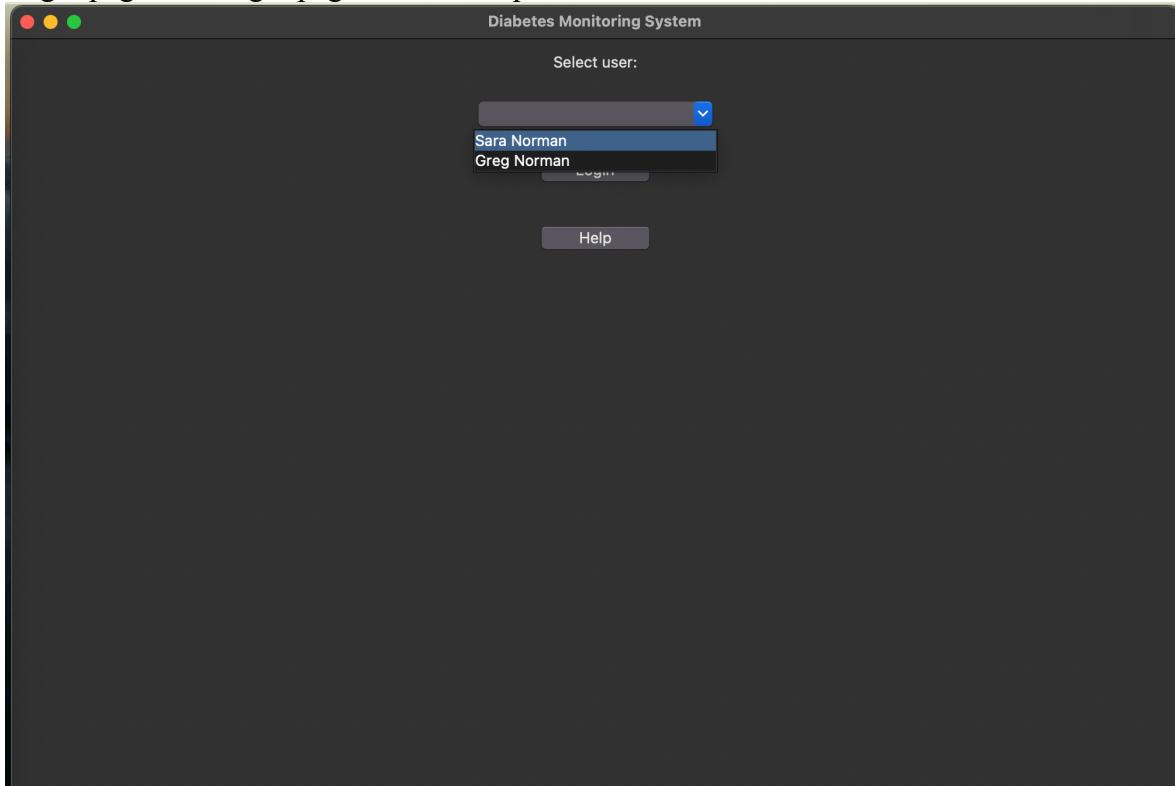


Emily Ridge  
HCI  
OU ID: 113585745

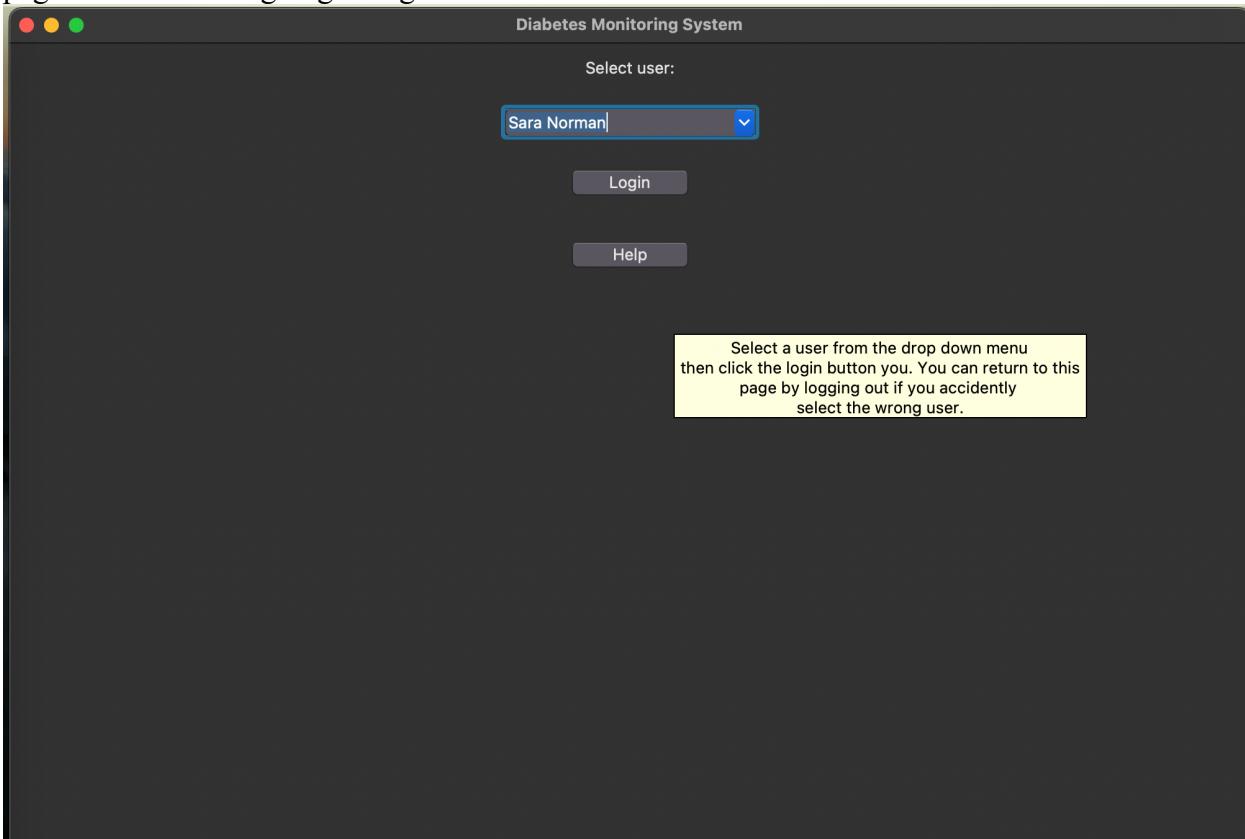
## Diabetes Monitoring System

Functionality:

