ResiStay

Session 2023 – 2027

Submitted by:

ESA IRFAN 2023-CS-104

Supervised by:

Mam Maida Shahid

Course:

CSC-102 Programming Fundamental

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Contents

Figures

1.1 About ResiStay:

The **ResiStay** is a user-friendly platform with distinct logins for students, resident tutors, and senior wardens. It streamlines hostel operations by allowing students to manage profiles, request leave, and submit complaints. Resident tutors oversee students in their respective hostels, while senior wardens coordinate across all hostels, ensuring efficient management.

1.1.1 Why ResiStay?

The ResiStay is a preferred choice due to its user-friendly interface, which simplifies administrative tasks, such as room assignments, leave requests, and complaint submissions. It streamlines processes, saving time and reducing manual effort for students and staff. Effective communication through notifications and updates keeps all stakeholders informed. Senior wardens benefit from centralized control over multiple hostels, facilitating efficient management. The app contributes to an enhanced hostel experience by offering a structured and organized platform.

1.1.2 Expectations from ResiStay:

Users can expect an efficient and user-friendly experience with the Hostel Management App. The platform simplifies various administrative tasks, such as room assignments, leave requests, and complaint submissions, reducing manual work and saving time. Effective communication is facilitated through timely notifications and updates, ensuring that students, resident tutors, and senior wardens are well-informed. With centralized control, senior wardens can oversee multiple hostels, contributing to more organized and effective management. The app's structured and organized platform enhances the overall hostel experience for all users. Additionally, access to student data, leave records, and complaint history simplifies issue resolution and trend monitoring, promoting transparency and a smoother living environment.

1.1.3 Contribution towards CS

ResiStay make substantial contributions to the field of computer science by incorporating various principles and techniques. These applications leverage optimization algorithms for efficient room allocation, a fundamental aspect of computer science that involves balancing multiple factors to arrive at the most optimal solution. Additionally, the integration of robust security measures, such as encryption and secure authentication processes, addresses the critical domain of cybersecurity, emphasizing the importance of safeguarding personal data within these systems. The development of user-friendly interfaces within hostel apps follows principles of Human-Computer Interaction (HCI) and user experience design, fostering advancements in the application of HCI methodologies. These apps also play a role in database management, handling extensive student information and transaction records, contributing to the broader field of database systems and data modeling. The use of machine learning algorithms for predictive analysis further extends their impact, providing insights into student behavior and resource utilization, thereby contributing to the evolving landscape of artificial intelligence and machine learning in computer science.

1.2 User Types on ResiStay:

ResiStay Studios provides different access for different type of users. The users are divided into three categories, Students, Resident tutor and Senior Warden (Admin). Each user type has access to different types of command related to their need and requirements. User can login and verify their type by inputting the issued username and password for authentication. Furthermore, each client type user has different database connected to it. So that data is stored and kept for specific person.

The hierarchy and functionality of user types is as under: -

1.2.1 Warden (Owner):

The Warden Profile functionality encompasses a comprehensive suite of features, allowing wardens to efficiently oversee hostel operations. Wardens can check the availability of students, allot rooms, manage room occupancy, and access detailed student data. The module includes tools for statistical analysis, presenting graphical representations of hostel statistics for informed decision-making. Wardens can also take actions such as removing students or resident tutors, adding new resident tutors, making announcements, displaying and managing rules, changing passwords securely, and logging out. This multifaceted functionality equips wardens with the necessary tools to maintain order, address issues promptly, and ensure the overall well-being of the hostel community.

1.2.2 Resident Tutor:

The Resident Tutor (RT) Profile module is designed to empower RTs with tools for effective hostel management. RTs can conveniently view the list of students assigned to them, set and enforce rules, and access the mess menu to facilitate a smooth living experience for the residents. Furthermore, the RT can stay informed about announcements, hostel events, and rules, enhancing communication and rule adherence. RTs can play a pivotal role in resolving complaints, making announcements, reviewing student complaints, and updating their passwords. This feature-rich functionality aims to streamline the RT's responsibilities and foster a positive and organized living environment.

1.2.3 Student:

The Student Profile functionality caters to the diverse needs of hostel residents. Students can effortlessly navigate through options such as checking their profile details, submitting complaints or requests for issue resolution, reviewing and understanding hostel rules, and staying updated on notices and announcements. Additionally, students have quick access to the mess menu, hostel events calendar, and can efficiently check replies to their submitted complaints. The module further allows students to mark their attendance, pay their mess bills, change their passwords securely, and log out seamlessly, promoting a user-centric and interactive platform for managing various facets of hostel life.

1.3 Functional Requirements of CrossFit Studios:

Some of the functional requirements expected from CrossFit Studios are as under

User Type	Required Functions to be performed	Result of Action Performed
1 – Warden	Check available student	The Students which are not allotted in hostels are shown in tabular form.
	Allotment of room	Allot a room to the students in their respective hostels according to their study year and gender.
	Available rooms	Display all the Rooms that are unallotted.
	Students data	Display students which are allotted w.r.t hostels are shown in tabular form.
	Show hostel statistics graph	Display the graph of number of all hostel residents w.r.t their respective hostels (Edhi, Khalid and Ayesha).
	Remove student	Remove the student and his respective data from all parallel arrays.
	Remove RT	Remove the rt and his respective data from all parallel arrays.
	Add RT	Add the rt and set all the default data in his respective parallel arrays.

1 - Warden	Announce something	Set the announcements to resident tutors and students.
	Display announcements	Display the announcements sets by both warden and Rt.
	Set rules	Set the rules for resident tutors and students.
	Display Rules	Display the rules that are implemented by both warden and Rt.
	Delete rules	Delete the selected rule from the lists of rules.
	Change password	Change the password after verification of old password.
	Log out	Warden can log out from his menu and going back to login Menu.
2-Resident Tutor	View assigned students	Display all the students that are assigned to his hostel.
	Set rules	Set the rules for students.
	Display rules	Display the rules that are implemented by both warden and Rt.
	Announce something	Set the announcements to students.
	Display announcements	Display the announcements sets by both warden and Rt.

2-Resident Tutor	Displaying complains	Display complains from only those students which are assigned to his hostel
	Giving response to complains	Give a auto response to only those students which are assigned to his hostel
	View mess menu	Display the menu
	View events menu	Display the menu
	Change password	Change the password after verification of old password.
	Log out	Warden can log out from his menu and going back to login Menu.
3-Student	Students Profile	Student can check his profile which includes about his details of room and hostel.
	Submitting the complains	Student can submit complains to their respective resident tutors.
	Checking the reply for complains	Student can check the specific reply after different processing of complains
	Display announcements	Display the announcements sets by both warden and Rt.
	Display rules	Display the rules that are implemented by both warden and Rt.
	Mark mess attendance	Student can mark the attendance until the bill become 1000. After this bill, student will have to pay bill first

3-Student	Pay mess bill	Student can pay the bill which was
	View mess menu	Display the menu
	View events menu	Display the menu
	Change password	Change the password after verification of old password.
	Log out	Student can log out from his menu and going back to login Menu.

1.4 Data Structure:

Parallel 1-dimensional array data structure is implied in the current version of ResiStay. Data storing is organized in an orderly fashion way, having related data stored in a single file using CSV format and different data stored in different files. Files are named in an organized way to improve readability.

- int arrsize = 1000;
- string usernames[arrsize];
- string roles[arrsize];
- string passwords[arrsize];
- string years[arrsize];
- string genders[arrsize];
- string alloteds[arrsize];
- string hostels[arrsize];
- string aggregates[arrsize];
- string iscomplained[arrsize];
- string isReplyed[arrsize];
- string complains[arrsize];
- int attendance[arrsize];
- int messBill[arrsize];
- int rooms[arrsize];
- int edhiRoomIndex = 30;

- int edhiRooms[edhiRoomIndex];
- int khalidRoomIndex = 30;
- int khalidRooms[khalidRoomIndex];
- int ayeshaIndex = 30;
- int ayeshaRooms[ayeshaIndex];
- int rulesIndex = 0;
- string rules[20];
- int annoucementIndex = 0;
- string annoucemet [20];
- int userCount = 0;
- int loginOption = 0;
- bool valid = false;
- int a = 0;
- string name;
- string password;
- string role;
- string gender;
- string aggregate;
- string year;

1.5 Wireframes:

The following is the wireframe of ResiStay Displayed in Command Line Interface:

1.5.1 Basic Features (For All types of Users):

1.5.1.1 Startup Face:

Startup interface is filled with animations to provide a premium experience to user about using the application.

1.5.1.1.1 Main Header

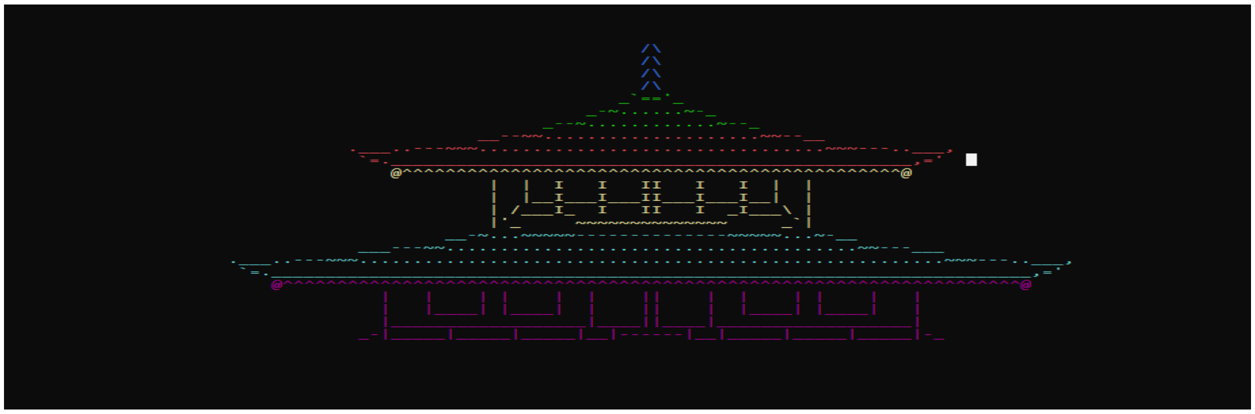
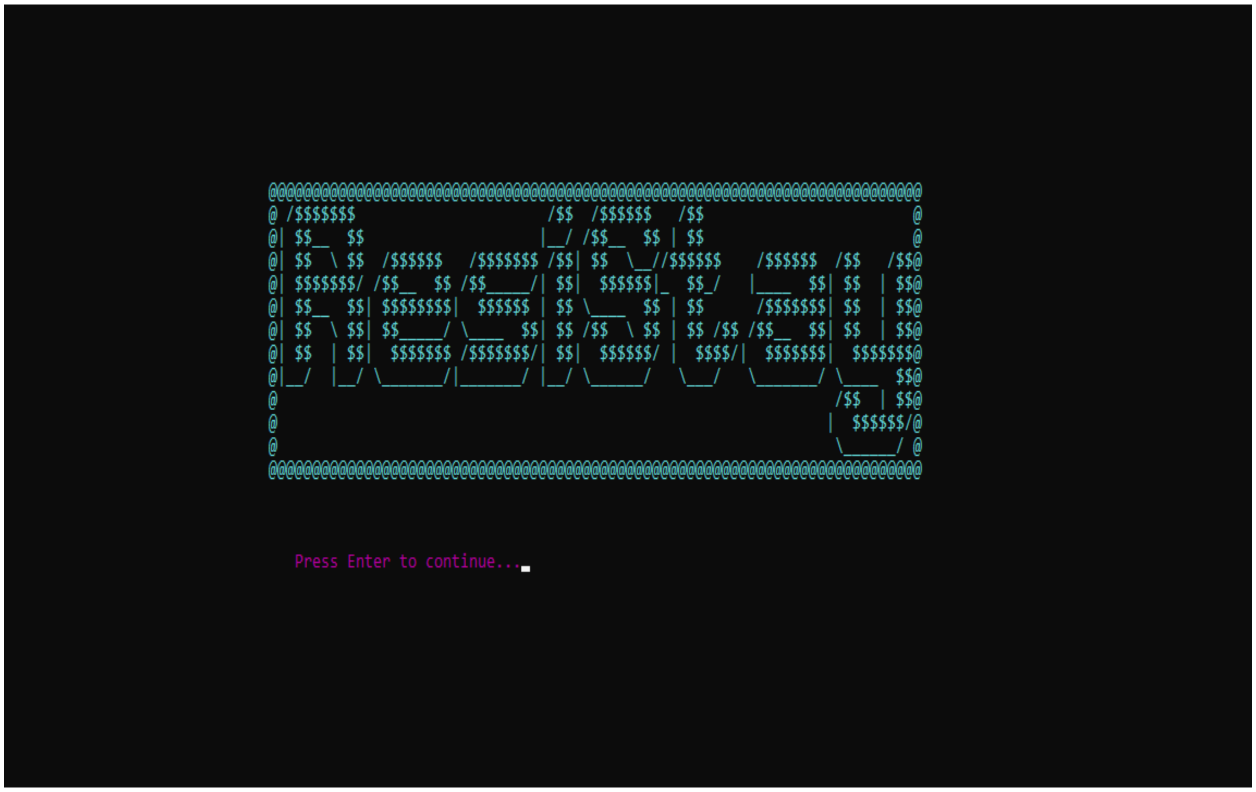


Figure 1 Main Header

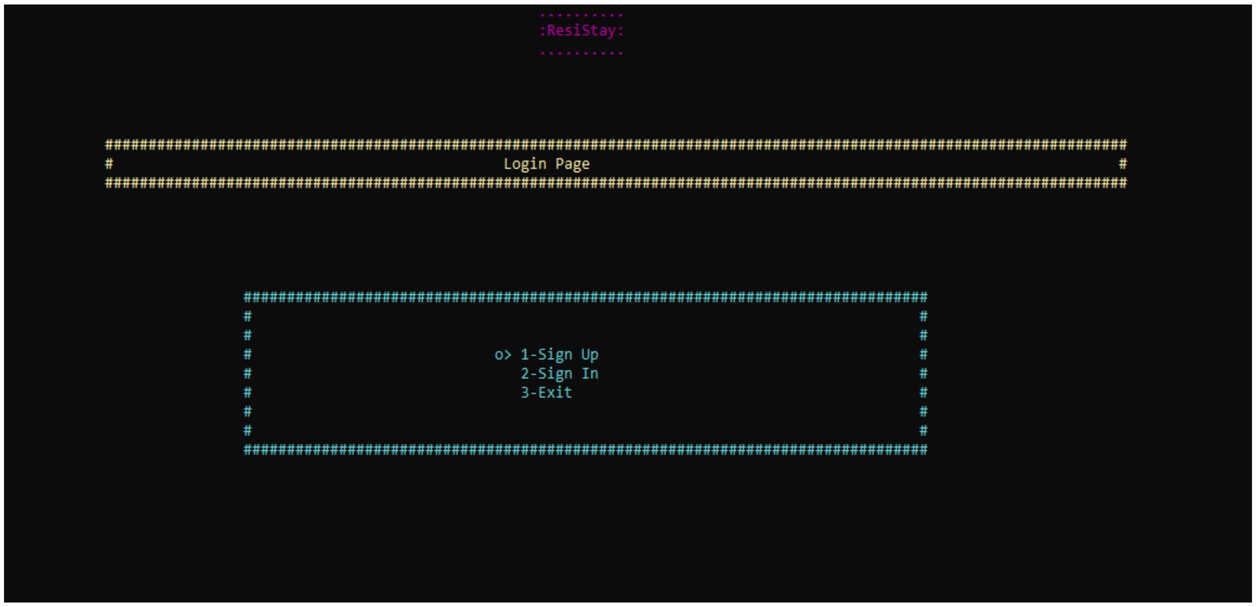
1.5.1.1.2 Main Header 2



```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ /$$$$$$$                      /$$  /$$$$$$   /$$                        @
@| $$__  $$                    |__/ /$$__  $$ | $$                        @
@| $$  \ $$  /$$$$$$   /$$$$$$$ /$$| $$  \__//$$$$$$    /$$$$$$  /$$   /$$@
@| $$$$$$$/ /$$__  $$ /$$_____/| $$|  $$$$$$|_  $$_/   |____  $$| $$  | $$@
@| $$__  $$| $$$$$$$$|  $$$$$$ | $$ \____  $$ | $$      /$$$$$$$| $$  | $$@
@| $$  \ $$| $$_____/ \____  $$| $$ /$$  \ $$ | $$ /$$ /$$__  $$| $$  | $$@
@| $$  | $$|  $$$$$$$ /$$$$$$$/| $$|  $$$$$$/ |  $$$$/|  $$$$$$$|  $$$$$$$@
@|__/  |__/ \_______/|_______/ |__/ \______/   \___/   \_______/ \____  $$@
@                                                                /$$  | $$@
@                                                               |  $$$$$$/@
@                                                                \______/ @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@


   Press Enter to continue...
```

Figure 2 Main Header 2

1.5.1.2 Login Menu

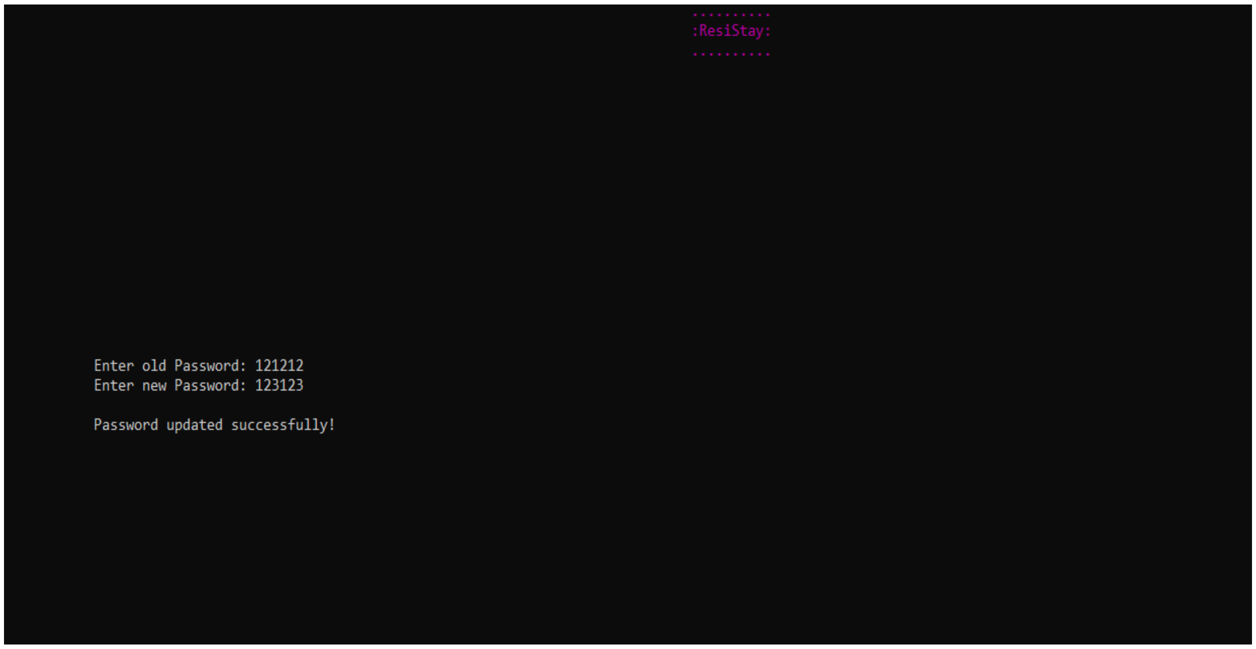


```
                                                 ..........
                                                 :ResiStay:
                                                 ..........



#####################################################################################################################
#                                            Login Page                                                             #
#####################################################################################################################



                ###############################################################################
                #                                                                             #
                #                                                                             #
                #                            o> 1-Sign Up                                     #
                #                               2-Sign In                                     #
                #                               3-Exit                                        #
                #                                                                             #
                #                                                                             #
                ###############################################################################
```

Figure 3 Login Menu

1.5.1.3 Change Password

Change Password Menu verifies logged in user old password and authenticate password twice to minimize human error during password change.

