**DCM User Guide and Documentation**

**Assignment 1**

# Group 10 SFWRENG 3K04 Instructor: Thomas Chiang

# Members: Yash Bhatia (400362372) Connor Usaty (400409624) Daniel Lupas (400396848) Gurleen Kaur Rahi (400377038) Varunan Varathan (400375287)

Table of Contents

**No table of contents entries found.**

# Overview

This document presents a comprehensive overview of the design and implementation of Device Controller Monitor, requirements of which are presented in Assignment 1 of the SFWRENG 3K04 Pacemaker project, and serves as a user guide for anyone authorized to use the Device Controller Monitor.

The Device Controller-Monitor (DCM) is a frontend Graphical User Interface that is primarily designed to interface with the Pacemaker also designed in this Project. Presently, the DCM allows for the following:

1. Signing up and storing information of up to 10 users on a local database
2. Displaying the information as programmable Pacemaker parameters as per Boston Scientific’s pacemaker documentation.
3. Allowing the modification of certain Parameters based on four Operating modes- AOO, VOO, AAI, and VVI.

**For Testers:** A default user with default credentials has been set up for testing purposes. For those seeking to interact with the DCM GUI, please employ the following credentials on the Log In Page:

|  |  |
| --- | --- |
| **Username** | **Password** |
| admin | admin |

To run the program, store all the provided files in a common directory and run the provided main.py file

# Technical Documentation

**Program Structure:**

The DCM is programmed in Python 3.9. *(*[*https://www.python.org/downloads/*](https://www.python.org/downloads/)*)*It has been tested to work on the latest version of Python (3.12.0).

The DCM uses the Qt Window Manager ([*https://www.qt.io/product/ui-design-tools/*](https://www.qt.io/product/ui-design-tools/)) to render the GUI. The included Qt Designer program was used to create the basic elements and windows.

The translation from Qt to Python was handled by the PyQt5 library.

All user data is stored locally on an SQLITE database. (https://www.sqlite.org/)

**Dependencies:**

* PyQt5 5.15.9 ([*https://www.riverbankcomputing.com/software/pyqt/*](https://www.riverbankcomputing.com/software/pyqt/)*)*

**Files:**

|  |  |  |
| --- | --- | --- |
| **File Name** | **File Extension** | **File Purpose** |
| main | .py | Top-level program file |
| welcome | .ui | Stores XML render of the Welcome page generated by Qt Designer |
| login | .ui | Stores XML render of the login page generated by Qt Designer |
| signup | .ui | Stores XML render of the signup page generated by Qt Designer |
| landingpage | .ui | Stores XML render of the landing page generated by Qt Designer |
| [\*] | .png | Assets for rendering icons on the UI |

**High-Level Program Description:***(this section is intended for those who wish to contribute to the development of the Device Controller-Monitor)*

The program employs the use of QMainWindow, a standard object in Qt to render a window. To display different screens, the windows are stacked using a QStackedWidget upon which new windows are added. When transitioning back through the stack windows are deleted accordingly.

Each window is represented by its own Class, inheriting from QMainWindow, and employs QLabels styled with CSS to display text or display pictures through Pixmap. To enable events, the QPushButton is used, which creates a button that can have properties connected to its activation. In our case, we call custom functions that are defined within the class and actuated through ‘self’.

The database contains the ‘all\_users’ table, which contains a primary key and a username and hashed password. Other tables tie the primary keys in the all\_users database via relational mapping to programmable pacemaker parameters.

Database queries are done via a connection through the sqlite3 package. Upon login attempt, the database is queried with a hashed password through hashlib package. If the query passes, we go to the landing page. The signup page will add entries to the database in any condition, unless the user already exists or the limit of users is reached.

For now, the implementation of connection to the pacemaker is pending. The displayed state on the top right of the landing page is controlled by the pConnect Boolean variable, which can be hardcoded in the driver code.

All windows are standardized to be 1200x800 pixels.

**Variables and Constants**

**Constants:**

|  |  |  |
| --- | --- | --- |
| **Constant** | **Value** | **Notes** |
| QMainWindow.geometry | 1200x800 | This value was chosen as it presents a large canvas making the UI accessible to those who may find it hard to see small text, and also has a 3:2 aspect ratio friendly to old monitors which may be used in hospitals |
| Error colour | RED | The RGB value {255:0:0} is standardized as the colour for warnings or error messages as per OSHA and ANSI standards |
| Success colour | GREEN/BLUE |  |
| Wait times | 1 second |  |
| Default Popup Response | No | Ensures that in case of erroneous inputs, the user does not get logged out. |
| pConnect | True | Set to True for testing purposes- this will be a Variable in later implementations |
| Max Users | 10 |  |

**High-Level Program Flowchart**

**A diagram of a program

Description automatically generated**

**Requirements**

There were 6 major requirements for this assignment. The first being to develop an interface that includes a welcome screen, including the ability to register a new user (name and password), and to login as an existing user. A maximum of 10 users should be allowed to be stored locally. There are three parts to this requirement, a welcome screen, the registration and login of users, and a way to store this data. The welcome screen was designed in pyqt5 designer, by including the proper buttons for functionality and styled with simple CSS. Then the database was implemented using SQLlite. This allows new users to register, storing and encrypting their data. As well as making log ins possible by checking the users input against the database of registrations.

The second requirement was to develop essential aspects of the user interface – with respect to 3.2.2 in PACEMAKER. The first part of this requirement was properly displaying all programmable parameters and making them modifiable. This was done by developing a landing page that displays all possible programmable parameters and then having buttons beside them where you can modify the data, which will again be stored in the database. The second part of this requirement was to show when the device and pacemaker are communicating. This was completed by simply displaying an appropriate picture with the caption in the top right of the landing page based upon the pConnect variable, which is set to True for now as communication has not been added.

The third requirement