PROJECT REPORT

DATA VISUALIZATION DASHBOARD

Course: COMP6047001 - Algorithm and Programming

Method of Assessment: Project

Semester/Academic Year: Odd Semester / 2025 – 2026

By:

Vabielo Keanu Starworld

2902728352

L1AC

CHAPTER 1: PROJECT SPECIFICATION

1.1 Problem Description

Many datasets are stored in CSV files, but reading data in this format is not easy for everyone. When data is shown only as rows and columns, it is hard to see patterns or changes over time. This makes it difficult for users to understand the information, especially when the dataset is large.

Analysing data using the command line can also be confusing. Users must type commands and read text output, which is not very clear or interactive. Because of this, there is a need for a simple program that can load CSV files and display the data in a visual way using graphs, so users can understand the data more easily.

1.2 Project Objectives

The objectives of this project are:

1. To allow users to upload CSV files into the program
2. To identify categorical and numeric columns from the data
3. To let users select data to be displayed using dropdown menus
4. To display data using line graphs for better understanding
5. To create a simple graphical user interface instead of using the command line
6. To apply basic Python programming concepts and external libraries

1.3 Project Scope

This project focuses on creating a simple data visualization dashboard using Python. The program is designed to work only with CSV files that contain at least one categorical column, one numeric column, and a date column. The user can select a category and a numeric value to generate a line graph based on the selected data.

The project does not include advanced data analysis or complex calculations. It only supports basic data filtering and visualization using line graphs. The program is intended for small to medium datasets and is designed for learning purposes, not for professional or large-scale data analysis.

CHAPTER 2: SOLUTION DESIGN

2.1 Design Approach

The design of this project started with a simple command line interface (CLI) program. This version allowed basic data analysis using text output. However, it was not easy to use and did not show data in a visual way. To improve usability, the project was later changed into a graphical user interface (GUI) application using Tkinter.

A modular design approach was used to organise the program into different files. Each module has a specific task, such as loading data or handling the user interface. This makes the program easier to understand, maintain, and improve. The GUI uses an event-driven approach, where actions happen based on user input, such as selecting options from dropdown menus or clicking buttons.

2.2 Program Structure

The program is organised into several Python files to keep the code clear and well structured. The main file for the graphical user interface is *gui.py*, which handles user interaction, file upload, data selection, and graph generation.

A folder named *modules* is used to store supporting files. The *data_loader.py* file contains the *DataLoader* class, which is responsible for loading CSV files safely. Other files in the *modules* folder, such as *analyzer.py*, *visualizer.py*, and *dashboard.py*, were part of an earlier command line version of the project. These files are kept to show the development process and modular design but are not used in the final GUI version.

The *main.py* file was used to run the command line version of the program. It is not used in the final demonstration but remains in the project to show how different modules were integrated. An empty *__init__.py* file is included to mark the *modules* folder as a Python package and allow proper imports.

2.3 Data Structures Used

The main data structure used in this project is the pandas DataFrame. The DataFrame is used to store and manage data loaded from CSV files. It allows the program to filter data, group values, and select specific columns easily.

Python lists are used to store values for the dropdown menus in the graphical user interface. These lists are updated when the user selects different options. String variables (*StringVar*) from Tkinter are used to track user selections in the GUI. Basic Python variables are also used to store file paths and selected values during program execution.

2.4 Algorithms Implemented

Several simple algorithms are used in this project to process and display data. First, the program checks each column in the CSV file to decide whether it is a categorical column or a numeric column. This is done by checking the data type of each column using pandas. When the user selects a category and a category value, the program filters the data to show only the selected information.

After filtering, the data is grouped by the date column, and numeric values are added together for each date. The dates are treated as strings instead of date objects and are sorted based on their text order, which works correctly for date formats such as YYYY-MM-DD. Finally, the selected numeric data is plotted using a line graph.

