Testing Environment - Make Tests Actually Run

Playwright Configuration

javascript

```ts
// playwright.config.ts
import { defineConfig, devices } from '@playwright/test';

export default defineConfig({
  testDir: './tests/e2e',
  timeout: 5 * 60 * 1000, // 5 minutes
  expect: {
    timeout: 30000
  },
  fullyParallel: true,
  forbidOnly: !!process.env.CI,
  retries: process.env.CI ? 2 : 0,
  workers: process.env.CI ? 1 : undefined,
  reporter: 'html',
  use: {
    baseURL: 'http://localhost:3000',
    trace: 'on-first-retry',
    screenshot: 'only-on-failure',
  },
  projects: [
    {
      name: 'chromium',
      use: { ...devices['Desktop Chrome'] },
    },
    {
      name: 'Mobile Chrome',
      use: { ...devices['iPhone 12'] },
    },
  ],
  webServer: {
    command: 'npm run dev',
    port: 3000,
    reuseExistingServer: !process.env.CI,
```

```
  },
});
```

Jest Configuration

javascript

```javascript
// jest.config.js
module.exports = {
  preset: 'ts-jest',
  testEnvironment: 'jsdom',
  setupFilesAfterEnv: ['<rootDir>/jest.setup.js'],
  moduleNameMapper: {
    '^@/(.*)$': '<rootDir>/src/$1',
    '\\.(css|less|scss|sass)$': 'identity-obj-proxy',
  },
  transform: {
    '^.+\\.(ts|tsx)$': 'ts-jest',
  },
  collectCoverageFrom: [
    'src/**/*.{ts,tsx}',
    '!src/**/*.d.ts',
    '!src/**/*.stories.tsx',
  ],
};
```

javascript

```javascript
// jest.setup.js
import '@testing-library/jest-dom';

// Mock window.ethereum
global.window.ethereum = {
  request: jest.fn(),
  on: jest.fn(),
  removeListener: jest.fn(),
};

// Mock IntersectionObserver
global.IntersectionObserver = class IntersectionObserver {
  constructor() {}
  disconnect() {}
  observe() {}
  unobserve() {}
};
```

Mock Functions for E2E Tests

javascript

```typescript
// tests/e2e/helpers/setup.ts
import { MetaMaskWallet } from '@chainsafe/dappeteer';

export async function setupMetaMask(): Promise<string> {
  // This would be replaced with actual MetaMask setup
  return process.env.METAMASK_PATH || '/path/to/metamask';
}

export async function getMetaMask(context: any): Promise<any> {
  // Mock implementation
  return {
    importWallet: async (mnemonic: string) => true,
    approve: async () => true,
    acceptNetworkSwitch: async () => true,
  };
}

export async function connectWallet(page: any, metamask: any): Promise<void> {
  await page.click('[data-testid="connect-wallet-button"]');
  await page.click('[data-testid="wallet-option-metamask"]');
  await page.waitForSelector('[data-testid="wallet-address"]');
}

export async function setupConnectedWallet(page: any): Promise<void> {
  await page.goto('http://localhost:3000');
  await page.evaluate(() => {
    // Mock wallet connection
    (window as any).mockWalletConnection = (config: any) => {
      localStorage.setItem('walletConnected', 'true');
      localStorage.setItem('walletAddress', config.address);
      localStorage.setItem('chainId', config.chainId.toString());
      window.dispatchEvent(new Event('walletConnected'));
    };
```

```
  });

  await page.evaluate(() => {
    (window as any).mockWalletConnection({
      address: '0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266',
      chainId: 1,
      balance: '10000000000000000000'
    });
  });
}
```

Environment Variables Template

bash

```
# .env.example
# RPC URLs
ETHEREUM_RPC_URL=https://eth-mainnet.g.alchemy.com/v2/YOUR_KEY
ARBITRUM_RPC_URL=https://arb-mainnet.g.alchemy.com/v2/YOUR_KEY
POLYGON_RPC_URL=https://polygon-mainnet.g.alchemy.com/v2/YOUR_KEY
OPTIMISM_RPC_URL=https://opt-mainnet.g.alchemy.com/v2/YOUR_KEY

# Fork RPCs for testing
ETHEREUM_FORK_RPC=http://localhost:8545
ARBITRUM_FORK_RPC=http://localhost:8546
POLYGON_FORK_RPC=http://localhost:8547
OPTIMISM_FORK_RPC=http://localhost:8548

# Private Keys (NEVER commit real keys!)
PRIVATE_KEY=0xac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae784d7bf4f2ff80

# API Keys
ETHERSCAN_API_KEY=YOUR_KEY
ARBISCAN_API_KEY=YOUR_KEY
POLYGONSCAN_API_KEY=YOUR_KEY
OPTIMISM_API_KEY=YOUR_KEY

# Chainlink
CHAINLINK_DATA_STREAMS_URL=https://api.chain.link/v1/data-streams
CHAINLINK_FUNCTIONS_URL=https://api.chain.link/v1/functions
CHAINLINK_AUTOMATION_URL=https://api.chain.link/v1/automation

# WalletConnect
WALLETCONNECT_PROJECT_ID=YOUR_PROJECT_ID

# Test Configuration
```

```
TEST_MNEMONIC="test test test test test test test test test test test junk"
METAMASK_PATH=/path/to/metamask-extension
```

Docker Setup for Testing

yaml

```yaml
# docker-compose.test.yml
version: '3.8'

services:
  ethereum-fork:
    image: trufflesuite/ganache:latest
    ports:
      - "8545:8545"
    command: >
      -f https://eth-mainnet.g.alchemy.com/v2/${ALCHEMY_KEY}
      --accounts 10
      --account_keys_path /keys.json
      --networkId 1
      --chainId 1
      --gasLimit 30000000
    volumes:
      - ./test-keys.json:/keys.json

  arbitrum-fork:
    image: trufflesuite/ganache:latest
    ports:
      - "8546:8545"
    command: >
      -f https://arb-mainnet.g.alchemy.com/v2/${ALCHEMY_KEY}
      --accounts 10
      --networkId 42161
      --chainId 42161

  polygon-fork:
    image: trufflesuite/ganache:latest
    ports:
      - "8547:8545"
    command: >
```

```
      -f https://polygon-mainnet.g.alchemy.com/v2/${ALCHEMY_KEY}
      --accounts 10
      --networkId 137
      --chainId 137

  optimism-fork:
    image: trufflesuite/ganache:latest
    ports:
      - "8548:8545"
    command: >
      -f https://opt-mainnet.g.alchemy.com/v2/${ALCHEMY_KEY}
      --accounts 10
      --networkId 10
      --chainId 10
```

Test Running Scripts

json

```
// package.json scripts section
{
  "scripts": {
    "test:setup": "docker-compose -f docker-compose.test.yml up -d",
    "test:teardown": "docker-compose -f docker-compose.test.yml down",
    "test:contracts": "hardhat test",
    "test:e2e": "playwright test",
    "test:e2e:ui": "playwright test --ui",
    "test:unit": "jest",
    "test:all": "npm run test:setup && npm run test:contracts && npm run test:unit && npm
  }
}
```