Государственное бюджетное профессиональное образовательное учреждение  
“Нижегородский Губернский колледж”

Специальность: Информационные системы и программирование.

**План тестирования**по дисциплине «МДК 05.01 Проектирование и дизайн информационных систем»

Выполнила студентка группы 32В:

Самыловская А.П.

Нижний Новгород

2021

**Оглавление**

[**Введение** 3](#_Toc76293461)

[**Цели тестирования** 3](#_Toc76293462)

[**Виды тестирования** 4](#_Toc76293463)

# **Введение**

В настоящем плане тестирования описаны и определены стратегия и принципы тестирования, применяемые при тестировании системы удаленного доступа. План будут использовать исполнители работ для получения представления о тестировании на проекте. Документ определяет распределение обязанностей при тестировании и описывает тесты, намеченные к выполнению. План тестирования разработан для решения следующих задач:

* Спланировать управление тестированием и техническую поддержку тестирования в ходе всего жизненного цикла разработки системы.
* Определить исчерпывающий план тестирования, который описывает природу и рамки тестирования, достаточные для достижения целей и решения задач тестирования в проекте.

# **Цели тестирования**

Цели тестирования Основными целями тестирования являются:

* обеспечение выполнения всех системных требований и критериев, установленных к программному продукту;
* повышение вероятности того, что приложение при любых обстоятельствах будет функционировать надлежащим образом и соответствовать установленным требованиям за счет обнаружения максимально возможного числа дефектов;
* обеспечение работоспособности каждого разрабатываемого модуля согласно спецификации требований, к данному модулю;
* обеспечение работоспособности всей системы в целом согласно спецификации требований, к системе;
* обеспечение отказоустойчивости системы и каждого отдельного модуля;
* обеспечение установленных параметров производительности;
* обеспечение нормального качества исходных материалов и исходных кодов;
* оперативное информирование заинтересованных лиц об уровне качества регулярных сборок;
* обеспечение пользователя наиболее удобным графическим интерфейсом.

# **Виды тестирования**

В идеале необходимо покрыть тестами каждую строку исходного кода. Когда продукт находится на ранней стадии разработки — исправление ошибок обходится гораздо дешевле. По мере продвижения разработки выявление и исправление ошибок становится все более и более затратным. В большинстве случаев предоставление модульных тестов является ответственностью разработчика, их создание производится в момент разработки класса.

Интеграционное тестирование — после разработки тестов на отдельные классы необходимо проверить, как они будут работать вместе в рамках одного исполняемого процесса. Необходимо проверить, как соотносятся классы, разработанные по-разному разными разработчиками. Данный вид тестирования базируется на предыдущем и также производится на уровне исходного кода. Обычно тестовые примеры строятся на основе вызова одного компонента из другого.

Сквозное тестирование — проверяет работоспособность компонентов системы на уровне взаимодействия нескольких отдельных исполняемых процессов. На данном этапе тестируется функционирование клиентсерверных систем, их взаимодействие внутри и с внешними компонентами. Обычно сквозные тесты содержатся во внешнем по отношению к системе процессе, который делает тестирующие запросы. Так проверяются функции макроуровня, надежность, производительность, координация.

Функциональное тестирование — рассматривает продукт, состоящий из множества классов, процессов, компонентов, данных как единое целое. На этом этапе проверяется в целом его работоспособность, функциональные и технические характеристики, а также бизнес-логика. Такая проверка может осуществляться в нескольких конфигурациях окружения оборудования и наборов данных.

Тестирование интерфейса — проверка клиентских и административных интерфейсов пользователя на возможность выполнения с их помощью сценариев использования. Сценарий использования представляет собой последовательность действий пользователя, которые имитируют его активность при работе с интерфейсами системы. Сценарий использования должен покрывать спецификацию требований к пользовательскому интерфейсу. Такое тестирование производится в автоматическом режиме с помощью специализированных утилит. Тестирование должно проверять корректность работы интерфейсной части приложения при любых возможных настройках экрана (различное разрешение, масштаб, шрифт), при изменениях фокуса, при работе с мышью и клавиатурой.

Нагрузочное тестирование — определение и проверка характеристик производительности системы в заданной конфигурации оборудования и набора данных.

Тестирование базы данных — проверка функционирования внешней базы данных и хранимых процедур в соответствии со спецификацией требований. Проверка политики безопасности доступа к базе в соответствии с ролями системы. Определение и проверка характеристик базы данных, таких как производительность, среднее время доступа, максимальное количество обслуживаемых клиентов, минимальная и максимальная длительность обработки запроса и т.п.

Тестирование безопасности — определение ролей и проверка списка функций системы, доступных для каждой роли. Может осуществляться на уровне интерфейса, на уровне компонента, на уровне базы данных, на уровне модуля и на сетевом уровне. Проводится в соответствии с принятым документом «Политика безопасности». Включает проверку методов шифрования данных при хранении и передаче, отказа доступа к запрещенным функциям, перехвата данных, подделки удостоверения личности, отказа в обслуживании и других атак.

Тестирование удобства использования интерфейса — разработка отчета об удобстве использования, быстроте освоения, наглядности пользовательских интерфейсов системы. Данный отчет может содержать предложения по улучшению этих характеристик.

Тестирование конфигурации — проверка работоспособности системы в заданном окружении конфигурации оборудования и набора данных.

Верификация — проверка успешности исправления разработчиком ошибки, проведенная тестировщиком в тестовом окружении.

Регрессивное тестирование — повторное выборочное тестирование продукта с модифицированными частями после исправления ошибки или добавления новой функции. Внесение изменений в исходный код может повлечь цепочку зависимостей и получение новых ошибок во взаимозависимых функциях. Данный вид тестирования минимизирует риск подобного события