SC2006 Software Engineering

Test Cases

for

ShopBlock

Version 1.0 approved

Prepared by Group 45 NTU 13/10/2024

Group Members (Group 45)	
Name	Matriculation No.
Kua Li Min	U2322513H
Hudzaifah Bin Muhammad Taufiq	U2320600F
Cheng Hui Wen	U2322123G
Ng Hoe Ping	U2321991F
Seow Jia Xian Jackson	U2322995F
Chua Ze Ming	U2321797B
Zhang Yuxuan	U2321475L

Table of Contents

1. Black Box Testing

1.1 Controller

1.2 Equivalence Class and Boundary Value Testing

1.2 Test Cases and Results

1.2.1 User Account Management System

1.2.1.1 Sign Up / Register

Test Case ID	**T-1.2.1.1**	**Test Case Priority**	High
Test Case Description	Create an account for user		
Prerequisite	1. User should have their credentials ready for registering for an account	**Postrequisite**	1. User will be redirected to login page
Test Execution			

#	**Description**	**Action**	**Input**	**Expected Output**	**Actual Output**	**Test Result**
1	Creating a new account	User enters username, email, password and phone number and tick the terms and conditions box User clicks on Create Account button	Username: user1 Email: user1@gmail.com Password: P@ssword1 Phone number: 98765432 Terms & Conditions: ✅	New account is created	New account is created	Pass
2	Submitting an empty form	User clicks on Create Account button without any inputs	NIL	Displays “All fields are required.”	Displays “All fields are required.”	Pass
3	Entering a username	User enters a username with less	Username: 123	Displays “Username	Displays “Username must	Pass

	with less than 4 characters	than 4 characters, email, password and phone number and tick the terms and conditions box User clicks on Create Account button	Email: 123@gmail.com Password: P@ssword1 Phone number: 98765432 Terms & Conditions: ✅	must be at least 4 characters long.”	be at least 4 characters long.”	
4	Entering an invalid email	User enters a username, invalid email, password and phone number and tick the terms and conditions box User clicks on Create Account button	Username: 1234 Email: 1234@gmail Password: P@ssword1 Phone number: 98765432 Terms & Conditions: ✅	Displays “Please enter a valid email address.’”	Displays “Please enter a valid email address.’”	Pass
5	Entering a password with does not meet the requirements of 1 lower, 1 upper, 1 digit and 1 special character	User enters a username, email, basic password and phone number and tick the terms and conditions box User clicks on Create Account button	Username: 1234 Email: 1234@gmail.com Password: password Phone number: 98765432 Terms & Conditions: ✅	Displays “Password does not meet the password requirements.’ ”	Displays “Password does not meet the password requirements.’”	Pass
6	Did not check terms and conditions box	User enters a username, email, password and phone number but does not tick the terms and conditions box User clicks on Create Account button	Username: 1234 Email: 1234@gmail.com Password: P@ssword1 Phone number: 98765432	Displays “You must agree to the terms and conditions.”	Displays “You must agree to the terms and conditions.’”	Pass

			Terms & Conditions:			
7	Enter a phone number that belongs to another account	User enters a username, email, password and phone number and tick the terms and conditions box User clicks on Create Account button	Username: 1234 Email: 1234@gmail.com Password: P@ssword1 Phone number: 99999999 Terms & Conditions: ✅	Displays “An account with this phone number exists!”	Displays “An account with this phone number exists!”	Pass
8	Enter an email that belongs to another account	User enters a username, email, password and phone number and tick the terms and conditions box User clicks on Create Account button	Username: 1234 Email: user1@gmail.com Password: P@ssword1 Phone number: 98765432 Terms & Conditions: ✅	Displays “An account with this email already exists.”	Displays “An account with this email already exists.”	Pass
Test Case Status		Success				

1.2.1.2 Login

Test Case ID	**T-1.2.1.2**	**Test Case Priority**	High
Test Case Description	Login as a user		
Prerequisite	1. User should have their login credentials ready.	**Postrequisite**	1. User will be redirected to home page with his account logged in
Test Execution			

#	Description	Action	Input	Expected Output	Actual Output	Test Result
1	Login with valid account details	Enter email and password User clicks on Log In button	Email: user1@gmail.com Password: P@ssword1	Redirected to home page with account logged in	Redirected to home page with account logged in	Pass
2	Submitting an empty form	User clicks on Log In button without any inputs	NIL	Displays “All fields are required.”	Displays “All fields are required.”	Pass
3	Enter a email that is not registered	Enter a unregistered email and a password User clicks on Log In button	Email: qwerty@gmail.com Password: qwerty	Displays “No account found with this email address.”	Displays “No account found with this email address.”	Pass
4	Enter a registered email with incorrect password	Enter a registered email with incorrect password User clicks on Log In button	Email: user1@gmail.com Password: password	Displays “Incorrect Password! Please try again.”	Displays “Incorrect Password! Please try again.”	Pass
5	Enter a registered email with incorrect password and submits the form 3 times	Enter a registered email with incorrect password User clicks on Log In button 3 times	Email: user1@gmail.com Password: password	Displays “Your account will be locked after two more attempts.”	Displays “Your account will be locked after two more attempts.”	Pass
6	Enter a registered email with incorrect password and submits the form 4 times	Enter a registered email with incorrect password User clicks on Log In button 4 times	Email: user1@gmail.com Password: password	Displays “Your account will be locked after one more attempt.”	Displays “Your account will be locked after one more attempt.”	Pass
7	Enter a registered email with incorrect password and submits the form 5 times	Enter a registered email with incorrect password User clicks on Log In button 5 times	Email: user1@gmail.com Password: password	Displays “Your account has been temporarily locked for 5 mins. Please try again later	Displays “Your account has been temporarily locked for 5 mins. Please try again later or	Pass

				or reset your password Time of Login: 02:12:00 **Account will unlock at: 02:17:00**”	reset your password Time of Login: 02:12:00 **Account will unlock at: 02:17:00**”	
8	Navigate user to reset password page (1.2.1.3)	Clicking on reset your password hyperlink from Test case 6	NIL	Navigate user to reset password page	Navigate user to reset password page	Pass
9	Clicking on Forget Password? (1.2.1.3)	Click on Forget Password? hyperlink	NIL	Navigate user to reset password page	Navigate user to reset password page	Pass
10	Navigate user to sign up page (1.2.1.1)	Click on Create a ShopBlock account hyperlink	NIL	Navigate user to sign up page	Navigate user to sign up page	Pass
Test Case Status		Success				

1.2.1.3 Reset Password

Test Case ID	**T-1.2.1.3**	**Test Case Priority**	High
Test Case Description	Reset Password for a user account		
Prerequisite	1. User should have their account credentials ready.	**Postrequisite**	1. User will be redirected to login page
Test Execution			

#	**Description**	**Action**	**Input**	**Expected Output**	**Actual Output**	**Test Result**
1	Reset password with	Enter email and phone number and a new password	Email: user1@gmail.com	Snackbar appears with a message	Snackbar appears with a message	Pass

	registered account	User clicks on Reset Password button	Phone number: 98765432 Password: P@ssword123	displaying “Password reset successfully!” User is redirected to Log In page after a 2.5s delay	displaying “Password reset successfully!” User is redirected to Log In page after a 2.5s delay	
2	Submitting an empty form	User clicks on Reset Password button without any inputs	NIL	Displays “All fields are required.”	Displays “All fields are required.”	Pass
3	Attempt to reset password with registered account and input basic password	Enter email and phone number and a new password that does not meet the password requirements User clicks on Reset Password button	Email: user1@gmail.com Phone number: 98765432 Password: password	Displays “New password does not match the password requirements.”	Displays “New password does not match the password requirements.”	Pass
4	Enter unregistered email and phone number	Enter an unregistered email and phone number with a new password User clicks on Reset Password button	Email: qwerty@gmail.com Phone number: 88888888 Password: P@ssword123	Displays “Both email and phone number not found!”	Displays “Both email and phone number not found!”	Pass
5	Enter a valid email with unregistered phone number	Enter a valid email with unregistered phone number and a new password User clicks on Reset Password button	Email: user1@gmail.com Phone number: 45674567 Password: P@ssword123	Displays “Phone number not found!”	Displays “Phone number not found!”	Pass
6	Enter a unregistered email with valid phone number	Enter a unregistered email with valid phone number and a new password User clicks on Reset Password button	Email: qwerty@gmail.com Phone number: 98765432	Displays “Email not found!”	Displays “Email not found!”	Pass

			Password: P@ssword123			
Test Case Status	Success					

1.2.1.4 Change Password

Test Case ID	**T-1.2.1.4**	**Test Case Priority**	High
Test Case Description	Change password for a registered account		
Prerequisite	1. User must be logged in 2. User must have their account credentials ready	**Postrequisite**	1. User will be redirected to home page
Test Execution			

#	Description	Action	Input	Expected Output	Actual Output	Test Result
1	Change password with logged in account	Enter old password, new password and confirm password User clicks on Change Password button	Password: P@ssword1 New Password: P@ssword123 Confirm Password: P@ssword123	Displays “Your password has been changed!” User is redirected to home page after a 2.5s delay	Displays “Your password has been changed!” User is redirected to home page after a 2.5s delay	Pass
2	Submitting an empty form	User clicks on Change Password button without any inputs	NIL	Displays “All fields are required.”	Displays “All fields are required.”	Pass
3	Entering a password not tied to your account	Enter incorrect old password, new password and confirm password User clicks on Change Password button	Password: password New Password: P@ssword123 Confirm Password: P@ssword123	Displays “Password does not match!”	Displays “Password does not match!”	Pass

4	Entering a new password which does not meet password requirements	Enter old password, new password and confirm password User clicks on Change Password button	Password: P@ssword1 New Password: password Confirm Password: password	Displays "New password does not meet the password requirements. "	Displays "New password does not meet the password requirements."	Pass
5	Confirm password does not match with new password	Enter old password, new password and confirm password User clicks on Change Password button	Password: P@ssword1 New Password: P@ssword123 Confirm Password: P@ssword456	Displays "Please make sure your passwords match."	Displays "Please make sure your passwords match."	Pass
Test Case Status		Success				

1.2.1.5 Edit User Account Details

Test Case ID	**T-1.2.1.5**	**Test Case Priority**	High
Test Case Description	Allows the user to edit their account information under user profile page that can be accessed in sidebar		
Prerequisite	1. User must be logged in	**Postrequisite**	1. Updated details will be reflected on frontend and backend
Test Execution			

#	Description	Action	Input	Expected Output	Actual Output	Test Result
1	Upload an image file to be profile picture	User clicks on profile picture icon User is prompted to upload an image file of any format (jpg, jpeg, png, svg)	Image: User's selected image	Profile picture icon is updated with the newly uploaded image Snackbar appears and	Profile picture icon is updated with the newly uploaded image Snackbar appears and displays the message	Pass

