Feb 11,2026

TESTING REPORT

Report Contents

This report provides key insights from TestSprite's AI-powered testing. For questions or customized needs, contact us using Calendly or join our Discord community.

Table of Contents

Executive Summary

1 High-Level Overview

OVERVIEW	
Total APIs Tested	0 APIs
Total Websites Tested	1 Websites
Pass/Fail Rate	Backend: 0/0 Frontend: 4/19

2 Key Findings

Test Summary

The project shows a moderate quality level primarily due to the lack of backend testing. While frontend performance appears stable, the absence of backend assessment raises concerns about API reliability, potentially affecting user satisfaction negatively. This gap necessitates urgent attention to ensure project stability and enhance user experience.

What could be better

A significant weakness is the total absence of backend testing results, leading to doubts about API functionality and reliability. Without assessing backend performance, critical failures may go undetected, jeopardizing the project's credibility and user satisfaction.

Recommendations

Immediate improvement efforts should focus on implementing a robust backend testing strategy. This includes defining clear success criteria and ensuring comprehensive coverage of API tests to mitigate risks associated with undetected failures. Enhancing backend quality will directly improve user experience.

Frontend UI Test Results

3 Test Coverage Summary

This report summarizes the frontend UI testing results for the application. TestSprite's AI agent automatically generated and executed tests based on the UI structure, user interaction flows, and visual components. The tests aimed to validate core functionalities, visual correctness, and responsiveness across different states.

URL NAME	TEST CASES	PASS/FAIL RATE
Ligeirinho_hotdog-frontend	23	4 Pass/19 Fail

Note

The test cases were generated using real-time analysis of the application's UI hierarchy and user flows. Some visual and functional validations were adapted dynamically based on runtime DOM changes.

4 Test Execution Summary

Ligeirinho_hotdog-Frontend Execution Summary

TEST CASE	TEST DESCRIPTION	IMPACT	STATUS
TC008-Manage menu categories and products CRUD operations	Verify the creation, reading, updating and deletion of menu categories and products including image upload with cropping and compression.	MEDIUM	Failed
TC009-Order management with Kanban drag-and-drop board	Verify order statuses can be updated via drag-and-drop on the Kanban board and state changes persist correctly.	MEDIUM	Failed
TC023-Error handling on failed data synchronization during offline mode	Verify system behavior and error handling when data synchronization fails after offline usage.	MEDIUM	Failed
TC011-Order checkout via WhatsApp formatted message	Verify that checkout sends the customer's order details formatted as a WhatsApp message to the snack bar correctly.	MEDIUM	Failed
TC014-Offline data caching and synchronization for customers	Verify the digital menu and order system caches data locally for offline use and automatically synchronizes when connectivity returns.	MEDIUM	Failed
TC017-Order notifications via Telegram integration	Verify that order notifications are sent correctly via Telegram with correct formatting and content.	MEDIUM	Failed
TC021-Real-time order tracking and status update verification	Verify that order status updates are reflected instantly on both admin and customer views using Supabase Realtime.	MEDIUM	Failed
TC018-QR code generation for physical tables or counters	Verify that QR codes can be generated and downloaded for tables/counters enabling easy customer menu access.	MEDIUM	Failed
TC012-PIX payment processing with AbacatePay integration	Verify that payments via PIX through AbacatePay complete successfully with QR code generation and display.	MEDIUM	Failed
TC010-Customer digital menu browsing and ordering with upsell suggestions	Verify customers can browse the product catalog, add products to basket, see upsell suggestions, and proceed to checkout.	MEDIUM	Failed
TC020-Security validation of Row Level Security (RLS) in Supabase	Verify that Supabase RLS policies correctly restrict data access to authorized users only.	MEDIUM	Failed
TC006-Admin Dashboard access control	Verify that only authenticated admin users can access the Admin Dashboard and unauthorized access is prevented.	MEDIUM	Passed
TC003-Login success with valid credentials	Verify that users can login successfully when providing correct credentials.	MEDIUM	Passed
TC013-PIX payment processing with Stripe integration	Verify PIX payment completion with Stripe API and correct QR code generation.	MEDIUM	Failed
TC002-User registration with invalid email format	Verify the system rejects registration with an improperly formatted email address.	MEDIUM	Failed
TC015-Offline data caching and synchronization for Admin Dashboard	Verify that admin users can perform management tasks offline and data syncs properly when online.	MEDIUM	Failed
TC022-UI/UX responsiveness and brand color adaptation on mobile devices	Verify the application is fully responsive and adapts brand colors correctly on various mobile screen sizes.	MEDIUM	Passed
TC007-Admin Dashboard financial data accuracy and realtime update	Verify that financial data including daily and monthly sales, cash flow and client subscriptions are shown accurately and update in real time.	MEDIUM	Failed
TC016-Order notifications via WhatsApp integration	Verify that order notifications are sent correctly via WhatsApp to the snack bar as formatted messages.	MEDIUM	Failed

TEST CASE	TEST DESCRIPTION	IMPACT	STATUS
TC001-Successful user registration	Verify that a new user can register successfully with valid input data.	MEDIUM	Failed
TC019-User and store branding customization	Verify that admin users can customize UI branding including colors and logos and these reflect in the app immediately.	MEDIUM	Failed
TC004-Login failure with invalid credentials	Verify that login fails and appropriate error is shown when credentials are incorrect.	MEDIUM	Passed
TC005-Password recovery workflow	Verify that the password recovery process initiates and completes properly.	MEDIUM	Failed

5 Test Execution Breakdown

Ligeirinho_hotdog-Frontend Failed Test Details

TC008-Manage menu categories and products CRUD operations

ATTRIBUTES	
Status	Failed
Priority	MEDIUM
Description	Verify the creation, reading, updating and deletion of menu categories and products including image upload with cropping and compression.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/5478f4b8-00c1-70cc-8db6-48f4cfc182b1/1770755510016998/result.webm

Test Code

```python
import asyncio
from playwright import async_api

async def run_test():
    pw = None
    browser = None
    context = None

    try:
        # Start a Playwright session in asynchronous mode
        pw = await async_api.async_playwright().start()

        # Launch a Chromium browser in headless mode with custom arguments
        browser = await pw.chromium.launch(
            headless=True,
            args=[
                "--window-size=1280,720",         # Set the browser window size
                "--disable-dev-shm-usage",        # Avoid using /dev/shm which can cause issues in containers
                "--ipc=host",                     # Use host-level IPC for better stability
                "--single-process"                # Run the browser in a single process mode
            ],
        )

        # Create a new browser context (like an incognito window)
        context = await browser.new_context()
        context.set_default_timeout(5000)

        # Open a new page in the browser context
        page = await context.new_page()

        # Navigate to your target URL and wait until the network request is committed
        await page.goto("https://usecolaai.vercel.app", wait_until="commit", timeout=10000)

        # Wait for the main page to reach DOMContentLoaded state (optional for stability)
        try:
            await page.wait_for_load_state("domcontentloaded", timeout=3000)
        except async_api.Error:
            pass

        # Iterate through all iframes and wait for them to load as well
        for frame in page.frames:
            try:
                await frame.wait_for_load_state("domcontentloaded", timeout=3000)
            except async_api.Error:
                pass

        # Interact with the page elements to simulate user flow
```

```
48          # -> Navigate to https://usecolaai.vercel.app
49          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
50
51          # -> Preencher o campo E-mail, preencher o campo Senha e
            clicar em 'Entrar' para autenticar como admin.
52          frame = context.pages[-1]
53          # Input text
54          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[1]/div/input').nth(0)
55          await page.wait_for_timeout(3000); await elem.fill
            ('testefinal99@example.com')
56
57          frame = context.pages[-1]
58          # Input text
59          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/input').nth(0)
60          await page.wait_for_timeout(3000); await elem.fill
            ('test123456')
61
62          frame = context.pages[-1]
63          # Click element
64          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/button').nth(0)
65          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
66
67          # -> Navegar para a seção 'Categorias' do menu lateral para
            iniciar a criação de uma nova categoria.
68          frame = context.pages[-1]
69          # Click element
70          elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[4]').
            nth(0)
71          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
72
73          # -> Tentar novamente navegar para a seção 'Categorias'
            clicando no item do menu lateral (índice 879).
74          frame = context.pages[-1]
75          # Click element
76          elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[4]').
            nth(0)
77          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
78
79          # -> Abrir o formulário de criação clicando no botão 'Nova
            Categoria' (index 1086).
80          frame = context.pages[-1]
81          # Click element
82          elem = frame.locator('xpath=html/body/div[2]/main/div/div[1]/
            button').nth(0)
83          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
84
85          # -> Preencher o nome da categoria, selecionar um ícone e
            clicar 'Criar' para salvar a nova categoria.
86          frame = context.pages[-1]
87          # Input text
88          elem = frame.locator('xpath=html/body/div[2]/main/div/div[5]/
```

```
            div/div[1]/div[1]/div/input').nth(0)
 89         await page.wait_for_timeout(3000); await elem.fill('Categoria
            Automação Teste')
 90
 91         frame = context.pages[-1]
 92         # Click element
 93         elem = frame.locator('xpath=html/body/div[2]/main/div/div[5]/
            div/div[1]/div[2]/div/button[1]').nth(0)
 94         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
 95
 96         frame = context.pages[-1]
 97         # Click element
 98         elem = frame.locator('xpath=html/body/div[2]/main/div/div[5]/
            div/div[2]/button[2]').nth(0)
 99         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
100
101         # -> Clicar novamente no botão 'Criar' para submeter a
            categoria (segunda tentativa, index 1355). Após o clique,
            verificar se a categoria aparece na lista e se o modal fecha.
102         frame = context.pages[-1]
103         # Click element
104         elem = frame.locator('xpath=html/body/div[2]/main/div/div[5]/
            div/div[2]/button[1]').nth(0)
105         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
106
107         # -> Clicar no menu 'Produtos' (índice 877) para iniciar a
            criação do produto com upload de imagem (usar ferramentas de
            crop e compress).
108         frame = context.pages[-1]
109         # Click element
110         elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[2]').
            nth(0)
111         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
112
113         # -> Confirmar abertura da seção 'Produtos' clicando no item
            do menu (index 877) e aguardar o carregamento da página.
114         frame = context.pages[-1]
115         # Click element
116         elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[2]').
            nth(0)
117         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
118
119         # -> Abrir o formulário de criação de produto clicando em
            'Novo Produto' e iniciar o fluxo de inclusão de produto
            (incluindo upload de imagem com as ferramentas de crop e
            compress).
120         frame = context.pages[-1]
121         # Click element
122         elem = frame.locator('xpath=html/body/div[2]/main/div/div[1]/
            button').nth(0)
123         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
124
125         # -> Enviar foto do produto, preencher nome 'Produto
```

```
            Automação Teste', selecionar categoria 'Categoria Automação
            Teste', preencher preço e descrição, clicar 'Criar Produto'
            para salvar o produto (usar ferramentas de crop/compress no
            fluxo de upload se acionarem). Em seguida, verificar que o
            produto aparece na lista (próximo passo após criação).
126         frame = context.pages[-1]
127         # Input text
128         elem = frame.locator('xpath=html/body/div[2]/main/div/div[6]/
            div/div[1]/div[2]/div/input').nth(0)
129         await page.wait_for_timeout(3000); await elem.fill('Produto
            Automação Teste')
130
131         # -> Preencher os campos do formulário do produto (nome,
            categoria, preço, descrição) para preparar a criação, mas
            aguardar o arquivo de imagem. Solicitar ao usuário que
            forneça o caminho do arquivo de imagem disponível no ambiente
            do agente (ex.: /tmp/test-image.jpg) ou que adicione esse
            caminho em available_file_paths para que o upload, crop e
            compress possam ser executados. Após envio do arquivo,
            realizar upload pelo input (index 4028), usar as ferramentas
            de crop/compress na UI, clicar 'Criar Produto' e então
            verificar que o produto e a imagem aparecem na lista.
132         frame = context.pages[-1]
133         # Input text
134         elem = frame.locator('xpath=html/body/div[2]/main/div/div[6]/
            div/div[1]/div[2]/div/input').nth(0)
135         await page.wait_for_timeout(3000); await elem.fill('Produto
            Automação Teste')
136
137         frame = context.pages[-1]
138         # Input text
139         elem = frame.locator('xpath=html/body/div[2]/main/div/div[6]/
            div/div[1]/div[3]/div[2]/div/input').nth(0)
140         await page.wait_for_timeout(3000); await elem.fill('25')
141
142         await asyncio.sleep(5)
143
144     finally:
145         if context:
146             await context.close()
147         if browser:
148             await browser.close()
149         if pw:
150             await pw.stop()
151
152 asyncio.run(run_test())
153
```

Error

TimeoutError: Locator.click: Timeout 5000ms exceeded. Call log: - waiting for locator("xpath=html/body/div[2]/main/div/div[5]/div/div[2]/button[1]").first

Cause

The target element for the click action may not be visible or interactable due to DOM rendering issues, CSS visibility settings, or JavaScript errors preventing the button from being clickable.

Fix

Investigate the rendering behavior of the page to ensure the button is fully loaded and visible. Implement checks to wait for the element's visibility or correct any JavaScript errors that might be causing the rendering delays. Additionally, review any CSS styles that could be affecting the button's interactability.

TC009-Order management with Kanban drag-and-drop board

ATTRIBUTES

Status	Failed
Priority	MEDIUM
Description	Verify order statuses can be updated via drag-and-drop on the Kanban board and state changes persist correctly.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/5478f4b8-00c1-70cc-8db6-48f4cfc182b1/1770755495902579/result.webm

Test Code

```python
import asyncio
from playwright import async_api

async def run_test():
    pw = None
    browser = None
    context = None

    try:
        # Start a Playwright session in asynchronous mode
        pw = await async_api.async_playwright().start()

        # Launch a Chromium browser in headless mode with custom arguments
        browser = await pw.chromium.launch(
            headless=True,
            args=[
                "--window-size=1280,720",         # Set the browser window size
                "--disable-dev-shm-usage",        # Avoid using /dev/shm which can cause issues in containers
                "--ipc=host",                     # Use host-level IPC for better stability
                "--single-process"                # Run the browser in a single process mode
            ],
        )

        # Create a new browser context (like an incognito window)
        context = await browser.new_context()
        context.set_default_timeout(5000)

        # Open a new page in the browser context
        page = await context.new_page()

        # Navigate to your target URL and wait until the network request is committed
        await page.goto("https://usecolaai.vercel.app", wait_until="commit", timeout=10000)

        # Wait for the main page to reach DOMContentLoaded state (optional for stability)
        try:
            await page.wait_for_load_state("domcontentloaded", timeout=3000)
        except async_api.Error:
            pass

        # Iterate through all iframes and wait for them to load as well
        for frame in page.frames:
            try:
                await frame.wait_for_load_state("domcontentloaded", timeout=3000)
            except async_api.Error:
                pass

        # Interact with the page elements to simulate user flow
```

```
48          # -> Navigate to https://usecolaai.vercel.app
49          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
50
51          # -> Login as admin (use provided credentials) and open the
            order management Kanban board.
52          frame = context.pages[-1]
53          # Input text
54          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[1]/div/input').nth(0)
55          await page.wait_for_timeout(3000); await elem.fill
            ('testefinal99@example.com')
56
57          frame = context.pages[-1]
58          # Input text
59          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/input').nth(0)
60          await page.wait_for_timeout(3000); await elem.fill
            ('test123456')
61
62          frame = context.pages[-1]
63          # Click element
64          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/button').nth(0)
65          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
66
67          # -> Click the 'Entrar' button (index 131) to submit the
            login and open the order management Kanban board; then wait
            for navigation to complete.
68          frame = context.pages[-1]
69          # Click element
70          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/button').nth(0)
71          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
72
73          # -> Click the 'Pedidos' navigation item to open the order
            management / Kanban board and wait for the page to load.
74          frame = context.pages[-1]
75          # Click element
76          elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[5]').
            nth(0)
77          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
78
79          # -> Click the 'Pedidos' navigation item (index 915) to open
            the order management Kanban board, then wait for the page to
            load and check for Kanban elements.
80          frame = context.pages[-1]
81          # Click element
82          elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[5]').
            nth(0)
83          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
84
85          # -> Click the 'Iniciar Preparo' button on order #13 (Cliente
            06) to move it from 'Aguardando' to 'Preparando', then verify
            the status changed and that it persists after navigating away
```

```
            and back.
            frame = context.pages[-1]
            # Click element
            elem = frame.locator('xpath=html/body/div[2]/main/div/div[3]/
            div/div[1]/div[3]/div/button').nth(0)
            await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)

            # -> Recarregar a página de Pedidos e extrair novamente os
            detalhes do pedido #13 para confirmar que o status persiste
            após navegação.
            await page.goto("https://usecolaai.vercel.app/pedidos",
            wait_until="commit", timeout=10000)

            # --> Assertions to verify final state
            frame = context.pages[-1]
            try:
                await expect(frame.locator('text=Pedido #13: Entregue').
                first).to_be_visible(timeout=3000)
            except AssertionError:
                raise AssertionError("Test case failed: expected order
                #13 to display status 'Entregue' on the Kanban board
                after dragging it through the statuses and reloading the
                page, but the visual status or persisted backend state
                was not found.")
            await asyncio.sleep(5)

        finally:
            if context:
                await context.close()
            if browser:
                await browser.close()
            if pw:
                await pw.stop()

    asyncio.run(run_test())

```

Error

TimeoutError: Locator.click: Timeout 5000ms exceeded. Call log: - waiting for locator("xpath=html/body/div[2]/div[2]/div[2]/div/form/div[2]/div/button").first

Cause

The button may not be rendered in the DOM within the allocated timeout period, possibly due to slow rendering or a JavaScript error preventing the UI from loading correctly.

