Nov 21,2025

TESTING REPORT

Report Contents

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: 0/12

2 Key Findings

Test Summary

The project has achieved a moderate quality score of 75, indicating basic reliability but also suggesting potential vulnerabilities due to the absence of detailed testing data. The lack of frontend and backend tests prevents a comprehensive analysis, limiting insights into system stability and user experience.

What could be better

The project faces significant challenges due to the absence of frontend and backend test data, which hinders the identification of failure patterns and critical issues. Without testing insights, the project risks overlooking performance bottlenecks and may struggle to ensure a satisfactory user experience.

Recommendations

Implement comprehensive testing protocols for both backend APIs and frontend URLs. Conduct thorough tests to gather actionable insights about system performance, and use this data to identify critical failures and prioritize areas for improvement, ultimately enhancing the project's reliability.

Backend API Test Results

3 Test Coverage Summary

API NAME	TEST CASES	TEST CATEGORY	PASS/FAIL RATE
front	10	5 Edge Cases 5 Functional 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

Front Execution Summary

TEST CASE	TEST DESCRIPTION	IMPACT	STATUS

Edge Cases			
Large Input Size	Send a POST request with a significantly large payload and assess how the API manages this data without crashing or slow responses.	Medium	Failed
Malformed JSON	Send a POST request with incorrect JSON format and check if the API properly recognizes the format error and responds accordingly.	High	Failed
Empty POST Request	Submit a completely empty POST request to see how the API handles lack of data and ensure an appropriate error is generated.	High	Failed
Minimum Required Fields	Make a POST request with exactly the minimum required fields and verify the response is successful while containing the correct data.	Medium	Failed
Special Characters	Test the API by including special characters in the data fields to determine if it processes them correctly or returns errors.	Medium	Failed
Functional Tests			
Valid POST Request	Send a valid POST request to the API endpoint with required fields and ensure it responds with a success status and proper data format.	High	Failed
Exceeding Field Length	Send a POST request where text fields exceed the maximum character limit and confirm that the API rejects the input with an error.	Medium	Failed
Invalid Data Types	Submit a POST request with fields containing invalid data types and verify the API's response shows an error due to type validation failures.	Medium	Failed
Duplicate Entries	Post duplicate data entries and inspect the API's response to ensure it handles duplicates correctly, returning relevant error information.	Medium	Failed
Missing Required Fields	Attempt a POST request without essential fields and check that the API returns an appropriate error message indicating which fields are missing.	High	Failed

5 Test Execution Breakdown

Front Failed Test Details

Large Input Size

ATTRIBUTES	
Status	Failed
Priority	Medium
Description	Send a POST request with a significantly large payload and assess how the API manages this data without crashing or slow responses.

Test Code

```python
import requests
import json

def test_large_input_size():
    url = "http://localhost:5000"
    # Generate a large input payload (e.g., a large list of numbers)
    large_payload = {"data": list(range(1000000))}

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

test_large_input_size()
```

Error

HTTPConnectionPool(host='localhost', port=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7febddbf9970>: Failed to establish a new connection: [Errno 111] Connection refused'))

Large Input Size

```
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 0x7febddbf9970>: 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=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7febddbf9970>: Failed to establish a new connection: [Errno 111] Connection refused'))
```

```

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/var/task/main.py", line 60, in target
    exec(code, env)
  File "<string>", line 13, in <module>
  File "<string>", line 9, in test_large_input_size
  File "/var/task/requests/api.py", line 115, in post
    return request("post", url, data=data, json=json, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/api.py", line 59, in request
    return session.request(method=method, url=url, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/sessions.py", line 589, in request
    resp = self.send(prep, **send_kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/sessions.py", line 703, in send
    r = adapter.send(request, **kwargs)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/adapters.py", line 700, in send
    raise ConnectionError(e, request=request)
requests.exceptions.ConnectionError: HTTPConnectionPool(host='localhost', port=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7febddbf9970>: Failed to establish a new connection: [Errno 111] Connection refused'))

```

Cause

The API is not currently running or is unreachable on port 5000, resulting in a connection refusal error.

Fix

Ensure that the API server is started and actively listening on port 5000. Verify the server's configuration and firewall settings to allow incoming connections.

Malformed JSON

ATTRIBUTES

Status	Failed
Priority	High
Description	Send a POST request with incorrect JSON format and check if the API properly recognizes the format error and responds accordingly.

