Complete and submit your final project.

# 1 (required) Readme Documentation

To submit your assignment:

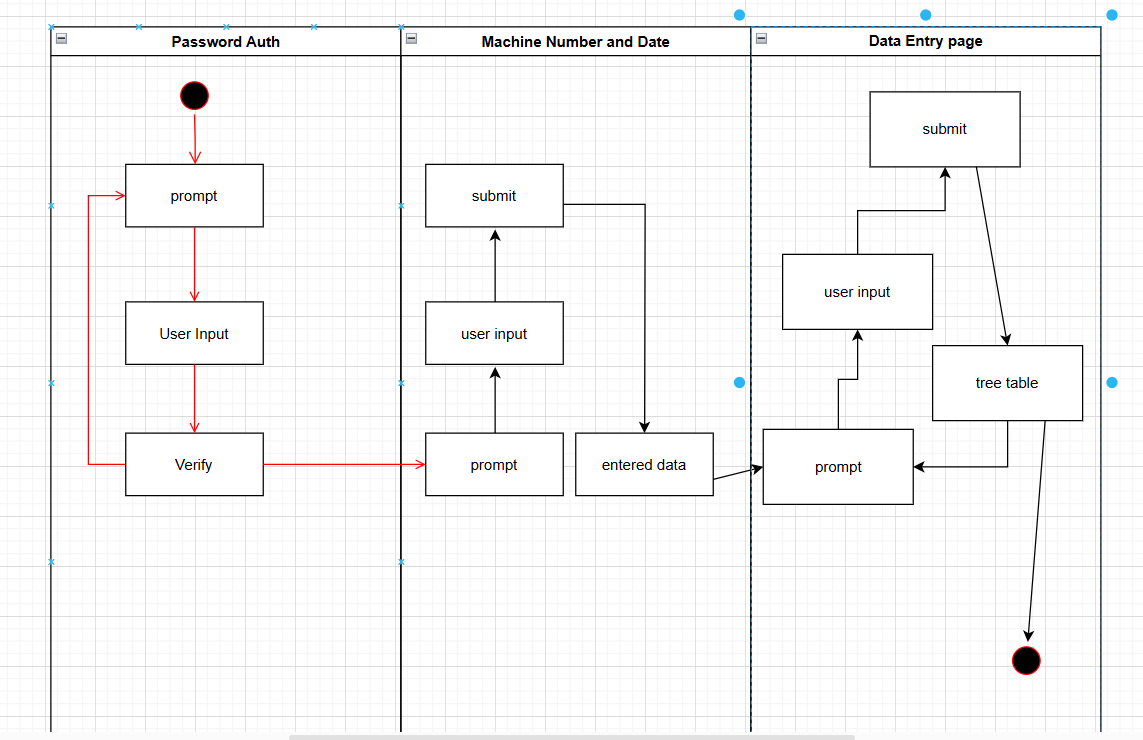
1. Submit using [Assignment Submission Template](https://ivylearn.ivytech.edu/courses/1271405/files/118978825?wrap=1)[Download Assignment Submission Template](https://ivylearn.ivytech.edu/courses/1271405/files/118978825/download?download_frd=1)rename the file LNameFNameFP.docx (using your name) - 130 points.
   * Optional sections are not optional for your Final Project (See list of requirements)
   * Add / Remove sections to the Template as necessary for the Final Project
   * Documentation of source code using [Good Programming Practices](https://ivylearn.ivytech.edu/courses/1271405/pages/good-programming-standards)
2. Submit the link to the GitHub repository for your final project
   * Use the repository and push all of the components of your Final Project onto GitHub (Make sure the repository is Public so that your instructor can view it).

**Please follow the following requirements for your final project, this shows a detailed breakdown of how the points are distributed:**

