**Automating Google Pay (GPay) Payment in Cypress**

**⚡ Challenges with Automating GPay UI**

1. **Google Pay opens in a separate window/tab**, which Cypress does not support directly.
2. **Requires real payment credentials** (Sandbox mode needed for automation).
3. **Uses iframes and browser-based authentication**.
4. **Might require mocking API responses** for a smooth test execution.

**✅ Possible Approaches**

**Approach 1: Mocking API Responses (Recommended)**

Instead of automating the actual GPay UI (which is difficult due to popups), we **mock** the response that GPay sends to your application.

cy.intercept('POST', '/gpay/payment/confirm', {

statusCode: 200,

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

}).as('gpayMock');

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

cy.get('#gpay-button').click(); // Click GPay button

cy.wait('@gpayMock'); // Wait for the mocked request

cy.get('.order-confirmation').should('contain', 'Payment Successful');

✅ **Fast & Reliable**  
✅ **Does not depend on actual Google Pay**  
✅ **Works in CI/CD pipelines**

**Approach 2: Automating Google Pay UI**

Since GPay opens in an **iframe**, you must handle it like this:

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

cy.get('#gpay-button').click(); // Click GPay button

// Switch to the Google Pay iframe

cy.get('iframe').then($iframe => {

const doc = $iframe.contents();

cy.wrap(doc).find('#email').type('testuser@gmail.com');

cy.wrap(doc).find('#password').type('testpassword');

cy.wrap(doc).find('#submit-payment').click();

});

// Validate payment success

cy.url().should('include', '/order-success');

cy.get('.order-confirmation').should('contain', 'Payment Successful');

🚨 **Issues:**  
❌ May break due to security policies (Google blocks Cypress from controlling GPay).  
❌ Hard to execute on real payment gateways.

**🎯 Best Solution**

* ✅ **For E2E Tests:** **Mock API Responses** (cy.intercept)
* ✅ **For UI Interaction:** **Handle iFrames properly (if allowed)**

To set up a **mock API server for Google Pay (GPay) testing** in your Cypress automation, follow these steps:

**🔹 Step 1: Understand How GPay Works**

When a user clicks the **Google Pay button**, the system makes a request to the backend to process the payment. We will **mock this API request** instead of using real payments.

In a real system:

* A POST request is sent to your backend (e.g., /gpay/payment/confirm).
* The backend interacts with Google Pay servers and returns a response.
* If successful, the UI shows "Payment Successful."

**🔹 Step 2: Mock API Response in Cypress**

Instead of processing an actual payment, we **intercept the network request** and return a **fake success response**.

**Mock API in Cypress (cypress/e2e/gpay\_test.cy.js)**

describe('Google Pay Mock Test', () => {

it('should mock GPay payment and verify success', () => {

// Intercept the GPay request and return a mocked response

cy.intercept('POST', '/gpay/payment/confirm', {

statusCode: 200,

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

}).as('mockGPay');

// Visit the checkout page

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

// Click the Google Pay button

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

// Wait for the mock API to be called

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

// Validate success message

cy.get('.order-confirmation').should('contain', 'Payment Successful');

// Verify the mocked transaction ID

cy.get('.transaction-id').should('contain', 'MOCK-12345');

});

});

✅ **Why is this useful?**

* **No real payments required** (safe for automation).
* **Works in CI/CD pipelines** without dependencies.
* **Fast and reliable testing** without waiting for real payments.

**🔹 Step 3: Run Cypress Test**

1. Open Cypress:

npx cypress open

1. Select and run gpay\_test.cy.js.

**🎯 Bonus: Testing Different Payment Scenarios**

You can modify the mock API to simulate different cases:

**🔴 Mock Payment Failure**

cy.intercept('POST', '/gpay/payment/confirm', {

statusCode: 400,

body: { success: false, error: 'Insufficient funds' },

}).as('mockGPayFail');

✅ Verify that the UI displays **"Payment Failed"**.

If you're integrating **Google Pay (GPay) automation** in Cypress for your e-commerce application, here's a **detailed approach** without using real payments.

**🔹 Steps to Automate Google Pay in Cypress**

**1️. Identify Key Elements**

Google Pay buttons are typically embedded using an <iframe>. Cypress does **not interact directly** with cross-origin iframes, so we use **mocking & intercepting API requests**.

