Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»
(ВлГУ)**

УТВЕРЖДАЮ
Заведующий кафедрой ИСПИ
_____ И.Е. Жигалов
«20» марта 2025 г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ПРАКТИЧЕСКИМ РАБОТАМ МЕЖДИСЦИПЛИНАРНОГО КУРСА

«ТЕСТИРОВАНИЕ ИНФОРМАЦИОННЫХ РЕСУРСОВ»

В РАМКАХ ПРОФЕССИОНАЛЬНОГО МОДУЛЯ

«ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ИНФОРМАЦИОННЫХ РЕСУРСОВ»

09.02.09 Веб-разработка
Разработчик веб приложений

Владимир, 2025

Методические указания к практическим работам междисциплинарного курса «Тестирование информационных ресурсов» разработал ассистент кафедры ИСПИ Хлызова В.Г.

Методические указания к практическим работам рассмотрены и одобрены на заседании УМК специальности 09.02.09 Веб-разработка
протокол № 1 от «10» марта 2025 г.

Председатель УМК специальности _____ И.Е. Жигалов

Методические указания к практическим работам рассмотрены и одобрены на заседании кафедры ИСПИ
протокол № 7а от «12» марта 2025 г.

Рецензент от работодателя:
руководитель группы обеспечения
качества программного обеспечения
ООО «БСЦ МСК» _____ С.С. Смирнова

Содержание

Практическая работа №1. Изучение этапов тестирования. Правила работы с системами для управления проектами небольших групп. 4

Практика №2. Оформление тестовых планов на основе бизнес-требований. Чек-листы. Тест-туры 7

Практика №3. Майнд-карты. Использование современных средств для структурирования информации 12

Примеры применения майнд-мап в тестировании: 13

Практическая работа №4. Классификация видов тестирования. Оформление дефектов 17

Практическая работа №5. Выбор автоматизированного средства тестирования. 28

Практическая работа №6. Изучение и использование автоматизированного средства тестирования. 32

Планирование, Проектирование и Разработка 35

Практическая работа №7. Открытое бета-тестирование 37

Практическая работа №8. Тестирование установки и приемочное тестирование. 42

СПИСОК ЛИТЕРАТУРЫ 46

Настоящие методические указания по дисциплине "Тестирование информационных ресурсов" предназначены для студентов, работающих над курсовым проектом по реализации веб-приложения. В ходе практических занятий студенты будут изучать процесс тестирования на всех его этапах, начиная от анализа бизнес-требований и разработки тест-кейсов и заканчивая принятием решения о внедрении проекта в эксплуатацию.

Процесс тестирования будет рассмотрен как системный подход, включающий в себя несколько ключевых фаз. Сначала студенты освоят методы анализа бизнес-требований, что позволит им четко определить функциональные и нефункциональные характеристики системы. На основании этих требований будут разработаны тест-кейсы, которые помогут в дальнейшем выявить возможные дефекты и несоответствия. Наконец, студенты смогут оценить готовность проекта к внедрению, принимая обоснованные решения о его запуске в эксплуатацию.

Цель данных методических указаний — предоставить студентам необходимые знания и навыки для эффективного проведения тестирования информационных систем, что в свою очередь обеспечит высокое качество и надежность разрабатываемых ими проектов.

Практическая работа №1. Изучение этапов тестирования. Правила работы с системами для управления проектами небольших групп.

ЦЕЛЬ

Познакомиться с инструментом для управления проектами небольших групп - Yougile.

Для выполнения практических работ в качестве метода управления проектом был выбран подход Kanban. Цель данного метода заключается в получении готового приложения в заданные сроки, путем деления задачи на более мелкие подзадачи.

Подход Kanban представляет собой визуальную и гибкую методологию управления проектами, которая обеспечивает прозрачность процессов и непрерывное улучшение рабочих потоков. В контексте тестирования информационных ресурсов Kanban помогает организовать и оптимизировать процесс тестирования, повысить его эффективность и гибкость, а также улучшить взаимодействие между членами команды.

Колонки (Этапы)

Разработка начинается с создания специальной доски Kanban и определения плана работ. Колонки отображают различные стадии процесса работы, через которые проходят задачи. Они могут быть настроены в зависимости от конкретных нужд команды и проекта, но часто включают:

- Backlog (Бэклог): Список всех задач или требований, которые еще не начали выполнять.
- To Do (К выполнению): Задачи, которые планируется начать в ближайшее время.
- In Progress (В процессе): Задачи, над которыми активно работают в данный момент.
- Review (Обзор): Задачи, которые завершены и ожидают проверки или одобрения.
- Done (Завершено): Завершенные задачи, которые прошли все стадии проверки и одобрения.

Каждый шаг определяет стадию выполнения задачи. Все проектные задачи представлены на доске в виде карточек, которые перетаскиваются по мере реализации между колонками в зависимости от их статуса выполнения.

Карточки

Карточки представляют собой единицы работы и содержат информацию о конкретной задаче или элементе работы. Основное содержание карточек может включать:

- **Название задачи:** Краткое описание того, что требуется сделать.
- **Описание задачи:** Более подробная информация о задаче, включая требования и критерии выполнения.
- **Ответственное лицо:** Член команды, ответственный за выполнение задачи.
- **Сроки:** Дата начала и окончания, если применимо.
- **Приоритет:** Уровень важности задачи.
- **Теги или метки:** Дополнительная информация, такая как тип задачи или связанная функциональность.
- **Комментарии:** История изменений, заметки или дополнительная информация по задаче.

Лимиты WIP (Work In Progress)

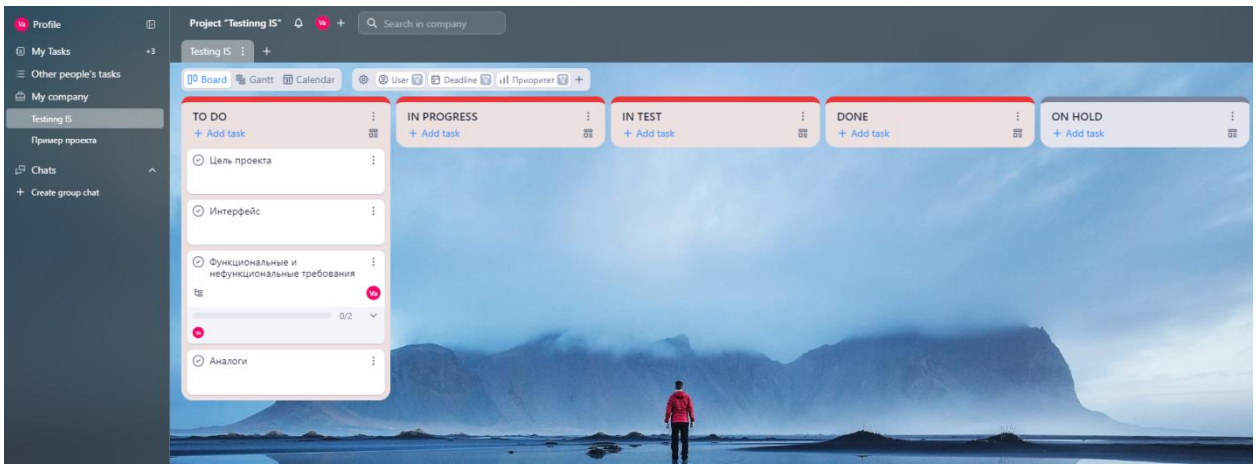
Лимиты WIP устанавливают максимальное количество задач, которые могут находиться в каждой колонке одновременно. Это помогает предотвратить перегрузку команды и обеспечить более равномерное распределение задач. Например, можно установить лимит в 3 задачи для колонки "In Progress", чтобы избежать ситуаций, когда слишком много задач одновременно находятся в процессе выполнения.

Адаптивные элементы

В зависимости от специфики команды и проекта, Kanban-доска может содержать адаптивные элементы, такие как:

- **Индикаторы приоритета:** Обозначения или цветовые метки для обозначения срочных задач.
- **Исторические записи:** Заметки о выполненных задачах и их изменениях.

Такая визуализация процесса разработки позволяет видеть картину целиком, корректировать отдельные процессы разработки, вносить в проект изменения и оценивать их влияние на весь проект. При использовании подхода Kanban разработчику важно выбрать темп работы, который будет удобен и не навредит срокам проекта. Пример Kanban доски:



Принципы Kanban:

1. Визуализация работы: Представление задач и их состояния на Kanban-доске.
2. Ограничение незавершенной работы: Установка лимитов на количество задач в каждом состоянии, чтобы предотвратить перегрузку.
3. Управление потоком: Оптимизация рабочего процесса для повышения эффективности.
4. Непрерывное улучшение: Регулярный анализ и улучшение процессов на основе данных и фидбэка.

В качестве виртуальной доски была использована система для управления небольшими проектами - Yougile (https://ru.yougile.com/). Система Yougile используется для наглядного представления рабочего процесса и представляет рабочий стол, на котором можно создать специальные колонки и карточки с задачами. В Yougile есть возможность задать срок выполнения задачи, назначить задаче приоритет, добавить чек листы и вложения – данные инструменты позволяют отслеживать и документировать все этапы разработки.

Продуктивность команды зависит от эффективных инструментов и комфортной рабочей обстановки. Интуитивно понятные функции Yougile позволяют команде быстро настроить рабочие процессы для любых задач: от совещаний и проектов до мероприятий и постановки целей. Yougile помогает командам эффективно решать рабочие задачи.

Рекомендации по применению Kanban:

1. Поддерживайте доску в актуальном состоянии: убедитесь, что информация на доске всегда актуальна и точна.
2. Обратите внимание на коммуникацию: Регулярное общение и сотрудничество между членами команды помогут эффективно использовать Kanban.

3. Будьте готовы к изменениям: Kanban предполагает гибкость и адаптацию к изменениям в проекте и требованиях.

ЗАДАНИЕ К ПРАКТИЧЕСКОЙ РАБОТЕ

1. Зарегистрироваться на Yougile, создать канбан доску для своего проекта;
2. Оформить канбан-доску под нужды своего проекта (создать столбцы для этапов разработки, описать функциональные и нефункциональные требования, прикрепить ссылку на Figma и т.д.);
3. Оформить отчет по практической работе, в отчет добавить описание проекта, ссылку на созданную канбан-доску и скриншоты из Yougile.

ВОПРОСЫ

1. Что такое Kanban-доска и какова её основная цель?
2. Какие основные компоненты включает стандартная Kanban-доска?
3. Каковы основные принципы Kanban?
4. Как определить стадии рабочего процесса для Kanban-доски?
5. Как правильно использовать карточки на Kanban-доске?
6. Как установить лимиты WIP (Work In Progress) и зачем это нужно?
7. Какие преимущества и недостатки использования Kanban-доски для управления проектами?
8. Как Kanban-доска помогает в выявлении и устранении узких мест в процессе?
9. Как часто нужно обновлять Kanban-доску и кто должен быть ответственным за это?
10. Как Kanban-доска способствует улучшению командной работы и коммуникации?

