# Lab 6.2 Report: Fitbit Data Access and Automation

Course: IoMT-Based Stress Monitoring System  
Student: Miltone  
Institution: Michigan Technological University  
Instructor: Dr. Ronghua Xu  
Duration: Week 7  
Lab Title: Fitbit Data Access, Storage, and Automation in AWS  
Objective: To develop automated AWS Lambda workflows for fetching Fitbit data, storing it securely in DynamoDB, and maintaining continuous synchronization through EventBridge scheduling.

## 1. Introduction

Building upon the authentication system established in Lab 6.1, this lab focused on full-scale Fitbit data access and automation. The goal was to implement AWS Lambda functions that fetch, process, and store Fitbit heart rate data in DynamoDB, while ensuring periodic synchronization using AWS EventBridge. This marks the completion of the IoMT backend pipeline, enabling end-to-end communication between the user’s wearable device, the cloud infrastructure, and the mobile application.

## 2. Objectives

* Create AWS Lambda functions to retrieve and store Fitbit data using valid access tokens.
* Integrate the functions with DynamoDB to record heart rate and activity data.
* Automate data synchronization using EventBridge scheduling rules.
* Develop corresponding Flutter functions to request Fitbit data through API Gateway.
* Test and validate the complete data flow from Fitbit API to DynamoDB and Flutter frontend.

## 3. System Architecture

The final architecture integrates all previously configured AWS services into a continuous data loop. The workflow follows this sequence:

1. Fitbit → Fitbit API → AWS Lambda (getHeartRate\_lambda)  
2. Lambda → API Gateway → Flutter Mobile App (data request)  
3. Lambda → DynamoDB (saveHeartRate\_lambda for storage)  
4. EventBridge → Periodic Lambda triggers (automatic data updates)

This structure ensures that user Fitbit data is automatically retrieved, securely stored, and made available to the mobile client without manual refresh operations.

## 4. Implementation Steps

1. Stage 1: Lambda Function for Data Retrieval (getHeartRate\_lambda)

Developed a Python 3.12 Lambda function that sends a GET request to Fitbit’s heart rate API endpoint. The function used the stored access token from DynamoDB and fetched data in JSON format. The retrieved data included time-series heart rate readings and resting heart rate values.

1. Stage 2: Lambda Function for Data Storage (saveHeartRate\_lambda)

A second Lambda function was developed to parse and save Fitbit data into DynamoDB. Each record was tagged with a user\_id, date, and heart rate metrics. The data schema ensured easy retrieval for future analysis and visualization.

1. Stage 3: DynamoDB Schema Update

The 'FitbitData' table was updated with the following attributes:  
- user\_id (Partition Key)  
- timestamp (Sort Key)  
- heart\_rate  
- steps  
- collection\_source (Lambda/API)  
This design allowed the database to support historical tracking and prevent duplicate entries.

1. Stage 4: EventBridge Scheduling

AWS EventBridge was configured to trigger the getHeartRate\_lambda function every 6 hours. This automation ensured that the backend continually synchronized data without user interaction.

1. Stage 5: Flutter Integration and Data Display

A Flutter function was developed using Dart’s http and FutureBuilder classes to call the API Gateway endpoint. The data was rendered in a dashboard-like interface, displaying the latest heart rate, activity summary, and collection timestamp.

## 5. Testing and Validation

The system was tested across all layers: Lambda, API Gateway, DynamoDB, and Flutter interface.

|  |  |
| --- | --- |
| Component | Validation Result |
| Lambda (getHeartRate\_lambda) | Successfully retrieved Fitbit data using access token. |
| Lambda (saveHeartRate\_lambda) | Data stored in DynamoDB with unique user identifiers. |
| API Gateway | Endpoints responded with correct Fitbit JSON payloads (HTTP 200). |
| EventBridge | Triggered Lambda functions on schedule without manual intervention. |
| Flutter Client | Displayed real-time heart rate data fetched from API Gateway. |

## 6. Challenges and Resolutions

|  |  |  |
| --- | --- | --- |
| Challenge | Description | Resolution |
| Expired Access Tokens | EventBridge triggers failed when tokens expired. | Integrated token refresh logic from Lab 6.1 before each API call. |
| Lambda Timeout | Large Fitbit data caused Lambda function to time out. | Extended timeout limit to 60 seconds and optimized JSON parsing. |
| Duplicate Entries | Repeated triggers caused data redundancy in DynamoDB. | Implemented conditional write using DynamoDB ‘PutItem’ with unique timestamp checks. |
| Data Parsing Errors | Fitbit API returned nested JSON structures. | Used json.loads and dictionary comprehension to extract key-value pairs correctly. |

## 7. Outcome and Deliverables

* Lambda functions developed and deployed for Fitbit data retrieval and storage.
* DynamoDB table updated and validated for accurate data persistence.
* EventBridge scheduling successfully automated backend operations.
* API Gateway endpoints accessible to Flutter for live data fetching.
* Flutter dashboard displayed synchronized Fitbit health metrics.

## 8. Reflection

This lab represented the realization of a fully operational IoMT data ecosystem. For the first time, the system operated autonomously—fetching, processing, and storing real-time biometric data. It exemplified the essence of the Internet of Medical Things: constant, intelligent, and secure monitoring. Through AWS automation, the backend gained self-sufficiency, bridging human physiology with machine intelligence.

“When the system began to collect data on its own, it stopped being a project—and started becoming a living organism of information.”

## 9. Forward Outlook

The next step beyond this lab is full application integration and testing. The Flutter app will incorporate visualization tools and analytical features to interpret heart rate and stress data. In the final project phase, these datasets will be analyzed and compared to user self-reports, completing the feedback loop of stress monitoring.

✅ Lab 6.2 Successfully Completed. The Fitbit data pipeline is fully automated and synchronized, enabling real-time IoMT health monitoring.