USCDoorDrink

1. Project Title and Authors

a. Team number: 29
b. Team Name: Binary Asymmetry
c. Members Name:

Jingyi (Lily) Qiang USC ID: 2815038793
Jinghao (Daniel) Wang USC ID: 4237145509
Zhikai (Leo) Li USC ID: 3104080090

2. Preface

This android app, USCDoorDrink, is designed as an online drink ordering and delivery platform for two types of users in the USC community: merchants and customers. USCDoorDrink is a specialized app that only supports ordering and delivery of tea or coffee, helping customers and merchants to track their order histories.

3. Instruction

Location of test cases:
All Black-box testing test cases are located in the
USCDoorDrink/app/src/androidTest/java/com/csci310/uscdoordrink
All White-box testing test cases in testing files are located in
USCDoorDrink/app/src/test/java/com/csci310/uscdoordrink

How to execute the test cases:
Right the click the test case file and click run. All test cases are automated and could be executed multiple times. In case of unable to start certain test cases or getting constant fails, check for the emulator's internet connection or reboot the emulator. The are typically significant lags/delays when the emulator first boots up.

4. Black-box Testing

Number of Black-box Test Cases Summary (Total 29 test cases):

-Daniel: 8 test cases
-Lily: 11 test cases
-Leo: 10 test cases

(Leo) Login_UI.java: no bugs found in the testing process (5 test cases)
All test cases in this file will automatically execute upon running Login_UI.java

1.checkShowRegisterView()
-Description: tests whether the app will show regiter view after clicking the register button
-Rationale: simulate a click on the register button to make sure the register button is responsive and will lead to the correct view

2.checkSuccessFulLogin()
-Description: test whether the app will show the main page after loging in with the correct username/password combination
-Rationale: enter a pair of matching username/password thats already in the database then click login. This test makes sure the system is able to gloss the database, return permission to login, and show main page view

3.checkFailedLogin()
-Description: test whether the app will staying on login page and prompt for correct authentication info after attempting to log in in with the wrong username/password combination
-Rationale: enter a password/username pair does not exist in the data base and then click log in. This test ensures the system is able to detect wrong log in information, stay on the current page, and prompt the user to re-enter login information

4.checkMissingPassword()
-Description: test whether a error message will be display when attempting to log in without password
-Rationale: this ensures that when user attempt to login without password, the app will stay on the log in page and prompt the user to not leave password blank

5.checkMissingUserName()
-Description:test whether a error message will be display when attempting to log in without username
-Rationale: this ensures that when user attempt to login without username, the app will stay on the log in page and prompt the user to not leave username blank

(Leo) Register_UI.java: no bugs found in the testing process (4 test cases)
All test cases in this file will automatically execute upon running Register_UI.java

6.check_Merchant_Successful_Register()
-Description: test whether the app will show create menu page for merchant upon successfully resgtering a merchant account
-Rationale: enter name and password and check the “register as a merchant” checkbox. Then simulate a click on resgiter. This simuulate merchant registration and should lead to a create menu view and prompt the user to enter more information as it is required by a merchant account

7.check_Customer_Successful_Register()
-Description: test whether the app will show the main page for customer upon successfully registering a customer account
-Rationale: enter name and password and does not check the “register as a merchant” checkbox. Then simulate a click on resgiter. This simluates user registration and the app should then show the main page.

8.checkRegisterWithoutPassword()
-Description: test wheter the app stops registration without password and give a corresponding error message
-Rationale: only a username will be given and then a simulate click on register account. This tests for the app's ability to prevent user from regsitering account withou entering all information need. A waring message will show up.

9.checkRegisterWithoutUserName()
-Description: test wheter the app stops registration without username and give a corresponding error message
-Rationale:only a password will be given and then a simulate click on register account. This tests for the app's ability to prevent user from regsitering account withou entering all information need. A waring message will show up.