Практика №2. Оформление тестовых планов на основе бизнес-требований. Чек-листы. Тест-туры

ЦЕЛЬ РАБОТЫ

Научиться описывать сценарии использования, тест-кейсы, тест-туры и чек-листы.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

1. Тест-кейсы.

[IEEE 610] - Согласно стандартному глоссарию терминов, используемых в тестировании программного обеспечения ISTQB®/GTB/TAV, Версия 2014-0 (от 9 июля 2014 года).

Тестовый сценарий (test case) - набор входных значений, предусловий выполнения, ожидаемых результатов и постусловий выполнения, разработанный для определенной цели или тестового условия, таких как выполнения определенного пути программы или же для проверки соответствия определенному требованию. [IEEE 610].

Тест-кейс — это профессиональная документация тестировщика, последовательность действий, направленная на проверку какого-либо функционала, описывающая как прийти к фактическому результату.

Зачем нужны тест-кейсы?

Тест-кейсы должен помочь нам провести проверку продукта без ознакомления со всей документацией. Написанный один раз, удобный в поддержке тест-кейс сэкономит много времени и сил тестировщикам.

Атрибуты тест-кейса

Любой тест-кейс обязательно включает в себя:

Уникальный идентификатор тест-кейса — необходим для удобной организации хранения и навигации по нашим тест-наборам.

Название — основная тема, или идея тест-кейса. Кратное описание его сути.

Предусловия — описание условий, которые не имеют прямого отношения к проверяемому функционалу, но должны быть выполнены.

Например, оставить комментарий на вашем портале может только зарегистрированный пользователь. Значит для тест-кейса «Создание комментария» будет необходимо выполнение предусловия «пользователь зарегистрирован», и «пользователь авторизован»

Шаги — описание последовательности действий, которая должна привести нас к ожидаемому результату

Ожидаемый результат — результат: что мы ожидаем увидеть после выполнения шагов.

Пример тест-кейса для проверки операции умножения в калькуляторе для целых чисел.

№ 1

Название Операция «Умножение». Умножение простых чисел.

Предусловия Приложение «Калькулятор» запущено. Поле ввода пустое.

Последовательность действий 1. Ввести в поле ввода число.

2. Ввести математический оператор.

3. Ввести ещё одно число.

4. Нажать кнопку «равно».

Ожидаемый результат В окне ввода/вывода появляется число, которое является произведением нескольких введённых чисел.

Жизненный цикл тест-кейса

Создан — типичное начальное состояние практически любого артефакта. Тест-кейс автоматически переходит в это состояние после создания.

Запланирован — в этом состоянии тест-кейс находится, когда он или явно включён в план ближайшей итерации тестирования, или как минимум готов для выполнения.

Не выполнен — в некоторых системах управления тест-кейсами это состояние заменяет собой предыдущее («запланирован»). Нахождение тест-кейса в данном состоянии означает, что он готов к выполнению, но ещё не был выполнен.

Выполняется — если тест-кейс требует длительного времени на выполнение, он может быть переведён в это состояние для подчёркивания того факта, что работа идёт, и скоро можно ожидать её результатов. Если выполнение тест-кейса занимает мало времени, это состояние, как правило, пропускается, а тест-кейс сразу переводится в одно из трёх следующих состояний — «провален», «пройден успешно» или «заблокирован».

Пропущен — бывают ситуации, когда выполнение тест-кейса отменяется по соображениям нехватки времени или изменения логики тестирования.

Провален — данное состояние означает, что в процессе выполнения тест-кейса был обнаружен дефект, заключающийся в том, что ожидаемый результат как минимум одному шагу тест-кейса не совпадает с фактическим результатом. Если в процессе выполнения тест-кейса был «случайно» обнаружен дефект, никак не связанный с шагами тест-кейса и их ожидаемыми результатами, тест-кейс считается пройденным успешно (при этом, естественно, по обнаруженному дефекту создаётся отчёт о дефекте).

Пройден успешно — данное состояние означает, что в процессе выполнения тест-кейса не было обнаружено дефектов, связанных с расхождением ожидаемых и фактических результатов его шагов.

Заблокирован — данное состояние означает, что по какой-то причине выполнение тест-кейса невозможно (как правило, такой причиной является наличие дефекта, не позволяющего реализовать некий пользовательский сценарий).

Закрыт — очень редкий случай, т.к. тест-кейс, как правило, оставляют в состояниях «провален / пройден успешно / заблокирован / пропущен». В данное состояние в некоторых системах управления тест-кейс переводят, чтобы подчеркнуть тот факт, что на данной итерации тестирования все действия с ним завершены.

Требует доработки— как видно из схемы, в это состояние (и из него) тест-кейс может быть переведён в любой момент времени, если в нём будет обнаружена ошибка,

если изменятся требования, по которым он был написан, или наступит иная ситуация, не позволяющая считать тест-кейс пригодным для выполнения и перевода в иные состояния.

Рисунок 1. Жизненный цикл тест-кейса

2. Сценарии использования

Сценарии использования, варианты использования или прецеденты (Use cases) — спецификация последовательностей действий, которые может осуществлять система, подсистема или класс, взаимодействуя с внешними действующими лицами.

Изначально варианты использования задумывались для описания поведения и возможностей программных систем, сейчас они с тем же успехом применяются для описания систем любого типа вне зависимости от того, реализована ее функциональность программно или нет.

Пример:

Название: «Покупка товара».

Предусловие: покупатель авторизован в системе и находится на главной странице.

Основной сценарий:

1. Покупатель просматривает каталог и выбирает товары для покупки.
2. Покупатель оценивает стоимость всех товаров.
3. Покупатель вводит информацию, необходимую для доставки товара.
4. Система предоставляет полную информацию о цене товара и его доставке.
5. Покупатель вводит информацию о кредитной карточке.
6. Система осуществляет авторизацию счета покупателя.
7. Система подтверждает оплату товаров немедленно.
8. Система посылает подтверждение оплаты товаров по адресу электронной почты покупателя.

Альтернативный сценарий

1. Покупатель просматривает каталог и выбирает товары для покупки.
2. Покупатель оценивает стоимость всех товаров.
3. Покупатель вводит информацию, необходимую для доставки товара.
4. Система предоставляет полную информацию о цене товара и его доставке.
5. Покупатель вводит информацию о кредитной карточке.
6. Система осуществляет авторизацию счета покупателя.
7. Система посылает ответ, что средств на счёте недостаточно.
8. Система перенаправляет покупателя на страницу просмотра каталога.

3. Чеклисты

Чек—лист – список, содержащий ряд необходимых проверок для какой-либо работы.

Чек – лист в тестировании – список проверок для тестирования продукта. Чек-листы устроены предельно просто. Любой из них содержит перечень блоков, секций, страниц, других элементов, которые следует протестировать.

Выполненные пункты отмечаются статусами, например: “Passed”, “Failed”, “Blocked”, “Skipped”, “Not run”.

Чек-лист по юзабилити.

4. Тест-туры

Под тестированием по методике туров понимается, что ваше приложение – это незнакомый город, а тестировщик – это турист.

Для того, чтобы пользоваться этой методикой, надо изучить цели тура. Время для исследования по туру выбирает сам тестировщик, например, час. Далее уже проводим само исследование системы строго по целям тура ни на что не отвлекаясь, только «миссия» тура.

Например, цель денежного тура: найти функции, которые заставляют людей тратить деньги на ваше ПО и проверить их работу.

ЗАДАНИЯ ДЛЯ ВЫПОЛНЕНИЯ

1. Изучить представленный материал.

2. Напишите основной и альтернативный сценарий поведения вашей системы (опциально, легче будет понять основной сценария поведения вашего веб-приложения и написать тест-кейсы).

3. Опишите 5 тест-кейсов, 2 из которых на основе ранее разработанных основного и альтернативного сценария.

4. Составить чеклист для тестирования функционала вашего приложения, выберите из предложенного или можете придумать сами:

- Чеклист для тестирования безопасности (нацелена на поиск недостатков и пробелов с точки зрения безопасности вашего приложения).
- Чеклист для тестирования форм (регистрация, авторизация, поля)
- Чеклист для тестирования удобства использования.

5. Тест-туры, выбрать два любых понравившихся тест-тура (http://testbase.ru/bugs)

ВОПРОСЫ

1. Что такое тест-кейс и какие основные компоненты он включает?
2. Каковы цели и преимущества использования тест-кейсов?
3. Какие атрибуты тест-кейса являются обязательными, и как они помогают в организации тестирования?

4. Каковы возможные состояния тест-кейса в его жизненном цикле?
5. В каких случаях тест-кейс может быть переведен в состояние "требует доработки"?
6. Какое значение имеет "ожидаемый результат" в тест-кейсе и как его правильно определить?
7. Что такое предусловия в тест-кейсе и как они влияют на выполнение теста?
8. Как можно улучшить тест-кейсы для повышения их эффективности и удобства использования?
9. Что такое сценарий использования и как он связан с тест-кейсами?
10. Как правильно оформить сценарий использования и какие ключевые элементы в него включаются?
11. В чем разница между основным и альтернативным сценариями использования?
12. Как сценарии использования могут помочь в создании более эффективных тест-кейсов?
13. Что такое чек-лист и как он используется в тестировании?
14. Каковы преимущества и недостатки использования чек-листов по сравнению с тест-кейсами?
15. Как правильно создавать и использовать чек-листы в процессе тестирования?
16. Что такое тест-тур и как он отличается от других методов тестирования?
17. Каковы основные цели и принципы тестирования по методике туров?
18. Как выбрать цели для тест-тура и как это влияет на результаты тестирования?
19. Какие рекомендации можно дать для эффективного проведения тест-туров?
20. Как интегрировать методику тест-туров с другими методами тестирования?

Практика №3. Майнд-карты. Использование современных средств для структурирования информации

ЦЕЛЬ РАБОТЫ

Познакомиться со способом изучения продукта и проектированием с помощью MindMap. Рассмотреть майнд-карты в тестировании.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Исследование продукта

Перед началом проектирования необходимо исследовать предметную область, определить цели и задачи ИС. Исследование продукта помогает в изучении предметной области.

Майнд-карты

Майнд-карты (так же их называют интеллект-карты) – это способ упорядочивания информации в виде ассоциативных или логических схем. Они помогают при работе с анализом и запоминанием информации, планированием и принятием решений.

Майнд-мап (или интеллект-карта) в тестировании — это визуальное представление идей, процессов и информации, которое помогает организовать и систематизировать данные о тестировании. В тестировании майнд-мапы могут быть использованы для различных целей, таких как планирование тестирования, разработка тест-кейсов, управление дефектами и другие аспекты процесса тестирования.

Преимущества использования майнд-мап в тестировании:

1. Визуализация информации: Обеспечивает наглядное представление всех аспектов тестирования, упрощая понимание и коммуникацию.

2. Организация идей: Помогает структурировать сложные идеи и процессы, делая их более управляемыми.
3. Идентификация пропусков: Позволяет увидеть, какие части тестирования могут быть упущены или недостаточно проработаны.
4. Улучшение креативности: Способствует генерации новых идей и подходов к тестированию.

