



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных
технологий

Отчет по практической работе №4

по дисциплине «Тестирование и верификация ПО»

Выполнил:

Студент группы ИКБО-03-22

Смирнов Д.А.

Проверил:

Ассистент

Овчинникова М.А.

Москва, 2024

Содержание

Цель работы	3
Участник: Осипов Андрей Константинович	4
Разработка плана тестирования	4
Автоматизация тестирования Web-приложения	8
Участник: Смирнов Даниил Анатольевич	11
Разработка плана тестирования	11
Автоматизация тестирования Web-приложения	15
Участник: Хохлинов Дмитрий Иванович	18
Разработка плана тестирования	18
Автоматизация тестирования Web-приложения	22
Участник: Чикунов Данила Сергеевич	24
Разработка плана тестирования	24
Автоматизация тестирования Web-приложения	27
Вывод	29
Список использованных источников	30

Цель работы

Освоить основные принципы разработки плана тестирования программного обеспечения и применить их на практике, а также научиться проводить автоматизированное тестирование Web-приложений.

Постановка задачи

Для реализации практической работы необходимо решить следующие задачи:

1. Анализ требований к приложению
2. Разработка плана тестирования
3. Подготовка тестовой среды
4. Автоматизация тестирования
5. Документирование результатов тестирования
6. Анализ и предоставление рекомендаций

Ход работы

Участник: Осипов Андрей Константинович

Разработка плана тестирования

В практической работе №3 было разработано приложение «Генератор мелодий по заданным аккордам». Ниже представлен план тестирования данного приложения

1. Идентификатор тестового плана

- Тестовый план для приложения "Генератор мелодий по заданным аккордам"
- Идентификатор: СТ-MelodyGenerator-001

2. Ссылки на используемые документы

- Документация приложения "Генератор мелодий по заданным аккордам"
- Полное руководство по тестированию приложений

3. Введение

- Цель.

Определить и документировать процесс тестирования приложения «Генератор мелодий по заданным аккордам» для обеспечения его качества и стабильности.

- Область применения.
 - Тестирование функциональности.
 - Тестирование пользовательского интерфейса.
 - Тестирование многопользовательских функций .

4. Тестируемые элементы

- **Пользовательский интерфейс и навигация**
 - Ввод и вывод данных (заданные аккорды)
 - Управление процессом (генерация мелодии)
- **Логика приложения**
 - Правила приложения (аккорды должны существовать)
- **Интеграция с дополнительными функциями**

- Возможность прослушать мелодию
- Обработка ошибок и исключений

5. Проблемы риска тестирования ПП

- Ошибки в логике приложения, неправильное определение существующих аккордов
- Утечки памяти или другие проблемы с производительностью

6. Особенности или свойства, подлежащие тестированию

- Функциональное тестирование.
 - Проверка соответствия процесса заданным правилам.
 - Тестирование установки и запуска приложения без проблем.
- Комбинаторное тестирование.
 - Проверка всех возможных вариантов значений параметров (например, различные аккорды и их последовательности).
- Тестирование выносливости.
 - Проверка приложения на генерацию длительных мелодий для выявления проблем с утечкой памяти и стабильностью.

7. Особенности (свойства), не подлежащие тестированию

- Внешний дизайн и графика, если они не влияют на функциональность приложения.

8. Подход

- Разработка тест-кейсов.

Создание позитивных и негативных тест-кейсов для покрытия всех аспектов приложения.

- Альфа-тестирование и бета-тестирование.

Проведение тестирования на разных этапах разработки для выявления и исправления ошибок.

- Автоматизированное тестирование.

Использование инструментов для автоматизации повторяющихся тестов, особенно для тестирования выносливости.

9. Критерии смоук-тестирования

- Проверка основной функциональности приложения (ввод и вывод данных, генерация мелодии).
- Успешный запуск и завершение процесса.
- Отсутствие критических ошибок, препятствующих продолжению работы приложения.

10. Критерии прохождения тестов

- Все тест-кейсы пройдены успешно без критических ошибок.
- Процесс соответствует заданным правилам и требованиям.
- Пользовательский интерфейс интуитивно понятен и удобен для использования.

11. Критерии приостановки и возобновления работ

- Обнаружение критических ошибок, которые невозможно исправить в текущем цикле тестирования.
- Недостаточная подготовка тестовой среды или необходимость дополнительных ресурсов.
- Возобновление работ после исправления выявленных ошибок и обеспечения необходимых ресурсов.