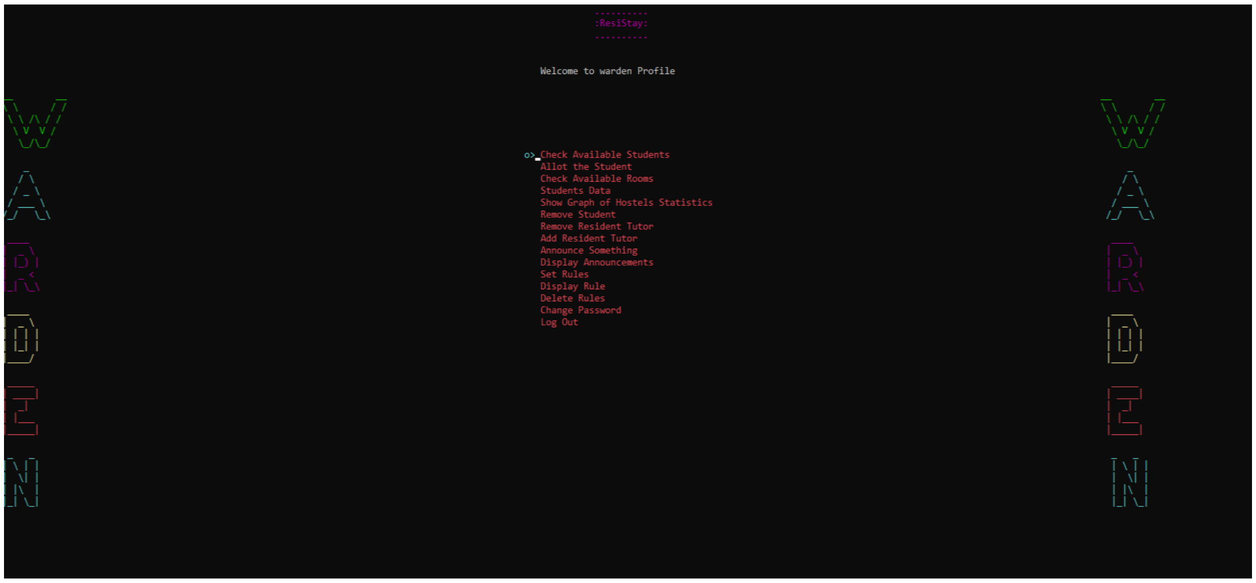


```
..........
:ResiStay:
..........

Enter old Password: 121212
Enter new Password: 123123

Password updated successfully!
```

Figure 4 Change Password

1.5.2 User Type (Warden):

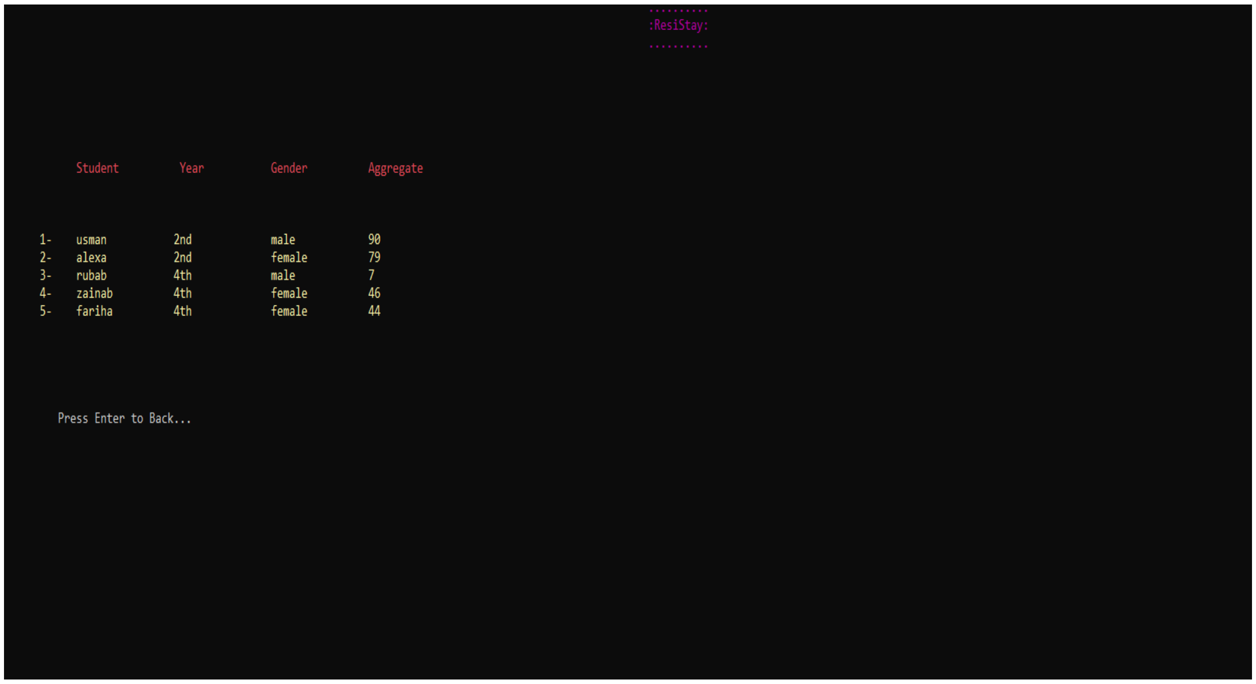
1.5.2.1 Warden Menu



```
..........
:ResiStay:
..........

Welcome to warden Profile

o> Check Available Students
   Allot the Student
   Check Available Rooms
   Students Data
   Show Graph of Hostels Statistics
   Remove Student
   Remove Resident Tutor
   Add Resident Tutor
   Announce Something
   Display Announcements
   Set Rules
   Display Rule
   Delete Rules
   Change Password
   Log Out
```

Figure 5 Warden Menu

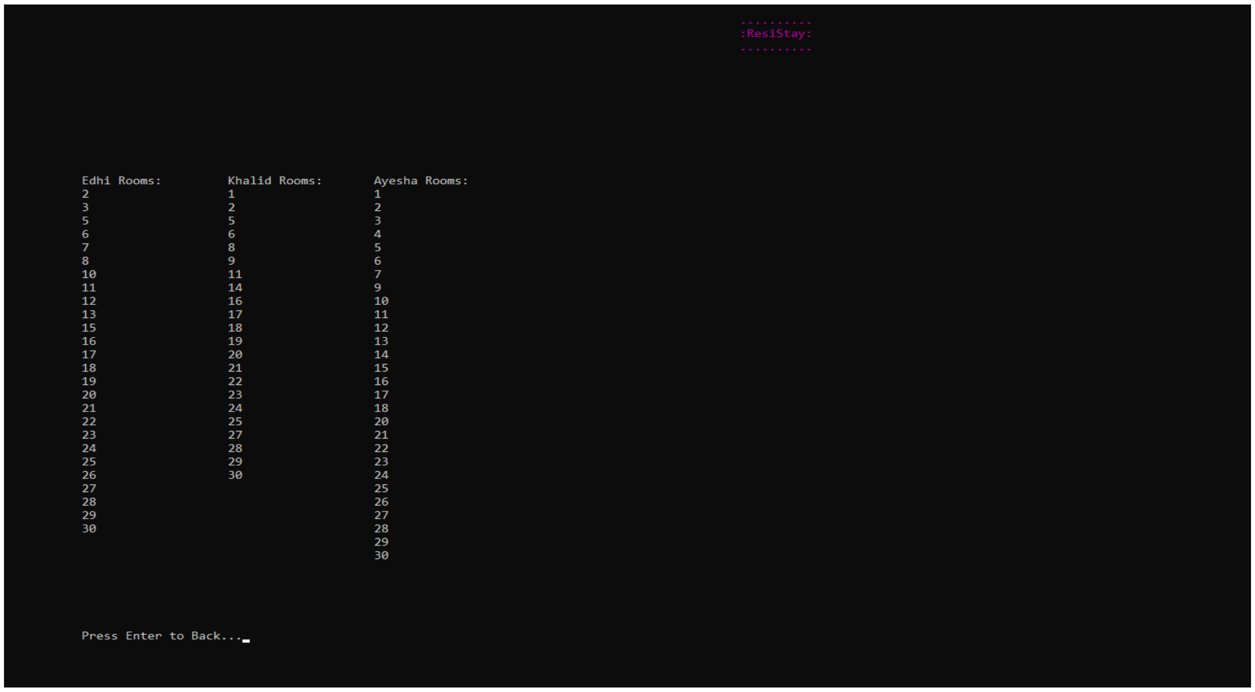
1.5.2.2 Available Student



```
                                                                                  ..........
                                                                                  :ResiStay:
                                                                                  ..........




        Student        Year          Gender         Aggregate


   1-   usman          2nd           male           90
   2-   alexa          2nd           female         79
   3-   rubab          4th           male           7
   4-   zainab         4th           female         46
   5-   fariha         4th           female         44




      Press Enter to Back...
```

Figure 6 Available Students

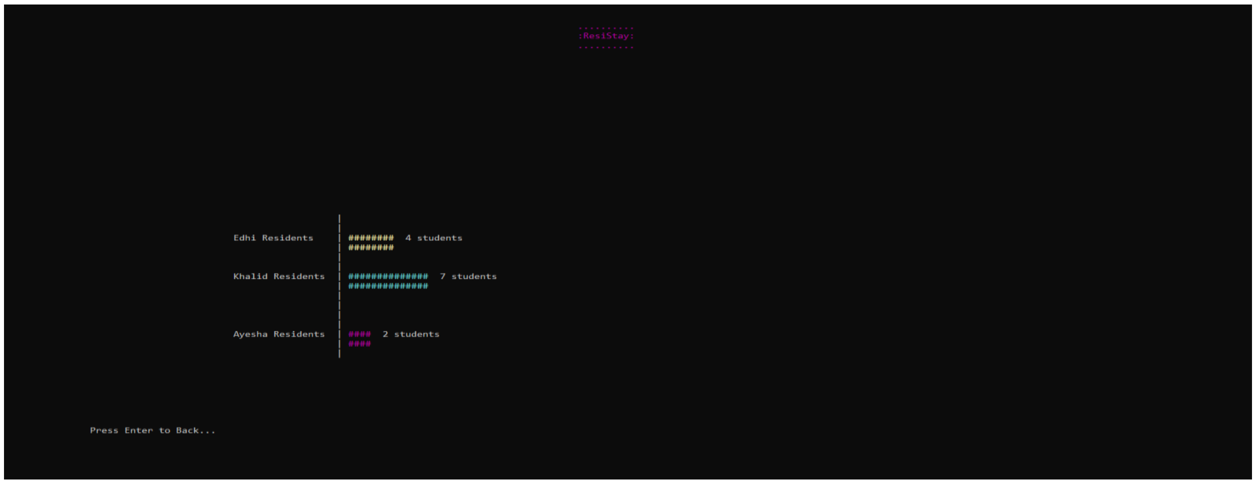
1.5.2.3 Available Room



```
                                                                                          ..........
                                                                                          :ResiStay:
                                                                                          ..........




          Edhi Rooms:         Khalid Rooms:       Ayesha Rooms:
          2                   1                   1
          3                   2                   2
          5                   5                   3
          6                   6                   4
          7                   8                   5
          8                   9                   6
          10                  11                  7
          11                  14                  9
          12                  16                  10
          13                  17                  11
          15                  18                  12
          16                  19                  13
          17                  20                  14
          18                  21                  15
          19                  22                  16
          20                  23                  17
          21                  24                  18
          22                  25                  20
          23                  27                  21
          24                  28                  22
          25                  29                  23
          26                  30                  24
          27                                      25
          28                                      26
          29                                      27
          30                                      28
                                                  29
                                                  30



          Press Enter to Back...
```

Figure 7 Available Rooms

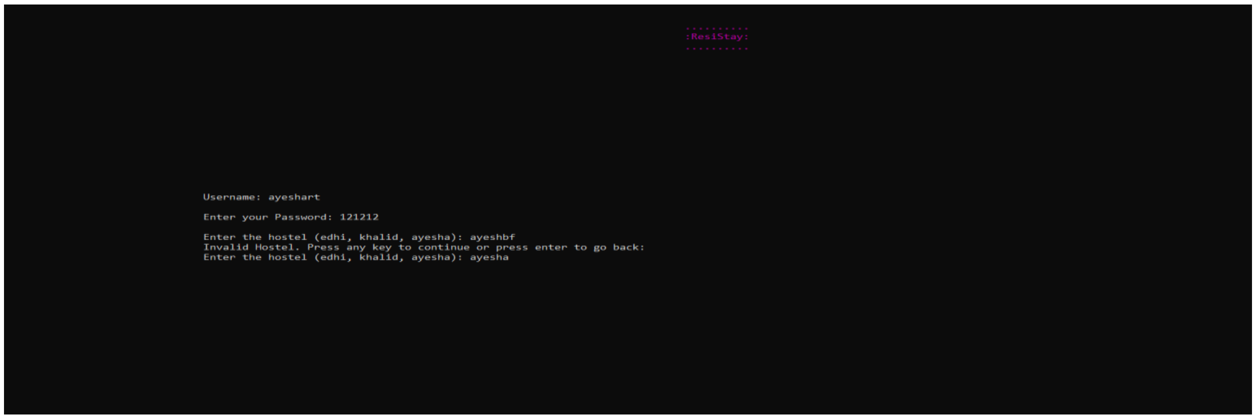
1.5.2.4 Graphs of Hostel Statistics



```
                                                    ..........
                                                    :ResiStay:
                                                    ..........

                         |
                         |
   Edhi Residents        | ########  4 students
                         | ########
                         |
                         |
   Khalid Residents      | ##############  7 students
                         | ##############
                         |
                         |
                         |
                         |
   Ayesha Residents      | ####  2 students
                         | ####
                         |

Press Enter to Back...
```

Figure 8 Graph

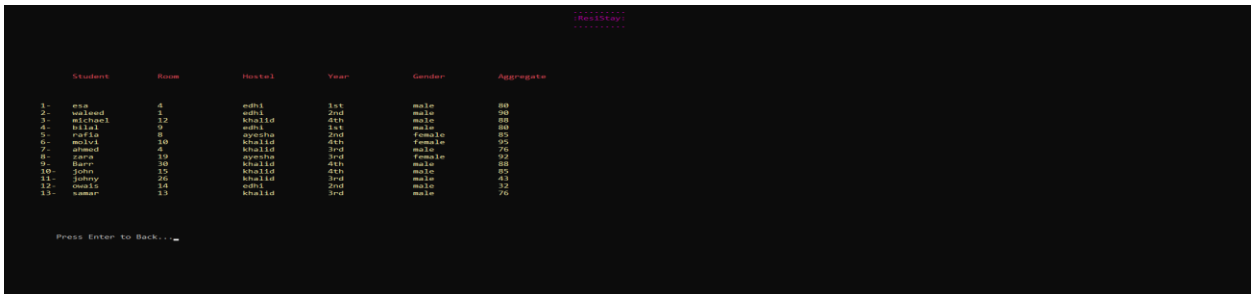
1.5.2.5 Adding Resident Tutor



```
                                                    ..........
                                                    :ResiStay:
                                                    ..........

Username: ayeshart

Enter your Password: 121212

Enter the hostel (edhi, khalid, ayesha): ayeshbf
Invalid Hostel. Press any key to continue or press enter to go back:
Enter the hostel (edhi, khalid, ayesha): ayesha
```

Figure 9 Adding RT

1.5.2.6 Students Data

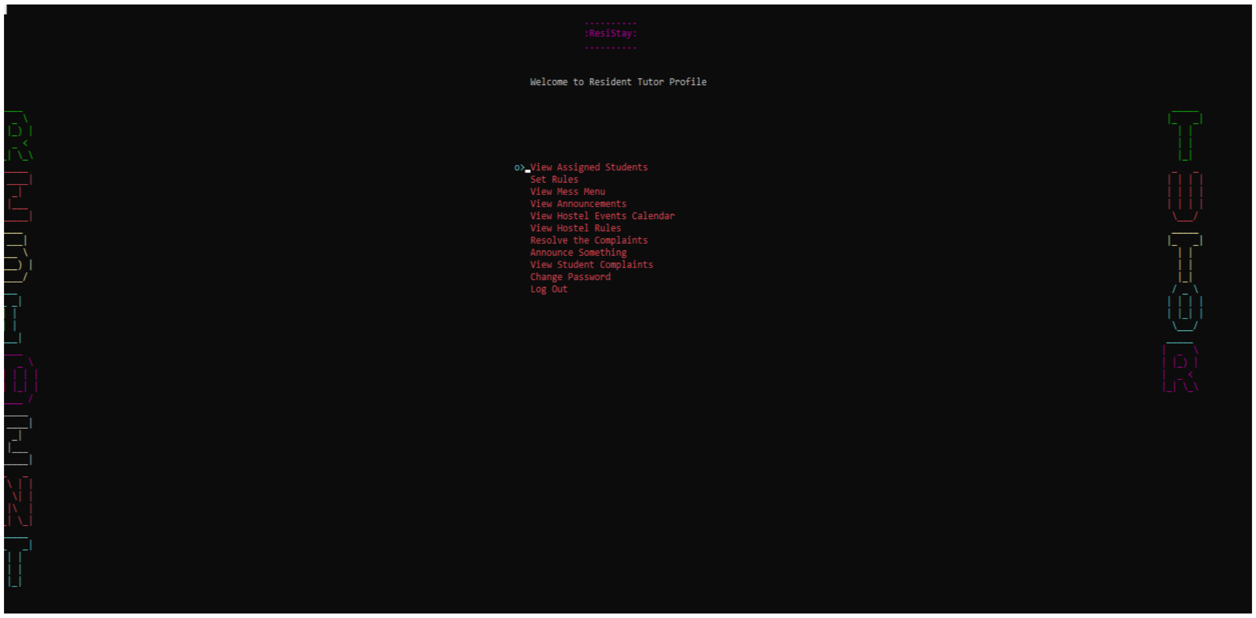


```
                                                                      ..........
                                                                      :ResiStay:
                                                                      ..........

        Student      Room         Hostel       Year         Gender       Aggregate


   1-   esa          4            edhi         1st          male         80
   2-   waleed       1            edhi         2nd          male         90
   3-   michael      12           khalid       4th          male         88
   4-   bilal        9            edhi         1st          male         80
   5-   rafia        8            ayesha       2nd          female       85
   6-   molvi        10           khalid       4th          female       95
   7-   ahmed        4            khalid       3rd          male         76
   8-   zara         19           ayesha       3rd          female       92
   9-   Barr         30           khalid       4th          male         88
   10-  john         15           khalid       4th          male         85
   11-  johny        26           khalid       3rd          male         43
   12-  owais        14           edhi         2nd          male         32
   13-  samar        13           khalid       3rd          male         76


     Press Enter to Back...
```

Figure 10 Students Data

1.5.3 User Type (Resident Tutor)

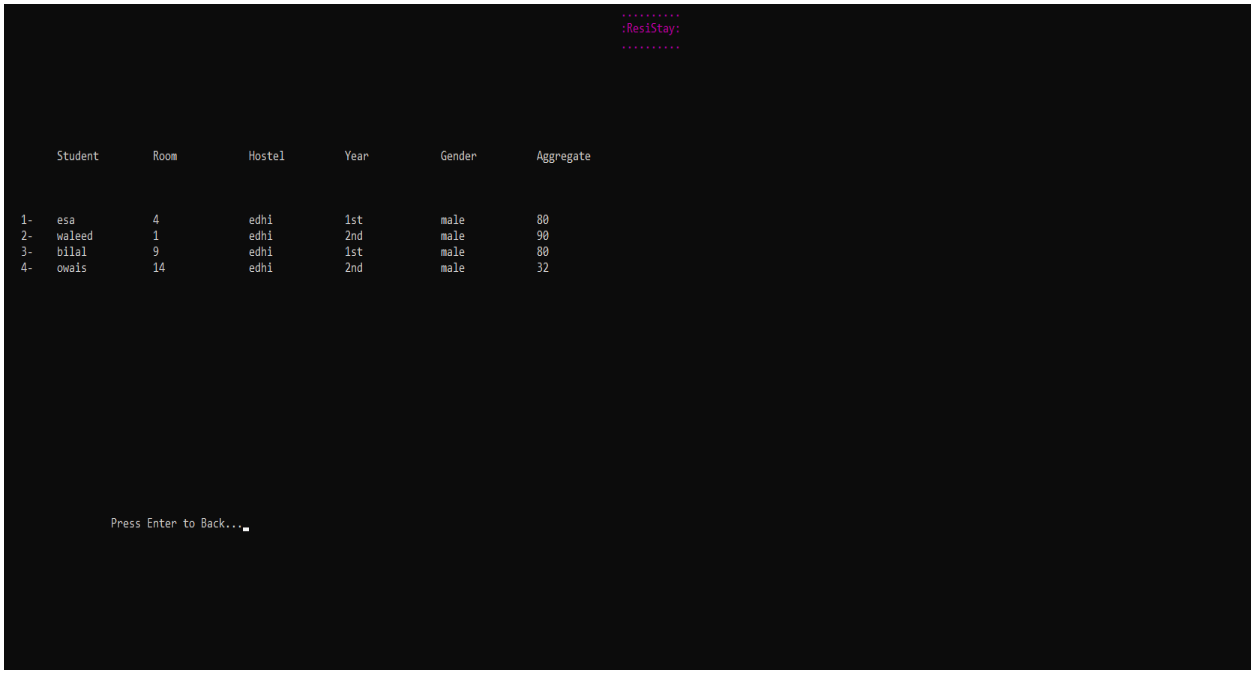
1.5.3.1 Rt Menu



```
                         ..........
                         :ResiStay:
                         ..........

               Welcome to Resident Tutor Profile



           o> View Assigned Students
              Set Rules
              View Mess Menu
              View Announcements
              View Hostel Events Calendar
              View Hostel Rules
              Resolve the Complaints
              Announce Something
              View Student Complaints
              Change Password
              Log Out
```

