UNIVERSITI
TEKNOLOGI
PETRONAS

“ONE NIGHT HOTEL MANAGEMENT”

TEB1013 : STRUCTURED PROGRAMMING

(ASSIGNMENT)

Lecturer’s Name: *Dr. Ahmad Fadli Saad*

Date Submitted: 17 November 2023

No.	Name	Matric ID	Course
1	Mohamad Akram bin Mohd Faisal	22006626	Bachelor’s Degree in Computer Science (Hons)
2	Arleen April Chong	22006582	

Table of Contents

1. INTRODUCTION

In our modern age, technology has become a driving force reshaping the way hotels operate. To keep up with the pace, hotel owners need a management system that doesn't just simplify tasks but enhances productivity. A well-designed Hotel Management System (HMS) has transitioned from being a mere convenience to an absolute essential in the digital landscape. This project is a strategy that utilizes structured programming techniques for the development and implementation of a powerful HMS.

Our vision is to create a seamless platform, intuitively designed to empower hotel staff. This program will not only optimize daily operations but also enhance customer experiences. This project aims to propose an effective system that's easy to maintain. This ensures that the system can adapt and contribute to the changing needs of the industry, making it a practical and efficient tool.

2. SCOPE OF PROJECT

The Hotel Management System (HMS) proposed in this project will encompass a wide array of features and functionalities to enhance the overall efficiency of hotel operations. The scope of the project includes:

i. Room Management: The program will provide real-time information on room availability allowing customer to book rooms. Bedrooms are classified based on types (single, double) with corresponding rates and amenities.

ii. Inventory Management: The program tracks room service meal capacity. This feature allows guests to order food, and beverages, directly to their rooms.

iii. Reporting and Analytics: The program will generate daily reports for analysis and planning.

3. POTENTIAL CUSTOMER (HOTEL COMPANY)

A potential company in Malaysia that could benefit from using the proposed program is "Berjaya Hotels & Resorts." It is a well-known hospitality group in Malaysia that operates luxury hotels and resorts in various popular tourist destinations, including Kuala Lumpur, Langkawi, and Penang. By implementing an advanced HMS, Berjaya Hotels & Resorts could optimize their operations, enhance guest experiences, and ensure efficient management of their high-end establishments. This would not only streamline their complex operations but also increase their brand reputation and customer satisfaction across their luxury properties.

Other than that, is "Tune Hotels." Tune Hotels is a budget hotel chain with multiple properties across Malaysia, catering to budget-conscious travellers and backpackers. By implementing a cost-effective and tailored HMS, it would enable them to efficiently manage their budget-friendly accommodations and provide a seamless experience to their guests, ultimately improving their competitiveness in the market.

4. WHY DO WE USE STRUCTURED PROGRAMMING

Structured programming is a subset of procedural programming that enforces a logical structure on the program being written. We use structured programming in our project to make it more efficient and easier to understand and modify.

Furthermore, Structured programming encourages dividing an application program into a hierarchy of modules, which, may contain other such elements. Within each element, code may be further structured using blocks of related logic designed to improve readability and maintainability.

5. WHY DO WE USE C++

5.1. *Efficiency*

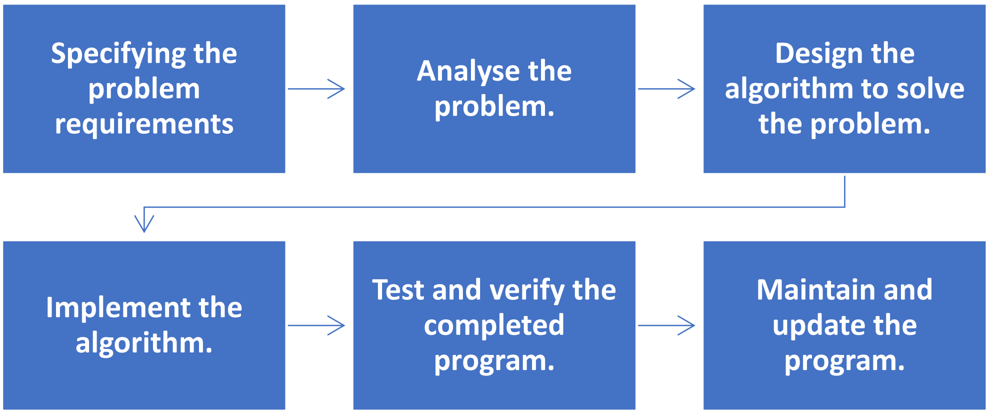
C++ is a high-performance language that can handle complex computations and large amounts of data quickly and efficiently. This is important for a hotel management system that needs to process reservations, payments, and customer data in real-time. This ensures that these operations are executed swiftly, providing a seamless experience for both customers and hotel staff.

Object-oriented programming: C++ supports object-oriented programming, which allows developers to organize code into reusable modules and classes. This can make the code more modular, easier to maintain, and less prone to errors.

5.2. *File handling*

C++ has built-in support for file handling, which is useful for storing and retrieving data from files. This is important for a hotel management system that needs to store customer information, room reservations, and payment records. File handling ensures that this information is persistent across sessions, allowing our system to maintain accurate records even after restarting. This functionality is essential for auditing, generating reports, and ensuring data integrity over time.

6. PROGRAMMING LIFE CYCLE


Specifying the problem requirements
Analyse the problem.
Design the algorithm to solve the problem.
Implement the algorithm.
Test and verify the completed program.
Maintain and update the program.

6.1. *Specifying the problem requirements*

Hotel owners need a management system that doesn't just simplify tasks but enhances productivity.

- Identify the key features: Room Management, Inventory Management, Reporting and Analytics.
- Define the types of rooms (single, double) and their corresponding rates and amenities.
- Clearly outline the reporting and analytics needs, such as the data to be included in daily reports.

6.2. ***Analyse the problem.***

- Consider the data structures needed (e.g., arrays, structs, classes) to represent rooms, orders, and reports.
- Identify algorithms for booking rooms, managing inventory, and generating reports.
- Define input and output mechanisms for user interaction.

6.3. ***Design the algorithm to solve the problem.***

- Design classes or structures for representing rooms, orders, and reports.
- Create algorithms for booking rooms, managing inventory, and generating reports.
- Plan the user interface for interacting with the program.

6.4. ***Implement the algorithm.***

- Write the C++ code based on the designed algorithms and data structures.
- Implement functions/methods for room booking, inventory management, and report generation.
- Develop user interfaces for ease of interaction.

6.5. ***Test and verify the completed program.***

- Conduct unit tests for individual components and functionalities.
- Perform integration tests to ensure the different modules work together.

- Test the program with various scenarios to ensure robustness

6.6. ***Maintain and update the program.***

- Monitor for any issues or bugs and address them promptly.
- Consider user feedback for potential improvements.
- Update the program to accommodate new features or changes in requirements.

7. MODULES

7.1. ***Login Module (login()):***

- This module handles user authentication.
- It allows users to log in as either an admin or a regular user.
- Admin credentials are hardcoded, while regular user credentials are stored in a file (**credentials.txt**).
- After successful login, users are directed to their respective menus.

7.2. ***Registration Module (registration()):***

- Allows users to register by providing a user ID and password.
- Stores the user credentials in a file (**credentials.txt**).
- After registration, users are automatically logged in.

7.3. ***Forgot Password Module (forgot()):***

- Assists users in recovering their password by entering their user ID.
- Checks if the user ID exists, and if so, allows the user to set a new password.

7.4. ***User Menu Module (userMenu()):***

- Displays options for users after login.
- Options include room booking, amenities request, checking the bill, and logging out.

- Users can navigate through these options.

7.5. ***Room Booking Module (roomBooking()):***

- Allows users to book single or twin rooms based on availability.
- Updates room availability and user bills accordingly.
- Creates a bill report file for the user.

7.6. ***Amenities Request Module (amenitiesRequest()):***

- Allows users to request amenities (towel, water, biscuit, pillow) after booking a room.
- Updates amenity availability and user bills accordingly.

7.7. ***Check Bill Module (checkBill()):***

- Displays the user's bill by reading from their bill report file.
- Shows the total spending on room bookings and amenities.
- Gives the option to exit or return to the user menu.

7.8. ***Reset Current Record Module (resetCurrentRecord()):***

- Admin-only module to reset the current record (quantities of rooms and amenities).
- Renames the current record file and creates a new one with initial quantities.

7.9. ***Admin Menu Module (AdminMenu()):***

- Displays options for the admin after login.
- Options include viewing room and amenity availability, generating a sales report, resetting the current record, and logging out.

7.10. ***View Rooms Module (viewRooms()):***

- Admin-only module to display the availability of single and twin rooms.

7.11. ***View Amenities Module (viewAmenities()):***

- Admin-only module to display the availability of amenities (towel, pillow, water, biscuit).

7.12. *View Prices Module (viewPrices()):*

- Currently not implemented but could display the prices of different room types and amenities.

7.13. *Sales Report Module (salesReport()):*

- Admin-only module to generate a sales report.
- Calculates and displays the total sales for each item and the grand total.

8. INPUTS OF THE PROJECT

8.1. *Menu Choices:*

- Users are presented with a menu when the program starts, giving them various options to choose from (e.g., registration, login, amenities request, room booking, etc.).
- The user is prompted to enter a numerical choice corresponding to the desired action.

8.2. *User Authentication:*

- During the login and registration processes, users are prompted to enter their user ID and password.
- The program expects input for user authentication and checks against stored credentials.

8.3. *Room Booking and Amenity Requests:*

- Users input their choices when booking rooms or requesting amenities. For example, they may choose between single and twin rooms or select specific amenities.
- The program validates and processes these choices, updating availability and generating bills.

8.4. ***Date Entry in Reset Current Record:***

- In the admin-specific module for resetting the current record, the user is prompted to enter the date (DD-MM-YYYY) for the reset.

8.5. ***Numeric Input:***

- For various quantities (e.g., the number of rooms or amenities), users are expected to input numeric values.
- The program validates and processes numeric input for calculations and updates.