Create_Menu_UI.java: no bugs found in the testing process (3 test cases)
All test cases in this file will automatically execute upon running Create_Menu_UI.java

(Leo)10.a_check_Merchant_Add_Item_to_Menu()
-Description: test whether the ui will display the corresponding information after adding an item to the menu
-Rationale: enter item name, price, and caffeine then click add. The add item should show below the textview row. This test ensures that the UI will display the correct change upon adding an item to the menu.

(Daniel)11.b_check_Merchant_Remove_Item_from_Menu()
-Description: test whether the ui will display the corresponding information after adding an item to the menu then removing it
-Rationale: enter item name, price, and caffeine then click add. The add item should show below the textview row. Then click remove to remove it. The page should remain blank afterwards. This test ensures that the UI will display the correct change upon removing an item from the menu.

(Daneil)12.c_check_Merchant_Submit_Menu()
-Description: test whether the ui will display set location view upon clicking submit menu button
-Rationale: this

(Daniel) Set_Location_UI.java: no bugs found in the testing process (1 test case)
All test cases in this file will automatically execute upon running Set_Location_UI.java

13.check_Merchant_Successful_EnterLocation()
-Description: test whether the app will display main page for merchant upon enter location information
-Rationale: a set of longitude and latitude will be enter and the submit button will be clicked. This simulates the last step of registering a merchant account. The store should be displayed on the map once the main page is displayed

(Daniel) Menu_PlaceOrder_UI.java: no bugs found in the testing process (4 test cases)

All test cases in this file will automatically execute upon running Menu_PlaceOrder_UI.java

14.checkAddItemtoCart()

-Description: test whether the ui will show different item numbers after adding item to cart

-Rationale: two of the second item will be added to cart. This tests for the app's ability to display UI changes after multiple items to cart.

15.checkRemoveItemfromCart()

-Description: test whether the ui will show different item numbers after adding then removing items from cart

-Rationale: three of the second items will be added to the cart and then one will be removed. This tests for the app's ability to display correct modified cart information after adding and removing.

16.checkShowCheckoutView()

-Description: test whether the app is able to display the check out view after adding items to the cart and the checkout button is clicked

-Rationale: this test will first add a few items to the cart to ensure the cart is not empy then click the check out button to go to the checkout view. This simuates the user's shopping process

17.checkCompleteCheckout()

-Description: test whether the app is able to submit the order and go back to the main

-Rationale: this test will first add a few items to the cart to ensure the cart is not empy then click the check out button to go to the checkout view. Once the checkout view is displayed, the place order button will be clicked. The app then should submi the order and go back to the main page. This simuates the process of user checking out and adding a few but not a single items ensures the system works properly.

(Lily) BlackBoxLily.java: no bugs found in the testing process (11 test cases)

All test cases in this file will automatically execute upon running BlackBoxLily.java

Please set the location in the emulator to "usc":(lat:34.0209,lon:-118.2901) before launching the test. In case the location is not immediately adjusted in the emulator, in rare occasions you need to launch it one more time.

18.CustomerTest()

-Description: Test whether a customer can enter into the map page from login.

-Rationale: Used a customer account to login-in automatically, and used the MyLocation button to verify the customer has entered into the map page.Also verified the customer can click the mark,see the ETA, view menu button, and traveling routes on map.

19.MerchantTest()

-Description: Test whether a merchant can enter into the map page from login.

-Rationale: Used a merchant account to log-in automatically, and used the MyLocation button to verify the merchant has entered into the map page. Also verified the merchant can click the mark and view corresponding information.

20.CusHistoryButtonTest()
-Description: Imitating the action of a customer clicking the history button.
-Rationale: Use a customer account to log-in. Verified whether the history button is clickable, and also used assertEquals to show the button selected display the correct text. After entering into the history page, also verified we have entered into the correct page by clicking the button displayed on that page.

21.MHistoryButtonTest()
-Description:Imitating the action of a merchant clicking the history button.
-Rationale: Use a merchant account to log-in. Verified whether the history button is clickable, and also used assertEquals to show the button selected display the correct text. After entering into the history page, also verified we have entered into the correct page by clicking the button displayed on that page.

