Table of Contents

1. Introduction 2

2. Scope 2

3. Quality Objectives 3

3.1 Primary Objectives 3

4. Roles and Responsibilities 4

5. Test Approaches 5

5.1 Overview: Manual Tests 6

6. Test Types 10

7. Test Strategy 11

7.1 QA Role in the Test Process 11

7.2 Bug Triage 12

8. Entry and Exit Criteria 15

8.1 Entry Criteria 15

8.2 Exit Criteria 15

9. Suspension Criteria and Resumption Requirements 16

9.1 Suspension Criteria 16

9.2 Resumption Criteria 16

10. Resource and Environment Needs 17

10.1 Testing Tools: 17

10.2 Test Environment 18

11. Test Schedule 18

12. Approvals 18

13. Terms/Acronyms 19

NASA Website Test Plan

1. Introduction

The NASA website test plan outlines the approach, resources, and schedule for testing activities. It ensures that the website meets quality standards and functions as expected. The plan covers the following aspects:

- **Scope**: Defines what features will be tested.
- **Approach**: Describes the testing methodology.
- **Resources**: Identifies personnel and tools.
- **Schedule**: Outlines the testing timeline.
- **Risks**: Addresses potential challenges.

2. Scope

2.1 In Scope

The following features, as defined in the software requirement specifications, will be thoroughly tested:

1. **GUI (Graphical User Interface)**:
   - Validate the visual elements, layout, and responsiveness of the website.
   - Ensure consistent branding and user-friendly design.
2. **Registration Form for NASA's Virtual Guests**:
   - Verify the functionality of the registration process.
   - Confirm that user data is captured accurately.
3. **NASA Social Media Module**:
   - Test integration with social media platforms.
   - Validate content sharing and interaction features.
4. **NASA's API Collections**:
   - Evaluate the functionality and reliability of APIs.
   - Confirm data retrieval and communication.
5. **Website Performance**:
   - Measure page load times, responsiveness, and scalability.
   - Identify bottlenecks and optimize performance.
6. **Data Validation in Report Output**:
   - Validate data presented in reports against specified requirements.
   - Ensure accuracy and consistency.

7. **Website Security** (Handled Separately):
   - Conduct security-specific testing to identify vulnerabilities.
   - Address authentication, authorization, encryption, and other security aspects.

2.2 Out of Scope

The following features are not part of the testing scope due to their exclusion from the software requirement specifications:

- **Hardware Interfaces**: Testing interactions with physical devices.
- **Software Interfaces**: Testing interactions with external software systems.
- **Database Logic**: Detailed database testing.
- **Communication Interfaces**: Network-related testing.

3. Quality Objectives

3.1 Primary Objectives

1. **Functionality Verification**:
   - Confirm that all features of the NASA website work as intended.
   - Ensure seamless navigation and interaction.
2. **GUI Testing**:
   - Validate the visual presentation, consistency, and responsiveness.
   - Verify alignment with design guidelines.
3. **Registration Form Testing**:
   - Ensure successful user registration.
   - Validate form fields, error handling, and data persistence.
4. **API Testing**:
   - Verify the functionality of NASA’s APIs.
   - Test data retrieval, authentication, and error handling.
5. **Performance Testing**:
   - Measure website load times, responsiveness, and resource utilization.
   - Identify performance bottlenecks.
6. **Real Business Environment Validation**:
   - Ensure that all tested features function normally in a production environment.
   - Address any discrepancies or issues.

3.2 Secondary Objectives

1. **Issue Identification and Risk Exposure**:
   - Thoroughly scrutinize all areas of the NASA website to identify and expose any issues (bugs).
   - Document all known issues and associated risks.
   - Communicate these issues promptly to the project team.
2. **Appropriate Issue Resolution**:

    - Ensure that all identified issues are addressed effectively before the website’s release.
    - Prioritize issues based on their impact and severity.
    - Maintain a methodical approach to issue resolution.

By rigorously testing against these objectives, our goal is to ensure the NASA website is deployed and operational with optimal quality, surpassing functional requirements, delighting users, and mitigating potential risks.

