Test Plan

NASA

project

www.linkedin.com/in/kristina-tablev

Table of Contents:

Change Log:

Version	Change Date	By	Description
Version number	Date of Change	Name of person who made changes	Description of the changes were made

Document Revision History:

Date	Version	Author	Description	Reviewer	Approver

1. INTRODUCTION

The **Test Plan** is designed to prescribe the *scope, approach, resources,* and *schedule* of all testing activities of the project NASA website https://www.nasa.gov/

The plan identifies the *items* to be tested, the *features* to be tested, the *types of testing* to be performed, the *personnel responsible* for testing, the *resources* and *schedule* required to complete testing, and the *risks* associated with the plan.

Customers want a perfect website, which has passed the full cycle of manual testing. Given the specificity of the site it is very important to have the same quality and the site.

2. SCOPE

2.1 In Scope:

All the features of website NASA which were defined in software requirement *specs need to be tested.* The document mainly targets to check the Registration form for NASA's Virtual Guests, GUI of the website, NASA Social Media Module, NASA's API, Website Performance and validates data in report output as per *Requirements Specifications* provided by Client.

Functions to be tested:

- GUI
- Registration form NASA's Virtual Guests
- NASA Social Media Module
- NASA's API Collections
- Website Performance

2.2 Out of Scope:

These features are *not to be tested* because they are not included in the software requirement specs.

Functions *not* to be tested:

- Hardware Interfaces
- Software Interfaces
- Database logical
- Communications Interfaces
- Website Security

3. QUALITY OBJECTIVES

The *test objectives* are to verify the functionality of website NASA, the project should focus on testing the Website's GUI, Registration form, API, Performance and guarantee all these operations can work normally in a real business environment.

3.1 Primary Objectives:

A *primary objective* of testing is to: assure that the *system meets the full requirements*, including quality requirements (functional and non-functional requirements) and fit metrics for each quality requirement and satisfies the use case scenarios and maintain the quality of the product. At the end of the project development cycle, the user should find that the project has m*et or exceeded all of their expectations as detailed in the requirements.*
Any changes, additions, or deletions to the requirements document, Functional Specification, or Design Specification *will be documented* and tested at the highest level of quality allowed within the remaining time of the project and within the ability of the test team.

3.2 Secondary Objectives:

The *secondary objectives* of testing will be to: identify and expose all *issues* and *associated risks*, communicate all known issues to the project team, and ensure that all issues are addressed in an appropriate manner before release. As an objective, this requires *careful and methodical testing* of the website to first ensure all areas of the system are scrutinized and, consequently, all issues (bugs) found are dealt with appropriately.

4. ROLES AND RESPONSIBILITIES

The project should use outsource members as the testers also to save the project cost.

No.	Member	Tasks
1.	Test Manager	Manage the whole project Define project directions Acquire appropriate resources
2.	Testers	Identifying and describing appropriate test techniques/tools/automation.Verify and assess the Test Approach Execute the tests, Log results, Report the defects.
3.	Developer in Test	Implement the test cases, test program, test suite etc.
4.	Test Administrator	Builds up and ensures test environment and assets are managed and maintained Support Tester to use the test environment for test execution
5.	SQA members	Take in charge of quality assurance Check to confirm whether the testing process is meeting specified requirements

5. TEST APPROACHES

In the project NASA, there are 4 types of testing that should be conducted:

- Manual Tests
- Automation Tests
- API Tests
- Performance Tests

The project is using an ***Agile approach*****,** with weekly iterations (every 14 days). At the end of the *second week* the requirements identified for that iteration will be delivered to the team and will be tested.

Team also must use experience-based testing and error guessing to utilize testers' skills and intuition, along with their experience with similar applications or technologies.

