**Теоретическая часть**

**Основы тестирования:**

1. **Что такое тестирование ПО? Каковы его цели?**

Тестирование ПО – это процесс оценки программного обеспечения с целью выявления ошибок, а также проверка соответствия программы техническому заданию.

Основная цель – убедиться, что ПО соответствует требованиям, работает корректно и обеспечивает ожидаемое качество. Тестирование также направлено на минимизацию рисков, связанных с выпуском некачественного продукта.

1. **Какие существуют виды тестирования? Чем они отличаются?**

Различают множество видов тестирования по различным свойствам:

* По степени автоматизации
  + Ручное – выполняется тестировщиком вручную, так как это делал конечный пользователь, без специализированного ПО
  + Автоматизированное тестирование – это проверка ПО с использованием специального ПО и скриптов
* По уровню тестирования
  + Модульное или «Unit» – проверка корректности работы отдельных модулей или компонентов программы. Каждый модуль тестируется независимо от других
  + Интеграционное - проверка взаимодействия между отдельными модулями или компонентами программы. Тестируются группы связанных модулей
  + Системное - проверка функциональности и соответствия ПО требованиям заказчика. Проводится тестирование всей системы, включая её компоненты и подсистемы
  + Приемочное - проверка, соответствует ли система требованиям заказчика и готова ли она к внедрению

1. **В чем разница между валидацией и верификацией?**

Валидация — это проверка соответствия продукта потребностям пользователя и его требованиям к функциональности. Цель - убедиться, что продукт решает проблемы пользователей

Верификация – это проверка того, что продукт или система соответствуют заданным требованиям и спецификациям. Цель - убедиться, что процесс идёт правильно

1. **Какие методологии тестирования вы знаете? Как они применяются в проектах?**

* Каскадный «Waterfall» метод — сначала заканчивают один, потом начинают другой. Каждый этап должен быть полностью завершен, прежде чем начнется следующий, формируя своего рода “каскад” задач. Наиболее эффективен в проектах с четко определенными и стабильными требованиями, где требуется высокая предсказуемость и повторяемость результатов.
* Scrum метод — частный случай реализации Agile методологии для гибкого управления проектами. Суть Agile - разбить проект на небольшие циклы (итерации или спринты), каждая из которых направлена на создание рабочего продукта. Agile позволяет гибко реагировать на изменения в проекте и корректировать задачи по мере необходимости. Scrum предполагает деление всего времени, отведённого на реализацию проекта, на равные временные отрезки — спринты, в течение которых команда поэтапно продвигается к готовому продукту.

1. **Чем отличается функциональное тестирование от нефункционального?**

Функциональное тестирование – проверка соответствия ПО заявленным функциям и требованиям (например, авторизация пользователя). К функциональным тестам относят, к примеру, unit или системное тестирование. Цель – проверить ли работает в принципе ПО.

Нефункциональное тестирование – оценка аспектов, не связанных с конкретными функциями (например, производительность, безопасность, удобство использования). К нефункциональным тестам относят, к примеру, нагрузочное или usability тестирование. Цель – насколько хорошо работает ПО.

1. **В чем разница между позитивным и негативным тестированием?**

Позитивное тестирование – проверка ПО с использованием корректных входных данных и ожидание успешного результата. Проверка работоспособности ПО.

Негативное тестирование – проверка обработки некорректных или неожиданных входных данных с целью выявления ошибок и сбоев. Проверка отказоустойчивости ПО.

1. **Что такое регрессионное тестирование и зачем оно нужно?**

Регрессионное тестирование - повторное тестирование уже протестированных функций после внесения изменений. Проверяет - не повлияли ли изменения на существующую функциональность и не появились ли новые ошибки. Основная цель - убедиться, что исправления не создали новых ошибок.

1. **В чем разница между тестированием «черного ящика» и «белого ящика»?**

Тестирование «черного ящика» - тестирование функциональности ПО, когда внутренняя структура проекта не известна. Все что доступно для анализа – это входные и выходные данные. Тестировщик имитирует взаимодействие с программой от лица конечного пользователя.

