Test Strategy

History

Version	Date	Rectificare
1	Nov 20, 2023	
	19/12/2023	Adina’s edits

1. Introduction

1.1 **Project Overview**

"Peviitor.ro" is a dynamic job search engine that aggregates jobs through scrapers, with core functionality provided by Apache SOLR via an API.

The user interface is built using elementary technologies.

2. Testing Overview

2.1 **Test Levels**

Component Testing:

- Focus on individual components: UI, API, SOLR, Scrapers, and CloudFlare.
- Test independently before integration.
- Manual testing.
- Automation testing.

Integration Testing:

- Test the integration of UI, API, and SOLR.
- Include end-to-end testing, data testing, and pipeline testing.
- Manual testing.
- Automation testing.

System Testing:

- Comprehensive testing of the entire system.
- Include performance, reliability, security, accessibility, usability, cross-browser, mobile, caching, data, and scraper tests.
- Manual testing
- Automation testing

UAT (User Acceptance Testing):

- Validate usability and gather user feedback.
- Ensure user satisfaction with the system.

- Manual testing

2.2 **Testing Activities**

Test Planning:

- planning how we will test by creating a test plan
- defining the objectives and the abordation
- it'll be decided which parts of the application are to be tested
- assign roles to people involved in the project
- the entry and exit criteria are defined

Test Analysis:

- Occurs at the beginning of each sprint.
- Estimation of testing effort.
- Identification of test scenarios for each story.
- analyze of the documentation (requirements)
- generate the test conditions

Test Design:

- Define test scenarios and test cases.
- Identify test data.
- Create the test plan for each sprint/iteration.
- Group test cases in a Test Management tool.
- Generate the Test Cases Report.
- creating the test environment

Test Implementation

- creating Test data
- creating Test suites
- prioritization of test cases by developer lead based on importance about business and risks
- grouping the tests based on their objectives (functional testing, regression testing, UAT testing)

Monitoring and control

An ongoing activity carried out with the purpose of comparing the current progress with the Test Plan.

Test Data

Test Execution and Test Completion (Reporting):

- the results are reported in the tool in which the tests were written
- reporting discrepancies as incidents
- the exit criteria are evaluated
- Execute the test plan.
- Log bugs in the corresponding scraper’s GitHub Issues section .
- Generate the Test Report.

- Share results with the development team.

Retesting:

- Retesting all the fixed bugs, for functionality validation

Regression Testing:

- Validate each build.
- Run test cases for a specific build.
- Declare a build as a Release Candidate based on exit criteria.

Test Closure:

- Document Test Closure for each Release Candidate.
- Include known issues and results for each type of testing.
- handover of testware

Status Report:

- Daily report at the end of testing activities.
- Includes new, closed, and reopened bugs, test case status, and a centralized report on the testing process.
- Evaluation of exit criteria and generating a closure reporting

2.3 **Scope of Testing**

The scope of testing is to check the functionality of the website PE VIITOR.

- improving the performance of the website by finding defects and preventing them
- product quality assurance
- increasing confidence in the quality of the website
- reporting defects to be repaired or eliminated

In Scope:

- Functional Testing
- GUI testing
- Component testing
- Sanity testing
- Regresion testing
- Exploratory testing
- Automated Testing
- Compatibility testing
- Usability testing
- Performance Testing
- Security Testing
- Cross-Browser Testing, Responsive design testing
    - Google Chrome, Mozilla Firefox, Safari, Microsoft Edge
- Responsive Design testing
    - IOS, Android, Tablet
- SEO (Optimizing website for search engines)
- API testing
- Unit testing
- Data testing
- Accessibility testing

- Recovery from disaster (involves planning and implementing strategies to restore data and systems to normal operation after a disruptive event, ex.: regular and automated backups, cloud based solution, offline backups)

Out of Scope:

- Localization testing
- stress Testing
- anything not explicitly mentioned in the original scope document

2.4 **Entry Criteria**

- Verify if the test data is available and validated.
- Requirements are defined and approved.
- The test environment is set up.
- The tools and devices are available.

