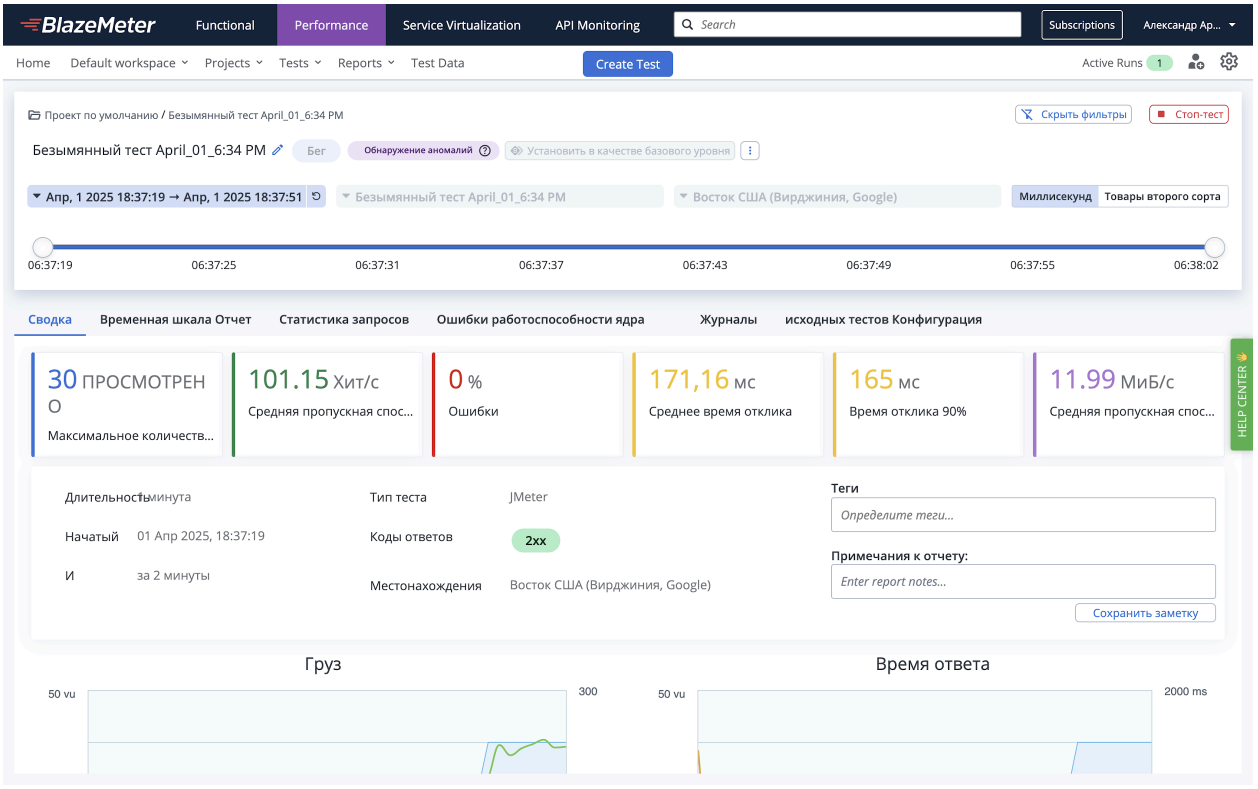
Задание 1

https://colab.research.google.com/drive/1UXXLn3l7qbyYUneleJYpUJ6ihKoPj88o?usp=sharing
(тут все вычисления по network o/i)

Задание 2



1.
Engine Health (здоровье движка) — это показатель состояния системы или сервиса во время тестирования. В отчёте он может отражать:

Стабильность работы (наличие/отсутствие сбоев).

Загрузку CPU, памяти, диска.

Количество ошибок и предупреждений.

Доступность сервиса.

Важная информация для тестировщика:

Насколько система устойчива под нагрузкой.

Есть ли утечки памяти или перегрузка CPU.

Нужна ли оптимизация кода/инфраструктуры.

2.
Latency time (время задержки) — это промежуток между отправкой запроса и получением первого ответа от сервера. Включает:

Время на обработку запроса сервером.

Сетевые задержки (например, из-за географической удалённости).

