Quality Assurance Testing Procedures and Criteria

QA Procedure

When the product team has finished work on a story, they will ensure the story has QA acceptance criteria and move it to the QA column of the GitHub Development project board, and they will tag the reviewer(s) to let them know the story is ready for QA.

****As a general rule of thumb, QA should start testing a story no later than 1 business day after it was tagged for QA in GitHub. All efforts should be made to test an item the same business day if it lands in QA before 12pm ET.****

That said...testing should never be rushed. We do not work on hard deadlines. Be thorough and always test with the user in mind. Don't hesitate to ask the product team questions about how things work or why a certain way of doing something was chosen. Just because a card meets the acceptance criteria does not mean it is Ready for Deploy. If follow-up is needed from other departments or if additional clarification is needed from development, ask questions!

How to Test a Story

To test a story, you use a staged version of the app to follow the logical steps that a customer would take when using the feature affected by the story. For example, if the story is a change to the way sign up works, then you will test multiple variations of how a user might sign up for different plans and options. Or, if the story is related to how a particular feature works, then you will test the various ways a user might use that feature.

Refer to the Regression Test documentation as needed to identify details and intertwined features that may be affected by any given story. Always err on the side of testing a broader scope than may be strictly necessary to locate and

prevent any bugs that may have been unintentionally introduced with the new code.

Edgar Regression Testing: Edgar's Manual Test Suite

Edgar Expected Behavior Index: Expected Behavior

GitHub Development Board

When a story is ready to be tested, the product team will move it to the QA column and tag the testers to let them know the story is ready for review. Test the story thoroughly. If it passes QA, move the story to the Passed QA column and tag the developer to let them know the story is ready to be merged.

If the story does not pass QA and requires additional work from a developer, tag the developer in a comment explaining the issue and return the card to that developer's list on the Development board.

When the story is ready to be checked again, the developer will move the story back to the QA column and tag the reviewer again. Check it again and either return for further revision or pass it.

PR App Branches for Testing

When the product team tags a story for QA, they will create a staged branch of the app to test the story on and will include the pull request (PR) number on the card for the story. These branches are located at edgar-staging-pr-PRNUMBER.herokuapp.com. Just put the PR number for the story in the URL to access the branch. For example, if the PR number for the story is 2501, the URL would be edgar-staging-pr-2501.herokuapp.com.

NOTE: PR branches do not work to test billing, link shortening, or adding/refreshing accounts. For stories involving these issues, ask dev to place the story on edgar-acceptance.herokuapp.com.

QA criteria

Key Questions to Consider While Testing a Story:

- Does it work as expected and meet the 5 Es?
    - Efficient (can users find information and complete tasks in a timely manner?

    - Effective (does it do what it's supposed to do well?)
    - Engaging (does it look correct based on the designs? Is it rendering properly? Does it fit the Edgar voice & keep things on brand?)
    - Error tolerant (will users get errors as a result of this story or have their current workflow altered? If so, can they understand it and recover on their own? Does it cause any unintended issues in related parts of the app?)
    - Easy to learn (does this story fit in well within the larger app. Is it easy for users to get started on their own with this?)
- Does the feature meet the acceptance criteria laid out by the development team?
- Does it affect other aspects outside of the app? Note any changes that should be made to:
    - Marketing site copy
    - Intercom messaging
    - Help Docs
    - MailChimp messaging
    - Other Notion or Google Doc procedure pages

When a Story Passes QA

When you're satisfied that a story meets acceptance criteria and is ready to be published to the production version of the app, move the story to the Passed QA column of the Development board and tag the developer to let them know the story is ready to be merged.

Help doc updates: After a story passes QA, make notes on any design or function changes that may affect help docs. Create an Asana task for the Help Doc Owner and another for the Intercom Owner with the story details, approximate date of upcoming deployment, and any additional information needed so that they can plan any necessary help doc and Intercom changes.

Manual Test Suite Updates: note any new test cases that the development work has introduced to be added to the Manual Test Suite

Deploy: When a story is being deployed, the product team will announce it in the #deploy channel in Slack. Then they will release it to the live production version of the app that customers use.

QA Post Deploy Tasks

Once a story has been deployed, QA should complete the following tasks or assign them to the appropriate person for that area of ownership:

- **Testers:**
  - Log into the production version of Edgar and confirm that the newly deployed feature/bug fix is working as expected.
  - Update the Expected Behavior Index and Manual Test Suite for anything that changed or was added as a result of the story deployed.
  - Create Asana tasks for Help Docs/Intercom/Marketing Owners with any design or copy updates that will require changes to help docs, Intercom messaging, or Marketing site.
- **Help Doc Owner:** Update any related Help Docs, screenshots, or videos that are now out of date due to the deployed change.
- **Intercom Owner:**
  - Update any intercom messaging that is out of date.
  - Coordinate a customer email or in-app message and blog post if appropriate to announce a new feature.
- **CX:** Contact any customers that were affected by a bug and notify them that a fix has been deployed.