**Test Strategy: Simpals Feature Automation**

1. Introduction & Scope

This document outlines the testing strategy for the core features of the web application hosted at https://999.md/ru/. The primary goal of this test suite is to ensure the reliability and correctness of key user functionalities through automated end-to-end testing.

The scope of this testing effort, as defined in the simpals.feature file, covers the following areas:

* **User Authentication:** Verifying that users can successfully log in and log out.
* **User Notifications:** Ensuring the notification system correctly reflects the user's status.
* **Advertisement Management:** Testing the complete lifecycle of a commercial advertisement, including creation, editing (update), and deletion of ad drafts.

2. Test Strategy & Approach

2.1. Overall Approach

We are employing a **Behavior-Driven Development (BDD)** approach for this project. This methodology allows us to define application behavior in a human-readable language called Gherkin. This improves clarity and ensures that the tests are aligned with business requirements.

The automated tests are designed to simulate real user scenarios from start to finish, validating the integration of different application components.

2.2. Design Pattern

The automation framework is built using the **Page Object Model (POM)** design pattern. This is evident from the project structure, which separates responsibilities:

* pages directory: Contains classes that represent pages of the web application and encapsulate the UI elements and interactions on those pages (e.g., AdPage, MainPage).
* steps directory: Contains the step definition classes that link the Gherkin steps from the feature file to the Java code that executes the actions.
* utils directory: Contains helper classes and utilities for common functions like configuration management.

This separation enhances test maintainability, readability, and reusability.

3. Test Coverage Areas

The current test suite provides coverage for the following specific user flows:

* **Scenario Outline: Check for Notifications**
  + **Given:** The test starts with the browser in a visible (non-headless) mode.
  + **When:** The user navigates to the application's homepage and logs in with specified credentials.
  + **Then:** The system checks the number of notifications for the logged-in user.
  + **And:** The user successfully logs out.
  + **Coverage:** This scenario is data-driven and validates the notification check and logout process for multiple users (brinkc, brinkc2).
* **Scenario: Add commercial ad for Logitech M170**
  + **Given:** The test starts in a non-headless browser and the user is logged in.
  + **When:** The user creates a new advertisement with a specific title ("M170") and price ("120").
  + **Then:** The system verifies that a draft of the "Logitech M170" ad is successfully created and displayed.
  + **And:** The user edits the title of this draft to "Logitech M170 - updated".
  + **Then:** The system verifies that the updated draft is displayed correctly.
  + **And:** The user removes the draft advertisement.
  + **Then:** The system verifies that neither the original nor the updated ad is displayed, confirming successful deletion.
  + **Coverage:** This scenario covers the Create, Read, Update, and Delete (CRUD) operations for advertisement drafts.

4. Tools and Technology Stack

|  |  |  |
| --- | --- | --- |
| **Tool/Technology** | **Purpose** | **Justification** |
| **Java** | Core Programming Language | A robust, object-oriented language with strong community support and a vast ecosystem of libraries for test automation. |
| **Cucumber** | BDD Framework | Enables writing test scenarios in a human-readable Gherkin format, fostering collaboration between technical and non-technical team members. |
| **Selenide** | Browser Automation Library | A powerful wrapper for Selenium that simplifies browser interactions, automatically handles most waits, and leads to more stable and concise UI tests. |
| **Maven** | Build & Dependency Management | A standard tool for managing project dependencies, compiling the code, and running tests in a consistent and reproducible way. |
| **IntelliJ IDEA** | Integrated Development Environment (IDE) | Provides a rich environment for code development, debugging, and running tests. |

5. How to Run the Tests

There are two primary methods for executing the automated tests:

1. **Via Command Line (using Maven):**
   * Open a terminal or command prompt at the root of the project directory.
   * Execute the following command: mvn clean test
   * Maven will handle downloading dependencies, compiling the code, and running all tests defined in the feature files.
2. **Via IntelliJ IDEA IDE:**
   * Open the project in IntelliJ IDEA.
   * Navigate to the simpals.feature file in the project explorer (src/test/resources/features).
   * To run all scenarios in the file, right-click the file and select Run 'Feature: Simpals'.
   * To run a specific scenario, click the green "play" icon next to the scenario name in the editor.

6. Assumptions and Limitations

6.1. Assumptions

* A compatible version of Java Development Kit (JDK) and Apache Maven are installed and configured on the machine running the tests.
* The web application at https://999.md/ru/ is accessible and in a stable, known state.
* The test user accounts (brinkc, brinkc2) are valid and have the necessary permissions to perform the tested actions in the target environment.

6.2. Limitations

* **Execution Mode:** The tests are currently configured to run in a non-headless mode (browser headless is 'false'), meaning a browser window will open during execution. This may require adjustments for running in a CI/CD pipeline.
* **Scope:** The current test suite is not exhaustive and only covers the scenarios defined in the simpals.feature file. Additional features of the application are not yet covered.
* **Cross-Browser Testing:** The current configuration does not explicitly define a cross-browser testing strategy. Tests are likely running against a single default browser (e.g., Chrome).