Test Plan for

BUDGETING APPLICATION

Prepared by Lionel Yarboi

19 May 2020

This document outlines the testing effort I will undertake in regards to the budgeting application.

Approval

AUTHOR			
Lionel Yarboi	Automation		
APPROVED BY			
ACKNOWLEDGMENT			

Glossary

Term	Definition
Bug	See Defect
Defect	Software function does not work as per specification
Defect Owner	The person who fixes the defect
Issue	Software function does not work as expected or is not specified
BDD	Behaviour Driven Development
SDLC	Software Development Life Cycle
SME	Subject Matter Expert
TDD	Test Driven Development
UAT	User Acceptance Testing
WCAG	Web Content Accessibility Guidelines

Contents

Introduction

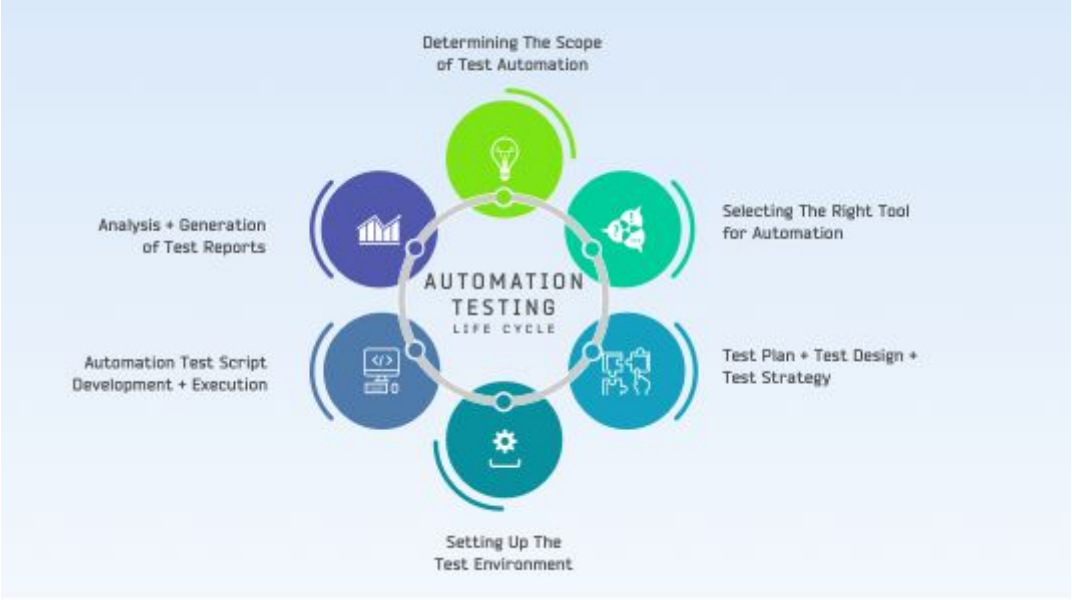
Purpose

This document is a high level presentation of the test approach to be undertaken in relation to the budgeting application. It will outline and create a shared understanding of the overall targeted approach, tools and timing of activities.

