Feb 9,2026

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	1 APIs
Total Websites Tested	1 Websites
Pass/Fail Rate	Backend: 0/10 Frontend: 0/11

2 Key Findings

Test Summary

The project lacks actual test results for backend and frontend components, which hinders a comprehensive evaluation of quality and reliability. The absence of performance metrics limits the ability to predict stability and user experience, highlighting potential risks before deployment. Addressing these gaps through thorough testing is crucial for accurate assessment.

What could be better

The most significant weakness is the absence of success and failure test results for both frontend and backend components. This lack of data prevents identifying areas for improvement, raises concerns about project readiness, and limits confidence in stability and user experience.

Recommendations

Conduct comprehensive testing for both frontend and backend components to gather actual success and failure rates. This data will provide insights into performance, allow for identification of weaknesses, and support enhancements in reliability and user experience ahead of deployment.

Backend API Test Results

3 Test Coverage Summary

API NAME	TEST CASES	TEST CATEGORY	PASS/FAIL RATE
TMS	10	5 Basic Functionality Tests 5 Edge Case Tests	0 Pass/10 Fail

Note

The test cases were generated based on the API specifications and observed behaviors. Some tests were adapted dynamically during execution based on API responses.

4 Test Execution Summary

TMS Execution Summary

TEST CASE	TEST DESCRIPTION	IMPACT	STATUS

Basic Functionality Tests			
Test POST with Missing Required Fields	Send a POST request that omits necessary required fields and check that the API returns a validation error with appropriate error codes and messages.	High	Failed
Test POST for Duplicate Entries	Send a POST request with data that would create a duplicate entry in the database and ensure the API correctly handles this by returning a relevant conflict error.	High	Failed
Test POST with Extra Fields	Send a POST request that includes additional fields not specified in the API documentation and check if the API handles this correctly, returning the appropriate response without error.	Medium	Failed
Test POST with Invalid Data Types	Attempt to send a POST request with fields that have incorrect data types and verify the API responds with validation errors indicating data type mismatch.	Medium	Failed
Test POST with Valid Data	Send a POST request to the API endpoint with valid data fields and verify that a successful response is received with correct status code and expected response structure.	High	Failed
Edge Case Tests			
Test POST with Malformed JSON	Send a POST request with a malformed JSON body and check that the API returns a correct error response indicating the JSON formatting issue.	High	Failed
Test Maximum Length of Fields	Attempt to send a POST request where the input fields are at their maximum allowed character length to verify that the API processes it correctly without issues.	Medium	Failed
Test POST with Special Characters	Send a POST request that includes special characters in its fields to check if the API can handle such inputs gracefully without any crashes or errors.	Medium	Failed
Test Empty Payload POST	Send a POST request with an empty body and verify that the API responds with a proper error message and status indicating that the request was invalid.	High	Failed
Test Unauthorized POST Request	Attempt to send a POST request without appropriate authentication or permissions, ensuring the API promptly rejects the request with an authorization error.	High	Failed

5 Test Execution Breakdown

TMS Failed Test Details

Test POST with Missing Required Fields

ATTRIBUTES

Status	Failed
Priority	High
Description	Send a POST request that omits necessary required fields and check that the API returns a validation error with appropriate error codes and messages.

Test Code

```
import requests
import json

def test_post_with_missing_required_fields():
    url = "http://localhost:3000/"
    headers = {
        "Authorization":
        "sk-user-T8LnOjBZvBnkQ9J5uPaXHscOQPNiVXjhkBZIj7TGffmQnG0X7GZFJs
        y1TM3hA4f96Lz_YJlMpIC5EzGruKmdOtpT_zxsTQT_x3zYeXf1HunQSmzt8wNoQ
        xb8IuLZbuv_fhM."
    }
    # Intentionally omitting required fields
    payload = {
        # Required fields should be here, but leaving them out for
        this test
        "optionalField": "someValue"
    }
    response = requests.post(url, headers=headers, json=payload)

    print(response.json())  # Print the response for debugging

test_post_with_missing_required_fields()
```

Error

HTTPConnectionPool(host='localhost', port=3000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f6ec6807dd0>: Failed to establish a new connection: [Errno 111] Connection refused'))

Trace

Test POST with Missing Required Fields

```
Traceback (most recent call last):
  File "/var/task/urllib3/connection.py", line 198, in _new_conn
    sock = connection.create_connection(
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/util/connection.py", line 85, in
  create_connection
    raise err
  File "/var/task/urllib3/util/connection.py", line 73, in
  create_connection
    sock.connect(sa)
ConnectionRefusedError: [Errno 111] Connection refused

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/var/task/urllib3/connectionpool.py", line 787, in urlopen
    response = self._make_request(
               ^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/connectionpool.py", line 493, in
  _make_request
    conn.request(
  File "/var/task/urllib3/connection.py", line 445, in request
    self.endheaders()
  File "/var/lang/lib/python3.12/http/client.py", line 1333, in
  endheaders
    self._send_output(message_body, encode_chunked=encode_chunked)
  File "/var/lang/lib/python3.12/http/client.py", line 1093, in
  _send_output
    self.send(msg)
  File "/var/lang/lib/python3.12/http/client.py", line 1037, in send
    self.connect()
  File "/var/task/urllib3/connection.py", line 276, in connect
    self.sock = self._new_conn()
                ^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/connection.py", line 213, in _new_conn
    raise NewConnectionError(
urllib3.exceptions.NewConnectionError: <urllib3.connection.
HTTPConnection object at 0x7f6ec6807dd0>: Failed to establish a new
connection: [Errno 111] Connection refused

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/var/task/requests/adapters.py", line 667, in send
    resp = conn.urlopen(
           ^^^^^^^^^^^^
  File "/var/task/urllib3/connectionpool.py", line 841, in urlopen
    retries = retries.increment(
              ^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/util/retry.py", line 519, in increment
    raise MaxRetryError(_pool, url, reason) from reason  # type: ignore
    [arg-type]
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
urllib3.exceptions.MaxRetryError: HTTPConnectionPool(host='localhost',
port=3000): Max retries exceeded with url: / (Caused by
NewConnectionError('<urllib3.connection.HTTPConnection object at
0x7f6ec6807dd0>: Failed to establish a new connection: [Errno 111]
Connection refused'))
```

```
47
48  During handling of the above exception, another exception occurred:
49
50  Traceback (most recent call last):
51    File "/var/task/main.py", line 60, in target
52      exec(code, env)
53    File "<string>", line 18, in <module>
54    File "<string>", line 14, in test_post_with_missing_required_fields
55    File "/var/task/requests/api.py", line 115, in post
56      return request("post", url, data=data, json=json, **kwargs)
57             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
58    File "/var/task/requests/api.py", line 59, in request
59      return session.request(method=method, url=url, **kwargs)
60             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
61    File "/var/task/requests/sessions.py", line 589, in request
62      resp = self.send(prep, **send_kwargs)
63             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
64    File "/var/task/requests/sessions.py", line 703, in send
65      r = adapter.send(request, **kwargs)
66          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
67    File "/var/task/requests/adapters.py", line 700, in send
68      raise ConnectionError(e, request=request)
69  requests.exceptions.ConnectionError: HTTPConnectionPool
    (host='localhost', port=3000): Max retries exceeded with url: /
    (Caused by NewConnectionError('<urllib3.connection.HTTPConnection
    object at 0x7f6ec6807dd0>: Failed to establish a new connection:
    [Errno 111] Connection refused'))
70
```

Cause

The API server is not running or is not reachable at the specified URL (localhost:3000). This causes the connection to be refused when the test attempts to send a POST request.

Fix

Ensure that the API server is running on the correct port (3000) and is accessible from the machine where the test is executed. If necessary, check firewall settings, server configurations, and logs for any issues that may be preventing the server from starting.

