Report On

Excel Sheet Handling

Submitted in partial fulfillment of the requirements of the Course project in Semester IV of Second Year Computer Engineering

by
Vedanti Rane (Roll No. 66)
Purva Rokade (Roll No. 67)

Mentor
Prof.Sneha Mhatre

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

(A.Y. 2023-24)

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

CERTIFICATE

This is to certify that the Course Project entitled **"Excel Sheet Handling"** is a bonafide work, by " **Vedanti Rane (Roll No. 66), Purva Rokade (Roll No. 67)** submitted to the University of Mumbai in partial fulfillment of the requirement forthe award of the degree of "Bachelor of Engineering" in Semester IV of Second Year "Computer Engineering".

_____
Prof. Sneha Mhatre
Mentor

_____
Dr Megha Trivedi
Head of Department

_____
Dr. H.V. Vankudre
Principle

ABSTRACT

In the era of big data, handling and visualizing data efficiently is of paramount importance. This project, developed in Python, aims to streamline the process of handling Excel spreadsheets and presenting data in a meaningful way.

The project leverages the power of Python libraries such as pandas for data manipulation, numpy for numerical computations, matplotlib for data visualization, and tkinter for building a user-friendly GUI. These libraries work in harmony to provide a robust and efficient solution for Excel sheet handling.

The project simplifies the process of reading, writing, and manipulating data in Excel spreadsheets. It allows users to perform complex data manipulations effortlessly within the Python environment, enhancing user experience by eliminating the need to switch between multiple platforms or software.

One of the key features of this project is its user-friendly GUI developed using tkinter. This interface allows users to interact with their data in a more intuitive and efficient manner. The project also uses matplotlib to create dynamic and interactive plots, enabling users to visualize their data and gain insights effectively.

CONTENTS : **Pg No**

PROBLEM STATEMENT

In the current digital age, data is abundant and often stored in Excel spreadsheets due to its ease of use and wide acceptance. However, handling and manipulating this data efficiently, especially for non-technical users, can be a challenging task. Existing solutions often require switching between multiple platforms or software, which can be time-consuming and inefficient.

Moreover, visualizing this data in a meaningful way for analysis and decision-making purposes is another significant challenge. Existing data visualization tools often have a steep learning curve and may not provide the flexibility to customize the visualizations as per user requirements.

Additionally, the lack of a user-friendly interface in existing solutions makes it difficult for users to interact with their data. This often leads to a decrease in productivity and an increase in the time taken to derive insights from the data.

MODULE DESCRIPTION

1. **Data Handling Module (pandas, numpy):**

   This module is responsible for reading, writing, and manipulating data in Excel spreadsheets. It uses the pandas library for data manipulation and analysis, and numpy for numerical computation. This module provides functionalities such as importing data from Excel files, performing operations on the data, and exporting the manipulated data back to Excel.

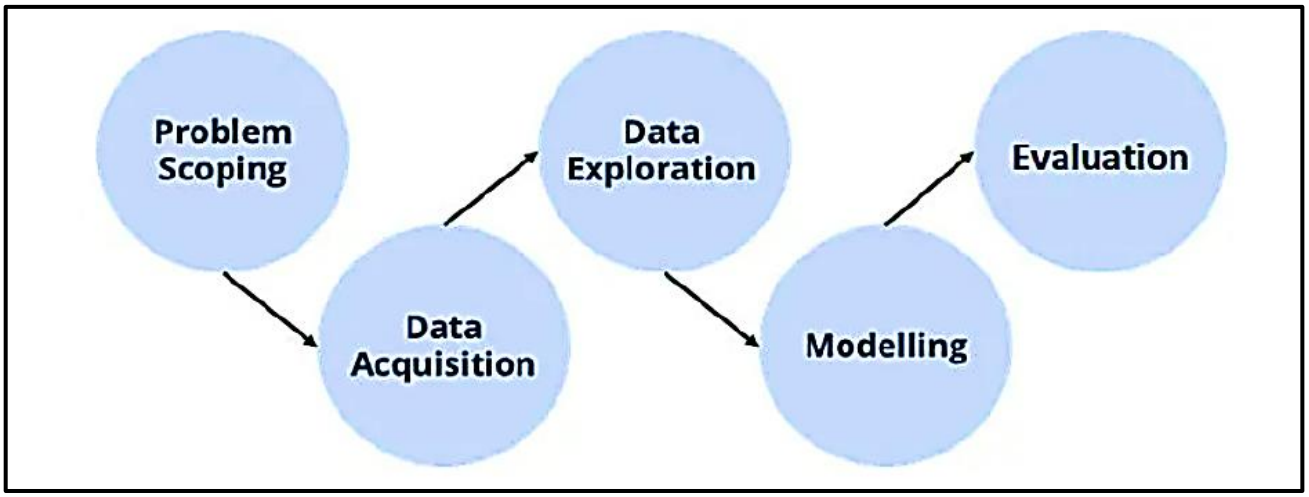
2. **Visualization Module (matplotlib):**

   This module is used for creating dynamic and interactive plots from the data. It leverages the matplotlib library to generate a variety of data visualizations such as bar charts, line graphs, histograms, scatter plots, etc. This helps in better understanding of the data and aids in decision making.

