зрозуміти, яке навантаження він витримає. Старий жарт розробки якісного ПО говорить, що краще мати софт, що працює гарантовано погано, ніж софт, що працює хоч і добре, але не гарантовано добре. У першому випадку ми хоча б будемо знати, на що твердо можемо розраховувати. Далі, якщо наш сервіс уміє масштабуватись тим або іншим способом, то потрібно зрозуміти, наскільки, з ростом навантаження, масштабування виявляється корисним і чи виконує воно покладені на нього проектом задачі.

Направляємо на наш розроблений сервіс навантаження та спостерігаємо за результатом: нас, мабуть цікавить ситуація, коли сервіс або стане відповідати на запити з неприйнятною затримкою, або буде повертати невірні дані, або зовсім перестане подавати ознаки життя для всіх запитів або лише для їхньої частини.

Давайте уявимо, що ми з вами написали деякий програмний комплекс – для визначеності скажемо, що веб-сервіс, але це не настільки важливо. Щоб переконатися, на що ми з ним можемо розраховувати, ми починаємо «обстрілювати» його запитами, спостерігаючи за поводженням як самого сервісу, так і за навантаженням на серверах, де він крутиться. Добре, якщо заздалегідь зрозуміло, які запити нам потрібно відправляти сервісові (у цьому випадку ми можемо підготувати масив запитів заздалегідь, а після відправити його в наш додаток одним махом). Якщо ж другий запит залежить від результатів першого (простий приклад – спочатку відбувається авторизація користувача, і в наступні звертання до сервісу включається інформація про ID сесії), то генератор навантаження повинний бути здатний генерувати тестові запити дуже швидко, у реальному часі. З урахуванням обставин і нашого знання про об'єкт тестування вибираємо інструмент(и) [1–2].

2.2 Універсальні інструменти для тестування

2.2.1 Програма JMeter

Старий добрий JMeter (рисунок 2.1). Він от уже без малого двадцять років є частим вибором для багатьох варіантів і типів навантажувального тестування, тому що:

- зручний GUI;
- незалежний від платформи;
- підтримує багатопоточність;
- підтримує розширюваність;
- має відмінні можливості по створенню звітів;
- підтримує багато протоколів для запитів.

Завдяки модульній архітектурі JMeter можна розширити в потрібну користувачеві сторону, реалізуючи навіть досить екзотичні сценарії тестування – причому, якщо жоден з написаних за минулий час плагінів користувача не влаштує, можна взяти API і написати власний. При необхідності з JMeter можна побудувати, хоч і обмежене, але розподілене тестування, коли навантаження буде створюватися відразу декількома машинами [5].

Рисунок 2.1 – Логотип програми JMeter

Однією зі зручних функцій JMeter є робота в режимі проксі: вказуємо в налаштуваннях браузера в якості проксі «127.0.0.1:8080» і відвідуємо браузером потрібні нам сторінки, потрібного сайту, JMeter збереже всі наші дії і всі супутні запити у виді скрипту, що пізніше можна буде відредагувати, як потрібно – це робить процес створення HTTP-тестів помітно простішим.

До речі, остання версія (3.2), що вийшла в квітні цього року, навчилася віддавати результати тестування в InfluxDB за допомогою асинхронних HTTP-запитів. Правда, починаючи саме з версії 3.2, JMeter став вимагати тільки Java 8, але це, напевно, не найвища ціна за прогрес.

Збереження тестових сценаріїв у JMeter реалізовано в XML-файлах, що як виявилося, створює масу проблем: їх зовсім незручно писати руками, як незручна і робота з такими файлами в системах керування версіями (особливо в момент, коли потрібно зробити diff). Конкуруючі на полі навантажувального тестування продукти, такі як Яндекс Танк або Taurus, навчилися самостійно і на льоту формувати файли з тестами і передавати їх у JMeter на виконання, у такий спосіб користуючись міццю і досвідом JMeter, але даючи можливість користувачам створювати тести у більш читаємому і легше модифікованому CVS форматі тестових скриптів [5–7].

2.2.2 Програма LoadRunner

Ще один давно існуючий на ринку й у визначених колах дуже відомий продукт, більшому поширенню якого перешкодила прийнята компанією-виробником політика ліцензування (до речі, після злиття підрозділу ПО компанії Hewlett Packard Enterprise з Micro Focus International, звична назва HPE LoadRunner змінилася на Micro Focus LoadRunner) (рисунок 2.2).

Рисунок 2.2 – Логотип програми Micro Focus LoadRunner

Інтерес представляє логіка створення тесту, де кілька (напевно, правильно сказати – «багато») віртуальних користувачів паралельно щось роблять з тестуємим додатком. Це дає можливість не тільки оцінити здатність додатка об-

робити потік одночасних запитів, але і зрозуміти, як впливає робота одних користувачів, активно щось роблячих із сервісом, на роботу інших. При цьому мова йде про широкий вибір протоколів взаємодії з тестуємим додатком [7].

HP у свій час створила чудовий набір інструментів автоматизації функціонального і навантажувального тестування, що, при необхідності, інтегруються в процес розробки ПО, і LoadRunner вміє інтегруватися з ними (зокрема, з HP Quality Center, HP QuickTest Professional).

Якийсь час назад виробник вирішив повернутися лицем до тих, хто не готовий відразу платити за ліцензію, і поставляє LoadRunner з безкоштовною ліцензією (де уписаний ліміт на 50 віртуальних користувачів, і заборонена невелика частина всього набору підтримуваних протоколів), а гроші беруться за подальше розширення можливостей. Складно сказати, наскільки це буде сприяти підвищенню інтересу до цього, без сумніву, цікавому інструментові, при наявності в нього настільки сильних конкурентів.

2.2.3 Програма Gatling

Досить потужний і серйозний інструмент (не даремно названий на честь скорострільного кулемета) – у першу чергу, через продуктивність і широту підтримки протоколів «з коробки». Наприклад, там, де навантажувальне тестування з JMeter буде повільним і болісним (на жаль, плагін підтримки роботи з веб-сокетами не особливо швидкий, що ідейно конфліктує зі швидкістю роботи самих веб-сокетов), Galting майже напевно створить потрібне навантаження без особливих ускладнень (рисунок 2.3).

Рисунок 2.3 – Логотип програми Galting

Варто врахувати, що, на відміну від JMeter, Gatling не використовує GUI і взагалі вважається засобом, орієнтованим на досвідчену, «грамотну» аудиторію, здатну створити тестовий скрипт у виді текстового файлу.

Є в Gatling і мінуси, за які його критикують. По-перше, документація могла б бути і кращою.

По-друге, для роботи з ним непогано знати Scala: і сам Gatling, як інструмент тестування, і тестові сценарії пишуться саме на цій мові.

По-третє, розробники «іноді» у минулому кардинально змінювали API, у результаті можна було знайти, що тести, написані навіть півроку раніше, «не йдуть» на новій версії, або вимагають доопрацювання/міграції. У Gatling також відсутня можливість робити розподілене тестування, що обмежує можливі області застосування [7].

2.2.4 Враппер Яндекс Танк

Yandex Tank – це враппер, над декількома утилітами навантажувального тестування (включаючи JMeter), що надає уніфікований інтерфейс для їхньої конфігурації, запуску і побудови звітів поза залежністю від того, яка утиліта використовується «під капотом».

Він уміє стежити за основними метриками тестуємого додатку (процесор, пам'ять, своп), за ресурсами системи (вільна пам'ять/місце на диску), може зупинити тест на основі різних зрозумілих критеріїв («якщо час відгуку перевищує задане значення», «якщо кількість помилок за одиницю часу вище, ніж х» і т.д). До речі, вміє відображати в реальному часі основні статистичні дані тесту, що буває дуже корисно прямо в процесі тесту [7].

Танк використовується й у самому Яндексі, і в інших компаніях уже близько 10 років (рисунок 2.4). Ним "обстрілюють" зовсім різні сервіси, з різними вимогами до складності тестових сценаріїв і до рівня навантаження. Майже завжди для тестування навіть високо навантажених сервісів вистачає всього одного генератора навантаження.

Рисунок 2.4 – Логотип Yandex Tank

Танк підтримує різні генератори навантаження, як написані спеціально для нього (Phantom, BFG, Pandora), так і широко сторонні (JMeter). Модульна архітектура дозволяє написати свій плагін під потрібний генератор навантаження і взагалі прикрутити практично що завгодно [7].

Для чого використовувати різні генератори навантаження? Phantom – це швидкий програмний комплекс написаний на C++. Один такий генератор може видати до сотні тисяч запитів у секунду. Але для досягнення такої швидкості приходиться генерувати запити заздалегідь і не можна (не виходить) використовувати одержувані від тестуємого сервісу дані для генерації чергового запиту.

У випадках, коли потрібно виконувати складний сценарій або сервіс використовує нестандартний протокол, варто використовувати JMeter, BFG, Pandora.

У BFG, на відміну від Jmeter, немає GUI, тестові сценарії пишуться на Python. Це дозволяє використовувати будь-як бібліотеки (а їх величезна кількість). Часто буває, що для сервісу написані біндинги для Python, тоді їх зручно використовувати при написанні навантажувальних сценаріїв.

Pandora – це експериментальна гармата на GoLang, досить швидких і розширювана, підходить для тестів по протоколі HTTP/2 і буде використовуватися там, де потрібні швидкі сценарії.

Усередині Яндекса для збереження і відображення результатів навантажувальних тестів використовується спеціальний сервіс. Зараз назовні відкритий його спрощений аналог за назвою Overload – він цілком безкоштовний, його використовують, у тому числі, для тестування відкритих бібліотек [7].

2.2.5 Програма Taurus

Taurus – ще одна програма над декількома утилітами навантажувального тестування (рисунок 2.5). Можливо, вам сподобається цей продукт, що використовує схожий на Яндекс Танк підхід, але інший набір, що має декілька, «фіч» і мабуть, більш адекватний формат конфігураційних файлів [7–8].

Рисунок 2.5 – Логотип програми Taurus

Taurus добре підійде в ситуації, коли потужність, скажемо, Gatling важлива для створення тесту, але розбиратися з Gatling (а також з написанням скриптів тестування на Scala) немає бажання або можливості: досить описати тест у куди більш простому форматі файлу Taurus, настроїти його на використання Gatling як інструмента створення навантаження, і всі Scala-файли будуть згенеровані автоматично. Так би мовити, «автоматизація автоматизації» у дії.

Taurus можна настроїти на відправлення статистики тесту на онлайн-сервіс BlazeMeter.com, що відобразить дані у виді ошатних графіків і таблиць. Підхід не дуже звичайний, але движок, що заслуговує уваги: виведення звітів, мабуть, удосконалиться згодом, і поступово буде виводити інформацію ще більш привабливо [7–8].

2.3 Підходи до тестування продуктивності

Тестувати продуктивність сервісу або додатка можна і потрібно не тільки після завершення процесу розробки, але і під час розробки, буквально так само, як ми робимо регулярні юніт – або регресійні тести. Правильно організовані, регулярні тести продуктивності дозволяють відповісти на дуже «тонке» питання: чи не привели останні зміни в коді додатка до погіршення продуктивності ПО?

Здавалося б, поміряти продуктивність – це так просто! Два рази взяти timestamp (бажано з високою точністю), порахувати різницю, зложилися-поділили, і все – можна оптимізувати. Як би не так!

Хоча на словах це питання звучить просто, на ділі такого роду виміри досить важко робити, а порівнювати результати різних вимірів узагалі не завжди розумно. Одна з причин: для зіставлення результатів тести повинні проходити над тими самими вихідними даними, що, серед іншого, має на увазі відтворення тестового середовища при кожнім прогоні перевірки, інша причина – порівняння суб'єктивного сприйняття часу роботи тестового сценарію може виявитися неточним.

Ще однією причиною є складність виділення впливу на продуктивність цілого додатка роботи окремого його модуля, того, котрий ми зараз правимо. Збільшуючи ситуацію, уточнюємо: ще складніше цей вплив вичленувати, якщо над кодом працює колектив з більш ніж одного розробника.

Один з підходів у такій ситуації складається в ретельному створенні повноцінного тестового сценарію, що повторює роботу із сервісом дійсного клієнта, і прогоні його багато разів, з паралельним аналізом навантаження на сервер, де проходить тестування (таким чином, буде зрозуміло, яка частина сценарію створює навантаження на окремі ресурси тестового сервера, що може дати додаткову інформацію з пошукові місць, де варто підійти до продуктивності більш серйозно) – на жаль, не завжди можна таке дозволити собі в реальній ситуації, просто тому, що об'ємний тест, та ще повторений 10...20 разів, швидше за все буде занадто довгим, щоб проводити його досить часто, а це цілком уб'є ідею.

Інший підхід, більш підходящий до процесу розробки, полягає в організації обмеженого за масштабом, «мікро-» або навіть «нано-» тестування окремих місць коду (скажемо, запуску одного методу або однієї функції, але велике число разів – тобто, скоріше, бенчмаркінгу). Планування такого тестування вимагає додаткових зусиль з боку розробки, але результат окупається і загальним поліпшенням продуктивності коду, і розумінням, як поводяться окремі частини

проекту в міру роботи як над ними, так і над іншими частинами. Нижче наведено приклади інструментів для тестування продуктивності [6].

2.4 Інструменти для тестування продуктивності

2.4.1 Фреймворк JMH

JMH (Java Microbenchmark Harness) – це оснащення Java для зборки, запуску й аналізу нано/мікро/мілі/макро-бенчмарків, написаних на Java і інших мовах з цільовою платформою JVM (рисунок 2.6).

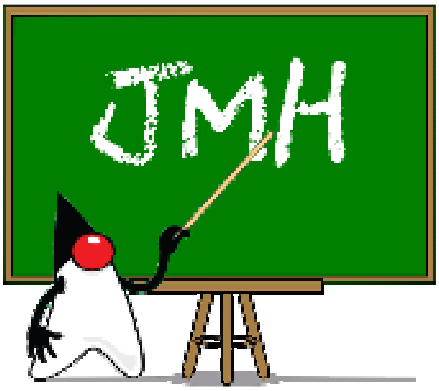


Рисунок 2.6 – Логотип фреймворку JMH

Порівняно молодий фреймворк, у якому розробники постаралися врахувати всі нюанси JVM. Один з найзручніших інструментів, з тих, котрі приємно мати під рукою. JMH підтримує наступні типи вимірів:

– Throughput (вимірювання чистої продуктивності);

– AverageTime (вимірювання середнього часу виконання);

– SampleTime (перцентиль часу виконання);

– SingleShotTime (час виклику одного методу – актуально для виміру «холодного» запуску тестуємого коду).

Оскільки мова йде про Java, фреймворк враховує в т.ч. і роботу механізму кешування JVM, і перед запуском бенчмарка кілька разів виконує тестуємий код для «прогріву» кеша байта-коду Java-машини [7].

2.4.2 Програма BenchmarkDotNet

BenchmarkDotNet бере на себе рутинні дії при складанні бенчмарків для .NET-проектів і забезпечує широкі можливості форматування результатів ціною мінімальних зусиль (рисунок 2.7).

Рисунок 2.7 – Логотип програми BenchmarkDotNet

На сьогоднішній день BenchmarkDotNet – використовують у першу чергу, для бенчмарків, а не для перфоменс-тестів. Ведеться серйозна робота над тим, щоб продукт можна було також використовувати на CI-сервері для автоматичного детектування перфоменс регрессій, але поки ці розробки не довершені [7].

2.4.3 Програма Google Lighthouse

Виміри продуктивності фронтенда завжди стояли трохи поодаль: з одного боку, часто затримки пов'язані зі швидкістю реакції бекенда, з іншого боку – саме по поводженню фронтенда (точніше, по швидкості його реакції) користувачі часто судять про весь додаток, особливо, якщо мова йде про веб.

У веб-фронтенді у відношенні вимірів продуктивності зараз все йде убік використання Performance API і вимірюванні саме тих параметрів, що мають значення для конкретного проекту.

Google Lighthouse (рисунок 2.8) дозволить побачити картину не зі свого комп'ютера, а з одного з серверів тестування, що існують у світі, і оцінити вплив часу передачі-прийому-передачі даних через канали інтернет на роботу фронтенда.

Рисунок 2.8 – Логотип програми Google Lighthouse

Цей продукт більше підходив би для перевірки сторінок сайту на відповідність рекомендаціям Google (і взагалі best practices) як для веб-сайтів, так і для Progressive Web Apps, якби не одна з його функцій: серед перевірок є і тест на поводження сайту при поганій якості веб-з'єднання, а також при повній відсутності зв'язку [5–7].

Такий підхід не дуже співвідноситься з перформенс-тестуванням як таким, однак, якщо задуматися, у деяких випадках веб-додатком сприймається «повільним» не тому, що повільно готує дані, а тому, що умови його роботи на машині користувача, у його браузері, з урахуванням його з'єднання з інтернетом – на жаль, не ідеальні. Google Lighthouse саме і дозволяє оцінити цей вплив.

2.5 Вибір програмного продукту для проведення навантажувального тестування сайту NURE

У результаті аналізу програмних продуктів, представлених на ринку, було обрано HP LoadRunner. Розглянемо цей програмний комплекс докладніше.

HP LoadRunner утиліта для автоматизованого навантажувального тестування яка може виконувати як тестування різних додатків, так і тестування сайтів різного рівня складності. Підключаючи віртуальних користувачів виконуючих різні скрипти (дії), по різних сценаріях. Програма має відповідні набори інструментів для проведення тестування. Так само до складу HP LoadRunner входить набір інструментів для роботи з різних протоколів з додатком (дистанційно, через проксі-сервер і т.п.) [10].

HP LoadRunner складається з наступних компонентів:

– Virtual User Generator (VUGen);

– Controller;

– Analysis.

2.5.1 Virtual User Generator

Модуль Virtual User Generator призначений для розробки скриптів, що будуть задіяні для подальшого тестування. Має великий набір інструментів, що дозволяють написати максимально продуктивні скрипти для тестування програмного продукту.

Частина інструментів дозволяє вести автоматичне написання скриптів. Досить включити запис і всі дії виконувані користувачем на комп'ютері будуть записуватися в скрипт. Хоча надалі такі скрипти бажано вручну оптимізувати, підвищуючи їхню ефективність і безвідмовність.

Також даний модуль має функції для настроювання роботи з параметрами захисту тестуємого додатку. Допустимо, якщо трафік сайту захищений недовіреним сертифікатом, то при вході на такий сайт захист буде видавати попередження про те, що надійність сайту підозріла. У результаті настроювань HP LoadRunner для роботи з таким сертифікатом, в автоматичне написання скриптів не будуть попадати зайві дані про захист сайту, що істотно поліпшить роботу скрипту.

Скрипти створені даним модулем мають гнучку структуру, яку можна переконфігурувати залежно від вимог до тесту. Структура скрипту за замовчуванням складається з трьох частин:

– Vuser_init – у дану секцію записуються початкові дії користувача, що приведуть до запуску тестуємого модуля додатку.

– Action – у цю секцію записується основна частина скрипту, що і буде генерувати навантаження на об'єкт тестування.

– Vuser_end – в останній секції записуються дії, що приводять до коректного закриття модуля і завершення роботи користувача з тестуємим додатком.

Такий підхід до написання скриптів забезпечує високу ефективність роботи скриптів [10, 11].

2.5.2 Controller

Модуль Controller– основний модуль програми, що створює і виконує сценарій проведення тестування. У цей модуль підключаються скрипти, написані в Virtual User Generator.

Тестувальник має можливість створити сценарій тестування, та задати такі параметри:

– налаштувати кількість віртуальних користувачів – сформувати них у групи;

– задати інтервали часу, у які визначені групи користувачів будуть підключатися або відключатися;

– додати скрипти які будуть виконувати різні групи користувачів;

– настроїти час виконання сценарію.

Розглянутий модуль має дуже інформативний інтерфейс, тобто після запуску виконання сценарію, можна детально стежити за процесом його виконання.

Тестувальник має можливість стежити за наступними процесами:

– відстеження групи віртуальних користувачів, на якому етапі вони знаходяться.

– моніторинг графіків, що відображають проходження процесу тестування. Різні графіки можна підключити в будь-який момент виконання сценарію і вони відобразять дані, що записувалися із самого старту сценарію.

Графіки мають також різні налаштування для зручного моніторингу процесу. По завершенню виконання сценарію адміністратор може перейти в модуль Analysis [10, 11].

2.5.3 Analysis

Модуль Analysis призначений для складання детальних звітів про пророблене тестування. Звіти можна формувати двох типів: звіт у вигляді документа (файлу *.doc) і звіт у вигляді html сторінки.

Перед створенням звіту адміністратор набудовує цікавлячі його показники, що потраплять у звіт, тобто різні графіки, що записувалися під час проведення тестування; стилі відображення цих графіків (лінійні, об'ємні і т.п.); помилки, що виникали в ході виконання скриптів користувачами і т.п.

Загалом, у звіт можна включити усе, що відбувалося під час тестування. Після настроювання звіту і його формування адміністратор одержує детальну звітність про проведене тестування.

На відміну від GUI-тестів навантажувальні тести (VuGen) працюють із трафіком між клієнтом і сервером (прикладний і транспортний рівень), а для генерації навантаження можливо використовувати одну робочу станцію.

HP LoadRunner дозволяє моніторити системні ресурси і може інтегруватися з HP Quality Center для збереження навантажувальних скриптів, сценаріїв, результатів іспитів і з HP QuickTest Professional для навантаження за допомогою GUI тесту [10, 11].

3 КОНЦЕПЦІЯ СИСТЕМИ

3.1 Опис основної функціональності системи

Під час розробки програм навантажувального тестування потрібно охопити всі найчастіше використовувані тест-кейси, які виникають під час роботи з веб-сайтом у звичайних користувачів:

– перегляд різних сторінок веб-сайту;

– перехід на сторінки веб-сайту з заздалегідь відкритих сторінок;

– відкриття одного з’єднання з однієї сесії;

– відкриття декількох з’єднань з однієї сесії.

Як було показано в розділі 2, в якості програмного продукту для тестування сайту було обрано програму LoadRunner компанії Hewlett Packard Enterprise.

Для розробки скриптів, що будуть задіяні для подальшого тестування, програмою LoadRunner доцільно задіяти модуль Virtual User Generator.

Для створення і виконання сценаріїв проведення тестування програмою LoadRunner доцільно задіяти модуль Controller. У цей модуль підключаються скрипти, написані в Virtual User Generator.

Для складання детальних звітів про пророблене тестування програмою LoadRunner доцільно задіяти модуль Analysis.

Взаємодія програмних компонентів та інструментів тестування з тестувальником наведена на рисунку 3.1.

