**МИНОБРНАУКИ РОССИИ** **САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)** **Кафедра МО ЭВМ**

# ОТЧЕТ по «UI-тестирование» практике

# Тема: Тестирование системы Reddit

|  |  |  |  |
| --- | --- | --- | --- |
| Студенты гр. 3343 |  |  | Никишин С.А. Иванов П.Д. Силяев Р. |
| Руководитель |  |  | Шевелёва А.М. |

Санкт-Петербург

2025

**ЗАДАНИЕ** **НА «UI-ТЕСТИРОВАНИЕ» ПРАКТИКУ**

Технологии: Java, Selenide (Selenium), Junit, Maven, логирование. Чеклист: подробное описание созданных тестов в одном файле: название системы, которую тестируете, название тестов, входные данные (по возможности), описание теста – то, как это бы делал пользователь.

Тесты: нужно написать 10 тестов по одному определенному блоку / функционалу системы.

Примечание: была выбрана система reddit.com

Сроки прохождения практики: 25.06.2025 – 08.07.2025

Дата сдачи отчета: 07.07.2025

Дата защиты отчета: 07.07.2025

|  |  |  |
| --- | --- | --- |
| Студенты |  | Никишин С.А. Иванов П.Д. Силяев Р. |
| Руководитель |  | Шевелёва А.М. |

# АННОТАЦИЯ

В ходе практики проведено тестирование ключевых функций социальной сети Reddit, включая авторизацию, создание публикаций, добавление комментариев, лайки и дизлайки. Особое внимание уделено взаимодействию с публикациями, поиску по ключевым словам и отправке, а также навигации по разделам платформы. Дополнительно тестируются подписки, взаимодействие с контентом, корректность отображения интерфейс. Результаты подтверждают соответствие платформы заявленным требованиям и выявляют возможные ошибки в ключевых сценариях работы.

# SUMMARY

During the internship, key functions of the Reddit social network were tested, including authorization, creating publications, adding comments, likes and dislikes. Particular attention was paid to interaction with publications, searching by keywords and sending, as well as navigation through the platform sections. Subscriptions, interaction with content, and correctness of the interface display were additionally tested. The results confirm the platform's compliance with the stated requirements and identify possible errors in key work scenarios.

# СОДЕРЖАНИЕ

[АННОТАЦИЯ 3](#_Toc202651596)

[СОДЕРЖАНИЕ 4](#_Toc202651598)

[ОПИСАНИЕ КЛАССОВ И МЕТОДОВ 9](#_Toc202651599)

[ЗАКЛЮЧЕНИЕ 20](#_Toc202651600)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 21](#_Toc202651601)

**ВВЕДЕНИЕ**

**Цель:** Проведение комплексного тестирования функционала социальной сети Reddit, включая проверку основных пользовательских сценариев взаимодействия с контентом и профилем.

**Задачи:** Тестирование авторизации:

* Проверка позитивной авторизации.
* Проверка негативной авторизации.
* Выход из аккаунта.

Тестирование взаимодействия с лентой:

* Создание текстового поста.
* Сохранение публикации к себе.
* Добавление комментария к посту.
* Скрытие поста из ленты.
* Лайк/дизлайк у поста.
* Отправление жалобы на пост.
* Поиск поста по названию
* Подписка на группу

Тестирование взаимодействия с панелью навигации:

* Переключение на различные разделы через панель навигации.

**Методология:** Автоматизированное тестирование (JUnit, Selenium), ручное тестирование критических сценариев.

**ОПИСАНИЕ РЕАЛИЗУЕМЫХ ТЕСТОВ**

Тестируемая система [-](http://rutube.ru/)  https://www.reddit.com/

Для тестирования было создано 2 аккаунта: AdSafe1407 и No-Customer3367.

****Группа 1: Авторизация и Выход****

* ****ТС1.1 Позитивная авторизация:**** Проверяет успешный вход. Нажимается кнопка "Log In", вводятся корректные логин и пароль (Аккаунт 1), нажимается "Войти". Ожидается успешная авторизация и отображение ленты постов.
* ****ТС1.2 Негативная авторизация:**** Проверяет реакцию на неверные данные. Нажимается "Log In", вводятся логин (Аккаунт 1) и неверный пароль (Аккаунт 2), нажимается "Войти". Ожидается сообщение об ошибке авторизации, пользователь не должен войти в систему.
* ****ТС1.3 Выход из системы:**** Проверяет корректный выход. Пользователь авторизуется, открывает меню профиля и нажимает "Выйти". Ожидается, что пользователь будет разлогинен.

