**Проект №2 — Тестовые артефакты**

Привет, участник Школы 21!😊 Наверное, тебе уже не терпится по-настоящему что-нибудь протестировать, поэтому уже в этом проекте мы подготовили для тебя практические задания, в результате выполнения которых ты составишь небольшой план и с его помощью проведёшь тестирование демо-версии одного веб-приложения.

**Instructions**

Напоминаем, что все отчёты по результатам выполнения заданий тебе придется оформлять в файлах с расширением .md. Все созданные отчёты и файлы тебе нужно будет загрузить в папку src/ в корне проекта (обязательно в ветку *develop*). Если они уже созданы, то пересоздавать или удалять их не нужно (просто отредактируй этот файл).

**Contents**

1. [Chapter I](https://edu.21-school.ru/project/62664/task#chapter-i)  
   1.1. [Общая инструкция](https://edu.21-school.ru/project/62664/task#%D0%BE%D0%B1%D1%89%D0%B0%D1%8F-%D0%B8%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D0%B8%D1%8F)
2. [Chapter II](https://edu.21-school.ru/project/62664/task#chapter-ii)  
   2.1. [Чёрный и белый ящики](https://edu.21-school.ru/project/62664/task#%D1%87%D1%91%D1%80%D0%BD%D1%8B%D0%B9-%D0%B8-%D0%B1%D0%B5%D0%BB%D1%8B%D0%B9-%D1%8F%D1%89%D0%B8%D0%BA%D0%B8)  
   2.2. [Задание №1](https://edu.21-school.ru/project/62664/task#%D0%B7%D0%B0%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-1-%D1%81%D0%B5%D1%80%D1%8B%D0%B9-%D1%8F%D1%89%D0%B8%D0%BA)  
   2.3. [Позитивное и негативное тестирование](https://edu.21-school.ru/project/62664/task#%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5-%D0%B8-%D0%BD%D0%B5%D0%B3%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5-%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)  
   2.4. [Задание №2](https://edu.21-school.ru/project/62664/task#%D0%B7%D0%B0%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-2-%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA-%D1%82%D0%B5%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D1%85-%D1%81%D1%86%D0%B5%D0%BD%D0%B0%D1%80%D0%B8%D0%B5%D0%B2)
3. [Chapter III](https://edu.21-school.ru/project/62664/task#chapter-iii)  
   3.1. [Основные тестовые артефакты](https://edu.21-school.ru/project/62664/task#%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%BD%D1%8B%D0%B5-%D1%82%D0%B5%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B5-%D0%B0%D1%80%D1%82%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D1%8B)  
   3.2. [Тест-кейс](https://edu.21-school.ru/project/62664/task#%D1%82%D0%B5%D1%81%D1%82-%D0%BA%D0%B5%D0%B9%D1%81)  
   3.3. [Задание №3](https://edu.21-school.ru/project/62664/task#%D0%B7%D0%B0%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-3-%D1%82%D0%B5%D1%81%D1%82-%D0%BA%D0%B5%D0%B9%D1%81%D1%8B)  
   3.4. [Баг-репорт](https://edu.21-school.ru/project/62664/task#%D0%B1%D0%B0%D0%B3-%D1%80%D0%B5%D0%BF%D0%BE%D1%80%D1%82)
4. [Chapter IV](https://edu.21-school.ru/project/62664/task#chapter-iv)  
   4.1. [Чек-лист и тестовый набор](https://edu.21-school.ru/project/62664/task#%D1%87%D0%B5%D0%BA-%D0%BB%D0%B8%D1%81%D1%82-%D0%B8-%D1%82%D0%B5%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9-%D0%BD%D0%B0%D0%B1%D0%BE%D1%80)  
   4.2. [Тест-план](https://edu.21-school.ru/project/62664/task#%D1%82%D0%B5%D1%81%D1%82-%D0%BF%D0%BB%D0%B0%D0%BD)  
   4.3. [Задание №4](https://edu.21-school.ru/project/62664/task#%D0%B7%D0%B0%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-4-%D1%87%D0%B5%D0%BA-%D0%BB%D0%B8%D1%81%D1%82)

**Chapter I**

**Общая инструкция**

Методология Школы 21 может быть не похожа на тот образовательный опыт, который случался с тобой ранее. Её отличает высокий уровень автономии: у тебя есть задача, ты должен её выполнить. По большей части тебе нужно будет самому добывать знания для её решения. Второй важный момент — это peer-to-peer обучение. В образовательном процессе нет менторов и экспертов, перед которыми ты защищаешь свой результат. Ты это делаешь перед такими же учащимися, как и ты сам. У них есть чек-лист, который поможет им качественно выполнить приемку вашей работы.