Тестирование «белого ящика» - проверка функциональности ПО, когда тестировщику известна структура и исходный код проекта. Применяется для более детального тестирования внутренней архитектуры и контроля внутренней логики исполнения проекта.

**Тестовая документация:**

1. **Какие виды тестовой документации вы знаете?**

Основные виды тестовой документации:

* тест-планы (определяют стратегию тестирования)
* тест-кейсы (описывают конкретные шаги для проверки функциональности)
* чек-листы (краткие перечни проверок)
* отчеты о тестировании (результаты тестирования и выводы)
* баг-репорты (описание обнаруженных дефектов)

1. **Что такое тест-кейс? Из чего он состоит?**

Тест-кейс – это подробное описание шагов, необходимых для проверки конкретной функциональности. Он состоит из: идентификатора, названия, предусловия, шагов исполнения, ожидаемого результата, фактического результата, статуса кейса (пройден/провален)

1. **Чем тест-кейс отличается от чек-листа?**

Тест-кейс – подробный документ, детально описывающий конкретные шаги тестирования определенной функции проекта. Чек-лист - более общий перечень пунктов, в котором отражены ключевые аспекты проверяемой системы.

1. **Что такое тестовый план и что он должен содержать?**

Тестовый план – это документ, описывающий цели, стратегию, ресурсы и расписание тестирования. Он должен содержать: цели тестирования, scope (какую функциональность тестируем), подходы к тестированию, критерии начала и завершения тестирования, ресурсы, риски, план работ. Тестовый план служит руководством для процесса тестирования.

1. **Как задокументировать найденный дефект?**

Каждый дефект должен быть задокументирован в баг-репорт.

1. **Какие атрибуты включает баг-репорт?**

Баг-репорт должен содержать: идентификатор, заголовок (суть дефекта в одном предложении), описание дефекта (развернутое описание), шаги воспроизведения ошибки (описать каждое действие), ожидаемый результат, фактический результат, приоритет (насколько критичен дефект для работоспособности проекта), скриншот или логи дефекта, среда выполнения (браузер/ОС)

1. **Как определить приоритет и серьезность бага? Приведите примеры.**

* Приоритет - определяет порядок исправления дефекта
  + Критический - требуется немедленное исправление
  + Высокий - исправить как можно быстрее
  + Средний - исправить в
  + Низкий - исправить при возможности
* Серьезность - оценивает влияние дефекта на функциональность.
  + Критический - приложение не работает (например, невозможность авторизации или ошибка получения доступа к ресурсу)
  + Высокий - функция не работает или работает некорректно (например, некорректные расчеты стоимости)
  + Средний - незначительные проблемы (например, ошибки в отображении интерфейса)
  + Низкий - косметические проблемы (например, незначительные опечатки в тексте)

**SQL и базы данных:**

1. **Какие SQL-операторы вы знаете? Как они применяются?**

Самые основные SQL-операторы:

* Определение данных (DDL - Data Definition Language)
  + CREATE (Определение новых объектов в БД)
  + ALTER (Изменение существующих объектов в БД)
  + DROP (удаление объекта в БД)
* Управление данными (DML - Data Manipulation Language)
  + SELECT (выборка)
  + INSERT (добавление)
  + UPDATE (изменение)
  + DELETE (удаление)

1. **Как с помощью SQL можно отфильтровать данные по определенному условию?**

Для фильтрации по условию используется оператор WHERE.

Пример: Выборка из таблицы students (поля: id, name, grade), где отобразятся только студенты из 4 класса.

SELECT id, name

FROM students

WHERE grade = 4;

1. **Что делает оператор JOIN в SQL? Какие его виды существуют?**

Оператор JOIN используется для объединения строк из двух или более таблиц на основе связанного столбца.