3. **User Interface Module (tkinter):**

   This module is responsible for creating a user-friendly graphical user interface (GUI) for the application. It uses the tkinter library, which is Python's standard GUI package. The GUI allows users to easily interact with the application, select Excel files, choose operations to be performed, and view the resulting visualizations.

BLOCK DIAGRAM



BRIEF DESCRIPTION OF TECHNOLOGY

This project is a Python-based solution for efficient handling and visualization of Excel data. It leverages libraries like pandas, numpy, matplotlib, and tkinter to provide a user-friendly interface for data manipulation, analysis, and interactive visualization.

1. **Pandas:**

   Pandas is a powerful Python library for data manipulation and analysis. It provides data structures and functions needed to manipulate structured data. It's particularly useful for working with numerical tables or time series data.

2. **NumPy:**

   NumPy is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, fourier transform, and matrices. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

3. **Matplotlib:**

   Matplotlib is a plotting library for Python. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. Matplotlib is also a popular library for creating static, animated, and interactive visualizations in Python.

4. **Tkinter:**

   Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit, and can create a wide range of GUI applications, ranging from simple forms to complex application layouts.

These technologies together provide a robust framework for handling, analyzing, and visualizing Excel data in an efficient and user-friendly manner

Brief description of software & hardware used and its programming

Software:

1. **Python:**
   The project is developed using Python, a high-level, interpreted programming language known for its simplicity and readability.

2. **Pandas:**
   This Python library is used for data manipulation and analysis, particularly with numerical tables or data frames.

3. **NumPy:**
   NumPy, another Python library, is used for handling large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.

4. **Matplotlib:**
   This Python library is used for creating static, animated, and interactive visualizations in Python.

5. **Tkinter:**
   Tkinter is the standard GUI library for Python, used in this project to create a user-friendly graphical user interface.

Hardware

The hardware requirements for this project are minimal and it should run on any modern computer capable of running Python. The specific requirements would depend on the size and complexity of the Excel data being processed. Larger datasets might require a computer with more RAM and a faster CPU.

CODE

```
import tkinter as tk
from tkinter import filedialog, messagebox
import openpyxl
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from tkinter import ttk

class ExcelHandlerApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Excel Handler")
        self.root.geometry("600x400")

        self.file_path = None

        self.style = ttk.Style()
        self.style.configure("TButton", padding=6, relief="flat", background="#007bff",
foreground="white")
        self.style.map("TButton", background=[("active", "#0056b3")])

        self.lbl_status = tk.Label(self.root, text="No file selected", font=("Helvetica", 10))
        self.lbl_status.pack(pady=10)

        self.btn_open = ttk.Button(self.root, text="Open Excel File", command=self.open_file)
        self.btn_open.pack(pady=5)

        self.btn_describe = ttk.Button(self.root, text="Describe Data", command=self.describe_data)
        self.btn_describe.pack(pady=5)

        self.btn_plot = ttk.Button(self.root, text="Plot Data", command=self.plot_data)
        self.btn_plot.pack(pady=5)

        self.btn_read_cell = ttk.Button(self.root, text="Read Cell", command=self.read_cell)
        self.btn_read_cell.pack(pady=5)

        self.btn_write_cell = ttk.Button(self.root, text="Write Cell", command=self.write_cell)
        self.btn_write_cell.pack(pady=5)

        self.btn_read_row = ttk.Button(self.root, text="Read Row", command=self.read_row)
        self.btn_read_row.pack(pady=5)

        self.btn_read_column = ttk.Button(self.root, text="Read Column",
command=self.read_column)
        self.btn_read_column.pack(pady=5)

        self.btn_count_rows = ttk.Button(self.root, text="Count Rows", command=self.count_rows)
        self.btn_count_rows.pack(pady=5)
```

