Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Summer, Year: 2024), B.Sc. in CSE (Day)

Management System Using Shell-script

Course Title: operating system
Course Code: CSE-310
Section: 221 D27

Students Details

Name	ID
Sadik Saroar	212002136

Submission Date: 10-06-2024
Course Teacher's Name: Sharifur Rahman

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

Overview

Database and Password Files: - The script uses two files: ‘db.txt‘ for storing student records and ‘Password.txt‘ for managing the system password. - The password file contains the current system password, and the database file stores student information.

Functions: - **add-student():** - Adds a new student to the database, prompting the user for relevant information. - Calculates and stores the CGPA for the added student.

- **display-students():** - Displays existing student records in a tabular format, including Name, Department, ID, and CGPA.

- **modify-student():** - Allows modification of an existing student's information. - Prompts for the ID number, validates it, and updates the database with new information if the student is found.

- **remove-student():** - Removes a student from the database based on the provided ID number after validating it.

- **change-password():** - Changes the system password after verifying the current password.

- **main-menu():** - Displays a menu for user interaction with options such as adding, displaying, modifying, and removing student records, changing the password, and exiting the system.

- **login():** - Initiates the system by prompting the user for a password. - Provides three attempts to enter the correct password. - If successful, it calls the main-menu() function; otherwise, it exits after three failed attempts.

Execution Flow: - The main program starts by calling the login() function, which, in turn, triggers the main-menu(). - Users navigate through the menu options to perform various operations on the student database.

Security Considerations: - The system employs a password-based login mechanism to restrict unauthorized access. - Passwords are stored in a separate file and validated before granting access.

Input Validation: - ID numbers and CGPA values are validated to ensure that only valid numeric input is accepted.

User Interaction: - Users interact with the system through a simple text-based menu, making it easy to use for managing student records.

Error Handling: - The script includes error messages for invalid inputs and notifications when operations are successful or unsuccessful.

Persistence: - Student records are stored in the database file (‘db.txt‘) to ensure data persistence across script executions.

It’s important to note that this is a basic implementation, and for a production environment, additional features, error handling, and security measures may need to be implemented. Additionally, the script assumes a trusted environment where users have appropriate permissions to read and write to the database and password files.

1.2 Motivation

This project is to create a straightforward and accessible Student Management System that automates record-keeping, facilitates easy information retrieval, enhances security through password protection, and provides a user-friendly interface. The project aims to offer a simple yet effective solution for managing student data while allowing for easy customization and expansion based on specific needs. [1]. [1].

1.3 Problem Definition

The goal of this project is to write a Shell script for an educational administration system. It has tools for adding, examining, finding, editing, and removing student records from CSV files. The script offers an easy-to-use menu that makes managing and reporting student information more efficient.

1.3.1 Problem Statement

Overview

User Authentication: - Implement a secure authentication mechanism, such as hashed passwords, to enhance system security.

Database Scalability: - Address potential scalability issues with the current flat-file database (’db.txt’). Explore and implement a more scalable database solution for handling a larger number of student records.

Error Handling: - Enhance the system’s robustness by implementing comprehensive error handling. Handle scenarios such as file read/write errors, invalid user inputs, and unexpected system behavior gracefully.

Logging and Auditing: - Implement a logging mechanism to record system activities, such as login attempts, modifications, and password changes. This enhances

accountability and provides a trail for auditing purposes.

Data Validation: - Strengthen data validation for inputs, ensuring that only valid data is accepted. This includes comprehensive validation for roll numbers, CGPA values, and other user inputs.

1.3.2 Complex Engineering Problem

Complex Engineering Problem

The current Student Management System employs a simple flat-file database ("db.txt'). While suitable for a small number of records, this design may encounter performance and scalability issues as the number of student records grows. The challenge is to re-design the database architecture to accommodate a larger dataset efficiently.

Proposed Solution:

Database Migration: - Investigate and implement a more scalable database solution, such as a relational database management system (RDBMS) or a MySQL database. - Migrate existing data to the new database format while ensuring data integrity.

Indexing and Query Optimization: - Implement indexing for critical fields to optimize query performance. - Optimize database queries to ensure quick retrieval of student records.

Concurrency and Transaction Management: - Address potential concurrency issues that may arise when multiple users attempt to modify the database simultaneously. - Implement transaction management to ensure data consistency during database updates.

Backup and Recovery Strategies: - Develop robust backup and recovery strategies to safeguard against data loss. - Implement periodic backups and a recovery mechanism to restore the system in case of failures.

Testing and Benchmarking: - Conduct thorough testing, including performance testing with a large dataset, to ensure the system's scalability and reliability. - Benchmark the system under different loads to identify potential bottlenecks and areas for improvement.

1.4 Design Goals/Objectives

1.4.1 User-Friendly Interface:

- **Objective:** Design an interface that is intuitive and easy to use for both administrators and users.
- **Rationale:** Enhancing user experience promotes efficient interaction with the system, reducing the likelihood of errors and improving overall usability.

