Software Design Document

for

Diabetic Patient Monitoring

Version 2.0 approved

Prepared by:
Christopher Fong
Koenrad Macbride
Yosep Kim
Andrew Padilla
Wun Woo

Medtronic / Alex Zhong

April 14, 2018

Table of Contents

Table of Contents **2**

Revision History **3**

1. Introduction **4**

1.1 Purpose 4

1.2 Intended Audience and Reading Suggestions 4

1.3 System Overview 4

2. Design Considerations **5**

2.1 Assumptions and Dependencies 5

2.2 Goals and Guidelines 5

2.3 Development Methods 5

3. Architectural Strategies **6**

4. System Architecture **7**

4.1 - Level 0 DFD 7

4.2 - Level 1 DFD 7

4.3 - Database Schema 9

4.4 - Detailed descriptions 10

5. Policies and Tactics **12**

5.1 Technology in current RTD Framework 12

Appendix A: Glossary **13**

Revision History

Name	Date	Reason For Changes	Version
et. all	7/12/17	Document Creation	1.0
K. MacBride	4/10/17	Dependency Changes	1.1
A. Padilla	4/10/17	Architecture Changes	1.1
Y .Kim	4/14/17	Technology in RTD Framework	1.2

1. Introduction

1.1 Purpose

The purpose of this document is three-fold:

a) Completely define a full set of requirements for the this project
b) Completely define the design for this project.
c) Define and partially implement feasible modules for this project

1.2 Intended Audience and Reading Suggestions

This document is intended for future Medtronic developers and CSULA senior design students.

1.3 System Overview

The purpose of this project is to obtain insight on glucose level based on activity level, and find any correlation between them. In addition, to come up with a solution to predict glucose levels primarily using FitBit metrics in order to reduce reliance on expensive sensors. In doing so we will dynamically process data and display information to patients or authorized people.

2. Design Considerations

2.1 Assumptions and Dependencies

Dependencies		
#	**Dependency**	**Owner**
1	Web Browser w/JavaScript	End User
2	Web-App Front End	CSULA Team
3	Real Time Data Feed	Medtronic

2.2 Goals and Guidelines

- Finding insight/relationship between physical activity and glucose levels and visualizing results
- Investigating the possibility of predicting continuous glucose level based on physical activity and blood glucose samples

2.3 Development Methods

This project is being conducted using Agile development. The development process is formal with document and code reviews.

- Time series Research
- API Gateway and API endpoint creation

Data Manipulation/Machine Learning Process

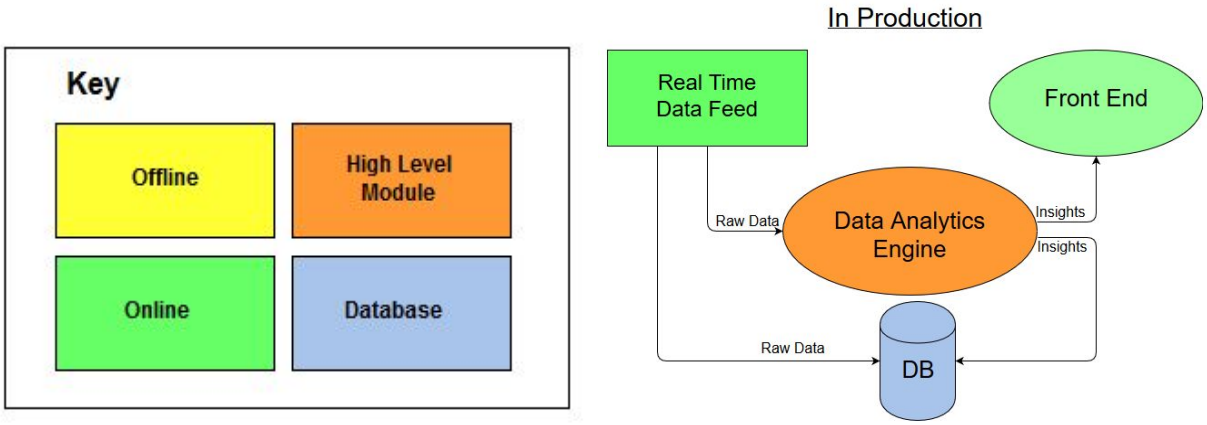
- Back-end Python Machine Learning Structure and Code
- Data preprocessing and data cleaning
    - Store raw glucose and fitness data
    - store trained data set
- Sample data feature extraction
    - Prepare data for machine learning algorithm
- Feature selection and dimensionality reduction
- Knowledge extraction/ data analytics, deriving insights
- Data testing and training
- Continuous data processing