5.1 Overview:

1. **Website Manual Tests:**

1. The NASA Website's Module **Menu "FOLLOW NASA"** will be tested In this part.

Sub Menu Links for testing:

- **"NASA Newsletters"** :

Check the ability of user sign up for NASA's Newsletters with:

1. Valid datas (valid email) - Positive testing
2. Invalid datas (invalid email) - Negative testing

- **"Social Media at NASA"** :

Check the ability of user to reach all NASA's social medias pages by clicking on the following provided Social Media Icons:

Twitter, Facebook, Instagram, SnapChat, YouTube, Tumblr, Pinterest, LinkedIn, GIPHY, Flickr, Twitch, Soundcloud, Reddit, Dailymotion.

Automation tests with Selenium IDE will be *also* created for this testing part.

- **"Join the Virtual Guest List"** :

Check the ability of user to make a registration for NASA's upcoming events with:

1. Valid datas (valid email) - Positive testing
2. Invalid datas (invalid email) - Negative testing

2. The **UI/UX *Smoke testing***

will be performed for NASA's website for verification UI response to Design requirements.

Sub Plan for UI/UX testing:

1. Verify that all images' links on Homepage are working correctly according to Business requirements.
2. Verify that Logo leads at the website's Homepage after clicking on it.
3. Verify that the "Search" field is working correctly according to Business requirements.

- Positive/Negative/Ad hoc testing should be used *also* if necessary.

Environments for this part:

1. OS:

- MacOS
- iOS
- Windows 10

2. Devises:

- MacBook
- iPhone 12 Pro Max
- Android

3. Browsers:

- Google Chrome
- Safari (for MacOS)
- FireFox (for Windows OS)

3. Website Automation Tests:

Tests which are indicated in this part will be automated:

1. Test for Module **Menu** "FOLLOW NASA" with Sub Menu "Social Media at NASA" (to verify the ability of users to reach all NASA's social media pages).

2. Tests for UI/UX Smoke testing (to verify the response of UI/UX to design requirements).
3. Test for “Search” field (to verify the correct functionality of this field).

Tools for this part:

- Selenium IDE
- XPath (ChroPath/TruPth)

4. Website API Tests:

NASA's API Collections will be tested through the Server response in this part.

API Collections for testing:

- **Asteroids - NeoWs**

NeoWs (Near Earth Object Web Service) is a RESTful web service for near earth Asteroid information. With NeoWs a user can: search for Asteroids based on their closest approach date to Earth, lookup a specific Asteroid with its NASA JPL small body id, as well as browse the overall data-set.

- **Earth**

Landsat imagery is provided to the public as a joint project between NASA and USGS. This API is powered by Google Earth Engine API, and currently only supports pan-sharpened Landsat 8 imagery.

- **TLE API**

The TLE API provides up to date two line element set records, the data is updated daily from CelesTrak and served in JSON format. A two-line element set (TLE) is a data format encoding a list of orbital elements of an Earth-orbiting object for a given point in time.

Tools for this part:

- Postman 2
- JSON Formatter & Validator

5. Website Performance Tests:

Tests for evaluation of the website's *speed, responsiveness and stability* under a workload will be done in this part.

Sub Plan for Performance testing:

1. Check NASA's website:
    - Performance
    - Accessibility
    - Best practice
    - SEO

by using performance testing Tools:

1. Lighthouse
2. GT Metrix
3. SpeedLab

Tools for this part:

- Lighthouse
- GT Metrix
- SpeedLab

6. TEST TYPES

In the NASA project there are 9 types of testing that should be conducted:

1. Exploratory testing
2. Smoke Testing
3. GUI Testing
4. Functional Testing
5. Positive testing
6. Negative testing
7. ADHOC testing
8. API testing
9. Performance testing

Exploratory testing :

Exploratory testing will include a type of software testing where Test cases are not created in advance but QA check system on the fly. QA may note down ideas about what to test before test execution.

Smoke Testing:

Smoke Testing is a software testing process that determines whether the deployed software build is stable or not. Smoke testing is a confirmation for the QA team to proceed with further software testing. It consists of a minimal set of tests run on each build to test software functionalities. Smoke testing is also known as "Build Verification Testing" or "Confidence Testing."