Figure 11 RT Menu

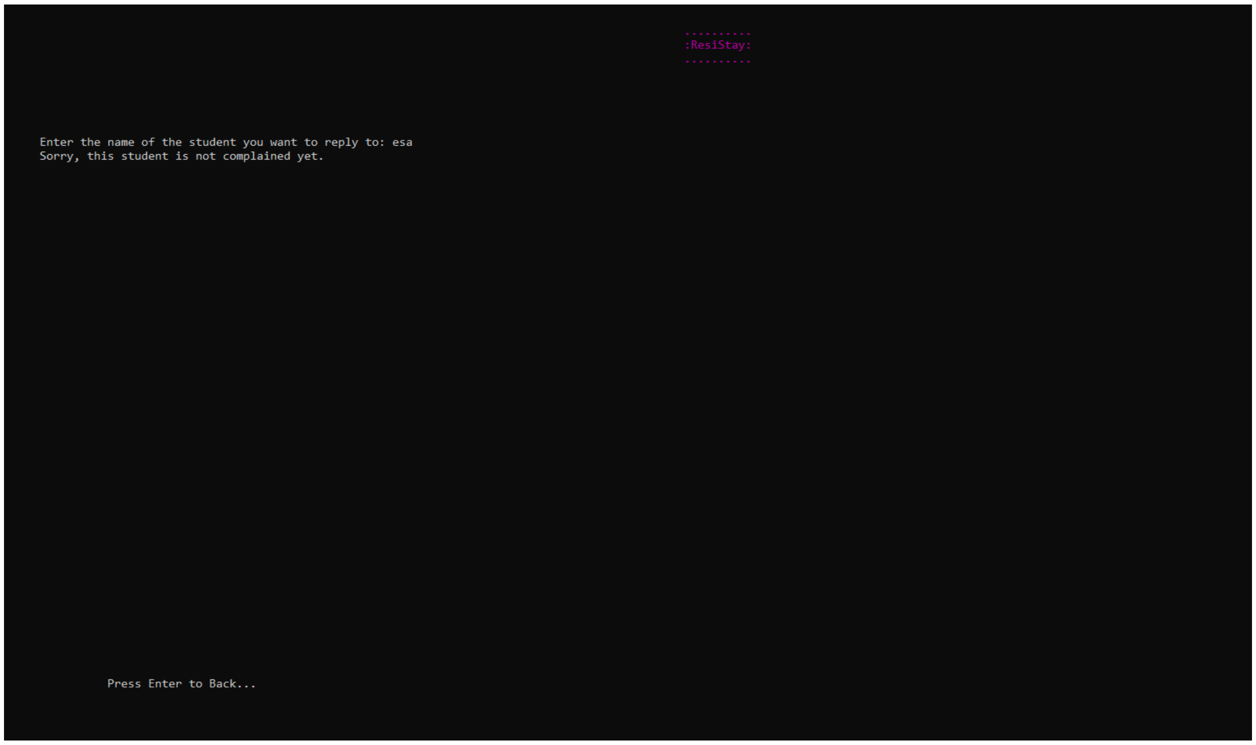
1.5.3.2 Assigned Students



```
                                                      ..........
                                                      :ResiStay:
                                                      ..........


      Student      Room        Hostel      Year        Gender      Aggregate


1-    esa          4           edhi        1st         male        80
2-    waleed       1           edhi        2nd         male        90
3-    bilal        9           edhi        1st         male        80
4-    owais        14          edhi        2nd         male        32



            Press Enter to Back...
```

Figure 12 Assigned Students

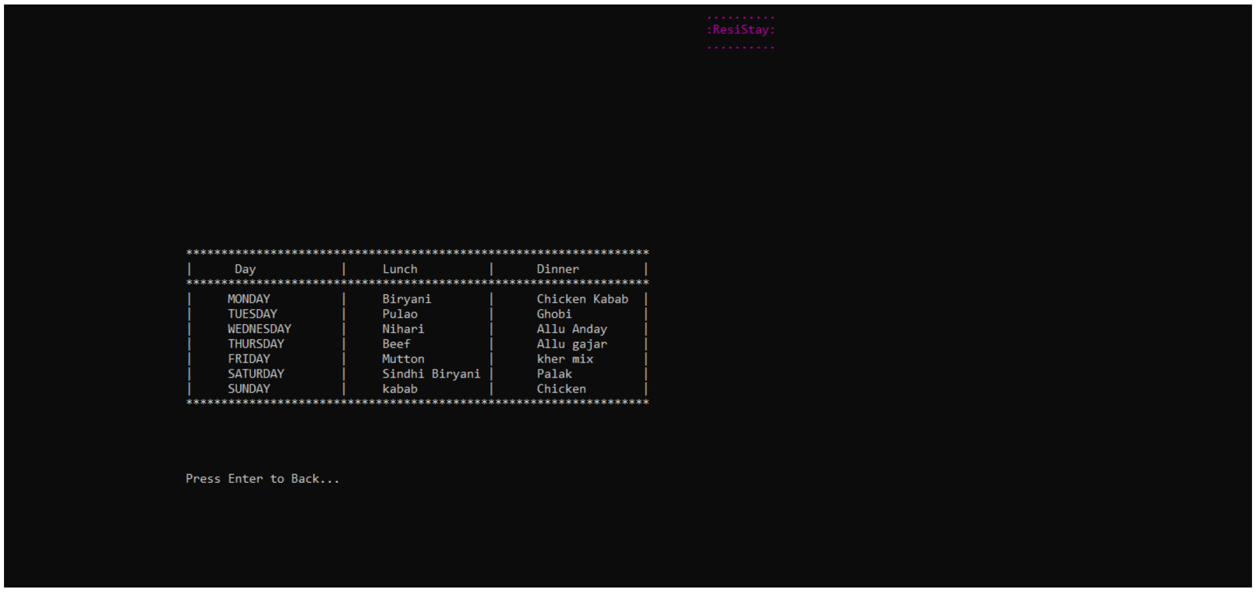
1.5.3.3 Giving Reply



```
..........
:ResiStay:
..........

Enter the name of the student you want to reply to: esa
Sorry, this student is not complained yet.

         Press Enter to Back...
```

Figure 13Giving Reply

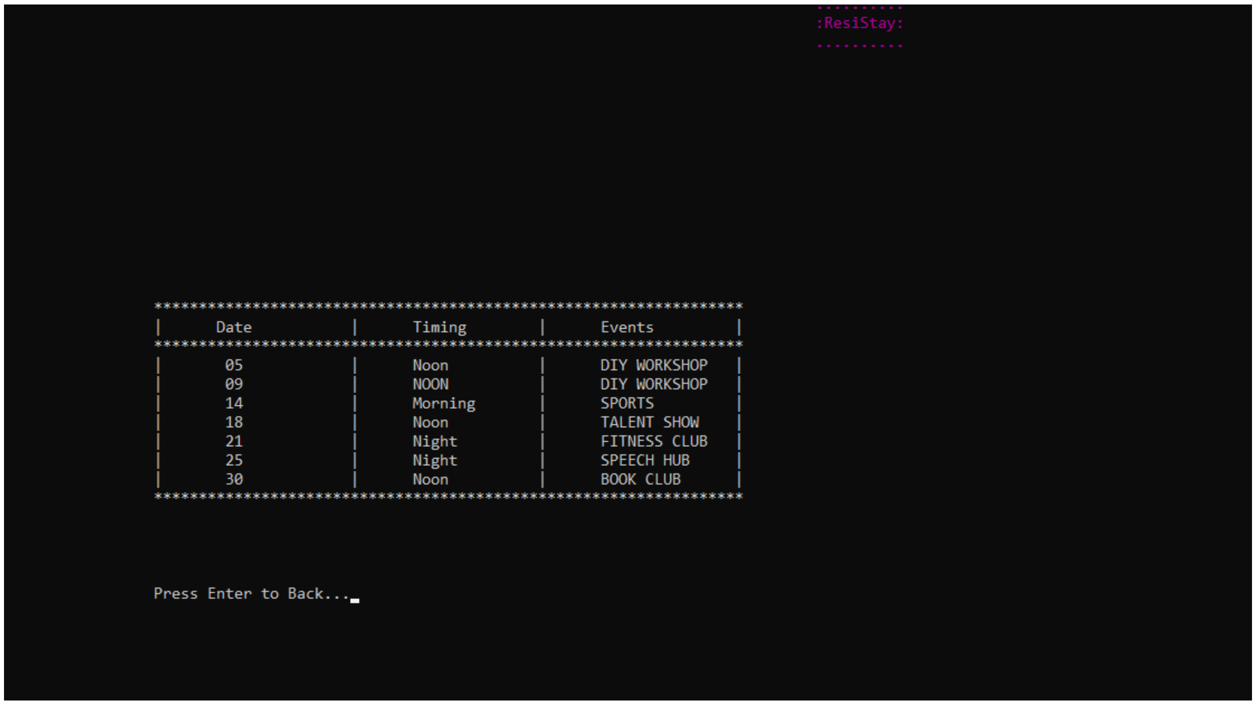
1.5.3.4 Mess Menu



```
                                                                         ..........
                                                                         :ResiStay:
                                                                         ..........

******************************************************************
|       Day            |      Lunch           |      Dinner       |
******************************************************************
|      MONDAY          |      Biryani         |      Chicken Kabab |
|      TUESDAY         |      Pulao           |      Ghobi         |
|      WEDNESDAY       |      Nihari          |      Allu Anday    |
|      THURSDAY        |      Beef            |      Allu gajar    |
|      FRIDAY          |      Mutton          |      kher mix      |
|      SATURDAY        |      Sindhi Biryani  |      Palak         |
|      SUNDAY          |      kabab           |      Chicken       |
******************************************************************

Press Enter to Back...
```

Figure 14 Mess Menu

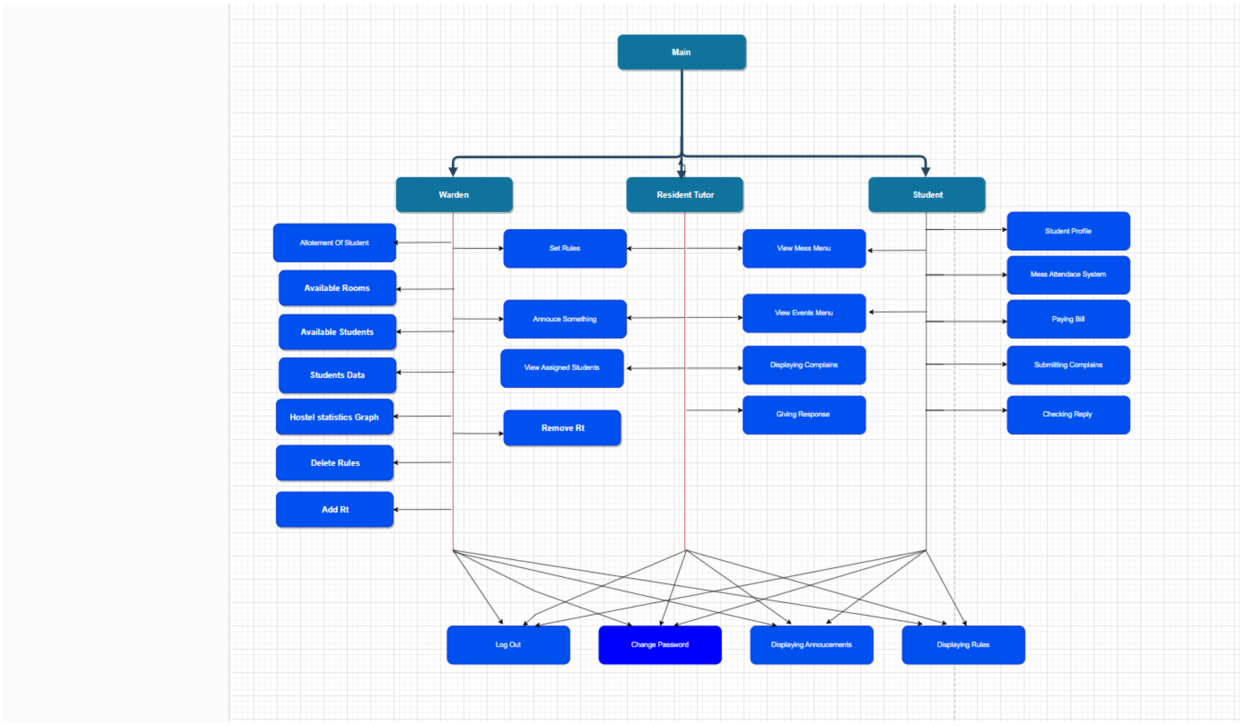
1.5.3.5 Events Menu



```
                                                                                    :ResiStay:

****************************************************************
|      Date          |      Timing       |      Events         |
****************************************************************
|       05           |      Noon         |      DIY WORKSHOP   |
|       09           |      NOON         |      DIY WORKSHOP   |
|       14           |      Morning      |      SPORTS         |
|       18           |      Noon         |      TALENT SHOW    |
|       21           |      Night        |      FITNESS CLUB   |
|       25           |      Night        |      SPEECH HUB     |
|       30           |      Noon         |      BOOK CLUB      |
****************************************************************


Press Enter to Back...
```

Figure 15 Events Menu

1.6 Flow Chart:



1.7 Function Prototypes:

ResiStay is built by keeping Single Responsibility of Functions in view. Each function is tried best to be independent of other function. Following are the Function Prototypes Used:

- void printMainHeader ();
- void loading Function ();
- void secondHeader ();
- bool isValidName (const string &username, string namearr [], int &a, int count);
- bool isValidPassword (const string &password, string passwordarr [], int &a, int count);
- bool isValidRole (const string role);
- bool isValidGender (const string gender);
- bool isValidYear (const string year);
- bool isValidAggregate (const string aggregate);
- string signInConditions (const string username, string namearr [], const string password, string passwordarr [], int count, string rolearr []);
- int movementOfArrow (int x, int y, int minOption, int maxOption);
- bool updatePassword (const string &username, const string &oldPassword, const string &newPassword, string namearr [], string passarr [], int count);
- void gotoxy (int x, int y);
- string lowerLetters (string word);
- void readDataFromFileInParallelArrays (string usernames [], string passwords [], string hostels [], string genders [], string years [], string alloteds [], string role [], string aggregates [], int attendance [], int rooms [], int messBill [], string iscomplained [], string isReplyed [], string complains [], int &userCount);
- string getField (string record, int field);
- void UpdataDataToFileForParalleArrays(const string usernames[], const string passwords[], const string hostels[], const string genders[], const string years[], const string alloteds[], const string role[], const string aggregates[], const int attendance[], const int rooms[], const int messBill[], const string iscomplained[], const string isReplyed[], const string complains[], const int userCount);
- void readdatafromFiles(int array[], int &index, const char *filename);
- void storedatainFiles(int array[], int size, const char *filename);
- void readdatafromstringFiles(string array[], int &index, const char *filename);
- void storestringdatainFiles(string array[], int size, const char *filename);
- void setConsoleColor(int color);
- int loginPage();
- int studentProfile();
- int rtProfile();
- int wardenProfile();
- void drawArt();

- void rtLogo();
- void studentLogo();
- void wardenLogo();
- void printMessMenu();
- void printEventsMenu();
- void settingRules(int &rulesIndex, string rules[], string role);
- void displayingRules(int rulesIndex, string rules[]);
- bool deletingRule(int deletedrule, int &rulesIndex, string rules[]);
- void AnnouceSomething(int &annoucementIndex, string annoucement[], string role);
- void displayingAnnoucement(int annoucementIndex, string annoucement[]);
- void submittingComplains(string complains[], string isComplained[], string isReplyed[], string username, string namearray[], int count);
- void checkingReplyOnComplain(string isComplained[], string isReplyed[], string username, string namearray[], int count);
- void displayingComplains(string name, string username[], string complains[], string isComplained[], string hostel[], string role[], int count, string isReplyed[]);
- void replyToComplains(string complains[], string isComplained[], string isReplyed[], string role[], int count, string hostel[], string name, string namearray[]);
- void availabilityofstudent(string username[], string year[], string gender[], string aggregate[], string alloted[], int count);
- void checkAvilablityOfRoom(int edhiRoom[], int khalidRoom[], int ayesharoom[], int edhiIndex, int khalidIndex, int ayeshaIndex);
- void allotementOfStudent(int count, string namearr[], string isAlloted[], int edhiRoom[], int khalidRoom[], int ayesharoom[], string hostel[], int room[], int &edhiIndex, int &khalidIndex, int &ayeshaIndex);
- void studentData(string username[], string year[], string gender[], string aggregate[], string alloted[], int room[], string hostel[], string role[], int count);
- void markTheAttendance(int count, string name, string namearray[], int attendance[], int messBill[]);
- void payingMessBill(int count, string name, string namearray[], int attendance[], int messBill[]);
- void addingNewRt(string usernames[], string passwords[], string hostels[], string genders[], string years[], string alloteds[], string role[], string aggregates[], int attendance[], int rooms[], int messBill[], string iscomplained[], string isReplyed[], string complains[], int &userCount);
- void removingOfRt(string username[], string year[], string gender[], string aggregate[], string alloted[], int room[], string hostel[], string role[], string password[], string iscomplained[], string isReplyed[], int attendance[], int messBill[], string complains[], int &count);
- void removingOfStudent(string username[], string year[], string gender[], string aggregate[], string alloted[], int room[], string hostel[], string role[], string password[], string iscomplained[], string isReplyed[], int attendance[], int messBill[], string

complains[], int &count, int edhiRoom[], int khalidRoom[], int ayesharoom[], int &edhiIndex, int &khalidIndex, int &ayeshaIndex);

- void checkingStudentProfile(string name, string username[], string year[], string gender[], string aggregate[], string alloted[], int room[], string hostel[], string role[], int count);
- void checkingassignedstudents(string name, string username[], string year[], string gender[], string aggregate[], string alloted[], int room[], string hostel[], string role[], int count);
- int findStudentByName(string name, string usernames[], int userCount);
- void statisticsOfHostelStudents(string allotted[], string role[], string hostel[], int userCount);
- void calculationOfHostelStatistics(int &edhiCount, int &khalidCount, int &ayeshaCount, string allotted[], string role[], string hostel[], int userCount);
- void printHashPattern(int count,int color, int &x, int y);

