KHULNA UNIVERSITY OF ENGINEERING & TECHNOLOGY, KHULNA

Project Title : Student Management System

Course Code : CSE 1102

Group : B1

Submission Date : 27-01-2025

Submitted To	Submitted By
• Md Nazirulhasan Shawon Lecturer, Department of CSE KUET, Khulna • Md Tajmilur Rahman Lecturer, Department of CSE KUET, Khulna	• Khalid Bin Atik – 2307073 • MD. Abdullahil Kafi – 2307074 • MD. Habibur Rahman - 2307076

Introduction :

This project focuses on developing a **Student Management System** using the C programming language to efficiently handle student information at our university. It includes features for student sign-up, login, searching, updating, and deleting records, along with modules for **Class Representative elections**, **course registration**, **term-wise grade sheets**, and **GPA calculation**.

Description :

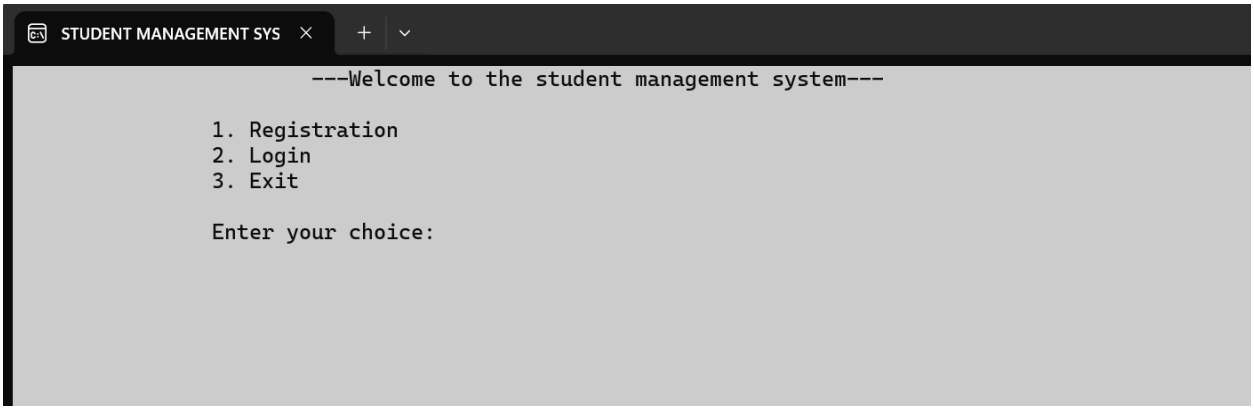
Students can-

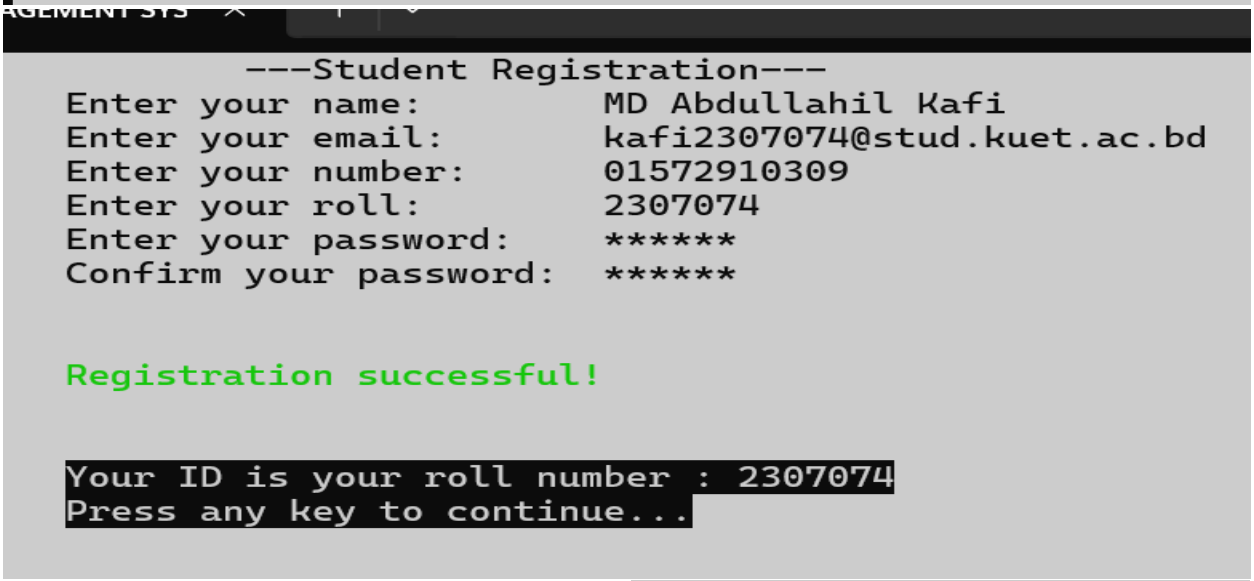
- **Sign-up :** Students can create a new account by providing their personal and academic information.
- **Log in :** Students can securely access their accounts using their credentials.
- **View information :** Students can check their stored personal and academic details.
- **Edit information :** Students can update or modify their personal and academic details as needed.
- **Delete account :** Students have the option to permanently remove their account from the system.
- **Register for a course :** Students can enroll in available courses for the current term or semester.
- **View Result :** Students can view their term-wise results and grades.
- **Log out :** Students can securely exit the system after completing their tasks.