Test Code

```
import requests
import json

def test_malformed_json():
    url = "http://localhost:5000"
    # Malformed JSON example
    malformed_json = '{"key": "value",}'  # Extra comma

    response = requests.post(url, data=malformed_json, headers=
    {"Content-Type": "application/json"})

    print(f"Response Status Code: {response.status_code}")
    print(f"Response Text: {response.text}")

    # Simple check for response status
    assert response.status_code != 200, f"Expected non-200 status
    code, got {response.status_code}"

test_malformed_json()
```

Error

HTTPConnectionPool(host='localhost', port=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0×7f5f8414b050>: Failed to establish a new connection: [Errno 111] Connection refused'))

Trace

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 0x7f5f8414b050>: 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=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f5f8414b050>: Failed to establish a new connection: [Errno 111] Connection refused'))
```

```

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/var/task/main.py", line 60, in target
    exec(code, env)
  File "<string>", line 17, in <module>
  File "<string>", line 9, in test_malformed_json
  File "/var/task/requests/api.py", line 115, in post
    return request("post", url, data=data, json=json, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/api.py", line 59, in request
    return session.request(method=method, url=url, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/sessions.py", line 589, in request
    resp = self.send(prep, **send_kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/sessions.py", line 703, in send
    r = adapter.send(request, **kwargs)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/adapters.py", line 700, in send
    raise ConnectionError(e, request=request)
requests.exceptions.ConnectionError: HTTPConnectionPool(host='localhost', port=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f5f8414b050>: Failed to establish a new connection: [Errno 111] Connection refused'))

```

Cause

The API server is not running on localhost:5000, which leads to a connection refusal error.

Fix

Ensure the API server is properly started and listening on port 5000 before executing the test.

Valid POST Request

ATTRIBUTES

Status	Failed
Priority	High
Description	Send a valid POST request to the API endpoint with required fields and ensure it responds with a success status and proper data format.

Test Code

```python
import requests
import json

def test_valid_post_request():
    url = "http://localhost:5000"
    headers = {
        "Authorization": "Basic ."
    }
    payload = {
        "key": "value"
    }

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

    assert response.status_code == 200, f"Expected status code 200 but got {response.status_code}"
    assert 'expected_field' in response.json(), "Response JSON does not contain 'expected_field'"

test_valid_post_request()
```

Error

HTTPConnectionPool(host='localhost', port=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f6875825cd0>: Failed to establish a new connection: [Errno 111] Connection refused'))

Trace

Valid 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 0x7f6875825cd0>: 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=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f6875825cd0>: Failed to establish a new connection: [Errno 111] Connection refused'))
```

```

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/var/task/main.py", line 60, in target
    exec(code, env)
  File "<string>", line 19, in <module>
  File "<string>", line 13, in test_valid_post_request
  File "/var/task/requests/api.py", line 115, in post
    return request("post", url, data=data, json=json, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/api.py", line 59, in request
    return session.request(method=method, url=url, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/sessions.py", line 589, in request
    resp = self.send(prep, **send_kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/sessions.py", line 703, in send
    r = adapter.send(request, **kwargs)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/adapters.py", line 700, in send
    raise ConnectionError(e, request=request)
requests.exceptions.ConnectionError: HTTPConnectionPool(host='localhost', port=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f6875825cd0>: Failed to establish a new connection: [Errno 111] Connection refused'))

```

Cause

The API server is not running or is not accessible at the specified endpoint (localhost:5000).

Fix

Ensure that the API server is up and running on localhost:5000 before executing the test.

Exceeding Field Length

ATTRIBUTES	
Status	Failed
Priority	Medium
Description	Send a POST request where text fields exceed the maximum character limit and confirm that the API rejects the input with an error.

