**Project Report on Student Database Management System Using C**

A Learning Project:  
 Submitted as part of the **Bachelor of Technology (B.Tech.) in Computer Science and Engineering**, this project brings theory to practice through hands-on coding.

We developed a **Student Database Management System using C**, integrating **file handling, data structures, modular programming, and user authentication** to efficiently manage student records, including **projects, internships, fees, and messaging**.

With a **menu-driven interface, input validation, role-based access control, and data persistence**, this system ensures **security, accuracy, and reliability**. This project highlights our ability to build **structured, scalable, and industry-relevant C programs**.

|  |  |
| --- | --- |
| Course | Employment Enhancement Program |
| Program | C-Programming |
| Batch | EEPCVR 06 |
| Group Name | Exception Elites |
| Group | 15 |



**Under The Guidance of Mr. Moshin Khan**



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

**C. V. Raman Global University, Odisha, Bhubaneswar**

PIN -752054, India.

TEAM MEMBERS

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name |  | Debadatta Rout |
| Clg Reg No. |  | 2201020036 |
| Cranes Reg No |  | CL20250106019416120 |
| Email |  | 2201020036@cgu-odisha.ac.in |
|  | Name |  | **Smrutirekha Sethi** |
| Clg Reg No. |  | 2201020023 |
| Cranes Reg No |  | CL20250106019411115 |
| Email |  | 2201020023@cgu-odisha.ac.in |
|  | Name |  | **Amit Ranjan Bastia** |
| Clg Reg No. |  | 2201020054 |
| Cranes Reg No |  | CL20250106019421125 |
| Email |  | 2201020054@cgu-odisha.ac.in |
|  | Name |  | **Hruddhinath Nayak** |
| Clg Reg No. |  | 2201020456 |
| Cranes Reg No |  | CL2025010601947211 |
| Email |  | 2201020456@cgu-odisha.ac.in |
|  | Name |  | **Hussain Mehwish Tanweer** |
| Clg Reg No. |  | 2201020457 |
| Cranes Reg No |  | CL2025010601947312 |
| Email |  | 2201020457@cgu-odisha.ac.in |

**CERTIFICATE FROM SUPERVISOR**

This is to certify that the **Project Report** entitled **“Student Database Management System Using C”**, submitted in partial fulfilment of the requirements for the **Employment Enhancement Program in C** from **Cranes Varsity, Odisha**, during the year **2025**, By Group 15 ,Group Name – Exception Elites successfully done a genuine record of the Bonafide work carried out under the guidance and supervision of **Mr. Moshin Khan.**

The project is a part of their academic curriculum and has been carried out with dedication, sincerity, and hard work.

The project demonstrates a thorough understanding of programming concepts, including **file handling**, **data structures**, **input validation**, and **modular programming**, implemented using the **C programming language**. The system developed is a comprehensive solution for managing student-related data, including academic records, projects, internships, and fee details, with a user-friendly interface and role-based access control.

We hereby confirm that the work presented in this project is original and has not been copied or reproduced from any other source. The students have adhered to the highest standards of academic integrity throughout the development process.

We wish them success in their future endeavors and hope this project serves as a stepping stone for further innovation and learning.

**Trainer-** **Mr. Moshin Khan**



**ACKNOWLEDGEMENT**

We would like to express our heartfelt gratitude to all the individuals who contributed to the development and successful completion of this project, **“Student Database Management System Using C”.** Their expertise, insights, and dedication have been invaluable in making this a meaningful and enriching learning experience.

A special thanks to our mentor, of **Mr. Moshin Khan**, for his invaluable guidance, support, and direction throughout this journey. His expertise has played a crucial role in shaping our understanding and implementation of Java concepts.

We also extend our sincere appreciation to our instructors, content creators, and reviewers for their tireless efforts in providing us with the knowledge and resources needed to excel in this project. Additionally, we acknowledge the encouragement and support from our colleagues and mentors, whose motivation has been instrumental in our learning process.

Finally, we are deeply grateful to **Cranes Varsity** for offering this **Employment Enhancement Program in C**, equipping us with industry-relevant skills and hands-on experience.

**Thank you.**

**Declaration**

We, the undersigned, hereby declare that the project titled **"Academix Hub - Student Database Management System"** is our original work and has been developed as part of our academic curriculum. This project has been undertaken with the aim of enhancing our understanding of programming concepts and their practical applications.

We affirm that the work presented in this report has not been copied or reproduced from any other source, except where explicitly acknowledged through proper citations and references. All efforts have been made to ensure the authenticity and originality of the content, code, and documentation included in this project.

We take full responsibility for any errors or omissions that may be present in this work and are committed to upholding the highest standards of academic integrity.

* Group Name: Exception Elites
* Group Number- 15
* Program: C-Programming

**ABSTRACT**

**Academix Hub - Student Database Management System**

**Academix Hub** is a **Student Database Management System (SDMS)** designed to streamline and automate the management of student-related data in educational institutions. Built using the **C programming language**, this system provides a robust and user-friendly platform for administrators and students to efficiently handle academic records, projects, internships, fee details, and communication.

The system features **role-based access control**, allowing administrators to add, modify, delete, and search student records, while students can view their academic details, check messages, and track fee payments. Key functionalities include **file handling** for data persistence, **input validation** for data integrity, and a **menu-driven interface** for seamless navigation. Additionally, the system supports the management of **projects**, **internships**, and **college fees**, making it a comprehensive solution for academic institutions.

**Academix Hub** emphasizes **modular programming**, ensuring code reusability and maintainability. It addresses the limitations of manual record-keeping by providing a centralized, efficient, and error-free platform for managing student data. This project demonstrates the practical application of programming concepts like **data structures**, **file handling**, and **user authentication**, making it a valuable tool for modern educational administration.

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| Serial No. | Title | Page No. |
|  | Introduction | 7 |
|  | System Analysis | 8 |
|  | System Design | 8-9 |
|  | Data Flow Diagram | 10 |
|  | Flowchart | 11 |
|  | Terminal visual View | 12 |
|  | Implementation | 12-16 |
|  | Explanation of our Project | 16-18 |
|  | Snapshots of our Project | 19-24 |
|  | Functionalities | 24-15 |
|  | Results and Discussion | 26-28 |
|  | Conclusion | 29 |
|  | References | 30 |

**Introduction**

The Student Database Management System (SDMS), named Academix Hub, is a C-based console application designed to efficiently manage student-related data in educational institutions. This system provides a centralized platform for storing, retrieving, and managing student records, academic performance, projects, internships, fees, and communication between admins and students.

**Project Scope & Objectives**

The Academix Hub system aims to:  
✔ Streamline student management – Add, view, modify, and delete records.  
✔ Automate academic tracking – Maintain performance, attendance, and results.  
✔ Manage financial records – Track fees, payments, and pending dues.  
✔ Enhance communication – Enable secure messaging between admins and students.  
✔ Ensure data security – Implement role-based access control for integrity.

