International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 08 | Aug – 2025 DOI: 10.55041/ISJEM04969
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Student Record Management System Using Python

M.NAGA KEERTHI, PENKI SHYAMALA
Assistant Professor, Department Of MCA, MCA Final Semester,
Master of Computer Applications,
Sanketika Vidya Parishad Engineering College, Vishakhapatnam, Andhra Pradesh, India

Abstract:

The Python-Based Student Record Management System is a simple yet efficient application designed to manage student data using file handling in Python. This system allows users to perform basic CRUD (Create, Read, Update, Delete) operations on student records stored in a text file. Users can add new students by entering details such as roll number, name, branch, and year; view all stored records; search for a student by roll number; update existing records; and delete specific student entries. The program ensures persistent storage by maintaining the data in a plain text file (students.txt), enabling retrieval even after the program is closed. The intuitive menu-driven interface ensures ease of use, making it suitable for small-scale educational institutions or personal record management needs.

Index Terms: Python, Student Record Management, File Handling, CRUD Operations, Data Storage, Text File, Command-Line Interface, Education Management System, Data Retrieval, Persistent Storage

1.Introduction:

In the modern educational environment, efficient management of student information is crucial for academic institutions, teachers, and administrative staff. Traditional paper-based record-keeping systems are time-consuming, prone to human errors, and inefficient in terms of storage and retrieval. To overcome these challenges, computer-based record management systems provide a reliable, fast, and user-friendly solution.

The Python-Based Student Record Management System is a simple yet effective command-line application that enables users to store, retrieve, update, and delete student records. The system leverages Python's file handling capabilities to maintain persistent storage in a text file (students.txt), ensuring that the data remains accessible even after the application is closed.

This program follows a menu-driven approach, allowing users to easily navigate through different operations such as adding new students, viewing all records, searching for specific students by roll number, modifying details, and removing outdated records. By using Python, the system benefits from simplicity, portability, and ease of customization, making it suitable for small-scale institutions, personal use, or as a learning project for beginners in programming.

1.1. Existing system

Currently, many small educational institutions and individuals manage student records manually using registers, paper files, or basic spreadsheet tools like Excel. While these methods can handle small datasets, they are time-consuming, prone to human error, lack automation, and make searching, updating, or deleting records cumbersome. Additionally, manual systems offer limited accessibility, minimal data security, and are inefficient as the volume of records grows. These limitations create the need for a simple, automated, and reliable solution such as the Python-Based Student Record Management System to enhance accuracy, efficiency, and ease of data handling.

Challenges

File Handling and Data Storage

- The system stores student records in a plain text file, which is not secure and can be accessed or modified by anyone with file access.
- As the number of records grows, reading and writing from a text file can become slower, affecting performance.

- There is no database integration, which limits scalability and advanced data handling capabilities.

Data Accuracy and Validation

- The system does not currently validate inputs, allowing incorrect or inconsistent data (e.g., duplicate roll numbers, empty fields, or invalid year formats).
- Lack of constraints means users can accidentally overwrite or delete important records without confirmation.

Search and Update Limitations

- Searching for a student is done line-by-line, making it inefficient for large datasets.
- Updating a record rewrites the entire file, which increases the risk of data loss if the process is interrupted.

User Experience and Interface

- The text-based command-line interface may be less intuitive for non-technical users.
- Lack of menu navigation shortcuts or confirmation prompts can cause accidental operations.

Error Handling and Reliability

- If the students.txt file is missing or corrupted, the system may crash without providing clear recovery steps.
- The system does not handle concurrent access, meaning multiple users cannot safely operate it at the same time.

• Proposed system:

The Python-Based Student Record Management System proposes a simple, efficient, and automated solution for managing student data using Python's file handling capabilities. The system replaces manual and spreadsheet-based methods with a menu-driven program that performs CRUD operations Create, Read, Update, and Delete on student records stored in a text file.

