Задание 1.

Стратегия тестирования web интернет-магазина

Введение

Данный документ описывает стратегию тестирования web интернет-магазина, разрабатываемого в течение 3 месяцев с последующим месяцем поддержки. Стратегия основана на принципах пирамиды тестирования, учитывает различные виды и уровни тестирования, а также включает тест-дизайн, тестовую документацию, метрики качества, оценку трудоемкости, управление рисками и анализ требований.

Этапы разработки и тестирования (сроки и действия)

Этап 1: Анализ требований и планирование тестирования (Неделя 1-2)

* Анализ требований к функциональности, производительности, безопасности и удобству использования.
* Определение видов и уровней тестирования.
* Разработка плана тестирования с указанием сроков, ресурсов и ожидаемых результатов.
* Подготовка тестовой документации (чек-листы, тест-кейсы, баг-репорты).
* Оценка трудоемкости тестирования.
* Идентификация и анализ рисков.

Этап 2: Тестирование альфа-версии (Неделя 3-4)

* Фокус: Дымовое тестирование, тестирование основных функций (просмотр каталога, добавление в корзину, оформление заказа, выбор доставки, оплата).
* Методы: Исследовательское тестирование, тестирование по чек-листам.
* Цель: Выявить критические ошибки, блокирующие дальнейшую разработку.

Этап 3: Разработка и тестирование основных функций (Неделя 5-10)

* Фокус: Функциональное тестирование, интеграционное тестирование, тестирование API (если есть).
* Методы: Функциональное тестирование по тест-кейсам, интеграционное тестирование компонентов системы.
* Цель: Проверить корректность работы всех функций, взаимодействие между модулями, соответствие требованиям.

Этап 4: Тестирование производительности и безопасности (Неделя 11-12)

* Фокус: Нагрузочное тестирование, стресс-тестирование, тестирование безопасности (аутентификация, авторизация, защита от SQL-инъекций и XSS-атак).
* Инструменты: JMeter, OWASP ZAP (или аналоги).
* Цель: Убедиться, что приложение стабильно работает под нагрузкой, устойчиво к высоким нагрузкам и безопасно.

Этап 5: Тестирование удобства использования (Неделя 13-14)

* Фокус: Юзабилити-тестирование, тестирование кроссбраузерной совместимости.
* Методы: Опросы пользователей, A/B-тестирование.
* Цель: Проверить удобство использования приложения для целевой аудитории, корректное отображение в различных браузерах.

Этап 6: Регрессионное тестирование и подготовка к релизу (Неделя 15-16)

* Фокус: Повторное тестирование исправленных ошибок и измененного функционала, проверка стабильности приложения перед релизом.
* Методы: Автоматизированное регрессионное тестирование (Selenium, Cypress, или другие).
* Цель: Убедиться, что исправления ошибок не привели к появлению новых дефектов.

Этап 7: Поддержка после релиза (Месяц 4)

* Фокус: Мониторинг работы приложения, исправление ошибок, обнаруженных пользователями.
* Методы: Анализ логов, обратная связь от пользователей.
* Цель: Обеспечить стабильную работу приложения после релиза, быстро реагировать на проблемы.

Ресурсы

* Внутренние: Команда тестирования, документация по проекту.
* Внешние: Инструменты для автоматизированного тестирования (Selenium, Cypress), инструменты для нагрузочного тестирования (JMeter), инструменты для тестирования безопасности (OWASP ZAP).

Метрики качества

* Количество найденных и исправленных ошибок.
* Покрытие кода тестами.
* Время выполнения тестов.
* Удовлетворенность пользователей.
* Время отклика приложения.

Управление рисками

* Риск: Недостаточное время на тестирование.
* Решение: Приоритизация задач, автоматизация тестирования.
* Риск: Неполные или некорректные требования.
* Решение: Тщательный анализ требований, общение с заказчиком.

Заключение

Данная стратегия позволит эффективно протестировать web интернет-магазин, выявить и исправить ошибки на ранних этапах разработки, обеспечить высокое качество продукта и удовлетворенность пользователей. Регулярный мониторинг метрик качества и управление рисками помогут контролировать процесс тестирования и достичь поставленных целей.

Задание 2

1. Создание базы данных и таблиц

CREATE DATABASE academy;

c academy;

CREATE TABLE Students (

s\_id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

start\_year INT CHECK (start\_year > 2000)

);

CREATE TABLE Courses (

c\_no SERIAL PRIMARY KEY,

title VARCHAR(100) NOT NULL,

hours INT CHECK (hours > 0)

);

CREATE TABLE Exams (

s\_id INT REFERENCES Students(s\_id) ON DELETE CASCADE,

c\_no INT REFERENCES Courses(c\_no) ON DELETE CASCADE,

score DECIMAL(5, 2) CHECK (score >= 0 AND score <= 100),

PRIMARY KEY (s\_id, c\_no)

);

2. Вставка записей в таблицы

INSERT INTO Students (name, start\_year) VALUES

('Alice Johnson', 2021),

('Bob Smith', 2020),

('Charlie Brown', 2019),

('Diana Prince', 2022);

INSERT INTO Courses (title, hours) VALUES

('Mathematics', 40),

('Physics', 30),

('Chemistry', 35),

('Biology', 25);

INSERT INTO Exams (s\_id, c\_no, score) VALUES

(1, 1, 85.00), -- Alice in Mathematics

(1, 2, 90.50), -- Alice in Physics

(2, 1, 75.00), -- Bob in Mathematics

(3, 3, 65.00); -- Charlie in Chemistry

3. Запрос для получения студентов, которые еще не сдали ни одного экзамена

SELECT \* FROM Students

WHERE s\_id NOT IN (SELECT s\_id FROM Exams);

4. Запрос для получения списка студентов и количества сданных им экзаменов

SELECT s.name, COUNT(e.c\_no) AS exam\_count

FROM Students s

JOIN Exams e ON s.s\_id = e.s\_id

GROUP BY s.s\_id

HAVING COUNT(e.c\_no) > 0;

5. Запрос для вывода списка курсов со средним баллом по экзамену

SELECT c.title, AVG(e.score) AS average\_score

FROM Courses c

JOIN Exams e ON c.c\_no = e.c\_no

GROUP BY c.c\_no

ORDER BY average\_score DESC;

6. Скрипт для генерации произвольных данных

import random

import psycopg2

# Подключение к базе данных

conn = psycopg2.connect("dbname=academy user=your\_username password=your\_password")

cur = conn.cursor()

# Генерация студентов

for i in range(10):

name = f'Student {i + 1}'

start\_year = random.randint(2018, 2023)

cur.execute("INSERT INTO Students (name, start\_year) VALUES (%s, %s)", (name, start\_year))

# Генерация курсов

for i in range(5):

title = f'Course {i + 1}'

hours = random.randint(20, 60)

cur.execute("INSERT INTO Courses (title, hours) VALUES (%s, %s)", (title, hours))

# Генерация экзаменов

for \_ in range(20):

s\_id = random.randint(1, 10)

c\_no = random.randint(1, 5)

score = round(random.uniform(50, 100), 2)

cur.execute("INSERT INTO Exams (s\_id, c\_no, score) VALUES (%s, %s, %s)", (s\_id, c\_no, score))

# Сохранение изменений и закрытие соединения

conn.commit()

cur.close()

conn.close()