|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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**

**План тестирования**

по дисциплине

«Тестирование и верификация ПО»

|  |  |
| --- | --- |
| **Выполнил:**  Студент группыИКБО-15-22 | Оганнинян Г.А. |
| **Проверил:**  Доцент | Чернов Е.А. |

2024 г.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc181641605)

[1 ЧАСТЬ 1 — РАЗРАБОТКА ПЛАНА ТЕСТИРОВАНИЯ 4](#_Toc181641606)

[1.1 Подробное описание плана тестирования 4](#_Toc181641607)

[2 ЧАСТЬ 2 — АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ WEB-ПРИЛОЖЕНИЯ 7](#_Toc181641608)

[2.1 Описание сценариев и запуск тестов 7](#_Toc181641609)

[2.2 Сценарий 1 — Авторизация в аккаунт Spotify 8](#_Toc181641610)

[2.3 Сценарий 2 — Поисковый запрос и подписка 9](#_Toc181641611)

[2.4 Сценарий 3 — Добавление песни в плейлист 10](#_Toc181641612)

[2.5 Сценарий 4 — Удаление песни из плейлиста **Ошибка! Закладка не определена.**](#_Toc181641613)

[ЗАКЛЮЧЕНИЕ 12](#_Toc181641614)

ВВЕДЕНИЕ

**Цель работы:** освоить основные принципы разработки плана тестирования программного обеспечения и применить их на практике, а также научиться проводить автоматизированное тестирование Web-приложений.

**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

На данный момент актуальным стандартом IEEE для документации по тестированию программного обеспечения и систем является IEEE Std 829-2019. Он определяет требования к содержанию, стилю и оформлению тестовой документации. Он содержит рекомендации по написанию тестовых планов, сценариев, процедур, отчётов и других документов, связанных с тестированием. Этот стандарт может быть полезен для разработчиков, тестировщиков и всех, кто участвует в процессе разработки программного обеспечения.

**Оборудование и ПО:** Компьютер с установленной средой разработки (IDE) и выбранным языком программирования.Библиотеки для модульного тестирования (при необходимости). Любые другие инструменты для тестирования в зависимости от языка программирования.

**1 ЧАСТЬ 1 — РАЗРАБОТКА ПЛАНА ТЕСТИРОВАНИЯ**

* 1. **Подробное описание плана тестирования**

Для выполнения работы необходимо взять приложение, разработанное в практической работе №3, и подробно описать для него каждый пункт плана тестирования. План тестирования для программы «Игра угадай мелодию»:

1. **Идентификатор тестового плана**

* **Название:** Тестирование функционала угадывания мелодий
* **Идентификатор:** ТестПлан\_УгадываниеМелодий\_v1.0

**2. Ссылки на используемые документы**

* Код приложения (практическая работа №3)
* Спецификация требований (если есть)
* Сценарии BDD (файл guess\_melody.feature).

**3. Введение**

Цель тестирования — проверить корректность реализации функционала игры на угадывание мелодий. Программа позволяет нескольким игрокам угадывать мелодии по очереди, накапливая очки за правильные ответы. Тесты включают проверку начальных и граничных условий, обработку правильных и неправильных ответов, а также корректный переход хода и очков.

**4. Тестируемые элементы**

* **Основная логика приложения (файл game/game.go)**
* **Ввод данных от пользователя в main.go**
* **Модульное тестирование (файл game\_test.go)**
* **Поведенческое тестирование (файл game\_bdd\_test.go)**.

**5. Проблемы риска тестирования ПП**

* Возможные проблемы с регистрацией игроков и мелодий
* Риск несоответствия между пользовательским вводом и обработкой программы
* Потенциальные проблемы с некорректным подсчетом очков.

**6. Особенности или свойства, подлежащие тестированию**

* Начисление очков за правильные ответы
* Обнуление начисления очков за неправильные ответы
* Переход к следующему игроку и мелодии после каждого ответа
* Корректное завершение игры

**7. Особенности (свойства), не подлежащие тестированию**

* Пользовательский интерфейс (CLI)
* Производительность при высоких нагрузках (вне рамок теста)

**8. Подход**

Используем модульное тестирование (файл game\_test.go) и BDD тестирование с использованием Ginkgo и Gomega (файл game\_bdd\_test.go и сценарии guess\_melody.feature). Эти подходы позволят проверить корректность функционала с разных сторон.

**9. Критерии смоук-тестирования**

* Запуск игры с 2 игроками и 3 мелодиями
* Проверка возможности угадывания для каждого игрока
* Переход хода к следующему игроку после каждой попытки

**10. Критерии прохождения тестов**

* Все сценарии в guess\_melody.feature должны завершаться успешно
* Модульные тесты не должны содержать ошибок и предупреждений.