22.CusProfileButtonTest()
-Description: Imitating the action of a customer clicking the profile button
-Rationale: Use a customer account to log-in. Verified whether the profile button is clickable, and also used assertEquals to show the button selected display the correct text. After entering into the profile page, also verified we have entered into the correct page by clicking the button (such as the tab of different time period) on the profile page.

23.MProfileButtonTest()
-Description: Imitating the action of a merchant clicking the profile button
-Rationale: Use a merchant account to log-in. Verified the button displays the correct text using assert equals. Verified the profile button has no transition effect for the merchant by demonstrating that the merchant is still on the map page even after they clicked the button.

24.CusMapButtonTest()
-Description:Imitating the action of a customer clicking the map button
-Rationale: Use a customer account to log-in. Verified the button displays the correct text using assert equals. Verified the customer is still on the map page by showing the location button can be selected.

25.MMapButtonTest()
-Description:Imitating the action of a merchant clicking the map button
-Rationale: Use a merchant account to log-in. Verified the button displays the correct text using assert equals. Verified the customer is still on the map page by showing the location button can be selected.

26.MyLocationandDailyRecommendationTest()
-Description: Used a customer account to log-in, and test the mylocation button and recommendation tab

-Rationale: Verified the MyLocation button is clickable. Also verified the Recommendation Tab exists and has no clickable effects.

27.DirectionTest()
-Description: Verified the effect of the DirectionTest button
-Rationale: Used assertEquals to verify the button exists and is clickable.

28.ExternalMapTest()
-Description: Verified the effect of the External Map Button
-Rationale: Used assertEquals to verify the button exists and is clickable.

(Daniel) Place_Order_Excessive_Caffeine_UI.java: no bugs found in the testing process (1 test case)
All test cases in this file will automatically execute upon running Place_Order_Excessive_Caffeine_UI.java

29.CheckExcessiveCaffeineTest()
-Description: test that whether a toast message that warns customer for excessive caffeine intake will show up when checking for items that leads to excessive caffeine intake
-Rationale: created a separated user for this specified purpose. The same item with high caffeine content is added 21 times into the cart to ensure an absolute caffeine overdose. Then checkout is perfomed to see if the toast message shows up. Finally reattempt checkout to see if the order may be placed and ui goes back to main pagel

5. White-box Testing

Number of White-box Test Cases Summary (Total: 65 test cases):

-Lily: 22 test cases
-Daniel: 24 test cases
-Leo: 19 test cases

(Lily) TestItem.java: no bugs found in the testing process (11 test cases)
All test cases in this file will automatically execute upon running TestItem..java

1.getItemNameTest()
- Description: Testing the method getItemName() of the Item class to see if this method could correctly get the item's name.
- Result: Passed

2.setItemNameTest()
- Description: Testing the method setItemName() of the Item class to see if this method could correctly set the item's name.
- Result: Passed

3.getItemPriceTest()
- Description: Testing the method getItemPrice() of the Item class to see if this method could correctly get the item's price.

- Result: Passed

4.setItemPriceTest()
- Description: Testing the method setItemPrice() of the Item class to see if this method could correctly set the item's price.
- Result: Passed

5.getItemQtyInOrderTest1()
- Description: Testing the method getItemQtyInOrder() of the Item class to see if this method could correctly get the item's qty (with the itemQty being a parameter in the constructor)
-Result: Passed

6.getItemQtyInOrderTest2()
- Description: Testing the method getItemQtyInOrder() of the Item class to see if this method could correctly get the item's itemQty(without itemQty being a parameter in the constructor)
- Result: Passed

7.setItemQtyInOrderTest()
- Description: Testing the method setItemQtyInOrder() of the Item class to see if this method could correctly set the item's itemQtyInOrder.
- Result: Passed

8.getItemCaffeineTest()
- Description: Testing the method getItemCaffeine() of the Item class to see if this method could correctly get the item's itemCaffeine.
- Result: Passed

