System and Unit Test Report

SPLITR - 03/3/21

We used a manual testing approach to test our application.

System Test Scenarios

- Sprint 1
  - Stories
    - As a “payer”, I want to be able to upload a picture of a receipt to the website so that it will convert to a checklist
    - As a "payer", I want to be able to view what I've uploaded
    - “As a "payer", I want to be able to confirm that the items on the checklist are correct so that I don't pay an incorrect amount”
  - Scenario 1
    1. User opens the web app; click upload button
       a. A dialog should pop up requesting to upload a file from the user’s device
    2. User selects receipt image; click upload button again
       a. The user should see the image that was uploaded on the webpage
       b. The user should see the list of items that were parsed from the image
    3. User clicks on next button; should navigate to split page.
- Sprint 2
  - Stories
    - As a "payer", I want to be able to select what I've ordered so that I may pay for them
    - As a “payer”, I want to be able to select which items I shared with other people and how many people i shared them with so that I am not stuck paying the full price by default
    - As a “payer”, I want to be able to manually input the tip amount (percentage or cash) because it is not pre populated by default on most bills
    - As a “payer”, I want my tax and tip amounts to reflect how much of the bill I was responsible for and for that amount to be added to my total cost so that I end up paying a fair share of these expenses

        - As a “payer”, I would like to select an additional expense like delivery cost in case this was a group delivery order
    - Scenario 1
        1. User is on split page with following receipt items:
            a. Bananas - $5.00
            b. Cupcake - $5.00
        2. Click checkbox next to Bananas, should see “Your Split: $5.00”
        3. At the bottom, user clicks on input fields and types:
            a. Tax: 1.00
            b. Tip: 1.00
            c. Misc. Fees: 1.00
            d. User should now see “Your Split: $6.50”
    - Scenario 2
        1. User is on split page with following receipt items:
            a. Bananas - $3.00
        2. Click checkbox next to bananas, should see “Your Split: $3.00”
        3. Click on dropdown arrow on the same row
            a. Click on plus icon next to “Shared Ways”
            b. User should see “Shared Ways = 1”
        4. User should see “Your Split: $1.50” at the bottom
    - Scenario 3
        1. User is on split page with following receipt items:
            a. Bananas - $5.00
            b. Cupcake - $5.00
        2. Click checkbox next to bananas, should see “Your Split: $5.00”
        3. User selects Tip dropdown and clicks “%”
            a. Type in the field: 10
        4. User should see “Your Split: $5.50” at the bottom

- ## Sprint 3
    - Stories
        - As a user, I would like a payment page so that I may process my payment
        - As a "payer", I want to be able to select who I want to pay so that I may complete the transaction
        - As a “payer”, I would like to be able to just input a venmo username and press pay and immediately be routed to venmo to confirm the transaction so I don’t need to manually switch apps
        - As a “requester”, I would like to be able to just input a venmo username and press request and immediately be routed to venmo to confirm the transaction so I don’t need to manually switch apps
        - As a user, I would like a Back Button on each page past the Homepage to be able to navigate to previous pages
    - Scenario 1 (payer)
        1. User is on pay page with “Your Split: $5.00”
        2. Click on user input field; type “JohnDoe11”

        3. Click on button pay
        4. User should be routed to venmo page to pay $5.00 to user JohnDoe11
    - Scenario 2 (requester)
        1. User is on pay page with “Your Split: $5.00”
        2. Click on user input field; type “JohnDoe11”
        3. Click on button request
        4. User should be routed to venmo page to request $5.00 from user JohnDoe11
    - Scenario 2 (requester)
        1. User is on pay page; click on back arrow button on top left of page
        2. User should be on split page with same receipt info; click on back arrow button again
        3. User should be on upload page

- Sprint 4
    - Stories
        - As a "user" of the app, I would like it the UI to be responsive across platforms so the experience is smooth no matter what size device I am using
        - As a "user" of the app, I would like the UI to be unique and pleasant to use to create a nice user experience
        - ● As a "user" of the app, I want to be able to upload .HEIC pictures because that is the default photo format for iPhones
        - ● As a “user” of the app, I would like to be able to share a receipt url to populate the items
        - ● As a "user" of the app, I'd like my inputs on the payment page to be validated so I know that my formatting is correct
        - ● As a "user" of the app, I would like to have a Delete All Button on the edit page to quickly delete all items
    - Scenario 1
        1. User is on split page, click on share button icon on the top right.
        2. User should see popup dialog with link to share
        3. Click on “copy link”
        4. User should have the link in their clipboard
    - Scenario 2
        1. User is using app on mobile and desktop
        2. User upload image, click next, click pay on mobile and desktop
        3. User should confirm page layout fits both mobile and desktop screens
    - Scenario 3
        1. User is on pay page
        2. Click on user input field; type “^$*$#gh%^^@”
        3. Pay and request button should be disabled, and a validation text should show under the input in red.
    - Scenario 4
        1. User is on split page with following receipt items:

        a. Bananas - $5.00
        b. Cupcake - $5.00
    2. Click “Delete All” button
    3. All receipt items should be deleted from the list