12. Тестовая документация

- Тест-кейсы и сценарии тестирования.
- Отчеты о результатах тестирования, включая видеоролики и скриншоты.
- Журнал дефектов с описанием ошибок и статусом их исправления.

13. Основные задачи тестирования

- Обеспечение соответствия процесса заданным правилам.
- Проверка стабильности и производительности приложения.
- Оценка пользовательского опыта и дизайна приложения.

14. Необходимый персонал и обучение

- Тестировщики с опытом функционального тестирования.
- Обучение по использованию автоматизированных инструментов

тестирования.

- Знакомство с правилами приложения.

15.Требования среды

- Тестовая среда, максимально приближенная к реальной.
- Необходимые серверные ресурсы для хранения пользовательских результатов генерации.
- Инструменты для автоматизированного тестирования и мониторинга производительности.

16.Распределение ответственности

- Тестировщики.

Выполнение тест-кейсов, документирование результатов.

- Разработчики.

Написание кода, исправление выявленных ошибок, поддержка тестировщиков.

- Менеджер проекта.

Координирование процессов, контроль сроков и ресурсов.

17.График работ (календарный план)

- Фаза подготовки. Срок – 1 неделя.
 - Разработка тест-кейсов
 - Настройка тестовой среды.
- Фаза тестирования. Срок – 2 недели.
 - Выполнение тест-кейсов.
 - Документирование результатов.
- Фаза исправления ошибок. Срок – 1 неделя.
 - Исправление выявленных ошибок.
- Фаза повторного тестирования. Срок – 1 неделя.
 - Повторное выполнение тест-кейсов после исправления ошибок.

18.Риски и непредвиденные обстоятельства

- Нестабильная связь между клиентами и сервером.
- Нехватка ресурсов или времени для полного тестирования.

- Внезапные изменения в требованиях или правилах приложения.

19. Утверждение плана тестирования

- План тестирования должен быть утвержден менеджером проекта и ключевыми заинтересованными сторонами.
- Все участники проекта должны быть проинформированы о плане и своих ролях в нем.

20. Глоссарий

- Тест-кейс.

Документ, описывающий конкретный сценарий тестирования.

- Смоук-тестирование.

Набор базовых тестов для проверки основной функциональности.

- Автоматизированное тестирование.

Использование инструментов для автоматизации повторяющихся тестов.

Автоматизация тестирования Web-приложения

Для тестирования были выбраны следующие веб-приложения:

[Hollow Knight](#), [Домашняя страница | Дистанционное обучение РТУ МИРЭА](#) и [Яндекс — быстрый поиск в интернете](#).

В соответствии с заданием было разработано 3 скрипта.

	Command	Target
1	✓ <i>open</i>	https://www.hollowknight.com/
2	✓ <i>set window size</i>	974x1032
3	✓ <i>select frame</i>	index=0
4	✓ <i>click</i>	css=.ytp-large-play-button
5	✓ <i>mouse down</i>	css=.ytp-timed-markers-container
6	✓ <i>mouse up</i>	css=.ytp-scrubber-pull-indicator
7	✓ <i>click</i>	css=.ytp-progress-bar
8	✓ <i>click</i>	css=.ytp-progress-bar
9	✓ <i>click</i>	css=.ytp-play-button

Рисунок 1 – 1-й разработанный скрипт

Running 'test1'

1. open on <https://www.hollowknight.com/> OK
2. setWindowSize on 974x1032 OK
3. selectFrame on index=0 OK
4. click on css=.ytp-large-play-button OK
5. mouseDown on css=.ytp-timed-markers-container OK
6. mouseUp on css=.ytp-scrubber-pull-indicator OK
7. click on css=.ytp-progress-bar OK
8. click on css=.ytp-progress-bar OK
9. click on css=.ytp-play-button OK

'test1' completed successfully

Рисунок 2 – Результат тестирования

	Command	Target
1	✓ open	https://online-edu.mirea.ru/my/
2	✓ set window size	974x1032
3	✓ click	css=.list-group-item:nth-child(1) .aalink
4	✓ pause	2000
5	✓ click	linkText=Домашняя страница
6	✓ click	css=.list-group-item:nth-child(2) .aalink
7	✓ pause	2000
8	✓ click	linkText=Домашняя страница
9	✓ click	id=user-menu-toggle
10	✓ click	linkText=О пользователе

