Веб-Cервіс BookBag

План тестування

Рев. <1,0>

Історія версій

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версія** | **опис** | **Автор** |
| <20/09/2023> | <1,0> | <Розробка веб-сервісу з продажу/купівлі книг з використанням ReactJS, Next.js та Strapi.io> | <Мельник О.В.> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Зміст

1. Вступ 4

1.1 Мета 4

1.2 Передумови 4

1.3 Сфера застосування 4

1.4 Ідентифікація проекту 5

2. Вимоги до тестування 5

3. Стратегія тестування 5

3.1 Типи тестування 5

3.1.1 Перевірка цілісності даних і бази даних 5

3.1.2 Функціональне тестування 6

3.1.3 Тестування бізнес-циклу 7

3.1.4 Тестування інтерфейсу користувача 8

3.1.5 Профілювання продуктивності 9

3.1.6 Тестування навантаження 10

3.1.7 Стрес-тестування 11

3.1.8 Тестування обсягу 12

3.1.9 Тестування безпеки та контролю доступу 13

3.1.10 Тестування після відмови та відновлення 14

3.1.11 Тестування конфігурації 16

3.1.12 Тестування встановлення 17

3.2 Інструменти 18

4. Ресурси 19

4.1 Ролі 19

4.2 Система 20

5. Основні етапи проекту 21

6. Результати 21

6.1 Тестова модель 21

6.2 Журнали тестування 21

6.3 Звіти про дефекти 21

Додаток А Завдання проекту 22

План тестування

# вступ

## Призначення

Цей документ плану тестування <Веб-Cервіс BookBag> підтримує такі цілі:

• Курсовий проект на тему Веб-Cервіс BookBag, реалізовано у виді веб інтерфейсу та БД для керування данними.

• нижче наведено кілька загальних прикладів рекомендованих вимог до тестування програмного продукту на високому рівні:

Повнота тестування: Програмний продукт повинен бути підданий всебічному тестуванню, що охоплює всі основні функціональність, сценарії використання та можливість відновлення після помилок.

Надійність: Програмний продукт повинен бути стабільним та надійним у різних умовах експлуатації та навантаження. Тестування повинно включати тестування на помилки, обробку виключних ситуацій, а також наявність механізмів резервного копіювання та відновлення даних.

Продуктивність: Програмний продукт повинен працювати ефективно та забезпечувати прийнятний рівень продуктивності. Тестування повинно включати перевірку швидкості, завантаження системи, реакції на запити користувачів та масштабованості продукту.

Сумісність: Програмний продукт повинен бути сумісним з різними операційними системами, пристроями та іншими програмними засобами, з якими він взаємодіє. Тестування повинно включати перевірку сумісності з різними конфігураціями та вихідними данними.

Безпека: Програмний продукт повинен мати високий рівень безпеки, захищений від несанкціонованого доступу, вразливостей та атак. Тестування повинно включати перевірку наявності заходів безпеки, шифрування даних, аутентифікації та авторизації.

Відповідність вимогам: Програмний продукт повинен повністю відповідати встановленим вимогам та специфікаціям. Тестування повинно включати перевірку відповідності функціональним, нефункціональним та технічним вимогам.

Легкість в використанні: Програмний продукт повинен бути зручним у використанні, зрозумілим для користувачів та мати інтуїтивний інтерфейс. Тестування повинно включати перевірку наявності документації, навчання користувачів та зручність роботи з продуктом.

• Стратегії тестування, які слід застосовувати для програмного проекта  
 1. Ідентифікація цілей тестування:

Визначення основних функціональних та нефункціональних вимог до програмного продукту.

Встановлення пріоритетів тестування, враховуючи критичність функцій та ризики проекту.

2.Визначення обсягу тестування:

Встановлення обсягу функціональних блоків або модулів, які підлягають тестуванню.

Определение рівнів тестування, таких як модульне тестування, інтеграційне тестування та системне тестування.

3.Розробка тестових сценаріїв:

Створення тестових сценаріїв, які покривають різні аспекти функціональності та використання програмного продукту.

Врахування позитивних та негативних сценаріїв, різних шляхів виконання та реакції на помилки.

4. Вибір методів тестування:

Встановлення підходящих методів тестування, таких як ручне тестування, автоматизоване тестування, навантажувальне тестування та інше.

Врахування можливостей і обмежень з використанням різних методів.

5. Планування тестових циклів:

Розподіл тестових завдань та встановлення термінів виконання.

Врахування ресурсів, необхідних для виконання тестів, включаючи людські ресурси, обладнання та інструменти.

6. Виконання тестів:

Запуск тестових сценаріїв згідно з планом.

Збір та аналіз результатів тестування, реєстрація дефектів та їх налагодження.

7.Повторне тестування та верифікація:

Перевірка виправлених дефектів та перетестування відповідних функцій.

Виконання повторного тестування для перевірки впроваджених змін та впливу на інші частини продукту.

8.Завершення тестування:

Оцінка покриття тестування та оцінка якості продукту.

Підготовка звіту про тестування та рекомендацій щодо випуску продукту.

## Обґрунтування

Необхідно протестувати фронтенд, можливості авторизації, кальцуляція даних, взаємодію Космосу з базою даних.

Особливу увагу приділити проведенню функціонального тестування та тестуванню безпеки

## Область застосування

Етапи тестування, ­⎯

Блочне тестування, також відоме як модульне тестування, є одним з рівнів тестування програмного продукту. Воно спрямоване на перевірку правильності роботи окремих модулів або компонентів програмного продукту незалежно від інших частин системи.