**11. Критерии приостановки и возобновления работ**

* Приостановка: Обнаружение критической ошибки, нарушающей основной игровой процесс
* Возобновление: Исправление критических ошибок и успешное прохождение смоук-тестов

**12. Тестовая документация**

* Сценарии BDD (файл guess\_melody.feature)
* Лог тестирования, выводимый при запуске game\_test.go

**13. Основные задачи тестирования**

* Проверить правильность начисления очков
* Проверить корректность переключения хода и мелодий
* Проверить обработку граничных и негативных условий

**14. Необходимый персонал и обучение**

* QA инженер с базовыми знаниями Go и инструментов для тестирования Ginkgo и Gomega

**15. Требования среды**

* Golang 1.22 или выше
* Ginkgo и Gomega для BDD тестирования

**16. Распределение ответственности**

* Разработчик: написание кода и тестов
* QA инженер: разработка плана тестирования, проведение и анализ тестов

**17. График работ (календарный план)**

1. **Составление тест-плана — 1 день**
2. **Написание тест-кейсов — 1 день**
3. **Проведение тестов — 1 день**
4. **Анализ результатов и отчет — 1 день**

**18. Риски и непредвиденные обстоятельства**

* **Неисправность оборудования или зависимость от устаревших версий Ginkgo/Gomega**
* **Изменения в спецификации или функционале игры в процессе тестирования**

**19. Утверждение плана тестирования**

План тестирования утверждается руководителем проекта или ответственным лицом по QA.

**20. Глоссарий**

* BDD — поведенческое тестирование
* Ginkgo и Gomega — фреймворки для тестирования на Go
* Модульное тестирование — проверка отдельных функций

**2 ЧАСТЬ 2 — АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ WEB-ПРИЛОЖЕНИЯ**

**2.1 Описание сценариев и запуск тестов**

Для начала разработаем сценарии автоматизированного тестирования с помощью Selenium IDE. Это удобный инструмент для записи и выполнения тестов в браузере, который позволяет записывать действия пользователя и воспроизводить их.

Для проведения тестирования будет использоваться Web-приложение музыкального стриминг-сервиса ЯндексМузыка.

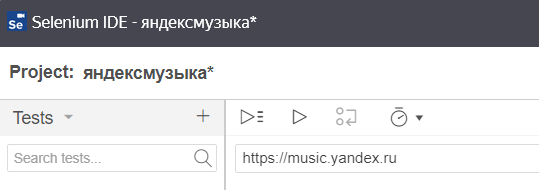


Рисунок 1 — Запуск Web-приложения ЯндексМузыка

Для демонстрации выберем следующие функциональные элементы приложения:

* Авторизация в аккаунт
* Поиск исполнителя
* Подписка на исполнителя
* Переход на новый дизайн

Напишем несколько сценариев для скриптов.

**2.2 Сценарий 1 — Авторизация в аккаунт ЯндексМузыка**

**Цель:** Проверить, что форма авторизации успешно принимает корректные данные и перенаправляет пользователя на главную страницу.

**Шаги теста:**

1. Открыть страницу авторизации
2. Ввести корректные данные в поля «Имя пользователя» и «Пароль»
3. Нажать на кнопку «Войти»