1. Your Python Application should include the following (worth 130 points):
   * A working GUI tkinter application with at least two windows.  - 50 points
   * Implementing a modular approach in your application.  - 10 points
   * Consistent clear navigation throughout the GUI application.  - 10 points
   * Use at least two images in your application(images should have alternate text).  - 10 points
   * Include at least three labels. - 10 points
   * Include at least three buttons. - 10 points
   * Include at least three call-back functions with each button, including the exit button. - 20 points
   * Implement secure coding best practices, including input validation to check if the user entered the correct data type, make sure the entry box is not empty, etc.  - 10 points
2. Validation testing. Include all the following items: (20 points)
   * Develop an appropriate set of test data to fully validate the program.
   * The data sets you tested against uploaded to the GitHub repository.
   * A  brief written explanation of the results of your tests and what you had to fix.
   * At least three screenshots of your good test data working in your assignment submission template.
3. User manual Word Document (20 points).
   * You will write and submit a User's manual for your final project and submit it according to the instructions.
4. Documentation of source code (20 points)
   * You will fully document (comment) the corrected Python tkinter source code with appropriate comments including:
   * A brief explanation of the purpose of each module (Sub) at the beginning of each Sub. (A header's comment)
   * Explanation of the purpose of each variable where it is declared. (An end-line comment)
   * Line by line, or at least section by section comments within the code, explaining what the line/section does.
5. The link to the GitHub repository for your final project. (10 points)

**View Rubric**

# 3 (optional) UML and Use Case Diagrams



Using [Draw.IO](https://app.diagrams.net/) or [Creately](https://creately.com/?gclid=Cj0KCQiAwbitBhDIARIsABfFYIIXPQCFDSTVf_UpKVQDxs2pGiOP0KwSLm9mfW2hF-lLvsFvzo7wc58aAip6EALw_wcB)

# 4 (required) Source Code of All files (.py)

"""

Author: Corey Lambert

Date Written:7/01/2024

Assignment: Final Project

This Program is designed as an interface to hold and operate all paperwork documnetation inside of factory. It starts with password validation.

Upon entering password it opens the user profile. This them gains access to the data entry page with text boxes for entry. the inputed inforamtion is then put into

a tree table for the user to view.

"""

import tkinter as tk

from tkinter import messagebox, ttk

#initialize main app window and the frames

class LYBOperationBrainApp:

    def \_\_init\_\_(self, root):

        self.root = root

        self.root.title("LYB Operation Brain")

        self.root.geometry("500x500")

    #declare the paswords used for validation

        self.user\_passwords = ["user1", "user2"]

        self.entry\_widgets = {}

        self.current\_frame = None

    # declare the varables for use in time and box numbers

        self.times = [f"{hour if hour != 0 else 12}:00 AM" for hour in range(12)] + \

                     [f"{hour - 12 if hour > 12 else 12}:00 PM" for hour in range(12, 24)]

        self.box\_numbers = [str(i) for i in range(1, 101)]

            # create and declare the  app frames

        self.create\_main\_frame()

        self.create\_machine\_info\_frame()

        self.create\_main\_app\_frame()

        self.switch\_frame(self.main\_frame)

            # this creates frame for the password entry

    def create\_main\_frame(self):

        self.main\_frame = tk.Frame(self.root)

        self.main\_frame.pack(fill="both", expand=True)

            # prompt and label password entry

        password\_label = tk.Label(self.main\_frame, text="Enter Password:")

        password\_label.pack(pady=10)

        self.password\_entry = tk.Entry(self.main\_frame, show="\*")

        self.password\_entry.pack(pady=10)

            # label buttons on frame

        validate\_button = tk.Button(self.main\_frame, text="Validate", command=self.password\_access)

        validate\_button.pack(pady=15)

        exit\_button\_main = tk.Button(self.main\_frame, text="Exit", command=self.root.quit)

        exit\_button\_main.pack(pady=10)

#add the image to this frame and locate it

        password\_image = tk.PhotoImage(file="C:/Users/CTSL0/OneDrive/Documents/SDEV140/LYB\_logo.png")

        password\_image\_label = tk.Label(self.main\_frame, image=password\_image)

        password\_image\_label.pack(pady=10)

        self.main\_frame.password\_image = password\_image  # Keep a reference to prevent garbage collection

        # create frame used for machine information and date

    def create\_machine\_info\_frame(self):

        print("Switching to machine info page")

        self.machine\_info\_frame = tk.Frame(self.root)

            # machine number text box

        machine\_number\_label = tk.Label(self.machine\_info\_frame, text="Enter Machine Number:")

        machine\_number\_label.pack(pady=10)

        self.machine\_number\_entry = tk.Entry(self.machine\_info\_frame)

        self.machine\_number\_entry.pack(pady=10)

