

# 3K04 | Assignment 1 & 2

Group 24 | Members:

Deborah Udokwe

Sherry Liu

Joelle D'Amico

Khawja Labib

Ankit Aggarwal

Saliha Hussain

# Table of Contents

<b>Requirements.....</b>	<b>4</b>
Graphical User Interface.....	4
Welcome, Login, & Account Creation.....	4
Main Selection Menu.....	4
Programmable Parameters.....	4
Patient Data Storage.....	4
Username & Password.....	4
Programmable Parameters.....	5
<b>Potential Requirement Modifications.....</b>	<b>6</b>
Graphical User Interface.....	6
Delete User Functionality.....	6
Serial Communication.....	6
Pacing Modes.....	6
Patient Data Storage.....	6
<b>Design Decisions.....</b>	<b>7</b>
Graphical User Interface.....	7
Python & TKinter.....	7
User Interface Pages & Formatting.....	7
User Interface Page Transitions.....	8
Data Storage.....	8
Patients.....	8
Programmable Parameters.....	8
<b>Potential Design Modifications.....</b>	<b>9</b>
Graphical User Interface.....	9
Patient Data Storage.....	9
<b>Modules.....</b>	<b>9</b>
Module One   main_3k04.py.....	9
Function   main().....	9
Class   mainWindow.....	10
Class   welcome_page.....	10
Class   user_page.....	11
Class   main_page.....	12
Class   set_date_time_page.....	12
Class   parameters_page.....	13
Module Two   patient.py.....	13

Class   Patient.....	13
<b>Testing &amp; Results.....</b>	<b>14</b>
Summary of Roadblocks.....	14
Welcome & Login   Testing.....	17
User Creation   Testing.....	23
Main Menu   Testing.....	25
Parameters Page   Testing.....	30
<b>Assignment Two   Requirements.....</b>	<b>35</b>
<b>Assignment Two   Potential Requirement Modifications.....</b>	<b>35</b>
<b>Assignment Two   Design Decisions.....</b>	<b>35</b>
<b>Assignment Two   Case Assurance.....</b>	<b>35</b>
<b>Assignment Two   Potential Design Modifications.....</b>	<b>35</b>
<b>Assignment Two   Modules.....</b>	<b>35</b>
Module Three   serial_comm.py.....	35
Module Four   global_vars.py.....	36
<b>Assignment Two   Testing &amp; Results.....</b>	<b>36</b>

# Requirements

Overall, the necessary requirements as of Assignment 1 for the DCM include having a graphical user interface which is capable of creating a maximum of ten patients, the ability to store patient data, and set programmable parameters for different heart modes. Further details are described below:

## Graphical User Interface

### Welcome, Login, & Account Creation

The graphical user interface must have a welcome screen that allows the user to type an existing username and password to access the pacemaker application. The welcome page must also provide a new user with the ability to create a new account. The welcome page should not allow a user to create additional users if the maximum number of ten users is reached.

### Main Selection Menu

The graphical user interface must set date and time for reports, have the ability to end the currently running telemetry, and the ability to read about the pacemaker device connected. This page must also display when there is a valid connection to the pacemaker. Note that all of these requirements are provisionary requirements rather than functional requirements as connection to the hardware will be completed in future assignments which will enable further functionality.

### Programmable Parameters

The graphical user interface must have a screen in which the user is able to choose from heart pacing modes AOO, VOO, VVI, and AAI. After selecting the heart pacing mode, the user must be able to set the respective programmable parameters for the mode selected. Default parameters must be set if the user does not input their own values.

## Patient Data Storage

### Username & Password

Must be able to save the newly created patients username and password combinations locally on the DCM database such that when a user exits the program and they are able log back in upon reopening the application. A maximum of ten users can be stored.

### Programmable Parameters

Must be able to set parameters for the four pacing modes AOO, VOO, AAI, VVI and review them once selected. The current parameters present in the design are

S.No	Parameter	Programmable Values	Increment
1	Lower Rate Limit	30 - 50 ppm 50 - 90 ppm 90 - 175 ppm	5 ppm 1 ppm 1ppm
2	Upper Rate Limit	50 - 175 ppm	5 ppm
3	Atrial or Ventricular pulse Amplitude	Off, 0.5 - 3.2 V 3.5v - 7 V	0.1 V 0.5 V
4	Atrial or Ventricular Pulse Width	0.05 ms 0.1 - 1.9 ms	- 0.1 ms
5	Atrial or Ventricular Sensitivity	0.25, 0.5, 0.75 1 - 10 mV	- 0.5mV
6	Atrial Refractory Period	150 - 500 ms	10 ms
7	Ventricular Refractory Period	150 - 500 ms	10 ms
8	PVARP	150 - 500 ms	10 ms
9	Hysteresis	Off, 30 - 50 ppm 50 - 90 ppm 90 - 175 ppm	- 5 ppm 1 ppm 1ppm

10	Rate Smoothing	Off, 3, 6, 9, 12, 15, 18, 21, 25	-
----	----------------	----------------------------------	---

## Potential Requirement Modifications

### Graphical User Interface

#### Delete User Functionality

A future update to requirements may be the addition of the ability to delete users from the graphic user interface. Currently the user is able to delete patients directly from the patientData.csv file. To create a more robust program, it would be important to have this functionality entirely on the front end of the application without the user needing to directly access the backend database to remove users when the maximum number of users is reached.

#### Serial Communication

A future update to requirements will include several aspects of serial communication updates. Functionality will need to be provided to the main page which displays a non-functional temporary label that notifies the user if the pacemaker device is successfully connected and ready for serial communication. Additionally, functionality will need to be provided to the programmable parameters such that the values can be set in the graphical user interface and the user's set values are reflected in the pacemaker heartview application after being serially communicated to the hardware. Analog response from the NXP FRDM-K64F board to signify successful communication would be useful to verify DCM to hardware communication is working as expected.

#### Pacing Modes

A future update to requirements will include the addition of at least four more pacing modes namely AOOR, VOOR, AAIR, VVIR and the addition of various parameters to measure the aforementioned pacing modes.

## Patient Data Storage

- Able to delete user and user data from .csv
- Additional Modes

## Design Decisions

While reviewing the requirements of this assignment, several design decisions needed to be made.

## Graphical User Interface

### Python & TKinter

For this project we decided to use Python and the TKinter framework to create the DCM. The reason we chose python to create the DCM is because it has simple syntax which is useful for rapid prototyping and libraries (PySerial) that support serial communication which will be useful for the second assignment. TKinter was chosen as the GUI framework because it is a package already bundled with python and has a significant amount of documentation to develop from.

### User Interface Pages & Formatting

It is very important for the user interface to be friendly and straightforward while at the same time having the least amount of coding blocks and lesser time complexity. While there are many other factors at play as well, we decided to focus on the one's listed above.

For a better experience we added multiple pages with adequate spacing between the buttons/dropdown menus so the user could easily equip themselves with the application, multiple pop ups to display errors and success messages were added for the same reason.

Our program consists of 4 pages -

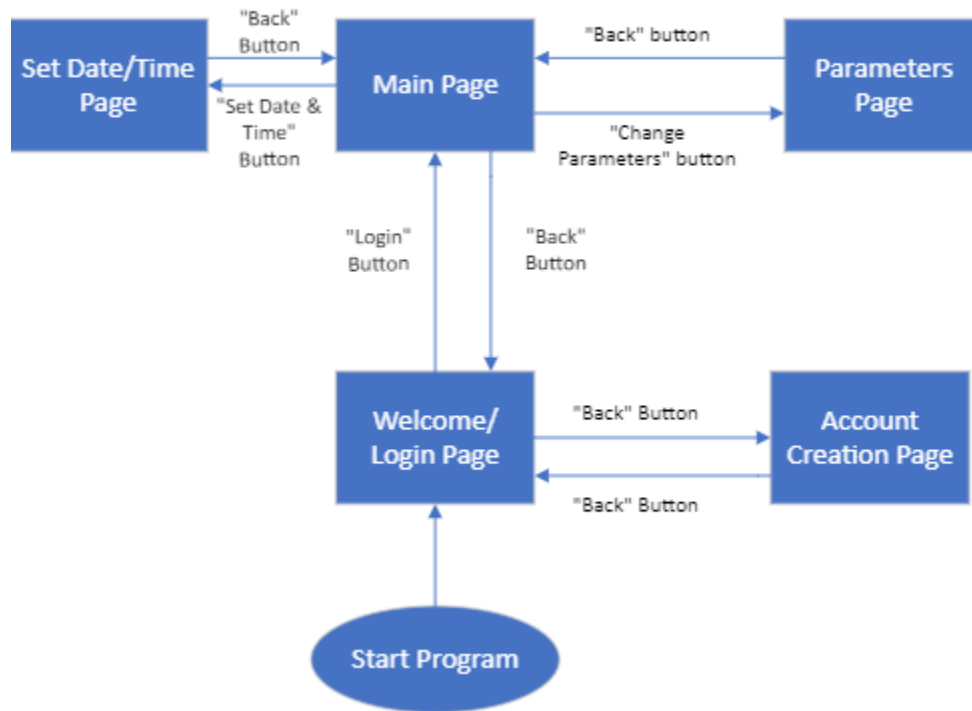
Page 1 - Login page : Allows the user to login an existing user(separate page)/ check current available slots for creating a user/ creating a new user (separate page)

Page 2 - Create a new user : Allows the user to create a new ID by entering a unique username and password

Page 3 - Main Page : Allows the user to see the about menu, set date and time (on a new pop up screen), end telemetry session, return to the login page (sign out) or go to the next page to change pace modes

Page 4 - Parameters Page : Allows the user to choose a pacing mode and based on the same choose appropriate parameters. In order to save time only one page was created for the same and the parameter features not in use (based on the pacing mode) were disabled.

## User Interface Page Transitions



## Data Storage

### Patients

Patients are to be implemented as objects with username and password attributes. It was decided that patient username and password attributes would be stored respectively in a patientData.csv file that is generated when the first patient is created on a local system and further appended to when additional patients are created. The comma-separated values text file format was chosen as our database format because data in this format is easily accessible through columns and rows related to a header name.

Patient usernames, passwords, and



## Programmable Parameters

The parameters are stored in a dictionary with keys named after the parameters themselves. The programmable parameters once set and saved by the user are currently only available during the lifetime of the program run.

## Potential Design Modifications

In the future, we are required to set an interface between the pacemaker and DCM using serial communication.

In order to achieve this some design decisions may need modification.

In order to make the UI more user friendly we would need to display the saved parameters values and have an option to edit the users

## Graphical User Interface

In the future, we would need to add an option to the login/main page to edit and delete the current users to avoid opening the csv file for doing the same

On the main page, a button would be enabled to show the last saved pacing mode parameters for that user.

The message reading 'Device not connected' would need to change when the pacemaker is connected to the DCM thus its design would need to be modified possibly into a button or a pop up.

## Patient Data Storage

In the future, the patient data module would have to be updated to have an additional attribute to save the parameters last set by the user in the form of a dictionary which is currently being printed on the python console

## Modules

### Module One | main\_3k04.py

The purpose of this module is to display the graphical user interface to the user. This module holds all of the visual frames that the user is guided through depending on their conditional action that they make with the application. This front end module interacts with the patient.py module for backend logic. The following functions, classes, and methods are all part of this module to create a seamless user experience and friendly developer backend.

## Function | main()

The main function calls on patient.py createPD() function to create the database patientData.csv file if not already created. The function then creates a main loop of the pacemaker DCM application to run continuously when the user starts the program.

## Class | mainWindow

Method	Inputs	Responsibility
__init__	self,*args,**kwargs	<p>Black Box: Creates Window for Pacemaker DCM Application.</p> <p>This method is an instantiation of a TKinter Window. This creates a page for the frames to be displayed to the user. The frames are loaded onto the TKinter window with display_page function.</p>
display_page	Self and instance of a frame object, defined as 'display'	<p>This method is the general method used to switch the current that the user is on to a different page. The name of the page is the frame object which is used as the input to this function. When this function is called ie. display_page(welcome_page) , the welcome page will be shown to the user.</p>
on_login	Self and instance of integer variable, defined as 'state_status'.	<p>This method takes output from the patient.py loginUser function as an input to determine what to display to the user based on the username and password they are attempting to log in with.</p>
on_create	Self and instance of integer variable, defined as 'max_val'	<p>This method takes output from the patient.py availUsers function as an input to display a to the user if the maximum number of patients is reached when attempting to create a new user.</p>
logTime	Self	<p>This method is used to check a calendar and log the current time for future telemetry data reports.</p>

## Class | welcome\_page

Private Method	Inputs	Responsibility
__init__	self, parent, controller	<p>Black Box: Displays Welcome Page to User</p> <p>This method is an instantiation of the TKinter frame widget. The controller allows for low coupling design as frames can easily be accessed and controlled through the root TKinter window with 'controller' variable rather than directly accessing the TKinter window itself.</p> <p>This frame reveals the welcome page to the user with private variables inputUser and inputPass to get user text box information. Variable values are passed as inputs for patient.py backend functions.</p>
checkAvail	No Inputs	<p>Calls the availUsers function from patient.py and displays the current number of available users when user presses the "New Patient Slots Available" button.</p>

## Class | user\_page

Private Method	Inputs	Responsibility
__init__	self, parent, controller	<p>Black Box: Displays Account Creation page to User</p> <p>This method is an instantiation of the TKinter frame widget. The controller allows for low coupling design as frames can easily be accessed and controlled through the root TKinter window with 'controller' variable rather than directly accessing the TKinter window itself.</p> <p>This frame reveals the account creation page to the user with private variables inputPassnew, inputPassVal, and inputUsernew to get user text box information. Variable values are passed as</p>

		inputs for patient.py backend functions.
createUser	User input from username textbox, defined as 'userInput', user input from password textbox, defined as 'passInput', and user input from verification password textbox defined as 'passCheck'	This function checks for valid user creation. If the password and verification password values are not equal to each other, the user is notified with a pop-up window that their password is invalid. If the username is already taken by checking with function uniqueUser from patient.py, the user is notified with a pop-up window their attempted username is invalid. If password and username are validated, addPatient function from patient.py is called to append the username and password to patient database (patientData.csv). The user is then prompted by this method to login through the main page with their new account.

