Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Fall, Year: 2023), B.Sc. in CSE (Day)

Design And Development of Database Controller Using Shell Script

Course Title: Operating System Lab
Course Code: CSE-310
Section: 212-D4

Students Details

Name	ID
Md. Sajjad Hossen.	212902032
Md. Ziaur Rahman	212902002

Submission Date: 07-01-2024
Course Teacher's Name: Md. Jahidul Islam

[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

A Database Controller System (DBCS) is a software suite that enables users to store, organize, and manage data efficiently. This controller aims to provide a convenient and efficient means of handling database operations through the command line, utilizing shell scripting capabilities. In the realm of information technology, databases are vital components for various applications, ranging from simple websites to complex business systems. A Database DBCS not only ensures data integrity and security but also provides mechanisms for data retrieval and manipulation.

1.2 Motivation

The motivation of this project is to offer a lightweight solution that is readily incorporated into different contexts, as well as to make database management duties simpler for users who prefer command-line interfaces. Using shell scripting for database management can improve productivity and simplify graphical user interfaces since it is an effective technique for automating repetitive operations.

1.3 Problem Definition

1.3.1 Problem Statement

Many traditional database management systems feature graphical user interfaces, which may not be to everyone's taste—especially for users who are more comfortable with command-line environments. This gap is filled by developing a shell script-based database controller, which offers a versatile and scriptable approach to database management. Furthermore, a compact and effective solution that can manage routine database activities without the requirement for a full-featured database management system is required.

1.3.2 Complex Engineering Problem

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	Advanced understanding of database systems and shell scripting.
P2: Range of conflicting requirements	Balancing performance, security, and scalability while ensuring compatibility with various databases.
P3: Depth of analysis required	In-depth analysis of database architecture, query optimization, and shell scripting intricacies.
P4: Familiarity of issues	Proficient knowledge of database management challenges and nuances in shell scripting.
P5: Extent of applicable codes	Comprehensive utilization of shell scripting for database query optimization, backup, and maintenance tasks.
P6: Extent of stakeholder involvement and conflicting requirements	Collaboration with database administrators, developers, and system architects to address conflicting requirements and ensure stakeholder satisfaction.
P7: Interdependence	Internet

1.4 Objectives

The primary objectives of this project are as follows:

1. Ease of Use: The main objective is to develop an intuitive shell script interface for database management so that users, even those with no database experience, may easily carry out routine tasks.

2. Scriptability: The architecture ought to make it easier to write automation scripts, enabling users to easily incorporate the database controller into their current workflows.

3. Compatibility: Make sure the controller is compatible with widely used database systems like MySQL, PostgreSQL, SQLite, and so on. This will allow it to be flexible and useful in a variety of settings.

4. Security: To protect the security of the data and the database system, put secure practices into place. Examples of these include managing credentials securely and offering access controls.

5. Efficiency: To allow for rapid and responsive interactions with the database, the controller must be efficient in both resource utilization and execution speed.

1.5 Application

The process of designing and developing a Database Controller with Shell Script includes writing a script that acts as a user interface for a database system. Using shell commands and SQL statements, this script would make operations like administering, updating, and querying the database easier. The script can perform tasks like managing errors, running SQL queries, establishing a connection to the database, and generating prompts that are easy to understand for the user. It might also include security features like authorization and authentication to provide restricted database access. The major objectives of the Database Controller Shell Script are to improve user experience, simplify database operations, and safeguard the security and integrity of the underlying data.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

Creating a Shell Script Database Controller is a crucial project that aims to automate and streamline database management chores by employing a flexible, script-driven methodology. This chapter delves into the project's thorough design, development, and implementation phases, highlighting the strategic choices made to guarantee effectiveness, dependability, and flexibility in database management.

2.2 Project Details

The main goal of the project is to improve database management in a Unix/Linux environment by developing a Database Controller with Shell Script. The principal aim is to offer an intuitive and effective means of executing diverse database functions, such as retrieving, inserting, updating, and deleting data. The project's objectives are to automate repetitive processes, maintain consistency, and eliminate errors to lessen the manual labor required for database maintenance. The Database Controller will be engineered to ensure compatibility and flexibility by integrating smoothly with widely used database management systems. It will also have security features built to protect sensitive data, giving it a complete solution for a range of database management requirements.

