# Lab 5 Report: Fitbit API Backend Integration

Course: IoMT-Based Stress Monitoring System  
Student: Miltone  
Institution: Michigan Technological University  
Instructor: Dr. Ronghua Xu  
Duration: Week 5  
Lab Title: Integration of Fitbit API into AWS Lambda Backend  
Objective: To connect the AWS backend with the Fitbit API using secure authentication and data retrieval techniques, enabling access to wearable device metrics such as heart rate and activity levels.

## 1. Introduction

This lab focused on expanding the AWS backend by integrating Fitbit’s Web API. The goal was to establish secure communication between the serverless AWS environment and Fitbit’s cloud platform using OAuth 2.0 authentication. This integration allowed the IoMT system to retrieve real-time biometric data—particularly heart rate and activity metrics—which would later feed into the stress monitoring logic.

## 2. Objectives

By the end of this lab, the following objectives were achieved:

* Register the IoMT application on the Fitbit Developer Platform to obtain client credentials.
* Implement OAuth 2.0 for secure access token generation and management.
* Develop AWS Lambda functions in Python to fetch heart rate and activity data from Fitbit API.
* Integrate Lambda with API Gateway to expose REST endpoints for the Flutter app.
* Test and validate Fitbit API communication using Postman and AWS console.

## 3. Fitbit Developer Account Setup

The Fitbit Developer Portal (https://dev.fitbit.com/apps) was used to register a new application. The app was configured with a redirect URI for OAuth authorization flow and assigned a unique client ID and client secret. These credentials were securely stored in AWS Systems Manager for use in Lambda functions.

Fitbit’s OAuth 2.0 framework requires an access token for each user session. This token enables the application to access Fitbit resources such as heart rate, steps, and sleep metrics through authorized API calls.

## 4. AWS Lambda Development and Integration

1. Stage 1: Lambda Function Creation

A new Lambda function named 'fitbitDataFetcher' was created using Python 3.12 runtime. The function utilized the 'requests' library to send HTTPS GET requests to Fitbit endpoints.

1. Stage 2: OAuth Token Handling

Access tokens were generated through Fitbit’s authorization server. The Lambda function included headers for Authorization using the Bearer token format. A refresh mechanism was added to handle token expiration, ensuring continuous data access.

1. Stage 3: DynamoDB Data Storage

Lambda was configured to store retrieved Fitbit heart rate and activity data in a DynamoDB table named 'FitbitData'. Each record included user ID, collection timestamp, and heart rate metrics.

1. Stage 4: API Gateway Integration

An API Gateway REST endpoint was configured to trigger the Lambda function. The endpoint supported GET requests for Fitbit data retrieval. Deployed under the stage 'fitbit\_dev', this endpoint enabled secure access from the mobile application.

## 5. Fitbit API Data Retrieval

Fitbit API endpoints were tested to verify correct data extraction. The following URLs were utilized:

• Heart Rate: https://api.fitbit.com/1/user/-/activities/heart/date/today/1d.json  
• Activity Summary: https://api.fitbit.com/1/user/-/activities/date/today.json  
• Profile: https://api.fitbit.com/1/user/-/profile.json

The Lambda function successfully retrieved JSON responses from Fitbit’s cloud servers. Example response snippet:

{'activities-heart': [{'dateTime': '2025-10-18', 'value': {'restingHeartRate': 68}}]}

## 6. Testing and Validation

Testing was conducted using Postman and the AWS Console. The workflow validated secure token exchange and proper data flow through AWS infrastructure.

|  |  |
| --- | --- |
| Component | Validation Result |
| Lambda Function Test | Successfully fetched heart rate data with valid JSON response. |
| API Gateway | Responded with HTTP 200 status, returning Fitbit data payload. |
| DynamoDB Storage | Heart rate records stored with correct timestamp and structure. |
| Token Refresh | Automatically refreshed expired tokens through OAuth endpoint. |
| Error Handling | Implemented retry mechanism for 401 Unauthorized responses. |

## 7. Challenges and Resolutions

|  |  |  |
| --- | --- | --- |
| Challenge | Description | Resolution |
| OAuth Authorization Complexity | Difficulties handling Fitbit’s authorization code grant flow. | Used Fitbit’s API Explorer and Postman to manually test and verify token requests. |
| Token Expiration | Access tokens expired every 8 hours. | Implemented refresh token logic and EventBridge scheduling for automatic renewal. |
| Permission Scopes | Insufficient API scopes prevented access to heart rate data. | Updated app permissions in Fitbit Developer Console to include 'heartrate' and 'activity'. |
| DynamoDB Data Formatting | Inconsistent timestamp formats caused retrieval errors. | Standardized all timestamps to ISO 8601 format during Lambda processing. |

## 8. Outcome and Deliverables

* Fitbit Developer account registered and configured with OAuth 2.0 credentials.
* Lambda function developed to fetch Fitbit data using secure HTTPS requests.
* API Gateway endpoint successfully deployed for external data access.
* DynamoDB integration validated through data persistence tests.
* Comprehensive test logs and screenshots submitted via AWS Console and Postman.

## 9. Reflection

This lab represented the heartbeat of the IoMT ecosystem—the direct connection between wearable technology and cloud analytics. Through Fitbit API integration, the project advanced from a theoretical design to a system capable of collecting real physiological data. This experience reinforced the principles of authentication, API management, and secure data handling in digital health systems.

“Each pulse of data retrieved was more than a number—it was proof that technology can listen to the human body.”

## 10. Forward Outlook

With Fitbit data successfully integrated, the next lab (6.1) focuses on automating token management and integrating Fitbit authentication directly into the mobile Flutter interface. This step will complete the end-to-end connectivity between the user, their wearable device, and the cloud infrastructure.

✅ Lab 5 Successfully Completed. Fitbit API is now fully connected to the AWS backend, enabling real-time health data flow into the IoMT system.