STUDENT MARKS MANAGEMENT SYSTEM

SUBMITTED BY: ADRUTHA LEELALAKSHMI

REG.NO: 25BCE10630

COURSE: COMPUTER SCIENCE AND ENGINEERING

SCHOOL: SCOPE

YEAR: 2025

INTRODUCTION

In modern education, managing student academic information is crucial for school operations and planning. Earlier, schools and teachers used to record student marks in registers. This method becomes less effective and more prone to mistakes as the number of students becomes bigger. Keeping manual records can lead to duplicate data, difficulties in retrieving information, loss of records, difficulty in accessing when needed and other situational/ personal errors. Here, computer-based tools that can help with storing, updating, retrieving, and managing student records.

This project, the Student Marks Management System deals with these issues by implementing solution in Python that uses CSV (Comma Separated Values) file handling. Instead of relying on large databases or complex setups, CSV files provide a simple, portable, and easy-to-maintain format.

The system features a menu-driven interface that allows users to perform basic database operations stated above.

Overall, this project shows how the application of Python programming can change a manual academic work into a structured, automated, and efficient system. It also improves the accuracy and accessibility of students' data.

PROBLEM STATEMENT

When an institution manages its data of students (here, marks), it can lead the following issues:

- Redundancy and errors in entered data
- Difficulty in correcting/updating data
- Difficulty in retrieving data

These errors may occur due to the situation a person is while doing the certain work, or generally since as humans, we can't avoid errors.

Introducing a computer-based system can help reduce these errors and also increase the efficiency of doing this work manually by a person.

FUNCTIONAL REQUIREMENTS

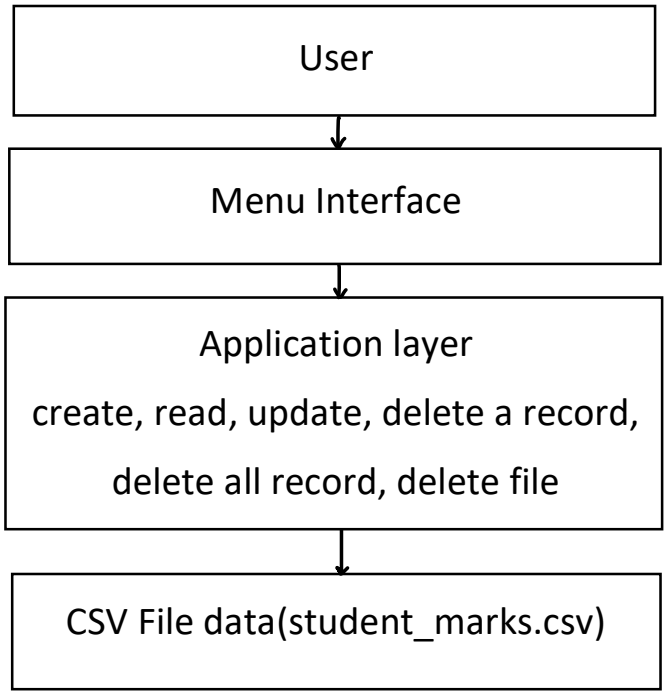
SL. NO	FUNCTION	FUNCTION USE
1	create()	To add and save data of a student in the CSV file
2	read()	To display the added data in the CSV file
3	update()	To make changes in the already existing data in the CSV file
4	delete_record()	To delete the record of a student with their specific roll. number
5	delete_allrecords()	To delete all the records in the CSV file
6	delete_file()	To delete the whole CSV file itself
7	menu()	To implement a menu for the user to access the CSV file

NON-FUNCTIONAL REQUIREMENTS

- **Reusability:** The same code can be re-used for other similar works.
- **Scalability:** The CSV file can store data of many students
- **Performance:** The file can load/display data faster than a human searching manually.
- **Efficiency:** Use of user-defined functions make the code more space efficient.