9.setItemCaffeineTest()
- Description: Testing the method setItemCaffeine() of the Item class to see if this method could correctly set the item's itemCaffeine.
- Result: Passed

10.constructorTest1()
- Description: Testing the constructor of the Item class (with the itemQty as one of its parameter) sets member vars correctly.
- Result: Passed

11.constructorTest2()
- Description: Testing the constructor of the Item class (without the itemQty as one of its parameter) sets member vars correctly.
- Result: Passed

(Lily) TestUser.java: no bugs found in the testing process (5 test cases)
All test cases in this file will automatically execute upon running TestUser.java

12. getUsrNameTest()

- Description: Testing the method getUsrName() of the User class to see if this method could correctly get the user's username.
- Result: Passed

13. getUsrPasswordTest()

-Description: Testing the method of getUsrPassword() of the User class to see if the method could correctly get the user's password.
-Result: Passed

14. constructorTest()

- Description:Testing the constructor of the User class to see if the method could correctly take user's username, password to create a User object.
- Result: Passed

15. setUsrNameTest()

-Description: Testing the method setUsrName() of the User class to see if this method could correctly set the user's username.
-Result: Passed

16. getUsrPasswordTest()

-Description: Testing the method setUsrPassword() of the User class to see if this method could correctly get the user's password.
-Result: Passed

(Lily) WhiteBoxLily.java: 1 tiny bugs found in the testing process (6 test cases)

All test cases in this file will automatically execute upon running WhiteBoxLily.java

17. GetDurationTest1()

-Description: Testing whether the function responsible for interacting with the API pulls down the result
-Result: Passed

18. insertDanielTest1()

-Description: Testing whether the function interact with the database correctly, and returns back the result
-Result: Passed

19. RecommendationTestNormal()

-Description: Testing whether the function interact with the database correctly given a normal customer name with order history, and returns back the result
-Result: Passed

20. RecommendationNoSuchCustomer()

-Description: Testing whether the function interacts with the database correctly given a customer name that does not exist in the database, and returns back the result
-Result: 1 tiny bug found when the resultset is empty. Has fixed the code

21. RecommendationTestNULL()
-Description: Tested whether the function interacts with the database correctly given a null input, and returns back the result
-Result: Passed

22. RecommendationNoOrder()
-Description: Tested whether the function interacts with the database correctly given a customer name that has no order, and returns back the result
-Result: Passed

(Daniel) TestDeliveryRoute.java: no bugs found in the testing process (13 test cases)
All test cases in this file will automatically execute upon running TestDeliveryRoute.java

23.getMerchantUsrNameTest()
-Description: Testing the method getMerchantUsrName() of the DeliveryRoute class to see if this method could correctly get merchant's username from the DeliveryRoute object.
-Result: Passed

24.setMerchantUsrNameTest()
-Description: Testing the method setMerchantUsrName() of the DeliveryRoute class to see if this method could correctly change merchant's username for the DeliveryRoute object.
-Result: Passed

25.getCustomerUsrNameTest()
-Description: Testing the method getCustomerUsrName() of the DeliveryRoute class to see if this method could correctly get customer's username from the DeliveryRoute object.
-Result: Passed

26.setCustomerUsrNameTest()
-Description: Testing the method setCustomerUsrName() of the DeliveryRoute class to see if this method could correctly change customer's username for the DeliveryRoute object.
-Result: Passed

27.getOrderPlacedDateTest()
-Description: Testing the method getOrderPlacedDate() of the DeliveryRoute class to see if this method could correctly get the order placed date from the DeliveryRoute object.
-Result: Passed

28.setOrderPlacedDateTest()

-Description: Testing the method setOrderPlacedDate() of the DeliveryRoute class to see if this method could correctly change the order placed date for the DeliveryRoute object.
-Result: Passed

29.getOrderPlacedTimeTest()
-Description: Testing the method getOrderPlacedTime() of the DeliveryRoute class to see if this method could correctly get the order placed time from the DeliveryRoute object.
-Result: Passed

