Programming With Python

AAPP010-4-2-PWP

PWP Group Assignment

Hang in Date: 3 September 2023

Lecturer: Ms Sathiapriya A/P Ramiah and DR. Masrina Akmal Binti Saller

Group Member			
No.	***Name***	***TP Number***	***Intake***
1	LEW WAI HOW	TP071331	UCDF2208ICT(SE)
2	WONG JIN WEI	TP065346	UCDF2208ICT(SE)
3	TAN PO YEH	TP070780	UCDF2208ICT(SE)

Table of Contents

1.0 Workload Matrix

Name	Area of Responsibility
Lew Wai How	1.0 Workload Matrix 3.1 Import Module and Direct User to Main Page 3.2 Common Function in Entire Program 3.6 Admin Home Page 5.0 Explanation of Additional Feature 6.0 Sample of Input and Output
Tan Po Yeh	3.3 Main Page 3.4 View News Page 3.7 Receptionist Functionalization 4.0 Explanation of Program Source Code 8.0 Reference
Wong Jin Wei	2.0 Introduction and Assumption 3.5 Login Page 3.8 Tutor Functionalization 3.9 Student Functionalization 7.0 Conclusion

2.0 Introduction and Assumption

2.1 Introduction

In this rapid development era, people are enjoying the convenience that brought by the technology, which is helps them to complete their work efficiently and effectively. Most of the organizations has the requirement of developing a management information system to assist them in handling Big Data for making better decisions. Brillian Tuition center spend a lot of cost and time for managing and storing their staffs' and students' data. To address this problem, they develop a tuition centre management information system to cost-effective and provide better experience to user. Therefore, this system content some feature that provide a platform for students and staffs to interact with the tuition centre.

2.2 Assumption

The System is using Menu Driver Interface. Menu Driver Interface provide a user interface for user to interact with the system by selecting the choice of function. This type of user interface allow user who does not have IT relevant knowledge to communicate with the system, can also reduce the users' training period. There are 4 roles involve in this program such as Admin, Receptionist, Tutor, and Student. Each role has own different or similar permissions and functions.

Admin is the manager to coordinate and maintain the operation of entire system. Admin has authorization to register new receptionists' account and tutors' account, view and delete receptionists' account and tutors' account, view monthly income of tuition centre, change account's password, and update own profile. For registering new receptionists' account, admin must key in the name, IC or Passport, address, contact number and email address of the receptionist. For registering new tutors' account, admin must key in the information of tutor same as registering new receptionists' account, additionally key in the assign level and assign subject of the tutor. For deleting receptionists' account and tutors' account, admin allow to view the account information of tutor and receptionists and delete the user's account by entering the user's id. Moreover, admin can enter month and year to view the month of monthly income of tuition centre. After that, the system will show the income amount of each subject, total income amount of each level and total amount of the month.

In addition, receptionist is the intermediary between tuition centre and students. They responsible to manage students' account and service related to student. Receptionist has authorization to register students' account and enrol subject, view student's request for changing subject, update students' subject enrolment, generate students' education fee bill, accept payment, generate receipt, view and delete students' account, add news of tuition centre, delete news of tuition centre, change password and update own profile. Receptionist must key in student's personal information, enrolled level and up to 3 enrolled subjects for registering student's account. Receptionist process student's request for changing subject enrolment either approve or reject. If a request approved, the system would update the student subject enrolment based on the request. Moreover, receptionist can generate student's education fee bill while the first day of the month to enable student to deal with payment. Receptionist enters the student's id and the month and year of the bill to receive student's payment and save the payment record to the system after the student pay for education fee for the month. After that, receptionist can generate payment's receipt by entering student's id, the month and year of the payment. Furthermore, receptionist can add and delete news related to tuition centre to allow all user to know the event or activities at the tuition centre. For deleting students' account, admin allow to view the account information of students and delete the students' account by entering the student's id when the students have completed their study.

Moreover, Tutor has the authorization to add class schedule, update class schedule, delete class schedule, view the student list in the subject, change password and update own profile. For adding class schedule, tutor must enter the class date, class venue, class start time and class end time. Tutor can update class schedule by enter the class no., the type of class information needs to be replaced and the new information when has to replace class or postpone class due to some reasons. Furthermore, tutor also allow to delete class schedule if the class has been cancelled. Tutor can view the list of students in the subject to know how many students in the class to prepare material.

In addition, student has the authorization to view the schedule, send requests, view and delete requests, view education fee bill, change passwords, and update own profile. Student can view the class schedule related to the student's subject enrolment. Students allow to send request to receptionist for changing subject enrolment by provide suitable reason of changing subject. Moreover, students can view the request status to know the request was approved or rejected and delete the request which in still pending. Student also can view the education fee bill which in unpaid status.

3.0 Pseudocode

3.1 Import module and direct user to main page

```
START
    IMPORT datetime FROM datetime module
    IMPORT randint FROM random module
    IMPORT hashlib
    CALL main function
END
```

Figure 1 Import module and direct user to main page

3.2 Common Function in Entire Program

3.2.1 Read and retrieve all information from file function

```
FUNCTION read_all with parameter file_name
    DECLARE read_list as empty list
    OPEN file named file_name for reading as read_file
        LOOP read_line from start row to end row step 1 in read_file file
            Declare read_line_str as temporarily storage
            REPLACE \n to empty string in read_line and store the replacing result to read_line_str
            Declare read_line_list as a list which contain each element of read_line_str where splitting by comma
            ADD read_line_list list to read_list list
            NEXT row
        ENDLOOP
    CLOSE read_file file
    RETURN read_list
ENDFUNCTION
```

Figure 2 Read and retrieve all data from a specific file

3.2.2 Retrieve a specific information from function

```
FUNCTION find_one with parameter file_name, start_reference_column, end_reference_column and reference_item
    CALL read_all function where file_name parameter is file_name
    LOOP read_line from first list to last list step 1 in the return result of read_all function
        DECLARE find_line as temporarily storage

        find_line contain elements of read_line where from index start_reference_column
        until index end_reference_column

        DECLARE find_list_str as temporarily storage
        JOIN each find_line element with comma and store to find_list_str
        IF find_line_str is reference_item then
            RETURN read_line
        ENDIF
        NEXT list
    ENDLOOP
ENDFUNCTION
```

Figure 3 Retrieve a specific information from a specific file

3.2.3 Retrieve some specific information in file function

```
FUNCTION find_more with parameter file_name, start_reference_column, end_reference_column and reference_item
    DECLARE find_list as empty list
    CALL read_all function where the file_name parameter is file_name
    LOOP read_line from first list to last list step 1 of the return result of read_all function
        DECLARE find_line as temporarily storage

        find_line contain elements of read_line where from index start_reference_column
        until index end_reference_column

        DECLARE find_list_str as temporarily storage
        JOIN each find_line element with comma and store to find_list_str
        IF find_line_str is reference_item then
            ADD read_line to find_list
        ENDIF
        NEXT list
    ENDLOOP
    RETURN find_list
ENDFUNCTION
```

Figure 4 Retrieve specific information from a specific file

3.2.4 Modify a specific information to file function

```
FUNCTION edit with parameter file_name, start_reference_column, end_reference_column, reference, edit_column
and edit_item
    DECLARE edit_str as empty string
    Call read_all function where file_name parameter is file_name
    LOOP read_line from first list to last list step 1 in the return result of read_all function
        DECLARE find_line as temporarily storage

        find_line contain elements of read_line where from index start_reference_column
        until index end_reference_column

        DECLARE find_list_str as temporarily storage
        JOIN each find_line element with comma and store to find_list_str
        IF find_line_str is reference then
            replace element of read_line where index is edit_column to edit_item
        ENDIF
        DECLARE read_list_str as temporarily storage
        JOIN each read_line element with comma and store to read_list_str
        ADD read_line_str followed by newline to edit_str
        NEXT list
    ENDLOOP
    OPEN file named file_name for writing as change_file
        WRITE edit_str to change_file
    CLOSE change_file file
ENDFUNCTION
```

Figure 5 Modify a specific data in a specific file

3.2.5 Delete file specific information function

```
FUNCTION delete with parameter file_name and delete_row
    OPEN file named file_name for reading as old_file
        OPEN file named file_name for reading and writing as new_file
            INITIALIZE start_row
            DOWHILE start_row smaller than delete_row
                Read line start_row from old_file file
                start_row = start_row + 1
            ENDDO
            SET current_point TO the current file pointer position of old_file file
            MOVE file pointer of new_file file to current point, starting from first place
            READ next line of line start_row from old_file file
            DECLARE next_row as temporarily storage
            READ next line from old_file and store to next_row
            DOWHILE next_row is not empty
                WRITE next_row to new_file file
                READ next line from old_file file and store to next_row
            ENDDO
            DELETE the remain row of new_file
        CLOSE new_file file
    CLOSE old_file file
ENDFUNCTION
```

Figure 6 Delete a specific information in a specific file

3.2.6 Add a specific information to file function

```
FUNCTION register with parameter file_name and register_list
    DECLARE register_item as temporarily storage
    JOIN each element of register_list with comma, followed by newline and store to register_item
    OPEN file named file_name for appending as register_file
        ADD register_item to register_file file
    CLOSE register_file file
ENDFUNCTION
```

Figure 7 Add a specific information to a specific file

3.2.7 Retrieve Level or Subject function

```
FUNCTION retrieve_level_or_subject with parameter levels_or_subjects and level
    DECLARE level_and_subject as dictionary which keys are levels and values are subjects under that level
    IF levels_or_subjects dictionary is string(levels) then
        DECLARE levels_list as list which contain the keys of level_and_subject dictionary
        RETURN levels_list
    ELSE
        DECLARE subject_in_level as subjects from level_and_subject dictionary which belong to the level parameter
        RETURN subjects_in_level
    ENDIF
ENDFUNCTION
```

Figure 8 Retrieve valid level or subject

3.2.8 Retrieve Subject Fee function

```
FUNCTION retrieve_subjects_fee with parameter subject_name
    DECLARE subjects_fee as dictionary which keys are subjects and values are subject fee under that subject
    DECLARE subject_fee as temporarily storage
    RETRIEVE subject fee from subject_fee dictionary which belong to the subject_name given
    subject_fee contain the subject fee retrieved
    RETURN subject_fee
ENDFUNCTION
```

Figure 9 Retrieve subject fee related to the subject

3.2.9 View Levels and Subjects function

```
FUNCTION view_levels_and_subjects without parameter
    DECLARE levels_list as temporarily storage

    CALL retrieve_level_or_subject function where levels_or_subjects parameter is string(levels)
    and level parameter is None

    levels_list contain the list of level in the return result of retrieve_level_or_subject function
    LOOP level from first element to last element step 1 in levels_list
        DECLARE subjects_list as temporarily storage

        CALL retrieve_level_or_subject where levels_or_subjects parameter is string(subjects)
        and level parameter is level

        subjects_list contain the return result of retrieve_level_or_subject function
        DECLARE subjects as temporarily storage
        JOIN each subject_list element with slash and store to subjects
        DISPLAY level concatenated with subjects, following by newline
        NEXT element
    ENDLOOP
ENDFUNCTION
```

Figure 9 Display all levels and subjects

3.2.10 Return home page function

```
FUNCTION return_home_page with parameter continue_statement
    DOWHILE True
        PROMPT user to return to home page or continue
        READ back_choice
        IF back_choice is string(1) then
            DISPLAY newline Return to home page newline
            RETURN True
        ELSE IF back_choice is string(0) then
            RETURN False
        ELSE
            DISPLAY Invalid command!!!newline Please enter right command
        ENDIF
    ENDDO
ENDFUNCTION
```

Figure 10 Allow user to return to home page

3.2.11 Get and Verify Month function

```
FUNCTION get_and_verify_month without parameter
    DOWHILE True
        PROMPT user to enter the month
        Read month
        TRY
            CONVERT class_date to date format (month)
            RETURN month
        CATCH ValueError
            DISPLAY Invalid month
            CALL return_home_page function where continue_statement parameter is string(re-enter the month)
            IF the return result of return_home_page function is True then
                RETURN None
            ENDIF
        ENDIF
    ENDDO
ENDFUNCTION
```

Figure 11 Prompt user to enter month and identify the validation of user's input

3.2.12 Get and Verify Year function

```
FUNCTION get_and_verify_year without parameter
    DOWHILE True
        PROMPT user to enter the year
        Read year
        TRY
            CONVERT class_date to date format (year)
            RETURN year
        CATCH ValueError
            DISPLAY Invalid Year
            CALL return_home_page function where continue_statement parameter is string(re-enter the year)
            IF the return result of return_home_page function is True then
                RETURN None
            ENDIF
        ENDIF
    ENDDO
ENDFUNCTION
```

Figure 12 Prompt user to enter year and identify the validation of user's input

3.2.13 Get and Verify Class Date function

```
FUNCTION get_and_verify_class_date without parameter
    DOWHILE True
        PROMPT user to enter the class date
        Read class_date
        TRY
            CONVERT class_date to date format (day month year)
            RETURN class_date
        CATCH ValueError
            DISPLAY Invalid date
            CALL return_home_page function where continue_statement parameter is string(re-enter class date)
            IF the return result of return_home_page function is True then
                RETURN None
            ENDIF
    ENDDO
ENDFUNCTION
```

Figure 13 Prompt user to enter date and identify the validation of user's input

3.2.14 Get and Verify Class Venue function

```
FUNCTION get_and_verify_class_venue without parameter
    DOWHILE True
        PROMPT user to enter class venue
        Read class_venue
        If class_venue follow the expected format (block-classroom no.) then
            RETURN class_venue
        ELSE
            DISPLAY Invalid Venue
            CALL return_home_page function where continue_statement parameter is string(re-enter class venue)
            IF the return result of return_home_page function is True then
                RETURN None
            ENDIF
        ENDIF
    ENDDO
ENDFUNCTION
```

Figure 14 Prompt user to enter venue and identify the validation of user's input

3.2.15 Get and Verify Class Time function

```
FUNCTION get_and_verify_class_time with parameter start_or_end
    DOWHILE True
        PROMPT user to enter Class start_or_end Time
        Read class_time
        TRY
            CONVERT class_time to time format (hour.minute AM/PM)
            RETURN class_time
        CATCH ValueError
            DISPLAY Invalid Time

            CALL return_home_page function where continue_statement parameter is
            string(re-enter Class start_or_end Time)

            IF the return result of return_home_page function is True then
                Return None
            ENDIF
    ENDDO
ENDFUNCTION
```

Figure 15 Prompt user to enter time and identify the validation of user's input

3.2.16 Crypt Password function

```
FUNCTION crypt_password with parameter user_password
    CREATE SHA-256 hash object
    UPDATE hash object with user_password encoded as UTF-8
    Compute hash of user_password
    RETURN encrypted user_password
ENDFUNCTION
```

Figure 16 Encrypt user's password

3.2.17 Generate User ID and Password function

```
FUNCTION generate_user_id_and_password with parameter role_name
    IF role_name is string(stud) then
        DECLARE check_file as string(payment.txt)
    ELSE
        DECLARE check_file as string(role_name concatenate with .txt)
    DECLARE all_user_id as empty list
    CALL read_all function where file_name parameter is check_file
    LOOP user_info from first list to last list step 1 in the return result of read_all function
        ADD the first element of user_info to all_user_id
        NEXT list
    ENDLOOP
    DOWHILE True
        GENERATE random code between 1 and 9999
        DECLARE id_code as temporarily storage
        pad code with zeros to make it 4 digits and store the result to id_code
        concatenate role_name and id_code to create user_id
        IF user_id not in all_user_id then
            concatenate user_id with @B and id_code to create user_password
            DECLARE user_password_crypt as temporarily storage
            CALL crypt_password function where user_password parameter is user_password to encrypt user_password
            user_password_crypt contain the encrypted user_password
            RETURN a list that contain user_id and user_password_crypt
        ENDIF
    ENDDO
ENDFUNCTION
```

Figure 17 Generate user's id and password randomly

3.2.18 Get and Verify Role Level for Registration function

```
FUNCTION get_and_verify_role_level_for_registration with parameter prompt_statement
    DOWHILE True
        PROMPT user to enter role level for registration
        READ role_level
        DECLARE level_list as temporarily storage

        CALL retrieve_level_or_subject where levels_or_subjects parameter is string(levels) and level parameter is None
        levels_list contain the return result of retrieve_level_or_subject function

        IF role_level is in levels_list then
            RETURN role_level
        ELSE
            DISPLAY newline Invalid level
            CALL return_home_page function where continue_statement parameter is string(re-enter the prompt_statement)
            IF the return result return_home_page function is True then
                RETURN None
            ENDIF
        ENDIF
    ENDDO
ENDFUNCTION
```

Figure 18 Prompt user to enter level for registration and identify the validation of user's input

3.2.19 Get and Verify Student ID in Payment File function

```
FUNCTION get_and_verify_stud_id_in_payment_file with parameter prompt_statement
    DOWHILE True
        PROMPT user to enter student ID concatenate with prompt_statement
        READ stud_id
        DECLARE stud_info_id as temporarily storage

        CALL find_one function where file_name parameter is string(stud.txt),
        start_reference_column parameter is integer(0), end_reference_column parameter is integer(1)
        and reference_item parameter is stud_id

        stud_info_id contain the return result of find_one function
        IF stud_info_id is not empty then
            RETURN stud_id
        ELSE
            DISPLAY Invalid Student ID
            CALL return_home_page function where continue_statement parameter is string(re-enter the student id)
            IF the return result of return_home_page function is True then
                RETURN None
            ENDIF
        ENDIF
    ENDDO
ENDFUNCTION
```

Figure 19 Prompt user to enter student's ID and identify the validation of user's input with the payment file data

3.2.20 Change User's Password Page

```
FUNCTION change_password with parameter user_file and user_id
    DISPLAY newline Home Page >>> Change Password
    DISPLAY Change Password
    DOWHILE True
        DECLARE current_password as temporarily storage

        CALL find_one function where file_name parameter is user_file, start_reference_column is integer(0),
        end_reference_column is integer(1) and reference_item parameter is user_id

        current_password contain the second element of the return result of find_one function
        PROMPT user to enter original password
        READ ori_password
        DECLARE ori_password_crypt as temporarily storage
        CALL crypt_password function where user_password parameter is ori_password to encrypt the ori_password
        ori_password_crypt contain the return result of crypt_password function
        IF ori_password_crypt is current_password then
            PROMPT user to enter new password
            READ new_password
            PROMPT user to enter confirm password
            READ confirm_password
            IF new_password is confirm password then
                IF the length of new_password greater or equal to 8 and smaller or equal to 16 then
                    DECLARE new_password_crypt as temporarily storage
                    CALL crypt_password function where user_password parameter is new_password
                    new_password_crypt contain the return result of crypt_password function
```

```
                    CALL edit function where file_name parameter is user_file,
                    start_reference_column parameter is integer(0), end_reference_column parameter is integer(1),
                    reference parameter is user_id, edit_column parameter is 1
                    and edit_item parameter is new_password_crypt

                    DISPLAY newline Change password success newline newline Return to home page newline
                    RETURN
                ELSE
                    DISPLAY Length of your password must minimum 8 and maximum 16
                ENDIF
            ELSE
                DISPLAY New password and Confirm password does not match
            ENDIF
        ELSE
            DISPLAY Wrong original password
        ENDIF
        CALL return_home_page function where continue_statement parameter is string(re-enter the password)
        IF the return result of return_home_page function is True then
            RETURN
        ENDIF
    ENDDO
ENDFUNCTION
```

Figure 20 Allow user to change own password

3.2.21 Update User's Profile Page

```
FUNCTION update_profile with parameter user_file and user_id
    DISPLAY newline Home Page >>> Update Profile
    DISPLAY Update Profile
    DECLARE user as temporarily storage

    CALL find_one function where file_name parameter is user_file, start_reference_column parameter is integer(0),
    end_reference_column parameter is integer(1), reference_item parameter is user_id

    user contain the return result of find_one function
    DISPLAY the elements of user concatenate with its category
    DECLARE update_data as list which contain editable user information
    CALL return_home_page function where continue_statement parameter is string(update own profile)
    IF the return result of return_home_page function is True then
        DISPLAY Return to home page
        RETURN
    ENDIF
    LOOP index and data from index 0 = element 1 until last index = last element, step 1 in update_data
        CALCULATE index = index + 1
        DISPLAY index concatenate with its data
        NEXT index and element
    ENDLOOP
    DOWHILE True
        PROMPT user to enter the data number that want to update
        READ value
        TRY
            convert value to integer
            IF value is greater than 0 and smaller or equal to the length of update_data then
                BREAK DO
            ELSE
                DISPLAY Invalid number
                CALL return_home_page function where continue_statement parameter is string(re-enter the number)
```

```
                    RETURN None
                ENDIF
            ENDIF
        CATCH ValueError
            DISPLAY Invalid user data number
            CALL return_home_page function where continue_statement parameter is string(re-enter the number)
            IF the return result of return_home_page function is True then
                RETURN None
            ENDIF
    ENDDO
    IF value is equal to 1 THEN
        PROMPT user to enter new address
        READ update_item
    ELSE IF value is equal to 2 THEN
        PROMPT user to enter new email
        READ update_item
    ELSE
        PROMPT user to enter new contact number
        READ update_item
    CALCULATE value = value + 3

    CALL edit function where file_name parameter is user_file, start_reference_column parameter is integer(0),
    end_reference_column parameter is integer(1), reference parameter is user_id, edit_column parameter is value
    and edit_item parameter is update_item

    DISPLAY Update profile success
    DISPLAY Return to home page
ENDFUNCTION
```

Figure 21 Allow user to update own profile

3.3 Main Page

```
FUNCTION main without parameter
    DOWHILE True
        DISPLAY Welcome to Brilliant Education Center newline
        DISPLAY Enter 1: Login account newline Enter 2: View news
        DOWHILE True
            PROMPT user to enter a number
            READ choice
            IF choice is string(1) then
                DISPLAY newline Please enter your id and password for login
                DECLARE user_login as temporarily storage
                CALL login function
                user_login contain the return result of login function
                IF user_login is empty then
                    DISPLAY You have use all chance and not allow to login
                    EXIT the program
                ELSE IF the first element of user_login is string(admin.txt) then
                    CALL admin_ui where user_id parameter is the second element of user_login
                    BREAK DO
                ELSE IF the first element of user_login is string(recep.txt) then
                    CALL recep_ui where user_id parameter is the second element of user_login
                    BREAK DO
                ELSE IF the first element of user_login is string(tutor.txt) then
                    CALL tutor_ui where user_id parameter is the second element of user_login
                    BREAK DO
                ELSE
                    CALL stud_ui where user_id parameter is the second element of user_login
                    BREAK DO
```

```
            ENDIF
        ELSE IF choice is string(2) then
            DISPLAY newline Main page >>> View news
            CALL view_news function
            IF the return result of view_news function is True then
                DISPLAY No available of news
            DISPLAY Return to main page newline
            BREAK DO
        ELSE
            DISPLAY Invalid command
        ENDIF
    ENDDO
ENDDO
ENDFUNCTION
```

Figure 22 Main Page

3.4 View News Page

```
FUNCTION view_news without parameter
    DISPLAY View News
    OPEN file named news.txt for reading as news_file
        DECLARE news_info as temporarily storage
        READ one line from news_file and store the line to news_info
        IF new_info is empty then
            RETURN False
        ELSE
            Set news_num to 1
            DOWHILE news_info is not empty
                DISPLAY News concatenate with news_num and followed by newline
                DECLARE news as temporarily storage
                REPLACE \\n to \n in news_info
                news contain the result after replacing
                DISPLAY news
                READ one line from news_file and store the line to new_info
                new_num = news_num + 1
            ENDDO
            RETURN True
        ENDIF
    CLOSE news_file file
ENDFUNCTION
```

Figure 23 View News Page

3.5 Login Page

```
FUNCTION login without parameter
    DECLARE roles_file as a list which contain all role's file name such admin.txt, recep.txt, tutor.txt and stud.txt
    INITIALIZE login_num
    DOWHILE login_num smaller than 3
        PROMPT user to enter id
        READ user_id
        LOOP current_file from first element to last element step 1 in roles_file
            DECLARE user_info as temporarily storage

            CALL find_one function where file_name parameter is current_file, start_reference_column is integer(0),
            end_reference_column is integer(1) and reference_item parameter is user_id

            user_info contain the return result of find_one function
            IF user_info is not empty then
                DECLARE password as temporarily storage
                password contain the second element of user_info
                PROMPT user to enter password
                READ user_password
                DECLARE user_password_crypt as temporarily storage
                CALL crypt_password function where user_password parameter is user_password to encrypt user_password
                user_password_crypt contain the return result of crypt_password function
                IF user_password_crypt is password then
                    DISPLAY newline Login success newline Welcome newline
                    RETURN list which contain current_file and user_id
                ELSE
                    DISPLAY Invalid Password
                    IF current_file is string(admin.txt)
                        login_num = login_num - 1
                        DISPLAY newline Invalid password newline Only Admin login failed will reset login chance
                    BREAK LOOP
```

```
            ENDIF
        NEXT element
    ENDLOOP
    ELSE
        DISPLAY Invalid ID
    ENDIF
    login_num = login_num + 1
    CALCULATE login_num = 3 - login_num
    DISPLAY You remain login_num chance !!! newline
ENDDO
ENDFUNCTION
```

Figure 24 Login Page

3.6 Admin Functionalization

3.6.1 Admin Home Page

```
FUNCTION admin_ui where parameter user_id

    DECLARE admin_functions as a list which contain the function service that can be used by admin
    such as Register Receptionist, Register Tutor, View and Delete Receptionist, View and Delete Tutor,
    View monthly income report, Change Password, Update Profile and Log out

    DOWHILE TRUE
        DISPLAY Admin Home Page

        LOOP index and function from first index = first element to last index = last element step 1 of
        admin_functions list

            CALCULATE index = index + 1
            DISPLAY index CONCATENATE with function
            NEXT index and element
        ENDLOOP
        PROMPT user for choice
        READ choice

        IF choice equal to string(1) then
            CALL register_recep function
        ELSE IF choice equal to string(2) then
            CALL register_tutor function
        ELSE IF choice equal to string(3) then
            CALL delete_recep function
        ELSE IF choice equal to string(4) then
            CALL delete_tutor function
```

```
        ELSE IF choice equal to string(5) then
            CALL view_income function
        ELSE IF choice equal to string(6) then
            CALL change_password function where user_file parameter is string(admin.txt) user_id parameter is user_id
        ELSE IF choice equal to string(7) then
            CALL update_profile function where user_file parameter is string(admin.txt) user_id parameter is user_id
        ELSE IF choice equal to string(8) then
            DISPLAY Log out success
            RETURN out
        ELSE
            DISPLAY Invalid command
        ENDIF
    ENDDO
ENDFUNCTION
```

Figure 25 Admin Home Page

3.6.2 Register Receptionist's Account Page

```
FUNCTION register_recep without parameter
    DISPLAY newline Home Page >>> Register Receptionist
    DISPLAY Register Receptionist Home Page

    DECLARE recep_info_list as a list which contain all type of Receptionist information
    such as Receptionist's Name, Receptionist's IC or Passport, Receptionist's Address, Receptionist's Contact Number
    and Receptionist's Email

    DECLARE new_recep_info as empty list
    DECLARE id_and_password as temporarily storage
    id_and_password contain the return result of calling generate_user_id_and_password function
    where role_name parameter is string(recep)
    DECLARE recep_id as temporarily storage
    recep_id contain the first element of id_and_password
    DISPLAY Receptionist's ID generated: CONCATENATE with recep_id
    DECLARE recep_password as temporarily storage
    recep_password contain the second element of id_and_password
    ADD recep_id and recep_password to new_recep_info
```

```
    LOOP recep_data from first element to last element step 1 of recep_info_list list
        DOWHILE True
            PROMPT user to enter recep_data to register Receptionist account
            READ data
            IF data is not empty string THEN
                BREAK DO
            ELSE
                DISPLAY You must enter something !!!
                CALL return_home_page function where continue_statement is string(re-enter recep_data)
                IF the return result of return_home_page function is True then
                    RETURN None
                ENDIF
            ENDIF
        ENDDO
        ADD data TO new_recep_info
        NEXT element
    ENDLOOP
    CALL register function where file_name parameter is string(recep.txt) register_list parameter is new_recep_info
    DISPLAY Register Receptionist Success
    DISPLAY Return to home page
ENDFUNCTION
```