#### Class | main\_page

Method	Inputs	Responsibility
__init__	Self, Parent, Controller	The page consists of 5 buttons for moving to the parameters page, opening the about menu to see the dcm specifications, moving to the date and time page, ending telemetry and returning to the login page. It also confirms the DCM - Pacemaker communicaton.

#### Class | set\_date\_time\_page

Method	Inputs	Responsibility
__init__	Self, Parent, Controller	Showcases a calendar menu to

		set the date of the pacemaker. Includes a button to log the current time and print the date and time.
--	--	--

## Class | parameters\_page

Method	Inputs	Responsibility
<code>__init__</code>	Self, Parent, Controller	Introduces a drop down menu to choose pacing mode. Allows parameters values to be set using drop down menus. A save button to save parameter values and a back button to return to main_page.
parameter (private method)	E (event)	Based on the pacing mode, enables the required parameters and disables the rest.
get_val (private method)	No inputs	Gets the mode selected in the dropdown menu and prints the values set by the user (this code is run when the user presses the save button.

## Module Two | patient.py

The purpose of this module is to handle the backend functionality of the graphical user interface. Data from user input is passed as a value to the patient.py functions to perform logical manipulation to save data to .csv or return values that determine what state the program is in based on the provided user input. The state is returned to the main\_3k04.py file to determine what to display to the user.

## Class | Patient

Patient class is created to have patient objects. The patient object currently has attributes username and password. Patient objects are appended to the patientData.csv database file. The patient class also contains several functions that are concerned with the backend functionality of the graphical user interface. These functions are called from the main\_3k04.py file to elicit the correct graphical response based on user inputs. By having these backend functions in a separate file, it provides a lowly coupled and highly cohesive design. This design allows for accessible modification and additions of backend functions without modifying front end functionality.

Function	Inputs	Responsibility
createPD	No Inputs	Checks if there is an already existing patientData.csv file when a user attempts to create a new account, if not, then the new patientData.csv file is created.
addPatient	Instance of a patient object, defined as 'patientObj'	Appends a new patient's username and password to the patientData.csv file.
uniqueUser	Instance of a patient object, defined as 'patientObj'	When a user is attempting to create a new account, this function attempts to check if there is already an existing patient object in the patientData.csv file with the same username. This function returns true if the user has typed a unique username.
loginUser	User input from username textbox, defined as 'userInput' and user input from password textbox, defined as 'passInput'	When a user is attempting to login to the main application page through the welcome page, the text fields are checked by this function to verify if the username/password combination is correct. If the username/password combination is correct, this function returns a value of 0. If the username is not found in the database, this function returns a value of 1. If the username is found in the database but the password does not match the username of the account, this function returns a value of 1. If the username and password are both correct, this function returns a value of 2.
availUsers	No Inputs	This function checks the amount of users in the patientData.csv file. This function will return the current number of users in the database.

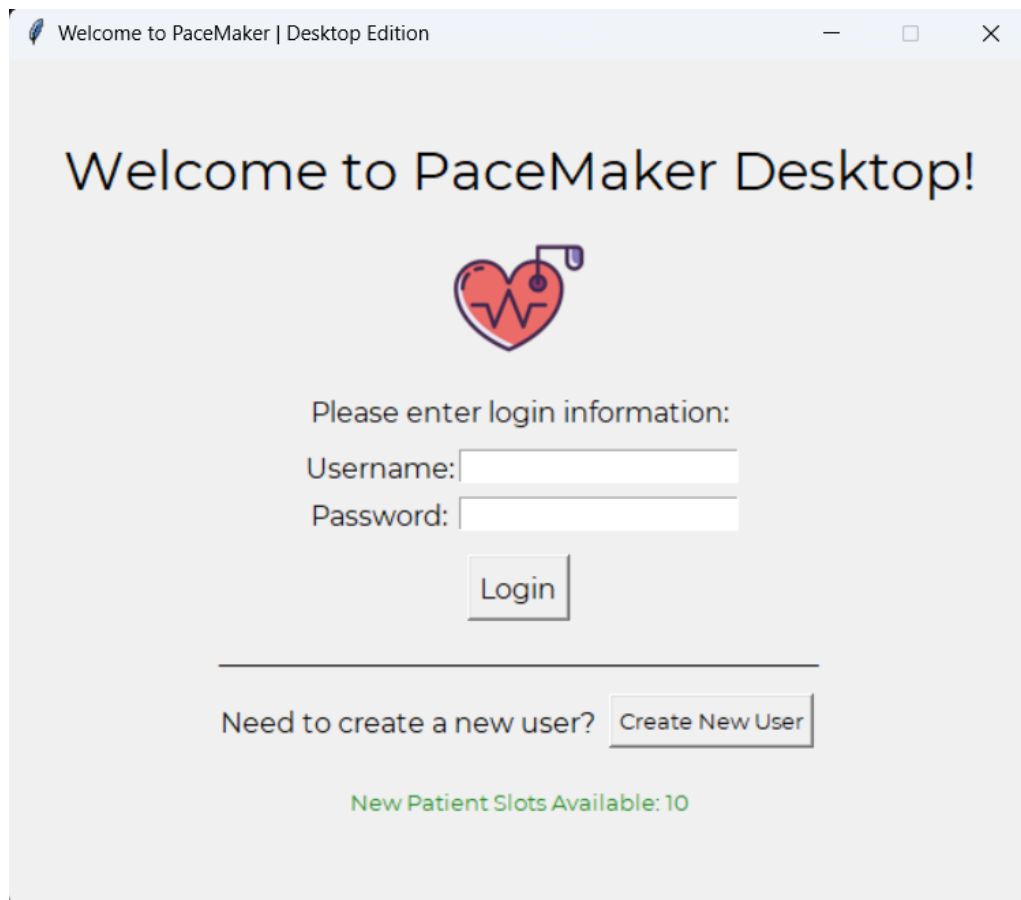
# Testing & Results

## Summary of Roadblocks

Throughout the development process, there were several areas in which the original design and implementation of requirements needed to be modified or corrected for successful requirement fulfillment. When implementing the frame transitions, the code was initially formatted to have frames in separate python files. Due to the nature of how TKinter works with python, it was more efficient to have all the GUI frames in the same python file rather than different python files to avoid circular importing which would result in high coupling.

Additionally, the 'New Patient Slots Available' button was originally a label as pictured below but was modified to a button to provide live updates of how many users slots were still available after the user created more accounts without exiting the application.

Figure 1. Previous Iteration of Patient Slots Available Label.



In order to avoid multiple typing errors w.r.t to enter the parameter values drop down menus were used so only one of the required values would be selected. The first value for the drop downs was also set to avoid a null error from occurring thus eliminating numerous possibilities of errors while entering parameter values.

Furthermore, page number was reduced from having a single page for each pacing mode (AOO, VOO, AAI, VVI) to a single page to improve the ease of use of the program. This also reduced lines of code by approximately 150 lines which provided easier code readability as well.

Figure 2. Previous Iteration of Main Menu Page - Upon clicking 'Open' button, the user is brought to AAI specific parameters page as shown in Figure 3.

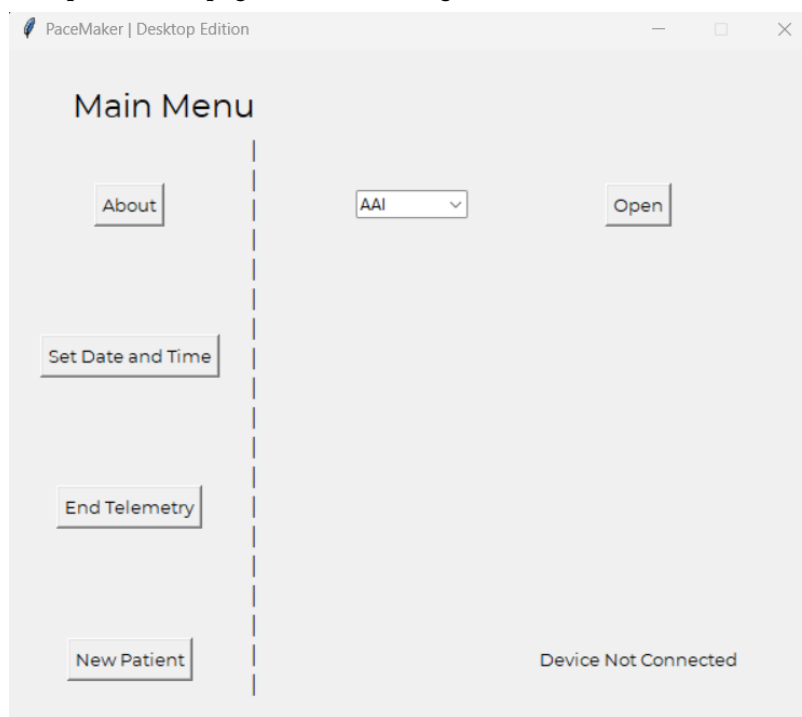
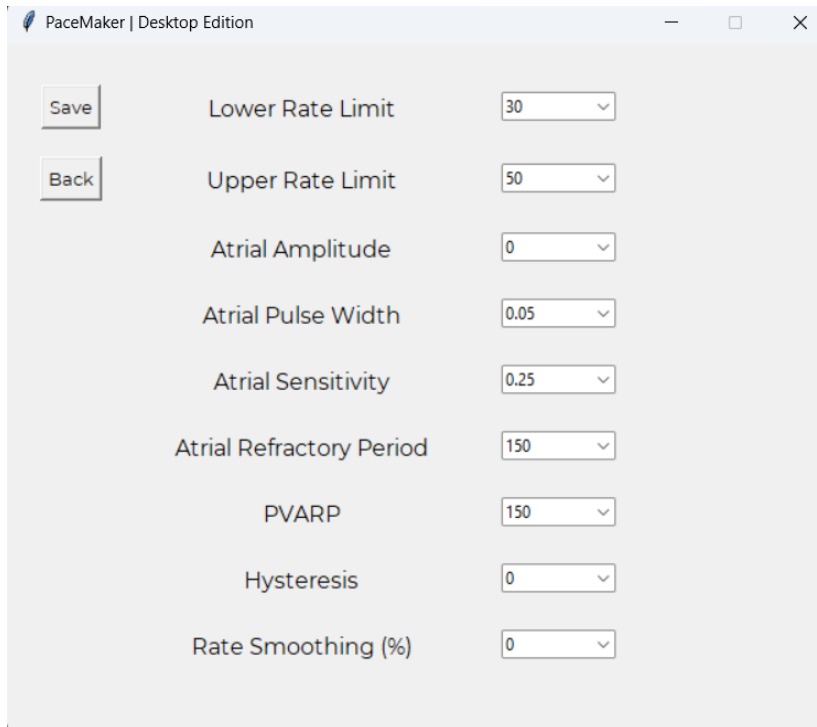


Figure 3. Previous Iteration of AAI specific Parameters page. Each method had their own respective Parameters page.





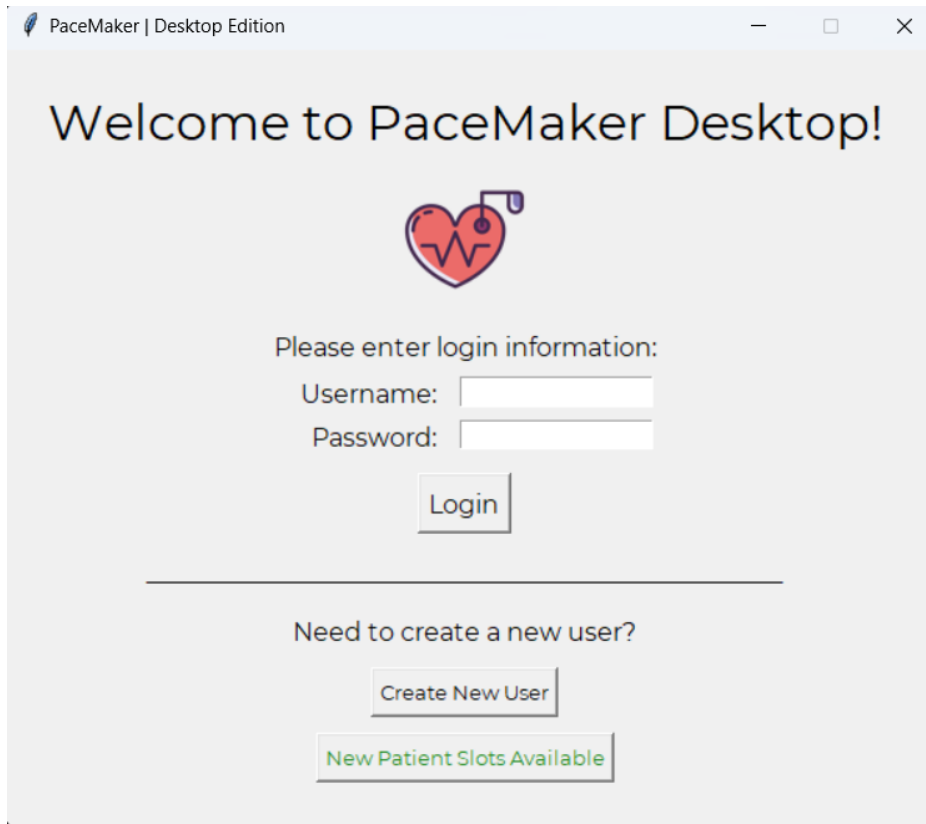
Setting	Value
Lower Rate Limit	30
Upper Rate Limit	50
Atrial Amplitude	0
Atrial Pulse Width	0.05
Atrial Sensitivity	0.25
Atrial Refractory Period	150
PVARP	150
Hysteresis	0
Rate Smoothing (%)	0

The most updated version of the application pages are shown in the sections below in which further requirement testing was performed.