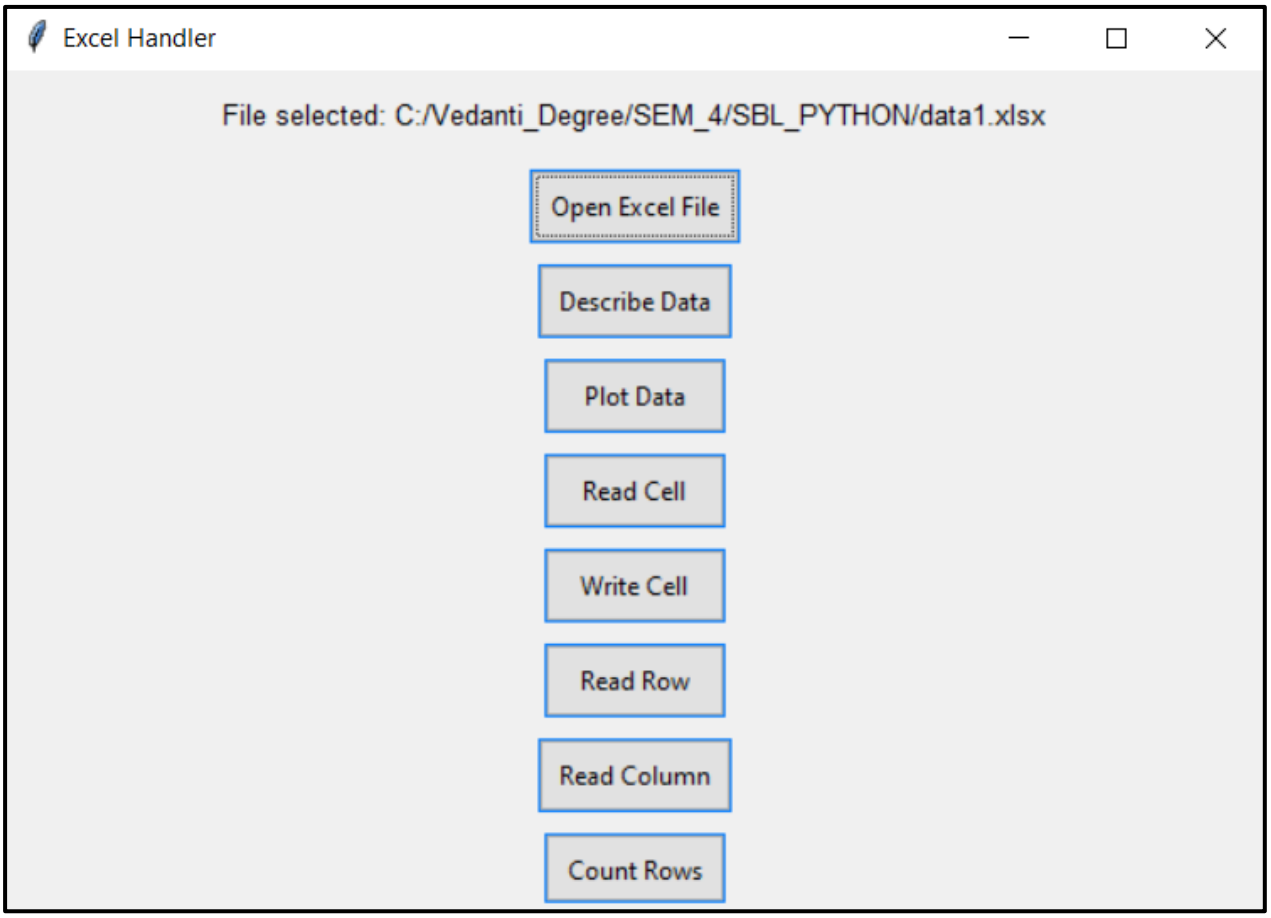
```
        self.btn_count_columns = ttk.Button(self.root, text="Count Columns",
command=self.count_columns)
        self.btn_count_columns.pack(pady=5)

        self.btn_read_table = ttk.Button(self.root, text="Read Table", command=self.read_table)
        self.btn_read_table.pack(pady=5)

        self.canvas = None

    def open_file(self):
        self.file_path = filedialog.askopenfilename(filetypes=[("Excel files", "*.xlsx")])
        if self.file_path:
            self.lbl_status.config(text=f"File selected: {self.file_path}")
        else:
            self.lbl_status.config(text="No file selected")

    def describe_data(self):
        if self.file_path:
            try:
                df = pd.read_excel(self.file_path)
                description = df.describe()
                messagebox.showinfo("Data Description", str(description))
            except Exception as e:
                messagebox.showerror("Error", f"An error occurred: {e}")
        else:
            messagebox.showwarning("Warning", "No file selected")

    def plot_data(self):
        if self.file_path:
            try:
                df = pd.read_excel(self.file_path)
                if self.canvas:
                    self.canvas.get_tk_widget().pack_forget()

                fig = plt.Figure(figsize=(6, 4), dpi=100)
                ax = fig.add_subplot(111)
                ax.plot(df.index, df.values)
                ax.set_xlabel("Index")
                ax.set_ylabel("Values")
                ax.set_title("Plot Data")

                self.canvas = FigureCanvasTkAgg(fig, master=self.root)
                self.canvas.draw()
                self.canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=1)
            except Exception as e:
                messagebox.showerror("Error", f"An error occurred: {e}")
        else:
            messagebox.showwarning("Warning", "No file selected")

    def read_cell(self):
        if self.file_path:
            try:
```

