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-33-22 Шило Ю.С.

Проверил:
Ассистент Петрова А.А.

2024 г.

Введение

Тестирование программного обеспечения представляет собой важный этап разработки, обеспечивающий соответствие продукта заявленным требованиям и его работоспособность. В условиях динамичного развития технологий и быстро меняющихся требований к программным решениям создание эффективного плана тестирования становится необходимым. Основной целью данной работы является освоение принципов разработки тестового плана для программного обеспечения и применение этих принципов на практике, включая обучение методам автоматизированного тестирования приложений.

Для достижения цели необходимо решить ряд задач. Во-первых, требуется проанализировать требования к приложению, чтобы определить функции и характеристики, подлежащие тестированию. На основе анализа будет разработан тестовый план, включающий подходы, методы и необходимые ресурсы для тестирования.

Следующий этап предполагает подготовку тестовой среды, включая настройку оборудования и программного обеспечения. Затем будет произведена автоматизация тестирования, что позволит повысить его эффективность и точность. По завершении тестирования важно документировать результаты для обеспечения прозрачности и возможности дальнейшего анализа.

Завершающим шагом станет анализ полученных данных и формулирование рекомендаций по улучшению качества программного обеспечения. В этом процессе полезным стандартом является IEEE Std 829-2019, который устанавливает требования к содержанию, стилю и оформлению тестовой документации. Этот стандарт помогает разработчикам и тестировщикам следовать структурированному подходу в создании тестовых планов, сценариев и отчетов.

Таким образом, работа направлена на формирование целостного понимания процессов тестирования программного обеспечения, с акцентом на автоматизацию и использование современных стандартов.

План тестирования

1. Идентификатор тестового плана

Идентификатор: TP-001

Версия: 1.0

Дата: 15 ноября 2024 года

2. Ссылки на используемые документы

1. Отчет по практической работе №3 / Шило Ю. С. — М.: РТУ МИРЭА, 2024. — 7 с.

Содержит описание структуры и кода приложения "Кто хочет стать миллионером", включая TDD и BDD тесты. Используется для анализа архитектуры приложения и выделения функций для тестирования.

2. Требования к функционалу приложения "Кто хочет стать миллионером " // Документ внутреннего использования. — М.: РТУ МИРЭА, 2024. — 10 с.

Основной документ, определяющий функциональные требования к приложению. Применяется для разработки тест-кейсов, задает обязательные функции, такие как проверка ответов и подсчет очков.

3. IEEE Standard for Software and System Test Documentation (IEEE Std 829-2019). — IEEE, 2019. — 64 с.

Международный стандарт, включающий требования к структуре и оформлению тестовой документации. Используется для создания тест-плана и других отчетных документов.

3. Введение

Этот тестовый план предназначен для приложения «Кто хочет стать миллионером», разработанного в рамках практической работы № 3. Это приложение позволяет игроку отвечать на заранее прописанные вопросы. В случае успеха игроку задается другой вопрос, в случае провала игра заканчивается. Тестирование приложения направлено на выявление дефектов, проверку функциональности и обеспечение соответствия требованиям.

Цели тестирования

Основные цели тестирования включают:

- Проверка корректности работы функции, GiveAnswer, которая определяет, ответил ли игрок правильно на заданный вопрос.
- Оценка производительности функции, ParserLoads которая должна обрабатывать JSON файл и выводить вопросы в консоль.

Аудитория: тестовый план будет использоваться командой тестировщиков и разработчиков для координации действий и обеспечения качества приложения.

Объем тестирования: тестирование охватывает следующие ключевые аспекты приложения:

- Функциональные модули: проверка работы метода Parser.
- Логика игрового процесса: корректное выполнение игры в методе RootClass.

Этот документ служит основой для систематического подхода к тестированию «Кто хочет стать миллионером», обеспечивая чёткость и контроль на всех этапах процесса.

4. Тестируемые элементы

Тестируемые элементы приложения «Кто хочет стать миллионером» включают в себя следующие компоненты:

1. Функциональные модули

Parser: Открывает JSON файл и парсит его. Выписывает в консоль вопрос, найденный в файле:

- Проверка существования файла.
- Возврат массива вопросов, найденных в JSON файле.

2. Логика игрового процесса

RootClass: Основной метод для запуска игры.

- Корректное исполнения метода класса Parser.
- Обработку ввода игрока.
- Правильное завершение игры при вводе неправильного ответа на вопросы.