4. Roles and Responsibilities

1. **Test Manager**:
    - **Role**: Manages the entire testing project.
    - **Tasks**:
        - Define project directions and testing goals.
        - Acquire appropriate testing resources (both human and technical).
        - Oversee the overall testing process.
        - Ensure alignment with project objectives.
2. **Testers**:
    - **Role**: Responsible for executing test activities.
    - **Tasks**:
        - Identify and describe suitable test techniques, tools, and automation approaches.
        - Assess the chosen test approach.
        - Execute test cases.
        - Log test results.
        - Report defects to the development team.
3. **Developer in Test**:
    - **Role**: Focuses on test implementation.
    - **Tasks**:
        - Implement test cases, test programs, and test suites.
        - Collaborate with developers to ensure testability of features.
        - Contribute to test automation efforts.
4. **Test Administrator**:
    - **Role**: Establishes and maintains the test environment.
    - **Tasks**:
        - Set up the necessary test infrastructure (hardware, software, databases, etc.).
        - Manage and maintain test assets.
        - Support testers in using the test environment effectively during test execution.
5. **SQA (Software Quality Assurance) Members**:
    - **Role**: Oversee quality assurance aspects.
    - **Tasks**:

        - Monitor the testing process to ensure compliance with specified requirements.
        - Verify that testing aligns with quality standards.
        - Participate in reviews and audits.

By clearly defining these roles and responsibilities, we ensure effective collaboration and efficient execution of the testing activities for the NASA website.

5. Test Approaches

In the project NASA, we will employ the following test approaches to ensure comprehensive testing coverage:

1. **Manual Tests**:
    - Manual testing will be conducted by testers to validate various aspects of the website.
    - Includes exploratory testing, usability testing, and ad-hoc testing.
    - Focuses on user interactions, visual elements, and overall functionality.
2. **Automation Tests**:
    - Automated test scripts will be developed to execute repetitive and regression tests.
    - Covers scenarios that can be efficiently automated.
    - Ensures consistent and repeatable testing.
3. **API Tests**:
    - API testing will verify the functionality, reliability, and security of NASA's APIs.
    - Includes testing endpoints, data exchange, authentication, and error handling.
    - Ensures seamless communication between components.
4. **Performance Tests**:
    - Performance testing evaluates the website's responsiveness, scalability, and resource utilization.
    - Measures page load times, response times, and system behavior under load.
    - Identifies bottlenecks and optimizes performance.
5. **Security Tests**:
    - Security testing focuses on identifying vulnerabilities and ensuring data protection.
    - Includes penetration testing, vulnerability scanning, and authentication checks.
    - Validates compliance with security standards.

Agile Approach

The project follows an Agile methodology with bi-weekly iterations (sprints) lasting 14 days each. At the end of each sprint, the requirements identified for that iteration will be delivered to the team for testing. This iterative approach allows for continuous feedback and adaptation.

Experience-Based Testing and Error Guessing

The testing team will leverage their skills, intuition, and experience with similar applications or technologies. Experience-based testing and error guessing will complement formal test procedures, ensuring thorough coverage and early issue detection.

5.1 Overview: Manual Tests

1. Website Manual Tests

In this section, we will perform manual tests on the NASA website's "FOLLOW NASA" module. The following sub-menu links will be tested:

1. **"NASA Newsletters"**:
    - **Positive Testing**:
        - Verify that users can successfully sign up for NASA's newsletters using valid email addresses.
    - **Negative Testing**:
        - Confirm that invalid email addresses (e.g., missing "@" or incorrect format) are rejected during sign-up.
2. **"Social Media at NASA"**:
    - Check the ability of users to reach all NASA's social media pages by clicking on the provided Social Media Icons:
        - Twitter
        - Facebook
        - Instagram
        - SnapChat
        - YouTube
        - Tumblr
        - Pinterest
        - LinkedIn
        - GIPHY
        - Flickr
        - Twitch
        - Soundcloud
        - Reddit
        - Dailymotion
    - Additionally, automation tests using Selenium IDE will be created for this testing part.
3. **"Join the Virtual Guest List"**:
    - **Positive Testing**:
        - Verify that users can successfully register for NASA's upcoming events using valid email addresses.
    - **Negative Testing**:

        - Confirm that invalid email addresses are rejected during registration.

2. UI/UX Smoke Testing

UI/UX smoke testing will be performed to verify the website's response to design requirements. The following sub-plan outlines specific tests:

1. **Image Links on Homepage**:
    - Verify that all image links on the homepage are working correctly according to business requirements.
    - Ensure that clicking on images leads to the expected destinations.
2. **Logo Functionality**:
    - Verify that clicking on the NASA logo leads to the website's homepage.
    - Confirm consistent behavior across different pages.
3. **Search Field Testing**:
    - Verify that the search functionality works correctly according to business requirements.
    - Test positive and negative scenarios (valid and invalid search queries).

