**Assignment**

**CSA0814 – Python Programming**

|  |  |
| --- | --- |
| **Register Number** | **192324298** |
| **Name** | **Akash. S** |

**Title:** **1. CSV Reader: Develop a Python program that reads data from a CSV file and displays it in tabular format, allowing users to specify the delimiter and handle missing or malformed data gracefully.**

**Problem Statement:**

**CSV Reader with Custom Delimiter and Error Handling**

**Objective: Create a Python program that reads data from a CSV file and displays it in a tabular format, allowing users to specify the delimiter and handle missing or malformed data gracefully.**

**Background: CSV (Comma-Separated Values) files are commonly used for storing and exchanging data in a structured, tabular format. However, CSV files can have variations, such as different delimiters (e.g., commas, semicolons, tabs) and potential issues like missing or malformed data. Users need a flexible tool to read, display, and manage data from CSV files while addressing these variations.**

**Requirements:**

1. **CSV File Reading: The program should accept a user-specified file path and delimiter to read a CSV file. It should be able to handle different delimiters commonly used in CSV files.**
2. **Error Handling:**
   * **Missing Data: The program should identify and manage missing data, replacing it with a user-defined placeholder (e.g., "N/A").**
   * **Malformed Data: The program should skip or alert the user about rows with incorrect formatting (e.g., rows with missing or extra columns).**
3. **Data Display: The program should display the read data in a tabular format within a GUI or console, ensuring the data is clearly organized and easy to read.**
4. **User Interface: Provide an intuitive interface where users can:**
   * **Specify the delimiter.**
   * **Handle missing or malformed data according to their preferences.**
   * **Adjust the display settings, such as column width.**
5. **Performance: The program should be efficient, handling large files without significant delays or memory issues.**

**Deliverables:**

* **A Python script that implements the CSV reader with the above functionalities.**
* **Documentation explaining how to use the program and handle various cases like custom delimiters and missing data.**
* **Test cases with sample CSV files to demonstrate the program's capabilities.**

**Code:**

**import pandas as pd**

**import tkinter as tk**

**from tkinter import messagebox**

**from tkinter import ttk**

**def read\_and\_display\_csv(file\_path, delimiter=','):**

**try:**

**# Reading the CSV file with the specified delimiter**

**data = pd.read\_csv(file\_path, delimiter=delimiter, on\_bad\_lines='skip')**

**# Handling missing data by filling with 'N/A'**

**data = data.fillna('N/A')**

**# Convert all columns to strings to ensure proper display**

**data = data.astype(str)**

**# Display the data in a GUI table**

**display\_data\_in\_gui(data)**

**except FileNotFoundError:**

**messagebox.showerror("Error", "The specified file was not found.")**

**except pd.errors.ParserError:**

**messagebox.showerror("Error", "There was an issue with parsing the CSV file. Please check the delimiter and the file format.")**