3. Пользовательский интерфейс

Интерфейс ввода данных: Проверка взаимодействия пользователя с приложением, включая:

- Корректность отображения запросов к пользователю.
- Обработка ввода через консоль и корректное отображение результатов.

5. Интеграция

Взаимодействие с внешними системами: если приложение будет интегрировано с внешними API для получения вопросов и ответов.

Эти тестируемые элементы помогут обеспечить высокое качество приложения и соответствие его функциональности заявленным требованиям.

5. Проблемы риска тестирования ПП

При тестировании приложения «Кто хочет стать миллионером» могут возникнуть следующие риски:

1. Недостаточное покрытие тестами

Описание: Возможность упущения критических дефектов.

Методы управления: Полное тестовое покрытие всех функций.

2. Неполное понимание требований

Описание: Ошибки из-за недостаточного понимания логики игры.

Методы управления: Регулярные встречи с разработчиками.

3. Непредсказуемость поведения

Описание: Ошибки могут проявляться только в определенных сценариях.

Методы управления: Тестирование на основе рисков.

4. Ограниченные ресурсы

Описание: Недостаток времени и бюджета для тестирования.

Методы управления: Расстановка приоритетов по важности задач.

5. Отсутствие резервного плана

Описание: Необходимость плана на случай непредвиденных обстоятельств.

Методы управления: Разработка альтернативных подходов к тестированию. Эти риски следует учитывать для повышения качества приложения «Кто хочет стать миллионером».

6. Особенности или свойства, подлежащие тестированию

При тестировании приложения «Кто хочет стать миллионером» следует обратить внимание на следующие ключевые особенности:

1. Корректность функциональности

- **Методы игры:** Проверка работы классов Parser RootClass на соответствие игровым правилам.

2. Обработка ошибок

- **Сообщения об ошибках:** убедитесь, что отображаемые сообщения верны при вводе недопустимых данных.

3. Пользовательский интерфейс

- **Ввод данных:** Проверка удобства ввода названий городов через консоль и понятности инструкций для пользователей.

4. Безопасность

- **Защита данных:** проверка обработки пользовательских данных и предотвращение уязвимостей.

Эти функции должны быть протестированы для обеспечения качества и надёжности приложения «Кто хочет стать миллионером».

7. Особенности (свойства), не подлежащие тестированию

При тестировании приложения «Кто хочет стать миллионером» следующие функции и свойства не будут подвергаться тестированию:

1. Внешние системы

- **Интеграция с внешними API:** Проверка взаимодействия с внешними сервисами, которые не являются частью приложения, проводиться не будет.

2. Неподдерживаемые функции

- **Устаревшие или удаленные функции:** Любые функции, которые были исключены из спецификации или устарели, не будут протестированы.

3. Игровой баланс

- **Балансировка игрового процесса:** Оценка «фановости» и баланса игры (например, сложность уровней) не входит в область тестирования, так как это больше относится к игровому дизайну.

4. Пользовательский опыт

- **Эмоциональная реакция игроков:** Тестирование восприятия игроками игрового процесса и их эмоциональной реакции проводиться не будет, так как это требует плейтестов с участием реальных пользователей.

5. Эстетические аспекты

- **Дизайн интерфейса:** Оценка визуального оформления и графики приложения не является приоритетом при тестировании.

Эти аспекты не подлежат тестированию в рамках данного плана, что позволяет сосредоточиться на критически важных функциональных элементах приложения «Кто хочет стать миллионером».

8. Подход

1. Модульное тестирование

- Тестирование отдельных классов приложения, таких как Parser, RootClass.

2. Интеграционное тестирование

- Проверка взаимодействия между модулями после завершения модульного тестирования.

3. Функциональное тестирование

- Тестирование всех функциональных возможностей приложения в соответствии с требованиями.

4. Тестирование пользовательского интерфейса

- Оценка удобства ввода данных и отображения информации для пользователей через консоль.

5. Нагрузочное тестирование

- Оценка производительности приложения при увеличении числа игроков и вводимых данных.

6. Тестирование безопасности

- Проверка защиты данных пользователей и устойчивости к возможным атакам.

9. Критерии смоук-тестирования

1. Запуск приложения

- Приложение успешно запускается без ошибок.
- Отсутствуют критические ошибки на этапе запуска.

2. Основные функции

- Проверка корректности работы основных функций, таких как:
- Вход в систему (если применимо).
- Добавление нового города.
- Начало игры с игроками.

