**Test Plan**: Best Buy Health Customer Contact information updates

**Prepared by**: Hemadri Dasari

**Date**: 01/09/2021

**TABLE OF CONTENTS**

1.0 INTRODUCTION

2.0 OBJECTIVES AND TASKS

3.0 Architecture diagram

4.0 SCOPE

5.0 TESTING STRATEGY

6.0 ENVIRONMENT REQUIREMENTS

7.0 TEST SCHEDULE

8.0 CONTROL PROCEDURES

9.0 FEATURES TO BE TESTED and high level Test Cases

10.0 FEATURES NOT TO BE TESTED

11.0 RESOURCES/ROLES & RESPONSIBILITIES

12.0 Deliverables

13.0 RISKS/ASSUMPTIONS

14.0 TOOLS

15.0 APPROVALS

**APPENDIX:**

1.0 Automation Approach

2.0 Test Metrics

3.0 Defect Definitions

**1.0 INTRODUCTION**

This test plan is for testing changes to Contact information UI screen used by customer service representative to provide fanatical support and for customers to update their contact information.

**2.0 OBJECTIVES AND TASKS**

**2.1 Objectives**

Define plan to test acceptance criteria for the below user stories:

**User story 1:**

As a customer service representative, I want to track new customers so that I can provide greater customer experience and grow my client base.

*Acceptance criteria:*

* Add a section to billing address for phone number
* The billing address section:
  + Phone Number
    - A customer phone number can be used by our customer representative
  + Tracking these new customers
    - A new customer could be called to confirm an order
* Only numeric characters will be accepted
* When a new customer billing address is added, a phone number verification will occur for a valid area code.
* Phone number will be required

**User story 2:**

As a customer, I would like my address information previously entered available to be entered in my address, city, name, phone number, zipcode.

*Acceptance criteria:*

* A customer must be created with the following fields:
  + first name
  + last name
  + address
  + city
  + state
  + zipcode
  + phone number
  + email address
* A customer that does not contain required information will display an error message underneath the field
* A customer that has a country of United States of America will see a different field
  + A customer will have a zip code vs postal code
  + A customer will be able to select a “STATE” from a dropdown list
* A customer that has a country other than United States of America will see different fields
  + A customer will have a postal code vs zip code
  + A customer will need to type in a “STATE"
* A customer that has a country of Canada will see different fields
  + A customer will have a postal code vs zip code
  + A customer will be able to select a “STATE” from a dropdown list
* A customer may have a tag added to it consisting of at least two alphanumeric characters
  + A contact will only have tags saved if the contact is saved

**2.2 Tasks**

* Clarifications on User Stories
* Testing
* Post-testing
* problem reporting

**3.0 Architecture diagram** showing al components that are part of E2E flow

**4.0 SCOPE**

Test the UI page for retrieving, saving and updating customer contact information used by CSR and Customers.

**Tactics:**

Identify the test data requirements, test cases, testing framework and create automated test script with test scenarios that can later be incorporated into existing regression test suite and integrated into CICD pipeline.

**5.0 TESTING STRATEGY**

Perform functional and performance testing of the changes and document entry/exit criteria for the same.

**5.1 Unit and Mutation Testing:**

Test internal logic and verify code design, Test branches, conditions, and paths in the code & Test exception conditions.

Participants: Development team.

Methodology: It is assumed that developers perform this testing (preferably using TDD Red-Green-Refactor approach) with a code coverage of 80% or more.

**5.2 Functional Testing**

Definition: Perform functional testing. Based on the testing time available, we may need to prioritize test cases using Equivalence Partitioning technique for the test data instead of exhaustive testing to deliver high quality with limited amount of time. Validate all acceptance criteria for individual user stories

Participants: SQA Engineer

Methodology: Identify proper data sets, testing framework and identify the web elements and perform the actions that a customer or customer service representative would perform and validate the actual result matches expected result for each set of test data that is part of data driven testing.

**Exit Criteria:**

For QE to sign off on a release, the following criteria must be met:

No open severity 1 or 2 issues (Defect Priority and Severity Definitions)

Deferred issues list is reviewed and signed off by QE, UAT, Business and support.

Verify all deferred customer impacting issues and the verified potential workarounds are documented.

All resolved issues are verified and closed.

**5.3 Performance and Stress Testing**

Definition: Perform API testing to measure the response is within the defined SLO.

Participants: SQA Engineer

Methodology: Perform a short duration continuous performance testing to validate the response time is under 200ms and returns a HTTP sttaus code of 200.

**5.4 User Acceptance Testing**

Definition: The purpose of acceptance test is to confirm that the system is ready for operational use.