2.5 **Exit Criteria**

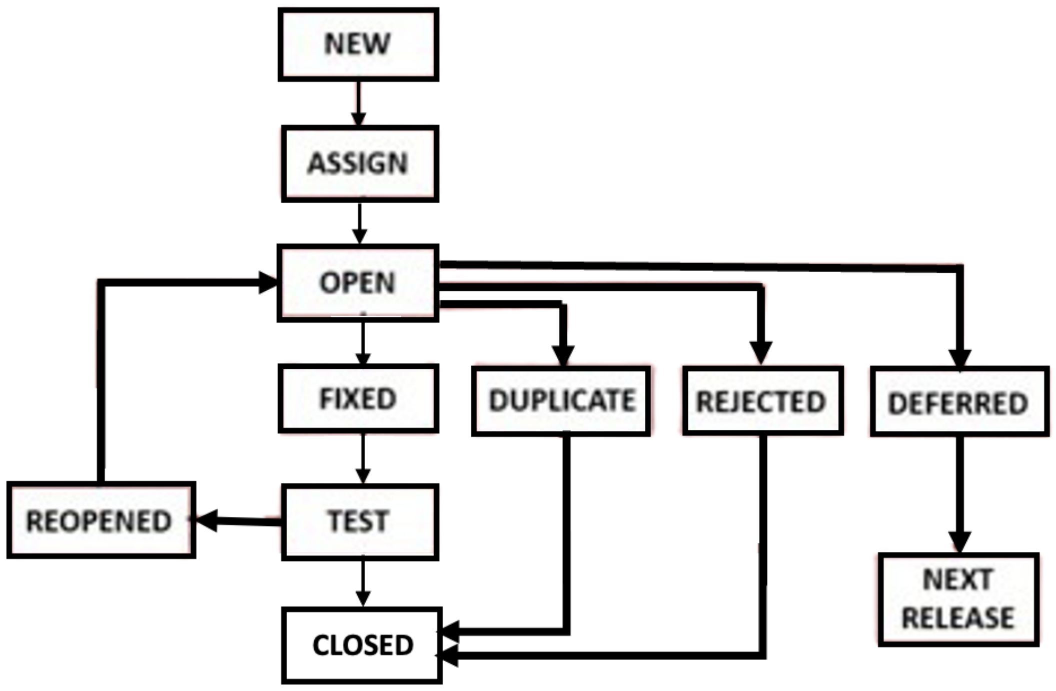
- 97% Test Cases run
- 95% Test Cases PASS
- No open bugs with HIGH severity
- 95% of bugs fixed & closed with low severity

2.6 **Test results capture**

During Test Execution, the GitHub Project will be used to capture all test results.

Should the test case fail, the status will be changed to “Failed”. Any divergence between expected and actual results will be regarded as an issue. The issue will be raised to the corresponding scraper’s GitHub <Issues> section and have its label set to “bug”, after describing it using the bug template.

Bug Lifecycle



New:
- A bug has been identified.

Assign:
- A person is assigned to fix the bug.

Open:
- The bug is in the queue to be implemented or worked on.

Duplicate:
- The bug is investigated, and it's found to be the same as another bug.

Rejected:
- The bug is considered not important or not affecting the user’s experience.

Deferred:
- The bug is postponed and scheduled for fixing in a future release.

Fixed:
- The developer has fixed the bug.

Retesting:
- The fixed bug is being tested to ensure the issue is resolved.

Reopened:
- If the bug persists or reoccurs after being marked as fixed.

Closed:
- The bug is considered fixed, and the issue is closed.

Bug Severity

High Severity:

- Critical functionality is broken.
- Security vulnerabilities.
- Major performance issues affecting usability.
- Data loss or corruption.
- The application cannot be used and we will stop until it is fixed

Medium Severity:

- Important features are not working correctly.
- Performance issues that do not severely impact usability.
- Non-critical security issues.

Low Severity:

- Minor issues that do not impact critical functionality.
- Cosmetic issues affecting the user interface.
- Low-priority performance improvements.
- Misspelled words

2.7 **Test team organization**

Laurentiu	1. Adriana Cioaca 2. Adina Boeru 3. Alexandra Nicoara 4. Dana Catavei 5. Emese Egeto 6. Adi Vanca 7. Ana Maria Lazar
Cristi Olteanu	1. Gariel Negru 2. Adelina Guliman 3. Diana Str 4. Adrian Sandu 5. Iulian Benchea 6. Teodor Csordas
Andrei Cojocaru	1. Laura Scutariu 2. Bogdan Scutariu
Rares	1. Danut Bese
Mircea(Grasum)	1. Fulger Irina 2. Ale Dana 3. Pravat Mihai
Melania	1. Larisa 2. Gabriela
David Mondoc	1. Emese
Cristina Anghelet	2. David Mondoc 3. Veronica 4. Luiza

3. Environments

3.1 **DEV Environment**

- Developer's local machine.
- Each developer should set up SOLR, API, and UI locally.

3.2 **TEST Environment**

- https://test.peviitor.ro/
- Validate scraper data.

3.3 **PRODUCTION Environment**

- https://peviitor.ro/
- API: https://api.peviitor.ro/
- SOLR: https://solr.peviitor.ro/

4. Tools

- POSTMAN
- Apache JMeter
- GitHub Test Cases
- GitHub Issues
- Figma Designs
- GitHub Actions
- GitHub Desktop

5. Deliverables

- **Test Strategy**
  A test strategy is an outline that describes the testing approach of the software development cycle. The purpose of a test strategy is to provide a rational deduction from organizational, high-level objectives to actual test activities to meet those objectives.

- **Test Plan**
  A test plan is a document detailing the objectives, resources, and processes for a specific test session for a software or hardware product. The plan typically contains a detailed understanding of the eventual workflow.

- **Test Case**
  A test case contains the actions required to verify a specific feature or functionality in software testing. It details the description, preconditions, test steps in which are also known as the sequence of actions, expected outcomes, and post-conditions.

- **Test Report**
  A Test Report provides a comprehensive overview of the testing activities conducted during a specific phase of the software development lifecycle. It summarizes the test results, highlights key findings, and assesses the overall quality of the software.

- **Status Report**
  A Status Report offers a snapshot of the current state of a project, highlighting progress, challenges, and upcoming activities. It is a valuable communication tool for keeping stakeholders informed about project health.

- **Bug Report**
  A Bug Report, also known as a Defect Report, details information about a discovered issue or bug in the software. It is essential for communication between testers and developers, providing clear and structured information about the defect.