Рисунок 3 – 2-й разработанный скрипт

Running 'test'

1. open on <https://online-edu.mirea.ru/my/> OK
2. setWindowSize on 974x1032 OK
3. click on css=.list-group-item:nth-child(1) .aalink OK
4. pause on 2000 OK
5. click on linkText=Домашняя страница OK
6. click on css=.list-group-item:nth-child(2) .aalink OK
7. pause on 2000 OK
8. click on linkText=Домашняя страница OK
9. click on id=user-menu-toggle OK
10. click on linkText=О пользователе OK

'test' completed successfully

Рисунок 4 – Результат тестирования

	Command	Target	Value
1	✓ open	https://yandex.ru/search/?text=hollow+knight&clid=2411726&lr=213	
2	✓ set window size	974x1032	
3	✓ click	css=.HeaderForm-Clear > svg	
4	✓ mouse over	css=.mini-suggest__button-text	
5	✓ click	css=.mini-suggest__button-text	
6	✓ type	name=text	hello
7	✓ mouse out	css=.mini-suggest__button-text	
8	✓ click	css=.HeaderForm-Clear path	
9	✓ type	name=text	buy
10	✓ send keys	name=text	\$(KEY_ENTER)

Рисунок 5 – 3-й разработанный скрипт

```
Running 'test'
1. open on https://yandex.ru/search/?text=hollow+knight&clid=2411726&lr=213 OK
2. setWindowSize on 974x1032 OK
3. click on css=.HeaderForm-Clear > svg OK
4. mouseOver on css=.mini-suggest__button-text OK
5. click on css=.mini-suggest__button-text OK
6. type on name=text with value hello OK
7. mouseOut on css=.mini-suggest__button-text OK
8. click on css=.HeaderForm-Clear path OK
9. type on name=text with value buy OK
10. sendKeys on name=text with value ${KEY_ENTER} OK
'test' completed successfully
```

Рисунок 6 – Результат тестирования

Участник: Смирнов Даниил Анатольевич

Разработка плана тестирования

В практической работе №3 была разработана игра «Города». Ниже представлен план тестирования данного приложения

1. Идентификатор тестового плана

- Тестовый план для игры "Города"
- Идентификатор: TP-CG-0.1 (Test plan - City Game v0.1).

2. Ссылки на используемые документы

- Документация игры "Города"
- Полное руководство по тестированию игр

3. Введение

- Цель.

Определить и документировать процесс тестирования игры «Города» для обеспечения ее качества и стабильности.

- Область применения.
 - Тестирование функциональности.
 - Тестирование пользовательского интерфейса.
 - Тестирование многопользовательских функций .

4. Тестируемые элементы

- **Пользовательский интерфейс и навигация**
 - Ввод и вывод данных (названия городов)
 - Управление игровым процессом (переход между игроками)
- **Логика игры**
 - Правила игры (последняя буква предыдущего города должна совпадать с первой буквой следующего)
 - Проверка на повторные названия городов
- **Интеграция с дополнительными функциями**
 - Возможность сдачи игры
 - Обработка ошибок и исключений

5. Проблемы риска тестирования ПП

- Ошибки в логике игры, неправильное определение победителя или продолжения игры
- Утечки памяти или другие проблемы с производительностью

6. Особенности или свойства, подлежащие тестированию

- Функциональное тестирование.
 - Проверка соответствия игрового процесса заданным правилам.
 - Тестирование установки и запуска игры без проблем.
- Комбинаторное тестирование.
 - Проверка всех возможных вариантов значений параметров (например, различные названия городов и их последовательности).
- Play-тестирование.
 - Оценка игрового опыта и уровня развлекательности,.
- Тестирование выносливости.
 - Проверка игры на длительное время для выявления проблем с утечкой памяти и стабильностью.

7. Особенности (свойства), не подлежащие тестированию

- Внешний дизайн и графика, если они не влияют на функциональность игры.

8. Подход

- Разработка тест-кейсов.

Создание позитивных и негативных тест-кейсов для покрытия всех аспектов игры.

- Альфа-тестирование и бета-тестирование.

Проведение тестирования на разных этапах разработки для выявления и исправления ошибок.