Test POST for Duplicate Entries

ATTRIBUTES

Status	Failed
Priority	High
Description	Send a POST request with data that would create a duplicate entry in the database and ensure the API correctly handles this by returning a relevant conflict error.

Test Code

```python
import requests
import json

API_URL = "http://localhost:3000/"
API_KEY = "sk-user-T8LnOjBZvBnkQ9J5uPaXHscOQPNiVXjhkBZIj7TGffmQnG0X7GZFJsy1TM3hA4f96Lz_YJlMpIC5EzGruKmdOtpT_zxsTQT_x3zYeXf1HunQSmzt8wNoQxb8IuLZbuv_fhM"

def test_post_duplicate_entries():
    headers = {
        'Authorization': f'Bearer {API_KEY}',
        'Content-Type': 'application/json'
    }

    payload = {
        'data': 'test_entry'
    }

    # First POST request
    response1 = requests.post(API_URL, headers=headers, data=json.dumps(payload))
    print("First POST response:", response1.json())

    # Second POST request (duplicate)
    response2 = requests.post(API_URL, headers=headers, data=json.dumps(payload))
    print("Second POST response:", response2.json())

    # Vague check for duplicate entries
    assert response2.status_code == response1.status_code, f"Expected status code {response1.status_code}, but got {response2.status_code}."

test_post_duplicate_entries()
```

Error

HTTPConnectionPool(host='localhost', port=3000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7fd5feccd2b0>: Failed to establish a new connection: [Errno 111] Connection refused'))

Trace

Test POST for Duplicate Entries

```
Traceback (most recent call last):
  File "/var/task/urllib3/connection.py", line 198, in _new_conn
    sock = connection.create_connection(
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/util/connection.py", line 85, in
  create_connection
    raise err
  File "/var/task/urllib3/util/connection.py", line 73, in
  create_connection
    sock.connect(sa)
ConnectionRefusedError: [Errno 111] Connection refused

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/var/task/urllib3/connectionpool.py", line 787, in urlopen
    response = self._make_request(
               ^^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/connectionpool.py", line 493, in
  _make_request
    conn.request(
  File "/var/task/urllib3/connection.py", line 445, in request
    self.endheaders()
  File "/var/lang/lib/python3.12/http/client.py", line 1333, in
  endheaders
    self._send_output(message_body, encode_chunked=encode_chunked)
  File "/var/lang/lib/python3.12/http/client.py", line 1093, in
  _send_output
    self.send(msg)
  File "/var/lang/lib/python3.12/http/client.py", line 1037, in send
    self.connect()
  File "/var/task/urllib3/connection.py", line 276, in connect
    self.sock = self._new_conn()
                ^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/connection.py", line 213, in _new_conn
    raise NewConnectionError(
urllib3.exceptions.NewConnectionError: <urllib3.connection.
HTTPConnection object at 0x7fd5feccd2b0>: Failed to establish a new
connection: [Errno 111] Connection refused

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/var/task/requests/adapters.py", line 667, in send
    resp = conn.urlopen(
           ^^^^^^^^^^^^^
  File "/var/task/urllib3/connectionpool.py", line 841, in urlopen
    retries = retries.increment(
              ^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/util/retry.py", line 519, in increment
    raise MaxRetryError(_pool, url, reason) from reason  # type: ignore
    [arg-type]
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
urllib3.exceptions.MaxRetryError: HTTPConnectionPool(host='localhost',
port=3000): Max retries exceeded with url: / (Caused by
NewConnectionError('<urllib3.connection.HTTPConnection object at
0x7fd5feccd2b0>: Failed to establish a new connection: [Errno 111]
Connection refused'))
```

```
47
48  During handling of the above exception, another exception occurred:
49
50  Traceback (most recent call last):
51    File "/var/task/main.py", line 60, in target
52      exec(code, env)
53    File "<string>", line 28, in <module>
54    File "<string>", line 18, in test_post_duplicate_entries
55    File "/var/task/requests/api.py", line 115, in post
56      return request("post", url, data=data, json=json, **kwargs)
57             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
58    File "/var/task/requests/api.py", line 59, in request
59      return session.request(method=method, url=url, **kwargs)
60             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
61    File "/var/task/requests/sessions.py", line 589, in request
62      resp = self.send(prep, **send_kwargs)
63             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
64    File "/var/task/requests/sessions.py", line 703, in send
65      r = adapter.send(request, **kwargs)
66          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
67    File "/var/task/requests/adapters.py", line 700, in send
68      raise ConnectionError(e, request=request)
69  requests.exceptions.ConnectionError: HTTPConnectionPool
    (host='localhost', port=3000): Max retries exceeded with url: /
    (Caused by NewConnectionError('<urllib3.connection.HTTPConnection
    object at 0x7fd5feccd2b0>: Failed to establish a new connection:
    [Errno 111] Connection refused'))
70
```

Cause

The API server at http://localhost:3000/ is not running or accessible, resulting in a connection refused error.

Fix

Ensure that the API server is up and running on localhost:3000. Check that the server process has started correctly and is listening on the specified port.

Test POST with Extra Fields

ATTRIBUTES

Status	Failed
Priority	Medium
Description	Send a POST request that includes additional fields not specified in the API documentation and check if the API handles this correctly, returning the appropriate response without error.

Test Code

```python
import requests
import json

def test_post_with_extra_fields():
    url = "http://localhost:3000/"
    headers = {
        "Authorization": "Bearer sk-user-T8LnOjBZvBnkQ9J5uPaXHscOQPNiVXjhkBZIj7TGffmQnG0X7GZFJsy1TM3hA4f96Lz_YJlMpIC5EzGruKmdOtpT_zxsTQT_x3zYeXf1HunQSmzt8wNoQxb8IuLZbuv_fhM."
    }
    data = {
        "field1": "value1",
        "field2": "value2",
        "extra_field": "extra_value"
    }
    response = requests.post(url, headers=headers, json=data)
    print("Response Status Code:", response.status_code)
    print("Response Body:", response.text)

    response_json = response.json()
    assert "success" in response_json, f"Expected 'success' key in response but got: {response_json}"

test_post_with_extra_fields()
```

Error

HTTPConnectionPool(host='localhost', port=3000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7fa63564f050>: Failed to establish a new connection: [Errno 111] Connection refused'))