Test Code

```python
import requests
import json

def test_exceeding_field_length():
    url = "http://localhost:5000"
    credentials = "."

    # Example payload with an exceeding field length
    payload = {
        "field1": "a" * 256,  # Assuming the maximum length is 255
        "field2": "value2"
    }

    response = requests.post(url, json=payload, auth=(credentials,
    ''))

    # Print the response for debugging purposes
    print("Response Status Code:", response.status_code)
    print("Response Body:", response.text)

    # Check if the response status code indicates a failure
    assert response.status_code != 200, f"Expected a failure status
    code but got {response.status_code}"

test_exceeding_field_length()
```

Error

HTTPConnectionPool(host='localhost', port=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f9d637c2ae0>: Failed to establish a new connection: [Errno 111] Connection refused'))

Trace

Exceeding Field Length

```
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 0x7f9d637c2ae0>: 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=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f9d637c2ae0>: Failed to establish a new connection: [Errno 111] Connection refused'))
```

```

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/var/task/main.py", line 60, in target
    exec(code, env)
  File "<string>", line 23, in <module>
  File "<string>", line 14, in test_exceeding_field_length
  File "/var/task/requests/api.py", line 115, in post
    return request("post", url, data=data, json=json, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/api.py", line 59, in request
    return session.request(method=method, url=url, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/sessions.py", line 589, in request
    resp = self.send(prep, **send_kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/sessions.py", line 703, in send
    r = adapter.send(request, **kwargs)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/adapters.py", line 700, in send
    raise ConnectionError(e, request=request)
requests.exceptions.ConnectionError: HTTPConnectionPool
(host='localhost', port=5000): Max retries exceeded with url: /
(Caused by NewConnectionError('<urllib3.connection.HTTPConnection
object at 0x7f9d637c2ae0>: Failed to establish a new connection:
[Errno 111] Connection refused'))

```

Cause

The API server at localhost:5000 is not running or is not reachable, causing connection refusals when trying to send a request.

Fix

Ensure that the API server is up and running on localhost:5000. Check service configurations and restart the server if necessary.

Invalid Data Types

ATTRIBUTES

Status	Failed
Priority	Medium
Description	Submit a POST request with fields containing invalid data types and verify the API's response shows an error due to type validation failures.

Test Code

```python
import requests
import json

def test_invalid_data_types():
    url = "http://localhost:5000"
    headers = {
        "Authorization": "."
    }

    # Invalid data types example
    payload = {
        "integerField": "notAnInteger",  # should be an integer
        "stringField": 12345               # should be a string
    }

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

    print("Response Status Code:", response.status_code)
    print("Response JSON:", response.json())

    # Check if the response contains a key that indicates failure
    if 'error' in response.json():
        print(f"Error Message: {response.json()['error']}")

test_invalid_data_types()
```

Error

HTTPConnectionPool(host='localhost', port=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7fcfb0f56210>: Failed to establish a new connection: [Errno 111] Connection refused'))

Trace

</> 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 0x7fcfb0f56210>: 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=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7fcfb0f56210>: Failed to establish a new connection: [Errno 111] Connection refused'))
```

```

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/var/task/main.py", line 60, in target
    exec(code, env)
  File "<string>", line 25, in <module>
  File "<string>", line 16, in test_invalid_data_types
  File "/var/task/requests/api.py", line 115, in post
    return request("post", url, data=data, json=json, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/api.py", line 59, in request
    return session.request(method=method, url=url, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/sessions.py", line 589, in request
    resp = self.send(prep, **send_kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/sessions.py", line 703, in send
    r = adapter.send(request, **kwargs)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/adapters.py", line 700, in send
    raise ConnectionError(e, request=request)
requests.exceptions.ConnectionError: HTTPConnectionPool
(host='localhost', port=5000): Max retries exceeded with url: /
(Caused by NewConnectionError('<urllib3.connection.HTTPConnection
object at 0x7fcfb0f56210>: Failed to establish a new connection:
[Errno 111] Connection refused'))

```

Cause

The server at http://localhost:5000 is not running or is misconfigured, causing connection attempts to be refused.

Fix

Ensure that the API server is running on port 5000. Check the server configuration and any firewall settings that might be blocking access to this port.

Empty POST Request

ATTRIBUTES	
Status	Failed
Priority	High
Description	Submit a completely empty POST request to see how the API handles lack of data and ensure an appropriate error is generated.

Test Code

