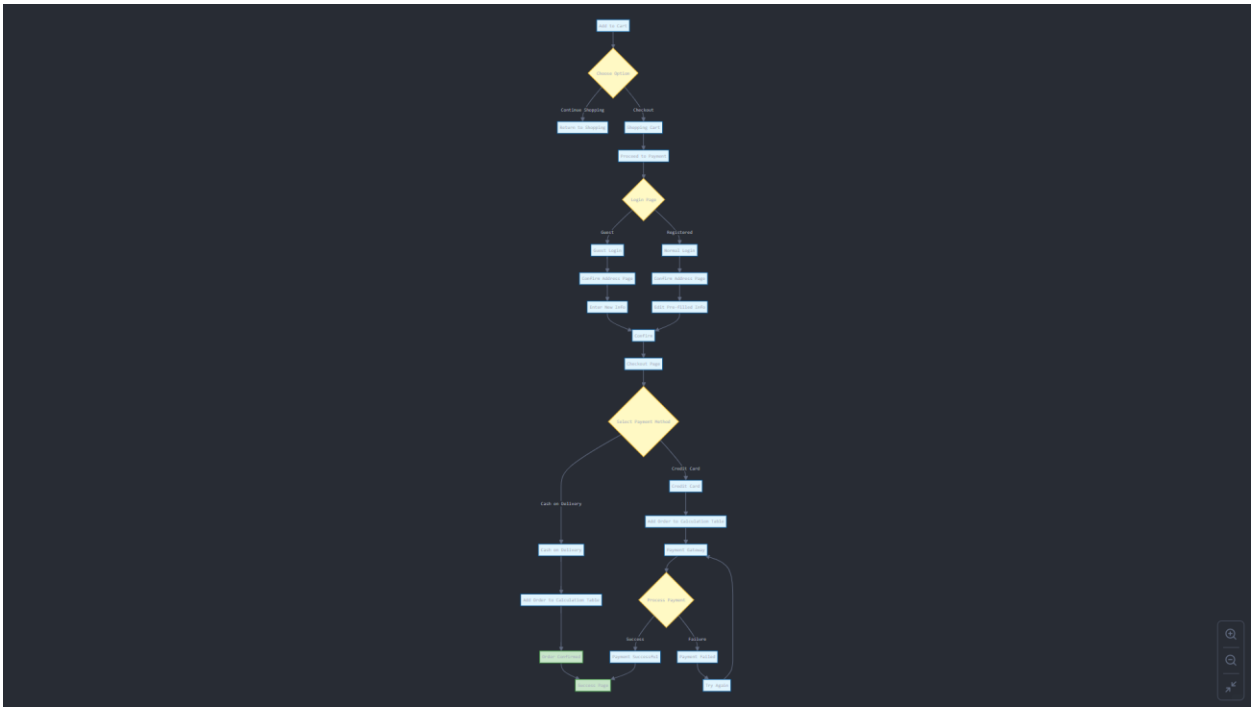
Payment Flow And Test Cases

1- Payment Flow Diagram



2- Flow Illustration

Step 1: Adding to Cart

It all starts when a customer sees something they like and clicks that "Add to Cart" button. Boom! The item's in their cart.

Step 2: The Popup Decision

Right after adding to cart, we show a little popup. It's like a fork in the road:

- Option 1: "Continue Shopping" - For those who want to browse more.
- Option 2: "Checkout" - For those ready to buy.

Step 3: Proceeding to Checkout

Let's say they choose "Checkout". They'll see their shopping cart with all the items they've added. There's a big "Proceed to Payment" button. When they click it, that's when things get interesting.

Step 4: The Login Page

Now we need to know who we're dealing with. The customer sees two options:

1. Guest Login: For those who don't want to create an account.
2. Normal Login: For our registered users.

Step 5: Confirming the Address

Here's where it gets a bit different depending on their login choice, but the end result is the same:

- Guest Login: They'll need to enter all their info: email, name, phone, address. It's like filling out a quick form.
- Normal Login: After they log in, we'll show them their saved info. But here's the cool part - they can still change their email, phone, or address for this specific order. Maybe they're sending a gift to a friend!

Step 6: The Checkout Page

After confirming the address, everyone ends up at the same checkout page. This is where they choose how they want to pay.

Step 7: Choosing the Payment Method

We offer two options:

1. Cash on Delivery: Pay when you get the package. Simple!
2. Credit Card: Pay now with your card.

Step 8: Behind the Scenes

No matter which payment method they choose, we add the order to our calculation table. It's marked as "To Be Confirmed" at this point.