Figure 26 Register Receptionist's Account Page

3.6.3 Register Tutor’s Account Page

```
FUNCTION register_tutor without parameter
    DISPLAY newline Home Page >>> Register Tutor
    DISPLAY Register Tutor

    DECLARE tutor_info_list as a list which contain the all type of tutor information
    such as Tutor's Name, Tutor's IC or Passport, Tutor's Address, Tutor's Contact Number and Tutor's Email

    DISPLAY (Levels and Subjects)
    CALL view_levels_and_subjects function
    DECLARE new_tutor_info as empty list
    DECLARE id_and_password as temporarily storage
    CALL generate_user_id_and_password function where role_name parameter is string(tutor)
    id_and_password contain the return result of generate_user_id_and_password function
    DECLARE tutor_id as temporarily storage
    tutor_id contain the first element of id_and_password
    DISPLAY Tutor's ID generated: CONCATENATE with tutor_id
    DECLARE tutor_password as temporarily storage
    tutor_password contain the second element of id_and_password
    ADD tutor_id and tutor_password to new_tutor_info
    LOOP tutor_data from first element to last element step 1 of tutor_info_list list
        DOWHILE True
            PROMPT user to enter tutor_data to register Tutor account
            READ data
            IF data is not empty string THEN
                BREAK DO
            ELSE
                DISPLAY You must write something !!!
                CALL return_home_page function where continue_statement parameter is string(re-enter tutor_data)
                IF the return result of return_home_page function is True THEN
                    RETURN None
                ENDIF
```

```
        ENDIF
    ENDDO
    ADD data to new_tutor_info
    NEXT element
ENDLOOP
DECLARE tutor_assign_level as temporarily storage

tutor_assign_level contain the return result of calling get_and_verify_role_level_for_registration function
where prompt_statement parameter is string(Tutor's in-charge level)

IF tutor_assign_level is empty THEN
    RETURN None
ELSE
    ADD tutor_assign_level TO new_tutor_info
ENDIF
DOWHILE TRUE
    PROMPT user to enter Tutor's in-charge subject to register the account
    READ tutor_assign_subject
    CONVERT all word in tutor_assign_subject to capitalize
    DECLARE subject_info as temporarily storage

    subject_info contain the return result of calling retrieve_level_or_subject function where
    levels_or_subjects parameter is string(subjects) and level parameter is tutor_assign_level

    IF tutor_assign_subject existed in subject_info THEN
        ADD tutor_assign_subject to new_tutor_info
        BREAK DO
    ELSE
        DISPLAY newline Invalid Subject
```

```
            IF the return result of calling return_home_page function where continue_statement parameter is
            string(re-enter the Tutor's in-charge subject) is True THEN

                RETURN None
            ENDIF
        ENDIF
    ENDDO
    CALL register function where file_name parameter is string(tutor.txt) register_list parameter is new_tutor_info
    DISPLAY Register Tutor Success
    DISPLAY Return to home page
ENDFUNCTION
```

Figure 27 Register Tutor's Account Page

3.6.4 Delete Receptionist’s Account Page

```
FUNCTION delete_recep without parameter
    DISPLAY newline Home Page >>> Delete Receptionist
    DISPLAY Delete Receptionist
    DECLARE recep_list as temporarily storage
    recep_list contain the return result of calling read_all function where file_name parameter is string(recep.txt)
    IF recep_list is empty THEN
        DISPLAY No Receptionist Information
        RETURN None
    ENDIF
    SORT recep_list order by the first element of recep_list
    LOOP recep_info from first list to last list step 1 of recep_list
        DISPLAY Receptionist ID: concatenate with the first element of recep_info
        DISPLAY Receptionist Name: concatenate with the third element of recep_info
        DISPLAY Receptionist IC or Passport: concatenate with the fourth element of recep_info
        DISPLAY Receptionist Address: concatenate with the fifth element of recep_info
        DISPLAY Receptionist Contact Number: concatenate with the sixth element of recep_info
        DISPLAY Receptionist Email: concatenate with the seventh element of recep_info
        DISPLAY ~ which repeated 70 times
        NEXT list
    ENDLOOP

    IF the return result of calling return_home_page function where continue_statement parameter
    is string(delete receptionist's information) is True THEN

        RETURN None
    ENDIF
    DOWHILE TRUE
        PROMPT user to enter Receptionist's ID to delete the account
        READ delete_recep_id
```

```
        DECLARE delete_recep_info as temporarily storage

        delete_recep_info contain the return result of calling find_one function where file_name parameter
        is string(recep.txt), start_reference_column parameter is integer(0), end_reference_colum is integer(1)
        and reference_item parameter is delete_recep_id

        IF delete_recep_info is not empty THEN
            DECLARE recep_delete_row as temporarily storage
            recep_delete_row contain the index value of delete_recep_info where in recep_list
            BREAK DO
        ELSE
            DISPLAY newline Receptionist ID does not exist

            IF the return result of calling return_home_page where continue_statement parameter
            is string(re-enter the receptionist's id) is True THEN

                RETURN None
            ENDIF
        ENDIF
    ENDDO
    CALL delete function where file_name parameter is string(recep.txt), delete_row parameter is recep_delete_row
    DISPLAY Delete Receptionist Success Page
    DISPLAY Return to home page
ENDFUNCTION
```

Figure 28 Delete Receptionist's Account Page

3.6.5 Delete Tutor’s Account Page

```
FUNCTION delete_tutor without parameter
    DISPLAY newline Home Page >>> Delete Tutor
    DISPLAY Delete Tutor Home Page
    DECLARE tutor_list as temporarily storage
    tutor_list contain the return result of calling read_all function where file_name parameter is string(tutor.txt)
    IF tutor_list is empty THEN
        DISPLAY No Tutor Information
        RETURN None
    ENDIF
    SORT tutor_list order by the first element of tutor_list
    LOOP tutor_info from first list to last list step 1 of tutor_list list
        DISPLAY Tutor's ID: concatenate with the first element of tutor_info
        DISPLAY Tutor's Name: concatenate with the third element of tutor_info
        DISPLAY Tutor's IC or Passport: concatenate with the fourth element of tutor_info
        DISPLAY Tutor's Address: concatenate with the fifth element of tutor_info
        DISPLAY Tutor's Contact Number: concatenate with the sixth element of tutor_info
        DISPLAY Tutor's Email: concatenate with the seventh element of tutor_info
        DISPLAY Tutor's Assign Level: concatenate with the eighth element of tutor_info
        DISPLAY Tutor's Assign Subject: concatenate with the ninth element of tutor_info
        DISPLAY ~ which repeated 70 times
        NEXT list
    ENDLOOP

    IF the return result of calling return_home_page function where continue_statement parameter
    is string(delete tutor's information) is TRUE THEN

        RETURN None
    ENDIF
    DOWHILE TRUE
```

```
        PROMPT user to enter Tutor ID to delete the account
        READ delete_tutor_id
        DECLARE delete_tutor_info as temporarily storage

        delete_tutor_info contain the return result of calling find_one function where
        file_name parameter is string(tutor.txt), start_reference_column parameter is integer(0),
        end_reference_column parameter is integer(1), reference_item parameter is delete_tutor_id

        IF delete_tutor_info is not empty THEN
            DECLARE tutor_delete_row as temporarily file
            tutor_delete_row contain the index value of delete_tutor_info where in tutor_list
            BREAK DO
        ELSE
            DISPLAY newline Tutor's ID does not exist

            IF the return result of calling return_home_page function where
            continue_statement parameter is string(re-enter the tutor's id) is TRUE THEN

                RETURN None
            ENDIF
        ENDIF
    ENDDO

    CALL delete function where file_name parameter is string(tutor.txt), delete_row parameter is tutor_delete_row
    DISPLAY Delete Tutor Success
    DISPLAY Return to home page
ENDFUNCTION
```

Figure 29 Delete Tutor's Account Page

3.6.6 View Monthly Income of Tuition Centre Page

```
FUNCTION view_income without parameter
    DISPLAY newline Home Page >>> View Student Monthly Income"
    DISPLAY View Monthly Income Home Page
    DECLARE month as temporarily storage
    month contain the return result of calling get_and_verify_month function
    DECLARE year as temporarily storage
    year contain the return result of calling get_and_verify_year function
    IF month is empty or year is empty THEN
        RETURN None
    ENDIF
    DECLARE date as temporarily storage
    date contain month CONCATENATE with year
    DECLARE payment_list_date as temporarily storage

    payment_list_date contain the return result of calling find_more function where
    file_name parameter is string(payment.txt), start_reference_column parameter is integer(1),
    end_reference_column parameter is integer(2) and reference_item parameter is date

    DECLARE payment_list_status as temporarily file

    payment_list_status contain the return result of calling find_more function where
    file_name parameter is string(payment.txt), start_reference_column parameter is integer(10),
    end_reference_column parameter is integer(11) and reference_item parameter is string(PAID)

    DECLARE payment_list_date_level_status as empty list
    IF payment_list_status is empty OR payment_list_date is empty THEN
        DISPLAY No available income in this month
        DISPLAY Return to home page
        RETURN None
```

```
ELSE
    LOOP payment_info_status from first list to last list step 1 of payment_list_status list
        IF payment_info_status existed in payment_list_date THEN
            ADD payment_info_status TO payment_list_date_level_status list
        ENDIF
        NEXT list
    ENDLOOP
ENDIF
DISPLAY newline date concatenate with date and newline
INITIALIZE total_fee

LOOP level from first element to last element step 1 of the return result of calling
retrieve_level_or_subject function where levels_or_subjects parameter is string(levels), level parameter is None

    INITIALIZE total_level_fee\
    DISPLAY Level: CONCATENATE with level
    DECLARE subject_info as temporarily storage

    subject_info contain the return result of calling retrieve_level_or_subject function
    where levels_or_subjects parameter is string(subjects), level parameter is level

    LOOP subject_num and subject from index 0 = element 1 to last index = last element step 1 of subject_info
        INITIALIZE total_subject_fee
        LOOP subject_enrolled_num from 1 to 3 step 1
            DECLARE payment_list_subject_enrolled as temporarily storage
            CALCULATE current_subject_enrolled_num = 3 + subject_enrolled_num
            CALCULATE next_subject_enrolled_num = 4 + subject_enrolled_num

            payment_list_subject_enrolled contain the return result of calling find_more function where
```

```
        file_name parameter is string(payment.txt),
        start_reference column parameter is current_subject_enrolled_num,
        end_reference_column parameter is next_subject_enrolled_num
        and reference_item parameter is subject

        IF payment_list_subject_enrolled is not empty THEN
            DECLARE payment_list_level as temporarily storage

            payment_list_level contain the return result of calling
            find_more function where file_name parameter is string(payment.txt),
            start_reference column parameter is integer(2),end_reference_column parameter is integer(3),
            reference_item parameter is level

            LOOP payment_info_subject_enrolled from first list to last list
            step 1 of payment_list_subject_enrolled

                IF payment_info_subject_enrolled existed in payment_list_date_level_status AND
                payment_info_subject_enrolled existed in payment_list_level THEN
                    CALCULATE subject_fee_num = subject_enrolled_num + 6
                    DECLARE subject_fee as temporarily storage

                    subject_fee contain the element of the payment_info_subject_enrolled
                    where index value is subject_fee_num

                    CONVERT subject_fee TO integer
                    CALCULATE total_subject_fee = total_subject_fee + subject_fee
                ENDIF
                NEXT list
            ENDLOOP
        ENDIF
```

```
                NEXT number
            ENDLOOP
            CALCULATE subject_num = subject_num + 1
            DISPLAY tab Subject concatenate with subject_num and subject
            DISPLAY tab Total income: RM concatenate with total_subject_fee newline
            CALCULATE total_level_fee = total_level_fee + total_subject_fee
            NEXT index and element
        ENDLOOP
        CALCULATE total_fee = total_fee + total_level_fee
        DISPLAY Total income of concatenate with level, total_level_fee and newline
        NEXT element
    ENDLOOP
    DISPLAY Total income of concatenate with month, year and newline
    DISPLAY Return to home page newline
ENDFUNCTION
```

Figure 30 View Monthly Income Report

3.7 Receptionist Functionalization

3.7.1 Receptionist Home Page

```
FUNCTION recep_ui with parameter user_id

    DECLARE recep_functions as a list which contain the function that can be used by receptionist
    such as Register and Enroll Students, View Student's Request and Update student subject enrollment,
    Generate Student Payment, Accept payment, Generate receipt, View and Delete Student, Add News, Delete News,
    Change Password, Update Profile and Log out

    DOWHILE True
        DISPLAY Receptionist Home Page
        LOOP index and function from index 0 = element 1 to last index = last element step 1 of recep_functions list
            CALCULATE index = index + 1
            DISPLAY index concatenate with function
            NEXT index and element
        ENDLOOP
        PROMPT user to enter number
        READ choice
        IF choice equal to string(1) THEN
            CALL register_stud function
        ELSE IF choice equal to string(2) THEN
            CALL update_stud function
        ELSE IF choice equal to string(3) THEN
            CALL generate_stud_payment function
        ELSE IF choice equal to string(4) THEN
            CALL accept_payment function where parameter user_id
        ELSE IF choice equal to string(5) THEN
            CALL generate_receipt function
        ELSE IF choice equal to string(6) THEN
            CALL delete_stud function
```

```
        ELSE IF choice equal to string(7) THEN
            CALL add_news function
        ELSE IF choice equal to string(8) THEN
            CALL delete_news function
        ELSE IF choice equal to string(9) THEN
            CALL change_password function where file_name is string(recep.txt), user_id parameter is string(user_id)
        ELSE IF choice equal to string(10) THEN
            CALL update_profile function where file_name is string(recep.txt), user_id parameter us string(user_id)
        ELSE IF choice equal to string(11) THEN
            DISPLAY Log out success
            RETURN out
        ELSE
            DISPLAY Invalid command
        ENDIF
    ENDDO
ENDFUNCTION
```

Figure 31 Receptionist Home Page

3.7.2 Register Student's Account Page

```
FUNCTION register_stud function without parameter
    DISPLAY newline Home Page >>> Register Students
    DISPLAY Register Students Home Page

    DECLARE stud_info_list as list which contain the the type of student information
    such as Student's Name, Student's IC or Passport, Student's Address, Student's Contact Number and Student's Email

    CALL view_levels_and_subjects function
    DECLARE new_stud_info as empty list
    DECLARE id_and_password as temporarily storage
    CALL generate_user_id_and_password function where role_name parameter is string(stud)
    id_and_password contain the return result of generate_user_id_and_password function
    DECLARE stud_id as temporarily storage
    stud_id contain the first element of id_and_password
    DISPLAY Student's ID generated: CONCATENATE with stud_id
    DECLARE stud_password as temporarily storage
    stud_password contain the second element of id_and_password
    ADD stud_id and stud_password TO new_stud_info list
    LOOP stud_data from first element to last element step 1 of stud_info_list list
        DOWHILE True
            PROMPT user to enter stud_data to register Student account
            READ data
            IF data is not empty string THEN
                BREAK DO
            ELSE
                DISPLAY You must enter something !!!
                CALL return_home_page function where continue_statement parameter is string(re-enter stud_data)
                IF the return result of return_home_page function is True THEN
                    RETURN None
```

```
            ENDIF
        ENDIF
    ENDDO
    ADD data TO new_stud_info list
    NEXT element
ENDLOOP
DECLARE month_of_enrollment temporarily storage
GET today's date (format is day month year) and store the date to month_of_enrollment
ADD month_of_enrollment TO new_stud_info list
DECLARE stud_level as temporarily storage
CALL get_and_verify_role_level_for_registration function where prompt_statement parameter is string(Student's level)
stud_level contain the return result of get_and_verify_role_level_for_registration function
IF stud_level is empty THEN
    RETURN None
ELSE
    ADD stud_level TO new_stud_info list
ENDIF
LOOP stud_subject_num FROM 1 TO 3 step 1
    DOWHILE TRUE
        PROMPT user to enter Student Subject stud_subject_num to register the account
        READ student_subject
        CONVERT all word in student_subject to capitalize
        DECLARE subject_info as temporarily file

        subject_info contain the return result of calling retrieve_level_or_subject function
        where levels_or_subjects parameter is string(subjects), level parameter is stud_level

        IF student_subject existed in subject_info OR student_subject is string(NULL) THEN
            ADD student_subject TO new_stud_info list
            BREAK DO
```

```
            ELSE
                DISPLAY newline Invalid Subject

                IF the return result of calling return_home_page function where continue_statement parameter
                is string(re-enter the Subject stud_subject_num) is TRUE THEN

                    RETURN None
                ENDIF
            ENDIF
        ENDDO
        NEXT number
    ENDLOOP
    CALL register function where file_name parameter is string(stud.txt), register_list parameter is new_stud_info
    CALL generate_payment_after_register where stud_id parameter is stud_id
    DISPLAY Register Student Success
    DISPLAY Add Student Education fee list of this month Success
    DISPLAY Return to home page
ENDFUNCTION
```

Figure 32 Register Student's Account Page

3.7.3 Generate Student's Payment Information for Registration Month Function

```
FUNCTION generate_payment_after_register with parameter stud_id
    DECLARE stud_info_id as temporarily file

    CALL find_one function where file_name parameter is string(stud.txt),
    start_reference_column parameter is integer(0),end_reference_column parameter is integer(1),
    reference_item parameter is stud_id

    stud_info_id contain the return result of find_one function
    DECLARE date_of_education_fee as temporarily storage
    GET today's date (format is month year) and store the date to date_of_education_fee
    DECLARE date_of_payment_done as string(NULL)
    DECLARE level as temporarily storage
    level contain the ninth element of stud_info_id
    DECLARE subject1 as temporarily storage
    subject1 contain the tenth element of stud_info_id
    DECLARE subject2 as temporarily storage
    subject2 contain the eleventh element stud_info_id
    DECLARE subject3 as temporarily storage
    subject3 contain the twelfth element of stud_info_id
    DECLARE subject1_fee as temporarily file

    subject1_fee contain the return result of calling retrieve_subjects_fee function
    where subject_name parameter is subject1

    DECLARE subject2_fee as temporarily file

    subject2_fee contain the return result of calling retrieve_subjects_fee function
    where subject_name parameter is subject2
```

```
    DECLARE subject3_fee as temporarily file

    subject3_fee contain the return result of calling retrieve_subjects_fee function
    where subject_name parameter is subject3

    DECLARE payment_status as string(UNPAID)
    DECLARE accept_payment_by as string(NULL)

    DECLARE new_payment_info as list which contain stud_id, date_of_education_fee, level, date_of_payment_done,
    subject1, subject2, subject3, subject1_fee, subject2_fee, subject3_fee, payment_status and accept_payment_by

    CALL register function where file_name parameter is string(payment.txt),register_list parameter is new_payment_info
ENDFUNCTION
```

Figure 33 Generate Student's Payment Information for Registration Month

3.7.4 View Student Request which in Still Pending Status Function

```
FUNCTION view_stud_request without parameter
    DECLARE stud_request_list_pending as temporarily file

    CALL find_more function where file_name parameter is string(request.txt), start_reference_column
    parameter is integer(5), end_reference_column is None, reference_item is string(STILL PENDING)

    stud_request_list_pending contain the return result of find_more function
    IF stud_request_list_pending is empty THEN
        DISPLAY No available student requests in still pending
        DISPLAY Return to home page newline
        RETURN None
    ENDIF
    LOOP stud_request_info_pending from first list to last list step 1 of stud_request_list_pending
        DISPLAY Student ID: CONCATENATE the first element of stud_payment_info_unpaid
        DISPLAY Month: CONCATENATE the second element of stud_payment_info_unpaid
        DISPLAY Level: CONCATENATE the third element of stud_payment_info_unpaid
        DISPLAY Payment Detail: Subject 1 CONCATENATE the fifth element of stud_payment_info_unpaid
                CONCATENATE (RM CONCATENATE the eighth element of stud_payment_info_unpaid) |
                CONCATENATE Subject 2 CONCATENATE the sixth element of stud_payment_info_unpaid
                CONCATENATE (RM CONCATENATE the ninth element of stud_payment_info_unpaid) |
                CONCATENATE Subject 3 CONCATENATE the seventh element of stud_payment_info_unpaid
                (RM CONCATENATE the tenth element of stud_payment_info_unpaid)
        DISPLAY Payment Status: CONCATENATE the eleventh element of stud_payment_info_unpaid
        DISPLAY ~ which repeated 100 times
    ENDLOOP
    RETURN 1
ENDFUNCTION
```

Figure 34 View Student's Request which in Still Pending status

3.7.5 Update Student's Subject Enrolment Page

```
FUNCTION update_stud without parameter
    DISPLAY newline Home Page >>> View Student's Request and Update Student Enrollment Subject
    DISPLAY View Student's Request and Update Student Enrollment Subject Home Page
    CALL view_stud_request function
    IF the return result of view_stud_request is empty THEN
        RETURN None
    ENDIF
    CALL return_home_page function where continue_statement parameter is string(update student information)
    IF the return result of return_home_page function IS TRUE
    THEN
        RETURN None
    ENDIF

    DOWHILE TRUE
        DECLARE request_no as temporarily file
        PROMPT user to enter the Request No. to approve or reject the request
        READ request_no
        IF request_no is a digit string THEN
            DECLARE request_info_request_no as temporarily file

            CALL return result of find_one function where file_name parameter is string(request.txt),
            start_reference_column parameter is integer(0), end_reference_column parameter is integer(1),
            reference_item parameter is string(request_no) and store to request_info_request_no

            IF request_info_request_no is not empty THEN
                BREAK DO
            ENDIF
        ENDIF
        DISPLAY Invalid Request No.
```

```
    IF the return result of calling return_home_page function where continue_statement parameter
    is string(re-enter the request no.) IS True THEN
        RETURN None
    ENDIF
ENDDO
DECLARE stud_id as second element of request_info_request_no
DECLARE change_subject as third element of request_info_request_no
DECLARE new_subject as fourth element of request_info_request_no
DECLARE stud_info_id as temporarily storage

CALL return result of find_one function where file_name parameter is string(stud.txt), + start_reference_column
parameter is integer(0), end_reference_column parameter is integer(1), reference_item is string(stud_id)

DECLARE change_subject_column as temporarily storage
FIND the index of change_subject where in stud_info_id and store the index to change_subject_column
DOWHILE TRUE:
    PROMPT user for approve_reject
    READ approve_subject
    CONVERT all words in approve_subject to capitalize
    IF approve_reject is APPROVE or REJECT THEN
        BREAK DO
    ELSE
        DISPLAY You can only enter APPROVE or REJECT
        IF the return result of calling return_home_page function where continue_statement parameter
        is string(re-enter APPROVE or REJECT) is True THEN
            RETURN None
        ENDIF
    ENDIF
ENDDO
```

```
    IF approve_reject equal to APPROVE THEN
        CALL edit function where file_name parameter is string(stud.txt), start_reference_column parameter is
        integer(0), end_reference_item parameter is integer(1), reference parameter is string(stud_id),
        edit_column parameter is int(change_subject_column), edit_item parameter is string(new_subject)

        CALL edit function where file_name parameter is string(request.txt), start_reference_column parameter is
        integer(0), end_reference_column parameter is integer(1), reference parameter is string(stud_id), edit_column
        paremeter is integer(change_subject_column), edit_item parameter is string(APPROVED)

        DISPLAY newline Student's Request approved and Update Student Enrollment Subject Success newline
        DISPLAY Return to home page newline
    ELSE
        CALL edit function where file_name parameter is string(request.txt), start_reference_column parameter is
        integer(0), end_reference_column parameter is integer(1), reference parameter is string(request_no),
        edit_column is integer(5), edit_item parameter is string(REJECTED)

        DISPLAY newline Student's Request rejected newline
        DISPLAY Return to home page newline
    ENDIF
ENDFUNCTION
```

Figure 35 Update Student's Subject Enrolment Page

3.7.6 View Student Payment which in Unpaid Status Function

```
FUNCTION view_student_payment_in_unpaid without parameter
    DECLARE stud_payment_list_unpaid as temporarily storage

    CALL find_more function where file_name parameter is string(payment.txt),
    start_reference_column parameter is integer(10), end_reference_column parameter is integer(11), reference_item
    parameter is string(UNPAID)

    stud_payment_list_unpaid contain the return result of find_more function
    IF stud_payment_list_unpaid is empty THEN
        DISPLAY Unavailable of student unpaid payment
        RETURN None
    ENDIF
    SORT stud_payment_list_unpaid order by the first element of stud_payment_list_unpaid list
    LOOP stud_payment_info_unpaid from first list to last list step 1 of stud_payment_list_unpaid list
        DISPLAY Student ID: CONCATENATE the first element of stud_payment_info_unpaid
        DISPLAY Month: CONCATENATE the second element of stud_payment_info_unpaid
        DISPLAY Level: CONCATENATE the third element of stud_payment_info_unpaid
        DISPLAY Payment Detail: Subject 1 CONCATENATE the fifth element of stud_payment_info_unpaid
                CONCATENATE (RM CONCATENATE the eighth element of stud_payment_info_unpaid) |
                CONCATENATE Subject 2 CONCATENATE the sixth element of stud_payment_info_unpaid
                CONCATENATE (RM CONCATENATE the ninth element of stud_payment_info_unpaid) |
                CONCATENATE Subject 3 CONCATENATE the seventh element of stud_payment_info_unpaid
                CONCATENATE (RM CONCATENATE the tenth element of stud_payment_info_unpaid)
        DISPLAY Payment Status: CONCATENATE the eleventh element of stud_payment_info_unpaid
        DISPLAY ~ which repeated 100 times
        NEXT list
    ENDLOOP
    RETURN stud_payment_list_unpaid
ENDFUNCTION
```

Figure 36 View Student Payment which in Unpaid Status

3.7.7 Generate All Student's Payment Information for the month Page

```
FUNCTION generate_stud_payment without parameter
    DISPLAY newline Home Page >>> Generate Student Payment
    DISPLAY Generate Student Payment Home Page

    IF the return result of calling return_home_page function where continue_statement parameter is
    string(generate all student education fee list in this month) IS True THEN

        RETURN None
    ENDIF
    DECLARE stud_list as temporarily file
    CALL read_all function where file_name parameter is string(stud.txt) and store to stud_list
    stud_list contain the return result of read_all function
    DECLARE date_of_generate_education_fee as temporarily file
    GET today's date (format is month year) and store the date to date_of_generate_education_fee
    LOOP stud_info from first list to last list step 1 of stud_list list
        DECLARE stud_id as the first element of stud_info
        DECLARE stud_level as the ninth element of stud_info
        DECLARE payment_date as string(NULL)
        DECLARE subject1 as the tenth element of stud_info
        DECLARE subject2 as the eleventh element of stud_info
        DECLARE subject3 as the twelfth element of stud_info
        DECLARE subject1_fee as temporarily file

        CONVERT the return result of calling the retrieve_subjects_fee function
        where subject_name parameter is subject1 to string and store the string to subject1_fee

        DECLARE subject2_fee as temporarily file

        CONVERT the return result of calling the retrieve_subjects_fee function
        where subject_name parameter is subject2 to string and store the string to subject2_fee
```

```
        DECLARE subject3_fee as temporarily file

        CONVERT the return result of calling the retrieve_subjects_fee function
        where subject_name parameter is subject3 to string and store the string to subject3_fee

        DECLARE payment_status as temporarily storage
        payment_status contain string(UNPAID)
        DECLARE accept_payment_by as temporarily storage
        accept_payment_by contain string(NULL)

        DECLARE new_payment_info as list which contain stud_id,
        date_of_generate_education_fee, stud_level, payment_date, subject1, subject2,subject3, subject1_fee,
        subject2_fee, subject3_fee, payment_status and accept_payment_by

        CALL register function where file_name parameter is string(payment.txt), register_list parameter is
        new_payment_info
        NEXT list
    ENDLOOP
    DISPLAY Generate Student Payment Success
    DISPLAY Return to home page
ENDFUNCTION
```

