Buggy Rating Web Application
Test Approach

Test approach for Buggy Rating application can be divided into following areas.

<u>Unit Testing</u>

It’s very crucial to have a good number of quality unit test cases for this application which covers all the basic core functionalities (including backend, frontend and integration layers like APIs). The unit testing should have most of the code coverage and needs to be created, updated and executed as a part of every code change or release.
Recommends automated unit testing frameworks - JUnit, testNG or NUnit

<u>Functional Testing</u>

Functional testing focuses on verifying key functionalities and features in the Buggy Rating application. Type of testing can include system, integration, acceptance testing for,

- Web UI (functionalities and look/feel)
- API requests
- Database operations

Both manual and automation testing is important for functional testing

<u>Manual Testing</u>

Manual testing focuses on test scenarios which,

- Are difficult to automate
- Doesn’t require repeatable execution
- Are edge and negative scenarios which may not necessarily focus on requirements or specifications

Manual Testing Approach

- UI testing - Verify the look and feel against the designs
- Exploratory testing - Verify edge/negative scenarios and scenarios that may not necessarily focus on the requirements. This should be carried out by a test analyst/s with a good level of business/product knowledge related to Buggy Rating. Testing techniques like session-base and/or risk-based testing can be used to make test execution efficient.

<u>Automation Testing</u>

Automation testing focuses on test scenarios which,

- Are core functionalities in the web interface
- Requires repeatable execution (and/or part of regression test-suites)
- Are related to integration components (i.e. API layer)

Automation Testing Approach

- Headful-browser testing - This focuses on creating automated test scenarios in real web-browsers (i.e. Chrome, Firefox etc). This can be useful to verify the behavior of the web application in various web-browsers and mobile devices (useful for compatibility

testing). However, the cost for building and maintaining these types of automation test cases are high as UI behaviour in various browsers (with different versions and operating systems) can be unstable and unreliable. Tools for,
  - Desktop browsers - Selenium
  - Mobile browsers - Appium
- Headless-browser testing - This focuses on creating automated test scenarios for UI functional tests for web browsers but it skips loading the browser's UI. These tests don't render web pages and tests bypass interacting with the page to control the browser more directly. These types of tests are more efficient and reliable which requires less effort to maintain. Recommended tool is PhantomJS with Selenium.
- API testing - It's very cost effective and has a high ROI to the automation API layer of the solution. Because API layer is fairly more stable compared to Web UI and it's easier to create and maintain API automation tests. As the Buggy Rating is a less complex web solution, the recommended tool for API automation is Postman.

Compatibility Testing

Involves testing features and functionality of the Buggy Rating in various browsers and mobile devices. It's crucial to get browser stats (browsers, their versions, mobile devices and operating systems that are used by users and number of users) for Buggy Rating. Then the test team can decide testing can be focused on which,

- Browsers
- Devices
- Operating Systems

Compatibility tests can run in both manual and automation manner (check Functional Testing section for more details).

Usability Testing

Based on the initial testing of the Buggy Rating application, it's evident that there are multiple usability issues and the system isn't very user-friendly. Hence, it's important to conduct usability testing for the application and this can be done by the focused-user groups or experienced usability QAs.

Performance Testing

It's very important to understand the user statistics for the Buggy Rating and performance requirements before designing a detailed test approach for performance testing. The user statistics can be based on the data collected in the past and the projected growth. Example user statistics can include but not limited to,

- Average number of users per second, minute, hour and day
- Maximum recorded number of users per second, minute, hour and day
- Projected growth of the users in the future
- Number of users based on the geographical locations

- Number of the users based on the features on the application
- Details about any reported issues related to performance or reliability of the application

Based on the information available, a high-level performance testing approach for Buggy Rating application can focus on three different techniques to evaluate performance.

Webpage Speed Test

This focuses on testing the webpage's speed on a real browser. It's fairly simple to conduct and free online tools can be utilized. This isn't a load test but the test tool analyses the content of the webpage and generates a lot of insights and suggestions to make the web page faster. Recommended tools are,

- https://developers.google.com/speed/pagespeed/insights/
- https://www.webpagetest.org/

Browser Load Testing

Focuses on generating load on the web-pages by real browsers loading web-pages. These tests can represent real user scenarios (i.e. login, registration etc.). Performance metrics like response time are recorded based on the actual time it takes to perform an operation on a real browser when the application is on a certain load.

These types of tests can generate different levels of load from different geographical locations on different types of browsers. These are particularly useful if the Buggy Rating application is used by the users from different geographical locations in the world (and Content Delivery Network is utilized to deliver content).

Selenium scripts written for functional testing can be reused in conjunction with cloud load testing tools for these types of testing. Recommended tool is Flood.io

Protocol Performance Testing

This involves conducting standard performance testing on the Buggy Rating application which involves performance evaluation of the dynamic (i.e. JSP, Servlets etc.) and static (DOM elements like JavaScript and HTML) resources of the application. This can include the different types of performance testing like load, stress, endurance, spike, volume and scalability testing. Recommended tool is JMeter.

If the performance bottlenecks or issues are identified during the Protocol Performance Testing, they need to be investigated by conducting more lower level performance testing on different components of the application. These include but not limited to,

- API
- Database servers
- Cloud or servers