МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРІЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ
Кафедра Комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
Аліна САВЧЕНКО
"___" _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

(ДИПЛОМНИЙ ПРОЄКТ, ПОЯСНЮВАЛЬНА ЗАПИСКА)
ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ **"БАКАЛАВР"**
ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ
"ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ"

Тема: "Програмні засоби комплексного тестування програмного продукту на прикладі Playzone Minecraft"

Виконавець: студент групи УС-411Б Кравченко Микола Олексійович

Керівник: ст. викладач Єрмачков Юрій Олексійович

Нормоконтролер: Олександр ШЕВЧЕНКО

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних наук та технологій*

Кафедра *Комп'ютерних інформаційних технологій*

Галузь знань, спеціальність, освітньо-професійна програма: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології" .

ЗАТВЕРДЖУЮ
Завідувач випускової кафедри
_____ Аліна САВЧЕНКО
«____»_____2023р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи студента

Кравченка Миколи Олексійовича

1. **Тема роботи:** «Програмні засоби комплексного тестування програмного продукту на прикладі Playzone Minecraft» затверджена наказом ректора № 623/ст. від 01.05.2023р.
2. **Термін виконання роботи:** 15.05.2023р. - 25.06.2023р.
3. **Вихідні дані до роботи:** користувальницькі вимоги власника на тестування сайту проекту Playzone, програмні засоби JMeter і OWASP ZAP, пакет JSR223 Sampler.
4. **Зміст пояснювальної записки (перелік питань, що підлягають розробці):** аналіз проблем тестування програмного забезпечення та існуючих підходів, методик, практик та методологій, які використовуються при комплексному тестуванні; аналіз вимог та розробка тестів, реалізація тестування класами методик програмного продукту.
5. **Перелік обов'язкового ілюстративного матеріалу:** рисунки етапів життєвого циклу моделей тестування; таблиці порівняльного аналізу моделей і методик тестування; рисунок файлової структури проекту Playzone; рисунки

діаграм і таблиць виконання тестів, знаходжень уразливостей проекту, обчислення метрик та тест-кейсів.

6. Календарний план-графік

№ з/п	*Завдання*	*Термін виконання*	*Підпис керівника*
1.	Аналіз предметної області дослідження	15.05.2023–17.05.2023	
2	Збір, аналіз і опрацювання інформації за тематикою кваліфікаційної роботи	18.05.2023–21.05.2023	
3	Співбесіда з власником проекту Playzone	22.05.2023	
4	Аналіз користувальницьких вимог та розробка плану тестування	23.05.2023–25.05.2023	
5	Розробка і виконання тестів	26.05.2023 – 28.05.2023	
6	Написання пояснювальної записки кваліфікаційного проекту	29.05.2023–31.05.2023	
7	Підготовка демонстраційного матеріалу та доповіді	01.06.2023 – 18.06.2023	

7. Дата видачі завдання: 15 травня 2023 р.

Керівник дипломного проекту _____ Юрій ЄРМАЧКОВ
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Микола КРАВЧЕНКО
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Програмні засоби комплексного тестування програмного продукту на прикладі Playzone Minecraft» викладена на 115 сторінках, містить 52 рисунки, 14 таблиць та 19 літературно-наукових джерел.

Об’єкт дослідження: некомерційний проект Playzone Minecraft.

Предмет дослідження: тестування канонічності, функціональності, якості і безпеки веб-сайту проекту Playzone для підвищення досконалості програмного продукту.

Мета кваліфікаційної роботи: огляд існуючих програмних засобів, що використовуються для комплексного тестування, а також розробка власних тестів на основі цих засобів та проведення їх аналізу.

Метод дослідження: експериментальне спостереження метааналізу поведінки інформаційної системи некомерційного надання послуг.

Результат проекту: звіти розроблених і проведених тестів веб-сайту проекту.

ТЕСТУВАННЯ ПЗ, КЛАСИ ТЕСТІВ, ФУНКЦІОНАЛЬНЕ ТЕСТУВАННЯ, НАВАНТАЖУВАЛЬНЕ ТЕСТУВАННЯ, ТЕСТУВАННЯ БЕЗПЕКИ, ЖИТТЄВИЙ ЦИКЛ ПЗ, ВИМОГИ КОРИСТУВАЧА, МЕТОДОЛОГІЯ ТЕСТУВАННЯ, КОМПЛЕКСНЕ ТЕСТУВАННЯ, ЗАСОБИ ТЕСТУВАННЯ, ТЕСТ-КЕЙСИ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БД	База даних
ЕОМ	Електро обчислювальна машина
ІС	Інформаційна система
МН	Машинне навчання
ПЗ	Програмне забезпечення
ПП	Програмний продукт
ПС	Програмне середовище
ШІ	Штучний інтелект
JS	JavaScript
UI	User interface

ВСТУП

Кваліфікаційна робота розроблена для представлення програмних засобів, що можуть використовуватись для комплексного тестування програмного продукту, а також оцінці ефективності їх використання.

Мета роботи – розглянути існуючі програмні засоби, що використовуються для комплексного тестування, а також розробити власні тестів на основі цих засобів та проведення їх аналізу.

Ідея роботи – огляд принципів та методів комплексного тестування, а також практичного ознайомлення з інструментами, які використовуються при такому тестуванні.

Об’єкт дослідження - не тестований некомерційний проект Playzone.

Предмет дослідження - тестування канонічності, функціональності, якості і безпеки веб-сайту проекту Playzone для підвищення досконалості програмного продукту.

Методи дослідження - експериментальне спостереження метааналізу поведінки інформаційної системи некомерційного надання послуг.

Результат проекту - звіти розроблених і проведених тестів веб-сайту проекту.

Також, у роботі буде проведено аналіз існуючих програмних засобів, що використовуються для автоматизації тестування, та їх порівняння з альтернативними інструментами.

Кваліфікаційна робота може зацікавити спеціалістів галузі розробки, тестування і підтримки якості ПЗ.

Як зазначають автори книги "Фундаментальне тестування програмного забезпечення", Дороті Грехем, Еріка ван Флітта та Ізабелли Уертон : «Підхід, заснований лише на ручному або автоматизованому тестуванні, часто нездатний виявити проблеми в продукті. Використання комплексного тестування, що поєднує

в собі різні методи та підходи, дозволяє підвищити ефективність процесу тестування та покращити якість продукту.» [1].

У сучасному світі ПЗ використовується в різних галузях життя: від банківської справи до медицини та промисловості. Якість ПЗ має безпосередній вплив на функціонування та безпеку систем, що ґрунтується на цьому ПЗ.

З урахуванням вищезазначеного, кваліфікаційна робота на тему "Програмні засоби комплексного тестування програмного продукту на прикладі Playzone Minecraft " є актуальною та має практичну значимість як для загальної структуризації комплексного підходу до тестування, так і покращення якості існуючого бізнес-проекту.

РОЗДІЛ 1
ПРОБЛЕМИ ТА ОГЛЯД ІСНУЮЧИХ ПІДХОДІВ ДО РІШЕННЯ ТЕСТУВАННЯ

1.1. Існуючі проблеми підходів і методів

Існує ваговий обсяг проблем із існуючими методами та підходами до комплексного тестування програмного забезпечення, які можуть призводити до неповної перевірки програмного забезпечення та неполадок у роботі програми, або ж виконувати зовсім не актуальні перевірки.

До основних проблем існуючих підходів та методів відносять:

1. Недостатнє покриття тестування – велика кількість тестів не покривають усі можливі шляхи взаємодії користувача з ПЗ, що може призвести до непередбачених багів у роботі програми;
2. Обмеженість тестових даних - часто використовувані тестові дані можуть бути обмежені і не враховувати всі можливі комбінації вхідних даних, що також може призвести до помилок;
3. Недостатній баланс поєднання тестування між «необхідно» та «можливо» - Як зазначає автор статті «Принципи тестування» : «Наскільки ретельним тестування не було, не можна врахувати всі можливі сценарії і передбачати всі можливі помилки.»[2] - певні аспекти тестування можуть бути недостатньо протестовані, або навпаки - занадто багато часу та зусиль може бути витрачено на тестування, яке менш важливе. Це може стати на заваді якісному та ефективному тестуванню;
4. Недостатня автоматизація – мануальне тестування, зазвичай, може бути досить повільним та трудомістким процесом, що ускладнює проведення його у великих проектах;

Проте, не завжди можна автоматизувати всі тести, але через це може бути ситуація, коли відбувається неповна перевірка функціональності;

5. Недостатня інтеграція з тестовими даними - тестування може залежати від користувацьких даних з інших джерел або сценаріїв, що може ускладнити проведення тестів;

6. Недостатня перевірка безпеки - у деяких випадках може бути недостатньо лише тестування функціональності, через це існує необхідність проводити перевірку безпеки ПЗ;

7. Недостатня перевірка продуктивності - у деяких випадках існує необхідність проведення тестування продуктивності, для підтвердження, що ПП може обробляти великі обсяги даних з мінімальною затримкою і без сторонніх проблем;

8. «Пізнє» тестування – часто трапляються випадки, коли ПП починають тестувати на фінальних етапах життєвого циклу розробки, що призводить до додаткових бізнес-витрат на переробку готового ПП. «У будь-якій галузі розробки, не вдасться відточити продукт до досконалості, якщо його з самого початку неправильно сконструйовано». [3]

1.2. Огляд підходів до існуючих проблем тестування програмних продуктів

1.2.1. Огляд традиційних підходів до тестування

Прийнято вважати традиційними підходами до тестування – мануальне та автоматизоване тестування.

Мануальне та автоматизоване тестування - це два традиційні підходи до тестування ПЗ, які використовуються у всіх проектах розробки ПП.

Мануальне тестування полягає у виконанні тестових сценаріїв спеціалістів manual-тестування. Тестувальники взаємодіють з програмним забезпеченням, виконуючи алгоритми дій описаних створеною документацією розробників або розробників-тестувальників та перевіряючи результати.

Мануальне тестування може бути корисним для виявлення проблем, які не можуть бути виявлені автоматизованими інструментами.

Головним недоліком такого підходу є повільність, витратність, а також пряма залежність від навичків та вмінь спеціаліста і його уваги до деталей.

Автоматизоване тестування використовує спеціалізовані інструменти – спеціальні розроблені ПП, які виконують тестові сценарії автоматично. Це дозволяє збільшити швидкість та ефективність виконання тестів та виявлення багів.

Автоматизоване тестування може бути корисним для тестування функціональних та регресійних вимог, адже, зазвичай, такі методи тестування потребують велику кількість ітерацій.

Проте, автоматизоване тестування не може остаточно замінити мануальне тестування, оскільки воно не дає повне зображення взаємодії користувачів з ПП та показувати точну поведінку ПП у цифровому середовищі за реальних умов. Крім того, автоматизоване тестування може бути кориснішим для тестування великих проектів або проектів з частими змінами (як приклад світових корпорацій Microsoft, Goggle, та їх ПП OS Windows, ChromeOS), тоді як мануальне тестування більш ефективне для менших проектів з низьким рівнем складності (як приклад українські стартапи 3DLOOK з однойменним додатком для цифрового виміру одягу, або PLAYZONE – тематичний ігровий проект).

Автоматизоване тестування може бути менш витратним, оскільки програми для його виконання можуть бути перевикористані та легко змінені, але тут існує пропорційна залежність вартості від складності програмного забезпечення та тест-сценаріїв.

Варто заначити, що мануальне тестування та автоматизоване тестування не виключають одне одного. Для прикладу наведемо факт, що автоматизовані інструменти тестування можуть бути використані для швидкого виконання регресійних тестів, тоді як манулаьне тестування може бути використане для тестів нових функцій або для перевірки здатності програмного забезпечення працювати в різних умовах.

1.2.2. Огляд нових тенденцій та підходів до тестування

Останні роки характеризуються швидким розвитком технологій та зростанням вимог до ПЗ. До послуг тестування ПЗ зазвичай вдається великий бізнес, соціально-значущі проекти, де дефекти можуть нести великі репутаційні та економічні ризики. Це призводить до зміни тенденцій та підходів до тестування, що дозволяє компаніям ефективніше та якісніше виконувати процес тестування програм.

Тестування з штучним інтелектом

Новою тенденцією до тестування є використання штучного інтелекту та машинного навчання для автоматизованого тестування. ШІ та МН можуть допомогти виявити тестові складнощі та проблеми в ПЗ, що може значно знизити час, необхідний для проведення тестів та поліпшити якість тестів.

Основною перевагою використання ШІ у тестуванні - здатність швидкого та точного виявлення проблем та помилок в ПЗ: ШІ може аналізувати великі обсяги даних, виявляти нові тенденції, створювати нові шаблони та здійснювати передбачувані дії посилаючись на велику базу знань. Крім того, тестування ШІ може бути ефективнішим, оскільки нейромережі не вимагають фізичної присутності користувача.

Незважаючи на переваги, тестування, використовуючи таку тенденцію, також має свої недоліки: ШІ може бути обмеженим в своїх можливостях тестування та не здатний виявляти некоректну поведінку в ПЗ, яку може виявити тільки людина. Крім того, розробка та налаштування системи тестування на основі нейромереж, може бути більш економічно та часово витратним процесом.

Тестування без тестувальників

Ще однією тенденцією до тестування є підхід «Тестування без тестувальників».

Тестування без тестувальників – це підхід до тестування ПЗ, який передбачає використання автоматизованих інструментів та методів тестування для забезпечення якості ПП без прямої участі людських ресурсів.

Основною перевагою тестування без тестувальників – значна фінансова та часова економія, оскільки такий підхід дозволяє збільшити швидкість тестування та значно знизити фінансові витрати на людські ресурси. Крім того, такий підхід забезпечує більш точне та повне тестування, що дозволяє виявляти баги значно швидше та знижує ризик появи недоліків на фінальних етапах розробки.

Проте, тестування без тестувальників не забезпечує повного охоплення сценаріїв та ситуацій взаємодії ПП у цифровому середовищі за реальних умов, що може призвести до пропуску помилок коду. Також, такий підхід недостатньо гнучкий для складних сценаріїв тестування та вимагає значних зусиль їх налагодження.

Тестування без розробників

Тестування без розробників означає, що для тестування ПЗ запрошуються люди, не пов’язані з розробкою. Такий підхід може бути корисним для забезпечення незалежного огляду та виявлення деяких проблем, розглядаючи ПП з інших граней.

Використовуючи підхід тестування без розробників, частою практикою є зіткнення з проблемою обмеженої експертизи, при якій запрошені люди в якості тестувальників, можуть мати обмежені знання про технічні деталі розробки ПЗ, через що можуть не розглядатись деякі проблеми, також для виправлення проблем є потреба у розробниках: якщо тестувальники не можуть виправити знайдені проблеми, то розробники все одно повинні бути залучені до процесів виправлення помилок.

Тестування в реальному часі

Тестування в реальному часі - це також один із підходів нових тенденцій до тестування, який забезпечує перевірку роботи ПЗ в режимі реального часу. Цей підхід використовується для ПЗ, яке взаємодіє з фізичними пристроями або мережами.

Основною перевагою тестування в реальному часі є виявлення проблем ПЗ, які можуть виникати у час реальної взаємодії: затримки, які виникають при передачі даних по мережі або проблеми зі зчитуванням даних з фізичних пристроїв.

Однак, такий підхід може бути складним у реалізації, особливо для майбутнього ПП, який взаємодіє з фізичними пристроями і може вимагати значної кількості фізичних пристроїв та мережевих засобів.

Тестування у контейнері

Тестування у контейнерах - це підхід до тестування, який базується на використанні контейнерів для забезпечення ізольованого тестового середовища. Контейнери є знайомими для розробників і тестувальників, які використовують їх для розгортання програмного забезпечення та тестів на різних платформах та середовищах. Testcontainers – це бібліотека тестування, яка дозволяє писати тести з використанням реальних залежностей за допомогою одноразових контейнерів Docker [4].

До основних переваг тестування у контейнерах відносять ізольованість середовища, яке забезпечує ізоляцію середовища і дозволяє тестувальникам запускати тести без впливу зовнішніх чинників; ефективність таких тестів, адже контейнери дозволяють запускати багато тестів одночасно на одному пристрої, що зменшує час, необхідний для виконання тестів; гнучку масштабованість виконання тестів ПС: від конкретної функції, до загального виконання поставленої цілі ПП; переносимість таких тестів, інтегруючи тести у крос-платформену ІС.

З іншого боку, такий підхід прийнято вважати складним через потребу у специфічних знаннях та навичок а також досить потужну витратність обчислювальних ресурсів ЕОМ.

Хмарне тестування

Нова тенденція тестування за допомоги хмарних технологій - це процес тестування ПЗ, який виконується за допомогою хмарних технологій, де тестова ІС розташована в хмарному сервісі. В цьому випадку спеціалісти виконують тестування безпосередньо в хмарному середовищі, а не на власних серверах або локальних пристроях.

Найбільшою перевагою хмарного тестування є збільшення ефективності тестування і зниження витрат фінансових і часових ресурсів, необхідних для

запуску тестових наборів, оскільки можна використовувати розподілені системи з більшою кількістю ресурсів, ніж у звичайних умовах.

Однак, наявність обмеження стосовно використання інфраструктури хмарного сервісу, необхідність передачі конфіденційної інформації до зовнішніх сервісів, а також проблеми зі сумісністю зі сторонніми інструментами є менш гнучким і надійним, оскільки використання підходу залежить від стану хмарного сервісу, доступності Інтернету та безпечного з'єднання.

Тестування з використанням блокчейну

Тестування з використанням блокчейну - це підхід до тестування, що базується на застосуванні технології блокчейн для отримання безпеки і надійності ПЗ. Він дозволяє забезпечити надійність, цілісність тестових даних, а також можливість створення безпечних і надійних середовищ для тестування.

Завдяки такому підходу, можна виконувати тестування з безпекою і надійністю до даних, адже тестові дані зберігаються в безпечному середовищі, що забезпечує їх надійне зберігання і захист від несанкціонованого доступу, а технологія блокчейн забезпечує цілісність тестових даних, що гарантує, що тестові результати будуть точними і надійними. А також з таким підходом можлива реалізація автоматизації тестів і прозорого відслідковування кожного кроку процесу тестування.

Однак, при використанні підходу тестування з використанням блокчейну існують складності у самому використанні технології блокчейн, яка потребує додаткового розуміння технології і в даний час такий підхід має обмежені можливості у використанні, пов'язані із забезпеченням повністю відповідної продуктивності і масштабованості для великих проектів.

1.3. Використання методологій у тестуванні

Методології тестування - це організований підхід до процесу тестування ПЗ, який включає функціональне та нефункціональне тестування для різних перевірок.

Використання методологій у тестуванні допомагає забезпечити системний та організований підхід до процесу тестування ПЗ. Такий підхід допомагає зменшити ризик пропуску багів і збільшує швидкість їх виявлення та виправлення, що підвищує загальну якість ПП.

Кожна методологія має свої переваги та недоліки, тому при виборі методології тестування потрібно враховувати вимоги проекту та можливості команди.

Кожна методологія тестування має свої власні особливості до підходу тестування, проте загалом всі методології включають в себе такі елементи тестування як:

- Планування тестування;
- Виконання тестування;
- Аналіз результатів тестування;
- Комунікація та звітність;
- Автоматизація тестування.

Планування тестування включає в себе визначення цілей, вимог та потреб користувачів, визначення тестових кейсів та інших складових процесу тестування.

Виконання тестування включає в себе виконання тестових кейсів, аналіз результатів тестування та реєстрацію виявлених проблем.

Аналіз результатів тестування включає в себе дослідження виявлених багів та їх причин, а також загальну оцінку ризиків та планування подальших кроків.

Комунікація та звітність включає в себе забезпечення комунікування між всіма учасниками процесу розробки, тестування та випуску продукту та декларування звітності виконаної роботи.

Автоматизація тестування включає в себе використання спеціалізованих інструментів та технологій для автоматизації тестування та покращення їхньої ефективності та швидкості.

Основні методології тестування включають:

– Waterfall (класичний);
– Spiral;
– Extreme;

– Test Driven Development (TDD);
– Agile-методології;
– DevOps методології;
– Дослідницьке тестування.

1.3.1. Тестування з використанням методології Waterfall

Waterfall-методологія є лінійною послідовністю етапів розробки, в якій кожен наступний етап розпочинається після завершення попереднього. Тестування з використанням даної методології, зазвичай, виконується на фінальних етапах розробки - перед передачею продукту в експлуатацію.

Водоспадна модель тестування (див. рис. 1.1) як і модель розробки, передбачає послідовне проходження процесу, розбитого на стадії. Перехід до нового етапу можливий тільки після завершення попереднього. [5]

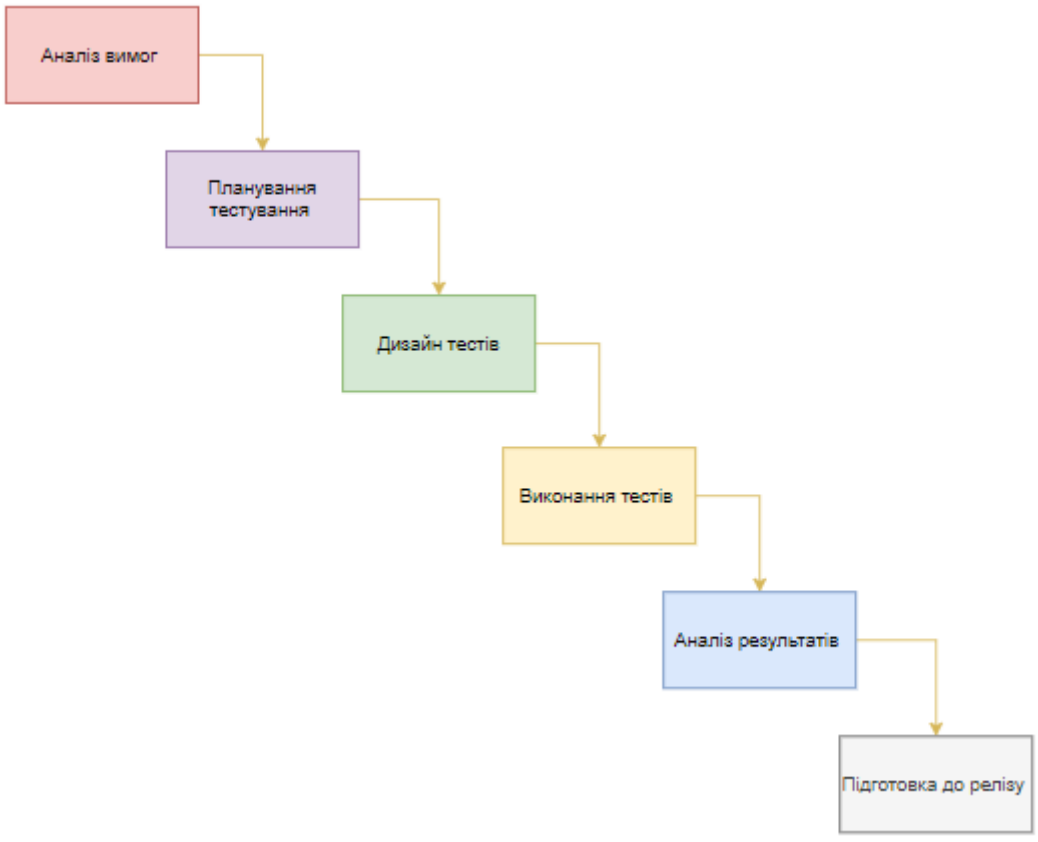


Рис. 1.1. Етапи тестування з використанням моделі Waterfall

Для класичної моделі тестування ПЗ виділяють такі етапи:

1. Аналіз вимог - першим етапом є аналіз вимог, де тестувальники ознайомлюються з вимогами до продукту та створюють план тестування;

2. Планування тестування - у другому етапі тестувальники створюють детальний план тестування, який включає у себе список тестових сценаріїв, джерела даних, тестові середовища, терміни та бюджет;

3. Дизайн тестів - на третьому етапі тестувальники створюють тестові сценарії та тестові набори для перевірки відповідності продукту вимогам;

4. Виконання тестів - після створення тестових сценаріїв та наборів, тестувальники виконують тестування та документують результати;

5. Аналіз результатів – під час цього етапу, тестувальники аналізують результати тестування та допомагають розробникам виправити баги;

6. Підготовка до релізу – після етапу аналізу результатів та виправлення помилок, тестувальники проводять останню перевірку тестування, щоб переконатися, що всі помилки виправлені та продукт готовий до передачі в експлуатацію.

Переваги моделі Watrefall:

– Висока прозорість розробки та фаз проекту;
– Точна послідовність;
– Стабільність вимог;
– Суворий контроль керуванням проекту;
– Високий рівень визначення контролю якості.

Недоліки моделі Watrefall:

– Watrefall моделі необхідна постійна актуалізація документації, що витрачає додаткові часові ресурси і загально ускладнює комунікацію;
– З точки зору методологій, гнучкість дуже низького рівня;
– Відсутній інструмент комунікації із замовником розробки ПП;
– Відсутнє розуміння Замовником деталей кожного етапу розробки;

– Усі вимоги мають бути відомі на початку життєвого циклу проекту.

1.3.2. Тестування з використанням моделі Spiral

Spiral-методологія тестування - це ітеративна методологія розробки ПЗ, яка включає в себе поєднання елементів Waterfall-методології та Agile-методології. Spiral-методологія передбачає поетапне вдосконалення продукту на основі результатів попередніх етапів тестування.

У спіральної моделі (див. рис. 1.2) життєвий шлях розроблювального продукту зображується у вигляді спіралі, яка, розпочавшись на етапі планування, розкручується з проходженням кожного наступного кроку. Головна особливість спіральної моделі – концентрація на можливі ризики. [6]

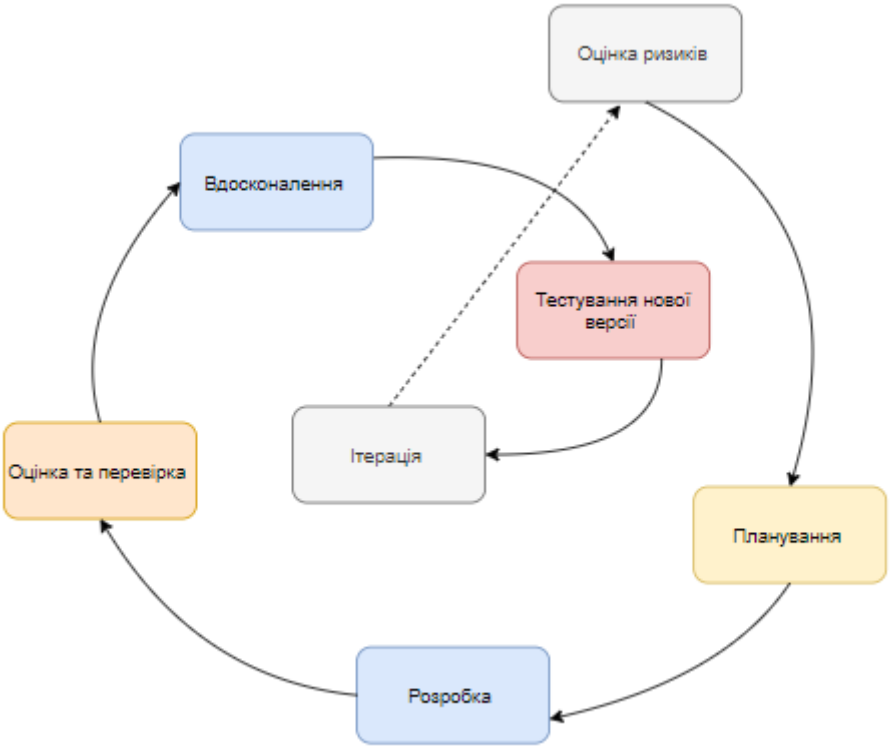


Рис. 1.2. Етапи тестування з використанням моделі Waterfall

Етапи тестування з використанням Spiral-моделі:

1. Оцінка ризиків – на першому етапі, тестувальники оцінюють ризики, пов'язані з розробкою продукту та створюють список ризиків, які можуть виникнути на різних етапах розробки;

2. Планування – на другому етапі, тестувальники створюють план тестування, який включає у себе детальні етапи плану тестування та бюджет.