Основна мета блочного тестування - перевірити, чи виконують модулі програмного продукту очікувані функції та чи працюють вони коректно у межах заданих вимог. Для цього розробники створюють невеликі тестові сценарії, які виконуються з метою перевірки окремих блоків коду.

Деякі важливі аспекти блочного тестування:

Ізоляція модулів: Кожен модуль тестується окремо без залучення інших частин системи. Це дозволяє зосередитись на конкретних функціональних можливостях та виявити дефекти, які можуть впливати на роботу модуля.

Використання стабільних данних: Для тестування модулів часто використовуються фіксовані тестові дані, що дозволяє забезпечити стабільність результатів тестування та виявити проблеми, пов'язані з обробкою даних.

Моки та заглушки: У разі, якщо певні модулі залежать від інших, можуть використовуватися моки (mocks) або заглушки (stubs) для емуляції функціональності залежних компонентів. Це дозволяє провести тестування, навіть якщо інші модулі ще не готові.

Автоматизація: Блочне тестування може бути автоматизоване за допомогою спеціалізованих фреймворків і інструментів, що дозволяє легко виконувати тести та швидко отримувати результати.

В цілому, блочне тестування важливе для виявлення дефектів на ранніх етапах розробки та забезпечення правильності роботи окремих модулів перед інтеграцією всієї системи.,

**Інтеграційне тестування** є одним з рівнів тестування програмного продукту, яке спрямоване на перевірку взаємодії між різними компонентами, модулями або системами, що утворюють програмний продукт в цілому. Його ціль - виявити дефекти, які можуть виникати під час обміну даними, викликом функцій, інтеграцією компонентів та іншими аспектами, пов'язаними з взаємодією між частинами системи.

Основні аспекти інтеграційного тестування:

Планування тестування: Визначення стратегії інтеграційного тестування, включаючи порядок і послідовність інтеграції компонентів або модулів, а також визначення тестових сценаріїв.

Створення тестових середовищ: Підготовка необхідного інфраструктурного середовища для проведення інтеграційних тестів, включаючи налаштування серверів, баз даних, мережевих з'єднань тощо.

Документування тестових сценаріїв: Розробка детальних тестових сценаріїв, які охоплюють різні комбінації взаємодії між компонентами, передачу даних та обробку повідомлень.

Виконання тестів: Запуск інтеграційних тестів відповідно до розроблених тестових сценаріїв. Перевірка правильності взаємодії компонентів, передачу даних, обмін повідомленнями та інші аспекти, що впливають на цілісність і працездатність системи.

Виявлення та усунення дефектів: Реєстрація та відстеження дефектів, виявлених під час інтеграційного тестування. Взаємодія з розробниками та іншими членами команди для виправлення дефектів та проведення повторного тестування після виправлення.

Перевірка результів тестування: Оцінка результатів інтеграційного тестування, аналіз покриття тестування та оцінка якості взаємодії компонентів та системи в цілому Надайте короткий перелік функцій і функцій об’єкта тестування, які будуть або не будуть перевірені.

**РИЗИКИ**

Існує кілька ризиків і непередбачених обставин, які можуть впливати на дизайн, розробку або впровадження тестування програмного продукту. Деякі з них включають:

**Зміни вимог**: Неочікувані зміни вимог до програмного продукту можуть вплинути на дизайн тестових сценаріїв та необхідність їх змін.

**Обмежені ресурси:** Недостатня кількість ресурсів, таких як людські, фінансові або технічні, можуть обмежити можливості проведення повноцінного тестування.

**Зміни в графіку**: Зміни в графіку розробки або впровадження програмного продукту можуть впливати на планування тестування і змусити змінити пріоритети або обсяги тестових робіт.

**Недоступність середовищ**: Відсутність доступу до необхідних середовищ (наприклад, середовищ з реальними данними або реальними обмеженнями) може ускладнити проведення певних типів тестів.

**Залежності від інших команд**: Якщо тестування залежить від роботи інших команд або поставщиків, затримки або непередбачувані зміни з боку цих команд можуть вплинути на планування тестування.

**Непередбачувані проблеми**: Неочікувані технічні проблеми, помилки або непередбачувані обставини, такі як збої сервера або системи, можуть призвести до затримок у тестуванні або потребувати додаткових зусиль для їх вирішення.

**Конфлікти пріоритетів**: Відмінні пріоритети та вимоги між різними зацікавленими сторонами можуть призвести до незгоди щодо обсягу, пріоритетів та ресурсів, що призначені для тестування.

Обмеження, які можуть впливати на дизайн, розробку або впровадження тестування програмного продукту, включають наступні аспекти:

***Обмеження ресурсів:*** *Недостатність людських, фінансових або технічних ресурсів може обмежити обсяг тестування, тривалість або якість тестових робіт.*

***Обмеження часу:*** *Встановлені терміни для розробки та впровадження програмного продукту можуть вимагати обмеження часу, яке доступне для проведення тестування.*

***Обмеження доступу до середовищ****: Відсутність доступу до потрібних середовищ, таких як середовища реальних даних або середовища з обмеженими ресурсами, може ускладнити або обмежити проведення деяких типів тестів.*

***Обмеження доступу до функцій або даних****: Іноді певні функції або дані можуть бути обмеженими або недоступними для тестування, що може вплинути на покриття тестування або якість виявлених дефектів.*

***Обмеження сторонніх компоненті****в: Використання сторонніх компонентів або інтеграція зі сторонніми системами може створювати обмеження щодо доступності, налаштування або підтримки під час тестування.*

***Вимоги безпеки та конфіденційності:*** *Вимоги до безпеки і конфіденційності можуть обмежувати доступ до певних даних або обмежувати можливості тестування в реальних умовах.*