Trace

Test POST with Extra Fields

```
Traceback (most recent call last):
  File "/var/task/urllib3/connection.py", line 198, in _new_conn
    sock = connection.create_connection(
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/util/connection.py", line 85, in
  create_connection
    raise err
  File "/var/task/urllib3/util/connection.py", line 73, in
  create_connection
    sock.connect(sa)
ConnectionRefusedError: [Errno 111] Connection refused

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/var/task/urllib3/connectionpool.py", line 787, in urlopen
    response = self._make_request(
               ^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/connectionpool.py", line 493, in
  _make_request
    conn.request(
  File "/var/task/urllib3/connection.py", line 445, in request
    self.endheaders()
  File "/var/lang/lib/python3.12/http/client.py", line 1333, in
  endheaders
    self._send_output(message_body, encode_chunked=encode_chunked)
  File "/var/lang/lib/python3.12/http/client.py", line 1093, in
  _send_output
    self.send(msg)
  File "/var/lang/lib/python3.12/http/client.py", line 1037, in send
    self.connect()
  File "/var/task/urllib3/connection.py", line 276, in connect
    self.sock = self._new_conn()
                ^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/connection.py", line 213, in _new_conn
    raise NewConnectionError(
urllib3.exceptions.NewConnectionError: <urllib3.connection.
HTTPConnection object at 0x7fa63564f050>: Failed to establish a new
connection: [Errno 111] Connection refused

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/var/task/requests/adapters.py", line 667, in send
    resp = conn.urlopen(
           ^^^^^^^^^^^^
  File "/var/task/urllib3/connectionpool.py", line 841, in urlopen
    retries = retries.increment(
              ^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/util/retry.py", line 519, in increment
    raise MaxRetryError(_pool, url, reason) from reason  # type: ignore
    [arg-type]
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
urllib3.exceptions.MaxRetryError: HTTPConnectionPool(host='localhost',
port=3000): Max retries exceeded with url: / (Caused by
NewConnectionError('<urllib3.connection.HTTPConnection object at
0x7fa63564f050>: Failed to establish a new connection: [Errno 111]
Connection refused'))
```

```
47
48  During handling of the above exception, another exception occurred:
49
50  Traceback (most recent call last):
51    File "/var/task/main.py", line 60, in target
52      exec(code, env)
53    File "<string>", line 21, in <module>
54    File "<string>", line 14, in test_post_with_extra_fields
55    File "/var/task/requests/api.py", line 115, in post
56      return request("post", url, data=data, json=json, **kwargs)
57             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
58    File "/var/task/requests/api.py", line 59, in request
59      return session.request(method=method, url=url, **kwargs)
60             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
61    File "/var/task/requests/sessions.py", line 589, in request
62      resp = self.send(prep, **send_kwargs)
63             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
64    File "/var/task/requests/sessions.py", line 703, in send
65      r = adapter.send(request, **kwargs)
66          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
67    File "/var/task/requests/adapters.py", line 700, in send
68      raise ConnectionError(e, request=request)
69  requests.exceptions.ConnectionError: HTTPConnectionPool
    (host='localhost', port=3000): Max retries exceeded with url: /
    (Caused by NewConnectionError('<urllib3.connection.HTTPConnection
    object at 0x7fa63564f050>: Failed to establish a new connection:
    [Errno 111] Connection refused'))
70
```

Cause

The API server is not running or is not reachable on the specified host and port (localhost:3000), leading to a connection refused error.

Fix

Ensure the API server is running and listening on port 3000. Verify the server's logs for any issues and start the server if it's not running.

Test POST with Malformed JSON

ATTRIBUTES

Status	Failed
Priority	High
Description	Send a POST request with a malformed JSON body and check that the API returns a correct error response indicating the JSON formatting issue.

Test Code

```python
import requests
import json

def test_post_malformed_json():
    url = "http://localhost:3000/"
    api_key = "sk-user-T8LnOjBZvBnkQ9J5uPaXHscOQPNiVXjhkBZIj7TGffmQnG0X7GZFJsy1TM3hA4f96Lz_YJlMpIC5EzGruKmdOtpT_zxsTQT_x3zYeXf1HunQSmzt8wNoQxb8IuLZbuv_fhM."

    headers = {
        "Authorization": f"Bearer {api_key}",
        "Content-Type": "application/json"
    }

    # Sending a POST request with malformed JSON
    malformed_json = "{'key': 'value'"  # Missing closing brace
    response = requests.post(url, headers=headers, data=malformed_json)

    print("Response Status Code:", response.status_code)
    print("Response Body:", response.text)

    assert response.status_code != 200, f"Expected failure status code, got {response.status_code} instead."


test_post_malformed_json()
```

Error

HTTPConnectionPool(host='localhost', port=3000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7ffb3da3c770>: Failed to establish a new connection: [Errno 111] Connection refused'))

Trace

Test POST with Malformed JSON

```
Traceback (most recent call last):
  File "/var/task/urllib3/connection.py", line 198, in _new_conn
    sock = connection.create_connection(
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/util/connection.py", line 85, in
  create_connection
    raise err
  File "/var/task/urllib3/util/connection.py", line 73, in
  create_connection
    sock.connect(sa)
ConnectionRefusedError: [Errno 111] Connection refused

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/var/task/urllib3/connectionpool.py", line 787, in urlopen
    response = self._make_request(
               ^^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/connectionpool.py", line 493, in
  _make_request
    conn.request(
  File "/var/task/urllib3/connection.py", line 445, in request
    self.endheaders()
  File "/var/lang/lib/python3.12/http/client.py", line 1333, in
  endheaders
    self._send_output(message_body, encode_chunked=encode_chunked)
  File "/var/lang/lib/python3.12/http/client.py", line 1093, in
  _send_output
    self.send(msg)
  File "/var/lang/lib/python3.12/http/client.py", line 1037, in send
    self.connect()
  File "/var/task/urllib3/connection.py", line 276, in connect
    self.sock = self._new_conn()
                ^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/connection.py", line 213, in _new_conn
    raise NewConnectionError(
urllib3.exceptions.NewConnectionError: <urllib3.connection.
HTTPConnection object at 0x7ffb3da3c770>: Failed to establish a new
connection: [Errno 111] Connection refused

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/var/task/requests/adapters.py", line 667, in send
    resp = conn.urlopen(
           ^^^^^^^^^^^^
  File "/var/task/urllib3/connectionpool.py", line 841, in urlopen
    retries = retries.increment(
              ^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/util/retry.py", line 519, in increment
    raise MaxRetryError(_pool, url, reason) from reason  # type: ignore
    [arg-type]
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
urllib3.exceptions.MaxRetryError: HTTPConnectionPool(host='localhost',
port=3000): Max retries exceeded with url: / (Caused by
NewConnectionError('<urllib3.connection.HTTPConnection object at
0x7ffb3da3c770>: Failed to establish a new connection: [Errno 111]
Connection refused'))
```

```
47
48  During handling of the above exception, another exception occurred:
49
50  Traceback (most recent call last):
51    File "/var/task/main.py", line 60, in target
52      exec(code, env)
53    File "<string>", line 23, in <module>
54    File "<string>", line 15, in test_post_malformed_json
55    File "/var/task/requests/api.py", line 115, in post
56      return request("post", url, data=data, json=json, **kwargs)
57             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
58    File "/var/task/requests/api.py", line 59, in request
59      return session.request(method=method, url=url, **kwargs)
60             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
61    File "/var/task/requests/sessions.py", line 589, in request
62      resp = self.send(prep, **send_kwargs)
63             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
64    File "/var/task/requests/sessions.py", line 703, in send
65      r = adapter.send(request, **kwargs)
66          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
67    File "/var/task/requests/adapters.py", line 700, in send
68      raise ConnectionError(e, request=request)
69  requests.exceptions.ConnectionError: HTTPConnectionPool
    (host='localhost', port=3000): Max retries exceeded with url: /
    (Caused by NewConnectionError('<urllib3.connection.HTTPConnection
    object at 0x7ffb3da3c770>: Failed to establish a new connection:
    [Errno 111] Connection refused'))
70
```

Cause

The API service is not running or accessible at the specified URL (localhost:3000), resulting in connection refusal when attempting to post malformed JSON.

Fix

Ensure that the API service is running on localhost:3000 and is correctly configured to accept POST requests. Additionally, verify any firewall or network settings that might prevent access.

Test Maximum Length of Fields

ATTRIBUTES

Status	Failed
Priority	Medium
Description	Attempt to send a POST request where the input fields are at their maximum allowed character length to verify that the API processes it correctly without issues.