3. Architectural Strategies

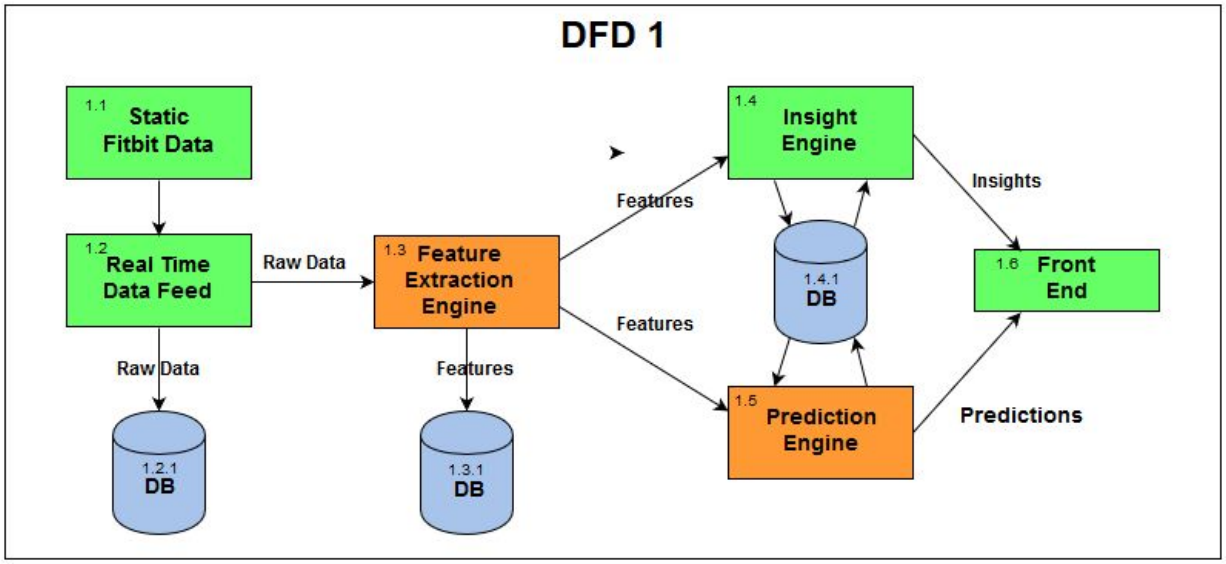
The real time data stream will be handled by Apache Kafka, which will send the data stream to Apache Storm for data analysis using python data analysis libraries. Storm will send the analysis to a Flask front end for viewing by the end-user. Each module is a Microservice and can be run independently from each other Microservice. These microservices can be easily scaled.