8.6. ***Yes/No Choices:***

- In some cases (e.g., when users are asked if they want to exit), the program expects a yes/no choice from the user.
- Users enter 1 for "Yes" and 2 for "No" in these cases.

8.7. ***Input for Administrative Actions:***

- In admin-specific modules, the interface may prompt for additional information specific to administrative tasks, such as confirmation for a reset operation.

8.8. ***Handling Invalid Input:***

- The program includes mechanisms to handle invalid input gracefully. For instance, if a user enters an invalid choice, the program may display an error message and prompt the user again.

9. PROCESSES AND CALCULATIONS

9.1. ***Room Booking Process (roomBooking() function):***

9.1.1. User Input:

- Users choose between single and twin rooms based on availability.

9.1.2. Calculation:

- The program calculates the total cost based on the chosen room type.
- Updates the global variables (**Total_rooms**, **Ssingle**, **Stwin**, **Qsingle**, **Qtwin**) to reflect the booking.

9.1.3. File Operations:

- Updates the **current_record.txt** file to reflect the reduced availability of rooms.
- Creates a bill report file for the user.

9.2. ***Amenities Request Process (amenitiesRequest() function):***

9.2.1. User Input:

- Users choose from available amenities (towel, water, biscuit, pillow).

9.2.2. Calculation:

- The program calculates the total cost based on the chosen amenities.
- Updates the global variables and quantities accordingly (**Total_towel**, **Total_water**, **Total_biscuit**, **Total_pillow**, **Qtowel**, **Qwater**, **Qbiscuit**, **Qpillow**).

9.2.3. File Operations:

- Updates the **current_record.txt** file to reflect reduced availability of amenities.
- Updates the user's bill report file.

9.3. *Check Bill Process (checkBill() function):*

9.3.1. File Operations:

- Reads the user's bill report file to display the items and their costs.
- Calculates the total spending based on the bill report.

9.4. *Reset Current Record Process (resetCurrentRecord() function):*

9.4.1. User Input:

- Admin provides the date for the reset.

9.4.2. File Operations:

- Renames the existing **current_record.txt** file to **[Date]_record.txt**.
- Creates a new **current_record.txt** file with initial quantities for rooms and amenities.

9.5. *Sales Report Process (salesReport() function):*

9.5.1. Calculation:

- Calculates the total sales for each item and the grand total (**Total_single**, **Total_twin**, **Total_towel**, **Total_water**, **Total_biscuit**, **Total_pillow**).

9.5.2. Display:

- Displays a sales report with quantities sold and total sales.

10. OUTPUTS OF THE PROJECT

10.1. ***Main Menu Output:***

- Welcomes the user and presents a clean interface with options for various actions.
- Encourages the user to choose from registration, login, password recovery, or exit.

10.2. ***Login Output:***

- Prompts the user to enter their credentials (User ID and Password).
- Informs the user about the success or failure of the login attempt.
- For administrators, it allows access to the AdminMenu.

10.3. ***Registration Output:***

- Guides the user through creating a new account.
- Informs the user about the success or failure of the registration process.
- After successful registration, automatically logs in the user.

10.4. ***Password Recovery Output:***

- Assists users in recovering their password.
- Prompts the user to enter a new password.
- Informs the user about the success or failure of the password update.

10.5. ***User Menu Output:***

- Presents the user with a menu for common actions after login.
- Options include room booking, amenities request, checking the bill, and logout.

10.6. ***Room Booking Output:***

- Checks and displays the availability of single and twin rooms.
- Allows the user to choose the room type and confirms the booking.
- Updates the user's bill report with the booked room details.

10.7. ***Amenities Request Output:***

- Checks if the user has booked a room before processing amenities requests.
- Prompts the user to select an amenity and updates the bill accordingly.
- Updates the stock of available amenities.

10.8. ***Check Bill Output:***

- Displays the user's bill, detailing room bookings and amenities.
- Calculates and shows the total spending.
- Gives the option to exit or return to the user menu.

10.9. ***Admin Menu Output:***

- Offers administrative functions such as viewing availability and generating reports.
- Options include viewing room and amenity availability, viewing sales reports, and resetting the current record.

10.10. ***Viewing Room and Amenity Availability (Admin) Output:***

- Shows the administrator the current availability of rooms and amenities.
- Allows the administrator to decide whether to exit or continue.

10.11. ***Reset Current Record (Admin) Output:***

- Guides the administrator in resetting the current record, including entering the date.
- Informs about the success or failure of the file renaming process.

10.12. ***Viewing Sales Report (Admin) Output:***

- Displays a comprehensive sales report for the administrator.
- Includes details on quantities sold and total sales for each item.
- Allows the administrator to exit or return to the admin menu.

11. BENEFITS OF OUR PROJECT

11.1. ***Efficient Operations:***

Streamlined bookings reduce manual errors and save time for both staff and guests. Besides that, automated room assignment based on availability optimizes occupancy rates. Lastly, our program in efficient in tracking of supplies, hence, minimizing wastage, and ensuring timely restocking.

11.2. ***Data-Driven Decisions:***

This program produces a detailed analytic report on occupancy rates and revenue, which provide insights for strategic planning. This historical data analysis aids in forecasting demand, enabling better resource allocation.

12. APPENDIX

12.1. *Algorithm*

1.0 Initialization:

1.1 Declare global variables for room and amenity quantities.

1.2 Define modules for resetting the current record, reading the current record, user login, registration, password recovery, user menu, admin menu, room booking, amenities request, check bill, view rooms, view amenities, view prices, sales report.

2.0 Main Module:

2.1 Call the readCurrentRecord module to initialize or read the current record.

2.2 Display the main menu options:

2.2.1 Register

2.2.2 Login

2.2.3 Forgot Password

2.2.4 Exit

2.3 Read the user's choice.

2.4 Switch based on the user's choice:

2.4.1 Case 1: Call registration module

2.4.2 Case 2: Call login module

2.4.3 Case 3: Call forgot module

2.4.4 Case 4: Display farewell message and terminate the program

2.4.5 Default: Display an invalid choice message and call the main module recursively.

3.0 Read Current Record Module:

```
  3.1 Open the current_record.txt file for reading.
  3.2 If the file exists:
      3.2.1 Read the quantities of each item from the file.
      3.2.2 Close the file.
      3.2.3 Copy the quantities to global variables.
  3.3 If the file does not exist:
      3.3.1 Create the file and initialize quantities.
      3.3.2 Close the new file.
      3.3.3 Copy initialized quantities to global variables.

4.0 Reset Current Record Module (Admin):
  4.1 Display a reset current record header.
  4.2 Prompt the admin to enter the reset date.
  4.3 Rename the current_record.txt file to [Date]_record.txt.
  4.4 Call the readCurrentRecord module to create a new current_record.txt file.
  4.5 Display a message indicating the successful file renaming.
  4.6 Call the AdminMenu module.

5.0 User Login Module:
  5.1 Display a login header.
  5.2 Prompt the user to enter their user ID and password.
  5.3 If admin credentials, call AdminMenu module.
  5.4 If regular user credentials, call userMenu module.
  5.5 Display a login error if credentials are invalid.
```

```
6.0 User Registration Module:
   6.1 Display a registration header.
   6.2 Prompt the user for a desired user ID and password.
   6.3 Store the credentials in the "credentials.txt" file.
   6.4 Call the login module.

7.0 Forgot Password Module:
   7.1 Display a forgot password header.
   7.2 Prompt the user for their user ID.
   7.3 Search for the user's ID in the "credentials.txt" file.
   7.4 If found, prompt the user for a new password and update the file.
   7.5 Call the login module to log in the user with the new password.
   7.6 Display a message if the user ID is not found.

8.0 User Menu Module:
   8.1 Display user menu options (room booking, amenities request, check bill, logout).
   8.2 Read the user's choice.

9.0 Room Booking Module:
   9.1 Check room availability.
   9.2 Prompt the user to select a room type.
   9.3 Update the current record file and the user's bill report file.
   9.4 Display a success message.
```

