Chapter 1

Introduction

1.1 Background

- **Context**: The **Excel Data Management System (EDMS)** is a software project designed to enhance the efficiency, accuracy, and accessibility of managing large volumes of data using Microsoft Excel. This project aims to provide a streamlined system for organizing, analyzing, and storing data in a manner that simplifies tasks like data entry, retrieval, and reporting.

- **Problem**: The **Excel Data Management System (EDMS)** project addresses the critical problems of manual data entry errors, the project solves the problem of efficiently managing, updating, and visualizing student data (including marks and personal details) while storing it in a structured and easily accessible format (Excel). The system reduces manual errors, increases productivity, and makes it easier for users to perform data operations.

- **Opportunity:** A simple opportunity for the Student Data Management System project is the ability to expand its functionality to include student self-service.

1.2 Problem Statement

To develop a application with a graphical user interface (GUI) to manage data from an Excel file which allows users to view, add, update, and delete data records stored in an Excel file, as well as visualize selected data through graphs

1.3 Objective of the System

The objective of the problem you are addressing is to create a **Excel Data Management System** using Python with **Tkinter** for a user interface and **Pandas** for handling student data stored in an Excel file. The system should allow users to:

1. **Add New Student Data**:
    - Users should be able to enter information like USN, student name, and marks for various subjects (e.g., MongoDB, PHP, Data Structure), which will be stored in an Excel file.
2. **Update Existing Student Data**:
    - Users should be able to update the information of an existing student by specifying their USN. The data corresponding to that USN will be updated in the Excel file.
3. **Delete Student Data**:
    - Users should be able to delete a student's record from the Excel sheet based on the provided USN.
4. **Display Data in a Treeview**:
    - The system should display all the student data from the Excel file in a table-like view (Treeview widget) in the interface. This allows users to visually inspect the data and interact with it.
5. **Plot Graph**:
    - The system should allow the plotting of a **bar graph** that visualizes the relationship between total marks and percentage for each student. This helps users understand the data distribution.
6. **User-Friendly Interface**:
    - The application should be designed with an intuitive user interface using **Tkinter**.

1.4 Significance of the System

- **Excel Data Handling**:

  This project teaches users how to manage and manipulate data stored in Excel files, which is a widely used format for data storage in real-world scenario

- **GUI Design**:

  By using **Tkinter**, this project enhances your ability to create user- friendly graphical interfaces. Tkinter is one of the most used libraries for GUI development in Python, and this project provides hands-on experience in designing interactive windows, forms, and buttons.

- **Plotting Graphs**:

  The inclusion of the **bar graph** plot (showing the relationship between total marks and percentage) helps users visually interpret the data. This is a valuable skill, as data visualization is a key aspect of presenting data insights clearly.

- **Excel as a Database:**

  By using Excel as a data storage backend, users are introduced to how databases can store structured data. While not as powerful as SQL databases, Excel is commonly used for small-scale data management.

- **Python Programming**:

  This project deepens your understanding of Python, especially libraries like Pandas for data manipulation and Matplotlib for data visualization. These are highly sought-after skills in data science, data analysis, and backend development.

1.5 Scope of the Project

- **In Scope**:
    - To create a data management system using Tkinter and pandas, allowing users to add, update, delete, and visualize student data (including subjects and marks) from an Excel file, with a graphical representation of student performance via matplotlib.
    - **Data Entry**: The system will allow users to input student details such as **USN, Student Name, MongoDB Marks, PHP Marks, Data Structure Marks, Total Marks, and Percentage**.
    - **Data Update**: Users will be able to update existing student records using the unique **USN** (University Serial Number) as an identifier. This allows editing details for a specific student.
    - **Data Deletion**: Users will be able to delete student records based on the **USN**.
    - **Data Persistence**: All student data will be stored in an **Excel file**, ensuring data is saved between application sessions.
    - **CRUD Operations**: The project will implement the basic **Create, Read, Update, and Delete** functionalities for student records.
- **Out of Scope**:
    - Support for Non-Excel Spreadsheet Software
    - Advanced Data Analytics or Machine Learning
    - External Data Storage Solutions

1.6 Methodology

- **Approach:** The approach of the system is to provide an intuitive graphical user interface (GUI) for managing student data, leveraging Python libraries such as Tkinter for the frontend, pandas for data manipulation, and matplotlib for data visualization.