CHAPTER 3: IMPLEMENTATION and WORKINGS

3.1 Core Features

The program includes several core features that allow users to explore and visualise data easily. The main features of the system are:

- Uploading CSV files through the graphical user interface
- Automatically detecting categorical and numeric columns
- Displaying dropdown menus based on the selected data
- Filtering data based on user selection
- Generating line graphs to show data changes over time
- Handling basic errors, such as invalid file selection

3.2 Program Execution

When the program starts, the graphical user interface window is displayed. The user uploads a CSV file using the file selection button. After the file is loaded, the program reads the data and identifies the available categorical and numeric columns. The dropdown menus are then updated based on the selected CSV file.

Next, the user selects a category column, a category value, and a numeric column from the dropdown menus. Once all selections are made, the program filters the data and groups it by the date column. Finally, a line graph is generated and displayed to the user. If an error occurs, such as missing selections or an invalid file, the program shows a message to inform the user.

3.3 Object-Oriented Programming Implementation

Object-oriented programming is used in this project to organise the code in a clearer way. The DataLoader class is implemented to handle the task of loading CSV files. This keeps the file loading process separate from the user interface logic. Using a class helps make the program easier to understand and maintain. It also allows the same loading logic to be reused if the program is expanded in the future. Other parts of the program interact with the DataLoader class instead of handling file operations directly.

3.4 Input Validation and Error Handling

Input validation and error handling are used to prevent the program from crashing and to guide the user when a mistake happens. When a CSV file is uploaded, the program checks whether the file can be read correctly. If the file is invalid or cannot be loaded, an error message is shown to the user.

The program also checks whether the user has selected all required options before generating a graph. If a category column, category value, or numeric column is missing, the program displays a message asking the user to complete the selection. These checks help make the program more stable and easier to use.

3.5 Use of Modules and External Libraries

This project uses both built-in Python modules and external libraries to complete different tasks in the program. The code is separated into multiple modules to make it easier to read, understand, and manage. Each module is responsible for a specific part of the system, such as loading data or handling the user interface.

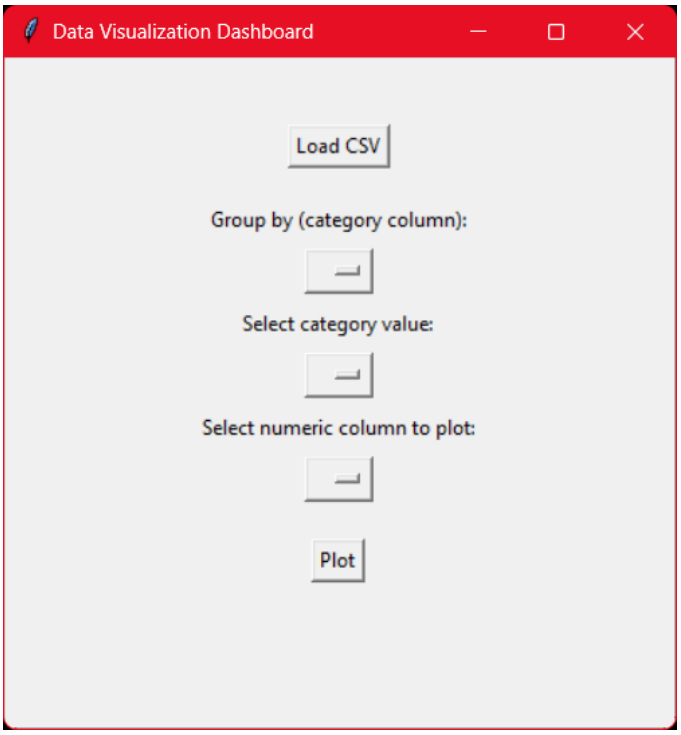
The *data_loader.py* module is used to load CSV files safely using pandas. It returns the data in a structured format so it can be processed by the main program. Other modules, such as *analyzer.py*, *visualizer.py*, and *dashboard.py*, were created during an earlier command line version of the project. These modules are not used in the final GUI version, but they are kept to show modular design and the development process.