3. Навигация

- Убедиться, что навигация между основными разделами приложения работает без проблем.

4. Ввод данных

- Проверка корректности ввода данных (например, названий городов).
- Поля ввода ограничивают ввод некорректных данных.

5. Отображение данных

- Основной пользовательский интерфейс отображается корректно и без существенных дефектов.

6. Обработка ошибок

- Проверка наличия и правильности сообщений об ошибках при вводе недопустимых данных (например, несуществующего города).

7. Производительность

- Оценка общей производительности приложения на базовом уровне, включая время отклика.

10. Критерии прохождения тестов

1. Корректность выполнения функций

- Все класс, такие как RootClass и Parser, должны возвращать ожидаемые результаты при тестировании.

2. Успешное завершение игры

- Игра должна корректно завершаться при вводе недопустимого города или истечении времени, с соответствующими сообщениями об ошибках.

3. Обработка ввода

- Приложение должно корректно обрабатывать ввод названий городов, включая существующие и несуществующие, а также реагировать на истечение времени.

4. Отображение сообщений

- Все сообщения об ошибках и уведомления должны отображаться правильно и соответствовать заданным условиям.

5. Производительность

- Время отклика приложения должно находиться в допустимых пределах, обеспечивая плавный игровой процесс.

6. Безопасность

- Данные пользователей должны быть защищены, а приложение должно устойчиво реагировать на потенциальные атаки.

11. Критерии приостановки и возобновления работ

Критерии приостановки работ

1. **Обнаружение критических дефектов**: если в процессе тестирования выявлены серьезные ошибки, которые препятствуют дальнейшему тестированию или могут повлиять на функциональность приложения.

2. **Недостаток ресурсов**: при отсутствии необходимых ресурсов (времени, оборудования или персонала) для продолжения тестирования.

3. **Необходимость доработки функционала**: если требуется внести изменения в приложение, которые могут повлиять на текущие тестовые сценарии.

4. **Внешние факторы**: влияние внешних обстоятельств, таких как изменения в требованиях или задержки со стороны команды разработчиков.

Критерии возобновления работ

1. **Исправление критических ошибок**: Тестирование возобновляется после устранения всех выявленных критических ошибок.

2. **Восстановление ресурсов**: возобновление работы возможно после предоставления всех необходимых ресурсов для тестирования.

3. **Завершение доработки**: после внесения изменений в приложение и подтверждения их корректности тестирование может быть возобновлено.

4. **Устранение внешних факторов**: как только внешние обстоятельства, препятствующие тестированию, будут устранены.

12. Тестовая документация

Необходимая тестовая документация должна иметь структуру такого вида:

- **Тестовый план.** Документ, описывающий цели тестирования, ключевые задачи, критерии прохождения тестов и исключенные из тестирования функции. Он служит основным руководством для всех этапов тестирования.
- **Сценарии тестов.** Набор детализированных сценариев, в которых описаны шаги для проверки каждой функции, включая ожидания и возможные ошибки. Они упрощают тестирование отдельных аспектов и обеспечивают структурированную проверку.
- **Результаты тестирования.** Краткие отчеты, фиксирующие итог каждого теста (успех или неудача), с примечаниями об обнаруженных

ошибках. Это позволяет легко отслеживать статус тестирования и текущие результаты.

- **Журнал ошибок.** Документ для записи всех найденных ошибок, включая описание, приоритет и статус исправления. Журнал помогает в управлении и отслеживании выполнения исправлений.

Скриншоты и лог-файлы. Визуальные подтверждения и логи тестирования, включающие скриншоты выполнения тестов и результаты автоматизированных проверок. Эти файлы служат дополнением к текстовым отчетам.

13. Основные задачи тестирования

Основные задачи тестирования приложения «Кто хочет стать миллионером» направлены на проверку его функциональности и стабильности работы. Выполнение этих задач поможет подтвердить, что приложение соответствует всем требованиям и обеспечивает стабильную работу для пользователей. Ниже перечислены ключевые направления, на которых сосредоточены усилия команды:

1. **Проверка логики ввода городов.** Убедитесь, что приложение правильно обрабатывает ввод городов, проверяя соответствие правилам игры. Ошибки в этой логике могут привести к неправильному завершению игры и негативному опыту для пользователя.
2. **Тестирование последовательности ходов.** Обеспечить корректную фиксацию приложением последовательности введенных городов и завершение игры при ошибках. Ошибки в этой области могут исказить оценку успешности игрока.
3. **Проверка отображения информации.** Убедитесь, что каждый введенный ответ и все сообщения об ошибках корректно отображаются на экране. Это гарантирует, что игроки видят полную информацию для принятия решений.
4. **Проверка устойчивости к некорректному вводу.** Убедитесь, что приложение адекватно реагирует на ошибки ввода, такие неверный ответ. Это

предотвратит сбои программы и улучшит взаимодействие пользователя с игрой.