**Technologies Used**

🔹 Programming Language: C  
🔹 Data Storage: File handling (students.dat)  
🔹 Development Environment: Dev-C++   
🔹 Operating System Compatibility: Windows

**System Limitations**

While Academix Hub offers an efficient approach to student data management, it has a few limitations:

Single-Institution Use – The system is designed to manage student records for one institution only and does not support multi-institution functionality.

Console-Based Interface – The application operates via a text-based interface and does not include a graphical user interface (GUI) for enhanced user experience.

Limited Data Storage – Student records are stored in a single file (students.dat), which may impact scalability and data retrieval efficiency as the dataset grows.

Despite these limitations, Academix Hub provides a structured, automated, and secure solution for managing student-related data in educational institutions. Future enhancements can focus on expanding scalability, integrating GUI support, and improving storage mechanisms for broader usability.

**System Analysis**

**Existing System**

The current **manual student data management system** relies on **physical records, spreadsheets, and paper-based processes**, leading to several inefficiencies:

* **Time-Consuming** – Searching for, updating, and managing student records manually is **slow and inefficient**.
* **Error-Prone** – Manual data entry increases the likelihood of **mistakes** in student details, fee records, and academic results.
* **Lack of Centralization** – Data is **scattered across multiple files**, making access and management difficult.
* **Limited Communication** – There is no structured way for **admins to communicate with students** efficiently.
* **Data Security Issues** – Physical records are at risk of **damage, loss, or unauthorized access**, leading to potential data breaches.

**Proposed System**

To overcome these limitations, **Academix Hub** provides a **centralized, automated, and secure** student database management system with the following key features:

* **Automated Processes** – Student record operations (**add, update, search, delete**) are **automated**, reducing manual effort and saving time.
* **Centralized Data Storage** – All records, including student details, projects, internships, fees, results, and messages, are stored in a **single, secure file (students.dat)**.
* **Error Reduction** – Built-in **validation functions** ensure accurate input for **phone numbers, emails, and percentages**.
* **Improved Communication** – The system enables **direct messaging** between admins and students.
* **Enhanced Security** – Digital storage ensures **data confidentiality**, restricting access to **authorized users** only.
* **Scalability** – Designed to handle a **growing number of student records and activities** efficiently.

**Feasibility Study**

The feasibility of **Academix Hub** is evaluated based on three key factors:

🔹 **Technical Feasibility**

* Developed in **C programming**, a widely-used and **reliable** language.
* Utilizes **file handling** for **data persistence and security**.
* Lightweight and compatible with **any system supporting a C compiler**.

🔹 **Economic Feasibility**

* Uses **free and open-source tools** (C compilers, IDEs), minimizing development costs.
* Eliminates the need for **physical storage and manual labor**, resulting in long-term **cost savings**.

🔹 **Operational Feasibility**

* **User-friendly** interface requiring **minimal training** for admins and students.
* **Simplifies complex tasks** such as **fee tracking, result management, and communication**, enhancing **overall efficiency**.

**System Design:**

**System Architecture**

The **Academix Hub** system follows a modular and structured architecture, ensuring efficiency, scalability, and maintainability. The architecture consists of the following key components:

**1. User Interface (UI)**

* A console-based interface that provides an interactive environment for both admins and students.
* Admins can manage student records, projects, internships, fees, messages, and results.
* Students can access their details, messages, results, and financial records.

**2. Data Processing Layer**

* This layer is responsible for handling CRUD (Create, Read, Update, Delete) operations.
* Functions are designed to process student records, projects, internships, fees, messages, and results efficiently.
* Ensures logical data flow and integrates validation checks before storing or retrieving data.

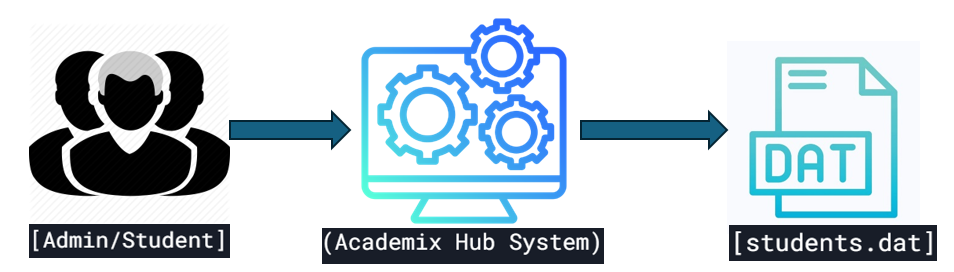
**3. Data Storage Layer**

* All records are stored in a persistent data file (students.dat) using file handling techniques in C.
* The system loads data from this file upon startup and updates it during runtime.

**4. Validation Layer**

* Ensures data integrity and correctness before storing it in the database.
* Key validation functions include:
  + validatePhone() – Ensures valid phone numbers (10 digits).
  + validateRegNo() – Checks the correctness of registration numbers.
  + validateEmail() – Ensures a valid email format.
  + validatePercentage() – Validates attendance and performance data within an acceptable range (0-100%).

The architecture is designed with a **clear separation of concerns**, allowing smooth data processing while maintaining efficiency and modularity.

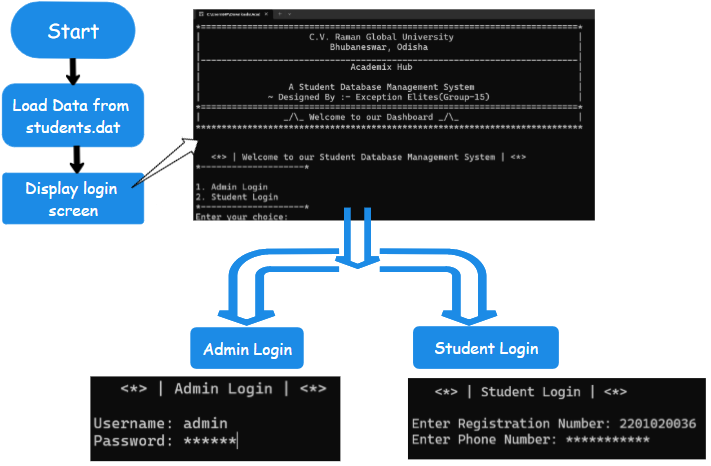
**Data Flow Diagram**

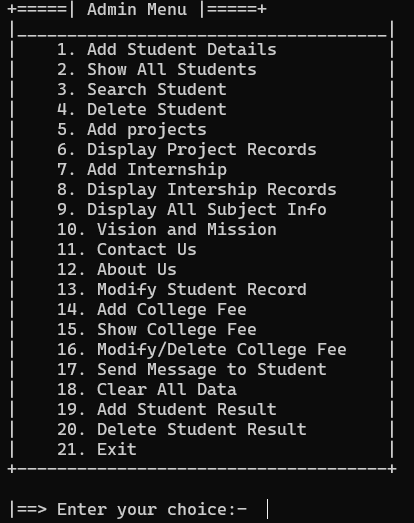
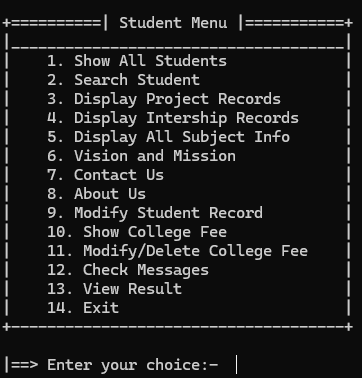
**Main Processes**