Positive, negative, and ad hoc testing will be used as necessary to ensure thorough coverage.

By conducting these tests, we aim to validate the functionality, usability, and visual aspects of the NASA website.

6. Environments

For this testing phase, we will consider the following environments:

1. **Operating Systems (OS)**:
    - **MacOS**: Testing will be performed on MacOS-based systems.
    - **iOS**: Testing will be conducted on iOS devices.
    - **Windows 10**: Testing will cover Windows 10 environments.
2. **Devices**:
    - **MacBook**: Verify compatibility and functionality on MacBook devices.
    - **iPhone 12 Pro Max**: Test responsiveness and usability on iOS mobile devices.
    - **Android**: Ensure cross-platform compatibility.
3. **Browsers**:
    - **Google Chrome**: Validate the website's behavior on the Chrome browser.
    - **Safari (for MacOS)**: Test compatibility with Safari on MacOS.
    - **Firefox (for Windows OS)**: Verify functionality on Firefox running on Windows.

By testing across these diverse environments, we aim to ensure that the NASA website performs consistently and flawlessly for users regardless of their chosen platform.

3. Website Automation Tests

In this section, we will automate specific tests for the NASA website:

1. **Test for Module Menu “FOLLOW NASA” with Sub Menu “Social Media at NASA”**:
   - Verify the ability of users to reach all NASA’s social media pages.
   - Automation tools:
     1. **Selenium IDE**: Create automated test scripts to simulate user interactions.
     2. **XPath (ChroPath/TruPath)**: Use XPath expressions to locate elements on the web page.
2. **Tests for UI/UX Smoke Testing**:
   - Verify the website’s response to design requirements.
   - Sub-plan for UI/UX testing:
     1. Verify that all image links on the homepage are working correctly according to business requirements.
     2. Confirm that clicking on the NASA logo leads to the website’s homepage.
     3. Validate the functionality of the “Search” field according to business requirements.
   - Positive, negative, and ad hoc testing will be used as necessary.

4. Website API Tests

NASA’s API Collections will be tested through server responses:

1. **Asteroids - NeoWs**:
   - NeoWs (Near Earth Object Web Service) provides information about near-Earth asteroids.
   - Test scenarios:
     - Search for asteroids based on their closest approach date to Earth.
     - Look up a specific asteroid using its NASA JPL small body ID.
     - Browse the overall dataset.
2. **Earth**:
   - Landsat imagery is provided to the public as a joint project between NASA and USGS.
   - This API, powered by Google Earth Engine, currently supports pan-sharpened Landsat 8 imagery.
3. **TLE API**:
   - The TLE API provides up-to-date two-line element set records.
   - Data is updated daily from CelesTrak and served in JSON format.
   - A two-line element set (TLE) encodes orbital elements of an Earth-orbiting object for a given point in time.

Tools for this Part:

1. **Postman**: Use Postman for API testing, sending requests, and validating responses.
2. **JSON Formatter & Validator**: Ensure the correctness of JSON data returned by the APIs.

5. Website Performance Tests

In this section, we will evaluate the NASA website's speed, responsiveness, and stability under various workloads.

Evaluate the following aspects:

- **Performance**: Measure page load times, response times, and overall speed.
- **Accessibility**: Verify compliance with accessibility standards (e.g., WCAG).
- **Best Practices**: Ensure adherence to industry best practices for web development.

Performance Testing Tools

1. **Lighthouse**:
    - Lighthouse is an open-source tool by Google that audits web pages for performance, accessibility, and other best practices.
    - It generates scores and provides actionable recommendations for optimization.
    - We'll use Lighthouse to assess various performance metrics.
2. **GT Metrix**:
    - GT Metrix is a web performance testing tool that analyzes page load times and provides detailed reports.
    - It offers insights into performance bottlenecks, resource optimization, and overall website health.
    - We'll use GT Metrix to gain a comprehensive view of the website's performance.
3. **SpeedLab**:
    - SpeedLab is another performance testing tool that focuses on real-world user experiences.
    - It simulates various network conditions and device types to evaluate responsiveness.
    - We'll use SpeedLab to understand how the website performs under different scenarios.

