Buggy Cars Rating

May 28, 2021

# Test PLan

## Introduction

|  |  |
| --- | --- |
|  | The purpose of testing is to provide the Product Owners and relevant stakeholders with information regarding the quality of the Application Under Test (AUT) so they can make informed decision around it.  This strategy document defines the general test approach and facilitate the successful delivery of the BUGGY CARS RATING application. |

It will define the following:

* Scope of testing
* Test Process, including test management and defect management

## Audience

|  |  |
| --- | --- |
|  | The intended audience for this document is:   * Project Managers * Architects * Testers * Product Owner * Business Analyst * Scrum Teams |

## Test Approach

|  |  |
| --- | --- |
|  | The main goal of the Testing Team is to test the all the known features, error handling and process flow of the AUT.  Each team is responsible for the system development, which when completed and entry criteria satisfied, will be handed over to the testing team for testing. |

*\*ASSUMPTIONS:*

* *Working on Agile environment*
* *No documentation provided.*
* *Current version of the AUT is only available*
* *Rules are validated based on existing error messages*

Given that we are working on a less than perfect situation, the team will be leaning to the reported issues by the users; and testers judgement and intuition.

Since the application under test is a Voting app that requires Registration, Login and etc. We have arrived to following basic scenarios:

* New User can register to the app/User can only register once.
* User can login with correct credentials.
* Initial page is homepage after logging in.
* Vote can be updated or removed.
* User cannot vote twice on the same car.
* User can logout the app and exit the last transaction.
* Vote count is correctly incremented.
* Comments are correctly displayed.
* User are able to sort the cars ranking by vote.
* Users Login credentials adhere to the minimum standard.

The Testing Team will perform the following:

* Functional Testing in conjunction with defect management and UAT test support if required.
  + System Testing (End to End Testing)
  + Regression test will be required when doing new releases/deployments.
  + Exploratory Testing (Manual) – end of system testing, before go live
  + Graphical User Interface (GUI) Testing
  + Boundary Value Testing
* Discovered defects will be collated and recorded for analysis. Valid defects will be logged in *<Defect Management Tool>* and assigned to appropriate teams for resolution.

## In Scope

|  |  |
| --- | --- |
|  | * + 1. In Scope Testing Activities (UI based testing) |

* Basic Features (Registration, Login/Logout, Voting) of BUGGY CARS Rating
* GUI Test – Validate look and feel of the application (navigation, filters, sorting and etc.).
* Application Error Handling

## Out of Scope

|  |  |
| --- | --- |
|  | 2.2.1 Out of Scope Testing Activities |

* Service Level Testing – Validate web service level features (APIs)
* Non-functional Testing – Performance test (load, stress)
* Database Testing

## System Workflow

|  |  |
| --- | --- |
|  |  |

## Types of Testing

|  |  |
| --- | --- |
|  |  |

* System Testing – since there is no requirement provided, the entire system is tested as per the obvious features of the application under test. Please see system workflow for the basic features that we are aiming to test.
* Graphical User Interface Testing – This testing includes the look and feel validation of the application such as navigation, filters, sorting and etc.
* Regression Testing – This testing will be required when having new release/deployments. All features (Registration, Login, Voting, Logout) and even the End-to-End flow are included on this testing.
* Exploratory Testing – The objective of this testing is to explore the application and looking for defects that exist in the application. Sometimes it may happen that during this testing major defect discovered can even cause a system failure. Some Regression Test Cases can be originated during this testing period.

## DOR and DOD for Testing Team

|  |  |
| --- | --- |
|  | * 1. Definition of Ready: |

* Test Environment is ready and code has been deployed.
* Acceptance Criteria is clearly defined
* Dependencies are identified
* Story points estimated

|  |  |
| --- | --- |
|  | * 1. Definition of Done: |

* No Severity 1 or 2 defects open
* Acceptance Criteria met
* Automated tests scripted, passed, result available.
* Automated scripts code have been peer reviewed.
* Automation test has been merged in Repository Master
* Setup Pipeline and hook up with development build releases or scheduled run.

## Test Management

|  |  |
| --- | --- |
|  | * 1. Test Case Management |

* Test Cases for Testing Team will be recorded and managed in Specflow integrated **in https://github.com/karyll-urma/.** Test cases will be linked to user stories.

|  |  |
| --- | --- |
|  | * 1. Defect Management |

* Defects will be tracked in <Defect Management Tool>. The business severity and priority will be categorized based on the definitions below, (These definitions are subject for approval by the product owner)

4.2.1 Business Severity

|  |  |
| --- | --- |
| Business Severity | Definition |
| 1 - Critical | * A showstopper. No system availability and it does not have a work around. Cannot continue until this defect is fixed, e.g., a system crash. Issue is preventing further execution. An immediate fix is required. |
| 2 - High | * Major impact with no existing workaround. Fix as soon as possible. A defect that impacts the ability to perform a significant function of the system, with no workaround potential. Any such defect will prevent the system being place in Production or consumed by the business. |
| 3 - Medium | * The defect affects minor functionality. There is a workaround in place for this defect. |
| 4 - Low | * Cosmetic issue with minimal impact on the system. There is a workaround in place for this defect. |

4.2.2 Priority

|  |  |
| --- | --- |
| Priority | Definition |
| Blocker | * Requires an immediate fix/patch to current sprint. |
| High | * Required fix must be next sprint. |
| Minor | * Fix maybe in the next sprint or maybe deferred and has higher fix priority than Trivial defects. |
| Trivial | * Lowest priority order to fix. |

|  |  |
| --- | --- |
|  | * 1. Test Deliverables |

|  |  |
| --- | --- |
| Deliverables | Definition |
| Test Strategy | High Level test strategy. |
| Test Cases | End to End test cases |
| Automated Test Suites | For regression/system tests |
| Test Exit Summary Reports |  |

## Tools and Data

|  |  |
| --- | --- |
|  | * 1. Test Tools |

The following tools and utilities may be required to support the project

|  |  |
| --- | --- |
| Tools | Purpose |
| Specflow | To define BDD test cases |
| Visual Studio C# with Selenium | For UI automated tests |
| GitHub | Repository will be uploaded to GitHub**(https://github.com/karyll-urma/Test.BCRating)** |
| <Test Case Management Tool> e.g., JIRA | To define manual test cases |

|  |  |
| --- | --- |
|  | * 1. Test Data |

Test Data requirement will vary depending on the type of testing

|  |  |
| --- | --- |
| Type of Testing | Comments |
| Automated Regression Test | It is expected that the automated regression test will call on existing data, or set up the required data itself. |
| Exploratory Test | Call on existing data, or set up the required data itself. |
| System Test (can be automated) | Call on existing data, or set up the required data itself. |
| GUI Test | Call on existing data, or set up the required data itself. |