9.5 Call the userMenu module.

10.0 Amenities Request Module:

10.1 Check if the user has booked a room.

10.2 Prompt the user to select an amenity.

10.3 Update the current record file and the user's bill report file.

10.4 Display a success message.

10.5 Call the userMenu module.

11.0 Check Bill Module:

11.1 Check if the user has a bill report file.

11.2 Read the bill file, calculate total spending, and display the bill.

11.3 Prompt the user to exit or go back to the user menu.

12.0 Admin Menu Module:

12.1 Display admin menu options (view room availability, view amenities, sales report, reset current record, logout).

12.2 Read the admin's choice.

13.0 View Room Availability Module (Admin):

13.1 Display the number of available single and twin rooms.

13.2 Prompt the admin to exit or go back to the admin menu.

14.0 View Amenities Availability Module (Admin):

14.1 Display the number of available amenities.

14.2 Prompt the admin to exit or go back to the admin menu.

15.0 View Prices Module (Admin):

15.1 Display the prices for each item.

16.0 Sales Report Module (Admin):

16.1 Calculate and display the total sales for all items.

16.2 Prompt the admin to exit or go back to the admin menu.

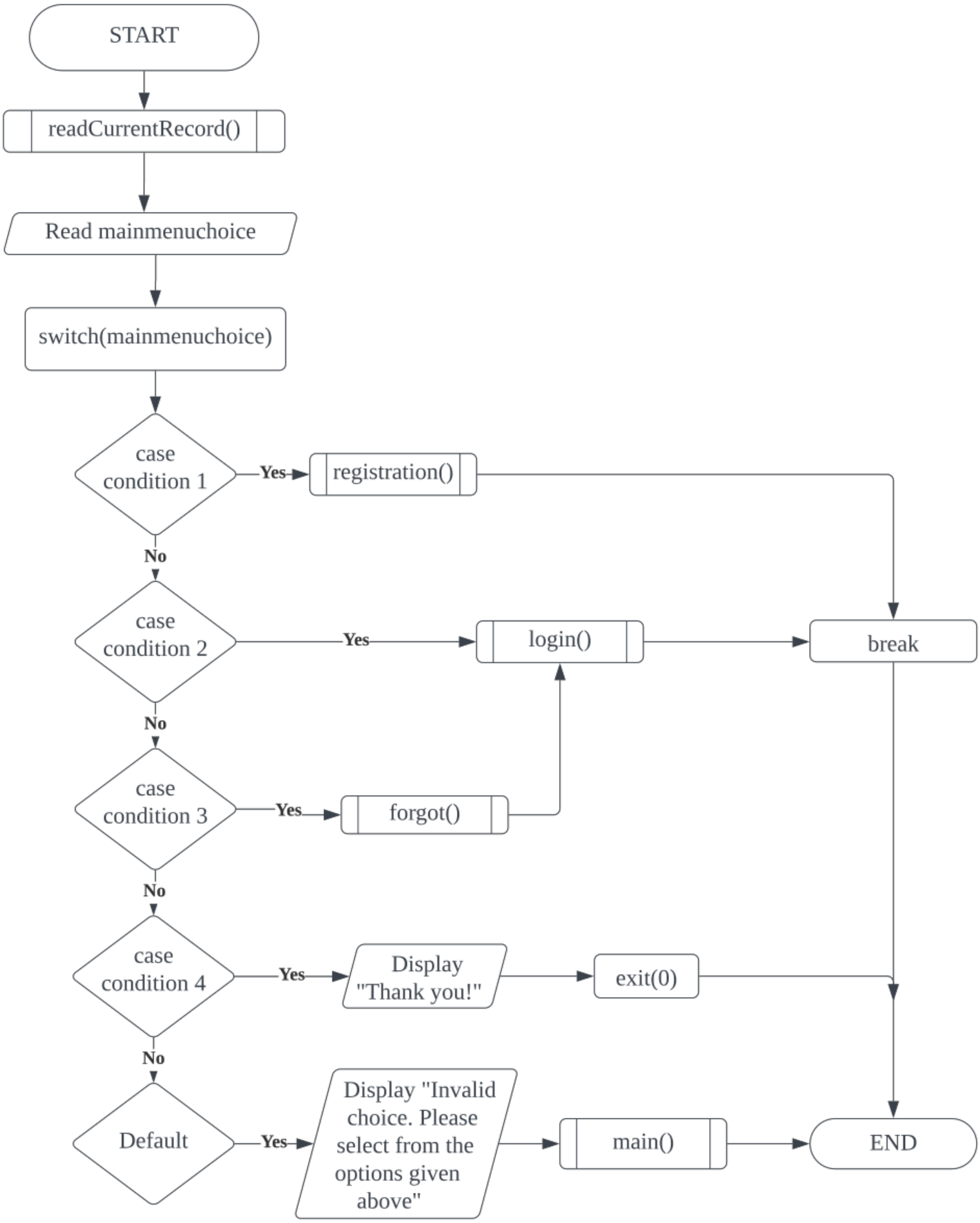
12.2. ***Pseudocode***

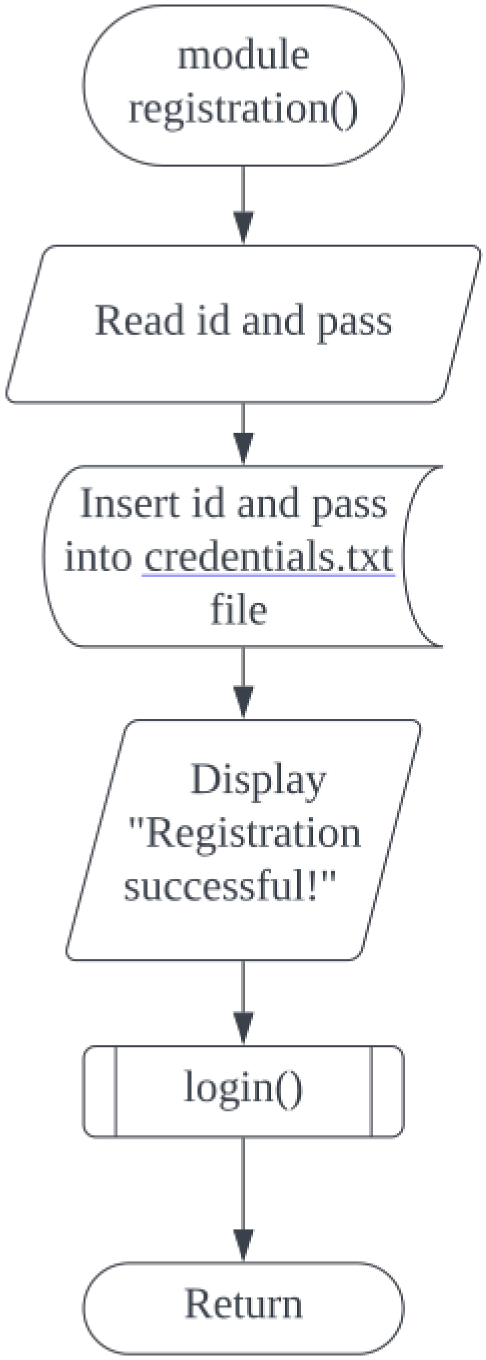
```
1.0 global variable initialization
    1.1 define global variable currentRecordFile = "current_record.txt"

    // Declare global variables
    1.2 int single = 65, twin = 100;
    1.3 int Qsingle = 20, Qtwin = 40, Qtowel = 75, Qwater = 100, Qbiscuit = 100, Qpillow = 75;
    1.4 int quant, choice, total_room = 0;

    // Declare variables to track the total price for each item
    1.5 int Total_rooms = 0, Total_single = 0, Total_twin = 0, Total_towel = 0, Total_water = 0, Total_biscuit = 0, Total_pillow = 0;

    // Declare variable to track sold items
    1.6 int Ssingle, Stwin, Stowel, Swater, Sbiscuit, Spillow;


    1.7 int count = 0;  // A variable to track login attempts

    1.8 string userID, password, id, pass;  // Strings to store user credentials


2.0 function main()

    2.1 call readCurrentRecord() // Read initial quantities from currentRecordFile

    2.2 display welcome message and menu options // Display welcome message and menu options

    2.3 read user's choice using cin

    2.4 switch(choice)

        2.4.1 case 1.0: call registration() // User chooses registration

        2.4.2 case 2.0: call login() // User chooses login

        2.4.3 case 3.0: call forgot() // User forgot password

        2.4.4 case 4.0: display farewell message and exit // User chooses to exit

        2.4.5 default: display invalid choice message and call main() // Invalid choice, go back to the main menu


3.0 module login()

    3.1 prompt user for User ID and Password using cout and cin

    3.2 check if credentials match admin credentials

        3.2.1 if yes, display admin login success message and call AdminMenu()

        3.2.2 if no, check credentials in credentials.txt file

            3.2.2.1 if found, set login attempt count to 1, display user login success message, and call userMenu()
            3.2.2.2 if not found, display login error message and call main()

4.0 module registration()
    4.1 prompt user for desired User ID and Password using cout and cin
    4.2 store user's credentials in credentials.txt file
    4.3 display registration success message
    4.4 call login()

5.0 module forgot()
    5.1 prompt user for User ID using cout and cin
    5.2 search for user's credentials in credentials.txt file
    5.3 if found, prompt user for a new password, update the password in the file, display password updated message, and call login()
    5.4 if not found, display User ID not found message and call forgot()

6.0 module userMenu()
    6.1 Display user menu options
    6.2 read user's choice using cin
    6.3 switch(choice)
        6.3.1 case 1.0: call roomBooking() // User chooses room booking
        6.3.2 case 2.0: call amenitiesRequest() // User chooses amenities request
        6.3.3 case 3.0: call checkBill() // User chooses to check the bill
        6.3.4 case 4.0: display logout message and call main() // User chooses to log out
        6.3.5 default: display invalid choice message and call userMenu() // Invalid choice, go back to the user menu
```