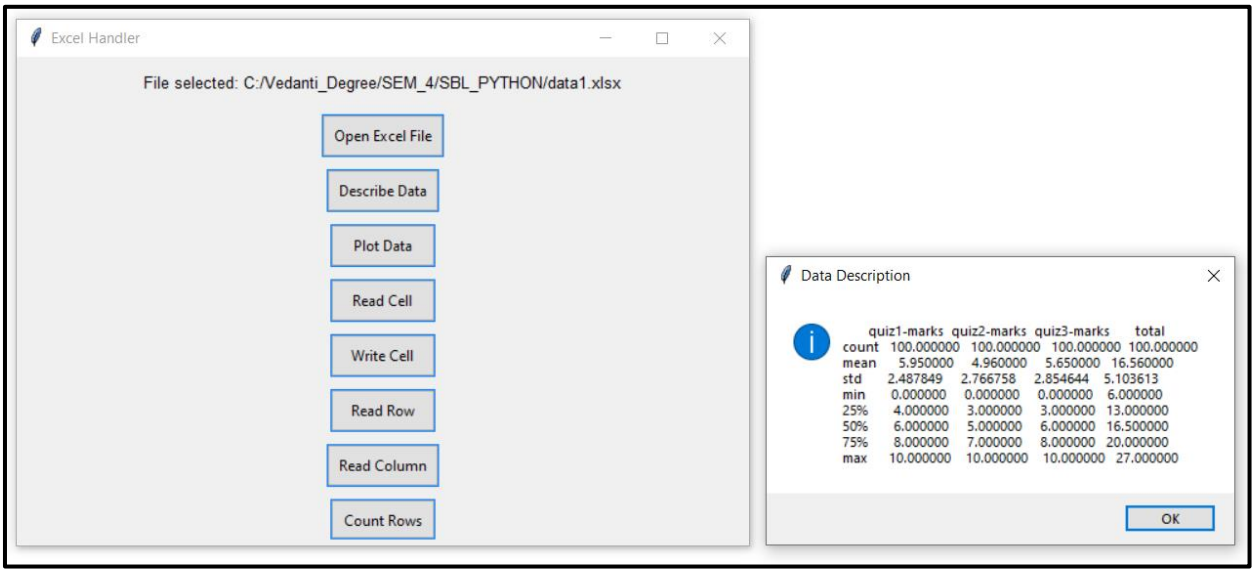
```
                workbook = openpyxl.load_workbook(self.file_path)
                worksheet = workbook.active
                cell_value = worksheet['A1'].value
                messagebox.showinfo("Cell Value", f"The value of cell A1 is: {cell_value}")
            except Exception as e:
                messagebox.showerror("Error", f"An error occurred: {e}")
        else:
            messagebox.showwarning("Warning", "No file selected")

    def write_cell(self):
        if self.file_path:
            try:
                workbook = openpyxl.load_workbook(self.file_path)
                worksheet = workbook.active
                worksheet['A2'] = 'xyz'
                workbook.save(self.file_path)
                messagebox.showinfo("Success", "Data written to cell A2 successfully.")
            except Exception as e:
                messagebox.showerror("Error", f"An error occurred: {e}")
        else:
            messagebox.showwarning("Warning", "No file selected")

    def read_row(self):
        if self.file_path:
            try:
                workbook = openpyxl.load_workbook(self.file_path)
                worksheet = workbook.active
                row_values = []
                for cell in worksheet[1]:
                    row_values.append(cell.value)
                messagebox.showinfo("Row Values", f"The values of the first row are: {row_values}")
            except Exception as e:
                messagebox.showerror("Error", f"An error occurred: {e}")
        else:
            messagebox.showwarning("Warning", "No file selected")

    def read_column(self):
        if self.file_path:
            try:
                workbook = openpyxl.load_workbook(self.file_path)
                worksheet = workbook.active
                column_values = []
                for cell in worksheet['A']:
                    column_values.append(cell.value)
                messagebox.showinfo("Column Values", f"The values of column A are:
{column_values}")
            except Exception as e:
                messagebox.showerror("Error", f"An error occurred: {e}")
        else:
            messagebox.showwarning("Warning", "No file selected")

    def count_rows(self):
```