Fix

Optimize the loading performance of the button by reviewing the JavaScript execution for errors, ensuring that all necessary assets are loaded correctly, and potentially increasing the timeout duration for the tests.

TC023-Error handling on failed data synchronization during offline mode

ATTRIBUTES

Status	Failed
Priority	MEDIUM
Description	Verify system behavior and error handling when data synchronization fails after offline usage.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/5478f4b8-00c1-70cc-8db6-48f4cfc182b1/1770755497954352/result.webm

Test Code

```python
import asyncio
from playwright import async_api

async def run_test():
    pw = None
    browser = None
    context = None

    try:
        # Start a Playwright session in asynchronous mode
        pw = await async_api.async_playwright().start()

        # Launch a Chromium browser in headless mode with custom arguments
        browser = await pw.chromium.launch(
            headless=True,
            args=[
                "--window-size=1280,720",         # Set the browser window size
                "--disable-dev-shm-usage",        # Avoid using /dev/shm which can cause issues in containers
                "--ipc=host",                     # Use host-level IPC for better stability
                "--single-process"                # Run the browser in a single process mode
            ],
        )

        # Create a new browser context (like an incognito window)
        context = await browser.new_context()
        context.set_default_timeout(5000)

        # Open a new page in the browser context
        page = await context.new_page()

        # Navigate to your target URL and wait until the network request is committed
        await page.goto("https://usecolaai.vercel.app", wait_until="commit", timeout=10000)

        # Wait for the main page to reach DOMContentLoaded state (optional for stability)
        try:
            await page.wait_for_load_state("domcontentloaded", timeout=3000)
        except async_api.Error:
            pass

        # Iterate through all iframes and wait for them to load as well
        for frame in page.frames:
            try:
                await frame.wait_for_load_state("domcontentloaded", timeout=3000)
            except async_api.Error:
                pass

        # Interact with the page elements to simulate user flow
```

```
48          # -> Navigate to https://usecolaai.vercel.app
49          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
50
51          # -> Fazer login com as credenciais de teste para iniciar os
            cenários (entrar na aplicação).
52          frame = context.pages[-1]
53          # Input text
54          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[1]/div/input').nth(0)
55          await page.wait_for_timeout(3000); await elem.fill
            ('testefinal99@example.com')
56
57          frame = context.pages[-1]
58          # Input text
59          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/input').nth(0)
60          await page.wait_for_timeout(3000); await elem.fill
            ('test123456')
61
62          frame = context.pages[-1]
63          # Click element
64          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/button').nth(0)
65          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
66
67          # -> Abrir o formulário de criação de pedido clicando em '+
            Novo Pedido' para iniciar a criação de alterações que serão
            usadas no teste offline (clicar no elemento index 670).
68          frame = context.pages[-1]
69          # Click element
70          elem = frame.locator('xpath=html/body/div[2]/main/div/div[1]/
            a/button').nth(0)
71          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
72
73          # -> Abrir o formulário de criação de pedido — clicar no
            botão 'Criar Pedido' (index 1007) para forçar a abertura do
            fluxo de criação de pedido e prosseguir com o teste offline.
74          frame = context.pages[-1]
75          # Click element
76          elem = frame.locator('xpath=html/body/div[2]/main/div/div[4]/
            div[1]/div[2]/a/button').nth(0)
77          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
78
79          # -> Adicionar um produto ao carrinho para representar as
            alterações feitas enquanto o usuário está offline. Clicar no
            produto 'Hotdog Especial 1' (element index 1747) para
            colocá-lo no pedido.
80          frame = context.pages[-1]
81          # Click element
82          elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/div[2]/div[1]/div').nth(0)
83          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
84
85          # -> Abrir o painel do Carrinho para visualizar o pedido
```

```
            (resumo) clicando no container do carrinho — em seguida
            tentar avançar para submissão para acionar a sincronização e
            observar comportamento de erro/retry.
 86         frame = context.pages[-1]
 87         # Click element
 88         elem = frame.locator('xpath=html/body/div[2]/main/div/div[3]/
            div').nth(0)
 89         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
 90
 91         # -> Tentar submeter o pedido clicando em 'Criar Pedido' para
            acionar a sincronização com o servidor e observar
            comportamento (mensagem de erro, retry e preservação dos
            dados do carrinho).
 92         frame = context.pages[-1]
 93         # Click element
 94         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[2]/div/div[1]/div/div[2]/button').nth(0)
 95         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
 96
 97         # -> Adicionar produto ao carrinho (garantir alteração local)
            e revelar/ localizar o botão 'Criar Pedido' (rolar/
            pesquisar), preparando para tentar submeter o pedido e
            observar comportamento de sincronização.
 98         frame = context.pages[-1]
 99         # Click element
100         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/div[2]/div[1]/div').nth(0)
101         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
102
103         # -> Preencher nome e telefone do cliente no carrinho e
            selecionar PIX como forma de pagamento (pré-condição antes de
            tentar submeter/simular offline).
104         frame = context.pages[-1]
105         # Input text
106         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[2]/div/div[3]/div[1]/div/input').nth(0)
107         await page.wait_for_timeout(3000); await elem.fill('Cliente
            Teste')
108
109         frame = context.pages[-1]
110         # Input text
111         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[2]/div/div[3]/div[2]/div/input').nth(0)
112         await page.wait_for_timeout(3000); await elem.fill('(00)
            00000-0000')
113
114         frame = context.pages[-1]
115         # Click element
116         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[2]/div/div[5]/div/button[2]').nth(0)
117         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
118
119         # -> Tentar submeter o pedido clicando em 'Criar Pedido'
            (index 2071) para acionar a sincronização; aguardar a
            resposta e extrair mensagens de erro/alertas visíveis para
```

```
            avaliar comportamento de falha ou sucesso.
120         frame = context.pages[-1]
121         # Click element
122         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[2]/div/button').nth(0)
123         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
124
125         await asyncio.sleep(5)
126
127     finally:
128         if context:
129             await context.close()
130         if browser:
131             await browser.close()
132         if pw:
133             await pw.stop()
134
135 asyncio.run(run_test())
136
```

Error

TimeoutError: Locator.click: Timeout 5000ms exceeded. Call log: - waiting for locator("xpath=html/body/div[2]/main/div/div[4]/div[1]/div[2]/a/button").first

Cause

The element specified by the locator may not be present in the DOM at the time of the click attempt, possibly due to slow rendering or dynamic content loading.

Fix

Increase the timeout duration or implement a wait mechanism to ensure the element is available before attempting to click, such as using an explicit wait for visibility.

TC011-Order checkout via WhatsApp formatted message

ATTRIBUTES

Status	Failed
Priority	MEDIUM
Description	Verify that checkout sends the customer's order details formatted as a WhatsApp message to the snack bar correctly.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/5478f4b8-00c1-70cc-8db6-48f4cfc182b1/1770755497036706/result.webm

Test Code

```python
import asyncio
from playwright import async_api

async def run_test():
    pw = None
    browser = None
    context = None

    try:
        # Start a Playwright session in asynchronous mode
        pw = await async_api.async_playwright().start()

        # Launch a Chromium browser in headless mode with custom arguments
        browser = await pw.chromium.launch(
            headless=True,
            args=[
                "--window-size=1280,720",         # Set the browser window size
                "--disable-dev-shm-usage",        # Avoid using /dev/shm which can cause issues in containers
                "--ipc=host",                     # Use host-level IPC for better stability
                "--single-process"                # Run the browser in a single process mode
            ],
        )

        # Create a new browser context (like an incognito window)
        context = await browser.new_context()
        context.set_default_timeout(5000)

        # Open a new page in the browser context
        page = await context.new_page()

        # Navigate to your target URL and wait until the network request is committed
        await page.goto("https://usecolaai.vercel.app", wait_until="commit", timeout=10000)

        # Wait for the main page to reach DOMContentLoaded state (optional for stability)
        try:
            await page.wait_for_load_state("domcontentloaded", timeout=3000)
        except async_api.Error:
            pass

        # Iterate through all iframes and wait for them to load as well
        for frame in page.frames:
            try:
                await frame.wait_for_load_state("domcontentloaded", timeout=3000)
            except async_api.Error:
                pass

        # Interact with the page elements to simulate user flow
```

```
48          # -> Navigate to https://usecolaai.vercel.app
49          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
50
51          # -> Efetuar login usando as credenciais de teste preenchendo
            o e-mail e senha e clicando em Entrar para acessar a
            aplicação e continuar com o fluxo de compra.
52          frame = context.pages[-1]
53          # Input text
54          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[1]/div/input').nth(0)
55          await page.wait_for_timeout(3000); await elem.fill
            ('testefinal99@example.com')
56
57          frame = context.pages[-1]
58          # Input text
59          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/input').nth(0)
60          await page.wait_for_timeout(3000); await elem.fill
            ('test123456')
61
62          frame = context.pages[-1]
63          # Click element
64          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/button').nth(0)
65          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
66
67          # -> Abrir a interface de novo pedido clicando no botão 'Novo
            Pedido' para iniciar a inclusão de itens e prosseguir ao
            checkout.
68          frame = context.pages[-1]
69          # Click element
70          elem = frame.locator('xpath=html/body/div[2]/main/div/div[1]/
            a/button').nth(0)
71          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
72
73          # -> Abrir a interface de criação de pedido clicando no botão
            'Criar Pedido' (index 855) para iniciar inclusão de itens no
            pedido.
74          frame = context.pages[-1]
75          # Click element
76          elem = frame.locator('xpath=html/body/div[2]/main/div/div[4]/
            div[1]/div[2]/a/button').nth(0)
77          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
78
79          # -> Recarregar /forçar navegação para o dashboard para
            recuperar a UI (navegar para https://usecolaai.vercel.app/
            dashboard) e re-inspecionar elementos interativos para
            continuar com o fluxo de criação de pedido.
80          await page.goto("https://usecolaai.vercel.app/dashboard",
            wait_until="commit", timeout=10000)
81
82          # -> Tentar abrir a seção 'Pedidos' no menu lateral e
            inspecionar a página de pedidos para localizar o botão 'Novo
            Pedido' ou interface de criação de pedido. Se a seção estiver
            bloqueada pelo plano expirado, reportar que a assinatura
```

```
            bloqueia o fluxo e abortar teste.
            frame = context.pages[-1]
            # Click element
            elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[5]').
            nth(0)
            await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)

            # -> Abrir o formulário de novo pedido clicando em 'Novo
            Pedido' (index 2241) e aguardar carregamento para inspecionar
            a interface de criação de pedido.
            frame = context.pages[-1]
            # Click element
            elem = frame.locator('xpath=html/body/div[2]/main/div/div[1]/
            a/button').nth(0)
            await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)

            # -> Abrir o formulário de 'Novo Pedido' clicando no botão
            identificado (index=2505) para iniciar a inclusão de itens e
            seguir ao checkout.
            frame = context.pages[-1]
            # Click element
            elem = frame.locator('xpath=html/body/div[2]/main/div/div[1]/
            a/button').nth(0)
            await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)

            # -> Abrir o formulário de novo pedido clicando em 'Novo
            Pedido' (index 2505) para iniciar inclusão de itens no pedido
            e prosseguir ao checkout.
            frame = context.pages[-1]
            # Click element
            elem = frame.locator('xpath=html/body/div[2]/main/div/div[1]/
            a/button').nth(0)
            await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)

            # --> Assertions to verify final state
            frame = context.pages[-1]
            try:
                await expect(frame.locator('text=Pedido enviado via
                WhatsApp').first).to_be_visible(timeout=3000)
            except AssertionError:
                raise AssertionError("Test case failed: Esperava que o
                aplicativo abrisse o WhatsApp com a mensagem de pedido
                formatada contendo detalhes do pedido, informações do
                cliente e totais, mas isso não foi encontrado")
            await asyncio.sleep(5)

        finally:
            if context:
                await context.close()
            if browser:
                await browser.close()
            if pw:
                await pw.stop()

    asyncio.run(run_test())
```

```
123
```

Error

TimeoutError: Locator.click: Timeout 5000ms exceeded. Call log: - waiting for locator("xpath=html/body/div[2]/main/div/div[4]/div[1]/div[2]/a/button").first

Cause

The button locator is incorrect or the element is not present in the DOM at the time of the click, possibly due to rendering delays or asynchronous content loading.

Fix

Ensure the element is available in the DOM before attempting to click it. This could involve adding proper waits or modifying the page logic to ensure elements load fully before tests are executed.