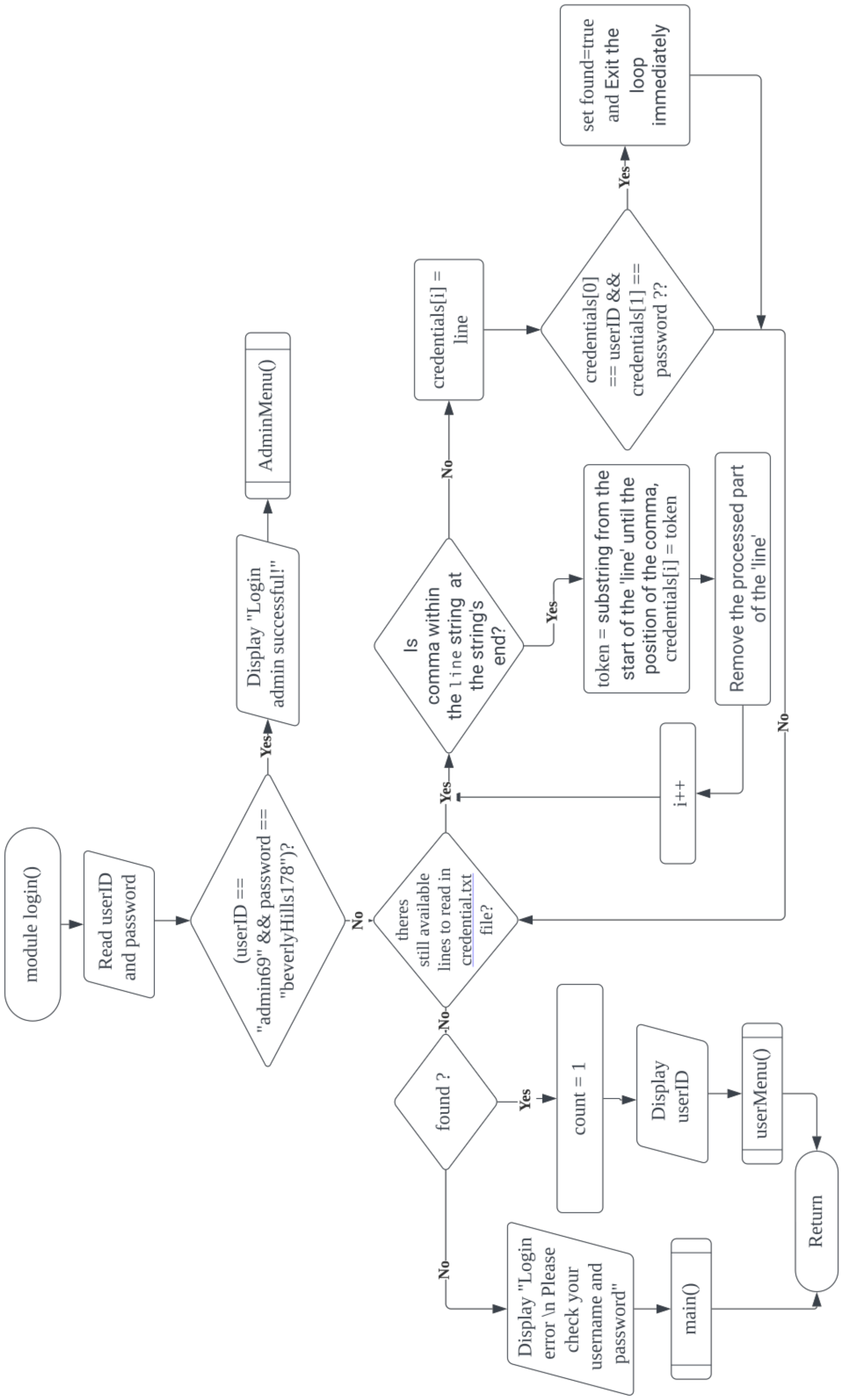
```
7.0 module roomBooking()
    7.1 call readCurrentRecord() // Read current stock quantities from currentRecordFile
    7.2 if no rooms available, display no rooms available message and call userMenu()
    7.3 prompt user to select a room type using cout and cin
    7.4 update currentRecordFile to reflect booking
    7.5 create a user's bill report file (userID_bill_report.txt) and add room booking details
    7.6 calculate the room booking cost based on the selected room type
    7.7 display room booked successfully message and call userMenu()

8.0 module amenitiesRequest()
    8.1 call readCurrentRecord() // Read current stock quantities from currentRecordFile
    8.2 check if the user has booked a room by looking for their bill report file (userID_bill_report.txt)
    8.3 if not booked, display must book a room first message and call userMenu()
    8.4 prompt the user to select an amenity using cout and cin
    8.5 add the amenity to the bill report file
    8.6 update stocks in currentRecordFile
    8.7 calculate the cost of the requested amenity
    8.8 display amenities request made successfully message and call userMenu()

9.0 module checkBill()
    9.1 check if the user has a bill report file (userID_bill_report.txt)
    9.2 if no bill, display no bill to check message and prompt the user to exit or go back to the user menu
    9.3 read the bill report file, extract the total spending, and display bill details
    9.4 prompt the user to exit or go back to the user menu

10.0 module readCurrentRecord()
    10.1 open current_record.txt for reading
    10.2 if the file does not exist, create it, initialize quantities, and copy quantities to global variables
    10.3 if the file exists, read quantities from the file and copy to the global variables

11.0 module resetCurrentRecord()
    11.1 display reset current record message
    11.2 prompt the admin for the date of reset using cout and cin
    11.3 rename currentRecordFile to [Date]_record.txt
    11.4 call readCurrentRecord()
    11.5 call adminMenu()

12.0 module adminMenu()
    12.1 Display admin menu options
    12.2 read admin input using cin
    12.3 switch(input)
        12.3.1 case 1.0: call viewRooms() // Admin chooses to view room availability
        12.3.2 case 2.0: call viewAmenities() // Admin chooses to view available amenities
        12.3.3 case 3.0: call salesReport() // Admin chooses to view the sales report
        12.3.4 case 4.0: call resetCurrentRecord() // Admin chooses to reset the current record
        12.3.5 case 5.0: call logout() // Admin chooses to log out
```

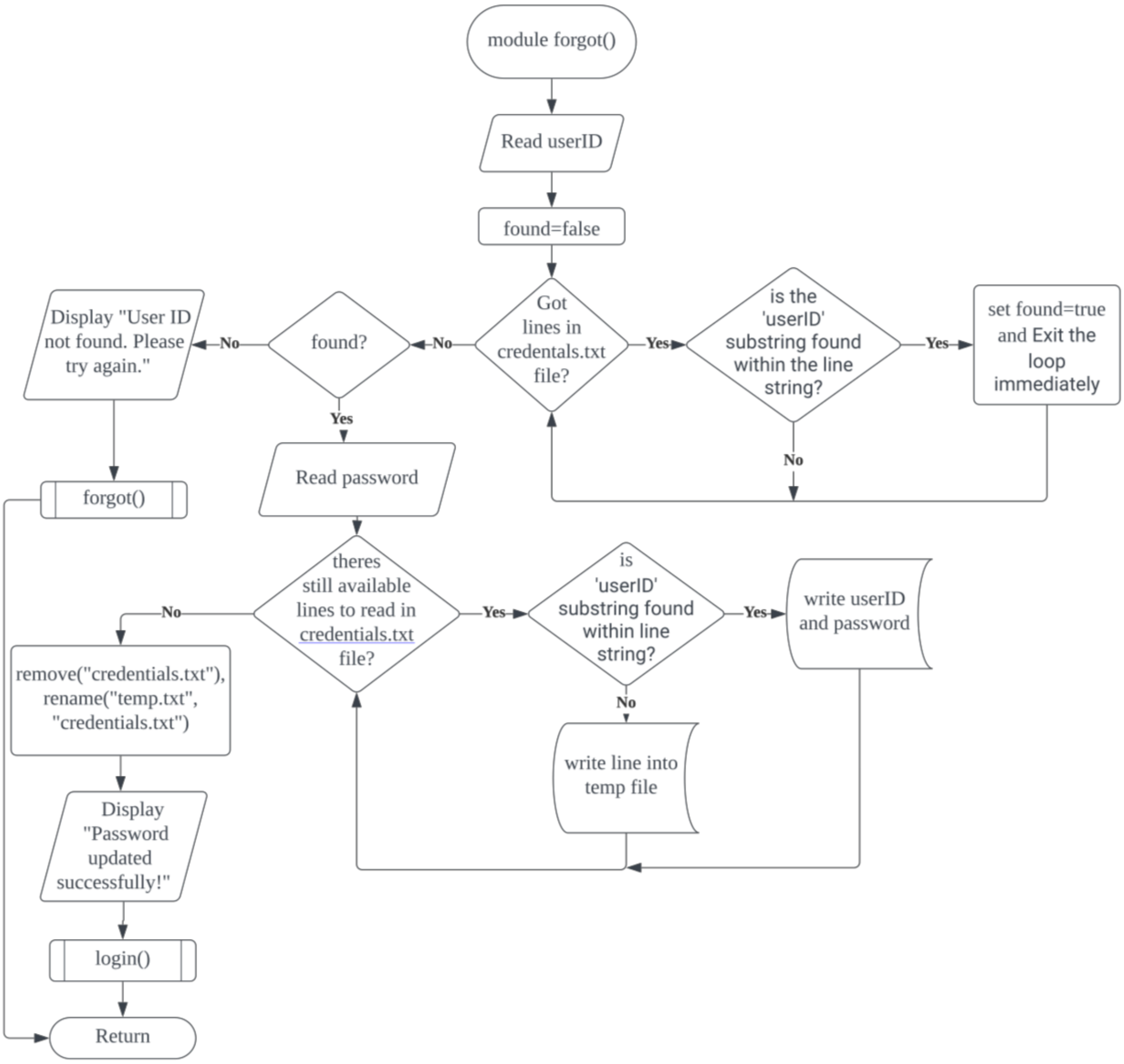
```
13.0 module viewRooms()
    13.1 call readCurrentRecord() // Read current stock quantities from currentRecordFile
    13.2 display available single and twin rooms
    13.3 prompt the admin to exit or go back to the admin menu

14.0 module viewAmenities()
    14.1 display available amenities
    14.2 prompt the admin to exit or go back to the admin menu

15.0 module viewPrices()
    15.1 display prices for each room types

16.0 module salesReport()
    16.1 calculate total sales from global variables
    16.2 display the sales report
    16.3 prompt the admin to exit or go back to the admin menu

17.0 END
```