During acceptance test, end-users (customers) of the system compare the system to its initial requirements.

Participants: Business stakeholder (eg: Product owner or Customer Service representative or Business analyst etc)

Methodology: This will take place in production to ensure software meets all the acceptance criteria and does not break other existing functionality.

**5.5 Automated Regression Testing**

Definition:

Regression testing is the selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still works as specified in the requirements.

Participants: SQA engineer

**6.0 ENVIRONMENT REQUIREMENTS**

Testing environments (Dev Sandbox, Test, Staging, Pre-prod, Prod etc) and a smoke test to validate that the environment is healthy.

**7.0 TEST SCHEDULE**

Depends on the planning done during program increment or iteration planning and agreed upon by all the owners of dependencies to have all components available as required by E2E flow. Need to plan time for each phase of testing as per the team commitments.

**8.0 CONTROL PROCEDURES**

**Problem Reporting:** All defects will be logged and triaged as per the test strategy defined and adhering to best practices defined. Testing will be signed off by SQA manager after discussing any remaining defects with business and agreed upon the acceptable workarounds in place and have no Critical or Major defects.

**9.0 FEATURES TO BE TESTED and high-level Test Cases**

Identify all software features and combinations of software features that will be tested.

*Questions to ask related to Phone Number during grooming meeting for User story 1:*

* Does it need to support international numbers. If yes, What is the length of the field?
* Where does this persist and what DDL rules (type & length) are applied for this field?
* Will the numeric characters allow spaces in between? I am assuming there is no space when entering numeric characters?
* Will the UI Auto-format the input with geolocation and show a country flag as a cue to the CSR for better Customer Experience?
* What is the expected behavior when invalid input is provided?
* Will it trim any spaces on the left or right of the numeric value?

**Test cases for User Story 1 (Phone Number Field):**

Assumption: Country code is not required and numeric field size is 10 digits and only Valid area codes are allowed

**Positive Test Cases**: Create a table of valid area codes and test it iterating through it and validate that there is no error and saved properly

Create a table based on <https://nationalnanpa.com/nanp1/npa_report.csv> with columns NPA\_ID, type\_of\_code, COUNTRY, IN\_SERVICE and test that application accepts all ten digit numerical values with first three digits match the are code field NPA\_ID and “Y” for IN\_SERVICE.

**Negative Test Cases**: Enter the following inputs and validate proper error message is displayed and the data entered is not persisted - (a)Invalid area code (b)More than allowed size of the field (c)Negative Number (d)Decimal Number (e)Alphanumeric characters (f) punctuation characters (g)space/blank entry in the first position (h)space/blank entry in between numeric values (i) space/blank entry in the end position

*Questions to ask related to Phone Number during grooming meeting for User story 2:*

* Is this for a screen where customer can update previously entered contact information so that they can update them?
* Will it auto populate the following fields that are previously entered and saved : first name, last name, address, city, zipcode, phone number?
* Will it not auto populate state & "email address" fields if they were already entered previously?
* What are the rules behind acceptable values for all the mandatory fields(first name,last name, address, city, state, zipcode, phone number, email address)? Need to know the length, characters allowed and restrictions, eg:  email with multiple continues dot not accepted, Trailing dot in address is not allowed etc.
* What is the error message displayed underneath for each of the fields when customer does not enter a value before clicking submit or go to next screen?
* How does the system know the country value as it is not available as a field in the previous screenshot?
* If we are retrieving previously saved information so that customers can update the contact information, why would they all not have a tag of at least at least two alphanumeric characters?
* I am assuming customer has a choice to provide this optional tag value whenever they retrieve and update their contact information.

**Test cases for User Story 2:**

Prepare Test data for customers from different states in US and Canada with all possible combinations of positive and negative values along with expected result.

Use the following techniques in creating a dataset that provides a good test coverage: Equivalence Class Partitioning (a sub set of valid states, zip code and area codes and invalid characters) and Boundary Value Analysis (less than minimum length, minimum length, maximum length and more than maximum length) and leading and trailing spaces to narrow the dataset.

Iterate through the above values as part of data driven testing and ensure actual result matches with expected result and appropriate "postal code vs zip code" and Valid States are available on the UI screen.

**10.0 FEATURES NOT TO BE TESTED**

Other features that are part of the E2E flow of member signing up for a service in the shopping cart experience will be out of scope for this test plan. Separate effort outside the current sprint will be needed to update the E2E flow that is part of the regression suite to include these changes.