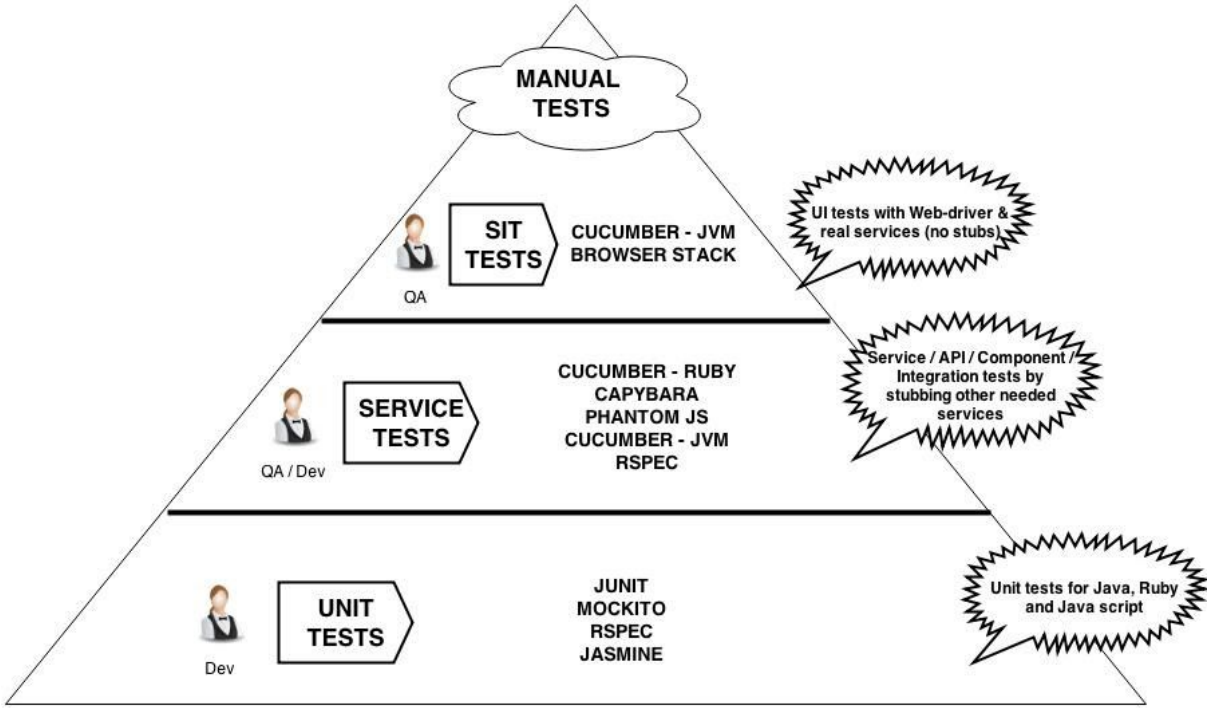
After reading this document, the reader will have a good understanding of:

- The objectives of QA in regards to the Budgeting Application
- Test activities in different stages of the Software lifecycle
- The different test levels and test types
- Entry and exit criteria including reporting
- Creating and maintaining test artefacts
- Key testing processes and procedures

This test plan will include all stages of the automation testing life cycle..



Automation objectives



In reference to the current Agile pyramid, I have assumed the application does not have unit tests, meaning most of our testing will be conducted in the service layer.

Automation Technologies

Our tests will consist of End to end automation flows, testing all user journeys, using the following technologies:

Cucumber: BDD tool which runs automated acceptance tests written in gherkin style format

Site Prism: Page Object Model pattern tool, used to map our page elements into readable classes

Capybara: Acceptance test framework allowing us to simulate across browsers how a real user would interact with our application.

BrowserStack: Cross browser testing against 2000+ real devices and browsers

CI server(e.g Jenkins): Automatically run tests on developer commit or periodically throughout the day.

What will be tested

The following functionality will be covered by automation tests/manual testing which I will outline below, this will give us a high percentage of functional test coverage.

1. Updating of categories
2. Updating/Deletion of description
3. Updating/Deletion of amount
4. Checking Description with special characters
5. Checking amounts with special characters and alphanumeric characters
6. Checking application can handle duplicate entries
7. Checking total working balance displayed on correct page
8. Checking reports tab link directs user to correct page
9. Checking legend on reports page displays all categories with amounts
10. Adding new inflow/outflow includes (category, description, amount)
11. Checking correctness of total working balance
12. Checking correctness of Inflow vs Outflow charts categories/amounts
13. Checking spending in category displays the correct amounts

Following a discussion with key stakeholders or through analytics a Browser/Mobile matrix will be created, this will drive which devices and browsers will be tested, if this isn't available the latest device/browser and known troublesome browsers (i.e IE 11) will be tested against.

- Chrome
- Firefox
- IE(BrowserStack)
- iPhone simulator

What will not be automated

As a responsive web application, manual verification across devices will be carried out(compatibility testing), to ensure all elements display correctly on various browsers/viewports.