		User submits the image of their choice		displays the message “Avatar updated successfully!”	“Avatar updated successfully!”	
2	Edit and Save Biography content on biography text box	User clicks on Edit button to enter editing mode Edit button is changed to Save button User enters new biography content User clicks on Save button to save changes	Old biography content: “Hey there! Nice to meet you!” New biography content: “Hey there! Nice to meet you! 12345”	Biography text box is replaced with the new content: “Hey there! Nice to meet you! 12345” Snackbar appears and displays the message “Biography saved!”	Biography text box is replaced with the new content: “Hey there! Nice to meet you! 12345” Snackbar appears and displays the message “Biography saved!”	Pass
3	Edit and Save username and phone number in About me section	User clicks on Edit button to enter editing mode Edit button is changed to Save button User enters new username and phone number User clicks on Save button to save changes	Old username: Hecarim Old phone number: 98765432 New username: Swain New phone number: 83453453	Username field is replaced with: “Swain” Phone number field is replaced with: “83453453” Snackbar appears and displays the message “Username and Phone Number saved!”	Username field is replaced with: “Swain” Phone number field is replaced with: “83453453” Snackbar appears and displays the message “Username and Phone Number saved!”	Pass
4	Edit and Save username but enter a phone number that is tied to another account in About me section	User clicks on Edit button to enter editing mode Edit button is changed to Save button	Old username: Swain Old phone number: 83453453 New username:	Snackbar appears and displays the message “Phone number has been taken!”	Snackbar appears and displays the message “Phone number has been taken!”	Pass

		User enters new username and a phone number belonging to another account User clicks on Save button to save changes	Swain New phone number: 99999999			
5	Allow user to change password	User clicks on Change Password button in About Me section	NIL	Navigate user to Change Password page	Navigate user to Change Password page	Pass
6	Allow user to delete account	User clicks on Delete Account button in About Me section Pop-up window appears to prompt user to delete account User presses “OK” button on window	NIL	Delete account for user in the backend and log user out of the account Redirect user to home page	Delete account for user in the backend and log user out of the account Redirect user to home page	Pass
7						Pass
Test Case Status		Success				

1.2.2 Browsing

Test Case ID	**T-1.2.2**	**Test Case Priority**	High
Test Case Description	To test if browsing categories show correct listings and behaviour based on category selection.		
Prerequisite	1. Listings exist in the system for various categories (Electronics, Supplies, Services), including at least one empty category.	**Postrequisite**	1. Listings display behaviour based on the selections.
Test Execution			

#	Description	Action	Input	Expected Output	Actual Output	Test Result

1	Open listing categories from NavBar.	User clicks on a category in NavBar.	Select category (e.g., Electronics).	Page displays listings filtered by category, electronics	Listing shows category electronics	Pass
2	Verify empty category behaviour.	Click on a category with no listings.	Empty category selected.	"Showing 0 products in [Category]" displayed.	"Showing 0 products in [Category]" displayed.	Pass
3	Verify populated category behaviour.	Click on a populated category.	Populated category selected.	"Showing X products in [Category]" displayed.	"Showing X products in [Category]" displayed.	Pass
4	Click on a listing to confirm category correctness.	User selects a product.	Select any listed product.	Product matches the selected category.	Product matches the selected category.	Pass
5	Navigate back to browse listings via breadcrumb.	User clicks the breadcrumb link.	Click breadcrumb.	Returns to category listings view.	Returns to category listings view.	Pass
7	Select filtering options.	User selects filter options from the sidebar.	Choose rate types (e.g., hourly, daily, weekly, one time payment).	Filters are displayed with selected options highlighted.	Filters are displayed with selected options highlighted.	Pass
8	Verify rate selection behaviour.	User checks the desired rate boxes.	Select multiple rate types (e.g., hourly + daily).	Selected boxes reflect user's choice.	Selected boxes reflect user's choice.	Pass
9	Click the filter button to apply filters.	User clicks the "filter" button.	NIL	Listings filtered by selected rates are shown.	Listings filtered by selected rates are shown.	Pass
10	Verify filtered listings.	User clicks on a filtered listing.	Select a product.	Verify listing matches the filtered criteria.	Verify listing matches the filtered criteria.	Pass
11	Open the search bar.	User clicks on the search bar.	NIL	Search bar becomes active and	Search bar becomes active and ready for input.	Pass

				ready for input.		
12	Input search letter.	User types a letter in the search bar.	“A”	Drop down menu appears showing listing titles with “A”	Drop down menu appears showing listing titles with “A”	Pass
13	Display search result.	Dropdown with hyperlink appears.	Matching title exists	Dropdown with hyperlink to the product is shown.	Dropdown with hyperlink to the product is shown.	Pass
14	Handle no match scenario.	User types any lettter in the search bar	“Anaconda” - No listing with this title	“no match found”	“no match found”	Pass
Test Case Status		Success				

1.2.3 Listing

Create Listing

Test Case ID	**T-1.2.3**	**Test Case Priority**	High
Test Case Description	Create Listing as a user who is logged in		
Prerequisite	3. User must be logged in 4. User must be at their Listing page	**Postrequisite**	
Test Execution			

#	Description	Action	Input	Expected Output	Actual Output	Test Result
1	User cancels without submitting	1. User clicks on “Create Listing” 2. User clicks on “cancel” button	Nil	Form exits with no listing created. All inputs cleared.	Form exits with no listing created. All inputs cleared.	Pass
2	Submit empty form	1. User clicks on “Create Listing” 2.User clicks on “submit” without filing in the form	Nil	Display alert: "Please fill in all fields”	Display alert: "Please fill in all fields”	Pass

3	Submit form with only some required fields filled	1. User clicks on “Create Listing” 2. User fills in 4 out of the 8 fields 3. User clicks on “submit” button	Title: Test Listing Rate: 5 OneTime Category: Supplies Listing Type: Rental	Display alert: "Please fill in all fields”	Display alert: "Please fill in all fields”	Pass
4	Submit form with all required fields	1. User User clicks on “Create Listing” 2. User fills in all the required fields 3. User clicks on “submit” button	Title: Test Listing Rate: 5 OneTime Category: Supplies Listing Type: Rental Description: Test case 3 Photo: test.jpeg Location: Jurong Point Additional Notes: Meet be at the main entrance	Form exits upon submission and new listing is automatically displayed on user’s listing page.	Form exits upon submission and new listing is automatically displayed on user’s listing page.	Pass
5	Submit form with all required field and multiple rates and locations	1. User User clicks on “Create Listing” 2. User fills in all the required fields, including adding 2 rates and 2 locations 3. User clicks on “submit” button	Title: Test Listing2 Rate: 5/ OneTime 50/ weekly Category: Supplies Listing Type: Rental Description: Test case 3 Photo: test.jpeg Location: Jurong Point	Form exits upon submission and new listing is automatically displayed on the user's listing page.	Form exits upon submission and new listing is automatically displayed on the user's listing page.	Pass

			westgate Additional Notes: Meet me at the main entrance			
6	Submit Listing with Multiple Photos					Pass
Test Case Status		Success				

1.2.4 Offer

1.2.4.1 Make Offer

Test Case ID	**T-1.2.4.1**	**Test Case Priority**	High
Test Case Description	Make Offer on a selected listing		
Prerequisite	1. User must be logged in 2. User must select a listing from the browsing page	**Postrequisite**	
Test Execution			

#	Description	Action	Input	Expected Output	Actual Output	Test Result
1	Successfully made offer to a listing with time unit (D)	1. In the listing with “D” available, , user clicks on selects Time unit (D) 2. User clicks on “View Availability” 3. User selects start date, and end date 4. User confirms by clicking “Submit Availability” 5. User inputs desired amount 6. User clicks on “Make Offer”	Start Date: 3/11/2024 End Date: 3/11/2024 Amount: 300	Displays “Offer of S$ 300/D given. Total Price: S$300”	Displays “Offer of S$ 300/D given. Total Price: S$300”	Pass
2	Successfully made offer to a listing with time unit (H)	1. In the listing with “H” available, user clicks on selects Time unit (H) 2. User clicks on “View Availability” 3. User selects start date and start time, and end date and end time 4. User confirms by clicking “Submit Availability” 5. User inputs desired amount	Start Date: 1/11/2024 Start Time: 12 End Date: 1/11/2024 End Time: 18 Amount: 10	Displays “Offer of S$ 10/H given. Total Price: S$60”	Displays “Offer of S$ 10/H given. Total Price: S$60”	Pass

		6. User clicks on “Make Offer”				
3	Successfully made offer to a listing with time unit (W)	1. In the listing with “W” available, user clicks on selects Time unit (W) 2. User clicks on “View Availability” 3. User selects start date, and end date 4. User confirms by clicking “Submit Availability” 5. User inputs desired amount 6. User clicks on “Make Offer”	Start Date: 16/11/2024 End Date: 22/11/2024 Amount: 150	Displays “Offer of S$ 150/W given. Total Price: S$150”	Displays “Offer of S$ 150/W given. Total Price: S$150”	Pass
4	Successfully made offer to a listing with time unit (OT)	1. In the listing with “OT” available, user clicks on selects Time unit (OT) 2. User clicks on “View Availability” 3. User selects start date and start time, and end date and end time 4. User confirms by clicking “Submit Availability” 5. User inputs desired amount 6. User clicks on “Make Offer”	Start Date: 10/11/2024 Start Time: 12 End Date: 11/11/2024 End Time: 12 Amount: 70	Displays “Offer of S$ 70/OT given. Total Price: S$70”	Displays “Offer of S$ 70/OT given. Total Price: S$70”	Pass
5	Start date is later than end date	1. User selects any of the Time Unit available (H/D/W/OT) 2. User clicks on “View Availability” 3. User selects start date and end date 4. User confirms by clicking “Submit Availability”	Start Date: 10/11/2024 End Date: 5/11/2024	Displays “Start date/time cannot be later than End date/time”	Displays “Start date/time cannot be later than End date/time”	Pass

6	Start time is later than end time	1. In the listing with “H” or “OT” available, user clicks on selects any of the Time unit 2. User clicks on “View Availability” 3. User selects start date and start time, and end date and end time 4. User confirms by clicking “Submit Availability”	Start Date: 1/11/2024 Start Time: 18 End Date: 1/11/2024 End Time: 12	Displays “Start date/time cannot be later than End date/time”	Displays “Start date/time cannot be later than End date/time”	Pass
7	Start date is in the past	1. User selects any of the Time Unit available (H/D/W/OT) 2. User clicks on “View Availability” 3. User selects start date and end date 4. User confirms by clicking “Submit Availability”	Start Date: 15/10/2024 End Date: 17/10/2024 *Current Date: 20/10/2024	Displays “Start date cannot be in the past”	Displays “Start date cannot be in the past”	Pass
8	User cannot make offer to his own listing	1. User selects any of the Time Unit available in his listing (H/D/W/OT) 2. User clicks on “View Availability” 3. User selects start date/time and end date/time 4. User confirms by clicking “Submit Availability” 5. User inputs desired amount 6. User clicks on “Make Offer”	Start Date: 1/11/2024 Start Time: 12 End Date: 1/11/2024 End Time: 18 Amount: 50	Displays “Error. You can not make an offer on your own listing”	Displays “Error. You can not make an offer on your own listing”	Pass
9	Offer made with time unit “OT” must be less than 24 hours	1. In the listing with “OT” available, user clicks on selects Time unit (OT) 2. User clicks on “View Availability”	Start Date: 10/11/2024 Start Time: 12 End Date:	Displays “Error. Must be less than or equals to 24 hours”	Displays “Error. Must be less than or equals to 24 hours”	Pass