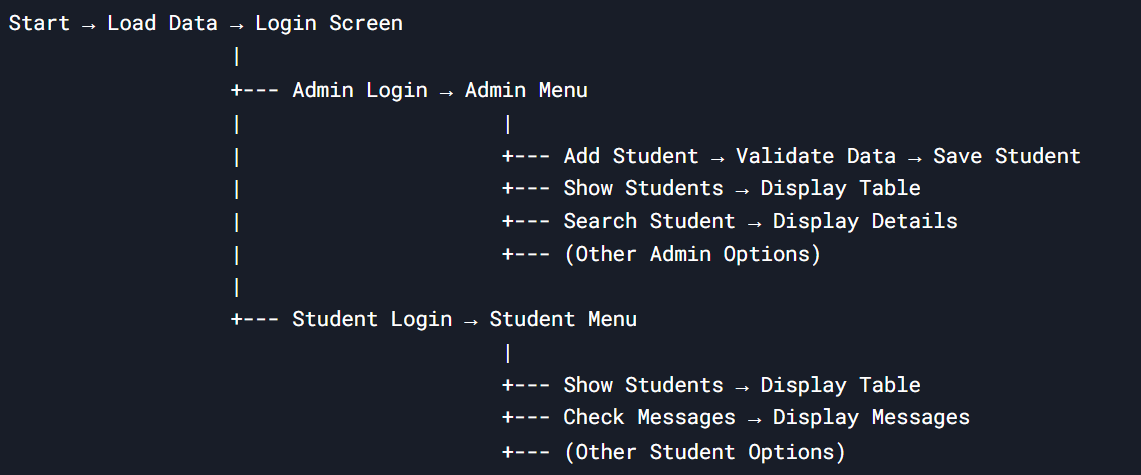
1. **Admin Processes**:
   * Add/Update/Delete Student Records
   * Manage Projects, Internships, Fees, Messages, and Results
   * Clear All Data
2. **Student Processes**:
   * View Student Records
   * Check Messages
   * View Results

**Level 2: Detailed Data Flow**

* **Admin**:
  + Input → Add Student → Validate Data → Save to students.dat
  + Input → Search Student → Retrieve from students.dat → Display
  + Input → Send Message → Save to students.dat
* **Student**: Input → View Results → Retrieve from students.dat → Display

**Flowchart for Academix Hub**

****

**Terminal Visual Flowchart**

**Implementation**

**Modules**

The **Academix Hub** system is structured into multiple modules to ensure efficient management of student records and related functionalities. The key modules include:

* **Admin Module:** Allows administrators to manage student records, projects, internships, fees, messages, and results.
* **Student Module:** Enables students to view their details, projects, internships, fee records, messages, and results.
* **Fee Management Module:** Handles the addition, modification, and display of student fee records.
* **Project & Internship Management Module:** Allows administrators to add and track student projects and internships.
* **Messaging Module:** Facilitates communication between administrators and students via in-system messages.
* **Result Management Module:** Enables the addition, deletion, and display of student academic results.

**Functionality**

Each module provides a well-defined set of functionalities:

* The **Admin Module** allows administrators to add, update, search, and delete records while ensuring data integrity through validation functions.
* The **Student Module** enables students to access their information and receive updates from administrators.
* The **Fee Management Module** maintains records of course fees, paid amounts, and pending dues.
* The **Project & Internship Management Module** keeps track of student academic projects and industrial training experiences.
* The **Messaging Module** ensures seamless communication between administrators and students.
* The **Result Management Module** allows students to check their semester-wise performance and CGPA.

**Code Structure**

The code follows a modular approach with clear separation of concerns. The key components include:

* **User Interface Layer:** Handles input/output operations via a console-based interface.
* **Processing Layer:** Implements core functionalities like adding, updating, and retrieving records.
* **Data Storage Layer:** Uses file handling (students.dat) to store and retrieve student-related information.
* **Validation Functions:** Ensures data integrity through checks on phone numbers, registration numbers, emails, and other fields.

The program is designed to be scalable and maintainable, ensuring efficient student database management.