Test Code

```python
import requests
import json

def test_maximum_length_of_fields():
    url = "http://localhost:3000/"
    headers = {
        "Authorization": "Bearer sk-user-T8LnOjBZvBnkQ9J5uPaXHscOQPNiVXjhkBZIj7TGffmQnG0X7GZFJsy1TM3hA4f96Lz_YJlMpIC5EzGruKmdOtpT_zxsTQT_x3zYeXf1HunQSmzt8wNoQxb8IuLZbuv_fhM.",
        "Content-Type": "application/json"
    }

    # Prepare a payload with maximum length fields
    payload = {
        "field1": "A" * 1000,  # Assuming maximum length is 1000 characters
        "field2": "B" * 1000,
        "field3": "C" * 1000
    }

    response = requests.post(url, headers=headers, json=payload)

    print("Response Status Code:", response.status_code)
    print("Response Body:", response.text)

test_maximum_length_of_fields()
```

Error

HTTPConnectionPool(host='localhost', port=3000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7fe1bc12e2d0>: Failed to establish a new connection: [Errno 111] Connection refused'))

Trace

Test Maximum Length of Fields

```
Traceback (most recent call last):
  File "/var/task/urllib3/connection.py", line 198, in _new_conn
    sock = connection.create_connection(
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/util/connection.py", line 85, in
  create_connection
    raise err
  File "/var/task/urllib3/util/connection.py", line 73, in
  create_connection
    sock.connect(sa)
ConnectionRefusedError: [Errno 111] Connection refused

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/var/task/urllib3/connectionpool.py", line 787, in urlopen
    response = self._make_request(
               ^^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/connectionpool.py", line 493, in
  _make_request
    conn.request(
  File "/var/task/urllib3/connection.py", line 445, in request
    self.endheaders()
  File "/var/lang/lib/python3.12/http/client.py", line 1333, in
  endheaders
    self._send_output(message_body, encode_chunked=encode_chunked)
  File "/var/lang/lib/python3.12/http/client.py", line 1093, in
  _send_output
    self.send(msg)
  File "/var/lang/lib/python3.12/http/client.py", line 1037, in send
    self.connect()
  File "/var/task/urllib3/connection.py", line 276, in connect
    self.sock = self._new_conn()
                ^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/connection.py", line 213, in _new_conn
    raise NewConnectionError(
urllib3.exceptions.NewConnectionError: <urllib3.connection.
HTTPConnection object at 0x7fe1bc12e2d0>: Failed to establish a new
connection: [Errno 111] Connection refused

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/var/task/requests/adapters.py", line 667, in send
    resp = conn.urlopen(
           ^^^^^^^^^^^^
  File "/var/task/urllib3/connectionpool.py", line 841, in urlopen
    retries = retries.increment(
              ^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/util/retry.py", line 519, in increment
    raise MaxRetryError(_pool, url, reason) from reason  # type: ignore
    [arg-type]
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
urllib3.exceptions.MaxRetryError: HTTPConnectionPool(host='localhost',
port=3000): Max retries exceeded with url: / (Caused by
NewConnectionError('<urllib3.connection.HTTPConnection object at
0x7fe1bc12e2d0>: Failed to establish a new connection: [Errno 111]
Connection refused'))
```

```
47
48  During handling of the above exception, another exception occurred:
49
50  Traceback (most recent call last):
51    File "/var/task/main.py", line 60, in target
52      exec(code, env)
53    File "<string>", line 23, in <module>
54    File "<string>", line 18, in test_maximum_length_of_fields
55    File "/var/task/requests/api.py", line 115, in post
56      return request("post", url, data=data, json=json, **kwargs)
57             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
58    File "/var/task/requests/api.py", line 59, in request
59      return session.request(method=method, url=url, **kwargs)
60             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
61    File "/var/task/requests/sessions.py", line 589, in request
62      resp = self.send(prep, **send_kwargs)
63             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
64    File "/var/task/requests/sessions.py", line 703, in send
65      r = adapter.send(request, **kwargs)
66          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
67    File "/var/task/requests/adapters.py", line 700, in send
68      raise ConnectionError(e, request=request)
69  requests.exceptions.ConnectionError: HTTPConnectionPool
    (host='localhost', port=3000): Max retries exceeded with url: /
    (Caused by NewConnectionError('<urllib3.connection.HTTPConnection
    object at 0x7fe1bc12e2d0>: Failed to establish a new connection:
    [Errno 111] Connection refused'))
70
```

Cause

The API service may not be running on the specified port (3000), leading to a connection refusal when attempting to reach 'http://localhost:3000/'

Fix

Ensure that the API service is running and listening on port 3000. If necessary, check the API server's configuration or logs for any errors preventing it from starting.

Test POST with Special Characters

ATTRIBUTES

Status	Failed
Priority	Medium
Description	Send a POST request that includes special characters in its fields to check if the API can handle such inputs gracefully without any crashes or errors.

Test Code

```python
import requests
import json

def test_post_with_special_characters():
    url = "http://localhost:3000/"
    api_key = "sk-user-T8LnOjBZvBnkQ9J5uPaXHscOQPNiVXjhkBZIj7TGffmQnG0X7GZFJsy1TM3hA4f96Lz_YJlMpIC5EzGruKmdOtpT_zxsTQT_x3zYeXf1HunQSmzt8wNoQxb8IuLZbuv_fhM."

    headers = {
        "Authorization": f"Bearer {api_key}",
        "Content-Type": "application/json"
    }

    data = {
        "test_key": "!@#$%^&*()_+-=[]{};':,.<>?/~`"
    }

    response = requests.post(url, headers=headers, data=json.dumps(data))

    print(f"Response status code: {response.status_code}")
    print(f"Response body: {response.text}")

test_post_with_special_characters()
```

Error

HTTPConnectionPool(host='localhost', port=3000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7ffb3db96600>: Failed to establish a new connection: [Errno 111] Connection refused'))

Trace

Test POST with Special Characters

```
Traceback (most recent call last):
  File "/var/task/urllib3/connection.py", line 198, in _new_conn
    sock = connection.create_connection(
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/util/connection.py", line 85, in
  create_connection
    raise err
  File "/var/task/urllib3/util/connection.py", line 73, in
  create_connection
    sock.connect(sa)
ConnectionRefusedError: [Errno 111] Connection refused

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/var/task/urllib3/connectionpool.py", line 787, in urlopen
    response = self._make_request(
               ^^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/connectionpool.py", line 493, in
  _make_request
    conn.request(
  File "/var/task/urllib3/connection.py", line 445, in request
    self.endheaders()
  File "/var/lang/lib/python3.12/http/client.py", line 1333, in
  endheaders
    self._send_output(message_body, encode_chunked=encode_chunked)
  File "/var/lang/lib/python3.12/http/client.py", line 1093, in
  _send_output
    self.send(msg)
  File "/var/lang/lib/python3.12/http/client.py", line 1037, in send
    self.connect()
  File "/var/task/urllib3/connection.py", line 276, in connect
    self.sock = self._new_conn()
                ^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/connection.py", line 213, in _new_conn
    raise NewConnectionError(
urllib3.exceptions.NewConnectionError: <urllib3.connection.
HTTPConnection object at 0x7ffb3db96600>: Failed to establish a new
connection: [Errno 111] Connection refused

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/var/task/requests/adapters.py", line 667, in send
    resp = conn.urlopen(
           ^^^^^^^^^^^^^
  File "/var/task/urllib3/connectionpool.py", line 841, in urlopen
    retries = retries.increment(
              ^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/util/retry.py", line 519, in increment
    raise MaxRetryError(_pool, url, reason) from reason  # type: ignore
    [arg-type]
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
urllib3.exceptions.MaxRetryError: HTTPConnectionPool(host='localhost',
port=3000): Max retries exceeded with url: / (Caused by
NewConnectionError('<urllib3.connection.HTTPConnection object at
0x7ffb3db96600>: Failed to establish a new connection: [Errno 111]
Connection refused'))
```

```
47
48  During handling of the above exception, another exception occurred:
49
50  Traceback (most recent call last):
51    File "/var/task/main.py", line 60, in target
52      exec(code, env)
53    File "<string>", line 22, in <module>
54    File "<string>", line 17, in test_post_with_special_characters
55    File "/var/task/requests/api.py", line 115, in post
56      return request("post", url, data=data, json=json, **kwargs)
57             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
58    File "/var/task/requests/api.py", line 59, in request
59      return session.request(method=method, url=url, **kwargs)
60             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
61    File "/var/task/requests/sessions.py", line 589, in request
62      resp = self.send(prep, **send_kwargs)
63             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
64    File "/var/task/requests/sessions.py", line 703, in send
65      r = adapter.send(request, **kwargs)
66          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
67    File "/var/task/requests/adapters.py", line 700, in send
68      raise ConnectionError(e, request=request)
69  requests.exceptions.ConnectionError: HTTPConnectionPool
    (host='localhost', port=3000): Max retries exceeded with url: /
    (Caused by NewConnectionError('<urllib3.connection.HTTPConnection
    object at 0x7ffb3db96600>: Failed to establish a new connection:
    [Errno 111] Connection refused'))
70
```

Cause

The API server is not running or listening on port 3000, which results in connection refusal when trying to access the endpoint.