		3. User selects start date and start time, and end date and end time 4. User confirms by clicking “Submit Availability”	11/11/2024 End Time: 13			
10	Offer made with time unit “W” must be in multiple of 7	1. In the listing with “W” available, user clicks on selects Time unit (W) 2. User clicks on “View Availability” 3. User selects start date, and end date 4. User confirms by clicking “Submit Availability”	Start Date: 16/11/2024 End Date: 23/11/2024	Displays “Error. Days must be in multiple of 7”	Displays “Error. Days must be in multiple of 7”	Pass
11	Missing Start Date	1. User selects any of the Time Unit available in his listing (H/D/W/OT) 2. User clicks on “View Availability” 3. User selects Start time and end date/time 4. User confirms by clicking “Submit Availability”	Start Time: 12 End Date: 1/11/2024 End Time: 18	Displays “Missing input, please input Start Date”	Displays “Missing input, please input Start Date”	Pass
12	Missing Start Time	1. In the listing with “H” or “OT” available, user clicks on selects any of the Time unit 2. User clicks on “View Availability” 3. User selects Start date and end date/time 4. User confirms by clicking “Submit Availability”	Start Date: 1/11/2024 End Date: 1/11/2024 End Time: 18	Displays “Missing input, please input Start Time”	Displays “Missing input, please input Start Time”	Pass
13	Missing End Date	1. User selects any of the Time Unit available in his listing (H/D/W/OT)	Start Date: 1/11/2024 Start Time:	Displays “Missing input, please input End Date”	Displays “Missing input, please input End Date”	Pass

		2. User clicks on “View Availability” 3. User selects Start date/time and end time 4. User confirms by clicking “Submit Availability”	12 End Time: 18			
14	Missing End Time	1. In the listing with “H” or “OT” available, user clicks on selects any of the Time unit 2. User clicks on “View Availability” 3. User selects Start date/time and end date 4. User confirms by clicking “Submit Availability”	Start Date: 1/11/2024 Start Time: 12 End Date: 1/11/2024	Displays “Missing input, please input End Time”	Displays “Missing input, please input End Time”	Pass
15	Missing Availability	1. User selects any of the Time Unit available in his listing (H/D/W/OT) 2. User inputs desired amount 3. User clicks on “Make Offer”	Amount: 50	Displays “Error: Please set availability!”	Displays “Error: Please set availability!	Pass
16	Missing Amount	1. User selects any of the Time Unit available in his listing (H/D/W/OT) 2. User clicks on “View Availability” 3. User selects Start time and end date/time 4. User confirms by clicking “Submit Availability” 5. User clicks on “Make Offer”	Start Date: 10/11/2024 Start Time: 12 End Date: 11/11/2024 End Time: 12	Displays “Please enter an amount”	Displays “Please enter an amount”	Pass
Test Case Status		Success				

1.2.4.2 Accept and Decline Offer

Test Case ID	**T-1.2.4.2**	**Test Case Priority**	High
Test Case Description	Allows user to accept or decline offer on a selected listing		
Prerequisite	1. User must be logged in. 2. User must have been offered by another user on any of the logged in user's listing.	**Postrequisite**	
Test Execution			

#	Description	Action	Input	Expected Output	Actual Output	Test Result
1	Accept Offer	1. User goes to "Offers page" 2. User selects any of the available listings under "Offers received" 3. User accepts an offer by clicking on accept button	N/A	Offer accepted and displayed from "Pending Offer" to under "Accepted Offers"	Offer accepted and displayed from "Pending Offer" to under "Accepted Offers"	Pass
2	Decline Offer	1. User goes to "Offers page" 2. User selects any of the available listings under "Offers received" 3. User declines an offer by clicking on decline button	N/A	Offer declined and displayed removed from "Pending Offers"	Offer declined and displayed removed from "Pending Offers"	Pass
3	No listings to select	1. User goes to "Offers page"	N/A	Displays "No listings available"	Displays "No listings available"	Pass
4	No offers to accept or decline	1. User goes to "Offers page" 2. User selects any of the available listings under "Offers received"	N/A	Nothing displayed under "Pending Offers"	Nothing displayed under "Pending Offers"	Pass

Test Case Status	Success

1.2.5 Payment

Test Case ID	**T-1.2.5**	**Test Case Priority**	High
Test Case Description	User makes payment for an offer that he made, and has been accepted by the user that listed the service/product.		
Prerequisite	1. User must be logged in 2. User must have made an offer to another user's listing 3. Said offer must have been accepted by the user who made the listing	**Postrequisite**	1. Transaction created after successful payment
Test Execution			

#	Description	Action	Input	Expected Output	Actual Output	Test Result
1	User makes payment successfully via Paypal	1. User goes to "Offers" page 2. User selects any of the available listings under "Offers made" 3. User chooses an offer under "Offers Pending Payment" 4. User makes payment for an offer by clicking on the "Paypal" button 5. User inputs valid credentials into the pop-up and clicks "Login" 6. User selects a payment option and makes payment by clicking "Pay"	Email: sb-yw3r232752961@personal.example.com Password: m&0IKVb= *Payment amount is $70	Displays "Transaction of S$70 successful"	Displays "Transaction of S$70 successful"	Pass
2	User enters invalid Paypal credentials	1. User goes to "Offers" page 2. User selects any of the available listings under "Offers made" 3. User chooses an offer under "Offers Pending Payment"	Email: sb-yw3r232752961@personal.example.com Password: aaaa	Paypal pop-up displays "Please check your entries and try again"	Paypal pop-up displays "Please check your entries and try again"	Pass

		4. User makes payment for an offer by clicking on the “Paypal” button 5. User inputs credentials into the pop-up and clicks “Login”				
3	User makes payment with valid paypal credentials, but has insufficient balance	1. User goes to “Offers” page 2. User selects any of the available listings under “Offers made” 3. User chooses an offer under “Offers Pending Payment” 4. User makes payment for an offer by clicking on the “Paypal” button 5. User inputs credentials into the pop-up and clicks “Login” 6. User selects a payment option and makes payment by clicking “Pay”	Email: sb-yw3r232752961@personal.example.com Password: m&0IKVb= *Bank balance is 0	User is prompted to retry payment with different payment methods.	User is prompted to retry payment with different payment methods.	Pass
4	User exits paypal popup halfway through	1. User goes to “Offers” page 2. User selects any of the available listings under “Offers made” 3. User chooses an offer under “Offers Pending Payment” 4. User makes payment for an offer by clicking on the “Paypal” button 5. User exits the pop-up page	Email: sb-yw3r232752961@personal.example.com Password: m&0IKVb=	Offer selected still remains under “Offer Pending Payment”	Offer selected still remains under “Offer Pending Payment”	Pass
Test Case Status		Success				

1.2.6 Review

Test Case ID	**T-1.2.6**	**Test Case Priority**	Medium
Test Case Description	Allow users to make review on completed transactions		
Prerequisite	1. User must be logged in 2. Transaction must have been completed	**Postrequisite**	1. Review is displayed on the profile of the user who provided the service
Test Execution			

#	Description	Action	Input	Expected Output	Actual Output	Test Result
1	Successful review	1. User goes to "Purchases" page 2. User clicks on toggle switch to select "Completed" 3. User chooses a transaction to review by clicking "Rate" button 4. User fills in description and star rating on the pop-up dialog. 5. User clicks submit.	Star rating: 5 Description: Awesome seller!	Displays "Review submitted"	Displays "Review submitted"	Pass
2	Unsuccessful due to missing star rating input	1. User goes to "Purchases" page 2. User clicks on toggle switch to select "Completed" 3. User chooses a transaction to review by clicking "Rate" button 4. User fills in the description on the pop-up dialog. 5. User clicks submit.	Description: Awesome seller!	Displays "Please input a star rating"	Displays "Please input a star rating"	Pass

3	Unsuccessful due to missing description	1. User goes to “Purchases” page 2. User clicks on toggle switch to select “Completed” 3. User chooses a transaction to review by clicking “Rate” button 4. User fills in the star rating on the pop-up dialog. 5. User clicks submit.	Star rating: 5	Displays “Please input a review!”	Displays “Please input a review!”	Pass
4	User exits halfway through review	1. User goes to “Purchases” page 2. User clicks on toggle switch to select “Completed” 3. User chooses a transaction to review by clicking “Rate” button 4. User fills in description and star rating on the pop-up dialog. 5. User clicks cancel.	Star rating: 5 Description: Awesome seller!	Dialog box is closed and review is not submitted	Dialog box is closed and review is not submitted	Pass
5	No transactions available	1. User goes to “Purchases” page 2. User clicks on toggle switch to select “Completed”	N/A	Displays: “No transactions completed found!”	Displays: “No transactions completed found!”	Pass
Test Case Status		Success				

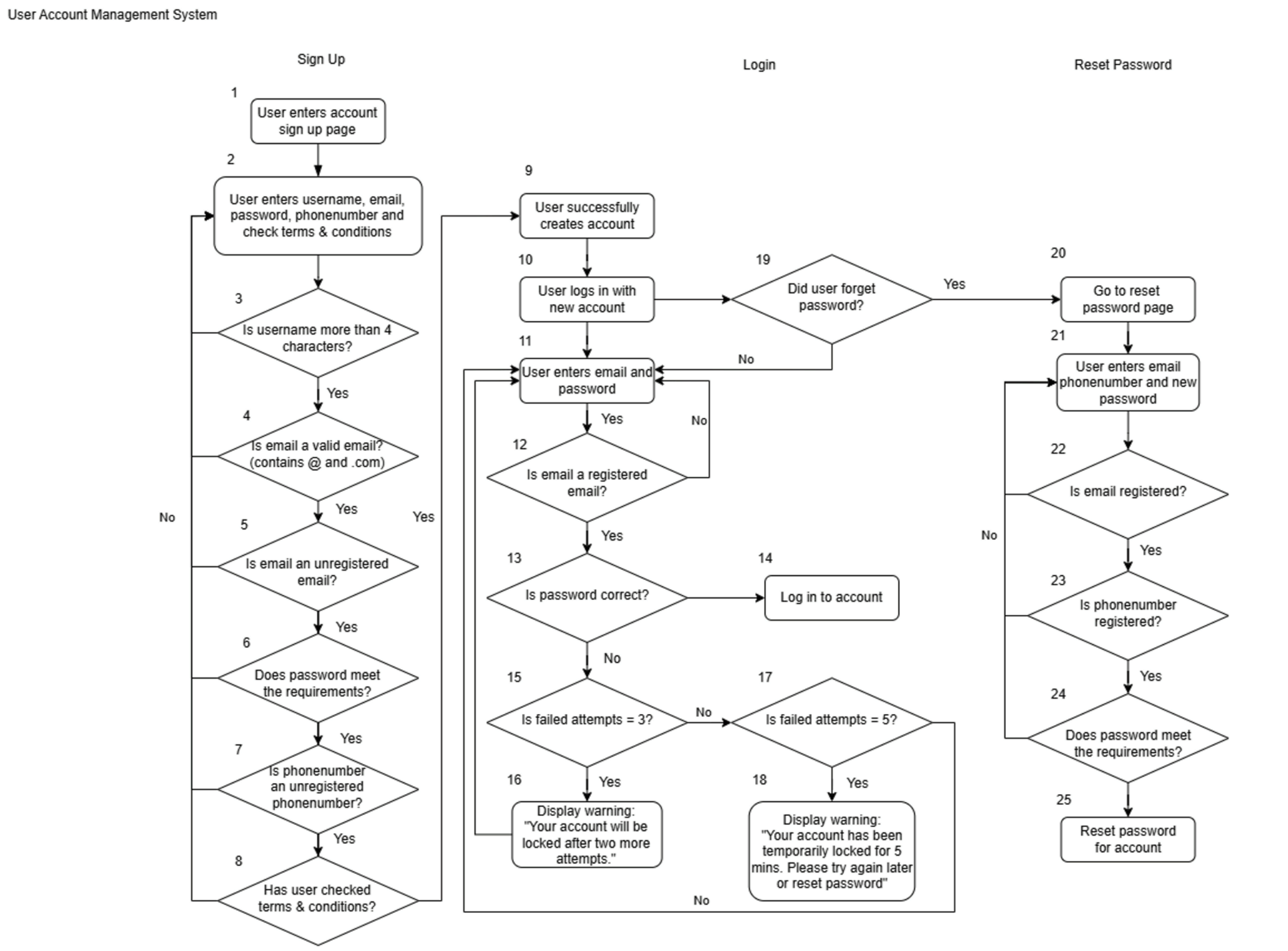
2. White Box Testing

2.1 User Account Management System

A fundamental feature for any system is comprehensive user account management, adhering to CRUD principles. In Shopblock, users will have access to essential account functionalities, including account registration, login, password reset (for forgotten passwords), password change (when logged in), and the ability to update or edit their account information (such as username, phone number, and other profile details). The system will incorporate robust validation checks to ensure entered details meet specified requirements and to prevent duplicate accounts from being created with existing information.