3. Розробка – після етапу планування, розробники створюють першу версію продукту;

4. Оцінка та перевірка - після створення першої версії продукту, тестувальники перевіряють ПП на відповідність вимогам та виявляють баги, що можуть виникнути на наступних етапах розробки ПЗ;

5. Вдосконалення – під час цього етапу, розробники виправляють помилки та покращують функціональність ПП;

6. Тестування нової версії - після вдосконалення ПП, тестувальники проводять тестування нової версії та знову оцінюють ризики, пов'язані з версією продукту;

7. Ітерація – після етапу тестування нової версії ПП, цикл може знову повторюватись, починаючи з оцінки ризиків, доки ПП не досягне поставленої досконалості.

Переваги Spiral моделі:

– Високий рівень аналізу і контролю ризиків;

– Ранній випуск вдосконалених ПП;

– З точки зору методологій, гнучкість дуже високого рівня.

Недоліки Spiral моделі

– З точки зору масштабування проектів, модель не підходить для малих або середніх проектів;

– Вдалість створення ПП пропорційно залежить від якості аналізу ризиків і кваліфікованості спеціалістів.

1.3.3. Тестування з використанням методології TDD

Test Driven Development (TDD) - це методологія розробки ПЗ, яка передбачає написання тестів перед написанням коду. Всі тести, що проводяться у процесі розробки, створюють тестову піраміду. Незалежно від специфікації ПП, можна виділити такі обов'язкові види тестування:

Функціональне тестування – цей вид тестування дозволяє переконатися у коректності функціоналу ПП.

Модульне тестування – цей вид тестування дозволяє виконувати тести окремих частин - модулів ПП.

Циклічне тестування – цей вид тестування дозволяє виконувати замість одного циклу модулів тестів, повторення виконання тестів до досягнення визначеної «самостійності» ПП.

Для методології TDD, існують три закони: [8]

1) Не можна писати жодного вихідного коду, доки спершу не написано падаючого юніт-тесту;
2) Не можна писати більше юніт-тесту ніж необхідно для падіння (непроходження тесту). Помилка компіляції - це також падіння;
3) Не можна писати більше вихідного коду, ніж необхідно для проходження впалого юніт-тесту.

Падіння тесту - це ситуація, коли тест, який призначений для перевірки функціональності ПЗ, не проходить успішно. Таке тестування, зазвичай, здійснюється з метою виявлення помилок та недоліків у ПЗ, які потрібно виправити перед релізом.

Падіння тесту може бути пов'язано з великою кількістю причин: помилки у коді ПП, неправильні дані введені користувачем, відсутність необхідного з'єднання з мережею, тощо. Важливо відслідковувати та аналізувати падіння тестів, щоб виявити та виправити помилки в ПЗ та запобігти їх повторенню в майбутньому.

Основні принципи тестування з використанням методології TDD (див. рис. 1.3):

– Тести пишуться перед написанням коду;

– Тестування передбачається на всіх етапах розробки;

– Кожен новий етап розробки передбачає написання нових тестів;

– Код написаний під час етапу розробки ПЗ, повинен відповідати вимогам тестів;

– Після написання коду, виконується перевірка успішності виконання всіх тестів.

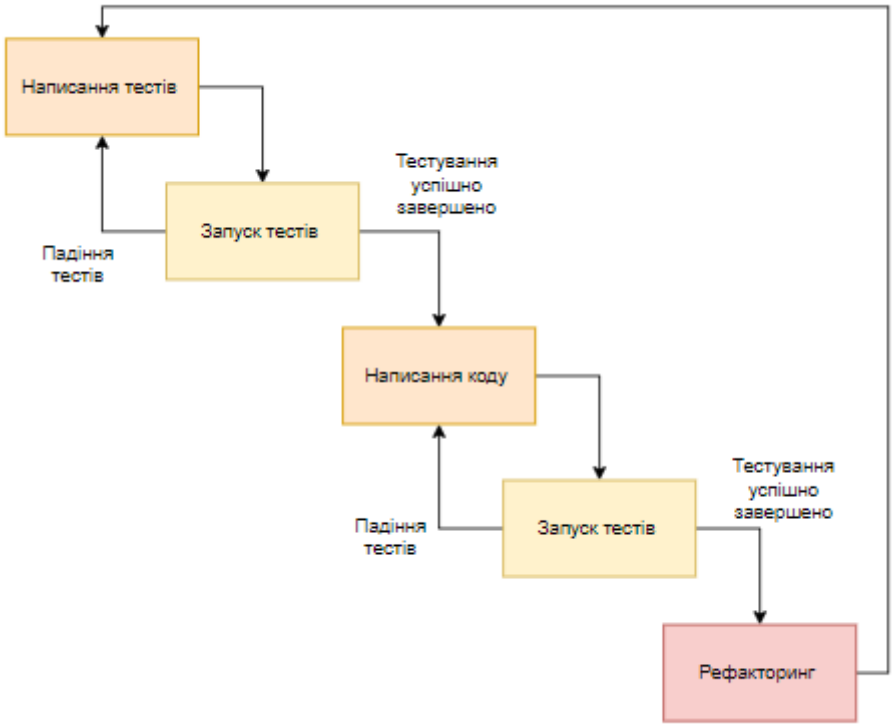


Рис. 1.3. Етапи тестування з використанням моделі TDD

Етапи тестування з використанням методології TDD:

1. Написання тестів – на першому етапі, тестувальники пишуть тести, які передбачають певну функціональність ПП;

2. Запуск тестів – на другому етапі, тестувальники запускають тести для визначення успішності проходження тестів;

3. Написання коду – після етапу запуску тестів, розробники створюють код, який відповідає вимогам тесту;

4. Запуск тестів – на цьому етапі, тестувальники запускають тести для визначення успішності проходження тестів нового коду;

5. Рефакторинг – на останньому етапі, після успішного проходження всіх тестів, розробники можуть вдосконалити код, зробивши його більш досконалим та естетичним для читання.

1.3.4. Тестування з використанням методології Agile

Agile — система ідей і принципів «гнучкого» управління проектами, де ключовий принцип — розробка через короткі ітерації (цикли), в кінці кожного з яких замовник (користувач) отримує робочий код або продукт. [5]

Тестування з використанням методології Agile (див. рис. 1.4) - популярний варіант для тестування в тенденціях сучасної розробки ПЗ. Agile є ітеративною та інкрементальною методологією розробки ПС, де функціональність програми розробляється за короткі періоди, які називаються ітераціями або спринтами і мають терміни 1-3 тижні.

Тестування з використанням методології Agile орієнтоване на постійну безперервну інтеграцію та постійне вдосконалення коду до вимог ПП. Використання автоматизованих тестів є ключовим елементом Agile тестування для швидкого виявлення багів та недоліків.

У принципі Agile, тестування відбувається кожної ітерації одразу після розробки модуля. Тестування проводиться в першу чергу на функціональному та інтеграційному рівнях і на рівні прийняття користувачів. Тестувальники та розробники працюють однією командою для забезпечення високої якості ПЗ. Основна особливість методології Agile полягає у тісній взаємодії тестувальників і розробників для забезпечення високої якості ПП з швидким реагуванням на зміни вимог та внесенню корективів.

Методологія Agile описана «Маніфестом гнучкої розробки ПО» з прописаними 12 принципами Agile- розробки, які зведені у 4 тези [5]:

– Люди і взаємодія важливіші процесів та інструментів;

– Робочий продукт важливіший вичерпної документації;

– Співпраця з замовником важливіша узгодження умов контракту;

– Готовність до змін важливіша проходження попереднім планом.

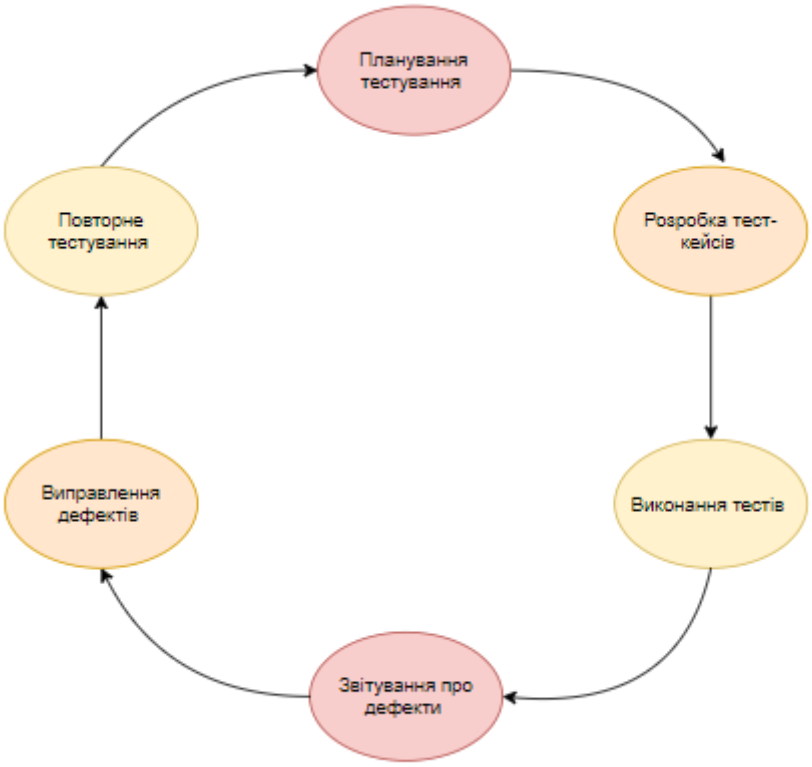


Рис. 1.4. Етапи тестування з використанням моделі Agile

Етапи тестування з використанням методології Agile:

1. Планування тестування - на першому етапі, тестувальники та розробники обговорюють вимоги до ПЗ та створюють план тестування, визначаючи автоматизацію тестування;

2. Розробка тест-кейсів - на другому етапі, тестувальники створюють тест-кейси, які можуть бути створені для кожної нової функції ПЗ, або для тестування загального функціоналу, який може змінитися під час нової ітерації і будуть використовуватись для виконання тестування;

3. Виконання тестів – на третьому етапі, тестувальники виконують тест-кейси для перевірки працездатності ПЗ, згідно з створеними вимогами;

4. Звітування про дефекти – після етапу виконання тестів і знаходження дефектів, тестувальники документують у баг-репорти про помилки відповідальним за виправлення помилок розробникам;

5. Виправлення дефектів - після отримання повідомлення про баг, розробники виправляють його і виконують перевірку правильності роботи ПЗ;

6. Повторне тестування – після етапу виправлення дефектів, ПЗ повторно тестується для перевірки усунення знайдених багів і пошуку створення нових можливих.

Переваги тестування з використанням методології Agile:

– Швидкий цикл розробки;

– Високий рівень гнучкості;

– Прозорість дій команди.

Недоліки тестування з використанням методології Agile:

– Високі вимоги до комунікації команди;

– Необхідність великого обсягу документації;

– Необхідність високого рівня компетентності команди;

– Підвищені ризики, пов'язані з постійними змінами вимог.

1.3.5. Тестування з використанням моделі Extreme Programming

Extreme Programming (XP) – це одна з Agile — методологія розробки ПЗ, де важлива роль відводиться періодичній грі в планування із залученням замовника[5], яка передбачає тісне співробітництво між розробниками та тестувальниками на всіх етапах розробки.

Дана методологія (див. рис. 1.5) має на меті поліпшення якості ПЗ та чутливість до змін у вимогах замовників. Як вид гнучких методологій, XP радить часті "випуски" програми у коротких циклах розробки, що має на меті поліпшити продуктивність праці та покращити можливості виконання вимог замовника що змінюються.[7]

Основні принципи тестування з використанням моделі Extreme Programming:

– Тестування передбачається на всіх етапах розробки;

– Тестування як і розробка виконується попарно: перший спеціаліст виконує саму роботу, а другий контролює процес виконання і загальну картину;

– Тестування проводиться автоматично за допомогою тестових фреймворків;

– Тестувальники тісно взаємодіють з розробниками для виявлення та виправлення помилок на ранніх етапах розробки;

– Кожен етап розробки ПЗ завершується проведенням тестів, які дозволяють знаходити баги одразу після розробки та зменшують ризики їх виникнення в подальшому.

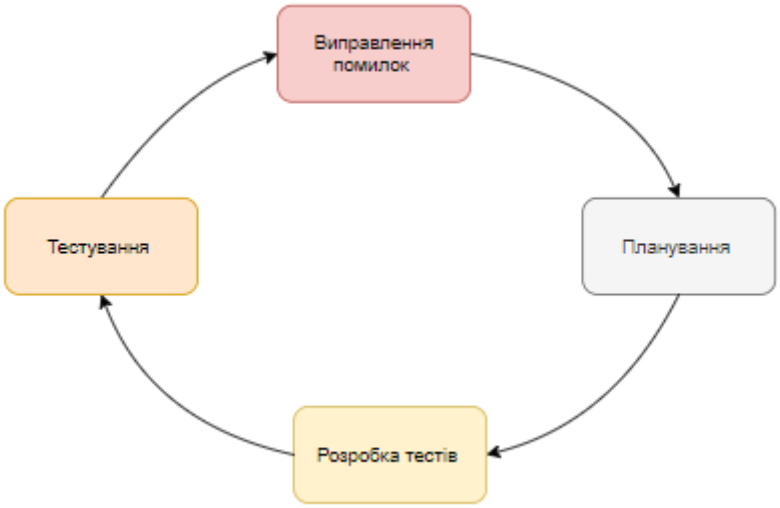


Рис. 1.5. Етапи тестування з використанням моделі XP

Етапи тестування з використанням моделі Extreme Programming:

1. Планування - на першому етапі тестувальники разом з розробниками планують проведення тестів для кожного етапу розробки;

2. Розробка тестів – на другому етапі, тестувальники разом з розробниками розробляють тест-кейси та тестові фреймворки;

3. Тестування – після етапу розробки тестів, тестувальники проводять тести та виявляють баги;

4. Виправлення помилок – під час цього етапу, розробники виправляють виявлені помилки;

5. Повторення - цикл повторюється, починаючи з етапу планування.

1.3.6. Тестування з використанням методології DevOps

DevOps - це методологія, яка об'єднує розробку ПЗ (Development - Dev) та експлуатацію (Operations - Ops). Метою даної методології є збільшення швидкості розробки та впровадження ПЗ за рахунок зменшення часових ресурсів на процеси розробки, тестування, забезпечення якості та впровадження ПЗ.

DevOps (див. рис. 1.6) включає в себе автоматизацію процесів, засоби контролю якості, інструменти моніторингу, системи відстеження помилок та багів, а також швидке впровадження змін та автоматичну забезпечення безперервної інтеграції та безперервної поставки.

Agile і DevOps схожі, але Agile являє собою зміну мислення і практики (що має привести до організаційних змін), в той час коли DevOps приділяє більше уваги впровадженню організаційних змін для досягнення своєї мети. Потреба в DevOps зросла у відповідь на дедалі більший успіх Agile-розробки через прагнення організацій готувати випуски частіше й швидше.[9]

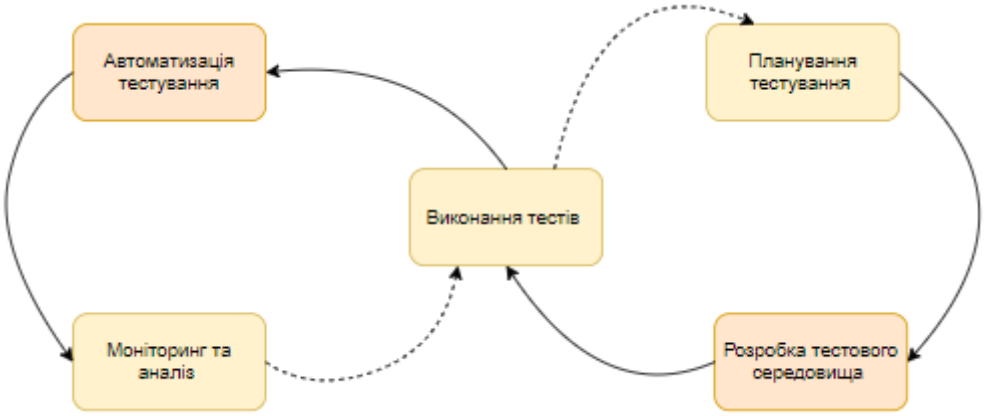


Рис. 1.6. Етапи тестування з використанням моделі DevOps

Етапи тестування з використанням методології DevOps:

1. Планування тестування – на першому етапі визначаються тестові вимоги та плануються тестові сценарії;

2. Розробка тестового середовища – на другому етапі, створюється інфраструктура для тестування, налаштування систем автоматизованого тестування та інших спеціалізованих інструментів;

3. Виконання тестів – на цьому етапі, тестувальники запускають тестові сценарії, аналізують результати та документують баги та проблем;

4. Автоматизація тестування – після етапу виконання тестів, тестувальники додатково автоматизують тести для забезпечення постійної інтеграції та постачання ПЗ;

5. Моніторинг та аналіз – на останньому етапі, тестувальники виконують постійний моніторинг процесу розробки та аналізу результатів тестування для виявлення багів та забезпечують постійне вдосконалення процесу розробки.

Переваги тестування з використанням методології DevOps:

– Зменшення часу на процеси розробки, тестування, забезпечення якості та впровадження ПЗ;

– Автоматизація процесів тестування та забезпечення безперервної інтеграції допомагає зменшити кількість багів та ризиків їх подальшого виникнення в ІС;

– Тісна співпраця між тестувальниками і розробниками, а також спеціалістами з експлуатації систем;

– Забезпечення високої якості ПП.

Недоліки тестування з використанням методології DevOps:

– Складність у першому впровадженні DevOps;

– Необхідність високого рівня компетенції спеціалістів;

– Підвищені фінансові витрати на специфічні інструменти і автоматизацію;

– Відсутність гнучкості масштабованості, через що не підходить малим компаніям.

1.4. Висновок до 1 розділу

У даному розділі було досліджено проблеми тестування програмного забезпечення та огляд існуючих підходів, методів, практик та методологій, які використовуються при комплексному тестуванні. Було визначено, що:

1. Тестування - не якість програмного продукту, проте обидва тісно взаємопов’язані;
2. Проблематика низької якості програмного продукту взаємозалежна з компетенцією спеціалістів;
3. Не існує загального комплексного тестування, яке може бути досконалим для будь-якого розроблюваного програмного продукту;
4. Існує велика сукупність підходів до тестування як традиційних, так і набуваючих тенденцію, які комплексно вирішують питання якості програмного продукту;
5. Для великого і малого бізнесу необхідні різні комплексні комбінування підходів.

Також було проведено огляд найпоширеніших підходів та методологій до тестування розроблюваного ПЗ, визначені відмінності, переваги і недоліки, які зазначені у таблицях.

Таблиця 1.1

Переваги і недоліки різних підходів до тестування

Тип тестування	Опис	Переваги	Недоліки
Тестування з ШІ	Використання штучного інтелекту для автоматичного тестування	Швидкість, точність, ефективність	Високі витрати на розробку та налаштування

Продовження таблиці 1.1

Тестування без тестувальників	Автоматичне тестування без присутності людських тестувальників	Швидкість, ефективність	Не може виявити всі можливі помилки
Тестування без розробників	Тестування, яке виконується без присутності розробників	Можливість виявлення помилок з точки зору клієнта	Обмежена можливість виявлення технічних проблем
Тестування в реальному часі	Тестування в режимі реального часу, коли система активна	Можливість виявлення реальних помилок	Вимагає високої якості мережі та обладнання
Тестування у контейнері	Тестування, яке виконується в ізольованому середовищі	Можливість виконання багатьох тестів одночасно	Обмежена можливість виявлення проблем, пов'язаних зі збіркою
Хмарне тестування	Тестування, яке виконується на серверах у хмарі	Зменшення витрат на обладнання та підтримку	Вимагає швидкого та стабільного Інтернет-з'єднання
Тестування з технологією блокчейн	Тестування, яке використовує блокчейн для збереження результатів	Безпека, надійність, захист від змін результатів	Високі витрати на розробку та налаштування

Таблиця 1.2

Відмінності використання різних моделей у тестуванні

Модель	Опис	Рівень гнучкості	Цикл розробки	Роль тестувальника
Waterfall	Послідовний підхід до розробки ПЗ, де кожен етап виконується послідовно, поки не завершиться передачею продукту на наступний етап	Низький	Строго послідовний	Окрема роль, що виконує тестування після закінчення розробки
Spiral	Підхід, що поєднує підхід Waterfall з ітераційним підходом, де кожна ітерація включає в себе всі етапи Waterfall	Високий	Ітераційний	Окрема роль, що виконує тестування на кожному етапі розробки
Test Driven Development (TDD)	Методологія розробки, де спочатку пишуться тести, а потім код для проходження цих тестів	Високий	Ітераційний	Розробники виконують тестування на рівні коду
Agile	Гнучка методологія розробки, що дозволяє ефективно працювати з змінними вимогами	Високий	Ітераційний	Тестувальник займається тестуванням на різних етапах розробки, виконує все тестування

Продовження таблиці 1.2

Extreme Programming	Гнучкий підхід до розробки, де програмісти працюють в парах та здійснюють постійну інтеграцію та тестування коду	Високий	Ітераційний	Тестувальник займається тестуванням на різних етапах розробки, співпрацює з розробниками
DevOps	Гнучкий підхід до розробки, де програмісти працюють в парах та здійснюють постійну інтеграцію та тестування коду	Високий	Ітераційний	Тестувальник бере участь в усьому процесі розробки, виконує тестування на різних етапах, співпрацює з розробниками та адміністраторами

Продовження таблиці 1.2

Waterfall	Тестування на кінці процесу розробки	Функціонально розділені команди	Простота, чіткість, передбачуваність, можливість заздалегідь планувати	Висока ступінь ризику, обмежений зворотний зв'язок, не ефективний для проектів з багатьма невідомими
Spiral	Тестування на кожній ітерації	Функціонально розділені команди	Гнучкість, можливість змінювати специфікації, ефективність для проектів з багатьма невідомими	Складність управління, можливість не виконати кінцеву мету проекту
Test Driven Development (TDD)	Автоматичне тестування на кожній ітерації	Функціонально розділені команди	Висока якість коду, документація тестів, менша кількість помилок, швидший процес розробки	Необхідність у високій компетентності тестувальників
Agile	Тестування на кожній ітерації, а також весь час під час розробки	Функціонально злиті команди	Гнучкість, взаємодія з клієнтом, зменшення часу розробки	Відсутність повної специфікації, необхідність постійного оновлення планів

Продовження таблиці 1.2

Extreme Programming	Тестування на кожній ітерації, а також весь час під час розробки	Функціонально злиті команди	Швидкість розробки, висока якість коду, стабільність продукту, взаємодія з клієнтом, зменшення ризику, простота управління проектом	Необхідність високої компетентності програмістів, складність в організації парного програмування, не підходить для всіх проектів
DevOps	Постійний цикл інтеграції та доставки (CI/CD)	Функціонально злиті команди і спеціалісти	Швидкість випуску, підвищена ефективність, зниження ризику, автоматизація процесу розробки та випуску продукту, краща взаємодія між розробниками та іншими зацікавленими сторонами	Потрібність високої компетентності команди, висока складність організації, ризик негативного впливу на безпеку даних та конфіденційність

РОЗДІЛ 2
ВИБІР ТЕХНОЛОГІЙ ТА ІНСТРУМЕНТІВ ДЛЯ КОМПЛЕКСНОГО ТЕСТУВАННЯ

2.1. JavaScript

Для написання тестів було обрано мову програмування JavaScript, оскільки JS є дуже популярною для автоматизованого тестування веб-додатків і вона є однією з найпопулярніших мов програмування веб-розробки.

В кваліфікаційні роботі, мова програмування JS використовується для написання функціональних та модульних тестів з використанням фреймворка JSR223 Sampler.

Основні переваги використання JS:

1) JavaScript – кроссбраузерна мова програмування: однією з головних причин, чому JavaScript є таким популярним, є те, що він може виконуватись у більшості браузерів;

2) JavaScript має велику кількість фреймворків для тестування таких як Jest, Mocha, Jasmine та ін., які дозволяють створювати високоорганізовані, підтримувані та надійні автотести;

3) JavaScript має велику кількість інструментів, які допомагають виконувати автоматизоване тестування, як приклад: Selenium, JMeter, Puppeteer, TestCafe та ін., за допомоги яких легко автоматизується взаємодія з веб-додатками та створюються складні тестові сценарії;

4) JavaScript - основна мова програмування веб-розробки, через це, написані тести, можуть бути ефективніше, оскільки вони використовують ту саму мову, що й для розробки веб-додатків;

5) JavaScript – велика цифрова екосистема - вона має велику кількість пакетів та бібліотек, які дозволяють зробити процес написання тестів більш ефективним та зручним;

6) JavaScript є мовою високого рівня, що означає, що дозволяє не занадто детально знати процеси, що відбуваються під час виконання програм;

7) JavaScript дозволяє створювати тести для більшості типів додатків – від веб-додатків, до комп'ютерних програм.

2.2. JMeter

JMeter - це безкоштовний універсальний інструмент для тестування, який написаний на мові Java і використовується для вимірювання функціональності та продуктивності веб-додатків. Цей інструмент, зазвичай, призначений для тестування веб-додатків, але на сьогоднішній день, він здатний проводити навантажувальні тести для JDBC-з'єднань, FTP, LDAP, SOAP, JMS, POP3, IMAP, HTTP і TCP.[11]

JMeter надає можливості для створення та виконання різних типів тестів, таких як навантаження, стрес-тестування, тестування витривалості, тестування безпеки та інших. Інструмент дозволяє створювати великі обсяги вхідних даних, що допомагає перевірити роботу додатків за різних умов, зокрема, при великому навантаженні. JMeter може бути використаний для тестування різних типів протоколів, таких як HTTP, HTTPS, FTP, SOAP, REST і JDBC.

Для кваліфікаційної роботи був обраний саме JMeter як інструмент комплексного тестування оскільки:

– JMeter є безкоштовним, кросплатформений і відкритим інструментом;

– JMeter має простий інтерфейс, що дозволяє швидко створювати тести;

– JMeter дозволяє створювати складні сценарії тестування, які можуть імітувати поведінку користувачів на веб-сайті або веб-додатку;

– JMeter може працювати з багатьма користувачами, що дозволяє перевірити продуктивність веб-додатків під високим навантаженням;

– JMeter підтримує автоматизацію створення тестів, що дозволяє їхнє швидке виконання та збереження часу;

– JMeter має велику кількість плагінів та розширень, які дозволяють підвищити функціональність інструмента.

Функції Apache JMeter включають:

1. Можливість завантаження та тестування продуктивності безлічі Повнофункціональна IDE для тестування, яка дозволяє швидко записувати план тестування (з браузерів або власних програм), створювати та налагоджувати файли:

– різних типів додатків/серверів/протоколів;

– Інтернет - HTTP, HTTPS (Java, NodeJS, PHP, ASP.NET, …);

– Веб-сервіси SOAP/REST;

– FTP;

– База даних через JDBC;

– LDAP;

– Проміжне програмне забезпечення, орієнтоване на повідомлення (MOM) через JMS;

– Пошта - SMTP(S), POP3(S) та IMAP(S);

– Власні команди або сценарії оболонки;

– TCP;

– Java-об'єкти.

