Beverage Booker

Testing Manifest

Customer App

Use-Case	Test Type	Test Name	Test Results	Tester
Create User	Test Scripts	Test Case 1 - isCreateUserSuccesful	Test Case 1 Results	Emily Carter
	Postman Tests	Test Case 2 - isUserCreatedWithEmptyEmail	The **Local Server** test results are located here. The **Web Server** test results are located here.	Benn Curby
		Test Case 3 - isUserCreatedWithEmptyPassword		Benn Curby
		Test Case 4 - isUserCreatedWithEmptyFirstName		Benn Curby
		Test Case 5 - isUserCreatedWithEmptyLastName		Benn Curby
		Test Case 6 - isUserCreatedWithEmptyPhone		Benn Curby
		Test Case 7 - isUserCreatedWithValidData		Benn Curby
		Test Case 8 - isDuplicateUserCreatedWithValidData		Benn Curby
	Android Instrumented Tests	Test Case 9 - createUserTest_checkFields	The results for these tests are located here.	Benn Curby
		Test Case 10 - createUserTest_EmptyEmail		Benn Curby
		Test Case 11 - createUserTest_EmptyPassword		Benn Curby
		Test Case 12 - createUserTest_EmptyFirstName		Benn Curby
		Test Case 13 - createUserTest_EmptyLastName		Benn Curby
		Test Case 14 - createUserTest_EmptyPhone		Benn Curby
		Test Case 15 - createUserTest_IncorrectEmailFormat		Benn Curby
		Test Case 16 - createUserTest_PasswordUnderRequiredLength		Benn Curby

		Test Case 17 - createUserTest_SuccessfulAccountCreation		Benn Curby

Use-Case	Test Type	Test Name	Test Results	Tester
Log In	Test Scripts	Test Case 18 - isLogInSuccessful	Test Case 18 Results	Emily Carter
		Test Case 19 - isSharedPrefSuccesful	Test Case 19 Results	Jake Durnford
	Postman Tests	Test Case 20 - isUserLoggedInWithValidData	The **Local Server** test results are located here. The **Web Server** test results are located here.	Benn Curby
		Test Case 21 - isUserLoggedInWithIncorrectEmai		Benn Curby
		Test Case 22 - isUserLoggedInWithIncorrectPassword		Benn Curby
		Test Case 23 - isUserLoggedInWithEmptyEmail		Benn Curby
		Test Case 24 - isUserLoggedInWithEmptyPassword		Benn Curby
		Test Case 25 - isUserLoggedInWithEmptyEmailAndPassword		Benn Curby
	Android Instrumented Tests	Test Case 26 - loginTest_checkFields	The results for these tests are located here.	Benn Curby
		Test Case 27 - loginTest_EmptyEmail		Benn Curby
		Test Case 28 - loginTest_EmptyPassword		Benn Curby
		Test Case 29 - loginTest_IncorrectEmailFormat		Benn Curby
		Test Case 30 - loginTest_SuccessfulLogin		Benn Curby

Use-Case	Test Type	Test Name	Test Results	Tester
Browse Menu	Test Scripts	Test Case 31 - isMenuDisplayingCorrectlyInApp	Test Case 31 Results	Jake Durnford
		Test Case 32 - isMenuSentCorrectlyFromServer	Test Case 32 Results	Jake Durnford
	Postman Tests	Test Case 33 - areMenuItemsReturnedFromServer	The **Local Server** test results are located here. The **Web Server** test results are located here.	Benn Curby

Use-Case	Test Type	Test Name	Test Results	Tester
Fill Cart	Test Scripts	Test Case 34 - areCartItemsAddedToDatabase	Test Case 34 Results	Jacob Kennedy
		Test Case 35 - areCartItemsReturnedByUserID	Test Case 35 Results	Jacob Kennedy
		Test Case 36 - areCartItemsReturnedToApp	Test Case 36 Results	Jacob Kennedy
	Postman Tests	Test Case 37 - isItemAddedToCartWithUserIDEmpty	The **Local Server** test results are located here. The **Web Server** test results are located here.	Benn Curby
		Test Case 38 - isItemAddedToCartWithItemIDEmpty		Benn Curby
		Test Case 39 - isItemAddedToCartWithItemTitleEmpty		Benn Curby
		Test Case 40 - isItemAddedToCartWithItemPriceEmpty		Benn Curby
		Test Case 41 - isItemAddedToCartWithItemQuantityEmpty		Benn Curby
		Test Case 42 - isItemAddedToCartWithAllFieldsEmpty		Benn Curby
		Test Case 43 - isItemAddedToCartWithValidData		Benn Curby
		Test Case 44 - isItemAddedToCartWithValidDataButItemAlreadyInCart		Benn Curby
	Android Instrumented Tests	Test Case 45 - ClickAddToCartButton_AddsToCart	The results for these tests are located here.	Jacob Kennedy
		Test Case 46 - ClickViewCartButton_VerifiesCartOpenWithRecycler		Jacob Kennedy
		Test Case 47 - ClickViewCartButton_VerifiesCartOpensWithCartTotal		Jacob Kennedy
		Test Case 48 - ClickViewCartButton_VerifiesItemInCartWithName		Jacob Kennedy