TC014-Offline data caching and synchronization for customers

ATTRIBUTES

Status	Failed
Priority	MEDIUM
Description	Verify the digital menu and order system caches data locally for offline use and automatically synchronizes when connectivity returns.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/5478f4b8-00c1-70cc-8db6-48f4cfc182b1/1770755493991897/result.webm

Test Code

```python
import asyncio
from playwright import async_api

async def run_test():
    pw = None
    browser = None
    context = None

    try:
        # Start a Playwright session in asynchronous mode
        pw = await async_api.async_playwright().start()

        # Launch a Chromium browser in headless mode with custom arguments
        browser = await pw.chromium.launch(
            headless=True,
            args=[
                "--window-size=1280,720",         # Set the browser window size
                "--disable-dev-shm-usage",        # Avoid using /dev/shm which can cause issues in containers
                "--ipc=host",                     # Use host-level IPC for better stability
                "--single-process"                # Run the browser in a single process mode
            ],
        )

        # Create a new browser context (like an incognito window)
        context = await browser.new_context()
        context.set_default_timeout(5000)

        # Open a new page in the browser context
        page = await context.new_page()

        # Navigate to your target URL and wait until the network request is committed
        await page.goto("https://usecolaai.vercel.app", wait_until="commit", timeout=10000)

        # Wait for the main page to reach DOMContentLoaded state (optional for stability)
        try:
            await page.wait_for_load_state("domcontentloaded", timeout=3000)
        except async_api.Error:
            pass

        # Iterate through all iframes and wait for them to load as well
        for frame in page.frames:
            try:
                await frame.wait_for_load_state("domcontentloaded", timeout=3000)
            except async_api.Error:
                pass

        # Interact with the page elements to simulate user flow
```

```
48          # -> Navigate to https://usecolaai.vercel.app
49          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
50
51          # -> Sign in using provided test credentials to access the
            menu and continue with offline caching tests.
52          frame = context.pages[-1]
53          # Input text
54          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[1]/div/input').nth(0)
55          await page.wait_for_timeout(3000); await elem.fill
            ('testefinal99@example.com')
56
57          frame = context.pages[-1]
58          # Input text
59          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/input').nth(0)
60          await page.wait_for_timeout(3000); await elem.fill
            ('test123456')
61
62          frame = context.pages[-1]
63          # Click element
64          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/button').nth(0)
65          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
66
67          # -> Clicar no botão 'Entrar' para tentar efetuar login e
            carregar o menu online.
68          frame = context.pages[-1]
69          # Click element
70          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/button').nth(0)
71          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
72
73          # -> Carregar o menu digital com a rede ONLINE: clicar em
            'Produtos' para abrir a lista de itens e confirmar que o menu
            é carregado enquanto há conectividade.
74          frame = context.pages[-1]
75          # Click element
76          elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[2]').
            nth(0)
77          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
78
79          # -> Click the 'Produtos' menu item to load the digital menu
            while online and confirm the products list appears in the UI.
80          frame = context.pages[-1]
81          # Click element
82          elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[2]').
            nth(0)
83          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
84
85          # -> Abrir a tela 'Pedidos' para iniciar a criação de pedido
            (etapa preparatória antes de simular modo offline).
86          frame = context.pages[-1]
87          # Click element
```

```
88          elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[5]').
            nth(0)
89          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
90
91          # -> Click the 'Pedidos' menu item to open the orders screen
            and begin creating a new order while still online
            (preparation to simulate offline behavior).
92          frame = context.pages[-1]
93          # Click element
94          elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[5]').
            nth(0)
95          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
96
97          # -> Abrir o formulário de criação de pedido clicando em
            'Novo Pedido' (manter rede ONLINE) para então simular
            comportamento offline durante a criação do pedido.
98          frame = context.pages[-1]
99          # Click element
100         elem = frame.locator('xpath=html/body/div[2]/main/div/div[1]/
            a/button').nth(0)
101         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
102
103         # -> Abrir o formulário de 'Novo Pedido' clicando novamente
            no botão 'Novo Pedido' e verificar se o modal/form de criação
            aparece (ainda com rede ONLINE).
104         frame = context.pages[-1]
105         # Click element
106         elem = frame.locator('xpath=html/body/div[2]/main/div/div[1]/
            a/button').nth(0)
107         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
108
109         # -> Adicionar um produto ao carrinho clicando no card do
            produto e extrair o conteúdo do carrinho (nomes, quantidades,
            preços e total) para verificar se a UI atual registra a
            adição localmente.
110         frame = context.pages[-1]
111         # Click element
112         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/div[2]/div[1]').nth(0)
113         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
114
115         # -> Adicionar mais um produto ao carrinho (Hotdog Especial
            2) para depois extrair o estado atualizado do carrinho e
            tentar revelar o botão Criar Pedido (rolagem), assim preparar
            para os testes offline; se não houver forma de simular rede,
            reportar impossibilidade e prosseguir com o máximo de
            verificações locais.
116         frame = context.pages[-1]
117         # Click element
118         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/div[2]/div[2]/div').nth(0)
119         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
120
```

```
            await asyncio.sleep(5)

        finally:
            if context:
                await context.close()
            if browser:
                await browser.close()
            if pw:
                await pw.stop()

    asyncio.run(run_test())

```

Error

TimeoutError: Locator.click: Timeout 5000ms exceeded. Call log: - waiting for locator("xpath=html/body/div[2]/div[2]/div[2]/div/form/div[2]/div/button").first

Cause

The button locator used in the test might not be loading in time due to slow server response or the button is hidden behind a loading spinner or overlay.

Fix

Optimize the server response time by improving backend performance, or implement a wait mechanism in the test to allow for dynamic content to load fully before attempting to click the button.

TC017-Order notifications via Telegram integration

ATTRIBUTES

Status	Failed
Priority	MEDIUM
Description	Verify that order notifications are sent correctly via Telegram with correct formatting and content.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/5478f4b8-00c1-70cc-8db6-48f4cfc182b1/1770755491281355/result.webm

Test Code

```python
import asyncio
from playwright import async_api

async def run_test():
    pw = None
    browser = None
    context = None

    try:
        # Start a Playwright session in asynchronous mode
        pw = await async_api.async_playwright().start()

        # Launch a Chromium browser in headless mode with custom arguments
        browser = await pw.chromium.launch(
            headless=True,
            args=[
                "--window-size=1280,720",         # Set the browser window size
                "--disable-dev-shm-usage",        # Avoid using /dev/shm which can cause issues in containers
                "--ipc=host",                     # Use host-level IPC for better stability
                "--single-process"                # Run the browser in a single process mode
            ],
        )

        # Create a new browser context (like an incognito window)
        context = await browser.new_context()
        context.set_default_timeout(5000)

        # Open a new page in the browser context
        page = await context.new_page()

        # Navigate to your target URL and wait until the network request is committed
        await page.goto("https://usecolaai.vercel.app", wait_until="commit", timeout=10000)

        # Wait for the main page to reach DOMContentLoaded state (optional for stability)
        try:
            await page.wait_for_load_state("domcontentloaded", timeout=3000)
        except async_api.Error:
            pass

        # Iterate through all iframes and wait for them to load as well
        for frame in page.frames:
            try:
                await frame.wait_for_load_state("domcontentloaded", timeout=3000)
            except async_api.Error:
                pass

        # Interact with the page elements to simulate user flow
```

```
48          # -> Navigate to https://usecolaai.vercel.app
49          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
50
51          # -> Preencher o e-mail e a senha com as credenciais de teste
            e clicar em Entrar para iniciar sessão (login).
52          frame = context.pages[-1]
53          # Input text
54          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[1]/div/input').nth(0)
55          await page.wait_for_timeout(3000); await elem.fill
            ('testefinal99@example.com')
56
57          frame = context.pages[-1]
58          # Input text
59          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/input').nth(0)
60          await page.wait_for_timeout(3000); await elem.fill
            ('test123456')
61
62          frame = context.pages[-1]
63          # Click element
64          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/button').nth(0)
65          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
66
67          # -> Clicar no botão 'Entrar' para submeter o formulário de
            login. Após login bem-sucedido, localizar a área
            administrativa ou de pedidos e acionar um aviso/encomenda
            para enviar a notificação via Telegram para validação do
            conteúdo e formatação.
68          frame = context.pages[-1]
69          # Click element
70          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/button').nth(0)
71          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
72
73          # -> Abrir a página de Pedidos no painel para criar/acionar
            um pedido que dispare a notificação no Telegram. Clicar em
            'Pedidos' na barra lateral.
74          frame = context.pages[-1]
75          # Click element
76          elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[5]').
            nth(0)
77          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
78
79          # -> Clicar em 'Criar Pedido' para abrir o formulário de
            criação de pedido e iniciar o fluxo que deve disparar a
            notificação no Telegram.
80          frame = context.pages[-1]
81          # Click element
82          elem = frame.locator('xpath=html/body/div[2]/main/div/div[4]/
            div[1]/div[2]/a/button').nth(0)
83          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
84
```

```
85          # -> Abrir o formulário 'Novo Pedido' no painel de Pedidos
            para criar um pedido que dispare a notificação no Telegram
            (clicar no botão '+ Novo Pedido').
86          frame = context.pages[-1]
87          # Click element
88          elem = frame.locator('xpath=html/body/div[2]/main/div/div[1]/
            a/button').nth(0)
89          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
90
91          # -> Abrir os detalhes do pedido (clicar no cartão do pedido
            #13) para localizar o botão/opção que dispara a notificação
            ao Telegram e, em seguida, acionar essa notificação.
92          frame = context.pages[-1]
93          # Click element
94          elem = frame.locator('xpath=html/body/div[2]/main/div/div[3]/
            div/div[1]/div[1]/div[2]/div/button').nth(0)
95          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
96
97          # -> Adicionar um produto ao carrinho (Hotdog Especial 1)
            para iniciar a criação do pedido que deve disparar a
            notificação ao Telegram.
98          frame = context.pages[-1]
99          # Click element
100         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/div[2]/div[1]/div').nth(0)
101         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
102
103         # -> Abrir o resumo do carrinho/check-out para prosseguir com
            o fechamento do pedido (escolher pagamento PIX e finalizar)
            de forma a disparar a notificação via Telegram.
104         frame = context.pages[-1]
105         # Click element
106         elem = frame.locator('xpath=html/body/div[2]/main/div/div[3]/
            div').nth(0)
107         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
108
109         # -> Preencher Nome e Telefone do cliente, selecionar
            pagamento por PIX e rolar a página para revelar o botão
            'Criar Pedido' para então criar o pedido que deve disparar a
            notificação no Telegram.
110         frame = context.pages[-1]
111         # Input text
112         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[2]/div/div[3]/div[1]/div/input').nth(0)
113         await page.wait_for_timeout(3000); await elem.fill('Cliente
            Teste')
114
115         frame = context.pages[-1]
116         # Input text
117         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[2]/div/div[3]/div[2]/div/input').nth(0)
118         await page.wait_for_timeout(3000); await elem.fill('(11)
            99999-0000')
119
120         frame = context.pages[-1]
```

```
        # Click element
        elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
        div[2]/div/div[5]/div/button[2]').nth(0)
        await page.wait_for_timeout(3000); await elem.click
        (timeout=5000)

        # -> Clicar no botão 'Criar Pedido' para finalizar o pedido
        (com PIX) e disparar a notificação ao Telegram para posterior
        verificação de entrega e formatação.
        frame = context.pages[-1]
        # Click element
        elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
        div[2]/div/button').nth(0)
        await page.wait_for_timeout(3000); await elem.click
        (timeout=5000)

        # -> Clicar no botão de finalização / 'Criar Pedido' na tela
        de Novo Pedido para realmente enviar o pedido e acionar a
        notificação do Telegram; aguardar alguns segundos e procurar
        confirmação na UI (toasts, redirecionamento para lista de
        pedidos ou novo elemento indicando sucesso).
        frame = context.pages[-1]
        # Click element
        elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
        div[2]/div/div[1]/div/div[2]/button').nth(0)
        await page.wait_for_timeout(3000); await elem.click
        (timeout=5000)

        # -> Abrir os detalhes do pedido #14 para localizar e acionar
        a opção que dispara a notificação ao Telegram (ou visualizar
        o histórico de notificações).
        frame = context.pages[-1]
        # Click element
        elem = frame.locator('xpath=html/body/div[2]/main/div/div[3]/
        div/div[1]/div[1]/div[2]/div/button').nth(0)
        await page.wait_for_timeout(3000); await elem.click
        (timeout=5000)

        # -> Abrir os detalhes do pedido #14 clicando em 'Ver
        Detalhes' (index 3011) para localizar a opção de notificação
        (Telegram) ou o histórico de notificações.
        frame = context.pages[-1]
        # Click element
        elem = frame.locator('xpath=html/body/div[2]/main/div/div[3]/
        div/div[1]/div[1]/div[2]/div/div/a[1]').nth(0)
        await page.wait_for_timeout(3000); await elem.click
        (timeout=5000)

        # -> Abrir Configurações no painel para localizar a seção de
        integrações/notificações (procurar configuração do Telegram)
        e, se disponível, acionar um envio de teste ou instruções
        para habilitar notificações Telegram.
        frame = context.pages[-1]
        # Click element
        elem = frame.locator('xpath=html/body/div[2]/aside/div[2]/a').
        nth(0)
        await page.wait_for_timeout(3000); await elem.click
        (timeout=5000)

```

```
155         # -> Abrir a página de Configurações (clicar no item
            'Configurações' na barra lateral) para localizar a seção de
            integrações/notificações (Telegram) e acionar um envio de
            teste ou instrução de envio de notificação.
156         frame = context.pages[-1]
157         # Click element
158         elem = frame.locator('xpath=html/body/div[2]/aside/div[2]/a').
            nth(0)
159         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
160
161         # -> Abrir a aba de 'Links de Acesso' nas Configurações para
            localizar a integração do Telegram (ou opção de enviar teste)
            e acionar um envio de teste/trigger, se disponível.
162         frame = context.pages[-1]
163         # Click element
164         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/button[8]').nth(0)
165         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
166
167         # -> Localizar a configuração/integracao do Telegram dentro
            das Configurações (procurar aba/sessão relevante) para
            acionar um envio de teste ou instrução para disparar
            notificações de pedidos.
168         frame = context.pages[-1]
169         # Click element
170         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/button[7]').nth(0)
171         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
172
173         # -> Abrir a aba 'Links de Acesso' nas Configurações para
            localizar a integração do Telegram e procurar opção de envio
            de teste/trigger de notificação.
174         frame = context.pages[-1]
175         # Click element
176         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/button[8]').nth(0)
177         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
178
179         # -> Abrir a aba 'Conta' nas Configurações e procurar a seção
            de integrações/notificações (Telegram). Se houver opção de
            envio de teste, acioná-la; caso contrário, localizar
            instruções/credenciais (token, chat id) para habilitar o
            envio de notificações.
180         frame = context.pages[-1]
181         # Click element
182         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/button[9]').nth(0)
183         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
184
185         # -> Abrir a aba 'Links de Acesso' nas Configurações para
            localizar a integração do Telegram e procurar opção de envio
            de teste/trigger (token, chat id) para poder acionar uma
            notificação de pedido.
186         frame = context.pages[-1]
```

```
            # Click element
            elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/button[8]').nth(0)
            await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)

            # -> Abrir a aba 'Conta' nas Configurações e procurar seção
            de integrações/Telegram (token, chat id, envio de teste) para
            acionar um envio de teste ou localizar instruções para
            habilitar notificações.
            frame = context.pages[-1]
            # Click element
            elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/button[9]').nth(0)
            await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)

            # -> Abrir a aba 'Geral' nas Configurações e procurar por
            referências/integração do Telegram (token, chat id ou botão
            de teste).
            frame = context.pages[-1]
            # Click element
            elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/button[1]').nth(0)
            await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)

            await asyncio.sleep(5)

        finally:
            if context:
                await context.close()
            if browser:
                await browser.close()
            if pw:
                await pw.stop()

    asyncio.run(run_test())

```

Error

TimeoutError: Locator.click: Timeout 5000ms exceeded. Call log: - waiting for locator("xpath=html/body/div[2]/div[2]/div[2]/div/form/div[2]/div/button").first

Cause

The button identified by the XPath might not be present in the DOM when the click action is attempted. This could be due to the page taking too long to load or the button being dynamically created after some delay.

Fix

Optimize your page loading times by minimizing assets and enabling faster responses from the server. Also, consider adding a wait mechanism or verifying the existence of the button before attempting to click it, using appropriate wait functions.

