SE3K04 – GROUP 29

DCM Documentation- Assignment 1

OCTOBER 31, 2021

Carter Moore

Cassidy Smith

Charlotte Neumann

Julia Lisboa

Rebecca Byers

Sharon Cai

Contents

1– Current Requirements

The current DCM software must allow for operation of a simple GUI that allows users to input and store patient's pacemaker parameters. This will connect with hardware in the future.

1.1 – Welcome Screen

The DCM software must include a welcome screen that allows users to create an account (new name and password) or login as an existing user with a username and password. A maximum of 10 users should be stored.

1.2 – Essential Aspects of the User Interface

The DCM software must be able to operate the following functions:

- Utilize and manage windows for display of text and graphics
- Process user position and input buttons
- Display programmable parameters and allow them to be reviewed and modified
- Indicate when the DCM and the device are communicating. *
- Indicate when a different pacemaker device is approached than previously. *

*Since at this stage of development there is no communication between DCM and hardware, the user interfaces will be created, but not connected to the physical pacemaker yet.

1.3 – Pacing Mode

The DCM software must be able to present all pacing modes. Currently, the modes that must be users can input and display are AOO, VOO, AAI and VVI.

1.4 – Parameter Storage

The DCM must take input for programmable parameters and allow them to be stored and modified. Currently, the parameters and their programmable values that must be stored are as follows:

Parameter	**Units**	**Range**	**Increment**
Lower Rate Limit	ppm	30-50 50-90 90-175	5 1 5
Upper Rate Limit	ppm	50-175	5
Atrial Amplitude*	V	0.5-5	-
Ventricular Amplitude*	V	0.5-5	-
Atrial Pulse Width**	ms	0.05, 0.1-1.9	0.1
Ventricular Pulse Width**	ms	0.05, 0.1-1.9	0.1

Atrial Refractory Period	ms	150-500	10
Ventricular Refractory Period	ms	150-500	10
PVARP***	ms	150-500	10
Hysteresis Rate Limit***	ppm	30-50 50-90 90-175	5 1 5
Rate Smoothing****	-	-	-

*Although PACEMAKER documentation provides different programmable values, for the hardware program to run in the future, the amplitude must be in the stated limit.
**Although ranges were listed in PACEMAKER documentation, they were not implemented as per TA request.
***Although these parameters are not required in this deliverable, they were stated as possible required parameters and will be stored.
****No range error catching to be implemented since this parameter is not required for this deliverable.

2 – Requirements Likely to Change

Since the current DCM does not interact with the pacemaker hardware, there are many aspects that must change to allow for interactions with the hardware.

2.1 – Welcome Screen

The welcome screen and user login are unlikely to change.

2.2 – Essential Aspects of the User Interface

The DCM software will remain consistent; however, some features may change and be added. The program will change to add the following functions:

- Indicate when the DCM and the device are communicating. *
- Indicate when telemetry is lost due to device being out of range.
- Indicate when telemetry is lost due to noise.
- Indicate when a different pacemaker device is approached than previously. *

*These functions currently exist but it will change to communicate with the hardware. Arbitrary constants will be passed through until hardware connection is established.

2.3 – Pacing Mode

The DCM software must be able to present all current pacing modes as well as change to implement DOO, AOOR, VOOR, AAIR, VVIR and DOOR modes.

2.4 – Parameter Storage

The DCM must take input for programmable parameters and allow them to be stored and modified. Several parameter inputs with allowable ranges will have to be added for the new modes. They can be found in the document PACEMAKER. These parameters will have to be passed to the hardware to simulate pacemaker activity.

3 – General Design Decisions

The DCM was programmed in Python 3. The decision to use Python 3 was based on the programmer’s familiarity with the language. It has simplified syntax, clear error catching and emphasizes natural language. Additionally, it is a high-level language that has many options for implementing GUIs and storing data. The language used is unlikely to change as it has the potential to be further developed to meet all necessary future requirements.

3.1 – GUI

To implement the DCM and provide graphics for the GUI, the package ‘PySimpleGui’ was used. The module works by transforming the tkinter, Qt, WxPython, and Remi (browser-based) GUI frameworks into a simpler interface [(PySimpleGUI)]. was installed by the package manager ‘pip’. The decision to use this package was based on the programmer’s familiarity and experience. Additionally, the package offers several easy-to-implement features (such as customizable graphics, buttons, text boxes & drop-down menus) that are useful for a DCM. The GUI package used is unlikely to change as it has the capabilities to be edited to meet all necessary future requirements.

3.2 – Parameter Storage and Manipulation

To store the username, password, mode, parameters and pacemaker ID, a Comma Separated Values (.csv) file was used. This file type was chosen as it is widely used, human readable, and easy to read and edit using Python. The following headers were used to store user inputted data:

Name	Password	Mode	LRL	URL	AA	APW	AS	ARP	PVARP	VA	VPW	VS	VVP	Hyst	RS	ID

At this point in the design process, these are all the necessary values that must be stored and checked. Every field (except for ID) is inputted by the user, and it is likely that the parameters will be passed to the pacemaker hardware upon connection. Pacemaker ID (ID) will be read from the pacemaker when hardware is implemented, but for now, an arbitrary integer is assigned to it to test if the pacemaker has been encountered before. The storage file type used is unlikely to change, however, headers with the necessary new fields will likely be added to meet new mode parameter requirements.

For parameter reading and appending, the ‘csv’ public module was imported and for editing, the ‘pandas’ public module was used. The ‘csv’ module allowed fields to be read as indexes on a list or as a dictionary (using DictReader functions), both of which were became useful for programming. For file reading, the syntax using the ‘csv’ module used was easier than the ‘pandas’ equivalent. The ability to append a new line seemed the simplest using this program, thus this module was used. A limitation of the ‘csv’ module was parameter editing, therefore, ‘pandas’ was used instead. With ‘csv’, the process to edit a specific field was too complex and ‘pandas’ offered an easier way for users to make modifications, so it was primarily used for when users needed to make mode or parameter edits. The use of both these packages is unlikely to change as they allow for a wide range of functionality that can meet future necessary requirements.

4 – Module Specification

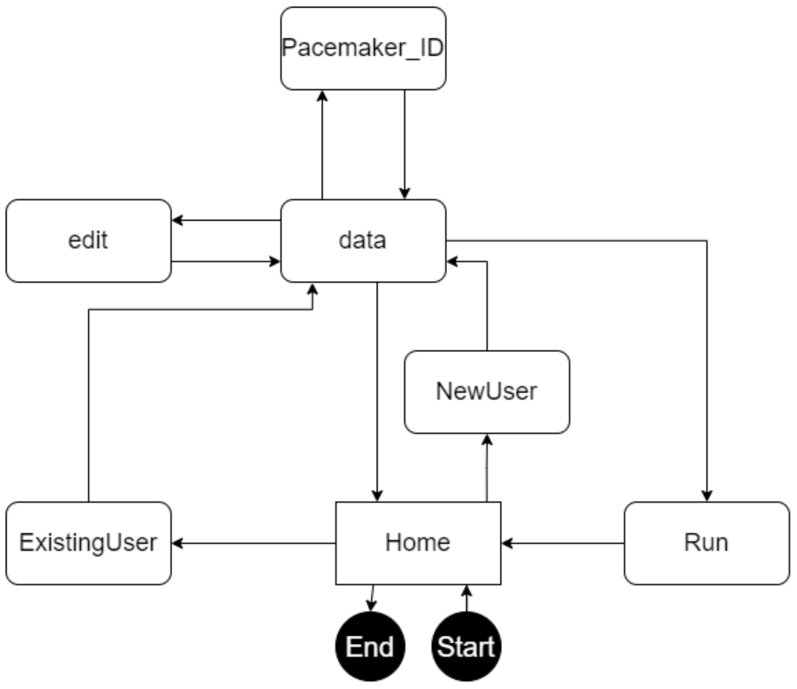


Figure 1: Diagram displaying program state flow and the behaviour of each module by calling other modules. (Note: the 'Error' module is not included as most modules call it at some point for error catching.)

Apart from the preinstalled public libraries mentioned in Design Decisions, there are 8 custom modules that were created for the DCM. Figure 1 shows the accessibility of the modules from each other and their behaviour.

4.1 – Home.py

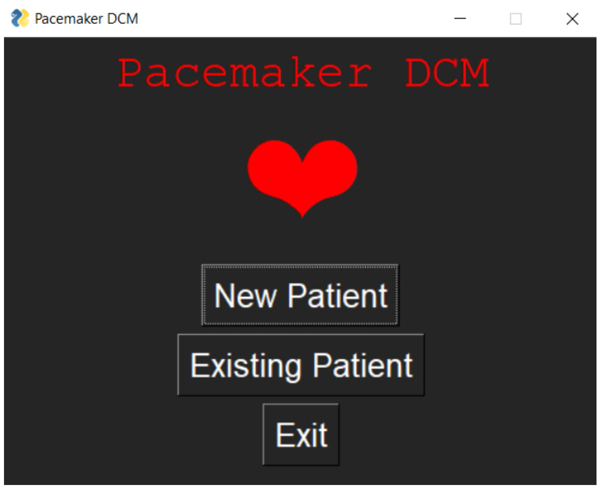


Figure 2: Welcome Screen of the pacemaker DCM.

Home.py is a module in which its main purpose is to welcome the user and redirect them to login a new user, existing patient or exit the program. Most of the other modules relay back to the home screen as it allows users to navigate across the program. It imports PySimpleGui, NewUser, ExistingUser and emojize from emoji. It shows 3 buttons that redirect the user to the next screen:

Buttons:

sg.Button("New Patient")	NewUser.py -> new_patient()
sg.Button("Existing Patient")	ExistingUser.py -> login()
sg.Button("Exit")	exit()

The only function within this module is the main function. The theme for the entire DCM is set to the theme package within PySimpleGui, 'DarkBrown4'. The layout is set up by adding the buttons listed above. The emojize function within the emoji library is called to display a red heart to add a graphic to the window.

A loop is used to detect an event in user button input. Depending on the input, another function module is called according to Figure 3. When a user clicks a button, 'Home' window closes, the loop breaks and the window of the selected module opens.

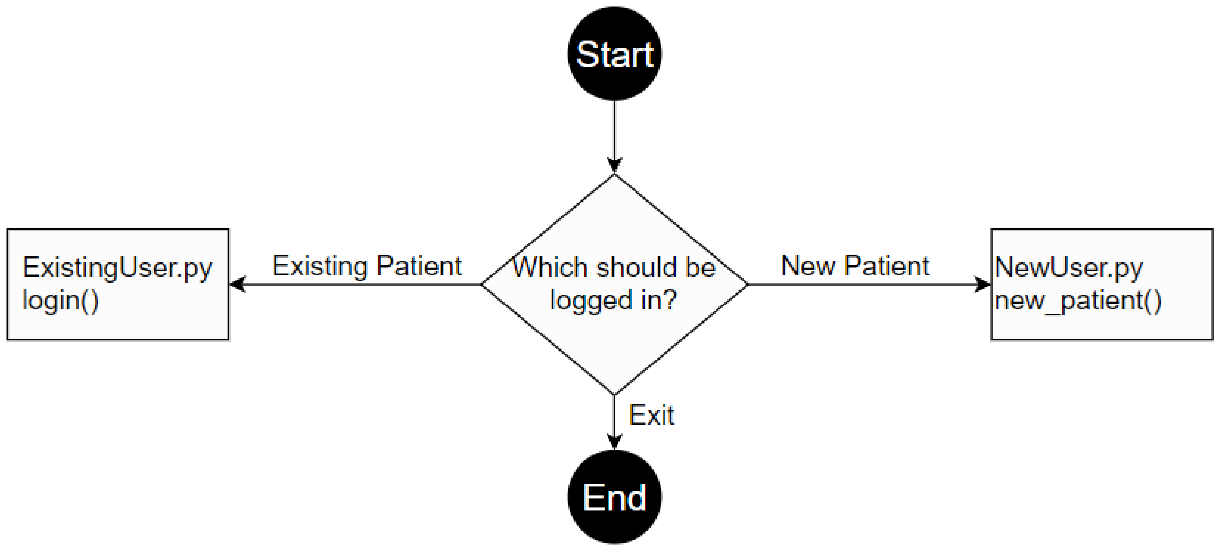


Figure 3: Flowchart describing the behaviour of the main function within Home.py. It redirects user to the ExistingUser or NewUser modules or terminates the program.

4.2 – NewUser.py

This module's purpose is to allow a new patient to be added to the data base. It uses PySimpleGui, csv, and os as well as custom modules Error, Run, data, and existing user.

The functions in this module include:

Function	**Parameter**	**Purpose**
new_patient()	none	Serve as the main function in the module, will display the initial window to allow information to be passed the program and call the next necessary functions.
mode_input(name, pw, ID)	Patient name, password, and pacemaker ID	Acts as a *main* function to call mode_change, AOO_change, etc where appropriate.
AOO_input(name, pw, ID)	Patient name, password, and pacemaker ID	Gets input from the user for each parameter required for mode AOO. Call the functions to check that parameters are in the allowable range and writing to the csv file.
VOO_input(name, pw, ID)	Patient name, password, and pacemaker ID	Gets input from the user for each parameter required for mode VOO. Call the functions to check that parameters are in the allowable range and writing to the csv file
AAI_input(name, pw, ID)	Patient name, password, and pacemaker ID	Gets input from the user for each parameter required for mode AAI. Call the functions to check that parameters are in the allowable range and writing to the csv file

VVI_input(name, pw, ID)	Patient name, password, and pacemaker ID	Gets input from the user for each parameter required for mode VVI. Call the functions to check that parameters are in the allowable range and writing to the csv file
append_to_file(info)	List of values to add to the csv file.	Write given values to the csv file.
check_length()	none	Check that the number of patients on file is less than or equal to 10. Display an error message if the file has hit max number of patients. Return the current number of patients.