Время сериализации/десериализации данных.

Чем меньше latency, тем быстрее система реагирует.

3. Коды состояния HTTP
1XX (Информационные) – запрос принят, обработка продолжается (например, 100 Continue).

2XX (Успешные) – запрос успешно обработан (200 OK, 201 Created).

3XX (Перенаправления) – требуется дополнительное действие (301 Moved Permanently, 302 Found).

4XX (Ошибки клиента) – проблема на стороне пользователя (400 Bad Request, 404 Not Found).

5XX (Ошибки сервера) – проблема на стороне сервера (500 Internal Server Error, 503 Service Unavailable).

4.
Timestamp (метка времени) — это запись даты и времени в машиночитаемом формате (обычно в секундах или миллисекундах с 1 января 1970 года — Unix-epoch).

Используется для:

Логирования событий.

Синхронизации данных.

Анализа временных интервалов (например, в тестах производительности).

Пример: 1712345678 (Unix timestamp).

5.
Mock Service — это имитация реального сервиса или API, которая возвращает заранее подготовленные ответы. Используется в тестировании для:

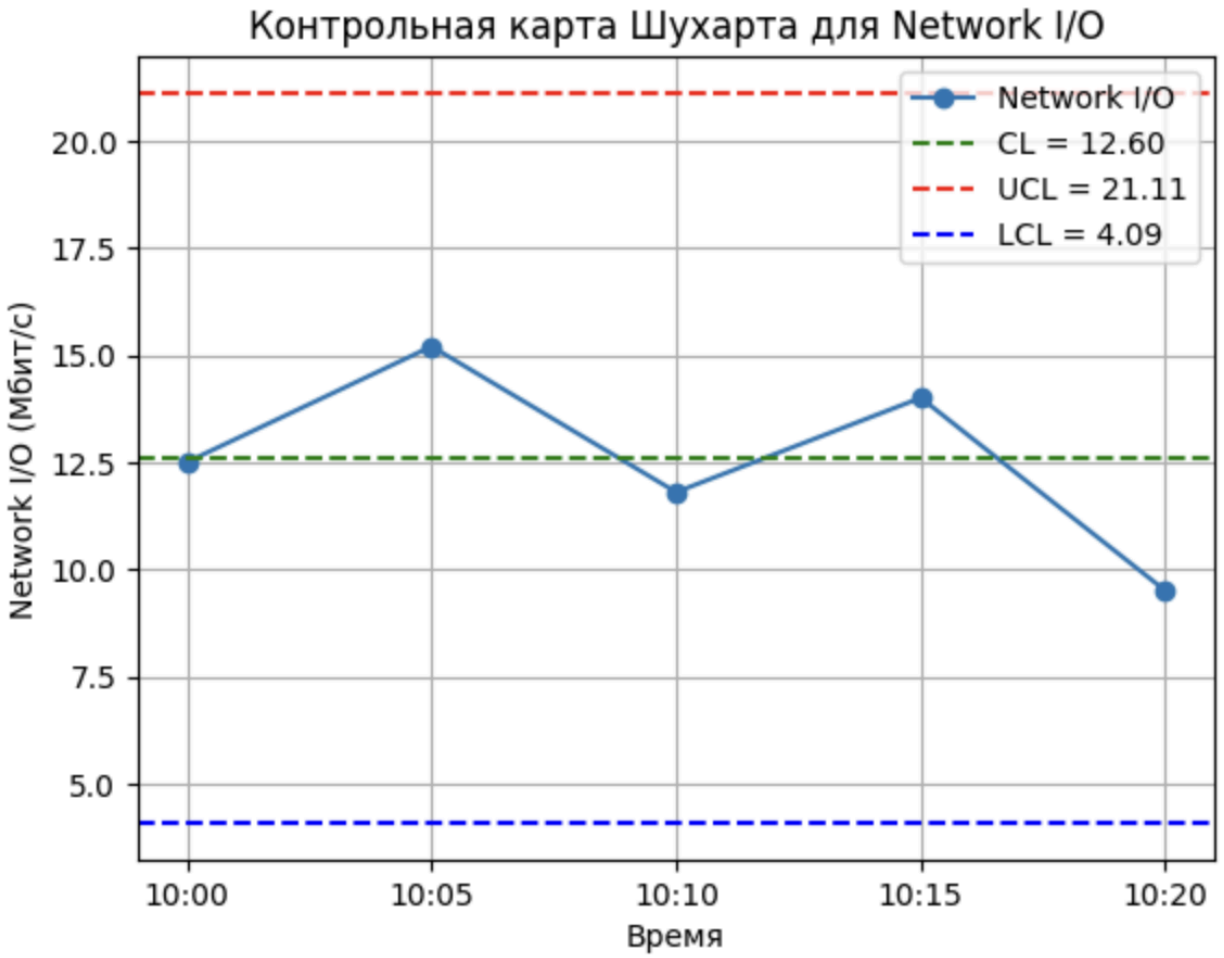
Замены зависимостей (например, платёжного шлюза).