Примеры применения майнд-мап в тестировании:

1. Планирование тестирования:
   - Цели тестирования: Определите цели и задачи тестирования.
   - Области тестирования: Разделите систему на компоненты или функции для тестирования.
   - Методы и подходы: Определите используемые методы (например, функциональное тестирование, тестирование производительности и т.д.).
   - Ресурсы: Укажите необходимые ресурсы и инструменты для тестирования.
2. Разработка тест-кейсов:
   - Тестовые сценарии: Разработайте сценарии для различных функций и сценариев использования.
   - Шаги тестирования: Определите последовательность шагов и ожидаемые результаты.
   - Предусловия: Установите условия, которые должны быть выполнены до начала тестирования.
3. Управление дефектами:
   - Классификация дефектов: Разделите дефекты по типам (например, критические, серьезные, незначительные).
   - Приоритеты: Установите приоритеты для исправления дефектов.
   - Ответственные лица: Назначьте ответственных за исправление дефектов.
4. Анализ рисков:
   - Типы рисков: Определите потенциальные риски, связанные с тестированием.
   - Вероятность и влияние: Оцените вероятность возникновения рисков и их влияние на проект.
   - Стратегии управления: Разработайте стратегии для минимизации рисков.
5. Отчеты и результаты:
   - Статистика тестирования: Визуализируйте результаты тестирования, такие как процент выполненных тестов и обнаруженные дефекты.
   - Анализ производительности: Представьте информацию о производительности системы и тестов.
   - Рекомендации: Подготовьте рекомендации по улучшению процесса тестирования или системы.

Майнд-карты являются инструментом для визуального отображения информации, который помогает эффективно её структурировать.

Отметим основные преимущества таких карт:

1. Наглядность и визуализация – главное преимущество в таких картах для тестировщика, это наглядное видение тестируемого продукта, его функций и зависимостей между собой.
2. Отличная альтернатива документации – хороший вариант для демонстрации карты новым сотрудникам в качестве альтернативы или дополнения к документации.
3. Легко поддерживать – с выходом новых функций карту будет несложно дополнить и вновь отследить новые части приложения, возможно где-то сделать продукт проще и понятнее пользователю.

Составление майнд карты

Среда выполнения (можете использовать какой-нибудь другой инструмент).

При начале работы вы сможете просмотреть инструкцию по созданию майнд-карты.

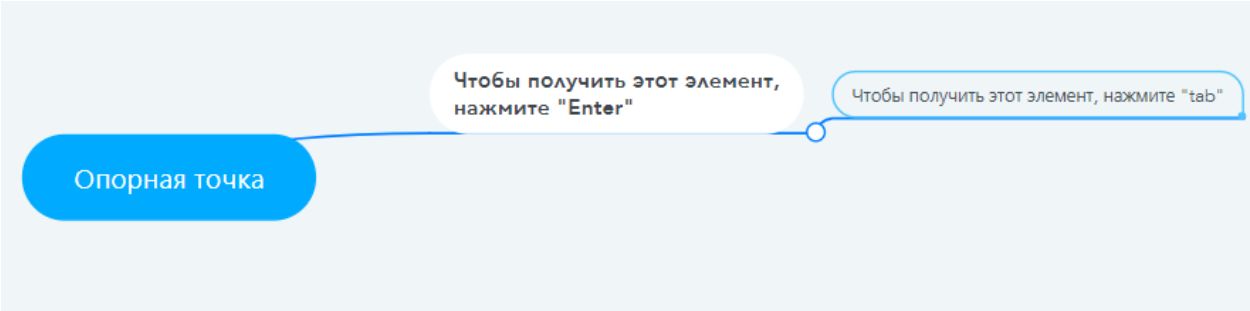


Рисунок 1. Добавление элементов

Фрагмент майнд-карты для сайта [«Читай-город»](#).