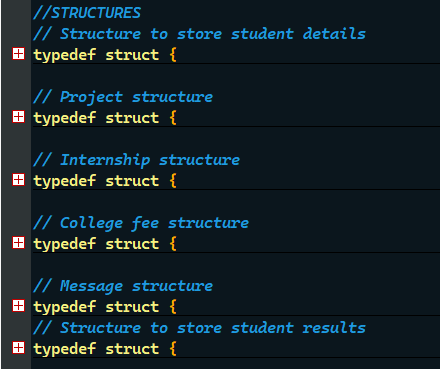
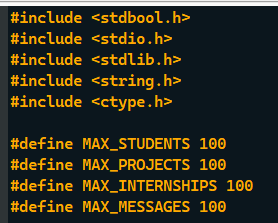
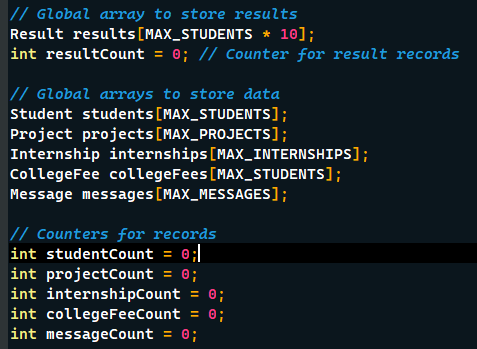
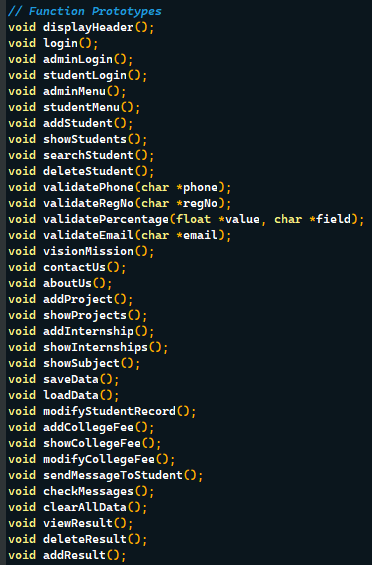
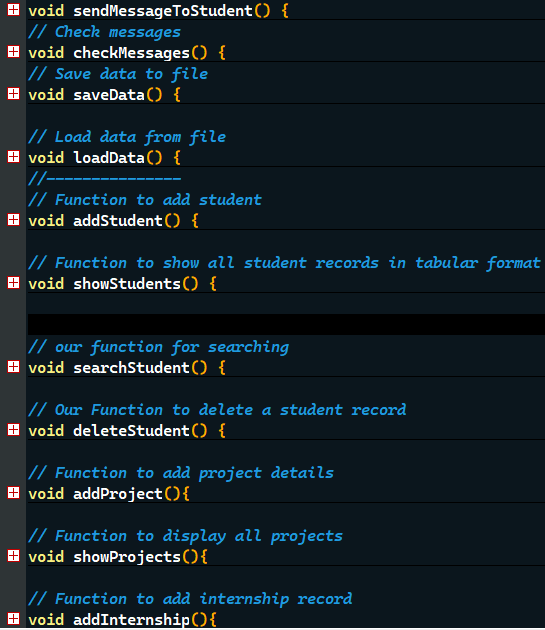
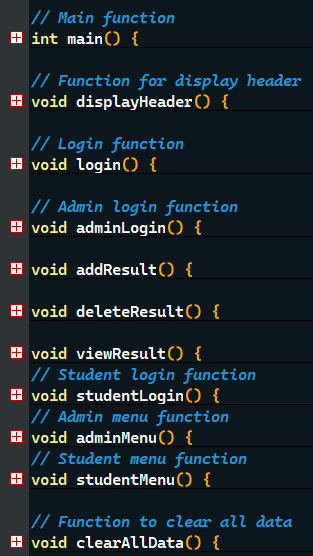


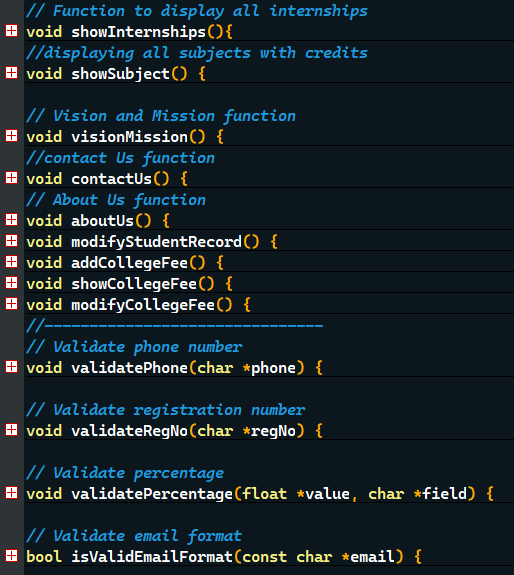
Fig. - Header files , Constants and Strutures.

** Fig: - Arrays and counters**

****

**Fig: - Function prototypes**

****

****

**Fig: - Functions and Validations**

## **Explanation of our project:**

Once upon a time, in the bustling halls of **C.V. Raman Global University**, there was a magical system called **Academix Hub**. This system was no ordinary tool—it was a **Student Database Management System** created by a group of brilliant minds known as the **Exception Elites (Group-15)**. Academix Hub was designed to bring order to the chaos of managing student records, projects, internships, fees, and even messages. It was a place where admins and students could work together seamlessly, like characters in a well-written story.

**Chapter 1: The Login Portal**

The story begins at the **Login Portal**, the gateway to Academix Hub. Here, users were greeted with a choice:

* **Admin Login**: For the wise guardians of the university.
* **Student Login**: For the eager learners seeking knowledge.

The **Admin** had to prove their worth by entering the sacred username (admin) and password (12345). Meanwhile, **Students** needed their unique **Registration Number** and **Phone Number** to gain access. Only those who entered the correct credentials could proceed to the next chapter.

**Chapter 2: The Admin’s Realm**

Once inside, the **Admin** found themselves in a grand hall filled with endless possibilities. Here’s what they could do:

1. **Add a Student**:  
   The admin could summon a new student into the system by entering their details—name, registration number, phone, email, course, performance, attendance, and more. Each student was like a new character added to the story.
2. **View All Students**:  
   The admin could gaze upon a magical table that displayed all the students in the system, complete with their performance and attendance stats.
3. **Search for a Student**:  
   If the admin needed to find a specific student, they could simply enter the student’s **Registration Number**, and the system would reveal all their secrets—academic details, subjects, and more.
4. **Delete a Student**:  
   If a student’s journey at the university came to an end, the admin could remove them from the system with a single command.
5. **Manage Projects and Internships**:  
   The admin could record the heroic deeds of students—like their **projects** (topics, programming languages) and **internships** (company names, positions, durations).
6. **Handle College Fees**:  
   The admin could keep track of the gold coins (fees) paid by students, ensuring no one fell behind in their dues.
7. **Send Messages**:  
   The admin could send magical scrolls (messages) to students, guiding them on their academic journey.
8. **Manage Results**:  
   The admin could add, delete, or view the **CGPA** of students for each semester, ensuring their academic progress was recorded.
9. **Clear All Data**:  
   In rare cases, the admin could wipe the slate clean, erasing all records from the system.

**Chapter 3: The Student’s Journey**

For the **Students**, Academix Hub was a treasure trove of information. Once logged in, they could:

1. **View Their Details**:  
   Students could see their own records—name, registration number, course, performance, attendance, and more.
2. **Check Projects and Internships**:  
   They could relive their academic adventures by viewing their **projects** and **internships**.
3. **View College Fees**:  
   Students could see how much gold they had paid and how much was still owed.
4. **Read Messages**:  
   They could open the magical scrolls (messages) sent by the admin, filled with guidance and wisdom.
5. **View Results**:  
   Students could check their **CGPA** for each semester, celebrating their victories and learning from their challenges.

**Chapter 4: The Guardians of Data**

Academix Hub was not just a system—it was a living, breathing entity that protected its data like a dragon guards its treasure. Every time a change was made, the system would save the data to a magical file called students.dat. When the system was restarted, it would load the data from this file, ensuring no information was ever lost.