The new_patient() function does not take any inputs and initially calls the check_length() function to ensure that there are less than 10 users currently in the database. If there is room to add another patient, the program displays this window to the user:

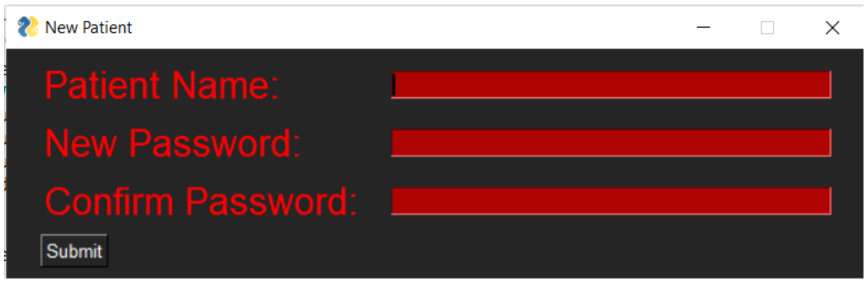
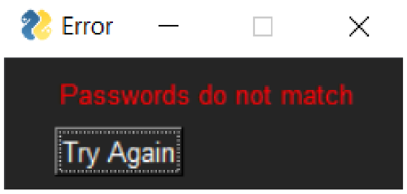


Figure 4: Window prompting user to enter new user information.

PySimpleGui stores the user's inputs in a list and then it is checked that the two passwords the user entered match. If they do not match the pw_error() function from the Error module is called with the parameter 'mismatch'. That displays this window to the user:


Error

Passwords do not match

Figure 5: Error window indicating passwords do not match.

The user can then try again to enter passwords that match.

If the passwords match, the window closes and the mode_input() function is called, passing the patient name and password as well as the length of the file to be used as the ID currently. A different number could be passed as the pacemaker ID in the future, but for now they are just numbered arbitrarily from 1 to 10.

The mode_input() function takes the patient name, password, and pacemaker ID as parameters and serves to get the desired mode from the user. It displays this window:

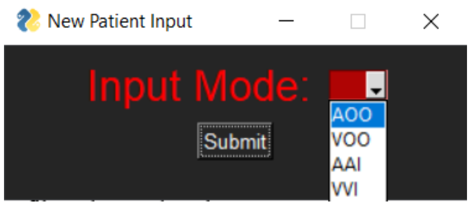


Figure 6: Drop-down menu prompting the user to enter one of 4 possible modes.

using a drop down menu from PySimpleGui. The user's selection is tested to ensure that they selected a mode and did not leave it blank. If it was left blank, using the pw_error() function from the Error module with the parameter 'mode' the following window is displayed:

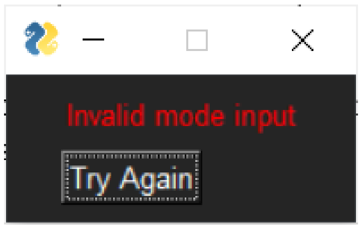
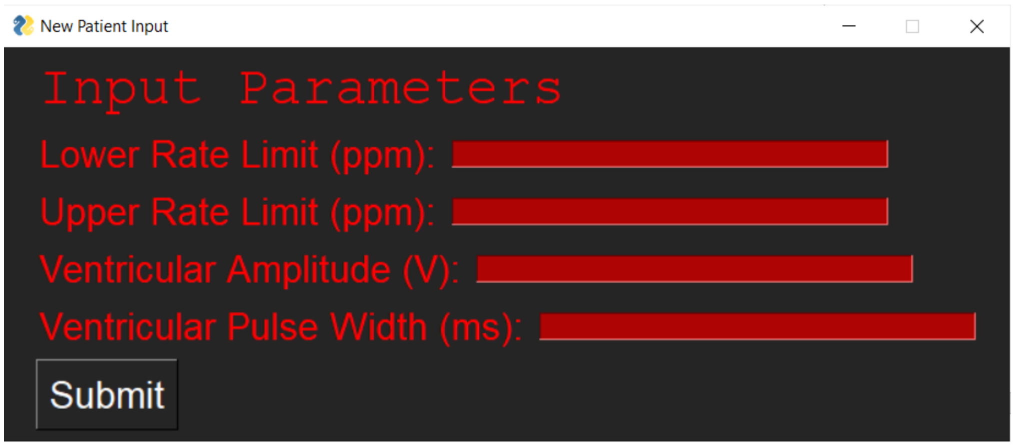

Invalid mode input

Figure 7: Error message if user fails to insert a mode input.

The user can then try again and select a valid input. Once there is a valid input, the input is tested to determine which of the four allowable modes was selected. Based on that, the next function is called according to the following:

User Selection	**Function Call**
AOO	AOO_input()
VOO	VOO_input()
AAI	AAI_input()
VVI	VVI_input()

The AOO_input(), VOO_input(), AAI_input(), and VVI_input() functions all operate similarly. They take the patient name, password, and pacemaker ID as inputs and serve to output a list that represents the patient's row of information in the data base. Each function displays their own window that asks for input for their necessary parameters:



Input Parameters

Lower Rate Limit (ppm):

Upper Rate Limit (ppm):

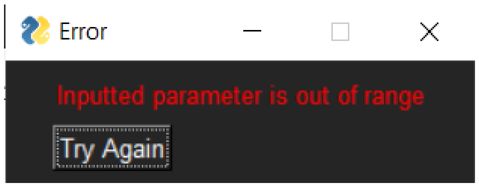
Ventricular Amplitude (V):

Ventricular Pulse Width (ms):

Figure 8: Window showing all possible parameters that must be inputted. This changes depending on the mode the user inputted previously.

The above window is an example for VOO inputs. The necessary parameters for each mode are outlined in the PACEMAKER document.

Once the user presses ‘Submit’, the values entered are stored in a values list by PySimpleGui. A list containing the corresponding parameters is also created, and then a loop is used to check that each parameter’s value is in the allowable range using the range_error() function from the Error module. Those allowable ranges are taken from the PACEMAKER document. If one of the parameters is out of the allowable range, the following window is displayed:



Inputted parameter is out of range

Figure 9: Error window displaying the inputted parameter is out of the range listed in requirements.

Once the user presses ‘Try Again’, they will be redirected to the same window as previously. When the inputted parameters are within their allowable ranges, the input functions make a list containing the patient name, password, mode, pacemaker ID, and the parameters just entered. It then calls the append_to_file() function with that list as the parameter.

The append_to_file() takes a list containing patient info as its parameter and opens the csv file and writes that list to a new row, effectively adding the new patient to the data base. The csv module is used to do this. The function then calls the printing() function from the data module to display to the user the patient information.

The check_length() function takes no parameters and opens the csv file in read mode to count the number of rows it currently contains. It uses this number to determine if there is room to add more patients, meaning if there are less than 10 people currently having their data stored. If there are already 10 people in the data base, it displays the following error window:

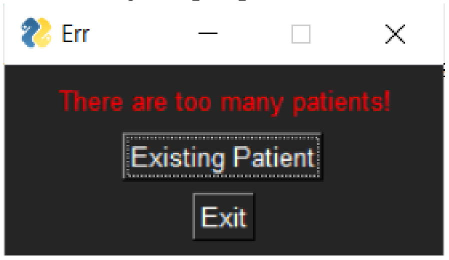


Figure 10: Error window displaying that there are more than 10 patients in the file.

The function outputs the current number of rows in the csv file for use at the beginning of the new_patient() function.

4.3 – Data.py

This module's main purpose is to display the current parameters stored in the csv file to the user. It shows the specific patient's data and allows the user to choose between running a simulation, returning to the home page, or edit the parameters that are being displayed. It uses PySimpleGui, csv, and os as well as custom modules Error, Pacemaker_ID, Run, and edit.

The module serves to display this window to the user:

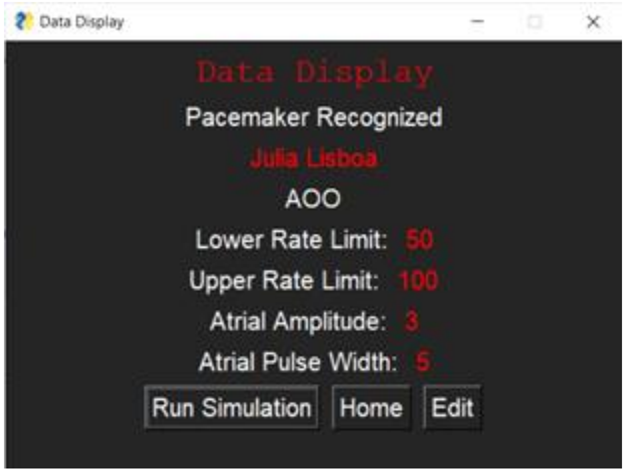


Figure 11: Window displaying that the pacemaker ID was recognized, patient name, mode and the parameters. There are 3 buttons that dictate the next window the user can go to.

There is only one function contained in the module, defined as printing(Name_given) where the function is just called with the name of the current patient. The function takes the name of the current patient and generates the window above to display the patient's information.

In the function, the csv file is opened in read mode and find the row with the appropriate patient's name.

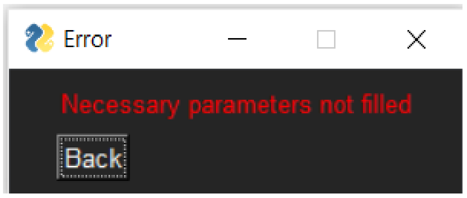
Next, using the Pacemaker_ID module, it is tested if patient's registered ID is the same ID of the pacemaker being "approached". Currently there is no communication between the pacemaker and DCM so this currently always returns True and allows the function to continue. In the future, its output may change depending on if the pacemaker has been approached.

Next, the function looks for the current mode stored for the patient. Depending on that mode, the appropriate parameters are displayed for review in the window above using PySimpleGui. The parameters correspond to the required parameters for each mode as outlined in the PACEMAKER document.

Using the PySimpleGui function to make buttons, the user is given three options.

sg.Button("Run Simulation")	Error.py ->para_filled() Run.py -> running()
sg.Button("Home")	os.system(Home.py)
sg.Button("Edit")	Edit.py->to_edit()

If the *Run Simulation* button is pressed, we first call the para_filled(Name) function from the Error module with the patient name. This checks to see if the parameters required for the current mode are filled. If that function returns False, meaning that there is a parameter that is unfilled, it displays an error window shown below and allows the user to go back to the data display window above so that they can edit the parameters before trying to run the pacemaker again.



Necessary parameters not filled

Figure 12: Error window displaying that the necessary parameters are not filled.

sg.Button("Back")	data.py->printing()

If the function para_filled(Name) returns True, meaning that all the necessary parameters are filled, we call the running function from the Run module to run the pacemaker.

If the *Home* button is pressed we use os.system function to call the home module and display the welcome screen again.

If the *Edit* button is pressed, we call the to_edit(Name) function with the patient name from the edit module.

In all the button cases, the current window is closed to allow the user to focus on the next one that is set to appear.

4.4 – Edit.py

This module's main purpose is to allow the user to edit the parameters used for each patient that we are storing in the csv file. It uses PySimpleGui, csv, and pandas as well as custom modules data and Error.

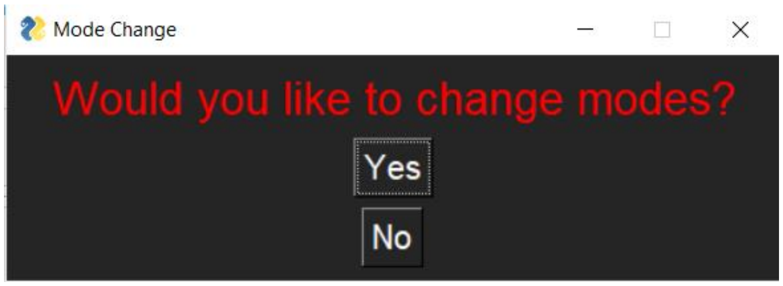
The functions in this module include:

Function	Parameter	Purpose
row_num(Name)	Patient name	Find and output the row number in the csv file where the current patient's information can be found.
to_edit(Name)	Patient name	Acts as a *main* function to call mode_change, AOO_change, etc where appropriate.
mode_change(Name)	Patient name	Allow the user to switch between modes AOO, VOO, AAI, and VVI.
AOO_change(Name)	Patient name	Allow the user to edit the parameters corresponding to mode AOO.
VOO_change(Name)	Patient name	Allow the user to edit the parameters corresponding to mode VOO.
AAI_change(Name)	Patient name	Allow the user to edit the parameters corresponding to mode AAI.
VVI_change(Name)	Patient name	Allow the user to edit the parameters corresponding to mode VVI.

The row_num(Name) function takes the current users name, opens the csv file, and finds the row containing that user's name. It counts how many rows it took to get there and returns an integer corresponding to the correct row in the file.

The to_edit(Name) function takes the current user's name and opens the csv file in read mode. It calls the mode_change(Name) function to find out if the user would like to change modes. After the mode_change function is executed, it enters a series of if statements to find which mode the patient is currently using. It does this by testing the 'Mode' column of the csv file against strings such as "AOO", "VOO", etc. In each if statement, the corresponding edit function is called for the user to edit the patient parameters. For example when the Mode == "AOO" we call AOO_change().

The mode_change(Name) function takes the current user's name and serves to display this window to the user:



Would you like to change modes?

Figure 13: Window asking user if they would like to make mode modifications.

Buttons:

sg.Button("Yes")	Using panadas edit the csv file to contain the new mode selected by the user. data.py->printing()
sg.Button("No")	window.close()

If the user selects 'Yes' the window is closed and a new one is displayed, it is shown below.

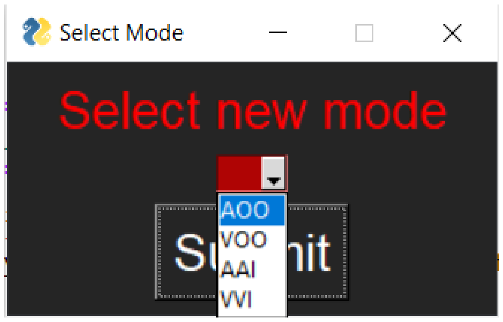


Figure 14: Window with drop-down menu of mode selection.

The user is given a drop-down menu to select their new mode from and a submit button to finish the process.

Buttons:

modes =['AOO','VOO','AAI','VVI']
sg.Combo(values = modes)
	Used to ensure that the user can only select modes that exist.

sg.Button("Submit")	window.close() use pandas to edit the csv and save the new mode in the Mode column.

When the user selects a new mode and presses submit, the current window is closed.

It is tested if the user selected a new mode and did not leave it blank. If it was left blank, the pw_error(error) function is called from the Error module with the parameter 'mode' that displays this window:

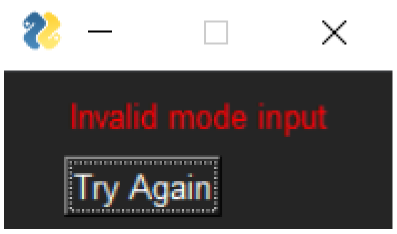


Figure 15: Same error window as Figure 7

The user can then try again and ensure they selected a real mode.