            # date entry box with prompt to ensure user enters in correct format

        date\_label = tk.Label(self.machine\_info\_frame, text="Enter Date: (MM/DD/YYYY)")

        date\_label.pack(pady=10)

        self.date\_entry = tk.Entry(self.machine\_info\_frame)

        self.date\_entry.pack(pady=10)

        # buttons for this frame

        submit\_button = tk.Button(self.machine\_info\_frame, text="Submit", command=self.submit\_machine\_info)

        submit\_button.pack(pady=15)

        back\_button\_machine\_info = tk.Button(self.machine\_info\_frame, text="Back", command=lambda: self.switch\_frame(self.main\_frame))

        back\_button\_machine\_info.pack(pady=10)

        exit\_button\_machine\_info = tk.Button(self.machine\_info\_frame, text="Exit", command=self.root.quit)

        exit\_button\_machine\_info.pack(pady=10)

        # create main app frame for inouts and display it

    def create\_main\_app\_frame(self):

        self.main\_app\_frame = tk.Frame(self.root)

            # this validates password and if validate moves to next frame

    def password\_access(self):

        password = self.password\_entry.get()

        if password in self.user\_passwords:

            messagebox.showinfo("Let's Work", "User Access Granted.")

            self.show\_machine\_info\_page("User")

            # shows user they need valid password

        else:

            messagebox.showerror("Error", "Invalid Password")

        # switches to machine frame

    def show\_machine\_info\_page(self, role):

        self.switch\_frame(self.machine\_info\_frame)

        self.machine\_info\_frame.role = role

        # this defines all the logic for the main app. sizes and labels

    def LYB\_OperationBrain(self, role, machine\_number, date):

        print("Switching to main app frame")

        self.switch\_frame(self.main\_app\_frame)

        self.root.geometry("1450x750") # this size allows everythign to display

        self.main\_app\_frame.title = f"{role} Version"

            # this shows its the user version

        label = tk.Label(self.main\_app\_frame, text=f"Welcome - {role} Version")

        label.grid(row=0, column=0, columnspan=3, pady=10, sticky="nsew")

        # this transfers info from previous frame as a header

        header\_label = tk.Label(self.main\_app\_frame, text=f"Machine Number: {machine\_number}, Date: {date}")

        header\_label.grid(row=1, column=0, columnspan=3, pady=10, sticky="nsew")

        # create and name data entry boxes

        self.create\_label\_entry(self.main\_app\_frame, "Time:", 2, ttk.Combobox, values=self.times)

        self.create\_label\_entry(self.main\_app\_frame, "Box Number:", 3, ttk.Combobox, values=self.box\_numbers)

        self.create\_label\_entry(self.main\_app\_frame, "Box Weight (LBS):", 4, ttk.Spinbox, from\_=0, to=2000)

        self.create\_label\_entry(self.main\_app\_frame, "Temperature (°F):", 5, ttk.Spinbox, from\_=0, to=800)

        self.create\_label\_entry(self.main\_app\_frame, "RPM:", 6, ttk.Spinbox, from\_=0, to=10000)

        self.create\_label\_entry(self.main\_app\_frame, "Rate:", 7, ttk.Spinbox, from\_=0, to=16000)

        self.create\_label\_entry(self.main\_app\_frame, "Run Comment:", 8, tk.Text, height=4, width=30)

            # this puts the image on screen and locates it to the right of entry boxes

        tree\_image = tk.PhotoImage(file="C:/Users/CTSL0/OneDrive/Documents/SDEV140/Extruder.png")

        tree\_image\_label = tk.Label(self.main\_app\_frame, image=tree\_image)

        tree\_image\_label.grid(row=2, column=3, rowspan=7, padx=10, pady=5, sticky="nsew")

        self.main\_app\_frame.tree\_image = tree\_image  # Keep a reference to prevent garbage collection

        # this is the submit button

        submit\_button = tk.Button(self.main\_app\_frame, text="Submit", command=self.submit\_all\_data)

        submit\_button.grid(row=9, column=1, pady=10, sticky="nsew")

