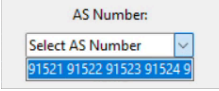
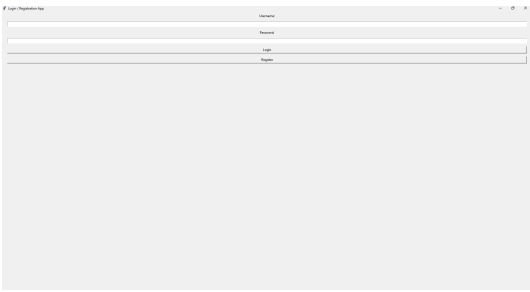


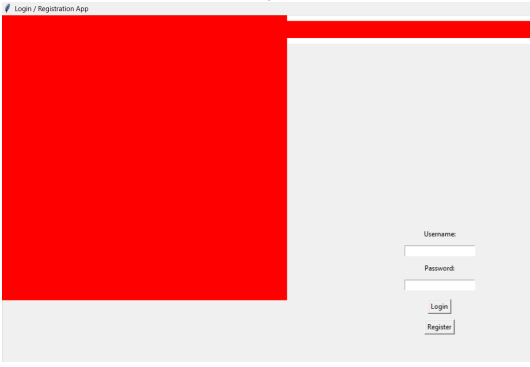
Code testing	Input tested	Desired (Y/N)	Fixed how
Login Testing			
<pre> With open('users.txt', 'a') as file:     file.write(f"{username},{password}\n")      messagebox.showinfo("Registration Success", "Registration successful! You can now log in.") </pre>	Qwe,wde (registering)	Yes, allowed registration	N/A
<pre> '' </pre>	" (login)	Yes allowed login	N/A
<pre> messagebox.showwarning("Registration Failed", "Username already exists!") </pre>	Dufnv, sdjnf (was already used)	Yes	N/A
<pre> messagebox.showerror("Login Failed", "Invalid username or password.") </pre>	Wedf,qwsd (not registered)	Yes	N/A
Adding Assessment Testing			
<pre> number_dropdown = ttk.Combobox(root, textvariable=number_var, values=[AS_NUMBERS], state="readonly") </pre>	Click	No, would display all standards in a line 	Had to change values to the individual numbers rather than the dictionary AS_NUMBERS
<pre> tk.Label(new_assessment_frame, text="AS Name:").pack(pady=5) name_var = tk.StringVar() name_dropdown = ttk.Combobox(new_assessment_frame, textvariable=name_var, values=[details["Name"] for details in standard_details.values()], state="readonly") name_dropdown.set("Select AS Name") name_dropdown.pack(pady=5) </pre>	Click	No, would cut off the names of the assessments.	Had to change the max length to set dynamically.

<pre>max_length = max(len(name) for name in standard_details.values())  tk.Label(new_assessment_frame, text="AS Name:").pack(pady=5) name_var = tk.StringVar() name_dropdown = ttk.Combobox(new_assessment_frame, textvariable=name_var, values=[details["Name"] for details in standard_details.values()], state="readonly", width=max_length) name_dropdown.set("Select AS Name") name_dropdown.pack(pady=5)</pre>	Click	No, made the box even smaller	Had to change <b>max_length = max(len(name) for name in standard_details.values())</b> to <b>max_length = max(len(details["Name"])) for details in standard_details.values())</b>
Menu Testing			
<pre>def page_menu():     def collapse_menu():         toggle_menu.destroy()  toggle_button.configure(text='☰') toggle_menu = tk.Frame(root, bg='red')     window_height = root.winfo_height()     window_width = root.winfo_width()     toggle_menu.place(x=0, y=50, height=window_height, width= window_width * 0.25)     toggle_button.config(text ="X" )  toggle_button.config(command=collapse_men u)</pre>	Home button	No, would open the assessments page on the same page multiple times	Had to add a function to destroy the widget if it was already there.
<pre>as_btn = tk.Button(toggle_menu, text='New Assessment', font=('Bold, 20'), fg='white', command=new_assessment) as_btn.place(x=20,y=140)</pre>	Click on New assessment page	Yes, would open the form for new assessments	N/A
Adding Assessment to home page			