1.8 Code:

```
 // Libraries

#include <iostream>  //   Input/output stream functions
#include <windows.h> //   For Windows-specific functions, like console manipulation
#include <conio.h>    //   For console input/output functions (getch)
#include <cctype>      //   For character classification functions (e.g., isdigit)
#include <string>       //   For string manipulation and handling
#include <limits>       //   For numeric limits, useful for input validation
#include <fstream>    //   For File Handling
#include <algorithm> //   For Counts the occurrences of a specified value within a range.

using namespace std;

// Functions

// Main Header Function
void printMainHeader();

// loadimg Function
void loadingFunction();

// Second Header Function
void secondHeader();

// validation
bool isValidName(const string &username, string namearr[], int &a, int count);
bool isValidPassword(const string &password, string passwordarr[], int &a, int count);
bool isValidRole(const string role);
bool isValidGender(const string gender);
bool isValidYear(const string year);
bool isValidAggregate(const string aggregate);
string signInConditions(const string username, string namearr[], const string password, string
passwordarr[], int count, string rolearr[]);
int movementOfArrow(int x, int y, int minOption, int maxOption);
bool updatePassword(const string &username, const string &oldPassword, const string
&newPassword, string namearr[], string passarr[], int count);

// Function for going to specific cordinates of console
void gotoxy(int x, int y);

// lowering Of Letters Function
string lowerLetters(string word);

// Functions for file Handling
void readDataFromFileInParallelArrays(string usernames[], string passwords[], string
hostels[], string genders[], string years[], string alloteds[], string role[], string aggregates[], int
attendance[], int rooms[], int messBill[], string iscomplained[], string isReplyed[], string
complains[], int &userCount);
string getField(string record, int field);
void UpdataDataToFileForParalleArrays(const string usernames[], const string passwords[],
const string hostels[], const string genders[], const string years[], const string alloteds[], const
string role[], const string aggregates[], const int attendance[], const int rooms[], const int
messBill[], const string iscomplained[], const string isReplyed[], const string complains[],
const int userCount);
void readdatafromFiles(int array[], int &index, const char *filename);
void storedatainFiles(int array[], int size, const char *filename);
void readdatafromstringFiles(string array[], int &index, const char *filename);
void storestringdatainFiles(string array[], int size, const char *filename);

// Functions for coloring on console
void setConsoleColor(int color);

// Login Functions
int loginPage();

// profiles
int studentProfile();
int rtProfile();
int wardenProfile();

// Animations and Logos

void drawArt(); // Main Animation
void rtLogo();
void studentLogo();
void wardenLogo();

// Functionalities

// Function to print mess menu
void printMessMenu();

// Function to print events menu
void printEventsMenu();

// Functions for setting , displaying and deleting rules
void settingRules(int &rulesIndex, string rules[], string role);
void displayingRules(int rulesIndex, string rules[]);
bool deletingRule(int deletedrule, int &rulesIndex, string rules[]);

// Functions for making and displaying Annouecements
void AnnouceSomething(int &annoucementIndex, string annoucement[], string role);
void displayingAnnoucement(int annoucementIndex, string annoucement[]);

// Functions to submitting complains and checking replying on complains
void submittingComplains(string complains[], string isComplained[], string isReplyed[], string
username, string namearray[], int count);
void checkingReplyOnComplain(string isComplained[], string isReplyed[], string username,
string namearray[], int count);

// Functions for displaying complains and giving replying to it
void displayingComplains(string name, string username[], string complains[], string
isComplained[], string hostel[], string role[], int count, string isReplyed[]);
void replyToComplains(string complains[], string isComplained[], string isReplyed[], string
role[], int count, string hostel[], string name, string namearray[]);

// Allotement Functions
//  Function to check the availability of students
void availabilityofstudent(string username[], string year[], string gender[], string aggregate[],
string alloted[], int count);

// Function to check room availability
void checkAvilablityOfRoom(int edhiRoom[], int khalidRoom[], int ayesharoom[], int
edhiIndex, int khalidIndex, int ayeshaIndex);

// Function for student allotment
void allotementOfStudent(int count, string namearr[], string isAlloted[], int edhiRoom[], int
khalidRoom[], int ayesharoom[], string hostel[], int room[], int &edhiIndex, int &khalidIndex,
int &ayeshaIndex);
```

```
// Function to display student data
void studentData(string username[], string year[], string gender[], string aggregate[], string
alloted[], int room[], string hostel[], string role[], int count);

// Attendance and Bill Function
//  Function to mark attendance
void markTheAttendance(int count, string name, string namearray[], int attendance[], int
messBill[]);

// Function to pay mess bill
void payingMessBill(int count, string name, string namearray[], int attendance[], int
messBill[]);

// Function to add a new resident tutor
void addingNewRt(string usernames[], string passwords[], string hostels[], string genders[],
string years[], string alloteds[], string role[], string aggregates[], int attendance[], int rooms[],
int messBill[], string iscomplained[], string isReplyed[], string complains[], int &userCount);

// Function to remove a resident tutor
void removingOfRt(string username[], string year[], string gender[], string aggregate[], string
alloted[], int room[], string hostel[], string role[], string password[], string iscomplained[],
string isReplyed[], int attendance[], int messBill[], string complains[], int &count);

// Function to remove a student
void removingOfStudent(string username[], string year[], string gender[], string aggregate[],
string alloted[], int room[], string hostel[], string role[], string password[], string
iscomplained[], string isReplyed[], int attendance[], int messBill[], string complains[], int
&count, int edhiRoom[], int khalidRoom[], int ayesharoom[], int &edhiIndex, int
&khalidIndex, int &ayeshaIndex);

// Function to check a student's profile
void checkingStudentProfile(string name, string username[], string year[], string gender[],
string aggregate[], string alloted[], int room[], string hostel[], string role[], int count);

// Function to check assigned students for a specific resident tutor
void checkingassignedstudents(string name, string username[], string year[], string gender[],
string aggregate[], string alloted[], int room[], string hostel[], string role[], int count);

// Function to find a student by name
int findStudentByName(string name, string usernames[], int userCount);

// Function for Graphs
void statisticsOfHostelStudents(string allotted[], string role[], string hostel[], int userCount);
void calculationOfHostelStatistics(int &edhiCount, int &khalidCount, int &ayeshaCount,
string allotted[], string role[], string hostel[], int userCount);
void printHashPattern(int count, int color, int &x, int y);
```

```
// Main
int main()
{

    // Define the maximum size for various arrays
    const int arrsize = 1000;

    // Arrays to store user information
    string usernames[arrsize];
    string roles[arrsize];
    string passwords[arrsize];
    string years[arrsize];
    string genders[arrsize];
    string alloteds[arrsize];
    string hostels[arrsize];
    string aggregates[arrsize];
    string iscomplained[arrsize];
    string isReplyed[arrsize];
    string complains[arrsize];
    int attendance[arrsize];
    int messBill[arrsize];
    int rooms[arrsize];

    // Define the index and available rooms for Edhi Hostel
    int edhiRoomIndex = 30;
    int edhiRooms[edhiRoomIndex];

    // Define the index and available rooms for Khalid Hostel
    int khalidRoomIndex = 30;
    int khalidRooms[khalidRoomIndex];

    // Define the index and available rooms for Ayesha Hostel
    int ayeshaIndex = 30;
    int ayeshaRooms[ayeshaIndex];

    // Initialize index for rules and announcements
    int rulesIndex = 0;
    string rules[20];
    int annoucementIndex = 0;
    string annoucemet[20];

    // Initialize user count
    int userCount = 0;

    // Initialize login option
    int loginOption = 0;

    // Reading the data form required files
    readDataFromFileInParallelArrays(usernames, passwords, hostels, genders, years, alloteds,
roles, aggregates, attendance, rooms, messBill, iscomplained, isReplyed, complains,
userCount);
    readdatafromFiles(khalidRooms, khalidRoomIndex, "Khalid Rooms.txt");
    readdatafromFiles(edhiRooms, edhiRoomIndex, "Edhi Rooms.txt");
    readdatafromFiles(ayeshaRooms, ayeshaIndex, "Ayesha Rooms.txt");
    readdatafromstringFiles(rules, rulesIndex, "rules.txt");
    readdatafromstringFiles(annoucemet, annoucementIndex, "annoucement.txt");

    system("cls");

    // Print the main header of the program
    printMainHeader();

    system("cls");

    // Loop until the user chooses to exit (loginOption == 3)
    while (loginOption != 3)
    {
        // Display the second header
        secondHeader();

        // Get the user's choice from the login page
        loginOption = loginPage();

        // If the user chooses to register
        if (loginOption == 1)
        {
            // Declare variables to store user information
            string name, password, gender, role, year, aggregate;
            bool valid = false;
            int a = 0;

            // Display loading animation
            loadingFunction();

            // Clear the console
            system("cls");
            secondHeader();

            // Set initial y coordinate for user input
            int y = 21;
```

```
// Loop until valid input is received
while (!valid)
{
    a = 0;
    gotoxy(36, y);
    cout << "Username: ";
    getline(cin, name);
    valid = isValidName(name, usernames, a, userCount);

    // Display error messages based on validation result
    if (!valid && a == 1)
    {
        setConsoleColor(12);
        gotoxy(36, y + 1);
        cout << "Invalid Input . Plz enter a correct Username";
        setConsoleColor(7);
    }
    if (!valid && a == 2)
    {
        setConsoleColor(12);
        gotoxy(36, y + 1);
        cout << "Your UserName must not have Spaces and Capital letters";
        setConsoleColor(7);
    }
    if (!valid && a == 3)
    {
        setConsoleColor(12);
        gotoxy(36, y + 1);
        cout << "Your username does not contain any special characters";
        setConsoleColor(7);
    }
    if (!valid && a == 4)
    {
        setConsoleColor(12);
        gotoxy(36, y + 1);
        cout << "Your username does not contain any number";
        setConsoleColor(7);
    }
    if (!valid && a == 5)
    {
        setConsoleColor(12);
        gotoxy(36, y + 1);
        cout << "This name is already taken.";
        setConsoleColor(7);
    }
```

```
        y = y + 2;
    }

    // Store the username in the array
    usernames[userCount] = name;

    // Reset validation flag
    valid = false;

    // Loop until valid input is received for password
    while (!valid)
    {
        a = 0;
        gotoxy(36, y);
        cout << "Enter your Password: ";
        getline(cin, password);
        valid = isValidPassword(password, passwords, a, userCount);

        // Display error messages based on validation result
        if (!valid && a == 1)
        {
            setConsoleColor(12);
            gotoxy(36, y + 1);
            cout << "Your Password must be of at least six characters";
            setConsoleColor(7);
        }
        if (!valid && a == 2)
        {
            setConsoleColor(12);
            gotoxy(36, y + 1);
            cout << "Your Password must be less than twelve characters";
            setConsoleColor(7);
        }
        else if (!valid && a == 3)
        {
            setConsoleColor(12);
            gotoxy(36, y + 1);
            cout << "Your Password must have at least one digit";
            setConsoleColor(7);
        }
        else if (!valid && a == 4)
        {
            setConsoleColor(12);
            gotoxy(36, y + 1);
            cout << "Your Password must have at least one special character";
            setConsoleColor(7);
        }
        else if (!valid && a == 5)
        {
            setConsoleColor(12);
            gotoxy(36, y + 1);
            cout << "Your Password must not contain comma";
            setConsoleColor(7);
        }

        y = y + 2;
    }

    // Store the password in the array
    passwords[userCount] = password;

    // Reset validation flag
    valid = false;

    // Loop until valid input is received for role
    while (!valid)
    {
        gotoxy(36, y);
        cout << "Enter your Role: ";
        getline(cin, role);
        role = lowerLetters(role);

        // Validate the role input
        if (role != "rt" && role != "warden")
        {
            valid = isValidRole(role);
        }
        else if (role == "rt" && role != "warden")
        {
            setConsoleColor(12);
            gotoxy(36, y + 1);
            cout << "You can't register as Resident Tutor.";
            setConsoleColor(7);
        }
        else if (role == "warden" && role != "rt")
        {
            setConsoleColor(12);
            gotoxy(36, y + 1);
            cout << "You can't register as Warden.";
            setConsoleColor(7);
        }
        // Display error message for invalid role
        if (!valid && role != "rt" && role != "warden")
        {
            setConsoleColor(12);
            gotoxy(36, y + 1);
            cout << "Invalid role.";
            setConsoleColor(7);
        }

        y = y + 2;
    }

    // Store the role in the array
    roles[userCount] = role;

    // If the role is a student, gather additional information
    if (roles[userCount] == "student")
    {
        // Reset validation flag
        valid = false;

        // Loop until valid input is received for gender
        while (!valid)
        {
            gotoxy(36, y);
            cout << "Gender (male/female): ";
            getline(cin, gender);
            gender = lowerLetters(gender);
            valid = isValidGender(gender);

            // Display error message for invalid gender
            if (!valid)
            {
                setConsoleColor(12);
                gotoxy(36, y + 1);
                cout << "Invalid Gender.";
                setConsoleColor(7);
            }

            y = y + 2;
        }

        // Store the gender in the array
        genders[userCount] = gender;

        // Reset validation flag
        valid = false;
```

```
// Loop until valid input is received for year
while (!valid)
{
    gotoxy(36, y);
    cout << "Year (1st/2nd/3rd/4th): ";
    getline(cin, year);
    year = lowerLetters(year);

    // Validate the year input
    valid = isValidYear(year);

    // Display error message for invalid year
    if (!valid)
    {
        setConsoleColor(12);
        gotoxy(36, y + 1);
        cout << "Invalid Year.";
        setConsoleColor(7);
    }

    y = y + 2;
}

// Store the year in the array
years[userCount] = year;

// Reset validation flag
valid = false;

// Loop until valid input is received for aggregate
while (!valid)
{
    gotoxy(36, y);
    cout << "Enter your Aggregate: ";
    getline(cin, aggregate);

    // Validate the aggregate input
    valid = isValidAggregate(aggregate);

    // Display error message for invalid aggregate
    if (!valid)
    {
        setConsoleColor(12);
        gotoxy(36, y + 1);
        cout << "Invalid Aggregate.";
```

```
            setConsoleColor(7);
          }

          y = y + 2;
        }

        // Store the aggregate in the array
        aggregates[userCount] = aggregate;

        // Determine the hostel based on year, gender, and role
        if ((years[userCount] == "1st" || years[userCount] == "2nd") && genders[userCount]
== "male")
        {
          hostels[userCount] = "edhi";
        }
        else if ((years[userCount] == "3rd" || years[userCount] == "4th") &&
genders[userCount] == "male")
        {
          hostels[userCount] = "khalid";
        }
        else if (genders[userCount] == "female")
        {
          hostels[userCount] = "ayesha";
        }

        // Set default values for other user information
        alloteds[userCount] = "No";
        iscomplained[userCount] = "No";
        isReplyed[userCount] = "No";
        attendance[userCount] = 0;
        rooms[userCount] = 0;
        messBill[userCount] = 0;
        complains[userCount] = "Default";
      }
      else
      {
        // For warden and rt roles, set default values
        genders[userCount] = "Default";
        years[userCount] = "Default";
        alloteds[userCount] = "Yes";
        hostels[userCount] = "Default";
        aggregates[userCount] = "Default";
        attendance[userCount] = 0;
        rooms[userCount] = 0;
        messBill[userCount] = 0;
        iscomplained[userCount] = "No";
```

```
            isReplyed[userCount] = "No";
            complains[userCount] = " ";
        }

        // Increment the user count
        userCount++;

        // Reset validation flag
        valid = false;
        system("cls");
        secondHeader();
        // Display loading animation
        loadingFunction();
    }

    // If the user chooses to log in
    if (loginOption == 2)
    {
        // Display loading animation
        loadingFunction();

        // Clear the console
        system("cls");
        secondHeader();

        // Declare variables for login
        string name;
        string password;
        string role;
        int y = 21;
        bool valid = false;

        // Loop until valid login is achieved
        while (!valid)
        {
            // Get user input for name and password
            gotoxy(36, y);
            cout << "Enter your Name: ";
            cin >> name;
            gotoxy(36, y + 1);
            cout << "Enter your Password: ";
            cin >> password;

            // Check login conditions and get the role
            role = signInConditions(name, usernames, password, passwords, userCount, roles);
            // For ignoring the line
            cin.ignore(numeric_limits<streamsize>::max(), '\n');

            // If the role is student
            if (role == "student")
            {
                // Display loading animation
                loadingFunction();

                // Clear the console
                system("cls");

                // Loop for student profile options
                while (true)
                {
                    // Get student profile option
                    int studentOption = studentProfile();

                    // Execute corresponding actions based on the option
                    if (studentOption == 1)
                    {
                        checkingStudentProfile(name, usernames, years, genders, aggregates,
    alloteds, rooms, hostels, roles, userCount);
                    }
                    else if (studentOption == 2)
                    {
                        submittingComplains(complains, iscomplained, isReplyed, name, usernames,
    userCount);
                    }
                    else if (studentOption == 3)
                    {
                        displayingRules(rulesIndex, rules);
                    }
                    else if (studentOption == 4)
                    {
                        displayingAnnoucement(annoucementIndex, annoucemet);
                    }
                    else if (studentOption == 5)
                    {
                        printMessMenu();
                    }
                    else if (studentOption == 6)
                    {
                        printEventsMenu();
                    }
                    else if (studentOption == 7)
                    {
                        checkingReplyOnComplain(iscomplained, isReplyed, name, usernames,
    userCount);
                    }
                    else if (studentOption == 8)
                    {
                        markTheAttendance(userCount, name, usernames, attendance, messBill);
                    }
                    else if (studentOption == 9)
                    {
                        payingMessBill(userCount, name, usernames, attendance, messBill);
                    }
                    else if (studentOption == 10)
                    {
                        // Change password option
                        string oldPassword;
                        string newPassword;
                        y = 21;
                        bool valid = false;

                        // Loop until valid password change is achieved
                        while (!valid)
                        {
                            // Clear the console
                            system("cls");

                            // Display the second header
                            secondHeader();

                            // Display confirmation message
                            cout << endl;
                            cout << "Are you sure you want to change Password? (Press Enter to
    confirm, any key to go back): ";

                            // Get user input (Enter or any other key)
                            char key = _getch();

                            // If Enter is pressed
                            if (key == 13)
                            {
                                // Clear the console
                                system("cls");

                                // Display the second header
                                secondHeader();
```

```
                              // Get old and new passwords
                              gotoxy(36, y);
                              cout << "Enter old Password: ";
                              getline(cin, oldPassword);
                              gotoxy(36, y + 1);
                              cout << "Enter new Password: ";
                              getline(cin, newPassword);

                              // Validate and update password
                              valid = updatePassword(name, oldPassword, newPassword, usernames,
passwords, userCount);

                              // Display error message for invalid password change
                              if (!valid)
                              {
                                 gotoxy(36, y + 3);
                                 cout << "Error: User not found or old password does not match!";
                                 if (_getch() == 13)
                                 {
                                    break;
                                 }
                                 else
                                 {
                                    y = y + 5;
                                    continue;
                                 }
                              }
                              else
                              {
                                 setConsoleColor(14);
                                 // Password updated successfully
                                 gotoxy(36, y + 3);
                                 cout << "Password updated successfully!";
                                 setConsoleColor(7);
                                 _getch(); // Pause to display the success message
                                 break;
                              }
                           }
                           else
                           {
                              // User pressed any key other than Enter, go back
                              break;
                           }