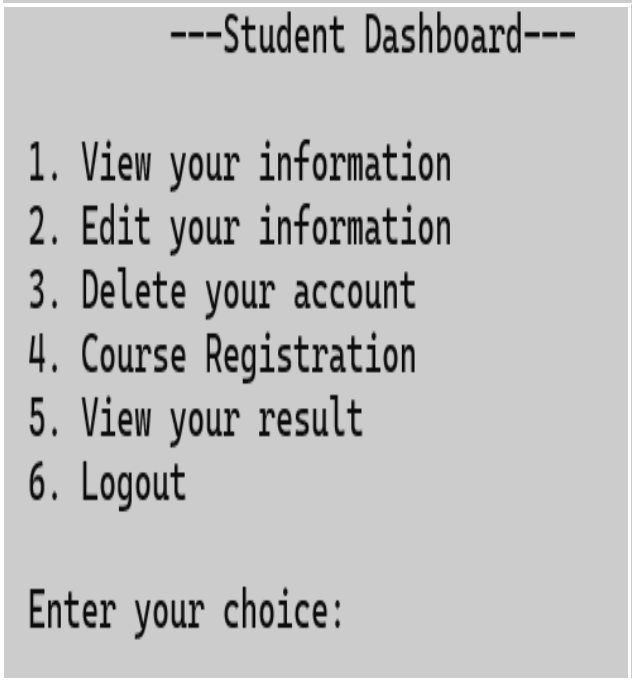
Teachers can-

- **Sign-up :** Teachers can create their accounts by providing the required information.
- **Log in :** Teachers can log in to their accounts using their credentials.
- **View information :** Teachers can access and review their stored profile details.
- **Edit information :** Teachers can update or modify their profile details.
- **Insert Students Result :** Teachers can input and save students' results and grades into the system.
- **Calculate CGPA :** Teachers can calculate the Cumulative Grade Point Average (CGPA) for students over multiple terms.
- **Elect Class Representative :** Teachers can facilitate and manage the election process for Class Representatives.
- **Search student's information and result :** Teachers can search for and view details of students, including their academic performance.
- **Log out :** Teachers can securely log out of the system after completing their activities.