		Test Case 49 - ClickViewCartButton_VerifiesItemInCartWithPrice		Jacob Kennedy

Use-Case	Test Type	Test Name	Test Results	Tester
Place Order	Postman Tests	Test Case 50 - isEmptyCartResponseCorrect	The **Local Server** test results are located here. The **Web Server** test results are located here.	Benn Curby
		Test Case 51 - isCartContentsResponseCorrect		Benn Curby
		Test Case 52 - isOrderWithValidFieldsPlacedSuccessfully		Benn Curby
		Test Case 53 - isOrderWithEmptyUserIDPlaced		Benn Curby
		Test Case 54 - isOrderWithEmptyCreditCardNumberPlaced		Benn Curby
		Test Case 55 - isOrderWithEmptyCVVPlaced		Benn Curby
		Test Case 56 - isOrderWithEmptyExpiryMonthPlaced		Benn Curby
		Test Case 57 - isOrderWithEmptyExpiryYearPlaced		Benn Curby
		Test Case 58 - isOrderWithEmptyOrderTotalPlaced		Benn Curby
		Test Case 59 - isOrderWithAllFieldsEmptyPlaced		Benn Curby
		Test Case 60 - isOrderWithDefaultPaymentValuesPlaced		Benn Curby
	Android Instrumented Tests	Test Case 61 - GoToCheckout	The results for these tests are located here.	Jacob Kennedy
		Test Case 62 - PaymentInformation_ErrorAtCardNumber_TooLittleNumbers		Jacob Kennedy
		Test Case 63 - PaymentInformation_ErrorAtCVV_TooLittleNumbers		Jacob Kennedy
		Test Case 64 - PaymentInformation_ErrorAtMonth_OutOfBounds		Jacob Kennedy
		Test Case 65 - PaymentInformation_ErrorAtYear_OutOfBounds		Jacob Kennedy

		Test Case 66 - zPaymentInformation_PlaceOrder_OrderConfirmation_MainMenu		Jacob Kennedy

Use-Case	Test Type	Test Results	Testers
All Customer Side Use-Cases	FATs	Please find our FAT document for the customer app use-case testing here	Benn Curby Jake Durnford Emily Carter Jacob Kennedy

Use-Case: Create User

Test Scripts:

Test Case 1 - isCreateUserSuccesful:

Tester: Emily Carter

Description:
The test script tries to create a new user account using valid user data. The expected result is the new user should be successfully created.

Pre-conditions:
1. The user cannot already exist in the database.
2. All input data is valid.
3. The connection to the local server is stable.

Post-conditions:
The new user record is created and the new user is visible in the local database 'users' table.

Data required:
Valid input user data for registering a new account.
Email: ecarter@gmail.com
First Name: Emily
Last Name: Carter
Phone: 0468516009

Test Case 1 Results

Postman Tests:

Overview:
The following Postman tests are part of a testing collection for Create User.
They were first tested against our local server, and then again against the web server once the API was launched.

The **Local Server** test results are located here.
The **Web Server** test results are located here.

Test Case 2 - isUserCreatedWithEmptyEmail:

Tester: Benn Curby

Description:
The test tries to create a new user sending valid user data to the server, but with the ‘email’ field empty.

Pre-conditions:
1. The user cannot already exist in the database.
2. All input data is valid, barring the empty ‘email’ field.
3. The connection to the local server is stable.

Post-conditions:
1. New user record is not created
2. 422 response code from the local server
3. Error status is true
4. Return message is correct from server

Data required:
Email:
Password: 123456
First Name: Jane
Last Name: Doe
Phone: 0402000111

Test Case 3 - isUserCreatedWithEmptyPassword:

Tester: Benn Curby

Description:
The test tries to create a new user sending valid user data to the server, but with the ‘password’ field empty.

Pre-conditions:
1. The user cannot already exist in the database.
2. All input data is valid, barring the empty ‘password’ field.
3. The connection to the local server is stable.

Post-conditions:
1. New user record is not created
2. 422 response code from the local server
3. Error status is true
4. Return message is correct from server

Data required:
Email: janedoe@gmail.com
Password:
First Name: Jane
Last Name: Doe
Phone: 0402000111

Test Case 4 - isUserCreatedWithEmptyFirstName:

Tester: Benn Curby

Description:
The test tries to create a new user sending valid user data to the server, but with the ‘first name’ field empty.

Pre-conditions:
1. The user cannot already exist in the database.
2. All input data is valid, barring the empty ‘first name’ field.
3. The connection to the local server is stable.

Post-conditions:
1. New user record is not created
2. 422 response code from the local server
3. Error status is true
4. Return message is correct from server

Data required:
Email: janedoe@gmail.com
Password: 123456
First Name:
Last Name: Doe
Phone: 0402000111

Test Case 5 - isUserCreatedWithEmptyLastName:

Tester: Benn Curby

Description:
The test tries to create a new user sending valid user data to the server, but with the ‘last name’ field empty.

Pre-conditions:
1. The user cannot already exist in the database.
2. All input data is valid, barring the empty ‘last name’ field.
3. The connection to the local server is stable.