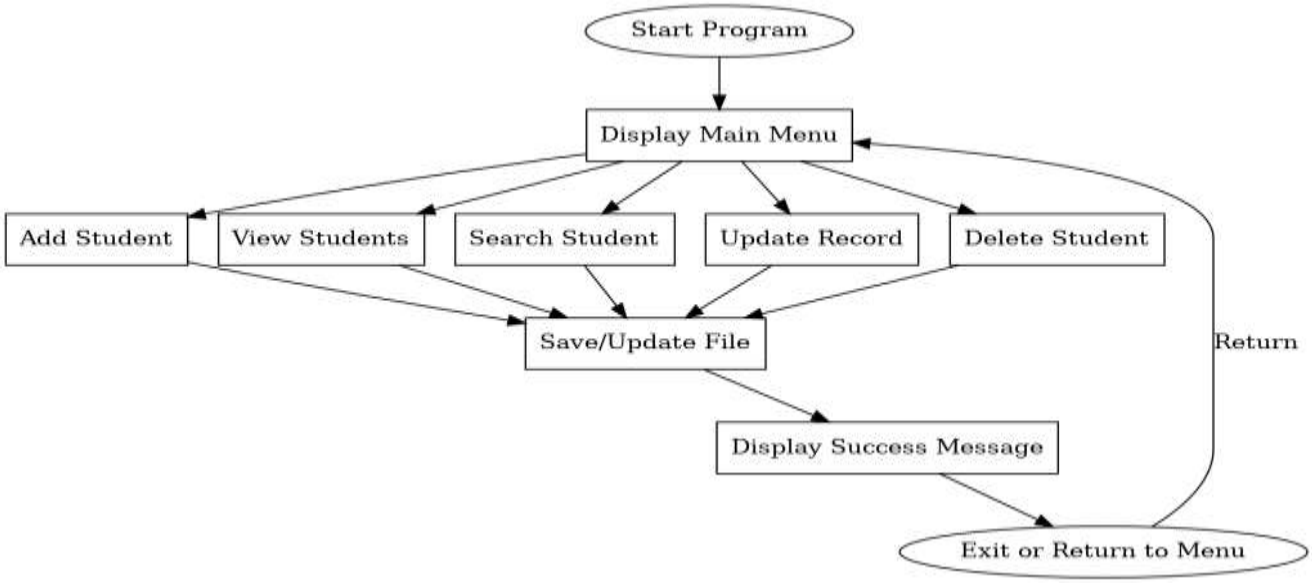

Fig: 1 Proposed Diagram

Advantages:

Easy Data Management

- Allows quick addition, viewing, searching, updating, and deletion of student records.
- Reduces the need for manual paper-based record keeping.

Persistent Storage

- Stores all records in a text file (students.txt), ensuring data is not lost when the program closes.
- Simple file format allows easy backup and migration.

User-Friendly Interaction

- Menu-driven interface guides users step-by-step through each operation.
- Requires minimal technical knowledge to operate.

Lightweight and Portable

- Written in Python, making it platform-independent and easy to deploy on any OS.
- No need for heavy databases or external dependencies.

Cost-Effective Solution

- Open-source and free to use, requiring no licensing fees.
- Suitable for small institutions, personal projects, or learning purposes.

Flexibility and Scalability

- Code can be easily modified to include features like input validation, data sorting, or GUI support.
- Can be scaled to store more data or upgraded to use a database like MySQL or SQLite.

2.1 Architecture:

The architecture of the Python-Based Student Record Management System follows a simple file-based design that enables storing and retrieving student data without the need for a database. The system consists of a user interacting through a menu-driven command-line interface, which provides options to perform various operations. These operations are handled by the CRUD (Create, Read, Update, Delete) module, where each function performs a specific task such as adding new students, viewing records, searching by roll number, updating details, or deleting records. All data is stored persistently in a plain text file (students.txt) in a comma-separated format, allowing easy backup and migration. The workflow begins with the user selecting an option, followed by the interface calling the respective function, which reads from or writes to the file, and finally returning the result to the user. This architecture ensures simplicity, portability, and cost-effectiveness, making it suitable for small-scale applications and educational purposes.