Fix

Ensure that the API server is up and running on port 3000. Check any firewall settings or configurations that may prevent access to the server. If the server is running, verify that it is correctly configured to listen for incoming connections.

Test Empty Payload POST

ATTRIBUTES

Status	Failed
Priority	High
Description	Send a POST request with an empty body and verify that the API responds with a proper error message and status indicating that the request was invalid.

Test Code

```python
import requests
import json

def test_empty_payload_post():
    url = "http://localhost:3000/"
    headers = {
        "Authorization":
        "sk-user-T8LnOjBZvBnkQ9J5uPaXHscOQPNiVXjhkBZIj7TGffmQnG0X7GZFJs
        y1TM3hA4f96Lz_YJlMpIC5EzGruKmdOtpT_zxsTQT_x3zYeXf1HunQSmzt8wNoQ
        xb8IuLZbuv_fhM."
    }
    payload = {}

    response = requests.post(url, headers=headers, json=payload)
    print("Response Status Code:", response.status_code)
    print("Response Body:", response.text)

    # Check if the response contains a success or failure indication
    assert response.status_code in [200, 400], f"Expected status code
    200 or 400 but got {response.status_code}"
    if 'error' in response.json():
        print("Error message:", response.json()['error'])

test_empty_payload_post()
```

Error

HTTPConnectionPool(host='localhost', port=3000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7fda0a399e20>: Failed to establish a new connection: [Errno 111] Connection refused'))

Trace

Test Empty Payload POST

```
Traceback (most recent call last):
  File "/var/task/urllib3/connection.py", line 198, in _new_conn
    sock = connection.create_connection(
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/util/connection.py", line 85, in
  create_connection
    raise err
  File "/var/task/urllib3/util/connection.py", line 73, in
  create_connection
    sock.connect(sa)
ConnectionRefusedError: [Errno 111] Connection refused

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/var/task/urllib3/connectionpool.py", line 787, in urlopen
    response = self._make_request(
               ^^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/connectionpool.py", line 493, in
  _make_request
    conn.request(
  File "/var/task/urllib3/connection.py", line 445, in request
    self.endheaders()
  File "/var/lang/lib/python3.12/http/client.py", line 1333, in
  endheaders
    self._send_output(message_body, encode_chunked=encode_chunked)
  File "/var/lang/lib/python3.12/http/client.py", line 1093, in
  _send_output
    self.send(msg)
  File "/var/lang/lib/python3.12/http/client.py", line 1037, in send
    self.connect()
  File "/var/task/urllib3/connection.py", line 276, in connect
    self.sock = self._new_conn()
                ^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/connection.py", line 213, in _new_conn
    raise NewConnectionError(
urllib3.exceptions.NewConnectionError: <urllib3.connection.
HTTPConnection object at 0x7fda0a399e20>: Failed to establish a new
connection: [Errno 111] Connection refused

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/var/task/requests/adapters.py", line 667, in send
    resp = conn.urlopen(
           ^^^^^^^^^^^^^
  File "/var/task/urllib3/connectionpool.py", line 841, in urlopen
    retries = retries.increment(
              ^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/util/retry.py", line 519, in increment
    raise MaxRetryError(_pool, url, reason) from reason  # type: ignore
    [arg-type]
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
urllib3.exceptions.MaxRetryError: HTTPConnectionPool(host='localhost',
port=3000): Max retries exceeded with url: / (Caused by
NewConnectionError('<urllib3.connection.HTTPConnection object at
0x7fda0a399e20>: Failed to establish a new connection: [Errno 111]
Connection refused'))
```

```
47
48  During handling of the above exception, another exception occurred:
49
50  Traceback (most recent call last):
51    File "/var/task/main.py", line 60, in target
52      exec(code, env)
53    File "<string>", line 20, in <module>
54    File "<string>", line 11, in test_empty_payload_post
55    File "/var/task/requests/api.py", line 115, in post
56      return request("post", url, data=data, json=json, **kwargs)
57             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
58    File "/var/task/requests/api.py", line 59, in request
59      return session.request(method=method, url=url, **kwargs)
60             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
61    File "/var/task/requests/sessions.py", line 589, in request
62      resp = self.send(prep, **send_kwargs)
63             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
64    File "/var/task/requests/sessions.py", line 703, in send
65      r = adapter.send(request, **kwargs)
66          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
67    File "/var/task/requests/adapters.py", line 700, in send
68      raise ConnectionError(e, request=request)
69  requests.exceptions.ConnectionError: HTTPConnectionPool
    (host='localhost', port=3000): Max retries exceeded with url: /
    (Caused by NewConnectionError('<urllib3.connection.HTTPConnection
    object at 0x7fda0a399e20>: Failed to establish a new connection:
    [Errno 111] Connection refused'))
70
```

Cause

The API server is not running or not accessible on localhost:3000, leading to connection refusal errors.

Fix

Ensure that the API server is running on the specified host and port. You may also check firewall settings or network configurations that could block access to the server.

Test Unauthorized POST Request

ATTRIBUTES

Status	Failed
Priority	High
Description	Attempt to send a POST request without appropriate authentication or permissions, ensuring the API promptly rejects the request with an authorization error.

Test Code

```python
import requests
import json

def test_unauthorized_post_request():
    url = "http://localhost:3000/"
    headers = {
        "Authorization": "Bearer invalid-api-key"
    }
    data = {
        "key": "value"
    }

    response = requests.post(url, headers=headers, json=data)
    print(f"Response Status Code: {response.status_code}")
    print(f"Response Body: {response.text}")

    assert response.status_code == 401, f"Expected status code 401 but got {response.status_code}"

test_unauthorized_post_request()
```

Error

HTTPConnectionPool(host='localhost', port=3000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f6ec66ac740>: Failed to establish a new connection: [Errno 111] Connection refused'))

