Project Requirements

Product requirements

Target release	06 Mar 2023	
Document status	FINISHED	
Document owner	Maruf Sourav	
Designer	Jennifer Sager, Maruf Sourav	
Tech lead	Matthew Krol	
Technical writers	Joe Heaphy, Jaheda Lima	
Code Compliance	Jaheda Lima	

Objective

This project seeks to create a program for data storage to be used by an organization to modify and store files containing user and order data.

Milestones

Requirements

Requirement	User Story	Importance
Must store, collect, and search data sets collected from .csv files	User wants to store data from a file using the application	HIGH
A graphical UI must be included	User needs to be able to interact with the program easily	HIGH
Storage system must be robust	User wants to be able to search the data by categories	HIGH
Application to be packaged and transmitted as a .exe using pyinstaller	User should be able to use the program outside of the coding application	HIGH
Storage solutions to include lists, strings and dictionaries	Better organization for the user to search through	MEDIUM
Fields such as SSN, email, order numbers, product ID numbers, and phone numbers must be in proper format	Data should be readable for the user to know what theyre looking at	MEDIUM
Code can't be executed if imported	Maintain integrity of the program	MEDIUM
Comments are appropriate and explanatory; contain necessary information	Useful for other programmers examining the code	LOW
Coding must user doc strings/pydoc throughout program	Allows coders to find aspects of the code easier	LOW

Not Doing

Things we didn't get around to:

- Success messages for various button functions
- Save feature for program

Source code

```
76  def import_data():
77      """This fucntion imports data"""
78      #Destroy Entry and Label that were existance before click
79      destroy_children(store_frame)
80      destroy_children(search_frame)
81      destroy_children(import_frame)
82      destroy_children(delete_frame)
83      import_frame.pack()
84      #Create label and entry for file name
85      Label(import_frame, text="Enter file name", background="red").pack()
86      file_entry=Entry(import_frame, textvariable=file_var)
87      file_entry.focus_set()
88      file_entry.pack()
89      #create button to submit file name
90      Button(import_frame, text = 'Submit file', command =lambda: [open_file(),
91          destroy_children(import_frame)]).pack(pady=10)
92      #set file name back to nothing
93      file_var.set("")
```

In this code snippet, the destroy_children() function is used to destroy all entries and labels that are on the screen in a particular frame, so that if you click the add data button and do not finish it, it will remove those entries and labels.

```
121 def delete_data():
122     """This fucntion will delete the data"""
123     destroy_children(store_frame)
124     destroy_children(search_frame)
125     destroy_children(import_frame)
126     destroy_children(delete_frame)
127 
128     delete_frame.pack()
129     #Creating Label and Entry for deleting
130     Label(delete_frame, text="Delete this row from the list by SSN", background="red").pack()
131     delete_entry=Entry(delete_frame, textvariable=delete_var)
132     delete_entry.focus_set()
133     delete_entry.pack()
134 
135     #Create button to delete ssn
136     Button(delete_frame,text="Delete", command=lambda:[delete_snn(),
137     destroy_children(delete_frame)]).pack(pady=5)
138 
139 def show_data():
140     """This will show all the data"""
141     if data:
142         #Create window
143         show_window = Toplevel(root)
144         show_window.title("Showing all data")
145         show_window.geometry("800x800")
146         Label(show_window, text="This is all the employees").pack()
147         #Create Scroolbar
```

```
148         scroll_bar = Scrollbar(show_window)
149         scroll_bar.pack(side=RIGHT, fill=Y)
150         #Create listbox for scrolling
151         employee_list = Listbox(show_window,width=800,height=100, yscrollcommand=scroll_bar.set)
152         employee_list.pack(side=LEFT, fill=BOTH, expand=True, pady=25)
153         for employee in data:
154             employee_list.insert(END, employee)
155         Button(show_window, text="Copy", command=pyperclip.copy(str(data))).place(x=250, y=775)
156         Button(show_window, text="Quit", command=show_window.destroy).place(x=750, y=775)
157         scroll_bar.config(command=employee_list.yview)
158 
159 def delete_snn():
160     """This funcion will delete the row from the table"""
161     ssn_delete = delete_var.get()
162     #Checking ssn to make sure it is only numbers
163     if not check_ssn(ssn_delete):
164         return
165     flag=True
166     #check for ssn in data
167     for ssn in data:
168         if ssn_delete == ssn[2]:
169             temp_ssn = ssn
170             data.remove(ssn)
171             flag=False
172     if flag:
173         #Print window with not found
174         file_window = Toplevel(root)
175         file_window.title("SSN not found")
176         file_window.geometry("400x100")
177         Label(file_window, text="That SSN was not found").pack()
```

