### **Задание 1. Изучение ресурса ISTQB**

1. **Исследуй сайт ISTQB, изучи функциональность и возможности данного ресурса.**
2. **Найди и выпиши назначение разделов «Certifications», «SCR» и «Glossary».**
3. **Также с помощью данного ресурса выпиши определение понятия «отладка». Понимание этого термина необходимо для эффективного взаимодействия с разработчиками и правильной классификации и устранения дефектов.**

#### 1. **Certifications**

Раздел «Certifications» предоставляет информацию о различных уровнях и направлениях сертификации ISTQB. Здесь можно ознакомиться с требованиями к экзаменам, учебными программами и доступными сертификациями, такими как Foundation Level, Advanced Level и Specialist. Этот раздел полезен как для начинающих, так и для опытных тестировщиков, а также для работодателей, стремящихся развивать навыки своих команд. ​

#### 2. **SCR (Successful Candidate Register)**

SCR — это официальный глобальный реестр сертифицированных специалистов ISTQB. Он позволяет проверить подлинность сертификатов, введя имя и фамилию кандидата. Это особенно полезно для работодателей и рекрутеров при проверке квалификации кандидатов. ​

#### 3. **Glossary**

Глоссарий ISTQB содержит стандартизированные определения терминов, используемых в тестировании программного обеспечения. Он служит справочным ресурсом для специалистов, обеспечивая единое понимание терминологии в индустрии. ​

Согласно глоссарию ISTQB, отладка (debugging) — это процесс поиска, анализа и устранения причин отказов в программном обеспечении.

**Задание 2. Виды тестирования**

1. **В этом задании тебе нужно найти как можно больше информации про различные виды тестирования и дать краткое описание каждому виду.**

**Классификация по уровню:**

* **Модульное (unit testing)** — тестирование отдельных функций или методов программы. Проводится разработчиками. Цель — убедиться, что конкретный модуль работает правильно.
* **Интеграционное (integration testing)** — проверка взаимодействия между модулями. Например, работает ли связка "база данных + API + фронт".
* **Системное (system testing)** — тестирование всей системы целиком. Проверяется, соответствует ли приложение требованиям.
* **Приемочное (acceptance testing)** — проводится заказчиком или QA перед выпуском продукта. Проверяется, решает ли программа задачи пользователя.

**По степени автоматизации:**

* **Ручное тестирование** — тестировщик сам проходит сценарии без использования автоматических скриптов.
* **Автоматизированное тестирование** — выполняется с помощью программных инструментов (например, Selenium, JUnit).

**По доступности коду:**

* **White-box testing (тестирование "белого ящика")** — тестировщик знает внутреннюю структуру кода и использует это для построения тестов.
* **Black-box testing (тестирование "черного ящика")** — проверяется функциональность без знания кода. Тестируется только внешний интерфейс.
* **Gray-box testing** — частичное знание внутренней структуры. Сочетание двух предыдущих.

**По цели:**

* **Функциональное тестирование** — проверка, как система выполняет функции, заявленные в требованиях.
* **Нефункциональное тестирование** — проверяются такие параметры, как производительность, безопасность, удобство использования.

**По подходу:**

* **Позитивное тестирование** — проверка, что система правильно работает при корректном вводе.
* **Негативное тестирование** — тестируется поведение при ошибочном, некорректном или пустом вводе.

**По степени изоляции:**

* **Изолированное тестирование** — отдельный модуль, без внешних зависимостей.
* **Интегрированное тестирование** — модули тестируются совместно.

### **Задание 3. Жизненный цикл разработки ПО**

1. Изучи следующие методологии разработки ПО (особенно важно понять достоинства и недостатки каждой методологии):
   * Waterfall (водопадная модель);
   * V-model;
   * Incremental Model;
   * Scrum;
   * Kanban.
2. Выпиши плюсы и минусы каждой из перечисленных выше методологий. Формат выполнения — произвольный. Единственное условие: понятность описанного не только тебе, но и любому, кто будет это читать.
3. Изучи, что такое «Agile-манифест» (например, тут <https://agilemanifesto.org/iso/ru/manifesto.html>) и выпиши 4 основные идеи этого манифеста с кратким пояснением, как ты его понял. Обязательно обрати внимание на фразу «То есть **не отрицая важности того, что справа**, мы всё-таки больше ценим то, что слева». Дай краткое описание и этой фразе.

