4/4/2025

Software Testing Document

Team CloudSmiths – Jaimin Savaliya, Jaynesh Bhandari, Krunal Savaj, Nithin Murthy, Rajath Gokhale

Team Member Document Review

Member Name	Signature or Initial	Reviewed on (Date)
Jaimin Savaliya	JS	4th April 2025
Jaynesh Bhandari	JB	4th April 2025
Krunal Savaj	KS	4th April 2025
Nithin Murthy	NM	4th April 2025
Rajath Gokhale	RG	4th April 2025

Document Revision History

Version	Changes	Date
1.0	First Draft	4th April 2025

Table of Contents

1 Project Description

The WinterGreen Healthcare Provider On-boarding application automates healthcare provider onboarding by:

- Integrating with EHR systems (e.g., Cerner, eCW or Athena health) via modular AWS Lambda functions.
- Storing structured provider data in AWS RDS and processed healthcare data in S3.
- Enabling scheduled data refreshes every 30 days.
- Ensuring IAM roles and secure credential management via AWS Secrets Manager.

Key Features:

- Serverless architecture for scalability.
- Configuration-driven EHR integration (no code changes needed for new systems).
- Step Functions for workflow orchestration.

2 Overall Testing Plan

2.1 Testing Approach:

- **Test-as-you-go:** Unit and integration tests during development; system/validation tests post-development.

- **Team Assignments:**
  - **Frontend Team:** UI/UX validation, form input testing, penetration testing.
  - **Backend Team:** Lambda function logic, EHR API integration.

- **Bias Avoidance:** Cross-team reviews (e.g., backend team tests frontend integration).

- **Testing Types:**
  - Static (code conventions, security checks).
  - Unit (Lambda functions, modules).
  - Integration (EHR ↔ Lambda ↔ HealthLake workflows).
  - System (end-to-end onboarding, stress testing).
  - Validation (client UAT).

3 Static Testing

3.2 Automated Static Analysis

Test ID	Scope	Method/Tool	Success Criteria
STC-001	Python Lambda Functions	Codiga + Pylint	Zero critical code smells; PEP8 compliance.
STC-002	Security Vulnerabilities	Bandit (Python)	No SQLi/XSS risks; secrets managed via AWS Secrets Manager.
STC-003	EHR Integration Configurations	Prospector (YAML/DB checks)	Valid EHR API endpoints; no hardcoded credentials.
STC-004	Data Flow (FHIR/JSON)	SonarQube	HealthLake transformations match FHIR standards.

3.3 Manual Reviews

Test ID	Scope	Method	Success Criteria
STC-005	Modularity & Separation of Concerns	Peer Code Walkthrough	Frontend Lambdas do not contain backend business logic.
STC-006	Error Logging Design	Technical Review	Logs include timestamps, error codes, and actionable messages.
STC-007	EHR Configuration Database	Database Schema Review	EHR Systems table includes **api_endpoint**, **auth_type**, **refresh_interval**.
STC-008	User Documentation	UX Team Audit	Onboarding forms align with HIPAA data entry guidelines.

3.4 Testing Approach

1. **Automated Tools Configuration**
   - Integrate Codiga into CI/CD pipelines to scan code during commits.
2. **Manual Review Process**
   - **Weekly peer reviews** using checklist:
     - IAM policies follow least-privilege principle.
     - Lambda functions are stateless/idempotent.
     - EHR configs support adding new systems via database (no code changes).
   - **Formal inspections** by security team for:
     - Secrets Manager usage for EHR API credentials.
3. **Healthcare-Specific Checks**
   - Validate FHIR resource mappings in HealthLake (e.g., **Patient ↔ Provider** entities).

3.5 Traceability Matrix

Test ID	SDD Requirements	Search Result Best Practices
STC-001	NFR-02 (Security)	Automated PEP8 checks + security scans.
STC-003	FR-02 (EHR Integration)	Validate EHR configs via YAML/DB reviews.

3.6 Expected Defects & Mitigation

- **Hardcoded Secrets (Critical):**
  - **Mitigation:** Enforce Secrets Manager usage; block commits with **AWS_ACCESS_KEY** patterns via Git hooks.