Post-conditions:
1. New user record is not created
2. 422 response code from the local server
3. Error status is true
4. Return message is correct from server

Data required:
Email: janedoe@gmail.com
Password: 123456
First Name: Jane
Last Name:
Phone: 0402000111

Test Case 6 - isUserCreatedWithEmptyPhone:

Tester: Benn Curby

Description:
The test tries to create a new user sending valid user data to the server, but with the ‘phone’ field empty.

Pre-conditions:
1. The user cannot already exist in the database.
2. All input data is valid, barring the empty ‘phone’ field.
3. The connection to the local server is stable.

Post-conditions:
1. New user record is not created
2. 422 response code from the local server
3. Error status is true
4. Return message is correct from server

Data required:
Email: janedoe@gmail.com
Password: 123456
First Name: Jane
Last Name: Doe
Phone:

Test Case 7 - isUserCreatedWithValidData:

Tester: Benn Curby

Description:
The test tries to create a new user sending valid user data to the server.

Pre-conditions:
1. The user cannot already exist in the database.
2. All input data is valid.
3. The connection to the local server is stable.

Post-conditions:
1. New user record is not created
2. 422 response code from the local server
3. Error status is true
4. Return message is correct from server

Data required:
Email: johndoe@gmail.com
Password: 123456
First Name: John
Last Name: Doe
Phone: 0400222333

Test Case 8 - isDuplicateUserCreatedWithValidData:

Tester: Benn Curby

Description:
The test tries to create a new user, however the record already exists in the database.

Pre-conditions:
1. The user record already exists in the database.
2. All input data is valid.
3. The connection to the local server is stable.

Post-conditions:
1. New user record is not created
2. 403 response code from the local server
3. Error status is true
4. Return message is correct from server

Data required:
Email: johndoe@gmail.com
Password: 123456
First Name: John
Last Name: Doe
Phone: 0400222333

Android Instrumented Tests:

Overview:
The following Instrumented Android tests are part of a testing java file.
The results for these tests are located here.

Test Case 9 - createUserTest_checkFields:

Tester: Benn Curby

Description:
The test checks the fields for the create user UI are displaying correctly.

Pre-conditions: .
1. Android Studio emulator is open with the app.

Post-conditions:
1. The views are checked and are found to either be displaying correctly or failing to display.

Data required:
No input data required.

Test Case 10 - createUserTest_EmptyEmail:

Tester: Benn Curby

Description:
The test tries to create a new user using the Android emulator running the app. However, the ‘email’ field is empty.

Pre-conditions:
1. The user cannot already exist in the database.
2. All input data is valid, barring the empty ‘email’ field.
3. The connection to the local server is stable.

Post-conditions:
1. New user record is not created
2. Error status is true
3. Message displayed by the app is correct: “Email is required”

Data required:
Email:
Password: 123456
First Name: Jane
Last Name: Doe
Phone: 0402555444

Test Case 11 - createUserTest_EmptyPassword:

Tester: Benn Curby

Description:
The test tries to create a new user using the Android emulator running the app. However, the ‘password’ field is empty.

Pre-conditions:
1. The user cannot already exist in the database.
2. All input data is valid, barring the empty ‘password’ field.
3. The connection to the local server is stable.

Post-conditions:
1. New user record is not created
2. Error status is true
3. Message displayed by the app is correct: “Password required”

Data required:
Email: janedoe@gmail.com
Password:
First Name: Jane
Last Name: Doe
Phone: 0402555444

Test Case 12 - createUserTest_EmptyFirstName:

Tester: Benn Curby

Description:
The test tries to create a new user using the Android emulator running the app. However, the ‘first name’ field is empty.

Pre-conditions:
1. The user cannot already exist in the database.
2. All input data is valid, barring the empty ‘first name’ field.
3. The connection to the local server is stable.

Post-conditions:
1. New user record is not created
2. Error status is true
3. Message displayed by the app is correct: “First Name required”

Data required:
Email: janedoe@gmail.com
Password: 123456
First Name:
Last Name: Doe
Phone: 0402555444

Test Case 13 - createUserTest_EmptyLastName:

Tester: Benn Curby

Description:
The test tries to create a new user using the Android emulator running the app. However, the ‘last name’ field is empty.

Pre-conditions:
1. The user cannot already exist in the database.
2. All input data is valid, barring the empty ‘last name’ field.
3. The connection to the local server is stable.

Post-conditions:
1. New user record is not created
2. Error status is true
3. Message displayed by the app is correct: “Last Name required”

Data required:
Email: janedoe@gmail.com
Password: 123456
First Name: Jane
Last Name:
Phone: 0402555444

Test Case 14 - createUserTest_EmptyPhone:

Tester: Benn Curby

Description:
The test tries to create a new user using the Android emulator running the app. However, the ‘phone’ field is empty.

Pre-conditions:
1. The user cannot already exist in the database.
2. All input data is valid, barring the empty ‘phone’ field.
3. The connection to the local server is stable.

Post-conditions:
1. New user record is not created
2. Error status is true
3. Message displayed by the app is correct: “Phone Number required”

Data required:
Email: janedoe@gmail.com
Password: 123456
First Name: Jane
Last Name: Doe
Phone:

Test Case 15 - createUserTest_IncorrectEmailFormat:

Tester: Benn Curby

Description:
The test tries to create a new user using the Android emulator running the app. However, the ‘email’ field has incorrectly formatted data.