30.setOrderPlacedTimeTest()
-Description: Testing the method setOrderPlacedTime() of the DeliveryRoute class to see if this method could correctly change the order placed time for the DeliveryRoute object.
-Result: Passed

31.getDeliveryDateTest()
-Description: Testing the method getDeliveryDate() of the DeliveryRoute class to see if this method could correctly get the order delivery date from the DeliveryRoute object.
-Result: Passed

32.setDeliveryDateTest()
-Description: Testing the method setDeliveryDate() of the DeliveryRoute class to see if this method could correctly change the order delivery date for the DeliveryRoute object.
-Result: Passed

33.getDeliveryTimeTest()
-Description: Testing the method getDeliveryTime() of the DeliveryRoute class to see if this method could correctly get the order delivery time from the DeliveryRoute object.
-Result: Passed

34.setDeliveryTimeTest()
-Description: Testing the method setDeliveryTime() of the DeliveryRoute class to see if this method could correctly change the order delivery time for the DeliveryRoute object.
-Result: Passed

35.constructorTest()
-Description: Testing the constructor of the DeliveryRoute class to see if the method could correctly take merchant's username, customer's username, order placed date, order placed time, order delivery date and order delivery time to create an DeliveryRoute object.
-Result: Passed

(Daniel) TestQuery.java: no bugs found in the testing process (11 test cases)
All test cases in this file will automatically execute upon running TestQuery.java

36.getMenuFromDatabaseTest()

-Description: Testing if we can run SQL query to get the menu information for one merchant from the table MENU in the database.
-Result: Passed

37.insertOrderIntoDatabaseTest()
-Description: Testing if we can run SQL query to insert the order information into the tables ORDER and ITEMS_IN_ORDER in the database.
-Result: Passed

38.getOrderFromDatabaseTest1()
-Description: Testing if we can run SQL query to get all orders information made by one specific customer from the table ORDER and ITEMS_IN_ORDER in the database
-Result: Passed

39.getOrderFromDatabaseTest2()
-Description: Testing if we can run SQL query to get all orders information for one specific merchant from the table ORDER and ITEMS_IN_ORDER in the database
-Result: Passed

40.getCaffeineTest()
-Description: Testing if we can run SQL query to get the caffeine intake amount for one specific customer from the table CUSTOMER in the database
-Result: Passed

41.updateCaffeineTest()
-Description: Testing if we can run SQL query to update the customer's caffeine intake amount after the customer placed an order, updating the CaffeineIntake in the table CUSTOMER of the database
-Result: Passed

42.refreshCaffeineTest1()
-Description: Testing if we can run SQL query to refresh the CaffeineIntake (update it to 0) in the table CUSTOMER of the database for the specific customer if the customer did not place any order on that day but is trying to place an order.
-Result: Passed

43.refreshCaffeineTest2()
-Description: Testing if we can run SQL query to keep the CaffeineIntake (does not update it to 0) in the table CUSTOMER of the database for the specific customer if the customer did place order on that day and is trying to place an order.
-Result: Passed

44.getDailyAnalysisTest()
-Description: Testing if we can run SQL query to get the total daily spending for each merchant a specific customer visited on that day from the table ORDER in the database..
-Result: Passed

45.getWeeklyAnalysisTest()
-Description: Testing if we can run SQL query to get weekly total spending summary for a specific customer if the customer placed an order on any day in the past 7 days, it will only show the day the customer placed an order from the table ORDER in the database..
-Result: Passed

46.getMonthlyAnalysisTest()
-Description:Testing if we can run SQL query to get monthly total spending summary for a specific customer if the customer placed an order in any week in the past 4 weeks, it will show how much the customer spent for each week from the table ORDER in the database..
-Result: Passed

(Leo) TestOrder.java: no bugs found in the testing process (8 test cases)
All test cases in this file will automatically execute upon running TestOrder.java

47.getTotalPriceTest()
-Description: Testing the method getTotalPrice() of the Order class to see if this method could correctly get the total item price from the Order object.
-Result: Passed