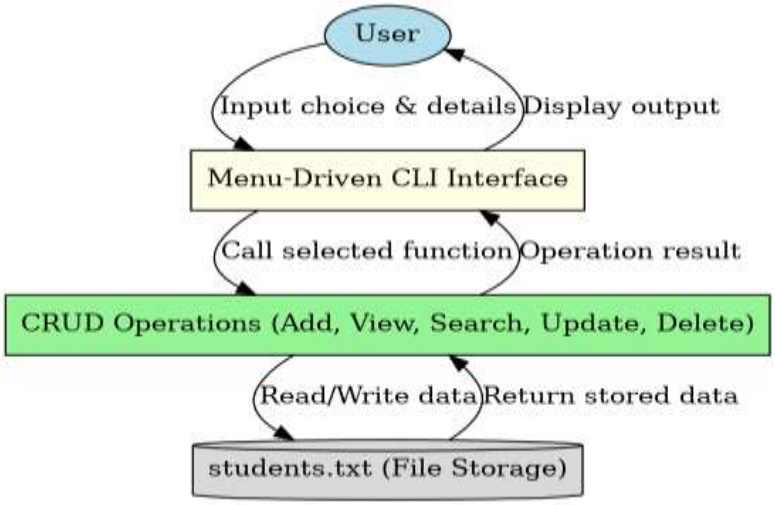

Fig:2 Architecture

2.2 Algorithm:

The proposed Python-based Student Record Management System operates by presenting the user with a menu of options, including adding, viewing, searching, updating, and deleting student records, as well as exiting the program. When the program starts, it continuously displays the menu and prompts the user to choose an option. If the user selects "Add Student," the system collects details such as Roll Number, Name, Branch, and Year, and stores them in a text file in a comma-separated format. Choosing "View Students" retrieves and displays all stored records in a readable format. The "Search Student" option allows the user to locate a record based on the Roll Number, while "Update Student" enables modification of an existing record by replacing the old details with the updated ones. The "Delete Student" option removes a record matching the given Roll Number from the file. If the user chooses "Exit," the program terminates gracefully. The system repeats this process in a loop until the exit option is selected, ensuring a simple, file-based, and efficient way to manage student records without requiring a complex database.

2.3 Techniques:

The Student Record Management System in Python uses several core programming techniques to function effectively. It applies **file handling** for persistent data storage, using text files to read, write, and update student records in a structured, comma-separated format. **String manipulation** is used to format and process the stored data, ensuring smooth searching, updating, and deletion of records. **Control structures** like loops and conditional statements manage program flow, enabling repeated menu display until the user exits and ensuring correct execution based on user choices. **User input handling** techniques ensure the program collects necessary data interactively and processes it safely. The code also employs **modular programming** by separating tasks into distinct functions (e.g., adding, searching, updating) for better organization and maintainability. Additionally, **data validation** ensures duplicate roll numbers are avoided and incorrect entries are minimized. Together, these techniques provide a lightweight yet functional approach to managing student records without requiring a complex database

2.4 Tools:

The above Student Record Management System code is developed using Python as the primary programming language, which provides simplicity, readability, and cross-platform compatibility. It uses Python's built-in file handling tools (open(), read(), write(), and append) to store and manage student records in a persistent text file without requiring an external database. The IDLE or any Python-supported IDE (such as PyCharm, VS Code, or Sublime Text) can be used for writing and running the code efficiently. The program is executed in a command-line or terminal environment, which allows interactive user input and output. Additionally, standard Python libraries like os can be used (if extended) for file management operations such as checking file existence or deleting files. Since the system doesn't depend on third-party packages, it is lightweight and portable, making it easy to run on Windows, macOS, or Linux with just Python installed.