2. Режим командного рядка (режим командного рядка (який раніше був без графічного інтерфейсу користувача)/безголовий режим) для завантаження тесту з будь-якої ОС, сумісної з Java (Linux, Windows, Mac OSX, …);

3. Повний та готовий до подання динамічний HTML-звіт;

4. Проста кореляція завдяки можливості отримувати дані з найпопулярніших форматів відповідей, HTML, JSON, XML або будь-якого текстового формату;

5. Повна переносимість та 100% чистота мови програмування;

6. Повна багато потокова структура дозволяє виконувати паралельну вибірку багатьма потоками та одночасну вибірку різних функцій окремими групами потоків;

7. Кешування та автономний аналіз/відтворення результатів тестування;

8. Розширюване ядро:

– Вбудовуванні плагіни, які відкривають необмежені можливості тестування;

– Scriptable Samplers (специфіка стандарту API – “JSR223”-сумісні мови, такі як JS, Groovy, BeanShell, Ruby, Python та ін.);

– За допомогою змінних таймерів можна вибрати кілька статистичних даних для завантаження;

– Плагіни для аналізу та візуалізації даних забезпечують велику розширюваність, а також персоналізацію;

– Функції можна використовувати для динамічного введення даних у тест або обробки даних;

– Проста безперервна інтеграція через сторонні бібліотеки з відкритим вихідним кодом для Maven, Gradle та Jenkins. [12]

Для проведення функціонального тестування, JMeter також має вбудовані компоненти, які дозволяють виконувати різні дії над запитами, такі як:

– Assertions: перевірка правильності відповіді від сервера, наприклад, перевірка наявності певного тексту або статус-коду відповіді;

– Extractors: дозволяє витягти певну інформацію з відповіді сервера та використовувати її у наступних запитах;

– Pre-Processors та Post-Processors: дозволяють виконувати певні дії перед відправкою запиту або після отримання відповіді;

– Контрольні точки (Checkpoints): дозволяють встановлювати точки в коді, які потрібно перевірити під час виконання тесту.

Щоб провести модульне тестування з JMeter, можна використовувати HTTP запити до кожної сторінки веб-додатку, яку потрібно протестувати. HTTP запити

можуть бути створені вручну або записані за допомогою HTTP(S) Test Script Recorder. Після того, як запити будуть створені або записані, їх можна сконфігурувати для виконання різних функціональних тестів, наприклад, перевірка правильності відображення сторінок, робота форм та інших елементів веб-додатку.

2.3. OWASP ZAP

OWASP ZAP - це безкоштовний інструмент з відкритим кодом, який використовується для проведення тестів на проникнення. Основною метою Zap є забезпечення легкого тестування на проникнення для виявлення вразливостей у веб-додатках.[13]

Для кваліфікаційної роботи був обраний OWASP ZAP як інструмент комплексного тестування, оскільки:

- Основними можливостями OWASP ZAP є сканування веб-застосунків на наявність вразливостей, автоматизація тестування, моніторинг сеансів користувача та аналіз трафіку;
- Інструмент має графічний інтерфейс, що робить його використання зручнішим для користувачів, які не мають досвіду роботи з командним рядком;
- Цей інструмент надає користувачеві багато налаштувань, які дозволяють встановити його під конкретні потреби. Наприклад: користувач може встановити критерії фільтрації, які будуть використовуватися для визначення вразливостей. Також є можливість налаштування способу автентифікації, що дозволяє проводити тестування веб-застосунків, що вимагають авторизації.

Переваги OWASP ZAP:

1. Підтримка різних видів тестування, такі як сканування вразливостей, тестування на перехоплення даних та тестування на проникнення. Це дозволяє проводити комплексне тестування веб-додатків на наявність вразливостей;
2. OWASP ZAP також має функцію автоматичного виявлення вразливостей, що спрощує процес тестування та дозволяє виявляти вразливості

ефективніше. Інструмент також дозволяє створювати звіти про знайдені вразливості, що спрощує процес аналізу результатів тестування;

3. Безкоштовність використання, що робить його доступним для широкої аудиторії користувачів. Завдяки відкритому вихідному коду користувачі можуть налаштовувати інструмент під свої потреби та вносити свої покращення;

4. Розробка інструменту спільнотою OWASP, яка включає експертів в галузі безпеки веб-додатків. Це забезпечує його актуальність та відповідність сучасним вимогам безпеки;

5. Наявність графічного інтерфейсу, що робить його використання зручнішим для користувачів, які не мають досвіду роботи з командним рядком;

6. Створення звітів про знайдені вразливості, що спрощує процес аналізу результатів тестування.

В цілому, OWASP ZAP є потужним інструментом для тестування безпеки веб-застосунків, який дозволяє виявляти вразливості та покращувати рівень безпеки веб-додатків. Завдяки підтримці спільноти OWASP, інструмент забезпечує своєчасні оновлення та відповідність вимогам безпеки, що робить його одним з найбільш популярних інструментів для тестування безпеки.

2.4. JSR223 Sampler

JSR223 Sampler - це один із видів семплерів для Apache JMeter.

Семплери виконують фактичну роботу JMeter. Кожен семплер генерує один або кілька результатів вибірки. Результати вибірки мають різні атрибути (успіх/невдача, час, розмір даних і т. д.) і можуть бути переглянуті в різних слухачах.[14]

Для кваліфікаційної роботи був обраний JSR223 Sampler, оскільки він дозволяє використовувати скрипти різними мовами програмування, включаючи Java, JavaScript, Ruby, Python і Groovy, для створення користувацьких сценаріїв тестування.

У JSR223 Sampler є кілька спільних особливостей для всіх мов:

– У скрипті можна використовувати змінні JMeter, наприклад ${__time()} - вбудована змінна, що повертає поточний час в мілісекундах;

– JSR223 Sampler може використовуватись для виконання як простих, так і складних скриптів;

– У скрипті можна використовувати спеціальні методи, наприклад log.info() - для запису повідомлень в журнал JMeter;

– При використанні JSR223 Sampler необхідно забезпечити безпеку скрипту. В іншому випадку зловмисник може використовувати скрипт для виконання шкідливого коду на сервері. Для цього можна використовувати засоби, що надаються Apache JMeter, наприклад, налаштування безпеки та обмеження виконання скрипту;

– JSR223 Sampler має доступ до контексту виконання тесту, що дозволяє використовувати змінні та функції, визначені в інших компонентах тесту.

2.5. Висновок до 2 розділу

У поточному розділі було обрано і інструменти та технології для реалізації комплексного тестування ПП і аргументовано їх вибір:

- JavaScript, як основна мова програмування для написання автотестів;
- JMeter, як комплексний програмний засіб для проведення функціонального, модульного, системного та навантажувального тестування;
- JSR223 Sampler як доповнення до JMeter для реалізації функціонального, модульного та системного тестування;
- OWASP ZAP, як комплексний програмний засіб для проведення тестування безпеки, ризиків вразливостей і отримання документації результатів тестів.

Завдяки цих технологій, буде проведено комплексне тестування існуючого ПП – проекту Playzone класифікаціями тестів, а саме:

- За знанням нутрощів ІС: тестування сірого ящику (grey box testing);
- За фокусом на рівні архітектур додатків: тестування представлення (presentation tier testing), тестування бізнес-логіки (business logic tier testing);

- За об’єктами тестування: функціональне тестування (functional testing), швидкості і надійності / навантажувальне тестування (load/stress/performance testing), тестування безпеки (security testing), бізнес-тестування: тестування, тестування UI / користувацького інтерфейсу (UI testing), досвіду користувача (usability testing);
- За часом проведення: реактивне тестування (reactive testing);
- За критерієм позитивності сценаріїв: позитивне тестування (positive testing), негативне тестування (negative testing);
- За ступенем втручання у роботу системи: неінвазивне тестування (nonintrusive testing), інвазивне тестування(intrusive testing);
- За вибором вхідних даних: доменне тестування (domain testing);
- За ступенем ізольованості: компонентне тестування (component testing), інтеграційне тестування (integration testing), системне тестування (system or end-to-end testing);
- За ступенем автоматизації: ручне/мануальне тестування (manual testing), автоматизоване тестування (automated testing);
- За ступенем підготовки до тестування: тестування по тест-кейсам (documented testing), дослідницьке тестування (exploratory testing), інтуїтивне тестування (ad hoc testing).

РОЗДІЛ 3
ТЕСТУВАННЯ МЕТОДИКАМИ КЛАСІВ ПРОГРАМНОГО ПРОДУКТУ

Існує велика кількість різних методів тестування програмного продукту які описуються різними стандартами тестування таких як IEEE 829, ISO/IEC 29119, ISTQB (International Software Testing Qualifications Board) та інших. Вони використовуються для забезпечення належного рівня якості продукту.

Оскільки планування тестування вимагає від тестувальника найбільшої творчості та професіоналізму, (тому що саме в ньому вирішується безліч головоломок, які відповідають на одне просте запитання: "Як будемо тестувати?»)[15], проведемо тестування за найпопулярнішими класифікаціями.

Оскільки більшість класів тестування взаємопов’язані, виділимо класи тестів, за якими буде проводитись комплексне тестування:

1. Тестування безпеки:
   - Пантестування;
   - Тестування на вразливості.
2. Навантажувальне тестування:
   - Тестування навантаження;
   - Стресове тестування;
   - Тестування на масштабованість.
3. Функціональне тестування:
   - Тестування критичних значень;
   - Тестування еквівалентних значень;
   - Тестування сценаріїв використання.

3.1. Тестування безпеки (security testing)

Тестування безпеки (security testing) – це клас комплексного тестування, який визначається як процес перевірки ІС на наявність вразливостей, можливих кібератак та недоліків у механізмах захисту, а також оцінки загального рівня безпеки системи.

Для тестування безпеки проекту Playzone, було використано такі методики тестування:

Тестування на проникнення (penetration testing або як ще її називають - пантестування), яке включає активне тестування з метою виявлення вразливостей і перевірки захисту системи від атак;

Тестування на вразливості (vulnerability assessment), яке використовується для виявлення вразливостей у системі та оцінки їхнього ступеня критичності;

Суть пантестів полягає в тому, щоб спробувати зламати систему, як це міг би зробити зловмисник, використовуючи різні методи атаки: соціальна інженерія, зламування пароля, експлойти (виведення з ладу функціональності ІС).

Основною метою проведення пентестингу є виявлення вразливостей у системі та оцінка рівня її захищеності. Пентестинг можна проводити як вручну, так і автоматизовано (що і було зроблено), використовуючи спеціалізовані інструменти.

Дана методика класу тестів проводилася за згодою власника проекту, оскільки несанкціонований пентестинг може призвести до незаконних дій і завдати шкоди ІС або організації загалом.

Суть перевірки на вразливості є сканування системи, використовуючи комплексні інструменти, які виконують саме сканування системи та ПЗ для знаходження можливих вразливостей, які можуть включати в себе перевірку конфігурації системи та її складових, перевірку безпеки мережі та інші процеси.

Основною метою перевірки на вразливості є забезпечення захисту системи шляхом виявлення та виправлення можливих проблем з безпекою. Перевірка на вразливості повинна проводитись регулярно, оскільки зловмисники постійно шукають нові способи атак на системне та ПЗ.

Найголовніша відмінність між пентестами та тестами на вразливість полягає у тому, що пентести – це активний процес тестування, який дозволяє протестувати систему на реальних умовах, використовуючи різноманітні методи атак, тоді як тести на вразливості сканують ІС для виявлення можливих проблем з безпекою, але не перевіряє їх в реальних умовах.

Такі тести є актуальними для виконання після випуску ПП.

Як зазначає автор книги «Тестування ПЗ», С. Куліков – тестування безпеки можливе за методами чорного і білого ящика [16], а отже і можливе їхнє поєднання у методі сірого ящика, що і було зроблено у кваліфікаційній роботі.

Тестування безпеки включає також в себе класи: тестування представлення, реактивне тестування, негативне тестування, не інвазивне тестування, інвазивне тестування, доменне тестування, компонентне тестування, системне тестування, ручне/мануальне тестування, автоматизоване тестування, дослідницьке тестування, інтуїтивне тестування.

Тестування безпеки було виконано, використовуючи комплексне ПЗ AWASP ZAP.

У процесі тестування безпеки проекту Playzone також була використана методологія тестування зверху-вниз (top-down testing), яка починається з сканування і тестування всієї системи загалом для перевірки коректної роботі. Після чого, тестувалися окремі компоненти системи для перевірки їхньої поведінки.

Загальний алгоритм дій виглядає таким чином:

1. Системне тестування безпеки:
    a. Запуск тестування вразливостей:
        i. Сканування системи;
        ii. Пошук вразливостей;
        iii. Аналіз вразливостей.
    b. Зустріч і обговорення результатів аналізу з власником проекту;
    c. Пантестування:
        i. Вибір методик тестування;

ii. Налаштування автотестів;

iii. Виконання тестування;

iv. Аналіз пантестів.

d. Створення звітів тестування;

e. Зустріч і обговорення результатів тестування з власником проекту;

f. Виправлення багів.

Перед запуском системних тестів на безпеку, створюємо у комплексному ПЗ AWASP ZAP нову сесію, налаштовуючи сканер ІС проекту Playzone для дослідження дерева файлів ІС через посилання http-запитів до сервера.

Після дослідження проекту, маємо таку файлову структуру Playzone (див. рис. 3.1):

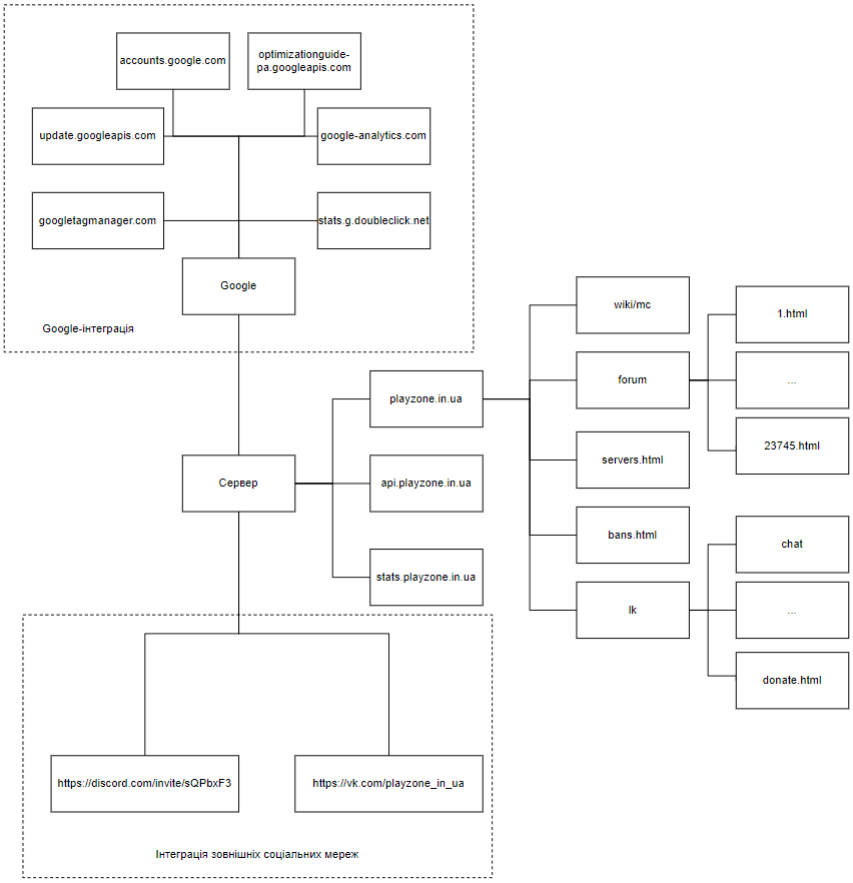


Рис. 3.1. Файлова структура проекту Playzone.in.ua

За результатами сканування і пошуку вразливостей (див. рис. 3.2), було знайдено 27 вузлів з різними рівнями ризику вразливостей (див. рис. 3.3 – 3.7).

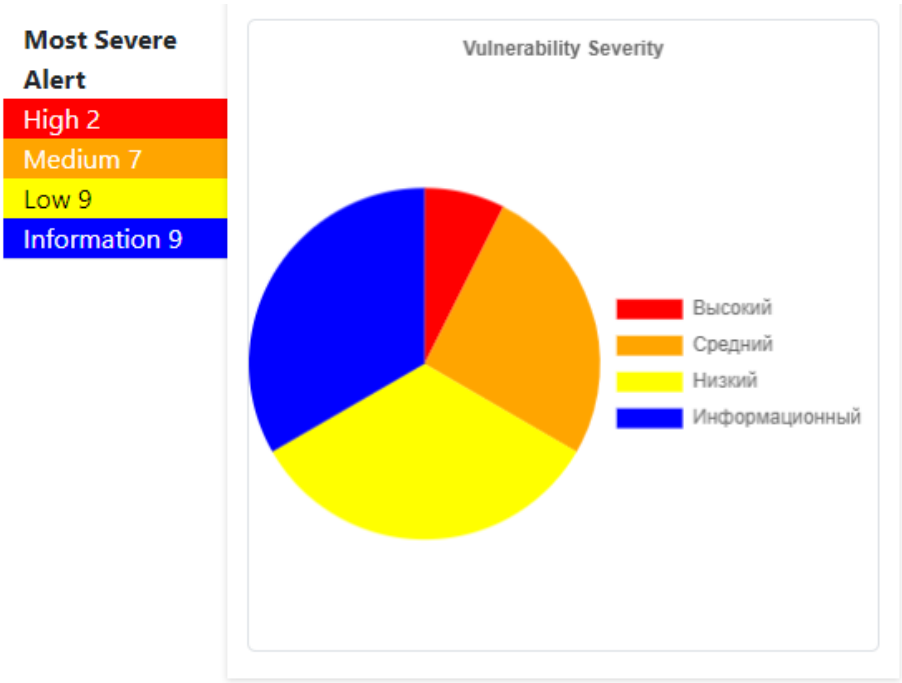


Рис. 3.2. Графік знайдених вузлів ризику Playzone.in.ua

Название	Уровень риска	Количество экземпляров
SQL-инъекция	Высокий	1
SQL-инъекция - SQLite	Высокий	3
Referer предоставляет идентификатор сеанса	Средний	35
Заголовок Content Security Policy (CSP) не задан	Средний	199
Идентификатор (ID) сеанса при перезаписи URL	Средний	35
Междоменная неправильная конфигурация	Средний	11
Отсутствует заголовок (Header) для защиты от кликджекинга	Средний	186
Отсутствуют токены против CSRF атак	Средний	660
Уязвимость JS Библиотеки (Library)	Средний	1
Cookie No HttpOnly Flag	Низкий	294
Cookie без атрибута SameSite	Низкий	632
Cookie без флажка безопасности	Низкий	637
Включение исходного файла междоменного JavaScript	Низкий	386
Заголовок Strict-Transport-Security не установлен	Низкий	273
Заголовок X-Content-Type-Options отсутствует	Низкий	213
Раскрытие отметки времени - Unix	Низкий	1376
Раскрытие ошибок приложения	Низкий	1
Сервер утечка информации о версии через поле заголовка HTTP-ответа «Server»	Низкий	4
Атрибут элемента HTML, управляемый пользователем (потенциальный XSS)	Информационный	172
Заголовок Content-Type отсутствует	Информационный	1
Пересмотрите директивы управления кэшем	Информационный	189
Получено из кеша	Информационный	145
Пользовательский Агент Fuzzer	Информационный	53
Раскрытие информации - конфиденциальная информация в URL	Информационный	6
Раскрытие информации - конфиденциальная информация в заголовке реферера HTTP	Информационный	1
Раскрытие информации - подозрительные комментарии	Информационный	75
Современное веб-приложение	Информационный	183

Рис. 3.3. Таблиця екземплярів знайдених вузлів ризику Playzone.in.ua

Рис. 3.4. Деталізований опис першого вузла критичного рівня вразливостей

Рис. 3.5. Деталізований опис другого вузла, першого екземпляра критичного рівня вразливостей

Рис. 3.6. Деталізований опис другого вузла, другого екземпляра критичного рівня вразливостей

Рис. 3.7. Деталізований опис другого вузла, третього екземпляра критичного рівня вразливостей

У розмові з власником проекту, який є і головним його розробником, було прийнято рішення детальніше розглянути лише вузли критичного рівня вразливостей, а саме перший вузол критичного рівня можливої вразливості SQL-ін’єкції, оскільки «у другому вузлі з 3 екземплярами затримка пов'язана з обходом кешу CloudFlare при зміні параметра GET».

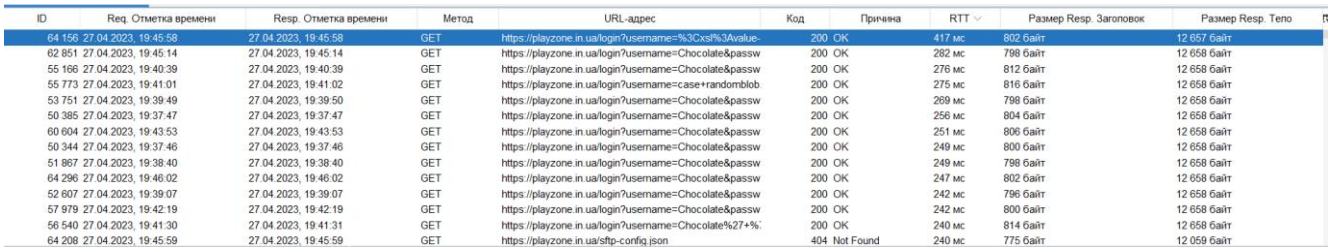
У першому вузлі критичного рівня вразливостей, а саме SQL-ін’єкції сканування визначило можливе враження у ІС проекту можливу SQL-ін’єкцію на етапі авторизації користувача.

Для виконання пантестів використовувались мануальне тестування шляхом вбудовування SQL-запитів та скриптів мови програмування JavaScript у поля форми авторизації: значення атрибута value для елемента input HTML-розмітки для поля «Логін», «Пароль», натиску кнопки авторизації, а потім автотестів за допомоги комплексного ПЗ AWASP ZAP.

Результатами мануального тестування була зорово помітна відповідь від сервера, а саме додаткова затримка перед відповіддю, яка не характерна стандартній поведінці форми проекту.

Оскільки мануальним тестуванням не було визначено точного часу відповіді сервера на запит, використаємо комплексне ПЗ AWASP ZAP для визначення часу відповіді.

За рис. 3.8 можна побачити, що максимальний час відповіді сервера 417 мс, що може казати про можливість виконання SQL-ін’єкції, завдяки якій можлива крадіжка конфіденційної інформації з БД проекту.



ID	Req. Отметка времени	Resp. Отметка времени	Метод	URL-адрес	Код	Причина	RTT	Размер Resp. Заголовок	Размер Resp. Тело
64 156	27.04.2023, 19:45:58	27.04.2023, 19:45:58	GET	https://playzone.in.ua/login?username=%3Cxsl%3Avalue-	200	OK	417 мс	802 байт	12 657 байт
62 851	27.04.2023, 19:45:14	27.04.2023, 19:45:14	GET	https://playzone.in.ua/login?username=Chocolate&passw	200	OK	282 мс	798 байт	12 658 байт
55 166	27.04.2023, 19:40:39	27.04.2023, 19:40:39	GET	https://playzone.in.ua/login?username=Chocolate&passw	200	OK	276 мс	812 байт	12 658 байт
55 773	27.04.2023, 19:41:01	27.04.2023, 19:41:02	GET	https://playzone.in.ua/login?username=case+randomblob	200	OK	275 мс	816 байт	12 658 байт
53 751	27.04.2023, 19:39:49	27.04.2023, 19:39:50	GET	https://playzone.in.ua/login?username=Chocolate&passw	200	OK	269 мс	798 байт	12 658 байт
50 385	27.04.2023, 19:37:47	27.04.2023, 19:37:47	GET	https://playzone.in.ua/login?username=Chocolate&passw	200	OK	256 мс	804 байт	12 658 байт
60 604	27.04.2023, 19:43:53	27.04.2023, 19:43:53	GET	https://playzone.in.ua/login?username=Chocolate&passw	200	OK	251 мс	806 байт	12 658 байт
50 344	27.04.2023, 19:37:46	27.04.2023, 19:37:46	GET	https://playzone.in.ua/login?username=Chocolate&passw	200	OK	249 мс	800 байт	12 658 байт
51 867	27.04.2023, 19:38:40	27.04.2023, 19:38:40	GET	https://playzone.in.ua/login?username=Chocolate&passw	200	OK	249 мс	798 байт	12 658 байт
64 296	27.04.2023, 19:46:02	27.04.2023, 19:46:02	GET	https://playzone.in.ua/login?username=Chocolate&passw	200	OK	247 мс	802 байт	12 658 байт
52 607	27.04.2023, 19:39:07	27.04.2023, 19:39:07	GET	https://playzone.in.ua/login?username=Chocolate&passw	200	OK	242 мс	796 байт	12 658 байт
57 979	27.04.2023, 19:42:19	27.04.2023, 19:42:19	GET	https://playzone.in.ua/login?username=Chocolate&passw	200	OK	242 мс	800 байт	12 658 байт
56 540	27.04.2023, 19:41:30	27.04.2023, 19:41:31	GET	https://playzone.in.ua/login?username=Chocolate%27+%3	200	OK	240 мс	814 байт	12 658 байт
64 208	27.04.2023, 19:45:59	27.04.2023, 19:45:59	GET	https://playzone.in.ua/sftp-config.json	404	Not Found	240 мс	775 байт	12 059 байт

Рис. 3.8. Логування запитів результатів пантестування

Продовжуючи вбудовування SQL-запитів та скриптів мови програмування JavaScript у поля форми авторизації з метою викрадення даних з БД проекту, ІС блокувала дане інформаційне вторгнення шляхом перенаправлення інформаційних запитів на інтегрований проектом плагін CloudFlare, через що блокувала хакерську атаку.

У розмові з власником проекту зійшлися на висновку достатнього рівня захищеності ІС з подальшим доопрацюванням рівня безпеки.

3.2. Навантажувальне тестування

Навантажувальне тестування - це клас комплексного тестування, який визначається як процес перевірки продуктивності системи шляхом створення імітуючих умов реального робочого навантаження.

Навантажувальне тестування дозволяє виявити вразливі місця потужності в системі та визначити необхідність збільшення її потужності або оптимізації роботи коду.

Суть даних тестів полягає в перевірці, чи зможе система витримати заплановане навантаження: дії великої кількості користувачів або операцій за один момент часу.

Як зазначає автор книги «Тестування ПЗ», С. Куліков – тестування швидкості і надійності / навантажувальне тестування (load/stress/performance testing) зазвичай проводять методом чорного ящика і доволі рідко методом білого ящика [16], оскільки перший метод є нижчим за метод сірого ящика, було змодульовано тестування методом чорного ящика.

Тестування на навантаження включає також в себе класи: тестування представлення, реактивне тестування, позитивне тестування, негативне тестування, інвазивне тестування, доменне тестування, компонентне тестування, інтеграційне тестування, автоматизоване тестування, дослідницьке тестування, тестування по тест-кейсам.

Для виконання тестів проекту Playzone, було змодульовано сценарій-імітацію реального використання ІС взаємодії з віртуальними користувачами і були використані такі методики тестування:

Тестування навантаження (load testing), яке включає визначення максимальної кількості запитів, які ІС зможе обробити за нормальної поведінки.

Стресове тестування (stress testing) - це нефункціональна методологія, яка включає визначення поведінки ІС за виконання більшого навантаження, ніж у тестуванні навантаження.

Тестування на масштабованість (scalability testing), яке включає визначення поведінки ІС маніпулюючи масштабованістю кількості запитів до ІС.

Суть тестування навантаження полягає в тому, щоб у проведені позитивних тестів, встановити максимально можливе навантаження на ПЗ, яке ІС зможе обробити, визнаючи базову продуктивність ІС.

Суть стрес-тестування полягає в тому, щоб при проведені негативних тестів до ПЗ на критичній межі її можливостей та ресурсів, визначити її поведінку в екстремальних умовах і можливих ризиків.

Суть тестування на масштабованість полягає в тому, щоб визначити змогу ІС масштабуватися під час збільшення кількості користувачів або обчислень даних, збільшуючи показники продуктивності відповідно до збільшення кількості доступних системі ресурсів.

Для всіх використаних методик навантажувального тестування, застосовувалось комплексне ПЗ JMeter з основними семплерами, а також додатково інтегрований JSR223 Sampler з використанням мови програмування JavaScript для подальшого можливого аналізу виконання тестів.