- **Agile Development:** The system follows an agile development approach, enabling iterative enhancements with continuous feedback and improvements in functionality such as adding, updating, and visualizing student data in response to user needs.

- **Testing:** Testing for the system is conducted by validating data input, ensuring correct data manipulation in the Excel file, checking the accuracy of visualized reports, and ensuring that all features such as adding, updating, deleting, and graphing function as expected.

1.7 Target Audience

- **Educational Institutions (Schools, Colleges, and Universities):** Educational institutions manage vast amounts of data related to student records, grades, attendance, staff performance, and more.

- **Healthcare Organizations**: Hospitals, clinics, and healthcare providers manage sensitive patient data, treatment records, and billing information.

- **Project Management Teams**: Companies managing large projects need to keep track of budgets, timelines, resources, and progress.

1.8 **Overview of the Report**

This report is structured into several chapters that detail the development and design of the **Student Attendance System**. The following chapters include:

- **Chapter 2: System Design: -** Describes the architecture and design of the system.
- **Chapter 3: Implementation: -** Discusses the system's development and the technologies used.
- **Chapter 4: Testing and Validation: -** Details the testing process and results.
- **Chapter 5: Results and Discussions: -** Presents and results obtained and discusses the limitations
- **Chapter 6: Conclusion and Future enhancement: -** Summarizes the project and suggests future improvements.

Chapter 2

System Design

The Excel Data Management System (EDMS) is designed to automate, organize, and streamline the management of data within Microsoft Excel. The system will focus on automating repetitive tasks, improving data accuracy, ensuring data integrity, and facilitating the creation of reports and analysis. The design of this system involves multiple components that work together to deliver these functions, ranging from user interfaces to backend processes.

2.1 System Architecture

The **EDMS** will have a modular and layered architecture, with distinct components that interact with handling various data management tasks.

- **User Interface (UI) Layer**:
  - Provides a user-friendly experience for data input, report generation, and analytics.
  - Includes forms for data entry, dropdowns for selecting categories, and customizable dashboards for quick access to key metrics.
  - Built with Excel's built-in functionalities like **Data Validation**, **Forms**, and **Excel Tables**.
- **Business Logic Layer**:
  - Handles the core logic of automating tasks, validating data, and applying rules.
  - Includes macros, VBA scripts, and Excel formulas that automate data entry, manage data processing, and create reports.
  - Power Query will be used to automate data import, transformation, and export processes from various sources (e.g., CSV, databases, APIs).

- **Data Layer:**
  - Organizes data in a structured format within Excel, utilizing Excel Tables and named ranges.
  - Ensures data consistency by organizing data into standardized templates and enforcing validation rules.

2.2 Module Design

The **Excel Data Management System (EDMS)** project is designed to provide an efficient and user-friendly solution for managing, analyzing, and automating data tasks within Microsoft Excel.

2.2.1 Data Entry and Validation Module*:*

The Data Entry and Validation Module allows users to input student data such as USN, name, and marks into the system, ensuring valid entries are made by checking if the USN exists when updating or deleting records.

2.2.2 Data Organization and Structure Module:

The Data Organization and Structure Module manages student data in a structured format, storing records in an Excel file with columns for USN, student name, marks in various subjects, total marks, and percentage.

2.2.3 Automation and Scripting Module:

The Automation and Scripting Module automates tasks such as adding, updating, and deleting student data by using pandas to read, modify, and save changes to the Excel file without manual intervention.

2.2.4 Reporting and Analytics Module:

The Reporting and Analytics Module generates visual reports by plotting student percentages in a bar chart using matplotlib, helping to analyze and compare student performance visually.

2.3 Database Design:

The Database Design stores student data in an Excel file where each row represents a student and the columns store information like USN, name, subjects, marks, and percentage, allowing easy retrieval and manipulation of data.