In simple terms, we are verifying whether the important features are working and there are no showstoppers in the build that is under testing.

GUI Testing:

GUI testing will include testing the UI/UX part of the report. It covers users' Report format, look and feel, error messages, spelling mistakes, GUI guideline violations.

Functional Testing:

Functional testing is carried out in order to find out *unexpected behavior* of the report. The characteristics of functional testing are to provide correctness, reliability, testability and accuracy of the report output/data.

Positive testing:

Positive testing will includes the type of testing that can be performed on the system by providing the *valid data* as input. It checks whether an application behaves a*s expected* with positive inputs.

Negative testing:

Negative testing will include a method of testing an application or system that ensures that the application is according to the requirements and can handle the unwanted input and user behavior. *Invalid data* is inserted to compare the output against the given input. Negative testing is also known as *failure testing* or error path testing. When performing negative testing *errors messages* are expected.

ADHOC testing:

ADHOC testing will include an informal testing type with an aim to **"break"** the system.

API testing:

API testing is a type of software testing that analyzes an application program interface (API) to verify it fulfills its expected functionality, security, performance and reliability.An API test is generally performed by making requests to one or more API endpoints and comparing the response with expected results.

Performance testing:

Performance testing is the practice of evaluating how a system performs in terms of responsiveness and stability under a particular workload. Performance tests are typically executed to examine speed, robustness, reliability, and application size.

7. TEST STRATEGY

7.1 QA role in test process:

- Understanding *Requirements.*
- Requirement *specifications* will be sent by client.
- Understanding of requirements will be done by QA.
- Preparing *Test Cases*:

QA will be preparing test cases based on the exploratory testing. This will cover all scenarios for requirements.

- Preparing *Test Matrix*:

QA will prepare a RTM which maps test cases to respective requirements. This will ensure the coverage for requirements.

- Reviewing test cases and matrix.
- Peer review will be conducted for test cases and test matrix by QA Lead.
- Any comments or suggestions on test cases and test coverage will be provided by the reviewer respective Author of Test Case and Test Matrix.
- Suggestions or improvements will be re-worked by the author and will be sent for approval.
- Re-worked improvements will be reviewed and approved by the reviewer.
- *Creating Test Data:*

Test data will be created by respective QA on client's developments/test site based on scenarios and Test cases.

- *Executing Test Cases*:

Test cases will be executed by respective QA on the client's development/test site based on designed scenarios, test cases and Test data.

- Test result (Pass/Fail) will be updated in test case document *Defect Logging and Reporting*: QA will be logging the defect/bugs in Word document and JIRA, found during execution of test cases.
- QA will inform the respective developer about the defect/bugs.
- QA will perform Retesting and Regression Testing:

Retesting for fixed bugs will be done by respective QA once it is resolved by the respective developer and bug/defect status will be updated accordingly. In certain cases, regression testing will be done if required.

