

# 4. Разработка тестов и тестирование программного продукта

## 4.1. Разработка плана тестирования

### 4.1.1. Введение

Данный документ описывает план тестирования для системы «Разработка распределенной системы мониторинга технического состояния экзомассажоров» согласно спецификации. Полная стратегия тестирования программного обеспечения состоит из следующих типов испытаний и выполняется в следующем порядке:

1. Тестирование компонентов (модульное тестирование). Тестируются все программные компоненты (при этом проверяется покрытие кода тестами). Анализ кода.
2. Тестирование интеграции. Тестируется программное обеспечение, чтобы гарантировать, что компоненты взаимодействуют правильно.

**4.1.2. Область тестирования**

Целью этого тестирования является определить, соответствует ли разработанное программное обеспечение заявленным функциональным требованиям, а также выявить ошибки и представить их исправления, улучшив тем самым качество разработанного программного обеспечения.

Основные области тестирования включают:

*Функциональное тестирование*: Проверка выполнения всех функциональных требований системы.

*Производительность*: Оценка времени отклика системы, пропускной способности и масштабируемости.

*Надежность*: Проверка стабильности и устойчивости системы под различными условиями эксплуатации.

*Безопасность*: Проверка системы на уязвимости и соответствие требованиям безопасности.

**4.1.3. Начальные условия**

Перед началом тестирования необходимо выполнить следующие задачи:

1. Имеется законченная программная спецификация: Полное описание функциональности и нефункциональных требований к системе
2. Работающее программное обеспечение: Система должна быть готова к тестированию, включая установку и настройку всех компонентов.
3. Все необходимые компоненты находятся в рабочем состоянии: Все аппаратные и программные компоненты, необходимые для работы и тестирования системы, должны быть правильно установлены и функционировать.

### 4.1.4. Типы тестирования

Для всестороннего тестирования системы будут использоваться следующие типы тестирования:

1. Модульное тестирование: Тестирование отдельных модулей или компонентов системы. Целью является проверка правильности работы каждого компонента в отдельности. Модульное тестирование будет выполняться с использованием автоматизированных тестов.
2. Интеграционное тестирование: Проверка взаимодействия между модулями и компонентами системы. Целью является выявление проблем, возникающих при интеграции различных частей системы.
3. Системное тестирование: Комплексное тестирование всей системы целиком для проверки соответствия системы спецификациям и требованиям. Включает в себя функциональное, производительное, нагрузочное и стресс-тестирование.
4. Приемочное тестирование: Финальное тестирование, проводимое с целью подтверждения того, что система соответствует всем требованиям заказчика и готова к эксплуатации. Включает в себя тестирование с реальными данными и сценариями использования.

Составлен следующий набор тестов (табл. 4.1):

Таблица 4.1. Набор тестов для тестирования системы

|  |  |  |
| --- | --- | --- |
| № | Тестовый вариант | Ожидаемый результат |
| 1 | Добавление нового устройства | Устройство успешно добавлено в базу данных |
| 2 | Авторизация пользователя | Пользователь успешно вошел в систему |
| 3 | Регистрация пользователя при помощи кода | Пользователь успешно зарегистрирован |
| 5 | Регистрация пользователя | Пользователь успешно зарегистрирован |
| 6 |  |  |
| 7 |  |  |

### 4.1.4. Приоритеты тестирования

Проверки перечислены в порядке уменьшения приоритетного уровня:

1. Функционал - все ли заданные функции Web-платформы выполняются, как ожидалось?
2. Удобство и простота - реально ли Web-платформа удобна и легко понимаема для пользователя?
3. Защита – тщательно ли обеспечивается защита данных?
4. Выполнение – Web-платформа соответствует всем критериям выполнения?

### 4.1.5. Методы тестирования

Будут использоваться **тестовые сценарии** – сценарии вариантов использования (с предопределенным вводом и ожидаемыми выходными данными), варианты использования взяты из диаграммы вариантов использования.

### 4.1.6. Среда тестирования

Для тестирования данного программного продукта необходимо следующее оборудование:

Персональный компьютер со следующей конфигурацией:

* Ryzen 7 2700, 16Gb
* Операционная система Microsoft Windows 10
* Установленные программы jetbrains pycharm, Google Chrome

## 4.2. Модульное тестирование

**Модульное тестирование** — это процесс программирования, который позволяет проверять правильность блоков исходного кода, правильную работу программных модулей, а также наборов одного или нескольких программных модулей вместе с соответствующими управляющими данными. [4]

Суть метода заключается в создании методов тестирования для тестирования отдельного метода в программе. Таким образом, можно проверить правильность кода при его изменении и добавлении функциональности в программу. Кроме того, модульные тесты позволяют обнаруживать ошибки в работе программных алгоритмов, что, в свою очередь, повышает качество и надежность программного обеспечения.