Загальний алгоритм виконаних дій виглядає таким чином:

1) Співбесіда з власником проекту:
    a. Визначення області тестування проекту;
    b. Визначення базової теоретичної потужності сервера проекту;
    c. Створення тест-кейсів для проведення тестування.
2) Налаштування комплексного ПЗ JMeter:
    a. Налаштування JSR223 Sampler;
    b. Запуск тестування.
3) Аналіз результатів тестування:
    a. Створення звітів тестування;
    b. Співбесіда з власником проекту.
4) Впровадження рішення багів і оптимізація коду;
5) Повторне тестування;
6) Аналіз результатів тестування;
7) Співбесіда з власником.

Після розмови з власником проекту Playzone, було прийнято рішення проводити навантажувальне тестування лише для головної сторінки, оскільки аналітика проекту показує найчастіше використання лише головної сторінки. Варто зазначити, що головна сторінка сайту проекту є найоптимізованішою і розрахована теоретично для використання 50 користувачів в один момент часу. У таблицях 3.1-3.3 наведені тест-кейси, створені під час співбесіди із власником проекту, опираючись на його вимоги.

Оскільки власником проекту було зазначено кількість користувачів, які можуть використовувати головну сторінку в один момент часу у розмірі 50, що еквівалентно ≈ 500 запитів у хвилину, було погоджено з власником виконання тестів на валідацію базової теоретичної потужності з перевіркою від 4% до 1000%

від його значення. Всі тести проводилися у синхронному режимі, очікуючи відповідь від сервера.

Табл. 3.1

Тест-кейси для тестування навантаження

Ідентифікатор тест-кейсу	Опис тест-кейсу	Очікуваний результат
TC-01	Запуск тесту на 20 користувачів протягом 1 хвилини	Усі 20 користувачів повинні успішно завантажити головну сторінку сайту
TC-02	Запуск тесту на 500 користувачів протягом 1 хвилини	Усі 500 користувачів повинні успішно завантажити головну сторінку сайту
TC-03	Запуск тесту на 1000 користувачів протягом 1 хвилини	Усі 1000 користувачів повинні успішно завантажити головну сторінку сайту
TC-04	Запуск тесту на 2000 користувачів протягом 1 хвилини	Усі 2000 користувачів повинні успішно завантажити головну сторінку сайту

Завершення Табл. 3.1

ТС-05	Запуск тесту на 5000 користувачів протягом 1,5 хвилин	Усі 5000 користувачів повинні успішно завантажити головну сторінку сайту

Всі тести навантаження ТС-01 - ТС-05 пройшли успішно. Графік на рис. 3.9 показує загальну кількість запитів тестів ТС-01 - ТС-05. У 8520 запитів з 8520 була отримана відповідь від сервера.

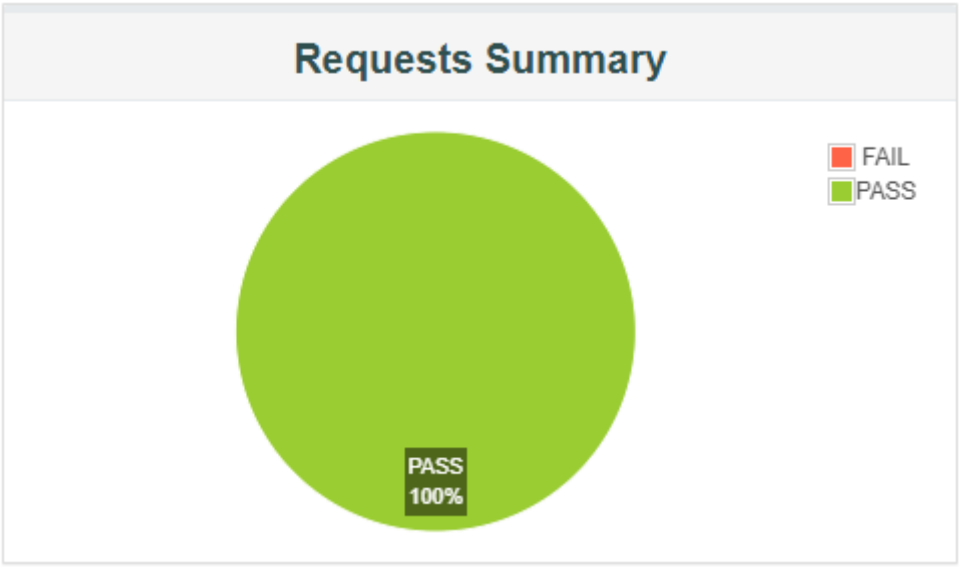
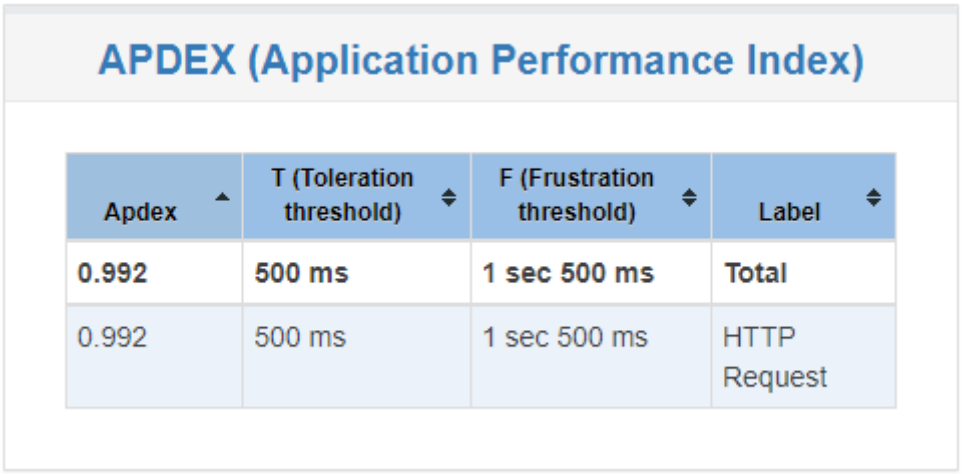


Рис. 3.9. Загальний графік виконання навантажувальних тестів ТС-01- ТС-05

Таблиця на рис. 3.10 зображує середнє значення індексу продуктивності серед 8520 запитів IC у розмірі 99,2% з порогом толерантності у 500мс і межею відгуку у 1,5 секунди за http-запитами.

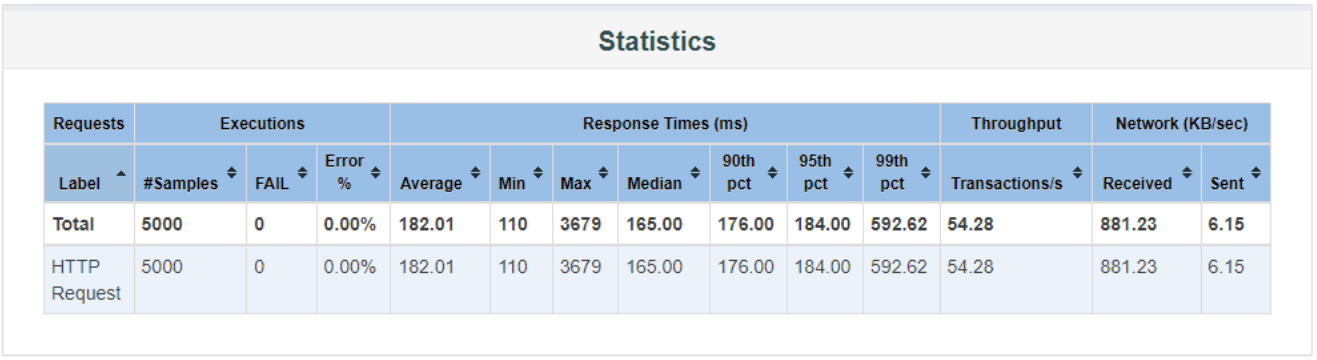


APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.992	**500 ms**	**1 sec 500 ms**	**Total**
0.992	500 ms	1 sec 500 ms	HTTP Request

Рис. 3.10. Загальна таблиця індексу продуктивності ІС навантажувальних тестів ТС-01- ТС-05

Таблиця на рис. 3.11 відображає статистику навантажувального тесту ТС-05. Тест ТС-05 виконав 5000 http-запитів на сервер проекту і отримав 100% відповідей. Час відгуку ТС-05 становив від 110 мс до 3,7 секунд. За метриками тесту ТС-05 90% запитів були виконані за час, що <=176мс, 95% <= 184мс, а 99% відповідно <=593мс. Середня кількість запитів за час виконання тесту склала 54,28 запитів за секунду.



Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**5000**	**0**	**0.00%**	**182.01**	**110**	**3679**	**165.00**	**176.00**	**184.00**	**592.62**	**54.28**	**881.23**	**6.15**
HTTP Request	5000	0	0.00%	182.01	110	3679	165.00	176.00	184.00	592.62	54.28	881.23	6.15

Рис. 3.11. Таблиця результатів виконання тесту ТС-05

З графіків, що зображені на рис. 3.12, рис. 3.13 і рис. 3.14 можна побачити, що під час навантажувального тесту ТС-05, 97,12 % (4856 запитів) були відпрацьовані за час від 100 до 200мс, 1% (83 запити) від 200 до 300мс, і ще 1% (61 запит) проведено за час від 300 до 3679 мс.

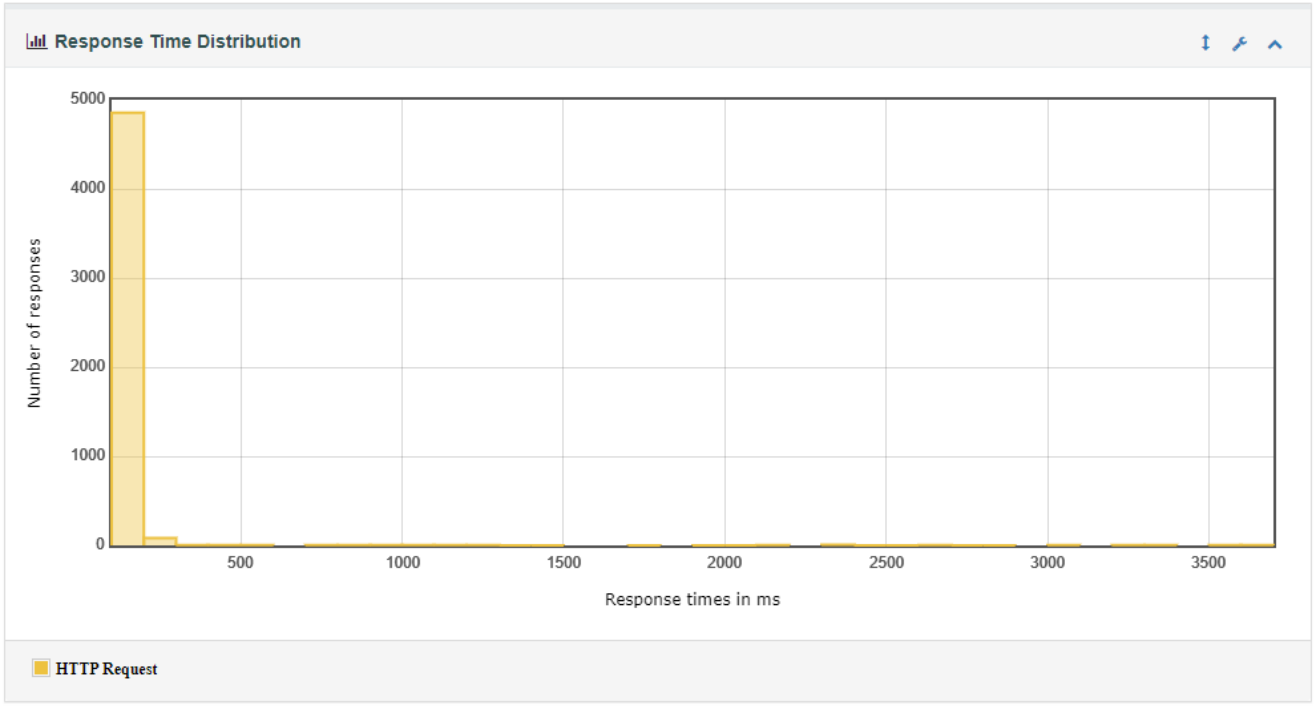


Рис. 3.12. Графік розподілу часу відповіді виконання навантажувального тесту ТС-05

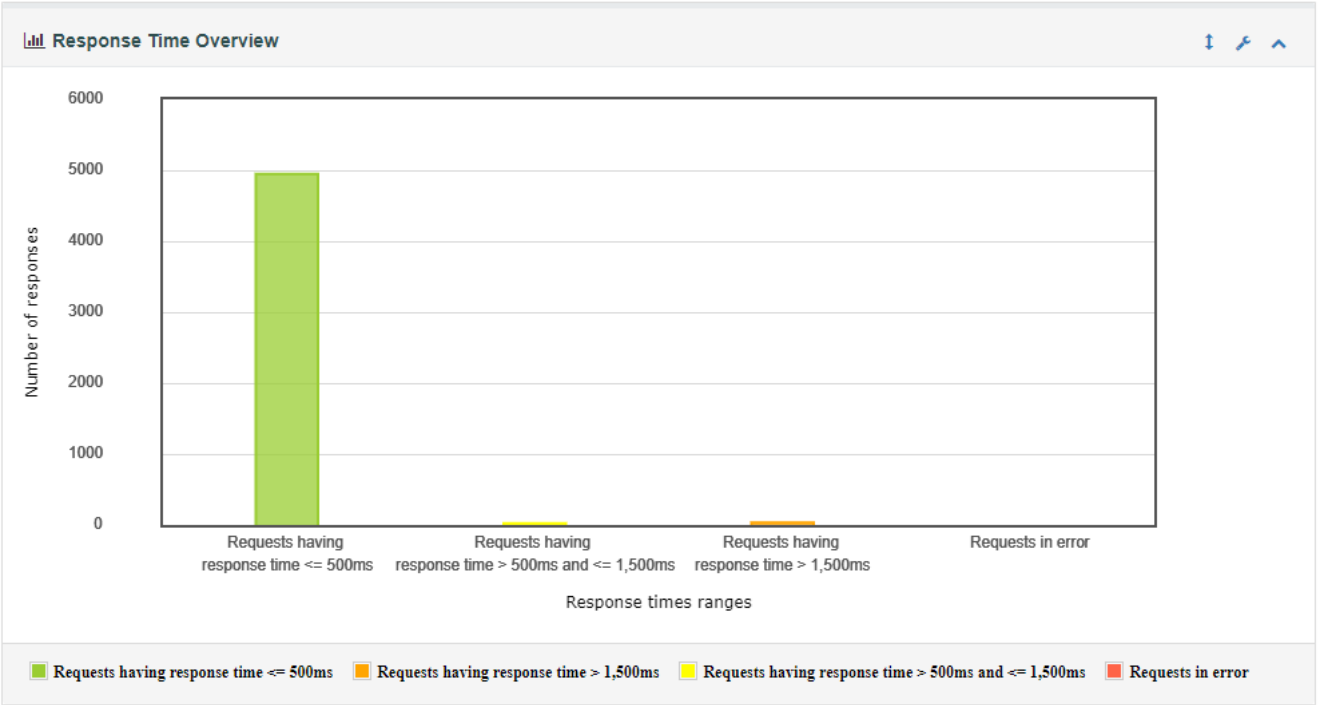


Рис. 3.13. Графік часу відгуку на http-запити навантажувального тесту ТС-05

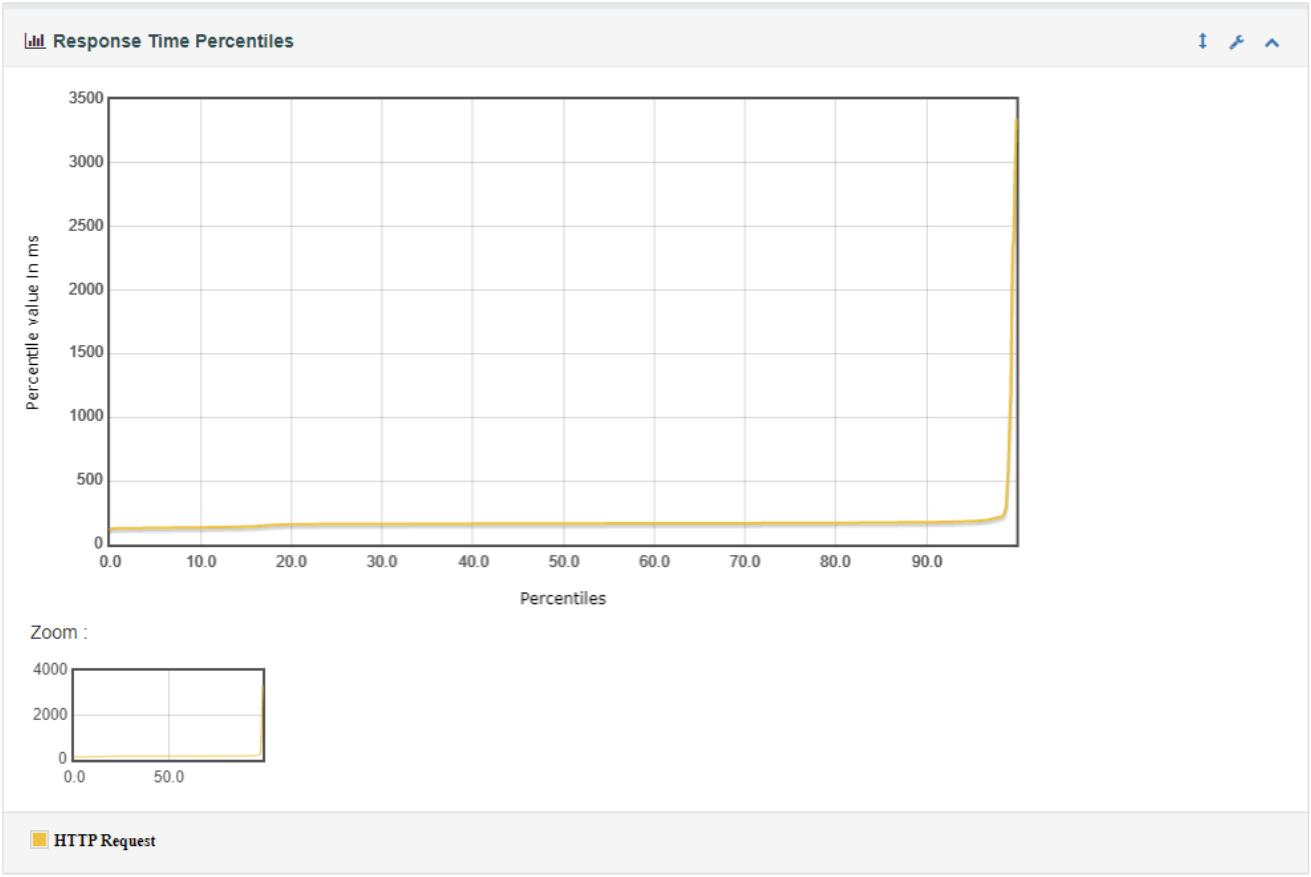


Рис. 3.14. Графік процентилю часу відповіді навантажувального тесту ТС-05

Під час виконання стресового тестування, тести ТС-06 - ТС-07 повністю пройшли успішно. На рис. 3.15 зображено загальний графік, який відсотково зображує виконання тестів на 100%. 6000 запитів з 6000 відправили http-запити і отримали відповідь.

Табл. 3.2

Тест-кейси для стресового тестування

Ідентифікатор тест-кейсу	Опис тест-кейсу	Очікуваний результат
ТС-06	Запуск тесту на 1000 користувачів протягом 1 хвилин із поступовим збільшенням навантаження	При досягненні 800 користувачів має бути уповільнення швидкості завантаження сторінки, при 500 користувачах повинні бути помилки завантаження
ТС-07	Запуск 2-х екземплярів тесту на 5000 користувачів протягом 2 хвилин із поступовим збільшенням навантаження	При досягненні 4000 користувачів має бути уповільнення швидкості завантаження сторінки, при 1000 користувачах повинні бути помилки завантаження
ТС-08	Запуск тесту на 10000 користувачів протягом 2 хвилин із поступовим збільшенням навантаження	При досягненні 8000 користувачів має бути уповільнення швидкості завантаження сторінки, при 9000 користувачах повинні спостерігатися помилки завантаження

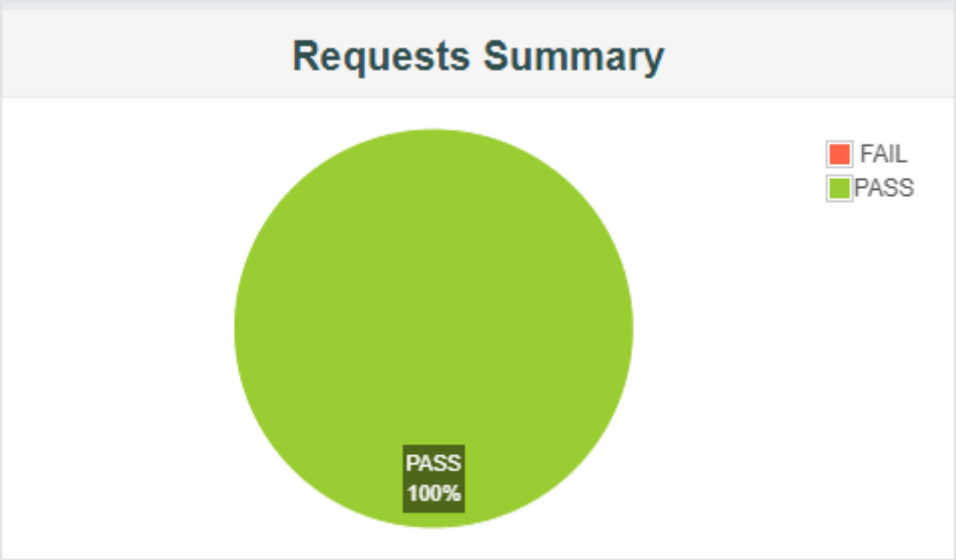


Рис. 3.15. Загальний графік виконання стресових тестів ТС-06- ТС-07

Таблиця на рис. 3.16 зображує середнє значення індексу продуктивності тестів ТС-06- ТС-07 серед 6000 запитів до ІС у розмірі 99,9% з порогом толерантності у 500мс і межею відгуку у 1,5 секунди за http-запитами.

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.999	**500 ms**	**1 sec 500 ms**	**Total**
0.999	500 ms	1 sec 500 ms	HTTP Request

Рис. 3.16. Загальна таблиця індексу продуктивності ІС стресових тестів ТС-06-ТС-07

Таблиця на рис. 3.17 відображає статистику стресового тесту ТС-06. Тест ТС-06 виконав 1000 http-запитів на сервер проекту і отримав 100% відповідей. Час відгуку ТС-06 становив від 114 мс до 655 мс. За метриками тесту ТС-06 90% запитів

були виконані за час, що <=179мс, 95% <= 198мс, а 99% відповідно <=376мс. Середня кількість запитів за час виконання тесту склала 16,67 запитів за секунду.

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**1000**	**0**	**0.00%**	**172.54**	**114**	**655**	**167.00**	**179.00**	**197.85**	**376.70**	**16.67**	**270.62**	**1.89**
HTTP Request	1000	0	0.00%	172.54	114	655	167.00	179.00	197.85	376.70	16.67	270.62	1.89

Рис. 3.17. Таблиця результатів виконання тесту ТС-06

Таблиця на рис. 3.18 відображає статистику стресового тесту ТС-07. Тест ТС-07 виконав 10000 http-запитів на сервер проекту і отримав 100% відповідей. Час відгуку ТС-07 становив від 77 мс до 1023 мс. За метриками тесту ТС-07 90% запитів були виконані за час, що <=174мс, 95% <= 181мс, а 99% відповідно <=220мс. Середня кількість запитів за час виконання тесту склала 82,55 запитів за секунду.

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**10000**	**0**	**0.00%**	**145.58**	**77**	**1023**	**139.00**	**174.00**	**181.00**	**219.99**	**82.55**	**1340.05**	**9.35**
HTTP Request	10000	0	0.00%	145.58	77	1023	139.00	174.00	181.00	219.99	82.55	1340.05	9.35

Рис. 3.18. Таблиця результатів виконання тесту ТС-07

З графіків, що зображені на рис. 3.19, рис. 3.21 і рис. 3.23 можна побачити, що під час стресового тесту ТС-06, 95,2 % (952 запитів) були відпрацьовані за час від 100 до 200мс, 3,5% (35 запитів) від 200 до 300мс, і ще 1,3% (13 запитів) проведено за час від 300 до 655 мс.

З графіків, що зображені на рис. 3.20, рис. 3.22 і рис. 3.24 можна побачити, що під час стресового тесту ТС-07, 5 % (502 запити) були відпрацьовані за час від 0 до 100мс, 93,37% (9337 запитів) від 100 до 200мс, 1,1% (116 запитів) від 200 до 300мс і ще 0,4% (45 запитів) проведено за час від 300 до 1023 мс.