        # this transposes the inputed data into a table

        tree = ttk.Treeview(self.main\_app\_frame, columns=("Time", "Box Number", "Box Weight", "Temperature", "RPM", "Rate", "Run Comment"), show="headings")

        for col in tree["columns"]:

            tree.heading(col, text=col)

            tree.column(col, width=150)

        tree.grid(row=10, column=0, columnspan=3, pady=10, sticky="nsew")

        self.treeview = tree

        self.current\_machine\_number = machine\_number

        self.current\_date = date

        # back and exit buttons

        back\_button = tk.Button(self.main\_app\_frame, text="Back", command=lambda: self.switch\_frame(self.machine\_info\_frame))

        back\_button.grid(row=11, column=0, pady=10, sticky="nsew")

        exit\_button = tk.Button(self.main\_app\_frame, text="Exit", command=self.root.quit)

        exit\_button.grid(row=11, column=2, pady=10, sticky="nsew")

        # grid weights changes column size with window size

        for i in range(13):

            self.main\_app\_frame.grid\_rowconfigure(i, weight=1)

        for i in range(4):

            self.main\_app\_frame.grid\_columnconfigure(i, weight=1)

        # this creates label and widgets for entry

    def create\_label\_entry(self, parent, text, row, widget\_class, \*\*options):

        label = ttk.Label(parent, text=text)

        label.grid(row=row, column=0, padx=5, pady=5, sticky=tk.W)

         # allow for spinbox inputs and typed inputs

        if widget\_class == ttk.Spinbox:

            validate\_cmd = parent.register(self.validate\_spinbox\_input)

            options.update({'validate': 'key', 'validatecommand': (validate\_cmd, '%P')})

        entry = widget\_class(parent, \*\*options)

        entry.grid(row=row, column=1, padx=5, pady=5, sticky=tk.W)

        key = text.replace(" ", "").replace("(", "").replace(")", "").replace(":", "").lower()

        self.entry\_widgets[key] = entry

        print(f"Added entry widget for key: {key}")

    def validate\_spinbox\_input(self, value\_if\_allowed):

        return value\_if\_allowed.isdigit() or value\_if\_allowed == ""

        # transfers data from input fields to the tree table upon submiting

    def submit\_all\_data(self):

        try:

            print(f"Current keys in entry\_widgets: {self.entry\_widgets.keys()}")

            time = self.entry\_widgets["time"].get()

            box\_number = self.entry\_widgets["boxnumber"].get()

            box\_weight = self.entry\_widgets["boxweightlbs"].get()

            temperature = self.entry\_widgets["temperature°f"].get()

            rpm = self.entry\_widgets["rpm"].get()

            rate = self.entry\_widgets["rate"].get()

            run\_comment = self.entry\_widgets["runcomment"].get("1.0", tk.END).strip()

            # checks to make sure all fields have data , no empty text boxes

            if all([time, box\_number, box\_weight, temperature, rpm, rate, run\_comment]):

                self.treeview.insert("", "end", values=(time, box\_number, box\_weight, temperature, rpm, rate, run\_comment))

                messagebox.showinfo("Data Entered", "All data has been entered successfully.")

                self.clear\_entries()

            else:

                messagebox.showwarning("Input Error", "Please fill in all fields.")

        except KeyError as e:

            messagebox.showerror("Error", f"KeyError: {e}")

        # clears text boxes for new inputs after submited

    def clear\_entries(self):

        for key, entry in self.entry\_widgets.items():

            if isinstance(entry, tk.Text):

                entry.delete("1.0", tk.END)

            else:

                entry.delete(0, tk.END)

        # defines the ability to switch frame to frame

    def switch\_frame(self, new\_frame):

        if self.current\_frame is not None:

            self.current\_frame.pack\_forget()

        self.current\_frame = new\_frame

        self.current\_frame.pack(fill="both", expand=True)

        # defines entry  of date and machine number ensures both are entered

    def submit\_machine\_info(self):

        machine\_number = self.machine\_number\_entry.get()

        date = self.date\_entry.get()

        if machine\_number and date:

            self.LYB\_OperationBrain(self.machine\_info\_frame.role, machine\_number, date)

        else:

            messagebox.showwarning("Input Error", "Please enter both machine number and date.")