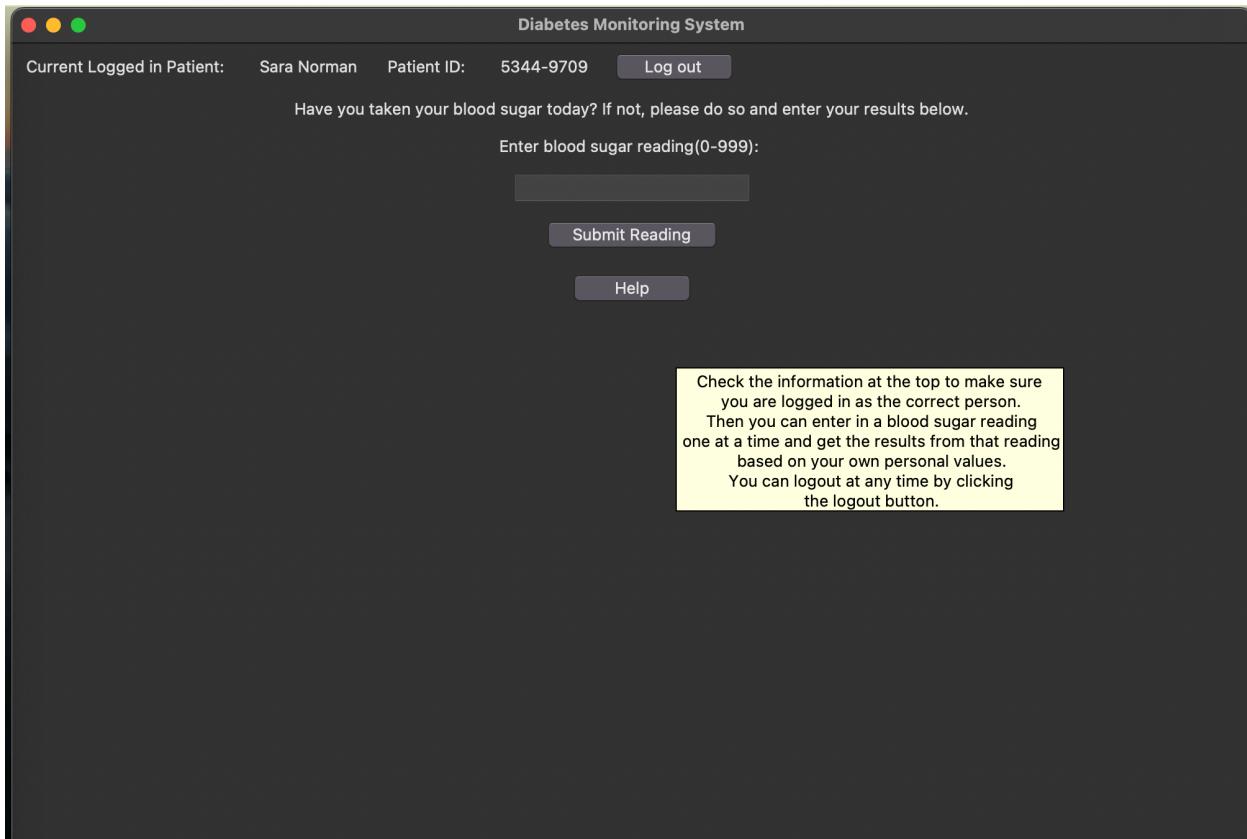
Login page: The login page offers a drop-down menu where users can select their name to log in:



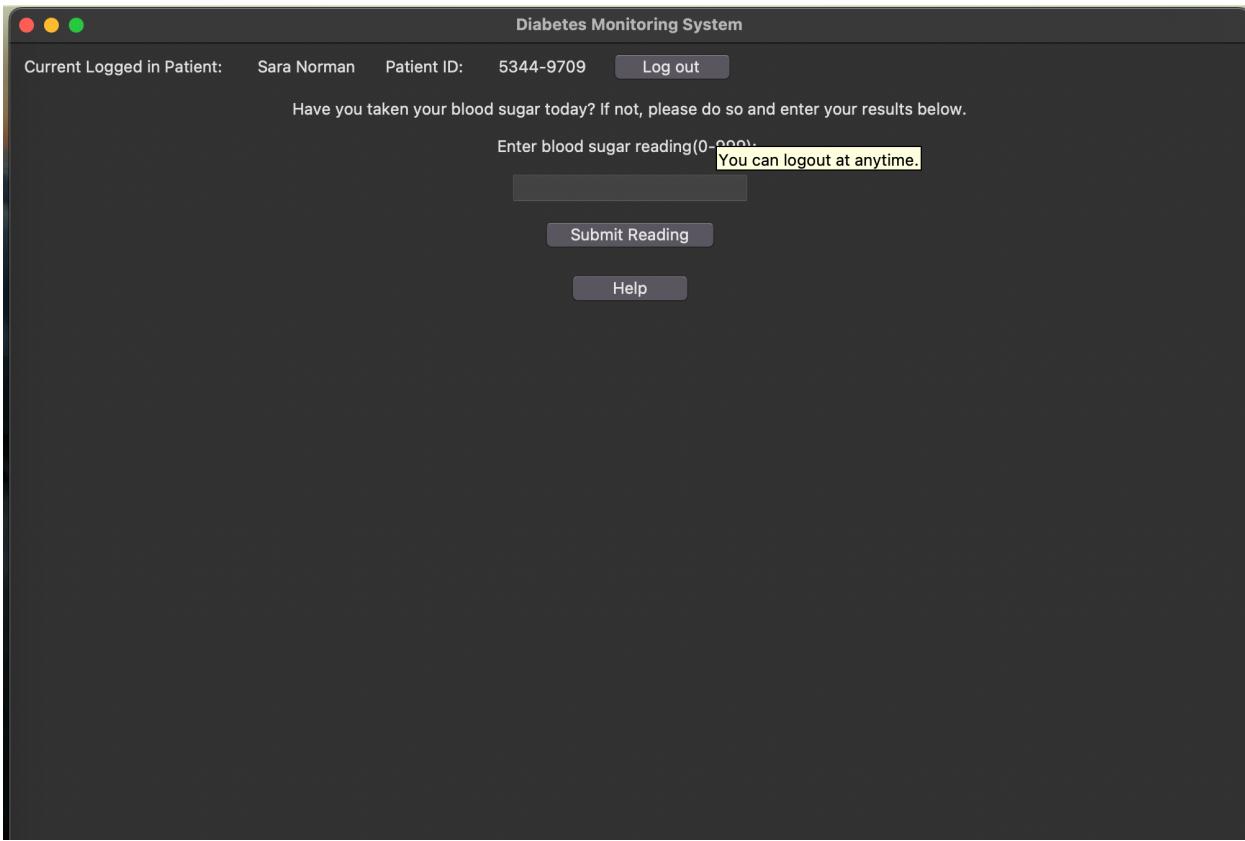
It also has a help option that the user can hover over, and it gives them a description of how the page functions. I'm going to login as Sara Norman for these screenshots:



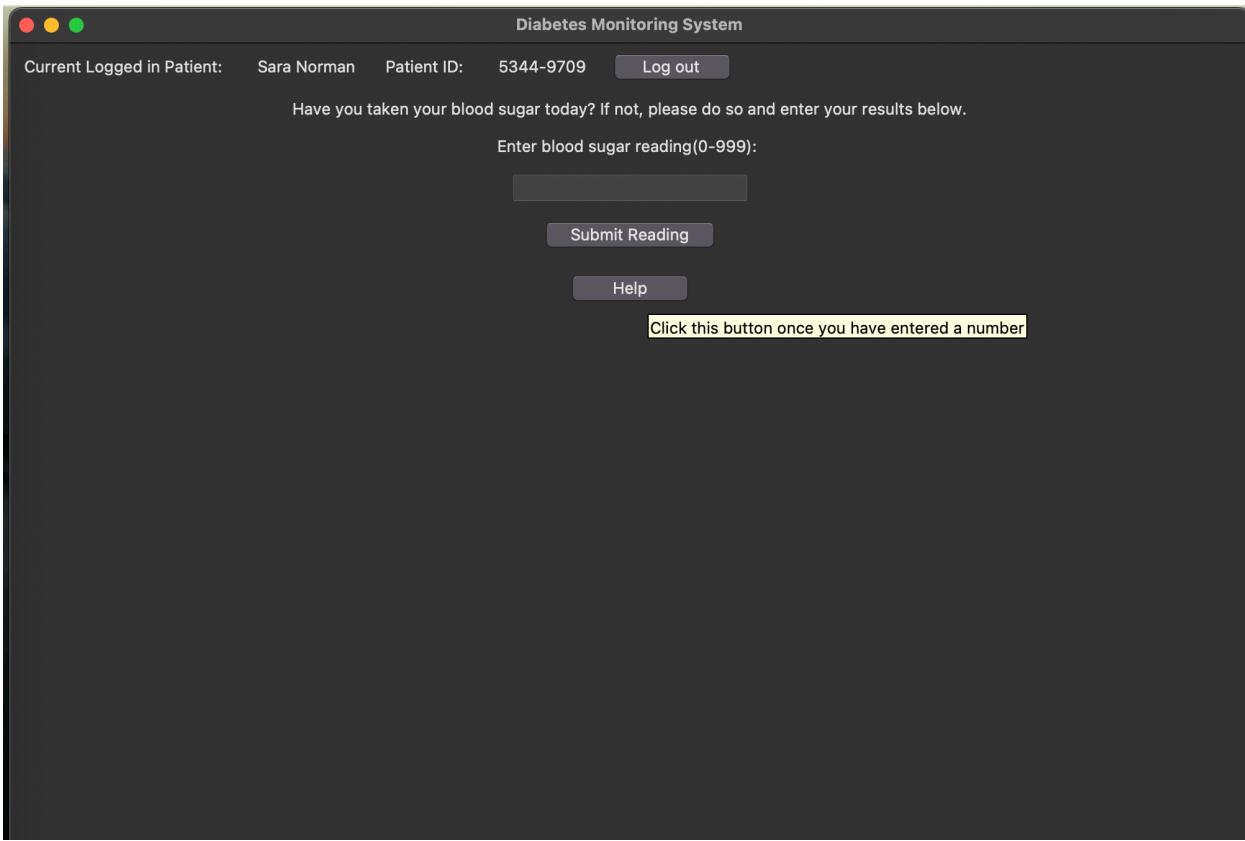
Once the login button is clicked the user is directed to the diabetes monitoring page, that also has the help option, this is what appear when you hover over the help:



