Todo.ly Public API tests

Functional test plan

Table of contents

[Revision History 1](#_Toc504468218)

[Introduction 2](#_Toc504468219)

[Scope 2](#_Toc504468220)

[Manual testing 2](#_Toc504468221)

[Automated testing 2](#_Toc504468222)

[Out of scope 2](#_Toc504468223)

[Test environment 2](#_Toc504468224)

[Test server 2](#_Toc504468225)

[Client configuration 2](#_Toc504468226)

[Test approach 3](#_Toc504468227)

[Tools 3](#_Toc504468228)

[Deliverables 3](#_Toc504468229)

[Assumptions 4](#_Toc504468230)

[Risks 4](#_Toc504468231)

## Revision History

| Rev. | Date | Author | Reason for Changes |
| --- | --- | --- | --- |
| 1 | 2019-03-21 | Diego Bruno | First draft |
|  |  |  |  |
|  |  |  |  |

## Introduction

The purpose of this document is to define scope, tools and strategy to carry out functional testing of the Todo.ly public API.

## Scope

Exploratory manual tests will be executed against some of the API endpoint with the goal of understanding the available functionalities a possible scenarios. Automation of the most critical functions will be implemented immediately after.

Due to time limitations, in-scope endpoints and functionalities will be limited to:

* User API
* Authentication API
* Projects API (partial)
  + Get all projects
  + Get project by Id
  + Delete project by Id
* Error Codes (partial)

As required by the assignment document, authentication scenarios were targeted first. This resulted in defects being found early on. See Defects file in Deliverables section for more details.

## Out of scope

All other APIs not explicitly mentioned as in-scope will be considered out of scope.

## Test environment

### Test server

The following URL will be considered to be the test environment:

https://todo.ly/api/

## Test approach

Functional tests of API will be executed according to the task specification received in the file "Test Case for Automation QA Engineer.pdf".

The goal of these tests is to demonstrate the QA Engineer’s ability to:

* test an API
* identify important functionalities
* find and report bugs
* automate test cases

### Exploratory

A first round of manual exploratory test is executed using Postman. The goal of this initial approach is to better understand the behavior of the API and to discover potential bugs.

As a way to have a more diverse test coverage, the manual exploratory test was performed with XML format requests while the automated tests focus on JSON format.

Defect reports resulting from of this execution will be available as specified in the Deliverables section of this document.

Due to time constraints not all of the exploratory effort was translated into automated tests. As it can be seen in the Defects report, there is a defect that has no automated test coverage at this time.

### Test automation

REST-assured is used as the preferred automation framework because of being a platform independent library available under Apache License.

Java is the preferred language because it leverages the experience of the test team. TestNG is used to handle the automated test suite execution and basic report generation. All of these automation technologies can be seamlessly added to a Continuous Integration pipeline.

The test suite can be executed as a Maven project from command line. Notice that the log level is high so that all requests and responses are visible in the console during test execution.

The test suite is intended to be an example. The CreateUser api method has a higher quantity of tests in an attempt to demonstrate some boundary testing over some fields. Other api methods have less quantity of tests, however they are covered by high value tests which in many cases are triggering bugs (see Defects section).

## Tools

The following tools will be utilized to perform all the test activities during this project.

|  |  |  |  |
| --- | --- | --- | --- |
| Activity | Tool | Vendor | Version |
| Test Plan | Word | Microsoft | N/A |
| Defect reporting | Excel | Microsoft | N/A |
| Exploratory testing | Postman |  | 7.0.6 |
| Test automation | Java  REST-assured  Maven  TestNG |  | 1.8  3.3.0  3.5.3  6.11 |

## Deliverables

|  |  |  |
| --- | --- | --- |
| Deliverable | Contents | Location or file name |
| Test plan document (this file) | Text document describing decisions | /docs folder |
| Defect report | Details and evidence about defects found during testing | See file “Defects – TodoLy.xlsx  ” in the docs folder. Evidence files can be found under the docs/evidence folder. |
| Automated test cases | Files required to execute the automated tests | See "README.md" file in the GitHub repo for instructions. |
| Postman collection | Postman was only used as a manual exploratory testing tool. However, the collection is provided only as a reference and should be considered as a draft. | See “docs/postman-exploratory” folder. |
| Video of tests running | Video showing an execution of the complete test suite, including the tests that fail due to bugs found. | See file “Todo\_ly suite including failing tests due to bugs.mp4” in docs/evidence |

## Assumptions

The Test Server environment is properly configured, fully functional and accessible to the test team.

No new code deployments will occur during test activities. The database structure and the behavior of the application will not change during test activities.

## Risks

Test coverage is limited due to time and personnel availability constraints. Only the most essential features will be tested (see In-scope section).

The test server is exposed to unrestricted public use and 3rd party actions. Additional verification or multiple test passes are recommended as a means to mitigate this issue.

Because of being a production environment, the database is not being cleaned after test execution. Due to not having full control over the text environment, pre and post test activities have been ignored.