- Автоматизированное тестирование.

Использование инструментов для автоматизации повторяющихся тестов, особенно для тестирования выносливости.

9. Критерии смоук-тестирования

- Проверка основной функциональности игры (ввод и вывод данных, переход между игроками).
- Успешный запуск и завершение игрового процесса.
- Отсутствие критических ошибок, препятствующих продолжению игры.

10. Критерии прохождения тестов

- Все тест-кейсы пройдены успешно без критических ошибок.
- Игровой процесс соответствует заданным правилам и требованиям.
- Пользовательский интерфейс интуитивно понятен и удобен для использования.

11. Критерии приостановки и возобновления работ

- Обнаружение критических ошибок, которые невозможно исправить в текущем цикле тестирования.
- Недостаточная подготовка тестовой среды или необходимость дополнительных ресурсов.
- Возобновление работ после исправления выявленных ошибок и обеспечения необходимых ресурсов.

12. Тестовая документация

- Тест-кейсы и сценарии тестирования.
- Отчеты о результатах тестирования, включая видеоролики и скриншоты.
- Журнал дефектов с описанием ошибок и статусом их исправления.

13. Основные задачи тестирования

- Обеспечение соответствия игрового процесса заданным правилам.
- Проверка стабильности и производительности игры.
- Оценка пользовательского опыта и игрового дизайна.

14. Необходимый персонал и обучение

- Тестировщики с опытом функционального и play-тестирования.
- Обучение по использованию автоматизированных инструментов

тестирования.

- Знакомство с правилами игры и ее механиками.

15.Требования среды

- Тестовая среда, максимально приближенная к реальной.
- Необходимые серверные ресурсы для многопользовательского режима.
- Инструменты для автоматизированного тестирования и мониторинга производительности.

16.Распределение ответственности

- Тестировщики.

Выполнение тест-кейсов, документирование результатов.

- Разработчики.

Написание кода, исправление выявленных ошибок, поддержка тестировщиков.

- Менеджер проекта.

Координирование процессов, контроль сроков и ресурсов.

17.График работ (календарный план)

- Фаза подготовки. Срок – 1 неделя.
 - Разработка тест-кейсов
 - Настройка тестовой среды.
- Фаза тестирования. Срок – 2 недели.
 - Выполнение тест-кейсов.
 - Документирование результатов.
- Фаза исправления ошибок. Срок – 1 неделя.
 - Исправление выявленных ошибок.
- Фаза повторного тестирования. Срок – 1 неделя.
 - Повторное выполнение тест-кейсов после исправления ошибок.

18.Риски и непредвиденные обстоятельства

- Нестабильная связь между клиентами и сервером.
- Нехватка ресурсов или времени для полного тестирования.

- Внезапные изменения в требованиях или правилах игры.

19. Утверждение плана тестирования

- План тестирования должен быть утвержден менеджером проекта и ключевыми заинтересованными сторонами.
- Все участники проекта должны быть проинформированы о плане и своих ролях в нем.

20. Глоссарий

- Тест-кейс.

Документ, описывающий конкретный сценарий тестирования.

- Смоук-тестирование.

Набор базовых тестов для проверки основной функциональности.

- Play-тестирование.

Тестирование игры с точки зрения пользовательского опыта и развлекательности.

- Автоматизированное тестирование.

Использование инструментов для автоматизации повторяющихся тестов.

Автоматизация тестирования Web-приложения

Для тестирования было выбрано следующее веб-приложение: [ссылка](#).

В соответствии с заданием было разработано 3 скрипта.

	Command	Target	Value
1	<i>open</i>	https://ambientcg.com/	
2	<i>set window size</i>	921x982	
3	<i>click</i>	linkText=Explore all 2468 assets	
4	<i>pause</i>	2000	
5	<i>type</i>	id=searchInput	rock
6	<i>send keys</i>	id=searchInput	\${KEY_ENTER}
7	<i>click</i>	css=.AssetBox:nth-child(2) .OnlyShowDark	

Рисунок 7 – 1-й разработанный скрипт

Running 'Test 1'	13:13:22
1. open on https://ambientcg.com/ OK	13:13:22
2. setWindowSize on 921x982 OK	13:13:22
3. click on linkText=Explore all 2468 assets OK	13:13:22
4. pause on 2000 OK	13:13:25
5. type on id=searchInput with value rock OK	13:13:27
6. sendKeys on id=searchInput with value \${KEY_ENTER} OK	13:13:27
7. click on css=.AssetBox:nth-child(2) .OnlyShowDark OK	13:13:27
'Test 1' completed successfully	13:13:27