Figure 37 Generate All Student's Payment Information for the month Page

3.7.8 Accept Student's Payment Page

```
FUNCTION accept_payment with parameter user_id
    DISPLAY newline Home Page >>> Accept payment
    DISPLAY Accept Payment Home Page
    DECLARE stud_payment_list_unpaid as temporarily storage
    CALL view_student_payment_in_unpaid function
    stud_payment_list_unpaid contain the return result of view_student_payment_in_unpaid function
    IF stud_payment_list_unpaid is empty THEN
        RETURN None
    ENDIF

    IF the return result of calling return_home_page function where continue_statement
    is string(accept payment) IS True THEN

        RETURN None
    ENDIF
    DECLARE stud_id as temporarily file
    CALL get_and_verify_stud_id_in_payment_file function where prompt_statement parameter is string(accept the payment)
    stud_id contain the return result of the get_and_verify_stud_id_in_payment_file function
    IF stud_id is empty THEN
        RETURN None
    ENDIF
    DECLARE month as temporarily storage
    month contain the return result of calling get_and_verify_month function
    DECLARE year as temporarily storage
    year contain the return result of calling get_and_verify_year function
    IF month is empty OR year is empty THEN
        RETURN None
    ENDIF
    DECLARE date as temporarily storage
```

```
date contain month concatenate year
DECLARE stud_payment_list_id as temporarily storage

stud_payment_list_id contain the return result of calling find_more function where file_name parameter
is string(payment.txt), start_reference_column parameter is integer(0),
end_reference_column parameter is integer(1) and reference_item is stud_id

DECLARE stud_payment_list_date as temporarily file

stud_payment_list_date contain the return result of calling find_more function where file_name parameter
is string(payment.txt), start_reference_column parameter is integer(1),
end_reference_column parameter is integer(2) and reference_item is stud_id

DECLARE stud_payment_list_id_date_unpaid as temporarily storage
LOOP stud_payment_info_unpaid from first list to last list step 1 of stud_payment_list_unpaid list

    IF stud_payment_info_unpaid existed in stud_payment_list_date AND stud_payment_info_unpaid existed in
    stud_payment_list_id THEN

        stud_payment_list_id_date_unpaid contain stud_payment_info_unpaid
    ENDIF
    NEXT list
ENDLOOP
IF stud_payment_list_id_date_unpaid is empty THEN
    DISPLAY Unavailable of student payment which contain required ID, Month and Year
    RETURN None
ELSE
    DECLARE payment_date as temporarily file
    GET today's date (format is month year) and store the date to payment_date
ENDIF
```

```
    DECLARE recep_name as temporarily file

    recep_name contain the third element in the return result of calling find_one function
    where file_name parameter is string(recep.txt), start_reference_column parameter is integer(0),
    end_reference_column is integer(1) and reference_item parameter is user_id

    CALL edit function where file_name parameter is string(payment.txt), start_reference_column parameter is integer(0),
    end_reference_column is integer(2), reference parameter is string(stud_id CONCATENATE date),
    edit_column is integer(11), edit_item parameter is recep_name

    CALL edit function where file_name parameter is string(payment.txt), start_reference_column parameter is integer(0),
    end_reference_column is integer(2), reference parameter is string(stud_id CONCATENATE date),
    edit_column is integer(3), edit_item parameter is payment_date

    CALL edit function where file_name parameter is string(payment.txt), start_reference_column parameter is integer(0),
    end_reference_column is integer(2), reference parameter is string(stud_id CONCATENATE date),
    edit_column is integer(10), edit_item parameter is string(PAID)

    DISPLAY Accept Payment Success
    DISPLAY Return to home page
ENDFUNCTION
```

Figure 38 Accept Student's Payment Page

3.7.9 Generate Student's Payment Receipt Page

```
FUNCTION generate_receipt without parameter
    DISPLAY newline Home Page >>> Generate Receipt"
    DISPLAY Generate Receipt Home Page
    DECLARE stud_payment_list_paid as temporarily file

    stud_payment_list_paid contain the return result of calling find_more function
    where file_name parameter is string(payment.txt), start_reference_column parameter is integer(10),
    end_reference_column parameter is integer(11), reference_item is string(PAID)

    IF stud_payment_list_paid is empty THEN
        DISPLAY No available of Student Payment which is in paid status
        DISPLAY Return to home page
        RETURN None
    ENDIF
    DECLARE stud_id as temporarily file

    stud_id contain the return result of calling get_and_verify_stud_id_in_payment_file function
    where prompt_statement parameter is string(generate the receipt)

    IF stud_id is empty THEN
        RETURN None
    ENDIF
    DECLARE month as temporarily storage
    month contain the return result of the calling of get_and_verify_month function
    DECLARE year as temporarily storage
    year contain the return result of the calling get_and_verify_year function
    IF month is empty OR year is empty THEN
        RETURN None
    ENDIF
    DECLARE date as string(month CONCATENATE year)
```

```
DECLARE stud_payment_list_id as temporarily storage

stud_payment_list_id contain the return result of calling find_more function
where file_name parameter is string(payment.txt), start_reference_column parameter is integer(0),
end_reference_column parameter is integer(1), reference_item is stud_id

DECLARE stud_payment_list_date as temporarily storage

stud_payment_list_date contain the return result of calling find_more function
where file_name parameter is string(payment.txt), start_reference_column parameter is integer(1),
end_reference_column parameter is integer(2) and reference_item is date

DECLARE stud_payment_list_id_date_paid as temporarily storage
LOOP stud_payment_info_paid from first list to last list step 1 of stud_payment_list_paid list

    IF stud_payment_info_paid existed in stud_payment_list_date AND stud_payment_info_paid existed in
    stud_payment_list_id THEN

        stud_payment_list_id_date_paid contain stud_payment_info_paid
    ENDIF
    NEXT list
ENDLOOP
```

```
    IF stud_payment_list_id_date_paid is empty THEN
        DISPLAY No available student payment which contains required ID, Month, Year, and Status
        DISPLAY Return to home page
        RETURN None
    ELSE
        INITIALIZE total_amount_paid
        LOOP amount_paid from the eighth element to eleventh element step 1 of stud_payment_list_id_date_paid
            CONVERT amount_paid to integer
            total_amount_paid = total_amount_paid + amount_paid
            NEXT list
        ENDLOOP
        DECLARE stud_info_id as temporarily storage

        CALL find_one function where file_name parameter is string(stud.txt),
        start_reference_column parameter is integer(0), end_reference_column parameter is integer(1)
        and reference_item parameter is stud_id

        stud_info_id contain the return result of find_one function
        DISPLAY Student Receipt
        DISPLAY Student ID: CONCATENATE with the first element of stud_payment_list_id_date_paid
        DISPLAY Student Name: CONCATENATE with the third element of stud_info_id
        DISPLAY Month of Education fee: CONCATENATE with the second element of stud_payment_list_id_date_paid
        DISPLAY Payment Date: CONCATENATE with the fourth element of stud_payment_list_id_date_paid
        DISPLAY Total Amount: RM CONCATENATE with total_amount_paid
        DISPLAY Accept payment by Receptionist: CONCATENATE with the twelfth element of stud_payment_list_id_date_paid
        DISPLAY Student's Receipt Generated
        DISPLAY Return to home page
    ENDIF
ENDFUNCTION
```

Figure 39 Generate Student's Payment Receipt Page

3.7.10 Delete Student's Account Page

```
FUNCTION delete_stud without parameter
    DISPLAY newline Home Page >>> Delete Student
    DISPLAY Delete Student Home Page

    DECLARE stud_info_list as list which contain all type of student information
    such as Student's ID, Student's Password, Student's Name, Student's IC or Passport, Student's Address,
    Student's Email, Student's Contact Number, Month of Enrollmen

    DECLARE stud_list as temporarily storage
    CALL read_all function where file_name parameter is string(stud.txt)
    stud_list contain the return result of read_all function
    IF stud_list is empty THEN
        DISPLAY No student information
        RETURN None
    ENDIF
    SORT stud_list order by first element of stud_list list
    LOOP stud_info from first list to last list step 1 of stud_list list
        LOOP index and stud_data from index 0 = element 1 to index 8 = ninth element step 1 of stud_info
            IF index is not equal to 1 THEN
                DISPLAY the element of stud_info_list where index value is index CONCATENATE stud_data
            ENDIF
            NEXT index and stud_data
        ENDLOOP
        DISPLAY ~ which repeated 70 times
        NEXT list
    ENDLOOP
    CALL return_home_page function where continue_statement parameter is string(delete student's account)
    IF the return result of return_home_page function is True THEN
        RETURN None
    ENDIF
```

```
    DOWHILE TRUE
        PROMPT user to enter student id to delete the account
        READ stud_id
        DECLARE stud_info_id as temporarily storage

        CALL find_one function where file_name parameter is string(stud.txt), start_reference_column
        parameter is integer(0), end_reference_colmn parameter is integer(1) and reference_item parameter is stud_id

        stud_info_id contain the return result of find_one function
        DECLARE delete_row as temporarily storage
        IF stud_info_id is not empty THEN
            delete_row contain the index value of stud_info_id in stud_list
            BREAK DO
        ELSE
            DISPLAY This student id does not exist
            CALL return_home_page function where continue_statement parameter is string(re-enter the student's id)
            IF the return result of return_home_page function is True THEN
                RETURN None
            ENDIF
        ENDIF
        CALL delete function where file_name parameter is string(stud.txt) and delete_row parameter is delete_row
        DISPLAY Delete Student Information Success
        DISPLAY Return to home page
    ENDDO
ENDFUNCTION
```

Figure 40 Delete Student's Account Page

3.7.11 Add Tuition Centre's News Page

```
FUNCTION add_news without parameter
    DISPLAY newline Home Page >>> Add News
    DISPLAY Add News
    DECLARE new_news_str as empty string
    SET num = 1
    DISPLAY You can add tuition center news (Enter NULL if no more news)
    DOWHILE TRUE
        PROMPT user for to enter Paragraph num of news
        IF news_data is string(NULL) THEN
            BREAK DO
        ELSE
            ADD news_data CONCATENATE newline to new_news_str string
            num = num + 1
        ENDIF
    ENDDO
    DECLARE new_news_list as list which contain new_news_str
    CALL register function where file_name parameter is string(news.txt), register_list parameter is new_news_list
    DISPLAY Add News Success
    DISPLAY Return to home page
ENDFUNCTION
```

Figure 41 Add Tuition Centre's News Page

3.7.12 Delete Tuition Centre's News Page

```
FUNCTION delete_news without parameter
    DISPLAY newline Home Page >>> Delete News
    DISPLAY Delete News
    CALL view_news function
    IF the return result of view_news function is not empty THEN
        DISPLAY No any news
        RETURN None
    ENDIF

    IF the return result of calling return_home_page function where continue_statement parameter is string(delete news)
    is True THEN

        RETURN None
    ENDIF
    DOWHILE TRUE
        PROMPT user to enter the News No. that you would like to delete
        READ  delete_news_num
        TRY
            CONVERT delete_news_num to integer
            CALL read_all function where file_name parameter is string(news.txt)
            IF delete_news_num is between 0 and the length of the return result of read_all function THEN
                BREAK DO
            ELSE
                DISPLAY Invalid news no
            ENDIF
```

```
        CATCH ValueError
            DISPLAY Invalid news no

        IF the return result of calling return_home_page where continue_statement parameter
        is string(re-enter the News Number) is True THEN

            RETURN None
        ENDIF
    ENDDO
    CALCULATE delete_news_num = delete_news_num - 1
    CALL delete function where file_name parameter is string(news.txt), delete_row parameter is delete_news_num
    DISPLAY Delete News Success
    DISPLAY Return to home page
ENDFUNCTION
```

Figure 42 Delete Tuition Centre's News Page

3.8 Tutor Functionalization

3.8.1 Tutor Home Page

```
FUNCTION tutor_ui with parameter user_id
    DECLARE tutor_functions as a list which contain the function service that tutor can be used
    such as Add class information, Update class information, Delete Class Information,
    View Student List in the Subject, Change Password, Update Profile and Log out

    DOWHILE True
        DISPLAY Tutor Home Page
        LOOP index and function from index 0 = element 1 to last index = last element step 1 of tutor_functions
            CALCULATE index = index + 1
            DISPLAY index concatenate with function
            NEXT index and element
        ENDLOOP
        PROMPT user to enter a number for their choice
        READ choice
        IF choice is string(1) then
            CALL add_class function where user_id parameter is user_id
        ELSE IF choice is string(2) then
            CALL update_class function where user_id parameter is user_id
        ELSE IF choice is string(3) then
            CALL delete_class function where user_id parameter is user_id
        ELSE IF choice is string(4) then
            CALL view_stud function where user_id parameter is user_id
        ELSE IF choice is string(5) then

            CALL change_password function where user_file parameter is string(tutor.txt)
            and user_id parameter is user_id
```

```
        ELSE IF choice is string(6) then

            CALL update_profile function where user_file parameter is string(tutor.txt)
            and user_id parameter is user_id

        ELSE IF choice is string(7) then
            DISPLAY Log out success
            RETURN out
        ELSE
            DISPLAY Invalid command
        ENDIF
    ENDDO
ENDFUNCTION
```

Figure 43 Tutor Home Page

3.8.2 Add Class Schedule Page

```
FUNCTION add_class with parameter user_id
    DISPLAY newline Home Page >>> Add Class Schedule
    DISPLAY Add Class Schedule
    DECLARE tutor_info as temporarily storage

    tutor_info contain the return result of calling find_one function where file_name parameter is string(tutor.txt),
    start_reference_column parameter is integer(0), end_reference_column parameter is integer(1)
    and reference_item parameter is user_id

    DECLARE tutor_name as temporarily storage
    tutor_name contain the third element of tutor_info
    DECLARE tutor_assign_level as temporarily storage
    tutor_assign_level contain the eighth element of tutor_info
    DECLARE tutor_assign_subject as temporarily storage
    tutor_assign_subject contain the ninth element of tutor_info
    DECLARE subject_charge as temporarily storage
    CALL retrieve_subjects_fee function where subject_name parameter is tutor_assign_subject
    CONVERT the return result of retrieve_subjects_fee function to string and store the string to subject_charge
    DECLARE class_date as temporarily storage
    class_date contain the return result of calling get_and_verify_class_date function
    IF class_date is empty then
        RETURN None
    ENDIF
    DECLARE class_venue as temporarily storage
    class_venue contain the return result of calling get_and_verify_class_venue function
    IF class_venue is empty then
        RETURN None
    ENDIF
    DECLARE class_start_time as temporarily storage
```

```
    class_start_time contain the return result of calling get_and_verify_class_time where start_or_end parameter
    is string(Start)

    IF class_start_time is empty then
        RETURN None
    ENDIF
    DECLARE class_end_time as temporarily storage

    class_end_time contain the return result of calling get_and_verify_class_time function where start_or_end parameter
    is string(End)

    IF class_end_time is empty then
        RETURN None
    ENDIF

    DECLARE new_class_info as list which contain the class's information
    such as class_date, class_venue, class_start_time, class_end_time, tutor_assign_level, tutor_assign_subject,
    subject_charge, tutor_name and user_id

    CALL register function where file_name parameter is string(class.txt) and register_list parameter is new_class_info
    DISPLAY newline Class information has been saved newline newline Return to home page newline
ENDFUNCTION
```

Figure 44 Add Class Schedule Page

3.8.3 Tutor View Class Schedule Page

```
FUNCTION tutor_view_class with parameter user_id
    DECLARE class_info_list as a list which contain all type of class information
    such as Date, Venue, Start time, End time, Level, Subject, Charge, Tutor name and Tutor id

    DECLARE tutor_class_list as temporarily storage

    CALL find_more function where file_name parameter is string(class.txt), start_reference_column
    parameter is integer(8), end_reference_column parameter is None and reference_item parameter is string(user_id)

    tutor_class_list contain the return result of find_more function
    IF class_list is empty THEN
        DISPLAY No class schedule
        RETURN None
    END IF
    SORT class_list order by date and start time

    LOOP index and tutor_class from index 0 = list 1 to last index = last list step 1 of tutor_class_list
        DECLARE tutor_class_str as empty string
        CALCULATE index = index + 1
        ADD Class No. concatenate with index to tutor_class_str

        LOOP class_data_num and class_data from index 0 = element 1 to last index = last element
        step 1 of class_info_list list

            ADD newline class_data concatenate with the element of tutor_class where index value is class_data_num
            to tutor_class_str
```

```
            NEXT index and element
        ENDLOOP
        ADD newline concatenate with string(~) which repeated 70 times
        NEXT index and list
    ENDLOOP
    RETURN tutor_class_list
ENDFUNCTION
```

Figure 45 Tutor View Class Schedule Page

3.8.4 Update Class Schedule Page

```
FUNCTION update_class with parameter user_id
    DISPLAY Home Page >>> Update Class Information
    DISPLAY Update Class Information
    DECLARE class_list as temporarily storage
    CALL tutor_view_class function where file_name parameter is user_id
    class_list contain the return result of tutor_view_class function

    DECLARE class_info_list as a list which contain all type of class information
    such as Date, Venue, Start time, End time, Level, Subject, Charge, Tutor name and Tutor id

    IF class_list is empty THEN
        RETURN None
    ENDIF
    CALL return_home_page function where continue_statement parameter is string(update class schedule)
    If the return result of return_home_page function is True then
        RETURN None
    ENDIF
    DOWHILE True
        PROMPT user to enter Class No. to update the class information
        READ update_choice
        TRY
            CONVERT update_choice to integer
            IF update_choice between 1 and the length of class_list then
                DECLARE chosen_class as temporarily storage
                CALCULATE update_choice = update_choice - 1
                chosen_class contain the list of class_list where the index value is update_choice
                BREAK DO
            ELSE
                Display Class No. does not exist!!!
                CALL return_home_page function where continue_statement parameter is string(re-enter the class no.)
```

```
            If the return result of return_home_page function is True then
                RETURN None
            ENDIF
        ENDIF
    CATCH ValueError
        DISPLAY Class No. does not exist!!!
        CALL return_home_page function where continue_statement parameter is string(re-enter the class no.)
        IF the return result of return_home_page function is True THEN
            RETURN None
        ENDIF
ENDDO
DOWHILE True
    PROMPT user to enter class information to edit the information
    READ class_data
    IF class_data is existed in the elements of class_info_list list where the index value is smaller than 4 then
        DECLARE class_column as temporarily storage
        class_column contain the index value of class_data where in class_info_list list
        BREAK DO

    ELSE IF class_data is exited in the elements of class_info_list list where the index value
    is greater and equal to 4 then

        DISPLAY newline You cannot update this information newline Please enter again
    ELSE
        DISPLAY Invalid class information!!!
        CALL return_home_page function where continue_statement parameter is string(re-enter the class information)
        IF return_home_page("re-enter the class information") is True then
            RETURN None
        ENDIF
```

```
        ENDIF
    ENDDO
    DECLARE new_data as temporarily storage
    IF class_data is string(Date) then
        CALL get_and_verify_class_date function
        new_data contain the return result of get_and_verify_class_date function
    ELIF class_data is string(Venue) then
        CALL get_and_verify_class_venue function
        new_data contain the return result of get_and_verify_class_venue function
    ELIF class_data is string(Start Time) then
        CALL get_and_verify_class_time function where start_or_end parameter is string(Start)
        new_data contain the return result of get_and_verify_class_time function
    ELSE
        CALL get_and_verify_class_time function where start_or_end parameter is string(End)
        new_data contain the return result of get_and_verify_class_time function
    ENDIF
    IF new_data is empty then
        RETURN None
    ENDIF
    DECLARE chosen_class_str as temporarily storage
    JOIN each element of chosen_class with comma and store to chosen_class_str

    CALL edit function where file_name parameter is string(class.txt), start_reference_column parameter is integer(0),
    end_reference_column parameter is None, reference parameter is chosen_class_str,
    edit_column parameter is class_column and edit_item parameter is new_data

    DISPLAY Update Class success
ENDFUNCTION
```

Figure 46 Update Class Schedule Page

3.8.5 Delete Class Schedule Page

```
FUNCTION delete_class with parameter user_id
    DISPLAY Home Page >>> Delete Class Information
    DISPLAY Delete Class Information
    DECLARE class_list as temporarily storage
    CALL tutor_view_class function where user_id parameter is user_id
    class_list contain the return result of tutor_view_class function
    IF class_list is empty then
        RETURN None
    ENDIF
    CALL return_home_page function where continue_statement parameter is string(delete class schedule)
    IF return_home_page("delete class schedule") is True then
        RETURN None
    ENDIF
    DOWHILE True
        PROMPT user to enter Class No. to delete the class information
        READ delete_choice
        TRY
            CONVERT delete_choice to integer
            IF delete_choice is between 1 and the length of class_list then
                BREAK DO
            ELSE
                DISPLAY newline Class No. does not exist!!!
                CALL return_home_page function where continue_statement parameter is string(re-enter the class no.)
                IF return_home_page("re-enter the class no.") is True then
                    RETURN None
                ENDIF
            ENDIF
        CATCH ValueError
            DISPLAY Class No. does not exist!!!
            CALL return_home_page function where continue_statement parameter is string(re-enter the class no.)
```

```
            IF return_home_page("re-enter the class no.") is True then
                RETURN None
            ENDIF
    ENDDO
    DECLARE chosen_class as temporarily storage
    CALCULATE delete_choice = delete_choice -1
    chosen_class contain the list of class_list where the index value is delete_choice
    CALL read_all function where file_name parameter is string(class.txt)

    LOOP class_row and class_list from index 0 = list 1 to last index = last list
    step 1 of the return result of read_all function

        IF chosen_class is class_list then
            DECLARE delete_row as temporarily storage
            delete_row contain class_row
            CALL delete function where file_name parameter is string(class.txt) and delete_row parameter is delete_row
            DISPLAY newline Delete Class Information Success newline newline Return to home page
            RETURN None
        ENDIF
        NEXT index and list
    ENDLOOP
ENDFUNCTION
```

Figure 47 Delete Class Schedule Page

3.8.6 View Student list in the Class Page

```
FUNCTION view_stud with parameter user_id
    DISPLAY Home Page >>> View Student List in the Subject
    DISPLAY View Student List in the Subject
    DECLARE tutor_info as temporarily storage

    CALL find_one function where file_name parameter is string(tutor.txt),
    start_reference_column parameter is integer(0), end_reference_column parameter is integer(1)
    and reference_item parameter is user_id

    tutor_info contain the return result of find_one function
    DECLARE assign_level as temporarily storage
    assign level contain the eighth element of tutor_info
    DECLARE assign_subject as temporarily storage
    assign_subject contain the ninth element of tutor_info
    DECLARE stud_list_assign_level as temporarily storage

    CALL find_more function where file_name parameter is string(stud.txt),
    start_reference_column parameter is integer(8), end_reference_column parameter is integer(9)
    and reference_item parameter is assign_level

    stud_list_assign_level contain the return result of find_more function
    DECLARE stud_list as temporarily storage
    CALL read_all function where file_name parameter is string(stud.txt)
    stud_list contain the return result of read_all function
    DECLARE stud_list_assign_subject as empty list
    LOOP stud_information from first list to last list step 1 of stud_list
        IF assign_subject is existed in the elements of stud_information where index value is greater than 8 then
            ADD stud_information to stud_list_assign_subject list
```

```
        ENDIF
        NEXT list
    ENDLOOP
    IF stud_list_assign_level is empty OR stud_list_assign_subject list is empty then
        DISPLAY No student in this class
        RETURN None
    ENDIF
    LOOP stud_info_assign_subject from first list to last list step 1 of stud_list_assign_subject
        IF stud_info_assign_subject is existed in stud_list_assign_level then

            DISPLAY Student ID: concatenate with the first element of stud_info_assign_subject,|  Student Name:
            and the third element of stud_info_assign_subject

        ENDIF
        NEXT list
    ENDLOOP
ENDFUNCTION
```

Figure 48 View Student List in The Class Page

3.9 Student Functionalization

3.9.1 Student Home Page

```
FUNCTION stud_ui with parameter user_id
    DECLARE stud_functions as a list which contain the function service that student can be used
    such as View Schedule, Send Request, View and Delete Request, View Payment, Change Password, Update Profile
    and Log out

    DOWHILE True
        DISPLAY Student Home Page
        LOOP index and function from index 0 = element 1 to last index = last element step 1 of stud_functions list
            CALCULATE index = index + 1
            DISPLAY Enter concatenate with index and function
            NEXT index and element
        ENDLOOP
        PROMPT user to enter number
        READ choice
        IF choice is string(1) then
            CALL stud_view_class function where user_id parameter is user_id
        ELSE IF choice is string(2) then
            CALL send_request function where user_id parameter is user_id
        ELSE IF choice is string(3) then
            CALL delete_request function where user_id parameter is user_id
        ELSE IF choice is string(4) then
            CALL view_payment function where user_id parameter is user_id
        ELSE IF choice is string(5) then

            CALL change_password function where user_file parameter is string(stud.txt)
            and user_id parameter is user_id
```

```
        ELSE IF choice is string(6) then

            CALL update_profile function where user_file parameter is string(stud.txt)
            and user_id parameter is user_id

        ELSE IF choice is string(7) then
            DISPLAY Log out success
            RETURN out
        ELSE
            DISPLAY Invalid command, please try again
        ENDIF
    ENDDO
ENDFUNCTION
```

Figure 49 Student Home Page

3.9.2 Student View Class Schedule Page

```
FUNCTION stud_view_class with parameter user_id
    DISPLAY newline Home Page >>> View Class Schedule
    DISPLAY Class Schedule
    DECLARE stud_info_id as temporarily storage

    CALL find_one function where file_name is string(stud.txt),
    start_reference_column is integer(0), end_reference_column is integer(1)
    and reference_item is user_id

    stud_info_id contain the return result of find_one function
    DECLARE stud_level as temporarily storage
    stud_level contain the ninth element of stud_info_id
    DECLARE stud_subjects as temporarily storage
    stud_subjects contain the elements of stud_info_id where the index value is greater than 8
    DECLARE stud_class_list_level_subjects as empty list
    LOOP stud_subject from first element to last element step 1 of stud_subjects
        DECLARE stud_class_list_level_subject as temporarily storage

        CALL find_more function where file_name parameter is class.txt,
        start_reference_column is integer(4), end_reference_column is integer(6)
        and reference_item parameter is stud_level concatenate with , and stud_subject

        stud_class_list_level_subject contain the return result of find_more function
        IF stud_class_list_level_subject is not empty then
            LOOP stud_class_info_level_subject from first list to last list step 1 of stud_class_list_level_subject
                ADD stud_class_info_level_subject to stud_class_list_level_subjects list
                NEXT list
            ENDLOOP
        ENDIF
        NEXT element
    ENDLOOP
```

```
    IF stud_class_list_level_subjects is empty then
        DISPLAY newline No Class Schedule newline newline Return to home page newline
        RETURN None
    ENDIF
    SORT stud_class_list_level_subjects order by date and time
    LOOP stud_class_info_level_subjects from first list to last list step 1 of stud_class_list_level_subjects
        DISPLAY Date:  concatenate with the first element of stud_class_info_level_subjects
        DISPLAY Venue:  concatenate with the second element of stud_class_info_level_subjects
        DISPLAY Time: From concatenate with the third element of stud_class_info_level_subjects
                ,to and the fourth element of stud_class_info_level_subjects
        DISPLAY Level: concatenate with the fifth element of stud_class_info_level_subjects
        DISPLAY Subject: concatenate with the sixth element of stud_class_info_level_subjects
        DISPLAY Charge: concatenate with the seventh element of stud_class_info_level_subjects
        DISPLAY Lecturer: concatenate with eighth element of stud_class_info_level_subjects
        DISPLAY ~ which repeated 30 times
        NEXT list
    ENDLOOP
    DISPLAY newline Return to home page newline
ENDFUNCTION
```