## Welcome & Login | Testing

When the application is first launched, the user is initially greeted with the Welcome Page as shown in Figure 4.

Figure 4. Welcome Page



The following conditions were tested on the welcome page to fulfill requirements of a robust, easy to use program:

Condition	Sub-Conditions	Required Result
“Login” button is pressed.	Correct username & password combination.	User is taken to the Main Menu page (main_page). Figure 5.
	Incorrect username or username textbox is left blank by user. Username input by user does not exist in patientData.csv.	Prompt user with pop-up window that their username input is incorrect. User remains on the Welcome Page (welcome_page). Figure 6
	Correct username but incorrect password or password textbox is left blank by the user. Password input by user does not match the	Prompt user with pop-up window that their password input is incorrect. User remains on the Welcome Page (welcome_page). Figure 7

	username password combination in patientData.csv.	
“Create New User” button is pressed.	Patient limit of 10 accounts is already in patientData.csv.	Prompt user with a pop-up window that the maximum number of user accounts has been reached. User remains on the Welcome Page (welcome_page). Figure 8
	Number of patients in patientData.csv is less than 10.	User is taken to the Create New User page (user_page). Figure 9
“New Patient Slots Available” button is pressed.	N/A	User is shown the number of account slots available for new patient account creation. Figure 10

Figure 5. Successful Username & Password Login

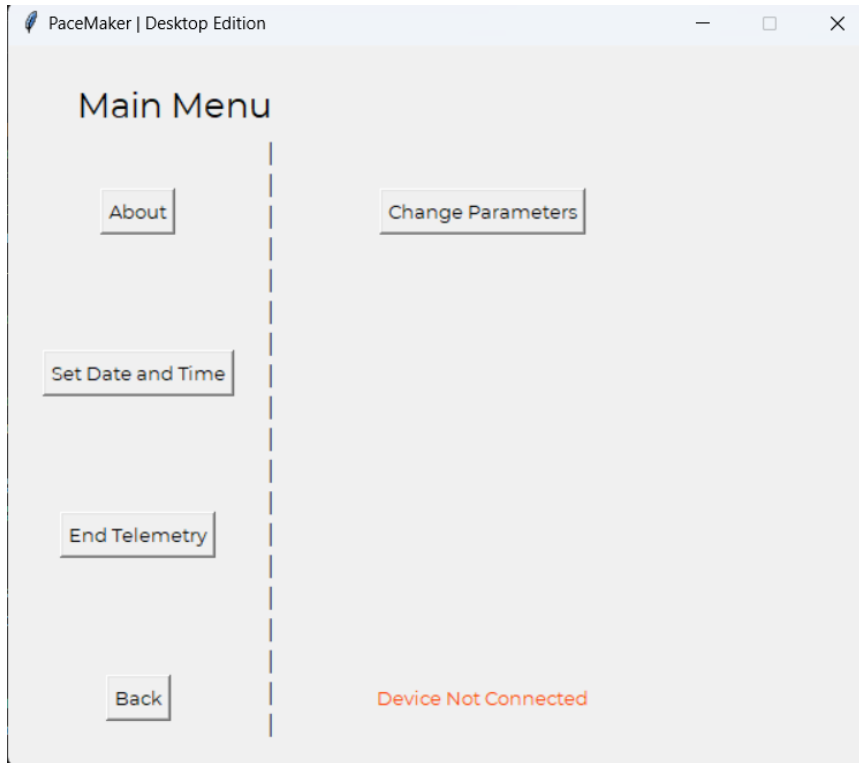


Figure 6. Incorrect Username Input

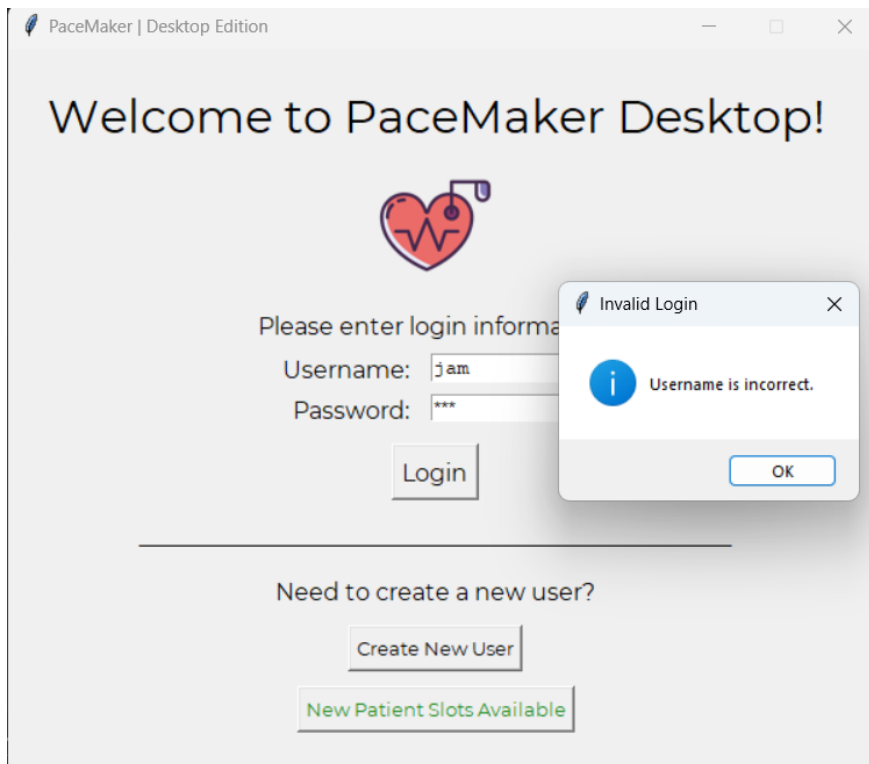


Figure 7. Incorrect Password Input

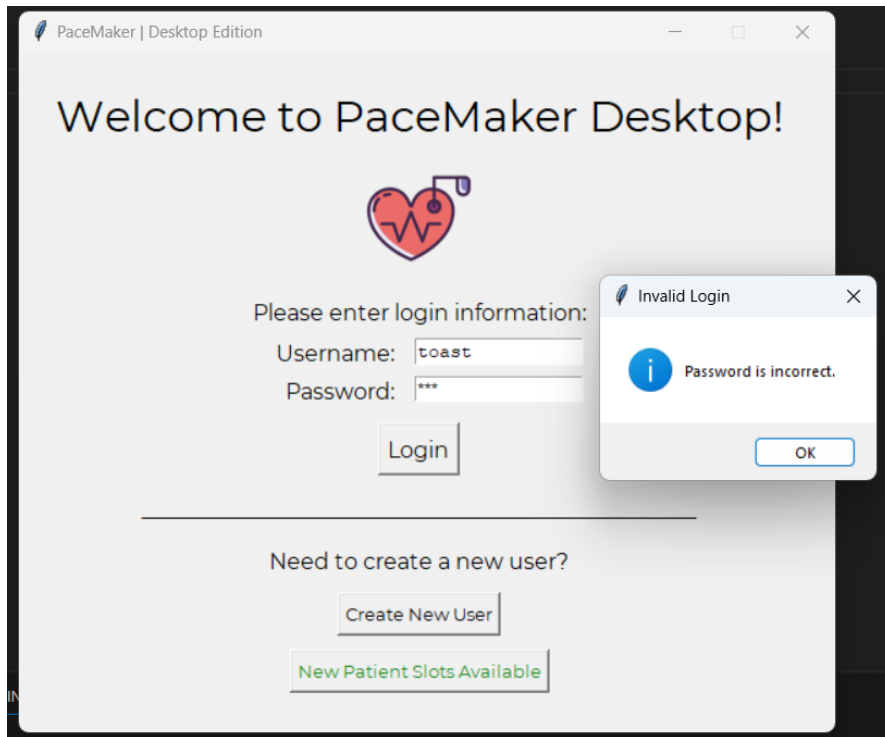


Figure 8. Patient Limit Reached

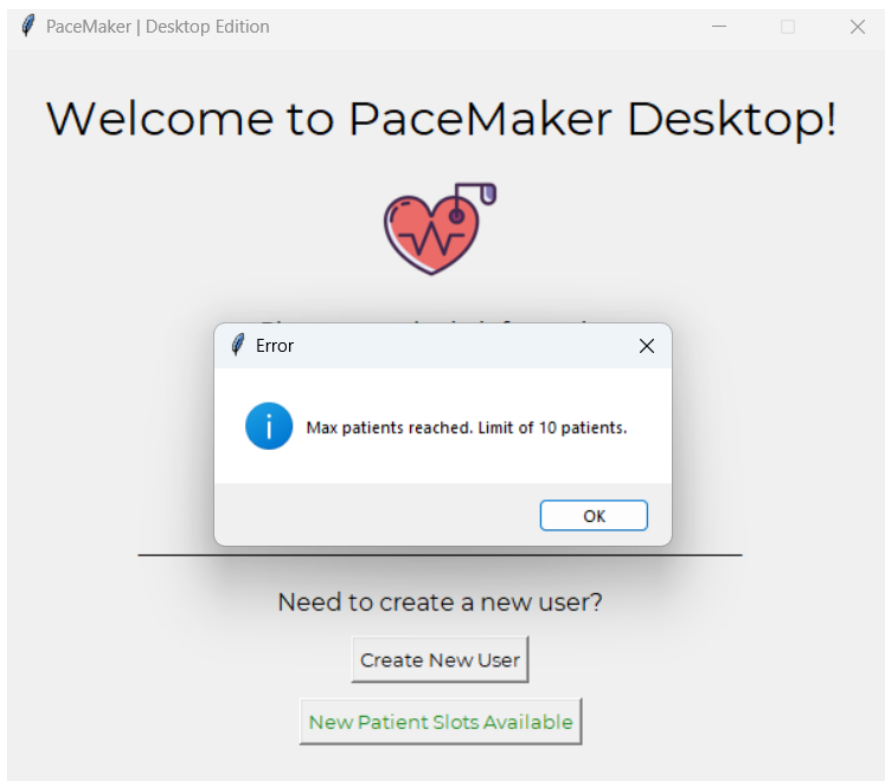
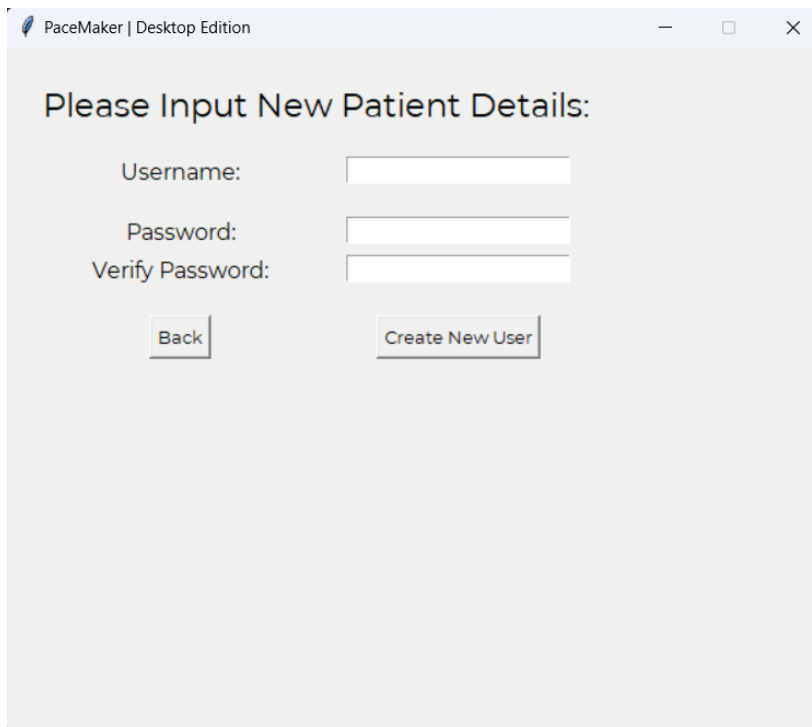
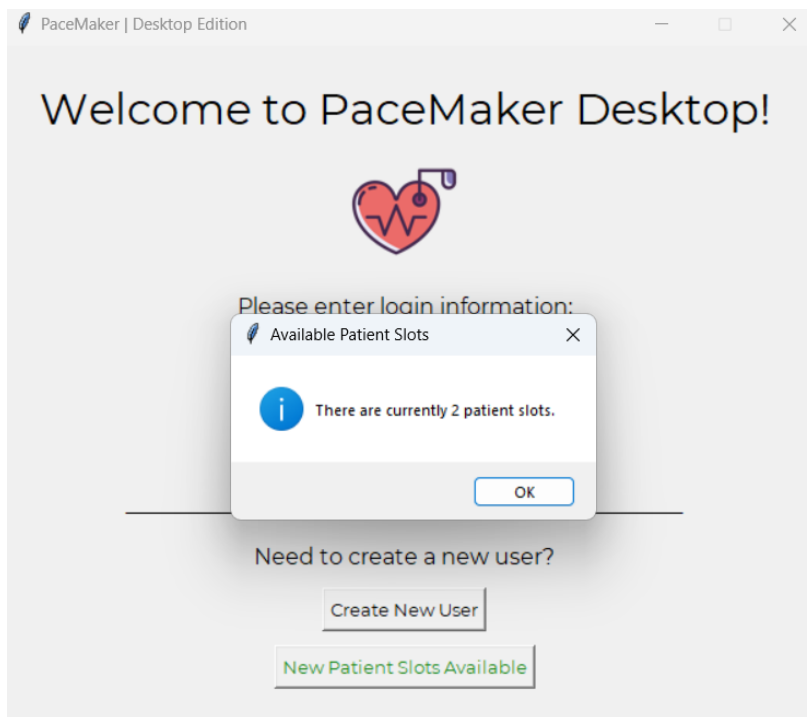


Figure 9. Create New User Page



The screenshot shows a window titled "PaceMaker | Desktop Edition". The main heading is "Please Input New Patient Details:". Below this, there are three input fields: "Username:", "Password:", and "Verify Password:". At the bottom, there are two buttons: "Back" and "Create New User".