```python
import requests
import json

def test_empty_post_request():
    url = "http://localhost:5000"
    headers = {
        "Authorization": "Bearer ."
    }

    response = requests.post(url, headers=headers, json={})

    print("Response Status Code:", response.status_code)
    print("Response JSON:", response.json())

    assert response.status_code >= 200 and response.status_code < 300, f"Expected status code in 2xx range, got {response.status_code}"
    assert 'error' in response.json(), "Expected 'error' field in response JSON"

test_empty_post_request()
```

Error

HTTPConnectionPool(host='localhost', port=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f68ddb03ef0>: Failed to establish a new connection: [Errno 111] Connection refused'))

Trace

Empty 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 0x7f68ddb03ef0>: 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=5000): Max retries exceeded with url: /
(Caused by NewConnectionError('<urllib3.connection.HTTPConnection
object at 0x7f68ddb03ef0>: Failed to establish a new connection:
[Errno 111] Connection refused'))
```

```

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/var/task/main.py", line 60, in target
    exec(code, env)
  File "<string>", line 18, in <module>
  File "<string>", line 10, in test_empty_post_request
  File "/var/task/requests/api.py", line 115, in post
    return request("post", url, data=data, json=json, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/api.py", line 59, in request
    return session.request(method=method, url=url, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/sessions.py", line 589, in request
    resp = self.send(prep, **send_kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/sessions.py", line 703, in send
    r = adapter.send(request, **kwargs)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/adapters.py", line 700, in send
    raise ConnectionError(e, request=request)
requests.exceptions.ConnectionError: HTTPConnectionPool(host='localhost', port=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f68ddb03ef0>: Failed to establish a new connection: [Errno 111] Connection refused'))

```

Cause

The API server is not running or is inaccessible on localhost:5000, leading to connection refusal.

Fix

Ensure that the API server is running and listening on port 5000. Verify network configuration and firewall settings to allow connections to this port.

Duplicate Entries

ATTRIBUTES

Status	Failed
Priority	Medium
Description	Post duplicate data entries and inspect the API's response to ensure it handles duplicates correctly, returning relevant error information.

Test Code

```python
import requests
import json

def test_duplicate_entries():
    url = "http://localhost:5000"
    credentials = '.'

    headers = {
        'Authorization': f'Basic {credentials}'
    }

    # First entry submission
    response1 = requests.post(url, headers=headers, json={"data": "test_entry"})
    print("First submission response:", response1.text)

    # Submit the same entry again to check for duplicates
    response2 = requests.post(url, headers=headers, json={"data": "test_entry"})
    print("Second submission response:", response2.text)

    # Check if the second response indicates a duplicate entry
    assert 'duplicate' in response2.text.lower(), f"Expected duplicate response but got: {response2.text}"

test_duplicate_entries()
```

Error

HTTPConnectionPool(host='localhost', port=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7ff6365646b0>: Failed to establish a new connection: [Errno 111] Connection refused'))

Trace

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 0x7ff6365646b0>: 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=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7ff6365646b0>: Failed to establish a new connection: [Errno 111] Connection refused'))
```

```

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/var/task/main.py", line 60, in target
    exec(code, env)
  File "<string>", line 23, in <module>
  File "<string>", line 13, in test_duplicate_entries
  File "/var/task/requests/api.py", line 115, in post
    return request("post", url, data=data, json=json, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/api.py", line 59, in request
    return session.request(method=method, url=url, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/sessions.py", line 589, in request
    resp = self.send(prep, **send_kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/sessions.py", line 703, in send
    r = adapter.send(request, **kwargs)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/adapters.py", line 700, in send
    raise ConnectionError(e, request=request)
requests.exceptions.ConnectionError: HTTPConnectionPool
(host='localhost', port=5000): Max retries exceeded with url: /
(Caused by NewConnectionError('<urllib3.connection.HTTPConnection
object at 0x7ff6365646b0>: Failed to establish a new connection:
[Errno 111] Connection refused'))

```

Cause

The API server is not running or listening on port 5000, resulting in connection refused errors.

Fix

Ensure that the API server is up and running. Verify the server configuration to confirm it's set to listen on the correct port (5000) and that there are no firewall rules blocking the connection.

Minimum Required Fields

ATTRIBUTES	
Status	Failed
Priority	Medium
Description	Make a POST request with exactly the minimum required fields and verify the response is successful while containing the correct data.

Test Code

```python
import requests
import json

def test_minimum_required_fields():
    url = "http://localhost:5000"
    credentials = "."