Once there is a valid mode selected, we write to the csv file using pandas.

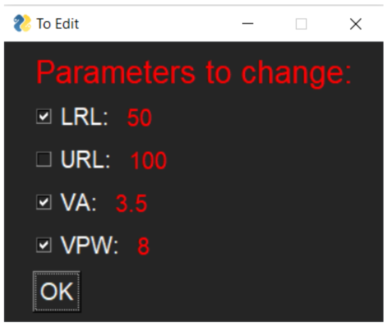
pandas is imported as pd:

```
df = pd.read_csv("Demo.csv")

df.at[row_num(Name),"Mode"] = values[0]

df.to_csv("Demo.csv", index=False)
```

the built in function .at[] in pandas takes the general form .at[row number, column] = value to assign. The row_num function is used as defined earlier to fill the first parameter. The ‘Mode’ column must be modified to the value that was just selected by the user. That value is stored in a values list by PySimpleGui so we use values[0].

If in Figure 13 (mode modification window) the user selects no, the window is closed and nothing else occurs. Since this function is called in to_edit(), that function is allowed to continue.

AOO_change(Name), VOO_change(Name), AAI_change(Name), and VVI_change(Name) all operate similarly. They take the name of the current patient and open the csv file to the appropriate row. Each function takes the appropriate parameters for their respective modes, as outlined in the PACEMAKER document, and outputs a window shown below.


To Edit

Parameters to change:

☑ LRL: 50

☐ URL: 100

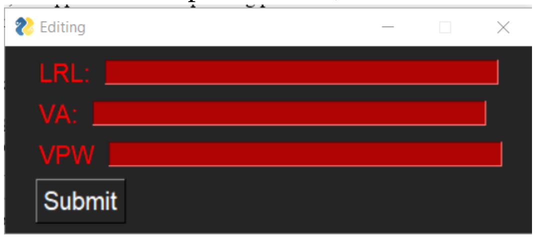
☑ VA: 3.5

☑ VPW: 8

Figure 16: Window for user to select which parameters to modify.

The current numbers for each parameter are read from the csv file and displayed beside a check box. The user can check which parameters they would like to edit and press OK. When that is done, PySimpleGui saves True or False where appropriate to a list called values. The program then goes through the list to find where there are values marked True (where the user would like to make edits.) If say values[0] == True, that corresponding parameter is appended to the list of text for our next window. That parameter is also appended to a list called *change* that is used to store the names of the columns we need to edit in the csv file.

Once the user presses OK, that window is closed a new one appears as shown below.


Editing

LRL:

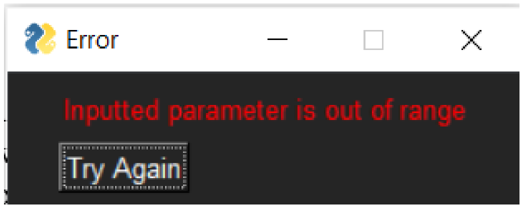
VA:

VPW

Figure 17: Modified parameter input window.

The parameters that appear here are the ones that user checked off in the previous window. The user can enter their new entries, which are stored in a new values list. The user clicks ‘Submit’ to close their window and pass their numbers into the program.

Once that process is finished, a loop is used and the range_error() function found in the Error module checks that all of the parameters entered are in the allowable range, as outlined in the PACEMAKER document. If the entered parameters are out of range, this error message will appear:



Inputted parameter is out of range

Figure 18: Same error window as Figure 9

The user can press ‘Try Again’ to go back to the previous window. Once the parameters given are within the allowable range the program continues.

Then a loop is used and the same pandas function as outlined above in the mode_change() explanation to edit the appropriate columns in the csv file. This loop is why the columns the user wanted to edit was stored, as they are passed into the pandas .at[] function along with the numbers they entered that are found in the values list.

4.5 – Run.py

This module’s main purpose is to indicate to the user when the pacemaker is running. Upon execution, it shows an animated loading bar to the patient that can be terminated using a ‘Cancel’ button. When this button is clicked, it takes the user back to the home screen. It uses public libraries, PySimpleGui, csv and os.

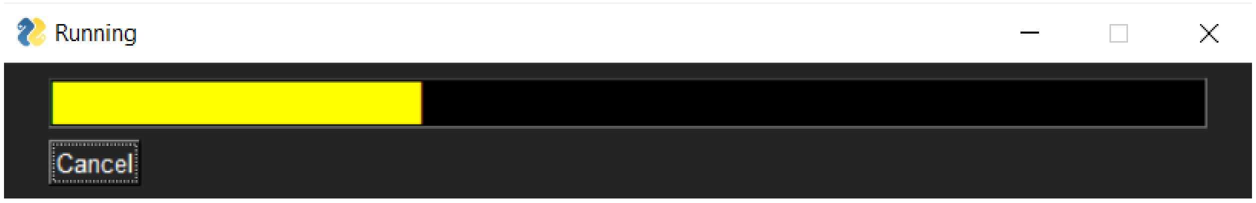


Figure 19: Animated running bar to show pacemaker progress.

There are two functions within this module that take the following inputs and do the following operations.

running()	Creates animated loading bar display graphic that user can cancel when necessary
load_parameters(Name_given)	Called when simulation must be run, loads necessary parameters, and stores them as variables.

The load_parameters() function takes the patients name as input and calls the csv module to read the file as a dictionary. It locates the row with the inputted name. Conditional variables are used to determine the patient's mode and loads certain parameters based on the requirements listed in PACEMAKER. In the future, this module will be linked to the hardware, and it will be responsible for passing parameters to the Simulink model, but for now, they are initialized and stored. After, the running() function is called.

The running() function takes no inputs and is based on a function found on Stack Overflow. It is responsible for displaying a window that shows an animated loading bar that indicates when the hardware is running. The layout including a loading bar and cancel button are set up. A loop is initialized that has a text box constantly filling to give the appearance of a loading bar until the user presses 'Cancel'. When the user presses 'Cancel', the os library is called to restart the 'Home.py' module, close the current window and send the user to the home screen. For now, the length of time the program runs is user-controlled and the parameters are initialized and stored in variables but not passed through anything. In the future, this function running will be dependent on if the Simulink model is running.

4.6 – ExistingUser.py

This module is used to allow existing users to log into the system. It uses PySimpleGui and csv as well as our own modules Error and data.

There is only one function contained in the module:

Function	**Parameter**	**Purpose**
login()	none	To display a window allowing existing patients to log into the system.

The login() function takes no parameters and displays the window below:

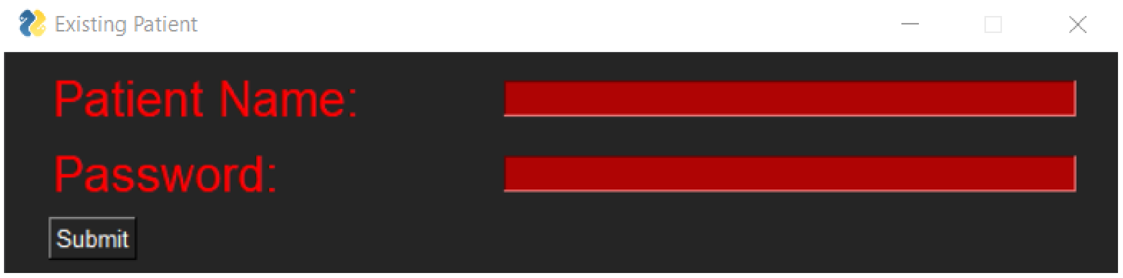


Figure 20: Window prompting user to input an existing patient name and password.

It takes the patient name and password from the user and reads the csv file. The strings entered by the user are stored in a values list by PySimpleGui. The function then loops through rows of the csv file until it finds that the name and password can be found in the same row.

If the patient name and password are found, the window is closed and the printing(name) function is called from the data module.

If the patient name and password cannot be found in the csv file, the pw_error('incorrect') function is called from the Error module.

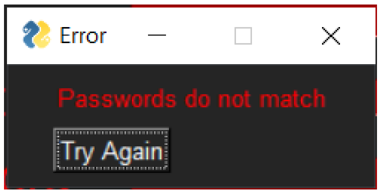
4.7 – Error.py

The purpose of this module is to catch errors in other modules of the program and alert the user when an error of invalid input is passed through the program.

Function	**Parameter**	**Purpose**
pw_error(error)	Type of error (defined and used in NewUser / ExistingUser modules. Options are mismatch, incorrect, or mode.	To output error messages corresponding to the error entered as a parameter.
range_error(user_input, para)	The value (integer/float) entered by the user and the parameter that the value is meant to fill. Ex. 300, "URL"	To ensure that the user-entered values are within the allowable range for their parameter (ranges taken from the PACEMAKER document)
para_filled(Name)	Patient name	Ensure that the parameters required for each specific mode are filled. Ex. AOO requires that the parameters URL, LRL, AA, and APW are filled. This is used before allowing simulations to run. Function returns boolean value *True* or *False*.

The pw_error(error) function takes in a string that corresponds to the type of error message that needs to be displayed. It is used in the NewUser and ExistingUser modules.

If 'mismatch' is passed into the function, the following window is displayed:

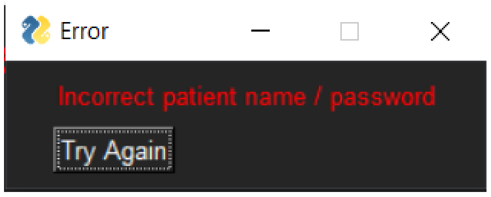


Passwords do not match

Figure 21: Error window indicating password mismatch.

This is used in the NewUser module when the user is entering a new patient's information it is required that they input the new password twice. If they do not match, the pw_error function is called.

If 'incorrect' is passed into the function, the following window is displayed:


Error

Incorrect patient name / password

Figure 22: Error window indicating incorrect password when a user tries to login.

This is used in the ExistingUser module when a patient is trying to login and their entered name and password do not match.

If ‘mode’ is passed into the function, the following window is displayed:

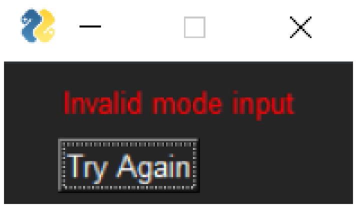
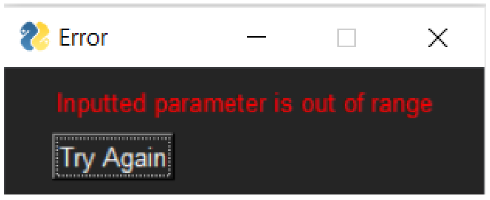

Invalid mode input

Figure 23: Error window indicating blank mode input.

This is used in the NewUser module and the edit module when a user fails to select a mode from their given drop down menu.

The range_error(user_input, para) takes a number entered by the user as user_input and the parameter that the value is meant to be used for as para. This could be 400, “URL” for example. The function has different conditions for each possible parameter where the numbers are tested to be in the acceptable range as outlined in the PACEMAKER document. The variable *Flag* is used to indicate whether there are any errors found. It is initially set to false and if any of the conditional statements testing the allowable range are True, *Flag* is set to True. At the end of the function, if *Flag* == True, the following window is displayed:


Error

Inputted parameter is out of range

Figure 24: Error window indicating parameter inputted did not meet listed requirements.

In all the above functions, the user pressing ‘*Try Again*’ after the error message appears will return them to the window previous to fix their error.

The para_filled(Name) function takes the patient name as an input and opens the csv file to the appropriate row. It then checks the mode that is currently stored for the patient and from there checks that all the necessary parameters are filled. These are found in the PACEMAKER document. The function uses a similar structure with *Flag* as the range_error() function above, but we return *Flag* at the end. There is no window created within the function. Its main purpose is to return True or False.

4.8 – Pacemaker_ID.py

This module will become more useful in the next stages of this project. Its purpose is to return true when the pacemaker ID we expect from the current patient matches that of the pacemaker being approached.

The module contains a single function:

Function	**Parameter**	**Purpose**
ID_num(ID)	ID number	To indicate if the entered ID number is the same as the one we expect (being in the csv file)

The ID_num(ID) function opens the csv file to the current patient and reads their corresponding ID number. If the ID number passed as a parameter to the function matches the one on file for the patient, the function returns True. Otherwise it returns False.

5 – Testing

5.1 – New Users

When a new user logs into the DCM they must provide their name and passwords that match to confirm their selection. They must also provide a valid mode that they would like to start in, as well as valid values for that mode.

