**Automating PayPal Payment Gateway in Cypress UI Automation**

Automating PayPal payments in **Cypress UI tests** is tricky because:

1. PayPal uses **iframed checkout pages** (cross-origin).
2. Real payments require **sandbox accounts**.
3. Cypress does **not handle multiple domains well** (PayPal redirects to a new page).

**✅ Approach**

* **Option 1:** **Mock the PayPal API calls** (Best for automation).
* **Option 2:** **Use UI automation** with Cypress commands.
* **Option 3:** **Hybrid Approach** – Mock API + UI actions.

**🔹 Option 1: Mocking PayPal API (Recommended)**

Since real PayPal transactions can’t be fully automated, Cypress can **mock API responses** to simulate payment success/failure.

describe('PayPal Payment Automation', () => {

beforeEach(() => {

cy.visit('https://your-ecommerce.com/checkout'); // Replace with your checkout page

});

it('should mock a successful PayPal payment', () => {

// Intercept PayPal's API call and return a successful response

cy.intercept('POST', '/paypal/payment/process', {

statusCode: 200,

body: { success: true, transactionId: 'PAYPAL-98765' }

}).as('mockPayPalSuccess');

// Click on PayPal payment option

cy.get('#paypal-button').click();

// Wait for the API response

cy.wait('@mockPayPalSuccess').its('response.statusCode').should('eq', 200);

// Verify payment success message

cy.get('.payment-status').should('contain', 'Payment Successful');

// Confirm transaction ID is displayed

cy.get('.transaction-id').should('contain', 'PAYPAL-98765');

});

});

✅ **Key Takeaways:**

* Intercepts the API request made to PayPal.
* Mocks a successful payment response.
* Verifies that the UI updates correctly.

**🔹 Option 2: Automating PayPal UI (Using Sandbox Account)**

If you want Cypress to **actually interact** with the PayPal UI:

1. **Use PayPal Sandbox Account** (Create one at https://developer.paypal.com).
2. **Handle PayPal Iframe** and complete payment.

describe('PayPal Payment - UI Automation', () => {

it('should complete a PayPal payment via UI', () => {

cy.visit('https://your-ecommerce.com/checkout');

// Click the PayPal button (inside an iframe)

cy.get('iframe')

.its('0.contentDocument.body')

.should('not.be.empty')

.then(cy.wrap)

.find('#paypal-button')

.click();

// Wait for the PayPal popup

cy.origin('https://www.sandbox.paypal.com', () => {

cy.get('input#email').type('sandbox-buyer@example.com');

cy.get('button[type="submit"]').click();

cy.get('input#password').type('sandbox-password');

cy.get('button[type="submit"]').click();

cy.get('button#confirm-button').click();

});

// Verify success message

cy.get('.payment-status').should('contain', 'Payment Successful');

});

});

✅ **Key Takeaways:**

* Cypress handles **iframe interaction** to click the PayPal button.
* Uses cy.origin() to **switch to PayPal’s domain**.
* Completes the sandbox login & confirms the payment.

**🔹 Option 3: Hybrid Approach**

You can **mock PayPal’s API response** but still **test the UI interaction**:

1. Click the **PayPal checkout button**.
2. **Mock PayPal API response** instead of completing payment.

cy.get('#paypal-button').click();

cy.intercept('POST', '/paypal/payment/process', {

statusCode: 200,

body: { success: true, transactionId: 'PAYPAL-12345' }

}).as('mockPayPal');

cy.wait('@mockPayPal');

**📌 Final Recommendations**

| **Approach** | **Pros** | **Cons** |
| --- | --- | --- |
| **Mock PayPal API** | Fast, reliable, no sandbox needed | Doesn't test actual UI flow |
| **Full UI Automation** | Tests real interaction | Requires sandbox, slow, iframe issues |
| **Hybrid (UI + Mock API)** | Balance of both | Still involves sandbox setup |