Lab 11 - Part A

Test Planning & Implementation

*please note: testing plans for each section were divided and completed among pairs in our group, as indicated below:

Checkout - Roxanne & Charlotte
Features - Andrew & Ryan
Search - Sean
Home - Ben

Checkout Testing Plan (Following work completed by Charlotte & Roxanne):

Case 1: User checks out with correct credentials

- **Description of Case:**
  - We test if the user checks out and purchases items from their cart with the correct credentials (ie credit card number, address being in the correct format).
- **Description of Test:**
  - We plan to test this case with valid and invalid data through direct input (ie integration testing).
- **Acceptance Criteria, with mandatory fields are as follows:**
  - Credit Card
    - valid credit card number format (proper format; XXXX-XXXX-XXXX-XXXX)
  - Address
    - (proper format; 1111 Street Name, Boulder, CO, XXXXX)
  - Product detail
    - Make sure they know what they are checking out (amount > 1 item)
- **Description of Test Data:**
  - We plan to use a plethora of different inputs, both valid and invalid that would cover edge cases.
    - Credit card:
      - 1234-1234-1234-1234
        - Passes
      - Aaaa - 1234 - 1234 - 1234
        - Fails - no characters
      - An empty input
        - Fails - no input to check
    - Address:
      - 1086 Pancake Dr, Boulder, CO 80706
        - Pass → credit card and address are in the correct format
      - 11111 sdfk nmasd Denver, CO 70201
        - Fails → not in correct format
      - Empty input
        - Fails → no input

        - 5092 !!!! Waffle Way Denver, CO, 80703
            - Fails → incorrect format (special characters are to not be included)
    - Product details
        - Checked out 1 item
            - Passes → they checked out an item
            - Ensuring correct data presents itself
        - No items to check out
            - Fails → they need to be checking out at least 1 item

- **Description of Test Environment:**
    - This will be tested on QA environment, which will allow us to actively test and stage the application and its capabilities.
- **Description of Test Results:**
    - Credit card:
        - Fails → if the test data is invalid we would want an error message to present itself and direct the user to a correct input
        - Passes → should not present a message and a user can move on to the next input and checkout
    - Address:
        - Fails → if the test data is invalid we would want an error message to present itself and direct the user to a correct input
        - Passes → should not present a message and a user can move on to the next input and checkout
    - Product details
        - Fails → if the test data is invalid we would want an error message to present itself and direct the user to a correct input
        - Passes → should not present a message and a user can move on to the next input and checkout
- **Description of User Acceptance Tester:**
    - These would be the actual users that input data through our testing environment. They could be software developers or other employees

Case 2: Delete an item from cart (Other items remain in cart); Display message to user

- **Description of Case:**
    - We test that if the user deletes an item from their cart, and there are still a few items that remain, a message will be displayed to the user.
- **Description of Test:**
    - We plan to test this case with valid and invalid data through direct input (ie integration testing).
- **Acceptance Criteria, with mandatory fields are as follows:**
    - Manual Click
        - Pass → user clicks on an item, the item is deleted from the cart
        - Fail → user fails to click on the item, which does not delete the item

- **Test Data:**
  - Clicks
    - This test will be implemented by using clicks from the user to confirm that the item has been deleted from the cart.
- **Test Environment:**
  - This will be tested on QA environment, which will allow us to actively test and stage the application and its capabilities.
- **Test Results:**
  - Pass:
    - If user deletes an item from their cart by manually clicking the delete button, a message should be displayed, informing the user that the item has been successfully deleted.
  - Fail:
    - No message is displayed when the user deletes an item from their cart, and other items are still remaining in the cart.
- **User Acceptance Tester:**
  - These would be the actual users that input data through our testing environment. They could be software developers or other employees

Case 3: Delete an item from cart (last item, cart will be empty); Bring user back to home page

- **Description of Case:**
  - We test that if the user deletes an item from their cart, the user will be redirected to the home page.
- **Description of Test:**
  - We plan to test this case with valid and invalid data through direct input (ie integration testing).
- **Acceptance Criteria, with mandatory fields are as follows:**
  - Manual Click
    - Pass → user clicks on an item, the item is deleted from the cart
    - Fail → user fails to click on the item, which does not delete the item
- **Test Data:**
  - Clicks
    - This test will be implemented by using clicks from the user to confirm that the item has been deleted from the cart.
- **Test Environment:**
  - This will be tested on QA environment, which will allow us to actively test and stage the application and its capabilities.
- **Test Results:**
  - Pass:
    - If user deletes an item from their cart (ie the last item, leaving the cart empty), by manually clicking the delete button, the user should be redirected to the home page, where they will be given the option to obtain more information from the homepage or use the search bar.

- ➢ Fail:
  - The user is not redirected to the home page when the last item is deleted from the cart; rather, there is an error message or the user remains stagnant on the checkout page.

❖ **User Acceptance Tester:**

- ➢ These would be the actual users that input data through our testing environment. They could be software developers or other employees

Case 4: User checks outwith no overdue materials

❖ **Description of Case:**

- ➢ We test if the user checks out and purchases items from their cart without any overdue items (ie if a user already has an overdue item, then they would get a warning and wouldnt be able to check out until they extended their rental time or returned the item).

