|  |  |
| --- | --- |
|  | **Тестовый документ** |

## **Аннотация теста**

|  |  |
| --- | --- |
| **Название проекта** | StroyModern |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Кирилл |
| **Дата(ы) теста** | 15.02.2024 |

## **Расшифровка тестовых информационных полей:**

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| **Название проекта** | Название тестируемого проекта |
| **Рабочая версия** | Версия проекта/программного обеспечения (первый тест считается 1.0). |
| **Имя тестирующего** | Имя того, кто проводил тесты |
| **Дата(ы) теста** | Дата(ы) проведения тестов – это один или несколько дней. Если тесты проводились в более протяженный период времени, нужно отметить отдельную дату для каждого теста. |
| **Тестовый пример #** | Уникальный ID для каждого тестового примера. Следуйте некоторым конвенциям, чтобы указать типы тестов. Например,‘TC\_UI\_1′ означает‘user interface test case #1′ ( ТС\_ПИ\_1: тестовый случай пользовательского интерфейса#1) |
| **Приоритет тестирования**  *(Низкий/Средний/Высокий)* | Насколько важен каждый тест. Приоритет тестирования для бизнес-правил и функциональных тестовых случаев может быть средним или высоким, в то время как незначительные случаи пользовательского интерфейса могут иметь низкий приоритет. |
| **Заголовок/название теста** | Название тестового случая. Например, Подтвердите страницу авторизации с действительным именем пользователя и паролем. |
| **Краткое изложение теста** | Описание того, что должен достичь тест. |
| **Этапы теста** | Перечислите все этапы теста подробно. Запишите этапы теста в том порядке, в котором они должны быть реализованы. Предоставьте как можно больше подробностей и разъяснений. Пронумерованный список – хорошая идея. |
| **Тестовые данные** | Перечислите/опишите все тестовые данные, используемые для данного тестового случая. Так, фактические используемые входные данные можно отслеживать по результатам тестирования. Например, Имя пользователя и пароль для подтверждения входа. |
| **Ожидаемый результат** | Каким должен быть вывод системы после выполнения теста? Подробно опишите ожидаемый результат, включая все сообщения/ошибки, которые должны отображаться на экране. |
| **Фактический результат** | Каким должен быть фактический результат после выполнения теста? Опишите любое релевантное поведение системы после выполнения теста. |
| **Предварительное условие** | Любые предварительные условия, которые должны быть выполнены до выполнения теста. Перечислите все предварительные условия для выполнения этого тестового случая. |
| **Постусловие** | Каким должно быть состояние системы после выполнения теста? |
| **Статус**  *(Зачет/Незачет)* | Если фактический результат не соответствует ожидаемому результату, отметьте тест как неудачный. В ином случае обновление пройдено. |
| **Примечания/комментарии** | Используйте эту область для любых дополнительных заметок/комментариев/вопросов. Эта область предназначена для поддержки вышеуказанных полей (например, если есть некоторые особые условия, которые не могут быть описаны в любом из вышеуказанных полей, или если есть вопросы, связанные с ожидаемыми или фактическими результатами). |

## Тестовый пример #1:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_isValidUser\_1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка правильности проверки пользователя и его роли |
| **Краткое изложение теста** | Проверить, что возвращается верная роль после указания пароля и логина пользователя |
| **Этапы теста** | 1. Создать строку «Ожидаемая роль»; 2. Вызвать метод «isValidUser» 3. Создать строку «Ожидаемая роль2»; 4. Вызвать метод «isValidUser» 5. Проверка правильности ожидаемых и полученных ролей. |
| **Тестовые данные** | * Роль №1: Администратор * Роль №2: Менеджер * Пользователь №1: Логин: admin, Пароль: 1234 * Пользователь №2: Логин: manager, Пароль: 1234 |
| **Ожидаемый результат** | Из метода должна вернуться верные роли пользователей |
| **Фактический результат** | Метод вернул корректные роли указанных пользователей |
| **Статус** | Зачёт |
| **Предварительное условие** | Необходимо чтобы Пользователь №1 и Пользователь №2 существовали в базе данных |
| **Постусловие** | Метод пройден |
| **Примечания/комментарии** | - |