2.3 Implementation

Below is the project code of the Database Controller Using Shell Script.

```
// #!/bin/bash

# Function to show information for Admin
Show_Admin_Info() {
```

```
    whiptail --title "Admin Info" --msgbox "$(ls -l)" 20 78
}

# Function to create a user
Create_User() {
    user=$(whiptail --inputbox "Enter your username" 8 39 --title "Create
        a new user" 3>&1 1>&2 2>&3)
    password=$(whiptail --passwordbox "Enter your password" 8 39 --title
        "Create a new user" 3>&1 1>&2 2>&3)
    # Store the username and password securely (you might want to enhance
        the security)
    echo "$user:$password" >> user_info.txt
    whiptail --title "Create a new user" --msgbox "User $user created
        successfully!" 8 78
}

# Function to authenticate a user
Authenticate_User() {
    user=$(whiptail --inputbox "Enter your username" 8 39 --title "User
        Login" 3>&1 1>&2 2>&3)
    password=$(whiptail --passwordbox "Enter your password" 8 39 --title
        "User Login" 3>&1 1>&2 2>&3)

    # Check if the username and password match
    if grep -q "$user:$password" user_info.txt; then
        whiptail --title "User Login" --msgbox "Welcome, $user!" 8 78
        return 0 # Return success
    else
        whiptail --title "User Login" --msgbox "Authentication failed.
            Invalid username or password." 8 78
        return 1 # Return failure
    fi
}

# Function to delete a user (Admin only)
Delete_User() {
    user_to_delete=$(whiptail --inputbox "Enter the username to delete" 8
        39 --title "Delete User" 3>&1 1>&2 2>&3)

    # Check if the user exists in the user_info.txt file
    if grep -q "^$user_to_delete:" user_info.txt; then
        # Delete the user from the file
        sed -i "/^$user_to_delete:/d" user_info.txt
        whiptail --title "Delete User" --msgbox "User $user_to_delete
            deleted successfully!" 8 78
    else
        whiptail --title "Delete User" --msgbox "User $user_to_delete not
            found!" 8 78
    fi
}
```

```
# Admin Panel Function
Admin_Panel() {
    select=$(whiptail --title "Admin Panel" --menu "Choose an option" 25
        78 10 \
        "1" "Show Admin Info" \
        "2" "Delete User" \
        "3" "Back to Main Menu" 3>&1 1>&2 2>&3)

    case "$select" in
    1)
        Show_Admin_Info
        ;;
    2)
        Delete_User
        ;;
    3)
        return
        ;;
    esac
}
Main_Menu_Loop() {
    while true; do
        select=$(whiptail --title "Main Program" --menu "Choose an
            option" 25 78 10 \
            "1" "Admin Panel" \
            "2" "Existing User" \
            "3" "Create User" \
            "4" "Exit" 3>&1 1>&2 2>&3)

        case "$select" in
        1)
            Admin_Panel
            ;;
        2)
            Authenticate_User
            if [ $? -eq 0 ]; then
                Menu_Loop # Only call Menu if authentication is successful
            fi
            ;;
        3)
            Create_User
            ;;
        4)
            whiptail --title "Exit" --msgbox "! ! Exiting from the
                program ! !" 8 78
            exit
            ;;
        esac
    done
}
```

```
# Main program
Menu_Loop() {
    select=100
    while [ $select -gt 0 ]; do
        select=$(whiptail --title "Data Base Control Unit" --menu "Choose
            an option" 25 78 10 \
            "1" "Create a database" \
            "2" "Manage an existing database" \
            "3" "Copy a database" \
            "4" "Remove a database from system" \
            "5" "Show the databases" \
            "6" "Quit from Program" \
            "7" "Back to main menu" 3>&1 1>&2 2>&3)
        case "$select" in
        1)
            name=$(whiptail --inputbox "Enter your database name" 8 39
                --title "Create a database" 3>&1 1>&2 2>&3)
            touch $name
            whiptail --title "Create a database" --msgbox "! ! Database
                Created ! !" 8 78
            ;;
        2)
            db=$(whiptail --inputbox "Type your database name" 8 39
                --title "Student Database System" 3>&1 1>&2 2>&3)
           menu=100
while [ $menu -gt 0 ]; do
    menu=$(whiptail --title "Manage an existing database" --menu "Choose
        an option" 25 78 5 \
        "1" "Insertion" \
        "2" "Delete" \
        "3" "Show Info" \
        "4" "Query" \
        "5" "Jump to Home" 3>&1 1>&2 2>&3)
    case "$menu" in

        1)
         num=$(whiptail --inputbox "Number of entries" 8 39 --title
             "Insertion" 3>&1 1>&2 2>&3)
        for ((i = 0; i < num; i++)); do
                    id=$(whiptail --inputbox "ID" 8 39 --title "Type
                        your data" 3>&1 1>&2 2>&3)
                    nam=$(whiptail --inputbox "Name" 8 39 --title "Type
                        your data" 3>&1 1>&2 2>&3)