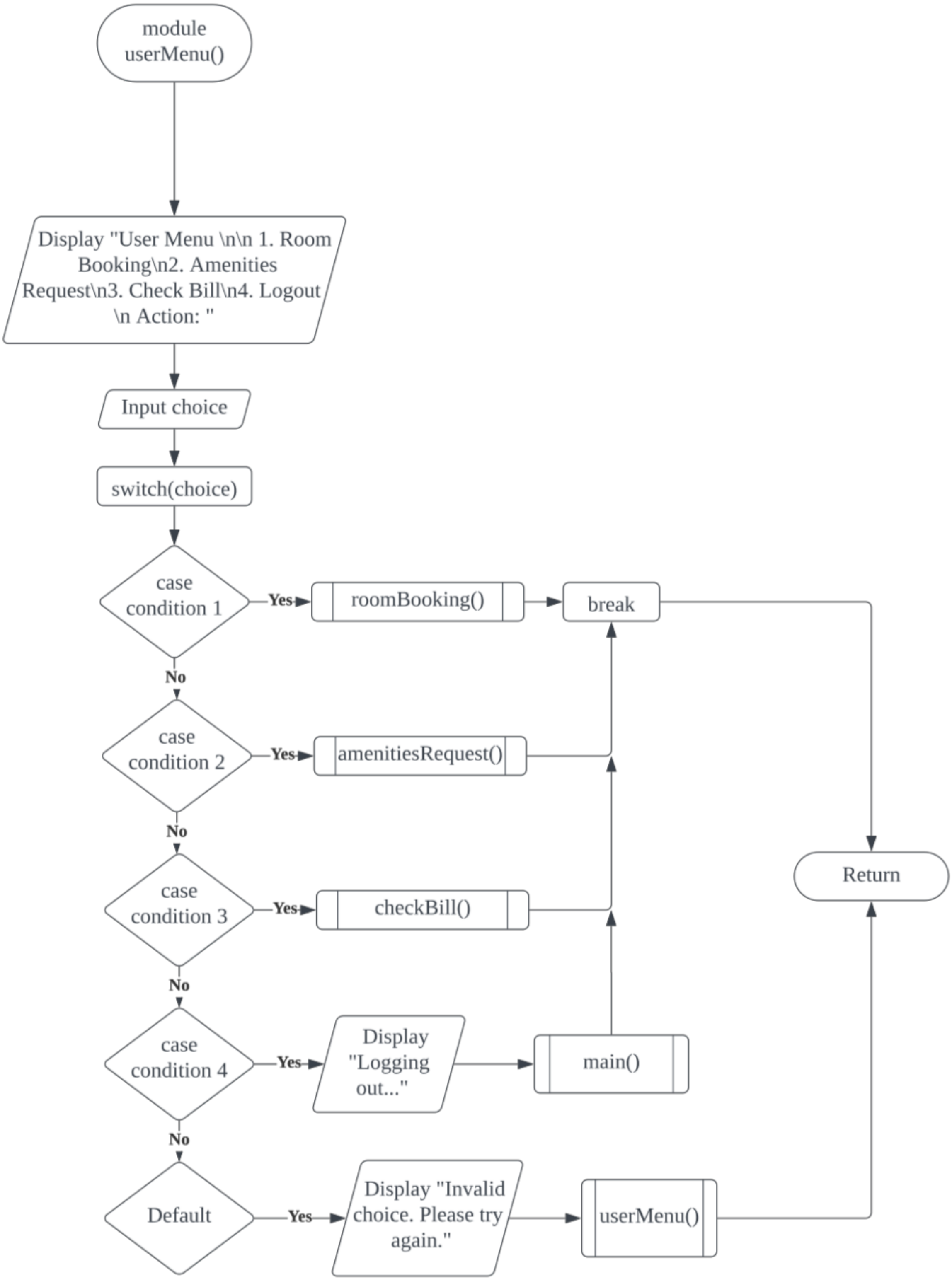
12.3. ***Flowchart***

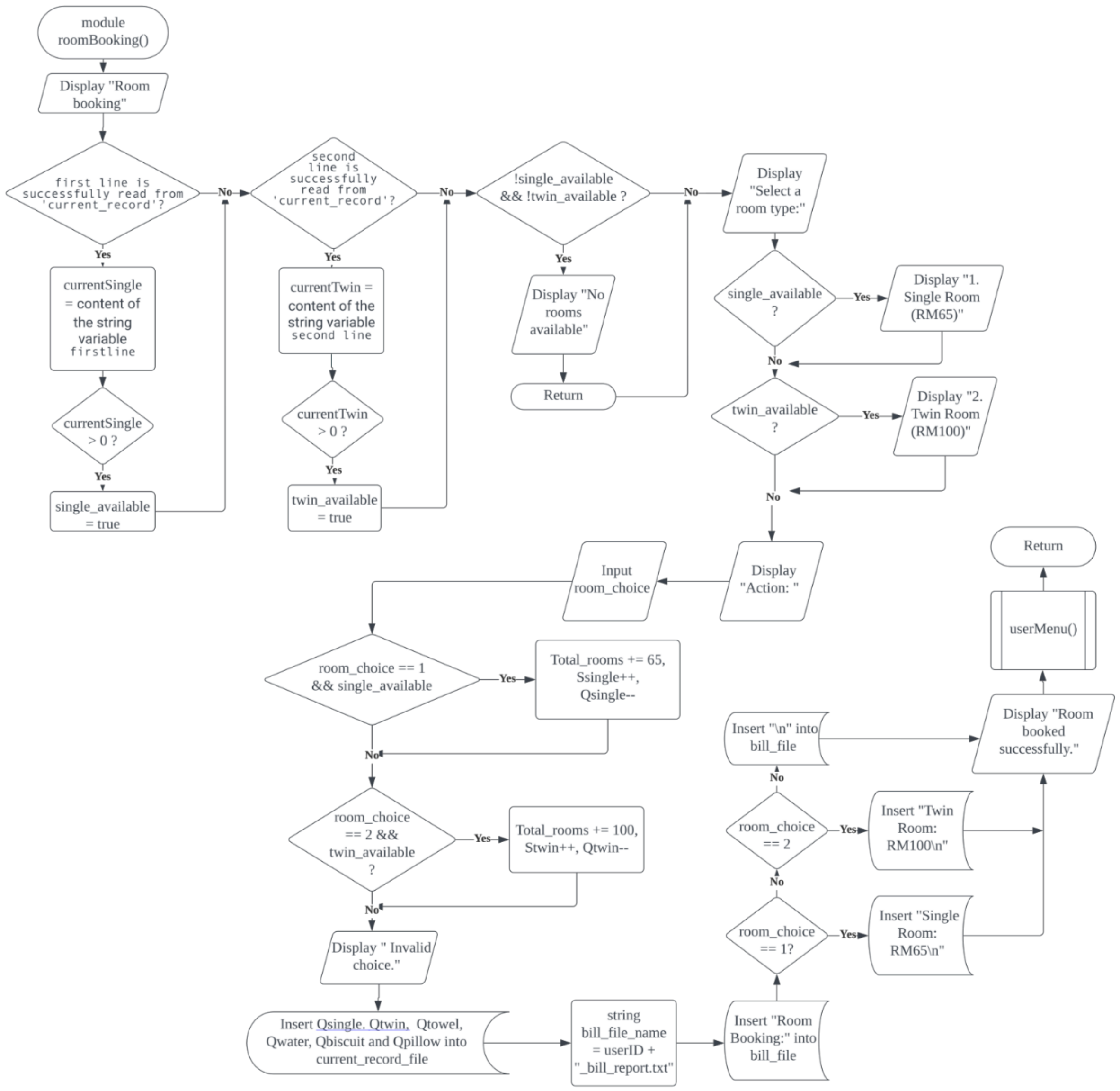


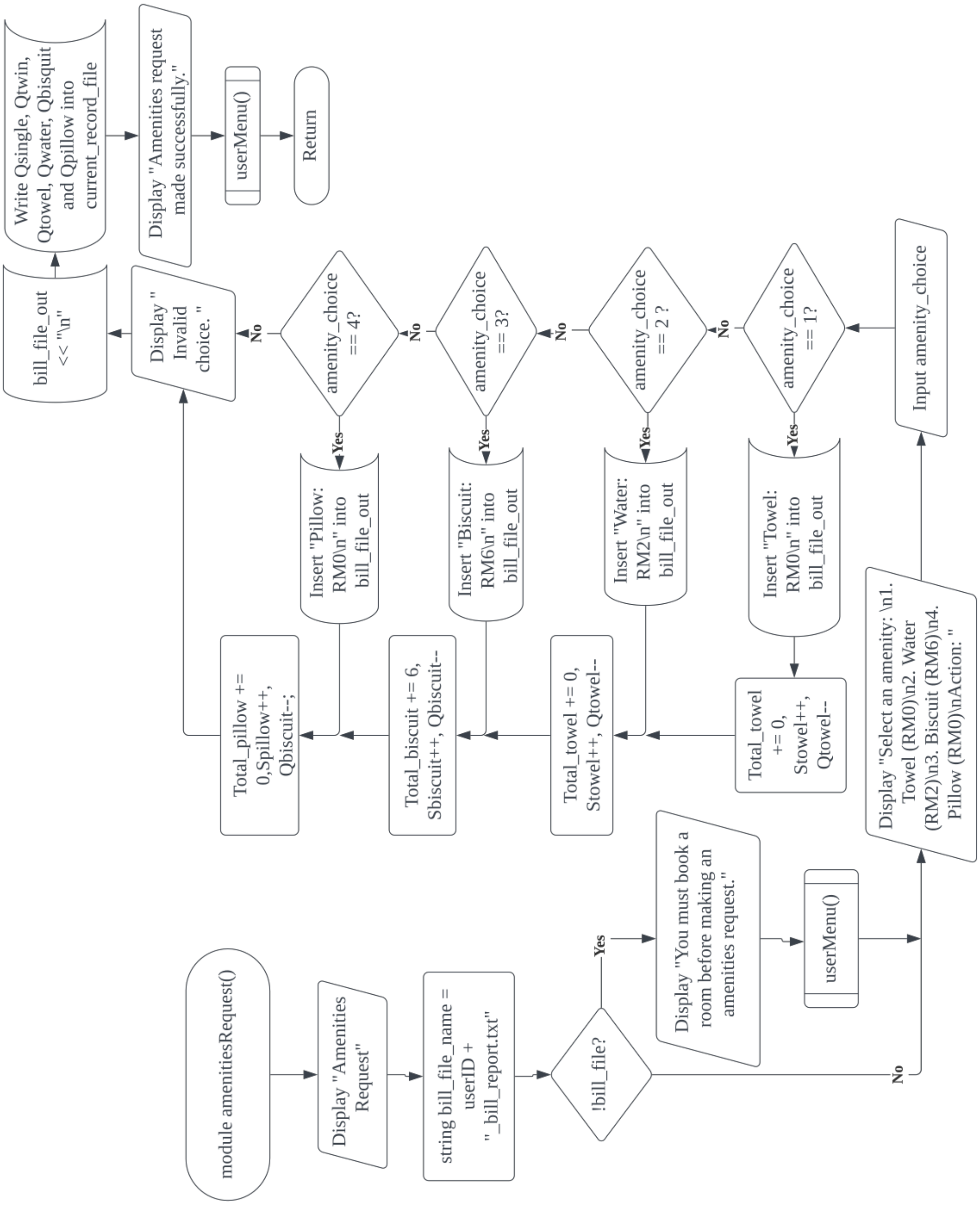

module registration()
Read id and pass
Insert id and pass into credentials.txt file
Display "Registration successful!"
login()
Return