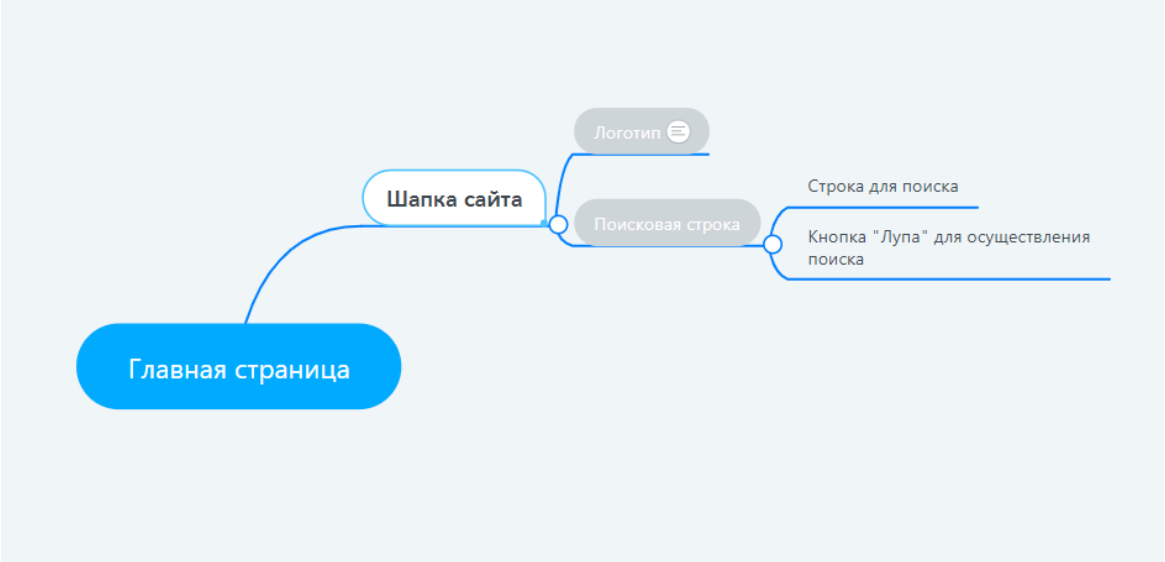


Рисунок 2. Фрагмент карты

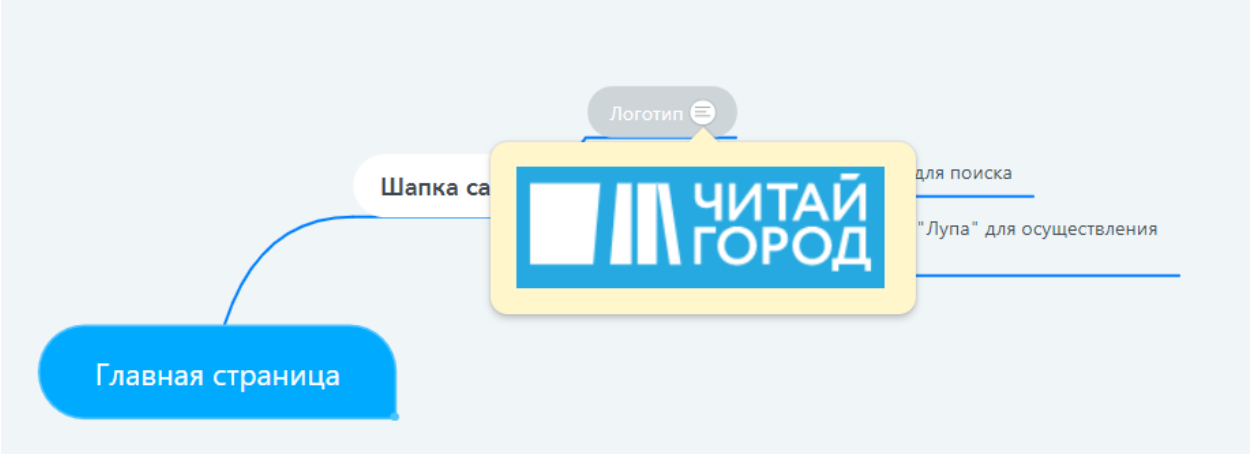


Рисунок 3. Пример добавления описания для элемента

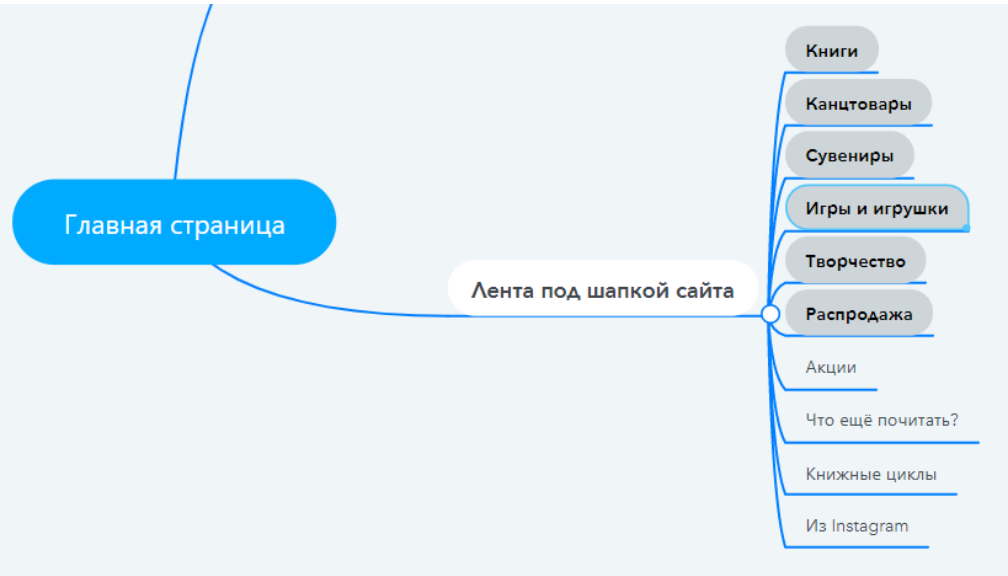


Рисунок 4. Фрагмент 2

Рисунок 5. Фрагмент 3

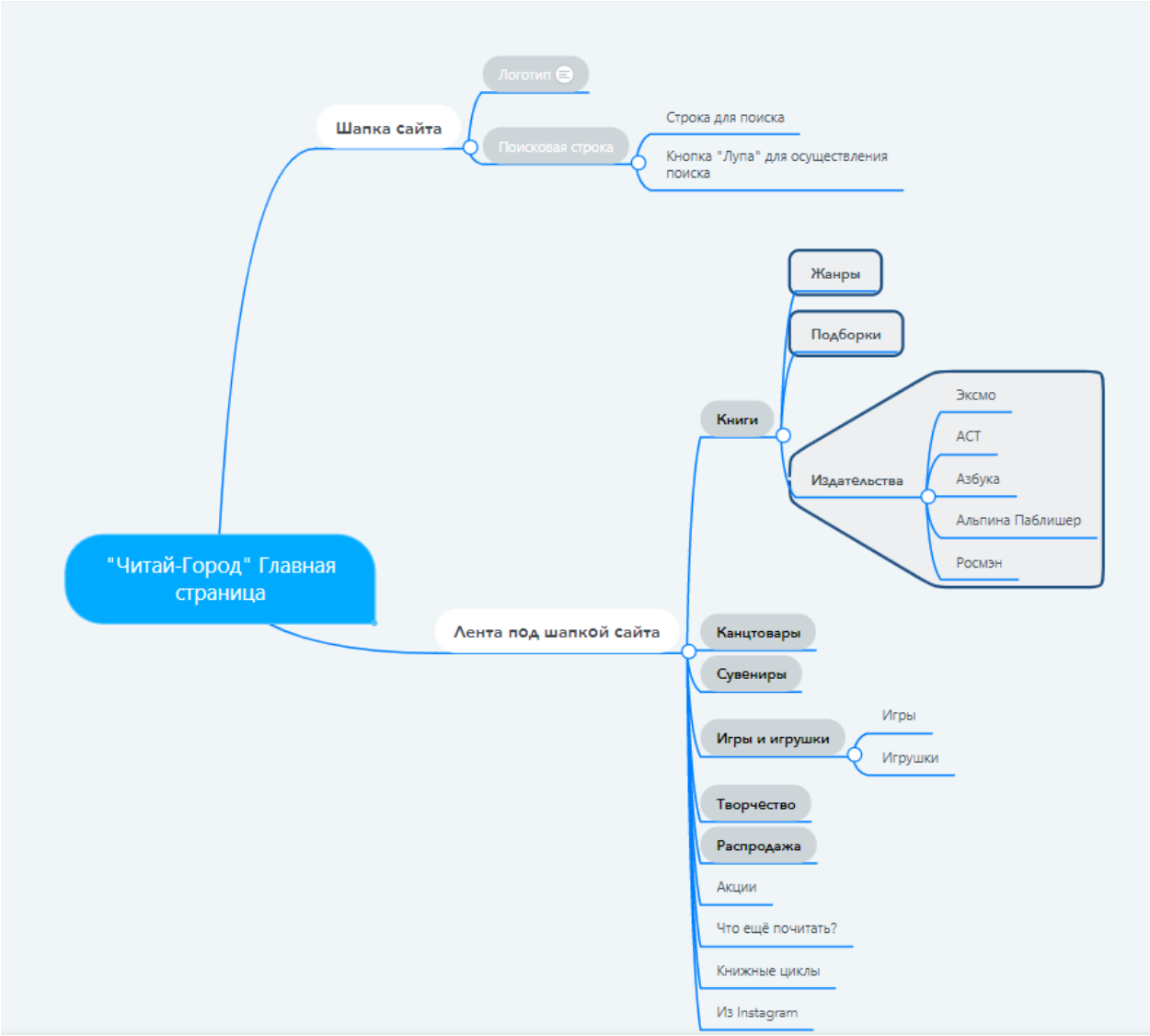


Рисунок 6. Пример результата

Пример карты: https://mm.tt/app/map/2989966553?t=KgbztDxMa3.

Рекомендации по созданию майнд-карт:

1. Используйте простые и понятные ключевые слова.
2. Организуйте информацию логично, начиная с центральной идеи и распространяясь на более детализированные аспекты.
3. Используйте цвета и изображения для улучшения визуализации и понимания.
4. Обновляйте майнд-мап по мере изменения информации или требований.

ЗАДАНИЯ ДЛЯ ВЫПОЛНЕНИЯ

1. Изучить представленный материал.

2. Спроектировать манд-карту вашего разрабатываемого приложения, как вы видите своё приложение.

3. Напишите своими словами, как вы понимаете понятие «Майнд-карты».

ВОПРОСЫ

1. Что такое майнд-карта и каковы её основные элементы?
2. Каковы основные преимущества использования майнд-карт в управлении проектами и тестировании?
3. Как майнд-карты могут помочь в планировании тестирования?
4. Каким образом можно использовать майнд-карты для разработки тест-кейсов?
5. Как майнд-карты могут помочь в управлении дефектами и отслеживании их статусов?
6. Какие способы могут быть использованы для интеграции майнд-карт с другими инструментами тестирования?
7. Какие рекомендации следует учитывать при создании майнд-карт для тестирования?
8. Как можно использовать цветовые схемы и изображения в майнд-картах для улучшения восприятия информации?
9. Какие цифровые инструменты и программы можно использовать для создания майнд-карт, и как выбрать наиболее подходящий?
10. Как обновлять и поддерживать актуальность майнд-карт в условиях изменения требований или процесса тестирования?
13. Можете ли вы привести примеры успешного применения майнд-карт в тестировании?
14. Какие типичные ошибки следует избегать при создании майнд-карт для тестирования?

Практическая работа №4. Классификация видов тестирования. Оформление дефектов

ЦЕЛЬ

Обеспечить студентов глубоким пониманием различных видов тестирования программного обеспечения, их целей и методов, а также научить их правильно оформлять дефекты, чтобы обеспечить эффективное взаимодействие между тестировщиками и разработчиками.

Уровни тестирования

Уровни тестирования – уровни, которые определяют то, над чем производятся тесты: над отдельным модулем, группой модулей или системой в целом. Тестирование на разных уровнях производится на протяжении всего жизненного цикла разработки и

сопровождения ПО. Проведения тестирования на всех уровнях системы – залог успешной реализации.

В тестировании выделяют четыре уровня тестирования:

1. Компонентное тестирование (модульное) – проверяет функциональность и ищет дефекты в частях приложения, которые доступны и могут быть протестированы по отдельности (модули программ, объекты, классы, функции).
2. Интеграционное тестирование – предназначено для проверки связи между компонентами, а также взаимодействия с различными частями системы (ОС, оборудованием, связи между различными системами).
    - Компонентный интеграционный уровень - проверяется взаимодействие между компонентами системы после проведения компонентного тестирования.
    - Системный интеграционный уровень - проверяется взаимодействие между разными системами после проведения системного тестирования.
3. Системное тестирование – проверка функциональных требований, дефекты системы в целом. При этом выявляется неверное использование ресурсов системы, непредусмотренные комбинации данных пользовательского уровня, несовместимость с окружением, непредусмотренные сценарии использования, отсутствующая или неверная функциональность, неудобство использования и т.д. Для минимизации рисков, связанных с особенностями поведения в системы в той или иной среде, во время тестирования рекомендуется использовать окружение, максимально приближенное к тому, на которое будет установлен продукт после выдачи.
4. Приёмочное тестирование – формальный процесс тестирования, который проверяет соответствие системы требованиям и проводится с целью:
    - определения удовлетворения системой приемочным критериям;
    - вынесения решения заказчиком или другим уполномоченным лицом принятия приложения.
    - Продукт достиг необходимого уровня качества.
    - Заказчик ознакомлен с Планом Приемочных Работ (Product Acceptance Plan) или иным документом, где описан набор действий, связанных с проведением приемочного тестирования, дата проведения, ответственные.

Классификация видов тестирования:

1. По формальности
    a. По тестам

b. Исследовательское
c. Интуитивное - необходимость такого тестирования возникает, когда у тестировщиков мало времени и они не могут выполнить тщательное тестирование с использованием тест-кейсов и тест-планов. Интуитивное тестирование выполняется после проведения формального тестирования приложения.

2. По позитивности сценария
    a. Позитивное - это тестирование с применением сценариев, которые соответствуют нормальному (штатному, ожидаемому) поведению системы. С его помощью мы можем определить, что система делает то, для чего и была создана.
    b. Негативное - это тестирование на данных или сценариях, которые соответствуют нештатному поведению. Основной целью “негативного” тестирования является проверка устойчивости системы к воздействиям различного рода, валидация неверного набора данных.
3. По исполнению кода
    a. Статическое – тестирование без запуска кода на исполнение.
    b. Динамическое - включает в себя тестирование ПО в режиме реального времени путем предоставления входных данных и изучения результата поведения программы. Проверка осуществляется с помощью ручного или автоматического выполнения заранее подготовленного набора тестов. Оно является частью процесса валидации программного обеспечения. То есть любое тестирование, в котором мы начинаем взаимодействовать с приложением, является динамическим.
4. По знанию системы
    a. Белый ящик – тестирование с доступом к внутренней структуре и коду приложения.
    b. Черный ящик - тестирование, основанное на анализе функциональной или нефункциональной спецификации системы, при котором программа рассматривается как объект, внутренняя структура которого неизвестна.
    c. Серый ящик – тестировщик имеет доступ к части кода и архитектуры.
5. По уровню детализации приложения
    a. Модульное
    b. Интеграционное
    c. Системное

d. Приёмочное

6. По разработке тестовых сценариев
    a. На основе требований
    b. По вариантам использования
    c. На основе моделей
7. По привлечению конечных пользователей
    a. Альфа-тестирование - имитация реальной работы с системой штатными разработчиками, либо реальная работа с системой потенциальными пользователями/заказчиком.
    b. Бета-тестирование - интенсивное использование почти готовой версии продукта (как правило, программного или аппаратного .обеспечения) с целью выявления максимального числа ошибок в его работе для их последующего устранения перед окончательным выходом продукта на рынок, к массовому потребителю.
    c. Гамма-тестирования - финальная стадия тестирования перед выпуском продукта, направленная на исправление незначительных дефектов, обнаруженных в бета-тестировании.
8. По целям
    a. Функциональное - это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает.
        i. Смоук тестирование - в тестировании программного обеспечения означает минимальный набор тестов на явные ошибки.
        ii. Критического пути - это тестирование функциональности, используемой пользователями в типичной повседневной деятельности, наиболее часто используемые функции ПО.
        iii. Расширенное - вид углубленного тестирования, при котором проверяется нестандартное использование программного продукта, границы переполнения массивов данных, ввод специальных символов.
    b. Нефункциональное - это тестирование программного приложения или системы на предмет его нефункциональных требований: того, как система работает, а не конкретного поведения этой системы.
        i. Пользовательского интерфейса

ii. Удобства использования
iii. Защищённости
iv. Инсталляционное - проверка возможности развертывания и конфигурирования системы из дистрибутива, предоставленного разработчиком.
v. Конфигурационное
vi. Совместимости
vii. Локализации
viii. Надёжности и восстановления после сбоев
ix. Производительности
1. Нагрузочное тестирование- тестирование производительности, сбор показателей и определение производительности и времени отклика программно-технической системы или устройства в ответ на внешний запрос с целью установления соответствия требованиям, предъявляемым к данной системе (устройству)
2. Стабильности
3. Стрессовое - оценивает надёжность и устойчивость системы в условиях превышения пределов нормального функционирования.
4. Объёмное
5. Масштабируемости - это метод нефункционального тестирования, который измеряет производительность системы или сети при увеличении или уменьшении количества пользовательских запросов.

c. Тестирование изменений
i. Повторное
ii. Регрессионное

9. По степени автоматизации
a. Ручное
b. Автоматизированное
c. Автоматическое.