                    gpa=$(whiptail --inputbox "CGPA" 8 39 --title "Type
                        your data" 3>&1 1>&2 2>&3)
                    loc=$(whiptail --inputbox "Location" 8 39 --title
                        "Type your data" 3>&1 1>&2 2>&3)
                      bat=$(whiptail --inputbox "Batch" 8 39 --title
                          "Type your Batch" 3>&1 1>&2 2>&3)
                  ema=$(whiptail --inputbox "email" 8 39 --title "Type
                      your E-mail" 3>&1 1>&2 2>&3)
```

```
                    var=" $id $nam $gpa $loc $bat $ema "
                    echo " $var " >>$db
                    whiptail --title "Insertion" --msgbox "! !
                        Insertion Complete ! !" 8 78
                done
                ;;

            2)
                ln=$(whiptail --inputbox "Enter the id number" 8 39
                    --title "Delete" 3>&1 1>&2 2>&3)
                sed -i "/$ln/d" $db
                whiptail --title "Deletion" --msgbox "! ! Deletion
                    Complete ! !" 8 78
                ;;
            3)
                whiptail --title "Show Info" --msgbox "$(cat $db)" 30
                    78
                ;;
            4)
                query_menu
                ;;
            5)
                menu=0
                ;;
            esac
        done
        ;;
    3)
        copy_db
        ;;
    4)
        remove_db
        ;;
    5)
        whiptail --title "Show the databases" --msgbox "$(ls -p |
            grep -v /)" 20 78
        ;;

      6)
        whiptail --title "Quit from Program" --msgbox "! ! Exiting
            from the program ! !" 8 78
        exit
        ;;
       7)
        return
        ;;
    esac
  done
}

# Function to handle queries
```

```
query_menu() {
    que=100
    while [ $que -gt 0 ]; do
        que=$(whiptail --title "Query" --menu "Choose an option" 25 78 7 \
            "1" "Sort By ID Ascending Order" \
            "2" "Sort By ID Descending Order" \
            "3" "Total number of students" \
            "4" "Search By ID" \
            "5" "Search By Name" \
            "6" "Search By Location" \
            "7" "Jump to Sub menu" 3>&1 1>&2 2>&3)
        case "$que" in
        1)
            whiptail --title "Query" --msgbox "$(sort $db)" 20 78
            ;;
        2)
            whiptail --title "Query" --msgbox "$(sort -r $db)" 20 78
            ;;
        3)
            line=$(wc -l < $db | tr -d ' ')
            whiptail --title "Query" --msgbox "Total number of students:
                $line " 8 78
            ;;
        4)
            search_id
            ;;
        5)
            search_name
            ;;
        6)
            search_location
            ;;
        7)
            que=0
            ;;
        esac
    done
}

# Function to search by ID
search_id() {
    en=$(whiptail --inputbox "Enter the ID" 8 39 --title "Search By ID"
        3>&1 1>&2 2>&3)
    result=$(grep -w "$en" $db)

    if [ -n "$result" ]; then
        whiptail --title "Query" --msgbox "$result" 20 78
    else
        whiptail --title "Query" --msgbox "No matching records found for
            ID: $en" 20 78
    fi
```

```
}

# Function to search by name
search_name() {
    en=$(whiptail --inputbox "Enter the name" 8 39 --title "Search By
        Name" 3>&1 1>&2 2>&3)
    whiptail --title "Query" --msgbox "$(grep $en $db)" 20 78
}

# Function to search by location
search_location() {
    el=$(whiptail --inputbox "Enter the location" 8 39 --title "Search By
        Location" 3>&1 1>&2 2>&3)
    whiptail --title "Query" --msgbox "$(grep $el $db)" 20 78
}

# Function to copy a database
copy_db() {
    file=$(whiptail --inputbox "Enter file name" 8 39 --title "Copy a
        database" 3>&1 1>&2 2>&3)
    file1=$(whiptail --inputbox "Enter second file name" 8 39 --title
        "Copy a database" 3>&1 1>&2 2>&3)
    if [ -f $file ]; then
        cp $file $file1
        whiptail --title "Copy a database" --msgbox "! ! File Copied ! !"
            8 78
    else
        whiptail --title "Copy a database" --msgbox "! ! File does not
            exist ! !" 8 78
    fi
}

# Function to remove a database
remove_db() {
    file=$(whiptail --inputbox "Enter a file name to be removed" 8 39
        --title "Remove a database from system" 3>&1 1>&2 2>&3)
    if [ -f $file ]; then
        rm -i $file
        whiptail --title "Remove a database from system" --msgbox "! !
            Database Removed ! !" 8 78
    else
        whiptail --title "Remove a database from system" --msgbox "! !
            $file does not exist ! !" 8 78
    fi
}

# Check if users.txt exists and is not empty
if [ ! -e users.txt ]; then
    touch users.txt
fi
```

```
#if [ ! -s users.txt ]; then
    #Create_User
#fi

# Execute the main menu loop
Main_Menu_Loop
```