5. **Тестирование завершения игры и отображения результата.** Подтвердить, что приложение корректно завершает игровой процесс, отображая итоговое сообщение и результаты. Это помогает игроку получить полное представление о своем результате и завершить игровой процесс без неудобств.

14. Необходимый персонал и обучение

Для успешной реализации тестирования приложения «Кто хочет стать миллионером» необходим следующий персонал и обучение:

1. Персонал

- **Тестировщики**: специалисты, ответственные за выполнение тестов, выявление дефектов и оценку функциональности приложения. Они должны обладать навыками работы с тестовой документацией и методами тестирования.
- **HR-специалисты**: профессионалы, занимающиеся подбором и оценкой персонала. Их задача — обеспечить наличие квалифицированных тестировщиков и специалистов по обучению.
- **Разработчики**: команда, которая будет исправлять выявленные дефекты и вносить изменения в приложение на основе результатов тестирования.
- **Аналитики**: специалисты, которые помогут интерпретировать результаты тестирования и предложить улучшения.

2. Обучение

- **Курсы по тестированию ПО**: обучение тестировщиков основам тестирования, методам выявления дефектов и использованию инструментов автоматизации.
- **Обучение HR-специалистов**: программы, направленные на развитие навыков оценки кандидатов, интерпретации результатов тестирования и составления отчетов.

- **Методическое сопровождение**: консультации опытных специалистов для решения конкретных задач в процессе тестирования.
- **Онлайн-курсы**: доступ к ресурсам для самообучения, где сотрудники могут изучать новые методики и подходы к тестированию.

Эти меры помогут создать квалифицированную команду, способную эффективно выполнять задачи по тестированию и обеспечивать высокое качество приложения «Кто хочет стать миллионером».

15. Требования среды

Для тестирования приложения «Кто хочет стать миллионером» необходимо подготовить тестовую среду, которая поддерживает выполнение как ручных, так и автоматизированных тестов. Это позволит проверить стабильность и корректность работы приложения в условиях, максимально приближенных к реальным.

Основные требования к среде:

- **Операционная система**. Рекомендуется использовать Linux или Windows с поддержкой командной строки. Эти операционные системы позволяют корректно тестировать консольное приложение.
- **Инструменты разработки**. Среда разработки C#, а также текстовый редактор, такой как Rider, для удобного редактирования и отладки кода.
- **Аппаратное обеспечение**. Минимальная конфигурация включает процессор с 2 ядрами и 4 ГБ ОЗУ. Этого достаточно для стабильного выполнения тестов без задержек.
- **Средства автоматизированного тестирования.** Необходимы библиотеки для тестирования на C# для автоматизации основных проверок и упрощения повторного тестирования.
- **Инструменты для нагрузочного тестирования.** Рекомендуется использовать такие инструменты, как Apache JMeter или Siege, для оценки производительности приложения под нагрузкой.

Эта среда обеспечит все необходимые условия для эффективного тестирования приложения «Кто хочет стать миллионером», позволяя команде проверять функциональность и устойчивость программы в предсказуемых условиях.

16. Распределение ответственности

Для успешного тестирования приложения «Кто хочет стать миллионером» важно чётко распределить роли и обязанности между участниками проекта. Это обеспечит слаженную работу команды и минимизирует риски пропуска важных этапов. Ниже приведено распределение основных обязанностей:

Роль	Основные обязанности
Тестировшик	Выполнение функциональных и смоук-тестов, документирование обнаруженных ошибок и создание отчетов по результатам тестирования.
Инженер по автоматизации	Разработка и запуск автоматизированных тестов по методологии TDD, поддержка и обновление скриптов для функциональных проверок.
Менеджер по качеству	Управление процессом тестирования, координация команды, утверждение плана тестирования и финальных отчётов, анализ и обработка рисков.

Такое распределение ответственности позволяет каждому члену команды сосредоточиться на своих задачах, обеспечивая эффективное выполнение тестов и достижение стабильного качества приложения «Кто хочет стать миллионером».