Browserstack will be used, to run tests against a wide range of browsers/devices, this will be implemented, once we have setup automation tests for key functional areas:

- Checking correctness of total working balance
- Checking correctness of Inflow vs Outflow charts categories/amounts
- Checking spending in category displays the correct amounts

Test Environment

We will need the following environments to ensure, we cover testing on all levels:

Mock environment - I will need a mock test environment, where I have full control over data creation. The environment will need to be stable, up during working day minimum. I will be running most of my tests in this environment.

Staging environment(Integration)- This will be my integration environment, where the UI communicates with a production like backend. I will be running a subset of my tests here a few times a day.

Production environment - A very small amount of tests will be running here, tests will mostly be smoke tests, meaning quick validation checks.

Test Phases/Levels

Test levels are used to split testing phase into logical phases and set clear boundaries for each level of testing.

- Component testing
- System testing
- User Acceptance Testing (UAT)
- Non Functional Testing(NFR)

Component testing relates to the testing of individual features.

System testing is the behaviour of the whole system for given scenarios. Example a click to add description. This type of testing will be applied in an environment that mirrors production as closely as possible. This is an Automation task.

The business user takes part in UAT to establish full confidence in the system. The Business user may be provided a subset of tests this will need to be outlined beforehand.

Finally Non Functional Testing which includes Performance testing. Creating virtual users who take journeys on the application to ensure that the platform can provide the optimal user experience inline with the defined limits.

Test Types	Component Testing	System Testing	UAT
Unit test	Yes		
Functional test		Yes	Yes
Non-functional test (Performance, Accessibility testing)		Yes	Yes
Regression test	Yes	Yes	Yes

Table 1 - Test levels and test types

Definitions:

Unit Testing - Individual parts of application code, is tested to confirm code is fit for use.

Functional Testing- Based on what the system does, in relation to specification.

Non-Functional Testing- Testing the way the system operates.

Regression Testing - Testing functional areas of the system, after enhancements or configuration changes.

Pass/fail criteria

Automation tests will be developed as a part of every feature. The tests scripts will contain test steps, expected results, pass criteria and fail criteria.

Broadly speaking, deviation from what is written in a functional / technical specification will be considered a defect. A defect might not always result in failure.

Where a defect is identified, a defect will be raised in a bug tracking system, or whatever process has been defined.

Entry Criteria

The following conditions are preferred before any testing can begin:

- Business requirement documentation has been outlined
- Segments of development code are unit tested before being released into a test environment
- All Test environment requirements must be defined
- Test environment must be ready and accessible
- Actions for known defects or limitations are finalised and assigned to the correct area

Exit Criteria

These are the minimum acceptable conditions before promoting the application to a live production environment:

- There are no outstanding high priority issues unless the Business Owner has provided a waiver along with a timescale for resolution
- All Integration *P1 test cases Manual/Automated have been successfully executed
- Passing end to end cucumber tests with reporting
- During the course of the testing, certain tests (and by extension certain pieces of functionality) may not be tested due to project's constraint. If tests cannot be executed, approval will be sought from the relevant stakeholders and this will be recorded, as well as the reasons why.
- BA or Product Owner have reviewed defects and provided signoff

* P1 test cases are classed as core functional parts of the application.

Project Conditions

Assumptions

The following assumptions have been made in preparing this test strategy:

- The test environment is a stable environment and will be available at the start of the testing period
- All business requirements are documented and incorporated into bug tracking system which form the test base prior to the start of testing
- Browser matrix devices have been provided and ready to use.

Constraints

The ability for the test team to deliver appropriate test coverage is contingent upon the following factors.

- The project scope remains relatively stable
- I am not notified promptly of any changes in project scope so the impacts on testing can be properly assessed
- Development work is progressed and delivered within timeframes that enable full and detailed testing to take place inline with (Proposed Testing Approach)

Out of Scope

Following items are deemed out of scope for QA team:

- Security Testing

Non Functional Requirements

Where required non functional testing will be conducted, this is a requirement where I judge the operation of the system, rather than the system behaviour.

The types of Non Functional Testing I will undertake if required.

Accessibility Testing

Take into consideration the experience a user with a disability is presented according to WCAG guidelines.