By leveraging these tools, we aim to optimize the NASA website's performance, enhance user experience, and ensure stability even under heavy usage.

6. Test Types

In the NASA project, we will conduct the following 6 types of testing:

1. **Exploratory Testing**:
    - Exploratory testing involves checking the system on the fly without predefined test cases.
    - QA professionals note down ideas during test execution.
    - It allows for flexibility and adaptability during testing.
2. **Smoke Testing**:
    - Smoke testing determines whether the deployed software build is stable.
    - It confirms whether the QA team can proceed with further testing.
    - Minimal tests are run on each build to verify critical functionalities.
    - Also known as “Build Verification Testing” or “Confidence Testing.”
3. **GUI Testing**:
    - GUI testing focuses on the UI/UX aspects of the report.
    - It covers:
        - Report format
        - Look and feel
        - Error messages
        - Spelling mistakes
        - GUI guideline violations
4. **Functional Testing**:
    - Functional testing identifies unexpected behavior in the report.
    - Key characteristics:
        - Correctness
        - Reliability
        - Testability
        - Accuracy of report output/data
5. **Positive Testing**:
    - Positive testing involves providing valid data as input to the system.
    - It checks whether the application behaves as expected with positive inputs.
6. **Negative Testing**:
    - Negative testing ensures that the application handles unwanted input and user behavior.
    - Invalid data is intentionally inserted to validate error handling and robustness.

By conducting these various types of testing, we aim to thoroughly validate the NASA website’s functionality, usability, and resilience.

7. Test Strategy

7.1 QA Role in the Test Process

The QA team plays a crucial role in ensuring the quality of the NASA website. Here are the key responsibilities:

1. **Understanding Requirements**:
    - QA professionals thoroughly review the requirement specifications provided by the client.
    - They gain a deep understanding of the project’s objectives and functionalities.
2. **Preparing Test Cases**:
    - QA creates comprehensive test cases based on exploratory testing.
    - These test cases cover all possible scenarios related to the specified requirements.
3. **Preparing Test Matrix (RTM)**:
    - QA develops a Requirements Traceability Matrix (RTM) that maps test cases to specific requirements.
    - This ensures complete coverage of all requirements during testing.
4. **Reviewing Test Cases and Matrix**:
    - Peer reviews are conducted for both test cases and the RTM.
    - QA Lead and reviewers provide feedback on test coverage and accuracy.
5. **Creating Test Data**:
    - QA generates relevant test data based on scenarios and test cases.
    - This data is used for executing test scenarios.
6. **Executing Test Cases**:
    - QA executes test cases on the client’s development/test site.
    - They validate the functionality of the website against the designed scenarios and test data.
7. **Defect Logging and Reporting**:
    - QA logs defects/bugs found during test execution.
    - Defects are documented in Word documents and JIRA.
    - QA communicates identified issues to the respective developers.
8. **Retesting and Regression Testing**:
    - QA performs retesting for fixed bugs once they are resolved by developers.
    - Regression testing is conducted as needed to ensure no unintended side effects.
9. **Deployment/Delivery**:
    - Once all reported bugs are fixed, and no other critical issues are found, the report is deployed to the client’s test site.
    - QA ensures the stability and readiness of the website for further testing.

7.2 Bug Triage

All issues discovered during testing will be logged into JIRA for proper tracking and management.

7.2.1 Bug Life Cycle

The bug life cycle encompasses the stages a reported issue goes through from discovery to resolution. These stages typically include:

1. **New**: The bug is reported and enters the system.
2. **Assigned**: The bug is assigned to a developer or team member for investigation.
3. **In Progress**: The developer is actively working on fixing the bug.
4. **Fixed**: The bug has been resolved by the developer.
5. **Verified**: QA verifies that the bug is indeed fixed.
6. **Closed**: The bug is closed after successful verification.

7.2.2 Bug Severity and Priority Definition

Bug severity and priority are critical for effective bug management:

1. **Severity**:
   - **Critical**: The bug causes a system crash, data loss, or major functionality failure.
   - **High**: The bug impacts critical features but does not cause a system crash.
   - **Medium**: The bug affects non-critical features or has a workaround.
   - **Low**: The bug is minor and has minimal impact on functionality.
2. **Priority**:
   - **Immediate**: Requires immediate attention (e.g., security vulnerabilities).
   - **High**: Needs prompt resolution.
   - **Medium**: Requires attention but can wait for the next release.
   - **Low**: Can be addressed later.