Unit Tests

- Matt Ngo
    - Module: “Your Split” amount display
    - Tests:
        - Selecting Items should update the split amount
            - On a populated receipt table, click on one or more checkboxes
                - EX:
                - Items = Apple:$2.00, Orange:$3.00, Sandwich: $10.00
                - Select Apple and Orange
            - The prices corresponding to the items selected should simply total up at the bottom of the page under “Your Split: “
                - For EX above, split should be $5.00
        - Fees should be split according to the user’s % contribution to the bill
            - On a populated receipt table, click on one or more checkboxes
                - EX:
                - Items = Apple:$2.00, Orange:$3.00, Sandwich: $10.00
                - Select Apple and Orange
            - Ensure that there are valid fee values for tax, tip or misc fields
                - Let Tax be $2.00, Tip be $3.00
            - A fraction of the fees should be added to the “Your Split: “ amount corresponding to the amount of selected items’ contribution to the total bill
                - For this example. Split amount should be $7.50
                    - Selected item total = $5.00, this is 50% of the bill
                    - Fees total = $5.00, 50% of this is $2.50
                    - $5.00 + $2.50 = $7.50
        - Incrementing shared amount on Items should update the split amount
            - On a populated receipt table, click on one or more checkboxes
                - EX:
                - Items = Apple:$2.00, Orange:$3.00, Sandwich: $10.00
                - Select Sandwich
            - Click on the right hand dropdown arrow and press the (+) to increment the “shared amount”
                - Press the (+) 3 times to get “Shared 4 ways”
            - The amount under “Your Split: “ should equal the selected item divided by the amount of ways the item was shared
                - For EX above, split amount should be $2.50
    - Module: “Pay” and “Request” buttons

    - Tests:
        - Inputing a username then pressing Pay or Request populates the invoice
            - On a mobile device, Input a valid venmo username
            - Press Pay or Request
            - Venmo app should open with a partially filled invoice with that username already populated
        - Inputing a Phone Number then pressing Pay or Request populates the invoice
            - On a mobile device, Input a valid phone number
            - Press Pay or Request
            - Venmo app should open with a partially filled invoice with the recipient field already populated with the phone number given or the username associated with that phone number.
        - Buttons should not be available on Desktop
            - On a laptop device, open the payment page
            - Venmo buttons should not be available for use, and instead a message indicating this should be on the page
            - On a mobile device, navigate to the same page
            - Venmo buttons and input field should be available for use
    - Module: “Open In App” Button
        - Pressing “Open In App” button routes to friend list
            - On a mobile device, tap the “Open In App” button
            - Venmo app should open with a list of the user’s friends to select
- Richard Thai
    - Module: Image Uploading
    - Tests:
        - Upload Prompts
            - Clicking on the “Upload a Receipt” button displays a native OS (i.e. mobile and desktop) prompt to upload a file
            - Dragging an image into the “Upload a Receipt” button uploads a receipt on desktop platforms
        - Image Formats
            - Uploaded a variety of different supported image formats (.jpg/jpeg, .png, and .gif)
            - Attempted to upload unsupported image & file types to ensure that they were properly rejected
                - Initially, we did not support .heic format (native format for uploads from iOS devices)
                    - Support for .heic format
                        - Multiple areas of the code had to be modify to accommodate support for .heic format
                            - After support was integrated, re-tested all image formats to ensure their functionality had not regressed