Pre-conditions:
1. The user cannot already exist in the database.
2. All input data is valid, barring the incorrectly formatted ‘email’ field.
3. The connection to the local server is stable.

Post-conditions:
1. New user record is not created
2. Error status is true
3. Message displayed by the app is correct: “Enter a valid email”

Data required:
Email: janedoe.com
Password: 123456
First Name: Jane
Last Name: Doe
Phone: 0402555444

Test Case 16 - createUserTest_PasswordUnderRequiredLength:

Tester: Benn Curby

Description:
The test tries to create a new user using the Android emulator running the app. However, the ‘password’ entered is under the required 6 characters.

Pre-conditions:
1. The user cannot already exist in the database.
2. All input data is valid, barring the incorrect length ‘password’ field.
3. The connection to the local server is stable.

Post-conditions:
1. New user record is not created
2. Error status is true
3. Message displayed by the app is correct: “Password should be at least 6 characters long”

Data required:
Email: janedoe@gmail.com
Password: 12345
First Name: Jane
Last Name: Doe
Phone: 0402555444

Test Case 17 - createUserTest_SuccessfulAccountCreation:

Tester: Benn Curby

Description:
The test tries to create a new user using the Android emulator running the app. All input data is valid.

Pre-conditions:
1. The user cannot already exist in the database.
2. All input data is valid.
3. The connection to the local server is stable.

Post-conditions:
1. New user record is created
2. 201 response code from the local server
2. Error status is false
3. Message displayed by the app is correct: “Password should be at least 6 characters long”

Data required:
Email: janedoe@gmail.com
Password: 123456
First Name: Jane
Last Name: Doe
Phone: 0402555444

Use-Case: Log In

Test Scripts:

Test Case 18 - isLoginSuccesful:

Tester: Emily Carter

Description:
The test script tries to log in an existing user using valid user data. The expected result is the user should be logged in..

Pre-conditions:
1. The user already exists in the database.
2. All input data is valid.
3. The connection to the local server is stable.

Post-conditions:
The user is logged in with a successful login response message shown.

Data required:
Valid input user data for logging into an account.
Email: ecarter@gmail.com
Password: 123456

Test Case 18 Results

Test Case 19 - isSharedPrefSuccesful:

Tester: Jake Durnford

Description:
The test script tries to log in an existing user using valid user data. The expected result is the user should be logged in.

Pre-conditions:
1. The user already exists in the database.
2. All input data is valid.
3. The connection to the local server is stable.

Post-conditions:
The user is logged in with a successful login response message shown.

Test Case 19 Results

Postman Tests:

Overview:
The following Postman tests are part of a testing collection for Log In.
They were first tested against our local server, and then again against the web server once the API was launched.

The **Local Server** test results are located here.
The **Web Server** test results are located here.

Test Case 20 - isUserLoggedInWithValidData:

Tester: Benn Curby

Description:
The test tries to log in an existing user. It sends valid user credentials to the server.

Pre-conditions:
1. The user already exists in the database.
2. All input data is valid.
3. The connection to the server is stable.

Post-conditions:
1. User record is logged in.
2. 202 response code from the server
3. Error status is false
4. Return message is correct from the server

Data required:
Email: benn@gmail.com
Password: 123456

Test Case 21 - isUserLoggedInWithIncorrectEmail:

Tester: Benn Curby

Description:
The test tries to log in an existing user. It sends valid user credentials to the server, but with an incorrect email address.

Pre-conditions:
1. The user already exists in the database.
2. All input data is valid barring the incorrect email address.
3. The connection to the server is stable.

Post-conditions:
1. User record is not logged in.
2. 200 response code from the server
3. Error status is true
4. Return message is correct from the server

Data required:
Email: benn2@gmail.com
Password: 123456

Test Case 22 - isUserLoggedInWithIncorrectPassword:

Tester: Benn Curby

Description:
The test tries to log in an existing user. It sends valid user credentials to the server, but with an incorrect password.

Pre-conditions:
1. The user already exists in the database.
2. All input data is valid barring the incorrect email address.
3. The connection to the server is stable.

Post-conditions:
1. User record is not logged in.
2. 401 response code from the server
3. Error status is true
4. Return message is correct from the server

Data required:
Email: benn2@gmail.com
Password: 12345

Test Case 23 - isUserLoggedInWithEmptyEmail:

Tester: Benn Curby

Description:
The test tries to log in an existing user. It sends a valid user password to the server, but with an empty email field.

Pre-conditions:
1. The user already exists in the database.
2. All input data is valid barring the empty email address.
3. The connection to the server is stable.

Post-conditions:
1. User record is not logged in.
2. 422 response code from the server
3. Error status is true
4. Return message is correct from the server

Data required:
Email:
Password: 123456

Test Case 24 - isUserLoggedInWithEmptyPassword:

Tester: Benn Curby

Description:
The test tries to log in an existing user. It sends a valid user email to the server, but with an empty password field.

Pre-conditions:
1. The user already exists in the database.
2. All input data is valid barring the empty password address.
3. The connection to the server is stable.