**Chapter 5: The Vision and Mission**

Beyond the practical features, Academix Hub had a deeper purpose. It embodied the **Vision and Mission** of the university:

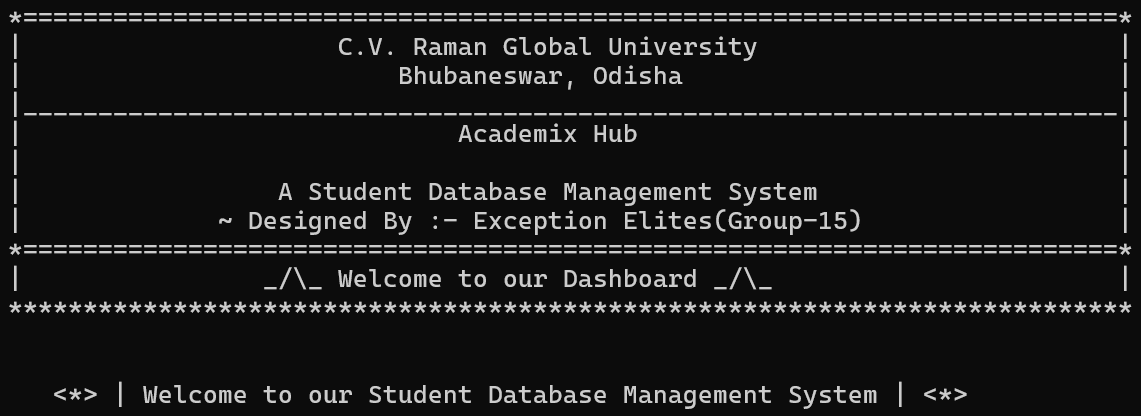
* **Vision**: To emerge as a global leader in technical education.
* **Mission**: To provide state-of-the-art education and foster innovation.

The system also had a **Contact Us** section, where students and admins could find the university’s contact details, and an **About Us** section, which told the story of the university’s evolution and legacy.

**Chapter 6: The Endless Cycle**

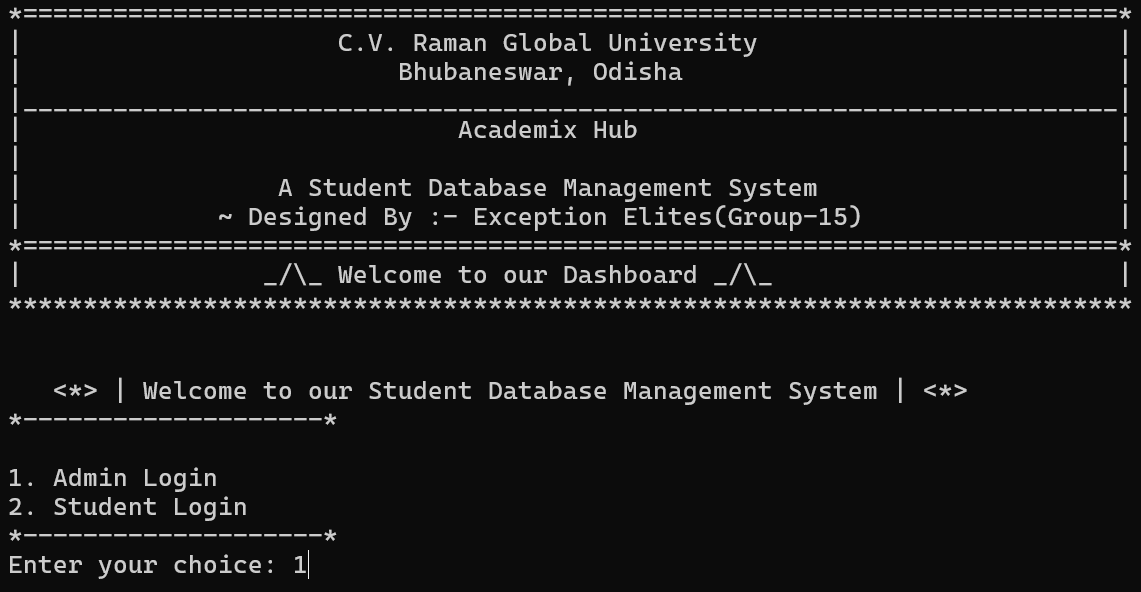
The story of Academix Hub never truly ended. Every day, new students were added, new projects were recorded, and new messages were sent. The system continued to grow, adapt, and serve the university, ensuring that every student’s journey was documented and every admin’s task was simplified.

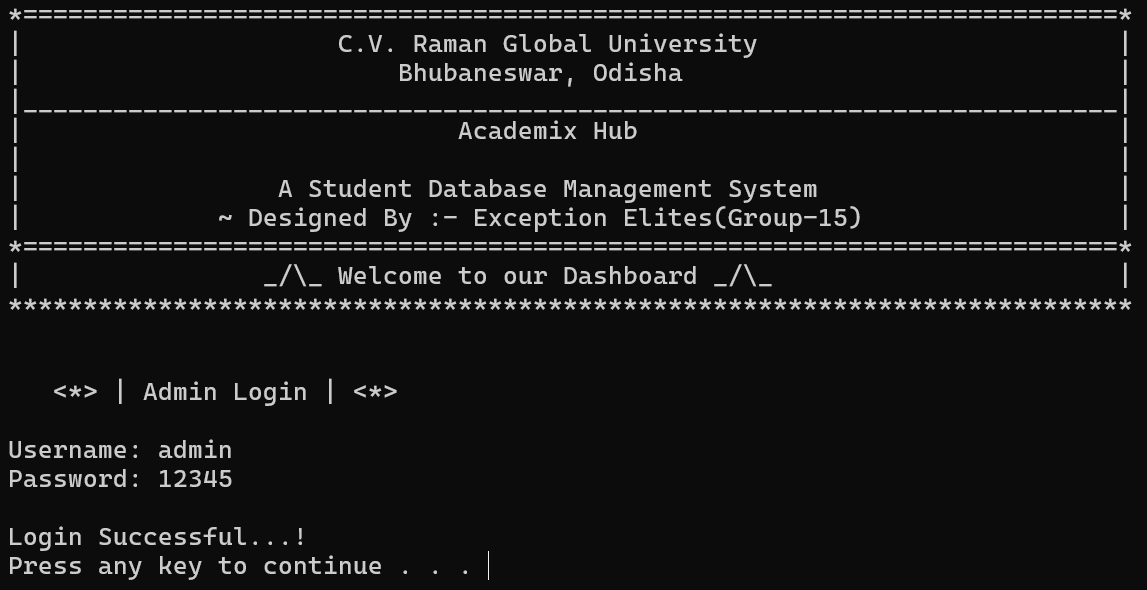
**A Legacy of Order and Efficiency**

And so, Academix Hub became a legend in the halls of C.V. Raman Global University. It was more than just a system—it was a symbol of order, efficiency, and collaboration. The **Exception Elites (Group-15)** had created something truly magical, a tool that would continue to serve the university for generations to come. 