4 UNIT TESTS

4.2 TEST CASES

- **Test Case 1:** Provider Management Lambda
- **Test ID:** UT-001
- **Requirements ID:** FR-01 (Provider Input), NFR-02 (Security)
- **Tester:** Backend Team
- **Environment:** Local AWS SAM CLI with mocked RDS instance.
- **Goal:** Validate CRUD operations for provider onboarding.
- **Testing Procedure:**
  - Mock valid/invalid provider data (e.g., missing NPI number).
  - Execute create_provider() with valid inputs; verify RDS entry.
  - Execute delete_provider() and confirm record removal.
- **Expected Results:**
  - Valid inputs persist in RDS; invalid inputs return 400 errors.
  - Deleted providers are archived, not fully removed

Test Case 2: EHR Integration Lambda

- **Test ID:** UT-002
- **Requirements ID:** FR-02 (EHR Data Fetch), NFR-01 (Performance)
- **Tester:** Frontend Team
- **Environment:** Isolated Lambda with mocked EHR API (Cerner sandbox).
- **Goal:** Verify secure EHR data retrieval.
- **Testing Procedure:**
  - Mock Secrets Manager credentials and EHR API responses.
  - Execute fetch_ehr_data() with valid/invalid credentials.
  - Measure response time for EHR API call.
- **Expected Results:**
  - Valid credentials return FHIR-formatted data in <15s.
  - Invalid credentials trigger CloudWatch alerts.

Test Case 3: Data Processing Lambda

- **Test ID:** UT-003
- **Requirements ID:** FR-03 (Data Processing), NFR-03 (Interoperability)
- **Tester:** Frontend Team
- **Environment:** Local HealthLake instance with sample EHR data.
- **Goal:** Validate FHIR-to-JSON normalization.
- **Testing Procedure:**
  - Feed raw FHIR data from mocked EHR response.
  - Execute normalize_healthlake_data().
  - Compare output to FHIR schema validator.
- **Expected Results:**
  - Output matches FHIR Practitioner resource schema.
  - Invalid FHIR data triggers S3 quarantine.

4.3 Traceability Matrix

Test ID	Functional Reqs	Non-Functional Reqs
UT-001	FR-01	NFR-02
UT-002	FR-02	NFR-01
UT-003	FR-03	NFR-03

4.4 Tools & Framework

- **Python: pytest**, **unittest**, **moto** (AWS mocking).
- **Coverage:** Aim for ≥80% branch coverage.

5 Integration Tests

Approach: Bottom-up (test Lambda functions first, then Step Function workflows).

5.2 Test Cases

Test Case 1: EHR Data Fetch

- **Test ID:** IT-001
- **Requirements ID:** FR-02 (EHR Data Fetch), NFR-02 (Security)
- **Tester:** Backend Team + Frontend Team
- **Environment:** Mocked AWS Lambda environment with Secrets Manager and Cerner API sandbox.
- **Goal:** Validate secure credential retrieval and API call functionality.
- **Testing Procedure:**
  - Mock valid and invalid credentials in AWS Secrets Manager.
  - Execute **fetch_ehr_data()** Lambda function to retrieve data from the Cerner sandbox API.
  - Monitor network traffic using AWS X-Ray to ensure encryption during data transmission.
  - Verify error handling for invalid credentials or API timeouts.
- **Expected Results:**
  - Valid credentials successfully retrieve FHIR-formatted data from the Cerner API.
  - Invalid credentials trigger error logs and CloudWatch alerts without exposing sensitive data.

Test Case 2: HealthLake Sync

- **Test ID:** IT-002
- **Requirements ID:** FR-03 (Data Processing), NFR-03 (Interoperability)
- **Tester:** Backend Team + Frontend Team
- **Environment:** Mocked AWS Lambda environment with HealthLake and S3 integration.
- **Goal:** Ensure raw EHR data is transformed into FHIR format and stored securely in S3.
- **Testing Procedure:**
  - Feed mocked raw EHR data into the **normalize_healthlake_data()** Lambda function.
  - Monitor HealthLake processing logs for successful transformation into FHIR-compliant JSON objects.
  - Verify that transformed data is stored in S3 with proper encryption (AWS KMS).
  - Test error handling by feeding invalid or corrupted EHR data into the function.
- **Expected Results:**
  - Valid EHR data is normalized to FHIR format and securely stored in S3 with appropriate metadata tags.
  - Invalid or corrupted data triggers quarantine procedures, storing it in a separate S3 bucket for review.