17. График работ (календарный план)

Для организации процесса тестирования и обеспечения своевременного выполнения всех этапов установлен следующий календарный план. Он позволяет распределить задачи таким образом, чтобы каждый шаг был выполнен полноценно и в нужные сроки.

Этап	Описание	Срок выполнения
Подготовка тестовой среды	Настройка и проверка среды для тестирования.	1 день
Разработка тест-кейсов	Создание и описание всех сценариев тестирования.	2 дня
Проведение смоук-тестирования	Базовая проверка запуска приложения и ключевых функций.	1 день
Функциональное тестирование	Подробная проверка основных функций приложения.	3 дня
Автоматизированное тестирование	Запуск и отладка тестов, созданных по TDD-методологии.	2 дня
Документирование результатов	Составление отчетов по каждому этапу тестирования.	1 день
Анализ и подведение итогов	Обсуждение результатов, внесение правок, финальный отчет.	1 день

Таким образом, соблюдение данного плана обеспечит слаженность работы команды и поможет достичь всех целей тестирования в установленные сроки.

18. Риски и непредвиденные обстоятельства

1. Проблемы с тестовой средой. Возможные сложности с настройкой и поддержкой тестовой среды, которые могут привести к задержкам в тестировании.

2. Совместное использование ресурсов. Конфликты при совместном использовании ресурсов между командами разработчиков и тестировщиков могут вызвать сбои и замедлить процесс.

3. Технические проблемы. Сбои в работе серверов или оборудования могут привести к простоям в процессе тестирования, что негативно скажется на графике работ.

4. Изменения в требованиях. Внезапные изменения в требованиях со стороны заказчика могут потребовать пересмотра плана тестирования и дополнительных затрат времени.

5. Уход ключевых сотрудников. Увольнение или болезнь ключевых членов команды могут замедлить процесс тестирования из-за нехватки необходимых знаний и навыков.

6. Непредвиденные ошибки в коде. Выявление критических ошибок в коде на поздних этапах тестирования может потребовать дополнительных временных затрат на исправление.

Эти риски могут повлиять на процесс тестирования приложения «Городская игра» и его результаты, поэтому важно учитывать их и заранее планировать действия по их минимизации.

19. Утверждение плана тестирования

Утверждение плана тестирования для приложения «Кто хочет стать миллионером» является важным этапом, определяющим стратегию, цели и подходы к тестированию. Введение в план должно содержать общее представление о проекте, его целях и ожидаемых результатах. Цели и задачи тестирования необходимо четко сформулировать, включая проверку функциональности, производительности и безопасности приложения. Описание тестируемого продукта должно включать обзор его функциональности, ключевых особенностей и требований. Важно определить область тестирования, указав конкретные функции и модули приложения, которые будут протестированы. Также следует указать необходимые ресурсы, такие как люди, оборудование и инструменты, а также определить временные рамки для проведения тестирования. Стратегия тестирования должна описывать методы и типы тестирования, которые будут использоваться, например, функциональное или нагрузочное тестирование. Критерии приемки должны четко указывать условия и требования для успешного завершения тестирования. Ожидаемые результаты включают описание того, какие отчеты о дефектах и показатели качества должны быть получены. Необходимо также

определить потенциальные риски, связанные с тестированием, и описать предположения. Наконец, организация и коммуникация в команде должны быть чётко определены, включая роли и обязанности участников тестирования, а также способы коммуникации и отчётности о прогрессе. Утверждение этого плана позволит команде чётко понимать свои задачи и обеспечит эффективное выполнение всех этапов тестирования приложения «Города».

20. Глоссарий

Автоматизированное тестирование — процесс тестирования программного обеспечения с использованием специальных программ для выполнения тестов и проверки результатов.

Альфа-тестирование — тестирование, проводимое разработчиками или потенциальными пользователями на ранней стадии разработки продукта, чтобы выявить ошибки до выпуска.

Верификация — процесс проверки, соответствует ли система или ее компонент спецификациям и требованиям, установленным в начале разработки.

Валидация — процесс оценки конечного продукта для определения его соответствия ожиданиям пользователей и требованиям бизнеса.

Тест-кейс — документ, описывающий набор условий и шагов для проверки определенной функциональности приложения.

Тест-план — документ, который описывает объем работ по тестированию, включая цели, стратегию, ресурсы и временные рамки.

Нагрузочное тестирование — процесс оценки производительности приложения под различными сценариями использования и нагрузками.

