

TEST APPROACH

Testing Scope

For the purposes of this exercise, the following functional areas were identified as main areas for testing and quality assurance:

- User Registration
- Login & Logout
- Profile Change
- Votes & Comments
- Page/Table Navigation

Manual Tests

All areas above were exercised by both manual and automated tests, with manual testing involving several black-box techniques, below are some case examples of when each technique was applied.

- Equivalence Partitioning
 - Test restricted characters for login names
 - Test expected characters for passwords
- State Transition Testing
 - Test page navigation for model tables
 - Test page navigation for buttons and links
- Boundary Value Analysis
 - Test accepted range for the Age field
 - Test login names and password lengths
- Decision Table Testing
 - Test ability/inability to cast votes/comments

Functional testing involved checking input validation for all fields on all pages. Both positive and negative tests were done around the 5 major functional areas, with profile changes including password changes.

Usability testing focused on ensuring page navigation and table browsing worked as intended. That basically involved validating that buttons, images, and static links performed the expected actions and led users where they were supposed to when clicked.

Performance testing was not addressed by any manual or automated tests in specific. That said, careful consideration was given to verify and validate load and processing times when executing usability tests.

Most manual testing was performed on Google Chrome, however, exploratory tests (smoke check) and defect confirmation checks were also done on Mozilla Firefox and Microsoft Edge. This decision was taken considering the following browser market share statistics:

<https://gs.statcounter.com/browser-market-share/desktop/worldwide>

Safari would have been used as test platform as well, but I do not own any Apple device. Smoke checks and defect confirmation checks for the top 3 major issues were performed on a mobile device as well, a Samsung S9 using Samsung Internet and Google Chrome browsers.

Automated Tests

Automated tests were written in Gherkin language using SpecFlow feature files and step definitions. These were built on top of a Page Object Model framework written for Selenium in C# language.

The following 30 tests were created to cover the 5 main functional areas defined above as the scope of testing:

- EditProfileFeature
 - AlertInvalidInput_Age
 - AlertRequiredInput_FirstName
 - AlertRequiredInput_LastName
 - CancelUserProfileChanges
 - ChangeUserAdditionalInfo
 - ChangeUserPasswordInfo
- LoginLogoutFeature
 - LoginLogoutFromPage_Home
 - LoginLogoutFromPage_Maker
 - LoginLogoutFromPage_Model
 - LoginLogoutFromPage_Overall
 - LoginLogoutFromPage_Profile
 - LoginLogoutFromPage_Register
 - LoginWithInvalidCreds_Username
 - LoginWithInvalidCreds_Password
- RegisterUserFeature
 - AlertRequiredInput_ConfirmPassword
 - AlertRequiredInput_FirstName
 - AlertRequiredInput_LastName
 - AlertRequiredInput_LoginName
 - AlertRequiredInput_Password
 - CancelUserRegistration
 - RegisterUserWithSameId
 - RegisterUserWithUniqueId
- TableActionsFeature
 - BrowseTableBackwardInPage_Maker
 - BrowseTableBackwardInPage_Overall
 - BrowseTableForwardInPage_Maker
 - BrowseTableForwardInPage_Overall

- VoteCommentFeature
 - VoteAllowedWithComment
 - VoteAllowedWithoutComment
 - VoteDeniedWhenAlreadyCast
 - VoteDeniedWhenLoggedOut