Several external libraries are also used in this project. The pandas library is used to read CSV files, identify column types, filter data, and group values by date. Matplotlib is used to generate line graphs so data trends can be seen visually. Tkinter is used to create the graphical user interface, including buttons, dropdown menus, and user interaction. Using these libraries makes the program easier to develop and helps improve functionality without writing complex code from scratch.

CHAPTER 4: EVIDENCE of WORKING PROGRAM

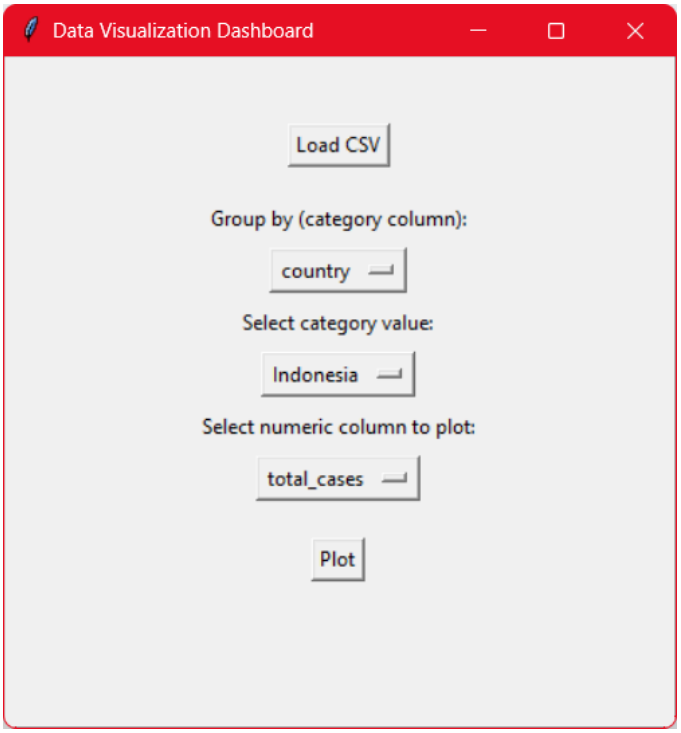
4.1 Running the Program

The program was started by running python gui.py. The graphical user interface window opened successfully.



4.2 Uploading a Valid CSV File

A valid CSV file was uploaded using the file selection button. The program loaded the file and displayed available options in the dropdown menus.



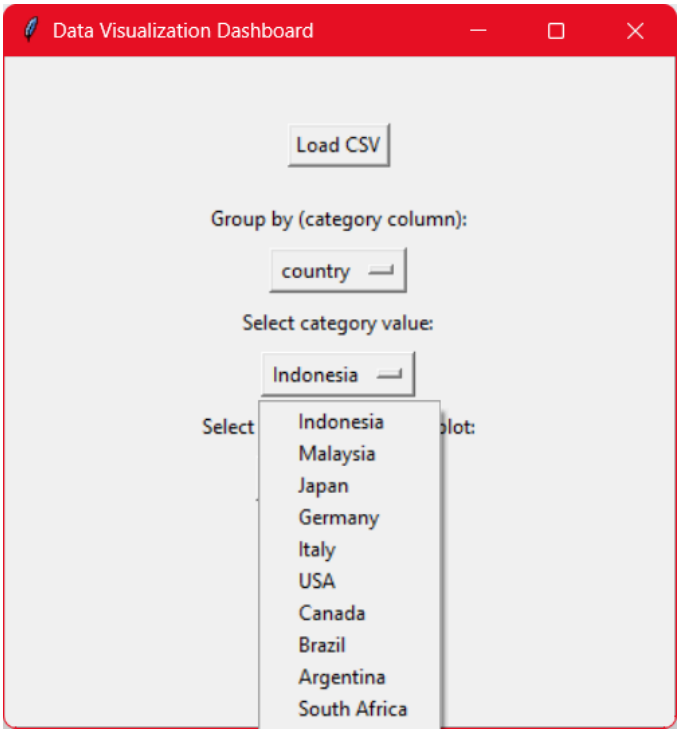
4.3 Uploading an Invalid File

An invalid or unsupported file was selected to test error handling. The program displayed an error message through the terminal.

```
Loaded: 3 rows, 2 columns
Columns: ['faculty', 'students']
Categorical columns: ['faculty']
Numeric columns: ['students']
No date column found.
Error loading CSV: No columns to parse from file
Failed to load CSV.
```