Post-conditions:
1. User record is not logged in.
2. 422 response code from the server
3. Error status is true
4. Return message is correct from the server

Data required:
Email: benn@gmail.com
Password:

Test Case 25 - isUserLoggedInWithEmptyEmailAndPassword:

Tester: Benn Curby

Description:
The test tries to log in an existing user. It sends both an empty email and password field.

Pre-conditions:
1. All input data is invalid.
2. The connection to the server is stable.

Post-conditions:
1. User record is not logged in.
2. 422 response code from the server
3. Error status is true
4. Return message is correct from the server

Data required:
Email:
Password:

Android Instrumented Tests:

Overview:
The following Instrumented Android tests are part of a testing java file.
The results for these tests are located here.

Test Case 26 - loginTest_checkFields:

Tester: Benn Curby

Description:
The test checks the fields for the Log In UI are displaying correctly.

Pre-conditions: .
1. Android Studio emulator is open with the app.

Post-conditions:
1. The views are checked and are found to either be displaying correctly or failing to display.

Data required:
No input data required.

Test Case 27 - loginTest_EmptyEmail:

Tester: Benn Curby

Description:
The test tries to log in using the Android emulator running the app. However, the ‘email’ field is empty.

Pre-conditions:
1. Password is valid, but the ‘email’ field is empty.
3. The connection to the local server is stable.

Post-conditions:
1. User is not logged in
2. Error status is true
3. Message displayed by the app is correct: “Email is required”

Data required:
Email:
Password: 123456

Test Case 28 - loginTest_EmptyPassword:

Tester: Benn Curby

Description:
The test tries to log in using the Android emulator running the app. However, the ‘password’ field is empty.

Pre-conditions:
1. User already exists in the database.
2. Password is empty, but the ‘email’ field is valid.
3. The connection to the local server is stable.

Post-conditions:
1. User is not logged in
2. Error status is true
3. Message displayed by the app is correct: “Password is required”

Data required:
Email: janedoe@gmail.com
Password:

Test Case 29 - loginTest_IncorrectEmailFormat:

Tester: Benn Curby

Description:
The test tries to log in using the Android emulator running the app. However, the ‘email’ field is incorrectly formatted.

Pre-conditions:
1. Password is valid, but the ‘email’ field is incorrectly formatted.
3. The connection to the local server is stable.

Post-conditions:
1. User is not logged in
2. Error status is true
3. Message displayed by the app is correct: “Please enter a valid email”

Data required:
Email: janedoe.com
Password: 123456

Test Case 30 - loginTest_SuccessfulLogin:

Tester: Benn Curby

Description:
The test tries to log in using the Android emulator running the app. Login credentials sent to the server are valid.

Pre-conditions:
1. User already exists in the database.
1. Login credentials are valid.
3. The connection to the local server is stable.

Post-conditions:
1. User is logged in
2. Error status is false

Data required:
Email: janedoe@gmail.com
Password: 123456

Use-Case: Browse Menu

Test Scripts:

Test Case 31 - isMenuDisplayingCorrectlyInApp:

Tester: Jake Durnford

Description:
The test script checks that Browse Menu is functioning correctly. The expected result is menu items should be sent to the app from the database.

Pre-conditions:
1. The menu items already exist in the database.
2. The connection to the local server is stable.

Post-conditions:
The menu items are displayed in the created debug logs.

Test Case 31 Results

Test Case 32 - isMenuSentCorrectlyFromServer:

Tester: Jake Durnford

Description:
The test script checks that Browse Menu is functioning correctly in conjunction with the server. The expected result is menu items should be sent to Postman from the database.

Pre-conditions:
1. The menu items already exist in the database.
2. The connection to the local server is stable.

Post-conditions:
The menu items are displayed in the return body of the Postman ‘get’ request.

Test Case 32 Results

Postman Tests:

Overview:
The following Postman tests are part of a testing collection for Browse Menu.
They were first tested against our local server, and then again against the web server once the API was launched.

The **Local Server** test results are located here.
The **Web Server** test results are located here.

Test Case 33 - areMenuItemsReturnedFromServer:

Tester: Benn Curby

Description:
The test tries to get menu items from the database.

Pre-conditions:
1. The menu items exist in the database.
2. The connection to the server is stable.

Post-conditions:
1. 200 response code from the server
2. Response body is present

Use-Case: Fill Cart

Test Scripts:

Test Case 34 - areCartItemsAddedToDatabase:

Tester: Jacob Kennedy

Description:
Tests for the addition of items to the cart database using Postman.

Pre-conditions:
1. An established database with ‘cart’ table.
2. The connection to the local server is stable.

Post-conditions:
It either gives an error response if the items aren’t added to cart successfully, or it passes and the items are displayed in the ‘cart’ table of the database.

Test Case 34 Results

Test Case 35 - areCartItemsReturnedByUserID:

Tester: Jacob Kennedy

Description:
Tests for the retrieval of cart items from the ‘cart’ table using user ID.

Pre-conditions:
1. An established database with ‘cart’ table.
2. The connection to the local server is stable.

Post-conditions:
It either gives an error response if the items aren’t returned successfully, or it passes and the items are displayed.