Примеры на основе сайта «Читай-город»:

Тесты	Примеры тестов

Различные виды тестирования	
Функциональные тесты	1. Заходим на сайт «Читай-город» пробуем найти книгу «Жало белого города» с помощью поисковой строки. 2. Пробуем добавить интересующую книгу в корзину с помощью кнопки «Купить».
Тесты производительности	1. Оценка времени отклика страницы «Канцтовары» на сайте «Читай-город» для 100 пользователей. 2. Проверка как быстро отображается форма оформленного заказа после оплаты товара на сайте. 3. Проверка как быстро отображается форма оформленного заказа после подтверждения заказа (указание оплаты товара при получении).
Нагрузочные тесты	1. Определение количества пользователей, которые могут работать с приложением. 2. Проверка наличия ресурсозатратных страниц на сайте. 3. Оценка времени загрузки сайта.
Тестирование совместимости	1. Проверка на кроссбраузерность, как работает сайт в разных браузерах и в разных их версиях. 2. Проверка работы сайта при различном расширении экрана.
Различные типы тестов	
Позитивные тесты	1. Поиск книги по названию (существующей книги «Жало белого города»). 2. Поиск книги по ISBN (в правильном формате) (978-5-04-113698-7). 3. Поиск книги по автору (чуществующему автору Гарсиа Саэнс).
Негативные тесты	1. Поиск несуществующей книги («прикерн»). 2. Поиск книги по артикулу не соответсвующего формату (97-85-04113-6-98-712).
Тестирование	1. Оценка лишних действий, которые приходится выполнять

интерфейса пользователя	пользователю (например, каждый раз указывать данные о заказчике. Или второй пример, на сайте читай-города в общем списке книг по жанру при переходе на следующую страницу, сама страница остаётся внизу и необходимо самой листать вверх, чтобы смотреть книги на второй странице по порядку). 2. Проверка чек-боксов (установка чек-бокса, его расположение. На сайте читай-города при отметке чек-бокса просмотр книг в наличии, в каком-то определённом жанре, всё нормально до тех пор, пока не нужно переходить на детальный просмотр книги, после возврата обратно в список книг, чек=false и появляются книги, которых нет в наличии).
Модульное тестирование	1. Проверка, действительно ли выведется книга «Жало белого города», если в поисковую строку ввести ISBN 978-5-04-113698-7. (то есть проверка метода поиска книги по ISBN)
Интеграционное тестирование	1. Проверка интерфейсной связи модуля входа в систему и модулем корзины пользователя (пользователь зашёл в свой аккаунт и в корзине находятся именно его книги).
Системное тестирование	1. Оценка простоты использования сайта с точки зрения пользователя. 2. Проверка работы приложения «Читай-город» на телефоне с точки зрения пользователя.

Понятия дефекта

Дефект - изъян в компоненте или системе, который может привести компонент или систему к невозможности выполнить требуемую функцию, например, неверный оператор или определение данных. Дефект, обнаруженный во время выполнения, может привести к отказам компонента или системы.

Жизненный цикл дефекта

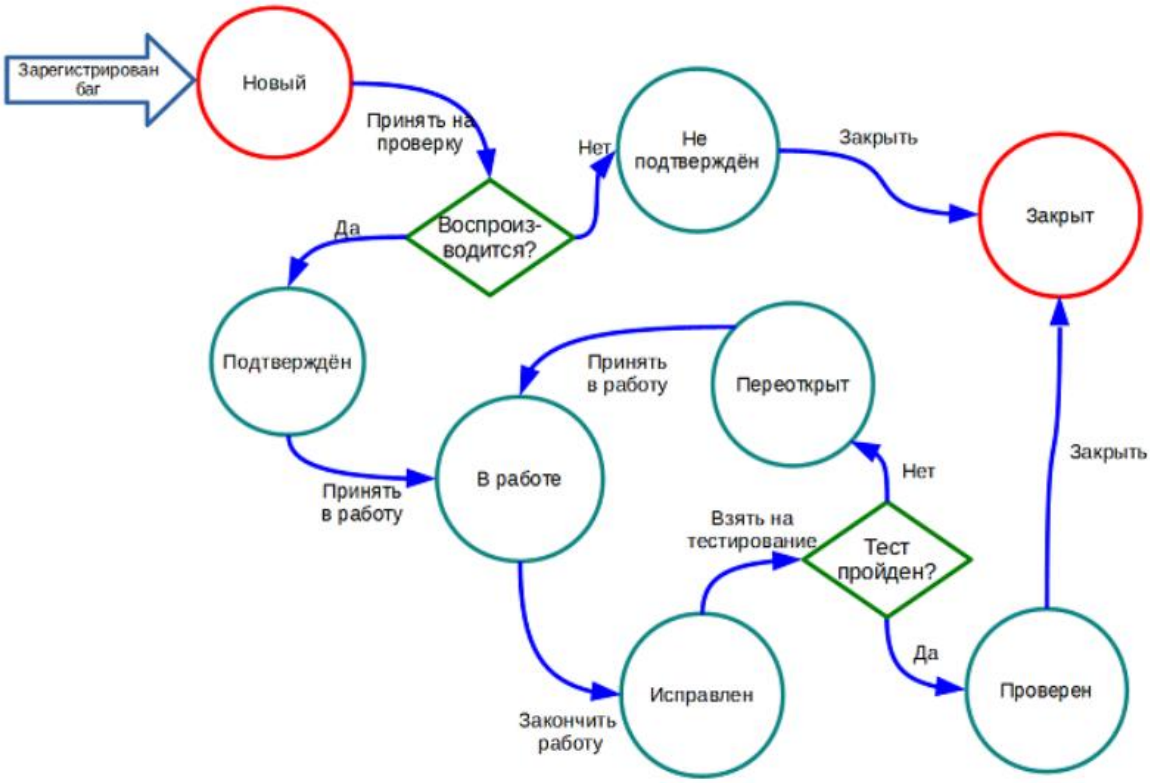


Рисунок 1. Жизненный цикл дефекта

Обнаружен (submitted) — начальное состояние отчёта (иногда называется «Новый» (new)), в котором он находится сразу после создания. Некоторые средства также позволяют сначала создавать черновик (draft) и лишь потом публиковать отчёт.

Назначен (assigned) — в это состояние отчёт переходит с момента, когда кто-то из проектной команды назначается ответственным за исправление дефекта. Назначение ответственного может производится решением лидера команды разработки, коллегиально, по добровольному принципу, иным принятым в команде способом или выполняется автоматически на основе определённых правил.

Исправлен (fixed) — в это состояние отчёт переводит ответственный за исправление дефекта член команды после выполнения соответствующих действий по исправлению.

Проверен (verified) — в это состояние отчёт переводит тестировщик, удостоверившись, что дефект на самом деле был устранён. Как правило, такую проверку выполняет тестировщик, изначально написавший отчёт о дефекте.

Закрыт (closed) — состояние отчёта, означающее, что по данному дефекту не планируется никаких дальнейших действий. Здесь есть некоторые расхождения в жизненном цикле, принятом в разных инструментальных средствах управления отчётами о дефектах.

Открыт заново (reopened) — в это состояние (как правило, из состояния «Исправлен») отчёт переводит тестировщик, удостоверившись, что дефект по-прежнему воспроизводится на билде, в котором он уже должен быть исправлен.

Рекомендован к отклонению (to be declined) — в это состояние отчёт о дефекте может быть переведён из множества других состояний, чтобы вынести на рассмотрение вопрос об отклонении отчёта по той или иной причине. Если рекомендация является обоснованной, то отчёт переводится в состояние «Отклонён».

Отклонён (declined) — в это состояние отчёт переводится в случаях, подробно описанных в пункте «Закрыт»: если средство управления отчётами о дефектах предполагает использование этого состояния вместо состояния «Закрыт» для тех или иных резолюций по отчёту.

Отложен (deferred) — в это состояние отчёт переводится в случае, если исправление дефекта в ближайшее время является нерациональным или не представляется возможным, однако есть основания полагать, что в обозримом будущем ситуация исправится (выйдет новая версия библиотеки, вернётся из отпуска специалист по необходимой технологии, изменятся требования заказчика и т.д.).

Атрибуты для оформления дефектов

На каждом проекте атрибуты дефекта могут отличаться. Основные атрибуты при оформлении дефекта:

1. Идентификатор – для каждого нового дефекта должен быть присвоен уникальный идентификатор.

2. Проект – название проекта, в котором обнаружен дефект.

3. Тема – краткое описание проблемы.

4. Описание – основной блок для описания проблемы, который включает в себя более подробное описание проблемы, указанной в теме, последовательность действий для обнаружения дефекта. Фактический и ожидаемый результат.

5. Приоритет – приоритет да т понять, как быстро необходимо исправить дефект.

6. Вложения – скриншоты, видео, которые помогут разработчику понять, что необходимо исправить.

- Immediately – для дефектов, наличие которых делает невозможным дальнейшую работу над проектом.
- High – для дефектов, дефектам, которые нужно исправить в самое ближайшее время.
- Normal – для дефектов, которые следует исправлять в порядке общей очереди.

- Low – для дефектов, которыми стоит заниматься в последнюю очередь.

Пример оформления дефектов

Идентификатор	1
Проект	Интернет-банк
Тема	Сообщение об ошибке при переходе в раздел «Услуги».
Описание	Описание: При переходе в раздел «Услуги» в шапке сайта, на странице появляется сообщение «Страница не найдена». Последовательность действий: 1. Авторизоваться. 2. Нажать на «Услуги» в шапке сайта. Фактический результат: На странице появилось сообщение об ошибке «Страница не найдена». Ожидаемый результат: Отобразилась страница с услугами.
Приоритет	High
Вложения	

ЗАДАНИЯ ДЛЯ ВЫПОЛНЕНИЯ

1. Заполните таблицу, количество примеров будет указано в скобках:

Тесты	Примеры тестов
Позитивные тесты (3)	

Негативные тесты (3)	
Тестирование белого ящика (1)	
Тестирование совместимости (2)	
Функциональное тестирование (3)	
Тестирование производительности (3)	
Тестирование интерфейса пользователя (3)	
Модульное тестирование (2)	
Интеграционное тестирование (1)	
Системное тестирование (2)	

2. Протестируйте уже имеющийся функционал по написанным во второй практической работе тест-кейсам, чеклисту (При работе с чеклистами использовать Yougile) и тест-турам, оформите найденные дефекты в Yougile (не менее 3). В случае, если дефекты не найдены, воспроизвести ситуацию для возникновения дефекта.

ВОПРОСЫ

1. В чем заключается разница между компонентным и системным интеграционным тестированием?
2. Каковы основные цели интеграционного тестирования?
3. Какие проблемы могут возникнуть при интеграции различных компонентов системы?

