Технологии и инструменты для автоматизации тестирования

1. Написание автотестов на **Java** в среде разработки **IntelliJ IDEA**.
2. Разработка UI и API автотестов с использованием фреймворков **Selenide** и **REST Assured**.
3. Использование библиотек **Lombok** и **Jackson** для формирования body в POST-запросе и для десериализации ответа, **AssertJ** для проверки ответа, а также для валидации JSON Schema.
4. Создание моделей и спецификаций для запросов и ответов. Работа с Allure Listener и кастомным шаблоном для информативного отчета.
5. Использование **JUnit 5** для написания параметризованных тестов с аннотациями @ValueSource, @CsvSource, @EnumSource и @MethodSource. Применение аннотаций @BeforeAll, @BeforeEach, @AfterAll, @AfterEach, @Test для запуска тестов, @DisplayName для отображения названий тестов, @Disabled для отключения тестов, @Tag/@Tags для отображения дополнительных данных в отчете, а также для запуска тестов по конкретным тегам.
6. Применение **JUnit 5** для проверки ожидаемого результата с использованием Assertions.
7. Написание автотестов с использованием лямбда шагов через Step (name, () -> {}) и аннотации @Step.
8. Автоматизированная сборка проекта при помощи **Gradle**. Подключение всех необходимых библиотек и зависимостей для запуска проекта, а также написание тасков для запуска тестов по тегам.
9. Создание Jobs в **Jenkins** для запуска тестов с различными параметрами.
10. Удаленный запуск тестов в браузерах через **Selenoid** в контейнерах **Docker**.
11. Использование **Allure Report** для формирования отчетов о прохождении тестов с прикреплением к тестам видео, скриншотов, PageSource и логов консоли браузера.
12. Интеграция автотестов с тест-менеджмент системой **Allure TestOps** и таск-трекер системой Jira.
13. Создание бота в Telegram для автоматической отправки уведомлений с отчетом о прохождении тестов.
14. Настройка отправки уведомлений о пройденном прогоне в Telegram с помощью фреймворка **allure-notifications**.
15. Применение паттернов Page Object (степовой и сценарный) и Page Element.
16. Использование библиотеки **JavaFaker** для генерации рандомных данных.
17. Работа (скачивание, загрузка, парсинг, проверка контента) с файлами разного формата (txt, rtf, pdf, xls/xlsx, doc/docx, zip) при помощи библиотек **Commons IO,** **PDF Test**, **XLS Test** и **OpenCSV**.
18. Применение библиотеки **Owner** для скрытия приватных данных.