Figure 50 Student View Class Schedule Page

3.9.3 Send Request for Changing Subject Enrolment to Receptionist Page

```
FUNCTION send_request with parameter user_id
    DISPLAY Home Page >>> Send Request
    DISPLAY Send Request

    CALL view_levels_and_subjects function
    DECLARE stud_info as temporarily storage

    CALL find_one function where file_name parameter is string(stud.txt),
    start_reference_column parameter is integer(0), end_reference_column is integer(1)
    and reference_item parameter is user_id

    stud_info contain the return result of find_one function
    DECLARE stud_level as temporarily storage
    stud_level contain the ninth element of stud_info
    DECLARE subjects_list as temporarily storage

    CALL retrieve_level_or_subject function where levels_or_subjects parameter is string(subjects)
    and level parameter is stud_level

    subjects_list contain the return result of retrieve_level_or_subject function
    DECLARE current_subjects as temporarily storage
    current_subjects contain the elements of stud_info where index value is greater than 8
    DISPLAY (Your current enrolled subject)
    DISPLAY Subject 1: concatenate with the first element of current_subjects
    DISPLAY Subject 2: concatenate with the second element of current_subjects
    DISPLAY Subject 3: concatenate with the third element of current_subjects
    CALL return_home_page function where continue_statement parameter is string(send request)
    IF the return result of return_home_page function is True then
        RETURN None
    ENDIF
```

```
DOWHILE True
    PROMPT user to enter the subject number that you want to change
    READ ori_subject_num
    TRY
        CONVERT ori_subject_num to integer
        IF ori_subject_num is between 1 and 3 then
            DECLARE ori_subject as temporarily storage
            CALCULATE ori_subject_num = ori_subject_num - 1
            ori_subject contain the element of current_subjects where index value is ori_subject_num
            BREAK DO
        ELSE
            DISPLAY newline You must write in 1, 2, or 3 only
            CALL return_home_page function where continue_statement parameter is string(re-enter the subject number)
            IF the return result of return_home_page function is True then
                RETURN None
            ENDIF
        ENDIF
    CATCH ValueError
        DISPLAY newline You must write in 1, 2, or 3 only
        CALL return_home_page function where continue_statement parameter is string(re-enter the subject number)
        IF the return result of return_home_page function is True then
            RETURN None
        ENDIF
ENDDO
DOWHILE True
    PROMPT user to enter subject name that want to change to
    READ change_subject
    CONVERT each word in change_subject to capitalize
    IF change_subject is existed in subjects_list and change_subject is not existed in current_subjects then
        BREAK DO
```

```
        ELSE
            DISPLAY Invalid subject
            CALL return_home_page function where continue_statement parameter is string(re-enter the subject name)
            IF the return result of return_home_page function is True then
                RETURN None
            ENDIF
        ENDIF
    ENDDO
    PROMPT user to provide suitable reason for changing subject
    DECLARE request_list as temporarily storage
    CALL read_all function where file_name parameter is string(request.txt)
    request_list contain the return result of read_all function
    DECLARE current_request_no as empty string
    IF request_list is empty then
        ADD string(1) to current_request_no
    ELSE
        FIND the integer value of the first element in the last list of request_list
        CALCULATE request_no = the integer value + 1
        CONVERT request_no to string and add the string to current_request_no
    ENDIF

    DECLARE request_info as a list which contain current_request_no, user_id, ori_subject, change_subject, reason
    and string(STILL PENDING)

    CALL register function where file_name parameter is string(request.txt) and register_list parameter is request_info
    DISPLAY newline Request has been sent to receptionist newline newline Return to home page newline
END FUNCTION
```

Figure 51 Send Request for Changing Subject Enrolment to Receptionist Page

3.9.4 Delete Own Request for Changing Subject Enrolment Page

```
FUNCTION delete_request with parameter user_id
    DISPLAY newline Home Page >>> Delete Request
    DISPLAY View and Delete Request
    DECLARE stud_request_list as temporarily storage

    CALL find_more function where file_name parameter is string(request.txt),
    start_reference_column parameter is integer(1), end_reference_column parameter is integer(2)
    and reference_item parameter is user_id

    stud_request_list contain the return result of find_more function
    IF stud_request_list is empty then
        DISPLAY You have not sent any request
        RETURN None
    ELSE
        LOOP request_info from first list to last list step 1 of stud_request_list
            DISPLAY Request No. concatenate with the first element of request_info
            DISPLAY Student ID: concatenate with the second element of request_info
            DISPLAY From Subject: concatenate with the third element of request_info
            DISPLAY Change to Subject: concatenate with the fourth element of request_info
            DISPLAY Reason: concatenate with the fifth element of request_info
            DISPLAY Request Status: concatenate with the sixth element of request_info
            DISPLAY ~ which repeated 30 times
            NEXT list
        ENDLOOP
    ENDIF
    CALL return_home_page function where continue_statement parameter is string(delete request)
    IF the return result of return_home_page function is True then
        RETURN None
    ENDIF
    DOWHILE True
```

```
PROMPT user to enter Request No. for deletion
READ delete_line
IF delete_line is a digit then
    request_info_request_no as temporarily storage

    CALL find_one function where file_name parameter is string(request.txt),
    start_reference_column parameter is integer(0), end_reference_column parameter is integer(1)
    and reference_item parameter is delete_line

    request_info_request_no contain the return result of find_one function
    IF request_info_request_no is not empty then
        IF the last element of request_info_request_no is string(STILL PENDING) then
            BREAK DO
        ELSE
            DISPLAY You cannot delete this request because it has already been approved or rejected

            CALL return_home_page function where continue_statement parameter
            is string(re-enter the request No.)

            IF the return result of return_home_page function is True then
                RETURN None
            ENDIF
        ENDIF
    ELSE
        DISPLAY Request No. does not exist
        CALL return_home_page function where continue_statement parameter is string(re-enter the request No.)
        IF the return result of return_home_page function is True then
            RETURN None
        ENDIF
    ENDIF
```

```
        ELSE
            DISPLAY Invalid Request No.
            CALL return_home_page function where continue_statement parameter is string(re-enter the request No.)
            IF the return result of return_home_page function is True then
                RETURN None
            ENDIF
        ENDIF
    ENDDO
    CONVERT delete_line to integer
    CALCULATE delete_row = delete_line - 1
    CALL delete function where file_name parameter is string(request.txt) and delete_row parameter is delete_row
    DISPLAY Delete request success
    DISPLAY Return to home page
ENDFUNCTION
```

Figure 52 Delete Own Request for Changing Subject Enrolment Page

3.9.5 Student View Own Payment Information which in Unpaid Status Page

```
FUNCTION view_payment with parameter user_id
    DISPLAY newline Home Page >>> View Payment
    DISPLAY View Payment
    DECLARE payment_list_id as temporarily storage

    payment_list_id contain the return result of calling find_more function
    where file_name parameter is string(payment.txt), start_reference_column parameter is integer(0),
    end_reference_column parameter is integer(1) and reference_item parameter is user_id

    DECLARE payment_list_unpaid as temporarily storage

    payment_list_unpaid contain the return result of calling find_more function
    where file_name parameter is string(payment.txt), start_reference_column parameter is integer(10),
    end_reference_column parameter is integer(11) and reference_item parameter is string(UNPAID)

    DECLARE payment_list_id_unpaid as empty list
    IF payment_list_unpaid is empty then
        DISPLAY No payment needs to be paid
        RETURN None
    ELSE
        LOOP payment_info_unpaid from first list to last list step 1 of payment_list_unpaid
            IF payment_info_unpaid is existed in payment_list_id then
                ADD payment_info_unpaid to payment_list_id_unpaid list
            ENDIF
            NEXT list
        ENDLOOP
    ENDIF
    LOOP payment_info_id_unpaid from first list to last list step 1 of payment_list_id_unpaid
        DECLARE subject1_fee as temporarily storage
```

```
        subject1_fee contain the eighth element of payment_info_id_unpaid
        DECLARE subject2_fee as temporarily storage
        subject2_fee contain the ninth element of payment_info_id_unpaid
        DECLARE subject3_fee as temporarily storage
        subject3_fee contain the tenth element of payment_info_id_unpaid
        CONVERT subject1_fee, subject2_fee and subject3_fee to integer
        CALCULATE payable = subject1_fee + subject2_fee + subject3_fee
        DISPLAY Date: concatenate with the second element of payment_info_id_unpaid
        DISPLAY Level: concatenate with the third element of payment_info_id_unpaid
        DISPLAY Payment Status: concatenate with the eleventh element of payment_info_id_unpaid
        DISPLAY Payment Detail: Subject 1 concatenate with the fifth element of payment_info_id_unpaid
                and (RM concatenate with subject1_fee),
                | Subject 2 concatenate with the sixth element of payment_info_id_unpaid
                and (RM concatenate with subject2_fee),
                | Subject 3  concatenate with the seventh element of payment_info_id_unpaid
                and (RM concatenate with subject3_fee)
        DISPLAY Payable: RM concatenate with payable
        DISPLAY ~ which is repeated 70 times
        NEXT list
    ENDLOOP
    DISPLAY newline Kindly remind: Please pay your education fee before the due date. Thank You
    DISPLAY Return to home page newline
ENDFUNCTION
```

Figure 53 Student View Own Payment Information which in Unpaid Status Page

4.0 Explanation of Program Source Code

4.1 Variable

4.1.1 String

The string variable is a type of data that store text or a sequence of character (Programiz, 2023). In python, string is declared by either putting single quotation marks or double quotation marks around the data. The string variable is used heavily throughout the whole program. Mostly, string variable is used to get user input with the *input()* method. Other than that, the string variable also used to process the function, conditional expression, or Boolean expression where they serve as a temporarily file or a reference. In this case, the string serves as temporarily file to store the year inserted by user. The year string variable will then be converted to a "datetime" object which carry the year information.

```
while True:
    year = input("Please enter Year of the payment pay for (exp: 2023): ").strip()
    try:
        datetime.strptime(year, "%Y")
        return year
    except ValueError:
        print("Invalid Year")
        if return_home_page("re-enter the year"):
            return
```

Figure 54:String Usage for Receive User Input

Other than that, the string variable also used to process the function, conditional expression or Boolean expression where they serve as a temporarily file or a reference. In this case, the string variable serves as a condition for checking the role_name of the user by comparing the string with the if-else conditional statement.

```
def generate_user_id_and_password(role_name):
    # Auto generate the role's ID and password
    if role_name == "stud":
        check_file = "payment.txt"
    else:
        check_file = role_name + ".txt"
```

Figure 55: String Usage for Conditional Statement

The string variable also mostly used in displaying message and prompt for the user. For instance, the return_home_page function will display message to the user as long as the user input is invalid.

```
def return_home_page(continue_statement):
    # Give user a chance to return to home page
    while True:
        back_choice = input(f"Enter 1 to return to home page / Enter 0 to {continue_statement}: ")
        if back_choice == "1":
            print("\nReturn to home page\n")
            return True
        elif back_choice == "0":
            return False
        else:
            print("Invalid command!!!\nPlease enter right command")
```

Figure 56: Display Message for Users

4.1.2 List

List is built-in structure that can store multiple items in a variable (Python list, 2023). In most case, list main task is to collect a bulk of variable and combine them together as a variable. Lists also serve as a temporarily storage for execution of the code. With the help of *enumerate ()* function, the element within the list is able to be displayed just like the picture shown below.

```
def stud_ui(user_id):
    stud_functions = ["View Schedule", "Send Request", "View and Delete Request", "View Payment", "Change Password",
                      "Update Profile", "Log out"]
    while True:
        print("-----------------------Student Home Page-----------------------")
        for index, function in enumerate(stud_functions):
            print(f"Enter {index + 1}: {function}")
        choice = input("Enter number: ")
        if choice == "1":
            stud_view_class(user_id)
        elif choice == "2":
            send_request(user_id)
        elif choice == "3":
            delete_request(user_id)
        elif choice == "4":
            view_payment(user_id)
        elif choice == "5":
            change_password("stud.txt", user_id)
        elif choice == "6":
            update_profile("stud.txt", user_id)
        elif choice == "7":
            print("Log out success\n")
            return "out"
        else:
            print("Invalid command, please try again")
```

Figure 57 List used for storing all student useable function

Besides, the element in the list also can be selected out individually by printing out the index of the element. The figure below shows the execution of block for printing out element of list to the users.

```
for stud_class_info_level_subjects in stud_class_list_level_subjects:
    print(f"Date: {stud_class_info_level_subjects[0]}\n"
          f"Venue: {stud_class_info_level_subjects[1]}\n"
          f"Time: From {stud_class_info_level_subjects[2]} "
          f"to {stud_class_info_level_subjects[3]}\n"
          f"Level: {stud_class_info_level_subjects[4]}\n"
          f"Subject: {stud_class_info_level_subjects[5]}\n"
          f"Charge: {stud_class_info_level_subjects[6]}\n"
          f"Lecturer: {stud_class_info_level_subjects[7]}\n" + "~" * 30)
```

Figure 58 Display data by selecting specific element from the list

4.1.3 Dictionary

Dictionary is a data structure that store the data in a pair. The data in the dictionary is changeable but the data cannot have duplicates. Most of the time, dictionary is used to store data that is needed to be retrieved based on the unique keys. The code below demonstrated the subject, and the subject fee is created in the form of dictionary so the data can be retrieved based on the subject name.

```
def retrieve_subjects_fee(subject_name):
    # Retrieve subject fee which belong to the subject given
    subjects_fee = {"BAHASA MALAYSIA": 30,
                    "ENGLISH": 30,
                    "MATHS": 45,
                    "SCIENCE": 45,
                    "SEJARAH": 30,
                    "ADD MATHS": 50,
                    "ACCOUNTING": 45,
                    "ECONOMIC": 45,
                    "PHYSICS": 50,
                    "CHEMISTRY": 50,
                    "BIOLOGY": 50,
                    "NULL": 0
                    }
    subject_fee = subjects_fee.get(subject_name)
    return subject_fee
```

Figure 59 Dictionary used for storing subject name and its subject fee

4.2 Selection Structure

4.2.1 Multi-way decision (If and Else)

Multi-way decision is an identifying process to determine the input compared to the condition. For example, the program ask user to enter 1 for returning to home page or enter 0 for continuing the task. After that, the program compares the input with the IF statement, if the input is not 1 then it goes to the ELIF statement to do comparison. The program executes the code under ELSE statement once the input is False withing the comparison between the IF statement and the ELIF statement.

```
while True:
    back_choice = input(f"Enter 1 to return to home page / Enter 0 to {continue_statement}: ")
    if back_choice == "1":
        print("\nReturn to home page\n")
        return True
    elif back_choice == "0":
        return False
    else:
        print("Invalid command!!!\nPlease enter right command")
```

Figure 60 multi-way decision used for allowing user return to home page

4.2.2 Try-except Block

Try-except block is used multiple times in this program to prevent any error within the system. While the IF statement can be used to identify a Boolean value, it may lead to system errors when Python execute certain lines of code. For example, the program prompts the tutor to enter class date for adding a class schedule. The program converts the user's input to a date format (Day Month Year). However, it may cause a system error when the tutor enters an incorrect date format or an invalid value. Therefore, this error causes the program to terminate abnormally, which can affect user's experience. To address this problem, try-except block is used to allow the program to continue running when errors occur. When any line of code encounters an error, the program skips the remaining code under the try block and proceeds to execute the code under the except block. To illustrate, if an error occurs during the process of converting user input to a date format, the program executes *print ("Invalid date")* and then continues with the following code.

```
while True:
    class_date = input("Please enter Class Date (exp: 1 August 2023): ")
    try:
        datetime.strptime(class_date, "%d %B %Y")
        return class_date
    except ValueError:
        print("Invalid date")
        if return_home_page("re-enter class date"):
            return
```

Figure 61 Try-Except Block used for identifying the validation of user input

4.3 Repetitive Statement

4.3.1 For Loop

For loop is a repetitive control structure that will repeat the statement if the condition is satisfied. The for loop mostly be used to iterate the list, dictionary, tuple, set and string in this program. For instance, the *stud_info_list* list below is iterated by the for-loop statement to show the users correct prompt so that the users can key in their information correctly.

```
stud_info_list = ["Student's Name",
                  "Student's IC or Passport",
                  "Student's Address",
                  "Student's Contact Number",
                  "Student's Email"]
```

Figure 62 stud_info_list as list which contain the types of students data

In this case, the for loop iterate all the string from the *stud_info_list* list to prompt the user's input. The input value then will be assigned in a temporarily storage called data. Once the loop finished the iterations, the data will then be appended to a list which will later be registered inside the stud.txt file.

```
for stud_data in stud_info_list:
    while True:
        data = input(f"Please enter {stud_data} to register Student account: ").strip()
        if data != "":
            break
        else:
            print("You must enter something !!!")
            if return_home_page(f"re-enter {stud_data}"):
                return
    new_stud_info.append(data)
```

Figure 63 Loop stud_data from first element to last element step 1 of stud_info_list to prompt user to enter the specific data

For some cases, for-loop also uses *range ()* method to loop a statement. While using this method to loop a statement, loop also able to perform some basic calculation in the program. The diagram below shows that the while for-loop repeat statement to print out the three subjects, it also aids to retrieve the data from payment.txt file by assigning integer to the find_more function.

```
# Loop 1: Find student payment list which subject 1 is the subject
# Loop 2: Find student payment list which subject 2 is the subject
# Loop 3: Find student payment list which subject 3 is the subject
for subject_enrolled_num in range(1, 4):
    payment_list_subject_enrolled = find_more("payment.txt", 3 + subject_enrolled_num, 4
                                              + subject_enrolled_num, subject)
    if payment_list_subject_enrolled:
        payment_list_level = find_more("payment.txt", 2, 3, level)
```

Figure 64 Loop subject_enrolled_num from 1 to 3 step 1 to retrieve the information of student's payment which has paid for specific subject

4.3.2 While

While loop is another repetitive control structure that similar to for loop. However, instead of iterating variable like for loop, while loop is used in repeating a block of code as long as the condition is true. The while-loop often being used with the selection structure in this program to decide whether the following sequence of code needed to be executed or not. In the delete tutor section, the while-loop below will execute the code to collect the user input repeatedly until the user decide to quit or the user delete the tutor successfully.

```
while True:
    delete_tutor_id = input(f"Please enter Tutor ID to delete the account: ").strip()
    delete_tutor_info = find_one("tutor.txt", 0, 1, delete_tutor_id)
    if delete_tutor_info:
        tutor_delete_row = tutor_list.index(delete_tutor_info)
        break
    else:
        print("\nTutor's ID does not exist")
        if return_home_page("re-enter the tutor's id"):
            return
delete("tutor.txt", tutor_delete_row)
print("-"*30 + "\nDelete Tutor Success\n" + "-"*30 + "\nReturn to home page\n")
```

Figure 65 Use Do while to loop the code under it infinitely and stop the loop when reaching break or return statement

4.4 Function

4.4.1 User-Defined Function

User-defined function refer to the custom function that created by the programmer to perform specific task in the algorithms. Many user-defined functions are created in the program. For instance, the real_all function is created to read the data within the file. This statement is created as function as reading data from txt file is very common in this program.

```
def read_all(file_name):
    # Read all row which in the file given and store to list
    # Return the list
    read_list = []
    with open(file_name, mode="r") as read_file:
        for read_line in read_file:
            read_line_str = read_line.replace("\n", "")
            read_line_list = read_line_str.split(",")
            read_list.append(read_line_list)
    return read_list
```

Figure 66 function for reading all data from a specific txt file

In this user-defined function, it also uses the return statement, so the function is able to return the read_list value in the event of the function call. The figure below shows that the function call event that calling the return result of the read_line function where it returns the data from the payment.txt file and append them in the list.

```
def generate_user_id_and_password(role_name):
    # Auto generate the role's ID and password
    if role_name == "stud":
        check_file = "payment.txt"
    else:
        check_file = role_name + ".txt"
    all_user_id = []
    for user_info in read_all(check_file):
        all_user_id.append(user_info[0])
```

Figure 67 Calling read_all function to obtain data from a specific file in generate_user_id_and_password function

4.4.2 Built-In Function

Built-In function also known as predefined function, is a default function that already be provided in a program language. The programmer can just call the built-in function out to perform specific task. Some common built-in function included, *input ()*, *print ()*, *len ()*, *max ()* etc. Each built-in function has its own unique purpose and name in the program. Built-in function like *join ()* concatenates the list with the string of symbol ",". This *join ()* is often be seen when the program trying to store data in the txt file.

```
def find_more(file_name, start_reference_column, end_reference_column, reference_item):
    # Retrieve some row that match the reference from the particular file
    find_list = []
    for read_line in read_all(file_name):
        find_line = read_line[start_reference_column: end_reference_column]
        find_line_str = ",".join(find_line)
        if find_line_str == reference_item:
            find_list.append(read_line)
    return find_list
```

Figure 68 join function is used to join each element of find_line list by comma

These built-in functions are extremely vital in programming, they provide a more convenient way for user to perform tasks. Instead of writing complex code to perform tasks like string manipulation, mathematical calculations, or data input/output, programmers can rely on these functions which are designed to work efficiently and reliably.

4.5 File I/O

A Python file is a storage medium on the disc that allows data to be retained in non-volatile memory indefinitely. Unlike Random Access Memory (RAM), which clears all data when the program is closed, files are employed to store user's data, class schedules, payment information and news for extended period within this program. There are 3 modes for managing a file such as reading mode, writing mode, and appending mode. To use a file, the program must follow these steps, opening the file and specifying the mode, managing the file to perform various operations such as reading data from it or writing data to it and closing the file when done working with it to ensures any changes made are saved.

4.5.1 Reading Mode

This function opens a specific file and reads the data from the file row by row.

```
def read_all(file_name):
    # Read all row which in the file given and store to list
    # Return the list
    read_list = []
    with open(file_name, mode="r") as read_file:
        for read_line in read_file:
            read_line_str = read_line.replace("\n", "")
            read_line_list = read_line_str.split(",")
            read_list.append(read_line_list)
    return read_list
```

Figure 69 This function employ reading mode to open and read data from a specific file

4.5.2 Writing Mode

This function use writing mode to open a specific file and overwrite the contents of *edit_str* in the file.

```
def edit(file_name, start_reference_column, end_reference_column, reference, edit_column, edit_item):
    # Find the row that match the reference from the particular file
    edit_str = ""
    for read_line in read_all(file_name):
        find_line = read_line[start_reference_column: end_reference_column]
        find_line_str = ",".join(find_line)
        if find_line_str == reference:
            # Update one information in the row
            read_line[edit_column] = edit_item
        read_line_str = ",".join(read_line)
        edit_str += read_line_str + "\n"
    with open(file_name, mode="w") as change_file:
        change_file.write(edit_str)
```

Figure 70 This function use writing mode to open and overwrite the contents of edit_str to a specific file

4.5.3 Appending Mode

This function uses appending mode to open a specific file and write the contents of *register_item* to the bottom line of the file. Unlike writing mode, this mode does not overwrite the data in the file, which mean the original data remains intact within the file.

```
def register(file_name, register_list):
    # Add a row with information to the txt file
    register_item = ",".join(register_list) + "\n"
    with open(file_name, mode="a") as register_file:
        register_file.write(register_item)
```

Figure 71 This function utilizes appending mode to open and add the contents of register_item to the bottom line of a specific file

5.0 Explanation of Additional Features Source Code

There are 3 additional features such as view news function, add news function, delete news function, and encrypt password function.

5.1 Source code of view news function

```
1336  def view_news():
1337      with open("news.txt", "r") as news_file:
1338          news_info = news_file.readline()
1339          if not news_info:
1340              return False
1341          else:
1342              news_num = 1
1343              while news_info:
1344                  print(f"News {news_num}\n")
1345                  news = news_info.replace("\\n", "\n")
1346                  print(news + "~" * 70)
1347                  news_info = news_file.readline()
1348                  news_num += 1
1349              return True
```

Figure 72 Source Code of View News Function

This function provides an interface to all user to know and follow the event and activities at tuition centre. User are prompted to enter "2" to call *view_news()* for directing to this page at main page. It means user can direct to view news page, no need to login to their account. This function read all row from news.txt file where store the news information related to tuition centre and display each news to user.

5.2 Source code of add news function

```
910  def add_news():
911      print("\nHome Page >>> Add News")
912      print("-" * 30 + "Add News" + "-" * 30)
913      new_news_str = ""
914      num = 1
915      print("You can add tuition center news (Enter NULL if no more news)")
916      while True:
917          news_data = input(f"Paragraph {num}: ")
918          if news_data.upper() == "NULL":
919              break
920          else:
921              new_news_str += news_data + "\\n"
922              num += 1
923      new_news_list = [new_news_str]
924      register("news.txt", new_news_list)
925      print("-"*30 + "\nAdd News Success\n" + "-"*30 + "\nReturn to home page\n")
```

Figure 73 Source Code of Add News Function

To ensure the news related to tuition centre are up to date, receptionist allow to add news and store to news.txt file. This function is one of the functionalities of receptionist since they are responsibility to manage event and activities related to students which mean this function only can be called at Receptionist Home Page. Receptionists are prompted to enter “7” to call *add_news()* for directing to this page at Receptionist Home Page.

5.3 Source code of delete news function

```
928  def delete_news():
929      print("\nHome Page >>> Delete News")
930      print("-" * 30 + "Delete News" + "-" * 30)
931      if not view_news():
932          print("No any news")
933          return
934      if return_home_page("delete news"):
935          return
936      while True:
937          delete_news_num = input("Please enter the News No. that you would like to delete: ")
938          try:
939              delete_news_num = int(delete_news_num)
940              if 0 < delete_news_num <= len(read_all("news.txt")):
941                  break
942              else:
943                  print("Invalid news no")
944          except ValueError:
945              print("Invalid news no")
946          if return_home_page("re-enter the News Number"):
947              return
948      delete("news.txt", delete_news_num-1)
949      print("-"*30 + "\nDelete News Success\n" + "-"*30 + "\nReturn to home page\n")
```

Figure 74 Source Code of Delete News Function

When an event or activity passed, receptionist allow to delete the outdated news. This function is also one of the functionalities of receptionist since receptionist can add news, logically must have authorization to delete news to manage the news effectively. Receptionists are prompted to enter "8" to *delete_news()* for directing to this page at Receptionist Home Page.