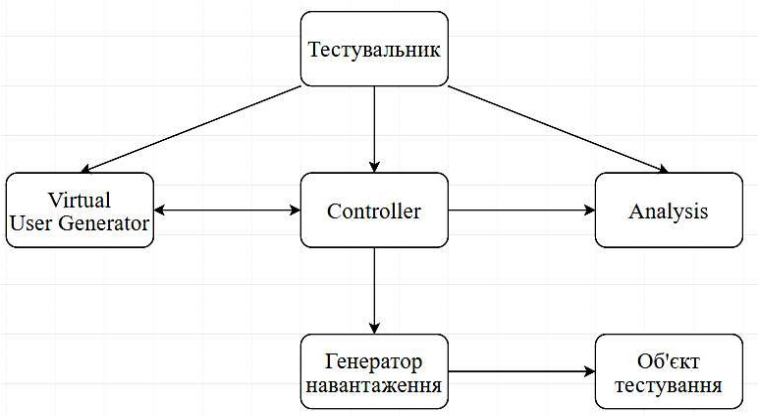


Рисунок 3.1 – Схема взаємодії з тестувальником

3.2 Функціональні можливості та робота з модулем Virtual User Generator

Основне призначення модуля Virtual User Generator полягає в створенні скриптів (програм) VUScripts, які використовуються для імітації реального віртуального користувача.

Virtual User Generator (Vuser) не тільки записує скрипти, а й відтворює їх для забезпечення правильного запису сценарію. Переконавшись, що скрипт записаний правильно, його можна додати в сценарій LoadRunner.

Почнемо створення скрипту (програми) навантажувального тестування:

У меню «Файл» (рисунок 3.2) присутні декілька команд управління.

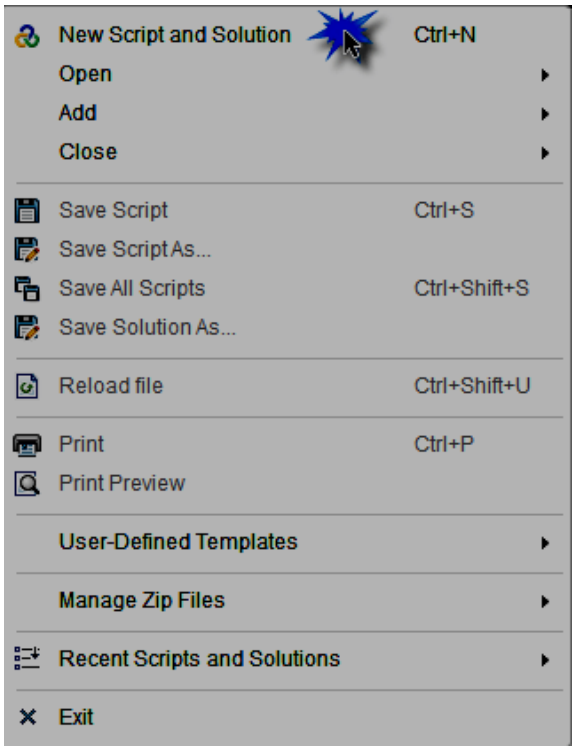


Рисунок 3.2 – Меню «Файл»

Саму розробку можна розподілити на чотири кроки.

Крок 1. Обираємо команду New Script and Solution (новий сценарій та рішення), за якою відкривається вікно для вибору протоколу наведене на рисунку 3.3.

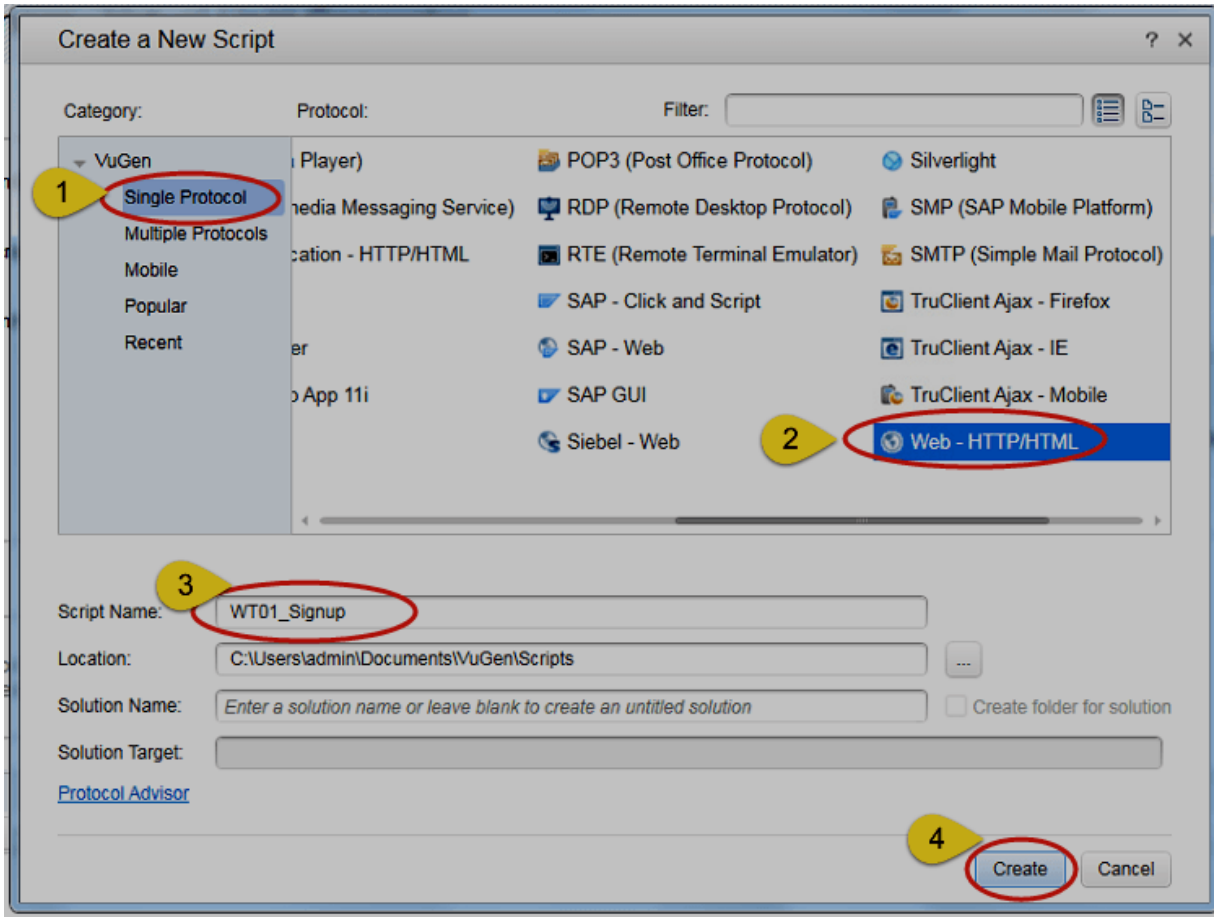


Рисунок 3.3 – Вікно для вибору протоколу

Крок 2. В лівій частині вікна активуємо опцію Single Protocol (один протокол) та обираємо протокол «Web – HTTP / HTML».

У вікні Script Name вводимо ім'я скрипту, наприклад WT01_Signup та натискаємо кнопку «Create» (створити).

Якщо необхідно перейменувати скрипт, використовується функція «Зберегти як» і вводиться нове ім'я. Добрим прикладом може бути щось на зразок WT01_Signup, де WT - це коротка форма імені додатки, 01 - послідовність бізнес-процесів, а реєстрація - це сценарії бізнес-процесів. Також необхідно звернути увагу на те, що використовувати пробіли в імені скрипту заборонено.

В тих випадках, коли потрібно більше одного протоколу для зв'язку зі своїм сервером, в лівій частині вікна (див. рисунок 3.3), активуємо опцію Multiple Protocols (декілька протоколів). В такому випадку у випадаючому вікні (рисунок 3.4), обираємо необхідні протоколи.

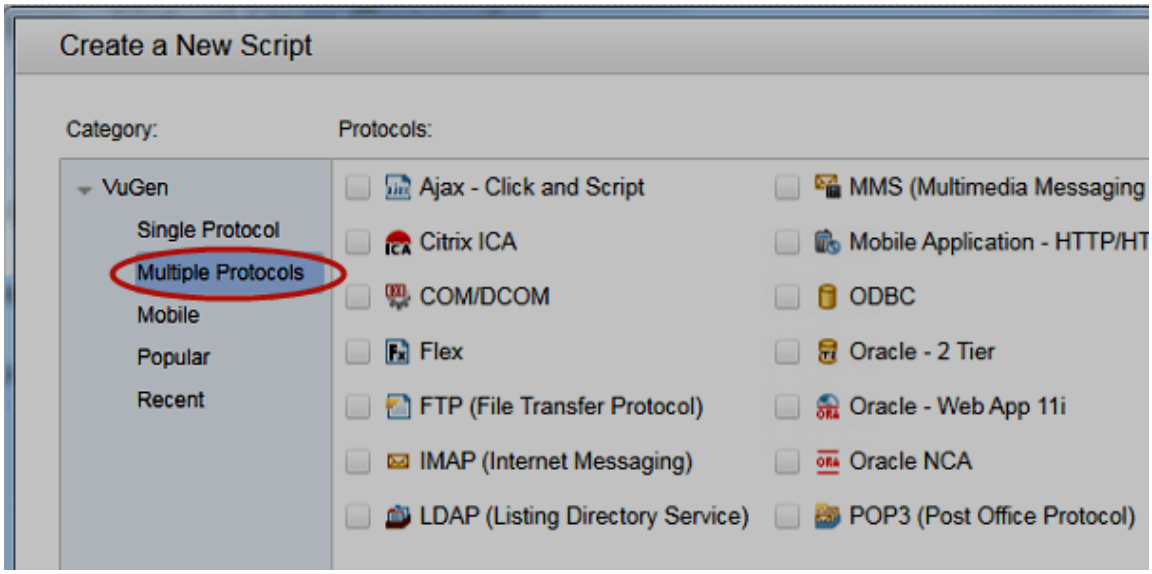


Рисунок 3.4 – Вікно для вибору декількох протоколу

Однак для роботи в веб ресурсами, як правило, потрібен тільки один протокол для скриптів VUGen.

Крок 3. Після натискання кнопки «Create» (Створити) HP VUGen відкриє IDE (інтегроване середовище розробки) або редактор коду (рисунок 3.5) з порожніми, крім базового підпису Function Action, файлами скриптів.

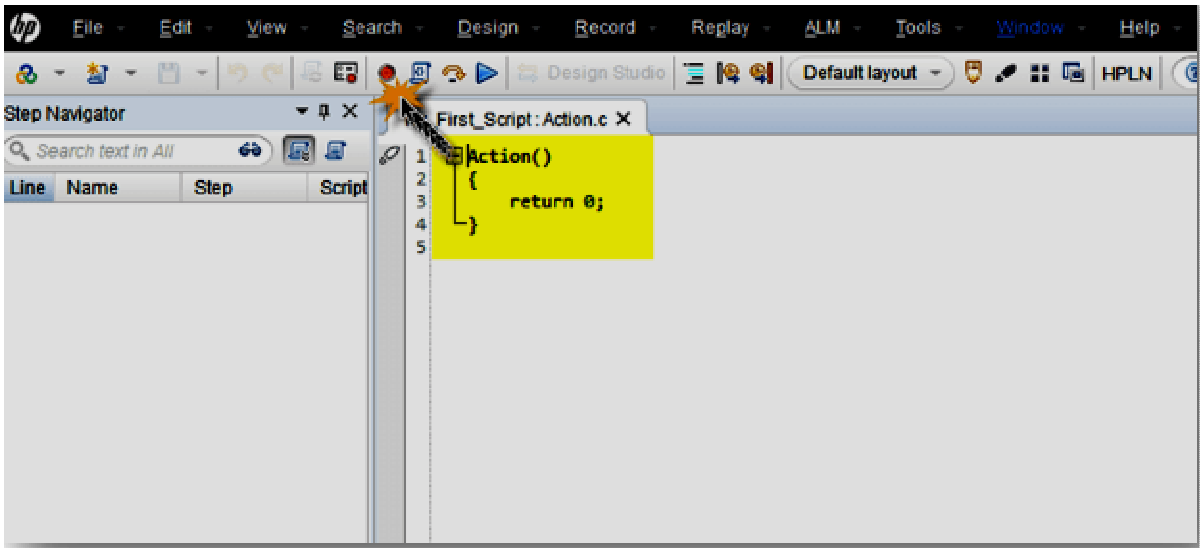


Рисунок 3.5 – Вікно редактору коду

Крок 4. Можна приступати до написання коду (розробки скрипу навантажувального тестування), згідно з бізнес логікою.

Всього було розроблено 9 скриптів (програм) навантажувального тестування:

- UC01_ChangeLanguage;
- UC02_Search;
- UC03_UNIVERSITY;
- UC04_APPLICANTS;
- UC05_STUDENTS;
- UC06_SCIENCE;
- UC07_EDUCATION;
- UC08_PRESS_CENTER;
- UC09_CONTACTS.

Програма UC01_ChangeLanguage імітує дії користувача при зміні мови на сайту NURE, всього на сайті доступні 3 мови: українська, російська та англійська.

Програма UC02_Search імітує дії користувача який використовує пошук по сайту NURE, кожного разу генерується нова послідовність для пошуку, це допомагає не зіткнутися з «кешуванням» результатів пошукової видачі, що в свою чергу зіпсує результати тесту.

Програма UC03_UNIVERSITY імітує дії користувача при послідовному переході користувача з головної сторінки на вкладку UNIVERSITY і далі на вкладку ABOUT_UNIVERSITY. Такі послідовні переходи прийняти назвати «хлібні крихти».

Програма UC04_APPLICANTS імітує дії користувача при послідовному переході користувача з головної сторінки на вкладку APPLICANTS і далі на вкладку EDUCATION_IN_ENGLISH. Таким послідовним переходом відстежуємо роботу «хлібних крихт».

Програма UC05_STUDENTS імітує дії користувача при послідовному переході користувача з головної сторінки на вкладку STUDENTS і далі на вкладку Timetable_of_Classes для перегляду розкладу. Таким послідовним переходом відстежуємо роботу «хлібних крихт».

Програма UC06_SCIENCE імітує дії користувача при послідовному переході користувача з головної сторінки на вкладку SCIENCE і далі на вкладку SCIENCE_Topic для перегляду розкладу. Таким послідовним переходом відстежуємо роботу «хлібних крихт».

Програма UC07_EDUCATION імітує дії користувача при послідовному переході користувача з головної сторінки на вкладку EDUCATION і далі на вкладку SCIENTIFIC_LIBRARY для перегляду розкладу. Таким послідовним переходом відстежуємо роботу «хлібних крихт».

Програма UC08_PRESS_CENTER імітує дії користувача при послідовному переході користувача з головної сторінки на вкладку PRESS_CENTER і далі на вкладку PRESS_SERVICE для перегляду розкладу. Таким послідовним переходом відстежуємо роботу «хлібних крихт».

Програма UC09_CONTACTS імітує дії користувача при послідовному переході користувача з головної сторінки на вкладку CONTACTS і далі на вкладку ABOUT_UNIVERSITY для перегляду розкладу. Таким послідовним переходом відстежуємо роботу «хлібних крихт».

Програмні коди всіх розроблених програм наведено в додатку А.

З метою визначення рівня навантаження який будемо подавати на сайт NURE, проведемо розрахунки навантаження

$$TPS = \frac{V_u}{P \cdot P_s}, \tag{3.1}$$

де TPS (Transaction per second) – кількість транзакцій за секунду,

Vu (Virtual user) – кількість віртуальних користувачів;

P (Pacing) – затримка між ітераціями;

Ps (Pack size) – розмір пачки.

Результати розрахунків наведено на рисунку 3.6.

Номер UC	Назва	Virtual user	Pacing	Pack size	TPS
1	UC01_ChangeLanguage	5	1	3	15
2	UC02_Search	6	1	2	12
3	UC03_UNIVERSITY	6	1	3	18
4	UC04_APPLICANTS	5	1	3	15
5	UC05_STUDENTS	5	1	3	15
6	UC06_SCIENCE	6	1	3	18
7	UC07_EDUCATION	6	1	3	18
8	UC08_PRESS_CENTER	6	1	3	18
9	UC09_CONTACTS	5	1	3	15
	Всього VU	50		Всього TPS	144

Рисунок 3.6 – Розрахунок навантаження

3.3 Функціональні можливості та робота з модулем Controller

Після відкриття модуля Controller, автоматично попадаємо на головну сторінку (рисунок 3.7).

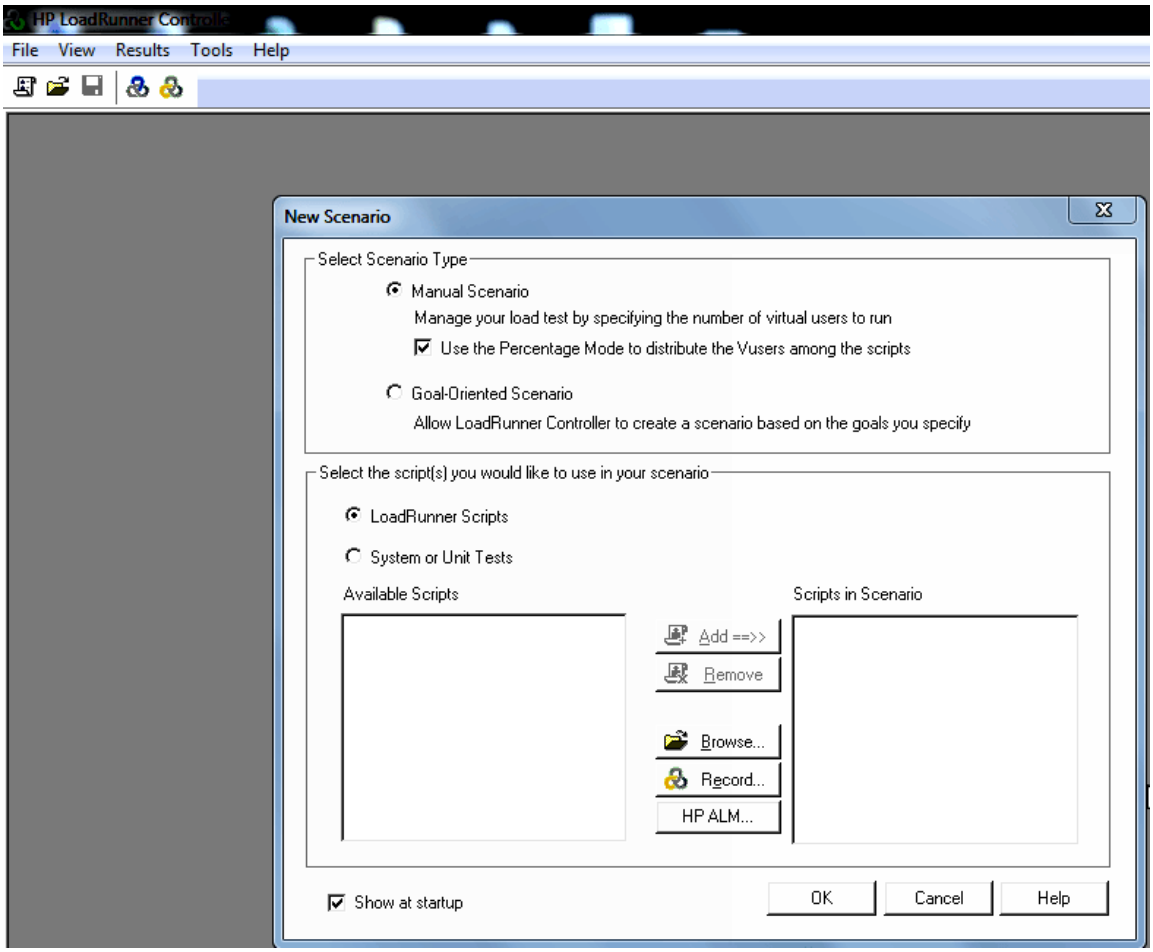


Рисунок 3.7 – Головна сторінка Controller

Для створення нового сценарію в меню «File» (рисунок 3.8) активізуємо опцію «New».

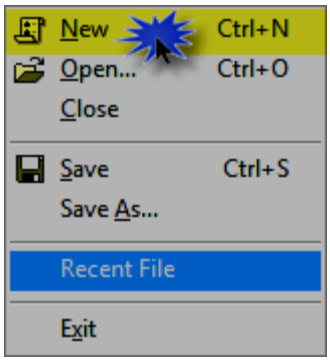


Рисунок 3.8 – Опції меню «File»

Модуль Controller підтримує два типи сценаріїв:

– ручний сценарій;

– цільовий сценарій.

Ручний сценарій є статичним і дає більше контролю над ситуацією. В цьому типі сценарію можна вирішити, яку транзакцію виконувати, скільки разів і як довго. Ручний сценарій, крім того, може додатково мати процентний режим. Грунтуючись на складі тестового набору, під час тесту можна побачити поведінку додатка, кількість звернень, час відгуку тощо.

Для створення ручного сценарію, у вікні Select Scenario Type обираємо Manual Scenario (рисунок 3.9) та натискаємо кнопку ENTER, щоб продовжити.

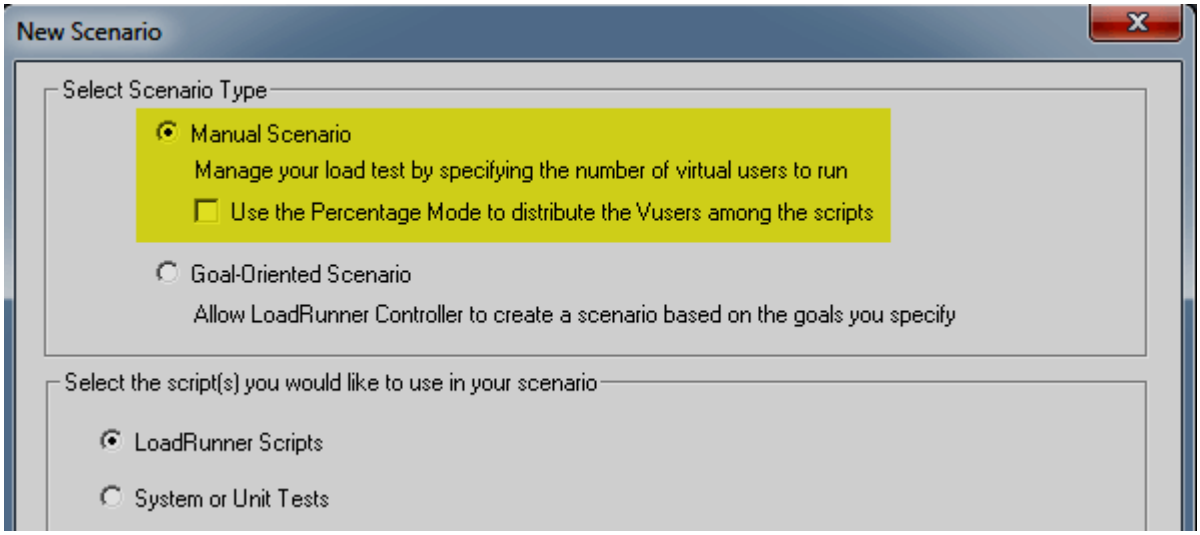


Рисунок 3.9 – Створення ручного сценарію

Це створить порожній сценарій і завантажить його на головний екран (рисунок 3.10).