❖ **Description of Test:**

- ➢ We plan to test this case with valid and invalid data through direct input (ie integration testing) such as trying to check something out with and without overdue items

❖ **Acceptance Criteria, with mandatory fields are as follows:**

- ➢ The user has overdue items
  - User would need to either need to pay overdue fee
    - Such as paying for more time on item and then they can continue
  - User would need to return item before checkout another item
    - User would need to physically return item and wait for the system to register that the item is received before checking out another time

❖ **Test Data:**

- ➢ We plan to use a plethora of different inputs, both valid and invalid that would cover edge cases.
  - Overdue item
    - Fails → user needs to pay or return item
  - Multiple overdue items
    - Fails → User needs to pay or return item
  - No overdue items
    - Passes → User may checkout and continue to use this software

❖ **Test Environment:**

- ➢ This will be tested on a QA environment, which will allow us to actively test and stage the application and its capabilities.

❖ **Test Results:**

- ➢ Items in cart
  - Fails → if the user has overdue items and they are still able to check out
  - Fails → if the user doesnt have overdue items and isnt able to check oout
  - Fails → if the user pays fee or returns item and isnt able to check out

        - Passes → if the user has overdue items and they are not able to check out
        - Passes → if the user doesnt have overdue items and is able to check out
        - Passes → if the user pays fee or returns item and is able to check out
- **User Acceptance Tester:**

Features (Following work completed by Ryan and Andrew):

1. Register - *Description:This api call inserts a new user into the database if the entered information passes the test cases and automatically logs the registered user in. It will then redirect the page to the home page.

    *Test Cases

        -Case 1: The api checks if the user is already in the database, if not then it increments the key and inserts a new user into the db.

            *if Acc70 (Username) is not in the database, then insert it into the database with the password 1234.

        -Case 2: Make sure that special characters and spaces do not interfere with searching the database. Also don't allow the user to have spaces in a username.

            * Acc 70(username) would return an error

            *Acc@70 (username would search the database for Acc@70 and not Acc70

        -Case 3: Make sure that the new user is logged in when a new user is registered.

            *no data involved

        -Case 4: Make sure that if a user is already logged in and they manage to get to the register page, that they are redirected to the home page.

            *no data involved

        -Case 5: If a username or password is too long to insert into the database let the user know that it is too long.

            * namenamenamenamenamenamenamenamenamenamename would return an error to the user since the username is too long.

    *We should have three fields in the table users in the database:

        -The serial key

        -username

        -a hashed password

    *The fields the user would need to input are a username and password.

    *Our testing environment will most likely be QA.

    *The testers would most likely include our groupmates and the TAs.

Search (Following work completed by Sean):

Note: there should be some data in the items table before testing.
Case 1: User inputs valid string:
The user types in a string which matches part of a name in the item table and hits search. Ex: “Laptop” The api then renders all items which have part of their names matching the search string and a button which is disabled if the item is not available and clicking on it will call the

checkout api. It should not return an error or no results. Testers could include team members or classmates.

Case 2: User inputs invalid string:
The user types in a string which does not match any names in the item table and hits search. Ex: “nfewnaihfj”. The page should then render a message saying “no results found”. The testers could include team members or classmates.

Case 3: User hits search button directly.
The user does not input anything into the text field and hits search. The api should then render an error message on the page. It should not render any results. The testers would be team members.
Environment: QA

Home Page - Following work completed by Ben

*Description:This api call presents the “site’s” available items in the database.

Case 1: We can see the site’s available items, with date that it is due and an image of the product.

- **Description of Case:**
  - We test if the page properly loads and presents the items checked out by the user and any relevant information having to do with these ‘loaned’ item(s).
- **Description of Test:**
  - Test this case by running the files via docker and checking for proper return status. We can create dummy data of different items to check out, and check if the program properly presents the data for each item, respectively.
- **Acceptance Criteria, with mandatory fields are as follows:**

The following Items should be present on the page.

- Item Name
- Image of item
- Button to “Checkout this Item”

- **Description of Test Data:**
  - We checked if all data is presented in full.
    - Item Name
      - Apple 2017 Macbook Pro Charger
        - Pass - we see the checked out item’s name.
      - Empty/Not Shown
        - Fail - we do not see the checked out item’s name.
    - Image of item
      - [image of item shown or default]
        - Pass - we see the checked out item’s image.
      - Empty/Not Show
        - Fail - we do not see the checked out item’s image.

        - Button to “Checkout this Item”
            - We see a clear clickable button on the line of a particular item, titled “Checkout this Item”
                - Pass - we see the checked out item’s “Checkout this Item” button.
            - Empty/Not Shown
                - Fail - we do not see the checked out item’s “Checkout this Item” button.

- **Description of Test Environment:**
    - This will be on a staging environment of the application and its capabilities.
- **Description of Test Results:**
    - Pass
        - You see all the information presented, with the “Checkout this Item” Button being clickable.
    - Fail
        - You do not see all the information presented, with the “Checkout this Item” Button not being clickable.
- **Description of User Acceptance Tester:**
    - These would be the actual users that input data through our testing environment. They could be software developers or other employees