Trace

Test Unauthorized POST Request

```
Traceback (most recent call last):
  File "/var/task/urllib3/connection.py", line 198, in _new_conn
    sock = connection.create_connection(
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/util/connection.py", line 85, in
  create_connection
    raise err
  File "/var/task/urllib3/util/connection.py", line 73, in
  create_connection
    sock.connect(sa)
ConnectionRefusedError: [Errno 111] Connection refused

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/var/task/urllib3/connectionpool.py", line 787, in urlopen
    response = self._make_request(
               ^^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/connectionpool.py", line 493, in
  _make_request
    conn.request(
  File "/var/task/urllib3/connection.py", line 445, in request
    self.endheaders()
  File "/var/lang/lib/python3.12/http/client.py", line 1333, in
  endheaders
    self._send_output(message_body, encode_chunked=encode_chunked)
  File "/var/lang/lib/python3.12/http/client.py", line 1093, in
  _send_output
    self.send(msg)
  File "/var/lang/lib/python3.12/http/client.py", line 1037, in send
    self.connect()
  File "/var/task/urllib3/connection.py", line 276, in connect
    self.sock = self._new_conn()
                ^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/connection.py", line 213, in _new_conn
    raise NewConnectionError(
urllib3.exceptions.NewConnectionError: <urllib3.connection.
HTTPConnection object at 0x7f6ec66ac740>: Failed to establish a new
connection: [Errno 111] Connection refused

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/var/task/requests/adapters.py", line 667, in send
    resp = conn.urlopen(
           ^^^^^^^^^^^^^
  File "/var/task/urllib3/connectionpool.py", line 841, in urlopen
    retries = retries.increment(
              ^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/util/retry.py", line 519, in increment
    raise MaxRetryError(_pool, url, reason) from reason  # type: ignore
    [arg-type]
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
urllib3.exceptions.MaxRetryError: HTTPConnectionPool(host='localhost',
port=3000): Max retries exceeded with url: / (Caused by
NewConnectionError('<urllib3.connection.HTTPConnection object at
0x7f6ec66ac740>: Failed to establish a new connection: [Errno 111]
Connection refused'))
```

```
47
48  During handling of the above exception, another exception occurred:
49
50  Traceback (most recent call last):
51    File "/var/task/main.py", line 60, in target
52      exec(code, env)
53    File "<string>", line 19, in <module>
54    File "<string>", line 13, in test_unauthorized_post_request
55    File "/var/task/requests/api.py", line 115, in post
56      return request("post", url, data=data, json=json, **kwargs)
57             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
58    File "/var/task/requests/api.py", line 59, in request
59      return session.request(method=method, url=url, **kwargs)
60             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
61    File "/var/task/requests/sessions.py", line 589, in request
62      resp = self.send(prep, **send_kwargs)
63             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
64    File "/var/task/requests/sessions.py", line 703, in send
65      r = adapter.send(request, **kwargs)
66          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
67    File "/var/task/requests/adapters.py", line 700, in send
68      raise ConnectionError(e, request=request)
69  requests.exceptions.ConnectionError: HTTPConnectionPool
    (host='localhost', port=3000): Max retries exceeded with url: /
    (Caused by NewConnectionError('<urllib3.connection.HTTPConnection
    object at 0x7f6ec66ac740>: Failed to establish a new connection:
    [Errno 111] Connection refused'))
70
```

Cause

The API service is not running or is inaccessible, resulting in a connection refusal error.

Fix

Ensure that the API server is running on the specified host and port (localhost:3000). Additionally, verify that there are no firewall or network issues blocking access to the API.

Test POST with Invalid Data Types

ATTRIBUTES

Status	Failed
Priority	Medium
Description	Attempt to send a POST request with fields that have incorrect data types and verify the API responds with validation errors indicating data type mismatch.

Test Code

```python
import requests
import json

def test_post_with_invalid_data_types():
    url = "http://localhost:3000/"
    headers = {
        "Authorization": "Bearer
        sk-user-T8LnOjBZvBnkQ9J5uPaXHscOQPNiVXjhkBZIj7TGffmQnG0X7GZFJsy
        1TM3hA4f96Lz_YJlMpIC5EzGruKmdOtpT_zxsTQT_x3zYeXf1HunQSmzt8wNoQx
        b8IuLZbuv_fhM.",
        "Content-Type": "application/json"
    }
    payload = {
        "name": 123,  # Invalid data type: should be a string
        "age": "twenty",  # Invalid data type: should be an integer
        "email": True  # Invalid data type: should be a string
    }

    response = requests.post(url, headers=headers, json=payload)
    print(f"Response Code: {response.status_code}")
    print(f"Response Body: {response.text}")

test_post_with_invalid_data_types()
```

Error

HTTPConnectionPool(host='localhost', port=3000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7fdc18c2e750>: Failed to establish a new connection: [Errno 111] Connection refused'))

Trace

Test POST with Invalid Data Types

```
Traceback (most recent call last):
  File "/var/task/urllib3/connection.py", line 198, in _new_conn
    sock = connection.create_connection(
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/util/connection.py", line 85, in
  create_connection
    raise err
  File "/var/task/urllib3/util/connection.py", line 73, in
  create_connection
    sock.connect(sa)
ConnectionRefusedError: [Errno 111] Connection refused

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/var/task/urllib3/connectionpool.py", line 787, in urlopen
    response = self._make_request(
               ^^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/connectionpool.py", line 493, in
  _make_request
    conn.request(
  File "/var/task/urllib3/connection.py", line 445, in request
    self.endheaders()
  File "/var/lang/lib/python3.12/http/client.py", line 1333, in
  endheaders
    self._send_output(message_body, encode_chunked=encode_chunked)
  File "/var/lang/lib/python3.12/http/client.py", line 1093, in
  _send_output
    self.send(msg)
  File "/var/lang/lib/python3.12/http/client.py", line 1037, in send
    self.connect()
  File "/var/task/urllib3/connection.py", line 276, in connect
    self.sock = self._new_conn()
                ^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/connection.py", line 213, in _new_conn
    raise NewConnectionError(
urllib3.exceptions.NewConnectionError: <urllib3.connection.
HTTPConnection object at 0x7fdc18c2e750>: Failed to establish a new
connection: [Errno 111] Connection refused

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/var/task/requests/adapters.py", line 667, in send
    resp = conn.urlopen(
           ^^^^^^^^^^^^^
  File "/var/task/urllib3/connectionpool.py", line 841, in urlopen
    retries = retries.increment(
              ^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/util/retry.py", line 519, in increment
    raise MaxRetryError(_pool, url, reason) from reason  # type: ignore
    [arg-type]
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
urllib3.exceptions.MaxRetryError: HTTPConnectionPool(host='localhost',
port=3000): Max retries exceeded with url: / (Caused by
NewConnectionError('<urllib3.connection.HTTPConnection object at
0x7fdc18c2e750>: Failed to establish a new connection: [Errno 111]
Connection refused'))
```

```
47
48  During handling of the above exception, another exception occurred:
49
50  Traceback (most recent call last):
51    File "/var/task/main.py", line 60, in target
52      exec(code, env)
53    File "<string>", line 20, in <module>
54    File "<string>", line 16, in test_post_with_invalid_data_types
55    File "/var/task/requests/api.py", line 115, in post
56      return request("post", url, data=data, json=json, **kwargs)
57             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
58    File "/var/task/requests/api.py", line 59, in request
59      return session.request(method=method, url=url, **kwargs)
60             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
61    File "/var/task/requests/sessions.py", line 589, in request
62      resp = self.send(prep, **send_kwargs)
63             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
64    File "/var/task/requests/sessions.py", line 703, in send
65      r = adapter.send(request, **kwargs)
66          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
67    File "/var/task/requests/adapters.py", line 700, in send
68      raise ConnectionError(e, request=request)
69  requests.exceptions.ConnectionError: HTTPConnectionPool
    (host='localhost', port=3000): Max retries exceeded with url: /
    (Caused by NewConnectionError('<urllib3.connection.HTTPConnection
    object at 0x7fdc18c2e750>: Failed to establish a new connection:
    [Errno 111] Connection refused'))
70
```

Cause

The API server is not running or is not reachable at the specified address (localhost:3000).

Fix

Start the API server and ensure it is configured to listen on the correct port (3000) and address (localhost).

Test POST with Valid Data

ATTRIBUTES

Status	Failed
Priority	High
Description	Send a POST request to the API endpoint with valid data fields and verify that a successful response is received with correct status code and expected response structure.