2.4 User Interface (UI) Design

- **Excel User Forms (VBA):**
    - Custom VBA user forms will be designed for efficient data entry and to provide users with a friendly, graphical interface for interacting with the system. These forms will include fields like text boxes, drop-down menus, checkboxes, and date pickers.
- **Conditional Formatting:**
    - Conditional formatting will be used to provide visual cues for users when certain criteria are met, such as flagging data errors or highlighting important data points.
- **Excel ActiveX Controls:**
    - For advanced UI elements like buttons, checkboxes, and list boxes, ActiveX controls will be employed within Excel, providing interactive and customizable elements for the user interface.

2.5 Technology Stack

- **Core Technology**: Microsoft Excel
- **Python**: The main programming language for the application.
- **Tkinter**: For creating the GUI, with widgets such as Buttons, Labels, and Treeview.
- **Treeview:** It is a UI component that displays hierarchical data in a tree structure, allowing users to expand or collapse nodes to navigate through the levels.
- **Pandas:** For reading from and writing to Excel files, as well as handling data manipulations.
- **Matplotlib**: For data visualization, plotting graphs based on selected data.

Chapter 3

Implementation

This chapter outlines the steps taken to implement the Student Attendance System, covering the backend, frontend, database, and integration processes. It describes the technologies used, the structure of the codebase, and any special development techniques.

3.1 Backend Implementation

The backend implementation of the excel Data Management System is responsible for managing the data of students, ensuring that the system can interact with an Excel file, update, retrieve, and delete records, and perform necessary data operations.

The backend will handle the following:

- Adding, updating, and deleting data in an Excel file.
- Loading data for display.
- Plotting data.

3.2 Frontend Implementation

- Implements a Tkinter-based GUI.
- Communicates with the backend to perform CRUD operations and show graphs.

3.3 Database Implementation

- **Database Setup**: The project uses **Excel** as a database for storing the student data. The Excel file Rescue_Students.xlsx acts as a simple database table where each student record is a row, and each subject, USN, and marks information is a column.

- **Database operations**:
  - **Add data:** The **add_data()** function simulates the INSERT operation in a database. It adds new records to the "database" (the Excel file).
  - **Update data**: The **update_data()** function simulates the UPDATE operation in SQL. It allows updating an existing record (row) in the Excel file by searching for the student's **USN**.
  - **Delete data:** The **delete_data()** function simulates the DELETE operation in SQL. It deletes a record from the Excel file based on USN.
  - **Read data**: The **refresh treeview()** function simulates the SELECT operation in SQL. It retrieves the data from the Excel file and displays it in the **Treeview** widget in the UI.
  - **Database visualization graph:** The plot graph() function simulates generating a report or visualization from the database. It uses Matplotlib to create a graph based on the data.

Chapter 4

Testing

This chapter covers the testing processes and methodologies applied to the Excel Data Management System. Testing is essential to identify and correct any issues, validate that the system meets functional and non-functional requirements, and ensure that it performs reliably under various conditions.

4.1 Testing Objectives

- Verify that all functionalities, such as adding, updating, and deleting student records, work correctly without errors.
- Ensure the application handles invalid inputs properly and provides clear error messages.
- Check that the data in the Excel file remains consistent after any operation is performed.
- Assess the user interface to confirm that all widgets are responsive and the Treeview displays the correct data.
- Validate that the application handles exceptions gracefully and informs users of any issues encountered.

4.2 Testing Environment

- **Hardware**: A computer with at least 4GB RAM and basic storage.
- **Processor**: Intel Core i5 or equivalent (minimum), i7 or higher (recommended).
- **RAM**: 8 GB (minimum), 16 GB (recommended for large datasets).
- **Storage**: 256 GB SSD or higher (enough to handle Excel file storage and Python libraries).
- **Display**: 1366 x 768 resolution (minimum), 1920 x 1080 or higher (recommended for better layout display).

- **Graphics**: Integrated GPU (sufficient for plotting graphs with matplotlib).
- **Software**: Python with Tkinter, pandas, and matplotlib libraries.
- **Python Version**: Python 3.8 or higher (tested with Python 3.10+ recommended).
- **Required Libraries**:
  - pandas (for data handling) – Install using pip install pandas.
  - openpyxl (for Excel file handling) – Install using pip install openpyxl
  - matplotlib (for graph plotting) – Install using pip install matplotlib
  - tkinter (for GUI) – Pre-installed with Python.
- **Operating System**: Windows 10/macOS/Linux.
- **Browser**: Google Chrome, Mozilla Firefox, and Microsoft Edge for cross-browser testing.