Given this window:

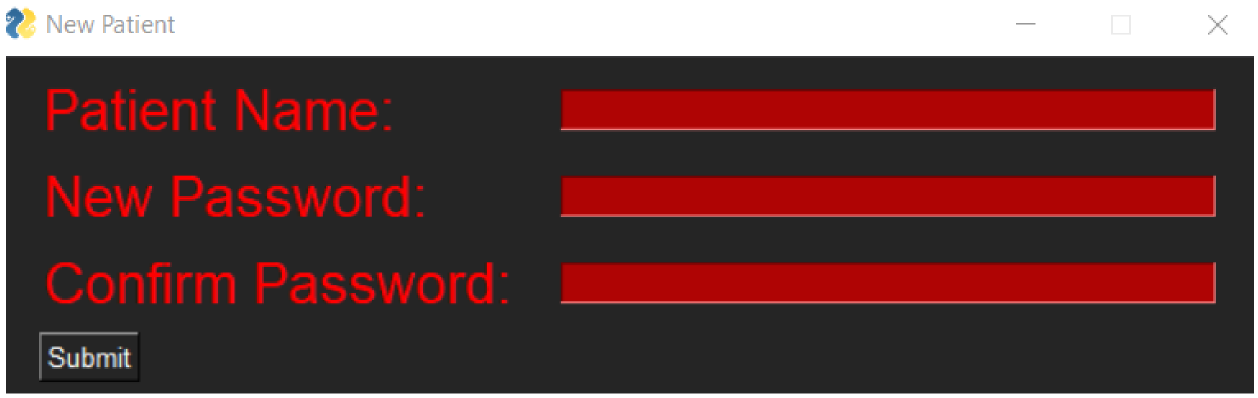
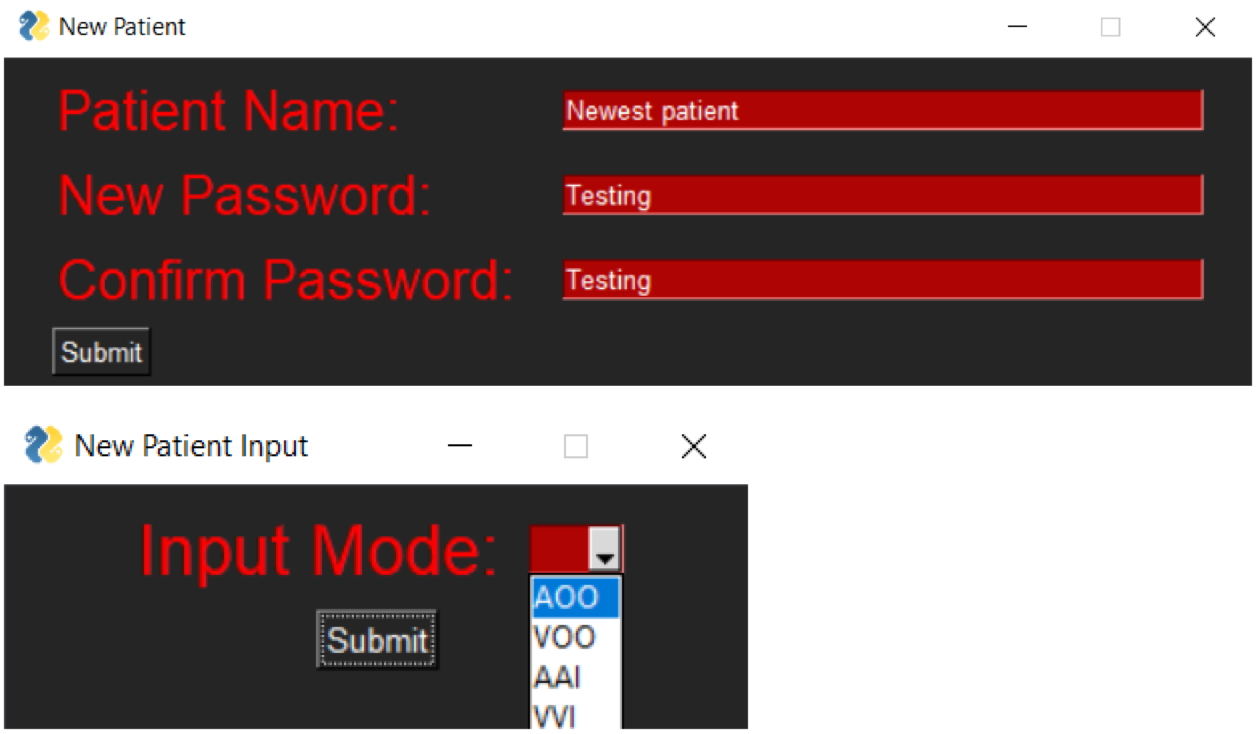


Figure 25: New user account creation window. Same as Figure 4.

If the user enters their name and passwords that match each other, it is expected for them to continue to the choose mode screen.


New Patient

Patient Name: Newest patient

New Password: Testing

Confirm Password: Testing

Submit

New Patient Input

Input Mode:

AOO

VOO

AAI

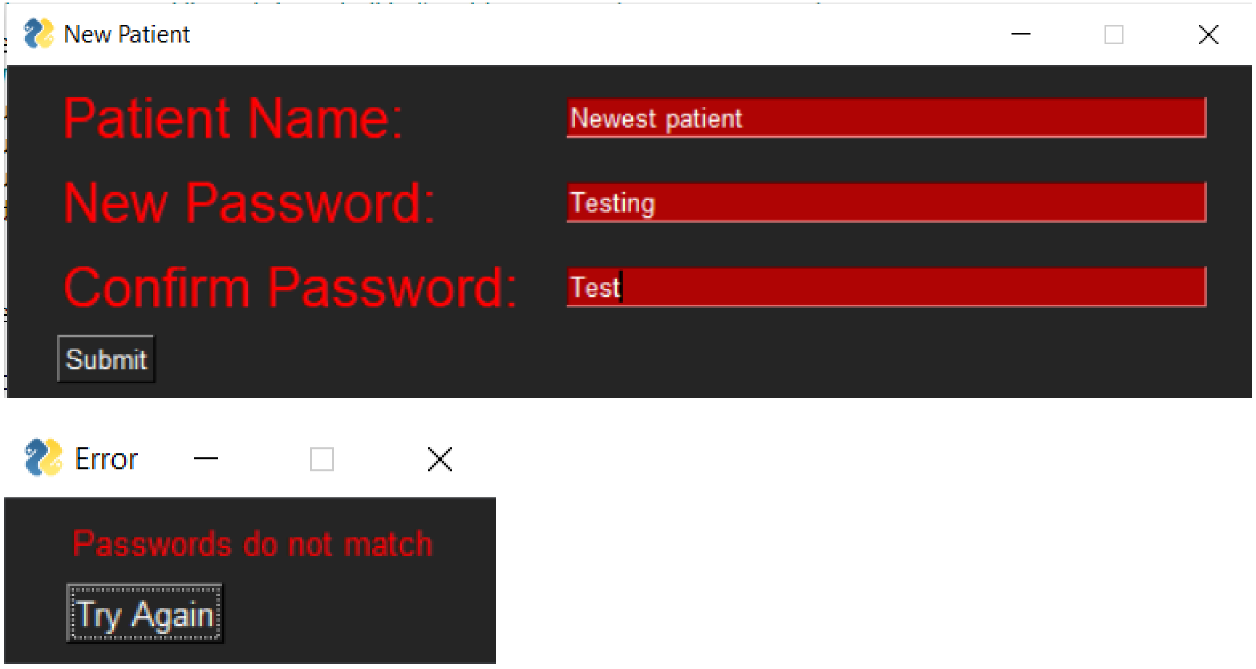
VVI

Submit

Figure 26: Creating a new patient in AOO mode.

As shown in the windows above, that functionality works as expected.

If the user is presented the new user screen and inputs passwords that do not match, they must be given an error message and be prompted to try again.


New Patient

Patient Name: Newest patient

New Password: Testing

Confirm Password: Test

Submit

Error

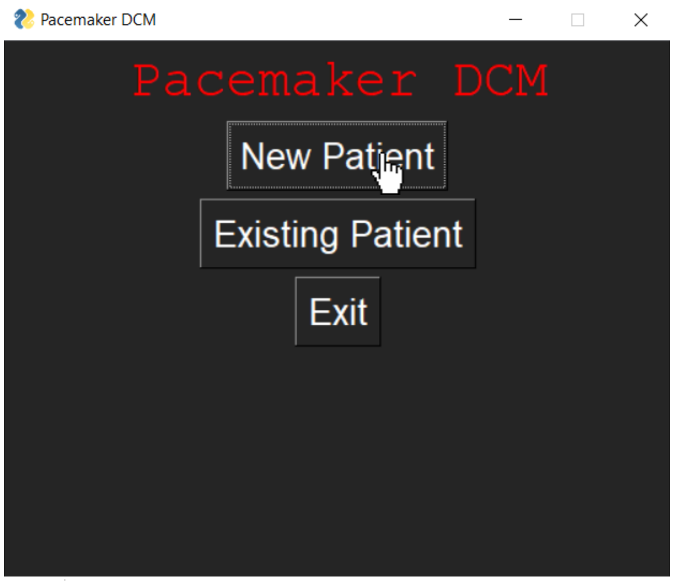
Passwords do not match

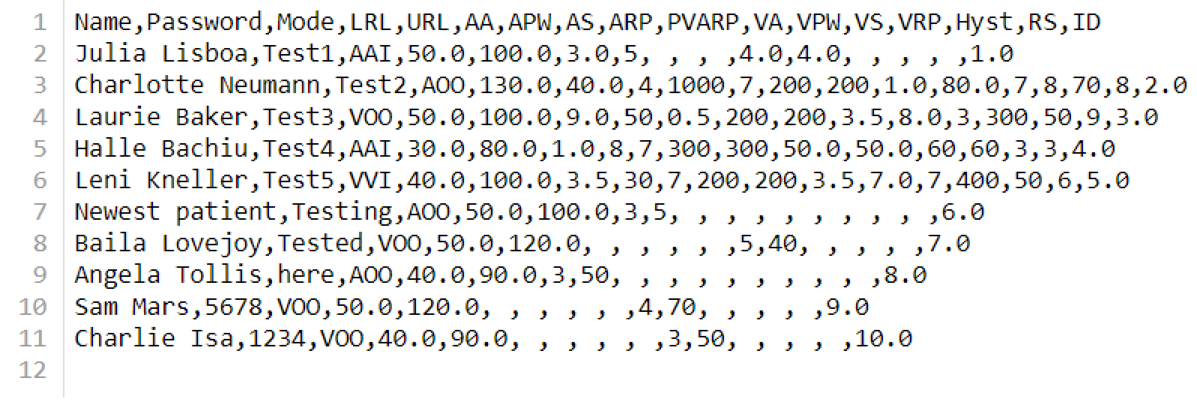
Try Again

Figure 27: Testing mismatched passwords.

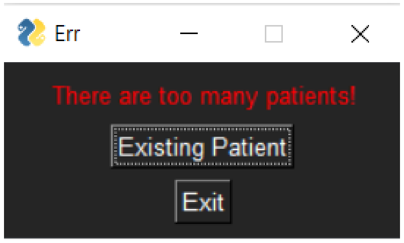
As shown above, that functionality works as expected.

The patient information is being stored in a csv file, we expect that if there are 10 entries already contained in the file the user will be given an error if they try to add another.





```
Name,Password,Mode,LRL,URL,AA,APW,AS,ARP,PVARP,VA,VPW,VS,VRP,Hyst,RS,ID
Julia Lisboa,Test1,AAI,50.0,100.0,3.0,5, , , ,4.0,4.0, , , , ,1.0
Charlotte Neumann,Test2,AOO,130.0,40.0,4,1000,7,200,200,1.0,80.0,7,8,70,8,2.0
Laurie Baker,Test3,VOO,50.0,100.0,9.0,50,0.5,200,200,3.5,8.0,3,300,50,9,3.0
Halle Bachiu,Test4,AAI,30.0,80.0,1.0,8,7,300,300,50.0,50.0,60,60,3,3,4.0
Leni Kneller,Test5,VVI,40.0,100.0,3.5,30,7,200,200,3.5,7.0,7,400,50,6,5.0
Newest patient,Testing,AOO,50.0,100.0,3,5, , , , , , , , , ,6.0
Baila Lovejoy,Tested,VOO,50.0,120.0, , , , , ,5,40, , , , ,7.0
Angela Tollis,here,AOO,40.0,90.0,3,50, , , , , , , , , ,8.0
Sam Mars,5678,VOO,50.0,120.0, , , , , ,4,70, , , , ,9.0
Charlie Isa,1234,VOO,40.0,90.0, , , , , ,3,50, , , , ,10.0
```



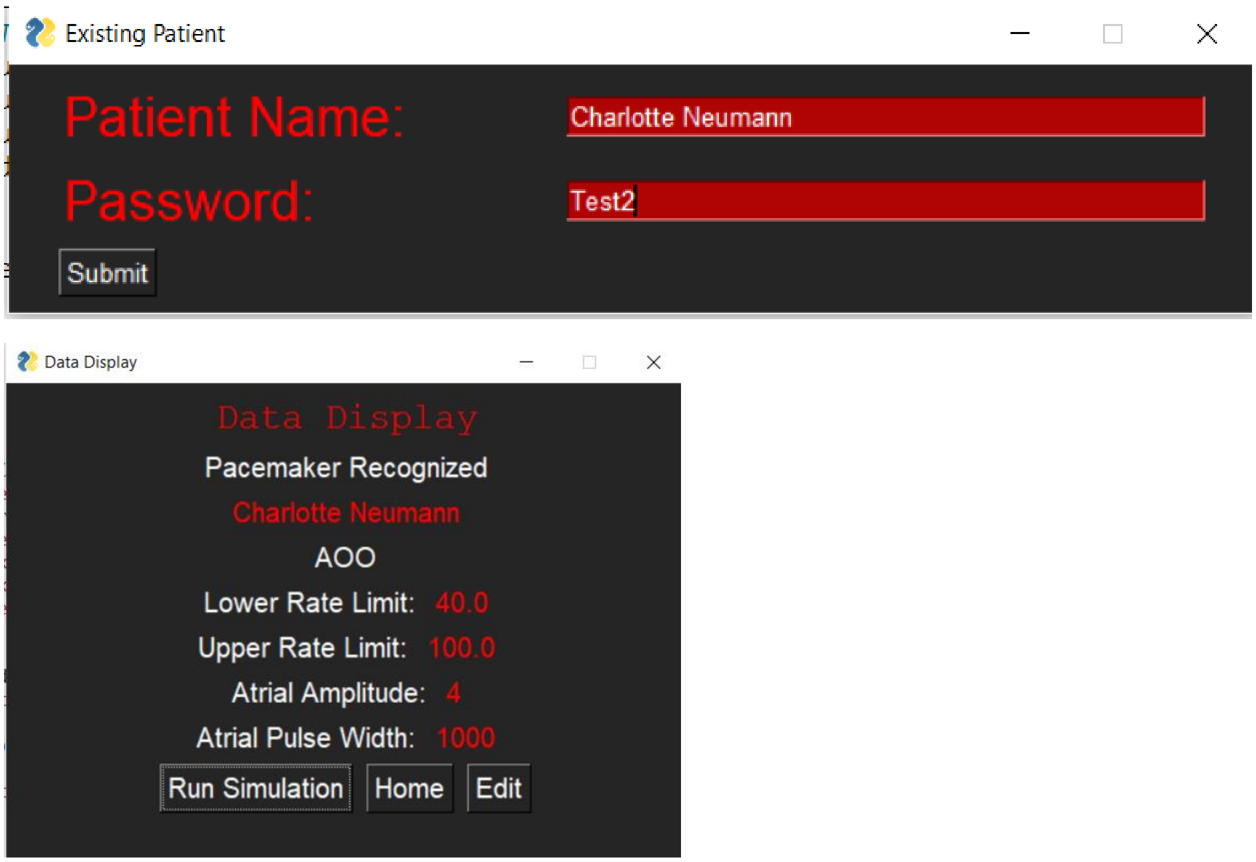
There are too many patients!

Figure 28: After filling the csv file with 10 entries, an error message shows up prompting a user to exit or login an existing patient.

As shown above, this functionality also works as expected.