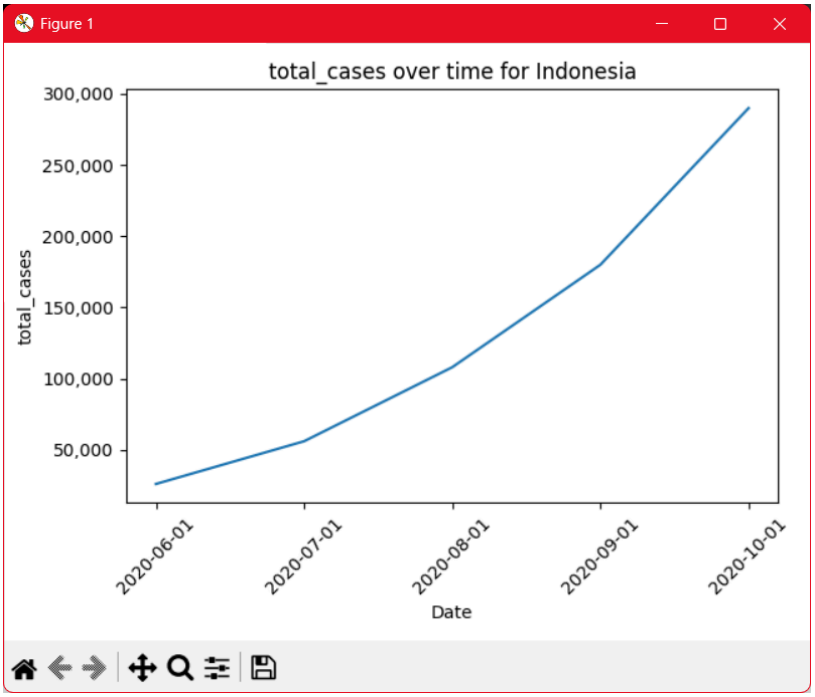
4.4 Selecting Category Column and Value from Dropdown

A list of category columns and category values were shown by the dropdown menus. The program updated the available options correctly.



4.5 Generating Line Graph

The plot button was clicked to generate the graph. The line graph was displayed based on the selected data.



STATEMENT on the USE of ARTIFICIAL INTELLIGENCE

Artificial intelligence was used in this project as a learning support tool. The AI tool used was ChatGPT (GPT-5).

ChatGPT was used during the coding phase to understand programming concepts and debug errors, and during the writing phase to review grammar and clarity in the project report. It was not used to generate final program code or to fully write the report.