****Группа 2: Работа с Публикациями****

* ****ТС2.1 Создание текстовой публикации:**** Проверяет создание поста. Пользователь авторизуется (Аккаунт 1), нажимает «Создать пост», выбирает текущий профиль как сообщество, вводит случайный заголовок и текст, нажимает «Опубликовать». Ожидается, что созданный пост появится в профиле пользователя.
* ****ТС3.1 Сохранение публикации к себе:**** Проверяет функцию сохранения поста. Авторизованный пользователь (Аккаунт 1) открывает настройки поста и нажимает «Сохранить». Ожидается, что пост добавится в раздел «Сохранено» профиля пользователя.
* ****ТС4.1 Добавление комментария:**** Проверяет возможность комментирования. Авторизованный пользователь (Аккаунт 1) открывает первую публикацию в ленте, вводит текст в поле «Начать беседу» и нажимает «Комментарий». Ожидается появление введенного комментария под постом.
* ****ТС5.1 Скрытие поста из ленты:**** Проверяет функцию скрытия поста. Авторизованный пользователь (Аккаунт 1) открывает настройки поста и нажимает «Скрыть». Ожидается, что публикация исчезнет из ленты пользователя.
* ****ТС6.1 Лайк поста:**** Проверяет постановку лайка. Авторизованный пользователь (Аккаунт 1) открывает пост в ленте и нажимает кнопку «↑» под ним. Ожидается подсветка кнопки и увеличение счетчика лайков.
* ****ТС6.2 Дизлайк поста:**** Проверяет постановку дизлайка. Авторизованный пользователь (Аккаунт 1) открывает пост в ленте и нажимает кнопку «↓» под ним. Ожидается подсветка кнопки и уменьшение счетчика лайков (или увеличение счетчика дизлайков, в зависимости от реализации).

****Группа 3: Поиск и Модерация****

* ****ТС7.1 Поиск по названию:**** Проверяет работу поиска. Авторизованный пользователь (Аккаунт 1) вводит ключевое слово «download» в строку поиска, нажимает Enter и открывает первый пост в результатах. Ожидается, что заголовок открытой публикации содержит слово «download».
* ****ТС8.1 Отправка жалобы на пост:**** Проверяет процесс отправки жалобы. Авторизованный пользователь (Аккаунт 1) открывает настройки поста, нажимает «Пожаловаться», выбирает категорию «Спам», затем конкретную причину «Размещение чрезмерного количества постов или комментариев в сообществе» и нажимает «Отправить». Ожидается получение сообщения об успешной отправке жалобы.

****Группа 4: Сообщества****

* ****ТС9.1 Подписка на группу:**** Проверяет вступление в сообщество. Авторизованный пользователь (Аккаунт 1) переходит в раздел «Исследовать» и нажимает «Вступить» на первой группе. Ожидается, что текст кнопки изменится на «В сообществе».

****Группа 5: Навигация по Разделам****

* ****ТС10.1 Раздел «Домашняя страница»:**** Проверяет переход в домашнюю ленту. Авторизованный пользователь (Аккаунт 1) нажимает «Домашняя страница». Ожидается загрузка ленты постов домашней страницы.
* ****ТС10.2 Раздел «Популярное»:**** Проверяет переход в раздел популярного. Авторизованный пользователь (Аккаунт 1) нажимает «Популярное». Ожидается загрузка популярных постов.
* ****ТС10.3 Раздел «Все»:**** Проверяет переход в общую ленту. Авторизованный пользователь (Аккаунт 1) нажимает «Все». Ожидается загрузка постов из всех сообществ.

# ОПИСАНИЕ КЛАССОВ И МЕТОДОВ

1. **Базовые элементы**

**1.1.** *BaseElement* - Абстрактный базовый класс для всех UI-элементов страницы. Обеспечивает универсальную обертку над WebElement, предоставляя безопасные и повторно используемые методы взаимодействия с элементами DOM.