Чек-лист — документ, содержащий перечень элементов или функций, которые должны быть протестированы.

Интеграционное тестирование — проверка взаимодействия между различными компонентами системы для выявления проблем в их совместной работе.

Функциональное тестирование — процесс проверки функциональности приложения на соответствие заявленным требованиям.

Разработка скриптов

Скрипт 1 пытается в google.com набрать в поиск ‘Selenium’ и найти на открывшейся странице текст ‘Selenium’. Результаты на рисунке 1.

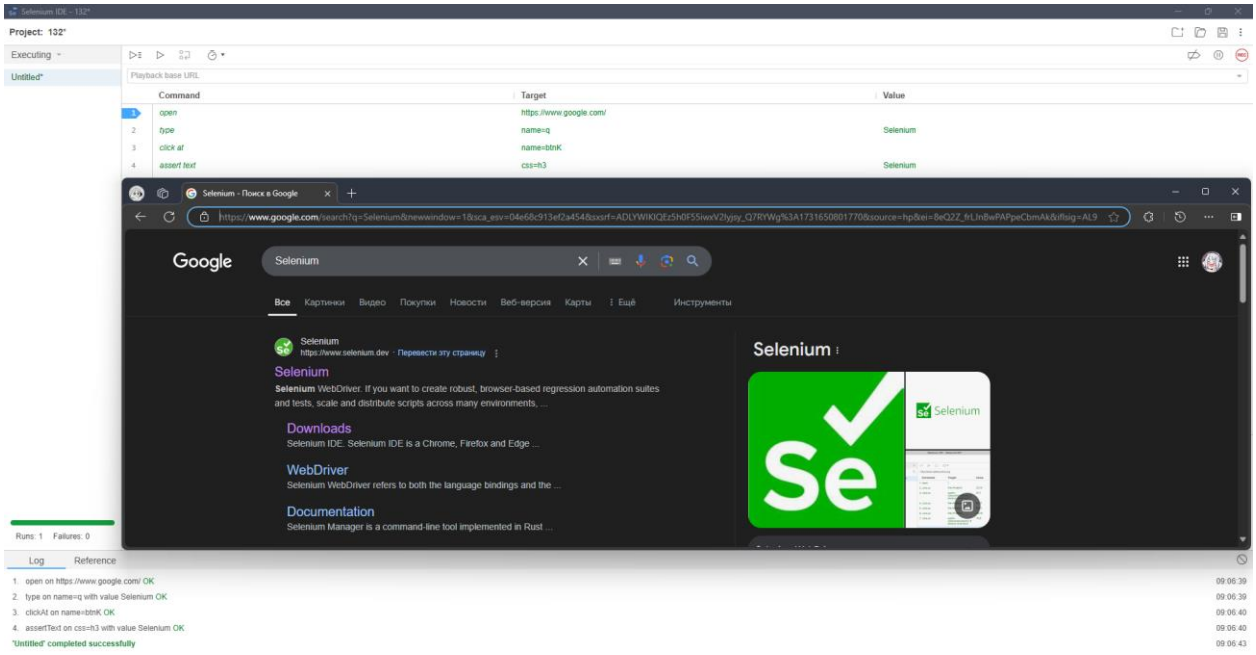

Рисунок 1 – первый скрипт и результат работы

Скрипт 2 на сайте steam.com пытается найти заголовок “Добро пожаловать в Steam ”.