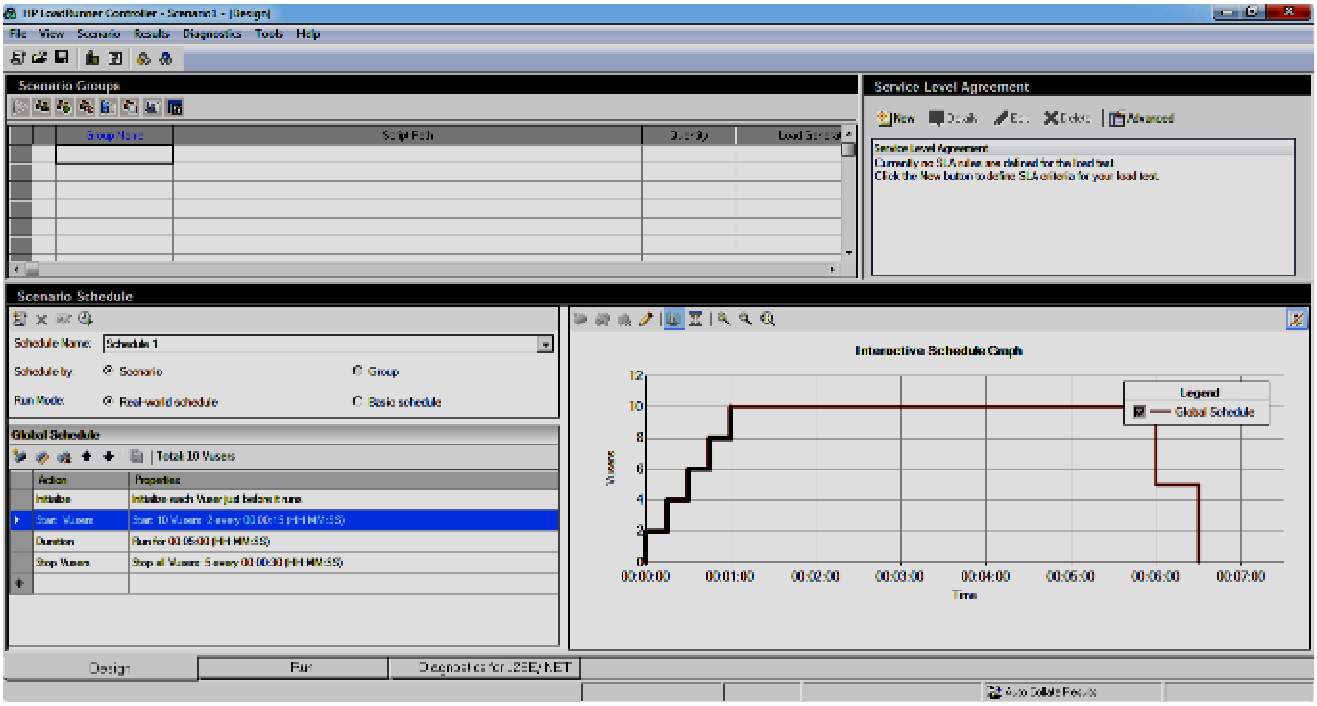


Рисунок 3.10 – Головний екран порожнього сценарію

Щоб додати у сценарій заздалегідь написаний скрипт (програму) потрібно натиснути лівою кнопкою миші у вільному місці та обрати потрібний скрипт. Після цього натиснути кнопку «OK», і скрипт з’явиться на головній сторінці і з ним можна буде працювати.

Після цього можемо приступати до налаштувань (рисунок 3.11).

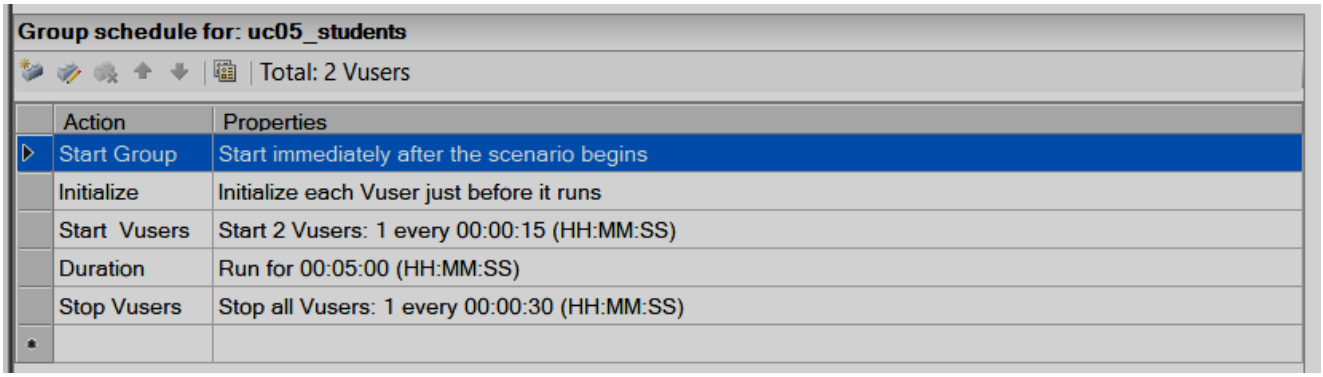


Рисунок 3.11 – Головна панель налаштувань

На вкладці «Start Vusers» (рисунок 3.12) обираємо кількість віртуальних користувачів та частоту їх виходу в роботу.

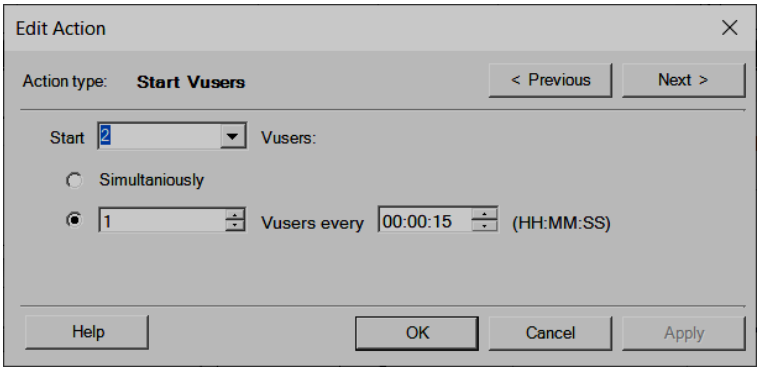


Рисунок 3.12 – Налаштування у вкладці «Start Vusers»

Також налаштовуємо час роботи всіх віртуальних користувачів скрипту, для цього скористаємося вкладкою «Duration» (рисунок 3.13).

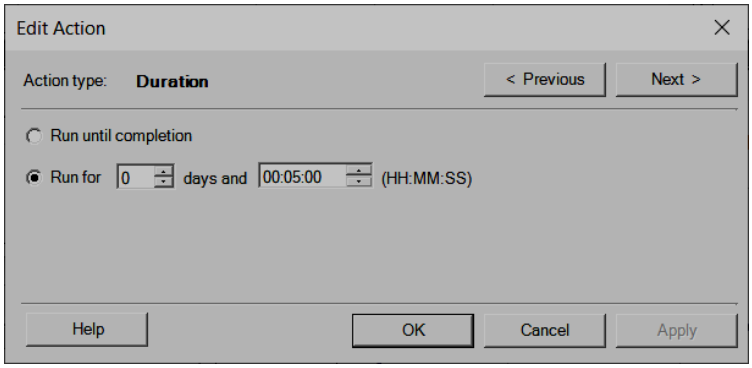


Рисунок 3.13 – Налаштування у вкладці «Duration»

Налаштування поведінки скипту здійснюється на вкладці «Run-time Settings» (рисунок 3.14).

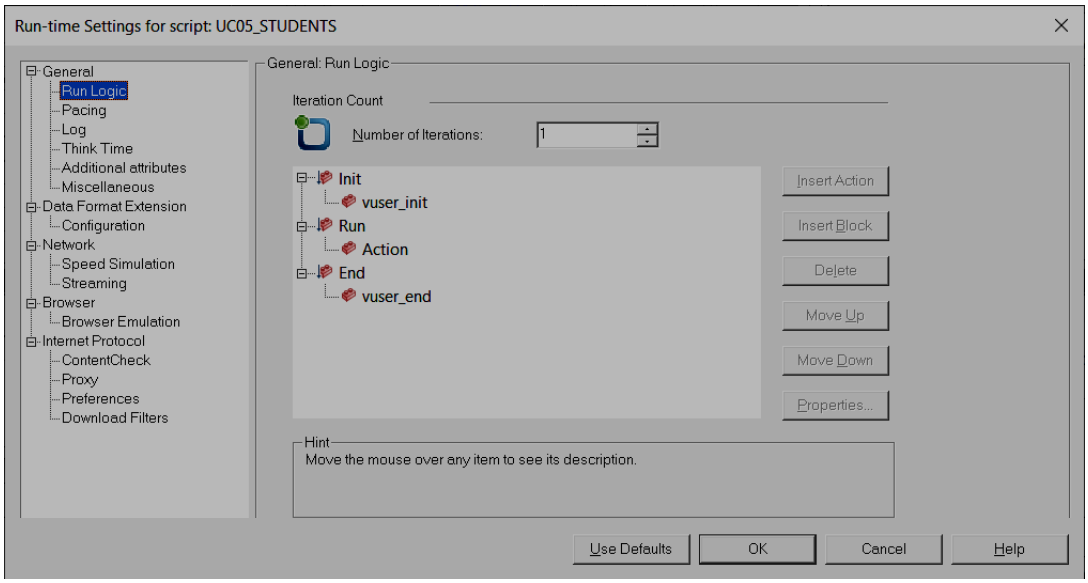


Рисунок 3.14 – Налаштування у вкладці «Run-time Settings»

Активуємо почергово опції «Pacing» (Стимуляція), «Log» (Журнал) та «Think Time» (Час на роздуми) та здійснюємо налаштування у цих вкладках (рисунки 3.15...3.17 відповідно). Ці налаштування виставляються індивідуально для кожного скрипту (програми).

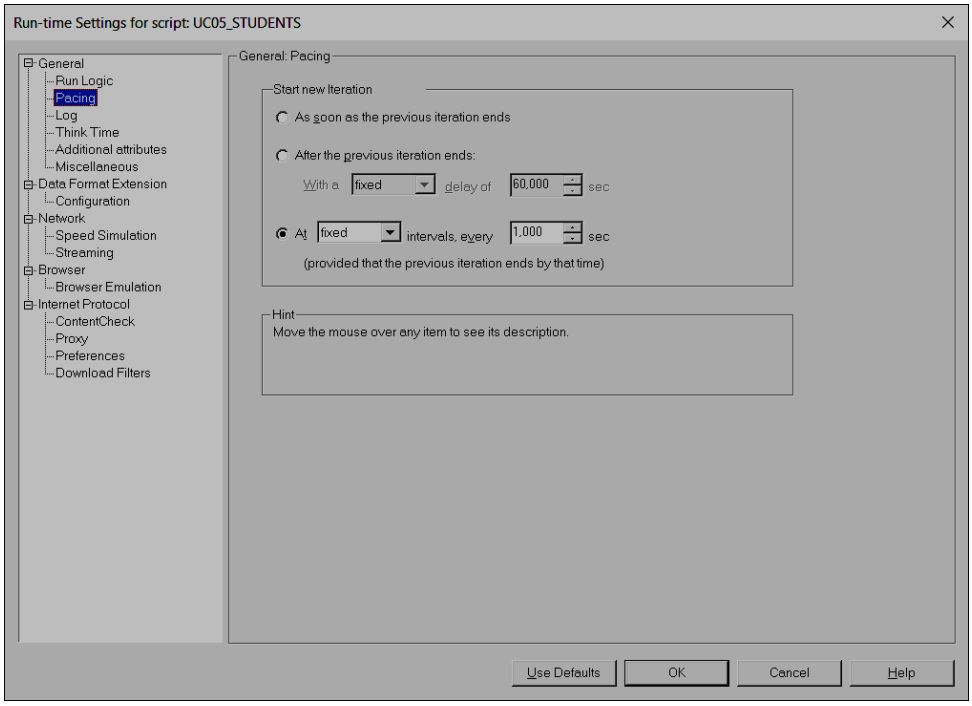


Рисунок 3.15 – Налаштування у вкладці «Pacing»

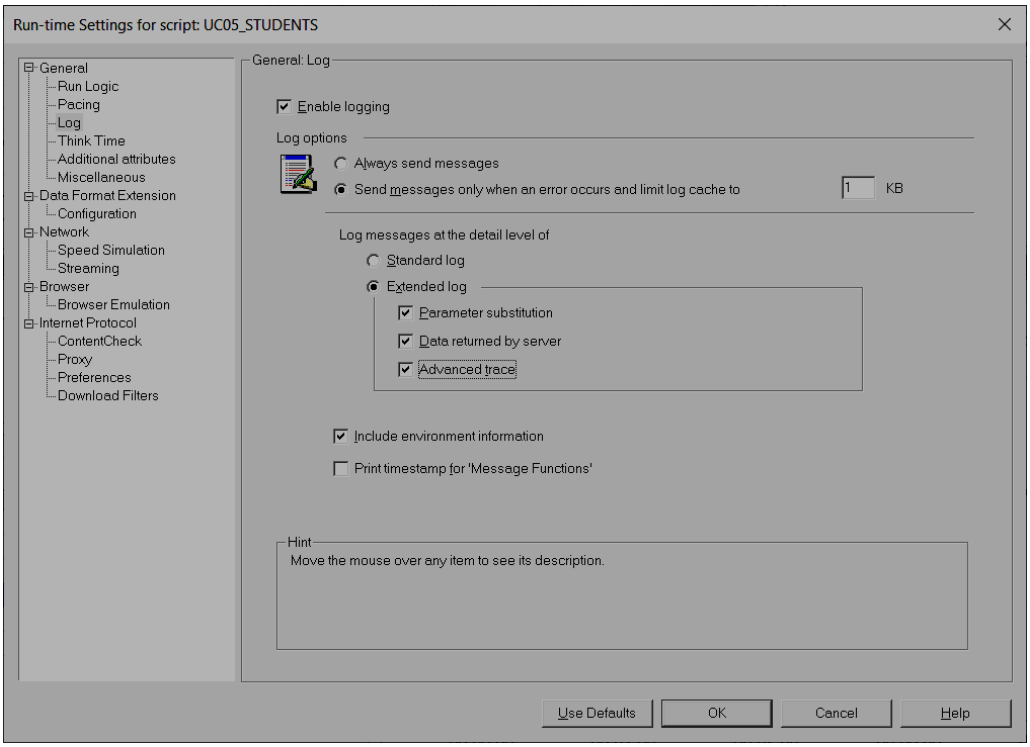


Рисунок 3.17 – Налаштування у вкладці «Log»

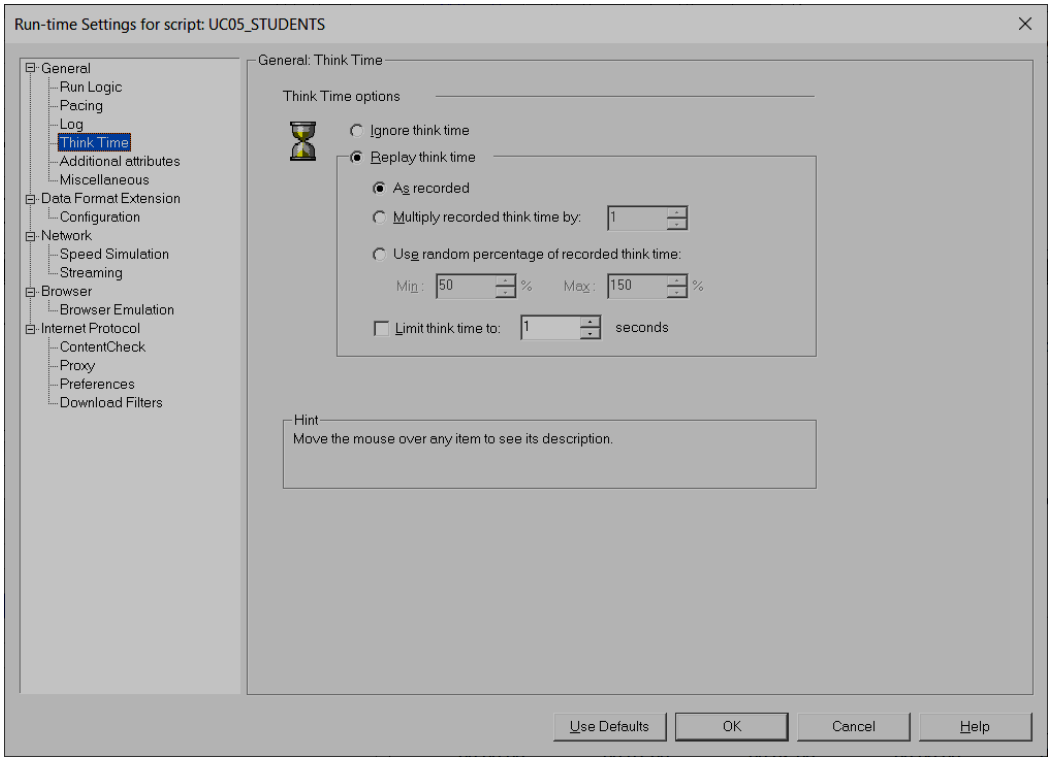


Рисунок 3.18 – Налаштування у вкладці «Think Time»

4 АНАЛІЗ РЕЗУЛЬТАТІВ ТЕСТУВАННЯ

4.1 Підготовка до аналізу результатів

Як було зазначено вище, для аналізу результатів тестування в даній роботі доцільно застосувати модуль Analysis.

Після відкриття модуля Analysis ми автоматично попадемо на головну сторінку (рисунок 4.1).

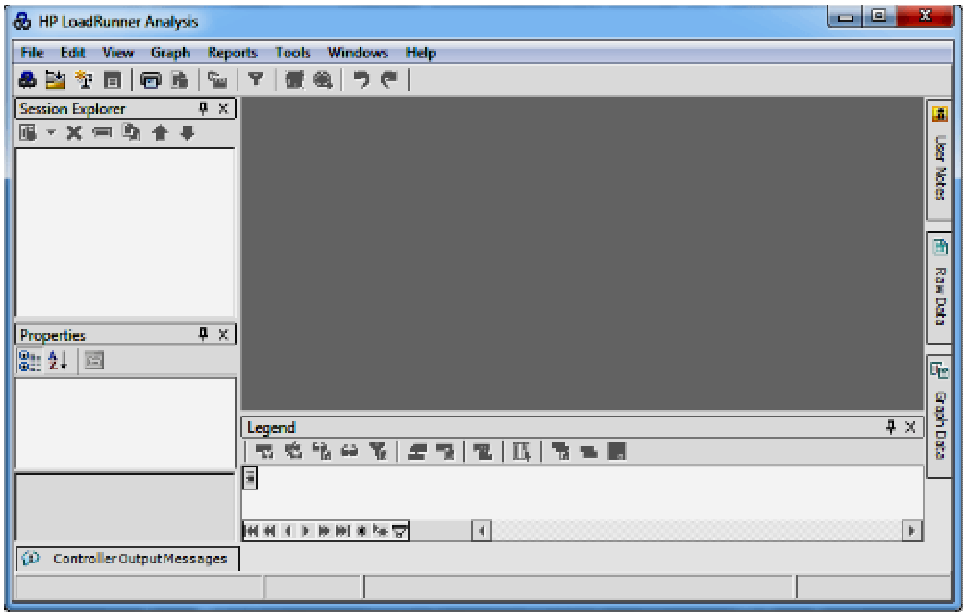


Рисунок 4.1 – Головна сторінка Analysis

Для створення нової сесії аналізу результатів, в меню «File» (рисунок 4.2) вибираємо опцію «New»

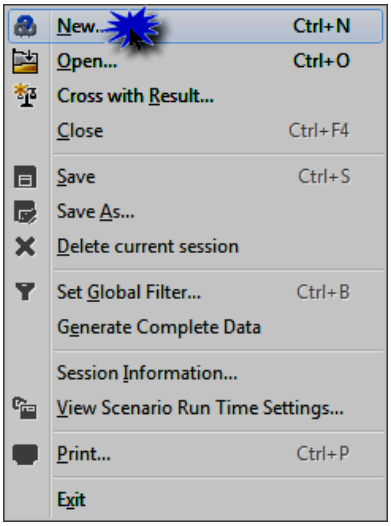


Рисунок 4.2 – Створення нової сесії аналізу результатів

Після цього відкриється діалогове вікно (рисунок 4.3), в якому потрібно вибрати необхідний для аналізу файл.

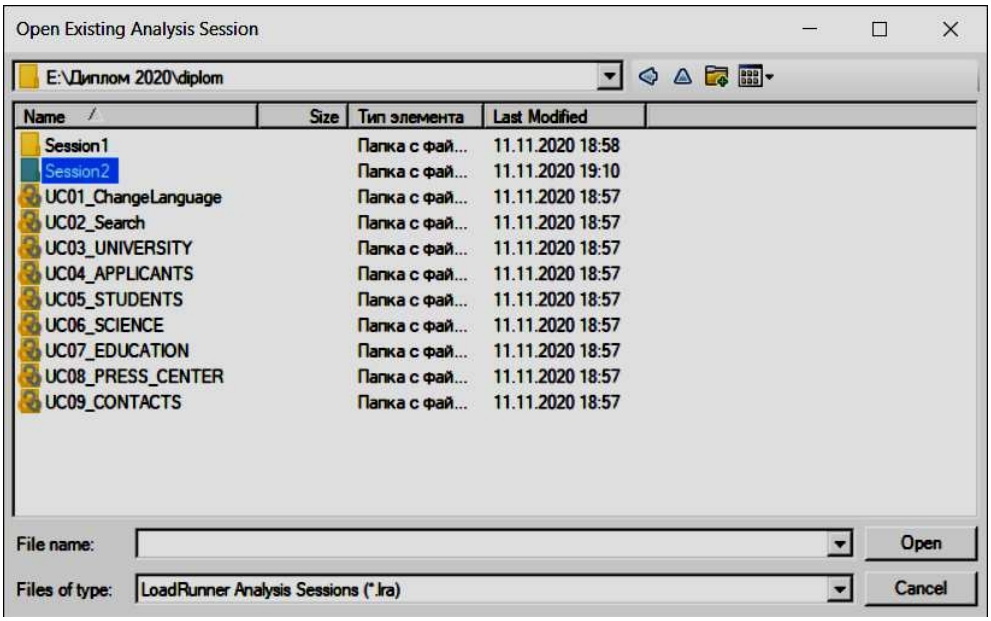


Рисунок 4.3 – Вікно вибору потрібних для Analysis файлів

Після вибору необхідного файлу, модуль Analysis спочатку перевіряє доступний дисковий простір, щоб переконатися, що під час процесу нам вистачить місця. Це пов'язано з тим, що час сеансу, особливо для складних сценаріїв, може бути дуже великим. Наприклад, коли підключено кілька сотень скриптів, то процес аналізу може досягати десятків годин.

Відразу після підтвердження наявності місця на диску відкривається вікно «цільового аналізу» (рисунок 4.4). Це свідчить про те, що модуль Analysis почав діяти.