Figure 10. Status Check of New Patient Slots Available



The screenshot shows the main login screen of "PaceMaker Desktop Edition". The heading is "Welcome to PaceMaker Desktop!". Below it is a heart icon with a pulse line. A dialog box titled "Available Patient Slots" is open, displaying an information icon and the text "There are currently 2 patient slots." with an "OK" button. Below the dialog, the text "Please enter login information:" is visible. At the bottom, there are two buttons: "Create New User" and "New Patient Slots Available".

## User Creation | Testing

When the user has not yet reached the maximum number of patients and chooses to create another patient account with the “Create New User” button, the following page is brought up as displayed in Figure 9 above.

The following conditions were tested on the user creation page to fulfill requirements of a robust, easy to use program:

Condition	Sub-Conditions	Required Result
“Create New User” button is pressed.	Username is unique and password is identical in both password creation textboxes (“Password” and “Verify Password”).	User is prompted with a pop-up that their user creation was successful. Username and password information is stored in patientData.csv. User is then prompted to login on the main page with their new account. Figure 11.
	Username is already taken and in the patientData.csv database.	User is prompted with a pop-up that the attempted username is already taken and they must create a different username for the new account. Figure 12.
	Password inputs in both text boxes do not match each other.	User is prompted with a pop-up that the attempted password verification is incorrect and to double-check that both passwords match. Figure 13
“Back” button is pressed.	N/A	User is brought to the Welcome Page (welcome_page). Figure #

Figure 11. Unique Username and Correct Password Verification | User Created

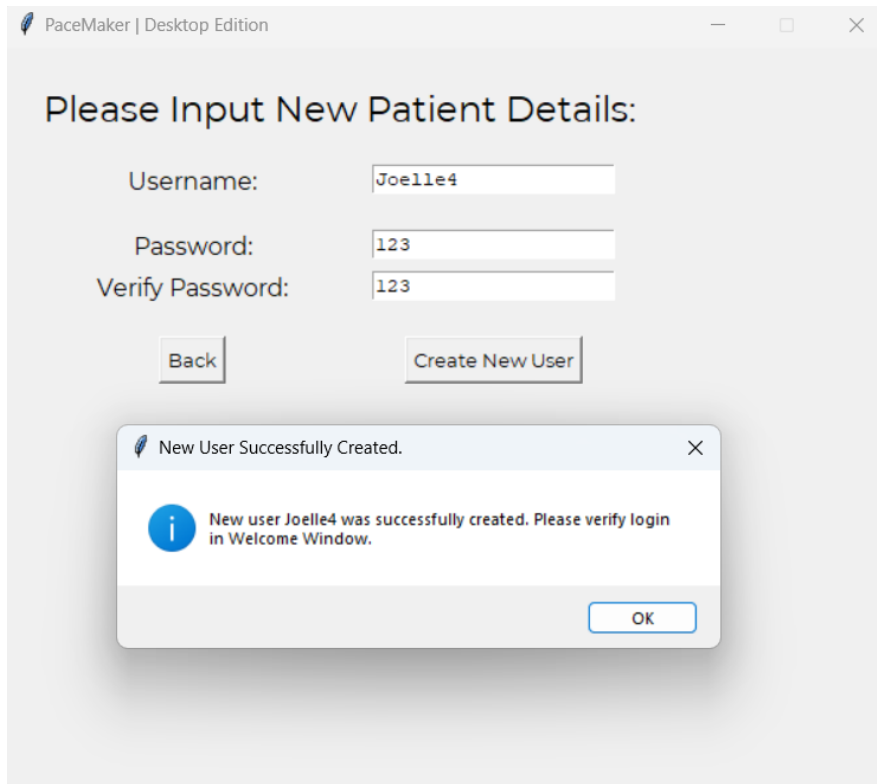


Figure 12. Username is Not Unique | User Not Created

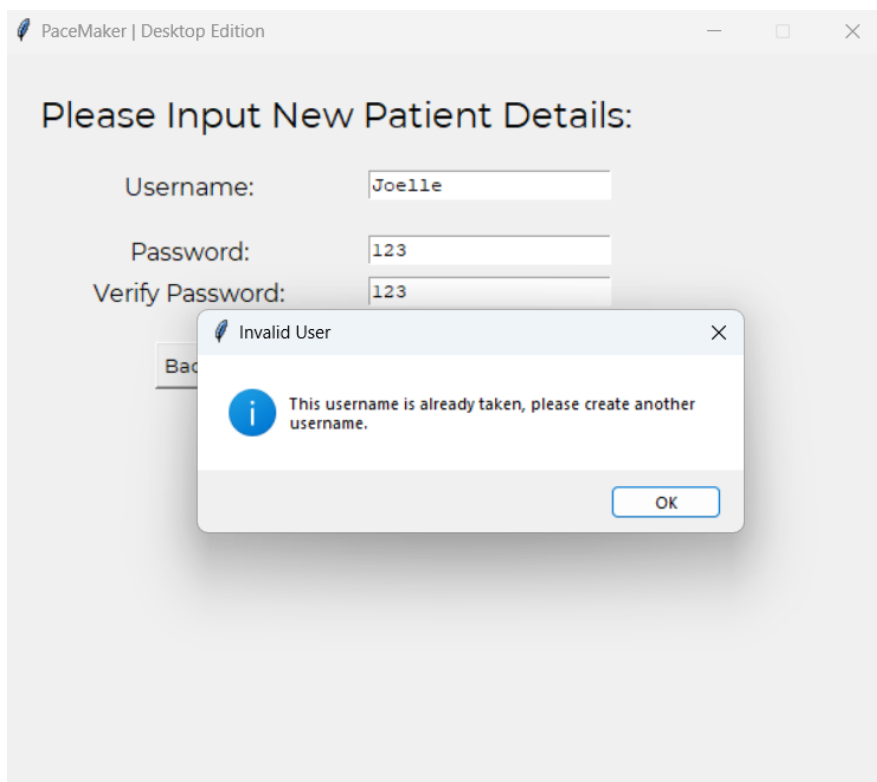
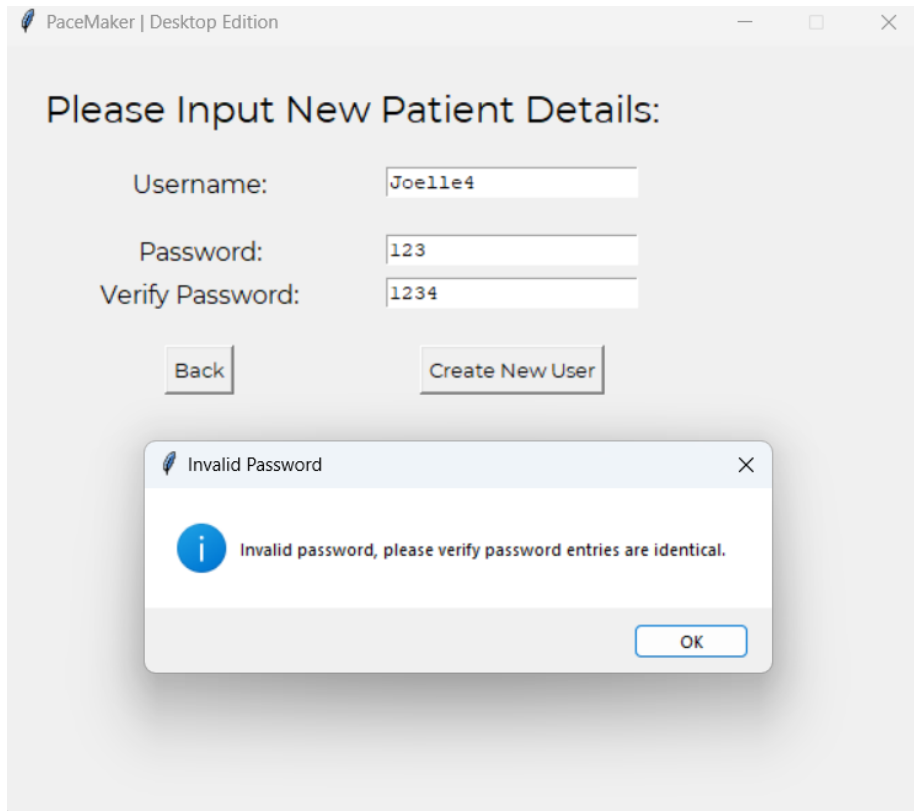


Figure 13. Password Verification Incorrect | User Not Created





## Main Menu | Testing

When the user has successfully logged in with a correct username and password combination, they are presented with the Main Menu page as displayed in Figure 5 above.

The following conditions were tested on the Main Menu page to fulfill requirements of a robust, easy to use program:

Condition	Sub-Conditions	Required Result
“About” button is pressed.	N/A	<p>Current Implementation: Placeholder label for future implementation. About currently shows placeholder labels for hardware model number and DCM serial number.</p> <p>Future Implementation: Model number and DCM serial number are</p>

		communicated through serial data to verify which device is connected. Figure 14.
“Set Date and Time” button is pressed.	N/A	<p>Current Implementation: Upon button press, the user is brought to the Calendar page. Figure 15. Here the user is able to view a calendar and log the current time. This time is then saved within a local variable.</p> <p>Future Implementation: Can be used to date telemetry reports. Figure 16.</p>
“End telemetry” button is pressed.	N/A	<p>Current Implementation: Notifies the user that the telemetry session has ended.</p> <p>Future Implementation: Communicate with the pacemaker hardware to end the telemetry session. Figure 17</p>
“Change Parameters” button is pressed.	N/A	Upon button press, the user is brought to the Parameters Page. Figure 18.
The DCM is attempting to serially communicate with the Pacemaker device.	Successful Communication	<p>Current Implementation: Placeholder label of “Device Not Connected”</p> <p>Future Implementation: Label will change to “Device Connected”. Figure 5.</p>
	Unsuccessful Communication	<p>Current Implementation: Placeholder label of “Device Not Connected”</p> <p>Future Implementation: Label will remain as “Device Not Connected”. Figure 5.</p>

Figure 14. About Button

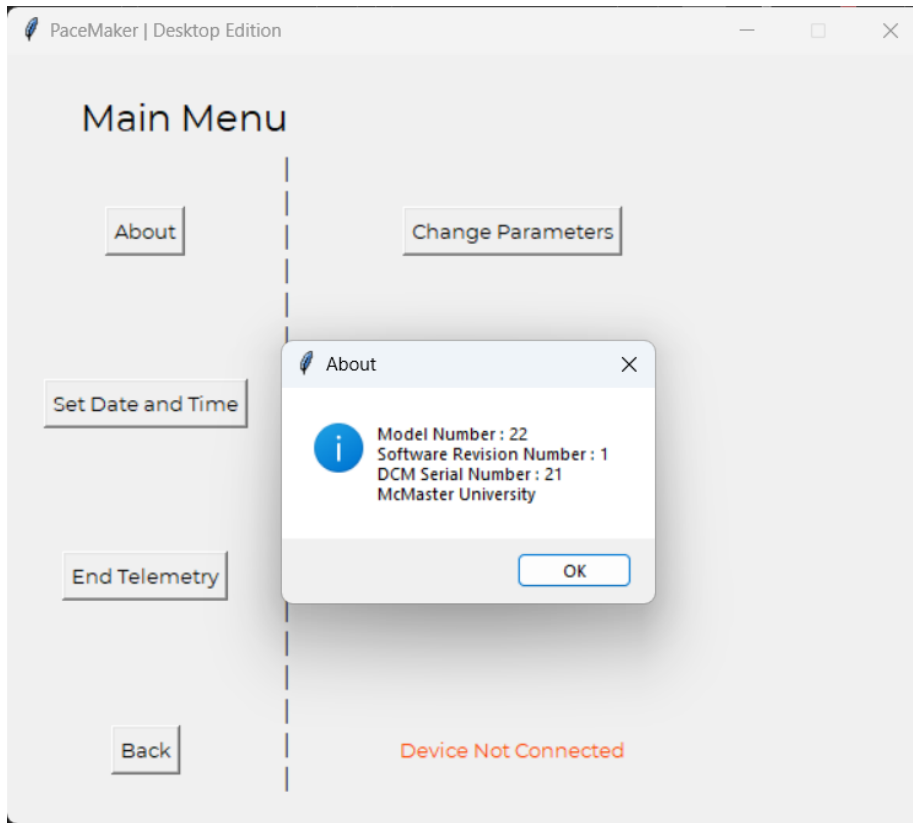
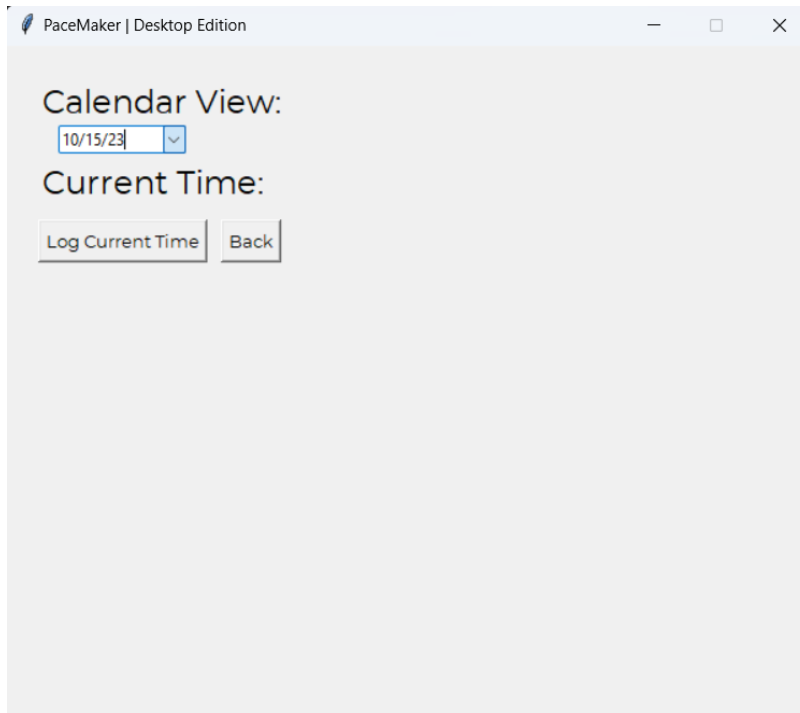