Test Case 35 Results

Test Case 36 - areCartItemsReturnedToApp:

Tester: Jacob Kennedy

Description:
Tests for cart items being displayed correctly within the Android Customer App.

Pre-conditions:
1. An established database with ‘cart’ table.
2. Cart items within the ‘cart’ table.
2. The connection to the local server is stable.

Post-conditions:
Cart items are either successfully displayed in the debug logs, or if the test fails the items are not displayed.

Test Case 36 Results

Postman Tests:

Overview:
The following Postman tests are part of a testing collection for Fill Cart.
They were first tested against our local server, and then again against the web server once the API was launched.

The **Local Server** test results are located here.
The **Web Server** test results are located here.

Test Case 37 - isItemAddedToCartWithUserIDEmpty:

Tester: Benn Curby

Description:
The test tries to add an item to cart sending valid data to the server, but with the ‘userID’ field empty.

Pre-conditions:
1. All test data is valid, barring the empty ‘userID’ field.
2. The connection to the local server is stable.

Post-conditions:
1. Item is not added to cart
2. 422 response code from the local server
3. Error status is true
4. Return message is correct from server

Data required:
userID:
itemID: 5
itemTitle: milkshake
itemPrice: 5.90
itemQuantity: 1

Test Case 38 - isItemAddedToCartWithItemIDEmpty:

Tester: Benn Curby

Description:
The test tries to add an item to cart sending valid data to the server, but with the ‘itemID’ field empty.

Pre-conditions:
1. All test data is valid, barring the empty ‘itemID’ field.
2. The connection to the local server is stable.

Post-conditions:
1. Item is not added to cart
2. 422 response code from the local server
3. Error status is true
4. Return message is correct from server

Data required:
userID: 3
itemID:
itemTitle: milkshake
itemPrice: 5.90
itemQuantity: 1

Test Case 39 - isItemAddedToCartWithItemTitleEmpty:

Tester: Benn Curby

Description:
The test tries to add an item to cart sending valid data to the server, but with the ‘itemTitle’ field empty.

Pre-conditions:
1. All test data is valid, barring the empty ‘itemTitle’ field.
2. The connection to the local server is stable.

Post-conditions:
1. Item is not added to cart
2. 422 response code from the local server
3. Error status is true
4. Return message is correct from server

Data required:
userID: 3
itemID: 5
itemTitle:
itemPrice: 5.90
itemQuantity: 1

Test Case 40 - isItemAddedToCartWithItemPriceEmpty:

Tester: Benn Curby

Description:
The test tries to add an item to cart sending valid data to the server, but with the ‘itemPrice’ field empty.

Pre-conditions:
1. All test data is valid, barring the empty ‘itemPrice’ field.
2. The connection to the local server is stable.

Post-conditions:
1. Item is not added to cart
2. 422 response code from the local server
3. Error status is true
4. Return message is correct from server

Data required:
userID: 3
itemID: 5
itemTitle: milkshake
itemPrice:
itemQuantity: 1

Test Case 41 - isItemAddedToCartWithItemQuantityEmpty:

Tester: Benn Curby

Description:
The test tries to add an item to cart sending valid data to the server, but with the ‘itemQuantity’ field empty.

Pre-conditions:
1. All test data is valid, barring the empty ‘itemQuantity’ field.
2. The connection to the local server is stable.

Post-conditions:
1. Item is not added to cart
2. 422 response code from the local server
3. Error status is true
4. Return message is correct from server

Data required:
userID: 3
itemID: 5
itemTitle: milkshake
itemPrice: 5.90
itemQuantity:

Test Case 42 - isItemAddedToCartWithAllFieldsEmpty:

Tester: Benn Curby

Description:
The test tries to add an item to cart with all fields empty.

Pre-conditions:
1. All test data is invalid.
2. The connection to the local server is stable.

Post-conditions:
1. Item is not added to cart
2. 422 response code from the local server
3. Error status is true
4. Return message is correct from server

Data required:
userID:
itemID:
itemTitle:
itemPrice:
itemQuantity:

Test Case 43 - isItemAddedToCartWithValidData:

Tester: Benn Curby

Description:
The test tries to add an item to cart with valid data.

Pre-conditions:
1. All test data is valid.
2. The connection to the local server is stable.

Post-conditions:
1. Item is added to cart
2. 200 response code from the local server
3. Error status is false
4. Return message is correct from server

Data required:
userID: 3
itemID: 5
itemTitle: milkshake
itemPrice: 5.90
itemQuantity: 1

Test Case 44 - isItemAddedToCartWithValidDataButItemAlreadyInCart:

Tester: Benn Curby

Description:
The test tries to add an item to cart with valid data, but the item already exists in the users cart.

Pre-conditions:
1. All test data is valid.
2. Item already exists in users cart.
2. The connection to the local server is stable.

Post-conditions:
1. Item is not added to cart
2. 403 response code from the local server
3. Error status is false
4. Return message is correct from server

Data required:
userID: 3
itemID: 5
itemTitle: milkshake
itemPrice: 5.90
itemQuantity: 1

Android Instrumented Tests:

Overview:
The following Instrumented Android tests are part of a testing java file.
The results for these tests are located here.