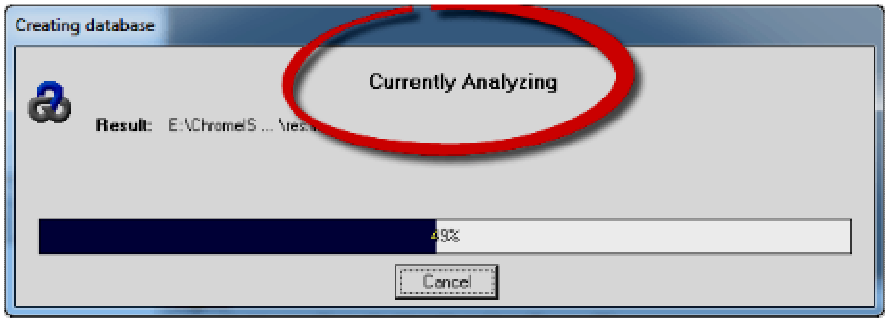


Рисунок 4.4 – Вікно прогресу цільового аналізу

Час, який займає вказане вище вікно, залежить від розміру папки результатів (або статистики, що міститься в цій папці)

Якщо аналіз займає надто багато часу або застряє, зупинити тестування можна натиснувши на кнопку скасування.

Після успішного завершення аналізу модуль Analysis автоматично відобразить вікно шаблону за замовчуванням (рисунок 4.5).

Рисунок 4.5 – Вікно шаблону за замовчуванням

Після завершення операцій відобразиться головне вікно сесії (рисунок 4.6)

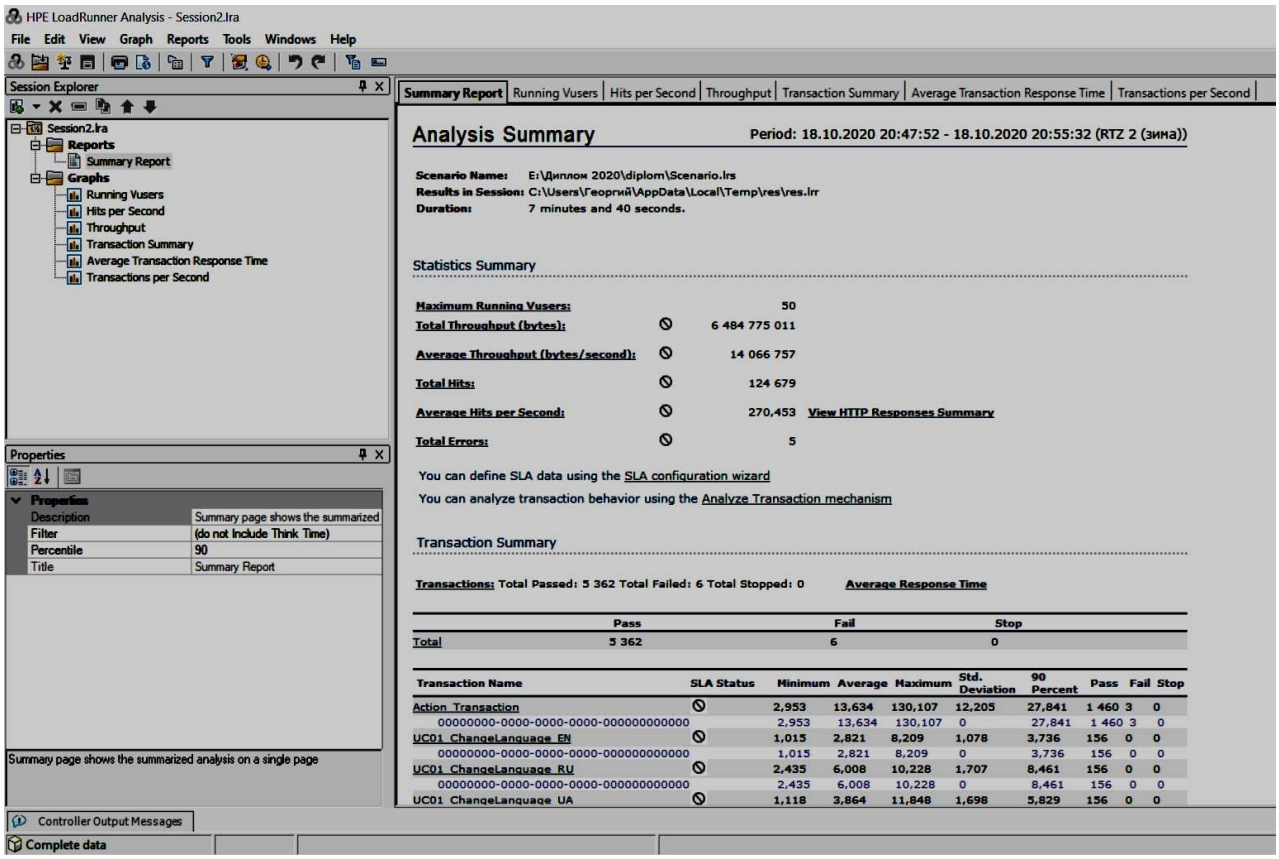


Рисунок 4.6 – Вікно поточної сесії

У цьому вікні доступний для перегляду час проведення тестування (див. поле Duration). Також можливо переглянути загальні суму транзакцій за секунду (див. поле Average Hits per Seconds), та загальну кількість помилок які вдалося відстежити під час роботи тесту (див. поле Total Errors).

4.2 Результати запуску тесту UC01_ChangeLanguage

Програма UC01_ChangeLanguage імітує дії користувача при зміні мови на сайті NURE, всього на сайті доступні 3 мови: українська, російська та англійська.

За результатами розрахунків (див. рисунок 3.6) плановане навантаження складає 15 tps. За допомогою модуля Analysis переглянемо чи вдалося досягти планованих показників, для цього відкриємо графік «Transaction per Second» (рисунок 4.7)

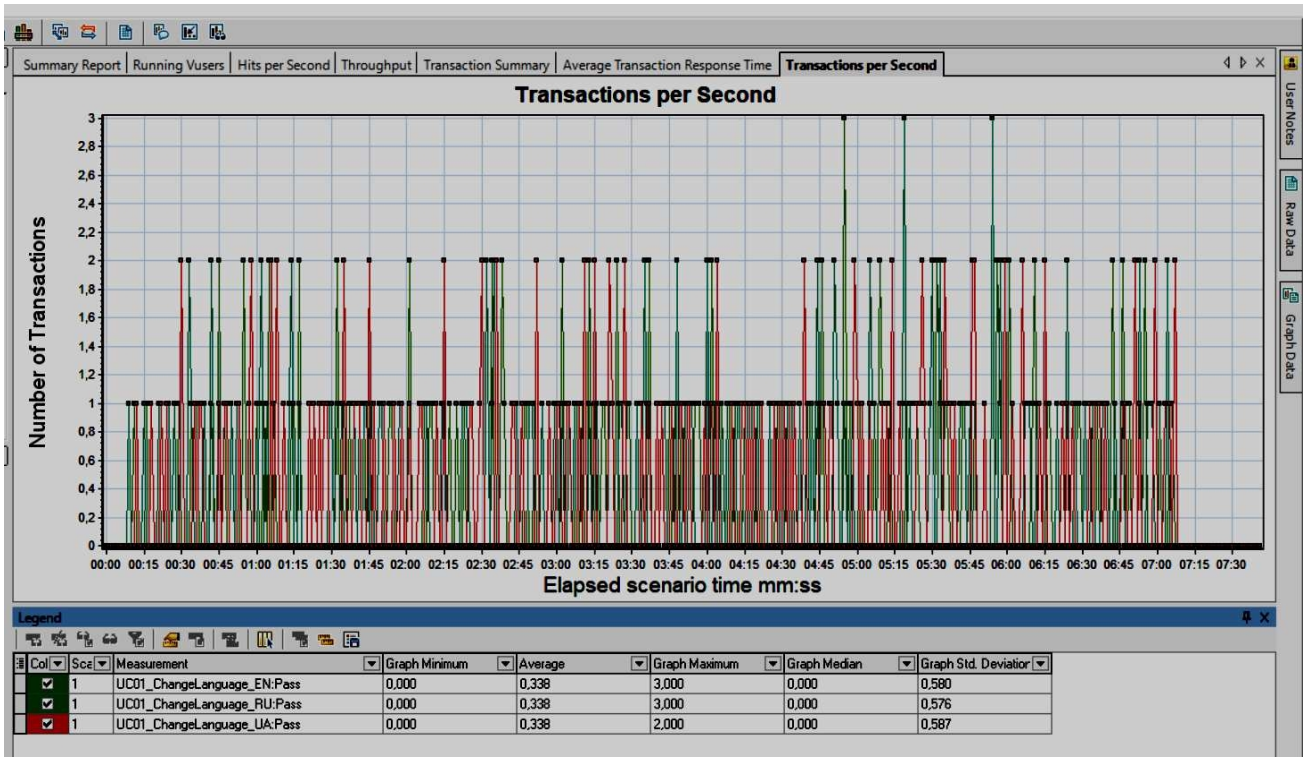


Рисунок 4.7 – Транзакції за секунду (Transaction per Second), UC01_ChangeLanguage

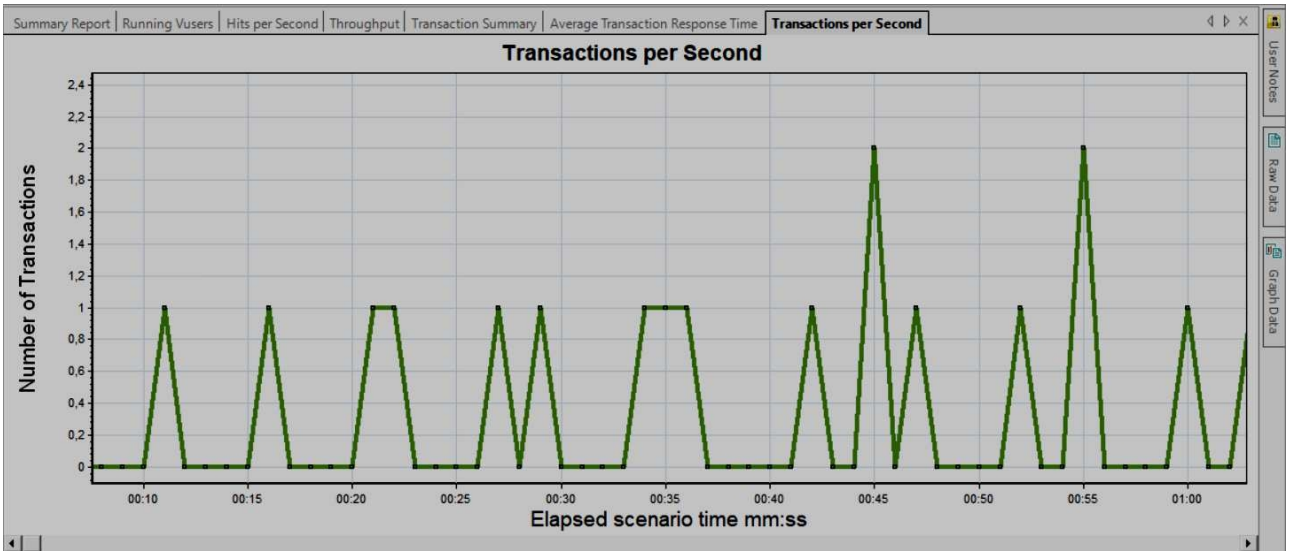
На графіках за горизонталлю відображений час тестування, за вертикаллю кількість вдало відпрацювавших транзакцій. В таблиці під графіком наведено розраховані показники для кожної мови:

– Graph Minimum – відображає мінімальний час проходженні транзакції;

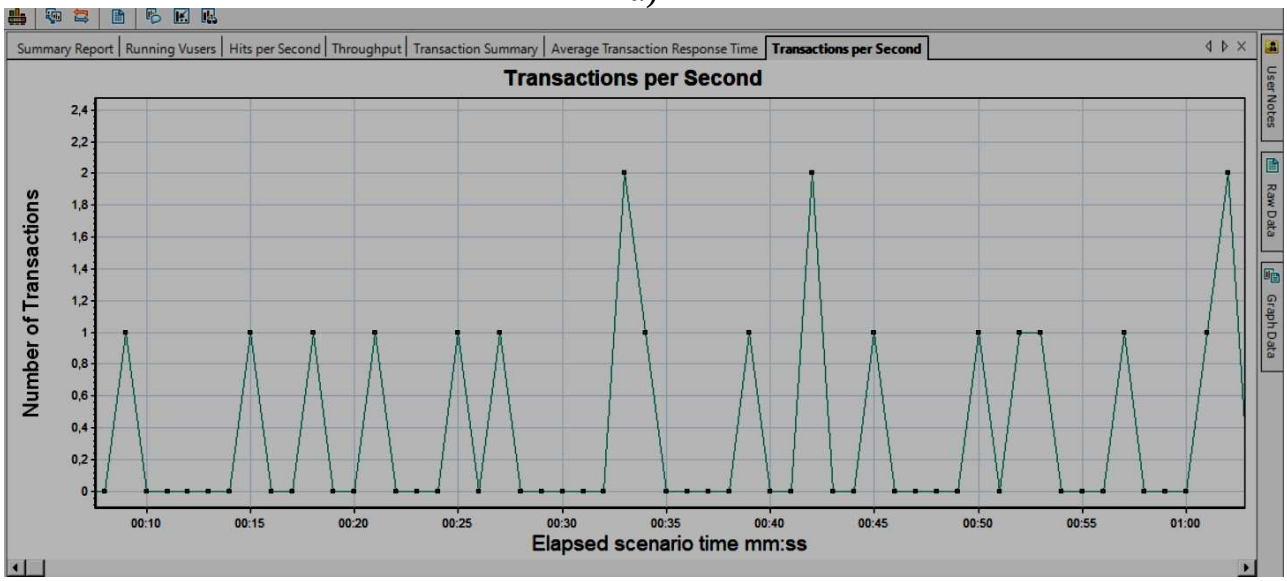
– Averang – відображає середній час проходженні транзакції;

– Graph Maximum – відображає максимальний час проходженні транзакції.

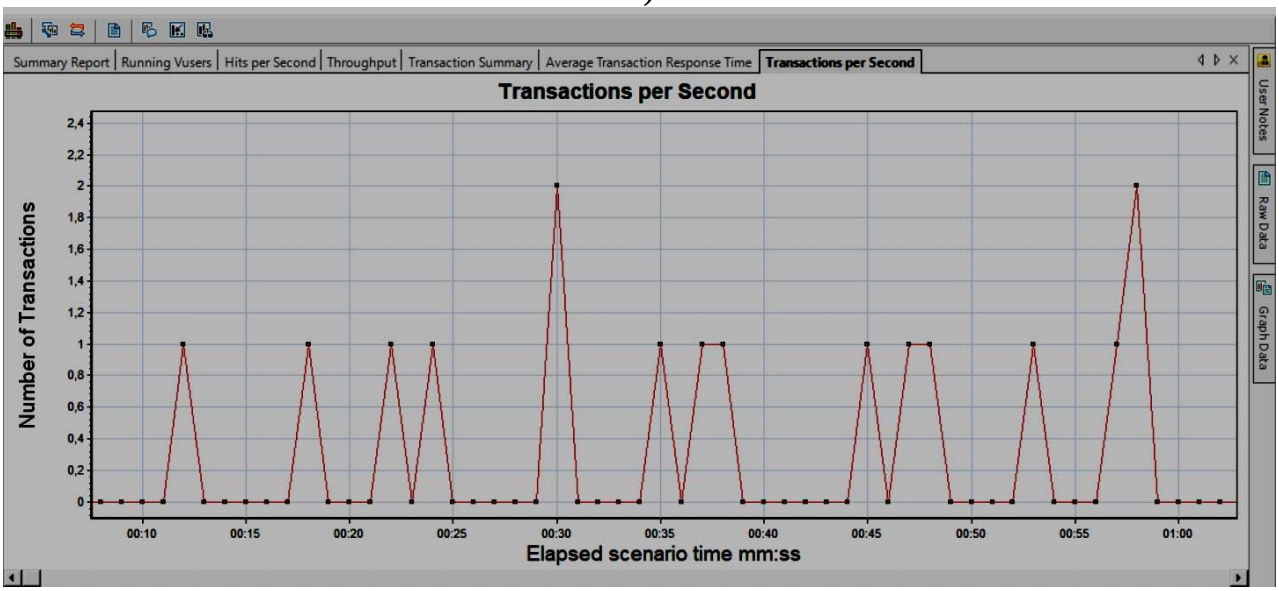
Для того, щоб зробити детальний аналіз отриманих результатів переглянемо графік за кожною мовою окремо, зі збільшеним масштабом (рисунок 4.8).



а)



б)



в)

Рисунок 4.8 – Транзакції за секунду (Transaction per Second): а - англійська мова, б – російська мова, в – українська мова

Візьмемо відрізок часу з 00.10 до 00.15. З графіку (рисунок 4.8,а) видно, що при використанні англійської мови, за цей час отримали одну вдалу транзакцію на 11-й секунді, а розрахунки (див. рисунок 3.6) показують, що було заплановано 15 транзакцій за секунду. Тобто за 5 секунд мали отримати 75 успішно пройдених транзакцій.

При використанні російської мови (рисунок 4.8,б) також отримали одну вдалу транзакцію на 15-й секунді, а при використанні української мови (рисунок 4.8,в) - одну вдалу транзакцію на 12-й секунді.

Як видно з колонки «Average» (див. рисунок 4.7), в середньому вдалося подати 0,338 tps з 15 планованих.

Для того, щоб розібратися в причинах такої поведінки, відкриємо графік відгуків транзакцій «Average Transaction Response Time» (рисунок 4.9).

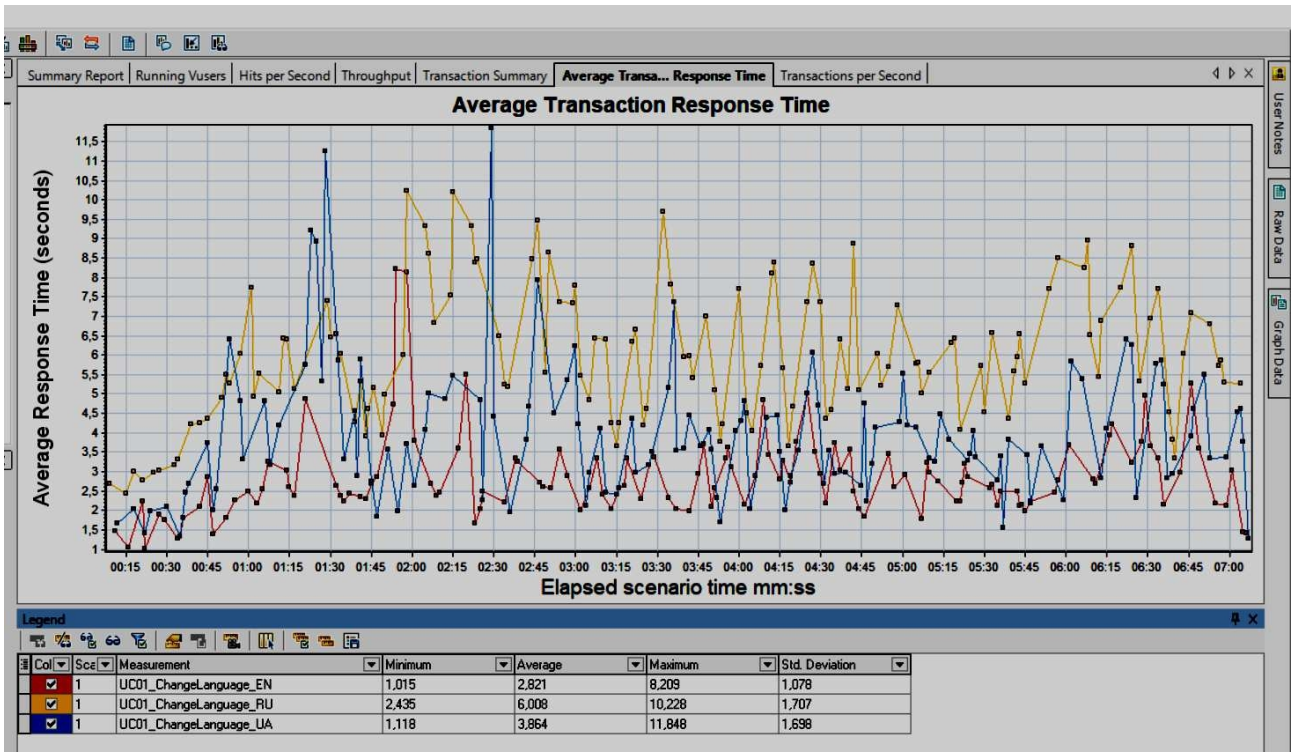


Рисунок 4.9 – Відгук транзакцій (Average Transaction Response Time), UC01_ChangeLanguage

За горизонталлю відображений час тестування, за вертикаллю час відгуку транзакцій.

Як видно з графіку та колонки «Average», в середньому відгук за англійською мовою був 2,821 с, з максимальним значенням 8,209 с. Відгук за російсь-

кою мовою був 6,002 с, з максимальним значенням 10,228 с, а відгук за українською мовою був 3,864 с, з максимальним значенням 11,848 с.

В результаті проведеного аналізу можна зробити висновок, що саме російська мова працює з найбільшими проблемами, тому і має такий великий час відгуку і саме великий час відгуку став причиною того, що потрібне навантаження подати не вдалося.

4.3 Результати запуску тесту UC02_Search

Програма UC02_Search імітує дії користувача який використовує пошук по сайту NURE.

Графік транзакцій за секунду наведено на рисунку 4.10.

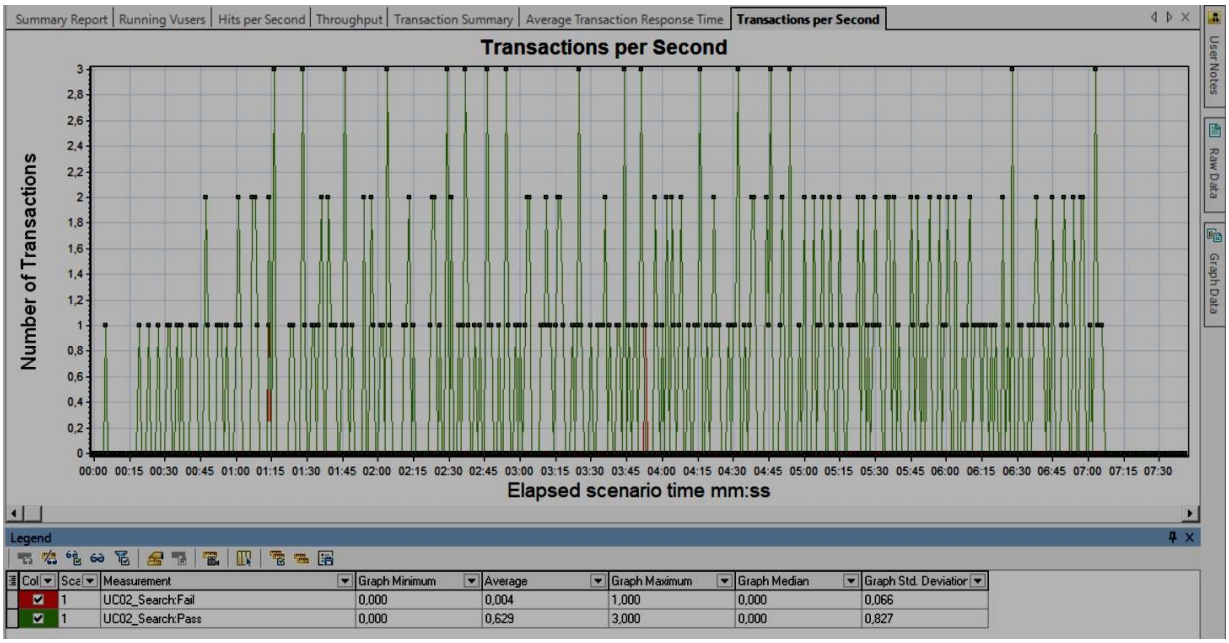


Рисунок 4.10 – Транзакції за секунду UC02_Search

За горизонталлю відображений час тестування, за вертикаллю кількість вдало відпрацювавших транзакцій.