#### **1. Waterfall (Водопадная модель)**

**Плюсы:** ✅ Четкая структура и последовательность этапов. ✅ Легко управлять. ✅ Хорошо подходит для проектов с неизменными требованиями. **Минусы:** ❌ Негибкая – изменения сложно вносить. ❌ Тестирование только в конце. ❌ Клиент видит результат поздно.

#### **2. V-model (V-модель)**

**Плюсы:** ✅ Тестирование параллельно разработке. ✅ Раннее выявление ошибок. ✅ Подходит для критических систем. **Минусы:** ❌ Негибкая. ❌ Дорогая и долгая. ❌ Не для меняющихся требований.

#### **3. Incremental Model (Инкрементная модель)**

**Плюсы:** ✅ Постепенная разработка. ✅ Клиент получает части продукта раньше. ✅ Гибче, чем Waterfall. **Минусы:** ❌ Нужно тщательное планирование. ❌ Риск ухудшения архитектуры. ❌ Менее гибкая, чем Agile.

#### **4. Scrum**

**Плюсы:** ✅ Гибкость и адаптивность. ✅ Короткие итерации (спринты). ✅ Улучшает командную работу. **Минусы:** ❌ Требует дисциплины. ❌ Хаотичность без Scrum-мастера. ❌ Не для жестких сроков.

#### **5. Kanban**

**Плюсы:** ✅ Гибкость без жестких итераций. ✅ Визуализация задач. ✅ Меньше бюрократии. **Минусы:** ❌ Риск перегрузки команды. ❌ Менее структурирован. ❌ Требует самодисциплины.

### **Agile-манифест: 4 идеи**

1. **Люди и взаимодействие важнее процессов.**
2. **Рабочий продукт важнее документации.**
3. **Сотрудничество с клиентом важнее контракта.**
4. **Готовность к изменениям важнее плана.**

**Фраза про "важность того, что слева"** означает, что Agile ценит гибкость и людей, но не отменяет полностью процессы и документацию.

**Вывод:** Waterfall и V-model – для стабильных проектов, Scrum и Kanban – для гибких, Incremental – промежуточный вариант. Agile делает ставку на адаптивность.

**Задание 4. Таблица: Уровни тестирования**

| **Уровень тестирования** | **Кем проводится** | **На каком этапе** | **Процесс проведения** |
| --- | --- | --- | --- |
| **Модульное** | Разработчиками | Во время написания кода | Написание unit-тестов, проверка логики функций |
| **Интеграционное** | Разработчиками/тестировщиками | После модульного тестирования | Проверка связей между модулями |
| **Системное** | Тестировщиками (QA) | После сборки всей системы | Тестируется вся программа целиком |
| **Приемочное** | Заказчиком/конечным пользователем | Перед релизом | Проверка, соответствует ли ПО ожиданиям клиента |

**Задание 5. 7 принципов тестирования ПО**

1. **Тестирование показывает наличие ошибок, но не доказывает их отсутствие**  
   → Даже если тесты пройдены, это не гарантирует полное отсутствие багов.
2. **Полное тестирование невозможно**  
   → Нельзя протестировать все возможные сценарии, особенно в сложных системах.
3. **Раннее тестирование экономит ресурсы**  
   → Чем раньше найдена ошибка, тем дешевле и быстрее её исправить.
4. **Скопление дефектов**  
   → Обычно большая часть багов концентрируется в нескольких модулях.
5. **Парадокс пестицида**  
   → Одни и те же тесты теряют эффективность. Тесты нужно регулярно обновлять.
6. **Тестирование зависит от контекста**  
   → Тестирование интернет-магазина и медсистемы будет разным. Всё зависит от целей.
7. **Отсутствие ошибок — не признак качества**  
   → ПО может не иметь багов, но не выполнять нужную функцию или быть неудобным.