Test Code

```python
import requests
import json

def test_post_with_valid_data():
    url = "http://localhost:3000/"
    headers = {
        "Authorization": "Bearer
        sk-user-T8LnOjBZvBnkQ9J5uPaXHscOQPNiVXjhkBZIj7TGffmQnG0X7GZFJsy
        1TM3hA4f96Lz_YJlMpIC5EzGruKmdOtpT_zxsTQT_x3zYeXf1HunQSmzt8wNoQx
        b8IuLZbuv_fhM"
    }
    valid_data = {
        "key": "value"
    }

    response = requests.post(url, headers=headers, json=valid_data)
    print(response.json())

test_post_with_valid_data()
```

Error

HTTPConnectionPool(host='localhost', port=3000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7ffb3db96cf0>: Failed to establish a new connection: [Errno 111] Connection refused'))

Trace

Test POST with Valid Data

```
Traceback (most recent call last):
  File "/var/task/urllib3/connection.py", line 198, in _new_conn
    sock = connection.create_connection(
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/util/connection.py", line 85, in
  create_connection
    raise err
  File "/var/task/urllib3/util/connection.py", line 73, in
  create_connection
    sock.connect(sa)
ConnectionRefusedError: [Errno 111] Connection refused

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/var/task/urllib3/connectionpool.py", line 787, in urlopen
    response = self._make_request(
               ^^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/connectionpool.py", line 493, in
  _make_request
    conn.request(
  File "/var/task/urllib3/connection.py", line 445, in request
    self.endheaders()
  File "/var/lang/lib/python3.12/http/client.py", line 1333, in
  endheaders
    self._send_output(message_body, encode_chunked=encode_chunked)
  File "/var/lang/lib/python3.12/http/client.py", line 1093, in
  _send_output
    self.send(msg)
  File "/var/lang/lib/python3.12/http/client.py", line 1037, in send
    self.connect()
  File "/var/task/urllib3/connection.py", line 276, in connect
    self.sock = self._new_conn()
                ^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/connection.py", line 213, in _new_conn
    raise NewConnectionError(
urllib3.exceptions.NewConnectionError: <urllib3.connection.
HTTPConnection object at 0x7ffb3db96cf0>: Failed to establish a new
connection: [Errno 111] Connection refused

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "/var/task/requests/adapters.py", line 667, in send
    resp = conn.urlopen(
           ^^^^^^^^^^^^^
  File "/var/task/urllib3/connectionpool.py", line 841, in urlopen
    retries = retries.increment(
              ^^^^^^^^^^^^^^^^^^
  File "/var/task/urllib3/util/retry.py", line 519, in increment
    raise MaxRetryError(_pool, url, reason) from reason  # type: ignore
    [arg-type]
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
urllib3.exceptions.MaxRetryError: HTTPConnectionPool(host='localhost',
port=3000): Max retries exceeded with url: / (Caused by
NewConnectionError('<urllib3.connection.HTTPConnection object at
0x7ffb3db96cf0>: Failed to establish a new connection: [Errno 111]
Connection refused'))
```

```
47
48  During handling of the above exception, another exception occurred:
49
50  Traceback (most recent call last):
51    File "/var/task/main.py", line 60, in target
52      exec(code, env)
53    File "<string>", line 16, in <module>
54    File "<string>", line 13, in test_post_with_valid_data
55    File "/var/task/requests/api.py", line 115, in post
56      return request("post", url, data=data, json=json, **kwargs)
57             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
58    File "/var/task/requests/api.py", line 59, in request
59      return session.request(method=method, url=url, **kwargs)
60             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
61    File "/var/task/requests/sessions.py", line 589, in request
62      resp = self.send(prep, **send_kwargs)
63             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
64    File "/var/task/requests/sessions.py", line 703, in send
65      r = adapter.send(request, **kwargs)
66          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
67    File "/var/task/requests/adapters.py", line 700, in send
68      raise ConnectionError(e, request=request)
69  requests.exceptions.ConnectionError: HTTPConnectionPool
    (host='localhost', port=3000): Max retries exceeded with url: /
    (Caused by NewConnectionError('<urllib3.connection.HTTPConnection
    object at 0x7ffb3db96cf0>: Failed to establish a new connection:
    [Errno 111] Connection refused'))
70
```

Cause

The API server is not running or not listening on port 3000, leading to connection refusal errors.

Fix

Ensure that the API server is running properly and listening on port 3000. Check server logs for any startup errors or misconfigurations.

Frontend UI Test Results

6 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
TMS	11	0 Pass/11 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.

7 Test Execution Summary

TMS Execution Summary

TEST CASE	TEST DESCRIPTION	IMPACT	STATUS
Offline mode, network throttling and retry behavior	Given cached data and UI that depends on network, when the network goes offline or is throttled, then cache renders where expected, optimistic UI/queued actions behave predictably, background retries and user-facing error banners are shown, and recovery upon reconnect syncs state with backend without duplication. Setup: simulate network conditions via devtools or test harness.	Medium	Failed
Role-based access control and protected route enforcement	Given users with different roles (regular, admin), when each user attempts to access role-protected pages and perform restricted actions, then the UI and backend enforce permissions (show/hide UI, 403 responses where appropriate), role changes take effect without full redeploy, and unauthorized attempts produce clear feedback. Setup: create users with roles via API. Cleanup: revoke test roles.	Medium	Failed
Real-time updates and reconnection handling (WebSockets/Server-Sent events)	Given a component that receives live updates (notifications, chat or status), when server pushes messages and the connection is interrupted, then the client displays updates in real time, attempts reconnection with backoff, and upon reconnect synchronizes missed messages without duplicates. Also validate teardown and subscription cleanup. Setup: use a test real-time server that can simulate disconnects.	Low	Failed
Complex form submission with client & server validation and file upload	Given an authenticated user on the multi-field form (including text, date, selects, file upload), when the user submits valid data and an image, then the UI shows upload progress, server-side validation passes, data persists and success UI displays. Also test client-side validation errors, server validation error handling, and file size/format rejection paths. Setup: mock or use test endpoints for file storage. Cleanup: remove created entries and files.	High	Failed
Email/password authentication, session persistence and logout	Given a registered test user, when the user logs in with valid credentials and then refreshes the browser, then the user remains authenticated (valid session/token), protected routes are accessible, and after logout protected routes redirect to login. Also verify invalid credentials show appropriate server/client validation messages. Setup: create/delete test user via API.	High	Failed
Shopping cart lifecycle and checkout (success and payment failure)	Given a catalog of items and a test payment sandbox, when a user adds/removes items, updates quantities and applies coupons, then totals recalc correctly, cart persists across refresh, and during checkout payment success completes order while payment failure surfaces recoverable error and allows retry. Setup: use sandbox payment gateway and create a test customer. Cleanup: cancel test orders.	High	Failed
Accessibility flows: keyboard navigation, focus management and screen reader landmarks	Given standard pages (homepage, form page, modal), when users navigate using keyboard and assistive tech, then focus order is logical, modals trap focus and restore on close, skip-links work, ARIA roles/labels are present for key controls, and critical interactions are operable without a mouse. Include checks for color contrast on critical UI. Setup: run with automated AX checks and manual keyboard runs.	Low	Failed
File/image upload preview, transformation and retry on failure	Given the upload flow that generates previews and server-side transformations, when a user selects images of various sizes/formats, then client previews render, server returns transformed assets, and interrupted uploads support resume/retry with correct deduplication. Also test invalid formats and maximum size enforcement. Setup: test storage endpoint and sample files. Cleanup: remove test assets.	Medium	Failed
Responsive layout and component behavior across breakpoints	Given desktop, tablet and mobile viewports, when users interact with navigation, modals, sidebars and forms, then layout adapts (menu collapses, modals full-screen on mobile), interactive elements remain reachable, and no content is clipped or overlapping. Verify critical flows (login, search, add-to-cart) on each breakpoint. Setup: run tests with emulated viewports.	Medium	Failed
Main navigation, deep links and 404 routing	Given a seeded app with pages A/B/C and a nonexistent path, when a user navigates via header links, direct deep links, and browser back/forward, then the correct pages render, route params persist, scroll restoration works, and unknown paths show the 404 page. Setup: create necessary test data. Cleanup: reset routing state.	High	Failed
Search, filtering, sorting and pagination end-to-end	Given a dataset with varied attributes, when a user performs a search, applies multiple filters, changes sort order and paginates, then results update correctly (matching backend), URLs reflect query params for deep linking, and edge cases (no matches, very large result sets) show proper empty states and performance remains acceptable. Setup: seed dataset. Cleanup: clear test data.	High	Failed

