# Lab 6.1 Report: Fitbit Authentication Integration

Course: IoMT-Based Stress Monitoring System  
Student: Miltone  
Institution: Michigan Technological University  
Instructor: Dr. Ronghua Xu  
Duration: Week 6  
Lab Title: Fitbit Authentication and Token Management Integration  
Objective: To implement secure Fitbit OAuth 2.0 authentication and token refresh mechanisms using AWS Lambda, DynamoDB, and API Gateway, enabling persistent communication between the Fitbit API and the IoMT backend.

## 1. Introduction

Following the successful Fitbit data retrieval in Lab 5, this lab focused on establishing a secure and automated authentication process for Fitbit API access. The key goal was to implement a dynamic token management system that eliminates manual token generation, ensuring uninterrupted Fitbit data communication. This was achieved through AWS Lambda functions that handle both initial authentication and automatic token refresh, integrated with DynamoDB for secure token storage.

## 2. Objectives

* Develop AWS Lambda functions for Fitbit OAuth 2.0 authentication and token refresh.
* Store access and refresh tokens securely in DynamoDB.
* Integrate API Gateway for remote invocation of authentication and refresh functions.
* Test token generation and refresh using Postman and AWS Console.
* Prepare for future integration with the mobile Flutter app for user Fitbit login.

## 3. System Architecture Overview

The Fitbit authentication workflow involves four core AWS components: Lambda, DynamoDB, API Gateway, and IAM roles. Below is a simplified architecture of the implemented system:

1. \*\*Fitbit Authorization\*\* – User grants permission to access Fitbit data via OAuth 2.0 authorization code flow.  
2. \*\*AWS Lambda (fitbit\_auth\_lambda)\*\* – Exchanges authorization code for access and refresh tokens.  
3. \*\*DynamoDB (fitbitToken table)\*\* – Stores tokens with user identifiers and expiration timestamps.  
4. \*\*AWS Lambda (fitbit\_refresh\_lambda)\*\* – Automatically refreshes tokens using the refresh token before expiration.  
5. \*\*EventBridge Scheduler\*\* – Triggers refresh function periodically to maintain valid authentication.

## 4. Implementation and Development Steps

1. Stage 1: Fitbit App Configuration

Configured Fitbit Developer App to include the following parameters:  
- Redirect URI: https://yourapiurl.com/dev/oauth\_callback  
- Scopes: heartrate, activity, profile, sleep, settings  
- OAuth Type: Authorization Code Grant  
Fitbit returned client\_id and client\_secret which were stored securely in AWS Systems Manager Parameter Store.

1. Stage 2: Lambda Function for Token Generation (fitbit\_auth\_lambda)

A Python 3.12 Lambda function was created to exchange the authorization code for access and refresh tokens using Fitbit’s token endpoint. This function used the 'requests' library for HTTPS POST requests. Tokens were parsed from Fitbit’s JSON response and stored in DynamoDB.

1. Stage 3: DynamoDB Table Creation

A DynamoDB table named 'fitbitToken' was created with 'user\_id' as the primary key. Each record contained the following fields:  
- access\_token  
- refresh\_token  
- expires\_in  
- last\_updated  
- user\_id

1. Stage 4: Lambda Function for Token Refresh (fitbit\_refresh\_lambda)

A second Lambda function was developed to refresh expired tokens automatically. It read the stored refresh token from DynamoDB, sent a POST request to Fitbit’s token endpoint, and updated DynamoDB with the new access token.

1. Stage 5: API Gateway Integration and Testing

API Gateway endpoints were created for both authentication and refresh functions. The endpoints were deployed under the stage 'auth\_dev'. Postman was used to simulate authorization code submission and token refresh requests.

1. Stage 6: EventBridge Scheduling

Configured AWS EventBridge to trigger the 'fitbit\_refresh\_lambda' every 4 hours to maintain active tokens. This ensured continuous backend access to Fitbit data without manual intervention.

## 5. Testing and Validation

|  |  |
| --- | --- |
| Component | Validation Result |
| Lambda Function (fitbit\_auth\_lambda) | Successfully exchanged authorization code for valid tokens. |
| DynamoDB Storage | Tokens stored with correct user\_id and expiration timestamps. |
| Lambda Function (fitbit\_refresh\_lambda) | Automatically refreshed tokens using refresh token flow. |
| EventBridge | Triggered token refresh function every 4 hours as scheduled. |
| API Gateway | Endpoints responded with HTTP 200 and returned valid JSON payloads. |

## 6. Challenges and Resolutions

|  |  |  |
| --- | --- | --- |
| Challenge | Description | Resolution |
| Authorization Code Expiry | Fitbit authorization codes expired quickly during manual testing. | Implemented immediate exchange of code for tokens upon receipt. |
| Invalid Token Format | Token endpoint returned malformed JSON due to encoding issue. | Used base64 encoding for client credentials and verified content type headers. |
| Lambda Timeout | Function execution exceeded default 3-second timeout. | Extended timeout to 30 seconds to handle network delays. |
| Refresh Token Rotation | Old refresh tokens invalidated before new token saved. | Updated DynamoDB immediately after each successful refresh. |

## 7. Outcome and Deliverables

* Fitbit authentication Lambda functions deployed and tested successfully.
* DynamoDB table configured for secure token storage and automatic updates.
* API Gateway endpoints verified through Postman and AWS testing.
* EventBridge automation confirmed via scheduled token refreshes.
* Prepared Flutter integration plan for future OAuth login integration.

## 8. Reflection

This lab established the heartbeat of secure communication between Fitbit and AWS. By automating token lifecycle management, the system achieved true autonomy—capable of maintaining valid authentication without human involvement. The implementation deepened understanding of OAuth 2.0 principles, security best practices, and event-driven architectures.

“Automation isn’t just convenience—it’s continuity. In this lab, the system learned to sustain its own pulse.”

## 9. Forward Outlook

The next lab (6.2) extends this authentication system by incorporating Fitbit data fetch and synchronization capabilities. Using the valid access tokens managed in this lab, the system will automatically retrieve heart rate and stress data at regular intervals, completing the IoMT data loop.

✅ Lab 6.1 Successfully Completed. Fitbit authentication and token management were automated through AWS services, ensuring secure and continuous IoMT data access.