Конструкторы:

*BaseElement(WebDriver driver, By locator)*-Позволяет отложенно инициализировать элемент по заданному локатору. Это удобно, если элемент может быть переинициализирован или его нужно искать каждый раз заново (динамические страницы).

*BaseElement(WebDriver driver, WebElement element)-*Инициализация на основе уже найденного DOM-элемента. Подходит для вложенных структур или если WebElement передается из других методов.

Методы:

*WebElement getElement()*-Возвращает WebElement, привязанный к этому объекту. При инициализации через By будет выполнено явное ожидание presenceOfElementLocated, что гарантирует наличие элемента в DOM.

*boolean isVisible()*-Проверяет, видим ли элемент для пользователя. Метод оборачивает возможные исключения (NoSuchElementException, TimeoutException), возвращая false, если элемент не найден или не отображается.

*String getText()-*Возвращает видимый текст из элемента. Полезно для валидации содержимого кнопок, меток, заголовков и т.п.

**1.2.** *ClickableElement* - наследник *BaseElement*, добавляющий поведение для кликабельных элементов. Используется для описания кнопок, ссылок и других интерактивных компонентов, по которым может быть выполнен клик.

Конструктор:

*ClickableElement(WebDriver driver, WebElement element)*-Конструктор, наследующий поведение от BaseElement. Используется, когда клик по элементу необходим.

Методы:

*void click()*-Выполняет обычный клик по элементу с предварительным ожиданием его кликабельности (ExpectedConditions.elementToBeClickable). Это защищает от попыток клика по еще не готовому (заблокированному, невидимому) элементу.

*void jsClick()-*Выполняет клик через JavaScript (arguments[0].click();). Полезно, если стандартный клик перехватывается другими элементами или не работает из-за нестандартной реализации UI.

**1.3.** *BasePage* - Абстрактный класс, представляющий любую страницу приложения. Реализует типичные действия и ожидания, которые повторяются на разных страницах, такие как обновление страницы, ожидание загрузки элементов и прокрутка.

Конструктор:

*BasePage(WebDriver driver)-*Инициализирует драйвер и стандартное ожидание (WebDriverWait), используемое в методах ожидания.

Методы:

*void refresh()-*Выполняет полную перезагрузку текущей страницы. Используется в тестах, где нужно сбросить состояние.

*boolean isTitleContains(String partialTitle)-*Ждет, пока заголовок страницы (<title>) не будет содержать указанный текст. Полезен для проверки перехода на нужную страницу.

*WebElement waitForVisible(By locator)-*Ожидает, пока элемент по заданному локатору станет видимым. Повышает стабильность тестов на медленно загружающихся страницах.

*List<WebElement> waitForAllVisible(By locator)*-Ожидает, пока все элементы, найденные по локатору, станут видимыми. Применяется при работе с таблицами, списками, каруселями и т.д.

*WebElement waitForClickable(By locator)-*Явное ожидание кликабельности элемента. Используется в сценариях, где важна точность действий пользователя.

*String getCurrentUrl()-*Возвращает текущий URL, загруженный в браузере. Применяется для валидации маршрутов/редиректов.

*void scrollToCenter(WebElement element)*-Скроллит указанный элемент в центр экрана, чтобы обеспечить его видимость. Полезно при работе с элементами вне viewport.

**1.4.** *BaseTest* - Абстрактный базовый класс для всех JUnit-тестов. Содержит общую логику инициализации и завершения работы браузера. Используется для устранения дублирования в тестах.

Поля:

*WebDriver driver* – экземпляр драйвера для взаимодействия с браузером.

BASE\_URL – URL начальной страницы для тестов (в данном случае, форма входа Reddit).

IMPLICIT\_WAIT – глобальный таймаут неявного ожидания.

Методы:

*void openBrowser()* (аннотирован @Before)-Запускается перед каждым тестом. Инициализирует ChromeDriver, настраивает неявное ожидание и открывает базовую страницу.

*void closeBrowser()* (аннотирован @After)-Выполняется после каждого теста. Корректно закрывает браузер, освобождая ресурсы.

*void openPage(String url)*-Загружает любую страницу по указанному URL. Может использоваться в дочерних классах для навигации на разные страницы сайта.