**Snapshots of our project**

Login

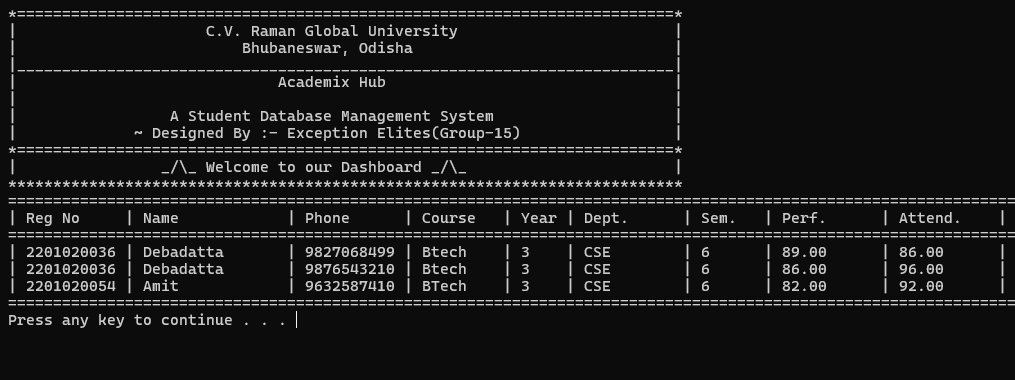




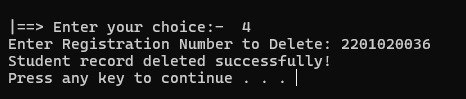
Admin Menu

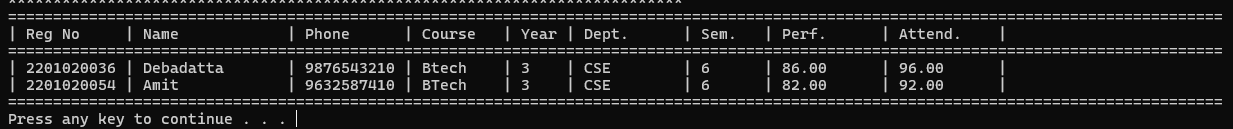


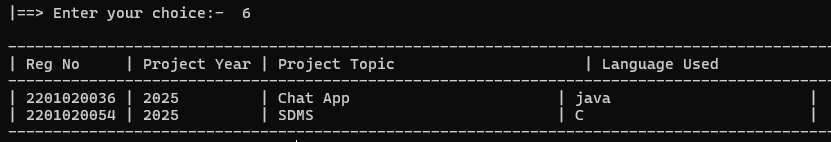
Show Record



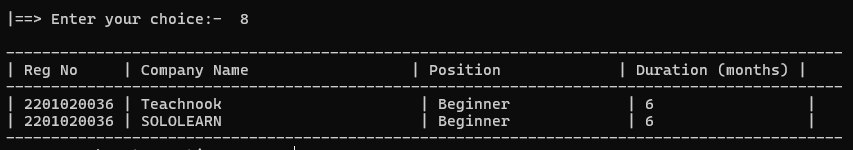
Search Record 

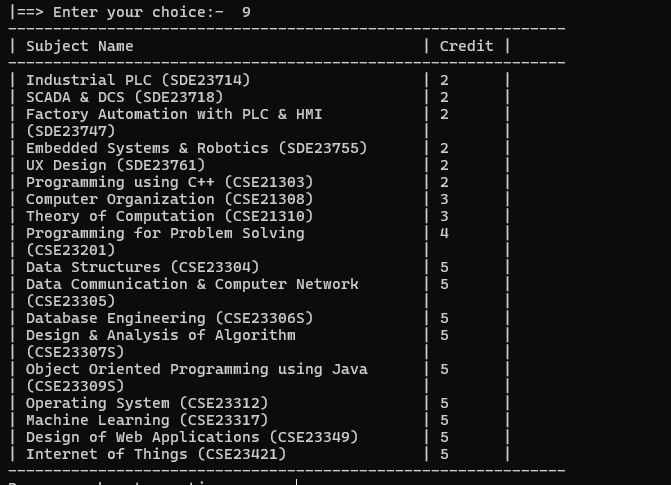
Delete Record ****

proof 

Project record

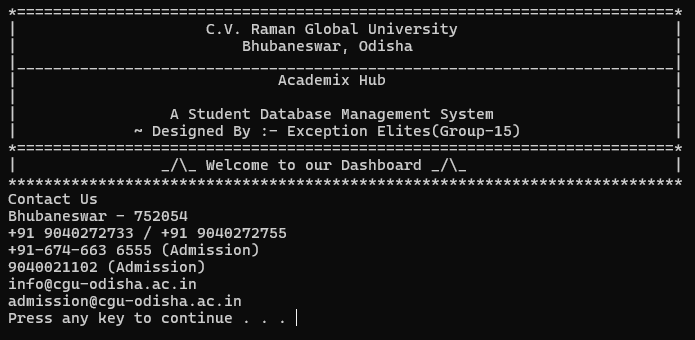
Internship record

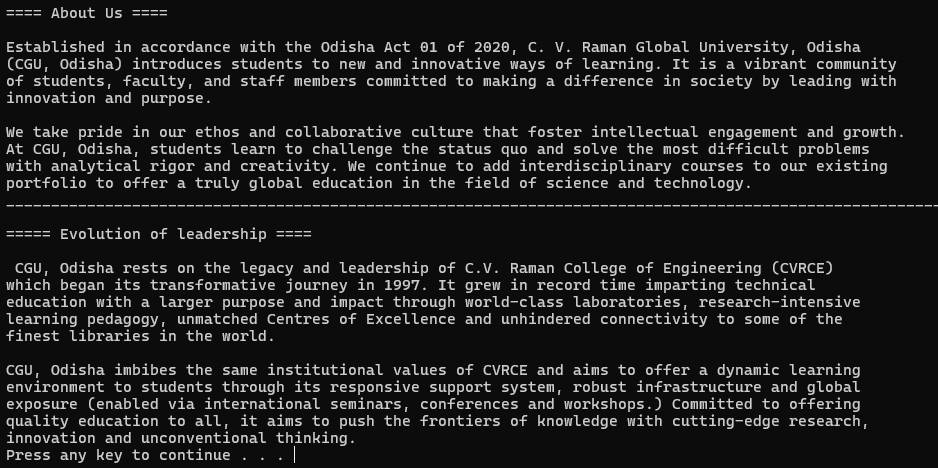


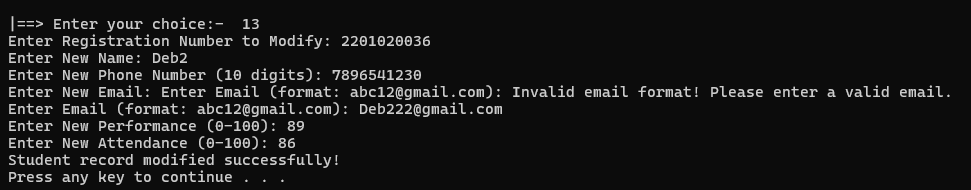
Display All Subject Info 

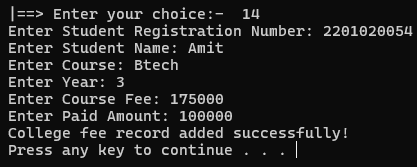
Vision and Mission 

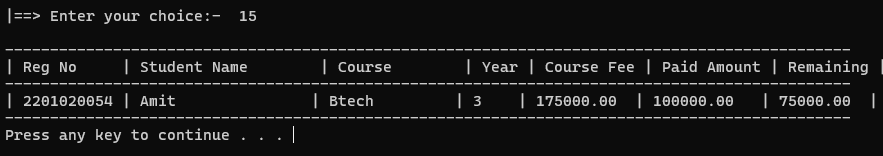
Contact us

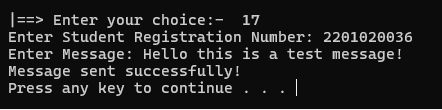


About us  


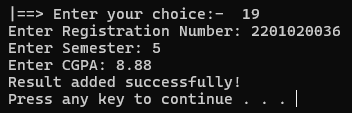
Modify Student Record 

Add college fee  


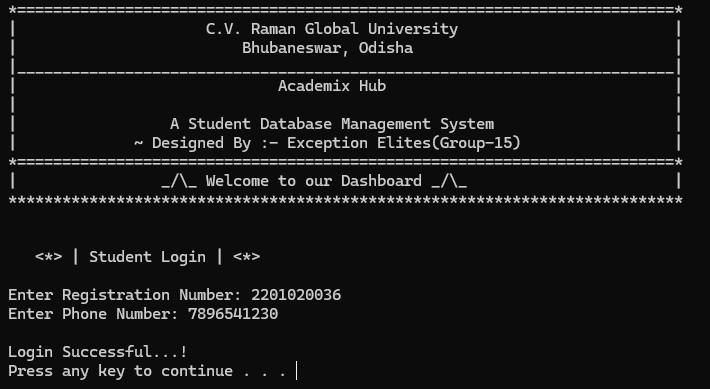
Show College Fee 

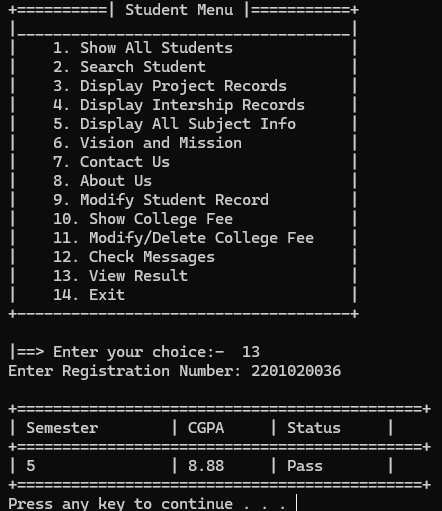
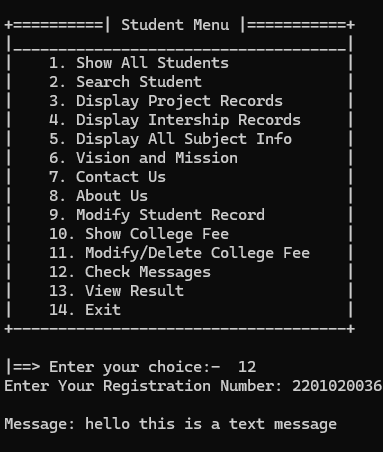
Send message to Students  


Add Student Result



Student login



View Result Check Message  
 

Remaining all Options are same…

**Functionalities / Concepts Used**

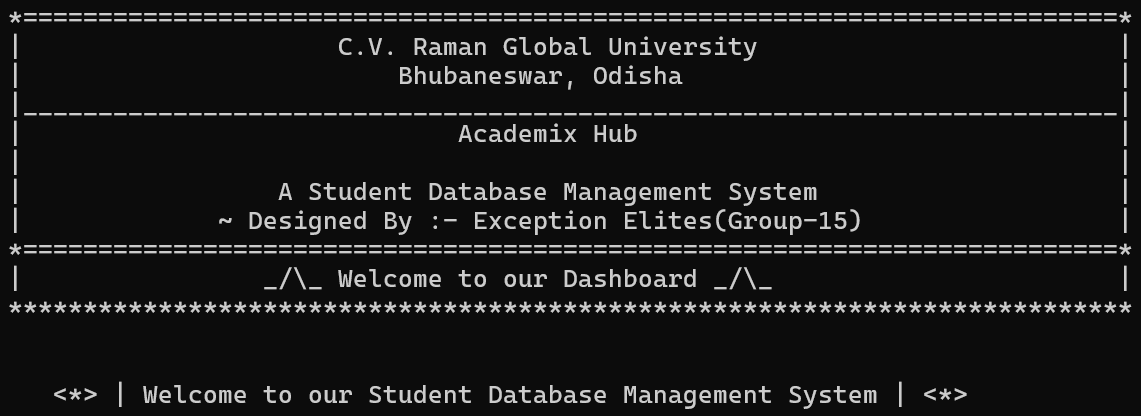
|  |  |  |
| --- | --- | --- |
| Concept | Use | Explanation |
| Structures | Define Student, Project, Internship, CollegeFee, etc. | Structures group related data types (e.g., name, regNo, phone) into a single unit for easy access. |
| File Handling | Save and load data in students.dat. | Allows persistent storage of data, so it is not lost when the program exits. |
| Arrays | Store multiple records (e.g., students[MAX\_STUDENTS]). | Arrays store multiple values of the same type, like a list of students or projects. |