    # Prepare the headers for public authentication
    headers = {
        "Authorization": f"Bearer {credentials}",
        "Content-Type": "application/json"
    }

    # Prepare the payload with minimum required fields
    payload = {
        # Add the minimum required fields based on API documentation
    }

    # Send the API request
    response = requests.post(url, headers=headers, data=json.dumps
    (payload))

    # Print the response for debugging
    print("Response Status Code:", response.status_code)
    print("Response Body:", response.text)

    # Vague check for response status
    assert response.status_code == 200, f"Expected status code 200
    but got {response.status_code} with response: {response.text}"

test_minimum_required_fields()
```

Error

HTTPConnectionPool(host='localhost', port=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f06e0603ef0>: Failed to establish a new connection: [Errno 111] Connection refused'))

Trace

Minimum 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 0x7f06e0603ef0>: 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=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f06e0603ef0>: Failed to establish a new connection: [Errno 111] Connection refused'))
```

```

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/var/task/main.py", line 60, in target
    exec(code, env)
  File "<string>", line 29, in <module>
  File "<string>", line 20, in test_minimum_required_fields
  File "/var/task/requests/api.py", line 115, in post
    return request("post", url, data=data, json=json, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/api.py", line 59, in request
    return session.request(method=method, url=url, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/sessions.py", line 589, in request
    resp = self.send(prep, **send_kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/sessions.py", line 703, in send
    r = adapter.send(request, **kwargs)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/adapters.py", line 700, in send
    raise ConnectionError(e, request=request)
requests.exceptions.ConnectionError: HTTPConnectionPool
(host='localhost', port=5000): Max retries exceeded with url: /
(Caused by NewConnectionError('<urllib3.connection.HTTPConnection
object at 0x7f06e0603ef0>: Failed to establish a new connection:
[Errno 111] Connection refused'))

```

Cause

The API server is not running or is not properly configured to listen on port 5000, resulting in a connection refused error when attempting to send a request.

Fix

Ensure the API server is started and is configured to listen on port 5000. Verify that any firewalls or network settings allow connections on that port.

Special Characters

ATTRIBUTES

Status	Failed
Priority	Medium
Description	Test the API by including special characters in the data fields to determine if it processes them correctly or returns errors.

Test Code

```
import requests
import json

def test_special_characters():
    url = "http://localhost:5000"
    credentials = "."

    # Prepare test data with special characters
    special_characters_data = {
        "field1": "!@#$%^&*()_+",
        "field2": "1234567890-=[]{};':\",.<>/?",
        "field3": "éèêëîïçà"
    }

    # Make a POST request with special characters
    response = requests.post(url, auth=(credentials, ""),
    json=special_characters_data)

    # Print the response for debugging purposes
    print("Response Status Code:", response.status_code)
    print("Response Body:", response.text)

    # Vague check for success or failure
    assert response.status_code in [200, 201], f"Expected status code
    in [200, 201], but got {response.status_code}"

test_special_characters()
```

Error

HTTPConnectionPool(host='localhost', port=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f00a53a5400>: Failed to establish a new connection: [Errno 111] Connection refused'))

Trace

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 0x7f00a53a5400>: 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=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7f00a53a5400>: Failed to establish a new connection: [Errno 111] Connection refused'))
```

```

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/var/task/main.py", line 60, in target
    exec(code, env)
  File "<string>", line 25, in <module>
  File "<string>", line 16, in test_special_characters
  File "/var/task/requests/api.py", line 115, in post
    return request("post", url, data=data, json=json, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/api.py", line 59, in request
    return session.request(method=method, url=url, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/sessions.py", line 589, in request
    resp = self.send(prep, **send_kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/sessions.py", line 703, in send
    r = adapter.send(request, **kwargs)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/adapters.py", line 700, in send
    raise ConnectionError(e, request=request)
requests.exceptions.ConnectionError: HTTPConnectionPool
(host='localhost', port=5000): Max retries exceeded with url: /
(Caused by NewConnectionError('<urllib3.connection.HTTPConnection
object at 0x7f00a53a5400>: Failed to establish a new connection:
[Errno 111] Connection refused'))

```

Cause

The API server is not running or listening on the specified port (5000). This may happen if the server process has not been started, crashed, or is bound to a different address.

Fix

Ensure that the API server is running and properly configured to listen on localhost:5000. Check server logs for any errors on startup, and make sure no firewall or network configurations are blocking connections to this port.

Missing Required Fields

ATTRIBUTES

Status	Failed
Priority	High
Description	Attempt a POST request without essential fields and check that the API returns an appropriate error message indicating which fields are missing.

Test Code

```python
import requests
import json

def test_missing_required_fields():
    url = "http://localhost:5000"
    headers = {
        "Authorization": "."
    }
    # Assuming the API has a POST method where missing required
    fields should be tested
    payload = {}  # Empty payload to simulate missing required fields
    response = requests.post(url, headers=headers, json=payload)