Рисунок 8 – Результат тестирования

	Command	Target	Value
1	<i>open</i>	https://ambientcg.com/	
2	<i>set window size</i>	924x982	
3	<i>click</i>	linkText=Gallery	
4	<i>pause</i>	2000	
5	<i>click</i>	css=.GalleryEntry:nth-child(9) > a > img	
6	<i>pause</i>	2000	
7	<i>click</i>	css=.btn-primary:nth-child(1)	

Рисунок 9 – 2-й разработанный скрипт

Running 'Test 2'	13:14:41
1. open on https://ambientcg.com/ OK	13:14:41
2. setWindowSize on 924x982 OK	13:14:41
3. click on linkText=Gallery OK	13:14:41
4. pause on 2000 OK	13:14:43
5. click on css=.GalleryEntry:nth-child(9) > a > img OK	13:14:45
6. pause on 2000 OK	13:14:45
7. click on css=.btn-primary:nth-child(1) OK	13:14:47
'Test 2' completed successfully	13:14:47

Рисунок 10 – Результат тестирования

	Command	Target	Value
1	<i>open</i>	https://ambientcg.com/	
2	<i>set window size</i>	926x982	
3	<i>click</i>	linkText= ▶ HDRIs	
4	<i>pause</i>	2000	
5	<i>click</i>	css=.AssetBox:nth-child(13) .OnlyShowDark	
6	<i>pause</i>	2000	
7	<i>click</i>	css=.ListThumbnail:nth-child(2)	
8	<i>pause</i>	2000	
9	<i>click</i>	linkText=Docs	
10	<i>select window</i>	handle=\${win5462}	
11	<i>click</i>	linkText=License Information	

Рисунок 11 – 3-й разработанный скрипт

Running 'Test 3'	13:18:36
1. open on https://ambientcg.com/ OK	13:18:37
2. setWindowSize on 926x982 OK	13:18:37
3. click on linkText= ▶ HDRIs OK	13:18:37
4. pause on 2000 OK	13:18:39
5. click on css=.AssetBox:nth-child(13) .OnlyShowDark OK	13:18:41
6. pause on 2000 OK	13:18:42
7. click on css=.ListThumbnail:nth-child(2) OK	13:18:44
8. pause on 2000 OK	13:18:44
9. click on linkText=Docs OK	13:18:46
10. selectWindow on handle=\${win5462} OK	13:18:46
11. click on linkText=License Information OK	13:18:46
'Test 3' completed successfully	13:18:46

Рисунок 12 – Результат тестирования

Участник: Хохлинов Дмитрий Иванович

Разработка плана тестирования

Для выполнения данного шага было взято приложение, разработанное в практической работе №3 (игра «Как стать миллионером»).

Для него подробно опишем каждый пункт плана тестирования:

Идентификатор тестового плана:

TP-MillionairGame-001

Ссылки на используемые документы:

IEEE Std 829 - Стандарт документации по тестированию программного обеспечения;

Спецификация требований к игре – документ SRD-MillionairGame-001;

План управления проектом – документ PMP-MillionairGame-001.

Введение:

Данный тестовый план описывает подход к тестированию программного обеспечения игры «Как стать миллионером». Цель тестирования — обеспечить соответствие функциональным требованиям и выявить возможные дефекты в работе ПО.

Тестируемые элементы:

- Чтение вопросов из файла в игру;
- Ответ на вопрос (правильный/неправильный);
- Выигрыш и проигрыш в игре.

Проблемы риска тестирования ПП:

- Возможно неправильное чтение вопроса из файла с вопросами;
- Неправильная обработка пользовательского ввода;
- Некорректная реакция игры на действия игрока.

Особенности и свойства, подлежащие тестированию:

- Корректность чтения вопросов из файла;
- Устойчивость к некорректному вводу пользователя;

- Корректность механик игры.

Особенности и свойства, не подлежащие тестированию:

- Пользовательский интерфейс (UI);
- Производительность при большом количестве доступных для выбора вопросов.