5.4 Source code of encrypt password function

```
202  def crypt_password(user_password):
203      # Encrypt password by using hash algorithm
204      sha256 = hashlib.sha256()
205      sha256.update(user_password.encode("utf-8"))
206      user_password_sha256 = sha256.hexdigest()
207      return user_password_sha256
```

Figure 75 Source Code of Encrypt Password Function

In this information generation, system security is a crucial component to protect user's personal information and account's data. This function enhances the security of user's account to prevent non owner individual to login account. The system call *crypt_password(user_password)* to encrypt user password by using SHA 256 algorithm, *user_password* is parameter, for example, the system will call *crypt_password(tutor0001@B0001)* when need to encrypt the user's password, tutor0001@B0001. SHA 256 is one part of SHA 2 (Secure Hash Algorithm 2) family which were designed by the National Security Agency (NSA) in the United States. (Baivab Kumar Jena, 2023) The hashing process of SHA 256 is taking an input and produces a fixed-size 256-bit (32-byte) hexadecimal number. This algorithm has consistency and deterministic which can produce the same hash value with the same input. SHA 256 is suitable used to protect the user's data such as password because it is a one-way function which mean cannot obtain original value from its hash value.

6.0 Sample of Input and Output

6.1 Login Account Page

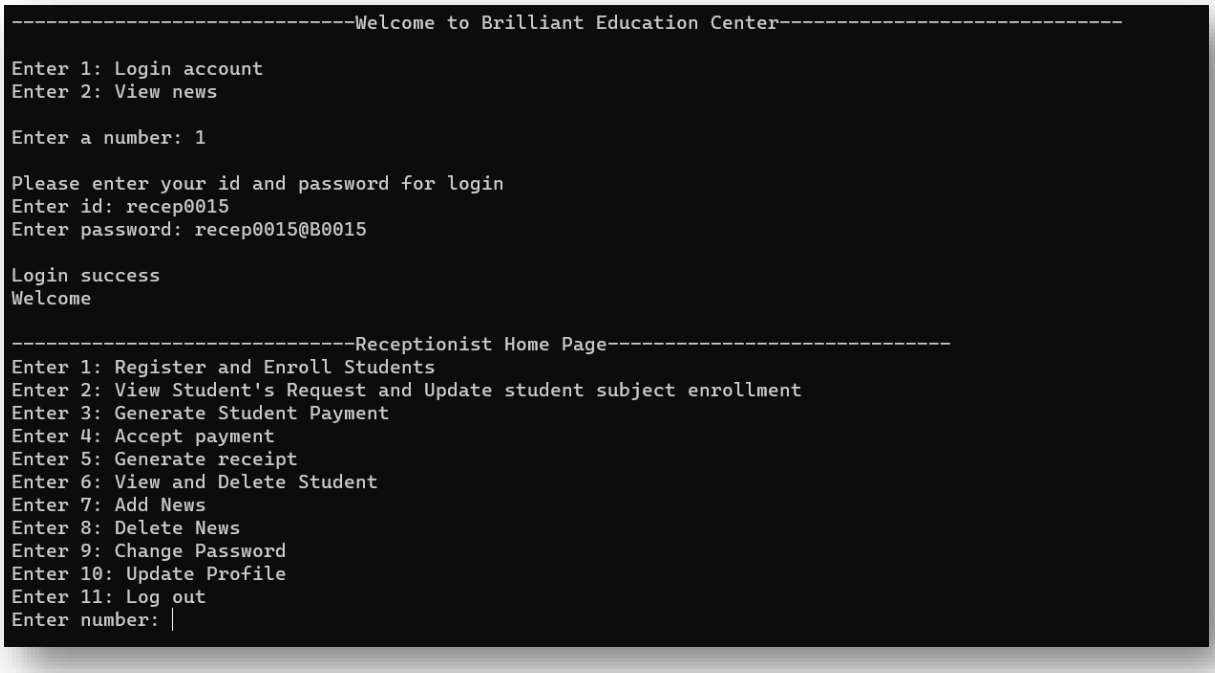

Figure 76 User Login Success

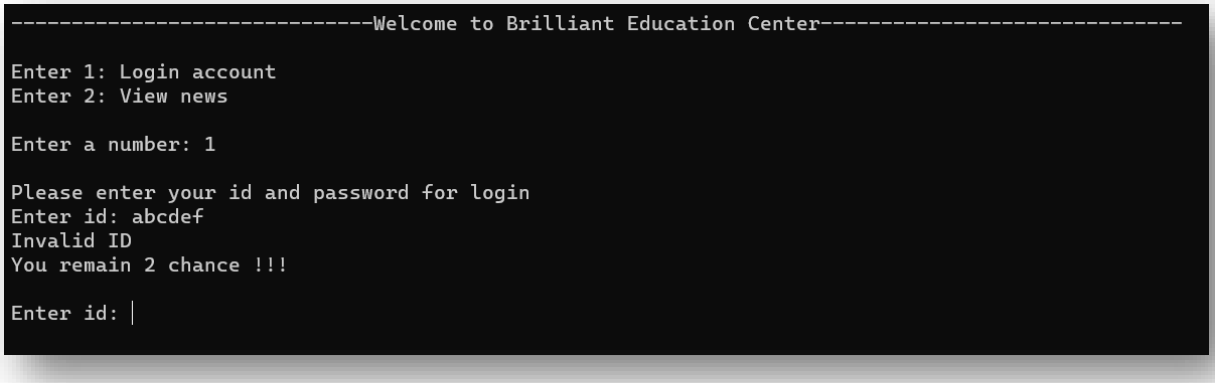

Figure 77 User Enter Wrong ID

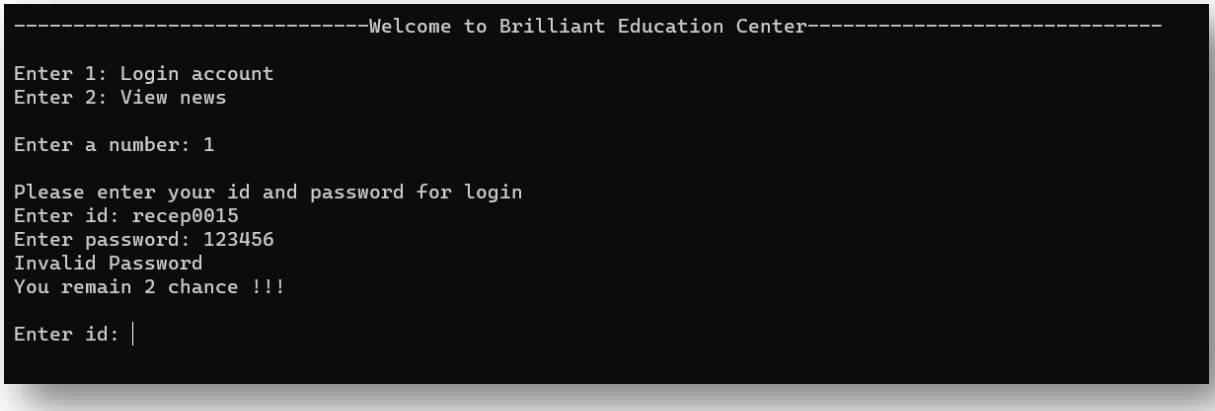

Figure 78 User Enter Wrong Password

6.2 View News Page

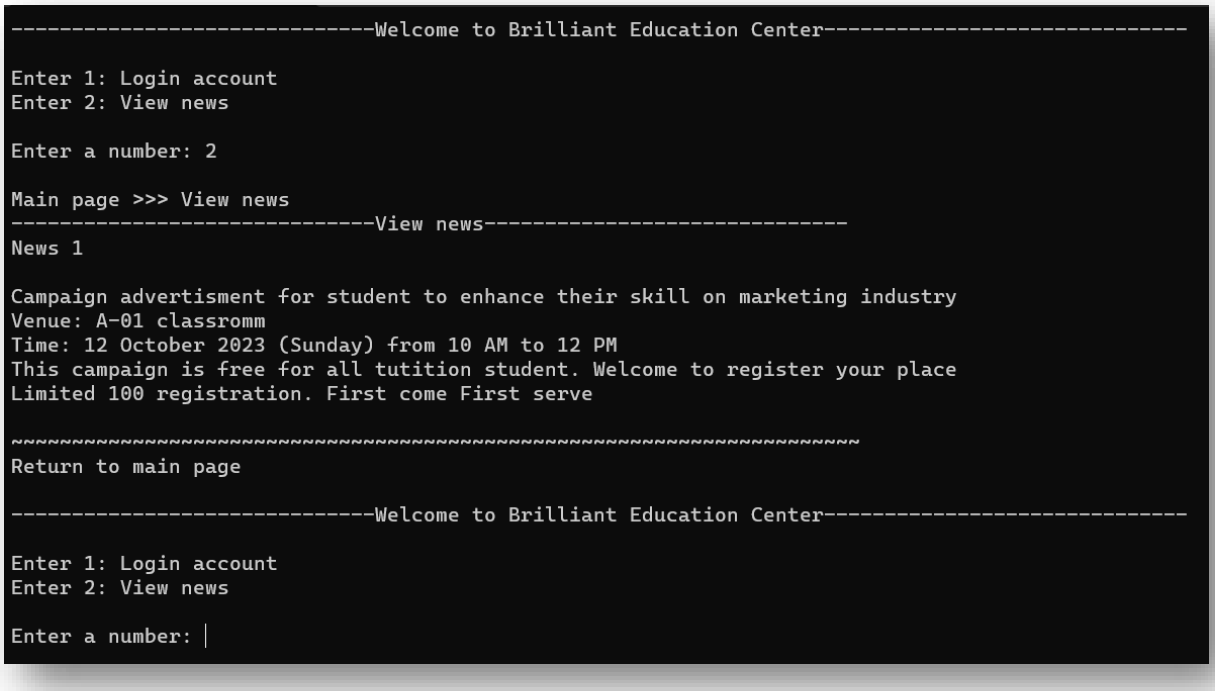

Figure 79 Access to View News Page

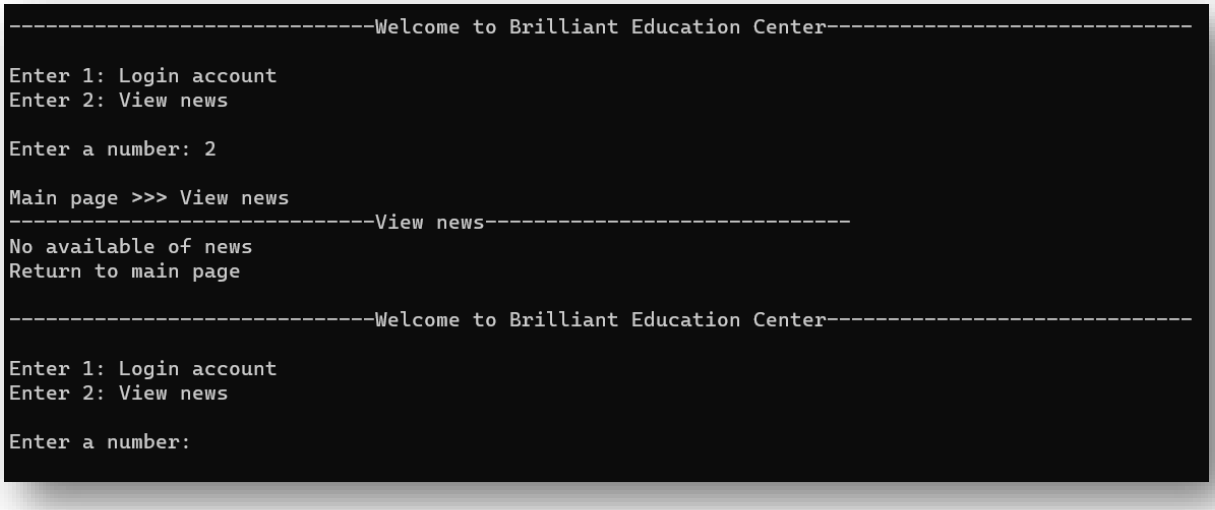

Figure 80 Access to View News Page when Not Available of News

6.3 Admin Functionalities

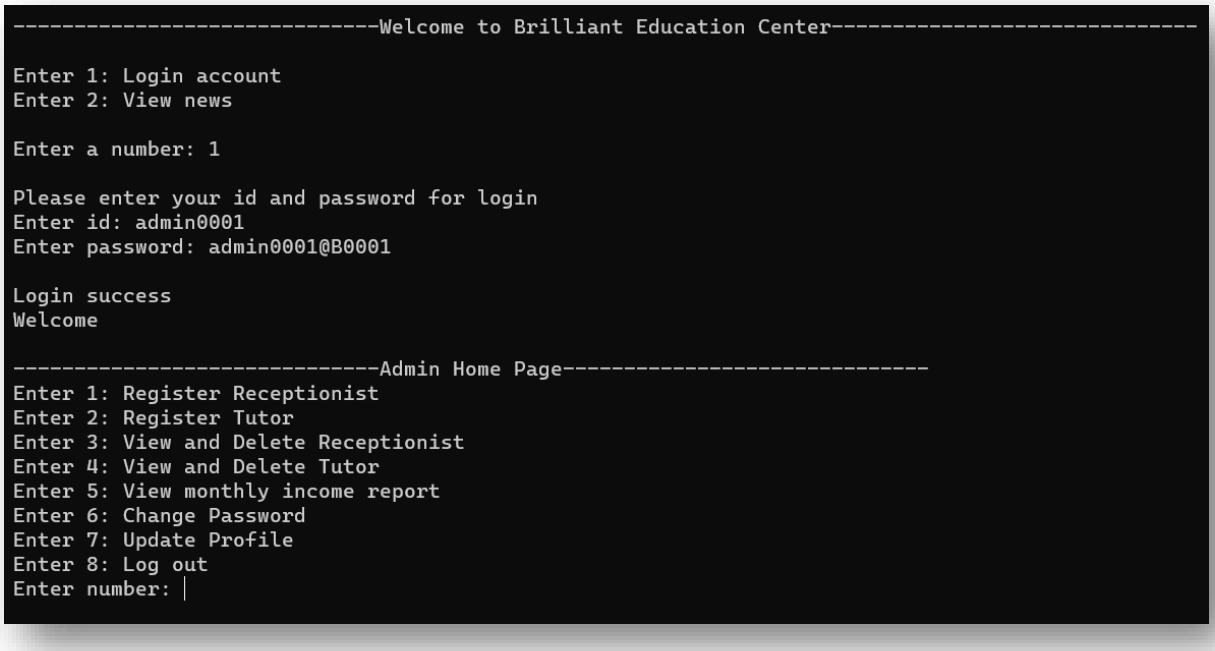

Figure 81 Admin Login Account Successful and Access to Admin Page

6.3.1 Register Receptionist Page

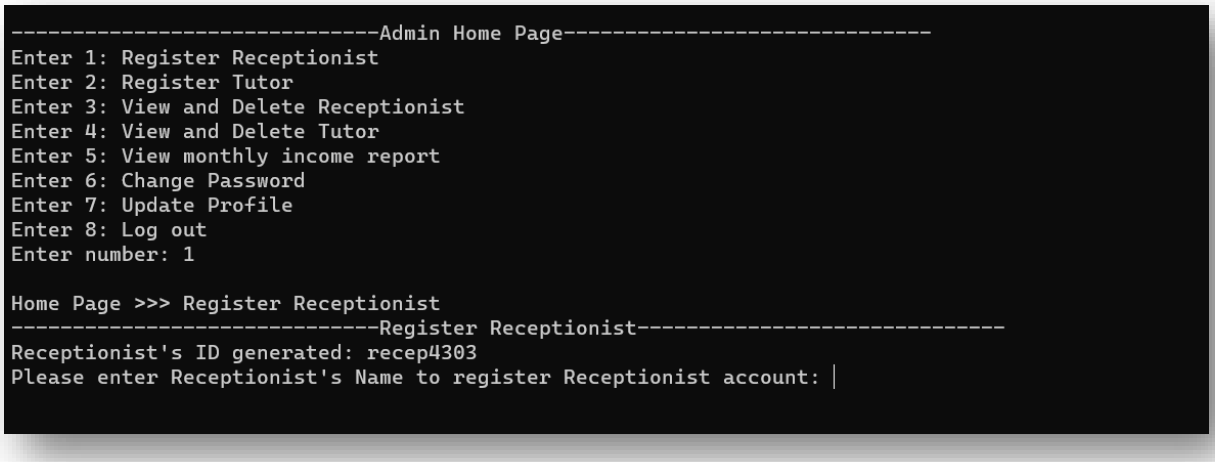

Figure 82 Access to Register Receptionist Page

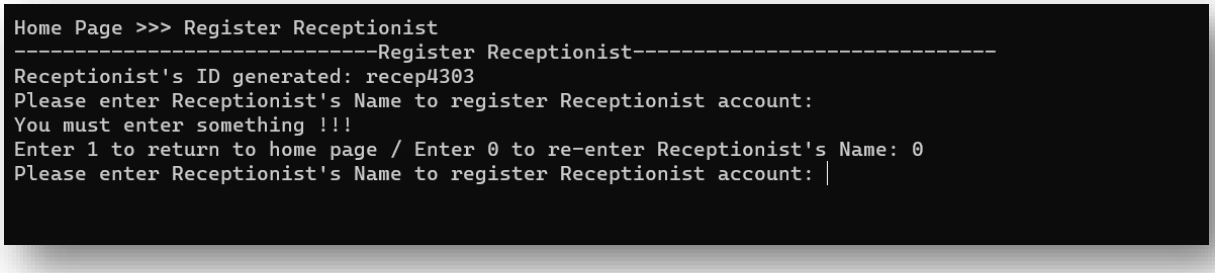

Figure 83 Admin does not enter any word for the Receptionist's Name

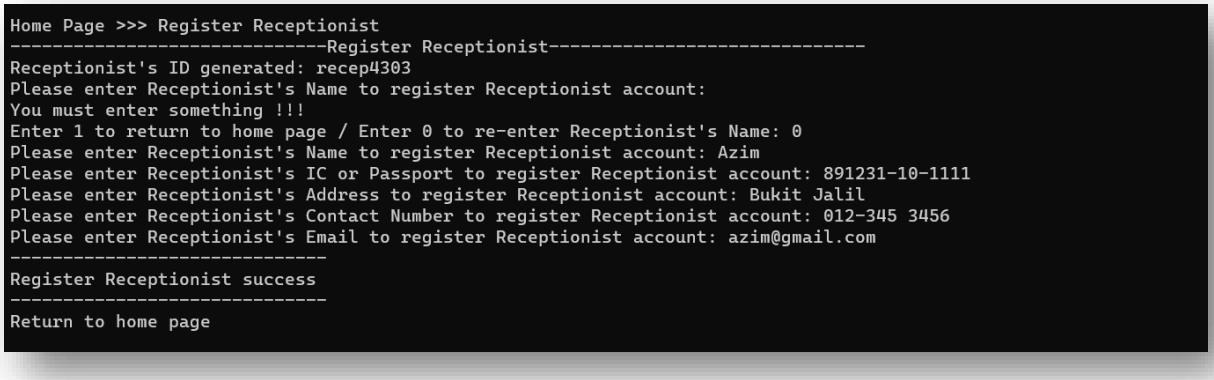

Figure 84 Register Receptionist's Account Successfully

6.3.2 Register Tutor Page

```
------------------------------Admin Home Page------------------------------
Enter 1: Register Receptionist
Enter 2: Register Tutor
Enter 3: View and Delete Receptionist
Enter 4: View and Delete Tutor
Enter 5: View monthly income report
Enter 6: Change Password
Enter 7: Update Profile
Enter 8: Log out
Enter number: 2

Home Page >>> Register Tutor
------------------------------Register Tutor------------------------------
(Levels and Subjects)
FORM1: BAHASA MALAYSIA / ENGLISH / MATHS / SCIENCE / SEJARAH

FORM2: BAHASA MALAYSIA / ENGLISH / MATHS / SCIENCE / SEJARAH

FORM3: BAHASA MALAYSIA / ENGLISH / MATHS / SCIENCE / SEJARAH

FORM4: BAHASA MALAYSIA / ENGLISH / MATHS / SCIENCE / SEJARAH / ADD MATHS / PHYSICS / CHEMISTRY / BIOLOGY

FORM5: BAHASA MALAYSIA / ENGLISH / MATHS / SCIENCE / SEJARAH / ADD MATHS / PHYSICS / CHEMISTRY / BIOLOGY

Tutor's ID generated: tutor5309
Please enter Tutor's Name to register Tutor account:
```

Figure 85 Access to Register Tutor Page

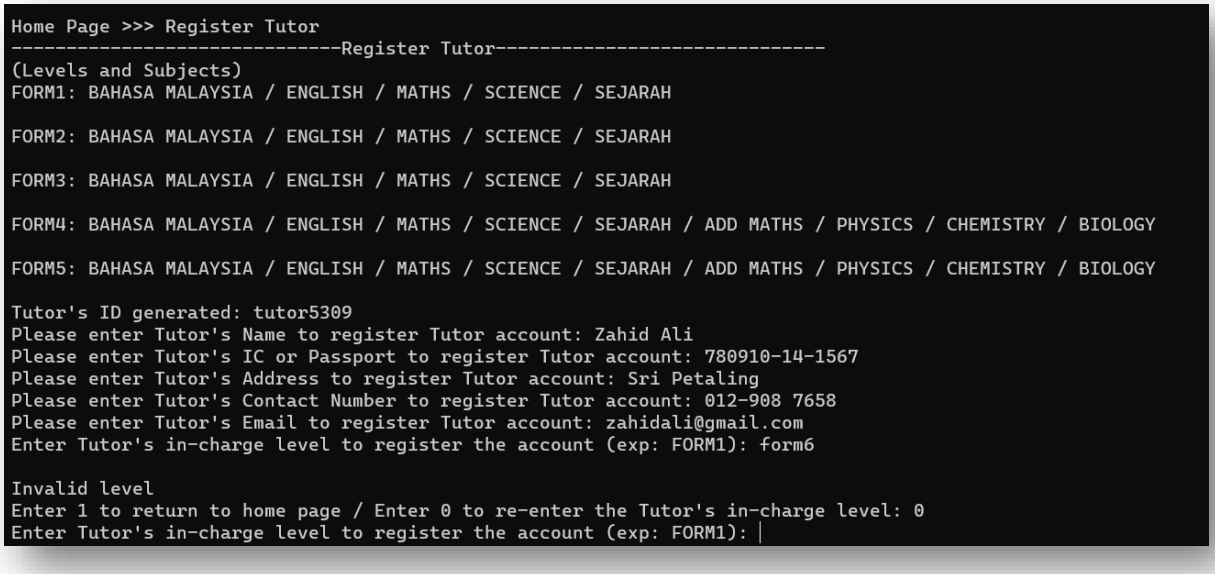

Figure 86 Admin Enter Invalid Level when Registering Tutor

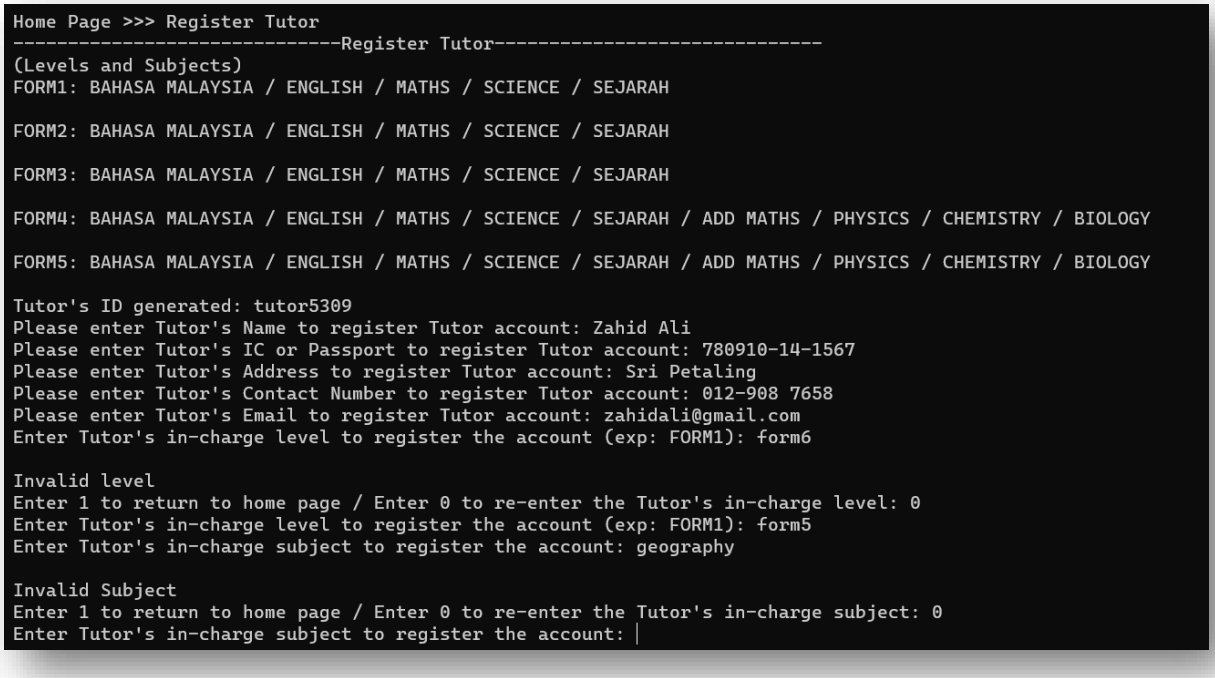

Figure 87 Admin Enter Invalid Subject when Registering Tutor

```
Home Page >>> Register Tutor
-----------------------------Register Tutor-----------------------------
(Levels and Subjects)
FORM1: BAHASA MALAYSIA / ENGLISH / MATHS / SCIENCE / SEJARAH

FORM2: BAHASA MALAYSIA / ENGLISH / MATHS / SCIENCE / SEJARAH

FORM3: BAHASA MALAYSIA / ENGLISH / MATHS / SCIENCE / SEJARAH

FORM4: BAHASA MALAYSIA / ENGLISH / MATHS / SCIENCE / SEJARAH / ADD MATHS / PHYSICS / CHEMISTRY / BIOLOGY

FORM5: BAHASA MALAYSIA / ENGLISH / MATHS / SCIENCE / SEJARAH / ADD MATHS / PHYSICS / CHEMISTRY / BIOLOGY

Tutor's ID generated: tutor5309
Please enter Tutor's Name to register Tutor account: Zahid Ali
Please enter Tutor's IC or Passport to register Tutor account: 780910-14-1567
Please enter Tutor's Address to register Tutor account: Sri Petaling
Please enter Tutor's Contact Number to register Tutor account: 012-908 7658
Please enter Tutor's Email to register Tutor account: zahidali@gmail.com
Enter Tutor's in-charge level to register the account (exp: FORM1): form6

Invalid level
Enter 1 to return to home page / Enter 0 to re-enter the Tutor's in-charge level: 0
Enter Tutor's in-charge level to register the account (exp: FORM1): form5
Enter Tutor's in-charge subject to register the account: geography

Invalid Subject
Enter 1 to return to home page / Enter 0 to re-enter the Tutor's in-charge subject: 0
Enter Tutor's in-charge subject to register the account: sejarah
------------------------------
Register Tutor Success
------------------------------
Return to home page
```

Figure 88 Register Tutor's Account Successfully

6.3.3 View and Delete Receptionist Page

```
-----------------------------Admin Home Page-----------------------------
Enter 1: Register Receptionist
Enter 2: Register Tutor
Enter 3: View and Delete Receptionist
Enter 4: View and Delete Tutor
Enter 5: View monthly income report
Enter 6: Change Password
Enter 7: Update Profile
Enter 8: Log out
Enter number: 3

Home Page >>> Delete Receptionist
-----------------------------Delete Receptionist-----------------------------
Receptionist ID: recep0015
Receptionist Name: Muhammad Ali
Receptionist IC or Passport: 123456-78-9999
Receptionist Address: Puchong
Receptionist Contact Number: 012-345 6789
Receptionist Email: muhammad@gmail.com
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Receptionist ID: recep4303
Receptionist Name: Azim
Receptionist IC or Passport: 891231-10-1111
Receptionist Address: Bukit Jalil
Receptionist Contact Number: 012-345 3456
Receptionist Email: azim@gmail.com
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Receptionist ID: recep4503
Receptionist Name: TianWei
Receptionist IC or Passport: 222222-22-2222
Receptionist Address: Bukit Jalil
Receptionist Contact Number: 012-222 2222
Receptionist Email: tianwei@gmail.com
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter 1 to return to home page / Enter 0 to delete receptionist's information:
```