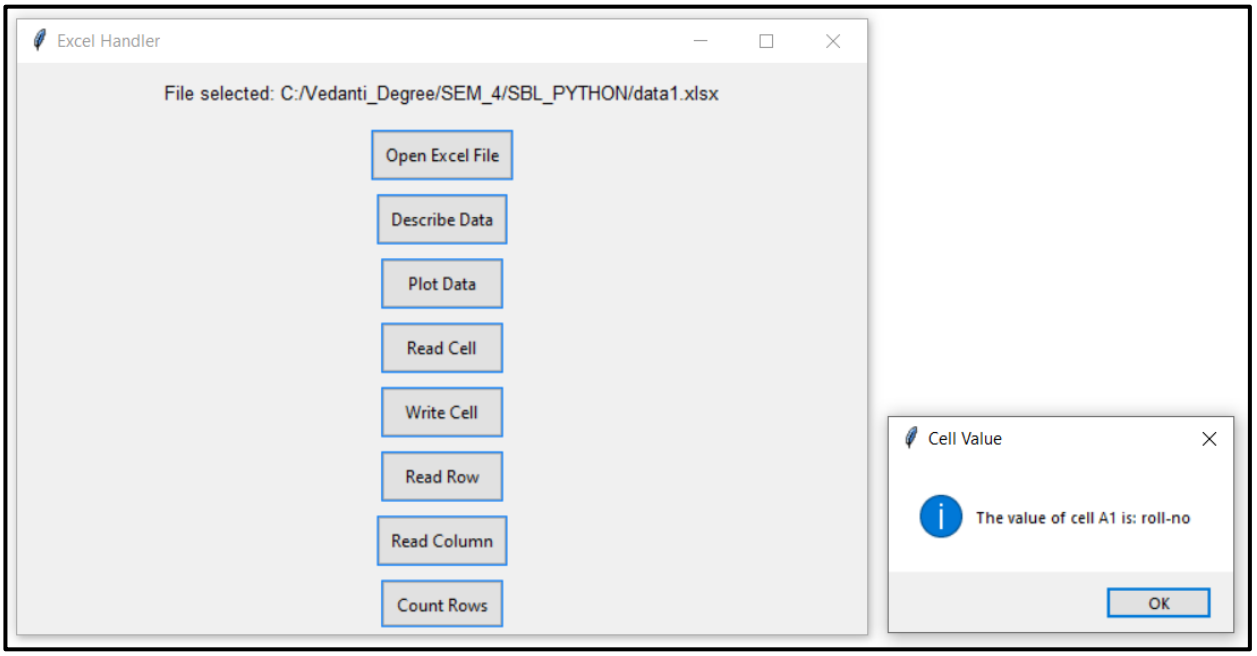
```
        if self.file_path:
            try:
                workbook = openpyxl.load_workbook(self.file_path)
                worksheet = workbook.active
                num_rows = worksheet.max_row
                messagebox.showinfo("Number of Rows", f"The number of rows is: {num_rows}")
            except Exception as e:
                messagebox.showerror("Error", f"An error occurred: {e}")
        else:
            messagebox.showwarning("Warning", "No file selected")

    def count_columns(self):
        if self.file_path:
            try:
                workbook = openpyxl.load_workbook(self.file_path)
                worksheet = workbook.active
                num_columns = worksheet.max_column
                messagebox.showinfo("Number of Columns", f"The number of columns is:
{num_columns}")
            except Exception as e:
                messagebox.showerror("Error", f"An error occurred: {e}")
        else:
            messagebox.showwarning("Warning", "No file selected")

    def read_table(self):
        if self.file_path:
            try:
                workbook = openpyxl.load_workbook(self.file_path)
                worksheet = workbook.active
                table = []
                for row in worksheet.iter_rows(min_row=1, max_row=worksheet.max_row, min_col=1,
max_col=worksheet.max_column):
                    row_data = []
                    for cell in row:
                        row_data.append(cell.value)
                    table.append(row_data)
                messagebox.showinfo("Table Data", f"The table data is:\n{table}")
            except Exception as e:
                messagebox.showerror("Error", f"An error occurred: {e}")
        else:
            messagebox.showwarning("Warning", "No file selected")

if __name__ == "__main__":
    root = tk.Tk()
    app = ExcelHandlerApp(root)
    root.mainloop()
```