                        }
                     }
                     else if (studentOption == 11)
```

```
                    {
                        // Logout option
                        cout << endl;
                        cout << "\t\t\t\t\t\t\t\t\tAre you sure you want to log out? (Press Enter to
confirm, any key to go back): ";
                        char key = _getch();

                        // If Enter is pressed
                        if (key == 13)
                        {
                            // Clear the console
                            system("cls");

                            // Display logout message
                            cout << "You have been successfully logged out. Goodbye!" << endl;

                            // Display prompt to continue
                            cout << "Press any key to continue...";
                            _getch();
                            break; // Exit the student profile loop and go back to the main loop
                        }
                        else
                        {
                            // User pressed any key other than Enter, go back
                            continue;
                        }
                    }
                }

                valid = true;
            }
            // wardenLogin
            else if (role == "warden")
            {

                loadingFunction();
                system("cls");

                while (true)
                {
                    int wardenoption = wardenProfile();
                    // Warden Functionalities
                    if (wardenoption == 1)
                    {
                        availabilityofstudent(usernames, years, genders, aggregates, alloteds,
userCount);
```

```
                        }
                        else if (wardenoption == 2)
                        {
                            allotementOfStudent(userCount, usernames, alloteds, edhiRooms,
khalidRooms, ayeshaRooms, hostels, rooms, edhiRoomIndex, khalidRoomIndex,
ayeshaIndex);
                        }
                        else if (wardenoption == 3)
                        {
                            checkAvilablityOfRoom(edhiRooms, khalidRooms, ayeshaRooms,
edhiRoomIndex, khalidRoomIndex, ayeshaIndex);
                        }
                        else if (wardenoption == 4)
                        {
                            studentData(usernames, years, genders, aggregates, alloteds, rooms, hostels,
roles, userCount);
                        }
                        else if (wardenoption == 5)
                        {
                            statisticsOfHostelStudents(alloteds, roles, hostels, userCount);
                        }
                        else if (wardenoption == 6)
                        {
                            removingOfStudent(usernames, years, genders, aggregates, alloteds, rooms,
hostels, roles, passwords, iscomplained, isReplyed, attendance, messBill, complains,
userCount, edhiRooms, khalidRooms, ayeshaRooms, edhiRoomIndex, khalidRoomIndex,
ayeshaIndex);
                        }
                        else if (wardenoption == 7)
                        {
                            removingOfRt(usernames, years, genders, aggregates, alloteds, rooms,
hostels, roles, passwords, iscomplained, isReplyed, attendance, messBill, complains,
userCount);
                        }
                        else if (wardenoption == 8)
                        {
                            addingNewRt(usernames, passwords, hostels, genders, years, alloteds, roles,
aggregates, attendance, rooms, messBill, iscomplained, isReplyed, complains, userCount);
                        }
                        else if (wardenoption == 9)
                        {
                            AnnouceSomething(annoucementIndex, annoucemet, role);
                        }
                        else if (wardenoption == 10)
                        {
                            displayingAnnoucement(annoucementIndex, annoucemet);
```

```
                    }
                    else if (wardenoption == 11)
                    {
                        settingRules(rulesIndex, rules, role);
                    }
                    else if (wardenoption == 12)
                    {
                        displayingRules(rulesIndex, rules);
                    }
                    else if (wardenoption == 13)
                    {
                        loadingFunction();
                        system("cls");
                        secondHeader();
                        if (rulesIndex > 0)
                        {
                            int y = 10;
                            int z = 0; // It is used for numbering of complains
                            gotoxy(15, y);
                            cout << "The Rules are following: " << endl;
                            for (int i = 0; i < rulesIndex; i++)
                            {
                                gotoxy(15, y + 1);
                                cout << i + 1 << "-" << rules[i] << endl;
                                y++;
                                z++;
                            }
                            int rulesOption = movementOfArrow(13, 11, 1, z);
                            deletingRule(rulesOption, rulesIndex, rules);
                        }
                        else
                        {
                            setConsoleColor(14);

                            gotoxy(15, 13);
                            cout << "No rules have been entered. Please insert some rules to define the
parameters ";
                            gotoxy(15, 14);
                            cout << "of this unique space. In the absence of guidelines, the realm
remains uncharted, ";
                            gotoxy(15, 15);
                            cout << "devoid of constraints that typically shape the course of creative
expression. " << endl;

                            setConsoleColor(7);
```

```
                gotoxy(15, 36);
                cout << "Press Enter to continue...";
                while (_getch() != 13) // 13 is the ASCII code for Enter key
                {
                    // Do nothing until Enter key is pressed
                }
            }
        }
        else if (wardenoption == 14)
        {
            string oldPassword;
            string newPassword;
            y = 21;

            while (!valid)
            {
                system("cls"); // Clear the console
                secondHeader();

                cout << endl;
                cout << "Are you sure you want to change Password? (Press Enter to
confirm, any key to go back): ";
                char key = _getch();
                if (key == 13)
                {                   // 13 is the ASCII code for Enter
                    system("cls"); // Clear the console
                    secondHeader();

                    gotoxy(36, y);
                    cout << "Enter old Password: ";
                    getline(cin, oldPassword);
                    gotoxy(36, y + 1);
                    cout << "Enter new Password: ";
                    getline(cin, newPassword);
                    valid = updatePassword(name, oldPassword, newPassword, usernames,
passwords, userCount);

                    if (!valid)
                    {
                        gotoxy(36, y + 3);
                        cout << "Error: User not found or old password does not match!";
                        if (_getch() == 13)
                        {
                            break;
                        }
                        else
```

```
                                {
                                    y = y + 5;
                                    continue;
                                }
                            }
                            else
                            {
                                // Password updated successfully
                                gotoxy(36, y + 3);
                                cout << "Password updated successfully!";
                                _getch(); // Pause to display the success message
                                break;
                            }
                        }
                        else
                        {
                            break; // User pressed any key other than Enter
                        }
                    }
                }
                else if (wardenoption == 15)
                {
                    cout << endl;
                    cout << "\t\t\t\t\t\t\t\t\tAre you sure you want to log out? (Press Enter to
confirm, any key to go back): ";
                    char key = _getch();

                    if (key == 13)
                    {               // 13 is the ASCII code for Enter
                        system("cls"); // Clear the console
                        cout << "You have been successfully logged out. Goodbye!" << endl;

                        cout << "Press any key to continue...";
                        _getch();
                        break;
                    }
                    else
                    { // 8 is the ASCII code for Backspace
                        continue;
                    }
                }
            }
            valid = true;
        }
        // rt login
```

```
            else if (role == "rt")
            {
                loadingFunction();
                system("cls");

                while (true)
                {
                    int rtoption = rtProfile();

                    // rt functionalities
                    if (rtoption == 1)
                    {
                        checkingassignedstudents(name, usernames, years, genders, aggregates,
alloteds, rooms, hostels, roles, userCount);
                    }
                    else if (rtoption == 2)
                    {
                        settingRules(rulesIndex, rules, role);
                    }
                    else if (rtoption == 3)
                    {
                        printMessMenu();
                    }
                    else if (rtoption == 4)
                    {
                        displayingAnnoucement(annoucementIndex, annoucemet);
                    }
                    else if (rtoption == 5)
                    {
                        printEventsMenu();
                    }
                    else if (rtoption == 6)
                    {
                        displayingRules(rulesIndex, rules);
                    }
                    else if (rtoption == 7)
                    {
                        replyToComplains(complains, iscomplained, isReplyed, roles, userCount,
hostels, name, usernames);
                    }
                    else if (rtoption == 8)
                    {
                        AnnouceSomething(annoucementIndex, annoucemet, role);
                    }
                    else if (rtoption == 9)
                    {
```

```
            displayingComplains(name, usernames, complains, iscomplained, hostels,
roles, userCount, isReplyed);
          }
          else if (rtoption == 10)
          {
            string oldPassword;
            string newPassword;
            y = 21;

            while (!valid)
            {
              system("cls"); // Clear the console
              secondHeader();
              cout << endl;
              cout << "Are you sure you want to change Password? (Press Enter to
confirm, any key to go back): ";
              char key = _getch();
              if (key == 13)
              {           // 13 is the ASCII code for Enter
                system("cls"); // Clear the console
                secondHeader();

                gotoxy(36, y);
                cout << "Enter old Password: ";
                getline(cin, oldPassword);
                gotoxy(36, y + 1);
                cout << "Enter new Password: ";
                getline(cin, newPassword);
                valid = updatePassword(name, oldPassword, newPassword, usernames,
passwords, userCount);

                if (!valid)
                {
                  gotoxy(36, y + 3);
                  cout << "Error: User not found or old password does not match!";
                  if (_getch() == 13)
                  {
                    break;
                  }
                  else
                  {
                    y = y + 5;
                    continue;
                  }
                }
                else
```

```
                    {
                        // Password updated successfully
                        gotoxy(36, y + 3);
                        cout << "Password updated successfully!";
                        _getch(); // Pause to display the success message
                        break;
                    }
                }
                else
                {
                    break; // User pressed any key other than Enter
                }
            }
        }
        else if (rtoption == 11)
        {
            cout << endl;
            cout << "\t\t\t\t\t\t\t\t\tAre you sure you want to log out? (Press Enter to
confirm, any key to go back): ";
            char key = _getch();

            if (key == 13)
            {               // 13 is the ASCII code for Enter
                system("cls"); // Clear the console
                cout << "You have been successfully logged out. Goodbye!" << endl;

                cout << "Press any key to continue...";
                _getch();
                break;
            }
            else
            { // 8 is the ASCII code for Backspace
                continue;
            }
        }
    }
    valid = true;
}
// If the role is undefined (invalid credentials)
else if (role == "undefined")
{
    // Display error message
    gotoxy(36, y + 3);
    cout << "You Entered wrong Credentials.Press Enter to go back and any key to
continue the process";
```

```
                // If Enter key is pressed, break the loop
                if (_getch() == 13)
                {
                    break;
                }
                else
                {
                    // Increment y and continue the loop
                    y = y + 5;
                    continue;
                }
            }
        }
    }
}

    // Updating the Data in Files
    UpdataDataToFileForParalleArrays(usernames, passwords, hostels, genders, years, alloteds,
roles, aggregates, attendance, rooms, messBill, iscomplained, isReplyed, complains,
userCount);
    storedatainFiles(khalidRooms, khalidRoomIndex, "Khalid Rooms.txt");
    storedatainFiles(edhiRooms, edhiRoomIndex, "Edhi Rooms.txt");
    storedatainFiles(ayeshaRooms, ayeshaIndex, "Ayesha Rooms.txt");
    storestringdatainFiles(rules, rulesIndex, "rules.txt");
    storestringdatainFiles(annoucemet, annoucementIndex, "annoucement.txt");

    loadingFunction();
    system("cls");
}

// Main header
void printMainHeader()
{

    const int frames = 3;

    // For animating the main building of hostel
    for (int frame = 0; frame < frames; ++frame)
    {
        system("cls");
        drawArt();
        Sleep(500);
    }

    loadingFunction();
    system("cls");
    int Mx = 70, My = 15;

    setConsoleColor(11);
    gotoxy(Mx, My);
    cout <<
"@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@" << endl;
    gotoxy(Mx, My + 1);
    cout << "@ /$$$$$$$                    /$$  /$$$$$$   /$$                      @" << endl;
    gotoxy(Mx, My + 2);
    cout << "@| $$__  $$                  |__/ /$$__  $$ | $$                      @" << endl;
    gotoxy(Mx, My + 3);
    cout << "@| $$  \\ $$  /$$$$$$   /$$$$$$$ /$$| $$  \\__//$$$$$$    /$$$$$$  /$$   /$$@" <<
endl;
    gotoxy(Mx, My + 4);
    cout << "@| $$$$$$$/ /$$__  $$ /$$_____/| $$|  $$$$$$|_  $$_/   |____  $$| $$  | $$@" <<
endl;
    gotoxy(Mx, My + 5);
    cout << "@| $$__  $$| $$$$$$$$|  $$$$$$ | $$ \\____  $$ | $$      /$$$$$$$| $$  | $$@" <<
endl;
    gotoxy(Mx, My + 6);
    cout << "@| $$  \\ $$| $$_____/ \\____  $$| $$ /$$  \\ $$ | $$ /$$ /$$__  $$| $$  | $$@" <<
endl;
    gotoxy(Mx, My + 7);
    cout << "@| $$  | $$|  $$$$$$$ /$$$$$$$/| $$|  $$$$$$/ |  $$$$/|  $$$$$$$|  $$$$$$$@" <<
endl;
    gotoxy(Mx, My + 8);
    cout << "@|__/  |__/ \\_______/|_______/ |__/ \\______/   \\___/   \\_______/ \\____  $$@"
<< endl;
    gotoxy(Mx, My + 9);
    cout << "@                                                          /$$  | $$@" << endl;
    gotoxy(Mx, My + 10);
    cout << "@                                                         |  $$$$$$/@" << endl;
    gotoxy(Mx, My + 11);
    cout << "@                                                          \\______/ @" << endl;
    gotoxy(Mx, My + 12);
    cout <<
"@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@" << endl;
    setConsoleColor(7);

    setConsoleColor(13);
    gotoxy(Mx + 3, My + 16);
    cout << "Press Enter to continue...";
    while (_getch() != 13) // 13 is the ASCII code for Enter key
    {
        // Do nothing until Enter key is pressed
    }
    setConsoleColor(7);
    loadingFunction();
}

// Function to display a loading animation
void loadingFunction()
{
    int Mx = 70, My = 16;
    string box = "\xDC"; // ASCII code for a filled box

    // Display "Loading..." message
    gotoxy(Mx + 27, My + 32);
    cout << "Loading...";

    setConsoleColor(10);
    // Position the loading cursor at the start
    gotoxy(Mx, My + 30);
    cout << box; // Set color to green and display a filled box
    setConsoleColor(7);

    // Display loading animation
    for (int i = 0; i < 60; i++)
    {
        Sleep(20);
        Mx++;

        setConsoleColor(10);
        gotoxy(Mx, My + 30);
        cout << box; // Display a filled box at the current position
        setConsoleColor(7);
    }
}

// Function to display a secondary header
void secondHeader()
{
    int M2x = 110, M2y = 3;

    setConsoleColor(13);
    // Display the secondary header
    gotoxy(M2x, M2y);
    cout << ".........." << endl;
    gotoxy(M2x, M2y + 1);
    cout << ":ResiStay:" << endl;
    gotoxy(M2x, M2y + 2);
    cout << ".........." << endl;
    setConsoleColor(7);
}

// Function to validate and check if a username is valid
bool isValidName(const string &username, string namearr[], int &a, int count)
{
    string str = username;
    bool validation = true;

    // Check if the username is empty
    if (username.length()<3)
    {
        validation = false;
        a++;
    }

    if (validation)
    {
        // Check for spaces or uppercase letters in the username
        for (int j = 0; j < str.length(); j++)
        {
            if (isspace(str[j]) || isupper(str[j]))
            {
                validation = false;
                a += 2;
                break;
            }
        }
    }
    if (validation)
    {
        for (int j = 0; j < str.length(); j++)
        {
            if (ispunct(str[j]))
            {
                validation = false;
                a += 3;
                break;
            }
        }
    }
    if (validation)
    {
        for (int j = 0; j < str.length(); j++)
        {
            if (isdigit(str[j]))
            {
                validation = false;
                a += 4;
                break;
            }
        }
    }
    // Check if the username is already taken
    if (validation)
    {
        for (int k = 0; k < count; k++)
        {
            if (username == namearr[k])
            {
                validation = false;
                a += 5;
                break;
            }
        }
    }

    return validation;
}

// Function to validate and check if a password is valid
bool isValidPassword(const string &password, string passwordarr[], int &a, int count)
{
    string str = password;
    bool validation = true;

    // Check if the password length is less than 6 characters
    if (str.length() < 6)
    {
        validation = false;
        a++;
    }
    if (validation)
    {
        if (str.length() > 12)
        {
            validation = false;
            a += 2;
        }
    }
```

```
if (validation)
{
    // Check if the password contains at least one digit
    bool hasDigit = false;
    for (int j = 0; j < str.length(); j++)
    {
        if (isdigit(str[j]))
        {
            hasDigit = true;
            break;
        }
    }

    if (!hasDigit)
    {
        validation = false;
        a += 3;
    }
}
if (validation)
{
    // Check if the password contains at least one special character
    bool hasCharacter = false;
    for (int j = 0; j < str.length(); j++)
    {
        if (ispunct(str[j]))
        {
            hasCharacter = true;
            break;
        }
    }

    if (!hasCharacter)
    {
        validation = false;
        a += 4;
    }
}
if (validation)
{
    for (int j = 0; j < str.length(); j++)
    {
        if (str[j] == ',')
        {
            validation = false;
            a += 5;
            break;
          }
        }
    }
    return validation;
}
// Function to validate and check if a role is valid
bool isValidRole(const string role)
{
    // Check if the role is either "student" or "warden"
    return (role == "student" || role == "warden");
}

// Function to validate and check if a gender is valid
bool isValidGender(const string gender)
{
    // Check if the gender is either "male" or "female"
    return (gender == "male" || gender == "female");
}

// Function to validate and check if a year is valid
bool isValidYear(const string year)
{
    // Check if the year is "1st", "2nd", "3rd", or "4th"
    return (year == "1st" || year == "2nd" || year == "3rd" || year == "4th");
}

// Function to validate and check if an aggregate is valid
bool isValidAggregate(const string aggregate)
{
    // Check if the first character is a digit
    if (!isdigit(aggregate[0]))
    {
        return false;
    }

    // Check if all characters are digits
    for (int i = 0; i < aggregate.length(); i++)
    {
        if (!isdigit(aggregate[i]))
        {
            return false;
        }
    }

    // Convert the string to an integer and check if it's within the range [0, 100]
    int inputValue = stoi(aggregate);
    if (inputValue < 0 || inputValue > 100)
    {
        return false;
    }

    return true;
}

// Function to authenticate user and return their role
string signInConditions(const string username, string namearr[], const string password, string
passwordarr[], int count, string rolesarr[])
{
    for (int i = 0; i < count; i++)
    {
        // Check if the entered username and password match stored credentials
        if (username == namearr[i] && password == passwordarr[i])
        {
            // Return the role of the authenticated user
            return rolesarr[i];
        }
    }
    // Return "undefined" if no match is found
    return "undefined";
}

// Function to handle arrow movement for menu selection
int movementOfArrow(int x, int y, int minOption, int maxOption)
{
    int key;
    setConsoleColor(11);
    gotoxy(x, y);
    cout << "o>";
    setConsoleColor(7);

    do
    {
        key = _getch();

        // Clear previous arrow
        gotoxy(x, y);
        cout << "  ";

        if (key == 72 && minOption > 1)
        {
            // Move up
```

```
          minOption--;
          y = y - 1;
        }
        else if (key == 80 && minOption < maxOption)
        {
          // Move down
          minOption++;
          y = y + 1;
        }

        setConsoleColor(11);
        // Draw new arrow
        gotoxy(x, y);
        cout << "o>";
        setConsoleColor(7);

    } while (key != 13); // 13 is the ASCII code for Enter key
    return minOption;
}

// Function to update the password
bool updatePassword(const string &username, const string &oldPassword, const string
&newPassword, string namearr[], string passarr[], int count)
{
    for (int i = 0; i < count; ++i)
    {
        if (username == namearr[i] && oldPassword == passarr[i])
        {