**11.0 RESOURCES/ROLES & RESPONSIBILITIES**

Developer – Complete development and Unit Testing, Defect Triage and Fix defects.

SQA Engineer: Prepare test plan, test scripts, execute test cases, defect triage and update test scripts, provide summary report.

Product Owner : Accept Story demos and perform UAT testing to approve code to be pushed to Production.

**12.0 Deliverables**

* Test Plan
* Detailed Test Cases with steps
* Test Incident Reports
* Test Summary Reports with test metrics

**13.0 RISKS/ASSUMPTIONS**

* Stability of the environment
* Development completed unit testing and met the exit criteria
* Resource avalability and any other high priorities
* Delays in turn around time for bug fixes
* Scope Creep

**14.0 TOOLS**

* Test Data Management
* Test Lifecycle Management
* Defect Lifecycle Management
* Agile Lifecycle Management
* Test Automation Framework/Tools for functional automation
* Performance Testing & Monitoring Tools
* Security Testing Tools
* Accessibility Testing Tools

**15.0 APPROVALS**

Specify the names and titles of all persons who must approve this plan. Provide space for

the signatures and dates.

Name (In Capital Letters) Signature Date

1.

2.

3.

4.

**APPENDIX:**

**Sections from Test Strategy:**

**1.0 Automation Approach**

The Automation framework needs support UI, API functional and performance testing.

**1.1 Data Setup**

We will use data driven testing with static data for the two user scenarios we are testing.

For other complex user stories, we may have to use TDM API calls for each execution cycle and design appropriate TDM methodology to cover Data Masking, Data Sub-Setting, Data Generation and have appropriate Test Data Governance & Metrics and techniques like Data Mining or Data Virtualization.

**1.2 Script Creation**

A Test Driven Development (TDD) approach needs to be followed where the development team creates Unit and Iteration (Functional) tests. Functional test scripts and reusable libraries are used by System Integration test teams to build the end-to-end integration scripts. Tests must be created with self-explanatory names and steps that can explain user actions. Tests should be logically grouped into test suites so that specific functionality can be tested in regression.

Coding standard should be defined and followed with appropriate comments for each logical block in test.

Automated scripts need to be checked in to GitHub repo after successful execution in local environment and are executed either through CI/CD pipeline or on demand run.

Exported API tests also needs to be checked-in to GitHub and integrated to pipeline.

**1.3 Review Process**

Any new script needs to be reviewed by a peer as a GitHub change request before merging to the main branch. To handle multiple version of code, a good branching strategy must be defined.

Any update to reusable code needs to be discussed with author or team owning the feature to understand overall impact. The impact of the change might be validated in a regression cycle. Code review must include coding standards check, coverage and local execution results.

**1.4 Execution**

Unit and Functional Smoke Tests are expected to run as part of CI/CD pipeline to indicate failures early in the cycle. Iteration tests and Integration tests need to have an on-demand execution option to run as part of any regression cycle.

Test suites need to have configuration option to run in different environment, browsers and devices in parallel or as a single threaded execution with minimal changes.

**1.5 Test Result Reporting**

Execution results must be updated in test management tool post execution or reports should be shared with stakeholders in agreed method of communication. Historical results and evidence must be maintained as per organizational guideline to measure coverage and effectiveness of automation.

**1.6 Defect Reporting for Automation Failures**

Automation framework can be integrated with Jira to report defects for execution failures in functional and integration tests. Defects must be created under a Jira user story relevant to the module. The framework should be able to capture screenshots for UI tests, request and response for API tests which can be added to the defect for further analysis.

Automation defects can be named with a specific pattern for easy identification. This will help in triaging and generating metrics.

**2.0 Test Metrics**

Below key metrics will be baselined and captured. In addition, we will work with key stakeholders to understand any additional metrics to be captured.