1. **Классы элементов**

**2.1.** *Button* - Класс-обертка над HTML-кнопками, предоставляющий типичный интерфейс взаимодействия с кнопками, в том числе в Shadow DOM.

Конструкторы и фабричные методы:

*protected Button(WebDriver driver, WebElement element)*-Инициализирует кнопку на основе уже найденного WebElement. Используется внутри фабричных методов.

*static Button fromLocator(WebDriver driver, By by)-*Ищет элемент кнопки по заданному локатору By и возвращает обертку Button.

*static Button fromShadowHost(WebDriver driver, By hostLocator, By shadowLocator)*-Ищет кнопку, находящуюся внутри Shadow DOM. Сначала ищет элемент-хост, затем кнопку внутри него.

*public String getAttribute(String name)* - Возвращает значение указанного атрибута элемента (например, "*href*", "*class*", "*id*"). Если атрибут отсутствует - возвращает *null*.

*public String getText()* - Возвращает видимый текст элемента, включая пробелы и переносы строк. Учитывает стили отображения (не возвращает текст скрытых элементов).

**2.2.** *Comment* - Объект для взаимодействия с пользовательскими комментариями на странице Reddit. Представляет обертку над элементом shreddit-comment.

Поля:

COMMENT\_TEXT – локатор, по которому ищется содержимое текста комментария.

Конструктор:

*Comment(WebDriver driver, WebElement element)*-Создает объект комментария на основе уже найденного DOM-элемента.

Методы:

*static Comment fromLocator(WebDriver driver, By by)*-Позволяет создать объект комментария по CSS или XPath-локатору.

*String getAuthor()*-Извлекает имя автора из атрибута author.

*String getText()*-Переопределенный метод: возвращает только текст из вложенного элемента div[id$='-post-rtjson-content'] p, а не всего комментария целиком.

**2.3.** *Input* - Представляет поле ввода, включая <input>, <textarea> и другие формы текстового взаимодействия. Поддерживает стандартные поля и элементы в Shadow DOM.

Конструктор:

*protected Input(WebDriver driver, WebElement element)*-Инициализация поля ввода на основе уже найденного DOM-элемента.

Методы:

*static Input fromLocator(WebDriver driver, By by)-*Создание поля ввода по любому локатору (id, xpath, cssSelector и т.д.).

*static Input byClass(WebDriver driver, String className)*-Упрощённый способ получения поля ввода по CSS-классу.

*static Input fromShadowHost(WebDriver driver, By hostLocator, By shadowLocator)*-Получение поля ввода из Shadow DOM (пример: hostLocator указывает на компонент, shadowLocator – на input внутри него).

*void sendKeys(String text)*-Симулирует ввод текста в поле. Текст добавляется к уже существующему содержимому.

**2.4.** *Link* - Обертка над HTML-ссылками (<a>), позволяющая удобно кликать и проверять назначение ссылок.

Конструктор:

*Link(WebDriver driver, WebElement element)*-Инициализация на основе найденного элемента-ссылки.

Методы:

*static Link fromLocator(WebDriver driver, By by)-*Получение ссылки по любому локатору.

*static Link byText(WebDriver driver, String linkText)*-Поиск по видимому тексту ссылки (<a>), аналог By.linkText().

*static Link fromShadowHost(WebDriver driver, By hostLocator, By shadowLocator)*-Доступ к ссылке внутри Shadow DOM.

*String getHref()-*Возвращает значение атрибута href, на который указывает ссылка.

*boolean pointsTo(String url)-*Сравнивает значение href со строкой URL (без учета регистра).

1. **Классы страниц**

**3.1.** *CreatePostPage* - Представляет страницу создания нового поста. Реализует взаимодействие с полями ввода заголовка и текста, выбором профиля (через Shadow DOM), и кнопкой отправки поста.

Методы:

*clickTitle()-*Кликает по полю заголовка поста. Используется для фокусировки на поле перед вводом текста.

*enterPostTitleText()-*Вводит фиксированный заголовок поста ("test title"). Используется для создания тестовых постов без параметризации.

*clickBody()-*Кликает по полю тела поста. Применяется для активации текстового редактора перед вводом.

*enterPostBodyText()-*Вводит стандартный текст тела поста ("test text"). Применяется в простых сценариях создания поста.