## Тестовый пример #2:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_deleteUser\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Проверка метода удаления пользователя по id |
| **Краткое изложение теста** | Проверить, что пользователь удаляется из базы данных |
| **Этапы теста** | 1. Указать id; 2. Вызвать метод «DeleteUser», с переданным и уже указанным ранее id. 3. Вызывать метод «isValidUser» с данными предположительно удаленного пользователя. 4. Проверить, что роль пустая |
| **Тестовые данные** | Пользователь: Логин – admin, Пароль – 1234, Id - 100 |
| **Ожидаемый результат** | Пользователь удален из базы данных, роль пустая |
| **Фактический результат** | Пользователь удален из базы данных, роль пустая |
| **Статус** | Зачтено |
| **Предварительное условие** | Пользователь с Id 100 и логином: «user» и паролем: «1234» должен существовать в базе данных |
| **Постусловие** | Пользователь удален |
| **Примечания/комментарии** | - |

## Тестовый пример #3:

|  |  |
| --- | --- |
| Тестовый пример **#** | TC\_updateOrder\_1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка корректности обновления заказа |
| **Краткое изложение теста** | Проверить, что заказ был успешно изменен |
| **Этапы теста** | 1. Указать полное имя; 2. Указать адрес; 3. Указать id заказа; 4. Вызвать метод «UpdateOrder», с полным именем, адресом и id, которые были указаны ранее; 5. Проверить, что данные были успешно изменены с помощью метода «GetAllOrderInfo». |
| **Тестовые данные** | Полное имя: «Яков Аким Абдулаевич»  Пароль: «ул. Заморская, д.25, кв.43»  Id: 2 |
| **Ожидаемый результат** | Данные о заказе были изменены |
| **Фактический результат** | Данные о заказе были успешно изменены |
| **Статус** | Зачет |
| **Предварительное условие** | Заказ с указанным id должен существовать в базе данных |
| **Постусловие** | Заказ с указанным id был изменен |
| **Примечания/комментарии** | - |

## Тестовый пример #4:

|  |  |
| --- | --- |
| Тестовый пример **#** | TC\_createProduct\_1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка создания товара |
| **Краткое изложение теста** | Проверить, что товар был успешно создан |
| **Этапы теста** | 1. Создать объект «Товар» 2. Вызван метод «InsertProduct», с переданными товаром; 3. Вызван метод на получение созданного товара; 4. Проверка того, что объект «Товар» не пустой; 5. Проверка того, что фактические значения полей совпадают с указанными. |
| **Тестовые данные** | - Товар: Название: «Цемент», стоимость: 500 руб., артикул: 1298473285746, изображение: «Resources\цемент.jpg», количество на складе: 200, номер цеха: 123, тип товара: «Строительные материалы» |
| **Ожидаемый результат** | Товар был создан с указанными тестовыми данными |
| **Фактический результат** | Товар был успешно создан с указанными тестовыми данными |
| **Статус** | Зачтено |
| **Предварительное условие** | Продукта с названием, которое указано в тестовых данных, не должно существовать в базе данных |
| **Постусловие** | Создан новый товар |
| **Примечания/комментарии** | - |

## Test case #5:

|  |  |
| --- | --- |
| **Test Case #** | TC\_getAllTypes\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Проверка получения всех типов товара |
| **Краткое изложение теста** | Проверка наличия и правильного получения типов товара |
| **Этапы теста** | 1. Вызывать метод «GetAllTypes» 2. Проверить, что полученный список не нулевой 3. Проверить, что полученный список не пустой |
| **Тестовые данные** | - |
| **Ожидаемый результат** | Типы товара были получены верно |
| **Фактический результат** | Типы товара были получены верно |
| **Статус** | Зачёт |
| **Предварительное условие** | Хотя бы один тип товара должен существовать в базе данных |
| **Постусловие** | Были получены все типы товаров |
| **Примечания/комментарии** | - |