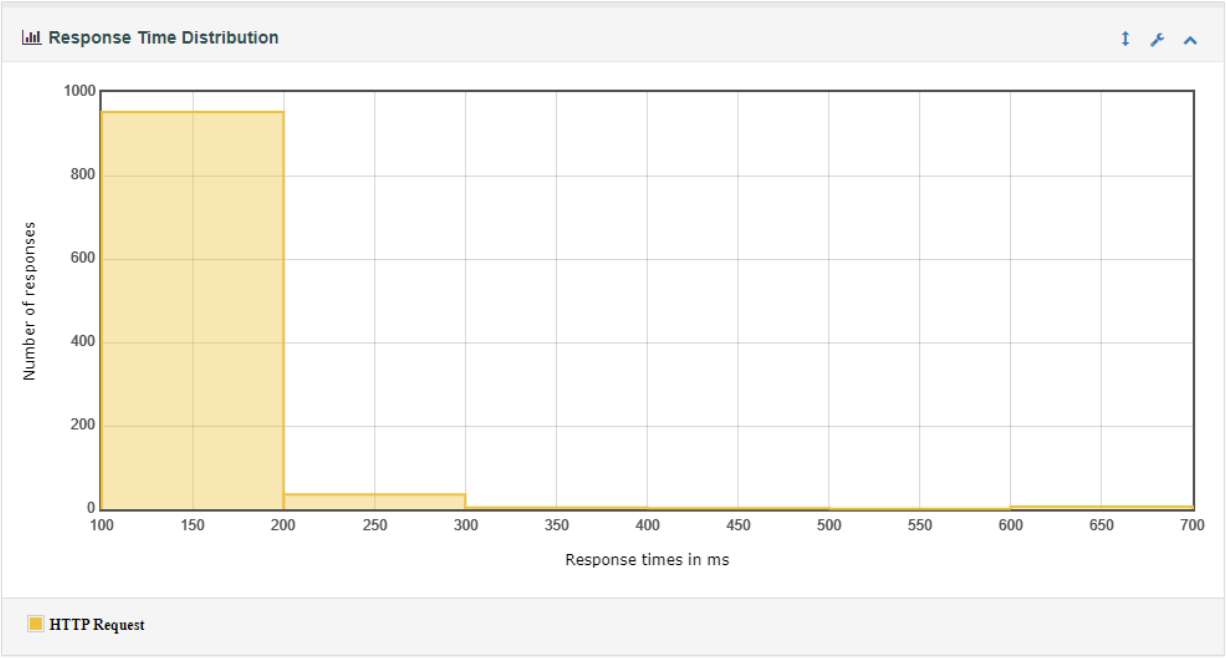


Рис. 3.19. Графік розподілу часу відповіді виконання стресових тесту ТС-06

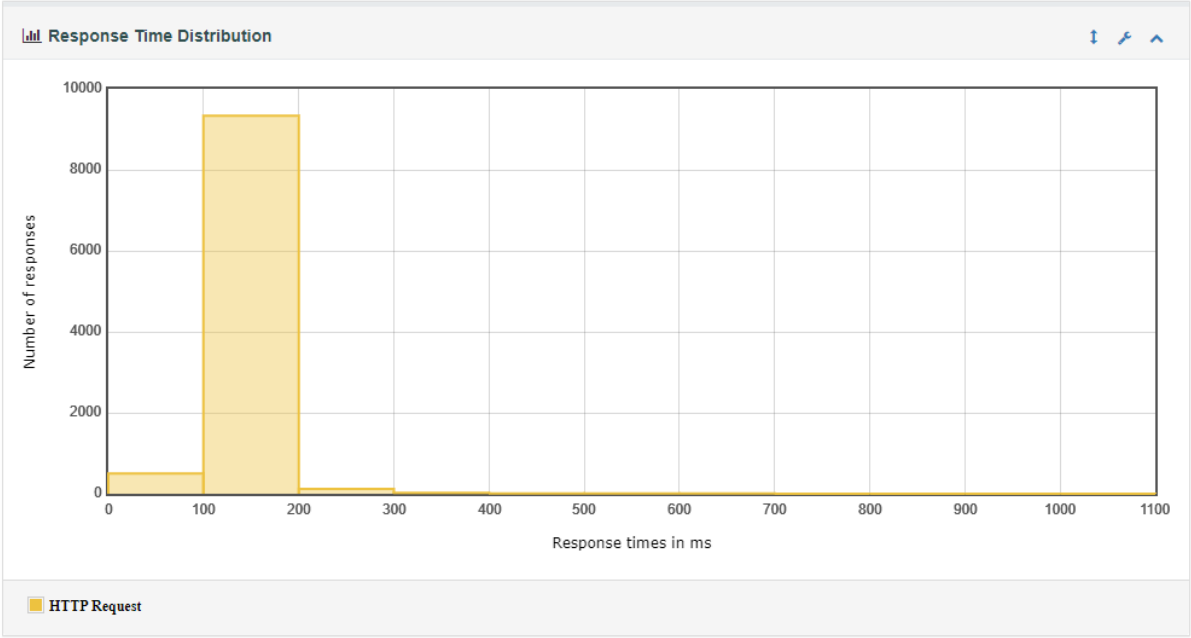


Рис. 3.20. Графік розподілу часу відповіді виконання стресового тесту ТС-07

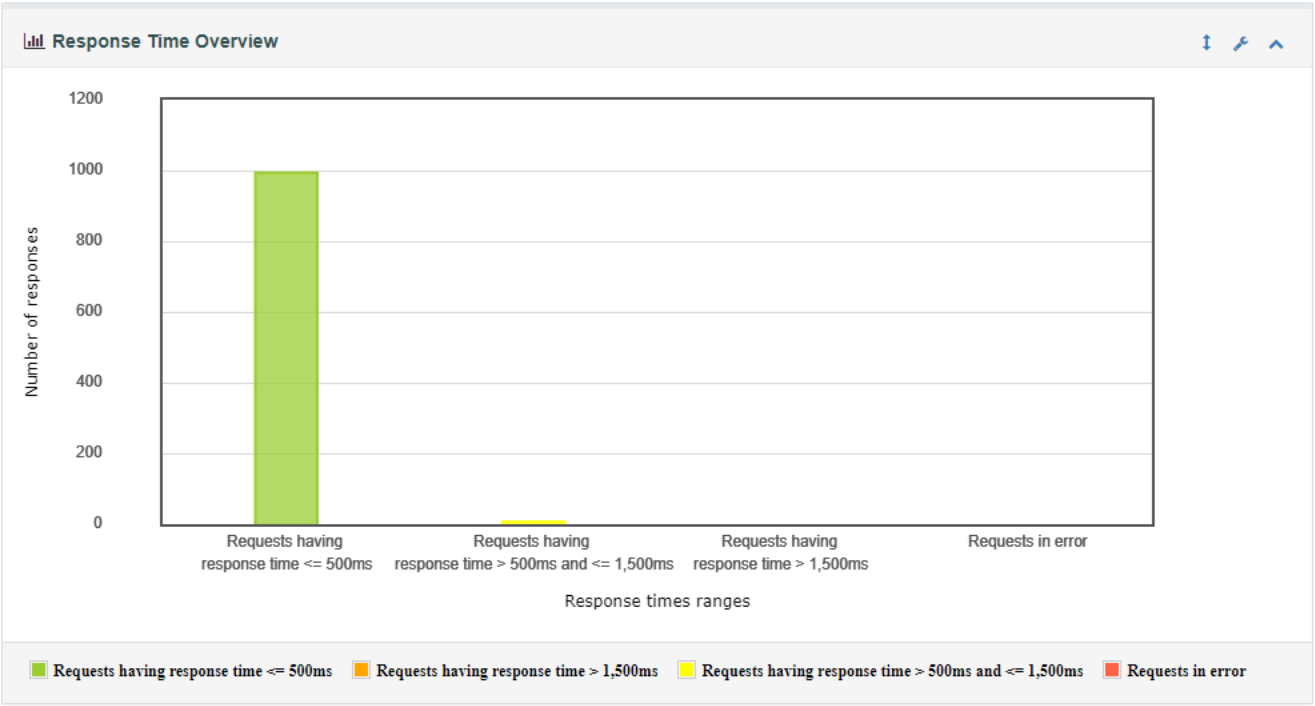


Рис. 3.21. Графік часу відгуку на http-запити стресового тесту ТС-06

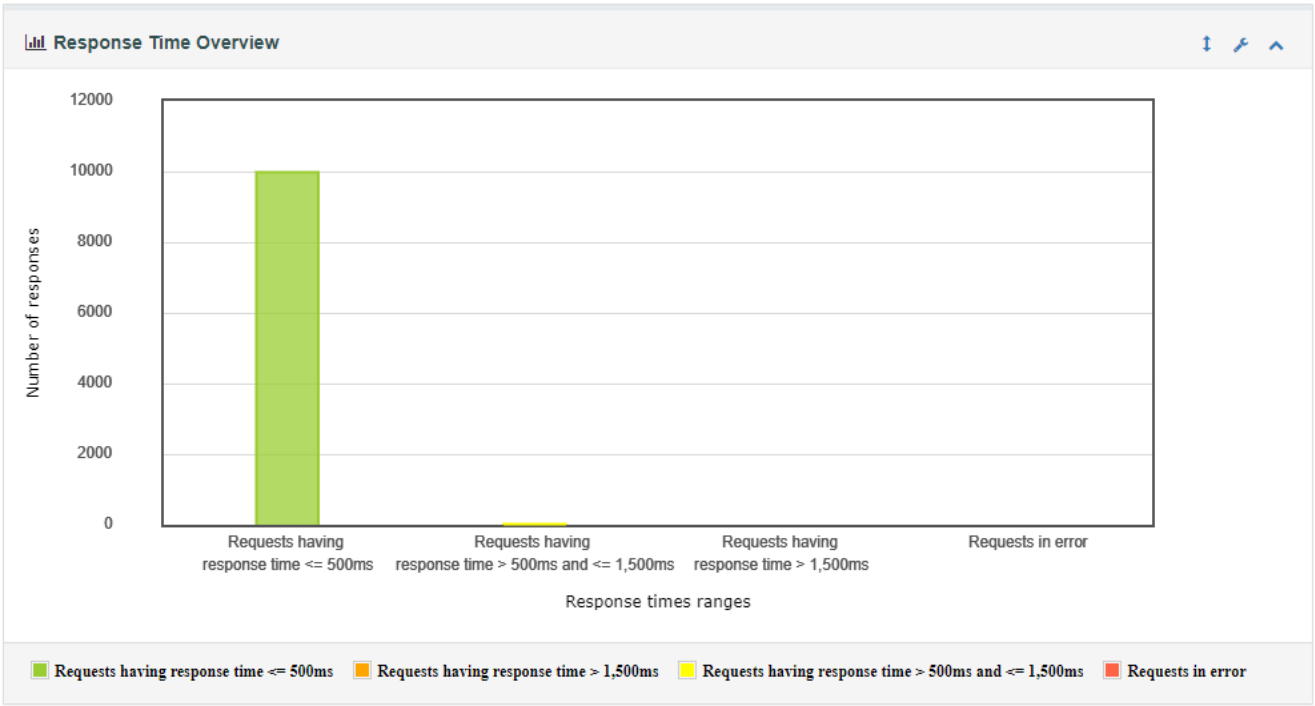


Рис. 3.22. Графік часу відгуку на http-запити стресового тесту ТС-07

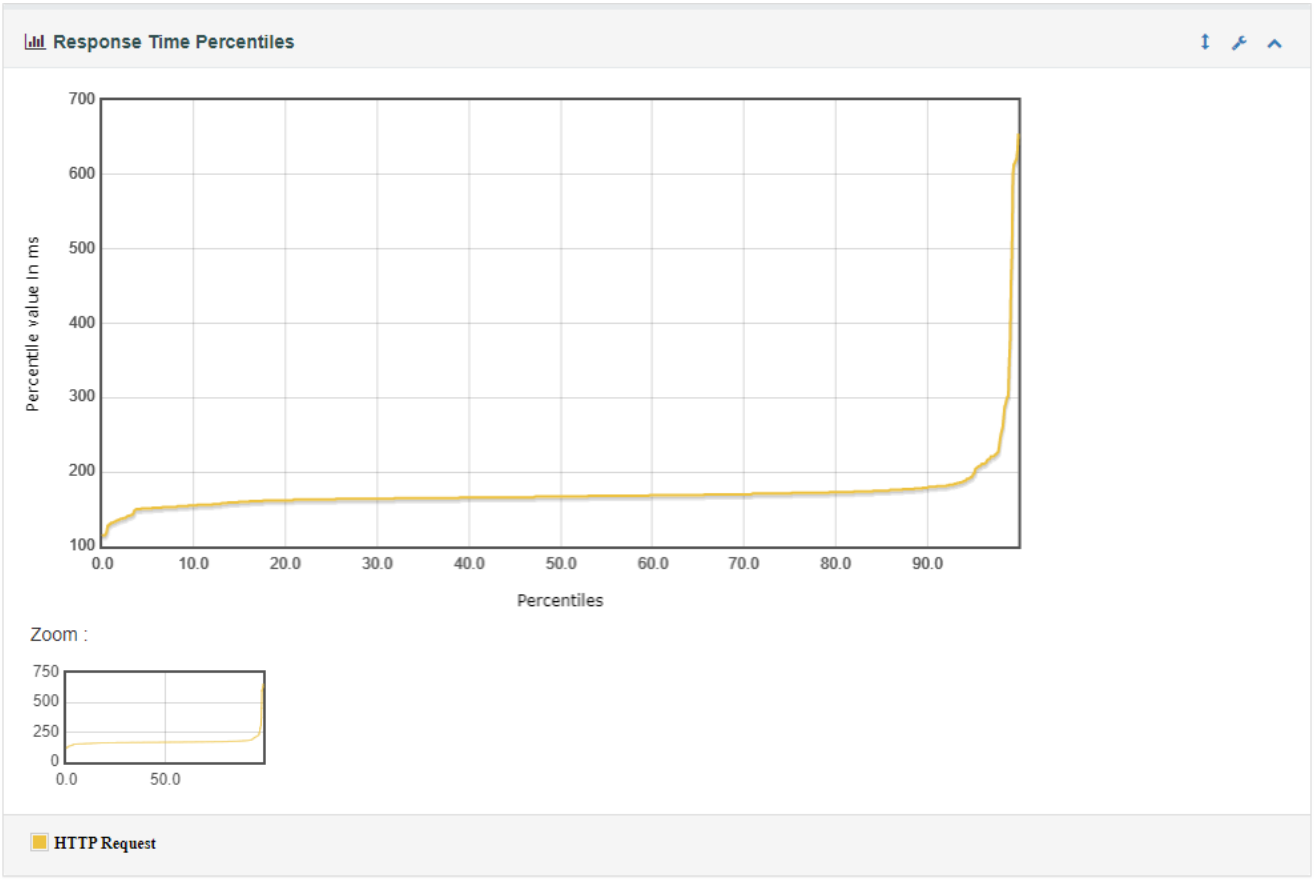


Рис. 3.23. Графік процентилю часу відповіді навантажувального тесту ТС-06

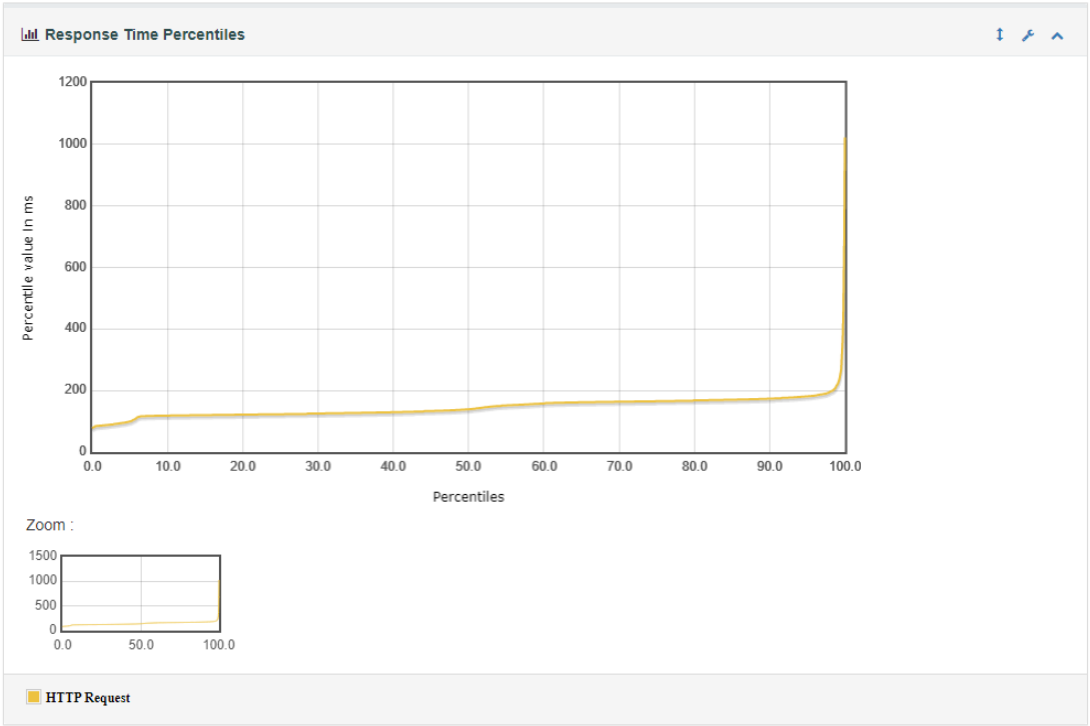


Рис. 3.24. Графік процентилю часу відповіді навантажувального тесту ТС-07

Під час виконання стресового тестування, тест ТС-08 пройшов успішно на 97,2%. На рис. 3.25 зображено графік, який відсотково зображує виконання тестів. 19439 запитів з 20000 відправили http-запити і отримали відповідь.

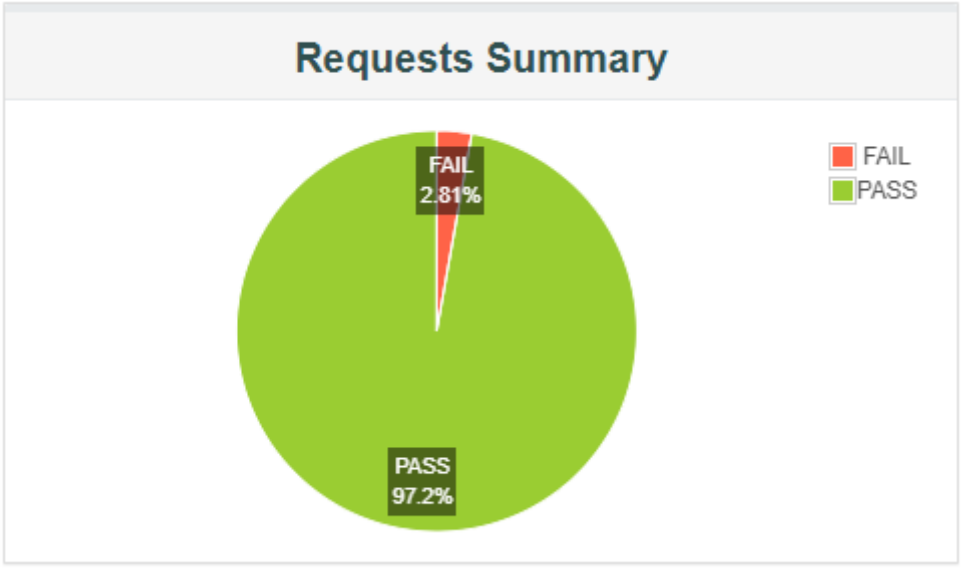


Рис. 3.25. Загальний графік виконання стресового тесту ТС-08

Таблиця на рис. 3.26 зображує середнє значення індексу продуктивності тесту ТС-08 серед 20000 запитів до ІС у розмірі 37,4% з порогом толерантності у 500мс і межею відгуку у 1,5 секунди за http-запитами.

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.374	**500 ms**	**1 sec 500 ms**	**Total**
0.374	500 ms	1 sec 500 ms	HTTP Request

Рис. 3.26. Загальна таблиця індексу продуктивності ІС стресового тесту ТС-08

Таблиця на рис. 3.27 відображає статистику стресового тесту ТС-08. Тест ТС-08 виконав 20000 http-запитів на сервер проекту і отримав 97,2% відповідей. Час відгуку ТС-08 становив від 61 мс до 1,68 секунди. За метриками тесту ТС-08 90% запитів були виконані за час, що <=3,3 с, 95% <= 3,44 с, а 99% відповідно <= 6 с. Середня кількість запитів за час виконання тесту склала 152,83 запити за секунду.

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**20000**	**561**	**2.81%**	**1628.37**	**61**	**16857**	**1404.50**	**3308.00**	**3439.00**	**6036.39**	**152.83**	**2186.08**	**22.46**
HTTP Request	20000	561	2.81%	1628.37	61	16857	1404.50	3308.00	3439.00	6036.39	152.83	2186.08	22.46

Рис. 3.27. Таблиця результатів виконання тесту ТС-08

З графіків, що зображені на рис. 3.28, рис. 3.29 і рис. 3.30 можна побачити, що під час стресового тесту ТС-08, 25,24% (5048 запитів) були відпрацьовані за час, менший 500мс, 24,36% (4872 запитів) від 500 мс до 1,5 с, 47,6% (9519 запитів) >1,5 с, і ще 2,8% (561 запитів) не отримали відповідь.

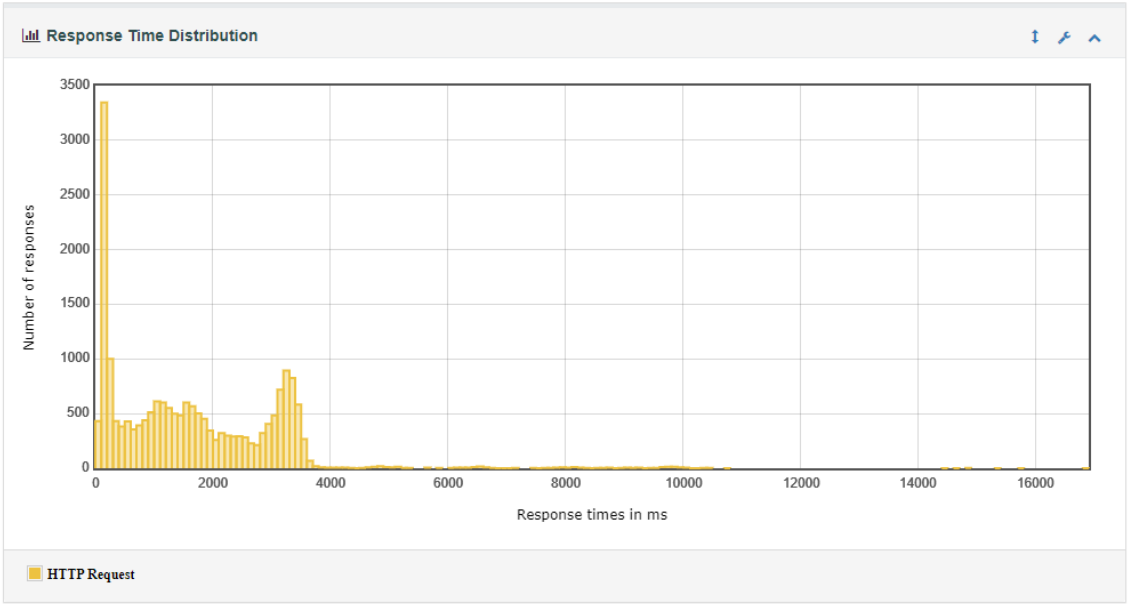


Рис.3.28. Графік розподілу часу відповіді виконання стресового тесту ТС-08

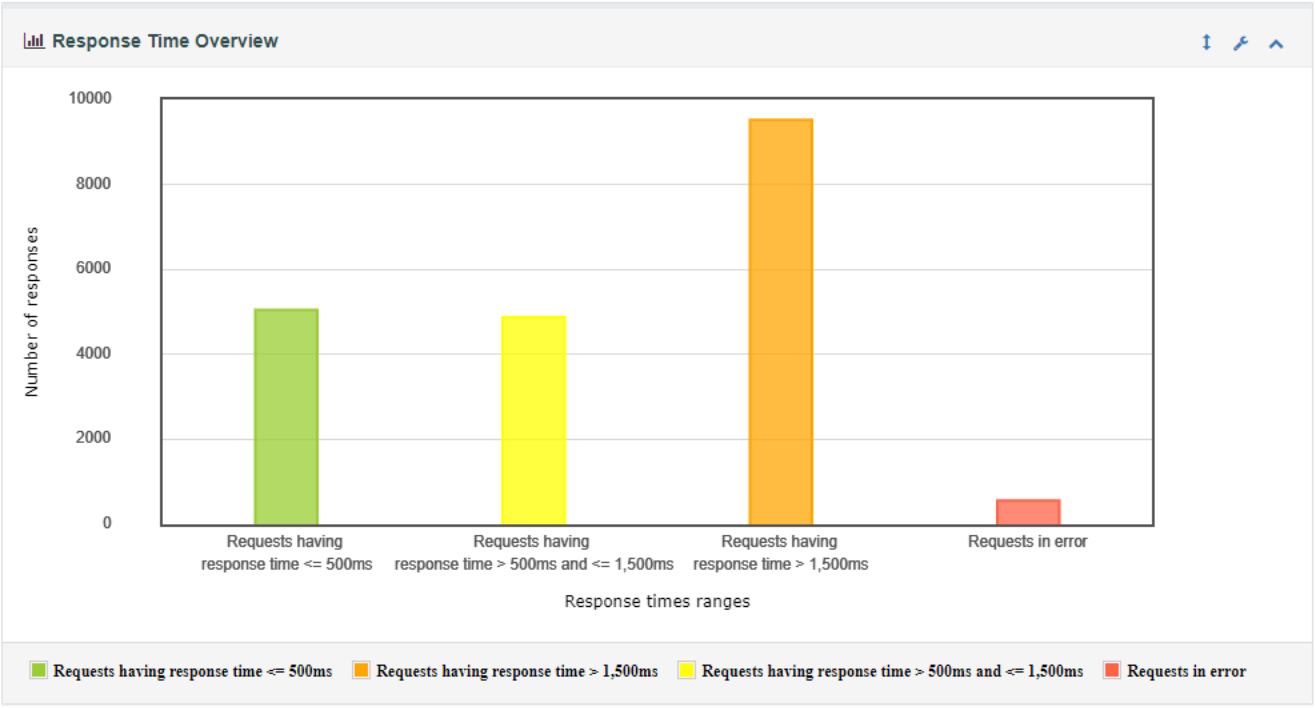


Рис. 3.29. Графік часу відгуку на http-запити стресового тесту ТС-08

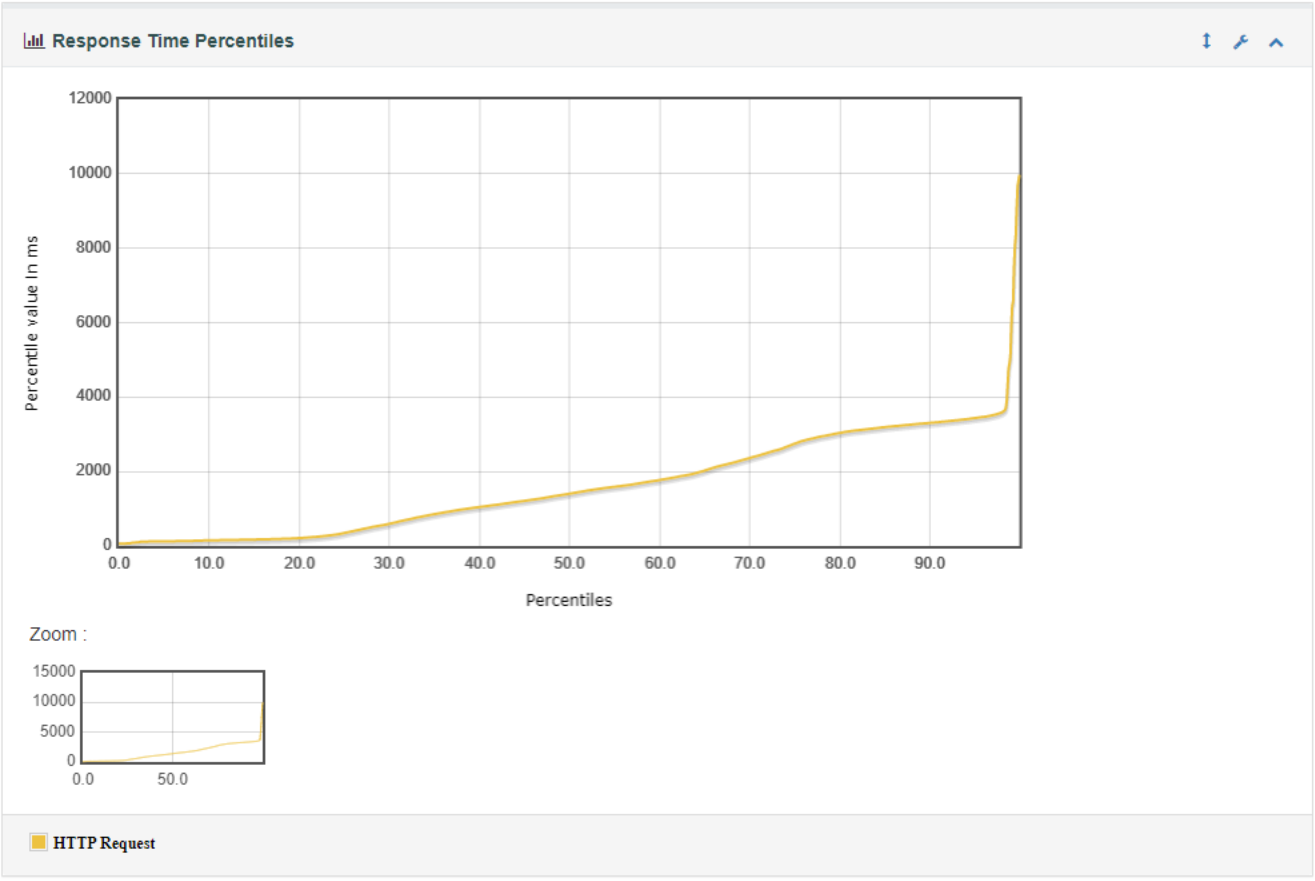


Рис. 3.30. Графік процентилю часу відповіді стресового тесту ТС-08

Табл. 3.3

Тест-кейси для тестування на масштабованість

Ідентифікатор тест-кейсу	Опис тест-кейсу	Очікуваний результат
ТС-09	Перевірка масштабування при додаванні навантаження протягом 1 хвилину з кроком 300 користувачів за 30 секунд	- Успішне виконання тесту зі збільшенням кількості користувачів - Стабільний час відповіді сервера, не більше 3 секунд - Відсутність помилок програми
ТС-10	Перевірка масштабування при додаванні навантаження протягом 3 хвилин із кроком 1000 користувачів за хвилину	- Успішне виконання тесту при збільшенні кількості користувачів - Стабільний час відповіді сервера, не більше 5 секунд - Відсутність помилок програми
ТС-11	Перевірка масштабування при додаванні навантаження протягом 5 хвилин із кроком 3000 користувачів за хвилину	- Успішне виконання тесту при збільшенні кількості користувачів - Стабільний час відповіді сервера, не більше 7 секунд - Відсутність помилок програми

Під час виконання тестування на масштабованість, тести TC-09 - TC-10 повністю пройшли успішно. На рис. 3.31 зображено загальний графік, який відсотково зображує виконання тестів на 100%. 5330 запитів з 5330 відправили http-запити і отримали відповідь.