**2️. Mock API Requests Instead of Real Payments**

Since you **can't automate real transactions**, Cypress will **mock API responses** to simulate a successful or failed payment.

**🔹 Cypress Test Script for Google Pay**

**📌 Test Case: Simulate a Successful GPay Payment**

describe('Google Pay Payment Automation', () => {

beforeEach(() => {

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

});

it('should mock Google Pay payment and verify success', () => {

// Intercept the API call when Google Pay is clicked

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

statusCode: 200,

body: { success: true, transactionId: 'GPay-123456' }

}).as('mockGPaySuccess');

// Click the Google Pay button

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

// Wait for the request to be intercepted

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

// Check for success message

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

// Verify the transaction ID

cy.get('.transaction-id').should('contain', 'GPay-123456');

});

});

✅ **Key Points:**

* The test intercepts the API call when the **Google Pay button is clicked**.
* Instead of sending a real request, Cypress **mocks a success response**.
* Verifies that the **UI displays a success message** and **transaction ID**.

**🔹 Test Case: Simulate Payment Failure**

You can also **test failure scenarios** by returning an error response.

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

statusCode: 400,

body: { success: false, error: 'Payment declined' }

}).as('mockGPayFail');

// Click the Google Pay button

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

// Wait for the failed request

cy.wait('@mockGPayFail').its('response.statusCode').should('eq', 400);

// Verify failure message

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

✅ **This ensures that your UI properly handles failed payments.**

**🔹 Running Your Cypress Test**

1. **Open Cypress UI**

npx cypress open

1. **Run the test file** (e.g., gpay\_test.cy.js).
2. **Verify that Cypress simulates both success & failure cases correctly.**

**🎯 Bonus: Handling Google Pay Iframes**

If the GPay button is inside an <iframe>, use this method:

cy.get('iframe').its('0.contentDocument.body')

.should('not.be.empty')

.then(cy.wrap)

.find('#gpay-button')

.click();

**✅ Final Thoughts**

* **Mock API requests** instead of performing real payments.
* **Intercept GPay responses** to test different scenarios.
* **Use iframe handling** if needed for button clicks.

**Setting Up PayPal Sandbox for Cypress Automation**

To automate PayPal transactions in Cypress, you'll need a **PayPal Developer account** and a **Sandbox environment**.

**🔹 Step 1: Create a PayPal Developer Account**

1. **Go to** PayPal Developer Portal.
2. **Sign in** with your PayPal account (or create one).
3. Click on **"Dashboard"** to access sandbox tools.

**🔹 Step 2: Create Sandbox Accounts**

1. Navigate to **"Sandbox" > "Accounts"**.
2. Click **"Create Account"**.
3. Select **"Business"** as account type (this acts as the seller).
4. Select another **"Personal"** account (this is the buyer).
5. Click **"Create"**.

✅ **Result:** You now have two test accounts:

* **Business Account** → Seller (receives money).
* **Personal Account** → Buyer (makes payment).

**🔹 Step 3: Get API Credentials**

1. Go to **"My Apps & Credentials"**.
2. Under **"REST API apps"**, click **"Create App"**.
3. Give your app a name (e.g., **Ecommerce Test**).
4. Select **"Sandbox"** and click **"Create App"**.
5. Copy your **Client ID** and **Secret**.

**🔹 Step 4: Use Sandbox Credentials in Cypress**

You'll use the **Sandbox Personal Account** (buyer) in your Cypress tests.

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

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

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

// Click the PayPal checkout button

cy.get('iframe')

.its('0.contentDocument.body')

.should('not.be.empty')

.then(cy.wrap)

.find('#paypal-button')

.click();

// Switch to PayPal's sandbox login

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

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

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

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

});

});

**🔹 Step 5: Test Your Automation**

1. **Run the Cypress test**:

npx cypress open

1. The test should:
   * Click **PayPal checkout**.
   * Log in using the **sandbox buyer credentials**.
   * Confirm the payment.
   * Return to your site with a **success message**.

**📌 Key Notes**

* Replace **sandbox-buyer@example.com** and **sandbox-password** with your **actual sandbox credentials**.
* If Cypress fails to interact with PayPal UI, try **increasing timeouts**:

cy.get('input#email', { timeout: 10000 }).type('sandbox-buyer@example.com');

* If PayPal uses an **iframe**, wrap it before interacting.