4. System Architecture

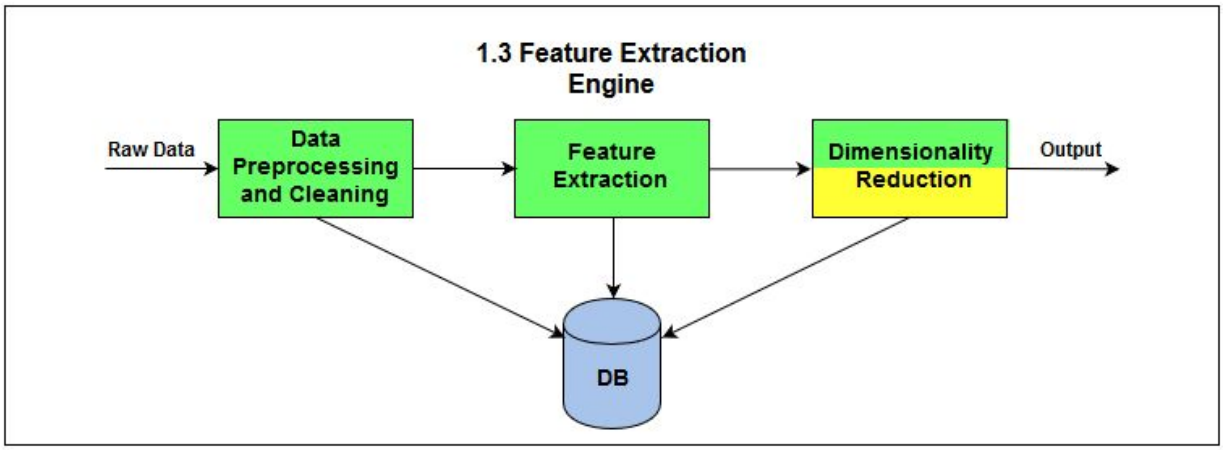
4.1 - Level 0 DFD



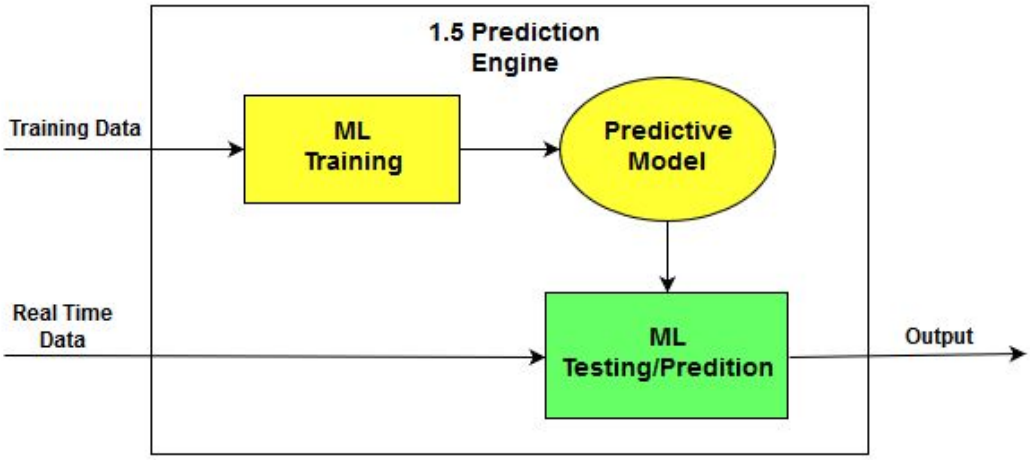
4.2 - Level 1 DFD



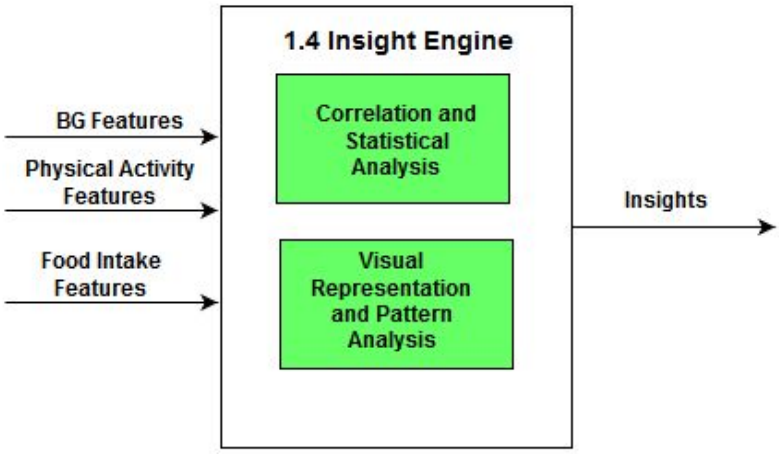
4.2.1 Feature Extraction Engine



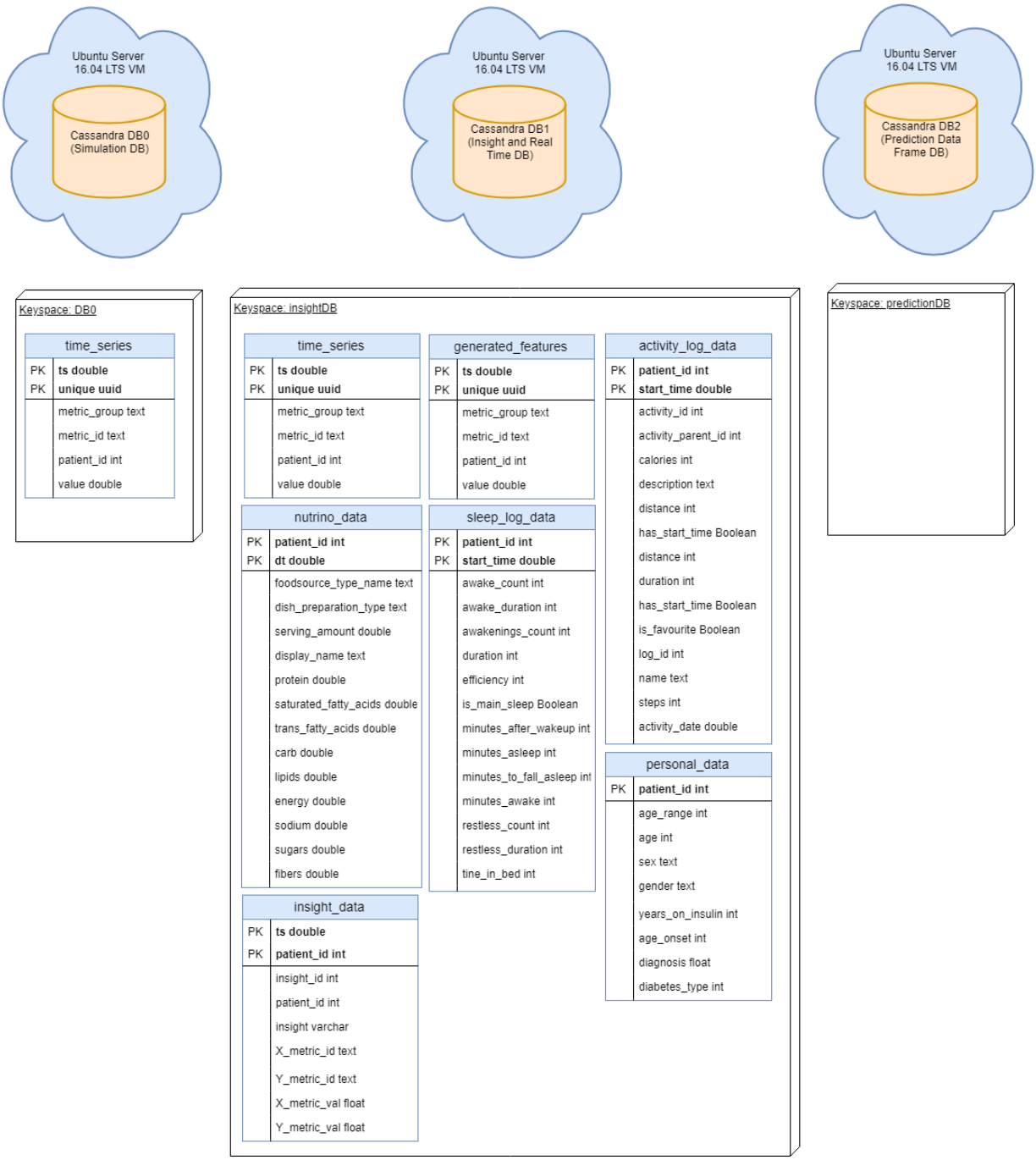
4.2.2 - Prediction Engine



4.2.3 - Insight Engine



4.3 - Database Schema



4.4 - Detailed descriptions

Major functional subunits shown in the DFDs in the previous subsections, are described in detail below. The section numbering below corresponds to DFD diagrams in section 4.2.

1.1 Static FitBit Data

FitBit analytics are received from a static directory in production.

1.2 Real Time Data Feed

This module handles the real time data. This includes continuous glucose readings, food intake, and manual glucose readings.

1.2.1 - Database for storing Raw Data as DataPackets
1.2.2 - Real Time Data Feed
1.2.3 - Cassandra Database shall be used to store incoming data as DataPackets
1.2.4 - Apache Kafka shall receive data stream from fitbit
1.2.5 - Apache Kafka shall receive data stream from SG
1.2.6 - Apache Kafka shall create a producer for streaming to Apache Storm
1.2.7 - Apache Storm shall split the stream by feature