***Обмеження зовнішніх факторів****: Зовнішні фактори, такі як правові обмеження, регуляторні вимоги або обмеження ринку, можуть впливати на дизайн, розробку або впровадження тестування.*

## Ідентифікація проекту

У таблиці нижче вказано документацію та доступність, використану для розробки *плану тестування* :

[Примітка: видаліть або додайте відповідні елементи.]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Документ  (і версія / дата)** | **Створено або доступно** | **Отримано або переглянуто** | **Автор або ресурс** | **Примітки** |
| Специфікація вимог | 🞏так 🞏ні | 🞏так 🞏ні |  |  |
| Функціональна специфікація | 🞏так 🞏ні | 🞏так 🞏ні |  |  |
| Звіти про випадки використання | 🞏так 🞏ні | 🞏так 🞏ні |  |  |
| План проекту | 🞏так 🞏ні | 🞏так 🞏ні |  |  |
| Специфікації конструкції | 🞏так 🞏ні | 🞏так 🞏ні |  |  |
| Прототип | 🞏так 🞏ні | 🞏так 🞏ні |  |  |
| Посібники користувача | 🞏так 🞏ні | 🞏так 🞏ні |  |  |
| Бізнес-модель або потік | 🞏так 🞏ні | 🞏так 🞏ні |  |  |
| Модель даних або потік | 🞏так 🞏ні | 🞏так 🞏ні |  |  |
| Бізнес-функції та правила | 🞏так 🞏ні | 🞏так 🞏ні |  |  |
| Оцінка ризиків проекту або бізнесу | 🞏так 🞏ні | 🞏так 🞏ні |  |  |

# Вимоги до тесту

Наведений нижче список визначає варіанти ⎯використання елементів, функціональні вимоги та нефункціональні вимоги, ⎯які визначено як цілі для тестування. Цей список представляє те, що буде перевірено.

**Специфікація вимог:** Перед початком тестування необхідно мати чітку і повну специфікацію вимог до продукту. Це допоможе тестувальникам зрозуміти очікувану функціональність та властивості продукту.

**Тестова документація:** Створення тестової документації, такої як тест-плани і тест-кейси, є ключовим етапом. Тестова документація має містити опис тестових сценаріїв і критеріїв прийняття.

**Сценарії використання:** Розробка сценаріїв використання допоможе забезпечити тестування з реального використання продукту з точки зору кінцевих користувачів.

**Незалежність від розробки:** Тестувальники повинні мати доступ до продукту ізольовано від розробницької команди, щоб уникнути конфліктів і забезпечити об'єктивність.

**Тестове середовище:** Потрібно забезпечити наявність тестового середовища, яке відображає реальне середовище, де буде використовуватися продукт.

**Забезпечення якості коду:** Перед тестуванням високого рівня важливо переконатися, що код продукту відповідає стандартам якості і не має критичних помилок.

**Тестування на велику кількість користувачів:** Для перевірки масштабованості продукту слід провести тестування на велику кількість користувачів або симулювати таке навантаження.

**Тестування безпеки:** Проведення тестів на вразливість і безпеку продукту є необхідним для запобігання потенційним загрозам та атакам.

**Тестування відмовостійкості:** Важливо перевірити, як продукт веде себе в умовах непередбачених ситуацій та відмов.

**Тестування відповідності вимогам:** Кожен тест повинен бути спрямований на перевірку відповідності продукту вимогам і специфікаціям.

**Тестування взаємодії з іншими системами:** Якщо продукт взаємодіє з іншими програмними чи апаратними системами, важливо перевірити цю взаємодію.

**Тестування відмінності від конкурентів**: Якщо на ринку існують аналогічні продукти, важливо визначити, в чому саме ваш продукт відрізняється і як це впливає на його функціональність та відмінність.

**Тестування зручності використання**: Оцінка зручності і інтуїтивності інтерфейсу користувача є важливою частиною тестування високого рівня.

**Тестування продуктивності:** Перевірка продуктивності та швидкодії продукту в різних сценаріях використання.

Звітність: Оформлення звітів про результати тестування, виявлені помилки і рекомендації для виправлення.

Ці вимоги є загальними і можуть варіюватися залежно від конкретного продукту та його специфікацій.

# Стратегія тестування

[Стратегія тестування представляє рекомендований підхід до тестування об’єкта тестування. У попередньому розділі «Вимоги до тесту» було описано, що буде перевірятися, ⎯а також те, як тестуватиметься мета тесту.

Для кожного типу тесту надайте опис тесту та причини його впровадження та виконання.

Якщо певний тип тесту не буде реалізовано та виконано, вкажіть це в реченні, у якому зазначено, що тест не буде реалізовано чи виконано, і вкажіть обґрунтування, наприклад «Цей тест не буде реалізовано чи виконано. Цей тест не підходить».

Основними міркуваннями щодо стратегії тестування є методи, які будуть використовуватися, і критерій, за яким можна знати, коли тестування завершено.

