**PROJECT AND TEAM INFORMATION**

## A red text on a black background AI-generated content may be incorrect. STUDENT MANAGEMENT SYSTEM

|  |
| --- |
| **STUDENT MANAGEMENT SYSTEM** |

## Student/Team Information

|  |  |
| --- | --- |
| Team Name:  Team # (Mentor needs to assign) |  |
| Team member 1 (Team Lead)  (Last Name, name: student ID: email, picture): | Prabhjot Singh  24051048  Prabhjotpvtt2007@gmail.com |
| Team member 2  (Last Name, name: student ID: email, picture): | Divyansh Tiwari  2423028  tiwaridivyansh539@gmail.com |
| Team member 3 | Ayush Kumar Chaudhary  24051062  ayushkumarchaudhary247@gmail.com |

|  |  |
| --- | --- |
| Team member 4 | Kapil  2423032  [ls94949478@gmail.com](mailto:ls94949478@gmail.com) |

**PROJECT PROGRESS DESCRIPTION (35 pts)**

## Project Abstract (2 pts)

(Brief restatement of your project’s main goal. Max 300 words).

|  |
| --- |
| This program is a console-based student management system protected by a password login. The password input is masked, so when the user types, asterisks (\*) appear instead of the actual characters, enhancing security. The program supports both Windows and Unix-like systems by implementing a custom getch() function to read characters without echoing.  Once the correct password is entered (with a maximum of three attempts), the user gains access to the student management menu. This menu allows adding new students by entering their roll number, name, and marks, viewing the list of all students, and searching for a student by roll number. Student data is stored in an array in memory and is managed using simple functions.  Overall, the program demonstrates basic password protection combined with fundamental student record management. It can be run on multiple platforms and serves as a good foundation for learning about input handling, data storage, and menu-driven applications in C. |

## Updated Project Approach and Architecture (2 pts) (Describe your current approach, including system design, communication protocols, libraries used, etc. Max 300 words).

|  |
| --- |
| The current project implements a console-based Student Management System secured by a password authentication mechanism. The core system design revolves around a modular, menu-driven architecture that allows users to securely access student records and perform basic operations such as adding, viewing, and searching students.  The system employs a password-based login to restrict access, enhancing security. Password input is masked using custom input handling to prevent the password from being displayed on the screen. To ensure cross-platform compatibility, the program includes a platform-specific implementation of the getch() function. On Windows, it uses Windows API calls to disable console input echo and read single characters, while on Unix-like systems, it configures terminal settings via POSIX termios to achieve similar behavior. This approach removes dependency on <conio.h>, which is not universally available.  Student data is managed in-memory using a fixed-size array of Student structures, which contain roll number, name, and marks fields. The program provides a simple text-based menu system allowing users to add new student records, view all existing records, and search for students by roll number. Input validation and buffer management ensure robustness against common input errors.  Communication in this project is limited to console input/output, which simplifies the interaction model and keeps the system lightweight without the need for network protocols or external databases. Standard C libraries such as <stdio.h> and <string.h> are used for I/O and string handling, while system-specific APIs support secure input.  This design balances simplicity and functionality, making it an excellent foundation for future expansion—such as adding persistent storage, enhanced security features, or a graphical user interface—while providing a secure and user-friendly environment for managing student data. |

## 

## Tasks Completed (7 pts) (Describe the main tasks that have been assigned and already completed. Max 250 words).

|  |  |
| --- | --- |
| Task Completed | Team Member |
| **Requirement Analysis:** The project objective was clearly defined – to create a menu-driven system that can manage a database of buses, including their details and provide search functionality.  **Program Design and Planning:** The structure of the program was planned, including the menu options, required data fields, and the overall flow of the application. Functions for each operation were outlined in the planning phase.  **Data Structure Implementation:** A C++ struct was created to store each bus’s details such as bus number, driver name, departure time, arrival time, and destination. An array of these structures was used as the temporary in-memory database.  **Menu-Driven Interface:** A user-friendly, text-based menu system was developed to allow interaction with the system. Options include adding new buses, displaying all buses, and searching by bus number.  **Linear Search Functionality:** A linear search algorithm was implemented to enable users to find a bus using its unique bus number during program execution.  **Testing and Debugging:** The entire system was tested with various inputs to ensure functionality. Minor bugs related to input handling and menu navigation were identified and resolved. | Lalit Kishore Nautiyal  Praveen Prasad Goswami  Lalit Kishore Nautiyal  Soumya Meharwal  Praveen Prasad Goswami  Soumya Meharwal |