Sample Screenshots:







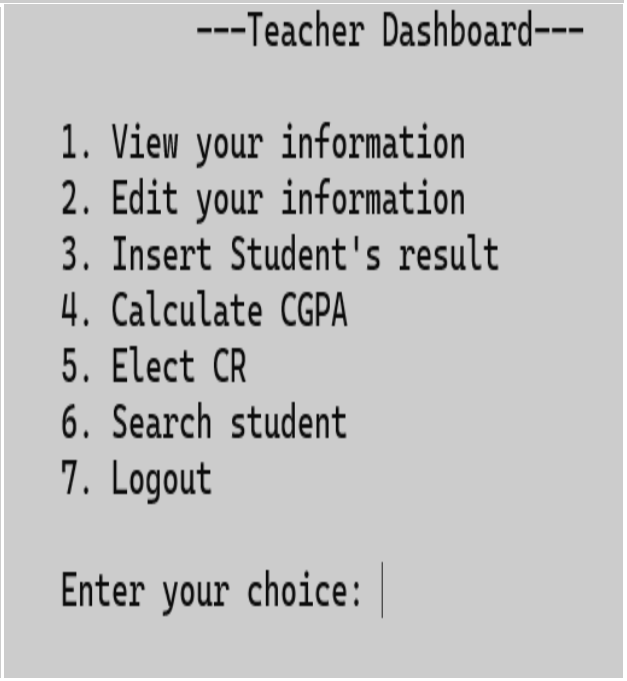
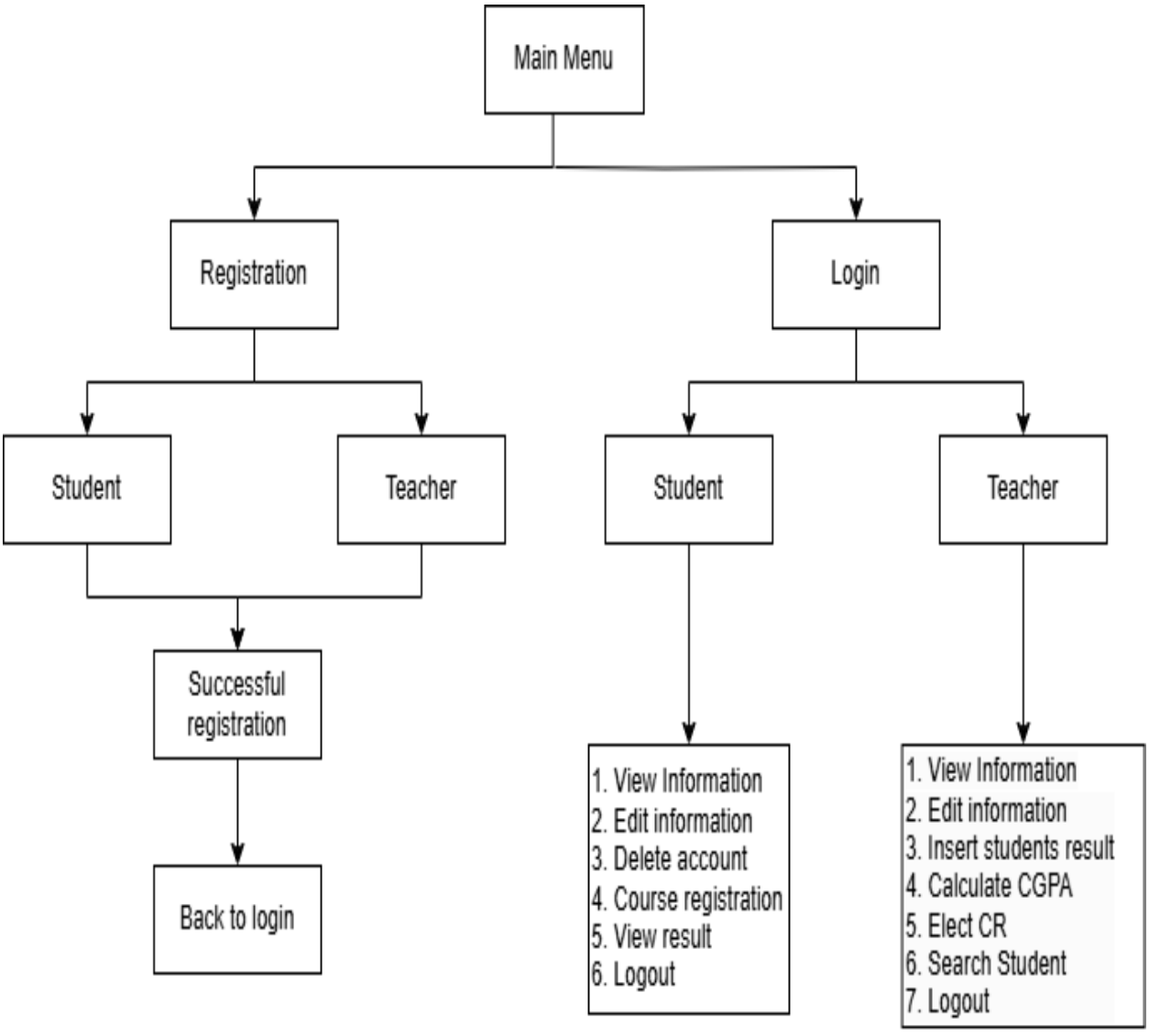