Figure 15. Calendar Page & Calendar Display



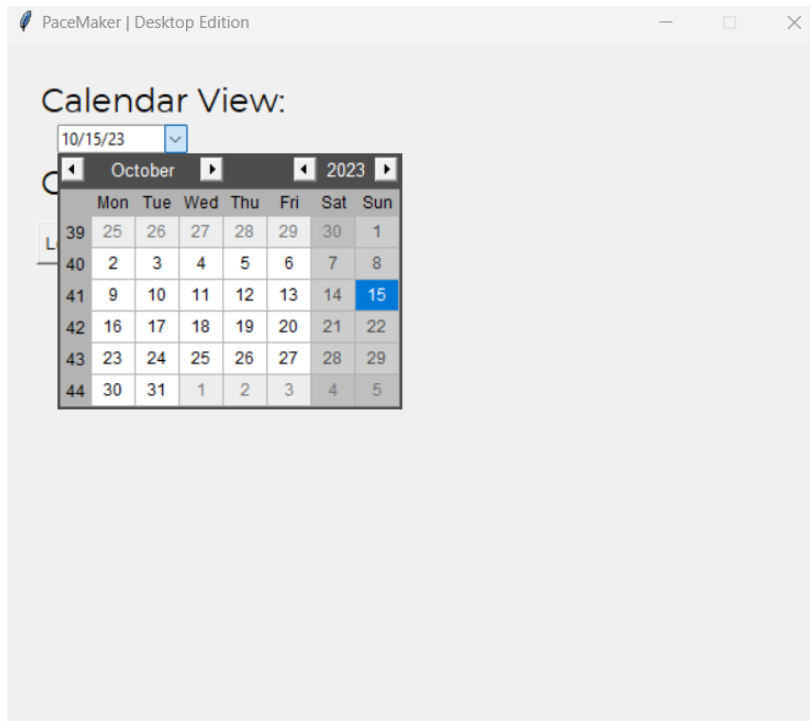


Figure 16. Log Current Time Button Function

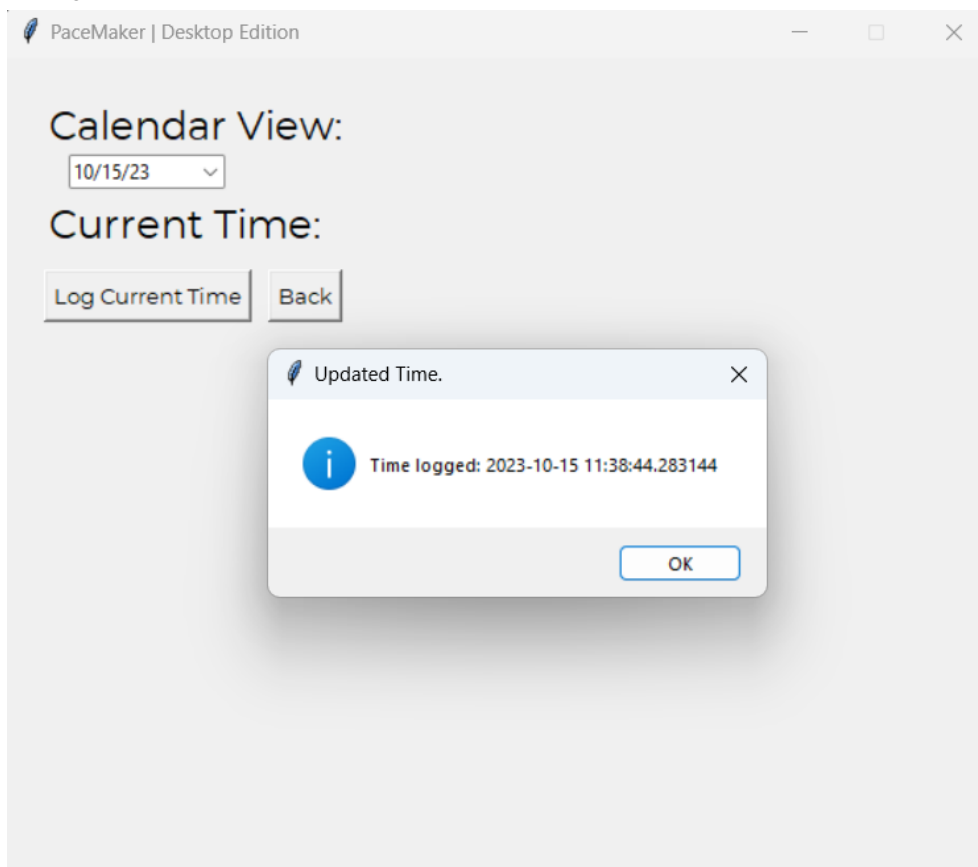


Figure 17. End Telemetry Button Pressed

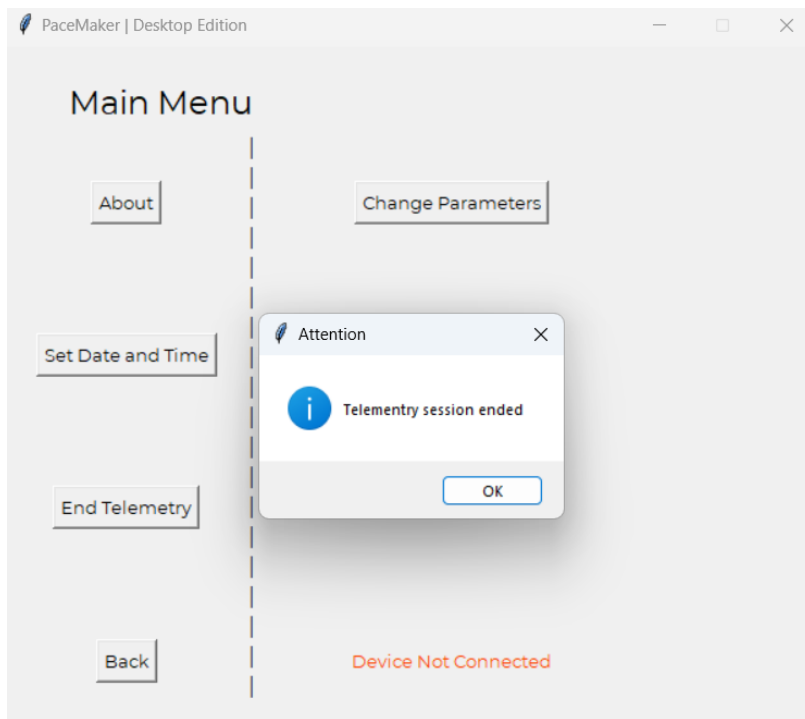
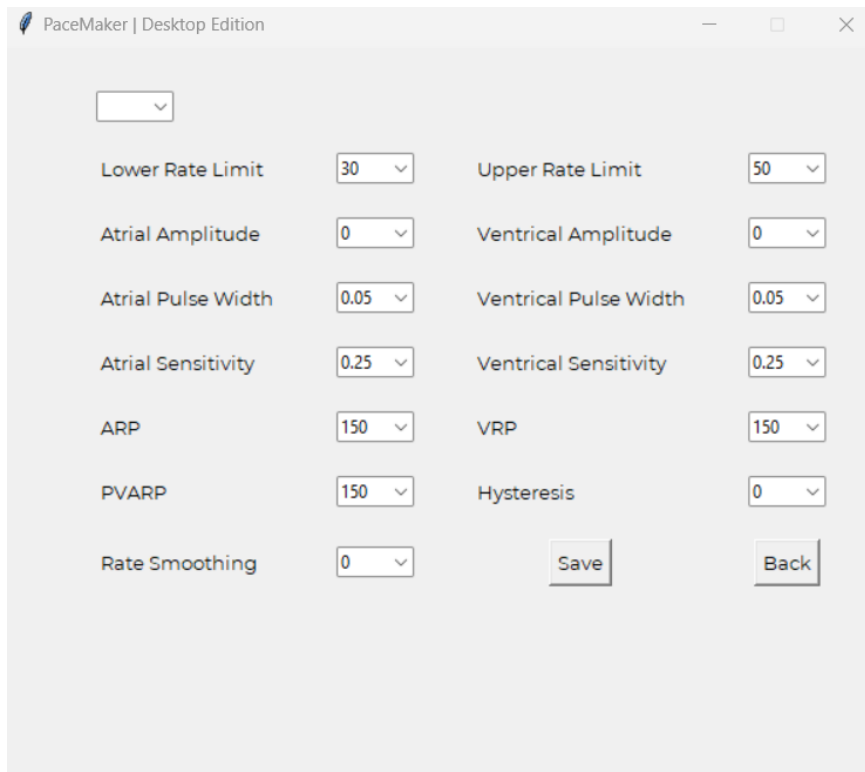


Figure 18. Programmable Parameters Page



## Parameters Page | Testing

Once the user presses the parameter button on the main menu they are presented with the parameters page.

The following conditions were tested for a robust and easy to use program:

Condition	Sub-Condition	Required Result
Pacing Modes drop down menu (top left corner )	N/A	Upon selecting a mode, the parameters used for the selected mode will be enabled and the rest will be disabled
“Save” Button is pressed	Pacing mode is selected	Current Implementation: The data set in the dropdown menus for each parameter will be saved and printed on the python console as shown in fig19  Future Implementation: The data.set by the user will be saved in a csv file in a dictionary
	Pacing mode is not selected	An message pops up asking the user to select a pacing mode as shown in fig 20
“Back” Button is pressed	N/A	The user will be taken back to the menu page
All the thirteen parameters i.e. 1. Lower Rate Limit 2. Upper Rate Limit 3. Atrial Amplitude 4. Ventricular Amplitude 5. Atrial Pulse Width 6. Ventricular Pulse Width 7. Atrial sensitivity 8. Ventricular Sensitivity 9. ARP	Enabled	Opens a drop down menu to showcase possible values to be set as shown in fig 21 (Shown for Lower Rate Limit,Upper Rate Limit,Atrial Amplitude and Atrial Pulse Width)
	Disabled	Does not perform any function and is faded out as shown in fig 22(for Atrial Amplitude, Atrial Pulse Width, Atrial Sensitivity,

10. VRP 11. PVARP 12. Hysteresis 13. Rate Smoothing  Perform the same function with different values When the drop down next to the parameter name is pressed		ARP and PVARP)
---	--	----------------

Figure 19. When the ‘Save’ button is pressed after choosing a pacing mode, the values set by the user are printed on the python console.