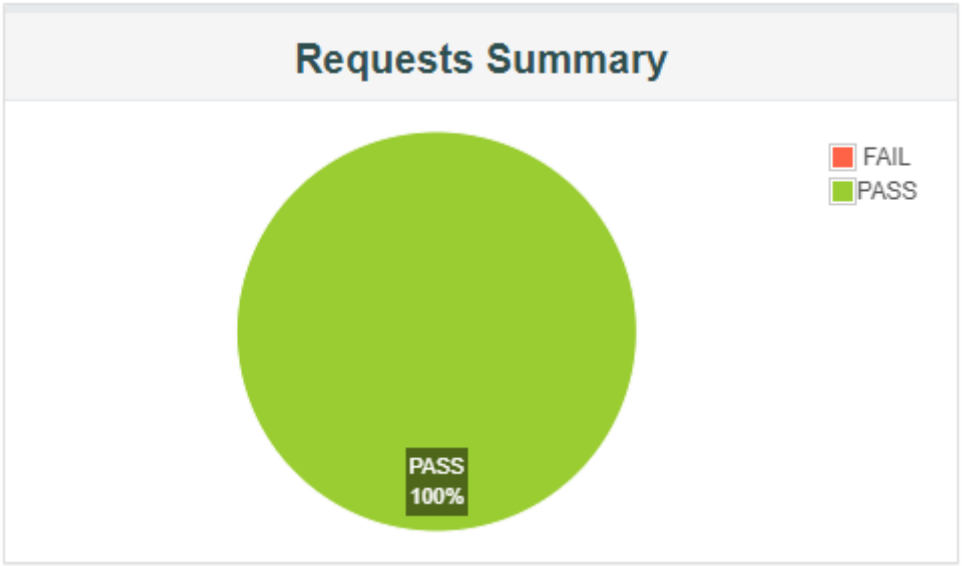


Рис. 3.31. Загальний графік виконання тестів на масштабованість TC-09 - TC-10

Таблиця на рис. 3.32 зображує середнє значення індексу продуктивності тестів TC-09 і TC-10 серед 5300 запитів до ІС у розмірі 99,4% з порогом толерантності у 500мс і межею відгуку у 1,5 секунди за http-запитами.

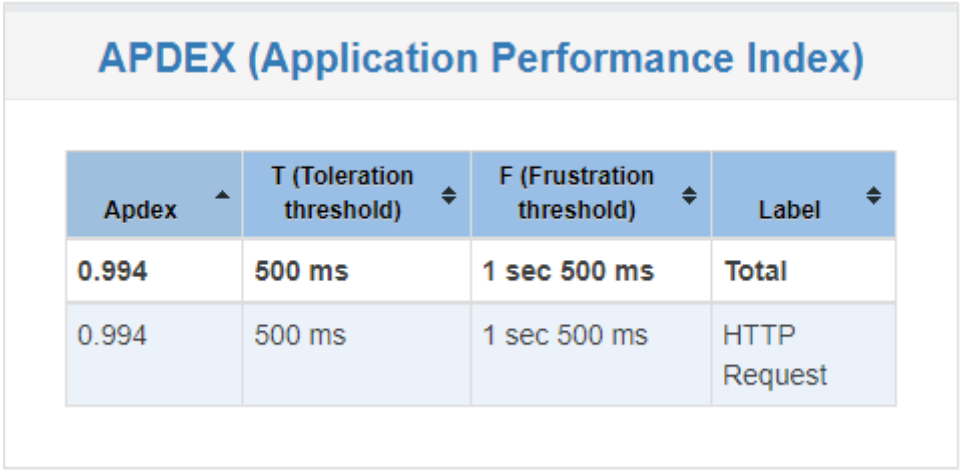


Рис. 3.32. Загальна таблиця індексу продуктивності ІС тесту на масштабованість TC-09 - TC-10

Таблиця на рис. 3.33 відображає статистику тесту на масштабованість ТС-09. Тест ТС-09 виконав 3300 http-запитів на сервер проекту і отримав 100% відповідей. Час відгуку ТС-09 становив від 101 мс до 1,83 секунди. За метриками тесту ТС-09 90% запитів були виконані за час, що <=168мс, 95% <= 188мс, а 99% відповідно <= 961мс. Середня кількість запитів за час виконання тесту склала 54,01 запити за секунду.

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**3300**	**0**	**0.00%**	**167.02**	**101**	**1831**	**151.00**	**168.00**	**188.00**	**960.90**	**54.01**	**876.74**	**10.00**
HTTP Request	3300	0	0.00%	167.02	101	1831	151.00	168.00	188.00	960.90	54.01	876.74	10.00

Рис. 3.33. Таблиця результатів виконання тесту ТС-09

Таблиця на рис. 3.34 відображає статистику тесту на масштабованість ТС-10. Тест ТС-10 виконав 5000 http-запитів на сервер проекту і отримав 100% відповідей. Час відгуку ТС-10 становив від 101 мс до 3,945 секунди. За метриками тесту ТС-10 90% запитів були виконані за час, що <= 166мс, 95% <= 178мс, а 99% відповідно <= 452мс. Середня кількість запитів за час виконання тесту склала 27,1 запити за секунду.

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**5000**	**0**	**0.00%**	**164.33**	**101**	**3945**	**152.00**	**166.00**	**178.00**	**451.99**	**27.10**	**439.96**	**5.70**
HTTP Request	5000	0	0.00%	164.33	101	3945	152.00	166.00	178.00	451.99	27.10	439.96	5.70

Рис. 3.34. Таблиця результатів виконання тесту ТС-10

З графіків, що зображені на рис. 3.35, рис. 3.37.1 і рис. 3.39 можна побачити, що під час тестів на масштабованість ТС-09, 95,8% (3162 запитів) були відпрацьовані від 0 мс до 200 мс, 1,9% (64 запити) від 200 мс до 300 мс, і ще 2,2% (74 запити) від 300 мс до 1831мс.

З графіків, що зображені на рис. 3.36, рис. 3.38 і рис. 3.40 можна побачити, що під час тестів на масштабованість ТС-10, 96,72% (4836 запитів) були відпрацьовані від 0 мс до 200 мс, 1,84% (92 запити) від 200 мс до 300 мс, і ще 1,44% (72 запити) від 300 мс до 3945мс.

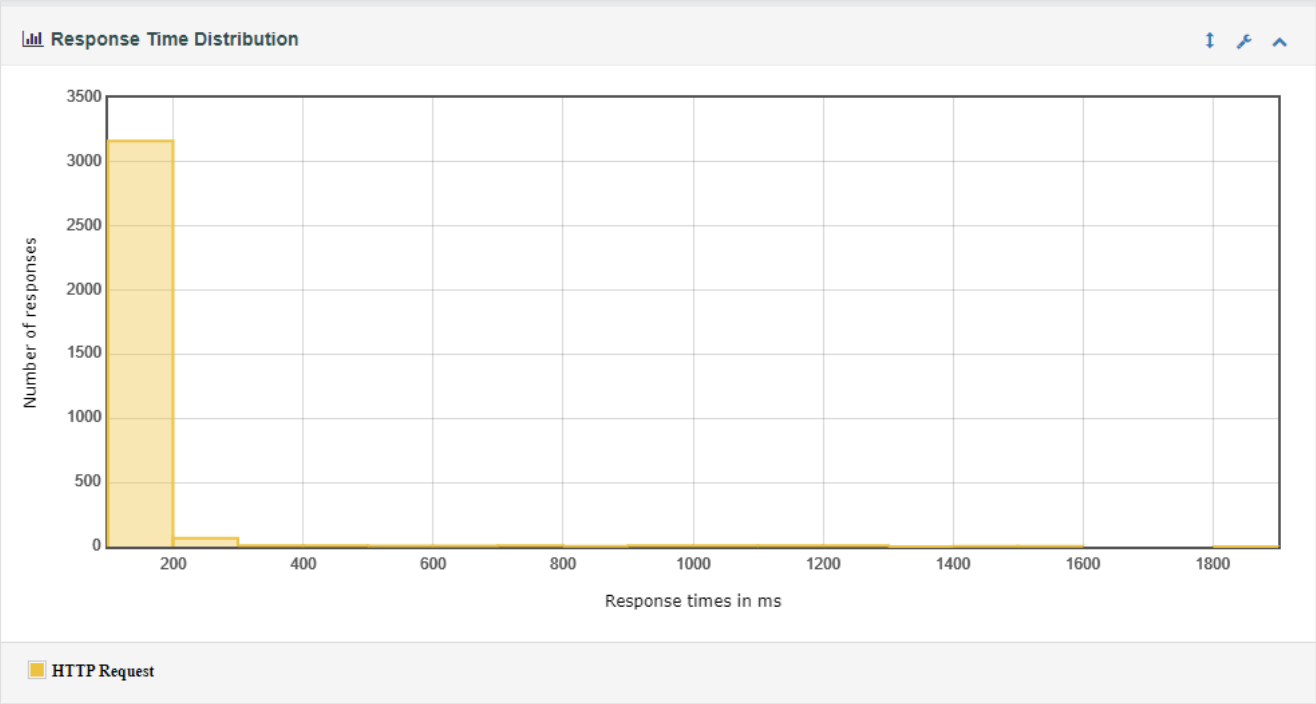


Рис. 3.35. Графік розподілу часу відповіді виконання тесту на масштабованість ТС-09

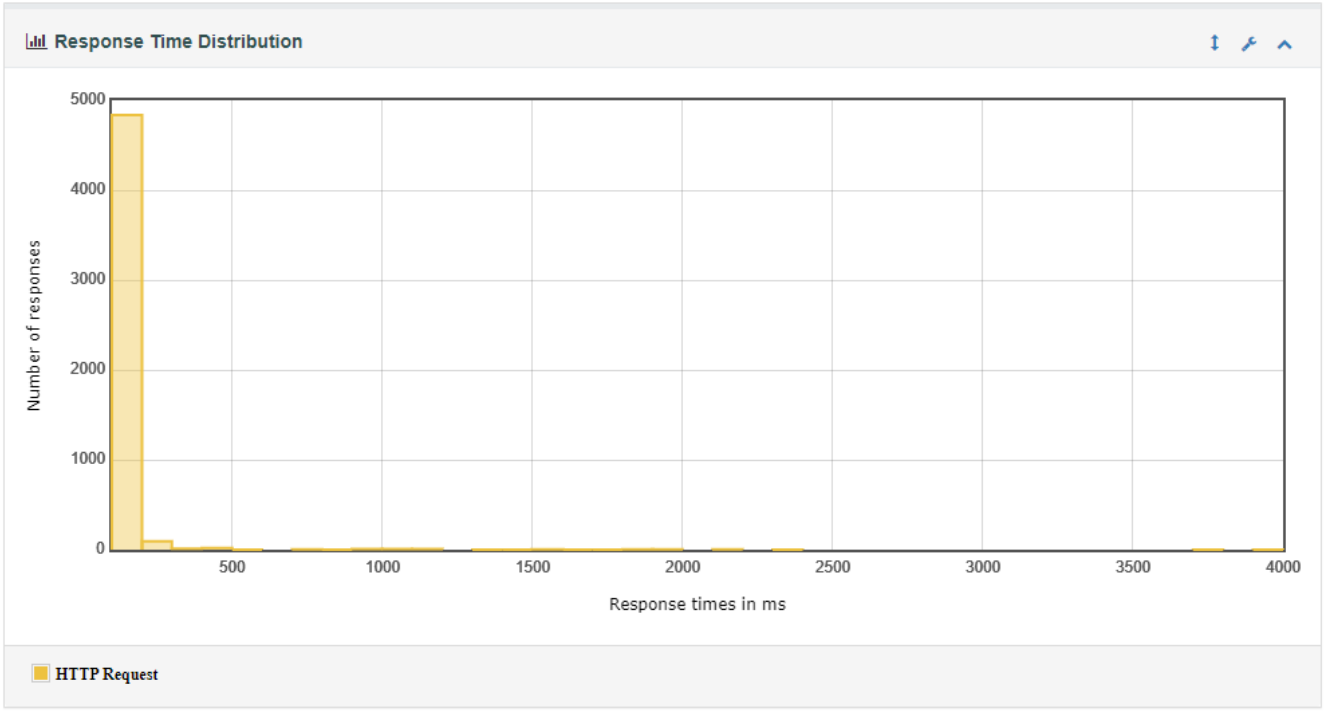


Рис. 3.36. Графік розподілу часу відповіді виконання тесту на масштабованість ТС-10

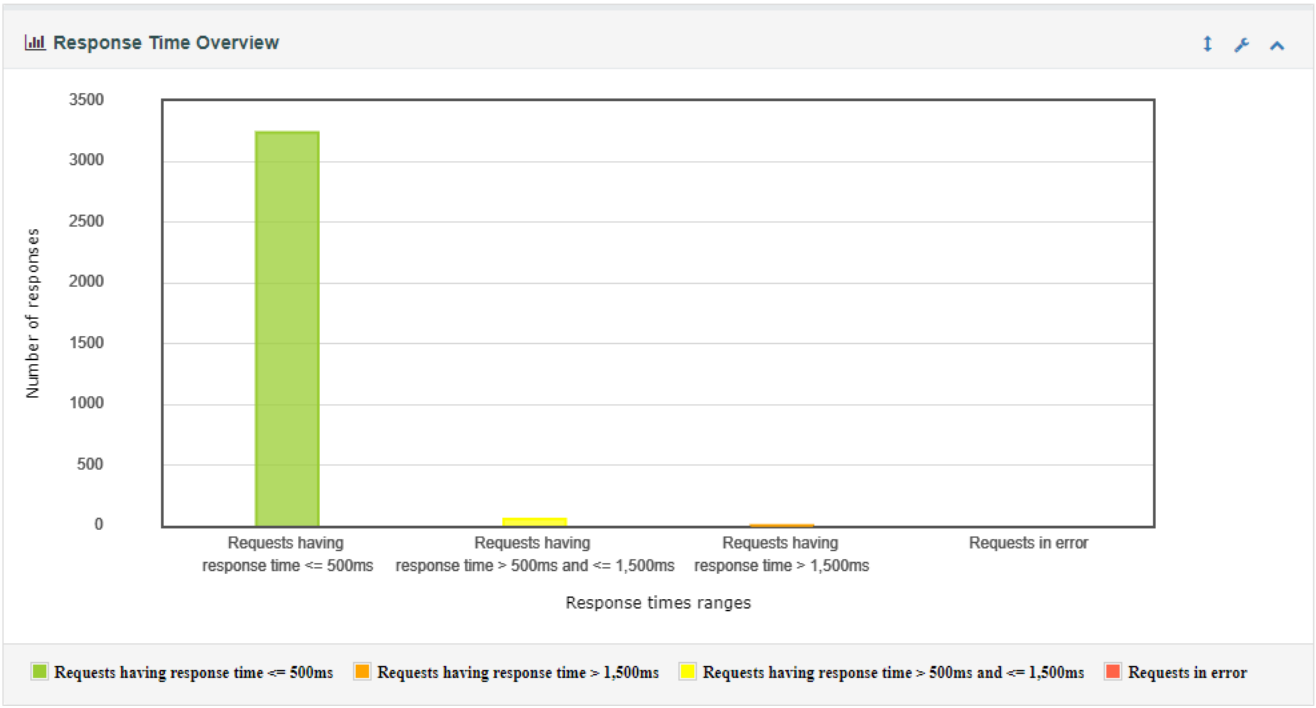


Рис. 3.37. Графік часу відгуку на http-запити тесту на масштабованість ТС-09

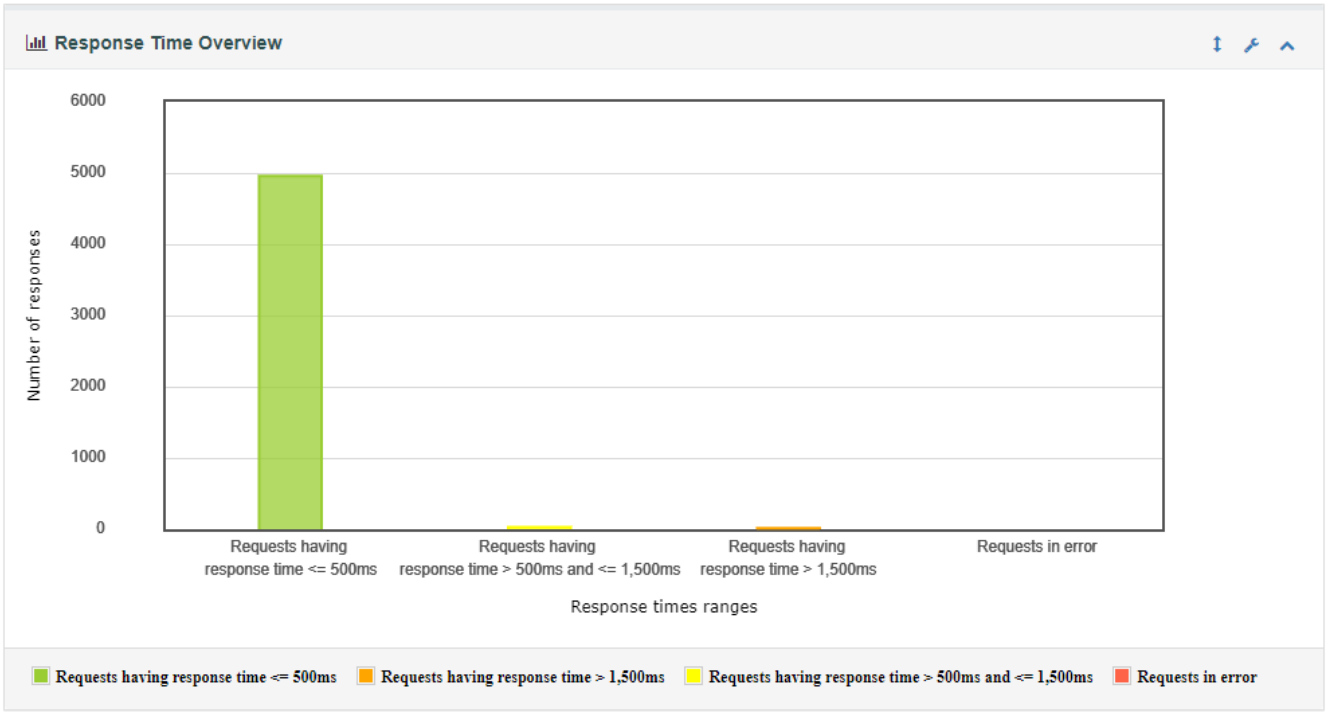


Рис. 3.38. Графік часу відгуку на http-запити тесту на масштабованість ТС-10

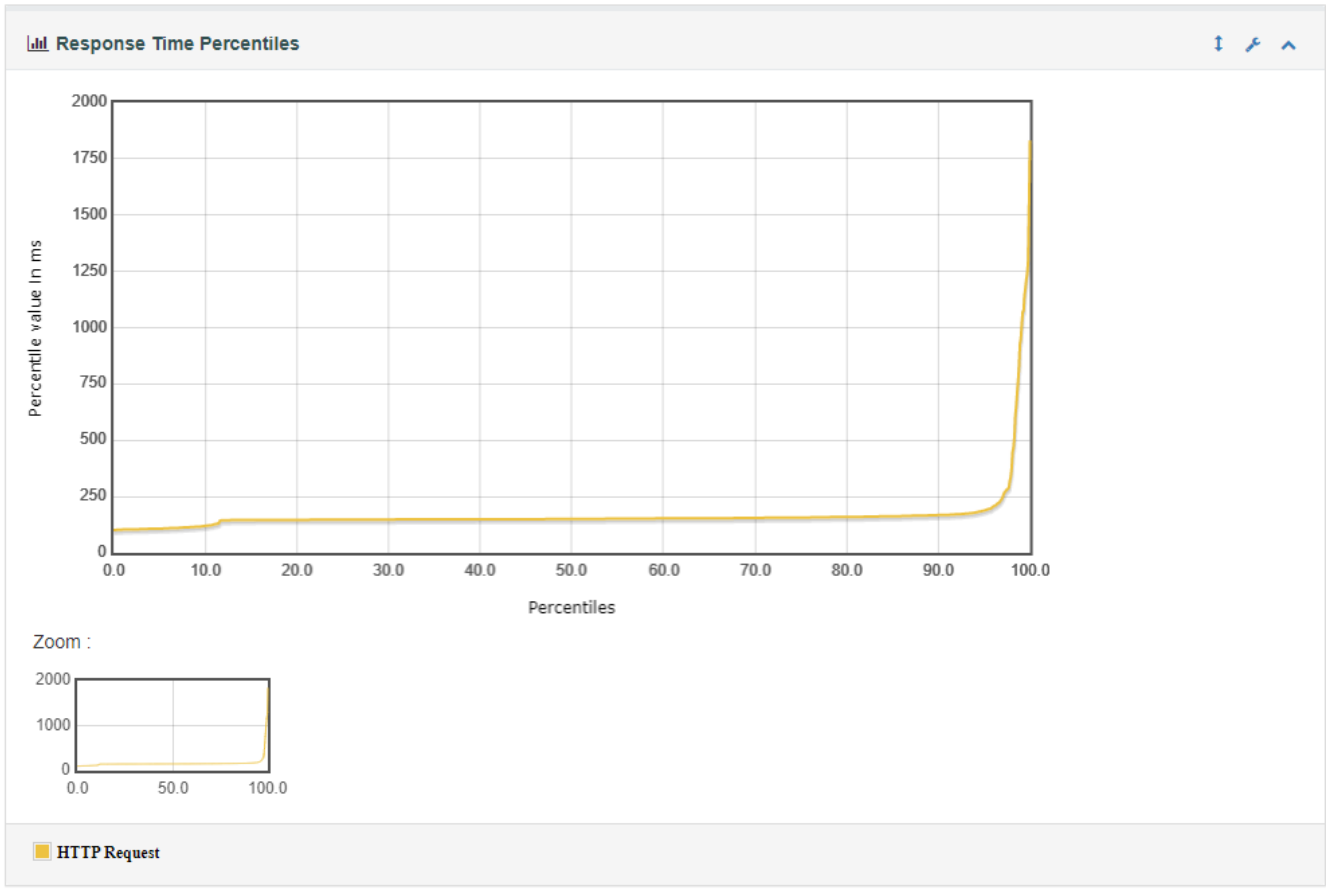


Рис. 3.39. Графік процентилю часу відповіді тесту на масштабованість ТС-09

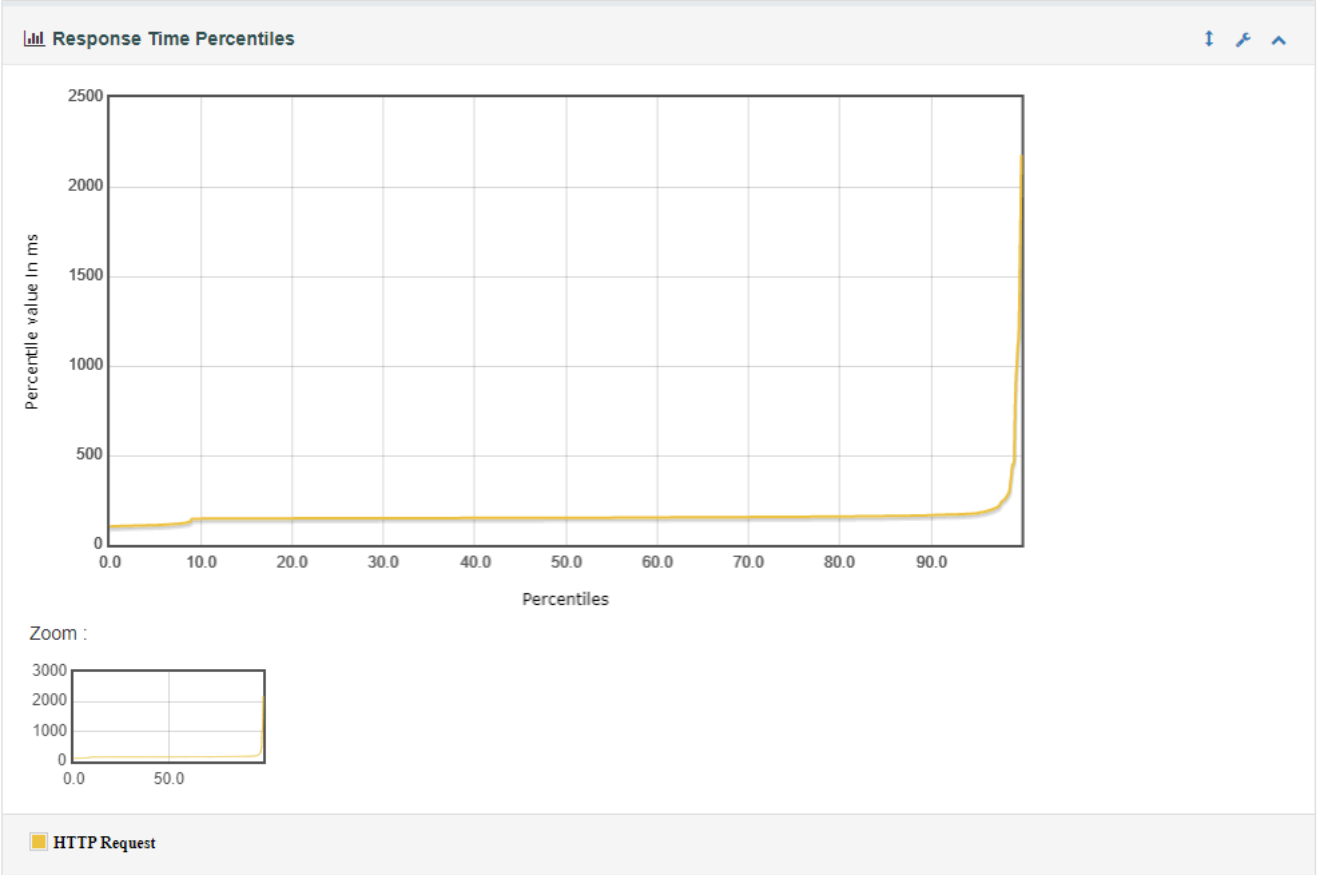


Рис. 3.40. Графік процентилю часу відповіді тесту на масштабованість ТС-10

Під час виконання тестування на масштабованість, тест ТС-11 пройшов успішно на 86,62%. На рис. 3.41 зображено загальний графік, який відсотково зображує виконання тестів. 19922 запитів з 23000 відправили http-запити і отримали відповідь.

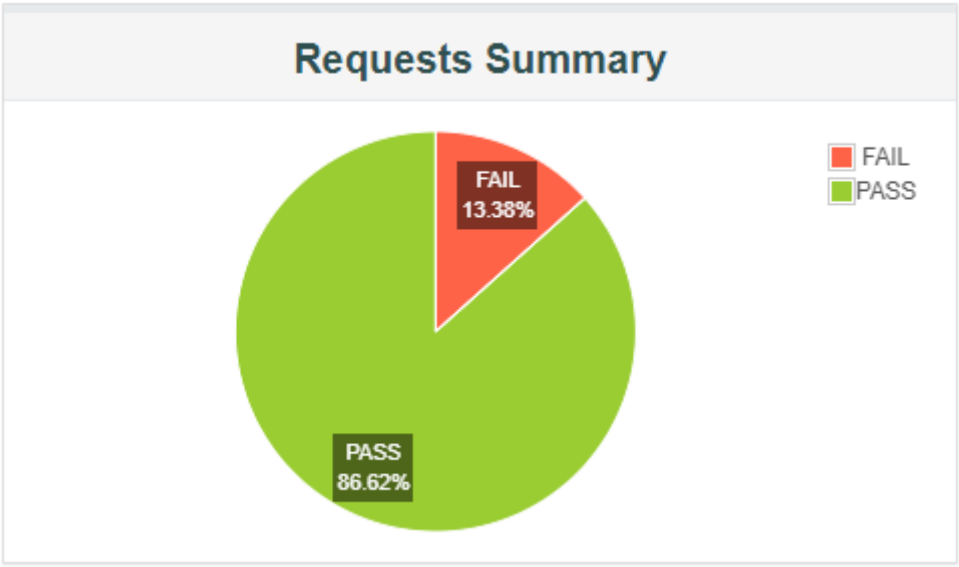


Рис. 3.41. Загальний графік виконання тесту на масштабованість ТС-11

Таблиця на рис. 3.42 зображує середнє значення індексу продуктивності тесту ТС-11 серед 23000 запитів до ІС у розмірі 40,8% з порогом толерантності у 500мс і межею відгуку у 1,5 секунди за http-запитами.

