TabDrop System and Unit Test Report

Brooke and Co

Mar 7, 2022

System Test scenarios

Sprint 1

No user stories completed

Sprint 2

1. As a user, I need to be able to enter names for people and new names so that I can split the bill with these people.

Scenario

1. Click the text field
    a. Enter a name (example: “Bob”)
2. Click “Add”
    a. The name will display on a list underneath

Sprint 3

1. As a user, I need to select a photo from my camera roll so that I can split an old receipt.

Scenario

1. Open the TabDrop app
2. Click “Choose photo”
    a. Choose a photo of a receipt
    b. User should see the photo on the landing page
3. Click “Upload photo”

2. As a user, I need to be able to take a photo of the receipt so I can split it at the same time I receive the receipt.

Scenario

1. Open the TabDrop app
2. Click “Take a photo”
    a. Take a photo of a receipt
    b. Click “Use Photo”
    c. User should see the photo on the landing page
3. Click “Upload photo”

Sprint 4

1. As a user, I need to be able to see the list of items from the photo of my receipt, so that I can confirm everything is correct.

Scenario

1. Open the TabDrop app
2. Click “Choose photo” or “Take a photo”
    a. Choose or take a photo of a receipt (Click “Use Photo” if you take a photo)
    b. User should see the photo on the landing page
3. Click the text field
    a. Enter a name (example: “Bob”)
4. Click “Add”
    a. User should see the name on a list underneath
5. Repeat steps 3 and 4 if you want to add more names
6. Swipe a name to the left
    a. User should see a red “Delete” button
    b. Click the “Delete” button or continue swiping to the left
    c. The name will no longer be displayed
7. Repeat step 6 if you want to delete more names
8. Click “Drop people in the next screen”
    a. User should see the list of names entered at the top
    b. User should see the receipt items underneath the names

2. As a user, I need to be able to drag people into their items so that I can assign each person to at least one item on the bill.

Scenario (continued from Sprint 4, user story 1)

9. Press and hold a name at the top
    a. User should be able to drag the name around the screen
10. Drag the name to the item they want to pay for
    a. User should see the name on the “Paid by:” line for that item
11. Repeat the drag and drop at least once for each item
12. Click “Calculate Final Split”
    a. User should see the subtotal, tax, tip, and total of the full receipt
    b. User should see the breakdown of subtotal, tax, tip, and total for each name entered
13. Optional, click the text field for “Tax Percent”
    a. Enter the tax percent (example: “9.13”)
    b. User should see the tax will be updated for full receipt and for every name
14. Optional, click the text field for “Tip Percent”
    a. Enter the tip percent (example: “15”)
    b. User should see the tip will be updated for full receipt and for every name

3. As a user, I need to be able to add or remove items so that I can modify the end list.

Scenario (continued from Sprint 4, user story 1)

7. Click “+” at the top right of the screen
8. Click the “New Item” text field
    a. Enter a name for the item you want to add (example: “Cheese”)
9. Click the “Item Price” text field
    a. Enter the item price (example: “3” or “3.0”)
10. Click “Add Item”
    a. An alert message will appear saying “Item is added!”
11. Click “OK” on the alert
12. Click “Back”
    a. The screen will return to the drag and drop view (Sprint 4, user story 2)
13. Swipe an item card to the left to delete
    a. User should see a red “Delete” button
    b. Click the “Delete” button or continue swiping to the left
    c. The item will no longer be displayed

Unit Tests

To test the backend files first follow these steps:

1. Clone the repository.
2. Create a python virtual environment and activate it.
3. Enter on the console/terminal, "pip install -r requirements.txt", within the folder with the repository to install all dependencies.
4. Create a file named ".env" with your Azure API information with the format below:

    API_KEY=<Azure api key>

    ENDPOINT=<Azure api endpoint>

5. To test if the Azure recognizer file successfully scans the receipt photo, we can run testing.py that is in the testing folder. The testing folder files include: receipt.json, receipt_pic.jpeg, and testing.py. Paste the base64 string of desired receipt photo into receipt.json in the form.
    {“img_string”: “<base64 string>”}

6. Run testing.py
    a. The file testing.py will print to console the list of items in the format:
        [{“item_name”: <item_name> },{“price”: <item_price>}, …]
    b. If the list of items match the items displayed in the receipt picture, then the Azure recognizer successfully has scanned the item.
    c. To do more black box testing, repeat step 5 with a base64 string of a different receipt picture.
7. To further deploy locally and test flaskapi.py:
    a. IF ON WINDOWS: in console, enter "set FLASK_APP=flaskapi.py"
    b. IF ON MAC: in console, enter "export FLASK_APP=flaskapi.py".

c. In the console, enter "flask run". The backend should be running locally.
d. If the deployment is successful, flaskapi.py is functioning properly.