Chapter 3

Performance Evaluation

3.1 Simulation Procedure

The simulation procedure for evaluating the performance of the designed and developed Database Controller using Shell Script involves a systematic and comprehensive approach. A set of relevant workloads and test cases are defined to mimic real-world conditions and start the evaluation process. The simulation mimics the anticipated usage patterns of the database controller by including a variety of database activities, including data retrieval, insertion, deletion, and updates. To evaluate how responsive the system is under different circumstances, parameters such as the volume of data, frequency of transactions, and number of concurrent users are methodically changed. During these simulations, the focus is on measuring the execution time, resource utilization, and system throughput. To make sure the database controller is reliable and effective, a variety of scenarios will be replicated.

3.2 Results Analysis

The outcomes of the simulation are examined to evaluate the Database Controller's performance. Numerous important indicators are the subject of the analysis, such as resource consumption, throughput, and reaction time. Response time is a metric that indicates how well a system executes database operations and how responsive it is overall. Resource consumption offers information about how well the system uses the hardware resources that are available available hardware resources, such as CPU, memory, and disk usage. The efficacy of the Shell Script-based Database Controller is ascertained by comparing the outcomes with predetermined performance benchmarks and industry standards. To improve overall performance, any anomalies or bottlenecks found during the investigation are fixed, and optimizations are taken into account. The results of this performance review are quite important.