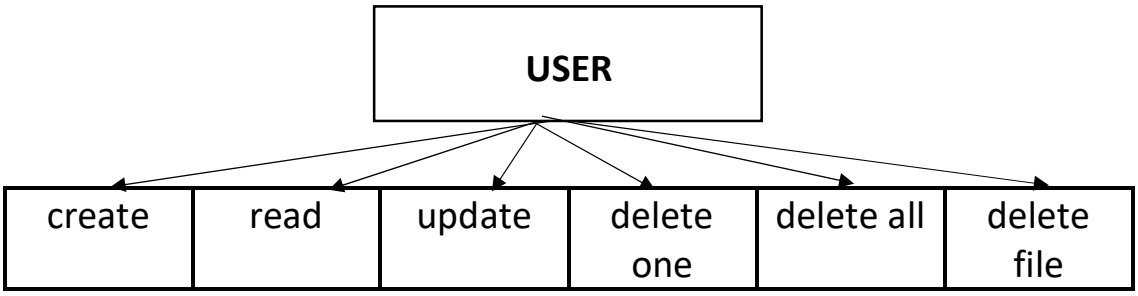
SYSTEM ARCHITECTURE

This project follows an architecture where the user interacts with a menu-driven interface, which triggers different program functions that perform file operations on the CSV database.