Подход:

Тестирование будет выполняться автоматически с написанием unit-тестов и behave-тестов для приложения, включающих в себя как позитивные, так и негативные сценарии.

Критерии смоук-тестирования:

Приложение должно корректно считывать вопросы из файла, а также корректно реагировать на действия игрока.

Критерии прохождения тестов:

Тест считается пройденным, если фактический результат работы ПО соответствует ожидаемому результату для всех тестовых случаев.

Критерии приостановки и возобновления работ:

Работы по тестированию будут приостановлены в случае обнаружения критических дефектов, которые блокируют дальнейшее тестирование. Возобновление работ возможно после исправления дефектов и повторного тестирования.

Тестовая документация:

Тестовые случаи – документ TC-MillionairGame-001;

Отчет о тестировании – документ TR-MillionairGame-001.

Основные задачи тестирования:

- Проверка корректности работы игры;
- Проверка обработки ошибок;
- Выявление и документирование дефектов в работе ПО.

Необходимый персонал и обучение:

Для выполнения тестирования потребуется один тестировщик, знакомый с методами функционального тестирования. Также потребуется

ознакомить его с игрой и ее механиками.

Требования среды:

Тестирование будет проводиться на следующих платформах:

- Windows 11 Pro;
- Ubuntu 22.04 LTS.

Распределение ответственности:

Тестировщик: Ответственность за выполнение функциональных тестов чтения вопросов, правильного/неправильного ответа на вопрос и выигрыша/проигрыша в игре.

Разработчик: Написание кода, исправление выявленных ошибок.

Менеджер проекта: Координирование процессов, контроль сроков и ресурсов.

График работ:

Этап	Дата начала	Дата окончания
Подготовка тестов	01.11.24	03.11.24
Выполнение тестов	04.11.24	05.11.24
Анализ результатов	06.11.24	09.11.24
Подготовка отчета	10.11.24	13.11.24

Риски и непредвиденные обстоятельства:

- Задержка в разработке функционала;
- Нехватка времени на выполнение необходимых тестов.

Утверждение плана тестирования:

Данный план тестирования будет утвержден руководителем проекта и командой разработки до начала этапа выполнения тестов.

Глоссарий:

- Смоук-тестирование: Быстрое проверочное тестирование для определения основных функциональных возможностей приложения.
- Тестовые случаи: Набор условий или переменных, при которых проверяется выполнение определенной функции программы.
- Дефект: Ошибка или несоответствие в программном обеспечении, требующее исправления.

Автоматизация тестирования Web-приложения

Для выполнения задания было выбрано следующее web-приложение:
https://dmitryh2004.github.io/RKChIR_2kurs_Kurovaya_rabota.

В соответствии с заданием было разработано 3 скрипта (рисунки 13 - 18):

	Command	Target	Value
1	open	https://dmitryh2004.github.io/RKChIR_kursach/html/main.html	
2	set window size	1296x688	
3	//		
4	click	id=sec1_content	
5	wait for text	id=sec1_content	История самокатов начинается с 1761 года: именно тогда появился его первый прототип. С этого момента самокаты претерпели значительные изменения: раньше они...
6	close		

Рисунок 13 - Скрипт 1: проверка появления текста при нажатии на карточку

```
Running 'test1'
1. open on https://dmitryh2004.github.io/RKChIR_kursach/html/main.html OK
2. setWindowSize on 1296x688 OK
3. //
4. click on id=sec1_content OK
5. waitForText on id=sec1_content with value История самокатов начинается с 1761 года: именно тогда появился его первый прототип. С этого момента самокаты претерпели значительные изменения: раньше они были похожи на беговел. В 1985 году Стив Патмонт сконструировал самокат с мотором, положив начало развитию электросамокатов. Сейчас самокаты подразделяются на несколько типов, среди которых есть и электросамокаты. Разделение обусловлено разными задачами, под которые создаются самокаты. OK
6. close OK
'test1' completed successfully
```

Рисунок 14 - Результат тестирования скрипта 1

1	open	https://dmitryh2004.github.io/RKChIR_kursach/html/about.html
2	set window size	1280x672
3	double click	id=authorImage
4	click	id=authorImage
5	click	id=authorImage
6	close	

