**Design Document**

**for**

**AIMS**

(Anesthesia Information Management System)

Prepared by Rohit\_sharma41

Organization: Infosys

Table of Contents

1. Introduction
2. System Overview

* High Level Description
* Technology Stack

1. Technical Approach

* Tools

1. PGI AIMS application flow
2. Addition of a new FXML screen
3. Add a new Device Parameter

1. Introduction

The purpose of this document is to outline the design for PGI AIMS Application. This will include a view of the high level architecture as well as the breakdown of the internal subsystems. Class

and sequence diagrams will be provided to show how the system will be put together and how

data will flow through the system. There will also be discussion on the technologies that we will

be using throughout this project. This document provides an outline of the user interface to

demonstrate how it will be formatted.

2. System Overview

**2.1 High Level Description**

The Anesthesia Information Management system (AIMS) being developed is a specialized form of an electronic health record (EHR) system that will allow automatic and reliable collection, as well as storage and presentation of patient data during pre-operative/ intra-operative and post-operative periods.

This system is used to capture Patients’ vital information from various medical devices in operation theatres (Patient Monitors, Ventilators & Infusion Pumps) and end it over server to ease the process of remote view. AIMS eradicated the manual process of recording patient health parameters during live surgeries by completely automating the entire procedure and minimizing the events of human errors.

**2.2 Technology Stack**

View – JavaFx, FXML, CSS

Model – POJO

Server – Apache Kafka

3. Technical Approach

**3.1. Tools**

There are a number of tools and technologies that this project will utilize for development.

● Server Side

Java 1.6 environment

Apache Kafka framework

Eclipse Neon

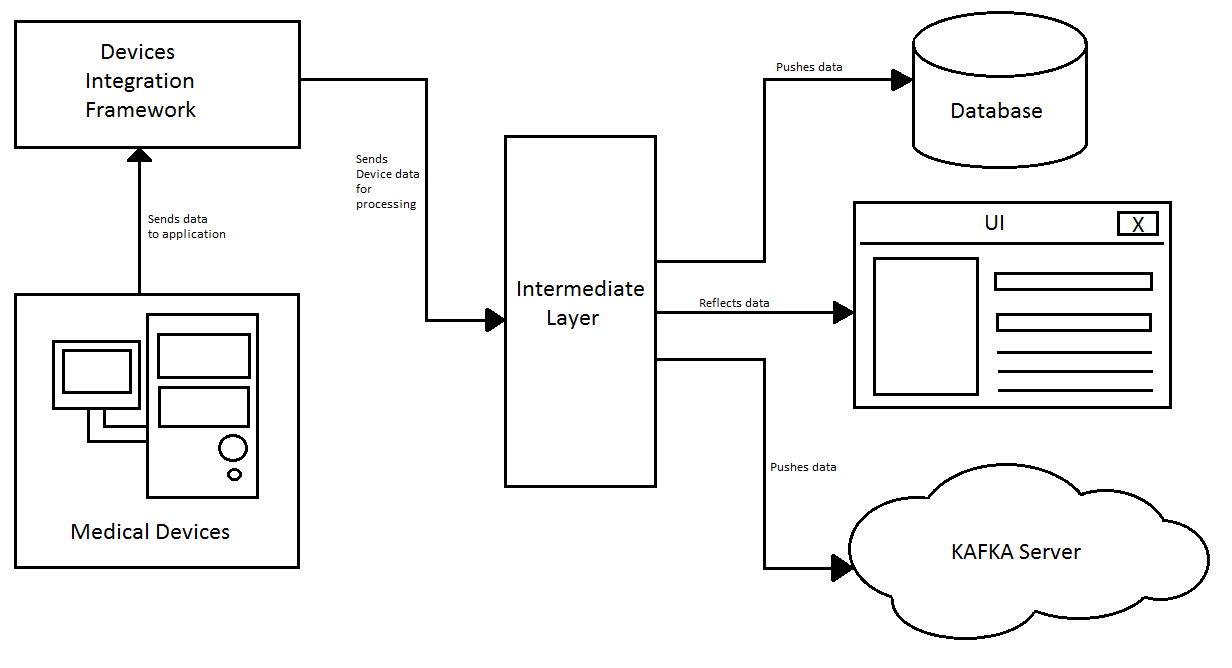
Git repository (on Github)

● Client Side

FXML from JavaFx

CSS

4. PGI AIMS application flow



Various medical devices such as Patient Monitors, Infusion Pumps etc. are connected to the system via Belkin/Prolific cross cable connectors. Our AIMS system collects this data in raw format and Device Integration Framework parse this byte data sets into readable strings and integers.

The formatted data is then passed onto Intermediate Layer where it is processed and made presentable in form of graphical and grid components. The integer values for various vital parameters are reflected in the cells of grid components available on UI and graphical plots are also generated.

Intermediate Layer processes and transfer data to three end points:

* Database: The formatted data is pushed into database which can be used later in historical graph screens, historical grid views and scenarios of unwanted application fails.
* UI: The data is also processed to fit into UI components to make it easily readable by the end user in form of graphs and grids.
* KAFKA server: This data is also pushed to KAFKA to ease the process of remote view at a distant location.

Additionally, UI offers modification in the live data, once recorded from the medical devices.

5. Addition of a new FXML screen

* Generate a FXML using SceneBuilder utility with desired components.
* Create a controller class for the generated FXML which will handle events associated with different components pres3ent in that FXML.
* Controller class implemetns Initializable interface and implements its initialize() method. Whenever FXML is loaded, the control first comes under this initialize() method.
* Mention this controller class under Controllers section on lower left side of the SceneBuilder utility for respective FXML.

6. Add a new Device Parameter

*1) ParamMinMaxRange.java and listeners*

* Add new parameter details in ParamMinMaxRange.java - Name, Legend, Color, MaxValue
* Capture parsed parameter in the corresponding listener
* Add new parameter to DeviceParamValMap in corresponding listener to show its value on live grid

*2) DrawTabbedCenter.java*

* Add new parameter to ptMntrTableRowHeader
* Add new parameter to respective tempArr in setLegendsArray() method
* Use appropriate scaling method (setScaleForAnesParameters() / setScaleForPtMntrParameters()) to set scale for new parameter
* Add condition for new parameter in saveDeviceData() and fillDeviceGridCell() method

*3) Config file*

* Add new parameter under CSV\_Param\_Index section in config file and define getter for the same in SystemConfiguration.java

*4) MainController.java*

* Call appropriate scaling method to set scale for new parameter

drawTabbedCenter.setScaleForAnesParameters(ParamMinMaxRange.NAME\_OF\_PARAM, 0, ParamMinMaxRange.PARAM\_MAX\_VALUE);

or use setScaleForPtMntrParameters()

*5) HistoricalDataController.java*

* Use appropriate method (plotStaticPtMntrGraph / plotStaticAnesGraph) to add plotting logic for new parameter for vitals chart
* Add condition specific to new parameter in plotPoint() method
* Add condition for new parameter in fetchAnesDBdata/fetchPtMntrDBdata as per requirement

*6) GridHistoricalViewController.java*

* Add new parameter to ptMntrTableRowHeader
* Add conditions for new parameter in fillGrid and fillDeviceGridCell methods

1. Create a textfield in respective fxml (PatientMonitor/ AnesthesiaMashine) and set values in the corresponding controller