    # loop logic

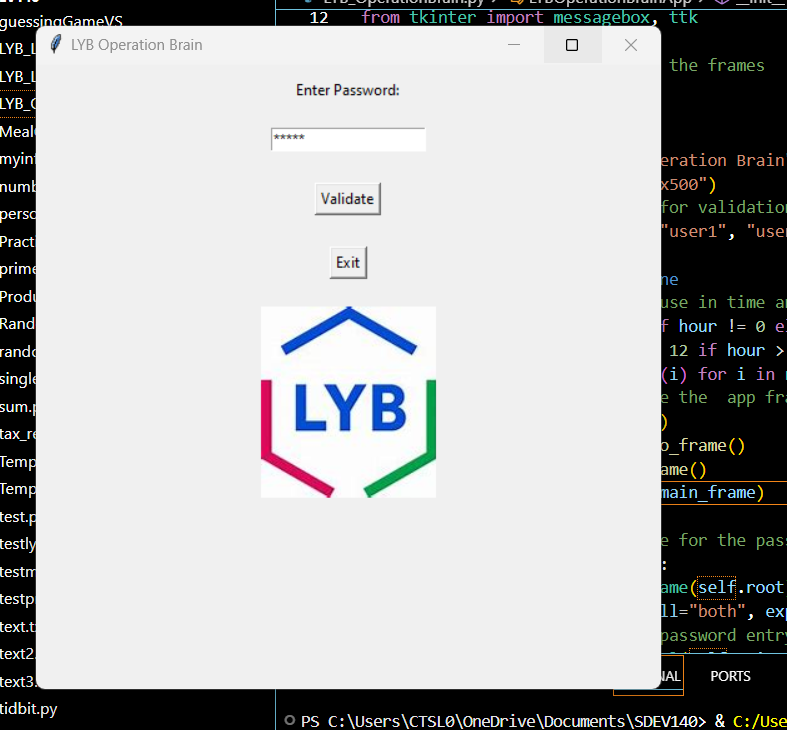
if \_\_name\_\_ == "\_\_main\_\_":

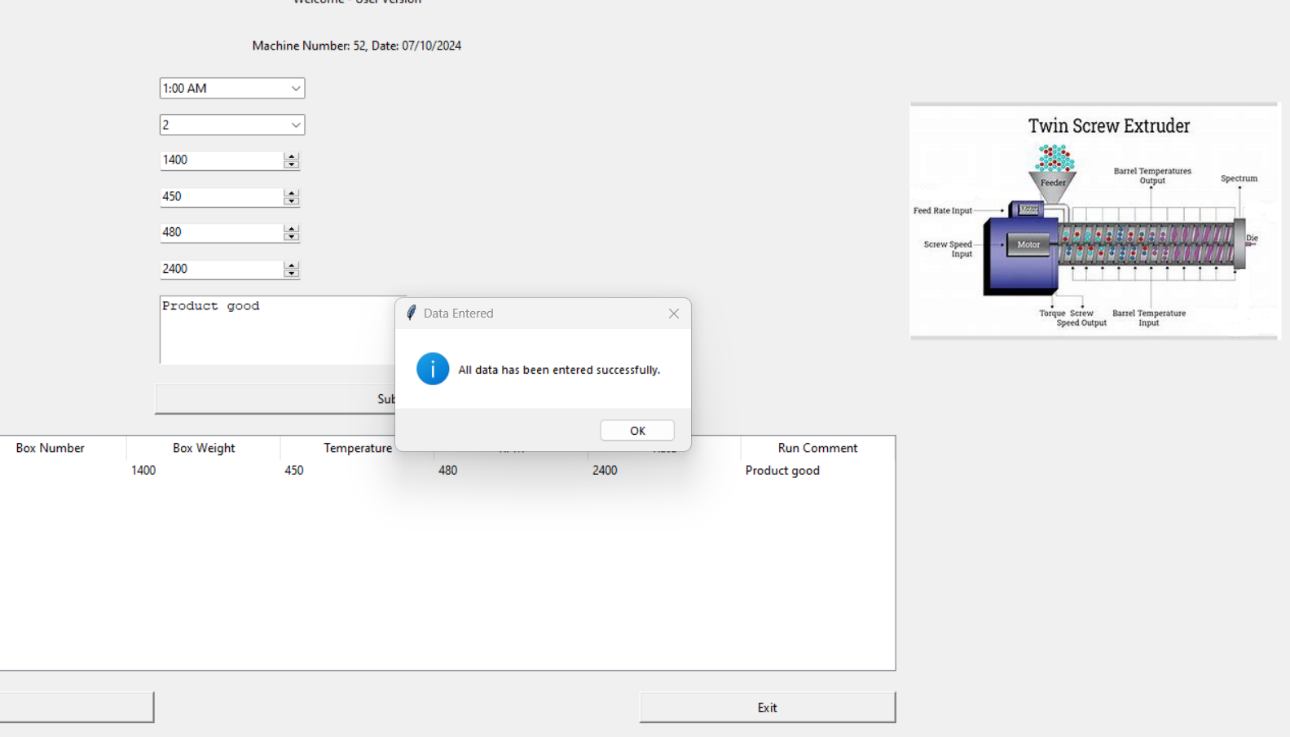
    root = tk.Tk()