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.408	**500 ms**	**1 sec 500 ms**	**Total**
0.408	500 ms	1 sec 500 ms	HTTP Request

Рис. 3.42. Загальна таблиця індексу продуктивності ІС тесту на масштабованість ТС-11

Таблиця на рис. 3.43 відображає статистику навантажувального тесту ТС-11. Тест ТС-11 виконав 23000 http-запитів на сервер проекту і отримав 86,62% відповідей. Час відгуку ТС-11 становив від 2 мс до 16,725 секунди. За метриками тесту ТС-11 90% запитів були виконані за час, що <= 3,41 с, 95% <= 3,51 мс, а 99%

відповідно <= 3,94 с. Середня кількість запитів за час виконання тесту склала 76,22 запитів за секунду.

Statistics

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**23000**	**3078**	**13.38%**	**1329.05**	**2**	**16725**	**1549.00**	**3414.00**	**3517.00**	**3946.98**	**76.22**	**1067.27**	**18.35**
HTTP Request	23000	3078	13.38%	1329.05	2	16725	1549.00	3414.00	3517.00	3946.98	76.22	1067.27	18.35

Рис. 3.43. Таблиця результатів виконання тесту ТС-11

З графіків, що зображені на рис 3.44, рис 3.45 і рис 3.46 можна побачити, що під час тесту на масштабованість ТС-11, 38,57% (8872 запитів) були відпрацьовані <= 500мс, 4,37% (1006 запитів) від 500 мс до 1,5 с, 43,67% (10044 запити) > 1,5 с, і ще 13,38% (3078 запити) не отримали відповіді від сервера.

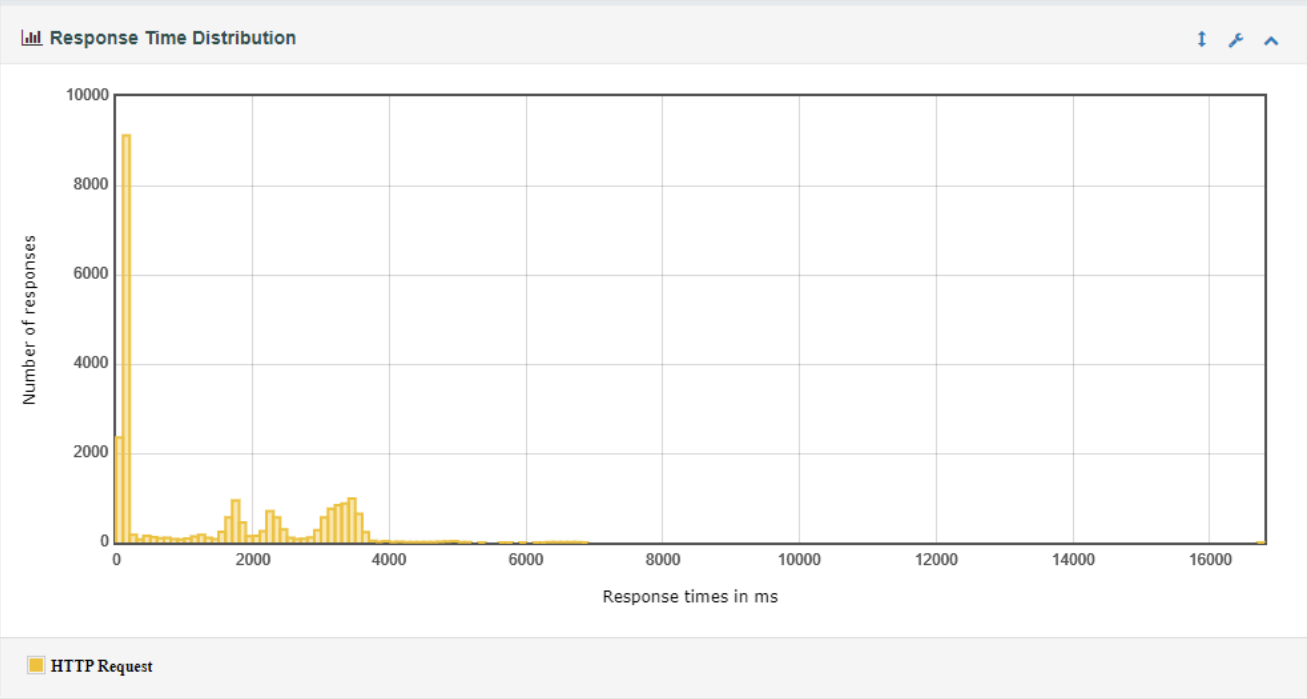


Рис. 3.44. Графік розподілу часу відповіді виконання тесту на масштабованість ТС-11

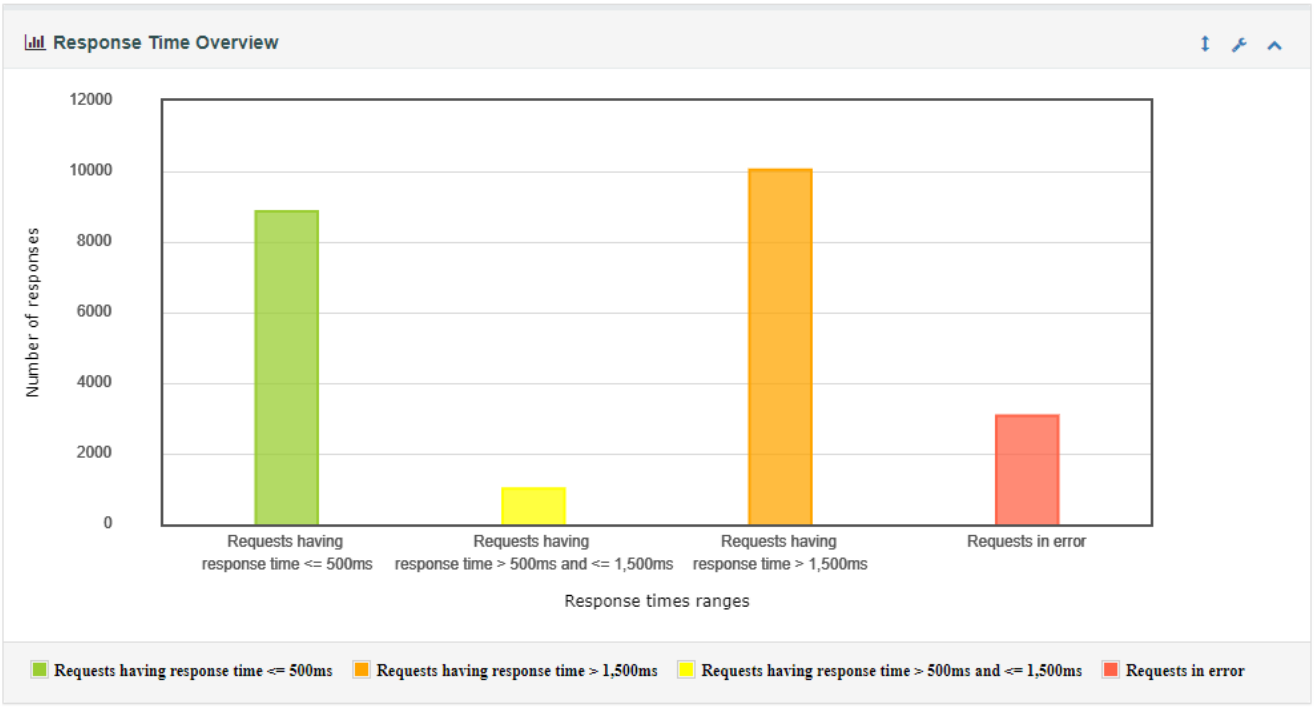


Рис.3.45. Графік часу відгуку на http-запити тесту на масштабованість ТС-11

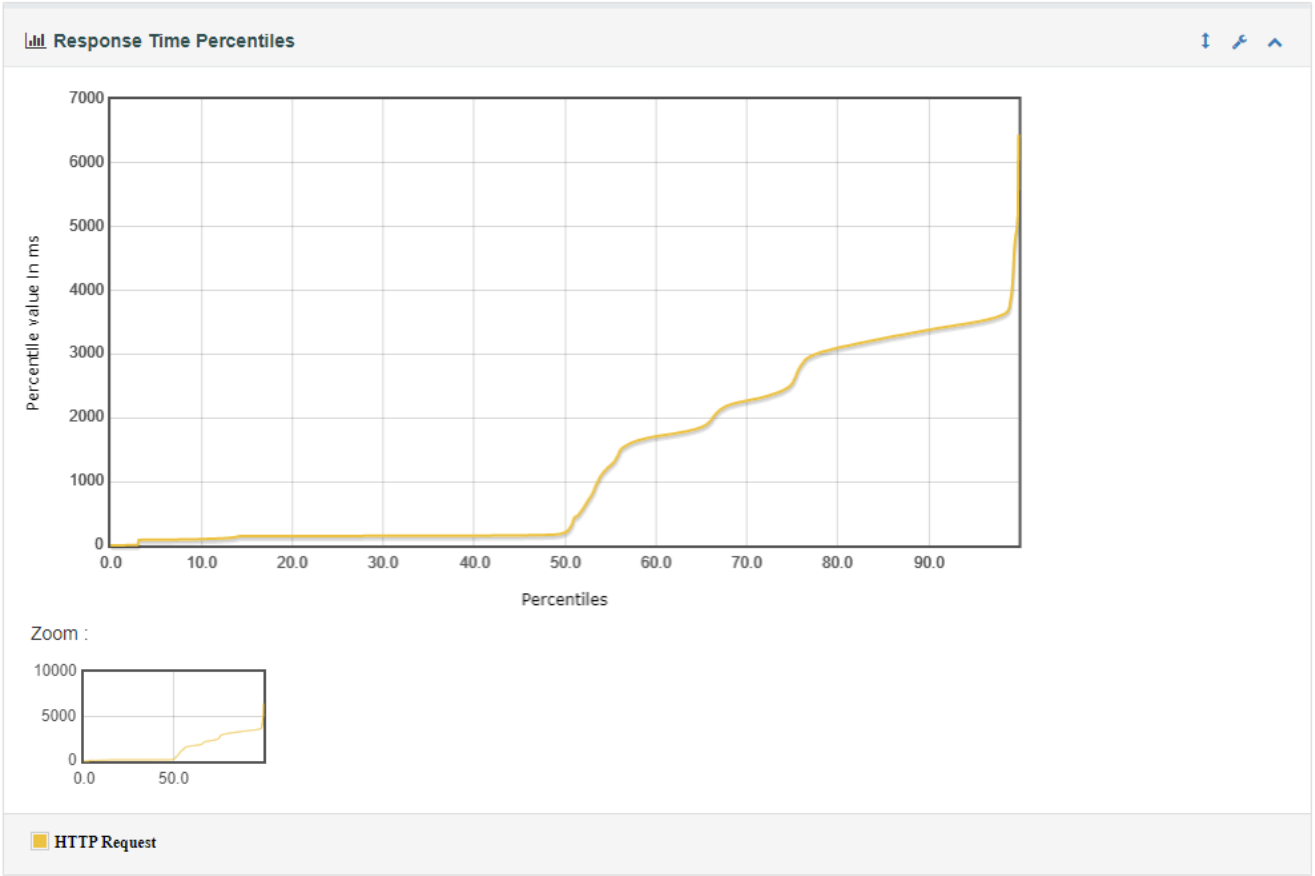


Рис.3.46. Графік процентилю часу відповіді тесту на масштабованість ТС-11

3.3. Функціональне тестування

Функціональне тестування – це клас комплексного тестування, який спрямований на перевірку відповідності функціональних вимог ПЗ до його реальних характеристик.

Суть функціонального тестування полягає в перевірці працездатності ПЗ або ІС загалом, відповідно до очікувань та вимог користувачів і/або власників.

Основною метою проведення функціонального тестування є виявлення багів та дефектів в роботі окремих функцій ПЗ, які можуть призвести до негативної поведінки функціоналу всієї ІС.

Існує різниця між функціональним тестуванням і тестуванням функціональності. Як зазначає автор книги «Тестування ПЗ», С. Куліков, посилаючись на статті «What is Functional testing (Testing of functions) in software?»[18] і «What is functionality testing in software?»[19]: «

- функціональне тестування (як антонім нефункціонального) спрямоване на перевірку того, які функції програми реалізовані, і що вони працюють правильно;
- тестування функціональності спрямоване на ті самі завдання, але акцент зміщений у бік дослідження програми у реальному робочому середовищі, після локалізації і в аналогічних ситуаціях».[16]

Оскільки проект Playzone вже є реалізованою самостійною ІС, в кваліфікаційній роботі виконується тестування функціональності проекту.

Для тестування функціональності проекту Playzone, було використано такі методики тестування:

– Тестування критичних значень;
– Тестування еквівалентних значень;
– Тестування сценаріїв використання.

Суть тестування критичних значень полягає в перевірці роботи ПЗ у межах допустимих значень: найбільшому та найменшому допустимому значенні, а також на значеннях, які можуть призвести до помилки або викликати збій функціональності.

Основною метою тестування критичних значень є перевірка коректності поведінки ПП при обробці критичних значень для забезпечення стабільної надійності функціоналу програми.

Суть тестування еквівалентних значень полягає у використанні представників класів еквівалентності вхідних даних при тестуванні ПЗ.

Основною метою тестування еквівалентних значень є зменшення кількості виконання тестів для валідації функціональності ПЗ з збереженням вичерпності даних тестів і збільшенням ефективності тестування.

Суть тестування сценаріїв використання полягає у перевірці поведінки програми до різних сценаріїв використання, які повинні виконуватися користувачами для використання ПЗ, відповідно до його призначення.

Основною метою тестування сценаріїв використання є перевірка коректної працездатності ПЗ в реальних умовах експлуатації, використовуючи різні сценарії взаємодії користувачів з ПЗ.

Тестування функціональності включає також в себе класи: тестування сірого ящику, тестування бізнес-логіки, реактивне тестування, позитивне тестування, негативне тестування, неінвазивне тестування, доменне тестування, інтеграційне тестування, ручне/мануальне тестування, тестування по тест-кейсам, дослідницьке тестування.

Загальний алгоритм виконаних дій виглядає таким чином:

1) Співбесіда з власником проекту:
    a. Визначення області тестування проекту;
    b. Визначення вимог до функціональності області тестування.
2) Тестування функціональності:
    a. Створення тест-кейсів для проведення тестування;
    b. Виконання тестів;
    c. Аналіз тестування;
    d. Оформлення баг-репортів.
3) Співбесіда з власником проекту.

Після розмови з власником проекту Playzone, було визначено область тестування функціональності, а саме необхідність тестування: реєстрації нового користувача, входу в систему та відновлення даних для авторизації; перевірка коректності системи оплати; пошукової системи.

Також, були визначені вимоги власника для перевірок до кожного об’єкту тестування.

Для системи реєстрації та входу:

- Реєстрація нового користувача:
  - Поле логін повинно бути не пустим і складатися мінімум з 3 символів;
  - Поле пароль повинно бути не пустим і складатися мінімум з 6 символів.
- Авторизація зареєстрованого користувача:
  - Етап авторизації повинен бути успішним за умови введення вірного логіну і паролю не авторизованим користувачем, які записані у БД і після натискання кнопки «Ввійти».
- Перевірка роботи функції відновлення пароля:
  - Етап відновлення паролю (функція зміни паролю у БД і надсилання його на зареєстровану пошту) повинен бути успішним за умови введення вірного логіну незареєстрованим користувачем, який записаний у БД, після натискання кнопки «Скидання».

Для перевірки коректності системи оплати:

- Система поповнення ігрового рахунку (система донату):
  - Нарахування ігрової валюти «Вірри» на ігровий рахунок повинно виконуватись після вказання суми авторизованим користувачем поповнення на сторінці «Донат», по натисканню кнопки «Пожертвувати», переходу на сторонній сервіс оплати транзакції з вказанням платіжної системи або використанням інтернет-банкінгу.
- Система обміну ігрової валюти:
  - Нарахування коштів «Голд» на ігровий рахунок повинно виконуватись після вказання невід’ємної суми обміну валюти «Вірри», яка не більше суми

валюти «Вірри» на рахунку авторизованого користувача, на сторінці «Магазин послуг», по натисканню кнопки «Обміняти».

– Система внутрішньо-ігрової купівлі:

  - Послуга «Плащ»/ «MiniViP» / «ViP» / «MaxiViP» повинна активуватись для авторизованого користувача, після натискання кнопки «Купити» вказаної послуги, списання кількості ігрової валюти у розмірі «10 000» / «3 000» / «5 000» / «8 000» «Вірр» з ігрового рахунку авторизованого користувача.

Для пошукової системи:

– Спроможності пошукової системи форуму:

  - Пошук повинен виконуватись при непустому значенню поля <input name = «keywords»> сторінки «Форум» і більше 3-х символів, не частіше 1 разу за 30 секунд.

При виконанні тестування функціональності, вимоги були розділені на 7 груп тест-кейсів, з загальною кількістю 24 тестування, використовуючи методики тестування критичних, еквівалентних значень і сценаріїв використання і описані у вигляді таблиць тест-кейсів, згідно стандарту 829/3787 – 2008.

У табл. А.1, А.2 додатку А, подані тест-кейси для тестування функціоналу процесу реєстрації користувачів на проекті. У формі реєстрації на сайті проекту Playzone, вказано 4 поля для введення: поле Логіну, поле Почтової адреси, поле Паролю і поле Повтор паролю. Під час створення тест-кейсів і проведення тестування, не приділялася увага тестуванню поля Поштової адреси, оскільки дане поле є елементом HTML <input> з атрибутом «type="email"», яке у 5-у поколінні мови гіпертексту автоматично визначається як поле для поштової адреси і перед відправленням даних на сервер автоматично виконує перевірку стандартного запису правил поштової адреси.

Тестування функціональності проводилося методикою класу мануального тестування, оскільки в даному випадку проект малого масштабу (до 5 000 користувачів), з обмеженим функціоналом, а отже створення автоматизованих тестів є не раціональним підходом.

У тестах TCF001- TCF009 були виконані перевірки процесу реєстрації нового користувача, використовуючи методики тестування функціональності, були визначені критичні параметри для тестування, а саме перевірки поля Логіну, де значення набували:

1) Пусте поле;
2) 2 символи;
3) 3 спец-символи;
4) 3 символи.

Відповідно значення поля пароля набували:

1) Пусте поле;
2) 6 пробілів;
3) 6 спец-символів;
4) 6 символів з пробілом поміж них;
5) 6 символів.

Тести TCF001 - TCF003 і TCF005 - TCF008 мали негативний сценарій, оскільки цей діапазон значень є одним із часто вразливих вузлів функціоналу полів реєстрації і одразу ж виконується перевірка неправильного використання функціоналу.

Результати тестів TCF006 - TCF008 провалені, що свідчить про знаходження багів у системі в процесі реєстрації. Якщо ж Логін користувача проекту може складатися з окремих символів (в даному випадку з тире і/або з нижніх підкреслень), то з точки зори безпеки персональних даних, пароль не повинен складатися з пробілів або знаків астериску. Для тестів TCF006 - TCF008 було сформовано баг-репорти у табл. Б.1 додатку Б. Таблиця баг-репортів для функціоналу реєстрації поля паролю створена і описана згідно стандарту IEEE 829-2008 [17]. Аналізуючи проведене тестування, можна стверджувати, що область функціоналу реєстрації потребує виправлення багів і подальшого дослідження.

У табл. А.3 додатку А, подані тест-кейси для тестування функціоналу авторизації користувачів на проекті. У формі авторизації на сайті проекту Playzone,

вказано 2 поля для введення: поле Ім’я користувача і поле Паролю. Згідно методик класу тестування, було скомпоновано тестування у вигляді 3-х тестів:

- введення користувачем невірного Ім’я користувача і вірного паролю;
- введення користувачем вірного Ім’я користувача і невірного паролю;
- введення користувачем вірного Ім’я користувача і вірного паролю.

Відповідно цьому, тести TCF010- TCF011 мали негативний сценарій, а тест TCF012 позитивний, оскільки у ньому виконується пряма перевірка правильності використання функціональності ІС. Усі 3 тести були успішно пройдені. Аналізуючи проведене тестування, можна стверджувати, що область функціоналу авторизації є канонічною.

У табл. А.4 додатку А, подано тест-кейс для тестування функціоналу системи оплати Донату для проекту. Для цієї області тестування було виконано лише тест TCF013, оскільки наступні бізнес-процеси для отримання зазначеної цілі виконується на сторонніх ресурсах інтернет-банкінгу. Тест TCF013 мав негативний сценарій, оскільки сума коштів для коду ІС є одним числовим параметром, який доводиться методикою еквівалентних значень нераціональним проведенням додаткового тестування. Аналізуючи проведене тестування, можна стверджувати, що функціональність системи оплати Донат має високий рівень контрольованості.

У табл. А.5 додатку А, подано тест-кейси для тестування функціоналу обміну ігрової валюти проекту. У цій групі згідно методик класу тестування було виконано 4 тести TCF014 - TCF017 для екземпляра класу , який конвертує ігрові валюти «Вірри» у «Голд». Кожен тест перевіряє граничні можливі значення введених користувачем. Тести TCF014 - TCF017 є негативного сценарію виконання, оскільки фінансові логічні компоненти будь-якої системи є одними з найкритичніших. Всі 4 тести були успішно пройдені. Аналізуючи проведене тестування, можна стверджувати, що область функціоналу обміну ігрової валюти досконала і має високий рівень контрольованості.

У табл. А.6 додатку А, подано тест-кейси для тестування функціоналу внутрішньо-ігрової купівлі в цифровому магазині проекту. Для цієї області було проведено 2 тести: тест TCF018 з негативним сценарієм, перевіряючий поведінку

системи не чесного надання послуг користувачу і тест TCF019 з позитивним сценарієм перевірки правильності виконання надання послуги. Обидва тести були успішно пройдені. Аналізуючи проведене тестування, можна стверджувати що функціональність внутрішньо-ігрової купівлі магазину цілісна і правильна.

У табл. А.7 додатку А, подано тест-кейси для тестування функціоналу пошукової системи Форуму проекту. Для пошукової системи було розроблено 5 сценаріїв тестування TCF020 - TCF024. Тести TCF020 - TCF023 несуть негативний характер для перевірки дій ІС на різні можливі пошукові запити користувачем всередині неї. Всі тести успішно пройдені. Аналізуючи проведене тестування, можна стверджувати, що пошукова система чутлива до запитів користувача і має високий рівень відгуку.

3.4. Висновок до 3 розділу

В даному розділі були розглянуті взаємопов’язані методики класів комплексного тестування ПЗ.

Виконана тісна співпраця з власником проекту Playzone: проаналізована ІС, її архітектура; визначені вразливі вузли ПП; визначені області ІС, потребуючі виконання тестів, формалізовані вимоги власника, створено алгоритми проведення тестування із взаємодією з замовником.

Проведено комплексне тестування засобами ПЗ і загальний аналіз проекту Playzone.

Були використані класи тестування, а саме:

1. Тестування безпеки:
    a. Пантестування;
    b. Тестування на вразливості.
2. Навантажувальне тестування :
    a. Тестування навантаження;
    b. Стресове тестування;
    c. Тестування на масштабованість.

3. Функціональне тестування:
    a. Тестування критичних значень;
    b. Тестування еквівалентних значень;
    c. Тестування сценаріїв використання.

Були створенні і оформлені звітові графіки, діаграми, табличне зображення тест-кейсів, баг-репортів з глибоким аналізом отриманих результатів. З результатами і звітами була донесена інформація у формі співбесіди власнику проекту Playzone про компоненти проекту і стану ІС в цілому. Після надання результатів і звітів, були впроваджені зміни в оптимізації даного проекту.

Аналізуючи проведене комплексне тестування Playzone, можна стверджувати, що проект в цілому є канонічним, має високий цілісний рівень контрольованості і досконалості та виконує основну мету функціоналу на високому рівні.

ВИСНОВОК

У кваліфікаційній роботі були розкриті проблеми тестування ПЗ та огляд існуючих класів тестування, їх методик та методів, які використовуються при комплексному тестуванні.

Також було проведено огляд найпоширеніших підходів та методологій до тестування розроблюваного ПЗ, а саме: класичного Waterfall, Spiral, тестування Extreme, Test Driven Development (TDD), Agile та DevOps методологій і окремо дослідницьке тестування. Визначені їхні відмінності, переваги і недоліки.

Було визначено, що:

1. Тестування - не якість програмного продукту, проте обидва тісно взаємопов’язані;
2. Проблематика низької якості програмного продукту взаємозалежна з компетенцією спеціалістів;
3. Не існує загального комплексного тестування, яке може бути досконалим для будь-якого розроблюваного програмного продукту;
4. Існує велика сукупність підходів до тестування як традиційних, так і набуваючих тенденцію, які комплексно вирішують питання якості програмного продукту;
5. Для великого і малого бізнесу необхідні різні комплексні комбінування підходів.

Було розглянуті і обрані різні інструменти та технології для реалізації комплексного тестування ПП:

- JavaScript;
- JMeter;
- JSR223 Sampler;
- OWASP ZAP.

Також було аргументовано їх вибір.

За допомоги обраних інструментів та технологій, було реалізовано комплексне тестування проекту Playzone в тісній співпраці з власником проекту Playzone: проаналізовані основні вразливі вузли ПП, визначені області ІС для яких виконувалось комплексне тестування з використанням різних класів тестування: тестування безпеки, навантажувального тестування і функціонального тестування з використанням їх методик.

Було створено звіти для власника проекту з діаграмами, графіками, описами тест-кейсів і оформленням баг-репортів, проведено детальний аналіз проекту загалом і надані рекомендації стосовно вдосконалення і впровадження оптимізації даного проекту.

Аналізуючи проведене комплексне тестування Playzone, можна стверджувати, що проект в цілому є канонічним, має високу контрольованість та виконує всю бізнес-функціональність високого рівня.

За результатами проведення комплексного тестування та виправлення багів, проект playzone.in.ua має більші перспективи для масштабованого розвитку в майбутньому.

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Дороті Грехем. Фундаментальне тестування програмного забезпечення / Дороті Грехем, Еріка ван Флітта, Ізабелли Уертон ; – 4-е видання, 2018. -256с.
2. Принципи тестування [Електронний ресурс]- Режим доступу - https://habr.com/ru/articles/699990/ (дата звернення 09.05.2023р.) – Назва з екрана.
3. James Whittaker. Як тестують в Google / James Whittaker, Jason Arbon, Jeff Carollo , Вид-цтво «Пітерс Перс»,-2-й том, 2013. -320с
4. Testcontainers: тестування з реальними залежностями [Електронний ресурс] - Режим доступу - https://habr.com/ru/articles/700286/ (дата звернення 13.05.2023р.) – Назва з екрана.
5. Agile чи Waterfall — який варіант відповідає вашому бізнесу? [Електронний ресурс] - Режим доступу - https://worksection.com/ua/blog/waterfall-vs-agile.html (дата звернення 13.05.2023р.) – Назва з екрана.
6. Spiral [Електронний ресурс] - Режим доступу - https://qalight.ua/baza-znaniy/cpiralna-model-spiral-model/ (дата звернення 13.05.2023р.) – Назва з екрана.
7. Екстремальне програмування (XP) [Електронний ресурс] - Режим доступу – https://wiki.cuspu.edu.ua/index.php/Екстремальне_програмування_(XP) (дата звернення 13.05.2023р.) – Назва з екрана.
8. Керована тестами розробка [Електронний ресурс] - Режим доступу - https://uk.wikipedia.org/wiki/Керована_тестами_розробка (дата звернення 13.05.2023р.) – Назва з екрана.
9. DevOps [Електронний ресурс] - Режим доступу - https://uk.wikipedia.org/wiki/DevOps (дата звернення 13.05.2023р.) – Назва з екрана.
10. JavaScript [Електронний ресурс] - Режим доступу - https://uk.wikipedia.org/wiki/JavaScript (дата звернення 14.05.2023р.) – Назва з екрана.
11. JMeter [Електронний ресурс] - Режим доступу - https://uk.wikipedia.org/wiki/JMeter (дата звернення 14.05.2023р.) – Назва з екрана.