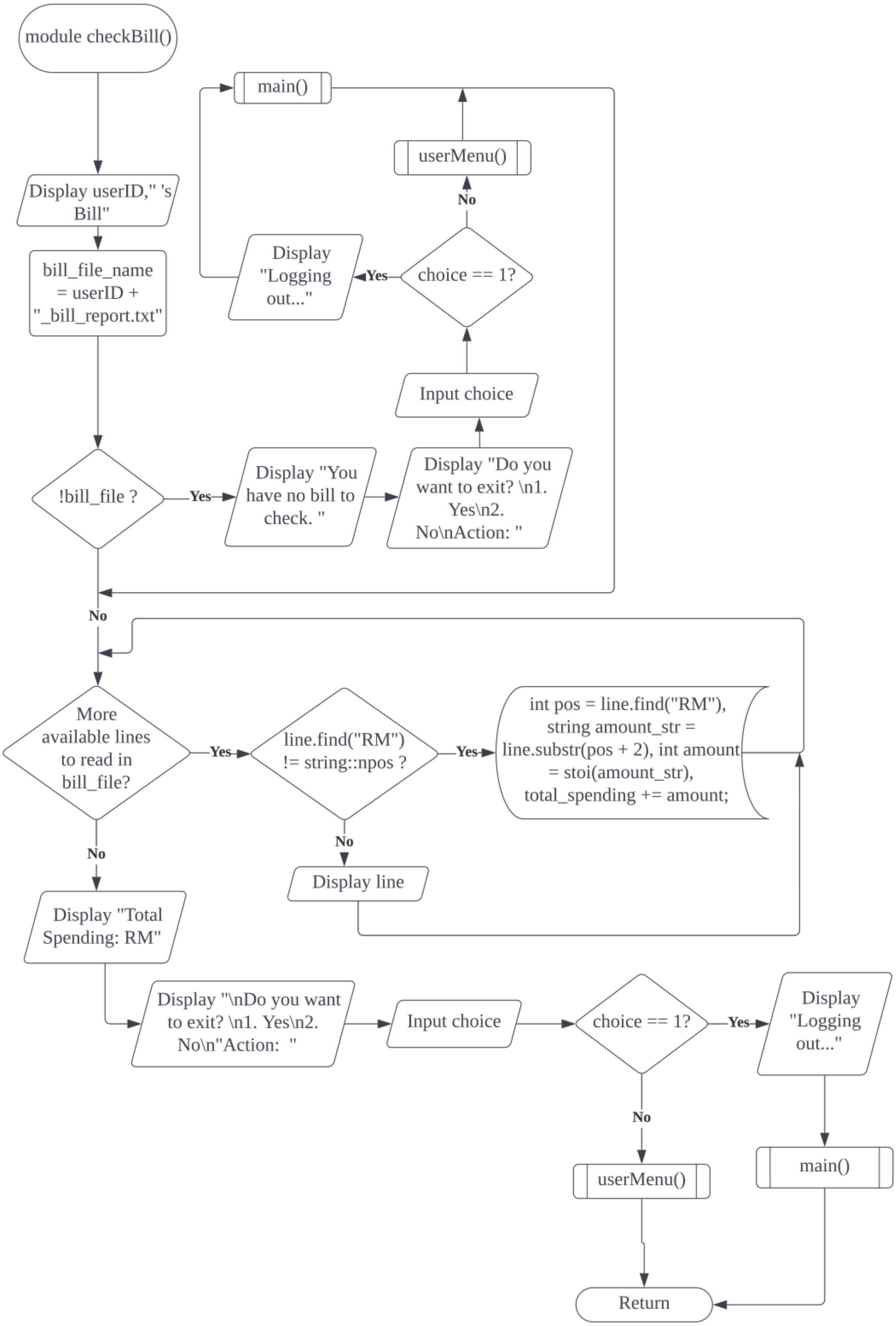

module login()
Read userID and password
(userID == "admin69" && password == "beverlyHills178")?
Yes
Display "Login admin successful!"
AdminMenu()
No
theres still available lines to read in credential.txt file?
Yes
Is comma within the line string at the string's end?
Yes
token = substring from the start of the 'line' until the position of the comma, credentials[i] = token
Remove the processed part of the 'line'
i++
No
credentials[i] = line
credentials[0] == userID && credentials[1] == password ??
Yes
set found=true and Exit the loop immediately
No
No
found ?
Yes
count = 1
Display userID
userMenu()
No
Display "Login error \n Please check your username and password"
main()
Return

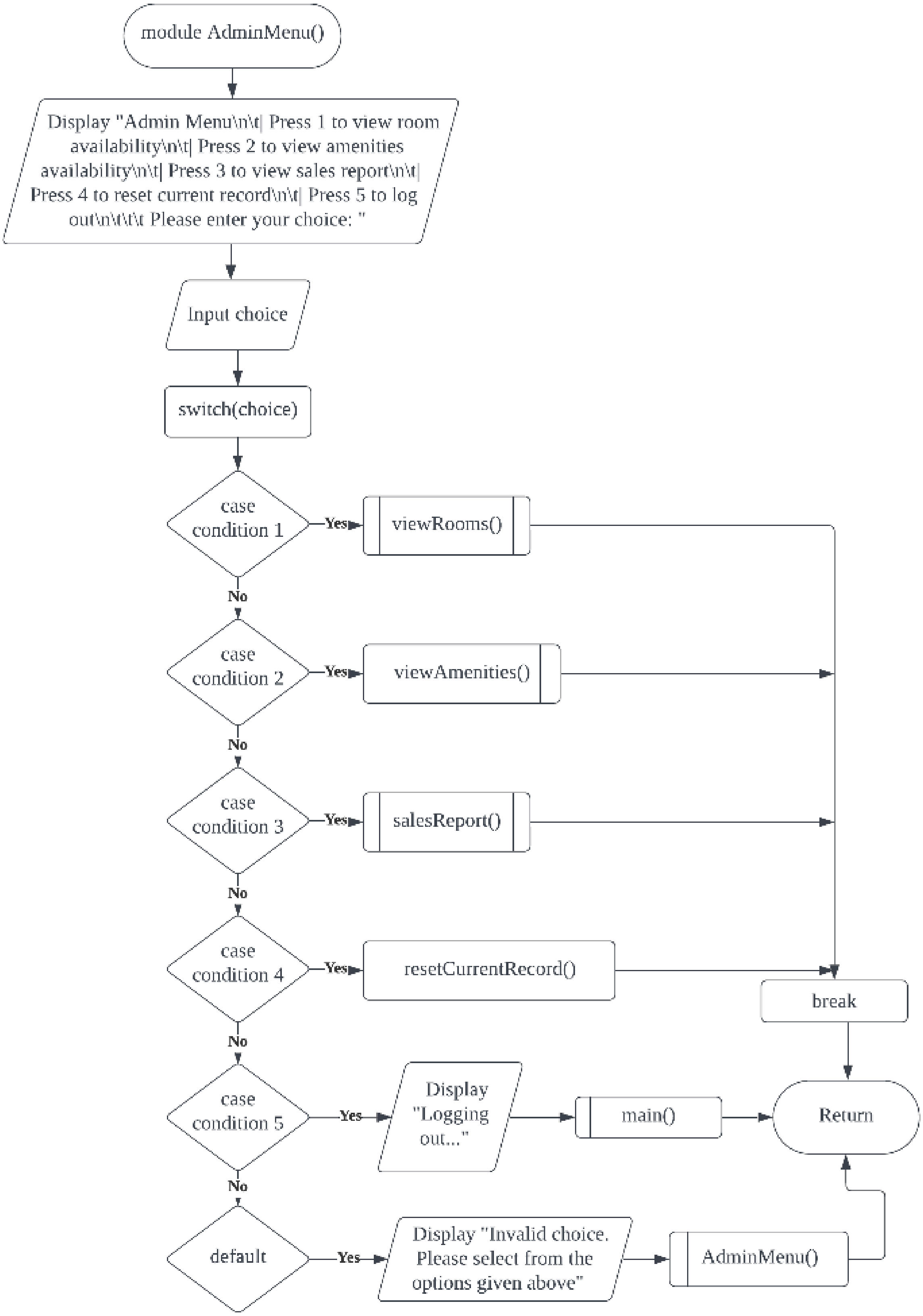

module forgot()
Read userID
found=false
Got lines in credentals.txt file?
Yes
is the 'userID' substring found within the line string?
Yes
set found=true and Exit the loop immediately
No
No
found?
No
Display "User ID not found. Please try again."
forgot()
Yes
Read password
theres still available lines to read in credentials.txt file?
Yes
is 'userID' substring found within line string?
Yes
write userID and password
No
write line into temp file
No
remove("credentials.txt"), rename("temp.txt", "credentials.txt")
Display "Password updated successfully!"
login()
Return

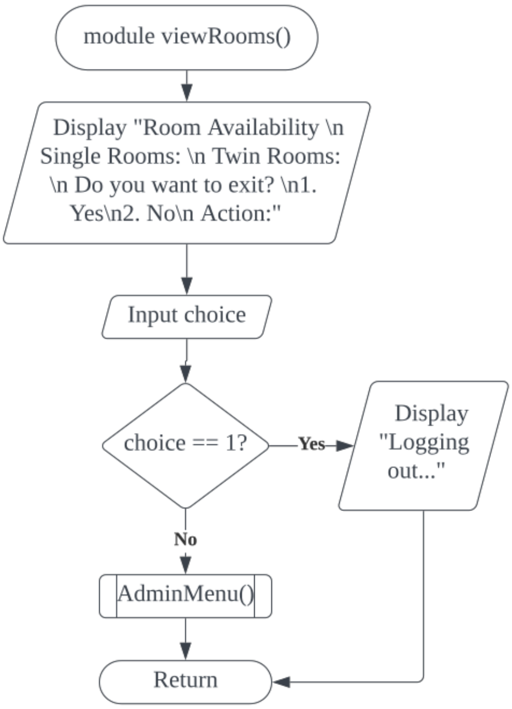

module userMenu()
Display "User Menu \n\n 1. Room Booking\n2. Amenities Request\n3. Check Bill\n4. Logout \n Action: "
Input choice
switch(choice)
case condition 1
Yes
roomBooking()
break
No
case condition 2
Yes
amenitiesRequest()
No
case condition 3
Yes
checkBill()
No
case condition 4
Yes
Display "Logging out..."
main()
No
Default
Yes
Display "Invalid choice. Please try again."
userMenu()
Return