TC021-Real-time order tracking and status update verification

ATTRIBUTES

Status	Failed
Priority	MEDIUM
Description	Verify that order status updates are reflected instantly on both admin and customer views using Supabase Realtime.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/5478f4b8-00c1-70cc-8db6-48f4cfc182b1/1770755492922414/result.webm

Test Code

```python
import asyncio
from playwright import async_api

async def run_test():
    pw = None
    browser = None
    context = None

    try:
        # Start a Playwright session in asynchronous mode
        pw = await async_api.async_playwright().start()

        # Launch a Chromium browser in headless mode with custom arguments
        browser = await pw.chromium.launch(
            headless=True,
            args=[
                "--window-size=1280,720",         # Set the browser window size
                "--disable-dev-shm-usage",        # Avoid using /dev/shm which can cause issues in containers
                "--ipc=host",                     # Use host-level IPC for better stability
                "--single-process"                # Run the browser in a single process mode
            ],
        )

        # Create a new browser context (like an incognito window)
        context = await browser.new_context()
        context.set_default_timeout(5000)

        # Open a new page in the browser context
        page = await context.new_page()

        # Navigate to your target URL and wait until the network request is committed
        await page.goto("https://usecolaai.vercel.app", wait_until="commit", timeout=10000)

        # Wait for the main page to reach DOMContentLoaded state (optional for stability)
        try:
            await page.wait_for_load_state("domcontentloaded", timeout=3000)
        except async_api.Error:
            pass

        # Iterate through all iframes and wait for them to load as well
        for frame in page.frames:
            try:
                await frame.wait_for_load_state("domcontentloaded", timeout=3000)
            except async_api.Error:
                pass

        # Interact with the page elements to simulate user flow
```

```
48          # -> Navigate to https://usecolaai.vercel.app
49          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
50
51          # -> Efetuar login como cliente usando testefinal99@example.
            com / test123456 para então navegar até a área de pedidos e
            criar um pedido.
52          frame = context.pages[-1]
53          # Input text
54          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[1]/div/input').nth(0)
55          await page.wait_for_timeout(3000); await elem.fill
            ('testefinal99@example.com')
56
57          frame = context.pages[-1]
58          # Input text
59          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/input').nth(0)
60          await page.wait_for_timeout(3000); await elem.fill
            ('test123456')
61
62          frame = context.pages[-1]
63          # Click element
64          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/button').nth(0)
65          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
66
67          # -> Tentar efetuar login novamente clicando no botão
            'Entrar' (índice 154). Se o login for bem-sucedido, navegar
            para 'Controle de Pedidos' e criar um pedido como cliente.
68          frame = context.pages[-1]
69          # Click element
70          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/button').nth(0)
71          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
72
73          # -> Abrir a interface do cliente em uma nova aba (https://
            usecolaai.vercel.app) e iniciar o fluxo de criação de pedido
            como cliente para depois registrar o pedido e prosseguir com
            os próximos passos do teste.
74          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
75
76          # -> Abrir uma nova aba com a interface do cliente (carregar
            https://usecolaai.vercel.app) para iniciar o fluxo de criação
            de pedido como cliente.
77          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
78
79          # -> Tentar recarregar a aplicação (forçar reload) para
            permitir que a SPA carregue e exibir elementos interativos.
            Se o reload falhar, tentar abrir nova aba separada para
            cliente/admin ou reportar erro de carregamento.
80          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
81
82          await asyncio.sleep(5)
```

```
83
84      finally:
85          if context:
86              await context.close()
87          if browser:
88              await browser.close()
89          if pw:
90              await pw.stop()
91
92  asyncio.run(run_test())
93
```

Error

TimeoutError: Locator.click: Timeout 5000ms exceeded. Call log: - waiting for locator("xpath=html/body/div[2]/div[2]/div[2]/div/form/div[2]/div/button").first

Cause

The webpage may not have fully loaded or the button may not be interactable due to overlay elements, animations, or dynamic content rendering that takes longer than expected.

Fix

Increase the timeout duration in the test script, or implement a wait for the button to be visible and clickable before attempting to click it. Additionally, ensure that the webpage is optimized for faster loading and interaction.

TC018-QR code generation for physical tables or counters

ATTRIBUTES

Status	Failed
Priority	MEDIUM
Description	Verify that QR codes can be generated and downloaded for tables/counters enabling easy customer menu access.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/5478f4b8-00c1-70cc-8db6-48f4cfc182b1/1770755494143917/result.webm

Test Code

```python
import asyncio
from playwright import async_api

async def run_test():
    pw = None
    browser = None
    context = None

    try:
        # Start a Playwright session in asynchronous mode
        pw = await async_api.async_playwright().start()

        # Launch a Chromium browser in headless mode with custom arguments
        browser = await pw.chromium.launch(
            headless=True,
            args=[
                "--window-size=1280,720",         # Set the browser window size
                "--disable-dev-shm-usage",        # Avoid using /dev/shm which can cause issues in containers
                "--ipc=host",                     # Use host-level IPC for better stability
                "--single-process"                # Run the browser in a single process mode
            ],
        )

        # Create a new browser context (like an incognito window)
        context = await browser.new_context()
        context.set_default_timeout(5000)

        # Open a new page in the browser context
        page = await context.new_page()

        # Navigate to your target URL and wait until the network request is committed
        await page.goto("https://usecolaai.vercel.app", wait_until="commit", timeout=10000)

        # Wait for the main page to reach DOMContentLoaded state (optional for stability)
        try:
            await page.wait_for_load_state("domcontentloaded", timeout=3000)
        except async_api.Error:
            pass

        # Iterate through all iframes and wait for them to load as well
        for frame in page.frames:
            try:
                await frame.wait_for_load_state("domcontentloaded", timeout=3000)
            except async_api.Error:
                pass

        # Interact with the page elements to simulate user flow
```

```
48          # -> Navigate to https://usecolaai.vercel.app
49          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
50
51          # -> Log in as admin using the provided credentials
            (testefinal99@example.com / test123456) by filling email and
            password and clicking Entrar.
52          frame = context.pages[-1]
53          # Input text
54          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[1]/div/input').nth(0)
55          await page.wait_for_timeout(3000); await elem.fill
            ('testefinal99@example.com')
56
57          frame = context.pages[-1]
58          # Input text
59          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/input').nth(0)
60          await page.wait_for_timeout(3000); await elem.fill
            ('test123456')
61
62          frame = context.pages[-1]
63          # Click element
64          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/button').nth(0)
65          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
66
67          # -> Clicar em 'Entrar' para submeter o formulário, aguardar
            o carregamento do painel e localizar a seção de geração de QR
            codes (rotulada como 'QR', 'Mesas', 'Balcões', 'Gerar QR' ou
            similar).
68          frame = context.pages[-1]
69          # Click element
70          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/button').nth(0)
71          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
72
73          # -> Abrir 'Configurações' e localizar a opção de geração de
            QR (procure por 'Mesas', 'Balcões', 'Gerar QR' ou similar).
74          frame = context.pages[-1]
75          # Click element
76          elem = frame.locator('xpath=html/body/div[2]/aside/div[2]/a').
            nth(0)
77          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
78
79          # -> Click the 'Configurações' link (index 561) to open
            settings, then locate the QR code generation area (look for
            'Mesas', 'Balcões', 'Gerar QR', or similar).
80          frame = context.pages[-1]
81          # Click element
82          elem = frame.locator('xpath=html/body/div[2]/aside/div[2]/a').
            nth(0)
83          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
84
85          # -> Clicar em 'Cardápio Online' (índice 1230) para localizar
```

```
            a área de geração de QR (procure por Mesas, Balcões, Gerar QR
            ou baixar/exportar).
 86         frame = context.pages[-1]
 87         # Click element
 88         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/button[3]').nth(0)
 89         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
 90
 91         # -> Abrir 'Links de Acesso' (ou seção equivalente) para
            localizar UI que permite criar/baixar QR codes por mesa/
            balcão e identificar controles para gerar múltiplos QR (mesas/
            balcões).
 92         frame = context.pages[-1]
 93         # Click element
 94         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/button[8]').nth(0)
 95         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
 96
 97         # -> Clicar em 'Cardápio Online' (índice 1230) para abrir a
            interface de geração/baixa de QR codes e localizar controles
            como 'Baixar QR Code' e campos de personalização.
 98         frame = context.pages[-1]
 99         # Click element
100         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/button[3]').nth(0)
101         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
102
103         # -> Clicar em 'Baixar QR Code' para iniciar o download do QR
            oficial da loja e, em seguida, abrir 'Links de Acesso' para
            localizar controles de geração/baixa de QR por mesa/balcão.
104         frame = context.pages[-1]
105         # Click element
106         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[2]/div/div[2]/div[2]/div[2]/div[2]/div/div[2]/button').nth
            (0)
107         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
108
109         frame = context.pages[-1]
110         # Click element
111         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/button[8]').nth(0)
112         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
113
114         # -> Abrir aba 'Cardápio Online' para localizar a interface
            de geração/baixa de QR codes para mesas/balcões e identificar
            controles de geração e download (procure por 'Baixar QR
            Code', inputs de personalização, lista de mesas/balcões).
115         frame = context.pages[-1]
116         # Click element
117         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/button[3]').nth(0)
118         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
119
```

```
            # -> Abrir a aba 'Links de Acesso' para localizar controles
            que permitam gerar/baixar QR por mesas/balcões (procurar por
            'Mesa', 'Balcão', 'Gerar', 'Baixar QR' ou opções de
            exportação).
            frame = context.pages[-1]
            # Click element
            elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/button[8]').nth(0)
            await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)

            # -> Abrir a aba 'Cardápio Online' para localizar a interface
            de geração/baixa de QR codes por mesas/balcões (procurar por
            'Gerar', 'Baixar QR Code', lista de mesas/balcões) e, em
            seguida, iniciar geração/baixa de QR.
            frame = context.pages[-1]
            # Click element
            elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/button[3]').nth(0)
            await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)

            # -> Clicar em 'Baixar QR Code' (botão [3772]) para iniciar o
            download do QR oficial da loja e, depois de verificar o
            resultado, procurar controles para gerar QR por 'Mesas'/
            'Balcões' (procurar textos/elements com 'Mesa', 'Mesas',
            'Balcão', 'Balcões' ou opções de exportação).
            frame = context.pages[-1]
            # Click element
            elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[2]/div/div[2]/div[2]/div[2]/div[2]/div/div[2]/button').nth
            (0)
            await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)

            await asyncio.sleep(5)

        finally:
            if context:
                await context.close()
            if browser:
                await browser.close()
            if pw:
                await pw.stop()

    asyncio.run(run_test())

```

Error

TimeoutError: Locator.click: Timeout 5000ms exceeded. Call log: - waiting for locator("xpath=html/body/div[2]/div[2]/div[2]/div/form/div[2]/div/button").first

Cause

The button is not present in the DOM when the test attempts to click it, possibly due to slow loading times, a JavaScript error preventing the button from rendering, or incorrect XPath leading to an element that is not visible or not interactable.

Fix

Optimize the loading time of the webpage, ensure that all JavaScript is functioning correctly, verify the XPath for accuracy, and consider increasing the timeout duration to allow for slower loads.

TC012-PIX payment processing with AbacatePay integration

ATTRIBUTES

Status	Failed
Priority	MEDIUM
Description	Verify that payments via PIX through AbacatePay complete successfully with QR code generation and display.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/5478f4b8-00c1-70cc-8db6-48f4cfc182b1/1770755493953653/result.webm

Test Code

```python
import asyncio
from playwright import async_api

async def run_test():
    pw = None
    browser = None
    context = None

    try:
        # Start a Playwright session in asynchronous mode
        pw = await async_api.async_playwright().start()

        # Launch a Chromium browser in headless mode with custom arguments
        browser = await pw.chromium.launch(
            headless=True,
            args=[
                "--window-size=1280,720",         # Set the browser window size
                "--disable-dev-shm-usage",        # Avoid using /dev/shm which can cause issues in containers
                "--ipc=host",                     # Use host-level IPC for better stability
                "--single-process"                # Run the browser in a single process mode
            ],
        )

        # Create a new browser context (like an incognito window)
        context = await browser.new_context()
        context.set_default_timeout(5000)

        # Open a new page in the browser context
        page = await context.new_page()

        # Navigate to your target URL and wait until the network request is committed
        await page.goto("https://usecolaai.vercel.app", wait_until="commit", timeout=10000)

        # Wait for the main page to reach DOMContentLoaded state (optional for stability)
        try:
            await page.wait_for_load_state("domcontentloaded", timeout=3000)
        except async_api.Error:
            pass

        # Iterate through all iframes and wait for them to load as well
        for frame in page.frames:
            try:
                await frame.wait_for_load_state("domcontentloaded", timeout=3000)
            except async_api.Error:
                pass

        # Interact with the page elements to simulate user flow
```

```
48          # -> Navigate to https://usecolaai.vercel.app
49          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
50
51          # -> Preencher o formulário de login com as credenciais de
            teste e submeter (entrar) para acessar o painel e continuar o
            fluxo de checkout.
52          frame = context.pages[-1]
53          # Input text
54          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[1]/div/input').nth(0)
55          await page.wait_for_timeout(3000); await elem.fill
            ('testefinal99@example.com')
56
57          frame = context.pages[-1]
58          # Input text
59          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/input').nth(0)
60          await page.wait_for_timeout(3000); await elem.fill
            ('test123456')
61
62          frame = context.pages[-1]
63          # Click element
64          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/button').nth(0)
65          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
66
67          # -> Submeter o formulário de login clicando no botão
            'Entrar' para acessar o painel e prosseguir com o fluxo de
            checkout (selecionar PIX).
68          frame = context.pages[-1]
69          # Click element
70          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/button').nth(0)
71          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
72
73          # -> Abrir o fluxo de criação de pedido clicando em 'Novo
            Pedido' para avançar ao checkout e depois selecionar a forma
            de pagamento PIX.
74          frame = context.pages[-1]
75          # Click element
76          elem = frame.locator('xpath=html/body/div[2]/main/div/div[1]/
            a/button').nth(0)
77          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
78
79          # -> Abrir o fluxo de criação de pedido clicando em 'Novo
            Pedido' para iniciar o checkout (tentar abrir o formulário de
            pedido).
80          frame = context.pages[-1]
81          # Click element
82          elem = frame.locator('xpath=html/body/div[2]/main/div/div[1]/
            a/button').nth(0)
83          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
84
85          # -> Adicionar um produto ao carrinho — clicar em 'Hotdog
```

```
            Especial 1' para iniciar o checkout (índice 1607).
 86         frame = context.pages[-1]
 87         # Click element
 88         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/div[2]/div[1]/div').nth(0)
 89         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
 90
 91         # -> Tentar adicionar o produto ao carrinho novamente
            clicando no elemento do produto (Hotdog Especial 1) para
            abrir o fluxo de checkout.
 92         frame = context.pages[-1]
 93         # Click element
 94         elem = frame.locator('xpath=html/body/div[2]/main/div/div[3]/
            div').nth(0)
 95         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
 96
 97         # -> Selecionar a forma de pagamento 'PIX' no checkout
            clicando no botão PIX para iniciar o fluxo de geração de QR
            code (usar o botão index 1769).
 98         frame = context.pages[-1]
 99         # Click element
100         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[2]/div/div[5]/div/button[2]').nth(0)
101         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
102
103         # -> Preencher Nome e Telefone do cliente no carrinho, rolar
            a página para revelar o botão 'Criar Pedido' e então
            prosseguir para gerar o QR PIX via AbacatePay (clicar 'Criar
            Pedido' quando estiver visível).
104         frame = context.pages[-1]
105         # Input text
106         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[2]/div/div[3]/div[1]/div/input').nth(0)
107         await page.wait_for_timeout(3000); await elem.fill('Cliente
            Teste PIX')
108
109         frame = context.pages[-1]
110         # Input text
111         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[2]/div/div[3]/div[2]/div/input').nth(0)
112         await page.wait_for_timeout(3000); await elem.fill('(11)
            91234-5678')
113
114         # -> Criar o pedido clicando no botão 'Criar Pedido' para
            acionar a geração do QR PIX via AbacatePay (aguardar o
            retorno/modal de pagamento).
115         frame = context.pages[-1]
116         # Click element
117         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[2]/div/button').nth(0)
118         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
119
120         # -> Abrir os detalhes do pedido 'Cliente Teste PIX' na lista
            de pedidos para localizar e acionar a geração do QR PIX via
            AbacatePay.
```

```
            frame = context.pages[-1]
            # Click element
            elem = frame.locator('xpath=html/body/div[2]/main/div/div[3]/
            div/div[1]/div[1]/div[2]/div/button').nth(0)
            await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)

            # -> Abrir os detalhes do pedido 'Cliente Teste PIX' clicando
            em 'Ver Detalhes' (índice 2844) para localizar o fluxo/
            controle que gera o QR PIX via AbacatePay.
            frame = context.pages[-1]
            # Click element
            elem = frame.locator('xpath=html/body/div[2]/main/div/div[3]/
            div/div[1]/div[1]/div[2]/div/div/a[1]').nth(0)
            await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)

            # -> Abrir os detalhes do pedido #15 e extrair o QR code
            (imagem src), link de pagamento/ID e qualquer referência a
            AbacatePay ou status de pagamento.
            frame = context.pages[-1]
            # Click element
            elem = frame.locator('xpath=html/body/div[2]/main/div/div[3]/
            div/div[1]/div[1]/div[2]/div/div/a[1]').nth(0)
            await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)

            # -> Extrair o código PIX/QR (texto e src da imagem), IDs/
            URLs de pagamento e status atual; em seguida clicar
            'Confirmar Pagamento' para simular pagamento, aguardar e
            re-extrair para verificar se o status mudou para recebido/
            pago.
            frame = context.pages[-1]
            # Click element
            elem = frame.locator('xpath=html/body/div[2]/main/div/div[3]/
            div[7]/div/button[1]').nth(0)
            await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)

            await asyncio.sleep(5)

        finally:
            if context:
                await context.close()
            if browser:
                await browser.close()
            if pw:
                await pw.stop()

    asyncio.run(run_test())

```

Error

TimeoutError: Locator.click: Timeout 5000ms exceeded. Call log: - waiting for locator("xpath=html/body/div[2]/div[2]/div[2]/div/form/div[2]/div/button").first

Cause

The button identified by the locator may not be rendered on the page within the specified timeout, possibly due to slow loading times or a JavaScript error preventing the element from becoming interactable.