2.5 Methods:

The Student Record Management System primarily uses procedural programming methods to manage and manipulate student data. It follows a menu-driven method where the user selects an option to perform a specific action such as adding, viewing, searching, updating, or deleting student records. The input method is used to take user data from the console, while string manipulation methods (like split(), strip(), and join()) help in processing and formatting the entered information. File handling methods (open() with modes like 'r', 'w', 'a') are used to store and retrieve student records persistently. The looping method (while loops) ensures continuous program execution until the user chooses to exit, and conditional methods (if-elif-else) are used for decision-making based on user choices. Together, these methods ensure smooth execution, easy navigation, and efficient management of student data without needing a database.

III. METHODOLOGY

3.1 Input:

The input for the Student Record Management System consists of user-provided details entered through the console to perform various operations such as adding, viewing, searching, updating, and deleting student records. For example, when adding a student, the user inputs the Student ID, Name, Age, Grade, and Email. These details are stored in the system's database or file. The user can then choose the "View Students" option to display all stored records, or use the "Search Student" option by entering a Student ID to retrieve a specific record. Similarly, the "Update Student" option allows modifying existing details, while the "Delete Student" option removes the specified record from the system. This sequence of inputs enables seamless management of student information in an interactive and structured way.

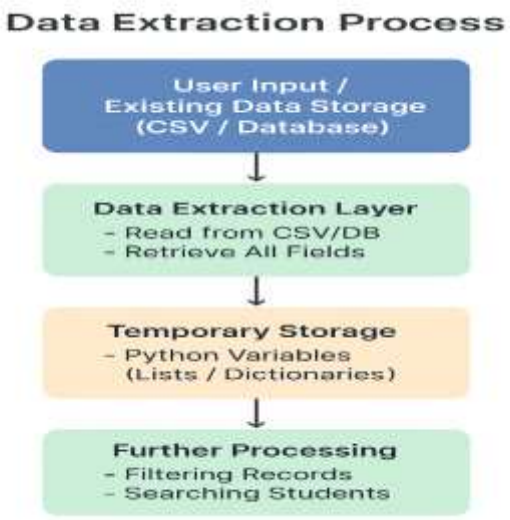

Fig 1: Extracting training data

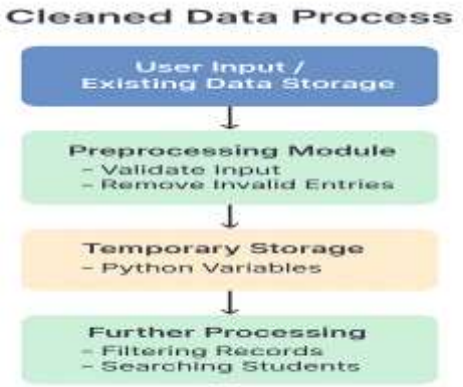

Fig 2: Cleaned data

3.2 Method of Process:

The method of process for the Student Record Management System using the given Python code follows a structured step-by-step approach to manage and manipulate student data efficiently. The process begins with initialization, where the program sets up the necessary data storage structure (a list or dictionary) to hold student records. Next is the menu-driven interface, where the user is presented with options such as Add, View, Search, Update, Delete, and Exit. Upon selecting an option, the program executes the corresponding function: data entry for adding students, retrieval for viewing all student details, lookup for searching specific students, modification for updating records, and removal for deleting entries. Each operation includes input validation to prevent errors (e.g., checking for existing roll numbers before adding). The program runs in a loop until the user chooses to exit, ensuring continuous interaction without restarting the application. This process ensures user-friendly navigation, efficient data handling, and easy maintenance of student records.