**Ожидаемый результат:** Переход на главную страницу приложения

**Запуск теста:**

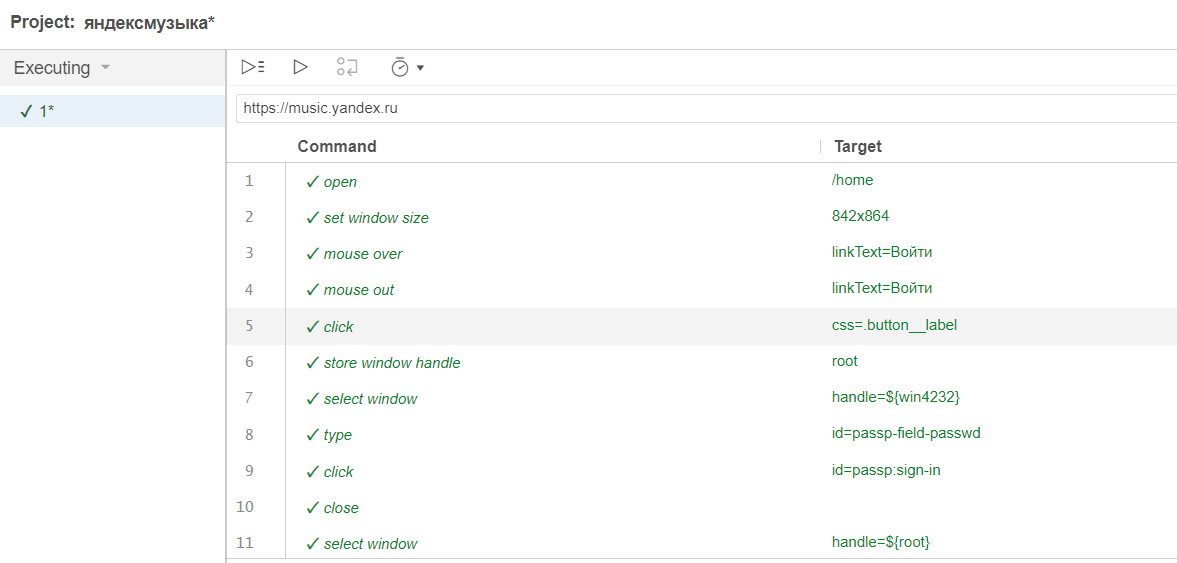
****

Рисунок 2 — Тестирование формы авторизации

**Результат:** Completed successfully

**2.3 Сценарий 2 — Поисковый запрос и подписка**

**Цель:** Проверить, что поиск исполнителя по имени работает корректно и отображает результаты, соответствующие запросу. Функция подписки на исполнителя работает корректно.

**Шаги теста:**

1. Открыть главную страницу ЯндексМузыка
2. В поле поиска ввести имя исполнителя (например, «Frank Ocean»).
3. Нажать Enter или значок поиска
4. Открыть страницу исполнителя
5. Подписаться на исполнителя

**Ожидаемый результат:** Отображение списка результатов, включающих нужного исполнителя, переход на страницу и подписка.

**Запуск теста:**

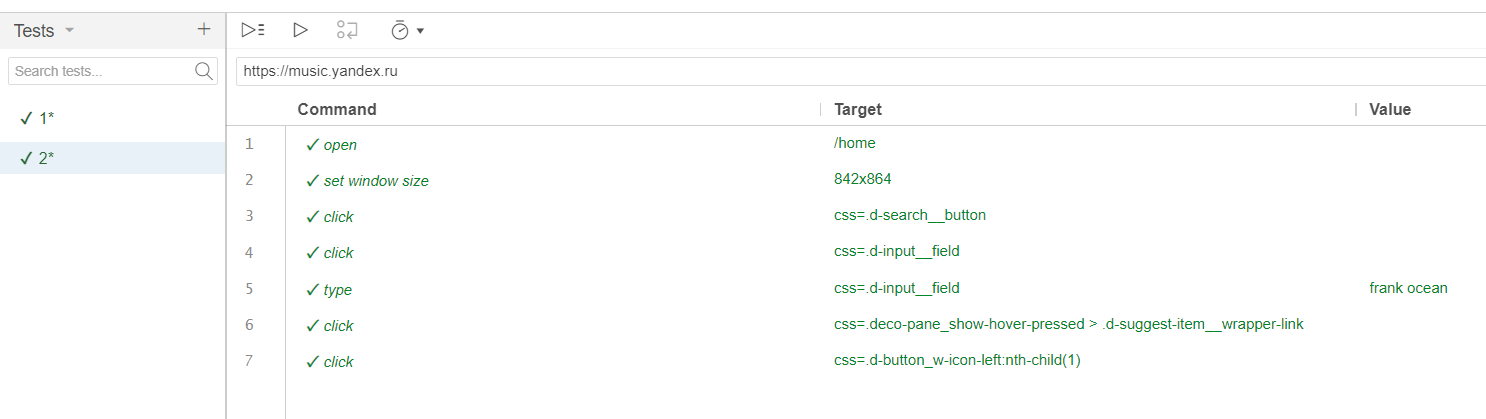
****

Рисунок 3 — Тестирование поиска и функции подписки

**Результат:** Completed successfully

****

Рисунок 4 — Переход на страницу исполнителя и подписка

**2.4 Сценарий 3 — Переход на новый дизайн**

**Цель:** Проверить, переход сайта на новый дизайн.

**Шаги теста:**

1. Найти кнопку для перехода.
2. Нажать на неё.
3. Ожидать перехода на новый дизайн