                - Tested new .heic format to ensure it was being converted and attached to the API call accordingly
    - Image File Size
        - Uploaded images within the bounds of the allowed image file size
        - Attempted to upload large images that weren't in the bounds of the allowed image file size
    - # of Images
        - Attempted to upload more than one image to ensure uploading multiple images was properly rejected
    - Image Display
        - Upon successful upload, the image should be rendered on the page
            - Attempted multiple images of varying formats and sizes, ensuring that it both displayed properly and within the bounds of the application (no run-off images)
            - "Upload a Receipt" button should be converted to a "Delete" button allowing the user to delete their current upload and replace it with a different image
                - If a user clicks the "Delete" button, the image should not longer be displayed and the "Delete" button should reset to a "Upload a Receipt" button
                    - Tested on all image formats
    - Upload Progress
        - During the uploading process, a progress bar should appear to indicate to the user that the image is being uploaded
            - Tested on a variety of different image sizes - how long you see the progress bar is dependent on the image size
                - Larger image => Longer upload time => Progress bar displayed for a longer period
                - Smaller image => Shorter upload time => Progress bar displayed for a shorter period, if at all
            - Output internal progress onto Console viewable through Chrome's Developer Tools
                - Test to ensure that progress bar is appearing/disappearing accordingly
            - Disable "Upload a Receipt" and "Input Manually" button to ensure users don't accidentally disrupt uploading process
                - Upload an image
                - During upload, ensure that the buttons are disabled
- Module: API Calling Mechanism
- Tests:
    - Processing Image Uploads
        - Click on Process button

            - Wait for API call to complete
                - A progress bar should be displayed to indicate to users that the request is being processed
                - Prior accessible buttons should be disabled to prevent users from accidentally disrupting API call
            - Should be automatically re-routed to the next page - populated with the API response
        - Conversion of Data Format
            - API does not accept data URLs (default data format after upload), which had to be converted into data blobs
                - Test on all formats to ensure the conversion function is capable of converting all formats from data URLs to data blobs
        - Progression
            - Ensure users are not sent to the next page before the API is able to respond
                - Use Promise to ensure the next steps are not executed until the API response comes back
        - Data Sanitation
            - Data has to be sanitized and integrated accordingly into the rest of the system
                - Monitor full API response through Chrome's networking logging tool
                    - Read and interpret responses
                - OCR has the tendency to misinterpret how letters are capitalized and lowercase resulting in weird cases being returned
                    - Capitalize all characters on display to user
                        - Uploaded images with variety of character cases and ensure displayed characters are all capitalized
- Wai Chun Leung
    - Module: Item table component
    - Tests:
        - Adding a new item should update the total
            - Click on edit button
            - Click on add item button
            - Add valid receipt item
            - Click on save button
            - Should save the item on the list
            - Should show new total with item price added at bottom
        - Invalid new item should be invalidated
            - Clicking on edit button -> add new item to table -> type in inputs
            - Empty inputs does not make an item

            - Price inputs that are not numbers are invalidated
        - Deleting items should delete the item
            - Item table has one item “banana”
            - Click on edit button
            - Click on remove button next to banana item
            - Click on save button
            - Should delete the item on the list
            - Should show new total with item price subtracted from it
        - Clicking on Item row dropdown shows number of shared ways
            - Item table has one item “banana” with price $5.00
            - Click on dropdown button next to banana
            - Should dropdown and show “shared ways: 0” and “5.00 ea.”
        - Modifying shared ways of an item should show the split for that item in dropdown
            - Item table has one item “banana” with price $5.00
            - Click on dropdown button next to banana
            - Click on + icon next to shared ways
            - Should show “shared ways: 1” and “2.50 ea.”
            - Click on - icon next to shared ways
            - Should show “shared ways: 0” and “5.00 ea.”
- Austin Seyboldt
    - Module: Text input processing / cleansing
    - Tests:
        - Test price / percentage fields for invalid inputs
            - Tests cases included any string with characters other than numbers or periods
            - Test cases included multiple periods
        - Tested responsiveness and accuracy of displayed prices and totals
            - Tested empty prices to see if the total displays property when given an empty price
            - Tested various prices to see if total update immediately on new input
    - Module: Bit.ly link generation
    - Tests:
        - Used different combinations of receipt items, prices, tips, tax and checked that bit.ly link was generated
        - Inspected json objects being sent to bit.ly with Chrome to see if they matched the data in the table
    - Module: Table population of data from query string
    - Tests:
        - Tested that the data was accurate by matching it with the original data that produced the link
        - Used invalid query strings to make sure app could deal with them properly
        - Inspected objects with chrome developer tools for accuracy (no items should have undefined values or values of the wrong type)
- Weston Cook
    - Module: Receipt items list: edit mode

- Tests:
    - Checked that the “delete all” button consistently removes all items from the receipt
    - Checked that item prices are automatically highlighted when clicked on
- Module: Receipt fees section: edit mode
- Tests:
    - Checked that reported fees are consistent with the given value and the state of the fee type selector ($/%)
    - Checked that fee prices are automatically highlighted when clicked on
    - Checked that the total cost is correctly calculated from the subtotal and the fees
    - Checked that the user split cost is correctly calculated from the selected items and the resulting percentage of the fees
- Module: Receipt fees section: select mode
- Tests:
    - Checked that the fees are all displayed as dollar values regardless of whether they were entered as dollars or percentages