## Challenges/Roadblocks (7 pts) (Describe the challenges that you have faced or are facing so far and how you plan to solve them. Max 300 words).

|  |
| --- |
| During the development of the password-protected Student Management System, several challenges arose, mainly related to cross-platform compatibility, secure password input, and managing user input effectively. One significant roadblock was handling the password masking functionality without relying on platform-specific headers like <conio.h>, which are not supported universally. This issue required implementing a custom getch() function for both Windows and Unix-like systems, involving different system calls and terminal settings. Ensuring this solution works seamlessly across multiple platforms demanded considerable testing and debugging.  Another challenge was managing input buffering and newline characters, especially when switching between scanf() and fgets() calls. Incorrect handling could cause skipped inputs or unexpected behavior. To address this, careful attention was given to consuming leftover newline characters after each input and sanitizing strings to avoid errors during string operations.  Data management also posed a challenge as the system currently stores student records in memory only, limiting scalability and persistence. Adding file-based storage or database integration was considered but deferred to maintain focus on core functionality. This limitation means all data is lost when the program terminates, reducing practical usability. Future plans involve implementing persistent storage to enhance data reliability.  Lastly, designing a user-friendly interface within the constraints of console I/O was challenging. The menu system had to be intuitive and robust against invalid inputs while maintaining simplicity. This required implementing input validation and clear prompts.  Going forward, these challenges will be addressed by adding persistent storage, improving input handling robustness, and potentially migrating to a graphical user interface or web-based platform to enhance usability. Cross-platform support will continue to be a priority, ensuring the program runs smoothly on different operating systems. |

## Tasks Pending (7 pts)

|  |  |
| --- | --- |
| Task Pending | Team Member (to complete the task) |
| **Implementation of File Handling:** Currently, all data is stored in memory and is lost once the program ends. The next step is to add file handling using C++ file streams (fstream) to enable **persistent storage** of bus details. This will allow the system to save and retrieve bus records across multiple sessions.  **Binary Search Integration:** At present, a linear search is used for finding buses by their number. To improve efficiency, especially as the dataset grows, a **binary search algorithm** will be implemented. This requires the bus records to be sorted by bus number, and it will significantly reduce search time.  **Ticket Booking System:** A ticket booking module will be added to allow passengers to reserve seats. This includes tracking **seat availability**, assigning seat numbers, and generating a simple ticket confirmation. This feature will be integrated into the existing menu structure.  **Graphical User Interface (GUI):** To improve usability and user experience, a basic **GUI will be implemented** using a C++-compatible toolkit such as Qt or using web-based integration with CGI scripts. This will replace or complement the current console-based interface. | Praveen Prasad Goswami  Lalit Kishore Nautiyal  Lalit Kishore Nautiyal  Soumya Mehrwal |

## 

## Project Outcome/Deliverables (2 pts)

(Describe what are the key outcomes / deliverables of the project. Max 200 words).

|  |
| --- |
| The primary outcome of this project is a functional, console-based Student Management System secured by a password authentication mechanism. The system enables authorized users to manage student records efficiently by providing key functionalities such as adding new students, viewing all student details, and searching for students by their roll number. Password protection ensures that only authenticated users can access the system, enhancing security.  The deliverables include:   1. **Secure Login Module:** A password input system with masked characters that prevents unauthorized access through multiple login attempts. 2. **Student Management Module:** Core functionalities to add, list, and search student records with proper input validation and user prompts. 3. **Cross-platform Compatibility:** Custom implementation of secure input handling that supports both Windows and Unix-like operating systems without relying on non-portable libraries. 4. **User-friendly Console Interface:** A clear and straightforward menu-driven interface that guides users through the system’s features.   This project lays a solid foundation for future enhancements, such as data persistence through file storage or database integration, and possibly transitioning to a graphical or web-based interface. Overall, it demonstrates essential programming concepts like secure input, data structuring, and user interaction in a practical application. |