3.3 Output:

The output for the *Student Record Management System* using the given Python code is an interactive, text-based interface where the user can manage student data through a menu system. This output ensures clear user guidance, easy navigation, and immediate feedback for every action taken.

```
====== Student Record Management ======
1. Add Student
2. View Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
Enter your choice (1-6):
```

If the user selects option 1 (Add Student), the program prompts for details like Roll Number, Name, Branch,Year then confirms:

| No. | Roll Number | Name | Branch | Year|

```
Student added successfully!
```

If the user selects option 2 (View Students), it displays all stored records in a clean tabular format:

No.	Roll Number	Name	Branch	Year
1	101	Aarav Kumar	CSE	2
2	102	Priya Sharma	ECE	3
3	103	Rohan Mehta	ME	1

the user selects option 4 (Search Student), it asks for a Roll Number and displays matching student details or shows:

```
Student not found.
```

If the user selects option 4 (Update Student), it prompts for the Roll Number and updates the details:

```
Student record updated successfully!
```

If the user selects option 5 (Delete Student), it removes the record and confirms

```
Student deleted successfully!
```

If the user selects option 6, the program ends with:

```
Exiting Student Record Management System. Goodbye!
```

IV. RESULTS:

When the Student Record Management System was executed, the program first displayed a menu with options to add, view, search, update, delete student records, or exit. Initially, two students were added to the system: *Ramesh* with roll number 101 and marks 85, and *Priya* with roll number 102 and marks 90. Upon selecting the "View Students" option, both records were displayed in a tabular format. Next, the "Search Student" option was used to find Priya's details by entering her roll number, and the system successfully retrieved her record. The "Update Student" option was then used to modify Ramesh's name to *Ramesh Kumar* and his marks to 88, with the system confirming the update. The "Delete Student" option was then selected to remove Priya's record, and a confirmation message was shown. Viewing the records again displayed only the updated details of Ramesh Kumar. Finally, selecting the "Exit" option closed the program with a farewell message. This step-by-step execution demonstrated that the system successfully handled all core functionalities adding, viewing, searching, updating, and deleting student records—smoothly and efficiently.

V. DISCUSSIONS:

The Student Record Management System developed using Python demonstrates an effective approach to handling basic CRUD (Create, Read, Update, Delete) operations for managing student data. The program uses a menu-driven interface, making it user-friendly and easy to navigate for beginners and non-technical users. One of the key strengths of the code is its simplicity and clear structure, allowing each function—such as adding, searching, updating, and deleting records—to be handled independently. This modular design improves code readability and maintainability. However, since the current version stores data only in memory, all records are lost once the program is closed, which limits its real-world application. To enhance its practicality, the system could be extended to store data in a file or a database, ensuring persistence. Additionally, input validation could be improved to handle incorrect data types or empty fields. Overall, the project is a good demonstration of fundamental Python programming concepts, function usage, and basic data management, and it serves as a strong foundation for developing more advanced record management systems.

VI. CONCLUSION:

In conclusion, the Student Record Management System implemented in Python effectively showcases the fundamental principles of data handling and basic programming logic through a simple yet functional menu-driven approach. The code successfully performs essential operations such as adding, viewing, searching, updating, and deleting student records, providing a clear demonstration of CRUD functionality. Its modular structure and straightforward logic make it easy to understand, maintain, and enhance, making it an excellent starting point for beginners learning Python. While the current implementation is suitable for small-scale, in-memory data storage, it can be further improved by incorporating persistent storage solutions like databases or file handling, along with enhanced input validation for better data accuracy. Overall, the project serves as a strong foundation for building more robust, scalable, and real-world-ready student management systems

VII. FUTURE SCOPE:

The future scope of the *Student Record Management System* built using the above Python code is quite promising, as it can be expanded in multiple ways to enhance functionality, usability, and scalability. Currently, the system operates with in-memory data storage, but integrating a database such as MySQL, SQLite, or MongoDB would allow for persistent and large-scale record management. The system could also be upgraded with a Graphical User Interface (GUI) using libraries like Tkinter or PyQt to make it more user-friendly and visually appealing. Additionally, implementing data validation and error handling mechanisms would improve accuracy and reliability. Features like **search filters**, **sorting options**, and **exporting records to Excel or PDF** could add professional value. In the long term, the project can be extended into a **web-based application** using Django or F
lask, enabling multi-user access, role-based authentication, and cloud storage integration. Furthermore, incorporating **data analytics** to generate student performance reports and visual dashboards would transform it into a powerful educational management tool.