2.1.1 Control Flow Graph

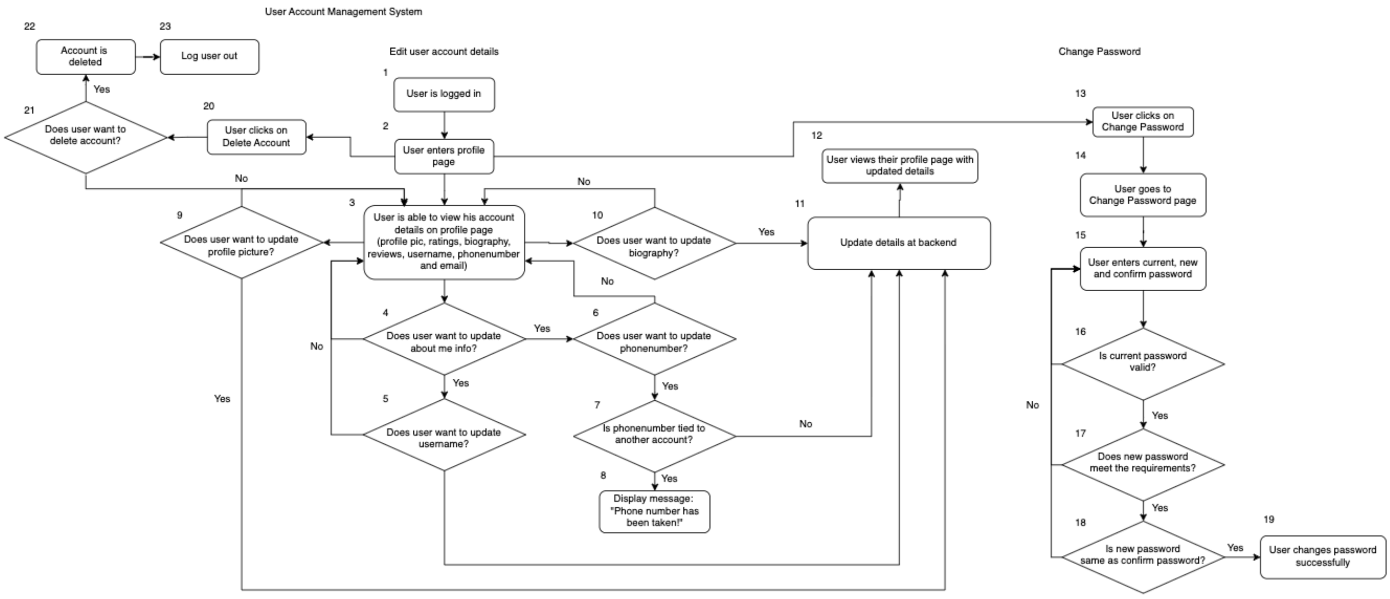
2.1.1.1 Sign Up 2.1.1.2 Login 2.1.1.3 Reset Password



User Account Management System
Sign Up
Login
Reset Password
1
User enters account sign up page
2
User enters username, email, password, phonenumber and check terms & conditions
3
Is username more than 4 characters?
Yes
4
Is email a valid email? (contains @ and .com)
Yes
No
Yes
5
Is email an unregistered email?
Yes
6
Does password meet the requirements?
Yes
7
Is phonenumber an unregistered phonenumber?
Yes
8
Has user checked terms & conditions?
9
User successfully creates account
10
User logs in with new account
19
Did user forget password?
Yes
No
11
User enters email and password
Yes
No
12
Is email a registered email?
Yes
13
Is password correct?
14
Log in to account
No
15
Is failed attempts = 3?
No
17
Is failed attempts = 5?
16
Yes
Display warning: "Your account will be locked after two more attempts."
18
Yes
Display warning: "Your account has been temporarily locked for 5 mins. Please try again later or reset password"
No
20
Go to reset password page
21
User enters email phonenumber and new password
22
Is email registered?
No
Yes
23
Is phonenumber registered?
Yes
24
Does password meet the requirements?
25
Reset password for account

2.1.1.4 Edit Account Details

2.1.1.5 Change Password


User Account Management System
Edit user account details
Change Password
1
User is logged in
2
User enters profile page
3
User is able to view his account details on profile page (profile pic, ratings, biography, reviews, username, phonenumber and email)
4
Does user want to update about me info?
5
Does user want to update username?
6
Does user want to update phonenumber?
7
Is phonenumber tied to another account?
8
Display message: "Phone number has been taken!"
9
Does user want to update profile picture?
10
Does user want to update biography?
11
Update details at backend
12
User views their profile page with updated details
13
User clicks on Change Password
14
User goes to Change Password page
15
User enters current, new and confirm password
16
Is current password valid?
17
Does new password meet the requirements?
18
Is new password same as confirm password?
19
User changes password successfully
20
User clicks on Delete Account
21
Does user want to delete account?
22
Account is deleted
23
Log user out
Yes
No

2.1.2 Basic Path Testing

Cyclomatic Complexity = | decision points | + 1

Sign Up = 7

Login = 6

Reset Password = 4

Edit User Account Details = 8

Change Password = 4

Basic Paths

Sign Up:

Baseline path 1:

1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9

Login:

Baseline path 1:

10 → 11 → 12 → 13 → 14

Reset Password:

Baseline path 1:

19 → 20 → 21 → 22 → 23 → 24 → 25

Edit User Account Details:

Baseline path 1: (Update biography)
1 → 2 → 3 → 10 → 11 → 12
Baseline path 2: (Update profile picture)
1 → 2 → 3 → 9 → 11 → 12
Baseline path 3: (Update username)
1 → 2 → 3 → 4 → 5 → 11 → 12
Baseline path 4: (Update phone number)
1 → 2 → 3 → 4 → 6 → 7 → 11→ 12
Baseline path 5: (Delete account)
1 → 2 → 20 → 21 → 22 → 23

Change Password:

Baseline path 1:
13 → 14 → 15 → 16 → 17 → 18 → 19

2.2 Browsing

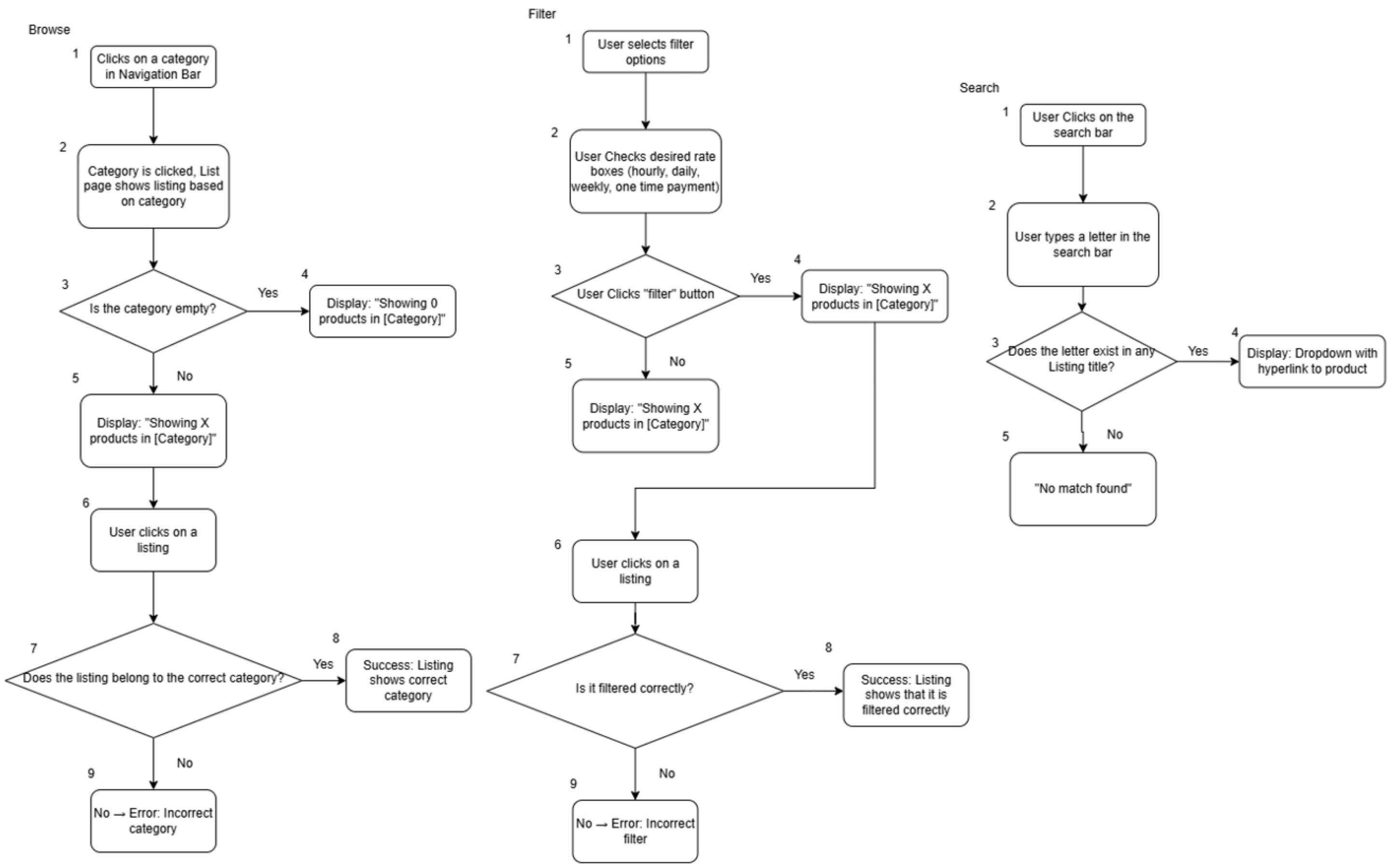
The browse feature is a key component of the ShopBlock application, serving as users' first interaction with the platform. It allows users to explore various listings created by other users. A search bar is available to help customers quickly find specific listings of interest. Additionally, the navigation bar provides category-based filtering, while a filter sidebar further refines searches based on rates and price, enhancing the browsing experience.

