**BUGGY CARS RATING WEBSITE - TEST APPROACH**

1. **Scope and Overview**

This document provides the Test Approach for Buggy Cars Rating website. The output of this documentation will then provide input to the deliverables related to but not limited to: test planning, test scripting, test scenario creation and test execution.

The following functionalities will be covered in testing this project.

1. Registration of new user
2. Login/Logout of an existing user
3. Editing User Profile
4. Voting for Car Ratings
5. Viewing Overall Car ratings
6. **Test Environment**

The testing environment will be used as sandbox is <https://buggy.justtestit.org/>.

1. **Types of Testing**

*Exploratory Testing*

This is an informal test conducted to explore the website's features with the aim of discovering specific or interesting bugs which can be missed by other testing types.

*User-Based Testing*

As per definition, this type of testing refers to the method used in the design process to evaluate the website with real users. Usability Test for the website can uncover design problems easily and can also promote opportunities to improve the project implementation.

*Performance Testing*

The website is observed to have poor response time when a link or an image is clicked, it is then necessary to conduct Performance Testing – Load Testing especially when there are many users accessing the site at the same time. The Load Testing has a goal to further assess long loading time against the number of concurrent users.

Below are the high-level steps to take for successful execution of Performance (Load) Testing:

1. Create Test Plan.
2. Get baseline from User Requirements.
3. Create test scripts.
4. Create test data.
5. Scale up the environment. Configure total users, duration of the entire run, and the ramp up time.
6. Run the Performance Test.
7. Publish Test Report.
8. **Automation Tools and Strategy**

Given that automating testing is highly encouraged for Agile project such as this, Test Automation Framework will be introduced to have a quick turnaround in terms of catching bugs and retesting them. It will also save time/effort and eliminate error-prone testing especially the critical paths of this project.

Below is the test automation technology stack to be used, provided that these tools and principles below are expertise of testing team.

*Automation Tool* - Selenium WebDriver (Java library)

*Reporting* - Allure

*Design Pattern* - Page Object Model

*Programming Language* - Java

*Test Framework / Runner* - TestNG

*Project Lifecycle* - Maven

1. **Test Schedule**

Software Testing must be involved from the early phase of the development lifecycle, commonly referred to as Shift Left concept. It has the goal of early defect detection hence enhancing quality of the software and minimising the cost when a bug is detected at the later stage of development lifecycle. Testing must begin at the start of Requirements (User Story) Gathering though Static Testing.

Refer to the planned testing timeline below.

|  |  |
| --- | --- |
| **Testing Activities** | **Planned Start Date** |
| Static Testing | Dec 3, 2021 |
| Test Scripting and Execution (includes Manual, Automated and Performance Testing) | Dec 6, 2021 |
| Defect Management | Dec 15, 2021 |
| Testing Sign Off | Dec 17, 2021 |

1. **Testing Entry and Exit Criteria**

The entry criteria below defines the conditions to be satisfied in order for the testing to begin and exit criteria define the conditions that have to be satisfied in order to sign off the testing.

|  |
| --- |
| **Entry Criteria** |
| Environment is working as expected. |
| Kick-off meeting must be set - Defect Management/Test Strategy |
| All walkthrough/Sprint Planning meetings are completed |
| **Exit Criteria** |
| All test cases (Manual, Automation and Performance Testing) are 100% executed. |
| All test documentation is published and signed off by QA Manager |
| No Critical/Major defects open  ETC/ Resolution plan for Minor/Cosmetic open defects |