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

    assert response.status_code >= 400, f"Expected a client error
    status code, but got {response.status_code}"
    assert 'error' in response.json(), "Expected 'error' field in
    response JSON, but it was missing"

test_missing_required_fields()
```

Error

HTTPConnectionPool(host='localhost', port=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7fa3bc35faa0>: Failed to establish a new connection: [Errno 111] Connection refused'))

Trace

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 0x7fa3bc35faa0>: 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=5000): Max retries exceeded with url: / (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7fa3bc35faa0>: Failed to establish a new connection: [Errno 111] Connection refused'))
```

```

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/var/task/main.py", line 60, in target
    exec(code, env)
  File "<string>", line 19, in <module>
  File "<string>", line 11, in test_missing_required_fields
  File "/var/task/requests/api.py", line 115, in post
    return request("post", url, data=data, json=json, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/api.py", line 59, in request
    return session.request(method=method, url=url, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/sessions.py", line 589, in request
    resp = self.send(prep, **send_kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/sessions.py", line 703, in send
    r = adapter.send(request, **kwargs)
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/var/task/requests/adapters.py", line 700, in send
    raise ConnectionError(e, request=request)
requests.exceptions.ConnectionError: HTTPConnectionPool
(host='localhost', port=5000): Max retries exceeded with url: /
(Caused by NewConnectionError('<urllib3.connection.HTTPConnection
object at 0x7fa3bc35faa0>: Failed to establish a new connection:
[Errno 111] Connection refused'))

```

Cause

The API server is not running or is not reachable on localhost:5000, leading to connection refused errors.

Fix

Ensure that the API server is properly started and is listening on port 5000. Verify that there are no firewall settings blocking access to this port.

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
back	12	0 Pass/12 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

Back Execution Summary

TEST CASE	TEST DESCRIPTION	IMPACT	STATUS
Search, filtering and sorting under mixed dataset	Given many items in the system, When the user performs a search, applies multiple filters and changes sort order, Then the results match combined criteria, pagination updates, and deeplinking preserves search state. Verify empty-state messaging, and that filter resets behave correctly. Cleanup: remove any test-only items created for the test.	High	Failed
Checkout flow with sandbox payment and coupon application	Given items in cart and an authenticated user, When the user proceeds to checkout, applies a valid coupon, and completes payment via the sandbox gateway, Then order is created, confirmation email is sent, and order page shows correct totals. Verify declined card, invalid coupon, and partial failures show appropriate recovery UI and do not create orders. Cleanup: cancel/delete test order.	High	Failed
Localization, language switching and RTL rendering	Given the app supports multiple locales, When the user switches language in settings or via URL, Then static and dynamic copy updates, currency/date formats adjust, and RTL languages render correctly (mirrored layouts). Also verify persisted preference and fallback to default for missing translations.	Low	Failed
Password reset and invalid/expired token handling	Given a user requesting password reset, When they submit their email, Then the system sends a reset link. Given a valid reset token, When the user sets a new password, Then login with the new password succeeds. Also verify expired or tampered tokens show correct errors and do not allow password change. Cleanup: revert password if needed.	Medium	Failed
Form validation, large file upload and error handling	Given a multi-field form that accepts a file, When the user submits with valid inputs and a supported file within size limits, Then submission succeeds and file is stored. When user uploads an unsupported type or too-large file, Then client and server validation reject it with informative errors and form state remains recoverable. Verify progress UI for large uploads and cancel behavior.	Medium	Failed
User registration with email confirmation	Given a new user on the signup page, When the user submits valid registration details, Then the app creates the account, sends a confirmation email, and the confirmation link activates the account. Also verify invalid inputs show inline errors and the same email cannot be registered twice. Cleanup: delete test account via API.	High	Failed
Accessibility critical flows: keyboard nav and screen-reader labels	Given primary user flows (login, create resource, checkout), When tested via keyboard-only navigation and a screen-reader simulation, Then focus order is logical, interactive elements are reachable, ARIA labels/roles are present, and no essential content is inaccessible. Verify color-contrast on critical CTAs and that skip-links work.	Medium	Failed
End-to-end CRUD: create, view, edit, delete resource	Given an authenticated user on the resource list page, When the user creates a new resource via the UI, Then it appears in the list, can be opened to view details, edited (changes persist), and deleted (removed from list). Verify server-side validation and optimistic UI updates. Cleanup: ensure test resource is removed.	High	Failed
Offline usage and network reconnection sync	Given the app supports offline actions, When the client goes offline and user performs allowed actions (e.g., composing a draft), Then actions are queued locally, UI indicates offline status, and upon reconnection queued actions sync to server with conflict resolution. Verify failures during sync produce informative errors and preserve user data.	Medium	Failed
Main navigation and page routing	Given an authenticated or anonymous user on the homepage, When the user clicks each main nav item (header, footer and hamburger on mobile), Then the app navigates to the correct route, page content loads, the URL updates, and browser Back/Forward restore previous state. Verify deep links (direct URL) load the correct page. Cleanup: return to homepage.	High	Failed
Responsive behavior across breakpoints (mobile, tablet, desktop)	Given a set of key pages (homepage, product/list, dashboard), When the viewport is resized to common breakpoints, Then layout, navigation (hamburger vs full menu), modals, and interactive components adapt and remain usable. Verify touch interactions on the mobile breakpoint and that no content is clipped or inaccessible.	Low	Failed
User login, session persistence and logout	Given a registered user, When the user logs in with valid credentials, Then the user is redirected to the protected dashboard, auth tokens are stored, and the session persists across page reloads and new tabs. When the user logs out, Then tokens are cleared and protected pages redirect to login. Verify failed login shows proper error messages.	High	Failed