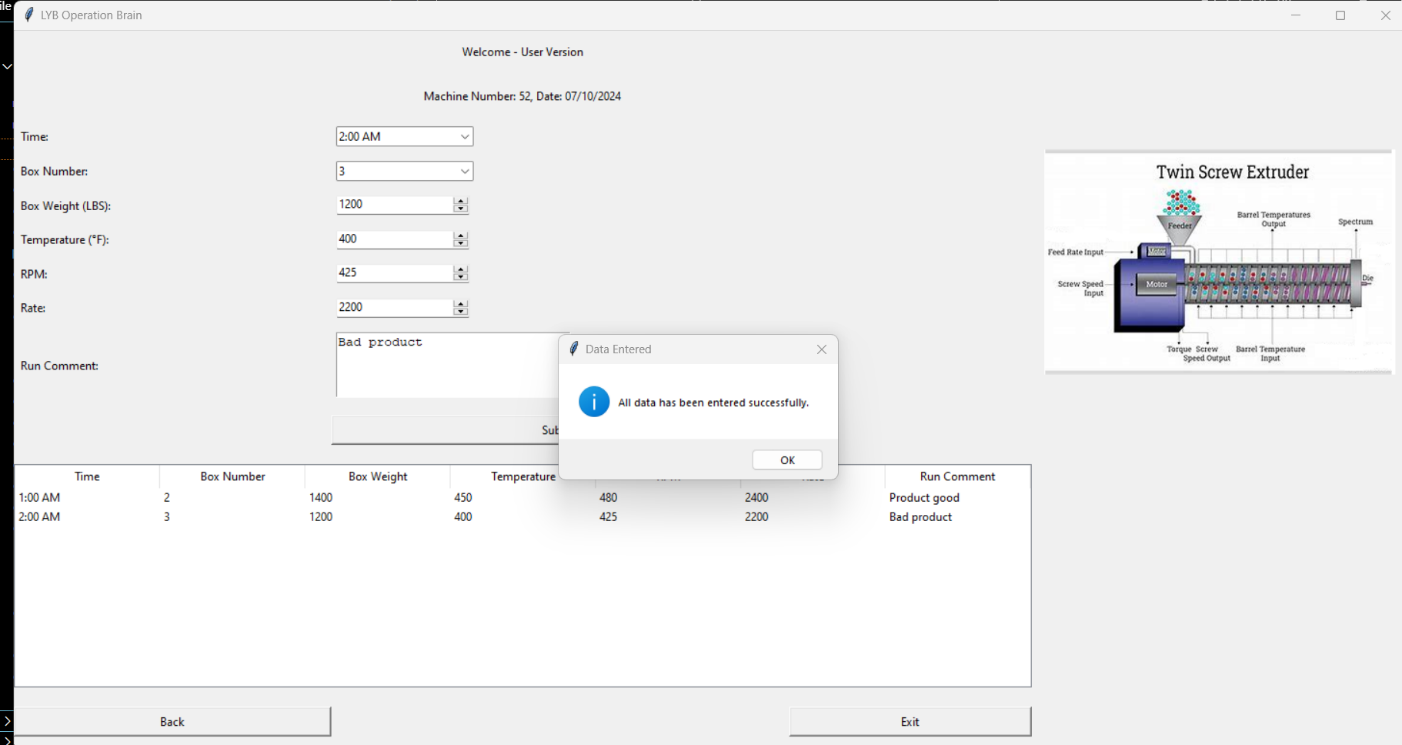
    app = LYBOperationBrainApp(root)

    root.mainloop()

# 5 (required) Three Use Case Screen Shots







# 6 (Optional) GitHub URL

[CsTL0522/SDEV140Final: LYB Brain APP (github.com)](https://github.com/CsTL0522/SDEV140Final)