**Занятие № 25**

**Номер учебной группы:** П-16

**Фамилия, инициалы учащегося:**Мозоль Павел Васильевич

**Дата выполнения работы:**10.12.2022

**Тема работы:** «Формирование тестов»

**Ход работы:**

Таблица 1 – Тестирование интерфейса

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Выполненные действия | Шаги | Ожидаемый результат |
| 1 | Открытие свёрнутого приложения | 1. Нажать кнопку «Свернуть» на устройстве.  2. Выбрать программу IBM Editor1 | 1. Откроется список со свёрнутыми приложениями  2. Программа откроется. |
| 2 | Сворачивание приложения | . Запустить программа  2. Нажать кнопку «Свернуть» на устройстве. | 1. Программа запустится  2. Программа свернётся |
| 3 | Открытие программа | 1. Запустите программа | 1. Программа запустится |
| 4 | Закрытие программы | 1. Запустите программу  2. Дважды нажать кнопку «Назад» | 1. Программа запустится  2. Программа закроется |
| 5 | Проверка режима нескольких окон | 1. Запустить программа  2. Удержать кнопку «Свернуть» на устройстве | 1. Программа запустится  2. Программа растянется на половину экрана. |
| 6 | Проверка работы режима «В отдельном окне» | 1. Запустить программу  2. Включить режим «В отдельном окне» | 1. Программа запуститься  2. Программа перейдёт в режим «В отдельном окне» |

Таблица 2 – Тестирование функционала

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Выполненные действия | Шаги | Ожидаемый результат |
| 7 | Ввод текста русского алфавита | 1. Запустить программу  2. Установить курсор на текстовом поле  3. Ввести текст на русском языке | 1. Программа запуститься  2. Текст отобразился, стабильная работа программы |
| 8 | Ввод текста английского алфавита | 1. Запустить программу  2. Установить курсор на текстовом поле  3. Переключиться на английскую раскладку  4. Ввести текст на английском языке | 1. Программа запуститься  2. Текст отобразился, стабильная работа программы |
| 9 | Ввод специальных символов | 1. Запустить программу  2. Установить курсор на текстовом поле  3. Ввести специальные символы | 1. Программа запуститься  2. Специальные символы отобразился, стабильная работа программы |
| 10 | Вырезать выделенный текст | 1. Запустить программу  2. Ввести произвольный текст  3. Выделить произвольную часть текста  4. Нажать на кнопку «Правка»  5. Выбрать в списке кнопку «Вырезать» | 1. Программа запустится  2. Произойдет вырезка выделенного текста |
| 11 | Копировать выделенный текст | 1. Запустить программу  2. Ввести произвольный текст  3. Выделить произвольную часть текста  4. Нажать на кнопку «Правка»  5. Выбрать в списке кнопку «Копировать» | 1. Программа запустится  2. Произойдет копирование выделенного текста |
| 12 | Вставить скопированный текст | 1. Запустить программу  2. Ввести произвольный текст  3. Предварительно выделить и скопировать произвольную часть текста  4. Нажать на кнопку «Правка»  5. Выбрать в списке кнопку «Вставить» | 1. Программа запустится  2. Произойдет вставка предварительно скопированного текста |
| 13 |  |  |  |
| 14 | Выделить весь текст | 1. Запустить программу  2. Ввести произвольный текст  3. Нажать на кнопку «Правка»  5. Выбрать в списке кнопку «Выделить всё» | 1. Программа запустится  2. Произойдёт выделение всего текста в документе |
| 15 | Просмотр справки | 1. Запустить программу  2. Нажать на кнопку «Справка»  3. Выбрать в списке кнопку «Просмотреть справку» | 1. Программа запустится  2. Откроется руководство пользователя по использованию программы |
| 16 | Просмотр О программе | 1. Запустить программу  2. Нажать на кнопку «Справка»  3. Выбрать в списке кнопку «О программе» | 1. Программа запустится  2. Откроется окно «О программе», содержащие информацию о версии приложения и тд. |

**Контрольные вопросы**

1. Что такое модульное тестирование?

***ОТВЕТ:*** Модульное тестирование (Unit Testing) – это тип тестирования программного обеспечения, при котором тестируются отдельные модули или компоненты программного обеспечения. Его цель заключается в том, чтобы проверить, что каждая единица программного кода работает должным образом.

2. Каким по очередности выполняется модульное тестирование?

***ОТВЕТ:*** Данный вид тестирование выполняется разработчиками на этапе кодирования приложения. Модульные тесты изолируют часть кода и проверяют его работоспособность.

3. Перечислите правила планирования тестов.

***ОТВЕТ:*** Требования для теста указывают, что будет тестироваться. Они относятся к конкретной цели теста. При подготовке требований придерживайтесь следующих общих правил:

* Требование должно относиться к наблюдаемой и измеряемой величине. Если требование относится к величине, которую нельзя наблюдать или измерить, то невозможно будет определить, выполнено ли требование.
* Требования варианта использования или вспомогательные требования не связаны однозначно с требованиями для теста. Часто варианты использования представляют более одного требования для теста, а вспомогательные требования могут представить одно требование, несколько требований или вообще ни одного (например, требования, относящиеся к маркетингу или созданию пакетов).