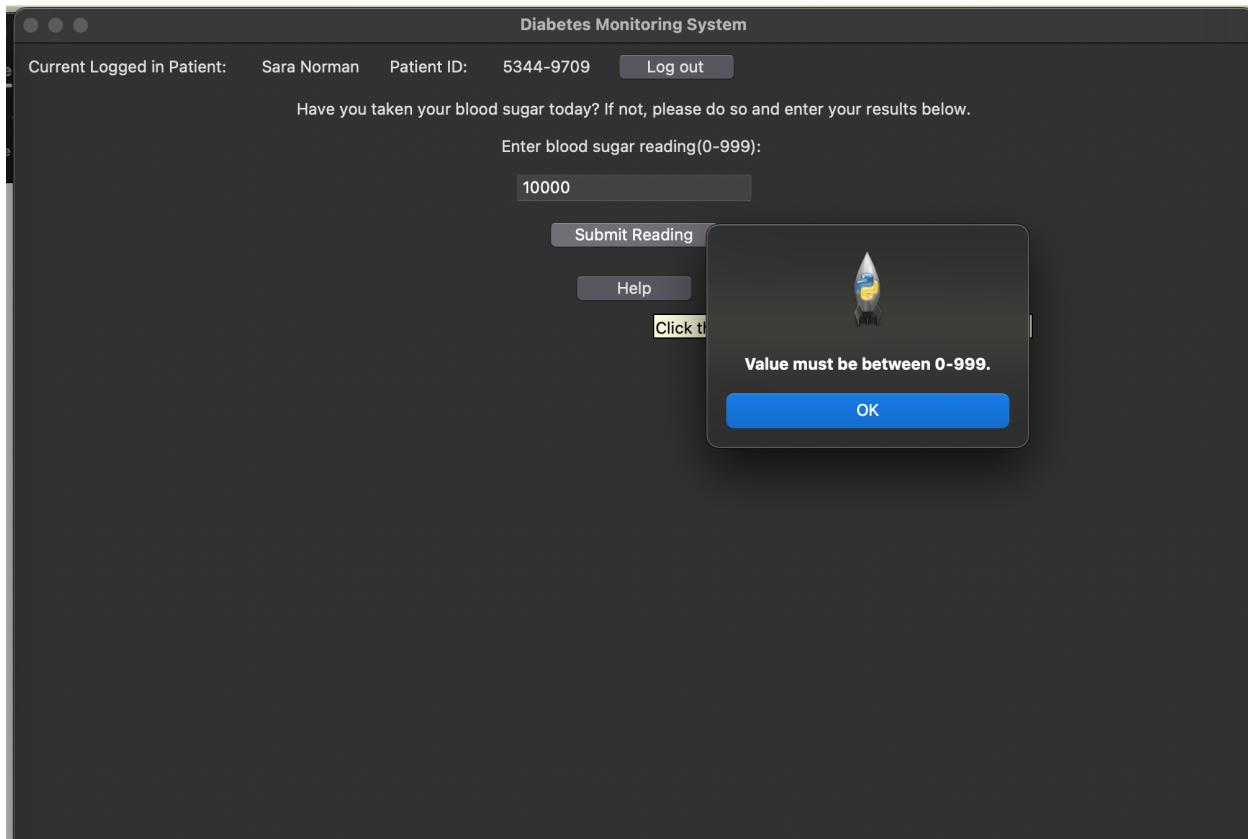
As you can see the user that is logged in, in this case Sara Norman, has their name and patient ID displayed on the top of the page. As well as a logout button with a hover option that displays:



The submit button has a hover action that displays:



The user can enter in a value between 0-200, this is error checked in the code and if they fail to meet specifications with their input an error message appears, and they get another chance to enter a valid input:



Once a valid input is entered, they will get either the normal, low, or high alert:

Diabetes Monitoring System

Current Logged in Patient: Sara Norman Patient ID: 5344-9709 Log out

Have you taken your blood sugar today? If not, please do so and enter your results below.

Enter blood sugar reading(0-999):

99

Submit Reading Help Click th



Your reading is in the healthy range!

OK

Diabetes Monitoring System

Current Logged in Patient: Sara Norman Patient ID: 5344-9709 Log out

Have you taken your blood sugar today? If not, please do so and enter your results below.

Enter blood sugar reading(0-999):

