Assignment 2

Student Management System

In this assignment, you will design and implement a Student Management System (SMS) in Python using an Object-Oriented Programming (OOP) approach. The goal is to practice designing classes, managing relationships between them, handling file-based data storage, and building a simple interactive menu system.

Your program should allow users to manage students, subjects, grades, attendance, and generate summary reports, all through a clean, text-based interface.

Project Structure

Create a folder named student_management_system/ with a structure similar to:

```
student_management_system/
 ├── main.py
 ├── models/
 │   ├── student.py
 │   ├── subject.py
 │   ├── record.py
 │  └── manager.py
 ├── data/
 │   ├── students.txt
 │   ├── subjects.txt
 │   ├── enrollments.txt    # format is flexible
 │  └── records.txt       # (or multiple files — your choice)
 └── README.md
```

You may add additional modules if useful, but main.py must remain the entry point.

Core Requirements

1. Classes

The system must be designed around classes. At minimum:

StudentClass
Attributes such as:

- student_id
- name
- section/batch
- num of subjects enrolled

SubjectClass

- subject_code
- subject_name
- credit_hours

RecordClass (grades + attendance for each subject a student is enrolled in)

- subject_code
- list of grades
- attendance count and total classes

SystemManagerClass
Handles:

- adding students
- adding subjects
- enrolling students
- adding grades
- marking attendance
- generating reports
- saving/loading data to text files

2. Required Functionality

Add Students and Subjects

- Add new students with unique IDs
- Add subjects with unique codes

Enrollment

- Enroll a student in one or more subjects
- Automatically create a record for that student–subject pair

Grades

- Add grade entries
- Calculate average grade per subject
- Display grade history

Attendance

- Mark attendance (present/absent)
- Track total classes per subject
- Compute attendance percentage

Reports (Text-Based)

Generate readable reports showing:

- Student details
- Subjects taken
- Grade summaries
- Attendance summaries
- Overall performance snapshot

Menu System

A simple CLI with options such as:

```
1. Add Student

2. Add Subject

3. Enroll Student

4. Add Grade

5. Mark Attendance

6. View Student Report

7. View All Students

8. Exit
```

3. Data Storage (Text Files)

All data must be stored inside the data/ folder using simple, readable text-based files. You may choose any consistent format (line-based, delimiter-based, etc.).

Examples:

students.txt

```
1001 | Alice Johnson | BSCS-3A

1002 | Ahmed Khan | BSSE-2B
```

subjects.txt

```
CS101 | Programming Fundamentals | 3

CS201 | Data Structures | 4
```

records.txt (format up to you):

```
student_id | subject_code | grades=[85,90] | attendance=12/14
```

No JSON, CSV, or database systems are required — only plain text files.

4. Code Quality Requirements

- Proper use of classes, methods, and encapsulation
- Avoid long functions; keep logic inside appropriate class methods
- No global variables for data storage
- Handle missing data files gracefully (create if not present)
- Clean, readable print output
- Use exceptions where needed
- Keep main.py minimal (menu + method calls only)

Implementation Steps

1. Design classes and determine attributes/methods.
2. Implement storage for students, subjects, and records.
3. Build enrollment, grading, and attendance features.
4. Add reporting functions.
5. Implement saving/loading from text files.
6. Build interactive menu.
7. Test with multiple students and subjects.

Optional Bonus Features

(Not required, but may improve your project quality if implemented well)

- GPA calculation
- Student ranking
- Subject-wise statistics
- Ability to update or remove records
- Export individual student reports as text files

Submission Instructions

- Upload the full student_management_system/ project folder to a GitHub repository.
- Only working repositories will be graded.
- Ensure the program runs by executing `python main.py`
- Include a basic README.md explaining:
    - How to run the system
    - Features implemented
    - How data is stored
    - Summary of classes used