12. Jmeter [Електронний ресурс] - Режим доступу - https://jmeter.apache.org (дата звернення 14.05.2023р.) – Назва з екрана.

13. Підручник з OWASP ZAP: Всебічний огляд інструменту OWASP ZAP[Електронний ресурс] - Режим доступу - https://uk.myservername.com/owasp-zap-tutorial-comprehensive-review-owasp-zap-tool (дата звернення 16.05.2023р.) – Назва з екрана.

14. Jmeter :component_reference [Електронний ресурс] - Режим доступу - https://jmeter.apache.org/usermanual/component_reference.html (дата звернення 17.05.2023р.) – Назва з екрана.

15. Савін Р. Тестування dot com / Р. Савін; Вид-во «Дело», 2006. – 316с.

16. Куліков С. Тестування ПЗ - Базовий курс / Куліков С. - 2-е видання, 2018. – 296с.

17. IEEE 829-2008IEEE Standard for Software and System Test Documentation [Електронний ресурс] - Режим доступу - https://standards.ieee.org/ieee/829/3787/ (дата звернення 18.05.2023р.) – Назва з екрана.

18. What is Functional testing (Testing of functions) in software? [Електронний ресурс] - Режим доступу - https://tryqa.com/what-is-functional-testing-testing-of-functions-in-software/ (дата звернення 18.05.2023р.) – Назва з екрана.

19. What is functionality testing in software? [Електронний ресурс] - Режим доступу - https://tryqa.com/what-is-functionality-testing-in-software/ (дата звернення 18.05.2023р.) – Назва з екрана.

ДОДАТКИ

Додаток А

Табл. А.1

Тест-кейси для тестування функціоналу реєстрації поля Логін

Номер тест-кейсу	TCF001	TCF002	TCF003	TCF004
Опис об’єкту	<input type="text" name="username" value="" placeholder="" required="" autofocus=""> веб-сторінки https://playzone.in.ua/login	<input type="text" name="username" value="" placeholder="" required="" autofocus=""> веб-сторінки https://playzone.in.ua/login	<input type="text" name="username" value="" placeholder="" required="" autofocus=""> веб-сторінки https://playzone.in.ua/login	<input type="text" name="username" value="" placeholder="" required="" autofocus=""> веб-сторінки https://playzone.in.ua/login
Опис функції	Перевірка поля Логін процесу реєстрації	Перевірка поля Логін процесу реєстрації	Перевірка поля Логін процесу реєстрації	Перевірка поля Логін процесу реєстрації
Опис сценарію	Введення користувачем некомпетентного значення поля «Логін» для виконання реєстрації	Введення користувачем некомпетентного значення поля «Логін» для виконання реєстрації	Введення користувачем некомпетентного значення поля «Логін» для виконання реєстрації	Введення користувачем компетентного значення поля «Логін» для виконання реєстрації

Продовження таблиці А.1

Попередні умови	Доступ до інтернету, браузер Google Chrome 112.0 або інший браузер версії від 01.01.2023	Доступ до інтернету, браузер Google Chrome 112.0 або інший браузер версії від 01.01.2023	Доступ до інтернету, браузер Google Chrome 112.0 або інший браузер версії від 01.01.2023	Доступ до інтернету, браузер Google Chrome 112.0 або інший браузер версії від 01.01.2023
Кроки виконання	1)Відкриття головної сторінки проекту Playzone https://playzone.in.ua 2)В правій частині меню натискання кнопки «Реєстрація» 3)Вибір поля «Логін» 4)Введення значення « » 5)Вибір поля «Пошта»	1)Відкриття головної сторінки проекту Playzone https://playzone.in.ua 2)В правій частині меню натискання кнопки «Реєстрація» 3)Вибір поля «Логін» 4)Введення значення « » 5)Вибір поля «Пошта»	1)Відкриття головної сторінки проекту Playzone https://playzone.in.ua 2)В правій частині меню натискання кнопки «Реєстрація» 3)Вибір поля «Логін» 4)Введення значення «***» 5)Вибір поля «Пошта»	1)Відкриття головної сторінки проекту Playzone https://playzone.in.ua 2)В правій частині меню натискання кнопки «Реєстрація» 3)Вибір поля «Логін» 4)Введення значення «------» 5)Вибір поля «Пошта»

Кроки виконання	6)Введення значення існуючої пошти 7) Вибір поля «Пароль» 8) Введення значення паролю (>=6 символів) 9)Вибір поля «Повтор паролю» 10) Введення значення з п.8 11) Натискання кнопки «Реєстрація»	6)Введення значення існуючої пошти 7) Вибір поля «Пароль» 8) Введення значення паролю (>=6 символів) 9)Вибір поля «Повтор паролю» 10) Введення значення з п.8 11) Натискання кнопки «Реєстрація»	6)Введення значення існуючої пошти 7) Вибір поля «Пароль» 8) Введення значення паролю (>=6 символів) 9)Вибір поля «Повтор паролю» 10) Введення значення з п.8 11) Натискання кнопки «Реєстрація»	6)Введення значення існуючої пошти 7) Вибір поля «Пароль» 8) Введення значення паролю (>=6 символів) 9)Вибір поля «Повтор паролю» 10) Введення значення з п.8 11) Натискання кнопки «Реєстрація»
Критерії позитивності сценарію	Негативний сценарій	Негативний сценарій	Негативний сценарій	Позитивний сценарій

Завершення таблиці А.1

Очікуваний результат	Виведення помилки на екран користувача	Виведення помилки на екран користувача	Виведення помилки на екран користувача	Успішна реєстрація, перехід до головної веб-сторінки проекту https://playzone.in.ua
Фактичний результат	Виведення помилки на екран користувача «Please enter a login»	Виведення помилки на екран користувача «Please enter a login»	Виведення помилки на екран користувача «Разрешены только буквы, цифры, тире и символы подчеркивания.»	Успішна реєстрація, перехід до головної веб-сторінки проекту https://playzone.in.ua
Статус	PASS / Пройдений	PASS / Пройдений	PASS / Пройдений	PASS / Пройдений

Продовження Додатку А

Табл. А.2. Тест-кейси для тестування функціоналу реєстрації поля Пароль

Номер тест-кейсу	TCF005	TCF006	TCF007	TCF008	TCF009
Опис об’єкту	<input type="password" id="registration_form_plainPassword_first" name="registration_form[plainPassword][first]" required="required" class="password-field" aria-autocomplete="list"> веб-сторінки https://playzone.in.ua/login	<input type="password" id="registration_form_plainPassword_first" name="registration_form[plainPassword][first]" required="required" class="password-field" aria-autocomplete="list"> веб-сторінки https://playzone.in.ua/login	<input type="password" id="registration_form_plainPassword_first" name="registration_form[plainPassword][first]" required="required" class="password-field" aria-autocomplete="list"> веб-сторінки https://playzone.in.ua/login	<input type="password" id="registration_form_plainPassword_first" name="registration_form[plainPassword][first]" required="required" class="password-field" aria-autocomplete="list"> веб-сторінки https://playzone.in.ua/login	<input type="password" id="registration_form_plainPassword_first" name="registration_form[plainPassword][first]" required="required" class="password-field" aria-autocomplete="list"> веб-сторінки https://playzone.in.ua/login

Продовження таблиці А.2

Опис функції	Перевірка поля Пароль процесу реєстрації	Перевірка поля Пароль процесу реєстрації	Перевірка поля Пароль процесу реєстрації	Перевірка поля Пароль процесу реєстрації	Перевірка поля Пароль процесу реєстрації
Опис сценарію	Введення користувачем некомпетентного значення поля «Пароль» для виконання реєстрації	Введення користувачем некомпетентного значення поля «Пароль» для виконання реєстрації	Введення користувачем некомпетентного значення поля «Пароль» для виконання реєстрації	Введення користувачем некомпетентного значення поля «Пароль» для виконання реєстрації	Введення користувачем компетентного значення поля «Пароль» для виконання реєстрації
Попередні умови	Доступ до інтернету, браузер Google Chrome 112.0 або інший браузер версії від 01.01.2023	Доступ до інтернету, браузер Google Chrome 112.0 або інший браузер версії від 01.01.2023	Доступ до інтернету, браузер Google Chrome 112.0 або інший браузер версії від 01.01.2023	Доступ до інтернету, браузер Google Chrome 112.0 або інший браузер версії від 01.01.2023	Доступ до інтернету, браузер Google Chrome 112.0 або інший браузер версії від 01.01.2023

Продовження Додатку А

Продовження таблиці А.2

Кроки виконання	1)Відкриття головної сторінки https://playzone.in.ua 2)В правій частині меню натискання кнопки «Реєстрація» 3)Вибір поля «Логін» 4)Введення значення Ім'я користувача (>3 символів, літер, цифр, тире, нижніх підкреслювань)	1)Відкриття головної сторінки https://playzone.in.ua 2)В правій частині меню натискання кнопки «Реєстрація» 3)Вибір поля «Логін» 4)Введення значення Ім'я користувача (>3 символів, літер, цифр, тире, нижніх підкреслювань)	1)Відкриття головної сторінки https://playzone.in.ua 2)В правій частині меню натискання кнопки «Реєстрація» 3)Вибір поля «Логін» 4)Введення значення Ім'я користувача (>3 символів, літер, цифр, тире, нижніх підкреслювань)	1)Відкриття головної сторінки https://playzone.in.ua 2)В правій частині меню натискання кнопки «Реєстрація» 3)Вибір поля «Логін» 4)Введення значення Ім'я користувача (>3 символів, літер, цифр, тире, нижніх підкреслювань)	1)Відкриття головної сторінки https://playzone.in.ua 2)В правій частині меню натискання кнопки «Реєстрація» 3)Вибір поля «Логін» 4)Введення значення Ім'я користувача (>3 символів, літер, цифр, тире, нижніх підкреслювань)

Продовження таблиці А.2

Кроки виконання	5)Вибір поля «Пошта» 6)Введення значення існуючої пошти 7) Вибір поля «Пароль» 8) Введення значення «123» 9)Вибір поля «Повтор паролю» 10) Введення значення з п.8 11) Натискання кнопки «Реєстрація»	5)Вибір поля «Пошта» 6)Введення значення існуючої пошти 7) Вибір поля «Пароль» 8) Введення значення « » 9)Вибір поля «Повтор паролю» 10) Введення значення з п.8 11) Натискання кнопки «Реєстрація»	5)Вибір поля «Пошта» 6)Введення значення існуючої пошти 7) Вибір поля «Пароль» 8) Введення значення «******» 9)Вибір поля «Повтор паролю» 10) Введення значення з п.8 11) Натискання кнопки «Реєстрація»	5)Вибір поля «Пошта» 6)Введення значення існуючої пошти 7) Вибір поля «Пароль» 8) Введення значення «ABC 123» 9)Вибір поля «Повтор паролю» 10) Введення значення з п.8 11) Натискання кнопки «Реєстрація»	5)Вибір поля «Пошта» 6)Введення значення існуючої пошти 7) Вибір поля «Пароль» 8) Введення значення «123456» 9)Вибір поля «Повтор паролю» 10) Введення значення з п.8 11) Натискання кнопки «Реєстрація»

Продовження таблиці А.2

Критерії позитивності сценарію	Негативний сценарій	Негативний сценарій	Негативний сценарій	Негативний сценарій	Позитивний сценарій
Очікуваний результат	Виведення помилки на екран користувача	Виведення помилки на екран користувача	Виведення помилки на екран користувача	Виведення помилки на екран користувача	Успішна реєстрація, перехід до головної веб-сторінки проекту https://playzone.in.ua
Фактичний результат	Виведення помилки на екран користувача «Длина пароля должна быть 6 символов или больше»	Успішна реєстрація, перехід до головної веб-сторінки проекту https://playzone.in.ua З логіном «_____» і паролем « »	Успішна реєстрація, перехід до головної веб-сторінки проекту https://playzone.in.ua З логіном «---___---» і паролем «******»	Успішна реєстрація, перехід до головної веб-сторінки проекту https://playzone.in.ua З логіном «-_--_--_-» і паролем «ABC 123»	Успішна реєстрація, перехід до головної веб-сторінки проекту https://playzone.in.ua

Завершення таблиці А.2

Статус	PASS / Пройдений	FAIL / Провалений	FAIL / Провалений	FAIL / Провалений	PASS / Пройдений

Табл. А.3.

Тест-кейси для тестування функціоналу авторизації

Номер тест-кейсу	TCF010	TCF011	TCF012
Опис об’єкту	<input type="text" name="username" value="" placeholder="" required="" autofocus="">, <input type="password" name="password" placeholder="" required=""> веб-сторінки https://playzone.in.ua/login	<input type="text" name="username" value="" placeholder="" required="" autofocus="">, <input type="password" name="password" placeholder="" required=""> веб-сторінки https://playzone.in.ua/login	<input type="text" name="username" value="" placeholder="" required="" autofocus="">, <input type="password" name="password" placeholder="" required=""> веб-сторінки https://playzone.in.ua/login

Продовження таблиці А.3

Опис функції	Перевірка поля Логін і Пароль процесу авторизації	Перевірка поля Логін і Пароль процесу авторизації	Перевірка поля Логін і Пароль процесу авторизації
Опис сценарію	Введення користувачем некомпетентного значення поля «Ім’я користувача» і компетентного значення поля «Пароль» для виконання авторизаї	Введення користувачем компетентного значення «Ім’я користувача» і некомпетентного значення поля «Пароль» для виконання авторизаї	Введення користувачем компетентного значення «Ім’я користувача» і компетентного значення поля «Пароль» для виконання авторизаї
Попередні умови	Доступ до інтернету, браузер Google Chrome 112.0 або інший браузер версії від 01.01.2023	Доступ до інтернету, браузер Google Chrome 112.0 або інший браузер версії від 01.01.2023	Доступ до інтернету, браузер Google Chrome 112.0 або інший браузер версії від 01.01.2023
Кроки виконання	1)Відкриття головної сторінки проекту Playzone https://playzone.in.ua	1)Відкриття головної сторінки проекту Playzone https://playzone.in.ua	1)Відкриття головної сторінки проекту Playzone https://playzone.in.ua

Продовження таблиці А.3

Кроки виконання	2)В правій частині меню натискання кнопки «Авторизація» 3)Вибір поля «Ім’я користувача» 4)Введення значення Ім’я користувача «123123456» 5) Вибір поля «Пароль» 6) Введення значення «123456» 7) Натискання кнопки «Ввійти»	2)В правій частині меню натискання кнопки «Авторизація» 3)Вибір поля «Ім’я користувача» 4)Введення значення Ім’я користувача «123456123456» 5) Вибір поля «Пароль» 6) Введення значення «123123456» 7) Натискання кнопки «Ввійти»	2)В правій частині меню натискання кнопки «Авторизація» 3)Вибір поля «Ім’я користувача» 4)Введення значення Ім’я користувача «123456123456» 5) Вибір поля «Пароль» 6) Введення значення «123456» 7) Натискання кнопки «Ввійти»
Критерії позитивності сценарію	Негативний сценарій	Негативний сценарій	Позитивний сценарій

Завершення таблиці А.3

Очікуваний результат	Виведення помилки на екран користувача	Виведення помилки на екран користувача	Успішна авторизація, перехід до головної веб-сторінки проекту https://playzone.in.ua
Фактичний результат	Виведення помилки на екран користувача «Логин или пароль не верны»	Виведення помилки на екран користувача «Логин или пароль не верны»	Успішна авторизація, перехід до головної веб-сторінки проекту https://playzone.in.ua
Статус	PASS / Пройдений	PASS / Пройдений	PASS / Пройдений

Табл. А.4

Тест-кейси для тестування функціоналу системи оплати Донат

Номер тест-кейсу	TCF013
Опис об’єкту	<input type="number" name="sum" id="sum" value="100"> веб-сторінки https://playzone.in.ua/lk/donate.html
Опис функції	Перевірка поля «Сума в грн» процесу оплати Донат

Завершення таблиці А.4

Опис сценарію	Регулювання значення поля «Сума в грн» для створення квитанції на оплату
Попередні умови	Реєстрація/Авторизація користувача. Перехід на сторінку https://playzone.in.ua/lk/donate.html
Кроки виконання	1)Обрання поля «Сума в грн» 2)Встановлення значення поля «10000000» 3) Натиснення кнопки «Пожертвувати» 4) Повернутися на сторінку https://playzone.in.ua/lk/donate.html 5) Встановлення значення поля «-1» 6) Натиснення кнопки «Пожертвувати»
Критерії позитивності сценарію	Негативний сценарій
Очікуваний результат	Виведення помилки на екран користувача
Фактичний результат	Значення поля «Сума в грн» автоматично залишається без змін з мінімальним значенням «20» і максимальним значенням «99999»
Статус	PASS / Пройдений

Продовження Додатку А
Табл. А.5

Тест-кейси для тестування функціоналу Обмін ігрової валюти

Номер тест-кейсу	TCF014	TCF015	TCF016	TCF017
Опис об’єкту	Екземпляр класу веб-сторінки https://playzone.in.ua/lk/services.html	Екземпляр класу веб-сторінки https://playzone.in.ua/lk/services.html	Екземпляр класу веб-сторінки https://playzone.in.ua/lk/services.html	Екземпляр класу веб-сторінки https://playzone.in.ua/lk/services.html
Опис функції	Перевірка бізнес-логіки поля «Вірри» процесу Обміну внутрішньо-ігрових валют	Перевірка бізнес-логіки поля «Вірри» процесу Обміну внутрішньо-ігрових валют	Перевірка бізнес-логіки поля «Вірри» процесу Обміну внутрішньо-ігрових валют	Перевірка бізнес-логіки поля «Вірри» процесу Обміну внутрішньо-ігрових валют
Опис сценарію	Введення користувачем некомпетентного значення поля «Вірри» для виконання обміну	Введення користувачем некомпетентного значення поля «Вірри» для виконання обміну	Введення користувачем некомпетентного значення поля «Вірри» для виконання обміну	Введення користувачем компетентного значення поля «Вірри» для виконання обміну

Продовження таблиці А.5

Попередні умови	Реєстрація/Авторизація користувача. Перехід на сторінку https://playzone.in.ua/lk/services.html	Реєстрація/Авторизація користувача. Перехід на сторінку https://playzone.in.ua/lk/services.html	Реєстрація/Авторизація користувача. Перехід на сторінку https://playzone.in.ua/lk/services.html	Реєстрація/Авторизація користувача. Перехід на сторінку https://playzone.in.ua/lk/services.html
Кроки виконання	1) Обрання поля «Вірри» 2) Вставновлення значення поля «-100» 3) Натиснення кнопки «Обміняти»	1) Обрання поля «Вірри» 2) Вставновлення значення поля «%&&%» 3) Натиснення кнопки «Обміняти»	1) Обрання поля «Вірри» 2) Вставновлення значення поля «100000» 3) Натиснення кнопки «Обміняти»	1) Обрання поля «Вірри» 2) Вставновлення значення поля «1» 3) Натиснення кнопки «Обміняти»
Критерії позитивності сценарію	Негативний сценарій	Негативний сценарій	Негативний сценарій	Позитивний сценарій

Завершення таблиці А.5

Очікуваний результат	Виведення помилки на екран користувача	Виведення помилки на екран користувача	Виведення помилки на екран користувача	Виведення повідомлення на екран користувача успішного обміну
Фактичний результат	Виведення помилки на екран користувача: «Ошибка проверки данных»	Поле чутливо до спецсимволів – відбуається уникнення заповнення поля	Виведення помилки на екран користувача:«Не хватает денег»	Виведення повідомлення на екран користувача успішного обміну: «Обмен успешно произведен»
Статус	PASS / Пройдений	PASS / Пройдений	PASS / Пройдений	PASS / Пройдений

Табл. А.6

Тест-кейси для тестування функціоналу Внутрішньо-ігрової купівлі

Номер тест-кейсу	TCF018	TCF019

Продовження таблиці А.6

Опис об’єкту	Екземпляр класу веб-сторінки https://playzone.in.ua/lk/services.html	Екземпляр класу веб-сторінки https://playzone.in.ua/lk/services.html
Опис функції	Перевірка бізнес-логіки внутрішньо-ігрової купівлі	Перевірка бізнес-логіки внутрішньо-ігрової купівлі
Опис сценарію	Обрання купівлі послуги внутрішньо-ігрового магазину з вартістю, вищою за ігровий баланс	Обрання купівлі послуги внутрішньо-ігрового магазину з вартістю, нижчьою за ігровий баланс
Попередні умови	Реєстрація/Авторизація користувача. Перехід на сторінку https://playzone.in.ua/lk/services.html	Реєстрація/Авторизація користувача. Перехід на сторінку https://playzone.in.ua/lk/services.html
Кроки виконання	1) В меню послуг обрання послуги MaxiVIP 2) Натискання кнопки «Купити»	1) В меню послуг обрання послуги MiniVIP 2) Натискання кнопки «Купити»
Критерії позитивності сценарію	Негативний сценарій	Позитивний сценарій

Завершення таблиці А.6

Очікуваний результат	Виведення помилки на екран користувача	Виведення повідомлення на екран користувача успішної купівлі
Фактичний результат	Виведення помилки на екран користувача: «Не хватает денег»	Виведення повідомлення на екран користувача успішної купівлі: «Вы купили услугу»
Статус	PASS / Пройдений	PASS / Пройдений

Табл. А.7

Тест-кейси для тестування функціоналу Пошукової системи Форум

Номер тест-кейсу	TCF020	TCF021	TCF022	TCF023	TCF024
Опис об’єкту	поле <input name="keywords" type="text" class="textbox"> веб-сторінки https://playzone.in.ua/forum/	поле <input name="keywords" type="text" class="textbox"> веб-сторінки https://playzone.in.ua/forum/	поле <input name="keywords" type="text" class="textbox"> веб-сторінки https://playzone.in.ua/forum/	поле <input name="keywords" type="text" class="textbox"> веб-сторінки https://playzone.in.ua/forum/	поле <input name="keywords" type="text" class="textbox"> веб-сторінки https://playzone.in.ua/forum/

Продовження таблиці А.7

Опис функції	Перевірка запиту функції пошуку веб-сторінки Форум	Перевірка запиту функції пошуку веб-сторінки Форум	Перевірка запиту функції пошуку веб-сторінки Форум	Перевірка запиту функції пошуку веб-сторінки Форум	Перевірка запиту функції пошуку веб-сторінки Форум
Опис сценарію	Введення некомпетентного значення поля «Пошук»	Введення некомпетентного значення поля «Пошук»	Введення некомпетентного значення поля «Пошук»	Введення некомпетентного значення поля «Пошук»	Введення компетентного значення поля «Пошук»

Продовження таблиці А.7

Попередні умови	Реєстрація/Авторизація користувача. Перехід на сторінку https://playzone.in.ua/forum/	Реєстрація/ Авторизація користувача. Перехід на сторінку https://playzone.in.ua/forum/	Реєстрація/Авторизація користувача. Перехід на сторінку https://playzone.in.ua/forum/	Реєстрація/ Авторизація користувача. Перехід на сторінку https://playzone.in.ua/forum/	Реєстрація/Авторизація користувача. Перехід на сторінку https://playzone.in.ua/forum/
Кроки виконання	1) Обрання поля «Пошук» 2) Введення значення поля «» 3) Натиснення кнопки «Пошук»	1) Обрання поля «Пошук» 2) Введення значення поля «12» 3) Натиснення кнопки «Пошук»	1) Обрання поля «Пошук» 2) Введення значення поля « » 3) Натиснення кнопки «Пошук»	1) Обрання поля «Пошук» 2) Введення значення поля «$%&» 3) Натиснення кнопки «Пошук»	1) Обрання поля «Пошук» 2) Введення значення поля «Пасха» 3) Натиснення кнопки «Пошук»
Критерії позитивності сценарію	Негативний сценарій	Негативний сценарій	Негативний сценарій	Негативний сценарій	Позитивний сценарій

Завершення таблиці А.7

Очікуваний результат	Виведення помилки на екран користувача	Виведення помилки на екран користувача	Виведення помилки на екран користувача	Виведення помилки на екран користувача	Виведення релевантних запиту відповідей
Фактичний результат	Виведення помилки на екран користувача: «Ошибка»	Виведення помилки на екран користувача: «Ошибка. Один или несколько поисковых терминов были короче минимальной длины. Минимальный срок поиска длина 3 символов.Если вы пытаетесь искать целую фразу, заключите её в двойные кавычки. Например "The quick brown fox jumps over the lazy dog".»	Виведення помилки на екран користувача: «Ошибка. Вы не ввели никаких условий поиска. Как минимум, необходимо ввести либо какие-то слова для поиска или имя пользователя для поиска.»	Виведення помилки на екран користувача: «Ошибка»	Виведення релевантних запиту відповідей
Статус	PASS / Пройдений	PASS / Пройдений	PASS / Пройдений	PASS / Пройдений	PASS / Пройдений

Табл. Б.1

Баг-репорти для функціоналу реєстрації поля Пароль

ID помилки	BR001	BR002	BR003
Номер тест-кейсу	TCF006	TCF007	TCF008
Назва помилки	Успішна реєстрація, перехід до головної веб-сторінки проекту https://playzone.in.ua з використанням паролю « »	Успішна реєстрація, перехід до головної веб-сторінки проекту https://playzone.in.ua з використанням паролю «******»	Успішна реєстрація, перехід до головної веб-сторінки проекту https://playzone.in.ua з використанням паролю «ABC 123»
Ступінь важливості	S3 Значний (Major)	S3 Значний (Major)	S3 Значний (Major)
Передумови	Відкриття сайту https://playzone.in.ua Перехід на сторінку реєстрації	Відкриття сайту https://playzone.in.ua Перехід на сторінку реєстрації	Відкриття сайту https://playzone.in.ua Перехід на сторінку реєстрації
Вхідні дані	Заповнення полів Клік на кнопку «Реєстрація»	Заповнення полів Клік на кнопку «Реєстрація»	Заповнення полів Клік на кнопку «Реєстрація»
Очікуваний результат	Виведення помилки на екран користувача	Виведення помилки на екран користувача	Виведення помилки на екран користувача

Завершення таблиці Б.1

Фактичний результат	Успішна реєстрація, перехід до головної веб-сторінки проекту https://playzone.in.ua	Успішна реєстрація, перехід до головної веб-сторінки проекту https://playzone.in.ua	Успішна реєстрація, перехід до головної веб-сторінки проекту https://playzone.in.ua
Кроки для відтворення	Відриття веб-сторінки https://playzone.in.ua/registration Введення Логіну(довільного не зайнятого), Почти (довільної, існуючої, не зайнятої), Паролю зі значенням « »	Відриття веб-сторінки https://playzone.in.ua/registration Введення Логіну(довільного не зайнятого), Почти (довільної, існуючої, не зайнятої), Паролю зі значенням «******»	Відриття веб-сторінки https://playzone.in.ua/registration Введення Логіну(довільного не зайнятого), Почти (довільної, існуючої, не зайнятої), Паролю зі значенням «ABC 123»
Середовище:	Windows 10(x64), Google Chrome 112.0	Windows 10(x64), Google Chrome 112.0	Windows 10(x64), Google Chrome 112.0
Прикріплення	-	-	-