3.2.1 Output

Figure 3.1: Main Interface

Figure 3.2: Inside the admin panel

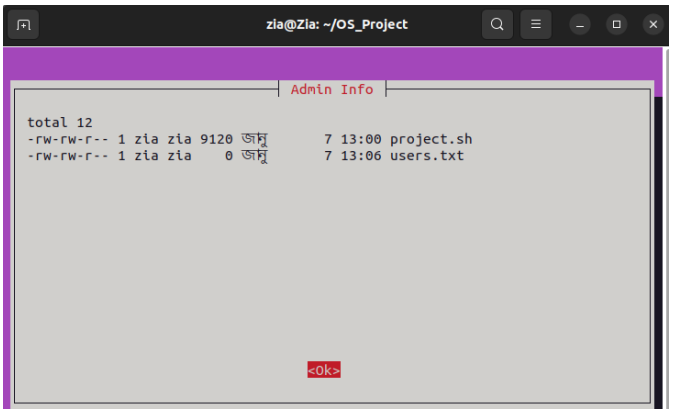

Figure 3.3: Admin info

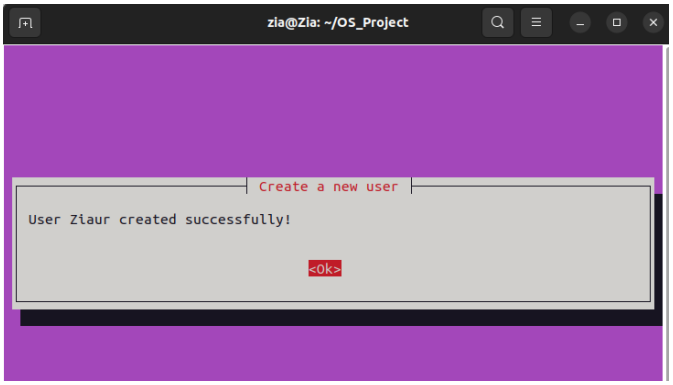

Figure 3.4: Create a new user/admin

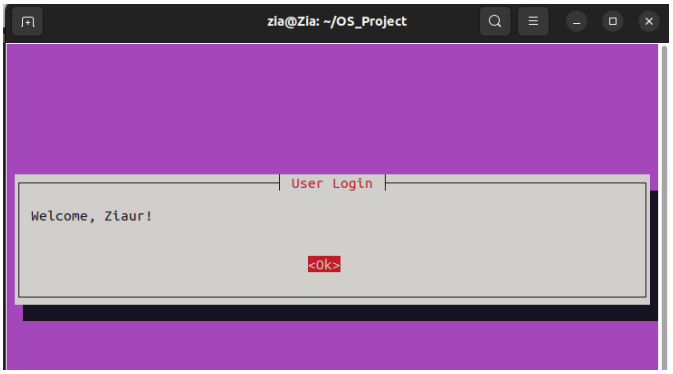

Figure 3.5: User/Admin login

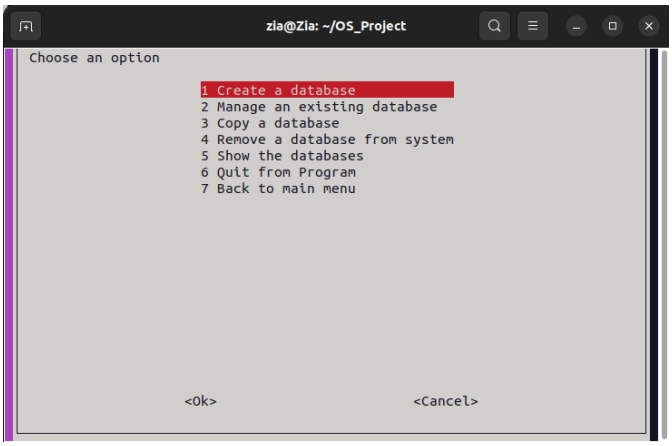

Figure 3.6: Database control panel

Figure 3.7: Database Creation and Deletation

Figure 3.8: Inside the Database insertion

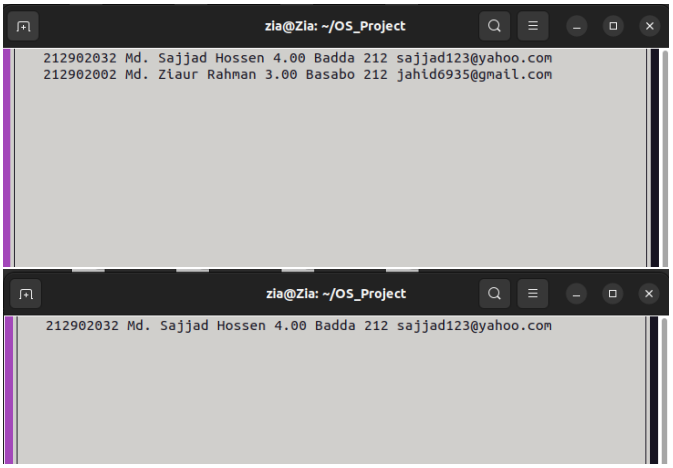

Figure 3.9: Inserted And Deleted entries

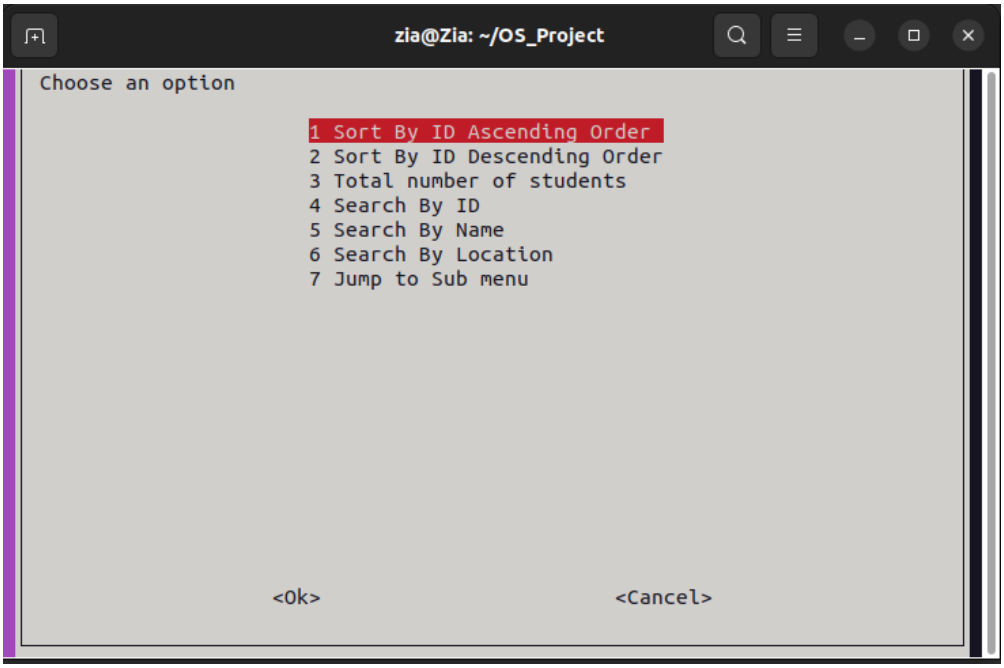

Figure 3.10: Query Part

Chapter 4

Conclusion

4.1 Discussion

In conclusion, the utilization of Shell Script in the design and development of the Database Controller has been a noteworthy advancement in improving the effectiveness of database management. The script efficiently automates several operations, including user management, data manipulation, and backup, and it gives database managers a simplified and intuitive user interface. We have struck a balance between usability and accessibility by utilizing Shell Script, making it suitable for both inexperienced and seasoned users. The script proved to be dependable and strong when managing database activities, which improved the efficiency and security of the database environment.

4.2 Limitations

The Database Controller using Shell Script has its limitations even with its triumphs. Its reliance on the shell capabilities of the underlying system is one significant drawback. When transferring the script to several settings with different shell implementations, compatibility problems could occur. Additionally, handling huge datasets or sophisticated queries may have an impact on the script's performance. To overcome these constraints, the script's codebase would need to be further optimized, and extensive testing on a variety of operating systems and database setups would be necessary.

4.3 Scope of Future Work

In the future, there will be plenty of space for the Database Controller to grow and improve. It might also be investigated to integrate machine learning algorithms for predictive analysis and cloud-based service integration. Moreover, the script's functionality may be expanded and improved with the help of user input and continuous community contributions, guaranteeing that it can be adjusted to meet changing database administration requirements. The research lays the groundwork for a flexible and dynamic tool that can adapt to the changing needs of contemporary database administration.

References

Course Teacher's Lecture, Lab manuals, Youtube, Various Internet Sites, etc.