Diagram :


Main Menu
Registration
Login
Student
Teacher
Student
Teacher
Successful registration
Back to login
1. View Information
2. Edit information
3. Delete account
4. Course registration
5. View result
6. Logout
1. View Information
2. Edit information
3. Insert students result
4. Calculate CGPA
5. Elect CR
6. Search Student
7. Logout

Aims & Objectives :

The goal of this project is to create a simple and user-friendly system to help students and teachers manage important tasks easily. Students can register, update their information, check results, and register for courses, while teachers can calculate GPAs, manage class representatives, and access student details. It's all about making things easier for everyone.

Software Requirements:

1. Programming Language : C/C++
2. Integrated Development Environment(IDE) : Code Blocks/Visual Studio Code
3. Operating System : Windows

Learnings :

Throughout the development of the project, we worked with many important programming concepts, and here’s what we learned from them:

- **If-else construct & Loops :**
  We utilized **if-else** constructs to implement decision-making logic in the program, allowing us to handle different scenarios based on user input or data validation.
  Loops (such as for, while, and do-while) were extensively used to process repetitive tasks like navigating through files, displaying menus, and iterating over arrays or structures.

- **Switch-case :**
  It was helpful in creating user-friendly menu-driven programs where multiple options needed to be handled efficiently.
- **Type conversion** :
  This enabled us to switch between different data types (e.g., converting strings to integers or floats) for operations like GPA calculations and input validation.
- **Arrays:**
  Arrays were used to store and manage data like GPA scores for multiple subjects and to perform operations such as calculations and data retrieval efficiently.
- **Strings :**
  String manipulation was an integral part of the project for handling user inputs like names, emails, and IDs. Functions like strcmp, strcpy, and strlen were used to compare, copy, and measure strings effectively**.**
- **Structures :**
  We learned to use structures to group related data into a single unit. For example, student data (name, ID, email, GPA) was stored in a structure, making data management more organized and modular.
- **Pointers :**
  Pointers were utilized for memory access and dynamic memory allocation. They allowed us to link structures and work with files more efficiently by directly addressing memory locations.
- **User defined function :**
  To maintain modularity and reduce code duplication, we wrote user-defined functions for repetitive tasks like login validation, file reading/writing, and data retrieval. This made the code easier to debug and maintain**.**

- **File Handling :**
  File handling was one of the key aspects of the project. It allowed us to store and retrieve student and teacher data persistently. We learned to use functions like fopen, fread, fwrite, fscanf, and fprintf to perform operations like reading and writing from files.
- **User defined header file :**
  We created a custom header file to organize function declarations, macros, and global variables. This enhanced code reusability and modularity.
- **Header Guards :**
  To prevent multiple inclusions of the same header file, we implemented header guards using #ifndef, #define, and #endif. This ensured a more efficient compilation process and eliminated redundancy**.**
- **Modularity:**
  We structured the program into smaller, manageable modules, each performing a specific task. This made the program easier to understand, debug, and scale.
- **Dynamic Memory Allocation :**
  Using functions like malloc and free, we dynamically allocated memory during runtime. This allowed the program to handle varying amounts of data efficiently and utilize system memory effectively.

Conclusion :

This project taught us how to use important programming tools like files, pointers, and structures. We learned how to organize our code and solve problems better. It was a great learning experience that improved our coding skills and helped us understand how everything works together in a program.

THANK YOU