8 Test Execution Breakdown

TMS Failed Test Details

Offline mode, network throttling and retry behavior

ATTRIBUTES

Status	Failed
Priority	Medium
Description	Given cached data and UI that depends on network, when the network goes offline or is throttled, then cache renders where expected, optimistic UI/queued actions behave predictably, background retries and user-facing error banners are shown, and recovery upon reconnect syncs state with backend without duplication. Setup: simulate network conditions via devtools or test harness.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/f4786438-60d1-7030-46d1-b1ac9b0e3b51/1770603194732772//tmp/501d64d5-6936-45f5-9718-a19e7f4b70da/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("http://localhost:3000", 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

```

```
            await asyncio.sleep(5)

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

asyncio.run(run_test())
```

Role-based access control and protected route enforcement

ATTRIBUTES

Status	Failed
Priority	Medium
Description	Given users with different roles (regular, admin), when each user attempts to access role-protected pages and perform restricted actions, then the UI and backend enforce permissions (show/hide UI, 403 responses where appropriate), role changes take effect without full redeploy, and unauthorized attempts produce clear feedback. Setup: create users with roles via API. Cleanup: revoke test roles.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/f4786438-60d1-7030-46d1-b1ac9b0e3b51/1770603194602732//tmp/36f6e0f6-a865-4e81-b283-3321dbd9bfd4/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("http://localhost:3000", 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

```

```
            await asyncio.sleep(5)

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

asyncio.run(run_test())
```

Real-time updates and reconnection handling (WebSockets/Server-Sent events)

ATTRIBUTES

Status	Failed
Priority	Low
Description	Given a component that receives live updates (notifications, chat or status), when server pushes messages and the connection is interrupted, then the client displays updates in real time, attempts reconnection with backoff, and upon reconnect synchronizes missed messages without duplicates. Also validate teardown and subscription cleanup. Setup: use a test real-time server that can simulate disconnects.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/f4786438-60d1-7030-46d1-b1ac9b0e3b51/1770603194501457//tmp/52890b95-ce4e-4c87-ab0d-972757f626a4/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("http://localhost:3000", 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

```

```
            await asyncio.sleep(5)

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

    asyncio.run(run_test())

```

Complex form submission with client & server validation and file upload

ATTRIBUTES

Status	Failed
Priority	High
Description	Given an authenticated user on the multi-field form (including text, date, selects, file upload), when the user submits valid data and an image, then the UI shows upload progress, server-side validation passes, data persists and success UI displays. Also test client-side validation errors, server validation error handling, and file size/format rejection paths. Setup: mock or use test endpoints for file storage. Cleanup: remove created entries and files.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/f4786438-60d1-7030-46d1-b1ac9b0e3b51/1770603195065735//tmp/7b00c85f-134b-439b-9fb1-7915a807070f/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("http://localhost:3000", 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

```

```
            await asyncio.sleep(5)

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

    asyncio.run(run_test())

```

Email/password authentication, session persistence and logout

ATTRIBUTES

Status	Failed
Priority	High
Description	Given a registered test user, when the user logs in with valid credentials and then refreshes the browser, then the user remains authenticated (valid session/token), protected routes are accessible, and after logout protected routes redirect to login. Also verify invalid credentials show appropriate server/client validation messages. Setup: create/delete test user via API.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/f4786438-60d1-7030-46d1-b1ac9b0e3b51/177060319477095//tmp/5adf6790-2e6d-4695-9019-2a6b66c521db/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("http://localhost:3000", 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

```

```
            await asyncio.sleep(5)

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

asyncio.run(run_test())
```

Shopping cart lifecycle and checkout (success and payment failure)

ATTRIBUTES

Status	Failed
Priority	High
Description	Given a catalog of items and a test payment sandbox, when a user adds/removes items, updates quantities and applies coupons, then totals recalc correctly, cart persists across refresh, and during checkout payment success completes order while payment failure surfaces recoverable error and allows retry. Setup: use sandbox payment gateway and create a test customer. Cleanup: cancel test orders.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/f4786438-60d1-7030-46d1-b1ac9b0e3b51/1770603196212873//tmp/d6de89bb-3877-46da-8c05-a9aa3af336df/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("http://localhost:3000", 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

```

```
            await asyncio.sleep(5)

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

asyncio.run(run_test())

```

Accessibility flows: keyboard navigation, focus management and screen reader landmarks

ATTRIBUTES

Status	Failed
Priority	Low
Description	Given standard pages (homepage, form page, modal), when users navigate using keyboard and assistive tech, then focus order is logical, modals trap focus and restore on close, skip-links work, ARIA roles/labels are present for key controls, and critical interactions are operable without a mouse. Include checks for color contrast on critical UI. Setup: run with automated AX checks and manual keyboard runs.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/f4786438-60d1-7030-46d1-b1ac9b0e3b51/1770603194754089//tmp/2d90dec4-d72e-4f7c-a72b-b16fc4aa5552/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("http://localhost:3000", 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

```

```
            await asyncio.sleep(5)

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

asyncio.run(run_test())
```

File/image upload preview, transformation and retry on failure

ATTRIBUTES

Status	Failed
Priority	Medium
Description	Given the upload flow that generates previews and server-side transformations, when a user selects images of various sizes/formats, then client previews render, server returns transformed assets, and interrupted uploads support resume/retry with correct deduplication. Also test invalid formats and maximum size enforcement. Setup: test storage endpoint and sample files. Cleanup: remove test assets.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/f4786438-60d1-7030-46d1-b1ac9b0e3b51/1770603195251439//tmp/7be08a1a-367e-48ad-8202-cbb865c5f909/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("http://localhost:3000", 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

```

```
            await asyncio.sleep(5)

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

asyncio.run(run_test())

```

Responsive layout and component behavior across breakpoints

ATTRIBUTES

Status	Failed
Priority	Medium
Description	Given desktop, tablet and mobile viewports, when users interact with navigation, modals, sidebars and forms, then layout adapts (menu collapses, modals full-screen on mobile), interactive elements remain reachable, and no content is clipped or overlapping. Verify critical flows (login, search, add-to-cart) on each breakpoint. Setup: run tests with emulated viewports.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/f4786438-60d1-7030-46d1-b1ac9b0e3b51/1770603193703234//tmp/6c477b69-742f-4f91-9ed7-f332a53e2e7d/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("http://localhost:3000", 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

```

```
            await asyncio.sleep(5)

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

asyncio.run(run_test())

```

Main navigation, deep links and 404 routing

ATTRIBUTES

Status	Failed
Priority	High
Description	Given a seeded app with pages A/B/C and a nonexistent path, when a user navigates via header links, direct deep links, and browser back/forward, then the correct pages render, route params persist, scroll restoration works, and unknown paths show the 404 page. Setup: create necessary test data. Cleanup: reset routing state.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/f4786438-60d1-7030-46d1-b1ac9b0e3b51/1770603193537674//tmp/010000f9-175a-449e-8f72-b8806e3afaa1/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("http://localhost:3000", 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

```

```
            await asyncio.sleep(5)

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

asyncio.run(run_test())
```

Search, filtering, sorting and pagination end-to-end

ATTRIBUTES

Status	Failed
Priority	High
Description	Given a dataset with varied attributes, when a user performs a search, applies multiple filters, changes sort order and paginates, then results update correctly (matching backend), URLs reflect query params for deep linking, and edge cases (no matches, very large result sets) show proper empty states and performance remains acceptable. Setup: seed dataset. Cleanup: clear test data.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/f4786438-60d1-7030-46d1-b1ac9b0e3b51/1770603196161743//tmp/1aa3e49e-9629-4d10-ac9a-16ed891d81be/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("http://localhost:3000", 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

```

```
            await asyncio.sleep(5)

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

asyncio.run(run_test())

```