**Ожидаемый результат:** Страница сменит свой дизайн.

**Запуск теста:**

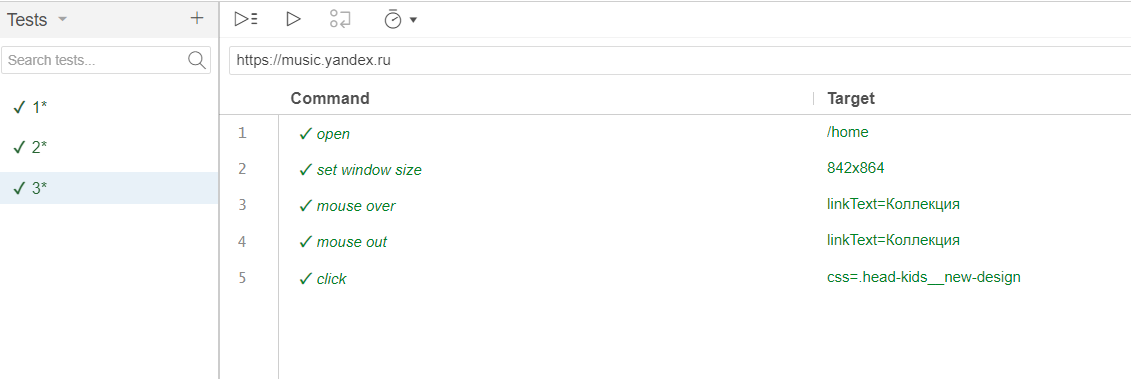
****

Рисунок 5 — Тестирование перехода на новый дизайн

**Результат:** Успешное прохождение теста

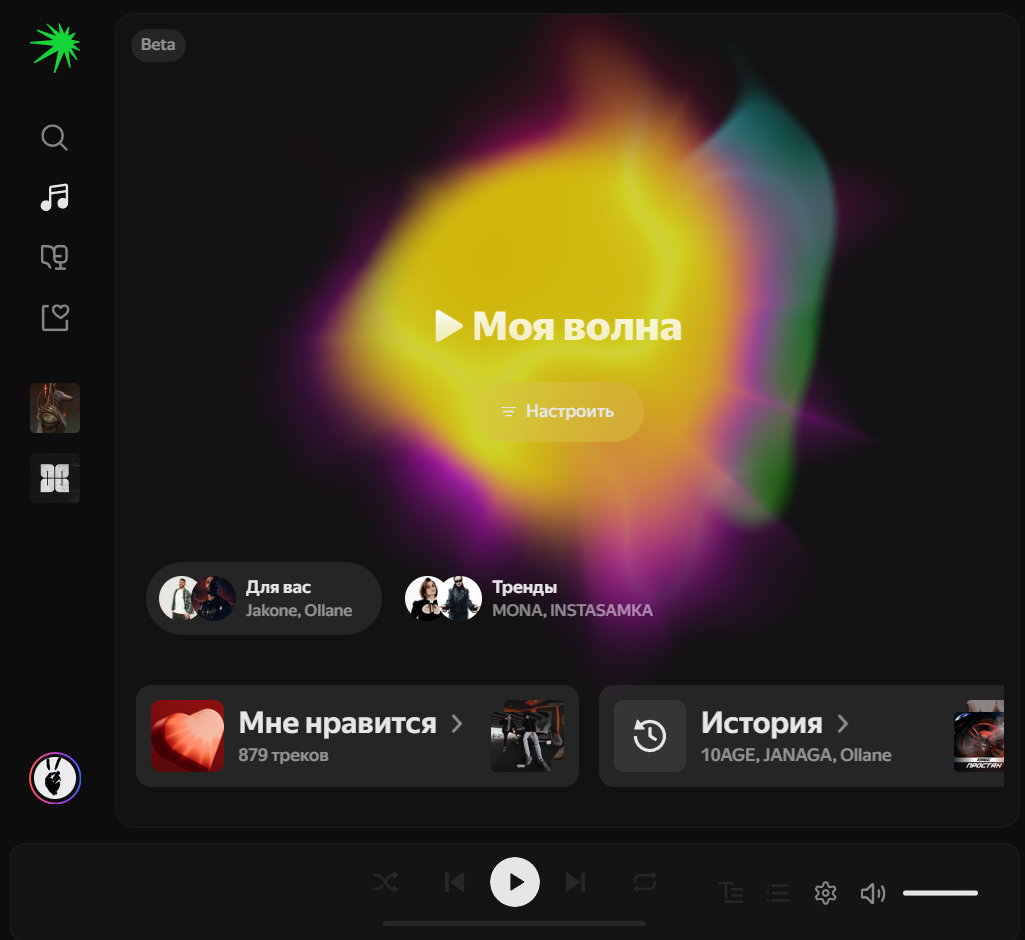


Рисунок 6 — Успешное добавление песни в плейлист

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения работы были изучены и применены на практике основные принципы разработки плана тестирования и проведения автоматизированного тестирования для Web-приложений.

На первом этапе проведен анализ требований к приложению и разработан детализированный план тестирования.

На втором этапе с помощью расширения Selenium IDE в качестве инструмента для создания и выполнения тестовых сценариев были разработаны скрипты для автоматизации тестирования базовых функциональных возможностей выбранного Web-приложения (ЯндексМузыка), таких как авторизация, поиск исполнителя, добавление песни в плейлист и её удаление.