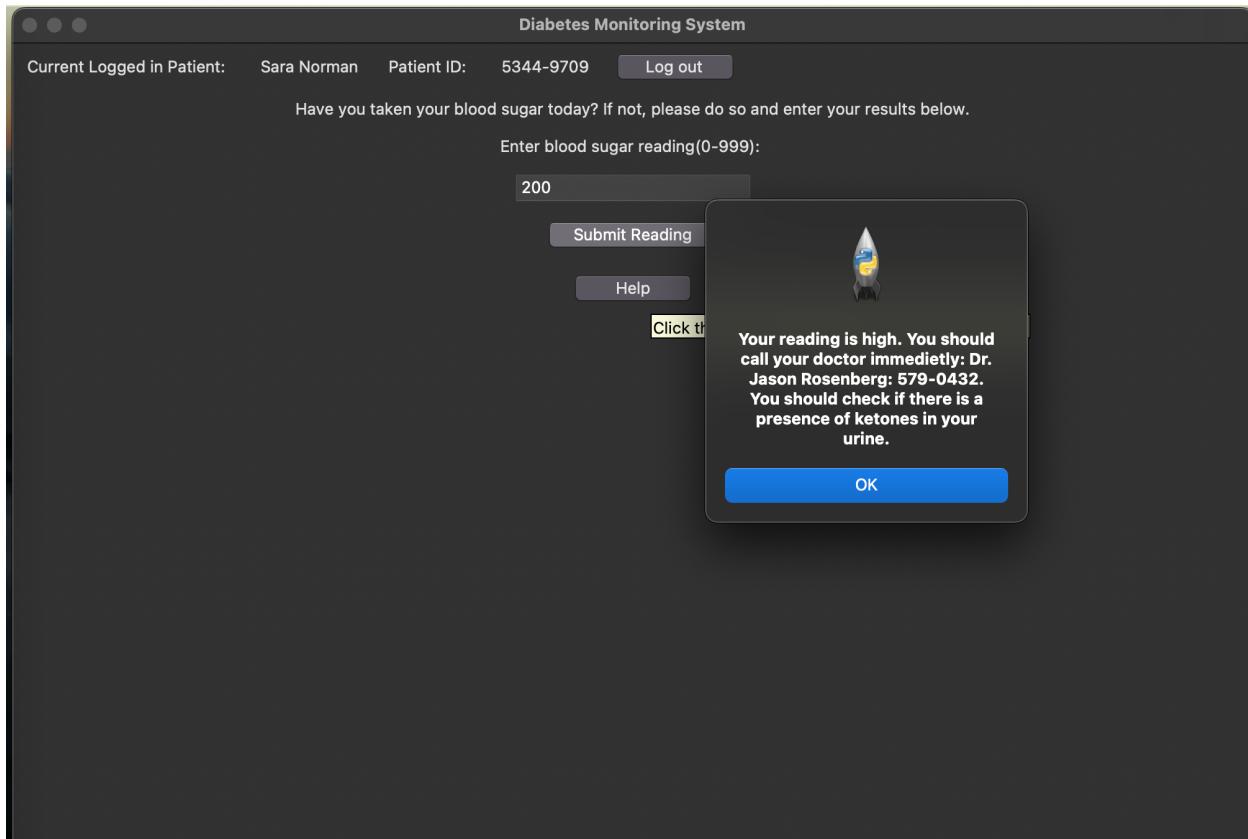
20

Submit Reading Help Click th

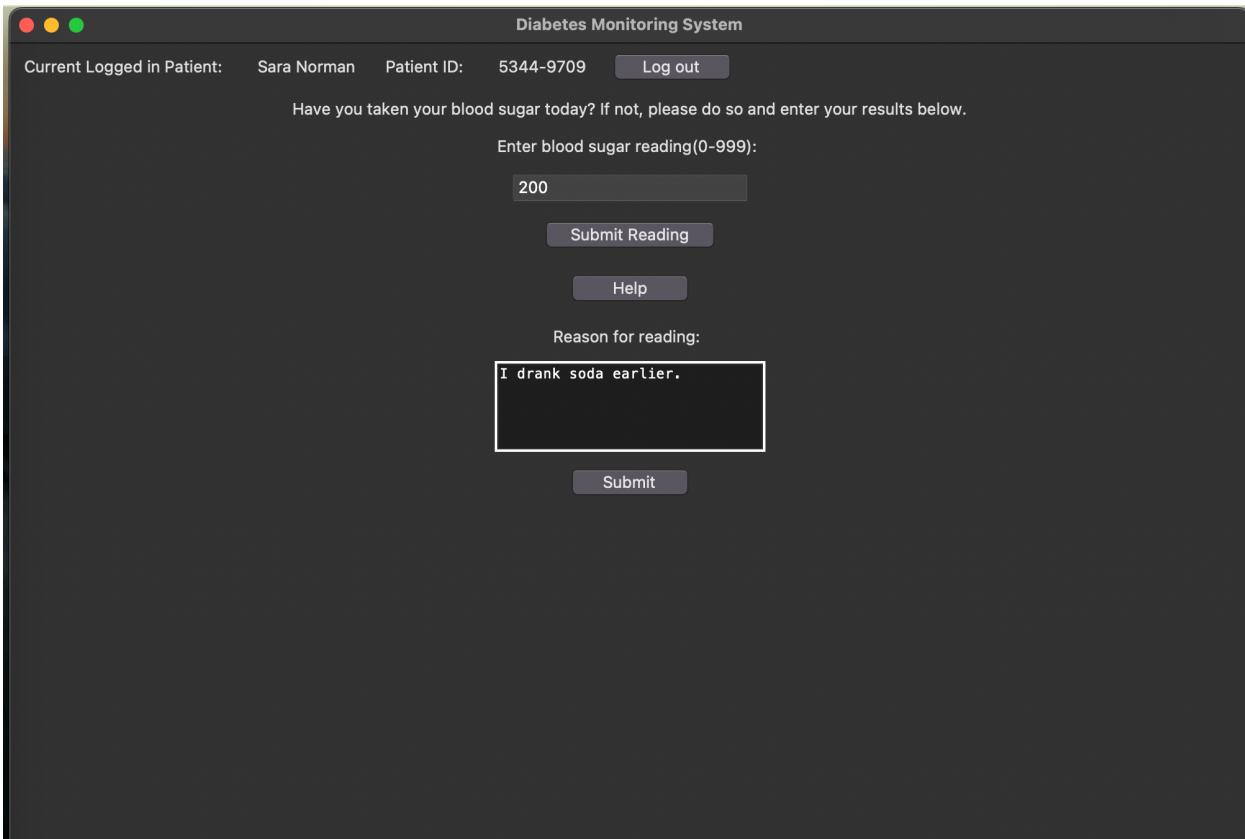


Your reading is low. To fix this you should: eat a sugar souce, take your medicine, and eat meals and snacks as described by you doctor.