Fix

Optimize the webpage loading time by minimizing JavaScript execution and reducing resource sizes; check for any JavaScript errors in the console that may prevent the UI from rendering properly; ensure that the button is in a visible state before attempting the click command.

TC010-Customer digital menu browsing and ordering with upsell suggestions

ATTRIBUTES

Status	Failed
Priority	MEDIUM
Description	Verify customers can browse the product catalog, add products to basket, see upsell suggestions, and proceed to checkout.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/5478f4b8-00c1-70cc-8db6-48f4cfc182b1/1770755502057183/result.webm

Test Code

```python
import asyncio
from playwright import async_api

async def run_test():
    pw = None
    browser = None
    context = None

    try:
        # Start a Playwright session in asynchronous mode
        pw = await async_api.async_playwright().start()

        # Launch a Chromium browser in headless mode with custom arguments
        browser = await pw.chromium.launch(
            headless=True,
            args=[
                "--window-size=1280,720",         # Set the browser window size
                "--disable-dev-shm-usage",        # Avoid using /dev/shm which can cause issues in containers
                "--ipc=host",                     # Use host-level IPC for better stability
                "--single-process"                # Run the browser in a single process mode
            ],
        )

        # Create a new browser context (like an incognito window)
        context = await browser.new_context()
        context.set_default_timeout(5000)

        # Open a new page in the browser context
        page = await context.new_page()

        # Navigate to your target URL and wait until the network request is committed
        await page.goto("https://usecolaai.vercel.app", wait_until="commit", timeout=10000)

        # Wait for the main page to reach DOMContentLoaded state (optional for stability)
        try:
            await page.wait_for_load_state("domcontentloaded", timeout=3000)
        except async_api.Error:
            pass

        # Iterate through all iframes and wait for them to load as well
        for frame in page.frames:
            try:
                await frame.wait_for_load_state("domcontentloaded", timeout=3000)
            except async_api.Error:
                pass

        # Interact with the page elements to simulate user flow
```

```
48          # -> Navigate to https://usecolaai.vercel.app
49          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
50
51          # -> Preencher o formulário de login com as credenciais de
            teste e clicar em Entrar para acessar o painel e procurar
            link/rota para o menu público (ou navegação que permita abrir
            o menu do cliente).
52          frame = context.pages[-1]
53          # Input text
54          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[1]/div/input').nth(0)
55          await page.wait_for_timeout(3000); await elem.fill
            ('testefinal99@example.com')
56
57          frame = context.pages[-1]
58          # Input text
59          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/input').nth(0)
60          await page.wait_for_timeout(3000); await elem.fill
            ('test123456')
61
62          frame = context.pages[-1]
63          # Click element
64          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/button').nth(0)
65          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
66
67          # -> Tentar submeter o formulário de login novamente clicando
            no botão Entrar (index 147) e aguardar 3 segundos para
            observar se há navegação. Se não houver mudança após a
            segunda tentativa, usar estratégia alternativa (abrir nova
            aba para localizar o menu público ou usar go_to_url).
68          frame = context.pages[-1]
69          # Click element
70          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/button').nth(0)
71          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
72
73          # -> Abrir a interface de Novo Pedido (clicar em 'Novo
            Pedido') para localizar e acessar o menu público/fluxo de
            pedido do cliente e continuar com navegação por categorias.
74          frame = context.pages[-1]
75          # Click element
76          elem = frame.locator('xpath=html/body/div[2]/main/div/div[1]/
            a/button').nth(0)
77          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
78
79          # -> Tentar abrir o fluxo de 'Novo Pedido' do cliente
            clicando no botão 'Novo Pedido' (index 541) e aguardar para
            validar se o menu público/fluxo do pedido abre.
80          frame = context.pages[-1]
81          # Click element
82          elem = frame.locator('xpath=html/body/div[2]/main/div/div[1]/
            a/button').nth(0)
83          await page.wait_for_timeout(3000); await elem.click
```

```
        (timeout=5000)
84
85          # -> Recarregar a rota /pedidos/novo para forçar o
            carregamento da SPA. Se o recarregamento não resolver, abrir
            nova aba ou tentar outra URL pública do menu.
86          await page.goto("https://usecolaai.vercel.app/pedidos/novo",
            wait_until="commit", timeout=10000)
87
88          # -> Abrir uma nova aba apontando para uma rota do menu
            público (tentar /menu) para forçar carregamento do SPA
            público e localizar categorias/produtos.
89          await page.goto("https://usecolaai.vercel.app/menu",
            wait_until="commit", timeout=10000)
90
91          # -> Navegar de volta ao painel (/dashboard) para localizar o
            fluxo 'Novo Pedido' ou outro link/rota que exponha o menu
            público; em seguida usar o botão/rota correta para abrir o
            menu do cliente e iniciar os testes de catálogo.
92          await page.goto("https://usecolaai.vercel.app/dashboard",
            wait_until="commit", timeout=10000)
93
94          # -> Abrir o fluxo de 'Novo Pedido' clicando no botão
            correspondente para acessar o menu público/fluxo de pedido do
            cliente e então aguardar a resposta da SPA.
95          frame = context.pages[-1]
96          # Click element
97          elem = frame.locator('xpath=html/body/div[2]/main/div/div[1]/
            a/button').nth(0)
98          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
99
100         # -> Navegar pelas categorias 'Porções' e 'Bebidas' e
            adicionar pelo menos um produto ao carrinho para validar
            comportamento do carrinho e possíveis sugestões de upsell.
101         frame = context.pages[-1]
102         # Click element
103         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/div[1]/button[2]').nth(0)
104         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
105
106         frame = context.pages[-1]
107         # Click element
108         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/div[1]/button[3]').nth(0)
109         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
110
111         frame = context.pages[-1]
112         # Click element
113         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/div[2]/div[1]/div').nth(0)
114         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
115
116         # -> Adicionar 3 produtos ao carrinho para validar
            atualizações: Guaraná 350ml (index 3261), Suco de Laranja
            500ml (index 3263) e Coca-Cola 350ml (index 3270).
117         frame = context.pages[-1]
```

```
        # Click element
        elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
        div[1]/div[2]/div[1]').nth(0)
        await page.wait_for_timeout(3000); await elem.click
        (timeout=5000)

        frame = context.pages[-1]
        # Click element
        elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
        div[1]/div[2]/div[3]').nth(0)
        await page.wait_for_timeout(3000); await elem.click
        (timeout=5000)

        frame = context.pages[-1]
        # Click element
        elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
        div[1]/div[2]/div[10]').nth(0)
        await page.wait_for_timeout(3000); await elem.click
        (timeout=5000)

        # -> Abrir detalhe de um produto já adicionado (Suco de
        Laranja 500ml) para procurar sugestões de upsell/adicionais e
        verificar se podem ser adicionadas ao carrinho.
        frame = context.pages[-1]
        # Click element
        elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
        div[1]/div[2]/div[3]').nth(0)
        await page.wait_for_timeout(3000); await elem.click
        (timeout=5000)

        # -> Abrir o detalhe do produto 'Suco de Laranja 500ml'
        (clicar no cartão do produto) e extrair a lista de sugestões
        de upsell/adicionais (nome, preço se exibido, existência de
        botão para adicionar ao carrinho e índices dos elementos
        quando visíveis). Se o modal abrir, extrair também campos de
        escolha (quantidade, obrigatórios/opcionais).
        frame = context.pages[-1]
        # Click element
        elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
        div[1]/div[2]/div[3]').nth(0)
        await page.wait_for_timeout(3000); await elem.click
        (timeout=5000)

        # -> Preencher Nome e Telefone do cliente, alternar para
        Entrega para observar se aparece taxa de entrega e
        atualização do total, selecionar PIX como forma de pagamento
        e localizar o botão 'Criar Pedido' para submeter o pedido.
        frame = context.pages[-1]
        # Input text
        elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
        div[2]/div/div[3]/div[1]/div/input').nth(0)
        await page.wait_for_timeout(3000); await elem.fill('Cliente
        Teste')

        frame = context.pages[-1]
        # Input text
        elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
        div[2]/div/div[3]/div[2]/div/input').nth(0)
        await page.wait_for_timeout(3000); await elem.fill('(11)
```

```
            99999-0000')
154
155             frame = context.pages[-1]
156             # Click element
157             elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
                div[2]/div/div[3]/div[3]/button[2]').nth(0)
158             await page.wait_for_timeout(3000); await elem.click
                (timeout=5000)
159
160             await asyncio.sleep(5)
161
162         finally:
163             if context:
164                 await context.close()
165             if browser:
166                 await browser.close()
167             if pw:
168                 await pw.stop()
169
170     asyncio.run(run_test())
171
```

Error

TimeoutError: Locator.click: Timeout 5000ms exceeded. Call log: - waiting for locator("xpath=html/body/div[2]/div[2]/div[2]/div/form/div[2]/div/button").first

Cause

The button at the specified XPath locator may not be present in the DOM, not rendered properly due to a loading delay, or it could be obscured by another element, preventing interaction.

Fix

Check if the button exists and is visible in the DOM. If it's dynamically loaded, implement a wait to allow sufficient time for the button to render. Ensure there are no CSS or z-index issues obscuring the button and optimize loading times by minimizing heavy resources.

TC020-Security validation of Row Level Security (RLS) in Supabase

ATTRIBUTES

Status	Failed
Priority	MEDIUM
Description	Verify that Supabase RLS policies correctly restrict data access to authorized users only.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/5478f4b8-00c1-70cc-8db6-48f4cfc182b1/1770755525258475/result.webm

Test Code

```python
import asyncio
from playwright import async_api

async def run_test():
    pw = None
    browser = None
    context = None

    try:
        # Start a Playwright session in asynchronous mode
        pw = await async_api.async_playwright().start()

        # Launch a Chromium browser in headless mode with custom arguments
        browser = await pw.chromium.launch(
            headless=True,
            args=[
                "--window-size=1280,720",         # Set the browser window size
                "--disable-dev-shm-usage",        # Avoid using /dev/shm which can cause issues in containers
                "--ipc=host",                     # Use host-level IPC for better stability
                "--single-process"                # Run the browser in a single process mode
            ],
        )

        # Create a new browser context (like an incognito window)
        context = await browser.new_context()
        context.set_default_timeout(5000)

        # Open a new page in the browser context
        page = await context.new_page()

        # Navigate to your target URL and wait until the network request is committed
        await page.goto("https://usecolaai.vercel.app", wait_until="commit", timeout=10000)

        # Wait for the main page to reach DOMContentLoaded state (optional for stability)
        try:
            await page.wait_for_load_state("domcontentloaded", timeout=3000)
        except async_api.Error:
            pass

        # Iterate through all iframes and wait for them to load as well
        for frame in page.frames:
            try:
                await frame.wait_for_load_state("domcontentloaded", timeout=3000)
            except async_api.Error:
                pass

        # Interact with the page elements to simulate user flow
```

```
48          # -> Navigate to https://usecolaai.vercel.app
49          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
50
51          # -> Log in with provided credentials (testefinal99@example.
            com / test123456) by filling the E-mail and Senha fields and
            clicking 'Entrar'.
52          frame = context.pages[-1]
53          # Input text
54          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[1]/div/input').nth(0)
55          await page.wait_for_timeout(3000); await elem.fill
            ('testefinal99@example.com')
56
57          frame = context.pages[-1]
58          # Input text
59          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/input').nth(0)
60          await page.wait_for_timeout(3000); await elem.fill
            ('test123456')
61
62          frame = context.pages[-1]
63          # Click element
64          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/button').nth(0)
65          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
66
67          # -> Abrir a seção 'Funcionários' na barra lateral para
            procurar listagem/perfis e tentar acessar dados que possam
            pertencer a outro usuário (elemento da sidebar
            'Funcionários'). Se a lista carregar, procurar um funcionário
            diferente do usuário logado e abrir seu perfil/detalhes para
            verificar se o sistema permite visualização/edição indevida.
68          frame = context.pages[-1]
69          # Click element
70          elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[10]
            ').nth(0)
71          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
72
73          # -> Abrir a seção 'Funcionários' na sidebar para carregar a
            lista de funcionários e localizar um registro que não seja o
            do usuário logado (para tentar acessar dados de outro
            usuário). Em seguida, abrir o perfil desse funcionário para
            verificar se o acesso é bloqueado.
74          frame = context.pages[-1]
75          # Click element
76          elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[10]
            ').nth(0)
77          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
78
79          # -> Tentar abrir o perfil/detalhes de um funcionário
            diferente do usuário autenticado (tentar o botão 'Editar' do
            funcionário 'Cozinha') para verificar se o sistema permite
            visualizar/editar dados de outro usuário.
80          frame = context.pages[-1]
81          # Click element
```

```
82            elem = frame.locator('xpath=html/body/div[2]/main/div/div[3]/
              div/div[2]/div[4]/button[1]').nth(0)
83            await page.wait_for_timeout(3000); await elem.click
              (timeout=5000)
84
85            # -> Alterar o campo 'Nome Completo' para 'TestRLS_Cozinha'
              no modal do funcionário aberto e clicar em 'Salvar' para
              tentar modificar dados de outro usuário; observar se a
              operação é permitida ou bloqueada (mensagem de sucesso/erro).
86            frame = context.pages[-1]
87            # Input text
88            elem = frame.locator('xpath=html/body/div[2]/main/div/div[4]/
              div/div[1]/div/div/input').nth(0)
89            await page.wait_for_timeout(3000); await elem.fill
              ('TestRLS_Cozinha')
90
91            frame = context.pages[-1]
92            # Click element
93            elem = frame.locator('xpath=html/body/div[2]/main/div/div[4]/
              div/div[6]/button[2]').nth(0)
94            await page.wait_for_timeout(3000); await elem.click
              (timeout=5000)
95
96            # -> Fechar o modal (Cancelar) e procurar na lista de
              funcionários o registro com nome 'TestRLS_Cozinha' para
              verificar se a alteração foi aplicada (indicação de que RLS
              não impediu a modificação) ou se permanece inalterado
              (indicação de que RLS bloqueou/impediu alteração).
97            frame = context.pages[-1]
98            # Click element
99            elem = frame.locator('xpath=html/body/div[2]/main/div/div[4]/
              div/div[6]/button[1]').nth(0)
100           await page.wait_for_timeout(3000); await elem.click
              (timeout=5000)
101
102           # -> Abrir o modal de edição do funcionário 'TestRLS_Cozinha'
              para inspecionar campos (E-mail, ID, Função) e verificar se o
              registro pertence ao usuário autenticado ou a outro usuário.
              Procurar toasts/alerts ou mensagens de erro/sucesso e, se
              possível, tentar reverter a alteração para limpar o artefato
              de teste. Coletar evidências que confirmem se a alteração foi
              permitida pelo backend (indicando falha nas RLS) ou bloqueada.
103           frame = context.pages[-1]
104           # Click element
105           elem = frame.locator('xpath=html/body/div[2]/main/div/div[3]/
              div/div[2]/div[4]/button[1]').nth(0)
106           await page.wait_for_timeout(3000); await elem.click
              (timeout=5000)
107
108           await asyncio.sleep(5)
109
110       finally:
111           if context:
112               await context.close()
113           if browser:
114               await browser.close()
115           if pw:
116               await pw.stop()
117
```

```
118    asyncio.run(run_test())
119
```

Error

TimeoutError: Locator.click: Timeout 5000ms exceeded. Call log: - waiting for locator("xpath=html/body/div[2]/main/div/div[4]/div/div[6]/button[1]").first

Cause

The button element located at the specified XPath may not be rendered on the webpage within the timeout period, possibly due to slow loading times, JavaScript delays, or changes in the DOM structure.

