Name	Yumeng Luo	Kevin Hwang	Sang Jun Chun	Shengtan Mao
UNI	yl4655	jch2169	sc4658	sm4954

Part 1:
https://github.com/yumeng-luo/COMS4156

Part 2:
User Stories

USId1. (NearBy) As a user who **wants to purchase items nearby**, I want to be able to **search for items near me and see where they are on a map**, so that I know **where I can purchase cheaper items I need**. My conditions of satisfaction are :

a. The tool correctly identifies my current location
b. The items searched are near me
c. The tool correctly represents the searched items's price and location on the map

USId2. (Alternative) As **a user looking for cheaper alternatives or closer alternatives or specific items but in different locations**, I want to **specify what type of alternatives I am looking for** so that **I can purchase better alternative items**. My conditions of satisfaction are:

a. I can customize the type of alternatives Im looking for
b. The tool proposes the alternatives satisfying my need

USId3. (Saving) As **a user with a tight monthly budget**, I want to **know and choose cheaper alternatives for daily expenses** so that **I can stay under budget and save money**. My condition of satisfactions are:

a. The tool can find information about item I want
b. The tool proposes a cheaper option
c. I can choose cheaper options
d. The tool correctly records my savings

Acceptance testing plan

USId1. (Nearby) Common Case

a. The tester logins with the tool.
b. The tester types in the name of item he is looking for
c. After the tool returns the list of locations on map, check if these locations are actually near the tester's location. If they are not the test fail, otherwise continue
d. Check if prices are reported with the locations and check if the item matches the search criteria, if yes, the test passes, otherwise the test fails

USId1. Special Case

a. (Branching from a) If the tester does not provide valid credentials (e.g. not logged in), then verify the tester cannot use the nearby search feature using Google Maps API.

b. (Branching from b) If the tester inputs an invalid name, and error message is reported
c. (Branching from c) If the tester does not grant the browser necessary permissions to track location, the tool should use the default location (currently set to Australia).

USId2. (Alternative) Common Case
a. The tester searches for an item and selects 1 from the given list. (This item is referred as selected item)
b. The tester is presented with 3 checkboxes about potential alternative criteria (Alternatives must be nearer than selected item; Alternatives must be cheaper than selected item; Alternatives must be the exact same item but at different locations)
c. The tester check some combinations of these boxes and ask for alternatives
d. The alternatives generated by the tool should be 1. Closer than selected item if “closer” box is checked, 2. Cheaper than selected item if “Cheaper” is checked 3. Same as the selected item if “Same” is checked. If the alternatives meet these criteria the test pass, otherwise the test fails

USId2. Special Case
a. (Branching from d) If no item is found that meets the criteria, an error message is reported.

USId3. (Finalization + Savings) Common Case
a. The tester searches for a product and chooses the best matching item from the search result. The tester should verify alternative options are recommended by the tool.
b. The tester chooses the most feasible alternative from the recommendations.
c. The tester should verify that the tool correctly shows the potential savings based on the original product and the final choice (cost of original - cost of alternative chosen)
d. The tool will pass the test if the weekly and total savings are incremented by the amount the user saved by choosing an alternative.
e. At the end of the week (Sunday), the tester should receive a weekly email highlighting their savings using the tool.

USId3. Special Case
a. (Branching from case a, b) The tool should protect the tool and the database from malicious inputs such as “input 11, DROP TABLE USERS;” via malicious manipulation of endpoints
b. (Branching from case d) After choosing an alternative, the tester decides to buy the alternative and returns to the search. The total savings should remain unchanged and the tester should return to step c.

Acceptance Testing: Discussions

USId1. (Nearby)

a. Expected behavior
   i. (Tested on backend) a user searches for an item, the user should see a list of relevant items available near them
   ii. User sees a map with markers to nearby stores carrying searched items and alternatives
   iii. If a user refuses to let the browser see their location, a default location (Australia) is assumed.
   iv. Only the shops shown in Google Maps are listed on the website
b. Unexpected behavior
   i. Bug found and fixed: Google Maps feature was available to unauthenticated users as well. This issue was fixed so that only those who are logged in through Google can use the feature.
   ii. Bug found: After authenticating, the user is redirected to a random, incorrect webpage (such as source code). Not fixed.
   iii. Unintended bug: User selects an initial product, goes to home page and does a different search, old request keeps living in database and messes up savings calculation. We kept this functionality to allow users whose search process was interrupted to continue with the search.
   iv. Bug found: Undefined behavior for invalid user inputs. We partially fixed any bugs we found but there are potentially many more.

USId2.

a. Expected behavior
   i. The tool provides a list of relevant alternatives to each “original” item.
   ii. When no alternatives are available, a warning message is raised.
   iii. The ongoing search process is saved to the database to prevent data being lost upon unexpected errors.
b. Unexpected behavior
   i. Bug found: The current implementation is susceptible to malicious attacks/injections. This issue has not yet been fixed and will be addressed as the UI is developed and endpoints are finalized.

USId3.

a. Expected behavior
   i. Only the items that have been confirmed are updated to the database.
   ii. The user correctly returns to the list of alternatives if they choose not to confirm an alternative they choose.
   iii. Once a user confirms a product, the user is able to confirm another product which also gets added to the database.
   iv. The ongoing search process (ongoing task) that is saved to the database is reset after the user confirms purchase.
   v. The total/weekly savings are incremented accordingly when a purchase is confirmed.

b. Unexpected behavior
    i. (Not implemented) Weekly savings are reset every week (Sunday).
    ii. (Not implemented) The system is not yet set to send out weekly emails but we confirmed it to be manually functional.

Part 3:
Link to the folder(s) within your github repository containing all the test cases that are automatically invoked by your unit testing tool:

- https://github.com/yumeng-luo/COMS4156/tree/main/src/test/java/savings/tracker
- https://github.com/yumeng-luo/COMS4156/tree/main/src/test/jest_tests

Link to the file(s) in your repository that configure the build tool and/or package manager and the automated unit testing tool.

- Maven project configuration: https://github.com/yumeng-luo/COMS4156/blob/main/pom.xml

Part 4:
Run stylecheck and bug finder and post links to the reports

- Spotbugs information in https://github.com/yumeng-luo/COMS4156/blob/main/target/site/spotbugs.html
- Style check information in: https://github.com/yumeng-luo/COMS4156/blob/main/target/site/checkstyle.html
- For both tools, we only have clean reports as we have been fixing bugs/style issues incrementally since we began coding