<pre>for assessment in assessments_list:     tree.insert("", "end", values=(         assessment["AS Number"],         assessment["Name"],         assessment["Credits"],         assessment["Type"],         assessment["Due Date"],         assessment["Test Type"]     ))</pre>	91521	Yes would add all of the assessments	N/A
	91522		
	91523		
	91524		
	91525		
	91526		
	91527		
<pre>''' '''</pre>	Pressed submit twice	No, would add the same assessment and data twice.	Needed to edit the function to check whether the as number was already there * also added ability to make changes
<pre>''' '''</pre>	Pressed submit without filling anything out	No, need it to come back with an error but it didn't	Needed to add an if statement to check that all the fields are filled.

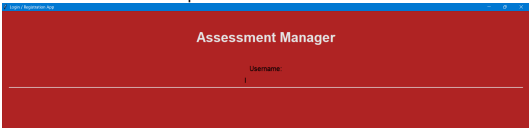
Problem code	Problem	Solution
<pre>1 Create the main window 2 3 window = tk.Tk() 4 window.title("HomePage") 5 window.geometry("400x300")</pre>	Wouldn't create the new window to the new page	Going back to the login and registration code and adding <b>login_window.destroy()</b> to close the login window and <b>assessment_page()</b> to create the new page.
<pre>login_window.resizable(width=True, height=True)</pre>	Would resize all contents and disregard padding constants	Added a frame using <b>..._frame</b>

		
	And looked ugly	
<pre>btn_NewA = tk.Button(assessment_frame, text="New Assessment", width=20, command=new_assessment) btn_NewA.pack(pady=10)</pre>	The same code would only work when logging/registering, would resize the window to the original size and not full screen.	Changed code so that it doesn't open a new window everytime and just stays on the same frame
<pre>def toggle_fullscreen(event=None):     current_fullscreen = root.attributes('-fullscreen') root.attributes('-fullscreen', not current_fullscreen)</pre>	This button on the main page wouldn't take me to another page for the new assessments	Was fixed by adding a <b>global</b> function in <b>def new_assessments</b>
<pre>if selected_number == 91521:     name_var.set(standard_details["Name"]) credits_var.set(standard_details["Credits"]) type_var.set(standard_details["Type"])</pre>	The three buttons that normally appear in the top right corner (minimise, exit) aren't there when it runs initially.	
	Would come up with error- name_var.set(standard_details["Name"]) ^^^^^^ SyntaxError: invalid syntax	Added [] around "Name" etc
<pre>if selected_number == 91521:     name_var.set(standard_details["Name"]) credits_var.set(standard_details["Credits"])</pre>	Error - name_var.set(standard_details["Name"]) ~~~~~ KeyError: 'Name'	Fixed by changing <b>if selected_number ==...</b> to <b>if selected_standard in standard_details:</b>

<pre>type_var.set(standard_details["Type"])</pre>		
<pre>def side_bar():     head_frame = tk.Frame(root, bg='red')     head_frame.pack(side=tk.TOP, fill=tk.X)     head_frame.configure(height=50)</pre>	Side bar would show up at the bottom	Had to call the sidebars function before the login frame
<pre>toggle_menu = tk.Frame(root, bg='red') toggle_menu.place(x=0, y=0, height=500, width=     500)</pre>	Would overlap the header 	Had to change <b>y=0</b> to <b>y=50</b>
<pre>def side_bar():     # designing the frame     head_frame = tk.Frame(root, bg='red',         highlightbackground='white',         highlightthickness=10)      toggle_button = tk.Button(head_frame,         text='☰', fg="white", bg="red", font=20)     toggle_button.pack(side=tk.LEFT)      head_frame.pack(side=tk.TOP, fill=tk.X)</pre>	The menu bar would appear continuously and there was no way to get rid of it	Had to make <b>toggle_menu = tk.Frame(root, bg='red')</b> <b>toggle_menu.place(x=0, y=50, height=500, width= 500)</b> Into a function

