INFORMATICS INSTITUTE OF TECHNOLOGY

UNIVERSITY OF WESTMINSTER

SOFTWARE DEVELOPMENT I
4COSC006C

Name = U.H.CHANUPA CHINTHAL

IIT Number = 20232284

UOW Number = W2083170

Table Of Content

Introduction

Research Data Management SystemA well-crafted tool to support you in your journey collecting, analysing and storing of your experimental data. It is designed to address the needs of investigators who need a more organized way to manage large amounts of data generated from scientific experiments. This system provides an easy-to-use interface with which researchers can enter data, perform statistical calculations and store work in a secure manner for future reference. This makes the system faster, more accurate and less time gets consumed which would otherwise be spent in manual calculations.

Fundamentally, the system provides minimal features with data entry and file management as well as statistical analysis. This allows users to quickly input new data sets, complete with a name of the test performed along with the calendar date it was carried out and even who run those tests including not only metric information. Being able to calculate statistical quantities (mean, median or standard deviation of a parliament shadow) gives instant access into underlying behaviors in the data and can provide researchers understanding needed for empiric decision making. The system also allows data to be saved indefinitely in a persistent storage that enables the retrieval of all experimental records at any time.

It lends itself to a wide range of research domains, thus rendering it a versatile tool for academic and professional researchers. Thanks to the integration with a Python GUI developed using Tkinter, it can be used comfortably even by people who are not too tech savvies. This system provides a complete data management solution that is not only flexible enough to respond to the requirements of sound scientific inquiry, but also promotes an orderly approach for managing research in such as way that it can be adapted and improved upon over time.

Code Structure

Part A

```python
import os
import statistics

def add_entry(entries):
    experiment_name = input("Enter experiment name: ")
    date = input("Enter date (YYYY-MM-DD): ")
    researcher = input("Enter researcher name: ")
    data_points = list(map(float, input("Enter data points separated by
commas: ").split(',')))
    entries.append((experiment_name, date, researcher, data_points))

def view_entries(entries):
    for entry in entries:
        print(f"Experiment: {entry[0]}, Date: {entry[1]}, Researcher:
{entry[2]}, Data Points: {entry[3]}")

def save_entries_to_file(entries, filename):
    with open(filename, 'w') as f:
        for entry in entries:
            f.write(f"{entry[0]},{entry[1]},{entry[2]},{','.join(map(str,
entry[3]))}\n")

def load_entries_from_file(filename):
    entries = []
    if os.path.exists(filename):
        with open(filename, 'r') as f:
            for line in f:
                parts = line.strip().split(',')
                experiment_name = parts[0]
                date = parts[1]
                researcher = parts[2]
                data_points = list(map(float, parts[3:]))
                entries.append((experiment_name, date, researcher,
data_points))
    return entries
```

```python
def analyze_data(entries):
    for entry in entries:
        data_points = entry[3]
        print(f"Experiment: {entry[0]}")
        print(f"Average: {statistics.mean(data_points)}")
        print(f"Standard Deviation: {statistics.stdev(data_points) if 
len(data_points) > 1 else 'N/A'}")
        print(f"Median: {statistics.median(data_points)}")

def main():
    filename = "research_data.txt"
    entries = load_entries_from_file(filename)

    while True:
        print("\nMenu:")
        print("1. Add a research data entry")
        print("2. View all entries")
        print("3. Analyze data")
        print("4. Save entries to file")
        print("5. Exit")

        choice = input("Enter your choice: ")
        if choice == '1':
            add_entry(entries)
        elif choice == '2':
            view_entries(entries)
        elif choice == '3':
            analyze_data(entries)
        elif choice == '4':
            save_entries_to_file(entries, filename)
        elif choice == '5':
            break
        else:
            print("Invalid choice, please try again.")

if __name__ == "__main__":
    main()
```