Test Case 3: Onboarding Flow

- **Test ID:** IT-003
- **Requirements ID:** FR-01 (Provider Input), FR-04 (30-Day Data Refresh), NFR-01 (Performance)
- **Tester:** Backend Team + Frontend Team
- **Environment:** AWS Step Functions orchestrating all Lambda functions in a production-like setup.
- **Goal:** Verify end-to-end provider onboarding workflow, including scheduled data refreshes every 30 days.
- **Testing Procedure:**
  - Submit provider details via the frontend form, triggering the Step Function workflow.
  - Validate that provider details are stored in RDS and linked to corresponding EHR configurations in Secrets Manager.
  - Trigger manual EHR data fetch through the workflow and verify data normalization/storage in S3 via HealthLake integration.
  - Simulate a 30-day refresh by manually advancing system time and validating automatic workflow execution using EventBridge rules.
  - Measure response times for each step to ensure performance requirements are met (<45s per request).
- **Expected Results:**
  - Provider onboarding completes successfully, storing all required details in RDS and fetching normalized EHR data into S3 within acceptable time limits (<45s).
  - Scheduled refreshes execute automatically every 30 days without manual intervention, updating S3 timestamps accordingly.

5.3 Traceability Matrix

Test ID	Functional Reqs	Non-Functional Reqs	Modules Involved	Goal
IT-001	FR-02	NFR-02	EHR Lambda ↔ Secrets Manager	Validate secure credential retrieval & API call functionality.
IT-002	FR-03	NFR-03	Data Processing Lambda ↔ S3	Ensure raw EHR data is transformed into FHIR format and stored securely in S3.
IT-003	FR-01, FR-04	NFR-01	Step Function ↔ All Lambdas	Verify end-to-end provider onboarding workflow, including scheduled refreshes every 30 days.

6 Validation Tests

6.2 Requirements Review

6.2.1 Functional Requirements Validation

Requirement ID	Test Scenario	Method	Success Criteria
FR-01	Provider onboarding form input	Black-box testing	Data persists in RDS with no errors.
FR-02	EHR data fetch via API	API automation (Postman)	Data retrieved matches EHR records.
FR-03	30-day data refresh	Scheduled job simulation	S3 timestamps update automatically.

6.2.2 Non-Functional Requirements Validation

Requirement ID	Test Scenario	Method	Success Criteria
NFR-01	100 concurrent onboarding requests	Load testing (Locust)	Latency ≤45s; 0% request failure.
NFR-02	Security	Security scan (Checkov)	No unencrypted data; IAM roles configured.
NFR-03	EHR system interoperability	EHR sandbox testing	Data flows seamlessly to HealthLake.

6.3 Client Evaluation

User Acceptance Testing (UAT)

- **Participants:** System Administrators.

- **Scenarios:**
  1. Complete end-to-end provider onboarding.
  2. Manually trigger EHR data refresh.
  3. Simulate edge cases (e.g., invalid EHR credentials).

- **Success Metric:** ≥90% satisfaction rate in post-test surveys37.

6.4 Functional & Non-Functional Test Cases

Test Case 1: Secure EHR Data Transmission

- **Test ID:** VT-001
- **Requirements ID:** FR-02 (Functional), NFR-02 (Non-functional)
- **Test Scenario:** Validate encrypted EHR data transfer between Lambda and Cerner API.
- **Procedure:**
  1. Trigger EHR data fetch.
  2. Capture network traffic via AWS X-Ray.
  3. Verify TLS 1.2+ encryption and Secrets Manager usage.
- **Expected Results:** No plaintext credentials or PHI exposed.

Test Case 2: Usability of Onboarding Forms

- **Test ID:** VT-002
- **Requirements ID:** FR-01 (Functional), NFR-03 (Non-functional)
- **Test Scenario:** Ensure forms are intuitive for healthcare staff.
- **Procedure:**
  1. Conduct heuristic evaluation (Nielsen's principles).
  2. Measure task completion time for 10 providers.
- **Expected Results:** Average task time <5 minutes; ≤2 UI redesign iterations.

Test Case 3: System Recovery After Failure

- **Test ID:** VT-003
- **Requirements ID:** NFR-01 (Non-functional)
- **Test Scenario:** Validate data integrity during AWS Lambda cold starts.
- **Procedure:**
  1. Force Lambda timeouts during EHR sync.
  2. Check S3 for partial/incomplete files.
- **Expected Results:** Transaction rollback; error logged in CloudWatch.