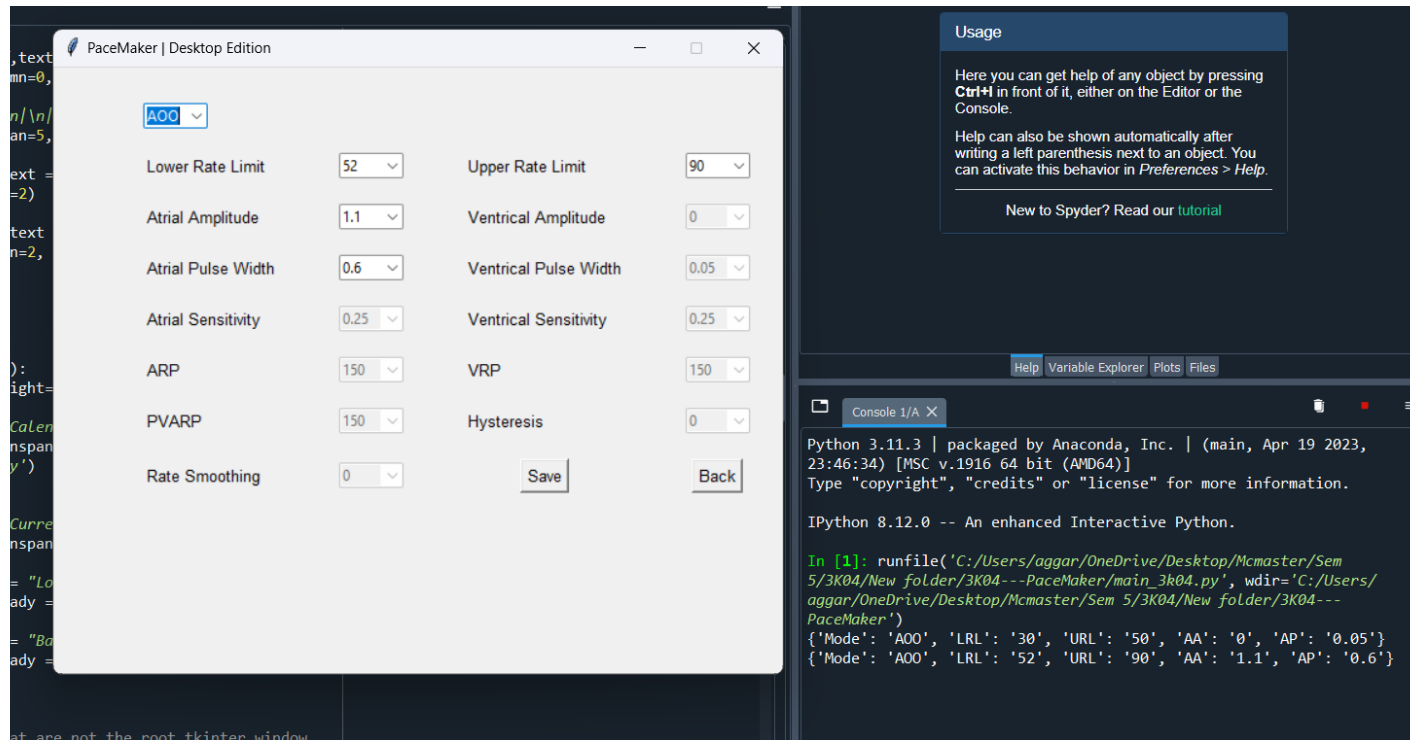


Figure 20. When ‘Save’ button is pressed without choosing a pacing mode

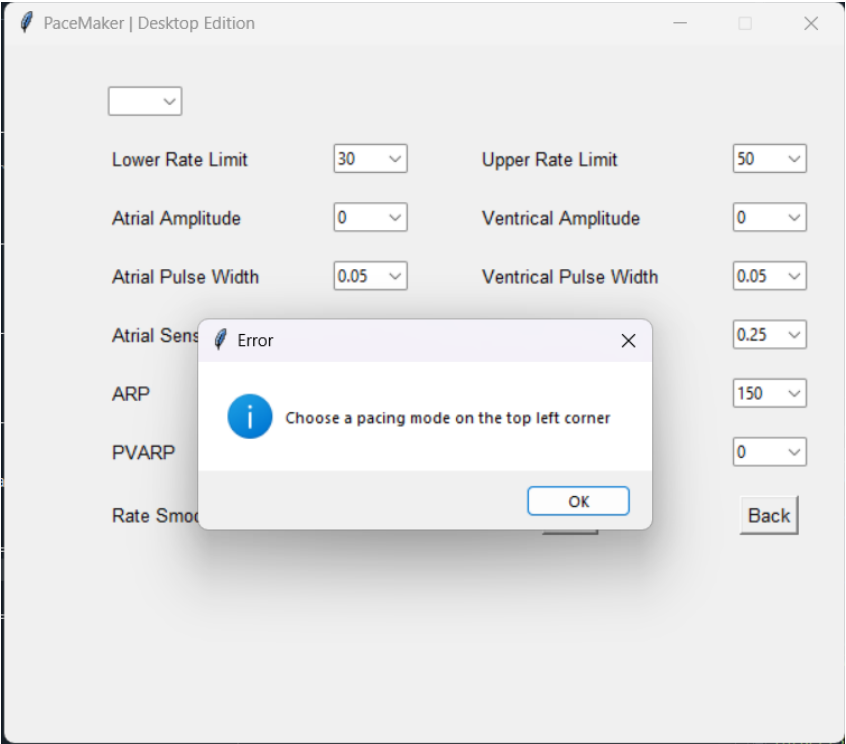




Figure 21. When the enabled Lower rate limit drop down box is selected it shows the possible values for the parameter.

The screenshot shows the 'PaceMaker | Desktop Edition' window. At the top left is a dropdown menu set to 'A00'. Below it, on the left side, are several parameters: 'Lower Rate Limit' (30), 'Atrial Amplitude' (40), 'Atrial Pulse Width' (50), 'Atrial Sensitivity' (52), 'ARP' (55), 'PVARP' (150), and 'Rate Smoothing' (0). The 'Lower Rate Limit' dropdown is open, showing a list of values: 30, 35, 40, 45, 50, 51, 52, 53, 54, and 55. On the right side, there are more parameters: 'Upper Rate Limit' (50), 'Ventricular Amplitude' (0), 'Ventricular Pulse Width' (0.05), 'Ventricular Sensitivity' (0.25), 'VRP' (150), and 'Hysteresis' (0). At the bottom right are 'Save' and 'Back' buttons.

Parameter	Value
Mode	A00
Lower Rate Limit	30
Atrial Amplitude	40
Atrial Pulse Width	50
Atrial Sensitivity	52
ARP	55
PVARP	150
Rate Smoothing	0
Upper Rate Limit	50
Ventricular Amplitude	0
Ventricular Pulse Width	0.05
Ventricular Sensitivity	0.25
VRP	150
Hysteresis	0

Figure 22. When a mode is selected which does not require a specific parameter in this case for (Atrial Amplitude, Atrial Pulse Width, Atrial Sensitivity, ARP and PVARP) the dropdowns for the same are faded out and disabled.

Parameter	Value
Mode	VVI
Lower Rate Limit	30
Upper Rate Limit	50
Atrial Amplitude	0
Ventricular Amplitude	0
Atrial Pulse Width	0.05
Ventricular Pulse Width	0.05
Atrial Sensitivity	0.25
Ventricular Sensitivity	0.25
ARP	150
VRP	150
PVARP	150
Hysteresis	0
Rate Smoothing	0

Save Back

# 3K04 | Assignment 2 Additions

Group 24

## Assignment Two | Requirements

For assignment two, there were several additional requirements that needed to be included in the project. Note that all the requirements mentioned are functional requirements with the potential for working connection with the hardware.

## Graphical User Interface Updates

### Modes & Parameters

In addition to the original 4 modes 'AOO', 'AAI', 'VOO', 'VVI', 4 new modes must be added namely, 'AOOR', 'AAIR', 'VOOR', 'VVIR'.

Updated List of parameters (updates are highlighted)

S.No	Parameter	Programmable Values	Increment
1	Lower Rate Limit	30 - 50 ppm 50 - 90 ppm 90 - 175 ppm	5 ppm 1 ppm 1ppm
2	Upper Rate Limit	50 - 175 ppm	5 ppm
3	Atrial or Ventricular pulse Amplitude	Off, 0.1 - 5.0 V	0.1 V
4	Atrial or Ventricular Pulse Width	1-30 ms	1 ms
5	Atrial or Ventricular Sensitivity	0 - 5V	0.1V
6	Atrial Refractory Period	150 - 500 ms	10 ms
7	Ventricular Refractory Period	150 - 500 ms	10 ms
8	PVARP	150 - 500 ms	10 ms
9	Hysteresis	Off, 30 - 50 ppm	- 5 ppm

		50 - 90 ppm 90 - 175 ppm	1 ppm 1ppm
10	Rate Smoothing	Off, 3, 6, 9, 12, 15, 18, 21, 25	-
11	Maximum Sensor Rate	50 - 175 ppm	5 ppm
12	Activity Threshold	V-LOW, LOW, MED-LOW, MED, MED-HIGH, HIGH, V-HIGH	-
13	Reaction Time	10 - 50 sec	10 sec
14	Response Factor	1-16	1
15	Recovery Time	2-16 min	1 min

### Egram Data

The user should be able to see electrogram data for either the patient's ventricle, atrium or both using the reception received when serially communicating with the pacemaker.

### Serial Communication

The user should be able to see when the device is connected/disconnected.

## Patient Data Storage Updates

### Parameter Data

The system is able to save parameter data to a specified patient and display the last parameter/mode values set and saved by a user.

## Serial Communication Transmission and Reception

### Device Connectivity

The system detects if the DCM has been connected to the pacemaker.

## Transmission

The system is able to transmit the last saved parameter values of the user to the device. The software submission for this assignment is designed for functioning between the pacemaker hardware and DCM software which should allow the DCM to set variables and have them received by the pacemaker.

## Verification

The system is able to receive and cross check the parameters transmitted to the pacemaker. The software submission for this assignment is designed for functioning between the pacemaker hardware and DCM software which should allow the pacemaker to send variables back to DCM for input verification.

## Electrogram Data

The system should be able to receive data from the device to plot on DCM as electrogram data in the form of ventricle and atrium signals for additional verification of the heart.

# Assignment Two | Potential Requirement Modifications

If there were to be additional modifications past assignment two, there are several areas in which the requirements of the product could change. For future iterations of this project, it may require that for an 'assignment three' the data must not be stored locally and instead stored through an encrypted online database which allows different clinics to access user data if the user chooses to get their pacemaker updated at a different location. Additionally, by using an encrypted data server to hold patient data, it is likely that more patient data can be stored with a dedicated memory server which means greater than 10 patients can be served at a time and more data can be saved per patient.

Furthermore, another application could be added to the requirements such that the DCM is able to send electrogram data to a separate application which the patient can access to view their own results of the data, similar to Ontario's LifeLabs company which is able to electronically share lab data with the respective tested patient through their online platform.

Lastly, one more potential requirement modification that could change is how the DCM is currently run when set up. It is required to have four python files (main\_3k04.py, patient.py, serial\_comm.py, and global\_vars.py). To take this DCM application one step further into the final product, the application should be formatted into an executable application in which the user can download and

run by clicking on a desktop application symbol rather than running the DCM through terminal or code editor.

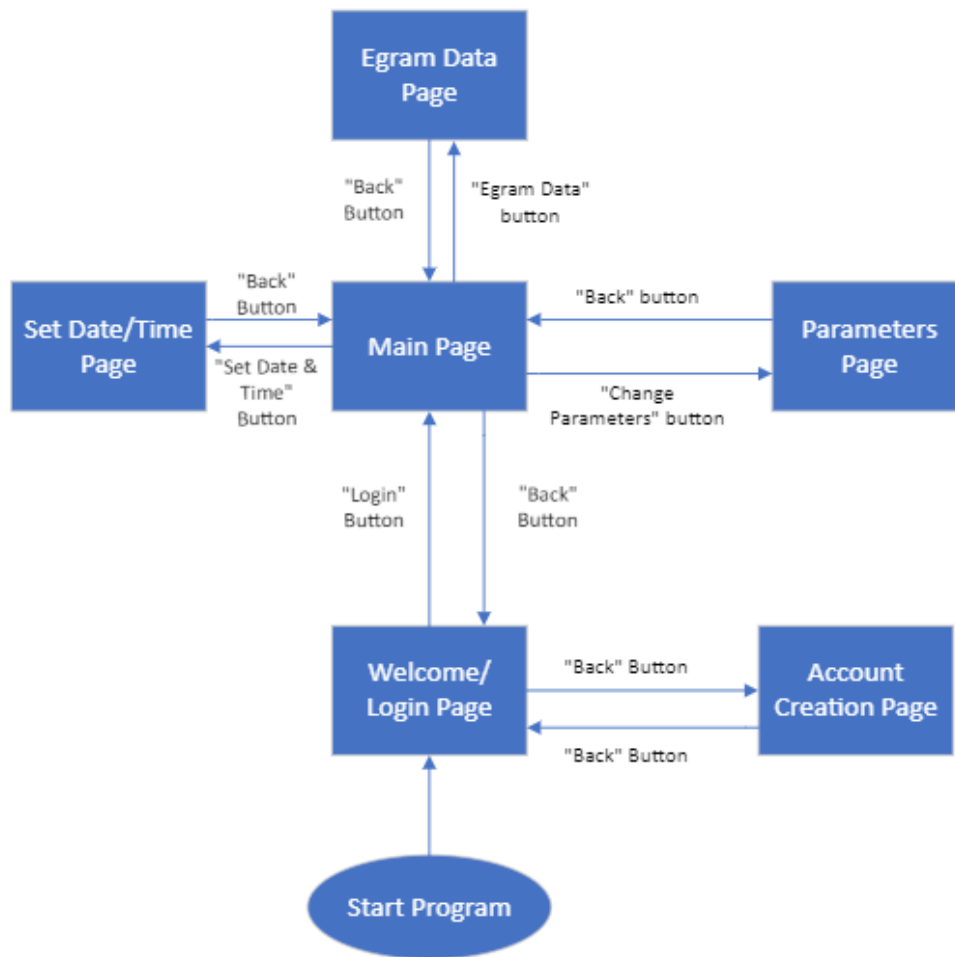
## Assignment Two | Design Decisions

There were several design decisions that needed to be made throughout the course of assignment two which resulted in some modifications to the work done to complete assignment one.