OK



For the low and high alerts, after they click OK a text box appears for them to enter in a reason for this reading:



Once they hit submit, the textbox disappears, and they can submit another reading if they want. Or they can click logout and be taken back to the original login page.

Code Implementation:

GitHub Repo: [https://github.com/emilyridge/HCI\\_Diabetes\\_Monitoring.git](https://github.com/emilyridge/HCI_Diabetes_Monitoring.git)

## main.py:

The screenshot shows a code editor window with the following details:

- Tab Bar:** The tabs include "Welcome", "main.py", "header.py", "monitoring\_screen.py", and "login\_screen.py".
- Code Area:** The main.py file is open and displays Python code for a Diabetes Monitoring System using Tkinter.
- Code Content:**

```
1 import tkinter as tk
2 from login_screen import LoginScreen
3 from monitoring_screen import MonitoringScreen
4 from header import Header
5 from user_data import user_data
6
7 class DiabetesMonitoringApp(tk.Tk):
8     def __init__(self):
9         super().__init__()
10        self.title("Diabetes Monitoring System")
11        self.geometry("1000x1000")
12
13        # Stores current user once picked
14        self.current_user = None
15
16        self.login_screen = LoginScreen(self, login_callback=self.login)
17        self.header = Header(self, logout_callback=self.logout)
18        self.monitoring_screen = None
19        # Used to keep user's name and ID displayed
20
21        # Initially hide header until user is logged in
22        self.header.pack_forget()
23
24
25        # Initial screen rendered
26        self.load_login_screen()
27
28
29
30    def load_login_screen(self):
31        # Do not display header yet
32        self.header.pack_forget()
33        # Show only login screen
34        self.login_screen.pack(fill=tk.BOTH, expand = True)
35
36    def load_monitoring_screen(self):
37        # Forget login screen
38        self.login_screen.pack_forget()
39        # Display header
40        self.header.pack(fill=tk.X)
41        # Set header values
42        self.header.set_patient_data(self.current_user, user_data.get(self.current_user, {}).get("ID", ""))
43        # Show only monitoring screen
44        self.monitoring_screen = MonitoringScreen(self, current_user=self.current_user)
```
- Status Bar:** The status bar at the bottom right shows "Ln 8, Col 24" and "Spaces: 4" and "UTF-8".

The screenshot shows a code editor interface with two tabs open: 'main.py' and 'header.py'. The 'main.py' tab is active, displaying Python code for a 'DiabetesMonitoringApp' class. The code includes methods for handling user login, monitoring, and logout, along with a main block for running the application. The 'header.py' tab is also visible in the background.

```
31     # Do not display header yet
32     self.header.pack_forget()
33     # Show only login screen
34     self.login_screen.pack(fill = tk.BOTH, expand = True)
35
36 def load_monitoring_screen(self):
37     # Forget login screen
38     self.login_screen.pack_forget()
39     # Display header
40     self.header.pack(fill=tk.X)
41     # Set header values
42     self.header.set_patient_data(self.current_user, user_data.get(self.current_user, {}).get("ID", ""))
43     # Show only monitoring screen
44     self.monitoring_screen = MonitoringScreen(self, current_user=self.current_user)
45     self.monitoring_screen.pack(fill = tk.BOTH, expand = True)
46
47
48 def login(self, user_selected):
49     self.current_user = user_selected
50     print("Logging In...", user_selected)
51     self.load_monitoring_screen()
52     self.update()
53
54 def logout(self):
55     print("Logging out...")
56     self.current_user = None
57     self.monitoring_screen.pack_forget()
58     self.update()
59     self.load_login_screen()
60     self.update()
61
62
63
64
65 if __name__ == "__main__":
66     app = DiabetesMonitoringApp()
67     app.mainloop()
68
69
```

header.py:

```
❶ Welcome ❷ header.py X ❸ monitoring_screen.py ❹ login_screen.py
❺ header.py > ❻ Header > ❼ __init__
1  import tkinter as tk
2  from tkinter import ttk
3  from user_data import user_data
4  from tooltip import ToolTip
5
6  class Header(tk.Frame):
7      def __init__(self, master = None, logout_callback=None, **kwargs):
8          super().__init__(master, **kwargs)
9          self.master = master
10
11         self.logout_callback = logout_callback
12
13         self.selected_user = None
14
15         # Variables to keep displayed in header
16         self.patient_name_label = ttk.Label(self, text="Current Logged in Patient: ")
17         self.patient_id_label = ttk.Label(self, text="Patient ID: ")
18
19         self.patient_name_display = ttk.Label(self, text="")
20         self.patient_id_display = ttk.Label(self, text="")
21
22
23         self.logout_button = ttk.Button(self, text="Log out", command=self.logout_button_clicked)
24         tooltip_button = ToolTip(self.logout_button, "You can logout at anytime.")
25
26
27         # Layout
28         self.patient_name_label.pack(side="left", padx=10, pady=5)
29         self.patient_name_display.pack(side="left", padx=10, pady=5)
30         self.patient_id_label.pack(side="left", padx=10, pady=5)
31         self.patient_id_display.pack(side="left", padx=10, pady=5)
32         self.logout_button.pack(side="left", padx=10, pady=5)
33
34
35
36     def set_patient_data(self, selected_user, patient_id):
37         # Set variables
38         self.selected_user = selected_user
39         self.patient_name_display.config(text=selected_user)
40         self.patient_id_display.config(text=patient_id)
41
42
43     def logout_button_clicked(self):
44         if self.logout_callback:
```