Як видно з графіку та колонки «Average», в середньому вдалося подати 0,629 tps з 12 tps планованих (див. рисунок 3.6). Також з’явився запис «UC02_Search:Fail», який свідчить про те, що деякі запити буди втрачені під час обробки.

Розглянемо графік втрачених запитів більш ретельно рисунок 4.11.

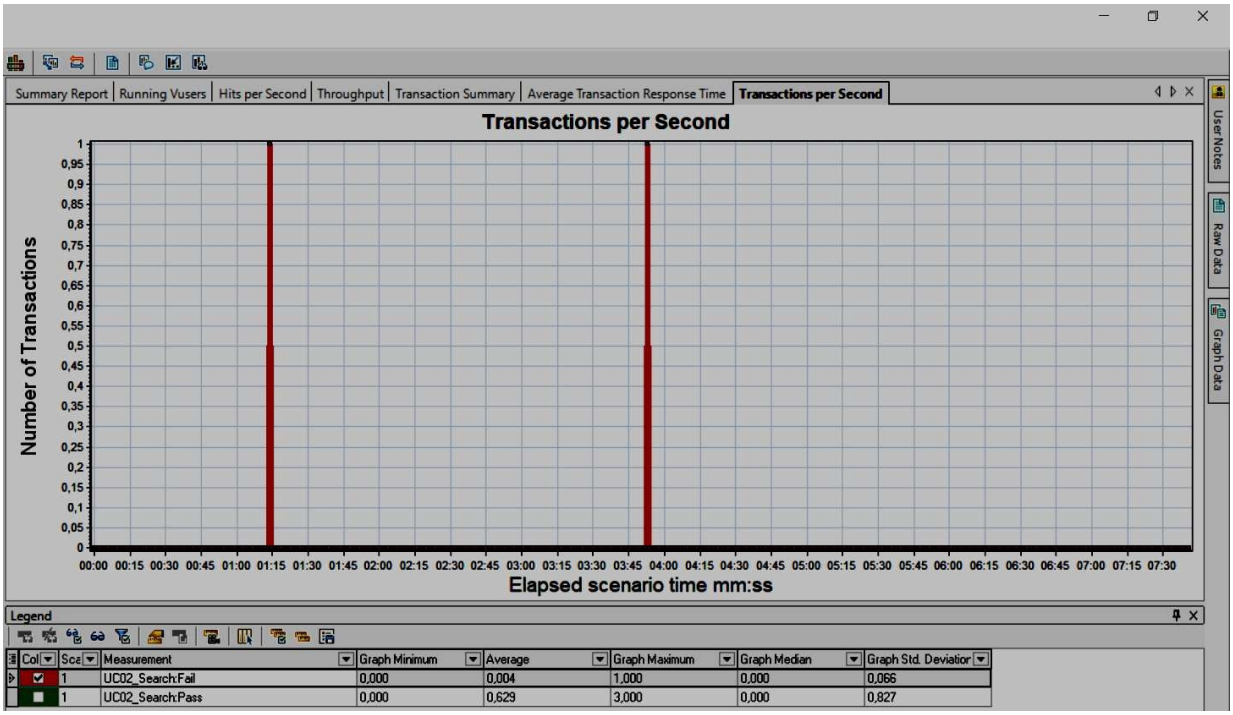


Рисунок 4.11 – Втрачені запити UC02_Search

Як видно всього було втрачено 2 запити за 10 хвилин тесту, це свідчить про погану оптимізацію процедури пошуку для праці під навантаженням. Втрата даних які вводить користувач - це серйозна проблема яку потрібно локалізувати та виправляти.

Для того, щоб розібратись в причинах того, чому не вдалося подати плановане навантаження, відкриємо графік відгуків транзакцій «Average Transaction Response Time» (рисунок 4.12).

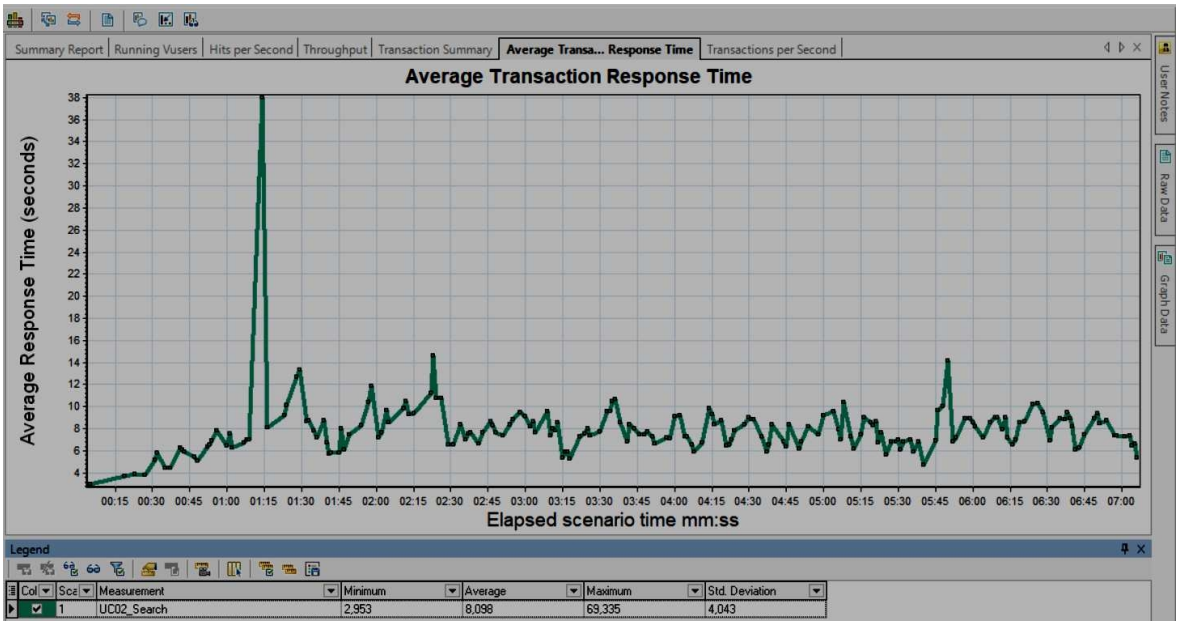


Рисунок 4.12 – Відгук транзакцій UC02_Search

Як видно з графіку та колонки «Average», в середньому відгук пошуку був 8,098 с, з максимальним значенням 69,335 с. Такий великий час відгуку і став причиною того, що потрібне навантаження подати не вдалося.

4.4 Результати запуску тесту UC03_UNIVERSITY

Програма UC03_UNIVERSITY імітує дії користувача при послідовному переході користувача з головної сторінки на вкладку UNIVERSITY і далі на вкладку ABOUT_UNIVERSITY (рисунок 4.13).

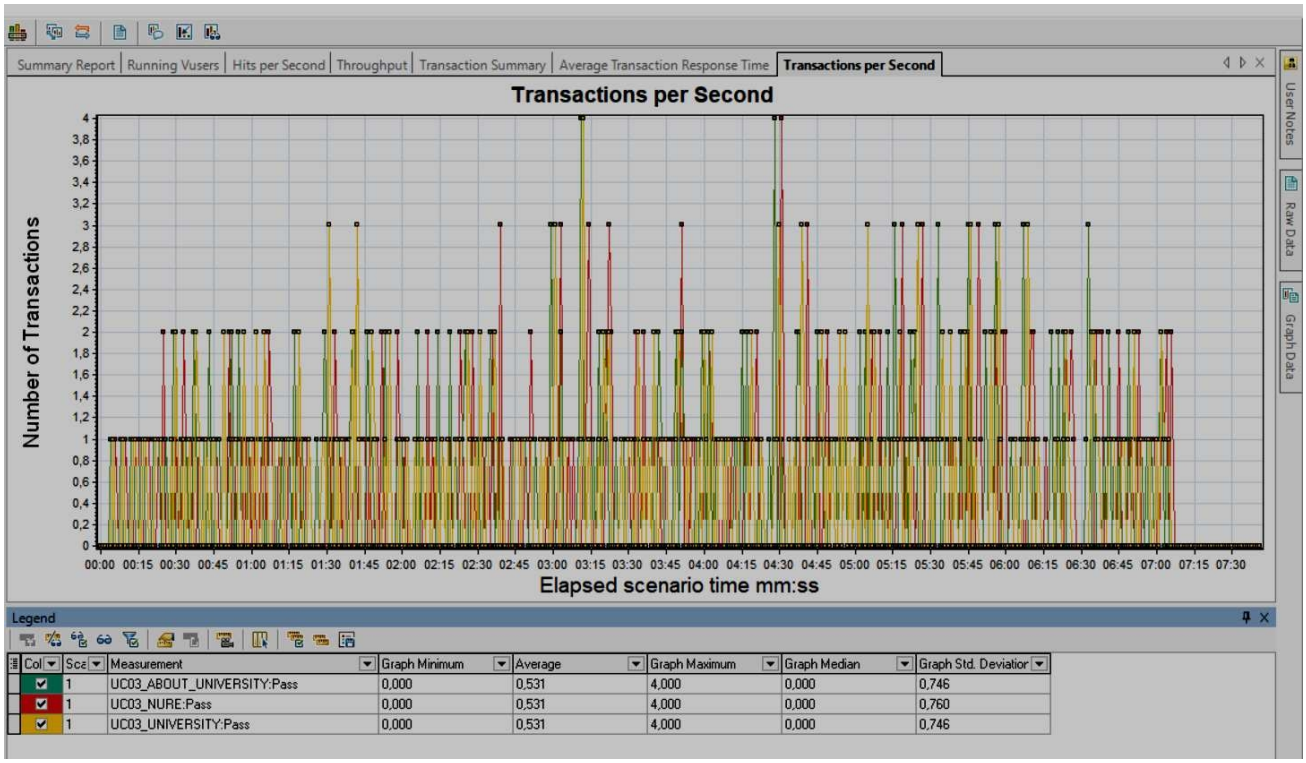


Рисунок 4.13 – Транзакції за секунду UC03_UNIVERSITY

Як видно з графіку та колонки «Average», в середньому вдалося подати 0,531 tps з 18 tps планованих (див. рисунок 3.6).

Скрипт UC03_UNIVERSITY складається з трьох транзакцій:

– UC03_UNIVERSITY, ABOUT_UNIVERSITY (рисунок 4.14);

– UC03_UNIVERSITY, NURE (рисунок 4.15);

– UC03_UNIVERSITY, UNIVERSITY (рисунок 4.16).

Для того, щоб зробити детальний аналіз потрібно переглянути графік за кожною з транзакцій окремо, зі збільшеним масштабом.

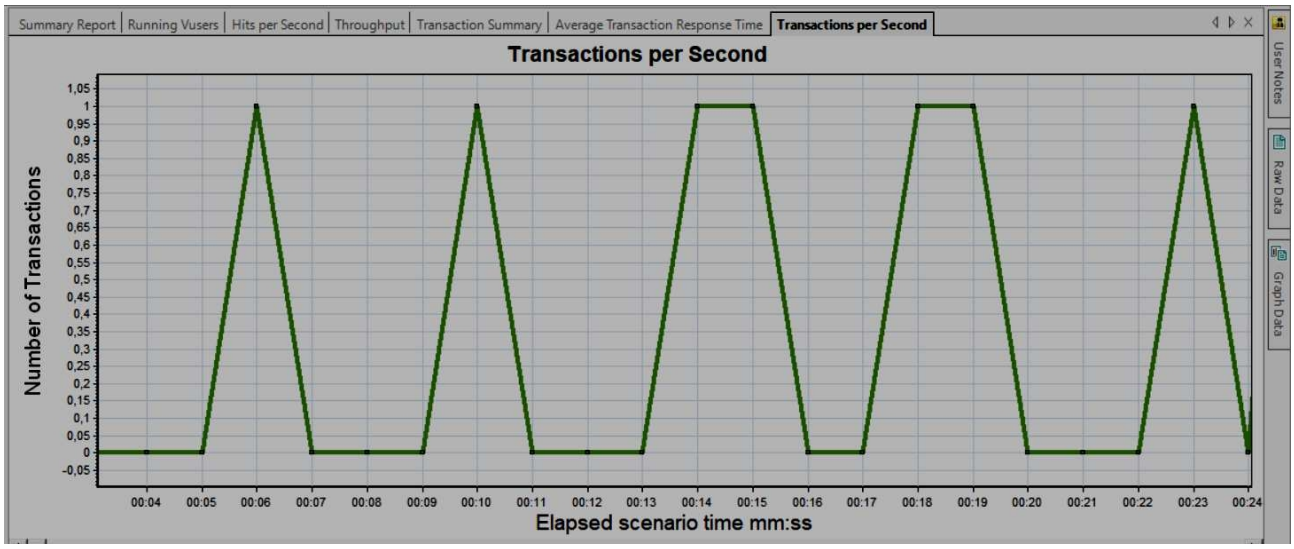


Рисунок 4.14 – Транзакції за секунду UC03_UNIVERSITY, ABOUT_UNIVERSITY

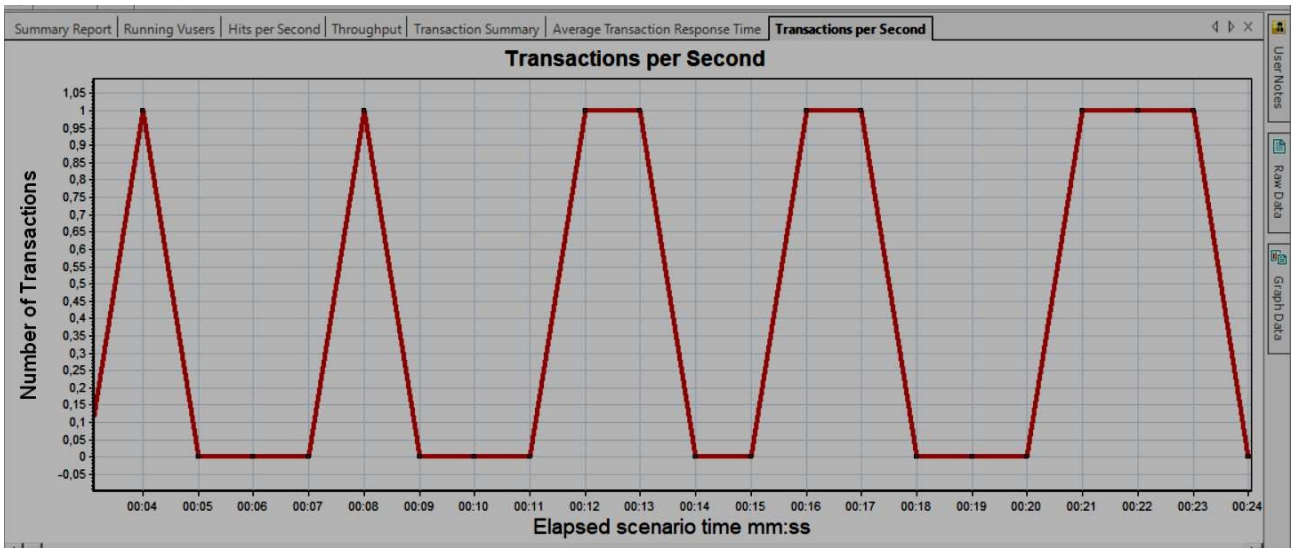


Рисунок 4.15 – Транзакції за секунду UC03_UNIVERSITY, NURE

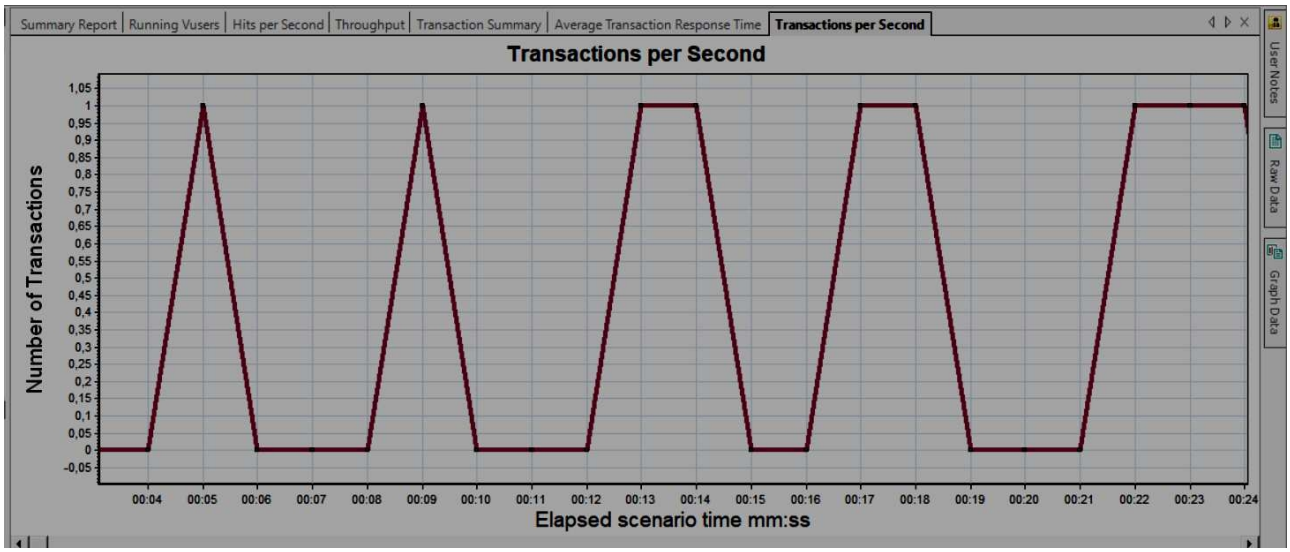


Рисунок 4.16 – Транзакції за секунду UC03_UNIVERSITY, UNIVERSITY

На рисунку 4.14, при виконанні транзакції UC03_UNIVERSITY, ABOUT_UNIVERSITY на відрізку часу з 00.05 до 00.10 видно дві вдалі транзакції на 6-й і 10-й секундах з планових 18, а за 5 секунд мали отримати 90 успішно пройдених транзакцій.

При виконанні транзакції UC03_UNIVERSITY, NURE (рисунок 4.15) за той же відрізок часу видно одну вдалу транзакцію на 8-й секунді замість очікуваних 90.

При виконанні транзакції UC03_UNIVERSITY, UNIVERSITY (рисунок 4.16) за той же відрізок часу видно дві вдалі транзакції на 5-й і 9-й секундах замість очікуваних 90.

Для того, щоб розібратися в причинах такої поведінки, відкриємо графік відгуків транзакцій «Average Transaction Response Time» (рисунок 4.17).

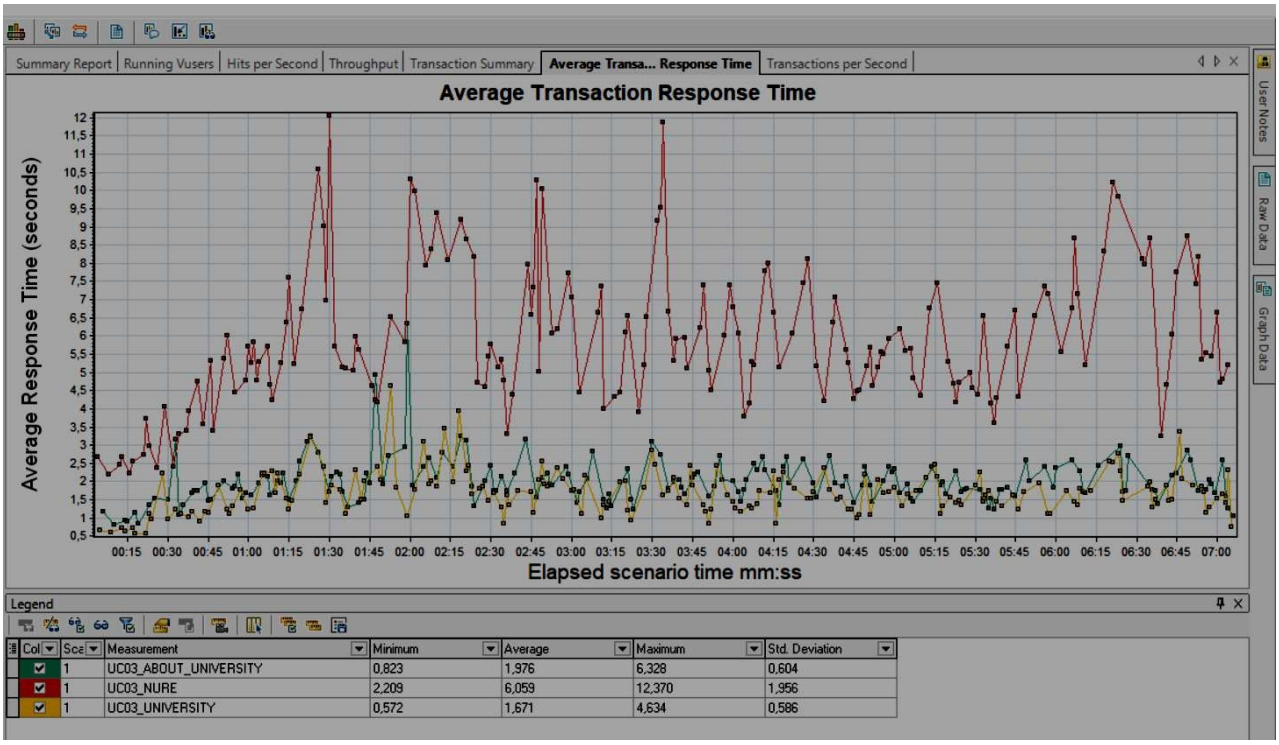


Рисунок 4.17 – Відгук транзакцій UC03_UNIVERSITY

За горизонталлю відображений час тестування, за вертикаллю час відгуку транзакцій.

Як видно з графіку та колонки «Average», в середньому відгук за відкриттям вкладки «ABOUT_UNIVERSITY» склав 1,976 с, з максимальним значенням 6,328 с.