*clickComunityPickerMenu()*-Кликает по меню выбора профиля/коммьюнити. Меню открывается для выбора от имени какого профиля публиковать пост.

*enterUsernameText(String username)-*Вводит имя пользователя (в формате u/username) в поле поиска профилей внутри открытого меню. Работает через Shadow DOM.

*clickSelectProfile(String profileName)-*Находит в списке предложений и выбирает профиль с заданным именем. Использует Shadow DOM и ожидание загрузки списка.

*clickSubmitPostButton()-*Кликает по кнопке публикации поста. Использует JavaScript-клик внутри Shadow DOM.

**3.2.** *FeedPage* - Представляет главную ленту (feed) с постами. Реализует действия над постами (скрытие, жалоба), навигацию по категориям и переход на страницу создания поста.

Методы:

*openFirstPost()-*Переходит на страницу первого поста из ленты. Кликает по ссылке полного перехода (full-post-link).

*clickPostOverflowMenu()-*Открывает выпадающее меню поста (три точки). Используется для скрытия или жалобы на пост.

*clickPostOverflowHide()-*Скрывает первый пост в ленте. Запоминает его permalink для последующей проверки скрытия.

*clickUserMenuButton()*- Переходит в профиль пользователя (кликает на иконку аккаунта справа сверху)

*clickQuitButton()* - Нажимает кнопку выхода из аккаунта, завершая текущую сессию пользователя.

*FindLoginButton()* - Осуществляет поиск кнопки входа в систему ("Log In" )и, возможно, кликает по ней, если это предусмотрено логикой функции.

*clickPostOverflowSaveButton()* - Кликает по кнопке "Сохранить" в контекстном меню ("⋯") публикации.

*isPostHidden()-*Проверяет, что ранее скрытый пост больше не отображается в ленте. Сравнивает permalink и наличие атрибута hidden.

*clickPostOverflowReport()-*Открывает форму жалобы на пост. Взаимодействует с Shadow DOM.

*clickReportSPAMButton()-*Выбирает причину жалобы как "СПАМ".

*clickNextReportButton()-*Переходит к следующему шагу формы жалобы.

*clickCategoryOfSPAM()*-Выбирает подкатегорию спама (например, "SPAM\_COMMENT\_FLOODING").

*clickSendReportButton()-*Отправляет жалобу. Использует JavaScript-клик по кнопке отправки.

*isReportSend()-*Проверяет, что жалоба успешно отправлена. Определяет по наличию подтверждающего элемента.

*expandSideBar()-*Раскрывает боковую панель навигации, если она скрыта. Используется перед переходом между категориями.

*clickCategoryButton(String nameCategory)*-Кликает по заданной категории в боковой панели (например, "home", "popular", "all").

*checkActivePage(String namePage)-*Проверяет, что текущая страница соответствует выбранной категории.

*ClickCreatePost()* *-* Переходит на страницу создания нового поста. Кликает по кнопке в боковой панели.

*searchFor(String keyword)* - Выполняет поиск по ключевому слову. Вводит текст в поле поиска и запускает поиск.

*getFirstSearchResultTitle()* - Возвращает заголовок первого результата в выдаче поиска.

*findSearchInput()* - Находит поле поиска через JavaScript (внутренний метод).

*waitForPageLoad()* - Ожидает полной загрузки страницы (внутренний метод).

**3.3.** *LoginPage* - Представляет страницу авторизации. Обеспечивает ввод логина, пароля и вход в систему.

Методы:

*enterUsername(String username)*-Вводит логин пользователя. Использует Shadow DOM.

*enterPassword(String password)-*Вводит пароль пользователя. Также работает с Shadow DOM.

*clickLoginButton()*-Кликает по кнопке входа. После клика ожидает загрузку (через Thread.sleep(6000)) и обновляет страницу.

**3.4.** *PostPage* - Страница отдельного поста. Предоставляет методы для комментирования и проверки наличия комментариев.

Методы:

*clickAddCommentButton()-*Кликает по кнопке добавления комментария. Прокручивает страницу до кнопки и кликает по ней.

*enterCommentText(String text)*-Вводит текст в поле комментария. Используется после открытия редактора комментариев.

*clickSubmitCommentButton()*-Отправляет комментарий. Использует JavaScript-клик для большей стабильности.