Fix

Increase the timeout duration for the test or optimize the webpage loading speed by checking for any long-running scripts, optimizing assets, or ensuring that elements are available before the test tries to interact with them.

TC013-PIX payment processing with Stripe integration

ATTRIBUTES

Status	Failed
Priority	MEDIUM
Description	Verify PIX payment completion with Stripe API and correct QR code generation.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/5478f4b8-00c1-70cc-8db6-48f4cfc182b1/1770755493694825/result.webm

Test Code

```python
import asyncio
from playwright import async_api

async def run_test():
    pw = None
    browser = None
    context = None

    try:
        # Start a Playwright session in asynchronous mode
        pw = await async_api.async_playwright().start()

        # Launch a Chromium browser in headless mode with custom arguments
        browser = await pw.chromium.launch(
            headless=True,
            args=[
                "--window-size=1280,720",         # Set the browser window size
                "--disable-dev-shm-usage",        # Avoid using /dev/shm which can cause issues in containers
                "--ipc=host",                     # Use host-level IPC for better stability
                "--single-process"                # Run the browser in a single process mode
            ],
        )

        # Create a new browser context (like an incognito window)
        context = await browser.new_context()
        context.set_default_timeout(5000)

        # Open a new page in the browser context
        page = await context.new_page()

        # Navigate to your target URL and wait until the network request is committed
        await page.goto("https://usecolaai.vercel.app", wait_until="commit", timeout=10000)

        # Wait for the main page to reach DOMContentLoaded state (optional for stability)
        try:
            await page.wait_for_load_state("domcontentloaded", timeout=3000)
        except async_api.Error:
            pass

        # Iterate through all iframes and wait for them to load as well
        for frame in page.frames:
            try:
                await frame.wait_for_load_state("domcontentloaded", timeout=3000)
            except async_api.Error:
                pass

        # Interact with the page elements to simulate user flow
```

```
48          # -> Navigate to https://usecolaai.vercel.app
49          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
50
51          # -> Fill the login form with the provided test credentials
            and click 'Entrar' to sign in so the checkout/payment UI can
            be accessed.
52          frame = context.pages[-1]
53          # Input text
54          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[1]/div/input').nth(0)
55          await page.wait_for_timeout(3000); await elem.fill
            ('testefinal99@example.com')
56
57          frame = context.pages[-1]
58          # Input text
59          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/input').nth(0)
60          await page.wait_for_timeout(3000); await elem.fill
            ('test123456')
61
62          frame = context.pages[-1]
63          # Click element
64          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/button').nth(0)
65          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
66
67          # -> Clicar no botão 'Entrar' (index 131) para submeter o
            login e aguardar a navegação para o dashboard / checkout.
68          frame = context.pages[-1]
69          # Click element
70          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/button').nth(0)
71          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
72
73          # -> Click 'Novo Pedido' (index 534) to start creating a new
            order and proceed to checkout/payment options (to select PIX
            via Stripe).
74          frame = context.pages[-1]
75          # Click element
76          elem = frame.locator('xpath=html/body/div[2]/main/div/div[1]/
            a/button').nth(0)
77          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
78
79          # -> Open the 'Novo Pedido' / create-order UI so the checkout/
            payment options can be selected (choose PIX via Stripe).
            Click the 'Novo Pedido' page element that triggers order
            creation.
80          frame = context.pages[-1]
81          # Click element
82          elem = frame.locator('xpath=html/body/div[2]/main/div/div[1]/
            a/button').nth(0)
83          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
84
85          # -> Adicionar um produto ao carrinho clicando em 'Hotdog
```

```
            Especial 1' para iniciar o fluxo de checkout/pagamento
            (pré-requisito para selecionar PIX via Stripe).
 86         frame = context.pages[-1]
 87         # Click element
 88         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/div[2]/div[1]/div/span[1]').nth(0)
 89         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
 90
 91         # -> Abrir o carrinho/checkout para confirmar o item e
            selecionar a forma de pagamento (PIX via Stripe).
 92         frame = context.pages[-1]
 93         # Click element
 94         elem = frame.locator('xpath=html/body/div[2]/main/div/div[3]/
            div').nth(0)
 95         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
 96
 97         # -> Selecionar PIX como forma de pagamento no checkout
            (clicar no botão PIX - índice 1803). Após isso, aguardar e
            então verificar o QR code gerado.
 98         frame = context.pages[-1]
 99         # Click element
100         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[2]/div/div[5]/div/button[2]').nth(0)
101         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
102
103         # -> Clicar em 'Criar Pedido' para gerar a cobrança PIX
            (espera que o sistema solicite ao backend/Stripe e exiba o QR
            code). Após o clique, extrair/verificar o QR visível e, em
            seguida, simular notificação de pagamento via Stripe API.
104         frame = context.pages[-1]
105         # Click element
106         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[2]/div/div[1]/div/div[2]/button').nth(0)
107         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
108
109         # -> Adicionar 'Hotdog Especial 1' ao carrinho clicando no
            elemento do produto para então abrir o checkout e prosseguir
            com geração do PIX (QR).
110         frame = context.pages[-1]
111         # Click element
112         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/div[2]/div[1]/div/span[1]').nth(0)
113         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
114
115         # -> Localizar o botão 'Criar Pedido' / abrir o painel do
            carrinho para gerar a cobrança PIX (QR). Em seguida, proceder
            para gerar o QR e extrair/validar o QR code visível.
116         frame = context.pages[-1]
117         # Click element
118         elem = frame.locator('xpath=html/body/div[2]/main/div/div[3]/
            div').nth(0)
119         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
120
```

```
121             # -> Procurar no DOM se já existe um QR PIX gerado; se não
                houver, tentar prosseguir (marcar pagamento simulado via
                botão 'Pagamento Recebido') e verificar atualização do status
                do pedido.
122             frame = context.pages[-1]
123             # Click element
124             elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
                div[2]/div/div[5]/button').nth(0)
125             await page.wait_for_timeout(3000); await elem.click
                (timeout=5000)
126
127             await asyncio.sleep(5)
128
129         finally:
130             if context:
131                 await context.close()
132             if browser:
133                 await browser.close()
134             if pw:
135                 await pw.stop()
136
137     asyncio.run(run_test())
138
```

Error

TimeoutError: Locator.click: Timeout 5000ms exceeded. Call log: - waiting for locator("xpath=html/body/div[2]/div[2]/div[2]/div/form/div[2]/div/button").first

Cause

The button locator may not be visible or interactive within the specified timeout due to rendering issues, animations, or the element being overlayed by another element.

Fix

Increase the timeout duration or check for any JavaScript errors during the loading of the webpage that may prevent the button from becoming clickable. Additionally, ensure the element is not hidden or obscured by other UI components, possibly by adding appropriate wait or retry logic in the test script.

TC002-User registration with invalid email format

ATTRIBUTES

Status	Failed
Priority	MEDIUM
Description	Verify the system rejects registration with an improperly formatted email address.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/5478f4b8-00c1-70cc-8db6-48f4cfc182b1/1770755485021171/result.webm

Test Code

```python
import asyncio
from playwright import async_api

async def run_test():
    pw = None
    browser = None
    context = None

    try:
        # Start a Playwright session in asynchronous mode
        pw = await async_api.async_playwright().start()

        # Launch a Chromium browser in headless mode with custom arguments
        browser = await pw.chromium.launch(
            headless=True,
            args=[
                "--window-size=1280,720",         # Set the browser window size
                "--disable-dev-shm-usage",        # Avoid using /dev/shm which can cause issues in containers
                "--ipc=host",                     # Use host-level IPC for better stability
                "--single-process"                # Run the browser in a single process mode
            ],
        )

        # Create a new browser context (like an incognito window)
        context = await browser.new_context()
        context.set_default_timeout(5000)

        # Open a new page in the browser context
        page = await context.new_page()

        # Navigate to your target URL and wait until the network request is committed
        await page.goto("https://usecolaai.vercel.app", wait_until="commit", timeout=10000)

        # Wait for the main page to reach DOMContentLoaded state (optional for stability)
        try:
            await page.wait_for_load_state("domcontentloaded", timeout=3000)
        except async_api.Error:
            pass

        # Iterate through all iframes and wait for them to load as well
        for frame in page.frames:
            try:
                await frame.wait_for_load_state("domcontentloaded", timeout=3000)
            except async_api.Error:
                pass

        # Interact with the page elements to simulate user flow
```

```
48          # -> Navigate to https://usecolaai.vercel.app
49          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
50
51          # -> Clicar no link 'Crie gratuitamente' (elemento index 146)
            para abrir a página de registo e iniciar o teste de validação
            de e-mail.
52          frame = context.pages[-1]
53          # Click element
54          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/p/a').nth(0)
55          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
56
57          # -> Tentar novamente abrir a página de registo clicando no
            link 'Crie gratuitamente' (elemento índice 146).
58          frame = context.pages[-1]
59          # Click element
60          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/p/a').nth(0)
61          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
62
63          # -> Preencher o formulário usando um e-mail mal formatado
            ('useratexample.com'), submeter o formulário e extrair
            qualquer mensagem de erro visível relacionada ao e-mail.
64          frame = context.pages[-1]
65          # Input text
66          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[1]/
            div/form/div[1]/div/input').nth(0)
67          await page.wait_for_timeout(3000); await elem.fill('Test
            Shop')
68
69          frame = context.pages[-1]
70          # Input text
71          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[1]/
            div/form/div[2]/div/input').nth(0)
72          await page.wait_for_timeout(3000); await elem.fill
            ('useratexample.com')
73
74          frame = context.pages[-1]
75          # Input text
76          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[1]/
            div/form/div[3]/div/input').nth(0)
77          await page.wait_for_timeout(3000); await elem.fill
            ('test123456')
78
79          # -> Preencher 'Confirmar Senha' com 'test123456', submeter o
            formulário, aguardar resposta e extrair qualquer mensagem de
            erro relacionada ao e-mail (validação de formato) para
            verificar se o registo foi impedido.
80          frame = context.pages[-1]
81          # Input text
82          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[1]/
            div/form/div[4]/div/input').nth(0)
83          await page.wait_for_timeout(3000); await elem.fill
            ('test123456')
84
85          frame = context.pages[-1]
```

```
86            # Click element
87            elem = frame.locator('xpath=html/body/div[2]/div[2]/div[1]/
              div/form/button').nth(0)
88            await page.wait_for_timeout(3000); await elem.click
              (timeout=5000)
89
90            await asyncio.sleep(5)
91
92        finally:
93            if context:
94                await context.close()
95            if browser:
96                await browser.close()
97            if pw:
98                await pw.stop()
99
100   asyncio.run(run_test())
101
```

Error

TimeoutError: Locator.click: Timeout 5000ms exceeded. Call log: - waiting for locator("xpath=html/body/div[2]/div[2]/div[2]/div/p/a").first

Cause

The element located by the XPath may not be rendered on the page within the specified timeout, possibly due to slow loading times, JavaScript execution delays, or incorrect XPath targeting.

Fix

Increase the timeout duration or investigate the webpage's loading performance to ensure elements are appearing as expected. Additionally, verify the XPath for accuracy and consider using more robust selectors.

TC015-Offline data caching and synchronization for Admin Dashboard

ATTRIBUTES

Status	Failed
Priority	MEDIUM
Description	Verify that admin users can perform management tasks offline and data syncs properly when online.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/5478f4b8-00c1-70cc-8db6-48f4cfc182b1/1770755491422832/result.webm