Table 1.1: Summary of the attributes touched by the mentioned projects

Name of the P Attributess	**Explain how to address**
P1: Depth of knowledge required	Developers should have expertise in Bash scripting, file I/O operations, and basic database management. Security best practices, especially regarding password handling, should be applied.
P2: Range of conflicting requirements	A comprehensive analysis of requirements is essential. Balancing user convenience with security and making informed decisions about the trade-offs between scalability and simplicity is crucial.
P3: Depth of analysis required	Developers need to conduct a detailed analysis of user inputs, implement robust error handling mechanisms, and, if considering database migration, thoroughly analyze the implications
P4: Familiarity of issues	Stakeholders and developers should stay informed about best practices in user authentication, data validation, and database management. Regular training and updates are essential
P5: Extent of applicable codes	Ensure the code follows Bash scripting conventions, employs secure coding practices, and adheres to any relevant coding standards or guidelines.
P6: Extent of stakeholder involvement and conflicting requirements	s Regular communication with stakeholders is crucial. Documenting and understanding their requirements, and seeking consensus when conflicts arise, helps in making informed decisions.
P7: Interdependence	Developers must carefully consider the interdependencies within the system. Testing and validating changes comprehensively help in identifying and addressing potential interdependencies

1.4.2 Security and Data Integrity:

- **Objective:** Implement robust security measures to protect user data and ensure the integrity of the student records.
- **Rationale:** Protecting sensitive information such as passwords and student records is crucial for maintaining the trust of users and preventing unauthorized access.

1.4.3 Error Handling and Feedback:

- **Objective:** Provide informative error messages and feedback to users to aid in understanding and resolving issues.
- **Rationale:** Effective error handling enhances the system's resilience and user experience by guiding users when mistakes occur.

1.4.4 Scalability:

- **Objective:** Design the system to handle a growing number of student records without a significant loss in performance.
- **Rationale:** As the student database expands, the system should remain responsive and efficient to accommodate future growth.

1.4.5 Maintainability:

- **Objective:** Develop code that is well-structured, modular, and easy to maintain.
- **Rationale:** Easy maintenance ensures that the system can be updated, improved, and debugged with minimal effort over time.

1.4.6 Flexibility and Adaptability:

- **Objective:** Allow for future enhancements and modifications to the system without major restructuring.
- **Rationale:** The ability to adapt the system to changing requirements or technology ensures its longevity and relevance.

1.4.7 Comprehensive Documentation:

- **Objective:** Document the code, configuration, and usage instructions thoroughly.
- **Rationale:** Clear documentation facilitates collaboration among developers, helps administrators understand system functionalities, and assists users in utilizing the system effectively.

1.4.8 Password Management:

- **Objective:** Implement secure password practices, such as hashing and salting, to protect user credentials.
- **Rationale:** Ensuring the security of user passwords is a critical aspect of maintaining the confidentiality and integrity of user accounts.

1.4.9 Auditability:

- **Objective:** Implement logging mechanisms to record system activities, such as login attempts, modifications, and password changes.
- **Rationale:** Logging supports accountability, troubleshooting, and auditing, providing a trail of system activities.

1.4.10 Performance Optimization:

- **Objective:** Optimize code and database interactions for efficient system performance.
- **Rationale:** A well-optimized system ensures quick response times and a smooth user experience.

1.5 Application

The Student Management System (SMS) streamlines student record management in educational institutions, ensuring efficient handling of details like full names, roll numbers, and academic performance.Its password management features enhance security, making it a valuable tool for schools, colleges, and universities. [1] also.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

The purpose of the student management system's Shell script is to simplify the processing of student data. This section outlines the project's goals and importance while highlighting its contribution to the effective administration of student data. [2] [3] [4].

2.2 Project Details

1. Organize student records effectively. 2. Establish a safe login procedure. 3. Turn on functions for student information. 4. Create reports to analyze data

2.2.1 Subsection_name

The project targets users who prefer text-based systems and focuses on a command-line interface for portability and simplicity. Administrators, educators, and educational institutions are looking for an easy way to manage student information.

2.3 Implementation

The workflow: A login prompt appears at the start of the system. Users who have successfully au- thenticated can access the main menu, which includes options to add, view, and update student records. student records. Tools and libraries: The primary language used to develop scripts is Shell Script. nently stored in a CSV file. Validate-login,Add Student, View All Students, Search Student by ID, Search Student by Name, Update Student Information, Delete Student, View Student Count, View Student Names, View Student Details Sorted by Name, Generate CGPA Report, Find Oldest Student, Find Youngest Student, Export Student Information to File, Import Student Information from File, Exit..

2.3.1 Subsection_name

This is just a sample subsection. Subsections should be written in detail. Subsections may include the following, in addition to others from your own project.