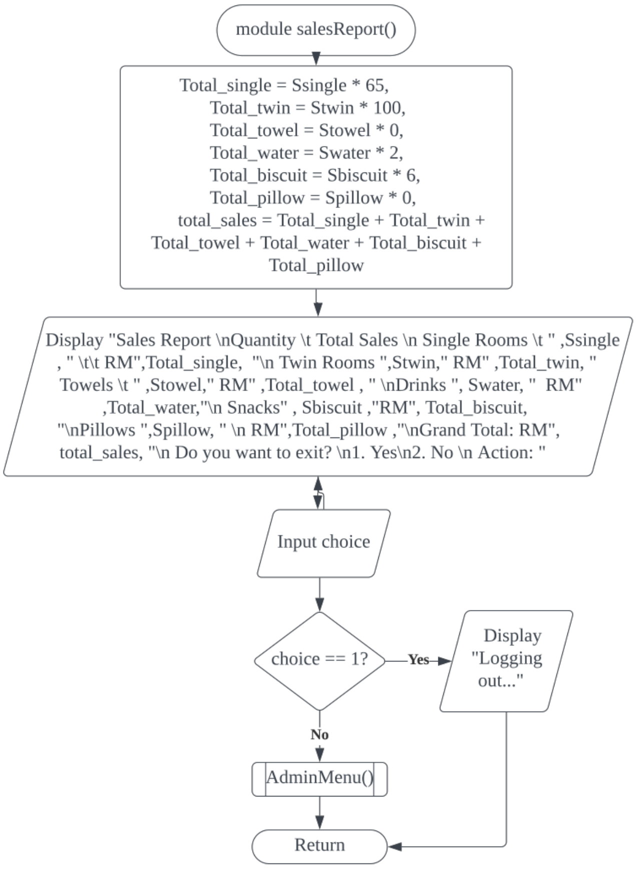

module roomBooking()
Display "Room booking"
first line is successfully read from 'current_record'?
Yes
No
currentSingle = content of the string variable firstline
currentSingle > 0 ?
Yes
single_available = true
second line is successfully read from 'current_record'?
Yes
No
currentTwin = content of the string variable second line
currentTwin > 0 ?
Yes
twin_available = true
!single_available && !twin_available ?
Yes
No
Display "No rooms available"
Return
Display "Select a room type:"
single_available ?
Yes
Display "1. Single Room (RM65)"
No
twin_available ?
Yes
Display "2. Twin Room (RM100)"
No
Display "Action: "
Input room_choice
room_choice == 1 && single_available
Yes
Total_rooms += 65, Ssingle++, Qsingle--
No
room_choice == 2 && twin_available ?
Yes
Total_rooms += 100, Stwin++, Qtwin--
No
Display " Invalid choice."
Insert Qsingle. Qtwin, Qtowel, Qwater, Qbiscuit and Qpillow into current_record_file
string bill_file_name = userID + "_bill_report.txt"
Insert "Room Booking:" into bill_file
room_choice == 1?
Yes
Insert "Single Room: RM65\n"
No
room_choice == 2
Yes
Insert "Twin Room: RM100\n"
No
Insert "\n" into bill_file
Display "Room booked successfully."
userMenu()
Return

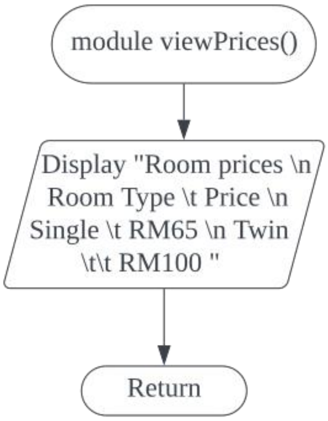

module amenitiesRequest()
Display "Amenities Request"
string bill_file_name = userID + "_bill_report.txt"
!bill_file?
Yes
No
Display "You must book a room before making an amenities request."
userMenu()
Display "Select an amenity: \n1. Towel (RM0)\n2. Water (RM2)\n3. Biscuit (RM6)\n4. Pillow (RM0)\nAction: "
Input amenity_choice
amenity_choice == 1?
Yes
No
Insert "Towel: RM0\n" into bill_file_out
Total_towel += 0, Stowel++, Qtowel--
amenity_choice == 2 ?
Yes
No
Insert "Water: RM2\n" into bill_file_out
Total_towel += 0, Stowel++, Qtowel--
amenity_choice == 3?
Yes
No
Insert "Biscuit: RM6\n" into bill_file_out
Total_biscuit += 6, Sbiscuit++, Qbiscuit--
amenity_choice == 4?
Yes
No
Insert "Pillow: RM0\n" into bill_file_out
Total_pillow += 0,Spillow++, Qbiscuit--;
Display " Invalid choice. "
bill_file_out << "\n"
Write Qsingle, Qtwin, Qtowel, Qwater, Qbisquit and Qpillow into current_record_file
Display "Amenities request made successfully."
userMenu()
Return

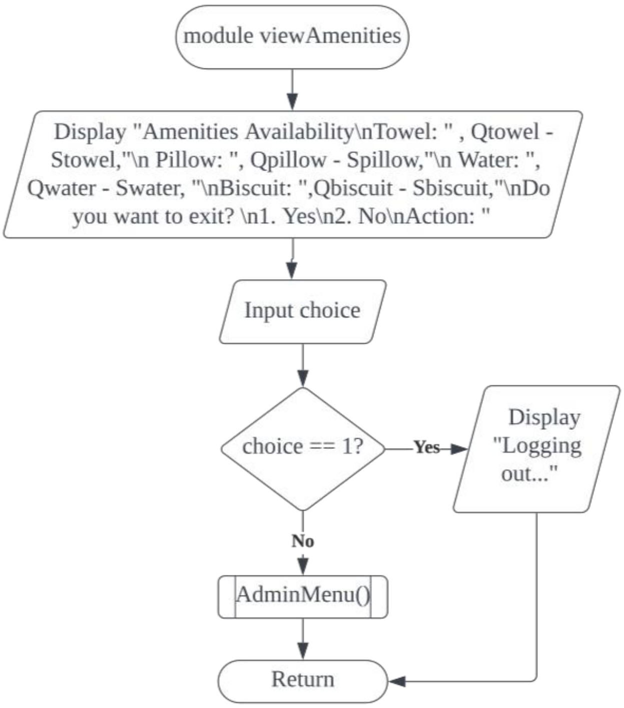

module checkBill()
Display userID," 's Bill"
bill_file_name = userID + "_bill_report.txt"
!bill_file ?
Yes
Display "You have no bill to check. "
Display "Do you want to exit? \n1. Yes\n2. No\nAction: "
Input choice
choice == 1?
Yes
Display "Logging out..."
main()
No
userMenu()
No
More available lines to read in bill_file?
Yes
line.find("RM") != string::npos ?
Yes
int pos = line.find("RM"), string amount_str = line.substr(pos + 2), int amount = stoi(amount_str), total_spending += amount;
No
Display line
No
Display "Total Spending: RM"
Display "\nDo you want to exit? \n1. Yes\n2. No\n"Action: "
Input choice
choice == 1?
Yes
Display "Logging out..."
main()
No
userMenu()
Return