Test Code

```python
import asyncio
from playwright import async_api

async def run_test():
    pw = None
    browser = None
    context = None

    try:
        # Start a Playwright session in asynchronous mode
        pw = await async_api.async_playwright().start()

        # Launch a Chromium browser in headless mode with custom arguments
        browser = await pw.chromium.launch(
            headless=True,
            args=[
                "--window-size=1280,720",         # Set the browser window size
                "--disable-dev-shm-usage",        # Avoid using /dev/shm which can cause issues in containers
                "--ipc=host",                     # Use host-level IPC for better stability
                "--single-process"                # Run the browser in a single process mode
            ],
        )

        # Create a new browser context (like an incognito window)
        context = await browser.new_context()
        context.set_default_timeout(5000)

        # Open a new page in the browser context
        page = await context.new_page()

        # Navigate to your target URL and wait until the network request is committed
        await page.goto("https://usecolaai.vercel.app", wait_until="commit", timeout=10000)

        # Wait for the main page to reach DOMContentLoaded state (optional for stability)
        try:
            await page.wait_for_load_state("domcontentloaded", timeout=3000)
        except async_api.Error:
            pass

        # Iterate through all iframes and wait for them to load as well
        for frame in page.frames:
            try:
                await frame.wait_for_load_state("domcontentloaded", timeout=3000)
            except async_api.Error:
                pass

        # Interact with the page elements to simulate user flow
```

```
48          # -> Navigate to https://usecolaai.vercel.app
49          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
50
51          # -> Fill the email and password fields with admin
            credentials and click 'Entrar' to sign in (attempt login).
52          frame = context.pages[-1]
53          # Input text
54          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[1]/div/input').nth(0)
55          await page.wait_for_timeout(3000); await elem.fill
            ('testefinal99@example.com')
56
57          frame = context.pages[-1]
58          # Input text
59          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/input').nth(0)
60          await page.wait_for_timeout(3000); await elem.fill
            ('test123456')
61
62          frame = context.pages[-1]
63          # Click element
64          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/button').nth(0)
65          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
66
67          # -> Submit the login form by clicking the 'Entrar' button
            (index 131) and wait for the dashboard to load.
68          frame = context.pages[-1]
69          # Click element
70          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/button').nth(0)
71          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
72
73          # -> Abrir a seção 'Produtos' para iniciar as operações de
            gestão do menu (primeira ação antes de alternar para modo
            offline).
74          frame = context.pages[-1]
75          # Click element
76          elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[2]').
            nth(0)
77          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
78
79          # -> Clicar no item 'Produtos' (index 831) para abrir a seção
            de gestão de produtos e preparar para alternar para modo
            offline e executar operações CRUD.
80          frame = context.pages[-1]
81          # Click element
82          elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[2]').
            nth(0)
83          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
84
85          # -> Abrir 'Configurações' para procurar opção de modo
            offline ou controle de rede dentro da aplicação (clicar
            elemento index 528). Se opção não existir, identificar
```

```
            alternativa para simular offline (ex.: controle no app, criar
            produto com rede desligada se possível).
 86         frame = context.pages[-1]
 87         # Click element
 88         elem = frame.locator('xpath=html/body/div[2]/aside/div[2]/a').
            nth(0)
 89         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
 90
 91         # -> Abrir a página de Configurações e localizar uma opção de
            modo offline/sincronização; se não houver, identificar
            alternativa para simular offline para executar CRUD
            localmente.
 92         frame = context.pages[-1]
 93         # Click element
 94         elem = frame.locator('xpath=html/body/div[2]/aside/div[2]/a').
            nth(0)
 95         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
 96
 97         # -> Abrir a aba 'Cardápio Online' nas Configurações para
            localizar um controle de modo offline/sincronização (ou opção
            alternativa para simular offline).
 98         frame = context.pages[-1]
 99         # Click element
100         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/button[3]').nth(0)
101         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
102
103         # -> Abrir a lista de Produtos para preparar e executar as
            operações CRUD (próximo clique em 'Produtos'). Se a opção de
            modo offline não estiver na aplicação, preparar para simular
            offline via ambiente de teste antes de criar/editar/excluir
            produtos e pedidos.
104         frame = context.pages[-1]
105         # Click element
106         elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[2]').
            nth(0)
107         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
108
109         # -> Abrir a lista de Produtos para preparar as operações
            CRUD e procurar controles na UI que permitam simular/offline;
            se não houver, preparar para executar CRUD e depois cortar
            rede no ambiente de teste.
110         frame = context.pages[-1]
111         # Click element
112         elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[2]').
            nth(0)
113         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
114
115         # -> Clicar em 'Novo Produto' para abrir o formulário de
            criação e preparar os dados que serão usados nos testes
            offline (se nenhum controle de modo offline for encontrado na
            aplicação, será necessário simular a desconexão de rede fora
            da aplicação antes de continuar).
116         frame = context.pages[-1]
```

```
117             # Click element
118             elem = frame.locator('xpath=html/body/div[2]/main/div/div[1]/
                button').nth(0)
119             await page.wait_for_timeout(3000); await elem.click
                (timeout=5000)
120
121             # -> Preencher o formulário de Novo Produto (Nome, Preço,
                Descrição) e clicar em 'Criar Produto' para adicionar um
                produto que será usado nos testes offline.
122             frame = context.pages[-1]
123             # Input text
124             elem = frame.locator('xpath=html/body/div[2]/main/div/div[6]/
                div/div[1]/div[2]/div/input').nth(0)
125             await page.wait_for_timeout(3000); await elem.fill('Produto
                Teste Offline')
126
127             frame = context.pages[-1]
128             # Input text
129             elem = frame.locator('xpath=html/body/div[2]/main/div/div[6]/
                div/div[1]/div[3]/div[2]/div/input').nth(0)
130             await page.wait_for_timeout(3000); await elem.fill('15')
131
132             frame = context.pages[-1]
133             # Input text
134             elem = frame.locator('xpath=html/body/div[2]/main/div/div[6]/
                div/div[1]/div[4]/textarea').nth(0)
135             await page.wait_for_timeout(3000); await elem.fill('Produto
                criado para testar operações em modo offline e sincronização
                posterior.')
136
137             # -> Clicar em 'Criar Produto' para adicionar o produto
                'Produto Teste Offline' antes de simular a desconexão e
                executar CRUD em modo offline.
138             frame = context.pages[-1]
139             # Click element
140             elem = frame.locator('xpath=html/body/div[2]/main/div/div[6]/
                div/div[2]/button[2]').nth(0)
141             await page.wait_for_timeout(3000); await elem.click
                (timeout=5000)
142
143             # -> Submeter o formulário 'Novo Produto' clicando no botão
                'Criar Produto' (index 7163) e aguardar a confirmação
                (fechamento do modal e aparecimento do produto na lista).
144             frame = context.pages[-1]
145             # Click element
146             elem = frame.locator('xpath=html/body/div[2]/main/div/div[6]/
                div/div[2]/button[2]').nth(0)
147             await page.wait_for_timeout(3000); await elem.click
                (timeout=5000)
148
149             # -> Fechar o modal 'Novo Produto' (clicar 'Cancelar' index
                7162) e verificar a lista de produtos para confirmar se
                'Produto Teste Offline' foi criado. Se não estiver na lista,
                preparar para simular desconexão de rede e continuar com os
                testes offline.
150             frame = context.pages[-1]
151             # Click element
152             elem = frame.locator('xpath=html/body/div[2]/main/div/div[6]/
                div/div[2]/button[1]').nth(0)
```

```
153             await page.wait_for_timeout(3000); await elem.click
                (timeout=5000)
154
155             # -> Fechar o modal 'Novo Produto' e procurar na lista o
                texto 'Produto Teste Offline' para confirmar se o produto
                existe (ou não). Se não for encontrado, preparar próximo
                passo para simular modo offline e prosseguir com CRUD.
156             frame = context.pages[-1]
157             # Click element
158             elem = frame.locator('xpath=html/body/div[2]/main/div/div[6]/
                div/div[2]/button[1]').nth(0)
159             await page.wait_for_timeout(3000); await elem.click
                (timeout=5000)
160
161             await asyncio.sleep(5)
162
163         finally:
164             if context:
165                 await context.close()
166             if browser:
167                 await browser.close()
168             if pw:
169                 await pw.stop()
170
171     asyncio.run(run_test())
172
```

Error

TimeoutError: Locator.click: Timeout 5000ms exceeded. Call log: - waiting for locator("xpath=html/body/div[2]/div[2]/div[2]/div/form/div[2]/div/button").first

Cause

The button element may not be present or visible in the DOM, possibly due to slow page loading, incorrect XPath selector, or dynamic content loading that hasn't completed.

Fix

Optimize the loading time of the page content, ensure the XPath selector correctly points to the button, or implement a wait mechanism that confirms the button's visibility before attempting to click.

TC007-Admin Dashboard financial data accuracy and realtime update

ATTRIBUTES

Status	Failed
Priority	MEDIUM
Description	Verify that financial data including daily and monthly sales, cash flow and client subscriptions are shown accurately and update in real time.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/5478f4b8-00c1-70cc-8db6-48f4cfc182b1/1770755497564842/result.webm

Test Code

```python
import asyncio
from playwright import async_api

async def run_test():
    pw = None
    browser = None
    context = None

    try:
        # Start a Playwright session in asynchronous mode
        pw = await async_api.async_playwright().start()

        # Launch a Chromium browser in headless mode with custom arguments
        browser = await pw.chromium.launch(
            headless=True,
            args=[
                "--window-size=1280,720",         # Set the browser window size
                "--disable-dev-shm-usage",        # Avoid using /dev/shm which can cause issues in containers
                "--ipc=host",                     # Use host-level IPC for better stability
                "--single-process"                # Run the browser in a single process mode
            ],
        )

        # Create a new browser context (like an incognito window)
        context = await browser.new_context()
        context.set_default_timeout(5000)

        # Open a new page in the browser context
        page = await context.new_page()

        # Navigate to your target URL and wait until the network request is committed
        await page.goto("https://usecolaai.vercel.app", wait_until="commit", timeout=10000)

        # Wait for the main page to reach DOMContentLoaded state (optional for stability)
        try:
            await page.wait_for_load_state("domcontentloaded", timeout=3000)
        except async_api.Error:
            pass

        # Iterate through all iframes and wait for them to load as well
        for frame in page.frames:
            try:
                await frame.wait_for_load_state("domcontentloaded", timeout=3000)
            except async_api.Error:
                pass

        # Interact with the page elements to simulate user flow
```

```
48          # -> Navigate to https://usecolaai.vercel.app
49          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
50
51          # -> Fill the login form with admin credentials and submit to
            sign in (then navigate to the finance section).
52          frame = context.pages[-1]
53          # Input text
54          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[1]/div/input').nth(0)
55          await page.wait_for_timeout(3000); await elem.fill
            ('testefinal99@example.com')
56
57          frame = context.pages[-1]
58          # Input text
59          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/input').nth(0)
60          await page.wait_for_timeout(3000); await elem.fill
            ('test123456')
61
62          frame = context.pages[-1]
63          # Click element
64          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/button').nth(0)
65          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
66
67          # -> Click the 'Entrar' button (index 14) to submit the login
            form. After successful login, navigate to the finance section.
68          frame = context.pages[-1]
69          # Click element
70          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/button').nth(0)
71          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
72
73          # -> Click the 'Fluxo de Caixa' menu item to open the cash
            flow view and inspect displayed financial figures.
74          frame = context.pages[-1]
75          # Click element
76          elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[16]
            ').nth(0)
77          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
78
79          # -> Click the 'Fluxo de Caixa' menu item to open the cash
            flow view and then extract the displayed financial figures.
80          frame = context.pages[-1]
81          # Click element
82          elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[16]
            ').nth(0)
83          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
84
85          # -> Extrair os valores financeiros exibidos na página 'Fluxo
            de Caixa' (Entradas, Receita de Pedidos, Saídas, Saldo do
            Período, Resumo Diário com data/valores, Top Entradas, Top
            Saídas, lista de Movimentações) e em seguida navegar para
            'Assinatura' para extrair dados de assinaturas.
```

```
86            frame = context.pages[-1]
87            # Click element
88            elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[11]
              ').nth(0)
89            await page.wait_for_timeout(3000); await elem.click
              (timeout=5000)
90
91            # -> Clicar em 'Assinatura' para abrir a página de
              assinaturas e extrair os dados de assinaturas (lista de
              clientes assinantes, planos, status, valores mensais e
              métricas agregadas).
92            frame = context.pages[-1]
93            # Click element
94            elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[11]
              ').nth(0)
95            await page.wait_for_timeout(3000); await elem.click
              (timeout=5000)
96
97            # -> Obter os dados do backend para Fluxo de Caixa e
              Assinatura (endpoints API) para comparar com os valores já
              extraídos do frontend. Abrir Fluxo de Caixa no UI e, em
              seguida, abrir as possíveis APIs em novas abas para recuperar
              JSON do backend.
98            frame = context.pages[-1]
99            # Click element
100           elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[16]
              ').nth(0)
101           await page.wait_for_timeout(3000); await elem.click
              (timeout=5000)
102
103           await page.goto("https://usecolaai.vercel.app/api/
              fluxo-caixa", wait_until="commit", timeout=10000)
104
105           await page.goto("https://usecolaai.vercel.app/api/
              assinatura", wait_until="commit", timeout=10000)
106
107           # -> Ler os arquivos extraídos (extracted_content_0.md e
              extracted_content_1.md) para obter os valores do frontend e,
              em seguida, buscar as respostas JSON dos endpoints backend /
              api/fluxo-caixa e /api/assinatura para extrair os dados e
              preparar a comparação.
108           await page.goto("https://usecolaai.vercel.app/api/
              fluxo-caixa", wait_until="commit", timeout=10000)
109
110           # -> Extrair o JSON bruto do backend para /api/fluxo-caixa e /
              api/assinatura (se disponível) e então comparar os campos
              relevantes com os valores extraídos do frontend (entradas,
              receita_de_pedidos, saídas, saldo_do_período, resumo_diário,
              top_entradas, top_saídas, movimentações e dados de assinatura/
              plano).
111           await page.goto("https://usecolaai.vercel.app/api/
              assinatura", wait_until="commit", timeout=10000)
112
113           await asyncio.sleep(5)
114
115       finally:
116           if context:
117               await context.close()
118           if browser:
```

```
119                 await browser.close()
120             if pw:
121                 await pw.stop()
122
123     asyncio.run(run_test())
124
```

Error

TimeoutError: Locator.click: Timeout 5000ms exceeded. Call log: - waiting for locator("xpath=html/body/div[2]/div[2]/div[2]/div/form/div[2]/div/button").first

Cause

The button identified by the XPath may not be visible or rendered correctly, possibly due to a loading issue or a JavaScript error preventing the page from fully functioning.

Fix

Ensure that the page content is fully loaded before executing the click action, perhaps by waiting for specific elements to be visible or by optimizing the load time of resources.

TC016-Order notifications via WhatsApp integration

ATTRIBUTES

Status	Failed
Priority	MEDIUM
Description	Verify that order notifications are sent correctly via WhatsApp to the snack bar as formatted messages.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/5478f4b8-00c1-70cc-8db6-48f4cfc182b1/1770755492734055/result.webm

Test Code

```python
import asyncio
from playwright import async_api

async def run_test():
    pw = None
    browser = None
    context = None

    try:
        # Start a Playwright session in asynchronous mode
        pw = await async_api.async_playwright().start()

        # Launch a Chromium browser in headless mode with custom arguments
        browser = await pw.chromium.launch(
            headless=True,
            args=[
                "--window-size=1280,720",         # Set the browser window size
                "--disable-dev-shm-usage",        # Avoid using /dev/shm which can cause issues in containers
                "--ipc=host",                     # Use host-level IPC for better stability
                "--single-process"                # Run the browser in a single process mode
            ],
        )

        # Create a new browser context (like an incognito window)
        context = await browser.new_context()
        context.set_default_timeout(5000)

        # Open a new page in the browser context
        page = await context.new_page()

        # Navigate to your target URL and wait until the network request is committed
        await page.goto("https://usecolaai.vercel.app", wait_until="commit", timeout=10000)

        # Wait for the main page to reach DOMContentLoaded state (optional for stability)
        try:
            await page.wait_for_load_state("domcontentloaded", timeout=3000)
        except async_api.Error:
            pass

        # Iterate through all iframes and wait for them to load as well
        for frame in page.frames:
            try:
                await frame.wait_for_load_state("domcontentloaded", timeout=3000)
            except async_api.Error:
                pass

        # Interact with the page elements to simulate user flow
```

```
48          # -> Navigate to https://usecolaai.vercel.app
49          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
50
51          # -> Fill the login form with provided credentials and submit
            to access the app dashboard.
52          frame = context.pages[-1]
53          # Input text
54          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[1]/div/input').nth(0)
55          await page.wait_for_timeout(3000); await elem.fill
            ('testefinal99@example.com')
56
57          frame = context.pages[-1]
58          # Input text
59          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/input').nth(0)
60          await page.wait_for_timeout(3000); await elem.fill
            ('test123456')
61
62          frame = context.pages[-1]
63          # Click element
64          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/button').nth(0)
65          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
66
67          # -> Submit the login form by clicking the 'Entrar' button
            (element index 157) and wait for the dashboard to load so the
            checkout flow can proceed.
68          frame = context.pages[-1]
69          # Click element
70          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/button').nth(0)
71          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
72
73          # -> Start a new order by clicking 'Novo Pedido' to begin the
            checkout flow that will allow creation of an order and
            performing the PIX payment to trigger the WhatsApp
            notification.
74          frame = context.pages[-1]
75          # Click element
76          elem = frame.locator('xpath=html/body/div[2]/main/div/div[1]/
            a/button').nth(0)
77          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
78
79          # -> Abrir a tela de criação de pedido clicando em 'Criar
            Pedido' para iniciar o fluxo de checkout e permitir adicionar
            itens e selecionar pagamento PIX.
80          frame = context.pages[-1]
81          # Click element
82          elem = frame.locator('xpath=html/body/div[2]/main/div/div[4]/
            div[1]/div[2]/a/button').nth(0)
83          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
84
85          # -> Adicionar um produto ao carrinho clicando em 'Hotdog
```

```
            Especial 1' para iniciar o fluxo de checkout (próxima ação:
            click no produto com índice 1827).
 86         frame = context.pages[-1]
 87         # Click element
 88         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/div[2]/div[1]/div').nth(0)
 89         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
 90
 91         # -> Abrir o carrinho/painel do pedido para prosseguir ao
            checkout (selecionar forma de pagamento PIX e finalizar
            pedido) para disparar a notificação WhatsApp.
 92         frame = context.pages[-1]
 93         # Click element
 94         elem = frame.locator('xpath=html/body/div[2]/main/div/div[3]/
            div').nth(0)
 95         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
 96
 97         # -> Preencher Nome e Telefone do cliente, selecionar PIX
            como forma de pagamento e clicar em 'Criar Pedido' para
            disparar a notificação WhatsApp (espera pelo snack bar será
            verificada após a ação).
 98         frame = context.pages[-1]
 99         # Input text
100         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[2]/div/div[3]/div[1]/div/input').nth(0)
101         await page.wait_for_timeout(3000); await elem.fill('Cliente
            Teste')
102
103         frame = context.pages[-1]
104         # Input text
105         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[2]/div/div[3]/div[2]/div/input').nth(0)
106         await page.wait_for_timeout(3000); await elem.fill('(11)
            91234-5678')
107
108         frame = context.pages[-1]
109         # Click element
110         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[2]/div/div[5]/div/button[2]').nth(0)
111         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
112
113         # -> Clicar em 'Criar Pedido' para concluir o pedido e
            disparar a notificação WhatsApp (capturar o snack bar em
            seguida).
114         frame = context.pages[-1]
115         # Click element
116         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[2]/div/button').nth(0)
117         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
118
119         # -> Abrir os detalhes do pedido #16 para procurar ações
            associadas ao envio de WhatsApp (botões de reenvio/ícone
            WhatsApp ou histórico de notificações) — clicar no botão do
            cartão do pedido #16 para abrir detalhes.
120         frame = context.pages[-1]
```

```
            # Click element
            elem = frame.locator('xpath=html/body/div[2]/main/div/div[3]/
            div/div[1]/div[1]/div[2]/div/button').nth(0)
            await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)

            # -> Clicar em 'Enviar WhatsApp' no menu do pedido #16,
            aguardar a notificação transitória (snackbar/toast) e extrair
            o texto completo da notificação (ou, se não existir, retornar
            NOT_FOUND e listar elementos visíveis relacionados ao pedido
            que mencionem WhatsApp com seus textos e índices).
            frame = context.pages[-1]
            # Click element
            elem = frame.locator('xpath=html/body/div[2]/main/div/div[3]/
            div/div[1]/div[1]/div[2]/div/div/button[1]').nth(0)
            await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)

            await asyncio.sleep(5)

        finally:
            if context:
                await context.close()
            if browser:
                await browser.close()
            if pw:
                await pw.stop()

    asyncio.run(run_test())

```

Error

TimeoutError: Locator.click: Timeout 5000ms exceeded. Call log: - waiting for locator("xpath=html/body/div[2]/div[2]/div[2]/div/form/div[2]/div/button").first

Cause

The button is not rendered properly or is obscured by other elements, causing the click action to timeout.