Роль Школы 21 заключается в том, чтобы обеспечить через последовательность заданий и оптимальный уровень поддержки такую траекторию обучения, при которой ты не только освоишь hard skills, но и научишься самообучаться.

* Не доверяй слухам и предположениям о том, как должно быть оформлено ваше решение. Этот документ является единственным источником, к которому стоит обращаться по большинству вопросов;
* твое решение будет оцениваться другими учащимися;
* подлежат оцениванию только те файлы, которые ты выложил в GIT (ветка develop, папка src);
* в твоей папке не должно быть лишних файлов — только те, что были указаны в задании;
* не забывай, что у вас есть доступ к интернету и поисковым системам;
* обсуждение заданий можно вести и в Rocket.Chat;
* будь внимателен к примерам, указанным в этом документе — они могут иметь важные детали, которые не были оговорены другим способом;
* и да пребудет с тобой Сила!

**Chapter II**

В предыдущем проекте ты уже познакомился с некоторыми видами тестирования, но, конечно, это ещё далеко не всё. Перед тем как приступить к рассмотрению тестовых артефактов, давай изучим ещё несколько видов тестирования, с которыми тебе предстоит столкнуться.😉

**Чёрный и белый ящики**

**"Белым ящиком"** называется метод тестирования программного обеспечения, который предполагает, что внутренняя структура системы известна тестировщику. Входные значения выбираются, основываясь на знании кода, который будет их обрабатывать. Аналогично, мы знаем, каким должен быть результат этой обработки. Знание всех особенностей тестируемой программы и ее реализации обязательны для этого подхода. Тестирование белого ящика основано на углублении во внутреннее устройство системы.

Несложно догадаться, что **"чёрным ящиком"** называется метод тестирования, напротив, основанный на работе только с внешними интерфейсами системы, без какой-либо информации о ее внутреннем устройстве. Именно этим методом ты будешь пользоваться чаще всего, если мы говорим о мануальном (ручном) тестировании пользовательского интерфейса.

**Задание №1. Серый ящик**

А что такое **"серый ящик"**? В каком случае он используется?🧐 Самостоятельно изучи эти вопросы. Ответы на них запиши в файл exercise1.md, предварительно его создав.

**Позитивное и негативное тестирование**

Тестирование делят на **позитивное** и **негативное**.

**Позитивное тестирование** подразумевает проверку системы с помощью сценариев, соответствующих её *ожидаемому* поведению. С его помощью проверяется, что система делает то, для чего была создана.

В **негативном тестировании** используются сценарии, соответствующие *внештатному* поведению программы, то есть мы проверяем наше приложение на заведомо неверных входных данных, которые могли бы вызвать сбои. Негативное тестирование считается пройденным лишь в том случае, если подобные тест-кейсы не вызывают никаких сбоев в приложении.

*P.S.: тут под сценариями понимаются тестовые сценарии (тест-кейсы), о них мы поговорим чуть позже в этом же проекте.*

**Задание №2. Список тестовых сценариев**

Перейди по ссылке <https://www.saucedemo.com/>. Ознакомься с функциональными возможностями демо-версии интернет-магазина. На странице авторизации снизу написан логин и пароль. Постарайся описать своими словами в файле exercise2.md (создай этот файл!) список сценариев позитивного тестирования так, чтобы они были как можно более исчерпывающими для проверки всех функциональных возможностей (также не забудь про страницу авторизации). В данном задании под сценариями тестирования понимается обычный набор действий, который необходимо выполнить, чтобы проверить отдельную функцию в приложении. Например:

Сценарий №1. Проверка успешного прохождения авторизации:

1. Перейти на страницу авторизации

2. В поле "Username" ввести "standard\_user"

3. В поле "Password" ввести "secret\_sauce"

4. Нажать на кнопку "LOGIN"

Как ты уже понял, мы рассматриваем только позитивные сценарии (не нужно проверять случай, когда мы вводим неправильный пароль из тысячи символов).