          // Found the username and old password, update the password
          passarr[i] = newPassword;
          return true; // Password updated successfully
        }
    }
    return false; // Username or old password not found
}

// Function to set the cursor position in the console
void gotoxy(int x, int y)
{
    COORD coordinates;
    coordinates.X = x;
    coordinates.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coordinates);
}

// Function to convert a string to lowercase
```

```
string lowerLetters(string word)
{
   string str = word;
   string lowercase = "";
   for (int i = 0; i < str.length(); i++)
   {
      lowercase += tolower(str[i]);
   }
   return lowercase;
}

// Functions for coloring and discoloring
void setConsoleColor(int color)
{
   HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
   SetConsoleTextAttribute(hConsole, color);
}
void resetConsoleColor()
{
   setConsoleColor(7);
}

// Function for reading data from file in parallel arrays
void readDataFromFileInParallelArrays(string usernames[], string passwords[], string
hostels[], string genders[], string years[], string alloteds[], string role[], string aggregates[], int
attendance[], int rooms[], int messBill[], string iscomplained[], string isReplyed[], string
complains[], int &userCount)
{
   string record;
   fstream data;
   data.open("Users.txt", ios::in);

   string attendances;
   string room;
   string bill;
   while (getline(data, record))
   {
      if (count(record.begin(), record.end(), ',') != 13)
      {
         // Skip the line if it doesn't have exactly 13 commas
         continue;
      }
      // assgin the data according to their respective arrays
      usernames[userCount] = getField(record, 1);
      passwords[userCount] = getField(record, 2);
      hostels[userCount] = getField(record, 3);
```

```
        genders[userCount] = getField(record, 4);
        years[userCount] = getField(record, 5);
        alloteds[userCount] = getField(record, 6);
        role[userCount] = getField(record, 7);
        aggregates[userCount] = getField(record, 8);

        // Assuming attendance, rooms, messBill are integers in the file
        attendances = getField(record, 9);
        if (attendances.length() > 0 && all_of(attendances.begin(), attendances.end(), ::isdigit))
        {
            attendance[userCount] = stoi(attendances);
        }
        else
        {
            attendance[userCount] = 0;
        }

        room = getField(record, 10);
        if (room.length() > 0 && all_of(room.begin(), room.end(), ::isdigit))
        {
            rooms[userCount] = stoi(room);
        }
        else
        {
            rooms[userCount] = 0;
        }

        bill = getField(record, 11);
        if (bill.length() > 0 && all_of(bill.begin(), bill.end(), ::isdigit))
        {
            messBill[userCount] = stoi(bill);
        }
        else
        {
            messBill[userCount] = 0;
        }

        iscomplained[userCount] = getField(record, 12);
        isReplyed[userCount] = getField(record, 13);
        complains[userCount] = getField(record, 14);

        userCount++;
    }

    data.close();
}
```

```
string getField(string record, int field)
{
   int commaCount = 1;
   string item;
   for (int x = 0; x < record.length(); x++)
   {
      if (record[x] == ',')
      {
         commaCount = commaCount + 1;
      }
      else if (commaCount == field)
      {
         item = item + record[x];
      }
   }
   return item;
}

// Function for updating data in file from parallel arrays
void UpdataDataToFileForParalleArrays(const string usernames[], const string passwords[],
const string hostels[], const string genders[], const string years[], const string alloteds[], const
string role[], const string aggregates[], const int attendance[], const int rooms[], const int
messBill[], const string iscomplained[], const string isReplyed[], const string complains[],
const int userCount)
{
   fstream data;
   data.open("Users.txt", ios::out);

   for (int i = 0; i < userCount; ++i)
   {
      data << usernames[i] << "," << passwords[i] << "," << hostels[i] << "," << genders[i] <<
"," << years[i] << ","
           << alloteds[i] << "," << role[i] << "," << aggregates[i] << "," << attendance[i] << ","
<< rooms[i] << ","
           << messBill[i] << "," << iscomplained[i] << "," << isReplyed[i] << "," <<
complains[i] << "\n";
   }

   data.close();
}

// Function for reading data from file in int arrays
void readdatafromFiles(int array[], int &index, const char *filename)
{
   fstream file;
   file.open(filename, ios::in);
```

```
    index = 0;
    string line;

    while (getline(file, line))
    {
        if (line.empty())
        {
            continue;
        }
        if (all_of(line.begin(), line.end(), ::isdigit))
        {
            // Convert the string to an integer and store it in the array
            array[index] = stoi(line);
            index++;
        }
        else
        {
            cerr;
        }
    }
    file.close();
}

// Function for updating data in file from parallel arrays
void storedatainFiles(int array[], int size, const char *filename)
{
    fstream file;
    file.open(filename, ios::out);
    for (int i = 0; i < size; i++)
    {
        file << array[i];
        file << "\n";
    }
    file.close();
}

// Function for reading data from file in string arrays
void readdatafromstringFiles(string array[], int &index, const char *filename)
{
    fstream file;
    file.open(filename, ios::in);
    string line;
    while (getline(file, line))
    {
        if (line.empty())
        {
            continue;
        }
        array[index] = line;
        index++;
    }
    file.close();
}

// Function for updating data from file in parallel arrays
void storestringdatainFiles(string array[], int size, const char *filename)
{
    fstream file;
    file.open(filename, ios::out);
    for (int i = 0; i < size; i++)
    {
        file << array[i];
        file << "\n";
    }
    file.close();
}

// Function to display the login page and get user choice
int loginPage()
{
    int L1x = 60, L1y = 10;
    int choice;

    system("cls");
    secondHeader();

    setConsoleColor(14);
    // Display the login page menu
    gotoxy(L1x, L1y);
    cout <<
"###################################################################################
###############################################" << endl;
    gotoxy(L1x, L1y + 1);
    cout << "#                                   Login Page                                    #"
<< endl;
    gotoxy(L1x, L1y + 2);
    cout <<
"###################################################################################
###############################################" << endl;
    setConsoleColor(7);

    L1x = 76, L1y = 20;
    setConsoleColor(11);

    gotoxy(L1x, L1y - 2);
    cout << 
"#####################################################################################
####" << endl;
    gotoxy(L1x, L1y - 1);
    cout << "#                                                     #" << endl;
    gotoxy(L1x, L1y);
    cout << "#                                                     #" << endl;
    gotoxy(L1x, L1y + 1);
    cout << "#                    1-Sign Up                        #" << endl;
    gotoxy(L1x, L1y + 2);
    cout << "#                    2-Sign In                        #" << endl;
    gotoxy(L1x, L1y + 3);
    cout << "#                    3-Exit                           #" << endl;
    gotoxy(L1x, L1y + 4);
    cout << "#                                                     #" << endl;
    gotoxy(L1x, L1y + 5);
    cout << "#                                                     #" << endl;
    gotoxy(L1x, L1y + 6);
    cout << 
"#####################################################################################
####" << endl;

    setConsoleColor(7);

    // Get user choice using the arrow movement function
    choice = movementOfArrow(L1x + 29, L1y + 1, 1, 3);
    return choice;
}

// Function to display Student's profile options
int studentProfile()
{
    system("cls");
    int choice;
    studentLogo();
    secondHeader();

    // Display Student profile options
    gotoxy(100, 8);
    cout << "Welcome to Student Profile \n";
    int Sx = 100, Sy = 15;
    setConsoleColor(12);
    gotoxy(Sx, Sy);
    cout << "Check Profile";
    gotoxy(Sx, Sy + 1);
    cout << "Submit Complaints or Requests";
    gotoxy(Sx, Sy + 2);
    cout << "Display The Hostel Rules";
    gotoxy(Sx, Sy + 3);
    cout << "View Notices and Announcements";
    gotoxy(Sx, Sy + 4);
    cout << "View Mess Menu";
    gotoxy(Sx, Sy + 5);
    cout << "View Hostel Events Calendar";
    gotoxy(Sx, Sy + 6);
    cout << "Check Reply on Complaints";
    gotoxy(Sx, Sy + 7);
    cout << "Mark the Attendance";
    gotoxy(Sx, Sy + 8);
    cout << "Pay Mess Bill";
    gotoxy(Sx, Sy + 9);
    cout << "Change Password ";
    gotoxy(Sx, Sy + 10);
    cout << "Log Out";

    setConsoleColor(7);
    // Get user choice using arrow movement
    choice = movementOfArrow(Sx - 3, Sy, 1, 11);
    return choice;
}

// Function to display Resident Tutor's profile options
int rtProfile()
{

    system("cls");
    int choice;
    rtLogo();
    secondHeader();
    // Display Resident Tutor profile options
    gotoxy(100, 8);
    cout << "Welcome to Resident Tutor Profile \n";
    int RTx = 100, RTy = 15;
    setConsoleColor(12);
    gotoxy(RTx, RTy);
    cout << "View Assigned Students";
    gotoxy(RTx, RTy + 1);
    cout << "Set Rules";
    gotoxy(RTx, RTy + 2);
    cout << "View Mess Menu";
    gotoxy(RTx, RTy + 3);
    cout << "View Announcements";
    gotoxy(RTx, RTy + 4);
    cout << "View Hostel Events Calendar";
    gotoxy(RTx, RTy + 5);
    cout << "View Hostel Rules";
    gotoxy(RTx, RTy + 6);
    cout << "Resolve the Complaints";
    gotoxy(RTx, RTy + 7);
    cout << "Announce Something";
    gotoxy(RTx, RTy + 8);
    cout << "View Student Complaints";
    gotoxy(RTx, RTy + 9);
    cout << "Change Password";
    gotoxy(RTx, RTy + 10);
    cout << "Log Out";
    setConsoleColor(7);
    // Get user choice using arrow movement
    choice = movementOfArrow(RTx - 3, RTy, 1, 11);
    return choice;
}

// Function to display Warden's profile options
int wardenProfile()
{
    system("cls");
    int choice;
    wardenLogo();
    secondHeader();

    // Warden login
    gotoxy(100, 8);
    cout << "Welcome to warden Profile \n";
    setConsoleColor(12);
    // Display Warden profile options
    int Sx = 100, Sy = 15;
    gotoxy(Sx, Sy);
    cout << "Check Available Students ";
    gotoxy(Sx, Sy + 1);
    cout << "Allot the Student ";
    gotoxy(Sx, Sy + 2);
    cout << "Check Available Rooms ";
    gotoxy(Sx, Sy + 3);
    cout << "Students Data";
    gotoxy(Sx, Sy + 4);
    cout << "Show Graph of Hostels Statistics";
    gotoxy(Sx, Sy + 5);
    cout << "Remove Student";
    gotoxy(Sx, Sy + 6);
    cout << "Remove Resident Tutor ";
    gotoxy(Sx, Sy + 7);
    cout << "Add Resident Tutor";
    gotoxy(Sx, Sy + 8);
    cout << "Announce Something ";
    gotoxy(Sx, Sy + 9);
    cout << "Display Announcements ";
    gotoxy(Sx, Sy + 10);
    cout << "Set Rules  ";
    gotoxy(Sx, Sy + 11);
    cout << "Display Rule ";
    gotoxy(Sx, Sy + 12);
    cout << "Delete Rules ";
    gotoxy(Sx, Sy + 13);
    cout << "Change Password ";
    gotoxy(Sx, Sy + 14);
    cout << "Log Out " << endl;
    setConsoleColor(7);

    // Get user choice using arrow movement
    choice = movementOfArrow(Sx - 3, Sy, 1, 15);
    return choice;
}

// Function to display and animate Main Animation
void drawArt()
{
    setConsoleColor(9);
    cout << "\n\n\n\n\n\n\n";
    cout << "\t\t\t\t\t\t\t                    /\\\n";
    cout << "\t\t\t\t\t\t\t                    /\\\n";
    cout << "\t\t\t\t\t\t\t                    /\\\n";
    cout << "\t\t\t\t\t\t\t                    /\\\n";
    resetConsoleColor();
    Sleep(100);
    setConsoleColor(10);
    cout << "\t\t\t\t\t\t\t                  _`=='_\n";
    cout << "\t\t\t\t\t\t\t                _-~......~-_\n";
    cout << "\t\t\t\t\t\t\t             _--~............~--_\n";
    resetConsoleColor();
    Sleep(100);
    setConsoleColor(12);
    cout << "\t\t\t\t\t\t\t\t                    __--~~...................~~--__\n";
    cout << "\t\t\t\t\t\t\t\t          .___..---~~~..............................~~~---..___,\n";
    cout << "\t\t\t\t\t\t\t\t
`=.______________________________________________,='\n";
    resetConsoleColor();
    Sleep(100);
    setConsoleColor(14);
    cout << "\t\t\t\t\t\t\t\t
@^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^@\n";
    cout << "\t\t\t\t\t\t\t\t                     | | I   I   II   I   I  |  |\n";
    cout << "\t\t\t\t\t\t\t\t                     |  |__I___I___II___I___I__|  |\n";
    cout << "\t\t\t\t\t\t\t\t                     | /___I_  I   II   I  _I___\\ |\n";
    cout << "\t\t\t\t\t\t\t\t                     |'_     ~~~~~~~~~~~~~~~     _`|\n";
    resetConsoleColor();
    Sleep(100);
    setConsoleColor(11);
    cout << "\t\t\t\t\t\t\t\t                  __-~...~~~~~--------------~~~~~...~-__\n";
    cout << "\t\t\t\t\t\t\t\t          ___---~~.....................................~~---___\n";
    cout << "\t\t\t\t\t\t\t\t .___..---~~~.....................................................~~~---..___,\n";
    cout << "\t\t\t\t\t\t\t\t
`=.___________________________________________________________________________,='\n
";
    resetConsoleColor();
    Sleep(100);
    setConsoleColor(13);
    cout << "\t\t\t\t\t\t\t\t
@^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^@\n";
    cout << "\t\t\t\t\t\t\t\t             |   |   | |   | |   ||   | |   | |   |   |\n";
    cout << "\t\t\t\t\t\t\t\t             |   |____| |____|  |   ||   |  |____| |____|   |\n";
    cout << "\t\t\t\t\t\t\t\t             |____________________|____||____|____________________|\n";
    cout << "\t\t\t\t\t\t\t\t           _-|______|______|______|__|------|__|______|______|______|-_  \n";
    resetConsoleColor();
}

// Function to display and animate warden logo
void rtLogo()
{
    system("cls");
    cout << "\n\n\n\n\n\n\n\n\n\n";
    setConsoleColor(10);
    cout << " ____            \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    ______ \n";
    cout << "|  _ \\          \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t   |_   _|\n";
    cout << "| |_) |          \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t     | |  \n";
    cout << "|  _ <           \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t     | |  \n";
    cout << "|_| \\_\\         \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t     |_|  \n";
```

```
    resetConsoleColor();
    Sleep(100);
    setConsoleColor(12);
    cout << " _____          \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    _   _ \n";
    cout << "| ____|         \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t   | | | |\n";
    cout << "|  _|           \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t   | | | |\n";
    cout << "| |___          \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t   | | | |\n";
    cout << "|_____|         \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    \\___/\n";
    resetConsoleColor();
    Sleep(100);
    setConsoleColor(14);
    cout << " ____           \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    _____ \n";
    cout << "/ ___|          \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t   |_   _|\n";
    cout << "\\___ \\          \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t     | |  \n";
    cout << " ___) |         \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t     | |  \n";
    cout << "|____/          \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t     |_|  \n";
    resetConsoleColor();
    Sleep(100);
    setConsoleColor(11);
    cout << " ___            \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    / _ \\\n";
    cout << "|_ _|           \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t   | | | |\n";
    cout << " | |            \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t   | |_| |\n";
    cout << " | |            \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    \\___/\n";
    cout << "|___|           \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t   _____ \n";
    resetConsoleColor();
    Sleep(100);
    setConsoleColor(13);
    cout << " ____           \t\t\t\t\t\t\t\t\t\t\t\t\t \t\t\t\t\t\t\t\t\t  |  _  \\\n";
    cout << "|  _ \\          \t\t\t\t\t\t\t \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t  | |_) | \n ";
    cout << "| | | |         \t\t\t\t\t\t\t\t\t\t\t\t\t \t\t\t\t\t\t\t\t\t  |  _ <  \n ";
    cout << "| |_| |         \t\t\t\t\t\t\t\t\t\t\t\t\t \t\t\t\t\t\t\t\t\t  |_| \\_\\\n";
    cout << "|____ /         \t\t\t\t\t\t\t \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t \n";
    resetConsoleColor();
    Sleep(100);
    cout << " _____          \t\t\t\t\t\t\t\t\t\t\t\t\t\t \n";
    cout << "| ____|         \t\t\t\t\t\t\t\t\t\t\t\t\t\t \n";
    cout << "|  _|           \t\t\t\t\t\t\t\t\t\t\t\t\t\t \n";
    cout << "| |___          \t\t\t\t\t\t\t\t\t\t\t\t\t\t \n";
    cout << "|_____|         \t\t\t\t\t\t\t\t\t\t\t\t\t\t \n";
    Sleep(100);
    setConsoleColor(12);
    cout << " _   _          \t\t\t\t\t\t\t\t\t\t\t\t\t\t \n";
    cout << "| \\ | |         \t\t\t\t\t\t\t\t\t\t\t\t\t\t \n";
    cout << "|  \\| |         \t\t\t\t\t\t\t\t\t\t\t\t\t\t \n";
    cout << "| |\\  |         \t\t\t\t\t\t\t\t\t\t\t\t\t\t \n";
    cout << "|_| \\_|         \t\t\t\t\t\t\t\t\t\t\t\t\t\t \n";
    resetConsoleColor();
    Sleep(100);
    setConsoleColor(11);
    cout << " _____         \t\t\t\t\t\t\t\t\t\t\t\t\t\t \n";
    cout << "|_   _|        \t\t\t\t\t\t\t\t\t\t\t\t\t\t \n";
    cout << "  | |          \t\t\t\t\t\t\t\t\t\t\t\t\t\t \n";
    cout << "  | |          \t\t\t\t\t\t\t\t\t\t\t\t\t\t \n";
    cout << "  |_|          \t\t\t\t\t\t\t\t\t\t\t\t\t\t \n";
    resetConsoleColor();
}

// Function to display and animate student logo
void studentLogo()
{
    system("cls");
    cout << "\n\n\n\n\n\n\n\n\n\n";
    setConsoleColor(10);
    cout << " ____    \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    ____  \n";
    cout << "/ ___/   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t   /  __/  \n";
    cout << "\\___    \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t   \\___ \\  \n";
    cout << " ___)|   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t     __) | \n";
    cout << "|____/   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t   |____/  \n";
    resetConsoleColor();
    Sleep(100);
    setConsoleColor(11);
    cout << "         \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t           \n";
    cout << " _____   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t     _____  \n";
    cout << "|_   _|  \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    |_   _| \n";
    cout << "  | |    \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t      | |   \n";
    cout << "  | |    \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t      | |   \n";
    cout << "  |_|    \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t      |_|   \n";
    resetConsoleColor();
    Sleep(100);
    setConsoleColor(14);
    cout << "         \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t           \n";
    cout << " _   _   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t     _   _ \n";
    cout << "| | | |  \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    | | | |\n";
    cout << "| | | |  \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    | | | |\n";
    cout << "| |_| |  \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    | |_| |\n";
    cout << " \\___/  \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t     \\___/ \n";
    cout << "         \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t           \n";
    resetConsoleColor();
    Sleep(100);
    setConsoleColor(12);
    cout << " ____    \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    ____  \n";
    cout << "|  _ \\  \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t   |  _ \\ \n";
    cout << "| | | |  \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t   | | | |\n";
    cout << "| |_| |  \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t   | |_| |\n";
    cout << "|____/   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t   |____/ \n";
    cout << "         \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t           \n";
    resetConsoleColor();
    Sleep(100);
    setConsoleColor(15);
    cout << " _____   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t     _____  \n";
    cout << "| ____|  \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    | ____| \n";
    cout << "|  _|    \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    |  _|   \n";
    cout << "| |___   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    | |___  \n";
    cout << "|_____|  \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    |_____| \n";
    cout << "         \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t            \n";
    resetConsoleColor();
    Sleep(100);
    setConsoleColor(9);
    cout << " _   _   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t     _   _ \n";
    cout << "| \\ | | \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    | \\ | |\n";
    cout << "|  \\| | \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    |  \\| |\n";
    cout << "| |\\  | \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    | |\\  |\n";
    cout << "|_| \\_| \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    |_| \\_|\n";
    cout << "         \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t            \n";
    resetConsoleColor();
    Sleep(100);
    setConsoleColor(13);
    cout << " _____   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t     _____  \n";
    cout << "|_   _|  \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    |_   _| \n";
    cout << "  | |    \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t      | |   \n";
    cout << "  | |    \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t      | |   \n";
    cout << "  |_|    \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t      |_|   \n";
    resetConsoleColor();
}

// Function to display and animate warden logo
void wardenLogo()
{
    system("cls");
    cout << "\n\n\n\n\n\n\n\n\n\n";
    setConsoleColor(10);
    cout << "__        __   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    __        __\n";
    cout << "\\ \\      / / \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    \\ \\      / /\n";
    cout << " \\ \\ /\\ / / \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t     \\ \\ /\\ / / \n";
    cout << "  \\ V  V /    \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t      \\ V  V /  \n";
    cout << "   \\_/\\_/    \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t       \\_/\\_/   \n";
    resetConsoleColor();
    Sleep(100);
    setConsoleColor(11);
    cout << "         \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t            \n";
    cout << "    _     \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t     _      \n";
    cout << "   / \\    \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t    / \\     \n";
    cout << "  / _ \\   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t   / _ \\    \n";
    cout << " / ___ \\  \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t  / ___ \\   \n";
    cout << "/_/   \\_\\ \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t /_/   \\_\\  \n";
    resetConsoleColor();
    Sleep(100);
    setConsoleColor(13);
    cout << "         \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t            \n";
    cout << " ____     \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t  ____      \n";
    cout << "|  _ \\    \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t |  _ \\     \n";
    cout << "| |_) |   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t | |_) |    \n";