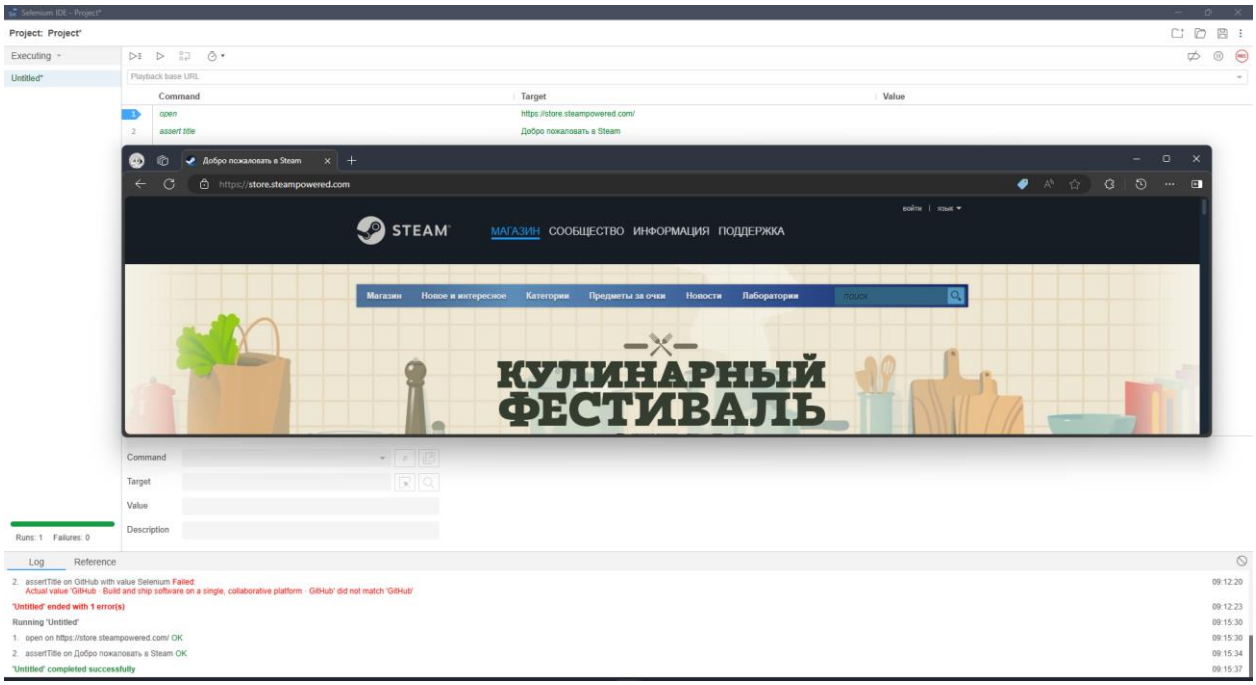

Рисунок 2 – Второй скрипт и результат работы

Скрипт 3 ищет на странице «DeepL» ищет слово «DeepL».