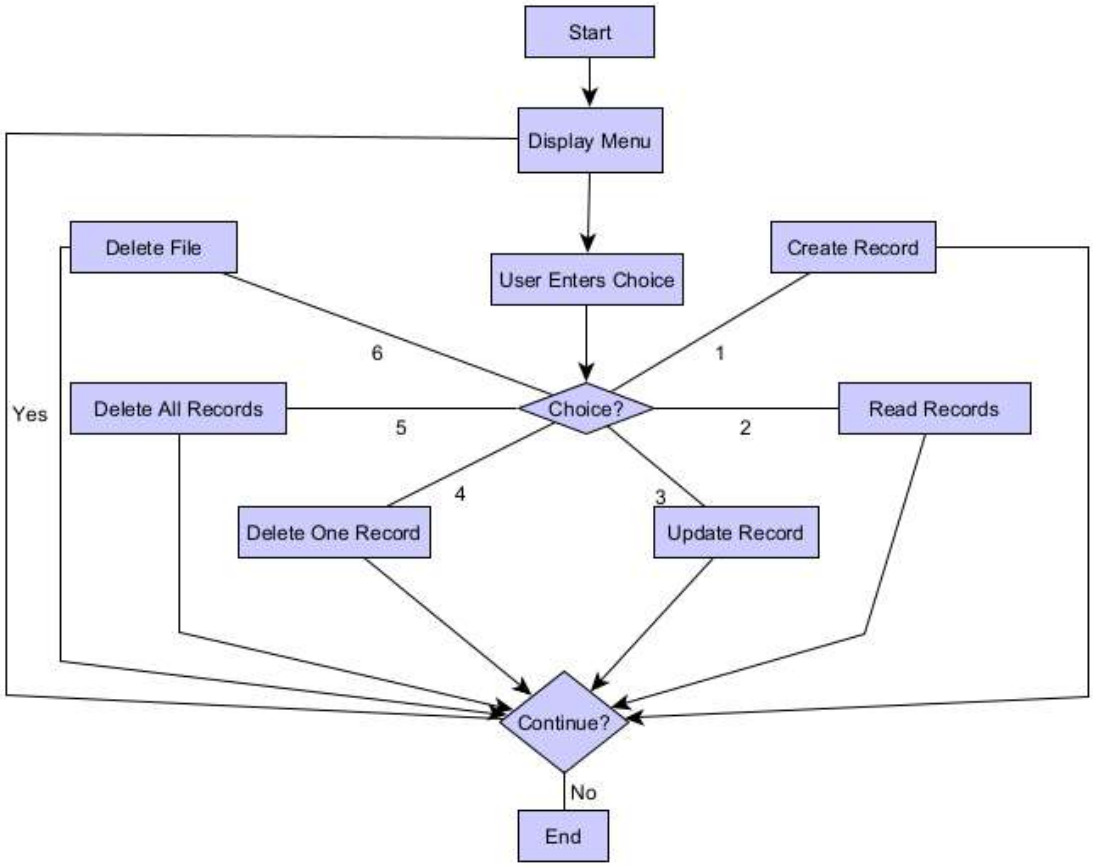


DESIGN DIAGRAMS

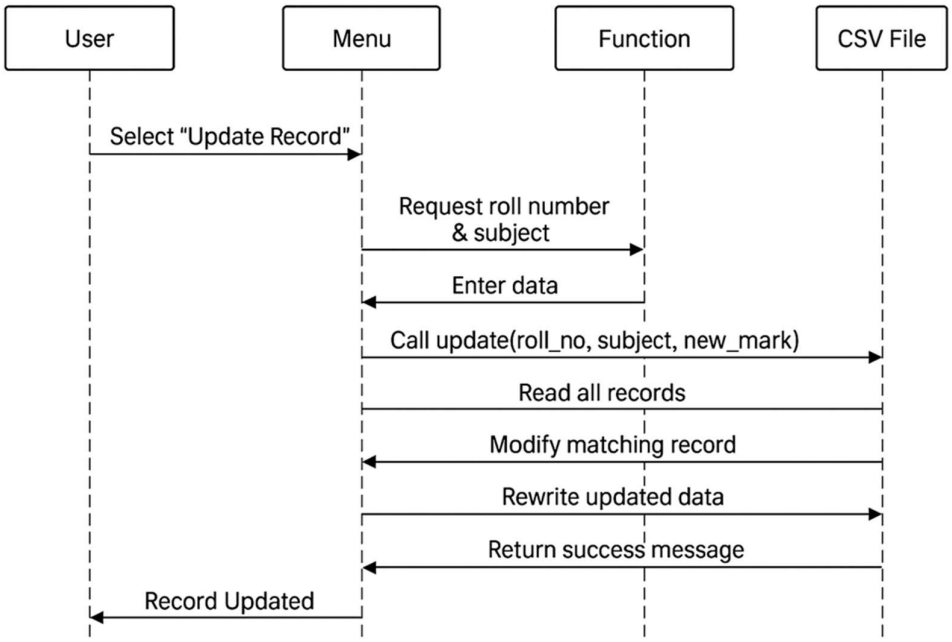
CASE DIAGRAM:



WORKFLOW DIAGRAM:



SEQUENCE DIAGRAM:



DESIGN DECISIONS & RATIONALE

- **Use of CSV to store data**
  - Easy to use, read and edit data
  - Easy to integrate with python
  - Can be converted into excel file

- **Using separate functions**
  - Increases readability of code
  - Makes Debugging easier
  - Makes code re-usable

- **Menu-Driven User interface**
  - Makes applications easier to use for the user
  - Works according to the user's choices

- **Using five predefined subjects**
  - Simplifies data entry
  - Similar to usual school curriculum structure
  - Makes indexing in CSV while updating/deleting

IMPLEMENTATION DETAILS

➢ Programming Language and Tools

- **Language Used:** Python
- **Storage Format:** CSV (Comma Separated Values)
- **Python Modules:**
  - **csv** – Used for reading and writing tabular data to the CSV file.
  - **os** – Used for checking file existence and deleting files safely.