    cout << "|  _ <    \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t |  _ <     \n";
    cout << "|_| \\_\\   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t |_| \\_\\    \n";
    resetConsoleColor();
    Sleep(100);
    setConsoleColor(14);
    cout << "         \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t          \n";
    cout << " ____     \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t  ____      \n";
    cout << "|  _ \\    \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t |  _ \\     \n";
    cout << "| | | |   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t | | | |    \n";
    cout << "| |_| |   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t | |_| |    \n";
    cout << "|____/    \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t |____/     \n";
    resetConsoleColor();
    Sleep(100);
    setConsoleColor(12);
    cout << "         \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t          \n";
    cout << " _____    \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t  _____     \n";
    cout << "| ____|   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t | ____|    \n";
    cout << "|  _|     \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t |  _|      \n";
    cout << "| |___    \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t | |___     \n";
    cout << "|_____|   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t |_____|    \n";
    cout << "         \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t          \n";
    resetConsoleColor();
    Sleep(100);
    setConsoleColor(11);
    cout << " _   _    \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t  _   _     \n";
    cout << "| \\ | |   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t | \\ | |    \n";
    cout << "|  \\| |   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t |  \\| |    \n";
    cout << "| |\\  |   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t | |\\  |    \n";
    cout << "|_| \\_|   \t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t |_| \\_|    \n";
    resetConsoleColor();
}
```

```
// Function to display the Mess Menu
void printMessMenu()
{
   loadingFunction();
   system("cls");
   secondHeader();
   gotoxy(36, 19);
   cout <<
"*****************************************************************" << endl;
   gotoxy(36, 20);
   cout << "|     Day          |    Lunch         |     Dinner        | " << endl;
   gotoxy(36, 21);
   cout <<
"*****************************************************************" << endl;
   gotoxy(36, 22);
   cout << "|    MONDAY        |    Biryani       |     Chicken Kabab  | " << endl;
   gotoxy(36, 23);
   cout << "|    TUESDAY       |    Pulao         |     Ghobi         | " << endl;
   gotoxy(36, 24);
   cout << "|    WEDNESDAY     |    Nihari        |     Allu Anday    | " << endl;
   gotoxy(36, 25);
   cout << "|    THURSDAY      |    Beef          |     Allu gajar    | " << endl;
   gotoxy(36, 26);
   cout << "|    FRIDAY        |    Mutton        |     kher mix      | " << endl;
   gotoxy(36, 27);
   cout << "|    SATURDAY      |    Sindhi Biryani |     Palak        | " << endl;
   gotoxy(36, 28);
   cout << "|    SUNDAY        |    kabab         |     Chicken       |" << endl;
   gotoxy(36, 29);
   cout <<
"*****************************************************************" << endl;
   gotoxy(36, 34);
   cout << "Press Enter to Back...";
   while (_getch() != 13) // 13 is the ASCII code for Enter key
   {
      // Do nothing until Enter key is pressed
   }
   loadingFunction();
   system("cls");
}

// Function to display the Events Menu
void printEventsMenu()
{
   loadingFunction();
   system("cls");
```

```
    secondHeader();
    gotoxy(36, 19);
    cout <<
"*************************************************************************" << endl;
    gotoxy(36, 20);
    cout << "|      Date           |      Timing        |      Events        | " << endl;
    gotoxy(36, 21);
    cout <<
"*************************************************************************" << endl;
    gotoxy(36, 22);
    cout << "|       05            |      Noon          |      DIY WORKSHOP   | " << endl;
    gotoxy(36, 23);
    cout << "|       09            |      NOON          |      DIY WORKSHOP   | " << endl;
    gotoxy(36, 24);
    cout << "|       14            |      Morning       |      SPORTS         | " << endl;
    gotoxy(36, 25);
    cout << "|       18            |      Noon          |      TALENT SHOW    | " << endl;
    gotoxy(36, 26);
    cout << "|       21            |      Night         |      FITNESS CLUB   | " << endl;
    gotoxy(36, 27);
    cout << "|       25            |      Night         |      SPEECH HUB     | " << endl;
    gotoxy(36, 28);
    cout << "|       30            |      Noon          |      BOOK CLUB      | " << endl;
    gotoxy(36, 29);
    cout <<
"*************************************************************************" << endl;
    gotoxy(36, 34);
    cout << "Press Enter to Back...";
    while (_getch() != 13) // 13 is the ASCII code for Enter key
    {
        // Do nothing until Enter key is pressed
    }
    loadingFunction();
    system("cls");
}

// Function to set new rules
void settingRules(int &rulesIndex, string rules[], string role)
{
    loadingFunction(); // Display loading animation
    system("cls");     // Clear screen
    secondHeader();    // Display the second header

    int y = 10;
    gotoxy(20, y);
    cout << "Enter the New Rule: ";
    string newRule;
    getline(cin, newRule);

    if (!newRule.empty()) // Check if the entered rule is not empty
    {
        rules[rulesIndex] = newRule + " (by " + role + ")"; // Add the new rule to the array
        rulesIndex++;                                   // Increment the index of rules array

        gotoxy(25, y + 6);
        cout << "Press Enter to Proceed...";

        while (_getch() != 13) // 13 is the ASCII code for Enter key
        {
            // Do nothing until Enter key is pressed
        }

        loadingFunction(); // Display loading animation
        system("cls");     // Clear screen
        secondHeader();    // Display the second header

        // Display the newly added rule
        gotoxy(20, y + 3);
        cout << "New Rule Added:";
        gotoxy(20, y + 5);
        cout << rules[rulesIndex - 1];
    }
    else
    {
        gotoxy(20, y + 1);
        cout << "Invalid Rule.";
    }

    gotoxy(25, y + 10);
    cout << "Press Enter to Back...";

    while (_getch() != 13) // 13 is the ASCII code for Enter key
    {
        // Do nothing until Enter key is pressed
    }

    loadingFunction(); // Display loading animation
    system("cls");     // Clear screen
    secondHeader();    // Display the second header
}

// Function to display existing rules
```

```
void displayingRules(int rulesIndex, string rules[])
{
    loadingFunction(); // Display loading animation
    system("cls");     // Clear screen
    secondHeader();    // Display the second header

    int y = 10;

    if (rulesIndex > 0)
    {
        setConsoleColor(12);
        gotoxy(15, y);
        cout << "The Rules are following: " << endl;
        setConsoleColor(7);

        for (int i = 0; i < rulesIndex; i++)
        {
            setConsoleColor(14);
            cout << i + 1 << "-" << rules[i] << endl; // Display each rule with its number
            setConsoleColor(7);
        }
    }
    else
    {
        setConsoleColor(14);
        // Display a message when there are no rules
        gotoxy(15, 13);
        cout << "No rules have been entered. Please insert some rules to define the parameters ";
        gotoxy(15, 14);
        cout << "of this unique space. In the absence of guidelines, the realm remains uncharted, 
";
        gotoxy(15, 15);
        cout << "devoid of constraints that typically shape the course of creative expression. " << 
endl;
        setConsoleColor(7);
    }

    gotoxy(25, 50);
    cout << "Press Enter to Back...";

    while (_getch() != 13) // 13 is the ASCII code for Enter key
    {
        // Do nothing until Enter key is pressed
    }

    loadingFunction(); // Display loading animation
    system("cls");     // Clear screen
}

// Function to delete a rule
bool deletingRule(int deletedrule, int &rulesIndex, string rules[])
{
    for (int i = deletedrule - 1; i < rulesIndex; i++)
    {
        rules[i] = rules[i + 1]; // Shift the array to fill the deleted rule's position
    }
    rulesIndex--; // Decrement the index of rules array

    gotoxy(10, 40);
    cout << "You have Successfully deleted the desired rule";

    gotoxy(25, 50);
    cout << "Press Enter to Back...";

    while (_getch() != 13) // 13 is the ASCII code for Enter key
    {
        // Do nothing until Enter key is pressed
    }

    return true; // Return true to indicate successful deletion
}

// Function to allow the user (with a specific role) to announce something
void AnnouceSomething(int &annoucementIndex, string annoucement[], string role)
{
    loadingFunction(); // Display loading animation
    system("cls");
    secondHeader(); // Display the header

    cout << "Announce Something: ";
    string newAnnouncement;
    getline(cin, newAnnouncement);

    // Check if the announcement is not empty
    if (!newAnnouncement.empty())
    {
        // Add the announcement to the array with the role information
        annoucement[annoucementIndex] = newAnnouncement + " (By " + role + ")";
        annoucementIndex++;

        cout << "Announcement added successfully!\n";
    }
    else
    {
        cout << "Invalid Announcement.";
    }

    cout << "Press Enter to Back...";
    while (_getch() != 13) // Wait for Enter key press
    {
        // Do nothing until Enter key is pressed
    }

    loadingFunction(); // Display loading animation
    system("cls");
}

// Function to display announcements
void displayingAnnoucement(int annoucementIndex, string annoucement[])
{
    loadingFunction(); // Display loading animation
    system("cls");
    secondHeader(); // Display the header

    int y = 10;

    setConsoleColor(12);
    gotoxy(15, y);
    cout << "You gave the following Announcements: " << endl;
    setConsoleColor(7);

    // Display each announcement with an index
    for (int i = 0; i < annoucementIndex; i++)
    {
        setConsoleColor(14);
        cout << i + 1 << "-" << annoucement[i] << endl;
        setConsoleColor(7);
    }

    // Prompt to go back
    gotoxy(25, 30);
    cout << "Press Enter to Back...";
    while (_getch() != 13) // Wait for Enter key press
    {
        // Do nothing until Enter key is pressed
    }

    loadingFunction(); // Display loading animation
```

```
  system("cls");
}

// Function to submitting complains to their respective rt
void submittingComplains(string complains[], string isComplained[], string isReplyed[], string
username, string namearray[], int count)
{
  loadingFunction();
  system("cls");
  secondHeader();
  int y = 10;

  for (int i = 0; i < count; i++)
  {
    if (username == namearray[i])
    {
      if (isComplained[i] == "No")
      {
        gotoxy(20, y);
        cout << "Submit Your Complain To RT: ";
        getline(cin, complains[i]);

        // Check if the complain is not empty and does not contain commas
        if (!complains[i].empty() && complains[i].find(',') == string::npos)
        {
          // Add the complain to the array with the role information
          complains[i] = complains[i] + " (by " + username + ")";

          cout << "Complain submitted successfully!\n";
        }
        else
        {
          cout << "Invalid Complain. Complain cannot contain commas.";
        }

        gotoxy(25, y + 6);
        cout << "Press Enter to Back...";

        while (_getch() != 13) // 13 is the ASCII code for Enter key
        {
          // Do nothing until Enter key is pressed
        }

        loadingFunction();
        system("cls");
        secondHeader();
```

```
            }
            else
            {
                setConsoleColor(14);
                gotoxy(20, y);
                cout << "You have already submitted your complain to RT. Please wait for your
reply." << endl;
                gotoxy(20, y + 1);
                cout << "You can submit a new complain after getting a reply from RT on the
previous problem. ";
                setConsoleColor(7);
                gotoxy(25, y + 10);
                cout << "Press Enter to Continue...";

                while (_getch() != 13) // 13 is the ASCII code for Enter key
                {
                    // Do nothing until Enter key is pressed
                }

                loadingFunction();
                system("cls");
                secondHeader();
                break;
            }
        }
    }
}

// Function to check the review on complains
void checkingReplyOnComplain(string isComplained[], string isReplyed[], string username,
string namearray[], int count)
{
    loadingFunction();
    system("cls");
    secondHeader();
    for (int i = 0; i < count; i++)
    {
        if (username == namearray[i])
        {
            if (isComplained[i] == "Yes")
            {
                if (isReplyed[i] == "Yes")
                {
                    setConsoleColor(14);
                    gotoxy(20, 20);
                    cout << "I'm pleased to announce that the problem has been successfully resolved.
" << endl;
                    gotoxy(20, 21);
                    cout << "After thorough investigation and collaborative efforts, we have identified
a " << endl;
                    gotoxy(20, 22);
                    cout << "comprehensive solution that addresses the root cause. Our team's
dedication and " << endl;
                    gotoxy(20, 23);
                    cout << "expertise played a crucial role in implementing the necessary measures
to overcome " << endl;
                    gotoxy(20, 24);
                    cout << "the challenge. With the resolution in place, we can now confidently
affirm that the " << endl;
                    gotoxy(20, 25);
                    cout << "issue has been effectively tackled. I want to extend my gratitude to the
entire team " << endl;
                    gotoxy(20, 26);
                    cout << "for their hard work and commitment throughout this process. This
successful resolution " << endl;
                    gotoxy(20, 27);
                    cout << "not only reflects our collective problem-solving skills but also
strengthens our " << endl;
                    gotoxy(20, 28);
                    cout << "ability to handle challenges proactively in the future. We can now shift
our focus to " << endl;
                    gotoxy(20, 29);
                    cout << "other priorities with the assurance that this particular issue has been
satisfactorily addressed." << endl;
                    setConsoleColor(7);
                    isComplained[i] = "No";
                }
                else
                {
                    setConsoleColor(14);
                    gotoxy(20, 20);
                    cout << "Your patience is highly appreciated as we diligently work to address and
resolve the issue at hand. " << endl;
                    gotoxy(20, 21);
                    cout << "Rest assured that a solution is currently in progress and on its way to
completion. Our team is actively " << endl;
                    gotoxy(20, 22);
                    cout << "engaged in the resolution process, applying their expertise to ensure that
we provide a comprehensive and " << endl;
                    gotoxy(20, 23);
                    cout << "effective fix. We understand the importance of a swift resolution, and we are committed to delivering a " << endl;
                    gotoxy(20, 24);
                    cout << "solution that meets your expectations. Thank you for your understanding and continued patience as we work " << endl;
                    gotoxy(20, 25);
                    cout << "towards resolving your concern." << endl;
                    setConsoleColor(7);
                }
            }
            else
            {
                setConsoleColor(14);
                gotoxy(20, 20);
                cout << "I want to ensure that your concerns are addressed appropriately. However, it seems that I haven't received " << endl;
                gotoxy(20, 21);
                cout << "the details of the specific problem or inquiry you have in mind. Your feedback is crucial, and I'm here to " << endl;
                gotoxy(20, 22);
                cout << "assist you. Please take a moment to submit your problem or question, and I'll do my utmost to provide the " << endl;
                gotoxy(20, 23);
                cout << "assistance and guidance you need. Your engagement is important, and I appreciate the opportunity to help." << endl;
                setConsoleColor(7);
            }
        }
      }
      gotoxy(25, 30);
      cout << "Press Enter to Back...";
      while (_getch() != 13) // 13 is the ASCII code for Enter key
      {
          // Do nothing until Enter key is pressed
      }
      loadingFunction();
      system("cls");
}

// Function to displaying complains
void displayingComplains(string name, string username[], string complains[], string isComplained[], string hostel[], string role[], int count, string isReplyed[])
{
      loadingFunction();
      system("cls");
      secondHeader();
```

```
    int a = 1;
    int y = 10;
    gotoxy(15, y);
    cout << "The Complains are following: " << endl;

    // Use the findStudentByName function
    int studentIndex = findStudentByName(name, username, count);
    if (studentIndex != -1)
    {
        for (int j = 0; j < count; j++)
        {
            if (role[j] == "student" && hostel[j] == hostel[studentIndex] && isComplained[j] ==
"Yes" && isReplyed[j] == "No")
            {
                gotoxy(15, y + 1);
                cout << a << "-" << complains[j] << endl;
                y++;
                a++;
            }
        }
    }

    gotoxy(25, 35);
    cout << "Press Enter to Back...";
    while (_getch() != 13) // 13 is the ASCII code for Enter key
    {
        // Do nothing until Enter key is pressed
    }
    loadingFunction();
    system("cls");
}

// Function to giving responce to complains by students
void replyToComplains(string complains[], string isComplained[], string isReplyed[], string
role[], int count, string hostel[], string name, string namearray[])
{
    loadingFunction();
    system("cls");
    secondHeader();

    int y = 10;
    int z = 0; // It is used for numbering of complains
    string replyedStudentName;

    gotoxy(15, y + 1);
    cout << "Enter the name of the student you want to reply to: ";
```

```
getline(cin, replyedStudentName);

// Find the index of the warden in the user array
int rtIndex = -1;
for (int i = 0; i < count; i++)
{
    if (name == namearray[i])
    {
        rtIndex = i;
        break;
    }
}

bool found = false;
// Check if the warden is found
if (rtIndex != -1)
{

    for (int j = 0; j < count; j++)
    {
        if (replyedStudentName == namearray[j])
        {
            found = true;
            if (hostel[j] == hostel[rtIndex])
            {
                if (isComplained[j] == "Yes" && isReplyed[j] == "No" && role[j] == "student")
                {

                    // Update the complain status
                    isReplyed[j] = "Yes";
                    complains[j] = "Default";

                    // Display success message
                    gotoxy(15, y + 6);
                    cout << "Reply successfully sent.";
                }
                else
                {
                    gotoxy(15, y + 2);
                    cout << "Sorry, this student is not complained yet.";
                }
            }
            else
            {
                // Display appropriate messages if conditions are not met
                gotoxy(15, y + 2);
                cout << "Sorry, this student is not registered in your hostel.";
            }
        }
    }
}
if (!found)
{

    // Display appropriate messages if the warden is not found
    gotoxy(15, y + 2);
    cout << "Sorry, this student is not registered.";
}

gotoxy(25, 50);
cout << "Press Enter to Back...";
while (_getch() != 13) // 13 is the ASCII code for Enter key
{
    // Do nothing until Enter key is pressed
}
}

// Function to display the availability of unalloted students
void availabilityofstudent(string username[], string year[], string gender[], string aggregate[],
string alloted[], int count)
{
    int a = 1;
    loadingFunction(); // Display loading animation
    system("cls");
    secondHeader(); // Display the header

    setConsoleColor(12);
    // Display the header for the table
    gotoxy(10, 12);
    cout << "\t"
         << "Student"
         << "\t\t"
         << " Year"
         << "\t\t"
         << "Gender"
         << "\t\t"
         << "Aggregate";
    setConsoleColor(7);

    int x = 10, y = 15;
    for (int i = 0; i < count; i++)
    {
        // Check if the student is not allotted
        if (alloted[i] == "No")
        {
            setConsoleColor(14);
            // Display student information in a tabular format
            gotoxy(x, y + 1);
            cout << a << "-"
                 << "\t" << username[i] << "\t\t" << year[i] << "\t\t" << gender[i] << "\t\t" <<
aggregate[i];
            y++;
            a++;
            setConsoleColor(7);
        }
    }

    // Display a prompt to go back
    gotoxy(x + 3, y + 6);
    cout << "Press Enter to Back...";
    while (_getch() != 13) // Wait for Enter key press
    {
        // Do nothing until Enter key is pressed
    }
}

// Function to check the availability of rooms in each hostel
void checkAvilablityOfRoom(int edhiRoom[], int khalidRoom[], int ayesharoom[], int
edhiIndex, int khalidIndex, int ayeshaIndex)
{
    loadingFunction(); // Display loading animation
    system("cls");
    secondHeader(); // Display the header
    int y = 15;

    // Display available rooms for each hostel
    gotoxy(20, y);
    cout << "Edhi Rooms:";
    gotoxy(40, y);
    cout << "Khalid Rooms:";
    gotoxy(60, y);
    cout << "Ayesha Rooms:";

    for (int i = 0; i < edhiIndex; i++)
    {
        gotoxy(20, y + 1);
        cout << edhiRoom[i];
        y++;
    }
    y = 15;
    for (int i = 0; i < khalidIndex; i++)
    {
        gotoxy(40, y + 1);
        cout << khalidRoom[i];
        y++;
    }
    y = 15;
    for (int i = 0; i < ayeshaIndex; i++)
    {
        gotoxy(60, y + 1);
        cout << ayesharoom[i];
        y++;
    }

    // Display prompt to go back
    gotoxy(20, y + 6);
    cout << "Press Enter to Back...";
    while (_getch() != 13) // Wait for Enter key press
    {
        // Do nothing until Enter key is pressed
    }
}

// Function to allot a room to a student
void allotementOfStudent(int count, string namearr[], string isAlloted[], int edhiRoom[], int
khalidRoom[], int ayesharoom[], string hostel[], int room[], int &edhiIndex, int &khalidIndex,
int &ayeshaIndex)
{
    // Ignore the newline character
    system("cls");
    secondHeader(); // Display the header

    string name;
    int y = 15;
    bool valid = false;
    bool studentFound = false;

    while (!valid)
    {
        system("cls");
        secondHeader(); // Display the header
        gotoxy(20, 20);
        cout << "Enter the student name to allot a room: ";
        getline(cin, name);
```

```
for (int i = 0; i < count; i++)
{
    if (name == namearr[i])
    {
        studentFound = true;

        if (isAlloted[i] == "No")
        {
            system("cls");
            secondHeader(); // Display the header
            gotoxy(20, 10);
            cout << "Available rooms are as follows:";

            int allottedOption = -1;

            // Display available rooms based on the hostel
            if (hostel[i] == "edhi")
            {
                for (int j = 0; j < edhiIndex; j++)
                {