Полученный список помести под заголовок # Список тестовых сценариев.

*P.S.: список сценариев должен затрагивать только демонстрационный сайт "saucedemo"*

**Chapter III**

**Основные тестовые артефакты**

В ходе тестирования создаются и используются различные тестовые артефакты (например: документы, модели, дизайны, рисунки и т.д.). Наиболее распространёнными из них являются:

* **Тест-кейс** — последовательность действий, по которой можно проверить, соответствует ли тестируемая функция установленным требованиям.
* **Баг-репорт** — документ, описывающий несоответствие реальной работы программы с предъявленными к ней требованиями.
* **Тест-план** — это артефакт тестирования, описывающий действия, которые будут происходить в процессе тестирования — от разработки стратегии до критериев поиска ошибок. Он также описывает логику завершения задач и оценку рисков со сценариями их разрешения.
* **Чек-лист** — список необходимых проверок.

А теперь давай познакомимся с каждым из них подробнее!😁

**Тест-кейс**

**Тест-кейс** — это тестовый артефакт, суть которого заключается в выполнении некоторого количества действий и условий, необходимых для проверки определенного функционала разрабатываемого ПО.

Из чего состоит тест-кейс? Типичную структуру можно представить вот так:

* ID кейса (уникальный идентификатор, который может быть числовым, буквенным, буквенно-числовым. Пока что мы будем просто их нумеровать)
* Заголовок
* Шаги (полная последовательность действий)
* Ожидаемый результат
* Предшествующие условия

То есть если ранее ты составлял просто заголовок и сценарии, которые являлись последовательностью действий, то теперь их нужно дополнить нумерацией, ожидаемым результатом и предшествующими условиями, чтобы эти сценарии превратились в тест-кейсы.

**Задание №3. Тест-кейсы**

Создай файл exercise3.md. На основе сценариев, которые ты получил в результате выполнения задания №2, составь тест-кейсы (дополнив сценарий ожидаемым результатом, предшествующими условиями), пронумеруй каждый тест-кейс.

Пример тест-кейса:

## Тест-кейс №1 — Нажатие кнопки "Избранное"

1. Предшествующие условия:

- пользователь авторизован

- открыта новостная лента

2. Шаги:

- навести курсор на запись

- нажать на появившуюся кнопку "Избранное"

3. Ожидаемый результат:

- при наведении появляется кнопка "Избранное"

- при нажатии на кнопку "Избранное", запись добавляется в избранные публикации

- к записи добавляется иконка "звёздочки"

**Баг-репорт**

Баг-репорт описывает найденный дефект. В связи с этим актуален вопрос "Что вообще является багом?". Объективного ответа на этот вопрос, как правило, нет. При решении этого вопроса в первую очередь стоит проверить требования к ПО. Если проверка требований, имеющейся базы знаний или технического задания не дала четкого ответа на вопрос, следует подумать о баге с перспективы пользовательского опыта: если пользователю может показаться что данное поведение системы — ошибка... скорее всего, так оно и есть. Если совсем просто, под багом мы будем понимать ошибку, когда система не соответствует заявленным требованиям.

Из чего состоит (должен состоять) баг-репорт? Всё чем-то напоминает тест-кейс, только дополняется несколькими полями:

* Заголовок
* Предшествующие условия
* Серьёзность
* Шаги воспроизведения
* Ожидаемый результат
* Фактический результат
* Окружение

**Заголовок** — краткое описание сути бага. Заголовок следует писать, основываясь на принципе "Что? Где? Когда?".

**Что?** — Что происходит/не происходит согласно документации/представлению о нормальной работе продукта.

**Где?** — В каком элементе ПО возникает дефект.

**Когда?** — При каких действиях возникает дефект.

При написании заголовка стоит избегать слов "Плохо", "Неправильно" и похожих, так как они неточно описывают суть проблемы.

Шаги для воспроизведения должны быть четко и подробно описаны. Их задача — перечислить все действия, необходимые, чтобы воспроизвести баг, они должны быть понятны не только пишущему их человеку, но и любому другому читателю баг-репорта.