Ln 24, Col 81   Spaces: 4   UTF-8

The screenshot shows a code editor with two tabs open: 'header.py' and 'login\_screen.py'. The 'header.py' tab is active, displaying Python code for a Tkinter application. The code defines a 'Header' class that inherits from 'tk.Frame'. It initializes variables for patient name and ID labels, a logout button, and a tooltip. It also sets up the layout for these elements. The 'set\_patient\_data' method updates the display for a selected user. The 'logout\_button\_clicked' method triggers the logout callback. The 'login\_screen.py' tab is visible in the background.

```
❶ Welcome ❷ header.py X ❸ monitoring_screen.py ❹ login_screen.py
❺ header.py > ❻ Header > ❼ __init__
  2  from tkinter import ttk
  3  from user_data import user_data
  4  from tooltip import ToolTip
  5
  6  class Header(tk.Frame):
  7      def __init__(self, master = None, logout_callback=None, **kwargs):
  8          super().__init__(master, **kwargs)
  9          self.master = master
 10
 11          self.logout_callback = logout_callback
 12
 13          self.selected_user = None
 14
 15          # Variables to keep displayed in header
 16          self.patient_name_label = ttk.Label(self, text="Current Logged in Patient: ")
 17          self.patient_id_label = ttk.Label(self, text="Patient ID: ")
 18
 19          self.patient_name_display = ttk.Label(self, text="")
 20          self.patient_id_display = ttk.Label(self, text="")
 21
 22
 23          self.logout_button = ttk.Button(self, text="Log out", command=self.logout_button_clicked)
 24          tooltip_button = ToolTip(self.logout_button, "You can logout at anytime.")
 25
 26
 27          # Layout
 28          self.patient_name_label.pack(side="left", padx=10, pady=5)
 29          self.patient_name_display.pack(side="left", padx=10, pady=5)
 30          self.patient_id_label.pack(side="left", padx=10, pady=5)
 31          self.patient_id_display.pack(side="left", padx=10, pady=5)
 32          self.logout_button.pack(side="left", padx=10, pady=5)
 33
 34
 35
 36      def set_patient_data(self, selected_user, patient_id):
 37          # Set variables
 38          self.selected_user = selected_user
 39          self.patient_name_display.config(text=selected_user)
 40          self.patient_id_display.config(text=patient_id)
 41
 42
 43      def logout_button_clicked(self):
 44          if self.logout_callback:
 45              self.logout_callback()

```

Ln 24, Col 81 Spaces: 4 UTF-8 L

login\_screen.py:

The screenshot shows a code editor with three tabs open:

- monitoring\_screen.py
- login\_screen.py
- login\_screen.py (active tab)

The active tab, `login_screen.py`, contains the following Python code:

```
1 import tkinter as tk
2 from tkinter import ttk
3 from user_data import user_data
4 from tooltip import ToolTip
5
6 class LoginScreen(tk.Frame):
7     def __init__(self, master=None, login_callback=None, **kwargs):
8         super().__init__(master, **kwargs)
9         self.master = master
10        self.login_callback = login_callback
11
12        # To display
13        self.label = ttk.Label(self, text="Select user:")
14        self.user_options = ttk.Combobox(self, values=["Sara Norman", "Greg Norman"])
15        self.login_button = ttk.Button(self, text="Login", command=self.login_button_clicked)
16        self.help_button = ttk.Button(self, text="Help")
17        tooltip_help_button = ToolTip(self.help_button, "Select a user from the drop down menu\nthen click the login button you."
18
19        self.label.pack(pady=10)
20        self.user_options.pack(pady=10)
21        self.login_button.pack(pady=10)
22        self.help_button.pack(pady=20)
23
24    def login_button_clicked(self):
25        # Get user that was selected
26        user_selected = self.user_options.get()
27        if self.login_callback:
28            self.login_callback(user_selected)
29
30
31
32
33
34
35
```

At the bottom right of the editor window, there is a status bar displaying: Ln 21, Col 40 Spaces: 4 UTF-8 L.

monitoring\_screen.py:

```
❶ Welcome ❷ monitoring_screen.py x
❸ monitoring_screen.py > MonitoringScreen > __init__
1  import tkinter as tk
2  from tkinter import ttk
3  from tkinter import messagebox
4  from user_data import user_data
5  from tooltip import ToolTip
6
7
8  class MonitoringScreen(tk.Frame):
9      def __init__(self, master=None, current_user = None, **kwargs):
10          super().__init__(master, **kwargs)
11          self.master = master
12          self.current_user = current_user
13
14          self.tell_label = ttk.Label(self, text="Have you taken your blood sugar today? If not, please do so and enter your results below")
15          self.tell_label.pack(pady=5)
16          self.label = ttk.Label(self, text="Enter blood sugar reading(0-999): ")
17          self.label.pack(pady=5)
18          self.entry = ttk.Entry(self)
19          tooltip_entry = ToolTip(self.entry, "You should submit one value at a time.")
20          self.entry.pack(pady=5)
21          self.button = ttk.Button(self, text="Submit Reading", command=self.check_reading)
22          tooltip_button = ToolTip(self.button, "Click this button once you have entered a number")
23          self.button.pack(pady=5)
24          self.help_button = ttk.Button(self, text="Help")
25          tooltip_help_button = ToolTip(self.help_button, "Check the information at the top to make sure\n you are logged in as the correct user")
26          self.help_button.pack(pady=10)
27
28
29
30
31      def check_reading(self):
32          user_input = self.entry.get()
33          user_low_val = int(user_data.get(self.current_user, {}).get("LowReading", "0"))
34          user_high_val = int(user_data.get(self.current_user, {}).get("HighReading", "0"))
35          user_dr_name = user_data.get(self.current_user, {}).get("DoctorName", "")
36          user_dr_num = user_data.get(self.current_user, {}).get("DoctorPhone", "")
37
38
39          try:
40              user_input = int(user_input)
41              if user_input >= 0 and user_input <= 999:
42                  if user_input < user_low_val:
43                      messagebox.showinfo("Message", "Your reading is low. To fix this you should: eat a sugar souce, take your medicine, or call your doctor." )
44
Ln 25, Col 148  Spaces: 4  UTF-8  L
```