Основные виды:

* INNER JOIN (возвращает только совпадающие строки)
* LEFT JOIN (возвращает все строки из левой таблицы и совпадающие из правой)
* RIGHT JOIN (возвращает все строки из правой таблицы и совпадающие из левой)
* FULL OUTER JOIN (возвращает все строки из обеих таблиц)

1. **Как отсортировать данные в SQL-запросе?**

Для сортировки данных в SQL используется оператор ORDER BY. В синтаксисе SQL после ORDER BY указывается столбец, по которому нужно сортировать, и направление сортировки: ASC (по возрастанию, по умолчанию) или DESC (по убыванию). Можно сортировать по нескольким столбцам, указывая их через запятую.

Пример: Таблица products (поля: id, name и price). Нужно выбрать все продукты и отсортировать их по цене в порядке возрастания.

SELECT id, name, price

FROM products

ORDER BY price ASC;

**Тестирование API и Postman:**

1. **Какие методы HTTP-запросов вы знаете?**

Основные HTTP-методы:

* GET (получение данных)
* POST (отправка данных для создания ресурса)
* PUT (обновление ресурса)
* PATCH (частичное обновление ресурса)
* DELETE (удаление ресурса)

1. **В чем разница между GET и POST?**

GET используется для получения данных с сервера.

POST используется для отправки данных на сервер, например для создания или изменения ресурса.

1. **Какой код состояния сервера говорит об успешном выполнении запроса?**

Коды категории **2xx** – указывают на успешное выполнение запроса. Пример:

* Код «**200 ОК**» - успешный запрос (GET, PUT, PATCH)
* Код «**201 Created**» указывает на успешное создание ресурса (POST)

1. **Что такое JSON и чем он отличается от XML?**

JSON (JavaScript Object Notation) – это формат обмена данными, основанный на текстовом представлении объектов JavaScript.

XML (Extensible Markup Language) – более сложный формат с использованием тегов.

JSON проще для чтения и обработки, особенно в веб-приложениях, и более распространен в современных API.

1. **Как с помощью Postman можно протестировать API без документации?**

В Postman можно исследовать API, отправляя запросы на различные адреса (endpoints) и анализируя ответы сервера. Можно изучать структуру данных, коды состояния, заголовки и поведение API при различных входных данных.

**Автоматизация тестирования:**

1. **Какие инструменты для автоматизации тестирования вы знаете?**

Инструменты для автоматизации тестирования:

* Playwright
* Cypress
* pytest
* Selenium
* Postman

1. **Что такое Selenium? Как с его помощью автоматизируют тестирование?**

Selenium – это инструментарий для автоматизации взаимодействия с веб-браузерами. Он позволяет имитировать действия пользователя в браузере (клик, ввод текста, выбор элементов) и проверять результаты.

С помощью Selenium можно автоматизировать проверку функциональности, пользовательского интерфейса и регрессионное тестирование веб-приложений.

1. **В чем разница между unit-тестированием и интеграционным тестированием?**

Unit-тестирование = модульное тестирование – это проверка отдельных изолированных компонентов кода (функций, классов).

Интеграционное тестирование – тестирование взаимодействия между компонентами системы, между несколькими модулями.

1. **Каковы преимущества и недостатки автоматизированного тестирования?**

Преимущества:

* Точность
* Воспроизводимость
* Скорость регрессионного тестирования

Недостатки:

* Изначально код сложнее покрыть автотестами
* Адаптация тестов при изменении в UI
* Сложность/невозможность автоматизации usability-тестирования

1. **Какие языки программирования чаще всего используются в тестировании?**

Сейчас в основном в тестировании применяются:

* JavaScript
* Python
* Java

1. **Как выбрать, какие тесты автоматизировать?**

При выборе какие тесты автоматизировать следует задастся для начала вопросами: «Насколько часто выполняется этот тест?», «Насколько часто этот тест изменяется?», «Сколько этот тест сложен в воспроизведении в ручном тестировании?». После этого можно определить какие тесты лучше автоматизировать, чтобы облегчить отладку ОП, а какие тесты либо имеют низкий приоритет автоматизации, либо в целом не нуждаются в этом, так как их реализация/поддержка будет неоправданно сложной/дорогой.