4.3 Types of Testing

4.3.1 Unit Testing

- **Objective**: Focus on testing individual functions or methods to ensure they work as expected.

Sl No	Tool Name	Usage
1	Tkinter	GUI
2	Pandas	Data handling
3	Matplotlib	Plotting
4	Openpyxl	Excel file operations

Table 4.1 : Tools Used

- **Example Test Cases**: Test cases include verifying add, update, delete, reset, plot

graph functionality, Treeview display, handling invalid inputs, and ensuring data consistency in the Excel file.

4.3.2 Integration Testing

- **Objective**: Integration testing for the code would involve validating interactions between components like GUI, Excel file operations, and data handling.
- **Example Test Cases**:
  - **Add Data**: Verify that adding a record update both the Excel file and the Treeview display.
  - **Update Data**: Ensure updating a record reflects changes in the Excel file .
  - **Delete Data and Plot Graph**: Confirm deleting a record updates the Excel file and excludes the deleted data from the graph.
  - **Input Field Reset and Treeview Interaction**: Test that resetting fields after selecting a Treeview row does not affect other components.
  - **Plot Graph with Treeview**: Ensure the graph matches the data displayed in Treeview after any operation.

4.3.3 Functional Testing

- **Objective**: To test the system against functional requirements to ensure it meets specified user needs.
- **Test Scenarios**:
  - **Add Data**: Verify that adding a new student updates the Excel file and Treeview.
  - **Update Data**: Check that updating an existing student's details updates the Excel file and treeview.
  - **Delete Data**: Ensure that deleting a student removes the record from both Excel and Treeview.
  - **Reset Fields**: Confirm that clicking "Reset" clears all input fields.
  - **Plot Graph**: Verify that the "Plot Graph" button displays a bar chart with the

correct data.

4.4 Test Cases

Below are sample test cases for various components:

Test Case ID	Description	Test Steps	Expected Result	Status
TC-001	Add a new student record.	1. Enter valid data in all fields. 2. Click "Add Data."	Record is added to Excel , displayed in Treeview , success message appears.	Pass
TC-002	Update an existing student record.	1. Enter a valid Student ID. 2. Modify details. 3. Click "Update."	Record is up dated in Excel, Treeview reflects changes, success message appears.	Pass
TC-003	Delete a student record	1. Enter a valid StudentID. 2. Click "Delete."	Record is removed from Excel and Treeview , success message appears.	Pass
TC-004	Reset input fields.	1. Enter data in fields. 2. Click "Reset."	All input fields are cleared.	Pass
TC-005	Plot graph for rescue data.	Click “Plot Graph”	Bar graph displaying student data appears.	Pass

TC-006	Select row from Treeview.	1. Select a row in Treeview. 2. Check the input fields.	Input fields populate with selected row data.	Pass