На додаток до міркувань, наведених для кожного тесту нижче, тестування слід виконувати лише з використанням відомих контрольованих баз даних у захищеному середовищі. ]

## Типи тестування

### Перевірка цілісності даних і бази даних

[Бази даних і процеси бази даних слід перевіряти як підсистему в . Ці підсистеми слід тестувати без інтерфейсу користувача об’єкта тестування як інтерфейсу до даних. Необхідно провести додаткові дослідження системи керування базами даних (СУБД), щоб визначити інструменти та методи, які можуть існувати для підтримки тестування, зазначеного нижче.]

|  |  |
| --- | --- |
| Мета тесту: | [Переконайтеся, що методи доступу до бази даних і процеси функціонують належним чином і без пошкодження даних.] |
| Техніка: | •[Викликати кожен метод і процес доступу до бази даних, заповнюючи кожен дійсними та недійсними даними або запитами на дані.  • Перевірте базу даних, щоб переконатися, що дані були заповнені належним чином, усі події бази даних відбулися належним чином, або перегляньте повернуті дані, щоб переконатися, що правильні дані були отримані з правильних причин] |
| Критерії завершення: | [Усі методи доступу до бази даних і процеси функціонують відповідно до плану та без будь-яких пошкоджень даних.] |
| Особливі міркування: | • [Для тестування може знадобитися середовище розробки СУБД або драйвери для введення або зміни даних безпосередньо в базах даних.  •Процеси слід запускати вручну.  • Малі бази даних або бази даних мінімального розміру (обмежена кількість записів) слід використовувати для підвищення видимості будь-яких неприйнятних подій.] |

### Тестування функцій

[Тестування функції об’єкта тестування має зосереджуватися на будь-яких вимогах до тестування, які можна відстежити безпосередньо до випадків використання або бізнес-функцій і бізнес-правил. Цілі цих тестів — перевірити належне прийняття, обробку й пошук даних, а також належне впровадження бізнес-правил. Цей тип тестування базується на методах чорної скриньки; це перевірка програми та її внутрішніх процесів шляхом взаємодії з програмою через графічний інтерфейс користувача (GUI) та аналізу вихідних даних або результатів. Нижче наведено схему тестування, рекомендованого для кожного застосування:]

|  |  |
| --- | --- |
| Мета тесту: | [Забезпечте належну функціональність цільового тестування, включаючи навігацію, введення даних, обробку та пошук.] |
| Техніка: | [Виконайте кожен варіант використання, потік варіантів використання або функцію, використовуючи дійсні та недійсні дані, щоб перевірити таке:  • Очікувані результати досягаються, коли використовуються дійсні дані.  • У разі використання недійсних даних відображаються відповідні повідомлення про помилку або попередження.  • Кожне бізнес-правило застосовано належним чином.] |
| Критерії завершення: | • [Усі заплановані випробування виконано.  • Усі виявлені дефекти були усунені.] |
| Особливі міркування: | [Визначте або опишіть елементи або проблеми (внутрішні чи зовнішні), які впливають на впровадження та виконання функціонального тесту] |

### Тестування бізнес-циклу

[Тестування бізнес-циклу має імітувати дії, що виконуються в рамках проекту . Необхідно визначити період, наприклад один рік, і виконати транзакції та дії, які відбуватимуться протягом річного періоду. Це включає в себе всі щоденні, тижневі та місячні цикли, а також події, які залежать від дати, наприклад, лоскотання.]

|  |  |
| --- | --- |
| Мета тесту | [Забезпечте належне функціонування цільового тестування та фонових процесів відповідно до необхідних бізнес-моделей і графіків.] |
| Техніка: | [Тестування буде симулювати кілька бізнес-циклів, виконавши такі дії:  • Тести, які використовуються для тестування функції target-of-test, будуть змінені або вдосконалені, щоб збільшити кількість виконання кожної функції для імітації кількох різних користувачів протягом певного періоду.  • Усі функції, чутливі до часу або дати, виконуватимуться з використанням дійсних і недійсних дат або періодів часу.  • Усі функції, які відбуваються за періодичним розкладом, виконуватимуться або запускатимуться у відповідний час.  • Тестування включатиме використання дійсних і недійсних даних для перевірки наступного:  • Очікувані результати досягаються, коли використовуються дійсні дані.  • У разі використання недійсних даних відображаються відповідні повідомлення про помилку або попередження.  • Кожне бізнес-правило застосовано належним чином. |
| Критерії завершення: | • [Усі заплановані випробування виконано.  • Усі виявлені дефекти усунено.} |
| Особливі міркування: | • [Системні дати та події можуть потребувати спеціальної підтримки  • Бізнес-модель потрібна для визначення відповідних вимог до тестування та процедур.] |

### Тестування інтерфейсу користувача

[Тестування інтерфейсу користувача (UI) перевіряє взаємодію користувача з програмним забезпеченням. Мета тестування інтерфейсу користувача полягає в тому, щоб переконатися, що інтерфейс користувача надає користувачеві відповідний доступ і навігацію функціями об’єкта тестування. Крім того, тестування інтерфейсу користувача гарантує, що об’єкти в інтерфейсі користувача функціонують належним чином і відповідають корпоративним або галузевим стандартам.]

|  |  |
| --- | --- |
| Мета тесту: | [Перевірте наступне:  • Навігація через об’єкт тестування належним чином відображає бізнес-функції та вимоги, включаючи вікно до вікна, поле до поля та використання методів доступу (клавіші табуляції, рухи миші, клавіші прискорення)  • Об’єкти та характеристики вікна, такі як меню, розмір, положення, стан і фокус, відповідають стандартам.] |
| Техніка: | [Створіть або змініть тести для кожного вікна, щоб перевірити правильну навігацію та стани об’єктів для кожного вікна програми та об’єктів.] |
| Критерії завершення: | [Кожне вікно успішно перевірено на відповідність еталонній версії або прийнятному стандарту] |
| Особливі міркування: | [Не всі властивості користувацьких об’єктів і об’єктів сторонніх розробників доступні.] |

### Тестування продуктивності

[Тестування продуктивності — це перевірка продуктивності, під час якої вимірюються та оцінюються час відгуку, частота транзакцій та інші чутливі до часу вимоги. Метою профілювання продуктивності є перевірка виконання вимог до продуктивності. Профілювання продуктивності впроваджується та виконується, щоб профілювати та налаштовувати поведінку продуктивності об’єкта тестування в залежності від таких умов, як робоче навантаження або конфігурація обладнання.