Part B

```
import os
import statistics

class ResearchDataManager:
    def __init__(self):
        self.entries = []
        self.filename = "research_data.txt"

    def add_entry(self):
        experiment_name = input("Enter experiment name: ")
        date = input("Enter date (YYYY-MM-DD): ")
        researcher = input("Enter researcher name: ")
        data_points = list(map(float, input("Enter data points separated
by commas: ").split(',')))
        self.entries.append((experiment_name, date, researcher,
data_points))

    def view_entries(self):
        for entry in self.entries:
            print(f"Experiment: {entry[0]}, Date: {entry[1]}, Researcher:
{entry[2]}, Data Points: {entry[3]}")

    def save_entries_to_file(self):
        with open(self.filename, 'w') as f:
            for entry in self.entries:

f.write(f"{entry[0]},{entry[1]},{entry[2]},{','.join(map(str,
entry[3]))}\n")

    def load_entries_from_file(self):
        if os.path.exists(self.filename):
            with open(self.filename, 'r') as f:
                self.entries = []
                for line in f:
                    parts = line.strip().split(',')
                    experiment_name = parts[0]
                    date = parts[1]
                    researcher = parts[2]
                    data_points = list(map(float, parts[3:]))
                    self.entries.append((experiment_name, date, 
researcher, data_points))

    def analyze_data(self):
        for entry in self.entries:
            data_points = entry[3]
            print(f"Experiment: {entry[0]}")
            print(f"Average: {statistics.mean(data_points)}")
            print(f"Standard Deviation: {statistics.stdev(data_points) if 
len(data_points) > 1 else 'N/A'}")
            print(f"Median: {statistics.median(data_points)}")

def main():
    manager = ResearchDataManager()
    manager.load_entries_from_file()

    while True:
        print("\nMenu:")
        print("1. Add a research data entry")
        print("2. View all entries")
        print("3. Analyze data")
        print("4. Save entries to file")
        print("5. Exit")

        choice = input("Enter your choice: ")
        if choice == '1':
            manager.add_entry()
        elif choice == '2':
            manager.view_entries()
        elif choice == '3':
            manager.analyze_data()
        elif choice == '4':
            manager.save_entries_to_file()
        elif choice == '5':
            break
        else:
            print("Invalid choice, please try again.")
```

```python
if __name__ == "__main__":
    main()
```

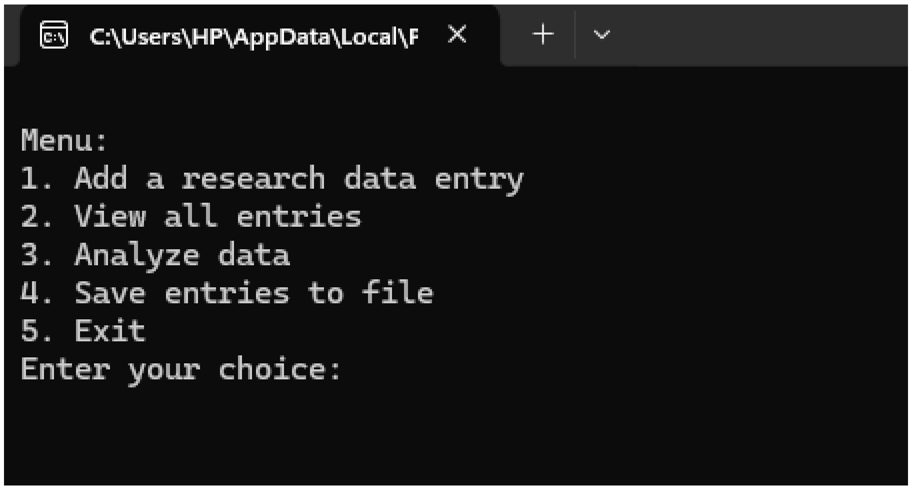
Part C,Part D

```python
import os
import tkinter as tk
from tkinter import messagebox
import statistics

class ResearchDataManager:
    def __init__(self):
        self.entries = []
        self.filename = "research_data.txt"

    def add_entry(self, experiment_name, date, researcher, data_points):
        entry = {
            "experiment_name": experiment_name,
            "date": date,
            "researcher": researcher,
            "data_points": data_points
        }
        self.entries.append(entry)

    def save_entries_to_file(self):
        with open(self.filename, 'w') as file:
            for entry in self.entries:
                data_points_str = ",".join(map(str, entry["data_points"]))

file.write(f"{entry['experiment_name']},{entry['date']},{entry['researcher
']},{data_points_str}\n")
```

```python
def load_entries_from_file(self):
    if not os.path.exists(self.filename):
        return
    self.entries = []
    with open(self.filename, 'r') as file:
        for line in file:
            parts = line.strip().split(",")
            experiment_name, date, researcher = parts[:3]
            try:
                data_points = list(map(float, parts[3:]))
            except ValueError:
                continue  # Skip lines with invalid data points
            entry = {
                "experiment_name": experiment_name,
                "date": date,
                "researcher": researcher,
                "data_points": data_points
            }
            self.entries.append(entry)

def analysis_data(self):
    if not self.entries:
        return []
    analysis_results = []
    for entry in self.entries:
        data_points = entry["data_points"]
        average = statistics.mean(data_points)
        if len(data_points) > 1:
            standard_deviation = statistics.stdev(data_points)
        else:
            standard_deviation = 0
        median = statistics.median(data_points)
        analysis_results.append({
            "experiment": entry["experiment_name"],
            "average": average,
            "standard_deviation": standard_deviation,
            "median": median
        })
    return analysis_results
```

