Student Information System (SIS) - Documentation

A**pplication** to manage student records, calculate GPAs, and monitor academic performance.

Purpose

- Add, update, or delete student records
- Calculate and view each student's GPA
- Identify the class topper
- View a summary of student performance

Data Structure

- **students Dictionary**:
  Stores all student records with the format:

```
{
  "student_id": {
    "name": "Student Name",
    "marks": [list of marks],
    "gpa": GPA_value
  }
}
```

# Functions Overview

add_student()

- Inputs: Student ID, name, and marks
- Converts marks to a list of integers
- Calculates GPA and adds the student to the dictionary

update_student()

- Updates name and marks for a given student ID

- Recalculates GPA

delete_student()

- Removes a student record using student ID

view_performance()

- Prints a summary of all students and their GPAs

calculate_gpa(marks)

- Calculates the average (GPA) from a list of marks

class_topper()

- Finds and prints the student with the highest GPA

visualize_data()

- Displays GPA for each student (text-based output)

# Menu Options

1. Add Student

2. Update Student

3. Delete Student

4. View Performance Summary

5. Class Topper

6. Visualize Data

7. Exit

# Usage Instructions

1. Run the script.
2. Choose options from the menu by entering the corresponding number.
3. Input requested data (like name, marks) when prompted.
4. Use option 7 to exit.

Expense Tracker - Documentation

This is **application** that helps users record daily expenses, view a summary against predefined budgets, and visualize their spending with a pie chart.

Purpose

- Add expense entries by category
- Compare total spending against set budgets
- View a pie chart of expense distribution

Data Structures

- **expenses** – a list that stores each expense entry as a dictionary:

```
{
  "Food": 500,
  "Transport": 300,
  "Entertainment": 200,
  "Other": 400
}
```

# Function Breakdown

add_expense()

- Takes user input for **category** and **amount**
- Appends a dictionary entry to the expenses list

show_summary()

- Summarizes total spending per category

show_chart()

- Generates a **pie chart** using matplotlib
- Shows how spending is divided across categories
- Only runs if there is data to display

# Menu Options

1. Add Expense

2. Show Summary

3. Show Chart

4. Exit

Example Usage Flow

1. Select **"1. Add Expense"** to log a new expense.
2. Choose **"2. Show Summary"** to see total expenses per category and budget alerts.
3. Pick **"3. Show Chart"** to view a visual pie chart of your expenses.
4. Select **"4. Exit"** to end the program.

COVID Dashboard Application - Documentation

COVID-19 tracking system that allows users to manage and analyze COVID data for different cities.

1. Features

- Add daily COVID-19 data (cases, recoveries, deaths) for a city.
- Analyze cities to determine if they are Low, Medium, or High Risk zones.
- Display trends of cases, recoveries, and deaths for a specific city.
- Predict potential hotspots based on recent case averages.

2. Data Structure

All data is stored in a list of dictionaries . Each dictionary entry contains:

- date: Date of the record
- city: City name
- cases: Number of new cases
- recoveries: Number of recoveries
- deaths: Number of deaths

3. Functions

add_daily_data()

- Prompts user to enter date, city, cases, recoveries, and deaths.
- Adds the entry to the covid_data list.

analyze_risk_zones()

- Calculates total cases per city.

show_trend(city)

- Shows the trend of cases, recoveries, and deaths for a given city.
- Displays entries sorted by date.

predict_hotspots()

- Checks the last 3 data entries for each city.
- If the average of those 3 days is more than 100 cases, it's marked as a potential hotspot.

save_to_file()

- Saves all data entries to a file named covid_data.txt in CSV format.

menu()

- Displays a user-friendly menu to interact with the application.
- Handles user input and routes to the correct function.

# COVID DASHBOARD MENU

1. Add Daily Data

2. Analyze Risk Zones

3. Show Trend for a City

4. Predict Hotspots

5. Save Data to File

6. Exit

Library Management System - Documentation

Library system that allows managing books, issuing and returning them and tracking borrow counts.

1. Features

- Add new books to the collection.
- View all available books.
- Issue books to borrowers.
- Return books and calculate late fines.
- Track and display the most borrowed books.

2. Data Structures Used

- books: A list to store available book names.
- issued: A dictionary to map borrower's name to the book they've borrowed.
- borrow_count: A dictionary to track how many times each book has been borrowed.

3. Functions

add_book()

- Prompts the user to enter a book name.
- Adds the book to the books list.

view_books()

- Displays all currently available books.
- Shows a message if no books are available.

issue_book()

- Asks for the borrower’s name and the book to issue.

return_book()

- Prompts for the borrower's name.
- If the borrower exists in issued, asks for the number of late days.

most_borrowed()

- Displays each book and the number of times it has been borrowed.

# LIBRARY MENU

1. Add Book

2. View Books

3. Issue Book

4. Return Book

5. Most Borrowed Books

6. Export Log

0. Exit

Health Tracker - Documentation

P**ersonal health tracker** helps users monitor daily health metrics and calculate their BMI.

1. Purpose

- Track their **daily health data**
- View a **weekly average report**
- **Calculate BMI**

2. Data Storage

The program stores data using **lists**:

- steps_list: Stores steps walked each day
- sleep_list: Stores sleep hours
- calories_list: Stores calories consumed
-

3. Function Descriptions

add_data()

  - Steps walked
  - Hours slept
  - Calories consumed

bmi_calculator()

- Asks the user to enter:
  - Weight in kilograms
  - Height in meters
- Calculates BMI.

show_report()

- Displays weekly averages for:

    - Steps walked
    - Sleep hours
    - Calories consumed
    - Water intake

main()

- Displays a **menu** and handles user input to call the appropriate functions.

# HEALTH TRACKER MENU

1. Add Today's Data

2. Calculate BMI

3. Show Weekly Report

4. Hydration Reminder

0. Exit