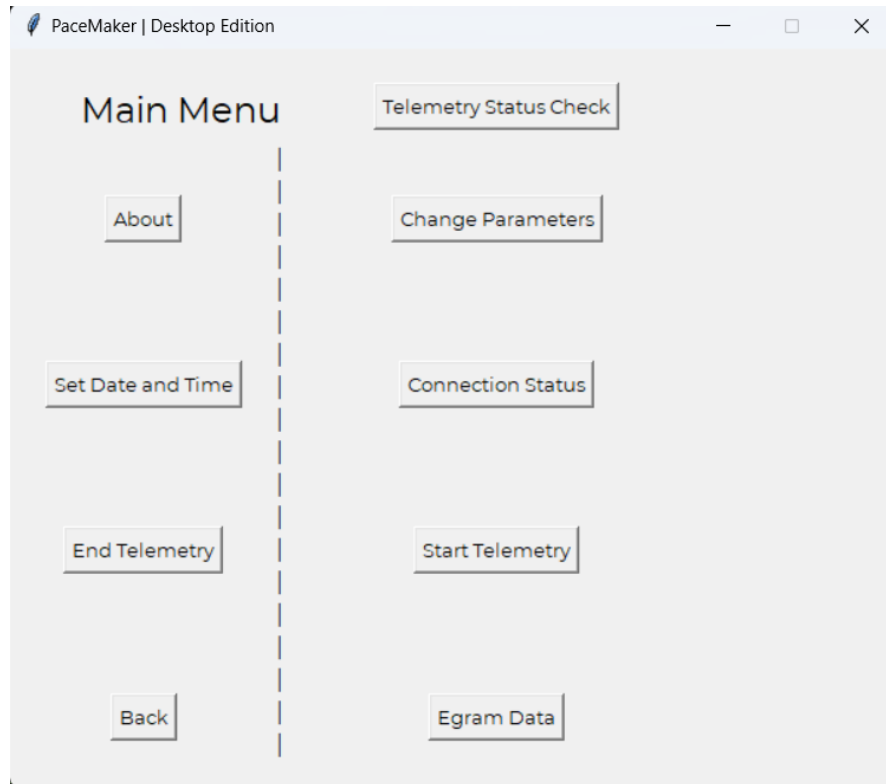
### Graphical User Interface

#### User Interface Pages, Formatting, & General Updates

Device connection status was changed from being text on the screen to a functional button which the user can press to check if the pacemaker device is connected to the DCM or if the pacemaker is not connected to the DCM. The main layout of the flow between pages is displayed below with the addition of the 'Egram Data Page'.



Functionality was added to the status of the connected device in which the main page had a few design updates to accommodate for the new functionality as pictured below:



The 'Connection Status' button has replaced the orange 'Device is not connected' font as displayed in assignment one. The user may press this button at any time to check a pop-up window that states if the pacemaker device is connected to the DCM or not.

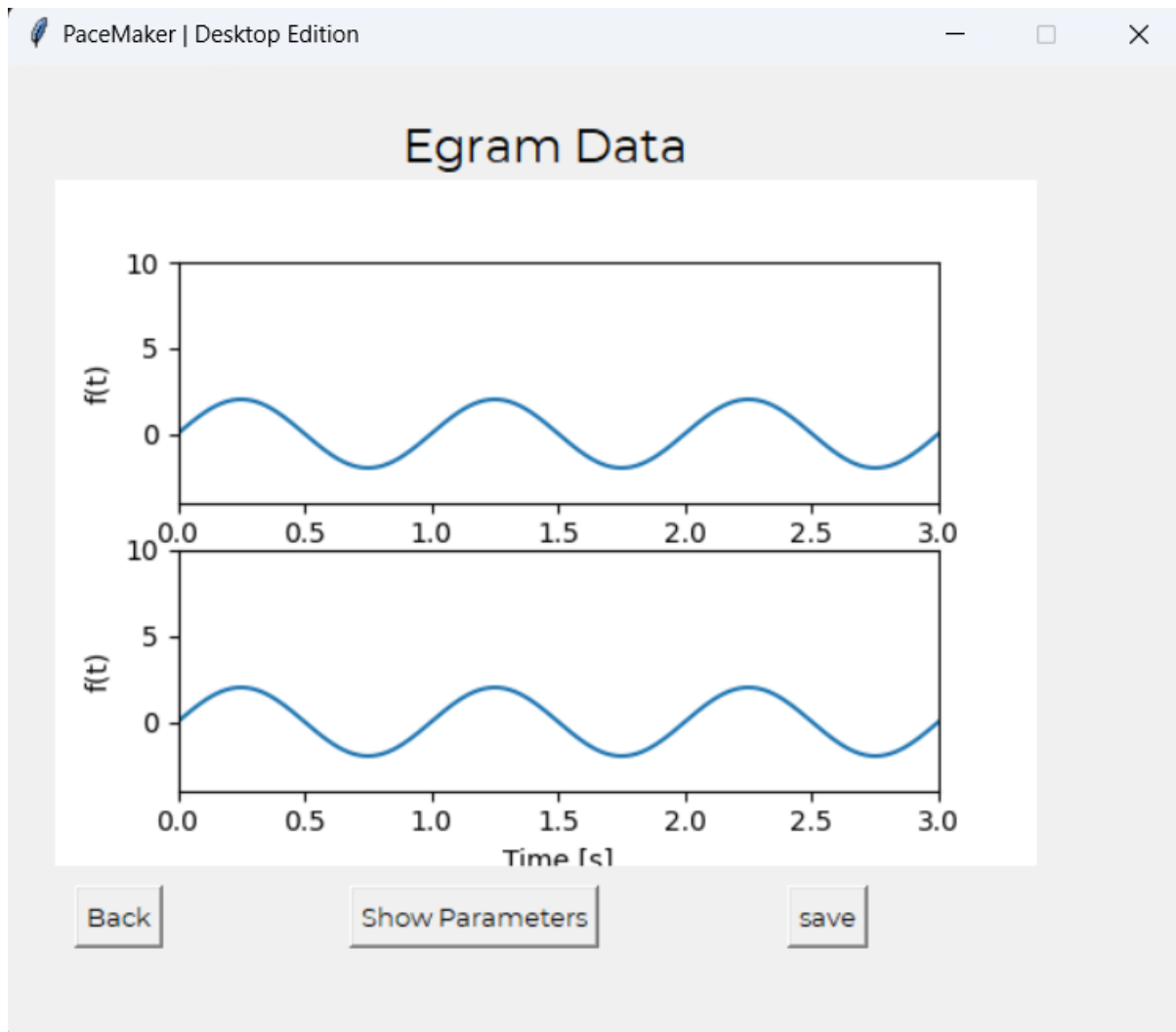


Additional changes in functionality and design were added to the Parameters page which allows the user to save their set parameters to the patientData.csv with the 'Save' button and load the parameters which they have most recently saved with the 'Load Values' button:

Parameter	Value	Parameter	Value
Lower Rate Limit	30	Upper Rate Limit	50
Atrial Amplitude	0	Ventricular Amplitude	0
Atrial Pulse Width	1	Ventricular Pulse Width	1
Atrial Sensitivity	0	Ventricular Sensitivity	0
ARP	150	VRP	150
PVARP	150	Maximum Sensor Rate	50
Hysteresis	0	Rate Smoothing	0
Activity Threshold	1	Reaction Time	10
Response Factor	1	Recovery Time	2

## Electrogram Page

Furthermore, the electrogram data page was created to display both the atrium and ventricle signals passing through the heart and collected via serial data through the pacemaker. Unfortunately there were issues in the implementation of functionality of the egram via serial communication with the pacemaker and the team was unable to successfully implement due to the unexpected results in the communication between pacemaker and DCM. Although the code exists to plot the atrium and ventricle signals if the serial communication between devices worked as expected.



## Data Storage

In the DCM, as mentioned before the programmable parameters are selected and stored in the change parameters page. The parameters are chosen via drop down menus and stored in a CSV file. This stored data can then be accessed by the user at any time by simply pressing the 'load values' button. The same string is then used for serial communication after being formatted to having the correct data types.

In python most of the data types are represented as integers, or floats, thus int is used for hex values and float is used for single

We have used unit16 data type for most of the data as some of the data values ranged from 0-500, in order to make it easier for the simulink and python teams to combine their respective codes and to cross check values with python/simulink with minimal confusion, a constant data type was used. Doing so made it easier for the python team to make the required serial communication string

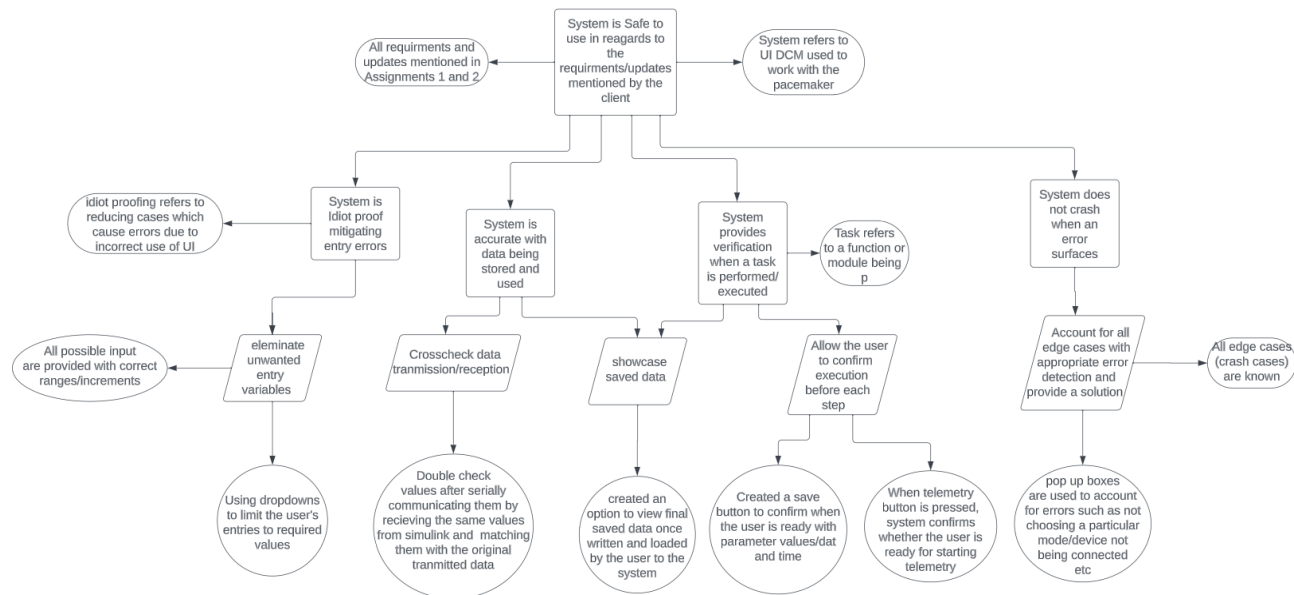
and made it easier for the simulink team to slice the received data. Single/float data type is only used for parameters with decimal values.

S.No	Parameter	Python Data Types	Simulink
1	Lower Rate Limit	int	unit16
2	Upper Rate Limit	int	unit16
3	Atrial or Ventricular pulse Amplitude	float	single
4	Atrial or Ventricular Pulse Width	float	single
5	Atrial or Ventricular Sensitivity	float	single
6	Atrial Refractory Period	int	unit16
7	Ventricular Refractory Period	int	unit16
8	PVARP	int	unit16
9	Hysteresis	int	single
10	Rate Smoothing	int	single
11	Maximum Sensor Rate	int	unit16
12	Activity Threshold	int	unit16
13	Reaction Time	int	unit16
14	Response Factor	int	unit16
15	Recovery Time	int	unit16
16	Mode	int	unit16

## Assignment Two | Assurance Cases

The assurance cases that were used to claim that the design was safe to use were to have the system be Idiot proof(mitigating entry errors), to make sure that the system be accurate with data being stored and used ,to have the system provide verification when a task is performed/ executed and to make sure the system does not crash when an error surfaces

The following arguments and their solutions/evidences are discussed in the flowchart below



## Assignment Two | Potential Design Modifications

For future design modifications of the project, further development can be done to create a design that is smart phone/tablet friendly such that the DCM application can be loaded and run using an iOS or Android device. This can allow more versatility for the user as different devices can be used to run this application depending on what the hospital or clinic can provide as a resource to the doctor.

Another design modification that could be added in the future is different colours being used for the atrium and ventricle signals in the electrogram data. This can provide the clinician to quickly take a look at the atrium and ventricle signals without much thought being put to which signal graph is which.

Furthermore, a final design modification that could be included in a future design of the project would be having a constant label at the top of the page to show which account the user is currently logged in as. If the doctor/clinician were to be seeing multiple patients in a day, it is imperative that the doctor is

logged into the right account as the patient data and parameters being saved must be saved to the correct file.

Lastly, one important feature that should be included in future designs would be a time-out period if the user is no longer active on the application for approximately 5 minutes. This will give the doctor a prompt to log in again to the same patient or provide a reminder to change the current log in to a different patient as they are setting up for the next patient. This avoids the risk of having the doctor log in, start telemetry, and save parameters with the incorrect patient file.

## Assignment Two | Modules

### Additions to Module One | main\_3k04.py

#### Class | mainWindow

Method	Inputs	Responsibilities
on_send_tele(self,user_params)	Self, user_params (user_params is the parameter data from the patientData.csv file where data for the user is stored)	If the patient does not have any parameter data able to send to the pacemaker yet the 'Start Telemetry' button was pressed, an error message will show up to the user stating that there is no parameter data to send for the current patient.