48.setTotalPriceTest()
-Description: Testing the method setTotalPrice() of the Order class to see if this method could correctly change the total item price for the Order object.
-Result: Passed

49.getTotalCaffeineTest()
-Description: Testing the method getTotalCaffeine() of the Order class to see if this method could correctly get the total caffeine.
-Result: Passed

50.setTotalCaffeineTest()
-Description: Testing the method setTotalCaffeine() of the Order class to see if this method could correctly set the total caefffeine.
-Result: Passed

51.getDeliveryRouteTest()
-Description: Testing the method getDeliveryRoute() of the Order class to see if this method could correctly get the delivery route.
-Result: Passed

52.setDeliveryRouteTest()
-Description: Testing the method setDeliveryRoute() of the Order class to see if this method could correctly set the delivery route.
-Result: Passed

53.getOrderItemsTest()
-Description: Testing the method getOrderItems() of the Order class to see if this method could correctly get items in order.
-Result: Passed

54.addOrderItemTest()
-Description: Testing the method addOrderItem() of the Order class to see if this method could correctly add item to order.
-Result: Passed

55.toStringTest()
-Description: Testing the method toString() of the Order class to see if this method could correctly add item to order.
-Result: Passed

(Leo) TestCart.java: no bugs found in the testing process (4 test cases)
All test cases in this file will automatically execute upon running TestCart.java

55.getItemsInCartTest()
-Description: Testing the method getItemsInCartTest() of the cart class to see if this method could correctly get items in cart.
-Result: Passed

56.constructorTest()
-Description: Testing the constructor of the cart class to see if the method could correctly take an arraylist of items and create an object of cart class
-Result: Passed

57.setItemsInCartTest()
-Description: Testing the method setItemsInCartTest() of the cart class to see if this method could correctly set items in cart.
-Result: Passed

58.addItemTest()
-Description: Testing the method addItemsInCartTest() of the cart class to see if this method could correctly add items in cart.
-Result: Passed

(Leo) TestInsertLeo.java: no bugs found in the testing process (3 test cases)

All test cases in this file will automatically execute upon running TestInsertLeo.java

59.insertIntoSqlTest1()

-Description: Test the insertIntoSqlTest1() of the InsertLeo class to see if this method will return false when attempting to insert a customer that already exists in the SQL user databse

-Result: Passed

60.insertIntoSqlTest2()

-Description: Test the insertIntoSqlTest2() of the InsertLeo class to see if this method could correctly insert a new customer in the SQL user databse and return true

-Result: Passed

61.insertIntoSqlTest3()

-Description: est the insertIntoSqlTest3() of the InsertLeo class to see if this method could correctly insert a new merchant in the SQL user databse and return true

-Result: Passed

62.insertIntoSqlTest4()

-Description: Test the insertIntoSqlTest4() of the InsertLeo class to see if this method will return false when attempting to insert a merchant that already exists in the SQL user databse

-Result: Passed

(Leo) TestInsertCustomer.java: no bugs found in the testing process (1 test case)

All test cases in this file will automatically execute upon running TestInsertCustomer.java

63.insertCusTest()

-Description: Test the insertCusTest() of the InsertCustomer class to see if this method could insert a new customer into the SQL customer databse

-Result: Passed

(Leo) TestInsertLocation.java: no bugs found in the testing process (1 test case)

All test cases in this file will automatically execute upon running TestInsertLocation.java

64.insertIntoSqlTest()

-Description: Test the insertIntoSqlTest() of the InsertLocation class to see if this method could insert a new merchant's location into the SQL merchant databse

-Result: Passed

(Leo) TestInsertMenu.java: no bugs found in the testing process (1 test case)

All test cases in this file will automatically execute upon running TestInsertMenu.java

65.InsertMenuTest()

-Description:Test the InsertMenuTest() of the InsertMenu class to see if this method could insert a menu into the SQL menu databse

-Result: Passed

Coverage of White-box Testing:
We are using path coverage for all white-box testing cases in testing files, the coverage level is 100%