➢ Data Storage

- [Roll No, Name, Physics, Maths, Chemistry, English, Computer Science]

➢ File Handling Logic

- Creating and Appending Records
  - **`create()`** opens the CSV file in append mode ("a") and writes a new row using:
    - **`writer.writerow([Student_roll, Student_name] + All_Marks)`**
- Reading Records
  - **`read()`** opens the file in read mode and prints all rows sequentially. If the file is empty, a message is shown.
- Updating a Record
  - The update process uses a **read–modify–rewrite workflow**:

    1. Read all rows from the CSV file
    2. Modify the matching row in memory
    3. Rewrite the entire file with updated data

- Deleting Records

1. Deletion works similarly:

   1. **`delete_record()`** removes only the matching row
   2. **`delete_allrecords()`** clears the file contents
   3. **`delete_file()`** completely removes the CSV file after verifying it exists to avoid runtime errors

CODE/PROGRAM

```
import csv
import os
def create():
    #data structure
    #[[rollno,name,mark1,mark2,mark3,mark4,mark5]]
    f=open("student_marks.csv","a",newline='')
    writer=csv.writer(f)
    Student_name=input("Enter name of the student")
    Student_roll=input("enter roll.no")
    #Lets say there are 5 subjects
    All_Marks=[]
    Subjects=['Physics','Maths','Chemistry','English','Comp.Science']
    for i in range(len(Subjects)):
        mark=input(f"Enter mark of subject {Subjects[i]}")
        All_Marks.append(mark)
    writer.writerow([Student_roll, Student_name] + All_Marks)
    f.close()
    print('Data added')
def read():
    f=open('student_marks.csv','r',newline='')
    reader=csv.reader(f)
    empty = True
    for data in reader:
        print(data)
        empty = False
    if empty:
        print('empty file')
    f.close()
```

```
def update(roll_no,subject,new_mark):
    f=open('student_marks.csv','r',newline='')
    reader=csv.reader(f)
    data=[]
    for i in reader:
        if i[0]==roll_no:
            i[subject]=new_mark
        data.append(i)
    f.close()
    f1=open('student_marks.csv','w',newline='')
    writer=csv.writer(f1)
    writer.writerows(data)
    f1.close()
    print('updated')
def delete_record(roll_no):
    f=open('student_marks.csv','r',newline='')
    reader=csv.reader(f)
    data=[]
    for i in reader:
        if i[0]!=roll_no:
            data.append(i)
    f.close()
    f=open('student_marks.csv','w',newline='')
    writer=csv.writer(f)
    writer.writerows(data)
    f.close()
    print('deleted')
```

```
def delete_allrecords():
    f=open('student_marks.csv','w')
    f.close()
def delete_file():
    if os.path.exists("student_marks.csv"):
        os.remove("student_marks.csv")
        print("File deleted")
    else:
        print("File does not exist")

def menu():
    print("1. Create/Add")
    print("2. Read")
    print("3. Update data")
    print("4. Delete a record")
    print("5. delete all records")
    print("6. Delete record")
    c=int(input("enter choice"))
    if c==1:
        create()
    elif c==2:
        read()
    elif c==3:
        roll_no=input("enter roll no")
        print('1.Physics')
        print('2.Maths')
        print('3.Chemistry')
        print('4.English')
        print('5. Computer science')
        subject=int(input("enter subject code"))+1
        new_mark=input("enter new marks")
        update(roll_no,subject,new_mark)
    elif c==4:
        roll_no=input('enter rollno')
        delete_record(roll_no)
    elif c==5:
        delete_allrecords()
    elif c==6:
        delete_file()
    else:
        print('enter valid choice')

while True:
    menu()
    choice=input("do you want to continue, yes/no")
    if choice.lower()=='no':
        print('terminated')
        break
    else:
        continue
```