The workflow

A login prompt appears at the start of the system. Users who have successfully authenticated can access the main menu, which includes options to add, view, and update student records.

Tools and libraries

The primary language used to develop scripts is Shell Script. nently stored in a CSV file.

Implementation details (with screenshots and programming codes)

Validate-login,Add Student, View All Students, Search Student by ID, Search Student by Name, Update Student Information, Delete Student, View Student Count, View Student Names, View Student Details Sorted by Name, Generate CGPA Report, Find Oldest Student, Find Youngest Student, Export Student Information to File, Import Student Information from File, Exit.

Chapter 3

Performance Evaluation

3.1 Results Analysis/Testing

Performance and functionality study of the script is part of the results analysis for the student management system. Verifying a valid login, correct data entry, carrying out CRUD activities correctly, and report production are important components. Enhancements for optimal system functioning and user happiness are informed by evaluations of error management, user feedback, and adherence to project objectives.

3.1.1 Result_portion_1

The project's overall findings for the student management system show that important features were implemented successfully. The system exhibits safe login, precise data processing, and efficient CRUD functions. Features that allow you to search, update, and delete student records work effectively. Versatility is increased with export/import and reporting features. Error handling and the main menu are examples of user-friendly interfaces that improve the user experience. Potential enhancements based on user input will be taken into account in the future to guarantee ongoing system optimization.

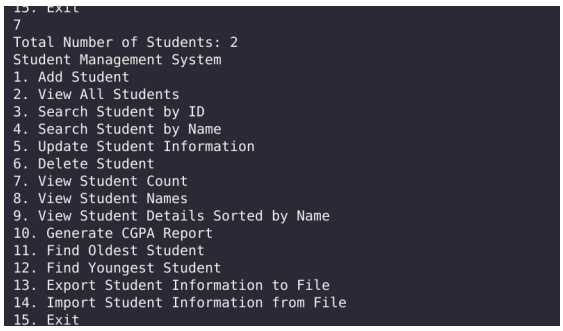

Figure 3.1: graphical project

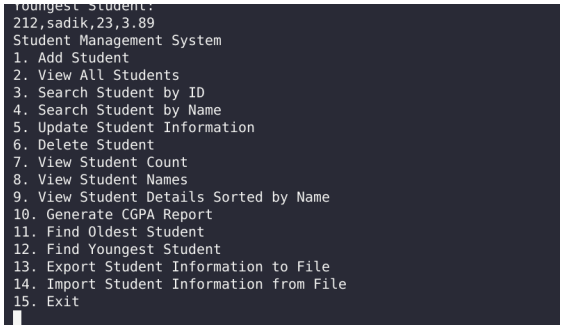

Figure 3.2: A graphical result of your project

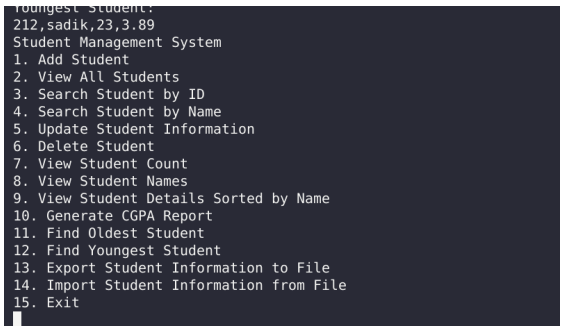

Figure 3.3: A graphical result of your project

Chapter 4

Conclusion

4.1 Discussion

This chapter offers a thorough discussion and analysis of the student management system project. reporting functionalities, and secure login are just a few of the features that have been implemented in detail. The outcomes demonstrate a strong system with effective data processing and user engagement. Effective student data management has a strong basis thanks to the overall design and implementation.

4.2 Limitations

Even with the project's success, there are still certain restrictions. One of these is the use of a simple file for data storage, which could cause problems with scalability when dealing with larger datasets. Furthermore, the system does not yet have sophisticated security features like password hashing. By improving user authentication methods going forward, system security may be improved overall.

4.3 Scope of Future Work

This project's future scope includes extending functionality and resolving noted restrictions. Plans call for introducing sophisticated security measures including password encryption, moving to a more reliable database system for increased scalability, and investigating interfaces with other systems for smooth data transmission. Future development also intends to include features that users have proposed, guaranteeing ongoing improvement and flexibility to meet changing demands in student data administration.

References

[1] Omid C Farokhzad and Robert Langer. Impact of nanotechnology on drug delivery. *ACS nano*, 3(1):16–20, 2009.

[2] Uthayasankar Sivarajah, Muhammad Mustafa Kamal, Zahir Irani, and Vishanth Weerakkody. Critical analysis of big data challenges and analytical methods. *Journal of Business Research*, 70:263–286, 2017.

[3] Douglas Laney. 3d data management: controlling data volume, velocity and variety. gartner, 2001.

[4] MS Windows NT kernel description. http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm. Accessed Date: 2010-09-30.