5.2 – Existing Users

If the user chooses to open an existing patient's information, they provide the patient name and password. It is expected that when they enter a valid name and password found in the csv file, the program will find their information and display their current mode and stored parameters.

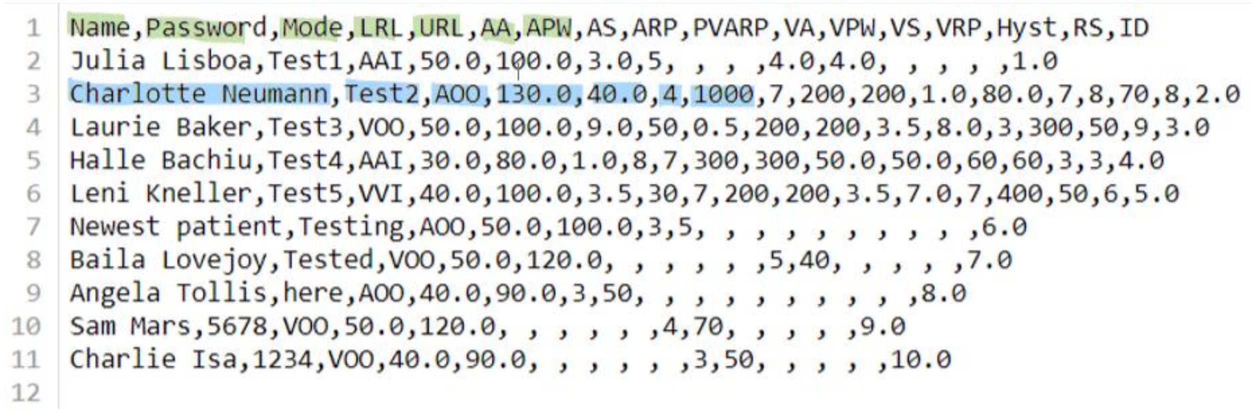


Patient Name: Charlotte Neumann
Password: Test2

Pacemaker Recognized
Charlotte Neumann
AOO
Lower Rate Limit: 40.0
Upper Rate Limit: 100.0
Atrial Amplitude: 4
Atrial Pulse Width: 1000

Figure 29: Logging in to an existing patient. The expected data display window appears.

The user login and display of data works as expected, and the data displayed matches the data held in the csv file here:

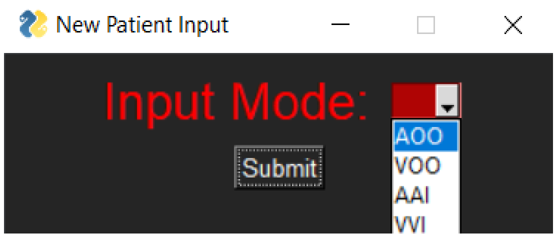


```
Name,Password,Mode,LRL,URL,AA,APW,AS,ARP,PVARP,VA,VPW,VS,VRP,Hyst,RS,ID
Julia Lisboa,Test1,AAI,50.0,100.0,3.0,5, , , ,4.0,4.0, , , , ,1.0
Charlotte Neumann,Test2,AOO,130.0,40.0,4,1000,7,200,200,1.0,80.0,7,8,70,8,2.0
Laurie Baker,Test3,VOO,50.0,100.0,9.0,50,0.5,200,200,3.5,8.0,3,300,50,9,3.0
Halle Bachiu,Test4,AAI,30.0,80.0,1.0,8,7,300,300,50.0,50.0,60,60,3,3,4.0
Leni Kneller,Test5,VVI,40.0,100.0,3.5,30,7,200,200,3.5,7.0,7,400,50,6,5.0
Newest patient,Testing,AOO,50.0,100.0,3,5, , , , , , , , , ,6.0
Baila Lovejoy,Tested,VOO,50.0,120.0, , , , , ,5,40, , , , ,7.0
Angela Tollis,here,AOO,40.0,90.0,3,50, , , , , , , , , ,8.0
Sam Mars,5678,VOO,50.0,120.0, , , , , ,4,70, , , , ,9.0
Charlie Isa,1234,VOO,40.0,90.0, , , , , ,3,50, , , , ,10.0
```

Figure 30: The user accessed stored in the csv file.

5. 3 – Mode selection

When there is a new patient or an existing patient, the user must select the mode. The window uses a drop-down menu and looks like this:



Or this:

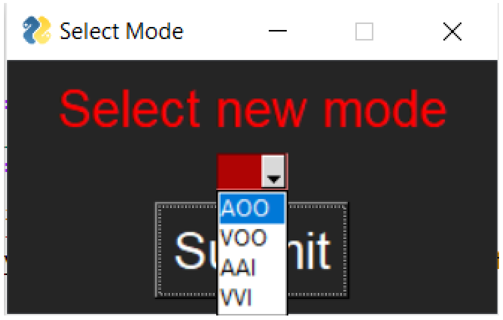


Figure 31: Possible mode input drop down menus.

It is expected that if the user does not select a mode and instead tries to submit the blank starter screen, an error window appears and prompts them to select a valid mode.

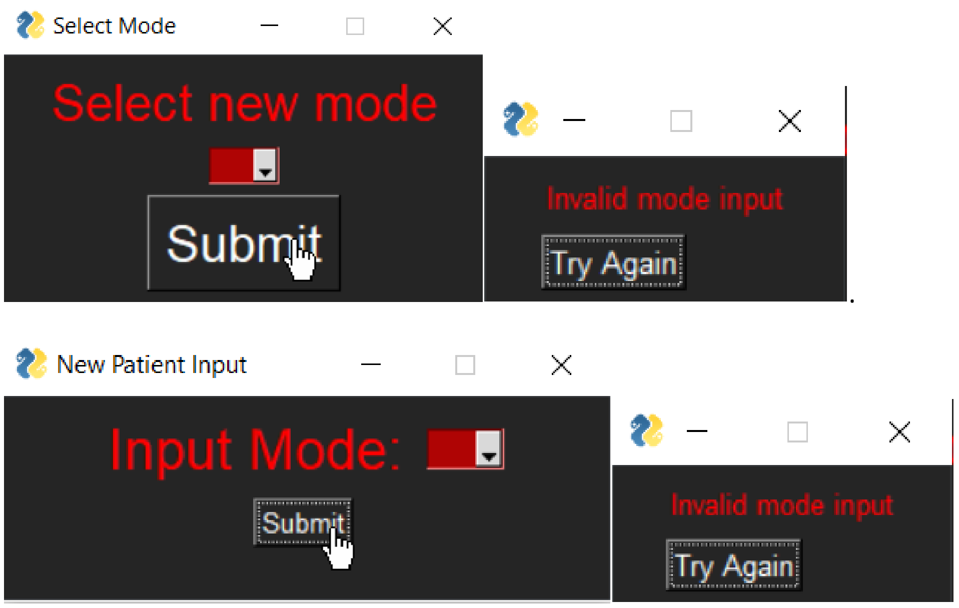
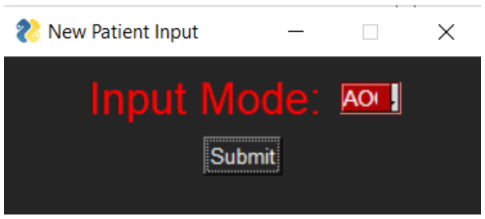


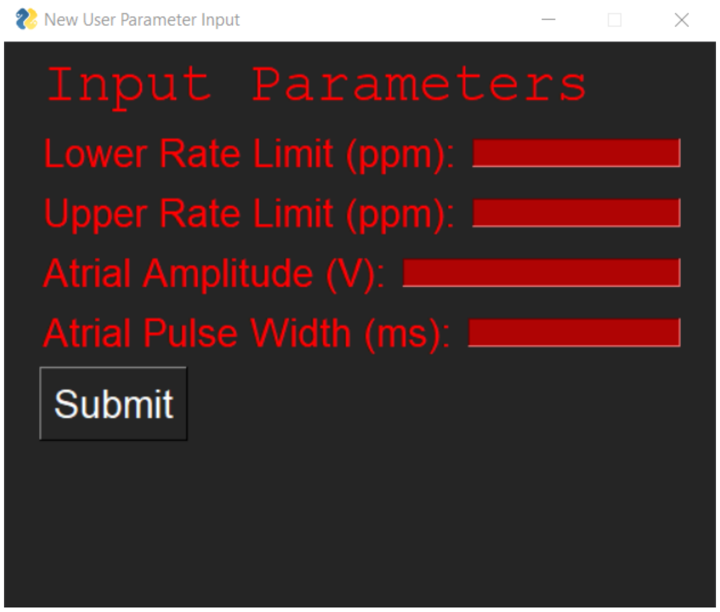
Figure 32: Invalid mode input error screen that appears when the mode input is blank

From Figure 32, it is evident that this is the case.

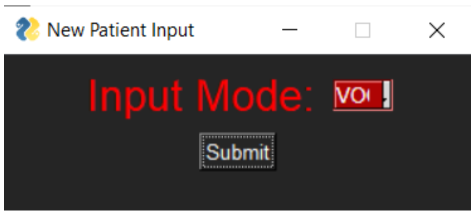
If it is a new user, it is expected that the mode they select with bring up a window with parameters to be entered corresponding to the mode. Those parameters are outlined in the PACEMAKER document. Examples are shown below:



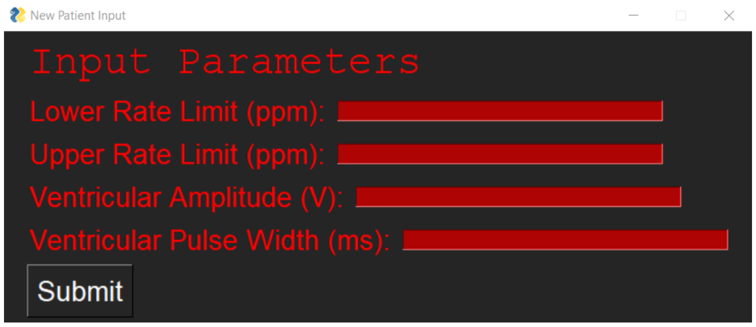
Input Mode: AOO



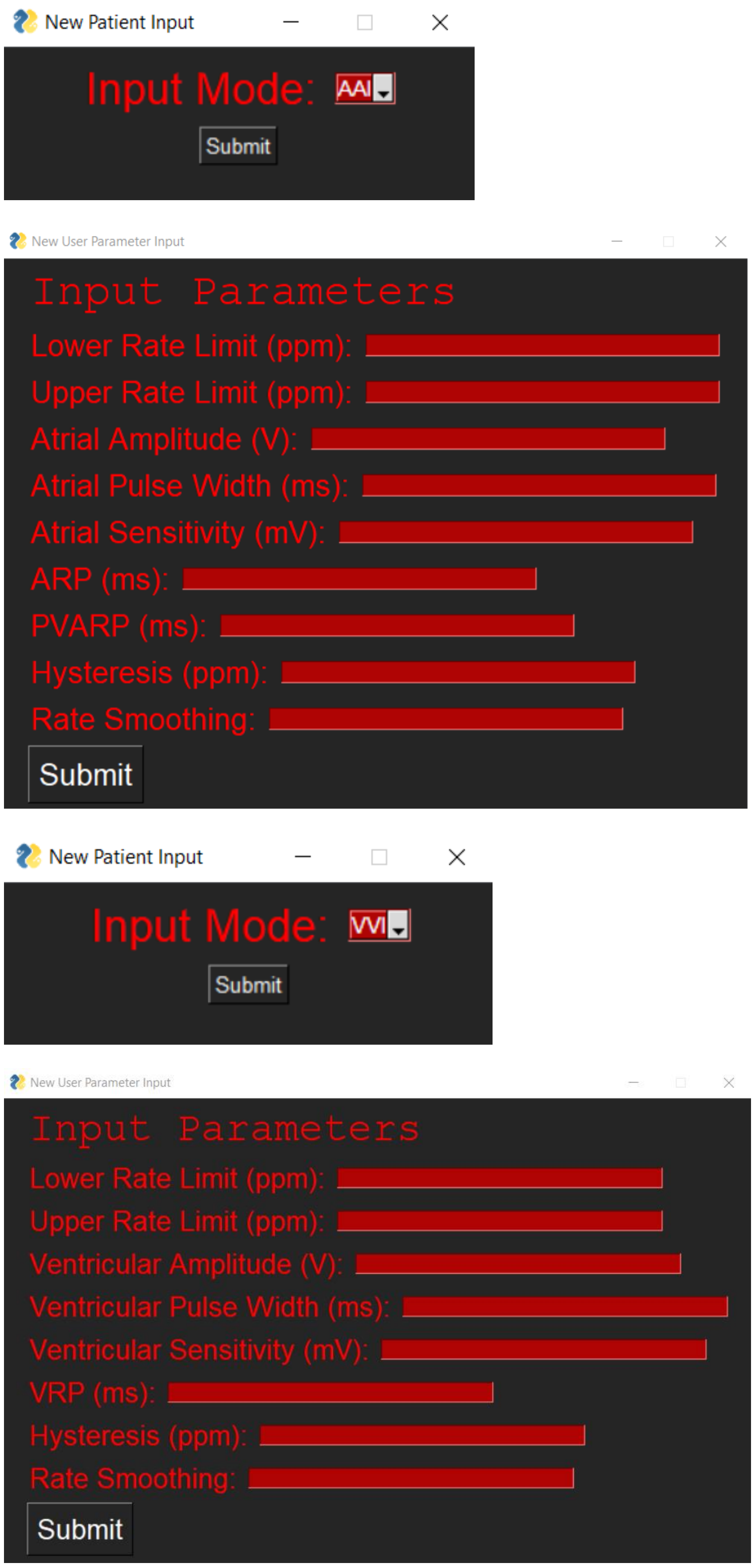
Lower Rate Limit (ppm):
Upper Rate Limit (ppm):
Atrial Amplitude (V):
Atrial Pulse Width (ms):



Input Mode: VOO



Lower Rate Limit (ppm):
Upper Rate Limit (ppm):
Ventricular Amplitude (V):
Ventricular Pulse Width (ms):


New Patient Input

Input Mode: AAI

Submit

New User Parameter Input

Input Parameters

Lower Rate Limit (ppm):

Upper Rate Limit (ppm):

Atrial Amplitude (V):

Atrial Pulse Width (ms):

Atrial Sensitivity (mV):

ARP (ms):

PVARP (ms):

Hysteresis (ppm):

Rate Smoothing:

Submit

New Patient Input

Input Mode: VVI

Submit

New User Parameter Input

Input Parameters

Lower Rate Limit (ppm):

Upper Rate Limit (ppm):

Ventricular Amplitude (V):

Ventricular Pulse Width (ms):

Ventricular Sensitivity (mV):

VRP (ms):

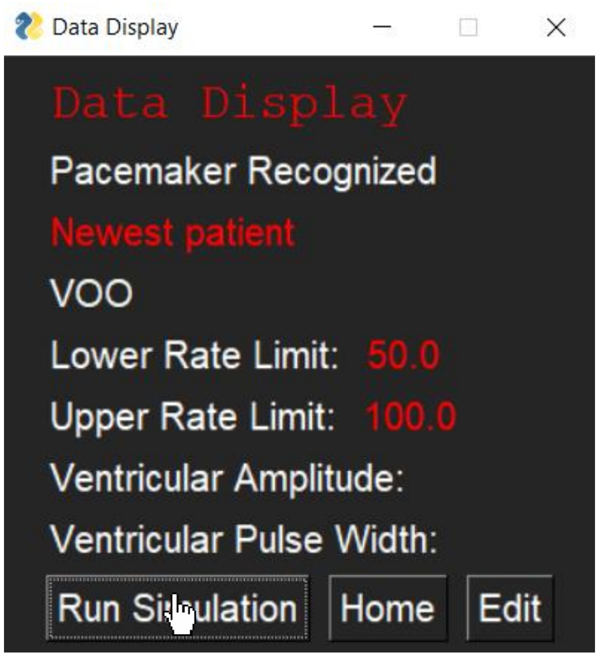
Hysteresis (ppm):

Rate Smoothing:

Submit

Figure 33: Expected user information field entering based on selected mode.