Более подробно степени серьёзности багов мы обсудим с тобой в проекте №4. А пока давай запомним, что выделяют Blocker, Critical, Major, Minor & Trivial степени серьёзности багов. Не стоит путать приоритетность задачи с серьёзностью бага. Например, приоритетность у задачи может иметь значение Low или вообще Backlog (то есть решение проблемы откладывается "в долгий ящик"), а серьёзность — Critical.

Теперь посмотрим на пример баг-репорта:

Заголовок:

Краш приложения при смене языка на арабский в настройках языка

Предшествующие условия:

Пользователь авторизован в приложении.

Серьёзность:

Critical

Шаги для воспроизведения:

1. Открыть меню настроек

2. Зайти в настройки языка

3. Изменить язык на арабский

Ожидаемый результат:

Язык текста в приложении меняется на арабский

Фактический результат:

Приложение падает с ошибкой, отображается сообщение об ошибке.

Окружение:

Версия 1.0, IPhone4/IOS 9.1

Также по возможности к баг-репорту следует приложить видео, скриншоты, логи, дампы дистрибутивов и любые другие относящиеся к багу материалы. Важно помнить и знать о принципе "Один дефект — один репорт". Если на проблему уже заведен баг-репорт, второй заводить не нужно, это только создаст замешательство о количестве багов в системе. С другой стороны, если багов несколько, даже если они возникают в одном и том же окне и имеют общие черты, не стоит пытаться описать их всех одним репортом, лучше завести по репорту на каждый.

Баг-репорт — ключевой тестовый артефакт. Любой найденный баг желательно сразу оформлять и отправлять разработчику именно в формате баг-репорта. В этом проекте вряд ли удастся найти баг, однако в будущих — обязательно!😉

**Chapter IV**

**Чек-лист и тестовый набор**

**Чек-лист** — это список проверок, которые нужно осуществить на тестируемом ресурсе. От тест-кейса чек-лист отличается степенью подробности. Для использования чек-листа при тестировании очень много информации нужно держать в голове в момент прогона тестов и знать логику работы приложения (тут не расписываются сами тестовые случаи). Иными словами, чек-листы описывают кратко то, что хотим протестировать.

**Тестовый набор (Test Suite)** — набор тест-кейсов, предназначенных для тестирования программы. Обычно используется для того, чтобы объединить несколько тест-кейсов, проверяющих один и тот же функционал, в один набор. Его не стоит путать с чек-листом, Test Suite является обычным объединением тест-кейсов для удобства их выполнения и отслеживания.

Если ты объединишь все полученные в прошлом задании тест-кейсы в один документ, то получишь тестовый набор, а если к тому же уберешь всё лишнее (шаги выполнения, ожидаемое поведение) и будешь держать всё в голове, то ты получишь чек-лист — список проверок, которые нужно провести для проверки приложения.

**Тест-план**

**Тест-план** — документ, описывающий весь объем работ по тестированию, начиная с описания тестируемых объектов, стратегии, расписания, критериев начала и окончания тестирования, до необходимого в процессе работы оборудования, специальных знаний, а также оценки рисков с вариантами их разрешения.

**Тест-план позволяет**:

* Приоритизировать задачи по тестированию;
* Выстроить стратегию тестирования, согласованную со всей командой;
* Вести учет всех требуемых ресурсов, как технических, так и человеческих;
* Планировать использование ресурсов на тестирование;
* Просчитать риски, возможные при проведении тестирования.

Подробнее мы с ним познакомимся в следующих проектах!😇

**Задание №4. Чек-лист**

В файле exercise4.md представлен шаблон будущего чек-листа и результаты его прогона. Кратко опиши в этом файле тест-кейсы, полученные в результате работы над прошлым заданием, и выполни каждый из них (вручную проверь [приложение](https://www.saucedemo.com/) на основании полученного чек-листа).

**Double-check**

Перед загрузкой выполненного проекта в репозиторий перепроверь наличие всех необходимых файлов, которые требовалось создать во время его выполнения:

exercise1.md

exercise2.md

exercise3.md

exercise4.md

💡 [Нажми здесь](https://forms.gle/MfA5wp92j3a4WnyU9) **чтобы отправить обратную связь по проекту**.