6.5 Traceability Matrix for Functional/Non-Functional Reqs

Test ID	Functional Reqs	Non-Functional Reqs
VT-001	FR-02	NFR-02
VT-002	FR-01	NFR-03
VT-003	–	NFR-01

6.6 Additional Test Cases for Frontend

Test Case 1: Dropdown Menu Functionality

- **Test ID:** VT-004
- **Requirements ID:** FR-01 (Provider Input), NFR-03 (Usability)
- **Tester:** Frontend Team
- **Environment:** Web browser (Chrome, Firefox, Edge) in a production-like setup using AWS Amplify-hosted frontend.
- **Goal:** Validate that dropdown menus display correct options and allow selection without errors.
- **Testing Procedure:**
  - Navigate to the onboarding form page.
  - Click on dropdown menus for "Provider Type" and "EHR System."
  - Verify that all expected options are displayed.
  - Select each option and ensure the selection is saved correctly in the form data.
  - Test edge cases like clicking outside the dropdown or selecting invalid options.
- **Expected Results:**
  - Dropdown menus display all expected options without missing or duplicated entries.
  - Selected options are retained in the form data and submitted correctly to the backend.

Test Case 2: Error Pop-Up Validation

- **Test ID:** VT-005
- **Requirements ID:** FR-01 (Provider Input), NFR-02 (Security)
- **Tester:** Frontend Team
- **Environment:** Web browser with simulated invalid inputs using developer tools or testing frameworks like Selenium.
- **Goal:** Ensure error pop-ups are triggered appropriately for invalid inputs and display meaningful messages.
- **Testing Procedure:**
  - Submit the onboarding form with invalid inputs (e.g., missing NPI number, incorrect email format).
  - Verify that error pop-ups are displayed immediately after submission attempts.
  - Check that error messages are clear and specific to the invalid field(s).
  - Test edge cases like submitting an empty form or exceeding input character limits.
- **Expected Results:**
  - Error pop-ups appear for invalid inputs with clear messages specifying the issue (e.g., "NPI number is required").
  - Pop-ups disappear after correcting the input and resubmitting the form.

Test Case 3: Success Pop-Up Validation

- **Test ID:** VT-006
- **Requirements ID:** FR-01 (Provider Input), FR-02 (EHR Data Fetch)
- **Tester:** Frontend Team
- **Environment:** Production-like web environment with mocked backend responses for successful submissions.
- **Goal:** Validate that success pop-ups appear after successful form submissions and provide confirmation details to users.
- **Testing Procedure:**
  - Submit the onboarding form with valid inputs (e.g., provider name, NPI number, EHR system).
  - Mock a successful backend response for data submission using testing tools like Postman or Jest mocks.
  - Verify that a success pop-up appears with confirmation details (e.g., "Provider successfully onboarded!").
  - Test edge cases like rapid consecutive submissions to ensure pop-up behavior remains consistent.
- **Expected Results:**
  - Success pop-up appears immediately after successful submission with accurate confirmation details.
  - The pop-up disappears automatically after a set time or upon user action (e.g., clicking "Close")

Test Case 4: Search Functionality on Data History List

- **Test ID:** VT-007
- **Requirements ID:** FR-04 (Data Refresh), NFR-03 (Interoperability)
- **Tester:** Frontend Team
- **Environment:** Production-like environment with populated data history.
- **Goal:** Validate that users can search the Data History list by provider name, EHR system, or date range.
- **Testing Procedure:**
  - Enter valid search terms (e.g., provider name "Dr. Smith") and verify results.
  - Test partial matches (e.g., "Smi" for "Smith").
  - Enter invalid terms (e.g., special characters like **#@!**).
  - Leave the search field empty and click "Search."
- **Expected Results:**
  - Valid searches return accurate results.
  - Partial matches display relevant entries.
  - Invalid/empty searches show a "No results found" message.

Test Case 5: Filtering on Data History List

- **Test ID:** VT-008
- **Requirements ID:** FR-04 (Data Refresh), NFR-03 (Interoperability)
- **Tester:** Frontend Team
- **Environment:** Data History list with 50+ entries.
- **Goal:** Ensure filters (e.g., EHR system type, refresh status) refine results correctly.
- **Testing Procedure:**
  - Apply a single filter (e.g., EHR system = "Cerner").
  - Apply multiple filters (e.g., EHR system = "Cerner" + status = "Completed").
  - Click "Reset Filters" and verify default view.
- **Expected Results:**
  - Filters reduce results to matching entries.
  - Multiple filters combine logically (AND condition).
  - Reset returns the list to its unfiltered state.