Примітка. Транзакції нижче відносяться до «логічних бізнес-операцій». Ці транзакції визначаються як конкретні випадки використання, які суб’єкт системи повинен виконувати за допомогою цільового тесту, наприклад, додавати або змінювати певний контракт.]

|  |  |
| --- | --- |
| Мета тесту: | [Перевірте продуктивність для призначених транзакцій або бізнес-функцій за таких умов:  • нормальне очікуване навантаження  • очікуване навантаження в найгіршому випадку] |
| Техніка: | • [Використовуйте процедури тестування, розроблені для тестування функцій або бізнес-циклу.  • Змініть файли даних, щоб збільшити кількість транзакцій, або сценарії, щоб збільшити кількість ітерацій кожної транзакції.  • Сценарії слід запускати на одній машині (найкращий випадок для порівняння одного користувача, однієї транзакції) і повторювати з кількома клієнтами (віртуальними чи реальними, див. Особливі зауваження нижче).] |
| Критерії завершення: | • [Одна транзакція або один користувач: успішне завершення тестових сценаріїв без будь-яких збоїв і в межах очікуваного або необхідного розподілу часу на транзакцію.]  • [Кілька транзакцій або кілька користувачів: успішне виконання тестових сценаріїв без будь-яких збоїв і в межах прийнятного розподілу часу.] |
| Особливі міркування: | [Комплексне тестування продуктивності включає фонове навантаження на сервер.  Для цього можна використовувати кілька методів, зокрема:  • «Направляйте транзакції» безпосередньо на сервер, зазвичай у формі викликів мови структурованих запитів (SQL).  • Створіть «віртуальне» навантаження користувача, щоб імітувати багато клієнтів, зазвичай кілька сотень. Для виконання цього навантаження використовуються засоби емуляції віддаленого терміналу. Цей прийом також можна використовувати для завантаження мережі «трафіком».  • Використовуйте кілька фізичних клієнтів, кожен із яких запускає тестові сценарії, щоб навантажувати систему.  Тестування продуктивності має виконуватися на спеціальному комп’ютері або у визначений час. Це забезпечує повний контроль і точні вимірювання.  Бази даних, які використовуються для тестування продуктивності, мають мати фактичний розмір або однаково масштабовані.] |

### Тестування навантаження

[Навантажувальне тестування — це перевірка продуктивності, яка піддає об’єкт тесту різним робочим навантаженням, щоб виміряти й оцінити поведінку продуктивності та здатність об’єкта тестування продовжувати належне функціонування за цих різних навантажень. Мета навантажувального тестування — визначити та переконатися, що система функціонує належним чином поза очікуваним максимальним навантаженням. Крім того, навантажувальне тестування оцінює характеристики продуктивності, такі як час відгуку, швидкість транзакцій та інші чутливі до часу проблеми).]

[Примітка: наведені нижче операції відносяться до «логічних бізнес-операцій». Ці транзакції визначаються як конкретні функції, які кінцевий користувач системи повинен виконувати за допомогою програми, наприклад додавати або змінювати даний договір.]

|  |  |
| --- | --- |
| Мета тесту: | [Перевірте час поведінки продуктивності для визначених транзакцій або бізнес-випадків за різних умов робочого навантаження.] |
| Техніка: | • [Використовуйте тести, розроблені для тестування функцій або бізнес-циклу.  • Змініть файли даних, щоб збільшити кількість транзакцій, або тести, щоб збільшити кількість разів кожної транзакції.] |
| Критерії завершення: | [Кілька транзакцій або кілька користувачів: успішне завершення тестів без будь-яких збоїв і в межах прийнятного розподілу часу.] |
| Особливі міркування: | • [Навантажувальне тестування має виконуватися на спеціальному комп’ютері або у визначений час. Це забезпечує повний контроль і точні вимірювання.  • Бази даних, які використовуються для навантажувального тестування, повинні мати фактичний розмір або однаково масштабовані.] |

### Стрес-тестування

[Стрес-тестування – це тип перевірки продуктивності, який реалізується та виконується для виявлення помилок через брак ресурсів або конкуренцію за ресурси. Нестача пам’яті або місця на диску може виявити дефекти в об’єкті тестування, які не помітні за нормальних умов. Інші дефекти можуть бути результатом конкуренції за спільні ресурси, як-от блокування бази даних або пропускна здатність мережі. Стрес-тестування також можна використовувати для визначення максимального робочого навантаження, яке може впоратися з об’єктом тестування.]