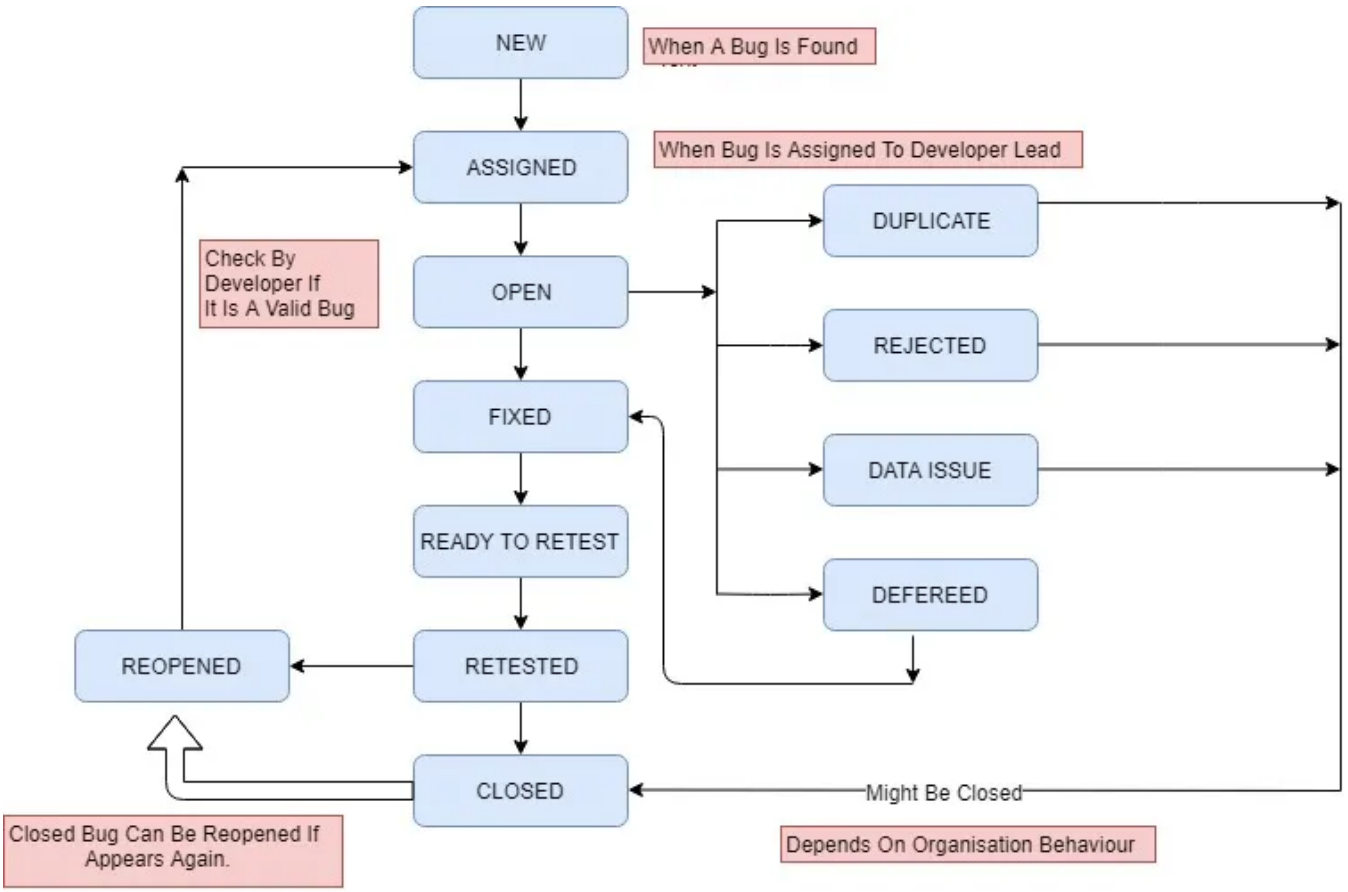
- Deployment/Delivery:

Once all bugs/defects reported after complete testing are fixed and no other bugs are found,the report will be deployed to the client's test site.

Once a round of testing will be done by QA on the client's test site if required Report will be delivered along with sample output by email to the respective lead and Report group.

7.2 Bug Triage:

All the issues found while testing will be logged into JIRA



7.2.1 Bug life cycle:

7.2.2 Bug Severity and Priority Definition:

Bug **Severity** and **Priority** fields are both very i*mportant for categorizing bugs* and *prioritizing* if and when the bugs will be fixed.

The bug Severity and Priority levels will be defined as outlined in the following tables below. Testing will assign a severity level to all bugs. The Test Lead will be responsible to see that a correct severity level is assigned to each bug.

The QA Lead, Development Lead and Project Manager will participate in bug review meetings to assign the priority of all currently active bugs.

This meeting will be known as "*Bug Triage Meetings*".

The QA Lead is responsible for setting up these meetings.