Тестирования сценариев, которые сложно воспроизвести на реальном сервисе (ошибки, задержки).

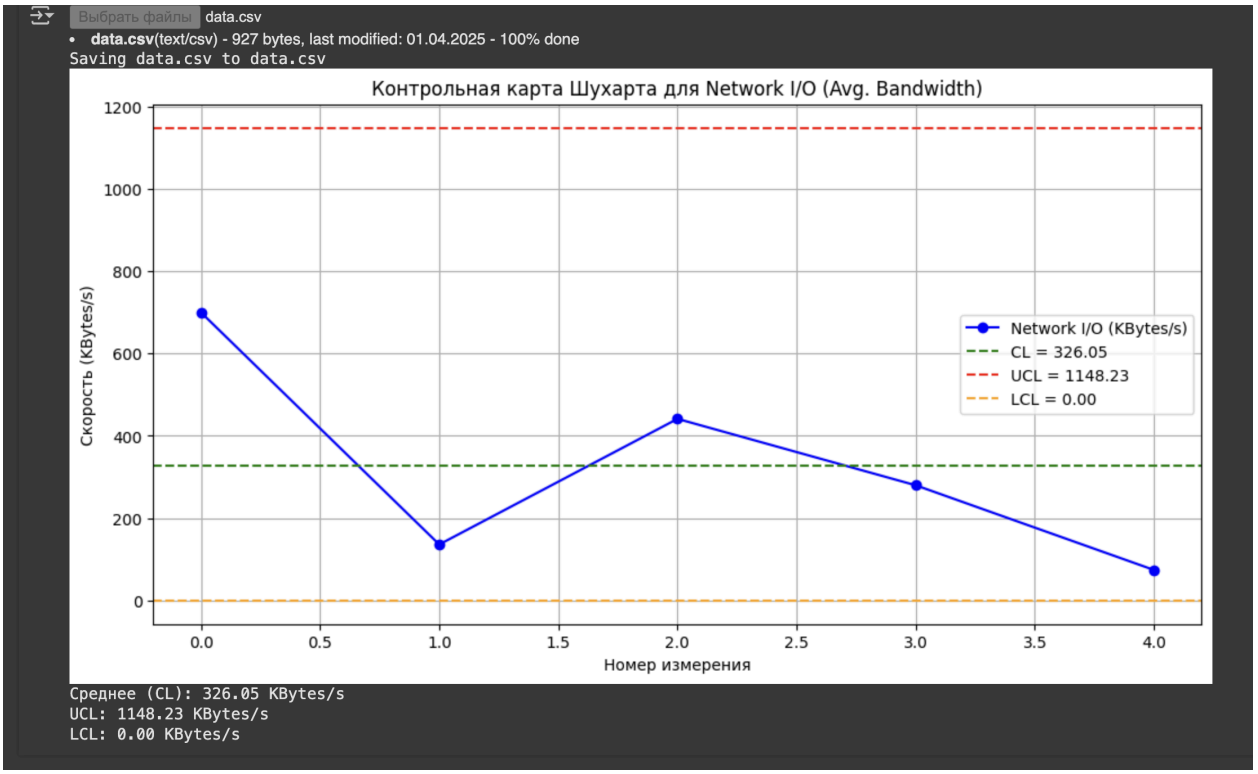
Ускорения тестов (не требует реальных вызовов).

Примеры инструментов: WireMock, MockServer, Postman Mock Server.