Software:

Native browser screen reader

Performance Testing

Determine how well the application operates in terms of responsiveness and stability under various workloads.

The following test types will be run if required:

Load Testing

Load testing simulates multi users or multi threaded access to an application or module to ensure components and databases can still be used under different boundaries.

Under the following conditions:

1. Based on requirements, pages load within set boundaries.
2. Representative data load based on requirements.
3. Measuring response times while uploading.
4. Measuring response times while releasing.

Software:

JMeter

Risk assessment

Testing risk assessment is used to prioritise Manual/Automated testing based on the following criteria:

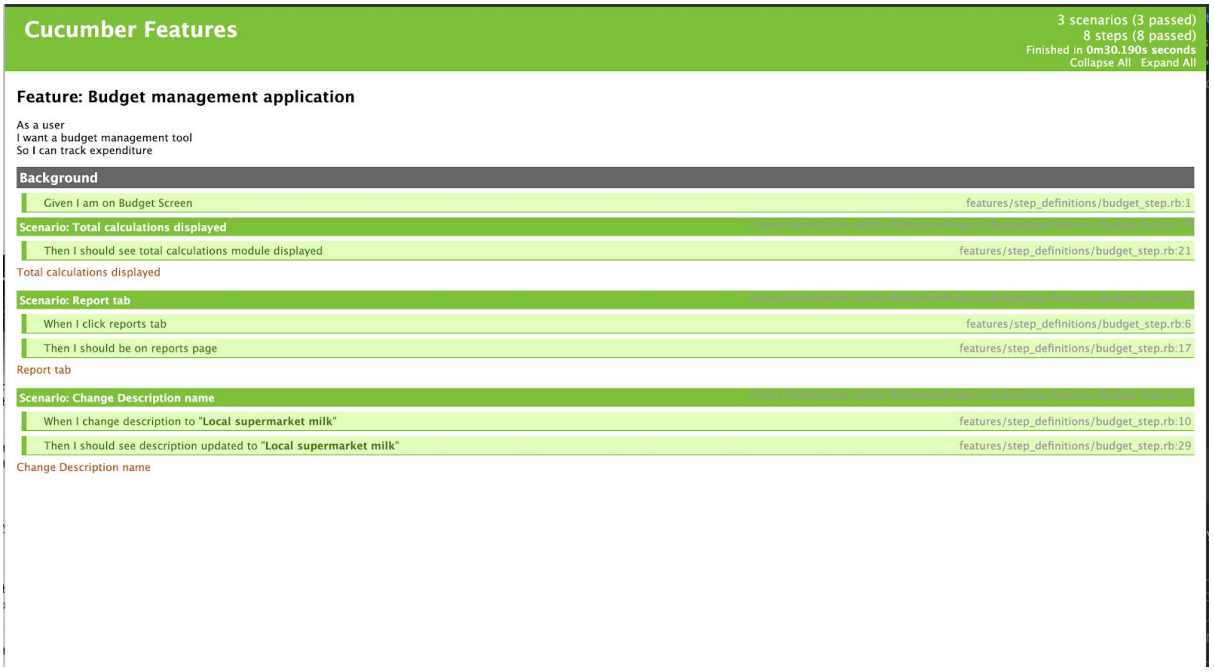
1. The prior classification of business requirements
2. The likelihood of code changes or new functionality affecting existing functionality
3. Areas where Business Analyst/Editorial team have deemed core user journey.

Test Reporting

Progress reports will be emailed to relevant stakeholders or created within an agreed space, at the end of each Integration Testing phase outlining:

- Completed tests
- Summary of defects raised(if any)
- Summary of defects fixed
- Outstanding defects for next sprint including their classification (Major(P1), Medium(P2) etc.)

Example report:



Defect Management

The detection, management and resolution of defects will be properly recorded and progressed using a bug tracking system

All defects must be assessed to determine a severity level as described in the table below.

Severity Level	Severity Description
Blocker/Critical (P1) *	Functionally unusable
Major (P2)*	Functions can partially be used
Trivial (P3)*	Core functionality is not impacted