4. Какие аспекты системы проверяются в процессе системного тестирования?
5. Почему важно использовать окружение, максимально приближенное к рабочему?
6. Какие типичные дефекты можно обнаружить на уровне системного тестирования?
7. Какие критерии используются для проведения приёмочного тестирования?
8. Каковы основные шаги и процессы при проведении приёмочного тестирования?
9. Как влияет приёмочное тестирование на принятие решения о выпуске продукта?
10. Каковы цели позитивного тестирования и в каких случаях оно наиболее полезно?
11. Почему негативное тестирование важно для обеспечения устойчивости системы?
12. В чем заключаются основные отличия между тестированием белым, черным и серым ящиком?
13. Как доступ к внутренней структуре системы влияет на выбор метода тестирования?
14. В чем заключаются преимущества и недостатки ручного тестирования по сравнению с автоматизированным?
15. Как автоматизация тестирования влияет на процесс разработки и тестирования?
16. Какие стадии жизненного цикла дефекта существуют и что происходит на каждой стадии?
17. Каковы причины для перевода дефекта в состояние "отложен" или "рекомендован к отклонению"?
18. Какие атрибуты должны быть включены в отчет о дефекте для его эффективного разрешения?
19. Как правильно указать приоритет дефекта и как это влияет на процесс исправления?
20. Какой формат и содержание отчета о дефекте наиболее эффективны для быстрого исправления проблемы?
21. Какие детали в описании дефекта важны для разработчика, чтобы понять и устранить проблему?

Практическая работа №5. Выбор автоматизированного средства тестирования.

ЦЕЛЬ РАБОТЫ

Научиться собирать и анализировать информацию из различных источников. Выбрать средство автоматизированного тестирования для конкретной системы.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Автоматизированное тестирование – это метод тестирования программного обеспечения, который выполняется с использованием специальных программных средств, которые, в свою очередь необходимы для выполнения набора тестовых примеров.

Цель автоматизации – уменьшить количество тестовых примеров, которые нужно запускать вручную, а не полностью исключить ручное тестирование.

Автоматизация это хороший способ повысить эффективность, а также увеличить охват и скорость тестирования программного обеспечения, когда вам нужно повторять одни и те же тестовые сценарии. Преимущества автоматизированного тестирования:

- Ручное тестирование всех рабочих процессов, полей и негативных сценариев требует больше времени и денег (при определенных условиях);
- Сложно тестировать многоязычные сайты вручную;
- Не требует вмешательства человека. Запускаете и переходите к другим задачам;
- Увеличивает скорость выполнения тестов;
- Помогает увеличить охват тестированием;
- Ручное тестирование может наскучить, и следствиями станут потеря вовлеченности и появление ошибок.

В процессе автоматизации тестирования выполняются следующие шаги:

1. Выбор тестового инструмента
2. Определение объема автоматизации
3. Планирование, дизайн и разработка
4. Выполнение теста
5. Техническое обслуживание

Автоматизированное тестирование стало важным инструментом в тестировании программного обеспечения по многим причинам. Вот основные из них:

1. Повышение эффективности

Скорость выполнения: Автоматизированные тесты выполняются значительно быстрее, чем ручные. Это особенно важно при регрессионном тестировании, когда необходимо повторно проверять большое количество функций после внесения изменений.

Повторяемость: Автоматизированные тесты можно запускать многократно без необходимости повторного ручного выполнения тех же действий, что уменьшает вероятность человеческой ошибки и обеспечивает стабильность результатов.

2. Увеличение покрытия тестов

Масштабируемость: Автоматизация позволяет легко запускать большое количество тестов на разных конфигурациях и устройствах, что помогает выявить проблемы, которые могут возникнуть в различных средах.

Сложные сценарии: Сложные и многократные сценарии, которые трудоемки для ручного тестирования, могут быть автоматически проверены, что расширяет область тестирования.

3. Раннее выявление дефектов

Регулярные тесты: Автоматизированные тесты могут быть интегрированы в процесс непрерывной интеграции (CI/CD), что позволяет выявлять дефекты на ранних стадиях разработки. Это помогает быстрее находить и исправлять проблемы до того, как они достигнут продакшн-среды.

4. Снижение затрат

Экономия времени: хотя начальные затраты на настройку автоматизированных тестов могут быть высокими, в долгосрочной перспективе они сокращают время, необходимое для выполнения тестов, что снижает общие затраты на тестирование.

Снижение зависимости от тестировщиков: Автоматизированные тесты могут выполняться без участия тестировщиков, освобождая их для выполнения более сложных задач, таких как исследовательское тестирование.

5. Улучшение точности и надежности

Избежание человеческой ошибки: Автоматизация снижает количество ошибок, связанных с человеческим фактором, таких как пропуск шагов или неправильное выполнение тестов.

Однородность: Тесты выполняются точно так же каждый раз, что позволяет получить стабильные и воспроизводимые результаты.

6. Поддержка сложных сценариев

Большие объемы данных: Автоматизированные тесты могут работать с большими объемами данных, что делает их идеальными для проверки систем, обрабатывающих большие наборы данных.

Нагрузочные тесты: для тестирования производительности и нагрузки автоматизация позволяет моделировать большое количество пользователей или запросов одновременно, что сложно сделать вручную.

7. Упрощение процесса тестирования

Модульное тестирование: Автоматизация облегчает выполнение модульных тестов, что помогает в выявлении ошибок на ранних этапах разработки.

Непрерывное тестирование: Интеграция с CI/CD позволяет автоматически запускать тесты при каждом изменении кода, что способствует более быстрому выявлению и исправлению дефектов.

8. Сохранение тестовых сценариев

Документация: Автоматизированные тесты служат документацией, показывая, как система должна работать и как она тестировалась. Это упрощает понимание работы системы для новых участников проекта.

Первый шаг автоматизации «Выбор тестового инструмента», считается самым значимым. Если неправильно выбрать инструмент, то весь процесс тестирования завершится неудачей. Выбор средства тестирования во многом зависит от технологий и архитектуры, на которой построено тестируемое приложение.

ЗАДАНИЯ ДЛЯ ВЫПОЛНЕНИЯ

1. Кратко опишите технологии и архитектуру системы, для которой нужно выбрать инструмент тестирования. Если это клиент-серверное приложение, то укажите какую часть нужно протестировать (клиентскую или серверную);
2. На основании 1 пункта определите наиболее значимые критерии выбора инструмента тестирования (можно подобрать несколько дополнительные критерии, не связанных с 1 пунктом). Подобрать минимум 10 критериев. Обосновать выбор каждого критерия, например: «Так как тестирование выполняется не в коммерческих целях, а исключительно в учебных, инструмент тестирования должен быть бесплатным или иметь бесплатную пробную версию». Примеры критериев:
    a) Инструмент подходит для тестирования веб-приложений;
    b) Инструмент позволяет провести нагрузочное тестирование;
    c) Цена;
    d) Простота обучения;
    e) Наличие хорошей документации;
    f) Поддержка языков/платформ/браузеров;
    g) И т.д.
3. Используя любые источники найти минимум 5 инструментов автоматизированного тестирования, подходящих для вашей системы. Привести краткое описать найденных инструментов. Например: «Postman – инструмент для тестирования API, позволяет протестировать различные виды запросов (регистрацию, добавление и удаление данных и т.д.)».
4. Подобрать критерии оценивания для сравнения выбранных систем. Критерии оценивания могут быть любыми, как количественными (2 – Полностью удовлетворяет критерию; 1 – Частично удовлетворяет, требует существенной доработки; 0 – Не удовлетворяет поставленному критерию.), так и качественными

(+ или -, подходит или не подходит, удовлетворяет критерию или не удовлетворяет).

5. Составить таблицу сравнения систем автоматизированного тестирования по выбранным критериям. На основе таблицы сравнения выбрать подходящий инструмент тестирования, сделать вывод.

Таблица 1 – Пример таблицы сравнения инструментов автоматизированного тестирования.

	Selenium	Postman	Apache JMeter
Инструмент подходит для тестирования веб-приложений	2	1	2
Простота обучения	2	2	0
Итог	4	3	2

ВОПРОСЫ

- Какие цели вы хотите достичь с помощью автоматизации тестирования?
- Какие типы тестов вы планируете автоматизировать?
- Какой инструмент для автоматизации тестирования вы выбрали и почему?
- Каковы критерии выбора тестов для автоматизации?
- Как вы будете поддерживать тесты в актуальном состоянии?
- Как будет организована интеграция автоматизированных тестов в процесс разработки?
- Как вы планируете управлять тестовыми данными?
- Какие метрики вы будете использовать для оценки эффективности автоматизированного тестирования?
- Как вы будете управлять результатами тестирования?
- Какие потенциальные риски и проблемы вы видите в автоматизации тестирования?
- Как вы будете обеспечивать надёжность и стабильность автоматизированных тестов?
- Как вы планируете тестировать автоматизированные тесты?

Практическая работа №6. Изучение и использование автоматизированного средства тестирования.

ЦЕЛЬ РАБОТЫ

Изучить документацию выбранного в 5 практической работе средства тестирования. Протестировать с помощью выбранного средства имеющийся функционал.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Автоматизированное тестирование – это метод тестирования программного обеспечения, который выполняется с использованием специальных программных средств, которые, в свою очередь необходимы для выполнения набора тестовых примеров.

Вот несколько ключевых причин, почему автоматизированное тестирование важно для тестировщиков:

1. Повышение эффективности

Автоматизация позволяет тестировщикам выполнять тесты значительно быстрее по сравнению с ручным тестированием. Это особенно полезно для тестирования большого количества функциональностей и сценариев, которые требуют регулярного повторного выполнения, таких как регрессионные тесты.

2. Уменьшение человеческого фактора

Ручное тестирование может быть подвержено человеческим ошибкам, таким как упущение важных шагов или неверное выполнение тестов. Автоматизированные тесты, выполненные с использованием скриптов, минимизируют вероятность таких ошибок и обеспечивают большую точность в проверке функциональности.

3. Повышение покрытия тестами

Автоматизированное тестирование позволяет охватить большее количество тестовых случаев и сценариев, что способствует более полному тестированию приложения. Это особенно важно для сложных систем, где необходимо проверить множество комбинаций данных и условий.

4. Снижение затрат на тестирование

Хотя первоначальные затраты на разработку автоматизированных тестов могут быть высокими, в долгосрочной перспективе автоматизация помогает сократить затраты на тестирование за счет уменьшения времени на повторные проверки и возможность многократного использования тестов.

5. Поддержка непрерывной интеграции и непрерывного развертывания (CI/CD)

В рамках процессов CI/CD автоматизированное тестирование играет ключевую роль в поддержке непрерывной проверки качества. Оно позволяет быстро обнаруживать и исправлять дефекты, интегрируя тестирование в процесс сборки и развертывания приложений.

6. Возможность проведения сложных тестов

Некоторые тесты, такие как нагрузочное и стресс-тестирование, требуют выполнения большого количества итераций и обработки большого объема данных. Автоматизация делает такие тесты более выполнимыми и управляемыми, позволяя тестировщикам сосредоточиться на интерпретации результатов и анализе производительности.

7. Увеличение возможностей тестирования

Автоматизация позволяет проводить тесты в разных средах, на разных устройствах и браузерах, обеспечивая кросс-платформенную совместимость и улучшая общее качество продукта. Это может включать в себя тестирование различных версий браузеров, операционных систем и мобильных устройств.

8. Быстрая обратная связь

Автоматизированные тесты могут быть выполнены быстро, что обеспечивает тестировщикам быструю обратную связь о качестве приложения. Это помогает выявлять и устранять дефекты на ранних стадиях разработки, что повышает эффективность процесса тестирования.