Источниками требований для тестов могут быть варианты использования, модели вариантов использования, вспомогательные спецификации, проектные требования, варианты бизнес-процесса, интервью с пользователями и документ архитектуры программного обеспечения. Все эти источники должны быть критическим образом пересмотрены, чтобы сформулировать требования для теста.

4. Что такое интеграционное тестирование?

***ОТВЕТ:*** Интеграционное тестирование – это тип тестирования, при котором программные модули объединяются логически и тестируются как группа. Как правило, программный продукт состоит из нескольких программных модулей, написанных разными программистами.

5. В чем отличие модульного тестирования от интеграционного?

***ОТВЕТ:* Модульное тестирование:** это тестирование, фактически выполняемое разработчиками, обладающими знаниями в области программирования. Это тестирование проводится на этапе кодирования и является частью тестирования белого ящика. Когда программное обеспечение поступает на разработку, оно разрабатывается в виде фрагмента кода или фрагментов кода, известных как единое целое.

**Функциональное тестирование:** это тестирование проводится на этапе тестирования (QA) и является частью тестирования черного ящика. Фактическое выполнение ранее написанных тестовых случаев. Это тестирование фактически проводится тестировщиками, они находят фактический результат любой функциональности на сайте и сравнивают этот результат с ожидаемым результатом. Если они обнаружили какое-либо несоответствие, то это ошибка.

6. Какие методы сборки модулей вы знаете?

***ОТВЕТ:*** Тем не менее, если вы придерживаетесь не нативных систем модулей, которые не могут интерпретироваться браузерами, например, CommonJS или AMD (или даже нативный ES6), то вам необходимо использовать специальный инструмент для преобразования модулей в упорядоченный и понятый браузеру код. Вот здесь то и вступают в игру Browserify, RequireJS, Webpack и другие "сборщики модулей" или "загрузчики модулей". В дополнении к сборке и/или загрузке ваших модулей, сборщики модулей предлагают массу дополнительных возможностей, например, авто-перекомпиляция кода при внесении изменений

7. Опишите восходящее тестирование.

***ОТВЕТ:*** Восходящее тестирование - это прекрасный способ локализации ошибок. Если ошибка обнаружена при тестировании единственного модуля, то очевидно, что она содержится именно в нем - для поиска ее источника не нужно анализировать код всей системы.

8. Опишите нисходящее тестирование.

***ОТВЕТ:*** Нисходящее тестирование (нисходящая разработка). Изолированно тестируется только головной модуль. После завершения тестирования этого модуля с ним соединяются модули, непосредственно вызываемые им, и тестируется полученная комбинация.

9. Перечислите плюсы и минусы восходящего и нисходящих тестирований.

***ОТВЕТ:*** Верификация и аттестация системной архитектуры. При нисходящем тестировании больше возможностей выявить ошибки в архитектуре системы на раннем этапе процесса разработки. Обычно это структурные ошибки, раннее выявление которых предполагает их исправление без дополнительных затрат. При восходящем тестировании структура высокого уровня не утверждается вплоть до последнего этапа разработки системы.

Демонстрация системы. При нисходящей разработке незаконченная система вполне пригодна для работы уже на ранних этапах разработки. Этот факт является важным психологическим стимулом использования нисходящей модели разработки систем, поскольку демонстрирует осуществимость управления системой. Аттестация проводится в начале процесса тестирования путем создания демонстрационной версии системы. Но если система создается из повторно используемых компонентов, то и при восходящей разработке также можно создать ее демонстрационную версию.

Реализация тестов. Нисходящее тестирование сложно реализовать, так как необходимо моделировать программы-заглушки нижних уровней. Программы-заглушки могут быть упрощенными версиями представляемых компонентов. При восходящем тестировании для того, чтобы использовать компоненты нижних уровней, необходимо разработать тестовые драйверы, которые эмулируют окружение компонента в процессе тестирования.

Наблюдение за ходом испытаний. При нисходящем и восходящем тестировании могут возникать проблемы, связанные с наблюдениями за ходом тестирования. В большинстве систем, разрабатываемых сверху вниз, более верхние уровни системы, которые реализованы первыми, не генерируют выходные данные, однако для проверки этих уровней нужны какие-либо выходные результаты. Испытатель должен создать искусственную среду для генерации результатов теста. При восходящем тестировании также может возникнуть необходимость в создании искусственной среды (тестовых драйверов) для исследования компонентов нижних уровней.

10. Что такое системное тестирование?

***ОТВЕТ:*** Системное тестирование — это уровень тестирования, который проверяет законченный и полностью интегрированный программный продукт. Целью системного теста является оценка сквозных технических характеристик системы.

11. Какие виды тестов используются при системном тестировании?

***ОТВЕТ:***

• функциональное тестирование;

• тестирование производительности;

• нагрузочное или стрессовое тестирование;

• тестирование конфигурации;

• тестирование безопасности;

• тестирование надежности и восстановления после сбоев;

• тестирование удобства использования