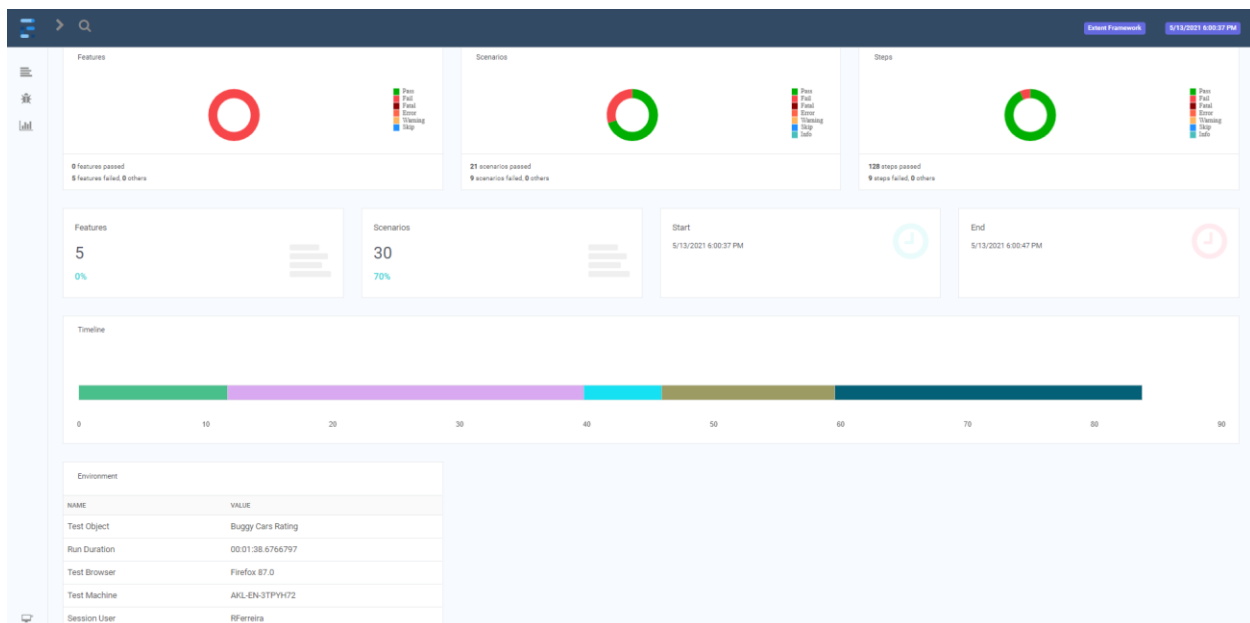
These tests are independent from each other and can be run in any order. They are self-documented thanks to the use of Gherkin language, so if their names are not descriptive enough, one may check their steps in the test reports (details in the next topic).

The test suite can be run against Chrome or Firefox. Edge would have been included as well, but Selenium does not yet have a WebDriver capable of driving automation on Microsoft Edge.

Test Reports

Furthermore, reports for each test run are generated using the Extent Reports framework. These reports contain detailed information about the test run, test features, test scenarios, and test steps.

These reports can be found in a folder named “ExtentReport”, located in the project repository’s main directory. Three HTML files are generated by each test run, but one may simply open the one called “dashboard.html” and use the top-left corner menu to navigate all its pages (Dashboard, Exceptions, and Tests).



Dashboard Page

We can check the steps taken for each scenario and, for each failed scenario, we have the respective exception thrown by the test and an image capture of the browser at the time of the failure to aid in failure investigation.

The screenshot displays a test framework interface. On the left, a sidebar lists several tests: 'Edit Profile', 'Login & Logout', 'Register User', 'Table Actions', and 'Vote & Comment'. The 'Edit Profile' test is selected. The main area on the right shows the details of the 'Edit Profile' test. It includes a title 'Edit Profile' with a timestamp '5/15/2021 5:58:13 PM'. Below the title, a list of test steps is shown, each with a status icon (green for passed, red for failed). The steps are: 'Alert Invalid Input - Age' (failed), 'Given the "profile" page is displayed' (passed), 'Given the current age is recorded for reference' (passed), 'When the text "Thirty One" is entered for Age' (passed), 'When the "Save" button is clicked' (passed), and 'Then the "Age is not valid" message pops up' (failed). The failed step is expanded, showing a screenshot of the browser and a detailed error message: 'Unable to locate element: //div[contains(@class,'alert')] and contains(text(),Age is not valid)'. Below the error message, there are several other test steps, all of which are marked as passed: 'Alert Required Input - First Name', 'Alert Required Input - Last Name', 'Cancel User Profile Changes', 'Change User Additional Info', and 'Change User Password Info'.

Tests Page

Note: The images on this document do not make justice to the reports framework, so I will keep the report files generated for my last test run within the repository in GitHub, so they can be checked without having to run the actual tests.

Mind the failed tests are all actually accusing valid bugs in the Buggy Cars Rating platform, but more details on that can be found in the AutoTestGuide.pdf provided with this practical assignment submission.