If it is an existing patient, it is expected that the user picking a new mode will then display the parameters currently stored for that mode. For example:


Data Display

Data Display

Pacemaker Recognized

Newest patient

VOO

Lower Rate Limit: 50.0

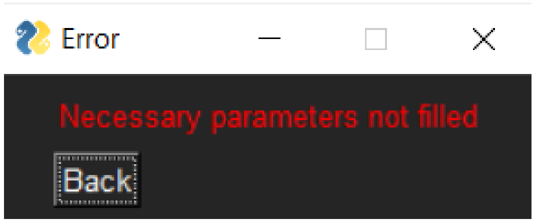
Upper Rate Limit: 100.0

Ventricular Amplitude:

Ventricular Pulse Width:

Figure 34: Data display window for patient for missing fields.

This user did not originally enter in VOO mode so they do not have parameters saved for VA or VPW. If they try to run a simulation we expect that an error will arise because the necessary parameters are not filled.


Error

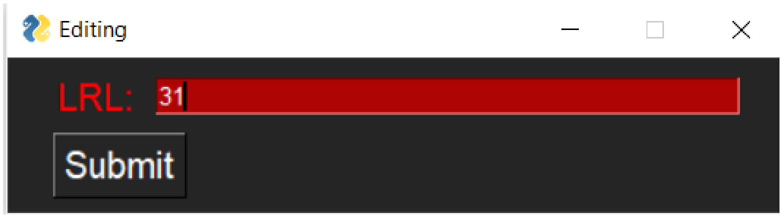
Necessary parameters not filled

Figure 35: Error message indicating some parameters are missing upon simulation running.

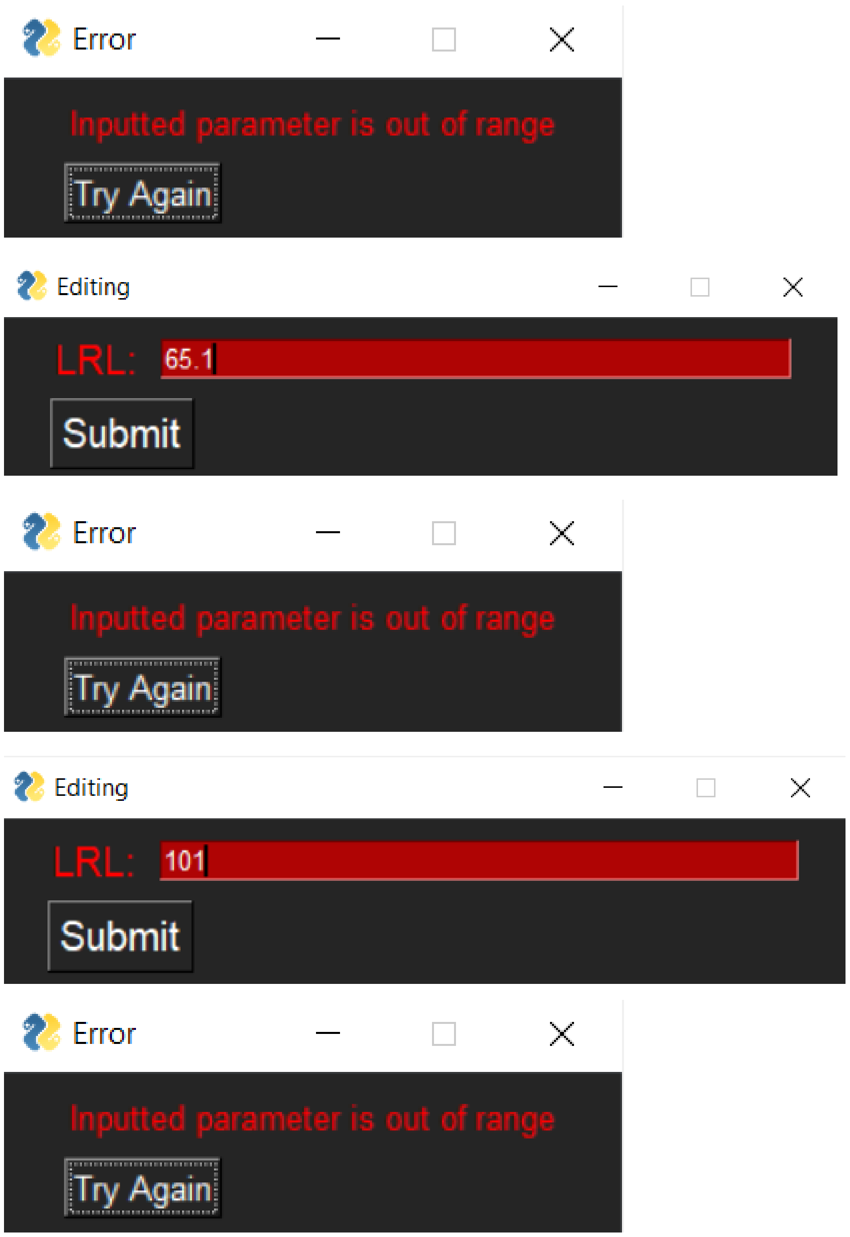
From the above example, it is evident that this is the case.

5.4 – Parameter Entry

No matter when values for parameters are being entered into the program, they are put through the same error checking function. The allowable ranges for each parameter being used are outlined in the PACEMAKER document and in Section 1.4. When the user tries to edit any of the parameters they can do it one at a time or choose as many as they would like to edit at once. We can show here the accuracy of each parameter's error catching.


Editing

LRL: 31


Error

Inputted parameter is out of range

Editing

LRL: 65.1

Error

Inputted parameter is out of range

Editing

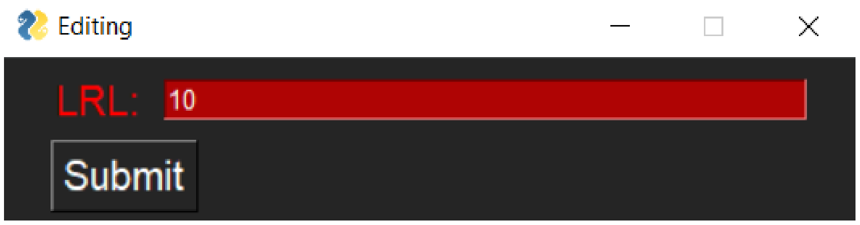
LRL: 101

Error

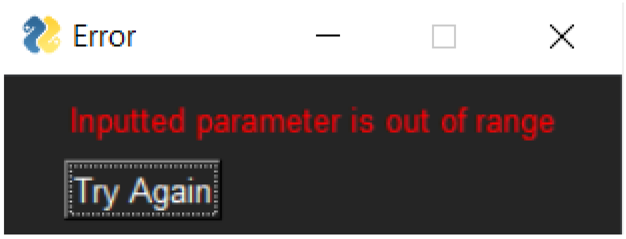
Inputted parameter is out of range

Figure 36: Entering test cases of various parameter values to test LRL error catching.

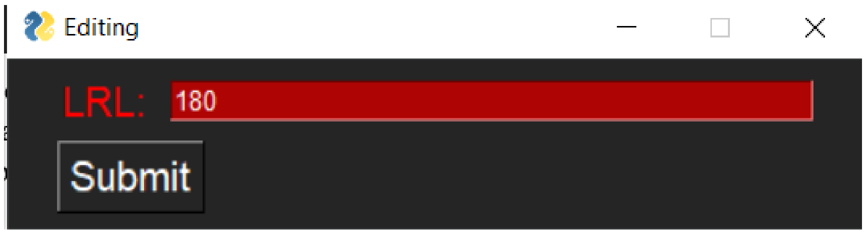
The above examples show the program catching errors in the allowable increments in the multiple allowable ranges for the Lower Rate Limit. The following will show the program catching numbers that are completely out of the allowable ranges.


Editing

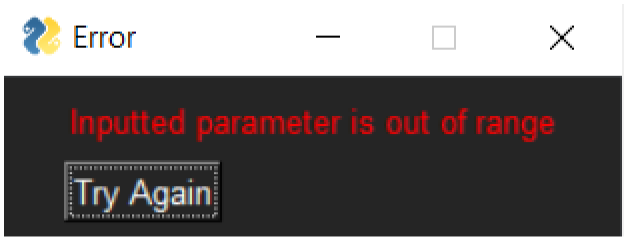
LRL: 10


Error

Inputted parameter is out of range


Editing

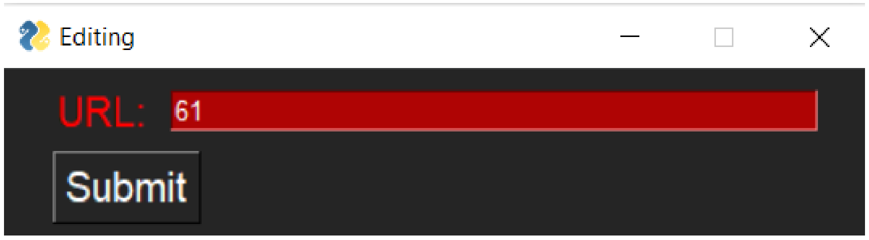
LRL: 180


Error

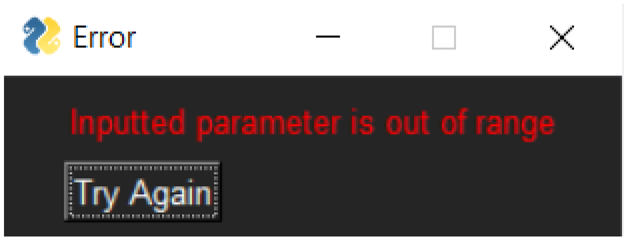
Inputted parameter is out of range

Figure 37: Testing further LRL test cases with values out of the range.

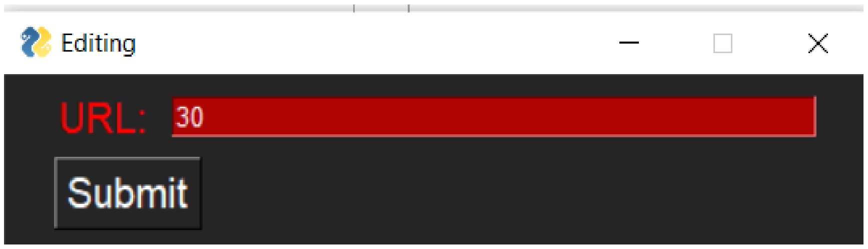
The same tests can be done with the Upper Rate Limit, its accuracy is proven below:


Editing

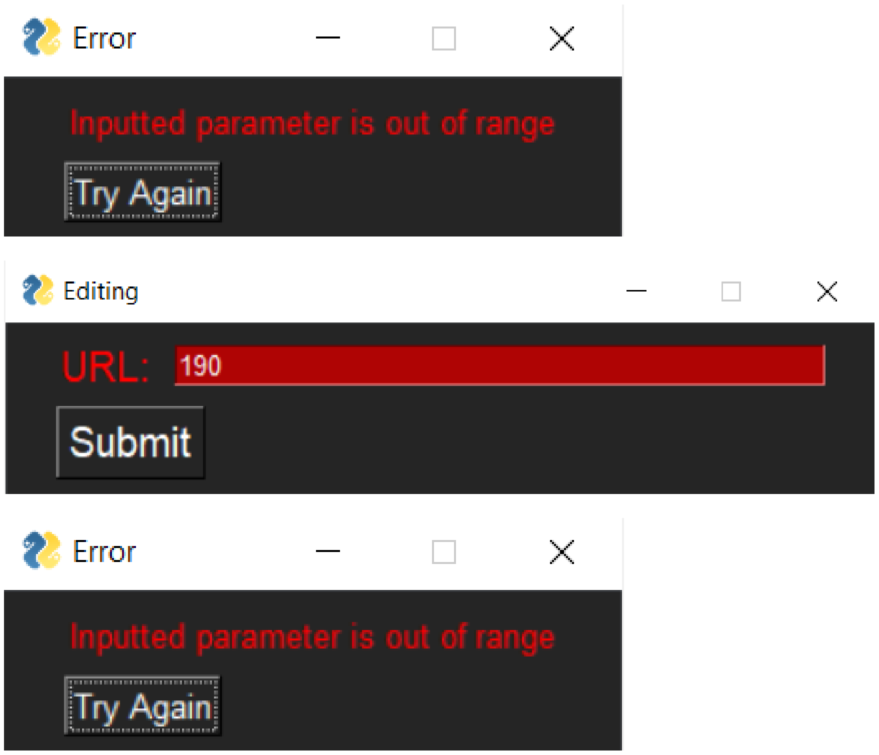
URL: 61


Error

Inputted parameter is out of range


Editing

URL: 30



Inputted parameter is out of range

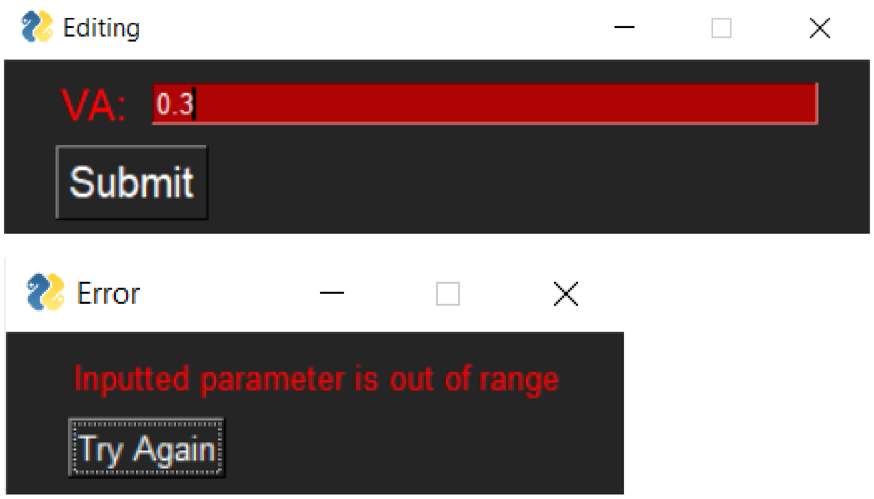
URL: 190

Inputted parameter is out of range

Figure 38: URL test cases and error messages.