*isCommentVisible(String username, String expectedText)-*Проверяет, что комментарий от указанного пользователя с нужным текстом отображается под постом.

*upvotePost()* - Ставит лайк посту. Автоматически ожидает подтверждения действия.

*downvotePost()* - Ставит дизлайк посту. Автоматически ожидает подтверждения действия.

*isUpvoted()* - Проверяет, активен ли лайк. Возвращает true, если лайк поставлен.

*isDownvoted()* - Проверяет, активен ли дизлайк. Возвращает true, если дизлайк поставлен.

*resetVote()* - Сбрасывает текущий голос (если он был поставлен).

*getVoteCount()* - Возвращает текущее количество голосов в числовом формате.

**3.5.** *CommunityPage -* Страница сообщества (сабреддита). Предоставляет методы для управления подпиской на сообщество.

Методы:

*clickSubscribeButton() -* Кликает по кнопке подписки/отписки. Автоматически находит элемент через Shadow DOM.

*getSubscribeButtonText() -* Возвращает текущий текст кнопки ("Присоединиться" или "В сообществе").

*isSubscribed() -* Проверяет состояние подписки. Возвращает true, если пользователь подписан.

*waitForSubscribeButton() -* Ожидает появления кнопки подписки (таймаут 25 секунд).

1. **Классы тестов**

**4.1.** *AllTests* — Класс с набором автотестов для проверки ключевой функциональности сайта Reddit.

Выполняет действия через Page Object-классы и проверяет сценарии: авторизация, добавление комментария, отправка жалобы, навигация по категориям, скрытие постов.

Поля:

TEST\_USERNAME — Логин тестового пользователя.

TEST\_PASSWORD — Пароль тестового пользователя.

TEST\_USERNAME\_2 — Неправильный логин от аккаунта

TEST\_PASSWORD\_2 — Неправильный пароль от аккаунта

COMMENT\_TEXT — Текст, используемый при добавлении комментария.

COMMENT\_AUTHOR — Ожидаемый автор комментария (совпадает с логином).

Методы:

*authorize(LoginPage loginPage)* — Пытается авторизовать пользователя через cookies, если не получилось — выполняет ручной вход с логином и паролем. Сохраняет cookies после успешной авторизации. Использует паузы для обхода антибот-защиты.

*testSuccessfulLogin()* — Тестирует успешный вход в аккаунт.

* Нажимает на кнопку для авторизации.
* Вводит корректные данные пользователя.
* Нажимается кнопка войти.
* Проверяет, что перешли на страницу с лентой новостей (URL страницы с лентой)

*testUnsuccessfulLogin()* – Тестирует негативную авторизацию.

* Нажимает на кнопку для авторизации.
* Вводит некорректные данные пользователя.
* Нажимает на кнопку входа.
* Проверяет, что остались на странице с авторизацией (URL страницы для авторизации)

*testSuccessfulLogout()* – Тестирует выход из аккаунта.

* Нажимает на кнопку для авторизации.
* Вводит корректные данные пользователя.
* Нажимает на кнопку входа.
* Нажимает на иконку профиля
* Нажимается Выйти.
* Проверяет, что появилась кнопка со входом в аккаунт.

*testCreatePost()* – Тестирует создание поста в профиле.

* Выполняет авторизацию.
* Переходит в меню профиля.
* Открывает редактор поста, вводит текст и отправляет.
* Проверяет, что произойдёт переход на страницу профиля.

*testSavePost()* – Тестирует сохранение поста в профиле.

* Выполняет авторизацию.
* Переходит к первому посту.
* Открывает меню выбора дейтсвий («…»).
* Нажимает на кнопку сохранить
* Проверяет, что пользователь остался в контекстном меню.

*testAddComment()* — Тестирует добавление комментария под первым постом.

* Выполняет авторизацию.
* Переходит к первому посту.
* Открывает редактор комментариев, вводит текст и отправляет.
* Проверяет, что комментарий отображается и принадлежит нужному пользователю.

*testSendReport()* — Тестирует отправку жалобы на пост.

* Авторизуется.
* Открывает меню поста и выбирает опцию "Пожаловаться".
* Выбирает категорию жалобы (СПАМ).
* Проходит шаги мастера отправки жалобы.
* Проверяет, что жалоба успешно отправлена.

