Use Case Explanation Document

Project Name: Healthcare Benefits API Testing Framework

Purpose

The purpose of this framework is to automate the testing of RESTful API endpoints for a healthcare benefits management system. This framework ensures that the API's authentication mechanisms, benefits retrieval, and user preference updates are working correctly and efficiently. It also validates that the API adheres to security and performance standards.

Use Cases

1. User Authentication Testing

- Description: Test the API's user authentication functionality, including login and logout operations.
- Actors: Automation Engineer, API Developer
- Preconditions:
  - The API endpoint for authentication is available.
  - Valid and invalid user credentials are known.
- Steps:
  1. Send a POST request to the login endpoint with valid user credentials.
  2. Validate that the response status code is 200 and a valid authentication token is returned.
  3. Send a POST request to the login endpoint with invalid credentials.
  4. Validate that the response status code is 401 and an appropriate error message is returned.
  5. Use the valid authentication token to send a POST request to the logout endpoint.
  6. Validate that the response status code is 200 and the token is invalidated.
- Postconditions: The authentication mechanism is verified for both successful and unsuccessful scenarios.
- Exceptions:
  - Network failures
  - Server errors

2. Benefits Management Testing

- Description: Verify that the API correctly handles requests for retrieving and updating user benefits.
- Actors: Automation Engineer, API Developer

- Preconditions:
  - The user is authenticated and has a valid token.
  - The benefits management endpoints are available.
- Steps:
  1. Use the authentication token to send a GET request to the benefits endpoint.
  2. Validate that the response status code is 200 and the benefits information is returned in the correct format.
  3. Send a PUT request to update user preferences for benefits with valid data.
  4. Validate that the response status code is 200 and the preferences are updated successfully.
- Postconditions: User benefits information is retrieved and updated as expected.
- Exceptions:
  - Invalid token errors
  - Incorrect data formats

3. Security Testing

- Description: Ensure that unauthorized access to API endpoints is prevented and only authenticated users can access sensitive information.
- Actors: Security Engineer, Automation Engineer
- Preconditions:
  - The API has defined authentication and authorization mechanisms.
- Steps:
  1. Send a GET request to a protected endpoint without an authentication token.
  2. Validate that the response status code is 401 or 403 and an appropriate error message is returned.
  3. Attempt to update user benefits with an expired or invalid token.
  4. Validate that the response status code is 401 or 403.
- Postconditions: The API is secure and does not allow unauthorized access.
- Exceptions:
  - Incorrect error handling in the API

4. Performance Testing

- Description: Measure the API's response times to ensure it meets performance requirements.
- Actors: Performance Engineer, Automation Engineer
- Preconditions:
  - The API endpoints are available and accessible.
- Steps:

1. Send multiple requests to the API endpoints and measure response times.
2. Validate that the response times are within acceptable limits.
3. Generate performance reports for analysis.

- Postconditions: The API performs efficiently under normal and peak load conditions.
- Exceptions:
  - Server overloads
  - Latency issues

Assumptions

- The API documentation is available, detailing endpoint URLs, request/response formats, and authentication requirements.
- The testing environment is properly configured with access to the API.

Dependencies

- Java, Maven, RestAssured, JUnit, Mockito, and Allure for testing and reporting.
- Docker for containerized testing.
- CI/CD tools (e.g., Jenkins, GitHub Actions) for automated test execution.

Stakeholders

- API Developers: Need to ensure the API is functioning correctly and securely.
- Automation Engineers: Responsible for writing and maintaining automated test cases.
- Security Engineers: Interested in the security aspects of the API.
- Performance Engineers: Measure and ensure the API meets performance standards.