2.2.1 Control Flow Graph

2.2.1.1 Browse

2.2.1.2 Filter

2.2.1.3 Search


Browse
1
Clicks on a category in Navigation Bar
2
Category is clicked, List page shows listing based on category
3
Is the category empty?
Yes
4
Display: "Showing 0 products in [Category]"
No
5
Display: "Showing X products in [Category]"
6
User clicks on a listing
7
Does the listing belong to the correct category?
Yes
8
Success: Listing shows correct category
No
9
No → Error: Incorrect category
Filter
1
User selects filter options
2
User Checks desired rate boxes (hourly, daily, weekly, one time payment)
3
User Clicks "filter" button
Yes
4
Display: "Showing X products in [Category]"
No
5
Display: "Showing X products in [Category]"
6
User clicks on a listing
7
Is it filtered correctly?
Yes
8
Success: Listing shows that it is filtered correctly
No
9
No → Error: Incorrect filter
Search
1
User Clicks on the search bar
2
User types a letter in the search bar
3
Does the letter exist in any Listing title?
Yes
4
Display: Dropdown with hyperlink to product
No
5
"No match found"

2.2.2 Basic Path Testing

Cyclomatic Complexity = | decision points | + 1

Browse: 3

Filter: 3

Search: 2

Basic Paths

Browse:

Baseline Path 1:

1 → 2 → 3 → 5 → 6 → 7 → 8

Baseline Path 2:

1 → 2 → 3 → 4

Baseline Path 3:

1 → 2 → 3 → 5 → 6 → 7 → 9

Filter:

Baseline Path 1:

1 → 2 → 3 → 4 → 6 → 7 → 8

Baseline Path 2:

1 → 2 → 3 → 5 → 6 → 7 → 9

Baseline Path 3:

1 → 2 → 3 → 4 → 6 → 7 → 9

Search:

Baseline Path 1:

1 → 2 → 3 → 4

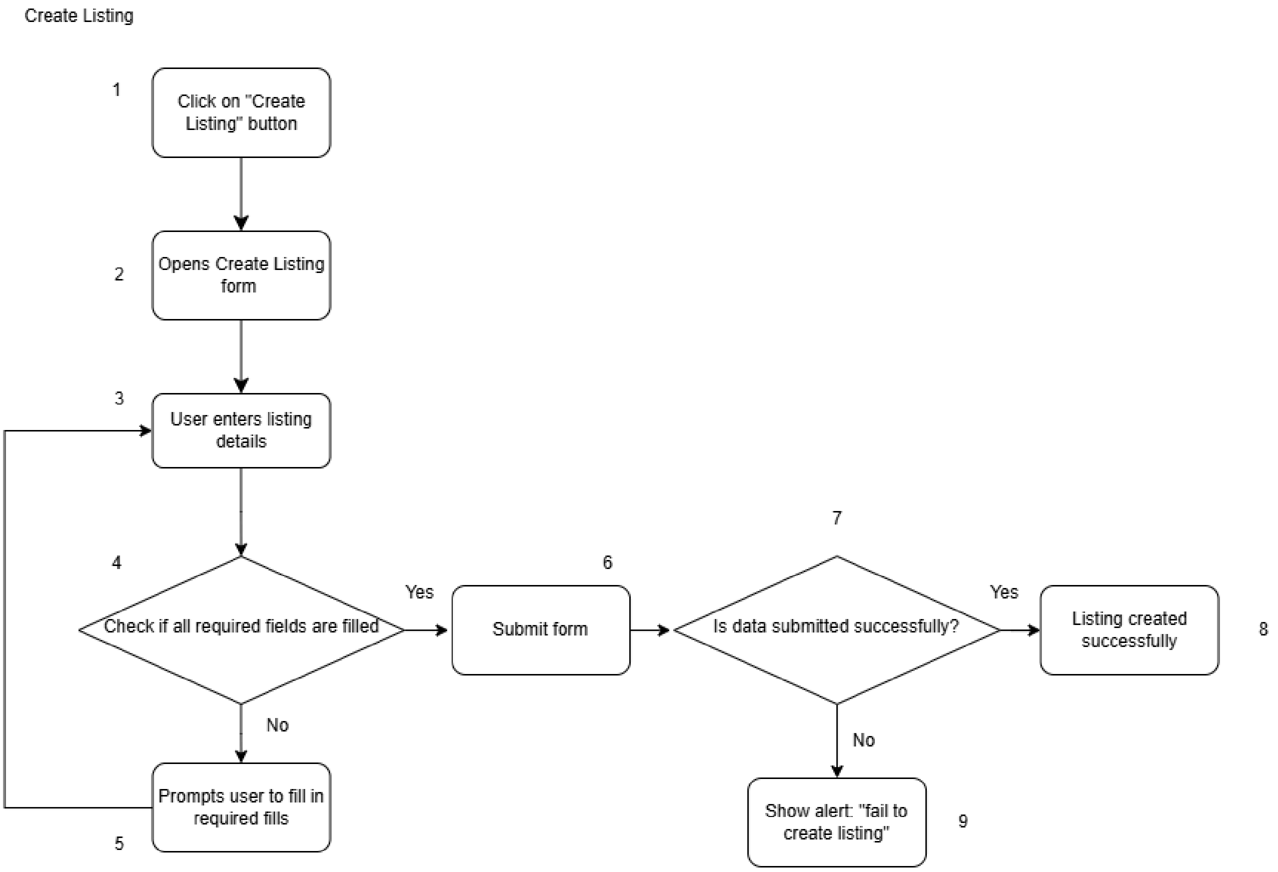
Baseline Path 2:

1 → 2 → 3 → 5

2.3 Listing

The listing feature plays a vital role in the ShopBlock application, enabling users to create and publish their own listings. When posting a listing, users are required to provide key details such as the title, price with its unit, category, listing type, location, description, and an accompanying photo.

2.3.1 Control Flow Graph


Create Listing
1
Click on "Create Listing" button
2
Opens Create Listing form
3
User enters listing details
4
Check if all required fields are filled
Yes
6
Submit form
7
Is data submitted successfully?
Yes
Listing created successfully
8
No
Prompts user to fill in required fills
5
No
Show alert: "fail to create listing"
9

2.3.2 Basic Path Testing

Cyclomatic Complexity = | decision points | + 1 = [fill me]

Cyclomatic Complexity=2+1=3

Basic Paths

Baseline path 1:

1 → 2 → 3 → 4 → 6 → 7 → 8

Baseline path 2:

1 → 2 → 3 → 4 → 5 → 3

Baseline path 3:

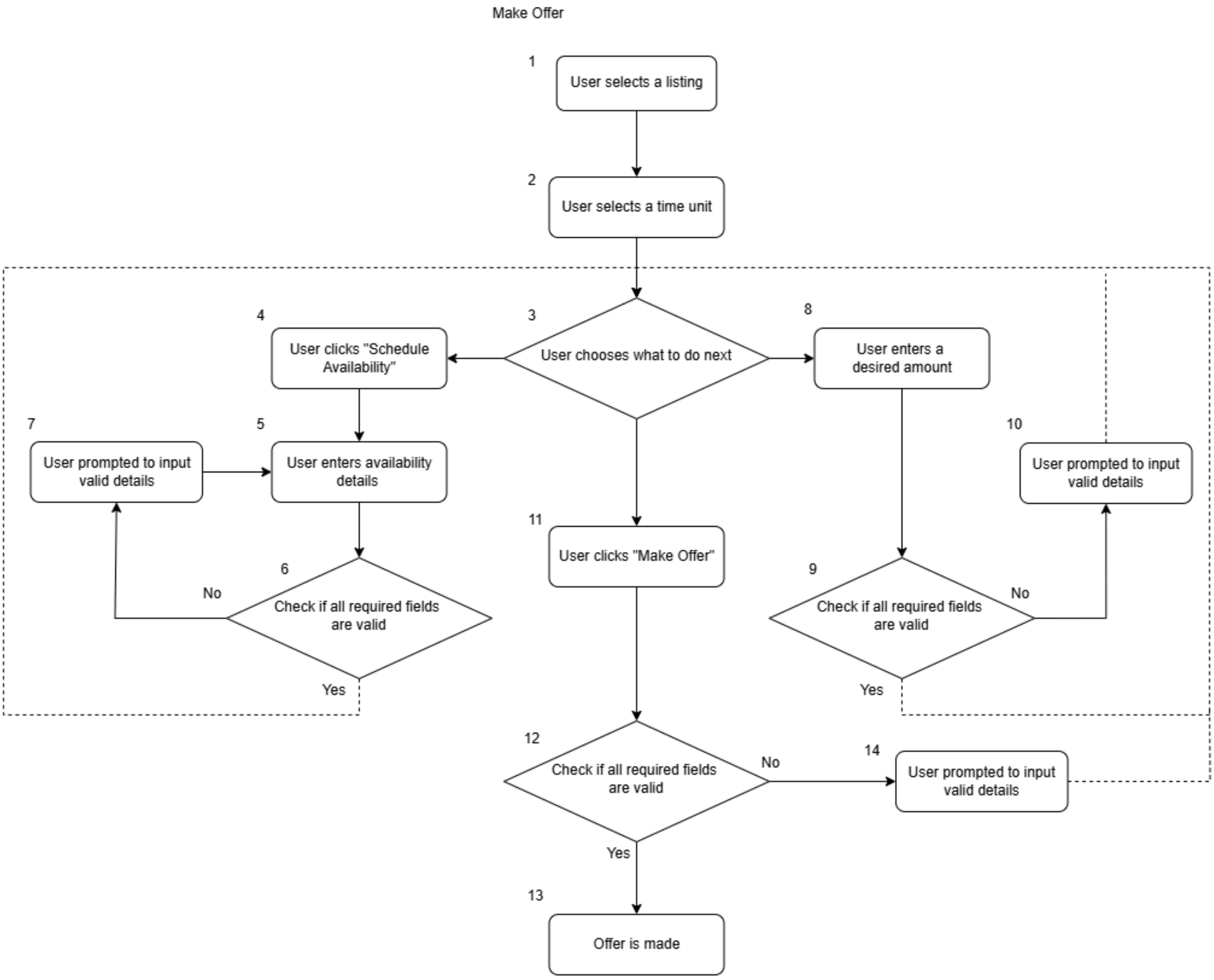
1 → 2 → 3 → 4 → 6 → 7 → 9

2.4 Offer

After browsing the listings, ShopBlock provides a "Make Offer" feature designed to help low-income families negotiate better deals. Customers can submit an offer on a listing, which will then be sent directly to the listing owner for review and consideration.