Figure 89 Access to View and Delete Receptionist's Account Page

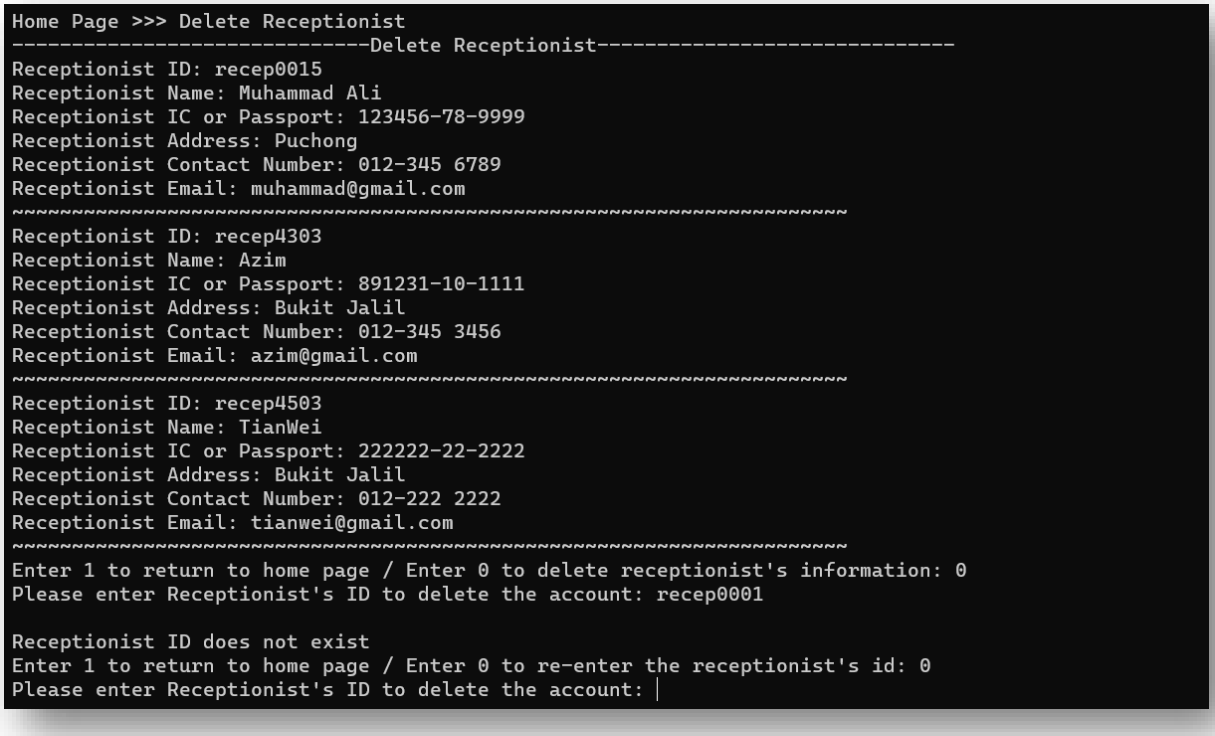

Figure 90 Admin enters a not existed Receptionist's ID

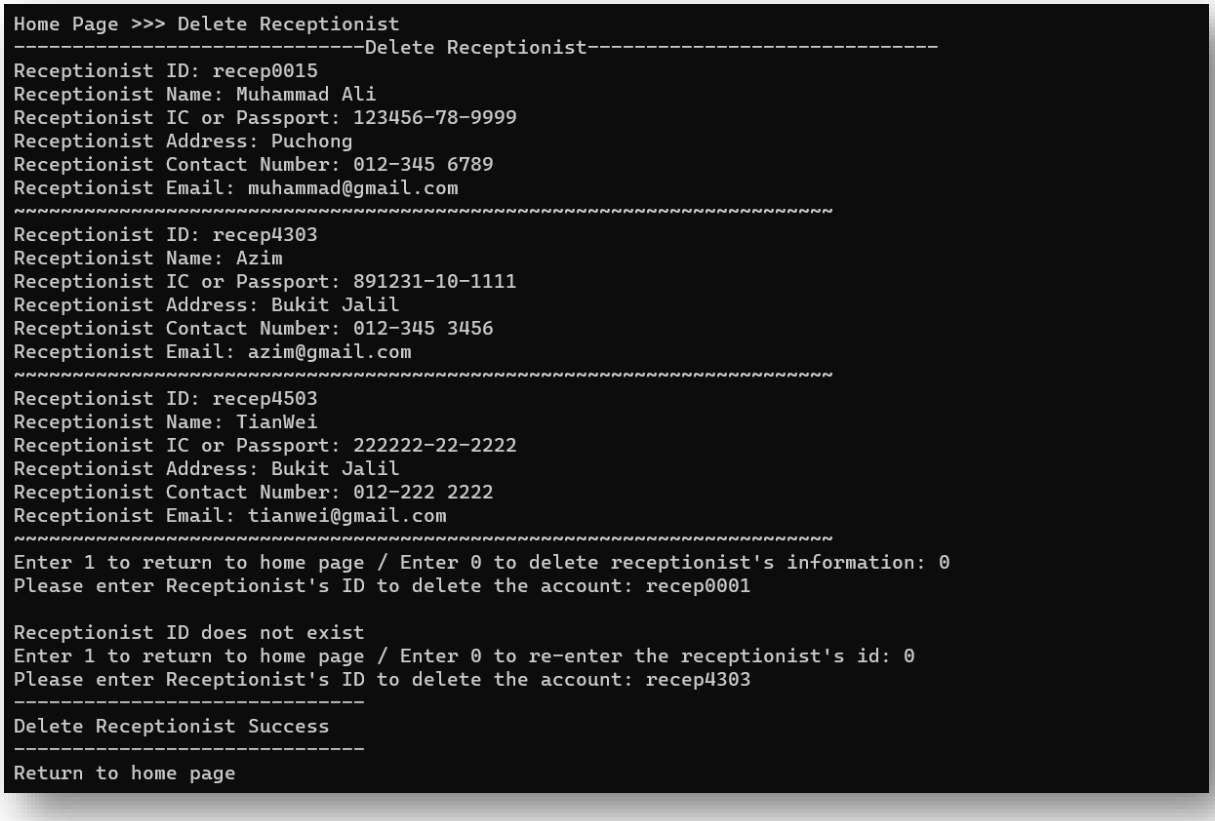

Figure 91 Delete Receptionist's Account Successful

6.3.4 View and Delete Tutor Page

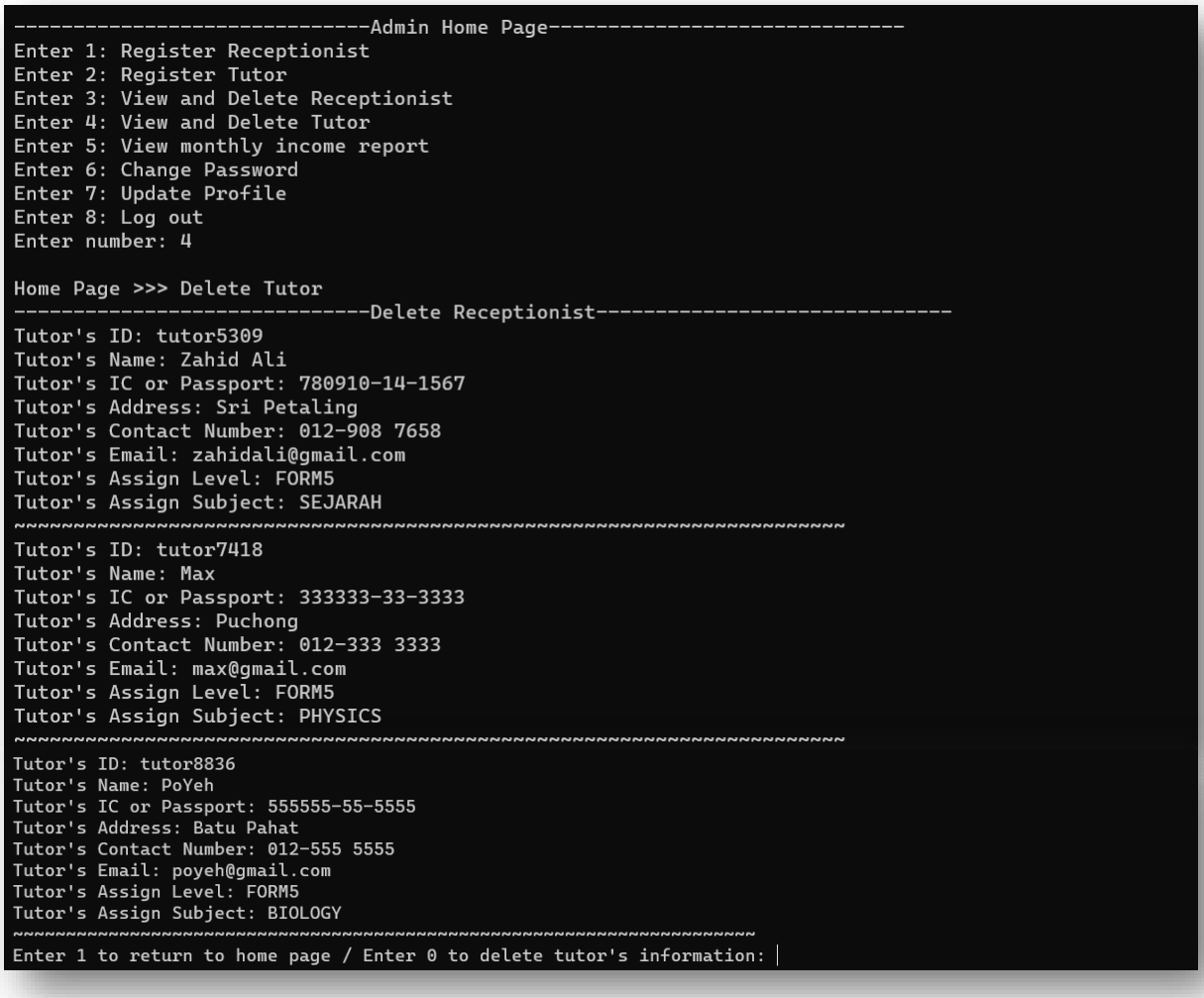

Figure 92 Access to View and Delete Tutor's Account

```
Home Page >>> Delete Tutor
------------------------------Delete Receptionist------------------------------
Tutor's ID: tutor5309
Tutor's Name: Zahid Ali
Tutor's IC or Passport: 780910-14-1567
Tutor's Address: Sri Petaling
Tutor's Contact Number: 012-908 7658
Tutor's Email: zahidali@gmail.com
Tutor's Assign Level: FORM5
Tutor's Assign Subject: SEJARAH
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Tutor's ID: tutor7418
Tutor's Name: Max
Tutor's IC or Passport: 333333-33-3333
Tutor's Address: Puchong
Tutor's Contact Number: 012-333 3333
Tutor's Email: max@gmail.com
Tutor's Assign Level: FORM5
Tutor's Assign Subject: PHYSICS
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Tutor's ID: tutor8836
Tutor's Name: PoYeh
Tutor's IC or Passport: 555555-55-5555
Tutor's Address: Batu Pahat
Tutor's Contact Number: 012-555 5555
Tutor's Email: poyeh@gmail.com
Tutor's Assign Level: FORM5
Tutor's Assign Subject: BIOLOGY
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter 1 to return to home page / Enter 0 to delete tutor's information: 0
Please enter Tutor ID to delete the account: tutor7418
------------------------------
Delete Tutor Success
------------------------------
Return to home page
```

Figure 93 Delete Tutor's Account Success

6.3.5 View Monthly Income Report Page

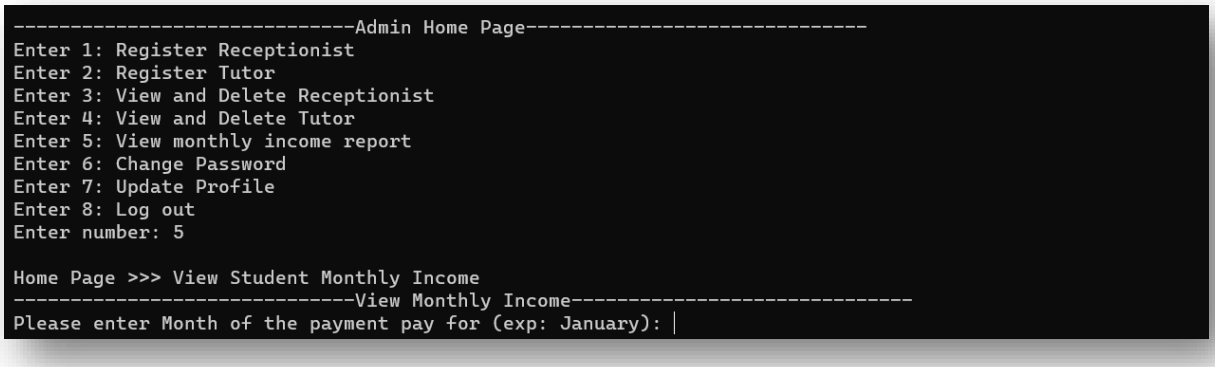

Figure 94 Access to View Monthly Income Report Page

```
Home Page >>> View Student Monthly Income
-----------------------------View Monthly Income-----------------------------
Please enter Month of the payment pay for (exp: January): august
Please enter Year of the payment pay for (exp: 2023): 2023

Date: August 2023

Level: FORM1
        Subject1: BAHASA MALAYSIA
        Total income: RM0

        Subject2: ENGLISH
        Total income: RM0

        Subject3: MATHS
        Total income: RM0

        Subject4: SCIENCE
        Total income: RM0

        Subject5: SEJARAH
        Total income: RM0

Total income of FORM1: 0

Level: FORM2
        Subject1: BAHASA MALAYSIA
        Total income: RM0

        Subject2: ENGLISH
        Total income: RM0

        Subject3: MATHS
        Total income: RM0

        Subject4: SCIENCE
        Total income: RM0

        Subject5: SEJARAH
        Total income: RM0

Total income of FORM2: 0

Level: FORM3
        Subject1: BAHASA MALAYSIA
        Total income: RM0

        Subject2: ENGLISH
        Total income: RM0

        Subject3: MATHS
        Total income: RM0

        Subject4: SCIENCE
        Total income: RM0

        Subject5: SEJARAH
        Total income: RM0

Total income of FORM3: 0
```

```
Level: FORM4
        Subject1: BAHASA MALAYSIA
        Total income: RM0

        Subject2: ENGLISH
        Total income: RM0

        Subject3: MATHS
        Total income: RM0

        Subject4: SCIENCE
        Total income: RM0

        Subject5: SEJARAH
        Total income: RM0

        Subject6: ADD MATHS
        Total income: RM0

        Subject7: PHYSICS
        Total income: RM0

        Subject8: CHEMISTRY
        Total income: RM0

        Subject9: BIOLOGY
        Total income: RM0

Total income of FORM4: 0

Level: FORM5
         Subject1: BAHASA MALAYSIA
         Total income: RM0

         Subject2: ENGLISH
         Total income: RM0

         Subject3: MATHS
         Total income: RM0

         Subject4: SCIENCE
         Total income: RM0

         Subject5: SEJARAH
         Total income: RM0

         Subject6: ADD MATHS
         Total income: RM0

         Subject7: PHYSICS
         Total income: RM50

         Subject8: CHEMISTRY
         Total income: RM0

         Subject9: BIOLOGY
         Total income: RM50

Total income of FORM5: 100

------------------------------
Total income of August 2023 is RM 100
------------------------------
Return to home page
```

Figure 95 View Monthly Income of August 2023

6.4 Receptionist Functionalities

```
------------------------------Welcome to Brilliant Education Center------------------------------

Enter 1: Login account
Enter 2: View news

Enter a number: 1

Please enter your id and password for login
Enter id: recep4303
Enter password: recep4303@B4303

Login success
Welcome

-----------------------------Receptionist Home Page-----------------------------
Enter 1: Register and Enroll Students
Enter 2: View Student's Request and Update student subject enrollment
Enter 3: Generate Student Payment
Enter 4: Accept payment
Enter 5: Generate receipt
Enter 6: View and Delete Student
Enter 7: Add News
Enter 8: Delete News
Enter 9: Change Password
Enter 10: Update Profile
Enter 11: Log out
Enter number:
```

Figure 96 Receptionist Login Account Successful and Access to Receptionist Home Page

6.4.1 Register Student and Subject Enrolment Page

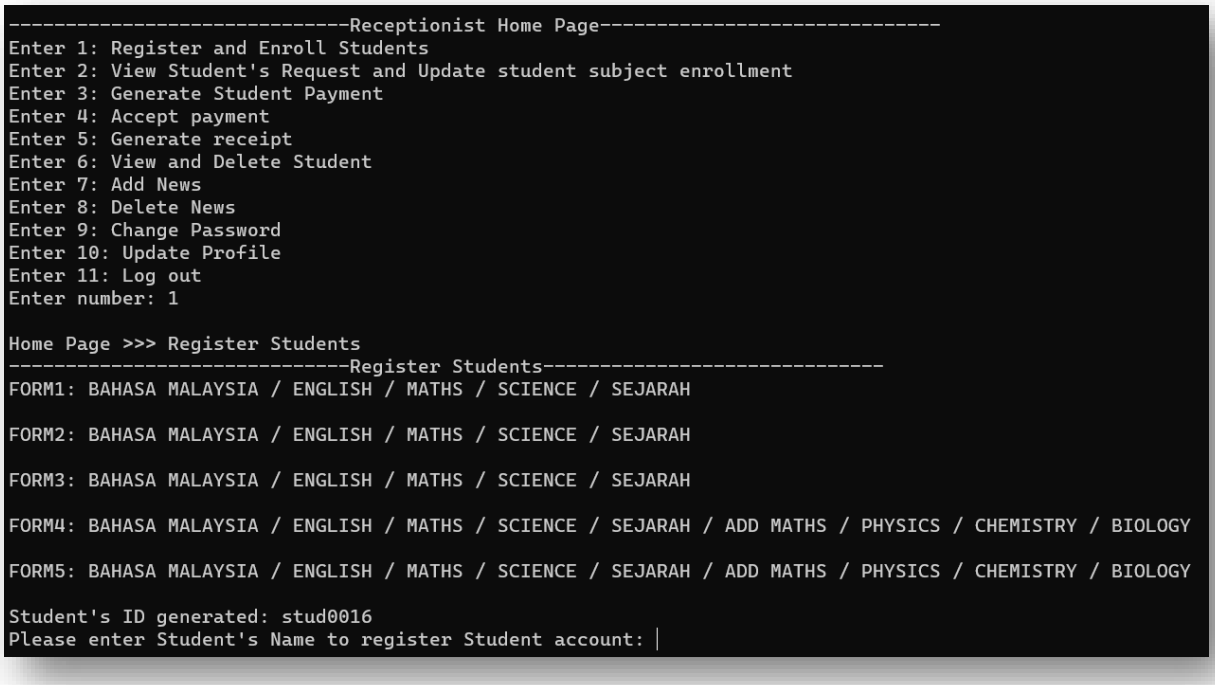

Figure 97 Access to Register Student and Enrol Student's Subjects Page

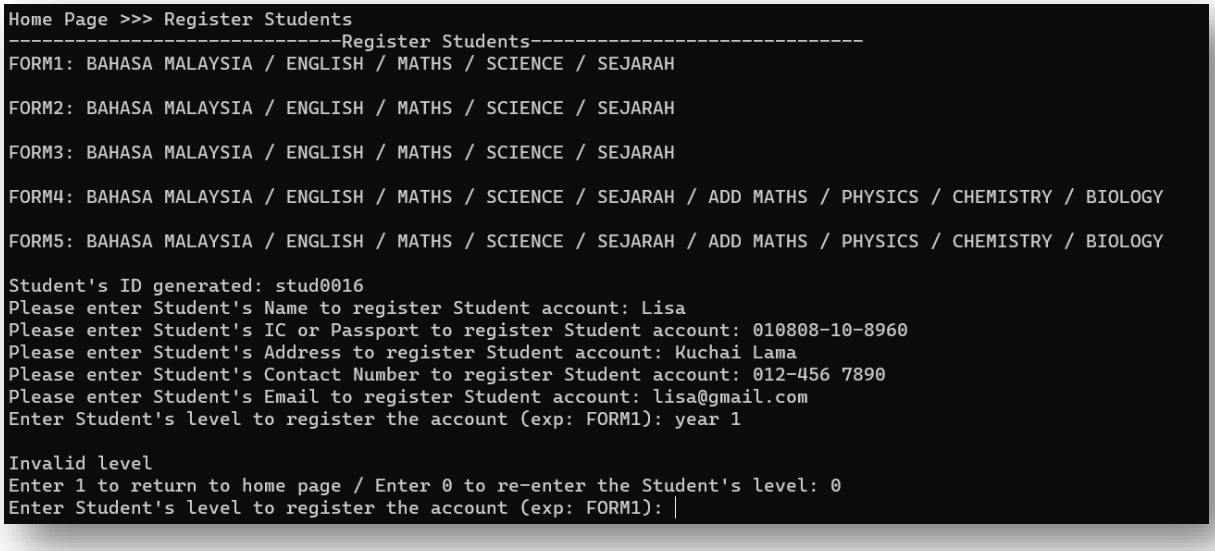

Figure 98 Receptionist Enter Invalid Level when Registering Student's Account

```
Home Page >>> Register Students
------------------------------Register Students------------------------------
FORM1: BAHASA MALAYSIA / ENGLISH / MATHS / SCIENCE / SEJARAH

FORM2: BAHASA MALAYSIA / ENGLISH / MATHS / SCIENCE / SEJARAH

FORM3: BAHASA MALAYSIA / ENGLISH / MATHS / SCIENCE / SEJARAH

FORM4: BAHASA MALAYSIA / ENGLISH / MATHS / SCIENCE / SEJARAH / ADD MATHS / PHYSICS / CHEMISTRY / BIOLOGY

FORM5: BAHASA MALAYSIA / ENGLISH / MATHS / SCIENCE / SEJARAH / ADD MATHS / PHYSICS / CHEMISTRY / BIOLOGY

Student's ID generated: stud0016
Please enter Student's Name to register Student account: Lisa
Please enter Student's IC or Passport to register Student account: 010808-10-8960
Please enter Student's Address to register Student account: Kuchai Lama
Please enter Student's Contact Number to register Student account: 012-456 7890
Please enter Student's Email to register Student account: lisa@gmail.com
Enter Student's level to register the account (exp: FORM1): year 1

Invalid level
Enter 1 to return to home page / Enter 0 to re-enter the Student's level: 0
Enter Student's level to register the account (exp: FORM1): form5
Enter Student Subject 1 to register the account (Enter NULL if no subject enroll): geography

Invalid Subject
Enter 1 to return to home page / Enter 0 to re-enter the Subject 1: 0
Enter Student Subject 1 to register the account (Enter NULL if no subject enroll):
```

Figure 99 Receptionist Enter Invalid Subject when Registering Student's Account

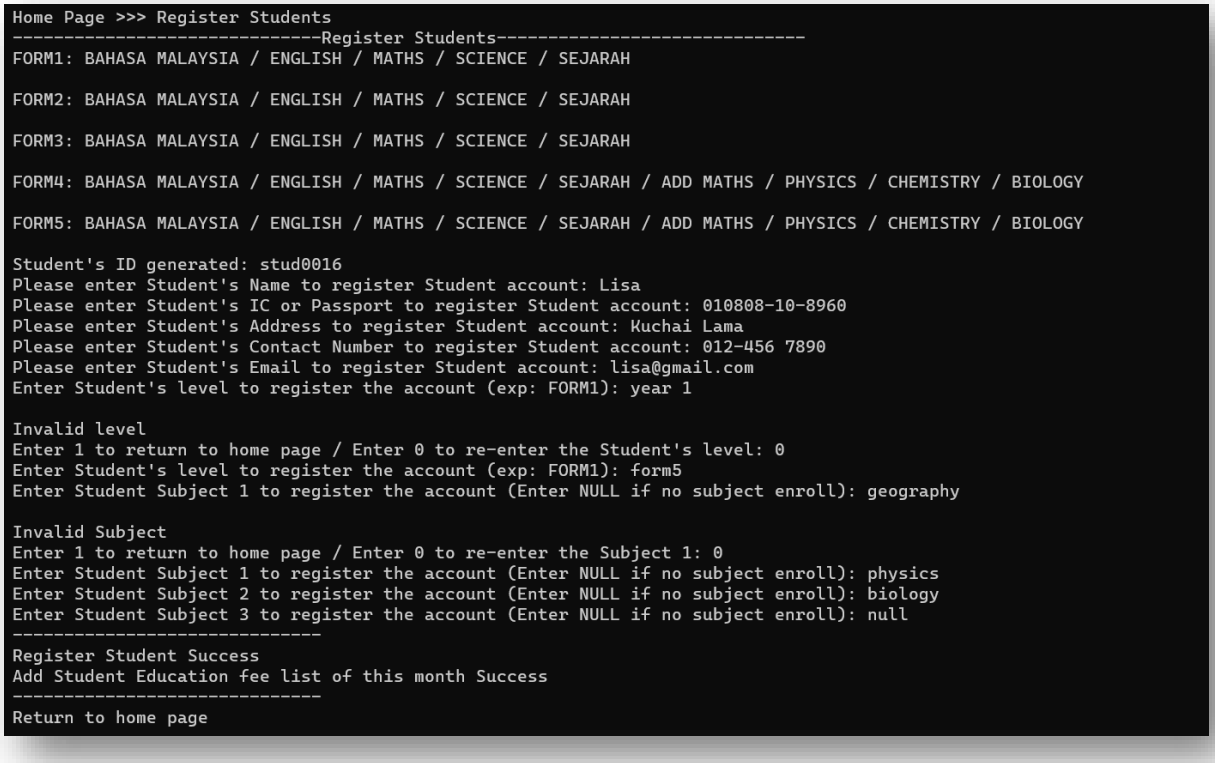

Figure 100 Register Student's Account and Enrol Student's Subject Successful and Generate the Student's Payment information for the month automatically after registering