#### Class | Egram

Method	Inputs	Responsibilities
__init__	Self, parent, controller	Display a graph representing electrogram data taken from heartview using simulink, allows you to view current parameters and return to the

		welcome page.
--	--	---------------

## Additions to Module One | patients.py

Method	Inputs	Responsibilities
saveParams	Param_vals (param_vals is data from the parameter inputs given in the DCM GUI Parameters page), user (user is the username of the current logged in user)	The purpose of this method is to save the parameters which are set in the Parameters page of the DCM and save them to the respective patient's database such that when the user logs out, and logs back in after closing the application, the same parameters can be loaded from the saved patientData.csv file.
set_val	user (user is the username of the current logged in user)	The purpose of this method is to load the saved parameters from the patientData.csv file for the currently logged in user based on what was saved most recently in their file from their current session or previous sessions.

In addition to modules 'main\_3k04.py' and 'patient.py' two other modules were added for requirement implementation.

## Module Three | serial\_comm.py

The purpose of this module is to provide the functionality of the serial communication methods between the pacemaker and DCM application.

Method	Inputs	Responsibility
--------	--------	----------------

write_param	curr_user	Use serial communication to take the parameters saved into the .csv file for the respective current patient (curr_user), convert the parameters to bytes, and send them to the pacemaker hardware.
read_param	N/A	Reads the parameter outputs from the pacemaker to assure that correct parameters were sent.
status	N/A	Checks to see if the pacemaker device is connected or not connected. Will trigger a message box to display the current status.

## Module Four | global\_vars.py

The purpose of this module is to store global variables for the project when required. Only one global variable is currently being used throughout the DCM.

Method	Inputs	Responsibility
init()	N/A	Initializes global variable 'curr_user' which allows the program to keep track of which user is currently logged in. This variable is set and used in both patient.py and main_3k04.py modules

## Assignment Two | Testing & Results

Throughout the following documentation, conditions, sub-conditions, and required results are listed with corresponding figures demonstrating the tests carried out on the program to determine viability of the software.

### Parameters Pages | Testing

Condition	Sub-Conditions	Required Result
Pressing 'Load Values' Button on Parameters Page.	User has parameter data saved in their respective dataPatient.csv file.	Pop-up window is created which displays the currently saved parameter data for the patient. Figure 1.

	User does not have parameter data saved in their respective dataPatient.csv username file	Pop-up window is created and prompts the user that no parameter data has been saved and is not available to load. Figure 2.
Pressing 'Save' Button on Parameters Page.	N/A	Save file patientData.csv is updated to the respective patient with correct values that have been input on the Parameters Page. Figure 3.

### Main Menu Page | Testing

Condition	Sub-Conditions	Required Result
Pressing Telemetry Status Check Button	Telemetry is currently happening and there is data being sent to and from the pacemaker.	Feedback for testing is provided in terminal to check if byte values are returning back in the same format to the DCM and values that were sent to the pacemaker are evaluated as expected. Note 1: Main Menu
	There is no connection and data is not being passed between the pacemaker and DCM	No status of byte data is given as telemetry is not happening. No changes on the screen.
Pressing Connection Status Button	Pacemaker connected to DCM computer	Display pop-up window that states that the pacemaker device is connected. Figure 4.
	Pacemaker not connected to DCM computer	Display pop-up window that states that the pacemaker device is not connected.
Pressing Egram Data Button	Pacemaker is connected to the computer and 'Start Telemetry' Button is pressed	Display the E-gram page with no live data feed, only temporary data to display what the figure could look like.
	Pacemaker is not connected to the computer.	Display the E-gram page with no live data feed, only temporary data to display what the figure could look like.



	Pacemaker is connected to the computer but 'Start Telemetry' button has not been pressed	Display the E-gram page with live data feed, from the pacemaker. Note that for this assignment we were not yet able to get to this stage as the serial communication code was created yet there were issues with the communication when attempting to combine with simulink team. Although proof of concept is available as Tutorial 3 serial communication worked.
Start Telemetry Button is Pressed	N/A	Most current data saved for the respective user is converted to byte values and sent to the pacemaker via serial communication. Note that for this assignment there were issues with the serial communication of the data but the code which converts the data to bytes and transmits to the pacemaker is created. Not enough time was available to debug the communication issue.

Note 1 | Main Menu: This output of this data was often b", which was not the expected data. There was an issue with the communication output of the pacemaker to the DCM, this was not resolved in the debugging process and displayed to the TAs during the demo.

Figure 1

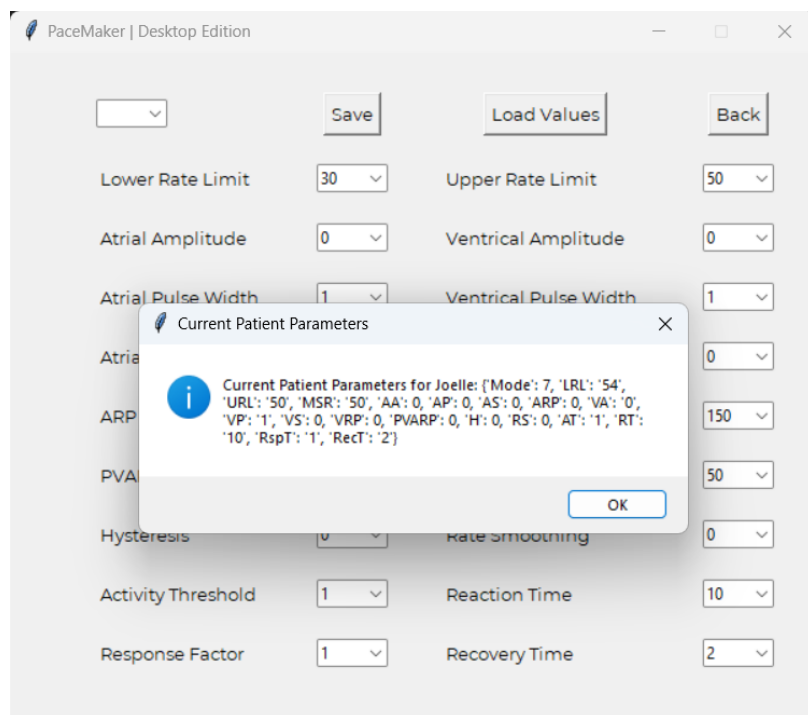


Figure 2

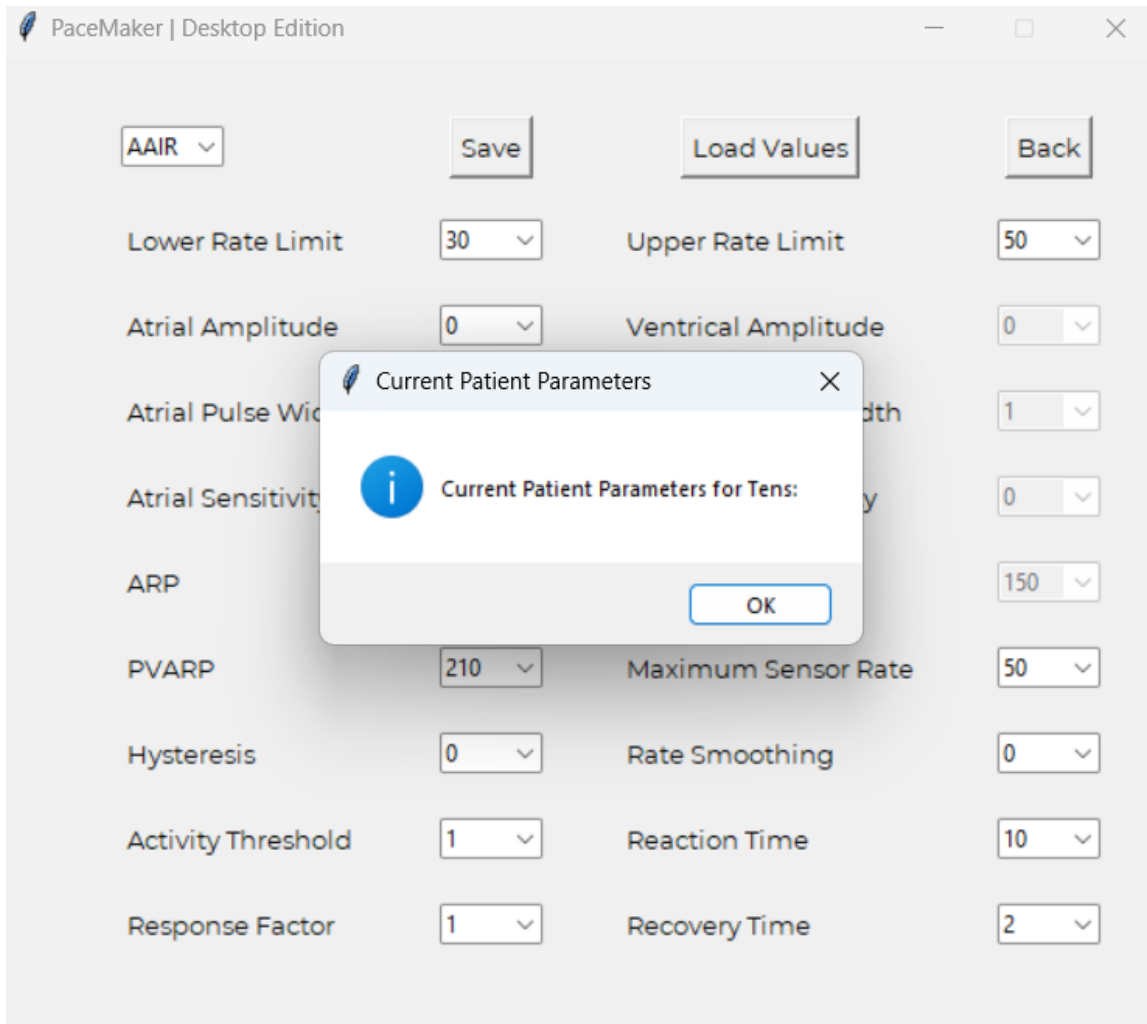


Figure 3

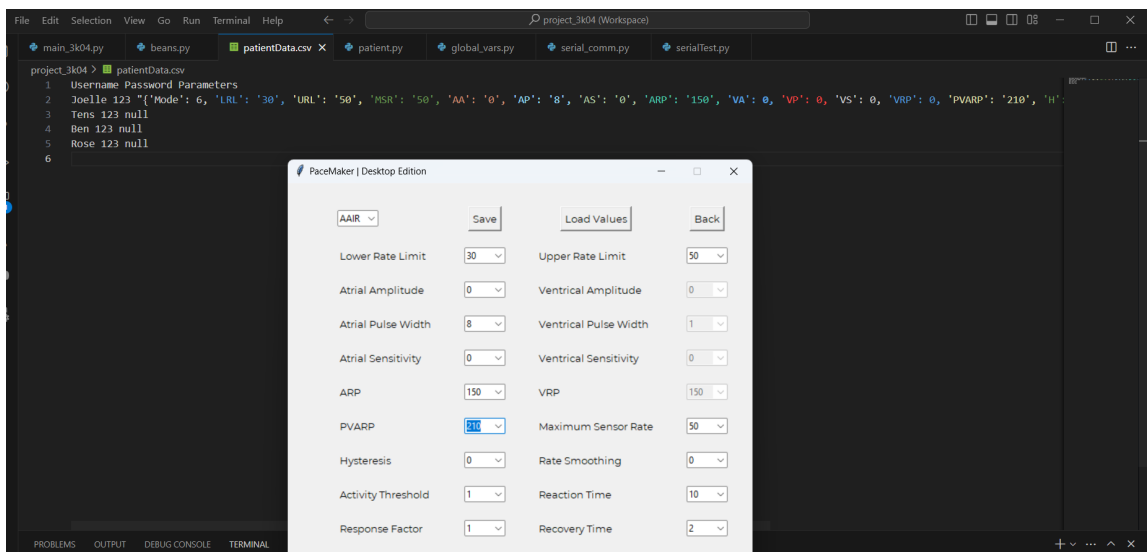


Figure 4

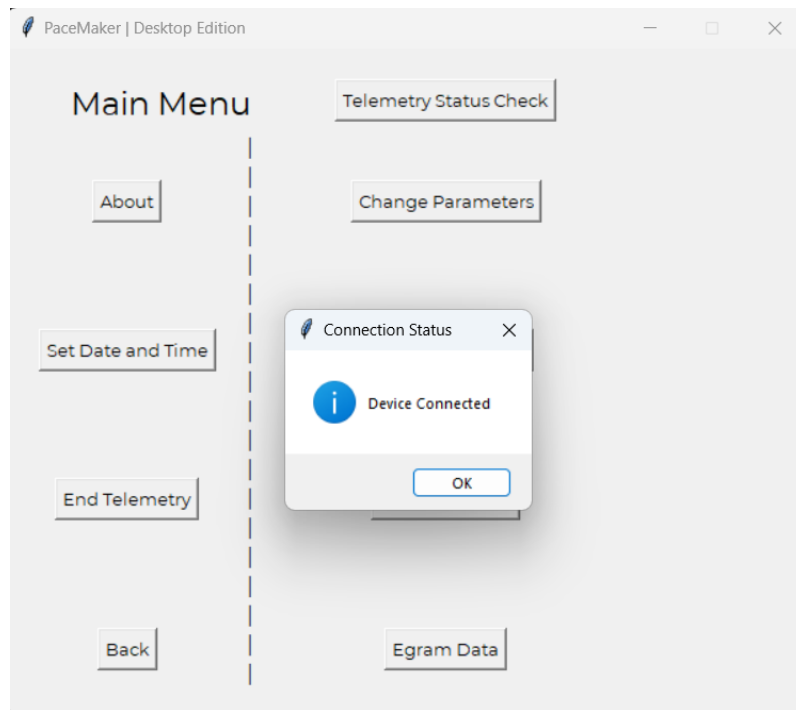


Figure 5