[Примітка: посилання на транзакції нижче стосуються логічних бізнес-транзакцій.]

|  |  |
| --- | --- |
| Мета тесту: | [Перевірте, що об’єкт тестування функціонує належним чином і без помилок за наступних умов стресу:  • мало або зовсім немає пам'яті на сервері (RAM і DASD)  • максимальна фактична або фізично дієва кількість клієнтів, підключених або імітованих  • кілька користувачів, які виконують однакові операції з однаковими даними або обліковими записами  • у найгіршому випадку обсяг або суміш транзакцій (див. Тестування продуктивності вище).  Примітки. Метою стрес-тестування також може бути визначено та задокументовано умови, за яких система НЕ продовжить функціонувати належним чином.  Стрес-тестування клієнта описано в розділі 3.1.11, Тестування конфігурації.] |
| Техніка: | • [Використовуйте тести, розроблені для профілювання продуктивності або тестування навантаження.  • Щоб перевірити обмежені ресурси, тести слід запускати на одній машині, а RAM і DASD на сервері слід зменшити або обмежити.  • Для решти стрес-тестів слід використовувати кілька клієнтів, які виконують ті самі тести або додаткові тести, щоб отримати найгірший обсяг транзакцій або суміш. |
| Критерії завершення: | [Усі заплановані тести виконуються, і визначені обмеження системи досягаються або перевищуються без збою програмного забезпечення або умов, за яких відбувається збій системи, виходять за межі вказаних умов.] |
| Особливі міркування: | • [Напруження мережі може вимагати мережевих інструментів для завантаження мережі повідомленнями або пакетами.  • DASD, що використовується для системи, має бути тимчасово зменшено, щоб обмежити доступний простір для зростання бази даних.  • Синхронізація одночасного доступу клієнтів до тих самих записів або облікових записів даних.] |

### Volume Test

[Тестування об’єму нагружае об’єкт тестування великим обсягам даних, щоб визначити, чи досягнуто обмежень, які призводять до збою програмного забезпечення. Тестування об’єму також визначає безперервне максимальне навантаження або об’єм, який об’єкт тестування може впоратися протягом певного періоду. Наприклад, якщо об’єкт тестування обробляє набір записів бази даних для створення звіту, Volume Test використовуватиме велику тестову базу даних і перевірить, чи програмне забезпечення працює нормально та створює правильний звіт.]

|  |  |
| --- | --- |
| Мета тесту: | [Перевірте, чи мета перевірки успішно функціонує в таких сценаріях великого обсягу:  • Максимальна (фактична або фізично дієздатна) кількість клієнтів, підключених або змодельованих, які виконують ту саму бізнес-функцію в найгіршому випадку (продуктивність) протягом тривалого періоду.  • Було досягнуто максимального розміру бази даних (фактичного чи масштабованого), і кілька запитів або транзакцій звіту виконуються одночасно.] |
| Техніка: | • [Використовуйте тести, розроблені для профілювання продуктивності або тестування навантаження.  • Слід використовувати кілька клієнтів, або запускаючи ті самі тести, або додаткові тести, щоб отримати найгірший обсяг транзакцій або суміш (див. Стрес-тестування вище) протягом тривалого періоду.  • Створюється максимальний розмір бази даних (фактичний, масштабований або заповнений репрезентативними даними), і кілька клієнтів використовуються для виконання запитів і звітів про транзакції одночасно протягом тривалих періодів.] |
| Критерії завершення: | • [Усі заплановані тести виконано, і визначені системні обмеження досягнуто або перевищено без збою програмного забезпечення чи програмного забезпечення.] |
| Особливі міркування: | [Який період часу вважатиметься прийнятним для умов великого обсягу, як зазначено вище?] |

### Тестування безпеки та контролю доступу

[Тестування безпеки та контролю доступу фокусується на двох ключових сферах безпеки:

• Безпека на рівні програми, включаючи доступ до Даних або бізнес-функцій

• Безпека на системному рівні, включаючи вхід або віддалений доступ до системи.

Безпека на рівні програми гарантує, що на основі бажаної безпеки суб’єкти обмежені певними функціями чи випадками використання або обмежені в даних, які їм доступні. Наприклад, кожному може бути дозволено вводити дані та створювати нові облікові записи, але лише менеджери можуть їх видаляти. Якщо існує безпека на рівні даних, тестування гарантує, що «користувач першого типу» може бачити всю інформацію про клієнта, включаючи фінансові дані, однак «користувач два» бачить лише демографічні дані для того самого клієнта.

Безпека на системному рівні гарантує, що лише ті користувачі, яким надано доступ до системи, зможуть отримати доступ до програм і лише через відповідні шлюзи.]

|  |  |
| --- | --- |
| Мета тесту: | * Безпека на рівні програми: [ *Переконайтеся, що актор може отримати доступ лише до тих функцій або даних, для яких його тип користувача має дозволи.]* * Безпека на рівні системи: *Переконайтеся, що доступ до них мають лише ті учасники, які мають доступ до системи та програм* .] |
| Техніка: | * Безпека на рівні програми: [ *Визначте та перелічіть кожен тип користувача та функції або дані, для яких кожен тип має дозволи.]*   • [Створіть тести для кожного типу користувача та перевірте кожен дозвіл, створивши транзакції, специфічні для кожного типу користувача.]  • Змініть тип користувача та повторно запустіть тести для тих самих користувачів. У кожному разі перевірте, чи ці додаткові функції чи дані належним чином доступні чи заборонені.   * Доступ на системному рівні: *[Див. Особливі зауваження нижче]* |
| Критерії завершення: | [Для кожного відомого типу актора доступна відповідна функція або дані, і всі транзакції функціонують належним чином і виконуються в попередніх тестах функцій програми.] |
| Особливі міркування: | [Доступ до системи необхідно переглянути або обговорити з відповідним адміністратором мережі чи системи. Це тестування може не знадобитися, оскільки воно може бути функцією адміністрування мережі чи системи.] |

### Тестування відмов і відновлення

[Тестування відмовостійкості та відновлення гарантує, що об’єкт тестування зможе успішно відновити відмову та відновити роботу після різноманітних збоїв у роботі обладнання, програмного забезпечення чи мережі з необґрунтованою втратою даних або цілісності даних.

Тестування відмов гарантує, що для тих систем, які повинні продовжувати працювати, коли виникає умова відмов, альтернативні або резервні системи належним чином «приймуть на себе» несправну систему без втрати даних або транзакцій.

