Тест-план для веб-приложения "Информационная справочная система библиотеки"

1. Введение

1.1 Цель

Цель данного тест-плана - определить объем и подход к тестированию веб-приложения "Информационная справочная система библиотеки". Тест-план помогает определить, что нужно тестировать и какие состояния можно считать успешным завершением тестов.

1.2 Описание приложения

"Информационная справочная система библиотеки" - это веб-приложение, разработанное на языке Java, предоставляющее пользователям доступ к библиотечному каталогу, возможность бронирования книг, а также управление пользовательскими учетными записями и администрирование библиотеки.

2. Область применения

2.1 Функциональные области тестирования

- Регистрация и авторизация пользователей

- Поиск и просмотр каталога книг

- Бронирование книг

- Управление учетными записями пользователей (личный кабинет)

- Административные функции (добавление, удаление, редактирование книг и пользователей)

- Уведомления и напоминания

2.2 Нефункциональные области тестирования

- Производительность (время отклика, нагрузочное тестирование)

- Безопасность (аутентификация, авторизация, защита данных)

- Юзабилити (удобство использования интерфейса)

- Совместимость (браузеры, платформы)

3. Среда тестирования

3.1 Платформы и операционные системы

- Windows 10,7,XP

3.2 Браузеры

- Google Chrome (последние три версии)

- Mozilla Firefox (последние три версии)

- Microsoft Edge (последние три версии)

- Safari (последние три версии)

4. Критерии завершения тестов

4.1 Критерии для функционального тестирования

- Все функциональные тесты успешно пройдены

- Все критические и высокоприоритетные дефекты устранены

- Количество средних и низкоприоритетных дефектов в пределах допустимых значений (не более 5% от общего числа)

4.2 Критерии для нефункционального тестирования

- Система успешно выдерживает нагрузку, определенную в требованиях (например, 1000 одновременных пользователей)

- Время отклика системы не превышает 3 секунд при нормальной нагрузке

- Отсутствие критических уязвимостей безопасности

- Положительные отзывы о юзабилити от тестировщиков

4.3 Общие критерии завершения

- Все виды тестирования завершены согласно плану

- Сформированы и переданы все отчеты о тестировании

- Получено одобрение заинтересованных сторон о готовности системы к релизу

5. Подход к тестированию

5.1 Фазы тестирования

- Планирование тестирования

- Разработка тестовых случаев

- Настройка среды тестирования

- Проведение функционального тестирования

- Проведение нефункционального тестирования

- Регрессионное тестирование

- Завершение тестирования и подведение итогов

5.2 Типы тестов

- Модульное тестирование

- Интеграционное тестирование

- Системное тестирование

- Приемочное тестирование

- Нагрузочное тестирование

- Тестирование безопасности

- Юзабилити тестирование

6. Роли и обязанности

- Тест-менеджер: Разработка тест-плана, координация команды, отчетность.

- Тестировщик: Разработка и выполнение тестов, документирование результатов.

- Разработчик: Исправление обнаруженных дефектов, участие в интеграционном тестировании.

- Аналитик по бизнесу: Проверка соответствия системы требованиям.

- Представитель заказчика: Приемочное тестирование, утверждение готовности к релизу.

Тест-кейсы для веб-приложения "Информационная справочная система библиотеки"

1. Регистрация пользователя

1.1. Позитивный сценарий: Успешная регистрация нового пользователя

Предусловия:

Открыта страница регистрации.

Шаги:

Ввести в поле "Имя пользователя" уникальное имя (например, testuser).

Ввести в поле "Электронная почта" действующий адрес электронной почты (например, testuser@example.com).

Ввести в поле "Пароль" корректный пароль (например, Password123!).

Повторить пароль в поле "Подтверждение пароля".

Нажать кнопку "Зарегистрироваться".

Ожидаемый результат:

Появляется сообщение об успешной регистрации.

Пользователь перенаправляется на страницу входа.

1.2. Негативный сценарий: Регистрация с уже существующим именем пользователя

Предусловия:

Открыта страница регистрации.

В системе уже существует пользователь с именем existinguser.

Шаги:

Ввести в поле "Имя пользователя" имя existinguser.