Відгук за відкриттям головної вкладки сайту NURE склав 6,059 с, з максимальним значенням 12,370 с.

Відгук за відкриттям вкладки «UNIVERSITY» склав 1,671 с, з максимальним значенням 4,634 с.

Такий великий час відгуку і став причиною того, що потрібне навантаження подати не вдалося.

Як видно, основною причиною невдалого запуску цього скрипту (програми) стала саме головна сторінка сайту NURE з середнім відгуком 6,059 с, яка і затримувала роботу інших запитів.

4.5 Результати запуску тесту UC04_APPLICANTS

Програма UC04_APPLICANTS імітує дії користувача при послідовному переході користувача з головної сторінки на вкладку APPLICANTS і далі на вкладку EDUCATION_IN_ENGLISH.

Навантаження: плановане навантаження 15 tps (див. рисунок 3.6). За допомогою Analysis переглянемо чи вдалося досягти планованих показників, для цього відкриємо графік «Transaction per Second» (рисунок 4.18):

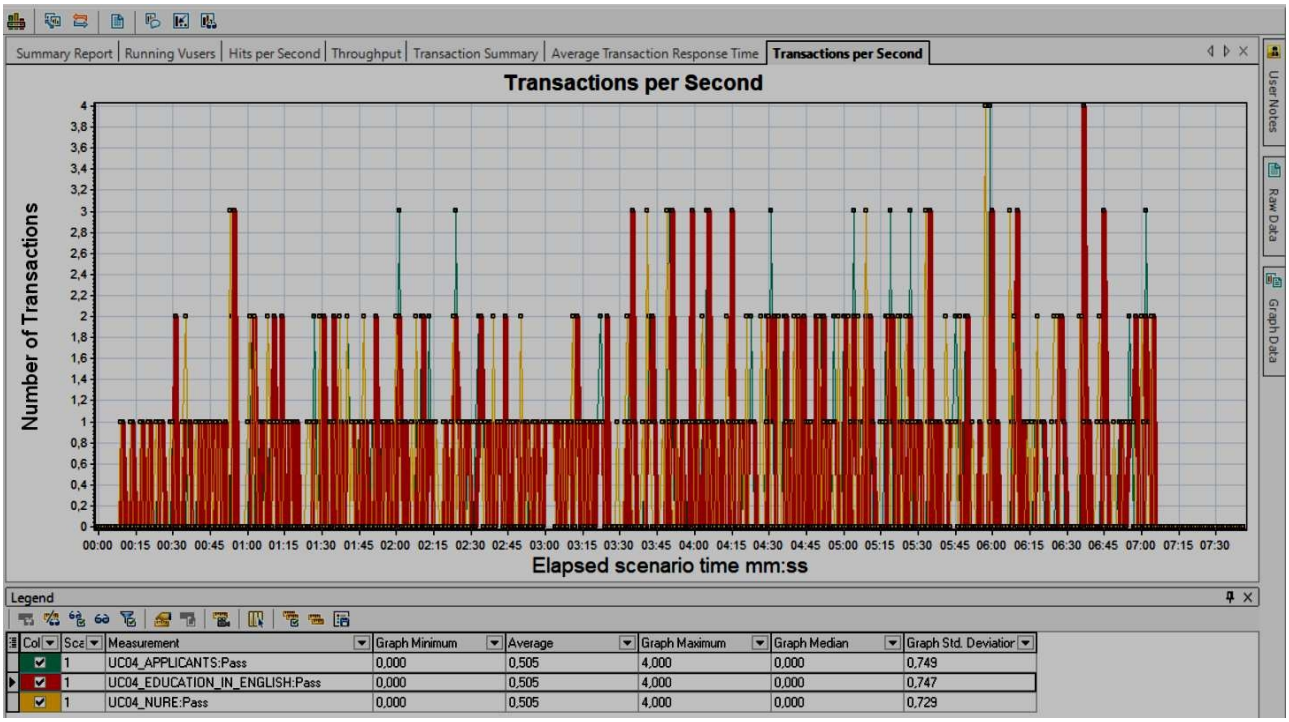


Рисунок 4.18 – Транзакції за секунду (Transaction per Second), UC04_APPLICANTS

Як видно з графіку та колонки «Average», в середньому вдалося подати 0,505 tps з 15 планованих.

Скрипт UC04_APPLICANTS складається з трьох транзакцій, для того, щоб зробити детальний аналіз потрібно переглянути графік за кожною з транзакцій окремо, зі збільшеним масштабом (рисунки 4.19...4.21).

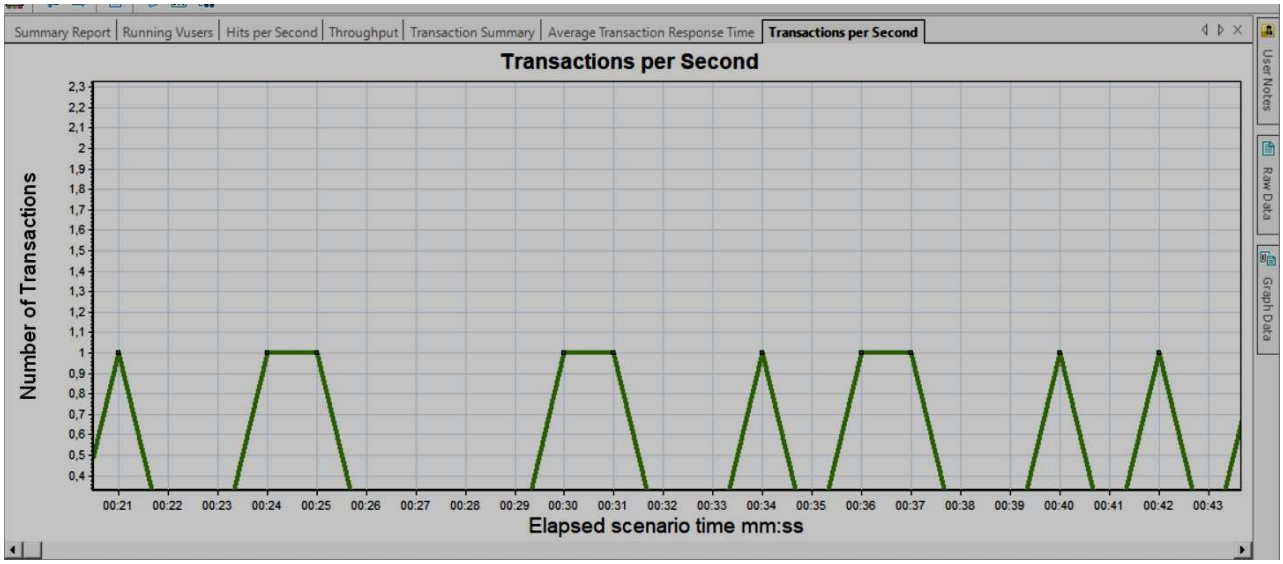


Рисунок 4.19 – Транзакції за секунду (Transaction per Second), UC04_APPLICANTS, APPLICANTS

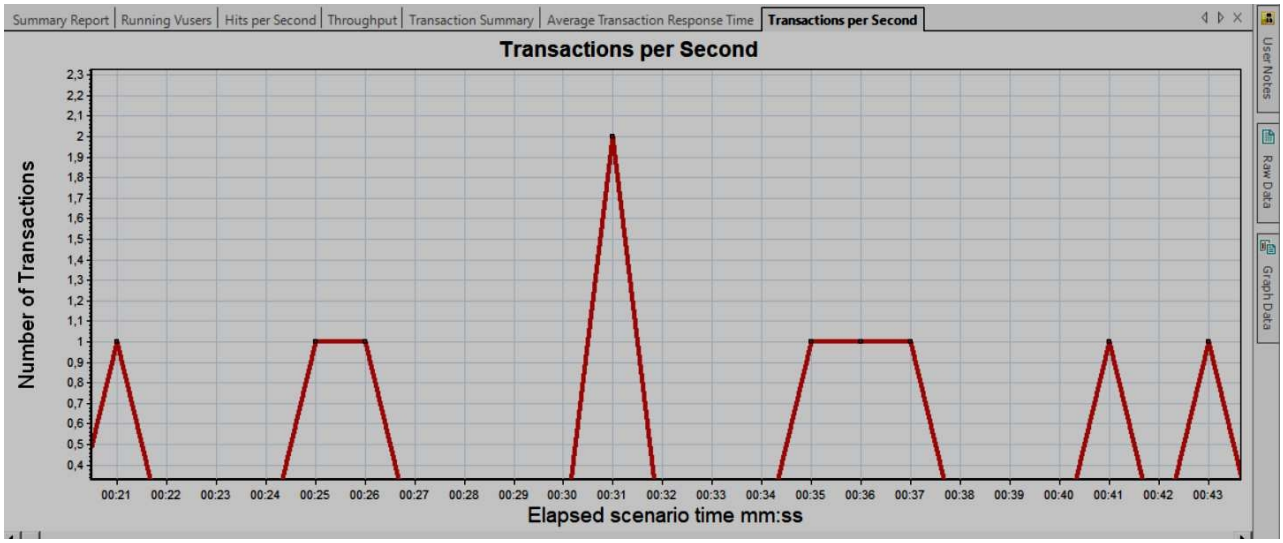


Рисунок 4.120 – Транзакції за секунду (Transaction per Second), UC04_APPLICANTS, EDUCATION_IN_ENGLISH

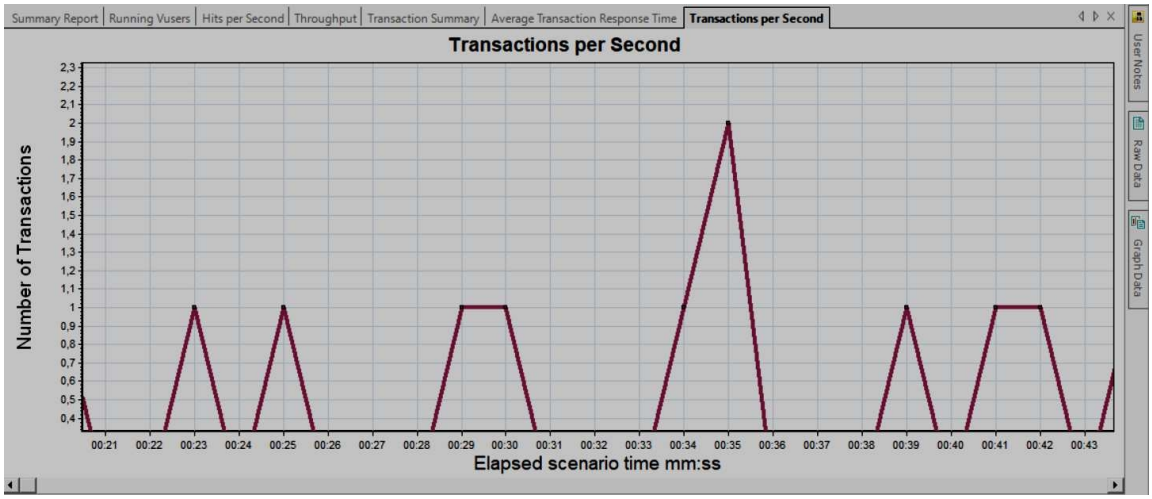


Рисунок 4.21 – Транзакції за секунду (Transaction per Second), UC04_APPLICANTS, NURE

Як видно з графіків на відрізку часу з 00.25 до 00.30 видно дві вдалі транзакції UC04_APPLICANTS, APPLICANTS на 25-й, та 30-й секундах при планових 15 транзакцій, тобто за 5 секунд мали отримати 75 успішно пройдених транзакцій.

На тому ж відрізку часу видно дві вдалі транзакції UC04_APPLICANTS, EDUCATION_IN_ENGLISH на 25-й та 26-й секундах та три вдалі транзакції UC04_APPLICANTS, NURE на 25-й, 29-й та 30-й секундах.

Потрібно розібратися в причинах такої поведінки, для цього відкриємо графік «Average Transaction Response Time» (рисунок 4.22).

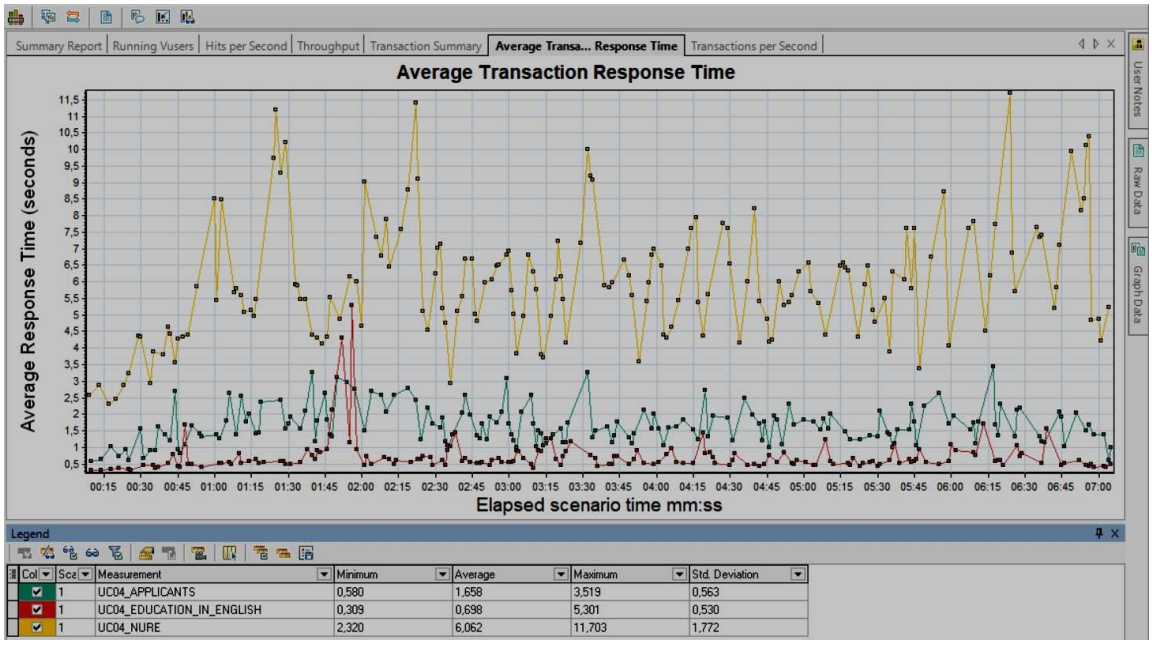


Рисунок 4.22 – Відгук транзакцій (Average Transaction Response Time), UC04_APPLICANTS

Як видно з графіку та колонки «Average», в середньому відгук за відкриттям вкладки «APPLICANTS» склав 1,658 с, з максимальним значенням 3,519 с.

Відгук за відкриттям вкладки «EDUCATION_IN_ENGLISH» склав 0,698 с, з максимальним значенням 5,301 с.

Відгук за відкриттям головної вкладки сайту NURE склав 6,062 с, з максимальним значенням 11,703 с.

Такий великий час відгуку і став причиною того, що потрібне навантаження подати не вдалося.

Необхідно звернути увагу на те, що причиною невдалого запуску цього скрипту (програми) стала знову головна сторінка сайту NURE яка і затримувала роботу інших запитів.

4.6 Результати запуску тесту UC05_STUDENTS

Програма UC05_STUDENTS імітує дії користувача при послідовному переході користувача з головної сторінки на вкладку STUDENTS і далі на вкладку Timetable_of_Classes для перегляду розкладу.

За допомогою Analysis переглянемо чи вдалося досягти планованих показників, для цього відкриємо графік «Transaction per Second» (рисунок 4.23).

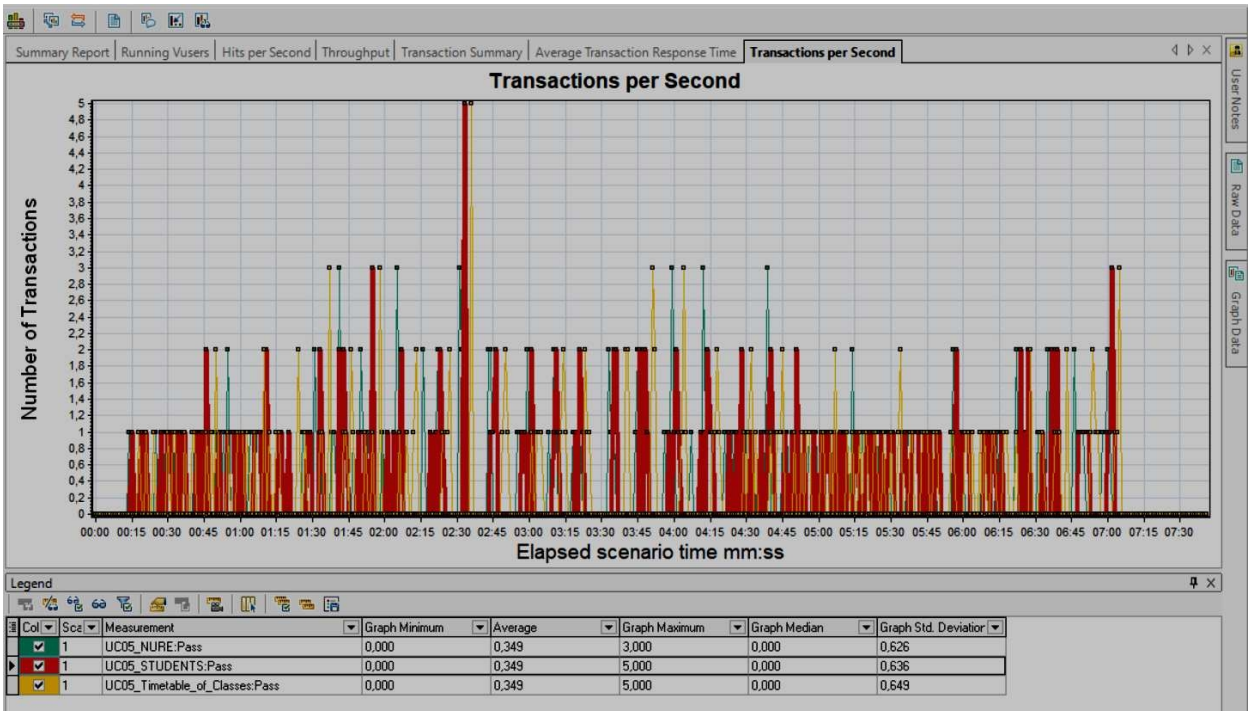


Рисунок 4.23 – Транзакції за секунду (Transaction per Second), UC05_STUDENTS

За горизонталлю відображений час тестування, за вертикаллю кількість вдало відпрацювавших транзакцій.

Скрипт UC05_STUDENTS складається з трьох транзакцій:

– UC05_STUDENTS, NURE;

– UC05_STUDENTS, STUDENTS;

– UC05_STUDENTS, Timetable_of_Classes.

Як видно з графіку та колонки «Average», в середньому вдалося подати 0,349 tps з 15 планованих для кожної із зазначених транзакцій.

Потрібно розібратися в причинах такої поведінки, для цього відкриємо графік «Average Transaction Response Time» (рисунок 4.24).

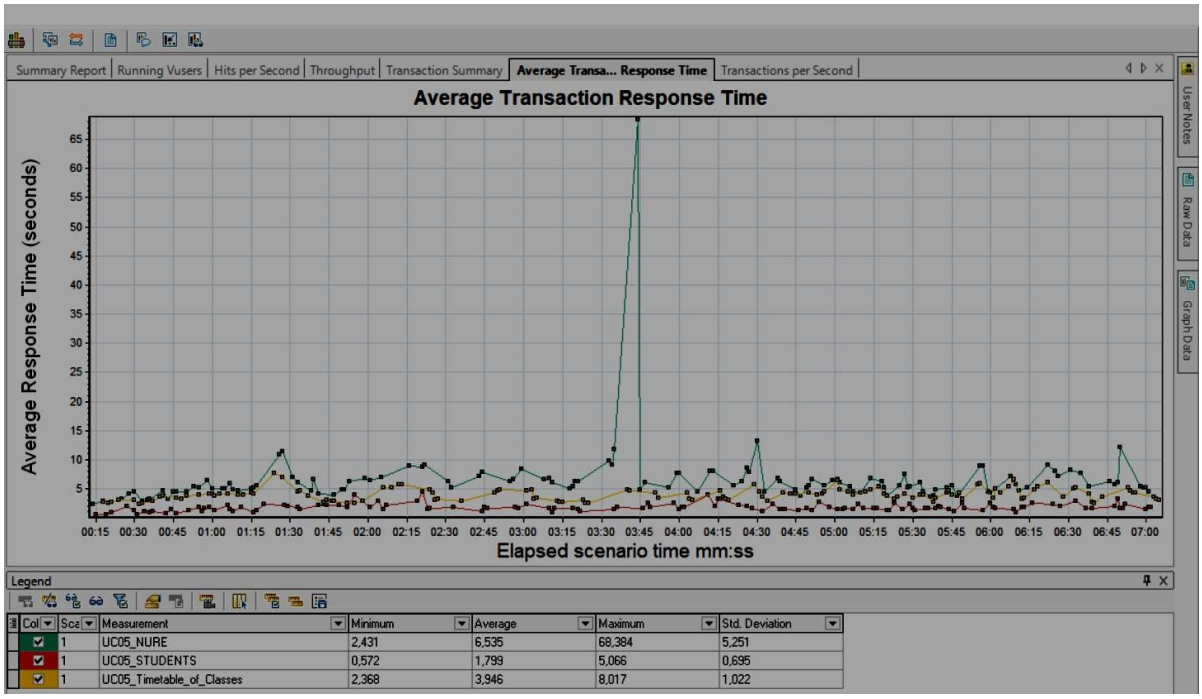


Рисунок 4.24 – Відгук транзакцій (Average Transaction Response Time), UC05_STUDENTS

За горизонталлю відображений час тестування, за вертикаллю час відгуку транзакцій.

Як видно з графіку та колонки «Average», в середньому відгук за відкриттям головної вкладки сайту NURE склав 6,535 с, з максимальним значенням 68,384 с.

Відгук за відкриттям вкладки «STUDENTS» склав 1,799 с, з максимальним значенням 5,066 с.

Відгук за відкриттям вкладки «Timetable_of_Classes» склав 3,946 с, з максимальним значенням 8,017 с. Такий великий час відгуку і став причиною того, що потрібне навантаження подати не вдалося.

Причиною невдалого запуску цього скрипту (програми) знову стала головна сторінка сайту NURE яка і затримувала роботу інших запитів.

4.7 Результати запуску тесту UC06_SCIENCE

Програма UC06_SCIENCE імітує дії користувача при послідовному переході користувача з головної сторінки на вкладку SCIENCE і далі на вкладку SCIENCE_Topic для перегляду розкладу.