Тестування відновлення — це антагоністичний процес тестування, під час якого програма або система піддається екстремальним умовам або симульованим умовам, щоб спричинити збій, наприклад збої вводу/виводу (I/O) пристрою або недійсні покажчики та ключі бази даних. Викликаються процеси відновлення, і програма або система відстежуються та перевіряються, щоб перевірити належність програми або системи та відновлення даних.]

|  |  |
| --- | --- |
| Мета тесту: | [Переконайтеся, що процеси відновлення (вручну чи автоматизовані) належним чином відновлюють базу даних, програми та систему до бажаного відомого стану. Наступні типи умов повинні бути включені в тестування:  • припинення живлення клієнта  • припинення живлення сервера  • переривання зв'язку через сервери мережі  • переривання, зв’язок або втрата живлення DASD та/або контролерів DASD  • неповні цикли (переривання процесів фільтрації даних, переривання процесів синхронізації даних).  • недійсний покажчик або ключі бази даних  • недійсний або пошкоджений елемент даних у базі даних] |
| Техніка: | [Тести, створені для тестування функцій і бізнес-циклів, слід використовувати для створення серії транзакцій. Після досягнення бажаної початкової точки тестування слід виконати або змоделювати наступні дії окремо:  • Перерва живлення клієнта: вимкніть ПК.  • Переривання живлення сервера: імітація або ініціювання процедур вимкнення живлення сервера.  • Переривання через мережеві сервери: імітуйте або ініціюйте втрату зв’язку з мережею (фізично від’єднайте комунікаційні дроти або вимкніть мережеві сервери чи маршрутизатори.  • Переривання, зв’язок або втрата живлення DASD і контролерів DASD: симулюйте або фізично усуньте зв’язок з одним або кількома контролерами або пристроями DASD.  Після досягнення зазначених вище умов або змодельованих умов слід виконати додаткові транзакції, а після досягнення цього стану другої контрольної точки слід запустити процедури відновлення.  Тестування на неповні цикли використовує ту саму техніку, що описана вище, за винятком того, що самі процеси бази даних повинні бути перервані або передчасно припинені.  Перевірка наступних умов вимагає досягнення відомого стану бази даних. Кілька полів бази даних, покажчиків і ключів слід пошкодити вручну та безпосередньо в базі даних (за допомогою засобів бази даних). Додаткові транзакції слід виконувати за допомогою тестів із тестування функцій додатків і бізнес-циклів, а також виконувати повні цикли.] |
| Критерії завершення: | [У всіх вищезазначених випадках програма, база даних і система повинні після завершення процедур відновлення повернутися до відомого, бажаного стану. Цей стан включає пошкодження даних, обмежене відомими пошкодженими полями, покажчиками або ключами, а також звіти, що вказують на процеси або транзакції, які не були завершені через переривання.] |
| Особливі міркування: | • [Тестування відновлення дуже нав'язливе. Процедури від’єднання кабелю (імітація втрати живлення або зв’язку) можуть бути небажаними або здійсненними. Можуть знадобитися альтернативні методи, такі як інструменти діагностичного програмного забезпечення.  • Потрібні ресурси з груп «Системи» (або «Комп’ютерні операції»), «База даних» і «Мережа».  • Ці тести слід проводити в неробочий час або на ізольованій машині.] |

### Тестування конфігурації

[Тестування конфігурації перевіряє роботу об’єкта тестування на різних конфігураціях програмного та апаратного забезпечення. У більшості виробничих середовищ специфікації апаратного забезпечення для клієнтських робочих станцій, мережевих підключень і серверів баз даних відрізняються. На клієнтських робочих станціях може бути завантажено різне програмне забезпечення, ⎯наприклад програми, драйвери тощо, і ⎯в будь-який момент часу може бути активним багато різних комбінацій, які використовують різні ресурси.]

|  |  |
| --- | --- |
| Мета тесту: | [Перевірте, чи об’єкт перевірки функціонує належним чином на необхідних апаратних і програмних конфігураціях.] |
| Техніка: | • [Використовуйте сценарії перевірки функцій.  • Відкривайте та закривайте різноманітне програмне забезпечення, не пов’язане з тестом, як-от програми Microsoft, Excel і Word, як частину тесту або перед початком тесту.  • Виконайте вибрані транзакції, щоб імітувати взаємодію актора з цільовим і нецільовим програмним забезпеченням.  • Повторіть описаний вище процес, мінімізуючи доступну звичайну пам’ять на робочій станції клієнта.] |
| Критерії завершення: | [Для кожної комбінації цільового та нецільового програмного забезпечення, усі транзакції успішно завершуються без збоїв.] |
| Особливі міркування: | • [Яке програмне забезпечення, яке не є цільовим для тестування, потрібне, доступне та доступне на робочому столі?  • Які програми зазвичай використовуються?  • Які дані запускають програми; наприклад, велика електронна таблиця, відкрита в Excel, або 100-сторінковий документ у Word?  • Усі системи, мережеве програмне забезпечення, мережеві сервери, бази даних тощо також мають бути задокументовані як частина цього тесту.] |

### Тестування установки

[Тестування встановлення має дві мети. По-перше, гарантувати, що програмне забезпечення можна інсталювати за різних умов, ⎯таких як нова інсталяція, оновлення та повна або спеціальна інсталяція ⎯за нормальних і нестандартних умов. До ненормальних умов належать недостатньо місця на диску, відсутність прав на створення каталогів тощо. Друга мета — переконатися, що після встановлення програмне забезпечення працює правильно. Зазвичай це означає виконання кількох тестів, розроблених для функціонального тестування.]