                    gotoxy(30, y);
                    cout << edhiRoom[j];
                    y++;
                }
                allottedOption = movementOfArrow(28, 15, 1, edhiIndex);
                room[i] = edhiRoom[allottedOption - 1];
                // Update indices and shift the array
                for (int k = allottedOption - 1; k < edhiIndex; k++)
                {
                    edhiRoom[k] = edhiRoom[k + 1];
                }
                edhiIndex--;
            }
            else if (hostel[i] == "khalid")
            {
                for (int j = 0; j < khalidIndex; j++)
                {
                    gotoxy(30, y);
                    cout << khalidRoom[j];
                    y++;
                }
                allottedOption = movementOfArrow(28, 15, 1, khalidIndex);
                room[i] = khalidRoom[allottedOption - 1];
                // Update indices and shift the array
                for (int k = allottedOption - 1; k < khalidIndex; k++)
```

```
                    {
                        khalidRoom[k] = khalidRoom[k + 1];
                    }
                    khalidIndex--;
                }
                else if (hostel[i] == "ayesha")
                {
                    for (int j = 0; j < ayeshaIndex; j++)
                    {
                        gotoxy(30, y);
                        cout << ayesharoom[j];
                        y++;
                    }

                    allottedOption = movementOfArrow(28, 15, 1, ayeshaIndex);
                    room[i] = ayesharoom[allottedOption - 1];
                    // Update indices and shift the array
                    for (int k = allottedOption - 1; k < ayeshaIndex; k++)
                    {
                        ayesharoom[k] = ayesharoom[k + 1];
                    }
                    ayeshaIndex--;
                }

                isAlloted[i] = "Yes"; // Mark the student as allotted
                // Display allotment details
                system("cls");
                gotoxy(20, 21);
                cout << "This student is allotted in " << hostel[i] << " Hostel in Room No " <<
room[i] << endl;
                gotoxy(20, 25);
                cout << "Press Enter to go back...";
                while (_getch() != 13) // Wait for Enter key press
                {
                    // Do nothing until Enter key is pressed
                }
                valid = true;
            }
            else
            {
                // Student is already allotted
                gotoxy(20, 21);
                cout << "This student is already allotted." << endl;
                gotoxy(20, 25);
                cout << "Press Enter to go back and any key to continue the process. ";
                if (_getch() == 13) // Wait for Enter key press
                {
                    valid = true;
                }
                else
                {
                    break;
                }
            }
        }
    }

    if (!studentFound)
    {
        // Student not found in the list
        gotoxy(20, 21);
        cout << "This student was not registered." << endl;
        gotoxy(20, 25);
        cout << "Press Enter to go back and any key to continue the process. ";
        if (_getch() == 13) // Wait for Enter key press
        {
            valid = true;
        }
    }
  }
}

// Function to display information of all allotted students
void studentData(string username[], string year[], string gender[], string aggregate[], string
alloted[], int room[], string hostel[], string role[], int count)
{
    int a = 1;
    loadingFunction(); // Display loading animation
    system("cls");
    secondHeader(); // Display the header

    setConsoleColor(12);
    // Display the header for the table
    gotoxy(10, 12);
    cout << "\t"
         << "Student"
         << "\t\t"
         << "Room"
         << "\t\t"
         << "Hostel"
         << "\t\t"
         << "Year"
        << "\t\t"
        << "Gender"
        << "\t\t"
        << "Aggregate";
    setConsoleColor(7);

    int x = 10, y = 15;
    for (int i = 0; i < count; i++)
    {
        // Check if the student is allotted and has a role of "student"
        if (alloted[i] == "Yes" && role[i] == "student")
        {
            setConsoleColor(14);
            // Display student information in a tabular format
            gotoxy(x, y + 1);
            cout << a << "-"
                 << "\t" << username[i] << "\t\t" << room[i] << "\t\t" << hostel[i] << "\t\t" << year[i]
<< "\t\t" << gender[i] << "\t\t" << aggregate[i];
            y++;
            a++;
            setConsoleColor(7);
        }
    }

    // Display prompt to go back
    gotoxy(x + 3, y + 6);
    cout << "Press Enter to Back...";
    while (_getch() != 13) // Wait for Enter key press
    {
        // Do nothing until Enter key is pressed
    }
}

// Function to mark the attendance for a student
void markTheAttendance(int count, string name, string namearray[], int attendance[], int
messBill[])
{
    loadingFunction();
    system("cls");
    secondHeader();
    for (int i = 0; i < count; i++)
    {
        if (name == namearray[i])
        {
            if (messBill[i] < 1000)
            {
```

```
            cout << "Do you want to mark the attendance? If yes, press Enter. Press any other
key to go back.";
            if (_getch() == 13)
            {
               attendance[i]++;
               messBill[i] = messBill[i] + 200;

               system("cls");
               secondHeader();

               setConsoleColor(12);
               gotoxy(25, 35);
               cout << "Your attendance has been marked. Press Enter to go back...";
               while (_getch() != 13) // 13 is the ASCII code for Enter key
               {
                  // Do nothing until Enter key is pressed
               }
               setConsoleColor(7);

               break;
            }
         }
         else
         {
            setConsoleColor(14);
            gotoxy(10, 12);
            cout << "We have noticed that your outstanding balance for your student account is
quite high." << endl;
            gotoxy(10, 13);
            cout << "Please be aware that you will not be able to mark your attendance until you
have made a payment towards your outstanding balance." << endl;
            gotoxy(10, 14);
            cout << "We urge you to make a payment as soon as possible to avoid any further
issues." << endl;
            setConsoleColor(7);

            gotoxy(25, 35);
            cout << "Press Enter to go back...";
            while (_getch() != 13) // 13 is the ASCII code for Enter key
            {
               // Do nothing until Enter key is pressed
            }
         }
      }
   }
}
```

```
// Function to pay the mess bill for a student
void payingMessBill(int count, string name, string namearray[], int attendance[], int
messBill[])
{
   loadingFunction();
   system("cls");
   secondHeader();
   for (int i = 0; i < count; i++)
   {
      if (name == namearray[i])
      {
         if (messBill[i] != 0)
         {

            setConsoleColor(12);
            gotoxy(25, 15);
            cout << "Do you want to pay the mess bill? Your bill is " << messBill[i]
                 << " Rs. If yes, press Enter. Press any other key to go back.";
            setConsoleColor(7);

            if (_getch() == 13)
            {
               messBill[i] = 0;
               system("cls");

               setConsoleColor(14);
               gotoxy(25, 35);
               cout << "Your bill has been paid. Press Enter to go back...";
               setConsoleColor(7);

               while (_getch() != 13) // 13 is the ASCII code for Enter key
               {
                  // Do nothing until Enter key is pressed
               }
               break;
            }
         }
         else
         {

            setConsoleColor(14);
            gotoxy(10, 12);
            cout << "I understand that you may be concerned about your outstanding bill." <<
endl;
            gotoxy(10, 13);
            cout << "However, I'm here to assure you that there is no pending balance for your
account." << endl;
            gotoxy(10, 14);
            cout << "We have reviewed your records and confirmed that your payments are up
to date." << endl;
            gotoxy(10, 15);
            cout << "Therefore, you can relax knowing that there are no outstanding bills to
worry about." << endl;
            setConsoleColor(7);

            gotoxy(25, 35);
            cout << "Press Enter to go back...";
            while (_getch() != 13) // 13 is the ASCII code for Enter key
            {
                // Do nothing until Enter key is pressed
            }
        }
    }
  }
}

// Function to remove a student
void removingOfStudent(string username[], string year[], string gender[], string aggregate[],
string alloted[], int room[], string hostel[], string role[], string password[], string
iscomplained[], string isReplyed[], int attendance[], int messBill[], string complains[], int
&count, int edhiRoom[], int khalidRoom[], int ayesharoom[], int &edhiIndex, int
&khalidIndex, int &ayeshaIndex)
{
    loadingFunction(); // Function to display loading animation
    system("cls");
    secondHeader(); // Function to display the header
    string name;
    bool valid = false;

    while (!valid)
    {
        system("cls");
        secondHeader(); // Function to display the header
        gotoxy(20, 20);
        cout << "Enter the student name which you want to remove: ";
        getline(cin, name);
        bool studentFound = false;

        for (int i = 0; i < count; i++)
        {
            if (name == username[i] && role[i] == "student")
        {

            studentFound = true;

            if (alloted[i] == "Yes")
            {
                if (hostel[i] == "edhi")
                {
                    edhiIndex++;
                    for (int j = edhiIndex - 1; j >= room[i]; j--)
                    {
                        edhiRoom[j] = edhiRoom[j - 1];
                    }
                    edhiRoom[room[i]-1] = room[i];
                }

                if (hostel[i] == "khalid")
                {
                    khalidIndex++;
                    for (int j = khalidIndex - 1; j >= room[i]; j--)
                    {
                        khalidRoom[j] = khalidRoom[j - 1];
                    }
                    khalidRoom[room[i] - 1] = room[i];
                }
                if (hostel[i] == "ayesha")
                {
                    ayeshaIndex++;
                    for (int j = ayeshaIndex - 1; j >= room[i]; j--)
                    {
                        ayesharoom[j] = ayesharoom[j - 1];
                    }

                    ayesharoom[room[i] - 1] = room[i];
                }
            }

            for (int j = i; j < count; j++)
            {
                // Shift all data to fill the gap for the removed student
                username[j] = username[j + 1];
                year[j] = year[j + 1];
                gender[j] = gender[j + 1];
                aggregate[j] = aggregate[j + 1];
                alloted[j] = alloted[j + 1];
                room[j] = room[j + 1];
                hostel[j] = hostel[j + 1];
                role[j] = role[j + 1];
                password[j] = password[j + 1];
                iscomplained[j] = iscomplained[j + 1];
                isReplyed[j] = isReplyed[j + 1];
                attendance[j] = attendance[j + 1];
                messBill[j] = messBill[j + 1];
                room[j] = room[j + 1];
                complains[j] = complains[j + 1];
            }
            count--; // Decrement the total count of students
            gotoxy(20, 21);
            cout << "This student has been removed." << endl;
            gotoxy(20, 25);
            cout << "Press Enter to go back...";
            while (_getch() != 13) // Wait for Enter key press
            {
                // Do nothing until Enter key is pressed
            }
            valid = true;
            break; // Exit the loop once a student is found and removed
        }
    }

    if (!studentFound)
    {
        gotoxy(20, 21);
        cout << "This student was not registered." << endl;

        gotoxy(20, 25);
        cout << "Press Enter to go back and any key to continue the process. ";
        if (_getch() == 13) // Wait for Enter key press
        {
            valid = true;
        }
    }
  }
}

// Function to remove a Resident Tutor (RT)
void removingOfRt(string username[], string year[], string gender[], string aggregate[], string
alloted[], int room[], string hostel[], string role[], string password[], string iscomplained[],
string isReplyed[], int attendance[], int messBill[], string complains[], int &count)
{
    loadingFunction(); // Function to display loading animation
    system("cls");
```

```
secondHeader(); // Function to display the header
string name;
bool valid = false;
while (!valid)
{
    system("cls");
    secondHeader(); // Function to display the header
    gotoxy(20, 20);
    cout << "Enter the Rt name which you want to remove: ";
    getline(cin, name);
    bool studentFound = false;

    for (int i = 0; i < count; i++)
    {
        if (name == username[i] && role[i] == "rt")
        {
            studentFound = true;
            for (int j = i; j < count; j++)
            {
                // Shift all data to fill the gap for the removed RT
                username[j] = username[j + 1];
                year[j] = year[j + 1];
                gender[j] = gender[j + 1];
                aggregate[j] = aggregate[j + 1];
                alloted[j] = alloted[j + 1];
                room[j] = room[j + 1];
                hostel[j] = hostel[j + 1];
                role[j] = role[j + 1];
                password[j] = password[j + 1];
                iscomplained[j] = iscomplained[j + 1];
                isReplyed[j] = isReplyed[j + 1];
                attendance[j] = attendance[j + 1];
                messBill[j] = messBill[j + 1];
                room[j] = room[j + 1];
                complains[j] = complains[j + 1];
            }
            count--; // Decrement the total count of RTs
            gotoxy(20, 21);
            cout << "This RT has been removed." << endl;
            gotoxy(20, 25);
            cout << "Press Enter to go back...";
            while (_getch() != 13) // Wait for Enter key press
            {
                // Do nothing until Enter key is pressed
            }
            valid = true;
            }
        }

        if (!studentFound)
        {
            gotoxy(20, 21);
            cout << "This RT was not registered." << endl;

            gotoxy(20, 25);
            cout << "Press Enter to go back and any key to continue the process. ";
            if (_getch() == 13) // Wait for Enter key press
            {
                valid = true;
            }
        }
    }
}

// Function to add a new Resident Tutor (RT)
void addingNewRt(string usernames[], string passwords[], string hostels[], string genders[],
string years[], string alloteds[], string role[], string aggregates[], int attendance[], int rooms[],
int messBill[], string iscomplained[], string isReplyed[], string complains[], int &userCount)
{
    loadingFunction(); // Function to display loading animation
    string name, password, hostel;
    bool valid = false;
    int a = 0;
    bool validation = false;
    while (!validation)
    {
        while (!valid)
        {
            system("cls");
            secondHeader(); // Function to display the header
            a = 0;

            gotoxy(36, 20);
            cout << "Username: ";
            getline(cin, name);
            valid = isValidName(name, usernames, a, userCount);

            if (!valid && a == 1)
            {
                gotoxy(36, 21);
                cout << "Your UserName must not have Spaces and Capital letters";
                a = 0;
            }
            if (!valid && a == 2)
            {
                gotoxy(36, 21);
                cout << "This name is already taken.";
            }
        }

        valid = false;
        int y = 22;
        while (!valid)
        {
            a = 0;
            gotoxy(36, y);
            cout << "Enter your Password: ";
            getline(cin, password);
            valid = isValidPassword(password, passwords, a, userCount);

            if (!valid && a == 1)
            {
                gotoxy(36, y + 1);
                cout << "Your Password must be at least six characters";
                a = 0;
            }
            else if (!valid && a == 2)
            {
                gotoxy(36, y + 1);
                cout << "Your Password must have at least one digit";
            }

            y = y + 2;
        }

        valid = false;
        while (!valid)
        {
            gotoxy(36, y);
            cout << "Enter the hostel (edhi, khalid, ayesha): ";
            getline(cin, hostel);

            // Validate the hostel
            if (!(hostel == "edhi" || hostel == "khalid" || hostel == "ayesha"))
            {
                gotoxy(36, y + 1);
                cout << "Invalid Hostel. Press any key to continue or press enter to go back: ";
                if (_getch() == 13)
```

```
                {
                    validation = true;
                }
            }
            else
            {
                valid = true;
            }
            y = y + 2;
        }
        // Add RT with default values
        usernames[userCount] = name;
        passwords[userCount] = password;
        hostels[userCount] = hostel;
        genders[userCount] = "Default";
        years[userCount] = "Default";
        alloteds[userCount] = "Yes";
        role[userCount] = "rt";
        aggregates[userCount] = "Default";
        attendance[userCount] = 0;
        rooms[userCount] = 0;
        messBill[userCount] = 0;
        iscomplained[userCount] = "No";
        isReplyed[userCount] = "No";
        complains[userCount] = "Default";
        gotoxy(36, y + 1);
        cout << "Resident Tutor added successfully!" << endl;
        userCount++;
        gotoxy(36, y + 2);
        cout << "Press Enter to go back...";
        while (_getch() != 13) // Wait for Enter key press
        {
            // Do nothing until Enter key is pressed
        }
        validation = true;
    }
}

// Function to display a student's profile
void checkingStudentProfile(string name, string username[], string year[], string gender[],
string aggregate[], string alloted[], int room[], string hostel[], string role[], int count)
{
    loadingFunction(); // Function to display loading animation
    system("cls");
    secondHeader(); // Function to display the header
    for (int i = 0; i < count; i++)
    {
        if (name == username[i] && alloted[i] == "Yes")
        {
            gotoxy(20, 20);
            cout << "Name: " << name;
            gotoxy(20, 21);
            cout << "Year : " << year[i];
            gotoxy(20, 22);
            cout << "Gender : " << gender[i];
            gotoxy(20, 23);
            cout << "Aggregate : " << aggregate[i];
            gotoxy(20, 24);
            cout << "Hostel : " << hostel[i];
            gotoxy(20, 25);
            cout << "Room No : " << room[i];
            break;
        }
    }
    gotoxy(25, 35);
    cout << "Press Enter to Back...";
    while (_getch() != 13) // Wait for Enter key press
    {
        // Do nothing until Enter key is pressed
    }
}

// Function to display assigned students for a Resident Tutor (RT)
void checkingassignedstudents(string name, string username[], string year[], string gender[],
string aggregate[], string alloted[], int room[], string hostel[], string role[], int count)
{
    loadingFunction(); // Function to display loading animation
    system("cls");
    secondHeader(); // Function to display the header
    gotoxy(10, 12);
    cout << "\t"
         << "Student"
         << "\t\t"
         << "Room"
         << "\t\t"
         << "Hostel"
         << "\t\t"
         << "Year"
         << "\t\t"
         << "Gender"
         << "\t\t"
         << "Aggregate";
    int x = 10, y = 15;
    int a = 1;
    for (int i = 0; i < count; i++)
    {
        if (name == username[i])
        {
            for (int j = 0; j < count; j++)
            {
                if (role[j] == "student" && hostel[j] == hostel[i] && alloted[j] == "Yes")
                {
                    gotoxy(x, y + 1);
                    cout << a << "-"
                         << "\t" << username[j] << "\t\t" << room[j] << "\t\t" << hostel[j] << "\t\t" <<
year[j] << "\t\t" << gender[j] << "\t\t" << aggregate[j];
                    y++;
                    a++;
                }
            }
        }
    }
    gotoxy(25, 35);
    cout << "Press Enter to Back...";
    while (_getch() != 13) // Wait for Enter key press
    {
        // Do nothing until Enter key is pressed
    }
}

// Function to find a student by name
int findStudentByName(string name, string usernames[], int userCount)
{
    for (int i = 0; i < userCount; i++)
    {
        if (name == usernames[i])
        {
            return i;
        }
    }
    return -1; // Return -1 if not found
}

// Functions for Hostel Statistics Graph
void statisticsOfHostelStudents(string allotted[], string role[], string hostel[], int userCount)
{
    loadingFunction();
    system("cls");
    secondHeader();

    int edhiCount = 0;
    int ayeshaCount = 0;
    int khalidCount = 0;
    int x = 70;
    int y = 23;

    gotoxy(50, 25);
    cout << "Edhi Residents";

    gotoxy(50, 29);
    cout << "Khalid Residents";

    gotoxy(50, 35);
    cout << "Ayesha Residents";

    for (int i = 0; i < 15; i++)
    {
        gotoxy(68, y);
        cout << "|";
        y++;
    }

    calculationOfHostelStatistics(edhiCount, khalidCount, ayeshaCount, allotted, role, hostel,
userCount);

    printHashPattern(edhiCount, 14, x, 25);
    gotoxy(x + 2, 25);
    cout << edhiCount << " students";

    x = 70;
    printHashPattern(khalidCount, 11, x, 29);
    gotoxy(x + 2, 29);
    cout << khalidCount << " students";

    x = 70;
    printHashPattern(ayeshaCount, 13, x, 35);
    gotoxy(x + 2, 35);
    cout << ayeshaCount << " students";

    gotoxy(25, 45);
    cout << "Press Enter to Back...";
    while (_getch() != 13) // Wait for Enter key press
    {
        // Do nothing until Enter key is pressed
    }
}
void calculationOfHostelStatistics(int &edhiCount, int &khalidCount, int &ayeshaCount,
string allotted[], string role[], string hostel[], int userCount)
{
    for (int i = 0; i < userCount; i++)
    {
        if (allotted[i] == "Yes" && role[i] == "student")
        {

            if (hostel[i] == "edhi")
            {
                edhiCount++;
            }
            if (hostel[i] == "ayesha")
            {
                ayeshaCount++;
            }
            if (hostel[i] == "khalid")
            {
                khalidCount++;
            }
        }
    }
}
void printHashPattern(int count, int color, int &x, int y)
{
    setConsoleColor(color);
    for (int i = 0; i < count; i++)
    {
        gotoxy(x, y);
        cout << "##";

        gotoxy(x, y + 1);
        cout << "##";
        x = x + 2;
    }
    Sleep(100);
    setConsoleColor(7);
}
```

1.9 Weakness

ResiStay did not contain chatting system among the students. Moreover, there is another weakness in my app is that my program can’t return to its previous state until I entered the corrected data in sign up menu.

1.10 Future Direction

I want to Implement an intelligent room allotment system that considers various factors such as student preferences, compatibility, and proximity to common facilities. This can enhance the overall satisfaction of residents. I can Foster a sense of community by incorporating features that encourage social interactions among hostel residents. This could include discussion forums, event planning, and group messaging to facilitate communication and collaboration.