| Functions | Modularize code (e.g., addStudent(), showStudents()). | Breaks the program into smaller, reusable tasks for better organization and readability. |
| Loops | Iterate through records (e.g., for, while). | Repeats a block of code, such as displaying all students or searching for a specific record. |
| Conditional Statements | Decision-making (e.g., if, else, switch). | Executes code based on conditions, like validating input or displaying menus. |
| String Manipulation | Handle text data (e.g., strcpy(), strcmp()). | Manipulates text, such as copying names or comparing registration numbers. |
| Validation Functions | Validate input (e.g., validatePhone(), validateEmail()). | Ensures data entered by users is accurate and follows specific rules. |
| Global Variables | Store counters (e.g., studentCount, projectCount). | Accessible throughout the program to track the number of records. |
| System Commands | Screen management (e.g., system("pause"), system("cls")). | Controls the console, like clearing the screen or pausing output. |
| Pointers | Used indirectly in functions like fgets() and scanf(). | Handle memory addresses, often used in input/output functions for efficiency. |
| Modular Programming | Organize code into separate functions and modules. | Improves code readability, reusability, and maintainability. |
| Data Persistence | Save and load data using saveData() and loadData(). | Ensures records are stored permanently and can be retrieved later. |
| Menu-Driven Interface | Provides options for admins and students. | Allows users to interact with the system through a series of choices. |