За допомогою Analysis переглянемо чи вдалося досягти планованих показників, для цього відкриємо графік «Transaction per Second» (рисунок 4.25):

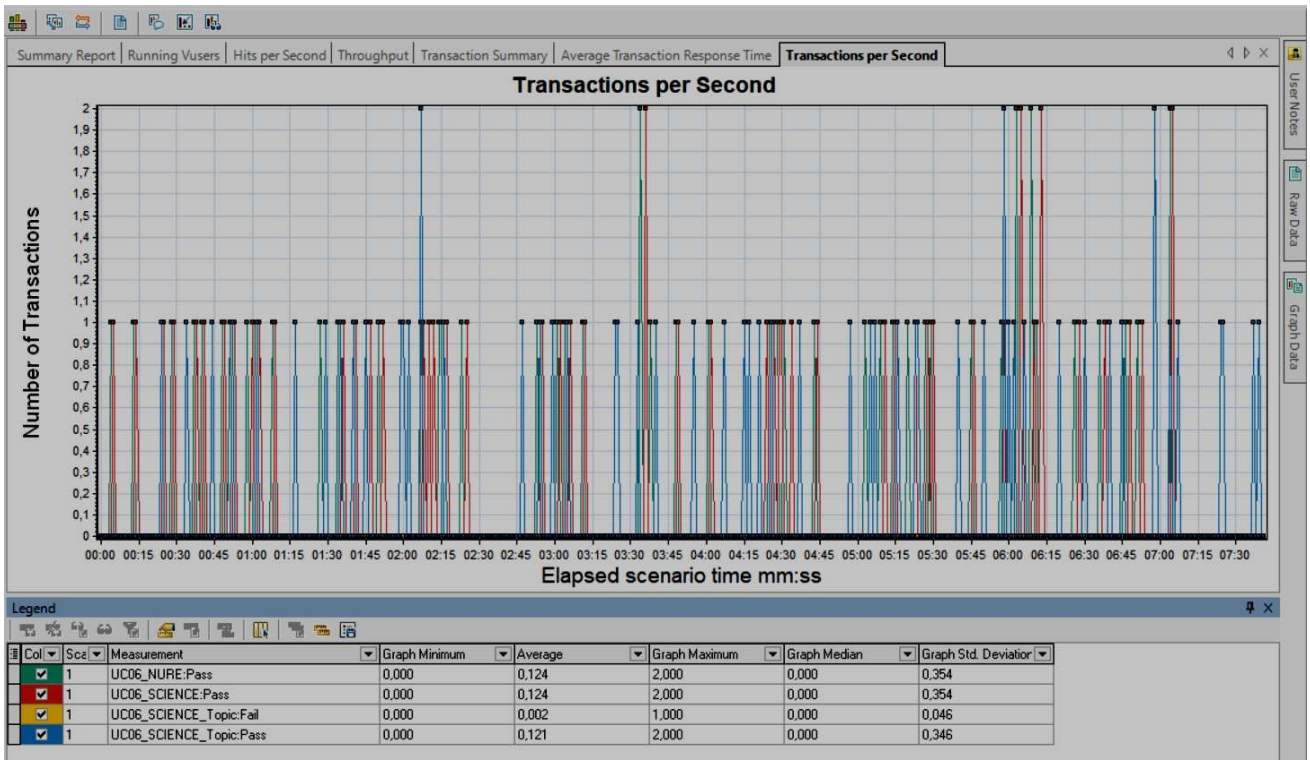


Рисунок 4.25 – Транзакції за секунду (Transaction per Second), UC06_SCIENCE

За горизонталлю відображений час тестування, за вертикаллю кількість вдало відпрацювавших транзакцій.

Як видно з графіку та колонки «Average», в середньому вдалося подати

0,124 tps з 18 планованих. Також з’явився запис «UC06_SCIENCE_Topic:Fail» це свідчить про те, що деякі запити буди втрачені під час обробки.

Розглянемо графік втрачених запитів більш ретельно рисунок 4.26.

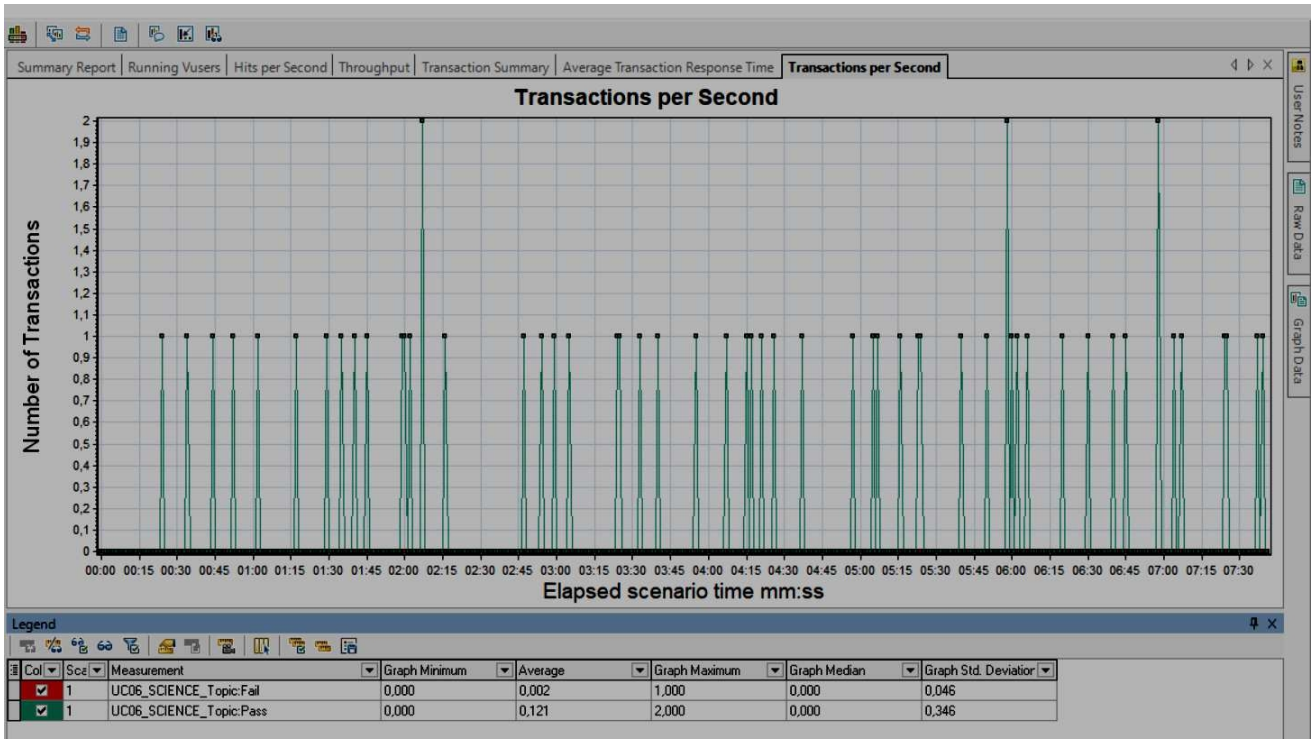


Рисунок 4.26 – Транзакції за секунду (Transaction per Second), SCIENCE_Topic (втрачені запити)

Як бачимо всього було втрачено декілька запитів, це свідчить про погану оптимізацію сторінки SCIENCE_Topic для праці під навантаженням. Проблему потрібно локалізувати та виправляти.

Тепер розберемося в причинах того, чому не вдалося подати плановане навантаження, для цього відкриємо графік «Average Transaction Response Time» (рисунок 4.27).

За горизонталлю відображений час тестування, за вертикаллю час відгуку транзакцій.

Як видно з графіку та колонки «Average», в середньому відгук за відкриттям головної вкладки сайту NURE склав 6,035 с, з максимальним значенням 10,149 с.

Відгук за відкриттям вкладки «SCIENCE» склав 1,601 с, з максимальним значенням 3,307 с.

Відгук за відкриттям вкладки «SCIENCE_Topic» склав 34,454 с, з максимальним значенням 68,518 с.

Такий великий час відгуку і став причиною того, що потрібне навантаження подати не вдалося.

Причиною невдалого запуску цього скрипту (програми) стала вкладка SCIENCE_Topic яка і затримувала роботу інших запитів.

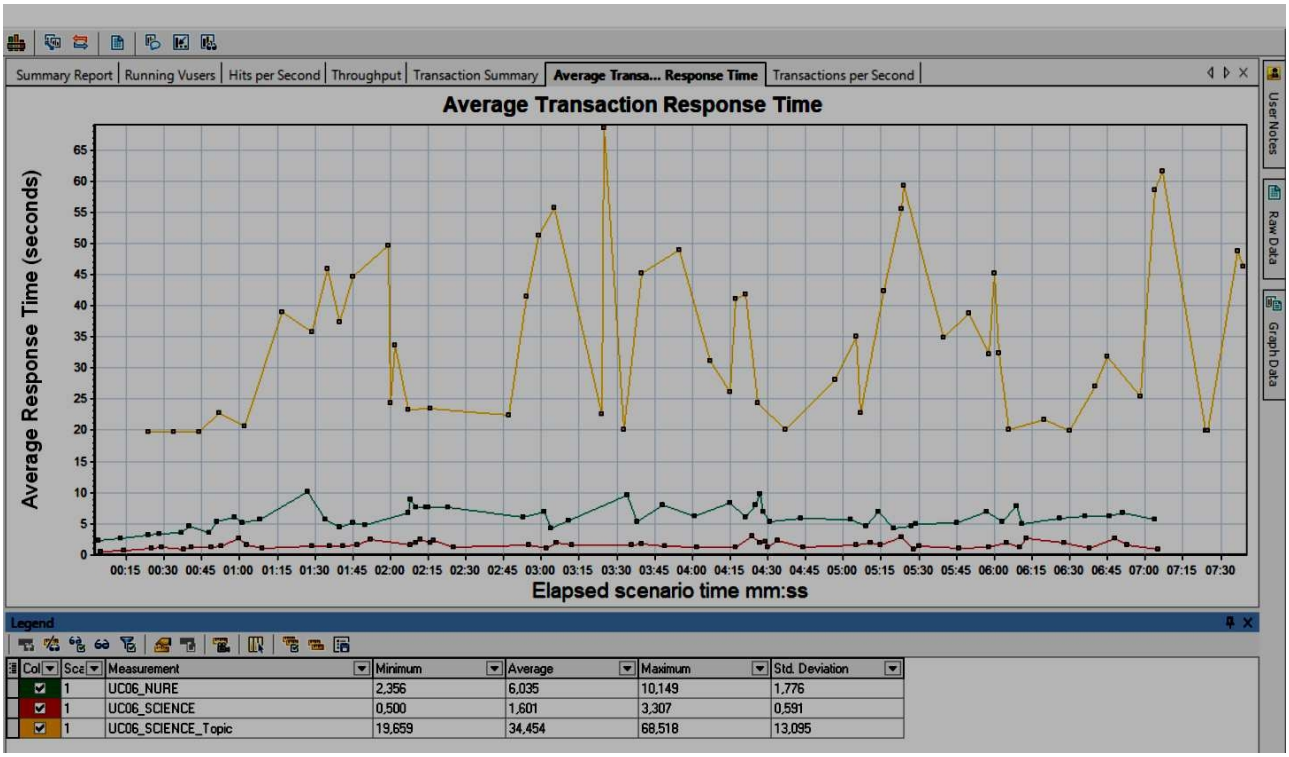


Рисунок 4.27 – Відгук транзакцій (Average Transaction Response Time), UC06_SCIENCE

4.8 Результати запуску тесту UC07_EDUCATION

Програма UC07_EDUCATION імітує дії користувача при послідовному переході користувача з головної сторінки на вкладку EDUCATION і далі на вкладку SCIENTIFIC_LIBRARY для перегляду розкладу.

Плановане навантаження 18 tps (див. рисунок 3.6). За допомогою Analysis переглянемо чи вдалося досягти планованих показників, для цього відкриємо графік «Transaction per Second» (рисунок 4.28):

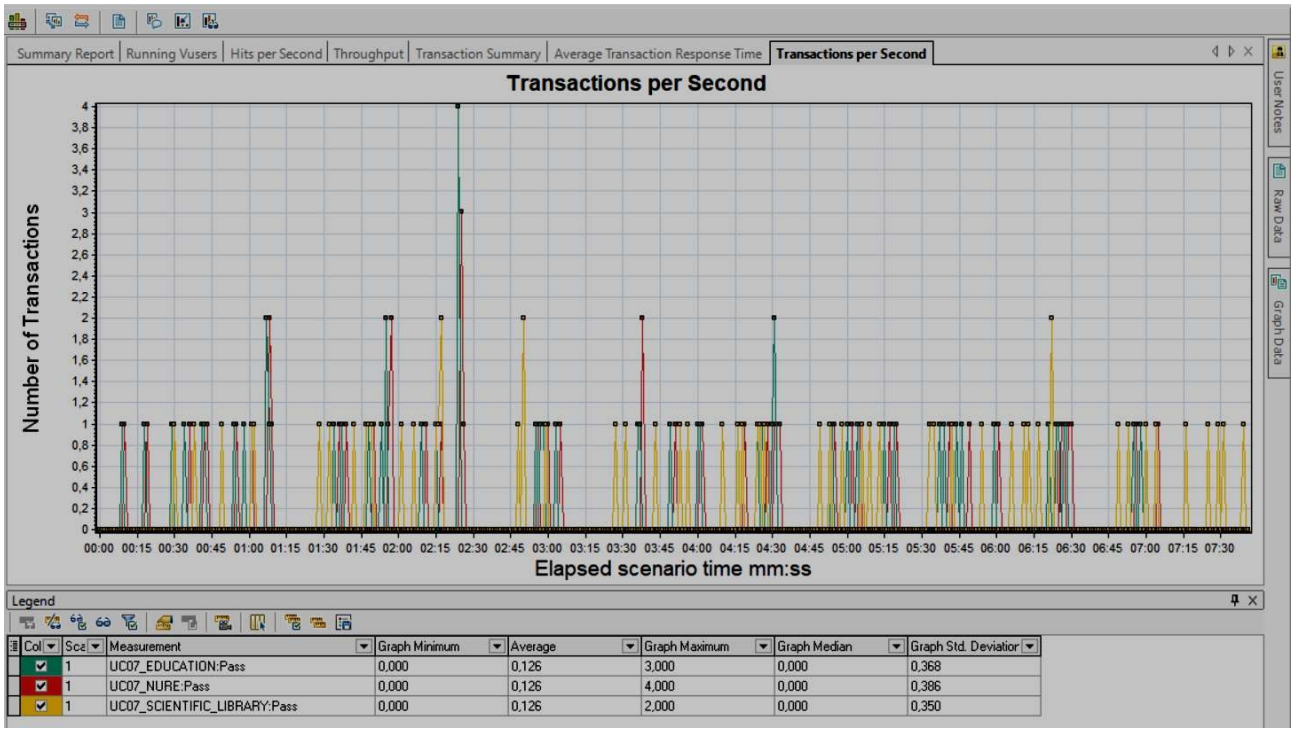


Рисунок 4.28 – Транзакції за секунду (Transaction per Second), UC07_EDUCATION

За горизонталлю відображений час тестування, за вертикаллю кількість вдало відпрацювавших транзакцій.

Скрипт UC07_EDUCATION складається з трьох транзакцій:

– UC07_EDUCATION, EDUCATION;

– UC07_EDUCATION, NURE;

– UC07_EDUCATION, SCIENTIFIC_LIBRARY.

Як видно з графіку та колонки «Average», в середньому вдалося подати 0,126 tps з 18 планованих за кожною з зазначених транзакцій.

Потрібно розібратися в причинах такої поведінки, для цього відкриємо графік «Average Transaction Response Time» (рисунок 4.29).

За горизонталлю відображений час тестування, за вертикаллю час відгуку транзакцій.

Як видно з графіку та колонки «Average», в середньому відгук за відкриттям головної вкладки сайту NURE склав 6,089 с, з максимальним значенням 11,326 с.

В середньому відгук за відкриттям вкладки «EDUCATION» склав 1,689 с,

з максимальним значенням 3,036 с.

Відгук за відкриттям вкладки «SCIENTIFIC_LIBRARY» склав 34,753 с, з максимальним значенням 118,496 с.

Такий великий час відгуку і став причиною того, що потрібне навантаження подати не вдалося.

Причиною невдалого запуску цього скрипту (програми) стала сторінка SCIENTIFIC_LIBRARY яка і затримувала роботу інших запитів.

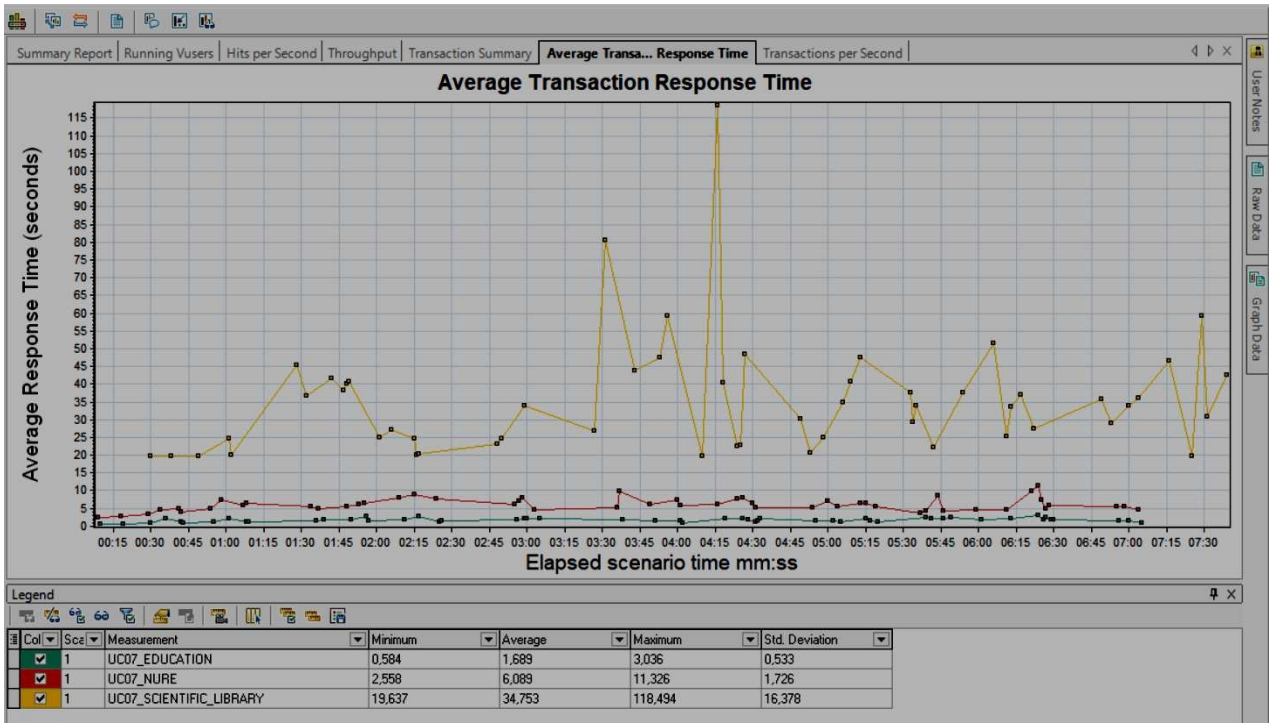


Рисунок 4.29 – Відгук транзакцій (Average Transaction Response Time), UC07_EDUCATION

4.9 Результати запуску тесту UC08_PRESS_CENTER

Програма UC08_PRESS_CENTER імітує дії користувача при послідовному переході користувача з головної сторінки на вкладку PRESS_CENTER і далі на вкладку PRESS_SERVICE для перегляду розкладу.

Плановане навантаження 18 tps (див. рисунок 3.6). За допомогою Analysis переглянемо чи вдалося досягти планованих показників, для цього відкриємо графік «Transaction per Second» (рисунок 4.30).

Рисунок 4.30 – Транзакції за секунду (Transaction per Second), UC08_PRESS_CENTER

За горизонталлю відображений час тестування, за вертикаллю кількість вдало відпрацювавших транзакцій.

Скрипт UC08_PRESS_CENTER складається з трьох транзакцій:

– UC08_PRESS_CENTER, NURE;

– UC08_PRESS_CENTER, PRESS_CENTER;

– UC08_PRESS_CENTER, PRESS_SERVICE.

Як видно з графіку та колонки «Average», в середньому вдалося подати 0,126 tps з 18 планованих за кожною із зазначених транзакцій..

Потрібно розібратися в причинах такої поведінки, для цього відкриємо графік «Average Transaction Response Time» (рисунок 4.31).

За горизонталлю відображений час тестування, за вертикаллю час відгуку транзакцій.

Як видно з графіку та колонки «Average», в середньому відгук за відкриттям головної вкладки сайту NURE склав 5,878 с, з максимальним значенням 10,522 с.

В середньому відгук за відкриттям вкладки «PRESS_CENTER» склав

1,699 с, з максимальним значенням 5,418 с.

Відгук за відкриттям вкладки «PRESS_SERVICE» склав 34,892 с, з максимальним значенням 80,407 с.

Такий великий час відгуку і став причиною того, що потрібне навантаження подати не вдалося.

Причиною невдалого запуску цього скрипту (програми) стала сторінка PRESS_SERVICE яка і затримувала роботу інших запитів.

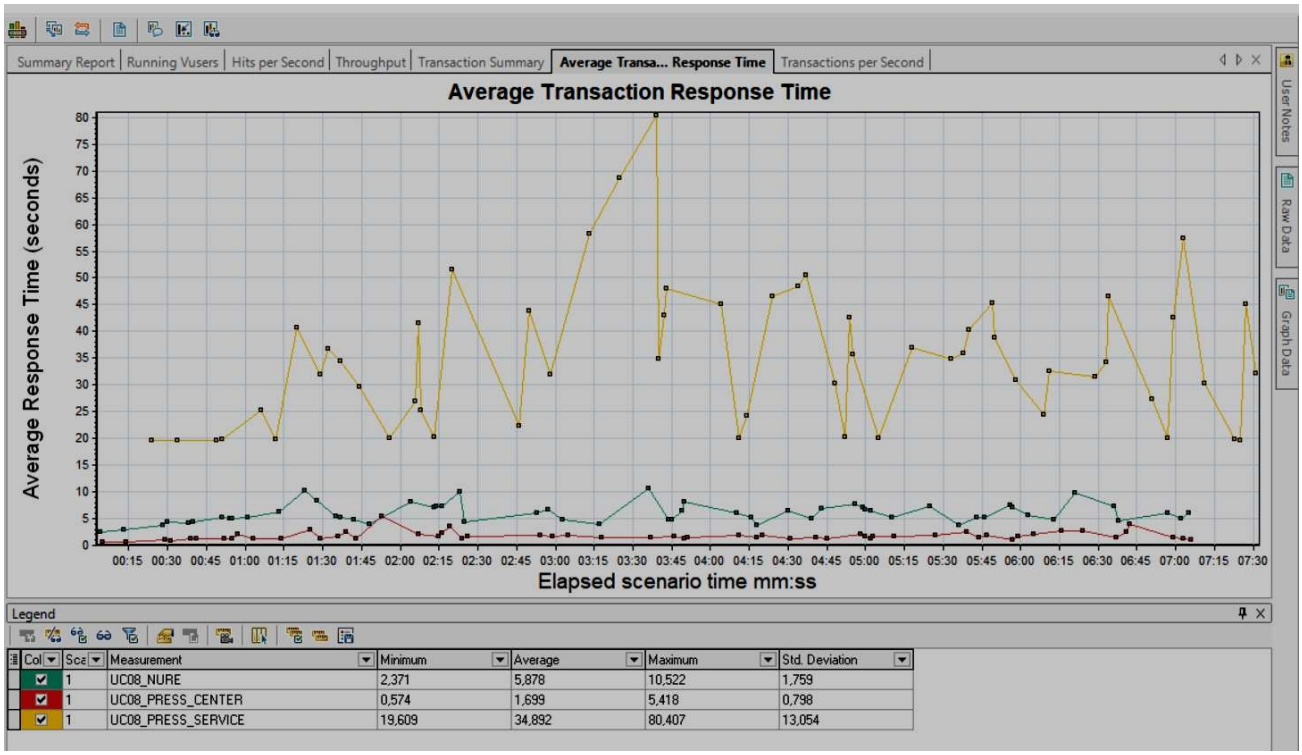


Рисунок 4.31 – Відгук транзакцій (Average Transaction Response Time), UC08_PRESS_CENTER

4.10 Результати запуску тесту UC09_CONTACTS

Програма UC09_CONTACTS імітує дії користувача при послідовному переході користувача з головної сторінки на вкладку CONTACTS і далі на вкладку ABOUT_UNIVERSITY для перегляду розкладу.