6.4.2 View Student's Request and Update Student's Subject Enrolment Page

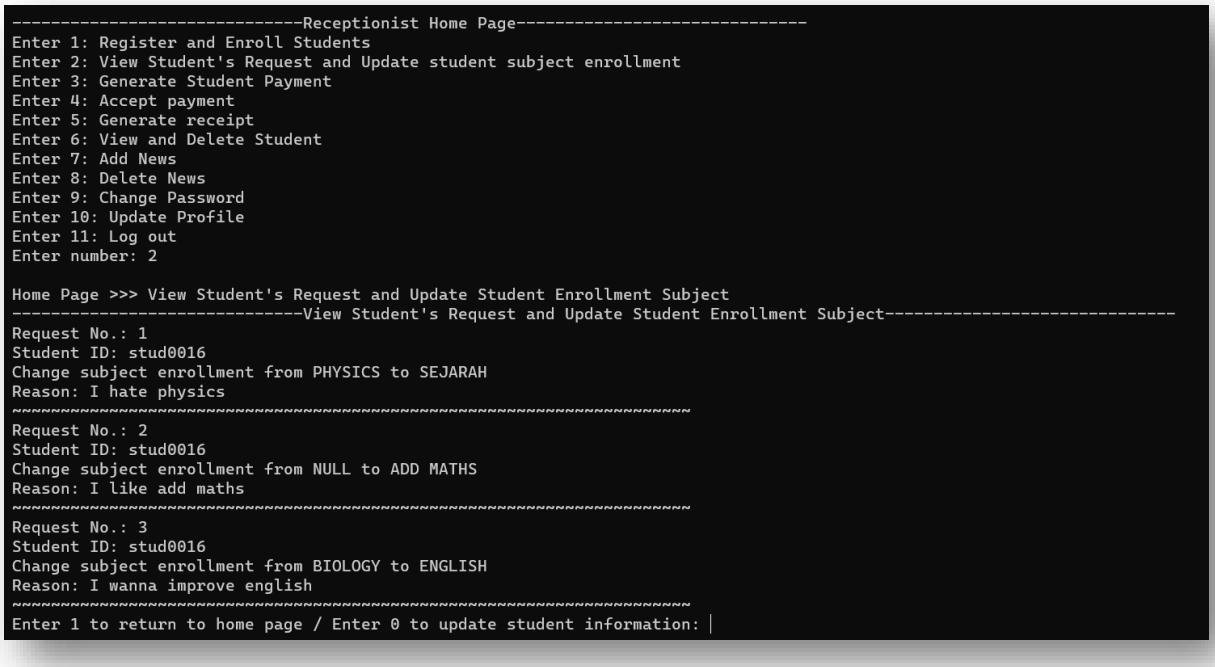

Figure 101 Access to View Student's Request and Update Student's Subject Enrolment Page

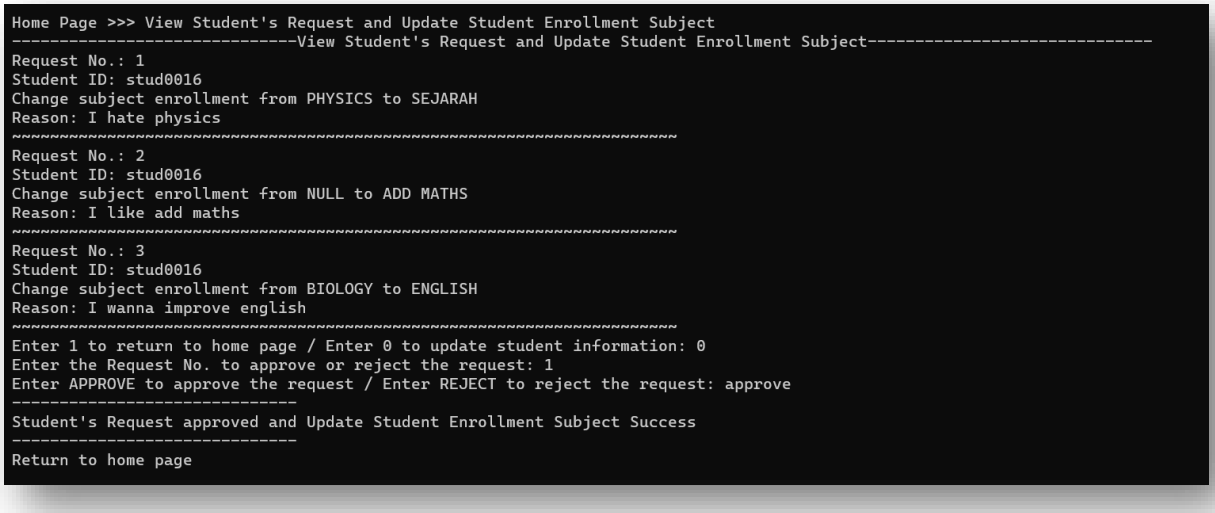

Figure 102 Approve Student's Request and Update the Enrol Subject

```
Home Page >>> View Student's Request and Update Student Enrollment Subject
------------------------------View Student's Request and Update Student Enrollment Subject------------------------------
Request No.: 2
Student ID: stud0016
Change subject enrollment from NULL to ADD MATHS
Reason: I like add maths
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Request No.: 3
Student ID: stud0016
Change subject enrollment from BIOLOGY to ENGLISH
Reason: I wanna improve english
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter 1 to return to home page / Enter 0 to update student information: 0
Enter the Request No. to approve or reject the request: 3
Enter APPROVE to approve the request / Enter REJECT to reject the request: reject
------------------------------
Student's Request rejected
------------------------------
Return to home page
```

Figure 103 Reject Student's Request

6.4.3 Generate Student's Payment Information Page

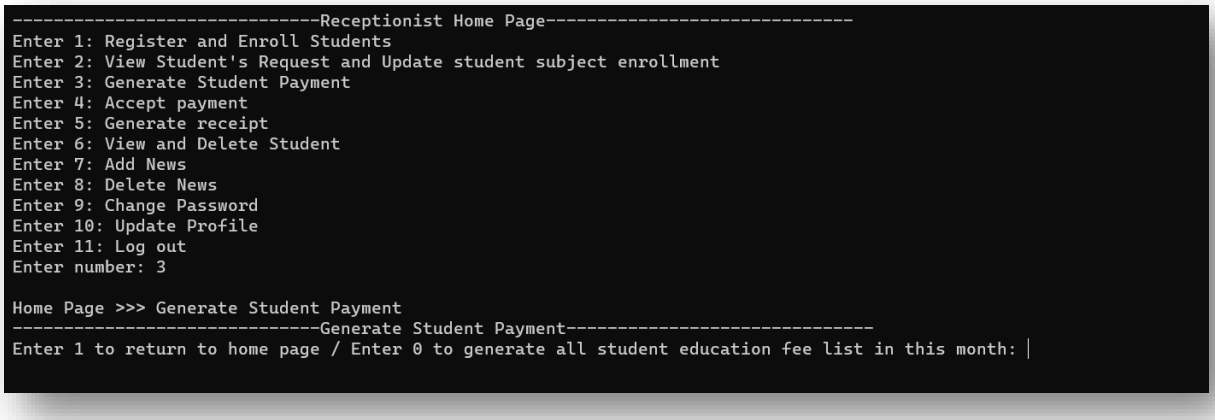

Figure 104 Access to Generate Student's Payment Information Page

```
------------------------------Receptionist Home Page------------------------------
Enter 1: Register and Enroll Students
Enter 2: View Student's Request and Update student subject enrollment
Enter 3: Generate Student Payment
Enter 4: Accept payment
Enter 5: Generate receipt
Enter 6: View and Delete Student
Enter 7: Add News
Enter 8: Delete News
Enter 9: Change Password
Enter 10: Update Profile
Enter 11: Log out
Enter number: 3

Home Page >>> Generate Student Payment
------------------------------Generate Student Payment------------------------------
Enter 1 to return to home page / Enter 0 to generate all student education fee list in this month: 0
------------------------------
Generate Student Payment Success
------------------------------
Return to home page
```

Figure 105 Generate Student's Payment Information for this month successful

6.4.4 Accept Student's Payment Page

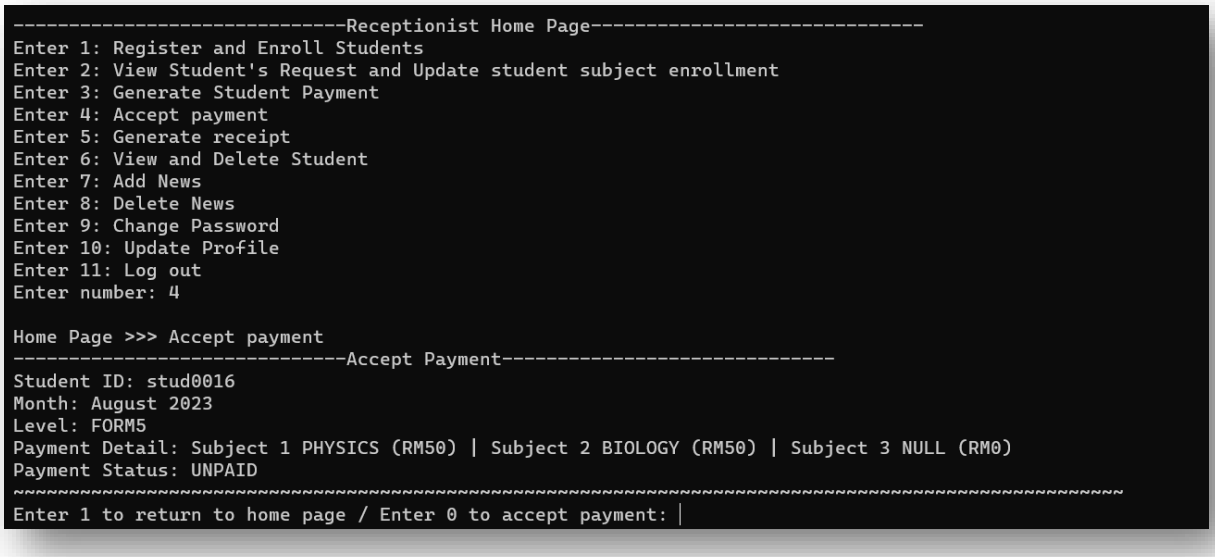

Figure 106 Access to Accept Student's Payment Page

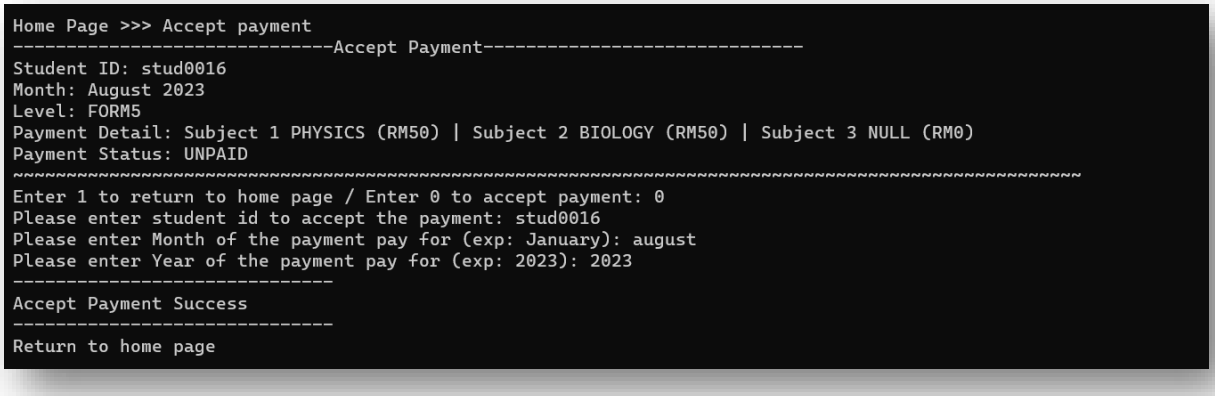

Figure 107 Accept Student's Payment Successful

6.4.5 Generate Receipt Page

```
------------------------------Receptionist Home Page------------------------------
Enter 1: Register and Enroll Students
Enter 2: View Student's Request and Update student subject enrollment
Enter 3: Generate Student Payment
Enter 4: Accept payment
Enter 5: Generate receipt
Enter 6: View and Delete Student
Enter 7: Add News
Enter 8: Delete News
Enter 9: Change Password
Enter 10: Update Profile
Enter 11: Log out
Enter number: 5

Home Page >>> Generate Receipt
------------------------------Generate Receipt------------------------------
Please enter student id to generate the receipt: stud0016
Please enter Month of the payment pay for (exp: January): august
Please enter Year of the payment pay for (exp: 2023): 2023
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Student Receipt~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Student ID: stud0016
Student Name: Lisa
Month of Education fee: August 2023
Payment Date: 25 August 2023
Total Amount: RM100
Accept payment by Receptionist: Azim

Student's Receipt Generated
Return to home page
```

Figure 108 Student's Receipt Generated

6.4.6 View and Delete Student's Account Page

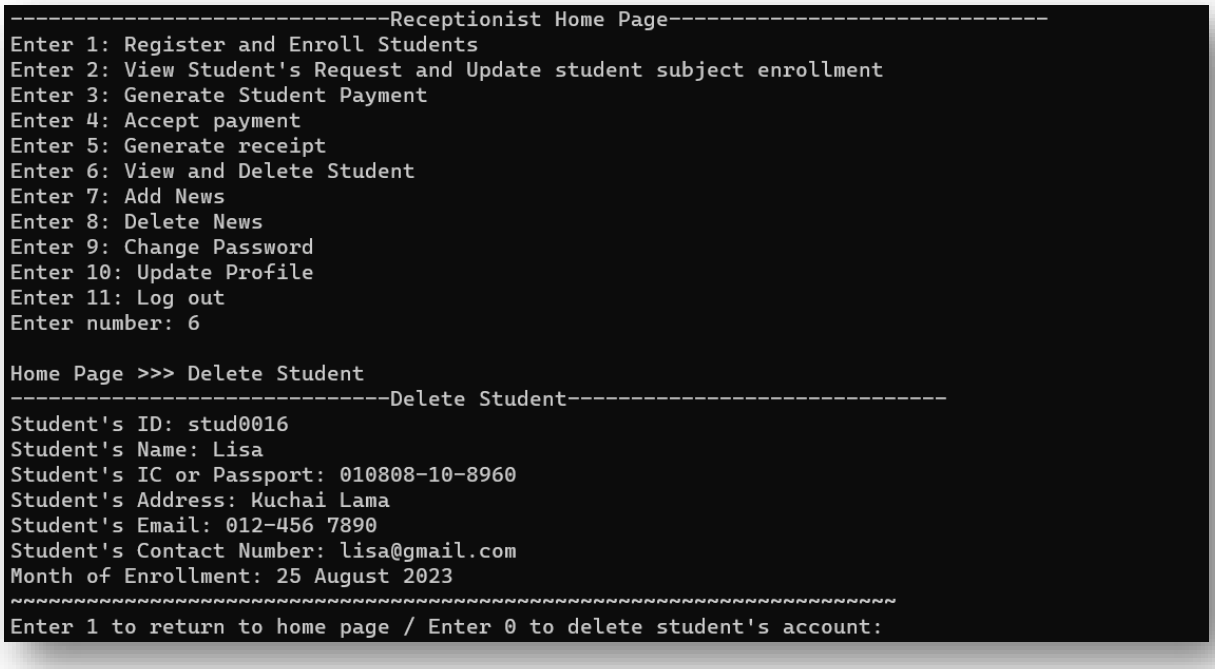

Figure 109 Access to View and Delete Student's Account Page

```
Home Page >>> Delete Student
-----------------------------Delete Student-----------------------------
Student's ID: stud0016
Student's Name: Lisa
Student's IC or Passport: 010808-10-8960
Student's Address: Kuchai Lama
Student's Email: 012-456 7890
Student's Contact Number: lisa@gmail.com
Month of Enrollment: 25 August 2023
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter 1 to return to home page / Enter 0 to delete student's account: 0
Please enter student id to delete the account: stud0016
-----------------------------
Delete Student Information Success
-----------------------------
Return to home page
```

Figure 110 Delete Student's Account Successful

6.4.7 Add News Page

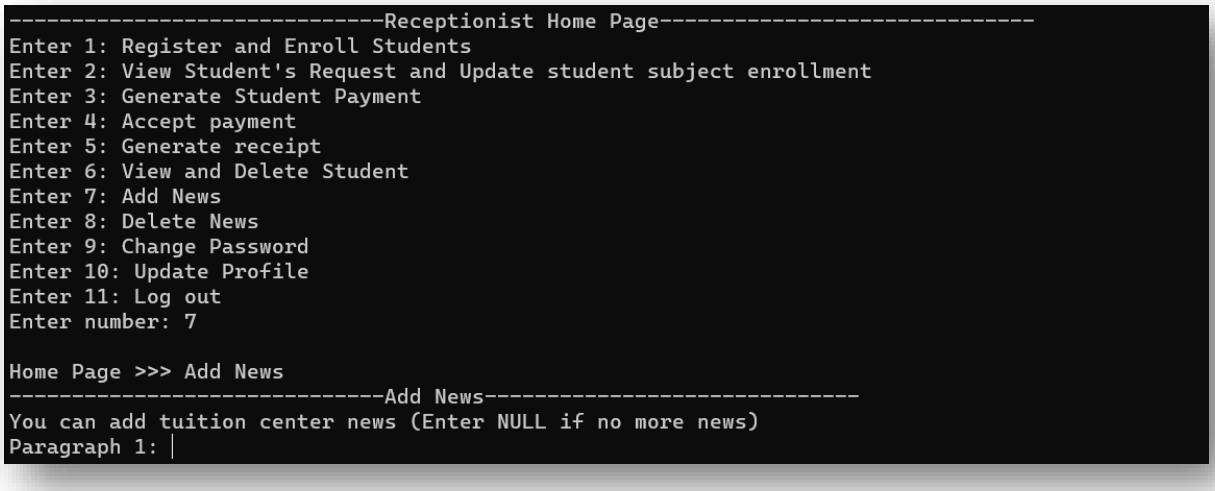

Figure 111 Access to Add News Page

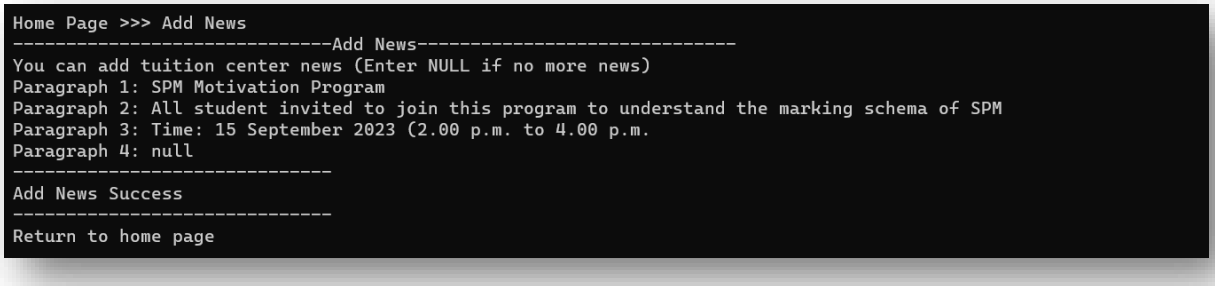

Figure 112 Add News Successful

6.4.8 Delete News Page

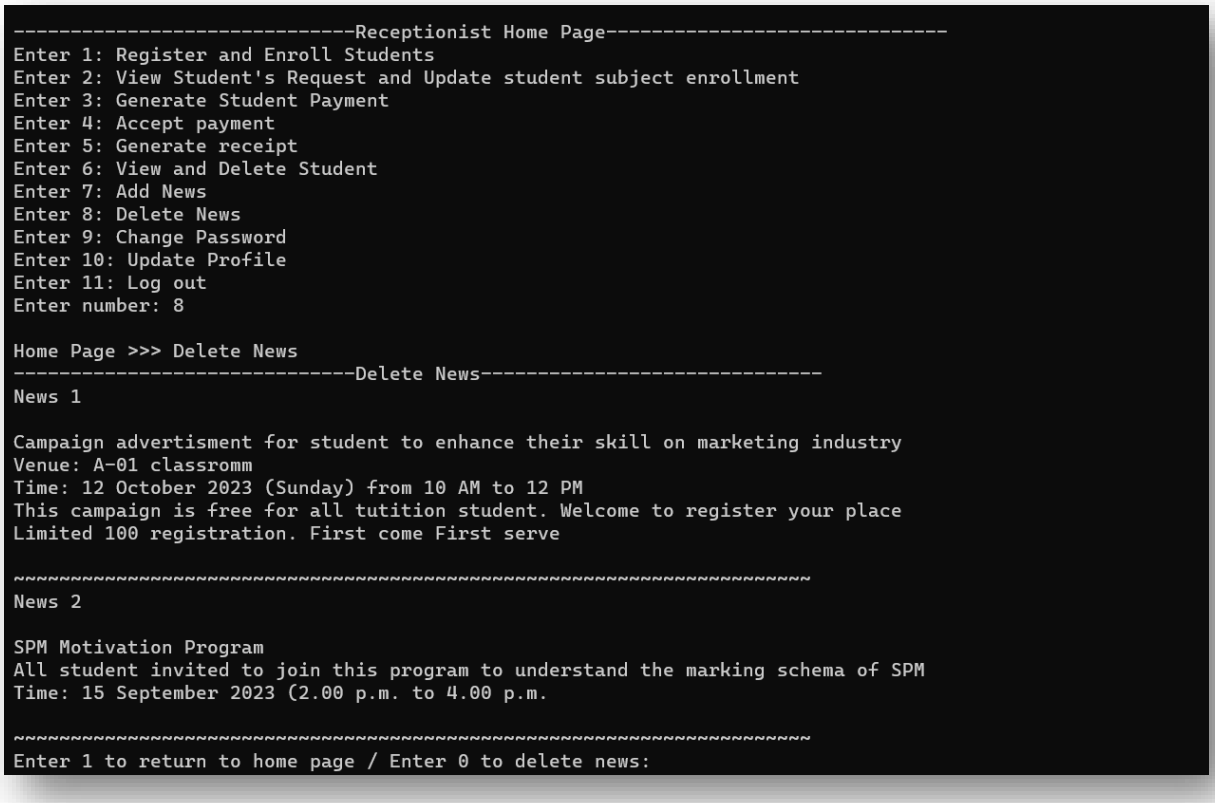

Figure 113 Access to Delete News Page

```
Home Page >>> Delete News
------------------------------Delete News------------------------------
News 1

Campaign advertisment for student to enhance their skill on marketing industry
Venue: A-01 classromm
Time: 12 October 2023 (Sunday) from 10 AM to 12 PM
This campaign is free for all tutition student. Welcome to register your place
Limited 100 registration. First come First serve

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
News 2

SPM Motivation Program
All student inviited to join this program to understand the marking schema of SPM
Time: 15 September 2023 (2.00 p.m. to 4.00 p.m.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter 1 to return to home page / Enter 0 to delete news: 0
Please enter the News No. that you would like to delete: 2
------------------------------
Delete News Success
------------------------------
Return to home page
```

Figure 114 Delete News Successful

6.5 Tutor Functionalities

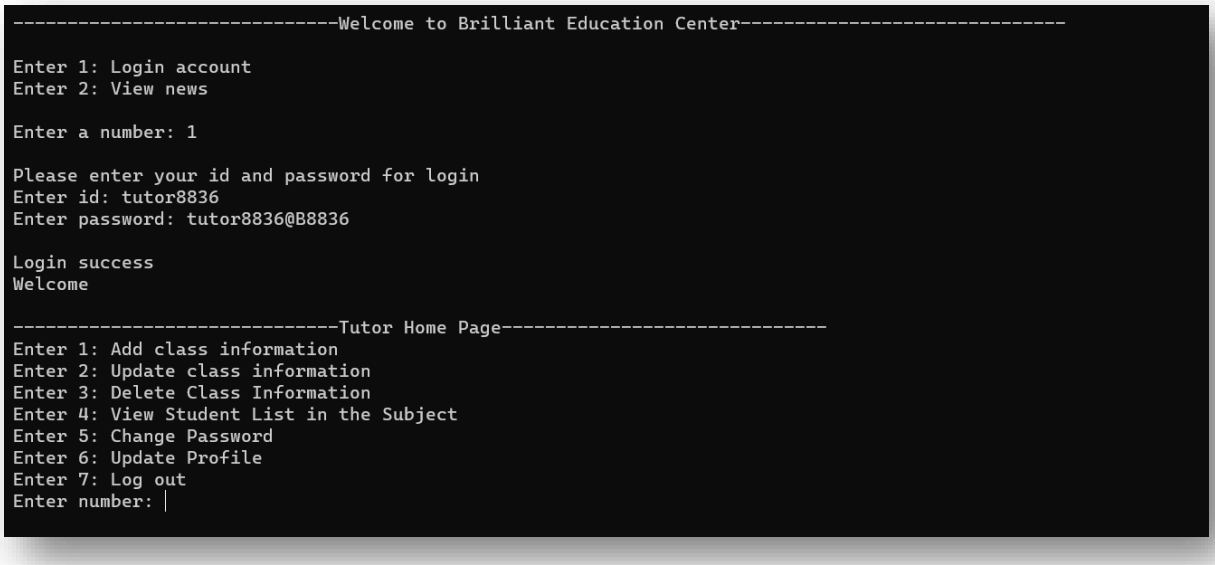

Figure 115 Tutor Login Account Successful and Access to Tutor Home Page

6.5.1 Add Class Schedule Page

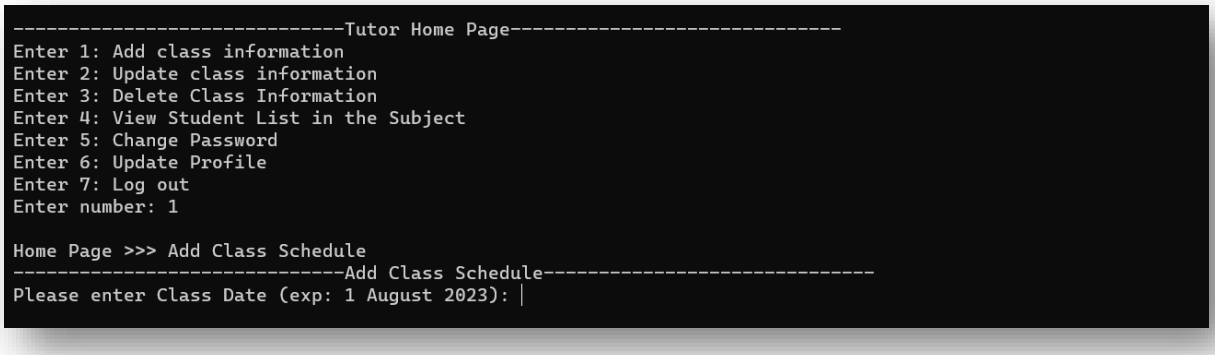

Figure 116 Access to Add Class Schedule Page

```
Home Page >>> Add Class Schedule
------------------------------Add Class Schedule------------------------------
Please enter Class Date (exp: 1 August 2023): 1 September 2023
Please enter Class Venue (exp: A-01): B-03
Please enter Class Start Time (exp: 10.00 AM): 1.00 PM
Please enter Class End Time (exp: 10.00 AM): 3.00 PM
------------------------------
Class information has been saved
------------------------------
Return to home page
```

Figure 117 Add Class Schedule Successful

6.5.2 Update Class Schedule Page

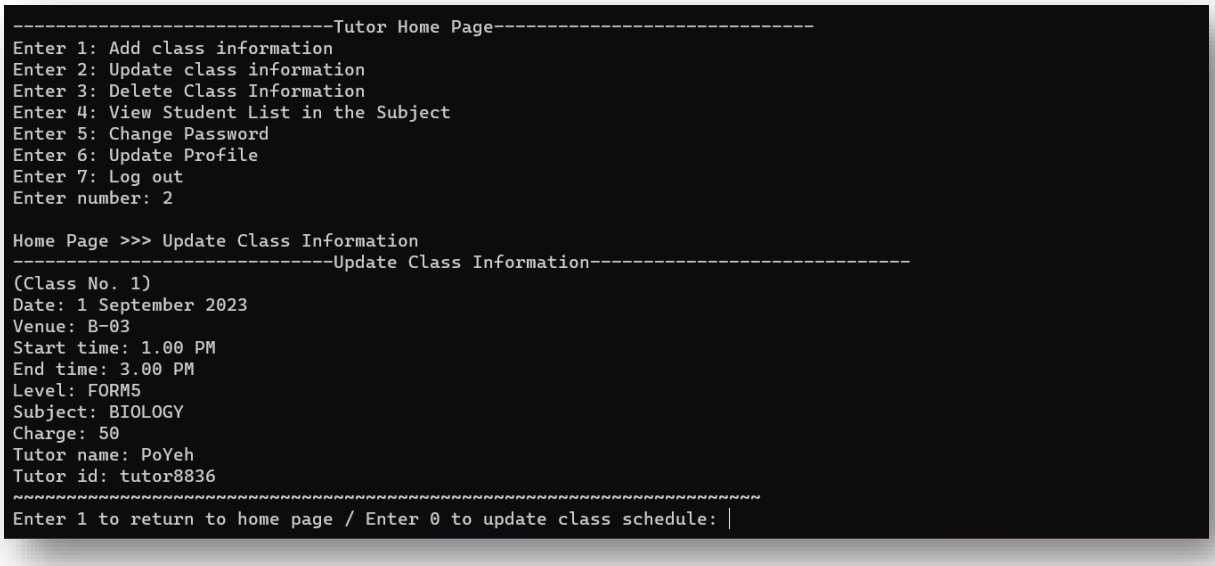

Figure 118 Access to Update Class Schedule Page

```
Home Page >>> Update Class Information
------------------------------Update Class Information------------------------------
(Class No. 1)
Date: 1 September 2023
Venue: B-03
Start time: 1.00 PM
End time: 3.00 PM
Level: FORM5
Subject: BIOLOGY
Charge: 50
Tutor name: PoYeh
Tutor id: tutor8836
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter 1 to return to home page / Enter 0 to update class schedule: 0
Please enter Class No. to update the class information: 1
Please enter one class information to edit the information (exp: Date): date
Please enter Class Date (exp: 1 August 2023): 4 September 2023
------------------------------
Update Class success
------------------------------
Return to home page
```