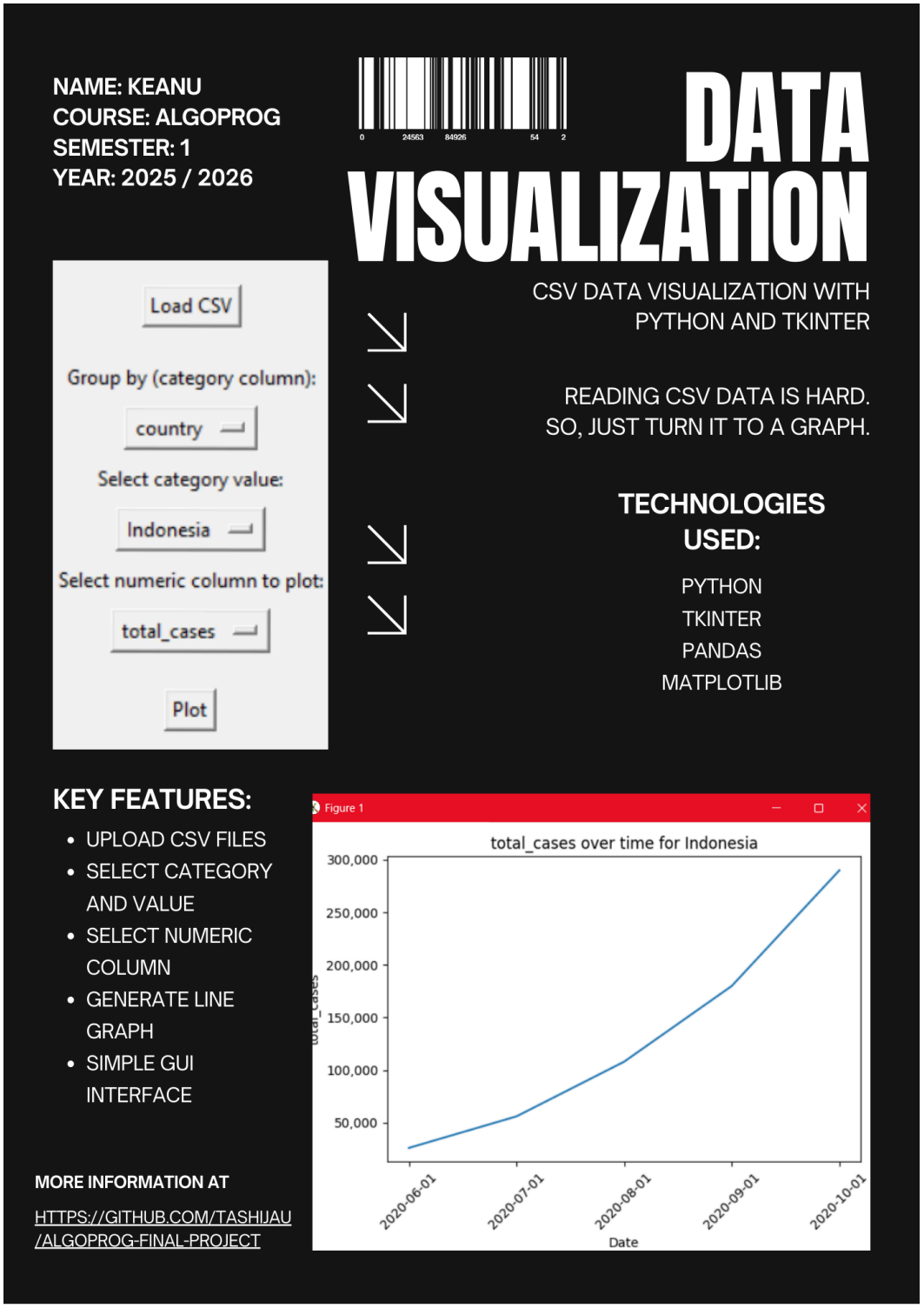
Examples of prompts used include:

- “What error occurred here?” (with screenshots)
- “How do you fix this error?” (with screenshots)
- “How do I implement Tkinter?”
- “Is the grammar here correct?”

All AI output was checked carefully. Code-related suggestions were tested and edited manually, and text suggestions were reviewed to match the student’s own understanding. No AI-generated code was directly used. All final work was completed by the student.

ADDITIONAL DOCUMENTATION

A3 POSTER



GITHUB LINK

https://github.com/TasHijau/AlgoProg-Final-Project

REFERENCES

Hunter, J. D. (2007). *Matplotlib: A 2D graphics environment*. Computing in Science & Engineering, 9(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

McKinney, W. (2010). *Data structures for statistical computing in Python*. Proceedings of the 9th Python in Science Conference, 51–56.

Python Software Foundation. (2024). *Python documentation*. https://docs.python.org/3/

Python Software Foundation. (2024). *Tkinter — Python interface to Tcl/Tk*. https://docs.python.org/3/library/tkinter.html