**except Exception as e:**

**messagebox.showerror("Error", f"An unexpected error occurred: {e}")**

**def display\_data\_in\_gui(data):**

**# Create the main window**

**root = tk.Tk()**

**root.title("CSV Data Viewer")**

**# Create a frame for the table**

**frame = tk.Frame(root)**

**frame.pack(fill='both', expand=True)**

**# Create a Treeview widget for the table**

**tree = ttk.Treeview(frame)**

**tree.pack(fill='both', expand=True, side='left')**

**# Define the columns**

**tree["columns"] = list(data.columns)**

**tree["show"] = "headings" # Hide the first empty column**

**# Set the column headings and alignment**

**for col in data.columns:**

**tree.heading(col, text=col)**

**tree.column(col, anchor="center") # Change "center" to "w" for left-align or "e" for right-align**

**# Add data to the table**

**for index, row in data.iterrows():**

**tree.insert("", "end", values=list(row))**

**# Add a scrollbar**

**scrollbar = ttk.Scrollbar(frame, orient="vertical", command=tree.yview)**

**scrollbar.pack(side='right', fill='y')**

**tree.configure(yscroll=scrollbar.set)**

**# Start the GUI main loop**

**root.mainloop()**

**# Directly specify the file path**

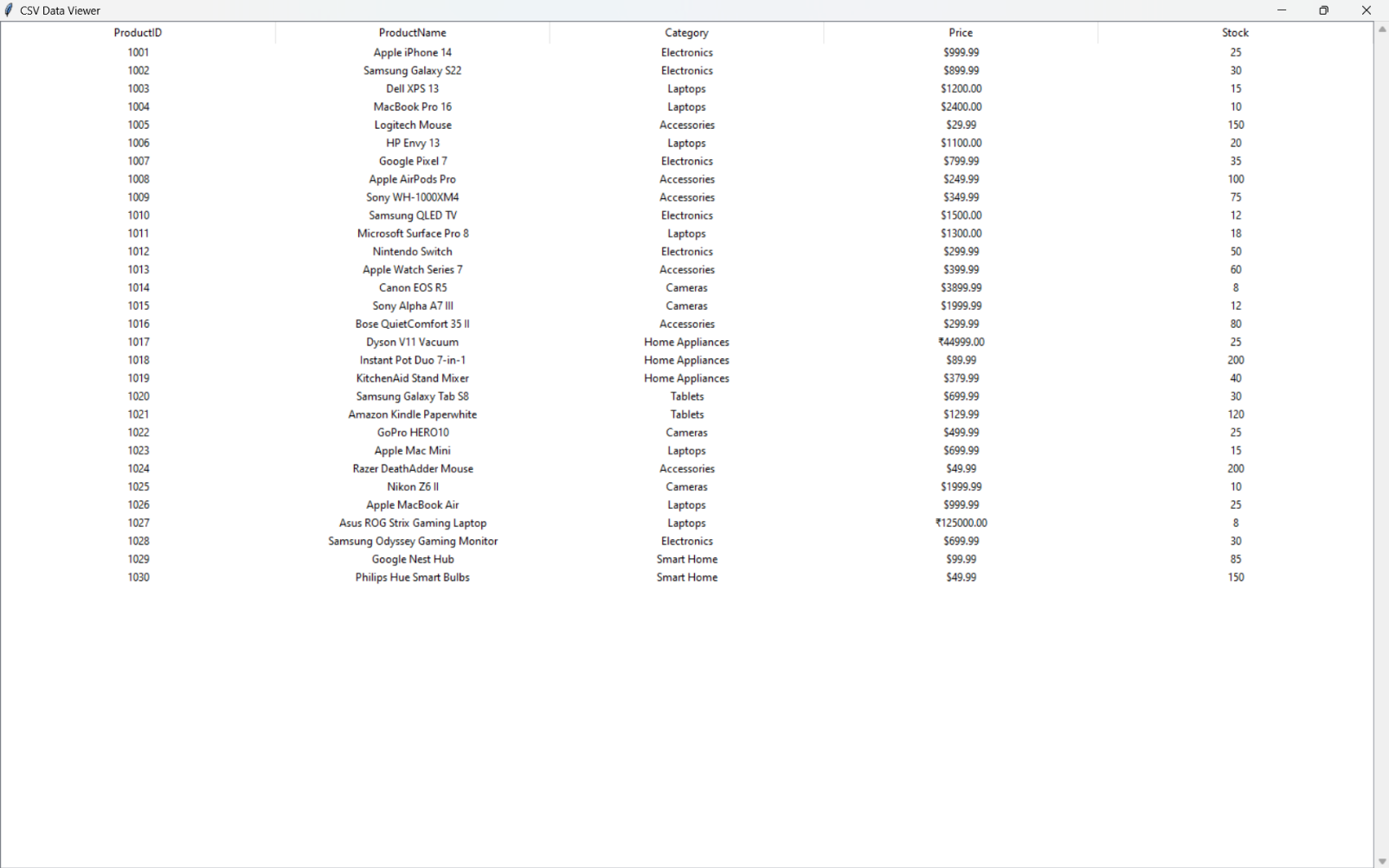
**file\_path = r'C:\Users\akash\OneDrive\Documents\example.csv' # Use raw string to avoid escape character issues**

**delimiter = ',' # Replace with your desired delimiter**

**# Display the CSV data**

**read\_and\_display\_csv(file\_path, delimiter)**

**Link to the csv file:** [**https://drive.google.com/file/d/1lq4MwkosAwF1KqvUEzc2UKFzkS4OZIsi/view?usp=sharing**](https://drive.google.com/file/d/1lq4MwkosAwF1KqvUEzc2UKFzkS4OZIsi/view?usp=sharing)

**Output Screen Shots:**

**Conclusion:**

**The program must efficiently handle large files, providing a smooth user experience and addressing the common challenges associated with CSV data processing, such as varying delimiters and incomplete or incorrect data rows. The final implementation should be robust, user-friendly, and adaptable to various CSV file formats and sizes.**