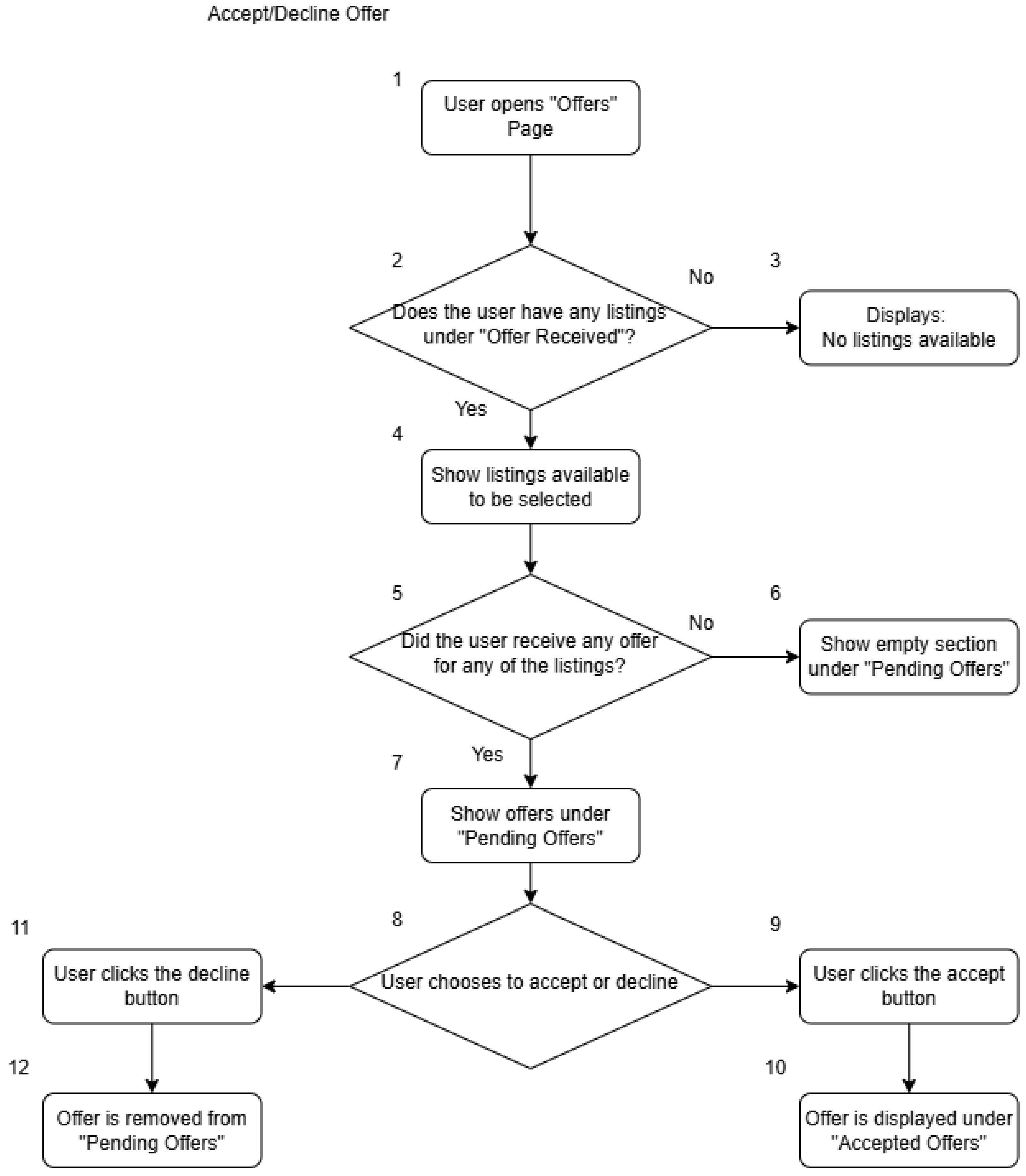
2.4.1 Control Flow Graph

2.4.1.1 Make Offer



Make Offer
1
User selects a listing
2
User selects a time unit
3
User chooses what to do next
4
User clicks "Schedule Availability"
8
User enters a desired amount
7
User prompted to input valid details
5
User enters availability details
10
User prompted to input valid details
11
User clicks "Make Offer"
6
No
Check if all required fields are valid
Yes
9
Check if all required fields are valid
No
Yes
12
Check if all required fields are valid
No
14
User prompted to input valid details
Yes
13
Offer is made

2.4.1.2 Accept/Decline Offer


Accept/Decline Offer
1
User opens "Offers" Page
2
Does the user have any listings under "Offer Received"?
No
3
Displays: No listings available
Yes
4
Show listings available to be selected
5
Did the user receive any offer for any of the listings?
No
6
Show empty section under "Pending Offers"
Yes
7
Show offers under "Pending Offers"
8
User chooses to accept or decline
11
User clicks the decline button
9
User clicks the accept button
12
Offer is removed from "Pending Offers"
10
Offer is displayed under "Accepted Offers"

2.4.2 Basic Path Testing

Cyclomatic Complexity = | decision points | + 1 = [fill me]
Cyclomatic Complexity for “Make Offer” = 4+1 = 5
Cyclomatic Complexity for “Accept/Decline Offer” = 3+1 = 4

Basic Paths

Make Offer:

Baseline path 1:
1 → 2 → 3 → 4 → 5 → 6 → 3 → 8 → 9 → 3 → 11 → 12 → 13
Baseline path 2:
1 → 2 → 3 → 4 → 5 → 6 → 5
Baseline path 3:
1 → 2 → 3 → 8 → 9 → 10 → 3
Baseline path 4:
1 → 2 → 3 → 11 → 12 → 14 → 3

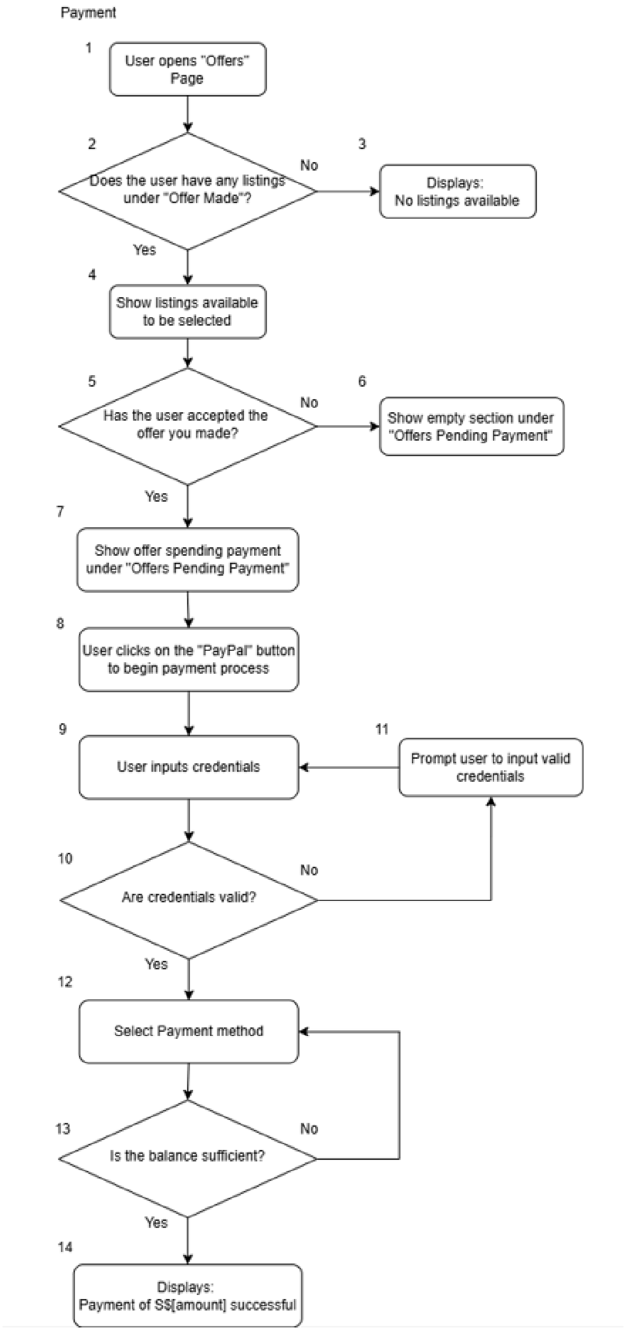
Accept/Decline Offer:

Baseline path 1:
1 → 2 → 3
Baseline path 2:
1 → 2 → 4 → 5 → 6
Baseline path 3:
1 → 2 → 4 → 5 → 7 → 8 → 9 → 10
Baseline path 4:
1 → 2 → 4 → 5 → 7 → 8 → 11 → 12

2.5 Payment

Allows a user to make payment on an accepted offer that he has made previously. This allows the offer to be confirmed and valid, making it a transaction both renter and rentee has to uphold.

2.5.1 Control Flow Graph



Payment
1
User opens "Offers" Page
2
Does the user have any listings under "Offer Made"?
No
3
Displays: No listings available
Yes
4
Show listings available to be selected
5
Has the user accepted the offer you made?
No
6
Show empty section under "Offers Pending Payment"
Yes
7
Show offer spending payment under "Offers Pending Payment"
8
User clicks on the "PayPal" button to begin payment process
9
User inputs credentials
11
Prompt user to input valid credentials
10
Are credentials valid?
No
Yes
12
Select Payment method
13
Is the balance sufficient?
No
Yes
14
Displays: Payment of S$[amount] successful

2.5.2 Basic Path Testing

Cyclomatic Complexity = | decision points | + 1 = [fill me]
Cyclomatic Complexity = 4 + 1 = 5

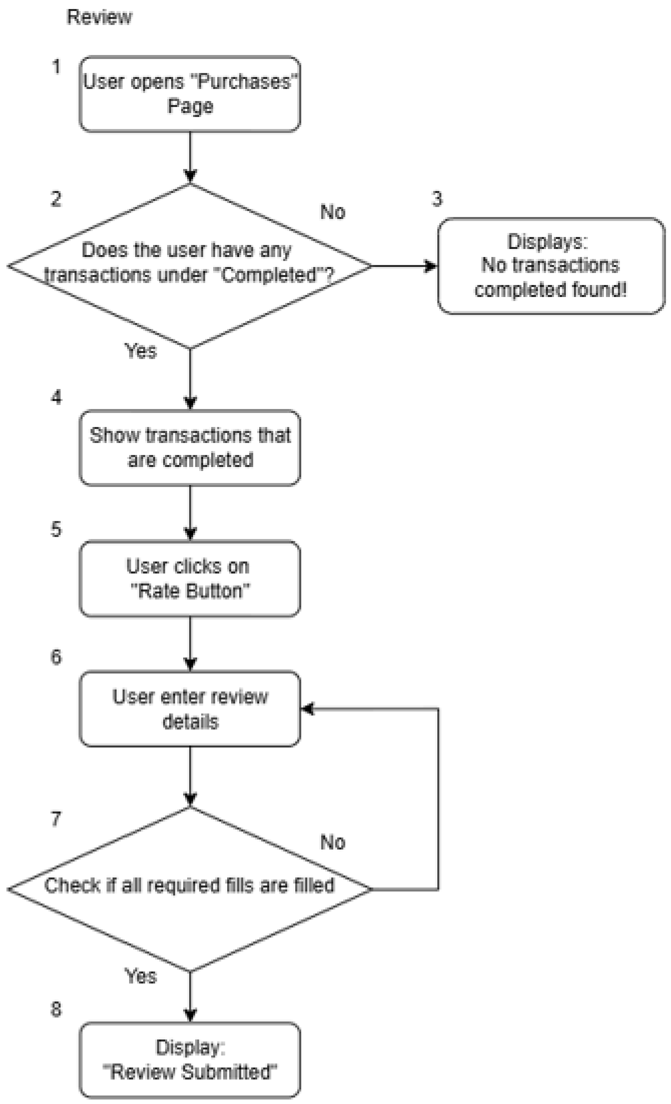
Basic Paths

Baseline path 1:
1 → 2 → 3
Baseline path 2:
1 → 2 → 4 → 5 → 6
Baseline path 3:
1 → 2 → 4 → 5 → 7 → 8 → 9 → 10 → 11 → 9
Baseline path 4:
1 → 2 → 4 → 5 → 7 → 8 → 9 → 10→ 12 → 13 → 12
Baseline path 5:
1 → 2 → 4 → 5 → 7 → 8 → 9 → 10→ 12 → 13 → 14

2.6 Review

The Review feature enables users to provide both a rating and written feedback for another user from whom they have purchased services or products.