Table 4.2 : Test Cases

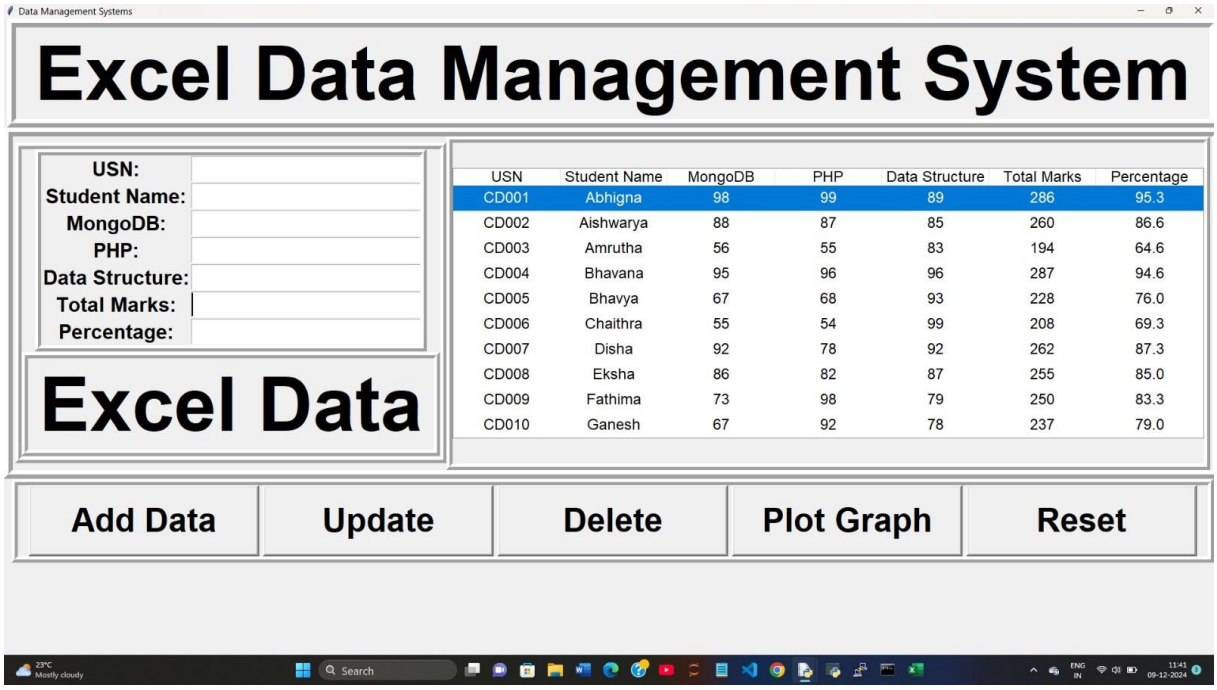
Chapter 5

Results and Discussion

This chapter summarizes the results of the Excel Data Management System, discussing its effectiveness, reliability, and alignment with the intended objectives. The chapter also covers any challenges encountered, key insights, and recommendations for future improvements.

5.1 Results

This successfully implements a GUI-based Excel data management system for student records, allowing seamless operations like add, update, delete, reset, data visualization, and interactive Treeview display.