Step 9: The Final Steps

Here's where the paths split again:

For Cash on Delivery:

1. The order is marked as not paid (obviously, since they'll pay later).
2. We show them a success page. Order complete!
3. The invoice is added and sent to the user and the system administrator

For Credit Card:

1. We send them to a secure payment page to enter their card details.
2. If the payment goes through smoothly, they see the success page. Woohoo! And the invoice will be sent to user and administrator
3. If there's an error (like insufficient funds), we let them know and give them a chance to try again.

3-Test Cases

1. Redirect Views and Model Action Binding

Test Case: Success Page Redirect

Objective: Ensure that the success page is correctly displayed and bound to the same model action return.

Steps:

1. Complete a successful payment transaction
2. Verify that the user is redirected to the success page
3. Check that the success page displays the correct order information
4. Confirm that the model data matches the action return from the payment process
5. The page contains options to go home or contact for company

Expected Result: The success page should load with accurate order details, reflecting the data returned by the payment action.

Test Case: Error Page, failure Page , Pending Page Redirect

Objective: Verify that the error page is properly displayed and contains relevant error information.

Steps:

1. Trigger a payment error (e.g., insufficient funds, invalid card)
2. Confirm that the user is redirected to the error page

3. Verify that the error page displays appropriate error messages
4. Check that the error details match the action return from the payment process

Expected Result: The error page should load with clear, relevant error messages that correspond to the specific issue encountered during the payment process.

2. Invoice Generation and Browser Closure Handling

Test Case: Invoice Generation on Success Page Load

Objective: Ensure that the invoice is generated and sent when the user reaches the success page.

Steps:

1. Complete a successful payment transaction
2. Allow the redirect to the success page to complete
3. Verify that the invoice is generated and sent to the user's email

Expected Result: The invoice should be generated and sent as soon as the success page starts loading, regardless of whether the page finishes loading.=> this means even if the user closed browser before redirection

Test Case: Invoice Generation on Premature Browser Closure

Objective: Confirm that the invoice is still generated and sent even if the user closes the browser before the success page fully loads.

Steps:

1. Initiate and complete a successful payment transaction
2. Close the browser immediately after the payment is processed but before the success page fully loads
3. Check the system logs and email service to verify that the invoice was generated and sent

Expected Result: The invoice should be generated and sent even if the user closes their browser before being redirected to or fully loading the success page.

Test Case: Multiple Browser Tabs

Objective: Ensure the system handles multiple open payment sessions correctly.

Steps:

1. Open the payment page in multiple browser tabs
2. Complete payment in one tab

3. Attempt to complete payment in another tab

Expected Result: The system should recognize that payment has been made and prevent duplicate payments in other tabs.

Test Case: Session Timeout During Payment

Objective: Verify that the system handles session timeouts gracefully during the payment process.

Steps:

1. Begin the payment process
2. Allow the session to timeout (this may require adjusting timeout settings for testing)
3. Attempt to complete the payment

Expected Result: The system should detect the timeout, securely end the session, and guide the user to re-authenticate without losing their cart or progress.

Test Case: Network Interruption

Objective: Ensure the system can recover from temporary network issues during payment.

Steps:

1. Initiate a payment transaction
2. Simulate a network interruption (e.g., disconnect the internet briefly)
3. Restore the network connection
4. Observe system behavior

Expected Result: The system should detect the network issue, attempt to reconnect, and either resume the transaction or provide clear instructions to the user on how to verify their payment status.

Test Case: Payment Retry Functionality

Objective: Ensure users can retry failed payments without losing their order information.

Steps:

1. Attempt a payment that fails (e.g., use a test card number that triggers a decline)
2. Verify that an error message is displayed on the same page
3. Use the provided option to retry the payment
4. Enter correct payment information
5. Complete the payment successfully

Expected Result: The user should be able to retry their payment immediately, with all their order information preserved. Upon successful payment, they should be directed to the success page.

Test Case: Page Error Handling

Objective: Verify that payment unexpected errors are displayed on the page without redirecting

Steps:

1. Trigger a payment error like assigning a var typed variable to null
2. Confirm that the user redirected to this error page and no shown for .net error
3. Verify that an error message is displayed clearly on the page
4. Check that the error details match the action return from the payment process
5. Ensure that the option to go home page and contact us are available

Expected Result: The payment page should display a clear error message providing an obvious way for the user to know something wrong happened and try again.

Notes => we have a defined structure when integrate with payment gateways we should flow the same way review the ngnuis , QI payment integration for standard case “not include webhooks ” and PayPal and tabby for advanced ones that contain “webhooks – prescoring step ”