**Results and Discussion**

**Achievements**

The **Academix Hub** project has successfully achieved its primary goals:

1. **Efficient Student Management**:
   * The system allows admins to add, view, search, modify, and delete student records seamlessly.
   * Students can access their own records, projects, internships, fees, and results with ease.
2. **Automation of Manual Tasks**:
   * Tasks like fee tracking, result management, and communication between admins and students are automated, reducing manual effort and errors.
3. **Data Persistence**:
   * All data is saved to a file (students.dat) and loaded when the program starts, ensuring no data is lost.
4. **User-Friendly Interface**:
   * The console-based menu system is intuitive and easy to navigate for both admins and students.
5. **Scalability**:
   * The system is designed to handle a growing number of student records and activities, making it suitable for small to medium-sized institutions.



**Challenges Faced**

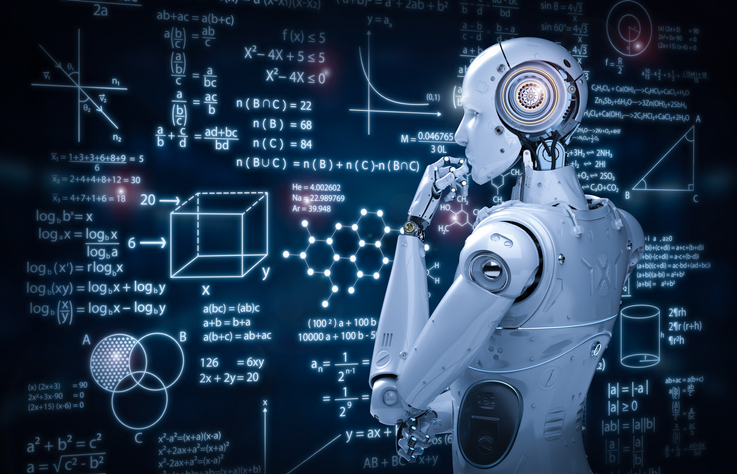
During the development of Academix Hub, several challenges were encountered:

1. Data Validation:
   * Ensuring accurate input (e.g., valid phone numbers, emails, and registration numbers) required careful implementation of validation functions.
2. File Handling:
   * Managing data persistence using file handling in C was initially complex, especially ensuring data integrity during read/write operations.
3. User Interface Limitations:
   * The console-based interface, while functional, lacks the visual appeal and interactivity of a graphical user interface (GUI).
4. Error Handling:
   * Implementing robust error handling for scenarios like invalid input or file access issues required additional effort.
5. Testing and Debugging:
   * Testing all functionalities (e.g., adding, deleting, and modifying records) and debugging errors was time-consuming.

**Future Enhancements**

To make Academix Hub even more powerful and user-friendly, the following enhancements can be considered:

1. Graphical User Interface (GUI):
   * Replace the console-based interface with a modern GUI using frameworks like GTK or Qt for a more interactive experience.
2. Cloud Integration:
   * Store data on the cloud (e.g., using Firebase or AWS) for better accessibility and scalability.
3. Mobile Application:
   * Develop a mobile app (iOS/Android) to allow students and admins to access the system on the go.
4. Advanced Search and Filtering:
   * Implement advanced search options (e.g., filter students by department, year, or performance).
5. Data Analytics:
   * Add features to analyze student performance, attendance trends, and fee payment patterns.
6. Multi-User Support:
   * Allow multiple admins to access the system simultaneously with role-based permissions.
7. Enhanced Security:
   * Implement encryption for sensitive data (e.g., student records) and add multi-factor authentication for login.
8. Integration with Other Systems:
   * Integrate with existing university systems (e.g., library management, attendance tracking) for a unified platform.
9. Report Generation:
   * Add functionality to generate PDF reports for student records, fees, and results.
10. Multi-Language Support:
    * Add support for multiple languages to make the system accessible to a wider audience.



**Conclusion**

The **Academix Hub** project represents a significant step forward in simplifying and automating student data management for educational institutions. By leveraging the power of **C programming**, the system provides a robust, efficient, and user-friendly platform for managing student records, academic performance, projects, internships, fees, and communication between admins and students. The successful implementation of key features such as **data persistence**, **input validation**, and **modular programming** demonstrates the system's ability to handle real-world challenges effectively.

Throughout the development process, the project faced challenges such as ensuring data accuracy, managing file operations, and creating an intuitive interface. However, these challenges were overcome through careful planning, iterative testing, and the use of structured programming techniques. The result is a scalable and reliable system that meets the needs of both admins and students.

Looking ahead, **Academix Hub** has immense potential for growth. Future enhancements, such as a **graphical user interface (GUI)**, **cloud integration**, **mobile app development**, and **advanced analytics**, can further elevate its functionality and usability. These improvements will not only make the system more accessible but also transform it into a comprehensive solution for modern educational institutions.

In conclusion, **Academix Hub** is more than just a Student Database Management System—it is a testament to the power of technology in streamlining administrative processes and enhancing the academic experience. By combining simplicity, efficiency, and scalability, the project lays a strong foundation for future innovations in the field of educational technology.

**References**

1. **Online Resources**:
   * GeeksforGeeks. (n.d.). *C Programming Language*. Retrieved from <https://www.geeksforgeeks.org/c-programming-language/>  
     *(A reliable online resource for learning C programming concepts, including file handling and data structures.)*
   * TutorialsPoint. (n.d.). *C Programming*. Retrieved from <https://www.tutorialspoint.com/cprogramming/>  
     *(A beginner-friendly tutorial website covering C programming basics and advanced topics.)*
   * Programiz. (n.d.). *Learn C Programming*. Retrieved from <https://www.programiz.com/c-programming>  
     *(An interactive platform for learning C programming with examples and exercises.)*
2. **Books**:
   * Kernighan, B. W., & Ritchie, D. M. (1988). The C Programming Language (2nd ed.). Prentice Hall.  
     (A classic book for learning C programming, covering syntax, structures, and file handling.)
   * Deitel, P. J., & Deitel, H. M. (2015). C How to Program (8th ed.). Pearson.  
     (A comprehensive guide to C programming, including advanced topics like data structures and algorithms.)
3. Tools/system :

* Dev c++
* Windows-64-bit