|  |  |
| --- | --- |
| Мета тесту: | Переконайтеся, що мета тестування належним чином інсталюється на кожну необхідну конфігурацію обладнання за таких умов:  • нова інсталяція, нова машина, яка раніше не встановлювалася  • оновлення, попередньо встановлена машина , та сама версія  • оновлення, машина попередньо встановлена , старіша версія |
| Техніка: | • [Вручну або розробити автоматизовані сценарії, щоб підтвердити стан цільової машини ⎯новий - ніколи не встановлювався; та сама версія або старіша версія вже встановлена).  • Запустіть або виконайте установку.  • Запустіть транзакції за допомогою попередньо визначеного піднабору сценаріїв функціонального тестування.] |
| Критерії завершення: | транзакції виконуються успішно без збоїв. |
| Особливі міркування: | [Які транзакції слід вибрати для перевірки впевненості в тому, що програму успішно встановлено та відсутні основні компоненти програмного забезпечення?] |

## Інструменти

Для цього проекту будуть використані такі інструменти:

[Примітка: видаліть або додайте відповідні елементи.]

|  |  |  |  |
| --- | --- | --- | --- |
|  | Інструмент | Постачальник/власний | Версія |
| Управління тестами | Test Rail | Gurock Software |  |
| Відстеження дефектів | Jira, Bugzilla | Atlassian |  |
| Інструмент ASQ для функціонального тестування | Selenium, Test Copmlete |  |  |
| Інструмент ASQ для тестування продуктивності | Apache JMeter: LoadRunner | Micro Focus, |  |
| Тестовий моніторинг покриття або профайлер | Test Rail | Gurock Software |  |
| Управління проектами | Jira, Confluence, Trello | Atlassian |  |
| Засоби СУБД | MySQL, Oracle |  |  |

# Ресурси

У цьому розділі представлено рекомендовані ресурси для проекту Веб-Cервіс BookBag, їхні основні обов’язки та набір знань чи навичок.

## Ролі

У цій таблиці показано кадрові припущення для проекту.

[ПРИМІТКА: видаліть або додайте відповідні елементи.]

|  |  |  |
| --- | --- | --- |
| Людські ресурси | | |
| робітник |  |  |
| Менеджер проекту |  |  |
| Розробник/ки |  |  |
| QA тестер |  |  |

## система

У наведеній нижче таблиці наведено системні ресурси для проекту тестування.

[Конкретні елементи тестової системи наразі не повністю відомі. Рекомендується, щоб система імітувала виробниче середовище, зменшуючи доступи та розміри бази даних, якщо і де це необхідно.]

[Примітка: видаліть або додайте відповідні елементи.]

|  |  |
| --- | --- |
| Системні ресурси | |
| Ресурс | Назва / Тип |
| Сервер бази даних |  |
| — Мережа або підмережа | Уточнюється |
| — Ім’я сервера | Уточнюється |
| — Назва бази даних | Уточнюється |
| Клієнтські тестові ПК |  |
| — Включіть спеціальні вимоги до конфігурації | Уточнюється |
| Репозиторій тестів |  |
| — Мережа або підмережа | Уточнюється |
| — Ім’я сервера | Уточнюється |
| Розробка тестів ПК | Уточнюється |

# Віхи тествуання

[Тестування має включати тестові дії для кожного з тестових зусиль, визначених у попередніх розділах. Необхідно визначити окремі етапи проекту, щоб повідомити про досягнення в статусі проекту.]

|  |  |  |  |
| --- | --- | --- | --- |
| **Етапне завдання** | **Зусилля** | **Дата початку** | **Дата закінчення** |
| Планування Тестування |  |  |  |
| Проектування процесу тестування |  |  |  |
| Реалізація Тестування |  |  |  |
| Виконати Тестування |  |  |  |
| Оцінювання Тестування |  |  |  |

# Результати

[У цьому розділі перелічіть різні документи, інструменти та звіти, які будуть створені, ким, кому доставлено та коли буде доставлено.]

## Тестова модель

[Цей розділ визначає звіти, які будуть створені та розповсюджені на основі тестової моделі. Ці артефакти в тестовій моделі потрібно створити або посилатися в інструментах ASQ.]

## Тест лог

[Опишіть метод та інструменти, що використовуються для запису та звітування про результати тестування та статус тестування.]

## Звіти про дефекти

[У цьому розділі визначте метод і інструменти, які використовуються для запису, відстеження та звітування про інциденти тестування та їх статус.]

# Додаток А Завдання тестування

Нижче вказані завдання, пов’язані з тестуванням:

• Тест план

* + визначити вимоги до тесту
  + оцінити ризик
  + розробити тестову стратегію
  + визначити тестові ресурси
  + створити розклад
  + створити план тестування

• Тест дизайну

- підготувати аналіз навантаження

- визначити та описати тестові випадки

- ідентифікувати та структурувати тестові процедури

- переглянути та оцінити тестове покриття

• Реалізувати тест

* + записувати або програмувати тестові сценарії
  + визначити специфічну для тесту функціональність у моделі проектування та реалізації
  + створити зовнішні набори даних

• Виконати тест

-виконати тестові процедури

- оцінити виконання тесту

-відновлення після зупиненого тесту

- перевірити результати

-дослідити несподівані результати

-дефекти журналу

• Оцініть тест

-оцінити охоплення тестових випадків

- оцінити покриття коду

- аналізувати дефекти

- визначити, чи досягнуто критеріїв завершення тесту та критеріїв успіху