Bug Triage Meetings

The QA Lead will organize bug triage meetings involving the QA team, development lead, and project manager. During these meetings, the priority of all active bugs will be assigned. The goal is to prioritize and address critical issues promptly.

7.2.3 Bug Severity List:

Severity ID		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 4	Mediu m Low	*Incorrect functiona*lity of component or process. There is a simple work around for the bug if it is **Severity 3.** 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

Before testing begins, the following conditions must be met:

1. **Test Hardware Platforms**:
   - All test hardware platforms (servers, devices, etc.) are successfully installed, configured, and functioning properly.
   - QA can operate the system and judge its correct behavior.
2. **Documentation and Requirements**:
   - All necessary documentation, design specifications, and requirements information are available.
   - QA can refer to these documents for accurate testing.
3. **Software Tools**:
   - All standard software tools, including testing tools, are installed and functioning correctly.
   - QA can use these tools effectively during testing.
4. **Test Data Availability**:
   - Proper test data is prepared and accessible.
   - QA can use this data for test scenarios.
5. **Test Environment Readiness**:
   - The test environment (lab, hardware, software) is set up and ready.
   - System administration support is available.
6. **QA Resources Preparedness**:
   - QA resources fully understand the requirements.
   - QA team members have strong knowledge of the system's functionality.
7. **Reviewed Test Artifacts**:
   - Test scenarios, test cases, and the Requirements Traceability Matrix (RTM) have been reviewed and approved.

8.2 Exit Criteria

The following conditions indicate the completion of testing:

1. **Requirements Coverage**:
   - A certain level of requirements coverage has been achieved through testing.
2. **Bug Status**:
   - No high-priority or severe bugs remain outstanding.
   - All critical issues have been addressed.
3. **High-Risk Areas Tested**:
   - All high-risk areas have undergone thorough testing.
   - Only minor residual risks remain outstanding.
4. **Budget and Schedule**:

    - Testing continues until the allocated budget is spent.
    - The project schedule has been met.

By adhering to these criteria, we ensure a systematic and well-managed testing process for the NASA website.

9. Suspension Criteria and Resumption Requirements

9.1 Suspension Criteria

Testing may be suspended under the following conditions:

1. **Serious Defects in the Build**:
    - If the build contains critical defects that significantly hinder or limit testing progress, testing may be suspended.
    - QA will assess the severity and impact of these defects.
2. **Significant Requirement Changes**:
    - If there are substantial changes to the requirements suggested by the client, testing may be temporarily halted.
    - QA needs to reevaluate the test approach based on the updated requirements.
3. **Software/Hardware Problems**:
    - Technical issues related to software or hardware (e.g., server downtime, infrastructure failures) may lead to suspension.
    - QA will wait until these problems are resolved.
4. **Resource Availability**:
    - If assigned resources (such as test environments, tools, or personnel) are unavailable when needed, testing may be suspended.
    - QA requires the necessary resources to proceed effectively.

9.2 Resumption Criteria

Testing will resume under the following conditions:

- Resumption occurs once the issues that caused the suspension have been resolved.
- QA ensures that the environment is stable and ready for testing.

By adhering to these criteria, we maintain a systematic approach to managing testing interruptions and ensuring timely resumption.

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, Selenium WebDriver
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, Selenium WebDriver, Google DevTools
Security Testing	

10. 2 Test Environment

The test environment will be set up as follows:

1. **Windows 10**:
    - **Chrome (Latest Version)**: Verify compatibility and functionality on the Chrome browser.
    - **Firefox (Latest Version)**: Test the website on the Firefox browser.
2. **Mac OS**:
    - **Chrome (Latest Version)**: Validate cross-platform compatibility.
    - **Safari (Latest Version)**: Test responsiveness and functionality on Safari.
3. **< Android**:
    - **Chrome (Latest Version)**: Ensure the website works seamlessly on Android devices.
    - **Firefox (Latest Version)**: Validate compatibility on the Firefox browser for Android. >

11. Test Schedule

Task Name	**Start**	**Finish**	**Effort**	**Comments**

12. Approvals

	Project Manager	**QA Lead**
Name		
Signature		

13. Terms/Acronyms

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