# Final Project Report Outline

Project Title: IoMT-Based Stress Monitoring System using Fitbit and AWS Cloud Integration  
Student: Miltone  
Institution: Michigan Technological University  
Instructor: Dr. Ronghua Xu  
Course: IoMT-Based Application Development  
Duration: Weeks 1–8

## 1. Abstract

The IoMT-Based Stress Monitoring System integrates wearable sensor data with cloud computing technologies to monitor user stress levels in real-time. Using Fitbit’s API, AWS serverless services, and a Flutter-based mobile app, the system automates data collection, storage, and synchronization. This project demonstrates the intersection of mobile health informatics, IoMT, and cloud infrastructure, building a robust, scalable platform for continuous stress assessment.

## 2. Introduction

### 2.1 Project Background

Stress is a measurable physiological state that impacts overall health. With advancements in the Internet of Medical Things (IoMT), wearable sensors can continuously collect biometric data. This project leverages Fitbit data and AWS infrastructure to automate the monitoring process, linking human physiology with cloud intelligence.

### 2.2 Project Objectives

* Establish secure Fitbit API integration using AWS backend.
* Develop a Flutter mobile client for user interaction and visualization.
* Automate data synchronization and cloud-based storage.

### 2.3 Scope

This project covers the frontend mobile development (Flutter), backend AWS configuration (Lambda, DynamoDB, API Gateway), Fitbit API integration, and data automation using EventBridge. Predictive modeling and diagnostic analysis are outside the current project scope.

## 3. System Overview

### 3.1 System Description

The IoMT system captures heart rate data through Fitbit devices, processes it using AWS Lambda, stores it in DynamoDB, and displays it via a Flutter mobile interface. It also collects user-reported stress levels for comparison.

### 3.2 System Architecture

The system follows a layered structure:  
• Frontend: Flutter mobile app for UI and user input.  
• Backend: AWS Lambda for data processing.  
• Database: DynamoDB for token and data storage.  
• Automation: EventBridge for scheduled sync.  
• Integration: Fitbit API using OAuth 2.0 authentication.

### 3.3 Architecture Diagram

Fitbit → Fitbit API → AWS Lambda → DynamoDB ↔ Flutter App  
 ↑  
 EventBridge

## 4. System Design and Implementation

1. Lab 1 – IDE Setup: Configured Flutter SDK, Android Studio, and emulator testing.
2. Lab 2 – Figma UI Design: Created prototypes and wireframes for all app screens.
3. Lab 3 – Frontend Development: Implemented Flutter app structure and navigation logic.
4. Lab 4 – AWS Service Setup: Configured DynamoDB, Lambda, API Gateway, and IAM roles.
5. Lab 5 – Fitbit API Backend Integration: Connected AWS Lambda with Fitbit API using OAuth 2.0.
6. Lab 6.1 – Authentication Integration: Automated token refresh and secure Fitbit login.
7. Lab 6.2 – Data Fetch and Automation: Automated Fitbit data retrieval, storage, and synchronization.

## 5. Testing and Validation

Testing was performed across AWS, Fitbit API, and Flutter environments using Postman, AWS Console, and Android emulator.

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Test Description | Expected Result | Status |
| Lambda Functions | Fetch heart rate data from Fitbit | Valid JSON response | ✅ |
| DynamoDB | Store and retrieve Fitbit data | Accurate record retrieval | ✅ |
| API Gateway | Enable mobile-to-cloud communication | HTTP 200 success | ✅ |
| EventBridge | Auto-trigger data sync events | Log successful triggers | ✅ |

## 6. Results and Discussion

The project achieved seamless communication between Fitbit’s cloud API and AWS Lambda. Data was fetched, refreshed, and stored successfully. Performance testing indicated low latency (<500ms per API call) and high reliability. The Flutter interface accurately displayed real-time heart rate data.

## 7. Challenges and Solutions

|  |  |  |
| --- | --- | --- |
| Challenge | Description | Resolution |
| Token Expiration | Fitbit tokens expired every 8 hours. | Automated refresh using EventBridge triggers. |
| Duplicate Entries | Repeated triggers caused redundant writes. | Implemented conditional DynamoDB writes. |
| Navigation Errors | Inconsistent Flutter routing. | Used named routes and pushReplacement(). |
| API Timeout | Large JSON payloads caused delay. | Extended Lambda timeout and optimized parsing. |

## 8. Reflection

This project unified multiple domains: cloud computing, mobile app development, and digital health informatics. It strengthened my understanding of serverless architecture and wearable device integration. Beyond technical achievement, it symbolized how technology can monitor and support human wellness.

“In teaching the system to listen to the heartbeat, I learned how deeply technology can echo human rhythm.”

## 9. Future Work

* Integrate AWS SageMaker for stress prediction models.
* Expand support for additional wearables (Apple Watch, Garmin).
* Develop advanced data visualization and alert notifications.
* Deploy system through AWS Amplify for live users.

## 10. Conclusion

The IoMT-Based Stress Monitoring System successfully demonstrated a real-time, autonomous data ecosystem linking human physiological signals with cloud infrastructure. The system now stands as a blueprint for scalable, secure, and intelligent digital health monitoring applications.

## 11. References

* AWS Documentation: https://docs.aws.amazon.com
* Fitbit Developer Portal: https://dev.fitbit.com
* Flutter Documentation: https://docs.flutter.dev
* OAuth 2.0 Standard: https://oauth.net/2/
* Figma Documentation: https://help.figma.com

## 12. Appendices

• Source Code (Flutter, AWS Lambda)  
• API Gateway Configuration  
• DynamoDB Schema Design  
• Screenshots (Flutter UI, Postman API Tests)  
• Demo Video Transcript