*testNavigationBarHome()* — Проверяет переход по категории "home" в боковом меню.

* Выполняет авторизацию.
* Раскрывает боковое меню.
* Кликает по кнопке "home".
* Проверяет, что активна нужная страница.

*testNavigationBarPopular()* — Проверяет переход в раздел "popular" через боковое меню.

* Авторизация.
* Раскрытие бокового меню.
* Клик по кнопке "popular".
* Проверка текущей страницы.

*testNavigationBarAll()* — Проверяет переход на страницу "all" через боковое меню.

* Авторизация.
* Раскрытие бокового меню.
* Клик по кнопке "all".
* Проверка соответствия активной страницы.

*testHidePost()* — Проверяет возможность скрыть пост.

* Авторизуется.
* Открывает меню поста и выбирает "Скрыть".
* Ожидает скрытия поста.
* Проверяет, что пост больше не отображается.

*testUpvoteFunctionality()* — Тестирует функциональность лайка поста.

* Авторизуется.
* Открывает первый пост в ленте.
* Сбрасывает текущий голос (если есть).
* Получает начальное количество голосов.
* Ставит лайк посту.
* Проверяет, что состояние лайка активно.
* Проверяет увеличение счетчика голосов на 1.

*testDownvoteFunctionality()* — Тестирует функциональность дизлайка поста.

* Авторизуется.
* Открывает первый пост в ленте.
* Сбрасывает текущий голос (если есть).
* Получает начальное количество голосов.
* Ставит дизлайк посту.
* Проверяет, что состояние дизлайка активно.
* Проверяет уменьшение счетчика голосов на 1.

*testSubscribeToSubreddit()* — Тестирует подписку на сообщество.

* Авторизуется.
* Переходит на страницу сообщества (сабреддита).
* Ожидает появления кнопки подписки.
* Определяет начальное состояние подписки.
* Кликает по кнопке подписки/отписки.
* Ожидает изменения состояния (до 15 секунд).
* Проверяет, что состояние подписки изменилось.

*testSearchPost()* — Тестирует поиск постов.

* Авторизуется.
* Выполняет поиск по ключевому слову "test".
* Получает заголовок первого результата.
* Проверяет, что заголовок содержит слово "test".

# ЗАКЛЮЧЕНИЕ

В ходе тестирования функционала Reddit была проведена комплексная проверка всех ключевых пользовательских сценариев работы с платформой. Основное внимание уделено процессу авторизации, включая успешный вход в систему, обработку ошибок при неверных данных и корректность выхода из аккаунта. Тщательно протестирована работа с публикациями: создание новых постов, а также добавление комментариев к существующим публикациям. Особое значение придавалось проверке системы взаимодействия с контентом, включая механизмы лайков и дизлайков, которые продемонстрировали корректную работу с подсветкой кнопок и изменением счетчиков.

Функционал поиска по ключевым словам показал стабильную работу с выдачей релевантных результатов. Отдельно проверены механизмы отправки жалоб на контент, которые работают в соответствии с ожиданиями. Навигация по основным разделам платформы ("Домашняя страница", "Популярное", "Все") функционирует корректно, обеспечивая быстрый доступ к соответствующему контенту.

Для проведения тестов были специально созданы два тестовых аккаунта с подготовленными данными, что позволило полноценно проверить социальные взаимодействия на платформе. Тестирование осуществлялось через автоматизированную систему с четкой архитектурой, где классы элементов страниц защищены от прямого взаимодействия, а работа с интерфейсом производилась исключительно через методы классов страниц, что обеспечило надежность и стабильность процесса проверки. Все основные функции платформы работают в соответствии с ожиданиями, подтверждая готовность системы к эксплуатации.

В ходе выполнения работы было использовано автоматизированное тестирование (JUnit, Selenium) и ручное тестирование.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Habr Пишем автотест с использованием Selenium Webdriver, Java 8 и паттерна Page Object [Электронный ресурс]/ Статья на Habr URL: <https://habr.com/ru/articles/502292/>
2. YouTube Java Testing with Selenium Course [Электронный ресурс]/ YouTube video URL: <https://www.youtube.com/watch?v=QQliGCtqD2w>