1.2.8 - Apache Storm shall send processed data to Feature Extraction Engine (1.3)

1.3 Feature Extraction Engine

The feature extraction engine will extract features, use feature selection and dimensionality reduction. These extracted features will be sent to modules 1.4 and 1.5.

1.3.1 - Data shall be received from Apache Storm.
1.3.1 - Python script shall analyze and sort data
1.3.2 - Pandas shall handle features of real time data
1.3.3 - Database shall store extracted features

1.4 Insight Engine

The insight engine will make insights.

1.4.1 - Python script shall find features to develop insights from the real time data
1.4.2 - Pandas shall handle good values for correlations of different features.
1.4.3 - Database shall store insights for use with Flask.

1.5 Prediction Engine

The prediction engine shall take training data, and use it to produce a predictive model. It then uses the predictive model on real time data to produce predictions.

1.5.1 - Data shall be received from Apache Storm.
1.5.2 - RNN shall be implemented using TensorFlow to create predictions
1.5.3 - Database shall store predictions.

1.6 Front End

The front end will display all of our results after our insight and prediction data is put together.

1.6.1 - Gentelella Flask shall display webpages
1.6.2 - ChartsJs shall display charts
1.6.3 - Data is streamed from the Prediction Engine
1.6.4 - Data is streamed from the Insight Engine

5. Policies and Tactics

The team shall use SCRUM developing methods

5.1 Technology in current RTD Framework

Technologies Used		
Name	**Version**	**Source**
Apache Kafka	1.0.0	http://kafka.apache.org/
Apache Storm	1.1.1	http://storm.apache.org/
Apache Cassandra	3.11.1	http://cassandra.apache.org/
Python	2.7.14	http://python.org/
Flask	0.12.2	http://flask.pocoo.org/
Flask Gentelella	1.4	https://github.com/afourmy/flask-gentelella
Charts.js (gentelella)	2.0	https://www.chartjs.org/
Amazon Web Services	----	https://aws.amazon.com/
TensorFlow	r1.4	https://www.tensorflow.org/
JavaScript	1.8.5	https://www.javascript.com/
SciKit-Learn	0.19.1	http://scikit-learn.org/
Pandas	0.210	https://pandas.pydata.org/
HTML5	5.2	https://www.w3.org/TR/html5/
JUpyter Notebook	5.4.1	http://jupyter.org/

Appendix A: Glossary

SDD	Software Design Document
BG	Blood Glucose Level
CGM	Continuous Glucose Monitoring
SG	Sensor Glucose Data from Real Time Glucose Monitor
RNN	Recurrent Neural Network
API	Application Programming Interface