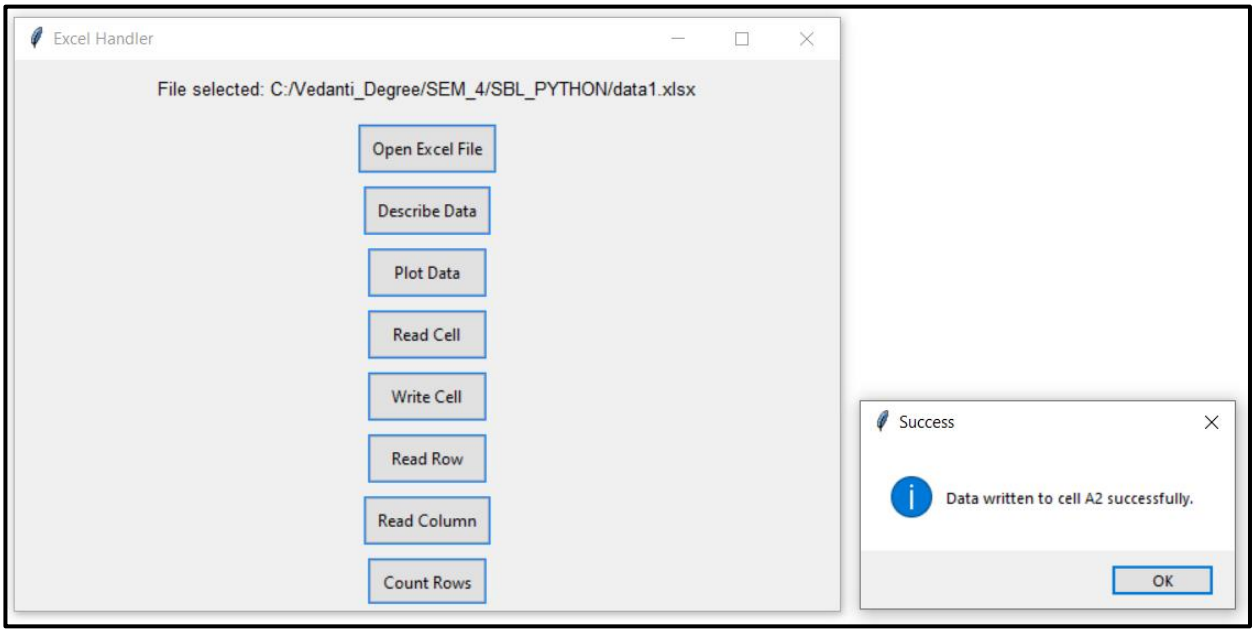
OUTPUT

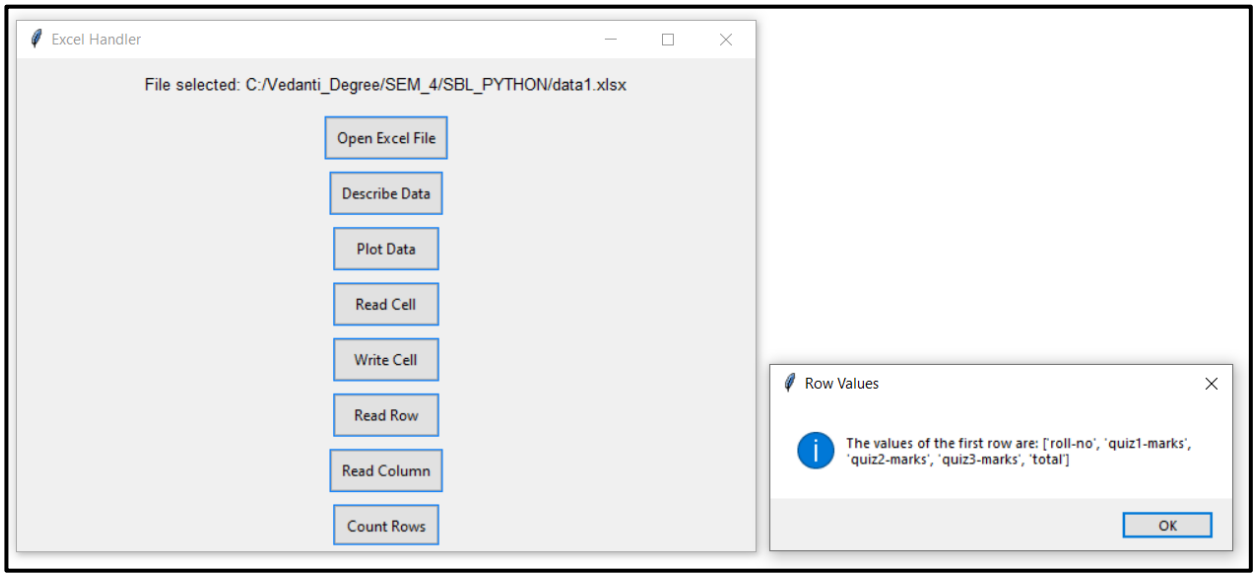

Excel Handler
No file selected
Open Excel File
Describe Data
Plot Data
Read Cell
Write Cell
Read Row
Read Column
Count Rows