8 Test Execution Breakdown

Back Failed Test Details

Search, filtering and sorting under mixed dataset

ATTRIBUTES	
Status	Failed
Priority	High
Description	Given many items in the system, When the user performs a search, applies multiple filters and changes sort order, Then the results match combined criteria, pagination updates, and deeplinking preserves search state. Verify empty-state messaging, and that filter resets behave correctly. Cleanup: remove any test-only items created for the test.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/c418e498-c0f1-70c7-f517-ba15c38a476d/1763754311701488//tmp/d48f7856-3d8b-4fc6-9b69-bc6c0ec8c2f7/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:5000", 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())

```

Checkout flow with sandbox payment and coupon application

ATTRIBUTES

Status	Failed
Priority	High
Description	Given items in cart and an authenticated user, When the user proceeds to checkout, applies a valid coupon, and completes payment via the sandbox gateway, Then order is created, confirmation email is sent, and order page shows correct totals. Verify declined card, invalid coupon, and partial failures show appropriate recovery UI and do not create orders. Cleanup: cancel/delete test order.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/c418e498-c0f1-70c7-f517-ba15c38a476d/1763754312509411//tmp/7921af12-56b5-4d1d-a3b9-d675ffa01486/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:5000", 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())

```

Localization, language switching and RTL rendering

ATTRIBUTES

Status	Failed
Priority	Low
Description	Given the app supports multiple locales, When the user switches language in settings or via URL, Then static and dynamic copy updates, currency/date formats adjust, and RTL languages render correctly (mirrored layouts). Also verify persisted preference and fallback to default for missing translations.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/c418e498-c0f1-70c7-f517-ba15c38a476d/1763754312367908//tmp/e0dc7930-d4a0-40f1-bba3-75b16e3ae6ca/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:5000", 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
```

```python

            await asyncio.sleep(5)

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

asyncio.run(run_test())

```

Password reset and invalid/expired token handling

ATTRIBUTES

Status	Failed
Priority	Medium
Description	Given a user requesting password reset, When they submit their email, Then the system sends a reset link. Given a valid reset token, When the user sets a new password, Then login with the new password succeeds. Also verify expired or tampered tokens show correct errors and do not allow password change. Cleanup: revert password if needed.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/c418e498-c0f1-70c7-f517-ba15c38a476d/1763754312266239//tmp/d145dc5a-0a71-45a0-9398-92e98c6ca759/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:5000", 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
```

```python

            await asyncio.sleep(5)

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

asyncio.run(run_test())