<pre>head_frame.pack_propagate(False) head_frame.configure(height=50)  toggle_menu = tk.Frame(root, bg='red') toggle_menu.place(x=0, y=50, height=500,                   width= 500)</pre>		
<pre>'''</pre>	Wouldn't reach the whole page	Had to change height to a variable and use <code>root.winfo_height()</code> to get the height of the screen
<pre>def page_menu():     def collapse_menu():         toggle_menu.destroy()         toggle_button.configure(text='≡')         toggle_menu = tk.Frame(root, bg='red')         window_height = root.winfo_height()         window_width = root.winfo_width()         toggle_menu.place(x=0, y=50,             height=window_height, width= window_width * 0.25)         toggle_button.config(text = "X" )         toggle_button.config(command=collapse_menu)</pre>	<b>Toggle_button</b> would have a yellow underline meaning that it was undefined and code wouldn't work	Had to make it a global function <b>global toggle_button</b>
<pre>'''</pre>	Side bar would only open once	Had to create separate functions for the opening and closing of the menu so that it can be called multiple times.
<pre>def page_menu():     def collapse_menu():         global toggle_menu         toggle_menu.pack_forget()         toggle_button.config(text='≡')</pre>	Now it wouldn't close at all	Had to change <b>toggle_menu.pack_forget()</b> to <b>toggle_menu.place_forget()</b> So that it would forget where it was placed rather than the whole function

<pre>toggle_button.config(command=expand_menu) def expand_menu():     global toggle_menu     if toggle_menu is None:         toggle_menu = tk.Frame(root,             bg='#253B80')         toggle_menu = tk.Frame(root,             bg='#253B80')         main_btn = tk.Button(toggle_menu,             text='Home', font=('Bold', 20), bg='#253B80',             fg='black', command=show_assessment_page,             relief="flat")         main_btn.place(x=20, y=20)          as_btn = tk.Button(toggle_menu, text='New Assessment', font=('Bold', 20), bg='#253B80',             fg='black', command=new_assessment, relief="flat")         as_btn.place(x=20, y=140)          window_height = root.winfo_height()         window_width = root.winfo_width()         toggle_menu.place(x=0, y=50, height=window_height,             width=window_width * 0.25)         toggle_button.config(text="X")  toggle_button.config(command=collapse_menu) expand_menu() toggle_button.config(command=collapse_menu)</pre>		
<pre># Create buttons for pages as_btn = tk.Button(toggle_menu, text='New Assessment', font=('Bold', 20), fg='white',     command=new_assessment) as_btn.place(x=20,y=140)</pre>	Would move the menu bar to the bottom of the page	Needed to move the call function of the sidebar so that it would get added before the widgets.

<pre>submit_button = tk.Button(new_assessment_frame,     text="Submit", command=lambda: add_assessment(number_var, name_var, credits_var,     type_var)) submit_button.pack(side=tk.BOTTOM, pady=20)</pre>	Would print the test type as external rather than test or essay	Had to add <b>test_var</b> to the def new assessments function to solve
<pre>entry_username = tk.Entry(root, bg="#B02324",     text= " ", relief="flat")</pre>	Wouldn't have a line underneath and the whole box would just disappear	Had to add a separate frame to act as the line.
<pre>entry_username = tk.Entry(root, bg="#B02324",     relief="flat") entry_username.pack(pady=10) username_underline = tk.Frame(root, height=2,     bg="#B6B6B6") username_underline.pack(fill="x", padx=20,     pady=(0, 10))</pre>	There would be a space between the line and the actual box 	Had to get rid of the padding for the entry box section



<pre>entry_username = tk.Entry(root, bg="#B02324",                            relief="flat") entry_username.pack()  username_underline = tk.Frame(root, height=2,                               bg="#E6E6E6") username_underline.pack(fill="x", padx=20,                         pady=(0, 10))</pre>	<p>The input wouldn't go to the end of the line and was centered to the middle</p>	<p>Had to match the <b>fill</b> and <b>padx</b> so that the line started and ended where the input box was</p>
--	--	--