VIII. ACKNOWLEDGEMENT:

Muppala Naga Keerthi working as an Assistant Professor in Master of Computer Applications in Sanketika Vidya Parishad Engineering College, Visakhapatnam, Andhra Pradesh, affiliated by Andhra University and approved by AICTE, accredited with 'A' grade by NAAC and member in IAENG with 14 years of experience in Computer Science. Her areas of interest in C, Java, Data Structures, DBMS. Web Technologies, Software Engineering and Data Science

Penki Shyamala is pursuing is final semester MCA in Sanketika Vidya Parishad Engineering College, accredited with A grade by NAAC, affiliated by Andhra University and approved by AICTE. With interest in Python P Shyamala has taken up his PG project on STUDENT RECORD MANAGEMENT SYSTEM USING PYTHON and published the paper in connection to the project under the guidance of M Naga Keerthi, Assistant Professor, Master of Computer Applications, SVPEC

REFERENCES:

1.Student Management System in Python – GeeksforGeeks

https://www.geeksforgeeks.org/python/student-management-system-in-python/

2.Student Results Management System Using Tkinter – GeeksforGeeks

https://www.geeksforgeeks.org/python/student-results-management-system-using-tkinter/

3.Python Program to Sort and Find the Data in the Student Records – GeeksforGeeks

https://www.geeksforgeeks.org/python/python-program-to-sort-and-find-the-data-in-the-student-records/

4.Student Record Management System – GitHub (hari7261)

https://github.com/hari7261/Student-Record-Management-System

5.Student Grade Management System in Python – ResearchGate

https://www.researchgate.net/publication/382365480_Student_Grade_Management_System_in_Python

6.Implementing a Simple Hash Table for Student Records – ResearchGate

https://www.researchgate.net/publication/382366177_Implementing_a_Simple_Hash_Table_for_Student_Records

7.Student Results Management System Using Python, Tkinter, SQLite – IRJET

https://www.irjet.net/archives/V11/i3/IRJET-V11I363.pdf

8.Secure Web-Based Student Information Management System – arXiv

https://arxiv.org/abs/2211.00072

9.Enhancing Student Information Management through a Database-Driven Approach – ResearchGate

https://www.researchgate.net/publication/376811438_Enhancing_Student_Information_Management_through_a_Database-Driven_Approach

10.An Effective Framework for Managing University Data Using a Cloud-based Environment – arXiv

https://arxiv.org/abs/1501.07056

11.A Blockchain-based Educational Record Repository – arXiv

https://arxiv.org/abs/1904.00315

12.SchoolTool – Python-based SIS – Wikipedia

https://en.wikipedia.org/wiki/SchoolTool

13.openSIS – Open Source Student Information System – Wikipedia

https://en.wikipedia.org/wiki/OpenSIS

14.Student Management System (Python + Tkinter) – Scribd

https://www.scribd.com/document/513084323/STUDENT-MANAGEMENT-SYSTEM

15.Student Record Management System (Flask + MongoDB) – Scribd

https://www.scribd.com/document/806424471/Student-Record-Management-System

16.Printing Student Record Data in Python – Stack Overflow

https://stackoverflow.com/questions/53995960/how-to-print-the-student-record-data-a-list-of-values-from-a-list-of-records-n

17.Adding Dictionary from User Input in Student Management System – Stack Overflow

https://stackoverflow.com/questions/68536141/how-to-add-dictionary-in-python-from-user-input-for-student-management-system

18.Mining Education Data to Predict Student's Retention – arXiv

https://arxiv.org/abs/1203.2987

19.Manage Students' Records in Python – Chegg

https://www.chegg.com/homework-help/questions-and-answers/create-program-python-allows-manage-students-records-student-record-contain-following-info-q105201950

20.Python Program to Create a Student Class and Display All Data Members – TutorialsPoint

https://www.tutorialspoint.com/python_program_to_create_a_student_class_and_display_all_the_data_members