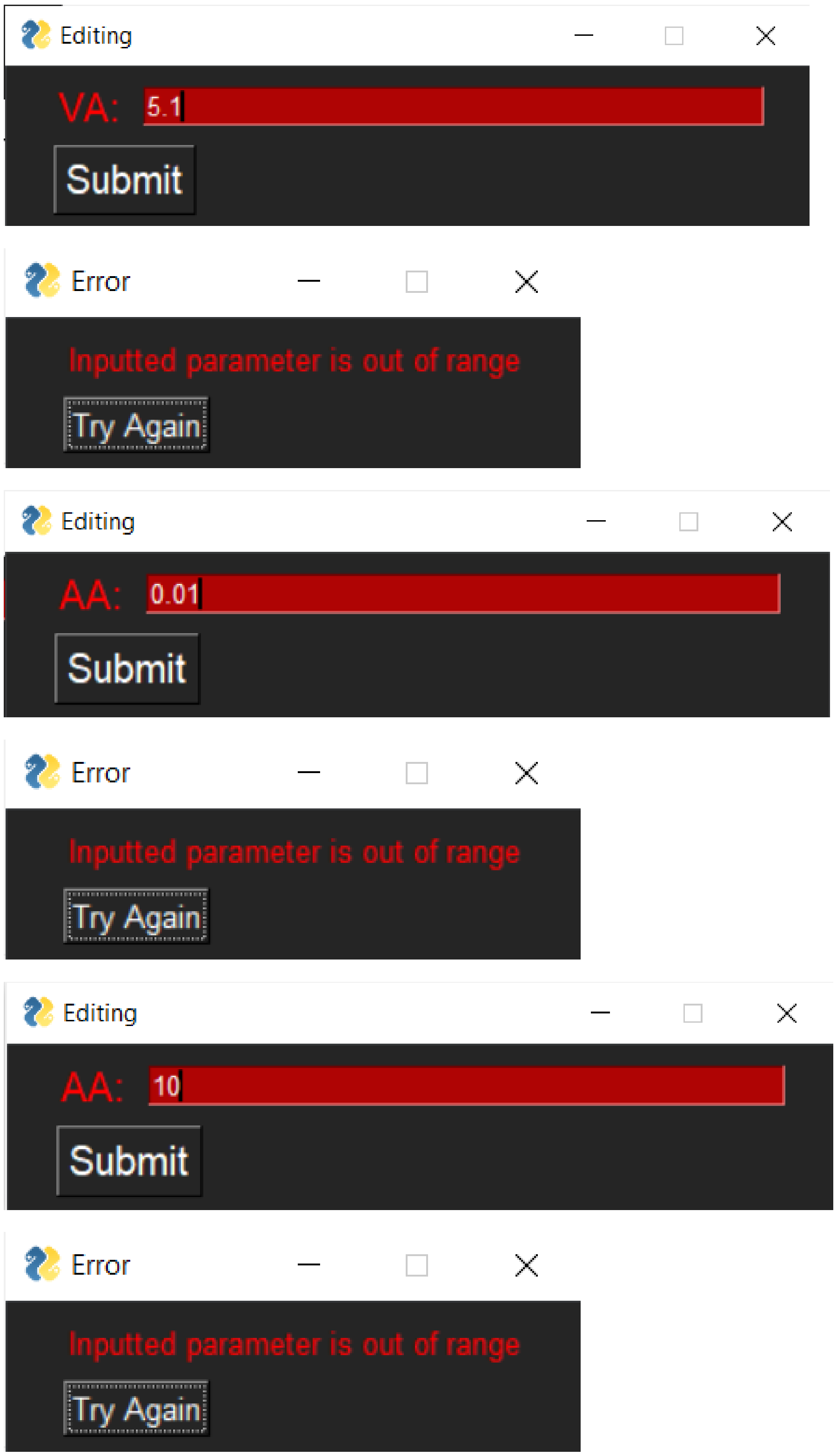
The program catches errors within the range but with the incorrect increment as well as parameters completely out of the allowable range.

The same thing for the Atrial and Ventricular Amplitudes, they do not have allowable increments so the program just catches parameters out of the allowable range.



VA: 0.3

Inputted parameter is out of range


Editing

VA: 5.1

Submit

Error

Inputted parameter is out of range

Try Again

Editing

AA: 0.01

Submit

Error

Inputted parameter is out of range

Try Again

Editing

AA: 10

Submit

Error

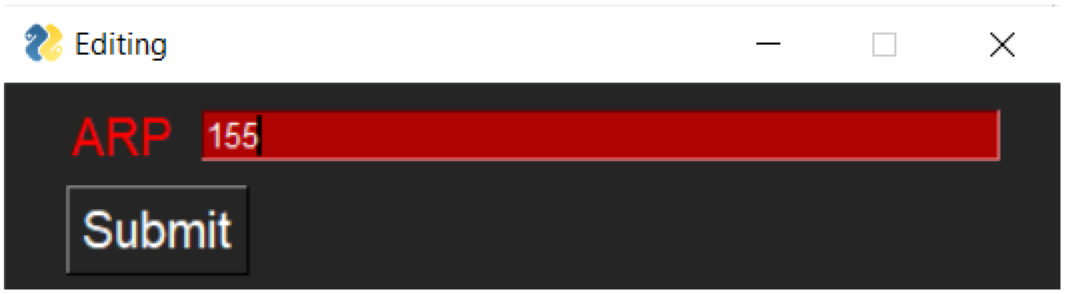
Inputted parameter is out of range

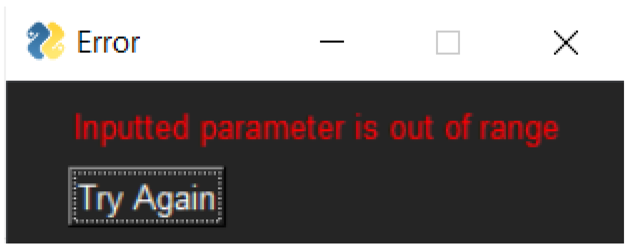
Try Again

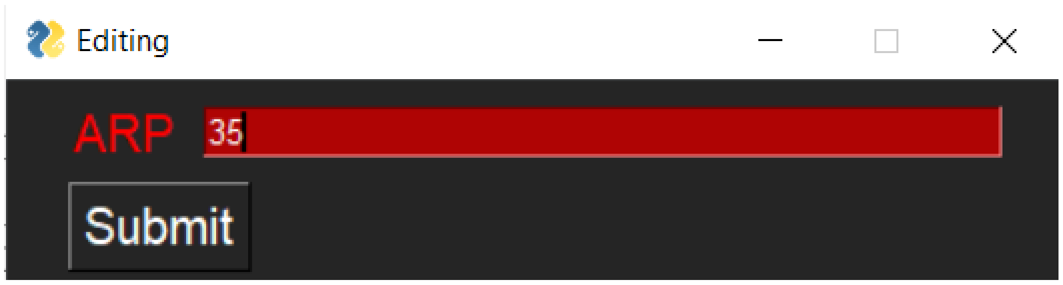
Figure 39: Inserting incorrect atrial pulse width bounds and displaying error messages.

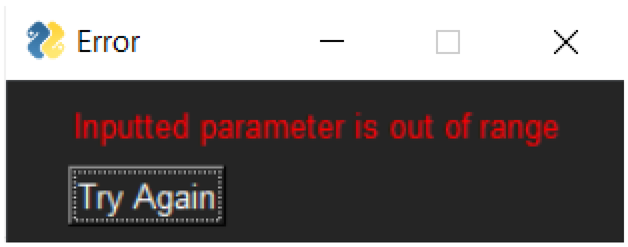
The next parameter we could test would be Atrial and Ventricular Pulse Width but per TA instruction, they do not have error catching implemented in this assignment.

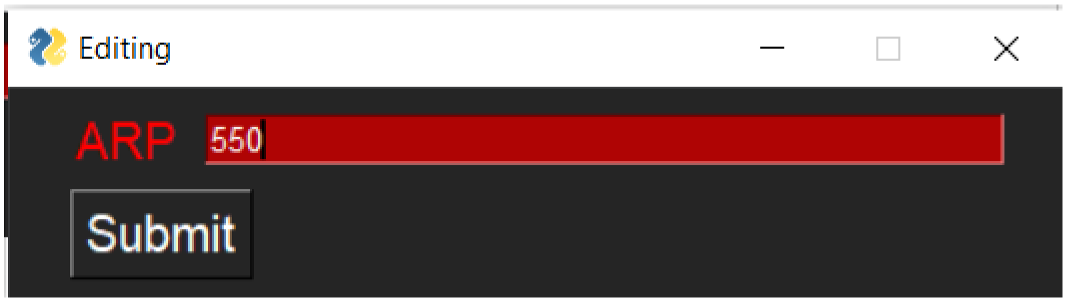
The parameters we can test next are the Atrial and Ventricular Refractory period as well as the PVARP. As shown below, the error catching enforces the increments required as well as the parameters being within the allowable range.

