Assignment 3
Part 2

Simulinkers
Group 8
November 27, 2017
SFWR ENG 3K04

Table of Contents

1: Requirements Likely to Change

This version of the DCM has complete functionality and all the required features. However, all designs have room for improvement. Here are some of the requirement changes that are likely to occur:

- Store more than 10 patients in the database.
- Error message signifying connection has been lost during transmission of data.
- Display previous EGRAM data

2: Design Decisions Likely to Change

- Improve the plot of the EGRAM data by implementing an external graphing library and running a better graphing script.
- A confirm prompt before reprogramming the pacemaker parameters.
- Safety check to ensure pacemaker parameters that have been inputted are of reasonable value.
- Instead of a text file, use SQL as a safer system to store patient information and login details.
- Display patient's name on top of the pacemaker parameter page for doctor's reference.

3: Module Descriptions

Module: Login Page

a) Reference login details with a database to ensure correct login credentials. If the correct login credentials are entered, the pacemaker parameters page is called, otherwise a message box notification is displayed.

b) The user data is stored in a hash table which is saved within a text file. The program uses the input username as a hash "key", and retrieves the saved password of the registered patient.

c) Public Function: loadHash()

d) This function takes no parameters and loads the hash table from a known text file.

e) Global Variables:

1. User: This is a class which contains patient details such as name, username, and password.
2. Database: This is a hash table which stores patient's information "keyed" to a username.

f) Global Variables:

1. loginfetch(sender As Object, e As EventArgs) Handles MyBase,Load
2. newUser(sender As Object, e As LinkLabelLinkClickedEventArgs)
3. loginUser(sender As Object, e As EventArgs) Handles loginButton.Click

g) Public Functions:

1. This function loads the hash table from the text file stored in a known directory. The function then de-serializes it, returning an object instance of a hash table.

Private Functions:

1. This function loads the hash table containg the user data of all registered users from a text file using the public function loadHash() as described above. Iti s an event handler.
2. This function checks the database for the number of patients registered. If there are more than 10 patients registered, a message box appears. Otherwise, the New Patient module is opened.
3. This function references the hash table from the database to determine whether the correct login credentials have been entered. If the username is not a key within the hash table, then the program displays a message box notification. If the username is a key within the hash table, then the program compares the value of the key with the entered password to check for a match. If there is a match then the program navigates to the parameters page, otherwise it displays a message box notification.

Module: New Patient

a) This module registered a new patient into the database. The database is stored as a hash table which contains the username as the “key”, and a class User as the “value. The class User will contain the following details: name[string], username[string], password[string], paceParameters[hash table].

b) There are no secrets to this module.

c) Public Function: createHash()

d) This function takes no parameters and creates a hash table and stores it within a text file.

e) The global variables in this function are strings containing: name, username, and password. There is also a hash table containing the pacemakers parameters.

f) Private Functions:

1. NewUserSignUp(sender As Object, e As EventArgs) Handles ConfirmNewUser.Click

g) Internal Behaviour:

Public Functions:

1. This function creates a directory called “K04” and saves within it a text file named “database.txt”. If the text file already exists, the function deletes the text file and rewrites it. The text file stores the “Database” hash table.

Private Functions:

1. If any of the textboxes on the registration page are left blank then a message box notification is prompted. Otherwise, the function then creates a new instance of the class User, and stores the textbox information from the Registration Page within the class. This class is then stored within the database hash table and “keyed” to the username. This database is then updated and saved.

Module: Pacemaker Parameters

a) The purpose of this module is to view, save, and alter pacemaker parameters.

b) There are no secrets to this module.

c) This module uses only private functions.

d) Not applicable.

e) This module does not use any global variables.

f) Private Functions:

1. saveParameters()
2. load_textbox_vals(user As User)
3. initializeVals(user As User)
4. DefaultButton_click(sender As Object, e As EventArgs) Handles DefaultButton.Click
5. setValue_click(sender As Object, e As EventArgs) Handles setValue.Click
6. Display_Click(sender As Object, e As EventArgs) Handles Display.Click
7. ListBox2_SelectedIndexChanged(sender As Object, e As EventArgs) Handles ListBox2.SelectedIndexChanged
8. ListBox1_SelectedIndexChanged(sender As Object, e As EventArgs) Handles ListBox1.SelectedIndexChanged

g) Internal Behaviour:

1. This function iterates through each textbox parameters and updates the database value for the parameters. The update occurs if the value in the textbox is different than that stored in the database.
2. This function iterates through each textbook and sets its value to the saved value in the database. The value in the database can easily be accessed by using the parameter "key" and the hash table.

3. This function sets the pacemaker parameters to their default values and calls initializeVals(user As User) to update the values.
4. If the "default" button is clicked, the function initialzieVals(user As User) is called.
5. If the "set value" button is clicked, the function saveParameters() is called.
6. If the "display" button is clicked, the function load_textbox_vals(user As User) is called. The textboxes in the DCM will display the saved value of each parameter.
7. This function will save the "state value within the database. The selected item in the list box will be converted to a string and saved within the parameter's hash table.