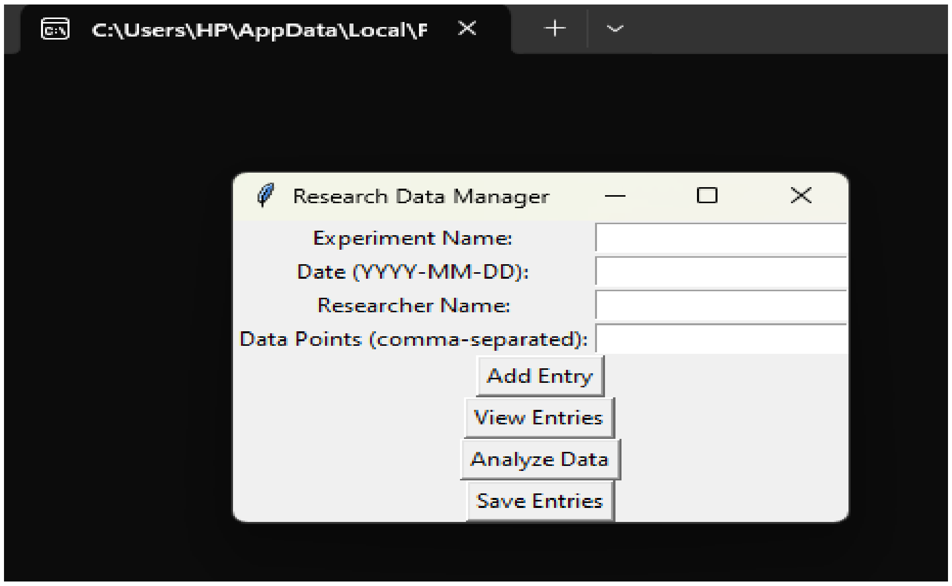
```
class ResearchDataGUI:
    def __init__(self, root, manager):
        self.manager = manager
        self.root = root
        self.root.title("Research Data Manager")

        self.experiment_label = tk.Label(root, text="Experiment Name:")
        self.experiment_label.grid(row=0, column=0)
        self.experiment_entry = tk.Entry(root)
        self.experiment_entry.grid(row=0, column=1)

        self.date_label = tk.Label(root, text="Date (YYYY-MM-DD):")
        self.date_label.grid(row=1, column=0)
        self.date_entry = tk.Entry(root)
        self.date_entry.grid(row=1, column=1)

        self.researcher_label = tk.Label(root, text="Researcher Name:")
        self.researcher_label.grid(row=2, column=0)
        self.researcher_entry = tk.Entry(root)
        self.researcher_entry.grid(row=2, column=1)

        self.data_point_label = tk.Label(root, text="Data Points
(comma-separated):")
        self.data_point_label.grid(row=3, column=0)
        self.data_point_entry = tk.Entry(root)
        self.data_point_entry.grid(row=3, column=1)

        self.add_button = tk.Button(root, text="Add Entry",
command=self.add_entry)
        self.add_button.grid(row=4, column=0, columnspan=2)

        self.view_button = tk.Button(root, text="View Entries",
command=self.view_entries)
        self.view_button.grid(row=5, column=0, columnspan=2)

        self.analysis_button = tk.Button(root, text="Analyze Data",
command=self.analysis_data)
        self.analysis_button.grid(row=6, column=0, columnspan=2)
        self.save_button = tk.Button(root, text="Save Entries",
command=self.save_entries)
        self.save_button.grid(row=7, column=0, columnspan=2)

    def add_entry(self):
        experiment = self.experiment_entry.get()
        date = self.date_entry.get()
        researcher = self.researcher_entry.get()
        try:
            data_points = [float(point.strip()) for point in
self.data_point_entry.get().split(",")]
        except ValueError:
            messagebox.showerror("Error", "Invalid data points. Please
enter numeric values.")
            return
        self.manager.add_entry(experiment, date, researcher, data_points)
        messagebox.showinfo("Info", "Entry added successfully!")

    def view_entries(self):
        entries = self.manager.entries
        if not entries:
            messagebox.showinfo("Info", "No entries to display.")
            return
        view_window = tk.Toplevel(self.root)
        view_window.title("View Entries")
        text = tk.Text(view_window)
        text.pack(expand=True, fill=tk.BOTH)
        for idx, entry in enumerate(entries, start=1):
            text.insert(tk.END, f"Entry {idx}:\n")
            text.insert(tk.END, f"  Experiment Name:
{entry['experiment_name']}\n")
            text.insert(tk.END, f"  Date: {entry['date']}\n")
            text.insert(tk.END, f"  Researcher: {entry['researcher']}\n")
            text.insert(tk.END, f"  Data Points:
{entry['data_points']}\n\n")

    def analysis_data(self):
        results = self.manager.analysis_data()
        if not results:
            messagebox.showinfo("Info", "No data available for analysis.")
            return
        analyze_window = tk.Toplevel(self.root)
        analyze_window.title("Analyze Data")
        text = tk.Text(analyze_window)
        text.pack(expand=True, fill=tk.BOTH)
        for result in results:
            text.insert(tk.END, f"Experiment: {result['experiment']}\n")
            text.insert(tk.END, f"  Average: {result['average']:.2f}\n")
            text.insert(tk.END, f"  Standard Deviation: 
{result['standard_deviation']:.2f}\n")
            text.insert(tk.END, f"  Median: {result['median']:.2f}\n\n")

    def save_entries(self):
        self.manager.save_entries_to_file()
        messagebox.showinfo("Info", "Entries saved successfully!")

def main():
    manager = ResearchDataManager()
    manager.load_entries_from_file()
    root = tk.Tk()
    gui = ResearchDataGUI(root, manager)
    root.mainloop()

if __name__ == "__main__":
    main()
```