Test Case 45 - ClickAddToCartButton_AddsToCart:

Tester: Jacob Kennedy

Description:
The test checks the fields for the Browse Menu UI are displaying correctly.

Pre-conditions: .
1. Android Studio emulator is open with the app.

Post-conditions:
1. The views are checked and are found to either be displaying correctly or failing to display.

Data required:
No input data required.

Test Case 46 - ClickViewCartButton_VerifiesCartOpenWithRecycler:

Tester: Jacob Kennedy

Description:
The test verifies that when the viewCart button is clicked, the cartRecyclerView is displayed.

Pre-conditions:
1. Android Studio emulator is open with the app.

Post-conditions:
1. cartRecyclerView displays correctly.

Data required:
No input data required.

Test Case 47 - ClickViewCartButton_VerifiesCartOpensWithCartTotal:

Tester: Jacob Kennedy

Description:
The test verifies that when the viewCart button is clicked, the 'Cart Total' is displayed.

Pre-conditions:
1. Android Studio emulator is open with the app.

Post-conditions:
1. 'Cart Total' displays correctly.

Data required:
No input data required.

Test Case 48 - ClickViewCartButton_VerifiesItemInCartWithName:

Tester: Jacob Kennedy

Description:
The test verifies that when the viewCart button is clicked, the cart item's name is displayed.

Pre-conditions:
1. Android Studio emulator is open with the app.

Post-conditions:
1. Cart item displays correctly.

Data required:
No input data required.

Test Case 49 - ClickViewCartButton_VerifiesItemInCartWithPrice:

Tester: Jacob Kennedy

Description:
The test verifies that when the viewCart button is clicked, the cart item’s price is displayed.

Pre-conditions:
1. Android Studio emulator is open with the app.

Post-conditions:
1. Cart item displays correctly.

Data required:
No input data required.

Use-Case: Place Order

Postman Tests:

Overview:
The following Postman tests are part of a testing collection for Place Order.
They were first tested against our local server, and then again against the web server once the API was launched.

The **Local Server** test results are located here.
The **Web Server** test results are located here.

Test Case 50 - isEmptyCartResponseCorrect:

Tester: Benn Curby

Description:
The test checks that the correct response is given when the user moves to the cart activity without adding any items to their cart.

Pre-conditions:
1. The user already exists in the database.
2. The user cart is empty.
3. The connection to the server is stable.

Post-conditions:
1. 303 response code from the server
2. Error status is false
3. Return message is correct from server

Data required:
userID: 4

Test Case 51 - isCartContentsResponseCorrect:

Tester: Benn Curby

Description:
The test checks that the correct response is given when the user moves to the cart activity with items in their cart.

Pre-conditions:
1. The user already exists in the database.
2. The user cart has at least one item in it.
3. The connection to the server is stable.

Post-conditions:
1. 201 response code from the server
2. Error status is false
3. Response body of menu items

Data required:
userID: 3

Test Case 52 - isOrderWithValidFieldsPlacedSuccessfully:

Tester: Benn Curby

Description:
Happy day scenario - Checks that an order with valid parameters can be placed successfully.

Pre-conditions:
1. The user already exists in the database.
2. The user cart has at least one item in it.
3. The user payment details are valid.
3. The connection to the server is stable.

Post-conditions:
1. 201 response code from the server
2. Error status is false
3. Return message is correct from server

Data required:
userID: 3
credit card number: 1111222233334444
credit card CVV: 123
expiry month: 12
expiry year: 20
order total: 5.90

Test Case 53 - isOrderWithEmptyUserIDPlaced:

Tester: Benn Curby

Description:
Tries to place an order with the userID parameter empty.

Pre-conditions:
1. The user payment details are valid.
2. The connection to the server is stable.

Post-conditions:
1. 422 response code from the server
2. Error status is true
3. Return message is correct from server

Data required:
userID:
credit card number: 1111222233334444
credit card CVV: 123
expiry month: 12
expiry year: 20
order total: 5.90

Test Case 54 - isOrderWithEmptyCreditCardNumberPlaced:

Tester: Benn Curby

Description:
Tries to place an order with the credit card number parameter empty.

Pre-conditions:
1. The user exists in the database.
2. The user payment details are valid, except for the empty credit card number.
3. The connection to the server is stable.

Post-conditions:
1. 422 response code from the server
2. Error status is true
3. Return message is correct from server

Data required:
userID: 3
credit card number:
credit card CVV: 123
expiry month: 12
expiry year: 20
order total: 5.90

Test Case 55 - isOrderWithEmptyCVVPlaced:

Tester: Benn Curby

Description:
Tries to place an order with the credit card CVV parameter empty.

Pre-conditions:
1. The user exists in the database.
2. The user payment details are valid, except for the empty credit card CVV.
3. The connection to the server is stable.

Post-conditions:
1. 422 response code from the server
2. Error status is true
3. Return message is correct from server

Data required:
userID: 3
credit card number: 1111222233334444
credit card CVV:
expiry month: 12
expiry year: 20
order total: 5.90

Test Case 56 - isOrderWithEmptyExpiryMonthPlaced:

Tester: Benn Curby

Description:
Tries to place an order with the credit card expiry month parameter empty.