2.6.1 Control Flow Graph


Review
1
User opens "Purchases" Page
2
Does the user have any transactions under "Completed"?
No
3
Displays: No transactions completed found!
Yes
4
Show transactions that are completed
5
User clicks on "Rate Button"
6
User enter review details
7
Check if all required fills are filled
No
Yes
8
Display: "Review Submitted"

2.6.2 Basic Path Testing

<u>Cyclomatic Complexity</u> = | decision points | + 1 = [fill me]
Cyclomatic Complexity = 2 + 1 = 3

Basic Paths

Baseline path 1:
1 → 2 → 3

Baseline path 2:
1 → 2 → 4 → 5 → 6 → 7 → 6
Baseline path 3:
1 → 2 → 4 → 5 → 6 → 7 → 8

3. API Testing

The software design of ShopBlock is heavily dependent on our RESTful backend API. Almost all functionality is done with a request to the backend in some form. This backend is crucial to the application functioning for the users as intended.

To ensure that the application functions as intended, we made use of unit testing to ensure that the APIs will take in the inputs as intended and return the outputs as intended as well. We tested all the endpoints that are exposed to the front-end.

The tests here are automated. These tests are also continuously run in our repository's continuous integration pipeline to ensure that anytime the tests fail - the developers are notified and can fix them as required.

3.1 User

We will test all functionality related to the user here.

3.1.1 User Registration

Test Case ID		**T-3.1.1**				
Endpoint		POST `/user`				
Endpoint Description		The POST endpoint will register a new user.				
Test Execution						
#	**HTTP Method, Description**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**

1	POST,
	Register a new user
		nil	{
			"username": "testuser",
			"email": "test@example.com",
			"password": "testpass123",
			"phone_number": "88888888",
			"avatar": Image,
			"biography": "Test biography",
			}
				nil	HTTP 201 (Created)	Pass

2	POST,	nil	{	nil	HTTP 400 (Bad Request)	Pass

	Register with an existing email in the system		"username": "newuser", "email": "existing@example.com", # Using existing email "password": "newpass123", "phone_number": "99999999", }			

3	POST,
	Register with incomplete data
		nil	{
			"username": "logintest",
			"email": "login@example.com",
			"password": "loginpass123",
			"phone_number": "88888888",
			"biography": "",
			}
				nil	HTTP 400 (Bad Request)	Pass

Test Case Status		Success				

3.1.2 User Login

Test Case ID		**T-3.1.2**				
Endpoint		POST `/api/login`				
Endpoint Description		The POST endpoint will take in the details of the user, and return a JWT to the user if the details are correct and return an error if not.				
Test Execution						
#	**HTTP Method, Description**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1	POST, Correct Credentials	nil	{"email": "login@example.com", "password": "loginpass123"}	nil	HTTP 200 (Ok)	Pass

2	POST, Wrong password	nil	{"email": "nonexistent@example.com" , "password": "somepass"}	nil	HTTP 401 (Unauthorized)	Pass
3	POST, Non existent user	nil	{"email": "nonexistent@example.com" , "password": "somepass"}	nil	HTTP 401 (Unauthorized)	Pass
Test Case Status		Success				

3.1.3 Get User Details

Test Case ID		**T-3.1.3**				
Endpoint		GET `/user`				
Endpoint Description		The GET endpoint will return the details of a user.				
Test Execution						
#	**HTTP Method, Description**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1	GET, Get details of the JWT user	Authorization : Bearer Token	nil	nil	HTTP 200 (Ok)	Pass
2	GET, No JWT Token	nil	nil	nil	HTTP 401 (Unauthorized)	Pass
3	GET, Get details of the user in the query param	nil	nil	?id=3	HTTP 200 (Ok)	Pass
4	GET, Get details of a non-existent user	nil	nil	?id=999	HTTP 404 (Not Found)	Pass
Test Case Status		Success				

3.1.4 Update User Details

Test Case ID	**T-3.1.4**		
Endpoint	PUT `/user`		
Endpoint Description	The PUT endpoint allows an authenticated user to update their details.		
Test Execution			

#	HTTP Method, Description	Header	Body	Params	Expected Status	Pass/Fail

1	PUT,
	Update user with new details
		Authorization: Bearer Token	{
			"username": "updateduser",
			"phone_number": "44444444",
			"avatar": new_avatar,
			"biography": "Updated bio",
			"password": "update123", # Current password
			"new_password": "newpass123",
			}
				nil	HTTP 200 (Ok)	Pass

2	PUT,
	Update user details with no authorization token
		nil	{
			"username": "updateduser",
			"phone_number": "44444444",
			"avatar": new_avatar,
			"biography": "Updated bio",
			"password": "update123", # Current password
			"new_password": "newpass123",
			}
				nil	HTTP 401 (Unauthorized)	Pass

3	PUT,
	Update password, but input the wrong current password
		Authorization : Bearer Token	{
			"password": "wrongpass",
			"new_password": "newpass123",
			}
				nil	HTTP 400 (Bad Request)	Pass

4	PUT,
	Missing fields for updating password
		Authorization : Bearer Token	{
			"username": "updateduser",
			"new_password": "newpass123", # Missing current password
			}
				nil	HTTP 400 (Bad Request)	Pass

5	PUT, Partial update of profile, only update username	Authorization : Bearer Token	{"username": "newusername"}	nil	HTTP 200 (Ok)	Pass
6	PUT, Partial update of profile, only update phone number	Authorization : Bearer Token	{"phone_number": "66666666"}	nil	HTTP 200 (Ok)	Pass
7	PUT, Partial update of profile, only update biography	Authorization : Bearer Token	{"biography": "New bio"	nil	HTTP 200 (Ok)	Pass
Test Case Status		Success				

3.1.5 Delete User

Test Case ID	**T-3.1.5**		
Endpoint	DELETE `/user`		

Endpoint Description		The DELETE endpoint allows an authenticated user to delete their account.				
Test Execution						
#	**HTTP Method, Description**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1	DELETE, Delete user details with authorization	Authorization : Bearer Token	nil	nil	HTTP 200 (Ok)	Pass
2	DELETE, Delete user details without authorization	nil	nil	nil	HTTP 401 (Unauthorized)	Pass
Test Case Status		Success				

3.2 Listing

We will test all functionality related to the listing here.

3.2.1 Get Listing Details

Test Case ID	**T-3.2.1**		
Endpoint	GET `/listing`		
Endpoint Description	The GET endpoint will return details related to the listing.		
Test Execution			

#	HTTP Method, Description	Header	Body	Params	Expected Status	Pass/Fail
1	GET, Retrieve all listings	nil	nil	nil	Returns an array of Listings HTTP 200 (Ok)	Pass
2	GET, Retrieve a single listing	nil	nil	?id=2	Returns the details of listing with id 2 HTTP 200 (Ok)	Pass
3	GET, Retrieve a non existing listing	nil	nil	?id=999	Listing not found HTTP 404 (Not Found)	Pass
4	GET, Retrieve listing with a search parameter	nil	nil	?search=Drill	Returns the listing with a Drill in the title HTTP 200 (Ok)	Pass
5	GET, Retrieve listing with a category parameter	nil	nil	?category=Services	Returns services listings only HTTP 200 (Ok)	Pass
6	GET,	nil	nil	?listing_typ	Returns rental listings only	Pass

	Retrieve listing with a listing type parameter			e=Rental	HTTP 200 (Ok)	
7	GET, Retrieve listing while sorting by price	nil	nil	?time_unit=Hourly&sort_by=price_asc	Returns listings with only the hourly time unit and sorted by price HTTP 200 (Ok)	Pass
8	GET, Retrieve listing while sorting by price, but did not provide time unti	nil	nil	?sort_by=price_asc	Time unit must be specified when sorting by price HTTP 400 (Bad Request)	Pass
Test Case Status		Success				

3.2.2 Create Listing

Test Case ID		**T-3.2.2**				
Endpoint		POST `/listing`				
Endpoint Description		The POST endpoint will allow a user to create a listing				
Test Execution						
#	**HTTP Method, Description**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1	POST, Create a listing with single location and single rates	Authorization: Bearer Token	{ "title": "New Test Listing", "description": "Test description", "category": Category.SUPPLIES, "listing_type": ListingType.RENTAL, "photos": [get_blank_photo()],	nil	HTTP 201 (Created)	Pass

			"rates": '[{"time_unit": "D", "rate": "25.00"}]', "locations": '[{"latitude": 1.35160, "longitude": 103.87119, "query": "Nex", "notes": "Test location"}]', "uploaded_by": self.user1.id, }			

2	POST,
	Create a listing with incomplete details
		Authorization : Bearer Token	{
			"title": "Incomplete Listing",
			# missing other required fields
			}
				nil	HTTP 400 (Bad Request)	Pass

3	POST,
	Create a listing with multiple location and single rates
		Authorization : Bearer Token	{
			"title": "Multi-Location Test",
			"description": "Test listing with multiple locations",
			"category": Category.SUPPLIES,
			"listing_type": ListingType.RENTAL,
			"photos": [get_blank_photo()],
			"rates": '[{"time_unit": "D", "rate": "25.00"}]',
			"locations": '[{"latitude": "1.35160", "longitude": "103.87119", "query": "Nex", "notes": "Location 1"}, {"latitude": "1.31745", "longitude": "103.80704", "query": "Farrer Road", "notes": "Location 2"}]',
			"uploaded_by": self.user1.id,
			}
				nil	HTTP 201 (Created)	Pass

4	POST, Create a listing with single location and multiple rates	Authorization : Bearer Token	{ "title": "Multi-Rate Test", "description": "Test listing with multiple rates", "category": Category.SUPPLIES,	nil	HTTP 201 (Created)	Pass

			"listing_type": ListingType.RENTAL, "photos": [get_blank_photo()], "rates": '[{"time_unit": "H", "rate": "10.00"}, {"time_unit": "D", "rate": "50.00"}, {"time_unit": "W", "rate": "250.00"}]', "locations": '[{"latitude": "1.35160", "longitude": "103.87119", "query": "Nex", "notes": "Single location"}]', "uploaded_by": self.user1.id, }			

5	POST,
	Create a listing with multiple locations and multiple rates
		Authorization : Bearer Token	{
			"title": "Multi-Location-Rate Test",
			"description": "Test listing with multiple locations and rates",
			"category": Category.SUPPLIES,
			"listing_type": ListingType.RENTAL,
			"photos": [get_blank_photo()],
			"rates": '[{"time_unit": "H", "rate": "15.00"}, {"time_unit": "D", "rate": "75.00"}, {"time_unit": "W", "rate": "350.00"}]',
			"locations": '[{"latitude": "1.35160", "longitude": "103.87119", "query": "Nex", "notes": "Location 1"}, {"latitude": "1.31745", "longitude": "103.80704", "query": "Farrer Road", "notes": "Location 2"}, {"latitude": "1.42953", "longitude": "103.83503", "query": "Yishun", "notes": "Location 3"}]',
			"uploaded_by": self.user1.id,
			}
				nil	HTTP 201 (Created)	Pass