3) Ошибок не обнаружено
5)



Задание 1 (со скриптом)



4) ошибок нет
5)
1) Данные для нагрузочного тестирования
Что нужно:

Целевой URL/IP, тестовые сценарии (API, действия пользователей).

Параметры нагрузки: кол-во пользователей (VU), длительность теста, RPS.

Данные для авторизации (логины, токены).

Метрики: время ответа, пропускная способность, ошибки, нагрузка на CPU/RAM.

2) Критерии успешности
Главное:

Время ответа ≤ SLA (например, 2 сек).

≤ 1% ошибок.

CPU < 80%, нет утечек памяти.

Пропускная способность (RPS) соответствует требованиям.

3) Форматы файлов
JSON

Текстовый формат для конфигов и данных API.

Пример: {"url": "example.com", "method": "GET"}.

JMX

XML-файл для скриптов JMeter.

Содержит сценарии тестов (запросы, настройки потоков).

«Оценка показателей для проведения нагрузочного тестирования»

План нагрузочного тестирования для интернет-магазина (распродажа)
1. Вопросы к менеджеру проекта
Технические характеристики сервера:

CPU (ядра, частота), RAM, HDD/SSD, пропускная способность сети.

Максимальная нагрузка (пиковые значения CPU/RAM за последние 3 месяца).

Требования к отказоустойчивости:

Допустимый процент ошибок (например, ≤ 1%).

Время восстановления после сбоя (SLA, например, ≤ 5 минут).

Прогнозируемая нагрузка:

Ожидаемое количество пользователей в час/день.

Среднее время сессии (например, 5 минут).

География пользователей (локации для тестирования).

Сценарии поведения пользователей:

Просмотр товаров, добавление в корзину, оплата.

Частота запросов к API (например, поиск товаров, обновление корзины).

2.
Виды тестирования
Нагрузочное тестирование – проверка работы при ожидаемой нагрузке (например, 10 000 пользователей).

Стресс-тестирование – выход за пределы нормальной нагрузки (например, 15 000 пользователей).

Объемное тестирование – заполнение БД большим количеством заказов (например, 50 000 транзакций).

3.

Тестирование отказоустойчивости – имитация сбоев сервера/сети.

Профили нагрузки
На основе данных из примера (кинотеатры) адаптируем сценарии для интернет-магазина:

Роль	**Сценарий**	**RPS**	**Пользователи**	**Запросы/час**
Покупатель	Добавление товара в корзину + оплата	0.1	5 000	500
Просмотрщик	Поиск товаров, чтение отзывов	0.05	3 000	150
Отказник	Добавление в корзину без оплаты	0.03	2 000	60
Итого		**0.18**	**10 000**	**710**

Поправочный коэффициент (если тестовая среда слабее прода):

Например, коэффициент = 2 (если продакшен в 2 раза мощнее).

Итоговый RPS для теста: 710 / 2 = 355.
4. План тестирования
Подготовка:

Развернуть тестовую среду (аналогичную продакшену).

Настроить JMeter/BlazeMeter:

Создать Thread Groups по ролям.

Добавить HTTP-запросы (API + UI).

Настроить таймеры (например, Uniform Random Timer).

Заполнить БД тестовыми товарами/заказами.

Запуск тестов:

Этап 1: Нагрузка = 50% от ожидаемой (5 000 пользователей, 1 час).

Этап 2: Пиковая нагрузка (10 000 пользователей, 2 часа).

Этап 3: Стресс-тест (15 000 пользователей до отказа системы).

Мониторинг:

Метрики: Response Time (≤ 2 сек), Error Rate (≤ 1%), CPU/RAM (≤ 80%).

Инструменты: Grafana (визуализация), Prometheus (сбор данных).

Анализ результатов:

Выявить узкие места (например, медленные SQL-запросы).

Оптимизировать код/инфраструктуру (кеширование, балансировка нагрузки).

Пример конфигурации JMeter
Thread Group:

Number of Threads (Users): 10 000.

Ramp-Up Period: 300 сек.

Loop Count: Forever.

HTTP Request:

Method: GET/POST.

Path: /api/products, /api/cart.

Listeners:

Summary Report, Response Time Graph.