User Interface (Part A,Part B)



```
Menu:
1. Add a research data entry
2. View all entries
3. Analyze data
4. Save entries to file
5. Exit
Enter your choice:
```

User Interface (Part C , Part D)



Experiment Name:	
Date (YYYY-MM-DD):	
Researcher Name:	
Data Points (comma-separated):	

Design Choices and Code Structure

The system was programmed using Python with the Tkinter library to build a wireframe for Graphic User Interface (GUI). It is simple and part of the Python standard library but also very user-friendly, hence making Tkinter a perfect choice to develop an interface that can be used easily running on almost every computer with native support in Python.

Class Design:

Now for the core off all of this, which is The ResearchDataManager class responsible to adding/Removing data's and saving or loading from file, handling statistics… This class houses all the logic around how manage these data, making sure that a code is modular and easier to maintain.

The main control class is named ResearchDataGUI and it connects with the User. Many buttons and entry fields are provided for entering experiment details as well as data points, viewing the saved entries, and analyzing the recorded data. The design adhere to the MVC pattern, and cleanly separates core data management logic from GUI handling.

File Management:

All the data is save to a text file (research_data. We store our generated datasets in a lightweight plain text format (example/results. This was made so to reduce the effort and improve reading. All EntriesSaved with experiment name, date and contributorsList of Data Points When you load the data, system will parse this file to rebuild entries list.

Statistical Analysis:

Out of the box, you get simple statistical functions such as mean or median and standard deviation etc in your data series. These calculations are extremely important for scientists who will use their experiments to interpret results, like the ones shown here. These calculations were computed using the Python statistics module to give trustworthy/efficient analysis.

Test Case

TEST CASE	Result	Expected Outcome	Actual Outcomes	Pass / Fail
Add valid data Entry	Enter valid experiment name, date, researcher name, and numeric data points. Click "Add Entry."	Entry is added successfully.	Entry added successfully.	Pass
Add entry with empty fields	Leave one or more fields empty. Click "Add Entry."	Error message.	Error message.	Pass
View all Entries	Add multiple valid entries. Click "View Entries."	A new window opens, displaying all the added entries.	Entries displayed correctly in the new window.	Pass
Save entries to file	Add multiple valid entries. Click "Save Entries."	Entries are saved to the `research_data.txt` file successfully.	Entries saved successfully to the file	Pass

Load entried From file	Ensure `research_data.txt` contains valid entries. Load the application again.	Entries from the file are loaded into the application on startup.	Entries loaded correctly from the file.	Pass
Analyze data with multiple data points	Add an entry with multiple numeric data points. Click "Analyze Data."	Statistical analysis (average, standard deviation, median) is displayed correctly for the data points.	Analysis results displayed correctly with accurate statistics.	Pass

Key Learnings

The development of this system provided several key learnings:

Important Design : Keeping the core data management completely separated from GUI helped out in mantainability and easier debugging of code base. Another advantage was the modular approach helped us to perform isolated testing of different components.

Error Handling and Validation : Having some hard code controls for exception handling are inevitable in any user-facing application. This makes the system secure to prevent breakage on any unexpected inputs and also provides a nice user experience.

User Interface Design- It forms an interactive relation between the user and system. End-to-end technical design considering the user experience is integral to this project as it ensures the interface has a life beyond functioning, but also that we cater for the end-business.

Challenges Faced

The development of the Research Data Management System met with many challenges:

Parsing and Validating DataThe user entering data meant this was one of the early challenges. Error handling and preventing literally anything else to go wrong were the big issues here: you must make sure that in cases where your users might submit invalid data (for example non-numeric values as dataypyes/fields), this does not cause a failure.

File Handling — Another challenge i faced was managing the save and load functionality to make sure that data is retained upon restart of game. The system must be able to support multiple cases, including if the data file is empty or contains corrupted values; also ensure that all saved information uses a common format.

Complex Statistical Calculations — Writing accurate and efficient code to handle complex statistical calculations (e.g. processing large datasets) was crucial, which required a good grasp of the underlying mathematics, as well the ability to use proper algorithms.