Test Case Status		Success				

3.2.3 Create Listing

Test Case ID	**T-3.2.3**		
Endpoint	PUT `/listing`		
Endpoint Description	The PUT endpoint will allow the user who created the listing to update the listing details.		
Test Execution			

#	HTTP Method, Description	Header	Body	Params	Expected Status	Pass/Fail

1	PUT,
	Update the listing details with valid data
		Authorization: Bearer Token	{
			"id": self.listing1.id,
			"title": "Updated Drill",
			"description": "Updated description",
			"category": Category.ELECTRONICS,
			"listing_type": ListingType.RENTAL,
			"rates": '[{"time_unit": "OT", "rate": "15.00"}]',
			"locations": '[{"latitude": 1.35160, "longitude": 103.87119, "query": "Nex", "notes": "Test location"}]',
			"uploaded_by": self.user1.id,
			}
				nil	HTTP 200 (Ok)	Pass

2	PUT, Trying to update another user's listing	Authorization: Bearer Token	{ "id": self.listing1.id, "title": "Updated Drill", "description": "Updated description", "category": Category.ELECTRONICS, "listing_type": ListingType.RENTAL, "rates": '[{"time_unit": "OT", "rate": "15.00"}]', "locations": '[{"latitude": 1.35160,	nil	HTTP 403 (Forbidden)	Pass

			"longitude": 103.87119, "query": "Nex", "notes": "Test location"}]', "uploaded_by": self.user1.id, }			
Test Case Status		Success				

3.2.4 Delete Listing

Test Case ID		**T-3.2.4**				
Endpoint		DELETE `/listing`				
Endpoint Description		The DELETE endpoint will allow the user who created the listing to remove the listing.				
Test Execution						
#	**HTTP Method, Description**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1	DELETE, The user deletes their own listing	Authorization : Bearer Token	nil	id=1	HTTP 200 (Ok)	Pass
2	DELETE, Trying to delete another user's listing	Authorization : Bearer Token	nil	id=2	You don't have permission to delete this listing HTTP 403 (Forbidden)	Pass
2	DELETE, Trying to delete a non-existent listing	Authorization : Bearer Token	nil	id=999	Listing not found HTTP 404 (Not Found)	Pass
Test Case Status		Success				

3.3 Offer

We will test all functionality related to the offer here.

3.3.1 Get Offers Details

Test Case ID		**T-3.3.1**				
Endpoint		GET `/offers`				
Endpoint Description		The GET endpoint will return offer details				
Test Execution						
#	**HTTP Method, Description**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1	GET, Retrieve all offers related to the listing	Authorization: Bearer Token	nil	?listing_id=1	Returns an array of Offers related to the Listing id provided HTTP 200 (Ok)	Pass
2	GET, Retrieve all offers by received	Authorization: Bearer Token	nil	?type=received	Returns an array of offers that the owner of the JWT token has received HTTP 200 (Ok)	Pass
3	GET, Retrieve all offers by made	Authorization: Bearer Token	nil	?type=madce	Returns an array of offers that the owner of the JWT token has made HTTP 200 (Ok)	Pass
Test Case Status		Success				

3.3.2 Create Offer

Test Case ID	**T-3.3.2**		
Endpoint	POST `/offers`		
Endpoint Description	The POST endpoint will allow an authenticated user to create an offer for a listing.		
Test Execution			

#	HTTP Method, Description	Header	Body	Params	Expected Status	Pass/Fail

1	POST,
	Create a new offer
		Authorization: Bearer Token	{
			"listing_id": self.listing.id,
			"price": 15.00,
			"scheduled_start": (timezone.now() + timedelta(days=2)).isoformat(),
			"scheduled_end": (timezone.now() + timedelta(days=2, hours=2)).isoformat(),
			"time_unit": TimeUnit.HOURLY,
			"time_delta": 2,
			}
				nil	HTTP 201 (Created)	Pass

2	POST,
	Create a new offer, but the scheduled time has a collision with an existing offer for the same listing
		Authorization: Bearer Token	{
			"listing_id": self.listing.id,
			"price": 20.00,
			"scheduled_start": self.offer1.scheduled_start.isoformat(),
			"scheduled_end": self.offer1.scheduled_end.isoformat(),
			"time_unit": TimeUnit.HOURLY,
			"time_delta": 2,
			}
				nil	HTTP 400 (Bad Request)	Pass

3	POST,
	Create a new offer, but the scheduled end time is before the start time
		Authorization: Bearer Token	{
			"listing_id": self.listing.id,
			"price": 20.00,
			"scheduled_start": timezone.now().isoformat(),
			"scheduled_end": (timezone.now() - timedelta(hours=1)).isoformat(),
			"time_unit": TimeUnit.HOURLY,
			"time_delta": 1,
			}
				nil	HTTP 400 (Bad Request)	Pass

4	POST,
	Create a new offer, but the listing does not exist
		Authorization : Bearer Token	{
			"listing_id":999,
			"price": 20.00,
			"scheduled_start": timezone.now().isoformat(),
			"scheduled_end": (timezone.now() - timedelta(hours=1)).isoformat(),
			"time_unit": TimeUnit.HOURLY,
			"time_delta": 1,
			}
				nil	HTTP 400 (Bad Request)	Pass

Test Case Status		Success				

3.3.3 Update Offer

Test Case ID		T-3.3.3				
Endpoint		PUT `/offers`				
Endpoint Description		The PUT endpoint will allow the user to update their offers received. Practically this refers to accepting and rejecting offers.				
Test Execution						
#	**HTTP Method, Description**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1	PUT, Accept offer	Authorization : Bearer Token	{"offer_id": self.offer1.id, "action": "accept"}	nil	HTTP 200 (Ok)	Pass
2	PUT, Reject offer	Authorization : Bearer Token	{"offer_id": self.offer2.id, "action": "reject"}	nil	HTTP 200 (Ok)	Pass
3	PUT, Unauthorized user trying to accept offer	nil	{"offer_id": self.offer2.id, "action": "accept"}	nil	HTTP 404 (Not FOUND)	Pass
Test Case Status		Success				

3.4 Transaction

We will test all functionality related to the transactions here.

3.4.1 Get Transaction Details

Test Case ID		**T-3.4.1**				
Endpoint		GET `/transactions`				
Endpoint Description		The GET endpoint allows the user to see their past transactions.				
Test Execution						
#	**HTTP Method, Description**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1	GET, Get all transactions related to the user	Authorization : Bearer Token	nil	nil	Returns an array of transactions related to the owner of the JWT token HTTP 200 (Ok)	Pass
2	GET, Unauthorized user	nil	nil	nil	HTTP 401 (Unauthorized)	Pass
Test Case Status		Success				

3.4.2 Create Transaction

Test Case ID		**T-3.4.2**				
Endpoint		POST `/transactions`				
Endpoint Description		The POST transaction request is used when the user finishes the payment flow. Then this endpoint will be used to log a record of the transaction.				
Test Execution						
#	**HTTP Method, Description**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**

1	POST,
	Create a new transaction
		Authorization : Bearer Token	{
			"offer_id": new_offer.id,
			"amount": 45.00, # 3 hours at $15/hour
			"status": Transaction.COMPLETED,
			"payment_id": "TEST_PAYMENT_2",
			}
				nil	HTTP 201 (Created)	Pass

2	POST,
	Create a new transaction, but with missing fields
		Authorization : Bearer Token	{
			"offer_id": new_offer.id,
			# missing amount
			"status": Transaction.COMPLETED,
			}
				nil	HTTP 400 (Bad Request)	Pass

3	POST,
	Create a new transaction with pending status
		Authorization : Bearer Token	{
			"offer_id": new_offer.id,
			"amount": 30.00,
			"status": Transaction.PENDING,
			"payment_id": "PENDING_PAYMENT",
			}
				nil	HTTP 201 (Created)	Pass

4	POST,
	Create a new transaction with failed status
		Authorization : Bearer Token	{
			"offer_id": new_offer.id,
			"amount": 30.00,
			"status": Transaction.FAILED,
			"payment_id": "FAILED_PAYMENT",
			}
				nil	HTTP 201 (Created)	Pass

5	POST,
	Create a new transaction with invalid status
		Authorization : Bearer Token	{
			"offer_id": new_offer.id,
			"amount": 30.00,
			"status":"X",
			"payment_id": "INVALID_STATUS_PAYMENT",
			}
				nil	HTTP 400 (Bad Request)	Pass

6	POST,	nil	{ "offer_id": self.offer.id,	nil	HTTP 401 (Unauthorized)	Pass

	Unauthorized user creating a transaction		"amount": 20.00, "status": Transaction.COMPLETED, "payment_id": "TEST_PAYMENT", }			
Test Case Status		Success				

3.5 Review

3.5.1 Get Review Details

Test Case ID		**T-3.5.1**				
Endpoint		GET `/reviews`				
Endpoint Description		The GET endpoint gets all reviews for the given user				
Test Execution						
#	**HTTP Method, Description**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1	GET, Gets all reviews for the user	Authorization : Bearer Token	nil	?user_id=1	Returns an array of reviews for user1 HTTP 200 (Ok)	Pass
2	GET, Gets all reviews for a user with no reviews	Authorization : Bearer Token	nil	?user_id=3	Empty array HTTP 200 (Ok)	Pass
3	GET, Get reviews for user of the token	Authorization : Bearer Token	nil	nil	Returns an array of reviews for the owner of the JWT token HTTP 200 (Ok)	Pass
4	GET,	nil	nil	?user_id=1	Returns an array of reviews for user1	Pass

	Get reviews of a specific user without a token. Users can view reviews of other users without a token				HTTP 200 (Ok)	
5	GET, Get reviews of a user without a specified parameter and without a token	nil	nil	nil	HTTP 401 (Unauthorized)	Pass
6	GET, Get the reviews of the user and ensure that the average rating is calculated correctly	nil	nil	?user_id=1	Asserts that the average rating is the same as the one calculated. HTTP 200 (Ok)	Pass
Test Case Status		Success				

3.5.2 Create Review

Test Case ID		**T-3.5.2**				
Endpoint		POST `/reviews`				
Endpoint Description		The POST endpoint allows a user to create a review for another user				
Test Execution						
#	**HTTP Method, Description**	**Header**	**Body**	**Params**	**Expected Status**	**Pass/Fail**
1	POST,	Authorization: Bearer Token	{	nil	HTTP 201 (Created)	Pass

	Create a review for another user		"user_id": self.user3.id, # Reviewing user3 "rating": 4, "description": "Good communication and reliable", }			

2	POST,
	
	Create a review for yourself user
		Authorization: Bearer Token	{
			"user_id": self.user2.id, # Reviewing user3
			"rating": 4,
			"description": "Good communication and reliable",
			}
				nil	Cannot review yourself
					
					HTTP 400 (Bad Request)
						Pass

3	POST,
	
	Create a review with invalid ratings, as ratings only go from 0.5 to 5
		Authorization: Bearer Token	{
			"user_id": self.user3.id, # Reviewing user3
			"rating": 6,
			"description": "Good communication and reliable",
			}
				nil	HTTP 400 (Bad Request)	Pass

3	POST,
	
	Create a review with missing fields
		Authorization: Bearer Token	{
			"user_id": self.user3.id,
			# missing rating
			"description": "Incomplete review",
			}
				nil	HTTP 400 (Bad Request)	Pass

Test Case Status		Success				