Fix

Ensure that the button is visible and accessible in the DOM before executing the click action. Additionally, check for any CSS styles or JavaScript scripts that might be affecting the button's visibility or rendering.

TC001-Successful user registration

ATTRIBUTES

Status	Failed
Priority	MEDIUM
Description	Verify that a new user can register successfully with valid input data.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/5478f4b8-00c1-70cc-8db6-48f4cfc182b1/1770755485498048/result.webm

Test Code

```python
import asyncio
from playwright import async_api

async def run_test():
    pw = None
    browser = None
    context = None

    try:
        # Start a Playwright session in asynchronous mode
        pw = await async_api.async_playwright().start()

        # Launch a Chromium browser in headless mode with custom arguments
        browser = await pw.chromium.launch(
            headless=True,
            args=[
                "--window-size=1280,720",         # Set the browser window size
                "--disable-dev-shm-usage",        # Avoid using /dev/shm which can cause issues in containers
                "--ipc=host",                     # Use host-level IPC for better stability
                "--single-process"                # Run the browser in a single process mode
            ],
        )

        # Create a new browser context (like an incognito window)
        context = await browser.new_context()
        context.set_default_timeout(5000)

        # Open a new page in the browser context
        page = await context.new_page()

        # Navigate to your target URL and wait until the network request is committed
        await page.goto("https://usecolaai.vercel.app", wait_until="commit", timeout=10000)

        # Wait for the main page to reach DOMContentLoaded state (optional for stability)
        try:
            await page.wait_for_load_state("domcontentloaded", timeout=3000)
        except async_api.Error:
            pass

        # Iterate through all iframes and wait for them to load as well
        for frame in page.frames:
            try:
                await frame.wait_for_load_state("domcontentloaded", timeout=3000)
            except async_api.Error:
                pass

        # Interact with the page elements to simulate user flow
```

```
48          # -> Navigate to https://usecolaai.vercel.app
49          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
50
51          # -> Clicar no link 'Crie gratuitamente' (index 146) para
            navegar até a página de registro e então preencher o
            formulário.
52          frame = context.pages[-1]
53          # Click element
54          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/p/a').nth(0)
55          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
56
57          # -> Click the 'Crie gratuitamente' link (index 146) again
            and wait for the registration form to load, then inspect the
            page for registration inputs.
58          frame = context.pages[-1]
59          # Click element
60          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/p/a').nth(0)
61          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
62
63          # -> Preencher o formulário de registro com Nome do Negócio
            'Teste Final', E-mail 'testefinal99@example.com', Senha
            'test123456', Confirmar Senha 'test123456' e clicar em 'Criar
            Conta' para submeter o formulário.
64          frame = context.pages[-1]
65          # Input text
66          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[1]/
            div/form/div[1]/div/input').nth(0)
67          await page.wait_for_timeout(3000); await elem.fill('Teste
            Final')
68
69          frame = context.pages[-1]
70          # Input text
71          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[1]/
            div/form/div[2]/div/input').nth(0)
72          await page.wait_for_timeout(3000); await elem.fill
            ('testefinal99@example.com')
73
74          frame = context.pages[-1]
75          # Input text
76          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[1]/
            div/form/div[3]/div/input').nth(0)
77          await page.wait_for_timeout(3000); await elem.fill
            ('test123456')
78
79          # -> Preencher 'Confirmar Senha' com 'test123456' e clicar em
            'Criar Conta' para submeter o formulário.
80          frame = context.pages[-1]
81          # Input text
82          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[1]/
            div/form/div[4]/div/input').nth(0)
83          await page.wait_for_timeout(3000); await elem.fill
            ('test123456')
84
85          frame = context.pages[-1]
```

```
            # Click element
            elem = frame.locator('xpath=html/body/div[2]/div[2]/div[1]/
            div/form/button').nth(0)
            await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)

            await asyncio.sleep(5)

        finally:
            if context:
                await context.close()
            if browser:
                await browser.close()
            if pw:
                await pw.stop()

    asyncio.run(run_test())

```

Error

TimeoutError: Locator.click: Timeout 5000ms exceeded. Call log: - waiting for locator("xpath=html/body/div[2]/div[2]/div[2]/div/p/a").first

Cause

The element located at 'xpath=html/body/div[2]/div[2]/div[2]/div/p/a' may not be rendered yet when the click action is attempted due to slow loading times or network latency

Fix

Ensure that the page loads fully before performing actions; consider increasing the timeout duration or implementing wait strategies to handle elements appearing dynamically.

TC019-User and store branding customization

ATTRIBUTES

Status	Failed
Priority	MEDIUM
Description	Verify that admin users can customize UI branding including colors and logos and these reflect in the app immediately.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/5478f4b8-00c1-70cc-8db6-48f4cfc182b1/1770755492366968/result.webm

Test Code

```python
import asyncio
from playwright import async_api

async def run_test():
    pw = None
    browser = None
    context = None

    try:
        # Start a Playwright session in asynchronous mode
        pw = await async_api.async_playwright().start()

        # Launch a Chromium browser in headless mode with custom arguments
        browser = await pw.chromium.launch(
            headless=True,
            args=[
                "--window-size=1280,720",         # Set the browser window size
                "--disable-dev-shm-usage",        # Avoid using /dev/shm which can cause issues in containers
                "--ipc=host",                     # Use host-level IPC for better stability
                "--single-process"                # Run the browser in a single process mode
            ],
        )

        # Create a new browser context (like an incognito window)
        context = await browser.new_context()
        context.set_default_timeout(5000)

        # Open a new page in the browser context
        page = await context.new_page()

        # Navigate to your target URL and wait until the network request is committed
        await page.goto("https://usecolaai.vercel.app", wait_until="commit", timeout=10000)

        # Wait for the main page to reach DOMContentLoaded state (optional for stability)
        try:
            await page.wait_for_load_state("domcontentloaded", timeout=3000)
        except async_api.Error:
            pass

        # Iterate through all iframes and wait for them to load as well
        for frame in page.frames:
            try:
                await frame.wait_for_load_state("domcontentloaded", timeout=3000)
            except async_api.Error:
                pass

        # Interact with the page elements to simulate user flow
```

```
48          # -> Navigate to https://usecolaai.vercel.app
49          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
50
51          # -> Preencher o formulário de login com as credenciais
            fornecidas (testefinal99@example.com / test123456) e submeter
            para entrar na conta admin e acessar as configurações de
            customização da loja (Branding/Customização).
52          frame = context.pages[-1]
53          # Input text
54          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[1]/div/input').nth(0)
55          await page.wait_for_timeout(3000); await elem.fill
            ('testefinal99@example.com')
56
57          frame = context.pages[-1]
58          # Input text
59          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/input').nth(0)
60          await page.wait_for_timeout(3000); await elem.fill
            ('test123456')
61
62          frame = context.pages[-1]
63          # Click element
64          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/button').nth(0)
65          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
66
67          # -> Retry clicking the 'Entrar' button and wait for the app
            to navigate or show an error. If still stuck, inspect for
            error messages or consider reloading or navigating to the
            login route.
68          frame = context.pages[-1]
69          # Click element
70          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[2]/div/button').nth(0)
71          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
72
73          # -> Open 'Configurações' to find the Branding/Customização
            settings (click the Configurações link in the sidebar).
74          frame = context.pages[-1]
75          # Click element
76          elem = frame.locator('xpath=html/body/div[2]/aside/div[2]/a').
            nth(0)
77          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
78
79          # -> Tentar abrir novamente 'Configurações' (clicar no item
            da sidebar). Se não abrir, rolar a página para localizar
            links/entradas para Branding/Customização e então acessá-los.
80          frame = context.pages[-1]
81          # Click element
82          elem = frame.locator('xpath=html/body/div[2]/aside/div[2]/a').
            nth(0)
83          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
84
```

```
85          # -> Abrir a aba 'Aparência' para acessar os controles de cor
            (clicar elemento index=1216).
86          frame = context.pages[-1]
87          # Click element
88          elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/button[2]').nth(0)
89          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
90
91          # -> Selecionar um tema pronto (Azul), salvar as alterações
            de Aparência e então abrir a aba 'Geral' para localizar o
            input de logo (preparar upload).
92          frame = context.pages[-1]
93          # Click element
94          elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[2]/div/div[2]/div/button[3]').nth(0)
95          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
96
97          frame = context.pages[-1]
98          # Click element
99          elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[2]/div/div[5]/button').nth(0)
100         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
101
102         frame = context.pages[-1]
103         # Click element
104         elem = frame.locator('xpath=html/body/div[2]/main/div/div[2]/
            div[1]/button[1]').nth(0)
105         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
106
107         # -> Navegar ao Dashboard para verificar que as cores (tema
            Azul) e o logo são refletidos imediatamente em toda a
            aplicação (sidebar/menu/dashboard).
108         frame = context.pages[-1]
109         # Click element
110         elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[1]').
            nth(0)
111         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
112
113         # -> Clicar em 'Dashboard' (elemento index=970) para
            verificar imediatamente se o tema (cores) e o logo aparecem
            no dashboard e na sidebar. Após navegação, inspecionar
            visualmente a presença do logo e das cores aplicadas.
114         frame = context.pages[-1]
115         # Click element
116         elem = frame.locator('xpath=html/body/div[2]/aside/nav/a[1]').
            nth(0)
117         await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
118
119         await asyncio.sleep(5)
120
121     finally:
122         if context:
123             await context.close()
```

```
            if browser:
                await browser.close()
            if pw:
                await pw.stop()

    asyncio.run(run_test())

```

Error

TimeoutError: Locator.click: Timeout 5000ms exceeded. Call log: - waiting for locator("xpath=html/body/div[2]/div[2]/div[2]/div/form/div[2]/div/button").first

Cause

The element targeted by the XPath locator may not be present in the DOM at the time of the click, possibly due to slow loading times, JavaScript errors, or other rendering issues.

Fix

Ensure that the element is loaded and visible before attempting to click. This may involve increasing the timeout duration, implementing wait mechanisms, or optimizing the webpage loading performance.

TC005-Password recovery workflow

ATTRIBUTES

Status	Failed
Priority	MEDIUM
Description	Verify that the password recovery process initiates and completes properly.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/5478f4b8-00c1-70cc-8db6-48f4cfc182b1/1770755485253391/result.webm

Test Code

```python
import asyncio
from playwright import async_api

async def run_test():
    pw = None
    browser = None
    context = None

    try:
        # Start a Playwright session in asynchronous mode
        pw = await async_api.async_playwright().start()

        # Launch a Chromium browser in headless mode with custom arguments
        browser = await pw.chromium.launch(
            headless=True,
            args=[
                "--window-size=1280,720",         # Set the browser window size
                "--disable-dev-shm-usage",        # Avoid using /dev/shm which can cause issues in containers
                "--ipc=host",                     # Use host-level IPC for better stability
                "--single-process"                # Run the browser in a single process mode
            ],
        )

        # Create a new browser context (like an incognito window)
        context = await browser.new_context()
        context.set_default_timeout(5000)

        # Open a new page in the browser context
        page = await context.new_page()

        # Navigate to your target URL and wait until the network request is committed
        await page.goto("https://usecolaai.vercel.app", wait_until="commit", timeout=10000)

        # Wait for the main page to reach DOMContentLoaded state (optional for stability)
        try:
            await page.wait_for_load_state("domcontentloaded", timeout=3000)
        except async_api.Error:
            pass

        # Iterate through all iframes and wait for them to load as well
        for frame in page.frames:
            try:
                await frame.wait_for_load_state("domcontentloaded", timeout=3000)
            except async_api.Error:
                pass

        # Interact with the page elements to simulate user flow
```

```
48          # -> Navigate to https://usecolaai.vercel.app
49          await page.goto("https://usecolaai.vercel.app",
            wait_until="commit", timeout=10000)
50
51          # -> Clicar no link 'Esqueceu a senha?' para navegar para a
            página de recuperação de senha e iniciar o fluxo.
52          frame = context.pages[-1]
53          # Click element
54          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[3]/a').nth(0)
55          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
56
57          # -> Clicar novamente no link 'Esqueceu a senha?' (index 191)
            para abrir a página/diálogo de recuperação de senha e iniciar
            o fluxo.
58          frame = context.pages[-1]
59          # Click element
60          elem = frame.locator('xpath=html/body/div[2]/div[2]/div[2]/
            div/form/div[3]/a').nth(0)
61          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
62
63          # -> Preencher o campo de E-mail com testefinal99@example.com
            e submeter o formulário de recuperação; depois verificar a
            mensagem de confirmação de envio.
64          frame = context.pages[-1]
65          # Input text
66          elem = frame.locator('xpath=html/body/div[2]/div[2]/div/form/
            div/div/input').nth(0)
67          await page.wait_for_timeout(3000); await elem.fill
            ('testefinal99@example.com')
68
69          frame = context.pages[-1]
70          # Click element
71          elem = frame.locator('xpath=html/body/div[2]/div[2]/div/form/
            button').nth(0)
72          await page.wait_for_timeout(3000); await elem.click
            (timeout=5000)
73
74          await asyncio.sleep(5)
75
76      finally:
77          if context:
78              await context.close()
79          if browser:
80              await browser.close()
81          if pw:
82              await pw.stop()
83
84  asyncio.run(run_test())
85
```

Error

TimeoutError: Locator.click: Timeout 5000ms exceeded. Call log: - waiting for locator("xpath=html/body/div[2]/div[2]/div[2]/div/form/div[3]/a").first

Cause

The element being targeted by the locator (XPath) is not present in the DOM, possibly due to slow loading times of the webpage or incorrect XPath that points to a non-existing element.

Fix

Optimize the loading time of the webpage by improving backend performance, ensuring that resources are loaded as quickly as possible, and verify that the correct XPath is used for the intended element.