Test Case 6: CRUD Operations via UI

- **Test ID:** VT-009
- **Requirements ID:** FR-01 (Provider Input), NFR-02 (Security)
- **Tester:** Frontend Team
- **Environment:** AWS Amplify frontend with mocked backend.
- **Goal:** Validate CRUD actions reflect in the UI and database.
- **Testing Procedure:**
  - **Create:** Add a new provider and verify the entry appears in the UI and RDS.
  - **Read:** Open the provider's details page and confirm data matches input.
  - **Update:** Edit the provider's contact info and verify UI/database updates.
  - **Delete:** Archive a provider and confirm removal from the active list.
- **Expected Results:**
  - UI updates immediately after CRUD actions.
  - Database reflects changes without data corruption.

Test Case 7: Search Filters

- **Test ID:** VT-010
- **Requirements ID:** FR-02 (EHR Data Fetch), NFR-01 (Performance)
- **Tester:** Frontend Team
- **Environment:** Search page with 10+ EHR records.
- **Goal:** Validate date-range and EHR-type filters under load.
- **Testing Procedure:**
  - Filter records by date range (e.g., "Jan 2025 – Mar 2025").
  - Combine date range + EHR system type (e.g., "eClinicalWorks").
  - Measure response time for complex queries.
- **Expected Results:**
  - Filters return results in <5s even with large datasets.
  - Pagination works correctly (e.g., 10 results per page).

Test Case 8: Input Validation for CRUD Forms

- **Test ID:** VT-011
- **Requirements ID:** FR-01 (Provider Input), NFR-02 (Security)
- **Tester:** Frontend Team
- **Environment:** Provider onboarding form.
- **Goal:** Prevent SQLi/XSS attacks via form inputs.
- **Testing Procedure:**
  - Enter malicious scripts (e.g., **<script>alert('test')</script>**) in text fields.
  - Attempt SQLi payloads (e.g., **' OR 1=1 --**).
- **Expected Results:**
  - Inputs are sanitized; scripts/SQLi payloads are blocked.
  - Error pop-up displays: "Invalid characters detected".

6.7 Traceability Matrix for Frontend Validation Tests

Test ID	Functional Reqs	Non-Functional Reqs
VT-004	FR-01	NFR-03
VT-005	FR-01	NFR-02
VT-006	FR-01, FR-02	–
VT-007	FR-04	NFR-03
VT-008	FR-04	NFR-03
VT-009	FR-01	NFR-02
VT-010	FR-02	NFR-01
VT-011	FR-01	NFR-02

6.8 Tools & Metrics

- **Functional Validation:** TestSigma (automated workflows), Postman (API testing).
- **Non-Functional Validation:**
  - JMeter (performance), OWASP ZAP (security), Heatmaps (UI/UX).
- **Metrics:** Defect density (<0.1 defects/KLOC), mean time to recover (<10min).

7 System Tests

7.2 TEST CASES

Test Case 1:

- **Test ID:** ST-001
- **Time/Date:** [To be filled]
- **Requirements ID:** FR-01 (Provider Input), NFR-02 (Security)
- **Tester:** Backend Team
- **Environment:** AWS Production-like environment (Amplify + RDS + S3).
- **Goal:** Validate secure provider onboarding with EHR data fetch.
- **Procedure:**
  - Submit provider details via UI.
  - Trigger EHR data fetch manually.
  - Verify data is stored in S3 and RDS.
- **Expected Results:** No unencrypted data; logs show no errors.

Test Case 2:

- **Test ID:** ST-002
- **Time/Date:** [To be filled]
- **Requirements ID:** FR-04 (Data Refresh)
- **Tester:** Backend Team
- **Environment:** AWS Cloud with scheduled EventBridge rule.
- **Goal:** Ensure 30-day data refresh executes automatically.
- **Procedure:**
  - Manually advance system clock to trigger refresh.
  - Verify HealthLake processes updated EHR data.
- **Expected Results:** S3 object timestamps update; no duplicates.

Test Case 3:

- **Test ID:** ST-003
- **Time/Date:** [To be filled]
- **Requirements ID:** NFR-01 (Performance)
- **Tester:** Backend Team
- **Environment:** Load-balanced Lambda functions.
- **Goal:** Confirm 100 concurrent onboarding requests are handled in <45s.
- **Procedure:** Use Locust to simulate traffic.
- **Expected Results:** Latency <45s; no failed requests.