Ввести в поле "Электронная почта" действующий адрес электронной почты.

Ввести в поле "Пароль" корректный пароль.

Повторить пароль в поле "Подтверждение пароля".

Нажать кнопку "Зарегистрироваться".

Ожидаемый результат:

Появляется сообщение об ошибке "Имя пользователя уже существует".

2. Авторизация пользователя

2.1. Позитивный сценарий: Успешный вход в систему

Предусловия:

Открыта страница входа.

Существующий пользователь testuser с паролем Password123!.

Шаги:

Ввести в поле "Имя пользователя" testuser.

Ввести в поле "Пароль" Password123!.

Нажать кнопку "Войти".

Ожидаемый результат:

Пользователь перенаправляется на главную страницу системы.

2.2. Негативный сценарий: Вход с неверным паролем

Предусловия:

Открыта страница входа.

Существующий пользователь testuser.

Шаги:

Ввести в поле "Имя пользователя" testuser.

Ввести в поле "Пароль" WrongPassword.

Нажать кнопку "Войти".

Ожидаемый результат:

Появляется сообщение об ошибке "Неверное имя пользователя или пароль".

3. Поиск книги в каталоге

3.1. Позитивный сценарий: Поиск существующей книги по названию

Предусловия:

Пользователь авторизован.

В каталоге имеется книга с названием "Effective Java".

Шаги:

Ввести в поисковую строку "Effective Java".

Нажать кнопку "Поиск".

Ожидаемый результат:

В результатах поиска отображается книга "Effective Java".

3.2. Негативный сценарий: Поиск несуществующей книги

Предусловия:

Пользователь авторизован.

Шаги:

Ввести в поисковую строку "Nonexistent Book".

Нажать кнопку "Поиск".

Ожидаемый результат:

Появляется сообщение "Книга не найдена".  
  
Тест-кейс 1: Регистрация нового пользователя  
  
import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.Assert;

import org.testng.annotations.AfterClass;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

public class LibrarySystemTests {

private WebDriver driver;

@BeforeClass

public void setUp() {

// Указываем путь к драйверу Chrome

System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");

driver = new ChromeDriver();

}

@Test

public void testUserRegistration() {

// Открываем страницу регистрации

driver.get("http://localhost:8080/register");

// Заполняем форму регистрации

WebElement usernameField = driver.findElement(By.id("username"));

WebElement passwordField = driver.findElement(By.id("password"));

WebElement confirmPasswordField = driver.findElement(By.id("confirmPassword"));

WebElement emailField = driver.findElement(By.id("email"));

WebElement registerButton = driver.findElement(By.id("register"));

usernameField.sendKeys("newuser");

passwordField.sendKeys("password123");

confirmPasswordField.sendKeys("password123");

emailField.sendKeys("newuser@example.com");

// Отправляем форму

registerButton.click();

// Проверяем успешную регистрацию

WebElement successMessage = driver.findElement(By.id("successMessage"));

Assert.assertTrue(successMessage.isDisplayed(), "Регистрация не удалась.");

}

@AfterClass

public void tearDown() {

if (driver != null) {

driver.quit();

}

}

}  
  
Тест-кейс 2: Авторизация пользователя  
  
import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.Assert;

import org.testng.annotations.AfterClass;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

public class LibrarySystemTests {

private WebDriver driver;

@BeforeClass

public void setUp() {

// Указываем путь к драйверу Chrome

System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");

driver = new ChromeDriver();

}

@Test

public void testUserLogin() {

// Открываем страницу авторизации

driver.get("http://localhost:8080/login");

// Заполняем форму авторизации

WebElement usernameField = driver.findElement(By.id("username"));

WebElement passwordField = driver.findElement(By.id("password"));

WebElement loginButton = driver.findElement(By.id("login"));

usernameField.sendKeys("newuser");

passwordField.sendKeys("password123");

// Отправляем форму

loginButton.click();

// Проверяем успешную авторизацию

WebElement dashboardElement = driver.findElement(By.id("dashboard"));

Assert.assertTrue(dashboardElement.isDisplayed(), "Авторизация не удалась.");

}

@AfterClass

public void tearDown() {

if (driver != null) {

driver.quit();

}

}

}