7.2.3 Bug Severity List:

Severity ID	Severity	Severity Description
1	**Highest**	The module/product *crashes* or the bug causes *non-recoverable conditions*. System crashes or database or file corruption, or potential data loss, program hangs requiring reboot are all examples of a **Severity 1 bug**.
2	**High**	*Major system components unusable due to failure* or incorrect functionality. **Severity2** bugs cause serious problems such as a *lack of functionality*, or insufficient or unclear error messages that can have a major impact to the user, prevents other areas of the app from being tested, etc. Severity 2 bugs can have a work around, but the work around is inconvenient or difficult.
3	Medium	*Incorrect functiona*lity of component or process. There is a simple work around for the bug if it is **Severity 3.**
4	Low	Documentation errors or signed off **Severity 4** bugs. Low severity bug occurs when there is *almost no impact* on the functionality but it is still a valid defect that should be corrected.

7.2.4 Bug Priority List:

Priority	Priority Level	Priority Description
1	**Highest**	This bug *must be fixed immediately*; the product cannot ship with this bug.
2	**High**	These are important problems that s*hould be fixed as soon as possible*. It would be an embarrassment to the company if this bug shipped.
3	Medium	The problem *should be fixed within the time available*. If the bug does not delay the shipping date, then fix it.
4	Low	It is not important (at this time) that these bugs be addressed. *Fix these bugs after all other bugs* have been fixed.
5	Lowest	Documentation errors.

8. ENTRY AND EXIT CRITERIA

8.1 Entry Criteria:

- All test *hardware platforms* must have been successfully installed, configured, and functioning properly.
- All the necessary d*ocumentation, design, and requirements* information should be available that will allow testers to operate the system and judge the correct behavior.
- All the standard *software tools* including the testing tools must have been successfully installed and functioning properly.
- Proper *test data* is available.
- The *test environment* such as lab, hardware, software, and system administration support should be ready.
- QA resources have completely *understood* the requirements.
- QA resources have strong *knowledge* of functionality.
- *Reviewed* test scenarios, test cases and RTM.

8.2 Exit Criteria:

- A certain level of requirements coverage has been achieved.
- No high priority or severe bugs are left outstanding.
- All high-risk areas have been fully tested, with only minor residual risks left outstanding.
- Cost – when the budget has been spent.
- The schedule has been achieved.

9. SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS

9.1 Suspension criteria:

- The *build* contains many serious defects which seriously or limit testing progress.
- Significant *change* in requirements suggested by client.
- Software/Hardware problems.
- Assigned resources are not available when needed by test team

9.2 Resumption criteria:

- Resumption will only occur when the problem(s) that caused the suspension have been resolved.

10. RESOURCE AND ENVIRONMENT NEEDS

10.1 Testing Tools:

Process	Tool
Test case creation	Microsoft Word, Microsoft Excel, JIRA
Test case tracking	JIRA, Confluence
Test case execution	Manual, Selenium IDE
Test case management	Microsoft Excel, JIRA, Confluence
Defect management	Microsoft Word, JIRA, Confluence
Test reporting	JIRA

API Testing	Postman
Performance Testing	Lighthouse, GT Metrix, SpeedLab
Automation Testing	Selenium IDE, XPath (for locators searching)

10.2 Test Environment x Support level 1 (browsers):

Test Environment to be setup as per figure below:

- Windows 10: Chrome (latest), Firefox (latest)
- Mac OS : Chrome (latest), Safari (latest)
- Android: Chrome (latest), Firefox (latest)

Support level 1 (devices):

- iPhone 12 Pro Max

Support level 1 (browsers):

- Safari (latest)

11. TEST SCHEDULE

Task Name	Start	Finish	Effort	Comments

12. APPROVALS

	Project Manager	QA Lead
Name		
Signature		

13. TERMS/ACRONYMS

The below terms are used as examples, please add/remove any terms relevant to the document.

TERM/ACRONYM	DEFINITION
API	Application Program Interface
GUI	Graphical user interface
PM	Project manager
UAT	User acceptance testing
CM	Configuration Management
QA	Quality Assurance
RTM	Requirements Traceability Matrix