RESULTS

```
1. Create/Add
2. Read
3. Update data
4. Delete a record
5. delete all records
6. Delete record
enter choice2
['2', 'B', '36', '37', '40', '37', '39']
['3', 'C', '35', '35', '38', '40', '39']
['1', 'A', '40', '45', '46', '50', '49']
do you want to continue, yes/noyes
1. Create/Add
2. Read
3. Update data
4. Delete a record
5. delete all records
6. Delete record
enter choice1
Enter name of the studentD
enter roll.no4
Enter mark of subject Physics36
Enter mark of subject Maths38
Enter mark of subject Chemistry39
Enter mark of subject English40
Enter mark of subject Comp.Science45
Data added
```

```
do you want to continue, yes/noyes
1. Create/Add
2. Read
3. Update data
4. Delete a record
5. delete all records
6. Delete record
enter choice3
enter roll no2
1.Physics
2.Maths
3.Chemistry
4.English
5. Computer science
enter subject code1
enter new marks42
updated
do you want to continue, yes/noyes
1. Create/Add
2. Read
3. Update data
4. Delete a record
5. delete all records
6. Delete record
enter choice4
enter rollno1
deleted
```

```
do you want to continue, yes/noyes
1. Create/Add
2. Read
3. Update data
4. Delete a record
5. delete all records
6. Delete record
enter choice2
['2', 'B', '42', '37', '40', '37', '39']
['3', 'C', '35', '35', '38', '40', '39']
['4', 'D', '36', '38', '39', '40', '45']
do you want to continue, yes/noyes
1. Create/Add
2. Read
3. Update data
4. Delete a record
5. delete all records
6. Delete record
enter choice5
do you want to continue, yes/noyes
1. Create/Add
2. Read
3. Update data
4. Delete a record
5. delete all records
6. Delete record
enter choice2
empty file
```

```
do you want to continue, yes/noyes
1. Create/Add
2. Read
3. Update data
4. Delete a record
5. delete all records
6. Delete record
enter choice6
File deleted
do you want to continue, yes/nono
terminated
```

TESTING APPROACH

The testing focused on seeing whether the code works correctly, reliably and as expected.

- **Black Box testing:** Tests were performed by giving inputs and validating outputs without inspecting internal logic. This aligned with end-user's perspective.
- **Functional Testing:** Each function (Create, Read, Update, Delete, etc.) was tested individually to verify correct operation

- **Code was tested to see:**
    1. Student records were correctly stored in CSV file
    2. Data is properly displayed
    3. Data is properly updated
    4. Various delete operations work accurately as in intended.

- **Test Environment**
    - **Operating System:** Windows
    - **Python Version:** Python 3.13
    - **Files Used:** students_marks.csv

CHALLENGES FACED

- Designing the Menu Workflow
  Creating a smooth, continuous menu-driven interface that:

  - Processes user choices
  - Executes the correct function
  - Returns to the main menu
  - Allows clean exit

  required careful looping logic

- Documentation
  Translating the working project into:

  - System diagrams
  - Architecture explanations
  - Reports and formal documentation

  was time-consuming but helped in better understanding the software development process end-to-end.

- Code Modularity
  Writing the program in a fully modular manner with separate functions for create, read, update, and delete required planning to avoid repeated code and maintain readability.

- Updating in CSV
  Since CSV does not support direct in-place editing, understanding and implementing the **read–modify–rewrite mechanism** had to be used

LEARNINGS & KEY TAKEAWAYS

1. **Application of file handling:** The project demonstrated how Python can be used to store data permanently using CSV files.

2. **Modular programming:** Modular programming increases readability and makes debugging easier

3. **Understanding CRUD operations**

4. **Understanding basics of software developing**

5. **Documentation and its role**

FUTURE ENHANCEMENTS

1. **Graphical User Interface**
   Program can be upgraded to console-based to GUI based application using python libraries like Tkinter, Kivy etc

2. **Integration of Databases**
   Databases like SQLite, or MySQL can be used to store the data.

3. **More features**
   Features like average of class, report card generation, Pass/Fail checking etc can be added.

4. **Advanced Search and Storing**
   Searching based on both name and roll number can be added. And feature to delete copies/duplicates can also be added.

5. **Data Visualization**
   Libraries such as Matplotlib can be used to visualize the data of the class.

REFERENCES

1. http://w3schools.com/python/ref_module_csv.asp
2. https://ncert.nic.in/textbook.php?lecs1=2-13
3. https://www.geeksforgeeks.org/python/working-csv-files-python/
4. https://www.programiz.com/python-programming/reading-csv-files

Last referred on 22nd November 2025