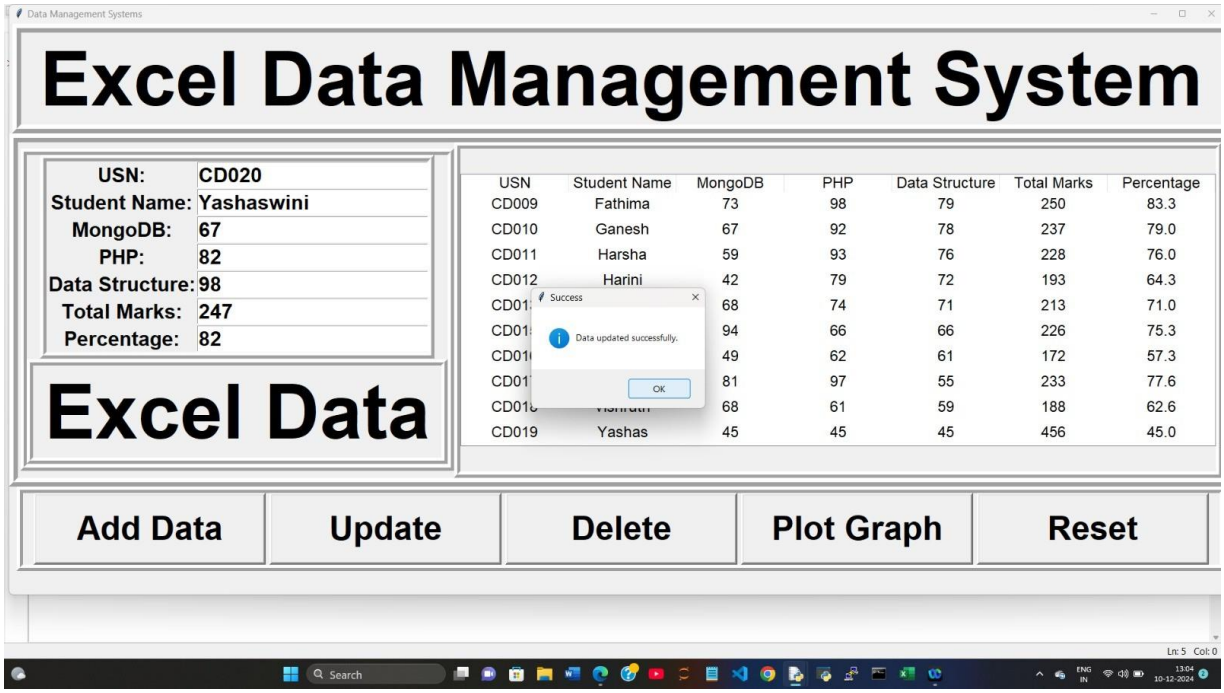


Snapshot 1 : Graphical User Interface

The image shows a GUI titled **"Excel Data Management System"** for managing student records. It has input fields on the left for entering details like USN, student name, scores in MongoDB, PHP, Data Structure, total marks, and percentage. On the right, a table displays student data with columns for these fields.

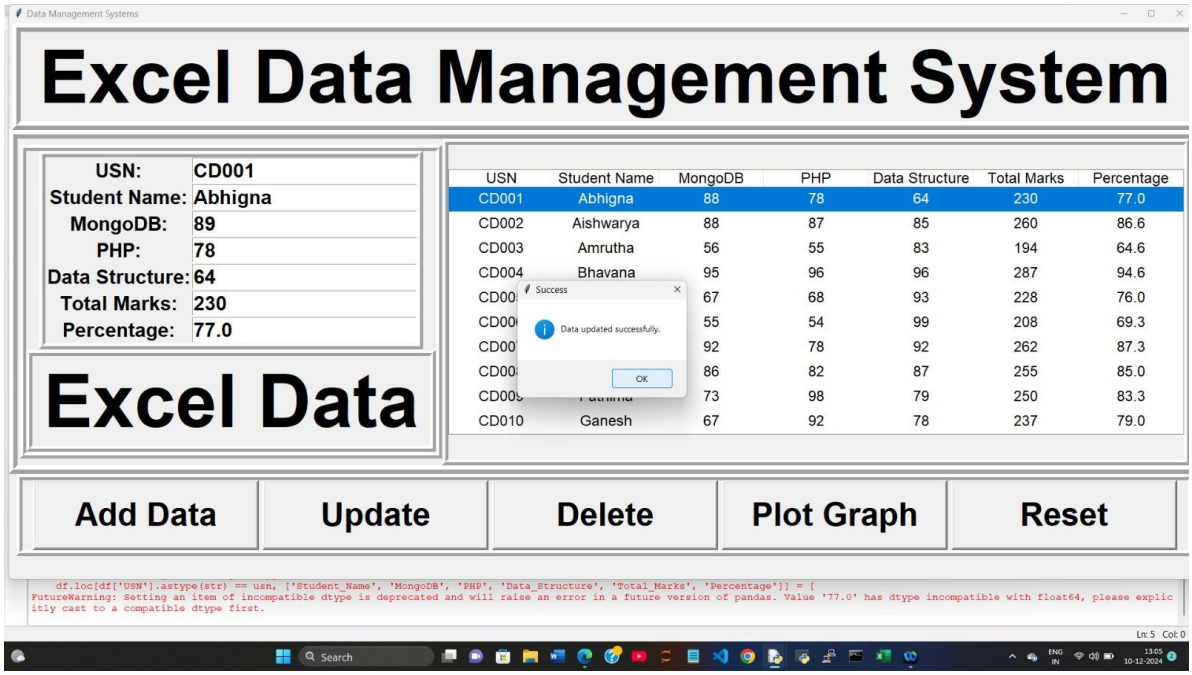
At the bottom are buttons for adding, updating, deleting records, plotting graphs, and resetting the form. The interface is designed for simplicity, likely integrating with Excel or a database, and includes options for data visualisation through graphs.