Рисунок 15 - Скрипт 2: проверка воспроизведения музыки при двойном нажатии на фото автора сайта

```
Running 'test2'
1. open on https://dmitryh2004.github.io/RKChIR_kursach/html/about.html OK
2. setWindowSize on 1280x672 OK
3. click on id=authorImage OK
4. click on id=authorImage OK
5. doubleClick on id=authorImage OK
6. close OK
'test2' completed successfully
```

Рисунок 16 - Результат тестирования скрипта 2

	Command	Target
1	<i>open</i>	https://dmitryh2004.github.io/RKChIR_kursach/html/electrosamokats_and_cars.html
2	<i>set window size</i>	1296x688
3	<i>click</i>	css=.right
4	<i>click</i>	css=.right
5	<i>click</i>	css=.right
6	<i>click</i>	css=.right
7	<i>click</i>	css=.block:nth-child(5) img
8	<i>store window handle</i>	root
9	<i>select window</i>	handle=\${win8625}
10	<i>close</i>	
11	<i>select window</i>	handle=\${root}
12	<i>close</i>	

Рисунок 17 - Скрипт 3: проверка функциональности карусели; проверка открытия изображения в новом окне при нажатии на него

Running 'test3'

1. open on https://dmitryh2004.github.io/RKChIR_kursach/html/electrosamokats_and_cars.html OK
2. setWindowSize on 1296x688 OK
3. click on css=.right OK
4. click on css=.right OK
5. click on css=.right OK
6. click on css=.right OK
7. click on css=.block:nth-child(5) img OK
8. storeWindowHandle on root OK
9. selectWindow on handle=\${win8625} OK
10. close OK
11. selectWindow on handle=\${root} OK
12. close OK

'test3' completed successfully

Рисунок 18 - Результат тестирования скрипта 3

Участник: Чикунов Данила Сергеевич

Разработка плана тестирования

В практической работе №3 была разработано приложение «Электронный терапевт». Ниже представлен план тестирования данного приложения

1. Идентификатор тестового плана

ТР-001

2. Ссылки на используемые документы

- Спецификация требований к программному обеспечению (SRS)
- Документация по классу Medic
- Рекомендации по тестированию программного обеспечения

3. Введение

Данный тестовый план описывает процесс тестирования класса Medic, который предназначен для диагностики заболеваний на основе введенных симптомов. Класс включает методы для добавления симптомов к болезни и получения диагноза.

4. Тестируемые элементы

- Метод add_symptom()
- Метод get_diagnosis()

5. Проблемы риска тестирования ПП

- Возможно неправильное чтение симптома;
- Некорректная реакция программы на действия пользователя.

6. Особенности или свойства, подлежащие тестированию

- Корректность добавления симптомов.
- Правильность диагностики на основе предоставленных симптомов.
- Обработка случаев, когда симптомы не соответствуют ни одному заболеванию.

7. Особенности (свойства), не подлежащие тестированию

- Функция инициализации

8. Подход

Тестирование будет проводиться с использованием модульного тестирования, где каждый метод будет проверяться отдельно с различными входными данными.

Также будет проводиться поведенческое тестирование.

9. Критерии смоук-тестирования

- Класс должен успешно создаваться без ошибок.
- Методы `add_symptom()` и `get_diagnosis()` должны вызываться без ошибок.

10. Критерии прохождения тестов

- Все методы должны возвращать ожидаемые результаты для заданных входных данных.
- Не должно возникать исключений или ошибок при нормальных условиях использования.

11. Критерии приостановки и возобновления работ

Работы могут быть приостановлены в случае обнаружения критических ошибок, которые препятствуют проведению дальнейшего тестирования. Работы возобновляются после исправления ошибок и повторного подтверждения функциональности.

12. Тестовая документация

Документация будет включать:

- Описание тестовых случаев.
- Ожидаемые результаты для каждого теста.
- Записи о проведенных тестах и их результатах.

13. Основные задачи тестирования

- Проверка корректности инициализации класса.
- Проверка корректности добавления симптомов.
- Проверка правильности диагностики для различных комбинаций симптомов.
- Проверка обработки ошибок и исключительных ситуаций.

14. Необходимый персонал и обучение

Для выполнения тестирования потребуется:

- Один тестировщик с опытом работы с Python и знанием методов модульного и поведенческого тестирования.

Необходимое обучение может включать:

- Обучение по использованию фреймворков для модульного тестирования pytest и поведенческого - behave.