9. Повторяемость и надежность

Автоматизированные тесты могут быть запущены неограниченное количество раз без необходимости вручную повторять тестовые сценарии, что обеспечивает повторяемость и стабильность результатов. Это особенно полезно для тестирования больших и сложных систем.

10. Документация и отчетность

Автоматизированные тесты обычно сопровождаются подробными отчетами о результатах, что упрощает отслеживание дефектов и помогает в дальнейшем анализе. Эти отчеты могут включать в себя информацию о пройденных и не пройденных тестах, а также детализированную информацию о найденных дефектах.

Автоматизированное тестирование не заменяет полностью ручное тестирование, но значительно расширяет возможности тестировщиков и повышает общую эффективность процесса тестирования. Оно помогает сосредоточиться на более сложных задачах и обеспечивает более высокий уровень уверенности в качестве тестируемого программного обеспечения.

В процессе автоматизации тестирования выполняются следующие шаги:

1. Выбор тестового инструмента;
2. Определение объема автоматизации;
3. Планирование, дизайн и разработка;
4. Выполнение теста;
5. Техническое обслуживание.

Выбор тестового инструмента мы уже рассмотрели в предыдущей практической работе. Теперь поговорим о дальнейших шагах.

Объем автоматизации тестирования — это область тестируемого приложения, которая подлежит автоматизации. Определение объема автоматизации играет ключевую роль в

успешном планировании и реализации автоматизированного тестирования, поскольку оно помогает сосредоточиться на наиболее критичных и ресурсозатратных аспектах приложения. Чтобы эффективно определить объем автоматизации, необходимо учитывать следующие пункты:

1. **Функции, важные для бизнеса**: Приоритет должен быть отдан функциям, которые критичны для бизнеса и имеют высокое значение для пользователей. Автоматизация таких функций обеспечивает стабильность ключевых бизнес-процессов и снижает риски, связанные с их нарушением.
2. **Сценарии с большим объемом данных**: Тесты, которые требуют обработки больших объемов данных, могут быть трудоемкими и сложными для ручного выполнения. Автоматизация таких сценариев позволяет эффективно и быстро проверять работу приложения при различных объемах данных.
3. **Общие функции приложений**: Функции, которые используются в различных частях приложения или на разных этапах пользовательского взаимодействия, также должны быть включены в автоматизацию. Это включает в себя базовые функции, такие как вход в систему, регистрацию пользователей и основные операции, которые выполняются часто.
4. **Техническая осуществимость**: оцените, насколько технически осуществима автоматизация выбранных функций. Это включает в себя анализ доступности API, возможность взаимодействия с интерфейсами приложения и наличие необходимых инструментов и ресурсов для автоматизации.
5. **Частота повторного использования бизнес-компонентов**: Функции, которые используются многократно или часто повторяются в различных тестовых сценариях, следует автоматизировать, чтобы повысить эффективность тестирования и сократить время на их проверку.
6. **Сложность тестовых случаев**: Более сложные тестовые случаи, которые требуют много времени на ручное выполнение и проверку, могут быть хорошими кандидатами для автоматизации. Это позволяет сократить количество ошибок и повысить точность тестирования.
7. **Возможность использовать одни и те же тестовые сценарии для кросс-браузерного тестирования**: Если ваше приложение должно работать в различных браузерах, автоматизация тестов, которые можно использовать для проверки кросс-браузерной совместимости, поможет сэкономить время и ресурсы.

Планирование, Проектирование и Разработка

После определения объема автоматизации следует этап планирования, проектирования и разработки. На этом этапе создается стратегия и план автоматизации, которые содержат следующие ключевые детали:

1. **Выбранные инструменты автоматизации**: Определите, какие инструменты будут использоваться для автоматизации тестирования. Выбор инструментов зависит от различных факторов, включая тип тестируемого приложения, доступные ресурсы и требуемые функции.
2. **Особенности тестируемой системы**: Понимание особенностей тестируемой системы, таких как её архитектура, используемые технологии и интерфейсы, поможет в создании эффективных тестов и выборе подходящих инструментов для автоматизации.
3. **Входящие и выходящие за рамки элементы автоматизации**: Установите, какие элементы будут включены в автоматизацию, а какие останутся вне её рамок. Это поможет четко определить границы проекта и избежать ненужного усложнения.
4. **Подготовка сценариев автоматизации**: Создайте и разработайте сценарии автоматизации на основе требований к тестированию. Это включает в себя написание тест-кейсов, подготовку данных и настройку окружения для выполнения тестов.
5. **График и временная шкала сценариев и выполнения**: Разработайте график выполнения тестов и временную шкалу, чтобы обеспечить согласованность и планомерность работы. Это поможет организовать процесс тестирования и следить за его ходом.
6. **Результаты тестирования автоматизации**: Определите, как будут оцениваться результаты автоматизированных тестов. Это включает в себя создание отчетов, анализ результатов и внесение необходимых изменений в тестовые сценарии на основе полученных данных.

Таким образом, тщательное планирование и проектирование автоматизации тестирования помогают организовать процесс, определить приоритетные области для автоматизации и обеспечить успешное внедрение автоматизированных тестов в процесс разработки.

ЗАДАНИЯ ДЛЯ ВЫПОЛНЕНИЯ

1. Изучите выбранный инструмент тестирования (официальный сайт, сообщество, документацию). Предоставьте ссылку на сайт поставщика выбранного инструмента;

2. Определите объем автоматизации и подготовьте сценарии тестирования, ориентированные на выбранный инструмент, в виде тест кейсов (минимум 7 тест-кейсов). Составьте график выполнения сценариев;
3. Протестируйте существующий функционал системы по написанным тест кейсам, оформите все найденные дефекты в Yougile.

ВОПРОСЫ

1. Что такое автоматизированное тестирование и какие его основные цели?
2. Какие преимущества и недостатки автоматизированного тестирования?
3. Какие типы тестов чаще всего автоматизируются и почему?
4. Какие инструменты автоматизации тестирования вы знаете и какие из них предпочитаете? Почему?
5. Какой процесс разработки и выполнения тестов в автоматизированном тестировании?
6. Как определить, какие тесты следует автоматизировать?
6. Что такое план автоматизации тестирования и что он должен содержать?
7. Как можно оценить техническую осуществимость автоматизации тестов?
8. Как разработать эффективные тестовые сценарии для автоматизации?
10. Какие шаги включены в разработку и настройку автоматизированных тестов?
11. Как анализировать и интерпретировать результаты автоматизированного тестирования?
12. Какие методы используются для отладки автоматизированных тестов?
13. Как обеспечить поддержку и обновление автоматизированных тестов?
14. Как управлять отчетами о тестировании и документировать результаты?
15. Какие проблемы могут возникнуть при автоматизации тестирования и как их можно решить?

Практическая работа №7. Открытое бета-тестирование

ЦЕЛЬ РАБОТЫ

Изучить принципы бета-тестирования. Отобрать аудиторию для бета-тестирования и провести тестирование продукта.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Бета-тестирование— интенсивное использование почти готовой версии продукта (как правило, программного или аппаратного обеспечения) с целью выявления максимального числа ошибок в его работе для их последующего устранения перед

окончательным выходом продукта на рынок, к массовому потребителю. Бета-тестирование – это этап приемочного тестирования продукта.

Это важное тестирование, которое проверяет программное обеспечение его реальными потенциальными пользователями. Чтобы определить, бета-тестирование — это тестирование, которое тестировщики из организации заказчика проводят в своей собственной бизнес-среде или производственной среде перед тем, как использовать продукт в реальных бизнес-целях. Подводя итог, можно сказать, что идея бета-тестирования состоит в том, чтобы оценить, полезна ли новая система для бизнеса.

Когда проводить бета-тестирование? Реальные пользователи или клиенты проводят бета-тестирование как заключительный этап тестирования продукта на своем собственном сайте. Поэтому они выполняют это после завершения всех тестов, таких как модульное тестирование, интеграция, системные тесты, такие как регрессия, производительность и альфа. Бета-тестирование также является частью приемочного тестирования, такого как альфа-тестирование. После бета-тестирования происходит развертывание продукта для реального использования в бизнесе.

Кто проводит бета-тестирование? Любое из следующих может выполнять бета-тестирование

1. Внешние пользователи (Клиенты или Конечные пользователи)

- **Кто это:** Рядовые пользователи или клиенты, которые могут стать потенциальными покупателями или пользователями продукта.
- **Роль:** они используют продукт в реальных условиях, что помогает выявить проблемы, которые могли быть упущены внутренними тестировщиками. Их отзывы и замечания касаются функциональности, удобства использования и совместимости продукта с реальными условиями.
- **Особенности:** Пользователи могут быть из числа тех, кто подписался на программу бета-тестирования, или выбраны по определённым критериям (например, их опыт с аналогичными продуктами).

2. Фокус-группы

- **Кто это:** специально отобранные группы пользователей, представляющие целевую аудиторию продукта.
- **Роль:** они проводят более целенаправленные тесты, оценивают продукт с точки зрения определённых характеристик или сценариев использования. Фокус-группы могут состоять из профессионалов, специалистов или представителей определённых организаций.

- **Особенности:** Фокус-группы могут предоставлять более структурированную обратную связь и участвовать в более глубоком анализе продукта.

3. Внутренние сотрудники (Внутренние бета-тестеры)

- **Кто это:** Сотрудники компании-разработчика или её партнёров, которые не участвовали в разработке продукта, но обладают хорошим пониманием его функциональности.
- **Роль:** Внутренние сотрудники проводят тестирование с точки зрения пользователей, не вовлечённых непосредственно в разработку. Они помогают выявить внутренние проблемы, не замеченные предыдущими тестами.
- **Особенности:** это может включать сотрудников из разных отделов, таких как поддержка пользователей, маркетинг или продажи, которые могут предоставить ценные мнения о том, как продукт будет восприниматься на рынке.

4. Партнёрские организации

- **Кто это:** Компании или организации, которые сотрудничают с разработчиком и могут использовать продукт в рамках своих бизнес-процессов.
- **Роль:** они могут предоставить обратную связь, основанную на том, как продукт интегрируется в их систему или как он улучшает их рабочие процессы.
- **Особенности:** Такие организации могут использовать продукт в более сложных и специфичных сценариях, что может помочь выявить проблемы, которые не были замечены в других тестах.

Бета-тестирование предлагает несколько преимуществ в обеспечении соответствия продукта требованиям бизнеса.

- Это первый этап тестирования, на котором продукт оценивается в реальной бизнес-среде с фактическими производственными данными.
- Он выявляет ошибки, проблемы или проблемы еще до того, как продукт будет запущен. Несомненно, это имеет решающее значение для общего успеха продукта.
- Бета-тестирование проверяет продукт с точки зрения реального пользователя. Таким образом, он дает обратную связь на раннем этапе, нравится ли это пользователям.
- Хорошие результаты бета-тестирования повышают доверие всех заинтересованных сторон, предоставляя потенциальным пользователям четкое представление о производительности продукта.