Figure 119 Update Class Schedule Successful

6.5.3 Delete Class Schedule Page

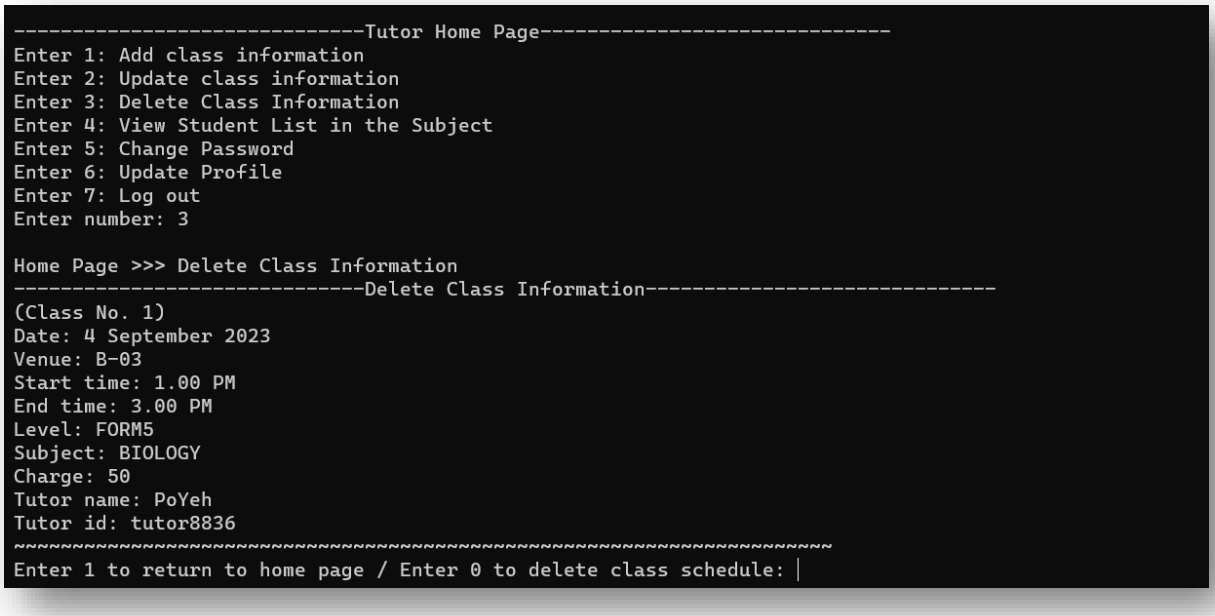

Figure 120 Access to Delete Class Schedule

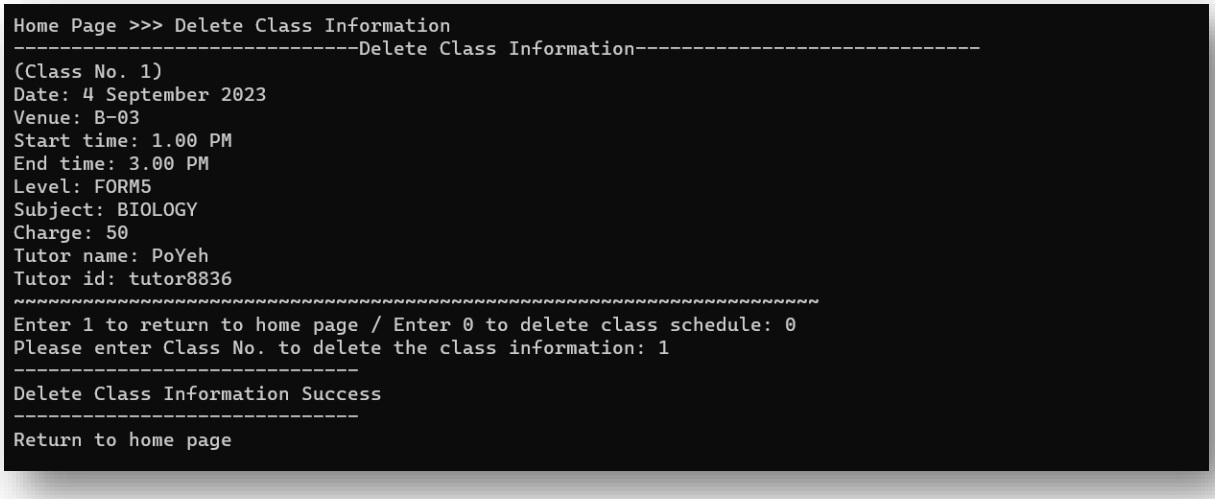

Figure 121 Delete Class Schedule Successful

6.5.4 View Students List in Own Assign Subject Page

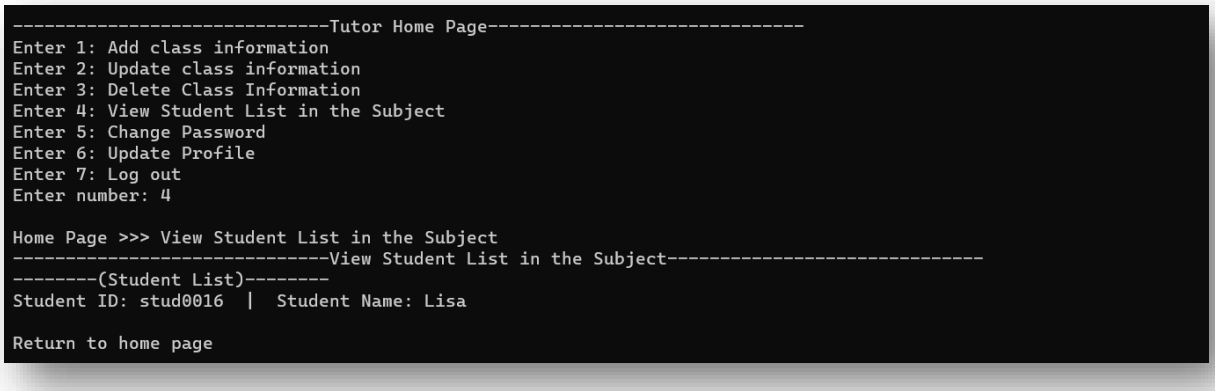

Figure 122 Access to View Student List in Own Assign Subject Page

6.6 Student Functionalities

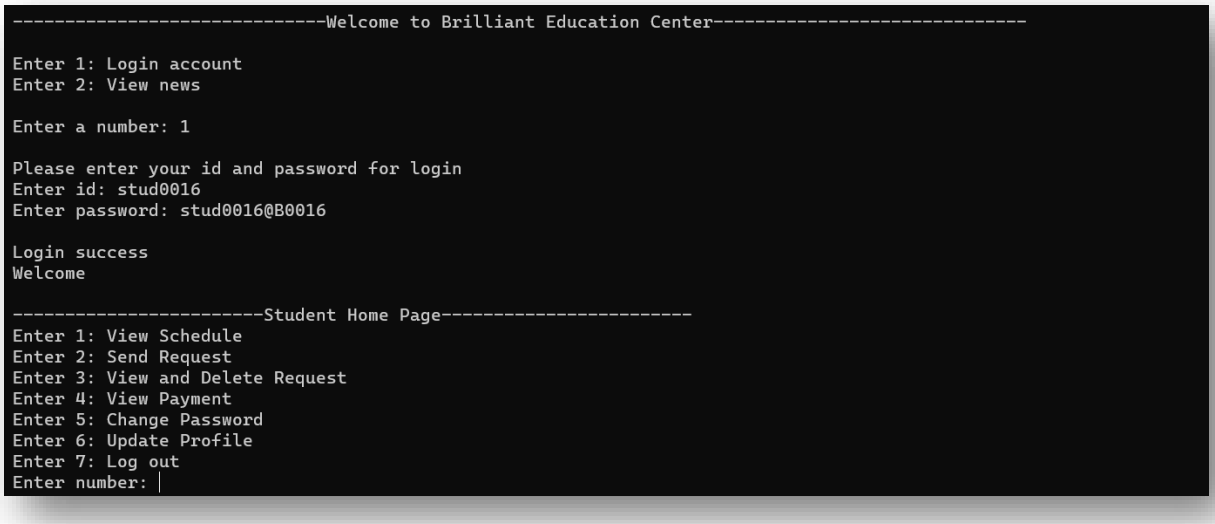

Figure 123 Student Login Account Successful and Access to Student Home Page

6.6.1 View Class Schedule Page

```
------------------------Student Home Page------------------------
Enter 1: View Schedule
Enter 2: Send Request
Enter 3: View and Delete Request
Enter 4: View Payment
Enter 5: Change Password
Enter 6: Update Profile
Enter 7: Log out
Enter number: 1

Home Page >>> View Class Schedule
-----------------------------Class Schedule-------------------------------
Date: 4 September 2023
Venue: B-03
Time: From 1.00 PM to 3.00 PM
Level: FORM5
Subject: BIOLOGY
Charge: 50
Lecturer: PoYeh
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Return to home page
```

Figure 124 Access to View Class Schedule Page

6.6.2 Send Request Page

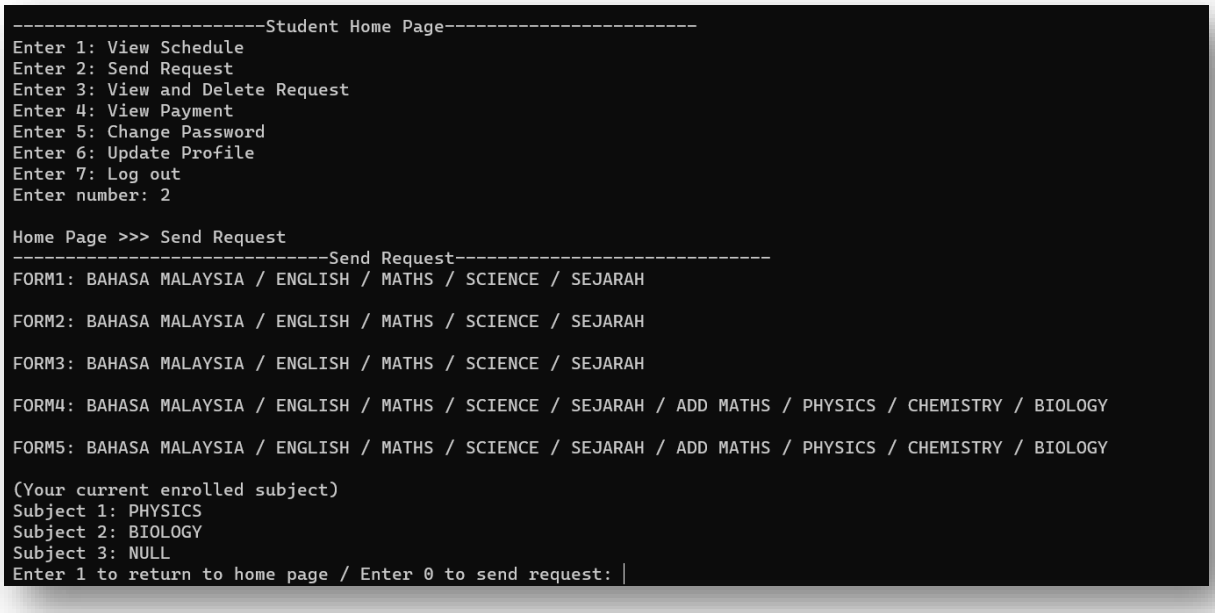

Figure 125 Access to Send Request Page

```
Home Page >>> Send Request
------------------------------Send Request------------------------------
FORM1: BAHASA MALAYSIA / ENGLISH / MATHS / SCIENCE / SEJARAH

FORM2: BAHASA MALAYSIA / ENGLISH / MATHS / SCIENCE / SEJARAH

FORM3: BAHASA MALAYSIA / ENGLISH / MATHS / SCIENCE / SEJARAH

FORM4: BAHASA MALAYSIA / ENGLISH / MATHS / SCIENCE / SEJARAH / ADD MATHS / PHYSICS / CHEMISTRY / BIOLOGY

FORM5: BAHASA MALAYSIA / ENGLISH / MATHS / SCIENCE / SEJARAH / ADD MATHS / PHYSICS / CHEMISTRY / BIOLOGY

(Your current enrolled subject)
Subject 1: PHYSICS
Subject 2: BIOLOGY
Subject 3: NULL
Enter 1 to return to home page / Enter 0 to send request: 0

Enter the subject number that you want to change (exp: 1/2/3): 1
Enter subject name that want to change to: sejarah
Please provide suitable reason for changing subject: I hate physics
------------------------------
Request has sent to receptionist
------------------------------
Return to home page
```

Figure 126 Sending Request Successful

6.6.3 View and Delete Request Page

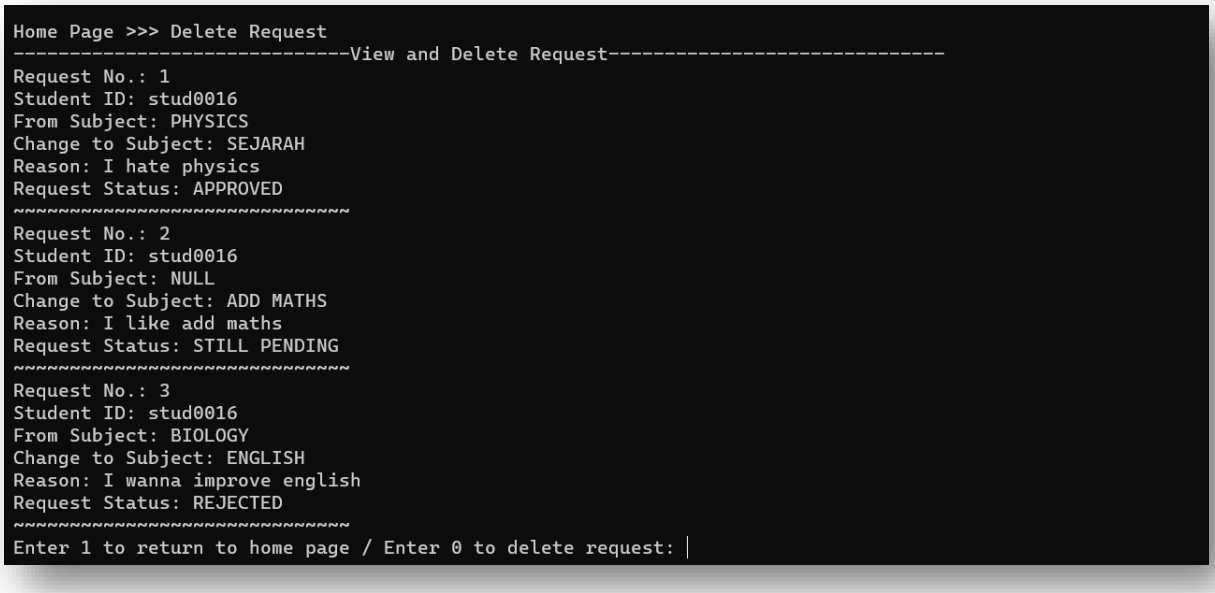

Figure 127 Access to View and Delete Request Page

```
Home Page >>> Delete Request
------------------------------View and Delete Request------------------------------
Request No.: 1
Student ID: stud0016
From Subject: PHYSICS
Change to Subject: SEJARAH
Reason: I hate physics
Request Status: APPROVED
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Request No.: 2
Student ID: stud0016
From Subject: NULL
Change to Subject: ADD MATHS
Reason: I like add maths
Request Status: STILL PENDING
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Request No.: 3
Student ID: stud0016
From Subject: BIOLOGY
Change to Subject: ENGLISH
Reason: I wanna improve english
Request Status: REJECTED
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter 1 to return to home page / Enter 0 to delete request: 0
Please enter Request No. for deletion: 1
You cannot delete this request because it already approved or rejected
Enter 1 to return to home page / Enter 0 to re-enter the request No.:
```

Figure 128 Delete Request Failed Because the Request Already Approved

```
Home Page >>> Delete Request
------------------------------View and Delete Request------------------------------
Request No.: 1
Student ID: stud0016
From Subject: PHYSICS
Change to Subject: SEJARAH
Reason: I hate physics
Request Status: APPROVED
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Request No.: 2
Student ID: stud0016
From Subject: NULL
Change to Subject: ADD MATHS
Reason: I like add maths
Request Status: STILL PENDING
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Request No.: 3
Student ID: stud0016
From Subject: BIOLOGY
Change to Subject: ENGLISH
Reason: I wanna improve english
Request Status: REJECTED
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter 1 to return to home page / Enter 0 to delete request: 0
Please enter Request No. for deletion: 1
You cannot delete this request because it already approved or rejected
Enter 1 to return to home page / Enter 0 to re-enter the request No.: 0
Please enter Request No. for deletion: 2
------------------------------
Delete request success
------------------------------
Return to home page
```

Figure 129 Delete Request Successful

6.6.3 View Payment which in Unpaid Status Page

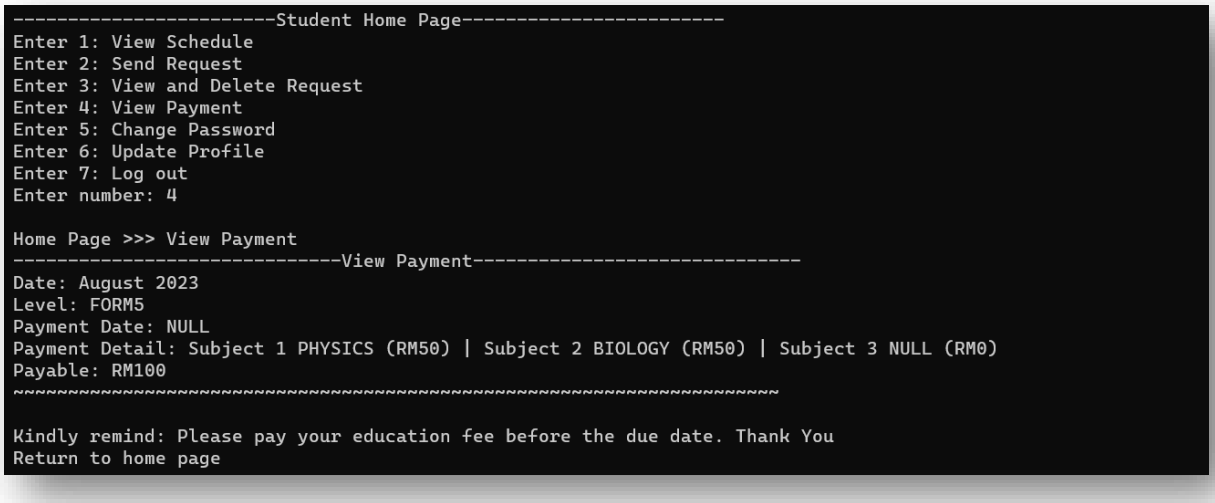

Figure 130 Access to View Payment which in Unpaid Status Page

6.7 User Change Password Page

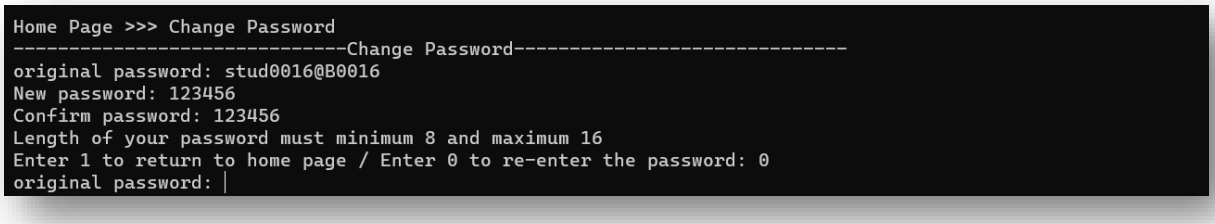

Figure 131 Change password failed because the length new password is not between 8 and 16

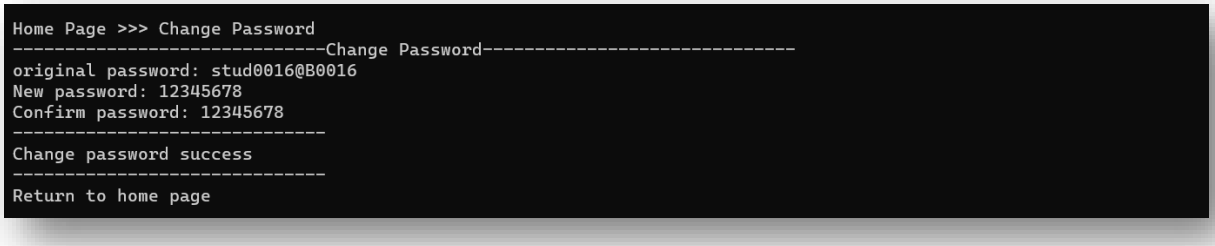

Figure 132 Changing Password Successful

6.8 User Update Own Profile Page

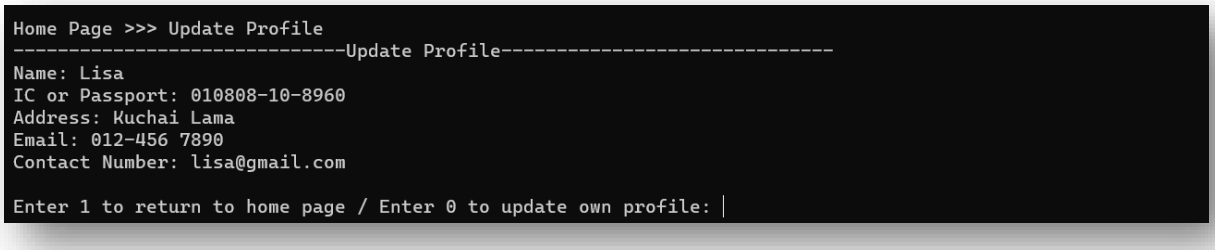

Figure 133 Access to Update Profile Page

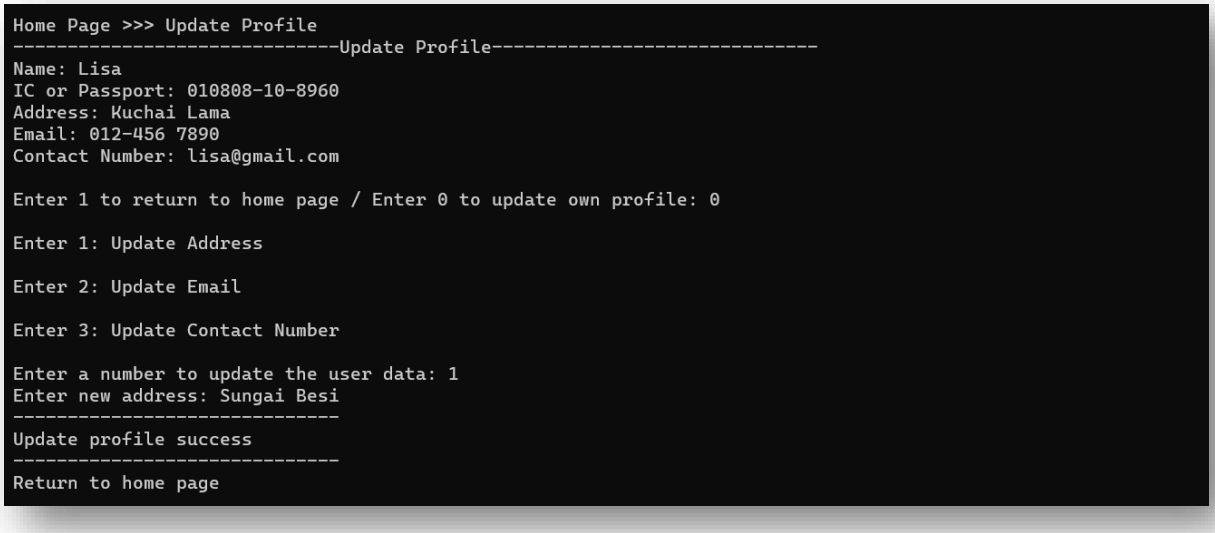

Figure 134 Update Own Profile Successful

7.0 Conclusion

To summarize, this system brings a huge benefit to Brilliant Tuition Centre to reduce cost for maintaining data and information. For example, the student's data will store to built-in file named "student.txt" while receptionist register a student's account and enrol subject. The system provides Menu-driven user interface to receptionist to modify student's data to student.txt file. Therefore, most of the Brilliant Tuition Centre's daily operational tasks can be efficiently managed through this system.

Moreover, this system utilizes encryption algorithm (SHA-256) to protect user's data such as password. The user's password is converted into a 256-bit hexadecimal number, making it difficult to decrypt. As a result, the hashing method enhance the user accounts security, ensuring that unauthorised access is nearly impossible even if others allow access to the user's data file.

Furthermore, the system supplies user-friendly environment to improve user's experience. For instance, the system prompts users to choose between returning to home page and continuing their activity once user enter input invalid or incorrect data. This ensures that no erroneous data is stored in any of the files related to system. Additionally, user also is not required to close the program if they don't want to continue the activity. Hence, this feature enable user to navigate tuition centre's functions easily and effectively.

Nevertheless, while the system uses strong security measures which is password encryption using SHA-256, it comes with a limitation concerning password recovery. Due to the irreversible nature of the encryption method, it is not possible for the system administrator to retrieve a user's password if the user forgets it. To address this limitation, it's crucial to emphasize that users are responsible for safeguarding their passwords. They should be encouraged to choose strong and memorable passwords and store them securely. In addition, the developer also can add forget password feature to the system. This feature allows users to change password by entering new password, confirm password and OTP (One-Time Password) to change account's password, the system will send the OTP to user's contact number when user require to change password.

8.0 References

Baivab Kumar Jena. (1 August, 2023). *A Definitive Guide to Learn The SHA-256 (Secure Hash Algorithms*. Retrieved from Simplilearn: https://www.simplilearn.com/tutorials/cyber-security-tutorial/sha-256-algorithm

Programiz. (2023). Retrieved 27 August, 2023, from Python Strings: https://www.programiz.com/python-programming/string

Python list. (2023). Retrieved 28 August, 2023, from w3school: https://www.w3schools.com/python/python_lists.asp

Python Software Foundation. (28 August, 2023). *Python*. Retrieved from Errors and Exceptions: https://docs.python.org/3/tutorial/errors.html

Toppr. (n.d.). Retrieved from Python File I/O: toppr.com/guides/python-guide/tutorials/python-files/