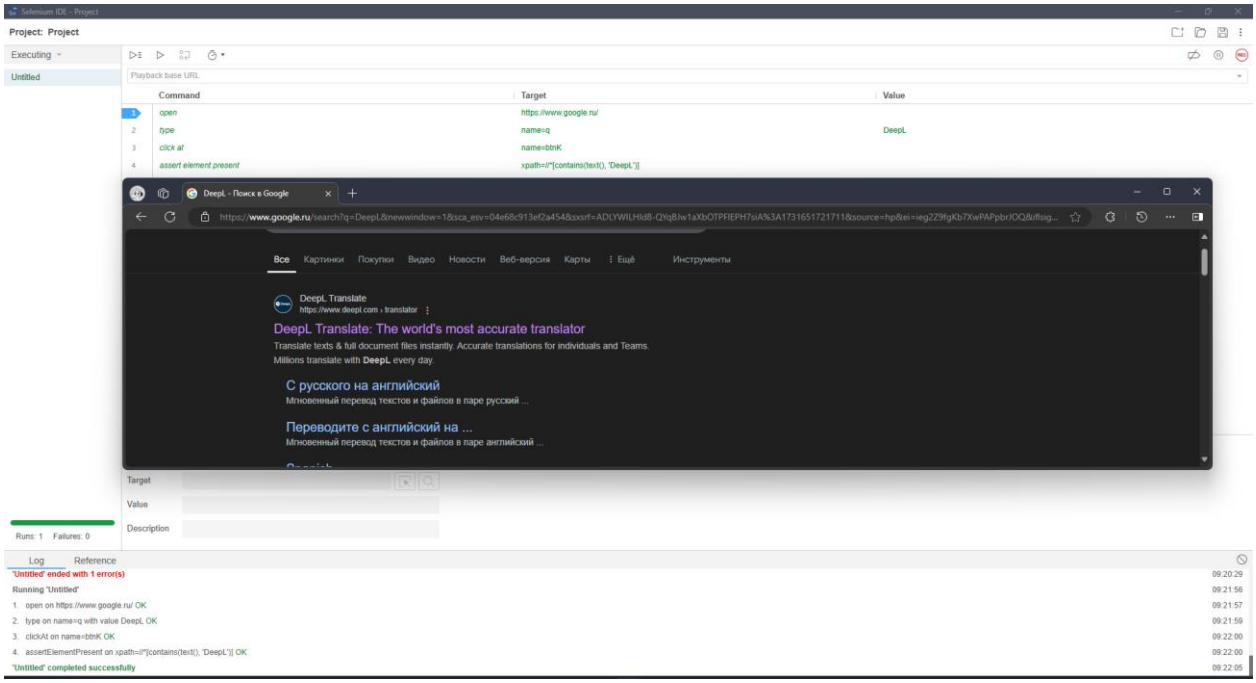

Рисунок 3 – Третий скрипт и результат работы

Заключение

В ходе разработки и тестирования приложения " Кто хочет стать миллионером " была составлена комплексная документация, охватывающая все аспекты тестирования, включая план тестирования, распределение ответственности, график работ и управление рисками. Эти элементы обеспечивают четкую структуру и организованность процесса тестирования, что является критически важным для достижения высокого качества конечного продукта.

План тестирования определяет стратегию и цели, а также конкретные методы и подходы, которые будут использоваться для проверки функциональности и надежности приложения. Распределение ролей и обязанностей среди членов команды способствует эффективному выполнению задач и минимизации рисков. График работ позволяет управлять временными рамками и ресурсами, обеспечивая своевременное выполнение всех этапов тестирования.

Управление рисками и непредвиденными обстоятельствами позволяет заранее подготовиться к возможным проблемам, что снижает вероятность задержек и сбоев в процессе тестирования. Глоссарий терминов служит полезным справочным материалом для участников проекта, помогая им лучше ориентироваться в специфике тестирования программного обеспечения.

В результате проведенной работы команда готова к эффективному тестированию приложения " Кто хочет стать миллионером ", что позволит обеспечить его высокое качество, соответствие требованиям пользователей и успешный запуск на рынок. Уверенность в стабильности и функциональности приложения создаст положительный опыт для пользователей и повысит репутацию продукта.

ПРИЛОЖЕНИЕ

Листинг 1 – Скрипт 1

```
// Generated by Selenium IDE
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Remote;
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.Interactions;
using NUnit.Framework;
[TestFixture]
public class UntitledTest {
  private IWebDriver driver;
  public IDictionary<string, object> vars {get; private set;}
  private IJavaScriptExecutor js;
  [SetUp]
  public void SetUp() {
    driver = new ChromeDriver();
    js = (IJavaScriptExecutor)driver;
    vars = new Dictionary<string, object>();
  }
  [TearDown]
  protected void TearDown() {
    driver.Quit();
  }
  [Test]
  public void untitled() {
    driver.Navigate().GoToUrl("https://www.google.ru/");
    driver.FindElement(By.Name("q")).SendKeys("Selenium");
    driver.FindElement(By.Name("btnK")).Click();
    Assert.That(driver.FindElement(By.CssSelector("h3")).Text,
Is.EqualTo("Selenium"));
  }
}
```

Листинг 2 – Скрипт 2

```
// Generated by Selenium IDE
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Remote;
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.Interactions;
using NUnit.Framework;
[TestFixture]
public class UntitledTest {
  private IWebDriver driver;
  public IDictionary<string, object> vars {get; private set;}
  private IJavaScriptExecutor js;
  [SetUp]
  public void SetUp() {
    driver = new ChromeDriver();
    js = (IJavaScriptExecutor)driver;
    vars = new Dictionary<string, object>();
  }
  [TearDown]
  protected void TearDown() {
    driver.Quit();
  }
  [Test]
  public void untitled() {

driver.Navigate().GoToUrl("https://store.steampowered.com/");
    Assert.That(driver.Title,  Is.EqualTo("Добро  пожаловать  в
Steam"));
  }
}
```

Листинг 3 – Скрипт 3

```
// Generated by Selenium IDE
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Remote;
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.Interactions;
using NUnit.Framework;
[TestFixture]
public class UntitledTest {
  private IWebDriver driver;
  public IDictionary<string, object> vars {get; private set;}
  private IJavaScriptExecutor js;
  [SetUp]
  public void SetUp() {
    driver = new ChromeDriver();
    js = (IJavaScriptExecutor)driver;
    vars = new Dictionary<string, object>();
  }
  [TearDown]
  protected void TearDown() {
    driver.Quit();
  }
  [Test]
  public void untitled() {
    driver.Navigate().GoToUrl("https://www.google.ru/");
    driver.FindElement(By.Name("q")).SendKeys("DeepL");
    driver.FindElement(By.Name("btnK")).Click();
    var                          elements                          =
driver.FindElements(By.XPath("//*[contains(text(),
\'DeepL\')]"));
    Assert.True(elements.Count > 0);
  }
}
```