Бета-тестирование может напрямую получить отзывы пользователей о продукте. Независимо от того, насколько хороший продукт делает группа разработчиков и сколько критических ошибок обнаруживает команда тестирования, продукт бесполезен, если он не

нравится пользователям. Подводя итог, можно сказать, что бета-тестирование оценивает, соответствует ли продукт его бизнес-целям и может ли он быть запущен в производственной среде.

ЗАДАНИЯ ДЛЯ ВЫПОЛНЕНИЯ

1. Определить потенциальных пользователей системы, дать данным пользователям характеристику. Например: потенциальными пользователями приложения являются, менеджеры по направлению товара, торговые представители, представители обслуживающего персонала и т.д. Примерные профили некоторых из названных категорий пользователей могут выглядеть следующим образом.

Таблица 2 - Профили пользователей системы.

Пользователи	Менеджер по направлению товара	Представители обслуживающего персонала
Социальные характеристики	Мужчины, женщины Взрослые Русскоязычные Средний уровень владения компьютером	Женщины Взрослые Русскоязычные Низкий уровень владения компьютером
Мотивационно целевая среда	Прямая производственная необходимость, удобство Мотивация к обучению высокая	Производственная необходимость, Престиж Мотивация к обучению низкая
Навыки и умения	Должны иметь значительный тренинг работы с программой	Прошли предварительный тренинг работы с программой
Требования к ПО ИС	Возможность использования ПО ИС в локальной сети Отсутствие жестких ограничений по времени Обеспечение текущей информацией по содержанию заказов Обеспечение текущей информацией по товарам Возможность проводить обобщение информации по заказам	Возможность использования программы одновременно с телефонным общением с клиентом Время реакции ПО ИС, допустимое для ожидания клиента Обеспечение текущей информацией по содержанию заказов Обеспечение текущей информацией по товарам Возможность формирования новых заказов
Задачи пользователя	Просмотр/фильтрация информации по заказам/клиентам/товарам	Просмотр данных по товарам Создание/поиск/модификация заказа Сохранение/печать заказа

	Сортировка информации по заказам/клиентам/товарам Агрегирование информации по заказам/клиентам/товарам	Формирование счета по заказу
Рабочая среда	Стандартизированные ПК, локальная сеть	Стандартизированные ПК, специализированное телефонное обслуживание

2. На основании первого пункта отобрать 3 кандидатов. Предоставить краткую информацию о кандидате: фамилию и имя, какую роль будет выполнять в системе (например: тестировать функционал администратора), навыки и умения кандидата.
3. Предоставить кандидатам доступ к системе: создать учетную запись, настроить доступы, предоставить сценарий тестирования.
4. Провести бета-тестирование, составить анкету для кандидатов и получить отзывы пользователей. Возможные критерии анкеты:
    a. Удобство интерфейса;
    b. Наполнение контента;
    c. Скорость работы приложения;
    d. Соответствует ли разработанное приложение заявленным требованиям (можно перечислить требования);
    e. Свои комментарии, если имеются.
5. Оформить найденные дефекты в Yougile.

ВОПРОСЫ

1. Что такое бета-тестирование?
2. Каковы ключевые этапы бета-тестирования?
3. Кто обычно участвует в бета-тестировании?
4. Каковы основные виды бета-тестирования?
5. Как определить успешность бета-тестирования?
6. Какие проблемы могут возникнуть в процессе бета-тестирования?
7. Какой тип обратной связи должен быть собран в ходе бета-тестирования?
8. Какие инструменты и методы могут быть использованы для бета-тестирования?
9. Какие документы или материалы необходимы для успешного проведения бета-тестирования?
10. Как интегрировать результаты бета-тестирования в процесс разработки?

11. Как обеспечить безопасность и конфиденциальность данных в ходе бета-тестирования?
12. Что делать после завершения бета-тестирования?
13. Как управлять ожиданиями участников бета-тестирования?
14. Какие преимущества и недостатки есть у бета-тестирования?

Практическая работа №8. Тестирование установки и приемочное тестирование.

ЦЕЛЬ РАБОТЫ

Провести тестирование установки и приемочное тестирование разработанного продукта.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Тестирование установки позволяет удостовериться в том, что система корректно устанавливается и настраивается, обновление новых версий происходит без ошибок. Тестирование установки (или инсталляционное тестирование) — это важный этап обеспечения качества программного обеспечения, целью которого является проверка корректности процесса установки программного продукта. Оно помогает удостовериться, что программное обеспечение устанавливается, настраивается и работает должным образом в целевых средах. Вот как оно проводится:

- Подготовка к тестированию:
  - Определение требований: Сбор и анализ требований к установке, которые могут быть указаны в документации к продукту.
  - Создание тест-плана: Определение сценариев установки, условий тестирования, инструментов и ресурсов, необходимых для тестирования установки.
  - Настройка тестовой среды: Подготовка тестовых систем, включая операционные системы, аппаратное обеспечение и необходимые зависимости.
- Виды тестирования установки:
  - Тестирование установки:
    - Простое и сложное: Проверка установки как в стандартных, так и в нестандартных условиях.
    - Полная установка: Проверка полной установки со всеми компонентами и функциями.
    - Минимальная установка: Проверка установки только базового набора функций.
    - Обновление: Проверка возможности обновления с предыдущих версий.

    - Удаление и переустановка: Проверка корректности удаления приложения и последующей переустановки.
  - Тестирование деинсталляции:
    - Полное удаление: Проверка, удаляет ли процесс все файлы, записи в реестре и другие компоненты.
    - Остаточные файлы: Проверка на наличие оставшихся файлов или записей после деинсталляции.
- Выполнение тестов:
  - Запуск установки: Запуск процесса установки и выполнение сценариев установки в соответствии с тест-планом.
  - Верификация: Проверка, что приложение установлено корректно, а все компоненты работают согласно требованиям.
  - Тестирование на разных конфигурациях: Проверка установки на разных операционных системах, версиях, языках и аппаратных конфигурациях.
- Документирование результатов:
  - Сбор данных: Регистрация всех обнаруженных проблем, ошибок или несоответствий.
  - Создание отчетов: Формирование отчетов о результатах тестирования, включая найденные дефекты, рекомендации по исправлению и улучшению установки.
- Анализ и устранение проблем:
  - Анализ дефектов: Исследование причин проблем, обнаруженных в процессе установки.
  - Корректировка: Внесение изменений в установочный пакет или процесс установки для устранения выявленных проблем.
- Проверка повторного тестирования:
  - Повторное тестирование: Проверка после внесения исправлений, чтобы убедиться, что проблемы были устранены и не возникли новые.
- Завершение тестирования:
  - Окончательная верификация: Подтверждение, что процесс установки завершен успешно и все требования выполнены.
  - Анализ тестов: Оценка результатов тестирования и внедрение рекомендаций в процесс установки для будущих релизов.

Тестирование инсталляции необходимо проводить при создании систем, после появления новой версии, а также при изменении конфигурации стенда.

Инсталляционное тестирование рекомендуется проводить на разных платформах, ручным методом или с помощью автоматизации. На данный тип работ по тестированию влияют следующие факторы:

- Какие платформы и операционные системы поддерживаются?
- Каким образом будет распространяться программное обеспечение?
- Кто будет устанавливать программное обеспечение?

Тестирование установки позволяет избежать таких проблем, как:

- Невозможность установить систему;
- Потеря данных после установки новой версии;
- Невозможность откатиться до предыдущей версии.

Приемочное тестирование – вид тестирования, проводимый на этапе сдачи готового продукта (или готовой части продукта) заказчику.

Целью приемочного тестирования является определение готовности продукта, что достигается путем прохода тестовых сценариев и случаев, которые построены на основе спецификации требований к разрабатываемому продукту.

Результатом приемочного тестирования может стать:

- Отправка проекта на доработку.
- Принятие его заказчиком, в качестве выполненной задачи.

Это финальный этап тестирования продукта перед его релизом. При этом, он не является особо тщательным, всеохватывающим и полным – тестируется только основной функционал.

ЗАДАНИЯ ДЛЯ ВЫПОЛНЕНИЯ

1. Подготовить информацию об окружении системы (какие платформы и операционные системы поддерживаются, кто будет заниматься установкой системы);
2. Составить подробную инструкцию развертывания системы, например:
    a. Код приложения для клиентской и серверной части хранится на GitHub;
        i. Клиентская часть: https://github.com/...
        ii. Серверная часть: https://github.com/...
    b. Создаем пустую папку на компьютере и клонируем в неё приложение используя следующую команду:
    git clone https://github.com/... .
    c. Открываем консоль и переходим в корневую папку проекта;

d. Запускаем команду *npm i*, данная команда установит все необходимые для работы системы зависимости;
e. Запускаем систему командой *npm start*;
f. Открываем браузер и переходим на localhost:3000.

3. Следуя составленной инструкции протестировать установку приложения, оформить найденные дефекты в Yougile, сделать вывод по тестированию установки;
4. Составить список основных функций системы (можно оформить в виде чек листа) в Yougile;
5. Протестировать основные функции системы, оформить найденные дефекты в Yougile;
6. Сделать вывод, готов ли проект к использованию или нужна доработка. Поставить предварительные сроки выпуска продукта и перечень функционала для доработки (если имеется).

ВОПРОСЫ

1. Каковы основные этапы подготовки к тестированию установки?
2. Какие виды тестирования установки описаны в тексте?
3. Какое тестирование проводится для проверки корректности удаления приложения?
4. Что должно быть проверено в процессе выполнения тестов установки?
5. Какие действия следует предпринять после обнаружения дефектов в процессе тестирования установки?
6. Какие факторы следует учитывать при проведении инсталляционного тестирования?
7. Какие проблемы могут быть предотвращены с помощью тестирования установки?
8. Что такое приемочное тестирование и на каком этапе оно проводится?
9. Какая цель приемочного тестирования?
10. Какие результаты могут быть достигнуты в результате приемочного тестирования?
11. Что тестируется в процессе приемочного тестирования?
12. Какие шаги нужно выполнить для подготовки информации об окружении системы?
13. Какие этапы включены в инструкцию развертывания системы?
14. Что нужно сделать после составления инструкции развертывания системы?
15. Какой вывод следует сделать после тестирования основных функций системы?

СПИСОК ЛИТЕРАТУРЫ

1. Старолетов, С. М. Основы тестирования и верификации программного обеспечения / С. М. Старолетов. — 3-е изд., стер. — СПб : Лань. URL: https://e.lanbook.com/book/319445 (дата обращения: 10.03.2025).
2. Морозова, Ю. В. Тестирование программного обеспечения : учебное пособие / Ю. В. Морозова. - Томск : Эль-Контент. URL: https://znanium.com/catalog/product/1845910 (дата обращения: 10.03.2025).
3. Игнатьев, А. В. Тестирование программного обеспечения / А. В. Игнатьев. — 3-е изд., стер. — СПб : Лань. URL: https://e.lanbook.com/book/269873 (дата обращения: 10.03.2025).