Цель модульного тестирования — выделить отдельные модули программы и проверить работу этих модулей в соответствии с определенными входными данными.

### 4.2.1. Тестовые случаи

Шаги тестирования:

* + - 1. Администратор авторизовывается на сайте и нажимает кнопку добавить нового редактора
      2. Вводит данные Логин и пароль

**Описание тестового случая**:

Тестирование метода register() – проверка создания нового пользователя в базе данных (табл. 4.2).

Таблица 4.2. Тестовые варианты метода register()

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Что проверяется | Ожидаемый результат |
| 1 | Логин: user | Наличие в базе данных пользователя с таким же логином   * Пользователь с таким логином не найден | Сообщение об успешной регистрации. |
| 2 | Логин: user | Наличие в базе данных пользователя с таким же логином   * Пользователь с таким логином найден | Сообщение, что пользователь с таким логином уже существует. |

**Описание тестового случая**:

Шаги тестирования:

1. Администратор заходит на сайт и нажимает кнопку авторизоваться
2. Вводит данные Логин и пароль

Тестирование метода login() – проверка создания нового пользователя в базе данных (табл. 4.3).

Таблица 4.3. Тестовые варианты метода login()

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Что проверяется | Ожидаемый результат |
| 1 | Логин: user | Корректность пароля   * Пароль соответствует | Переход на главную страницу. |
| 2 | Логин: user | Корректность пароля   * Пароль не соответствует | Сообщение, что пароль не верный. |

**Описание тестового случая**:

Тестирование метода create\_journal () – проверка добавления журнала в базу данных (табл. 4.4).

Таблица 4.4 Тестовые варианты create\_journal ()

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Что проверяется | Ожидаемый результат |
| 1 | name | Есть ли в базе данных уже журнал с таким же именем   * Такого журнала нет | Добавление содержимого в бд. |
| 2 | name | Есть ли в базе данных уже журнал с таким же именем   * Такой журнал уже есть | Сообщение, что такие данные уже есть |

## 4.3. Системное тестирование

Системное тестирование — это тестирование программного обеспечения, выполняемое на полной, интегрированной системе, с целью проверки соответствия системы исходным требованиям.

Системное тестирование выполняется методом “черного ящика”.

Было проведено тестирование соответствия системы функциональным требованиям, изложенным в спецификации вариантов использования.

Тестирование системы показало ее соответствие функциональным требованиям.

Система правильно реагирует на неверный ввод данных пользователем, выдавая ему подсказки. На рис. 4.1 показан пример неверного ввода с последующим выводом сообщений об ошибках.

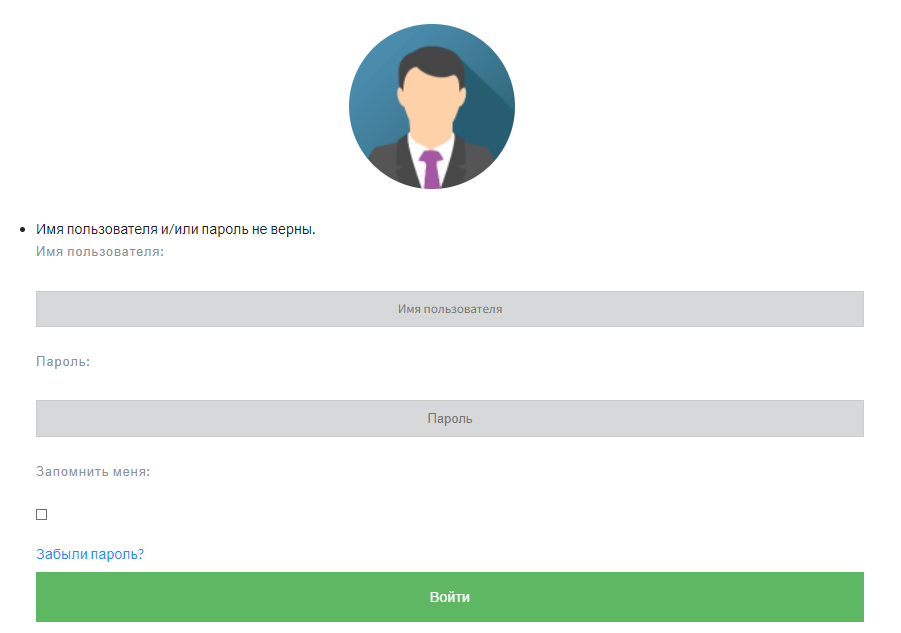


Рисунок 4.1. Попытка входа с неверными данными