The functions delete_data() and delete_ssn() are interconnected here, where the function for delete data creates a label and a button for deleting the data and attaches the delete_ssn() function to the button defined in delete_data().

```
def check_phone(phone):
    """Check if phone is correct format"""
    if re.match(r"\(\d{3}\)\d{3}-\d{4}",phone):
        return True
    phone_window = Toplevel(root)
    phone_window.title("phone number error")
    phone_window.geometry("400x50")
    Label(phone_window, text="The phone number was not the correct format\n\
            Try in the format of (888)555-4545").pack()
    return False

def check_email(email):
    """Check if email is correct"""
    if re.match(r"(^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$)",email):
        return True
    email_window = Toplevel(root)
    email_window.title("email error")
    email_window.geometry("400x50")
    Label(email_window, text="The email was not the correct format\n\
            Please try again").pack()
    return False

def check_position(position):
    """Checking if position is correct"""
    position_check_list = ['Helper', 'Manager', 'Assistant Manager', 'Staff', 'Employee']
    for position_check in position_check_list:
        if position_check == position:
            return True
    position_window = Toplevel(root)
    position_window.title("email error")
    position_window.geometry("450x50")
    Label(position_window, text="The position was not the correct\n\
            Please try again (Manager, Assistant Manager, Helper, Staff, Employee)").pack()
    return False
```

re in python is for regular expression operations, both check_phone and check_email use re.match to make sure input is in the right format for a phone number and email address, respectively. re.match() will check if zero or more characters at the beginning of the string match the regular expression pattern specified, and return a corresponding match object, or return none if the string does not match the pattern.

Testing

This project will include testing to ensure the program is functioning as intended, which will be done using multiple methods including software to check the code and manual test cases.

Test	Method
Code to be checked with Pylint or equivalent	run Pylint on the code, use suggestions to improve
User input to be filtered for special characters	Enter input containing special characters to program
Test add_data() and import_data() functions	attempt to enter data to store with file input
Test search_data() function	attempt to use all search buttons with different input for respective categories

Test delete_data() and delete_ssn() functions	attempt to use delete button with different SSN choices
Test show_data() function	attempt to use show button
Test search functions	enter various names, emails, etc with each search function using the search button, use copy and quit buttons after
Test check functions	enter data with incorrect formatting
test open_file() function	input a file to the program and observe results
test store_data_entry() function	test submit all and store buttons after entering data

End-user documentation

To start using the program: launch the .exe file, the GUI should be displayed with various buttons to get started.

from the main page, first use the Import button and enter the name of the file you want to open, then press the Submit file button. This will import your file to allow you to make use of the other functions of the program.

Next, you can use the Search button to enter the searching functionality, then select the button corresponding to which data element you'd like to search by, and enter the specified element. If your search query is found, you can then use the copy button to save that search, or the quit button to stop the program.

If you would like to delete an element from the data set, you can use the delete button, then specify the element you'd like to delete by its SSN.

The show button will display the full data set entered with the imported file.

Once finished with the program, the Quit button will exit the program.

References

https://www.altexsoft.com/blog/business/technical-documentation-in-software-development-types-best-practices-and-tools/

https://calculator.aws/#/addService/ec2-enhancement

Hosting Quotes

AWS EC2	IONOS	HostGator	BlueHost	DreamHost
Compute savings plans - 614.95	Grow plan - 1/mo for 12 months then 8/mo	Hatchling plan - 2.75/mo	Basic plan - 2.95/mo	Shared plan - 2.95/mo
Instance savings plans - 515.09	Start plan - 2/mo for 12 months then 4/mo	Baby plan - 3.50/mo	Plus plan - 5.45/mo	DreamPress - 16.95/mo
On-Demand - 37.96/mo	Boost plan - 6/mo for 12 months then 12/mo	Business plan - 5.25/mo	Online Store - 9.95/mo	VPS - 13.75/mo

Future Recommendations

- Add Success messages
- Add save feature to overall program
- Implement further data manipulation options

Why choose us?

Our group's program not only works exactly as intended with full functionality but also allows for searching multiple items at once, allows the user to copy the search results, can switch between functions without issue and ensures user input is in the correct format.

further, we have clear and concise documentation with extra features from Atlassian confluence for better readability.