15. Требования среды

Тестирование будет проводиться в среде:

- Python версии 3.x.
- Установленный интерпретатор Python и необходимые библиотеки для выполнения кода.

16. Распределение ответственности

- Тестировщик: отвечает за выполнение всех этапов тестирования и документирование результатов.

17. График работ (календарный план)

Этап	Дата начала	Дата окончания
Подготовка к тестированию	01/10/2024	02/10/2024
Проведение тестов	03/10/2024	05/10/2024
Анализ результатов	06/10/2024	07/10/2024
Подготовка отчета	08/10/2024	09/10/2024

18. Риски и непредвиденные обстоятельства

Риски могут включать:

- Изменения в требованиях к функциональности класса.
- Нехватка времени на завершение всех этапов тестирования.

19. Утверждение плана тестирования

План должен быть утвержден руководителем проекта перед началом

выполнения тестов.

20. Глоссарий

- Класс: Шаблон для создания объектов, содержащий данные и методы.
- Метод: Функция, связанная с классом.
- Диагностика: Процесс определения заболевания на основе симптомов.

Автоматизация тестирования Web-приложения

Для тестирования было выбрано следующая веб-страница: [Ссылка](#).

В соответствии с заданием было разработано 3 скрипта.

✓ Идем до "Большой волш		Command	Target	Value
✓ Идем до кинжала	1	✓ open	https://darksouls.fandom.com/ru/wiki/%D0%A1%D1%82%D1%80%D0%B5%D0%BB%D0%B0...	
✓ Идем до человечности	2	✓ set window size	1552x840	
	3	✓ run script	window.scrollTo(0,868)	
	4	✓ click	linkText=Рикерта из Винхейма	
	5	✓ run script	window.scrollTo(0,700)	
	6	✓ click	css=tr:nth-child(2) img	
	7	✓ mouse over	css=.fandom-community-header__local-navigation:nth-child(3) .explore-menu > .wds-tabs__tab...	
	8	✓ run script	window.scrollTo(0,500)	
	9	✓ close		

Рисунок 19 – 1-й разработанный скрипт

Search tests...	https://darksouls.tandom.com/ru/wiki/%D0%A1%D1%82%D1%80%D0%B5%D0%			
✓ Идем до "Большой волш		Command	Target	
✓ Идем до кинжала	1	✓ open	https://darksouls.fandom.com/ru/wiki/%D0%A1%D1%82%D1%80%D0%B5%D0%BB%D0%B0...	
✓ Идем до человечности	2	✓ set window size	1552x840	
	3	✓ run script	window.scrollTo(0,628.7999877929688)	
	4	✓ click	linkText=Волшебник	
	5	✓ click	linkText=Кинжал	
	6	✓ close		

Рисунок 20 – 2-й разработанный скрипт

✓ Идем до "Большой волш	Command		Target	1
✓ Идем до кинжала	1	✓ <i>open</i>	https://darksouls.fandom.com/ru/wiki/%D0%A1%D1%82%D1%80%D0%B5%D0%BB%D0%B0...	
✓ Идем до человечности*	2	✓ <i>set window size</i>	1552x840	
	3	✓ <i>click</i>	linkText=Волшебство	
	4	✓ <i>click</i>	linkText=Нагим Ситом	
	5	✓ <i>run script</i>	window.scrollTo(0,196.8000030517578)	
	6	✓ <i>click</i>	linkText=Человечность	
	7	✓ <i>close</i>		

Рисунок 21 – 3-й разработанный скрипт

Вывод

В ходе выполнения данной практической работы были изучены основные принципы разработки плана тестирования программного обеспечения и способы автоматизированного тестирования Web-приложений на примере работы с Selenium IDE. В результате выполнения данной практической работы каждым участником группы был написан план тестирования приложения, разработанного им в 3-й практической работы, и было протестировано Web-приложение.

Список использованных источников

1. Лекция №1 по тестированию и верификации ПО: https://online-edu.mirea.ru/pluginfile.php/983241/mod_resource/content/1/Презентация.Лекция%20№1.pdf (дата обращения: 10.09.24)
2. Лекция №2 по тестированию и верификации ПО: <https://online-edu.mirea.ru/mod/resource/view.php?id=408633> (дата обращения: 20.09.24)
3. Задание на практическую работу: <https://online-edu.mirea.ru/mod/resource/view.php?id=403073> (дата обращения: 20.09.24)