The screenshot shows a code editor window with a dark theme. The title bar says "monitoring\_screen.py". The code itself is a Python script for a monitoring screen. It uses the Tkinter library for a graphical user interface. The script checks a user input value against three thresholds: low\_val, user\_low\_val, and user\_high\_val. If the value is below low\_val, it shows a message about eating sugar. If it's above high\_val, it advises calling a doctor immediately. If it's in the healthy range, it informs the user. It also handles errors if the input is not a number. Below the main function, there are two helper methods: one for explaining the reason for reading and another for submitting that reason.

```
41     if user_input < user_low_val:
42         messagebox.showinfo("Message", "Your reading is low. To fix this you should: eat a sugar souce, take your medi
43         # ask user for reason for value
44         self.explain_reason()
45
46     elif user_input > user_high_val:
47         messagebox.showinfo("Message", "Your reading is high. You should call your doctor immedietly: " + user_dr_name
48         # ask user for reason for value
49         self.explain_reason()
50
51     else:
52         messagebox.showinfo("Message", "Your reading is in the healthy range!")
53
54
55
56
57     else:
58         messagebox.showerror("Error", "Value must be between 0-999.")
59         self.entry.delete(0, tk.END)
60
61 except ValueError:
62     messagebox.showerror("Error", "Invalid input, you must enter a number.")
63     self.entry.delete(0, tk.END)
64
65
66
67
68
69 def explain_reason(self):
70     if not hasattr(self, 'text_label'):
71         self.text_label = ttk.Label(self, text="Reason for reading: ")
72         self.text_label.pack(pady=5)
73         self.text_box = tk.Text(self, height=5, width=30)
74         self.text_box.pack(pady=5)
75         self.submit_button = ttk.Button(self, text="Submit", command=self.submit_reason)
76         self.submit_button.pack(pady=5)
77
78
79 def submit_reason(self):
80     self.text_label.destroy()
81     self.text_box.destroy()
82     self.submit_button.destroy()
83     self.entry.delete(0, tk.END)
84
```

Ln 66, Col 9 Spaces: 4 UTF-8 L

tooltip.py:

The screenshot shows a code editor interface with two tabs: "tooltip.py" and "user\_data.py". The "tooltip.py" tab is active, displaying Python code for creating a tooltip class using Tkinter. The "user\_data.py" tab is visible but empty. The code in tooltip.py is as follows:

```
tooltip.py > ToolTip > hide_tooltip
1 # Used https://www.w3resource.com/python-exercises/tkinter/python-tkinter-custom-widgets-and-themes-exercise-9.php
2 # as resource for building this class
3
4 import tkinter as tk
5
6 class ToolTip:
7     def __init__(self, widget, text):
8         self.widget = widget
9         self.text = text
10        self.tooltip_visible = False
11
12        self.widget.bind("<Enter>", self.show_tooltip)
13        self.widget.bind("<Leave>", self.hide_tooltip)
14
15    def show_tooltip(self, event=None):
16        if not self.tooltip_visible:
17            x, y, _, _ = self.widget.bbox("insert")
18            x += self.widget.winfo_rootx() + 25
19            y += self.widget.winfo_rooty() + 25
20
21            self.tooltip_label = tk.Label(text=self.text, background="lightyellow", foreground="black", relief="solid", borderwidth=1)
22            self.tooltip_label.place(x=x, y=y)
23            self.tooltip_visible = True
24
25    def hide_tooltip(self, event=None):
26        if self.tooltip_visible:
27            self.tooltip_label.place_forget()
28            self.tooltip_visible = False
29
30
31
```

The status bar at the bottom of the editor indicates: Ln 28, Col 41 Spaces: 4 UTF-8 LF ⌂ P

user\_data.py:

The screenshot shows a code editor window with a dark theme. The title bar says "Welcome" and the tab bar has a file named "user\_data.py". The code itself is a Python dictionary definition:

```
1 # Hard coded user data given
2 user_data = {
3     "Sara Norman": {
4         "ID" : "5344-9709",
5         "DoctorName" : "Dr. Jason Rosenberg",
6         "DoctorPhone" : "579-0432",
7         "LowReading" : "80",
8         "NormalReading" : "80-140",
9         "HighReading" : "140"
10    },
11    "Greg Norman": {
12        "ID" : "1275-4307",
13        "DoctorName" : "Dr. Nikhil Singh",
14        "DoctorPhone" : "334-2309",
15        "LowReading" : "70",
16        "NormalReading" : "70-120",
17        "HighReading" : "120"
18    }
19 }
20
21 # Ex access: sara's data = user_data["Sara Norman"], id = sara's data["ID"]
```

At the bottom right of the editor, there is status information: "Ln 3, Col 21 Spaces: 4 UTF-8 LF ⌂ P".