**Quality Metrics:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Metric | Objective | Calculation | Frequency | Test Phase |
| Execution Percentage and Pass rate | Provides execution status | No of test cases executed/ Total No of test cases planned  No of test cases passed/ Total No of test cases executed | Daily | SIT, E2E |
| Error Discovery Rate | Provides the rate of defects identified vs test cases executed | Total No of valid defects/Total number of test cases executed | Daily | SIT, E2E |
| Defect Re-Open Rate | Provides the Defects Re-open rate | Total Count of Reopened Defects/(Total number of Resolved Defects + Total Count of Reopened Defects) \* 100 | Daily | SIT, E2E |
| % Defect Rejection | Provides the list of invalid defects | (Total No of defects rejected/Total defects logged)\*100 | Daily | SIT, E2E |
| Defect Aging Trend Report | Highlights long outstanding defects | Defect Aging for open defect = (Current date – defect logged date) | Daily | SIT, E2E |
| First Time Pass Rate | It measures the percentage of the product passing without rework | Number of test cases that pass the test first time/Total number of test cases | Every Iteration | SIT, E2E |
| Environment Stability Matrix | It measures the stability of the application | (Number of downtime hours/Total number of hours the application expected to be available up & running) \* 100 | Every PI | Iteration Testing, SIT, E2E |
| Automation Coverage % | It determines the percentage of test cases automated vs total available test cases | (Number of automatable test cases/Number of test cases) \* 100 | Every PI | Iteration Testing, SIT, E2E |
| UAT | If a customer or end user identifies during User Acceptance Testing | Number of Defects raised by UAT/(Number of Defects raised by Tester - Number of invalid defects raised by Tester prior to UAT) \* 100 | End of UAT | UAT |
| Defect Leakage (Production) | If a customer or end user identifies a defect in Production environment project release, it is termed as Defect Leakage | Number of Defects raised in Production /(Number of Defects raised during UAT - Number of invalid defects raised during UAT) \* 100 | End of Production Validation | Production Validation |

**Predictability Metrics:**

| Metric | Objective | Calculation | Frequency |
| --- | --- | --- | --- |
| Burn Down | It monitors the hours of scrum team requires to complete work in a Iteration | Actual Points remaining = Planned Points - Actual Points | Daily |
| Percent Accepted | It measures the user story accepted vs the actual number of user stories | (User story accepted by the customer/Number of stories) \*100 | Daily |
| Velocity | It determines the total User Story Points that is completed by the team | Number of user story points completed across Iterations | Every Iteration |
| Points & % to address Defects | * Total “User Story” Points to resolve Defect. * Percentage of Velocity | Number of user story points to resolve Defects (Jira issue type = Bug, Status = Done).  Percentage is above divided by Velocity | Every PI |

**3.0 Defect Definitions**

|  |  |
| --- | --- |
| **Defect Severity Type** | **Description** |
| Severity 1(S1) | * CRITICAL - Generally reserved for fatal errors that mean that testing cannot continue without fix, and/or means that * the service cannot go-live. No workaround. Must be fixed before testing of a feature or in worst case, the entire * system, can continue. Must be fixed before Go Live. |
| Severity 2(S2) | * MAJOR – Used when there is a problem that means that testing can continue on the scenario using difficult * workarounds, and/or significantly impacts the business' ability to use the application. Must be fixed before Go Live. |
| Severity 3(S3) | * NORMAL – Used when there is a problem that means that testing can continue with relatively straightforward * workarounds, and/or has a minor impact on the business’ ability to use the application. |
| Severity 4(S4) | * MINOR – Used to highlight issues that will be fixed only if time permits and does not impact the businesses ability to * use the application (e.g. cosmetic). |

|  |  |
| --- | --- |
| **Defect Status** | **Description** |
| Dev Ready Open | Default state for New Issue |
| In Dev Open | RCA has been determined, issue is accepted and actively being fixed by development. |
| In QE Open | Fix has been delivered to QE and is being validated as resolved. |
| Dev Done Fixed | Issue is fixed by development and is awaiting regression/deployment |
| QE Approved Closed | – Issue has deployed and verified by QE |

|  |  |
| --- | --- |
| **Defect Cause** | **Description** |
| Environment | Issue was caused by an unexpected problem with the test environment or test data integrity Cycle. Often this is due to an issue with a dependent but not directly under test system |
| Set Up / Config | Issue was caused by a deployment setup or configuration problem. These are generally addressed with release note updates and/or fixes to deployment automation accordingly |
| Human Error | Issue was caused by invalid interaction, accidental use of incorrect test data or other execution flaw. These are generally addressed by updates to the automated test case, automated tool(s) or manual workflow procedures |
| Root Cause Unknown | RCA is still being determined |
| Duplicate | Duplicate of another issue. Generally closed in deference to the older issue |
| Cannot Reproduce | Issue was observed during testing; however, QE and development are unable to re-create the issue or properly explain the issue’s origin. These are most often low (S3 or S4) issues and if risk is not severe nor further reliability testing is unable to uncover the issue these are closed |
| Code | Issue was an error in the software code/configuration or an un-expected deviation in the code/configuration from requirements, user stories or defined acceptance criteria |
| Documentation | Issue with user documentation and/or release notes, however, the system itself is working as designed |
| Merge | Issue specifically related to a code merge of one or more code branches |
| Requirement | Issue is an error with the requirements, user stories or defined acceptance criteria |