QA

Version

0.2.0

Author(s):

Sajjad Hossain

Reviewer(s):

Round 1:

- Alejandro B.
- Tatiana T.

Round 2:

- Sady M.
- Luisa R.

Round 3:

- Denila P.
- Diana L.

Final:

- Kirthi J.
- John H.

Legend & Contents

- Current Process (written in black)
- Setbacks (written in red)
- Improvements (written in green)

Sprint Tasks

Planning

Ideation

1. Initiatives are created by business with business scenarios
    a. Epics are created from Initiatives
2. Epics are usually associated with at minimum 2 stories, a **frontend story** and a **backend story**
3. Stories [FE, BE] contain multiple defined scenarios

Currently, I am noticing not every team is following this Scenario-to-Story methodology. It is not a 1:1 ratio.

a. Stories should have acceptance criteria, because they do not there is an additional step required to understand how something should work and be tested
b. If we automate, we can duplicate the information from the Epic, and have it trickled down to tickets and subtasks as needed.
c. The user story should follow the following model:
    i. As a <user>
        1. developer
        2. tester
        3. business owner
        4. product owner
        5. user
    ii. I want
        1. To <completeAnAchievement>
        2. A <functionOrMethod>
        3. A <newBusinessFunction>
        4. A <newStory>
        5. A <feature>
    iii. So that
        1. I can <completeAnObjective>
        2. I can <doSomethingElse>
        3. We can <haveSomeBusinessValue>
d. All Scenarios in the user story should be a subtask in the FE/BE ticket, linked to the epic
e. Scenarios should follow the following model
    i. Scenario: I want to complete an action
    ii. Scenario: I want a new button on the homepage
    iii. Scenario: I want a page that lists all my optimizations
f. I think we would also be able to automate a lot of our work if we prepped for that in JIRA, for example:
    i. In our acceptance criteria, we should have an array or the scenarios:

1. ["Scenario: I want to complete an action", "Scenario: I want a new button on the home page", "Scenario: I want a page that lists my optimizations"]

ii. Our FE and BE Stories should be broken down further, in to sub tasks:

1. Testing should be a subtask
2. Each scenario defined in the epic, should be a subtask
3. Each dev subtask, FE or BE, should have an acceptance criteria that maps to Test Rails

4. Tickets that are created sit in the backlog unassigned to any sprint or resource

Grooming 1

1. Scenarios are written for frontend and backend stories
2. Designs are collected and reviewed
3. Ideas are discussed with the teams and feasibility is identified
4. Stories are assigned to sprint

Grooming 2

1. Scenarios are groomed thoroughly by frontend and backend team
    a. Scenarios do not use the same language as the stories, this causes a big gap in translation to code or test
        i. Acceptance criteria should be defined for EVERY ticket, or else we do not have definition of done
        ii. See the models above to see how it should be written
2. Functionality is confirmed
3. Testing strategy should be defined
4. Stories are pointed
5. Due dates are estimated
6. Sprint should be ready for start

Development

Sprint Start

1. When sprint is initiated,
    1. Test Runs are automatically generated:
        i. One run for each ticket in the sprint
        ii. API Automation Test Run
        iii. UI Automation Test Run
    2. Milestones are automatically generated:
        i. Sprint Performance
        ii. Sprint Feature Scenarios
        iii. Sprint Manual Regression
        iv. Sprint Automated Regression
    3. I think we should stop manually curating Test Rails
        i. Every task in that tool should be automated

ii. If we automate, we can duplicate the information from the Epic, and have it trickled down to tickets and subtasks as needed.

Mid-Sprint

2. While sprint is in progress, tests are created for each scenario in the story
    1. Each scenario has its own test case
    2. Tests are created with the following type information:
        i. Type
        ii. Milestone
        iii. Reference
        iv. Description
    3. Test cases are linked via reference to their JIRA ticket manually
    4. Test cases are linked to Test Runs manually
3. Feature branch is created and checked out for feature
    1. Automation and development work exist in the same branch

JIRA

4. When tickets are moved to in progress:
    1. Tester is assigned via a label
    2. User stories are defined
    3. Test cases are updated with user stories manually
    4. Test case is labeled and managed under in Test Rails either:
        i. Automation
        ii. To Be Automated
        iii. Isolated (Obsolete – One time manual)
    5. Cucumber step definitions are scripted where possible
    6. Configurations and data sets are curated where possible
5. Tickets are moved from In Progress to Ready for QA, when ready
    1. Tickets In QA are assigned to QA resource
    2. Steps to test are added to the ticket
        i. Test run is manually closed with a Pass/Fail status

GIT

6. When tickets are complete:
    1. If the ticket passes QA:
        i. Merge branch in to develop with a pull request
        ii. Review pull request with 2 developers
        iii. Passes SonarQube and Health checks
    2. If it fails it re-enters the development process until it passes or is moved to backlog
        i. The feature branch stays as is and is not merged

Sprint End

JIRA

7. Tickets are marked fixed by developer
8. Tickets In Review are peer reviewed by another developer
    1. When they pass review they are Ready for QA
9. Tickets in QA Review must be available to test in QA environment
    1. If tickets pass QA review they can be closed by a QA or Product resource
    2. If the feature fails QA, then they are returned to the developer to re-enter the development process.
10. Once the ticket is verified, it can be closed by QA or Product
    1. Developers sometimes include the pull request associated with the work in the ticket

GIT

11. Once tickets are complete there is usually a pull request that should be reviewed by 2 members of the team
12. Once reviewed the merge to develop is allowed

Test Rails

13. Tests that are in the "To be automated" are moved to status "Automated"
14. Tests that are in the "Sprint" buckets are moved to their respected component buckets

Delivery

Test Runs

1. When tests are executed, the associated test run is updated with the status of the test
2. General test runs like Automated Regression or Manual Regression are also updated with the status of the tests passed for tests that are linked
3. If test runs are not updated, builds will not be certified
4. Test runs are closed, test milestones are closed

Metrics

1. At the end of each sprint developers, product and QA have to update stakeholders with the following metrics:
    a. https://docs.google.com/spreadsheets/d/1Zuephskn1LHO9VBK80nk5-oc6T--Txqe4iBzlDNNV3w/edit?usp=sharing

Certification

2. In addition to code/project metrics, leads are expected help with the certification of builds
3. Leads are expected to report:
    a. Bugs opened, closed, found, deferred, in backlog
    b. Tests executed. Passed