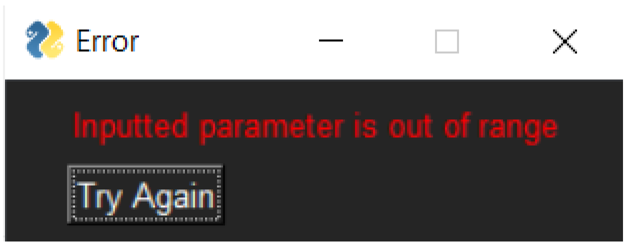




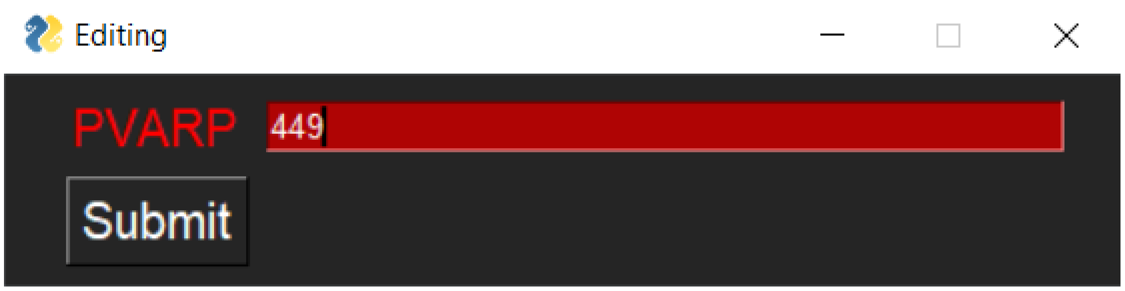




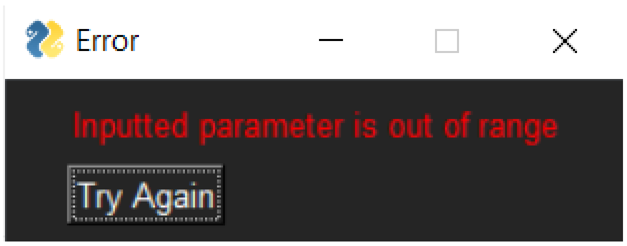




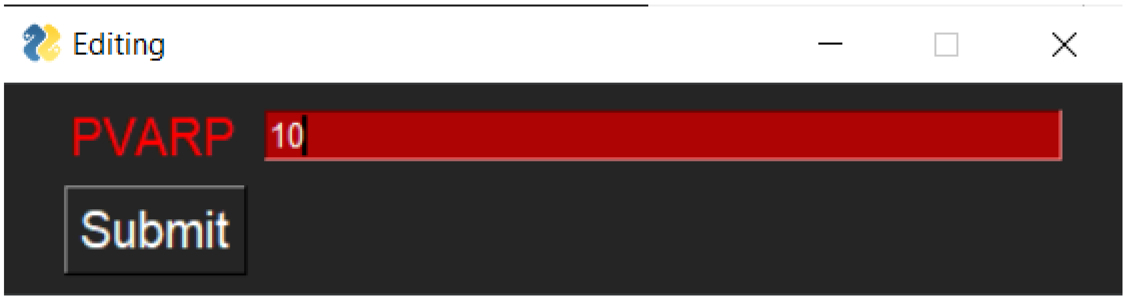
Inputted parameter is out of range



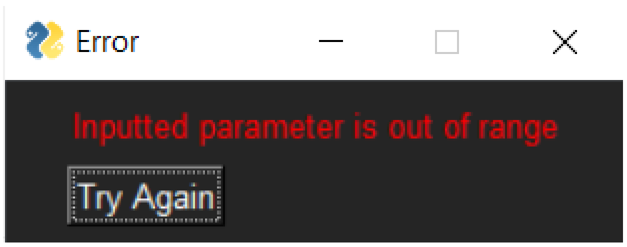
PVARP 449



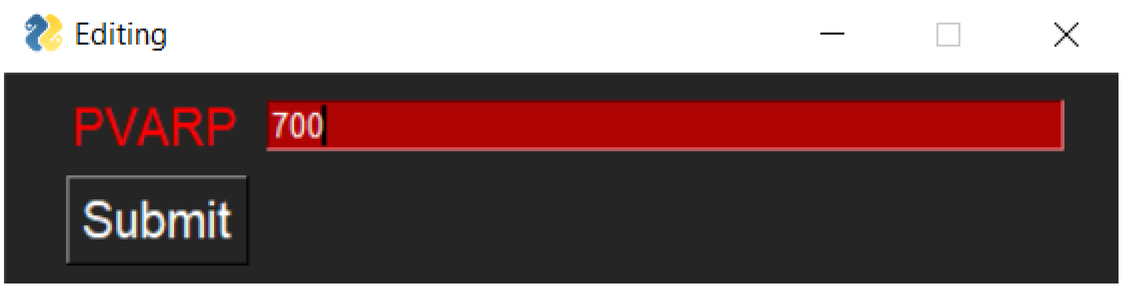
Inputted parameter is out of range



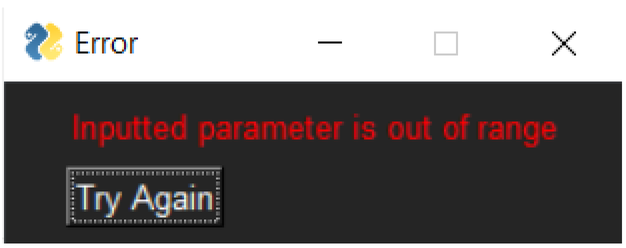
PVARP 10



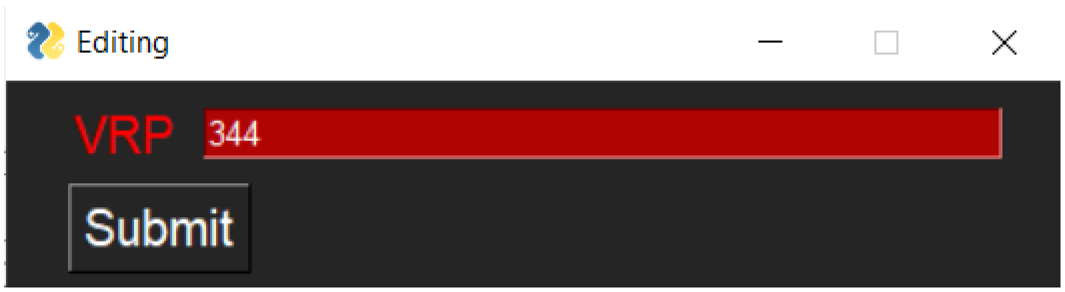
Inputted parameter is out of range



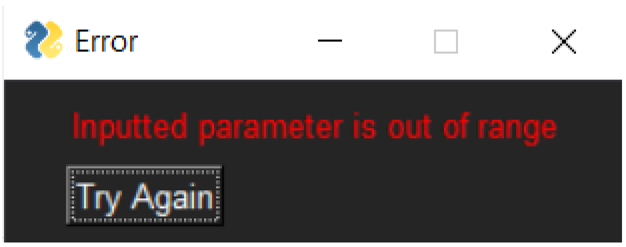
PVARP 700



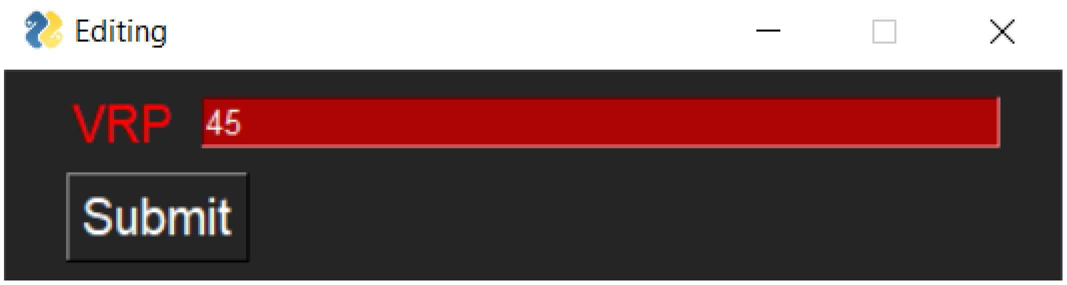
Inputted parameter is out of range



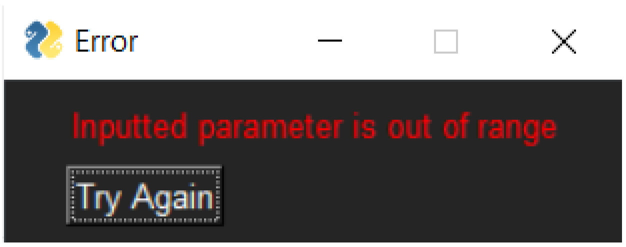
VRP 344



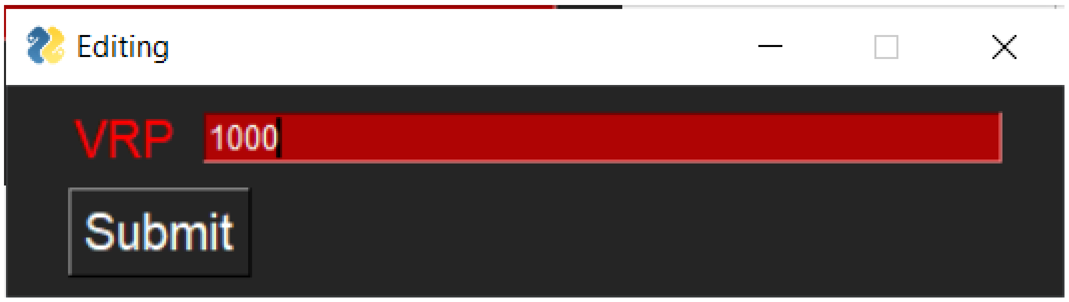
Inputted parameter is out of range



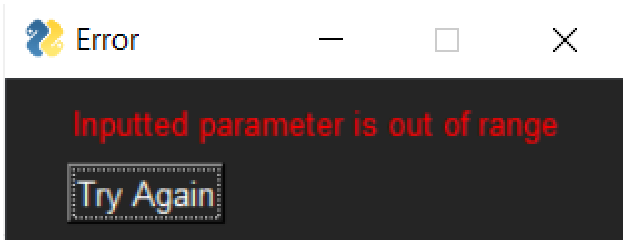
VRP 45



Inputted parameter is out of range



VRP 1000

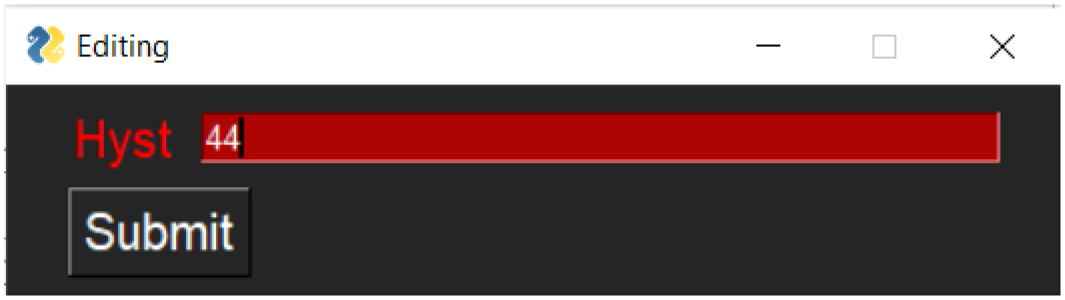

Error

Inputted parameter is out of range

Try Again

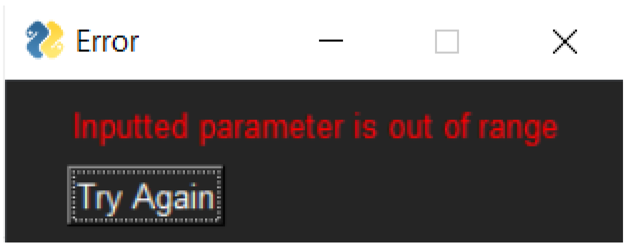
Figure 10: Testing refractory period parameter ranges.

The last parameter to test is Hysteresis Rate Limit, its cases of being tested within the range with incorrect increment as well as out of range parameters is shown below.


Editing

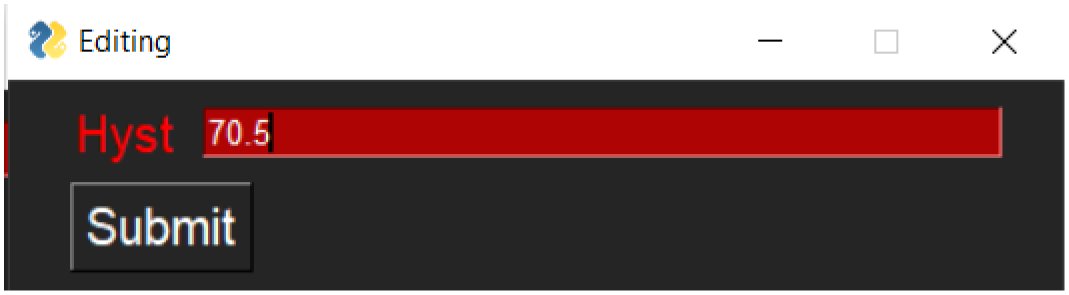
Hyst 44

Submit


Error

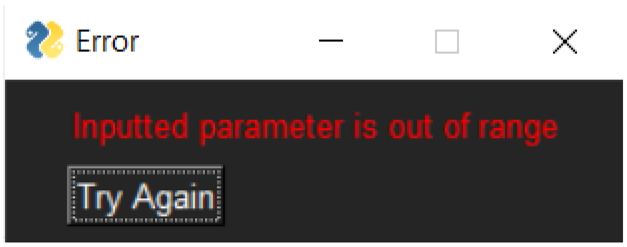
Inputted parameter is out of range

Try Again


Editing

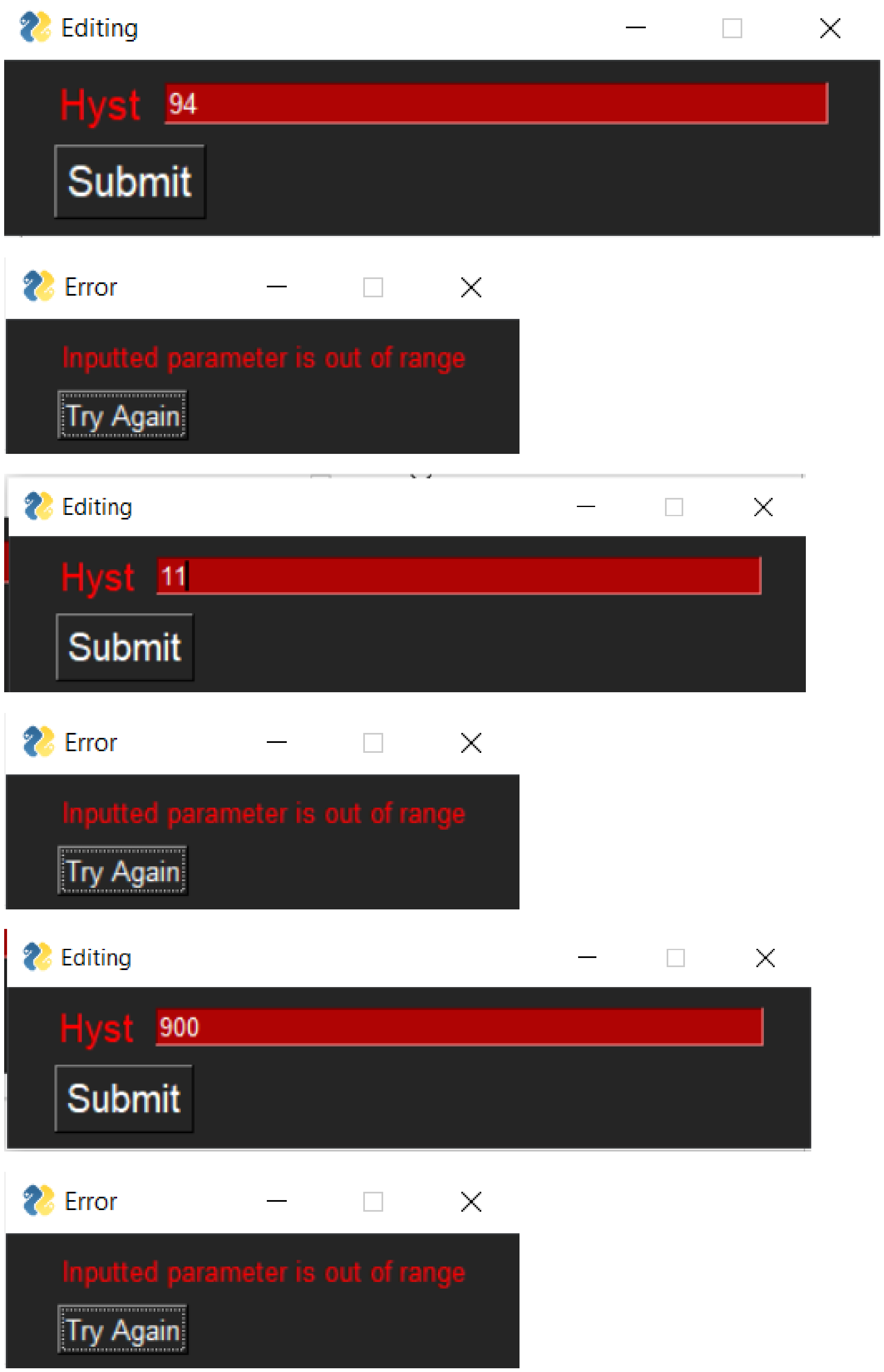
Hyst 70.5

Submit


Error

Inputted parameter is out of range

Try Again



Hyst 94

Inputted parameter is out of range

Hyst 11

Inputted parameter is out of range

Hyst 900

Inputted parameter is out of range

Figure 11: Testing hysteresis parameter ranges.

Since every time the parameters are entered into the program, they are put through the same error catcher, the above cases should adequately prove that the error checking works whether they are being edited or entered for the first time.

5.5 – Pacemaker Approaching

The program consists of functionality to check if the pacemaker device has been approached before. However, since no hardware is connected yet, there is no reason to test this functionality until a connection is established.

References

PySimpleGUI. *PySimpleGUI*. 24 October 2021. 22 October 2021. <https://pypi.org/project/PySimpleGUI/>.

Yang, Jason. *stack overflow*. 5 January 2021. 19 October 2021. <https://stackoverflow.com/questions/65575861/waiting-screen-while-data-is-being-extracted-from-an-api>.