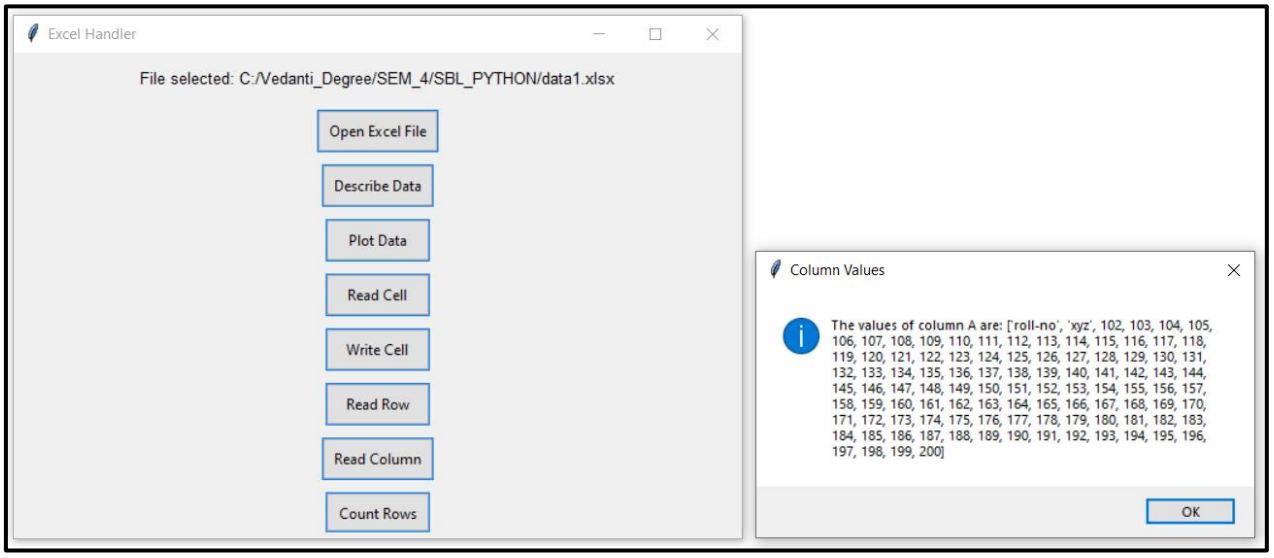

Excel Handler
File selected: C:/Vedanti_Degree/SEM_4/SBL_PYTHON/data1.xlsx
Open Excel File
Describe Data
Plot Data
Read Cell
Write Cell
Read Row
Read Column
Count Rows