# Progress Overview (2 pts) (Summarize how much of the project is done, what's behind schedule, what's ahead of schedule. Max 200 words.)

|  |
| --- |
| The project has successfully completed the core functionalities, including password-protected access and a fully functional student management system. The login mechanism with masked password input is implemented and tested across platforms, ensuring secure user authentication. The student management features—adding students, viewing all records, and searching by roll number—are fully operational and provide a smooth user experience via a console-based menu.  Currently, the project is on schedule regarding the development of basic features. All planned modules for this initial phase have been developed and tested thoroughly. No major delays have been encountered so far.  However, enhancements like data persistence (saving and loading student data from files or databases), error handling improvements, and additional features such as updating and deleting student records remain to be implemented. These upcoming tasks are planned for the next development phase and may require additional time for design and testing.  Overall, the project is progressing well, with foundational components completed ahead of schedule, giving ample time to focus on extending functionality and improving robustness in the coming weeks. |

# Codebase Information (2 pts) (Repository link, branch, and information about important commits.)

|  |
| --- |
| The project codebase is hosted on a GitHub repository at: [**https://github.com/yourusername/student-management-system**](https://github.com/yourusername/student-management-system)  The main development takes place on the **main** branch, which contains the latest stable version of the password-protected student management system.  Key commits include:   * **Initial commit**: Added basic project structure and implemented cross-platform getch() function for password masking. * **Password protection module**: Developed and tested secure password input with masked characters. * **Student management features**: Implemented functions for adding, viewing, and searching student records. * **Menu system integration**: Connected authentication to student management menu flow. * **Bug fixes and cross-platform support**: Resolved input handling issues and ensured compatibility across Windows and Unix-like systems.   All commits follow semantic commit messages for clarity, and the repository is regularly updated to track progress and improvements. |

## Testing and Validation Status (2 pts) (Provide information about any tests conducted)

|  |
| --- |
| The project has undergone thorough manual testing to ensure core functionalities work as expected. Password input masking was tested on both Windows and Linux environments, verifying that characters are masked correctly and the system properly handles backspaces and enter keys. The authentication mechanism was validated by confirming access is granted only with the correct password and denied after three failed attempts.  For the student management system, tests were conducted for adding new students, viewing the student list, and searching by roll number. Input validation checks ensure that student data is correctly stored and retrieved. Boundary cases, such as searching for non-existent roll numbers and adding students beyond the maximum limit, were also tested to confirm graceful handling of errors.  Future testing plans include automated unit tests for key functions and stress testing with large datasets to verify scalability and robustness. Additional validation will focus on input sanitization and security hardening to prevent invalid or malicious entries. |

## 

# Deliverables Progress (2 pts) (Summarize the current status of all key project deliverables mentioned earlier. Indicate whether each deliverable is completed, in progress, or pending.)

|  |
| --- |
| 1. **Password-Protected Access System** *Status:* Completed The password authentication module with masked input has been fully implemented and tested on multiple platforms, ensuring secure system access. 2. **Student Management System** *Status:* In Progress Core functionalities like adding, viewing, and searching students are working as intended. Additional features such as editing and deleting student records are planned for implementation. 3. **Cross-Platform Input Handling** *Status:* Completed Custom getch() implementations for Windows and Unix-based systems have been integrated, allowing consistent password input masking without relying on non-portable libraries. 4. **User Interface Improvements** *Status:* Pending A more user-friendly and colorful command-line interface is planned to enhance usability and feedback but has not been started yet. 5. **Testing and Validation Suite** *Status:* In Progress Basic unit tests for password validation and student data handling are being developed. Comprehensive testing coverage will follow in the next phase. |