Pre-conditions:
1. The user exists in the database.
2. The user payment details are valid, except for the empty credit card expiry month.
3. The connection to the server is stable.

Post-conditions:
1. 422 response code from the server
2. Error status is true
3. Return message is correct from server

Data required:
userID: 3
credit card number: 1111222233334444
credit card CVV: 123
expiry month:
expiry year: 20
order total: 5.90

Test Case 57 - isOrderWithEmptyExpiryYearPlaced:

Tester: Benn Curby

Description:
Tries to place an order with the credit card expiry year parameter empty.

Pre-conditions:
1. The user exists in the database.
2. The user payment details are valid, except for the empty credit card expiry year.
3. The connection to the server is stable.

Post-conditions:
1. 422 response code from the server
2. Error status is true
3. Return message is correct from server

Data required:
userID: 3
credit card number: 1111222233334444
credit card CVV: 123
expiry month: 12
expiry year:
order total: 5.90

Test Case 58 - isOrderWithEmptyOrderTotalPlaced:

Tester: Benn Curby

Description:
Tries to place an order with the order total parameter empty.

Pre-conditions:
1. The user exists in the database.
2. The user payment details are valid.
3. The order total is empty.
4. The connection to the server is stable.

Post-conditions:
1. 422 response code from the server
2. Error status is true
3. Return message is correct from server

Data required:
userID: 3
credit card number: 1111222233334444
credit card CVV: 123
expiry month: 12
expiry year: 20
order total:

Test Case 59 - isOrderWithAllFieldsEmptyPlaced:

Tester: Benn Curby

Description:
Tries to place an order with all the fields empty.

Pre-conditions:
1. All fields are empty.
4. The connection to the server is stable.

Post-conditions:
1. 422 response code from the server
2. Error status is true
3. Return message is correct from server

Data required:
userID:
credit card number:
credit card CVV:
expiry month:
expiry year:
order total:

Test Case 60 - isOrderWithDefaultPaymentValuesPlaced:

Tester: Benn Curby

Description:
Tries to place an order with all the fields empty.

Pre-conditions:
1. The user exists in the database.
2. The user payment details are equal to zero (default value).
3. The order total is equal to zero.
4. The connection to the server is stable.

Post-conditions:
1. 422 response code from the server
2. Error status is true
3. Return message is correct from server

Data required:
userID: 3
credit card number: 0
credit card CVV: 0
expiry month: 0
expiry year: 0
order total: 0

Android Instrumented Tests:

Overview:
The following Instrumented Android tests are part of a testing java file.
The results for these tests are located here.

Test Case 61 - GoToCheckout:

Tester: Jacob Kennedy

Description:
The test checks the fields for the Place Order UI screens are displaying correctly.

Pre-conditions: .
1. Android Studio emulator is open with the app.

Post-conditions:
1. The views are checked and are found to either be displaying correctly or failing to display.

Data required:
No input data required.

Test Case 62 - PaymentInformation_ErrorAtCardNumber_TooLittleNumbers:

Tester: Jacob Kennedy

Description:
The test verifies that the input validation for the credit card is working, checking that the card must equal 16 digits.

Pre-conditions: .
1. Android Studio emulator is open with the app.

Post-conditions:
1. The credit card number length is checked and is found to be invalid.
2. The error message is displayed: “Credit card number must be 16 digits”.

Data required:
No input data required.

Test Case 63 - PaymentInformation_ErrorAtCVV_TooLittleNumbers:

Tester: Jacob Kennedy

Description:
The test verifies that the input validation for the credit card CVV is working, checking that the CVV must equal 3 digits.

Pre-conditions: .
1. Android Studio emulator is open with the app.

Post-conditions:
1. The credit card CVV length is checked and is found to be invalid.
2. The error message is displayed: “CVV number must be 3 digits”.

Data required:
No input data required.

Test Case 64 - PaymentInformation_ErrorAtMonth_OutOfBounds:

Tester: Jacob Kennedy

Description:
The test verifies that the input validation for the credit card expiry month is working, checking that the month is a number between 01 and 12.

Pre-conditions: .
1. Android Studio emulator is open with the app.

Post-conditions:
1. The credit card expiry month is checked and is found to be invalid.
2. The error message is displayed: “Expiry month must be a number between 01 and 12”.

Data required:
No input data required.

Test Case 65 - PaymentInformation_ErrorAtYear_OutOfBounds:

Tester: Jacob Kennedy

Description:
The test verifies that the input validation for the credit card expiry year is working, checking that the year is a number greater than or equal to 20.

Pre-conditions: .
1. Android Studio emulator is open with the app.

Post-conditions:
1. The credit card expiry year is checked and is found to be invalid.
2. The error message is displayed: “Expiry year must be a number equal to current year or greater”.

Data required:
No input data required.

Test Case 66 - zPaymentInformation_PlaceOrder_OrderConfirmation_MainMenu:

Tester: Jacob Kennedy

Description:
The test verifies that an order with valid data can be placed.

Pre-conditions: .
1. Android Studio emulator is open with the app.

Post-conditions:
1. The order is placed by the app and the app moves to the order confirmation screen

Data required:
No input data required.