Плановане навантаження 15 tps (див. рисунок 3.6). За допомогою Analysis переглянемо чи вдалося досягти планованих показників, для цього відкриємо графік «Transaction per Second» (рисунок 4.32):

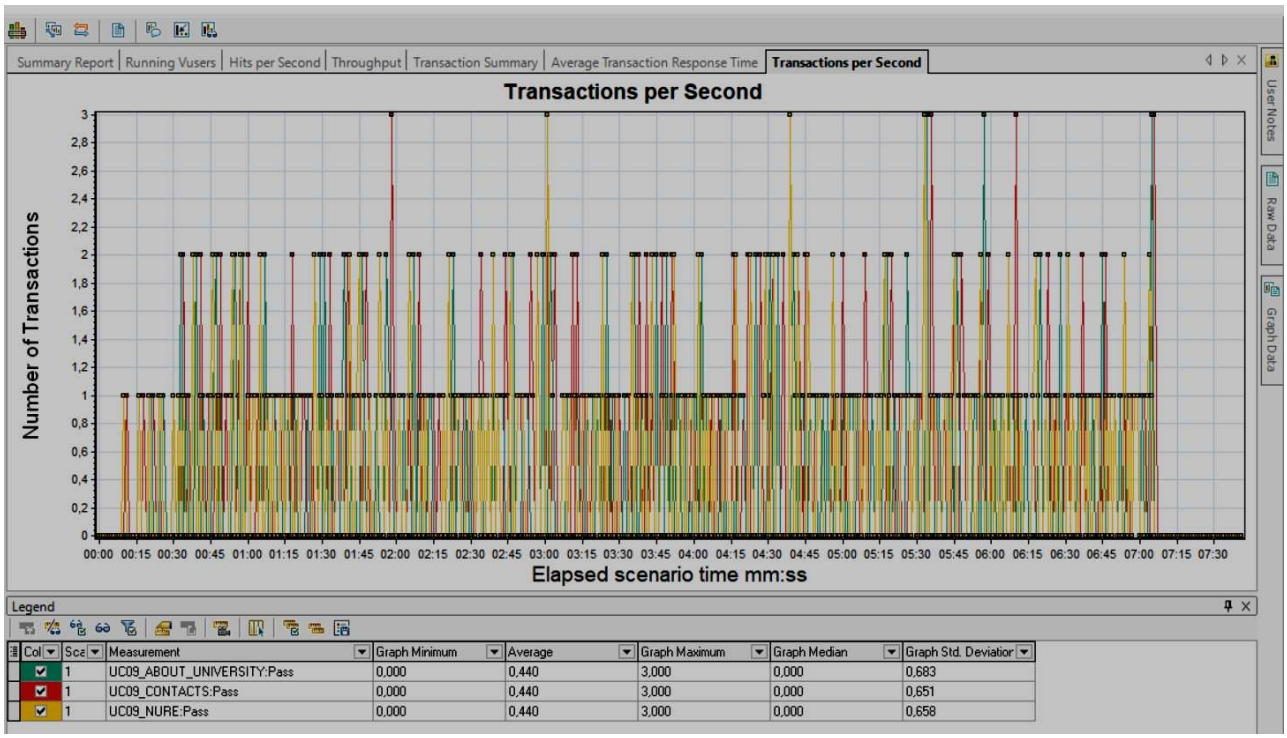


Рисунок 4.32 – Транзакції за секунду (Transaction per Second), UC09_CONTACTS

За горизонталлю відображений час тестування, за вертикаллю кількість вдало відпрацювавших транзакцій.

Скрипт UC09_CONTACTS складається з трьох транзакцій:

– UC09_CONTACTS, ABOUT_UNIVERSITY;

– UC09_CONTACTS, CONTACTS;

– UC09_CONTACTS, NURE.

Як видно з графіку та колонки «Average», в середньому вдалося подати 0,440 tps з 15 планованих за кожною із зазначених транзакцій.

Потрібно розібратися в причинах такої поведінки, для цього відкриємо графік «Average Transaction Response Time» (рисунок 4.33).

За горизонталлю відображений час тестування, за вертикаллю час відгуку транзакцій.

Для того, щоб зробити детальний аналіз потрібно переглянути графік за кожною з транзакцій окремо (рисунки 4.34...4.36.

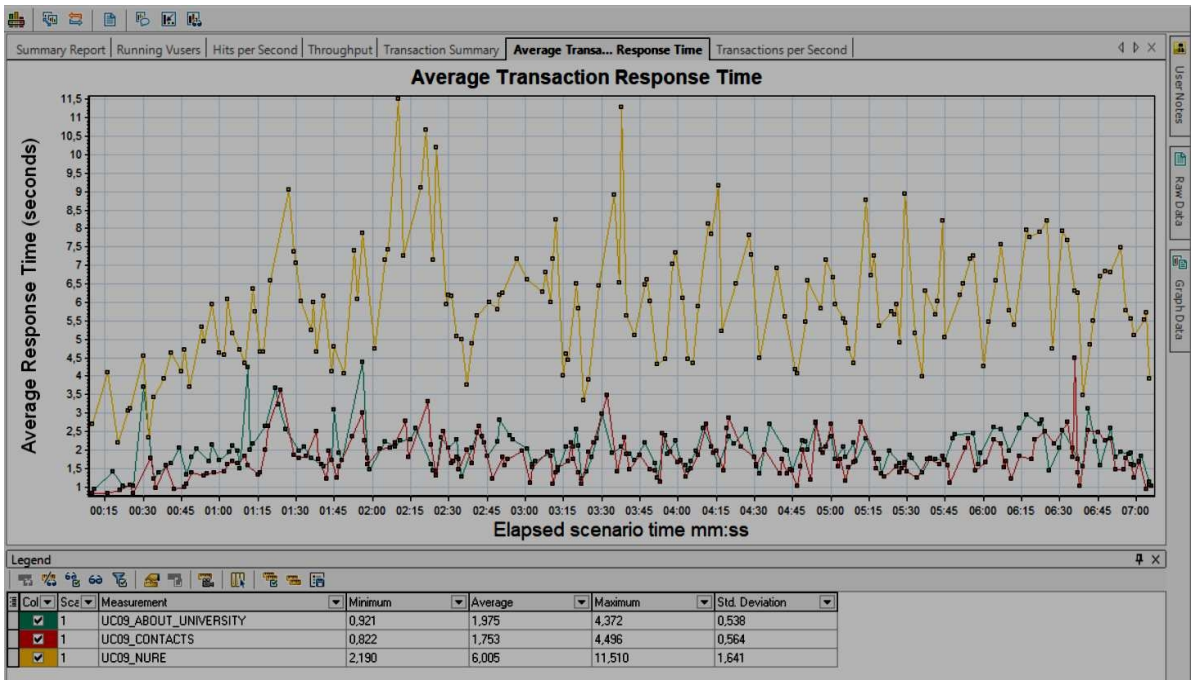


Рисунок 4.33 – Відгук транзакцій (Average Transaction Response Time), UC09_CONTACTS

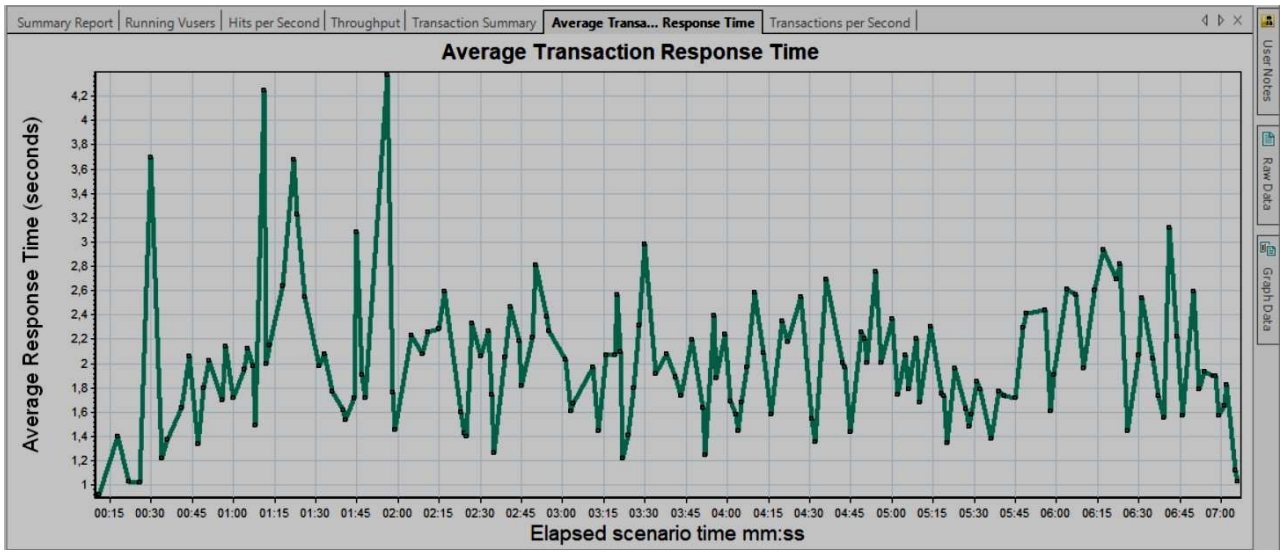


Рисунок 4.34 – Відгук транзакцій (Average Transaction Response Time), UC09_CONTACTS, ABOUT_UNIVERSITY

Як видно з графіку та колонки «Average», в середньому відгук за відкриттям вкладки «ABOUT_UNIVERSITY» склав 1,975 с, з максимальним значенням 4,372 с.

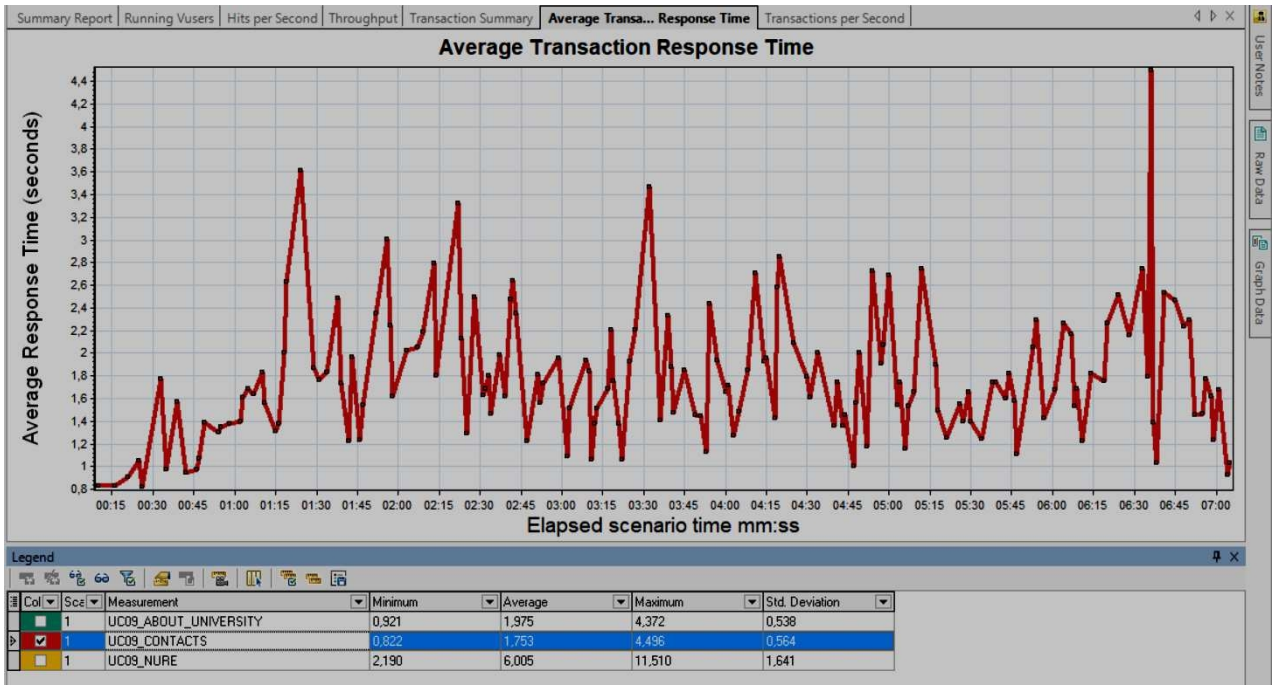


Рисунок 4.35 – Відгук транзакцій (Average Transaction Response Time), UC09_CONTACTS, CONTACTS

Як видно з графіку та колонки «Average», в середньому відгук за відкриттям вкладки «CONTACTS» склав 1,753 с, з максимальним значенням 4,496 с.

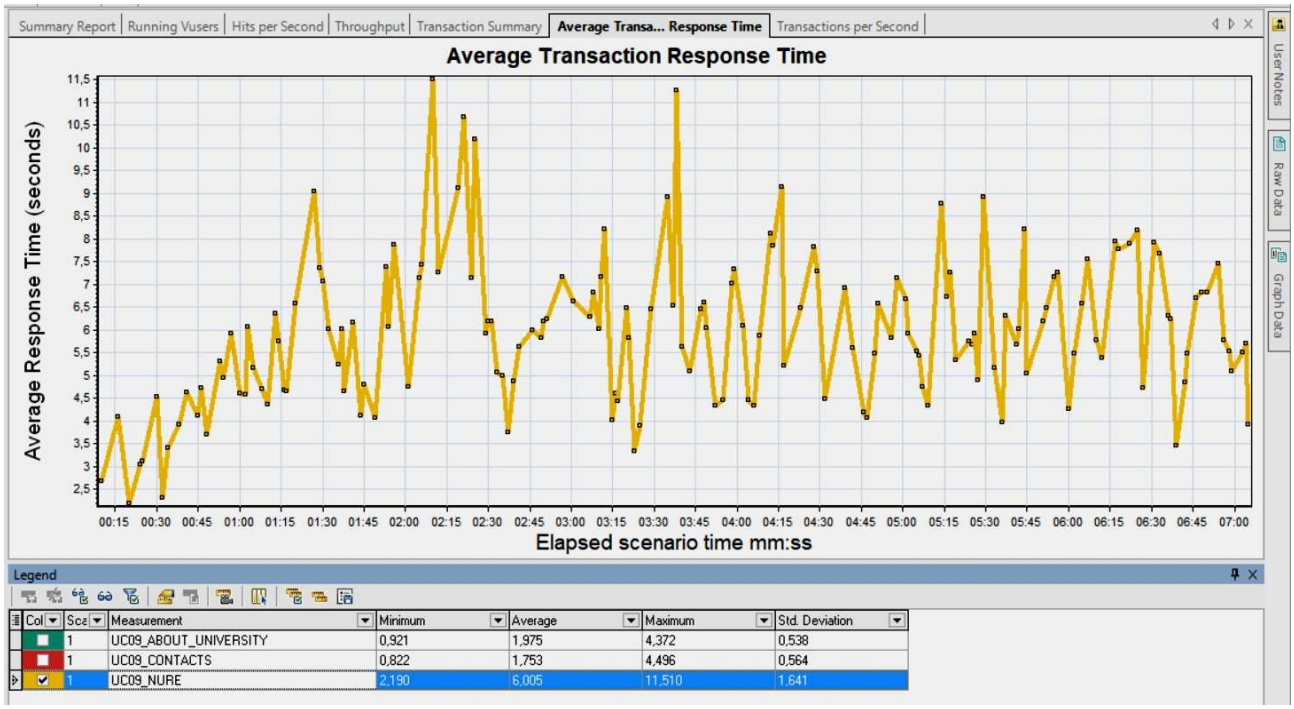


Рисунок 4.25 – Відгук транзакцій (Average Transaction Response Time), UC09_CONTACTS, NURE

Як видно з графіку та колонки «Average», в середньому відгук за відкриттям головної вкладки сайту NURE склав 6,005 с, з максимальним значенням 11,510 с.

Такий великий час відгуку і став причиною того, що потрібне навантаження подати не вдалося.

Причиною невдалого запуску цього скрипту (програми) стала головна сторінка сайту NURE, яка і затримувала роботу інших запитів.

Далі для подальшого аналізу звернемося до веб-сайту www.ashmanov.com на якому SEO спеціалісти стверджують, що відгук (швидкість відкриття) сторінок сайту не повинен перевищувати 2...3 секунди. Спираючись на ці данні зробимо висновок за результатами тестування (таблиця 4.1).

Таблиця 4.1 – Аналіз відгуку сторінок сайту

Назва транзакції	Середне значення відгуку	Результат
UC01_ChangeLanguage_RU	6,002	Незадовільно
UC01_ChangeLanguage_EN	2,821	Умовно задовільно
UC01_ChangeLanguage_UA	3,864	Незадовільно
UC02_Search	8,098	Незадовільно
UC03_NURE	6,059	Незадовільно
UC03_UNIVERSITY	1,671	Задовільно
UC03_ABOUT_UNIVERSITY	1,976	Задовільно
UC04_NURE	6,062	Незадовільно
UC04_APPLICANTS	1,658	Задовільно
UC04_EDUCATION_IN_ENGLISH	0,698	Задовільно
UC05_NURE	6,535	Незадовільно
UC05_STUDENTS	1,799	Задовільно
UC05_Timetable_of_Classes	3,946	Незадовільно
UC06_NURE	6,035	Незадовільно
UC06_SCIENCE	1,601	Задовільно
UC06_SCIENCE_Topic	34,454	Незадовільно
UC07_NURE	6,089	Незадовільно
UC07_EDUCATION	3,036	Умовно задовільно
UC07_SCIENTIFIC_LIBRARY	34,753	Незадовільно
UC08_NURE	5,878	Незадовільно
UC08_PRESS_CENTER	1,699	Задовільно
UC08_PRESS_SERVICE	34,892	Незадовільно
UC09_NURE	6,005	Незадовільно
UC09_CONTACTS	1,753	Задовільно
UC09_ABOUT_UNIVERSITY	1,975	Задовільно

Отриманий результат складно вважати прийнятним, тому за результатами навантажувального тестування більшість протестованих частин сайту NURE потребують вдосконалення, з метою покращення швидкодії.

ВИСНОВКИ

Метою даної атестаційної роботи було тестування навантажувальної здатності офіційного сайту Харківського національного університету радіоелектроніки.

Для досягнення поставленої мети в роботі були вирішені такі задачі:

- проаналізовано сучасні засоби навантажувального тестування;
- проведено огляд програмних продуктів для тестування сайтів;
- розроблено програми навантажувального тестування:

a) програму імітуючу дії користувача при зміні мови на сайті;

b) програма імітуючу дії користувача який використовує пошук по сайту NURE;

c) програма імітуючу дії користувача при послідовному переході користувача з головної сторінки на вкладку UNIVERSITY і далі на вкладку ABOUT_UNIVERSITY;

d) програма імітуючу дії користувача при послідовному переході користувача з головної сторінки на вкладку APPLICANTS і далі на вкладку EDUCATION_IN_ENGLISH;

e) програма імітуючу дії користувача при послідовному переході користувача з головної сторінки на вкладку STUDENTS і далі на вкладку Timetable_of_Classes для перегляду розкладу;

f) програма імітуючу дії користувача при послідовному переході користувача з головної сторінки на вкладку SCIENCE і далі на вкладку SCIENCE_Topic;

g) програма імітуючу дії користувача при послідовному переході користувача з головної сторінки на вкладку EDUCATION і далі на вкладку SCIENTIFIC_LIBRARY;

h) програма імітуючу дії користувача при послідовному переході користувача з головної сторінки на вкладку PRESS_CENTER і далі на вкладку PRESS_SERVICE;

i) програма імітуючу дії користувача при послідовному переході користувача з головної сторінки на вкладку CONTACTS і далі на вкладку

ABOUT_UNIVERSITY;

- проаналізовано отримані результати тестування;
- зроблено висновки за результатами тестування.

Отже всі поставлені в роботі задачі були виконані в повному обсязі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Software testing help. [Електронний ресурс] – Режим доступу: https:// www.softwaretestinghelp.com/types-of-software-errors/

2. Software Testing Fundamentals. [Електронний ресурс] – Режим доступу: https://softwaretestingfundamentals.com/defect/

3. Нагрузочное тестирование vs Тестирование производительности. [Електронний ресурс] – Режим доступу: https://www.performance-lab.ru/blog/load-testing/testirovanie-proizvoditelnosti

4. Терминология в нагрузочном тестировании. [Електронний ресурс] – Режим доступу: http://www.protesting.ru/automation/load/terminology.html

5. Инструменты для нагрузочного тестирования. [Електронний ресурс] – Режим доступу: https://training.qatestlab.com/blog/technical-articles/load-testing-tools/

6. Нагрузочное тестирование - автоматизация. [Електронний ресурс] – Режим доступу: https://intellect.icu/nagruzochnoe-testirovanie-avtomatizatsiya-6121

7. Обзор инструментария для нагрузочного и перформанс-тестирования. [Електронний ресурс] – Режим доступу: https://habr.com/ru/company/jugru/blog/337928/

8. Под предельной нагрузкой: обзор программ нагрузочного тестирования веб-серверов. [Електронний ресурс] – Режим доступу: https://xakep.ru/2008/04/21/43327/

9. Подводные камни в нагрузочном тестировании. [Електронний ресурс] – Режим доступу: https://2016.heisenbug-moscow.ru/talks/spisok-pokupok-chto-nuzhno-ne-zabyt-pri-zapuske-jmeter-testov/

10. Нагрузочное тестирование с использованием HP LoadRunner. [Електронний ресурс] – Режим доступу: https://thepresentation.ru/uncategorized /nagruzochnoe-testirovanie-s-ispolzovaniem-hp-loadrunner

11. Детальный обзор программного обеспечения HP LoadRunner. [Електронний ресурс] – Режим доступу: https://docplayer.ru/48254381-Detalnyy-obzor-programmnogo-obespecheniya-hp-loadrunner.html