```

Form validation, large file upload and error handling

ATTRIBUTES

Status	Failed
Priority	Medium
Description	Given a multi-field form that accepts a file, When the user submits with valid inputs and a supported file within size limits, Then submission succeeds and file is stored. When user uploads an unsupported type or too-large file, Then client and server validation reject it with informative errors and form state remains recoverable. Verify progress UI for large uploads and cancel behavior.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/c418e498-c0f1-70c7-f517-ba15c38a476d/1763754312311947//tmp/7bad9fea-e995-4f07-81a0-652033a472ba/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:5000", 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())

```

User registration with email confirmation

ATTRIBUTES

Status	Failed
Priority	High
Description	Given a new user on the signup page, When the user submits valid registration details, Then the app creates the account, sends a confirmation email, and the confirmation link activates the account. Also verify invalid inputs show inline errors and the same email cannot be registered twice. Cleanup: delete test account via API.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/c418e498-c0f1-70c7-f517-ba15c38a476d/1763754313151214//tmp/f9f79737-e47f-4dc7-b587-173d0fdfd98b/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:5000", 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 critical flows: keyboard nav and screen-reader labels

ATTRIBUTES

Status	Failed
Priority	Medium
Description	Given primary user flows (login, create resource, checkout), When tested via keyboard-only navigation and a screen-reader simulation, Then focus order is logical, interactive elements are reachable, ARIA labels/roles are present, and no essential content is inaccessible. Verify color-contrast on critical CTAs and that skip-links work.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/c418e498-c0f1-70c7-f517-ba15c38a476d/1763754312580457//tmp/269b0c4d-a4a4-45d1-a6b0-9c2641f02d1c/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:5000", 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
```

```python

            await asyncio.sleep(5)

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

    asyncio.run(run_test())

```

End-to-end CRUD: create, view, edit, delete resource

ATTRIBUTES

Status	Failed
Priority	High
Description	Given an authenticated user on the resource list page, When the user creates a new resource via the UI, Then it appears in the list, can be opened to view details, edited (changes persist), and deleted (removed from list). Verify server-side validation and optimistic UI updates. Cleanup: ensure test resource is removed.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/c418e498-c0f1-70c7-f517-ba15c38a476d/1763754312173639//tmp/b61fcdd6-48a5-4656-b449-e771be49e76f/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:5000", 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())

```

Offline usage and network reconnection sync

ATTRIBUTES

Status	Failed
Priority	Medium
Description	Given the app supports offline actions, When the client goes offline and user performs allowed actions (e.g., composing a draft), Then actions are queued locally, UI indicates offline status, and upon reconnection queued actions sync to server with conflict resolution. Verify failures during sync produce informative errors and preserve user data.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/c418e498-c0f1-70c7-f517-ba15c38a476d/1763754311028698//tmp/354087ff-33aa-47b6-a0b7-6818b2507096/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:5000", 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
```

```python

            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 and page routing

ATTRIBUTES

Status	Failed
Priority	High
Description	Given an authenticated or anonymous user on the homepage, When the user clicks each main nav item (header, footer and hamburger on mobile), Then the app navigates to the correct route, page content loads, the URL updates, and browser Back/Forward restore previous state. Verify deep links (direct URL) load the correct page. Cleanup: return to homepage.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/c418e498-c0f1-70c7-f517-ba15c38a476d/1763754312231691//tmp/851e0837-d50d-4ab3-9170-075d82e50820/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:5000", 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 behavior across breakpoints (mobile, tablet, desktop)

ATTRIBUTES	
Status	Failed
Priority	Low
Description	Given a set of key pages (homepage, product/list, dashboard), When the viewport is resized to common breakpoints, Then layout, navigation (hamburger vs full menu), modals, and interactive components adapt and remain usable. Verify touch interactions on the mobile breakpoint and that no content is clipped or inaccessible.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/c418e498-c0f1-70c7-f517-ba15c38a476d/1763754312186195//tmp/afebac82-2ecb-4ea6-a790-ccf1e85416c4/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:5000", 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
```

```python

            await asyncio.sleep(5)

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

asyncio.run(run_test())

```

User login, session persistence and logout

ATTRIBUTES

Status	Failed
Priority	High
Description	Given a registered user, When the user logs in with valid credentials, Then the user is redirected to the protected dashboard, auth tokens are stored, and the session persists across page reloads and new tabs. When the user logs out, Then tokens are cleared and protected pages redirect to login. Verify failed login shows proper error messages.
Preview Link	https://testsprite-videos.s3.us-east-1.amazonaws.com/c418e498-c0f1-70c7-f517-ba15c38a476d/1763754312161451//tmp/5b7f09f0-f070-4848-92fc-91bfceef859b/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:5000", 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())
```