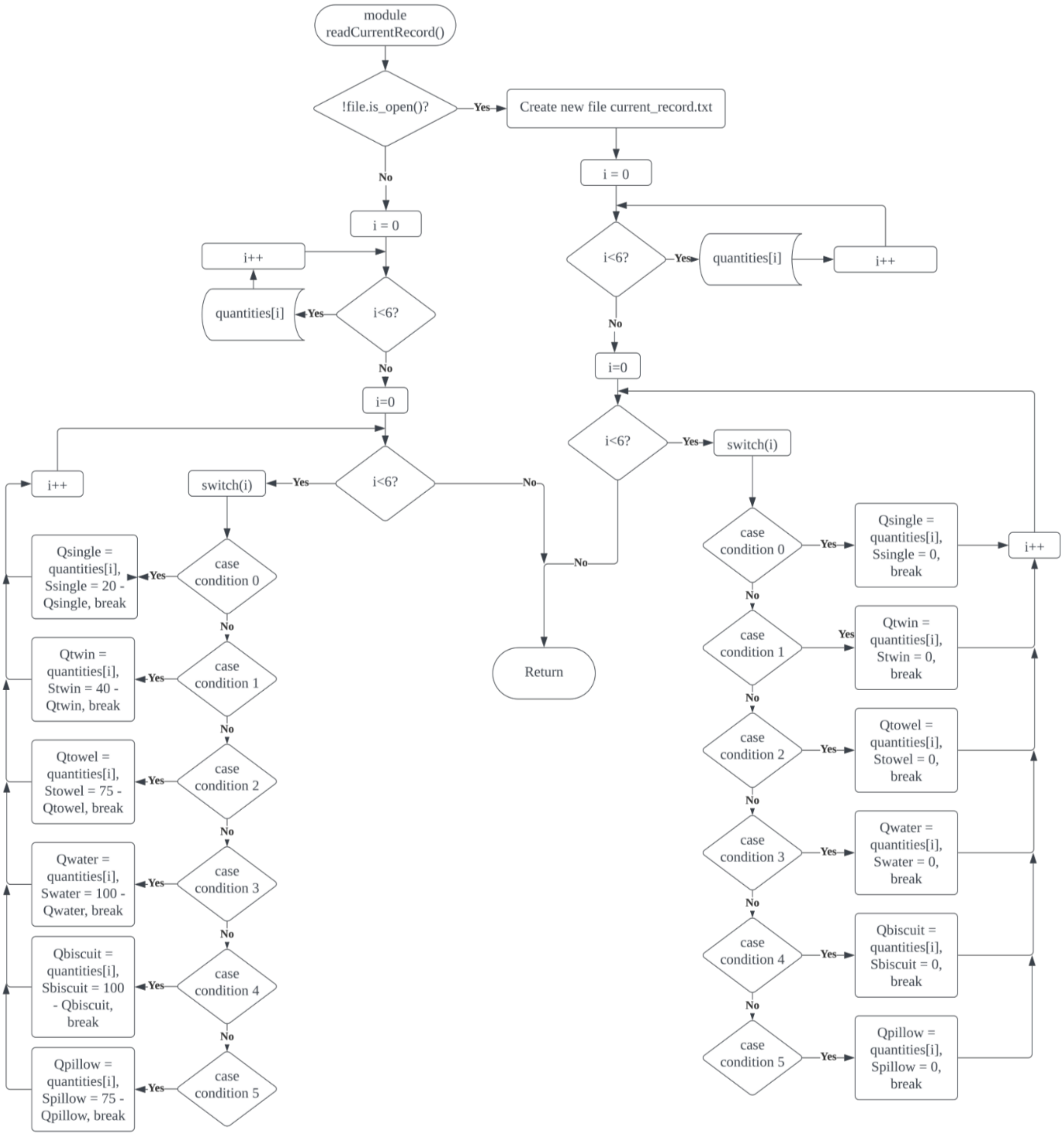

module AdminMenu()
Display "Admin Menu\n\t| Press 1 to view room availability\n\t| Press 2 to view amenities availability\n\t| Press 3 to view sales report\n\t| Press 4 to reset current record\n\t| Press 5 to log out\n\t\t\t Please enter your choice: "
Input choice
switch(choice)
case condition 1
Yes
viewRooms()
No
case condition 2
Yes
viewAmenities()
No
case condition 3
Yes
salesReport()
No
case condition 4
Yes
resetCurrentRecord()
No
break
case condition 5
Yes
Display "Logging out..."
main()
Return
No
default
Yes
Display "Invalid choice. Please select from the options given above"
AdminMenu()

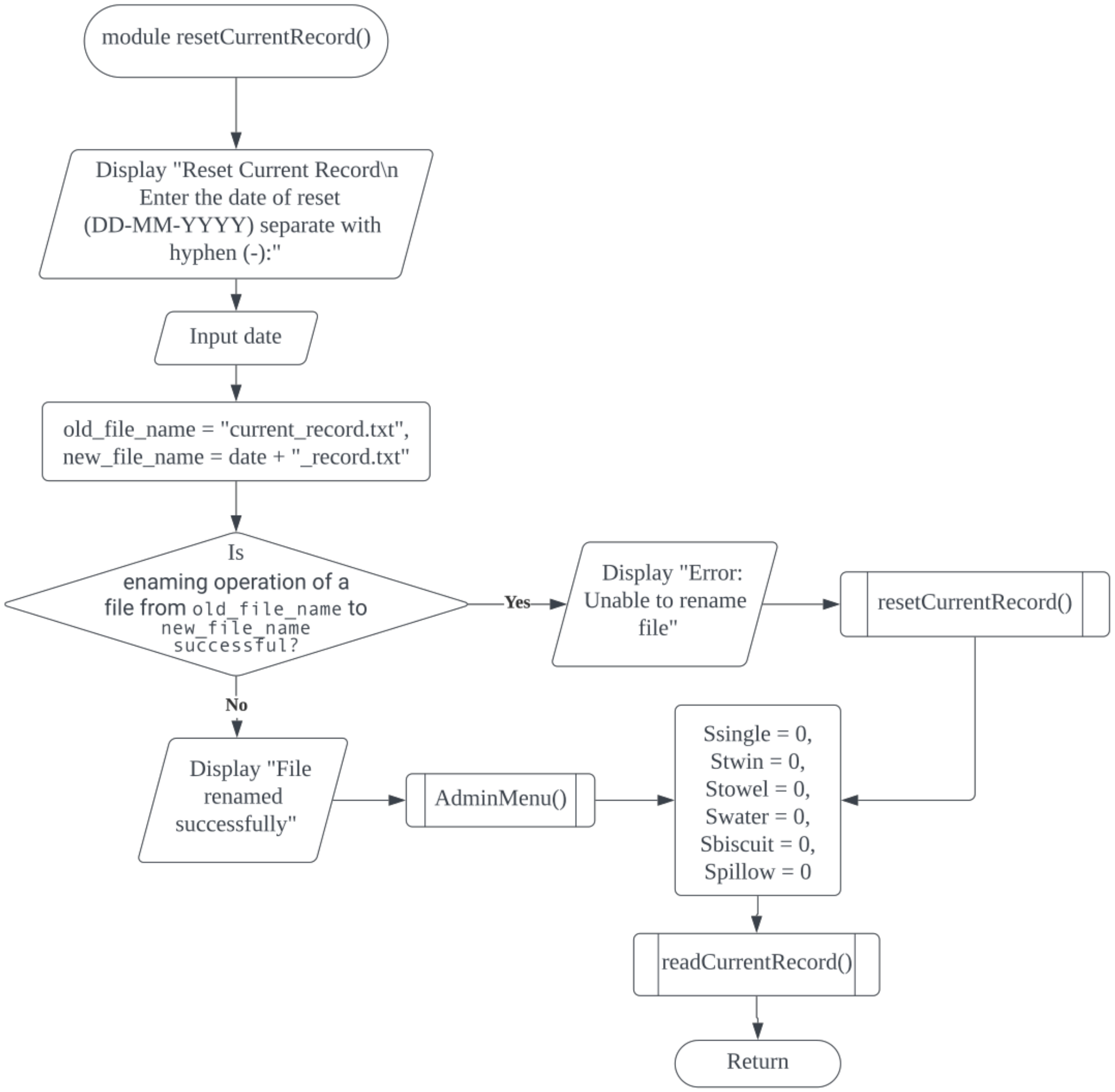

module viewRooms()
Display "Room Availability \n Single Rooms: \n Twin Rooms: \n Do you want to exit? \n1. Yes\n2. No\n Action:"
Input choice
choice == 1?
Yes
Display "Logging out..."
No
AdminMenu()
Return


module salesReport()
Total_single = Ssingle * 65,
Total_twin = Stwin * 100,
Total_towel = Stowel * 0,
Total_water = Swater * 2,
Total_biscuit = Sbiscuit * 6,
Total_pillow = Spillow * 0,
total_sales = Total_single + Total_twin + Total_towel + Total_water + Total_biscuit + Total_pillow
Display "Sales Report \nQuantity \t Total Sales \n Single Rooms \t " ,Ssingle , " \t\t RM",Total_single, "\n Twin Rooms ",Stwin," RM" ,Total_twin, " Towels \t " ,Stowel," RM" ,Total_towel , " \nDrinks ", Swater, " RM" ,Total_water,"\n Snacks" , Sbiscuit ,"RM", Total_biscuit, "\nPillows ",Spillow, " \n RM",Total_pillow ,"\nGrand Total: RM", total_sales, "\n Do you want to exit? \n1. Yes\n2. No \n Action: "
Input choice
choice == 1?
Yes
Display "Logging out..."
No
AdminMenu()
Return


module viewPrices()
Display "Room prices \n Room Type \t Price \n Single \t RM65 \n Twin \t\t RM100 "
Return


module viewAmenities
Display "Amenities Availability\nTowel: " , Qtowel - Stowel,"\n Pillow: ", Qpillow - Spillow,"\n Water: ", Qwater - Swater, "\nBiscuit: ",Qbiscuit - Sbiscuit,"\nDo you want to exit? \n1. Yes\n2. No\nAction: "
Input choice
choice == 1?
Yes
Display "Logging out..."
No
AdminMenu()
Return


module readCurrentRecord()
!file.is_open()?
Yes
Create new file current_record.txt
No
i = 0
i = 0
i++
i<6?
Yes
quantities[i]
i++
quantities[i]
Yes
i<6?
No
No
i=0
i=0
i<6?
Yes
switch(i)
i++
switch(i)
Yes
i<6?
No
No
case condition 0
Yes
Qsingle = quantities[i], Ssingle = 0, break
i++
Qsingle = quantities[i], Ssingle = 20 - Qsingle, break
Yes
case condition 0
No
No
case condition 1
Yes
Qtwin = quantities[i], Stwin = 0, break
Return
Qtwin = quantities[i], Stwin = 40 - Qtwin, break
Yes
case condition 1
No
No
case condition 2
Yes
Qtowel = quantities[i], Stowel = 0, break
Qtowel = quantities[i], Stowel = 75 - Qtowel, break
Yes
case condition 2
No
No
case condition 3
Yes
Qwater = quantities[i], Swater = 0, break
Qwater = quantities[i], Swater = 100 - Qwater, break
Yes
case condition 3
No
No
case condition 4
Yes
Qbiscuit = quantities[i], Sbiscuit = 0, break
Qbiscuit = quantities[i], Sbiscuit = 100 - Qbiscuit, break
Yes
case condition 4
No
No
case condition 5
Yes
Qpillow = quantities[i], Spillow = 0, break
Qpillow = quantities[i], Spillow = 75 - Qpillow, break
Yes
case condition 5


module resetCurrentRecord()
Display "Reset Current Record\n Enter the date of reset (DD-MM-YYYY) separate with hyphen (-):"
Input date
old_file_name = "current_record.txt", new_file_name = date + "_record.txt"
Is enaming operation of a file from old_file_name to new_file_name successful?
Yes
Display "Error: Unable to rename file"
resetCurrentRecord()
No
Display "File renamed successfully"
AdminMenu()
Ssingle = 0, Stwin = 0, Stowel = 0, Swater = 0, Sbiscuit = 0, Spillow = 0
readCurrentRecord()
Return

12.4. ***Source code***

SCAN ME

My GitHub page for this project:

https://github.com/kreee00/hotel_management_system_cpp/