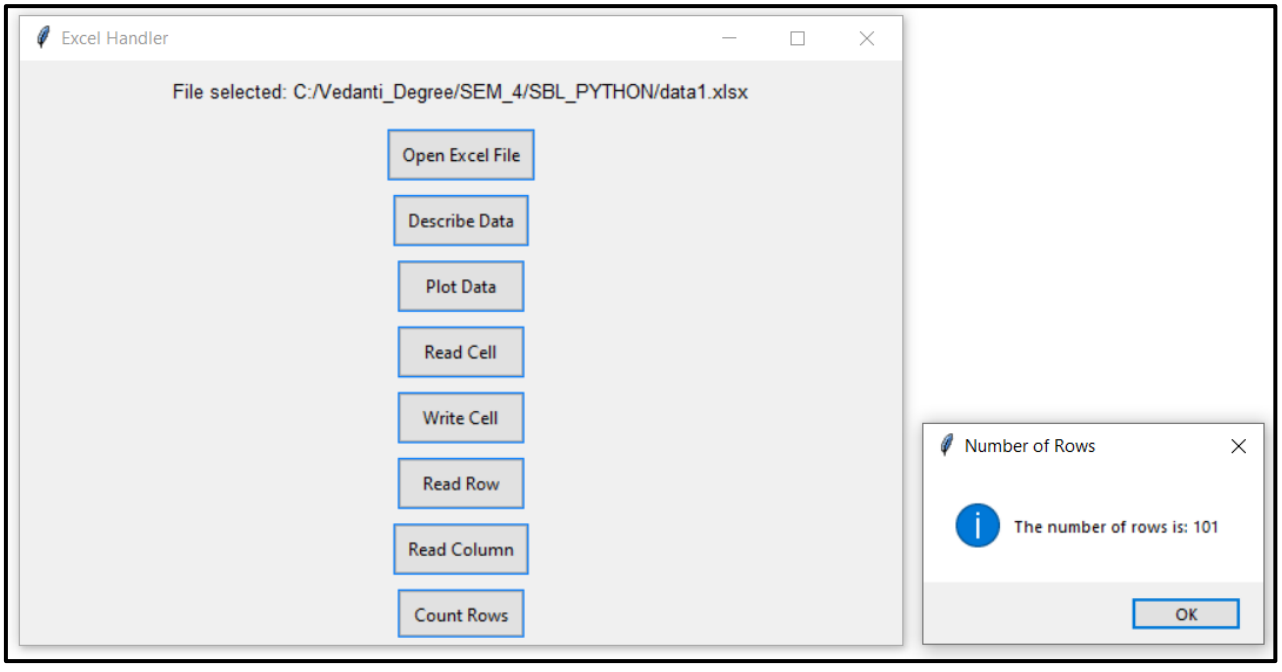

Excel Handler
File selected: C:/Vedanti_Degree/SEM_4/SBL_PYTHON/data1.xlsx
Open Excel File
Describe Data
Plot Data
Read Cell
Write Cell
Read Row
Read Column
Count Rows
Data Description
quiz1-marks quiz2-marks quiz3-marks total
count 100.000000 100.000000 100.000000 100.000000
mean 5.950000 4.960000 5.650000 16.560000
std 2.487849 2.766758 2.854644 5.103613
min 0.000000 0.000000 0.000000 6.000000
25% 4.000000 3.000000 3.000000 13.000000
50% 6.000000 5.000000 6.000000 16.500000
75% 8.000000 7.000000 8.000000 20.000000
max 10.000000 10.000000 10.000000 27.000000
OK


Excel Handler
File selected: C:/Vedanti_Degree/SEM_4/SBL_PYTHON/data1.xlsx
Open Excel File
Describe Data
Plot Data
Read Cell
Write Cell
Read Row
Read Column
Count Rows
Cell Value
The value of cell A1 is: roll-no
OK


Excel Handler
File selected: C:/Vedanti_Degree/SEM_4/SBL_PYTHON/data1.xlsx
Open Excel File
Describe Data
Plot Data
Read Cell
Write Cell
Read Row
Read Column
Count Rows
Success
Data written to cell A2 successfully.
OK


Excel Handler
File selected: C:/Vedanti_Degree/SEM_4/SBL_PYTHON/data1.xlsx
Open Excel File
Describe Data
Plot Data
Read Cell
Write Cell
Read Row
Read Column
Count Rows
Row Values
The values of the first row are: ['roll-no', 'quiz1-marks', 'quiz2-marks', 'quiz3-marks', 'total']
OK


Excel Handler
File selected: C:/Vedanti_Degree/SEM_4/SBL_PYTHON/data1.xlsx
Open Excel File
Describe Data
Plot Data
Read Cell
Write Cell
Read Row
Read Column
Count Rows
Column Values
The values of column A are: ['roll-no', 'xyz', 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200]
OK


Excel Handler
File selected: C:/Vedanti_Degree/SEM_4/SBL_PYTHON/data1.xlsx
Open Excel File
Describe Data
Plot Data
Read Cell
Write Cell
Read Row
Read Column
Count Rows
Number of Rows
The number of rows is: 101
OK

CONCLUSION

This Python-based project for Excel sheet handling and data visualization demonstrates the power of Python and its libraries in simplifying complex tasks. By leveraging pandas, numpy, matplotlib, and tkinter, the project provides a unified platform for efficient Excel data manipulation, analysis, and visualization. The user-friendly interface ensures a seamless experience for users of all backgrounds and technical proficiencies. The project not only enhances productivity but also makes the process of interacting with Excel data enjoyable. It stands as a testament to the potential of Python in transforming the way we work with data, paving the way for future innovations in data handling and visualization. With this project, we take a step closer to making data analysis more accessible and user-friendly for everyone.

REFERENCES

1. https://www.simplilearn.com
2. https://realpython.com
3. https://www.geeksforgeeks.org
4. https://www.freecodecamp.org