Add data:



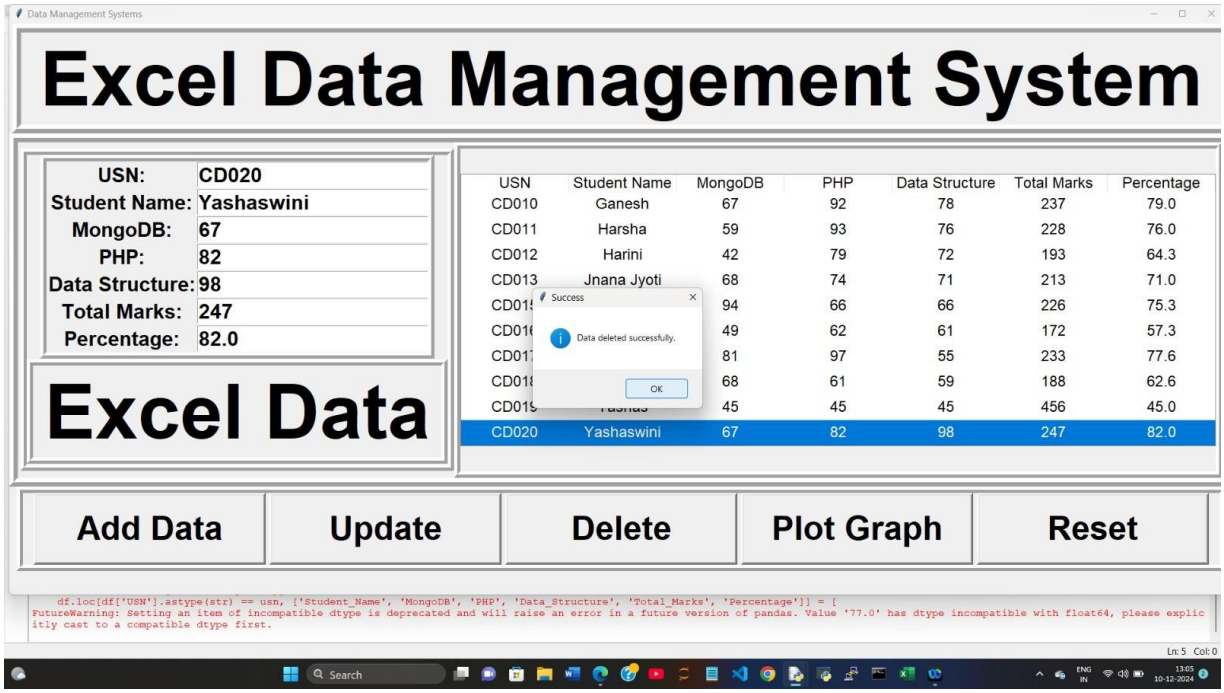
Snapshot 2 : Add new data

Update data:



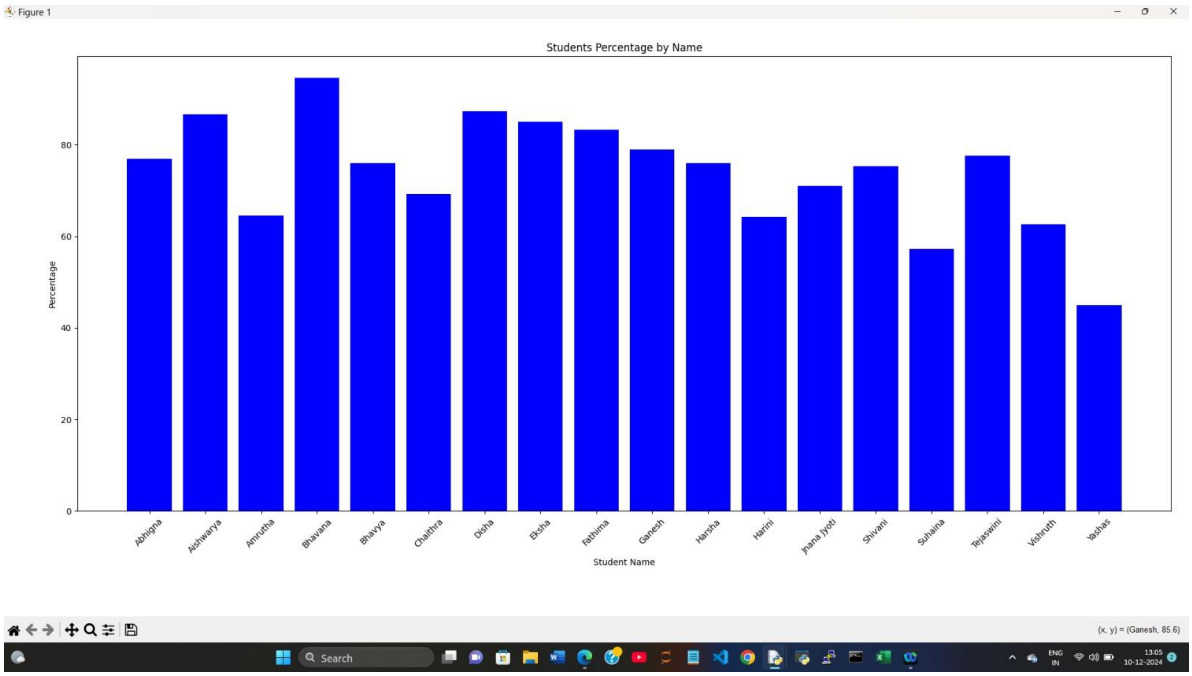
Snapshot 3: Update existing data

Delete data:



Snapshot 4: Delete existing data

Plot graph:



Snapshot 5: Plot graph

5.2 Discussion

- **Effectiveness of the System**
  - The effectiveness of the system lies in its seamless integration of Python's data manipulation capabilities with an intuitive Tkinter-based GUI, enabling efficient management of student records.
  - Features like real-time data addition, updates, deletion, and Excel file synchronization provide users with a reliable tool for data organization.
  - The incorporation of a Treeview for displaying and selecting data ensures easy navigation, while the visualization functionality with Matplotlib enhances data interpretation.
  - Error handling mechanisms ensure robustness, and the system's reset and refresh capabilities contribute to a smooth user experience, making it a versatile solution for educational data management.

- **Challenges Encountered**
  - Challenges encountered in the code include handling errors like missing or improperly formatted Excel files, ensuring synchronization between the Treeview and the Excel sheet, and managing user inputs to prevent invalid or incomplete data.
  - Designing a responsive GUI layout that works across various screen sizes can be tricky, and implementing robust error messages for better user understanding also requires attention.
  - Additionally, ensuring data persistence and handling potential issues with external dependencies like pandas and Matplotlib posed challenges.

- **Limitations of the Current System**
  - The limitations of the current system include its reliance on a single Excel file, which can lead to data inconsistency or corruption if multiple users try to access it simultaneously.
  - It lacks advanced features like user authentication, role-based access, or real-time data updates. The GUI may not scale well for larger datasets, making navigation difficult.
  - Error handling is basic, and there is no logging for tracking changes. Additionally, the system depends on external libraries like pandas and Matplotlib, which may not work if not properly installed or configured.

Chapter 6

Conclusion and Future Enhancements

6.1 Conclusion

In conclusion, the system provides a simple and effective interface for managing student data using an Excel file. It supports essential operations like adding, updating, deleting, and visualizing data, making it suitable for small-scale use. However, its limitations, such as reliance on a single file and lack of scalability, highlight the need for future improvements, like transitioning to a more robust database and enhancing error handling and user experience.

6.2 Future Enhancements

To further increase the effectiveness and usability of the Excel Data Management System, the following enhancements are recommended:

- Switching from Excel to a database like MySQL or SQLite for better scalability and faster performance.
- Adding user authentication for secure access.
- Enhancing error handling with detailed messages to improve reliability.
- Including advanced data visualisation features for deeper insights.
- Making the interface more user-friendly with better navigation and design.
- Allowing data export in multiple formats, such as CSV or PDF.
- Adding cloud integration for remote access and collaboration.

Chapter 7

References

- **Online Resources**: Chatgpt, Gemini, google

- **Software Tools**: **Python**: Commonly used for creating GUIs, managing Excel data, and plotting graphs.

- **Libraries**:
  - **Tkinter**: used to design the graphical interface.
  - **Matplotlib**: data visualization
  - **Pandas**: For reading, manipulating, and managing Excel data.