## Fluency QA Take Home Assessment

## Part 1: Test Plan

Author: Robert Gregory

### 1. Test Objective

The primary objective of this test plan is to ensure that the "Account Management" section of Fluency's Customer Facing API functions correctly, securely, and consistently to ensure a positive, reliable and seamless customer experience utilizing the API.

### 2. Scope

This test plan covers the following aspects of the "Account Management" section:

* Account creation and validation
* Updating account information
* Account synchronization

### 3. Test Cases

Test Case 1: Account Creation

* Verify that a new account can be successfully created.
* Verify that required fields (e.g., account name, contact information) are validated.
* Verify that invalid input data is rejected.

Test Case 2: Account Information Update

* Verify that account information (e.g., contact details) can be updated successfully.
* Verify that invalid updates (e.g., empty fields) are handled correctly.

Test Case 3: Account Synchronization

* Verify that the synchronization process works as expected.
* Confirm that data from the customer's systems is correctly integrated into Fluency’s system.

Test Case 4: Duplicate Account Creation

* Attempt to create an account with a name that already exists and verify that it is prevented.

Test Case 5: Unauthorized Access

* Verify that unauthorized access attempts are denied with the appropriate error message.
* Check the handling of invalid API tokens.

Test Case 6: Rate Limiting

* Test the API's rate-limiting mechanism by sending a high volume of requests in a short time.
* Ensure that rate limiting is effective in preventing excessive requests.

Test Case 7: Error Handling

* Verify that error responses (e.g., 4xx and 5xx status codes) are returned in response to invalid requests.
* Confirm that error messages are clear and informative.
* Verify that error messages are accurate based on scenario.

Test Case 8: Security Testing

* Perform security testing, including input validation and SQL injection attempts, to identify vulnerabilities.

### 4. Test Data

* Test data will include valid and invalid account information.
* Test data for account synchronization will include sample customer data.

### 5. Test Environments

* Testing will be conducted in a staging or test environment to avoid impacting production data.

### 6. Test Tools

* Utilize testing frameworks and tools for automation.
  + Postman - manual validation using collections, environment, and variables that can be shared amongst engineers and QA (and checked into GitHub)
  + Python / Pytest for API Test Automation
  + K6 for performance and load testing

### 7. Test Reports

* Detailed html test reports will be generated using automated tooling, and will include pass/fail status and any issues encountered.
* Defects will be captured with information around severity, criticality, and steps to retest. Screenshots will also be included, where applicable.

### 8. Test Automation

* Automated tests should be created based on the test plan, covering both functional and performance testing scenarios. These tests should include detailed assertions, data generation, and performance measurements.
* Performance testing scenarios should focus on the following:
  + Load Testing: Test the API's response time and throughput under different levels of concurrent users and transactions.
  + Stress Testing: Determine the breaking point and how the system behaves under extreme loads.
  + Scalability Testing: Evaluate the API's ability to scale horizontally or vertically.
  + Data Volume Testing: Assess and document performance with varying data volumes to ensure that the API can handle large datasets.
* Automated performance tests should provide clear metrics and analysis on how the API performs under various conditions, including response times, error rates, and resource utilization.