Module: Serial

a) The purpose of this module is to perform serial communication with the pacemaker and properly store and transmit the data.

b) The secret of this module is that the data received over serial communication is an ASCII code and must be converted to be properly interpreted.

c) serial()

d)
This function is a default constructor and constructs the serial form and displays it to the screen.

e) Global Variables:

array: This module uses the global variable array which stores the user parameters, which will be sent over serial communication to the pacemaker.

f)

1. connect2pacemaker_Click(sender As Object, e As EventArgs) Handles connect2pacemaker.Click
2. send2pacemaker_Click(sender As Object, e As EventArgs) Handles send2pacemaker.Click

3. SerialPort1_DataReceived(ByVal sender As Object, ByVal e As System.IO.Ports.SerialDataReceivedEventArgs) Handles SerialPort1.DataReceived

4. ReceivedText(ByVal [text] As String)

g)

Public Functions:

1. This function is a default constructor and initializes the form and displays an instance of it on to the screen.

Private Functions:

1. This function initializes the serial port and specifies the COM port as well as the baud rate, and finally opens it. It then displays a green indication on the screen signifying successful connection.

2. Input parameters are byte packed with respect to their associated size and stored within an array. 'If statements' determine the index by which each parameter is stored within the array. This function loads the array, and writes it to the serial port using the command, SerialPort1.Write().

3. This function calls an instance of the function 'ReceivedText' and passes the serial data as an input parameter.

4. This function typecasts the incoming serial data from 'ASCII' to the type 'String' which can then be appended to a RichTextBox and displayed to the user. The same data will later be used to plot EGRAM data.

Module: EGRAM Plot

a) This module uses the data received from the serial port and plots it to a real time line graph which is constantly updated.

b) This module must use the global variable ‘tempey’ to store the last value transmitted by the serial port. This greatly simplifies the implementation of the EGRAM plot.

c) 1. Public Function form1()

d) 1. This function takes no input parameters and initializes a window form containing a chart and displays it to the screen.

e)
The global variables in this function is a temporary string called ‘tempey’. This variable is stored in the public module ‘temp’, which allows this variable to be accessed by the other modules which require it. This string holds the last value read by the serial port and allows it to be plotted.

f) Private Functions:

1. Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click

2. Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load

3. Timer1_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

g) Internal Behaviour:

1. When ‘Button1’ is clicked, this function first converts the string value held in the global variable ‘tempey’ to an unsigned 8 bit integer. Then,the point is plotted on to the chart contained in form1 using the DataVisualization library.

2. This function initializes the chart on form1 and sets the timer interval to 100 ms. This value will determine the interval by which the EGRAM will update. Finally, it starts the timer.

3. This function handles the event that will occur at each time interval. This function simulate a click of Button1 at intervals of 100 ms as determined before.

DCM Test Cases

Module: Login Page

Test Case: 1
Input: unregistered username
Expected Output: “Invalid Username”
Actual Output: “Invalid Username”
Pass/Fail: Pass

Test Case: 2
Input: Registered username and password
Expected Output: Parameters Page
Actual Output: Parameters Page
Pass/Fail: Pass

Test Case: 3
Input: Registered username and invalid password
Expected Output: Invalid Password
Actual Output: Invalid Password
Pass/Fail: Pass

Module: New Patient

Test Case: 1
Input: Username: ##$!# , Password: ##@@
Expected Output: “New Patient Saved in Database”
Actual Output: New Patient Saved in Database”
Pass/Fail: Pass

Test Case: 2
Input: Blank textboxes
Expected Output: “Please input the required information”
Actual Output: “Please input the required information”
Pass/Fail: Pass

Test Case: 3
Input: Existing username in to the database
Expected Output: “New Patient Saved in Database”
Actual Output: Error: System.ArgumentException
Pass/Fail: Fail, Comment: Implement exception to handle this error in the next version of DCM

Module: Pacemaker Parameters

Test Case: 1
Input: String input for integer value.
Expected Output: “Invalid Entry”
Actual Output: No error or Prompt
Pass/Fail: Fail
Note: DCM should check for proper input type.

Test Case: 2
Input: Blank textboxes
Expected Output: No error
Actual Output: No error
Pass/Fail: Pass

Test Case: 3
Input: Integer values
Expected Output: No error, values saved and displayed upon login.
Actual Output: No error, values saved and displayed upon login.
Pass/Fail: Pass

Module: Serial

Test Case: 1
Input: Connect and send data from pacemaker
Expected Output: Display received live data in RichTextBox
Actual Output: Display received live data in RichTextBox
Pass/Fail: Pass

Test Case: 2
Input: Integer parameters
Expected Output: Successful Communication
Actual Output: Successful Communication
Pass/Fail: Pass

Test Case: 3
Input: Click ‘disconnect’ before establishing serial communication
Expected Output: Disconnect
Actual Output: Program Crashes
Pass/Fail: Fail
Note: DCM should have an exception case where a prompt notifies the user that the pacemaker must first be connected before disconnecting. This will be implemented in the next version of the DCM.

Module: EGRAM Plot [form1]

Test Case: 1
Input: Attempt to Plot EGRAM before connecting to serial communication
Expected Output: Plot displayed, no change
Actual Output: Plot displayed, no change
Pass/Fail: Pass

Test Case: 2
Input: Attempt to Plot strings instead of integers.
Expected Output: Plot displayed, no change
Actual Output: Plot displayed, no change
Pass/Fail: Pass

Test Case: 3
Input: Integer Values
Expected Output: Updating graph, in intervals of 100 ms
Actual Output: Updating graph, in intervals of 100 ms
Pass/Fail: Pass

Rate Adaptive

Parameter page test case:
Input: rate adaptive parameters
Expected output: successful transmission and storage of variables
Actual output: ^^same